INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI
INTEROPT
A Software System for
Interactive Function Minimization

by
Bahaa Guirguis

Submitted to the School of Graduate Studies
in partial fulfillment for the requirements of
the degree of Master of Applied Science

Department of Electrical Engineering
Faculty of Science and Engineering
University of Ottawa
Ottawa, Ontario

December 1976

© B. Guirguis, Ottawa, Canada, 1977
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.
ACKNOWLEDGMENT

I wish to express my deep gratitude and sincere appreciation to Dr. L.G. Birta, as my supervisor, for his valuable guidance, encouragement and the many useful suggestions throughout the course of the preparation of this thesis.

I would like to extend my thanks to Dr. J. Raymond and Mr. J. Gavrel for their valuable assistance in using the graphical package.

The secretarial assistance by Mrs. C. Henderson in typing and organizing this document is especially appreciated.

Finally, the financial support of the National Research Council of Canada through grant A8109 is gratefully acknowledged.
# Table of Contents

**ABSTRACT**

**CHAPTER 1** INTRODUCTION

1.1 Preliminary Remarks 1
1.2 The Optimization Problem 1
1.3 Software Requirements 2
1.4 Features Provided by INTEROPT 4
1.5 General Characteristics of the Methods Provided in INTEROPT 5
1.6 An Overview of INTEROPT 6

**CHAPTER 2** INTEROPT FEATURES AND OPERATING PROCEDURES 8

2.1 General Features and Notation 8
2.2 Run Control 9
2.3 Priming Guess Specification 11
2.4 Run Conditions
   2.4.1 Default Values 16
   2.4.2 Changing the Run Conditions 17
2.5 Initialization and Stopping Parameters 18
   2.5.1 Default Values 20
   2.5.2 Changing the Initialization and Stopping Parameters 20
2.6 Summary Table Display 22
2.7 Function Plots
   2.7.1 The Cursor Mode 25
2.8 Status Information 25
2.9 Execution of the Minimization Procedure 27
2.10 Output Documentation 29
2.11 Printer Output From Each Iteration 31
2.12 The Linear Search Display 33
2.13 The User-Provided Subprograms 36
2.14 Constrained Minimization 39
2.15 Diagnostic Messages 41
2.16 Line Drop 45
2.17 Synopsis of the Operating Procedures 45

CHAPTER 3 INTEROPT ORGANIZATION AND SUBROUTINES 58

3.1 General Structure and Flow-Charts 58
   3.1.1 Interactive Phase 58
   3.1.2 Minimization Phase 60
   3.1.3 The Single Iteration Execution Phase 61
   3.1.4 The Linear Search Phase 62

3.2 Subroutines Associated with Four INTEROPT Phases 63
   3.2.1 GETN 64
   3.2.2 ARGDIS 64
   3.2.3 ARGMT 64
   3.2.4 NEWRUN 65
   3.2.5 CONDIS 65
   3.2.6 EXAM 66
   3.2.7 CONCHN 66
   3.2.8 INTDIS 66
   3.2.9 INTCHN 66
   3.2.10 OUTCHN 67
   3.2.11 DISITR 67
3.2.12 CDIS
3.2.13 TABDIS
3.2.14 DISTAB
3.2 15 TRANS
3.2.16 PREP
3.2.17 PLTMSQ
3.2.18 PLOT
3.2.19 SRCH
3.2.20 PLT
3.2.21 PLTSUB
3.2.22 FUNPLT
3.2.23 CHANGE
3.2.24 TABLE
3.2.25 TSTORE
3.2.26 RUN
3.2.27 HEADER
3.2.28 UPDATE
3.2.29 ST$SUB
3.2.30 ACCESS
3.2.31 MESSAGE
3.2.32 CONJ, DFP, POWELL and ZANG
3.2.33 I$IOU
3.2.34 QADMIN
3.2.35 QUADFT, FBNACI & GOLDEN
3.2.36 DRAW
3.2.37 LS$DIS
CHAPTER 4  EXAMPLES OF INTEROPT APPLICATIONS 84

4.1 Introduction 84

4.2 Examples to Illustrate the Operating Procedures of INTEROPT 84

4.3 Comparative Study 93

4.4 Interactive Linear Search 96

4.5 An Example of Constrained Minimization 101

CHAPTER 5  CONCLUSIONS 139

APPENDIX A1  THE MULTIDIMENSIONAL MINIMIZATION TECHNIQUES IN INTEROPT 144

APPENDIX A2  THE LINEAR SEARCH ALGORITHMS 148

APPENDIX A3  TEST PROBLEMS 151

REFERENCES 153
The process of locating the minimum of a nonlinear function is highly experimental in nature since the determination of an acceptable solution may require many trials involving technique and parameter changes. If this problem is handled in a batch computing environment, the user is obliged to accept significant delays between minimization experiments. This difficulty can be overcome through the use of an interactive approach.

The work described in this thesis is concerned with the development of a software package, called INTEROPT, which applies interactive/graphics techniques to the solution of nonlinear minimization problems. INTEROPT is designed for use on the IBM 360/65 computer operated by the Computing Center of the University of Ottawa. A Tektronix 4013 (or equivalent) graphics display unit is utilized as an on-line input-output device.

The thesis describes the operation and structure of INTEROPT and gives examples of typical terminal sessions.
CHAPTER 1
INTRODUCTION

1.1 Preliminary Remarks

The work described in this thesis is concerned with the development of a software package, called INTEROPT, which applies interactive/graphics techniques to the solution of non-linear minimization problems. INTEROPT is an extension of an existing package, OPTPAK [1] which is wholly batch processing oriented. INTEROPT is designed for use on the IBM 360/65 computer operated by the Computing Center of the University of Ottawa. A Tektronix 4013 (or equivalent) graphics display unit is utilized as an on-line input-output device. A collection of S/360 assembly language routines is used to control the communication between INTEROPT and the Tektronix unit.

1.2 The Optimization Problem

The general optimization problem can be stated as follows [2]:

Find the argument \( x^* \) of the scalar function \( f(x) \), \( (x \) is an n-vector), which yields the minimum value for \( f \) and which satisfies the following conditions (the constraints):

\[
g_j(x^*) \geq 0 \quad j = 1, 2, \ldots, m_1
\]

\[
h_j(x^*) = 0 \quad j = m_1+1, m_1+2, \ldots, m
\]

The function \( f(x) \) is called the criterion or objective function.

Throughout the presentation, the minimization of a criterion function is considered since the maximization of \( f(x) \) is equivalent to the minimization of \( [-f(x)] \).
INTEROPT is primarily oriented toward the solution of the unconstrained problem; i.e. the case where the auxiliary conditions $g_j \geq 0$, $h_j = 0$ are absent. It is well known, however, that the more general problem with constraints can be handled as a sequence of unconstrained problems through the use of penalty function methods [3,4,5]. INTEROPT, in fact, relies on the interior penalty function approach [3] as the mechanism for handling the general problem.

In any specific application, the function being minimized is assumed to have continuity properties consistent with the requirements of the algorithmic technique (or techniques) that are applied by INTEROPT. Furthermore, the function is assumed to have at least one local minimum but convergence to the global minimum cannot be guaranteed.

The optimization problem arises in a variety of contexts [2], usually in a natural manner, e.g. in engineering design [6] and in minimization of cost functions or maximization of profit functions. In some circumstances, however, it appears via a circuitous route, e.g. in the context of function approximation or as a reformulation of a two-point boundary value problem.

1.3 Software Requirements

The numerical solution of the unconstrained minimization problem has been extensively studied in recent years and a variety of good algorithms (or minimization techniques) have appeared in the literature [7]. It is generally conceded, however, that no single one of these algorithms qualifies as being best under all possible circum-
stances, i.e. "shapes" for the criterion function. Experiments with
groups of algorithms have demonstrated that a given algorithm may,
with respect to a particular function, perform extremely poorly (or
even fail altogether) while for some other function it may out-per-
form the other methods being considered.

The implementation of any particular algorithm is usually asso-
ciated with peripheral computational issues that must be resolved
before the algorithm can be used. These matters may be relatively
straight-forward, such as the specification of the values to be
assigned to parameters in the algorithm. On the other hand, they
may themselves represent significant computational issues; e.g. the
formulation of a mechanism for solving a one-dimensional search
problem that may be embedded in the algorithm. Such secondary issues
can have a substantial influence on the success that can be attained
with a particular algorithm applied to a particular problem [9,10].

The above suggests that total reliance on a single algorithm
with pre-selected "values" for secondary parameters*, for all mini-
mization tasks is unwise. A degree of flexibility is essential and
such flexibility can be achieved through the use of a software
package which places at the user's disposal several different mini-
mization algorithms as well as flexible choice for any secondary
parameter. Several batch oriented software packages have been
developed based on this philosophy (e.g. [1], [8]).

* The notion of secondary parameter is used here in a very broad
sense. It includes, for example, the one-dimensional search question
and questions relating to the generation of gradient information.
The range of alternatives provided by such a package, in turn, gives rise to a specification problem inasmuch as the user is obliged to make selections based principally on his insight and experience. For example, he must choose the technique he feels is best for the problem at hand, as well as the values for the secondary parameters. In addition, a choice of the "priming guess" (the initial estimate $x^0$ of the minimizing argument) must be provided and it is well-known that the speed with which any algorithm moves toward the solution depends greatly on the "quality" of this guess. An improper selection of any of these alternatives can result in little or no movement toward the problem solution.

Once convergence to the solution appears to have occurred some precautions still have to be taken in order to try and ensure that a global, rather than a local, minimum has been uncovered. This usually involves experiments with various different priming guesses to verify that convergence to a better point is not possible.

Thus the function minimization process is highly experimental in nature. If it is undertaken in a batch processing computing environment, the time between problem initiation and problem solution can be excessive because of the unavoidable delays that take place between job submissions. This elapsed time can, however, be greatly reduced in an interactive computing environment.

1.4 Features Provided by INTEROPT

An interactive software system which uses a graphical display device increases the computer's potential for serving as a tool in
solving the minimization problem. In an interactive environment, the user is given a very high degree of flexibility to select, and even to change within the particular run, the minimization technique together with the available secondary parameters. This is achieved in INTEROPT via a dialog that is carried out with the user at various stages of the run. The immediate availability of results from the current run or from previous runs helps the user to make appropriate decisions based on the feed-back from these results.

INTEROPT also provides optional displays of various plots that can be valuable to the user in evaluating the progress of the current run and in deciding whether or not to abandon the run because convergence appears unlikely.

The minimization methods employed in INTEROPT require the solution of one or more linear search (one-dimensional minimization) problems on each iteration. INTEROPT provides the user with an optional capability to interact with these linear searches and thereby better control the solution procedure.

1.5 General Characteristics of the Methods Provided in INTEROPT

As noted earlier, many effective function minimization algorithms have appeared in the literature in recent years. These algorithms can be grouped into various categories using any of several possible criteria. INTEROPT provides the user with eight different methods which can be divided into two classes referred to as gradient dependent and gradient independent methods.

A typical iteration with any one of the methods in INTEROPT
can be summarized as follows:

1. The minimization technique selects a "search direction" $s^k$ (an $n$-vector) according to some criterion characteristic of the technique.

2. It then invokes a "linear search algorithm" to perform a one-dimensional minimization of the criterion function along this direction beginning at the current argument $x^k$. This results in an optimal step $a^*$ along the direction $s^k$.

3. Depending on the specific technique steps 1 and 2 may be performed once or several times in the course of the iteration. In either case the result is a new argument $x^{k+1}$ at which the criterion function has a smaller value.

4. A check is made to determine whether one of several prespecified termination conditions is satisfied. If so, the process is stopped and the final argument is accepted as an estimate of $x^*$. Otherwise, another iteration is started.

1.6 An Overview of INTEROPT

As noted earlier, INTEROPT makes available to the user eight different minimization algorithms. Five of these algorithms are gradient dependent (require evaluation of the function gradient) and the other three are gradient independent. These algorithms are outlined in appendix A1. In addition, the user is also provided with a convenient mechanism to incorporate any gradient dependent or gradient independent algorithm of his own choosing.

In cases where the evaluation of the function gradient is
required, INTEROPT provides two methods of calculation. The first one utilizes a subroutine provided by the user which contains a specification of the gradient of the function. The second method utilizes an approximation formula which has a control parameter that is set by the user.

With respect to the one-dimensional minimization problem, INTEROPT provides three linear searches algorithms (appendix A2). Any one of these can be coupled with any of the minimization algorithms. In this case also, INTEROPT allows the user to incorporate his own linear search method.

There is a parameter associated with the linear search to indicate the "fineness" with which the linear search is performed. This parameter is under the control of the user.

Five parameters are used in INTEROPT to control the execution (stopping and initializing) of the minimization algorithm besides the control provided by the interactive system via the dialog with the user.

In addition to the output given on the screen in the course of the run, INTEROPT produces some documentation for the user, on the line-printer. Three options are available for the user to control the amount of information outputted on the line-printer.
CHAPTER 2
INTEROPT FEATURES AND OPERATING PROCEDURES

2.1 General Features and Notation

INTEROPT is designed to handle optimization problems of dimension eight or less (i.e. n≤8). Up to 25 minimization trials (or "runs") can be executed per job submission. Each such run is initiated from a "priming guess" which represents the user's estimate of the minimizing argument x*.

The computing session begins with the submission of a deck of cards that serves to initiate the INTEROPT package. The submitted program also contains the specification of the criterion function to be minimized and optionally the specification of its gradient.

The user interacts with INTEROPT via a group of commands which are entered at the Tektronix keyboard. The characters "line feed" (LF), "carriage return" (CR) and "blank" are ignored from any alphanumeric character string entered. The character CNT/Q (i.e. the Q key with CNT simultaneously pressed) is used to terminate all user commands.

In describing the commands, the composite characters CNT/Q and CNT/T (the T key with CNT simultaneously pressed) are represented by Q and T respectively for simplicity. Square brackets, [], are used to identify the options within the commands that may be omitted.

In all cases, after INTEROPT plots a curve, the Tektronix is placed in the "cursor mode" in which the cross-hairs of the cursor appear on the CRT screen. One character only may then be entered
at the keyboard. In this case the position of the cursor and/or the entered character serves as the user command.

Whenever it is required to enter a non-integer constant any of the usual FORTRAN format I, F, E or D may be used. INTEROPT ultimately changes the entered value to the double precision mode.

2.2 Run Control

This is the main phase of the interactive procedure during which the user is able to:

(a) check and/or alter the operating conditions for the current run,
(b) request information to be displayed,
(c) continue the current run, start a new one or terminate the whole INTEROPT job.

INTEROPT is in this phase (which is called the "Control Mode" in the sequel) whenever the following message flashes on the screen:

WHAT NOW?

This occurs at the beginning of each run and at certain points during the course of the run.

In response to this message, a command is entered by the user from the available set to specify his requirements. The commands are briefly outlined in the following table but are described in detail in the sequel.
<table>
<thead>
<tr>
<th>Specification Commands</th>
<th>Command</th>
<th>Function</th>
<th>Sect.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CONDITION</td>
<td>Displays the run conditions for the current run, then puts the system in</td>
<td>2.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a mode which permits these conditions to be altered if the user so wishes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INITIAL</td>
<td>Displays the initialization and stopping parameters for the current run,</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>then puts the system in a mode which permits these parameters to be</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>altered if the user so wishes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OUTPUT[,ω]</td>
<td>Enables the user to specify an option relating to the data to be</td>
<td>2.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>outputed from each iteration on the line printer.</td>
<td></td>
</tr>
<tr>
<td>Display Commands</td>
<td>TABLE[/rho][,x]</td>
<td>Displays a table that summarizes the results of previous iterations for</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the current run or a similar table for an earlier run.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TABLE, SUB</td>
<td>Displays a table that summarizes the most recent 30 iterations for the</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>current run.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PLOT[/ITER][,r_1,r_2,...,r_k]</td>
<td>Displays a plot of function value against either number of function</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>evaluations or number of iterations and/or a similar plot for some earlier</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>run(s).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PLOT[/ITER], SUB</td>
<td>Displays a plot of function value against either number of function</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>evaluations or number of iterations using the results of the most recent</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>30 iterations for the current run.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITERATION</td>
<td>Displays status information about the current iteration.</td>
<td>2.8</td>
</tr>
<tr>
<td>Decision Commands</td>
<td>GO</td>
<td>Starts or continues the execution of the minimization procedure.</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>RESTART</td>
<td>Restarts (i.e. initializes) the minimization algorithm.</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>NEW RUN</td>
<td>Starts an entirely new run with a new priming guess specification.</td>
<td>2.3</td>
</tr>
<tr>
<td></td>
<td>END JOB</td>
<td>Terminates the INTEROPT job completely.</td>
<td></td>
</tr>
</tbody>
</table>

Summary of User Commands
2.3 Priming Guess Specification

Each run in INTEROPT starts with specification of the priming guess, i.e., the initial estimate of the minimizing argument of the criterion function. In this respect, the user has the following options:

(a) Initiate the run from a new estimate, i.e., the user explicitly specifies the value of the priming guess (NEW).

(b) Initiate the run from the last explicitly entered priming guess (REPEAT).

(c) Initiate the run from the final result obtained in the preceding run (CONTINUE).

Note that for the first run, only option (a) is meaningful, therefore INTEROPT starts with the message:

WHAT IS YOUR PRIMING GUESS?

as described later in this section.

