AD-A089 351 STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB F/G 12/2
MINOS/AUGMENTED USER'S MANUAL.(U)
JUN 80 B A MURTAGH, M A SAUNDERS DAAG29-79-C-0110
UNCLASSIFIED SOL-80-14 NL

1 of 1
AD A
089351

END
DATE
FILMED
10-80
DTIC

Systems
Optimization
Laboratory

LEVEL

AD A089351

DTIC
ELECTE
SEP [illegible] 1980

Department of Operations Research
Stanford University
Stanford, CA 94305

2

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

MINOS/AUGMENTED
USER'S MANUAL

by

Bruce A. Murtagh and Michael A. Saunders

TECHNICAL REPORT SOL 80-14
June 1980

Research and reproduction of this report were supported by the Department of Energy Contract DE-AC03-76SF00326, PA No. DE-AT03-76ER72018; National Science Foundation Grants MCS-7926009 and ENG77-06761; the Office of Naval Research Contract N00014-75-C-0267; and the U.S. Army Research Office Contract DAAG29-79-C-0110.

MINOS/AUGMENTED
User's Manual

Bruce A. Murtagh

Department of Industrial Engineering
The University of New South Wales
Kensington, New South Wales, Australia 2033

Michael A. Saunders

Systems Optimization Laboratory
Department of Operations Research
Stanford University
Stanford, CA 94305

ABSTRACT

MINOS/AUGMENTED is a general purpose nonlinear programming system, designed to solve large-scale optimization problems involving sparse linear and nonlinear constraints. Any nonlinear functions appearing in the objective or the constraints must be continuous and smooth. Users specify these functions and their gradients using two Fortran subroutines. The remaining constraint information is specified in standard MPS format, as for regular linear programming models.

MINOS/AUGMENTED (alias MINOS Version 4.0) employs a projected augmented Lagrangian algorithm to solve problems with nonlinear constraints. This involves a sequence of sparse, linearly constrained subproblems, which are solved by a reduced-gradient algorithm as implemented in the earlier version of MINOS.

This manual supplements Report SOL 77-9, the MINOS User's Guide.

CONTENTS

1. INTRODUCTION

1.1 Scope of the Manual

The scope of this manual is restricted to matters additional to those covered in the MINOS User's Guide [2]. We assume that you are either already familiar with that manual, or at least have a copy at hand to refer to.

1.2 Linear Programming

Unless nonlinearities are specified, MINOS/AUGMENTED solves the standard linear programming problem, using a reliable implementation of the revised simplex method. (A sparse *LU* factorization of the basis matrix is computed using the "bump and spike" algorithm of Hellerman and Rarick, and this is updated in a stable manner by the method of Bartels and Golub.)

1.3 Nonlinear Objective

Similarly, unless some nonlinear constraints are specified in the SPECS file, the system will use a reduced-gradient algorithm to solve the linearly constrained nonlinear programming problem, as in the earlier version of MINOS [2],[3].

1.4 Nonlinear Constraints

When nonlinear constraints exist, the optimization procedure used by MINOS/AUGMENTED is one that treats linear constraints and bounds specially, but does not necessarily satisfy the nonlinear constraints until an optimal point is reached. This means that functions involved in the constraints may need to be defined outside the region of interest.

The nature of the solution process itself can be summarized as follows. A sequence of "major iterations" is performed, each one requiring the solution of a linearly constrained subproblem. The subproblems contain the original linear constraints and bounds, as well as linearized versions of the nonlinear constraints.

It is safe to assume that the objective function will never be evaluated at a point x unless that point satisfies the linear constraints and the bounds on the variables.

Similarly, the constraint functions will *almost never* be evaluated unless the linear constraints and bounds are satisfied. The principal exception to this rule is the very first point x_0 (which may optionally be specified by the user). The nonlinear constraint functions will be evaluated at x_0 *regardless of feasibility*.

These matters must be borne in mind during the formulation of a nonlinear program (see below). The main point to remember is that the nonlinear constraints may be violated during the solution process.

1.5 Additional User-supplied Information

Most of the data for a problem is provided by means of the MPS file. This contains linear objective and constraint data in a format that is compatible with existing mathematical programming systems.

If the problem has a nonlinear objective function, the user provides a Fortran subroutine, CALCFG, to compute the function and its gradient.

Similarly, if the problem has any nonlinear constraints, the user provides a Fortran subroutine, CALCON, to compute the nonlinear terms and their gradients.

Input data is processed in the following order:

- The SPECS file
- The MPS file
- A basis file (optional)
- Data read by CALCON on its first entry
- Data read by CALCFG on its first entry
- Data read by CALCFG on its last entry
- Data read by CALCON on its last entry

This order is important if all the data is stored in the same input stream. For large problems the MPS data will usually be in a file of its own. Three types of basis file may be input (and output), and again, any that is used will normally be on a file of its own.

1.6 Problem Formulation

In general, it is worthwhile expending considerable prior analysis to make your constraints as near to linear as possible. Sometimes a simple transformation will suffice. For example, a pipeline optimization problem has pressure drop constraints of the form

$$\frac{K_1}{d_1^{4.814}} + \frac{K_2}{d_2^{4.814}} + \cdots \le P_T^2 - P_0^2$$

where d_i are the design variables (pipe diameters) and the other terms are constant. These constraints are highly nonlinear, but by re-defining the decision variables to be $x_i = 1/d_i^{4.814}$ we can make the constraints linear. Even if the objective function becomes more nonlinear by such a transformation, and this usually happens, the advantages of having linear constraints greatly outweigh this.

Similarly, it is important not to take nonlinearities out of the objective function into the constraints. Thus, we would *not* replace

$$\text{minimize } f^0(x)$$

by

$$\text{minimize } z \quad \text{subject to} \quad f^0(x) - z = 0.$$

Scaling is a very important matter during problem formulation. A general rule is to scale both the data and the variables to be as close to 1.0 as possible. When conflicts arise, one should again sacrifice the objective function in favor of the constraints. Real-world problems tend to have a natural scaling within each constraint, as long as the variables are expressed in consistent physical units. Hence it is often sufficient to apply a scale factor to each row.

Finally, *upper and lower bounds* on the variables (and on the constraints) are extremely useful in confining the region over which optimization has to be performed. If sensible values are known, they should always be used. They are also important for avoiding singularities in the problem functions. For safety when such singularities exist, the initial point x_0 discussed above should lie within the bounds.

1.7 Restrictions

The algorithm used in MINOS/AUGMENTED is designed to find solutions that are *locally optimal*. The nonlinear functions in a problem must be smooth, and their first derivatives must be computable. The functions need not be separable. Integer restrictions cannot be imposed directly.

A certain region is defined by the linear constraints in a problem and by the bounds on the variables. If the nonlinear objective and constraint functions are convex within this region, any optimal solution obtained will be a *global optimum*. Otherwise there may be several local optima, and some of these may not be global. In such cases the chances of finding a global optimum are usually increased by choosing a starting point that is "sufficiently close", but there is no general procedure for determining what "close" means, or for verifying that a given local optimum is indeed global.

MINOS/AUGMENTED uses one large array of main memory for most of its working storage. The length of this array may need to be adjusted to suit a particular problem, but otherwise the implementation places no intrinsic limitation on problem size.

Nevertheless, some a *priori* knowledge of a particular application should indicate whether or not the algorithm is likely to be efficient. Suppose there are m general constraints and $n + m$ variables (including m "slacks"), with upper and lower bounds on all variables. In an optimal solution there will be m "basic" variables and s "superbasic" variables that are strictly between their bounds. (The remaining "nonbasic" variables will be *equal* to one of their bounds.) Ideally s should be small. If it seems likely that s will be larger than about 200, some aggregation or reformulation of the problem should be considered.

Note that s will never be larger than the number of variables that occur nonlinearly in the problem. More importantly, s is often very much less than this upper bound. The question to ask is "How many variables, including slacks, are likely to be equal to one of their bounds in the optimal solution?" Subtracting this number from n will give the required estimate of s. (This value should then be specified by both the `SUPERBASICS LIMIT` and the `HESSIAN DIMENSION` keywords in the SPECS file.)

2. NONLINEAR CONSTRAINTS

2.1 Statement of the Problem

The problem to be solved must be expressed in the following standard form:

$$\text{minimize} \quad f^0(x) + c^T x + d^T y \tag{1}$$

$$\text{subject to} \quad f(x) + A_1 y = b_1, \tag{2}$$

$$A_2 x + A_3 y = b_2, \tag{3}$$

$$l \le \begin{bmatrix} x \\ y \end{bmatrix} \le u, \tag{4}$$

where

$$f(x) = \begin{bmatrix} f^1(x) \\ \vdots \\ f^{m_1}(x) \end{bmatrix}$$

and the functions $f^i(x)$ are smooth and have known gradients. The components of x are called the nonlinear variables, and they must be the first set of unknowns. Similarly, constraints (2) are called the nonlinear constraints and they must appear before the linear constraints (3).

All types of inequality are allowed in the general constraints. Thus, the "$=$" sign in (2) and (3) may mean "$\le$" or "$\ge$" or "free" for individual rows.

Upper and lower bounds (4) may be specified for all variables, and similar bounds (ranges) may be defined for the general constraints.

2.2 Solution Technique

The solution process [4],[5] consists of a sequence of "major iterations." At the start of each major iteration, the nonlinear constraints are linearized at the current point x_k. This just means that $f(x)$ in equation (2) is replaced by the approximation

$$\tilde{f}(x, x_k) = f(x_k) + J(x_k)(x - x_k),$$

which we shall write as

$$\tilde{f} = f_k + J_k(x - x_k). \tag{5}$$

Here, $J(x)$ is the Jacobian matrix whose ij-th element is $\partial f^i(x)/\partial x_j$.

The objective function is also modified, giving the following subproblem:

$$\text{minimize } f^0(x) + c^T x + d^T y - \lambda_k^T (f - \tilde{f}) + \frac{1}{2}\rho (f - \tilde{f})^T (f - \tilde{f}) \tag{6}$$

$$\text{subject to} \quad \tilde{f} + A_1 y = b_1, \tag{7}$$

$$A_2 x + A_3 y = b_2, \tag{8}$$

$$l \le \begin{bmatrix} x \\ y \end{bmatrix} \le u.$$

The objective function (6) is called an *augmented Lagrangian*. The vector λ_k is an estimate of the Lagrange multipliers for the nonlinear constraints, and the term involving ρ is a modified quadratic penalty function.

Using (5), we can see that the linear constraints (7) and (8) take the form

$$\begin{bmatrix} J_k & A_1 \\ A_2 & A_3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b_1 + J_k x_k - f_k \\ b_2 \end{bmatrix}. \tag{9}$$

Since MINOS takes advantage of sparsity within the constraint matrix, it is clear that a sparse Jacobian matrix J_k can be handled efficiently.

2.3 Choice of λ_k

Two choices of λ_k are allowed, according to the `LAGRANGIAN` keyword in the SPECS file. The choice `LAGRANGIAN = NO` sets both $\lambda_k = 0$ and $\rho = 0$, and corresponds to simple sequential linearization of the nonlinear constraints, with no modification to the original objective function. This choice is not usually recommended, since convergence cannot be guaranteed in general.

The preferred option is `LAGRANGIAN = YES`. In this case λ_k will be set to the first m_1 "simplex multipliers" from the previous subproblem (except λ_0 is zero, or may be specified by the user). The vectors λ_k should converge to the Lagrange multipliers for the original nonlinear constraints. The final λ_k will appear in the `ROWS` section of the printed solution under the heading `DUAL ACTIVITY`.

2.4 Choice of ρ

When `LAGRANGIAN = YES`, the penalty parameter ρ may also be specified, and this may be essential to obtain convergence. Some advice for setting ρ is given under `PENALTY PARAMETER` in section 4.1. In many cases, $\rho = 0$ will give the most rapid rate of convergence, but for highly nonlinear problems a positive value is recommended.

2.5 Convergence Conditions

Broadly speaking, if x_k is an optimal solution to the k-th subproblem, and if it satisfies the nonlinear constraints sufficiently well, then x_{k+1} (the solution to the next subproblem) will probably be an optimal solution to the original nonlinear program.

