An Alternating Direction Search Algorithm for Low Dimensional Optimization: An Application to Power Flow

by

Tina R. Burrell

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Mathematics

APPROVED:

Terry L Herdman

Prof. Terry L. Herdman, Chairman

John A. Burns

Prof. John A. Burns

Eugene M Cliff

Prof. Eugene M. Cliff

Robert P Broadwater

Prof. Robert P. Broadwater

Prof. Max D. Gunzburger

May, 1993

Blacksburg, Virginia

An Alternating Direction Search Algorithm for Low Dimensional Optimization: An Application to Power Flow

by

Tina R. Burrell

Committee Chairman: Prof. Terry L. Herdman

Department of Mathematics

(ABSTRACT)

Presented in this paper is a scheme for minimizing the cost function of a three-source power system. This scheme, the Line-Step Algorithm, uses an alternating direction step technique to arrive at an approximation point (I, J) that is within one unit of the true minimum. The Line-Step Algorithm is applied to several systems and is also compared to other minimization techniques, including the Equal Incremental Loss Algorithm. Variations are made on the Line-Step Algorithm for faster convergence and also to handle inequality constraints.

ACKNOWLEDGEMENTS

I would first like to thank Dr. Terry Herdman, my advisor and friend, for all of his support and encouragement. I would never have come this far had it not been for him. Next, for all of their work and input, I wish to thank the "Friday Morning Crew": Dr. Robert Broadwater, Rossi Karamikhova, and Paul Dolloff. I could not have done it without their help. I would also like to thank Dr. John Burns, Dr. Gene Cliff, and Dr. Max Gunzburger for their help and support.

I would next like to thank everyone at ICAM who gave me a hand in putting this paper together, especially Melissa for going above and beyond the call of duty.

Finally, I wish to thank my family, especially Dale, for all of their support, patience, and love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
INTRODUCTION

As the costs associated with generation and transmission of electric energy keep increasing, optimal economic dispatch in electric power systems is gaining increasing importance. Optimal Power Flow (OPF) is a generic name for constrained static scheduling calculations in power system transmission networks [4]. Many solution approaches, each with particular mathematical and computational characteristics, have been developed to solve the OPF problem. OPF methods vary considerably in their adaptablility to the modeling and solution requirements of different engineering applications [1].

The objective of Optimal Power Flow is to improve power system efficiency by satisfying a given set of loads at minimum generation costs and line losses. The OPF problem is defined by the transmission system, the set of loads, and the available generation. The optimum system state is achieved when all loads are supplied with power at total minimum line loss and generation costs. Previous developments in the field of Optimal Power Flow have used gradient methods, linear programming methods, and Lagrange Multiplier approaches [3].

By considering simple problems, new techniques for solving the Optimal Power Flow problem are continually being developed. Such simple problems may by modeled by three-source power systems. A general three-source power system, as illustrated in Figure 1.1, can be given by three current sources, five line sections modeled as resistors, and three constant current loads. Each line has a resistance labeled as "R" and each load has a current labeled as "x". The current travels from the source to the load through the line.

The cost function used here is a function that measures the loss which results from a current traveling through a line [7]. This is given by

$$Cost = Rx_i^2.$$

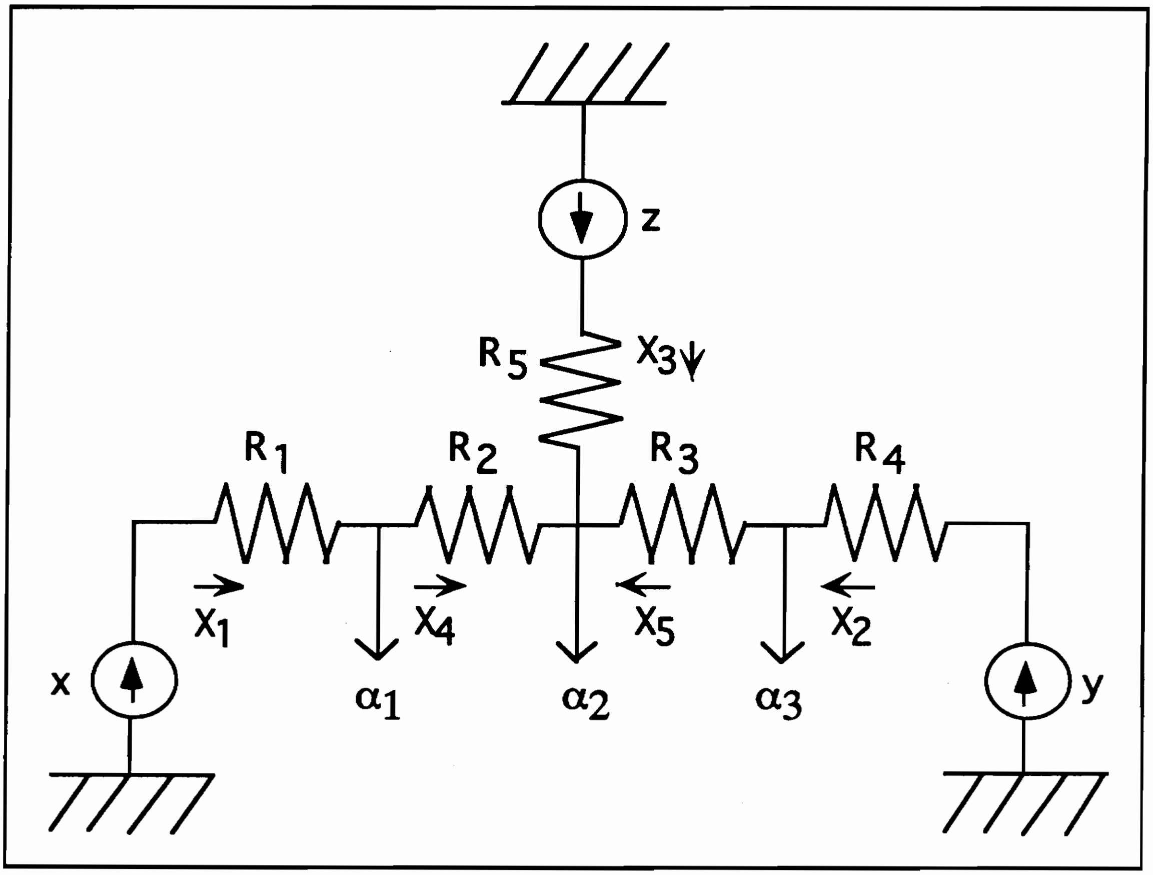

Figure 1.1: General Three-Source System

Therefore, the total cost of the system in Figure 1.1 is

$$Cost = R_1x_1^2 + R_2x_4^2 + R_3x_5^2 + R_4x_2^2 + R_5x_3^2.$$

However, this may be rewritten in three variables, corresponding to the sources, as

$$Cost = \hat{F}(x, y, z) = R_1x^2 + R_2(x - \alpha_1)^2 + R_3(y - \alpha_3)^2 + R_4y^2 + R_5z^2.$$

The only constraint on the system under consideration is that the sum of the loads must equal the sum of the current sources. That is,

$$x + y + z = \alpha_1 + \alpha_2 + \alpha_3. \tag{1.1}$$

Using this constraint, the problem may be reduced from minimizing a function of three variables subject to an equality constraint to one of finding the minimum of a function of two variables having no constraints. Solving equation (1.1) for z yields

$$z = \alpha_1 + \alpha_2 + \alpha_3 - (x + y). \tag{1.2}$$

Using this substitution, the cost function becomes

$$F(x, y) = R_1 x^2 + R_2(x - \alpha_1)^2 + R_3(y - \alpha_3)^2 + R_4 y^2 + R_5(\alpha_1 + \alpha_2 + \alpha_3 - x - y)^2 \tag{1.3}$$

and the problem restated as

$$\min_{x,y>0} F(x, y).$$

Presented in this paper is a step algorithm which, by taking steps of unit length, converges to an integer approximation to the minimum for a function of two variables subject to no constraints. In a finite number of steps, the scheme produces an ordered pair of integers (I, J) that gives an integer approximation for the true point of minimization (x^*, y^*). That is, $[x^*] \leq I \leq [x^*] + 1$ and $[y^*] \leq J \leq [y^*] + 1$, where $[a]$ denotes the greatest integer less than or equal to the real number a. This algorithm is later modified to take steps of varying lengths. Although this paper investigates the behavior of a general three-source power system, the results presented here should provide a foundation for similar results for n-source systems.

Chapter 2 establishes that, for a typical three-source system, there is a unique minimum. The Line-Step Algorithm is discussed and proven to work in Chapter 3. In Chapter 4, the Equal Incremental Loss Algorithm, another power flow minimization technique, is introduced. Real models and inequality constraints are considered in Chapter 5.

Chapter 2

QUADRATIC FORMS WITH POSITIVE DEFINITE MATRICES A

Solving equation (1.3) is equivalent to finding the point (x^*, y^*) that minimizes the quadratic function

$$f(x,y) = ax^2 + by^2 + cxy - dx - ey + h \tag{2.1}$$

where a, b, c, d, e, and h are real numbers satisfying $a > 0$, $b > 0$, $c \geq 0$, $d \geq 0$, $e \geq 0$, $2a > c$, $2b > c$, $d \geq c$, and $e \geq c$. These inequalities are the results of further calculations on equation (1.3). Clearly, $R_i > 0$ for $1 \leq i \leq 5$ and $\alpha_j > 1$ for $1 \leq j \leq 3$. Therefore, since $a = R_1 + R_2 + R_5$, $a > 0$. Similarly, $b = R_3 + R_4 + R_5 > 0$ and $c = 2R_5 \geq 0$. Also, $d = 2(R_1\alpha_1 + R_5\alpha_1 + R_5\alpha_2 + R_5\alpha_3) \geq 0$ and $e = 2(R_3\alpha_3 + R_5\alpha_1 + R_5\alpha_2 + R_5\alpha_3) \geq 0$. The inequalities $2a > c$ and $2b > c$ are determined by $2a = 2(R_1 + R_2 + R_5) = 2R_1 + 2R_2 + 2R_5 > 2R_5 = c$ and $2b = 2(R_3 + R_4 + R_5) = 2R_3 + 2R_4 + 2R_5 > 2R_5 = c$. Similarly, $d \geq c$ and $e \geq c$ follow from $d = 2R_1\alpha_1 + 2R_5(\alpha_1 + \alpha_2 + \alpha_3) \geq 2R_5 = c$ and $e = 2R_3\alpha_3 + 2R_5(\alpha_1 + \alpha_2 + \alpha_3) \geq 2R_5 = c$. Notice that h may be ignored when considering the problem of identifying the point (x^*, y^*) since the only role h plays is in obtaining the minimum function value, $f(x^*, y^*)$. Therefore, the problem of finding (x^*, y^*) is equivalent to minimizing the function F given by

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A \mathbf{x} - \mathbf{x}^T B \tag{2.2}$$

where $A = \begin{bmatrix} 2a & c \\ c & 2b \end{bmatrix}$, $B = \begin{bmatrix} d \\ e \end{bmatrix}$, and $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$.

It is useful to examine the matrix A and determine that A is positive definite. First, recall that a matrix G is positive definite if and only if $\mathbf{x}^T G\mathbf{x} > 0$ for all nonzero $\mathbf{x}$ [6]. So,

$$\begin{aligned}
\mathbf{x}^T A\mathbf{x} &= \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2a & c \\ c & 2b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\
&= \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2ax + cy \\ cx + 2by \end{bmatrix} \\
&= 2ax^2 + cxy + cxy + 2by^2 \\
&= 2(ax^2 + cxy + by^2).
\end{aligned}$$

The above inequalities on the coefficients of f yield

$$\begin{aligned}
2(ax^2 + cxy + by^2) &= 2ax^2 + 2cxy + 2by^2 \\
&> cx^2 + 2cxy + cy^2 \\
&= c(x + y)^2 \geq 0.
\end{aligned}$$

Therefore, A is positive definite. So, now $F(\mathbf{x})$ is a quadratic function with a positive definite matrix A.

To show that $F(\mathbf{x})$ has a unique minimum, $\mathbf{x}^* = \begin{bmatrix} x^* & y^* \end{bmatrix}^T$, it will suffice to show that there is a unique $\mathbf{x}^*$ such that $F(\mathbf{x}) > F(\mathbf{x}^*)$ for all $\mathbf{x} \neq \mathbf{x}^*$ [9]. First, write the Taylor expansion of $F(\mathbf{x})$ about $\mathbf{x}^*$. Let $\mathbf{x}^*$ denote the unique solution of $\nabla F(\mathbf{x}) = 0$. The uniqueness of $\mathbf{x}^*$ follows from the nonsingularity of the matrix A. It follows that

$$F(\mathbf{x}) = F(\mathbf{x}^*) + \nabla F(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^T H_F(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) \tag{2.3}$$

where $H_F(\mathbf{x})$ is the Hessian matrix for $F(\mathbf{x})$. Since $\nabla F(\mathbf{x}^*) = 0$, the expansion of $F(\mathbf{x})$ about $\mathbf{x}^*$ becomes

$$F(\mathbf{x}) = F(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^T H_F(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*),$$

where

$$H_F(\mathbf{x}^*) = \begin{bmatrix} 2a & c \\ c & 2b \end{bmatrix}.$$

This is exactly the matrix A. It has previously been shown that A is a positive definite matrix, and since $\mathbf{x} - \mathbf{x}^* \neq 0$,

$$(\mathbf{x} - \mathbf{x}^*)^T H_F(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) > 0$$

for all $\mathbf{x} \neq \mathbf{x}^*$. Hence, $F(\mathbf{x}) > F(\mathbf{x}^*)$ for all $\mathbf{x} - \mathbf{x}^* \neq 0$. Therefore, $F(\mathbf{x})$ has a unique global minimum at $\mathbf{x}^*$.

Next, it will be shown that the unique minimum $\mathbf{x}^*$ is such that $x^* \geq 0$ and $y^* \geq 0$. Since F has a unique global minimum (x^*, y^*),

$$\nabla F(x^*, y^*) = (F_x(x^*, y^*), F_y(x^*, y^*)) = (0, 0).$$

The above necessary condition $\nabla F(x^*, y^*) = (0, 0)$ together with the previously mentioned inequalities on the coefficients of f yield

$$x^* = (2bd - ce)/(4ab - c^2) \geq (2bc - ce)/(4ab - c^2) \geq (c(2b - e))/(4ab - c^2) \geq 0.$$

and

$$y^* = (2ae - cd)/(4ab - c^2) \geq (2ac - cd)/(4ab - c^2) \geq (c(2a - d))/(4ab - c^2) \geq 0$$

Therefore, the minimum of equation (2.1) will always lie in the first quadrant of the xy-plane.

Chapter 3
THE LINE-STEP ALGORITHM

3.1 Geometry of the Line-Step Algorithm

The solution to the problem of approximating the minimum of a quadratic function has long since been known. In fact, there are numerous software packages to handle this situation. In many applications a real-valued approximation is quite appropriate, however there are instances that benefit from an integer-valued solution. If one requires an integer approximation, the difficulty of the problem increases.

The Optimal Power Flow problem is one application in which discrete approximation techniques which can minimize the cost function are desired. Such techniques should have the ability to efficiently handle inequality constraints that may be placed on a power system. Also, such routines should be easy to implement and quick to converge to a solution. This is the motivation for the development of the Line-Step Algorithm.

