RECOVERABLE RESERVE ESTIMATION IN MULTIPLE SEAM MINES

by

Moncef Sellami

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

Mining and Minerals Engineering

APPROVED:

Christopher Haycocks, Chairman

Ertugrul Topuz

Michael E. Karmis, Department Head

June, 1993

Blacksburg, Virginia

RECOVERABLE RESERVE ESTIMATION IN MULTIPLE SEAM MINES

by

Moncef Sellami

Committee Chairman: Christopher Haycocks
Mining and Minerals Engineering

(ABSTRACT)

A reliable estimate of minable coal reserves is essential for long-term planning and for conservation of energy resources. In the majority of the world's coalfields, seams are found on multiple horizons, and coal recovery may be significantly reduced depending upon the magnitude of seam interaction.

In order to evaluate the effects of interaction in terms of coal losses, a hazard technique has been developed. The first stage in this technique was to develop an efficient and accurate system to grid spatial geologic and structural information developed from randomly placed boreholes. The second stage was to utilize these gridded values to determine total available coal tonnage and chemical compositions of the seam.

Depending on whether over or under mining effects are to be assessed, spatial geologic and structural conditions between the seams, are evaluated to determine if interaction is possible. Where interaction will occur, recoverable reserves are reduced accordingly. To demonstrate application of the hazard approach, potential interaction in a lower seam has been evaluated.

To facilitate implementation of this reserve evaluation method under multi-seam conditions, a user-friendly computer design program,

HAZARD, has been developed to calculate the recoverable coal reserves in an affected seam.

ACKNOWLEDGEMENTS

The author is indebted to Dr. Christopher Haycocks, his major advisor for his guidance, encouragement and support. Sincere thanks are also due to Dr. M. Karmis and Dr. E. Topuz, for their advice and many constructive suggestions and comments. A special note of thanks is given to LeeAnn Ellis for the final editing of this thesis.

Finally, the author is forever indebted to his mother, father, and each member of the family whose encouragement and support have proven limitless.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
Introduction

1.1 Statement of Problem

Today, many coal operators are mining in a multi-seam environment, simply because most of the world coal fields contain multiple seams. This situation is especially true in Appalachia, where up to 20 coal seams may exist in a given stratigraphic section as shown in Figure 1.1.

The extraction of individual coal seams below or above a previously mined seam may involve significant strata control problems due to interaction. As a result, depending upon the magnitude of seam interaction, coal recovery in a multi-seam environment can be significantly reduced. According to a study by Engineers International (1981), up to 156 of the 229 billion tons of bituminous coal in 28 states in the U.S. may be subject to underground multiple-seam mining. Seam ownership frequently changes vertically between seams and there is intense concern about the damage that both over- and under- mining is doing to unmined seams. As more and more operations are forced to mine in multi-seam environments because of continually increasing demands for coal and depletion of reserves, determining reserve losses due to interaction is becoming an urgent problem for the industry.

To meet the planning needs of a modern mining operation, coal resource characterization requires more than determination of the minable coal tonnage. Environmental considerations impose stringent restrictions on the emission of particulates, sulphur dioxide, and nitrogen oxides,

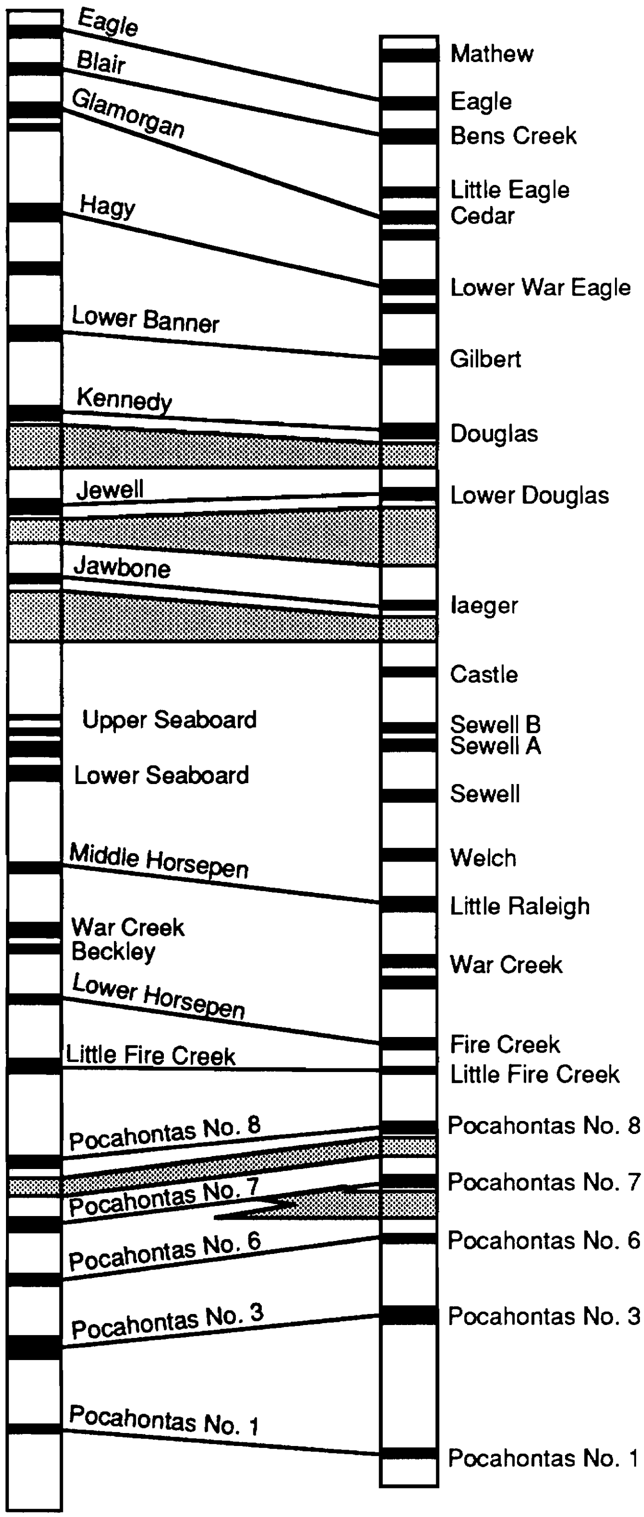

Figure 1.1 Stratigraphy Showing Multiple Coal Seams (After Zhou, 1989)

which in turn necessitate the development of optimum production schedules and blending strategies that will yield production of specified quantity and quality of coal from beds. The examination of the chemical and structural characteristics of the seam, which are factors critical in the determination of its functional and potential use, is also required.

1.2 Objectives and Methodology

The primary objective of this research is to develop a method to evaluate potentially recoverable coal reserves in multi-seam mining situations. The method will also specify the chemical characteristics of the seam in a proposed mining area, such as BTU content, average sulphur and ash percentage.

In order to evaluate the interaction effects in terms of amount of coal loss due to negative interactions, a hazard technique has been utilized. The first stage of this approach is to develop an efficient and accurate gridding system that grids irregularly spaced borehole data. The gridded data will be suitable for use in determining how much coal can be recovered from any specified seam.

Recoverable reserves are based on existing techniques for estimating damage due to interaction. Programs developed specifically for multi-seam mining, including MULSIM, RUMSIM, and MSEAM, are used to predict damage for specific mine conditions. Damage predicted can then be used with site characterization to estimate recoverable coal.

To facilitate the implementation of the evaluation method, a user friendly program called HAZARD has been developed to predict the recoverable tonnage in either underlying or overlying seams due to mining in an existing seam.

Chapter 2
Literature Review

2.1 Methods of Reserve Calculation

Various methods have been applied over the years for estimation of both grades and tonnages. The conventional methods can be classified into three categories (Crawford et al., 1979): 1) Geometrical Methods, 2) Distance Weighting Methods, and 3) Geostatistical Techniques. The last two categories are based on extrapolation technique between irregularly spaced sample points.

2.1.1 Geometrical Methods

The simplest method of computing the variances of any parameter across an irregularly sampled area is the method of triangles or triangular prisms. A plan of the mineral deposit showing the entire area of the mineral body can be divided graphically into a system of triangles by connecting holes with straight lines (Figure 2.1); no line should be crossed by another. Each triangle represents the base area of an imaginary prism of some thickness.

The area of the triangle and the value of the thickness assigned to that area can be determined by averaging the thickness at three corners. The rule of gradual changes of all variables from one data point to another is the main principle of this method. It implies that all sampled elements of a mineral body change gradually and continuously as a linear function along a straight line connecting two adjacent sample points.

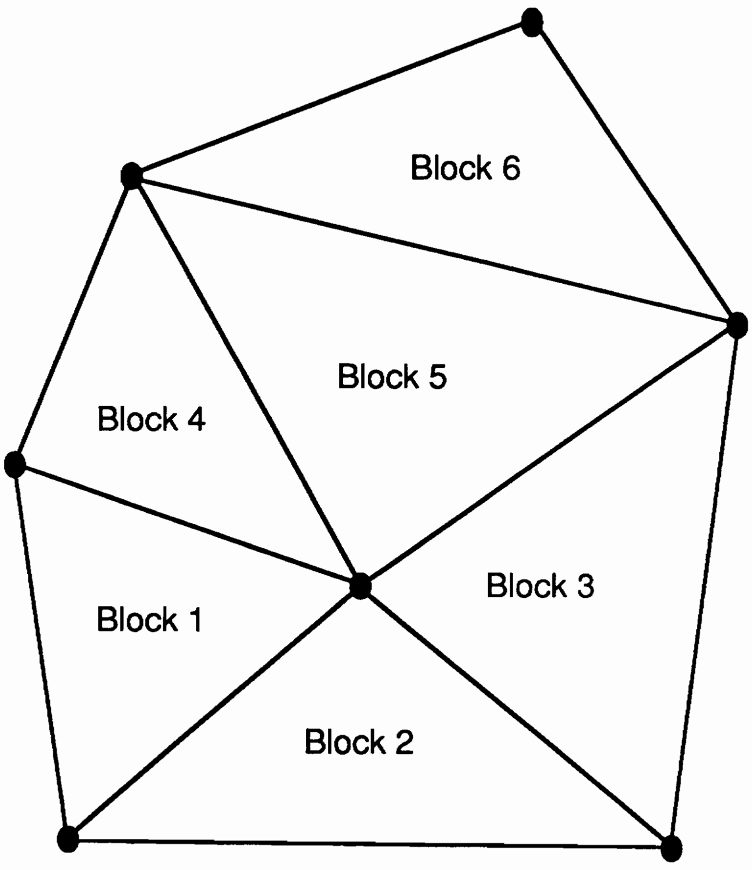

Figure 2.1 Triangulation Technique Of Calculating Reserves From Sampling Points

The method gives conservative results because of the averaging the data measurements at three corners.

Based on the rule of gradual changes of interpretation of exploration data, the triangular method is most applicable to sedimentary and disseminated ore deposits explored by regularly spaced drill holes (Soderberg, 1930).

Another much used method for calculating reserves based upon the gradual changes principle is the standard cross-section method. The initial step in the application of the cross-section method is to divide the mineral body into blocks by constructing geologic sections at intervals along the traverse lines. Each internal block is confined by two sections and by an irregular lateral surface, and each end block by a single section and by an uneven lateral surface (Figure 2.2). To calculate mineral reserves using the cross-section method, first determine the mineralized area in all sections by using either planimeter irregular outlines or other geometric schemes, then calculate average values for each section. Compute the volume for each block using the simplest and most used formula for volume between two partial sections of area A_1 and A_2 and perpendicular distance, L:

$$\text{Vol} = \frac{A_1 + A_2}{2} L \quad \text{and} \quad \text{Ton} = \frac{A_1 + A_2}{2} L\,F ,$$

where $F = t/m^3$; then sum the results of all blocks and compute the total volume and tonnage values for estimated mineral body. To increase the accuracy of computation using the cross-section method, the sections

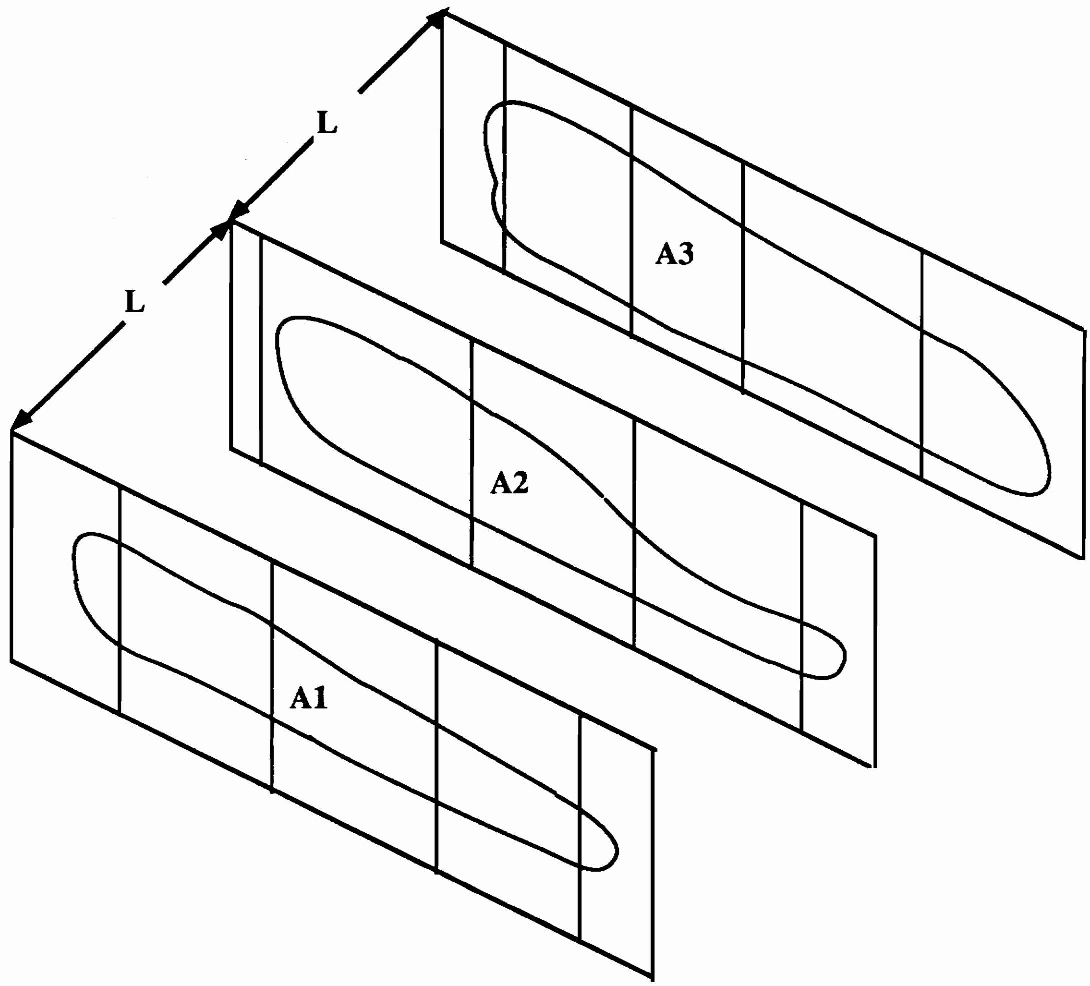

Figure 2.2 Cross Section Technique (After Crawford and Hustrulid, 1979)

should be placed close together, distance between sections is usually governed by the character of the mineral body and the distribution of mineral values. The method is applicable to irregular bodies (Stoddart, 1929; Thomas, 1929; Wolff, 1925), but when computations of several valuable components are required and the mineral body shows grade variations for each component, it may be difficult or impossible to apply cross sectional methods.

The method of triangles can be modified by drawing perpendicular lines that bisect the triangle sides, forming a polygon as shown in Figure 2.3. This method, called the polygonal method, was outlined by Harding (1923), and is claimed by Popoff (1966) to be more accurate than the simple triangulation method for coal reserves. It is based upon the rule of nearest points or equal sphere of influence which implies that the value of any point between two samples is constant and equal to the value of the nearest sample; the rule assumes that the value of a sample extends halfway to any adjacent sample.

In this case, a value, such as thickness, measured in a drillhole in the center of the polygon would extend over the entire area of the polygon. Multiplying the area by that thickness and applying the proper tonnage factor (m^3/t), one can calculate the tonnage of each polygon; the total tonnage for the deposit is the sum of all tonnages of all the polygons.

Another geometric method for computing volume and tonnage of mineral reserves is called isopach (or isothickness). This method is based on the assumption that unit values, from one point to another, undergo continuous and uninterrupted changes according to the rule of gradual

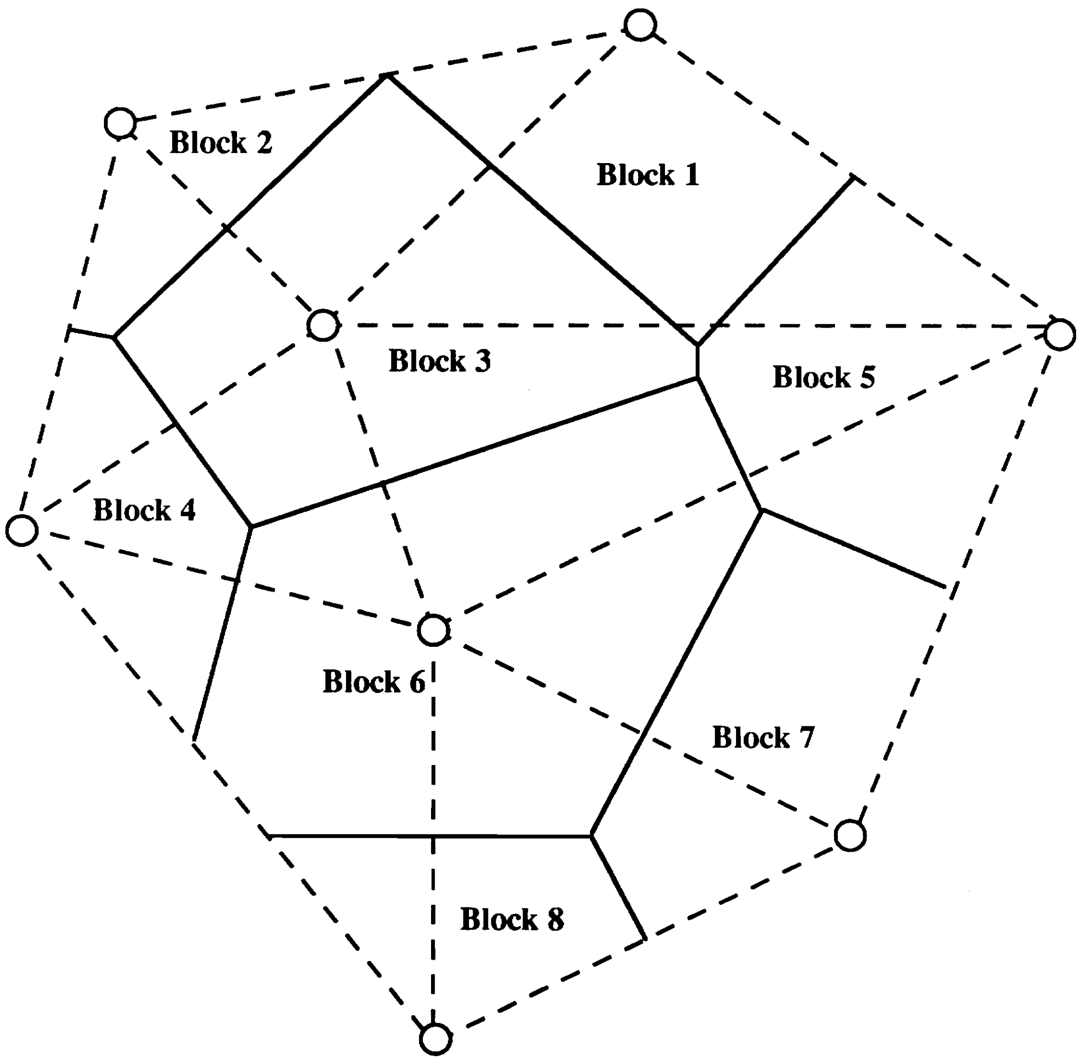

Figure 2.3 Polygonal Technique Of Calculating Reserves From Sampling Points

changes. The theory of the method, developed by Sobolevsky, is to construct isothickness lines joining all points of equal thickness (Figure 2.4); the volume and tonnage of each block formed by isopachs may be computed by standard cross-section formulas. This method is appropriate for mineral bodies where certain natural regularities occur in the variations in thickness, grade, and volume (Hughes, 1959; Shklyarskiy, 1921).

When considering these methods for use in coal reserve computations, they can only be used with confidence inside the boundary formed by the outer range of the sampling points.

2.1.2 Distance Weighting Method

Based upon the rule of gradual changes for making value estimates, the objective of the Distance Weighting Method is to assign a value to a point or a block from neighboring points. Since the potential influence of value at a point decreases as distance from that point increases, the value change is a function of inverse distance and is determined using the relationship:

$$G = \sum_{i=1}^{n} W_i G_i$$

$$W_i = \frac{\frac{1}{d_i^m}}{\sum_{i=1}^{n} \frac{1}{d_i^m}}$$

where W_i is the weight assigned to point i and G_i is its known grade. An example of this method for exponent m is equal to two is shown in

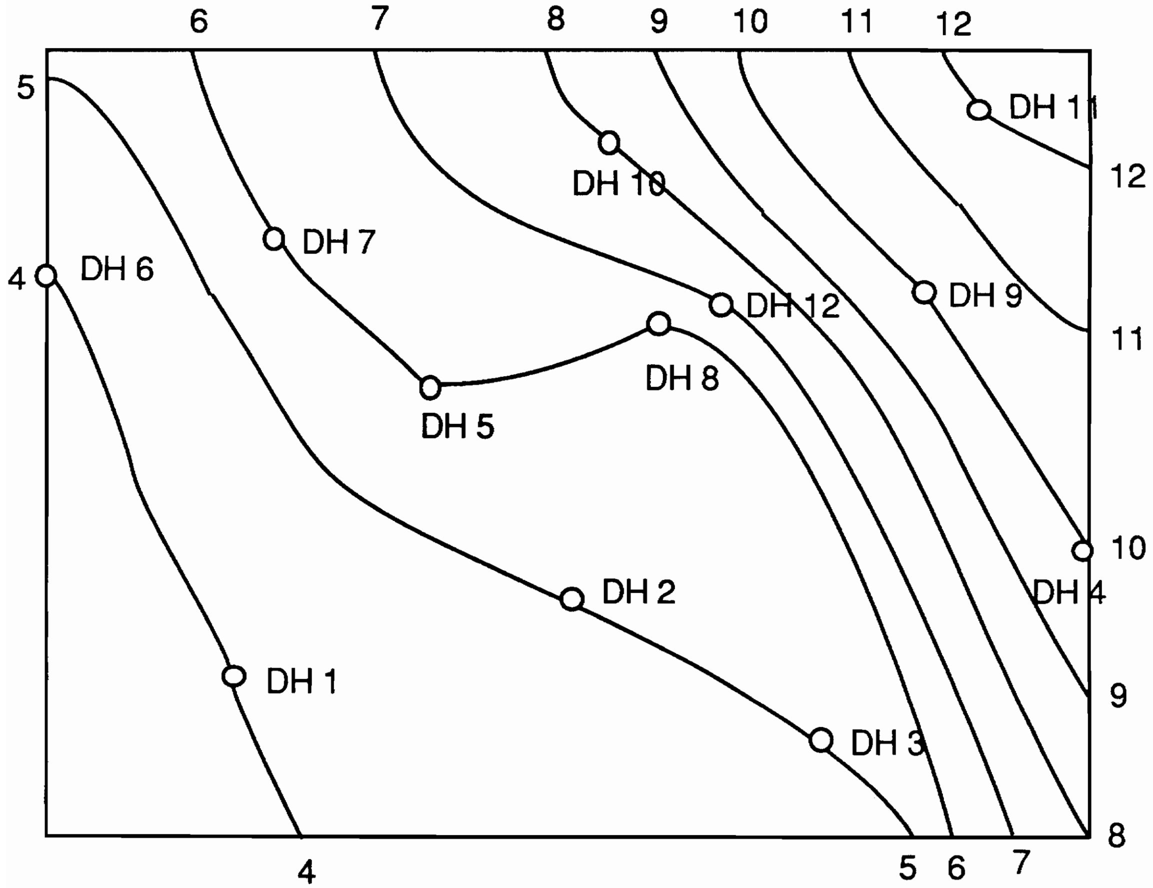

Figure 2.4 Isopach Technique

Figure 2.5 (O'Brien and Weiss, 1968). Point B is interpolated for the block assay by using five closest points as follows:

$$B = \frac{G_2 \text{ x } \frac{1}{(d_2)^2} + G_6 \text{ x } \frac{1}{(d_6)^2} + G_9 \text{ x } \frac{1}{(d_9)^2} + G_7 \text{ x } \frac{1}{(d_7)^2} + G_4 \text{ x } \frac{1}{(d_4)^2}}{\frac{1}{(d_2)^2} + \frac{1}{(d_6)^2} + \frac{1}{(d_9)^2} + \frac{1}{(d_7)^2} + \frac{1}{(d_4)^2}}$$

$$B = \frac{0.5 \text{ x } \frac{1}{(200)^2} + 0.5 \text{ x } \frac{1}{(200)^2} + 0.7 \text{ x } \frac{1}{(150)^2} + 1.0 \text{ x } \frac{1}{(250)^2} + 0.9 \text{ x } \frac{1}{(100)^2}}{\frac{1}{(200)^2} + \frac{1}{(200)^2} + \frac{1}{(150)^2} + \frac{1}{(250)^2} + \frac{1}{(100)^2}} = 0.77\ \%$$

where G is assay element value and d is distance.

The Distance Weighting Method has some limitation on predicting above or below the maximum or the minimum sample or borehole values (Fraher et al., 1991).

2.1.2 Geostatistical Methods

Among a spectrum of methods that have been used for estimating mineral inventories, the geostatistical approach, defined as the application of the theory of regionalized variables, is the most powerful technique and the most widely used. The theory was advanced by George Matheron in 1963 and has been recognized throughout the world as a superior method for estimating the grade of in-situ mineralization because it statistically quantifies the error of estimation. This is achieved by conceptualizing the spatial distribution of each measurable quantity of interest, such as seam thickness, as a random function. Formulation through random functions allows consideration of measurable quantity as a random variable at every

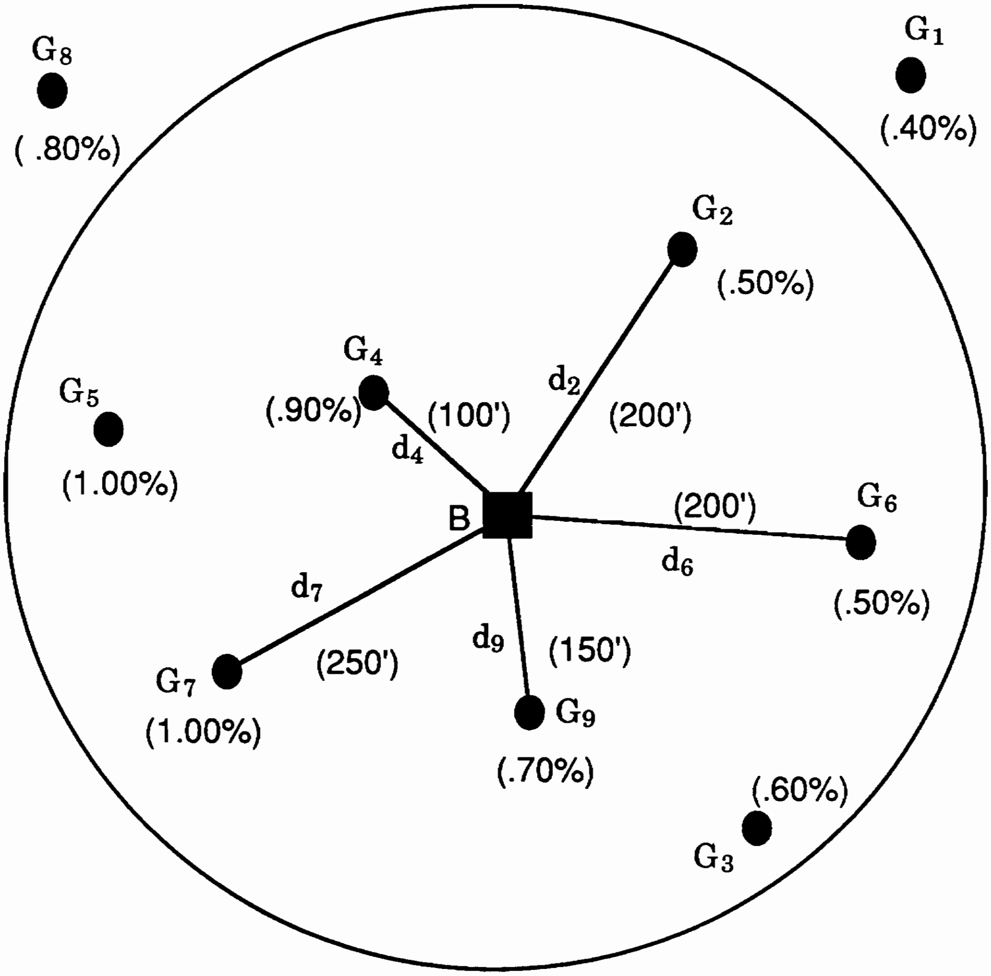

Figure 2.5 Hypothetical Block Calculation From Neighboring Samples Using An Inverse Squared Distance Method (After O'Brien and Weiss, 1968)

location, structured according to the correlation scheme dictated by the geological phenomena. The correlation structure may change from one deposit to another but is always dependent on the distance and direction, previously parameterized as zone of influence and anisotropy.