More precisely, let (x_k, λ_k) be the final solution and multiplier estimates that result from solving the k-th subproblem. The next subproblem is defined in terms of x_k and λ_k, and will terminate at some point (x_{k+1}, λ_{k+1}). Convergence is assumed to have occurred if the following conditions are true:

x_k is an optimal solution to its subproblem;
x_k satisfies the nonlinear constraints to within a specified tolerance ϵ_r;
λ_k is not substantially different from λ_{k-1};
x_{k+1} is an optimal solution to its subproblem;
a basis change did not occur during solution of subproblem $k + 1$;
the reduced gradient did not increase significantly during solution of that subproblem.

If all these conditions hold, (x_{k+1}, λ_{k+1}) will be accepted as an optimal solution to the original problem.

The point to remember here is that x_k is checked for feasibility and then the final point x_{k+1} is checked for optimality. Normally, very few minor iterations will occur on the last subproblem (ideally none). Hence the last two subproblem solutions x_k and x_{k+1} will be virtually identical, and therefore the tests for feasibility and optimality will have been applied to essentially the same point.

3. FUNCTION ROUTINES

3.1 Subroutine CALCFG

This subroutine is provided by the user to calculate the objective function $f^0(x)$ and its gradient $g^0(x)$. It remains essentially the same as in the earlier version of MINOS, but an option now exists for allowing MINOS to calculate some of the components of $g^0(x)$ by finite differences.

CALCFG is not needed if the objective function is entirely linear.

Specification:

```
SUBROUTINE CALCFG( MODE, N, X, F, G, NSTATE, NPROB )
IMPLICIT   REAL*8(A-H,O-Z)
DIMENSION  X(N), G(N)
```

(The IMPLICIT statement should not be used on machines for which single-precision floating-point is adequate; e.g. Burroughs and CDC.)

Parameters:

MODE (Input) If DERIVATIVE LEVEL=1 or 3, the value of MODE can be ignored; it will always be 2. You have undertaken to compute all gradient components. (This is highly recommended.)

If DERIVATIVE LEVEL=0 or 2, there are two relevant input values, and you must test MODE to decide what to do:

If MODE=2, compute the objective value F, and as many components of G as you can.

If MODE=0, compute the objective value F, but do not alter *any* of the components of G.

(Output) If for some reason you wish to terminate solution of the current problem, set MODE to a negative value, e.g. -1.

N (Input) The number of variables involved in $f^0(x)$. These must be the first N variables in the problem.

X(N) (Input) An array of dimension N containing the current values of the nonlinear variables x.

F (Output) The computed value of $f^0(x)$.

G(N) (Output) The computed gradient vector $g^0(x)$. For each relevant j, G(j) should contain the partial derivative $\partial f^0/\partial x_j$ (except if MODE=0 — see above).

NSTATE (Input) If NSTATE=0, there is nothing special about the current call to CALCFG.

If NSTATE=1, this is the first call to CALCFG. Some data may need to be input or computed and saved in local or COMMON storage, for use in the present and subsequent calls to CALCFG.

If NSTATE=2, the current solution in X has been determined to be optimal. You may wish to perform some additional computation on this solution. (This case will not arise unless the CALL keyword is used in the SPECS file.)

NPROB (Input) An integer that can be set by a card of the form PROBLEM NUMBER n in the SPECS file.

3.2 Subroutine CALCON

This subroutine is provided by the user to compute the nonlinear constraint functions $f(x)$ and the corresponding Jacobian matrix $J(x)$. Recall that the j-th column of $J(x)$ is defined to be $\partial f/\partial x_j$.

CALCON may be coded in two different ways, depending on the method used for storing the Jacobian.

JACOBIAN = DENSE

Specification:

```
SUBROUTINE CALCON( MODE, M, N, NJAC, X, F, G, NSTATE, NPROB )
IMPLICIT   REAL*8(A-H,O-Z)
DIMENSION  X(N), F(M), G(M,N)
```

Parameters:

MODE (Input) Options not implemented.

(Output) If for some reason you wish to terminate solution of the current problem, set MODE to a negative value, e.g. -1.

M (Input) The number of nonlinear constraints, not counting the objective function. These must be the first M constraints in the problem.

N (Input) The number of variables involved in $f(x)$. These must be the first **N** variables in the problem.

NJAC (Input) The value **M*N**. (This may or may not be useful.)

X(N) (Input) An array of dimension **N** containing the current values of the nonlinear variables x.

F(M) (Output) The computed value of the constraint vector $f(x)$.

G(M,N) (Output) The computed Jacobian matrix $J(x)$. The j-th column of $J(x)$ should be stored in the j-th column of the 2-dimensional array **G**. Equivalently, the gradient of the i-th constraint should be stored in the i-th row of **G**. Any constant elements that were specified in the MPS file need not be reset here. This includes elements that are identically zero.

Caution: Even if an element J_{ij} is constant (and nonzero), it still enters into the calculation of the i-th constraint. In fact, the value G(i,j)*X(j) should be added to F(i).

NSTATE (Input) If **NSTATE=0**, there is nothing special about the current call to **CALCON**.

If **NSTATE=1**, this is the first call to **CALCON**. Some data may need to be input or computed and saved in local or **COMMON** storage, for use in the present and subsequent calls to **CALCON**.

If **NSTATE=2**, the current solution in **X** has been determined to be optimal. You may wish to perform some additional computation on this solution. (As with subroutine **CALCFG**, this case will not arise unless the **CALL** keyword appears in the SPECS file.)

NPROB (Input) An integer that can be set by a card of the form **PROBLEM NUMBER** n in the SPECS file.

JACOBIAN = SPARSE

Specification:

```
SUBROUTINE CALCON( MODE, M, N, NJAC, X, F, G, NSTATE, NPROB )
IMPLICIT   REAL*8(A-H,O-Z)
DIMENSION  X(N), F(M), G(NJAC)
```

This is the same as for **JACOBIAN = DENSE**, except for the declaration of **G(NJAC)**.

Parameters:

NJAC (Input) The number of nonzero elements in the Jacobian matrix $J(x)$. This is exactly the number of entries in the MPS file that referred to nonlinear rows and nonlinear Jacobian columns.

Usually `NJAC` will be less than `M*N`. The actual value of `NJAC` may not be of any use when coding `CALCON`, but in all cases, any expression involving `G(i)` should have the subscript i between 1 and `NJAC`.

G(NJAC) (Output) The computed elements of the Jacobian matrix. These elements must be stored into `G` in exactly the same position as implied by the MPS file. There is no internal check for consistency (except indirectly via the `VERIFY CONSTRAINT GRADIENTS` option), so great care is essential.

If any element of the Jacobian is constant, and if the correct value was entered in the MPS file, the corresponding element `G(i)` need not be reassigned. (However, one of the elements of `F` requires a term of the form `G(i)*X(j)`.)

The other parameters are the same as for `JACOBIAN = DENSE`.

3.3 Reserved COMMON Blocks

When the above subroutines are coded, certain care must be exercised to avoid conflict with the coding of MINOS. In particular, the following labeled `COMMON` blocks are used internally by MINOS:

ALCOM1	DJCOM	INVCOM	PARMCM
ALCOM2	EPSCOM	IOCOMM	PRCCOM
BGCOM	FILES	ITNLOG	PRCCM2
CGCOM	FREQS	ITNLG2	RGTCLS
CONVCM	FXCOM	LPCOM	SOLNCM
CORE	FXCOM2	LUFILE	TOLS
CYCLCM	INTCOM	MPSCOM	WORDSZ

These `COMMON` blocks must not be overwritten.

In general we recommend that blank `COMMON` should not be overwritten either. This is because MINOS needs one large array for workspace, and in some installations it may be convenient to store this array in blank `COMMON` (e.g. to allow core to be allocated at run-time).

Note that on some computer systems (e.g. the Burroughs B6700), local data created by a subroutine may need to be saved in a `COMMON` block to ensure that the data won't "disappear" on exit from the subroutine. In this case it is easy to avoid conflict with the reserved names.

Occasionally it may be convenient to *use* data that is stored in the reserved `COMMON` blocks. In particular, the declaration

```
COMMON    /IOCOMM/ IREAD,IPRINT
```

provides access to two integer variables that define the standard Fortran reader and printer files. When MINOS was originally compiled on your computer system, `IREAD` and `IPRINT` will have been assigned the appropriate values (typically 5 and 6). These may be used in I/O statements if you wish; an example is given in section 7.2.

3.4 Reserved Subroutine Names

MINOS/AUGMENTED contains the subroutines listed below. These names must not be used for any auxiliary user routines.

ADDCOL	DELCOL	LOADB	R1ADD
ALAUX	DOT	LOADN	R1MOD
BTRANL	DRIVER	LPITN	R1PROD
BTRANU	DUMPN	LSOUTC	R1SUB
BUMPS	FACTOR	MINOS	SAVEB
CALCFG	FGMOD	MKLIST	SEARCH
CALCG	FORMC	MODLU	SETJAC
CALCON	FTRANL	MPS	SETPI
CG	FTRANU	MPSIN	SETX
CHKDIR	FUNGRD	NMSRCH	SOLN
CHKGRD	FUNJAC	PACKLU	SOLPRT
CHKJAC	GETGRD	PRICE	SPECS
CHUZQ	GETPTC	PRTJAC	SPECS2
CHUZR	GO	PUNCH	STATE
COMDFP	HASH	P3	TRNSVL
COPYA	INITLZ	P4	UNPACK
COPYD	INSERT	RESETR	
COPYH	INVERT	RGITN	
CRASH	ITEROP	RTRSOL	

In addition,

GETCOR

is used in the Burroughs version of MINOS, and

MATMOD MKCOL MODBND MODELM

are the subroutines defined in reference [6].

4. THE SPECS FILE

The SPECS file is supplied by the user; it contains a list of keywords and values to define various run-time parameters. The following keywords apply specifically to problems containing nonlinear constraints:

```
COMPLETION   PARTIAL or FULL
JACOBIAN     DENSE or SPARSE
LAGRANGIAN   YES or NO
MAJOR ITERATIONS
MINOR ITERATIONS
NONLINEAR CONSTRAINTS
NONLINEAR OBJECTIVE VARIABLES
NONLINEAR JACOBIAN VARIABLES
PENALTY PARAMETER
PRINT LEVEL
RADIUS OF CONVERGENCE
ROW TOLERANCE
```

The next section describes the way these keywords should be used. Also described are the following:

```
BACKUP BASIS FILE
CALL FUNCTION ROUTINES WHEN OPTIMAL
CRASH OPTION
CYCLE LIMIT
DERIVATIVE LEVEL
DIFFERENCE INTERVAL
MULTIPLE PRICE
PHANTOM COLUMNS
PIVOT TOLERANCE
PRINT SPIKE PATTERN
START and STOP gradient verification
SUPPRESS PARAMETERS
VERIFY GRADIENTS
```

Some of these keywords are new. The remainder were recognized by the earlier version of MINOS but have had their meaning expanded.

Remember that the first three characters of a keyword are always significant, and in some cases the first four characters of the next word are also significant. For example, in the SPECS card

```
NONLINEAR CONSTRAINTS   100
```

both `NON` and `CONS` are significant.

4.1 Keywords

BACKUP BASIS FILE k (default $k = 0$)

This is intended as a safeguard against losing the results of a long run. Suppose that a NEW BASIS FILE is being saved every 100 iterations, and that MINOS is about to save such a basis at iteration 2000. It is conceivable that the run may time-out during the next few milliseconds (i.e. in the middle of the save), or the host computer could unexpectedly crash. In this case the basis file will be corrupted and the run will have been essentially wasted.

To eliminate this risk, both a NEW BASIS FILE and a BACKUP BASIS FILE may be specified. The following would be suitable for the above example:

```
OLD BASIS FILE      10        (or 0)
NEW BASIS FILE      10
BACKUP BASIS FILE   11
SAVE FREQUENCY     100
```

The current basis will then be saved every 100 iterations, first on file 10 and then immediately on file 11. If the run is interrupted at iteration 2000 during the save on file 10, there will still be a useable basis on file 11 (corresponding to iteration 1900).

