DISCUSSION PAPER NO. 232

AN INTEGRATED SYSTEM FOR INTERACTIVE NONLINEAR PROGRAMMING COMPUTATIONS

by

Claude Cohen*†
Barry Robinson*

August 1976

† Graduate School of Management
* Vogelback Computing Center

Northwestern University, Evanston, Illinois 60201

Presented at the IX International Symposium on
Mathematical Programming, 23-27 August 1976,
Budapest, Hungary

Abstract

Existing optimization algorithms are highly formalized and do not allow much human intervention or interaction. This paper deals with the design and development of software for nonlinear optimization using a collection of well-known nonlinear programming codes: SUMT (Sequential Unconstrained Minimization Techniques), GRG (Generalized Reduced Gradient), GPM (Gradient Projection Method), and MCL (Method of Centers Linearized). The system has evolved from "stand-alone" programs to an integrated package which allows the user to define objective function and constraints in a simple FORTRAN subroutine and call one of the algorithms to initiate the solution process. The user can also interact with the process by changing strategies in mid-stream through the use of different algorithms. Because of this algorithm interrupt mechanism, one can study the structure of the problem or observe the solution process (for example, monitoring precision/convergence/number of function or gradient evaluations). As an experimental tool, it may also be used to study the interaction of algorithms and the best matching of an algorithm to a given class of problem.

Table of Contents

1. Introduction

The interactive capability of present-day computer systems can be applied to a wide variety of uses: program preparation, computer-aided instruction, text processing, statistics, graphics, numerical computations, etc. In each case, the value of the interactive use of the computer (as opposed to batch processing) is three-fold. First, the quick response characteristic (if the application is well designed) allows the user to make efficient use of his time. Second, the ability for the user to interpose decisions based on experience or intuition can often accelerate or make feasible the solution of problems of great complexity. This latter procedure can often save computer time as well, despite the overhead of interactive systems, since in the batch environment one is often forced to approach large problems by exhaustive consideration of cases. Third, the use of the computer as an instructional device can facilitate the learning of basic principles in diverse disciplines.

In mathematical programming, perhaps more than any other area of applied mathematics, theoretical problem formulation goes hand in hand with computational feasibility. As algorithmic techniques have evolved and multiplied and as problem sizes have increased, the user has been confronted more and more with choosing the appropriate algorithm, with problems of numerical analysis, and with complex software systems. These algorithms are highly formalized and do not allow much human intervention or interaction.

A number of authors, among them Lasdon [10], Polak [12], and Zangwill [15], have recognized the need for the study of problem manipulation and mixed algorithm strategies of solution. This paper was motivated by and addresses the needs of users as opposed to fulfilling the requirements of algorithm

developers. This is not to imply that the latter are of little importance or interest to the authors, but only to state that they are beyond the scope of the experimental system described here.

To our knowledge, there have been very few mathematical programming software development efforts to integrate several algorithms into a system and allow a user (typically a student) to run problems on the computer with minimum concern for system details and minimum programming effort. MPOS (Multi Purpose Optimization System) developed at Northwestern [2] is such a system. All that is required of the user is the ability to write a set of algebraic expressions and to specify in English-like commands the functions to be performed for solving linear, quadratic, and integer programming problems. Related work in this area is also conducted at other institutions: Argonne National Laboratory (MINPACK project), Stanford University (System Optimization Laboratory), NBER-Cambridge (Computer Research Center), and the National Physical Laboratory in England.

2. The need for interaction in optimization

Given an optimization problem P, we ask the natural and practical question: how should it be solved? In many instances, there exists a "standard" algorithm for the class of problems to which P belongs (e.g., simplex, transportation). Problem P is sometimes restated in an alternative form P' that is apt to be more amenable to solution. Familiar examples are: solving the dual P' of a linear program P, approximating a constrained convex objective function P by a piecewise linear function P', and transforming a constrained optimization problem P into a sequence of related unconstrained problems P'_i, i=1,2,... which are (hopefully) simpler to solve.

In integer and non linear programming (NLP), optimizing is not simply a matter of getting a computer program and using it. Much human judgment is needed to select an appropriate algorithm and even more important to assess the results of that method.

The following situations illustrate commonly encountered problems where the availability of a general and flexible interactive optimization system would clearly be of great help:

a) Suppose P is a linear programming problem with multiple objective functions. For this type of problem, it is often necessary to replace the concept of "optimum" with that of "best compromise". In an interactive environment, the decision maker learns to recognize good solutions and the relative importance of the (competing) objectives. In fact, phases of computation--guiding his exploration of the solution space--alternate with phases of decision. This problem has received a great deal of attention in the literature: Dyer [4,5], Geoffrion et.al. [6], Zionts and Wallenius [16].

b) Suppose P is a NLP problem and the following algorithms are available: Lagrangian, gradient projection, reduced gradient, and penalty function. Choosing one of them, a priori, does not guarantee obtaining a solution if the level sets of the objective function are "banana shaped" or the feasible set has an "unusual" geometrical configuration, e.g., ridges, or saddle points. In a batch processing environment, it is impractical and time consuming for the optimizer to "explore" the objective function and recognize that his successive points lie in a valley. If the user can interact with his program and change strategies through the use of different algorithms, then he makes use of the computer as a learning tool as well as a solution tool.

c) Suppose P is ill-structured and cannot be mathematically stated in the form of an optimization model. Heuristic methods, for example hill-climbing algorithms [1,9] can provide a crude qualitative description of the optimum and it would be highly desirable for a user to incorporate in such methods standard algorithms to provide additional information.

d) Finally, suppose P is an integer program, then many solution techniques are available as in NLP. For example, in a cutting-plane method there are a multitude of cuts which can be added to the problem at any solution point. With an interactive system one can decide which cuts to add and when to add them. If this heuristic approach proves ineffective, then perhaps a branch and bound algorithm, incorporating all the binding cuts added so far, could be attempted.

3. Software Design

As previously mentioned one motivation for developing an integrated NLP package is the difficulty our users encounter when attempting to solve their NLP problems with the existing software at Northwestern. The Vogelback Computing Center maintains four NLP codes - SUMT (Sequential Unconstrained Minimization Technique)[13], GRG (Generalized Reduced Gradient) [8], GPM (Gradient Projection Method) [7], and MCL (Method of Centers - Linearized)[3,11].These codes were developed at four independent sites and modified locally. They provide dramatic evidence of the lack of "standards" in the field of mathematical programming software. The user, particularly the novice, is confronted with a bewildering array of problem formats, constraints and variable bounds requirements, parameter and option choices and computer programming and control card requirements. We suspect that in most cases the choice of an algorithm is influenced more by ease of use and documentation readability than by any theoretical considerations.

