


MICROCOPY RESOLUTION TEST CHART mational bureau of standards-1963-A


ARD $18424.14-\mathrm{mm}$


TECHNICAL REPORT SOL 83-20
December 1983


Department of Operations Research Stanford University Stanford, CA 94305

## FILE COPY

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

## MINOS 5.0 USER'S GUIDE

## by

Bruce A. Murtagh ${ }^{\dagger}$ and Michael A. Saunders

## TECHNICAL REPORT SOL 83-20

December 1983

Copyright © 1983 Stanford University

${ }^{\dagger}$ Department of Industrial Engineering, The University of New Soulh Wales, Kensington, N.S.W., Australia 2033.

Research and reproduction of this report were supported by the Department of Energy Contract DE-AM03-76SF00326, PA No. DE-AT03-76ER72018; National Science Foundation Grants MCS7928009 and ECS-8012974; the Office of Naval Research Contract N00014-75-C-0267; and the Army Research Office Contract DAAG29-81-K-0156.
Any opinions, findings, and conclusions or recommendations expressed in this publicalion are those of the authors and do not neccssarily reflect the views of the above aponsors.

Reproduction in whole or in part is permitted for any purposes of the United States Government.

## PREFACE

Since the middle of 1980 , approximately 150 academic and rescarch institutions around the world have installed MINOS/AUGMENTED, the predecessor of the present system. About 30 further installations exist in private industry. With enquiries continuing to arrive almost daily, the need for a combined linear and nonlinear programming system is apparent in both environments. To date, many users have been able to develop substantial nonlinear models and have come to be fairly confident that the Optimal Solution message actually means what it says. Certainly, other less joyful exit messages will often have greeted eager eyes. These serve to emphasize that model building remains an art, and that nonlincar programs can be arbitrarily difficult to solve. Nevertheless, the success rate has been high, and the positive response from users with diverse applications has inspired us to pursue further development.

MINOS 5.0 is the resull of prolonged relinements to the same basic algorithms that were in MINOS/ $\Lambda$ UGMENTED:

- the simplex method (Dantzig, 1947, 1963),
- a quasi-Newton method (very many authors from Davidon, 1959, onward),
- the reduced-gradient method (Wolfe, 1962), and
- a projected Lagrangian method (Rubinson, 1972; Rosen and Kreuser, 1972).

From numerous potential options, it has been possible to develop these particular algorithms into a relatively harmonious whole. The resulting system permits the solution of both small and large problems in the four main areas of smooth optimization:

```
- lincar programming,
- unconstrained optimization,
- linearly constrained oplimization, and
- nonlinearly constrained optimization.
```

In rare cases, the quasi-Newion method may require excessive storage. We have chosen not to provide a nonlincar conjugate-gradient method, or a truncated linear conjugate-gradient method, for this situation. Instead, we retain the quasi-Newton method throughout, restricting il to certain subspaces where necessary. (The strategy for altering the subspaces remains experimental.)

We regret that other obvious algorithms (such as integer programming, piece-wise smooth oplimization, the dual simplex method) are still not available. Nor are ranging procedures or parametric algorithms. Sensitivity analysis is still confined to the usual interpretation of Lagrange multipliers.

As before, MINOS 5.0 is a stand-alone system that is inlended for use alongside commercial mathematical programming systems whenever such facilities are available. The systems should complement each other.

To users of MINOS/AUGMENTED, the most apparent extensions are a scaling option (for linear constraints and variables only), and the ability to cetimate some or all gradients numerically, if they are not computed by the user. On a more mundanc level, the names of the user subroutines for coinputing nonlinearitics have been changed from CALCFG and CALCON to FUnOBJ and FUNCOM, and two new parameters allow access to the workspace used by MINOS.

Internally, one of the major improvements has been the development of a new basis-handling package, which forms the foundation of LUSOL, (Gill, cl al., 1081), a sel of routines for computing and updating a sparse IN/ factorization. This package draws much from the work of lReid (1978, 1082). It replaces the $P^{4}$-based procedures in MINOS/AUGMENTEI) (Saunders, 1976) and is
substantially more efficient on problems whose basis matrices are not close to triangular. As before, column updates are performed by the method of Bartels and Golub (1969, 1971), but the implementation is more efficient and there is no severe degradation arising from large numbers of "spikes". We venture to say that LUSOL is the first truly stable basis package that has been implemented for production use.

A further vital improvement has been the development of two new linesearch procedures (Gill, et al., 1979) For finding a step length with and without the aid of derivatives. In particular they cater for function values that are somewhat "noisy"--a common practical circumstance.

From a software engineering viewpoint, the source code has been restructured to case the problems of maintenance and future development. MINOS still stands for Modular In-core Nonlincar Optimization System, and we have done our best to respect the implications of the " ${ }^{\prime}$ ". Nevertheless, MINOS 5.0 remains a parameler-driven system. It is a speeding train on a railroad that has parallel tracks and many switches but few closed circuits. Its various modules cannot be called upon in an arbitrary order. In fact, there are $\mathbf{8 0}$ parameters that can be set if necessary -these are the switching points along the railroad. Fortunately, only a handful need be set for any particular application. In most cases, the default values are appropriate for large and small problems alike.

For interactive users, a new feature is the SUMMARY file, which provides at the terminal a bricf commentary on the progress of a run. Unfortunately, a two-way conversation is not possible. The only input engendered by this feature is an occasional dive for the Break key to abort an errant run. While rarely called upon, such a facility can be crucial to the security of one's computer funds.

Throughout the development of MINOS, we have received a great deal of assistance from many kind people. Most especially, our thanks go to Philip Gill, Walter Murray and Margaret Wright, whose knowledge and advice have made much of this work possible. They are largely responsible for the linesearch procedures noted above (which are as vital to nonlinear optimization as basis factors are to linear programming), and they are authoritios on all of the algorithms employed within MINOS. Their patience has been called upon continually as other important work at SOL either languished or fell unfairly on their shoulders.

Further to basis factors, we acknowledge the pioncering work of John Reid in implementing the Markowitz-based $L U$ factorization and the Bartels-Golub update. The LUSOL procedures in MINOS 5.0 owe much to the ingenuity ennodied in his D A 05 package.

Users have naturally provided an essential guiding influence. In some cases they are algorithm developers themselves. At home, we have had constant encouragement from George Dantzig and the benefit of his modeling activity within SOL, notably on the energy-cconomic model PILOT. We thank him warmly for bringing the Systems Oplimization Laboratory into existence. We also thank I'alrick MeAllister, John Stone and Wesley Winkler for the feedback they have provided by running various versions of MINOS during their work on PILOT'. (We note that PILOT has grown to 1500 constraints and 1000 variables, and now has a quadratic objective. From our perspective, it is a nonlrivial test problem!) Likewise, Alan Manne has provided encouragement and assistance from the beginning. Two of his nonlinear economic models have been invaluable as test problems (and are included on the MINOS distribution lape). We also Lhank him and Paul Preckel for the development of procedures for solving sequences of related problems (Preckel, 1980). The main ingredients of these procedures are now an integral part of MINOS.
lirom industry, we have received immense benefit from the working relationship between SOI, and Robert Burchett of the General Electric Company (Bilectric ULility Systems Engincering Department) in Schenectady, New York. Many algorilhmic and user-oriented delaila have resulted
from his experience and from his interest in the fine points of optimization. Three years ago we did not envisage that problems involving thousands of nonlinear constraints would soon be solved successfully. Rob constantly pushed test versions of MINOS to their limits, and inspired the development of techniques to extend those limits. We thank him for his tireless contributions.

We are also grateful to \%enon Fortuna, Steven Gorelick, Marc Ilellman, Thomas McCormick, Larry Nazareth, Scott Rogers, John Rowse and John Tomlin for their helpful suggestions and/or assistance in tracking down bugs. Finally, we thank the staff of the Office of Technology Licensing and the Information Technology Services at Stanford University for undertaking the task of distributing MINOS.

Most of the software development was carried out at the Stanford Linear Accelerator Center with the aid of the Wylbur text editor and the University of Waterloo's WATFIV compiler. This User's Guide was typesct using THX* with editorial assistance from Philip Gill and Margaret Wright.

Bruce Murtagh<br>University of New South Wales<br>Michael Saunders<br>Stanford University

December, 1983


[^0]
## CONTENTS

1. INTRODUCTION ..... 1
1.1 Linear Programming ..... 1
1.2 Nonlinear Objective ..... 2
1.3 Nonlinear Constraints ..... 3
1.4 Problem Formulation ..... 5
1.5 Restrictions ..... 5
1.6 Files ..... 6
1.7 Input Data Flow ..... 7
1.8 Multiple SPECS Files ..... 8
1.9 Internal Modifications ..... 8
2. USER-WRITTEN SUBROUTINES ..... 9
2.1 Subroutine FUNOBJ ..... 9
2.2 Subroutine FUNCON ..... 11
2.3 Constant Jacobian Elements ..... 12
2.4 Subroutine MATMOD ..... 13
2.5 Subroutine MATCOL ..... 15
2.6 Matrix Data Structure ..... 15
3. THE SPECS FILE ..... 17
3.1 SPECS File Format ..... 17
3.2 SPECS File Checklist and Dcfaulta ..... 18
3.3 SPECS Filc Definitions ..... 21
4. THE MPS FILE ..... 41
4.1 The NAME Card ..... 41
4.2 The ROWS Section ..... 42
4.3 The COLUMNS Section ..... 43
4.1 The RIIS Section ..... 45
1.5 The RANGES Section ..... 45
4.6 The BOUNDS Section ..... 46
4.7 Comment Cards ..... 48
4.8 Restriclions and Extensions in MPS Format ..... 48
5. BASIS FILES ..... 49
5.1 OLD and NEW BASIS Files ..... 49
5.2 PUNCH and INSERT Files ..... 52
5.3 DUMP and LOAD Files ..... 53
5.4 Restarting Modified Problems ..... 55
6. OUTPUT ..... 57
6.1 Iteration Log ..... 57
6.2 Basis Factorization Statistics ..... 61
6.3 EXIT Conditions ..... 63
6.1 Solution Output ..... 70
6.5 SOLUTION File ..... 72
6.6 SUMMARY File ..... 73
7. SYSTEM INFORMATION ..... 75
7.1 Distribution Tape ..... 75
7.2 Source Files ..... 76
7.3 COMMON Blocks ..... 78
7.4 Machine-dependent Subroutines ..... 79
7.5 Subroutine Structure ..... 82
7.6 Test Problems ..... 83
8. EXAMPLES ..... 85
8.1 Linear Programming ..... 86
8.2 Unconstrained Optimization ..... 88
8.3 Linearly Constrained Optimization ..... 90
8.4 Nonlinearly Constrained Optimisation ..... 94
8.5 Use of Subroutine MATMOD ..... 109
8.6 Things to Remember ..... 112
REFERENCES ..... 113
INDEX ..... 115

## 1. INTRODUCTION

MINOS is a Fortran-based computer system designed to solve large-scale optimization problems expressed in the following standard form:

$$
\begin{array}{cc}
\underset{x, y}{\operatorname{minimize}} & F(x)+c^{T} x+d^{T} y \\
\text { subject to } & f(x)+A_{1} y=b_{1} \\
& A_{2} x+A_{3} y=b_{2} \\
& l \leq\binom{ x}{y} \leq u \tag{4}
\end{array}
$$

where the vectors $c, d, b_{1}, b_{2}, l, u$ and the matrices $A_{1}, A_{2}, A_{3}$ are constant, $F(x)$ is a smooth scalar function, and $f(x)$ is a vector of smooth functions $\left\{f^{i}(x)\right\}$. Ideally the first derivatives (gradients) of $\boldsymbol{F}(x)$ and $f^{i}(x)$ should be known and coded by the user. (If only some gradients are known, MINOS will estimate the missing ones using finite differences.)

The $n_{1}$ components of $x$ are called the nonlincar variables, and the $n_{2}$ components of $y$ are the linear variables. Similarly, the $m_{1}$ equations (2) are called the nonlinear constraints, and the $\boldsymbol{m}_{\mathbf{2}}$ equations (3) are the linear constraints. Equations (2) and (3) together are called the general constraints. We define $m=m_{1}+m_{2}$ and $n=n_{1}+n_{2}$.

The constraints (4) specify upper and lower bounds on all variables. These are fundamental to many problem formulations and are treated specially by the solution algorithms in MINOS. Some of the components of $l$ and $u$ may be $-\infty$ or $+\infty$ if desired.

Similar bounds may be defined for the general constraints (2), (3). These constraints may therefore be thought of as taking the form

$$
\begin{aligned}
& l_{1} \leq f(x)+A_{1} y \leq u_{1} \\
& \boldsymbol{l}_{2} \leq A_{2} x+A_{3} y \leq u_{2}
\end{aligned}
$$

though for historical reasons the bounds are specified in terms of a right-hand side $b_{i}$ and a range $\boldsymbol{u}_{\boldsymbol{i}}-\boldsymbol{l}_{\boldsymbol{i}}$.

In the following sections we introduce some of the terminology required, and give an overview of the algorithms used in MINOS and the main system features.

### 1.1 Linear Programming

If the functions $F(x)$ and $f(x)$ are absent, the problem becomes a linear program. Since there is no need to distinguish between linear and nonlincar variables, we preier to use $x$ rather than $y$. It is also convenient computationally to convert all general constraints into equalities, with the only inequalities being simple bounds on the variables. Thus, we will write linear programs in the form

$$
\underset{x, s}{\operatorname{minimise}} c^{T} x \quad \text { subject to } \quad A x+I s=0, \quad l \leq\binom{ x}{s} \leq u
$$

where the elements of $x$ are called structural variables (or column variables) and $s$ is a set of slack variables (called logical variables by some nuthors). The bounds $l$ and $u$ are suitably redefined.

MINOS solves linear programs using a reliable implementation of the primal simplex method (Dantzig, 1963). The simplex method partitions the constraints $A x+I s=0$ into the form

$$
B x_{B}+N x_{N}=0
$$

where the basis matrix $B$ is square and nonsingular. The elements of $x_{B}$ and $x_{N}$ are called the basic and nonbasic variables respectively; they are a permutation of the elements of $x$ and $s$. At any given stage, each nonbasic variable is equal to its upper or lower bound, and the basic variables take on whatever values are needed to satisfy the general constraints. (Clearly they may be computed by solving the linear equation $B x_{B}=-N x_{N}$.) It can be shown that if an optimal solution to a linear program exists, then it has this form. The simplex method reaches such a solution by performing a sequence of iterations, in which one column of $B$ is replaced by one column of $N$ (and vice versa), until no such interchange can be found that will reduce the value of $c^{T} x$.

If the components of $x_{B}$ do not satisfy their upper and lower bounds, we say that the current point is infeasible. In this case, the simplex method first aims to reduce the sum of infeasibilities to zero.

MINOS maintains a sparse $L U$ factorization of the basis matrix $B$, using a Markowitz ordering scheme and Bartels-Golub updates, as implemented in the LUSOL package of Gill, Murray, Saunders and Wright (1984). (For a description of the concepts involved, see Reid, 1976, 1982.) The basis factorization is central to the efficient handling of sparse linear and nonlinear constraints.

### 1.2 Nonlinear Objective

When nonlinearities are confined to the term $F(x)$ in the objective function, the problem is a linearly constrained nonlinear program. MINOS solves such problems using a reduced-gradient algorithm (Wolfe, 1962) in conjunction with a quasi-Newton algorithm (Davidon, 1959). The implementation follows that described in Murtagh and Saunders (1978).

In this case, the constraints $A x+I s=0$ arc partitioned into the form

$$
B x_{B}+S x_{s}+N x_{N}=0
$$

where $x_{s}$ is a set of superbasic variables. At a solution, the basic and superbasic variables will lie somewhere between their bounds, while the nonbasic variables will again be equal to one of their bounds. In broad terms, the number of superbasic variables (the number of columns in $S$ ) is a measure of how nonlinear the problem is. Let this number be s. (The context will always distinguish s from the vector of slack variables.) In many practical cases we have found that e remains reasonably small, say 200 or less, regardless of the sire of the problem.

In the reduced-gradient, algorithm, $x_{s}$ is regarded as a set of independent variables that are frec to move in any desirable direction, namely one that will improve the value of the objective function (or reduce the sum of infeasibilities). The basic variables can then be adjusted in order to continue satisfying the linear constraints.

If it appears that no improvement can be made with the current definition of $B, S$ and $N$, some of the nonbasic variables are selected to be added to $S$, and the process is repeated with an increased value of s. At all stages, if a basic or superbasic variable encounters one of its bounds, that variable is made nonbasic and the value of $s$ is reduced by one.

Users familiar with linear programs may interpret the simplex method as being exactly the above process, with soscillating between 0 and 1. (Later, one step of the simplex method or the reduced-gradient method will be called a minor iteration.)

A certain operator $Z$ will frequently be useful for descriptive purposes. In the reducedgradient algorithm it takes the form

$$
Z=\left(\begin{array}{c}
-B^{-1} S \\
I \\
0
\end{array}\right)
$$

though it is never computed explicitly. Since it has full column rank and satisfies ( $B S N$ ) $Z=0$, we say that $Z$ spans the null space of the constraint matrix $\left(\begin{array}{ll}A & I\end{array}\right)$. Given an $L U$ factorization of the basis matrix $B, Z$ allows us to work within a region defined by the linear conslraints.

An important part of MINOS is a stable implementation of the quasi-Newton algorithm for optimizing the superbasic variables. This can achicve superlinear convergence within each relevant subspace (defined by the current $B, S$ and $N$ ). It obtains a search direction $p_{s}$ for the superbasic variables by solving a system of the form

$$
R^{T} R p_{s}=-Z^{T} \mathbf{g}
$$

where $g$ is the gradient of $F(x), Z^{T} g$ is the reduced gradient, and $R$ is a dense upper triangular matrix that is updated in various ways in order to approximate the reduced Hessian according to $R^{T} R \approx Z^{T} I I Z$, where $H$ is the matrix of second derivatives of $F(x)$ (i.e., the Hessian).

Once $p_{s}$ is available, the search direction for all variables is defined by $p=Z p_{s}$. A line search is then performed to find an approximate solution to the one-dimensional problem

$$
\underset{\alpha}{\operatorname{minimize}} F(x+\alpha p) \quad \text { subject to } \quad 0 \leq \alpha \leq \alpha_{\max }
$$

where $\alpha_{\text {max }}$ is determined by the bounds on the variables. Another important part of MINOS is the step-length procedure used in the line search to determine the step-length $\alpha$. Two different. procedures are used, depending on whether all gradients are known. (Sce Gill, Murray, Saunders and Wright, 1979.) Interested users can influence the amount of work involved by setting a parameter called the LINESEARCH TOLERANCE.

Normally, the objective function $F(x)$ will never be evaluated at a point $x$ unless that point is feasible, i.e., it satisfies the linear constraints and the bounds on the variables. Facilitics are provided to check the calculation of gradient elements, and normally the check is performed at the first feasible point. However, users may request that the check be performed at the very beginning of a run, in which case $x$ may not be feasible.

For details of the matters mentioned here and many other essential aspects of numerical optimization, see Gill, Murray and Wright (1981).

### 1.3 Nonlinear Constraints

When the probletn contains nonlinear constraints, MINOS uses a projected augmented Lagrangian algorithm, based on a method due to Robinson (1972); see Murtagh and Saunders (1982). MINOS treats linear constraints and bounds specially, but the nonlincar constraints may not be satisfied until an optimal point is reached. Thus, $f(x)$ and its gradients (the Jacobian matrix $J(x)=$ [ $\left.\partial f^{i}(x) / \partial x_{j}\right]$ ) may need to be defined outside the region of interest.

In fact, the constraint functions will almost never be evaluated unicss the linear constraints and bounds arc satisfied. The principal exception is at the very first point $x_{0}$, which may optionally be specified by the user. The vector $f(x)$ and its Jacobian will be evaluated at $x_{0}$ regardless of feasibility. This matter must be borne in mind during the formulation of a nonlinear program. The main point to remember is that the nonlinear constraints may be violated during the solution process.

The nature of the solution process 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 lincarized versions of the nonlinear constraints. This just means that $f(x)$ in equation (2) is replaced by $L f$, its linear approximation at the current point. We shall write this approximation as

$$
\tilde{f}\left(x, x_{k}\right)=f\left(x_{k}\right)+J\left(x_{k}\right)\left(x-x_{k}\right)
$$

or more briefly

$$
\begin{equation*}
\tilde{f}=f_{k}+J_{k}\left(x-x_{\dot{k}}\right) \tag{5}
\end{equation*}
$$

where $x_{k}$ is the estimate of the nonlinear variables at the start of the $k$-th major iteration. The subproblem to be solved takes the form

$$
\left.\begin{array}{rl}
\underset{x, y}{\operatorname{minimize}} \quad F(x)+c^{T} x+d^{T} y-\lambda_{k}^{T}(f-\tilde{f})+\frac{1}{2} \rho(f-\tilde{f})^{T}(f-\tilde{f}) \\
\text { subject to } \quad \tilde{f}+A_{1} y & =b_{1} \\
A_{2} x+A_{3} y & =b_{2} \\
l & \leq\binom{ x}{y} \tag{9}
\end{array}\right) \leq u .
$$

The objective function (6) is called an augmented Lagrangian. The vector $\lambda_{k}$ is an estimate of $\lambda$, the Lagrange multiplicrs for the nonlinear constraints. The scalar $\rho$ is a penalty parameter, and the term involving $\rho$ is a modified quadratic penalty function.

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

$$
\left(\begin{array}{ll}
J_{k} & A_{1}  \tag{10}\\
A_{2} & A_{3}
\end{array}\right)\binom{x}{y}+\left(\begin{array}{ll}
I & 0 \\
0 & I
\end{array}\right)\binom{s_{1}}{s_{2}}=\binom{J_{k} x_{k}-f_{k}}{0}
$$

MINOS uses the reduced-gradient algorithm to minimize (6) subject to (10), with the original bounds on $x$ and $y$, and suitable bounds on the slack variables $s_{1}$ and $s_{2}$. The Jacobian $J_{k}$ is treated as a sparse matrix, the same as the matrices $\boldsymbol{A}_{i}$.

Unfortunately, there is no guarantee that the algorithrn just described will converge from an arbitrary starting point. The concerned user can influence the likelihood of convergence in several ways:

1. By specifying $x_{0}$ as carefully as possible.
2. By including sensible upper and lower bounds on all variables.
3. By specifying a PENALTY PARANETER $\rho$ that is higher than the default value, if the problem is suspected of being highly nonlinear.
4. By specifying a DAMPING PARAMETER that is lower than the default value, again if the problem is highly nonlinear.
In rare cases it may be safe to use $\lambda_{k}=0$ and $\rho=0$ for all subproblems, by specifying LaGRANGIAN $=$ NO. However, convergence is much more likely with the default setling, LAGRANGIAN = YES. The initial estimate of the lagrange multipliers is then $\lambda_{0}=0$, but for later subproblems, $\lambda_{k}$ is taken to be the Lagrange multipliers associated with the (linearized) nonlinear constraints at the end of the previous major itcration.

The penalty parameter is initially $100.0 / m_{1}$ by default, and it is reduced in stages for later subproblems when it appears that the sequence $\left\{x_{k}, \lambda_{k}\right\}$ is converging. In many cases it is safe to specify $\rho=0$ from the beginning, particularly if the problem is only mildly nonlinear. This may improve the overall efficiency.

### 1.4 Problem Formulation

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

$$
\frac{K_{1}}{d_{1}^{4.814}}+\frac{K_{2}}{d_{2}^{4.814}}+\cdots \leq P_{T}^{2}-P_{0}^{2}
$$

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 move nonlinearities from the objective function into the constraints. Thus, we would not replace minimize $F(x)$ by

$$
\operatorname{minimize} z \text { subject to } F(x)-z=0 .
$$

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. In general we suggest the range 1.0 to $\mathbf{1 0 . 0}$. When conflicts arise, one should 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. MINOS has an option to scale the linear constraints and variables automatically.

Finally, upper and lower bounds on the variables (and on the constraints) are extremely useful for 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 safoty when such singularities exist, the initial point $x_{0}$ discussed above should lie within the bounds.

### 1.5 Restrictions

MINOS is designed to find solutions that are locally optimal. The nonlinear functions in a problem must be smooth (i.e., their first derivatives must exist). The functions need not be separable. Integer restrictions cannot be imposed directly.

A certain region is delined 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 uses one large array of main storage for most of its workspace. The length of this array may need to be adjusted to suit a particular problem, but otherwise the implementation places no fixed limitation on the size of a problem or on its shape (many constraints and relatively few variables, or vice versa). In general, the limiting factor will be the amount of inain storage
available on a particular machine, and the amount of computation time that one's budget can stand.

Some a priori knowledge of a particular application will usually indicate whether the solution procedure is likely to be efficient. An important quantity is $m=m_{1}+m_{2}$, the total number of general constraints in (2) and (3). We note that $m \leq 100$ is considered "small", $m=1000$ or 2000 is "medium", and $m \geq 5000$ would be "large". On machines that use 16 -bit integers (INTEGER*2 on IBM and DEC VAX systems), the normal implementation of MINOS requires that $m \leq 32767$.

The amount of workspace required by MINOS is roughly 100 m words, where one "word" is the relevant storage unit for the floating-point arithmetic being used (REAL*8 on IBM and DEC VAX, REAL on Burroughs and most CDC machines). On IBM and VAX systems, this means about 800 m bytes for workspace. A further 300 K bytes, approximately, are needed for the program itself, along with buffer space for several files.

Another important quantity is $n=n_{1}+n_{2}$, the total number of variables in $x$ and $y$. For nonlinear problems, if $m_{1}$ and $n_{1}$ are small compared to $m$ and $n$, the total storage required should not be much greater than just described. If $n_{1}$ is "large" (say $n_{1} \geq 200$ ), the amount of storage required may or may not be substantial, depending on whether $F(x)$ or $f(x)$ are highly nonlinear or not.

In this context, the efficiency of MINOS depends on $s$, the number of superbasic variables. Recall that $m+s$ variables lie between their upper and lower bounds, where $s$ is zero for purely linear problems. We know that $s$ need never be larger than $n_{1}+1$. In practice, $s$ is often very much less than this upper limit.

In the quasi-Newton algorithm, the dense triangular matrix $R$ has dimension $s$ and requires about $\frac{1}{2} s^{2}$ words of storage. If it seems likely that $s$ will be very large, some aggregation or reformulation of the problem should be considered.

### 1.6 Files

MINOS operates primarily within central memory, and is well suited to a virtual storage environment. Certain disk files are accessed as follows.

| Input file | Status | Record Length (characters) |
| :--- | :---: | :---: |
| READ file | see below |  |
| SPECS file | required | 80 |
| MPS file | required | 81 |
| BASIS files | optional | 80 |
| Output file | Status | Record Length (characters) |
| SCRATCH file | required | 8 |
| PRINT file | required | 129 |
| SUMMARY file | optional | 80 |
| BASIS files | optional | 80 |
| SOLUTION file | optional | 111 |

Fixed-length, blocked records may be used in all cases, and the files are always accessed sequentially. The logical record length must be at least that shown. For efficiency, the physical block size should be several hundred characters in most cases. Note that the logical record length for the SCRATCH file is unusually small. Each record will contain an 8 -character name for a constraint or variable, and there will be $m+n$ such names.

Unit numbers for the READ, SPECS, PRINT and SCRATCII files are defined at compile time; typically they will be $5,5,6$ and 8 , but they may depend on the installation. The remaining unit numbers are specified at run time in the SPECS file.

Unit numbers for the READ, PRINT and SUMMARY files are stored in the following COMMOM block:

COMMON /MAFILE/ IREAD.IPRINT,ISURM
It may be convenient to reference these in the user subroutines FUNOBJ, FUNCON and MATMOD.
System Note: The READ file is not used explicitly by MINOS, but its unit number is used to test if a file should be rewound. (Thus, input files are subject to a Fortran REwIND as long as they are not the same as the READ file.) The PRINT file is used frequently. Other output files are rewound if they are not the same as the PRINT file.

### 1.7 Input Data Flow

Some or all of the following items are supplied by the user:

- Subroutine FUNOBJ
- Subroutine FUNCON
- Subroutine Matnod
- A SPECS file
- An MPS file
- A BASIS file
- Data read by FUNCON on its first entry
- Data read by FUNOBJ on its first entry
- Data read by FUNCON on its last entry
- Data read by funobj on its last entry

The order of the files and data is important if all are stored in the same input stream.
Subroutines FUNOBJ and FUNCON define the nonlinear objective and constraint functions respectively (if any); they are not needed if the functions are purely linear and are defined in the MPS file.

Subroutine MATMOD is occasionally needed, for applications involving a sequence of closely related problems.

The SPECS file defines various run-time parameters (ITERATION LIMIT, save frequency, etc.). Its file number is defined at compile time. It will normally be the first data set in the system card input stream.

The MPS file specifies names for the constraints and variables, and defines all the linear constraints and bounds. It may follow the SPECS file in the card input stream, but will often reside in a file of its own (as specified in the SPECS file). The data format is similar to that used in commercial mathematical programming systems (hence the name). The format has been generalized slightly for nonlinear problems.

If desired, a BASIS file may be loaded at the beginning of a run. This will normally have been saved at the end of an earlier run. Three kinds of basis file are available; they are used to restart the solution of a problem that was interrupted, or to provide a good starting point for some slightly modified problem.

### 1.8 Multiple SPECS Files

One or more problems may be processed during a run. The parameters for a particular problem are delimited by BEGIN and END in the SPECS file. While scanning for the keyword BEGIN, MINOS recognizes the keywords SKIP and ENDRUN. Thus in the following example:

```
BEGIN CASE 1
END CASE 1
SKIP CASE 2
END CASE 2
BEGIN CASE }
END CASE 3
ENDRUN
BEGIN CASE 4
END CASE 4
```

only the first and third problem will be processed.

### 1.9 Internal Modifications

A sequence of closely related problems may be specified within a single SPECS file, via the CYCLE parameter; for example,

BEGIN CYCLING EXAMPLE
CYCLE LIMIT 10
END EXAMPLE
indicates that up to 10 problems are to be processed. This is intended for cases where the solution of one problem $P_{k}$ is needed to dcfine the next problem $P_{k+1}$.

The actual method for defining the next problem in a cycle depends on the application. Sometimes it can be done by changing the output from the function subroutines FUNOBJ and/or FUNCON. Alternatively, the user may provide a third subroutine MATMOD to perform some modifications to the problem data. MATMDD is called by MINOS at the beginning of every cycle except the first.

If necessary, the number of linear variables can be increased when a problem $P_{k+1}$ is defined. We think of this as adding new columns to $P_{k}$. The new columns are not included in the MPS file, and their sparsity pattern need not be known until $P_{k}$ has been solved. Instead, an appropriate number of phantom colunns and phantom elements are defined in the SPECS file (to reserve a pool of storage), and the user's subroutine MATMOD generates each new column by calling the MINOS subroutine MATCOL.

## 2. USER-WRITTEN SUBROUTINES

To solve a purely linear problem, only a SPECS file and an MPS file (and possibly a BASIS file) need be supplied.