Note that a NEW BASIS will be saved at the end of a run if it terminates normally, but there is no need for a further BACKUP BASIS. In the above example, if an optimum solution is found at iteration 2050 (or if the iteration limit is 2050), the final basis on file 10 will correspond to iteration 2050, but the last basis saved on file 11 will be the one for iteration 2000.

CALL FUNCTION ROUTINES WHEN OPTIMAL

This requests a final call to subroutine CALCFG and/or subroutine CALCON (in that order) when an optimal solution is reached. This is the means by which the parameter value NSTATE=2 is obtained. See the specification of CALCFG and CALCON for further details.

COMPLETION PARTIAL
COMPLETION FULL (default)

This determines whether subproblems should be solved accurately (full completion), or whether each one should be terminated somewhat earlier (partial completion). MINOS effects this by using two sets of convergence tolerances for the subproblems.

Use of partial completion may reduce the work during early major iterations (unless the MINOR ITERATIONS limit is active). The optimal set of basic and superbasic variables will probably be determined for any given subproblem, but the reduced gradient may be larger than it would have been with full completion.

An automatic switch to full completion occurs when it appears that the sequence of major iterations is converging. The switch is made when the constraint error is reduced below $100\epsilon_r$ (where ϵ_r is specified by the `ROW TOLERANCE` keyword).

Full completion tends to give better Lagrange-multiplier estimates and may lead to fewer major iterations.

`CRASH OPTION` k (default $k = 1$)

If a starting basis is not specified, a triangular basis will be selected from certain columns of the constraint matrix A, depending on the value of k.

k	*Meaning*
0	The all-slack basis is set up.
1	All columns of A are considered.
2	Only the columns of A corresponding to the linear variables y will be considered. Linear programming will then be used to optimize y as much as possible, before the nonlinear variables x are altered from their initial values. This is an important option.
3	Nonlinear objective variables will be excluded from the initial basis.
4	Nonlinear Jacobian variables will be excluded from the initial basis.

In all cases, CRASH will refrain from selecting variables that were made superbasic by means of an `FX` indicator in the `INITIAL` bounds set.

`CYCLE LIMIT` l
`CYCLE PRINT` p
`CYCLE TOLERANCE` t

These keywords are documented elsewhere (Preckel [6]). They refer to a facility for constructing and solving a sequence of related problems. Modules are provided for modifying the constraint data internally, using information obtained from the previous problem.

`DERIVATIVE LEVEL` d (default $d = 3$)

This specifies which nonlinear function gradients are known analytically and will be supplied to MINOS by the user subroutines `CALCFG` and `CALCON`. The values planned for implementation are as follows.

d	*Meaning*
3	All objective and constraint gradients are known.
2	All constraint gradients are known, but some or all of the objective gradients are unknown.
1	All objective gradients are known, but some or all of the constraint gradients are unknown.
0	Some of the objective gradients are unknown and some of the constraint gradients are unknown.

The value $d = 3$ should be used whenever possible. It is the most reliable and will usually be the most efficient.

If $d = 2$, MINOS will estimate the missing objective gradients by finite differences. This may be convenient if most of the gradient elements are known and are computed by subroutine `CALCFG`. However, a special call to `CALCFG` is required for each missing element (this could be expensive), and in general the option is not entirely reliable. If the nonlinear variables are not well scaled, it may be necessary to specify a nonstandard `DIFFERENCE INTERVAL` (see below).

Note: In the present implementation, all constraint gradients must be provided by subroutine `CALCON`. Hence, the options $d = 0$ and $d = 1$ must not be used unless the constraints are entirely linear.

`DIFFERENCE INTERVAL` h (default $h = 2\sqrt{\epsilon}$)

This may be used to alter the finite-difference interval h that is used in the following circumstances:

1. In the initial ("cheap") phase of verifying the objective gradients.
2. For verifying the constraint gradients.
3. For estimating missing objective gradient elements.

In the last two cases, a derivative with respect to x_j is estimated by perturbing that component of x to the value $x_j + h(1 + |x_j|)$, and then evaluating $f(x)$ or $f^0(x)$ at the perturbed point. Judicious alteration of h may sometimes lead to greater accuracy. The machine precision, ϵ, should always be borne in mind.

JACOBIAN DENSE
JACOBIAN SPARSE (default)

This determines the manner in which the constraint gradients are evaluated and stored. It affects the MPS file and subroutine CALCON.

The DENSE option is convenient if there are not too many nonlinear constraints or variables. It requires storage for three dense matrices of order $m_1 \times n_1$. (One of these is J_k which forms part of the constraint matrix in equation (9). If J_k is large and dense, the basis factorization may contain an unnecessarily large "bump" and a large number of "spikes".)

When DENSE is specified, the MPS file may contain any number of Jacobian entries. Usually this means no entries at all, or else just ones that remain constant for all values of the nonlinear variables.

For efficiency, the SPARSE option is preferable in all nontrivial cases. The MPS file must then specify the position of all nonzero Jacobian elements. See section 5.2 for details.

LAGRANGIAN YES (default)
LAGRANGIAN NO

This determines the form of the objective function used for the linearized subproblems. The default value YES is highly recommended. The PENALTY PARAMETER value is then also relevant.

If NO is specified, subroutine CALCON will be called only once per major iteration. Hence this option may be useful if the nonlinear constraint functions are very expensive to evaluate. However, in general there is a great risk that convergence may not occur.

MAJOR ITERATIONS k (default $k = 20$)

This is the maximum number of major iterations allowed. It is intended to guard against an excessive number of linearizations of the constraints, since in some cases the sequence of major iterations may not converge.

For preliminary runs on a new problem, a fairly low MAJOR ITERATIONS limit should be specified (e.g. 10 or 20). See the advice given under PENALTY PARAMETER.

MINOR ITERATIONS k (default $k = 40$)

This is the maximum number of iterations allowed between successive linearizations of the nonlinear constraints, not counting infeasible iterations. A moderate value (e.g. $10 \le k \le 50$) prevents excessive effort being expended on early major iterations, but allows later subproblems to be solved to completion.

In general it is unsafe to specify a value as small as $k = 1$ or 2. (Even when an optimal solution has been reached, a few minor iterations may be needed for the corresponding subproblem to be recognized as optimal.)

Note that an independent limit on total iterations should be specified by the `ITERATIONS` keyword as usual. If the problem is linearly constrained, this is the only limit (i.e. the `MINOR ITERATIONS` keyword is ignored).

`MULTIPLE PRICE` k (default $k = 0$)

This option should be considered whenever an initial point is not specified. If the default value of zero is used, only one variable will be selected by each pricing operation to become superbasic. Hence in general, if few or no values are specified in the `INITIAL` bounds set, or if an `OLD BASIS FILE` contains very few superbasics, `MULTIPLE PRICE 10` or `20` may be beneficial (assuming the problem is nonlinear enough to have a large number of superbasic variables at its solution).

A full description of `MULTIPLE PRICE` is given in the MINOS User's Guide.

`NONLINEAR CONSTRAINTS` m_1 (default $m_1 = 0$)

`NONLINEAR VARIABLES` n_1 (default $n_1 = 0$)

`NONLINEAR OBJECTIVE VARIABLES` n_1' (default $n_1' = 0$)

`NONLINEAR JACOBIAN VARIABLES` n_1'' (default $n_1'' = 0$)

These keywords define the parameters `M` and `N` in subroutines `CALCFG` and `CALCON`. For example, `M` in `CALCON` will take the value m_1, if $m_1 > 0$.

If the objective function and the constraints involve the same set of nonlinear variables x, then `NONLINEAR VARIABLES` n_1 is the simplest way to set `N` to be the same value for both subroutines. Otherwise, the `NONLINEAR OBJECTIVE` and `NONLINEAR JACOBIAN` keywords should be used to specify n_1' and n_1'' separately.

Remember that the nonlinear constraints and variables must always be the first ones in the problem. It is usually best to place Jacobian variables before objective variables, so that $n_1'' \le n_1'$ (unless $n_1' = 0$). This affects the way the function subroutines should be programmed, and the order in which variables should be placed in the COLUMNS section of the MPS file.

`PENALTY PARAMETER` ρ (default $\rho = 100.0/m_1$)

This is the value of ρ in the modified augmented Lagrangian (equation (8) in section 2.2). It is used only if `LAGRANGIAN YES` is specified.

For early runs on a problem with unknown characteristics, something like the default value should be specified. In general, a positive value of ρ may be necessary to ensure convergence, but on the other hand, if the value is too large, the rate of convergence may be slow.

If the objective function and the constraints are known to be convex, a zero penalty is best (specify `PENALTY PARAMETER 0.0`). This value may also be satisfactory in the non-convex case, if the functions are not highly nonlinear.

In general, if several related problems are to be solved, the following strategy for setting the `PENALTY PARAMETER` may be useful:

1. Initially, use a moderate value of ρ, such as the default, and a reasonably low `MAJOR ITERATIONS` and/or (total) `ITERATIONS` limit.
2. If successive major iterations appear to be terminating with radically different solutions, the penalty parameter should be increased.
3. If there appears to be little progress between major iterations, the penalty parameter could be reduced.

`PHANTOM COLUMNS` c
`PHANTOM ELEMENTS` e

See Preckel [6].

`PIVOT TOLERANCE` t (default $t = \sqrt{\epsilon}$)

This allows the pivot tolerance to be altered if necessary. (The tolerance is used to prevent columns entering the basis if they would cause the basis to become almost singular.) The default value of t is the square root of the machine precision (roughly 10^{-8} for double precision on IBM systems). This should be satisfactory in most circumstances.

`PRINT LEVEL` p (default $p = 1$)

This varies the amount of information that will be output to the printer file. It is independent of the `LOG FREQUENCY`. Typical values are

`PRINT LEVEL 1`

which gives normal output for linear and nonlinear problems, and

`PRINT LEVEL 11`

which in addition gives the values of the nonlinear variables x_k at the start of each major iteration, for problems with nonlinear constraints.

In general, the value being specified is best thought of as a binary number of the form

`PRINT LEVEL JFLXI`

where each letter stands for a digit that is either 0 or 1. The quantities referred to are:

I INVERT statistics, i.e. information relating to the basis matrix whenever it is refactorized.

X x_k, the nonlinear variables involved in the objective function or the constraints.

L λ_k, the Lagrange-multiplier estimates for the nonlinear constraints. (Suppressed if the option `LAGRANGIAN NO` is specified, since $\lambda_k = 0$ then.)

F $f(x_k)$, the values of the nonlinear constraint functions.

J $J(x_k)$, the Jacobian matrix.

To obtain output of any item, set the corresponding digit to 1, otherwise to 0.

If J=1, the Jacobian matrix will be output column-wise at the start of each major iteration. Column j will be preceded by the value of the corresponding variable x_j and a key to indicate whether the variable is basic, superbasic or nonbasic. (Hence if J=1, there is no reason to specify X=1 unless the objective contains more nonlinear variables than the Jacobian.) A typical line of output is

```
3  1.250000D+01 BS      1  1.00000E+00      4  2.00000E+00
```

which would mean that x_3 is basic at value 12.5, and the third column of the Jacobian has elements of 1.0 and 2.0 in rows 1 and 4.

`PRINT LEVEL 0` may be used to suppress most output, including page ejects between major iterations. (Error messages will not be suppressed.) This print level should be used only for production runs on well-understood models. A high `LOG FREQUENCY` may also be appropriate for such cases, e.g. 100 or 1000. (For convenience, `LOG FREQUENCY 0` may be used as shorthand for `LOG FREQUENCY 99999`.)

PRINT SPIKES

This invokes an option for displaying the bump and spike structure of the basis matrix each time it is refactorized.

RADIUS OF CONVERGENCE r (default $r = 0.01$)

This determines when the penalty parameter ρ will be reset to zero (if initialized to a positive value). Both the nonlinear constraint error (see *ROWERR* below) and the relative change in consecutive Lagrange multipler estimates must be less than r at the start of a major iteration before ρ is set to zero. Thereafter the sequence of major iterations should converge quadratically to an optimum.