The Line-Step Algorithm is an iterative technique for finding an integer-valued approximation to the true minimum of the quadratic function F as given by equation (2.2). The algorithm begins by defining the line L in the xy-plane by

$$L = \{(x, y) : F_x(x, y) = F_y(x, y)\}.$$

It follows that (x^*, y^*) lies on L. The slope-intercept form of the equation for the line L is given by $y = mx+b_y$ with slope $m = (2a-c)/(2b-c) > 0$ and y-intercept $b_y = (e-d)/(2b-c)$. Notice that the y-intercept will be nonnegative if $e \geq d$ and nonpositive if $d \geq e$. Figure 3.1 shows a contour plot for the function F. The line L divides the xy-plane into two half-planes. The lower half plane contains all points (x, y) satisfying $y < mx + b_y$. It follows that $y < mx + b_y$ if and only if $F_y(x, y) < F_x(x, y)$. This equivalence can be established

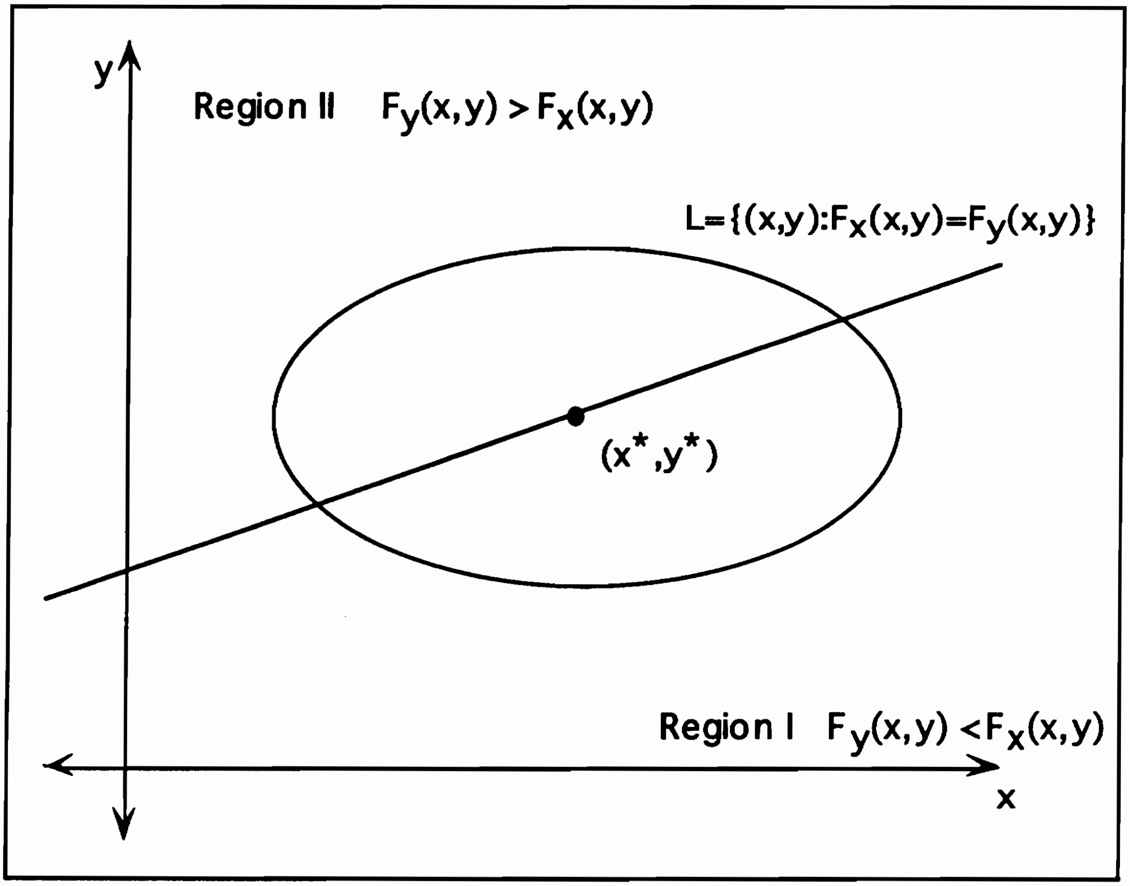
y
Region II $F_y(x,y) > F_x(x,y)$
$L=\{(x,y):F_x(x,y)=F_y(x,y)\}$
(x^*,y^*)
Region I $F_y(x,y) < F_x(x,y)$
x

Figure 3.1: Contour Plot for Equation (2.2)

from the inequalities on the coefficients of equation (2.1). That is,

$$\begin{aligned} y < ((2a-c)x+(e-d))/(2b-c) &\Leftrightarrow (2b-c)y < (2a-c)x+(e-d) \\ &\Leftrightarrow 2by+cx-e < 2ax+cy-d \\ &\Leftrightarrow F_y(x,y) < F_x(x,y). \end{aligned}$$

On the other hand, $y > mx + b_y$ if and only if $F_y(x,y) > F_x(x,y)$. Consequently, the xy-plane may be divided into two regions defined as follows:

$$\text{Region I}=\{(x,y) : F_y(x,y) < F_x(x,y)\}$$

and

$$\text{Region II}=\{(x,y) : F_y(x,y) > F_x(x,y)\}.$$

The Line-Step Algorithm begins at the point $(0,0)$. If $(0,0)$ is in Region I, that is if $F_y(0,0) < F_x(0,0)$, then the descent direction is $(0,1)$ since $-F_y(0,0) > -F_x(0,0)$. The number of steps taken in this direction (each one unit in length) is determined by the smallest integer k which results in $(x_k, y_k) = (0,0) + k(0,1)$ being in Region II. Since the point now lies in Region II, $F_y(x_k, y_k) > F_x(x_k, y_k)$. Therefore, the descent direction is $(1,0)$ since $-F_y(x_k, y_k) < -F_x(x_k, y_k)$. As before, the number of steps taken is determined by the smallest integer required to cross L and return to Region I. This procedure is repeated until the "stopping criteria" is satisfied. In Region I steps are always taken in the direction of $(0,1)$, with a few exceptions, and in Region II steps are always taken in the direction of $(1,0)$, again with a few exceptions.

Each time the line L is crossed, a check is made to be sure that at the previous point the selected direction to proceed is in the direction that results in being as "close" as possible to the line L. Also at each point (x_i, y_i), a comparison is made between $F(x_i, y_i)$ and $F(x_{i+1}, y_{i+1})$ to determine if the scheme should stop at this point or continue to the next. The stopping criteria is satisfied if $F(x_i, y_i) \leq F(x_{i+1}, y_{i+1})$. If this is the case, the point (x_i, y_i) is used as the integer approximation to the minimum; otherwise, the scheme chooses the next point.

As noted before, if (x_i, y_i) is in Region II, then the scheme would, on most occasions, choose to proceed in the $(1,0)$ direction. However, consider the case illustrated in Figure 3.2. Here, L has a large positive slope and the scheme is at the point denoted (x_i, y_i). If a step is taken in the $(1,0)$ direction then the distance between $(x_i + 1, y_i)$ and the line L is larger than the distance between $(x_i, y_i + 1)$ and L. In this case, an exception would be made and the scheme would choose $(x_i, y_i + 1)$ as the next point in the sequence since this choice yields a smaller distance to the line L.

If a non-stopping point (x_j, y_j) falls on the line L, then it makes no difference in which direction the next step is taken. If the stopping criteria is not satisfied by either of the points $(x_j + 1, y_j)$ or $(x_j, y_j + 1)$, then, after one more step, both choices yield the same point $(x_j + 1, y_j + 1)$. This can be seen in Figure 3.3.

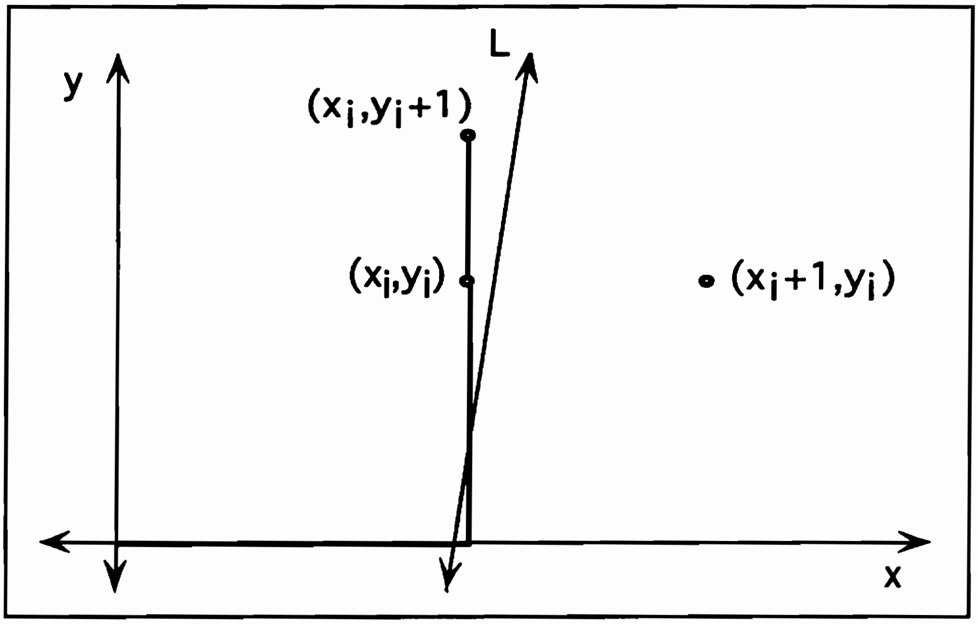

Figure 3.2: Line L with Large Positive Slope

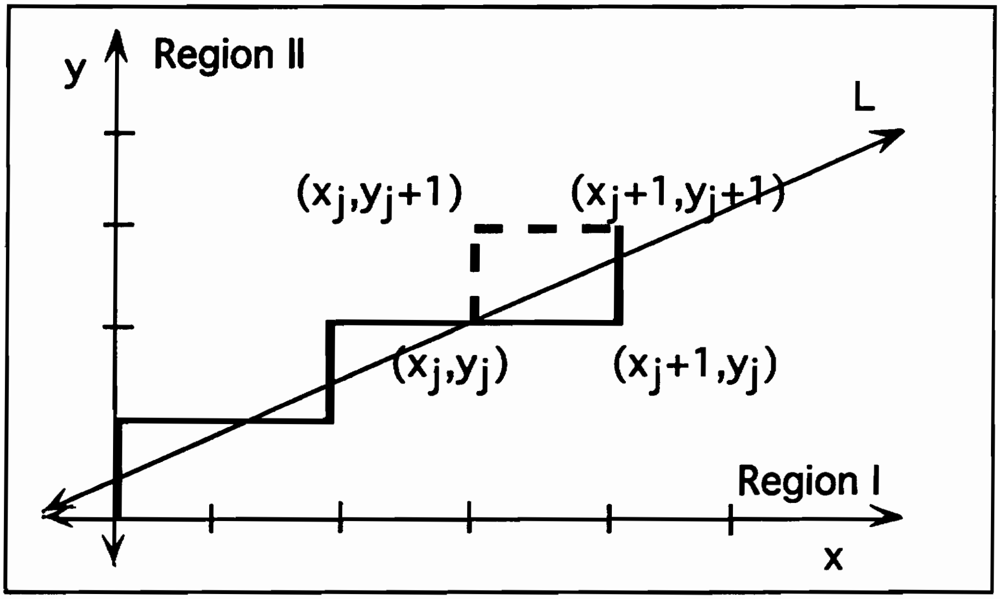

Figure 3.3: Non-Stopping Point (x_j, y_j) on L

3.2 Discussion of the Algorithm

The Line-Step Algorithm is now presented in full detail. It is a step method that takes unit steps in the positive x or positive y direction, crossing back and forth across the line L. It should be noted that a point could lie on the line L, yielding two choices for the selection of the next point. However, the desired end result, being within one unit of each component of the true minimum (x^*, y^*), will be satisfied without including this option in the scheme (see Figure 3.3). In the event that (x_i, y_i) does lie on L, the scheme automatically takes the next step in the same direction as the previous step.

The Line-Step Algorithm

Step 0. Let $m = (2a - c)/(2b - c)$, $b_y = (e - d)/(2b - c)$.

Step 1. Check the point $(x_1, y_1) = (0, 0)$ to see if it is in Region I, Region II, or lies on L.

Let $i = 1$.

If $(x_1, y_1) = (0, 0)$ is in Region I, go to Step 2.

If $(x_1, y_1) = (0, 0)$ is in Region II, go to Step 3.

If $(x_1, y_1) = (0, 0)$ lies on L, go to Step 4.

Step 2. Find the intersection point of $x = x_i$ and L. Denote this point by (x_i, η).

Let $\eta^* = [\eta] + 1$. Let $j = 1$.

Let $(x_{i+j}, y_{i+j}) = (x_i, y_i) + j(0, 1)$.

While $F(x_{i+j-1}, y_{i+j-1}) > F(x_{i+j}, y_{i+j})$ and $j \leq \eta^* - 1$, let $j = j + 1$.

If $F(x_{i+j-1}, y_{i+j-1}) \leq F(x_{i+j}, y_{i+j})$, let $i = i + j$ and go to Step 5.

If $j = \eta^*$, let $H = y_{i+j} - (mx_{i+j} + b_y)$ and $W = m(x_{i+j} + 1) + b_y - y_{i+j-1}$.

If $H \leq W$, let $i = i + j$ and go to Step 3.

If $H > W$, then let $(x_{i+j}, y_{i+j}) = (x_{i+j-1}, y_{i+j-1})$, $i = i + j$, and start Step 2 again.

Step 3. Find the intersection point of $y = y_i$ and L. Denote this point by (δ, y_i).
Let $\delta^* = [\delta] + 1$. Let $j = 1$.
Let $(x_{i+j}, y_{i+j}) = (x_i, y_i) + j(1,0)$.
While $F(x_{i+j-1}, y_{i+j-1}) > F(x_{i+j}, y_{i+j})$ and $j \leq \delta^* - 1$, let $j = j + 1$.
If $F(x_{i+j-1}, y_{i+j-1}) \leq F(x_{i+j}, y_{i+j})$, let $i = i + j$ and go to Step 5.
If $j = \delta^*$, let $Y = x_{i+j} - ((y_{i+j} - b_y)/m)$ and $X = ((y_{i+j} + 1 - b_y)/m) - x_{i+j-1}$.
If $Y \leq X$, let $i = i + j$ and go to Step 2.
If $Y > X$, then let $(x_{i+j}, y_{i+j}) = (x_{i+j-1}, y_{i+j-1})$, $i = i + j$, and
start Step 3 again.

Step 4. Choose $(x_{i+1}, y_{i+1}) = (x_i, y_i) + (1,0)$ or $(x_{i+1}, y_{i+1}) = (x_i, y_i) + (0,1)$
(the scheme will arrive at the same point in two steps unless the
stopping criteria is satisfied prior to making two steps).
If $F(x_i, y_i) \leq F(x_{i+1}, y_{i+1})$, let $i = i + 1$ and go to Step 5.
If $F(x_i, y_i) > F(x_{i+1}, y_{i+1})$, let $i = i + 1$ and go to Step 2 if
$(x_{i+1}, y_{i+1}) = (x_i, y_i) + (1,0)$ or go to Step 3 if
$(x_{i+1}, y_{i+1}) = (x_i, y_i) + (0,1)$.

Step 5. Let $(I, J) = (x_{i-1}, y_{i-1})$ be the integer approximate for (x^*, y^*).

3.3 Convergence of the Line-Step Algorithm

It is now proven that the Line-Step Algorithm converges to a point (I, J) with $|I - x^*| \leq 1$ and $|J - y^*| \leq 1$. It is further shown that the scheme reaches its stopping point in a finite number of steps.

Theorem. In k steps with $[x^*] + [y^*] \leq k \leq [x^*] + [y^*] + 2$, the Line-Step Algorithm will stop at $(I, J) = (x_k, y_k)$ with $|x_k - x^*| \leq 1$, $|y_k - y^*| \leq 1$.

Proof. Let $k = \lfloor x^* \rfloor + \lfloor y^* \rfloor + 2$ (i.e., the scheme ran for $\lfloor x^* \rfloor + \lfloor y^* \rfloor + 2$ steps). The scheme chooses points (x_i, y_i) in such a manner that (x_i, y_i) is always within one unit to the right or one unit above the line L. Since (x^*, y^*) lies on L and the scheme can only step to the right or up, (x_i, y_i) must stay inside or on the boundary of the rectangle Ω having coordinates $(0,0)$, $(0, \lfloor y^* \rfloor + 1)$, $(\lfloor x^* \rfloor + 1, 0)$, and $(\lfloor x^* \rfloor + 1, \lfloor y^* \rfloor + 1)$. Since one step is taken at each iteration, after k iterations $x_k = \lfloor x^* \rfloor + 1$ and $y_k = \lfloor y^* \rfloor + 1$. Without loss of generality, suppose that the next step attempted was in the x-direction (a similar argument considers an attempt to step in the y-direction). Thus, $x_{k+1} = \lfloor x^* \rfloor + 2$ and $y_{k+1} = \lfloor y^* \rfloor + 1$. Let $\epsilon = x^* - \lfloor x^* \rfloor$, $\tau = y^* - \lfloor y^* \rfloor$. It follows that $0 \leq \epsilon < 1$, $0 \leq \tau < 1$ and

$$\begin{aligned}
F(x_{k+1}, y_{k+1}) - F(x_k, y_k) &= 2a\lfloor x^* \rfloor + 3a + c(\lfloor y^* \rfloor + 1) - d \\
&= 2ax^* + cy^* - d + 3a + c - 2a\epsilon - c\tau \\
&= F_x(x^*, y^*) + a(3 - 2\epsilon) + c(1 - \tau) \\
&= a(3 - 2\epsilon) + c(1 - \tau) > 0.
\end{aligned}$$

Thus, the scheme would stop in $\lfloor x^* \rfloor + \lfloor y^* \rfloor + 2$ steps with $(I, J) = (x_k, y_k) = (\lfloor x^* \rfloor + 1, \lfloor y^* \rfloor + 1)$. In this case, the desired inequalities $0 \leq I - x^* \leq 1$, $0 \leq J - y^* \leq 1$ are clearly satisfied.

Now, a claim is made that if the scheme is stopped prior to making $\lfloor x^* \rfloor + \lfloor y^* \rfloor + 2$ steps, then it stops at one of the corners of the unit box Γ having corners $(\lfloor x^* \rfloor, \lfloor y^* \rfloor)$, $(\lfloor x^* \rfloor, \lfloor y^* \rfloor + 1)$, $(\lfloor x^* \rfloor + 1, \lfloor y^* \rfloor)$, $(\lfloor x^* \rfloor + 1, \lfloor y^* \rfloor + 1)$. It should be clear that Γ contains the point (x^*, y^*).

First consider the cases in which the scheme produces a point of the form $(\lfloor x^* \rfloor + 1, \lfloor y^* \rfloor - j)$ or of the form $(\lfloor x^* \rfloor - j, \lfloor y^* \rfloor + 1)$ for some $j \geq 1$. In either of these cases, the point is on the boundary of Ω. Since L has a positive slope, the scheme would only stop if

$$F(\lfloor x^* \rfloor + 1, \lfloor y^* \rfloor - j + 1) - F(\lfloor x^* \rfloor + 1, \lfloor y^* \rfloor - j) = b - 2b(\tau + j) + c(1 + \epsilon) \geq 0$$

or

$$F([x^*]-j+1,[y^*]+1)-F([x^*]-j,[y^*]+1)=a-2a(\epsilon+j)+c(1+\tau)\geq 0.$$

Without loss of generality, consider the first inequality given above. That is, consider the case where the point lies on the right boundary of Ω. Recall that once the line L is crossed, the scheme produces points within one unit of L. Since the point lies on the right boundary of Ω, L has been crossed at least one time. Therefore, L lies between $([x^*],[y^*]-j)$ and $([x^*]+1,[y^*]-j)$. Notice that in order to reach a point in the iterative procedure having x coordinate $[x^*]+1$, the last crossing of L, not necessarily at the previous step, must have occurred by moving in the x direction from $([x^*],i)$ to $([x^*]+1,i)$ for some nonnegative integer i, with $i\leq [y^*]-j$. In addition, at the point of the crossing, the x coordinate is $((i-b_y)/m)$ which yields

$$[x^*]+1-((i-b_y)/m)\leq ((i-b_y)/m)-[x^*]$$

or

$$1+2(x^*-((i-b_y)/m))\leq 2\epsilon.$$

The slope of L is positive, hence x^* must lie to the right of $(i-b_y)/m$. It should be noted that $x^*-(i-b_y)/m>0$ since $i<[y^*]$ (recall that $j\geq 1$). This observation, together with the above inequality, implies that $\epsilon>1/2$. This estimate for ϵ combined with the stopping criteria gives

$$b-2b(\tau+j)\geq -c(1-\epsilon)\geq -c+c/2=-c/2>-b$$

or

$$2b(1-\tau-j)>0$$

which holds only if $1>(\tau+j)$. However, this cannot hold since $j\geq 1$. Therefore, the stopping criteria will not be satisfied at any point $([x^*]+1,[y^*]-j)$ or $([x^*]-j,[y^*]+1)$ for some integer $j\geq 1$.