For nonlinear problems, one must also provide some appropriate Fortran code. Nonlinearities in the objective function are defined by subroutine FUNOBJ. Those in the constraints are defined separately by subroutine FUNCON. On every entry except perhaps the last, these subroutines must return appropriate function values $F$. Wherever possible, they should also return all gradient components in the array G. This provides maximum reliability and corresponds to the default setting, DERIVATIVE LEVEL $=3$.

In practice it is often convenient not to code gradients. MINOS is able to estimate gradients by finite differences, by making a call to FUNOBJ or FUNCON for each variable $x_{j}$ whose partial derivatives need to be estimated. However, this reduces the reliability of the optimization algorithms, and it can be very expensive if there are many such variables $x_{j}$.

As a compromise, MINOS allows you to code as many gradients as you like. This option is implemented as follows: just before a function routine is called, each clement of the gradient array $G$ is initialized to a specific value. On exit, any clement retaining that value must be estimated by finite differences.

Some rules of thumb follow:

1. For maximum simplicity and reliability, compute $F$ and all components of $G$.
2. If not all gradients are known, compute as many of them as you can. (It often happens that some of them are constant or even zero.)
3. If some gradients are known (but not all), it may be convenient to compute them each time the function routines are called, even though they will be ignored if MODE $=2$.
4. If the known gradients are expensive to compute, use the parameter MODE to avoid computing them on certain entries.
5. While the function routines are being developed, use the VERIFY parameter to check the computation of any gradient elements that are supposedly known.

### 2.1 Subroutine FUNO8J

This subroutine is provided by the user to calculate the objective function $\boldsymbol{F}(\boldsymbol{x})$ and as much of its gradient $g(x)$ as possible. (It is not needed if the objeclive function is entirely linear.)

## Specification:

SUBROUTINE FUNOBJ ( MODE, N, X, F, G, NSTATE, NPROB, Z, MWCORE )
IMPLICIT REAL*8(A-H, O-Z)
DIMENSION $\quad \mathbf{X}(\mathrm{N}), \mathrm{G}(\mathrm{N}), \mathrm{Z}$ (NWCORR)
(The IMPLICIT statement should not be used on machines for which single-precision floatingpoint is adcquate; e.g., Burroughs and CDC.)

Parameters:
MODE (Input) This parameter can be ignored if DERIVATIVE LEVEL $=1$ or 3 (i.e., if all elements of $G$ are computed). In this case, MODE will always have the value 2.
Otherwise, you must specify DERIVATIVE LEVEL $=0$ or 2 in the SPECS file to indicate that FUNOBJ will not compute all of G. MINOS will then call FUNOBJ sometimes with MODE $=2$ and sometimes with MODE $=0$. You may test MODE to decide what to do:

If $\mathrm{MODE}=2$, compute $F$ and as many components of $\mathbf{G}$ as possible.
If $\mathrm{MODE}=\mathbf{0}$, compute $F$ but set $G$ only if you wish. (On return, the contents of $\mathbf{G}$ will be ignored.)
(Output) If for some reason you wish to terminate solution of the current problem, set MODE to a negative value, e.g., -1 .
$N \quad$ (Input) The number of variables involved in $\boldsymbol{F}(x)$. These must be the first $N$ variables in the problem.
X(*) (Input) An array of dimension $N$ containing the current values of the nonlinear variables $x$.
$F$ (Output) The computed value of the objective function $F(x)$.
$G(*) \quad$ (Output) The computed gradient vector $g(x)$. In general, $G(j)$ should be set to the partial derivative $\partial F / \partial x_{j}$ for as many $j$ as possible (except perhaps if MODE $=0$-see above).
NSTATE (Input) If NSTATE $=0$, there is nothing special about the current call to FUNOBJ.
If NSTATE $=1$, MINOS is calling your subroutine for the first time. Some data may need to be input or computed and saved in local or COMmON storage. Note that if there are nonlinear constraints, the first call to FUNCON will occur before the first call to FUNOBJ.

If NSTATE $\geq 2$, MINOS is calling your subroutine for the last time. You may wish to perform some additional computation on the final solution. (If CYCLE LIMIT is specified, this call occurs at the end of each cycle.) Note again that if there are nonlincar constraints, the last call to FUNCON will occur before the last call to FUNOBJ.

In general, the last call is made with NSTATE $=2+$ IERR, where IERR indicates the status of the final solution. In particular, if NSTATE $=2$, the curront. $X$ is optimal; if NSTATE $=3$, the problem appears to be infeasible; if NSTATE $=4$, the problem appears to be unbounded; and if NSTATE $=5$, the iterations limit was reached. In some cases, the solution may be nearly optimal if NSTATE $=11$; this value occurs if the linesearch procedure was unable to find an improved point.
If the nonlinear functions are expensive to evaluate, it may be desirable to do nothing on the last call, by including a statement of the form IF (NSTATE .GE. 2) RETURN at the start of the subroutine.
NPROB (Input) An integer that can be set by a card of the form PROBLEM NUMBER $n$ in the SPECS file.

Z(*) (Input) The primary work array used by MINOS. In certain applications it may be desirable to access parts of this array, using various COMMON blocks to pinpoint the required locations. (For example, the dual variables are stored in $Z$ (LPI) onward, where LPI is the first integer in the COMMON block MSLOC.) Otherwise, 2 and NINCORE can be ignored.
NWCORE (Input) The dimension of 2.

### 2.2 Subroutine FUNCON

This subroutine is provided by the user to compute the nonlinear constraint functions $f(x)$ and as many of their gradients as possible. (It is not needed if the constraints are entirely linear.) Note that the gradients of the vector $f(x)$ define the Jacobian matrix $J(x)$. The $j$-th column of $J(x)$ is the vector $\partial f / \partial x_{j}$.

FUNCON may be coded in two different ways, depending on the method used for storing the Jacobian, as specified in the SPECS file.

$$
\text { JACOBIAN }=\text { DENSE }
$$

## Specification:

SUBROUTINE FUNCON( MODE, M. N. NJAC. X, F, G, MSTATE, KPROB, Z, NWCORE )
IMPLICIT REAL* 8 (A-H, O-Z)
DIMENSION $\quad X(N), F(N), G(M, N), Z(N W C O R E)$

## Parameters:

MODE (Input) This parameter can be ignored if DERIVATIVE LEVEL $=2$ or 3 (i.e., if all elements of $\mathbf{G}$ are computed). In this case, MODE will always have the value 2.

Otherwise, you must specify DERIVATIVE LEVEL $=0$ or 1 in the SPECS file to indicate that FUNCON will not compute all of G. You may then test MODE to decide what to do:

If $M O D E=2$, compute $F$ and as many components of $G$ as possible.
If MODE $=0$, compute $F$ but set $G$ only if you wish. (On return, the contents of $\mathbf{G}$ will be ignored.)
(Output) If for some reason you wish to terminate solution of the current problem, set MODE to a negative value, e.g., -1 .

4 (Input) The number of nonlinear constraints (not counting the objective function). These must be the first $M$ constraints in the problem.
$N \quad$ (Input) The number of variables involved in $f(x)$. These must be the first $N$ variables in the problem.

NJAC (Input) The value $\mathbf{M} * N$.
$X$ (*) (Input) An array of dimension I containing the current values of the nonlinear variables $\boldsymbol{x}$.
$P(*) \quad$ (Output) The computed values of the functions in the constraint vector $f(x)$.
$G(*, *)$ (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 $G$ (except perhaps if MODE $=0$ sce above). Equivalently, the gradient of the $i$-th constraint should be stored in the $i$-th row of $\mathbf{C}$.

The other parameters are the same as for subroutine FUNOBJ.

## JACOBIAN = SPARSE

Specification:

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

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 (the first $M$ rows in the ROWS section and the first $N$ columns in the COLUMNS section).

Usually NJAC will be less than $\mathbf{M} * \mathrm{~N}$. The actual value of NJAC may not be of any use when coding FUNCON, but in all cases, any expression involving $G(l)$ should have the subscript $l$ between 1 and NJAC.
(Output) The computed elennents of the Jacobian matrix (except perhaps if MODE $=0$ see previous page). These elements must be stored into $G$ in exactly the same positions as implied by the MPS file. There is no internal check for consistency (except indirectly via the VERIFY parameter), so great care is essential.

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

### 2.3 Constant Jacobian Elements

If all constraint gradients (Jacobian elements) are known (DERIVATIVE LEVEL = 2 or 3), any constant elements may be specified in the MPS file if desired. An element of $G$ that is not computed in FUNCON will retain the value implied by the MPS lile. (The value is taken to be zero if not given explicitly in the MPS file.)

This feature is useful when JACOBIAN = DEASE and many Jacobian elements are identically zero. Such elements need not be specified in the MPS file, nor set in FUNCON.

Note that constant nonzero elements do affect $F$. Thus, if $J_{i j}$ is defined in the MPS file and is constant, the array element $G(i, j)$ need not be set in FUNCON, but the value $G(i, j) * X(j)$ must be added to $\mathrm{F}(\mathrm{i})$.

When Jacobian $=$ sparse, constant Jacobian elements will normally not be listed in the MPS file unless they are nonzero. If the correct value is entered in the MPS file, the corresponding element $G(l)$ need not be reassigned, but a term of the form $G(l) * X(j)$ must be added to one of the elements of $\mathbf{F}$. (This feature allows a matrix generator to oulput constant data to the MPS filc; FUNCON does nol need to know that data at compile time, but can use il at run time to compute the elements of $P$.)

Remember, if derivative level < 2 , unassigned elements of $\mathbf{G}$ are not treated as constant; they are estimated by finite diferences, at significant expense.

### 2.4 Subroutine MATMOD

This subroutine is called at the start of every cycle except the first. If CYCLE LIMIT $k$ is specified with $k>1$, you may wish to provide your own MATMOD to make certain changes to the problem data. MINOS will make a flying start on the modified problem, retaining the current set of basic, superbasic and nonbasic variables, the reduced Ilcssian approximation $R$, and the Lagrange multiplier estimates $\lambda_{k}$ for any nonlinear constraints. (The $W$ factorization of the basis is not retained; it will be recomputed.)

Specification:
SUBROUTINE MATMOD(.NCYCLE, NPROB, FIMISH,


## Parameters:

NCYCLE (Input) The number of the cycle that has just terminated.
NPROB (Input) An integer that can be set by a card of the form PROBLEM NUNBER $n$ in the SPECS file.

FINISH (Input) On entry, FINISH = .FALSR.
(Output) On exit, if you wish the cycles to be terminated (e.g., if some convergence criterion has been satisfied), set FINISH = . TRUZ.

4 (Input) The number of rows in the constraint matrix.
M (Input) The number of variables, excluding slacks.
NB (Input) The number of variables, including slacks. (NB $=\mathbf{N}+\boldsymbol{M}$; it is the length of many arrays including BL and EU. The name is short for Number of Bounds.)

IE (Input) The number of elements in the constraint matrix (used only to dimension $A$ and HA).

MKA (Input) MKA $=\mathbf{N}+1$ (used to dimension KA ).
HS (Input) The number of superbasic variables.
MSCL (Input) If MSCL $=1$, the constraint matrix has not been scaled; there is only one element in the array ASCALE and it is undefined.

Otherwise, NSCL $=$ NB and the constraint matrix has been scalcd. An original element $a_{i j}$ is now stored in $A$ as $a_{i j} \times$ ASCALE $(N+i) / \operatorname{ASCALE}(j)$. Similarly, an original bound $l_{j}$ on a column or slack ( $j=1$ to NB) is now stored in BL ( $j$ ) as $l_{j} \times \operatorname{ASCALE}(j)$. (Likewise for upper bounds.) Any new or modified elements must be treated accordingly.
$A$ (*) (Input, output) The current elements of the constraint matrix, stored column-wise. They may be altered as desired. Additional entries in A, HA and KA associated with "phantom columns" must be set by calling subroutine MATCOL.

HA (*) (Input, output) The row indices associated with A.
KA(*) (Input, output) $\mathrm{KA}(j)$ points to the start of column $j$ in the arrays A (*), HA (*).
BL (*) (Input, output) The lower bounds on all column and slack variables, in that order. They may be altered as desired.
BU(*) (Input, output) The upper bounds on all variables, including slacks. They may be altered as desired.

ASCALE (*) (Input) The set of scale factors for columns and rows, in that order (if NSCL $>1$ ).
HS (*) (Input) The state vector for all variables.
HS $(\boldsymbol{j})=0$ if variable $\boldsymbol{j}$ is nonbasic at its lower bound;
HS $(j)=1$ if variable $j$ is nonbasic at its upper bound;
HS $(j)=2$ if variable $j$ is superbasic;
HS $(j)=3$ if variable $j$ is basic.
ID1 (*) (Input) The first half of the names of the columns and rows, in that order, in A4 format. (Sometimes it may be useful to determine the index of a column or row by searching ID1 and ID2.)
ID2 (*) (Input) The second half of the names of the columins and rows, in $\mathbf{A} 4$ format.
$X(*) \quad$ (Input) The numerical values of all columns and slack variables.
PI (*) (Input) The numerical valucs of the dual variables. The first $m_{1}$ components will be the current estimates of $\lambda$, the Lagrange multipliers for the nonlinear constraints.
$Z(\$) \quad$ (Input) The primary work array used by MINOS. In certain applications it may be desirable to access parts of this array, using various common blocks to pinpoint the required locations.
WiWCORE (Input) The dimension of 2 .

### 2.5 Subroutine MATCOL

If PHANTOM COLUMNS $c$ and PHANTOM ELEMENTS $e$ are defined in the SPLCS file (along with CYCLE LIMIT $k$ ), this subroutine may be called by MATMOD up to $c$ times throughout cycles 2 through $k$. The aim is to turn at most $c$ "phantom columns" into normal columns containing a total of at most $c$ nonzero elements. MATMOD must provide an array COL (*) and a zero tolerance ZTOL for each call. The significant elements of COL will be packed into the matrix data structure, to form a new column. The associated variable will be given the default LOWER and UPPER bounds, and a scale factor of 1.0 .

Specification:
SUBROUTINE MATCOL ( M, N. NB, NE, NKA,

```
*
    IMPLICIT
    INTEGER*2
    INTEGER
    DIMENSIOM
    A, HA, KA, BL, BU, COL, ZTOL )
    REAL*8(A-H,O-Z)
    HA(NE)
    KA(NKA)
    A(NE), BL(NB), BU(NB), COL(M)
```

Parameters:
$y$ (Input) The length of the array COL. Usually this will be $m$, the number of rows in the constraint matrix. In general, it may be anywhere in the range $1 \leq M \leq m$, if the new column is known to be zero beyond position $M$.
COL(*) (Input) The dense vector that is to become a new matrix column.
2TOL (Input) A zero tolerance for deleting negligible elements from COL when it is packed into $A$ and $H A$. On most machines, a reasonable value is $\mathrm{ZTOL}=1.0 \mathrm{E}-8$.
The other parameters come directly from MATMOD. For further details, see the CYCLE options in section 3.3 , and the example in section 8.5.

### 2.6 Matrix Data Structure

In the MINOS source code, the constraint matrix $A$ is stored column-wise in sparse format in the arrays $A, H A, K A$, as defined in the specifications of subroutine MATMOD (section 2.4). The matrix $I$ associated with the slack variables is represented implicitly. If the objective function contains linear terms $c^{T} x+d^{T} y$, then $\left(c^{T} d^{T}\right)$ is included as the IOBJ-th row of $A$ (see the COMMON block M5LOBJ below).

If there are nonlinear constraints, the lop left-hand corner of $A$ is loaded with the current Jacobian matrix at the start of each major iteration.

The following COMMON blocks contain dimensions and other items relating to the storage of A.

## COMMON /M3LEN / M .N .NB .N8CL

$m \quad m$, the number of rows in $A$, including the linear objective row (if any).
$m \quad n$, the number of columns in $A$, possibly including $c$ "phantom columns".
WB $n+m=N+M$, the total number of variables in the problem, including the slacks.
MSCL. Dither MB or 1, depending on whether SCALE has been specified or not.

|  | COMMON /M2MAPA/ NE .NKA .LA , LHA .LKA |
| :---: | :---: |
| NE | The number of nonzero elements in $A$, possibly including $e$ "phantom elements". $n+1=N+1$, the number of pointers in the array $K A$. |
| NKA |  |
| LA | The address of $A$ (*) in the work array $Z$ (*). |
| LHA | The address of HA (*) in the work array Z (*). |
| LKA | The address of KA (*) in the work array $\mathrm{Z}(*)$. |
|  | COMMON /MELEN / MAXR .MAXS .MBS .NN .NNO .NR .NX |
| MAXR | The hessian dimension. |
| MAXS | The SUPERBASICS LIMIT. |
| MBS | M+MAXS, the maximum number of basic and superbasic variables. |
| NN | $n_{1}=\max \{$ NNOBJ, NNJAC\}, the number of NONLINEAR VARIABLES. |
| NNO | $\max \{1, \mathrm{NN}\}$. |
| NR | The dimension of the array $R$ that is used to approximate the reduced Hessian, $R$. $\max \left\{\right.$ MBS $\left.^{\prime}, \mathrm{NN}\right\}$. |
|  | COMMON /M5LOBJ/ SINF ,WTOBJ ,MINIMZ,NINF .IOBJ |
| SINP | The current sum of infeasibilities. |
| WTOBJ | The scalar $w$ used in the composite objective technique. |
| MINIMZ | +1 if the objective is to be minimized; -1 if it is to be maximized. |
| NINF | The current number of infeasibilities. |
| IOBJ | The row number for the linear objective. (If IOBJ is zero, there is no such row.) |

IOBJ The row number for the linear objective. (If IOBJ is zero, there is no such row.)

COMMON /M7LEN / FOBJ .FOBJ2 .NNOBJ .NNOBJO
FOBJ The current value of the function value $F$ returned by FUNOBJ.
FOBJ2 A temporary value of FOBJ.
mNOBJ $n_{1}^{\prime}$, the number of NONLINEAR OBJECTIVE VARIABLES.
mnobjo max $\{1$, NNOBJ $\}$.

COMMON /MBLEN / MJAC .NNCON ,NNCONO,NNJAC
NJAC The number of elements in the Jacobian.
MHCOW $m_{1}$, the number of MONLIMEAR CONSTRAINTS.
MACONO max\{1, MNCOM $\}$.
majac nén the number of monlimear jacobian variables.

## 3. THE SPECS FILE

The SIPECS file sets various run-time parameters that describe the nature of the problem being solved and the manner in which a solution is to be oblained. The file consists of a sequence of card images, each of which contains a keyword and certain associaled values.

The first keyword is BEGIN and the last keyword is END. If the problem could be solved using defanlt values for all parameters, the SPECS file could consist of just those two keywords (on separate cards). Normally, however, at least some of the parameters must be specified; for example, the number of nonlinear variables if there are any.

### 3.1 SPECS File Format

Each card in the SPECS file contains a sequence of iterns in free format (they may appear anywhere in columns 1 to 72). The items are separated by spaces or cqual signs (' or ' $=$ '). Those selected from each card are:

1. The first word (the keyword). Only the first 3 characters are significant.
2. The second word (if any). Sometimes this is the keyword's associated name valuc, an 8character name. More often it qualifies the keyword, and its first 4 characters are significant.
3. The first number (if any). This may be an integer value or a real value; up to 8 characters in Fortran's I, F, E or D format.
In the following examples the significant characters are underlined:

| OBJECTIVE | PROFIT |
| :--- | :--- |
| SOLUTION FILE | $\mathbf{1 2}$ |
| ROWS | $\underline{500}$ |
| ROW TOLERANCE | 0.0001 |
| LOWER BOUND | $\frac{-1.0}{1.0 E-6}$ |
| AIJ TOL |  |

If the first character of an item is one of the following numeric characters
$1234567890+$. .
then the item is taken to be a number. The number may be from 1 to 8 contiguous numeric characters, including an E or a $D$ if need be. It is terminated by a non-numeric character such as a space.
(An exception is made for the keywords OBJECTIVE, RHS, RANGE and BOUND, which specify names to be extracted from the MPS file. For these keywords the second item is taken lo be the required name value even if it begins with a numeric character. Thus,

| AIJ TOLERANCE | .00001 |
| :--- | :--- |
| OBJECTIVE | .00001 |
| RHS | $\ldots .2 E 001$ |
| BOUND | +1000 |

are all allowed. However, names like OBJECTIVE = COST or RHS = DEMANDO2 will be more common.)

Blank cards and comments may be used to improve readability. A comment begins with an asterisk ('*') and includes all subsequent characters on the amme card; these are ignored. The ' $\#$ ' may be the first non-blank character on the card, or the first non-blank after a space or an equal sign. For example:
*

* MPS file parameters
* 

| ROWS | 1000 | * (or less) |
| :--- | :--- | :--- |
| COLUMNS | 2000 | * (or less) |
| ELEMENTS | 8000 | * (or less) |
| OBJECTIVE $=$ | PROFITO2 | * (the 2nd $N$ row) |

Scanning terminates once a number has been recognized. An asterisk is therefore not essential following a number:

WEIGHT ON OBJECTIVE $=10.0$ DURING PHASE 1

### 3.2 SPECS File Checklist and Defaults

The following example SPLCS file shows all valid keywords and their default values. The keywords are grouped according to the function they perform.

Some of the default values depend on $t$, the relative precision of the 'ine being used. The values given here correspond to double-precision arithmetic on IBM 3t • $\mathbf{3 7 0}$ systems and their successors ( $\epsilon \approx 2.22 \times 10^{-16}$ ). Similar values would apply to any machine having about 15 decimal digits of precision.

BEGIN checklist of SPECS file parameters and their default values
*

* Keywords for the MI'S file
* 

MINIMIZE * (opposite of MAXIMIZE)

| OBJECTIVE = | $?$ | * the firsi name cocountered |
| :---: | :---: | :---: |
| RHS = | ? | * the first name encountered |
| RANGE = | $?$ | * the first name encountered |
| BOUNDS = | ? | * the first narme encountered |
| ROWS | 100 | * |
| COLUMNS | 300 | * or 3*ROWS |
| ELEMENTS (or COEFFICIENTS) | 1500 | * or 5*COLUNAS |
| AIJ TOLERANCE | 1. OE-10 | * |
| LOWER BOUND | 0.0 | * |
| UPPER BOUND | 1. $\mathrm{OE}+20$ | * plus infinity |
| MPS FILE | ? | * depends on inslallation |
| LIST LIMIT | 0 | * for printing Ml'S data |
| ERROR MESSAGE LIMIT | 10 | * during MPS input |

- 
* Keywords for the simplex method
- 

CRASH OPTION 1
ITERATIONS LIMIT 300
PARTIAL PRICE 1
MULTIPLE PRICE 1
WEIGHT ON LINEAR OBJECTIVE 0.0

* all variables eligible for initial basis
* or 3*ROWS + 10*NONLINEAR VARIABLES
* or COLS/(2*ROWS) if COLS is large
* BEWARI: - not like commercial I,P
0.0 * during phase 1


```
LINESEARCH TOLERANCE 0.1 * smaller for more accurate search
SUBSPACE TOLERANCE
0.5 * affects when to PRICE
FUNCTION PRECISION
3.0E-13 * \epsilon 0.8 (almost full accuracy)
5.5E-7 * (FUNCTION PRECISION)
CENTRAL DIFFERENCE INTERVAL
6.7E-5 * (FUNCTION PRECISION)}\mp@subsup{}{}{\frac{1}{3}
*
* Further parameters for nonlinear constraints
*
JACOBIAN
```

DENSE *

```
LAGRANGIAN
MAJOR ITERATIONS
MINOR ITERATIONS
PENALTY PARAMETER
DAMPING PARAMETER
*
COMPLETION
ROW TOLERANCE
RADIUS OF CONVERGENCE
PRINT LEVEL (JFLXB)
*
* Sequences of related problems
*
CYCLE LIMIT
CYCLE PRINT
CYCLE TOLERANCE
PHANTOM COLUKNS
PHANTOM ELEMENTS
*
* Miscellaneous
*
DEBUG LEVEL. 
DEBUG LEVEL.
WORKSPACE (USER)
WORKSPACE (TOTAL)
0 *
999999
0 *
```

* SUPPRESS PARAMETER LISTING
END of SPECS file checklist


### 3.3 SPECS File Definitions

The following is an alphabetical list of recognized SPECS file keywords. A typical use of each keyword is given, along with a delinition of the quantities involved and comments on usage. In many cases the value associated with a keyword is denoted by a letter such as $k$, and allowable values for $k$ are subsequently defined.

## AIJ TOLERANCE $t \quad$ (default $t=1.0 \mathrm{E}-10$ )

During input of the MPS file, matrix coefficients $a_{i j}$ will be ignored if $\left|a_{i j}\right|<t$.
If $a_{i j}$ is a Jacobian element, it is not ignored. (Its position is recorded, and it will relain the value $t$ if DERIVATIVE LEVEL $=2$ or 3 and FUNCON does not reset the corresponding element of G.)

If CYCLE LIMIT > 1 and $a_{i j}$ is to be changed from zero to a value greater than $t$ during a later cycle, set $t=0.0$ to retain all entries in the MPS file.

## BACKUP BASIS FILE $k \quad$ (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 | 11 | (or 0) |
| :--- | ---: | ---: |
| BACKUP BASIS FILE | 11 |  |
| NEW BASIS FILE | 12 |  |
| SAVE FREQUENCY | 100 |  |

The current basis will then be saved every 100 iterations, first on file 12 and then immediately on file 11. If the run is interrupted at iteration 2000 during the save on file 12, 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 12 will correspond to iteration 2050, but the last basis saved on file 11 will be the one for itcration 2000.

BOUNDS
BOUNDO1
This specifies the 8-character name of the bound set to be selected from the MPS file.

1. BNDS is a valid alternative keyword.
2. If BOUNDS is not specified, or if the name is blank, the first bound set in the MPS file will be sclected.
3. If the MPS file contains one or more bound sets but you do not want any of them to be used, specify a duinmy name such as BOUND = NORE.

CENTRAL DIFFERENCE INTERVAL $h_{2} \quad$ (default $h_{2}=\left(\right.$ FUNCTION PRECISION) ${ }^{\frac{1}{3}}$ )
When DERIVATIVE LEVEL < 3, the central-difference interval $h_{2}$ is used near an optimal solution to obtain more accurate (but more expensive) estimates of gradients. Twice as many function evaluations are required compared to forward differencing. The interval used for the $j$-th variable is $h_{j}=h_{2}\left(1+\left|x_{j}\right|\right)$. The resulting gradient estimates should be accurate to $O\left(h_{j}^{2}\right)$, unless the functions are badly scaled.

## CHECK FREQUENCY $k \quad$ (default $k=30$ )

Every $k$-th iteration after the most recent basis factorization, a numerical test is made to see if the current solution $x$ satisfies the general linear constraints (including any linearized nonlinear constraints, if any). If these are $\boldsymbol{A x}+\boldsymbol{s}=0$ where $s$ is the set of slack variables, the residual vector $r=A x+s$ is computed. If the largest component of $r$ is judged to be too large, the current basis is refactorized and the basic variables are recomputed to salisfy the general constraints more accurately.

## COEFFICIENTS

See Elements.

## COLUMNS <br> $\boldsymbol{n}$ <br> (default $n=3 *$ ROWS)

This must specify an over-estimate of the number of columns in the constraint matrix (excluding slack variables, but including any PHANTOM COLUMNS). If $n$ proves to be too small, MINOS will continue reading the MI'S file to determine the true value of $n$, and an appropriate warning message will be issued. If the MPS file number is the same as the system card reader, the problem will then be terminated; otherwise the MPS file will be re-read.

## COMPLETION PARTIAL (default) <br> COMPLETION FULL

When there are nonlinear constraints, this determines whether subproblems should be solved to moderate accuracy (PARTIAL completion), or to full accuracy (FULL completion). MINOS effects the option 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 nonlinear constraint error is reduced below 100 *(ROW TOLERANCE), the relative change in $\lambda_{k}$ is 0.1 or less, and the previous subproblem was solved to optimality.

Full completion tends to give better Lagrange-multiplier estimates. It may lead to fewer major iterations, but may result in more minor iterations.

## CRASH OPTION $k \quad$ (default $k=1$ )

If a starting basis is not specified, a triangular basis will be selected from certain columns of the constraint matrix ( $\Lambda \quad I$ ). Free rows and variables are given priority. Columns are then chosen from $\Lambda$ according to the following values of $k$. Slack columns (from $I$ ) are then added where necessary.

## Meaning

0 The all-slack basis is set up.
1 All columns of $\boldsymbol{A}$ are considered.
2 Only the columns of $\boldsymbol{A}$ 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 inilial values. (If an accurate initial point $x_{0}$ is provided, this may increase overall efficiency.)
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 | $\boldsymbol{l}$ | (default $l=1$ ) |
| :--- | :--- | :--- |
| CYCLE PRINT | $p$ | (default $p=1$ ) |
| CYCLE TOLERANCE | $t$ | (default $t=0.0$ ) |
| PHANTOM COLUMNS | $c$ | (default $c=0$ ) |
| PHANTOM ELEMENTS | $e$ | (default $e=0$ ) |

These keywords refer to a facility for constructing and solving a sequence of related problems, as described in sections $1.9,2.4$ and 2.5. The COMMON block

COMMON /CYCLCM/ CNVTOL, JNEW, MATERR, MAXCY, NEPHNT, NPHANT, NPRINT
contains certain relevant variables.

1. $l=$ MAXCY is the maximum number of problems to be solved.
2. $p=$ NPRINT controls the printing of intermediate solutions. At most, the last $p$ solutions will be output.
3. $t=$ CNVTOL is a real number for possible use in a user-specified convergence test within subroutine MATMOD.
4. $c=$ NPHANT is the number of columns that can be added to the constraint matrix beyond those specified in the MPS file. Each column must be added by means of a call to subroutine MATCOL. If an error occurs, matcol increments MATERR (which is initially zero). Otherwise, JNEW records the index of the new column.
5. $e=$ NEPHNT is the number of nonzero elements that are allocated to the "phantom columns" beyond those specified in the MPS file.

## DAMPING PARAMETER $d$ (default $d=2.0$ )

This parameler may assist convergence on problems that have highly nonlinear constraints. It is intended to prevent large relative changes between subproblem solutions $\left(x_{k}, \lambda_{k}\right)$ and $\left(x_{k+1}, \lambda_{k+1}\right)$. For example, the default value 2.0 prevents the relative change in cither $x_{k}$ or $\lambda_{k}$ from exceeding 200 per cent. It will not be active on well-bchaved problems.

The parameter is used to interpolate between the solutions at the beginning and end of each major iteration. Thus, $x_{k+1}$ and $\lambda_{k+1}$ are changed to

$$
x_{k}+\sigma\left(x_{k+1}-x_{k}\right) \quad \text { and } \quad \lambda_{k}+\sigma\left(\lambda_{k+1}-\lambda_{k}\right)
$$

for some stcp-length $\sigma<1$. (In the casc of nonlincar cquations, this gives a damped Newton method.)

1. This is a very crude control. If the sequence of major iterations does not appear to be converging, one should first re-run the problem with a higher PENALTY PARAMETER $\rho$ (say 10 or 100 times the default $\rho$ ). (Skip this re-run in the case of nonlinear equations. There are no degrees of freedom and the value of $\rho$ is irrelevant.)
2. If the subproblem solutions continue to change violently, try reducing $d$ to 0.2 or 0.1 (say).
3. For implementation reasons, the shortened step $\sigma$ applics to the nonlinear variables $x$, but not to the linear variables $y$ or the slack variables $s$. This may reduce the efficiency of the control.

