COMPARATIVE STUDY OF LARGE-SCALE NONLINEAR OPTIMIZATION METHODS

By

SEYED ABOLGHASSEM ALEMZADEH

Bachelor of Science
Teacher Training University
Tehran, Iran
1970

Master of Education
Central State University
Edmond, Oklahoma
1975

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
DOCTOR OF EDUCATION
May, 1987

COMPARATIVE STUDY OF LARGE-SCALE NONLINEAR OPTIMIZATION METHODS

Thesis Approved:

H. G. Burchard

Thesis Adviser

Donald W. Grace

John Wolfe

M S Keener

John J. Gardiner

Norman N. Durham

Dean of the Graduate College

PREFACE

The roots of optimization research can be traced back many decades, when early attempts were made to use a scientific approach in management of organizations; however, according to Hillier and Lieberman [53], the beginning of the activity called operations research has generally been attributed to the military services early in World War II. Because of the war effort, there was urgent need to allocate limited resources to the various military operations and to the activities within each operation in an effective manner. Therefore, the British and the American military management called upon a large number of scientists to apply a scientific approach to dealing with this and other strategic and tactical problems. As Hillier and Lieberman [53] said, these teams of scientists were the first operations research teams.

The success of operations research in the military was a motivation for industry to become interested in this new field. As the industrial revolution following the war was running its course, the problems caused by the increasing complexity and specialization in organizations were again coming to the forefront. It was becoming clear to a growing number of people, including business consultants who had served on or with operations research teams during the war, that these were basically the same problems, but in a different context than had been faced by the military. In this manner, operations research began to move into industry, business, and government.

After the war, many of the scientists who had participated on operations research teams or who had heard about this work were well motivated to pursue research relevant to the field; and as Hillier and Lieberman [53] stated, "Important advancement in the state of art resulted." A prime example is the simplex method for solving linear programming problems developed by George Dantzig [26] in 1947. Many of the standard tools of operations research, such as linear programming, dynamic programming, queuing theory, and inventory theory were relatively well developed before the 1950s. By 1951, operations research had already taken hold in Great Britain and was in the process of doing so in the United States. In addition to the rapid advancement in the theory of operations research, the computer revolution caused a great impetus to the growth of the field.

Linear programming has its own history, but some of the early history of nonlinear optimization is described by Harold W. Kuhn in Nonlinear programming, SIAM-AMS Proceedings, edited by Richard W. Cottle and C. G. Lemke, Vol. IX (1976), "The history of nonlinear optimization can be traced back to the year 1939 when William Karush determined necessary and sufficient conditions for a relative minimum of a function f(x) subject to $(g_1(x),\ldots,g_m(x))^T \geq (0,\ldots,0)^T$." In 1948, Fritz John considered the nonlinear programming problem with inequality constraints. In 1949, Tucker invited Gale and Kuhn to generalize the duality of linear programs to quadratic problems; Gale declined, Kuhn accepted, and a paper developed by correspondence between Stanford and Princeton. In 1960, Rosen, J. B., introduced the gradient projection method for nonlinear programming. In 1961, a Duality Theorem for nonlinear programming was introduced by Philip

Wolfe; through 1964-1969, the Reduced Gradient method of Wolfe was extended to nonlinear programming by Abadie and Carpenter. MINOS-1.0 and MINOS-5.0 Codes for Large-Scale nonlinear problems were expand during 1977-1983, by Murtagh and Saunders.

The concept of optimization is now well-rooted as a principle underlying the analysis of many complex decision or allocation problems. It offers a certain degree of philosophical elegance that is hard to dispute, and it often offers an indispensable degree of operational simplicity. Using this view from optimization, one approaches a complex decision problem involving the selection of values for a number of interrelated variables through focusing attention on an objective function designed to quantify the performance and measure the quality of the decision. This function is maximized or minimized subject to the constraints that may limit the selection of decision variable values.

One obvious measure of the complexity of an optimization problem is its size. Problems can roughly be classified as small-scale problems if they have not more than five variables and constraints, intermediate-scale problems having between five to a hundred variables and constraints, and large-scale problems involving on the order of a thousand variables and constraints.

This paper is an expository study of Large-Scale Nonlinear Optimization Methods. The main emphasis is on the motivation and basic ideas leading to the development of MINOS-1.0 and MINOS-5.0; special attention has been given to the theoretical properties which form their foundation.

I would like to express my deep appreciation to my thesis advisor, Dr. Hermann G. Burchard, for his continued guidance and encouragement during the course of the study. Without his assistance this study would not have been possible.

Special gratitude is expressed to Dr. Donald W. Grace for his cooperation and guidance in my studies. I would also like to thank the other members of my committee, Drs. Marvin Keener, John Wolfe, and John J. Gardiner, for their assistance and suggestions in preparing this study.

A very special expression of love and gratitude is extended to my wife, Kobra, and to our children, Cauchy and Sara, for their love, patience, and support.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Solving nonlinear optimization problems involving large numbers of variables and equality and inequality constraints has been one of the active research areas for the last two decades. Several methods with software have been developed and implemented since 1966. These methods are developed for solving nonlinear programming problems of the following types:

$$\begin{aligned} &\text{maximize} && f(x), \\ &\text{subject to} && g(x) \leq 0, \\ &&& h(x) = 0 \qquad\qquad (P0) \\ &&& \ell \leq x \leq u, \end{aligned}$$

where f, g, and h are differentiable functions from E^n into R, E^p, and E^m, respectively.

The purpose of this study is to explore the nature of three of the codes which have been produced. These codes turn out to be highly complex. One of our goals is to identify the grounds for the complexity. Two such methods (Reduced Gradient and MINOS-1.0) which are applicable to linearly constrained nonlinear problems will be described and analyzed in the second and third sections of the second chapter, respectively. Three such methods (the Robinson, the Generalized Reduced Gradient, and MINOS-5.0) which are appropriate for nonlinearly constrained nonlinear problems will be discussed in the second, third, and fourth sections of the third chapter, respectively. This study will focus on (1) the Reduced Gradient (RG), (2) MINOS-1.0, (3) the Generalized-Reduced Gradient (GRG), and (4) the MINOS-5.0 methods. Software implementations for MINOS-1.0, GRG-2, and MINOS-5.0 will be described in the second, third, and fourth sections of the fourth chapter. Some results of test runs for the codes will be evaluated in the last section of this chapter. Finally, appropriate conclusions with some suggestions for future research will be made.

Preliminary Concepts and Definitions

For ease of reference we summarize in what follows some of the basic notations, terminologies, and results. All of this can be found in the standard textbook literature [45], [61]. From now on, by Ω we shall mean a nonempty connected open subset of E^n unless it has been specified otherwise. The following outlines are similar to Luenberger [61].

Definition 1.1 (Let Ω be arbitrary): Let $f \in C^1(\Omega)$ and $x^* \in \Omega \subset E^n$, a nonzero vector $d \in E^n$ is said to be a feasible direction at x^* if there is an $\alpha > 0$ such that

$$x^* + \Theta\, d \in \Omega, \text{ for all } \Theta,\ 0 \le \Theta \le \alpha.$$

Definition 1.2: Let $f \in C^1(\Omega)$. The 1 x n matrix of the partial derivatives of f

$$\nabla f = (\partial f/\partial x_1, \ldots, \partial f/\partial x_n),$$

is the gradient of f.

Lemma 1.1: (First order necessary condition): Let $f \in C^1(\Omega)$. If x^* is a relative maximum point of f over Ω, then for any $d \in E^n$ that is a feasible direction at x^*, we have: $\nabla f(x^*)d \le 0$.

Definition 1.3: Let f $\in C^2(\Omega)$. The Hessian matrix F of f is the n x n matrix of the second partial derivatives of f over Ω. Thus F can be written as

$$F = \begin{bmatrix} \partial^2 f/\partial x_1{}^2 , & . . . & ,\partial^2 f/\partial x_1 \partial x_n \\ . & & . \\ . & & . \\ . & & . \\ . & & . \\ . & & . \\ \partial^2 f/\partial x_n \partial x_1 , & . . . & ,\partial^2 f/\partial x_n{}^2 \end{bmatrix}$$

In the following chapters of this study we will not assume that Ω is convex or that we have a global extremum of f. Nonetheless, to fix ideas, we describe here the conditions for a global maximum of f on a convex set.

Definition 1.4: A real valued function f defined on a convex set $\Omega \subset E^n$ is said to be convex if, for every x,y $\in \Omega$ and every α, $0 \leq \alpha \leq 1$, there holds

$f(\alpha x + (1-\alpha)y) \leq \alpha\ f(x) + (1-\alpha)f(y)$. If, for every $0 \leq \alpha \leq 1$ and $x \neq y$, there holds,

$f(\alpha\ x + (1-\alpha)y) < \alpha f(x) + ((1-\alpha)f(y)$,

then f is said to be strictly convex.

Definition 1.5: A function g defined on a convex set Ω is said to be concave if the function f = -g is convex. The function g is said to be strictly concave if -g is strictly convex.

Lemma 1.2: Let f $\in C^1(\Omega)$ and Ω be a convex subset of E^n. Then f is strictly convex at x^* if and only if there is a neighborhood $N(x^*)$ such that

$f(y) \geq f(x) + \nabla f(x)(y-x)$ for all $x,y \in N(x^*) \cap \Omega$, $x \neq y$.

Lemma 1.3: (Second-order necessary conditions): Let $\Omega \subset E^n$ and $f \in C^2(\Omega)$. If x^* is a relative maximum point of f in Ω, then for any feasible direction d at x^* we have

(i) $\nabla f(x^*)d \leq 0$, and

(ii) if $\nabla f(x^*)d = 0$, then, $d^T F(x^*)d \leq 0$.

Lemma 1.4: (Second-order sufficient conditions): Let $f \in C^2(\Omega)$ and $x^* \in \Omega$. If $\nabla f(x^*) = 0$, and $F(x^*)$ is negative definite, then x^* is a relative maximum of f in Ω.

Theorem 1.1: If f is a concave function defined on Ω, then the set T where f achieves its maximum is convex, and any relative maximum of f is a global maximum point of f.

Theorem 1.2: If f is a concave function defined on the convex subset Ω of E^n and $x^* \in \Omega$ such that

$\nabla f(x^*)(y-x^*) \leq 0$ for all $y \in \Omega$,

then x^* is a global maximum point for f in Ω.

Definition 1.6: Let $\{x^k\}$ be a sequence which converges to x^*, and $e_k = ||x^k - x^*||$. If there exists a number p and a constant $c \in (0,\infty)$ such that

$$\limsup_{k\to\infty} \frac{e_{k+1}}{(e_k)^P} \leq c.$$

Then p is called the order of convergence of the sequence and c is called the asymptotic error constant. If $p = 2$ or 3, the convergence is said to be q-quadratic or q-cubic, respectively.

Definition 1.7: Let $\{x^k\}$ be a sequence converging with order p to x^*, and $e_k = ||x^k - x^*||$. If the sequence of the errors $\{e_k\}$ is bounded by another sequence of a q-order p and p=2, then the convergence of $\{x^k\}$ is said to be R-quadratic.

Necessary and Sufficient Conditions for Constrained Case

Consider the following typical nonlinear problem:

$$\begin{array}{ll} \text{maximize} & f(x) \\ \text{subject to} & h(x) = (h_1(x), h_2(x), \ldots, h_m(x))^T \le (0,0,\ldots 0)^T \\ & \ell \le x \le u, \end{array} \qquad (P1)$$

where $x \in E^n$, (ℓ,u) are some given vectors of nonnegative numbers, and $h_1(x) \le 0$, $h_2(x) \le 0$, ..., $h_m(x) \le 0$ with $\ell \le x \le u$ are called the constraint functions.

Definition 1.8: A point x^* is said to be feasible if it satisfies all constraints.

Definition 1.9: An inequality constraint $h_i(x) \le 0$ is said to be active at a feasible point x^* if $h_i(x^*) = 0$ and inactive if $h_i(x^*) < 0$.

Definition 1.10: The collection of the derivatives of all differentiable curves on the surface $h(x) = 0$ passing through point x^* is said to be the tangent plane to h at x^*.

Definition 1.11: A point x^* satisfying the constraints $h(x) = 0$ is said to be a regular point of the constraints if the gradient vectors $\nabla h_1(x^*)$, $\nabla h_2(x^*), \ldots, \nabla h_m(x^*)$ are linearly independent.

Definition 1.12: Let x^* be a point satisfying the constraints

$$h(x) = (h_1(x),h_2(s),\ldots,h_m(x))^T = (0.0,\ldots,0)^T$$

$$g(x) = (g_1(x),g_2(x),\ldots,g_p(x))^T < (0,0,\ldots,0)^T,$$

and let

$$J = \{j:g_j(x^*) = 0\}.$$

Then, x^* is said to be a regular point of these constraints if the gradient vectors

$$\nabla h_i(x) \text{ and } \nabla g_j(x),\ 1 \leq i \leq m,\ j \in J$$

are linearly independent.

The following theorem shows that at regular points it is possible to characterize the tangent plane in terms of the gradients of the constraints.

Theorem 1.3: Let the surface S be defined by $h(x) = 0$. If x^* is a regular point of $h(x) = 0$, then the tangent plane M to the surface S at point x^* is

$$M = \{y:\nabla h(x^*)y = 0\}.$$

Theorem 1.4: If x^* is a regular local extreme point of function $f(x)$ subject to the constraints $h(x) = 0$, then the $\nabla f(x^*)$ is orthogonal to the tangent plane M of $h(x) = 0$ at x^*. Furthermore, there is a $\lambda \in E^m$ (called the Lagrangian multipliers vector) such that

$$\nabla f(x^*) + \lambda \nabla h(x^*) = 0$$

and the matrix

$$L(x) = F(x) + \lambda H(x)$$

is negative definite on the tangent plane M to the constrained surface $h(x) = 0$ at the given point x^*.

Considering a more general nonlinear problem, two important theorems concerning the optimal conditions may be stated as follows:

$$\begin{array}{ll} \text{maximize} & f(x), \\ \text{subject to} & g(x) \leq 0, \\ & h(x) = 0, \\ & \ell \leq x \leq u, \end{array} \qquad \text{(P2)}$$

where everything is the same as in (P1) except h(x) = 0, a vector of equality functions is added to the constraints in (P1).

Theorem 1.6: (Kuhn-Tucker conditions): Suppose x^* is a regular maximum point for Problem (P2). Then, there exist two vectors $\lambda \in E^m$ and $\mu \in E^P, \mu \leq 0$ such that

$$\nabla f(x^*) + \lambda \nabla h(x^*) + \mu \nabla g(x^*) = 0,$$
$$\mu g(x^*) = 0.$$

The vectors λ,μ are referred to as Lagrange (or Kuhn-Tucker) multipliers.

Theorem 1.7: Suppose x^* is a regular point of the constraints of Problem (P2). Then x^* is a strict relative maximum point for Problem (P2) if and only if there exist two vectors of real numbers, $\lambda \in E^m$ and $\mu \in E^p$ with $\mu \leq 0$ such that

$$\nabla f(x^*) + \lambda \nabla h(x^*) + \mu \nabla g(x^*) = 0,$$
$$\mu g(x^*) = 0,$$

and the matrix

$$L(x) = F(x) + \lambda H(x) + \mu G(x)$$

is negative definite on the subspace

$$M = \{y: \nabla h(x^*)y = 0,\ \nabla g_j(x^*)y = 0,\}$$

for all $j \in J$ where

$$J = \{j: g_j(x^*) = 0,\ \mu_j < 0\},$$

and F, H, and G are the Hessian matrices for f, h, and g, respectively.

It is assumed in the following that the reader is familiar with Linear Programming and its fundamental concepts such as *basic feasible solutions*.

CHAPTER II

NUMERICAL ALGORITHMS FOR NONLINEAR OPTIMIZATION PROBLEMS WITH LINEAR CONSTRAINTS

Introduction

In this chapter, two optimization methods for which software is available for use will be described and analyzed. These methods are the reduced gradient (RG), proposed and developed first by Philip Wolfe in 1962 [111], and the Large-Scale Linearly Constrained (MINOS-1.0), developed by Gill, Murray, and Wright [45], and Murtagh and Saunders [72].

Both methods are essentially based on the Method of Steepest Descent, one of the oldest and most widely known methods for optimizing a function of several variables (often referred to as the gradient method). The simplicity of the method and existence of its satisfactory analysis [61] have made it popular and important among the comparable existing methods. However, it has become obsolete by the availability of NEWTON and QUASI-NEWTON methods. Indeed, these are incorporated into the MINOS-1.0 code.

Description of the Reduced Gradient (RG) Method

In 1962, Philip Wolfe [115] developed the RG method for determining an optimal solution of a linearly constrained differentiable function f of n real variables, $x \in E^n$. This method generates a sequence of points $\{x_k\}$, in an effort to locate a point at which the objective function f assumes its maximum. The ideas which form the basis for the RG method can be described as follows:

Consider the problem:

$$\begin{aligned} &\text{maximize} && f(x), \\ &\text{subject to} && A\,x = b, && \text{(P3)} \\ & && x \geq 0, \end{aligned}$$

where $x \in E^n$, $b \in E^m$, A is an m x n matrix, and f is a concave twice continuously differentiable function defined from E^n to R. The constraints are given in standard form of linear programming. Assuming that every collection of m columns from A is linearly independent and every basic feasible solution to the problem has m strictly positive variables, any feasible solution x_* will have at most n-m variables having zero values, and it can be partitioned into two groups:

$$x = (x_{*b}, x_{*n})^T \text{ with } x_{*b} > 0$$

where the components of x_{*b} are called the basic variables, having dimension m and components of x_{*n} are denoted nonbasic variables having dimension n-m. For conventional notation, the basic variables are indicated as being the first m components of x_*.

Partitioning A in the same manner as x_*, the original problem can be written

$$\begin{aligned} &\text{maximize} && f(x_b, x_n), \\ &\text{subject to} && Bx_b + Nx_n = b && \text{(P3)} \\ & && x_b, x_n \geq 0, \end{aligned}$$

and considering the equality constraints in (P3), x_b can be solved in terms of x_n as follows:

$$x_b = B^{-1}b - B^{-1}Nx_n. \tag{2.1}$$

The idea is to treat the nonbasic variables x_n as independent variables. Substituting (2.1) into the objective function we obtain the *reduced objective* function. The equation (2.1) shows that a small change Δx_n can be chosen that leaves $x_n + \Delta x_n$ and $x_b + \Delta x_b$ nonnegative. Since x_b was originally taken to be strictly positive, $x_b + \Delta x_b$ will also be positive for small $||\Delta x_b||$. We may, therefore, move from one feasible solution to another one by selecting a Δx_n and moving x_n on the line $x_n + \Delta x_n \geq 0$. As a result, x_b will move along a corresponding line $x_b + \Delta x_b$. While we are moving in this manner, if a dependent variable becomes zero, the partitions must be modified. The zero valued basic variable is declared independent and one of the strictly positive independent variables is made dependent. Operationally, this basic and nonbasic variable exchange will be associated with a pivot operation in the revised simplex method.

Following the above strategy, it is clear that the objective function f(x) can be considered as a function of the nonbasic variables x_n only. From this point of view, the only constraints are nonnegativity constraints on the independent variables and hence, a simple modification of the steepest descent method satisfying these constraints is executable. The gradient of the reduced objective function (which is called the reduced gradient) can be computed using the following formula:

$$rg = \nabla_{x_n} f(x_b, x_n) - \nabla_{x_b} f(x_b, x_n)\ B^{-1}\ N, \tag{2.2}$$

Convergence of Reduced Gradient Method

The RG method, overall, provides a simple solution to the problem of determining feasible directions of ascent without requiring the number of computations required in the gradient projection method [61]. The convergence of the RG method to an optimal solution was debated during the period of 1962-1966. In 1966, Philip Wolfe [112] published a simple example in which the algorithm converged to a nonoptimal solution. This required more precise conditions for ensuring the convergence of the method to an optimal solution of a given problem of type (P3).

In response to this need, Pierre Huard [56] in 1975 introduced a new version of the RG method which produces a sequence of feasible solutions whose accumulation points are optimal solutions of the given problem (P3). Huard's version imposes three rules and two assumptions on the original version of the RG method. The rules are that the variables leaving the basis are provisionally forbidden candidacy, new basis variables are chosen only from strictly positive variables, and finally, if during the course of certain iterations the improvement is quite small, the derivatives are not recomputed in order that the RG method, by becoming identified with the projected gradient, shall have its convergence ensured. Huard's assumptions are that the feasible space (Ω) for the given problem is bounded and the Hessian of the objective function is uniformly negative definite on the solution space. It is also assumed that all feasible points in (P3) are nondegenerate in the sense that a basis can be found on the nonzero components.

Huard's Version of the Reduced Gradient Method

Huard's version of RG proceeds as follows:

Initial Step: Assume that a feasible solution x_k, and index set J, and partitions $x_k = (x_{kb}, x_{kn})$ and A = (B N) are given. Set:

E = $\emptyset$ (i.e., initialize the set of forbidden candidates for becoming basic variables empty).

Step 1: 1-(Updating Values)

1.1: - Compute $f(x_k)$ and treat it as the coefficient of the objective function, then compute the reduced gradient vector rg using the formula

$$rg(B,x) = \nabla_{x_n} f(x_b,x_n) - \nabla_{x_b} f(x_b,x_n)\, B^{-1}N \mid x = x_k.$$

Set:

$$I(k) = \{i : x_{ki} \text{ is a basic variable}\},$$

$$\bar{I}(k) = J \setminus I(k),$$

$$S_k = \{j : j \in J \ \& \ x_j \neq 0\}.$$

Compute:

$$T(B) = B^{-1}(B\ N)$$

$$t(b) = B^{-1}b.$$

Define: The inward pointing reduced gradient $(\Delta_{x_b}, \Delta_{x_n})^T$,

$$(rg,x_n)_j = \begin{cases} 0 \text{ if } x_j = 0 \text{ and } rg_j < 0 \\ rg_j \quad\quad\quad \text{otherwise.} \end{cases} \quad \text{for all } j \in \bar{I}(k).$$

Set:

$$\Delta x_n = (rg,x_n)$$

$$\Delta x_b = -B^{-1}N\Delta x_n$$

$$h = 1,\ z_{kh} = x_k,$$

$$y_{kh} = (\Delta x_b, \Delta x_n)^T$$

1.2 - If $y_{kh} = 0$, x_k is an optimal solution of the Problem (P3), stop.

1.3 - Otherwise go to Step 2.

Step 2:

2.1 - Compute Θ_{kh} such that

$\Theta_{kh} = \min(z_{khj}/-y_{khj}: y_{khj} < 0, j \varepsilon J)$.

2.2 - Set $R_{kh} = (j \in J: \Theta_{kh} = z_{khj}/-y_{khj}, y_{khj} \leq 0)$.

2.3 - Set $z_{kh+1} = z_{kh} + \Theta_{kh}\, y_{kh}$, and go to Step 3.

Step 3:

3.1 - If $h = 1$ and $\Theta_{kh} = 0$, go to Step 5.

3.2 - If $h > 1$ and $\Theta_{kh} = 0$, go to Step 6.

3.3 - Otherwise, go to Step 4.

Step 4:

4.1 - If $\sum_{i=1}^{h} \Theta_{ki} \geq 1$,

modify if necessary Θ_{kh} such that

$$\sum_{i=1}^{h} \Theta_{ki} = 1$$

and consequently, modify z_{kh} and go to Step 6.

4.2 - Otherwise, compute:

$$y_{k(h+1)}$$

4.3 - If $y_{k(h+1)} = 0$, go to Step 6.

4.4 - Otherwise, go to Step 2, with h+1 instead of h.

Step 5:

5.1 - Choose $r \in R(k_1)$ and drop its corresponding column vector from the basis B and add it to the set E(k). Instead, add one of the column vectors corresponding to the positive components of nonbasic variables (if possible among those which were not previously basic variables).

Step 6:

6.1 - Identify x_{k+1} such that

$$x_{k+1} \text{ maximizes } f(x) \text{ on the interval } (x_k, z_k).$$

6.2 - Set $B(k+1) = B(k)$, $E(k+1) = E(k)$.

6.3 - Increment k by 1 and then go to Step 1. A flowchart for this algorithm is shown in Figure 1.

In what follows, a few results and one theorem related to the convergence of Huard's version of the Reduced Gradient methods are described:

Lemma 2.1: Suppose Problem (P3) satisfies all conditions and the assumption of Huard's version of the RG-method. Also assume that x_k, B, a feasible solution, and a basis matrix for solving Problem (P3) using Huard's algorithm are given, then the following hold:

2.1.1: If $\Delta x_{kn} = 0$, then x_k is an optimal solution for (P3).

2.1.2: If $\Delta x_{kb} \geq 0$ but nonzero, then the feasible region Ω is not bounded and hence for a linear objective function, Problem (P3) does not have a finite solution.

2.1.3: Define

$$R(x_k) = \{i : x_{ki} = 0 \text{ and } x_{ki} \text{ is a basic variable}\}$$

and

$$S(x_k) = \{i : x_{ki} > 0 \text{ and } x_{ki} \text{ is a nonbasic variable}\}$$

If rank $A = m$, then $\forall r \in R(x_k)$, $\exists s \in S(x_k)$ such the $T_r^s(B) \neq 0$.