The fundamental tool of geostatistical analysis that permits the geological parameters quantification is called the semi-variogram or simply the variogram (Matheron, 1971, David, 1977). A variogram is a plot showing how grade in a deposit varies versus distance. Taking the difference between two paired values yields different signs. Therefore, it is necessary to apply a technique of classical statistics and sum the difference squared, and divide by twice the number of pairs found. The result is called variance and defined as γ (gamma) and is represented by the following working relationship:

$$2\gamma = \frac{1}{n} \Sigma[Z(x) - Z(x + h)]^2$$

where n is the sample pairs, Z(x) is the sample grade at point x, Z(x + h) is the sample grade at a point h distance from Z(x), and h is the distance between samples.

In order for the experimental variogram to be useful as a tool for estimating the sample values within a deposit, several mathematical models have been proposed over the years to theoretically fit all realizations of the spatial behavior (Clark, 1982). The simplest of these is the linear model $\gamma(h) = A(h)$ (Figure 2.6) and the De wijsian model $\gamma(h) = 3 \log(h)$ (Figure 2.7). Both will produce a straight line when the log(h) is plotted to log scale.

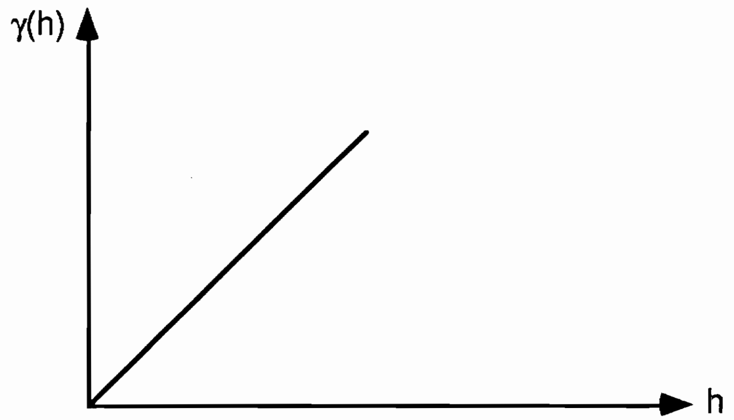

Figure 2.6 Linear Model

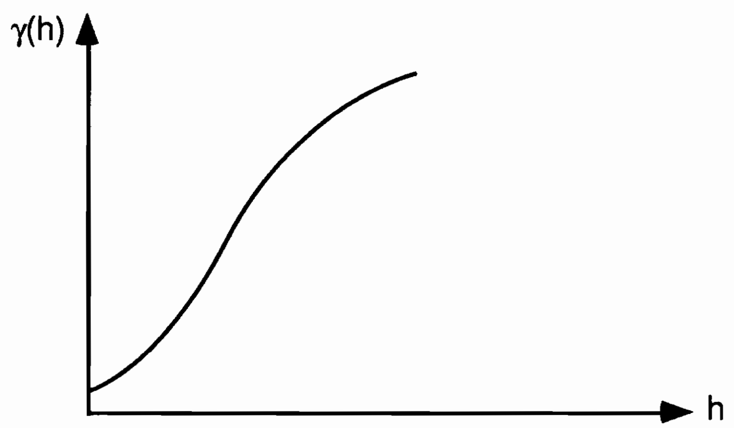

Figure 2.7 De Wijsian Model

The model found to be widely applicable in practice and an almost universal model is the spherical or Matheron model. It is defined by the formula :

$$\gamma(h) = C\left(\frac{3h}{2a} - \frac{1h^3}{2a^3}\right) + C_o \qquad \text{when } h \leq a$$

$$= C + C_o \qquad \text{when } h > a$$

where $C + C_o = \gamma(\infty)$ is called the sill: one in which the variogram reaches a finite value as h increases indefinitely, C_o is the nugget effect, and a is the range or maximum zone of influence (Figure 2.8).

Kriging Formulation:

Selecting the weights for an estimator, constructed as the linear combination of nearby samples so that the estimation variance is minimized, is called Kriging (David, 1977). Kriging is, in fact, a word which has been coined to cover both the best linear unbiased estimator (B.L.U.E.) of a point and the best linear weighted moving average of a block.

Let $(x_1, ..., x_k, ..., x_n)$ represent the set of drill holes locations and $Z(x_k)$, the realization of the random function Z at x_k. The Kriging estimator is constructed as the linear combination of $Z(x_k)$, from the n available drill holes, as follows:

$$Z^*(x_o) = \sum_{k=1}^{n} a_k Z(x_k)$$

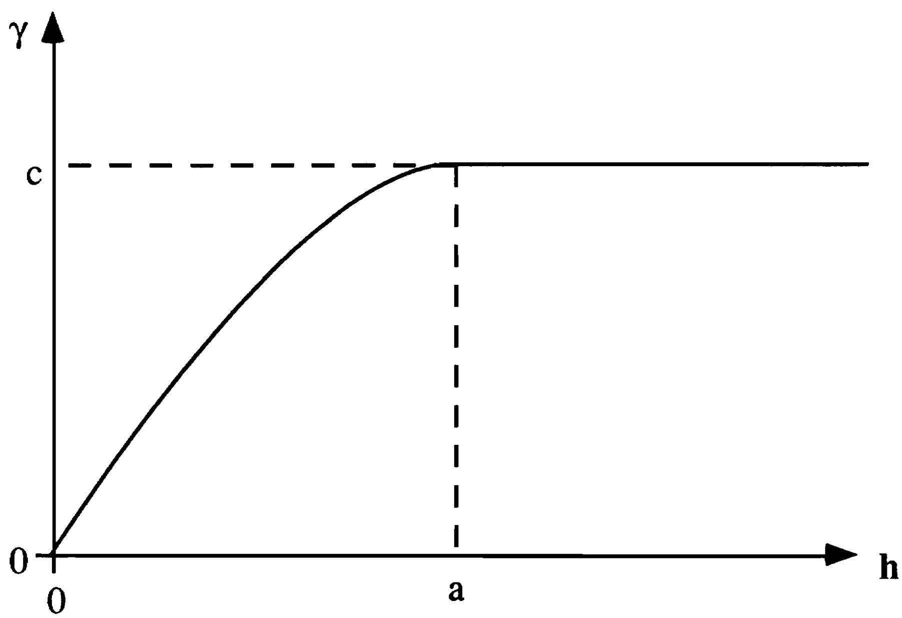

Figure 2.8 Spherical Model

Then, to meet the unbiased condition that the computed value should be, on the average, equal to the real value and not systematically higher or lower, since

$$E\left(Z^*(x_o)\right) = m$$

then,

$$E\left(\sum_{k=1}^{n} a_k Z(x_k)\right) = m$$

also,

$$E\left(Z(x_k)\right) = m$$

therefore,

$$\sum_{k=1}^{n} a_k = 1$$

and the variance of the estimate error is expressed as:

$$\sigma_e^2 = \mathrm{Var}\left\{Z(x_o) - Z^*(x_o)\right\} = \sigma_v^2 - 2\sum_{k=1}^{n} a_k \sigma_{vx_k} + \sum_{j=1}^{n}\sum_{k=1}^{n} a_j a_k \sigma_{x_j x_k}$$

where σ_v^2 is the variance of the grade of a block of volume v, σ_{vx_k} is COV[Z(v), Z(x)], the covariance of the grade of block v and the grade of sample x_k, $\sigma_{x_j x_k}$ is, COV[Z(x_j), Z(x_k)], the covariance of the grades of sample x_j and sample x_k.

Each of the variance and covariance values can be determined from the variogram function. The Kriging solution is thus, to minimize the estimation variance by finding the set of weights a_k (k=1, ..., n) , subject to the unbiased condition.

The minimization problem can be solved by the Lagrange method for constrained optimization, which results in the following system of n + 1 linear equations with n + 1 unknowns, the a_k's and μ . It can be written in the usual form:

$$\sum_{j=1}^{n} a_k \, \sigma_{x_j x_k} + \mu = \sigma_{v x_k} \qquad (k=1, ..., n)$$

$$\sum_{k=1}^{n} a_k = 1$$

or in the matrix form to efficiently solve such a system of equation:

$$[E] \; A = D$$

where [E] is a symmetrical matrix of the covariance between all sample values, A is a vector of kriging weights, and D is a vector of covariance between the block being estimated and the sample values.

Using the matrix formula: $A = [E]^{-1} D$, we can calculate the unknowns variables a_k's (David, 1977) as all σ's are determined from the variogram.

Even though, these methods are accurate and widely applicable, experience has shown that when operating with coal seams where the boreholes which may be spaced up to a mile or more apart, the geostatistical methods methods are not feasible. These indicate severe limitations dealing with coal and therefore an alternative system is being utilized (Fraher et al., 1991).

2.2 Interaction in Multi-Seam Mining

Mining operation in multi-seam conditions can result in ground control problems due to interaction between seams. Such problems not only increase operating cost, but also reduce recoverable reserves.

The problem of multi-seam mining interaction was first discussed in the 1920's in a study on ground movement and subsidence (Report of Sub-Committee, 1926). In the last decade, a good progress has been made by researchers as well as mine operators in defining the factors that affect the amplitude and propagation of interactions and identifying many of the basic interaction mechanisms (Haycocks and Karmis, 1983; Grenoble, et al., 1985; Peng, 1986).

Factors which govern interaction, and thus potential coal loss are (Zhou et al., 1989):

- Mining methods used in extracting the previously mined overlying or underlying seam, as well as the seam being currently mined;
- Percentage and uniformity of coal recovery in all seams concerned;
- Thickness of innerburden between the seams under consideration;
- Structural characteristics of the innerburden;
- Thickness of the coal seams mined; and
- Time elapsed following the mining of the previously mined seam.

Previous research has quantified various interaction mechanisms for various mining conditions and geologic environments. The most useful for evaluating reserve losses due to multi-seam are:

- Subsidence;
- Load transfer;
- Arching/Caving; and
- Innerburden shearing.

A successful combination of case studies and numerical modeling has resulted in significant developments in planning and design for multi-seam mining ground control (Haycocks and Karmis, 1982; Grenoble et al., 1985). Field data analysis allowed identification of the significant factors affecting interaction and assessment of their importance. Relationships derived from statistical analysis of case study data also provided methods for evaluating potential interaction damage with a minimal amount of input data (Webster et al., 1984; Chekan et al., 1986).

The use of large amounts of field data has also helped determine the physical limits of modeling for the development of design criteria (Ehgartner, 1982; Wu and Haycocks, 1986). Research has included selection of mining methods. Longwall and room-and-pillar methods tend to produce different effects in the superincumbent strata, depending upon the mining geometries and geologic setting (Holland, 1951; Stemple, 1956). Partial extraction, normally a part of room-and-pillar mining, can produce a variety of stress conditions in overlying seams depending on the size of pillars left and their location. This extraction ratio also affects subsidence amplitude which is an important factor in assessing multi-seam interaction damage and probable coal loss. Innerburden is critical in both methods of mining, but particularly longwall where interaction effects may extend a vertical distance of over 800 ft (Lazer, 1965). Geology of the innerburden is also very important (Szwilski, 1979; Johnson, 1973; Haycocks, 1980). More competent material such as sandstone inhibits arch formation and reduces subsidence (Styler and Dunham, 1980), thus reducing potential coal loss due to interaction.

The complex and far ranging multi-seam research lends itself to solution by computerized methods. The use of computer programs not only provides a rapid transfer of reproducible technology, but also improves the engineering and management control of the mining operations.

A series of computer programs developed specifically for multi-seam mining, including MSEAM, RUMSIM, and MULSIM (Wu et al., 1987; Grenoble et al., 1985; Zhou et al., 1989), have been used to predict the magnitude of interaction damage and its location for specific mine structures.

Chapter 3
Gridding Method

Few mining companies today have not suffered some effects from interaction. This is especially true in Appalachia, where the majority of the coal seams are found in multiple horizons. Also, with increasing demands for low sulphur, nonpolluting coals, and environmental regulation, use of more sophisticated method of evaluating quantity and quality of coal is required.

To provide more accurate information concerning factors that affect the quantity and quality of coal, a realistic method called a hazard technique has been developed. A primary requirement of the hazard method is to develop an efficient and accurate gridding system to grid the irregularly distributed exploration data points.

To estimate a value at each grid point, the approach uses either the Inverse Squared Distance or Inverse Distance technique for data interpolation. These techniques are simple in concept, capable of interpolating many differing types of value parameter. They lend themselves easily to computerization (David, 1977). The second requirement of this technique is to forecast and quantify the probable magnitude due to multi-seam interaction. Predicting damage is essential to determining how much coal can be recovered from any specified seam.

3.1 Grid Algorithm

Several basic methods exist for assigning values to grid nodes and no one can be automatically judged best, because each has its own properties that are of value under certain conditions (Thomas et al., 1986).

A grid is a set of values or numbers that is regularly arranged, commonly in a square or rectangular pattern, although other forms are also used, in this case, gradients and trends would be easier to detect and project than with scattered data, so the ordered data set would be easier to use. The use of a grid involves few steps :

1) delineation of the area and variable to be used;

2) Grid pattern over the area;

3) Calculation of values to be assigned to each grid point; and

4) Use of these values to draw contours and to perform other type of calculations.

Figure 3.1 shows a series of observations on a map; each point is characterized by its x_1 coordinate (east-west), its x_2 coordinate (north-south), and its z coordinate or the variable values to be gridded. The observations may be identified by numbering them sequentially as they are read in, from 1 to n. Therefore, an original data point n has coordinates x_{1n} in the east-west direction, x_{2n} in the north-south direction, and variable value z_n.

Figure 3.2 shows a grid pattern made up of rows of horizontal lines and columns of vertical lines. The rows and columns cross, defining grid intersections or grid nodes. Specifying the grid limits (x_{min} and y_{min}) and the grid spacing defines the grid frame work. The grid contains

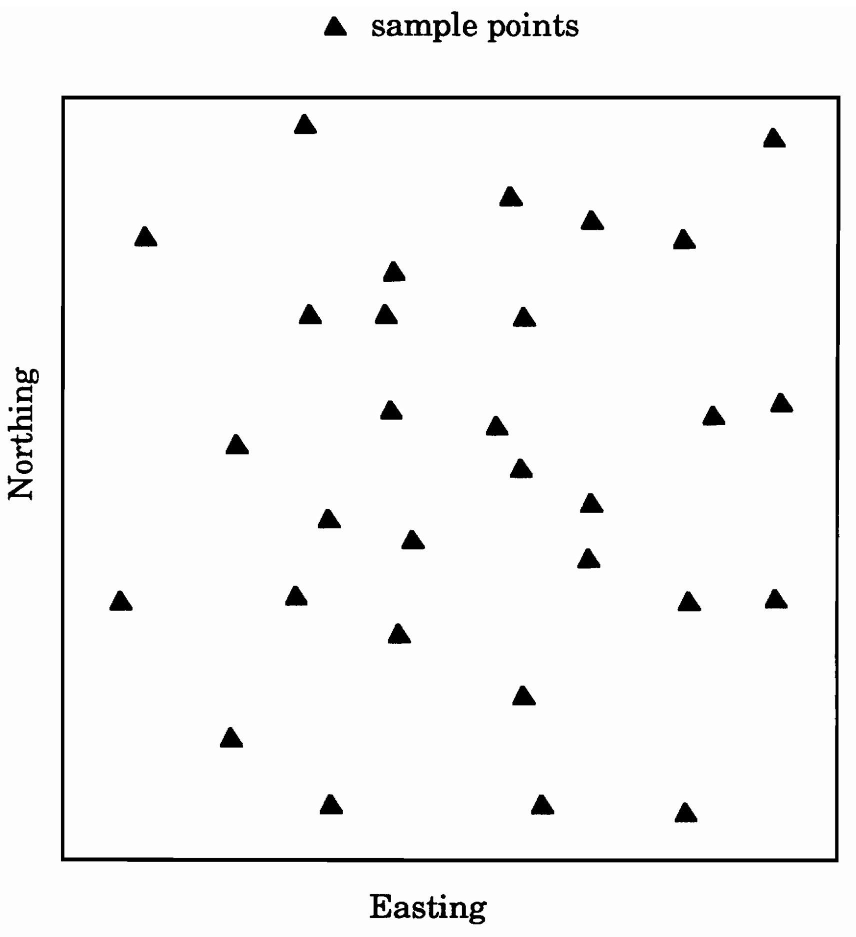

Figure 3.1 Base Map Showing Data In Area To Be Gridded

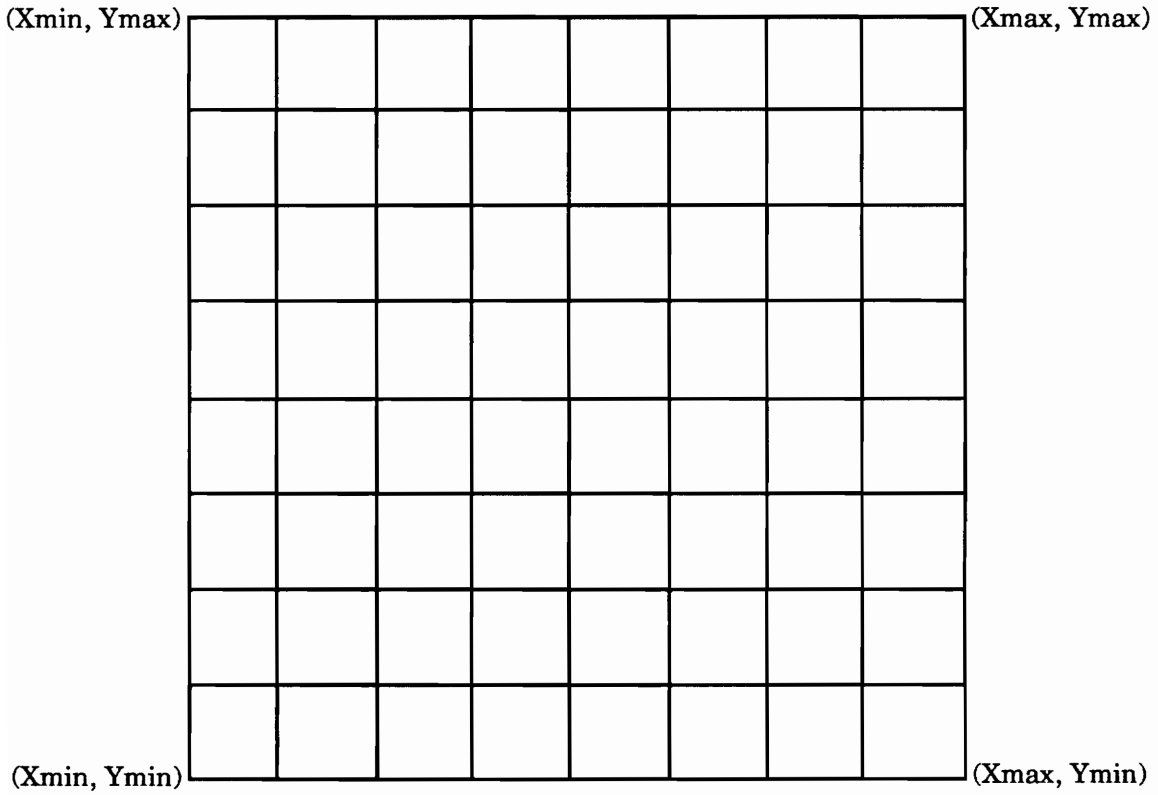

Figure 3.2 Definition Of Grid Framework

intersections that are spaced the same distance apart in both x and y direction; it could also be a series of rectangles if the row spacing is different from the column spacing. Square or rectangular grids are most common in mapping programs, and their advantage over a regular triangular grid is the ease with which they can be manipulated by computer (Davis, 1973).

Because points on a rectangular grid are uniformly distributed over the map area, there is no need to retain their x and y coordinates. The estimation of the value at the grid intersection is all that is needed for the second stage. Therefore, the grid may be stored in the form of a matrix. As a result, the algorithm that assigns values to the grid nodes may be coded as a nested do-loop, which indexes over the row and column subscripts of the matrix containing the gridded surface. When needed, the coordinates of a grid node can easily be calculated according to the equations :

$$x_c = x_{min} + (c - 1)\,dx$$

$$y_r = y_{min} + (r - 1)\,dy$$

where x_{min} and y_{min} are the minimum x and y values in the map area, dx is the distance between columns, dy is the distance between rows, c is the column of the grid node, and r is the row of the grid node.

The third step in making a grid is the assignment of a number to each of the grid nodes or intersections (Figure 3.3). These numbers are calculated from different types of algorithms that will be discussed in the next section. They represent values of the variables under consideration

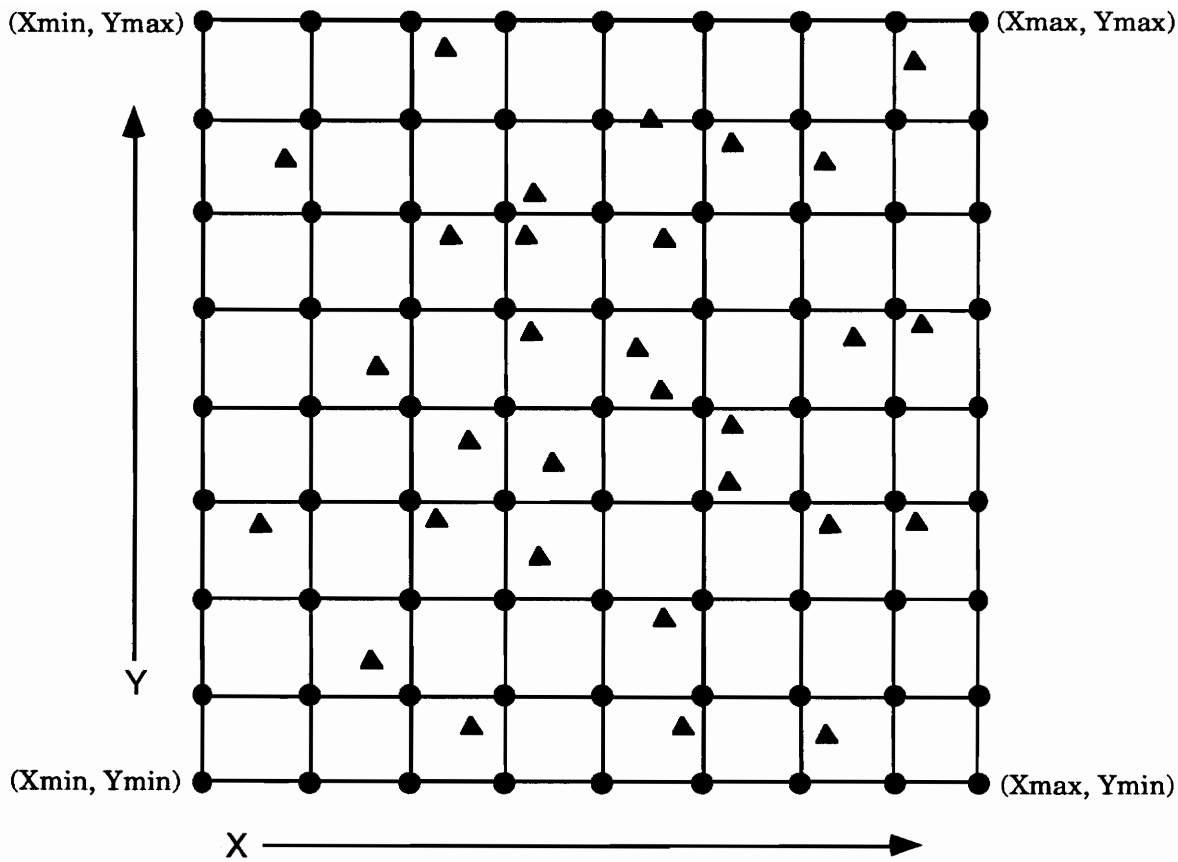

Figure 3.3 Rectangular Grid Superimposed On The x-y Plane. The Value At Each Grid Intersection, Denoted By ●, Is Estimated From The Sample Data Points, Represented By ▲, Which Are Irregularly Distributed Over The x-y Plane

(elevation, thickness, etc.). As far as the computer is concerned, all information is contained in the grid matrix, and the original observation points are unnecessary.

Two steps are involved in assigning values to grid nodes:

1) How to select the set of nearest neighbor sample points; and

2) How to assign weights to the sample points once they are selected.

Three basic methods were used in the reserves calculation HAZARD program to establish the first step.

3.1.1 Nearest Neighbor Search

The nearest neighbor search pattern finds the n nearest sample points regardless of their radial distribution around the element to be estimated. This search procedure calculates the straight line distance from the location of a grid node to the surrounding data points. The closest n points calculated by the search procedure are used to estimate a value z at the grid node.

Selection of an optimum number of closest points to be used in the estimating algorithm can be made only by considering the nature of the data point distribution (Sampson, 1975).

The disadvantage of this procedure is that if data are clustered, it is possible that the points nearest the node all lie together relative to the node. Hence, all closest n points would come from the same side, and other points in the other sides would be ignored (Thomas, 1973).

3.1.2 Quadrant Search

A quadrant search procedure divides the area around a grid node into four equal segments. The nearest n data points specified by the user are found in each quadrant.

A quadrant search introduces more smoothing than a nearest neighbor search. Points in a quadrant will be ignored if the maximum number of points have already been located in that quadrant, even though these points may be closer to the grid node than points in other quadrants. The effect is to increase the radius of search, using more distant points than would be the case with a simple nearest neighbor search.

3.1.3 Octant Search

An octant search divides the area around the grid point into eight equal segments. This procedure searches for nearest n points supplied by the user in each section.

This procedure may introduce more smoothing than will a nearest neighbor search using the same number of points. This is because points will be ignored in an octant in which the maximum number of points have already been located, even if these points are closer to the grid node than points in other octants. The effect is to spread the search over a greater area, using more distant points than would otherwise be the case.

An octant search insures better data distribution around the grid node being estimated.

In selecting a gridding algorithm, the user must be aware of the direct relationship between complexity of the algorithm and amount of computer time and memory required for implementation. It can not be shown that one method is absolutely superior to every other. Instead, the preferable technique for a given problem depends on the distribution, reliability, and volume of data; the memory requirements and operating expenses of the algorithm; and the degree of accuracy required for the application (Sampson, 1975; Thomas, 1986).

3.1.4 Inverse Squared Distance and Inverse Distance

Once the points for calculating a value to a given grid intersection are selected, either the inverse squared distance or the inverse distance is selected by the user to assign a value at each grid point. This corresponds to the second stage of gridding algorithm. This section first describes the notations used in the program to calculate the node value.

A data point, is the location of a known value of the surface on the x, y plane. The expression (x_i, y_i) will refer to the location of the i^{th} element in a random sample used as input to the gridding routine; the known value at (x_i, y_i) is z_i. The total number of sample data points is n. The x and y location of the element in the r^{th} row and c^{th} column of the interpolated grid is represented by (x_c, y_r). This is referred to as a grid node or grid point. The value of the interpolated grid at the grid node (x_c, y_r) is denoted $z_{c,r}$ where, again, the subscripts denote the row and the column of the interpolated grid in which the value occurs. For brevity, the entire grid of interpolated values may be referred to as $\hat{z}$. The number of rows in the grid is represented by ny and nx is the number of columns, thus there are ny x nx elements in the matrix $\hat{z}$.

The principle of inverse squared distance and inverse distance method states the closer a data point is to the grid point to be estimated, the greater its influence in the interpolation process. The weights should decrease with distance as fast as inverse distance-squared to prevent spikes at the data points (Thomas et al., 1986).