Now, consider the possibility of stopping the scheme at some point $([x^*] - i, [y^*]-j)$ where i and j are nonnegative integers satisfying $(i,j) \neq (0,0)$. Without loss of generality, assume that the scheme was stopped by attempting to take a step in the x-direction. The stopping criteria yields

$$\begin{aligned}
&F([x^*] - i + 1, [y^*] - j) - F([x^*] - i, [y^*] - j) \\
&\qquad = 2a[x^*] - 2ai + a + c([y^*] - j) - d \\
&\qquad = 2a(x^* - \epsilon) - 2ai + a + c(y^* - \tau - j) - d \\
&\qquad = F_x(x^*, y^*) - a(2\epsilon + 2i - 1) - c(\tau + j) \\
&\qquad = -a(2\epsilon + 2i - 1) - c(\tau + j) \geq 0
\end{aligned}$$

Notice that if $i \geq 1$, then the above inequality cannot hold. Consequently, the stopping criteria is not satisfied and the scheme is not stopped at $([x^*]-i, [y^*]-j)$.

If $i = 0$ then j is an integer satisfying $j \geq 1$ and thus

$$\begin{aligned}
&F([x^*] - i + 1, [y^*] - j) - F([x^*] - i, [y^*] - j) \\
&\qquad = F([x^*] + 1, [y^*] - j) - F([x^*], [y^*] - j) \\
&\qquad = -a(2\epsilon - 1) - c(\tau + j) \geq 0
\end{aligned}$$

with $\tau + j \geq 1$. If the point $([x^*], [y^*] - j)$ is above the line L, and the next attempted step is to go to $([x^*]+1, [y^*]-j)$, then, as shown above, ϵ must satisfy $\epsilon > 1/2$. This inequality together with $(\tau+j) \geq 0$ implies that the inequality that is necessary to satisfy the stopping criteria cannot hold. Therefore, the scheme does not stop at $([x^*], [y^*] - j)$ with $j \geq 1$.

It now follows that the sequence of iterates will contain at least one of the corners of Γ. If $([x^*], [y^*])$ is obtained, then the scheme will either stop or continue on to one of the points $([x^*] + 1, [y^*])$ or $([x^*], [y^*] + 1)$. If $([x^*] + 1, [y^*])$ or $([x^*], [y^*] + 1)$ is in the sequence, then either the stopping criteria is satisfied or the scheme moves on to $([x^*] + 1, [y^*] + 1)$. As shown above, the scheme stops if $([x^*] + 1, [y^*] + 1)$ is obtained.

Clearly, all corner points of Γ satisfy the desired inequalities. That is, each component is within one unit of the corresponding component of (x^*, y^*). The point $([x^*], [y^*])$ could be obtained in $[x^*] + [y^*]$ steps while $[x^*] + [y^*] + 2$ steps guarantees that the scheme will be at the point $([x^*] + 1, [y^*] + 1)$. Thus the scheme will stop in k steps with $[x^*] + [y^*] \leq k \leq [x^*] + [y^*] + 2$.

More evidence of the fact that the Line-Step Algorithm converges to an appropriate integer-valued approximation to the true minimum of equation (2.2) can be found by considering the same problem where, now, the scheme takes steps of size h in the positive x and positive y directions. Then, by a similar argument to the one presented in the theorem, it can be concluded that the scheme will stop at a point $(I, J) = (x_k, y_k)$ with $|x_k - x^*| \leq h$, $|y_k - y^*| \leq h$. Clearly, if h approaches 0, that is, if the step size diminishes to 0, then the sequence of iterates, (x_k, y_k), are exactly those points on the line L. Also the limiting inequalities $|x_k - x^*| \leq 0$, $|y_k - y^*| \leq 0$ hold. Therefore, $I = x_k = x^*$ and $J = y_k = y^*$, so the scheme has converged to the true minimum.

It should be evident now that the Line-Step Algorithm is more than a means of finding integer-valued approximations to quadratic functions. For instance, if the step size were chosen to be one-half, then the algorithm would converge to a point (I, J) where $|I - x^*| \leq \frac{1}{2}$ and $|J - y^*| \leq \frac{1}{2}$. Therefore, given any tolerance, τ, the Line-Step Algorithm can converge to a solution (I, J) with $|I - x^*| \leq \tau$ and $|J - y^*| \leq \tau$ by taking steps of size $h = \tau$.

Example 3.1 This example will illustrate the convergence of the Line-Step Algorithm. Consider the system depicted in Figure 3.4. For this configuration, the cost function is given by

$$Cost = F(x, y, z) = 2x^2 + (x - 2)^2 + (y - 4)^2 + y^2 + z^2$$

with the constraint

$$x + y + z = 12.$$

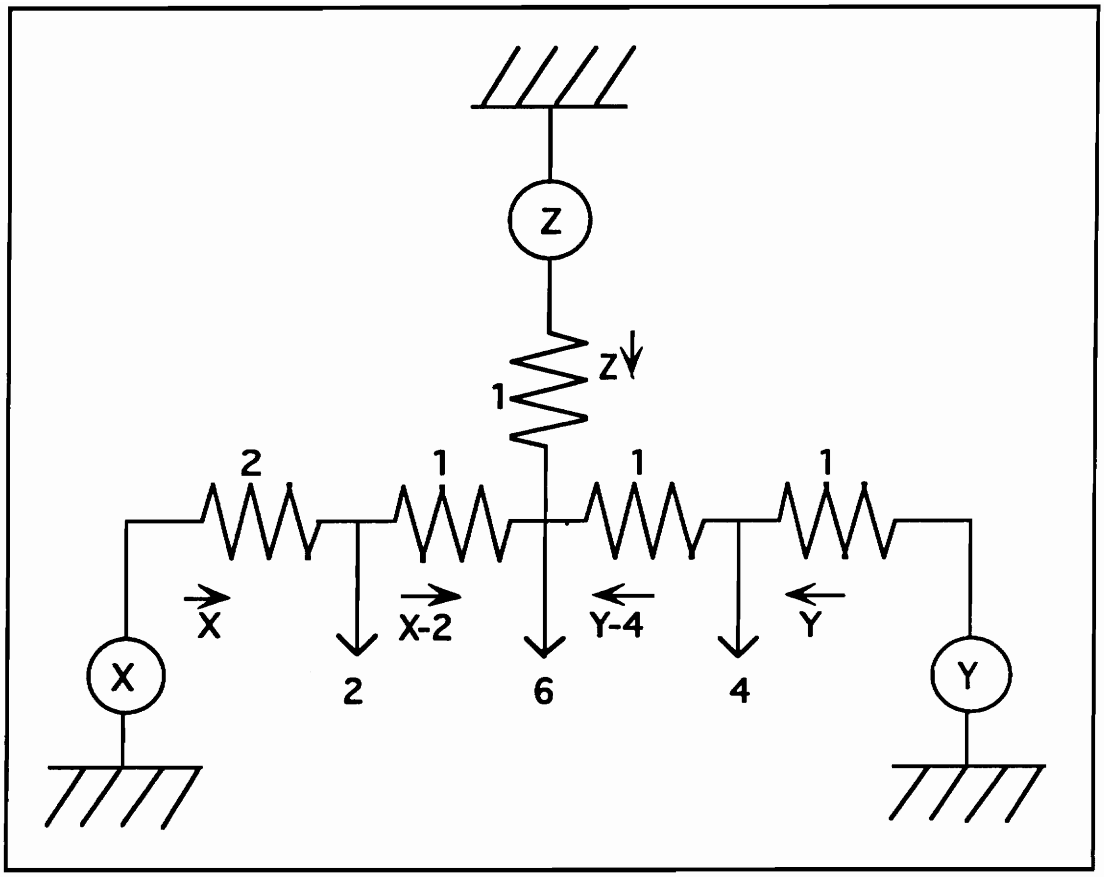

Figure 3.4: System for Example 3.1

Using this constraint, the cost function is reduced to

$$f(x,y) = 4x^2 + 3y^2 + 2xy - 28x - 32y + 164. \tag{3.1}$$

Using the FMINS function of MATLAB, the actual minimum of equation (3.1) is found to be $(x^*, y^*) = (2.3636, 4.5454)$. The Line-Step Algorithm is applied to equation (3.1) using a step size of $h = 1$.

Figure 3.5 shows the path of the scheme and the approximation point $(I, J) = (2, 5)$. Notice that $|I - x^*| = |2 - 2.3636| = 0.3636 < 1$ and $|J - y^*| = |5 - 4.5454| = 0.4546 < 1$. Next, the step size is reduced to $h = \frac{1}{2}$. Now, the algorithm converges to the point $(I, J) = (2.5, 4.5)$, as shown in Figure 3.6. Again, the difference $(I, J) - (x^*, y^*)$ yields

$|I - x^*| = |2.5 - 2.3636| = 0.1364 < .5$ and $|J - y^*| = |4.5 - 4.5454| = 0.0454 < .5$. Figure 3.7 depicts the results of applying the Line-Step Algorithm with a step size of $h = \frac{1}{4}$. This step size yields the point $(I, J) = (2.25, 4.5)$. Now the difference $(I, J) - (x^*, y^*)$ reveals $|I - x^*| = |2.25 - 2.3636| = 0.1136 < .25$ and $|J - y^*| = |4.5 - 4.5454| = 0.0454 < .25$. The scheme is applied once more with a step size of $h = \frac{1}{8}$. Figure 3.8 shows the path of the scheme and the approximation $(I, J) = (2.375, 4.5)$. Notice that $|I - x^*| = |2.375 - 2.3636| = 0.114 < .125 = h$ and $|J - y^*| = |4.5 - 4.5454| = 0.0454 < .125 = h$. It should now be clear that the scheme is converging to the actual minimum as smaller step sizes are chosen.