Proof 2.1.1: If $\Delta x_{kn} = 0$ at point x_k, according to the given formula (2.2) for rg, we have

$$rg(x_k) = \nabla_{x_n} f(x_k) - \nabla_{x_b} f(x_k) B^{-1} N \leq 0,$$

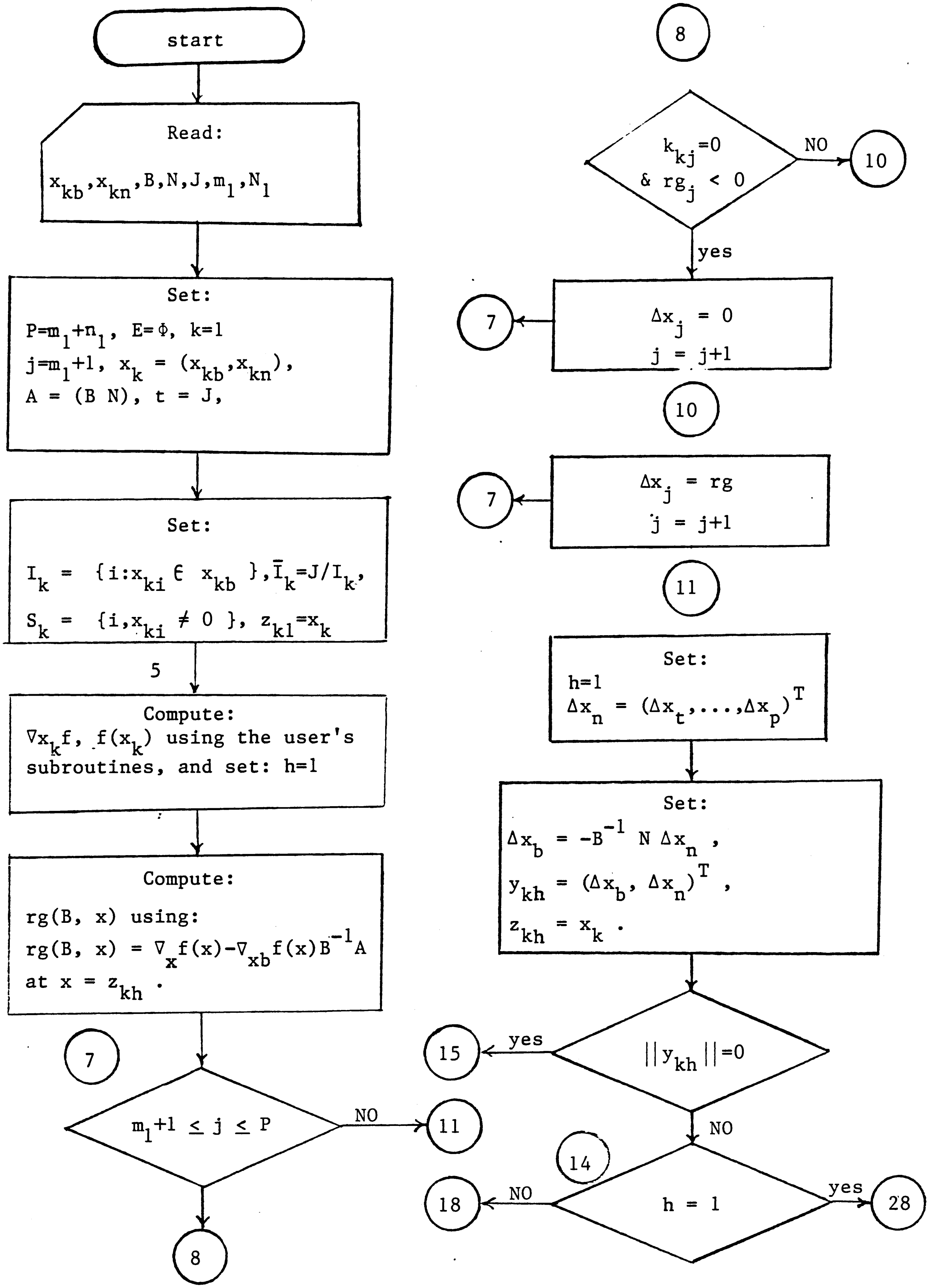
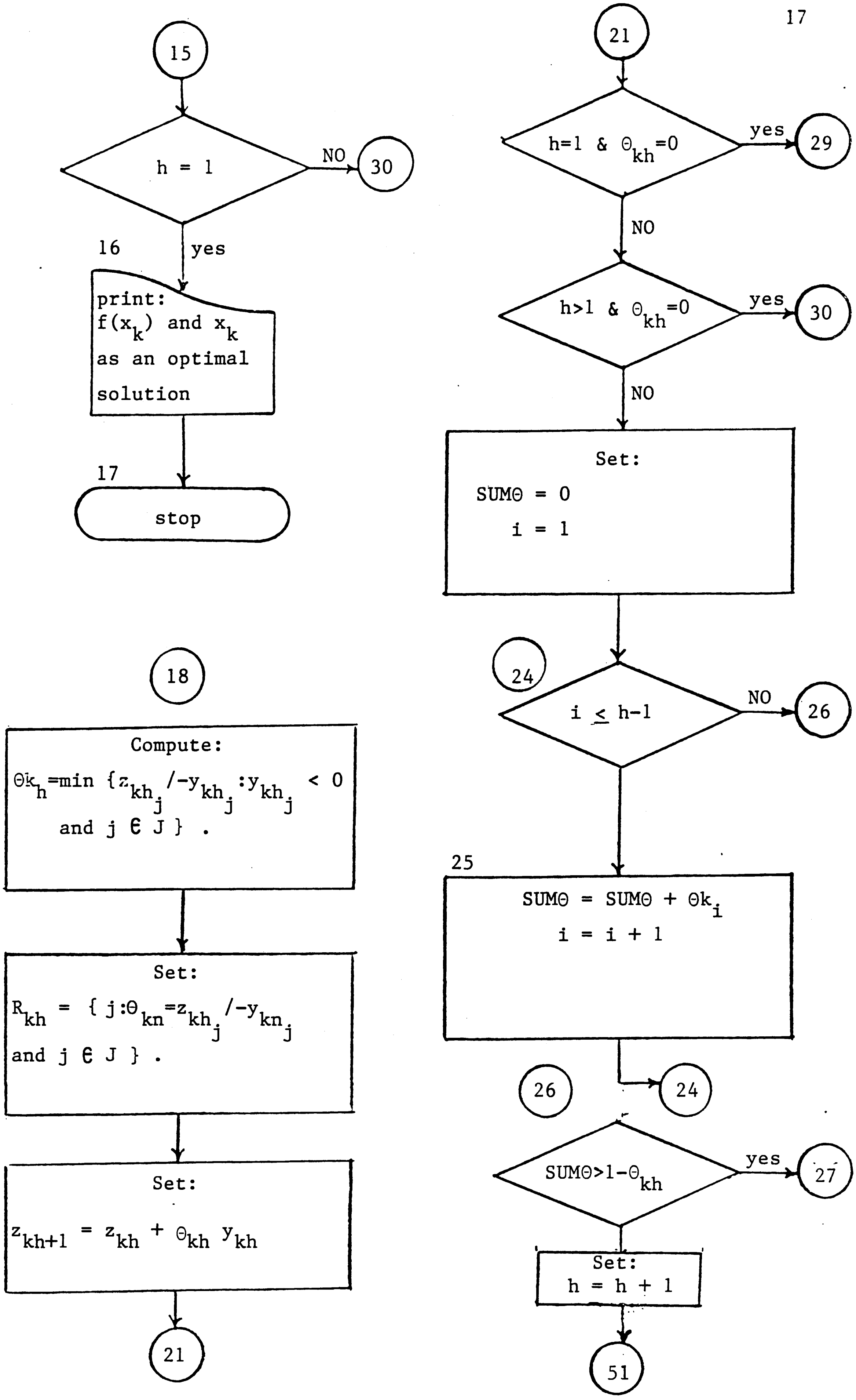
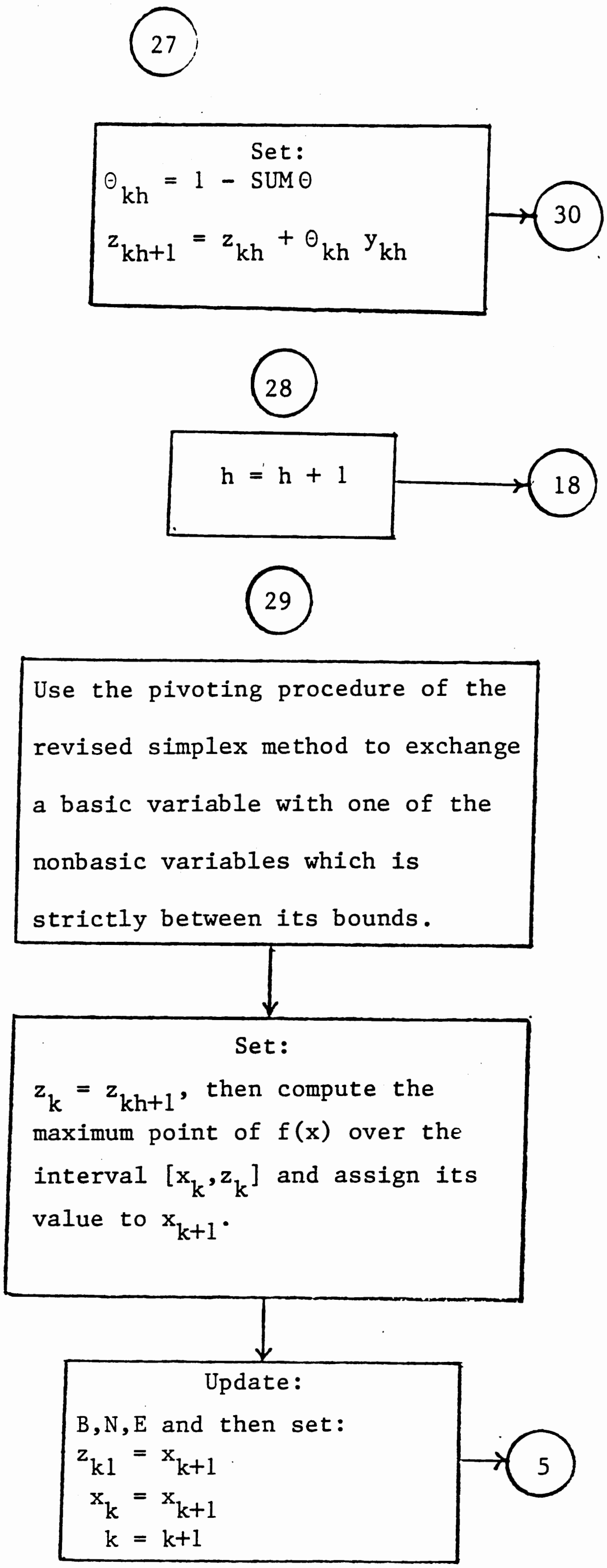

start
Read:
$x_{kb}, x_{kn}, B, N, J, m_1, N_1$
Set:
$P=m_1+n_1$, $E=\Phi$, $k=1$
$j=m_1+1$, $x_k = (x_{kb}, x_{kn})$,
$A = (B\ N)$, $t = J$,
Set:
$I_k = \{i : x_{ki} \in x_{kb}\}, \bar{I}_k = J/I_k$,
$S_k = \{i, x_{ki} \neq 0\}$, $z_{k1} = x_k$
5
Compute:
$\nabla x_k f$, $f(x_k)$ using the user's subroutines, and set: $h=1$
Compute:
$rg(B, x)$ using:
$rg(B, x) = \nabla_x f(x) - \nabla_{xb} f(x) B^{-1} A$
at $x = z_{kh}$.
7
$m_1+1 \leq j \leq P$
NO
11
8
8
$k_{kj}=0$
& $rg_j < 0$
NO
10
yes
$\Delta x_j = 0$
$j = j+1$
7
10
$\Delta x_j = rg$
$j = j+1$
7
11
Set:
$h=1$
$\Delta x_n = (\Delta x_t, \ldots, \Delta x_p)^T$
Set:
$\Delta x_b = -B^{-1} N \Delta x_n$,
$y_{kh} = (\Delta x_b, \Delta x_n)^T$,
$z_{kh} = x_k$.
$||y_{kh}||=0$
yes
15
NO
14
$h = 1$
NO
18
yes
28

Figure 1. Reduced Gradient Flowchart for Huard's Version


15
h = 1
NO
30
16
yes
print:
$f(x_k)$ and x_k
as an optimal
solution
17
stop
18
Compute:
$\Theta k_h = \min \{z_{kh_j} / -y_{kh_j} : y_{kh_j} < 0$
and $j \in J \}$.
Set:
$R_{kh} = \{ j : \Theta_{kn} = z_{kh_j} / -y_{kn_j}$
and $j \in J \}$.
Set:
$z_{kh+1} = z_{kh} + \Theta_{kh}\ y_{kh}$
21
21
$h=1$ & $\Theta_{kh}=0$
yes
29
NO
$h>1$ & $\Theta_{kh}=0$
yes
30
NO
Set:
$SUM\Theta = 0$
$i = 1$
24
$i \leq h-1$
NO
26
25
$SUM\Theta = SUM\Theta + \Theta k_i$
$i = i + 1$
24
26
$SUM\Theta > 1 - \Theta_{kh}$
yes
27
Set:
$h = h + 1$
51


27
Set:
$\Theta_{kh} = 1 - \mathrm{SUM}\,\Theta$
$z_{kh+1} = z_{kh} + \Theta_{kh}\, y_{kh}$
30
28
$h = h + 1$
18
29
Use the pivoting procedure of the revised simplex method to exchange a basic variable with one of the nonbasic variables which is strictly between its bounds.
Set:
$z_k = z_{kh+1}$, then compute the maximum point of $f(x)$ over the interval $[x_k, z_k]$ and assign its value to x_{k+1}.
Update:
B,N,E and then set:
$z_{k1} = x_{k+1}$
$x_k = x_{k+1}$
$k = k+1$
5

which means the point x_k is a local maximum point for the given problem, as it is assumed the Hessian of the objective function is negative definite. The fact that the objective function is concave on the feasible region guarantees that x_k is an optimal solution for (P3).

Proof 2.1.2: If f(x) is linear, then write $\Delta x = \Delta x_k$ and let $x = x_k + \Theta\Delta x$ be a new point with $\Theta \geq 0$. Replacing $x_k + \Theta\Delta x$ for x in (P3) results:

$$A\ x = A(x_k + \Theta\Delta x) \quad = Ax_k + \Theta A\Delta x$$

$$= b + \Theta\ (B\ N)(\Delta x_b + \Delta x_n)^T.$$

$$= b + \Theta(B\Delta x_b + N\Delta x_n). \qquad (2.3)$$

Substituting $(-B^{-1}N\Delta x_n)$ for Δx_b in (2.3) yields

$$A\ x = b + \Theta\ (-BB^{-1}N\Delta x_{kn} + N\Delta x_{kn}) = b + \Theta(0) = b. \qquad (2.4)$$

This shows that $x_k + \Theta\Delta x$ is a feasible solution for problem (P3) for all $\Theta \geq 0$.

Let us define:

$$w(\Theta) = f(x + \Theta\Delta x),\ \text{for}\ \Theta \geq 0. \qquad (2.5)$$

We claim that $dw(\Theta)/d\Theta \geq 0$ and, therefore, f(x) does not have a finite optimal solution. To prove this claim, taking the derivative of $w(\Theta)$, we have

$$dw(\Theta)/d\Theta \quad = \nabla f\Delta x$$

$$= (\nabla f_{x_b}, \nabla f_{x_n})(\Delta x_b, \Delta x_n)^T$$

$$= \nabla f_{x_b}\Delta x_b + \nabla f_{x_n}\Delta x_n$$

$$= -\nabla f_{x_b}\ B^{-1}N\Delta x_n + \nabla f_{x_n}\Delta x_n$$

$$= (\nabla f_{x_n} - \nabla f_{x_b}B^{-1}N)\Delta x_n$$

$$= rg\ (x_k)\Delta x_n. \qquad (2.6)$$

Since $\Delta x_b > 0$, $\Delta x_n \neq 0$, and thus, there exists at least a $j \in \bar{I}$ with $(rg\ (x))j \neq 0$. This result with (2.6) shows that

$dw(\Theta)/d\Theta \geq ((rg(x_k)j)^2 > 0.$

Proof 2.1.3: Since the rank A = m, for $r \in R(x)$ there must be an $s \in S$ with $T_r^S(B) \neq 0$. Otherwise, the nondegeneracy hypotheses will be violated.

Theorem 2.1: Huard's version of the RG-method in solving a problem of type (P3), satisfying Huard's conditions, and hypotheses, will generate a sequence of feasible solutions $\{x_k\}$ that converges to the unique optimal solution of $x*$ of the given problem.

Proof:

Recalling Huard's algorithm, it is clear that the algorithm in solving a problem of type (P3) generally produces an infinite sequence of feasible solutions $\{x_k\}$ such that

$$f(x_k) \leq f(x_{k+1}) \leq f(x*). \qquad (2.7)$$

This sequence will be finite only if we have at some step $\Delta x_k = 0$. In this case, according to Lemma (2.1), x_k would be an optimal solution. Therefore, without loss of generality, we may assume that this possibility will never occur, and hence the number of elements in the sequence $\{x_k\}$ is infinite. With this assumption, consider the following two cases:

Case 1: There exists an infinite subsequence $\{\Delta x_{k_i}\}$ with

$$\lim_{k \to \infty} \Delta x_{k_i} = 0. \qquad (2.8)$$

Case 2: There exists an $\alpha > 0$ such that

$$||\Delta x_{nk}|| \geq \alpha \text{ for all k.} \qquad (2.9)$$

The proof for the second case can be found in [56], and what follows is a description of the first case.

Since f(x) is a concave function in the given feasible region, the reduced objective function $w(x_n)$ is concave in the reduced feasible region obtained by replacing $(B^{-1}b-B^{-1}N\ x_n)$ for x_b in the original feasible region of (P3). This suggests that the reduced objective function $w(x_n)$ can be written as

$$w(x_n) = f(B^{-1}b-B^{-1}N\ x_n), \tag{2.10}$$

$$w(x_n) = f(x_b,x_n) = f(x), \tag{2.11}$$

for all feasible solutions x.

Using (2.7,2.8), and the fact $A\ x = b$ for all feasible x, yield

$$\begin{aligned} 0 \le f(x*) - f(x_k) &= w(x*_n)-w(x_{kn}) \\ &\le rg\ (x_k)(x*-x_{kn}) \qquad (2.12) \\ &\le \Delta x_n(x*_n-x_{kn}). \qquad (2.13) \end{aligned}$$

The second inequality holds because the reduced function $w(x_n)$ is a concave function, and the fact that $(\Delta x_n)_j \neq rg_j$ if and only if $x_j = 0$ and $rg_j < 0$ plus the fact $x* \ge 0$ yield the last inequality. Furthermore, since $\Delta x_{k_i} \longrightarrow 0$ and the sequence $\{f(x_k)\}$ is a monotonically nondecreasing sequence, the inequality (2.13) yields

$$\underset{k\to\infty}{\text{limit}}\ f(x_k) = f(x_*). \tag{2.14}$$

Finally, since Ω is convex and compact and the Hessian of f(x) is uniformly negative definite, there exist a unique optimal solution x_*. Now it similarly follows from (2.14) that

$$\underset{k\to\infty}{\text{limit}}\ x_k = x_*.$$

Description of MINOS-1.0 Method

MINOS-1.0 is designed to solve large, sparse nonlinear problems with linear constraints. This method is an extension of the RG method [111] which has been further developed variously by Philip E. Gill, Walter Murray [42], Margaret H. Wright [45], and Bruce A. Murtagh and Michael A. Saunders [73]. The method combines efficient sparse-matrix techniques as in the revised simplex method with stable quasi-Newton methods for handling the nonlinearities. MINOS-1.0 uses the active set strategy [45] in computing a search direction for improving a given feasible solution toward an optimal solution. This strategy will be discussed in the next section.

A general purpose software (MINOS-1.0) has been developed by Bruce A. Murtagh and Michael A. Saunders [73] based on this method. This method may better be described by considering problems which have the following standard form:

$$\begin{array}{lll} \text{maximize} & f(x) = f^0(x) + C^T x, & \\ \text{subject to} & A\ x \le b, & \text{(P4)} \\ & \ell \le x \le u, & \end{array}$$

where A is an m x n (m ≤ n) sparse matrix and the number of variables involved is considered to be large.

Assuming that every collection of m columns from A is linearly independent and every feasible solution to (P4) has at least m components between their given bounds, any feasible solution will have at most (n-m) components taking one of their boundary values. Having an initial solution x^0 for (P4), it is possible to perform the following two operations on x^0 and the matrix A.

First, the method partitions the given solution x^0 into three groups:

$$x^0 = (x^0{}_b, x^0{}_s,\ x^0{}_n)^T \tag{2.15}$$

where the components of $x^0{}_b$ are called basic variables having dimension m and components of $x^0{}_n$ are called nonbasic variables (those are taking one of their boundary values) having dimension r and components of $x^0{}_s$ are called superbasic variables with dimension $s = n - (m+r)$. It is clear that the dimensions of nonbasic and superbasic may vary from one solution to another solution, while the dimension of the basic variables, m, will remain fixed for all feasible solutions. Second, the method partitions the given matrix A into three matrices

$$A = (B\ S\ N), \tag{2.16}$$

where the m x m matrix B is assumed to be nonsingular and its columns correspond to the basic variables x_b, and the m x s matrix S and the m x r matrix N are corresponded to super and nonbasic variables x_s and x_n.

It is pertinent to mention that the number of superbasic variables at the given solution x indicates the number of ways which the given solution can be improved by changing one of the superbasic variables. Also, the name superbasic is chosen for the superbasic variables to highlight the role of these variables as the "driving force." They may be moved in any direction (particularly those that improve the objective function), and basic variables are then obligated to change in a definite way to satisfy the given constraints in (P4).

Active Set Strategy

To solve Problem (P4), MINOS-1.0 selects those constraints which are active at a given point x^0 and treats them as a "working set." it uses this set to compute a search direction for finding an improved solution. Obviously, the working set will be a subset of the original problem constraints and it can be used as an estimate for an active set compatible with the optimal solution. Some authors refer to the "working set," "active set," and "active surface" interchangeably. However, as Gill, Murray, and

Wright [45] believe, it is essential to recognize the set of constraints that are used to define the search direction. From now on, we refer to the set of active constraints as the active constraint surface.

Since at the given initial point x^0 the set of active constraints can become empty, MINOS-1.0 will solve Problem (P4) in two phases. The first phase will determine a feasible point that exactly satisfies a subset of the constraints $A\ x \leq b$. The second phase will generate an iterative sequence of feasible points that converge to an optimal solution of Problem (P4).

Recall that in the simplex method the basic variables may take any values between their boundary values, and the remaining variables are called nonbasic. In order to extend the simplex method concepts to Problem (P4), Murtagh and Saunders in designing MINOS-1.0 [73] introduced a new class of variables named superbasic variables. The basic and superbasic variables may vary between their bounds while in the RG approach used here their roles would be different. The superbasic variables are essentially free to move in any direction which will improve the given objective function; in fact, they are used to provide the driving force. Then the basic variables will be adjusted so that the variables x remain feasible with respect to the given constraints. If it happens that no progress can be made with the current set of superbasic variables, one or more of the nonbasic variables will be selected to become superbasic, n_s will be increased, and the process will be repeated. In the process of improving the objective function, if a basic or superbasic variable reaches one of its bounds, an adjustment will occur in which that variable is made nonbasic and the total number of superbasics will be reduced by 1.

The active constraints have a crucial influence on the computing procedure of an improved feasible point because they restrict feasible perturbations in the neighborhood of a feasible point. At a feasible point x^0, it is possible to move a non-zero distance from x^0 in any direction without violating inactive constraints which means for any vector p, $x^0 + \varepsilon p$ will stay feasible with respect to the inactive constraints if ε can be chosen small enough.

On the other hand, feasible perturbations will be restricted by the active constraints. To see this restriction, let us assume that the i-th constraint is active at the feasible point x^0, and also let us assume that the vector p is a feasible direction at x^0. The vector p can be characterized in two ways. If p satisfies $a_i{}^T p = 0$, the direction p is named a binding perturbation with respect to the i-th constraint because this active constraint remains active at all points $x^0 + \alpha p$ for all values of α; which means a move along the binding constraint i will remain "on" this constraint.

Next, if p satisfies

$$a_i^T p < 0,$$

p is named a non-binding perturbation with respect to the i-th active constraint because a positive move along the direction p will produce a new point which is "off" the i-th constraint.

In other words, since

$$a_i^T(x^0 + p) = b_i + a_i^T p,$$

the i-th constraint become inactive for any $\alpha > 0$ at the perturbed point $x^0 + \alpha p$.

Derivation of the MINOS-1.0 Method

Let us assume that f(x) is expandable using a Taylor's series with remainder of second-order

$$f(x^0 + \Delta x) = f(x^0) + g(x^0)\Delta x + 1/2(\Delta x)F(x^0 + \Theta\Delta x)\Delta x \qquad (2.17)$$

where $0 \le \Theta \le 1$, and g and F are the gradient and the Hessian matrix of the objective function f(x) respectively. Let us also assume, for the time being, that the objective function f(x) is a quadratic function. At the current solution x^0, the active constraint's surface can be described by

$$\bar{A}\,x = \begin{bmatrix} B & S & N \\ & & \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x_b \\ x_s \\ x_n \end{bmatrix} = \begin{bmatrix} \bar{b} \\ \\ b^0 \end{bmatrix} \qquad (2.18)$$

where the components of $\bar{b}$ are taken from b and the components of b^0 are taken from either ℓ or us, depending on whether the nonbasic variables x_n assume their lower or upper boundary values.

In order to define a feasible ascent direction P at the given point $x^0 + \Delta x$, we assume that $f(x^0)$ has a constrained stationary point at $x^0 + \Delta x$, satisfying (2.18). This assumption would yield the following:

$$\bar{A}\Delta x = \begin{bmatrix} B & S & N \\ & & \\ 0 & 0 & I \end{bmatrix} \Delta x = 0 \qquad (2.19)$$

Partitioning Δx into three groups

$$\Delta x = (\Delta x_b, \Delta x_s, \Delta x_n)^T, \qquad (2.20)$$

yields some conditions on Δx, for the point $x^0 + \Delta x$ being a constrained stationary point of f(x):

$$\begin{bmatrix} B & S & N \\ & & \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \Delta x_b \\ \Delta x_s \\ \Delta x_n \end{bmatrix} = 0 \qquad (2.21)$$

that is, the step Δx must remain on the surface of the active constraints.

Since $x^0 + \Delta x$ is a stationary point for f(x), according to the Kuhn-Tucker conditions (Theorem 1.2), the gradient vector g of the objective function f(x) at $x_k + \Delta x$ can be written as a linear combination of the active constraint normals. Thus, taking the derivatives from both sides of (2.17) and partitioning g into three groups of

$$g = (g_b, g_s, g_n)^T, \tag{2.22}$$

lead to the following results:

$$\begin{bmatrix} g_b \\ g_s \\ g_n \end{bmatrix} + F \begin{bmatrix} \Delta x_b \\ \Delta x_s \\ \Delta x_n \end{bmatrix} = \begin{bmatrix} B & 0 \\ S & 0 \\ N & I \end{bmatrix} \begin{bmatrix} \mu \\ \\ \lambda \end{bmatrix} = \begin{bmatrix} B\mu \\ S\mu \\ N\mu+\lambda \end{bmatrix} \tag{2.23}$$

which means that the gradient of f(x) at $x^0 + \Delta x$ is orthogonal to the surface of the active constraints. Since, in general, the objective function f(x) is not quadratic, the step Δx may not lead directly to a stationary point even though it does satisfy the given conditions (12.23). These conditions may now be used to define a feasible ascent direction as follows:

From (2.21) we have

$$\Delta x_n = 0, \tag{2.24}$$

and

$$\Delta x_b = -B^{-1} S \Delta x_s. \tag{2.25}$$

Thus, Δx can be written as a function of Δx_s as follows:

$$\Delta x = \begin{bmatrix} -B^{-1}S \\ I \\ 0 \end{bmatrix} \Delta x_s \tag{2.26}$$

Multiplication of (2.23) by the matrix

$$\begin{bmatrix} I & 0 & 0 \\ (-B^{-1}S)^T & I & 0 \\ 0 & 0 & I \end{bmatrix} \tag{2.27}$$

results in three pieces of useful information. First, it provides an expression for estimates of the Lagrange-multipliers for the general constraints

$$B^T\mu = g_b + [\ I \quad 0 \quad 0\ ]F(-B^{-1}S \quad I \quad 0)^T \Delta x_s \tag{2.28}$$

which, if x^0 becomes a stationary (i.e., $||x_s|| = 0$), (2.28) becomes

$$B^T\mu = g_b. \tag{2.29}$$

In this case, μ is analogous to the pricing vector in the revised simplex method. Solving (2.29) for μ gives

$$\mu = (B^{-1})^T g_b. \tag{2.30}$$

Referring to the solution of (2.30) by π, the next piece of information would result from the following:

$$\lambda = g_n - N^T \pi = g_n - N^T(BT)^{-1}g_b, \tag{2.32}$$

which is similar to the vector of reduced costs $C - CB^{-1}D$ in the revised simplex method. Finally, pre-multiplication of (2.23) by the matrix (2.27) produces an expression for the appropriate step Δx_s

$$[-(B^{-1}S)^T \quad I \quad 0]G[-B^{-1}S \quad I \quad 0]^T \Delta x_s = -h, \tag{2.33}$$

where

$$h = [-(B^{-1}S)^T \quad I \quad 0][g_b\ g_s\ g_n]^T = g_s - S^T\pi, \tag{2.34}$$

defining

$$Z = [-B^{-1}S \quad I \quad 0]^T, \text{ and } P_s = \Delta x_s, \tag{2.35}$$

will reduce the equation (2.33) to

$$Z^TFZP_s = -h, \tag{2.36}$$

and

$$h = Z^Tg. \tag{2.37}$$

For conventional notation, from now on, $G_a = Z^TFZ$ and $g_a = Z^Tg$ will be considered as the reduced Hessian and the reduced gradient of f(x).