ROW TOLERANCE ϵ_r (default $\epsilon_r = $ 1.0E-6)

This specifies how accurately you want the nonlinear constraints to be satisfied. (Both ROW and TOLE are significant on this data card.) The default value of 1.0E-6 is usually appropriate, since the MPS file usually contains data to about that accuracy.

Let $ROWERR$ be defined as the maximum component of the residual vector $f(x) + A_1 y - b_1$, normalized by the size of the solution. Thus,

$$ROWERR = \|f(x) + A_1 y - b_1\|_\infty / \|(x, y)\|_\infty.$$

The solution (x, y) is regarded as acceptably feasible if $ROWERR \leq \epsilon_r$.

If some of the data in your problem is known to be of low accuracy, a larger ROW TOLERANCE may be appropriate. Bear in mind, however, that non-convex problems may need a nonzero PENALTY PARAMETER ρ, and that ρ is automatically reset to zero if $ROWERR \leq 100\epsilon_r$).

START OBJECTIVE CHECK AT VARIABLE k
START CONSTRAINT CHECK AT VARIABLE k

STOP OBJECTIVE CHECK AT VARIABLE l
STOP CONSTRAINT CHECK AT VARIABLE l

These keywords may be used to abbreviate the verification of gradient elements computed by subroutines CALCFG and CALCON. For example:

1. If the first 100 objective gradients appeared to be correct in an earlier run, and if you have just found a bug in CALCFG that ought to fix up the 101-th component, then you might as well specify

   START OBJECTIVE VERIFICATION AT VARIABLE 101 .

   Similarly for columns of the Jacobian matrix.

2. If the first 100 variables occur nonlinearly in the constraints, and if the next 50 variables are nonlinear only in the objective, then CALCFG must set the first 100 components of G(*) to zero, but these hardly need to be verified. The above data card would again be appropriate.

For a normal verification (at the first feasible point), these keywords are effective only if a positive VERIFY LEVEL is specified. The default values are $k = 1$ and $l = n_1$, the appropriate number of nonlinear variables.

For an emergency verification (at the end of a run in which the linesearch procedure appears to have failed), all objective and constraint gradients will be checked, unless a negative VERIFY LEVEL was specified. An exception is if the "cheap" objective check proves to be satisfactory; in this case the specified k and l will be used for checking individual objective gradients.

TARGET OBJECTIVE VALUE t

This option is no longer supported.

SUPPRESS PARAMETERS

Normally MINOS prints the SPECS file as it is being read, and then prints a complete list of the available keywords and their final values. The **SUPPRESS PARAMETERS** option tells MINOS not to print the full list. (Both **SUP** and **PARA** are significant.)

VERIFY OBJECTIVE GRADIENTS
VERIFY LEVEL 1

VERIFY CONSTRAINT GRADIENTS
VERIFY LEVEL 2

VERIFY YES
VERIFY GRADIENTS
VERIFY LEVEL 3

VERIFY NO
VERIFY LEVEL 0 (default)

VERIFY LEVEL -1

The **VERIFY** keyword refers to a finite-difference check on the computed gradient components in the objective function or the nonlinear constraints. The various options should be self-explanatory. For example, the nonlinear objective gradients (if any) will be verified if either **VERIFY OBJECTIVE** or **VERIFY LEVEL 1** is specified. Similarly, both the objective and the constraint gradients will be verified if **VERIFY YES** or **VERIFY LEVEL 3** or just **VERIFY** is specified.

Gradients will be verified at the first point reached that satisfies the linear constraints and the upper and lower bounds. The current linearization of the nonlinear constraints must also be satisfied. Unfortunately, if the programmed gradients are seriously incorrect, there may not be any point at all that satisfies the resulting (incorrect) linearized constraints. In this case an emergency gradient check is performed before MINOS terminates the current problem. If the nonlinear functions are not well defined at the final (infeasible) point, a fatal error may result.

An emergency gradient check will also occur if MINOS is about to terminate because of a linesearch *failure*.

If you do not want an emergency check in either of these situations, you should specify `VERIFY LEVEL -1` .

Verification of the objective gradient occurs in two stages. An inexpensive test on all components is first performed, using two calls to subroutine `CALCFG`. A more reliable test then occurs on individual gradient components, within the ranges specified by the `START` and `STOP` keywords. A key of the form "`OK`" or "`BAD`" indicates whether or not each component appears to be correct.

5. THE MPS FILE

This file specifies most of the constraint data for a particular problem, in the so-called MPS format common to commercial mathematical programming systems. A commercial matrix generator may be used to construct the file, whether or not there are any nonlinear constraints.

5.1 The ROWS Section

The names of the nonlinear constraints must be listed *first* in the ROWS section, and their order must be consistent with the computation of the components of $f(x)$ and $J(x)$ in subroutine CALCON.

Note that the objective function is not included in this list. If the objective contains some linear terms ($c^T x + d^T y$ in equation (1)), then c and d should be specified in an objective row, and the name of this row should appear somewhere *after* the list of nonlinear row names. For simplicity we suggest that objective rows be listed last in the ROWS section.

If the objective function is nonlinear and defined wholly by subroutine CALCFG, there need not be any objective row in the MPS file.

5.2 The COLUMNS Section

Recall that the constraint matrix is of the form

$$\begin{bmatrix} J_k & A_1 \\ A_2 & A_3 \end{bmatrix}$$

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 subroutines CALCFG and CALCON.

Similarly, entries belonging to J_k must appear in an order that is consistent with their calculation in subroutine CALCON (as stored in the parameter G).

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

$$\begin{bmatrix} J_k \\ A_2 \end{bmatrix} = \begin{bmatrix} j_1 \; j_2 \cdots j_{n_1} \\ a_1 \; a_2 \ldots a_{n_1} \end{bmatrix},$$

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. The corresponding row names will need be moved towards the top of the ROWS section, but this is more easily accomplished.)

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 will be reset by subroutine `CALCON`, but they serve to identify the location of the elements.

In either case (`JACOBIAN = DENSE` or `SPARSE`), if some of the Jacobian elements are constant, their correct values may be specified in the COLUMNS section and then they need not be reset by subroutine `CALCON`. This includes values that are identically zero — such elements do not have to be specified anywhere (neither in the MPS file nor in `CALCON`). In other words, Jacobian elements are assumed to be zero unless specified otherwise.

Note that `X` may not be the same length in subroutines `CALCFG` and `CALCON` (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 of course be the same as 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 this case we recommend that these objective variables be placed *after* the Jacobian variables in the COLUMNS section, since this will keep the Jacobian as small as possible.

5.3 The RHS Section

The vectors b_1 and b_2 in (7) and (8) may be regarded as a normal right-hand-side vector b. Only the nonzero coefficients of b need to be specified. They may appear in any order.

The MPS file may contain several RHS vectors. A particular one may be specified in the SPECS file. Otherwise the first RHS will be used; in this case, if the name field is blank, the vector will be given the name `RHS`.

If $b = 0$, a card with `RHS` in columns 1–3 must appear as usual, but no rhs coefficients need follow. A dummy vector will be constructed, and again it will be given the name `RHS`.

Specifying λ_0.

The name `LAGRANGE` is reserved for a special RHS vector whose entries will be used to define Lagrange-multiplier estimates for the nonlinear constraints. These will be used as λ_0 in the objective function for the first major iteration. This facility should be used whenever possible, since the accuracy of the multiplier estimates can often have a significant effect on the rate of convergence of the optimization process. For any given constraint, if you happen to know that the optimal multiplier is going to be negative (say), an entry of `-1.0` will probably be better than the default value of zero.

Entries in the `LAGRANGE` RHS may be interspersed with entries for the true RHS. Any appearing in linear rows will be counted but otherwise ignored.

Note that `LAGRANGE` estimates will be used to define λ_0 even if a starting basis is provided. (This is in contrast to entries in an `INITIAL` bounds set (section 5.5), which will be used only for a cold start.) It is therefore important to revise the MPS file whenever new information comes to hand, e.g. from the solution obtained at the end of an earlier run.

5.4 The RANGES Section

Nonlinear rows may be ranged in the same manner as linear rows. Since the method for specifying ranges is difficult to remember, the following example will be useful. If the first constraint is called `CON1` and is of the form

$$l_1 \le f^1(x) + a_1^T y \le u_1,$$

one way of specifying it in the MPS file is as follows:

```
ROWS
 L  CON1
    :
    :
RHS
    RHS1      CON1      u1
    :
    :
RANGES
    RNG1      CON1      u1 - l1
    :
    :
```

Note that ranges typically make a problem *easier* to solve, since they confine the solution to a smaller region. Strangely enough, they are not often used by linear programmers even when reasonable values are known in advance. For nonlinear programs, we recommend that range constraints be used whenever possible.

5.5 The BOUNDS Section

Again we recommend very strongly that upper and lower bounds be placed on variables whenever sensible values are known. Even if they are not essential (e.g., to avoid singularities in some of the functions $f^i(x)$), they can only help by reducing the size of the feasible region.

In many cases it is very easy to place meaningful bounds on all variables. For example, if you know that all components of x and y lie in the range $(-100, 100)$, you should put

```
LOWER BOUND    -100.0
UPPER BOUND     100.0
```

in the SPECS file. Similarly, uniform bounds of the form $x_j \geq 10^{-5}$ may be necessary to avoid evaluating $\log x_j$ at zero (say), and there will always be *some* reasonable upper bound on the variables, such as $x_j \leq 1000$. In this case,

```
LOWER BOUND    1.0E-5
UPPER BOUND    1000.0
```

will suffice. If some of the elements of x and y are bounded differently, suitable values can be specified in a bounds set in the MPS file.

Specifying (x_0, y_0).
The name `INITIAL` is reserved for a special bounds set, which may be used to specify a starting point (x_0, y_0) (or some of its components) when no basis file is available.

Remember that several bounds sets may exist in the MPS file, and if an `INITIAL` bounds set exists, it must be the last.

MINOS/AUGMENTED allows both linear and nonlinear variables to be initialized. Also, those specified with an `FX` indicator will become superbasic at the specified values, whether or not the values are feasible with respect to the upper and lower bounds. (These points relax two restrictions on page 29 of the MINOS User's Guide.)

The best set of variables to initialize depends, of course, on the application. In some cases, as many nonlinear variables as possible should be initialized (especially Jacobian variables — see below). However, this should not be at the expense of forming a very large set of superbasic variables. Bear in mind that the `SUPERBASICS LIMIT` and the `HESSIAN DIMENSION` should always be larger than the number of `FX` indicators. Hence for very large problems, Jacobian variables should be given first preference, followed by any "critical" nonlinear objective variables, followed perhaps by some important linear variables.

For Jacobian variables, the values specified are particularly important because they will be used to evaluate the initial constraint functions and gradients, regardless of feasibility. Suppose the first 5 variables **XJAC1**, **XJAC2**, ..., **XJAC5** are involved in the nonlinear constraints, and that their upper and lower bounds have previously been specified to be $2 \le$ **XJAC**$j \le 25$. The data cards

```
FX INITIAL   XJAC1        10.0
LO INITIAL   XJAC2        20.0
UP INITIAL   XJAC3        30.0
```

will have an effect that can be summarized as follows: the numerical values specify a point x_0 which *defines* the first subproblem, while the indicators determine a starting point for *solving* that subproblem. (These points would be the same if **FX** were used for all Jacobian variables.)

In this case:

1. x_0 is the point $(10, 20, 30, 2, 2)$. This will be used in the first call to subroutine **CALCON** to evaluate $f(x_0)$ and $J(x_0)$, and these quantities will be used (along with λ_0) to define the first subproblem (5)–(8). Note that the functions must be well defined, even though the value for **XJAC3** lies above its upper bound.

2. The **FX** indicator means that **XJAC1** should retain its value of 10 at the beginning of iteration 1. It will initially be superbasic at this value.

3. The **LO** indicator means that **XJAC2** will be moved to its lower bound, 2, at the start of the first iteration. However, it may be selected by one of the CRASH options to become basic, and in this case its initial value is unpredictable. (If this arbitrariness sounds troublesome, use **CRASH OPTION 2, 4** or **0**.)

