OPTIMIZING ROOF CONTROL USING PROBABILISTIC
TECHNIQUES IN ROOF FAILURE PREDICTION

by

Richard Louis Fraher

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 Raycocks, Chairman

Mario G. Karfakis

Michael Karmis, Department Head

August, 1992

Blacksburg, Virginia
OPTIMIZING ROOF CONTROL USING PROBABILISTIC TECHNIQUES IN ROOF FAILURE PREDICTION

by

Richard Louis Fraher

Committee Chairman: Christopher Haycocks
Mining and Minerals Engineering

(ABSTRACT)

A major objective in the design stage of an underground mine is the reliable prediction of roof falls' size, frequency and location. Probabilistic simulation of potential roof control problems allows a designer to test the performance of competing mine layouts against assumed roof conditions. By comparing different roof control plans using the simulation, the option that provides the lowest overall cost can be selected. The program ROCSIM (Roof control Optimization Cost SIMulation) was developed to provide a theoretical solution to this problem.

The occurrence and frequency of roof falls are related to the type of roof support, support density, geology, structural discontinuities, location in the mine, and elapsed time between mining and the roof fall. Using a Roof Rating System (RRS) developed for this research, a numerical rating can be given to each area of roof. Using this rating, specific parameters can be assigned to these probability distributions to simulate the occurrence of roof falls within a given geologic setting.

Once the location of a roof fall is determined, a cost is calculated taking into account the production delay that would result and the direct cost
of cleaning up the fall and resupporting the roof. Assigning a cost to a roof fall allows the comparison of competing roof support designs relative to their overall cost. The final decision on the amount of support and room width must be determined based on legal restraints and minimization of mining costs.
ACKNOWLEDGEMENT

The author is indebted to Dr. Christopher Haycocks, his major advisor for his support, guidance and patience during his research at Virginia Tech. It would not have been possible to complete this thesis without his encouragement and patience. The author would also like to thank the other members of his advisory committee, Dr. M. Karmis and Dr. M. Karfakis who provided support, advice, and constructive suggestions which aided in the completion of this research. A special note of thanks is given to Margaret Radcliffe for the final editing of this thesis.