These two algorithms use only nearby samples to evaluate each element in $\hat{z}$. To estimate a value at any desired grid location (x_c,y_r), the inverse squared distance first determines the square of the distance while the inverse distance determines the distance from every sample point to (x_c,y_r). Then, sample points are ranked in order of increasing distance from the grid point; only the nearest samples calculated by the program are used in the evaluation procedure. A value is then assigned to $z_{c,r}$ that is equal to the weighted average of the values of the np nearest data points. When the distance between grid node (x_c,y_r) and the nearest sample point (x_i,y_i) is less than some small predetermined value, $z_{c,r}$ is set equal to the value of z_i at the sample point. The equation of inverse squared distance used for calculating the grid value is:

$$z_{c,r} = \frac{\sum_{i=1}^{k} \frac{z_i}{d_i^2}}{\sum_{i=1}^{k} \frac{1}{d_i^2}} \qquad \text{when } d_i > \varepsilon$$

$$z_{c,r} = z_i \qquad \text{when } d_i \leq \varepsilon$$

The equation of inverse distance used for calculating the grid value is:

$$z_{c,r} = \frac{\sum_{i=1}^{k} \frac{z_i}{d_i}}{\sum_{i=1}^{k} \frac{1}{d_i}} \qquad \text{when } d_i > \varepsilon$$

$$z_{c,r} = z_i \qquad \text{when } d_i \leq \varepsilon$$

where $d_i = \sqrt{(x_c - x_i)^2 + (y_r - y_i)^2}$, the distance of the i^{th} sample from the grid point and ε is a very small distance.

The advantage of these two algorithms is the speed of computation and the simplicity of programming (Thomas et al., 1986).

3.1.5 Factors that Affect the Gridding Procedure

In gridding algorithm, the user must be aware of the influences of certain parameters, they include sampling density, grid interval, and maximum or minimum number of points used in calculating a node value. Choice of algorithm is also influential.

3.1.5.1 Sampling Density

In general, increases in sample size are directly related to increase in the reliability of the entire map. The weighting function assigned to each data point is at least partially dependent on its distance. As the distance increases, the influence of a data point decreases. When the sample size is small, the nearest outer-most neighbors may be very far away and contribute little to the estimation process. Therefore, the closest data point has a much greater influence in the estimation of immediately surrounding grid points. As the sample size increases, the most distant nearest neighbors found will be closer and hence will contribute more to the interpolation process. This may introduce more variation in the estimated surface surrounding a given data point. At the same time, however, the nearest neighbors from a large sample size are likely to be more highly correlated than the nearest points from a small sample size. This tends to reduce the amount of variation in value estimation (Thomas et al., 1986).

3.1.5.2 Grid Interval

The grid interval controls the detail that can be retained in the grid; the interval should thus be made small enough to show required detail. On the other hand, creating a grid with too fine an interval wastes computer time and memory and, in some cases, causes unacceptable extrapolation. It is better to choose the grid interval so that a given grid square contains more than one data point. The best procedure for this is to estimate the typical spacing between near neighbor data points and then to use as the interval a convenient even value.

3.1.5.3 Proximity of Data Points

Most current algorithms require that good data selection routines be included, normally implying the use of search sectors. The number of such sectors usually ranges from two to sixteen, but eight is a common number. Better results generally come with more sectors, particularly with nonrandomly distributed data (Davis, 1973). Use of more than one point per sector usually increases stability.

3.2 Multi-Seam Predictions

The initial steps in the application of the hazard process begin with utilizing drill core and other data sources in the construction of borehole and fence diagrams over the proposed mining area. These diagrams not only allow for assessment of the reliability of geologic data, but also facilitate correlation of individual strata and development of elevation and thickness values. Depending on whether over- or under mining effects are to be assessed, spatial effects, including innerburden thickness, can be evaluated to determine if interaction is possible. Once features that can contribute to ground control problems are identified, the hazard technique can determine the severity of potential damage caused by interaction. To demonstrate application of the hazard approach, potential interaction in a lower seam has been evaluated.

The geologic, structural, and spatial distances between the seams are important in evaluating multi-seam operations, including the presence of sandstone in the innerburden (Szwilski, 1979; Johnson, 1973; Haycocks, 1991).

The traditional approach to multi-seam coal reserve estimation has been to set a fixed minimum innerburden thickness for all seams under consideration, regardless of previous mining activities. All of the coal undermined within that minimum thickness is considered lost. Any two seams are considered both minable only when the innerburden thickness between the two seams is greater than this minimum value (Zhou et al., 1990). This practice ignores the fact that even if an affected seam is minable, there may still be some coal losses due to interaction. Moreover, research into ground control in multi-seam mining has revealed that this distance varies considerably from mine to mine, sometimes from location to location within the same mine. Among the factors affecting this minimum innerburden thickness are rock type, sequence of mining, methods of extraction, mine geometries, thickness of the two seams, and the time lapse between mining of the two seams.

Based on the above arguments, it seems logical that the minimum innerburden thickness for room and pillar mining should have two values: a minable thickness and a recoverable thickness. The minable thickness is the critical innerburden thickness below which all coal will be considered lost. This value has been estimated to be equal to 12 feet, whereas the recoverable thickness determines if a seam is completely recoverable without the influence of interaction. This value has been estimated to be 110 feet (Haycocks and Karmis, 1983). Any seam with an innerburden thickness between these two values will suffer a corresponding loss of coal. Based on this procedure, the percentage of coal recovery in an affected lower seam is calculated as follow:

$$\% \text{ Recovery} = \frac{I_t}{D_c} \text{ X } 100$$

where I_t is the innerburden thickness and D_c is the minimum innerburden thickness as a function of percent hardrock, usually sandstone, (Haycocks and Karmis, 1983) from the case study data in Figure 3.4. It is defined as follows:

$$D_c = 110 - 0.65S$$

where S is the percent hardrock in the innerburden.

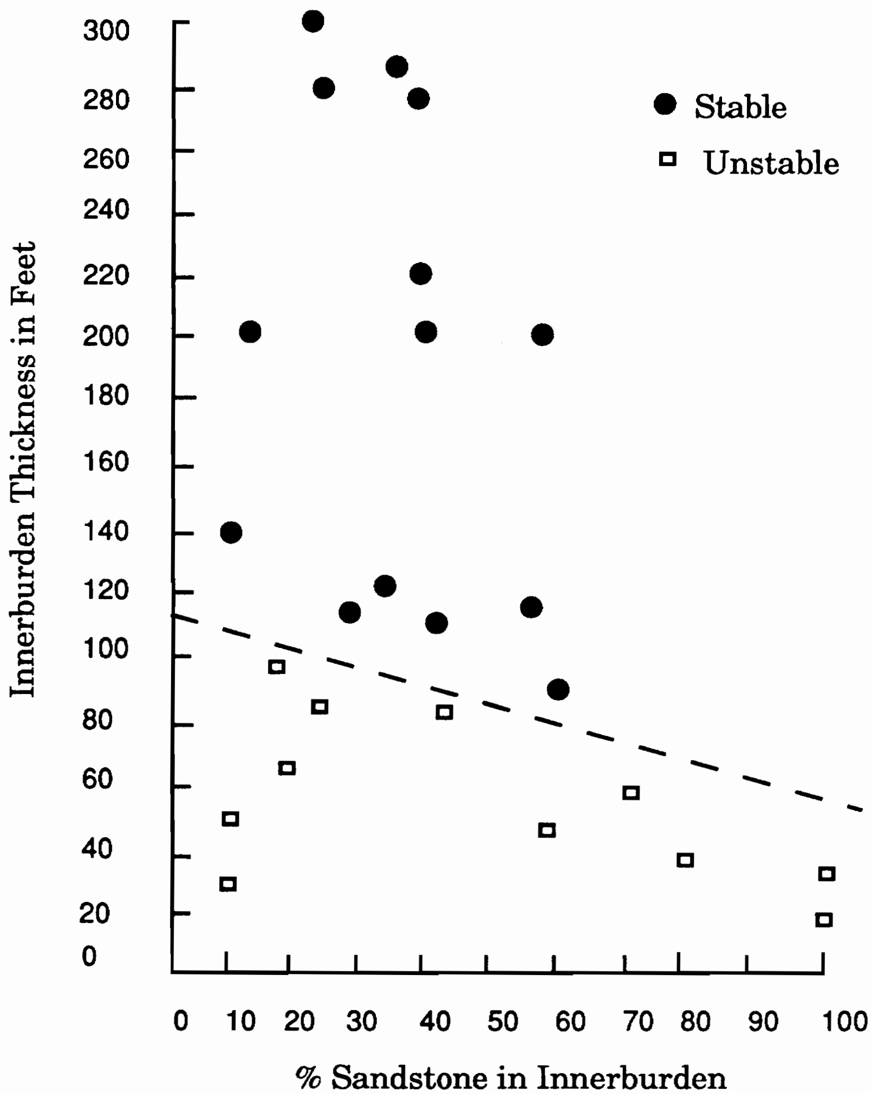

Figure 3.4 Influence Of Percent Hardrock In The Innerburden On Stability In The Lower Seam (Haycocks and Karmis, 1983)

Chapter 4
Program Formulation

4.1 Program Construction

As individual seams are mined out, coal recovery from seam that overlie or underlie the mined out seam may be significantly reduced, depending upon the magnitude of the multi-seam interaction (Peng, 1986). Under such conditions, predicting the precise location and amount of interaction damage in an affected seam must be determined to optimize overall seam extraction. Thus, the purpose of the HAZARD program is to provide calculated tonnages suitable for mine planning and design, in the form of recoverable coal tonnage determination and the chemical and structural characteristics in an affected seam.

The HAZARD program is straightforward in concept and application. It has been developed to convert raw borehole, outcrop, or other data pertaining to a proposed mining property into engineering parameters suitable for design purposes. This has been accomplished by determining the parametric various of seam characteristics over the entire mining property. To achieve this objective, an interpolative technique that can predict data variation across the entire mining area, was selected.

For systematic evaluation of the mining area, the ground is divided into regularly shaped blocks; thus, a gridding technique which is the basis and the first stage of the HAZARD program was developed. Three different gridding algorithm subroutines are used in the program; each subroutine uses the randomly placed values of exploration drillhole data to assign values to a regular grid. The dimensions and size of the grid are specified by the program user. The second stage is to determine the recovery percentage, based on data from boreholes, using the hazard technique described in Chapter three; hence, the recoverable coal is determined in an affected seam.

The flowchart illustrating the general construction and flow of the HAZARD program is shown in Figure 4.1. First, the borehole input data file is read which consists of the x and y coordinates, overburden depth, seam thickness, BTU's/lb, sulphur percentage, ash percentage, innerburden thickness, and the percent sandstone values. The mine boundary coordinates file is also read at this time and may contain a single or several mine boundaries.

Then the user will be asked to specify:

1) The number of grid points they wish to use in the x-direction. The number of grid spaces will be one less than the number of grid points specified.

2) The number of grid points they wish to use in the y-direction. The number of grid spaces will be one less than the number of grid points specified.

The number of grid spaces multiplied by the distance between the grid points will determine the full east-west (north-south) range of the grid. Care should be taken to ensure that this range encompasses the full east-west (north-south) extent of both the proposed mining area and the data points.

3) The distance between successive x grid points.

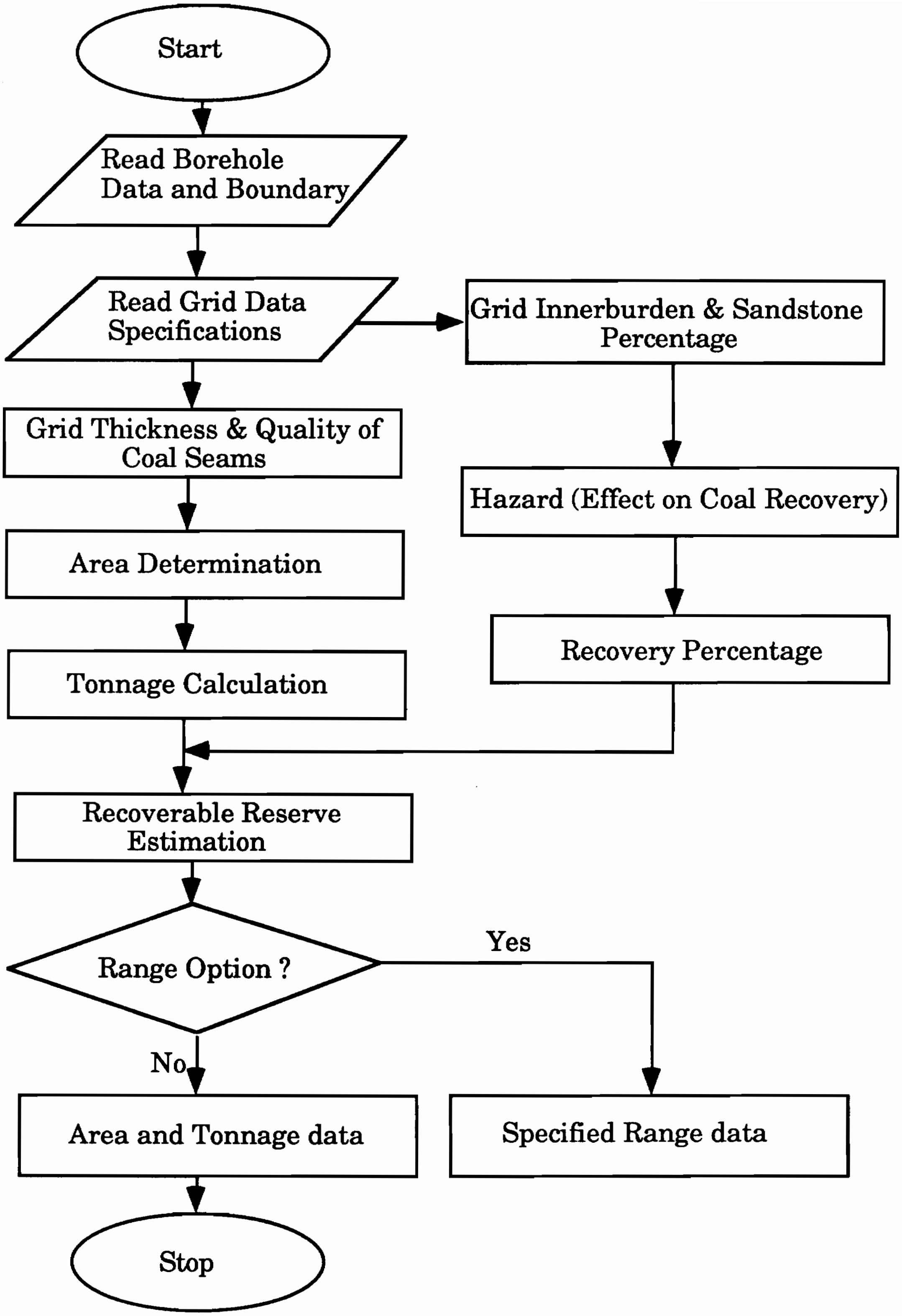

Figure 4.1 Flowchart Showing Construction Of The HAZARD Program

4) The distance between successive y grid points.

This distance multiplied by the number of grid points minus one will determine the East-West (North-South) range of the gridding area.

5) The x location of the corner coordinate point of the grid.

6) The y location of the corner coordinate point of the grid.

7) The number of neighboring points to estimate each grid point.

At this point, the information contained on this set of data will match the specifications used in gridding the data.

The program will then estimate coal thickness and coal quality parameters for each grid point. Furthermore, the program will grid the innerburden thickness and the percent sandstone to evaluate the potential interaction damage as percentage loss for each grid using the hazard technique.

To complete the coal reserves evaluation, MULSIM, RUMSIM, and MSEAM computer programs, developed for multi-seam mining, can be used to predict the damage level for specific mine conditions (Figure 4.2).

Recoverable tonnages are calculated by averaging the four corner thickness and percentage recovery values of the grid, and using these in conjunction with the area of the grid block to obtain the tonnage. In addition to the coal reserves, the mining area is evaluated in terms of the average sulphur percentage and the respective tonnages inside certain sulphur limitations. The coal is also evaluated in the same manner for BTU content and ash content.

In addition to determining the recoverable reserves, the program calculates the average seam thickness, which is critical in equipment

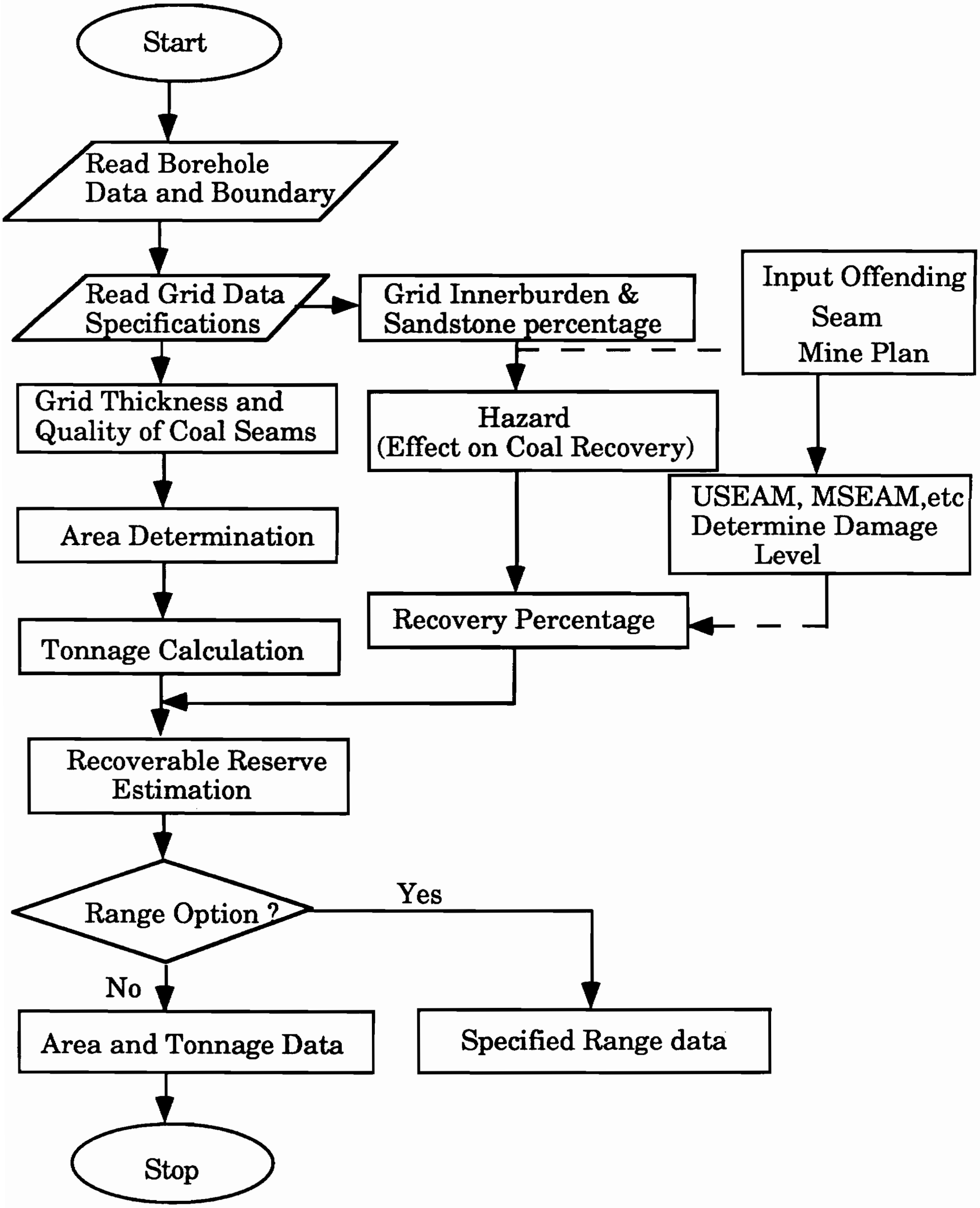

Figure 4.2 Correlation Between The HAZARD Program And Others Conducted On Multi-Seam Mining

selection and system development. Also, the mining area is broken up into a regular grid the size of which is specified by the program user. Recoverable tonnages, sulphur content, ash content, and BTU content, among other parameters, are all examined on a block-by-block basis, which facilitates mining layout, mine design, and proposed mandatory blending. Finally, the program calculates the seam stress due to the weight of the superincumbent strata. With all those calculations, the program assumes that the boreholes are drilled vertically and that corrections to the tonnage figures are not applied due to variations in the seam dip.

4.2 Construction Details

The HAZARD program contains the following major functions:

4.2.1 Neighbor

This function specifies the sample data points that are to be used to estimate an element in the grid matrix.be found by a nearest neighbor search around that element. This function is able to search out the points nearest to each grid intersection by calculating the distance from that grid intersection to the surrounding data points. Having found the distances to the n nearest data points, one can estimate the grid point by using either the inverse squared distance or inverse distance method described in Chapter 3. Estimated grid points are then stored in a matrix form.

4.2.2 Quadrant

This function specifies the sample points that are to be used to estimate an element in the grid matrix. First, the function divides the area around a grid point into four equal segments and determines the data points for each quadrant. Then, it finds a certain number of nearest data points, specified by the program user, by comparing the distance between the current grid point and all data points in each quadrant. Finally, one can estimate a grid point value by using either the inverse squared distance or inverse distance method.

4.2.3 Octant

This function determines the sample data points used to estimate an element in the grid matrix based on an octant search around that element. An octant search procedure divides the area around a grid node into eight equal segments and determines the data points for each octant. In a manner similar to a quadrant search, this function searches for the nearest data points, specified by the program user, by calculating the distances between the grid point and all data points in each octant. Using inverse squared distance formulas, the grid point values are estimated and stored in a matrix form.

4.2.4 Average Thickness Inside Boundary

This function calculates the average thickness inside a mining boundary. First, using the block inside boundary function determines how many complete blocks are inside the mining boundary, and sums the average thickness corresponding to each block. Second, the number of blocks intersecting the mining boundary is determined using the block boundary intersection function which uses the following functions:

* Segments intersect: It describes whether two segments, one segment of the mining boundary and the other from the block, intersect .

* Block segments intersection: This function examines if one segment of the mining boundary, from one point to the next, intersects with one segment of the block: bottom wall segment, right wall segment, top wall segment, and left wall segment. At the same time, this function counts the number of incomplete blocks intersecting the mining boundary and adds the average thickness of each block.

After examining each block of the entire map area with the mining boundary, we end up with the total number of blocks inside the boundary and total thickness of that particular boundary. Dividing the latter by the first, we obtain the average thickness inside the mining boundary.

If several separate boundaries exist, the program calculates the average thickness of each boundary individually. The boundary coordinates of one individual boundary should be placed in one sequence, followed by those of another and separated by number -1. Sequence of all boundaries is not important.

4.2.5 Area

The area function calculates the area inside the mining boundary. The boundary points describe a polygon which is divided into regular trapezoids by running vertical lines through each point on the boundary. The first and last figure will, in general, be triangles.

The assignment of the mine boundary points in sequence, regardless of their spatial location, is essential to the process used in the function for determining the area inside the mine boundary.

4.2.6 Recovery Percentage

This function assigns a recovery percentage value at each grid point using the hazard technique discussed earlier for a lower seam. At each grid point, this function compares the estimated values of innerburden thickness to the minable thickness (12 feet), if the first is less than the second, a zero recovery percentage value is assigned to the grid. If the innerburden thickness is greater than the recoverable thickness (110 feet), the grid point has one hundred percent recovery value. When the innerburden thickness is between the minable and the recoverable thickness values, the equation developed by Haycocks and Karmis (1983) is used to determine the minimum innerburden thickness as a function of percent sandstone. If the innerburden thickness is greater than this value, a one hundred percent recovery is obtained, otherwise the innerburden and the minimum thickness ratio is calculated, hence a recovery percentage value is assigned to the grid point.

4.2.7 Ton Function

Tonnage and recoverable reserves were calculated for both block and mine areas by finding an average value for the thickness and multiplying this by the area to determine the volume.

Using the ton inside block function, the tonnage of each block of the grid area was determined by multiplying the average thickness of a block by the area of a block by a density of a block deposit. Using the recoverable ton inside block function, the recoverable reserves of each block of the grid were determined by multiplying the average percentage recovery of a block by the ton inside a block.

Therefore, the average thickness and average recovery percentage of a single block would be calculated from the equation as follows:

$$Z_B = \frac{Z_{(x,y)} + Z_{(x+1,y)} + Z_{(x+1,y+1)} + Z_{(x,y+1)}}{4}$$

Where Z_B = Average parameter value determined for the block.

$Z_{(x,y)}$ = Parameter value at grid intersection on corner of the block determining by using either a nearest neighbor search function, or a quadrant search function, or an octant search function.

In this case the tonnage and recoverable tonnage inside the mining boundary can be calculated by:

$$T_t = \frac{\Sigma t_b}{N_b} \cdot W \cdot A$$

$$T_r = \frac{\Sigma r_b}{N_b} . T_t$$

Where T_t = Total tonnage inside the mine boundary.

T_r = Recoverable tonnage inside the mine boundary.

t_b = Average thickness of a single block inside mining area.

r_b = Average recovery percentage of a single block.

N_b = Number of blocks inside mining area.

W = Average density of seam inside mining area.

A = Mining area.

4.3 Operating Information

HAZARD program operates by fitting a rectangular grid, the dimensions of which are determined by the program user, over the borehole data. The program then interpolates the random data values enclosed in the grids using one of the three gridding methods, to grid values for calculating recoverable coal reserves. To place the grid in space, the user prescribes the starting coordinates and the number and size of the grid spacings. The starting coordinates will determine the position of the grid in space. In the event that the grid is misplaced in terms of the data (does not cover it), the program will terminate with a stack flow error.

Once the overall grid size has been determined, a suitable grid spacing must be selected. This is largely left up to the program user and may be considered to be a function of raw data. For most coal seams where variations in seam thickness, sulphur content, ash, BTU's, and other chemical characteristics, tend to be gradual, the larger grid sizes should be chosen. A very coarse grid network may cause a loss of resolution in the gridded output, while too fine a grid will not improve the output either and will waste computer time. In general, the grid spacing should be from one-fourth to one-half the average distance between exploration data points.

Chapter 5
Case Studies

5.1 Case Study No. 1

5.1.1 Description

This case study is based on an Appalachian coal operation. Data for this operation are given in Appendix B (Final Technical Report, 1981) and is composed of:

1) Coal lower seam thickness in inches;

2) Calorific value in BTU per pound;

3) Sulphur percentage;

4) Ash percentage;

5) Overburden thickness;

6) Innerburden thickness; and

7) Percent sandstone from 34 exploration boreholes.

The spatial location of these boreholes inside a proposed mining area is shown in Figure 5.1. This figure shows that those exploration boreholes were not drilled on a regular grid. The initial step in inputting data is to choose a grid location and spacing to completely cover the raw data. After reviewing the data spread, the decision was made to utilize 23 grid spacing to the east and 26 to the north at an individual spacing of 1000 feet as shown in Figure 5.2. This figure shows that the grid fits over the raw data, giving it a reasonable distribution inside the grid area.