4. The **UP** indicator means that **XJAC3** will be moved to its upper bound, 25, but again it may be selected by CRASH to become basic at an unpredictable value.

5. **XJAC4** and **XJAC5** take default values as described below.

The main point about Jacobian variables is that all numerical values are relevant, whether specified explicitly by the **FX**, **LO** and **UP** indicators or by default. For other variables, only the values on **FX** cards are used.

If the number of **FX** cards has reached the **SUPERBASICS LIMIT**, any further **FX** indicator will be treated as an **UP** or a **LO**, depending on which bound is closer to the specified numerical value.

By default, any variables not specified in the **INITIAL** bounds set will be made nonbasic at their upper or lower bounds (the smallest in absolute value), or at zero if a variable is free. Ties are broken in favor of lower bounds.

5.6 Comment Cards

Any card in the MPS file may contain the characters "*   " in columns 1–4 (i.e. an asterisk followed by three blanks), and arbitrary data in columns 5–12, 15–22 and 40–47. Such cards will be treated as comments. They will appear in the input listing but will otherwise be ignored.

Restriction: Columns 25–36 and 50–61 should preferably be blank. If not, they must contain valid numerical data whenever non-comment cards would do so. (This is a limitation of portable Fortran; data cannot be read under one format and then re-read under another.)

6. BASIS FILES

6.1 Cold Start

If there are no basis files available, any values specified in the INITIAL bounds set of the MPS file will be loaded (see section 5.5), the corresponding initial Jacobian will be evaluated, and then one of the CRASH options will be used to obtain a starting basis.

For large problems, CRASH OPTION 2 is often to be recommended. As many variables as possible (particularly nonlinear variables) should be assigned values by means of FX indicators in the INITIAL bounds set. They will then be held temporarily at the specified values, and efficient linear-programming iterations will be used to optimize any remaining *linear* variables as much as possible. There will be no calls to the nonlinear function subroutines during this phase.

If you happen to know that your problem is not particularly nonlinear (so there will not be many superbasic variables in the optimal solution), it may be preferable to use CRASH OPTION 1.

The remaining CRASH options have been implemented only for completeness. They may be useful in special circumstances.

6.2 Warm Start

A solution may be saved on a NEW BASIS FILE as described in the User's Guide [2], and this may be used as an OLD BASIS FILE to start a subsequent run, as long as the dimensions of the problem have not changed. When nonlinear constraints are present, the list of superbasic variables at the end of a NEW BASIS FILE is extended to include all basic nonlinear variables. (This is the set of values j, x_j on page 61 of the User's Guide.) The final Jacobian matrix can then be reconstructed exactly for a restart.

PUNCH and INSERT files may be used as documented in the User's Guide. (They already include values for basic nonlinear variables.) Similarly for DUMP and LOAD files.

7. EXAMPLES

Two example problems are described here to illustrate the subroutines and data required to specify a nonlinear program, and the corresponding output produced by MINOS/AUGMENTED.

The first example is small, dense and highly nonlinear; it shows how the Jacobian matrix may be handled most simply when there are very few nonlinear constraints or variables. The second example has both linear and nonlinear constraints, and illustrates most of the features that will be present in large-scale applications where it is essential to treat the Jacobian as a sparse matrix.

7.1 Test Problem MHW4D (Wright [8], example 4, starting point D)

Statement of problem:

$$\text{minimize} \quad (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^3 + (x_3 - x_4)^4 + (x_4 - x_5)^4$$

$$\text{subject to} \quad \begin{aligned} x_1 + x_2^2 + x_3^3 &= 3\sqrt{2} + 2, \\ x_2 - x_3^2 + x_4 &= 2\sqrt{2} - 2, \\ x_1 x_5 &= 2. \end{aligned}$$

Starting point: $x_0 = (-1, 2, 1, -2, -2)$

Notes:

1. The subroutines below happen to include code for a second problem (Wright [8], example 9). The parameter NPROB is used to branch to the appropriate calculations.

2. In subroutine CALCFG, F is the value of the objective function and G contains the corresponding 5 partial derivatives.

3. In subroutine CALCON, F is an array of constraint function values and the rows of G contain the derivatives for each constraint. In this example the Jacobian is best treated as a dense matrix, so G is a two-dimensional array. Note that several elements of G are actually zero; they do not need to be explicitly set.

4. Subroutine CALCON will be called before subroutine CALCFG. The parameter NSTATE is used to print a message on the very first entry to CALCON. This is just a matter of good practice, since it is often convenient to compile MINOS and the function routines into an executable code file, and it is easy to forget which particular function routines were used.

5. The SPECS file shown contains keywords that should in general be specified for small, dense problems (i.e. ones whose default values would not be ideal).

6. The MPS file should follow the SPECS file in the normal input stream, since it is not specified to be on any other file.

7. The COLUMNS section of the MPS file contains only the names of the variables, since they are all "nonlinear", and because there are no linear constraints.

8. The RHS section should, if possible, include estimates of the Lagrange multipliers. The more nonlinear a problem, the more valuable they are.

9. The BOUNDS section specifies only the initial point. (Uniform bounds on the variables are given in the SPECS file.)

10. Since `FX` indicators are used for the `INITIAL` bounds, the `SUPERBASICS LIMIT` needs to be at least 5 in this case.

11. This example has several local minima, and the performance of MINOS/ AUGMENTED is very dependent on the initial point x_0. See [4] or [8] for computational details.

(Example 1) Computation of the objective function:

```
      SUBROUTINE CALCFG( MODE,N,X,F,G,NSTATE,NPROB )
      IMPLICIT   REAL*8(A-H,O-Z)
      REAL*8     X(N),G(N)
C
C     MHW 4
C
      IF (NPROB .NE. 4) GO TO 500
      T1 = X(1) - 1.0
      T2 = X(1) - X(2)
      T3 = X(2) - X(3)
      T4 = X(3) - X(4)
      T5 = X(4) - X(5)
C
      F    =  T1**2 + T2**2 + T3**3 + T4**4 + T5**4
      G(1) =  2.0*(T1 + T2)
      G(2) = -2.0*T2 + 3.0*T3**2
      G(3) = -3.0*T3**2 + 4.0*T4**3
      G(4) = -4.0*T4**3 + 4.0*T5**3
      G(5) = -4.0*T5**3
      RETURN
C
C     MHW 9
C
  500 T1   = DSIN(X(5) - X(3))
      T2   = DCOS(X(5) - X(3))
      F    = 10.0*X(1)*X(4) + X(1)**3 * X(2) - 6.0*X(2)**2 * X(3)
     1       + 9.0*T1 + X(2)**3 * X(4)**2 * X(5)**4
      G(1) = 10.0*X(4)  +  3.0*X(1)**2 * X(2)
      G(2) = X(1)**3  -  12.0*X(2)*X(3)
     1       +  3.0*X(2)**2 * X(4)**2 * X(5)**4
      G(3) = -6.0*X(2)**2 - 9.0*T2
      G(4) = 10.0*X(1)  +  2.0*X(2)**3 * X(4) * X(5)**4
      G(5) =  9.0*T2  +  4.0*X(2)**3 * X(4)**2 * X(5)**3
      RETURN
C     END OF CALCFG FOR MHW4AND9
      END
```

(Example 1) Computation of the constraint functions:

```
      SUBROUTINE CALCON( MODE,M,N,NJAC,X,F,G,NSTATE,NPROB )
      IMPLICIT   REAL*8(A-H,O-Z)
      REAL*8     X(N),F(M),G(M,N)
C
C     MHW 4
C
      IF (NSTATE .EQ. 1) WRITE(6, 1000) NPROB
      IF (NPROB .NE. 4) GO TO 500
      F(1)   = X(1) + X(2)**2 + X(3)**3
      G(1,1) = 1.0
      G(1,2) = 2.0*X(2)
      G(1,3) = 3.0*X(3)**2
C
      F(2)   = X(2) - X(3)**2 + X(4)
      G(2,2) = 1.0
      G(2,3) = -2.0*X(3)
      G(2,4) = 1.0
C
      F(3)   = X(1)*X(5)
      G(3,1) = X(5)
      G(3,5) = X(1)
      RETURN
C
C     MHW 9
C
  500 F(1)   = X(1)**2 + X(2)**2 + X(3)**2 + X(4)**2 + X(5)**2
      G(1,1) = 2.0*X(1)
      G(1,2) = 2.0*X(2)
      G(1,3) = 2.0*X(3)
      G(1,4) = 2.0*X(4)
      G(1,5) = 2.0*X(5)
C
      F(2)   = X(1)**2*X(3) + X(4)*X(5)
      G(2,1) = 2.0*X(1)*X(3)
      G(2,3) = X(1)**2
      G(2,4) = X(5)
      G(2,5) = X(4)
C
      F(3)   = X(2)**2*X(4) + 10.0*X(1)*X(5)
      G(3,1) = 10.0*X(5)
      G(3,2) =  2.0*X(2)*X(4)
      G(3,4) = X(2)**2
      G(3,5) = 10.0*X(1)
      RETURN
 1000 FORMAT(/ 36H THIS IS PROBLEM  MHW4AND9.  NPROB =, I3)
C     END OF CALCON FOR MHW4AND9
      END
```

(Example 1) The SPECS file and the MPS file:

```
BEGIN MHW 4D
   MINIMIZE
   ROWS                           20
   COLUMNS                        20
   ELEMENTS                       50
   UPPER BOUND                   5.0
   LOWER BOUND                  -5.0

   NONLINEAR CONSTRAINTS           3
   NONLINEAR VARIABLES             5
   PROBLEM NO.                     4

   JACOBIAN                    DENSE
   MAJOR ITERATIONS               15
   MINOR ITERATIONS               20
   PENALTY PARAMETER            10.0
   PRINT LEVEL   (JFLXI)       10101

   SUPERBASICS                     6
   HESSIAN DIMENSION               6
   LINESEARCH TOLERANCE          0.1
   VERIFY OBJECTIVE GRADIENT
   VERIFY CONSTRAINT GRADIENTS

   CRASH OPTION                    1
   ITERATIONS                    100
END
```

```
NAME          MHW 4D
ROWS
 E  CON1
 E  CON2
 E  CON3
COLUMNS
    X1
    X2
    X3
    X4
    X5
RHS
    RHS       CON1         6.24263
    RHS       CON2         0.82842
    RHS       CON3         2.0
BOUNDS
 FX INITIAL   X1          -1.0
 FX INITIAL   X2           2.0
 FX INITIAL   X3           1.0
 FX INITIAL   X4          -2.0
 FX INITIAL   X5          -2.0
ENDATA
```

(Example 1) Solution obtained by MINOS/AUGMENTED:

```
PROBLEM NAME    MHW 4D              OBJECTIVE VALUE    2.7871880860D+01

STATUS          OPTIMAL SOLN        ITERATION   21     SUPERBASICS    2

OBJECTIVE                (MIN)
RHS             RHS
RANGES
BOUNDS

SECTION 1 - ROWS

 NUMBER  ...ROW..  AT  ...ACTIVITY...  SLACK ACTIVITY  ..LOWER LIMIT.  ..UPPER LIMIT.  .DUAL ACTIVITY   ..I

      7  CON1      EQ         6.24263       0.0             6.24263         6.24263         2.12527     1
      8  CON2      EQ         0.82842       0.0             0.82842         0.82842         1.55378     2
      9  CON3      EQ         2.00000       0.0             2.00000         2.00000         8.93568     3


SECTION 2 - COLUMNS

 NUMBER  .COLUMN.  AT  ...ACTIVITY...  .OBJ GRADIENT.  ..LOWER LIMIT.  ..UPPER LIMIT.  .REDUCED COST.   M+J

      1  X1        BS        -1.27305       -11.91292        -5.00000         5.00000        -0.00000     4
      2  X2        SBS        2.41035        11.79905        -5.00000         5.00000        -0.00000     5
      3  X3        BS         1.19486         5.38957        -5.00000         5.00000         0.00000     6
      4  X4        BS        -0.15424         1.55378        -5.00000         5.00000         0.0         7
      5  X5        SBS       -1.57103       -11.37559        -5.00000         5.00000        -0.00000     8
A     6  RHS       EQ        -1.00000         0.0            -1.00000        -1.00000       -32.42579     9
```

7.2 Test problem MANNE10 (Manne [1], $T = 10$)

Statement of problem.

$$\text{maximize} \quad \sum_{t=1}^{T} \beta_t \log C_t$$

$$\begin{aligned} \text{subject to} \quad & \alpha_t K_t^b \geq C_t + I_t, \quad && 1 \leq t \leq T, \quad && \text{(nonlinear constraints)} \\ & K_{t+1} \leq K_t + I_t, && 1 \leq t < T, && \text{(linear constraints)} \\ & g K_T \leq I_T, && && \end{aligned}$$

with various ranges and bounds.