Equations (2.36, 2.37, 2.23) show that, to find a feasible ascent direction P at the current point x^0, we need to compute the reduced gradient vector g_a from (2.37), and then we need to solve (2.36) for step P_s. Finally, using (2.26) returns

$$P = ZP_s. \tag{2.38}$$

Also, from equations (2.33, 2.37) it can be concluded that

$$||g_a|| = ||Z^Tg|| = 0, \tag{2.39}$$

at a stationary point. Thus, $||g_a|| = 0$ becomes a necessary condition for a point to be a stationary point on the current set of active constraints; therefore, if G_a is nonsingular, then equation (2.36) shows that $||P_s|| = ||\Delta x_s|| = 0$ at a stationary point.

Computational Procedure for a New Improved Feasible Point

Let us assume that (k=0), and $x^0 = (x_b^0\ x_s^0\ x_n^0)^T$, an initial solution for Problem (2.3) is given; also, let us assume that r, s, g, f, I, EG (gradient tolerance), MAXIT (maximum number of iterations), KFLAG (maximum number of different alternatives for the basic variables at a given point), and Z are available too. In order to compute a new feasible point x which is better than its previous one, as Figure 2 shows, the algorithm will proceed as follows:

Step 1: Check the optimality of the given solution x^0.

Compute:

1.1 - $||g_a|| = ||Z^T g||$

1.2 - If, $||g_a|| \geq EG$, set: CHECK = 1, and then go to Step 2. Otherwise, x^0 is nearly an optimal solution, therefore, compute

$$\lambda = g - N^T(B^T)^{-1} g_b.$$

1.2.1 - If there is a negative component in λ, release its corresponding nonbasic variable from its boundary, and update the set of super and nonbasic variables, r, s, Z, g_a, G, and I; then set: CHECK = 1, and go to Step 2. Otherwise, print the requested report, and then stop the procedure.

Step 2: Compute a feasible ascent direction P.

Compute:

$$P_s = -(Z^T F Z)^{-1} g_a$$

and

$$P = Z P_s.$$

Go to Step 3.

Step 3: Compute an improved point.

3.1 - Compute $\beta \geq 0$ such that $x^0 + \alpha P$ is feasible for all $0 \leq \alpha \leq \beta$. If $\beta = 0$ go to Step 2.

3.2 - Compute an α^*, using cubic or quadratic fit method, such that $f(x^k + \alpha^* P) = \max \{f(x^k + \alpha P): 0 < \alpha \leq \beta\}$.

3.3 - Set:

$$x^{k+1} = x^k + \alpha^* P.$$

and go to Step 5.

Step 4: Compute a new basis.

4.1 - If CHECK ≥ KFLAG, print CHECK and an error message, then stop.

4.2 - Identify a new set of basic components by exchanging some of the old basic components with some of the old superbasic components.

4.3 - Set

CHECK = CHECK + 1,

and then go to Step 3.

Step 5: Check the superbasic and the basic components of the new solution for being within their bounds. If there are any basic or superbasic variables encountering their bounds, exchange them with those components of the nonbasic variables which are no longer at one of their boundary values, then update r, s, g, I, A, F, and Z, and then go to Step 1. Otherwise, set K = K+1 and then go to Step 1. A flowchart for this algorithm is shown in Figure 2.

The preliminary results using MINOS-1.0 as reported in its User's Guide [72] by Murtagh and Saunders show that today MINOS-1.0 is one of the best available methods in the market. Also, the derivation and Figure 2 indicate that as Philip E. Gill, Walter Murray, and Margaret H. Wright said in Practical Optimization [45]:

> The best methods available today are extremely complex; their manner of operation is far from obvious, especially to users from other disciplines.

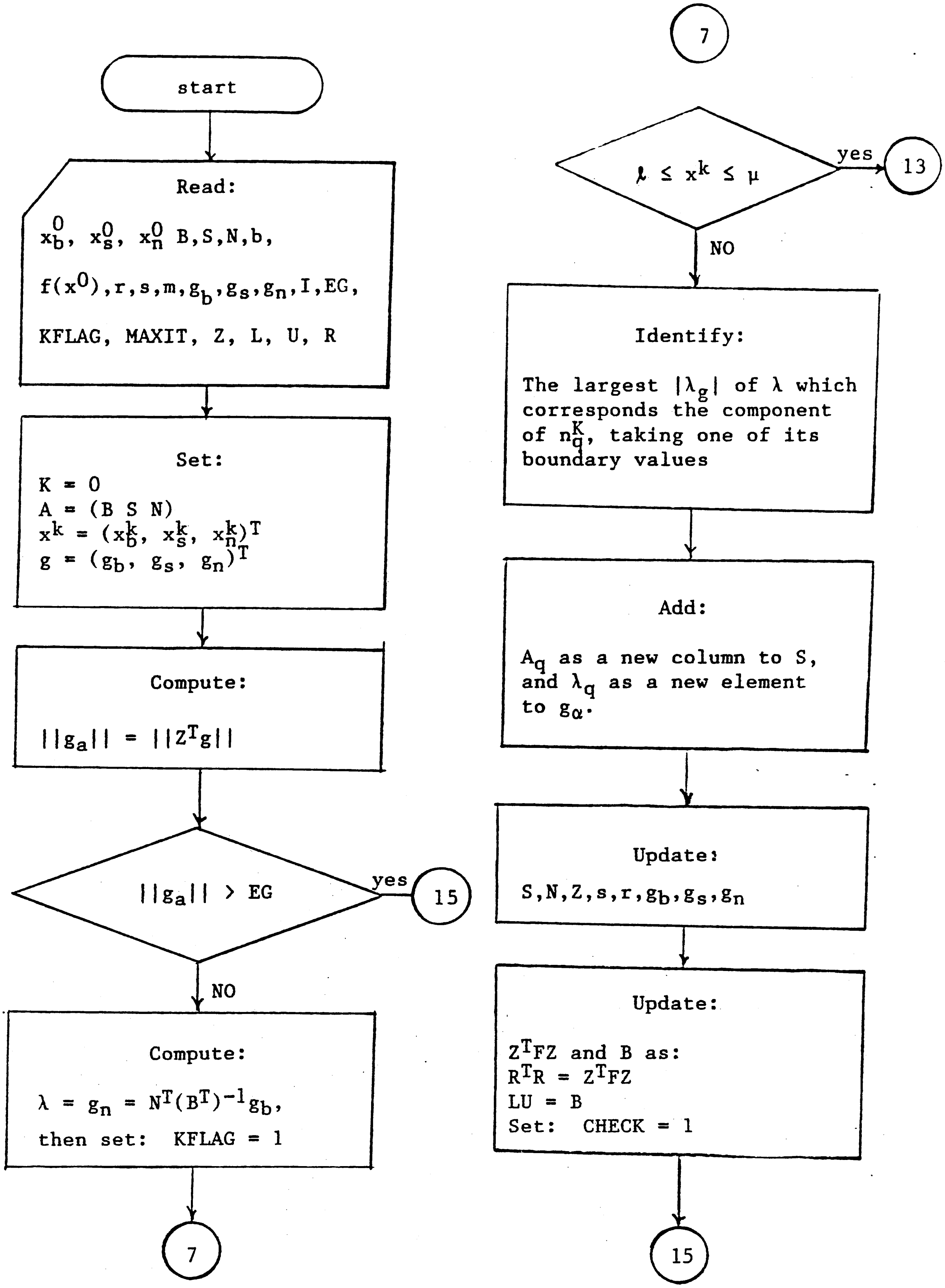
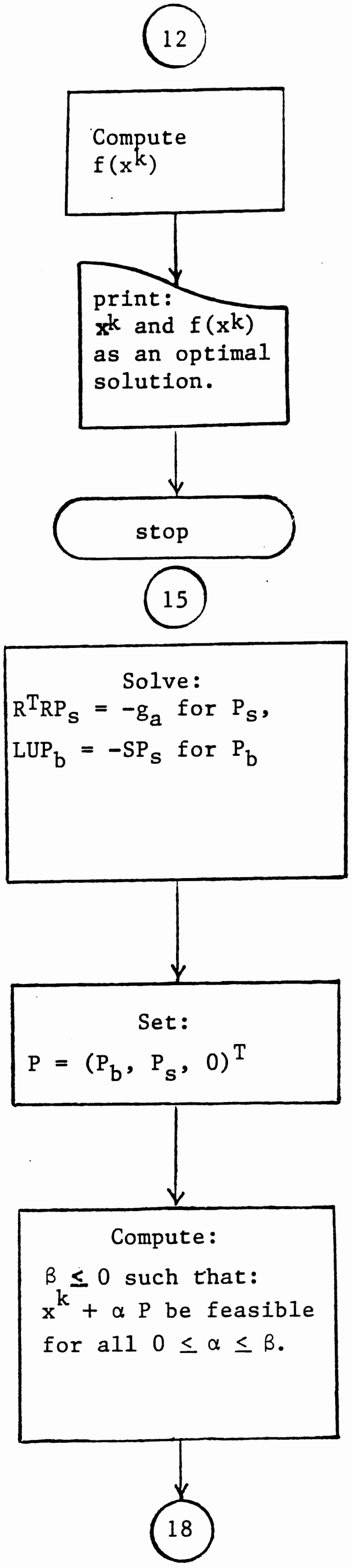
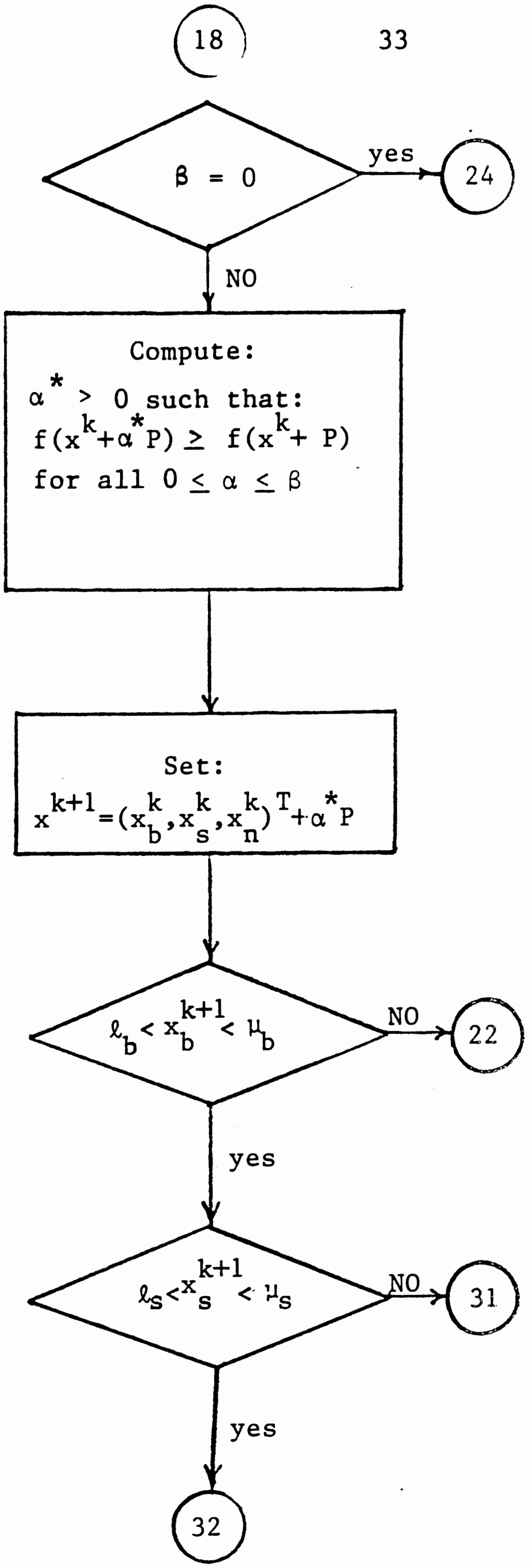
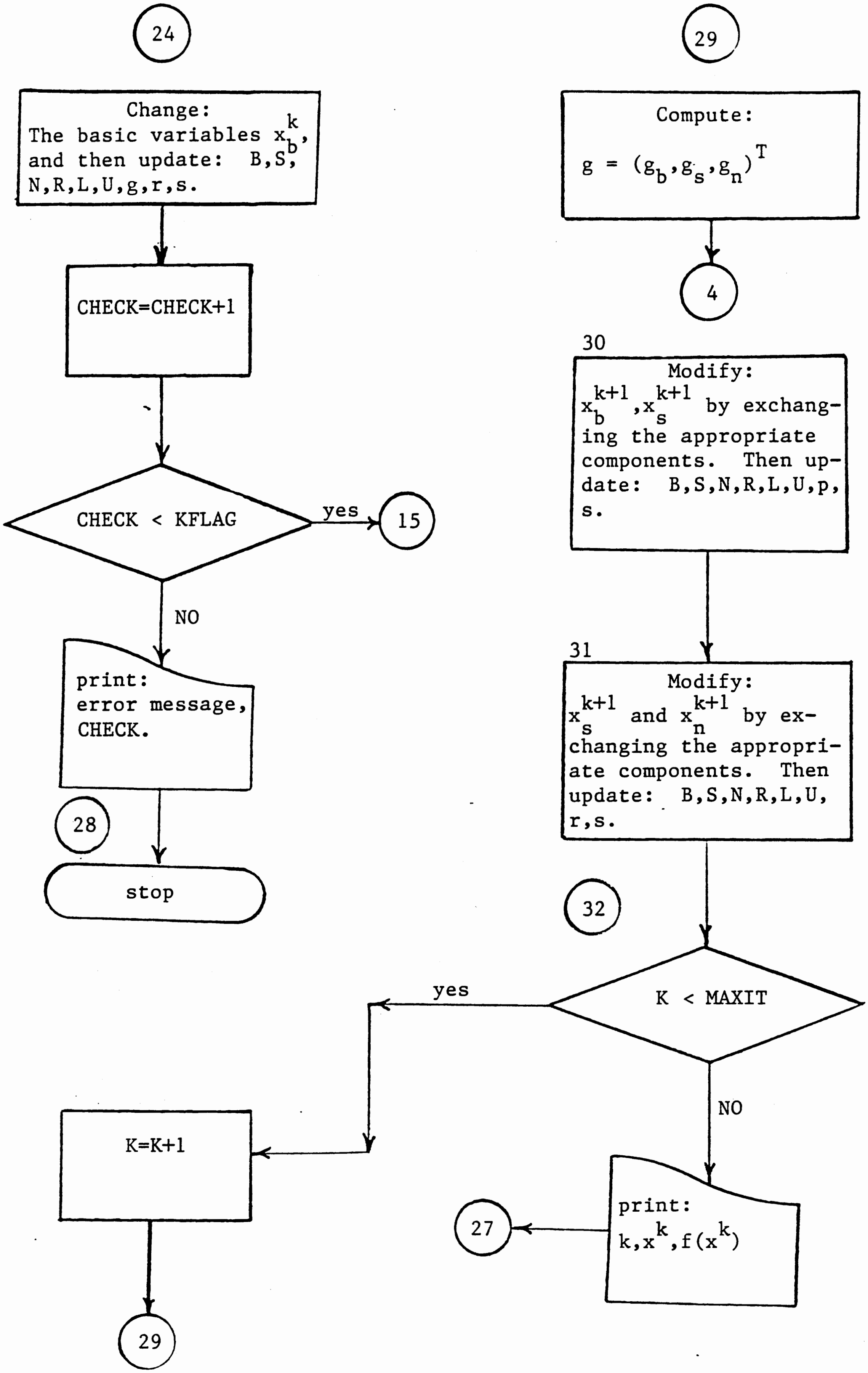

start
Read:
x_b^0, x_s^0, x_n^0 B,S,N,b,
$f(x^0)$,r,s,m,g_b,g_s,g_n,I,EG,
KFLAG, MAXIT, Z, L, U, R
Set:
K = 0
A = (B S N)
$x^k = (x_b^k, x_s^k, x_n^k)^T$
$g = (g_b, g_s, g_n)^T$
Compute:
$||g_a|| = ||Z^Tg||$
$||g_a|| > EG$
yes
15
NO
Compute:
$\lambda = g_n = N^T(B^T)^{-1}g_b$,
then set: KFLAG = 1
7
7
$\ell \leq x^k \leq \mu$
yes
13
NO
Identify:
The largest $|\lambda_g|$ of λ which corresponds the component of n_q^K, taking one of its boundary values
Add:
A_q as a new column to S, and λ_q as a new element to g_α.
Update:
S,N,Z,s,r,g_b,g_s,g_n
Update:
Z^TFZ and B as:
$R^TR = Z^TFZ$
LU = B
Set: CHECK = 1
15

Figure 2: MINOS-1.0 Flowchart


12
Compute $f(x^k)$
print: x^k and $f(x^k)$ as an optimal solution.
stop
15
Solve: $R^TRP_s = -g_a$ for P_s, $LUP_b = -SP_s$ for P_b
Set: $P = (P_b, P_s, 0)^T$
Compute: $\beta \leq 0$ such that: $x^k + \alpha P$ be feasible for all $0 \leq \alpha \leq \beta$.
18


18
$\beta = 0$
yes
24
NO
Compute: $\alpha^* > 0$ such that: $f(x^k+\alpha^* P) \geq f(x^k + P)$ for all $0 \leq \alpha \leq \beta$
Set: $x^{k+1} = (x_b^k, x_s^k, x_n^k)^T + \alpha^* P$
$\ell_b < x_b^{k+1} < \mu_b$
NO
22
yes
$\ell_s < x_s^{k+1} < \mu_s$
NO
31
yes
32


24
Change:
The basic variables x_b^k,
and then update: B,S,
N,R,L,U,g,r,s.
CHECK=CHECK+1
CHECK < KFLAG
yes
15
NO
print:
error message,
CHECK.
28
stop
29
Compute:
$g = (g_b, g_s, g_n)^T$
4
30
Modify:
x_b^{k+1}, x_s^{k+1} by exchanging the appropriate
components. Then update: B,S,N,R,L,U,p,
s.
31
Modify:
x_s^{k+1} and x_n^{k+1} by exchanging the appropriate components. Then
update: B,S,N,R,L,U,
r,s.
32
K < MAXIT
yes
NO
K=K+1
print:
$k, x^k, f(x^k)$
27
29

CHAPTER III

NUMERICAL ALGORITHMS FOR NONLINEAR OPTIMIZATION PROBLEMS WITH NONLINEAR CONSTRAINTS

Introduction

In this chapter, three algorithms of nonlinear programming will be described and analyzed. In the first section, Robinson's Lagrangian algorithm will be explained, and then a convergence theorem for it will be stated. In the next section the Generalized Reduced Gradient algorithm (GRG) for nonlinearly constrained problems will be described. Finally, the third section will be devoted to the description and the convergence behavior of MINOS-5.0.

Description of the Robinson Algorithm

This algorithm designed in 1972 [95] by Stephen M. Robinson for solving nonlinearly constrained nonlinear programming problems having the following type:

$$\begin{array}{ll} \text{maximize} & f(x) \quad , \\ \text{subject to} & g(x) \le 0 \quad , \\ & h(x) = 0 \\ & \quad x \ge 0 \quad , \end{array} \qquad \text{(P5)}$$

where f, h, and g are differentiable functions from En into R, E^m, and E^p respectively. The algorithm assumes that, f, h, g ε $C^2(\Omega)$, where Ω is an open connected neighborhood of an optimal solution $z^*=(x^*,u^*,v^*)$ for (P5). The Algorithm starts at a given solution $z^k = (x^k,u^k,v^k)$ with $u^k \geq 0$, where u^k,v^k are considered to be estimates of Lagrange-Multipliers associated with g and h respectively for (P5) and produces a sequence converging to z^*, if the starting point is sufficiently close to z^* at the given point z^k. The algorithm reduces the original problem to a linearly constrained problem and then it solves the new problem using an efficient algorithm such as RG and MINOS-1.0. Having $z^k=(x^k,u^k,v^k)$ as initial point available, the algorithm can be stated as it follows:

STEP 1:

Set: k = 0.

STEP 2:

Linearize the nonlinear constraints and then write a Lagrangian objective function.

2.1 for $1 \leq i \leq m$ and $1 \leq j \leq p$ compute:

$$Lg_i(x\ ,x^k)=g_i(x^k) + \nabla\ g_i(x - x^k)\quad ,$$

$$Lhj\ (x\ ,\ x^k) = h_j(x^k) + \nabla\ h_j(x - x^k),$$

and then set:

$$Lg(x,x^k) = (Lg_1(x,x^k),\ldots,Lg_m(x,x^k))^T,$$

$$Lh(x,x^k) = (Lh_1(x,x^k),\ldots,Lh_p(x,x^k))^T,$$

$$L(x,u^k,v^k) = f(x)+u^k[g(x)-Lg(x,x^k)]+$$

$$v^k[h(x)-Lh(x,x^k)].$$

STEP 3:

Reduce the original problem to a linearly constrained problem using the computed information from step 2 and then solve the new problem.

$$\begin{array}{lll} \text{maximize} & L(x\ ,u^k\ ,v^k)\ , & \\ \text{subject to} & Lg\ (x,x^k) \leq 0\ , & (P5^*) \\ & Lh\ (x,x^k) = 0\ , & \end{array}$$

call the solution to $(P5^*)$ by $\bar{z} = (\bar{x},\bar{u},\bar{v})$.

STEP 4:

Check the optimality conditions at point $\bar{z}$.

4.1 If $\nabla f(\bar{x}) \neq A(\bar{x})\ \bar{u}$ (where A is the matrix whose rows are the transposed gradient vectors of the active constraints evaluated at the new point z) set k = k + 1 and go to step 2.

4.2 If u has a negative component set k = k + 1 and go to step 2.

4.3 If the Hessian of $f(x) + \bar{u}\ g(x) + \bar{v}h(x)$ evaluated at $x = \bar{x}$ is negative definite on the tangent space, prepare the required report, and then stop. Otherwise, set k = k + 1 and go to step 2.

Convergence of the Robinson Algorithm.

Robinson has proved that his algorithm will converge quadratically to an optimal solution of (P5) if an initial solution for the sequence of subproblems can be chosen close enough to an optimal solution of (P5). In what follows, we first introduce some conventional notations, then state some properties of the algorithm, and finally, a convergence theorem for the algorithm will be outlined.

Notations:

1-) $q = n + m + p$.

2-) P(z): the subproblem generated by $z = (\ x\ ,\ u\ ,\ v)$.

3-) $Dg(x) = (\nabla g_1(x),\dots,\ \nabla\ g_m(x))^T$.

4-) $Dh(x) = (\nabla h_1(x),\dots,\ \nabla\ h_p(x))^T$, 5-) $DL(z) = \nabla f(x) + u^T Dg(x) + v^T h(x)$.

6-) $f_i(z) = DL_i(z)$, for all i , $1 \le i \le n$.

7-) $f_i(z) = u_{i-n} g_{i-n}(x)$, for all i, $n + 1 \le i \le n + m$.

8-) $f_i(z) = h_{i-(m+n)}(x)$, for all i, $n+m+1 \le i \le n+m+p$.

9-) $|| A ||$ = norm of A .

Theorem 3.1:

Let f,h and g $\in C^2(\Omega)$, where Ω is a feasible region for (P5). Then the following are equivalent:

1) Given (x^*,u^*,v^*), with $u^* \in E^m$ and $v^* \in E^p$, there exists $u^0 \in E^m$, $v^0 \in E^p$ such that (x^*,u^*,v^*) satisfies the necessary conditions for being an optimal point for the subproblem generated by (x^*,u^0,v^0).

2) The point (x^*,u^*,v^*) satisfies the necessary conditions for being an optimal solution for the original (P5).

3) For every $u \in E^m$ and ever $v \in E^p$ the point (x^*, u^*, v^*) satisfies the necessary conditions for being an optimal solution for the subproblem generated by (x^*, u, v).

Proof:

1⟶2: Since (x^*, u^*, v^*) is an optimal point for the subproblem generated by itself, we have

1-) $Lg(x,x^*) = g(x^*) + (\nabla g_1(x-x^*)\ldots\nabla g_m(x-x^*))^T|_{x = x^*} = g(x^*) \le 0.$

2-) $Lh(x_1,x^*) = h(x^*) + (\nabla h_1(x-x^*)\ldots\nabla h_p(x-x^*))^T|_{x = x^*} = h(x^*) = 0.$

Let $\nabla L(x^*, u^*, v^*) = A$, where A is the matrix whose rows are the transposed derivative vectors of the active constraints at $x = x^*$. Taking the derivatives from active constraints such as $Lh(x,x^*)$ shows that

$$\nabla Lg_i(x,x^*) = \nabla g_i(x^*),\ 1 \le i \le m$$

$$\nabla Lh_j(x,x^*) = \nabla h_j(x,x^*),\ 1 \le j \le p.$$

Evaluation of $L(x, u, v^*)$ at (x^*, u^*, v^*) yields that (x^*, u^*, v^*) is an optimal solution for the original problem.

2⟶3: Since (x^*, u^*, v^*) is satisfying the necessary conditions for being an optimal solution of the original problem (P5), we have:

1-) $g_i(x^*) = g_i(x) + \nabla g_i(x)(x - x^*)|_{x=x^*} \le 0, \qquad 1 \le i \le m$

2-) $h_j(x^*) = h_j(x) + \nabla h_j(x)(x - x^*)|_{x=x^*} = 0, \qquad 1 \le j \le p$

3-) $\nabla f(x^*) = \nabla f(x) + u^T[DLg(x) - DLg(x^*)]$

$$+ v^T[Dh(x)\text{-}DLh(x^*)]|_{x=x^*}$$

$$= \nabla L(x,\ u,\ v)|_{x=x^*} = A$$

where A is the matrix whose rows are the transposed gradient vectors of the active constraints at $x = x^*$.

Since

$h_i(x)|_{x=x^*} = Lh_i(x,x^*)$, for all $1 \le i \le m$,

$g_j(x)|_{x=x^*} = Lg_j(x,x^*)$, for all $1 \le j \le p$,

the matrix whose rows are the transposed gradient vectors of the active constraints of the problem generated by (x^*, u^*, v^*) satisfies the necessary conditions for being an optimal solution of the problem generated by (x^*, u, v).

3⟶1: Proof of this part is clear.

Corollary 3.1

Let f, h, and g be continuously differentiable functions defined in E^n, and let $S = \{(x^k, u^k, v^k)\}$ be a sequence generated by applying Robinson's algorithm to (P5). If the sequence $\{(x^k, u^k, v^k)\}$ converges to (x^*, u^*, v^*), then (x^*, u^*, v^*) satisfies the necessary conditions for being an optimal solution for (P5).

Proof:

Since the sequence $\{(x^k, u^k, v^k)\}$ is generated by the Robinson algorithm, then $(x^{k+1}, u^{k+1}, v^{k+1})$ satisfies the first order Kuhn-Tucker conditions for problems generated by (x^k, u^k, v^k), for each $k \geq 0$. Since (x^k, u^k, v^k) converges to (x^*, u^*, v^*), the continuity assumptions imply that (x^*, u^*, v^*) satisfies the necessary conditions for being an optimal solution for the problem generated by (x^k, u^k, v^k); therefore, using part (1→2) of Theorem 3.1 completes the proof.

For the remainder of the section we assume that the sequence $\{z^k\} = \{(x^k, u^k, v^k\}$ converges to a point $z^* = (x^*, u^*, v^*)$ where the second order sufficient conditions will satisfy for being an optimal solution.