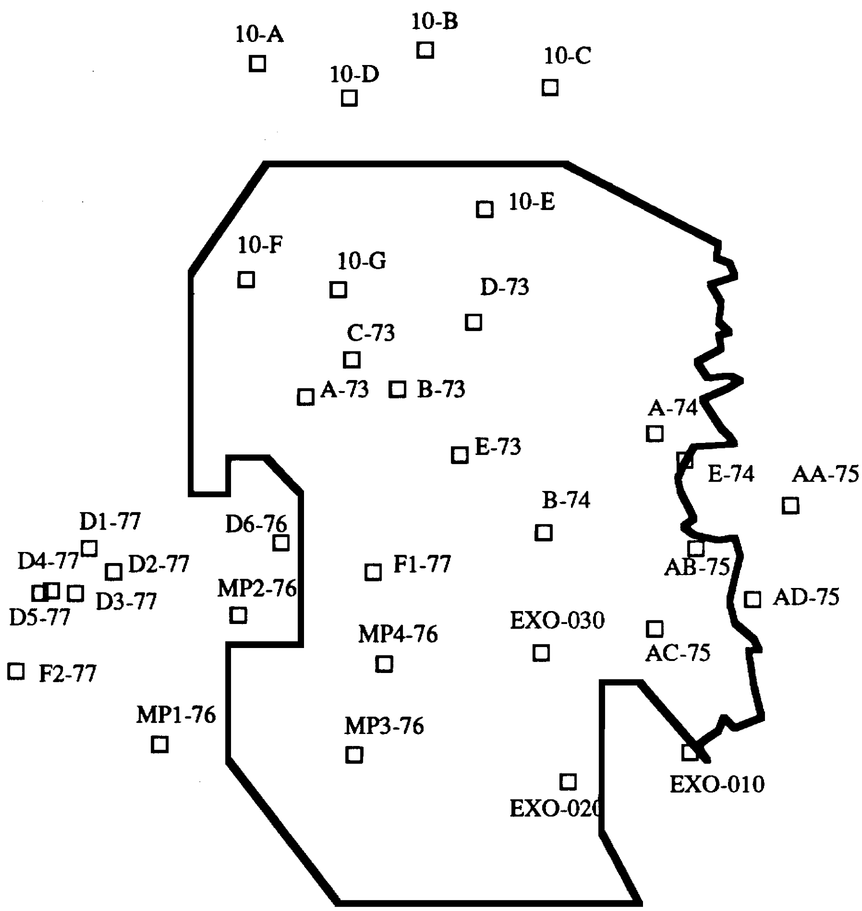

Figure 5.1 Borehole Locations Of Case Study No.1

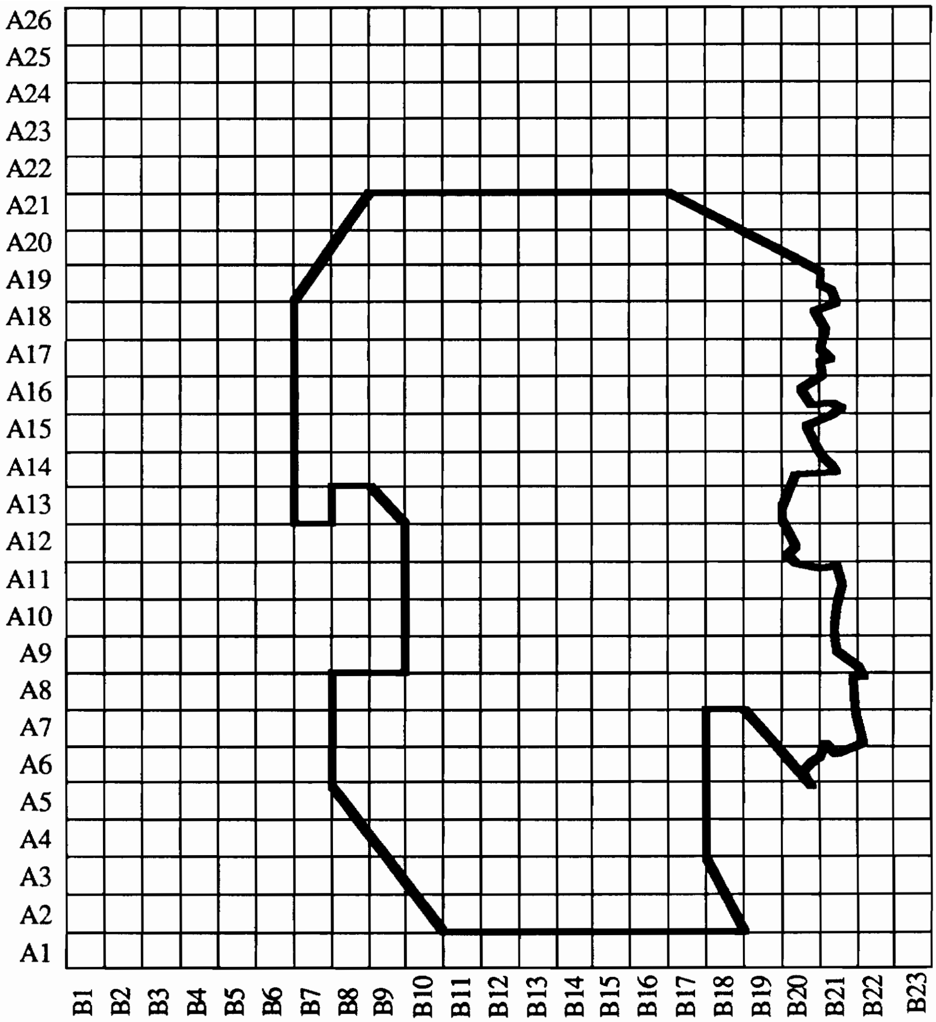

Figure 5.2 Relation Of Grid (1000 ft Grid Spacing) And Mine Boundary Of Case Study No.1

5.1.2 Analysis and Discussion

The borehole data were gridded using three different gridding algorithms including nearest neighboring search, quadrant search, and octant search. HAZARD program output reveals a slight difference between these three methods in terms of tonnages (Appendix B), but no method can be automatically judged best, because each has its own properties that are of value under given circumstances. These three methods are used in this program to provide the user with more choices and the user must decide which method is preferable for his/her data.

By changing the number of grids and grid spacing in both directions (east and north) so that the entire range of the exploration data and the proposed mining area are still covered (Figure 5.3), different output results are obtained (Appendix B). Thus, this grid interval parameter is very important in the technique used in HAZARD program. The user should be careful when selecting this distance because it controls the detail that can be retained in the grid. The same comment applies to the number of search sectors.

The percentage loss due to multi-seam interaction is calculated at each grid point. These values provide accurate estimates of available tonnages from the entire mine and from specific mining areas for different qualities of coal.

An examination of the data showed that part of the seam contained more than two percent sulphur; therefore, the distribution of sulphur in the seam was examined on a block-by-block basis. A breakdown on the block-

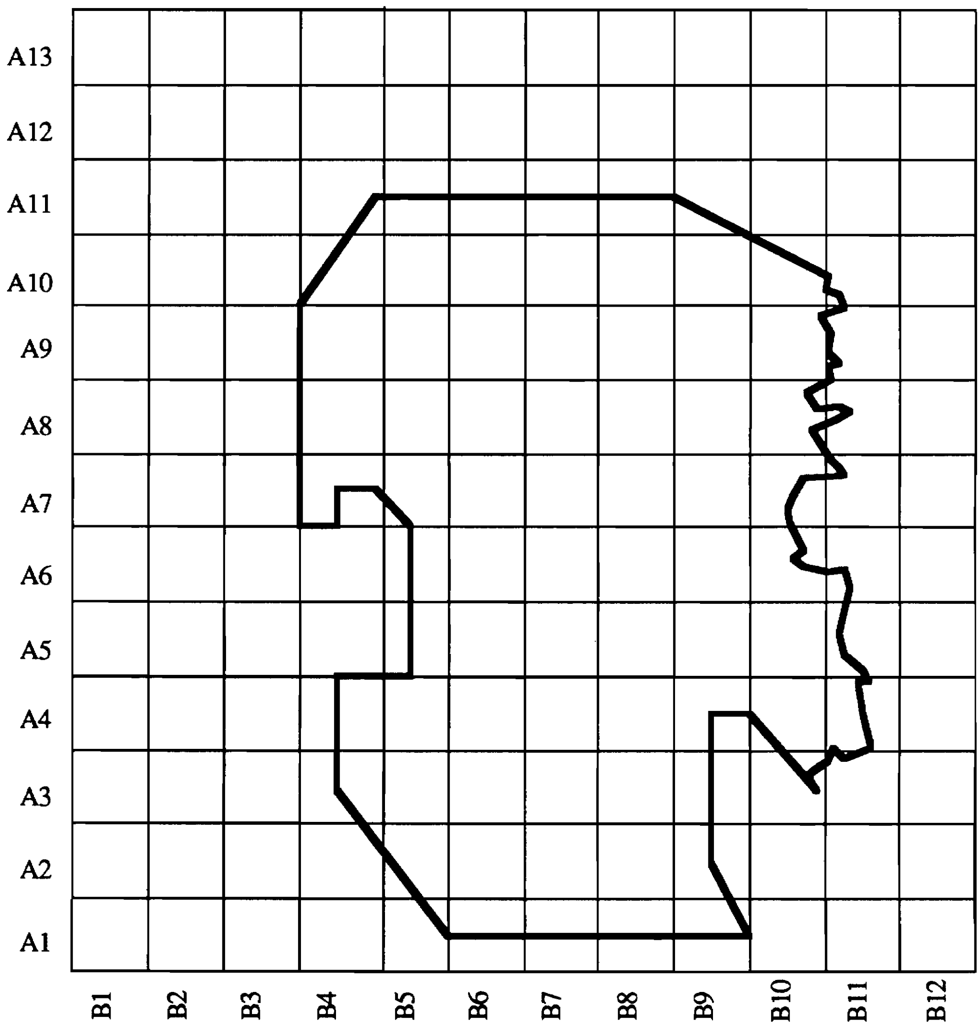

Figure 5.3 Relation Of Grid (2000 ft Grid Spacing) And Mine Boundary Of Case Study No.1

by-block tonnage for specific ranges of sulphur content was specified as shown in Appendix B.

To further expand the analysis, tonnage breakdowns were also specified in terms of various levels of ash and calorific value. The data resulting from the sulphur analysis could be used in estimating the amount of blending that would be necessary, as well as the actual coal tonnage that would be available inside a specific range.

This information and other chemical analyses, combined with the estimates of recoverable coal reserves, give the mining engineer a great deal of versatility in examining a seam for a number of potential uses. These data also facilitate coal blending as an integral part of the mining operation itself. Thus, the selection of face areas in suitable areas of the seam can yield a product of desired qualities.

The block-by-block breakdown of the seam into average thickness and average percentage loss provides an accurate means of assessing the sequence of mining. It also acts as a warning to the preparation plant that a high percentage loss may be encountered in mining certain areas.

The summarized HAZARD program output is shown in Appendix B; the initial figures are the accumulated tonnages inside the total grid area. Other figures included are the total tonnage figures inside the proposed mining area, as well as the recoverable coal figures inside the area based on the average value of the percentage loss due to multi-seam interaction. Average mining height is also included, as well as the average depth of the seam. The program output also shows the accumulated tonnages of sulphur, ash, and BTU's, as well as the accumulated recoverable tonnages in certain ranges set at any value convenient to the user. In Appendix C, block-by-block breakdown of the seam is summarized in terms of recoverable tonnages, average seam thickness, average mining depth, and average pressure.

5.2 Case Study No. 2

5.2.1 Description

This case study is based upon the Coors Energy Property at Kneesburg, Colorado (Hustrulid, 1985).

The reason to use this data was not only its ready availability, but also the opportunity to compare the results of the hazard technique used in HAZARD computer program with those from the Polygonal method used to calculate coal reserves within the boundary of the Coors Energy Property.

The coal density in this property is 73 lbs/ft^3. The surface is essentially horizontal with an elevation of 4800 feet. The coal seam dips at about 1° to the south west. The property boundary is described by the coordinates shown in Table 5.1.

The data shown in Appendix D were composed of:

1) Overburden thickness;

2) Coal Seam thickness;

3) Calorific value;

4) Sulphur percentage;

5) Ash percentage;

6) Innerburden thickness;and

Table 5.1 - Property Boundary Coordinates Of Case Study No.2

X- coordinates	Y - coordinates
41750.00	29565.62
41700.00	30650.00
41020.00	30870.00
40770.00	31970.00
40110.00	32830.00
39900.00	33030.00
38950.00	33000.00
38330.00	33620.00
37010.00	35240.00
34990.00	35360.00
34310.00	35300.00
33660.00	34960.00
32390.00	34810.00
31950.00	35660.00
31670.09	36400.00
31250.00	36400.00
31250.00	31300.00
36460.00	27850.00
41750.00	27850.00

7) Percent sandstone in the innerburden from 35 randomly distributed boreholes across the proposed mining area (Figure 5.4).
This figure shows that the boreholes are not extended beyond the edge of the proposed mining area.

The coal reserves lying within the minable boundary was computed using polygonal method. The polygons were constructed using a range of 1200 feet (Figure 5.5). The results are shown in Table 5.2 with 14.54 million tons of total coal reserves inside the mining boundary.

5.2.2 Analysis and Discussion

The borehole data were gridded using the neighbor search gridding algorithm. After reviewing the data spread, the decision was made to start the grid at 29,000 feet east, 25,000 feet north and utilize 16 grid spacing to both east and north directions at an individual spacing of 1,000 ft as shown in figure 5.6.

Based on the specifications used in gridding the data, HAZARD computer program output shows almost the same results (Appendix D) as polygonal method in terms of coal reserves inside the mining area: 14.42 million tons. However, the hazard technique used in the program has advantages over the polygonal method. The latter can only be used inside the boundary formed by the outer range of the sampling points (Figure 5.5), while the gridding and inverse squared distance techniques can be extended across the entire mining area, not just inside the range of sampling points. Also, the block-by-block breakdown of the ground provides

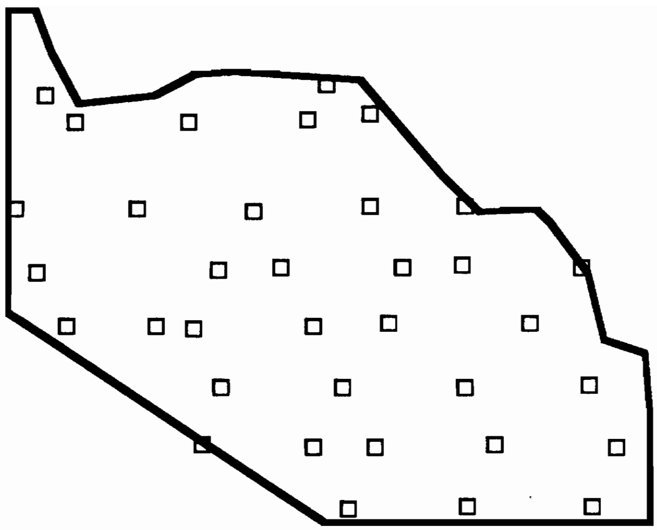

Figure 5.4 Borehole Locations Of Case Study No.2

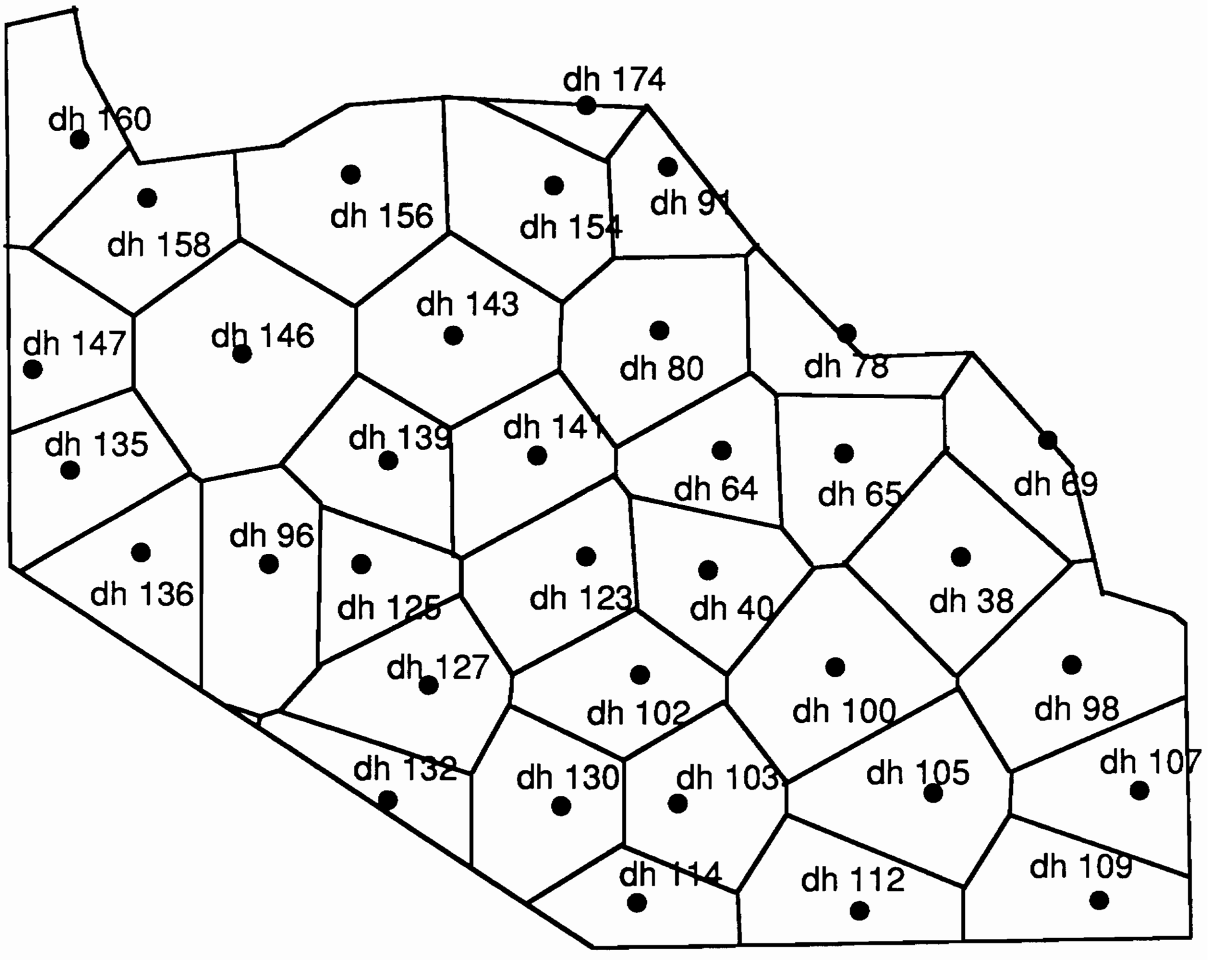

Figure 5.5 Polygonal Method With Property Boundaries Of Case Study No.2

Table 5.2 - Polygonal Method Results Of Case Study No.2

Hole Id	Area (ft^2)	Pre B Vol (ft^3)	Tons Coal
dh 147	1307806.9	968745.9	262542.2
dh 135	1536334.2	1138025.3	381318.1
dh 160	1464361.5	1084712.2	331385.0
dh 136	1556620.4	1153052.2	403398.2
dh 158	1697809.3	1257636.5	340835.2
dh 145	2916366.6	2160271.6	787710.6
dh 46	2057464.0	1524047.4	540701.5
dh 156	2562768.6	1898347.1	626725.1
dh 125	1133240.0	839437.1	301951.8
dh 132	942904.0	698447.4	178963.2
dh 139	1545057.0	1144486.7	377843.7
dh 127	1842716.7	1364975.3	470814.1
dh 143	2168098.1	1695998.6	633084.6
dh 141	1584945.5	1174033.7	468589.1
dh 154	1932152.6	1431224.2	500717.4
dh 130	1787953.9	1324410.3	489452.4
dh 123	1745061.8	1292638.4	471341.2
dh 174	433205.9	320893.3	79060.1
dh 102	1525069.3	1129681.0	423054.2
dh 114	1148671.1	850867.5	297678.1
dh 80	2201393.6	1630661.9	506210.5
dh 91	1067775.5	790944.8	280611.4
dh 103	1714044.3	1269662.5	381632.0
dh 40	1662698.1	1231628.2	503714.4
dh 64	1358679.3	1006429.1	292591.6
dh 65	1696191.5	1256438.1	439568.0
dh 100	2146430.3	1589948.4	493571.6
dh 78	1151905.1	853263.0	1493210.3
dh 112	1614592.5	119599.4	477354.3
dh 105	2041373.9	1512128.8	581179.2
dh 38	1997808.6	1479858.3	525024.1
dh 60	1106687.7	819768.7	210049.3
dh 98	2184975.6	1618500.5	582186.8
dh 109	1568232.1	1161653.4	274754.3
dh 107	1497029.5	1108910.8	404347.7
Totals:	57898424.9	42887722.1	**14538592.5** **14.53 million**

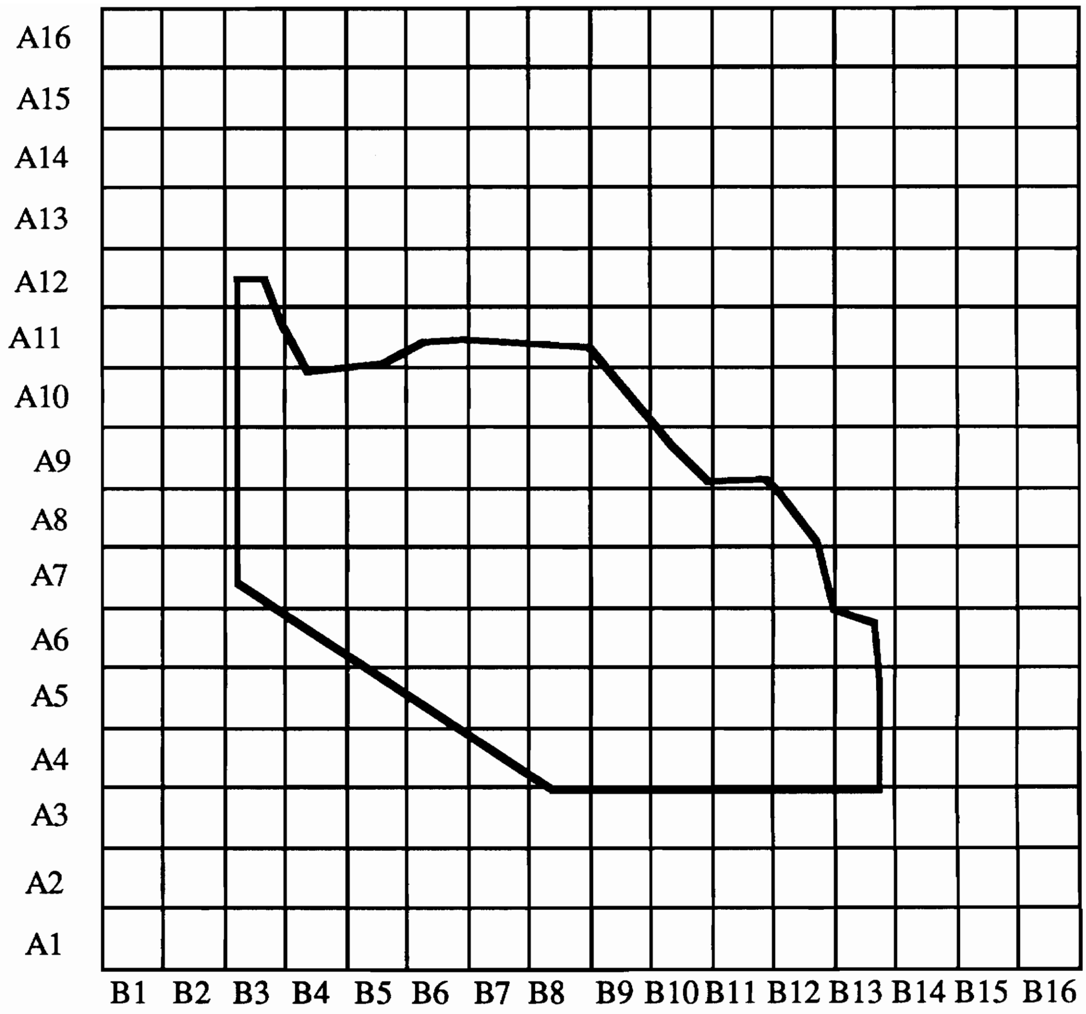

Figure 5.6 Relation Of Grid (1000 ft Grid Spacing) And Mine Boundary Of Case Study No.2

more accurate information to the preparation plant to qualitatively and quantitatively evaluate the coal deposit. In addition, the borehole statistics is a gross averaging technique, and the block accumulations is further averaging of the grid values, we can conclude that this hazard technique itself is flexible and capable of dealing with evaluation of coal deposit, and it is well suited to the interpolation of irregularly spaced borehole data.

5.3 Case Study No. 3

This case was made to test and verify the program dealing with multiple mining boundaries.

As in previous cases, the data in Appendix E were composed of coal thickness, calorific values, sulphur percentage, and ash percentage from 24 exploration boreholes randomly distributed in three different mining boundaries as shown in Figure 5.7.

The area was gridded using 10 grid points in the east direction and 9 grid points in the north direction at an individual spacing of 1,000 feet as shown in Figure 5.8. The HAZARD computer program output verifies that the program evaluates the mining boundaries individually in terms of reserves determination (Appendix E)

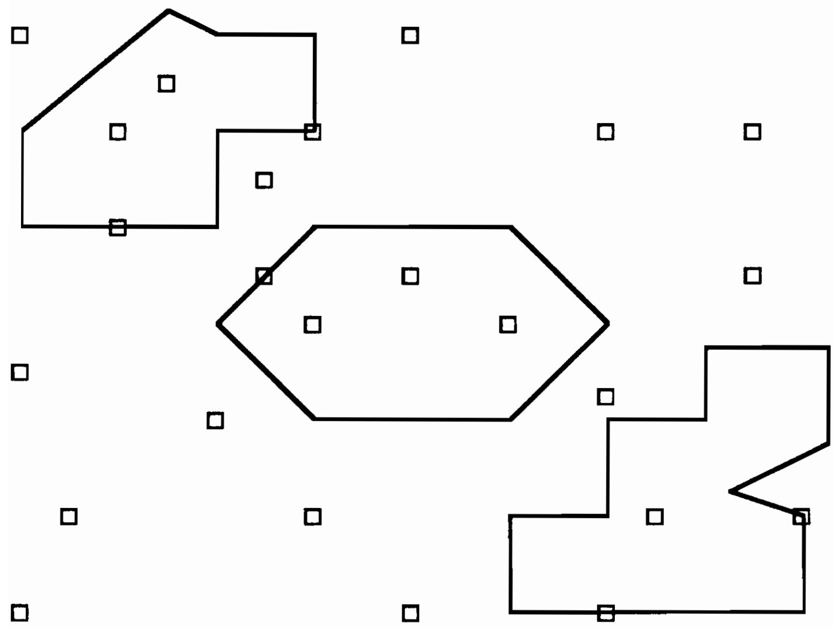

Figure 5.7 Borehole Locations Of Case Study No.3

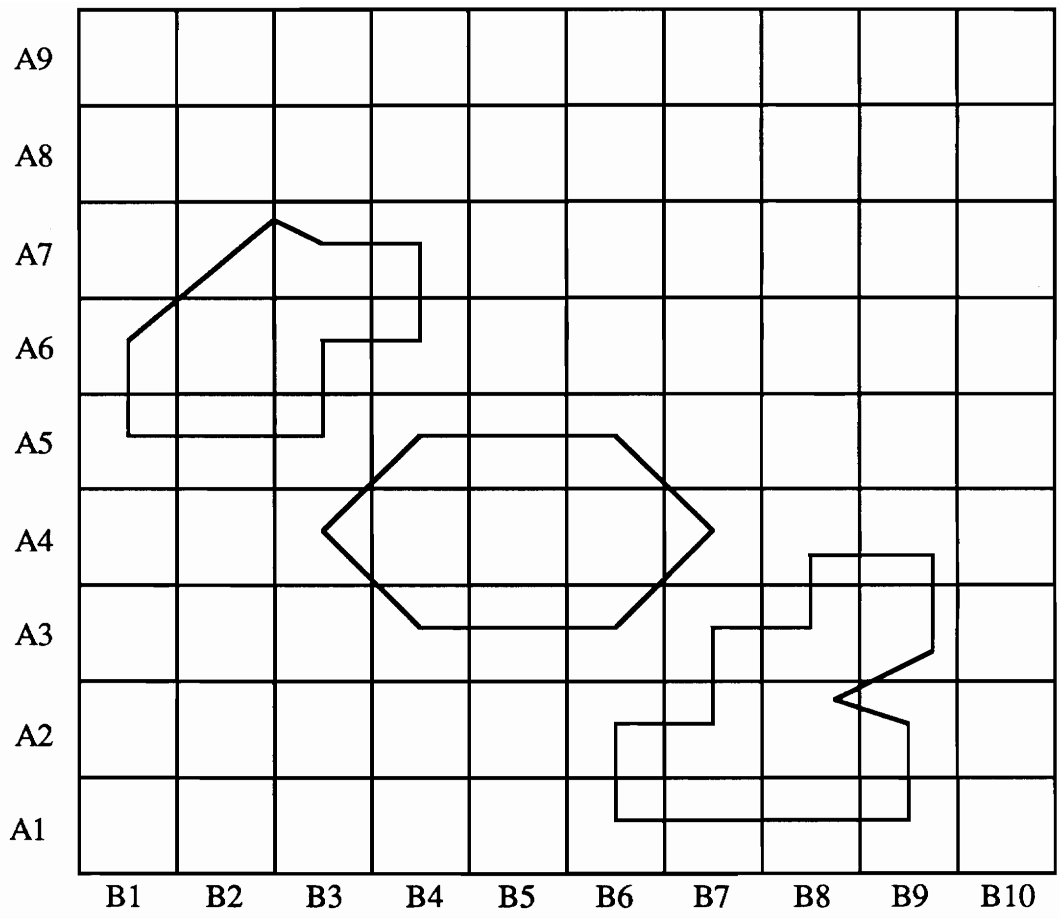

Figure 5.8 Relation Of Grid (1000 ft Grid Spacing) And Mine Boundary Of Case Study No.3

Chapter 6
Conclusions and Recommendations

6.1 Conclusions

Based on this research, the following conclusions are made:

1) The extrapolation and gridding techniques used in HAZARD can be extended across the entire mining property, and is not limited to the area inside the sampling points.