Finally, the author would like to thank his parents who provided the upbringing, encouragement and education that allowed him the chance to earn this degree.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>iv</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>viii</td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Statement of Problem</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Approach / Methodology</td>
<td>2</td>
</tr>
<tr>
<td>II. LITERATURE REVIEW</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Digital Simulation</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Roof Rating Methods</td>
<td>6</td>
</tr>
<tr>
<td>2.2.1 Rock Structure Rating</td>
<td>7</td>
</tr>
<tr>
<td>2.2.2 Rock Quality Designation</td>
<td>7</td>
</tr>
<tr>
<td>2.2.3 Q-system</td>
<td>9</td>
</tr>
<tr>
<td>2.2.4 Rock Mass Rating</td>
<td>10</td>
</tr>
<tr>
<td>2.3 Geological Methods to Predict Roof Falls</td>
<td>12</td>
</tr>
<tr>
<td>III. DEVELOPMENT OF ROOF RATING SYSTEM</td>
<td>15</td>
</tr>
<tr>
<td>3.1 Important Characteristics</td>
<td>15</td>
</tr>
<tr>
<td>3.1.1 Assignment of Rating Values</td>
<td>17</td>
</tr>
<tr>
<td>3.2 Implementation</td>
<td>19</td>
</tr>
<tr>
<td>3.3 Use of the RRS Values</td>
<td>20</td>
</tr>
<tr>
<td>IV. PROGRAM FORMULATION</td>
<td>22</td>
</tr>
<tr>
<td>4.1 Cleanup Costs</td>
<td>25</td>
</tr>
<tr>
<td>4.2 Production Delays</td>
<td>28</td>
</tr>
<tr>
<td>V. MODEL DEVELOPMENT</td>
<td>29</td>
</tr>
<tr>
<td>5.1 Model Design Consideration</td>
<td>29</td>
</tr>
<tr>
<td>5.2 Generating Random Numbers</td>
<td>29</td>
</tr>
<tr>
<td>5.3 Generating Random Variates</td>
<td>31</td>
</tr>
<tr>
<td>5.4 Time Advance Methods</td>
<td>36</td>
</tr>
<tr>
<td>5.5 Linked-Allocation List Storage</td>
<td>40</td>
</tr>
<tr>
<td>5.6 Method of Programming</td>
<td>41</td>
</tr>
<tr>
<td>5.6.1 Programming Language</td>
<td>41</td>
</tr>
<tr>
<td>5.6.2 User Interface</td>
<td>44</td>
</tr>
<tr>
<td>5.7 Sensitivity Analysis</td>
<td>47</td>
</tr>
<tr>
<td>5.8 Graphical Output</td>
<td>48</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.1</td>
<td>How the Roof Bolt Pattern is Defined in ROCSIM</td>
</tr>
<tr>
<td>4.2</td>
<td>Cut sequence used in ROCSIM</td>
</tr>
<tr>
<td>4.3</td>
<td>Roof Fall Shape Used by ROCSIM</td>
</tr>
<tr>
<td>4.3</td>
<td>Areas that Cause a Production Delay in ROCSIM</td>
</tr>
<tr>
<td>5.1</td>
<td>Triangular Density Function</td>
</tr>
<tr>
<td>5.2</td>
<td>Weibull Distributions with Different Shape Parameters</td>
</tr>
<tr>
<td>5.3</td>
<td>Gamma Distributions with Different Shape Parameters</td>
</tr>
<tr>
<td>5.4</td>
<td>Different Time Advance Methods Used by Simulations</td>
</tr>
<tr>
<td>5.5</td>
<td>Simplified ROCSIM Flowchart</td>
</tr>
<tr>
<td>5.6</td>
<td>Opening Menu of ROCSIM</td>
</tr>
<tr>
<td>5.7</td>
<td>Input Screens for ROCSIM</td>
</tr>
<tr>
<td>6.1</td>
<td>Timing and Relationship of Validation and Verification.</td>
</tr>
<tr>
<td>7.1</td>
<td>Mine Layout Used in Example Runs of ROCSIM</td>
</tr>
<tr>
<td>7.2</td>
<td>Contour Map of RRS Values and Symbols Showing Location, Type and Cost of Roof Falls from One Run</td>
</tr>
<tr>
<td>7.3</td>
<td>Cleanup, Production and Total Delays in Run 1</td>
</tr>
<tr>
<td>7.4</td>
<td>Cleanup, Production and Total Delays in Run 2</td>
</tr>
<tr>
<td>7.5</td>
<td>Changes in Cleanup Delays Between the Two Runs Caused by Worsening Roof Conditions</td>
</tr>
<tr>
<td>7.6</td>
<td>Changes in Production Delays Between the Two Runs Caused by Worsening Roof Conditions</td>
</tr>
<tr>
<td>7.7</td>
<td>Changes in Total Delays Between the Two Runs Caused by Worsening Roof Conditions</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Rock Structure Rating (RSR)</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Rock Mass Rating System (RMR)</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Roof Rating System (RRS) for Roof Fall Prediction</td>
<td>16</td>
</tr>
<tr>
<td>7.1</td>
<td>A Summary of Output from Run 1 and Run 2 of ROCSIM</td>
<td>59</td>
</tr>
<tr>
<td>7.2</td>
<td>Comparison of Delays Caused by Varying the P Parameter</td>
<td>68-69</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

1.1 Statement of Problem

Roof falls account for a large percentage of cost associated with fatalities and injuries within the coal mining industry. Within the five year period of 1984–1988, 106 coal miners were killed and 4,135 were injured by falling roofs and ribs. The cost of these injuries and fatalities in coal mines in 1987 has been estimated by the Bureau of Mines to be $24,961,000 (Farley, 1989). Since about 53% of those injured were under supported top, there is considerable incentive to develop better, more cost effective roof support systems.

A major difficulty in designing mine openings in room and pillar coal mines is the need to assume ground control conditions in an area before it is mined. These assumptions are necessary to select the optimum opening widths, pillar sizes, and the type and spacing of the roof bolts. In most cases, the parameters used in the design equations are based either on information that has been interpolated from borehole data, or on average strength values based on experience. Either of these choices can force the designer to use an excessively high safety factor to overcome uncertainties in the input parameters, thus leading to an overdesigned mine with resulting high costs and low recovery. However, the lack of precise structural information can be partially overcome by using a probabilistic simulation of roof conditions that will be encountered. The simulation can estimate the probability of a roof fall occurring at a specific location within the mine. This permits roof support to be modified to mimic the changes that occur as
a mine progresses.