DEBUG LEVEL
d
(de「ault $d=0$ )
This causes various amounts of information to be output to the PRINT file.

## $k \quad$ Meaning

0 No debug output.
2 (or more) Output from M5SETX showing the maximum residual after a row check.
40 Output from LUBRPC showing the position of the last nonzero in the transformed incoming column.

Output from LU2FAC showing each pivot row and column and the dimensions of the dense matrix involved in the associated elimination.

100
Output from M2BFAC and M5LOG listing the basic and superbasic variables and their values at every iteration.

## DERIVATIVE LEVEL

d
(default $d=3$ )
This specifies which nonlinear function gradients arc known analytically and will be supplied to MINOS by the user subroutines FUNOBJ and FUNCON.
d

## Meaning

3 All objective and constraint gradients are known.
2 All constraint gradients are known, but some or all components of the objective gradient are unknown.

1 The objective gradient is known, but some or all of the constraint gradients are unknown.

0 Some componenls of the objective gradient 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 $d=0$ or 2 , MINOS will estimate the missing components of the objective gradient, using finite differences. This may simplify the coding of subroutine FUNOBJ. However, it could increase the total run-time substantially (since a special call to FUNOBJ is required for each missing element), and there is less assurance that an acceptable solution will be located. If the nonlinear variables are not well scaled, it may be necessary to specily a nonstandard DIFFERENCE INTERVAL (see below).

If $d=0$ or 1, MINOS will estimate missing elements of the Jacobian. For each column of the Jacobian, one call to FUNCON is needed to estimate all missing elements in that column, if any. If JACOBIAN = SPARSE and the sparsity pattern of the Jacobian happens to be

$$
\left(\begin{array}{llll}
* & * & & * \\
& ? & ? & \\
* & & ? & \\
& * & & *
\end{array}\right)
$$

where * indicates known gradients and ? indicates unknown elements, MINOS will use one call to FUNCON to estimate the missing element in column 2, and another call to estimate both missing elements in column 3. No calls are needed for columns 1 and 4.

At times, central differences are used rather than forward differences. Twice as many calls to FUNOBJ and FUNCON are then needed. (This is not under the user's control.)

Remember: when analytic derivatives are not provided, the attainable accuracy in locating an optimal solution is usually less than when all gradients are available. DERIVATIVE LEVEL 3 is strongly recommended.

DIFFERENCE INTERVAL $\quad h_{1} \quad$ (default $h_{1}=$ (FUNCTION PRECISION) ${ }^{\frac{1}{2}}$ )
This alters the interval $h_{1}$ that is used to estimate gradients by forward differences 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 gradients.
4. For estimating missing Jacobian elements.

In the last threc cases, a derivative with respect to $x_{j}$ is estimated by perturbing that component of $x$ to the value $x_{j}+h_{1}\left(1+\left|x_{j}\right|\right)$, and then evaluating $F(x)$ or $f(x)$ at the perturbed point. The resulting gradient estimates should be accurate to $O\left(h_{1}\right)$ unless the functions are badly scaled. Judicious alteration of $h_{1}$ may sometimes lead to greater accuracy.

DUMP FILE $\quad f \quad$ (default $f=0$ )
If $f>0$, the last solution obtained will be output to file $f$ in the format described in section 5.3. The file will usually have been output previously as a LOAD file.

ELEMENTS $e \quad$ (default $e=5 *$ COLUNNS)
This must specify an over-estimate of the number of nonzero elements (coefficients $a_{i j}$ ) in the constraint malrix, including all entrics in a DENSE or SPARSE Jacobian, and all nonzeros in the matrices $\Lambda_{1}, \Lambda_{2}, \boldsymbol{A}_{3}$. (It should also include the nuinber of PHANTOM ELEMENTS, if any.)

1. COEFFICIENTS is a valid allernative keyword.
2. If e proves to be too small, MINOS continues in the manner described under COLUMN.

## EMERGENCY VERIFY LEVEL <br> See VERIfY level.

ERROR MESSAGE. LIMIT $e \quad$ (default $e=10$ )
This is the maximum number of error messages to be printed for each type of error occurring when the MI'S file is read. The default value is reasonable for early runs on a particular MPS file. If the same file is used repeatedly, e can be reduced to suppress warning of non-fatal errors.

## FACTORIZATION FREQUENCY $k \quad$ (default $k=50$ )

At most $k$ basis changes will occur between factorizations of the basis matrix.

1. With lincar programs, the basis factors are usually updated every iteration. The default $k$ is reasonable for typical problems. Higher values up to $k=100$ (say) may be more efficient on problems that are extremely sparse and well scaled.
2. When the objective function is nonlinear, fewer basis updates will occur as an optimum is approached. The number of iterations between basis factorizations will therefore increase. During these itcrations a test is made regularly (according to the CHECK FREQUENCY) to ensure that the general constraints are satisfied. If necessary the basis will be refactorized before the limit of $k$ updates is reached.
3. When the constraints are nonlinear, the MINOR ITERATIONS limit will probably preempt $\boldsymbol{k}$.

## FEASIBILITY TOLERANCE $t \quad$ (default $t=1.0 \mathrm{E}-6$ )

A feasible solution is one in which all variables satisfy their upper and lower bounds to within the absolute tolerance $t$. (This includes slack variables. Hence, the general linear constraints are also satisfied to within $t$.)

1. MINOS attempts to find a fcasible point before optimizing the objective function. If the sum of infeasibilities cannot be reduced to zero, the problem is declared INFEASIBLE. Iet SINF be the corresponding sum of infeasibilities. If SINF is quite small, it may be appropriate to raise $t$ by a factor of 10 or 100 . Otherwise, some error in the data should be suspected.
2. Note: if SINF is not small, there may be other points that have a significantly smaller sum of infeasibilities. MINOS does not attempt to find the solution that minimizes the sum.
3. If SCALE is used, feasibility is defined in terms of the scaled problem (since it is then more likely to be meaningful).
4. A nonlinear objective function $F(x)$ will be evaluated only at feasible points. If there are regions where $f^{\prime}(x)$ is undefined, every atlempt should be made to eliminate these regions from the problem. For example, if $r^{\prime}(x)=\sqrt{x_{1}}+\log x_{2}$, it is essential to place lower bounds on both variables. If Feasibility tolerance $=10^{-6}$, the bounds $x_{1} \geq 10^{-5}$ and $x_{2} \geq$ $10^{-1}$ might be appropriate. (The log singularity is more serious; in general, keep $x$ as far away from singularities as possible.)
5. Hounds should also be used to keep $x$ more than $t$ away from singularities in $f(x)$.
6. If there are any nonlincar constraints, each major iteration attempts to satisfy their linearization to within the tolerance $t$. If this is not possible, the bounds on the nonlinear constraints are relaxed temporarily (in several stages).
7. Peasibility with respect to the nonlincar constraints themselves is measured against the ROM TOLERANCE (not against $t$ ). The relevant test is made at the start of a major itcration.

## FUNCTION PRECISIOM $\quad \epsilon_{R} \quad$ (default $\epsilon_{R}=\epsilon^{0.8}$ )

The relative function precision $\epsilon_{\boldsymbol{R}}$ is intended to be a measure of the relative accuracy with which the nonlinear functions can be computed. For example, if $\boldsymbol{F}(\boldsymbol{x})$ is computed as 1000.56789 for some relevant $x$ and if the first 6 significant digits are known to be correct, the appropriate value for $\epsilon_{R}$ would be 1.OE-6.
(Ideally the functions $\boldsymbol{F}(\boldsymbol{x})$ or $f^{i}(x)$ should have magnitude of order 1 . If all functions are substantially less than 1 in magnitude, $\boldsymbol{C}_{\boldsymbol{R}}$ should be the absolute precision. For example, if $F(x)=1.23456789 E-4$ at some point and if the first 6 significant digits are known to be correct, the appropriate value for $\epsilon_{\boldsymbol{R}}$ would be 1.OE-10.)

1. The default value of $\epsilon_{R}$ is appropriate for simple analytic functions.
2. In some cases the function values will be the result of extensive computation, possibly involving an iterative procedure that can provide rather few digits of precision at reasonable cost. Specifying an appropriate FUNCTION PRECISION may lead to savings, by allowing the linesearch procedure to terminate when the difference between function values along the search direction becomes as small as the absolute error in the values.

## HESSIAN DIMENSION $r \quad$ (de「ault $r=$ SUPERBASICS LIMIT or 30)

This specifies that an $r \times r$ triangular matrix $R$ is to be available for use by the quasi-Newton algorithm (to approximate the reduced Hessian matrix according to $Z^{T} H Z \approx R^{T} R$ ). Suppose there are $s$ superbasic variables at a particular iteration.

1. If $s \leq r$, the first $s$ columns of $R$ will be used to approximate the reduced Hessian in the normal manner. If there are no further changes to the set of superbasic variables, the rate of convergence will ultimately be superlinear.
2. If $s>r$, a matrix of the form

$$
R=\left(\begin{array}{cc}
R_{r} & 0 \\
& D
\end{array}\right)
$$

will be used to approximate the reduced Hessian, where $R_{r}$ is an $r \times r$ upper triangular matrix and $D$ is a diagonal matrix of order $s-r$. The rate of convergence will no longer be superlinear.
3. The storage required is of order $\frac{1}{2} r^{2}$, which is substantial if $r$ is as large as 200 (say). In general, $r$ should be a slight over-estimate of the final number of superbasic variables, whenever storage permits. It need not be larger than $n_{1}+1$, where $n_{1}$ is the number of nonlinear variables. For many problems it can be much smaller than $n_{1}$.
4. If SUPERBASICS LIMIT $s$ is specified, the default value of $r$ is the same number, $s$ (and conversely). This is a safeguard to ensure superlinear convergence wherever possible. If neither $r$ nor $s$ is specified, both default to the value 30 .

## INSERT FILE

$$
f \quad \text { (default } f=0)
$$

If $f>0$, this references a file containing basis information in the format of section 5.2.

1. The file will usually have been output previously as a PUNCH file.
2. The file will not be accessed if an OLD BASIs file is specified.

## INVERT FREQUENCY

## Sce FACTORIZATION FREqUENCY.

ITERATIONS LIMIT $k \quad$ (default $k=3 * R O W S+10 *$ NONLINEAR VARS)
This is the maximum number of minor iterations allowed (i.e., iterations of the simplex method or the reduced-gradient algorithrn).

1. ITNS is an alternative keyword.
2. $k=0$ is valid. Both feasibility and optimality are checked.
3. If CYCLE LIMIT > 1 , the limit of $k$ minor iterations applies to each cycle separatcly.

| JACOBIAN | DENSE | (default) |
| :--- | :--- | :--- |
| JACOBIAN | SPARSE |  |

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

1. The DENSE option is convenient if there are not many nonlinear constraints or variables. It requires storage for three dense matrices of order $m_{1} \times n_{1}$.
2. The MP'S file may then contain any number of Jacobian entrics. Usually this means no entries at all.
3. For elficiency, the SPARSE option is preferable in all nontrivial cases. (Beware-it must be specifically requested.) The MPS file must then specify the position of all Jacobian elements (that are not identically zero), and subroutine FUNCON must store the elements of the Jacobian array $G$ in exactly the same order.
4. In both cases, if DERIVATIVE LEVEL $=2$ or 3 the MPS file may specify Jacobian elements that are constant for all values of the nonlinear variables. The corresponding elements of $\mathbf{G}$ need not be reset in FUNCON.

## LAGRANGIAN <br> LAGRANGIAN <br> YES <br> 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 FUNCON will be called only twice 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 greal risk that convergence may not occur. (Note: FUNCON will be called more often to estimate $J(x)$ if DERIVATIVE LEVEL < 2.)

## LINESEARCH DEBUG AFTER ITERATIONi

(default $i=999999$ )
This causes considerable information to be output by the linescarch procedures every iteration, once iteration $i$ has been completed. Its principal purpose is to assist the authors of the linescarch procedures to determine if the procedures are functioning correctly. In some cases it may confirm that the function values are very "noisy", or that the gradients computed in FUNOBJ or FUNCON are incorrect.

## LINESEARCH TOLERANCE $t \quad$ (default $t=0.1$ )

For nonlinear problems, this controls the accuracy with which an optimum of the merit function will be located along the direction of search each ileration.

1. $t$ must be a real value in the range $0.0 \leq t \leq 1.0$.
2. The default value $t=0.1$ requests a moderately accurate search. It should be satisfactory for many problems.
3. If the nonlinear functions are cheap to evaluate, a more accurate search may be appropriate; try $t=0.01$ or $t=0.001$. The number of iterations should decrease, and this will reduce total run time if there are many linear or nonlincar constraints.
4. If the nonlinear functions are expensive to evaluate, a less accurate search may be appropriate. If all gradients are known, try $t=0.5$ or perhaps $t=0.9$. (The number of iterations will probably increase, but the total number of function evaluations may decrease enough to compensate.)
5. If not all gradients are known, a reasonably accurate search remains appropriale. Each search will require only $2-5$ function values (typically), but many function calls will then be needed to estimate missing gradients for the next iteration.

## LIST LIMIT $\quad k \quad$ (default $k=0$ )

This limits the number of lines of the MPS file to be listed on the PRINT file during input. The header cards (NAME, ROWS, COLUMNS, RHS, RANGE, BOUNDS, ENDATA) and comment cards will always be listed, along with their position in the file.

LOAD FILE $\quad f \quad$ (default $f=0$ )
If $f>0$, this references a file containing basis information in the format of section 5.3.

1. The file will usually have been output previously as a DUMP file.
2. The file will not be accessed if an OLD BASIS file or an INSERT file is specified.

## LOG FREQUENCY

## $k$

(default $k=1$ )
One line of the iteration log will be printed every $k$-th minor iteration. A value such as $k=10$ is suggested for those interested only in the final solution.

## LOWER BOUND <br> $l \quad$ (default $l=0.0$ )

Before the BOUNDS section of the MPS file is read, all structural variables are given the default lower bound $l$. (Individual variables may subsequently have their lower bound altered by a BOUND set in the MPS file.)

1. LOWER BOUND $=1.0 E-5$ (say) is a useful method for bounding all variables away from singularities at zero. (Lixplicit bounds may also be necessary in the MPS file.)
2. If all or most variables are to be FRLEF, use LOWER BOUND $=-1.0 E+20$ to specify "minus infinity". (The default upper bound is already 1.OE+20, which is treated as "plus infinity".)

| LU FACTOR TOLERANCE | $t_{1}$ | (default $t_{1}=10.0$ ) |
| :--- | :--- | :--- |
| LU UPDATE TOLERANCE | $t_{2}$ | (default $t_{2}=10.0$ ) |

These tolerances affect the stability and sparsity of the basis factorization $L B=U$, during refactorization and updates respectively. Both lolerances must satisfy $t_{i} \geq 1.0$. The matrix $L$ is a product of matrices of the form

$$
\left(\begin{array}{ll}
1 & \\
\mu & 1
\end{array}\right)
$$

where the multipliers $\mu$ will satisfy $|\mu| \leq t_{i}$.

1. The default values $t_{i}=10.0$ usually strike a good compromise between stability and sparsity.
2. For large and relatively dense problems, $t_{i}=25.0$ (say) may give a marked improvement in sparsity without impairing stability to a scrious degree.
3. For certain very special structures (e.g., band matrices) it may be necessary to set $t_{1}$ in the range $1.0 \leq t_{1}<2.0$ to achicve stability.

## MAJOR ITERATIONS $\quad k \quad$ (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). Sce the advice given under PENALTY PARAMETER.

## MAXIMIZE <br> MINIMIZE

(default)
This specifies the required direction of optimization. It applies to both linear and nonlinear terms in the objective.

## MINOR ITERATIONS $\quad k \quad$ (default $k=40$ )

This is the maximum number of iterations allowed between successive lincarizations of the nonlinear constraints. A moderate value (e.g., $10 \leq k \leq 50$ ) 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 wy the ITERATIONS keyword as usual. If the problem is linearly constrained, this is the only limit (i.e., the MINOR ITERATIONS keyword is ignored).

## MPS FILE $f \quad$ (default $f=$ ?)

This is the file number for the MPS file. The default value is the system card reader IREAD, which is often $\int=5$.

1. INPUT FILE is a valid alternative keyword.
2. For nontrivial problems it is usually best to store the MPS file separately from the SPlSCS file. If the ROwS, COLUMNS or ELEMENTS estimates prove to be too low, MINOS will be able to rewind the MPS file and try again.

MULTIPLE PRICE $\quad k \quad$ (default $k=1$ )
Whenever a PIRICE operation is performed, the $k$ best nonbasic variables will be selected for admission to the superbasic set. ("Best" means the variables with largest reduced gradients of appropriate sign. If partial pricing is in effect, up to $k$ variables are selected from the current partition of $A$ and I.)

1. The default value $k=1$ is best for linear programs, since an optimal solution will have zero superbasic variables.
2. Warning: if $k>1$, MINOS will go into reduced-gradient mode even on purely lincar problems. The subsequent iterations do not correspond to the very efficient suboptimization. ("minor iterations") carried out by standard linear programıning systems using multiple pricing. (MINOS varies all superbasic variables simuliancously. However, its storage requirements are essentially independent of $k$ on linear problems. Thus, $k$ need not be limited to 5 or 6 as it is in standard systems, which require storage for $k$ dense vectors of dimension $m$.)
3. On large nonlinear problems it may be important to set $k>1$, if the starting point does not contain many superbasic variables. For example, if a problem has 3000 variables and 500 of them are nonlinear, the optimal solution may well have 200 variables superbasic. If the problem is solved in several runs, it may be beneficial to use $k=10$ (say) for carly runs, until it seems that the number of superbasics has levelled off.

NEW BASIS FILE $\quad f \quad$ (default $f=0$ )
If $f>0$, a basis map will be saved on file $f$ every $k$-th iteration, where $k$ is the SAVE FREQUENCY.

1. The first card of the file will contain the word PROCEEDING if the run is still in progress.
2. If $f>0$, a basis map will also be saved at the end of a run, with some other word indicating the final solution status.

| NONLINEAR CONSTRAINIS | $m_{1}$ | (default $m_{1}=0$ ) |
| :--- | :--- | :--- |
| NONLINEAR VARIABLES | $n_{1}$ | (default $n_{1}=0$ ) |
| NONLINEAR OBJECTIVE VARIABLES | $n_{1}^{\prime}$ | (default $n_{1}^{\prime}=0$ ) |
| NONLINEAR JACOBIAN VARIABLES | $n_{1}^{\prime \prime}$ | (default $n_{1}^{\prime \prime}=0$ ) |

These keywords define the parameters $M$ and $N$ in subroutines FUNOBJ and FUNCON. For example, $M$ in FUNCON will take the value $m_{1}$, if $m_{1}>0$.

1. If the objective function and the constraints involve the same sel of nonlinear variables $x$, then NONLINEAR VARIABLES $n_{1}$ is the simplest way to set $N$ to be the same value for both subroutines.
2. Otherwise, the NONLINEAR OBJECTIVE and NONLINEAR JACOBIAN keywords should be used to specify $n_{1}^{\prime}$ and $n_{1}^{\prime \prime}$ scparately.
3. If $m_{1}=0$, the value $n_{1}^{\prime \prime}=0$ is assumed regardless of $n_{1}$ or $n_{1}^{\prime}$.
4. 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 $\boldsymbol{n}_{1}^{\prime \prime} \leq$ $n_{1}^{\prime}$ (unless $n_{1}^{\prime}=0$ ). 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.

OBJECTIVE
C0ST
This specifies the 8 -character name of the type $N$ row in the MPS file to be selected as the linear part of the objective function (i.e., the objective function for linear programs).

1. If OBJECTIVE is not specified, or if the name is blank, the first $N$ row in the ROWS section of the MPS file will be selected. (Warning: objective rows must be listed after nonlinear constraint rows in the ROWS section of the MI'S file.)
2. If the ROWS section contains one or more $N$ rows but you do not want any of them to be used in the objective function, specify a dummy name. If the objective is defined entirely by subroutine FUNOBJ it may be helpful to specify OBJECTIVE = FUNOBJ. (However, don't expect a different name to invoke a different subroutine!)

OLD BASIS FILE $\quad f \quad$ (default $f=0$ )
If $f>0$, the starting point will be obtained from this file in the format of section 5.1.

1. The file will usually have been output previously as a NEW BASIS FILE.
2. The file will not be acceptable if the number of rows or columns in the problem has been altered.

## OPTIMALITY TOLERANCE $t \quad$ (default $t=1.0 \mathrm{E}-6$ )

This is used to judge the size of the reduced gradients $d_{j}=g_{j}-\pi^{T} a_{j}$, where $g_{j}$ is the gradient of the objective function corresponding to the $j$-th variable, $a_{j}$ is the associated column of the constraint matrix (or Jacobian), and $\pi$ is the set of dual variables.

1. By construction, the reduced gradients for basic variables are always zero. Oplimality will be declared if the reduced gradients for nonbasic variables at their lower or upper bounds satisfy

$$
d_{j} /\|\pi\| \geq-t \quad \text { or } \quad d_{j} /\|\pi\| \leq t
$$

respectively, and if

$$
\left|d_{j}\right| /\|\pi\| \leq t
$$

for superbasic variables.
2. In the above tests, $\|\pi\|$ is a measure of the size of the dual variables. It is included to make the tests independent of a scale factor on the objective function.
3. The quantity actually used is defined by

$$
\begin{aligned}
\sigma & =\sum_{i=1}^{m}\left|\pi_{i}\right|, \\
\|\pi\| & =\max \{\sigma / \sqrt{m}, 1\}
\end{aligned}
$$

so that only large scale factors are allowed for. If the objective is acaled down substantially, the test for optisnality reduces to comparing just $d_{j}$ against $t$.

## PARTIAL PRICE $p \quad$ (default $p=1$ or $c$ (see below))

This parameter is recommended for large problems that have significantly more variables than constraints. It reduces the work required for each "pricing" operation (when a nonbasic variable is selected to become superbasic).

1. When $p=1$, all columns of the constraint matrix ( $A I$ ) are searched.
2. Otherwise, $A$ and $I$ are partitioned to give $p$ roughly equal segments $A_{j}, I_{j}(j=1$ to $p$ ). If the previous pricing scarch was successful on $A_{j-1}, I_{j-1}$, the next search begins on the segments $\boldsymbol{A}_{j}, I_{j}$. ( $\Lambda l l$ subscripts here are modulo $p$.) If a reduced gradient is found that is larger than some dynamic tolerance, the variable with the largest such reduced gradient (of appropriate sign) is selected to become superbasic. (Several may be selected if MULTIPLE PRICE has been specilied.) If nothing is found, the search continues on the next segments $A_{j+1}, I_{j+1}$, and so on.
3. The default value of $p$ is 1 for moderate-sized problems, but may be greater than 1 otherwise. A quantity

$$
c=\max \{1000,4 * \text { ROWS }\}
$$

is defined. If COLUMNS $\geq c$ and PARTIAL PRICE has not been specified, $p$ will take the value COLUMNS / $2 *$ ROWS $\}$.
4. PARTIAL PRICE $p$ is recommended for time-stage models having $p$ time periods.

## PENALTY PARAMETER <br> $\rho \quad$ (default $\rho=100.0 / m_{1}$ )

This is the value of $\rho$ in the modified augmented Lagrangian. It is used only when LAGRANGIAN $=$ YES.

For carly runs on a problem with unknown chararteristics, something like the default value should be specified. If the problem is known to be highly nonlinear, specify a larger value, such as 10 times the default. In general, a positive value of $\rho$ may be necessary to ensure convergence, even for convex programs.

On the other hand, if $\rho$ is too large, the rate of convergence may be unnecessarily slow. If the functions are not highly nonlinear or a good starting point is known, it will often be safe to specify PENALTY PARAMETER 0.0.

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 $\rho$, such as the default, and a reasonably low ITERATIONS and/or MAJOR ITERATIONS limit.
2. If successive major ilerations appear to be terminating with radically different solutions, the penalty parameter should be increased. (See also the DAMPING PARAMETER.)
3. If there appears to be little progress beiween major iterations, the penalty parameter could be reduced.

| PHANTOM COLUNNS | c | $($ default $c=0$ ) |
| :--- | :--- | :--- |
| PHANTOM ELEMENTS | $e$ | (dcfault $e=0$ ) |

Sce the CYCLE parameters.

## PIVOT TOLERANCE $t \quad$ (default $t=\epsilon^{\frac{3}{s}}$ )

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 roughly $10^{-11}$ for double precision on IBM systems. This should be satisfaclory in most circumstances.

## PRINT LEVEL (JFLXB) $p \quad$ (default $p=00001$ )

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
JFLXB
where each letter stands for a digit that is either 0 or 1 . The quantities referred to are:
B BASIS statistics, i.e., information relating to the basis matrix whenever it is refaclorized.
X $\quad x_{k}$, the nonlincar variables involved in the objective function or the constraints.
L $\quad \lambda_{k}$, the Lagrange-multiplier estimates for the nonlinear constraints. (Suppressed if the option LAGRANGIAN $=N O$ is specified, since $\lambda_{k}=0$ then.)
$F \quad f\left(x_{k}\right)$, the values of the nonlinear constraint functions.
$J \quad J\left(x_{k}\right)$, 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. (Ilence if $J=1$, there is no reason to specify $\mathrm{X}=1$ unless the objective contains more nonlinear variables than the Jacobian.) A typical line of output is

$$
31.250000 \mathrm{D}+01 \mathrm{BS} \quad 1 \quad 1.00000 \mathrm{E}+00 \quad 4 \quad 2.00000 \mathrm{E}+00
$$

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.)

PROBLEM NUMBER
$n \quad$ (default $n=0$ )
For nonlincar problems, this assigns a value to the parameter NPROB in the user subroutines FUNOBJ, FUNCON and MATMOD.

## PUNCH FILE $\quad f \quad$ (default $f=0$ )

If $f>0$, the final solution obtained will be output to file $f$ in the format described in section 5.2. For linear programs, this format is compatible with various commercial systems.

## RADIUS OF CONVERGENCE $r \quad$ (default $r=0.01$ )

This determines when the penalty parameter $\rho$ will be reduced (if initialized to a positive value). Both the nonlinear constraint violation (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 $\rho$ is reduced or set to zero. Once $\rho$ is zero, the sequence of major iterations should converge quadratically to an optimum.

## RANGES

RANGEOO1
This specifies the 8 -character name of the range set to be selected from the MP'S file.

1. RNGS is a valid alternative keyword.
2. If Ranges is not specified, or if the name is blank, the first range set in the MPS file will be selected.
3. If the MPS file contains one or more range sets but you do not want any of them to be used, specify a dummy name such as ranges $=$ None.

RHS
RHSIDE3
This specifies the 8 -character name of the righthand side to be solected from the MPS file.

1. If RHS is not specified, or if the name is blank, the first righthand side in the MPS file will be selected.
2. If the MP'S file contains one or more righthand sides but you do not want any of them to be used, specify a dummy name such as RHS = NONE.

Rows
$m \quad$ (default $m=100$ )
This must specify an over-estimate of the number of rows in the constraint matrix. It includes the number of nonlincar constraints and the number of general linear constraints.

If $m$ proves to be too small, MINOS continues in the manner described under Coluxns.

## ROW TOLERANCE $\epsilon_{r} \quad$ (default $\epsilon_{r}=1.0 \mathrm{E}-8$ )

This specifics how accurately the nonlinear constraints should be satisfied. (Hoth "Row" and "TOLE" are significant on this data card.) The default value of $1.0 \mathrm{E}-8$ is often appropriate, since the MPS file contains data to about that accuracy.

Let ROWERR be defined as the maximum component of the residual vector $f(x)+\Lambda_{1} y-b_{1}$, normalized by the size of the solution. Thus,

$$
R O W E R R=\left\|f(x)+A_{1} y-b_{1}\right\|_{\infty} / X N O R M
$$

where $X N O R M$ is a measure of the size of the basic and superbasic variables. The solution $(x, y)$ is regarded as acceptably feasible if ROWISRR $\leq \epsilon_{r}$.

If some of the problem functions are known to be of low accuracy, a larger row tolerance may be appropriate.

```
SAVE FREQUENCY k (default k=100)
```

If a NEW BASIS file has been specified, a basis map describing the current solution will be saved on the appropriate file every $k$-th iteration. A BACKUP BASIS file will also be saved if specified.

## SCALE

SCALE YES
SCALE NO
(default)
scale tolerance
$t \quad$ (default $t=0.9$ )
SCALE, PRINT
SCALE, PRINT, tolerance $=\quad t \quad$ (default $t=0.9$ )
All forms of SCALE (except NO) request that the linear constraints and variables be scaled by an iterative procedure that attempts to make the matrix coefficients as close as possible to 1.0. This will sometimes improve the performance of the solution procedures.

The printed solution and the basis and solution files are output in original unscaled units. Scaling is therefore transparent to the user, except that more storage is required (NB=n+m double words) and precautions must be taken in applications involving subroutines MATMOD and MATCOL.

1. The tolerance $t$ must lic in the range $0.0<t<1.0$. It affects how many passes might be needed through the constraint matrix. On each pass, the scaling procedure computes for cach column $j$ the ratio of the largest and smallest nonzero cocflicients in the column:

$$
\rho_{j}=\max _{i\left(a_{i}, \neq 0\right)}\left|a_{i j}\right| / \min \left|a_{i j}\right|
$$

and records the largest such ratio, $\max _{j} \rho_{j}$. If this is less than $t$ times its previous value, another scaling pass is performed to adjust the row and column scales.
2. Raising $t$ from 0.9 to 0.99 (say) will probably increase the number of scaling passes. At most 10 passes will be made.
3. If PRINT is specified, the row-scales $r(i)$ and column-scales $c(j)$ will be output ( $i=1$ to $m$, $j=1$ to $n$ ). The scaled matrix cocflicients are

$$
\bar{a}_{i j}=a_{i j} r(i) / c(j)
$$

and the scaled bounds on the variables and slacks are

$$
l_{j}=l_{j} c(j), \quad \bar{u}_{j}=u_{j} c(j)
$$

where $c(j) \equiv r(j-n)$ if $j>n$. These scalcd iterns are stored permanently in place of the original data.
4. The scale factors for nonlinear constraints and nonlinear variables are defined to be 1.0 ; they are not printed.

| SOLUTION | YES | (default) |
| :--- | :--- | :--- |
| SOLUTION | NO |  |

SOLUTION IF OPTIMAL, INFEASIBLE, OT UNBOUNDED
SOLUTION IF ERROR CONDITION

## SOLUTION FILE $\quad f \quad$ (default $f=0$ )

The first four options determine whether the final solution obtained is to be output to the PIRINT file. The FILE option operates independently; if $f>0$, the final solution will be output to file $f$ (whether optimal or not).