2) The method successfully grids irregularly spaced borehole data. The gridded data is suitable for use in subsequent calculations to determine recoverable coal reserves.

3) The inverse squared distance and inverse distance methods, offer acceptable sensitive techniques that are suitable for interpolation of randomly distributed exploration data.

4) The hazard approach is applicable to the entire affected seam. It permits the engineer to assess the degree of possible damage due to interaction and therefore facilitates evaluation of the recoverable reserves.

5) Tonnages and chemical characteristics of the seam are all examined on a block-by-block basis in the mining area, which provides an accurate means of assessing the possible use of the seam. Where necessary, these breakdowns can form the first step in the blending determination of the coal production sequence.

6) An interactive computer program called HAZARD has been developed to evaluate the recoverable coal reserves in multi-seam mines. The program provides an efficient, accurate, and reproducible means of

determining tonnages inside a proposed mining area. It is simple to use and provides mining engineers with accurate information to aid in their evaluation, as well as providing a summarized listing of data for subsequent engineering analysis.

6.2 Recommendations

Research into determining recoverable reserves in a multi-seam environment will continue to improve toward an optimum mining plan. The method presented in this research will provide the coal operator with sufficient information to evaluate the recoverable reserves in an affected seam. However, some recommendations can be made in refining the method for further and more versatile engineering analyses. They are as follows:

1) Continue research into determining the most effective, reliable and efficient method of data interpolation, because the inverse squared distance and the inverse distance methods used in this research have limitations in determining whether peaks or valleys might exist in the data.

2) For wider applications, new options should be added to HAZARD, such as the ability to draw contour maps of some parameters such as average seam thickness and coal loss percentage. These will give the engineer a view of the seam and related characteristics as they vary throughout the proposed mining property. Hence, maximum reserve extraction can be achieved and optimal design can be facilitated.

3) The program has the potential to be expanded to suggest sequences of face mining for coordinated and pre-determined quality control.

4) Further testing and improvement of the program can be made under field conditions, particularly to increase the options available in terms of the data input.

REFERENCES

Chekan, G. J., Matetic, R. J., and Galek, J. A., 1986, "Strata Interactions in Multiple-Seam Mining-Two Case Studies in Pennsylvania," **RI 9056**, U.S. Bureau of Mines.

Clark, I., 1982, " Practical Geostatistics, "**Applied Science Publishers Ltd.**, London.

Crawford, J.J and Hustrulid, W.A., 1979, "Open Pit Mine Planning and Design," **Society of Mining Engineers**.

David , M., 1977, "Geostatistical Ore Reserve Estimation, " Ecole Polytechnique de Montreal, PQ, Canada.

Davis, J. C., 1973, " Statistics and Data Analysis in Geology," **John Wiley & Sons, Inc.**, NewYork.

Ehgartner, B. L., 1982, "Pillar Load Transfer Mechanisms in Multi-Seam Mining," **M.S. Thesis**, Virginia Polytechnic Institute and State University, Blacksburg, VA, May, 1964 pp.

Engineers International, Inc., 1981, "Investigation of Problems and Benefits of Underground Multiple Seam Coal Mining,"Final Technical Report. Dept. of Energy Contract DE-AC01-70ET14242, 292 pp.

Final Technical Report, 1981, "Design Optimization In Underground Coal Systems" Vol. II, **U. S. Department of Energy** pp 31-41.

Fraher, R., Sellami, M., Haycocks, C., and Karmis, M., 1991, "A hazard Approach to Multi-Seam Reserve Determination," **Proceedings**, 9th Annual Workshop Generic Mineral Technology Center Mine Systems Design and Ground Control, Lexington, KY, November 10-12, pp. 81-89.

Grenoble, A., Haycocks, C., and Wu, W., 1985, "Computerized Evaluation of Near-Seam Interaction," **Proceedings**, 2nd Conf. on the Appl. of Computers to the Coal Industry, Tuscaloosa, Al, pp. 317-322.

Grenoble, A. and Haycocks, C., 1985, "Design Factors in Near Seam Interaction," **Proceedings**, 4th Conf. on Ground Control in Mining, West Virginia Univ., Morgantown, WV, July.

Harding, James E., 1923, "How to Calculate Tonnage and Grade of an Ore Body," **Eng. and Mining Journal**.

Haycocks, C., 1980, "Optimization Extraction for Multi-Seam Mining," **Final Report**, Allotment Grant, O.S.M., 75 pp.

Haycocks, C., and Karmis, M., 1983, "Ground Control Mechanisms in Multi-seam Mining," **Final Report** prepared for the Office of Mineral Institutes, Bureau of Mines, U.S. Department of the Interior, Oct., 328 pp.

Haycocks, C., 1991, "Ground Control in Multi-Seam Mining," **Proceedings** 21st Institute on Coal Mining Health, Safety and Research, Blacksburg, VA, pp. 123-131

Holland, C. T., 1951, "Multiple-Seam Mining," **Coal Age**, August, pp. 89-93.

Hughes, R.L., Jr., 1959, "Volume Estimates from Contours, " **Econ. Geol.**, V. 54, No 5.

Hustrulid, W.A., 1985, "Surface Mine Design Laboratory MN 312L," **Colorado School of Mines.**

Johnson, G., 1973, "Rock Mechanics, a Nomogram for the Assessment of Roadway Conditions," **Colliery Guardian**, Vol. 221, pp. 16-20.

Lazer. B., 1965, "Mining Seams above Mined Out Seams," **Mining Engineering**, Sept., pp. 75-77.

Matheron, G., 1963, "Principles of Geostatistics," **Econ. Geol.**, 58: 1246-1266

Matheron, G., 1971, "The Theory of Regionalized Variables and its Applications." Les Cahiers Du Centre De Morphologic Mathematique, Booklet No.5, **Ecole Nat. Sup. Des Mines**, Paris.

O'Brien, D.T., and Weiss, A. 1967, "Practical Aspects of Computer Methods in Ore Reserve Analysis," **CIMM**, Special Vol. 9, pp. 109-113.

Peng, S.S, 1986, "Coal Mine Ground Control," 2nd Ed. **John Wiley**. pp. 295-313.

Popoff, C.C., 1966, "Computing Reserves of Mineral Products: Principles and Methods, " **I.C 8283**, U.S. Bureau of Mines.

Report of Sub-committee on Coal Mining to Committee on Ground Movement and Subsidence. 1926, Trans. **AIME**, Vol. 74: pp 734-809.

Sampson, R. J., 1975, "Surface II Graphics System," **Kansas Geological Survey**, Lawrence, Kansas.

Shklyarskiy, F.N., 1921, "A Method of Computing Reserves for Nestlike Ore Bodies," **Mining Journal**. No 4-5.

Sobolevsky, P. K., 1931, "Reserves Estimates of Metals In Placers by Method of Isolines," **Nonferrous Metals**, No.7, pp. 926-949.

Soderberg, A., 1930, "Mining Methods and Costs at The Utah Copper Co., " **Bureau of Mines Information Circular 6234**, Bingham Canyon,Utah.

Stemple, D. T., 1956, "A Study of Problems Encountered in Multiple-Seam Mining in the Eastern United States," **M.S. Thesis**, Virginia Polytechnic Institute and State University, Blacksburg, VA., July, 125 pp.

Stoddart, A.C., 1929, "Mining Practice and Methods at Inspiration Consolidated Copper Co.," Inspiration, Arizona **Bureau of Mines Information Circular**. 6169, 1929.

Styler, A. N. and Dunham, R. K., 1980, "Strata Deformation Above Longwall Faces," **Proceedings**, 21st U.S. Symposium on Rock Mechanics, Rolla, MO.

Szwilski, A. B., 1979, "Stability of Coal Seam Strata Under Mined by Room and Pillar Operations," **Proceedings**, 20th U.S. Symposium on Rock Mechanics, pp. 59-65.

Thomas, A. J., David, E. H., Carlton, R. J., 1986, " Contouring Geologic Surfaces with the Computer," **Van Nostrand Reinhold Company, Inc.**, NewYork

Thomas, R.W., 1929, " Mining Practice at Ray Mines, Nevada Consolidated Copper Co.," Ray, Arizona **Bureau of Mines Information Circular 6167**.

Webster, S., Haycocks, C., and Karmis, M., 1984, "Subsidence Interaction Effects in Multi-Seam Mining," **Proceedings**, 2nd Int'l. Conference on Stability in Mining, Lexington, KY, Aug., pp. 589-604.

Wolff, J.W., E.L. Derby, and W.A. Cole, 1925, "Sampling and Estimating Lake Superior Iron Ores," Trans. AIME, v.72, 1925.

Wu, W., Haycocks, C., and Zhou, Y., 1987, "Application in Design for Close Proximity Multi-Seam Mining," **Proceedings**, 6th International Conference on Ground Control in Mining, W. Va. University, Morgantown, pp. 97-105.

Wu, W. and Haycocks, C., 1986, "Statistical Analysis of Interaction Problems in Close-Proximity Multi-Seam Mines," **Proceedings**, 27th U.S. Symp. on Rock Mechanics, Tuscaloosa, AL, June, pp. 317-323.

Zhou, Y., 1989, " Evaluating the Impact of Mining and Underlying Seam on Recoverable Coal Reserves in an Overlying Seam,"Paper presented at the 28th International Geologic Congress, Washington, D.C., July 9-19, pp 3-445, 3-446.

Zhou, Y., Haycocks, C., and Forrest, P. A., 1989, "Designing and Planning for Multi-Seam Mining," **Proceedings**, 1989 Multi-National Conference on Mine Design and Planning, Lexington, KY, May 1989, pp.41-46.

APPENDIX A

HAZARD Program Listing

```
/*    RESRVES IN MULTIPLE SEAM MINES PACKAGE FOR PC'S
      MODULE NAME = HAZARD.C
      HAZARD REQUIRES IBM MICROSOFT QUICK C FOR WINDOWS VERSION
           3.0 MAY 1990
      PROGRAMMED BY MONCEF SELLAMI
           DEPARTMENT OF MINING AND MINERALS ENGINEERING
           VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
           BLACKSBURG, VA 24061
                MAY 1993
HAZARD program has been developed to calculate recoverable coal
reserves in multi-seam mines. */

                          /* HAZARD.h */
                          /************/
#define FALSE          0
#define TRUE           1

#define MAX_CROSS      10
#define  MAXROW             40
#define  MAXCOL                  40
#define MAX_CROSS_POINTS    10

#define MAXPT   35
#define MAXPNT  62

FILE *in,*out,*bound,*sulf,*btu,*ash, *recov;

                          /* Function Prototype */
                          /* ------------------ */

int block_inside_boundary(double x1,double x2,double block_y,int nbpnts,
                                   double xb[],double yb[]);
int compare_x(double * x1p, double * x2p);
int inside_block ( double x1,double x2,double y1,double y2,double xp,
                              double yp);
int even ( int number);

int segments_intersect(double xas,double yas,double xae,double yae,
                 double xbs,double ybs,double xbe,double ybe);
int block_boundary_segm_intersect(int row,int col,int segnum,
                                  double xb[],double yb[],int nb);
int sign(double x);
int compare_xs(double * x1p, double * x2p);
int point_inside_boundary(double x,double y,double xb[],double yb[],int
                 nb);
double block_area( int row, int col, double xb[], double yb[], int nb);
void tonnage_inside_boundary( double mine_area,double density,
                              double avg_thick, double *total_ton);
void retonnage_inside_boundary(double avg_recov,double total_ton,
                              double *total_reton);
void avgthick_inside_boundary(  double y_base,        double x_base,
double y_grid,double x_grid,int ny,int nx,double xb[],double yb[],
int nb,double thick[MAXROW][MAXCOL],double * avg_thick);
double area (int n,double x[],double y[],double *areap);
void grid_data (double x[MAXPT],double y[MAXPT],double z[MAXPT],
int n,int np,double xo, double yo,double dx,double dy,int ny,
int nx,double matrix[MAXROW][MAXCOL]);
void gridq_data (double x[MAXPNT],double y[MAXPNT],double z[MAXPNT],
int n,int np,double xo,double yo,double dx,double dy,int nx,int ny,
double matrix[MAXROW][MAXCOL]);
void grido_data (double x[MAXPNT],double y[MAXPNT],double z[MAXPNT],
int n,int np,double xo,double yo,double dx,double dy,int nx,int ny,
double matrix[MAXROW][MAXCOL]);
```

```
void Recovery_percentage (double matrix1[MAXROW][MAXCOL], matrix2[MAXROW][MAXCOL], int ny,
int nx, matrix[MAXROW][MAXCOL]);
void display_matrix ( double matrix[MAXROW][MAXCOL],int ny,int nx);
void block_thickness (  double matrix[MAXROW][MAXCOL], int ny, int nx,
double  avg_block[MAXROW][MAXCOL]);
voidblock_range(doublerange1,doublerange2,doubleblock_ton[MAXROW][MAXCOL],double
real_bl_ton[MAXROW][MAXCOL],int ny,int nx, double avg_block[MAXROW][MAXCOL],double *
accum_ton, int * block, double * accum_reton);
void tons_inside_grid ( int ny, int nx, double lock_ton[MAXROW][MAXCOL],
double real_bl_ton[MAXROW][MAXCOL],double * total_tons, double * total_retons);
void ton_inside_block ( int ny, int nx,double dx,double dy,
double avg_bl_thick[MAXROW][MAXCOL],double  block_ton[MAXROW][MAXCOL]);
void reton_inside_block ( int ny, intnx,doubleblock_ton[MAXROW][MAXCOL],
double avg_bl_recov[MAXROW][MAXCOL],double real_bl_ton[MAXROW][MAXCOL]);
void press_inside_block( intny,intnx,doubleavg_bl_depth[MAXROW][MAXCOL],
double burd_density, double avg_bl_pressure[MAXROW][MAXCOL]);
void avg_inter_percent(        int ny, int nx, doubleavg_block[MAXROW][MAXCOL],
                               double * avg_inter);
void main();
```