A new run is started when INTEROPT is in the control mode and the user enters the command:

NEW RUN Q

Then, INTEROPT responds by displaying the following:

CURRENT PRIMING GUESS IS

\[
\begin{align*}
&x_1^0 \\
&x_2^0 \\
&\vdots \\
&x_n^0 \\
\end{align*}
\]

FUNC VALUE = f(x_1^0)
CURRENT ARGUMENT IS

\[ x_1 \]
\[ x_2 \]
\[ \vdots \]
\[ x_n \]

FUNC VALUE = \( f(x) \)

REPEAT, CONTINUE OR NEW?

where \( (x_1^0, x_2^0, \ldots, x_n^0) \) is the last explicitly entered priming guess,

\( f(x^0) \) is the corresponding function value,

\( (x_1, x_2, \ldots, x_n) \) is the final estimate of the minimizing argument obtained in the preceding run,

and \( f(x) \) is the corresponding function value.

The user now enters either NEW, REPEAT or CONTINUE. In case of the latter two responses, INTEROPT proceeds to the next phase; i.e., the specification of the run conditions (section 2.4). If the user enters NEW (or in case of the first run), the following steps take place:

(a) The following prompting message is displayed:

WHAT IS YOUR PRIMING GUESS?

(b) The user enters the components of the priming guess vector as a string of \( n \) constants separated by commas, e.g.:

\[ x_1^0, x_2^0, \ldots, x_n^0 \]

where \( x_i^0 \) is the \( i^{th} \) component of the guess vector and \( n \leq 8 \).

(c) If all the \( x_i^0 \)'s are entered correctly (according to the previous specifications) and their number \( n \) agrees with the actual dimension of the argument of the submitted criterion function
INTEROPT proceeds to the next step (d); otherwise, an error message is displayed and the user re-enters the priming guess.

(d) The following is displayed as a check:

```
CURRENT PRIMING GUESS IS

0
x1
0
x2
.
.
.
0
xn
```

FUNC VALUE = f(x)

DO YOU WISH TO RESPECIFY THE PRIMING GUESS?

(e) The user may reply:

YES  

and the process is repeated starting from step (b), or

NO  

and the program proceeds to its next phase (the specification of the run conditions, section 2.4).

2.4 Run Conditions

The run conditions are a set of options specified by the user which govern various aspects of the solution procedure to be applied during the run. Two variables, MINTEC and LN$SRH$, and two parameters associated with them, G$PAR$ and LS$PAR$ respectively, are used to describe the options. The options and their admissible values are described below.

(a) The Minimization Technique (MINTEC)

INTEROPT makes available to the user two groups of multidimen-
sional minimization algorithms which are characterized as gradient dependent and gradient independent.

(A) Gradient dependent algorithms

(i) An optimal-step steepest descent algorithm (GRAD) [Appendix A1.1]

(ii) The conjugate gradient algorithm proposed by Fletcher and Reeves (F-R) [Appendix A1.2]

(iii) A variation on the conjugate gradient method suggested by Polack and Ribiere (P-R) [Appendix A1.3]

(iv) Another variation on the conjugate gradient method suggested by Sorenson (SOREN) [Appendix A1.4]

(v) The Fletcher-Powell formulation on the Davidon algorithm (D-F-P) [Appendix A1.5]

An arbitrary gradient dependent technique (YA) can be added to this group by submitting a user-provided subroutine* called YOURSA.

(B) Gradient independent algorithms

(i) The method proposed by Powell (POWELL) [Appendix A1.6]

(ii) A variation on Powell's method suggested by Zangwill (ZANG) [Appendix A1.7]

(iii) A basic gradient independent method based on the classical relaxation approach, also called the extended sequential search (ESQ) [Appendix A1.8].

It is also possible to add an arbitrary gradient independent technique (YB) by submitting a user-provided subroutine called YOURSB.

* All the user-provided subprograms are discussed later.
The mnemonics of the minimization techniques constitute the admissible values for MINTEC. They are listed below.

\begin{align*}
\text{GRAD} & \quad \text{F-R} \quad \text{P-R} \quad \text{SOREN} \quad \text{D-F-P} \quad \text{YA} \\
\text{POWELL} & \quad \text{ZANG} \quad \text{ESQ} \quad \text{YB}
\end{align*}

(b) The Gradient Parameter (G$\text{SPAR}$)

This parameter is relevant only in cases where a gradient dependent algorithm is being used. INTEROPT accommodates two approaches for generating the required gradient information:

(i) An analytical gradient method (AG) in which a user-provided subroutine, called GRAD, is used to calculate the gradient of the criterion function.

(ii) A finite difference approximation method which utilizes the following relation to calculate the gradient of the criterion function at the point $x^k$:

\[
\frac{\partial f}{\partial x_i} \bigg|_{x^k} \approx \frac{f(x^k + \Delta e_i) - f(x^k)}{\Delta}
\]

where $e_i$ is the $i^{th}$ column of the $n \times n$ identity matrix, and $\Delta$ is the perturbation size for the gradient approximation computed from the relation:

\[
\Delta = 10^{-a_1}, \quad a_1 \text{ is a positive integer.}
\]

The parameter G$\text{SPAR}$ may be assigned either the value AG or a positive integer value ($a_1$ in the above relation) where the former case is interpreted as a request for the use of analytic gradient information.

(c) The Linear Search Algorithm (LN$\text{SSRH}$)

INTEROPT provides three algorithms for handling the linear search problem, namely:
(i) A quadratic polynomial fitting technique (QUADFT)  
[Appendix A2.1]

(ii) A technique based on the Fibonacci Search (FBNACI)  
[Appendix A2.2]

(iii) An implementation of the Golden Section Method (GOLDEN)  
[Appendix A2.3]

The user may also add an arbitrary linear search algorithm (YL) by submitting a subroutine called YOURSL.

The mnemonics of the linear searches constitute the admissible values for IN$SRH. They are listed below:

QUADFT  FBNACI  GOLDEN  YL

(d) The Linear Search Parameter (LS$PAR)

This parameter specifies the fineness with which the linear searches are carried out. This fineness increases with increasing values of LS$PAR. The value assigned to LS$PAR should be a positive integer $a_2$.

2.4.1 Default Values

The run conditions are initially assigned default values which are used until they are changed by the user. The default values are as follows:

MINTEC = D-F-P

\[
G$PAR = \begin{cases} 
AG & \text{if subroutine GRAD is submitted} \\
9 & \text{otherwise} 
\end{cases}
\]

IN$SRH = QUAD

LS$PAR = 2

It should be noted that the run conditions are not reset to
their default values automatically at the beginning of each run.

2.4.2 Changing the Run Conditions

Whenever INTEROPT is in the control mode, the user can enter the following command.

\[ \text{CONDITION } \bar{Q} \]

INTEROPT responds in the following way:

(a) The current values of the run conditions (the default values in case of the first display) are displayed as shown below:

\[
\begin{align*}
\text{RUN CONDITIONS} \\
\text{MINTEC} &= m_1/G\text{PAR} = a_1 \\
\text{LS\$SRH} &= m_2/LS\text{PAR} = a_2 \\
\text{CHANGE}
\end{align*}
\]

where \( m_1 \) and \( m_2 \) are the mnemonics of the minimization technique and the linear search respectively, and \( a_1 \) and \( a_2 \) are the values of the gradient parameter and the linear search parameter respectively.

(b) INTEROPT waits until the user enters the required changes according to the following format:

\[
\text{m}_1'/a_1'; \text{m}_2'/a_2'; \bar{Q}
\]

where \( \text{m}_1' \), \( \text{m}_2' \), \( a_1' \) and \( a_2' \) are now the new values to be assigned to the run conditions. To maintain the current value of any one of the run conditions, the corresponding \( m_j \) or \( a_j \) may be omitted. If all existing conditions are acceptable, the only response necessary is \( \bar{Q} \).

(c) If the user's response does not have an acceptable syntax, an appropriate error message is displayed and the system again
awaits a proper response. If the response syntax is acceptable, the system returns to the control mode, but prior to doing so, the new run conditions, if specified by the user, are displayed.

2.5 Initialization and Stopping Parameters

With the exception of the GRAD and ESQ methods, the algorithms in INTEROPT embody a memory attribute which causes the generated search directions to be conditioned by the results of previous iterations. The stored information may however deteriorate in quality and this may lead to slow progress or even to a complete inability to make any progress. An "initialization" may be useful in such a case. By initialization it is meant that the accumulated information in the algorithm is discarded and the algorithm is restarted.

On the other hand, it may be desired to specify conditions which ensure that the algorithm stops at some point. Such conditions do exist in INTEROPT and they are specified via certain parameters that can be set by the user.

Five parameters are available in INTEROPT to control initialization and stopping. These are described below.

(a) The Automatic Restart Parameter (RESTRT)

This parameter enables the user to periodically initialize the minimization algorithm. When RESTRT is assigned the positive integer \( p_1 \), the algorithm being used will execute \( p_1 \) normal iterations, then will be automatically initialized on the \( (p_1+1)^{th} \) iteration and the cycle is repeated. This feature is disabled
if RESTRT is set to zero.

(b) **The Failure Parameter (FAIL)**

If the algorithm being used fails to make progress on some iteration, INTEROPT will either terminate the algorithm or initialize it, depending on the specification of FAIL (STOP or INIT respectively).

(c) **Maximum Number of Iterations (MXITER)**

MXITER is a positive integer that specifies the number of iterations permitted before the selected algorithm is stopped. When MXITER is set to zero, the INTEROPT system proceeds without any stopping condition based on the number of iterations.

(d) **Maximum Number of Function Evaluations (MXFE)**

This is a positive integer that specifies a limit on the number of function evaluations (i.e. evaluations of the criterion function). The minimization algorithm is stopped if, after any iteration, the number of function evaluations to date is greater than or equal to this limit. When MXFE is set to zero, the INTEROPT system proceeds without any stopping condition based on the number of function evaluations.

(e) **The Exit Parameter (ESPAR)**

This is a positive integer less than 14, used to determine the "natural" termination of the minimization algorithm. A natural termination, in the case of a gradient dependent algorithm, occurs when the gradient norm is less than $\epsilon$, where $\epsilon = 0.01 \times 10^{-p_5}$ ($p_5$ is the value of ESPAR); i.e. when

$$\left[ \sum_{i=1}^{n} \left( \frac{\partial f}{\partial x_i} \right)^2 \right]^{1/2} < \epsilon$$

In case of a gradient independent algorithm, it occurs when the
relative change in the criterion function value in two successive iterations is less than \( \epsilon' \), where \( \epsilon' = 10^{-P_S} \); i.e. when

\[
\frac{f(x^k) - f(x^{k+1})}{|f(x^k)|} < \epsilon'
\]

or

\[
f(x^k) - f(x^{k+1}) < \epsilon' \times |f(x^k)|
\]

2.5.1 Default Values

The initialization and stopping parameters are initially assigned default values which are used until they are changed by the user. These are as follows:

- `RESTRT = NOT APPL` (not applicable, i.e. no automatic restart)
- `FAIL = \{
  \begin{align*}
  \text{STOP} & \quad \text{in case of GRAD and ESQ} \\
  \text{INIT} & \quad \text{otherwise} \\
  \end{align*}
\}`
- `MXITER = NO LIMIT`
- `MXPF = NO LIMIT`
- `ESPAR = 14`

Note that these parameters are all automatically reset to their default values at the beginning of each run.

2.5.2 Changing the Initialization and Stopping Parameters

Whenever `INTEROPT` is in the control mode, the user can enter the command:

```
INITIAL Q.
```

The program then proceeds as follows:

(a) The current values of the initialization and stopping parameters are displayed with the following format:
INITIALIZATION & STOPPING PARAMETERS

1 RESTRT = \( p_1 \)
2 FAIL = \( p_2 \)
3 MXITER = \( p_3 \)
4 MXFE = \( p_4 \)
5 ESPAR = \( p_5 \)

CHANGE?

where \( p_j \)'s are the values of the parameters such that:

\( p_1 \) is a positive integer or "NOT APPL"

\( p_2 \) is either "INIT" or "STOP"

\( p_3 \) and \( p_4 \) are positive integers or "NO LIMIT"

\( p_5 \) is a positive integer (i.e. \( 1 \leq p_5 \leq 14 \))

(b) INTEROPT waits for the user to enter a change via a command of the following format:

\[ j[p_j'] \quad \overline{Q} \]

where \( j \) is the parameter identification number (as given in the display list) and \( p_j' \) is the new value of this parameter.

The new value \( p_j' \) is not required in case of parameter number 2 since it has two values only and it is changed from INIT to STOP or vice-versa by simply entering its identification number (i.e. 2 \( \overline{Q} \)).

(c) If the user enters \( \overline{Q} \) only (indicating no change), INTEROPT proceeds to step (d); otherwise, it returns to step (b) to receive further input.

(d) The newly assigned values of the parameters are displayed and INTEROPT returns to the control mode.
2.6 Summary Table Display

Data from each iteration, in the course of the run, is stored in a summary table which has four columns. The left most column contains the iteration number, the second column contains the iteration number of the current cycle (which will be different from the first column only if a restart has occurred), the third column contains the total number of function evaluations that have taken place and the rightmost column gives the criterion function value associated with the iteration number.

This table can accommodate the result of at most 118 iterations. It is updated by dropping some of its entries when this limit is exceeded. But, at the same time, data about the most recent iterations (maximum of 30) is stored in a sub-table to maintain complete data about the most recent past. Summary tables (but not sub-tables) of previous runs (at most 5) are stored in order to allow access to the results of earlier runs.

To display a summary table, INTEROPT must be in the control mode. The user then enters the following command:

```
TABLE[/\rho][,r] Q
```

where

(a) $\rho$ is the row number of the first row to be displayed from the table (at most 30 rows are shown in one display). If $\rho$ is omitted, the display will start from the first row of the table.

(b) $r$ specifies the run number for which the summary table is to be displayed. $r$ should refer to either the current run or one of the immediately preceding five runs. The current run
is assumed if $r$ is omitted.

To display a sub-table, the following command is used in the control mode:

```
TABLE, SUB
```

With any table or sub-table display, INTEROPT also displays a summary of the run conditions associated with the run. After the display of a table or sub-table, INTEROPT returns to the control mode when the user enters $Q$. The screen is erased before the message 'WHAT NOW?' appears.

2.7 Function Plots

Output routines in INTEROPT enable the user to obtain a plot of the values of the criterion functions generated during the run. The user may choose the horizontal co-ordinate in such a plot to be either number of iterations or number of function evaluations. These plots give an overall picture of the effectiveness of the minimization technique in driving the function value to its minimum. Furthermore, they enable the user to decide whether to continue using the same technique, try another technique, change associated parameters or change the linear search. The user may request a complete plot that covers the whole run (the results of some intermediate iterations may, however, not be available; e.g. in case of updating the summary table) or a plot of only the most recent iterations (at most 30).

At most five of the complete plots (covering a whole run) may
be simultaneously displayed on the CRT. This feature provides a means of comparing the results of different runs having different operating conditions. These plots must be selected from the current run and the five immediately preceding runs.

These output options are accessible from the control mode via the following two commands:

\[ \text{PLOT}[/\text{ITER}], \text{SUB } \overline{Q} \]

\[ \text{PLOT}[/\text{ITER}][, r_1, r_2, \ldots, r_k ] \overline{Q} \]

where

(a) \text{ITER} is a request for a plot of function value against number of iterations. If \text{ITER} is omitted from the command, the horizontal co-ordinate of the plot is the number of function evaluations.

(b) \text{SUB} is a request for a plot pertaining to the most recent iterations (the last 30 or less).

(c) \( r_1, r_2, \ldots, r_k (k \leq 5) \) are the run numbers for which the plots are required. Each \( r_j \) must be the run number of either the current run or one of the immediately preceding five runs. If no run numbers appear, then a plot for the current run is given.

For each function plot displayed, a summary of the last run conditions associated with the corresponding run is also displayed.

In addition the following information is provided on the CRT:

(a) Specification of the horizontal co-ordinate; i.e., number of function evaluations or number of iterations.

(b) The range of the function values displayed (maximum and minimum values).
(c) The range of the horizontal co-ordinate (maximum and minimum values).

Note that a logarithmic transformation may be applied to the function values prior to plotting if the rate of the function reduction is very rapid. This is decided on by the system, but the user can intervene as described below.

2.7.1 The Cursor Mode

After plotting, INTEROPT places the Tektronix in the cursor mode in which the cross-hairs of the cursor appear on the CRT (see section 2.1). Only the x-cursor (the vertical line) however, has relevance. This cursor may be positioned by the user via the manual control and by entering any character from the keyboard, the co-ordinates of the point on each displayed curve at the intersection of the curve and the vertical cursor line, are displayed (in appropriate units).

As noted above, the vertical co-ordinate of the displayed curve(s) may be subjected to a transformation under program control. If it is desired to over-ride the alternative selected by the program and to obtain the alternative opposite to the one displayed, the user simply enters $\bar{T}$.

A return to the control mode is achieved by entering $\bar{Q}$.

2.8 Status Information

The user can request the display of status information about the current iteration by entering the following command while
INTEROPT is in the control mode:

\text{ITERATION  \overline{Q}}

The information displayed is as follows:

\# OF ITERATIONS = n_1
\# OF SUB-ITER. = n_2 (n_3)
\# OF FUNC EVAL. = n_4

THE ARGUMENT

\begin{align*}
  x_1 \\
  x_2 \\
  \vdots \\
  x_n \\
\end{align*}

FUNC VALUE = f(x)

THE GRADIENT

\begin{align*}
  g_1 \\
  g_2 \\
  \vdots \\
  g_n \\
\end{align*}

GRAD NORM = \left[ \sum_{i=1}^{n} g_i^2 \right]^{1/2}

where
\begin{itemize}
  \item n_1 \text{ is the number of iterations executed so far,}
  \item n_2 \text{ is the number of iterations since the last restart (this line will be displayed only if a restart has occurred)}
  \item n_3 \text{ is the number of restarts that have occurred so far,}
  \item n_4 \text{ is the number of function evaluations so far,}
  \item (x_1, x_2, \ldots, x_n) \text{ the current value of the argument,}
  \item f(x) \text{ is the corresponding function value,}
  \item (g_1, g_2, \ldots, g_n) \text{ the current value of the gradient,}
\end{itemize}
and \[ \left( \sum_{i=1}^{n} \frac{g_i^2}{\bar{g}_i^2} \right)^{1/2} \] is the corresponding value of the gradient norm.