Table 1

Comparison of Four NLP Computer Codes

	SUMT	CRG	GPM	MCL
1) Optimization type	min f	max f	max f	max f
2) Constraints	$g_i(x) \geq 0$ $h_i(x) = 0$	$g_i(x) \leq 0$ $h_i(x) = 0$	$c_i \leq g_i(x) \leq d_i$	$g_i(x) \geq 0$
3) Variables	all free or all $x_i \geq 0$	all x_i bounded	x_i free or bounded	all x_i bounded
4) User must supply these FORTRAN routines	RESTNT (f and g_i)	PHIX (f)	Main program	CONT (f, g_i, ∇f, and ∇g_i)
	GRAD1 (∇f and ∇g_i)	CPHI (g_i)	FNCL (f, g_i, ∇f, and ∇g_i)	
	MATRIX (Hessian)	GRADFI (∇f)		
		JACOB (∇g_i)		
5) Data Input	fixed field data cards	fixed field data cards	COMMON blocks in main program	fixed field data cards
6) Restrictions on user-written subroutines	equalities last	equalities last	equalities in any order	no equalities
	return values in parameter lists and COMMON blocks	return all values in COMMON blocks	return all values in parameter lists	return values in COMMON blocks
	compute one function at a time	compute all g_i or all ∇g_i at one time	compute one function and its gradient at a time	compute all functions or gradients at one time
			FNCL only evaluates nonlinear constraints	
7) Numerical Differencing for ∇f, $\nabla^2 f$, or $\nabla^2 g_i$	Yes	No	No	No

Table 1 compares the major features of these packages.

There are several other consequences of these difficulties. The interested user is deterred from trying several different codes on his problem because the use of each new NLP package requires a complete duplication of effort. A programmer would have to code one main program, nine subroutines, and three sets of data cards for each NLP problem. Even using the same algorithm with different options is a tedious process. Research in nonlinear programming (effects of different tolerance checks, choice of starting points, comparison of algorithms on various problem classes) is thus discouraged and probably greatly curtailed. The maintenance of a library of NLP problems also becomes a large and time-consuming undertaking.

Our first step in developing a Multi Purpose Nonlinear Optimization System--MPNOS was, therefore, the definition of a standard NLP problem:

$$
\begin{array}{lll}
\max \text{ (or min)} & f(x) & \\
\text{subject to} & g_i(x) \leq 0 & i=1,\ldots,k \\
& h_i(x) = 0 & i=k+1,..,m \\
\\
& a_j \leq x_j \leq b_j. & j=1,\ldots,n
\end{array}
$$

We then converted the four programs to accept this standard description of the problem in the form of two FORTRAN subroutines, FUNCS for function evaluation and GRADS for gradient evaluation.

The next step was the design and implementation of an interactive computer program that would integrate the available codes into a unified NLP system. We feel that certain capabilities are of prime importance and should override other considerations:

- simple problem description
- quick option selection and problem rerun
- easy and natural switching between algorithms.

There are several other features that we believe should be included in the "ideal" system. Some of these are user-oriented, and others deal with the specifics of implementation. Table 2 lists some of these desirable system attributes.

The actual implementation of MPNOS Version 1.0 incorporates most of the features listed in Table 2. Some of the features are implemented in a rudimentary form and some are left out completely.

The user interfaces with the system by responding to prompts. The use of prompts is in direct contrast with the MPOS System [2] which is command oriented and allows the user to enter the problem in its algebraic form without coding FORTRAN subroutines. However, an analogous system for nonlinear programming would involve the development of a symbolic algebraic manipulator requiring a great deal more programming effort. Instead, MPNOS invokes a text editing program through which the user enters the FORTRAN subroutines, edits them, and writes them to a file for compilation. Next the FORTRAN compiler compiles the FUNCS and GRADS routines. If there are errors, they are listed on the terminal and the user is sent back to the text editor for another try. If there are no errors, control is returned to MPNOS for further prompts. (It should be obvious at this stage that no attempt was made to comply with the portability feature (no. 13). The very real constraints of time and economics dictated that we produce a working system in a short time. In fact, once the system was designed, the implementation phase of the project took approximately one month.)

After compiling the subroutines, MPNOS asks the user to describe the problem (max or min, number of variables, constraints, bounds, starting point, etc.). Many of the prompts require a "YES" or "NO" response. If instead the user

Table 2

Desirable Features of an Interactive NLP System

1. Problem description is simple and is done once for all algorithms.
2. Problems can be quickly rerun after a few option changes.
3. User can switch easily and naturally between algorithms.
4. User interface is prompt driven and uses traditional NLP terminology.
5. Prompting can be overriden by keyword commands; data input is free-form.
6. The novice user can receive "help" in the form of verbose prompts.
7. Automatic derivatives checking (numerical differencing)
8. User can monitor all system parameters (objective function, current point, constraint values, dual variables, gradient values, norms, etc.)
9. Interactive user can manipulate all the parameters available to the batch user.
10. New algorithms should be easily incorporated into the system.
11. Programs written in a manner conducive to interactive processing (i.e., they are broken into segments each requiring small amounts of computer storage).
12. Computer storage allocated dynamically based on problem size.
13. Entire system easily portable between different types of computers.

types "HELP", MPNOS prints up to five lines of information pertaining to the problem.

Once the problem has been described to the user's satisfaction, MPNOS asks the user to select an algorithm, the number of iterations, and a printout interval. Then execution begins on the selected algorithm. Each of the algorithms remains as a stand-alone main program with certain modifications. The data for the problem is transferred back and forth among MPNOS and the algorithm main programs on a temporary disk file. The algorithms have also been modified to print intermediate results on the terminal. (See Section V for a demonstration of the program operation.)

Upon completion of the specified number of iterations (or if an optimum or error occurs), the user can select an option from the following "menu":

1. Continue with this problem
2. Enter a command
3. Start a new problem
4. Stop

Option 1 prompts the user with regard to changing the current point and the variable bounds. The user is again asked to select an algorithm, iteration count, and printout interval and execution resumes.

The user who wishes to continue the problem without making any changes can select option 2. The system then expects to see a command of the form:

SUMT, 10, 3

which means make ten iterations using SUMT and print intermediate results every third iteration. MPNOS immediately proceeds to the execution of the algorithm.