1.2 Approach/Methodology

Probabilistic simulation was selected for ground control simulation as it has the advantage of being able to deal with a large number of input variables that change in a random manner. In ground control applications it also facilitates calculation of the probability of the number, size, or cost of roof falls exceeding specified values. Probabilistic simulation also allows a designer to test competing mine layouts against assumed roof conditions to see how well they perform. By carefully selecting the parameters for the probability distributions, it is possible to simulate the cost of roof falls. Then, by comparing alternative roof control plans using the simulation, the option that provides the lowest overall cost with acceptable confidence level can be selected. To this end the simulation program ROCSIM (Roof control Optimization Cost SIMulation) was developed.

The occurrence and frequency of roof falls are related to the type of roof support, support density, geology, structural discontinuities, location in the mine, and elapsed time between mining and the roof fall. Ideally, for future planning in new mining areas, one should be able to take information from core drilling and logging and use this to input appropriate parameters into the probability distributions. To accomplish this a Roof Rating System (RRS) that takes into account geological and mining conditions and reduces them to a single numerical value offers the best solution. Using such a rating, specific parameters can be assigned to probability distributions to simu-
late the occurrence of roof falls for a particular mine within a given geologic setting.

Once the location of a roof fall is found, a cost can be estimated taking into account any resulting production delays as well as the direct cost of cleaning up the fall and resupporting the roof. By assigning a cost to a roof fall, comparison of competing roof support designs relative to their overall costs can be achieved.
II. LITERATURE REVIEW

2.1 Digital Simulation

Digital simulation is a technique where a probabilistic approach is used to solve complex problems. The approach is useful in situations where analytical solutions contain many changing variables. A simulation models a real system by selecting values from probability distributions and then observing the model in action to find its response to changing input conditions. For example, changes in roof conditions and support can be monitored through the use of the simulation.

Simulation can substitute for actual experiments with a prototype where experiments are impractical. Simulation approaches are desirable because of cost, disruption, or time constraints within the actual system. The technique permits fast evaluation of more configurations than could ever be achieved by testing the actual system.

Simulation has been shown to be a useful tool in finding the total cost of a roof control plan (Fraher and Haycocks, 1992). It was able to deal with highly variable conditions while producing values that could be analyzed for statistical significance (Fraher et al., 1990). Using total cost values in concert with the statistical analysis allows the designer to judge the reliability of the simulation output.

The structural conditions in a coal mine roof can be determined using numerical modeling such as the finite element method. This approach is site specific and is only attractive in a research environment where a very limited area is studied. It is also very computer intensive and requires a
very large amount of information to be able to assign properties to elements in the grid. Discontinuities in the rock are extremely difficult to model using finite elements due to the fairly random nature of these fractures. In addition, the strength of intact joints is most difficult to determine either in-situ or with lab testing. Numerical methods have no realistic application to large scale face design problems (Kripakov, 1981; Matsui and Takahashi, 1990).

There are many reasons that simulation have proven a popular method for solving problems, as noted by Law and Kelton (1982).

1. Most complex real-world systems cannot be modeled using a mathematical model that can be evaluated analytically.
2. Simulation provides estimations of performance of a system for new policies, parameters, or operating conditions.
3. Alternative proposed designs can be compared and evaluated.
4. Simulation allows experimentation without disturbing the real system,
5. The system can be studied using different time frames such as compressed time to speed up a study, or expanded time to observe the details of a study.

There are, however, some difficulties involved in simulation also noted by Law and Kelton (1982).

1. Simulation models are often expensive and difficult to develop.
2. Many simulation models contain stochastic parameters and only produce estimates of a system's true nature.
3. Simulation modeling can be used to compare alternatives, but not to find the optimal solution to a problem.

4. Simulation requires that the model must be a valid representation of the system for the output to be accurate.

5. Simulation runs may be lengthy and expensive in terms of computer resources.

A discrete simulation is one in which events happen at discrete times. This is in contrast to a continuous simulation where the state of the simulation changes smoothly, such as in the flow of material down a conveyor belt. Since roof falls happen only at a specific times, they are good applications for discrete simulation.

A discrete simulation consists of entities that undergo changes at a specific point in time. Each entity has a certain number of attributes that describe its state. The system state is a record of the condition of the entities that make up the system. An event is an instantaneous occurrence that alters the state of the system. Events can trigger activities, causing entities to perform some activity for a length of time.