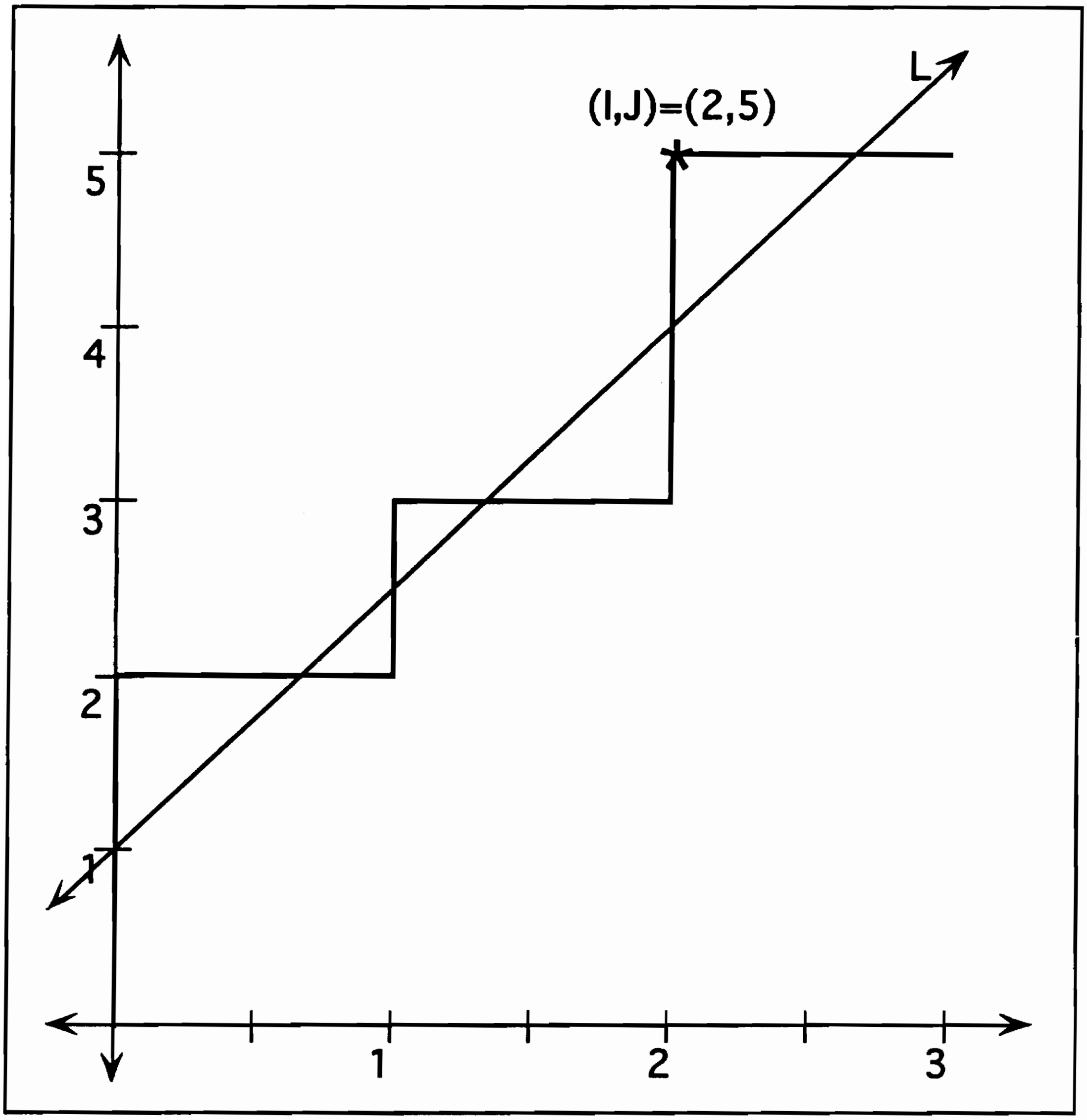

Figure 3.5: Solution for Equation (3.1) with Step Size of $h = 1$

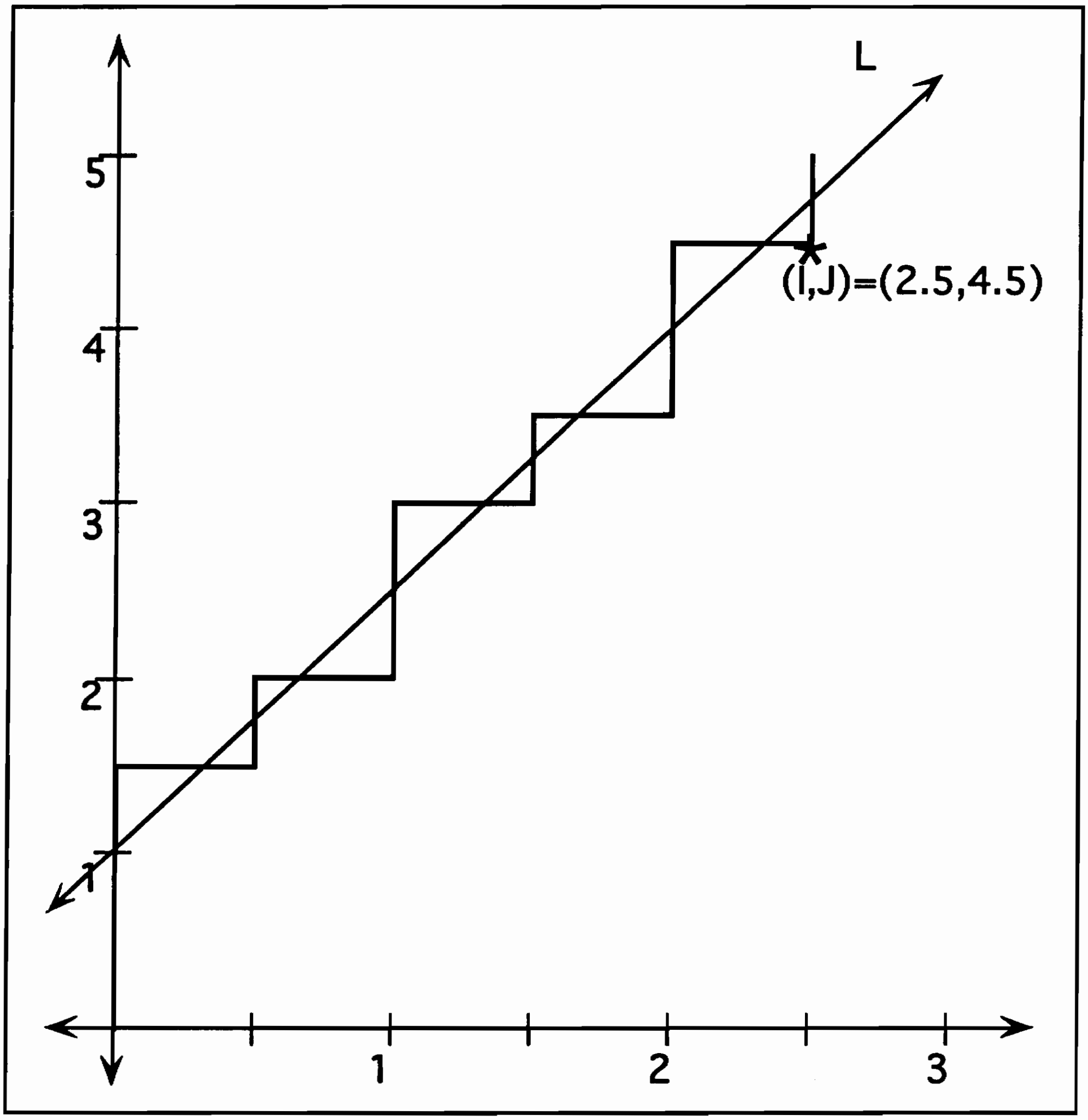

Figure 3.6: Solution for Equation (3.1) with Step Size of $h = \frac{1}{2}$

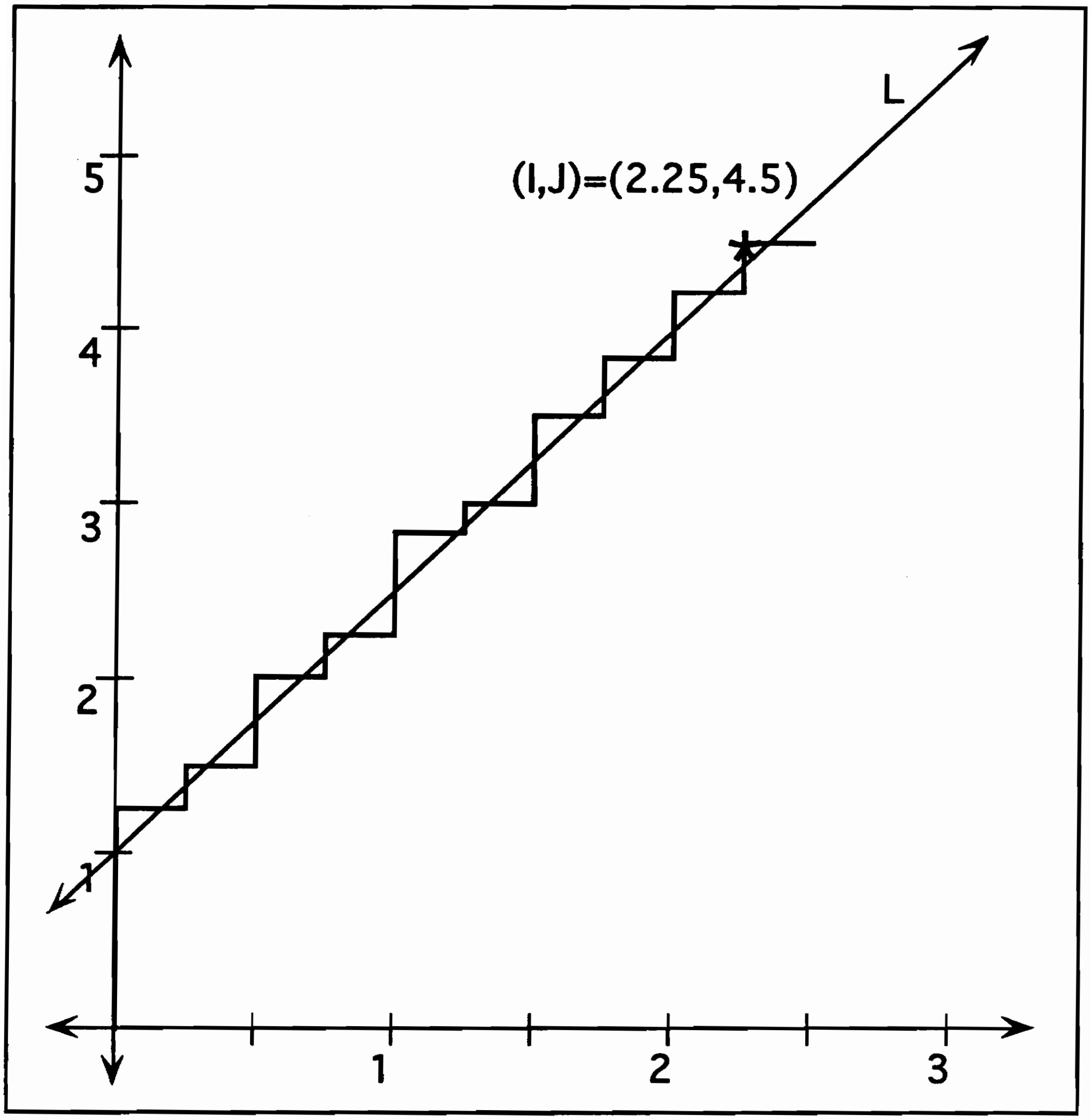

Figure 3.7: Solution for Equation (3.1) with Step Size of $h = \frac{1}{4}$

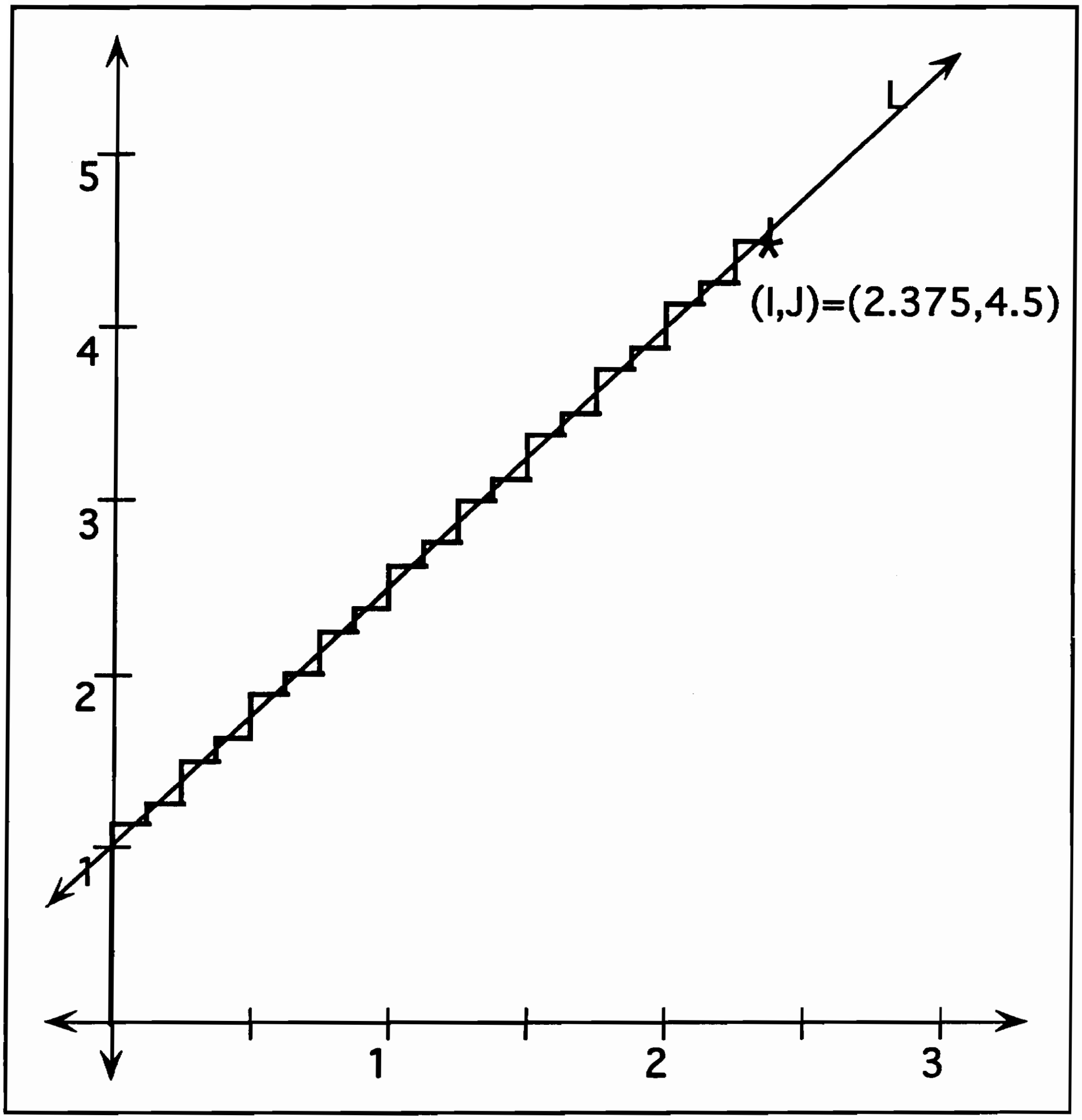

Figure 3.8: Solution for Equation (3.1) with Step Size of $h = \frac{1}{8}$

Chapter 4

ALTERNATE MINIMIZATION TECHNIQUES

4.1 The Lagrange Multiplier Method

The original problem considered was that of minimizing the cost function of a general three-source system with a linear equality constraint. Recall that the cost function for such a system is

$$Cost = F(x,y,z) = R_1x^2 + R_2(x-\alpha_1)^2 + R_3(y-\alpha_3)^2 + R_4y^2 + R_5z^2.$$

subject to

$$x + y + z = \alpha_1 + \alpha_2 + \alpha_3.$$

However, this constraint may be rewritten as

$$c(x,y,z) = x + y + z - (\alpha_1 + \alpha_2 + \alpha_3) = 0.$$

A classical approach to this constrained problem is to apply the Lagrange Multiplier method. This method involves introducing the Lagrangian function

$$\mathcal{L}(\mathbf{x},\lambda) = F(\mathbf{x}) - \lambda c(\mathbf{x})$$

where $\mathbf{x} = (x,y,z)$. Then the minimization problem reduces to one of finding $\mathbf{x}^*$ and λ^* which satisfy

$$\nabla\mathcal{L}(\mathbf{x}^*,\lambda^*) = 0.$$

This is equivalent to solving

$$\nabla F(\mathbf{x}) = \lambda\nabla c(\mathbf{x}). \tag{4.1}$$

The Lagrange multiplier of any constraint is a measure of the rate of change of F subject to changes in the constraint function [5].

4.2 The Equal Incremental Loss Algorithm

A new algorithm for Optimal Power Flow, called the Equal Incremental Loss (EIL) Algorithm, has been developed for minimizing constrained cost functions associated with power systems. The EIL method, a variation of the traditional Lagrange Multiplier method, yields an integer-valued solution for such problems.

The EIL Algorithm, a repetitive scheme which employs a discrete step size, converges to a solution that is within the step size of the optimal. Given the discrete step size and the total system load, the number of iterations required to reach the solution can be estimated by dividing the total load by the step size. The exact optimal solution may be obtained by varying the step size during the algorithm.

At the start of the algorithm, all system loads are assumed to be unsupplied, and as the EIL Algorithm progresses toward the solution, the loads are supplied in discrete step sizes. The initial loads to be evaluated for supply are those closest to the sources. With each step, line losses are calculated including only the previously supplied load currents and the line current due to the new load increment. The resulting function is referred to as a sub-cost function. The line losses due to supplying the load increment from every source is evaluated separately.

At each step, the algorithm answers the following question: Which combination of source and load will result in the next increment of load being supplied at minimum line loss? The source and load combination producing the minimum increase is selected. In the event that two or more sources produce the same minimum increase, an arbitrary choice of which source to increment is made. Only one source is incremented at each iteration. The algorithm continues in like manner until all loads are satisfied. When the algorithm is finished, the sources which should supply each load have been identified along with the power delivered by each source [2].

The basic steps followed by the EIL Algorithm are summarized below.

Step 1: Evaluate the sub-cost function associated with supplying the next increment of load at each ending load in the system.

Step 2: From the results of Step 1, choose the combination of the source and ending load which produces the minimum increase in loss.

Step 3: If all loads are fully supplied, the EIL Algorithm has converged; otherwise, return to Step 1.

4.3 Technique Comparisons

The comparison between the Equal Incremental Loss Algorithm and the Lagrange Multiplier method is best seen through a simple example.

Example 4.1 Consider the system in Figure 3.4. Using a load increment of one, the first step of the EIL Algorithm is to calculate the line loss associated with supplying one unit of power at each of the three sources. So there are three combinations to consider:

A - 1 unit at source x, 0 units at sources y and z

B - 1 unit at source y, 0 units at sources x and z

C - 1 unit at source z, 0 units at sources x and y

The associated cost function is given by $f(x,y,z) = 2x^2+y^2+z^2$. For case A, the constraint functions are $x = 1$, $y = 0$, $z = 0$. The resulting cost due to these constraints is 2. For case B, the constraint functions are $x = 0$, $y = 1$, $z = 0$. The cost subject to these constraints is 1. Lastly, for case C, the constraints $x = 0$, $y = 0$, $z = 1$ and the resulting cost is 1.

The EIL Algorithm now chooses the combination which produces the minimum increase in loss. For this example, both cases B and C yield the minimum increase, so without loss of generality, case B is chosen. Based on this choice, the next step of the algorithm yields three combinations to consider:

A - 1 unit at source x, 1 unit at source y, and 0 units at source z

B - 0 units at source x, 2 units at source y, and 0 units at source z

C - 0 units at source x, 1 unit at source y, and 1 unit at source z

Table 4.1 summarizes the twelve steps involved using the EIL Algorithm. The solution obtained using this technique indicates that, to satisfy the total load of 12, 2 units should be supplied by source x, 5 units by source y, and 5 units by source z.

Now, apply the Lagrange Multiplier method to each of the combinations from the EIL Algorithm. For the first step of the EIL Algorithm, there are again three combinations to consider:

A - 1 unit at source x, 0 units at sources y and z

B - 1 unit at source y, 0 units at sources x and z

C - 1 unit at source z, 0 units at sources x and y

The associated cost function is given by $f(x,y,z) = 2x^2 + y^2 + z^2$ and, for case A, the corresponding constraint functions are $x = 1$, $y = 0$, $z = 0$. So, the first Lagrange Multiplier problem is to solve

$$(4x, 0, 0) = \bar{\lambda}$$

where $\bar{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$ and λ_1 is the multiplier corresponding to the constraint $x = 1$, λ_2 is the multiplier corresponding to the constraint $y = 0$, and λ_3 is the multiplier corresponding to the constraint $z = 0$. It should be noted that, for any combination of constraint functions, the right hand side of equation (4.1), modified to the case of three constraint functions, will be $\lambda_1(1,0,0) + \lambda_2(0,1,0) + \lambda_3(0,0,1)$ because the constraint functions are always linear. So, for this combination, the solution is $\bar{\lambda} = (4,0,0)$. Similarly, for case B, the solution of the multiplier problem is $\bar{\lambda} = (0,2,0)$, and for case C, the solution is $\bar{\lambda} = (0,0,2)$. Now, for each of the three combinations, sum the values for λ_1, the values for λ_2, and the values for λ_3. The combination which will yield the smallest increase is the one in which the new load increment is in the direction of x if the sum of the values for λ_1 is less than those for λ_2 and λ_3. If the sum of the values for λ_2 is less than those for λ_1 and λ_3, then choose the combination in which the new load increment is in the direction of y. Similarly, choose the combination in which the new load increment is in the direction of z if the sum of the values for λ_1 and those for λ_2 are greater than the sum of the values for λ_3. In the event

Table 4.1. The EIL Algorithm

EIL	Source	Source	Source	Cost Function	Cost	EIL
Step	x	y	z	$f(x,y,z) =$		Choice
1	1	0	0	$2x^2 + y^2 + z^2$	2	
	0	1	0		1	←
	0	0	1		1	
2	1	1	0	$2x^2 + y^2 + z^2$	3	
	0	2	0		4	
	0	1	1		2	←
3	1	1	1	$2x^2 + y^2 + z^2$	4	←
	0	2	1		5	
	0	1	2		5	
4	2	1	1	$2x^2 + y^2 + z^2$	10	
	1	2	1		6	←
	1	1	2		6	
5	2	2	1	$2x^2 + y^2 + z^2$	13	
	1	3	1		12	
	1	2	2		10	←
6	2	2	2	$2x^2 + y^2 + z^2$	16	
	1	3	2		15	←
	1	2	3		15	
7	2	3	2	$2x^2 + y^2 + z^2$	21	
	1	4	2		22	
	1	3	3		20	←
8	2	3	3	$2x^2 + y^2 + z^2$	26	←
	1	4	3		27	
	1	3	4		27	
9	3	3	3	$3x^2 - 4x + 2y^2 - 8y + z^2 + 20$	37	
	2	4	3		33	←
	2	3	4		33	
10	3	4	3	$3x^2 - 4x + 2y^2 - 8y + z^2 + 20$	44	
	2	5	3		43	
	2	4	4		40	←
11	3	4	4	$3x^2 - 4x + 2y^2 - 8y + z^2 + 20$	51	
	2	5	4		50	
	2	4	5		49	←
12	3	4	5	$3x^2 - 4x + 2y^2 - 8y + z^2 + 20$	60	
	2	5	5		59	←
	2	4	6		60	

that the minimum sum occurs more than once at any step, then one could arbitrarily choose any combination with this value. So, for this step, the sum of the values for λ_1 is 4, the sum of the values for λ_2 is 2, and the sum of the values for λ_3 is 2. Therefore, one could choose either combination B or combination C. The relationship between the values for the sums of λ_i and the sub-cost functional values has not been fully explored. Therefore, this information is offered as an observation only. A summary of the Lagrange Multiplier method is given in Table 4.2. The end result is exactly the same as the result found by doing the EIL Algorithm.

In the next example, the correspondence between the Equal Incremental Loss Algorithm and the Line-Step Algorithm is established.