/***********/
HAZARD.C
/***********/

```
/#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <errno.h>
#include "hazard.h.h"

double x_base ;
double y_base ;
double x_grid;
double y_grid ;

                                     /*********************************/
                                     /* Segment Intersection Function */
                                     /*********************************/

int segments_intersect(double xas,double yas,double xae,double yae,
                       double xbs,double ybs,double xbe,double ybe)

{
  double xa = (xae-xas);
  double ya = (yae-yas);
  double xb = (xbe-xbs);
  double yb = (ybe-ybs);
  double alpha,beta;
#define xab     (xbs-xas)
#define yab     (ybs-yas)
#define IN_RANGE(x)     ((x)>0 && (x)<1)
  if(xa == 0)   // ya cannot be zero
  {
    if(xb == 0)
      return FALSE;
    beta = -xab / xb;
    if(!IN_RANGE(beta))
      return FALSE;
    alpha = (yab+beta*yb)/ya;
  }
  else
  if(ya == 0)   // xa cannot be zero
  {
    if(yb == 0)
      return FALSE;
    beta = -yab / yb;
```

```
    if(!IN_RANGE(beta))
      return FALSE;
    alpha = (xab+beta*xb)/xa;
  }
  else
  {
    beta = (xa*yab-ya*xab)/(xb*ya-xa*yb);
    if(!IN_RANGE(beta))
      return FALSE;
    alpha = (xab+beta*xb)/xa;
  }
   return IN_RANGE(alpha);
#undef xab
#undef yab
}
                                       /* Block Segments Intersection */
                                       /*******************************/

int block_boundary_segm_intersect(int row,int col, int segnum, double xb[],
                                       double yb[],int nb)

{
  register int spl;
  double bx,by;
  int count = 0;
  bx = x_base+col*x_grid;
  by = y_base+row*y_grid;
  spl = segnum+1;
  if(spl == nb)
    spl = 0;
// intersection with bottom wall
  if(segments_intersect(bx,by,bx+x_grid,by,xb[segnum],yb[segnum],xb[spl],
                                               yb[spl]))
    count++;
// intersection with right wall
  if(segments_intersect(bx+x_grid,by,bx+x_grid,by+y_grid,xb[segnum],
                                               yb[segnum],xb[spl],yb[spl]))
    count++;
// intersection with top wall
  if(segments_intersect(bx+x_grid,by+y_grid,bx,by+y_grid,xb[segnum],
                                               yb[segnum],xb[spl],yb[spl]))
    count++;
// intersection with left wall
  if(segments_intersect(bx,by+y_grid,bx,by,xb[segnum],yb[segnum],xb[spl],
                                               yb[spl]))
    count++;
  return count;
}
                                       /* Point Inside Block Function */
                                       /*******************************/

int point_inside_block(double x,double y,int row,int col)
{
  x -= (x_base+x_grid*col);
  y -= (y_base+y_grid*row);
  return( (x>=0) && (x<=x_grid) && (y>=0) && (y<=y_grid));
}

int sign(double x)
{
  if(x<0)
    return (-1);
  if(x>0)
    return 1;
  return 0;
}

int compare_xs(double * x1p, double * x2p)
```

```
{
  return sign((*x1p) - (*x2p));
}

                                        /* Point Inside Boundary Function */
                                        /*********************************/
int point_inside_boundary(double x,double y,double xb[],double yb[],int nb)
{
  double xcross[MAX_CROSS];
  double xc;
  double cross_sign;
  register int i, im1,ip1;
  int kc = 0;
  for(i=0; i<nb; i++)
  {
    im1 = i-1;
    if(im1 < 0)
      im1 = nb-1;
    if(yb[i] == yb[im1])
    {
      if(yb[i] == y)
      {
        xcross[kc++] = xb[i];
        xcross[kc++] = xb[i];
      }
    }
    else
    if(yb[i] == y)
    {
      ip1 = i+1;
      if(ip1 == nb)
        ip1 = 0;
      cross_sign = (yb[i]-yb[ip1])*(yb[i]-yb[im1]);
      xcross[kc++] = xb[i];
      if(cross_sign > 0)
        xcross[kc++] = xb[i];
    }
    else
    if(((yb[i]-y)*(yb[im1]-y)) < 0)
    {
      xc = (xb[im1]+(xb[i]-xb[im1])/(yb[i]-yb[im1])*(y-yb[im1]));
      xcross[kc++] = xc;
    }
  }
  qsort(xcross,kc,sizeof(double),compare_xs);
#if _DEBUG_
{
  register int i;
  printf("x = %lf, y = %lf  crosses at %d points\n",x,y,kc);
  for(i=0; i<kc; i++)
    printf("%2d: %lf\n",i,xcross[i]);
}
#endif
  for(i=0; i<(kc/2); i++)
  {
    if((x >= xcross[2*i]) && (x <= xcross[2*i+1]))
      return TRUE;
  }
  return FALSE;
}

                                  /* Block-Boundary Intersection Function */
                                  /****************************************/
double block_area( int row, int col, double xb[], double yb[], int nb)
{
        double bx = x_base+x_grid*col;
        double by = y_base+y_grid*row;
        register int i,j,count1,count3,count;
```

```
        for (count1 = 0,i=0; i<2; i++)
                for (j=0; j<2; j++)
                {
                if(point_inside_boundary(bx+x_grid*(i^j),by+y_grid*j,xb,yb,nb))
                                count1++;
                }
        if (count1 == 0)        return 0;
        elseif(count1 == 4)
                for (count=i=0; i<nb; i++)
                        if (point_inside_block(xb[i],yb[i],row,col))
                                count++;
                for (count3=i=0; i<nb; i++)
                        count3+=block_boundary_segm_intersect(row,col,i,xb,yb,nb);
                if ( (count1 >= 2 && count3 >= 1) || ( count1>=1 && count3 >=2))
                return 1;
                 else if ((count1 < 2 && count3 < 1) || ( count1<1 && count3 <2))
                 return 2;
                return 0;
}
int compare_x(double * x1p, double * x2p)
{
  return (int)((*x1p) - (*x2p));
}
                                /* Block Inside Boundary Function */
                                /**********************************/
int block_inside_boundary(double x1,double x2,double block_y,int nbpnts,
                                                double xb[],double yb[])
{
  register int i,iminus1;
  double xcross[MAX_CROSS_POINTS],x[MAX_CROSS_POINTS];
  int kc = 0,j=0,nn;
  for(i=0; i<nbpnts; i++)
  {
    iminus1 = i-1;
    if(iminus1 < 0)             iminus1 = nbpnts-1;
    if(((yb[i]-block_y)*(yb[iminus1]-block_y)) <= 0)
      if(yb[iminus1] != yb[i])
      {
         xcross[kc++] = (xb[i]+(block_y-yb[i])*(xb[iminus1]-xb[i]) /
                                            (yb[iminus1]-yb[i]));
      }
  }
  qsort(xcross,kc,sizeof(double),compare_x);
  /* delete equal number */
  for (i=0; i<kc; i++)
  {
        nn = i+1;
        if (i==kc-1)            nn=0;
        if (xcross[i] == xcross[nn])
        {
                x[j]=xcross[i];
                i = i+1;
        }
        else            x[j] = xcross[i];
         j++;
   }
  if (j == 1)    return FALSE;
  if ( even(j))
  {
  for(i=0; i<(j/2); i++)
  {
    if((x1 >= x[2*i]) && (x1 <= x[2*i+1]))     /* block is inside */
      return TRUE;
  }
  }
  else
  for(i=0; i<j; i++)
  {
```

```
    if((x1 >= x[i]) && (x1 <= x[i+1]))          /* block is inside */
      return TRUE;
  }
  return FALSE;
}

                              /* Point Inside Block Function */
                              /*******************************/
int inside_block ( double x1,double x2,double y1,double y2,double xp,double yp)
{
    int inside =0;
        if ( xp>= x1 && xp <= x2 && yp >= y1 && yp <= y2)     inside =1;
        return inside;
}
int even ( int number)
{
        int remainder;
        remainder = number % 2;
        if(remainder == 0)
        return 1;
        return 0;
}
                                      /* Average Thickness Function */
                                      /******************************/
void avgthick_inside_boundary(  double y_base,        double x_base, double y_grid,
 double x_grid, int ny, int nx,double xb[],double yb[],int nb,
double thick[MAXROW][MAXCOL],double *avg_thick)
{
        register int i,j;
        int block,block_with=0,block_in =0;
        double x1,y1;
        double cum_thick,cum_thick_with=0,cum_thick_in=0;
  for(i=0; i<ny; i++)
  {
    for(j=0; j<nx; j++)
    {
       if(block_area(i,j,xb,yb,nb)==1)
       {
                       block_with++;
                       cum_thick_with = cum_thick_with + thick[i][j];
  .    }
    }
   }
   for(i=0; i<ny-1; i++)
  {
         y1 = y_base+i*y_grid;
    for(j=0; j<nx-1; j++)
    {
        x1 = x_base+j*x_grid;
       if (block_inside_boundary(x1,x1+x_grid,y1,nb,xb,yb))
       {
       if(block_area(i,j,xb,yb,nb)==0)
       {
                       block_in++;
                       cum_thick_in = cum_thick_in + thick[i][j];
       }
       }
   }
   }
   block = block_in + block_with;
   cum_thick = cum_thick_in + cum_thick_with;
   (*avg_thick) = cum_thick/block;
}
                                      /* Tonnage Function */
                                      /********************/
void tonnage_inside_boundary(double mine_area, double density,
                              double avg_thick,double *total_ton)
{
```

```
   (*total_ton) = (mine_area*avg_thick*density)/(12*2000);
}
                                              /* Tonnage Recovery Function */
                                             /*******************************/
voidretonnage_inside_boundary(doubleavg_recov,doubletotal_ton,double*total_reton)
{
   (*total_reton) = (avg_recov*total_ton);
}
                                             /* Area Function */
                                             /*****************/
double area (int n,double x[],double y[],double *areap)
{
    int i;
    (*areap) = x[n-1]*y[0]-x[0]*y[n-1];
        for (i=1; i<n; ++i)
                (*areap) = (*areap) + x[i-1]*y[i]-x[i]*y[i-1];
                (*areap) = (*areap)/2;
                return *areap;
}
                        /* Nearest Search Grid Function */
                         /**********************************/
void grid_data (double x[MAXPT],double y[MAXPT],double z[MAXPT],intn,intnp,
double xo, double yo,double dx,doubledy,intnx,intny,doublematrix[MAXROW][MAXCOL])
{
        double  x1,x2,s1,s2;
        register int i,j,k,l,ic;
        double dist[MAXPT],d;
   for ( i = 0; i< ny; ++i)
       {
        x2 = yo + dy*(i);
        for ( j =0; j < nx; ++j)
        {
         x1 = xo + dx*(j);
                /* Calculate distance betw.current
                   grid point and all data points */
                /* -------------------------------*/
        for (k = 1; k< n; ++k)
         dist[k] =pow((x1-x[k]),2)+pow((x2-y[k]),2);
                /* Find the np nearest data
                   points and calculate sums */
                /* --------------------------*/
         s1 = 0.0;
         s2 = 0.0;
         for ( k=1; k<=np; ++k)
         {
         ic = 1;
         for ( l = 2; l< n;++l)
         {
         if (dist[ic] > dist[l])
          ic = l;
         }
          if (dist[ic]== 0.00)    goto divide_by_zero;
          d = sqrt (dist[ic]);
          s1 = s1 + z[ic]/d;
          s2 = s2 + 1.0/d;
          dist[ic] = 1.0e+30;
          }
                /* calculate grid point and store in matrix */
                /* --------------------------------------- */

         matrix[i][j]= s1/s2;
         goto keep;
           divide_by_zero: matrix[i][j] = z[ic];
          keep: ;
            }
          }
}
```

```
                              /* Quadrant Grid Function */
                              /**************************/
void gridq_data(doublex[MAXPNT],doubley[MAXPNT],doublez[MAXPNT],intn,intnp,
double xo,double yo,double dx,double dy,intnx,intny,doublematrix[MAXROW][MAXCOL])
{
        double  x1,x2,s1,s2;
        double m1,m2,q1,q2,p1,p2,r1,r2;
        register int i,j,k,l;
    int m,q,p,r,ic;
        double d1,d2,d3,d4;
        double dist1[MAXPT],dist2[MAXPT],dist3[MAXPT],dist4[MAXPT];
        double xru[MAXPT],xrd[MAXPT],xlu[MAXPT],xld[MAXPT],yru[MAXPT],
                        yrd[MAXPT],ylu[MAXPT],yld[MAXPT];

        /*static double matrix[50][50];*/

    for ( i = 0; i< ny; i++)
    {
        x2 = yo + dy*(i);
        for ( j =0; j < nx; j++)
        {
         x1 = xo + dx*(j);
        /* Determine the points for each quadrant */
        /* ------------------------------------- */
     m=0; q=0; p=0;r=0;
     for ( k=0; k<n; ++k)
     {
        if ( x[k]-x1 == 0 && y[k]-x2 == 0)
                goto same_value;
        if ( (x[k]-x1) >0  && (y[k]-x2)>=0 )
        {
                xru[m] = x[k];yru[m] = y[k];m++;
        }
        if ( (x[k]-x1) >= 0 && (y[k]-x2) < 0)
        {
                xrd[p] = x[k];yrd[p] = y[k];p++;
        }
        if ( (x[k]-x1) <= 0 && (y[k]-x2) > 0)
        {
                xlu[q] = x[k];ylu[q] = y[k];q++;
        }
        if ( (x[k]-x1) < 0 && (y[k]-x2) <= 0)
        {
                xld[r] = x[k];yld[r] = y[k];r++;
        }
      }
        if ( m == 0)
                {
                        m1 = 0; m2 = 0;
                }
                else
                {
           /* calculate distance betw.current grid point and all data points */
                                                     /* in the up right quadrant */
          /* ------------------------------------------------------------- */
        for (k = 0; k< m; ++k)
        {
         dist1[k] =pow((x1-xru[k]),2)+pow((x2-yru[k]),2);
        }
                /* find the np nearest data points and calculate sums */
                /* ------------------------------------------------ */
         m1 = 0.0; m2 = 0.0;
         for ( k=0; k<np; ++k)
         {
         ic = 0;
         for ( l = 1; l< m;++l)
         {
         if (dist1[ic] > dist1[l])      ic = l;
```

```
      }
     d1 = sqrt (dist1[ic]);m1 += z[ic]/d1; m2 +=  1.0/d1; dist1[ic] = 1.0e+30;
            }
 }
      if ( p == 0)
 {
            p1 = 0;p2 = 0;
 }
 else
 {
    /* calculate distance betw.current grid point and all data points */
                                                    /* in the down right quadrant */
    /* --------------------------------------------------------------- */
     for (k = 0; k< p; ++k)
    {
     dist2[k] =pow((x1-xrd[k]),2)+pow((x2-yrd[k]),2);
    }
            /* find the np nearest data points and calculate sums */
            /* ------------------------------------------------ */
     p1 = 0.0; p2 = 0.0;
    for ( k=0; k<np; ++k)
     {
     ic = 0;
     for ( l = 1; l< p;++l)
     {
     if (dist2[ic] > dist2[l])       ic = l;
      }
      d2 = sqrt (dist2[ic]); p1 += z[ic]/d2; p2 += 1.0/d2;dist2[ic] = 1.0e+30;
      }
 }
       if ( q == 0)
 {
            q1 = 0;q2 = 0;
 }
 else
 {
    /* calculate distance betw.current grid point and all data points */
                                                    /* in the up left quadrant */
    /* --------------------------------------------------------------- */
     for (k = 0; k< q; ++k)
    {
     dist3[k] =pow((x1-xlu[k]),2)+pow((x2-ylu[k]),2);
    }
            /* find the np nearest data points and calculate sums */
            /* ------------------------------------------------ */
     q1 = 0.0; q2 = 0.0;
     for ( k=0; k<np; ++k)
     {
     ic = 0;
     for ( l = 1; l< q;++l)
     {
     if (dist3[ic] > dist3[l])       ic = l;
      }
      d3 = sqrt (dist3[ic]); q1 += z[ic]/d3;q2 +=  1.0/d3;dist3[ic] = 1.0e+30;
      }
}
   if ( r == 0)
  {
            r1 = 0;r2 = 0;
  }
  else
  {
  /* calculate distance betw.current grid point and all data points */
                          /* in the down left quadrant */
    /* --------------------------------------------------------------- */
     for (k = 0; k< r; ++k)
    {
    dist4[k] =pow((x1-xld[k]),2)+pow((x2-yld[k]),2);
```

```
        }
                /* find the np nearest data points and calculate sums */
                /* ------------------------------------------------- */
         r1 = 0.0; r2 = 0.0;
         for ( k=0; k<np; ++k)
         {
         ic = 0;
         for ( l = 1; l< r;++l)
         {
         if (dist4[ic] > dist4[l])       ic = l;
          }
          d4 = sqrt (dist4[ic]); r1 += z[ic]/d4; r2 += 1.0/d4;dist4[ic] = 1.0e+30;
          }
    }
                /* calculate grid point and store in matrix */
                /* --------------------------------------- */
        matrix[i][j]= (m1+p1+q1+r1)/(m2+p2+q2+r2);
          goto jump;
          same_value: matrix[i][j] = z[k];
          jump:;
                }
               }
}
                                  /* Octant Grid Function */
                                  /************************/
voidgrido_data(doublex[MAXPNT],doubley[MAXPNT],doublez[MAXPNT],intn,intnp,
doublexo,doubleyo,doubledx,doubledy,intnx,intny,doublematrix[MAXROW][MAXCOL])
{
    double  x1,x2;
    double m1,m2,q1,q2,p1,p2,r1,r2,a1,a2,b1,b2,c1,c2,h1,h2;
    register int i,j,k,l;
    int m,p,q,r,l11,l12,l13,l14,ic;
    double dist1[MAXPT],dist2[MAXPT], dist3[MAXPT],dist4[MAXPT],
    dist5[MAXPT],dist6[MAXPT],dist7[MAXPT],dist8[MAXPT];
    double d1,d2,d3,d4,d5,d6,d7,d8;
    double xru[MAXPT],xrd[MAXPT],xlu[MAXPT],xld[MAXPT],yru[MAXPT],
    yrd[MAXPT],ylu[MAXPT],yld[MAXPT];
    double xru1[MAXPT],yru1[MAXPT],xrd1[MAXPT],yrd1[MAXPT],xlu1[MAXPT],
            ylu1[MAXPT],xld1[MAXPT],yld1[MAXPT];
    double dd1,dd2,dd3,dd4;
    for ( i = 0; i< ny; i++)
    {
        x2 = yo + dy*(i);
        for ( j = 0 ; j < nx; j++)
        {
                x1 = xo + dx*(j);
        /* Determine the points for each quadrant */
        /* ------------------------------------- */
                l11 = 0;l12 = 0;l13 = 0;l14 = 0;m = 0;p = 0;q = 0;r = 0;d1 = 0;
            d2 = 0;d3 = 0;d4 = 0;d5 = 0;d6 = 0;d7 = 0;d8 = 0;
                for ( k=0; k<n; ++k)
                {
                        if ( x[k]-x1 == 0 && y[k]-x2 == 0)     goto same_value;
                        if ( (x[k]-x1) >0  && (y[k]-x2)>=0 )
                        {
                                dd1 = (y[k]-x2) / (x[k]-x1);
                                if (dd1 >= 0 && dd1 < 1)
                        {
                                xru1[l11] = x[k];yru1[l11] = y[k];l11++;
                        }
                        else
                        {
                                xru[m] = x[k];yru[m] = y[k];m++;
                        }
                        }
                        if ( (x[k]-x1) >= 0 && (y[k]-x2) < 0)
                        {
                        dd2 = (x[k]-x1) / (y[k]-x2);
```

```
                        if (dd2 >= -1 && dd2 < 0)
                {
                        xrd1[ll2]=x[k];yrd1[ll2] = y[k];ll2++;
                        }
                else
                {
                        xrd[p] = x[k];yrd[p] = y[k];p++;
                }
            }
                if ( (x[k]-x1) <= 0 && (y[k]-x2) > 0)
                {
                dd3 = (x[k]-x1) / (y[k]-x2);
                        if (dd3 >= -1 && dd3 < 0)
                {
                        xlu1[ll3]=x[k];ylu1[ll3] = y[k];ll3++;
              }
                else
                {
                        xlu[q] = x[k];ylu[q] = y[k];q++;
              }
          }
                if ( (x[k]-x1) < 0 && (y[k]-x2) <= 0)
                {
                dd4 = (y[k]-x2) / (x[k]-x1);f (dd4 >= 0 && dd4 < 1)
                {
                        xld1[ll4] = x[k];yld1[ll4] = y[k];ll4++;
              }
                else
                {
                        xld[r] = x[k];yld[r] = y[k];r++;
              }
          }
      }
if ( m == 0 || m == 1)
{
     m1 = 0;m2 = 0;
}
else
{
/* calculate distance betw.current grid point and all data points */
                    /* in the up right octant (45-90) */
/* ------------------------------------------------------------- */
        for (k = 0; k< m; ++k)
        {
                dist1[k] =pow((x1-xru[k]),2)+pow((x2-yru[k]),2);
        }
/* find the np nearest data points and calculate sums */
/* ------------------------------------------------- */
        m1 = 0.0;m2 = 0.0;
        for ( k=0; k<np; ++k)
        {
                ic = 0;
                for ( i = 1; l< m;++l)
                {
                        if (dist1[ic] > dist1[l])       ic = l;
                }
d1 = sqrt (dist1[ic]);m1 += z[ic]/d1;m2 +=  1.0/d1;dist1[ic] =1.0e+30;
        }
}
if ( ll1 == 0 || ll1 == 1)
{
     a1 = 0;a2 = 0;
}
else
{
```

```
/* calculate distance betw.current grid point and all data points */
                    /* in the up most right octant(0-45) */
/* -------------------------------------------------------------- */
        for (k = 0; k< ll1; ++k)
        {
                dist2[k] =pow((x1-xru1[k]),2)+pow((x2-yru1[k]),2);
        }
/* find the np nearest data points and calculate sums */
/* ------------------------------------------------- */
        a1 = 0.0;a2 = 0.0;
        for ( k=0; k<np; ++k)
        {
                ic = 0;
                for ( l = 1; l< ll1;++l)
                {
                        if (dist2[ic] > dist2[l])       ic = l;
                }
d5 = sqrt (dist2[ic]);a1 += z[ic]/d5;a2 +=  1.0/d5;dist2[ic] = 1.0e+30;
        }
}
if ( ll2 == 0 || ll2 == 1)
{
     b1 = 0;b2 = 0;
}
else
{
/* calculate distance betw.current grid point and all data points */
                    /* in the down most right octant(315-360) */
/* -------------------------------------------------------------- */
        for (k = 0; k< ll2; ++k)
        {
                dist3[k] =pow((x1-xrd1[k]),2)+pow((x2-yrd1[k]),2);
        }

/* find the np nearest data points and calculate sums */
/* ------------------------------------------------- */

        b1 = 0.0;b2 = 0.0;
        for ( k=0; k<np; ++k)
        {
                ic = 0;
                for ( l = 1; l< ll2;++l)
                {
                        if (dist3[ic] > dist3[l])       ic = l;
                }
d6 = sqrt (dist3[ic]);b1 += z[ic]/d6;b2 +=  1.0/d6;dist3[ic] = 1.0e+30;
        }
}
if ( p == 0 || p == 1)
{
    p1 = 0;p2 = 0;
}
else    {
/* calculate distance betw.current grid point and all data points */
                    /* in the down right (270-315) */
/* -------------------------------------------------------------- */
        for (k = 0; k< p; ++k)
        {
                dist4[k] =pow((x1-xrd[k]),2)+pow((x2-yrd[k]),2);
        }
/* find the np nearest data points and calculate sums */
/* ------------------------------------------------- */
        p1 = 0.0;p2 = 0.0;
        for ( k=0; k<np; ++k)
        {
                ic = 0;
                for ( l = 1; l< p;++l)
                {
```

```
                        if (dist4[ic] > dist4[l])       ic = l;
                }
d2 = sqrt (dist4[ic]);p1 += z[ic]/d2;p2 += 1.0/d2;dist4[ic] = 1.0e+30;
        }
  }
   if ( q == 0 || q == 1)
  {
      q1 = 0;q2 = 0;
  }
  else
  {
  /* calculate distance betw.current grid point and all data points */
                        /* in the up left octant(90-135) */
  /* ------------------------------------------------------------ */
        for (k = 0; k< q; ++k)
        {
                dist5[k] =pow((x1-xlu[k]),2)+pow((x2-ylu[k]),2);
        }
  /* find the np nearest data points and calculate sums */
  /* ---------------------------------------------- */
        q1 = 0.0;q2 = 0.0;
        for ( k=0; k<np; ++k)
        {
                ic = 0;
                for ( l = 1; l< q;++l)
                {
                        if (dist5[ic] > dist5[l])       ic = l;
                }
d3 = sqrt (dist5[ic]);q1 += z[ic]/d3;q2 +=  1.0/d3;dist5[ic] = 1.0e+30;
        }
  }
  if ( r == 0 || r == 1)
  {
        r1 = 0;r2 = 0;
  }
  else
  {
  /* calculate distance betw.current grid point and all data points */
                        /* in the down left octant(225-270) */
  /* ------------------------------------------------------------ */
        for (k = 0; k< r; ++k)
        {
                dist6[k] =pow((x1-xld[k]),2)+pow((x2-yld[k]),2);
        }
  /* find the np nearest data points and calculate sums */
  /* ---------------------------------------------- */
        r1 = 0.0;r2 = 0.0;
        for ( k=0; k<np; ++k)
        {
                ic = 0;
                for ( l = 1; l< r;++l)
                {
                        if (dist6[ic] > dist6[l])       ic = l;
                }
d4 = sqrt (dist6[ic]);1 += z[ic]/d4;r2 += 1.0/d4;dist6[ic] = 1.0e+30;
        }
  }
  if ( ll3 == 0 || ll3 == 1)
  {
       c1 = 0;c2 = 0;
  }
  else
  {
  /* calculate distance betw.current grid point and all data points */
                        /* in the up most left octant (1135-180) */
  /* ------------------------------------------------------------ */
        for (k = 0; k< ll3; ++k)
        {
```

```
                    dist7[k] =pow((x1-xlul[k]),2)+pow((x2-ylul[k]),2);
              }
       /* find the np nearest data points and calculate sums */
       /* -------------------------------------------------- */
              c1 = 0.0;c2 = 0.0;
              for ( k=0; k<np; ++k)
              {
                    ic = 0;
                    for ( l = 1; l< l13;++l)
                    {
                           if (dist7[ic] > dist7[l])       ic = l;
                    }
       d7 = sqrt (dist7[ic]);c1 += z[ic]/d7;c2 +=  1.0/d7;dist7[ic] = 1.0e+30;
              }
       }
       if ( l14 == 0 || l14 == 1)
       {
            h1 = 0;h2 = 0;
       }
       else
       {
       /* calculate distance betw.current grid point and all data points */
                          /* in the down mostleft octant (225-270) */
       /* ---------------------------------------------------------- */
              for (k = 0; k< l14; ++k)
              {
                    dist8[k] =pow((x1-xldl[k]),2)+pow((x2-yldl[k]),2);
              }
       /* find the np nearest data points and calculate sums */
       /* -------------------------------------------------- */
              h1 = 0.0;h2 = 0.0;
              for ( k=0; k<np; ++k)
              {
                    ic = 0;
                    for ( l = 1; l< l14;++l)
                    {
                           if (dist8[ic] > dist8[l])       ic = l;
                    }
     d8 = sqrt (dist8[ic]);h1 += z[ic]/d8;h2 +=  1.0/d8;dist8[ic] = 1.0e+30;
              }
       }
       /* calculate grid point and store in matrix */
       /* ---------------------------------------- */
    matrix[i][j]= (m1+p1+q1+r1+a1+b1+c1+h1)/(m2+p2+q2+r2+a2+b2+c2+h2);
      goto jump;
      same_value: matrix[i][j] = z[k];
      jump:;
        }
       }
}

                                  /* Average block thickness function */
                                  /*************************************/

void block_thickness (  double matrix[MAXROW][MAXCOL],int ny,int nx,
                        double avg_block[MAXROW][MAXCOL])
{
    int i, j;
       for (i=0; i <ny-1;i++)
              for (j=0; j < nx-1; j++)
              {
avg_block[i][j] =(matrix[i][j]+matrix[i][j+1]+matrix[i+1][j+1]+matrix[i+1][j])/4;
       }
}
```

```
                                /* Average block range function */
                                /********************************/
void block_range (      double range1, doublerange2,doubleblock_ton[MAXROW][MAXCOL],
double real_bl_ton[MAXROW][MAXCOL],int ny,int nx,doubleavg_block[MAXROW][MAXCOL],
double * accum_ton, int * block, double * accum_reton)
{
    int i, j;
    (*block) = 0;(*accum_ton) = 0;(*accum_reton) = 0;
        for (i=0; i <ny-1;i++)
                for (j=0; j <nx-1; j++)
                {
                if ( range1 <= avg_block[i][j] && avg_block[i][j] < range2 )
                {
                        (*block)++;(*accum_ton) = (*accum_ton) + block_ton[i][j];
                        (*accum_reton) = (*accum_reton) + real_bl_ton[i][j];
                }
                }
}
                        /* Tons inside grid function */
                        /*****************************/
void tons_inside_grid ( int ny, int nx, double block_ton[MAXROW][MAXCOL],
double real_bl_ton[MAXROW][MAXCOL],double * total_tons, double * total_retons)
{
        int i, j;(*total_tons) = 0;(*total_retons) = 0;
        for (i=0; i <ny-1;++i)
                for (j=0; j <nx-1; ++j)
                {
                        (*total_tons) = (*total_tons) + block_ton[i][j];
                        (*total_retons) = (*total_retons) + real_bl_ton[i][j];
                }
}
                        /* Average interior percent function */
                        /*************************************/
void avg_inter_percent(int ny, int nx, double avg_block[MAXROW][MAXCOL],
                        double * avg_inter)
{
        int i, j;double total_average;total_average = 0;
        for (i=0; i <ny-1;++i)
                for (j=0; j <nx-1; ++j)
                {
                        total_average = total_average + avg_block[i][j];
                }
            (*avg_inter) = total_average/((ny-1)*(nx-1));
}
                        /* Ton inside a block function */
                        /*******************************/
voidton_inside_block(intny,intnx,doubledx,doubledy,
        doubleavg_bl_thick[MAXROW][MAXCOL],double  block_ton[MAXROW][MAXCOL])
{
        int i,j;double block_density=100;
        for (i=0; i < ny-1; i++)
                for ( j=0; j < nx-1; j++)
                {
          block_ton[i][j]= (dx*dy*avg_bl_thick[i][j]*block_density)/(12*2000);
                }
}
                /************************************/
                        /* Real Ton inside a block function */
                        /************************************/
void reton_inside_block ( int ny, int nx, double  block_ton[MAXROW][MAXCOL],
double avg_bl_recov[MAXROW][MAXCOL],double real_bl_ton[MAXROW][MAXCOL])
{
        int i,j;
        for (i=0; i < ny-1; i++)
                for ( j=0; j < nx-1; j++)
                {
        real_bl_ton[i][j]= (block_ton[i][j]*avg_bl_recov[i][j]);
                }
```

```
}
                        /* Pressure inside a block function */
                        /*************************************/
void press_inside_block( int ny, int nx, double avg_bl_depth[MAXROW][MAXCOL],
double burd_density, double avg_bl_pressure[MAXROW][MAXCOL])
{
        int i,j;
        for (i=0; i < ny-1; i++)
                for ( j=0; j < nx-1; j++)
                {
        avg_bl_pressure[i][j]= (avg_bl_depth[i][j]*burd_density)/(144);
                }
}
                                /* Recovery_percentage Function */
                                /********************************/
void Recovery_percentage (double matrix1[MAXROW][MAXCOL], matrix2[MAXROW][MAXCOL], int ny,
                        int nx, matrix[MAXROW][MAXCOL]);
{
        double crit_dis;
        int i,j;
        for ( i=0; i< ny; i++)
        {
                for ( j=0; j<nx; j++)
                {
                if (matrix1[i][j] < 12) matrix[i][j] = 0;
                else if (matrix1[i][j] > 110) matrix[i][j] = 1;
                else
                {
                crit_dist = 110 - .65*matrix2[i][j];
                if ( matrix1[i][j] > crit_dist) matrix[i][j] = 1;
                else matrix[i][j] = matrix1[i][j]/crit_dist;
                {
                }
        }

                                /* Display matrix function */
                                /***************************/
void display_matrix ( double matrix[MAXROW][MAXCOL],int ny,int nx)
{
     int i,j;
     for ( i = 0; i< ny; i++)
          {
                for (j = 0; j < nx; j++)
                fprintf (out,"%6.2lf\t",matrix[i][j]);
            fprintf (out,"\n");
          }
        fprintf (out,"\n");
}
/*****************************************************************************/
typedef struct boundary
{
  int nbp;double * xb;double * yb;
} BOUNDARY;
BOUNDARY bound_list[MAXPNT];
int bound_count = 0;
void main ()
{
        /* Declaration of variables */
        /* ------------------------ */
    int nx,ny,n,np,ns,nb,na;
        int  block,answer;
    register int i,j;
    static double thickness[MAXROW][MAXCOL],avg_bl_thick[MAXROW][MAXCOL];
    static double inner[MAXROW][MAXCOL],sandstone[MAXROW][MAXCOL];
    static double recovery[MAXROW][MAXCOL],avg_bl_recov[MAXROW][MAXCOL];
    double x_coord[MAXPT],y_coord[MAXPT],over_depth[MAXPT],dep_thick[MAXPT];
        double recov_per[MAXPT], inburd[MAXPT], sand[MAXPT];
        double btus[MAXPT],sulphur[MAXPT],ashs[MAXPT];
```

```
        double sulf_one[15],sulf_two[15],btu_one[15],btu_two[15];
        double ash_one[15],ash_two[15];
        char in_name[20], out_name[20],in_sulf[20],in_btu[20],in_ash[20],
        in_bound[25], in_recovery[25];
        static double block_ton[MAXROW][MAXCOL],avg_bl_pressure[MAXROW][MAXCOL];
        static double real_bl_ton[MAXROW][MAXCOL];
        static double sulfer[MAXROW][MAXCOL],avg_bl_sulf[MAXROW][MAXCOL];
        static double burden[MAXROW][MAXCOL],avg_bl_depth[MAXROW][MAXCOL];
        static double btunit[MAXROW][MAXCOL],avg_bl_btu[MAXROW][MAXCOL];
        static double ashunit[MAXROW][MAXCOL],avg_bl_ash[MAXROW][MAXCOL];
    double avg_inter_sulf,avg_inter_btu,avg_inter_ash;accum_ton,total_tons;
    double accum_reton,total_retons;tons_in_bound;double retons_in_bound;
    double avg_thick;double avg_recov,avg_depth,mine_bound_area,x,y;
        int choice;
     printf ( " which gridding method do you want to use?\n");
     printf ("\t       1 for closest points gridding method\n");
     printf ("\t       2 for quadrant gridding method\n");
     printf ("\t       3 for octant gridding method\n");
     printf (" Enter a number: ");
     scanf ("%d", &choice);
            /* Get the number of nearest points */
              /* -------------------------------- */
    printf ("How many points do you want to estimate each grid point:");
        scanf ("%d",&np);
              /* Get x and y grid spacing */
              /* ------------------------ */
        printf (" Enter  x grid spacing : ");  scanf ("%lf", &x_grid);
        printf (" Enter  y grid spacing : ");  scanf ("%lf", &y_grid);
              /* Get number of x and y grid points */
              /* --------------------------------- */
        printf("Enter the number of x grid points : ");       scanf("%d", &nx);
    printf("Enter the number of y grid points : ");   scanf("%d", &ny);
              /* Get starting grid coordinates */
              /* ----------------------------- */
        printf("enter starting x grid coordinate: ");  scanf("%lf", &x_base);
        printf("enter starting y grid coordinate: ");  scanf("%lf", &y_base);
printf("Do you want to calculate the percentage of sulphur,ash,btus in\n");
printf("each block? if yes enter 1.and don't forget to include their data\n");
    printf(" in the input data file:");    scanf("%d", &answer);
    gets (in_name);
        /* Open files to read and write data */
        /* -------------------------------- */
        while(TRUE)
    {
              printf ("enter name of data file : ");        gets (in_name);
          if(in_name[0] == 0)          exit(-1);
          if ( (in = fopen (in_name, "r")) == NULL )
        printf("Cannot open '%s' : %s\n",in_name,strerror(errno));
      else        break;
        }
   while(TRUE)
    {
          printf ("enter name of mining boundary data file : ");  gets (in_bound);
      if(in_bound[0] == 0)         exit(-1);
          if ( (bound = fopen (in_bound, "r")) == NULL )
        printf("Cannot open '%s' : %s\n",in_bound,strerror(errno));
      else        break;
        }
        while (TRUE)
    {
          printf ("enter name of output file : ");       gets (out_name);
      if(out_name[0] == 0)         exit(-1);
          if ( (out = fopen (out_name, "w")) == NULL )
        printf("Cannot open '%s' : %s\n",out_name,strerror(errno));
      else        break;
        }
```

```
        fprintf(out, "\t\t\t\t\t******************** \n ");
        fprintf(out, "\t\t\t\t\tTONNAGE INFORMATION \n ");
        fprintf(out, "\t\t\t\t\t******************** \n ");
        fprintf(out,"\n");
    // first, count number of boundaries
        while(!feof(bound))
        {
        fscanf(bound,"%lg %lg ",&x,&y);
        if(x == -1)             bound_count++;
        else            bound_list[bound_count].nbp++;
        }
        for(i=0; i<bound_count; i++)
        {
        bound_list[i].xb = (double *) calloc(bound_list[i].nbp,sizeof(double));
        bound_list[i].yb = (double *) calloc(bound_list[i].nbp,sizeof(double));
        if(bound_list[i].xb == NULL || bound_list[i].yb == NULL)
        {
         printf("Could not allocate %d bytes for bound_list[%d] - Aborting\n",
                bound_list[i].nbp*sizeof(double),i);   exit(-1);
        }
         }
   // now, rewind file and read all the points
        rewind(bound);
        for(i=0; i<bound_count; i++)
        {
   // read points for boundary i
        for(j=0; j<bound_list[i].nbp; j++)
         fscanf(bound,"%lg %lg ",&bound_list[i].xb[j],&bound_list[i].yb[j]);
   // read remaining -1, -1
        fscanf(bound,"%lg %lg ",&x,&y);
        }
        fclose(bound);
fprintf(out, " No. of points to estimate each grid = %i\n",np);fprintf(out,"\n");
fprintf(out, " x grid spacing = %9.3lf\n",x_grid);     fprintf(out,"\n");
fprintf(out, " y grid spacing = %9.3lf\n",y_grid);     fprintf(out,"\n");
fprintf(out, " Number of x grid points = %i\n",nx);    fprintf(out,"\n");
fprintf(out, " Number of y grid points = %i\n", ny);   fprintf(out,"\n");
fprintf(out, " Starting x grid coordinate = %4.2lf\n",x_base);fprintf(out,"\n");
fprintf(out, " Starting y grid coordinate = %4.2lf\n",y_base);frintf(out,"\n");
fprintf(out,"\t\t\tMining Boundary Co-ordinates \n"); fprintf(out,"\n");
fprintf(out,"There is %d boundarie(s):\n", bound_count);      fprintf(out,"\n");
        for(i=0; i<bound_count; i++)
        {
        fprintf(out,"Boundary #%d\n",i+1);      fprintf(out,"\n");
fprintf(out,"\t\t\t\tEAST\t\t\tSOUTH\n");fprintf(out,"\t\t\t\tCOORDS\t\t\tCOORDS\n");
        fprintf(out,"\n");
        for(j=0; j<bound_list[i].nbp; j++)
        fprintf(out,"  %3d:\t\t %8lg\t\t %8lg\n",j+1,bound_list[i].xb[j],
                                bound_list[i].yb[j]);  printf("\n");
        }       fprintf(out,"\n");
        if( answer == 1)
        {
                fprintf(out, "\t\t\t\t\t\t\tINPUT DATA \n ");  fprintf(out,"\n");
        fprintf(out,"\tEAST\t\t\tNORTH\t\t\tOVERBURDEN\t\DEPOSIT\t\t\BTUs\t"
                                       "\t\SULPHUR\t\tASH\n");
fprintf(out,"\tCOORDS\t\t\tCOORDS\t\t\tDEPTH\t\t\THICKNESS\n");fprintf(out,"\n");
        n =0;
while((fscanf(in,"%lf%lf%lf%lf%lf%lf%lf%lf%lf",&x_coord[n],&y_coord[n],&over_depth[n],
&dep_thick[n],&btus[n],&sulphur[n],&ashs[n],&inburd[n],sand[n]) != EOF)
            {
fprintf(out,"%10.2lf\t\t%10.2lf\t\t%10.2lf\t%10.2lf\t%10.2lf\t%"10.2lf\t%10.2lf\n",
x_coord[n],y_coord[n],over_depth[n],dep_thick[n],btus[n],sulphur[n],
ashs[n]/*,recov_per[n]*/);
                n++;
                        }       fprintf(out,"\n");
        while(TRUE)
        {
        printf ("enter name of sulphur percentage range file :");gets (in_sulf);
```

```
        if(in_sulf[0] == 0)     exit(-1);
        if ( (sulf = fopen (in_sulf, "r")) == NULL )
        printf("Cannot open '%s' : %s\n",in_sulf,strerror(errno));
        else            break;
        }
    while(TRUE)
{
printf ("enter name of btus range file :");    gets (in_btu);
        if(in_btu[0] == 0)      exit(-1);
        if ( (btu = fopen (in_btu, "r")) == NULL )
        printf("Cannot open '%s' : %s\n",in_btu,strerror(errno));
        else            break;
        }
    while(TRUE)
{
printf ("enter name of ash percentage range file :"); gets (in_ash);
        if(in_ash[0] == 0)      exit(-1);
        if ( (ash = fopen (in_ash, "r")) == NULL )
        printf("Cannot open '%s' : %s\n",in_ash,strerror(errno));
        else            break;
        }
        ns=0;
while ( ( fscanf (sulf, " %lf%lf",&sulf_one[ns],&sulf_two[ns])) != EOF)
        {
        ns++;
                }
nb=0;
while ( ( fscanf (btu, " %lf%lf",&btu_one[nb],&btu_two[nb])) != EOF)
        {
        nb++;
        }
na=0;
while ( ( fscanf (ash, " %lf%lf",&ash_one[na],&ash_two[na])) != EOF)
                {
                na++;
                }
switch (choice)
{
        case 1:
        grid_data(x_coord,y_coord,dep_thick,n,np,x_base,y_base,x_grid,
                                y_grid,nx,ny,thickness);
        grid_data (x_coord,y_coord,over_depth,n,np,x_base,y_base,x_grid,
                                y_grid,nx,ny,burden);
        grid_data (x_coord,y_coord,sulphur,n,np,x_base,y_base,x_grid,
                                y_grid,nx,ny,sulfer);
        grid_data (x_coord,y_coord,btus,n,np,x_base,y_base,x_grid,y_grid,
                                nx,ny,btunit);
        grid_data (x_coord,y_coord,ashs,n,np,x_base,y_base,x_grid,y_grid,
                                nx,ny,ashunit);
        grid_data (x_coord,y_coord,inburd,n,np,x_base,y_base,x_grid,y_grid,
                                nx,ny,inner);
        grid_data (x_coord,y_coord,sand,n,np,x_base,y_base,x_grid,y_grid,
                                nx,ny,sandstone);
        break;
        case 2:
        gridq_data (x_coord,y_coord,dep_thick,n,np,x_base,y_base,x_grid,
                                y_grid,nx,ny,thickness);
        gridq_data (x_coord,y_coord,over_depth,n,np,x_base,y_base,x_grid,
                                y_grid,nx,ny,burden);
        gridq_data (x_coord,y_coord,sulphur,n,np,x_base,y_base,x_grid,
                                y_grid,nx,ny,sulfer);
        gridq_data (x_coord,y_coord,btus,n,np,x_base,y_base,x_grid,
                                y_grid,nx,ny,btunit);
        gridq_data (x_coord,y_coord,ashs,n,np,x_base,y_base,x_grid,
                                y_grid,nx,ny,ashunit);
        gridq_data (x_coord,y_coord,inburd,n,np,x_base,y_base,x_grid,y_grid,
                                nx,ny,inner);
        gridq_data (x_coord,y_coord,sand,n,np,x_base,y_base,x_grid,y_grid,
```

```
                                    nx,ny,sandstone);
            break;
        case 3:
            grido_data (x_coord,y_coord,dep_thick,n,np,x_base,y_base,x_grid,
                                    y_grid,nx,ny,thickness);
            grido_data (x_coord,y_coord,over_depth,n,np,x_base,y_base,x_grid,
                                    y_grid,nx,ny,burden);
            grido_data (x_coord,y_coord,sulphur,n,np,x_base,y_base,x_grid,
                                    y_grid,nx,ny,sulfer);
            grido_data (x_coord,y_coord,btus,n,np,x_base,y_base,x_grid,
                                    y_grid,nx,ny,btunit);
            grido_data (x_coord,y_coord,ashs,n,np,x_base,y_base,x_grid,
                                    y_grid,nx,ny,ashunit);
            grido_data (x_coord,y_coord,inburd,n,np,x_base,y_base,x_grid,y_grid,
                                    nx,ny,inner);
            grido_data (x_coord,y_coord,sand,n,np,x_base,y_base,x_grid,y_grid,
                                    nx,ny,sandstone);
            break;
            default:printf(" No option was chosen!!!!");  exit(0);
            }
        recovery_percentage ( inner, sandstone, recovery);
        block_thickness( matrix,ny,nx,avg_bl_thick);
        block_thickness( recovery,ny,nx,avg_bl_recov);
        ton_inside_block(ny,nx,x_grid,y_grid,avg_bl_thick,block_ton);
        reton_inside_block(ny,nx,block_ton,avg_bl_recov,real_bl_ton);
            block_thickness( burden,ny,nx,avg_bl_depth);
            for(i=0; i<bound_count; i++)
            {
avgthick_inside_boundary(y_base,x_base,y_grid,x_grid,ny,nx,bound_list[i].xb,
        bound_list[i].yb,bound_list[i].nbp,avg_bl_thick,&avg_thick);
avgthick_inside_boundary(  y_base, x_base, y_grid, x_grid,ny, nx,
bound_list[i].xb,bound_list[i].yb,bound_list[i].nbp,avg_bl_recov,&&avg_recov);
avgthick_inside_boundary(  y_base, x_base, y_grid, x_grid,ny, nx,
  bound_list[i].xb,bound_list[i].yb,bound_list[i].nbp,avg_bl_depth,&avg_depth);
area(bound_list[i].nbp,bound_list[i].xb,bound_list[i].yb,&mine_bound_area);
tonnage_inside_boundary(mine_bound_area,73,avg_thick,&tons_in_bound);
retonnage_inside_boundary( avg_recov, tons_in_bound,&retons_in_bound);
        fprintf(out,"Area inside mining boundary #%d(square feet) = %.2lf\n",
                                i+1,mine_bound_area);
fprintf(out," = %.2lf acres\n", mine_bound_area/43560);
fprintf(out," Tonnage inside mining boundary #%d=%.2lftons\n",i+1,tons_in_bound);
fprintf(out," = %.2lf million tons\n",        tons_in_bound/1e6);
fprintf(out," Real tonnage inside mining boundary #%d = %.2lf "
                                    "tons\n",i+1, retons_in_bound);
fprintf(out,"= %.2lf million tons\n",retons_in_bound/1e6);
fprintf(out," Average percentage recovery inside mining boundary #%d"
                                    " = %.2lf percent\n",i+1,avg_recov*100);
fprintf(out," Average seam thickness inside mining boundary #%d "
                                    "= %.2lf inches\n",    i+1,avg_thick);
fprintf(out," Average depth insideminingboundary#%d=%.2lf"feet\n",i+1,avg_depth);
        fprintf(out,"\n");fprintf(out, "\t\t\t--------------------\n");
            }
            block_thickness( sulfer,ny,nx,avg_bl_sulf);fprintf(out, "\n");
            fprintf(out, "\t\t\t\t\t\t\t\t\tSULPHUR\n");
            fprintf(out, "\t\t\t\t\t\t\t\t\t-------\n");fprintf(out,"\n");
            fprintf(out, "\t\tPercentage\t\t\t\tAccumulated\t\t\tAccumulated"
                                        "\t\t\tNumber of\n");
            fprintf(out,"\t\t\t\t\t\tTonnage\t\t\t\tRealtonnage"\t\tBlocks\n");
f           printf(out,"\n");
            for ( i=0; i< ns; i++)
            {
        block_range(sulf_one[i],sulf_two[i],block_ton,real_bl_ton,ny,nx,
                        avg_bl_sulf,&accum_ton,&block,&accum_reton);
            fprintf(out,"%10.2lf - %10.2lf\t\t\t%10.2lf\t\t\t%10.2lf\t\t\t\t"
            "\t%d\n",sulf_one[i],sulf_two[i],accum_ton,accum_reton,block);
            }
        fprintf(out, "\n");
   tons_inside_grid ( ny,nx,block_ton,real_bl_ton,&total_tons,&total_retons);
```

```
        fprintf(out,"Total tons inside grid = %10.2lf\tTons\n",total_tons);
     fprintf(out,"Real total tons inside grid = %10.2lf\tTons\n",total_retons);
        fprintf(out,"Total blocks          = %d\tBlocks\n",(ny-1)*(nx-1));
fprintf(out,"Average tons/block  = %10.2lf\tTons\n",total_tons/((ny-1)*(nx-1)));
fprintf(out,"Average realtons/block= %10.2lf\tTons\n",
total_retons/((nyl)*(nxl)));avg_inter_percent(ny,nx,avg_bl_sulf,&avg_inter_sulf);
                fprintf(out,"Average interior percent=%10.2lf\n", avg_inter_sulf);
                fprintf(out,"\n");
                block_thickness( btunit,ny,nx,avg_bl_btu);fprintf(out, "\n");
                fprintf(out, "\t\t\t\t\t\t\t\t\tBTU\n");
                fprintf(out, "\t\t\t\t\t\t\t\t\t---\n");fprintf(out,"\n");
fprintf(out,"\t\tBTU/lb\t\t\t\t\tAccumulated\t\tAccumulated\t\t\t\tNumberof\n");
fprintf(out, "        \t\t\t\t\t\tTonnage\t\t\tRealtonnage"\t\t\tBlocks\n");
        fprintf(out,"\n");
                for ( i=0; i< nb; i++)
                {
block_range (btu_one[i],btu_two[i],block_ton,real_bl_ton, ny, nx,
                                        avg_bl_btu,&accum_ton,&block,&accum_reton);
fprintf(out,"%10.2lf - %10.2lf\t\t\t%10.2lf\t\t\t%10.2lf\t\t\t\t\t%d\n",
                btu_one[i],btu_two[i],accum_ton,accum_reton,block);
                }fprintf(out, "\n");
        fprintf(out,"Total tons inside grid = %10.2lf\tTons\n",total_tons);
 fprintf(out,"Real total tons inside grid = %10.2lf\tTons\n",total_retons);
        fprintf(out,"Total blocks          = %d\n",(ny-1)*(nx-1));
fprintf(out,"Average tons/block  = %10.2lf\tTons\n",total_tons/((ny-1)*(nx-1)));
fprintf(out,"Average realtons/block=%10.2lf\tTons\n",total_retons/((ny-1)*(nx-
      )));
        avg_inter_percent(ny,nx,avg_bl_btu,&avg_inter_btu);
fprintf(out,"Average interior BTU/lb=%10.2lf\n",avg_inter_btu);fprintf(out,"\n");
                block_thickness( ashunit,ny,nx,avg_bl_ash);
        fprintf(out, "\t\t\t\t\t\t\t\t\tASH\n");
                fprintf(out, "\t\t\t\t\t\t\t\t\t---\n");
                fprintf(out,"\n");
                fprintf(out, "\t\tPercentage\t\t\t\t\tAccumulated\t\t\tAccumulated"
                                                "\t\t\tNumber of\n");
fprintf(out, "   \t\t\t\t\t\tTonnage\t\t\t\tRealtonnage"\t\tBlocks\n");
                fprintf(out,"\n");
                for ( i=0; i< na; i++)
                {
                block_range (ash_one[i],ash_two[i],block_ton,real_bl_ton, ny, nx,
                                        avg_bl_ash,&accum_ton,&block,&accum_reton);
fprintf(out,"%10.2lf   -   %10.2lf\t\t\t%10.2lf\t\t\t%10.2lf\t\t\t\t\t%d\n",
ash_one[i],ash_two[i],accum_ton,       accum_reton,block);
                }
        fprintf(out, "\n");
        fprintf(out,"Total tons inside grid = %10.2lf\tTons\n",total_tons);
 fprintf(out,"Real total tons inside grid = %10.2lf\tTons\n",total_retons);
        fprintf(out,"Total blocks          = %d\n",(ny-1)*(nx-1));
fprintf(out,"Average tons/block  = %10.2lf\tTons\n",total_tons/((ny-1)*(nx-1)));
fprintf(out,"Average realtons/block= %10.2lf\tTons\n",total_retons/((ny-1)*(nx-
      1)));
avg_inter_percent(ny,nx,avg_bl_ash,&avg_inter_ash);
fprintf(out,"Average interiorpercent=%10.2lf\n",avg_inter_ash);fprintf(out,"\n");
press_inside_block(ny,nx,avg_bl_depth,150,avg_bl_pressure);
                fprintf(out,"\n");
                fprintf(out,"\tBLOCK\t\tTONS/BLOCK\t\tPERCENTAGE\tRETONS/BLOCK\t"
                                        "AVERAGE\t\tAVERAGE\t\tAVERAGE\t\tAVERAGE\n");
fprintf(out,"\t\t\tRECOVERY\t\t"DENSITY\t\tTHICKNESS\tDEPTH\t\tPRESSURE\n");
fprintf(out,"\t\t\t(percent)\t\t"P.C.F \t\t   FT    \t  FT\t\t  PSI \n");
        fprintf(out,"\n");
                for (i=0; i<ny-1; i++)
                        for (j=0; j<nx-1; j++)
                        {
fprintf(out, " block[%d][%d]\t%10.2lf\t%10.2lf\t\t%10.2lf\t\t"
"100.0\t\t%.2lf\t%10.2lf\t%10.2lf\n", i,j,block_ton[i][j],avg_bl_recov[i][j]*100,
real_bl_ton[i][j],avg_bl_thick[i][j]/12,avg_bl_depth[i][j],avg_bl_pressure[i][j])
                }
    }
```

```
        else
        {
         fprintf(out, "\t\t\t\tINPUT DATA \n ");fprintf(out,"\n");
fprintf(out, "\tEAST\t\t\tNORTH\t\t\tOVERBURDEN\t\DEPOSIT\t\t\PERCENT\n");
fprintf(out, "\tCOORDS\t\t\tCOORDS\t\t\tDEPTH\t\t\THICKNESS\t\RECOVERY\n");
                 fprintf(out,"\n");
        n =0;
while((fscanf(in,%lf%lf%lf%lf%lf%lf",&x_coord[n],&y_coord[n],&over_depth[n],
              &dep_thick[n],inburd[n], sand[n])) != EOF)
            {
fprintf(out,"%10.2lf\t\t%10.2lf\t\t%10.2lf\t\t%10.2lf\n",x_coord[n],y_coord[n],
                        over_depth[n],dep_thick[n]);
                                  n++;
                        }
        fprintf(out,"\n");
        switch (choice)
        {
                case 1:
                grid_data (x_coord,y_coord,dep_thick,n,np,x_base,y_base,x_grid,
                                      y_grid,nx,ny,thickness);
                grid_data (x_coord,y_coord,over_depth,n,np,x_base,y_base,x_grid,
                                           y_grid,nx,ny,burden);
                grid_data (x_coord,y_coord,inburd,n,np,x_base,y_base,x_grid,y_grid,
                                      nx,ny,inner);
                grid_data (x_coord,y_coord,sand,n,np,x_base,y_base,x_grid,y_grid,
                                      nx,ny,sandstone);
            break;
            case 2:
            gridq_data (x_coord,y_coord,dep_thick,n,np,x_base,y_base,x_grid,
                                 y_grid,nx,ny,thickness);
                gridq_data (x_coord,y_coord,over_depth,n,np,x_base,y_base,x_grid,
                                           y_grid,nx,ny,burden);
                gridq_data (x_coord,y_coord,inburd,n,np,x_base,y_base,x_grid,y_grid,
                                      nx,ny,inner);
                gridq_data (x_coord,y_coord,sand,n,np,x_base,y_base,x_grid,y_grid,
                                      nx,ny,sandstone);
            break;
            case 3:
            grido_data (x_coord,y_coord,dep_thick,n,np,x_base,y_base,x_grid,
                                 y_grid,nx,ny,thickness);
                grido_data (x_coord,y_coord,over_depth,n,np,x_base,y_base,x_grid,
                                           y_grid,nx,ny,burden);
                grido_data (x_coord,y_coord,inburd,n,np,x_base,y_base,x_grid,y_grid,
                                      nx,ny,inner);
                grido_data (x_coord,y_coord,sand,n,np,x_base,y_base,x_grid,y_grid,
                                      nx,ny,sandstone);
            break;
            default:
            printf(" No option was chosen!!!!");      exit(0);
                }
                 block_thickness( matrix,ny,nx,avg_bl_thick);
                block_thickness( burden,ny,nx,avg_bl_depth);
                read_matrix ( recovery, ny,nx);
                block_thickness( recovery,ny,nx,avg_bl_recov);
                press_inside_block(ny,nx,avg_bl_depth,150,avg_bl_pressure);
                ton_inside_block(ny,nx,x_grid,y_grid,avg_bl_thick,block_ton);
        reton_inside_block(ny,nx,block_ton,avg_bl_recov,real_bl_ton);
                for(i=0; i<bound_count; i++)
                {
avgthick_inside_boundary(y_base,x_base,y_grid,x_grid,ny,nx,bound_list[i].xb,
bound_list[i].yb,bound_list[i].nbp,avg_bl_thick,&avg_thick);
avgthick_inside_boundary(y_base,x_base,y_grid,x_grid,ny,nx,bound_list[i].xb,
bound_list[i].yb,bound_list[i].nbp,avg_bl_recov,&avg_recov);
avgthick_inside_boundary(y_base,x_base,y_grid,x_grid,ny,nx,bound_list[i].xb,
bound_list[i].yb,bound_list[i].nbp,avg_bl_depth,&avg_depth);
area(bound_list[i].nbp,bound_list[i].xb,bound_list[i].yb,&mine_bound_area);
tonnage_inside_boundary(mine_bound_area,100,avg_thick,&tons_in_bound);
retonnage_inside_boundary( avg_recov, tons_in_bound,&retons_in_bound);
```

```
fprintf(out," Area inside mining boundary #%d(square feet) ="
                    " %.2lf\n",      i+1,mine_bound_area);
fprintf(out,"="" %.2lf acres\n", mine_bound_area/43560);
fprintf(out," Tonnage inside mining boundary #%d=%.2lftons\n",i+1,tons_in_bound);
fprintf(out,"= ""%.2lf million tons\n", tons_in_bound/1e6);
fprintf(out," Real tonnage inside mining boundary #%d ="
                    " %.2lf tons\n",i+1,retons_in_bound);
fprintf(out,"="" %.2lf million tons\n",retons_in_bound/1e6);
fprintf(out," Average percentage recovery inside mining boundary #%d = %.2lf
       percent\n",i+1,avg_recov*100);
fprintf(out," Average seam thickness inside mining boundary #%d "
              "= %.2lf inches\n",      i+1,avg_thick);
fprintf(out," Average depth inside miningboundary#%d=%.2lffeet\n",i+1,avg_depth);
fprintf(out,"\n");fprintf(out, "\t\t\t--------------------\n");
              }       fprintf(out,"\n");
fprintf(out,"\tBLOCK\t\tTONS/BLOCK\t\tPERCENTAGE\tRETONS/BLOCK\tAVERAGE"
                    "\t\tAVERAGE\t\tAVERAGE\t\tAVERAGE\n");
fprintf(out,"\t\t\tRECOVERY\t\tDENSITY\t\tTHICKNESS\tDEPTH\t\tPRESSURE\n");
fprintf(out,"\t\t\t(percent)\t\tP.C.F \t\t   FT    \t  FT\t\t  PSI \n");
fprintf(out,"\n");
              for (i=0; i<ny-1; i++)
                     for (j=0; j<nx-1; j++)
                     {
fprintf(out,"block[%d][%d]\t%10.2lf\t%10.2lf\t\t%10.2lf\t\t100.0\t\t%.2lf\t
%10.2lf\t%10.2lf\n",i,j,block_ton[i][j],avg_bl_recov[i][j]*100,real_bl_ton[i][j],
avg_bl_thick[i][j]/12,avg_bl_depth[i][j],avg_bl_pressure[i][j]);      }
       }
      }
```