Lemma 3.1

Let $f^0(z)$ be a function defined from E^q itself by

$$f^0(z) = (f_1(z), f_2(z) \ldots f_q(z))^T,$$

where for each i, with $1 \leq i \leq q$, $f_i(z)$ is defined according to our notations. Then $Df^0(z)$ defined by

$$Df^0(z) = \begin{bmatrix} \frac{\partial f_1(z)}{\partial Z_1} & \cdots & \frac{\partial f_1(z)}{\partial Z_q} \\ \vdots & & \vdots \\ \frac{\partial f_q(z)}{\partial Z_1} & \cdots & \frac{\partial f_q(z)}{\partial Z_q} \end{bmatrix}$$

is nonsingular at $z=z^*$.

Lemma 3.2

Let f^0 be a function defined as in Lemma 3.1, and define the functions D and P by

$$D(z,\overline{z}) = f^0(\overline{z}) - [(\overline{u}-u)^T(Dg(\overline{x})-Dg(x))+(\overline{v}-v)^T(Dh(\overline{x})-Dh(x)),$$
$$u_1(g_1(\overline{x})-Lg_1(\overline{x},x)),\ldots,u_m(g_m(\overline{x})-Lg_m(\overline{x},x)),\ h_1(\overline{x})-Lh_1(\overline{x},x),$$
$$\ldots,h_p(\overline{x})-Lh_p(\overline{x},x)]^T,$$

$$P(z_1,z_2) = \partial D(z_1,z)/\partial z|_{z=z_2} \quad .$$

Then the following hold:

1-) $D(z,\overline{z}) = 0$ if and only if the equalities of the first-order Kuhn-Tucker conditions for $P(z)$ are satisfied at $\overline{z}$.

2-) There exist positive constants μ and M such that for all z_1, z_2 in the open ball $B(z^*,\mu)$:

2.1-) $||P(z_1,z_2)-P(z^*,z^*)|| < ||Df(z^*)||/2.$

2.2-) $||f^0(z_2)-D(z_1,z_2)|| \le M\ (||z_1 - z_2||)^2.$

2.3-) $g_i(x^*) < 0$ implies that $Lg_i(x_1,x_2) < 0$, for $1 \le i \le m$.

2.4-) $u^*_i > 0$ implies that $g_i(x^*) = 0$, for $1 \le i \le m$.

Lemma 3.3

Let $\overline{z} \in B(z^*, \mu/2)$ be such that:

$$4||f^0(\overline{z})||/||Df^0(z^*)|| \le \mu.$$

Then, there is a unique $z^0 \in B(z^*, \mu/2)$ such that it satisfies the first-order Kuhn-Tucker conditions for $P(\overline{z})$ in $B(z^*, \mu/2)$, and that $||z-z^*|| \le 2\beta||f(z^*)||$, where $\beta = ||Df^0(z^*)^{-1}||$.

Proof:

Since $\bar{z} \in B(z^*, \mu/2)$, we have $\bar{B}(\bar{z}, \mu/2) \subset B(z^*, \mu)$, and according to part (2.1) of Lemma 3.2, for all $z \in \bar{B}(z^*, \mu/2)$ the following holds:

$$||P(\bar{z},z) - P(z^*, z^*)|| < (2\beta)^{-1}.$$

Let us define $T:B(\bar{z}, \mu/2) \longrightarrow E^q$ by

$$T(z) = z -(Df^0(z^*))^{-1} D(\bar{z},z). \tag{3.1}$$

Taking the derivative of (3.1), we get:

$$\frac{\partial T(z)}{\partial z} = I -(Df^0(z^*))^{-1} P(\bar{z},z). \tag{3.2}$$

Since $P(z,z) = Df^0(z)$ for any z, replacing $P(z^*, z^*)$ for $Df^0(z^*)$ in (3.2) we have

$$\frac{\partial T(z)}{\partial z} = (P(z^*,z^*))^{-1} [P(z^*,z^*) - P(\bar{z}, z)];$$

hence,

$$\left|\left|\frac{\partial T(z)}{\partial z}\right|\right| \le ||P(z^*,z^*)^{-1}|| \; ||P(z^*,z^*) - P(\bar{z},z)|| < \beta(2\beta)^{-1} = 1/2,$$

and this shows that T is a contraction on $\bar{B}(\bar{z}, \mu/2)$.

Considering (3.1), since

$$\begin{aligned} ||T(\bar{z})-\bar{z}|| &= ||-Df^0(z^*)^{-1} D(\bar{z},\bar{z})|| \\ &= ||-Df^0(z^*)^{-1}|| \; ||D(\bar{z},\bar{z})|| \\ &= ||-Df^0(z^*)|| \; ||f^0(\bar{z})|| \\ &= \beta \; ||f^0(\bar{z})|| \\ &\le \beta(\mu/4)(1/\beta) = (1 - 1/2)\;(1/2)\mu, \end{aligned}$$

it is clear that $T(\bar{z}) \in B(\bar{z}, \mu/2)$. Therefore, according to the contraction mapping principle, T has a unique fixed point z^0 in $B(\bar{z}, \mu/2)$ for which

$$||\bar{z} - z^0|| \leq (1 - 1/2)\ ||T(\bar{z}) - \bar{z}|| \leq 2\beta\ ||f^0(\bar{z})||.$$

Once (3.1) is taken under consideration, it is clear that z^0 is a fixed point for T if and only if z^0 is a zero of D. Therefore, z^0 is the unique zero of D in $B(\bar{z}, \mu/2)$, and according to part 1 of Lemma (3.2), z^0 satisfies the first-order Kuhn-Tucker conditions for $P(z^0)$. But if there is another such point $z' \in \bar{B}(\bar{z}, \mu/2)$, then we have to have that $D(z', z') = 0$, and this contradicts our earlier conclusion about the uniqueness of zero of D in the ball $\bar{B}(\bar{z}, \mu/2)$. This completes our proof for this lemma.

Theorem 3.2

Let (z^*, u^*, v^*) be a regular triple, satisfying the sufficient second order conditions for (P5), such that for each i, $1 \leq i \leq m$, either $u_i > 0$ or $g_i(x) < 0$. Also, let us assume that f, h, and g are twice continuously differentiable in an open neighborhood $U(x^*)$. Then there is a positive number δ such that if the Robinson's algorithm starts at any point (x^0, u^0, v^0) with $d\ \{(x^*, u^*, v^*), (x^0, u^0, v^0)\} < \delta$, the sequence $\{(x^k, u^k, v^k)\}$ will be generated and will converge R-quadratically on (x^*, u^*, v^*); in particular, there is some constant M_1 such that for all $k \geq 0$,

$$d((x^k, u^k, v^k), (x^*, u^*, v^*)) \leq M_1 \sum_{i=k}^{\infty}(1/2)^{2^i}$$
$$\leq Q(1/2)^{2^k},$$

where

$$Q = 2\ M_1 \sum_{i=o}^{\infty}(1/2)^{2^i}.$$

Proof:

Since $f^0(z^*) = 0$, and is a continuous function, according to Lemma 3.2, there exist constants μ, M, and δ such that

$$||f^0(z)|| \leq ((4\beta^2M)^{-1})\tau, \text{ for all } z \ \varepsilon \ B(z^*, \delta),$$

where $\tau = \min(1/2,(1/4\beta M\mu)$, $\beta = ||Df^0(z^*)^{-1}||$ and $0 < \delta \leq \mu/4$. Letting z^0 be any point in $B(z^*,\delta)$, we get

$$||z^0 - z^*|| < \delta < \mu/2,$$

and

$$4\beta||f^0(z^0)|| \leq (\beta M)^{-1}\tau \leq (1/4)\ \mu < \mu:$$

hence Lemma 3.3 guarantees the existence of a unique point z' in the ball $B(z^0, \mu/2)$ with $||\ z^0 - z'\ || \leq 2\beta\ ||\ f(z^0)\ ||$ which satisfies the first-order Kuhn-Tucker conditions for $P(z^0)$. Since z^1 is the Kuhn-Tucker point for $P(z^0)$, hence $z^1 = z'$ and the following hold:

$$D(z^0,z^1) = 0,$$

$$||z^1 - z^0|| \leq 2\beta||f^0(z^0)||,$$

$$||f^0(z^1|| = ||f^0(z^1) - D(z^0,z^1)|| \leq M(||z^0-z^1||)^2.$$

$$\leq 4(\beta^2M)(||f^0(z^0)||)^2 \leq (4\beta^2M)^{-1}\tau^2.$$

Now let us assume that for some $k \geq 1$ and all j with $1 \leq j \leq k$ we have that

$$||z^j - z^{j-1}|| \leq 2\ \beta||f(z^{j-1})|| \tag{3.3}$$

and

$$||f(z^j)|| \leq (4\beta^2M)^{-1}\ \tau^2 \tag{3.4}$$

Then we have

$$||z^k - z^*|| \leq ||z^0 - z^*|| + \textstyle\sum_{j=1}^{k}||z_j - z_{j-1}|| \tag{3.5}$$

Thus, using (3.3, 3.4) for any j with $1 \leq j \leq k$, we get

$$||z^j - z^{j-1}|| \leq 2\beta||f^0(z^{j-1})|| \leq (2\beta M)^{-1}\ \tau^{2^{j-1}} \tag{3.6}$$

$$\leq (2\beta M)^{-1}\ \tau^j \leq (2\beta M)^{-1}((1/4)\beta M\mu)(1/2)^{j-1}$$

$$= (\mu/4)(1/2)^j.$$

Using the fact that $||z^0 - z^*|| \leq (\mu/4)$ in (3.5) we get

$$||z^k - z^*|| \leq (\mu/4) + (\mu/4) \textstyle\sum_{j=1}^{k}(1/2)^j < \mu/2. \qquad (3.7)$$

Also, using k for j in (3.4) yields

$$4\beta||f^0(z^k) \leq (\beta M)^{-1}\tau 2^{k-1} < \mu/4 < \mu. \qquad (3.8)$$

Now using (3.7, 3.8) and Lemma 3.3 guarantees the existence of z^{k+1} with

$$||z^{k+1} - z^k|| \leq 2\beta||f^0(z^{k+1})|| \text{ and}$$

$$\begin{aligned} ||f^0(z^{k+1})|| &= ||f^0(z^{k+1})|| + (D(z^k,z^{k+1})|| \\ &\leq M(||z^k - z^{k+1}|| \leq 4\beta^2M||f^0(z^k)||^2 \\ &\leq (4\beta^2M)^{-1} \zeta^{2^{k+1}}. \end{aligned}$$

Thus, by induction the sequence $S = \{z^k\} = \{(x^k, u^k, v^k)\}$ exists, and both (3.2) and (3.3) hold for all $k \geq 1$. it can be shown that $\{z^k\}$ is a Cauchy sequence and, therefore, converges to some point $z^{\sim} \in \bar{B}(z^*,\mu/2)$. According to Corollary 3.1, this limit point does satisfy the first-order Kuhn-Tucker conditions for (P5); furthermore, our uniqueness property of $z^1,z^2...,$ implies that $z^{\sim} = z^*$. Taking $M_1 = (2\beta M)^{-1}$, and rewriting (3.7) results in

$$||z^k - z^*|| \leq M_1 \textstyle\sum_{i=k}^{\infty}(1/2)^{2^i} \leq 2M_1[\sum_{i=o}^{\infty}(1/2)^{2^i}](\frac{1}{2})^{2^k}.$$

which completes our proof of this theorem.

Generalized Reduced Gradient Algorithm (GRG)

This algorithm is an extension of (RG) algorithm in which problems with nonlinear constraints as in (P5) are solvable. The main concepts of the (GRG) method go back to the years 1964-1965 [1,3]. It was during these years that Abadie for the first time extended Wolfe's RG method [2] to nonlinear problems having nonlinear constraints. The original GRG method was improved by J. Carpenter and Abadie during 1966 through 1969. Meanwhile, it had been compared with some thirty other methods in a series of experiments conducted by A. R. Colville [20]. The GRG method, as coded in 1966, is still leading in the Colville ranking. Nevertheless, since then, new codes such as GRG 69, GRG-2, MINOS-1.0, LSLC by Abadie, Lasdon and Waren, Murtagh and Saunders, Lasdon and Jain, and Saunders respectively have been written with much better results in computing time, as well as in accuracy and size. In this section, the general idea behind the GRG algorithms will be reviewed, then the required conditions for global convergence of GRG algorithms will be discussed.

Description of the Algorithm

Let us consider the following optimization problem,

$$\begin{array}{ll} \text{maximize} & f(x), \\ \text{subject to} & h(x) = 0 \\ & \ell \le x \le u, \end{array} \qquad \text{(P6)}$$

where x, ℓ, u are n-dimensional vectors, h(x) is an m-dimensional column vector, and f(x) is a real valued function defined over E^n. Both f and h are assumed to be continuously differentiable on the feasible region defined by (P6). Furthermore, it is assumed that for every feasible solution x there exists a partition of x into (y,z) such that y (basic or dependent variable) has dimension m and z (nonbasic or independent variable) has dimension n-m satisfying the following properties:

1. y is strictly between its bounds; and
2. The m x m Jacobian matrix A defined by

$$A(i,j) = \frac{\partial^2 h_i(x)}{\partial x_i \partial x_j}, \quad i,j = 1,2...m,$$

is nonsingular.

To describe the method, let us suppose that an initial feasible solution $x^0 = (y^0, z^0)$ satisfying our assumptions is available.

Since the matrix A is nonsingular, by the Implicit Function Theorem, there exists in some neighborhood $U(x^0)$, a unique continuous function $x = (y(z),z)$ such that $y(z^0) = y^0$, and $h(y(z),z)$ is identically zero in U. Furthermore, y(x) has a continuous derivative dy/dz which can be computed by the chain rule

$$\partial h/\partial z + (\partial h/\partial y)\ \partial y/\partial z = 0,$$

or more conveniently by

$$\partial y/\partial z = -(\partial h(y,z)/\partial y)^{-1} \partial h(y,z)/\partial z.$$

Now, substituting y(z) for y in the objective function f(y,z) yields the reduced objective function w(z)

$$w(z) = f(y(z),z).$$

The gradient of this function is called the reduced gradient

$$rg{=}\nabla_z f + \nabla_y f \partial y/\partial z = \nabla_z f(y,z) - \nabla_y f(y,z)[\nabla_y h(y,z)]^{-1}\nabla_z h(y,z).$$

Setting:

$$C = \nabla_z f,\ D = \nabla_y f,\ B = \nabla_y h,\ N = \nabla_z h,$$

rg can be rewritten as

$$rg = C - D\ B^{-1}\ N.$$

Let us define the projected gradient q by its components as

$$q_j = \begin{cases} 0 & \text{if } z_j = 1 \text{ and } rg_j < 0, \\ 0 & \text{if } z_j = u \text{ and } rg_j > 0, \\ rg_j & \text{otherwise} \end{cases}$$

It is clear that the Kuhn-Tucker conditions for (P6) at a given point x reduces to q = 0, and D B is the row vector of multipliers corresponding to the constraints h(x) = 0, so if $q \neq 0$, improvement toward optimality is possible; otherwise, the given point x is an optimal point (P6). Thus, to pursue the algorithm we may assume that q is not zero at the given point x.

Now, let P be any nonzero feasible direction such that $q.P \geq 0$. It is clear that this is an ascent direction for the reduced objective function w(z). We may use P = q as an ascent direction. Anyway, the maximum point of this function along this direction is computable by using the ordinary calculus.

Let z^* be the maximum point of w(z) along the ascent direction P, which means we have

$$w(z^*) = w(z + a^*P) = \max\{f(y(z + \alpha\ P), z + \alpha\ P)\}$$

for some value of α satisfying

$$0 < \alpha \leq 1.$$

Now $x^* = (y^*, z^*) = (y(z^*), z^*)$ can be thought of as an improved point for (P6). It is clear that the new point has been computed by moving linearly along the tangent surface defined by $z = z^0 + \Delta\ z = z^0 + q$, $y = y^0 + \Delta\ y$ with $\Delta y = -B^{-1}N\Delta z$. Therefore, the point $x^* = (y^*, z^*)$ is not on the constraining surface and it needs to be modified into one which is on the surface h(x) = 0. As Figure 3, and example corresponding to n = 3, m = 1, $\ell = 0$, $u > 0$, shows that to return to the surface h(x) = 0, an iterative method such as the Newton method,

$$y^{k+1} = y^k - [\nabla yh(x^0)]^{-1}h(y^k, z^*),$$

can be used to solve the nonlinear system,

$$h(y, z^*) = 0\ , \qquad (P6^*)$$

for y while using y^* as initial solution for (y^k). Then, the magnitude bounds on the dependent variables for the solution to the system will be checked. If any of the dependent variable components violates the boundary conditions, the computed α^* will be reduced to $\alpha^*/2$. Then, the process will be restarted from the maximization of the reduced objective function w(z) with the aim of producing a solution to (P6) which meets the boundary conditions.

Assuming that the m-dimensional vector y^* is a solution to ($P6^*$) which satisfies the boundary conditions, $x^* = (y^*,z^*)$ is an improved solution for (P6). Next, the given x^0 will be replaced with the improved x^* and then the whole process will be repeated. To terminate the process, the components of the projected reduced gradient vector q, as well as some termination tolerance after computing each solution x^* will be checked.

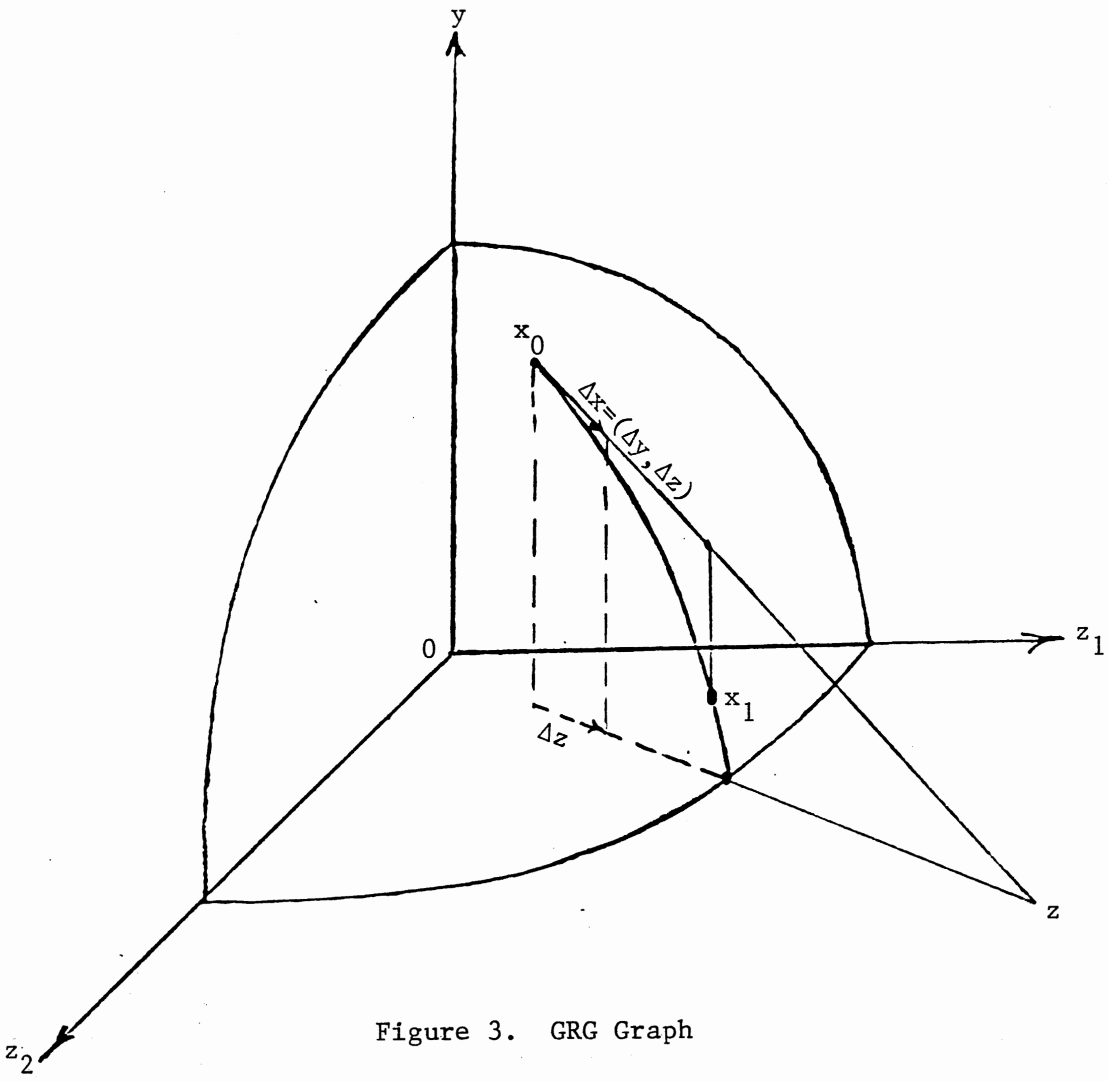


Figure 3. GRG Graph

Considering our discussion, an algorithm with its flowchart for the GRG method are outlined as they follow:

Step 0: Assume that some feasible point x^0 is known.

Step 1: This step, just for convention, is broken into substeps.

1.1. Compute the Jacobian A and the gradient g of the objective function.

1.2. Determine a partition (y^0, z^0) for x^0, and the corresponding partition (B, N) for A, such that y^0 is strictly between its bounds and B is an m x m nonsingular matrix.

1.3. Compute B^{-1}.

1.4. Compute the Lagrange multipliers λ and the reduced gradient vector rg.

1.5. Computer $q = \Delta z$.

1.6. If $q = 0$, then terminate; otherwise go to Step 2.

Step 2: Choose the ascent direction $P = q$.

Step 3: Choose a step size α.

Step 4: Maximize the reduced objective function w(z), where

$$w(z^0 + \alpha P) = f(y(z^0 + \alpha P), z^0 + \alpha P),$$

with respect to α. Let $\bar{z} = z^0 + \alpha^* P$ be the maximum point for the reduced function.

Step 5: Solve the nonlinear system of m equations,

$$h(y, \bar{z}) = 0,$$

for y while $\bar{z}$ is fixed and name its solution $\bar{y}$.

Step 6: Check the boundary conditions for the computed solution in Step 5. If they do satisfy set $\bar{x} = (\bar{y}, \bar{z})$, then go to Step 7; otherwise, reduce α to $\alpha/2$ and go back to Step 4.

Step 7: If the termination criteria is satisfied, prepare the output report and then stop; otherwise, set x^0 is equal to the computed solution $\bar{x}$ in Step 6 and then go back to Step 1.

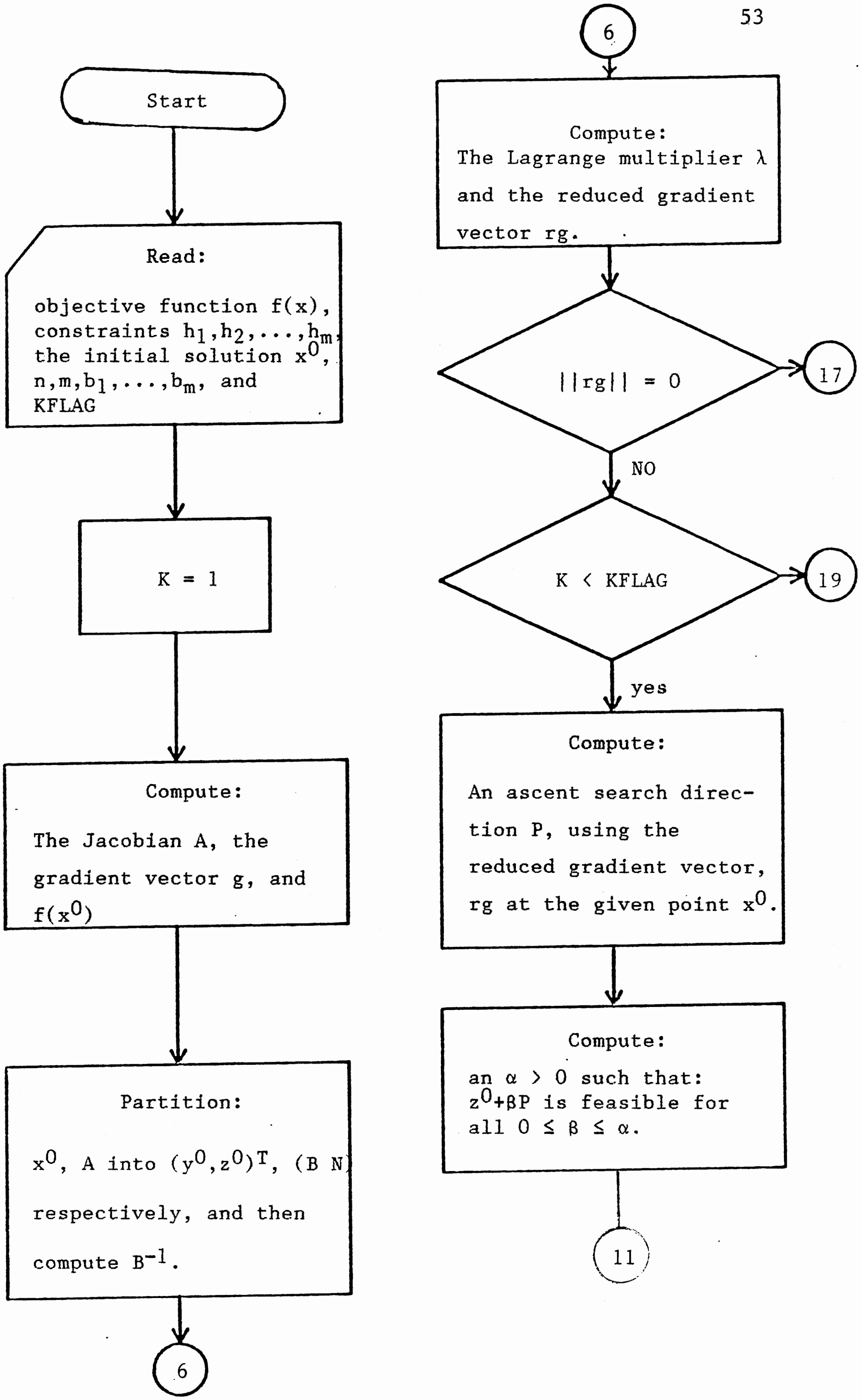
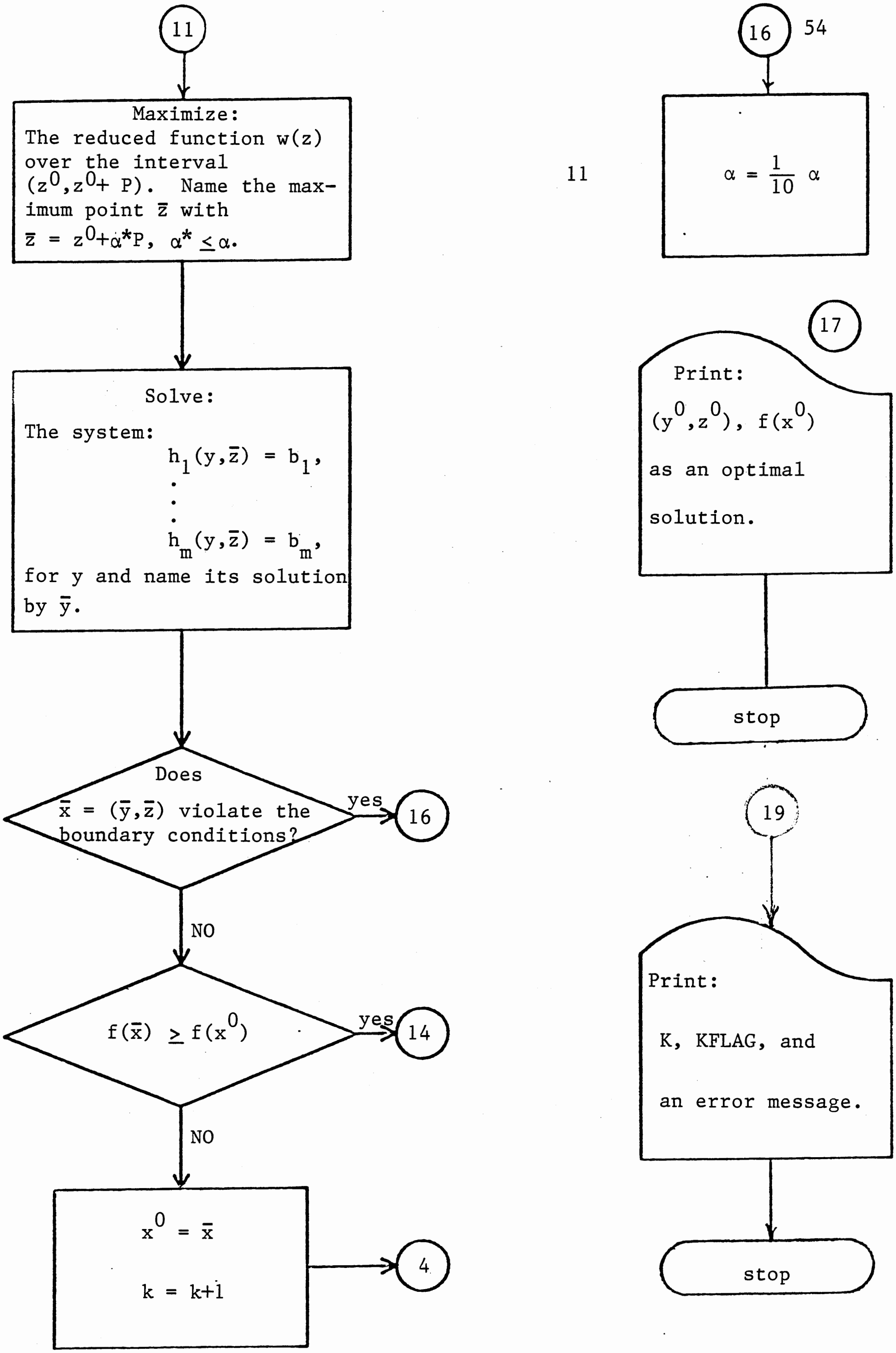