Example 4.2 Recall that the EIL Algorithm considers sub-cost functions for each step. Though this seems contrary to the action of the Line-Step Algorithm, the EIL Algorithm actually parallels the Line-Step Algorithm. At the k-th step, the EIL Algorithm is in fact minimizing the sub-cost function subject to the constraint $x + y + z = kl$ where l is the amount of load being incremented per step. This is easily seen by applying the Line-Step Algorithm to each of these sub-problems. Refer to Example 4.1 and the results of the EIL Algorithm in Table 4.1. The claim is that for Step 1, the EIL Algorithm is minimizing the sub-cost function given by $f(x, y, z) = 2x^2 + y^2 + z^2$ subject to $x + y + z = 1$. The algorithm chooses the point (0,1,0). When this subproblem is minimized using the Line-Step Algorithm, the result is (0,0). Since $z = 1-x-y$, the Line-Step Algorithm chooses the point (0,0,1). This point differs from that found by the EIL Algorithm, however, that scheme would also allow the choice of the point (0,0,1). Therefore, the two methods coincide at this step. Now, for Step 2, the EIL Algorithm is again minimizing $f(x, y, z) = 2x^2 + y^2 + z^2$ subject to $x + y + z = 2$. The point (0,1,1) is chosen. The result of applying the Line-Step Algorithm to this sub-problem is (1,0). Now, $z = 2-x-y$, so the scheme actually converges to the point (1,0,1). At first glance, it may appear that the two methods are stepping in different directions, however, this is not the case. In fact, the two methods again coincide

Table 4.2. The Lagrange Multiplier Method

Step	x	y	z	$\nabla f(x,y,z)$	λ	$\Sigma\lambda_i$	Choice
1	1	0	0	$(4x, 2y, 2z)$	(4,0,0)	$\Sigma\lambda_1 = 4$	
	0	1	0		(0,2,0)	$\Sigma\lambda_2 = 2$	←
	0	0	1		(0,0,2)	$\Sigma\lambda_3 = 2$	
2	1	1	0	$(4x, 2y, 2z)$	(4,2,0)	$\Sigma\lambda_1 = 4$	
	0	2	0		(0,4,0)	$\Sigma\lambda_2 = 8$	
	0	1	1		(0,2,2)	$\Sigma\lambda_3 = 2$	←
3	1	1	1	$(4x, 2y, 2z)$	(4,2,2)	$\Sigma\lambda_1 = 4$	←
	0	2	1		(0,4,2)	$\Sigma\lambda_2 = 8$	
	0	1	2		(0,2,4)	$\Sigma\lambda_3 = 8$	
4	2	1	1	$(4x, 2y, 2z)$	(8,2,2)	$\Sigma\lambda_1 = 16$	
	1	2	1		(4,4,2)	$\Sigma\lambda_2 = 8$	←
	1	1	2		(4,2,4)	$\Sigma\lambda_3 = 8$	
5	2	2	1	$(4x, 2y, 2z)$	(8,4,2)	$\Sigma\lambda_1 = 16$	
	1	3	1		(4,6,2)	$\Sigma\lambda_2 = 14$	
	1	2	2		(4,4,4)	$\Sigma\lambda_3 = 8$	←
6	2	2	2	$(4x, 2y, 2z)$	(8,4,4)	$\Sigma\lambda_1 = 16$	
	1	3	2		(4,6,4)	$\Sigma\lambda_2 = 14$	←
	1	2	3		(4,4,6)	$\Sigma\lambda_3 = 14$	
7	2	3	2	$(4x, 2y, 2z)$	(8,6,4)	$\Sigma\lambda_1 = 16$	
	1	4	2		(4,8,4)	$\Sigma\lambda_2 = 22$	
	1	3	3		(4,6,6)	$\Sigma\lambda_3 = 14$	←
8	2	3	3	$(4x, 2y, 2z)$	(8,6,6)	$\Sigma\lambda_1 = 16$	←
	1	4	3		(4,8,6)	$\Sigma\lambda_2 = 20$	
	1	3	4		(4,6,8)	$\Sigma\lambda_3 = 20$	
9	3	3	3	$(6x-4, 4y-8, 2z)$	(14,6,6)	$\Sigma\lambda_1 = 30$	
	2	4	3		(8,8,6)	$\Sigma\lambda_2 = 20$	←
	2	3	4		(8,6,8)	$\Sigma\lambda_3 = 20$	
10	3	4	3	$(6x-4, 4y-8, 2z)$	(14,8,6)	$\Sigma\lambda_1 = 30$	
	2	5	3		(8,12,6)	$\Sigma\lambda_2 = 28$	
	2	4	4		(8,8,8)	$\Sigma\lambda_3 = 22$	←
11	3	4	4	$(6x-4, 4y-8, 2z)$	(14,8,8)	$\Sigma\lambda_1 = 30$	
	2	5	4		(8,12,8)	$\Sigma\lambda_2 = 28$	
	2	4	5		(8,8,10)	$\Sigma\lambda_3 = 26$	←
12	3	4	5	$(6x-4, 4y-8, 2z)$	(14,8,10)	$\Sigma\lambda_1 = 30$	
	2	5	5		(8,12,10)	$\Sigma\lambda_2 = 28$	←
	2	4	6		(8,8,12)	$\Sigma\lambda_3 = 32$	

after Step 3. For Step 3, the EIL Algorithm minimizes $f(x,y,z) = 2x^2 + y^2 + z^2$ subject to $x + y + z = 3$, and chooses the point (1,1,1). The Line-Step Algorithm also converges to the point (1,1) when applied to this sub-problem. This is actually the point (1,1,1) since $z = 3 - x - y$. So, when the EIL Algorithm has the option of choosing two paths to follow, the point chosen may differ from that chosen by the Line-Step Algorithm, but after two steps, the two schemes arrive at the same point. See Table 4.3 for the results of the remaining nine steps.

Table 4.3. A Comparison between the EIL Algorithm and the Line-Step Algorithm

Step	Cost Function $f(x,y,z) =$	Constraint $x+y+z=$	EIL Choice	Line-Step Choice
1	$2x^2+y^2+z^2$	1	(0,1,0)	(0,0,1)
2	$2x^2+y^2+z^2$	2	(0,1,1)	(1,0,1)
3	$2x^2+y^2+z^2$	3	(1,1,1)	(1,1,1)
4	$2x^2+y^2+z^2$	4	(1,2,1)	(1,1,2)
5	$2x^2+y^2+z^2$	5	(1,2,2)	(1,2,2)
6	$2x^2+y^2+z^2$	6	(1,3,2)	(1,2,3)
7	$2x^2+y^2+z^2$	7	(1,3,3)	(1,3,3)
8	$2x^2+y^2+z^2$	8	(2,3,3)	(2,3,3)
9	$3x^2-4x+2y^2-8y+z^2+20$	9	(2,4,3)	(2,4,3)
10	$3x^2-4x+2y^2-8y+z^2+20$	10	(2,4,4)	(2,4,4)
11	$3x^2-4x+2y^2-8y+z^2+20$	11	(2,4,5)	(2,4,5)
12	$3x^2-4x+2y^2-8y+z^2+20$	12	(2,5,5)	(2,5,5)

Chapter 5

REAL MODELS AND INEQUALITY CONSTRAINTS

Now, several examples are presented in which the Line-Step Algorithm is applied to minimize quadratic functions.

Example 5.1 Suppose that quadratic function

$$f(x,y) = 25x^2 - 50x + 4y^2 - 32y + 89 \tag{5.1}$$

is to be minimized. One advantage of the Line-Step Algorithm is that it can minimize some functions with fewer iterations than other methods. In this example, the Line-Step Algorithm is compared to the method of Steepest Descent. Recall that the method of Steepest Descent is an iterative process that begins with an initial value $\bar{\mathbf{x}}_o$ and each successive iterative is given by

$$\bar{\mathbf{x}}_{k+1} = \bar{\mathbf{x}}_k + \alpha_{k+1}\bar{r}_k$$

where

$$\bar{r}_k = -\nabla F(\bar{\mathbf{x}}_k)$$

and

$$\alpha_{k+1} = \frac{(\bar{r}_k, \bar{r}_k)}{(A\bar{r}_k, \bar{r}_k)}.$$

Now the method of Steepest Descent is applied to equation (5.1). The results after 20 iterations are given in Table 5.1. Figure 5.1 shows the path of the Steepest Descent iterates and a contour plot of the function given in equation (5.1). Clearly, the true minimum of equation (5.1) is given by $(x^*, y^*) = (1, 4)$. The Steepest Descent iterates are oscillating about the minimum; that is, they are changing direction at each iteration.

Table 5.1. Steepest Descent

k	$\bar{\mathbf{x}}_k$	k	$\bar{\mathbf{x}}_k$	k	$\bar{\mathbf{x}}_k$
0	$\bar{\mathbf{x}}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$	7	$\bar{\mathbf{x}}_7 = \begin{bmatrix} 1.034887 \\ 3.65944 \end{bmatrix}$	14	$\bar{\mathbf{x}}_{14} = \begin{bmatrix} 0.994452 \\ 3.977783 \end{bmatrix}$
1	$\bar{\mathbf{x}}_1 = \begin{bmatrix} 1.323 \\ 0.84672 \end{bmatrix}$	8	$\bar{\mathbf{x}}_8 = \begin{bmatrix} 0.948559 \\ 3.794274 \end{bmatrix}$	15	$\bar{\mathbf{x}}_{15} = \begin{bmatrix} 1.001791 \\ 3.982485 \end{bmatrix}$
2	$\bar{\mathbf{x}}_2 = \begin{bmatrix} 0.5239 \\ 2.0949 \end{bmatrix}$	9	$\bar{\mathbf{x}}_9 = \begin{bmatrix} 1.016615 \\ 3.837822 \end{bmatrix}$	16	$\bar{\mathbf{x}}_{16} = \begin{bmatrix} 0.997357 \\ 3.989420 \end{bmatrix}$
3	$\bar{\mathbf{x}}_3 = \begin{bmatrix} 1.1538 \\ 2.4982 \end{bmatrix}$	10	$\bar{\mathbf{x}}_{10} = \begin{bmatrix} 0.9755 \\ 3.902031 \end{bmatrix}$	17	$\bar{\mathbf{x}}_{17} = \begin{bmatrix} 1.000853 \\ 3.991659 \end{bmatrix}$
4	$\bar{\mathbf{x}}_4 = \begin{bmatrix} 0.77322 \\ 3.09279 \end{bmatrix}$	11	$\bar{\mathbf{x}}_{11} = \begin{bmatrix} 1.0079 \\ 3.922769 \end{bmatrix}$	18	$\bar{\mathbf{x}}_{18} = \begin{bmatrix} 0.998741 \\ 3.994961 \end{bmatrix}$
5	$\bar{\mathbf{x}}_5 = \begin{bmatrix} 1.07325 \\ 3.28483 \end{bmatrix}$	12	$\bar{\mathbf{x}}_{12} = \begin{bmatrix} 0.988351 \\ 3.953346 \end{bmatrix}$	19	$\bar{\mathbf{x}}_{19} = \begin{bmatrix} 1.000406 \\ 3.996028 \end{bmatrix}$
6	$\bar{\mathbf{x}}_6 = \begin{bmatrix} 0.89199 \\ 3.567992 \end{bmatrix}$	13	$\bar{\mathbf{x}}_{13} = \begin{bmatrix} 1.003762 \\ 3.963222 \end{bmatrix}$	20	$\bar{\mathbf{x}}_{20} = \begin{bmatrix} 0.999400 \\ 3.997600 \end{bmatrix}$

Now, the Line-Step Algorithm is applied to equation (5.1). As seen in Figure 5.1, the Line-Step Algorithm has stopped at the point $(I, J) = (1, 4)$ in five steps. So, the Line-Step iterates converge to the exact minimum much faster than the method of Steepest Descent.

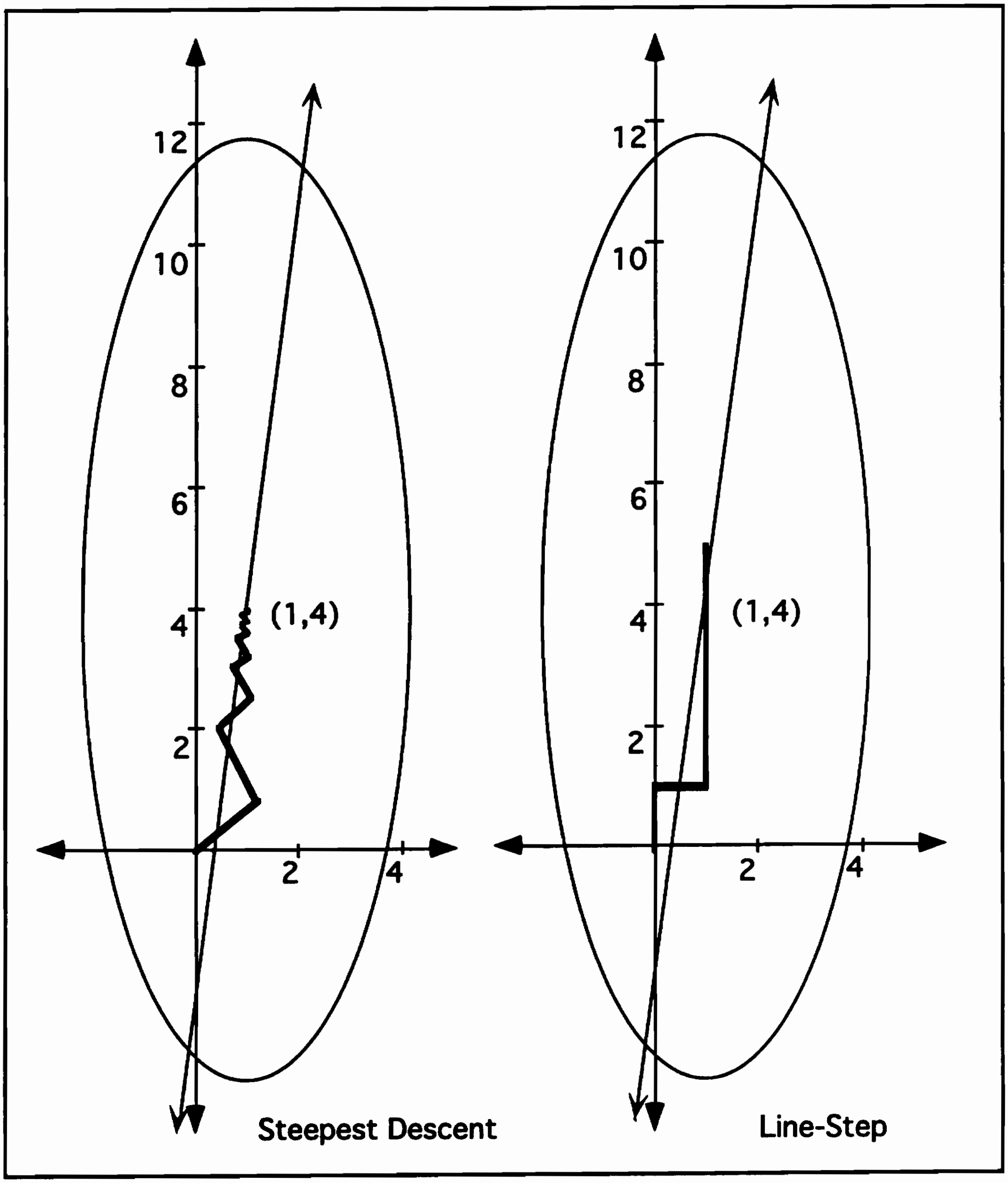

Figure 5.1: Steepest Descent Method vs. Line-Step Algorithm

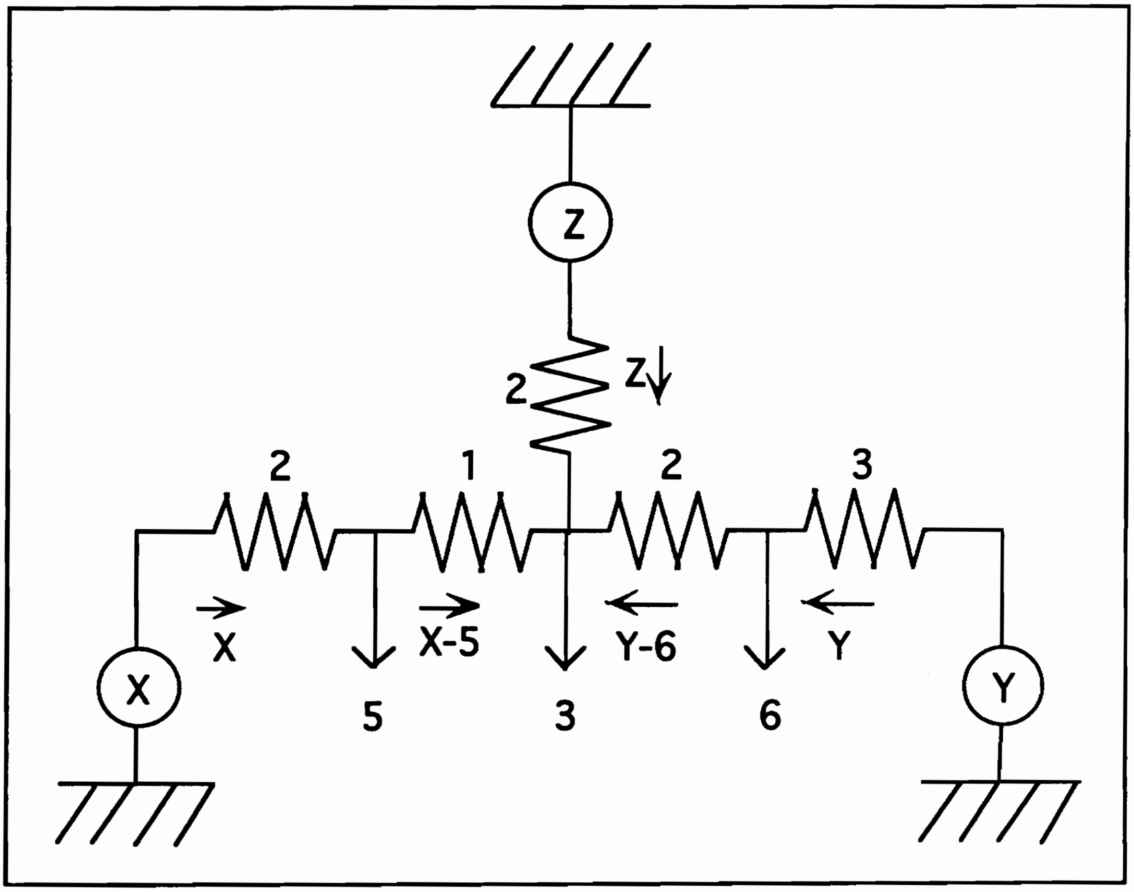

Figure 5.2: System for Example 5.2

Next, several systems are examined and the Line-Step Algorithm is applied to find the minimum of the cost function.