1. For the YES, IF OPTIMAL, and IF ERROR options, floating-point numbers are printed in $\operatorname{F16.5}$ format, and "infinite" bounds are denoted by the word NONE.
2. For the FILE option, all numbers are printed in 1PE18.6 format, including "infinite" bounds which will have magnitude $1.000000 \mathrm{E}+20$.
3. To see more significant digits in the printed solution, it will sometimes be useful to make $f$ refer to the system PRINT file.

| START OBJECTIVE CHECK AT COLUNN $k$ | (default $k=1$ ) |
| :--- | :--- |
| START CONSTRAINT CHECK AT COLUNN $k$ | (default $k=1$ ) |
| STOP OBJECTIVE CHECK AT COLUNN $l$ | (default $l=n_{1}^{\prime}$ ) |
| STOP CONSTRAINT CHECK AT COLUNN $l$ | (default $l=n_{1}^{\prime \prime}$ ) |

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

1. If the first 100 objective gradients appeared to be correct in an carlier run, and if you have just. found a bug in FUNOBJ that ought to fix up the 101 -th component, then you might as well specify START OBJECTIVE CHECK AT COLUXN 101. Similarly for columns of the Jacobian matrix.
2. If the first 100 variables occur nonlinearly in the constraints, and the remaining variables are nonlinear only in the objective, then FUNOBJ 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.
These keywords are effective if VERIFY LEVEL > 0 . For an emergency verification at the end of a run, all objective and constraint gradients will be checked if EMERGENCY VERIFY LEVEL > 0.

SUBSPACE TOLERANCE $t \quad$ (default $t=0.5$ )
This controls the extent to which oplimization is confincd to the current set of basic and superbasic variables (Phase 4 iterations), before one or more nonbasic variables are added to the superbasic set (Phase 3).

1. $t$ must be a real number in the range $0.0<t \leq 1.0$. It is used as follows.
2. When a nonbasic variable $z$; is made superbasic, the resulting norm of the reduced-gradient vector (for all superbasies) is recorded. Let this be $\left\|Z^{T} g_{0}\right\|$. (In fact, the norm will be $\left|d_{j}\right|$, the size of the reduced gradient for $x_{j}$.)
3. Subsequent Phase 1 iterations will continue at least until the norm of the reduced-gradient vector satisfies $\left\|Z^{T} g\right\| \leq t \times\left\|Z^{T} y_{0}\right\|$. ( $\left\|Z^{T} g\right\|$ is the size of the largest reduced-gradient component among the superbasic variables.)
4. A sinaller value of $t$ is likely to increase the total number of iterations, but may reduce the number of basis changes. A larger value such as $t=0.9$ may sometimes lead to improved overall efficiency, if the number of superbasic variables has to increase substantially between the starting poinl and an optimal solution.
5. Other convergence teats on the change in the function being minimized and the change in the variables may prolong I'hase 1 iterations. This helps to make the overall performance insensitive to larger values of $t$.
SUMMARY FILE
$f \quad($ default $f=0)$
SUMMARY FREQUENCY
$k \quad$ (default $k=100$ )

If $f>\mathbf{0}$, a brief $\log$ will be output to file $f$, including one line of information every $k$-th iteration. In an interactive environment, it is useful to direct this output to the terminal, to allow a run to be monitored on-line. (If something looks wrong, the run can be manually terminated.) Further details are given in section 6.6.

SUPERBASICS LIMIT $\quad$ (default $s=$ HESSIAN DIMENSION, 30, or 1)
This specilies "how nonlinear" you expect a problem to be.

1. Normally, $s$ need not be greater than $n_{1}+1$, where $n_{1}$ is the specified number of nonlinear variables.
2. For many problems (that are not highly nonlinear), $s$ may be considerably smaller than $\boldsymbol{n}_{1}$. This will save storage if $n_{1}$ is very large.
3. This parameter also sets the HESSIAN DIMENSION, unless the latter is specified explicitly (and conversely). If neither parameter is specified, both default to the value 30 (except if there are no nonlincar variables, in which case both default to 1).

SUPPRESS PARAMETERS
Normally MIN()S prints the SPFCS 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.)

| UNBOUNDED OBJECTIVE VALUE | $F_{\text {max }}$ | (default $F_{\text {max }}=1.0 \mathrm{E}+20$ ) |
| :--- | :--- | :--- |
| UNBOUNDED STEP SIZE | $\alpha_{\text {max }}$ | (default $\alpha_{\text {max }}=1.0 \mathrm{E}+10$ ) |

These parameters are intended to detect unboundedness in nonlinear problems. (They may or may not achieve that purpose!) During a linesearch of the form

$$
\min _{0} F(x+\alpha p),
$$

if $|F|$ exceeds $F_{\text {max }}$ or $\alpha$ exceeds $\alpha_{\text {max }}$, iterations are terminated with the exit message PROBLEM IS UNBOUNDED (OR BADLY SCALED).

1. If singularitics are present, unhoundedness in $\vec{r}(x)$ may be manifested by a floating-point overflow (during the evaluation of $\boldsymbol{F}(x+\alpha p)$ ), before the test against $F_{\text {max }}$ can be made.
2. Unboundedness in $x$ is best avoided by placing finite upper and lower bounds on the variables. (For convenience, this can be accomplished in the SPECS filc; sce the LOWER and UPPER BOUND parameters.)

UPPER BOUND
Before the BOUNDS section of the MPS file is read, all structural variables are given the default upper bound $u$. (Individual variables may subsequently have their upper bound altered by the BOUNDS section in the MPS file.)

| VERIFY LEVEL | $l$ | (default $l_{1}=0$ ) |
| :---: | :---: | :---: |
| EMERGENCY VERIFY LEVEL | $l_{2}$ | (default $l_{2}=\max \left(l_{1}, 0\right)$ ) |
| VERIFY | NO |  |
| VERIFY LEVEL | 0 |  |
| VERIFY OBJECTIVE GRADIENTS |  |  |
| VERIFY LEVEL | 1 |  |
| VERIFY CONSTRAINT GRADIENTS |  |  |
| VERIFY LEVEL | 2 |  |
| VERIFY |  |  |
| VERIFY | YES |  |
| VERIFY GRADIENTS |  |  |
| VERIFY LEVEL | 3 |  |

These keywords refer to finite-difference checks on the gradient elements computed by the user subroutines FUNOBJ and FUNCDN. It is possible to specify VERIFY LEVELs 0-3 in several ways, as indicated above. 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.

If $0 \leq l_{1} \leq 3$, 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. If $l_{1}=0$, only a "cheap" test will be performed, requiring 3 calls to FUNOBJ or 2 calls to FUNCON. If $1 \leq l_{1} \leq 3$, a more reliable check will be made on individual gradient components, within the ranges specified by the START and STOP keywords. A key of the form "OK" or "BAD?" indicales whether or not each component appears to be correct.

If $10 \leq l_{1} \leq 13$, the action is the same as for $l_{1}-10$, except that it will take place immediately after the first basis factorization. Any superbasic variables will retain their initial value (for example, those specificd with FX indicators in the INITIAL bound set, if no basis file is loaded). This option may be prefcrable, or even necessary if the first set of linearized constraints has no feasible solution. However, if the nonlinear functions are not well defined at the first (infeasible) point, a fatal error may result.

An EMERGENCY gradient check takes place at the end of a run if the solution algorithm is unable to make proper progress. The various levels $l_{2}=0-3$ have the same meaning as for $l_{1}$, but the individual gradient checks are not controlled by the START and STOP keywords.

1. VERIFY LEVEL 3 should be specified whenever a new function routine is being developed.
2. Missing gradients are not checked; i.e., they result in no overhead.
3. The default action is to perform a cheap check on the gradients at the first feasible point. Even on debugged function roulines, the message "GRADIENTS SEEM TO BE OK" will provide certain comfort at nominal expense.
4. If necessary, checking can be suppressed by specifying VERIFY LEVEL -1 and/or EMERGENCY VERIFY LEVEL -1.

## Weight on linear objective $\quad w \quad$ (default $w=0.0$ )

This keyword invokes the so-called composite objective technique, if the first solution obtained is infeasible, and if linear terms for the objective function are specified in the MPS file. While trying to reduce the sum of infeasibilities, the method also attempts to optimize the linear objective.

1. At each infeasible iteration, the objective function is defined to be

$$
\text { minimize } \sigma w\left(c^{T} x\right)+\text { (sum of infeasibilities), }
$$

where $\sigma=1$ for MINIMIZE, $\sigma=-1$ for MAXIMIZE, and $c$ is the linear objective row.
2. If an "optimal" solution is reached while still infeasible, $w$ is reduced by a factor of 10. This helps to allow for the possibility that the initial $\boldsymbol{w}$ is too large. It also provides dynamic allowance for the fact the sum of infeasibilities is tending towards zero.
3. The effect of $\boldsymbol{w}$ is disabled after 5 such reductions, or if a feasible solution is oblained.

```
WORKSPACE (USER) maxw (default maxw = 0)
WORKSPACE (TOTAL) maxz (default maxz = NWCORE)
```

These keywords define the limits of the region of storage that MINOS may use in solving the current problem. The main work array is declared in the main program, along with its length, by statements of the form

| DOUBLE PRECISION $\quad Z(25000)$ |  |
| :--- | :--- |
| DATA | NWCORE/25000/ |

where the actual length of $z$ must be specified at compile time. The values specified by the WORKSPACE keywords are stored in

COMMON /M2MAPZ/ MAXW,MAXZ,LEN(30),LOC (B0)
and workspace may be shared according to the following rules:

1. $Z(1)$ through 2 (maxW) is available to the user.
2. $Z(M A X W+1)$ through $Z$ (MAXZ) is available to MINOS, and should not be altered by the user.
3. $\mathbf{Z}$ (MAXZ +1 ) through $Z$ (NWCORE) is unused (or available to the user).

The arrays LEN and LOC are not used by MINOS.
The wORKSPACE parameters are most useful on machines with a virtual (paged) store. Some systems will allow NWCORE to be set to a very large number (say 500000) with no overhead in saving the resulting object code. At run time, when various problems of different size are to be solved, it may be sensible to confine MINOS to a portion of $\mathbf{Z}$ to reduce paging activity slightly. (However, MINOS accesses storage contiguously wherever possible, so the benefit may be slight. In general it is far better to have too much storage than not enough.)

## 4. THE MPS FILE

An MPS file is required for all problems to specify names for the variables and constraints, and to define the constraints themselves. In contrast to the relatively free format allowed in the SPECS file, a very fixed format must be used for the M1'S file. (This means that each item of data must appear in specific columns.)

Various "header cards" divide the MI'S file into several sections as follows:

## NAME

ROWS

COLUMNS

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

| Columns | $2-3$ | $5-12$ | $15-22$ | $25-36$ | $40-17$ | $50-61$ |
| :--- | :---: | :---: | :---: | :---: | :---: | :---: |
| Contents | Kcy | Name0 | Name1 | Valuc1 | Name2 | Value2 |

In addition, "comment" cards are allowed; these have an asterisk " ${ }^{*}$ " in column 1 and any characters in columns 2-22.

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

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

### 4.1 The NAME Card

NAME MODELOO1 (for example)
This card contains the word NAME in columns 1-4, and a name for the problem in columns 15-22. (The name may be from 1 to 8 characters of any kind, or it may be blank.) The name is used to label the solution outpul, and it appears on the first card of each basis file.

The NAMI; card is normally the first card in the MI'S file, but it may be preceded or followed by comment cards.

### 4.2 The ROWS Section

```
ROWS
    E FUNO1
    G FUNO2
    (for example)
    L CAPITAL1
    N COST
```

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

| Key | Row-type |
| :---: | :---: |
| E | $=$ |
| G | $\geq$ |
| L | $\leq$ |
| H | Objective |
| N | Free |

(The 1-character Kcy may be in column 2 or column 3.)
Row-types E, $G$ and $L$ are casily understood in terms of a linear function $a^{T} x$ and a right-hand side $\beta$. They would be used to specify constraints of the form

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

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

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

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

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

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

## Row-names for Nonlinear Constraints

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

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

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

### 4.3 The COLUMNS Section

| $15 \ldots 12$ | $15 \ldots 22$ | $25 \ldots \ldots .36$ | $40 \ldots 47$ | $50 \ldots \ldots .61$ | (fields) |
| :---: | :--- | :--- | :---: | :---: | :---: | :--- |
| COLUNNS |  |  |  |  |  |
| XO1 | FUNO6 | 1.0 | ROW09 | -3.0 |  |
| XO1 | ROW08 | 2.5 | ROW12 | 1.123456 | (example) |
| XO1 | ROW03 | -11.111111 |  |  |  |
| X02 | FUNO2 | 1.0 |  |  |  |
| XO2 | COSTO1 | 5.0 |  |  |  |

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

In general, Key is blank (except for comments), Name 0 is the column name, and Namel, Valuel give a row name and value for some coefficient in that column. If there is another row name and value for the same column, they may appear as Niame 2 , Valuc2 on the same card, or they may be on the next card.

If either Namel or Name2 is blank, the corresponding value is ignored.
Values are read by MINOS using Fortran format E12.0. This allows values to be entered in several forms; for example, 1.2345678, 1.2345678E+0, 123.45678E-2 and 12345678E-07 all represent the same number. It is usually best to include an explicit decimal point.

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

$$
\begin{aligned}
& 1.23 \mathrm{E}-02 \\
& 1.23 \mathrm{E}-2
\end{aligned}
$$

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

In the example above, the variable called X01 has 5 nonzero cocflicients in the constraints named FUNO6, ROWO9, ROW08, ROW12 and ROW03. The row names and values may be in an arbitrary order, but they must all appear before the entries for column X02.

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

## Nonlinear Variables

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

$$
\begin{array}{lc}
\text { minimize } & (x+y+z)^{2}+3 z+5 w \\
\text { subject to } & x^{2}+y^{2}+z \\
& x^{4}+y^{4}+w \\
& 2 x+4 y \\
& z \geq 0, \quad w \geq 0
\end{array}
$$

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

$$
A=\left(\begin{array}{ll}
J_{k} & A_{1} \\
A_{2} & A_{3}
\end{array}\right)
$$

where $J_{k}$ is the Jacobian matrix. The variables associated with $J_{k}$ and $\Lambda_{2}$ must appear first in the COLUMNS section, and their order must be consistent with the array $X(*)$ in subroutine FUNCON. Similarly, entries belonging to $J_{k}$ must appear in an order that is consistent with the array $G(*)$ in subroutine FUNCON.

For convenience, let the first $n_{1}$ columns of the constraint matrix be

$$
\binom{J_{k}}{A_{2}}=\left(\begin{array}{llll}
j_{1} & j_{2} & \ldots & j_{n_{1}} \\
a_{1} & a_{2} & \ldots & a_{n_{1}}
\end{array}\right),
$$

where $j_{1}$ is the first column of $J_{k}$ and $a_{1}$ is the first column of $A_{2}$. The cocficients of $j_{1}$ and $a_{1}$ must appear before the cocflicients of $j_{2}$ and $a_{2}$ (and so on for all columns). Usually, those belonging to $j_{1}$ will appear before any in $a_{1}$, but this is not essential. (If certain linear constraints are made nonlinear at a later date, this means that entries in the COLUMNS section will not have to be reordered. However, the corresponding row names will need be moved towards the top of the ROWS section.)

If Jacobian = dense, the elements of $J_{k}$ need not be sperified in the Mi's filc. If jacobian $=$ SPARSE, all nonzero elements of $J_{k}$ must be specificd. Any variable cocflicients should be given a dummy value, such as zero. These dummy entries identily the location of the elements; their actual values will be computed later by subroutine FUNCON or by finite differences.

If all constraint gradients are known (Derivative level $=2$ or 3 ), any Jacobian elements that are constant may be given their correct values in the COLIMNNS section, and then they need not be reset by subroutine FUNCON. This includes values that are identically aero such elements do not have to be specified anywhere (in the M1'S filc or in FUNCON). In other words, Jacohian elements are assumed to be zero unless specified otherwise.

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

A nonlinear objective function will often involve variables that occur only linearly in the constraints. In such eases we recommend that the objeretive variables be placed after the lacobian variables in the COLUMNS section, since the Jacobian will then be as small as possible. (See the variable $z$ in the example above.)

### 4.4 The RHS Section

| 1 | $5 \ldots \ldots 12$ | $15 \ldots 22$ | $25 \ldots \ldots .36$ | $40 \ldots 47$ | $50 \ldots \ldots .61$ |
| :--- | :---: | :---: | :---: | :---: | :---: |
| RHS |  |  |  |  |  |
|  | RHSO1 | FUNO1 | 1.0 | ROW09 | -3.0 |
|  | RHSO1 | ROW08 | 2.5 | ROW12 | 1.123458 |
|  | RHSO1 | ROW03 | -11.111111 |  |  |
|  | RHSO2 | FUNO2 | 1.0 |  |  |
|  | RHSO2 | FUNO4 | 5.0 |  |  |

This section sprcifies the elements of $b_{1}$ and $b_{2}$ in (2)-(3). Together these vectors comprise what is called the right-hand side. Only the nonzero cocfficients nced to be specified. They may appear in any order. The format is exactly the same as in the COLUMNS section, with Name0 giving a name to the right-hand side.

If $b_{1}=0$ and $b_{2}=0$, the RHS header card must appear as usual, but no rhs coefficients need follow.

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

### 4.5 The RANGES Section (Optional)

```
1 5 ... 12 15 ... }2
ROWS
    E FUNO1
    E FUNO2
    G CAPITALI
    L CAPITAL2
COLUMNS
RHS
    RHSO1
                    FUNO1
                                4 . 0
                                FUNO2
        4 . 0
RANGES
\begin{tabular}{llllr} 
RANGEO1 & FUNO1 & 1.0 & FUNO2 & -1.0 \\
RANGEO1 & CAPITAL1 & 1.0 & CAPITAL2 & 1.0
\end{tabular}
```

Ranges are used for constraints of the form

$$
l \leq a^{T} x \leq u
$$

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

| Row-type | Sign of r Lower limit, l | Upper limit, $u$ |  |
| :---: | :---: | :---: | :---: |
| E | + | $b$ | $b+\|r\|$ |
| E | - | $b-\|r\|$ | $b$ |
| C | + or - | $b$ | $b+\|r\|$ |
| L | + or - | $b-\|r\|$ | $b$ |

The format is exactly the same as in the COIUMNS section, with Name 0 giving a name to the range set. The constraints listed above will have the following limits:

$$
\begin{array}{ll}
4.0 \leq \text { FUNO1 } & \leq 5.0 \\
3.0 \leq \text { FUNO2 } & \leq 4.0 \\
4.0 \leq \text { CAPITAL1 } & \leq 5.0 \\
3.0 \leq \text { CAPITAL2 } \leq 4.0
\end{array}
$$

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

### 4.6 The BOUNDS Section (Optional)

| $1 \quad$ 6....12 | $15 \ldots . \ldots 2$ | $25 \ldots \ldots .36$ |
| :--- | :--- | :---: |
| BOUNDS |  |  |
| UP BOUND01 | X01 | 4.0 |
| UP BOUND01 | X02 | 4.0 |
| LO BOUNDO1 |  |  |
| X04 | -1.0 |  |
| UP BOUND01 | X04 | 4.0 |
| FR BOUND01 |  |  |
| X06 |  |  |
| UP BOUND01 | X08 | 4.0 |

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

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

In this section, Key gives the type of bound required, Vitme 0 is the name of the bound set, and Namel and Valuel are the column name and bound value. (Vamed and Valued are ignored.)

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

| Key | Bound-type | Resulting bounds |
| :---: | :--- | :---: |
| LO | Lower bound | $b \leq x \leq u$ |
| UP | Upper bound | $l \leq x \leq b$ |
| FX | Fixed variable | $b \leq x \leq b$ (i.e., $x=b$ ) |
| FR | Frce variable | $-\infty \leq x \leq+\infty$ |
| MI | Minus infinity | $-\infty \leq x \leq \quad u$ |
| PL | Plus infinity | $l \leq x \leq+\infty$ |

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

$$
\begin{aligned}
l \leq X 01 & \leq 1.0 \\
l \leq X 02 & \leq 1.0 \\
-1.0 & \leq X 04 \leq 1.0 \\
-\infty & \leq X 08 \leq 1.0
\end{aligned}
$$

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

## Nonlinear Problems

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

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

## The INITIAL Bounds Set

The name INITIAL is reserved for a special bound set that may be used (optionally) to assign initial value's to any number of the column variables. The INITIAL bounds set must appear after any normal bound sets (if any). A warning is given if it is the first set encountered after the BOUNDS card.

All bound types except FR have a meaning. In the example

| FX INITIAL | X1 | 1.0 |
| :--- | :--- | :--- |
| LO INITIAL | X2 | 2.0 |
| UP INITIAL | X3 | 3.0 |
| MI INITIAL | Y1 |  |
| PL INITIAL | $Y 2$ |  |

suppose that the five variables have already been given the upper and lower bounds $0.0 \leq$ $\mathrm{X} 1, \mathrm{X} 2, \mathrm{X} 3, \mathrm{Y} 1, \mathrm{Y} 2 \leq 5.0$. The initial basis and the starting point will then be affected as follows:

1. X1 will be made superbasic at the value 1.0 .
2. X2 will initially be made nonbasic at its lower bound, but if it is involved in the nonlinear constraints, it will have the value 2.0 when subroutine FUNCON is called for the first time.
3. X3 will initially be made nonbasic at its upper bound, but if it is involved in the nonlinear constraints, it will have the value 3.0 when subroutine $\operatorname{FUNCON}$ is called for the first time.
4. Y1 will initially be made nonbasic at its lower bound, and its initial value will be that bond (0.0 in this case).
5. Y2 will initially be made nonbasic at its upper bound, and its initial value will be that bound ( 5.0 in this case).

The key FX should be used if good starting values are known, particularly for nonlinear variables. However, this should not be at the expense of forming a very large set of superbasic variables, if the optimal solution is likely to contain only a few. If the number of FX entries has reached the SUPERBASICS LIMIT, any further FX will be treated as LO or UP, depending on which bound is closer to the specified numerical value.

Variables that are not specified in the INITIAL bounds set will initially be nonbasic at their lower or upper bounds (whichever is smaller in absolute value), or at zero if a variable is free.

In this context, variables that are inilially nonbasir may be selected by the CRASII procedure to become basie, in which case their initial values are unpredictable. If this arbitrariness sounds troublesome, use one of the CRASH OPTIONs to prevent various sets of variables from being chosen for the initial basis.

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

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

### 4.7 Comment Cards

Any card in the MPS file may contain an asterisk "*" in column 1 and arbitrary data in columns 2-22. Such cards will be treated as comments. They will appear in the printer listing but will otherwise be ignored.

Note that comment cards are input as if they were genuine data in each section of the MPS file. Thus, columns $25-36$ and $50-61$ should preferably be blank. If not, they should contain valid numerical data whenever non-comment cards would do so. (This is a limitation of ANSI 1966 Standard Fortran; data cannot be read under one format and then re-read under another.)

### 4.8 Restrictions and Extensions in MPS Format

1. Blanks are significant in the 8 -character name fields. We recommend that all names be leftjustified with no imbedded blanks. In particular, names referred to in the SPECS file must be left-justified in the MPS file; for example, OBJECTIVE $=$ COSTO2 specifies an 8 -character name whose last two characters are blank.
2. Comments ideally should use only columns 1-22 as noted above.
3. Scale factors cannot be entered in the ROWS section.
4. It does not matter if there is no row of type $N$.
5. There must be at least one row in the ROWS section, even for problems with no general constraints. (It may have row-type N.)
6. Nonlinear constraints must appear before linear constraints in the ROWS section.
7. Markers such as INTORG and INTEND are not recognized in the COLUMNS section.
8. Numerical values may be entered in E or F format. Spaces within the 12 -character ficlds are treated as if they were 0 's.
9. Nonlinear variables must appear before linear variables in the COLUMNS section.
10. If RANGES and BOUNDS sections are both present, the RANGES section must appear firsi.
11. In the BOUNDS section, if an UP entry specifics a zero upper bound, the corresponding lower bound is not affected. (Beware-in some MP systems, the lower bound is converted to $-\infty$.)
12. The bounds name INITIAL has a special meaning.

## 5. BASIS Files

For non-trivial problems, it is advisable to save a BASIS file at the end of a run, in order to restart the run if necessary, or to provide a good starting point for some closely related problem.

Three formats are available for saving basis descriptions. They are invoked by SPECS cards of the following form:

| NEW BASIS | FILE | 10 |  |
| :--- | :--- | :--- | :--- |
| BACKUP | FILE | 11 |  |
| PUNCH | FILE | (same as NEW BASIS but on a different file) |  |
| DUUP | FILE | 30 |  |

The file numbers may be whatever is convenient, or zero for files that are not wanted.
NEW BASIS and BACKUP files are saved every $k$-th iteration, in that order, where $k$ is the save frequency.

NEW, PUNCH and DUMP files are saved at the end of a run, in that order. They may be re-loaded at the start of a subsequent run by specifying SPECS cards of the following form respectively:

| OLD BASIS | FILE | 10 |
| :--- | :--- | :--- |
| INSERT | FILE | 20 |
| LOAD | FILE | $\mathbf{3 0}$ |

Only one such file will actually be loaded. If more than one positive file number is specified, the order of precedence is as shown. If no BASIS files are specified, one of the CRASH OPTIONs takes effect.

Figures 5.1-5.3 illustrate the data formats used for BASIS files. 80-character fixed-length records are suitable in all cases. ( 36 -character records would be adequate for PUNCH and DUMP files.) The files shown correspond to the optimal solution for the economic-growth model MANNE, described in section 8.4. Selected column numbers are included to deline significant data fields. The problem has 10 nonlinear constraints, 10 linear constraints, and 30 variables.

### 5.1 NEW and OLD BASIS Files

We sometimes call these files basis maps. They contain the most compact representation of the state of each variable. They are intended for restarting the solution of a problem at a point that was reached by an earlier run on the same problen or a related problem with the same dimensions. (Perhaps the ITERATIONS LIMIT was previously too small, or some other objective row is to be used.)

As illustrated in Figure 5.1, the following information is recorded in a NEW BASIS file.

1. A card containing the problem name, the iteration number when the file was created, the status of the solution (OPTIMAL SOLN, INFEASIBLE, UNBOUNDED, EXCESS ITNS, ERROR CONDN, or PROCEEDING), the number of infeasibilities, and the current objective value (or the sum of infeasibilitics).
2. $\Lambda$ card containing the OBJECTIVE, RHS, RANGES and BOUNDS namiss, $M=$ the number of rows in the constraint matrix, $N=$ the number of columns in the constraint matrix, and $S B=$ the number of superbasic variables.
3. A set of $(N+M-1) / 80+1$ cards indicating the state of the $N$ column variables and the $M$ slack variables in that order. One character $H S(j)$ is recorded for each $j=1,2, \ldots, N+M$ as follows, written with FORMAT (BOI1).

| HS $(j)$ | State of the $j$-th variable |
| :---: | :--- |
| 0 | Nonbasic at lower bound |
| 1 | Nonbasic at upper bound |
| 2 | Superbasic |
| 3 | Basic |

If variable $j$ is fixed (lower bound $=$ upper bound), then HS ( $j$ ) may be 0 or 1 . The same is true if variable $j$ is free (infinite bounds) and still nonbasic, although free variables will almost always be basic.
4. A set of cards of the form

$$
j \quad x_{j}
$$

written with FORMAT (I8, 1PE24.14) and terminated by an entry with $j=0$, where $j$ denotes the $j$-th variable and $x$ is a real value. The $j$-th variable is cither the $j$-th column or the $(j-N)$-th slack, if $j>N$. Typically, HS $(j)=2$ (superbasic). When nonlinear constraints are present, this list of superbasic variables is extended to include all basic nonlinear variables. The Jacobian matrix can then be reconstructed exactly for a restart.

## Loading a NEW BASIS file

A file that has been saved as an OLD BASIS file may be input at the beginning of a later run as a NEW BASIS file. The following notes are relevant:

1. The first card is input and printed but otherwise not used.
2. The values labelled $M$ and $N$ on the second card must agree with those for the MPS file that has just been read. The valuc labelled SB is input and printed but is not used.
3. The next set of cards must contain exactly $m$ values $H S(j)=3$, denoting the basic variables.
4. The list of $j$ and $x_{j}$ values must include an entry for every variable whose state is HS $(j)=2$ (the superbasic variables).
5. Further $j$ and $x_{j}$ values may be included, in any order.
6. For any $j$ in this list, if $H S(j)=3$ (basic), the value $x$; will be recorded for nonlinear variables, but the variable will remain basic.
7. If $\operatorname{HS}(j) \neq 3$, variable $j$ will be initialized at the value $x_{j}$ and its state will be reset to 2 (superbasic). If the number of superbasic variables has already reached the SUPERBASICS LIMIT, then variable $j$ will be made nonbasic at the bound nearest to $x$, (or at zero if il is a free variable).


Figure 5.1. Format of NEW and OLD BASIS Giles

Warning: This format is not quite compatible with MINOS 4.0 in the following respects.

1. On the second card, $\mathbf{u}$ is the number of constraints ( $m$, as before) but $\boldsymbol{N}$ is now the number of variables excluding slacks (i.e., $n$, the number of columns in the MPS file plus the number of phantom columns, if any). Previously, N had the value $n+1+m$; this included 1 for the right-hand side and $m$ for the slacks.
2. The basis map starting at card 3 does not contain an entry for the right-hand side, which was previously in position $n+1$. The length of the map is now $n+m$, not $n+1+m$.
3. In the list of ( $j x_{j}$ ) entries, the values of $j$ referring to slacks are now one less than before. (These are entries for which $j>n$.)
A basis map from MINOS 4.0 can thercfore be converted to the present format with reasonable ease. PUNCII and DUMP files from MINOS 4.0 should be acceptable as INSEIRT and LOAD files wilhout change.

### 5.2 PUNCH and INSERT Files

These files provide compatibility with commercial mathematical programming systems. The PUNCH file from a previous run may be used as an INSERT file for a later run on the same problem. It may also be possible to modify the INSERT file and/or problem and still obtain a useful advanced basis.

The standard MPS format has been slightly generalized to allow the saving and reloading of nonbasic solutions. It is illustrated in Figure 5.2. Apart from the first and last card, each entry has the following form:

| Columns | 2-3 | 5-12 | 15-22 | 25-36 |
| :--- | :---: | :---: | :---: | :---: |
| Contents | Key | Namel | Name2 | Value |

The various keys are best defined in terms of the action they cause on input. It is assumed that the basis is initially set to be the full set of slack variables, and that column variables are initially at their smallest bound in absolute magnitude.

| Key | Action to be taken during INSERT |
| :--- | :--- |
| XL | Make variable Name1 basic and slack Name2 nonbasic at its lower bound. |
| XU | Make variable Name1 basic and slack Name2 nonbasic at its upper bound. |
| LL | Make variable Namel nonbasic at its lower bound. |
| UL | Make variable Namel nonbasic at its upper bound. |
| SB | Make variable Namel superbasic at the specified Value. |

Note that Namel may be a column name or a row name, but (on XI and XU cards) Name2 must be a row name. In all cases, row names indicate the associated slack variable, and if Namel is a nonlinear variable then its Value is recorded for possible use in defining the initial Jacobian matrix.

The key SB is an addition to the standard MPS format to allow for nonbasic solutions.