Start
Read:
objective function f(x), constraints $h_1,h_2,\ldots,h_m$, the initial solution x^0, $n,m,b_1,\ldots,b_m$, and KFLAG
K = 1
Compute:
The Jacobian A, the gradient vector g, and $f(x^0)$
Partition:
x^0, A into $(y^0,z^0)^T$, (B N) respectively, and then compute B^{-1}.
6
6
Compute:
The Lagrange multiplier λ and the reduced gradient vector rg.
||rg|| = 0
17
NO
K < KFLAG
19
yes
Compute:
An ascent search direction P, using the reduced gradient vector, rg at the given point x^0.
Compute:
an $\alpha > 0$ such that: $z^0+\beta P$ is feasible for all $0 \le \beta \le \alpha$.
11

Figure 4. GRG Flowchart


11
Maximize:
The reduced function w(z) over the interval $(z^0, z^0 + P)$. Name the maximum point $\bar{z}$ with $\bar{z} = z^0 + \alpha^* P$, $\alpha^* \leq \alpha$.
Solve:
The system: $h_1(y,\bar{z}) = b_1$, . . . $h_m(y,\bar{z}) = b_m$, for y and name its solution by $\bar{y}$.
Does $\bar{x} = (\bar{y},\bar{z})$ violate the boundary conditions?
yes
16
NO
$f(\bar{x}) \geq f(x^0)$
yes
14
NO
$x^0 = \bar{x}$
$k = k+1$
4
11
16
$\alpha = \frac{1}{10}\alpha$
17
Print:
(y^0, z^0), $f(x^0)$
as an optimal solution.
stop
19
Print:
K, KFLAG, and
an error message.
stop

Global and Local Convergence of GRG Algorithm

In considering iterative algorithms for finding either maximum or minimum points, there are two essential issues involved: global convergence properties and local convergence properties. The first issue is concerned with whether a given algorithm starting at an arbitrary point will, in fact, generate a sequence that converges to a solution point. This aspect is referred to as global convergence analysis since it addresses the important question of whether the algorithm, when initiated far from the solution point, will eventually converge to it. The global convergence of the GRG is assured according to the Global Convergence Theorem if it can be shown that the GRG algorithm is a closed map. Local convergence properties are a measure of the ultimate speed of convergence and they generally determine the relative advantage of one algorithm to another while both are able to perform the same task. In what follows, the global convergence for the GRG algorithm is going to be discussed.

Global Convergence

The GRG method, overall, provides a simple solution to the problem of computing feasible directions of ascent without the many complexities required in the gradient projection method; however, the resulting algorithm is not closed and, therefore, is subject to the possibility of jamming [61]. The algorithm is not closed essentially because a slight movement away from the boundary of an inequality constraint can cause a sudden change in the direction of search. Fortunately, as suggested by Luenberger [61], modifying the process of determining feasible direction to one which is closed is possible and hence the modified algorithm is not subject to jamming.

Definition 3.1: Let $\Omega \subseteq E^n$ be a given feasible region. A set $T \subseteq E^{2n}$, consisting of pairs (x,P), with P as a feasible direction at x, is said to be a uniformly feasible direction set for Ω if there is an $\alpha > 0$ such that (x,P) $\in$ T implies that $x + \beta P$ is feasible for all β, $0 \leq \beta$ α. The number α is referred to as the feasibility constant of the set T.

Definition 3.2: Let $T \in E^{2n}$ be a set of uniformly feasible direction vectors for Ω with feasibility constant α. Also, let f be a real valued function defined on Ω. A map $M_\alpha : T \longrightarrow \Omega$ defined by:

$$M_\alpha(x,P) = \{y|f(y) \geq f(x + \beta\ P),\ ) \leq \beta \leq \alpha;\ y = x + \beta^* P$$

for some $\beta^*, 0 \leq \beta^* \leq \alpha\}$,

is called a feasible constrained line search map.

Theorem 3.3: Let $\Omega \subset E^n$ be a given feasible region, and let T be a set of uniformly feasible direction vectors with the feasibility constant α. If $P \neq 0$, then any feasible constrained line search map is closed at (x,P).

Now let us define a modified GRG algorithm by following Steps 0 - 7 in Section 3.3, and using Luenberger suggestions [61] for the modifications in Steps 1 and 4:

Step 1:

$$q_j = \begin{cases} 0 & \text{if } z_j = \ell_j \text{ and } rg_j < 0, \\ 0 & \text{if } z_j = u_j \text{ and } rg_j > 0, \\ x_j rg_j & \text{otherwise.} \end{cases} \qquad (3.9)$$

Step 4:

4.1: Set $T = \{(y,z,P):x = (y,z)$ is a feasible point and P is a feasible direction at z obtained using (3.9)$\}$.

4.2: Maximize the reduced function w(z) in direction P by choosing a feasible constrained line search map M_α defined on T(the set of uniformly feasible direction vectors with feasibility constant α).

Corollary 3.2: Let Ω be a region for (P6) and let FD be a function defined from E^n to E^{2n} by

$$FD(y,z) = (y,z,q),$$

where q is computed according to (3.9). Then FD is a closed function.

Theorem 3.4: The GRG Modified Algorithm is a closed map and when it is applied to (P6) it will generate a sequence $S = \{x^k\}$ converging to an optimal solution of (P6).

Proof: The proof of the first part follows from Theorem (3.3) and Corollary (3.2), and the proof of the second part follows from the global convergence theorem [61].

MINOS-5.0 Algorithm

This algorithm is a new version of the MINOS/Augmented, which itself is an extension of MINOS-1.0 algorithm, which was originally designed by Murtagh and Saunders [73] for solving large nonlinear optimization problems having just linear constraints. The satisfactory performance of MINOS-1.0 in solving problems such as chemical equilibrium, the weapon assignment, and the expanded energy system model has encouraged Murtagh and Saunders to develop a new version of the existing algorithm for solving nonlinear optimization problems having nonlinear constraints as well.

The results of their effort, "MINOS/AUGMENTED," was introduced to the market in the year 1980. This algorithm has solved problems such as the Electric Power Scheduling [74] (involving 1200 nonlinear constraints and 1300 variables) and the Air Pollution Control [75] (involving 4150 variables and about 850 constraints) with satisfactory results. Such satisfactory results motivated, in less than three years, about 180 academic and research institutions around the world to install the MINOS/AUGMENTED as a system.

The continuing inquiries and the positive response from the users of MINOS/AUGMENTED with diverse applications have inspired its authors to pursue further development to meet the needs of its users. The result of their prolonged refinements to the basic algorithms such as:

1. The simplex method (Dantzig, 1947, 1963),
2. A quasi-Newton method (Davidson, 1959),
3. The reduced gradient method (Wolfe, 1962), and
4. A projected Lagrangian method (Robinson, 1972; Rosen and Kreuse, 1972)

MINOS-5.0, while it was available for use, was published in 1983. The new algorithm incorporates the advanced linear programming technique of the revised simplex method, an appropriate quasi-Newton method for approximation of the reduced Hessian matrix of the objective function, and Robinson's method to solve problems from small-scale to large-scale in the four areas of smooth optimization:

1. Linear programming,
2. Unconstrained optimization,
3. Linearly constrained optimization, and

4. Nonlinearly constrained optimization.

Mathematically speaking, MINOS-5.0 is a Fortran-based algorithm designed to solve large-scaled optimization problems of the following form:

$$\begin{aligned} \text{maximize} \quad & f(x,y) = \overline{f}(x) + c^T x + d^T y, \\ \text{subject to} \quad & f^0(x) + A_1\ y = b_1, \\ & A_2\ x + A_3\ y = b_2, \qquad \text{(P7)} \\ & \ell \leq \begin{bmatrix} x \\ y \end{bmatrix} \leq u, \end{aligned}$$

where the vectors, c,d,b_1,b_2,ℓ,u are constant and A_1,A_2,A_3 are $m_1 \times n_2$, $m_2 \times n_1$, $m_2 x n_2$ constant matrices respectively, $\overline{f}(x)$ is a twice (scalar) differentiable function, and $f^0(x)$ is an $m_1 \times 1$ vector of twice continuously differentiable functions. Components of x are denoted by nonlinear variables with dimension n_1, and the components of y are known by linear variables having dimension n_2. MINOS-5.0 assumes that the Problem (P7) has at least a regular local maximum point named (x^*, y^*, λ^*).

Linear Programming

When the functions $\overline{f}(x)$ and $f^0(x)$ are absent in (P7), the Problem (P7) becomes a linear programming problem. Since there is no need for nonlinear variables in our problem, our variable can be referred to just by x. Thus, the linear programming problem can be written as

$$\begin{aligned} \text{maximize} \quad & f(x) = c_1{}^T x \\ \text{subject to} \quad & Ax + Is = b, \qquad (\overline{\text{P}}7) \\ & \ell \leq \begin{bmatrix} x \\ s \end{bmatrix} \leq u, \end{aligned}$$

where $c_1 = (c^T,d^T)$, $B = (b_1,b_2)^T$, and

$$A = \begin{bmatrix} 0 & A_1 \\ A_2 & A_3 \end{bmatrix}$$

The components of x are referred to as structural variables and the components of s are named slack (logical) variables.

MINOS-5.0 solves linear optimization problems such as (P7) using a reliable implementation of the primal simplex method (the revised simplex method). The revised simplex method partitions $A\ x + Is = b$ into

$$A\ x + I\ s = (B\quad N) \begin{bmatrix} x_b \\ x_n \end{bmatrix}$$

where the basis matrix B is nonsingular, and the components of x_b, x_n are named the basic and nonbasic variables, respectively. At any given pivoting iteration, each nonbasic variable is equal to its upper or lower bound, and the basic variables take on whatever values are needed to satisfy the following equations:

$$x_b = B^{-1}b - B^{-1}NX_n.$$

The revised simplex method reaches an optimal solution for ($\bar{P}7$) by performing a sequence of iterations. In each iteration, one column of B is replaced by one column of N (and vice-versa), until no such interchange can be found that will increase the value of the objective function $c_1{}^T x$.

Linearly Constrained Optimization

When the function $f^0(x)$ is absent in (P7), the problem is considered as a linearly constrained nonlinear optimization problem, and again it can be written as

$$\begin{aligned} &\text{maximize} && f(x) = \overline{f}(x) + c_1 x \\ &\text{subject to} && Ax + Is = b, && (P7^{**}) \\ &&& \ell \leq x \leq \mu \end{aligned}$$

where A, C_1, x, s, and b are defined as in the previous case. MINOS-5.0 solves such problems using MINOS-1.0 algorithm (Murtagh and Saunders, 1976) given in Section 3 of Chapter II.

Nonlinearly Constrained Optimization

When the functions $\overline{f}(x)$ and $f^0(x)$ are present in (P7), the problem (P7) is classified as nonlinearly constrained nonlinear optimization problem and MINOS-5.0 solves this type of problem through a sequence of major iterations, each one involving a linearization of the nonlinear constraints at some given point x^k, using a first-order Taylor's series approximation:

$$f^i(x) = f^i(x^k) + \nabla f^i(x^k)^T(x-x^k) + 0||x-x^k|| \tag{3.10}$$

Thus, using (3.10), an approximation for $f^0(x)$ at a given point x^k will be computed as

$$\begin{aligned} f^*(x-x^k) &= f^0(x^k) + J(x^k)(x-x^k), \\ f^* &= f^0{}_k + J_k(x-x^k), \end{aligned} \tag{3.11}$$

where J(x) is the m_1 x n_1 Jacobian matrix whose (i,j)-th element is

$$\frac{\partial f^i}{\partial x_j} ;$$

and then using 3.11 yields

$$f^0 - f^* = (f^0 - f^0_k) - J_k(x-x^k). \tag{3.12}$$

Thus, (at k-th major iteration), the linearly constrained subproblem $P(x^k, \lambda^k, P)$ will be formed as

$$\begin{aligned} &\text{maximize} && L(x,y,x^k,\lambda^k,\rho) = \overline{f}(x) + c^Tx + d^Ty - (\lambda^k)^T(f^0-f^*) \\ &&& \qquad + 1/2\ \rho(f^0-f^*)^T(f-f^*), \\ &\text{subject to} && f^* + A_1\ y = b_1, \\ &&& A_2x + A_3y = b_2, \\ &&& \ell \leq \begin{bmatrix} x \\ \\ y \end{bmatrix} \leq u. \end{aligned} \tag{$\overline{P7}$}$$

where the objective function L is referred by merit function, which is a modified Lagrangian, in which $f - f^*$ is used instead of $f + A_1y - b_1$, and it is similar to the one that Robinson used in his algorithm for producing the subproblems with the exception that Robinson used $\rho = 0$. λ^k is an estimate of the Lagrangian multipliers λ^* for the original Problem (P7) which typically is unknown. The penalty term is used in ($\overline{P7}$) to ensure that the Lagrangian function L maintains a positive definite Hessian in the tangent surface to the constrained surface at the given point x^k. The use of this term in L was suggested by Arrow and Solow and adopted later by Sargent and Murtagh [72]. The necessity for using this term will be discussed shortly.

As Figure 5 shows, in order to solve this subproblem, the algorithm MINOS-1.0 given in Chapter II will be called and then a convergence test on

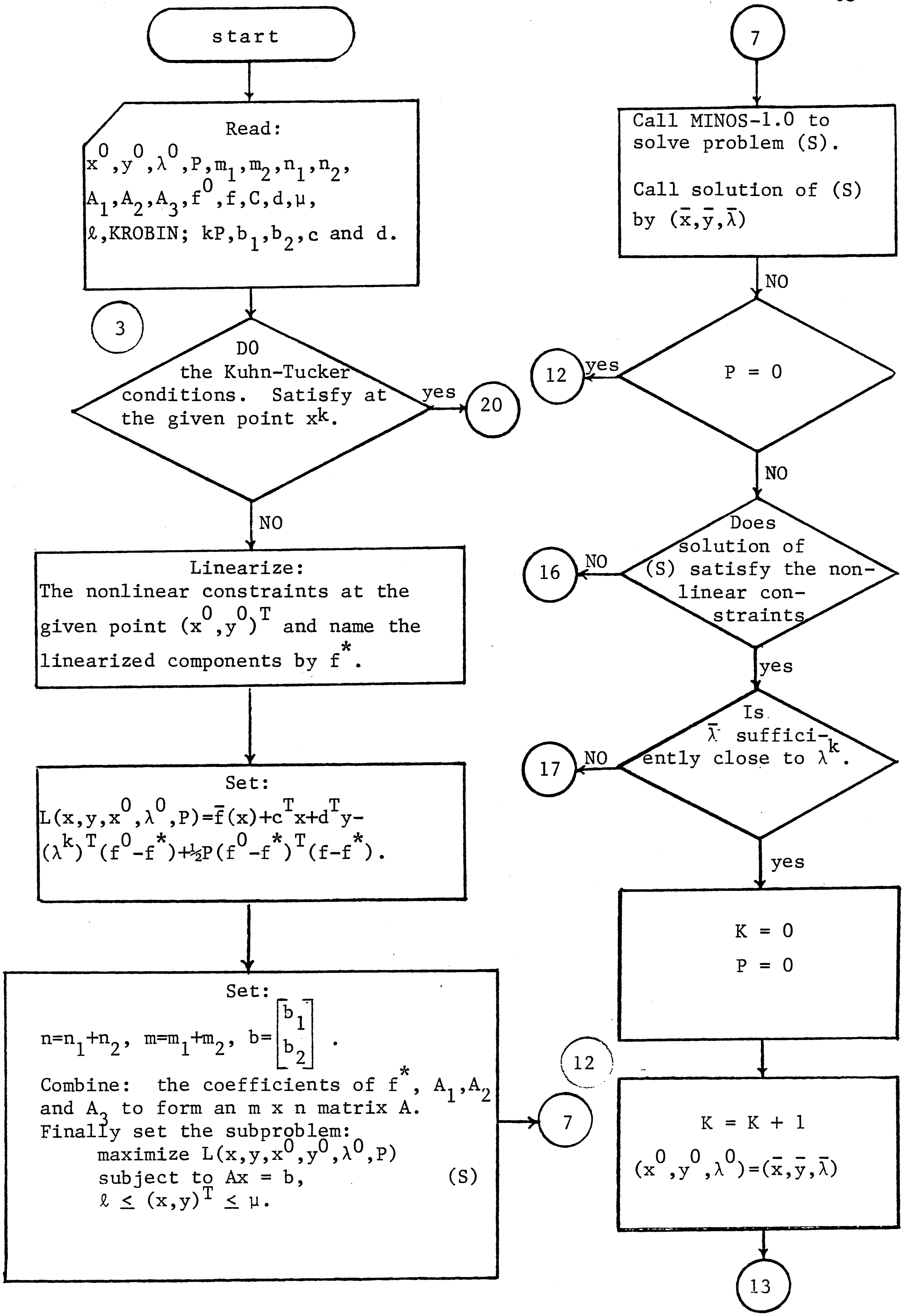
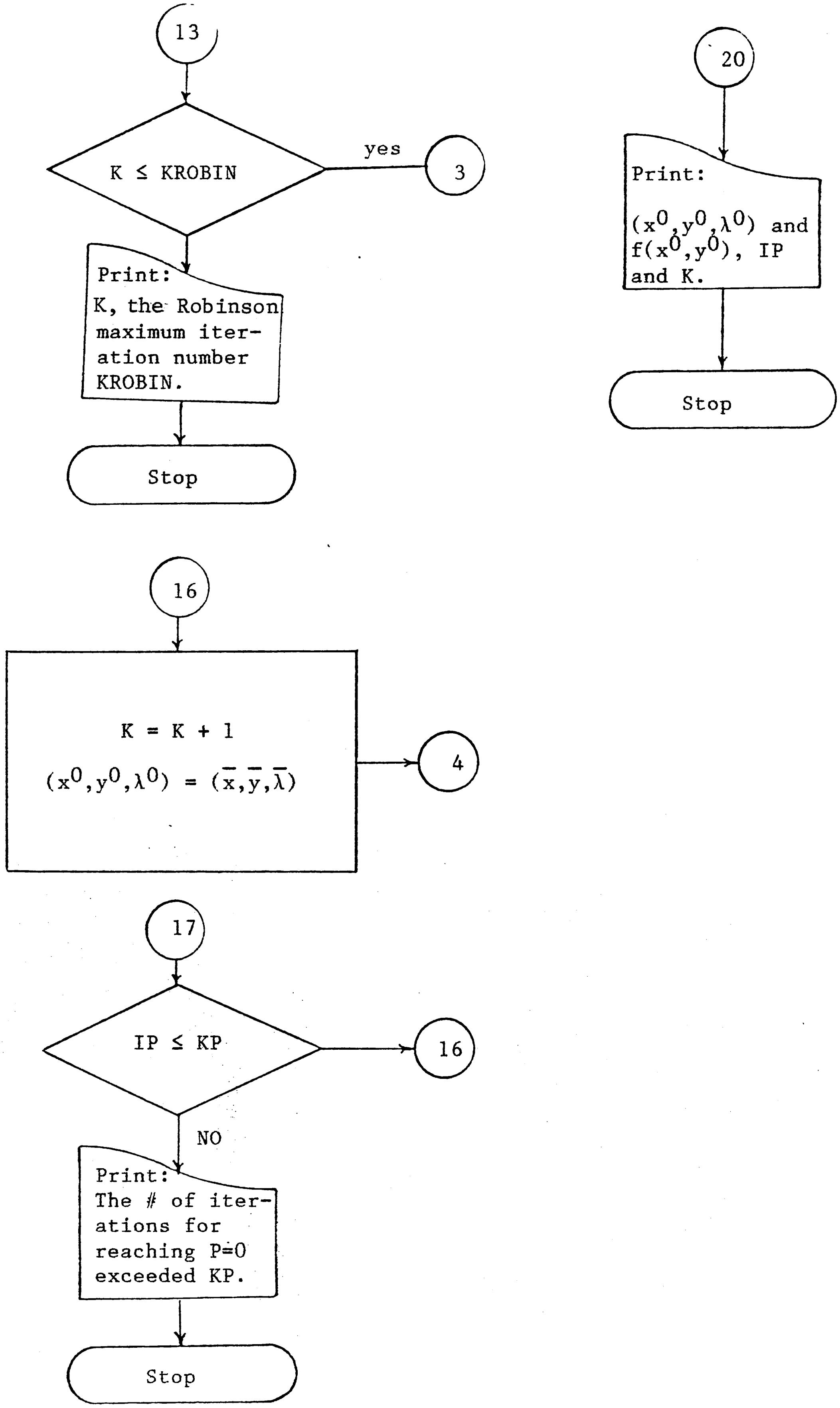

start
Read:
$x^0, y^0, \lambda^0, P, m_1, m_2, n_1, n_2,$
$A_1, A_2, A_3, f^0, f, C, d, \mu,$
ℓ,KROBIN; kP,b_1,b_2,c and d.
3
DO the Kuhn-Tucker conditions. Satisfy at the given point x^k.
yes
20
NO
Linearize:
The nonlinear constraints at the given point $(x^0, y^0)^T$ and name the linearized components by f^*.
Set:
$L(x,y,x^0,\lambda^0,P) = \bar{f}(x) + c^T x + d^T y - (\lambda^k)^T (f^0 - f^*) + \frac{1}{2} P (f^0 - f^*)^T (f - f^*)$.
Set:
$n = n_1 + n_2$, $m = m_1 + m_2$, $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$.
Combine: the coefficients of f^*, A_1, A_2 and A_3 to form an m x n matrix A.
Finally set the subproblem:
maximize $L(x,y,x^0,y^0,\lambda^0,P)$
subject to $Ax = b$, (S)
$\ell \leq (x,y)^T \leq \mu$.
7
7
Call MINOS-1.0 to solve problem (S).
Call solution of (S) by $(\bar{x}, \bar{y}, \bar{\lambda})$
NO
P = 0
yes
12
NO
Does solution of (S) satisfy the non-linear constraints
NO
16
yes
Is $\bar{\lambda}$ sufficiently close to λ^k.
NO
17
yes
K = 0
P = 0
12
K = K + 1
$(x^0, y^0, \lambda^0) = (\bar{x}, \bar{y}, \bar{\lambda})$
13

Figure 5. MINOS-5.0 Flowchart


13
K ≤ KROBIN
yes
3
Print:
K, the Robinson maximum iteration number KROBIN.
Stop
20
Print:
(x^0,y^0,λ^0) and $f(x^0,y^0)$, IP and K.
Stop
16
K = K + 1
$(x^0,y^0,\lambda^0) = (\bar{x},\bar{y},\bar{\lambda})$
4
17
IP ≤ KP
16
NO
Print:
The # of iterations for reaching P≐0 exceeded KP.
Stop

the sequence of $\{x^k\}$ will be checked. If $||x^k-x^{k-1}|| \leq$ some given tolerance, then the penalty parameter will be set to zero and then a Robinson's type algorithm will be called to solve the new subproblems; otherwise, if the sequence $\{x^k\}$ does not behave smoothly, then the value of the penalty parameter p will be increased and the process will be continued. It is clear that if the sequence of $\{x^k\}$ is behaving nicely (i.e., $||x^k - x^{k-1}||$ getting smaller and smaller) the process of introducing new subproblems and solving them will be continued.

The Necessity of the Penalty Term

It can be shown that (x^*,λ^*) may not satisfy the necessary and sufficient Kuhn-Tucker conditions for the Lagrangian function.

$$L(x,\lambda) = \overline{f}(x) + c^T x + d^T y - \lambda^T (f(x) + A_1 y - b). \tag{3.13}$$

Taking the derivative of $L(x,\lambda)$ with respect to x yields

$$\nabla_x L(x,\lambda)|_{x=x^*} = \nabla\overline{f}(x) + c^T - \lambda^T J(x)|_{x=x^*}, \tag{3.14}$$

and this shows that if it is assumed that $J(x^*)$ is nonsingular, then λ^* exists and (3.13) is stationary at (x^*,λ^*), while $L(x^*,\lambda^*)$ may have a negative curvature, which means in this case (x^*,λ^*) is not a candidate for being an optimal solution for (3.13).

Now considering the components of (x) as basic variables and applying the MINOS-1.0 algorithm to $(\overline{P7})$, show that (x^*,λ^*) is a local maximum for $L(x,\lambda)$ if, and only if:

$$z(x^*)^T \partial L/\partial x|(x,\lambda) = (x^*,\lambda^*) = 0,$$

and

$$z(x^*)^T(\partial^2 L/\partial x^2|(x,y) = (x^*,\lambda^*))z(x^*)$$

is positive semidefinite, where $z(x^*)$ is as defined in (2.35). When elements of $\{x^k\}$ get close enough (in the Robinson sense) to x^*, then as Robinson's algorithm has shown the solutions to the $L(x^k,\lambda^k)$ will converge to x^*; this means that as long as the elements of the sequence $\{x^k\}$ are not close enough to x^*, the penalty term with relatively large value for ρ must be included in the modified Lagrangian objective function. But the main issue is how the distance of the elements of $\{x^k\}$ from the x^* can be measured. The following theorems [76] show that, if a subproblem $P(x^k,\lambda^k,\rho)$ can be solved such that its solution (x^{k+1},λ^{k+1}) falls within the radius of convergence in Robinson's theorem, then the parameter ρ can safely be reduced to zero and hence according to Theorem 3.2, the convergence of the subproblem's solutions to an optimal solution of (P7) can be assumed.