Note that in a case where a gradient independent technique is being used, no gradient information is displayed.

After this display, INTEROPT returns to the control mode.

2.9 Execution of the Minimization Procedure

INTEROPT does not execute the minimization procedure continuously but rather steps at certain "decision points" and gives control back to the user (i.e. returns to the control mode). This gives the user the opportunity to examine the progress of the solution and to decide whether changes in some of the operating conditions might be worthwhile. Alternately, he may decide to stop the run.

A decision point is established prior to the execution of the minimization procedure by setting the "run control cycle" which is the number of iterations to be executed before INTEROPT returns to the control mode. The specification of the run control cycle is given in the control mode whenever the user requests the minimization procedure to be started or continued for a given run. Two commands have relevance for starting or continuing the execution; namely:

\[ \text{GO} \quad \bar{Q} \]

and

\[ \text{RESTART} \quad \bar{Q} \]

where the second alternative implies an initialization of the minimization algorithm prior to continuing. The system then requests from the user the data for the run control cycle. In addition, it
requests the data for the "linear search display cycle" which specifies the linear search with which the user wishes to interact (see section 2.12).

After entering one of the commands GO or RESTART, the following is displayed:

```
RUN CONTROL CYCLE = c_1
LS DISPLAY CYCLE = c_2
CHANGE?
```

where $c_1, c_2$ are current values. The user can then enter the desired changes in the following format:

```
c'_1, c'_2, Q
```

where $c'_1$ or $c'_2$ or both can be omitted if current values are to be maintained. (In case of specifying a new value for $c_2$ only, the comma must be included in the response).

Both the run control cycle and the linear search display cycle are assigned the default value of 5 at the beginning of each run.

After receiving the user's response to the CHANGE? request, the minimization procedure is executed until one of the two cycles is completed or one of the stopping conditions is satisfied (section 2.5). When the run control cycle is completed, the current status is displayed (as described in section 2.8) and INTEROPT enters the control mode. In the case where the program stops because of a stopping condition, the current status is also displayed together with the reason for the stop. The linear search display cycle is completed, a graph of $F(\alpha)$ is displayed on the CRT screen and the system enters the interactive linear search mode (section 2.12).
2.10 Output Documentation

For documentation purposes, INTEROPT provides line-printer output for each INTEROPT run as well as for the whole INTEROPT job.

The output documentation associated with each INTEROPT run consists of the following items:

(a) The general header, which is a text specified by the user to identify the problem and the output. This text (which can be any character string) is entered via an input data card called the "problem description card". This card can be completely omitted. In this case, no general header appears in the output.

(b) A summary of the operating conditions, which consists of the run conditions and the values of the initialization and stopping parameters for the run.

(c) The initialization data which consists of the priming guess, the corresponding value of the criterion function and, if a gradient dependent minimization algorithm is being used, the gradient vector and its norm.

(d) Output from each iteration, which is specified by selecting an option related to the amount of data to be outputted from each iteration (see section 2.11).

(e) A summary of the results generated in the course of the run which is presented in the summary table described earlier (section 2.6). This is provided at the completion of the run.

For the whole INTEROPT job a final documentation is provided which lists each of the runs in turn and gives the following
information for each of them:

MINTEC - The minimization technique used.

G$PAR - The value of the gradient parameter.

[NA (Not Applicable), in case of the gradient independent methods].

LN$SRH - The linear search algorithm used.

LS$PAR - The value of the linear search parameter.

RESTRT - The value of the restart parameter.

[NA indicates the absence of a specification].

E$PAR - The value of the exit parameter.

Note that for all the above items, the values given are those which were in effect at the completion of the run.

TCODE - One of the following codes which serves to specify the reason for the termination (see section 2.5):

GR$E the gradient norm less than $\epsilon=0.01 \times 10^{-p}$, where $-p$ is the value specified by E$PAR$.

FC$E$ the relative function change less than $\epsilon'=10^{-p}$.

IT=L number of iterations equals the limit value specified by MXITER.

FE=L number of function evaluations equals or exceeds the value given by MXFE

F$NR$ failure to make progress on an iteration and automatic restart not requested; i.e. FAIL=STOP.

F$IL$ failure to make progress on the first iteration of a restart cycle.

STOP the run is terminated by the user; i.e. a new run is initiated before any of the previous termination
criteria occurs.

#ITER - The number of iterations carried out in the run. In addition, the number of restarts which occurred is given in parenthesis.

#EVAL - The number of function evaluations made during the run.

F-INIT - The initial value of the criterion function; i.e. the value corresponding to the point where the run started.

F-FINAL - The final value of the criterion function.

In addition to the above information, the priming argument used in each run is given together with the final estimate of the minimizing argument generated in the run.

2.11 Printer Output From Each Iteration

The user may wish to generate information characterizing each iteration in the course of the run. There are two options in this respect:

Option 1 With this option, the following items are given:

(i) the iteration number (with additional information if a restart has occurred),

(ii) the current value of the criterion function,

(iii) the corresponding value of the function argument,

and (iv) the number of function evaluations that have taken place thus far.

Option 2 The output begins with item (i) in Option 1, then information about the linear search (or searches) within the iteration is given followed by the other three items in Option 1. In the
case of a gradient dependent algorithm the following items are also provided:

(v) the current gradient vector,
(vi) the euclidean norm of the gradient,
(vii) the search direction used in arriving at the current estimate of the minimizing argument,
and (viii) the cosine of the angle between the current gradient vector and the search direction.

The specification of the line printer output option desired by the user takes place when INTEROPT is in the control mode, via the command:

\[ \text{OUTPUT}[\omega] \]

where \( \omega \) is the option number; hence, it is 1 or 2. In fact, \( \omega \) may also be 0 which implies that the output from each iteration is to be completely suppressed. When \( \omega \) is omitted from this command, INTEROPT displays a brief description of the options and requests a selection from the user. The display in this case is as follows:

PRINTER OUTPUT FOR EACH ITERATION

ENTER THE DESIRED OPTION IDENTIFICATION NUMBER

0 NO OUTPUT
1 ITER#, FUNC & ARG VALUES, # OF EVAL
2 SAME AS 1 PLUS GRAD & LINEAR SEARCH INFO.

Note that each INTEROPT run begins with \( \omega \) set to zero.

Finally, it should be noted that when \( \omega=0 \), appropriate line printer output is produced to document user modifications to the run conditions or to the initialization and stopping parameters, during the course of the run. Also appropriate reasons are
documented to clarify the initialization of an algorithm or its stopping.

2.12 The Linear Search Display

As mentioned earlier, the minimization algorithms within INTEROPT require a mechanism for solving the following one-dimensional minimization (or linear search) problem.

Find the scalar \( \alpha^* \) such that

\[
F(\alpha^*) = \min_{\alpha} F(\alpha)
\]

where \( F(\alpha) = f(x+\alpha s) \)

and \( x \) and \( s \) are given n-vectors

INTEROPT provides three different algorithms for handling this problem (see section 2.4).

The input to each of these algorithms is an "initial interval of uncertainty" (IIOU), namely an interval on the real line which is (more or less) guaranteed to contain \( \alpha^* \). The algorithms themselves then proceed to locate an estimate of \( \alpha^* \) within this interval.

INTEROPT provides the user with various interactive capabilities relative to the solution of the linear search problem. These capabilities can be exercised at the end of each "linear search display cycle". (The number \( c_2 \) referred to in section 2.9). At the end of a linear search display cycle, (i.e. after \( c_2 - 1 \) linear searches have been performed completely by INTEROPT) the main minimization procedure is interrupted and control over the current linear search is passed to the user.
This activity begins with a display of $F(\alpha)$ which is generated on the screen using 10 values of $\alpha$. Via the procedures described below, the user has the choice of either (a) simply examining the form of the $F(\alpha)$, curve, (b) prescribing the IIOU to be passed on to the selected linear search algorithm, or (c) actually specifying the point which he feels is the solution and by-passing the linear search algorithm altogether.

Note that A and B are used to denote the left-end and the right-end points of the interval to be considered from $F(\alpha)$. Initially, they are the limits of the first displayed interval of $F(\alpha)$. The user may change these, as described below, if he so wishes. Later, the interval determined by the final values of A and B can be used as the IIOU or to display a larger or smaller portion of $F(\alpha)$.

After the initial display of $F(\alpha)$, the Tektronix is placed in the cursor mode and the user is given various interrogation/specification capabilities by entering a single character as described below:

P - the value of $\alpha$ and $F(\alpha)$ corresponding to the point at the intersection of the vertical cursor line and the curve are displayed (This requires one function evaluation).

X - same effect as P, except that only the $\alpha$ value is displayed (hence no function evaluation is required).

A - the co-ordinate on the horizontal axis corresponding to the current position of the vertical cursor line, is interpreted as the user's specification of the left-hand end point of the IIOU.
B - analogous to A except that the specification is that of the right-hand end point of the IIOU.

M - also analogous to A except that the point is interpreted as the user's specification of \( a^* \). In this case the current linear search problem is taken to be solved.

E - after this entry the system responds with the following message:

\[ \text{EXTENSION, ENTER A or B} \]

The user is then able to enter a new value for either A or B. Since this is done mainly to extend the current interval to the left or to the right, the entered number should either be to the left of the current A value or to the right of the current B value.

D - a new display of \( F(a) \) is generated using the current values of A and B as the display extremities.

R - the current values of A and B are taken to define the IIOU and this interval is passed to the specified linear search algorithm which is then activated.

G - the interactive linear search activity is terminated and INTEROPT simply proceeds with its normal algorithmic procedure for conducting the linear search. The count of function evaluations made during the user's interaction with the current linear search is discarded.

The system's response to any one of character inputs, R H or G is the display of the message:

\[ \text{LS DISPLAY CYCLE = } c_2. \text{ CHANGE?} \]

where \( c_2 \) is the current value of the linear search display cycle.
The value of $c_2$ can be altered by entering any positive integer. A null response (simply $0$) results in the system maintaining the current value of $c_2$. At this point INTEROPT resumes execution of the minimization procedure.

2.13 The User-Provided Subprograms

The user is required to prepare at least one FORTRAN subprogram which specifies the criterion function. Other subprograms can also be added to extend the features of INTEROPT. The different possibilities are described below.

(a) The Function Subprogram $F$

The user must prepare a DOUBLE PRECISION FUNCTION subprogram called $F$ which serves to specify the criterion function to be minimized. The required structure of this subprogram is as follows:

```
DOUBLE PRECISION FUNCTION F(X)

IMPLICIT REAL*8 X(8)

{ statements specifying
  \{ F= \} the criterion function

RETURN

END
```

where X is the argument vector.

(b) The Subroutine Subprogram GRAD

If the user wishes to use a gradient dependent minimization algorithm with analytic gradient information, then a subroutine subprogram called GRAD is required. GRAD serves to calculate the
gradient of the function from an explicit specification of the partial derivatives; i.e. \( \frac{\partial f}{\partial x_i} \), \( i=1,2,...,n \). The required structure of this subprogram is as follows:

```fortran
SUBROUTINE GRAD (X,G)

IMPLICIT REAL*8 (A-H, O-Z)

REAL*8 X(8), G(8)

\[
\begin{align*}
G(1) &= \\
G(2) &= \\
\vdots &
\end{align*}
\]

statements specifying the components of the gradient vector.

where \( G \) is the gradient vector.

(c) The Subroutine Subprogram YOURSA

If the user wishes to utilize gradient dependent minimization scheme which is not available in the package, a SUBROUTINE subprogram called YOURSA is required. The algorithm coded in YOURSA would typically operate on \( x^k \) and \( g^k \) (the gradient vector at \( x^k \); i.e., \( g^k = f_x(x^k) \)) to produce \( s^k \) (a search direction). The required structure of this subprogram is as follows:

```fortran
SUBROUTINE YOURSA (X,G,S)

IMPLICIT REAL*8 (A-H, O-Z)

REAL*8 X(8), G(8), S(8)

\[
\begin{align*}
\text{RETURN} \\
\end{align*}
\]

where \( S \) is the search direction vector.
(d) The Subroutine Subprogram YOURSB

If the user wishes to utilize a gradient independent minimization scheme which is not available in INTEROPT, a SUBROUTINE subprogram called YOURSB is required. The algorithm coded in YOURSB would typically generate directly a new estimate of the minimizing argument using the current estimate as input (i.e. generate $x^{k+1}$ given $x^k$). The implementation of the procedure may require direct access to the linear search algorithms within INTEROPT since it may be necessary to solve one or more linear search problems within the context of the algorithm. This access is achieved via a call to the subroutine SCAN (XD,SD,ALPHA) where the arguments have the following interpretation:

XD is the base point of the linear search
SD is the direction along which the linear search is conducted
ALPHA is the optimal distance from XD to SD.

Note that upon return from SCAN, XD is updated to contain the minimizing argument along the SD direction.

The required structure of YOURSB is as follows:

SUBROUTINE YOURSB (X,FLAG)

IMPLICIT REAL*8 (A-H,O-Z)

REAL*8 X(8)

INTEGER FLAG

{} statements for updating the value of X from $x^k$ to $x^{k+1}$

RETURN

END

The argument FLAG is used as an indication to indicate whether
adequate progress has been achieved during the iteration. It should normally be set to 0, and set to 1 when a lack of progress has been detected.

(e) **The Subroutine Subprogram YOURSL**

The user can utilize a linear search algorithm of his own choice by appropriately coding a SUBROUTINE subprogram called YOURSL. The purpose of this algorithm is to find the value of the scalar argument \( a \) of the function:

\[
F(a) = f(x^k + as^k)
\]

which yields the smallest value of \( F(a) \), \( (x^k, s^k) \) are taken to be given. When coding this subroutine, the evaluation of \( F \) at some argument \( AA \) is achieved via a call to the FUNCTION subprogram FUNCS (e.g. \( Y=\text{FUNCS}(AA) \)). The required structure of YOURSL is as follows:

```
SUBROUTINE YOURSL (ALPHA,FMIN)
IMPLICIT REAL*8 (A-H,O-Z)
{
  statements required in the algorithm
}
RETURN
END
```

Upon exit, ALPHA should contain the estimate of the minimizing argument of \( F(a) \) and FMIN should contain the value of \( F \) at ALPHA.

2.14 Constrained Minimization

INTEROPT provides a very simple means for handling constrained minimization problems based on "Interior Penalty Function Method" [3].
This method is based on defining a "penalty" function of the form

\[ P(x, R) = f(x) + R \sum_{i=1}^{n_c} h_i(x) \]

where \( f(x) \) is the criterion function,
\( h_i(x) \) for \( i = 1, 2, \ldots, n_c \) are the constraints which are assumed to have the form
\( h_i(x) \leq 0, \)

and \( R \) is a positive scalar.

To find an estimate of the minimizing argument \( x^* \), several INTEROPT runs, with different values of \( R \), are required. \( R \) is chosen arbitrarily at the first run, then it is reduced at the beginning of each successive run. The requirements for this type of jobs are (a) a special structure for the user-provided FUNCTION subprogram \( F \), and (b) modifying \( R \) at the beginning of each run.

In this case, the user provided subprogram \( F \) will have the following general structure:

```fortran
DOUBLE PRECISION FUNCTION F(X)
IMPLICIT REAL 8 (A-H,O-Z)
REAL 8 X(8), H(nc)
LOGICAL 1 PENFLG
COMMON /PENALT/R,PENFLG
PENFLG = .TRUE.
{ statements to compute }
{ F = ... } f(x)
{ H(1) } statements to compute
{ H(2) }
{ \vdots } the constraints \( h_i(x) \)
```

DO 10 I=1,h
    IF (H(I).GT.O.ODO) GO TO 20
10    F=F R/H(I)
    RETURN

20    F=1.0D10
    RETURN

END

In the above, PENFLG is a logical flag which is set to "TRUE" to indicate that a priority function problem is being considered and statement number 20 is used to make the value of the penalty function very high in the forbidden region.

R is (initially) set to zero by the program. At the beginning of the first run as well as at the beginning of each new run, the following message is displayed:

    R=a CHANGE?

where a is the current value of R. The user, then, enters the new value of R (which must be positive). The program then proceeds to the priming guess specification as described in section 2.3. Except for the first run, the CONTINUE option would normally be selected.

2.15 Diagnostic Messages

Several diagnostic messages are provided in INTEROPT to signal improper user responses. These are described in this section.

(a) If an error occurs in specifying a numeric value, the following message is displayed:
INVALID NUMBER, PLEASE RE-ENTER

The whole user command or specification, which contains this number, should be re-entered; e.g. if when specifying the priming guess (section 2.3), the following is entered:

\[-3.0, -1.EE2, +5.0D1, 1 \bar{Q}\]

the above error message will be displayed. The corrected specification should then be re-entered by the user; e.g.

\[-3.0, -1.E2, +5.0D1, 1 \bar{Q}\]

(b) If an inadmissible value is entered for a parameter in the program, the following message is displayed:

ERROR, PLEASE RE-ENTER

INTEROPT then awaits the corrected specification to be re-entered.

(c) In case of an error within a command in the control mode, one of the following messages is displayed:

(i) ERROR, PLOT/?