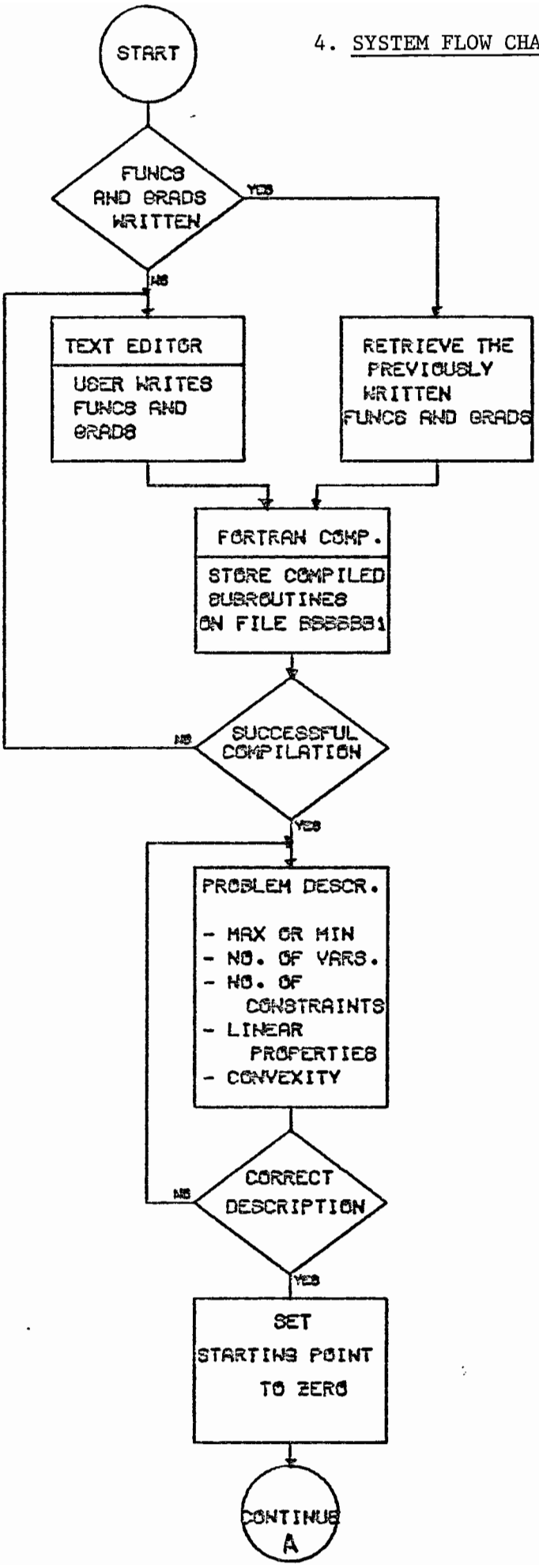
Option 3 returns the user to the text editor for entry of a new set of FUNCS and GRADS routines.

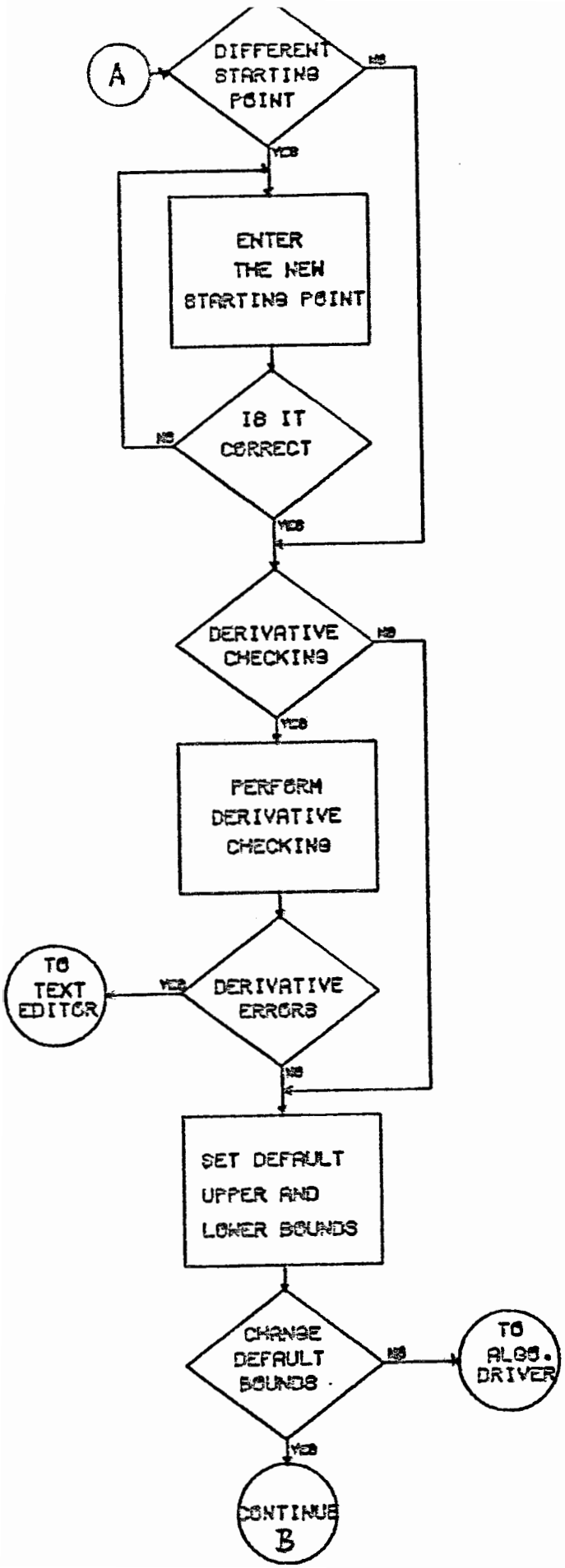
In this initial system, the user is only asked about a few algorithm parameters. For example, the GRG user is asked to set the tolerances for constraint feasibility and two stopping criteria. All other parameters available to the batch user are set to their default values. Future versions of MPNOS will allow the user to manipulate all the algorithm parameters.