The variables here are K_t, C_t and I_t, representing capital, consumption and investment during T time periods. This problem is described more fully in [4], where results are given for the case $T = 100$.

Notes:

1. For efficiency, the Jacobian variables K_t are made the first T components of x, followed by the objective variables C_t. Since the objective does not involve K_t, subroutine CALCFG must set the first T components of the objective gradient to zero. The parameter N will have the value $2T$. Verification of the objective gradients may as well start at variable $T + 1$.

2. For subroutine CALCON, N will be T. The Jacobian matrix is particularly simple in this example; in fact $J(x)$ has only one nonzero element per column (i.e. it is diagonal). The parameter NJAC will therefore be T also. It is used only to dimension the array G.

3. NSTATE enables B, AT and BT to be initialized on the first entry to CALCON, for subsequent use in both of the function subroutines. (Remember that the first call to CALCON occurs before the first call to CALCFG.) The name chosen for the labeled COMMON block holding these quantities must be different from the other COMMON names used by MINOS, as listed in section 3.3.

4. The COMMON block IOCOMM is one of the blocks used by MINOS.

5. NSTATE is also used to produce some output on the final call to CALCON, at the optimal solution.

6. The SPECS file uses keywords that you should become familiar with before running large problems. Other values will be appropriate for other applications.

7. The MPS file specifies a sparse T by T Jacobian in the top left corner of the constraint matrix. An arbitrary value of 0.1 has been used for the nonzero variable coefficients. A zero or blank numeric field would be equally good.

(Example 2) Calculation of the objective function:

```
      SUBROUTINE CALCFG( MODE,N,X,F,G,NSTATE,NPROB )
      IMPLICIT   REAL*8(A-H,O-Z)
      REAL*8     X(N),G(N)
      COMMON    /MANNE / B,AT(100),BT(100)
C
      NT = N/2
      F  = 0.0
      DO 50 J = 1, NT
         XCON = X(NT+J)
         F    = F + BT(J)*DLOG(XCON)
         G(J) = 0.0
         G(NT+J) = BT(J)/XCON
   50 CONTINUE
      RETURN
C
C     END OF CALCFG FOR MANNE
      END
```

(Example 2) Calculation of the constraint functions:

```
      SUBROUTINE CALCON( MODE,M,N,NJAC,X,F,G,NSTATE,NPROB )
      IMPLICIT   REAL*8(A-H,O-Z)
      REAL*8     X(N),F(M),G(NJAC)
      COMMON     /IOCOMM/ IREAD,IPRINT
      COMMON     /MANNE / B,AT(100),BT(100)
C
      NT = N
      IF (NSTATE .NE. 1) GO TO 100
C
C     FIRST ENTRY
C     -----------
      ONE   = 1.0
      GROW  = 0.03
      BETA  = 0.95
      XKO   = 3.0
      XCO   = 0.95
      XIO   = 0.05
      B     = 0.25
      BPROB = NPROB
      IF (NPROB .NE. 1) B = BPROB/100.0
      WRITE(IPRINT, 1000) B
C
      A     = (XCO + XIO)/XKO**B
      GFAC  = (ONE + GROW)**(ONE - B)
      AT(1) = A*GFAC
      BT(1) = BETA
      DO 10 J = 2, NT
         AT(J) = AT(J-1)*GFAC
         BT(J) = BT(J-1)*BETA
   10 CONTINUE
      BT(NT) = BT(NT)/(ONE - BETA)
C
C     NORMAL ENTRY
C     ------------
  100 DO 150 J = 1, NT
         XKAP = X(J)
         FJ   = AT(J)*XKAP**B
         F(J) = FJ
         G(J) = B*FJ/XKAP
  150 CONTINUE
      IF (NSTATE .NE. 2) RETURN
C
C     FINAL ENTRY
C     -----------
      WRITE(IPRINT, 2000) (F(J), J = 1, NT)
      RETURN
C
 1000 FORMAT(// 30H THIS IS PROBLEM  MANNE.   B =, F8.3)
 2000 FORMAT(// 32H FINAL NONLINEAR FUNCTION VALUES / (5F12.5))
C     END OF CALCON FOR MANNE
      END
```

(Example 2) The SPECS file:

```
BEGIN MANNE10
   MAXIMIZE
   NONLINEAR CONSTRAINTS     10
   NONLINEAR JACOBIAN VARS   10
   NONLINEAR OBJECTIV VARS   20

   OBJECTIVE = CALCFG
   PROBLEM NUMBER    1

   MPS FILE                   5
   ROWS                     100
   COLUMNS                  100
   ELEMENTS                 200
   UPPER BOUND            100.0

   COMPLETION              FULL
   JACOBIAN              SPARSE
   LAGRANGIAN               YES
   MAJOR ITERATIONS          10
   MINOR ITERATIONS          20
   PENALTY PARAMETER        0.1

   FEASIBILITY TOL       1.0E-6
   DJ TOLERANCE          1.0E-6
   ROW TOLERANCE         1.0E-6
   RADIUS OF CONVERGENCE   0.01

   SUPERBASICS               10
   HESSIAN DIMENSION         10
   LINESEARCH TOLERANCE     0.1
   VERIFY GRADIENTS
   START OBJECTIVE GRADIENT CHECK AT VARIABLE    11
   STOP CONSTRAINT GRADIENT CHECK AT VARIABLE     5

   CRASH OPTION               1
   ITERATIONS               100
   MULTIPLE PRICE             5
   PRINT LEVEL (JFLXI)      101
   SOLUTION                 YES
   CALL FUNCTION ROUTINES WHEN OPTIMAL
END MANNE10
```

(Example 2) The MPS file:

```
NAME          MANNE10
ROWS
 G  MON001
 G  MON002
 G  MON003
 G  MON004
 G  MON005
 G  MON006
 G  MON007
 G  MON008
 G  MON009
 G  MON010
 L  CAP002
 L  CAP003
 L  CAP004
 L  CAP005
 L  CAP006
 L  CAP007
 L  CAP008
 L  CAP009
 L  CAP010
 L  TERMINV
COLUMNS
    KAPQ01    MON001       .1         CAP001       1.0
    KAP001    CAP002     -1.0
    KAP002    MON002       .1         CAP002       1.0
    KAP002    CAP003     -1.0
    KAP003    MON003       .1         CAP003       1.0
    KAP003    CAP004     -1.0
    KAP004    MON004       .1         CAP004       1.0
    KAP004    CAP005     -1.0
    KAP005    MON005       .1         CAP005       1.0
    KAP005    CAP006     -1.0
    KAP006    MON006       .1         CAP006       1.0
    KAP006    CAP007     -1.0
    KAP007    MON007       .1         CAP007       1.0
    KAP007    CAP008     -1.0
    KAP008    MON008       .1         CAP008       1.0
    KAP008    CAP009     -1.0
    KAP009    MON009       .1         CAP009       1.0
    KAP009    CAP010     -1.0
    KAP010    MON010       .1         CAP010       1.0
    KAP010    TERMINV      .03
    CON001    MON001     -1.0
    CON002    MON002     -1.0
    CON003    MON003     -1.0
    CON004    MON004     -1.0
    CON005    MON005     -1.0
    CON006    MON006     -1.0
    CON007    MON007     -1.0
    CON008    MON008     -1.0
    CON009    MON009     -1.0
    CON010    MON010     -1.0
    INV001    MON001     -1.0         CAP002      -1.0
    INV002    MON002     -1.0         CAP003      -1.0
    INV003    MON003     -1.0         CAP004      -1.0
    INV004    MON004     -1.0         CAP005      -1.0
    INV005    MON005     -1.0         CAP006      -1.0
    INV006    MON006     -1.0         CAP007      -1.0
    INV007    MON007     -1.0         CAP008      -1.0
    INV008    MON008     -1.0         CAP009      -1.0
    INV009    MON009     -1.0         CAP010      -1.0
    INV010    MON010     -1.0         CAP011      -1.0
    INV010    TERMINV    -1.0
```

The MPS file, continued:

```
RHS
*
*   THE  RHS  IS ZERO
*
    LAGRANGE  MON001      1.0             MON010     10.0
RANGES
    RANGE1    MON010     10.0             TERMINV    20.0
BOUNDS
 FX BOUND1    KAP001      3.05
 LO BOUND1    KAP002      3.05
 LO BOUND1    KAP003      3.05
 LO BOUND1    KAP004      3.05
 LO BOUND1    KAP005      3.05
 LO BOUND1    KAP006      3.05
 LO BOUND1    KAP007      3.05
 LO BOUND1    KAP008      3.05
 LO BOUND1    KAP009      3.05
 LO BOUND1    KAP010      3.05
 LO BOUND1    CON001       .95
 LO BOUND1    CON002       .95
 LO BOUND1    CON003       .95
 LO BOUND1    CON004       .95
 LO BOUND1    CON005       .95
 LO BOUND1    CON006       .95
 LO BOUND1    CON007       .95
 LO BOUND1    CON008       .95
 LO BOUND1    CON009       .95
 LO BOUND1    CON010       .95
 LO BOUND1    INV001       .05
 LO BOUND1    INV002       .05
 LO BOUND1    INV003       .05
 LO BOUND1    INV004       .05
 LO BOUND1    INV005       .05
 LO BOUND1    INV006       .05
 LO BOUND1    INV007       .05
 LO BOUND1    INV008       .05
 LO BOUND1    INV009       .05
 LO BOUND1    INV010       .05
 UP BOUND1    INV008       .112
 UP BOUND1    INV009       .114
 UP BOUND1    INV010       .116
 FX INITIAL   KAP002      3.1
 FX INITIAL   KAP003      3.2
 FX INITIAL   KAP004      3.3
 FX INITIAL   KAP005      3.4
 FX INITIAL   KAP006      3.5
 FX INITIAL   KAP007      3.6
 FX INITIAL   KAP008      3.7
 FX INITIAL   KAP009      3.8
 FX INITIAL   KAP010      3.9
ENDATA
```