## Notes on PUNCH Data

1. Variables are output in natural order. For example, on the first XL or XU card, Name $I$ will be the first basic column and Name2 will be the first row whose slack is not basic. (The slack could be nonbasic or superbasic.)
2. LL cards are not output for nonbasic variables if the corresponding lower bound value is zero.
3. Superbasic slacks are output last.
4. PUNCH and INSERT files deal with the status and values of slack variables. This is in contrast to the printed solution and the SOLUTION file, which deal with rows.

## Notes on INSERT Data

1. Before an INSERT file is read, column variables are made nonbasic at their smallest bound in absolute magnitude, and the slack variables are made basic.
2. Preferably an INSERT file should be an unmodified PUNCII file from an earlicr run on the same problem. If some rows have been added to the problem, the INSEIRT file need not be altered. (The slacks for the new rows will be in the basis.)
3. Entries will be ignored if Namel is already basic or superbasic. XI and $X U$ cards will be ignored if Name2 is not basic.
4. SB cards may be added before the ENDATA card, to specify additional superbasic columns or slacks.
5. An SB card will not alter the status of Namel if the SUPERBASICS LIMIT has been reached. However, the associated Value will be retained if Namel is a Jacobian variable.

### 5.3 DUMP and LOAD Files

These files are similar to PUNCH and INSERT files, but they record solution information in a manner that is more direct and more easily modified. In particular, no distinction is made between columns and slacks. Apart from the first and last card, each entry has the form

| Columns | $2-3$ | $5-12$ | $25-36$ |
| :--- | :---: | :---: | :---: |
| Contents | Key | Name | Value |

as illustrated in Figure 5.3. The keys LL, UL, BS and SB mean Lower Limit, Upper Limit, Basic and Superbasic respectively.

## Notes on DUMP Data

1. A card is output for every variable, columns followed by slacks.
2. Nonbasic free variables will be output with either LL or UL keys and with Value zero.

## Notes on LOAD Data

1. Before a LOAD file is read, all columns and slacks are made nonbasic at their smallest bound in absolute magnitude. The basis is initially empty.
2. Each LL, UL or BS card causes Name to adopt the sperified status. The associated Value will be retained if Name is a Jacobian variable.
3. An SB card causes Name to become superbasic at the spocified Value.
4. An entry will be ignored if Name is already basic or superbasic. (Thus, only the first BS or SB card takes effect for any given Name.)
5. An SB card will not alter the status of Name if the SUPERBASICS LIMIT has been reached, but the associated Value will be retained if Name is a Jacobian variable.
6. (Partial basis) Let $M$ be the number of rows in the problem. If fewer than $M$ variables are specified to be basic, a tentative basis list will be constrinted by adding the reguisite number of slacks, starting from the first row and taking those that were not previously specified to be basic or superbasic. (If the resulting basis proves to be singular, the basis factorization routine will replace a number of basic variables by other slacks.) The starting point obtained in this way will not necessarily be "good".
7. (Too many basics) If M variables have already been specified as hasir, any further BS keys will be treated as though they were SB. This feature may be useful for combining solutions to smaller problems.


Figure 5.s. Format of DUMP and LOAD files

### 5.4 Restarting Modified Problems

Sections 5.1-5.3 document three distinct starting methods (OLD BASIS, INSERT and LOAD files), which may be preferable to any of the cold start (CRASII) options. The best choice depends on the extent to which a problem has been modified, and whether it is more convenient to specify variables by number or by name. The following notes offer some rules of thumb.

## Protection

In general there is no danger of specifying infinite values. For example, if a variable is specified to be nonbasic at an upper bound that happens to be $+\infty$, it will be made nonbasic at its lower bound. Conversely if its lower bound is $-\infty$. If the variable is free (both bounds infinite), it will be made nonbasic at value zero. No warning message will be issued.

## Default Status

If the status of a variable is not explicitly given, it will initially be nonbasic at the bound that is smallest in absolute magnitude. Ties are broken in favor of lower bounds, and free variables will again take the value zero.

## Restarting with Different Bounds

Suppose that a problem is to be restarted after the bounds on some variable $\mathbf{X}$ have been altered. Any of the basis files may be used, but the starting point obtained depends on the status of $X$ at the time the basis is saved.

If X is basic or superbasic, the starting point will be the same as before (all other things being equal). The value of $X$ may lic outside its new set of bounds, but there will be minimal loss of Ceasibility or optimality for the problem as a whole.

If $\mathbf{X}$ was previously fixed, it is likely to be nonbasic at its lower bound (which happens to be the same as its upper bound). Increasing its upper bound will not affect the solution.

In contrast, if X is nonbasic at its upper bound and if that bound is altered, the starting values for an arbitrary number of basic variables could be changed (since they will be recomputed from the nonbasic and superbasic variables). This may not be of great consequence, but sometimes it may be worthwhile to retain the old solution precisely. To do this, one must make X superbasic at the original bound value.

For example, if $\mathbf{X}$ is nonbasic at an upper bound of 5.0 (which has now been changed), one should insert a card of the form

$$
\begin{array}{ll}
j & 5.0
\end{array}
$$

near the end of an OLD BASIS file, or the card
SB X
5.0
near the end of an INSERT or LOAD file. Note that the SPECS file must specify a SUPERBASICS LIXIT at least as large as the number of variables involved, even for purcly linear problems.

## Sequences of Problems

Whenever practical, a scries of related problems should be ordered so that the most tightly constrained cases are solved first. Their solutions will often provide feasible starting points for subsequent relaxed problems, as long the above precautions are taken.

## Altering Bounds with the CYCLE Option

Sequences of problems will sometimes be defined in conjunction with the CYCLE facilities. Various allerations can be made to each problem from within your own subroutine MATMOD. In particular, it is straightforward to alter the bounds on any of the columns or slacks.

Unfortunately, the present implementation of MINOS does not make it easy to alter the set of superbasic variables from within MATMOD. If the bound on a nonbasic variable is altered, it is simplest to accept the resulting perturbation to the values of the basic variables (rather than making the variable superbasic as suggested above).

## 6. OUTPUT

The following information is output to the PRINT file during the solution of each problem referred to in the SPECS file.

- A listing of the relevant part of the SPECS file.
- A listing of the parameters that were or could have been set in the SPECS file.
- An estimate of the amount of working storage needed, compared to how much is available.
- A listing of the MPS file, possibly abbreviated to the header cards and comment cards.
- Some statistics about the problem in the MPS file.
- The amount of storage available for the $L U$ factorization of the basis matrix.
- A summary of the scaling procedure, ir SCALE was specified.
- Notes about the initial basis resulting from a CRASII procedure or a BASIS file.
- The iteration log.
- Basis factorization statistics.
- The EXIT condition and some statistics about the solution obtained.
- The printed solution, if requested.

The last four items are described in the following sections. Further brief output may be directed to the SUMMARY file, as discussed in section 6.6.

### 6.1 Iteration Los

One line of information is output to the PRINT file every $k$-th minor iteration, where $k$ is the specified LOG FREQUENCY (default $k=1$ ). A heading is printed before the first such line following a basis factorization. The heading contains the items described below. In this description, a PRICE operation is defined to be the process by which one or more nonbasic variables are selected to become superbasic (in addition to those already in the superbasic set). Normally just one variable is selected, which we will denote by JQ. If the problem is purely linear, variable jQ will usually become basic immediately (unless it should happen to reach its opposite bound and return to the nonbasic set).

If PARTIAL PRICE is in effect, variable JQ is selected from $A_{P P}$ or $I_{P P}$, the PP-th segments of the constraint matrix ( $A \quad l$ ). If MULTIPLE PRICE is in effect, several variables may be selocted from $A_{P P}$ or IPP. In this case, JQ refers to the variable with the largest favorable reduced cost, DJ.

## Label

ITN

PH

## Description

The current iteration number. For problems with nonlinear constraints, this is the cumulative number of minor iterations.

The current phase of the solution procedure, as follows:
1 Phase 1 of the simplex method is being used to find a feasible point.
2 Phase 2 of the simplex method is being used to optimize the linear objective.
Normally, Phase 1 and 2 are used for purely linear problems. They may also be used at the start of a run even for nonlinear problems, if the initial basis contains only linear variables. Any superbasir variables will be temporarily held at their initial values.

3 Phase 3 of the reduced-gradient procedure is being used. This is the same as Phase 4 except that a PRICE operation is performed prior to the iteration, adding one or more nonbasic variables to the superbasic set.

4 Phase 4 of the reduced-gradient procedure is being used. Optimization is performed on the basic and superbasic variables (ignoring the nonbasics).

The Partial Price indicator. The variable(s) selected by the last PRICE operation came from the PP-th partition of $A$ and $I$. PP is set to zero when the basis is refactored. It is reset during Phase 1,2 or 3.

The number of "non-optimal" variables present in the set of nonbasic variables that were scanned during the last PRICE operation. It is reset during Phase 1, 2 or 3.

In Phase 1,2 or 3 , this is DJ, the reduced cost (or reduced gradient) of the variable JQ selected by PRICE at the start of the present iteration. Algebraically, DJ is $d_{j}=g_{j}-\pi^{T} a$, for $j=\mathrm{JQ}$, where $g_{j}$ is the gradient of the current objective function, $\pi$ is the vector of dual variables, and $a_{j}$ is the $j$-th column of the constraint matrix ( $A \quad I$ ).

In Phase 4, this quantity is RG, the norm of the reduced-gradient vector after the present iteration. (It is the largest value of $\left|d_{j}\right|$ for variables $j$ in the superbasic set.)
Note that for Phase 3 iterations, DJ is the norm of the reduced-gradient vector at the start of the iteration, just after the PIRICE operation.

The variable JQ selected by PRICE to be added to the superbasic set. (This is zero in Phase 4.)

The variable chosen to leave the set of superbasics. It has become basic if the entry under - BS is nonzero; otherwise it has become nonbasic.
The variable removed from the basis (if any) to become nonbasic.
The step length $\alpha$ taken along the current search direction $p$. The basic and superbasic variables $x_{R s}$ have just been changed to $x_{\theta}+\infty p$.

If column $a_{q}$ replaces the $r$-th column of the basis /S, PIVOT is the $r$-th element of a vector $y$ satisfying $B y=a_{q}$. Wherever possible. STEP is chosen to avoid extremely small values of PIVOT (since they cause the basis to be nearly singular). In rare cases, it may be necessary to increase the PIVOT TOLERANCE to exclude very small elements of $y$ from consideration during the computation of STEP.

The number of nonzeros representing the basis factor $I$. Immediately after a basis factorization $L / B=U$, this is LENL, the number of subuliugonal elements in the columns of a lower triangular matrix. Further nonarros are added to L when various columns of $I$ are later replaced. (Thus, $L$ increases monotonically.)

The number of nonzeros in the basis fartor 1 . Immediately aftor a basis factorization, this is LENU, the number of diagonal and superdiagonal elements in the rows of an upper triangular matrix. As columns of 13 are replaced, the matrix $U$ is maintained explicilly (in sparse form). The value of $U$ may lluctuate up or down; in general it will tend to increase.

NCP The number of compressions required to recover storage in the data structure for $U$. This includes the number of compressions needed during the previous basis factorization. Normally NCP should increase very slowly. If not, the amount of workspace available to MINOS should be increased by a significant amount. As a suggestion, the work array $Z(*)$ should be extended by $L+U$ elements.

NINF The number of infeasibilities before the present iteration. This number decreases monotonically.

SINF. OBJECTIVE If NINF $>0$, this is SINF, the sum of infeasibilities before the present iteration. (It will usually decrease at each nonzero STEP, but if NINF decreases by 2 or more, SINF may occasionally increase.)

Otherwise, it is the value of the current objective function after the present iteration. Note that "current objective function" can mean different things when $N I N F=0$. For linear programs, it means the true linear objective function. For problems with linear constraints, it means the sum of the linear objective and the value returned by subroutine FUNOBJ. For problems with nonlinear constraints, it is the quantity just described if LAGRANGIAN $=$ NO; otherwise it is the value of the augmented Lagrangian function for the current major iteration (which tends to the true objective function as convergence is approached).

The following items are printed if the problem is nonlinear or if the superbasic set is non-emply (i.e., if the current solution is nonbasic).

## Label

## Descriplion

NCON The number of times subroutine FUNCON has been called to evaluate the nonlinuar constraint functions.

NOBJ The number of times subroutine FUNOBJ has been called to evaluate the nonlinc:ar objective furction.

NSB The current number of superbasic variables.

HMOD An indication of the type of modifications made to the triangular matrix $/ 8$ that is used to approximate the reduced Hessian matrix. T'wo integers $i_{1}$ and $i_{2}$ are shown. They will remain zero for linear problems. If $i_{1}=1$, a IBFCS quasi-Newton update has been made to $R$, to account for a move within the current subspace. (This will not occur if the solution is infeasible.) If $i_{2}=1, R$ has been moditied to account for a change in basis. This will sometimes occur even if the solution is infeasible (if a feasible point was obtained at some earlier stage).

Both updates are implemented by triangularizing the matrix $R+v w^{T}$ for some vectors $v$ and $w$. If an update fails for mumerical reasons, $i_{1}$ or $i_{2}$ will be set $t o 2$, and the resulting $\boldsymbol{R}$ will be nearly singular. (However, this is highly unlikely.)

H-CONDA An estimate of the condition number of the reduced Hessian. It is the square of the ratio of the largest and smallest diagonals of the upper triangular matrix $R$. This constitutes a lower bound on the condition number of the matrix $R^{T} R$ that approximates the reduced Hessian. H-CONDN gives a rough indication of whether or not the optimization procedure is having difficulty. If $\epsilon$ is the relative precision of the machine being used, the reduced-gradient algorithm will make slow progress if H-CONDN becomes as large as $\epsilon^{-1 / 2}$, and will probably fail to find a better solution if H -CONDN reaches $\epsilon^{-3 / 4}$ or larger. (On IBM-like machines, these values are about $10^{8}$ and $10^{12}$.)

To guard against high values of H-CONDN, attention should be given to the scaling of the variables and the constraints. In some cases it may be necessary to add upper or lower bounds to certain variables to keep them a reasonable distance from singularities in the nonlinear functions or their derivatives.

A set of four logical variables $C_{1}, C_{2}, C_{3}, C_{4}$ that are used to determine when to discontinue optimization in the current subspace (Phase 4) and consider releasing a nonbasic variable from its bound (the PRICE operation of Phase 3). Let RG be the norm of the reduced gradient, as described above. The meaning of the variables $C_{j}$ is as follows:
$C_{1}$ is TRUE if the change in $x$ was sufficiently small;
$C_{2}$ is TRUE if the change in the objective was sufficiently small;
$C_{3}$ is TRUE if RG is smaller than some loose tolerance TOLRG; $C_{4}$ is TRUE if RG is smaller than some tighter tolerance.

The test used is of the form

$$
\text { if }\left(C_{1} \text { and } C_{2} \text { and } C_{3}\right) \text { or } C_{4} \text { then go to Phase } 3 .
$$

In the present implementation, TOLRG $=t|\mathrm{DJ}|$, where $t$ is the SUBSPACE TOLERANCE (nominally 0.5 ) and DJ is the reduced-gradient norm at the most recent Phase 3 iteration. The "tighter tolerance" is the maximum of 0.1 TOLRG and $10^{-7}\|\pi\|$. Only the tolerance $t$ can be altered at run-time (see section 3.3).

### 6.2 Basis Factorization Statistics

The following items are output whenever the basis matrix $B$ is factored. Caussian elimination is used to compute an $L U$ factorization of the form

$$
L B=U,
$$

where $L$ is unit lower triangular and $P / / Q$ is upper triangular for some permutation matrices $P$ and $Q$. This factorization is stabilized in the manner described under LU FACTOR TOLERANCE in section 3.3 .

Label

## Description

FACTORIZE
DEMAND

ITERATION
INFEAS
OBJECTIVE

NONLINEAR
LINEAR
SLACKS
ELEMS
DENSITY The percentage nonzero density of $I ; 100 \times$ ELEMS $/(M \times M)$, where $M$ is the number of rows in the problem ( $M=$ NONLINEAR + LINEAR + SLACKS $)$.

COMPRSSNS The number of times the data structure holding the partially factored matrix needed to be compressed, to recover umused storage. Ideally this number should be zero. If it is more than 3 or 4 , the amount of workspace available to NINOS should be increased for efficiency.

MERIT

LENL

LENU The number of nonzeros in U. The storage required for each nonzero is the same as for the nonzeros of $L$.

LMAX The maximum subdiagonal element in the columns of $L$. This will be no larger

INCREASE

BMAX
usax

GROWTH

The percentage increase in the number of nonzeros in $L$ and $U$ relative to the number of nonzeros in $B$; i.e., $100 \times$ (LENL + LENU - ELEMS)/ELEMS. than the LU FACTOR TOLERANCE.

The maximum nonzero element in $B$.
The maximum nonzero element in $U$, excluding elements of $B$ that remain in $U$ unaltered. (For example, if a slack variable is in the basis, the corresponding row of $B$ will become a row of $U$ without alteration. Elements in such rows wiil not contribute to UMAX. If the basis is strictly triangular, none of the elements of $B$ will contribute, and UMAX will be zero.)

Ideally, UMAX should not be substantially larger than BMAX. If it is several orders of magnitude larger, it may be advisable to reduce the LU FACTOR TOLERANCE to some value nearer 1.0. (The deiault value is $\mathbf{1 0 . 0}$.)
The smallest diagonal element of $P U Q$ in absolute magnitude.
The ratio UMAX/BMAX, which should not be too large (see above).
As long as LMAX is not large (say 10.0 or less), the ratio max \{BMAX, UMAX\}/UMIN gives an estimate of the condition number of $B$. If this number is extremely large, the basis is nearly singular and some numerical difficulties could conceivably occur. (However, an effort is made to avoid near-singularity by using slacks to replace columns of $B$ that would have made UMIN extremely small. Messages are issucd to this effect, and the modified basis is refactored.)

### 6.3 EXIT Conditions

For each problem in the SPECS file, a message of the form EXIT -- message is printed to summarite the final result. Here we describe each message and suggest possible courses of action.

System Note: A number is associated with each message below. It is the final value assigned to the integer variables INFORM and IERR, for possible use within subroutines MINOS1 and MINOS2. The variables appear in the declarations

```
SUBROUTINE MINOS2( Z,NWCORE,NCALLS,INFORM )
```

and

```
COMMON /M5LOG1/ IDEBUG,IERR,LPRINT
```

If a problem is infeasible, for example, their final values will be INFORM $=$ IERR $=1$.

The following messages arise when the SPECS file is found to contain no further problems.

## -2. EXIT -- INPUT ERROR. MINOS ENCOUNTERED END-OF-FILE OR AN <br> ENDRUN CARD BEFORE FINDING A SPECS FILE ON UNIT nn

The SPECS file may not be properly assigned. Its unit number nn is defined at compile time in subroutine MIFILE, and normally it is the system card input stream.

Otherwise, the SPECS file may be empty, or cards containing the keywords SKIP or ENDRUN may imply that all problems should be ignored (see section 1.8).
-1. ENDRUN
This messafe is printed at the end of a run if MINOS terminates of its own accord. Otherwise, the operating system will have intervened for one of many possible reasons (excess time, missing file, arithmetic error in user routines, etc.).

The following messages arise when optimization terminates gracefully. A solution exists, any of the BASIS files maty be saved, and the solution will be printed and/or saved on the told"IION file if requesled.

## O. EXIT -- OPTIMAL SOLUTION FOUND

This is the message we all hope to see! It is cortainly preforable to every other message, and we naturally want to believe what it says, because this is surcly ond siluation where the computer knows best. There may be cause for celcbration il the objective function has reached an astonishingly new high (or low). Or perhaps it will signal the cud of a strmuous series of runs that have iterated far into the night, depleting one's patiencr and computing funds to an equally alarming degree. (We hope not!)

In all cases, a distinct level of caution is in order, cyen if it can watit until urxt morning. For example, if the objective value is much better than experted, we may have olitained an optimal solution to the wrong problem! Almost any item of data could have that cffect, if it has the wrong value or is entered in the wrong columns of an input record. There may be thousands of items of data in the MI'S file, and the nonlinear functions (if any) could drpend on input files and other
data in innumerable ways. Verifying that the problem has been defined correctly is one of the more difficult tasks for a model builder. For early runs, we suggest that the LIST LIMIT be set to a suitably large number to allow the M1'S file to be printed for visual checking. It is also good practice in the function subroutines to print any data that is read in on the first entry.

If nonlinearities exist, one must always ask the question: could there be more than one local optimum? When the constraints are linear and the objective is known to be convex (e.g., a sum of squares) then all will be well if we are minimizing the objective: a local minimum is a global minimum in the sense that no other point has a lower function value. (However, many points could have the same objective valuc, particularly if the objective is largely linear.) Conversely, if we are maximizing a convex function, a local maximum cannot be expected to be global, unless there are sufficient constraints to confine the feasible region.

Similar statements could be made about nonlinear constraints defining convex or concave regions. However, the functions of a problem are more likely to be neither convex nor concave. Our advice is always to specily a starting point that is as good an estimate as possible, and to include reasonable upper and lower bounds on all variables, in order to confine the solution to the specific region of interest. We expect modellers to know something about their problem, and to make use of that knowledge as they themselves know best.

One other caution about "OPTIMAL SOLUTION"s. When nonlinearities are present, the final size of the reduced-gradient norm (NORM RG) should be examined to see if it is reasonably small compared to the norm of the dual variables (NORM PI). These quantitics are printed following the EXIT message. MINOS attempts to ensure that

## NORM RG / NORM PI $\leq$ OPTIMALITY TOLERANCE.

However, if messages of the form XXX SEARCH TERMINATED occur at the end of the run, this condition will probably not have been satisfied. The final solution may or may not be acceptably close to optimal. Broadly speaking, if

$$
\text { NORM RG / NORM PI }=10^{-d} \text {, }
$$

then the objective function would probably change in the $d$-th significant digit if optimization could be continued. One must judge whether or not $d$ is sufficiently large.

## 1. EXIt -- the problem is infeasible

When the constraints are linear, this message can probably be trusted. Feasibility is measured with respect to the upper and lower bounds on the variables. The message tells us that among all the points satisfying the gencral constraints $\Lambda x+s=0$, there is apparently no point that salisfies the bounds on $x$ and $s$. Violations as small as the FEASIbILITY TOLERANCE are ignored, but at least one component of $x$ or $s$ violates a bound by more than the tolerance.

Note: Although the objective function is the sum of infeasibilitics (when NINF $>0$ ), this sum will usually not have been minimized when MINOS recognizes the situation and exits. There may exist other points that have a significantly lower sum of infeasibilities.

When nonlinear constraints are present, infeasibility is much harder to recognize correctly. Even if a feasible solution exists, the current linearization of the constraints may not contain a feasible point. In an attempt to deal with this situation, MINOS is prepared to relax the bounds on the slacks associated with nonlinear rows. In the current implementation, the bounds are relaxed by increasingly large amounts up to 5 times per major iteration. Normally a feasible point
will be obtained to the perturbed constraints, and optimization can continue on the subproblem. However, if 5 consecutive subproblems require such perturbation, the problem is terminated and declared INFEASIBLE. Clearly this is an ad hoc procedure. Wherever possible, nonlinear constraints should be defined in such a way that fcasible points are known to exist when the constraints are linearized.

## 2. EXIT -- THE PROBLEM IS UNBOUNDED (OR BADLY SCALED)

For linear problems, unboundedness is detected by the simplex method when a nonbasic variable can apparently be increased or decreased by an arbitrary amount without causing a basic variable to violate a bound. A message prior to the EXIT message will give the index of the nonbasic variable. Consider adding an upper or lower bound to the variable. Also, examine the constraints that have nonzeros in the associated column, to sec if they have been formulated as intended.

Very rarely, the scaling of the problem could be so poor that numerical error will give an erroncous indication of unboundedness. Consider using the SCaLE option.

For nonlinear problems, MINOS monitors both the size of the current objective function and the size of the change in the variables at each step. If either of these is very large (as judged by the UNBOUNDED parameters - see section 3.3), the problem is terminated and declared UNBOUNDED. To avoid large function values, it may be necessary to impose bounds on some of the variables in order to keep them away from singularities in the nonlinear functions.

## 3. EXIT -- TOO MANY ITERATIONS

Either the ITERATIONS LIMIT or the MAJOR ITERATIONS LIMIT was exceeded before the required solution could be found. Check the iteration log to be sure that progress was being made. If so, restart the run using a basis file that was saved (or should have been saved!) at the end of the run.

## 4. EXIT -- THE OBJECTIVE HAS NOT CHANGED FOR THE LAST nnn ITERATIONS

This is an emergency measure for the rare occasions when the solution procedure appears to be cycling. Suppose that a zero step is taken for several consecutive iterations, with a basis change occurring each time. It is theoretically possible for the set of basic variables to become the same as they were one or more iterations earlier. The same sequence of iterations would then occur ad infinitum.

No direct attempt is made to recognize such cycling. The method used for determining the step size tends to guard against it happening, but nothing is guarantced. Furthermore, on so-called degenerate models (in which many basic variables are equal in value to their upper or lower bounds), a great number of consceutive zero sleps may have to occur before any progress can be made. A generous limit is therefore used on the number of consecutive zero steps allowed before this exit is taken. For small problems, the limit $n \Omega n$ is the maximum of 200 and $2(m+n)$. For large problems ( $m+n \geq 1000$ ) it is 1000 .

## 5. EXIT -- THE SUPERBASICS LIMIT IS TOO SMALL... nn

The problem appears to be more nonlinear than anticipated. The current set of basic and superbasic variables have been optimized as much as possible and a l'llICE operation is necessary to continue, but there are already nnn superbasics (and no room for any more).

In general, raise the SUPERBASICS LIMIT by a reasonable amount, bearing in mind the storage needed for the reduced Ilessian. (The HESSIAN DIMENSION $h$ will also increase to s.
unless specified otherwise, and the associated storage will be about $1 / 2 s^{2}$ words.) In extreme cases you may have to set $h<\boldsymbol{s}$ to conserve storage, but beware that the rate of convergence will probably fall off severely.

## 6. EXIT -- REQUESTED BY USER IN SUBROUTINE FUNOBJ (OR FUNCON) AFTER nan CALLS

This exit occurs if the subroutine parameter MODE is set to a negative number during some call to FUNOBJ or FUNCON. MINOS assumes that you want the problem to be abandoned forthwith.

In some environments, this exit means that your subroutines were not successfully linked to MINOS. If the default versions of FUNOBJ and FUNCON are ever called, they issue a warning message and then set MODE to terminate the run. For example, you may have asked the operating system to

LINK MINOS, FUNOBJ, FUNCON
when in fact you should have said

LINK FUNOBJ, FUNCON, MINOS

(or something similar) to give your own subroutines priority. Most linkers or loaders retain the first version of any subprogram that they see.
7. EXIT -- SUBROUTINE FUNOBJ SEEMS TO BE GIVING INCORRECT GRADIENTS

A check has been made on some individual elements of the gradient array, and at least one component $\mathbf{G}(j)$ is being set to a value that disagrees markedly with a forward-difference estimate of $\partial F / \partial x_{j}$. (The relative difference between the computed and estimated values is 1.0 or morc.) This exit is a safeguard, since MINOS will usually fail to make progress when the computed gradients are seriously inaccurate. In the process it may expend considerable effort before terminating with exit 9 below.

Check the function and gradient computation very carcfully. $\Lambda$ simple omission (such as forgetting to divide $F$ by 2) could explain everything. If $F$ or $G(j)$ is very large, then give serious thought to scaling the function or the nonlinear variables.

If you feel certain that the computed $G(j)$ is correct (and that the forward-difference estimate is therefore wrong), you can specify VERIFY LEVEL 0 to prevent individual elements from bring checked. However, the optimization procedure is likely to terininate unsuccessfully.

## 8. EXIT -- SUBROUTINE FUNCON SEEMS TO BE GIVING INCORRECT GRADIENTS

This is analogous to the preceding exit. At least one of the compuled Jacobian elements is significantly different from an estimate obtained by forward-dilferencing the constraint vector $f(x)$. Follow the advice given above, trying to ensure that the arrays $F$ and $G$ are being set correctly in subroutine FUNCON.

## 9. EXIT -- THE CURRENT POINT CANNOT BE IMPROVED UPON

Several circumstances could lead to this exit.

1. Subroutine FUNOBJ and/or subroutine FUNCON could be returning aceurate function values but inaccurate gradients (or vice versia). This is the most likely cause. Study the comenents given for exits 7 and 8 , and do your utmost to ensure that the subroutincs are coded correctly.
2. The function and gradient values could be consistent, but their precision could be too low. For example, accidental use of a single-precision data type when double-precision was intended throughout, would lead to a relative function precision of about $10^{-6}$ instead of something like $10^{-15}$. The default OPTIMALITY TOLERANCE of $10^{-6}$ would need to be raised to about $10^{-3}$ for optimality to be declared (at a rather suboptimal point). Of course, it is better to revise the function coding to obtain as much precision as economically possible.
3. If function values are obtained from an expensive iterative process, they may be accurate to rather few significant figures, and gradients will probably not be available. One should specify

| FUNCTION PRECISION | $t$ |
| :--- | :---: |
| OPTIMALITY TOLERANCE | $\sqrt{t}$ |

but even then, if $t$ is as large as $10^{-5}$ or $10^{-6}$ (only 5 or 6 significant figures), the same exit condition may occur. At present the only remedy is to increase the accuracy of the function calculation.
10. EXIT -- NUMERICAL ERROR. GENERAL CONSTRAINTS CANNOT BE SATISFIED ACCURATELY An $L U$ factorization of the basis has just been obtained and used to recompute the basic variables $x_{B}$, given the present values of the superbasic and nonbasic variables. A single step of "iterative refinement" has also been applied to increase the accuracy of $x_{B}$. However, a row check has revealed that the resulting solution does not satisfy the current constraints $A x+s=0$ sufficiently well.

This probably means that the current basis is very ill-conditioned. Request the SCALE option if there are any linear constraints and variables.

For certain highly structured basis matrices (notably those with band structure), a systematic. growth may occur in the factor $U$. Consult the description of UNAX, UMIN and GROWTH in section 6.2 , and set the LU FACTOR TOLERANCE to 2.0 (or possibly even smaller, but not less than 1.0 ).

## 11. EXIT -- CANNOT FIND SUPERBASIC TO REPLACE BASIC VARIABLE

If this exit occurs, the problem must be very badly scaled. A basic variable has reached a bound and must be replaced, but none of the superbasic columns has a pivot element excreding the PIVOT TOLERANCE. The latter could be reduced (at great risk). You should first try specifying SCALE.

## 12. EXIT -- BASIS FACTORIZATION REQUESTED TWICE IN A ROW

This exit may occur after the linesearch has terminated unsuccessfully several times in a row. It is a safeguard to prevent the various recovery measures from being repeated endessly. It should probably be treated as if it were exit 9.

> If the following exits occur during the first basis factorization, the basic variables $x_{B}$ will have certain default values that may not be particularly meaningful, and the dual vector $\pi$ will be zero. BASIS files will be saved if requested, but certain values in the printed solution will not be meaningful. The problem will be terminated, even if the CYCLE LIMIT has not yet been reached.

## 20. EXIT -- NOT ENOUGH STORAGE FOR THE BASIS FACTORIZATION

The main storage array $Z(*)$ is apparently not large enough for this problem. The routine declaring $Z$ is probably the main program. It should be recompiled with a larger dimension for 2. The new value should also be assigned to NWCORE.