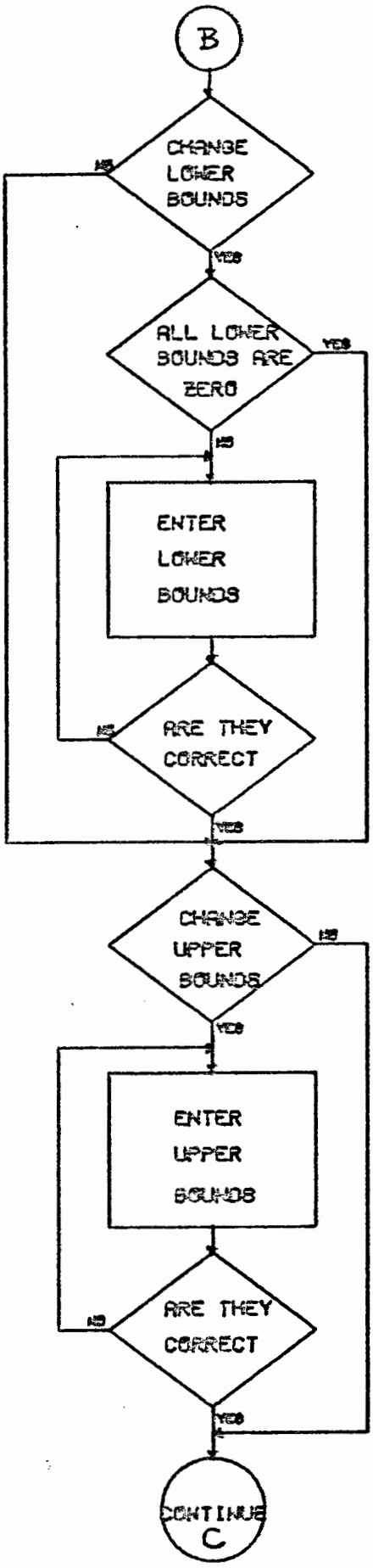
The intermediate printout at the terminal consists of the iteration number, the objective function value, and the current point. In order to fulfill the requirements of feature number 8, all four algorithms produce their standard output which the user can send to the batch printer after the MPNOS run. An obvious benefit of this batch output is that it provides the user with a log of the MPNOS run.

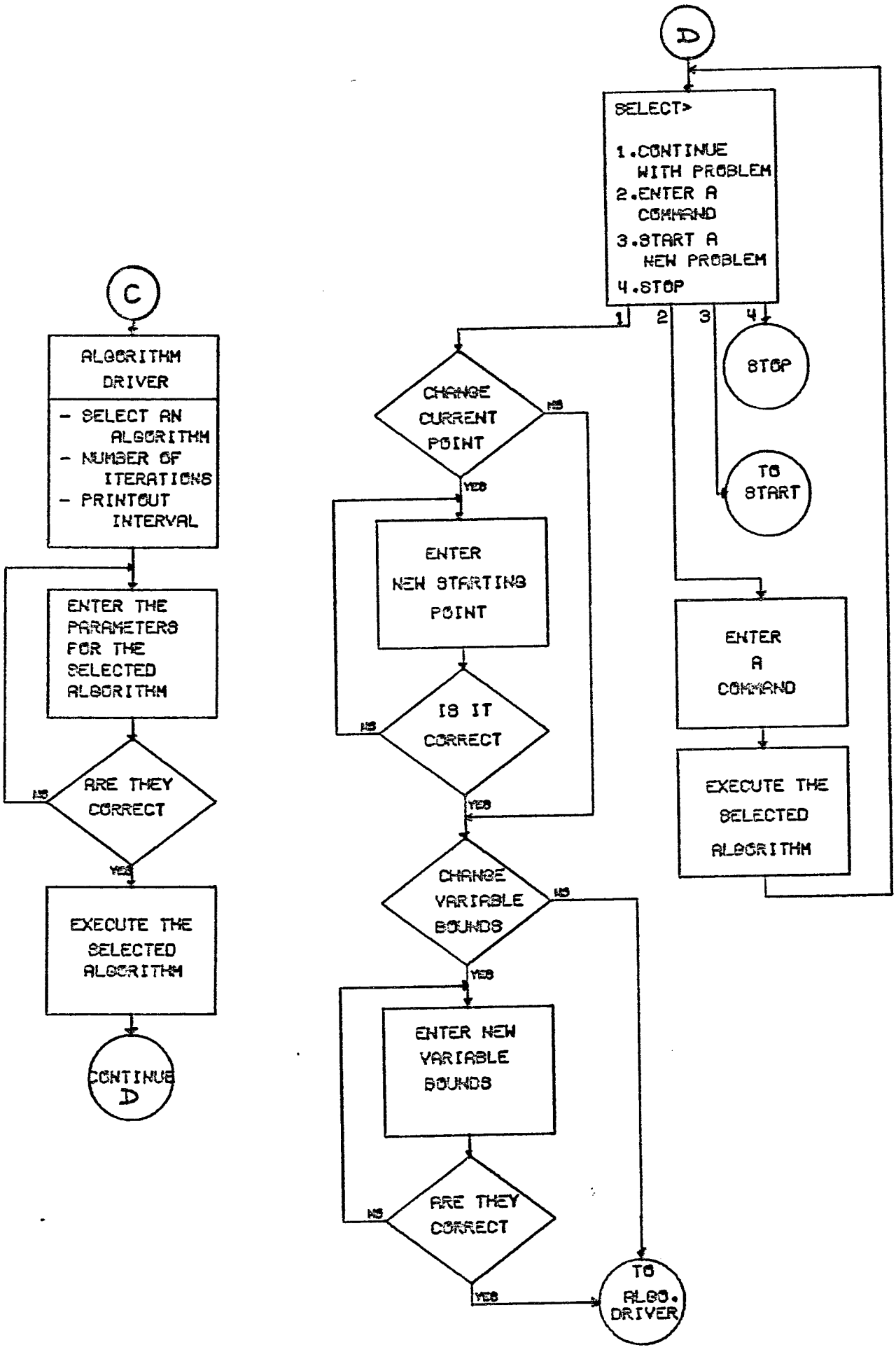
The implementation described here and illustrated in Section 5 is a subset of the final version of MPNOS as we envision it. The system flow chart follows in the next section.