The model of the system defines how all the entities, attributes, events, and their interrelationships behave. Any outside influence that can affect the model is termed the system environment. If there is no outside influence on the system it is said to be closed. In simulating roof falls, there are many outside influences that cannot be accounted for, so the system environment has a great influence on the performance of the simulation.
2.2 Roof Rating Methods

A great deal of work has been carried out in trying to rate the condition of the rock mass around a proposed mine opening. Most of these systems were not originally developed for coal mining. Of the methods available, there are four major ones used to rate rock. These are Roof Structure Rating (RSR), Rock Quality Designation (RQD), Q-system, and the Rock Mass Rating (RMR). Within the RMR system, there are many variations to take into account parameters that other investigators have found important for specific mining applications.

2.2.1 Rock Structure Rating

The Rock Structure Rating (RSR) concept was developed by Wickham, Tiedemann, and Skinner (1974). It is a system used to assess the stability of hard rock tunnels. The system is based on three weighted parameters A, B and C, that when summed together can add up to a possible rating of 100. The A parameter is an appraisal of the rock structure, the B parameter considers the effect of the discontinuities present on the tunnel, and the C parameter takes into account the effect of water inflow into the tunnel. The methods to find parameters A-B are shown in table 2.1. The RSR can be correlated with tunnel support requirements (Wickham et al., 1974).

2.2.2 Rock Quality Designation

Rock Quality Designation (RQD) was developed as an index property of drill cores (Deere et al., 1967). This system evaluates the amount of sound
### Table 2.1 - Rock Structure Rating (RSR)

**Parameter A: General Geology**

<table>
<thead>
<tr>
<th>Basic Rock Type</th>
<th>Hard</th>
<th>Medium</th>
<th>Soft</th>
<th>Decomposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Igneous</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Metamorphic</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Sedimentary</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Slightly Faulted or Folded</th>
<th>Moderately Faulted or Folded</th>
<th>Intensely Faulted or Folded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td>30</td>
<td>22</td>
<td>15</td>
</tr>
<tr>
<td>Type 2</td>
<td>27</td>
<td>20</td>
<td>13</td>
</tr>
<tr>
<td>Type 3</td>
<td>24</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>Type 4</td>
<td>19</td>
<td>15</td>
<td>10</td>
</tr>
</tbody>
</table>

**Parameter B: Joint Pattern, Direction of Drive**

<table>
<thead>
<tr>
<th>Strike Perpendicular to Axis</th>
<th>Strike Parallel to Axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direction of Drive</td>
<td>Direction of Drive</td>
</tr>
<tr>
<td>Both With Dip</td>
<td>Both Dip of Prominent Joints</td>
</tr>
<tr>
<td>Dip of Prominent Joints</td>
<td>Flat Dipping Vertical</td>
</tr>
<tr>
<td>Both Against Dip</td>
<td>Dipping Vertical</td>
</tr>
<tr>
<td>1. Very Closely Jointed, &lt; 2 in.</td>
<td>9 Flat 11 Dipping 13 Vertical</td>
</tr>
<tr>
<td>2. Closely Jointed, 2-6 in.</td>
<td>13 Flat 16 Dipping 19 Vertical</td>
</tr>
<tr>
<td>3. Moderately Jointed, 6-12 in.</td>
<td>23 Flat 24 Dipping 28 Vertical</td>
</tr>
<tr>
<td>4. Moderate to Blocky, 1-2 ft.</td>
<td>30 Flat 32 Dipping 36 Vertical</td>
</tr>
<tr>
<td>5. Blocky To Massive, 2-4 ft.</td>
<td>38 Flat 40 Dipping 45 Vertical</td>
</tr>
<tr>
<td>6. Massive, &gt; 4 ft.</td>
<td>40 Flat 43 Dipping 45 Vertical</td>
</tr>
</tbody>
</table>

Dip: flat 0-20°; dipping 20-50°; and vertical 50-90°

**Parameter C: Groundwater, Joint Condition**

<table>
<thead>
<tr>
<th>Anticipated Water Inflow (gpm/1000 ft.)</th>
<th>Sum of Parameters A + B</th>
<th>Joint Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(gpm/1000 ft.)</td>
<td>13 - 44</td>
<td>45-75</td>
</tr>
<tr>
<td>None</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>Slight, &lt; 200 gpm</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td>Moderate, 200-1000 gpm</td>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>Heavy, &gt; 1000 gpm</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

Joint Condition: Good = tight or cemented; Fair = slightly weathered or altered; Poor = severely weathered, altered, or open

After Wickham et al. (1974)
core in pieces that are greater in length than four inches measured along the centerline. The sum of the length of these pieces is divided by the total length of the core run, and this number (in percent) is the RQD of the core.