Theorem 3.5: Let (x^0,λ^0) be an approximate solution to the problem

$$\text{P0:} \quad \text{maximize} \quad \bar{f}(x),$$
$$\text{subject to} \quad f(x) = b$$

where $\bar{f}$ and f are twice continuously differentiable with bounded Hessians. Also let $(\bar{x}, \bar{\lambda})$ be a solution to the linearized subproblem

$$S_1: \quad \text{maximize} \quad \bar{f}(x) - (\lambda^0)^T(f-f^*) + 1/2\ \rho(f-f^*)^T(f-f^*),$$
$$\text{subject to} \quad f^*(x,x^0) = b.$$

If $\bar{\lambda} - \lambda^0 = \epsilon_1$ and $f(\bar{x})-f^*(\bar{x}) = \epsilon_2$, then $(\bar{x}, \bar{\lambda})$ is also a solution to the perturbed problem

$$\text{P:} \quad \text{maximize} \quad \bar{f}(x) + (\epsilon_1 + \rho\ \epsilon_2)^T(f-\bar{f}),$$
$$\text{subject to} \quad f(x) = \epsilon_2 + b,$$

for sufficiently small ϵ_1 and ϵ_2.

Theorem 3.6: Let (x^0,λ^0) be an approximate solution to P0 given in Theorem (3.5) and let $(\bar{x},\bar{\lambda})$ be a solution to the linearized subproblem:

$$S_2: \quad \text{maximize} \quad \bar{f}(x) - (\lambda^0)^T(f-f^*) + (1/2)\rho f^T f,$$
$$\text{subject to} \quad f^*(x,x^0) = b.$$

If $\bar{\lambda} - \lambda^0 = \epsilon_1$ and $f(\bar{x})-f^*(\bar{x}) = \epsilon_2$, then $(\bar{x},\bar{\lambda})$ is also a solution to the perturbed problem

$$P_2: \quad \text{maximize} \quad \bar{f}(x) + \epsilon_1{}^T(f-f^*) + \rho\ \epsilon_2{}^T f$$
$$\text{subject to} \quad f^*(x,x^0) = b + \epsilon_2.$$

While these theorems do not provide a full analysis of convergence of MINOS-5.0, at least they give some indications that the convergence proof for Robinson's method might be carried over to MINOS-5.0. Such a proof may depend on the detailed specification in the algorithm of how the parameter ρ is varied.

CHAPTER IV

SOFTWARE CODES BASED ON RG/ROBINSON METHODS

Introduction

This chapter will describe three of the software packages which are commercially available for solving the nonlinear optimization problems. These codes are (1) MINOS-1.0, developed by Murtagh and Saunders [73] for linearly constrained nonlinear problems having sparse Jacobian matrices; (2) GRG-2, designed by Lasdon, Waren et. al [58] for nonlinear optimization problems having nonlinear/linear constraints; and, (3) MINOS-5.0, developed by Murtagh and Saunders [76] for large-scale nonlinear problems having nonlinear/linear constraints.

Sections one, two, and three of this chapter describe MINOS-1.0, GRG-2, and MINOS-5.0 as collections of several subroutines. Some desirable features for nonlinear programming software will be described in the last section of this chapter.

MINOS-1.0 Code

MINOS-1.0 is an optimization code developed by Murtagh and Saunders [72]. The word MINOS is an abbreviation for "a modular in core nonlinear system"; it is pronounced like "minus". The code is designed to optimize an objective function F(x), satisfying some condition by finding such a point x^*

which makes $F(x^*)$ as close to ±∞ as possible. This software is based on the reduced gradient method of Wolfe [115] and the variable metric method of McCormick [65]. Also, it uses the unconstrained optimization techniques of Gill and Murray [40] and the pivoting procedure of the revised simplex method.

Mathematically speaking, the software is designed to solve problems of the following form:

$$\begin{array}{ll} \text{maximize} & f(x) + C^T x \\ \text{subject to} & A(x) = b \\ & \ell_i \le x_i \le \mu_i, \quad i = 1,2,\ldots, n, \end{array} \tag{4.1}$$

where $f(x)$ is a continuously differentiable function defined from E^n into R, A is an m x n matrix with $m \le n$, and the assumptions made for the (LSLC) method in Chapter II are invoked.

To solve problems (4.1), the software incorporates the LU factorization for the m x m matrix corresponding to the basic variables and the $R^T R$ factorization of a Quasi-Newton approximation for the reduced Hessian matrix. The software is intended for use primarily as a system, which simply solves a sequence of problems and then terminates the entire procedure. As Figure 6 shows, to invoke the software the user needs to write a main program and call in the subprogram GO. In what follows, descriptions for several subroutines of the software are given.

Main Program

The main program declares a single array of length 10,000 as the working space (the size of working space can be easily increased whenever it

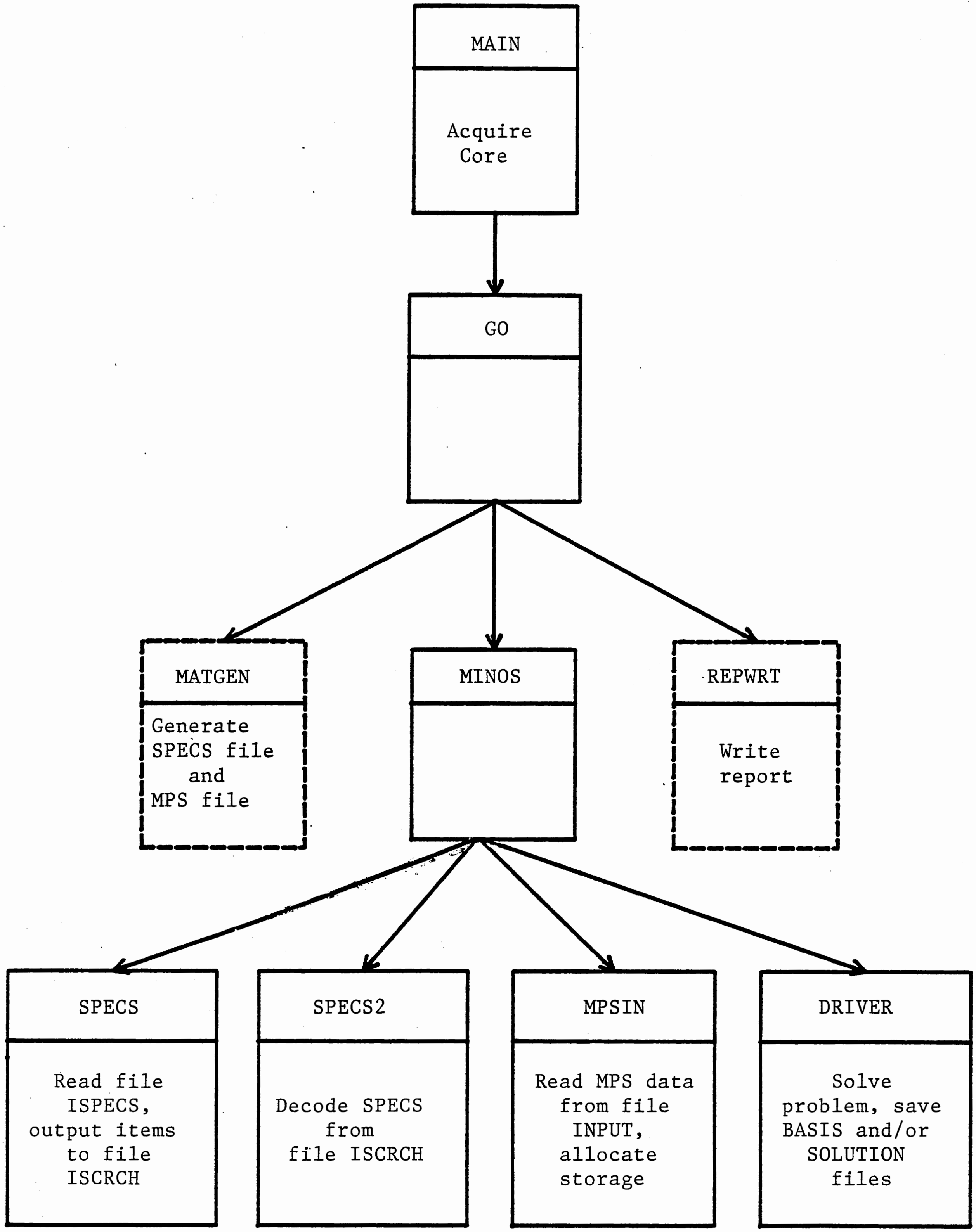

MAIN
Acquire Core
GO
MATGEN
Generate SPECS file and MPS file
MINOS
REPWRT
Write report
SPECS
Read file ISPECS, output items to file ISCRCH
SPECS2
Decode SPECS from file ISCRCH
MPSIN
Read MPS data from file INPUT, allocate storage
DRIVER
Solve problem, save BASIS and/or SOLUTION files

Figure 6. Subroutine Structure of MINOS-1.0
from
MINOS USER'S GUIDE
TECHNICAL REPORT 77-9

is needed); then it provides the input parameters for the subroutine MINOS-1.0 through calling subprogram GO.

Subroutine GO

This subprogram is a control routine. It provides all of the necessary input parameters for MINOS-1.0 by calling the subroutine INPUT. Then it calls the system MINOS-1.0 to solve the given problem. Finally, when the problem is processed, it prints the required output by using the subroutine REPORT.

Subroutine MINOS-1.0

This subroutine is a composition of several subprograms, which as a whole incorporates the advanced linear programming techniques of the revised simplex method, the reduced gradient method, and an appropriate Quasi-Newton method for approximation of the reduced Hessian matrix to solve the given problem. MINOS-1.0 and its subprograms communicate with their users by means of files such as SPECS, MPS, BASIS, REPORT, and SOLUTION. Among the output arguments of MINOS-1.0, there are parameters which define:

a) the condition at which the process of solving was terminated;

b) the dimensions of variables of the problem that has been solved;

c) the positions of certain subarrays such as the solution vector x, the dual vector, and the state vector defining the state of each variable x_j as basic, superbasic, or nonbasic at upper or lower bound.

Since, on one hand, the analysis of the code is not appropriate in this study because of its length (about 10,000 lines of Fortran) and, on the other hand, the significance of the code would not become clear to its users without an understanding of its major steps, in what follows, a pseudocode program for this code is given.

Assuming that the following information are available:

1) a feasible vector x;

2) the corresponding functional value F(x);

3) a decomposition of A as {B S N} = A;

4) the corresponding gradient vector partition, $g(x) = \{gb, gs, gn\}^T$;

5) the number of basic, superbasic, and nonbasic variables m,s,r, with $0 \leq s \leq n-m$ and $n=m + s + r$;

6) a factorization, LU for matrix B;

7) a factorization, R^TR for a Quasi-Newton approximation for the s x s matrix Z^TGZ;

8) a vector π satisfying $B^T\pi=g_b$;

9) the reduced gradient vector $h = g_s - S^T\pi$

10) convergence tolerances TOLRG and TOLDJ.

Considering the above discussion, a pseudo code for MINOS-1.0 can be written as follows:

Step 1: "Test for possibility of improving the current solution,"

1.1) If ($||h|| >$ TOLRG), go to Step 3.

Step 2: "Identify the nonbasic variable x_q which its corresponding lagrangian multiplier λ has the largest magnitude among all lagrangian multipliers, then add this variable to the group of superbasic variables."

2.1) Computer $\lambda = g_n - N^T\pi$;

2.2) If ($||\lambda_q|| >$ TOLDJ) for at least one of the lagrangian variables λ_q go to Step 2.3. Otherwise, print the current solution as an optimal solution and terminate the process, which means that the Kuhn-Tucker conditions for optimality at current solution x are satisfied;

2.3) Select $\lambda_{q_1} < -TOLDJ$, $\lambda_{q_2} > TOLDJ$ if possible;

2.3.1) Set $q = q_1$ or $q = q_2$ corresponding to $||q|| = \max(||q_1||, ||q_2||)$ if there exists such q_1 and q_2. Otherwise, set q to the existing one;

2.3.2) Add a_q to the matrix S and x_q to the group of superbasic variables;

2.3.3) Add λ_q to the vector of reduced gradient h;

2.3.4) Add an appropriate column to R;

2.3.5) Increase s by 1.

Step 3: Compute the search direction p.

3.1) Compute P_s by solving $R^T R P_s = -h$;

3.2) Compute P_b by solving $\lambda P_b = -SP_S$'

3.3) Set $P = (P_b \ P_s \ 0)T$.

Step 4:

4.1) Compute $\alpha\max > 0$ such that $x + \alpha P$ is feasible for all $0 \leq \alpha \leq \alpha\max$;

4.2) If $\alpha\max = 0$, go to Step 7, "i.e., if x is the only feasible point along the vector $x = \alpha P$, modify the basic or superbasic variables by deleting one of the variables which are in one of their bounds."

Step 5: Compute the maximum of F(x) along the search direction p.

5.1) Compute a^* such that $0 < a^* < \alpha\max$ and $F(x + a^*P) = \max F(x + \alpha P)$; $0 < \alpha < \alpha\max$;

5.2) Set x to $x^o + a^*P$ and compute F and g at the new point x.

Step 6: "Compute the reduced gradient $h = Z^T g = (-B^{-1}S \ I \ o)g$."

6.1) Compute π by solving $U^T L^T \pi = g_b = B^T \pi$;

6.2) Compute the reduced gradient vector at the new point $x + a^*P$ using $\bar{h} = g_s - S^T p$;

6.3) Update R, using a*, P_S and $\bar{h}$ - h;

6.4) change h to $\bar{h}$;

6.5) If a* < αmax go to Step 1; i.e., none of the basic and superbasic variables have hit their bounds, we use the correct superbasic variables as driving force to improve the objective function.

Step 7: Since a* = αmax = 0, there exists a basic or a superbasic variable x_j (o<j<m + s) that has reached one of its bounds.

7.1) If x_j is a nonbasic variable, go to 7.2. Otherwise, go to 7.1.1.

7.1.1) Exchange the J-th column of B with an appropriate column of S, say the q-th of S to keep B nonsingular

7.1.2) Update L, U, R, and p using the recent change in B. Compute the new reduced gradient vector h using h = $g_S - S^T\pi$ and then go to Step 7.3.

7.2) Since a superbasic variable has hit one of its bounds, it must become a nonbasic variable. Set q = J - m;

7.3) Delete the q-th column of S and R, restore R to triangular form and set s = s - 1, then go to Step 1.

SPECS Subroutine

This subroutine reads all input parameters from ISPECS (input specifications) file onto ISCRCH file.

SPECS 2 Subroutine

The purpose of this subprogram is to translate the input specification parameters given in ISCRCH file to machine code language.

MPSIN Subroutine

This subroutine reads all required data given in MPS file into the working space array Z and saves their locations in Z for future use. Use of MPS format specification is required in putting the data into MPS file.

Subroutine DRIVER

This subroutine incorporates the revised simplex method, the Reduced gradient method, a Quasi-Newton approximation method for reduced-Hessian matrix on LU factorization technique with some convergence testing criteria to solve a given problem. In the solution process, it saves the basis and the solution into BASIS and SOLUTION files iteratively, according to the values of their corresponding input parameters, given in SPECS file.

Subroutine REPORT

The purpose of this subroutine is to write the final report for the user of the system (MINOS-1.0). This subroutine has access to SOLUTION and BASIS files. The subprogram prints the solution point and the state of its components (basic, superbasic, nonbasic), the objective value at the optimal solution, the total number of required iterations. It also prints error messages if the computed sequence of solutions does not converge or if the objective function is unbounded in the given solution space.

GRG-2 Code

The main concepts of Generalized Reduced Gradient (GRG) software for nonlinearly constrained problems go back to the years 1963-1965 [1, 115]. In 1963, Graves and Wolfe proposed the Reduced Gradient (RG) method for nonlinear problems having just linear constraints; later, in 1964-65, J. Abadie extended the RG method to nonlinearly constrained problems. While some preliminary numerical experiments were showing some success for the new method, the first software was written by J. Abadie and it was ranked highest among about 30 methods which were used in the Colville Study [20] of 1968.

The outcome of the Colville Study about the code was an encouragement for its author to continue his effort to increase the robustness, accuracy, and speed of the code. The result of this effort was presented in 1968 as new version [2]. In the recent version, J. Abadie used the Fletcher-Reeves conjugate gradient method [36]. The new code, without having some antidegeneracy procedure, became first in the Colville Study [21] of 1970. In 1971, J. Ciugou wrote a new version containing some antidegeneracy procedures; since then, both codes were used with some success and failure. The failure led to modifications which have increased the size and complexity of the code. A new code was written by D. R. Heltes and Littschwager [52] in 1973. This code was named GRG73.

For giving correct values to the various parameters and tolerances, users of the last three codes needed some knowledge of GRG methods and numerical analysis. Since this requirement made the use of codes by people outside the field more difficulty, J. Abadie wrote a new version (GRGA) in 1975 [2] which had not only all the advantage so fits predecessor, but all the advantage of being easy to use for people outside the field.

The result of the joint effort of L. S. Lasdon and A. D. Waren from "Generalized Reduced Gradient Software for Linearly and Nonlinearly Constrained Problems" [59] of 1980 and "Design and Testing of a GRG Code for Nonlinear Optimization" [58] of 1978 named GRG-2 which is available in

FORTRAN IV. This code is a combination of a main program and about fourteen subroutines. Figure 7 shows a diagram of its major subroutine structure. A flow chart for the code is given in Figure 7.

In what follows, the major subroutines of GRG-2 are described.

Main Program

Main program first provides a working area for the entire program through dimensioning a one-dimensional array, namely Z (10,000). Because of this action, 10,000 spaces from the main memory of the computer will be devoted to storing all available information about the given problem. Also, the computed information during the process of the program will be stored in the working area, Z (10,000). After dimensioning Z (10,000), the main program calls the GRG subroutine to use the GRG algorithm.

GRG Subroutine

This subroutine first calls the subroutine SETUP to calculate the addresses of all data in the working area, Z (10,000). Next it calls the subroutine DATAIN to read all input information. After reading the necessary information for the given problem, it calls the subroutine GRGITN to solve the given problem. Finally, it calls the subroutine OUTERS to print the final computed results, i.e., the optimal solution.

Subroutine GCOMP

Given the current point x^k, the objective function, and the constraint functions, this subroutine checks the feasibility of x^k through evaluation of the constraints. If any constraints are violated, it calls the PHASE I

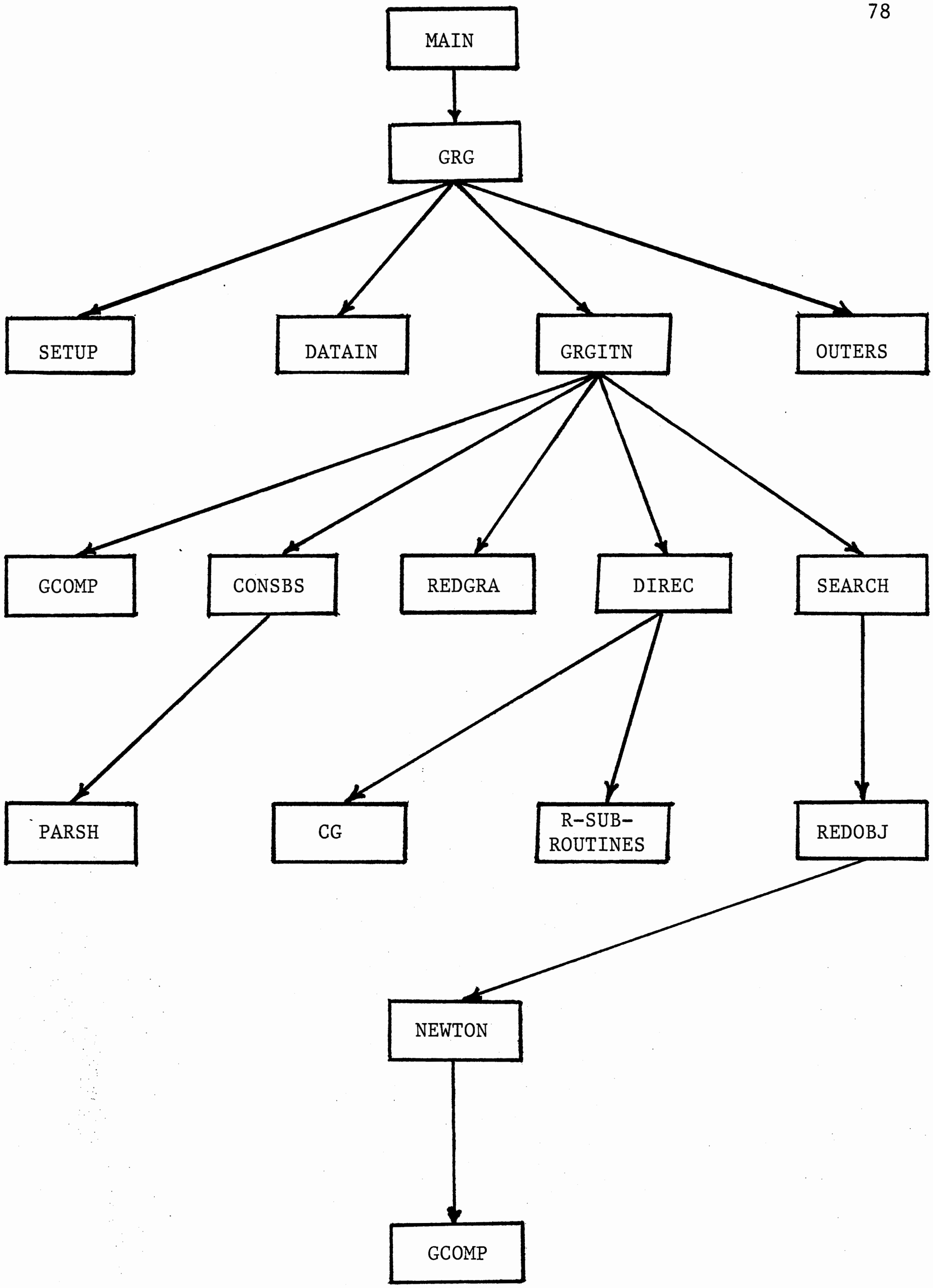


Figure 7. Subroutine Structure of GRG

subroutine for providing a feasible point, and then it evaluates the objective function. Otherwise, the objective function will be evaluated and saved first.

Subroutine PHASE I

When the sum of the absolute values of the constraint violations of the given point x^k is given, this subroutine minimizes the function of sum subject to the binded constraints at the given point. Finally, it assigns the computed result to x^k as a feasible point for the original problem.

Subroutine PARSH

This is a user supplied subroutine and is designed to compute the gradient of the objective function, g, and partial derivatives of the constraint functions at the given feasible point. While it is computing partial derivatives, it also saves them into a two dimensional array named for GRAD. But, if evaluation of the partial derivative function becomes expensive, then the process of forward difference approximation, which is built into the code, will be used to compute the Jacobian matrix, $J(x^k)$.

Subroutine REDGRA

This subroutine computes Lagrange multiplier vector π and reduced gradient vector, Rg (the gradient of the reduced objective function) through using the formula,

$$\pi = g_b(x^k)\mathrm{BINV}$$

$$Rg = g_{nb}(x^k)-B_{nb}\pi^T,$$

where BINV, g_b and g_n are made available by subroutine CONSB and PARSH, respectively.

Subroutine CHECK

This subroutine checks the optimality conditions for the given feasible point x^k. The process of checking is composed of two tests, and the current feasible point x^k will be considered an optimal solution for the given problem if either of these two tests are satisfactory. The first test checks the Kuhn-Tucker conditions (given in Chapter I) in some given range of EPSTDP, with the default value of 10^{-4}. The second test checks to find if the amount of change for the objective function is less than EPSTOP for NSTOP consecutive iterations. The default value of NSTOP is considered to be 3. If the result of any of the two tests become true, the subroutine will send the control to the OTHER subroutine for printing the results and then terminates the program. Otherwise, the subroutine DIREC will be called to compute a feasible search direction, namely P.

Subroutine DIREC

Given nn, x^K_{nb}, and $(Rg)_{nb}$ which are the number of nonbasic variables, nonbasic variables, and nonbasic components of the reduced gradient vector, respectively. If the number of nonbasic variables nn is less than nq, a user supplied number, the subroutine calls the QUASI subroutine to compute the search direction P_k. Otherwise, it computes P_k by calling the CG (conjugate gradient) subroutine.

At this state, the subroutine computes an $\alpha^* > 0$ such that $\alpha^* = \max\{\alpha | x_{nb}^{k} + \alpha P$ is feasible$\}$, if $\alpha^* = 0$, the subroutine CONSB will be called for a new choice of x_{nb}^{k}. Otherwise, the subroutine SEARCH will be called to find the maximum of the reduced function over the interval $[x_{nb}^{k}, x_{nb}^{k} + \alpha^* P_k]$. Assuming the minimum of the reduced function happens at $x_{nb}^{K} + \bar{\alpha} P_K$ with $0 \leq \bar{\alpha} \leq \alpha^*$, the new value of x_{nb}^{K+1} is set as the following:

$$x_{nb}^{K+1} = x_{nb}^{K} + \bar{\alpha} P_K.$$

Next, the tangent vector, v_K, corresponding to P_K, is computed using the formula:

$$v_K = B^{-1} B_{nb} P_K.$$

Having the tangent vector v_K available, the subroutine CHRUZR is called to compute the largest value β which can be taken in the direction $(\alpha_K p_k)$ before any basic variable violates its bound. If β becomes smaller than EPFFS, subroutine CHUZO is called to replace one of the basic variables which is causing a degeneracy case with a superbasic variable. After the time at which a pivotal operation is performed for adjusting the basis elements, the subroutine REDGRA again will be called to update Lagrange vector π and the reduced gradient vector Rg.

Subroutine REDOBJ

Given x_{nb}^{K}, α_K, and the search direction P_K, this subroutine computes the reduced object function $F(x_b(x_{nb}^{K}), x_{nb}^{K} + \alpha_k P_k)$. It also calls the subroutine NEWTON to solve the system of binding constraints:

$$h_i(x_b, x_{nb}^{K+1}) = 0, i, \in \text{BINDC}$$

where BINDC is the index set of the binding constraints and

$$x_{nb}^{k+1} = x_{nb}^{k} + \alpha P_{k}.$$

Subroutine GRGITN

This subroutine checks the feasibility and optimality of the current point x^k through calling GCOMP. It calls CONSBS to compute basic, nonbasic variables, the basic inverse, and BINDC, i.e., the index set of binding constraints. It calls REDGRA to compute the reduced gradient and the Jacobian matrix $J(x^k)$. It also calls the one dimensional search subroutine SEARCH to compute the maximum value of the reduced function, as well as the maximum of the objective function overall, as Figure 7 shows. This subroutine controls main iterative loop.