4. SYSTEM FLOW CHART




A
DIFFERENT STARTING POINT
NO
YES
ENTER THE NEW STARTING POINT
IS IT CORRECT
NO
YES
DERIVATIVE CHECKING
NO
YES
PERFORM DERIVATIVE CHECKING
TO TEXT EDITOR
YES
DERIVATIVE ERRORS
NO
SET DEFAULT UPPER AND LOWER BOUNDS
CHANGE DEFAULT BOUNDS
NO
TO ALGS. DRIVER
YES
CONTINUE B


B
CHANGE LOWER BOUNDS
NO
YES
ALL LOWER BOUNDS ARE ZERO
YES
NO
ENTER LOWER BOUNDS
ARE THEY CORRECT
NO
YES
CHANGE UPPER BOUNDS
NO
YES
ENTER UPPER BOUNDS
ARE THEY CORRECT
NO
YES
CONTINUE C


C
ALGORITHM DRIVER
- SELECT AN ALGORITHM
- NUMBER OF ITERATIONS
- PRINTOUT INTERVAL
ENTER THE PARAMETERS FOR THE SELECTED ALGORITHM
ARE THEY CORRECT
NO
YES
EXECUTE THE SELECTED ALGORITHM
CONTINUE D
D
SELECT>
1.CONTINUE WITH PROBLEM
2.ENTER A COMMAND
3.START A NEW PROBLEM
4.STOP
1
2
3
4
STOP
TO START
CHANGE CURRENT POINT
NO
YES
ENTER NEW STARTING POINT
IS IT CORRECT
NO
YES
ENTER A COMMAND
EXECUTE THE SELECTED ALGORITHM
CHANGE VARIABLE BOUNDS
NO
YES
ENTER NEW VARIABLE BOUNDS
ARE THEY CORRECT
NO
YES
TO ALGO. DRIVER

```
 WELCOME TO MPNOS (VERSION 1.0)
 MULTI-PURPOSE NONLINEAR OPTIMIZATION SYSTEM

  07/27/76        14.25.06.

 HAVE YOU WRITTEN THE FUNCS AND GRADS ROUTINES?   (Y OR N)
? Y

 ARE THEY EDITOR SAVE FILES?   (Y OR N)
? Y
EDITOR READY
? OLD FUNCS PW=FFF
 CREATED 07/15/76 10.03.11
 30 LINES READ
? LIST
10        SUBROUTINE FUNCS(I,X,F)
20        DIMENSION X(2)
30        J=I+1
40        GO TO (100,200,300,400),J
50    100 F=X(1)**2-X(2)**2
60        RETURN
70    200 F=-X(1)-2.*X(2)**2+7.
80        RETURN
90    300 F=X(1)*X(2)-9.
100       RETURN
110   400 F=X(1)-X(2)-5.
120       RETURN
130       END
140       SUBROUTINE GRADS(I,X,G)
150       DIMENSION X(2),G(2)
160       J=I+1
170       GO TO (100,200,300,400),J
180   100 G(1)=2.*X(1)
190       G(2)=-2.*X(2)
200       RETURN
(S,C OR RETURN)?
210   200 G(1)=-1.
220       G(2)=-4.*X(2)
230       RETURN
240   300 G(1)=X(2)
250       G(2)=X(1)
260       RETURN
270   400 G(1)=1.
280       G(2)=-1.
290       RETURN
300       END
? STOP

 SUBROUTINES FUNCS AND GRADS ARE BEING COMPILED.

 IS THIS A MAX OR MIN PROBLEM?
? MAX

 ENTER THE NUMBER OF VARIABLES IN THE PROBLEM.
? 2

 HOW MANY INEQUALITY CONSTRAINTS?
? 2

 HOW MANY EQUALITY CONSTRAINTS?
? 1

 IS THE OBJECTIVE FUNCTION LINEAR?   (Y OR N)
? N

 ARE ALL THE CONSTRAINTS LINEAR?   (Y OR N)
? N

 IS THE CONSTRAINT SET CONVEX?   (Y OR N)
? HELP

 THE CONSTRAINT SET WILL BE CONVEX IF EACH INDIVIDUAL CONSTRAINT
 IS CONCAVE. MAXIMIZING A CONCAVE OBJECTIVE FUNCTION OVER A CONVEX
 CONSTRAINT SET WILL RESULT IN A GLOBAL MAXIMUM. MINIMIZING A CONVEX
 FUNCTION WILL RESULT IN A GLOBAL MINIMUM. IF YOU ARE NOT SURE
 WHETHER THE CONSTRAINT SET IS CONVEX, ANSWER 'N'.

 IS THE CONSTRAINT SET CONVEX?   (Y OR N)
? N

 CHECK THE PROBLEM DESCRIPTION.
          THIS IS A MAX PROBLEM WITH  2 VARIABLES,
          2 INEQUALITY CONSTRAINTS AND  1 EQUALITY CONSTRAINT .
          THE OBJECTIVE FUNCTION IS NONLINEAR.
          THE CONSTRAINT SET IS NONLINEAR AND NONCONVEX.

 IS THE DESCRIPTION CORRECT?   (Y OR N)
? Y
```

retrieve previously written text

list the text

Problem 1

$$\text{Max } f(x) = x_1^2 - x_2^2$$

$$\text{s.t.} \quad g_1(x) = -x_1 - 2x_2^2 + 7 \leq 0$$

$$g_2(x) = x_1 x_2 - 9 \leq 0$$

$$g_3(x) = x_1 - x_2 - 5 = 0$$

See page 19 for a plot of the feasible region and level curves of f.

Leave text editing and compile subroutines

Note use of 'HELP' command

```
 DO YOU HAVE A STARTING POINT FOR THE PROBLEM?   (Y OR N)
? Y

 ENTER THE NEW STARTING POINT COORDINATES (SEPARATED BY COMMAS)
? -25,-25

 CHECK THE NUMBERS.
    -25.0000   -25.0000

 IS THE DATA CORRECT?   (Y OR N)
? Y

 DO YOU WANT TO CHECK YOUR DERIVATIVES?   (Y OR N)
? N

 THE VARIABLES ARE BOUNDED BELOW BY -1000 AND ABOVE BY +1000.

 DO YOU WANT TO CHANGE THESE BOUNDS?   (Y OR N)
? N

 SELECT AN ALGORITHM (GPM,GRG,MCL,SUMT)
? GRG

 HOW MANY ITERATIONS SHOULD BE MADE USING GRG ?
? 5

 ENTER THE ITERATION PRINTOUT INTERVAL.
? 1

 THE CURRENT TOLERANCES ARE      EPSIL0   .10000E-05
                                 EPSIL1   .10000E-05
                                 EPSIL2   .10000E-05

 DO YOU WANT TO CHANGE THE TOLERANCES?   (Y OR N)
? HELP

 EPSIL0 - ZERO TOLERANCE FOR THE CONSTRAINTS WHICH MUST BE 'LE' TO 0.
 EPSIL1 - ZERO TOLERANCE FOR THE GRADIENT OF THE OBJECTIVE FUNCTION.
 EPSIL2 - RELATIVE CHANGE TOLERANCE FOR THE PROJECTED REDUCED GRADIENT.
 ONLY ENTER THE ITEM TO BE CHANGED. TO CHANGE EPSIL2 TO 1E-8 TYPE IN
          ,,1E-8

 DO YOU WANT TO CHANGE THE TOLERANCES?   (Y OR N)
? N

 EXECUTING GRG

 ITER    OBJ FUNC      CURRENT POINT
   1 -.13419E+06 .13364E+02 .10556E+02
   2 -74509.3992    11.9791     6.9791
   3 -19951.1737     8.4381     3.4381
   4     39.0512     6.4051     1.4051
 OPTIMAL SOLUTION      GRG EXIT CODE=1

 DO YOU WANT TO      1-CONTINUE WITH THIS PROBLEM
                     2-ENTER A COMMAND
                     3-START A NEW PROBLEM
                     4-STOP
? 1
```