(Example 2) Output from MINOS/AUGMENTED:

```
M I N O S    ---    VERSION 4.0    MAY 1980
* * * * *

PROBLEM SPECIFICATIONS
----------------------
0000.       BEGIN MANNE10
0001.          MAXIMIZE
0002.          NONLINEAR CONSTRAINTS     10
0003.          NONLINEAR JACOBIAN VARS   10
0004.          NONLINEAR OBJECTIV VARS   20
0005.
0006.          OBJECTIVE = CALCFG
0007.          PROBLEM NUMBER    1
0007.1
0007.2         MPS FILE                   5
0009.          ROWS                     100
0010.          COLUMNS                  100
0011.          ELEMENTS                 200
0012.          UPPER BOUND            100.0
0013.
0014.          COMPLETION              FULL
0015.          JACOBIAN              SPARSE
0016.          LAGRANGIAN               YES
0017.          MAJOR ITERATIONS          10
0018.          MINOR ITERATIONS          20
0019.          PENALTY PARAMETER        0.1
0020.
0021.          FEASIBILITY TOL       1.0E-6
0022.          DJ TOLERANCE          1.0E-6
0023.          ROW TOLERANCE         1.0E-6
0024.          RADIUS OF CONVERGENCE  0.01
0025.
0026.          SUPERBASICS               10
0027.          HESSIAN DIMENSION         10
0028.          LINESEARCH TOLERANCE     0.1
0029.          VERIFY GRADIENTS
0029.1         START OBJECTIVE GRADIENT CHECK AT VARIABLE   11
0029.2         STOP CONSTRAINT GRADIENT CHECK AT VARIABLE    5
0030.
0031.          CRASH OPTION               1
0032.          ITERATIONS               100
0033.          MULTIPLE PRICE             5
0034.          PRINT LEVEL (JFLXI)      101
0035.          SOLUTION                 YES
0035.1         CALL FUNCTION ROUTINES WHEN OPTIMAL
0036.       END MANNE10


PARAMETERS
----------
MPS INPUT DATA.
ROW LIMIT..............       100     LIST LIMIT.............         0     LOWER BOUND DEFAULT....  0.0
COLUMN LIMIT...........       100     ERROR MESSAGE LIMIT....        10     UPPER BOUND DEFAULT....  1.00E+02
ELEMENTS LIMIT (COEFFS)       200     PHANTOM ELEMENTS.......         0     AIJ TOLERANCE..........  1.00E-10

FILES.
MPS FILE (INPUT FILE)..         5     OLD BASIS FILE (MAP)...         0     (CARD READER)..........         5
SOLUTION FILE..........         0     NEW BASIS FILE (MAP)...         0     (PRINTER)..............         6
INSERT FILE............         0     BACKUP BASIS FILE......         0     (SCRATCH FILE).........         8
PUNCH FILE.............         0     LOAD FILE..............         0     DUMP FILE..............         0

FREQUENCIES.
LOG ITERATIONS.........         1     CHECK ROW ERROR........        30     CYCLE LIMIT............         1
SAVE NEW BASIS MAP.....       100     FACTORIZE (INVERT).....        60     CYCLE TOLERANCE........  0.0

LP PARAMETERS.
ITERATIONS LIMIT.......       100     FEASIBILITY TOLERANCE..  1.00D-06     PARTIAL PRICE FACTOR...         1
CRASH OPTION...........         1     DJ TOLERANCE...........  1.00D-06     MULTIPLE PRICE.........         5
WEIGHT ON OBJECTIVE....   0.0         PIVOT TOLERANCE........  1.49D-08

NONLINEAR PROBLEMS.
NONLINEAR CONSTRAINTS..        10     SUPERBASICS LIMIT......        10     DERIVATIVE LEVEL.......         3
NONLINEAR JACOBIAN VARS        10     HESSIAN DIMENSION......        10     VERIFY LEVEL...........         3
NONLINEAR OBJECTIV VARS        20     LINESEARCH TOLERANCE...   0.10000     DIFFERENCE INTERVAL....  2.58D-08
PROBLEM NUMBER.........         1     REDUCED-GRADIENT TOL...   0.20000     CONJUGATE-GRADNT METHOD         1

AUGMENTED LAGRANGIAN.
JACOBIAN...............    SPARSE     MAJOR ITERATIONS LIMIT.        10     RADIUS OF CONVERGENCE..  1.00D-02
LAGRANGIAN.............       YES     MINOR ITERATIONS LIMIT.        20     ROW TOLERANCE..........  1.00D-06
PENALTY PARAMETER......  1.00D-01     COMPLETION.............      FULL

MISCELLANEOUS.
LU ROW TOLERANCE.......  1.00D-03     PRINT LEVEL..(JFLXI)...       101     IMBED..................       YES
LU COL TOLERANCE.......   0.10000     DEBUG LEVEL............         0     PRINT SPIKES...........        NO
LU MOD TOLERANCE.......   0.50000

NUMBER OF WORDS OF CORE AVAILABLE FOR WORKSPACE     8204
```

Output, continued:

```
INPUT LISTING
-------------
     1     NAME           MANNEIG
     2     ROWS
    23     COLUMNS

XXXX  WARNING - NO LINEAR OBJECTIVE FUNCTION FOUND

XXXX  NON-EXISTENT ROW SPECIFIED -- CAPO01   -- ENTRY IGNORED IN LINE     24

XXXX  NON-EXISTENT ROW SPECIFIED -- CAPO11   -- ENTRY IGNORED IN LINE     63
    65     RHS
    66     *
    67     *   THE  RHS  IS ZERO
    68     *
    70     RANGES
    72     BOUNDS
   115     ENDATA


NAMES SELECTED
--------------
OBJECTIVE      CALCFG   (MAX)       0
RHS            RHS                  0
RANGES         RANGE1               2
BOUNDS         BOUND1              33


MATRIX STATISTICS
-----------------
              TOTAL      NORMAL        FREE       FIXED    BOUNDED
ROWS             20          18           0           0          2
COLUMNS          30           0           0           1         29

NO. OF MATRIX ELEMENTS         59     DENSITY      9.516
NO. OF REJECTED COEFFS          0      AIJTOL 1.00000E-10
BIGGEST AND SMALLEST COEFFS   1.00000E+00  3.00000E-02  (EXCLUDING OBJ AND RHS)

XXXX  TOTAL NO. OF ERRORS DURING INPUT    2

LENGTH OF ROW-NAME HASH TABLE           211
COLLISIONS DURING TABLE LOOKUP            0

NO. OF JACOBIAN ENTRIES SPECIFIED        10
NO. OF LAGRANGE MULTIPLIERS SPECIFIED     2

NO. OF  INITIAL  BOUNDS PROCESSED         9
NO. OF SUPERBASICS SPECIFIED              5
```

Output, continued:

```
ITERATIONS
----------


CRASH OPTION  1
FREE ROWS     0     FREE COLS     0     PASS2 (E ROWS)     0     PASS3    20     REMAINDER     0


THIS IS PROBLEM  MANNE.   B =   0.250

MULTIPLIER ESTIMATES
--------------------
  1.0000000D+00  0.0            0.0            0.0            0.0
  0.0            0.0            0.0            0.0            1.0000000D+01

FACTORIZE     1    DEMAND      0    ITERATION      0    INFEAS       1    OBJECTV   0.0
SLACKS        0    LINEAR     10    NONLINEAR     10    ELEMS       30    DENSITY   7.5
P4 BUMPS      0    SPIKES      0    CORE REQD    579    L LIMIT   1864    U LIMIT   3728
LU BUMPS      0    SPIKES      0    AIJ ELEMS     30    L ELEMS     21    U ELEMS      1     F ELEMS     0   0.0
ITN     0 -- INFEASIBLE.  NUM =     1    SUM =    9.5999654295D-04

 ITN PH PP NOPT  DJ/RG  +SBS -SBS  -BS   STEP    PIVOT   NSPK    L     U NINF  SINF/OBJECTIVE   NFG  NSB  RIM H-CONDN CONV
   1  4  0    0 0.0          0   10   30  1.1D+00 -3.0D-02    0   21     1    1  5.59996543D-04     1    8  1 0 0.0     TTTT

ITN     1 -- FEASIBLE SOLUTION.  OBJECTIVE =   2.668996414D+00



VERIFICATION OF OBJECTIVE GRADIENTS RETURNED BY SUBROUTINE CALCFG.

COMPUTED GTP VIA CALCFG      4.57304926089D-01   8.28544880395D-01
DIFFERENCE APPROXIMATION     4.57304916192D-01   8.28544868426D-01

OBJECTIVE GRADIENTS SEEM TO BE OK.


    J        X(J)          DX             G(J)           DIFFERENCE APPROXN

   11   9.76650000D-01  1.13D-05     9.72712833D-01     9.72707224D-01   OK
   12   9.53940674D-01  1.14D-05     9.46075581D-01     9.46069919D-01   OK
   13   9.86152359D-01  1.19D-05     8.69414305D-01     8.69409066D-01   OK
   14   1.01907706D+00  1.23D-05     7.99258683D-01     7.99253841D-01   OK
   15   1.05273325D+00  1.28D-05     7.35020854D-01     7.35016383D-01   OK
   16   1.08714572D+00  1.33D-05     6.76166796D-01     6.76162674D-01   OK
   17   1.12233674D+00  1.37D-05     6.22217209D-01     6.22213412D-01   OK
   18   1.15832620D+00  1.41D-05     5.72740533D-01     5.72737041D-01   OK
   19   1.22847402D+00  1.47D-05     5.13034322D-01     5.13031256D-01   OK
   20   1.21395205D+00  2.05D-06     9.86425649D+00     9.86424818D+00   OK

OBJECTIVE   GRADIENTS    11   THRU    20   SEEM TO BE OK.



VERIFICATION OF CONSTRAINT GRADIENTS RETURNED BY SUBROUTINE CALCON.

COLUMN          X            DX      ELEMENT NO.    ROW    JACOBIAN VALUE     DIFFERENCE APPROXN

     1   3.05000019D+00  1.21D-07         1           1    8.41516617D-02     8.41516640D-02   OK

     2   3.10000038D+00  1.22D-07         2           2    8.49951617D-02     8.49951616D-02   OK

     3   3.19999981D+00  1.25D-07         3           3    8.48556901D-02     8.48556898D-02   OK

     4   3.30000019D+00  1.28D-07         4           4    8.47785295D-02     8.47785311D-02   OK

     5   3.39999962D+00  1.31D-07         5           5    8.47598386D-02     8.47598374D-02   OK
CONSTRAINT GRADIENTS     1  THRU     5  SEEM TO BE OK.


CHOLESKY FACTOR OF HESSIAN RESET TO  1.
   2  4  0   0  2.5D-02    0    0    0  4.7D-01  0.0        1     21     4    0  2.66973324D+00     4     8  4 4 2.3D+00 TTFF
   3  4  0   0  1.9D-02    0    2   11  1.5D-01  1.0D+00    1     21     4    0  2.66982983D+00     5     7  2 2 2.3D+00 TTFF
   4  4  0   0  9.5D-03    0    0    0  1.0D+00  0.0        2     21     7    0  2.67002687D+00     6     7  4 4 2.5D+00 TTTF
RG TOLS REDUCED.  TOLRG =     1.501D-05

   5  4  0   0  4.9D-03    0    0    0  6.9D-01  0.0        2     21     7    0  2.67007306D+00     8     7  4 4 3.0D+00 FFFF
   6  4  0   0  2.6D-03    0    0    0  1.0D+00  0.0        2     21     7    0  2.67009168D+00     9     7  4 4 3.0D+00 FFFF
   7  4  0   0  1.3D-03    0    0    0  1.0D+00  0.0        2     21     7    0  2.67010007D+00    10     7  4 4 3.1D+00 FFFF
   8  4  0   0  1.7D-03    0    0    0  2.2D+00  0.0        2     21     7    0  2.67010552D+00    12     7  4 4 3.2D+00 FFFF
   9  4  0   0  7.3D-04    0    0    0  1.6D+00  0.0        2     21     7    0  2.67011176D+00    14     7  4 4 3.3D+00 FFFF
  10  4  0   0  5.6D-05    0    0    0  1.0D+00  0.0        2     21     7    0  2.67011231D+00    15     7  4 4 3.4D+00 FFFF
  11  4  0   0  7.7D-06    0    0    0  1.2D+00  0.0        2     21     7    0  2.67011232D+00    17     7  4 4 3.5D+00 TFTF
  12  4  0   0  4.6D-07    0    0    0  1.0D+00  0.0        2     21     7    0  2.67011232D+00    18     7  4 4 3.4D+00 TTTT

BIGGEST DJ =     0.0          NORM RG =    4.577D-07   NORM PI =     1.492D+01   NORM X =     3.900D+00

END OF MAJOR ITN    1  -  OPTIMAL SOLN AT MINOR ITN  12  -  TOTAL ITNS =     12
```