In some cases it may be sufficient to increase the specified WORKSPACE (USER), if it is currently less than WORKSPACE (TOTAL).

An estimate of the additional storage required is given in messages preceding the exit.

## 21. EXIT -- ERROR IN BASIS PACKAGE

A preceding message will describe the error in more detail. One such message says that the current basis has more than one element in row $i$ and column $j$. This could be caused by a corresponding error in the MPS file. (MINOS does not check for duplicate row names within each column.) Determine the name of row $i$ (e.g., by consulting the $i$-th entry in the rows section of the printed solution), and scan the COLUMNS section of the MI'S file for that name. Alternatively, check the $(j-l)$-th variable in the COLUMNS section of the MPS file, where $l$ is the number of slack variables in the basis.
22. EXIT -- THE BASIS IS STRUCTURALLY SINGULAR AFTER TWO FACTORIZATION ATTEMPTS

This exit is highly unlikely to occur. The first factorization attempt will have found the basis to be structurally or numerically singular. (Some diagonals of the triangular matrix $P U Q$ were respectively zero or smaller than a certain tolerance.) The associated variables are replaced by slacks and the modified basis is refactorized. The ensuing singularity must mean that the problem is badly scaled, or the LU FACTOR TOLERANCE is too high.

If the following messages arise, the MP'S file was read sucerssfully. However, either an OLD BASIS file could not be loaded properly, or some fatal system error has occurred. New IBASIS tiles ramot be saved, and there is no solution to print. The problem is abandoned.

## 30. EXIT -- THE BASIS FILE DIMENSIONS DO NOT MATCH THIS PROBLEM

On the first card of the OLD 1 UASIS file, the dimensions labelled M and N ard different from those associated with the MPS file that has just been read. You have probahly haded a lile that belongs to some other problem.

Remember, if you have added rows or columns to the MI'S file, you will have to alter $M$ and $N$ and the map beginning on the third card (a hazardous operation). It may be easier to restart with a PUNCH or DUMP file from the carlier version of the problem.

## 31. EXIT -- THE BASIS FILE STATE VECTOR DOES NOT MATCH THIS PROBLEM

For some reason, the OLD BASIS file is incompatible with the present problem, or is not consistent within itself. The number of basic entries in the state vector (i.e., the number of 3 's in the map) is not the same as $M$ on the first card, or else some of the 2 's in the map did not have a corresponding $j x_{j}$ entry following the map.
32. EXIT -- SYSTEM ERROR. WRONG NO. OF BASIC VARIABLES... nn

This exit should never happen. If it docs, something is seriously awry in the MINOS source code. Perhaps the single- and double-precision files have been mixed up.

The following messages arise if the MPS file is seriously deficient, or if additional storage is needed to allow the MPS file to be input or to allow optimization to begin. The problem is abandoned.

## 40. EXIT -- FATAL ERRORS IN THE MPS FILE

One of the following conditions exists:

1. There are no entries in the ROWS section.
2. There are no entries in the COIUMNS section.
3. A type $N$ row has been selected to be the linear objective row, but it is one of the first $\boldsymbol{m}_{1}$ rows, where $m_{1}$ is the number of NONLINEAR CONSTRAINTS.
The first two conditions speak for themselves. If condition 3 occurs, the $N$ row may be have been selected by default (if you did not specify any OBJECTIVE name in the SPECS file). To prevent this, specify some other (possibly fictitious) row name. Otherwise, you must put the $N$ row after the nonlinear row names in the ROWS section.

## 41. EXIT -- NOT ENOUGH STORAGE TO READ THE MPS FILE

One of the ROWS, COLUMNS, or ELEMENTS estimates in the SI'ECS file proved to be too small. The minimum (exact) values are shown in carlier messages. You must specify these values, or higher values, and re-run the problem.

If the MPS data had been on a file of its own (rather than in the card input stream), AIIN(). would have been able to continue by rewinding the MPS file and trying again.
42. EXIT -- NOT ENOUGH STORAGE TO START SOLVING THE PROBLEM

The MPS file was read succesfully, but the main storage array $Z$ (*) is not large enough to provide workspace for the optimization procedure. Be sure that the SUPERBASICS LIMIT and HESSIAN DIMENSION are not unreasonably large. Otherwise, sec the advice given for exit 20.

### 6.4 Solution Output

At the end of a run, the final solution will be output to the PRINT file in accordance with the SOLUTION keyword. Some header information appears first to identify the problem and the final state of the optimization procedure. A ROWS section and a COLUMNS section then follow, giving one line of information for each row and column. The format used is similar to that seen in commercial systems, though there is no rigid industry standard.

## ROWS Section

The general constraints take the form $l \leq f(x)+A y \leq u$, where $x$ and $y$ are the nonlinear and linear variables respectively. The $i$-th constraint is therefore of the form

$$
\alpha \leq f^{\prime}(x)+a^{T} y \leq \beta
$$

and we define the $i$-th "row" to be the linearization of $f^{i}(x)+a^{T} y$. For linear constraints, the $i$-th row is just $a^{T} y$.

Internally, the constraints take the form $L f(x)+A y+s=0$ where $L f(x)$ is the current linearization of $f(x)$, and $s$ is the set of slack variables (which happen to satisfy the bounds $-u \leq s \leq-l$ ). For the $i$-th constraint it is the slack variable $s_{i}$ that is directly available, and it is sometimes convenient to refer to its state.

Label
Description
NUMBER
ROW The name of the $i$-th row.
STATE $\quad$ The state of the $i$-th row relative to the bounds $\alpha$ and $\beta$. The various states possible are as follows.
LL The row is at its lower limit, $\alpha$.
UL The row is at its upper iimit, $\beta$.
EQ The row is equal to the RHS element, $\alpha=\beta$.
BS The constraint is not binding. $s_{i}$ is basic.
SBS The constraint is not binding. $s_{i}$ is superbasic.
A key is sometimes printed before the STATE to give some additional information about the state of the slack variable.
A Alternative optimum possible. The slack is nonbasic, but its reduced gradient is essentially zero. This means that if the slack were allowed to start moving away from its bound, there would be no change in the value of the objective function. The values of the basic and superbasic variables might change, giving a genuine alternative solution. Ilowever, if there are any degenerate variables (labelled D), the actual change might prove to be zero, since one of them could encounter a bound immediately. In cither case, the values of dual variables might also change.
D Degencrate. The slack is basic or superbasic, but it is cyual to (or very close to) one of its bounds.

I Infeasible. The slack is basic or superbasic and it is currently violating one of its bounds by more than the FEASIBILITY TOLERANCE.
$N \quad$ Not precisely optimal. The slack is nonbasic or superbasic. If the OPTIMALITY TOLERANCE were tightened by a factor of 10 (c.g., if it were reduced from $10^{-5}$ to $10^{-6}$ ), the solution would not be declared optimal because the reduced gradient for the slack would not be considered negligible. (If a loose tolerance has been used, or if the run was terminated before optimality, this key might be helpful in deciding whether or not to restart the run.)
Note: If SCALE is specified, the tests for assigning the $A, D, I, N$ keys are made on the scaled problem, since the keys are then more likely to be correct.

ACTIVITY The row valuc; i.e., the value of $a^{T} y$ for linear constraints, or the value of the linearization $I f^{i}(x)+a^{T} y$ if the constraint is nonlinear.

SLACK ACTIVITY The amount by which the row differs from its nearest bound. (For free rows, it is taken to be minus the ACTIVITY.)

LOWER LIMIT $\alpha$, the lower bound on the row.
UPPER LIMIT $\beta$, the upper bound on the row.
DUAL ACTIVITY The value of the dual variable $\pi_{1}$, often called the shadow price (or simplex multiplier) for the $i$-th constraint. The full vector $\pi$ always satisfies $B^{T} \pi=g_{B}$, where $B$ is the current basis matrix and $g_{B}$ contains the associated gradients for the current objective function.

If the solution is feasible, the first $m_{1}$ components of $\pi$ are used at the start of the $k$-th major iteration to define $\lambda_{k}$, the estimate of the Lagrange multipliers for the nonlinear constraints.

The constraint number, $i$.

## COLUMNS Section

Here we talk about the "column variables" $(x, y)$. For enitenience we let the $j$-th romponent of $(x, y)$ be the variable $x$, and assume that it satisfies the bounds $\alpha \leq x, \leq \beta$. linear and nonlinear variables are treated the same.

## Label <br> Description

NUMBER

COLUNN $\quad$ The name of $\boldsymbol{x}_{\boldsymbol{j}}$.
STATE The state of $x_{j}$ relative to the bounds $\alpha$ and $\beta$. The various states possible are as follows.

LL $\quad x_{j}$ is nonbasic at its lower limit, $a$.
UL $\quad x_{j}$ is nonbasic at its upper limit, $\beta$.
EQ $\quad x_{j}$ is nonbasic and fixed at the value $\alpha=\beta$.

FR $x_{j}$ is nonbasic and currently zero, even though it is free to take any value. (Its bounds are $\alpha=-\infty, \beta=+\infty$. Such variables are normally basic.)

BS $x_{j}$ is basic.
SBS $\quad x_{j}$ is superbasic.
A key is sometimes printed before the STATE to give some additional information about the state of $x_{j}$. The possible keys are A, D, I and $N$. They have the same meaning as described above (for the ROWS section of the solution), but the words "the slack" should be replaced by " $x_{j}$ ".
activity The value of the variable $x_{j}$.
OBJ GRADIENT $g_{j}$, the $j$-th component of the combined linear and nonlinear objective function $F(x)+c^{T} x+d^{T} y$. (We define $g_{j}=0$ if the current solution is infeasible.)
LOWER LINIT $\alpha$, the lower bound on $x_{j}$.
UPPER LIMIT $\beta$, the upper bound on $x_{j}$.
REDUCED GRADNT The reduced gradient $d_{j}=g_{j}-\pi^{T} a_{j}$, where $a_{j}$ is the $j$-th column of the constraint matrix (or the $j$-th column of the Jacobian at the start of the final major iteration).
$\mathbf{M}+\mathrm{J} \quad$ The value $m+j$.

An example of the printed solution is given in chapter 8. Infinite UPPER and LOWER LIMITS are output as the word NONE. Other real values are output with format F 18.5 . The maximmm record length is 111 characters, including the first (carriage-control) character.

Note: If two problems are the same except that one minimizes $\boldsymbol{F}(x)$ and the other maximizes $-F(x)$, their solutions will be the same but the signs of the dual variables $\pi_{1}$ and the reduced gradients $d_{j}$ will be reversed.

### 6.5 SOLUTION File

If a positive SOLUTION FILE is specified, the information contained in a printed solution may also be output to the relevant file (which may be the PRINT file if so desired). Intimit. UPPER and LOWER LIMITS appear as $\pm 10^{20}$ rather than NONE. Other real values are output wilh format 1PE16.6. Again, the maximum record length is 111 characters, including what would be the carriage-control character if the file were printed.

A SOLUTION file is intended to be read from disk by a self-contained program that wrats and saves certain values as required for possible further computation. Typically the lirn 11 records would be ignored. Each subsequent record may be read using

FORMAT (I8, $2 \mathrm{X}, 2 \mathrm{~A} 4,1 \mathrm{X}, \mathrm{A1}, 1 \mathrm{X}, \mathrm{A3}, 5 \mathrm{E} 16.6$. 17)
adapted to suit the occasion. The end of the ROWS section is marked by a record that starts with a 1 and is otherwise blank. If this and the next 1 records are skipped, the colimivis section can then be read under the same format. (There should be no need to use any backspace statements.)

### 6.6 SUMMARY File

If SUMMRY FILE $f$ is specified with $f>0$, artain bricf information will be output to file $f$. When MNOS is run interactively, file $f$ will usually be the terminal. For batch jobs, a disk file should be used to retain a concise log of each run (if desired; a SUMMARY file is more easily perused than the associated PRINT file).

A SUMMARY file (like the PRINT file) is not rewound after a problem has been processed. It can therefore accumulate a log for every problem in the SPECS file, if each specifies the same file. The maximum record iength is $\mathbf{7 2}$ characters, including a carriage-control character in column 1.

The following information is included:

1. The BEGIN card from the SPECS file.
2. The actual number of rows, columns and elements in the MPS file.
3. The basis file loaded, if any.
4. The status of the solution after each basis factorization (whether feasible; the objective value; the number of function calls so far).
5. The same information every $k$-th iteration, where $k$ is the specified SUMMARY FREqUENCY (default $k=100$ ).
6. Warnings and error messages.
7. For nonlinear constraints, $\left\|x_{k+1}-x_{k}\right\|,\left\|\lambda_{k+1}-\lambda_{k}\right\|$ and the norm of the nonlinear constraint violation at the start of each major iteration.
8. The exit condition and a summary of the final solution.

Item 4 is preceded by a blank line, but item 5 is not. Nll items are illustrated in Figure 6.1, which shows the SUMMARY file for the test problem MANNE, using SUMMARY FREquENCY 1.


Figure 6.1. Format of SUMMARY file for test probleni MANNE


Figure 6.1 (continued). Format of SUMMARY file for test problem MANNE

## 7. SYSTEM INFORMATION

### 7.1 Distribution Tape

The source code and test problems for MINOS are distributed on a magnetic tape containing 14 files. The tape characteristics are described in a document accompanying the tape; normally they are: 9 track, 1600 bpi , unlabeled, ASCII, 80 -character records (card images), 3600 -character blocks.

The following is a list of the files and a summary of their contents. For reference purposes we give a name to each file. However, the names will not be recorded on unlabeled tapes. The HEAD and BODY files are composed of several smaller files deseribed in section 7.2.

| File Name | Type | Cards | Description |
| :---: | :---: | :---: | :---: |
| 1. HEAD 1 | FORTRAN | 923 | Source files 1 2: MIOOMAIN and MIIOMACH |
| 2. BODY 1 | Fortran | 12846 | Source files 3 13: MI15BLAS thru MI8ONCON |
| 3. HEAD2 | FORTRAN | 923 |  |
| 4. BODY2 | Fortran | 12846 |  |
| 5. HEAD3 | FORTRAN | 923 |  |
| 6. BODY3 | FORTRAN | 12846 |  |
| 7. Manne | data | 146 | SPECS and MPS files for test problem MANNE |
| 8. WEAPON | FORTRAN | 56 | Double-precision file MIOOMAIN for WEAPON |
| 9. WEAPON | SINGLE | 56 | Single-precision version of file 8 |
| 10. WEAPON | data | 151 | SPECS and MPS files for WEAPON |
| 11. Etamacro | FORTRAN | 135 | Double-precision file MIOOMAIN for ETAMACRO |
| 12. ETAMACRO | SINGLE | 135 | Single-precision version of file 11 |
| 13. ETAMACRO | SPECS | 38 | SPECS Rile for liTAMACRO |
| 11. Etamacro | MPS | 2134 | MPS file for ETSMACRO |

One HEAD and one BODY file should be selected for any given installation. HEADI and BODY1 are intended for machines using fortran declarations of the form

| IMPLICIT | REAL*8(A-H,O-Z) |  |
| :--- | :--- | :--- |
| DOUBLE PRECISION | Z(NWCORE) |  |
| INTEGER | KA(NKA) | (long integers) |
| INTEGER*2 | $\mathrm{HA}(\mathrm{NE})$ | (short integers) |

For example: IBM Systems 360, 370, 3033, 3081, etr.; Amdahl 170, Facom, Fujitsu, Hitachi, and other systems analogous to IBM; DEC VAX $11 / 750$ and $11 / 780$; 1)ata (ieneral MV/8000; ICl, 2900 series; recent PRIME systems.

HEAD2 and BODY2 are intended for machines using liortran declarations of the form
IMPLICIT DOUBLE PRECISION (A-H, O-Z)
DOUBLE PRECISION $Z$ (NWCORE)
INTEGER KA(NKA) (long integers only)
For example: DEC Systems 10 and 20; Honeywell systems; Univac 1100 series.
HEAD3 and BODY3 are intended for machines using Fortran declarations of the form

| REAL | $Z(N W C O R E)$ |
| :--- | :--- |
| INTEGER | KA (NKA) |

For example: Burroughs 6700 and 7700 series; CDC: 6000 and 7000 series and their Cyber counterparts; Cray-I.

## Installation Procedure

1. Obtain the appropriate $\operatorname{HEAD}$ and BODY files from the tape, along with the test data in file 7 (manne data).
2. Split the HEAD file into file MIOOMAIN and file MIIOMACH as suggested in section 7.2 .
3. If necessary, edit the subroutines in MIIOMACH according to section 7.4.
4. Decide whether or not to split the BODY file into files MI15BLAS through MIBONCON as suggested in section 7.2.
5. If all source code must be compiled together (e.g., with the Watfiv compiler), compile all the routines that were originally in the HEAD and BODY files, and run them on the test data in file manne data. Check the output against that shown in section 8.4.
6. If all source code can be compiled together and saved as a load module, and if various routines can later be compiled and linked to the load module, then do as described in step 5.
7. If individual routines cannot be recompiled to replace those in an already compiled collection, it is essential to compile the four routines in file MIOOMAIN separately: these contain the default user routines (appropriate for the test problem MANNE), and they will be replaced for other problems. Compile all remaining HEAD and BODY routines together (or separately if more convenient). In some circumstances it may be desirable to keep subroutines MIFILE, MINOSI and MINOS2 separate. Run the resulting code on the test data in file MANNE DATA, and check the output against that shown in section 8.4.
8. If further testing is desired, compile the appropriate WEAPON FORTRAN file and link it to the previously compiled MINOS code. Run it on the SPECS and MPS files contained in WEAPON DATA. See section 7.6 for a summary of the test problem results.
9. For a more demanding test, perform the same steps on the three relevant Eiamacro files.

### 7.2 Source Files

The source code for MINOS is intended to be acceptable to hoth Fortran 66 and Fortran 77 compilers, with a minimum of editing required for any partirular installation. Certain unavoidable machine dependencies are confined to a few short subrontines, involving lile definitions, word lengths, and end-of-file recognition. The only widespread dilliculty arises in the definition of character strings in DATA statements and format statements. For example,

```
DATA LWORD /4HWORD/
FORMAT(33H FANCY HAVING TO COUNT CHARACTERS)
```

is accepted by most Fortran 66 compilers, but may result in warning messages from Fortran 77 compilers. We have chosen to use quoles to delimil strings, as in

```
DATA LWORD /'WORD'/
FORMAT(' THIS STRING IS EASY TO TYPE')
```

since it is legal in Fortran 77 and it fails on very few Fortran 66 compilers. (In the past, CDC, compilers have allowed strings to be delimited by asterisks (*) rather than quotes.)

DATA statements are used to initialize integer variables to character strings in the manner just shown. The strings vary from 1 to 4 characters in longth. Implicitly lyped integers are therefore assumed to be at least 32 bits long. On some systems (e.f., Plalilli) this means that a rompiler option must be invoked to treat implicit integers ats "long". (E) (Bewhere in the soure conte, variables that are intended to be 16 bits long are explicitly typed INTEGER*2.)

The source code is divided into 13 logical parts. For ease of handling, these are combined into the HEAD and BODY files on the distribution tape, but for subsequent maintenance we recommend that 13 separate files be kept. In the description below we suggest a name for each file and summarize its purpose. We then list the names of the Fortran subroutines and functions involved. The naming convention used should minimize the risk of a clash with user-written routines.

File 1. MIOOMAIN Main program and default user routines. MAIN FUNOBJ FUNCON MATMOD

File 2. MIIOMACH Machine-dependent routines.
MIFILE MINOSI MINOS2 MIHASH MIINIT MIREAD
File 3. MI15BLAS Basic Linear Algebra Subprograms (a subset).
DASUM DAXPY DCOPY DDOT DNRM2 DSCAL
These routines are functionally similar to members of the BLAS package (Lawson, et al., 1979). Beware that they perform the correct function only when the parameters INCX and INCY are both equal to 1 (which is the only way MINOS uses them). If possible they should be replaced by authentic BLAS routines. There may exist versions that have been tuned to your particular machine.

DZERO HCOPY ICOPY
These are additional utility routines that could be tuned to your machine. DZERO is used the most, to set a vector to zero. If tuned versions of the BLAS are available, DzERO could call DCOPY with appropriate arguments.

File 4. mizoamat Constraint matrix coutines.

| M2APRD | M2APR5 | M2APR7 | M2APR8 | M2BSPR | M2CRSH | m2SCAL | WPK |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| M2UN | Matcol |  |  |  |  |  |  |

$\begin{array}{llllllllll} & \text { File 5. } & \text { MI25BFAC } & \text { Basis factorization routines. } & & & \\ & \text { LUIFAC } & \text { LU2FAC } & \text { LU3CP } & \text { LU4AC } & \text { LU4AR } & \text { LU5PQ1 } & \text { LU5PQ2 } & \text { LU5PQ3 } \\ & \text { LUBSOL } & \text { LU7KVR } & \text { LU7NVW } & \text { LUBRPC } & \text { M2BELM } & \text { M2BFAC } & \text { M2BMAP } & \text { M2BSOL }\end{array}$

| File 8. MI3OSPEC | SPECS file routines. |  |
| :--- | :--- | :--- |
|  | u3SPCO $\quad$ M3SPC1 | u3SPC2 |

File 7. MI35INPT Storage allocation and MPS file input.
H3CORE YSINPT MЗMPS YONAME MUREAD
File 8. MI40BFIL BASIS file input/output and SOLUTION printing.
M4GETB MADUMP M4INST M4LOAD N4NEWB MAOLDB MAPNCH M4SAVB

File 9. MI50LP Primal simplex method.
MSCHZR M5FRMC M5LOG M5LPIT M5PRIC M5SETP M5SETX M5SOLV

File 10. MIBOSRCH Linesearch and merit function.
GETPTC GETPTQ MERFUN MERGRD MERSAV SEARCH MGDCON MGDOBJ MEDMIY MEFCON M6FOBJ

File 11. MI65RMOD Maintaining the quasi-Newton factor $R$.
MGBFGS MGBSWP MGRADD MGRCND MERDEL MGRMOD MGRSET MERSOL M6SWAP

File 12. MI7ONOBJ Nonlinear objective; reduced-gradient algorithm. MTBSG M7BSX M7CHKD M7CHKG H7CHZQ MTRG MTRGIT MTSDIR u7SSCV

File 13. MIBONCON Nonlinear constraints; projected Lagrangian algorithm. mbajac mbaugl mbaugi u8chkJ mbprtj mbsetj

### 7.3 COMMON Blocks

Certain Fortran COMMON blocks are used in the MINOS source code to communicate between subroutines. Their names are listed below.

| M1EPS | Mfifie | M1WORD |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| M2File | M2LU1 | M2LU2 | M2LU3 | M2MAPA | M2MAPZ | m2Pary |  |
| M3LEN | M3LOC | M3MPS1 | M3MPS2 | M3MPS3 | M3MPS4 | M3SCAL |  |
| M5LEN | M5LOC | M5FREQ | M5LOBJ | M5log | M5LOG2 | M5log3 | M6LP1 |
| M5LP2 | M5PRC | M5TOLS |  |  |  |  |  |
| MTLEN | M7LOC | M7CG1 | M7C62 | m7CONV | M7PhES | M7tols |  |
| MELEN | MELOC | M8ALI | H8AL2 | M8diff | MEFUNC | masave | M ${ }_{\text {MVERI }}$ |
| CYCLCy |  |  |  |  |  |  |  |

A complete listing of the COMMON blocks and their contents appears in subroutine MINOS2. (Also see section 2.6). It may be convenient to make use of these occasionally; for example,

COMMON /MIFILE/ IREAD,IPRINT,ISUMM
gives the file numbers for the system reader and printer and for the st MMIMRY file. Otherwise, the naming convention should again minimize the risk of a clash with user-defined comuon blocks and subroutines.

As supplied, MINOS does not use blank COMMON. Howrver, in some installations it may be desirable to store the workspace array $\mathbf{z}$ there, as noted in the next section.

### 7.4 Machine-dependent Subroutines

Some of the routines in the HEAD file may require modification to suit a particular machine or a non-standard application. We discuss each of them in turn.

## The Main Program

The workspace for MINOS is allocated in the main program by code of the following form:

| DOUBLE PRECISION $\quad Z(10000)$ |  |
| :--- | ---: |
| DATA | NWCORE/10000/ |

C
CALL MINOSI ( 2,NWCORE ) RETURN
C
C END OF MAIN
END
Ten thousand words of storage are sufficient to solve small examples such as the test problems MANNE and WEAPON. About 25000 words are needed for ETAMACRO, which has approximately 400 constraints and 700 variables. For linear programs containing $m$ constraints, the length of $Z$ should be roughly 100 m , depending on the density of the constraint matrix. Nonlinear programs may require more workspace if there are many nonlinear variables.

On some machines it is possible to replace the main program by a non-Fortran routine that allocates storage for $Z$ at run-time.

For Burroughs installations, the main program should allocate $Z$ by calling an Algol procedure GETCOR (not provided), which in turn should call MINOS1 as above. This will overcome two problems in the process:

1. The binder can replace GETCOR in a compiled code file (but it cannot replace the main program).
2. The length of $Z$ is effectively unlimited if declared in an Algol procedure (but is restricted to be $\mathbf{6 5 5 3 5}$ or less when declazed in Fortran).
In some installations it will be desirable to put 2 in blank COMMON and then extend it at run-time if necessary. This could be done in MAIN, or in subroutine MINOS2 (see below).

On Honeywell machines, 2 must be in blank or labeled cOMmON to avoid a limit on the total storage for local variables ( 18 K words).

The CDC version of MAIN will need to begin with PROGRAM cards of the following general Corm:

```
PROGRAM MINOS ( INPUT, OUTPUT, TAPE5=INPUT, TAPE6=OUTPUT,
1 TAPE8. TAPE9, TAPE10. TAPE11,
2 TAPE12, TAPE13, TAPE14, TAPE16)
```

The unit numbers are suggested for use as follows:

| TAPE8 | The SCRATCH file (record length 8) |
| :--- | :--- |
| TAPE9 | A SUMMAR:' file (e.g., a terminal) |
| TAPE10 | An MPS file |
| TAPE11 | An OLD BASIS file and/or a BACKUP file |
| TAPE12 | ANFW BASIS file |

## TAPE13 An INSERT file or a LOAD file <br> TAPE14 A PUNCH file or a DUMP file <br> TAPE15 A SOLUTION file (record length 111)

## Subroutines FUNOBJ, FUNCON, MATMOD

These are the user routines described in Chapter 2. The HEAD file contains default versions that are appropriate for the test problem MANNE. They test if PROBLEM NUMBER 1111 has been specified, as in the file MANNE DATA. If not, they terminate the run with a message indicating that the required subroutine has not been loaded. (See exit 6 in section 6.3.)

## Subroutine MIFILE

This subroutine assigns explicit unit numbers to certain global filcs, namely the "card reader", the "line printer", the SCRATCH file, and the SPECS file. Typical values are 5, 6,8 , and 5 , but these will not be suitable for all installations.

In some cases (e.g., DEC 10 and 20), MIFILE must use explicit OPEN statements to open both the global files and certain others that are defined in the SPECS file for a particular problem, and to assign symbolic names to these files. In such cases, it may be sensible to let the user compile his own version of MIFILE each time MINOS is run.

For Burroughs installations, MIFILE may need to be compiled separately with some Burroughs FILE statements inserted at the start (since they must be the first statements that the compiler sees). The attributes in these FILE statements can be altered by the usual WFL statements associated with a run, but their default values cannot be altered by binding in a different version of MIFILE. It is therefore advisable to consider the FILE statements carefully before compiling the entire source code.

## Subroutine MINOS1

This opens some global files and then passes the array $\mathbf{z}$ to subroutine MINOS2 as many times as necessary, until a signal is given that all problems in the SPECS file have been processed.

For special applications, MINOS1 may need to be expanded. The most likely extension would be to call a matrix generator and a report writer (before and after the call to subroutine MINOS2).

## Subroutine winos2

For special applications, MINOS2 may requise modification. Again, one may wish to insert calls to a matrix generator or a report writer.

Some systems (e.g., CDC ) allow blank common to be extended at run-time. If so, a sensible place to do this is after the call to M3CORE, making use of the integer variables NWCORE, MINCOR and maxz. MINCOR will contain an estimate of the amount of storage required, and the user may assijn a value to MAXZ by means of a data card of the form

## WORKSPACE (TOTAL) 50000

in the SPECS file. See the in-line documentation for further details.

## Subroutine MiHASH

This subroutine should not require modification if a word containing four characters of leftjustificd data (read under A4 format) can be treated as a valid INTEGER.

On certain machines that regard INTEGER variables as a subset of the REALs, the characters must be right-shifted in order to produce a zero floating-point exponent. The in-line documentation points to four words requiring conversion.

For CDC and Cray installations, an offending word KEY1 may be converted into an acceptable integer K1 as follows:

DECODE ( 4,10,KEY1 ) K1
10 FORMAT (R4)
For Burroughs machines, the following statement has the required effect if KZERO $=0$ : K1 = CONCAT ( KZERO, KEY1,31,47,32 )
Burroughs installations need some further non-standard Fortran to facilitate character comparisons. In MiHASH, the statement

IF (KEY1 .EQ. NAME1 (KT) .AND. KEY2 .EQ. NAME2 (KT)) GO TO 60
must be changed to use ". IS." in place of ".EQ.", and the same change must be made to many ".EQ." tests in subroutine M3SPC1 (in the BODY file).

## Subroutine MIINIT

The variable EPS in this subroutine should be set to the relative precision of the machine's floating-point arithmetic, when it operates on words of the same type as the main storage array Z. Typical values are as follows:

| EPS | $=16.0 * *(-13)$ | for IBM systems, |
| :--- | :--- | :--- |
| EPS | $=2.0 * *(-55)$ | for DEC VAX $11 / 780$ with standard double precision, |
| EPS | $=2.0 * *(-61)$ | for DEC 10 and 20, |
| EPS | $=2.0 * *(-62)$ | for Honeywell systems, |
| EPS | $=2.0 * *(-59)$ | for Univac systems, |
| EPS | $=2.0 * *(-47)$ | for CDC and Cray systems, |
| EPS | $=2.0 * *(-37)$ | for Burroughs systems. |

Some VAX systems have additional double-precision hardware with slightly lower precision but a greater exponent range. Use of this option may be worthwhile, since it will essentially climinate some annoying (but otherwise harmless) occurrences of floating-point underflow.

## Subroutine M1READ

This subroutine contains a READ statement that is required to recognize an end-of-file condition (when reading the SPECS file). Most compilers allow the form

READ (ISPECS, 1000. END=900) L,LINE
RETURK
but CDC compilers require the statements
READ (ISPECS, 1000) L,LINE
IF (EOF(ISPECS)) 900. 100
100 RETURN
Note: Once installed correctly, subroutines M1HASH, VIINIT and M1READ will not be changed by the user. They should be treated the same as the subroutines in the BODY file.

### 7.5 Subroutine Structure

The first five levels of the subroutine hierarchy are shown below.