free-form input of data

augmented objective function

```
 DO YOU WANT TO CONTINUE FROM THE CURRENT POINT?   (Y OR N)
? N

 ENTER THE NEW STARTING POINT COORDINATES (SEPARATED BY COMMAS)
? -25,30

 CHECK THE NUMBERS.
    -25.0000     30.0000

 IS THE DATA CORRECT?   (Y OR N)
? Y

 DO YOU WANT TO CHANGE ANY VARIABLE LOWER BOUNDS?   (Y OR N)
? N

 DO YOU WANT TO CHANGE ANY VARIABLE UPPER BOUNDS?   (Y OR N)
? N

 SELECT AN ALGORITHM (GPM,GRG,MCL,SUMT)
? SUMT

 HOW MANY ITERATIONS SHOULD BE MADE USING SUMT?
? 3

 ENTER THE ITERATION PRINTOUT INTERVAL.
? 1

 THE CURRENT PARAMETERS ARE      EPSI      .10000E-05
                                 THETA0    .10000E-05
                                 RHO       .59605E-07

 DO YOU WANT TO CHANGE THE PARAMETERS?   (Y OR N)
? HELP

 EPSI   - ZERO TOLERANCE FOR UNCONSTRAINED SUBPROBLEM.
 THETA0 - ZERO TOLERANCE FOR OPTIMALITY.
 RHO    - STARTING VALUE FOR SUBPROBLEM PARAMETER R(K).
 ONLY ENTER THE ITEM TO BE CHANGED. TO CHANGE THETA0 TO 1E-7 TYPE IN
        ,1E-7
 DO YOU WANT TO CHANGE THE PARAMETERS?   (Y OR N)
? Y

 OK. PLEASE ENTER THE PARAMETERS. (SEPARATED BY COMMAS)
? ,,1

 CHECK THE NUMBERS.
    .10000E-05    .10000E-05    .10000E+01

 IS THE DATA CORRECT?   (Y OR N)
? Y

 CURRENT METHOD FOR UNCONSTRAINED MINIMIZATION IS NEWTON-RAPHSON

 DO YOU WANT TO USE A DIFFERENT METHOD?   (Y OR N)
? N

 EXECUTING SUMT

 ITER   OBJ FUNC     CURRENT POINT
   1  .24322E+25 .15596E+13-.23860E+01   <-- 3 iterations using SUMT
   2     15.7761      4.1488     -1.1986
   3     12.3804      3.7430     -1.2764

 DO YOU WANT TO      1-CONTINUE WITH THIS PROBLEM
                     2-ENTER A COMMAND
                     3-START A NEW PROBLEM
                     4-STOP
? 2

 ENTER THE COMMAND.
? GRG                <-- note command mode switch from SUMT to GRG

 EXECUTING GRG

 ITER   OBJ FUNC     CURRENT POINT
   4     -6.5067      3.7427     -1.2762
   5     -3.6262      3.7391     -1.2769
   6     12.1922      3.7192     -1.2808   <-- 3 iterations using GRG
 OPTIMAL SOLUTION      GRG EXIT CODE=1

 DO YOU WANT TO      1-CONTINUE WITH THIS PROBLEM
                     2-ENTER A COMMAND
                     3-START A NEW PROBLEM
                     4-STOP
? 3
```

```
 HAVE YOU WRITTEN THE FUNCS AND GRADS ROUTINES?   (Y OR N)
? Y

 ARE THEY EDITOR SAVE FILES?   (Y OR N)
? Y
EDITOR READY
? OLD TP1 PW=TTT
 CREATED 07/18/76 00.27.00
 25 LINES READ
? L
10        SUBROUTINE FUNCS(I,X,F)
20        DIMENSION X(2)
30        J=I+1
40        GO TO (100,200,300),J
50    100 F=(X(1)-2.)**2+(X(2)-1.)**2
60        RETURN
70    200 F=0.25*X(1)**2+X(2)**2-1.
80        RETURN
90    300 F=X(1)-2.*X(2)+1.
100       RETURN
110       END
120       SUBROUTINE GRADS(I,X,G)
130       DIMENSION X(2),G(2)
140       J=I+1
150       GO TO (100,200,300),J
160   100 G(1)=2.*(X(1)-2.)
170       G(2)=2.*(X(2)-1.)
180       RETURN
190   200 G(1)=0.5*X(1)
200       G(2)=2.*X(2)
(S,C OR RETURN)?
210       RETURN
220   300 G(1)=1.
230       G(2)=-2.
240       RETURN
250       END
? STOP

 SUBROUTINES FUNCS AND GRADS ARE BEING COMPILED.

 IS THIS A MAX OR MIN PROBLEM?
? MIN

 ENTER THE NUMBER OF VARIABLES IN THE PROBLEM.
? 2

 HOW MANY INEQUALITY CONSTRAINTS?
? 1

 HOW MANY EQUALITY CONSTRAINTS?
? 1

 IS THE OBJECTIVE FUNCTION LINEAR?   (Y OR N)
? N

 ARE ALL THE CONSTRAINTS LINEAR?   (Y OR N)
? N

 IS THE CONSTRAINT SET CONVEX?   (Y OR N)
? N

 CHECK THE PROBLEM DESCRIPTION.
          THIS IS A MIN PROBLEM WITH  2 VARIABLES,
          1 INEQUALITY CONSTRAINT  AND  1 EQUALITY CONSTRAINT .
          THE OBJECTIVE FUNCTION IS NONLINEAR.
          THE CONSTRAINT SET IS NONLINEAR AND NONCONVEX.

 IS THE DESCRIPTION CORRECT?   (Y OR N)
? Y

 DO YOU HAVE A STARTING POINT FOR THE PROBLEM?   (Y OR N)
? Y

 ENTER THE NEW STARTING POINT COORDINATES (SEPARATED BY COMMAS)
? 2,2

 CHECK THE NUMBERS.
      2.0000      2.0000

 IS THE DATA CORRECT?   (Y OR N)
? Y

 DO YOU WANT TO CHECK YOUR DERIVATIVES?   (Y OR N)
? N

 THE VARIABLES ARE BOUNDED BELOW BY -1000 AND ABOVE BY +1000.

 DO YOU WANT TO CHANGE THESE BOUNDS?   (Y OR N)
? N

 SELECT AN ALGORITHM (GPM,GRG,MCL,SUMT)
? SUMT

 HOW MANY ITERATIONS SHOULD BE MADE USING SUMT?
? 50

 ENTER THE ITERATION PRINTOUT INTERVAL.
? 1
```