Output, continued:

```
START OF MAJOR ITN   2  -  PENALTY PARAMETER =    1.00D-01

MULTIPLIER ESTIMATES
--------------------
  1.0110493D+00   9.2184688D-01   8.5895573D-01   7.5182982D-01   7.2995850D-01
  6.7289923D-01   6.2024976D-01   5.7164699D-01   5.2676331D-01   9.8642565D+00

ROW ERROR AFTER RELINEARIZATION =   2.3493D-06
RELATIVE CHANGE IN MULTIPLIERS  =   3.3283D-01

FACTORIZE     2    DEMAND      0    ITERATION    12    INFEAS      0    OBJECTV   2.670112318D+00
SLACKS        0    LINEAR      9    NONLINEAR    11    ELEMS      33    DENSITY   8.3
P4 BUMPS      0    SPIKES      0    CORE REQD   580    L LIMIT  4196    U LIMIT  1398
LU BUMPS      0    SPIKES      0    AIJ ELEMS    33    L ELEMS    21    U ELEMS     1    F ELEMS     0   0.0
ITN   12 -- FEASIBLE SOLUTION.  OBJECTIVE =   2.670095339D+00

 ITN PH PP NOPT  DJ/RG  +SBS -SBS  -BS   STEP     PIVOT    NSPK    L     U NINF  SINF/OBJECTIVE   NFC  NSB  RIM H-CONDN CONV
  13  4  0   0 2.0D-05    0    0    0 1.0D+00  0.0         0    21     1    0  2.67009598D+00   21    7  4 4 3.3D+00 TTFT

RG TOLS REDUCED.  TOLRG =    1.493D-05

  14  4  0   0 8.7D-07    0    0    0 1.0D+00  0.0         0    21     1    0  2.67009599D+00   22    7  4 4 3.3D+00 FFTT

BIGGEST DJ =    0.0          NORM RG =    8.687D-07   NORM PI =    1.493D+01   NORM X =    3.867D+00

END OF MAJOR ITN   2  -  OPTIMAL SOLN AT MINOR ITN   2  -  TOTAL ITNS =    14




START OF MAJOR ITN   3  -  PENALTY PARAMETER =    1.00D-01

MULTIPLIER ESTIMATES
--------------------
  1.0106338D+00   9.3193104D-01   8.5926408D-01   7.5216711D-01   7.3020976D-01
  6.7299396D-01   6.2015130D-01   5.7134097D-01   5.2624756D-01   9.8643303D+00

ROW ERROR AFTER RELINEARIZATION =   5.5570D-06
RELATIVE CHANGE IN MULTIPLIERS  =   1.4115D-04

PENALTY PARAMETER DECREASED TO    0.0


FACTORIZE     3    DEMAND      0    ITERATION    14    INFEAS      0    OBJECTV   2.670095985D+00
SLACKS        0    LINEAR      9    NONLINEAR    11    ELEMS      33    DENSITY   8.3
P4 BUMPS      0    SPIKES      0    CORE REQD   580    L LIMIT  4663    U LIMIT   932
LU BUMPS      0    SPIKES      0    AIJ ELEMS    33    L ELEMS    21    U ELEMS     1    F ELEMS     0   0.0
ITN   14 -- FEASIBLE SOLUTION.  OBJECTIVE =   2.670096032D+00
NORM RG IS ALREADY SMALL    9.679D-07 --- RETURN TO PHASE 3.   NORM PI =    1.493D+01

BIGGEST DJ =    0.0          NORM RG =    9.679D-07   NORM PI =    1.493D+01   NORM X =    3.867D+00

END OF MAJOR ITN   3  -  OPTIMAL SOLN AT MINOR ITN   0  -  TOTAL ITNS =    14




EXIT -- OPTIMAL SOLUTION FOUND.

NO. OF ITERATIONS                       14      OBJECTIVE VALUE          2.6700960319077D+00

NO. OF MAJOR ITERATIONS                  3      LINEAR OBJECTIVE         0.0

OBJECTIVE  FUNCN AND GRADIENT CALLS     21      NONLINEAR OBJECTIVE      2.6700960319077D+00

CONSTRAINT FUNCN AND GRADIENT CALLS     24      PENALTY PARAMETER                  0.0

NORM OF X                        3.867D+00      NORM OF PI                   1.493D+01

NO. OF SUPERBASICS                       7      NORM OF REDUCED GRADIENT     9.679D-07

FINAL NONLINEAR FUNCTION VALUES
     1.02665     1.05620     1.08738     1.11942     1.15233
     1.18612     1.22078     1.25632     1.29271     1.32994
```

Output, continued:

```
PROBLEM NAME   MANNE10            OBJECTIVE VALUE   2.6700960319D+00

STATUS         OPTIMAL SOLN       ITERATION   14    SUPERBASICS   7

OBJECTIVE      CALCFG   (MAX)
RHS            RHS
RANGES         RANGE1
BOUNDS         BOUND1

SECTION 1 - ROWS
```

NUMBER	...ROW..	AT	...ACTIVITY...	SLACK ACTIVITY	..LOWER LIMIT.	..UPPER LIMIT.	.DUAL ACTIVITY	..I
32	MON001	LL	0.0	0.0	0.0	NONE	1.01063	1
33	MON002	LL	0.0	0.0	0.0	NONE	0.93193	2
34	MON003	LL	0.0	0.0	0.0	NONE	0.85926	3
35	MON004	LL	0.0	0.0	0.0	NONE	0.79217	4
36	MON005	LL	0.0	0.0	0.0	NONE	0.73021	5
37	MON006	LL	0.0	0.0	0.0	NONE	0.67299	6
38	MON007	LL	0.0	0.0	0.0	NONE	0.62015	7
39	MON008	LL	0.0	0.0	0.0	NONE	0.57134	8
40	MON009	LL	0.0	0.0	0.0	NONE	0.52625	9
41	MON010	LL	0.0	0.0	0.0	10.00000	9.86433	10
42	CAP002	UL	0.0	0.0	NONE	0.0	-1.01063	11
43	CAP003	UL	0.0	0.0	NONE	0.0	-0.93193	12
44	CAP004	UL	0.0	0.0	NONE	0.0	-0.85926	13
45	CAP005	UL	0.0	0.0	NONE	0.0	-0.79217	14
46	CAP006	UL	0.0	0.0	NONE	0.0	-0.73021	15
47	CAP007	UL	0.0	0.0	NONE	0.0	-0.67299	16
48	CAP008	UL	0.0	0.0	NONE	0.0	-0.62015	17
49	CAP009	UL	0.0	0.0	NONE	0.0	-0.57134	18
50	CAP010	UL	0.0	0.0	NONE	0.0	-0.52625	19
51	TERMINV	UL	0.0	0.0	-20.00000	0.0	-10.73212	20

```
SECTION 2 - COLUMNS
```

	NUMBER	.COLUMN.	AT	...ACTIVITY...	.OBJ GRADIENT.	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.	M+J
	1	KAP001	EQ	3.05000	0.00000	3.05000	3.05000	1.09568	21
	2	KAP002	BS	3.12665	0.00000	3.05000	100.00000	0.00000	22
	3	KAP003	SBS	3.21443	0.00000	3.05000	100.00000	0.00000	23
	4	KAP004	SBS	3.30400	0.00000	3.05000	100.00000	0.00000	24
	5	KAP005	SBS	3.39522	0.00000	3.05000	100.00000	0.00000	25
	6	KAP006	SBS	3.48788	0.00000	3.05000	100.00000	0.00000	26
	7	KAP007	SBS	3.58172	0.00000	3.05000	100.00000	-0.00000	27
	8	KAP008	SBS	3.67643	0.00000	3.05000	100.00000	-0.00000	28
	9	KAP009	SBS	3.77158	0.00000	3.05000	100.00000	-0.00000	29
	10	KAP010	BS	3.86667	0.00000	3.05000	100.00000	0.00000	30
	11	CON001	LL	0.95000	1.00000	0.95000	100.00000	-0.01063	31
	12	CON002	BS	0.96842	0.93193	0.95000	100.00000	0.0	32
	13	CON003	BS	0.99780	0.85926	0.95000	100.00000	0.0	33
	14	CON004	BS	1.02820	0.79217	0.95000	100.00000	0.0	34
	15	CON005	BS	1.05967	0.73021	0.95000	100.00000	0.0	35
	16	CON006	BS	1.09227	0.67299	0.95000	100.00000	0.0	36
	17	CON007	BS	1.12606	0.62015	0.95000	100.00000	0.0	37
	18	CON008	BS	1.16116	0.57134	0.95000	100.00000	0.0	38
	19	CON009	BS	1.19763	0.52625	0.95000	100.00000	0.0	39
	20	CON010	BS	1.21394	9.86433	0.95000	100.00000	0.0	40
	21	INV001	BS	0.07665	0.0	0.05000	100.00000	0.0	41
	22	INV002	BS	0.08778	0.0	0.05000	100.00000	0.0	42
	23	INV003	BS	0.08957	0.0	0.05000	100.00000	0.0	43
	24	INV004	BS	0.09122	0.0	0.05000	100.00000	0.0	44
	25	INV005	BS	0.09266	0.0	0.05000	100.00000	0.0	45
	26	INV006	BS	0.09385	0.0	0.05000	100.00000	0.0	46
	27	INV007	BS	0.09471	0.0	0.05000	100.00000	0.0	47
	28	INV008	BS	0.09515	0.0	0.05000	0.11200	0.0	48
	29	INV009	BS	0.09508	0.0	0.05000	0.11400	0.0	49
	30	INV010	UL	0.11600	0.0	0.05000	0.11600	0.86779	50
A	31	RHS	EQ	-1.00000	0.0	-1.00000	-1.00000	0.0	51

REFERENCES

[1] Manne, A. S. (1979). Private communication.

[2] Murtagh, B. A. and Saunders, M. A. (1977). MINOS User's Guide, Report SOL 77-9, Department of Operations Research, Stanford University.

[3] Murtagh, B. A. and Saunders, M. A. (1978). Large-scale linearly constrained optimization, *Math. Prog.* **14**, pp. 41–72.

[4] Murtagh, B. A. and Saunders, M. A. (1980). The implementation of a Lagrangian-based algorithm for sparse nonlinear constraints, Report SOL 80-1, Department of Operations Research, Stanford University.

[5] Robinson, S. M. (1972). A quadratically convergent algorithm for general nonlinear programming problems, *Math. Prog.* **3**, pp. 145–156.

[6] Preckel, P. V. (1980). Modules for use with MINOS/AUGMENTED in solving sequences of mathematical programs, Report SOL 80-15, Department of Operations Research, Stanford University.

[7] Saunders, M. A. (1977). MINOS System Manual, Report SOL 77-31, Department of Operations Research, Stanford University.

[8] Wright, M. H. (1976). Numerical Methods for Nonlinearly Constrained Optimization, Ph. D. Thesis, Stanford University.

INDEX

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. ~~REPORT NUMBER~~ (14) SOL-80-14	2. GOVT ACCESSION NO. AD-A089 351	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MINOS/AUGMENTED USER'S MANUAL . (9)		5. TYPE OF ~~REPORT & PERIOD~~ COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Bruce A. MURTAGH ~~and~~ Michael A. SAUNDERS		8. CONTRACT OR GRANT NUMBER(s) (15) DAAG29-79-C-0110 N00014-75-C-0267
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research - SOL Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Office P.O. Box 1221 Research Triangle Park, NC 27709 (11)		12. ~~REPORT DATE~~ Jun[redacted]980
		13. NUMBER OF PAGES 51
Office of Naval Research Department of the Navy 800 N. Quincy Street Arlington, VA 22217 (12) 51		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

AUGMENTED LAGRANGIAN	NONLINEAR PROGRAMMING
FORTRAN CODE	OPTIMIZATION
LARGE-SCALE OPTIMIZATION	PROJECTED LAGRANGIAN
NONLINEAR CONSTRAINTS	SPARSE MATRIX

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE ATTACHED

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

408765

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT

MINOS/AUGMENTED is a general purpose nonlinear programming system, designed to solve large-scale optimization problems involving sparse linear and nonlinear constraints. Any nonlinear functions appearing in the objective or the constraints must be continuous and smooth. Users specify these functions and their gradients using two Fortran subroutines. The remaining constraint information is specified in standard MPS format, as for regular linear programming models.

MINOS/AUGMENTED (alias MINOS Version 4.0) employs a projected augmented Lagrangian algorithm to solve problems with nonlinear constraints. This involves a sequence of sparse, linearly constrained subproblems, which are solved by a reduced-gradient algorithm as implemented in the earlier version of MINOS.

This manual supplements Report SOL 77-9, the MINOS User's Guide.

DATE
ILME