By taking into account the soundness of the core and the spacing of discontinuities and joints, RQD provides information on rock mass continuity. Since only cores over 4 in. minimum length are considered highly jointed rocks do have a low RQD. Unsound rocks that are weathered, altered, or in weakened condition will rarely produce large pieces of core and so will tend to be excluded. This causes RQD to be a measure of rock quality as well as one of joint frequency.

The RQD of a rock mass has been correlated with the strength of the rock and its support requirements within hard rock mines (Deere and Deere, 1989). The attractiveness of this method is its simplicity, and its wide acceptance in this country and overseas. But more importantly it is an input variable that is used by both the RMR and Q-system.

2.2.3 Q-system

The Q-system was developed by the Norwegian Geotechnical Institute as a numerical rating of the roof conditions that ranges from 0.001 to 1000 (Barton et al., 1974). This rating is based on six different parameters: RQD, number of joint sets, roughness of the most unfavorable joint or discontinuity, degree of alteration along the weakest joint, water inflow, and stress conditions at the site. The equation for Q is shown in equation 2.1.
\[ Q = \frac{RQD}{J_n} \times \frac{J_r}{J_a} \times \frac{J_w}{SRF} \]

Q - Q rating
RQD - rock quality designation
\( J_n \) - joint set number
\( J_r \) - joint roughness number
\( J_a \) - joint alteration number
\( J_w \) - joint water reduction number
SRF - stress reduction factor

(2.1)

The Q value can be used to find support requirements and maximum span that can exist unsupported. The system is widely used in tunneling and hard rock mining. It is not used in coal mining; however, it provides some guidance on how to construct a rock rating system.

2.2.4 Rock Mass Rating

Rock Mass Rating (RMR) is probably the most widely used rock rating system in America, and has been modified over time as more case histories are gathered (Bieniawski, 1989). It is used in hard rock mining, tunnel design, and the design of slopes and foundations. Like the other systems, it is an empirical classification of a rock mass based on case histories.

The value of RMR ranges from 0 to 100 with different ranges earning ratings from “very good rock” to “very poor rock”. Table 2.2 shows the RMR system. This table also includes the adjustments made for the orientation of discontinuities in relation to the direction of the mine opening.
Table 2.2 – Rock Mass Rating System (RMR)

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Range of Values</th>
<th>For this low range, uniaxial test is preferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strength of intact rock material</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point-load strength index (MPa)</td>
<td>&gt; 10</td>
<td>4-10</td>
</tr>
<tr>
<td>Uniaxial compressive strength (Mpa)</td>
<td>&gt; 250</td>
<td>100 - 250</td>
</tr>
<tr>
<td>Rating</td>
<td>15</td>
<td>12</td>
</tr>
<tr>
<td>Drill Core Quality RQD (%)</td>
<td>90 - 100</td>
<td>75 - 90</td>
</tr>
<tr>
<td>Rating</td>
<td>20</td>
<td>17</td>
</tr>
<tr>
<td>Spacing of discontinuities</td>
<td>&gt; 2 m</td>
<td>0.6 - 2 m</td>
</tr>
<tr>
<td>Rating</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>Condition of discontinuities</td>
<td>Very rough surfaces</td>
<td>Slightly rough surfaces</td>
</tr>
<tr>
<td></td>
<td>No continuous</td>
<td>Separation &lt; 1 mm</td>
</tr>
<tr>
<td></td>
<td>Unweathered wall rock</td>
<td>Slightly weathered walls</td>
</tr>
<tr>
<td>Rating</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>Groundwater</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inflow per 10 m tunnel length (L/min)</td>
<td>None</td>
<td>&lt; 10</td>
</tr>
<tr>
<td>Rating</td>
<td>0</td>
<td>&lt; 0.1</td>
</tr>
<tr>
<td>Joint water pressure Major principal stress</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>General conditions</td>
<td>Completely dry</td>
<td>Damp</td>
</tr>
<tr>
<td>Rating</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>Adjustments for Strike and Dip</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orientations of Discontinuities</td>
<td>Very Favorable</td>
<td>Favorable</td>
</tr>
<tr>
<td>Ratings</td>
<td>0</td>
<td>-2</td>
</tr>
</tbody>
</table>

(Bieniawski 1989)
2.3 Geological Methods to Predict Roof Falls

Many researchers have taken the approach of correlating roof falls with specific predominant geological conditions existing at roof fall sites. In many ways this is just a sophisticated replica of an experienced miner knowing what conditions lead to roof falls.