Example 5.2 For this example, consider the system depicted in Figure 5.2. The cost function for this system is given by

$$f(x,y,z) = 2x^2 + (x-5)^2 + 2(y-6)^2 + 3y^2 + 2z^2 \tag{5.2}$$

subject to the constraint

$$x + y + z = 14.$$

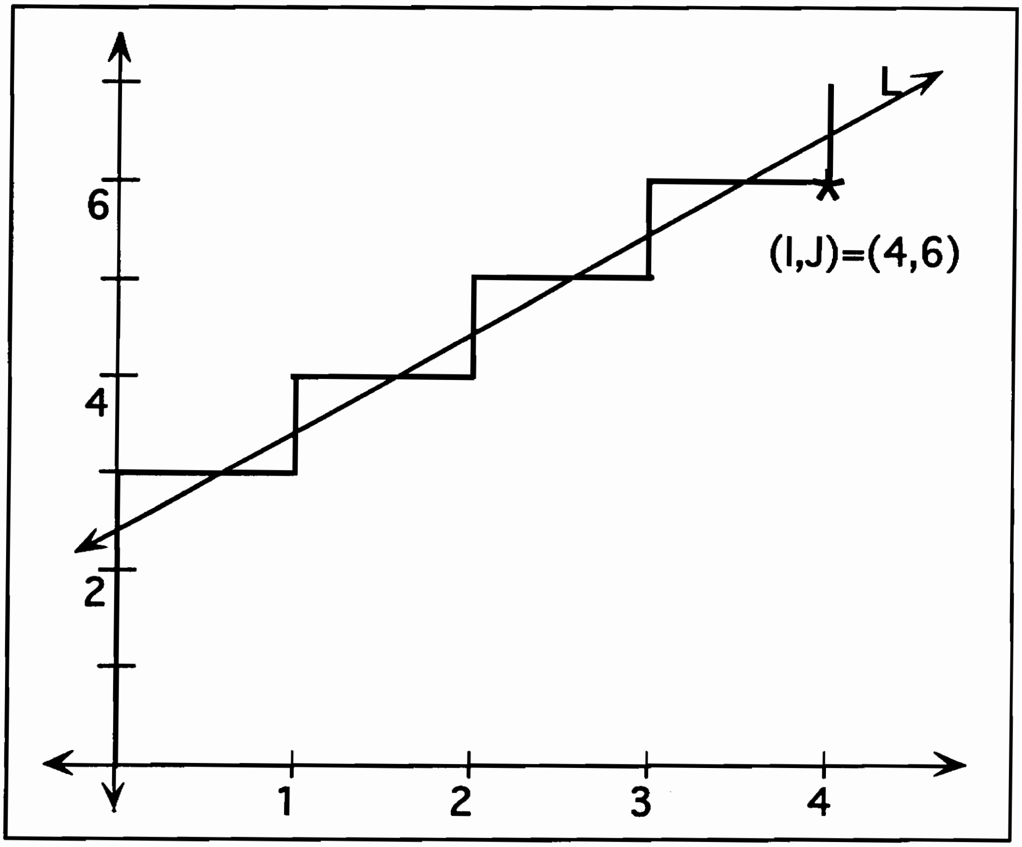

Figure 5.3: Solution of Equation (5.3)

Equation (5.2) may be reduced to a function of two variables by using the substitution

$$z = 14 - x - y.$$

By doing this reduction, the cost function becomes

$$f(x, y) = 5x^2 + 5y^2 + 4xy - 66x - 80y + 489. \tag{5.3}$$

When the Line-Step Algorithm is applied to equation (5.3) the result is $(I, J) = (4, 6)$. Comparing this answer to the true minimum, $(x^*, y^*) = (4.0476, 6.3809)$, reveals $|I - x^*| = |4 - 4.0476| = 0.0476 \leq 1$ and $|J - y^*| = |6 - 6.3809| = 0.3809 \leq 1$. See Figure 5.3 for the path taken by the algorithm.

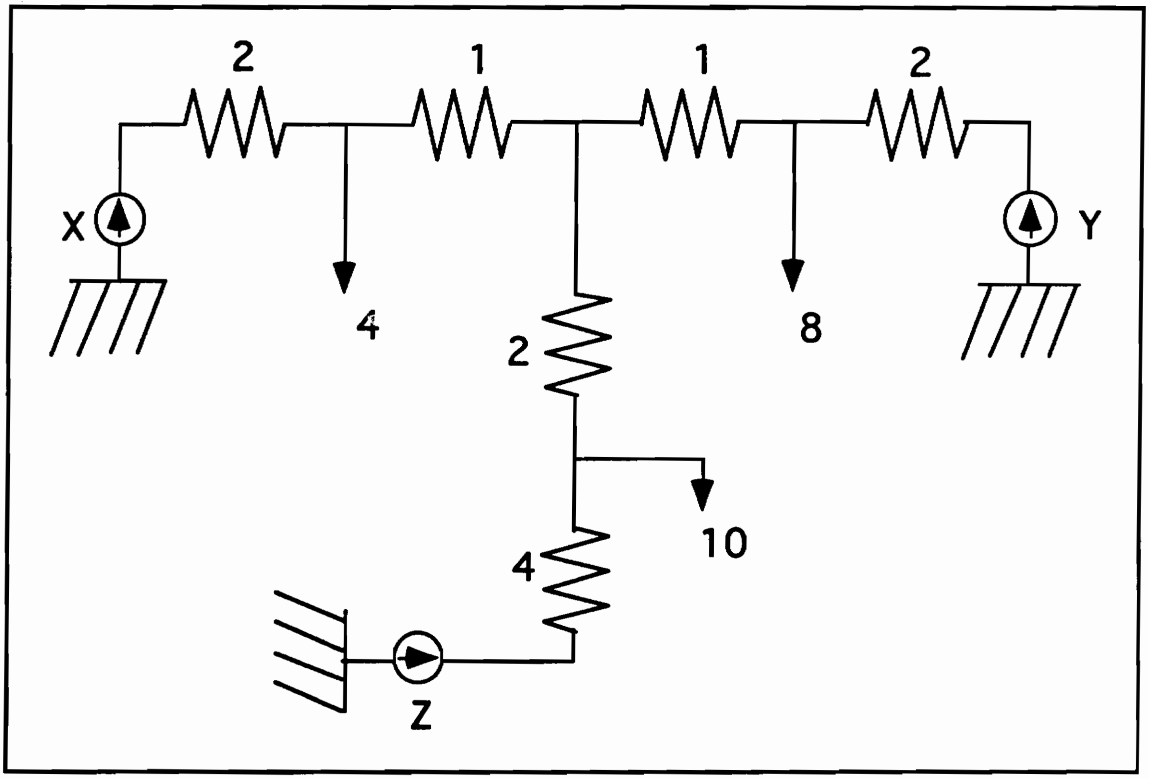

Figure 5.4: System for Example 5.3

Example 5.3 Next, the system in Figure 5.4 is considered. For this example, the cost function is

$$f(x, y, z) = 2x^2 + (x - 4)^2 + (y - 8)^2 + 2y^2 + 2(z - 10)^2 + 4z^2 \tag{5.4}$$

and the constraint has the form

$$x + y + z = 22.$$

Reducing equation (5.4) to one of two variables results in the following function:

$$f(x, y) = 9x^2 + 9y^2 + 12xy - 232x - 240y + 2304. \tag{5.5}$$

Using a step size of $h = 2$, the Line-Step Algorithm yields a minimum with coordinates $(I, J) = (8, 8)$ (see Figure 5.5). The true minimum of this function is $(x^*, y^*) = (7.20, 8.53)$.

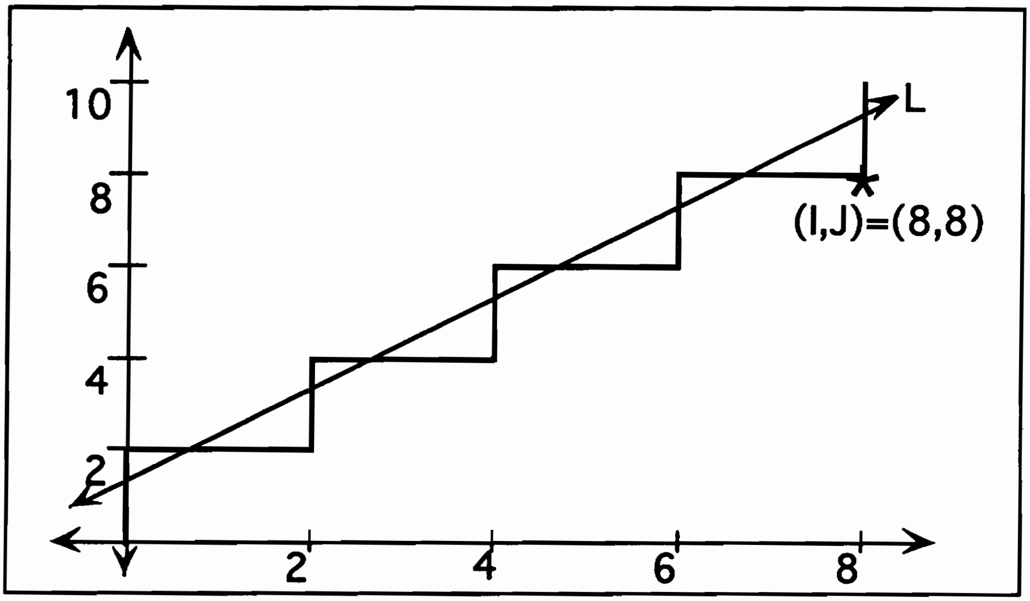

Figure 5.5: Solution for Equation (5.5) with Step Size of $h = 2$

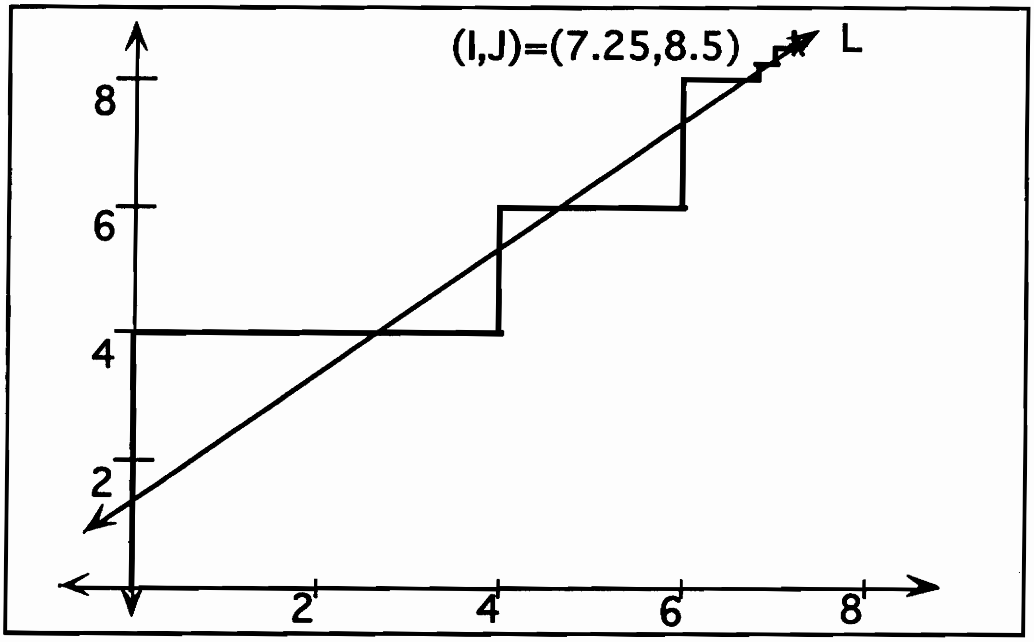

Figure 5.6: Solution for Equation (5.5) with Adaptive Step Size

Now, the algorithm is reapplied using an adaptive step size. The initial step size is $h = 4$. The step size is reduced until $h = 0.25$ which yields an approximation of $(I, J) = (7.25, 8.50)$. This result can be seen in Figure 5.6. Clearly, it is advantageous to use the adaptive step size to better approach the minimum.

Thus far, the problem under consideration has been to minimize the cost function subject to the constraint that the sum of the loads is equal to the sum of the output. However, under realistic conditions, power systems may be hampered by other constraints, both from within the system and from outside conditions. Examples of constraints which arise from within the system are limitations on the amount of power that a line may carry and operational limits of a plant. Outside factors include environmental constraints such as loading limits on equipment and plant emissions. When such conditions are taken into consideration, the complexity of the problem increases because inequality constraints must be introduced.

Now, a problem involving inequality constraints is examined. The constraints for this problem are on the amounts of power that the lines can carry. The lines are bounded both above and below. The upper bounds may be due to a constraint on the amount of generation of the plant. The lower bounds might be present because the system becomes unstable if generation levels fall below a certain amount.

The problem considered earlier is now revised to include inequality constraints. As before, a general three-source system is used as the model and the objective is to minimize the cost function given by

$$Cost = \hat{F}(x, y, z) = R_1 x^2 + R_2(x - \alpha_1)^2 + R_3(y - \alpha_3) + R_4 y^2 + R_5 z^2.$$

subject to the following constraints:

$$x + y + z = \alpha_1 + \alpha_2 + \alpha_3$$

$$\delta_1 \leq x \leq \gamma_1$$

$$\delta_2 \leq y \leq \gamma_2$$

$$\delta_3 \leq z \leq \gamma_3.$$

Using the same substitution as given by equation (1.2) yields

$$\delta_1 \leq x \leq \gamma_1 \tag{5.6}$$

$$\delta_2 \leq y \leq \gamma_2 \tag{5.7}$$

$$\delta_3 \leq \alpha_1 + \alpha_2 + \alpha_3 - (x + y) \leq \gamma_3. \tag{5.8}$$

Further calculations on equation (5.8) yield

$$(\alpha_1 + \alpha_2 + \alpha_3) - \gamma_3 \leq x + y \leq (\alpha_1 + \alpha_2 + \alpha_3) - \delta_3.$$

Also, combining equations (5.6) and (5.7) yields

$$\delta_1 + \delta_2 \leq x + y \leq \gamma_1 + \gamma_2.$$

Therefore, $x + y$ must satisfy two sets of inequalities, namely

$$\delta_1 + \delta_2 \leq x + y \leq \gamma_1 + \gamma_2$$

and

$$(\alpha_1 + \alpha_2 + \alpha_3) - \gamma_3 \leq x + y \leq (\alpha_1 + \alpha_2 + \alpha_3) - \delta_3.$$

To fulfill both of these requirements, take the lower bound of $x + y$ to be the maximum of $\delta_1 + \delta_2$ and $(\alpha_1 + \alpha_2 + \alpha_3) - \delta_3$ and take the upper bound to be the minimum of $\gamma_1 + \gamma_2$ and $(\alpha_1 + \alpha_2 + \alpha_3) - \delta_3$. In summary,

$$max\{\delta_1 + \delta_2, (\alpha_1 + \alpha_2 + \alpha_3) - \gamma_3\} \leq x + y \leq min\{\gamma_1 + \gamma_2, (\alpha_1 + \alpha_2 + \alpha_3) - \delta_3\}.$$

Let $p = max\{\delta_1 + \delta_2, (\alpha_1 + \alpha_2 + \alpha_3) - \gamma_3\}$ and $q = min\{\gamma_1 + \gamma_2, (\alpha_1 + \alpha_2 + \alpha_3) - \delta_3\}$. Then, $p \leq x + y \leq q$. Consider $p = x + y$. Then $y = p - x$ defines a line in the xy-plane which serves as a lower bound for values of $x + y$. Similarly, $x + y = q$ yields the line $y = q - x$ in the xy-plane which serves as an upper bound for values of $x + y$. This can be seen in Figure 5.7 where the shaded region defines feasible points (x, y).