Problem No. 2

$$\min \quad f(x) = (x_1-2)^2 + (x_2-1)^2$$

$$\text{s.t.} \quad g_1(x) = 0.25x_1^2 + 2x_2^2 - 1 \leq 0$$

$$g_2(x) = x_1 - 2x_2 + 1 = 0$$

```
 THE CURRENT PARAMETERS ARE       EPSI      .10000E-05
                                  THETA0    .10000E-05
                                  RHO       .39063E-02

 DO YOU WANT TO CHANGE THE PARAMETERS?   (Y OR N)
? Y

 OK. PLEASE ENTER THE PARAMETERS. (SEPARATED BY COMMAS)
? ,,1

 CHECK THE NUMBERS.
    .10000E-05    .10000E-05    .10000E+01

 IS THE DATA CORRECT?   (Y OR N)
? Y

 CURRENT METHOD FOR UNCONSTRAINED MINIMIZATION IS NEWTON-RAPHSON

 DO YOU WANT TO USE A DIFFERENT METHOD?   (Y OR N)
? N

 EXECUTING SUMT

 ITER   OBJ FUNC     CURRENT POINT
   1      1.4122       .8689        .6355
   2      1.3787       .8309        .8911
   3      1.3924       .8234        .9102
   4      1.3934       .8229        .9114
   5      1.3935       .8229        .9114
 OPTIMAL SOLUTION

 DO YOU WANT TO      1-CONTINUE WITH THIS PROBLEM
                     2-ENTER A COMMAND
                     3-START A NEW PROBLEM
                     4-STOP
? 1

 DO YOU WANT TO CONTINUE FROM THE CURRENT POINT?   (Y OR N)
? N

 ENTER THE NEW STARTING POINT COORDINATES (SEPARATED BY COMMAS)
? 2,2

 CHECK THE NUMBERS.
      2.0000      2.0000

 IS THE DATA CORRECT?   (Y OR N)
? Y

 DO YOU WANT TO CHANGE ANY VARIABLE LOWER BOUNDS?   (Y OR N)
? N

 DO YOU WANT TO CHANGE ANY VARIABLE UPPER BOUNDS?   (Y OR N)
? N

 SELECT AN ALGORITHM (GPM,GRG,MCL,SUMT)
? SUMT

 HOW MANY ITERATIONS SHOULD BE MADE USING SUMT?
? 50

 ENTER THE ITERATION PRINTOUT INTERVAL.
? 1

 THE CURRENT PARAMETERS ARE       EPSI      .10000E-05
                                  THETA0    .10000E-05
                                  RHO       .95367E-06

 DO YOU WANT TO CHANGE THE PARAMETERS?   (Y OR N)
? Y

 OK. PLEASE ENTER THE PARAMETERS. (SEPARATED BY COMMAS)
? 1E-9,1E-9,1

 CHECK THE NUMBERS.
    .10000E-08    .10000E-08    .10000E+01

 IS THE DATA CORRECT?   (Y OR N)
? Y

 CURRENT METHOD FOR UNCONSTRAINED MINIMIZATION IS NEWTON-RAPHSON

 DO YOU WANT TO USE A DIFFERENT METHOD?   (Y OR N)
? N

 EXECUTING SUMT

 ITER   OBJ FUNC     CURRENT POINT
   1      1.4122       .8689        .6355
   2      1.3786       .8309        .8911
   3      1.3924       .8234        .9102
   4      1.3934       .8229        .9114
   5      1.3935       .8229        .9114
   6      1.3935       .8229        .9114
   7      1.3935       .8229        .9114

 NO FEASIBLE SOLUTION.   (OPT SUBROUTINE)

 DO YOU WANT TO      1-CONTINUE WITH THIS PROBLEM
                     2-ENTER A COMMAND
                     3-START A NEW PROBLEM
                     4-STOP
? 4
```