Moebs (1977) related ground control problems in coal mines in southwestern Pennsylvania to changes in the geological structure within the mine. In the region of the study, most roof falls were in the areas beneath small sandstone-filled channels and scours. The resulting slickensided zones presented very unstable roof conditions. These sandstone channels could be inferred from drill core logs, and the sandstone isopach maps generated from these cores. He also found that even poorly formed roof could be held up if it was devoid of irregular structures or had no slickensided zones within it.

An alternative approach was to divide problem coal mine roofs into four general types to aid in the prediction of future roof fall activity (Hylbert, 1976; Hylbert, 1978). It was determined that laminated shales could be subdivided into two classes, types A and B. In type A there is shale that has slip surfaces, but they do not run parallel to the bedding plane of the shale. If the number of discontinuities is high, this type of roof will be very difficult to support. In type B, the slip surface is parallel to the bedding plane and causes the roof to fall before support can be placed. Type C roof condition is flat to rolling sandstone with shale in pockets. Type D roof is an exaggeration of type C roof with zones of major sandstone rolls. By using
the same methods as Moebs (1977) the trend of these geological conditions was determined in advance of mining. Correlation of roof conditions to the occurrence of roof falls within the mine in the study were not conclusive. Lineament studies, however, could be correlated to roof control problems (Hylbert, 1978).

Patrick and Aughenbaugh (1979) classified roof falls as to Dome type IA, Arch type IB, Minor IIA, Sloughing or rashing IIB. They also collected statistical data on frequency and extent of falls in relationship to the geological conditions in the area.

A study of overlay maps at the Beckley mine in West Virginia correlated geologic data and roof fall conditions within the mine (Ealy et al., 1979). It was found that roof fall frequency could be related to the composition and thickness of the roof rock. Falls occurred much more frequently in thinly laminated shale with interbedded sandstone and fall sizes became bigger as the immediate roof rock thickness increased. Roof fracture zones underground can be related to surface drainage patterns and their severity is dependent on the roof rock type. It was also found that as the extraction ratio increased, the roof fall frequency increased. A stability number could be assigned for an area based on the number of roof falls per acre of mine area in the best region, divided into the number of roof falls in the total area in question. Numbers ranged from 1 to 24 in the study with a high number signifying unstable conditions. These values were used to generate hazard maps to show areas of maximum roof instability.

Chugh and Silverman (1982) published guidelines for premining in-
vestigations of a site. They found that roof falls causes can be grouped into five subheadings: geological, geotechnical properties, hydrogeologic factors, in situ stresses, and mining factors. A detailed analysis is performed and maps are created showing elevation of coal seam, overburden thickness, thickness of anchor zones above coal seam, rose diagrams for joints and fractures, roof shale thickness, number of beds within 15 meters of the coal seam, underclay thickness, lithofacies change maps, and distance to the first thick sandstone. In addition, lineament data is examined for its effect on water inflow rates and on fracture density. When all the data has been collected, it is the responsibility of the planner to select the parameters that he finds most important to produce a set of hazard maps. These can be overlaid with each other to determine critical areas of roof instability. The techniques used in this research would appear to have applicability to large coal mines where the extra engineering could be justified.
III. Development of Roof Rating System

The development of a realistic Roof Rating System (RRS) is essential to the successful simulation of roof fall activities in a coal mine face area. A RRS provides a systematic method of bringing into the program a single numerical value that will describe geology around the opening and the important operating parameters. The rating system is designed so that different people rating the same area would get similar results, but is not so detailed that excessive input data is required.