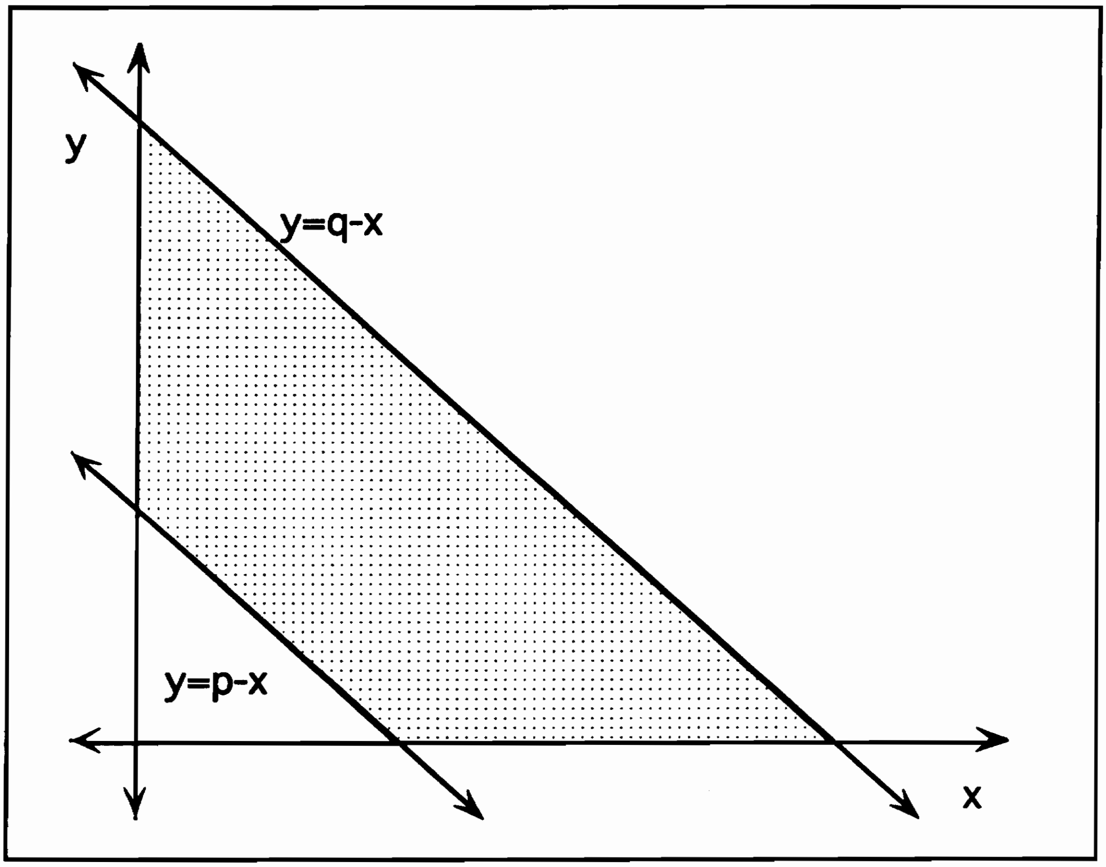

Figure 5.7: Region of Feasible Points for Inequality Constraints

Example 5.4 The system in Figure 5.8 provided by Arkansas Power and Light will be considered in this example. The cost function for this system is given by

$$f(x,y,z) = 1.767x^2+0.57y^2+4.389z^2-672.488z-1188.817y-1764.685z+155739.52 \quad (5.9)$$

with the constraint

$$x+y+z=753.066.$$

Using the substitution $z = 753.066 - x - y$, equation (5.9) is reduced to

$$f(x,y) = 6.156x^2 + 4.959y^2 + 8.778xy - 5518.2166x - 4964.5458y + 1315854.3. \quad (5.10)$$

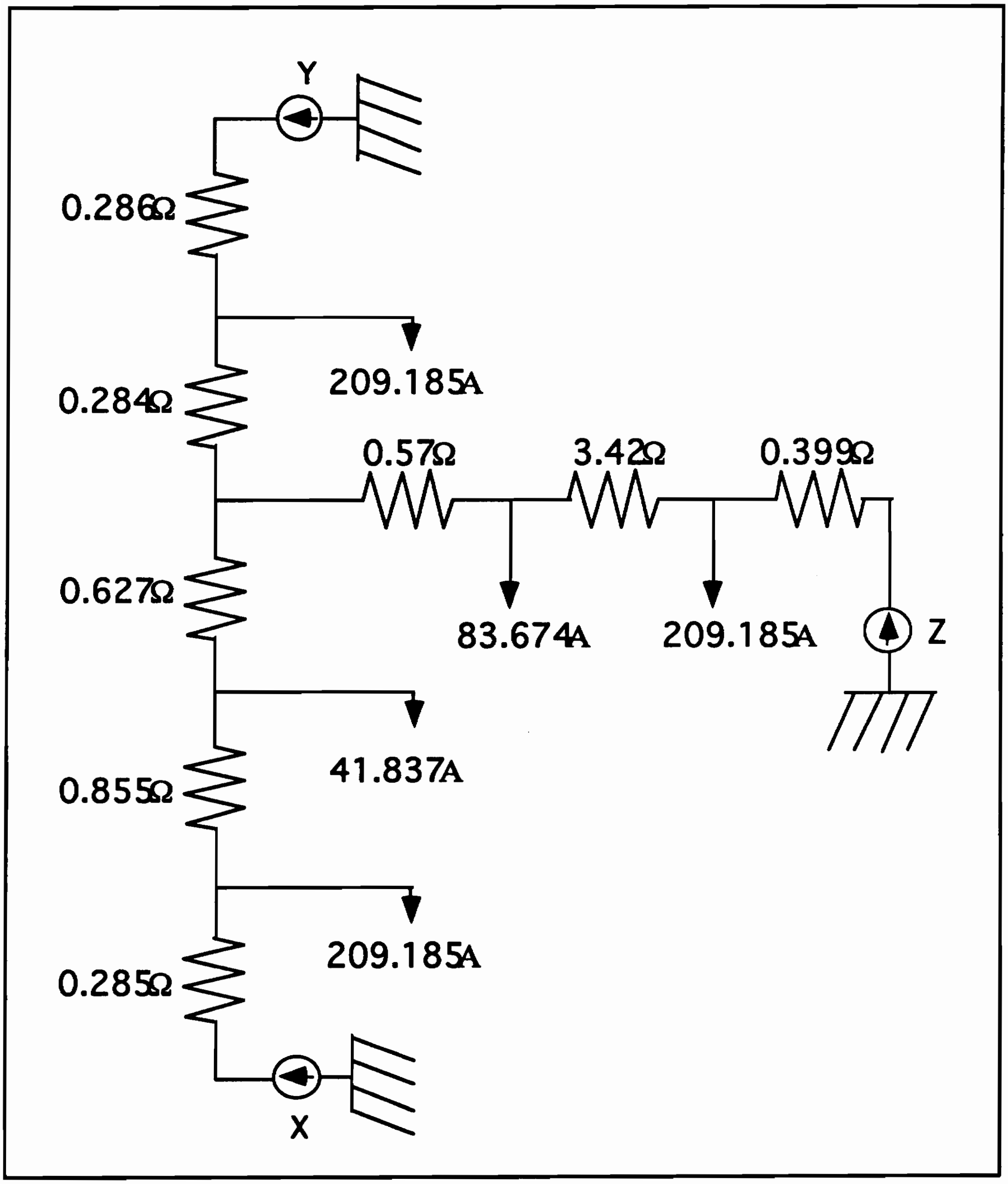

Figure 5.8: Arkansas Power and Light System for Example 5.4

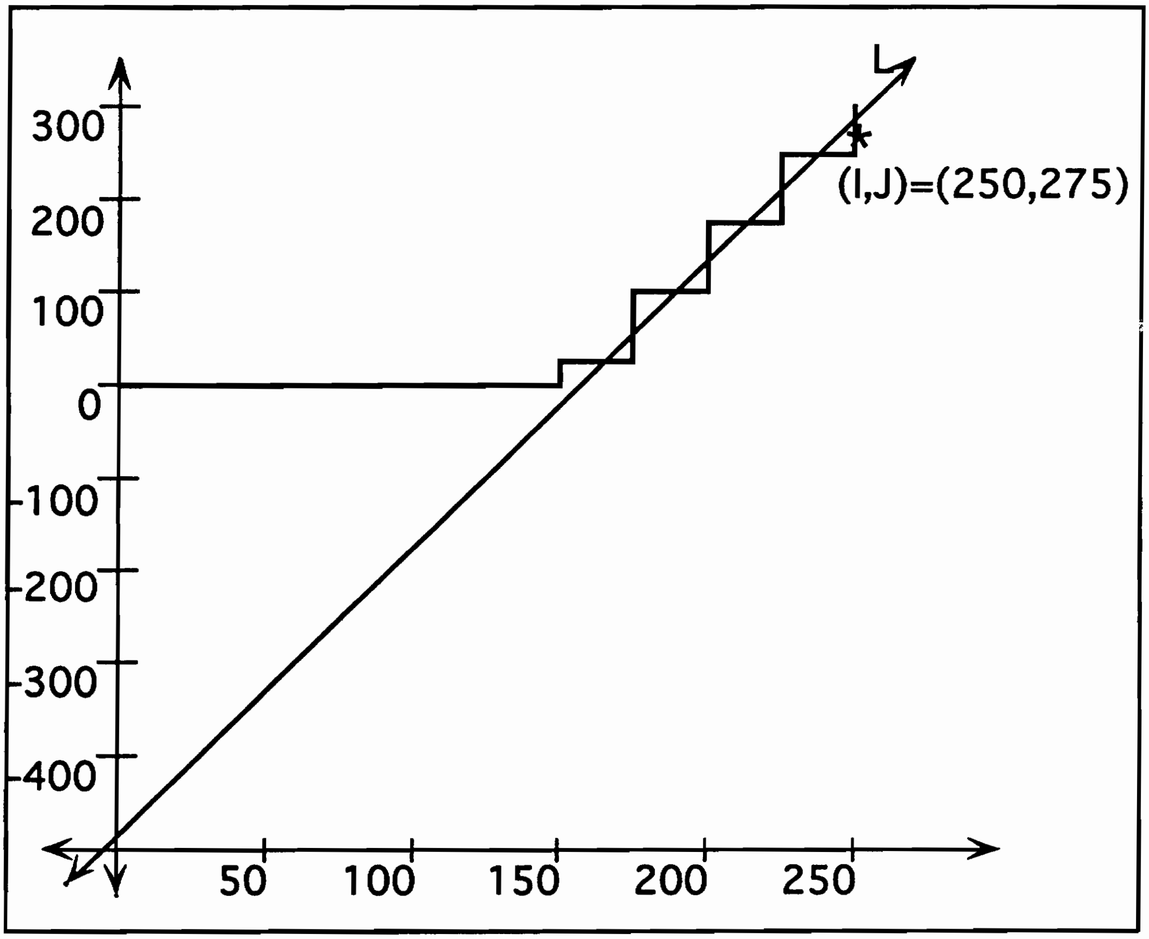

Figure 5.9: Solution for Equation (5.10) with Step Size of $h = 25$

The Line-Step Algorithm is applied to equation (5.10) using a step size of $h = 25$. The scheme converges to the point $(I, J) = (250, 275)$ as seen in Figure 5.9. The true minimum of equation (I) is $(x^*, y^*, z^*) = (247.4834, 281.5222, 224.0605)$. The Equal Incremental Loss Algorithm, when applied to the same system, yields a solution of $x = 250$, $y = 275$, $z = 225$.

Example 5.5 Continuing with the system in Figure 5.8, the Line-Step Algorithm is now applied using an adaptive step size. Initially, the step size is $h = 64$. Once the scheme converges to the point $(I, J) = (x_k, y_k)$, the process repeats starting from the point

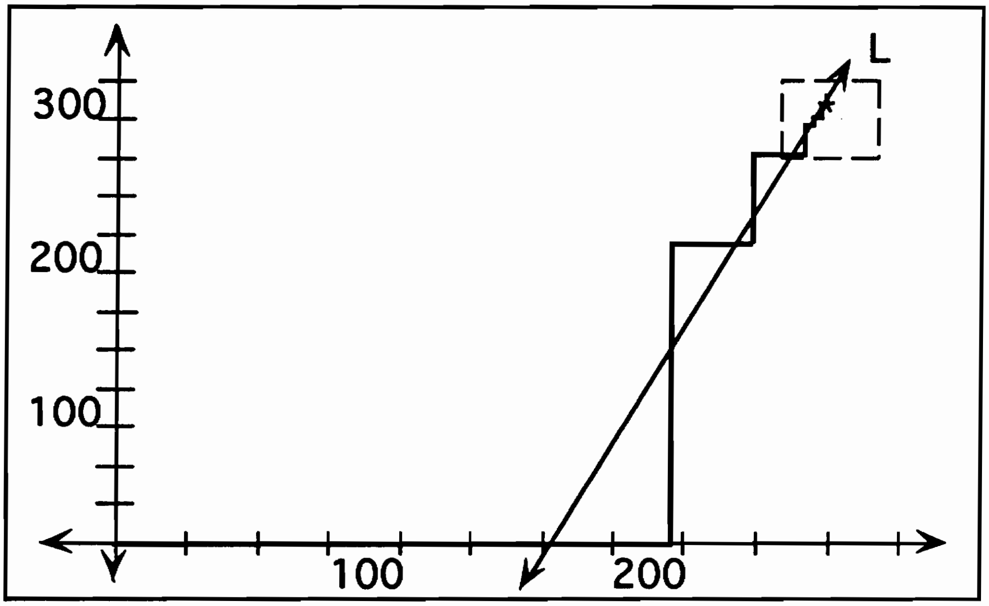

Figure 5.10: Solution for Equation (5.10) with Adaptive Step Size

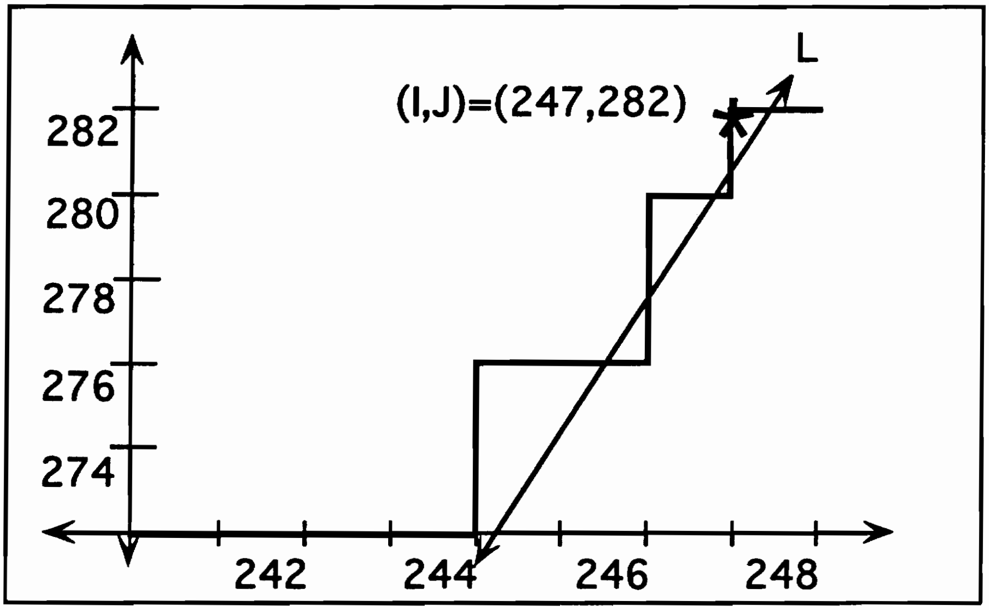

Figure 5.11: Detailed Graph of Figure 5.10

(x_{k-2}, y_{k-2}) and uses a step size of $h = 32$. This is repeated until $h = 1$. Figures 5.10 and 5.11 illustrate the path of the algorithm and its point of convergence, $(I, J) = (247, 282)$.

Example 5.6 The cost function for the system in Figure 5.8 was minimized in Example 5.4. Although the results in that example are correct, they do not reflect the best solution for the system because in actuality this particular system, provided by Arkansas Power and Light, has a constraint on the generation levels of the sources. The solution found in Example 5.4 violates a constraint on this system.

The generation levels for each of the sources are given by:

$$75 \leq x \leq 265$$

$$60 \leq y \leq 250$$

$$35 \leq z \leq 300.$$

Clearly, the solution from Example 5.4, $(x, y, z) = (250, 275, 225)$, violates the constraint on source y since that source cannot output more than 250 units of power. So, the Line-Step Algorithm is now reapplied to this system taking these inequality constraints into consideration. The step size used is $h = 25$. The results are $(x, y, z) = (250, 250, 250)$. Figure 5.12 shows the path taken by the algorithm as well as the region of feasible points (x, y). Notice that the algorithm now begins at the minimum values for x and y, namely, $(x_0, y_0) = (75, 60)$. By beginning here instead of the origin, the number of steps required to reach the region of feasible points is reduced.

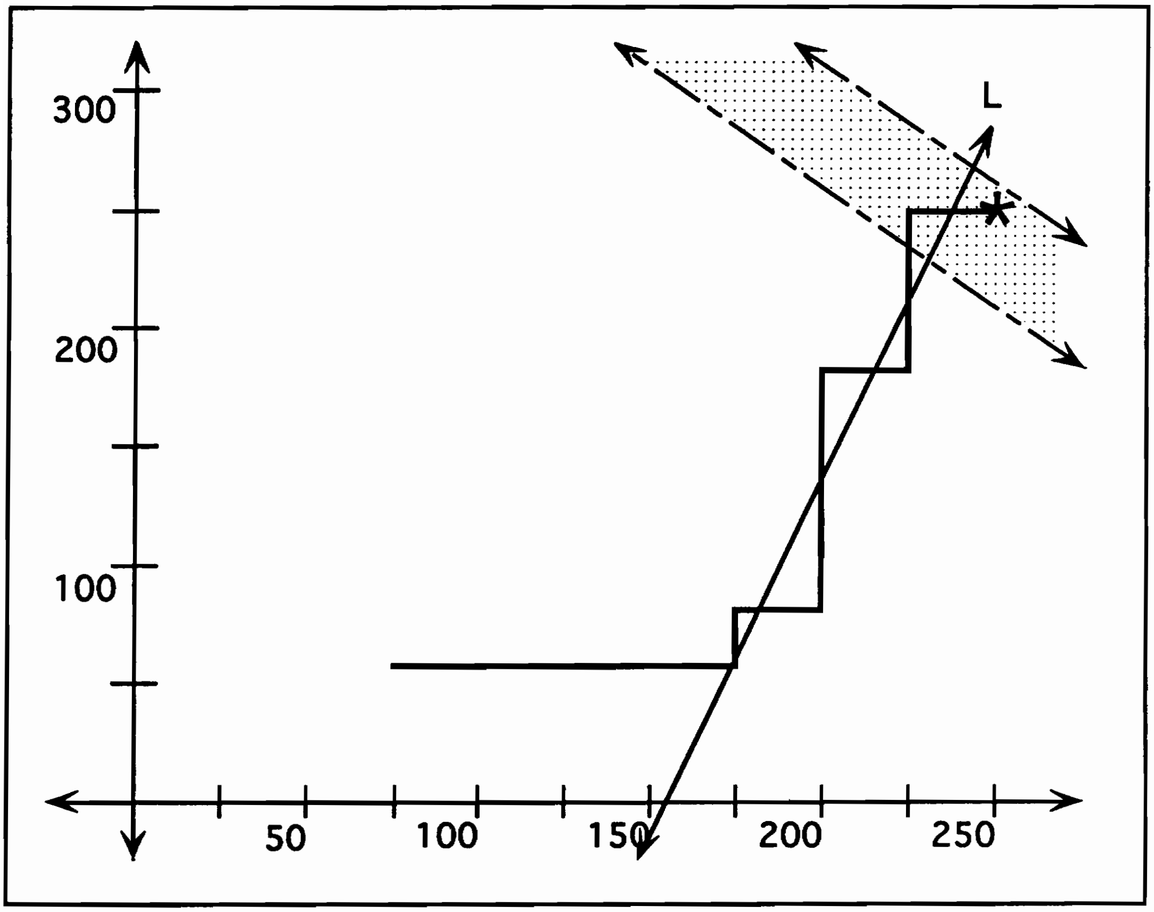

Figure 5.12: Solution for Equation (5.10) and Region of Feasible Points

REFERENCES

[1] Alsac, O., Bright, J., Prais, M., Stott, B., Further Developments in LP-Based Optimal Power Flow, *IEEE Transactions on Power Systems*, Vol. 5, No. 3, pp 697-706, August 1990.