(the command should have been PLOT or PLOT/ITER)

(ii) ERROR, OUTPUT,?

(the command should have been OUTPUT or OUTPUT,ω where ω is 0, 1 or 2)

(iii) INVALID COMMAND

(the entered command is not one of the admissible set)

(iv) INVALID NUMBER

(the command contains an invalid numeric value)

(v) INVALID RUN#

(for table display or function plot, the specified run number is not within the permitted range)
(vi) TABLE CONTAINS $\rho$' ROWS ONLY

(the user has specified a row number, $\rho$, (in the command
TABLE/$\rho$,r) which is greater than the actual number of rows,
$\rho'$, in this table)

INTEROPT, in all the above cases, returns to the control mode.

(d) In the explicit specification of the priming guess, the
following message may be displayed:

N=n BUT YOUR GUESS VECTOR HAS n' COMPONENTS; PLEASE
RE-ENTER

This means that the number of components $n'$, entered for the
priming guess by the user, does not equal the actual dimen-
sion $n$, of the argument vector.

(e) In the linear search display, one of the following diagnostic
messages may be displayed:

(i) INSIDE THE CURRENT INTERVAL

(the user requested the graph to be extended and gave a limit
which in fact lies within the presently displayed interval;
the value entered is ignored)

(ii) NEG VALUES ARE NOT ADMISSIBLE

(the user requested the graph to be extended and specified a
negative value for the left-end of the interval; the value
entered is ignored.

(iii) SPECIFIED IOU HAS LENGTH < .001

(the user requested a new interval of the function $F(a)$ to be
plotted but the length of the new interval is less than the
allowed limit of .001)

In all the above cases, INTEROPT returns to the cursor mode.
of the linear search display.

(f) When changing the run conditions, one of the following warnings may appear:

(i) SECOND CHANGE IS IGNORED
(the user has twice specified values for either MINTEC or LN$SRH; the first one is taken)

(ii) GRADIENT INFO. IS NOT REQUIRED
(the user has specified a value for G$PAR while the minimization technique is gradient independent; the specification is ignored)

(iii) ANALYTIC GRAD IS NOT AVAILABLE
(the user has assigned G$PAR the value AG but the subroutine GRAD has not been submitted; the current value of G$PAR is retained)

(iv) DEFAULT VALUE OF G$PAR IS USED
(the user has assigned G$PAR a negative value; G$PAR is assigned the value AG if the subroutine GRAD has been submitted or the numeric value 9 otherwise)

(v) DEFAULT VALUE OF LS$PAR IS USED
(the user has assigned LS$PAR a negative value; LS$PAR is given the default value 2)

(g) If the user attempts to assign a value to RESTRT or change FAIL while using GRAD to ESQ as the minimization technique, the following message is displayed:

NOT INIT IN CASE OF GRAD/ESQ

and the specification is ignored.

(h) While receiving an input from the Tektronix, the system may
experience an I/O error. This situation is brought to the user's attention via the message

I/O ERROR, PLEASE RE-ENTER

The user should re-enter his last command or specification.

2.16 Line Drop

Generally in the case of a telecommunication line failure, the INTEROPT job is not terminated. Instead, it detects the situation and waits until the user signs on again. It then starts from the last display which existed on the screen prior to the line drop. However, if more than three telecommunication line failures occur, then the INTEROPT job automatically terminates.

2.17 Synopsis of the Operating Procedures

The sequence of flow-charts in figures 2.1 to 2.12 provide a synopsis of the operating procedures for INTEROPT. In these flow-charts, a box drawn with solid lines is used to indicate the display of information or messages on the screen. A box drawn with a broken line indicates a stage where a user response is required. In some cases, a broken-line box is horizontally separated into segments by broken lines. This indicates that a response of the form shown in any one of the sub-boxes is admissible. A box (broken-line or solid-line) with an asterisk '*' indicates that this stop is executed only in case of a penalty function run.
START

R=r CHANGE?

[r']\bar{Q}

WHAT IS YOUR PRIMING GUESS?

0, \ x_1^0, \ x_2^0, \ldots, \ x_n^0 \ \bar{Q}

CURRENT PRIMING GUESS IS

\[ x_1^0 \]
\[ x_2^0 \]
\[ \ldots \]
\[ \ldots \]
\[ x_n^0 \]

FUNC VALUE = f(x^0)

DO YOU WISH TO RESPECIFY THE PRIMING GUESS?

YES \ \bar{Q}

NO \ \bar{Q}

2
RUN CONDITIONS

MINTEC = m_1 / G$PAR = a_1
LN$SRH = m_2 / LS$PAR = a_2
CHANGE?

3

RUN CONDITIONS

MINTEC = m'_1 / G$PAR = a'_1
LN$SRH = m'_2 / LS$PAR = a'_2
WHAT NOW?

CONDITION $\bar{Q}$

INITIAL $\bar{Q}$

OUTPUT, $\omega \bar{Q}$

OUTPUT $\bar{Q}$

ITERATION $\bar{Q}$

TABLE, SUB $\bar{Q}$

TABLE [$/p$ [$,r]$ $\bar{Q}$

PLOT [$/ ITER$, SUB $\bar{Q}$

PLOT [$/ ITER$ [$,r_1, r_2, \ldots, r_k$] $\bar{Q}$

NEW RUN $\bar{Q}$

GO $\bar{Q}$

RESTART $\bar{Q}$

END JOB $\bar{Q}$

END
INITIALIZATION & STOPPING PARAMETERS

1 RESTRT = p_1
2 FAIL = p_2
3 MXITER = p_3
4 MXFE = p_4
5 ESPAR = p_5

CHANGE?

INITIALIZATION & STOPPING PARAMETERS

1 RESTRT = p'_1
2 FAIL = p'_2
3 MXITER = p'_3
4 MXFE = p'_4
5 ESPAR = p'_5

Fig. 2.4
PRINTER OUTPUT FOR EACH ITERATION
ENTER THE DESIRED OPTION IDENTIFICATION NUMBER
0 NO OUTPUT
1 ITER #, FUNC & ARG VALUES, # OF FUNC EVAL
2 SAME AS 1 PLUS GRAD & LINEAR SEARCH INFO.

Fig. 2.5
# OF ITERATIONS = \( n_1 \)
# OF SUB-ITER. = \( n_2(n_3) \)
# OF FUNC EVAL. = \( n_4 \)

THE ARGUMENT

\[
x_1 \\
x_2 \\
\vdots \\
\vdots \\
x_n
\]

FUNC VALUE = \( f(x) \)

THE GRADIENT

\[
g_1 \\
g_2 \\
\vdots \\
\vdots \\
g_n
\]

GRAD NORM = \( \left[ \sum_{i=1}^{n} g_i^2 \right]^{\frac{1}{2}} \)
The subtable of the current run is displayed

RUN SUMMARY (SUB-TABLE)

ITER    SUBITR    #EVAL    FUNCVALUE
n₁      n₂      n₄      f(ₓ)

A summary of the run conditions is also displayed

Fig. 2.7
The summary table of the current run [or run number r] is displayed starting from row number ρ.

**RUN SUMMARY**

<table>
<thead>
<tr>
<th>ITER</th>
<th>SUBITR</th>
<th>#EVAL</th>
<th>FUNC VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>n₁</td>
<td>n₂</td>
<td>n₄</td>
<td>f(x)</td>
</tr>
</tbody>
</table>

A summary of the run conditions is also displayed.
Plot: function value (a logarithmic transformation may be used), Y-axis, versus number of function evaluations [or number of iterations], X-axis; the subtable of the current run is used. A summary of the run conditions is also displayed.

Plot: function value (a logarithmic transformation may be used), Y-axis, versus number of function evaluations [or number of iterations], X-axis; the summary table[s] of the current run [or of runs number $r_1, r_2, \ldots, r_k$] is [are] used. A summary of the run conditions is also displayed.

For each displayed curve, the coordinates of the point whose abscissa is given by the vertical cursor are displayed.

Transformation of the vertical coordinates; obtain the alternative opposite to the current one.
CURRENT PRIMING GUESS IS
\[ x_1^0, x_2^0, \ldots, x_n^0 \]

FUNC VALUE = \( f(x^0) \)

CURRENT ARGUMENT IS
\[ x_1, x_2, \ldots, x_n \]

FUNC VALUE = \( f(x) \)

REPEAT, CONTINUE OR NEW?
RUN CONTROL CYCLE = $c_1$
LS DISPLAY CYCLE = $c_2$
CHANGE?

Execute the minimization technique until one of the following events occurs

<table>
<thead>
<tr>
<th>Run control cycle is completed</th>
<th>A stopping condition is satisfied</th>
<th>LS display cycle is completed</th>
</tr>
</thead>
</table>

The reason for stopping is displayed

Fig. 2.11
Plot: $F(\alpha)$, Y-axis, versus $\alpha$, X-axis, for the interval $\alpha=(A,B)$ (initially chosen by the system); $F(\alpha)=f(x+as)$
CHAPTER 3

INTEROPT ORGANIZATION AND SUBROUTINES

3.1 General Structure and Flow-Charts

The INTEROPT program can be separated into the following four segments or phases:

1. The Interactive Phase
2. The Minimization Phase
3. The Single Iteration Execution Phase
4. The Linear Search Phase

Each of these phases is supervised by a "principal" subroutine: namely INTACT, MONITR, MINMIZ and SCAN respectively. The structure of each of the four phases is described below and in the flow-charts of Fig. 3.1 through Fig. 3.6. In these flow-charts, the vertical flow on the left side of the figure (Fig. 3.3 is excepted) represents the sequence of events within the subroutine under consideration, while the horizontal flow on the right side of the figure represents subroutine calls where the name of the called subroutine is given in a box to the right of the calling subroutine. The subroutines associated with the four phases are described in the next section.

3.1.1 Interactive Phase

The subroutine INTACT is called by the main program when the INTEROPT package is started (Fig. 3.1). INTACT performs four main tasks (Fig. 3.2) as described below:
Task 1 (Set-up of a new run)

The priming guess for the run is obtained via a call to NEWRUN and
the run conditions are established via a call to CONCHN. Then,
INTACT proceeds to Task 2.

Task 2 (Supervision of the control mode)

INTACT accepts user commands within the control mode and then
calls an appropriate subroutine (Fig. 3.3) as summarized below:

<table>
<thead>
<tr>
<th>User command</th>
<th>Subroutine called</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONDITION</td>
<td>CONCHN</td>
</tr>
<tr>
<td>INITIALIZATION</td>
<td>INTCHN</td>
</tr>
<tr>
<td>OUTPUT ...</td>
<td>OUTCHN</td>
</tr>
<tr>
<td>ITERATION</td>
<td>DISITR</td>
</tr>
<tr>
<td>TABLE ...</td>
<td>DISTAB</td>
</tr>
<tr>
<td>PLOT ...</td>
<td>FUNPLT</td>
</tr>
</tbody>
</table>

For each of these commands, the system returns to the control mode
after the processing of the command is completed. Each of the
remaining commands (NEW RUN, RESTART, GO and END JOB) directs
INTACT to one of its four tasks as summarized below:

<table>
<thead>
<tr>
<th>User command</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW RUN</td>
<td>1</td>
</tr>
<tr>
<td>RESTART or GO</td>
<td>3</td>
</tr>
<tr>
<td>END JOB</td>
<td>4</td>
</tr>
</tbody>
</table>

Task 3 (Starting the minimization phase)

INTACT first calls RUN to handle any output documentation and
information storage required prior to the minimization phase and to establish the values of the parameters for the linear search and control cycles (see the description of RUN, section 3.3.2( ). It then calls MONITR. When a return from MONITR is made, INTACT returns to Task 2.

**Task 4 (Termination of the job)**

By way of terminating the job, INTACT first calls TABLE to print the summary table of the last run and then calls SMRY to print the job summary. A return to the main program is then made, which results in the termination of the job.

**3.1.2 Minimization Phase**

The minimization phase is concerned with the execution of the selected minimization algorithm; hence it is the main computational part of INTEROPT. The subroutine MONITR is called from INTACT to execute a series of iterations. This series is terminated either if one of the stopping conditions is satisfied or a control cycle is completed. MONITR, as shown in Fig. 3.4, proceeds as follows:

(a) When the current iteration is the first one in a new run, the minimization algorithm is initialized, the documentation describing the run is printed, and the information to be retained for the summary table is stored via a call to the subroutine ACCESS.

(b) Prior to each iteration, MONITR performs tests to determine
if:

(i) A stopping condition is satisfied or the control cycle is completed. In this circumstance the present status is displayed via the subroutine DISITR. In the case of satisfying a stopping condition, the reason for stopping is also displayed (and may be printed if the user has requested detailed output from each iteration) via the subroutine MSSAGE.

(ii) A restart is required. Such a requirement could arise from several conditions: namely, a request for a restart has been entered by the user, a restart is necessary due to a change in MINTEC, an automatic restart is necessary because of the current value of RESTRT, or a restart is necessary due to a failure in the previous iteration. The reason for the restart is printed (via MSSAGE) together with the initialization data (via ACCESS) if detailed documentation has been requested.

In case of (i) MONITR returns to INTACT (the control mode), otherwise MONITR proceeds to step (c) below.

(c) The subroutine MINMIZ is called to execute one iteration.

Then ACCESS is called to store data for the summary table and, if it has been requested by the user, ACCESS prints data from the iteration. The procedure then returns to step (b) above.

3.1.3 The Single Iteration Execution Phase

The subroutine MINMIZ (Fig. 3.5) proceeds as follows:
(a) If the minimization technique being used is gradient dependent, one of the following subroutines is called according to the value of the parameter MINTEC:

   YOURSA for the user-provided technique
   CONJ for the conjugate gradient methods (Fletcher-Reeves, Polack-Ribiere and Sorenson)
   DFP for the Davidon-Fletcher-Powell method

Each of these subroutines utilizes the gradient vector to generate a search direction (of unit length) along which MINMIZ performs a linear search via a call to SCAN. If the method being used is the steepest descent method, then the search direction is generated directly within MINMIZ by simply normalizing the available negative gradient vector.

(b) If the minimization technique being used is gradient independent, one of the following subroutines is called according to the value of the parameter MINTEC:

   YOURSB for the user-provided technique
   POWELL for both Powell's method and the extended sequential search method
   ZANG for Zangwill's method

The linear searches required by these algorithms are executed via calls to the subroutine SCAN.

3.1.4 The Linear Search Phase

   The subroutine SCAN is called each time a linear search along
a given direction is required. It proceeds as follows (Fig. 3.6):

A call to the subroutine I$IOU is made. If the linear search being initiated is not the last one in a linear search display cycle, I$IOU determines an initial interval of uncertainty. This interval is passed to the selected linear search subroutine (YOURSL, QUADFT, FBNACI, GOLDEN) which establishes an approximation to the minimizing argument within the interval. The subroutine selection is controlled by the parameter LN$SRH and the basis for the respective algorithms is summarized below:

YOURSL for the arbitrary user-provided algorithm
QUADFT for the sequential quadratic fitting method
FBNACI for the Fibonacci search
GOLDEN for the Golden section search

Alternately if the linear search being initiated is the last one in the linear search display cycle, then the call to I$IOU initiates the interactive linear search procedure via a call to LS$DIS. Depending on the user's responses the initial interval of uncertainty or the minimizing argument itself, may be interactively generated. In the former situation, this initial interval of uncertainty is returned to SCAN which then calls the appropriate linear search subroutine for further processing. In the latter case, the task of SCAN is complete and a return is made.

3.2 Subroutines Associated with the Four INTEROPT Phases

As noted in the previous section, the principal subroutines perform their respective tasks via calls to secondary subroutines.
These second level subroutines and the subroutines associated with them in turn, are described briefly in this section.

3.2.1 GETN

This subroutine is used to determine the dimension, \( n \), of the vector argument \( x \) of the submitted function \( P \). It is called only once by the subroutine ARGMT. The priming guess is initially assumed to contain 8 components (the maximum allowed). Each of these is perturbed in turn and the effect of the perturbation on the function value is checked. The number of the components which affect the function value is taken as the required \( n \).

3.2.2 ARGDIS

This subroutine is called each time a display of one of the following is required:

1. The current priming guess vector (the last guess explicitly specified by the user) and the corresponding function value.
2. The current estimate of the minimizing argument and the corresponding function value.

3.2.3 ARGMT

This subroutine is called each time a priming guess is to be explicitly specified by the user. When called for the first time, it in turn, calls GETN to determine \( n \). It accepts the vector
entered by the user, checks the number of components in it against
n, and stores it as both the current priming guess and the current
estimate of the minimizing argument. The corresponding function
value is evaluated and both the priming guess vector and the func-
tion value are displayed via ARGDIS. Finally, ARGMTN enquires
from the user whether he wishes to respecify the priming guess.

3.2.4 NEWRUN

This subroutine is called from INTACT each time a new run is
requested. It resets to their respective default values, the
initialization & stopping parameters and the parameter controlling
the output documentation from each iteration. Then, if the run
being initiated is the first run in the INTEROPT job, it calls
ARGMT which accepts the user priming guess. Otherwise, it calls
ARGDIS twice, first to display the current priming guess, then to
display the current estimate of the minimizing argument. According
to the user response (REPEAT, CONTINUE, or NEW) it either assigns
the value of the priming guess to the new estimate of the minimizing
or keeps the present estimate of the minimizing argument, or calls
ARGMTN to obtain from the user a new priming guess, respectively.