3.1 Important Characteristics

To construct the RRS the Rock Mass Rating (RMR) system was modified specifically for coal mine roof conditions by considering only those factors which had the greatest influence on the size and timing of roof falls (Cox, 1974; Ealy et al., 1979; Moebis 1977). RMR was selected as a basis for the RRS due to its widespread acceptance and ability to produce a single output value. The sections of RMR designed for hard rock tunneling uses was modified. The RRS rating is structured so that the factors can have an individual rating from −2 to +2, and the final aggregate rating lies within this same range. This provides one single numerical rating for each roof location during the execution of the simulation. The RRS system is shown in table 3.1.

The effects of vertical and horizontal stresses are not included directly in the rating, because stress effects are incorporated in the probability distributions. Highly stressed areas will have far different scale and shape pa-
<table>
<thead>
<tr>
<th>Factors/Rating</th>
<th>Very Good (+2)</th>
<th>Good (+1)</th>
<th>Fair (0)</th>
<th>Poor (-1)</th>
<th>Very Poor (-2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immediate Roof Rock (IRR)</td>
<td>Massive, clean, smooth gray sandstones</td>
<td>Some massive hard shales</td>
<td>Interbedded shales and sandstones; crystallized sandstones and conglomerates</td>
<td>Thinly laminated shale beds</td>
<td>Slumps, deposits, channel scours, fire clays, kettlebottoms, slickensides and pinchouts</td>
</tr>
<tr>
<td>Root Structure Location (RSL)</td>
<td>Shafts, portals, between pillars</td>
<td>L- intersections</td>
<td>T- intersections, entry crosscuts</td>
<td>Four-way intersections</td>
<td>Face area, straight headings (within 20 ft. outby)</td>
</tr>
<tr>
<td>Distance to First Competent Strata (DFCS)</td>
<td>&lt; 5 ft. (&lt; 1.5 m)</td>
<td>5 - 10 ft. (1.5 - 3.3 m)</td>
<td>10 - 25 ft. (3.3 - 7.6 m)</td>
<td>25 - 35 ft. (7.6 - 10.7 m)</td>
<td>&gt; 35 ft. (&gt; 10.7 m)</td>
</tr>
<tr>
<td>Time Before Placement of Support (TS)</td>
<td>&lt; 2.5 hours</td>
<td>2.5 - 5 hours</td>
<td>5 - 8 hours</td>
<td>8 - 12.5 hours</td>
<td>&gt; 12.5 hours</td>
</tr>
<tr>
<td>Ground Water (GW)</td>
<td>Completely dry</td>
<td>Mostly dry or moist; rock not sensitive to water</td>
<td>Moist to moderately wet; rock not sensitive to water</td>
<td>Very wet; dripping rock sensitive to water</td>
<td>Flowing; rock very sensitive to water</td>
</tr>
<tr>
<td>Drill Core RQD(^2) of Roof (RQD)</td>
<td>90 - 100%</td>
<td>75 - 90%</td>
<td>50 - 75%</td>
<td>25 - 50%</td>
<td>&lt; 25%</td>
</tr>
<tr>
<td>Uniaxial Comp. Strength of Intact Roof Rock (SIR)</td>
<td>&gt; 36,250 psi (&gt; 250 MPa)</td>
<td>14,500 - 36,250 psi (100 - 250 MPa)</td>
<td>7,500 - 14,500 psi (50 - 100 MPa)</td>
<td>3,625 - 7,250 psi (25 - 50 MPa)</td>
<td>&lt; 3,625 (&lt; 25 MPa)</td>
</tr>
</tbody>
</table>

1 Based on study by Cox, R.M., 1974
2 Bieniawski, Z.T., 1984
rameters for their probability distribution than areas with low stresses. These different shape and scale parameters affect the shape of the density function, and this changes the time of the fall as well as its shape. Traditionally, stress was not included by other researchers in their rating systems. Experience has shown that geology, structure and mining conditions around an opening are stress fields. In coal mining where stresses, particularly horizontal stresses, cause increases in roof failures, these stresses must be accounted for in the probability distributions.

3.1.1 Assignment of Rating Values

Of the seven separate ratings that make up the RRS, four refer to the geological conditions in the immediate roof which create the majority of roof falls. The remaining three factors are mining related conditions that have an effect on the stability of the mine opening. If there is no competent stratum to tie a roof bolt into, it will not be as effective as a well-anchored bolt. Wet conditions could cause the physical strength of the roof to deteriorate over time and eventually to fail. The final factor, the location in the mine, is critical to the stability of the opening. A four-way intersection will be less stable than a location in a cross cut between the intersections.