Subroutine SEARCH

Given x_{nb}^k, $\alpha*$, P_k and the reduced objective function $F(xb(x_{nb}), x_{nb})$, the subroutine SEARCH computes a maximum point for $F(x_{nb})$ in the interval $\{x_{nb}^k, x_{nb}^k + \alpha * P_k]$ using a quadratic fitting algorithm. This algorithm searches for three values, α_1, $\alpha 2$, $\alpha 3$, satisfying

$$0 < \alpha\ , \alpha 2 < \alpha 3 \le \alpha*,$$

and

$$F(x_{nb}^K + \alpha_1 P_K) \le F(x_{nb}^K + \alpha_2 P_K) \le F(x_{nb}^K + \alpha_3 P_K),$$

where α^* and P_k are the stepsize and the search direction computed in subroutine DIREC. Then having α_1,$\alpha 2$, $\alpha 3$, a quadratic function is passed through $\alpha 1$,$\alpha 2$,$\alpha 3$, and its maximum in the interval [$\alpha 1$, $\alpha 3$] is taken as an approximation for $\bar{\alpha}$. Next, x_{nb}^{k+1} is set to its new value, using the formula

$$x_{nb}^{K+1} = x_{nb}^{K} + \alpha P_K,$$

and the subroutine REDOBJ is called to compute the reduced objective function $F(x_{nb})$. The search for α is terminated if REDOBJ produces an improved point at which either a super/basic variable is meeting its bound.

Then the NEWTON subroutine is called to solve the system of binding constraints

$$h_i\ (x_b(x_{nb}),\ x_{nb}) = 0,\ i \in \text{BINDC},$$

for the basic variables x_b, using x_b^k as an initial solution.

If Newton algorithm used in NEWTON subroutine fails to converge, and an improved feasible point has already been found, the search for x_b is terminated and the subroutine OUTER is called to print the request results. Otherwise, the step size β_k is halved and the NEWTON algorithm is restarted.

Subroutine NEWTON

Given P_k, x_b^k, x_{nb}^{k+1},$\bar{\alpha}_k\beta_k$, and the system of binding constraints of the current feasible point x^k, the subroutine solves the binding system,

$$h_i(x_b,\ x_{nb}^{K+1}) = 0,\ i \in \text{BINDC},$$

over the interval $[x_b, x_b^k + \beta_X \alpha_K P_K]$ considering x_b^k as an initial solution for the system. Using the formula

$$x_b^{K+1} = x_b^K - J_b(x_b^K)h(x_b^K),$$

where $J_b(x_b^k)$ is the Jacobian of the binded system evaluated at the recent basic variables x_b^k.

After computing x_b^{k+1}, first the convergence test for the Newton algorithm is checked, and then the subroutine GCOMP is called to check the optimality tests for the original problem at the most recent computed point $x^{k+1} = (x_b^{k+1}, x_{nb}^{k+1})$. If Newton algorithm fails to converge in six iterations and an improved solution to the given problem has been found, the Newton algorithm is terminated, and the subroutine OUTER is called to print the requested results. Otherwise, the step size β_K is halved and if the new β_K is not smaller than EPFFS (i.e., check nondegeneracy), Newton algorithm is restarted. Otherwise, subroutine CHUZO is called for new x_b^k, x_{nb}^k.

Finally, as soon as convergence test is satisfied, GCOMP is called to check the optimality tests.

Subroutine QUASI

Given nonbasic variables, x_{nb}^k, nonbasic components of the reduced gradient vector, $(Rg)_{nb}$, and a symmetric positive definite matrix S_k, the subroutine computes a search direction P_k, using the formula,

$$q_K = -S_K(Rg)_{nb},$$

and then it calls the subroutine SEARCH to minimize the reduced function, $F(x_{nb})$ with respect to $\alpha \geq 0$ to obtain:

$$x_{nb}^{K+1} = x_{nb}^{K} + \alpha_k q_k ,$$

$$P_k = \alpha_k q^k .$$

Then it uses the variable metric method to update the Hessian matrix of the reduced objective function. This method in GRG-2 updates an approximation to the reduced Hessian $\partial^2 F / \partial x_{nb}^2$ rather than its inverse. At a typical step, the reduced Hessian S_k, is updated by the sum of two symmetric rank one matrice, using the complimentary DFP formula.

$$S_{K+1} = S_K + (\bar{g}_k \bar{g}_k^T) / (\bar{g}_k^T P_k) + (g*_k g*_k^T)/(g*_K g_k^T),$$

where S_{k+1} is an approximation for the reduced Hessian at the new point x_{nb}^{k+1},

$$\bar{g}_K = (Rg(x^{K+1}))_{nb} - (Rg(x^K))_{nb},$$

$$g*_K = Rg(x^K),$$

and P_k and q_k are vectors computed in earlier steps. Note that this subroutine maintains the approximation for the reduced Hessian in factorized form, i.e., as $R_k^T R_k$, where R_k is an upper triangular matrix.

Subroutine CG

This subroutine becomes active if the number of non-basic variables nn become greater than ng (i.e., updating the reduced Hessian is extensive, using the variable matrix method). The subroutine uses given nonbasic variables x^k_{nb} and nonbasic components of the reduced gradient vector, $(Rg(x^k))_{nb}$, at the first time to compute a search direction q_k, using the formula

$$q_K = -(Rg(x^K))_{nb},$$

and calls the subroutine SEARCH to maximize the reduced function, $F(x_{nb})$ with respect to

$$\alpha \geq 0 \text{ to obtain } x^{K+1}_{nb} = x^K_{nb} + \alpha\, q_k,$$
$$P_K = \alpha\, q_K.$$

After the first time, it uses one of the five variants of the conjugate gradient method (which are included in the subroutine) to obtain:

$$q_k = -(Rg(x^k))_{nb} + \beta_k p_{k-1},\ \beta_k = \frac{||(Rg(x^K))_{nb}||}{||(Rg(x^{K-1}))_{nb}||},\ \beta 0 = 0.$$

Then the subroutine SEARCH will be called to minimize the reduced function, $F(x_{nb})$ with respect to $\alpha \geq o$ to obtain:

$$x^{K+1}_{nb} = x^K_{nb} + \alpha_K q_K,$$
$$P_K = \alpha_K q_K.$$

The five variants included in this subroutine are: (1) Fletcher and Reeves [36], (2) Polak, E. [79], (3) Perry, A. [78], (4) 1-step version of the DFP, and (5) the complementary DFP formula. All of these methods follow same strategy for finding a new search direction except they offer their own formula for computing β_k.

MINOS-5.0 Code

In 1976-1977, Murtagh and Saunders [72] developed software to optimize a linear or nonlinear objective function F(x) satisfying some given conditions, by finding a point x which makes F(x) as close to $\pm\infty$ as possible. The name MINOS, which stands for "Modular In-Core Optimization System", was given to the code. MINOS was originally designed to solve problems from small unconstrained problems with or without nonlinear terms in their objective functions.

The satisfactory results of MINOS motivated Murtagh and Saunders [74] to extend their software to nonlinearly constrained problems as well. The result of their effort, "MINOS/AUGMENTED", was introduced [75] in 1978. This version of MINOS was designed to solve large-scale nonlinearly constrained problems whose objective and constraint functions are continuously differentiable.

MINOS-5.0 is an available software written in FORTRAN IV, designed to optimize unconstrained, linear, linearly constrained, and nonlinearly constrained problems whose objective and constraints functions are continuously differentiable. The code is a combination of two iterative processes, major and minor. To describe these processes, let us consider the following general problem:

$$\begin{aligned} &\text{maximize } F(x,y) = f(x) + c^T x + d^T y, && x\, eE^{n1},\ y\, eE^{n2} \\ &\text{subject to } h(x) + A_1 y = b_1, && b_1\ \varepsilon E^{m1} \\ &\qquad A_2 x + A_3 y = b_2, && b_2\ \varepsilon E^{m2} \\ &\qquad \ell \le \begin{bmatrix} x \\ \\ y \end{bmatrix} \le u \end{aligned} \tag{4.2}$$

where the matrices A_1, A_2, A_3 and the vectors c, d, b_1, b_2, μ and λ are constants. $F(x)\ \varepsilon C^2(\Omega)$, and the components of h(x) belong to $C^2(\Omega)$.

Assuming that x^k, an estimate for the nonlinear variables x, λ^k, an estimate of the Lagrangian multipliers vector λ and a scalar P_k for the penalty parameter P are given, the nature of the major and the minor processes can be described as follows:

In a typical step of the major process, a linearly constrained subproblem will be made out of the original problem (4.2). The subproblem will contain the original linear constraints, bounds, and a linear approximation of nonlinear constraints. This approximation can be written as,

$$\bar{h}(x,x^k) = h(x^k) + J(x^k)(x-x^k),$$

or briefly

$$\bar{h} = h_K + J_K(x-x^k),$$

and the subproblem can be written as,

$$\begin{aligned} &\text{maximize } F(x) + c^T x + d^T y - \lambda_K(\bar{h}-h) + \tfrac{1}{2}p_k(\bar{h}-h), \\ &\text{subject to } \quad A_2 x + A_3 y = b_2, \\ &\qquad\qquad \ell \le (x,y)^T \le u, \end{aligned} \tag{4.2*}$$

where the new objective function is named an augmented Lagrangian functions. Each major process will be followed by a minor process. In the minor process, an improved version of the original MINOS-1.0 will be used to solve the established subproblem (4.2*), with the original bounds, u and l, in effect.

As Figure 8 suggests, MINOS-5.0 is a composition of a main program which needs to be provided by its users, and several subprograms. The main program and the major subprograms of MINOS-5.0 are described briefly below.

Main Program

The main program provides the working space for the whole system through declaring a one dimensional array Z of length 10,000, then, by calling the subprogram MINOS-1.0, attempts to solve the given problem or problems.

Subroutine MINOS-1.0

The subprogram MINOS-1.0 first defines the SPECS, SCRATCH, READ and PRINT files by advocating the subroutine MIFILE, then calls MINOS-2.0 once for each problem found in the SPECS file. After completing each call to MINOS-2.0, the value of the parameter INFORM.SOLUTION process terminates if INFORM = -1 which means there is no problem left in SPECS to solve. Otherwise, MINOS-2.0 will be called again to solve the next problem.

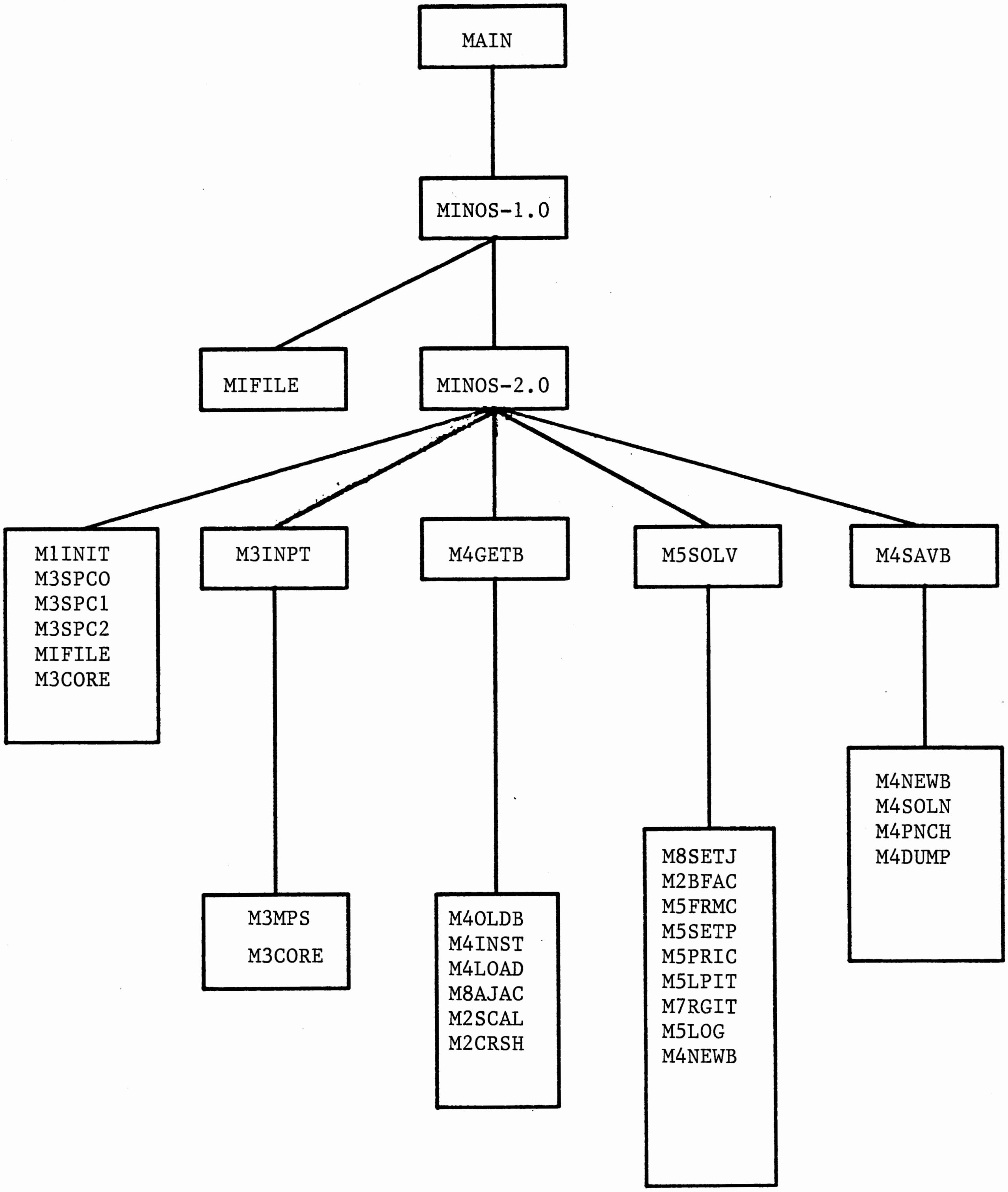


Figure 8. Subroutine Structure of MINOS-5.0
from
MINOS-5.0 USER'S GUIDE
TECHNICAL REPORT SOL 83 - 20

Subroutine MIFILE

This subroutine defines the global files: READ, PRINT, SCRATCH and SPECS. The united number for the READ and PRINT files could vary from machine to machine, but usually 5 and 6 are used for the READ and PRINT files respectively.

Subroutine MINOS-2.0

This subprogram performs the following:

1) defines the version of MINOS in use, the authors' names, the date that version was completed and the institute through which the coding was done, calling the subroutine MLINIT;

2) sets default values for those input parameters that can be altered through the SPECS file by calling the subroutine M3SCP0;

3) reads all necessary input information (parameters and date) for a given problem from SPECS file into SCRATCH FILE by calling the subroutine M3SCP1;

4) extracts the required parameter values from the SCRATCH file by using the subroutine M3SCP2;

5) defines the files needed for the given problem by calling the MIFILE subroutine M3SCP2;

6) allocates sufficient space for the rows, columns, initial solution and other variables from the working space Z (10,000) by using the subroutine M3CORE;

7) reads the MPS data from IMPS file through calling the subroutine M3INPT;

8) inputs a basis and saves it into BASIS file by calling the subroutine M4GETB or calls the subroutine CRASH to establish a basis;

9) calls the subprogram M5SOLV to solve the given problem while some initial estimates x_o, y_o, λ_o, $P_o > 0$ and a convergence tolerance $E_C > 0$ are provided for the subprogram M5SOLVE;

10) saves the result of M5SOLV, prints this result according to the value of the input parameter MSOLN:

MSOLN= 0 means not to print
= 1 means print if optimal, infeasible or unbounded
= 2 means print,
= 3 means print if there is an error condition,

and finally terminates the process for the given problem or calls the subroutine M5SOLV to solve a new subproblem.

Subroutine M3INPT

This subroutine reads the given data from IMPS file and makes it usable for the subroutine M5SOLV by using subroutines M3MPS and M3CORE.

Subroutine M3MPS

This subprogram converts the data format from MPS format into machine code format and puts it into appropriate places in the work space array Z (10,000) by using the subroutine M3CORE.

Subroutine M3CORE

This subroutine allocates sufficient storage spaces to the given subarrays such as variables, boundaries, Lagrangian multipliers and objective coefficients from the working space array Z (10,000) which has been defined in the main program.

Subroutine M4GETB

This subprogram performs the following:

1) copies a basis from OLDB file into IPRINT file in a compact form by calling the subroutine M4OLDB;

2) reads the list of basis names, their states and their values from the file IPNCH which is produced by subroutine M4PNCH by calling the subroutine M4INST;

3) reads the list of row and column names, their states and their values through calling the subroutine M4LOAD;

4) computes the Jacobian by using the provided users subroutine, or the numerical finite differences by using the subroutine M8AJAC, also puts the Jacobian in A;

5) uses an iterative method which is derived from the routine written by Robert Fourer to scale the linear constraints and variables through calling the subroutine M2SCAL;

6) finally computes a triangular basis from the columns of [AJ] by using the subroutine M3CRSH.

Subroutine M5SOLV

This subroutine solves a given problem (while some estimates x^k, y^k, λ^k, P_k for the nonlinear, linear variables x, y, the Lagrangian multipliers vector λ and the penalty parameter P are provided for its users) through performing the following:

1) producing a linearly constrained subproblem through linearization of the nonlinear constraints of the given problem by calling the subroutine M8SETJ;

2) computing an LU factorization of the basis matrix produced by the M4GETB subroutine through calling the subroutine M2BFAC;

3) solving equation $B^T PI = g_b$ for PI by calling the M5FRMC and M5SETP subroutines;

4) finding the eligible variable to enter the basis and the eligible basic to leave the basis through calling the M5PRIC subroutine;

5) exchanging the eligible nonbasic and basic variables, updating B, h, U and the gradient vector g by calling the M5LPIT subroutine;

6) executing the reduced gradient algorithm to find a solution for the subproblem produced by the M8SETJ subroutine and testing the given optimality conditions of the original problem for the computed solution by calling the M7RGIT subroutine; if the optimality test fails, then all of these six steps will be repeated;

7) printing out the optimal solution with the state of variables and the number of major and minor iterations required by calling the subroutine M5LOG:

8) copying the most recent basis into the BASIS file by calling M4NEWB.

Subroutine M4SAVB

This subprogram's action is determined by the value of the parameter MODE in the following manner:

1) if MODE=1, first the subroutine saves the most recent basis on the JNEWB file, next it unscales the solution x and expands x by taking the

slack variables as the end tail of x, finally saves the SOLUTION, PUNCH and DUMP files by calling M4NEWB, M4SOLN, M4PNCH and M4DUMP subroutines respectively;

2) if MODE=2, then it prints the solution according to the value of the input parameter MSOLN as it follows:

2.1) MSOLN = 0, then it does not print the solution,

2.2) MSOLN = 1, then it prints the solution if it is optimal, infeasible or unbounded;

2.3) MSOLN = 1, then it prints the solution;

2.4) MSOLN = 2, then it prints the solution only if an error condition is founded.

Subroutine M4NEW

This subroutine copies the BASIS file on the INEWB file in a compact form.

Subroutine M4SOLN

This subroutine expands the solution X or prints the required information if the parameter MODE=1 or 2 respectively. In the latter case, it still checks the parameter MSOLN to print the output accordingly.

Subroutine M4PNCH

This subroutine copies a list of basis names, states, and their values on IPNCH file.

Subroutine M4DUMP

This subroutine saves the basis names on IDUMP file using a format specification which is compatible with MPS specification.

Computation Results for MINOS-1.0

MINOS-1.0 has undergone extensive testing. Several difficult problems such as the PILOT Energy Model [74], OIL Refinery Investment Model, Energy Submodel and Chemical Equilibrium problems [74] have been solved using MINOS-1.0 with satisfactory results. Computational results of MINOS-1.0 on 10 problems [20,54] are reported in tables (1,2) (readers interested in the statement of problems are referred to [20,54]). These problems are solved on Burroughs B6700 and IBM 370/168.

Computational Results for GRG-2:

The results for GRG-2 code on 24 problems specified in [54] are given in table (3). All of these problems were solved on an IBM 370/145 at Cleveland State University. In the table (3) the ratio of the total number of iterations of the quasi-Newton method to the number of calls to the subroutine NEWTON is shown by Newton-Average.

GRG-2 was successful in finding at least a local minimum for each problem. In all except problems 6 and 13, the final objective values founded by GRG-2 using the recommended initial points specified in [20] either matched the solutions specified in [20], to at least one part in one thousand, or were more qualified than those given in [20].

TABLE I

SOLUTIONS OF PROBLEMS 1-2, 4-8 ON BURROUGHS B6700

PN[1]	Row	Column	NZE[2]	NV[3]
1	10	5	47	5
2	8	16	80	16
4	12	100	147	100
5	10	24	240	24
6	74	83	529	15
7	95	200	504	24
8	324	425	1,404	91

PN[1]	I[4]	E[5]	FNS[6]	(Specs.)	STR[7]
1	8	9	1	0.63	0.008
2	15	16	3	1.50	0.018
4	133	296	18	48.30	0.580
5	8	8	14	1.65	0.019
6	80	40	3	37.03	0.450
7	103	72	0	42.43	0.510
8	348	215	0	538.30	6.480

[1] Problem Number
[2] Nonzero Elements
[3] Nonlinear Variables
[4] Includes Phase 1 Iterations
[5] Final Number of Superbasics
[6] Evaluations of f(x),g(x)
[7] Standard Time Ratio

TABLE II

SOLUTION OF PROBLEMS 3-4, 9-10
ON IBM 370/160

PN[1]	Rows	Columns	NZE[2]	NV[3]
3	16	45	99	45
4	12	100	147	100
9	356	1,134	4,180	0
10	320	679	2,519	44

PN[1]	I[4]	E[5]	FNS[6]	(Specs.)	STR[7]
3	103	452	24	2.9	0.74
4	139	355	18	2.6	0.66
9	539	0	0	33.3	8.50
10	350	902	26	26.9	6.90

[1] Problem Number
[2] Nonzero Elements
[3] Nonlinear Variables
[4] Iterations
[5] Evaluations of f(x),g(x)
[6] Final Number of Superbasis
[7] Standard Time Ratio

TABLE III

RESULTS OF SOLVING HIMMELBLAU PROBLEMS

PN[1]	BFVR[2]	BFUGRG[3]	FE[4]	GE[5]
1	1.39300	1.39300	25	4
2	0.00000	6.0 X 20-14	177	25
3	58.90300	58.90300	169	17
4	-47.76100	-47.72000	77	17
5	961.71500	916.71500	39	7
6	-1910.36100	-1865.98000	229	50
7	-1162.04000	-1162.03000	130	17
8	0.00000	1.0 X 10-7	255	47
9	0.00750	0.00750	89	19
10	-32.34900	-32.34900	63	9
11	-30,665.50000	-30,665.50000	16	6
12	-1.90500	-1.90500	48	6
13	-5,280,254	-5,280,338	19	6
14	255,303.50000	255,303.50000	118	19
15	8,927.59000	8,927.57000	172	17
16	-0.86600	-0.86604	244	18
17	-45.77800	-45.77800	32	5
18	32.38600	32.34900	564	42
19	-244.90000	-244.90000	162	37
20	0.05700	0.05566	200	31
21	0.00000	0.00000	6	2
22	0.01560	0.01560	8	7
23	-1,732.00000	-1.733.30000	239	41
24	1.00000	1.00000	26	4

PN[1]	ODS[6]	NA[7]	ET(sec)[8]	CST[9]
1	3	0.54	0.10	0.0013
2	25	0.00	0.44	0.0057
3	16	3.94	1.04	0.0130
4	16	0.00	3.81	0.0490
5	6	0.50	0.24	0.0031
6	49	0.00	227.26	2.9200
7	16	0.82	2.75	0.0350
8	43	0.00	1.32	0.0170

TABLE III (Continued)

PN[1]	ODS[6]	NA[7]	ET(sec)[8]	CST[9]
9	18	0.00	18.26	0.2350
10	9	0.00	1.53	0.0200
11	5	1.00	0.21	0.0027
12	5	1.71	1.01	0.0130
13	5	0.33	0.16	0.0021
14	18	0.38	2.93	0.0380
15	16	1.75	2.41	0.0310
16	17	2.28	4.39	0.0560
17	5	0.00	1.72	0.0220
18	41	2.96	18.65	0.2400
19	36	0.00	38.55	0.4950
20	29	1.00	10.85	0.1390
21	1	0.00	1.90	0.0250
22	6	0.00	0.28	0.0036
23	40	0.03	570.37	7.3280
24	3	1.17	0.08	0.0010

[1] Problem Number
[2] Best Function Value Reported
[3] Best Function Value with GRG
[4] Function Evaluation
[5] Gradient Evaluation
[6] One Dimensional Searches
[7] Newton Average
[8] Execution Time (sec)
[9] Colville Standard Time

For problem number 6, starting with the initial point specified in appendix II, GRG-2 found an objective value with 0.023230 relative error. Using a different starting point (X=0), GRG-2 reached an optimal value with 0.000074 relative error.

In problem number 13, starting with the initial point suggested in [20], GRG-2 attained an objective value with 0.042065 relative error. Using X_i=0, i≠4 X_4=2000 for the initial point, GRG-2 reached a minimum value with 0.003931 relative error.

Computational results for MINOS-5.0:

This software as its original code MINOS-1.0 has undergone extensive testing and has attained successful results in solving problems such as Electric Power [76], Air Pollution Control [74], Economic Growth, Optimal Control and Launch Vehicle Design [75]. The results for MINOS-5.0 on 12 problems (readers interested in the statement of problems are referred to [75]) are reported in tables (4) and (5). In solving these problems, the following parameter values were used in SPECS file:

LINESEARCH PARAMETER	ETA = 0.1
RADIUS OF CONVERGENCE	EC = 0.01
RAW TOLERANCE	ER = 10-6
MINOR ITERATIONS LIMIT	= 40

Evaluation of Codes

In evaluating software, it has historically been the case that a variety of test problems are solved using codes and summary statistics are presented for user's evaluation. In testing the system with standard test problems, the presence of some criteria for measurement is necessary. Criteria such as

TABLE IV

SOLUTION OF PROBLEMS 1-8 ON CDC CVBER 70

PN[1]	NC[2]	LC[3]	NV[4]	LV[5]	
1	15	0	5	10	
2	3	0	5	0	
3	7	0	3	0	
4	3	0	5	0	
5	91	0	79	0	
6	10	12	25	0	
7	13	4	20	0	
8	11	8	16	0	
PN[1]	**MI[6]**	**TI[7]**	**TFE[8]**	**ET[9]**	**ST[10]**
1	4	41	65	3.58	0.046
2	3	5	7	0.97	0.012
3	3	10	54	2.05	0.003
4	4	18	26	1.53	0.021
5	5	100	69	38.90	0.500
6	3	26	60	8.13	0.104
7	27	91	147	20.10	0.250
8	7	55	66	3.56	0.457

[1] Number of Problem
[2] Number of Nonlinear Constraints
[3] Linear Constraints
[4] Nonlinear Variables
[5] Linear Variables
[6] Major Iterations
[7] Total Iterations
[8] Total Function Evaluations
[9] Execution Time
[10] Colville Standard Time

TABLE V

SOLUTION OF PROBLEMS 9-12 ON IBM 370-168

PN[1]	NC[2]	LC[3]	NNV[4]	NLV[5]
9	3	0	5	0
10	3	0	5	0
11	100	100	202	100
12	100	200	300	0

PN[1]	MI[6]	TI[7]	TFE[8]	ET[9]	CST[10]
9	9	47	84	?	?
10	12	92	183	?	?
11	6	247	203	11.56	2.98
12	11	366	859	34.30	8.98

[1] Number of Problem
[2] Nonlinear Constraints
[3] Linear Constraints
[4] Number of Nonlinear Variables
[5] Number of Linear Variables
[6] Major Iterations
[7] Total Iterations
[8] Total Function Evaluations
[9] Execution Time (Secs.)
[10] Colville Standard Time

input, output, ease of use, problem solving ability, efficiency and reliability features of the three codes will be used in gathering our statistics. The brief description of these features can be given as the following:

1. Input features

1.1 Ability of assigning names to variables and constraints;

1.2 Ability of identifying the types of function, variables independent of their order;

1.3 Ability of computing the derivatives using a numerical method to compute the gradients in the absence of appropriate user's subroutines;

1.4 Ability of dividing all inputs into sections, each section having a heading and an END statement;

1.5 Ability of using default values for all controllable program tolerances and parameters.

2. Output features

2.1 Ability of printing out the requested results in tabular form;

2.2 Ability of multi-printing to improve the debugging procedures;

2.3 Ability of dumping and restarting for recovery from error conditions;

2.4 Ability of producing a periodic detailed printout for every KH iteration;

2.5 Ability of checking any user provided derivative computation.

3. Ease of Use features

3.1 Well documented;

3.2 Easy to use as part of a larger system;

3.3 Dynamic storage allocation;

3.4 Portable, requiring minimal modification to run on different machines.

4. Problem Solving Features

4.1 Ability to solve unconstrained problems (with free or bounded variables efficiently);

4.2 Ability of handling nonlinear equality constraints efficiently;

4.3 Ability of generating a sequence of improved feasible points, starting from a feasible or a non-feasible point;

4.4 Ability of handling problems from small to large sparse efficiently.

Codes for MINOS-1.0, GRG-2, and MINOS-5.0 have been discussed in this chapter. Research on the use of MINOS_1.0, GRG-2, shown [58,59] that MINOS-1.0 and GRG-2 incorporate all of the listed input and output features, while in terms of solving abilities only 4.2 feature is absent for MINOS-1.0 and only feature 4.4 is not present for GRG-2, MINOS-5.0 is a robust and an efficient software that incorporates all input, output and problem solving features. In order to make a fair conclusion for GRG-2 and MINOS-5.0, even though the results of MINOS_5.0 are very encouraging, as Lasdon L.S and Waren, and Murtagh and Saunders the authors of GRG-2 and MINOS-5.0 suggested in their investigations [58,77] more research needs to be done on both of them.