3.2.5 CONDIS

This subroutine displays the current run conditions; i.e.
the current values of MINTEC, G$PAR (if applicable), LN$SRH and
LN$PAR.
3.2.6 EXAM

The user response for changing the run conditions \((m_1/a_1, m_2/a_2)\) is handled in two parts (i.e. \(m_1/a_1\) and \(m_2/a_2\)). EXAM handles one part of this response at a time. It substitutes a numeric value for the mnemonic \(m_j\) and determines the value of the associated parameter according to the value of \(a_j\).

3.2.7 CONCHN

This subroutine displays the current values of the run conditions via a call to CONDIS, handles the user-specified changes via two calls to EXAM and displays the run conditions after modification via another call to CONDIS.

3.2.8 INTDIS

This subroutine displays the initialization and stopping parameters in effect, i.e. the current values of RESTRT, FAIL, MXITER, MXFE and E$PAR$.

3.2.9 INTCHN

This subroutine is called when it is required to check and/or change the values of the initialization and stopping parameters. It calls INTDIS to display the current values of these parameters. Then, it accepts the user changes (one change at a time), examines the entered change and modifies the corresponding parameters.
Finally, it calls INTDIS to display the parameters after modification.

3.2.10 OUTCHN

This subroutine accepts the user choice of the available options for the output documentation (on the line printer) from each iteration. Unless the user enters the option number explicitly; e.g. OUTPUT, 0, OUTCHN displays a summary for the available options and, then, accepts the user's choice.

3.2.11 DISITR

This subroutine is used to display status information about the current iteration in the following cases:

(a) a stopping condition is satisfied
(b) a run control cycle is completed
(c) the user requests information via the command ITERATION

DISITR displays the number of iterations (and possibly the number of subiterations) and the number of function evaluations so far. Then it calls ARGDIS to display the current values of the argument and the criterion function. Also, it displays the gradient and the gradient norm if relevant.

3.2.12 CDIS

This subroutine displays the summary of the run conditions required each time a summary table or a function plot is displayed.
3.2.13  TABDIS

This subroutine is used to display the summary table for a
given run or the sub-table of the current run. Then it calls CDIS
to display the run conditions.

3.2.14  DISTAB

This subroutine examines the syntax of the commands:

TABLE, SUB  and  TABLE[/p][,n]

It then displays the appropriate table via a call to TABDIS

3.2.15  TRANS

This subroutine determines whether a logarithmic transformation
is required when five or more function values are to be plotted.
The following test is performed:

The curve to be plotted has different slopes within the different
segments (see below). The maximum slope is found and if more than
75% of the slopes have a value which is less than 10% of the maximum
slope, then logarithmic transformation is made. This corresponds
to a situation where the function value drops significantly during
a few iterations but very little change occurs in most of the other
iterations.
3.2.16 PREP

This subroutine applies a logarithmic transformation (base 10) to the function values where required. If the function has a negative value at some point, the most negative value attained by the function is used to bias the other values.

3.2.17 PLTMSG

This subroutine displays the heading for a function plot.

3.2.18 PLOT

This subroutine plots function value against either the number
of function evaluations or the number of iterations. It also displays the associated run conditions via a call to CDIS.

3.2.19 SRCH

This subroutine responds to a user request for co-ordinate data from the displayed curves as specified by the current vertical cursor position (section 2.7.1). If the displayed curve(s) is (are) plotted against number of iterations. Then a search is made of the stored data points to locate the one for which the number of iterations most closely corresponds to the vertical cursor position. The co-ordinates of this point (number of iterations and function value) are then displayed. An analogous operation takes place if the abscissa of the plotted curves is number of function evaluations.

3.2.20 PLT

This subroutine is used to generate a plot of function value against either number of iterations or number of function evaluations, using the summary table(s) of the current run and/or of previous runs. As described in section 2.7, the user may request one function plot from a particular run as well as a group of function plots from different runs. In both cases the following steps are executed:

(a) TRANS is called once for each specified run to determine
whether a transformation is necessary for the plot for this run.
If any plot requires a transformation then it is applied to all
plots.

(b) PREP is called once for each plot to obtain the transformed
function values if they are required.

(c) PLTMSG is called once to display the heading for the whole
graph.

(d) PLOT is called as many times as required to display the various
plots.

(e) The cursor mode is then entered (section 2.7.1) and according
to the character entered by the user one of the following actions
is taken:

(i) $\overline{Q}$ - a return is made to the calling subroutine (FUNPLT)
(ii) $\overline{T}$ - if the logarithmic transformation has been used for the
current display a new display without the transformation is
given and vice versa.
(iii) any other character - SRCH is called for each displayed curve
to display the co-ordinates of the points on each curve which
correspond to the vertical cursor position.

3.2.21 PLTSUB

This subroutine is similar to PLT but it is used for a function
plot produced from the sub-table data of the current run.

3.2.22 FUNPLT

This subroutine examines the syntax of the two commands

\[ \text{PLOT}[/\text{ITER}],[,]_{1}, r_{2}, \ldots, r_{k} \]

and \[ \text{PLOT}[/\text{ITER}], \text{SUB}. \]

For the first command it calls PLT and passes all the desired
options whereas for the second command it calls PLTSUB.

3.2.23 CHANGE

This subroutine prints out all the changes in the run conditions
and the initialization and stopping parameters which are specified
by the user in the course of the run. This is done only if the user
requests a detailed printer output.

3.2.24 TABLE

This subroutine prints out the summary table of the current
run when it is completed (i.e. when a new run is initiated).

3.2.25 TSTORE

The summary table of the current run is stored by this subrou-
tine when it is completed in order to keep a history file of the
most recent five runs. The tables of these runs are arranged such
that the new table replaces the "oldest" table in the list.

3.2.26 RUN

Each time the user enters either GO or RESTART, this subroutine is called to prepare for the execution of the minimization procedure and set up the run control and linear search display cycles. If a new run is being initiated, RUN initializes several pointers and calls TABLE and TSTORE to store the summary table of the previous run. Otherwise, it calls CHANGE, if required, to print out any changes that have been requested.

3.2.27 HEADER

This subroutine prints out the heading at the beginning of each run as well as the operating condition for the run.

3.2.28 UPDATE

This subroutine is used to update the stored summary table when the number of entries in this table is about to exceed the available storage for this table (119 entries). This is achieved essentially by deleting every second entry and then compacting the remaining entries.
3.2.29 ST$SUB

This subroutine stores one entry (corresponding to the results of one iteration) in the summary sub-table. Forty entries are allowed in the sub-table (only the most recent thirty rows are displayed). When this limit is exceeded, the first ten entries are deleted and the table is appropriately re-organized.

3.2.30 ACCESS

This subroutine is used to produce the appropriate print-out and store the information in the summary table. It calls HEADER at the beginning of each run. After each iteration, it prints out the results of this iteration (if required) and adds the proper entry in the summary table for this iteration. The subroutine UPDATE is called each time an update of the summary table is required. ACCESS also calls ST$SUB at the end of each iteration to create an entry for this iteration in the sub-table.

3.2.31 MSSAGE

Sometimes a message is displayed on the screen and/or on the line-printer after a certain iteration to inform the user that a stopping condition has been satisfied or the minimization algorithm is being initialized. This message is generated by the subroutine MSSAGE.
3.2.32 CONJ, DFP, POWELL and ZANG

These four subroutines contain the different minimization algorithms as described in Appendix A4.

3.2.33 I$IOU

The initial interval of uncertainty for a linear search is determined by this subroutine. This interval is passed to the selected linear search to locate the minimum value of the function within this interval. When a linear search display cycle is completed, I$IOU call LS$DIS (for the interactive linear search) and depending on the user decision (which is reflected on the type of return from LS$DIS) an appropriate action is taken.

3.2.34 QADMIN

This subroutine is used to fit a quadratic using three points. It then computes the minimum of the quadratic curve.

3.2.35 QUADFT, PBNACI & GOLDEN

The three linear search algorithms of INTEROPT are implemented within these three subroutines (see Appendix A2). QADMIN is invoked by each of these subroutines whenever a quadratic fit is required.
3.2.36 DRAW

This subroutine is used to plot the linear search display curve of the function \( F(a) \) (see section 2.12). Ten points, and hence ten function evaluations, are used.

3.2.37 LS$DIS$

This subroutine invokes DRAW to plot the linear search display curve, places the Tektronix in the cursor mode and accepts a user command (one character) in this mode. It then takes the appropriate action according to the user command.
Fig. 3.1 Starting the Program
**Fig. 3.2 Interactive Phase (Subroutine INTACT)**

1. **Initiate a new run**
2. **Run conditions for the run**
3. **Control Mode** *(The User Commands)*  
   3.1 *The command is END JOB, RESTART, GO or NEW RUN*  
   3.2 *Any other command*
4. **Preparation for the minimization phase**
5. **Minimization Phase** *(Subroutine MONITR)*  
   5.1 *Fig. 3.4.a & b*
6. **Final documentation**
7. **Return**  
   *Fig. 3.1*
Fig. 3.3 Control Mode
Fig. 3.4.a The Minimization Phase (Subroutine MONITR)
Display the present status

(a) A T.C. is satisfied
(b) Control cycle completed

Display (and print if required) the reason for termination

RETURN Fig. 3.2

DISITR ARGDIS

(T.C. = termination condition)

MSSAGE

Fig. 3.4.b
Fig. 3.5 The Single Iteration Execution Phase (Subroutine MINMIZ)
Determine the initial IOU or perform an interactive linear search

(a) user determined α*
(b) program determined α*

Use the appropriate linear search algorithm

RETURN Fig. 3.5

Fig. 3.6 The Linear Search Phase (Subroutine SCAN)
CHAPTER 4
EXAMPLES OF INTEROPT APPLICATIONS

4.1 Introduction

This chapter discusses the operating procedures and some applications of INTEROPT in the context of examples taken from some runs using standard test functions. These functions are described in Appendix A3. Unless stated otherwise, the standard priming guesses for these functions are used.

4.2 Examples to Illustrate the Operating Procedures of INTEROPT

Example 4.1

Several experiments with the Wood Function [Appendix A3.1] are used to illustrate some of the INTEROPT operating procedures. Fig. 4.1.1 through 4.1.53 show the dialog with INTEROPT and some comments on these figures is given in the following tabular summary. Note that the job submission included the GRAD subroutine.

<table>
<thead>
<tr>
<th>Figure</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Run</td>
<td>Priming guess specification</td>
</tr>
<tr>
<td>4.1.1</td>
<td>The components are entered explicitly since this is the 1st run.</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Run Conditions</td>
</tr>
<tr>
<td></td>
<td>The values displayed are the default values (1st run)</td>
</tr>
<tr>
<td></td>
<td>MINTEC is changed to F-R</td>
</tr>
<tr>
<td></td>
<td>LSPAR is changed to 4.</td>
</tr>
</tbody>
</table>
Control Mode

The command GO is used to start the minimization process. The values of the LS display cycle and control cycle are not changed from their current default values.

4.1.3 Linear Search Display

This is the 5th linear search. G is entered, in the cursor mode, to permit the linear search to be executed without the user interaction. The LS display cycle is changed to 6.

4.1.4 Control Mode

INTEROPT returns to the control mode after completing a control cycle of 5 iterations. OUTPUT is entered to permit specification of an option for the printer output from each iteration.

4.1.5 Printer Output for Each Iteration

The available options are displayed since no specification is given after OUTPUT in Fig. 4.1.4. Option 1 is selected.

Control Mode

GO is entered to continue the same run. The values of the LS display cycle and control cycle are not changed.

4.1.6 Control Mode

Another control cycle of 5 iterations is completed. The command PLOT is entered to obtain a plot of function value versus number of function evaluations.
4.1.7 Function Plot

Q is entered in the cursor mode to return to the control mode.

4.1.8 Control Mode

The command CONDITION is used to set the stage for subsequent changes in the run conditions.

4.1.9 Run Conditions

MINTEC is changed to D-F-P (this results in an automatic restart).

Control mode

GO is used to continue in the same run. The control cycle is changed to 10 iterations.

4.1.10 Linear Search Display

This is the 6th linear search since the previous display. G is used again, in the cursor mode (no user interaction). The LS display cycle is changed to 10.

4.1.11 Control Mode

A control cycle of 10 iterations has been completed. The command PLOT/ITER is used to obtain a plot of function value versus number of iterations.

4.1.12 Function Plot

The value of MINTEC is D-D-P (the last technique used in this run). The coordinates of a point on the curve are displayed by adjusting the x-cursor and entering any character. Q is used to return to the control mode.
4.1.13 Control

The command NEW RUN is used to initiate another run.

Second Run

4.1.14 Priming Guess Specification

REPEAT is specified, hence the run will start from the previous priming guess.

4.1.15 Run Conditions

Q is entered, i.e. no change in the run conditions.

Control Mode

OUTPUT is entered with option 2 explicitly specified. INITIAL is used to modify the initialization and stopping parameters.

4.1.16 Initialization and Stopping Parameters

RESTRT is set to 10.
MXITER is set to 20.

Control Mode

GO is used to start the minimization phase. Both the control cycle and the LS display cycle are changed to 25.

4.1.17 Control Mode

The iteration limit (MXITER) stops the run after 20 iterations. A plot of the results for run 1 and 2 is requested. (function value versus number of iterations).
4.1.18 Function Plot

Notice the beneficial effect of using two different minimization techniques during the first run. \( \bar{Q} \) is used to return to the control mode.

4.1.19 Control Mode

A display of the summary table of the first run is requested.

4.1.20 Summary Table

Notice that the run was initialized after the 10th iteration due to the change in the value of MINTEC. \( \bar{Q} \) is used to return to the control mode.

4.1.21 Control Mode

NEW RUN is entered to initiate a third run.

Third run

4.1.22 Priming Guess Specification

REPEAT is specified.

4.1.23 Run Conditions

MINTEC is changed to ZANG.
LNSSRH is changed to FBNACI.

Control Mode

GO is entered and the LS display cycle is changed to 50.

4.1.24 Control Mode

5 iterations (the default value) have been completed and hence the current status is displayed on the screen.
OUTPUT is entered with option 1 specified. CONDITION is entered.

4.1.25 Run Conditions
MINTEC is changed to POWELL.
Control Mode
GO is entered and no change in either the control or the LS display cycles is made.

4.1.26 Linear Search Display
The 50th linear search since starting the run is occurring. G is entered and no change is made in the LS display cycle.

4.1.27 Control Mode
5 more iterations have been completed. OUTPUT is entered with option 0 specified (line printer output is suppressed for subsequent iterations).
CONDITION is entered.

4.1.28 Run Conditions
LNSSRH is changed to QUADPT.
Control Mode
GO is entered and the run control cycle is set to 10.

4.1.29 Linear Search Display
The 100th linear search starting from the beginning of the run is occurring. G is entered in the cursor mode. No change in the display cycle is requested.

4.1.30 Control Mode
The requested control cycle (i.e., 10) has been completed. The summary table for the current run is requested.
4.1.31 Summary Table

Notice the initialization after the 5th iteration due to the change in MINTEC. $\bar{Q}$ is used to return to the control mode.

4.1.32 Control Mode

A function plot against number of iterations for the three runs is requested.

4.1.33 Function Plot

$\bar{Q}$ is used to return to the control mode.

4.1.34 Control Mode

A function plot against number of function evaluations for the second and third runs is requested.

4.1.35 Function Plot

The X-cursor is positioned and the co-ordinates at the requested point are displayed. Notice the difference in the number of function evaluations between the two runs. $\bar{Q}$ is used to return to the control mode.

4.1.36 Control Mode

A new run is requested.

Fourth Run

4.1.37 Priming Guess Specification

REPEAT is specified to initiate the run from the previous priming guess.
4.1.38 Run Conditions
MINTEC is changed to GRAD and G$PAR is set to 9.

Control Mode
INITIAL is used to enable a change in the initialization and stopping parameters.

4.1.39 Initialization and Stopping Parameters
The values displayed are the default values for the case where the analytic gradient is being used. MXFE is set to 1000 evaluations.

Control Mode
GO is specified. Run control and LS display cycles are set to 100 and 400 respectively.

4.1.40 Control Mode
The execution stops after 100 iterations. GO is specified again to continue without change.

4.1.41 Control Mode
The run stops due to the function evaluation limit. INITIAL is specified to change this limit.

4.1.42 Initialization and Stopping Parameters
MXFE is reset to its default value (NO LIMIT) and MXITER is set to 200 iterations.

Control Mode
GO is used to continue. Run control and LS display cycles are not changed.

4.1.43 Control Mode
The run stops after the 200 iterations specified by MXITER are completed. Function plot versus number
of iterations is requested.

4.1.44 Function Plot
The plot is for the whole run (200 iterations). \( \bar{Q} \) is used to return to the control mode.

4.1.45 Control Mode
A plot for the last 30 iterations is requested.

4.1.46 Function Plot
(SUB-TABLE) indicates that the data for this plot is obtained from the sub-table. Note that the log scale is not used in this case. \( \bar{Q} \) is used to return to the control mode. Note also the obvious linear convergence rate.

4.1.47 Control Mode
A display of the summary table is requested starting from the 70th row.

4.1.48 Table Display
30 runs of the table are displayed starting from row number 70. Notice the effect of the table updating. \( \bar{Q} \) is used to return to the control mode.

4.1.49 Control Mode
A sub-table display is requested.

4.1.50 Table Display
The sub-table is displayed (all of the last 30 iterations). \( \bar{Q} \) is used to return to the control mode.

4.1.51 Control Mode
A function plot against number of function evaluations is requested for runs 3 and 4.
4.1.52 Function Plot

\( \overline{Q} \) is used to return to the control mode.

4.1.53 Control Mode

"END JOB" is used to terminate the job.