1. The main program allocates workspace and calls mimOSI.
2. wImOsi defines the READ, PRINT, SCRATCH and SPECS files via MIFILE, then calls MINOS2 once for each problem in the SPECS file.
3. mImos2 inputs the SPECS file and the MPS file, loads an initial basis, solves the problem (or a sequence of problems according to the CYCLE LIMIT), and finally saves BNSIS files and prints the solution.

### 7.6 Test Problems

## Tost Problem MANNE

This is a small example of an economic model due to Manne (1979). It has a nonlinear objective function, 10 nonlinear constraints, 10 linear constraints, and 30 variables. The nonlinearities are defined by the default function routines FUNOBJ and FUNCON in the MINOS source code. The starting point given in the MPS file is intentionally close to the optimum solution, to make this an inexpensive test problem. Other values in the INITIAL bounds set can be tried.

As supplied, FUNOBJ and FUNCON compute all gradients analytically if the SPIECS file specifics DERIVATIVE LEVEL 3. For test purposes, the first three nonzero gradients in each routine are not computed if DERIVATIVE LEVEL $=0$. We give a summary of the output produced by MINOS for the latter case. A full listing appears in section 8.4.

For this and later examples, the results were obtained on an IBM 3081 using the Fortran $H$ Extended (Enhanced) compiler with optimization level OPT=3.

| Maximum objective value: | 2.67009603 |
| :--- | :--- |
| Iterations to get feasible: | 1 |
| Total iterations: | 14 |
| Major iterations: | 3 |
| Evaluations of $F(x)$ and its gradient: | 21 |
| Evaluations of $f(x)$ and its Jacobian: | 24 |
| Number of superbasics at optimum: | 7 |
| CPU time (IBM 3081): | 0.3 seconds |

## The Weapon Assignment Problem, WEAPON

This problem has a nonlinear objective function and linear constraints. It is deweribed by Brarkrn and MrCormick (1969) and Himmelblau (1972). The constraint matrix is $12 \times 100$ and all 100 variables occur nonlinearly in the objective function $\mathscr{F}(x)$. The latlor drpords on 12 data cards which are read during the first entry to subroutine FUNOBJ.

The following are some solution statistics, obtained by $211 N 0.5$, 1 an $113 M 13081$ as noted above. They give an indication of the effort required to solve the problem. However, one should not expect to obtain identical results on some other machine.

| Minimum objective value: | -1735.56958 |
| :--- | :--- |
| Iterations to get feasible: | 3 |
| Total iterations: | 120 |
| Evaluations of $F(x)$ and its gradient: | 270 |
| Number of superbasics at oplimum: | 18 |
| CPU time (IBM 3081): | 2 seconds |

## Test Problem ETAMACRO (linear version)

This is one example of the energy model developed by Manne (1977). The constraint matrix is $401 \times 689$. To obtain a linear problem, we have included one linear objective row OPTIMALG in the MPS file. The latter also contains one right-hand-side vector RHS00001, and one bounds set BOUNDSO1.

The objective row OPTIMALG contains the optimal gradient values for the $\mathbf{8 0}$ nonlinear variables in the original (nonlinear) form of ETAMACRO. Hence the linear version of the problem has the same optimal dual variables $\pi$ as the nonlinear version (but rather different primal variables $x$ ).

The file EtAMACro SPECS is set up to solve this lincar program first. It asks for the linear variables and constraints to be scaled. (Note that it also asks for a BASIS map to be saved on unit 11 every 100 iterations. This may be used as a starting basis for the nonlinear version of the problem.)

Typical solution statistics follow.

| Maximum objective value: | $\mathbf{7 5 5 . 7 1 5 2 1 3}$ |
| :--- | :--- |
| Iterations to get feasible: | 240 |
| Total iterations: | 904 |
| CPU time (IBM 3081): | 15 seconds |

## Test Problem ETAMACRO (nonlinear version)

The objective function for the original form of the energy model is entirely nonlinear, and involves the first $\mathbf{8 0}$ variables. It is defined by subroutine funObJ in file ETAMACRO FORTRAN. It depends on 3 data cards which are included at the end of file ETAMACRO SPECS and are read during the first entry to F UNOBJ.

The MPS file docs not initialize any of the nonlinear variables. When started from the optimal solution to the preceding linear problem, typical solution statistics (with scaling requested) are as follows.

$$
\begin{array}{ll}
\text { Maximum objective value: } & 1337.72468 \\
\text { Iterations to get feasible: } & 0 \\
\text { Total iterations: } & 235 \\
\text { Evaluations of } F(x) \text { and its gradient: } & \mathbf{4 1 4} \\
\text { Number of superbasics at optimum: } & 28 \\
\text { CPU time (IBM 3081): } & \mathbf{7 s e c o n d s}
\end{array}
$$

From a cold start, with and without scaling, typical statistics are as follows.

|  | SCALE YES | SCALE NO |
| :--- | :--- | :--- |
| Maximum objective value: | 1337.72488 | 1337.72468 |
| Iterations to get feasible: | 235 | 213 |
| Total iterations: | 1022 | 1267 |
| Evaluations of $F(x)$ and its gradient: | 1271 | 1554 |
| Number of superbasics at optimum: | 28 | 28 |
| CPU time (IBM 3081): | 21 seconds | 26 |
|  |  |  |

## 8. EXAMPLES

The following sections define some example problems and show the input required to solve them using MINOS. The last example in section 8.4 is test problem MANNE as supplied on the distribution tape. For this example we also give the output produced by MINOS.

As the examples show, certain Fortran routines may be required to run a particular problem, depending on the problem and on the Fortran installation:

- A main program to allccate workspace
- Subroutine FUNOBJ to define a nonlinear objective function (if any)
- Subroutine FUNCON to define nonlinear constraint functions (if any)
- Subroutine MATMOD for special applications

The following input items are always required:

- A SPECS file
- An MPS file

Additional input may include a BASIS file and data read by the Fortran routines.
Load modules and file specifications are inevitably machine-dependent. A resident expert will be needed to install MINOS on your particular machine and to recommend job control or operating system commands. On some machines it will be possible to run lincar programs through MINOS without compiling any routines or linking thern to the MINOS code file. For nonlinear problems, some compilation and linking is unavoidable.

For some installations it may also be convenient to have your own copy of subroutine MIFILE, to define certain file attributes in (non-standard) Fortran, rather than via operating system commands. The resident expert will know best.

Good luck! We hope the examples that follow are general enough to set you on the right track.

### 8.1 Linear Programming

One of the classical applications of the simplex method was to the so-called diet problcm. Given the nutritional content of a selection of foods, the cost of each food, and the consumer's minimum daily requirements, the problem is to find the combination that is least expensive. The following example is taken from Chvátal (1983).

$$
\operatorname{minimize} c^{T} x \quad \text { subject to } \quad A x \geq b, \quad 0 \leq x \leq u
$$

where

$$
A=\left(\begin{array}{rrrrrr}
110 & 205 & 160 & 160 & 420 & 260 \\
4 & 32 & 13 & 8 & 4 & 14 \\
2 & 12 & 54 & 285 & 22 & 80
\end{array}\right), \quad b=\left(\begin{array}{r}
2000 \\
55 \\
800
\end{array}\right)
$$

and

$$
c=\left(\begin{array}{llllll}
3 & 21 & 13 & 9 & 20 & 10
\end{array}\right)^{T}, \quad u=\left(\begin{array}{llllll}
4 & 3 & 2 & 8 & 2 & 2
\end{array}\right)^{T} .
$$

Main program (not needed for some installations)

$$
\begin{aligned}
& \text { DOUBLE PRECISION } Z(10000) \\
& \text { DATA }
\end{aligned}
$$

C
CALL MINOSI ( Z.NWCORE ) STOP
END

Dummy user routines (not needed for some installations)

```
SUBROUTINE FUMOBS
ENTRY FUNCON
ENTRY MATMOD
RETURN
END
```


## SPECS File

```
BEGIN DIET PROBLEM
        MINIMIZE
        ROW8 20
        COLUMN8 30
        ELEMENTS 50
        SUMMARY FILE 9
        SUMMARY FREQUENCY 1 * (for amall problems only)
        NEW BASIS FILE 11
    END DIET PROBLEM
```

\section*{MPS File <br> | NAME <br> ROWS | DIET |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |
| G ENERGY |  |  |  |  |
| G PROTEIN |  |  |  |  |
| G Calcium |  |  |  |  |
| N COST |  |  |  |  |
| COLUMNS |  |  |  |  |
| OATMEAL | ENERGY | 110.0 | PROTEIN | 4.0 |
| OATMEAL | CALCIUM | 2.0 | COST | 3.0 |
| CHICKEN | ENERGY | 205.0 | PROTEIN | 32.0 |
| CHICKEN | Calsium | 12.0 | COST | 24.0 |
| EGGS | ENERGY | 180.0 | PROTEIN | 13.0 |
| EGGS | CAlcium | 54.0 | COST | 13.0 |
| MILK | ENERGY | 180.0 | PROTEIN | 8.0 |
| MILK | CALCIUM | 285.0 | COST | 9.0 |
| PIE | ENERGY | 420.0 | PROTEIN | 4.0 |
| PIE | CALCIUM | 22.0 | COST | 20.0 |
| PORKBEAN | Energy | 280.0 | PROTEIN | 14.0 |
| Porkbean | CALCIUM | 80.0 | COST | 19.0 |
| RHS |  |  |  |  |
| DEMANDS | ENERGY | 2000.0 | PROTEIN | 55.0 |
| DEMANDS | CALCIUM | 800.0 |  |  |
| BOUNDS |  |  |  |  |
| UP SERVINGS | OATMEAL | 4.0 |  |  |
| UP SERVINGS | CHICKEN | 3.0 |  |  |
| UP SERVINGS | EGGS | 2.0 |  |  |
| UP SERVINGS | MILK | 8.0 |  |  |
| UF SERVINGS | PIE | 2.0 |  |  |
| UP SERVINGS | PORKBEAN | 2.0 |  |  |
| ENDATA |  |  |  |  |

## Notes on the Diet Problem

1. For small problems such as this, the SPECS file does not really need to specify cortain parameters, because the default values are large enough. However, we include them as a reminder for more substantial models.
2. In the MPS file we put the objective row last. Looking ahead, this is one way of cusuring that it does not get mixed up with nonlinear constraints, whose names must appear lirst in the ROWS section.
3. The constraint matrix is unusual in being $100 \%$ densc. Most models have at least a few zeros in each column and in $b$. They would not need to appear in the COLUMNS and RIIS sections.
4. MINOS takes three iterations to solve the problem. The optimal objective is $\boldsymbol{c}^{\boldsymbol{T}} \boldsymbol{r}=92.5$. The optimal solution is $x=(1,0,0,4.5,2,0)^{T}$ and $s=(0,-5,-534.5)^{T}$. The optimal dual variables are $\pi=(0.05625,0,0)^{T}$.

### 8.2 Unconstrained Optimization

The following is a classical unconstrained problem, due to Rosenbrock (1960):

$$
\operatorname{minimize} F(x)=100\left(x_{2}-x_{1}^{2}\right)^{2}+\left(1-x_{1}\right)^{2}
$$

We use it to illustrate the data required to minimize a function with no general constraints. Bounds on the variables are easily included; we specify $-10 \leq x_{1} \leq 5$ and $-10 \leq x_{2} \leq 10$.

## Calculation of $F(x)$ and its gradients

```
SUBROUTINE FUNOBJ( MODE, N, X, F, G, NSTATE, NPROB, Z, NWCORE )
IMPLICIT REAL*O(A-H,O-Z)
DOUBLE PRECISION X(N),G(N), Z(NWCORE)
```

C
C ROSENBROCK'S BANANA FUNCTION.
C
$X 1=X(1)$
$X 2=X(2)$
$\mathrm{T} 1=\mathrm{X} 2-\mathrm{X} 1 * * 2$
$\mathrm{T} 2=1.0-\mathrm{X} 1$
$\mathrm{F}=100.0 * \mathrm{~T} * * 2+\mathrm{T} 2 * * 2$
$G(1)=-400.0 * \mathrm{~T} * \mathrm{X} 1-2.0 * \mathrm{~T} 2$
$G(2)=200.0 * T 1$
RETURN
C
C END OF FUNOBJ FOR ROSENBROCK END

## SPECS File

BEGIN ROSENBROCK
OBJECTIVE = FUNOBJ
NONLINEAR VARIABLES 2
SUPERBASICS LIMIT 3

LOWER BOUND -10.0
UPPER BOUND 10.0

SUMMARY FILE 9
SUMMARY FREQUENCY 1 ITERATIONS LIMIT 50

END ROSENBROCK

## MPS File

```
NAME
    ROSENBROCK
ROWS
    N DUMMYROW
COLUMNS
            X1
            X2
RHS
BOUNDS
    UP BOUND1 X1 5.0
    FX INITIAL X
                                    -1.2
    FX INITIAL X2 1.0
ENDATA
```


## Notes on Rosenbrock's function

1. There is nothing special about subroutine $F U N O B J$. It returns the function value $F(x)$ and two gradient values $g$, $=\partial F / \partial x$, on every entry. If $G(1)$ or $G(2)$ were not assigned values, MINOS would "notice" and proceed to estimate either or both by finite differences.
2. The SPECS file apparently does not need to estimate the dimensions of the constraint matrix A, which is supposed to be void anyway. But in fact, MINOS will represent $A$ as a $1 \times n_{1}$ matrix containing $n_{1}$ elements that are all zero. For very large unconstrained problems, the COLUMNS and ELEMENTS keywords must be specified accordingly.
3. The SPECS file must specify the exact number of nonlinear variables, $n_{1}$. To allow a little elbow room, the SUPERBASICS LIMIT must be set to $n_{1}+1$, unless it is known that some of the bounds will be active at the solution.
4. The MPS file must specify at least one row. Here it is a dummy free row (type $\mathbf{N}==$ nonbinding constraint). The basis matrix will remain $B=1$ throughout, corresponding to the slack variable on the free row.
5. The COLUMNS section contains just a list of the variable names. The RIIS header card must appear, but a free row has no right-hand-side entry.
6. Uniform bounds $-10 \leq x, \leq 10$ are specified in the SPECS file as a matter of good prutior. Their presence does not imply additional work. If the LOWER and UPPER BOUND keywords did not appear, the variables would implicitly have the bounds $0 \leq x, \leq \infty$, which will not always be appropriate.
7. With the uniform bounds specified, only one additional rard is needed in the BOI'NI)s aection to impose the restriction $x_{1} \leq 5$.
8. The INITIAL bound set illustrates how the starting point $\left(x_{1}, x_{2}\right)=(-1.2,1.0)$ is speciliod. These cards must appear at the end of the IBOUNDS section. Since the SUPERBASICS LIMIT is sufficiently high, both variables will initially be superbasic at the indicated values.
9. If the INITIAL bound set were absent (and if no BASIS file were loaded), $x_{1}$ and $x_{2}$ woulid initially be nonbasic at the bound that is smaller in absolute value (with ties broken in favor of lower bounds); in this case, $x_{1}=u_{1}=5$ and $x_{2}=l_{2}=-10$.
10. From the standard starting point shown, a quasi- Newton method with a moderately acrura'd linesearch takes about 20 iterations and 60 function and gradient evaluations to reach the unique solution $x_{1}=x_{2}=1.0$.



MICROCOPY RESOLUTION TEST CHART mational bureau of standards-1963-A

### 8.8 Linearly Constrained Optimization

Quadratic programming (QP) is a particular case of linearly constrained optimization, in which the objective function $F(x)$ includes linear and quadratic terms. There is no special way of informing MINOS that $F(x)$ is quadratic, but the algorithms in MINOS will tend to perform more efficiently on quadratics than on other nonlinear functions. The following items are required to solve the quadratic program

$$
\text { minimize } F(x)=\frac{1}{2} x^{T} Q x+c^{T} x \quad \text { subject to } \quad A x \leq b, \quad x \geq 0
$$

for the particular data

$$
Q=\left(\begin{array}{lll}
4 & 2 & 2 \\
2 & 4 & 0 \\
2 & 0 & 2
\end{array}\right), \quad c=\left(\begin{array}{c}
-8 \\
-6 \\
-4
\end{array}\right), \quad A=\left(\begin{array}{lll}
1 & 1 & 2
\end{array}\right), \quad b=3 .
$$

## Calculation of quadratic term and its gradients

subroutine funobj ( mode, n, X, F, G, mstate, nprob, 2, micore )
IMPLICIT REAL*B(A-H,0-Z)
DOUBLE PRECISIOM $X(N), G(N), 2(N W C O R E)$
COMMON /GPCOMY/ Q(50,50)
c
$C$ Computation of $F=1 / 2 \times \prime Q x, \quad g=9 x$.
C The COMMON statement and subroutine SETQ are problem dependent.
c
c
if (nstate .eq. 1) CALl setq( Q, 50, N)
$F=0.0$
c
DO $200 \mathrm{I}=1$,
GRAD $=0.0$
DO $100 \mathrm{~J}=1$, w
GRAD $=G R A D+Q(I, J) * X(J)$
100 COMTIMUS
$\mathrm{F}=\mathrm{F}+\mathrm{X}(\mathrm{I}) *$ GRAD
$G(I)=$ CRAD
200 COMTINUE
c

$$
\begin{aligned}
& \mathrm{FETUR} \\
& \text { R }
\end{aligned}=0.5 * F
$$

c
C EMD OR FUNOBJ FOR QP EnD

## SPECS Filo

```
BEGIM QP
    NONLIMEAR VARINBLES 3
    SUPERBASICS LIMIT 3
    SUMMARY FILE O
    SUMMARY FREQUEMGY 1
    ITERATIONS LIMIT }5
EMD QP
```


## MPS File

| MAME | QP |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
| ROW8 |  |  |  |  |
| L A |  |  |  |  |
| W C |  |  |  |  |
| colunins |  |  |  |  |
| X2 | A | 1.0 | C | -8.0 |
| X2 | A | 1.0 | C | -6.0 |
| X3 | A | 2.0 | c | -4.0 |
| RH8 |  |  |  |  |
| B | A | 3.0 |  |  |
| EmDATA |  |  |  |  |

## Notes on the QP example

1. In subroutine FUNOBJ we assume that the array $Q(*, *)$ is initialized during the first entry by another subroutine SETQ, which is problem-dependent. The COMMON statement is also problem-dependent and is included to ensure that $Q$ will retain its values for later entrics. (In some Fortran implementations, local variables are not retained between entrics.)
2. The quadratic form will often involve only some of the variables. In such cases the variables should be ordered so that the nonzero rows and columns of $Q$ come first, thus:

$$
Q=\left(\begin{array}{ll}
\boldsymbol{Q} & \\
& 0
\end{array}\right)
$$

3. The parameter $M$ and the number of NONLINEAR VARIABLES would then be the dimension of Q.
4. FUMOBJ could have computed the linear term $c^{T} \boldsymbol{x}$ (and its gradient $c$ ). However we have included $c$ as an objective row in the MPS file, in the same manner as for linear programs. This is more general, because c could contain entries for all variables, not just those associated with 8.
5. Beware-if $c \neq 0$, the factor $\frac{1}{2}$ makes a vital difference to the function being minimized.
6. The optimal solution to the QP problem as stated is

$$
x=(1.3333,0.77777,0.44411), \quad \frac{1}{2} x^{T} Q x=8.2222, \quad c^{T} x=-17.111 \quad F(x)=-8.8888
$$

## Teot Probleme WEAPON and ETAMACRO

The MINO8 diatribution tape contains dats for these two linearly constrained problema. The SPECS Ale for ETAMACRO is as followa. It in set up to solve a linear form of the problem firus, and thea uee the optimal becio as a starting point for the nonlinear form.


## Lhomer Leent Squeree

Deta-litheg eas dive rive to a comotrained linear leant-equares problem of the form
minamise $\|X z-y\| z$ subject to $A s \geq b, \quad 1 \leq x \leq a$.
This problem may be solved with MaNOS an it staads, by coding subroutine rumond to compute
 matris, it may be more coaveaicat to exprome the problem in the form
misimise $F(r)=\frac{1}{2} r_{r} \quad$ subject to $\quad\left(\begin{array}{ll}I & X \\ & A\end{array}\right)\binom{r}{x} \geq\binom{ v}{i}, \quad r$ free, $l \leq x \leq w$.

## Notes on the least-squares problom

1. As usual, the constraints in $A x \geq b$ may include all types of inequality.
2. $r=y-X x$ is the residual vector and $r^{T_{r}}$ is the sum of squares.
3. The objective function is easily programmed as $F(r)=\frac{1}{2} r^{T} r$ and $g(r)=r$.
4. More stable methods are known for the least-squares problem. If there are no constraints at all, several codes are available for minimizing $\|X x-y\|_{2}$ when $X$ is either dense or sparse. When there are equality constraints only ( $A x=b$ ), we know of one specialized method that can treat $X$ and $A$ as sparse matrices (Björck and Duff, 1980). For the more general case with inequalities and bounds, MINOS is one of very few systems that could attempt to solve problems in which $X$ and $A$ are sparse. However, if $n$ (the dimension of $x$ ) is very large, MINOS will not be efficient unless almost $n$ constraints and bounds are active at the solution.
5. If it is expected that most of the elements of $x$ will be away from their bounds, it will be worthwhile to specify MULTIPLE PRICE 10 (say). This will allow up to 10 variables at a time to be added to the set currently being optimized, instead of the usual 1.

## The Diserete $l_{1}$ Problem

An apparently similar data-fitting problem is

$$
\text { minimize }\|X x-y\|_{1} \quad \text { subject to } \quad A x \geq b
$$

where $\|r\|_{1} \equiv \sum\left|r_{i}\right|$. However, this problem is best solved by means of the following purely linear program:
subject to

## Notes on the $l_{1}$ problem

1. The solution $x$ is recovered as the dual variables, i.c., the Lagrange multipliers associated with the general constraints.
2. The optimal value of $\|X x-y\|_{1}$ is the sum of the absolute values of the reduced costs aseociated with $\lambda$. (It is also the maximal value of $y^{\boldsymbol{T}} \lambda+b^{\boldsymbol{T}} \mu$.)
3. If a particular row in $A x \geq 6$ is required to be an equality constraint, the corresponding component of $\mu$ should be a frec variable.
4. It does not appear simple to include the bounds $1 \leq x \leq u$ except as part of $\boldsymbol{A x} \geq$ b. If there are many finite bounds, it may be beat to solve the original problem directly as a lincar program, thua:
subject to $\quad\left(\begin{array}{rl}A \\ I-I & X\end{array}\right)\left(\begin{array}{l}r \\ s \\ x\end{array}\right) \geq\binom{ b}{y}, \quad r, s \geq 0, \quad l \leq x \leq u$, where $e^{T}=\left(\begin{array}{lll}1 & 1 & \ldots\end{array}\right)$.

### 8.4 Nonlinearly Conatrained Optimization

Two example problems are deacribed here to illustrate the subroutines and data required to specify a problem with nonlinear constraints. The first example is small, dense and highly nonlinear; it shows how the Jacobian matrix may be handled most simply (as a dense matrix) when there are very few nonlincar 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.

Problem MHW4D (Wright (1976), example 4, starting point D)

$$
\begin{aligned}
& \text { minimize } \quad \begin{aligned}
&\left(x_{1}-1\right)^{2}+\left(x_{1}-x_{2}\right)^{2}+\left(x_{2}-x_{3}\right)^{3}+\left(x_{3}-x_{4}\right)^{4}+\left(x_{4}-x_{5}\right)^{4} \\
& \text { subject to } \quad x_{1}+x_{2}^{2}+x_{3}^{3}=3 \sqrt{2}+2 \\
& x_{2}-x_{3}^{2}+x_{4}=2 \sqrt{2}-2 \\
& x_{1} x_{5}=2
\end{aligned}
\end{aligned}
$$

Starting point:

$$
x_{0}=(-1,2,1,-2,-2)
$$

## Notes for problem MHW4D

1. The function subroutines include code for a second problem (Wright, 1976, example 9). The parameter KPROB is used to branch to the appropriate calculation.
2. In subroutine FUNOBJ, $F$ is the value of the objective function $\boldsymbol{F}(\boldsymbol{x})$ and G contains the corrosponding 5 partial derivatives.
3. In subroutine FUNCON, F is an array containing the vector of constraint functions $f(x)$, and G holds the Jacobian matrix; thus, the $i$-th row of $\mathbf{G}$ contains the partial derivatives for the $i$-th 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 $\mathbf{G}$ are zero; they do not need to be explicitly set.
4. Subroutine FUNCON will be called before subroutine FUNOBJ. The parameter KSTATE is used to print a message on the very first entry to FUNCON. This is just a matter of good practire, since it is often convenient to compile MINOS and the function routines into an executable code file, and one can easily 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 COIUMNS section of the MPS file contains only the names of the variables, since they are all "nonlinear", and because there are no lincar constraints.
7. The BOUNDS section specifies only the initial point. Uniform bounds on the variables are given in the SPECS fle.
8. Since PX indicatora are used for the IMITIAL bounds, the SUPERBASICs LImIT needs to be at least 5 in this case, plus 1 for elbow room during the optimization.
9. This example has several local minima, and the performance of MINOS is very dependent on the initial point $x_{0}$. See Wright (1976) or Murtagh and Saunders (1982) for computational details.

Preblom MHW4D; computation of the eljective function


```
        IMPLIEIT meAKEASA-NoO-2)
```



```
C
M Nam 4
    T1 = X(1) - 1.0
    T2=x(1)-x(2)
    T3=x(8)-x(3)
    T4 = X(3) - X(4)
    TS = X(4) - X(S)
C
```



```
    O(1): 2.0m(Ti * TE)
    B(2) = -2.0w72 3.0wT3ment
```



```
    (4) = -4.0wfyms * ©.0wflums
    B(S)= - .0.0.7'mus
    RETUM
C
500 T1 = 03IMKX(3) - X(31)
T2 = Dcos{x(S)-x(3i)
```




```
        0(1)= 10.0mx(4) * 3.0mx(1)mene x(2)
        B(2) = X(1)mms - 12.0nx(2)mx(3)
        1 + 3.0mx(2)mene x(4)mme x(3)mma
        8(3) = -6.0mx(8)wwe - 9.0mTE
        e(4) = 10.00x(1) * 2.0nx(2)wns m X(4) X(3)m04
```



```
        NETMN
C
```



```
    CN
```

Prekion MHWAD; compritation of the cemotraint frenctions


```
    IMPLEET NEALmA(A-N,O-21
    Dovis mmersmon x(N);F(N),O(M,N),z(NNcone)
6
    NMN*
```



```
    IF INMNC..N. ©) Co To see
    f(1) = X(1) + X(E)mane t X(3)mms
    O(1,1)=1.0
    e(1.2) = 2.0mx(2)
    O(1,3)= 3.0nx(3)mane
E
    F(2) = X(2) - X(3)mane + X(4)
    C(2,2) = 1.0
    O(2,3)=-2.0m\times(3)
    C(2,4)=1.0
c
    F(3) = X(1)30*(3)
    O(3,1) = X(S)
    8(3,3)= X(1)
    RETUN
C
```



```
B(1,1)=2.0mx(1)
e(1,E)=2.0mx(2)
8(1.3) = 2.0nx(3)
B(1.4)=2.0mx(4)
GCi,S) = 2.0.0x(SI
C
```



```
    8(2.1)=2.0nx(1) matas)
    e(2,3) a x(i) \me
    C(2,4) = X(3)
    O(2,8)=x(4)
c
    F(3) = X(2.9mamx(4) * 19.0wx(1)mx(s)
    (3.1) = 10.00x(3)
    8(3.8)= 2.0mx(8),0x(4)
    C(3,4) E X(8)mme
    C(3.5) - 10.0mxil)
    nETTMN
E
```



```
    EN ep Flucon Pe. mamuros
    EN
```

Problem MHW4D; the SPECS Ne

| EESIN MMH 4 | - |
| :---: | :---: |
| Nownmen cowsimarnts | 3 |
| Monlinear vaniasles | 5 |
| Jaccesan | DEver |
| UNFEX Doun | 8.8 |
| Lamb boun | - 5.0 |
| semunty ple | - |
| TTEMATIONS | 103 |
| Mald ITERATICM | 15 |
| MENOM ITERATEGE | 10 |
| mancti parancten | 1.0 |
| Sumamasics Lxut | 6 |
| CumT LEVEL (JPURS | 181 |
| VEREFY LEVEL | - |
| 4 D R Mam 4 |  |

Preblem MHW4D; the MPS the

| mare | nom 90 |  |
| :---: | :---: | :---: |
| Rows |  |  |
| 8 Cont |  |  |
| E cour |  |  |
| E CON3 |  |  |
| colunes |  |  |
| ${ }_{x 1}$ |  |  |
| x |  |  |
| X3 |  |  |
| $x$ |  |  |
| $x$ |  |  |
| mus | cown | 6.24863 |
| mus | cous | 0.ecene |
| 3 | cows | 2.0 |
| FX Enetzal | X1 | -1.0 |
| PX Enctial | X2 | 2.0 |
| Px mertal | ${ }^{x}$ | 1.0 |
| PX zmitzal | $x$ | -2.0 |
| PX InItial | $x$ | -8.0 |
| eveata |  |  |

Problem MANNE (Manne, 1979)

|  | maximise | $\sum_{t=1}^{T} \beta_{t} \log C_{t}$ |
| :---: | :---: | :---: |
|  |  |  |
|  |  |  |
| subject to | $\alpha_{t} K_{t}^{b} \geq C_{t}+I_{t}$, | $1 \leq t \leq T$, |$\quad$ (nonlinear constraints) $\quad$ (linear constraints)

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. The first $T$ constraints are nonlincar because the variables $K_{t}$ are raised to the power $b=0.25$. The problem is described more fully in Murtagh and Saunders (1982), where results are given for the case $\boldsymbol{T}=\mathbf{1 0 0}$.

The main program and subroutines shown on the following pages are part of the file HENDI on the MINOS distribution tape (see sections 7.1 and 7.4). The SPECS data and MPS data are contained in the file MANNE DATA; they apply to the case $\boldsymbol{T}=\mathbf{1 0}$.

## Notes for problem MANNE

1. For efficiency, the Jacobian variables $K_{\mathrm{t}}$ are made the first $\boldsymbol{T}$ components of $\boldsymbol{x}$, followed by the objective variables $C_{t}$. Since the objective does not involve $K_{t}$, subroutine Fowobj must set the first $\boldsymbol{T}$ components of the objective gradient to zcro. The parameter $\mathbb{N}$ will have the value $2 T$. Verification of the objective gradients may as well start at variable $\mathbf{T}+1$.
2. For subroutine FUNCON, $N$ will be $T$. The Jacobian matrix is particularly simple in this example; in fact $J(x)$ has only one nonsero element per coluinn (i.e., it is diagonal). The parameter WJAC will therefore be $T$ also. It is used only to dimension the array $\mathbf{G}$.
3. NSTATE enables $\mathrm{B}, \mathrm{AT}$ and BT to be initialized on the first entry to FUNCON, for subsequent use in both of the function subroutines. (Remember that the first call to FUNCON orcurs before the first call to $\operatorname{FNOBOB}$.) 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 7.3.
4. NSTATE is also used to produce some output on the final call to FUNCON.
5. The COMMOM block MIFILE is one of those used by MINOS; see section i.6. For lest purposes we also use COMMOM block MEDIFF to access the variable LDERIV.
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 $\boldsymbol{T} \times \mathbf{T}$ Jacobian in the top left corner of the constraint matrix. An arbitrary value of 0.1 has been used for the nonzero variable corfficients. A acro or blank numeric field would be equally good.