CHAPTER V

SUMMARY AND RECOMMENDATIONS

Summary

This study has focused on four optimization algorithms for small to medium size nonlinear programming problems, large-scale nonlinear programming problems with linear constraints. These algorithms are RG (Huard's version), LSLC (MINOS-1.0), GRG-2 and MINOS-5.0. Robinson's algorithm was also described fully in Chapter III because of its use in MINOS-5.0. Flowcharts for showing some of the complexities that arise in the process of translating the mathematical algorithms into their implementations were identified. Also, implementation for MINOS-1.0, GRG-2, and MINOS-5.0 have been described and evaluated in Chapter IV. Research on the use of GRG for the first class of problems has been under way since 1972 [1], and the reported results as shown in Table III would indicate that GRG-2 is one of the best methods for solving such problems. Research on the use of RG for the second class of problems has also been under way for the last decade [46]. L. S. Lasdon, the author of GRG-2 in Numerical Optimization, 1984 [46], says that, in his opinion, "the GRG-2 is one of the best general purpose nonlinear optimization codes now available." Also, according to Murtagh and Saunders' opinion which are partially based on the reported results in tables I and II, it may be said that MINOS-1.0 is a robust, efficient, thoroughly tested system for such problems, and a comparable system seems to be lacking in the literature. For nonlinear constrained problems, even the preliminary results using MINOS-5.0 as shown in table IV and V are encouraging, but its authors Murtagh and Saunders believe that for a better judgment more testing is needed. The largest nonlinear constrained optimization problem solved by MINOS-5.0 has come from an energy production model concerned with air pollution control [76]. This problem involved about 850 constraints and 4,000 variables. The objective function was nonlinear in 225 of the variables and 32 of the constraints were quadratic in 778 of those variables.

Some statistics follow for the solution of this problem (all parameters were used according to their default values, except that the MAJOR ITERATIONS limit was set to 100):

Major iterations	13
Minor iteration	5626
Objective function and its gradient evaluation	5955
Active nonlinear constraints of optimum	12
Superbasic variables at optimum	18
CPU time on a DEC VAX 11/780	63 min.

Since practicality of an optimization method as it has been expressed by, Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright, in Practical Optimization, [45] depends upon the existence of an implementation and a significant amount of reliable computation performed; also, since as Richard L. Burden and Douglas Faires said in the Numerical Analysis [19], the efficiency of an optimization code depends upon its ease of implementation, the choice of the appropriate software for approximating the solution to an optimization problem is influenced significantly by the advancements in the computer technology. About twenty years ago, before the widespread use of digital computers, codes like MINOS-1.0, GRG-2, and MINOS-5.0 could not be reasonably implemented. Since that time, however, the advances in computing technology not only have made these codes reasonably implementable, but also have made them very attractive. At present, the limiting factor generally involves the amount of computer storage requirements for the code, however, the cost factor associated with a large amount of computation time is, of course, also important.

Recommendations

It is a truism that no single algorithm can be expected to do uniformly better than all others in such a diverse field as nonlinearly constrained optimization. Tables I, II, IV, and V would seem to indicate that MINOS-5.0 is reasonably efficient on small, highly nonlinear problems, and more importantly, it can be considered an advancement in the development of general purpose software for large-scale optimization. But, according to its author's experience (Bruce A. Murtagh and Michael A. Saunders) [76], the convergence of MINOS-5.0 is not guaranteed when the starting point is chosen arbitrarily. It is also a truism that a mathematical algorithm cannot be treated as practical unless an implementation has been produced and a significant amount of reliable computations performed. Thus, as Walter Murray, Michael A. Saunders, and Margaret H. Wright said in Numerical Optimization [46], research on optimization methods necessarily overlaps heavily with the development of software. Since the development of numerical software, much has been said about the complexities that arise when translating any mathematical algorithm into an implementation (Cody J.

Cowell). Although the majority of investigators in optimization are aware of such issues, but according to the Gill, Saunders, Wright and Murray opinion, the effect of implementation on methods is much less widely understood and discussed. In fact, the relationship between algorithms and software is sometimes explained simply by defining an implementation as a concrete realization of theoretical algorithm. It is clear that this statement does not describe the critical influence that implementation may have on theoretical algorithms.

It is hereby recommended that future research may include an investigation of the following:

1) An algorithm for adjusting the penalty parameter between subproblems to make the convergence of MINOS-5.0 more promising.

2) An algorithm for adjusting the initial point x when MINOS-5.0 fails to converge.

3) Comparison of MINOS-5.0 with other large-scale algorithms such as successive linear programming (SLP) and successive quadratic programming (SQP).

4) The effect of implementation on mathematical algorithms. This is the main topic of [46], which should be consulted as an excellent elaboration on this question.

BIBLIOGRAPHY

[1] Abadie, J. "On the Kuhn-Tucker Theorem," *Nonlinear Programming*. Ed. John Wiley & Sons, Inc. New York, 1967, pp. 21-36.

[2] ____________________ "Method du Gradient Reduit Generalise: le code GRGA." Note Hi1756, Electricite de France, Paris, February 1975.

[3] Abadie, J. and Carpenter, J. "Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints," *Optimization*. R. Fletcher, Ed. Academic Press, 1969. pp. 37-47.

[4] Abadie, J. and Guigou, J., "Numerical Experiments with the GRG Method," *Integer and Nonlinear Programming*, J. Abadie, Ed., North-Holland, Amsterdam, 1970. 529-536.

[5] Assadi, J. "A Computational Comparison of Some Nonlinear Programs," *Mathematical Programming*, 4, 1973. pp. 144-154.

[6] Bard, Y. "Comparison of Gradient Methods for the Solution of Nonlinear Parameter Estimation," *SIAM J., Numer. Anal*, 7, 1970. pp. 157-187.

[7] Baker, T. E., and Bentker, R. *Successive Linear Programming in Refinery Logistic Models*. Unpublished paper, 1980.

[8] Bartels, R. *Constrained Least Squares, Quadratic Programming, Complementary Pivot Programming and Duality*. Technical Report, No. 218, Dept. of Math., the John Hopkins University, 1975.

[9] Beale, E. M. L. "Nonlinear Optimization by Simplex Like Method," *Optimization*. R. Fletcher, Ed. Academic Press, London & New York, 1969. pp. 273-281.

[10] ____________________. "An Introduction to Beale's Method of Quadratic Programming," *Nonlinear Programming* J. Abadie, Ed. John Wiley & Sons, Inc., New York, 1967. pp. 143-153.

[11] Best, M. J., Brauninger, J., Ritter, K., and Robinson, S. M., "A Globally and Quadratically Convergent Algorithm for General Nonlinear Programming Problems," Computing, 26, 1981. pp. 141-153.

[12] Box, M. J. "A Comparison of Several Current Optimization Methods and the Use of Transformations in Constrained Problems," Computer J., vol. 9, 1966. pp. 67-77.

[13] Broyden, C. G. "Quasi-Newton Methods and Their Application to Function Minimization," Math. Comput., Vol. 21, 1967. pp. 368-381.

[14] ____________________. "The Convergence of a Class of Double-Rank Minimization Algorithms, 1. General Considerations," J. Inst. Math. Applics. Vol. 6, 1970. pp. 76-90.

[15] ____________________. "The Convergence of a Class of Double-Rank Minimization Algorithms, 2. The New Algorithm," J. Inst. Math. Applics. Vol. 6, 1970. 222-231.

[16] ____________________. "Quasi-Newton Methods," Numerical Methods for Unconstrained Optimization, W. Murray, Ed. Academic Press, London & New York, 1972. pp. 87-106.

[17] Brent, R. Algorithms for Minimization Without Derivatives. Prentice-Hall, New Jersey, 1973.

[18] ____________________ "Some Efficient Algorithms for Solving Systems of Nonlinear Equations." SIAM J. Numer. Anal. 10, pp. 327-344.

[19] Burden, L. Richard and Faires, J. Douglas Numerical Analysis, Third Edition. Prindle, Weber & Schmidt, Boston, 1985

[20] Colville, A. R. A Comparative Study on Nonlinear Programming Codes. Report 320-2949 IBM New York, Scientific Center, New York, 1968.

[21] ____________________ "Nonlinear Programming Study." Results as of June 1970. Private Circulation.

[22] Cooper, I. D. and Fletcher, R. "Some Experience with Globally Convergent Algorithms for Nonlinearly Constrained Optimization," J. Optimization Theory and Applications 32(1), 1980. pp. 1-16.

[23] Davidon, W. C. Variable Metric Method for Minimization. A. E. C. Research and Development Report, ANL. 5990, 1959.

[24] ____________________. "Variance Algorithms for Minimization." Optimization, R. Fletcher, Ed. Academic Press, London & New York, 1969. pp. 187-202.

[25] Davies, D. and Shann, W. H. "Review of Constrained Optimization," Optimization R. Fletcher, Ed. Academic Press, London & New York, 1969. pp. 187-202.

[26] Dantzig, G. B., Orden, A. and Wolfe, P. Generalized Simplex Method for Minimizing a Linear Form Under Linear Inequality Constraints. Ran Report RM-1264, The Rand Corporation.

[27] Dembo, R. S. "A Set of Geometric Programming Test Problems and Their Solution," Mathematical Programming. Vol. 10, No. 2, 1976. pp. 192-193.

[28] ______________________. "The Current State-of-the-Art of Algorithms and Computer Software for Geometric Programming." Working Paper 88, School of Organization and Management, Yale University, New Haven, 1976.

[29] Dixon, L. C. W. "Quasi-Newton Techniques Generate Identical Points II: The Proofs of Four New Theorems," Math. Prog., Vol. 3, 1972. pp. 345-358.

[30] ______________________. "Variable Metric Algorithms: Necessary and Sufficient Conditions for Identical Behavior of Nonquadratic Functions," Optim. J. Theory Appls., Vol. 10, 1972. pp. 34-40.

[31] ______________________. "Choice of Step Length, A Crucial Factor in the Performance of Variable Metric Algorithms," Numerical Methods for Nonlinear Optimization. F. A. Lootsma, Ed. Academic Press, London & New York, 1972. pp. 149-170.

[32] ______________________. "Nonlinear Optimization: A Survey of the State of the Art." Software for Numerical Mathematics. D. J. Evans., Ed. Academic Press, London & New York, 1974. pp. 193-218.

[33] Dumitru, V. Gradient Methods for Unconstrained Optimization. Cybernetics Studies Res., No. 4, 1974. pp. 35-54.

[34] Fletcher, R. "A New Approach to Variable Metric Algorithms" Comput. J., Vol. 13, 1970. pp. 317-322.

[35] ______________________. "A Review of Methods for Unconstrained Optimization," Optimization. R. Fletcher, Ed. Academic Press, London & New York, 1969. pp. 1-12.

[36] Fletcher R. and C. M. Reeves. "Functions Minimization by Conguate Gradients," Computer J. 7. pp. 149-154, 1964.

[37] Fletcher R. and Powell, M. J. D. "A Rapidly Convergent Descent Method for Minimization." Comput. J., Vol. 6, 1963. pp. 163-168.

[38] Fiaco, V. Anthony and McCormick, F. Garth. "Sequential Unconstrained Minimization Techniques." Nonlinear Programming John Wiley & Sons, 1968.

[39] Franklin, Joel. "Methods of Mathematical Economics." Linear and Nonlinear Programming, Fixed-Point Theorems. Springer-Verlag, New York 1980.

[40] Gill, P. E. and Murray, W. Numerical Methods for Constrained Optimization. Academic Press, London & New York, 1974.

[41] ____________________. "Newton-Type Methods for Unconstrained and Linearly Constrained Optimization." Mathematical Programming. 1974. pp. 311-350.

[42] ____________________. Safeguarded Step Length Algorithms for Optimization Using Descent Methods Report NAC 37, 1974. National Physical Laboratory, Teddington, England.

[43] Gill, P. E., Murray, W. and Wright, M. H. Two Step Length Algorithms for Numerical Optimization Report 79-25, 1979. Department of Operations Research, Stanford University, CA. Revised, 1981.

[44] ____________________. "Q P-Based Methods for Large-Scale Nonlinear Constrained Optimization," Nonlinear Programming, J. B. Rosen, O. L. Mangasarian and K. Ritter, Eds., Academic Press, New York, 1970. pp. 67-98.

[45] Gill, P. E., Murray, W. and Wright, M. H. Practical Optimization Academic Press, 1981.

[46] Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. "Software and its Relationship to Methods." Numerical Optimization, Proceedings of the SIAM Conference on Numerical Optimization, Boulder, Colorado. June, 1984. pp. 139-159.

[47] Goldfarb, D. "Sufficient Conditions for the Convergence of a Variable Metric Algorithm: Optimization. R. Fletcher, Ed. Academic Press, London and New York, 1969. pp. 273-281.

[48] Griffith, R. E. and Stewart, R. A. "Nonlinear Programming Techniques for the Optimization of Continuous Processing Systems." Management Science, Vol. 4, 1961. pp. 379-392.

[49] Guigou, J. "Presentation et utilization du code GRG". Note Hi102, Electricite de France, Paris, June 1969.

[50] Han, S. P. "Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems." Mathematical Programming. Vol. 11, No. 3, 1976. pp. 263-282.

[51] Hartley, Ronald V. Operations Research: A Managerial Emphasis. Goodyear Publishing Company, Inc., 1976.

[52] Heltse, D. R. and Liittschwager Users' Guide for GRG 73. The University of Iowa, Iowa City, September 1973.

[53] Hillier and Lieberman, Introduction to Operations Research. Third Edition, Holden-Day, Inc., San Francisco, 1980.

[54] Himmelblau, D. M. Applied Nonlinear Programming. McGraw-Hill Book Co., New York, 1972.

[55] Huang, H. Y. and Levy, A. V. "Numerical Experiments on Quadratically Convergent Algorithm for Function Minimization," J. Optimization, Theory and Applications, Vol. 6, 1970. pp. 269-287.

[56] Huard, Pierre. "Convergence of the Reduced Gradient Method," Nonlinear Programming 2, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Eds. Academic Press, New York, 1975. pp. 29-54.

[57] Keller, E. L. "The General Quadratic Optimization Problem." Mathematical Programming, Vol. 5, No. 3, 1973. pp. 311-337.

[58] Lasdon, L. S. and Waren, A. D. "Design and Testing of a GRG Code for Nonlinear Optimization." in ACM Transactions on Mathematical Software, Vol. 4, No. 1, March 1978.

[59] ______________________ Generalized Reduced Gradient Software for Linearly and Nonlinearly Constrained Problems, Operation Resarch Department, Case Western Reserve University, Cleveland, Ohio, 1978.

[60] Lemarechal, Claude. Nonsmooth Optimization and Descent Methods, IIASA Research Report 78.4, 1978.

[61] Luenberger, David G. Introduction to Linear and Nonlinear Programming. Second Edition, Addison-Wesley Pub. Co., 1984.

[62] McCormick, G. P. "Second Order Conditions for Constrained Minima" SIAM J., Appl. Math. Vol. 15, No. 3, 1967. pp. 641-652.

[63] ____________________. "A Second-Order Method for the Linearly Constrained Nonlinear Programming Problem." Nonlinear Programming 1, J. B. Rosen, O. L. Mangasarian and K. Ritter, Eds. Academic Press, New York, 1970. pp. 207-243.

[64] ____________________. "Finding the Global Minimum of a Function of One Variable Using the Method of Constant Signed Higher Order Derivatives." Nonlinear Programming 4, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Eds. Academic Press, New York, 1981. pp. 223-244.

[65] McCormick, G. P. and Pearson, J. D. "Variable Metric Methods and Unconstrained Optimization." Optimization. R. Fletcher, Ed., Academic Press, London & New York, 1969. pp. 307-325.

[66] McKeown, J. J. "Specialized Versus General Purpose Algorithms for Minimizing Functions that Are Sums of Squared Terms." Mathematical Programming, Vol. 9, 1975. pp. 67-68.

[67] Minkoff, M. "Methods for Evaluating Nonlinear Programming Software." Nonlinear Programming 4. O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Eds., Academic Press, New York, 1981. pp. 519-548.

[68] Moore, J. J., Garbow, B. S., Hillstrom, K. E. "Testing Unconstrained Optimization Software." Technical Memorandum No. 324, Argonne National laboratory, Applied Mathematics Division, Argonne, 1978.

[69] Murray, W. "An Algorithm for Constrained Minimization," Optimization, R. Fletcher, Ed., Academic Press, London & New York, 1969.

[70] ____________________. "Methods for Constrained Optimization," Optimization in Action. L. C. W. Dixon, Ed., Academic Press, New York, 1976.

[71] Murtagh, B. A. "On the Simultaneous and Optimization of Large-Scale Engineering Systems," J. Computer and Chemical Engineering. Vol. 6, No. 1, 1982. pp. 1-5.

[72] Murtagh, B. A. and Sargent, R. W. H. "A Constrained Minimization Method With Quadratic Convergence." Optimization. R. Fletcher, Ed. Academic Press, London & New York, 1969. pp. 215-247.

[73] Murtagh, B. A. and Saunders, M. A. "Large-Scale Linearly Constrained Optimization." Mathematical Programming 4, 1978. pp. 41-72.

[74] ______________________. MINOS Users' Guide. Report SOL 77-9, 1977. Department of Operations Research, Stanford University, CA.

[75] ______________________. MINOS Augmented Users' Manual. Report SOL 80-14, 1980. Department of Operations Research, Stanford University CA.

[76] ______________________. "A Projected Lagrangian Algorithm and Its Implementation for Sparse Nonlinear Constraints." J. Mathematical Programming Study 16, 1982. pp. 84-117.

[77] ______________________. MINOS-5.0 Users' Manual. 1983. Department of Operations Research, Stanford University, CA.

[78] Perry, A. "An Improved Conguate Gradient Algorithm," Technical Note. Department of Design Sciences, Graduate School of Management, Northwestern University, Evanston, Illinois, March 1976.

[79] Polak, E. "Computational Methods in Optimization: A Unified Approach." Mathematics in Science and Engineering. Vol. 77, 1971. pp. 28-40; 126-150.

[80] Powell, M. J. D. "An Iterative Method for Finding Stationary Values of A Function of Several Variables." J. Computer, Vol. 5, 1962. pp. 147-151.

[81] ______________________. "A Method for Nonlinear Constraints in Optimization Problems." Optimization. R. Fletcher. Ed., Academic Press, New York, 1969. pp. 283-297.

[82] ______________________. "A Survey of Numerical Methods for Unconstrained Optimization." SIAM Review. vol. 12, 1970. pp. 79-97.

[83] ______________________. "Rank One Methods for Unconstrained Optimization." Integer and Nonlinear Programming. J. Abadie, Ed. Amsterdam: North-Holland Pub. Co., 1970. pp. 139-156.

[84] ______________________. "A New Algorithm for Unconstrained Optimization." Nonlinear Programming 1. Academic Press, London & New York, 1970. pp. 31-66.

[85] ______________________. "On the Convergence of the Variable Metric Algorithm." J. Inst. Math. Appl. Vol. 7, 1971. pp. 21-36.

[86] ______________________. "Recent Advances in Unconstrained Optimization." Mathematical Programming, Vol. 1, 1971. pp. 26-57.

[87] ____________________. "Some Properties of the Variable Metric Algorithm." *Numerical Methods for Nonlinear Optimization*. F. A. Lootsma, Ed. Academic Press, London & New York, 1972. pp. 1-17.

[88] ____________________. *Quadratic Termination Properties of a Class of Double-Rank Minimization Algorithms*. A.E.R.E. Report TP 471, Harwell, England. Atomic Energy Research Establishment, 1972.

[89] ____________________. "*Some Theorems on Quadratic Termination Properties of Minimization Algorithms*." A.E.R.E. Report TP 472, Harwell, England. Atomic Energy Research Establishment, 1972.

[90] ____________________. "Convergence Properties of a Class of Minimization Algorithms.: *Nonlinear Programming* 2, O. L. Mangasarian, R. R. Meyer, S. M. Robinson, Eds., Academic Press, New York, 1975. pp. 1-28.

[91] Ramsin, H. And Wedin, P. A. "A Comparison of Some Algorithms for the Nonlinear Least Squares Problem." *BIT*, Vol. 17, No. 1, 1977. pp. 72-90.

[92] Rijckaret, M. J. "Computational Aspects of Geometric Programming." *Design and Implementation of Optimization Software*. H. J. Greenberg, Sijhoff and Noordhoff, Eds., The Netherlands, 1977.

[93] Rijckaert, M. J. and Martens, X. M. "A Comparison of Generalized Geometric Programming Algorithms." *J. Optimization Theory and Applications*. Vol. 26, No. 2, 1978. pp. 205-242.

[94] Ritter, Klaus. "A Quasi-Newton Method for Unconstrained Minimization Problems." *Nonlinear Programming* 2, O. L. Mangasarian, R. R. Meyer, S. M. Robinson, Eds., Academic Press, New York, 1975. pp. 55-100.

[95] Robinson, S. M. "A Quadratically-Convergent Algorithm for General Nonlinear Programming Problems." *Mathematical Programming* 3, 1972. pp. 145-156.

[96] Rosen, J. B. "Two-Phase Algorithm for Nonlinear Constrained Problems." *Nonlinear Programming* 3, O. L. Mangasarian, R. R. Meyer, S. M. Robinson, Eds., Academic Press, London, 1978. pp. 74-97.

[97] ____________________. "The Gradient Projection Methods for Nonlinear Programming, Part D, Linear Constraints." *J. Soc. Indust. Appl. Math.* 8, 1960. pp. 191-217.

[98] ____________________. "The Gradient Projection Methods for Nonlinear Programming part 2, Nonlinear Constraints." J. Soc. Indust. Appl. Math. 9, 1960. pp. 514-532.

[99] Schittkowski, K. "The Construction of Degenerate, Ill-Conditioned, and Indefinite Nonlinear Programming Problems and Their Usage to Test Optimization Programs," in print.

[100] ____________________. "Information, Tests, Performance." Nonlinear Programming Codes, M. Beckmann and H. P. Kunzi, Eds. Springer, Berling, Heidelberg, New York, 1980.

[101] Schittkowski, K. and Store, J. "A Factorization Method for the Solution of Constrained Linear Least Squares Problems Allowing Subsequent Data Changes." Num. Math. 31, 1979. pp. 431-463.

[102] Shanno, D. F. "Conditioning of Quasi-Newton Methods for Function Minimization." Math. Comput. Vol. 24, 1970. pp. 647-656.

[103] Shanno, D. F. and Keller, D. C. "Optimal Conditioning of Quasi-Newton Methods." Math. Comput., Vol. 24, 1970. pp. 657-664.

[104] Shor, N. Z. "Convergence Rate of the Gradient Descent Method with Dilation of the Space," Kibernetika 2, 1970. pp. 102-108.

[105] Stewart, G. W. Introduction to Matrix Computations. Academic Press, London & New York, 1973.

[106] Stocker, D. C. "A Comparative Study of Nonlinear Programming Codes." M. S. Thesis, The University of Texas, Austin, Texas, 1969.

[107] Store, J. "On the Convergence Rate of Imperfect Minimization Algorithms." Mathematical Programming 9, 1975. pp. 313-335.

[108] Taylor, J. R. "The Davidson-Fletcher-Powell Method and Families of Variable Metric Methods for Unconstrained Minimization." Ed. D. Thesis, The University of Oklahoma, Norman, Oklahoma, 1976.

[109] Telgan, Jan. "Redundancy and Linear Programs," 1979. pp. 33-53.

[110] Villiers, De. Noel and Glasser, David. "A Continuation Method for Nonlinear Regression." J. Num. Anal., Vol. 18, No. 6, Dec. 1981. pp. 1139-1153.

[111] Wolfe, P. "A Method of Conjugate Subgradients for Minimizing Nondifferentiable Functions." Mathematical Programming Study 3, 1975. pp. 145-173.

[112] ______________________. "Methods for Linear Constraints." *Nonlinear Programming*. J. Abadie, Ed., John Wiley & Sons, Inc., New York, 1967. pp. 121-124.

[113] ______________________. "Convergence Theory in Nonlinear Programming." *Integer and Linear Programming*. J. Abadie, Ed., North-Holland Pub. Co., Amsterdam, London, 1970. pp. 1-36.

[114] ______________________. "Convergence Conditions for Ascent Methods." *SIAM Review* 11, 1969. pp. 226-235.

[115] ______________________. "Methods of Nonlinear Programming." *Recent Advances in Mathematical Programming*. R. L. Graves and P. Wolfe, Eds., McGraw-Hill Book Co., 1963. pp. 67-86.

[116] ______________________. "On the Convergence of Gradient Methods Under Constraint." *IBM Journal of Research and Development* 16, 1972. pp. 407-411.

[117] ______________________. "The Simplex Method for Quadratic Programming." *Econometrica*, Vol. 27, 1959. pp. 382-398.

2

VITA

Seyed Abolghassem Alemzadeh

Candidate for the Degree of

Doctor of Education

Thesis: COMPARATIVE STUDY OF LARGE-SCALE NONLINEAR OPTIMIZATION METHODS

Major Field: Higher Education

Biographical:

Personal Data: Born in Kerman, Iran, May 27, 1944, son of Mr. and Mrs. Alemzadeh.

Education: Graduated from the Shahpoor High School, Kerman, Iran, in 1963; received Bachelor of Science degree in Mathematics from the Teaching Training University in Tehran, Iran in August 1970; received Master of Education in Mathematics from Central State University in 1975; completed requirements for the Doctor of Education degree at the Oklahoma State University in May 1987.

Professional Experience: Served as an elementary school teacher, Ministry of Education, Iran, 1966-1970; served as a high school mathematics teacher, Ministry of Education, Iran, 1970-1973; served as Graduate Teaching Associate, Department of Mathematics, Oklahoma State University, 1977-1984; served as assistant professor of mathematics, Department of Mathematics, State University of New York, College at Cortland, 1984-1986.

Professional Organizations: Member of the American Mathematical Society and Society of Industrial Engineering and Applied Mathematics.