4.3 Comparative Studies

The plotting features provided in INTEROPT make it especially straightforward to conduct experiments for testing and comparing either of the following:

(a) the different minimization technique and the different linear searches (note that this applies equally to user-provided algorithms)

(b) the sensitivity of any of the techniques to the auxiliary parameters: e.g. RESTRT, G$\overline{P}A\$, and LS$\overline{P}A$.

The results of such experiments are often of great value and several studies of this type have appeared in the literature, (e.g. [10]). Some experiments of this type are given in examples 4.3(a) and 4.3(b) to illustrate this capability.

Example 4.2(a)

In this example the Powell Function [Appendix A3.2] is used. The job submission included the subroutine GRAD. Fig. 4.2.1 shows the priming guess. Fig. 4.2.2 through Fig. 4.2.4 give the results for 11 runs (in four groups), first as plots of function value against number of function evaluations and then as plots of function value against number of iterations.
The following summary provides some comments on these plots.

**Figure**    **Comments**

4.2.2  Run numbers 1, 2 and 3 - Comparison of linear searches:

Each run uses one of the 3 available linear searches.

The default values of all other run conditions are used (MINTEC=D-F-P, G$PAR=AG, LS$PAR=2). 30 iterations are allowed.

4.2.3  Run numbers 4, 5 and 6 - Comparison of minimization techniques:

Each run uses a different MINTEC value (D-F-P, F-R and SOREN respectively). FBNACI is used and G$PAR and LS$PAR are assigned their default values (AG and 2 respectively). 30 iterations are allowed.

4.2.4  Run numbers 7 and 8 - Experiment with RESTRT:

RESTRT is assigned the values 0 and 5 respectively in runs 7 and 8. F-R is used and all other run conditions are given default values. 800 function evaluations were allowed.

4.2.5  Run numbers 9, 10 and 11 - Experiment with LS$PAR:

LS$PAR is assigned the values 2, 3 and 4 respectively for the runs 9, 10 and 11. POWELL is used and all other run conditions are assigned their default values. 1000 function evaluations are allowed.

**Example 4.2(b)**

In this example the Multi-dimensional Banana Function [Appendix A3.3] is used. The job submission included the subroutine
GRAD. Fig. 4.2.6 gives the priming guess at the beginning of the job. Fig. 4.2.7 and Fig. 4.2.8 give the results of 7 runs (in two groups) as function plots (as in example 4.2(a)). A summary of these figures is given below.

<table>
<thead>
<tr>
<th>Figure</th>
<th>Comments</th>
</tr>
</thead>
</table>
| 4.2.7  | Run numbers 1, 2, 3, and 4 - Experiment with G$\text{SPAR}$:  
          G$\text{SPAR}$ is assigned the values AG, 7, 9 and 11 respectively in these four runs. D-F-P is used with default values assigned to all other parameters. 40 iterations were allowed*. |
| 4.2.8  | Run numbers 5, 6 and 7 - Experiment with LS$\text{SPAR}$:  
          LS$\text{SPAR}$ is assigned the values 2, 3 and 4 respectively, in these three runs. SOREN is used with default values assigned to all other parameters. 500 function evaluations are allowed. |

* Run number 2 was actually terminated due to successive failures in iterations 38 and 39 (i.e. iteration number 39 was an unsuccessful iteration after an initialization).
4.4 Interactive Linear Search

As outlined in section 2.12, INTEROPT permits the user to participate in (i.e. interact with) the solution of the linear search problem. The various features in this respect are demonstrated in the context of examples 4.3(a) and 4.3(b) which are described below. Example 4.3(a) corresponds to one run, while example 4.3(b) includes two distinct runs.

Example 4.3(a)

In this example, the Wood Function is used and the job submission included the subroutine GRAD. The following run conditions were used: MINTEC = D-F-P, GS$PAR = AG, LIN$SRH = QUADPT, LS$PAR = 2.

<table>
<thead>
<tr>
<th>Figure</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.1</td>
<td>The first linear search in the run and as noted, the IIOU is (0,2). The character 'E' is entered and the system responds with the message 'EXTENSION, ENTER A OR B'. Then the value 9 is entered indicating that the new interval to be considered is (0,9). The character 'D' is entered to draw the function over the new interval.</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Two points are displayed by adjusting the vertical cursor and entering the character 'P' in each case. The new interval to be considered is passed to the system by moving the vertical cursor to the previous two points and entering the characters 'A' and 'B' for the left and the right ends of the interval respectively. A request for a display of the function within the new interval is</td>
</tr>
</tbody>
</table>
given by entering the character 'D'.

4.3.3 The same procedures as in fig. 4.3.2.

4.3.4 The same procedures as in fig. 4.3.2.

4.3.5 In this figure, it can be noted that the function has a quadratic shape near the minimum, although the Wood Function itself is not quadratic. Many algorithms for function minimization are based on this fact. Now the character 'G' is entered to allow the system to perform the entire linear search and ignore the previous steps. A new value of 3 is specified for the LS display cycle.

4.3.6 A point is displayed by adjusting the vertical cursor and entering the character 'P'. This point is also used as the left end of the interval to be considered by entering 'A'. The interval is also extended by entering 'E' and then specifying 1 as the value of the right end (B). Also a request for a new graph is given (the character 'D').

4.3.7 The interval under consideration has now been extended on the left-hand-side to 0.

4.3.8 The X-coordinates of two points are displayed by adjusting the vertical cursor at these two points and entering the character X in each case. These same two points are then specified as the limits of the interval (A,B) as described above.

4.3.9 The function is plotted within the new interval. 'G' is entered and the value of the LS display cycle is changed to 6.
4.3.10 The display is that for linear search number 10. A point is displayed (via the character 'P') and it is then used as the right-end of the next interval (via the character 'B').

4.3.11 The same procedures as in fig. 4.3.10.

4.3.12 Again, this figure, the quadratic shape of the function near the minimum is apparent. 'G' is entered and 1 is assigned to the LS display cycle.

4.3.13 For this linear search, a point is displayed and used as the limit A. The other limit, B, is moved to 0.1. A new display is then requested.

4.3.14 No change is specified in the LS display cycle (simply 0).

Example 4.3(b)

Run number 2 is started from the same priming guess. The following run conditions were used: MINTEC = P-R, G$PAR = AG, LN$SRH = GOLDEN, LS$PAR = 2.

<table>
<thead>
<tr>
<th>Figure</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.15</td>
<td>'P' is used to display the coordinates of a point. 'B' is entered without altering the cursor position (right-end). 'D' is entered to display the curve for the new interval.</td>
</tr>
<tr>
<td>4.3.16</td>
<td>'P' is entered to display the coordinates of a point. 'M' is entered without altering the cursor position, to pass this point as a solution of the linear search.</td>
</tr>
</tbody>
</table>
4.3.17 'E' is entered to extend the graph and 0.4 is specified for B. 'P' is entered to display the coordinates of a point. 'A' is entered without altering the cursor position (left-limit). 'D' is entered.

4.3.18 'P' is used to display the coordinates of a point. 'B' is entered without altering the cursor position. 'R' is entered to pass the interval (0.194, 0.296) to the system as the IIOU.

4.3.19 The display is for linear search number 6. The coordinates of two points are displayed (via 'P') and these two points are specified as A and B, then 'D' is entered.

4.3.20 'P' is used to display the coordinates of a point, then 'M' is entered without altering the cursor position. The displayed point then becomes the user's specification for the solution of the linear search problem.

4.3.21 The coordinates of two points are displayed (via 'P') and these two points are assigned to A and B (the limits of the IOU). 'R' is used to pass these limits to the system.

4.3.22 As before, two points are displayed and they are assigned to A and B. 'D' is entered to draw the curve in the new interval.

4.3.23 'P' is used to display the coordinates of a point and then 'M' is entered without altering the cursor position. Thus the displayed becomes the user's specification for the solution of the linear search problem.
Here, this run is stopped and a new run (run number 3) is initiated starting from the same priming guess and under the same run conditions as run number 2. In this case, however, the run is allowed to proceed without any user interaction with the linear searches. The two runs (2 and 3 respectively) are compared in fig. 4.3.24 and fig. 4.3.25 where the function value is plotted against number of function evaluations and against number of iterations respectively.

Both runs 2 and 4 proceed through 10 iterations (see fig. 4.3.25). Note however (in fig. 4.3.24) that run 3, which takes place without user interaction on the linear searches, attains a slightly lower function value in fewer function evaluations, than does run 2.
4.5 An Example of Constrained Minimization

As described in Chapter 2, INTEROPT can accommodate constrained minimization problems through the use of the Interior Penalty Function Method [3]. The example given in this section describes a penalty function run with INTEROPT where the FUNCTION subprogram F is set up as described in section 2.14. The problem under consideration (Optimal Fuel Allocation in Power Plants) is taken from [11] and has the following form:

Minimize \[ C = x_3f_1 + x_4g_1, \]
subject to the following constraints:

(a) \[ 18 \leq x_1 \leq 30 \]
(b) \[ x_2 = 50 - x_1 \]
(c) \[ 14 \leq x_2 \leq 25 \]
(d) \[ 0 \leq x_3 \leq 1 \]
(e) \[ 0 \leq x_4 \leq 1 \]
(f) \[ \text{BFG} = (1-x_3)f_2 + (1-x_4)g_2 \leq 10.0 \]

where \[ f_1 = 1.4609 + .15186x_1 + .00145x_1^2, \]
\[ f_2 = 1.5742 + .1631x_1 + .001358x_1^2, \]
\[ g_1 = .8008 + .2031x_2 + .000916x_2^2, \]
and \[ g_2 = .7266 + .2256x_2 + .000778x_2^2. \]

The equality constraint (b), makes it possible to eliminate the variable \( x_1 \) from the problem formulation. Correspondingly, the inequality constraints become:

(a') \[ 20 \leq x_2 \leq 25 \]
(b') \[ 0 \leq x_3 \leq 1 \]
(c') \[ 0 \leq x_4 \leq 1 \]
(d') \[ \text{BFG} = (1-x_3)f_2 + (1-x_4)g_2 \leq 10.0 \]
where \( f_1 = 1.4609 + .15186(50-x_2) + .00145(50-x_2)^2 \)

\( f_2 = 1.5742 + .1631(50-x_2) + .001358(50-x_2)^2 \)

and \( c_1 \) and \( g_2 \) are as above.

The problem therefore has three independent variables, \( x_2, x_3 \) and \( x_4 \). The priming guess used in the computation was (22.5, .5, .5).

Figures 4.4.1 through 4.4.41 are displays selected from the terminal session. The following comments provide a synopsis of these figures.

1. Figures 4.4.1 through 4.4.29 provide details of the first, the second and the third runs while figures 4.4.30 through 4.4.41 provide details of the ninth and the tenth runs. The displays generated during the runs in between these are quite similar and hence are not included.

2. The value of the coefficient \( R \) associated with the "penalty term" is displayed and reduced in value at the beginning of each run in accord with the underlying concept of the penalty function approach.

3. The values 10, 1, 0.1, ..., \( 10^{-8} \) were assigned successively to \( R \) in the course of the ten runs.

4. The effect of the penalty term on the value of the penalty function in the forbidden region is clear from the linear search displays (see, fig. 4.4.3, fig. 4.4.14, fig. 4.4.15, fig. 4.4.33, fig. 4.4.35). The value of the function increases rapidly at the boundary of the feasible area and remains very high in the forbidden area.
WHAT IS YOUR PRIMING GUESS?
-3, -1, -3, -1
THE NEW PRIMING GUESS IS
-0.000005 01
-0.103425 01
-0.000005 01
-0.103425 01
FUNC VALUE = 0.121925 05

DO YOU WISH TO RESPECIFY THE PRIMING GUESS? NO

RUN CONDITIONS
MINTEO = D-F-P / GSPAR = AGAD
LINESHR = QUADET / LSPAR = 2
CHANGE?
F-8.4

RUN CONDITIONS
MINTEO = F-R / GSPAR = AGAD
LINESHR = QUADET / LSPAR = 4

WHAT NOW? GO

RUN CONTROL CYCLE = 5
LS DISPLAY CYCLE = 5
CHANGE?

Fig. 4.1.1

LINEAR SACH 8, 6
FUNC+ FAX 0.7330 62 MIN 0.3120 68
LOU+ FROM 0.0 TO 0.6355 68
LS DISPLAY CYCLE = 5, CHANGE? 6

Fig. 4.1.2

O OF ITERATIONS = 5
O OF FUNC EVAL. = 66
THE GRADIENT
0.320250 00
0.105410 00
0.170270 00
0.052200 00
FUNC VALUE = 0.313670 02
THE GRADIENT
0.135070 02
-0.479777 02
-0.135470 02
0.249990 01
GRAD NORM = 0.613527 02

WHAT NOW? OUTPUT

Fig. 4.1.3
PRINTER OUTPUT FOR EACH ITERATION

ENTER THE DESIRED OPTION IDENTIFICATION NUMBER
0 NO OUTPUT
1 ITER & FUNC & ARG VALUES, 8 OF FUNC EVAL
2 SAVE AS 1 PLUS GRAD & LINEAR SEARCH INFO.

WHAT NOW? Go

RUN CONTROL CYCLE = 8
LS DISPLAY CYCLE = 8
CHANGE?

8 OF ITERATIONS = 10
8 OF FUNC EVAL = 112
THE ARGCENT
0.02756D 00
0.32079D 00
0.23081D 00
0.25745D 00
FUNC VALUE = 0.29215D 00

THE GRADIENT
0.02530D 02
-0.06372D 02
-0.13159D 02
-0.12772D 02
GRAD NORM = 0.12992D 03

WHAT NOW? PLOT

FUNCTION VALUE / # OF FUNCTION EVALUATIONS

LOG SCALE
FUNC MAX 0.1919D 05 MIN 0.2922D 03
EVAL FROM 8 TO 112

RUN = 1
CPAR = AG
QUADFT
LSPAR = 4

WHAT NOW? CONDITION

Fig. 4.1.5

Fig. 4.1.6

Fig. 4.1.7

Fig. 4.1.8
RUN CONDITIONS

MINTEC = F-R  GSPAR = AGRAD
LNSSRH = QUADFT  LSPAR = 4

WHAT NOW? GO

RUN CONTROL CYCLE = 5
LS DISPLAY CYCLE = 5
CHANGE 10

FUNCTION VALUE/OF ITERATIONS
LOG SCALE
FUNC: MAX 0.1722D+03 MIN 0.1722D+02
ITER: FROM 0 TO 20
RUN 1 D-F-P
GSPAR = AG
QUADFT
LSPAR = 4

WHAT NOW? PLOT/ITER

ITER 10
RUN = 0.1722D+02

Fig. 4.1.9

Fig. 4.1.10
WHAT NOW? NEW RUN

CURRENT PRIMING GUESS IS
-0.200000 01
-0.100000 01
-0.032000 01
-0.010000 01
FUNC VALUE = 0.191220 05

CURRENT ARGUMENT IS
0.100000 01
0.100000 01
0.000000 00
0.000000 00
FUNC VALUE = 0.208850-09

REPEAT, CONTINUE OR NEW? REPEAT

RUN CONDITIONS

MINTEC = D-F-P GSPAR = AGRAD

LNSR = QUADFT LSPAR = 4

CHANGE?

WHAT NOW? OUTPUT.2
WHAT NOW? INITIAL

INITIALIZATION & STOPPING PARAMETERS

1 RESTRT = NOT APPL
2 FAIL = INIT
3 FXITER = NO LIMIT
4 FDPE = NO LIMIT
5 ESPAR = 14

CHANGE?

1.10 3.29

INITIALIZATION & STOPPING PARAMETERS

1 RESTRT = 10
2 FAIL = INIT
3 FXITER = 20
4 FDPE = NO LIMIT
5 ESPAR = 14

WHAT NOW? GO

RUN CONTROL CYCLE = 8
LS DISPLAY CYCLE = 8
CHANGE? 25.65
FUNCTION VALUE/ # OF ITERATIONS
$\log$ SCALE
Funct: Max 0.1919D-05 Min 0.3688D-09
Iter: From 0 To 19

THE GRADIENT
0.19565D 01
-0.23579D 01
-0.53154D 01
GRAD NOA = 0.150329D 01

THE REQUESTED NUMBER OF ITERATIONS HAVE NOW BEEN COMPLETED
(ITER = 20)

WHAT NOW? PLOT/ITER.1.2

WHAT NOW? TABLE.1

<table>
<thead>
<tr>
<th>RUNS</th>
<th>SUBTR</th>
<th>NVAL</th>
<th>FUNC VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.101520D 05</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.124520D 03</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.355260D 02</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>0.146202D 02</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>6</td>
<td>0.310370D 03</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>0.311670D 03</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
<td>0.279510D 02</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>6</td>
<td>0.032370D 02</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>0.057333D 02</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>12</td>
<td>0.023310D 02</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>12</td>
<td>0.023310D 02</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>12</td>
<td>0.170320D 04</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>14</td>
<td>0.032333D 01</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>152</td>
<td>0.037340D 00</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>161</td>
<td>0.012340D 00</td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>172</td>
<td>0.452250D 01</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
<td>104</td>
<td>0.652230D 03</td>
</tr>
<tr>
<td>17</td>
<td>7</td>
<td>185</td>
<td>0.370420D 02</td>
</tr>
<tr>
<td>18</td>
<td>8</td>
<td>217</td>
<td>0.037642D 02</td>
</tr>
<tr>
<td>19</td>
<td>8</td>
<td>233</td>
<td>0.300320D 09</td>
</tr>
</tbody>
</table>

Fig. 4.1.17

Fig. 4.1.18

Fig. 4.1.19

Fig. 4.1.20
RUN CONDITIONS

MINTEC = 240
LINSSRH = FINACI / LSIPAR = 4

CHANGE? POWELL

RUN CONDITIONS

MINTEC = POWELL
LINSSRH = FINACI / LSIPAR = 4

WHAT NOW? GO

RUN CONTROL CYCLE = 8
LS DISPLAY CYCLE = 50

WHAT NOW? OUTPUT.0

WHAT NOW? CONDITION

3 OF ITERATIONS = 10
3 OF CUR-ITER. = 5 ( 1 )
1 OF FLHC EVAL. = 1054
THC ARCHITECT
=0.10008 01
=0.81409 00
=0.07637 09
FLHC VALUE = 0.78855 01

WHAT NOW? GO

RUN CONDITIONS

MINTEC = POWELL
LINSSRH = QUADFT / LSIPAR = 4

CHANGE? QUADFT

RUN CONDITIONS

MINTEC = POWELL
LINSSRH = QUADFT / LSIPAR = 4

WHAT NOW? GO

RUN CONTROL CYCLE = 8
LS DISPLAY CYCLE = 50

CHANGE? 10
### Table 1.