APPENDIX B

Data For Case Study No. 1

INPUT DATA FILE FOR CASE STUDY NO.1

EAST COORDS	NORTH COORDS	OVERBURDEN DEPTH	DEPOSIT THICKNESS	BTUs	SULPHUR	ASH	Inner-burden	sands-tone
7820.00	2180.00	1822.00	41.00	10163.00	1.60	9.20	8	40
2280.00	1840.00	1859.00	37.00	10047.00	1.45	9.50	10	30
15600.00	2840.00	1876.00	36.00	10344.00	1.49	10.20	11	25
10240.00	3120.00	1845.00	35.00	10475.00	1.39	10.10	9	40
13880.00	6120.00	1873.00	43.00	10358.00	1.49	10.80	40	50
7540.00	8060.00	1828.00	39.00	9812.00	1.65	9.80	50	60
9980.00	8280.00	1832.00	40.00	9993.00	0.64	9.70	60	70
9060.00	11220.00	1752.00	42.00	9761.00	0.49	1.00	70	80
11540.00	11020.00	1759.00	42.00	10476.00	0.63	11.10	80	90
10340.00	10160.00	1786.00	41.00	9796.00	1.85	10.90	120	45
13600.00	9160.00	1814.00	38.00	9933.00	1.13	9.40	130	50
13240.00	12820.00	1729.00	51.00	9784.00	0.48	10.50	125	60
18400.00	12220.00	1742.00	48.00	9546.00	0.57	10.60	140	70
15420.00	14880.00	1696.00	38.00	10284.00	0.70	10.70	20	30
19180.00	12920.00	1742.00	58.00	10233.00	0.40	11.40	50	20
22000.00	14180.00	1678.00	54.00	9665.00	2.20	11.50	30	10
19520.00	15320.00	1669.00	50.00	10320.00	2.18	11.35	40	45
18420.00	17460.00	1640.00	48.00	9709.00	1.83	10.20	8	30
21000.00	16680.00	1642.00	46.00	10325.00	1.23	9.30	110	40
3320.00	15300.00	1680.00	40.00	10048.00	1.15	11.0	90	75
4000.00	16000.00	1681.00	40.00	10457.00	1.10	0.05	20	80
2960.00	16540.00	1674.00	45.00	9706.00	1.69	9.85	15	85
2320.00	16480.00	1688.00	46.00	9928.00	0.52	9.65	120	35
2040.00	16520.00	1675.00	36.00	9950.00	0.78	10.25	130	50
8420.00	15200.00	1680.00	40.00	10348.00	0.82	11.20	60	65
10900.00	16000.00	1671.00	40.00	9666.00	1.00	9.50	50	75
1400.00	18640.00	1666.00	39.00	10415.00	1.10	10.75	23	40
5160.00	20560.00	1678.00	33.00	10485.00	1.45	11.00	40	75
7280.00	17100.00	1668.00	43.00	9901.00	1.53	9.00	50	35
10360.00	20860.00	1619.00	40.00	10011.00	0.62	10.00	60	15
11160.00	18440.00	1634.00	31.00	9533.00	0.44	10.60	70	30
15380.00	18120.00	1641.00	41.00	10193.00	1.66	10.40	60	40
16080.00	21580.00	1580.00	46.00	9505.00	1.51	11.30	50	50
19380.00	20800.00	1577.00	37.00	10119.00	2.02	9.45	60	60

Sulphur Percentage Range File

0.00	-	0.20
0.20	-	0.50
0.50	-	0.70
0.70	-	1.00
1.00	-	1.20
1.20	-	1.50
1.50	-	1.70
1.70	-	2.00
2.00	-	2.20
2.20	-	2.50

BTU's Values Range File

0.00	-	9500.00
9500.00	-	9700.00
9700.00	-	9900.00
9900.00	-	10000.00
10000.00	-	10200.00
10200.0	-	10400.00
10400.00	-	10600.00

Ash Percentage Range File

0.00	-	9.20
9.20	-	9.50
9.50	-	9.70
9.70	-	10.00
10.00	-	10.20
10.20	-	10.50
10.50	-	10.70
10.70	-	11.00
11.00	-	11.20
11.20	-	11.50

****
TONNAGE INFORMATION
For Lower Seam Mining
(Case Study No.1)

Using Neighbor Search Gridding

With 1000 ft grid spacing
****

No. of points to estimate each grid = 6
x grid spacing = 1000.000
y grid spacing = 1000.000
Number of x grid points = 24
Number of y grid points = 27
Starting x grid coordinate = 0.00
Starting y grid coordinate = 0.00

Mining Boundary Co-ordinates

There is 1 boundarie(s):

Boundary #1

	EAST COORDS	NORTH COORDS
1:	8000	21000
2:	16000	21000
3:	20000	19000
4:	20000	18600
5:	20300	18400
6:	20500	18100
7:	19900	17900
8:	20200	17300
9:	20000	16900
10:	20300	16600
11:	20000	16500
12:	20100	16200
13:	19500	15800
14:	19800	15400

15:	20300	15400
16:	20600	15200
17:	20300	15100
18:	19700	14800
19:	19950	14250
20:	20100	13900
21:	20300	13800
22:	20500	13500
23:	19400	13400
24:	19200	13000
25:	19000	12600
26:	19000	12200
27:	19350	11500
28:	19150	11300
29:	19400	11100
30:	20000	11000
31:	20500	11000
32:	20600	10500
33:	20500	9900
34:	20400	9300
35:	20500	8700
36:	21000	8250
37:	21150	8000
38:	20900	7900
39:	21000	7000
40:	21200	6100
41:	20500	5900
42:	20200	6100
43:	20000	5800
44:	19800	5600
45:	19600	5400
46:	19700	5200
47:	19800	5000
48:	18000	6000
49:	17000	6000
50:	17000	3000
51:	18000	1000
52:	10000	1000
53:	7000	5000

54:	7000	8000
55:	9000	8000
56:	9000	12000
57:	8000	13000
58:	7000	13000
59:	7000	12000
60:	6000	12000
61:	6000	18000

Area inside mining boundary #1(square feet) = 233683750.00
= 5364.64 acres

Tonnage inside mining boundary #1 = 41178314.42 tons
= **41.18 million tons**

Recoverable tonnage inside mining boundary #1 = 28265809.34 tons
= **28.27 million tons**

Average percentage recovery inside mining boundary #1 = 68.64 percent
Average seam thickness inside mining boundary #1 = **42.29 inches**
Average depth inside mining boundary #1 = 1705.95 feet

SULPHUR

Percentage		Accumulated Tonnage	Accumulated Real tonnage	Number of Blocks
0.00 -	0.20	0.00	0.00	0
0.20 -	0.50	0.00	0.00	0
0.50 -	0.70	758513.59	473157.04	4
0.70 -	1.00	17569776.10	11458894.85	100
1.10 -	1.20	20609709.73	13574571.76	120
1.20 -	1.50	34796930.86	23298235.67	209
1.50 -	1.70	6191938.84	4761380.48	32
1.70 -	2.00	8152074.00	6493201.56	44
2.00 -	2.20	0.00	0.00	0
2.20 -	2.50	0.00	0.00	0

Total tons inside grid = 103200713.97 Tons
Total Revoverable tons inside grid = 70015451.15 Tons
Total blocks = 598
Average tons/block = 172576.44 Tons
Average recoverable tons/block = 117082.69 Tons
Average interior percent = 1.21

BTU

BTU/lb	Accumulated Tonnage	Accumulated Real tonnage	Number of Blocks
0.00 - 9500.00	0.00	0.00	0
9500.00 - 9700.00	0.00	0.00	0
9700.00 - 9900.00	10771318.06	7145254.68	63
9900.00 - 9950.00	9284379.03	6271236.89	53
9950.00 - 10000.00	23464506.98	15979791.44	133
10000.00 - 10200.00	49702140.16	33961113.09	286
10200.00 - 10400.00	7978369.74	6658055.05	63
10400.00 - 10600.00	0.00	0.00	0

Total tons inside grid = 103200713.97 Tons
Total Recoverable tons inside grid = 70015451.15 Tons
Total blocks = 598
Average tons/block = 172576.44 Tons
Average recoverable tons/block = 117082.69 Tons
Average interior BTU/lb = 10036.06

ASH

Percentage	Accumulated Tonnage	Accumulated Real tonnage	Number of Blocks
0.00 - 9.20	0.00	0.00	0
9.20 - 9.50	0.00	0.00	0
9.50 - 9.70	0.00	0.00	0
9.70 - 10.00	5096760.19	3316756.74	32
10.00 - 10.20	28050220.11	19005316.59	170
10.20 - 10.50	46635815.62	31907531.32	274
10.50 - 10.70	11803646.24	7922671.10	65
10.70 - 11.00	10308015.52	6992985.92	51
11.00 - 11.20	1306256.29	870189.48	6
11.20 - 11.50	0.00	0.00	0

Total tons inside grid = 103200713.97 Tons
Total Recoverable tons inside grid = 70015451.15 Tons
Total blocks = 598
Average tons/block = 172576.44 Tons
Average recoverable tons/block = 117082.69 Tons
Average interior percent = 10.33

```
                  ****************************
                  TONNAGE  INFORMATION
                  For Lower Seam Mining
                  (Case  Study  No.1)
                  Using  Quadrant  Search  Gridding
                  With  1000  ft  grid  spacing
                  ********************************
 No. of points to estimate each grid = 6
 x grid spacing =  1000.000
 y grid spacing =  1000.000
 Number of x grid points = 24
 Number of y grid points = 27
 Starting x grid coordinate = 0.00
 Starting y grid coordinate = 0.00

Area inside mining boundary #1(square feet) = 233683750.00
                                            = 5364.64 acres
 Tonnage inside mining boundary #1          = 39559544.93 tons
                                            = 39.56 million tons
 Recoverable tonnage inside mining boundary #1   = 26759050.63 tons
                                                 = 26.76 million tons
 Average percentage recovery inside mining boundary #1 = 67.64 percent
 Average seam thickness inside mining boundary #1      = 40.63 inches
 Average depth inside mining boundary #1               = 1803.91 feet
                  ****************************
                  TONNAGE  INFORMATION
                  For Lower Seam Mining
                  (Case  Study  No.1)
                  Using  Octant  Search  Gridding
                  With  1000  ft  grid  spacing
                  ********************************
 No. of points to estimate each grid = 6
 x grid spacing =  1000.000
 y grid spacing =  1000.000
 Number of x grid points = 24
 Number of y grid points = 27
 Starting x grid coordinate = 0.00
 Starting y grid coordinate = 0.00
Area inside mining boundary #1(square feet) = 233683750.00
```

= 5364.64 acres
Tonnage inside mining boundary #1 = 38545667.78 tons
= **38.55 million tons**
Recoverable tonnage inside mining boundary #1 = 25779895.29 tons
= **25.78 million tons**
Average percentage recovery inside mining boundary #1 = 66.88 percent
Average seam thickness inside mining boundary #1 = **39.59 inches**
Average depth inside mining boundary #1 = 1828.96 feet

****

TONNAGE INFORMATION
For Lower Seam Mining
(Case Study No.1)
Using Neighbor Search Gridding
With 2000 ft grid spacing