[2] Broadwater, R., Dolloff, P., Herdman, T., Karamikhova, R., Applying the Equal Incremental Loss Algorithm to Optimal Economic Dispatch, preprint.

[3] Broadwater, R., Dolloff, P., Herdman, T., Karamikhova, R., Sargent, A., A New Algorithm for Optimal Power Flow in Distribution Systems: Equal Incremental Loss Algorithm, preprint.

[4] El-Hawary, M., Mbamalu, G., Stochastic Optimal Load Flow using a Combined Quasi-Newton and Conjugate Gradient Technique, *Electrical Power & Energy Systems*, Vol. 11, No. 2, pp 85-93, April 1989.

[5] Fletcher, R., *Practical Methods of Optimization*, Second Edition, John Wiley and Sons, New York, 1987.

[6] Franklin, J., *Matrix Theory*, Prentice Hall Inc., New Jersey, 1968.

[7] Gungor, B., *Power Systems*, Harcourt Brace Jovanovich, San Diego, 1988.

[8] Gill, P., Murray, W., Wright, M., *Practical Optimization*, Academic Press, New York, 1981.

[9] Marsden, J., Tromba, A., *Vector Calculus*, Third Edition, W. H. Freeman and Company, New York, 1988.

Appendix A

MATLAB CODES FOR LINE-STEP ALGORITHM

File Name: Smthesis.m

% Input a vector of the form DATA=[a b c d e]
% a = the coefficient of the x^2 term in the cost function
% b = the coefficient of the y^2 term in the cost function
% c = the coefficient of the xy term in the cost function
% d = the coefficient of the x term in the cost function
% e = the coefficient of the y term in the cost function
% m = the slope of the line $F_x = F_y$
% by = the y-intercept of the line $F_x = F_y$
% f = the cost function
% x = the x component of the approximation
% y = the y component of the approximation
% Fx = the partial of f with respect to x
% Fy = the partial of f with respect to y
% n = the intersection point of $x(i)$ and $F_x = F_y$
% gin = the greatest integer of n
% nstar = the maximum number of steps that may be taken in the y direction per iteration
% H = the distance in x between the point $y(i + j)$ and the line
% W = the distance in y between the point $x(i + j)$ and the line
% D = the intersection point of $y(i)$ and $F_x = F_y$
% gid = the greatest integer of D
% dstar = the maximum number of steps that may be taken in the x direction per iteration
% Y = the distance in y between the point $x(i + j)$ and the line
% X = the distance in x between the point $y(i + j)$ and the line
% I = the integer approximation to the x coordinate of the true minimum
% J = the integer approximation to the y coordinate of the true minimum

```
a = DATA(1); b = DATA(2); c = DATA(3); d = DATA(4); e = DATA(5);
DATA
```

```
% Step 0
% Calculate the slope and the y-intercept of the line Fx = Fy

m = (2*a-c)/(2*b-c); by = (e-d)/(2*b-c);

% Step 1

i = 1; x(1) = 0;y(1) = 0;
f(i) = a*x(i)^2 + b*y(i)^2 + c*x(i)*y(i) - d*x(i) - e*y(i);

% Check to see which region (x(1), y(1)) is in

Fx(i) = 2*a*x(i) + c*y(i) - d;
Fy(i) = 2*b*y(i) + c*x(i) - e;

if Fx(i)<Fy(i),
    smstep3
    break
    return
end

if Fx(i)>Fy(i),
    smstep2
    break
    return
end

if Fx(i)==Fy(i),
    smstep4
    break
    return
end

end
```

File Name: Smstep2.m

```
% Find the intesection point of x(i) and L

n = (m*x(i) + by) - y(i);
```

```
% Find the greatest integer of n

for k=0:n,
    if (n-k)<1,
        if (n-k)>=0,
            gin = k;
        end
    end
end

nstar = gin + 1;

% Compare f(i+j-1) to f(i+j)

j = 0;
cond = 1;

while cond==1,
    j = j + 1;
    x(i+j) = x(i);
    y(i+j) = y(i) + j;
    p = x(i+j) + 1;
    f(i+j) = a*x(i+j)^2 + b*y(i+j)^2 + c*x(i+j)*y(i+j) - d*x(i+j) - e*y(i+j);

    r = 0:1:p;
    q = m*r + by;
    plot(r,q,x,y)
    pause

    Fx(i+j) = 2*a*x(i+j) + c*y(i+j) - d;
    Fy(i+j) = 2*b*y(i+j) + c*x(i+j) - e;
    if Fx==Fy,
        smstep4
    end

    if f(i+j-1)>f(i+j),
        if j<=(nstar-1),
            cond = 1;
        else
            cond = 0;
        end
    else
        cond = 0;
    end
end

if f(i+j-1)<=f(i+j),
    i = i + j;
    smstep5
    break
    return
end
```

% Handle the case of $(x(i), y(i))$ being on the line $F_x = F_y$

```
Fx(i) = 2*a*x(i) + c*y(i) - d;
Fy(i) = 2*b*y(i) + c*x(i) - e;
if Fx(i)==Fy(i),
    smstep4
    break
    return
end

if j==nstar,
    H = (y(i+j) - (m*x(i+j) + by));
    W = m*(x(i+j) + 1) + by - y(i+j-1);
    if H<=W,
        i = i + j;
        smstep3
    end
    if H>W,
        x(i+j) = x(i+j) + 1;
        y(i+j) = y(i+j-1);
        i = i + j;
        smstep2
    end
end
end
```

File Name: Smstep3.m

```
% Find the intesection point of y(i) and L

D=((y(i) - by)/m) - x(i);

% Find the greatest integer of D

for k=0:D,
    if (D-k)<1,
        if (D-k)>=0,
            gid = k;
        end
    end
end

dstar = gid + 1;

% Compare f(i+j-1) to f(i+j)

j = 0;
cond = 1;

while cond==1,
   j = j + 1;
   x(i+j) = x(i) + j;
   y(i+j) = y(i);

   p = x(i+j) + 1;

   f(i+j) = a*x(i+j)^2 + b*y(i+j)^2 + c*x(i+j)*y(i+j) - d*x(i+j) - e*y(i+j);

   r = 0:1:p;
   q = m*r + by;
   plot(r,q,x,y)
   pause

   Fx(i+j) = 2*a*x(i+j) + c*y(i+j) - d;
   Fy(i+j) = 2*b*y(i+j) + c*x(i+j) - e;
   if Fx==Fy,
       smstep4
   end
```

```
    if f(i+j-1)>f(i+j),
        if j<=(dstar-1),
            cond = 1;
        else
            cond = 0;
        end
    else
        cond = 0;
    end
end

if f(i+j-1)<=f(i+j),
    i = i + j;
    smstep5
    break
    return
end

% Handle the case of (x(i), y(i)) being on the line Fx = Fy

Fx(i) = 2*a*x(i) + c*y(i) - d;
Fy(i) = 2*b*y(i) + c*x(i) - e;
if Fx(i)==Fy(i),
smstep4
break
return
end

if j==dstar,
    Y = x(i+j) - ((y(i+j) - by)/m);
    X = ((y(i+j) + 1 - by)/m) - x(i+j-1);
    if Y<=X,
        i = i + j;
        smstep2
    end
    if Y>X,
        x(i+j) = x(i+j-1);
        y(i+j) = y(i+j) + 1;
        i = i + j;
        smstep3
    end
```

```
end
end
```

File Name: Smstep4.m

```
x(i+1) = x(i) + 1;
y(i+1) = y(i);

if f(i)<=f(i+1),
    i = i + 1;
    smstep5
    break
    return
end

if f(i)>f(i+1),
    i = i + 1;
    p = x(i+1) + 1;
    r = 0:1:p;
    q = m*r + by;
    plot(r,q,x,y)
    pause
    smstep2
end
end
```

File Name: Smstep5.m

```
% Determine the approximation to the minimum

p = x(i) + 1;
r = 0:1:p;
q = m*r + by;

if i==1,
    I = x(i);
    J = y(i);
else,
    I = x(i-1);
    J = y(i-1);
end

plot(r,q,x,y,I,J,'*')
pause

I
J
break
return
end
```

File Name: Vsstart.m

```
% Input:  a vector of the form DATA=[a b c d e]
          an initial step size of the form h = step
          i=1
          $x(i) = 0;y(i) = 0$
          a tolerance of the form tol = tolerance
% a = the coefficient of the $x^2$ term in the cost function
% b = the coefficient of the $y^2$ term in the cost function
% c = the coefficient of the $xy$ term in the cost function
% d = the coefficient of the $x$ term in the cost function
% e = the coefficient of the $y$ term in the cost function
% h = the variable step size
% m = the slope of the line $F_x = F_y$
% by = the $y$-intercept of the line $F_x = F_y$
% f = the cost function
% x = the $x$ component of the approximation
% y = the $y$ component of the approximation
% Fx = the partial of f with respect to $x$
% Fy = the partial of f with respect to $y$
% n = the intersection point of $x(i)$ and $F_x = F_y$
% s = the number of step size intervals between $x(i)$ and the line
% gis = the greatest integer of s
% sstar = the maximum number of steps that may be taken in the $y$ direction per iteration
% H = the distance in $x$ between the point $y(i + j)$ and the line
% W = the distance in $y$ between the point $x(i + j)$ and the line
% D = the intersection point of $y(i)$ and $F_x = F_y$
% t = the number of step size intervals between $y(i)$ and the line
% git = the greatest integer of t
% tstar = the maximum number of steps that may be taken in the $x$ direction per iteration
% Y = the distance in $y$ between the point $x(i + j)$ and the line
% X = the distance in $x$ between the point $y(i + j)$ and the line
% diff = the difference in functional values between the approximation and the previous iteration
% tol = a tolerance for the difference between two function values
% I = the integer approximation to the $x$ coordinate of the true minimum
% J = the integer approximation to the $y$ coordinate of the true minimum
a = DATA(1); b = DATA(2); c = DATA(3); d = DATA(4); e = DATA(5);
DATA

% Step 0
% Calculate the slope and the $y$-intercept of the line $F_x = F_y$
```

```
m = (2*a-c)/(2*b-c); by = (e-d)/(2*b-c);

% Step 1
% Designate the step size

h
f(i) = a*x(i)^2 + b*y(i)^2 + c*x(i)*y(i) - d*x(i) - e*y(i);

% Check to see which region (x(1), y(1)) is in

Fx(i) = 2*a*x(i) + c*y(i) - d;
Fy(i) = 2*b*y(i) + c*x(i) - e;

if Fx(i)<Fy(i),
    vsstep3
    break
    return
end

if Fx(i)>Fy(i),
    vsstep2
    break
    return
end

if Fx(i)==Fy(i),
    vsstep4
    break
    return
end
end
```

File Name: Vsstep2.m

```
% Find the intesection point of x(i) and L

n = (m*x(i) + by) - y(i);
s = n/h;

% Find the greatest integer of n
```

```
for k=0:s,
    if (s-k)<1,
        if (s-k)>=0,
            gis = k;
        end
    end
end

sstar = gis + 1;

% Compare f(i+j-1) to f(i+j)

j = 0;
cond = 1;

while cond==1,
    j = j + 1;
    x(i+j) = x(i);
    y(i+j) = y(i) + j*h;

    p = x(i+j) + 1;

    f(i+j) = a*x(i+j)^2 + b*y(i+j)^2 + c*x(i+j)*y(i+j) - d*x(i+j) - e*y(i+j);

    r = 0:1:p;
    q = m*r + by;
    plot(r,q,x,y)
    pause

    Fx(i+j) = 2*a*x(i+j) + c*y(i+j) - d;
    Fy(i+j) = 2*b*y(i+j) + c*x(i+j) - e;
    if Fx==Fy,
        vsstep4
    end

    if f(i+j-1)>f(i+j),
        if j<=(sstar-1),
            cond = 1;
        else
            cond = 0;
        end
    else
        cond = 0;
    end
end

if f(i+j-1)<=f(i+j),
    i = i + j;
    vsstep5
    break
    return
end
```

% Handle the case of $(x(i), y(i))$ being on the line $F_x = F_y$

```
Fx(i) = 2*a*x(i) + c*y(i) - d;
Fy(i) = 2*b*y(i) + c*x(i) - e;
if Fx(i)==Fy(i),
    vsstep4
    break
    return
end

if j==sstar,
    H = (y(i+j) - (m*x(i+j) + by));
    W = m*(x(i+j) + h) + by - y(i+j-1);
    if H<=W,
        i = i + j;
        vsstep3
    end
    if H>W,
        x(i+j) = x(i+j) + h;
        y(i+j) = y(i+j-1);
        i = i + j;
        vsstep2
    end
end
end
```

File Name: Vsstep3.m

% Find the intesection point of $y(i)$ and L

```
D = ((y(i) - by)/m) - x(i);
t = D/h;

% Find the greatest integer of D

for k=0:t,
    if (t-k)<1,
        if (t-k)>=0,
            git = k;
        end
    end
end

tstar = git + 1;

% Compare f(i+j-1) to f(i+j)

j = 0;
cond = 1;

while cond==1,
    j = j + 1;
    x(i+j) = x(i) + j*h;
    y(i+j) = y(i);

    p = x(i+j) + 1;

    f(i+j) = a*x(i+j)^2 + b*y(i+j)^2 + c*x(i+j)*y(i+j) - d*x(i+j) - e*y(i+j);

    r = 0:1:p;
    q = m*r + by;
    plot(r,q,x,y)
    pause

    Fx(i+j) = 2*a*x(i+j) + c*y(i+j) - d;
    Fy(i+j) = 2*b*y(i+j) + c*x(i+j) - e;
    if Fx==Fy,
        vsstep4
    end
    if f(i+j-1)>f(i+j),
        if j<=(tstar-1),
            cond = 1;
        else
            cond = 0;
        end
    else
        cond = 0;
    end
end

if f(i+j-1)<=f(i+j),
    i = i + j;
    vsstep5
    break
    return
end

% Handle the case of (x(i), y(i)) being on the line F_x = F_y

Fx(i) = 2*a*x(i) + c*y(i) - d;
Fy(i) = 2*b*y(i) + c*x(i) - e;
if Fx(i)==Fy(i),
    vsstep4
    break
    return
end

if j==tstar,
    Y = x(i+j) - ((y(i+j) - by)/m);
    X = ((y(i+j) + h - by)/m) - x(i+j-1);
    if Y<=X,
        i = i + j;
        vsstep2
    end
    if Y>X,
        x(i+j) = x(i+j-1);
        y(i+j) = y(i+j) + h;
        i = i + j;
        vsstep3
    end
```

```
end
end
```

File Name: Vsstep4.m

```
x(i+1) = x(i) + h;
y(i+1) = y(i);

if f(i)<=f(i+1),
    i = i + 1;
    vsstep5
    break
    return
end

if f(i)>f(i+1),
    i = i + 1;
    p = x(i+1) + 1;
    r = 0:1:p;
    q = m*r + by;
    plot(r,q,x,y)
    pause

    vsstep2
end
end
```

File Name: Vsstep5.m

```
% Determine the approximation to the minimum

% If the difference between the functional values of the approximation
% and the previous iteration is too large, decrease the step size by a
% half and begin the process again from the previous iteration.

diff = f(i-2) - f(i-1);
if diff>=tol,
    ii = i - 2;
    i = ii;
    for jj=1:i,
        tempx(jj) = x(jj); tempy(jj) = y(jj);
    end
    clear x;
    clear y;
    for jj=1:i,
        x(jj) = tempx(jj); y(jj) = tempy(jj);
    end
    temph = h;
    h = temph/2;
    vsstart
end

p = x(i) + 1;
r = 0:1:p;
q = m*r + by;

if i==1,
    I = x(i);
    J = y(i);
else,
    I = x(i-1);
    J = y(i-1);
end

plot(r,q,x,y,I,J,'*')
pause

I
J
break
return
end
```

VITA

Tina R. Burrell was born on November 5, 1969 in Fredericksburg, Virginia. She graduated from King George High School in 1987. Tina received her Bachelor's degree in mathematics from Virginia Tech in 1991. She continued to study math and earned her Master's degree from Virginia Tech in 1993.

Tina Burrell