<table>
<thead>
<tr>
<th>RUN SUMMARY</th>
<th>ITER</th>
<th>SUBITR</th>
<th>#EVAL</th>
<th>FUNC VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.402350   02</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>144</td>
<td>0.340360 02</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>250</td>
<td>0.261360 02</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>325</td>
<td>0.113250 02</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>483</td>
<td>0.172420 02</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>572</td>
<td>0.114320 02</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>716</td>
<td>0.837720 01</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>226</td>
<td>0.204320 01</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>227</td>
<td>0.722440 01</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>1001</td>
<td>0.765450 01</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>1054</td>
<td>0.768450 01</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>1182</td>
<td>0.784700 01</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>1147</td>
<td>0.715320 01</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>1158</td>
<td>0.795160 01</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>1226</td>
<td>0.762230 01</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>1659</td>
<td>0.761140 01</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>11</td>
<td>1328</td>
<td>0.774200 01</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>12</td>
<td>1335</td>
<td>0.768240 01</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>13</td>
<td>1207</td>
<td>0.764250 01</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>14</td>
<td>1401</td>
<td>0.742230 01</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>15</td>
<td>1420</td>
<td>0.746230 01</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 4.1.30**

**Fig. 4.1.32**
CURRENT PRIMING GUESS IS
-0.325000 01
-0.100000 01
-0.360000 01
-0.100000 01
FUNC VALUE = -0.191920 05

CURRENT ARGUMENT IS
-0.149760 01
-0.152440 01
-0.981180 01
-0.154820 01
FUNC VALUE = 0.748230 01

REPEAT, CONTINUE OR NEW? REPEAT.

RUN CONDITIONS

MINTEC = POWELL
LMNRMH = QUADF / LSPAR = 4
CHANGE?
GRAD? NO

RUN CONDITIONS

MINTEC = GRAD / GSPAR = 9
LMNRMH = QUADF / LSPAR = 4

WHAT NOW? INITIAL

INITIALIZATION & STOPPING PARAMETERS

1 RESTART = NOT APPL
2 FAIL = STOP
3 MIXTER = NO LIMIT
4 FIXE = NO LIMIT
5 ESPAR = 14

CHANGE? 4.1000

INITIALIZATION & STOPPING PARAMETERS

1 RESTART = NOT APPL
2 FAIL = STOP
3 MIXTER = NO LIMIT
4 FIXE = 1030
5 ESPAR = 14

WHAT NOW? GO

RUN CONTROL CYCLE = 5
LS DISPLAY CYCLE = 5
CHANGE? 100.400

6 OF ITERATIONS = 100
6 OF FUNC EVAL = 832
THE AGGRSM 0.113350 01
0.103270 01
0.842060 00
0.713310 00
FUNC VALUE = 0.742760 01

THE GRADIENT
0.256320 00
0.256320 01
-0.166720 00
-0.281540 00
GRAD NORM = 0.424440 00
WHAT NOW? GO

RUN CONTROL CYCLE = 100
LS DISPLAY CYCLE = 400
CHANGE?
$\text{# OF ITERATIONS} = 121$
$\text{# OF FUNC EVAL.} = 1005$

\text{FUNC VALUE} = 0.697810-01

\text{THE GRADIENT}
-0.364000 00
-0.379000 00
-0.131000 00
-0.015200 00

\text{GRAD NORM} = 0.583420 00

\text{THE REQUESTED # OF FUNCTION EVALUATIONS HAVE NOW BEEN COMPLETED}
(RNFE= 1000)

WHAT NOW? INITIAL

\text{INITIALIZATION & STOPPING PARAMETERS}

1 \text{ RESTART = NOT APPL}
2 \text{ FAIL = STOP}
3 \text{ MINITER = NO LIMIT}
4 \text{ RNFE = 1000}
5 \text{ ESPAR = 14}

\text{CHANGE? 4}

3.200

\text{INITIALIZATION & STOPPING PARAMETERS}

1 \text{ RESTART = NOT APPL}
2 \text{ FAIL = STOP}
3 \text{ MINITER = 200}
4 \text{ RNFE = NO LIMIT}
5 \text{ ESPAR = 14}

WHAT NOW? GO

\text{RUN CONTROL CYCLE = 100}
\text{LS DISPLAY CYCLE = 400}

\text{CHANGE?}

\text{FUNCTION VALUE/# OF ITERATIONS}

\text{BLOG SCALE}

\text{FMIN:} \text{ MAX 0.1910D 00 MIN 0.6558D-01}
\text{ITER: FROM 0 TO 200}

\text{RUN = 4}
\text{GRAD}
\text{ESPAR = 9}
\text{QUADFT}
\text{LSAPR = 4}

\text{WHAT NOW? PLOT/ITER}
### Function Value vs. Iterations

**RUN # 4**
- **GRAD**
- **GSPAR** = 9
- **QUADFT**
- **LSPAR** = 4

**Fig. 4.1.45**

### Run Summary

<table>
<thead>
<tr>
<th>Iter</th>
<th>Subtr</th>
<th>Iter</th>
<th>Func Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>133</td>
<td>133</td>
<td>1149</td>
<td>0.654702E-01</td>
</tr>
<tr>
<td>143</td>
<td>143</td>
<td>1166</td>
<td>0.663101E-01</td>
</tr>
<tr>
<td>146</td>
<td>146</td>
<td>1183</td>
<td>0.671501E-01</td>
</tr>
<tr>
<td>144</td>
<td>144</td>
<td>1200</td>
<td>0.684301E-01</td>
</tr>
<tr>
<td>146</td>
<td>146</td>
<td>1217</td>
<td>0.695701E-01</td>
</tr>
<tr>
<td>148</td>
<td>148</td>
<td>1234</td>
<td>0.706801E-01</td>
</tr>
<tr>
<td>150</td>
<td>150</td>
<td>1251</td>
<td>0.717701E-01</td>
</tr>
<tr>
<td>152</td>
<td>152</td>
<td>1268</td>
<td>0.728501E-01</td>
</tr>
<tr>
<td>154</td>
<td>154</td>
<td>1285</td>
<td>0.739201E-01</td>
</tr>
<tr>
<td>156</td>
<td>156</td>
<td>1302</td>
<td>0.749601E-01</td>
</tr>
<tr>
<td>158</td>
<td>158</td>
<td>1319</td>
<td>0.759801E-01</td>
</tr>
<tr>
<td>160</td>
<td>160</td>
<td>1336</td>
<td>0.769901E-01</td>
</tr>
<tr>
<td>162</td>
<td>162</td>
<td>1353</td>
<td>0.779801E-01</td>
</tr>
<tr>
<td>164</td>
<td>164</td>
<td>1370</td>
<td>0.789601E-01</td>
</tr>
<tr>
<td>166</td>
<td>166</td>
<td>1387</td>
<td>0.799301E-01</td>
</tr>
<tr>
<td>168</td>
<td>168</td>
<td>1404</td>
<td>0.809001E-01</td>
</tr>
<tr>
<td>170</td>
<td>170</td>
<td>1421</td>
<td>0.818701E-01</td>
</tr>
<tr>
<td>172</td>
<td>172</td>
<td>1438</td>
<td>0.828301E-01</td>
</tr>
<tr>
<td>174</td>
<td>174</td>
<td>1455</td>
<td>0.837901E-01</td>
</tr>
<tr>
<td>176</td>
<td>176</td>
<td>1472</td>
<td>0.847401E-01</td>
</tr>
<tr>
<td>178</td>
<td>178</td>
<td>1489</td>
<td>0.856901E-01</td>
</tr>
<tr>
<td>180</td>
<td>180</td>
<td>1506</td>
<td>0.866301E-01</td>
</tr>
<tr>
<td>182</td>
<td>182</td>
<td>1523</td>
<td>0.875701E-01</td>
</tr>
<tr>
<td>184</td>
<td>184</td>
<td>1540</td>
<td>0.885101E-01</td>
</tr>
<tr>
<td>186</td>
<td>186</td>
<td>1557</td>
<td>0.894501E-01</td>
</tr>
<tr>
<td>188</td>
<td>188</td>
<td>1574</td>
<td>0.903901E-01</td>
</tr>
<tr>
<td>190</td>
<td>190</td>
<td>1591</td>
<td>0.913301E-01</td>
</tr>
<tr>
<td>192</td>
<td>192</td>
<td>1608</td>
<td>0.922701E-01</td>
</tr>
<tr>
<td>194</td>
<td>194</td>
<td>1625</td>
<td>0.932101E-01</td>
</tr>
<tr>
<td>196</td>
<td>196</td>
<td>1642</td>
<td>0.941601E-01</td>
</tr>
</tbody>
</table>

**Fig. 4.1.46**

**Fig. 4.1.47**
What is your priming guess?
-1.2.1.1 2.1.1.2
The new priming guess is
-0.120000 01
-0.120200 01
-0.120300 01
-0.120500 01
Func value = 0.18164D 04
Do you wish to respecify the priming guess? No
**RUN CONDITIONS**

- **MINTEC**: D-F-P / GIPAR = 0
- **LHSVAR**: QUADF / LSIPAR = B

**Fig. 4.4.1**

**Fig. 4.4.2**

**LINEAR SEARCH**: 1

- **FUNC**: MAX 0.10000 05 MIN 0.9726D 03
- **IOU**: FROM 0.9 TO 0.99990 01

- **LS DISPLAY CYCLE**: 1. CHANGE?

- **# OF ITERATIONS**: 1
- **# OF FUNC EVAL.**: 17

**THE ARGUMENT**
- 0.22500D 02
- 0.42877D 00
- 0.00000D 00

- **FUNC VALUE**: 0.9726D 03

- **THE GRADIENT**
  - 0.16456D-01
  - 0.27432D-02
  - 0.44029D-02

- **GRAD NORM**: 0.17267D-01

**WHAT NOW?**: GO

- **RUN CONTROL CYCLE**: 1
- **LS DISPLAY CYCLE**: 1

**CHANGE?**
**RUN CONDITIONS**

<table>
<thead>
<tr>
<th>MINTC</th>
<th>D-F-P</th>
<th>CSEP = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRSRN</td>
<td>QADFT</td>
<td>LSSEP = 2</td>
</tr>
</tbody>
</table>

**WHAT NOW?**

- GO

**RUN CONTROL CYCLE**

- 5

**LS DISPLAY CYCLE**

- 5

**CHANGE**

- 4

---

**LINEAR SRCH**

- 2

<table>
<thead>
<tr>
<th>FUNC. MAX</th>
<th>FUNC. Min</th>
<th>MIN.</th>
<th>MAX.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9726D-03</td>
<td>0.3957D-03</td>
<td>0.9726D-03</td>
<td></td>
</tr>
</tbody>
</table>

**LS DISPLAY CYCLE**

- 1

**CHANGE?**

- 1

---

**R = 0.160D 03**

**Current Forcing Guess**

<table>
<thead>
<tr>
<th>Guess</th>
<th>0.225500D 00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.52000D 00</td>
<td></td>
</tr>
<tr>
<td>0.52000D 00</td>
<td></td>
</tr>
</tbody>
</table>

**FUNC. VALUE**

- 0.15302D 03

**Current Argument**

<table>
<thead>
<tr>
<th>Argument</th>
<th>0.22492D 03</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.42627D 00</td>
<td></td>
</tr>
<tr>
<td>0.45920D 04</td>
<td></td>
</tr>
</tbody>
</table>

**FUNC. VALUE**

- 0.15302D 03

**WHAT NOW?**

- NEW RUN

---

**Fig. 4.4.5**

**Fig. 4.4.6**
# OF ITERATIONS = 4
# OF FUNC EVAL. = 43
THE ARGUMENT
0.22455D 02
0.33234D 00
FIND VALUE = 0.14635D 02
THE GRADIENT
0.55809D 04
0.56317D 02
0.46715D 02
GRAD NORM = 0.54787D 02
WHAT NOT? TABLE

RUN SUMMARY
ITER   SUBITR  #EVAL  FUNC VALUE
1      1       13      0.14635D 02
2      2       26      0.14635D 02
3      3       34      0.14635D 02
4      4       43      0.14635D 02

WHAT NOT? NEW RUN

R = 0.100D 01 CHANGE? .1
CURRENT PRIMING GUESS IS
0.22500D 02
0.56309D 00
0.50000D 00
FIND VALUE = 0.71937D 01
CURRENT ARGUMENT IS
0.22455D 02
0.37836D 00
0.33234D 00
FIND VALUE = 0.57838D 01
REPEAT, CONTINUE OR NEW? CONTINUE

Fig. 4.4.9

Fig. 4.4.10
RUN CONDITIONS

MINVAR = D-F-P
LSVAR = G
LSVSR = QAQT
LSVSR = O
CHANGE?

WHAT NOW? GO

RUN CONTROL CYCLE = 5
LS DISPLAY CYCLE = 5
CHANGE? 2.8

Fig. 4.4.13

Fig. 4.4.14

LINEAR SRCH # 2
FUNC MAX 0.1000D 05 MIN 0.4822D 01
IQU = FROM 0.0 TO 0.2000D 00

LS DISPLAY CYCLE = 6
CHANGE?

LINEAR SRCH # 2
FUNC MAX 0.0422D 01 MIN 0.4800D 01
IQU = FROM 0.0 TO 0.8000D 01
LS DISPLAY CYCLE = 6
CHANGE?

# OF ITERATIONS = 2
# OF FUNC EVAL = 25
THE ARGUMENT
0.2203D 02
0.2703D 00
0.8704D 00
THE GRADIENT
0.1167D-01
0.1612D 01
0.1222D 01
GRAD NORM = 0.8023D 01
WHAT NOW? GO

RUN CONTROL CYCLE = 2
LS DISPLAY CYCLE = 2
CHANGE?

K = 0.0637D-01
# OF ITERATIONS = 4
# OF FUNC EVAL. = 30
THE ARGUMENT
0.224470 02
0.263430 03
0.266160 00
FUNC VALUE = 0.489750 01

THE GRADIENT
0.152510 01
-0.621000 01
-0.316560 00
GRAD NORM = 0.744230 01

WHAT NOW? GO
RUN CONTROL CYCLE = 2
LS DISPLAY CYCLE = 2
CHANGE?
RUN CURVARY
ITER SUBITR #EVAL FUNC VALUE
0 0 1 4 0.48834D+01
1 1 16 0.48834D+01
2 2 26 0.48834D+01
3 3 32 0.48834D+01
4 4 32 0.48834D+01
5 5 68 0.48834D+01
6 6 59 0.48834D+01

RUN & 3
RUN CONTROL CYCLE = 8
D-F-P
LS DISPLAY CYCLE = 8
LSISPAR = 0
QUADFT = 1.1
LSISPAR = 8

WHAT NOW? GO
R = 0.180D 00  CHANGE? .01
CURRENT PRINTING GUESS IS
0.25000D 02
0.50000D 00
0.50000D 00
FUNC VALUE = 0.63749D 01
CURRENT ARGUMENT IS
0.215540 02
0.261570 00
0.257270 00
FUNC VALUE = 0.35724D 01
REPEAT, CONTINUE OR NEW? CONTINUE

# OF ITERATIONS = 7
# OF FUNC EVAL. = 78
THE ARGUMENT
0.200360 02
0.780380 00
0.572220 00
FUNC VALUE = 0.38547D 01
THE GRADIENT
0.593549 01
0.601299 01
0.430559 01
GRAD NORM = 0.73974D 01
WHAT NOW? NEW RUN

Fig. 4.4.29

R = 0.160D-05  CHANGE? .1D-6
CURRENT PRINTING GUESS IS
0.22500D 02
0.50000D 00
0.50000D 00
FUNC VALUE = 0.62839D 01
CURRENT ARGUMENT IS
0.200310 02
0.783330 00
0.573200 00
FUNC VALUE = 0.39531D 01
REPEAT, CONTINUE OR NEW? CONTINUE

RUN CONDITIONS
MINTEC = D-F-P / GSPAR = 9
LHSRAN = QUADFT / LSPAR = 8
CHANGE?
WHAT NOW? GO
RUN CONTROL CYCLE = 5
LS DISPLAY CYCLE = 5
CHANGE? 4.4

Fig. 4.4.30

Fig. 4.4.31
LSE: 0.000D-06  MIN  0.3053D-01
   FROM  0.0  TO  0.1402D-01
LS DISPLA CYC LE  =  2; CHANGE7

LINEAR SRCH 2
LSE: 0.3053D-01  MIN  0.3053D-01
   FROM  0.0  TO  0.4824D-02
LS DISPLA CYC LE  =  4; CHANGE7

2 OF ITERATIONS  =  4
2 OF FUNK EVAL.  =  47
THE ARKUFNT
0.20023D-02
0.97923D-03
0.592225D 00
FUNK WILLE  =  0.385229D 01
THE GRADIENT
-0.840317D-02
-0.520233D 00
-0.250710D 00
GRAD NORM  =  0.836883D 09
STEP NOUT  =  00
RUN CONTROL CYC LE  =  4
LS DISPLA CYC LE  =  4
CHANGE7  1.1

Fig. 4.4.33

Fig. 4.4.34

Fig. 4.4.35

Fig. 4.4.36
Fig. 4.4.37

R = 0.100D-06 CHANGE? .1D-7

CURRENT PRIMING GUESS IS
0.235530 02
0.500000 99
0.628292 01
FLNQ VALUE = 0.628292 01

CURRENT ARGUMENT IS
0.203333 02
0.198433 99
0.522153 99
FLNQ VALUE = 0.385253 01

REPEAT, CONTINUE OR NEW? CONTINUE

Fig. 4.4.38

RUN CONDITIONS

MINTC = D-F-P / GSUP = 9
LSNSH = QUAHFT / LSUP = 8
CHANGE?