****

No. of points to estimate each grid = 6
x grid spacing = 2000.000
y grid spacing = 2000.000
Number of x grid points = 12
Number of y grid points = 14
Starting x grid coordinate = 0.00
Starting y grid coordinate = 0.00

Area inside mining boundary #1(square feet) = 233683750.00
= 5364.64 acres
Tonnage inside mining boundary #1 = 41200706.96 tons
= **41.20 million tons**
Recoverable tonnage inside mining boundary #1 = 28106955.26 tons
= **28.11 million tons**
Average percentage recovery inside mining boundary #1 = 68.22 percent
Average seam thickness inside mining boundary #1 = **42.31 inches**
Average depth inside mining boundary #1 = 1708.07 feet

```
                         ******************************
                         TONNAGE  INFORMATION
                         For Lower Seam Mining
                         (Case  Study  No.1)
                         Using  Quadrant  Search  Gridding
                         With  2000  ft  grid  spacing
                         ********************************
No. of points to estimate each grid = 6
x grid spacing =  2000.000
y grid spacing =  2000.000
Number of x grid points = 12
Number of y grid points = 14
Starting x grid coordinate = 0.00
Starting y grid coordinate = 0.00

Area inside mining boundary #1(square feet) = 233683750.00
                                            = 5364.64 acres
Tonnage inside mining boundary #1           = 39522938.01 tons
                                              = 39.52 million tons
Recoverable tonnage inside mining boundary #1   = 26912984.41 tons
                                                = 26.91 million tons
Average percentage recovery inside mining boundary #1     = 68.09 percent
Average seam thickness inside mining boundary #1            = 40.59 inches
Average depth inside mining boundary #1                    = 1804.98 feet
                         ******************************
                         TONNAGE  INFORMATION
                         For Lower Seam Mining
                         (Case  Study  No.1)
                         Using  Octant  Search  Gridding
                         With  2000  ft  grid  spacing
                         ********************************

No. of points to estimate each grid = 6
x grid spacing =  2000.000
y grid spacing =  2000.000
Number of x grid points = 12
Number of y grid points = 14
Starting x grid coordinate = 0.00
Starting y grid coordinate = 0.00
```

Area inside mining boundary #1(square feet) = 233683750.00
= 5364.64 acres

Tonnage inside mining boundary #1 = 38598369.88 tons
= **38.60 million tons**

Recoverable tonnage inside mining boundary #1 = 25827003.85 tons
= **25.83 million tons**

Average percentage recovery inside mining boundary #1 = 66.91 percent
Average seam thickness inside mining boundary #1 = **39.64 inches**
Average depth inside mining boundary #1 = 1829.01 feet

APPENDIX C

Summarized Block Data

BLOCK	TONS/BLOCK	PERCENTAGE RECOVERY (PERCENT)	RETONS/BLOCK	AVG. THICK FT	AVG DEPTH FT	Average Innerb- FT
block[0][0]	645999.27	67.00	432819.51	3.23	1820.25	49.79
block[0][1]	642026.74	66.00	423737.65	3.21	1828.48	43.75
block[0][2]	637806.33	65.25	416168.63	3.19	1840.54	36.23
block[0][3]	631988.97	63.75	402892.97	3.16	1847.97	29.50
block[0][4]	623936.67	62.75	391520.26	3.12	1851.61	22.87
block[0][5]	621492.58	65.00	403970.18	3.11	1854.26	19.80
block[0][6]	623773.98	67.00	417928.57	3.12	1856.96	22.99
block[0][7]	624264.24	67.00	418257.04	3.12	1859.21	27.39
block[0][8]	626173.46	67.00	419536.22	3.13	1859.19	30.24
block[0][9]	638732.41	67.00	427950.71	3.19	1850.99	39.60
block[0][10]	663913.23	66.75	443162.08	3.32	1839.12	49.59
block[1][0]	649461.77	66.75	433515.73	3.25	1812.40	46.95
block[1][1]	646107.13	66.75	431276.51	3.23	1820.29	38.49
block[1][2]	643962.45	66.00	425015.21	3.22	1825.81	26.39
block[1][3]	637659.04	64.25	409695.94	3.19	1837.60	22.42
block[1][4]	624637.20	62.75	391959.84	3.12	1847.37	27.13
block[1][5]	620844.98	64.50	400445.01	3.10	1852.56	28.78
block[1][6]	628173.60	67.25	422446.75	3.14	1856.69	33.24
block[1][7]	627971.84	67.75	425450.92	3.14	1860.55	45.23
block[1][8]	631429.55	67.25	424636.37	3.16	1857.82	55.01
block[1][9]	657629.23	66.75	438967.51	3.29	1844.44	61.85
block[1][10]	694282.88	66.25	459962.41	3.47	1829.92	57.06
block[2][0]	651816.95	66.25	431828.73	3.26	1793.31	55.18
block[2][1]	653175.56	67.75	442526.44	3.27	1809.14	50.91
block[2][2]	652352.67	67.75	441968.94	3.26	1814.20	41.09
block[2][3]	650080.54	67.25	437179.16	3.25	1823.01	39.66
block[2][4]	641708.89	66.25	425132.14	3.21	1836.78	41.11
block[2][5]	639435.57	66.50	425224.66	3.20	1844.96	37.49
block[2][6]	656677.30	69.50	456390.72	3.28	1855.01	44.97
block[2][7]	653929.42	70.50	461020.24	3.27	1860.70	56.69
block[2][8]	658378.77	68.25	449343.51	3.29	1847.76	61.94
block[2][9]	704498.88	66.25	466730.51	3.52	1824.45	67.46

BLOCK	TONS/BLOCK	PERCENTAGE RECOVERY (PERCENT)	RETONS/BLOCK	AVG. THICK FT	AVG DEPTH FT	Average Innerb- FT
block[2][10]	737370.78	65.25	481134.43	3.69	1808.19	64.65
block[3][0]	666869.27	65.25	435132.20	3.33	1754.99	62.38
block[3][1]	665097.87	67.00	445615.58	3.33	1784.93	59.11
block[3][2]	660262.82	69.75	460533.32	3.30	1803.75	57.84
block[3][3]	659668.47	72.00	474961.30	3.30	1813.60	65.14
block[3][4]	659499.31	71.25	469893.26	3.30	1823.21	69.81
block[3][5]	664203.20	71.25	473244.78	3.32	1826.18	67.05
block[3][6]	681163.35	73.25	498952.15	3.41	1832.70	69.07
block[3][7]	685681.16	72.25	495404.64	3.43	1836.29	70.19
block[3][8]	705213.99	67.75	477782.48	3.53	1822.12	64.76
block[3][9]	751362.00	64.75	486506.89	3.76	1800.52	70.37
block[3][10]	781938.53	64.50	504350.35	3.91	1780.27	67.61
block[4][0]	682516.98	64.00	436810.87	3.41	1716.09	66.30
block[4][1]	673918.02	65.25	439731.51	3.37	1752.58	66.24
block[4][2]	667047.37	70.25	468600.78	3.34	1786.29	75.02
block[4][3]	668670.10	74.50	498159.22	3.34	1796.84	87.93
block[4][4]	671673.60	73.25	492000.91	3.36	1800.96	95.94
block[4][5]	681047.30	72.75	495461.91	3.41	1798.68	97.11
block[4][6]	687986.23	73.75	507389.85	3.44	1800.09	91.12
block[4][7]	712396.18	70.75	504020.30	3.56	1799.71	80.47
block[4][8]	757783.75	65.25	494453.90	3.79	1780.27	67.38
block[4][9]	790725.98	63.25	500134.18	3.95	1760.47	70.92
block[4][10]	815170.18	65.25	531898.55	4.08	1740.96	70.27
block[5][0]	683338.03	63.50	433919.65	3.42	1690.87	69.71
block[5][1]	681794.99	63.25	431235.33	3.41	1709.09	71.57
block[5][2]	677456.22	66.25	448814.74	3.39	1742.31	82.92
block[5][3]	677652.30	71.25	482827.26	3.39	1765.98	95.01
block[5][4]	687177.65	72.00	494767.90	3.44	1773.36	101.87
block[5][5]	701574.08	71.00	498117.59	3.51	1769.74	101.24
block[5][6]	713387.41	69.75	497587.72	3.57	1770.16	96.81
block[5][7]	742756.40	66.00	490219.22	3.71	1767.31	85.00
block[5][8]	791479.53	62.00	490717.31	3.96	1746.88	68.76

BLOCK	TONS/BLOCK	PERCENTAGE RECOVERY (PERCENT)	RETONS/BLOCK	AVG. THICK FT	AVG DEPTH FT	Average Innerb- FT
block[5][9]	827339.68	61.75	510882.25	4.14	1726.01	71.42
block[5][10]	845198.60	66.00	557831.08	4.23	1707.66	71.46
block[6][0]	684080.80	63.00	430970.90	3.42	1678.05	65.56
block[6][1]	686899.10	60.75	417291.20	3.43	1681.29	63.56
block[6][2]	685717.55	60.00	411430.53	3.43	1697.53	74.73
block[6][3]	681992.59	64.50	439885.22	3.41	1721.54	84.09
block[6][4]	693438.60	68.75	476739.04	3.47	1735.64	89.16
block[6][5]	713423.60	68.25	486911.61	3.57	1736.35	84.52
block[6][6]	730532.81	65.25	476672.66	3.65	1731.37	78.00
block[6][7]	750725.08	63.00	472956.80	3.75	1726.41	72.48
block[6][8]	796574.40	62.75	499850.43	3.98	1720.18	58.01
block[6][9]	845262.28	64.75	547307.33	4.23	1709.64	77.44
block[6][10]	866930.33	67.50	585177.97	4.33	1694.41	63.23
block[7][0]	683999.30	65.00	444599.54	3.42	1678.18	48.92
block[7][1]	681442.24	59.25	403754.53	3.41	1679.28	55.62
block[7][2]	686756.60	55.25	379433.02	3.43	1680.97	64.22
block[7][3]	684744.05	59.50	407422.71	3.42	1683.85	68.99
block[7][4]	680185.35	64.25	437019.09	3.40	1692.28	66.80
block[7][5]	689224.87	66.00	454888.41	3.45	1696.22	58.56
block[7][6]	699831.79	65.25	456640.25	3.50	1692.17	50.12
block[7][7]	728956.09	66.00	481111.02	3.64	1691.83	54.37
block[7][8]	779923.09	70.00	545946.16	3.90	1689.16	56.30
block[7][9]	827805.04	74.75	618784.27	4.14	1683.88	73.16
block[7][10]	852813.80	74.75	637478.32	4.26	1679.03	59.11
block[8][0]	681256.48	68.00	463254.41	3.41	1676.83	45.46
block[8][1]	680037.45	62.50	425023.40	3.40	1678.23	52.32
block[8][2]	682178.38	56.50	385430.79	3.41	1677.85	58.00
block[8][3]	673159.93	58.50	393798.56	3.37	1671.49	60.53
block[8][4]	653629.67	62.50	408518.55	3.27	1665.26	55.37
block[8][5]	640360.64	65.25	417835.32	3.20	1662.40	47.42
block[8][6]	654289.64	67.00	438374.06	3.27	1659.60	41.10
block[8][7]	701222.55	69.75	489102.73	3.51	1659.12	47.28

BLOCK	TONS/BLOCK	PERCENTAGE RECOVERY (PERCENT)	RETONS/BLOCK	AVG. THICK FT	AVG DEPTH FT	Average Innerb-FT
block[8][8]	748220.07	76.25	570517.80	3.74	1657.08	60.06
block[8][9]	784452.98	82.25	645212.58	3.92	1654.88	58.89
block[8][10]	807274.93	82.25	663983.63	4.04	1658.39	57.64
block[9][0]	669948.40	69.75	467289.01	3.35	1675.15	49.12
block[9][1]	666003.99	67.25	447887.68	3.33	1676.11	47.75
block[9][2]	656380.31	62.00	406955.79	3.28	1674.76	56.83
block[9][3]	645459.16	60.50	390502.79	3.23	1665.28	59.26
block[9][4]	634577.59	62.50	396611.00	3.17	1652.05	54.86
block[9][5]	625652.77	63.75	398853.64	3.13	1643.20	48.64
block[9][6]	648828.51	66.00	428226.81	3.24	1637.95	45.10
block[9][7]	687974.03	69.25	476422.02	3.44	1631.81	49.00
block[9][8]	719599.86	76.00	546895.89	3.60	1627.82	56.01
block[9][9]	737435.89	82.25	606541.02	3.69	1625.92	54.31
block[9][10]	757483.02	83.00	628710.91	3.79	1631.34	53.44
block[10][0]	658793.05	69.75	459508.15	3.29	1675.41	46.81
block[10][1]	647530.82	68.50	443558.61	3.24	1675.73	47.33
block[10][2]	630874.93	65.25	411645.89	3.15	1670.72	56.02
block[10][3]	625079.78	63.00	393800.26	3.13	1659.87	58.16
block[10][4]	632868.20	62.25	393960.45	3.16	1646.06	55.87
block[10][5]	645522.60	62.00	400224.01	3.23	1634.10	50.90
block[10][6]	668125.10	64.25	429270.38	3.34	1624.31	50.91
block[10][7]	698710.47	65.50	457655.36	3.49	1611.74	54.35
block[10][8]	717896.22	71.25	511501.05	3.59	1607.65	55.55
block[10][9]	713020.15	79.50	566851.02	3.57	1608.96	55.23
block[10][10]	722844.65	81.50	589118.39	3.61	1614.36	53.25
block[11][0]	655890.10	68.75	450924.44	3.28	1676.14	47.55
block[11][1]	652309.92	67.75	441939.97	3.26	1673.65	47.78
block[11][2]	637673.76	65.50	417676.31	3.19	1665.38	54.27
block[11][3]	626850.94	63.50	398050.35	3.13	1653.74	56.93
block[11][4]	637992.64	62.00	395555.43	3.19	1640.80	55.32
block[11][5]	649610.39	62.25	404382.47	3.25	1628.83	52.05
block[11][6]	669051.67	65.00	434883.58	3.35	1616.89	52.99

BLOCK	TONS/BLOCK	PERCENTAGE RECOVERY (PERCENT)	RETONS/BLOCK	AVG. THICK FT	AVG DEPTH FT	Average Innerb-FT
block[11][7]	698821.62	65.50	457728.16	3.49	1604.47	55.07
block[11][8]	714700.17	69.50	496716.62	3.57	1602.65	55.62
block[11][9]	715249.21	77.25	552530.01	3.58	1607.99	57.06
block[11][10]	720334.26	80.25	578068.24	3.60	1612.97	53.48
block[12][0]	655762.76	68.50	449197.49	3.28	1674.22	46.30
block[12][1]	659542.22	67.25	443542.15	3.30	1669.16	46.85
block[12][2]	648319.16	65.00	421407.46	3.24	1661.71	52.78
block[12][3]	633810.33	63.50	402469.56	3.17	1649.44	56.28
block[12][4]	641652.53	62.00	397824.57	3.21	1637.94	54.90
block[12][5]	645047.64	63.00	406380.01	3.23	1626.66	52.29
block[12][6]	662333.72	66.25	438796.09	3.31	1614.83	53.96
block[12][7]	684973.08	68.00	465781.69	3.42	1608.90	55.43
block[12][8]	700663.22	70.75	495719.23	3.50	1607.40	55.04
block[12][9]	720468.34	75.75	545754.77	3.60	1610.88	54.80
block[12][10]	729310.96	79.00	576155.66	3.65	1615.61	55.65

APPENDIX D

Data for Case Study No.2

INPUT DATA FILE
FOR CASE STUDY NO.2

EAST COORDS	NORTH COORDS	OVERBURDEN DEPTH	DEPOSIT THICKNESS	BTUs	SULPHUR	ASH	Inner-burden	sands-tone
31350.00	33080.00	112.00	66.00	10163.00	0.54	11.10	10	20
31700.00	32050.00	143.00	81.60	10047.00	0.53	12.00	9	30
31860.00	35020.00	87.00	74.40	10344.00	0.51	10.80	11	40
32210.00	31120.00	154.00	85.20	10475.00	0.53	9.50	8	35
32350.00	34550.00	87.00	66.00	10358.00	0.66	10.20	12	25
33370.00	33100.00	111.00	88.80	9812.00	0.51	9.60	20	50
33670.00	31120.00	135.00	86.40	9993.00	0.64	10.60	30	40
34200.00	34580.00	89.00	80.40	9761.00	0.49	12.20	40	50
34300.00	31100.00	133.00	87.60	10476.00	0.63	9.20	50	60
34440.00	29200.00	167.00	62.40	9796.00	0.45	8.30	45	60
34680.00	32070.00	122.00	80.40	9933.00	0.65	14.00	60	40
34750.00	30110.00	156.00	84.00	9784.00	0.48	10.90	110	50
35270.00	33070.00	109.00	96.00	9546.00	0.57	11.20	120	45
35720.00	32120.00	110.00	97.20	10284.00	0.70	9.00	125	60
36150.00	34600.00	78.00	85.20	10233.00	0.40	11.00	130	55
36250.00	29130.00	143.00	90.00	9665.00	0.60	8.00	120	40
36250.00	31120.00	120.00	88.80	10320.00	0.65	12.70	110	35
36450.00	35200.00	70.00	60.00	9707.00	0.53	6.60	50	55
36740.00	30120.00	133.00	91.20	10325.00	0.53	12.50	60	70
36820.00	28100.00	144.00	85.20	10048.00	0.44	10.30	70	80
37180.00	33120.00	92.00	75.60	10457.00	0.47	11.00	80	85
37190.00	34680.00	77.00	86.40	9706.00	0.58	14.00	90	45
37250.00	29140.00	131.00	73.20	9928.00	0.52	14.70	100	35
37480.00	31170.00	131.00	99.60	9950.00	0.68	13.00	105	50
37700.00	32120.00	87.00	70.80	10348.00	0.62	11.30	140	30
38690.00	32150.00	82.00	85.20	9666.00	0.64	12.00	125	60
38720.00	30130.00	116.00	75.60	10415.00	0.58	8.00	60	20
38720.00	33150.00	55.00	62.40	10485.00	0.45	10.90	70	15
38760.00	28150.00	132.00	97.20	9901.00	0.53	11.10	80	20
39220.00	29180.00	104.00	93.60	10011.00	0.62	10.00	60	30
39780.00	31190.00	75.00	86.40	9533.00	0.44	0.20	50	40

Sulphur Percentage Range File

0.00	-	0.20
0.20	-	0.50
0.50	-	0.70
0.70	-	1.00
1.00	-	1.20
1.20	-	1.50
1.50	-	1.70
1.70	-	2.00
2.00	-	2.20
2.20	-	2.50

BTU's Values Range File

0.00	-	9500.00
9500.00	-	9700.00
9700.00	-	9900.00
9900.00	-	10000.00
10000.00	-	10200.00
10200.0	-	10400.00
10400.00	-	10600.00

Ash Percentage Range File

0.00	-	9.50
9.50	-	9.70
9.70	-	10.00
10.00	-	10.30
10.30	-	10.60
10.60	-	11.00
11.00	-	11.20
11.20	-	11.50
11.50	-	12.00
12.00	-	12.50
12.50	-	13.00
13.00	-	13.50
13.50	-	14.00
14.00	-	14.50

```
******************************
TONNAGE  INFORMATION
For Lower Seam Mining
(Case Study No.2)
Using Neighbor Search Gridding
With 1000 ft grid spacing
********************************
```

No. of points to estimate each grid = 6
x grid spacing = 1000.000
y grid spacing = 1000.000
Number of x grid points = 16
Number of y grid points = 16
Starting x grid coordinate = **29000.00**
Starting y grid coordinate = **25000.00**

Mining Boundary Co-ordinates

There is 1 boundarie(s):
Boundary #1

	EAST COORDS	**NORTH COORDS**
1:	41750	29565.6
2:	41700	30650
3:	41020	30870
4:	40770	31970
5:	40110	32830
6:	39900	33030
7:	38950	33000
8:	38330	33620
9:	37010	35240
10:	34990	35360
11:	34310	35300
12:	33660	34960
13:	32390	34810
14:	31950	35660
15:	31670.1	36400
16:	31250	36400
17:	31250	31300

18:	36460	27850
19:	41750	27850

Area inside mining boundary #1(square feet) = 57976973.80
= 1330.97 acres

Tonnage inside mining boundary #1 = 14417824.95 tons
= **14.42 million tons**

Recoverable tonnage inside mining boundary #1 = 7933500.60 tons
= 7.93 million tons

Average percentage recovery inside mining boundary #1 = 55.03 percent
Average seam thickness inside mining boundary #1 = 81.76 inches
Average depth inside mining boundary #1 = 108.89 feet

SULPHUR

Percentage	**Accumulated Tonnage**	**Accumulated Real tonnage**	**Number of Blocks**
0.00 - 0.20	0.00	0.00	0
0.20 - 0.50	2219945.98	1095914.73	7
0.50 - 0.70	72026022.61	38989823.04	218
0.70 - 1.00	0.00	0.00	0
1.10 - 1.20	0.00	0.00	0
1.20 - 1.50	0.00	0.00	0
1.50 - 1.70	0.00	0.00	0
1.70 - 2.00	0.00	0.00	0
2.00 - 2.20	0.00	0.00	0
2.20 - 2.50	0.00	0.00	0

Total tons inside grid = 74245968.59 Tons
Total Recoverable tons inside grid = 40085737.77 Tons
Total blocks = 225
Average tons/block = 329982.08 Tons
Average recoverable tons/block = 178158.83 Tons
Average interior percent = 0.54

BTU

BTU/lb	**Accumulated Tonnage**	**Accumulated Real tonnage**	**Number of Blocks**
0.00 - 9500.00	0.00	0.00	0

9500.00 - 9700.00	0.00	0.00	0
9700.00 - 9900.00	4685212.43	2457063.04	14
9900.00 - 9950.00	11353032.97	6011761.41	34
9950.00 - 10000.00	11237101.17	6104881.06	34
10000.00 - 10200.00	45314491.59	24596238.83	138
10200.00 - 10400.00	1656130.43	915793.42	5
10400.00 - 10600.00	0.00	0.00	0

Total tons inside grid = 74245968.59 Tons
Total Recoverable tons inside grid = 40085737.77 Tons
Total blocks = 225
Average tons/block = 329982.08 Tons
Average recoverable tons/block = 178158.83 Tons
Average interior BTU/lb = 10033.47

ASH

Percentage	**Accumulated Tonnage**	**Accumulated Real tonnage**	**Number of Blocks**
0.00 - 9.50	666802.41	351738.27	2
9.50 - 9.70	1999719.66	1056682.16	6
9.70 - 10.00	2689442.15	1419584.11	8
10.00 - 10.30	8022847.50	4329745.94	24
10.30 - 10.60	8854940.43	4675482.67	27
10.60 - 11.00	25012351.67	13304087.80	77
11.00 - 11.20	6669210.11	3591119.49	20
11.20 - 11.50	9995249.31	5514417.80	30
11.50 - 12.00	9988467.63	5647727.05	30
12.00 - 12.50	346937.74	195152.48	1
12.50 - 13.00	0.00	0.00	0
13.00 - 13.50	0.00	0.00	0
13.50 - 14.00	0.00	0.00	0
14.00 - 14.50	0.00	0.00	0

Total tons inside grid	= 74245968.59	Tons
Total Recoverable tons inside grid	= 40085737.77	Tons
Total blocks	= 225	
Average tons/block	= 329982.08	Tons
Average recoverable tons/block	= 178158.83	Tons
Average interior percent	= 10.84	

APPENDIX E

Data for Case Study No.3

INPUT DATA FILE
FOR CASE STUDY NO 3

EAST COORDS	NORTH COORDS	OVERBURDEN DEPTH	DEPOSIT THICKNESS	BTUs	SULPHUR	ASH	Inner-burden	sands-tone
500.00	6500.00	1822.00	41.00	10163.00	0.54	9.20	30	40
4500.00	6500.00	1859.00	37.00	10047.00	0.53	9.50	40	50
2000.00	6000.00	1876.00	36.00	10344.00	0.51	10.20	50	50
1500.00	5500.00	1845.00	35.00	10475.00	0.53	10.10	80	20
3500.00	5500.00	1873.00	43.00	10358.00	0.66	10.80	90	30
6500.00	5500.00	1828.00	39.00	9812.00	0.51	9.80	110	40
8000.00	5500.00	1832.00	40.00	9993.00	0.64	9.70	10	45
3000.00	5000.00	1752.00	42.00	9761.00	0.49	11.00	11	30
1500.00	4500.00	1759.00	42.00	10476.00	0.63	11.10	20	20
3000.00	4000.00	1786.00	41.00	9796.00	0.45	10.90	8	30
4500.00	4000.00	1814.00	38.00	9933.00	0.65	9.40	10	40
8000.00	4000.00	1729.00	51.00	9784.00	0.48	10.50	11	50
3500.00	3500.00	1742.00	48.00	9546.00	0.57	10.60	12	20
5500.00	3500.00	1696.00	38.00	10284.00	0.70	10.70	70	30
500.00	3000.00	1742.00	58.00	10233.00	0.40	11.40	90	40
2500.00	2500.00	1678.00	54.00	9665.00	0.60	11.50	50	50
6500.00	2750.00	1669.00	50.00	10320.00	0.65	11.35	120	60
1000.00	1500.00	1640.00	48.00	9709.00	0.53	10.20	130	40
3500.00	1500.00	1642.00	46.00	10325.00	0.53	9.30	110	35
7000.00	1500.00	1680.00	40.00	10048.00	0.44	11.01	125	40
8500.00	1500.00	1681.00	40.00	10457.00	0.47	10.05	110	30
500.00	500.00	1674.00	45.00	9706.00	0.58	9.85	70	50
4500.00	500.00	1688.00	46.00	9928.00	0.52	9.65	60	40
6500.00	500.00	1675.00	36.00	9950.00	0.68	10.25	50	20

****

TONNAGE INFORMATION

(For Case Study no 3)

Using Neighbor Search Gridding

With 1000 ft grid spacing

****

No. of points to estimate each grid = 2
x grid spacing = 1000.000
y grid spacing = 1000.000
Number of x grid points = 10
Number of y grid points = 9
Starting x grid coordinate = 0.00
Starting y grid coordinate = 0.00

Mining Boundary Co-ordinates

There is 3 boundarie(s):

Boundary #1

	EAST COORDS	NORTH COORDS
1:	500	4500
2:	2500	4500
3:	2500	5500
4:	3500	5500
5:	3500	6500
6:	2500	6500
7:	2000	6750
8:	500	5500

Boundary #2

	EAST COORDS	NORTH COORDS
1:	3500	2500
2:	5500	2500
3:	6500	3500
4:	5500	4500
5:	3500	4500
6:	2500	3500

Boundary #3

	EAST COORDS	**NORTH COORDS**
1:	5500	500
2:	8500	500
3:	8500	1500
4:	7750	1750
5:	8750	2250
6:	8750	3250
7:	7500	3250
8:	7500	2500
9:	6500	2500
10:	6500	1500
11:	5500	1500

Area inside mining boundary #1(square feet) = 4500000.00
= 103.31 acres

Tonnage inside mining boundary #1 = 542789.40 tons
= **0.54 million tons**

Recoverable tonnage inside mining boundary #1 = 217266.54 tons
= **0.22 million tons**

Average percentage recovery inside mining boundary #1 = 40.03 percent
Average seam thickness inside mining boundary #1 = **39.66 inches**
Average depth inside mining boundary #1 = 1821.49 feet

Area inside mining boundary #2(square feet) = 6000000.00
= 137.74 acres

Tonnage inside mining boundary #2 = 779383.35 tons
= **0.78 million tons**

Recoverable tonnage inside mining boundary #2 = 333381.23 tons
= **0.33 million tons**

Average percentage recovery inside mining boundary #2 = 42.78 percent
Average seam thickness inside mining boundary #2 = **42.71 inches**
Average depth inside mining boundary #2 = 1739.90 feet

Area inside mining boundary #3(square feet)	= 5781250.00
	= 132.72 acres
Tonnage inside mining boundary #3	= 753884.54 tons
	= **0.75 million tons**
Recoverable tonnage inside mining boundary #3	= 323880.39 tons
	= **0.32 million tons**
Average percentage recovery inside mining boundary #3	= 42.96 percent
Average seam thickness inside mining boundary #3	= **42.87 inches**
Average depth inside mining boundary #3	= 1685.79 feet

VITA

Moncef Sellami was born on November 5, 1965, in Sfax, Tunisia. He attended primary, middle, and high school in the same city he was born in. In 1987, he enrolled at the Colorado School of Mines in Golden, where he received his B. S. in Mining Engineering with a Mathematics Option.

He was awarded a Graduate Research Assistantship by the Department of Mining and Minerals Engineering, Virginia Polytechnic Institute and State University where he is pursuing a graduate program leading to a Masters degree in Mining Engineering.