Run with default tolerances

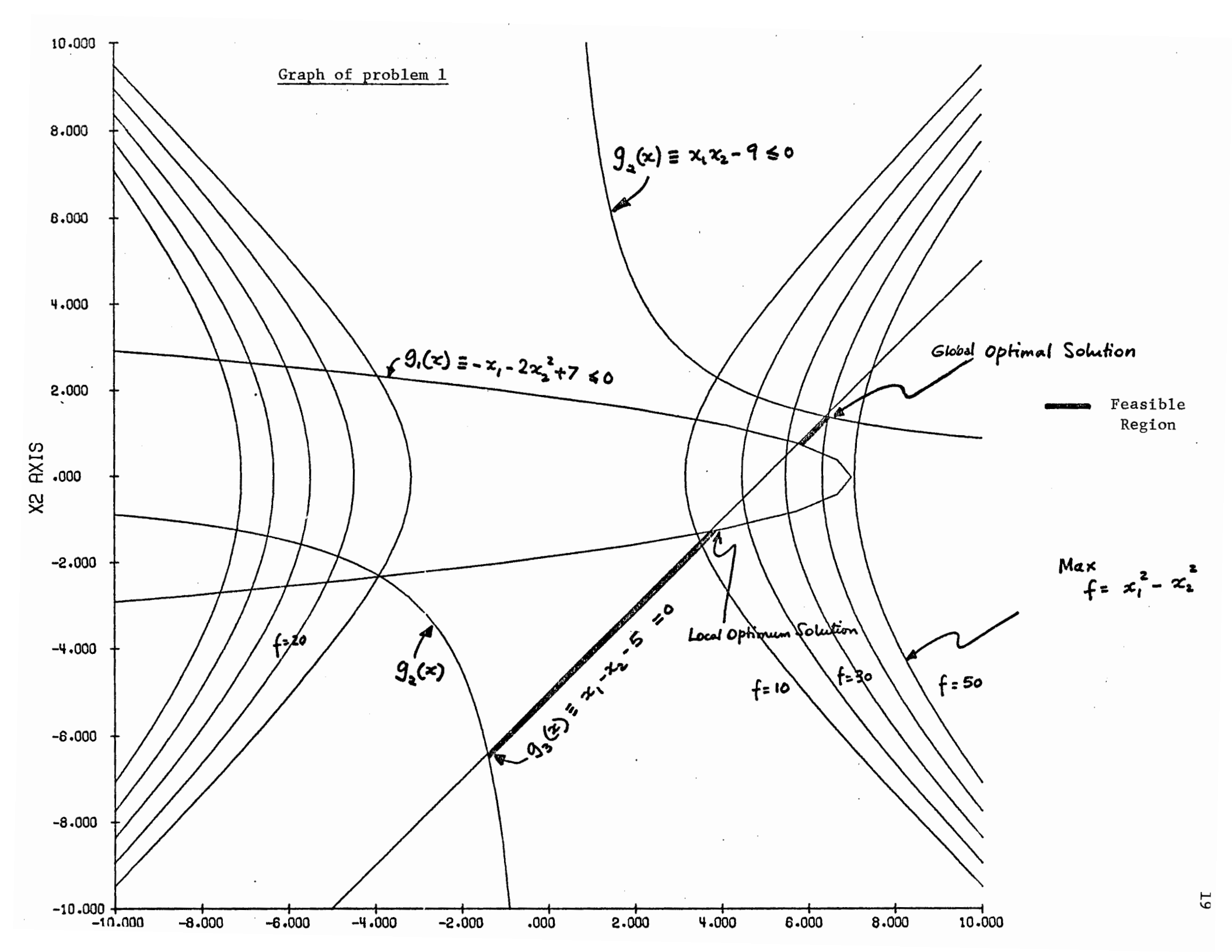
Tighten tolerances & rerun

Send batch output to remote printer

```
CONTROL CARDS:
? DISPOSE,OUTPUT,PR.
? ZE


DAYFILE AS OF  15.24.20.
DISPOSE,OUTPUT,PR.

CONTROL CARDS:
? %LOGOUT
```


Graph of problem 1
$g_2(x) \equiv x_1 x_2 - 9 \leq 0$
$g_1(x) \equiv -x_1 - 2x_2^2 + 7 \leq 0$
Global Optimal Solution
Feasible Region
Max $f = x_1^2 - x_2^2$
Local Optimum Solution
$g_3(x) \equiv x_1 - x_2 - 5 = 0$
$g_2(x)$
f=20
f=10
f=30
f=50
X2 AXIS
10.000
8.000
6.000
4.000
2.000
.000
-2.000
-4.000
-6.000
-8.000
-10.000
-10.000
-8.000
-6.000
-4.000
-2.000
.000
2.000
4.000
6.000
8.000
10.000

6. Evaluation and Possible Extensions

In this paper we described an interactive system for nonlinear optimization. The prototype as implemented on our CDC 6400 computer has not yet been used by students and researchers to enable us to gather information about the design of the interface between the prompting program and the separate algorithms. It will be important to reassess users' needs with respect to this experimental verseion and make the necessary changes/additions in future versions.

We have identified a number of practical and theoretical problems for study in thinking about possible extensions of MPNOS:

a) An archive of test problems which appear in the literature or can be obtained from other institutions or through the SIGMAP problem certification group []4] could be coded only once and run on our system. This would provide very useful and practical information about the relative performance of the four codes.

b) Provide graphics capability for bivariate problems on Tektronix or Imlac CRT's.

c) Further modularization of the individual codes to

- monitor performance of one-dimensional search and step sizes (cubic interpolation, bisections, Golden Section, Fibonacci search,
- evaluate convergence properties of the code based on the norm chosen,
- study search directions, e.g., conjugate, Newton, quasi-Newton.

The need for an interactive programming system for nonlinear optimization is great in both teaching and research. An effort such as MPNOS, properly carried out according to the objectives stated in Table 2, would provide a simple to use and hopefully reliable computing capability. While there is no distributable package at the moment, we would like to establish contacts with users and algorithm implementors to obtain information on existing codes and to discuss interfaces and documentation.

7. References

1. Aronofsky, J., Ed., Progress in Operations Research, Vol. III, John Wiley (1969).
2. Cohen, C., and J. Stein, MPOS - Multi Purpose Optimizations System Version 3 User's Guide, Vogelback Computing Center (1976).
3. Denel, J., "Solution of non linear optimization problems by the method of centers linearized" (in French), Bulletin de la Direction des Etudes et Recherches, Vol. 1, Electricite de France (1973).
4. Dyer, J., "Interactive Goal Programming", Management Science, Vol. 19, No. 1, (1972), pp. 62-70.
5. Dyer, J., "A time sharing computer program for the solution of the multiple criteria problem", Management Science, Vol. 19, No. 12 (1973), pp. 1379-1383.
6. Geoffrion, A., J. Dyer, and A. Feinberg, "Interactive approach for multi criteria optimization", Management Science, Vol. 19, No. 4 (1972), pp. 359-368.
7. GPM/GPMNLC - Gradient Projection Method for Non Linear Programming, Document No. 365, December 1975, Vogelback Computing Center, Northwestern University.
8. GRG - Generalized Reduced Gradient for Non Linear Programming, Document No. 315, February 1974, Vogelback Computing Center, Northwestern University.
9. Huyranne, L. P., and G. M. Weinberg, "Computerized hill climbing game for teaching and research" in Optimization, R. Fletcher, Ed., Academic Press (1970).
10. Lasdon, L., Optimization Theory for Large Scale Systems, McMillan (1970).

11. MCL - Method of Centers (Linearized) for Non Linear Programming, Document No. 396, July 1976, Vogelback Computing Center, Northwestern University.

12. Polak, E., Computational Methods in Optimization, Academic Press (1971).

13. SUMT - Sequential Unconstrained Minimization Techniques for Non Linear Programming, Document No. 200 (A), February 1975, Vogelback Computing Center, Northwestern University.

14. Tomlin, J., "Computational Standards for the Mathematical Programming Society", ACM SIGMAP News-letter, November 1973.

15. Zangwill, W., Nonlinear programming: a unified approach, Prentice Hall (1969).

16. Zionts, S. and J. Wallenius, "An interactive programming method for solving the multiple criteria problem", Management Science, Vol. 22, No. 6 (1976), pp. 652-662.