WHAT NOW? GO

RUN CONTROL CYCLE = 5
LS DISPLAY CYCLE = 5
CHANGE? .8

Fig. 4.4.40
CHAPTER 5
CONCLUSIONS

The process of locating the minimum of a nonlinear function is highly experimental in nature. Although there exist many powerful algorithms for function minimization, none is guaranteed to converge to a minimum under all circumstances. In addition, there are associated with most algorithms several secondary parameters and tasks which can significantly affect the success of their operation. Therefore, the usefulness of a software package that provides several algorithms together with various mechanisms for handling the secondary tasks, is apparent.

When such a package functions in a batch computing environment, the user is obliged to accept significant delays between minimization experiments. Since the determination of an acceptable solution may require many experiments involving technique and parameter changes, the total elapsed time in obtaining a solution may be long. This difficulty can be overcome through the use of an interactive approach in the implementation of the function minimization package. The development of the INTEROPT package was motivated by these considerations.

In addition to the basic features one would expect in a function minimization package (various techniques and alternatives for handling secondary tasks), INTEROPT also provides plotting facilities on the graphics terminal which serves as the user console (a Tektronix 4013 or equivalent). Thus numerical results can be easily presented in the more meaningful form of graphs. These, in turn,
help the user to rapidly evaluate the progress of a minimization trial.

User control over both the minimization process and the graphical output are available at the terminal during the running of the program. The response of the system to user commands rarely exceeds two or three seconds.

The package is designed to conveniently allow the testing and evaluations of new minimization algorithms as well as new linear search algorithms. This is achieved through the provision of subroutine calls to "dummy" subroutines (YOURSA, YOURSB, YOURSL) which are supplied by the user according to his own specifications.

The modular design of the package allows extensions and alterations to be made easily. This applies both to the inclusion of new algorithms as well as the addition of new interactive or graphical capabilities. Once a new algorithm has been evaluated and found to be efficient, it can be added permanently to the package with a few minor modifications. Similarly, if the need arises to extend the currently available graphical output or the present set of user commands, only a few changes in the package will be required.

In its present form, INTEROPT can accommodate functions having at most eight parameters (dimensions). In order to handle problems of higher dimension some re-dimensioning of arrays must be done in several of the subroutines in the package.

A particularly noteworthy feature of INTEROPT is its capability of directly handling the interior penalty function method for constrained minimization problems. With this approach, the constrained problem is treated as a sequence of unconstrained subproblems each
of which is concerned with minimizing a "penalty function" that is constructed from the original criterion function and the constraints. Each sub-problem must be solved to a "reasonable" degree of accuracy at which point a scalar parameter in the penalty function is altered to form a new sub-problem. Its solution begins at the last argument value determined for the preceding problem.

In a batch computing environment the user has very limited control over this procedure since he has no means of determining when the minimum of a particular sub-problem has been adequately reached. In an interactive environment, however, the user is able to determine from the results that are displayed on the screen, whether to go on or to stop the sub-program solution. Thus in an interactive environment such as provided by INTEROPT, only the amount of computing that is actually required, is committed to each sub-problem solution (as determined by the judgement of the user).

The present INTEROPT package is initiated by the submission of a deck of cards which contains the job control cards, a function sub-program for the criterion function and, optionally, a subroutine for the gradient of the function. Hence, INTEROPT is not strictly speaking, totally interactive. This is due to the fact that the IBM/360 model 65 on which INTEROPT presently operates, provides very limited interactive facilities. A significant improvement of the package would be to introduce a means of handling the whole job interactively. This could be achieved, for example, through the use of the YTS system on IBM/360.

Another enhancement would be the more extensive use of disk files to store the summary tables. This would reduce the amount of
core memory required by INTEROPT and improve the capability for storing more information about the various runs. However, this possibility should be studied carefully since it would lead to a significant increase in I/O activity which could counter any advantage that might be anticipated.
APPENDIX A1

THE MULTIDIMENSIONAL MINIMIZATION TECHNIQUES IN INTEROPT

This appendix outlines the various multidimensional minimization techniques implemented in INTEROPT. The following notation is used in the description of the algorithms:

- $x^k$ is the argument of the criterion function at the end of the $k^{th}$ iteration (an $n$-vector), ($x^0$ is the priming or starting point).
- $f(x^k)$ is the value of the criterion function at $x^k$.
- $f_x(x^k)$ is the value of the function gradient at $x^k$, $g_k = f_x(x^k)$.
- $s_k$ is the current search direction (an $n$-vector).
- $\alpha^*$ is the optimal step along the direction $s_k$, i.e. it is the minimizing argument of the function $F(\alpha)=f(x^k+\alpha s_k)$.

The notation $\alpha^* = \min_{\alpha} f(x^k, \alpha s_k)$ is used to denote the scalar $\alpha^*$ for which $f(x^k+\alpha^* s_k) = \min_{\alpha} f(x^k+\alpha s_k)$.

Al.1 The Steepest Descent Algorithm (GRAD)

1. Choose $x^0$ (the priming guess) and set $k=0$.
2. Let $s_k = -f_x(x^k)$.
3. Find $\alpha^* = \min_{\alpha} f(x^k, \alpha s_k)$
4. $x^{k+1} = x^k + \alpha^* s_k$.
5. Replace $k$ with $k+1$ and go to step 2.

Al.2 Conjugate Gradient Algorithm of Fletcher and Reeves (F-R) [12]

1. Choose $x^0$ (the priming guess), let $s_0 = g_0 = -f_x(x^0)$ and set $k=0$.
2. Let $x^{k+1} = x^k + \alpha^* s_k$ where $\alpha^* = \min_{\alpha} f(x^k, \alpha s_k)$.
3. Set $s_{k+1} = -g_{k+1} + \beta_k s_k$

   where $\beta_k = \frac{g_k^T g_{k+1}}{g_k^T g_k}$

4. Replace $k$ with $k+1$ and go to step 2.

Al.3 Conjugate Gradient Algorithm of Polack and Ribiere (P-R) [13]

Same as the Fletcher-Reeves algorithm, except that:

$$\beta_k = \frac{g_k^T y_k}{s_k^T y_k}$$

where $y_k = g_{k+1} - g_k$

Al.4 Conjugate Gradient Algorithm of Sorenson (SOREN) [14]

Same as the Fletcher-Reeves algorithm, except that:

$$\beta_k = \frac{g_k^T y_k}{s_k^T y_k}$$

where $y_k = g_{k+1} - g_k$

Al.5 The Davidon-Fletcher-Powell Algorithm (D-F-P) [15]

1. Choose $x^0$ (the priming guess), let $H_0 = I$ (the n x n identity matrix),

   $s_0 = -g_0 = -f(x^0)$ and set $k = 0$.

2. Let $x^{k+1} = x^k + \alpha_k s_k$

   where $s_k = -H_k g_k$ and $\alpha_k = \min_{\alpha} <f, x^k, s_k>$

3. Set $A_k = \frac{s_k^T s_k}{s_k^T y_k}$

   $B_k = \frac{(y_k') (H_k')^T}{y_k' H_k y_k}$
\[ H_{k+1} = H_k + A_k + B_k \]

4. Replace \( k \) with \( k+1 \) and go to step 2.

**Al.6  Powell's Algorithm (POWELL) [16]**

Let the \( k^{th} \) iteration begin at \( x^0 \)

1. For \( j=1,2,...,n \) find \( \alpha_j = \min <f, x^{j-1}\xi_j > \) and let \( x^j = x^{j-1} + \alpha_j \xi_j \).

2. For \( i=1,2,...,(n-1) \), replace \( \xi_i \) with \( \xi_{i+1} \).

3. Let \( \xi_n = (x^n - x^0) / |x^n - x^0| \)

4. Find \( \alpha_n = \min <f, x^n, \xi_n > \) and replace \( x^0 \) with \( (x^n + \alpha_n \xi_n) \).

**Note** For the first iteration, \( \xi_j = e_j = \) the \( j^{th} \) column of the \( n \times n \)
identity matrix.

**Al.7  Zangwill's Method (ZANG) [17]**

**First Iteration**

Let \( e_j \), \( j=1,2,...,n \) be the columns of the \( n \times n \) identity matrix
and let \( \xi_j \), \( j=1,2,...,n \) be a normalized set of \( n \) linearly independent directions.

Let \( x^0 \) be the initial guess, and find

\[ \alpha_n = \min <f, x^0, \xi_n > \]

Replace \( x^0 \) with \( (x^0 + \alpha_n \xi_n) \).

Set \( t=1 \).

**Subsequent Iterations**

Let the iteration begin at \( x^0 \).

1. Find \( \alpha = \min <f, x^0, e_t > \) and replace \( t \) with \( t+1 \) if \( t<n \) or 1 if \( t=n \).

2. If \( \alpha \neq 0 \), replace \( x^0 \) with \( (x^0 + \alpha e_t) \) and go to step 3. Otherwise go to step 1 (Note: if step 1 is repeated \( n \) times in succession,
then we can assume that \(x^0\) is the optimum.

3. For \(j=1,2,\ldots,n\), calculate \(a_j = \min_\delta \langle f, x^{j-1}, \xi_j \rangle\) and let
\[x^j = x^{j-1} + a_j \xi_j.\]

4. For \(j=1,2,\ldots,(n-1)\), replace \(\xi_j\) with \(\xi_{j+1}\).

5. Let \(\xi_n = \frac{x^n - x^0}{|x^n - x^0|}\) and find \(a = \min_\delta \langle f, x^n, \xi_n \rangle\)

6. Replace \(x^0\) with \(x^n + a \xi_n\) and go to step 1.

Al.8 The Extended Sequential Search Method (ESQ)

1. Choose \(x^0\) (the priming guess).

2. Set \(j=1\)

3. Let \(x^j = x^{j-1} + a_s j\)

   where \(a_s = \min_\delta \langle f, x^{j-1}, s_j \rangle\)

   and \(s_j = \begin{cases} e_j & \text{if } j \leq n \\ \frac{x^n - x^0}{|x^n - x^0|} & \text{if } j = n+1 \end{cases}\)

4. If \(j = n+1\), replace \(x^0\) with \(x^{n+1}\) and go to step 2; otherwise replace
   \(j\) with \(j+1\) and go to step 3.
The minimization algorithms within INTEROPT require a mechanism for solving the following one-dimensional minimization (or linear search) problem.

Find the scalar $\alpha^*$ such that

$$F(\alpha^*) = \min_{\alpha} F(\alpha)$$

where: $F(\alpha) = f(x^k + \alpha s^k)$, and $x^k$ and $s^k$ are given n-vectors.

There are three algorithms for handling this problem in INTEROPT. This appendix briefly outlines their operation.

### A2.1 The Quadratic Fit (QUADFT) [18]

If three points $\alpha_1$, $\alpha_2$ and $\alpha_3$ can be selected to span the minimum value of $F(\alpha)$, the function can be approximated with the quadratic polynomial

$$p(\alpha) = a_0 + a_1 \alpha + a_2 \alpha^2$$

The approximate position of the minimum of $F$ is then derived by setting $\frac{dp}{d\alpha} = 0$, giving

$$\alpha_{\min} = \frac{\alpha_1}{2a_2}$$

If the function values at the three selected points are $F_1$, $F_2$ and $F_3$ respectively, then upon solving for $a_1$, $a_2$ and $a_3$ we obtain

$$\alpha_{\min} = \frac{1}{2} \left( \frac{\alpha_2^2 - \alpha_3^2}{a_2 - a_3} F_1 + \frac{(\alpha_2 - \alpha_3)^2}{a_2^2 - a_3^2} F_2 + \frac{(\alpha_2 - \alpha_3)^2}{(a_2^2 - a_3^2)} F_3 \right)$$

This predicted minimum is then used with the two best of the three points $\alpha_1$, $\alpha_2$ and $\alpha_3$ for a second quadratic approximation and the process is repeated.
In INTEROPT, the subroutine I$IOU selects the three points that span the minimum of $F(\alpha)$. Then, the subroutine QUADFT performs successive quadratic approximations from three points provided by the predicted minimum at the previous iteration and the two best of the three points used to predict it. The process stops when the distance between two predicted minima in two successive iterations is less than or equal to $10^{-m}$ where $m$ is the parameter LS$\$PAR. The best point obtained in this procedure is taken to be approximation for the minimizing argument of $F(\alpha)$.

A2.2 The Fibonacci Search (FBNACI) [19]

The iterative steps of the Fibonacci search are implemented in the subroutine FBNACI. An initial interval of uncertainty $(A_1, B_1)$ is passed to the subroutine. The Fibonacci search reduces this interval according to the relation:

$$|A_n - B_n| = \frac{L_1 + F_{n-1} \epsilon}{F_{n+1}}, \quad n \geq 2$$

where $L_1$ is the length of the initial interval of uncertainty $(B_1 - A_1)$, $F_{n-1}$ and $F_{n+1}$ are the $n-1^{\text{th}}$ and $n+1^{\text{th}}$ Fibonacci numbers respectively, and $\epsilon$ is a small positive number set to $10^{-10}$. The reduction of the interval continues until one of the following occurs:

1. The interval becomes less than or equal to $10^{-m}$ where $m$ is the parameter LS$\$PAR.
2. The relative difference between two distinct function values of a particular iteration is less than $10^{-15}$.
3. A maximum of 70 steps are made before either 1 or 2 occurs.

The end points of the final interval together with the midpoint of this interval are used in a quadratic fit procedure (a
call to QUADFT) and the minimizing argument of this quadratic is taken as the final result for the linear search.

A2.3 The Golden Search (GOLDEN) [18]

The Golden Search Algorithm is implemented in the subroutine GOLDEN. First, the initial interval of uncertainty, which contain the minimum of the function, is determined by the subroutine I$IOU (its left and right limits are denoted by $A_1$ and $B_1$ respectively). Then the subroutine GOLDEN applies the iterative steps of the golden section search to reduce this interval according to the relation

$$|A_n - B_n| = K^n * L_1, \quad n \geq 1$$

where $L_1$ is the length of the initial interval of uncertainty $(B_1 - A_1)$, and $K$ is the constant $= (\sqrt{5} - 1)/2$.

The reduction process is stopped when the length of the interval is less than or equal to $10^{-m}$ where $m$ is the parameter LS$\$PAR. The last two points generated by the procedure ($A_n$ and $B_n$), together with their mid-points are then used in a quadratic approximation which yields one additional point; (namely, the minimum of the fitted quadratic). The best of the available points is taken to be the approximation for $a^*$. 
APPENDIX A3

TEST PROBLEMS

The various example problems discussed in Chapter 4, use standard test problems. These are summarized in this appendix.

A3.1 The Wood Function [20]

\[ f(x) = 100(x_2-x_1^2)^2+(1-x_1)^2+90(x_4-x_3^2)^2 + (1-x_3)^2+10.1[(x_2-1)^2+(x_4-1)^2]+19.8(x_2-1)(x_4-1) \]

\[ x^0 = (-3, -1, -3, -1) \]

\[ x^* = (1, 1, 1, 1) \]

By specific design, this function has a non-optional stationary points which can cause inappropriate termination of minimization algorithms. These stationary points are at \((-1.07, 1.16, -0.86, 0.76)\) and \((-0.968, 0.947, -0.9695, 0.951)\).

A3.2 The Powell Function [21]

\[ f(x) = (x_1+10x_2)^2+5(x_3-x_4)^2+(x_2-2x_3)^4+10(x_1-x_4)^4 \]

\[ x^0 = (3, -1, 0, 1) \]

\[ x^* = (0, 0, 0, 0) \]

This function has the distinctive feature of having a Hessian matrix (matrix of second partial derivatives) which is singular at the minimizing argument, \(x^*\). This feature can undermine the effectiveness of minimization algorithms.

* In the discussion, \(x^0\) is used to denote the standard starting (or priming) point and \(x^*\) is used to denote the minimizing argument.
A3.3 The Multi-dimensional Banana Function [22]

\[ f(x) = \sum_{k=1}^{4} 100(x_{k+1} - x_k^2)^2 + (1-x_k)^2 \]

\[ x^0 = (-1.2, 1, -1.2, 1, -1.2) \]

\[ x^* = (1, 1, 1, 1, 1) \]

This function is an extension of the Rosenbrock function [23] whose graph is distinguished by a banana-shaped valley. As such, \( f(x) \) represents a set of intersecting shallow parabolic valleys.
REFERENCES


1972, pp 107-122.


VITA

Name: Bahaa Guirguis

Born: Egypt
November 1, 1949

Education: Bachelor of Science (Honours) 1972
Department of Computer Science
University of Alexandria
Alexandria
Egypt

Enrolled in M.A.Sc. program in Electrical Engineering