Problem MANNE; main program and calculation of the objective function
C THIS 15 THE DEFAULT MAIN
C
C IT SHOULD PROVIDE AS MUCM MOMKSPACE AS SEELis APmomiate.
DONBLE PRECISION $2(10008)$
DATA
c
CALL HINDSII Z,NGCORE )
stop
${ }_{c}$
END OF MAXN
END

##  <br> IMPLICIT REALMA $A-H, O-2)$ <br> DOLSLE PRECISION $X(N), E(N), Z(N E C O N E)$

c
COHTION MBEOTFF/ DIFINT(2), EDUNT, LDERIV,LVLDIF

cosical exnown


Problom MANNE; calculetion of the constraint Pronetione


Problem MANNE; the SPECS the


Problom MANNE; the MPS tive

| MNS | Manctis |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |
| - Moveres |  |  |  |  |
| - Mowoer |  |  |  |  |
| - Movoes |  |  |  |  |
| - Mamer |  |  |  |  |
| - Movees |  |  |  |  |
| - nowaes |  |  |  |  |
| - Manver |  |  |  |  |
| - Mrwoes |  |  |  |  |
| - Howees |  |  |  |  |
| - Monoio |  |  |  |  |
| L capere |  |  |  |  |
| 4 capees |  |  |  |  |
| 1 Capoot |  |  |  |  |
| 1 capops |  |  |  |  |
| 4 Cap006 |  |  |  |  |
| L CAP007 |  |  |  |  |
| 1 capoos |  |  |  |  |
| 1 capoos |  |  |  |  |
| 1 capolo |  |  |  |  |
| 1 TENHINY |  |  |  |  |
| countes |  |  |  |  |
| Rapoes | Howeot | . 1 | capoet | 1.0 |
| Rapoes | Capooz | -1.0 |  |  |
| Kapoet | Meweot | . 1 | capeor | 1.0 |
| Kapoer | Capaes | -1.0 |  |  |
| Rapeos | Howeos | . 1 | capeos | 1.0 |
| Rapoes | Capoos | -1.0 |  |  |
| Rapoes | mownes | .1 | capeos | 1.0 |
| Rapoge | capoes | -1.0 |  |  |
| Kapeos | Hownes | . 1 | capeos | 1.8 |
| Kapoes | capoes | -1.0 |  |  |
| Kapees | Mowoes | . 1 | capoes | 1.0 |
| KAP006 | Capoer | -1.0 |  |  |
| Rapoot | Moweer | . 1 | Capue 7 | 1.8 |
| Kapoe7 | camose | -1.0 |  |  |
| Kapoes | Monges | . 1 | Cap008 | 1.0 |
| Kapoes | Capoos | -1.0 |  |  |
| Kapoes | noweos | . 1 | CAP009 | 1.0 |
| Kap009 | capole | -1.0 |  |  |
| Kapoto | Mowole | . 1 | CAP010 | 1.0 |
| Kapole | TERHINY | . 03 |  |  |
| conves | meveet | -1.0 |  |  |
| cowper | trwoer | -1.0 |  |  |
| compe3 | Mapeos | -1.0 |  |  |
| comal | mimeet | -1.0 |  |  |
| coneos | prevees | -1.0 |  |  |
| coswes | noveres | -1.0 |  |  |
| cowne | maveer | -1.0 |  |  |
| conece | mawee | -1.0 |  |  |
| covees | Hevees | -1.0 |  |  |
| comple | mever | -1.0 |  |  |
| Envees | Mowees | -1.0 | capees | -1.0 |
| saveos | trueer | -1.0 | capees | -1.0 |
| Invees | Hemees | -1.0 | capoes | -1.0 |
| Inveer | Hiviost | -1.0 | Capeos | -1.0 |
| Twvees | frovees | -1.0 | capoes | -1.0 |
| sinvoes | Mowees | -1.0 | Capee7 | -1.0 |
| INV007 | Hiveer | -1.0 | capoes | -1.0 |
| Invoes | nowees | -1.0 | capeos | -1.0 |
| Invoes | Hovees | -1.0 | capol | -1.0 |
| Invor | Hovele | -1.0 | CAPOt 1 | -1.0 |
| Inves | TERMIM | -1.0 |  |  |

Probinm MANNE; the MPS Pie, continued


Problom MANNE; output from MINOS



Problem MANNE; output from MINOS, continued


ENITIAL Casis



ITEMATEM


Problom MANNE; eutput frem MINOS, contiaued

Tin sucasm seme re te m.






mexam meter to





Problem MANNE; output from MINOS, continued




Problom MANNE; output from MINOS, continued


### 8.5 Use of Subreutine Matu00

The following example illuatrates the construction of a sequence of problems, based on the Diet problem of section 8.1. It assumes that the following cards have been added to the SPECS file:

```
CYCLE LIMIT
3
CYCLE PRINT 
CYCLE TOLERANCE 2.0
PHAMTOM COLUNMS 1 (or more)
phantOM ELEMEMT8 3 (or more)
```

1. Solution of the original problem constitutes cycle 1.
2. After cycle 1 , MATMOD will be called twice with WCYCLE $=2$ and 3 respectively, denoting the beginning of cycles 2 and 3 . The value of $\boldsymbol{M}$ will include the normal columne and the phantom columns; in this case, $\mathrm{M}=6+\mathbf{1}=7$. Likewise, $\mathbf{W E}$ includes normal and phantom elementa; in this case, $\mathrm{ME}=24+3=27$.
3. For cycle 2, we alter the cost coefficient on the variable called CHICXEM. This happens to be the second variable, but for illustrative purposes we use the MINOS subroutine m3mave to search the list of column names to find the appropriate iadex. In this case, щ3Mare will return the value JCHICX $=2$.
4. Similarly, we use M3MAME to scarch the list of row names to find the index for the objective row, whose name is known to be COST. In this case, M3MAME will return the value JCOST = 11. Since rows are stored after the M columns, this means that the objective is row number JCOst $-n=4$. (As it happens, this value is alrcady available in the comiom variable IOBJ.)
5. This example assumes that CHICKEM already had a nonzero cost coefficient, since it is not possible to increase the number of entries in existing columns. If the cost coefficient was previously sero, it would have to be entered as such in the MPS file, and the SPECS file would have to set AIJ TOLERAMCE $=0.0$ to prevent zero coefficients from being rejected.
6. For cycle 3, we gencrate one new column by calling upon the MINOS subroutine MATCOL. The PHANTOM COLUMMS and PHANTOM ELEMENTS keywords must define sufficient storage for this new column. (The eatimates defined by the normal COLUMM and ELEMENTS keywords must also allow for the phantom columns and elementa.)
7. For illustrative purposes, we make use of the specified CYCLE TOLERAMCE and the value of $X(1)$ in the current solution, to decide whether to proceed with cycle 3.
8. After the call to matcol, the COMOW variable Jmel points to the new column. It allows us to set a finite upper bound on the aseociated variable. If there had been insufficient storagr, or if $\mathbf{c o l}($ ( $)$ contained no significant elements, materr would have been increased from 0 to 1. Usually, this means that the sequence of cycles should be terminated (by welting FINISH = .tive.).



### 3.6 Things to Remember

Use the following space to record the fruits of your experience. They may be useful reminders the next time you come to run MINOS. (We suggest you use a pencil.)

## REFERENCES

Bartels, R. H. (1971). A stabilisation of the simplex method, Num. Math. 16, pp. 414-434.
Bartels, R. H. and Golub, G. H. (1969). The simplex method of linear programming using the LU decomposition, Comm. ACM 12, pp. 266-268.
Björck, $\AA$. and Duff, I. S. (1080). A direct method for the solution of aparse linear least squares problems, Linear Algebra and its Applics. 34, pp. 43-67.

Bracken, J. and McCormick, G. P. (1968). Selected Applications of Nonlinear Programming, John Wiley and Sons, New York and Toronto.
Chvátal, V. (1983). Linear Programming, W. H. Freeman and Company, New York and San Francisco.

Dantzig, G. B. (1963). Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey.
Davidon, W. C. (1959). Variable metric methods for minimization, A. E. C. Res. and Develop. Report ANL-5990, Argonne National Laboratory, Argonne, Illinole.
Gill, P. E., Murray, W. and Wright, M. H. (1981). Practical Optimisation, Academic Press, London.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1979). Two step-length algorithma for numerical optimization, Report SOL 79-25, Department of Operations Research, Stanford University.
Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1984). LUSOL: A package for updating a sparse $L U$ factorization (to appear).

Himmelblau, D. M. (1972). Applied Nonlinear Programming, McGraw-Hill.
Lawson, C. L., Hanson, R. J., Kincaid, D. R. and Krogh, F. T. (1979). Basic Linear Algebra Subprograms for Fortran usage, ACM Trans. Math. Software 5, pp. 308-323 and (Algorithm) 324-325.

Manne, A. S. (1977). ETA-MACRO: A Model of Energy-Economy Interactions, in C. J. Hitch (ed.), Modeling Energy-Economy Interactions, Resources for the Future, Washington, D.C. Also appears as pp. 205-233 in R. Pindyck (ed.), Advances in the Economics of Energy and Resources, Vol. 2: The Production and Pricing of Encrgy Resources, JAI Press, Inc., Greenwich, Connecticut, 1979.
Manne, A. S. (1979). Private communication.
Murtagh, B. A. and Saunders, M. A. (1978). Large-scale linearly constrained optimization, Math. Prog. 14, pp. 41-72.
Murtagh, B. A. and Saunders, M. A. (1982). A projected Lagrangian algorithm and its implementation for aparse nonlincar constraints, Math. Prog. Study 16, Algorithms for Constrained Minimisation of Smooth Nonlinear Functions, pp. 84-117.

Preekel, P. V. (1980). Modules for usc with MNOS/AUGMENTED in solving sequences of mathematical programs, Report SOL 80-15, Department of Operationa Remearch, Stanford Uaiversity.

Reid, J. K. (1976). Fortran subroutines for handling sparse linear programming bases, Report R8269, Atomic Energy Research Establishment, Harwell, England.

Reid, J. K. (1982). A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases, Math. Prog. 24, pp. 55-69.

Robinson, S. M. (1972). A quadratically convergent algorithm for general nonlinear programming problems, Math. Prog. 3, pp. 145-156.
Rosen, J. B. and Kreuser, J. (1972). A gradient projection algorithm for nonlinear constraints, in Numerical Methods for Non-Linear Optimisation (F. A. Lootsma, ed.), pp. 297-300, Academic Prese, London and New York.

Rosenbrock, H. H. (1980). An automatic method for finding the greatest or least value of a function, Computer J. 3, pp. 175-184.
Saunders, M. A. (1976). A fast, stable implementation of the simplex method using Bartels-Golub updating, in Sparse Matrix Computations (J. R. Bunch and D. J. Rose, eds.), pp. 213-228, Academic Press, New York.
Wolfe, P. (1962). The reduced-gradient method, unpublished manuscript, RAND Corporation.
Wright, M. H. (1978). Numerical methods for nonlinearly constrained optimization, Ph. D. thesia, Stanford University.

## INDEX

A. in printed solution, 70, 72

Accuracy, for satisfying linear constraints, 26, 67
for satisfying nonlinear constrainte, 35
for solving linearized subproblems, 22
of computed functiona, 27, 66-67
of linesearch procedure, 28-29
AIJ TOLERANCE, 21
Alternative optimum, 70
Augmented Lagrangian, 3-4, 59
B, Basis matrix, 2-3, 34
BACKUP BASIS FILE, 21, 40-51
Bartels, R. H., ii, 2
Banic variables, 2
BASIS filea, 49-56
Basis map, 49-51, 68-69
Basis matrix, $\boldsymbol{B}, \mathbf{2 - 3 , 3}$, 4
Bounds, 1-5, 46-48
specification of default valuen, 29, 38
BMAX, in basis factorization statintica, 62
BOUNDS section of MPS Ble, 16-48
BOUNDS keyword, specifying name of bound eet, 21
-BS, in iteration log, 58
CALCFG, subroutine, $\mathbf{i}$
CALCON, subroutine,
CENTRAL DIFFERENCE INTERYAL, 24
CHECK FREQUENCY, 22
COEFFICIENTS, 22
Cold start, see CRASH procedure
Column ordering, implicit, 31, 44
Column variablea, 1, 71
COLUMNS section, of MPS Gile, 31, 43-44
of printed solution, 71-72
COLUMNS, estimate of number of variables, 22
Comment cards, in MPS Ale, 11-48
in SPECS ile, 17-18
COMMON blocke, 7, 15-16, 63, 78, 79
reserved, 78
Compatibility with MINOS 4.0, 51
COMPLETION optiona, 22
Componite objective technique, 40
COMPRSSNS, in bacis factorisation statiatica, 61
Conjugate-gradient method, $\mathbf{i}$
Constant Jacobian elementa, 12, 44
CONV, in iteration los, 60
Convergence, likelihood of, 4, 23-24, 30
rate of, 3, 27, 34
tolerances, see FEASIBILITY TOLERAMCE, OPTIMALITY TOLERANCE And ROW TOLERANCE;
also sce CYCLE TOLERANCE
CRASH procedure, for selecting initial becie, 22-23, 47
CRASH options, 22-23, 47
Cycle facilities (for sequences of problems), 8 , 13-15, 23, 56, 109-111
CYCLE options, see cycle facilitioe
Cycling (endem iteration), es

D, in printed solution, 70, 72
Damped Newton method, 23
DAMPIMG PARAMETER, 23-24
Dantzig, G. B., i, ii, 1
Data, input sequence, 7
Davidon, W. C., i, 2
DEBUG LEVEL, 24
Default values for SPECS file keyworda, 18-20
Degenerate variable, 70
DEMAND, in basis factorization statistica, 61
Dense Jacobian matrix, 44, 94, 96
DENSITY, in bacis factorization atatistica, 61
DERIVATIVE LEVEL, 9-12, 24-25
DIFFERENCE INTERVAL, 25
Difference approximation to derivatives, see missing gradients
DJ, in iteration log, 58, 60
Dual simplex method, i
Dual variables, 10, 32, 64, 70, 71, 72
DUMP file, 25, 53-54
ELEMENTS, estimate of nonseros in $A, 25$
ELEMS, in basis factorisation atatiatics, 61
ENERGENCY VERIFY LEVEL, see VERIFY options
End-or-File condition when reading SPECS file, 63, 81
ENDRUN mesaage, 63
Equality constraints, 42
Error checks, on computed gradienta, 38-87, 38-39, 66
on satisfying $A x+a=0,22,67$
Error messages, 34, 63-60
during input of MPS Gile, 26
ETAMACRO, test problem, 75, 84, 82
Example problems, 85-108
Exit conditions, 63-69
F, parameter of FUNOBJ, 10, 16
F(*), parameter of FUNCON, 11, 12
$F(x)$, see nonlinear objective function
$f(x)$, see nonlinear constraint functions
Factorization of basis matrix, 26, 29, 33, 58-59, 61-62
FACTORIZATION FREQUENCY, 26
FACTORIZE, in basis factorization atatiatica, 61
FEASIBILITY TOLERANCE, 26, 47, 64
Feasible points, definition, 3
evaluation of functions at, 3, 26, 47
Files, 6-7, 80, 82, 85
Formulation of problems, 5, 6
Fortran source files for MINOS, 75-81
Fortran 66 versus Fortran 77, 76
Free rows, 42
Free variables, 46
Full completion (accurate solution of subproblems), 22
FUNSCON, subroutine, 7, 8 consistency with MPS Ele, 44 exampley, 96,100
speciferation, 11-12
FUNCTION PRECISIOM, 27, 67
FUNOBJ, subroutine, 7,8 consistency with MPS Alh, 44
examplea, 90,989
opecification, $9-10$

G(*), parameter of FUNOBJ, 10
G(*), parameter of FUNCON, 11-12, 44
Gill, P. E., il, 2, 3
Global optimum, 8,64
Golub, G. H., ii, 2
GROWIH, in basis factorisation atatistica, 68

Header cards in MPS ine, 41 HESSIAN DITENEION, 27, 37
Hewian matrix, 3
HIOD, in iteration $\log , 59$
H-CDMDN, in iteration $\log , 60$
HS (*), state vector, 14, 50-51, 09

I, in printed solution, 71, 72
IMCREASE, in banis factorisation atatintica, 68
Inequality conatrainta, 42
INFEAS, in basis factorisation statintics, 61
Infeasibilities, 28, 40
Infeasible problems, 26, 64-85
Infeanible subproblema, 64-85
Infinite bounda, 46
Initial point, $x_{0}, 3,4,5,23,47-48$
INITIAL bounds set in MPS Ale, 47-48
linput to MINOS, 7
examples of, 85-105
INSERT Gile, 27, 52-53, 54
Inatalling MINOS, 75-81, 85
Integer programming,
Internal modifications to problem, soe cycle facilltion
Invert procedure, see factorisation of bacis matrix
Iteration log, 29, 57-60
example, 105-108
ITERATION8 LIMIT, 2 :
ITM, in iteration los, 57

Jacobian matrix, J( x ) delinition, 8
computetion of, 11-12
constant coefiliciente, 12, 44
position within conatraint metrix $A, 4,44$
printing, 4
aparaity patiera, 12, 44
JACOBLAI option (DEMEE or BPAREE), 28, 44

Keywords in SPEC8 Ale, 17
checkliat and default valuec, 18-80
definitiona, 21-40
Kreuer, J., I
$\ell_{1}$ problem, 93
$\lambda_{h}$, see Lagrange multipliers
L , in iteration log, 58
LAOS bacis-handling pacloge, it
Lagrange multipliers, $\lambda_{k}, i, 4,18,14,71$ printing, 34
initial eatimate, $\lambda_{0}, 4$
Lagrangian, 4
LAGRNNGIAN option (YES or MO), 4, 28
Least squares, linear, 92-93
LENL, in basis factorisation otatidtica, 61
LENU, in banis factorisation statistica, 61
LINEAR, in basis factorisation statiotics, $\mathbf{6 1}$
Linear approximation to nonlinear constrainten, see linearised conotrainte
Linear constrainte, 1-5, 15
Linear programming, 1, 9
example, 86-87
lest problem, see ETAMACRO
Linearized constrainta, 4, 70
Linearly constrained optimization, 2-8
examples, 00-93
Line search, 3
securacy of, 29
failure of, 66-67
LINESEARCH DEEUG, 28
LINESEARCH TOLERANCE, 20-29
Linesearch procedures, ii, 28-29, 38
Linking subroutines to MINOS, 64
LIST LIMIT, for printing MPS flle, 29
LMAX, in basis factorization statistics, 62
LOAD file, 29, 53-54
LOG FREQUENCY, 29, 34
Local optimum, 5, 64
Logical variables (alacks), 1
Lower bounds, see bound
LOWER BOUND (default lower bound on all variables), 29
$L U$ factorization of basis matrix, $i$-ii, 2, 3, 61, 68 see factorization of bavi: matrix
LU FACTOR TOLERANCE, 30, 62
LU UPDATE TOLERANCE, 30
LUSOL basis-handling package, 1 i-ii, 2
$m=m_{1}+m_{2}$ (number of nonlinear and linear conatrainte), 1, 6
$m_{1}$ (number of nonlinear conatrainte), $1,6,16$
$\boldsymbol{m}_{\mathbf{2}}$ (number of linear conatraints), $1,6,16$
Machine-dependent aubroutines, 75, 79-81
Machine precision, $\subset$, 18, 81
Main program, 79, 82
Major iteration, 4
MAJOR ITERATIONs limit, 30
Manne, A. S., ii, 83, 84
MANNE, teat problem, 75, 76, 80, 85, 80-108
Markowitz, ordering for sparse LU factorisation, ii, 2, 61
MATCOL, subroutine, 14, 23
apecification, 15
Mathematical programenins ayotema, $i, 34,41,48$, 52-54
MATMOD, subroutine, 7, 8, 23, 82
example, 109-111
apecifleation, 18-14
Matrix coeflicienth, isnoring amall valuen, 21, 108 number of, 25
Matrix data atructure, 15
Minor iteration, 2
MINOR ITERATION8 limit, 30
MINOS, acronym, il
MERIT, in banis factorization atatiotics, 61
MHW4D, example problem, 04-97
Misuing gradients, $1,9,24-25$
MODE, parameter of PUNOBJ and FUNCOM, 9-10, 11, 12, 86
MPS ille, 6, 7, 30, 41-48, 60
examples, 87, 89, 91, 97, 108-103
restrictions and extension, 48
MULTIPLE PRICE option, 31
Murray, W., i, 2, 3
Murtagh, B. A., 2,8
$n=n_{1}+n_{2}$ (number of nonlizear and linear variables, excluding alacke), 1, 6
$n_{1}=\max \left\{n_{1}^{\prime}, n_{1}^{\prime \prime}\right\}$ (number of nonlinear variablee, $s$ ), 1, 8, 16, 37
$\boldsymbol{n}_{1}^{\prime}$ (number of nonlinear objective variables), $\mathbf{3 1}$
$\boldsymbol{n}_{1}^{\prime \prime}$ (number of nonlinear Jacobian variablea), 31
$n_{2}$ (number of linear variables, y), 1
$N$, matrix asociated with nonbacic variablee, 2
H, in printed solution, 71, 72
NAME card in MPS fle, 41
NCON, in iteration log, 50
NCP, in iteration log, 59
NIMF, in iteration log, 50
NJAC, parameter of FUMCOM, 11, 12, 16
NEW BASIS 5ile, 21, 31, 49-51
NOBJ, in iteration log, 50
Noiay functions, i, 27, 86-87
Nonbasic variables, 2
Nonlinear conatraint functions, $f(s), 1,3-4,7,11-12$ printing, 34
Nonlinear constrainte, 1, 3-4, 5
Nonlinear equatione, 28-24
Nonlinear Jacobian variablea, 31, 44
Nonlinear objective function, $F(x), 1,2-3,7,2-10$
Nonlinear objective variablee, 31, 44
Nonlinear variablee, 1, 4, 44
printing, 34
MONLIMRAR, in bacie factorisation stathatica, 61
NONLIMEAR CONSTRAIMTS and VARIABLES, 31
Nonlinearly conatrained optimication, 8-4 examples, 94-103
MOPT, in iteration log , 88
MPROB, parameter of FUNOBJ, FUNCON and MATMOD, 10
MSB, in Iteration log, 80
NSTATE, parameter of FUNOBJ and FUKCOM, 10
MWCORE, parameter of FUNOBJ, FUNCON and MATICD, $10,40,74$

Objective function $\left(F(x)+c^{T} s+d^{T} Y_{2}{ }^{1}\right.$
Objective row in MPS file (defining $\mathrm{e}^{\mathrm{P}^{2}}+d^{\mathrm{T}} \mathrm{y}$ ), 49
OBJECTIVE, in basia factorization etatiotien, 61
OBJECTIVE, in iteration log, 50
OBJECTIVE keyword, apecifying name of linear objective, 32
OLD BASIS file, 21, 32, 40-51
Optimal solutions, local and global, $5,68-64$
OPTIMALITY TOLERANCE, 32, 64, 67, 71
Ordering of constraints and variables, 31, 42, 44
Output from MINOS, 57-74, see also LOG FREQUEMCY, PRIMT LEVEL, SUMMARY EREQUENCY
$P^{4}$ ordering for aparse $L U$ factorisation, 1
Parameters, ii, 7
Parametric algorithums, $\mathbf{i}$
Partial completion, 22
Partial pricing, 35, 57-58
Penalty parameter, $p, 4,33$, 35
PENALTY PARAMETER, 4, 38
PH (Phase), in iteration los, $57-58$
PHANTOM COLUMNS and ELEMEATS, 8, 15, 23
Piece-wise amooth functions, $i, 93$
PLLOT energy-economic model, it
PIVOT, in iteration log, 58
PIVOT TOLERAMCE, 4 , 58 , 67
PP, in iteration log, 57
Preckel, P. V., ii
PRICE operation, 57
Primal simplex method, eee nimplex method
PRINT file, 6-T, 36
PRINT LEVEL options, 34
Problem forme colved by MINOS, 1
Problem formulation, s-8
PROBLEM MUMBER, $10,13,3$
PUNCH file, 35, 52, 84
Quadratic programming, it
example, $80-81$
Quari-Newton method, $i, 2,3,6,27,59-00$
$\boldsymbol{R}$, triangular matrix for approximation to reduced Hessian, 3, 6, 27, 50-60
RADIUS OF CONVERGENCE, 35
Ranges on general constrainte, 1, 45-46
RNNGES section of MPS Bile, 46-48
RANGES keyword, apecifying name of range eot, 35
Rangins procedures, $;$
READ file, E-7
Record length of Blea, 6-7
Reduced gradient (vector), 3, 32, 37, 58,72
Reduced-gradient algorithm, 2, 4, 58-60
Reduced Flessian (matrix), 13, 50-60
Reid, J. K., i, U, 2
Reatarting previous puns, 49, 55-58, 71
Reatriction, in MPS format, 43
on problem characteriatica, 5-6
Rewinding filen, 7
RG, in iteration los, 50
RHS section of MPS Ble, 45
RH8S keyword, specifying aame of right-hand side, 85

Right-hand side, 1, 45
Robinson, S. M., i, 8
Rocen, J. B., $\mathbf{I}$
ROW CHECK, meange in PRINT Sle, see CHECK FRECUEMCY, 22
ROW TOLERANCE, 22, 25, 35
ROWS section, of MPS file, 42-48
of printed solution, 70-71
ROWS, estimate of number of general comatrainta, 35
s, vector of alack variablea, ace slack variables
s, number of superbanie varisblea, 2,5
$S$, matrix asociated with superbasic variablea, 2
Saundera, M. A., i, 2, 3
SAVE FREQUEMCY, 21, 34
Saving basis filea, 21, 36, 65
+SBS, -SBS, in iteration log, 5
SCALE optiona, 36
Scaling of dats and varisbles, 5, 35-86
SCRATCH ile, 6-7
Search direction, 3
Senaitivity analynie, 1
Separable functions, 5
Sequence of probleme, 7, 2, 12-15
Simplex method, 1-2, 57
SINF, in iteration log, 50
Singular banin, 62, 63
Singularities in nonlinear functiona, $5,23,53$
Slack variablen, 1, 15, 36, 70-71
SLACKS, in bacie factorisation ctatiatice, 61
Smooth functions, 1,5
Solution output, 70-72
example, 108
SOLUTION 11 e, , 6-7, 33, 72
SOLUTION optione, $38-37$
Source files (MINOS Fortran code), 76-01
Sparse Jacobian matrix, 4, 44
Sparse constraint matrix, 4 , 15
SPECS file, 6-8, 17-40
checklist and default values, $10-20$
examples, 86, 88, 91, 92, 97, 101
format, 17-18
keyworde, 21-40
Spikes, i
Standard form for probleme, 1
START and STOP gradient verification, 87
State vector, H8 (*), 14, 50-81, 60
STEP, in iteration lot, 88
SLorage allocation and/or requirementa, see workepaee
Slructural variables, 1
Subproblem, deraition, 4
Subroutine hierarchy, 8
Subroutine namme, reverved, 77-78
Subroutinee, requiped from mer, 7,00
SUBSPACE TOLRAMCR, 87, $\omega 0$
SUMMARY file, 6-7, 28, 72-74
sUMMARY FRECUENCT, 20
Superbasic variables, 2, 8, 12, 28
SUPERBASICB LIMIT, 27, 2
Suppreseion of outpet, 34, 8

SUPPRESS PARNMETER8 option, 38
Syatem information, 6-8, 15-16, 63, 75-92
Tent problema, 75, 76, 83-84, 85-92, 84-108
TOO MANY ITERATIOMS, exit condition, es
Tranaformation of varisbles, 5
$\mathbf{U}$, in iteration $\mathbf{l o g}, 58$
UMXX, in bacis factoriaation atatiatice, 08
UMIM, in bacis factorisation statiotice, 68
Unbounded problems, 38, 65
Unconatrained optimisation, exampla, 80-80
Upper bounds, see boundr
UPPER BOUNID (default upper bound on all variablem), 38

## VERIFY optione for chocking Eradienta, 80

Werm etart, 49-58
WATFIV compiler, ith, 78
WEAPON, teat problem, 75, $8:$
WEIGHT ON LIMEAR OBJECTIVE, 4
Wolfe, P., $i, 2$
Workspace (storage requirements), $5-8,10,40,50$, 68, $69,74,80$
TORKSPACE paramoters in SPEC8 Sile, 40, 48
Wright, M. H., it, 2, 3, 94
Wylbur text editor, ${ }^{\text {in }}$
x, monlinear variabloe, 1, 4
$x_{0}$, see initial point
sht 4
printing, 34
y, Inear variablen, 1
2, null-space operator, a
Z, workopace erray, see workepece

UNCLASSIFIED


| REPONT Documentation Pace | bepan dintivetions |
| :---: | :---: |
|  |  |
|  <br> MINOS 5.0 USER'S CUIDE | a. Trat or acpont a penco covered Technical Report |
|  | C. Penfommine ove. Remont mumeen |
| 7. Authotial <br> Bruce A. MARTAGH and Michael A. SANHDERS |  $\begin{aligned} & \text { NOOO14-75-C-0267 } \\ & \text { DAAG29-81-K-0156 } \end{aligned}$ |
|  <br> Department of Operations Research - SOL <br> Stanford University <br> Stanford, CA 94305 | NR-047-143 |
| 11. Confrollime office mame amo noomese Office of Naval Research - Dept. of the Navy | 12. nifont oare December 1983 |
| 800 N. Quincy Street <br> Arlington, VA 22217 | 12. Mumen of PAOES |
|  |  |
| Research Triangle Park, NC 27709 | The per faigmicatiow/Downerading |

## 

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


## 18. SUPPLEMEMTARY MOTES


AUCMENTED LACRANGIAN
FORTRAN CODE
LARGE-SCALE OPTIMIZATION
linear constraints
LINEAR PROCRAMMING

NONLINEAR CONSTRAINTS NONLINEAR PROGRAMMING OPT IMIZATION PROJECTED LAGRANGIAN METHOD QUASI-NEWTON METHOD

REDUCED-GRADIENT METHOD SIMPLEX METHOD SPARSE MATRIX


SEE ATTACHED

SOL 83-20: MINOS 5.0 USER'S CUIDE, by Bruce A. Murtagh and Michael A. Saunders

MINOS is a large-scale optimization system, for the solution of sparse linear and nonlinear programs. The objective Punction and constraints may be linear or nonlinear, or a mixture of both. The nonlinear functions must be smooth.

Stable numerical methods are employed throughout. Features include a new basis package (for maintaining sparse LU factors of the basis matrix), automatic scaling of linear constraints, and automatic estimation of some or all gradients. Upper and lower bounds on the variables are handled efficiently. File formats for constraint and basis data are compatible with the industry MPS format.

The source code is sultable for machines with a Fortran 66 or 77 compiler and at least 500K bytes of storage.


[^0]:    ${ }^{*}$ D. E. Knuth, TEX and METAFONT, New Directions in Typesetling, American Mathematical Society and Digital Prem, Bedford, Maesachusetho (1979).