The first factor in the RRS is the Immediate Roof Rock (IRR). This is the rock within 20 feet of the top of the coal seam. As documented previously, many researchers have found a clear correlation between the type of rock in the roof and the frequency of roof falls. The best rating is given to massive, clean, smooth gray sandstones, and the worst rating to roofs that have
slumps, deposits, channel scours, fire clays, kettlebottoms, slickensides and pinchouts. This is the most subjective rating of all the categories, because the rocks that make up a roof are frequently difficult to numerically define. A person using ROCSIM can compare roof condition within their own mine to the example roofs shown within RRS to determine the − 2 to + 2 rating which would be appropriate.

The second rating is given to the Roof Structure Location (RSL). This varies from optimum near a shaft pillar, to minimum when within 20 feet of a working face. This value does not have to be entered by the user because the program determines where the cut is located, assigning this number automatically.

Distance to First Competent Strata (DFCS) is the third factor in the rating. This rating estimates how effectively the bolts will function based on a suspension concept, and how much of the immediate strata the bolts must support. This is defined as optimum at a distance of less than 5 feet to the competent strata, to worst for a distance greater than 35 feet.

The fourth factor, Time Before Placement of Support (TS) is used because research has shown that the time between mining and roof bolt placement is critical to the effectiveness of the bolts (Cox, 1974). The user does not enter this value because it is stochastically generated by the program during the simulation run.

Ground Water condition (GW) is the fifth factor, and it has an effect on the long term stability of the roof. Many types of coal mine top degenerate badly in the presence of moisture. This is another subjective part of the
RRS, and the user must rate conditions from completely dry to flowing water, taking into account the sensitivity of the roof rock to moisture.

The sixth factor is the Rock Quality Designation (RQD), as it has previously been used in a variety of rating systems such as the RMR and the Q system. The RQD is not as widely used in sedimentary rocks due to the difficulty in recovering intact cores. Since RQD is the percentage of the core with centerline lengths greater than 4 inches, it is critical that fractures caused by the drilling are differentiated from those occurring naturally. Cores whose size is greater than HQ (2.5 inches) should be used and careful drilling techniques observed. The cores must then be logged immediately to prevent them from drying out and disintegrating. Care is needed in shales where the drilling action will cause the core to “disk.” These breaks must be checked to see if they are recent, and should therefore be disregarded, or are from older fractures (Deere and Deere, 1989).

The final factor is the intact strength of the roof rock. This has a direct impact on the design and effectiveness of the roof support. The strength values were derived from earlier work done at Virginia Tech (Townsend, 1975).

3.2 Implementation

The technique for implementation of the RRS is to use as many known specimens from drillholes or other types of sampling as possible. This information is then interpolated for each new cut during the program run. The TBS (Time Before Support) and RSL (Roof Structure Location) information
are then added to this interpolated data to produce a RRS value for each individual cut that is made.

The RRS values are assigned to the sampling locations by the computer which asks how many samples are to be taken, and a series of five questions about these points. These answers are used to award an RRS value to each sampling location.

The interpolation subroutine uses the nearest two sample locations to find the value at a particular point, assuming a linear change between the two locations. If the cut is taken before it gets to the first sample location, the cut value is assumed to be equal to this first sample location. Similarly the cuts that occur after the last sample location are given the value of the final sample.

3.3 Use of the RRS Values

ROCSIM uses the RRS values in addition to roof bolt density and opening widths to assign parameters to the probability distributions. This permits geological and some mining factors to affect the selection of the probability parameters to be used. Since only the RRS values for each cut change during the program run, the RRS will be the sole determinant of which cut will have larger falls and when these falls will occur. By using the RRS values, the stochastic tendencies are better controlled than by using time to roof fall distribution and roof fall size distributions to determine when a fall will happen. This single parameter selection approach gives each cut in the mine an equal chance of roof fall occurrence.
The values of the parameters for the probability distributions come from roof falls in the Midwest (Patrick, 1978). They provide a good starting point for any simulation, and can be modified to better fit the available data.
IV. PROGRAM FORMULATION

Determining the costs involved in both supporting a section of coal mine roof and cleaning up any roof falls is critical to the success of the proposed simulation. Using these costs it is possible to make an informed decision on which support alternatives are best in terms of total ground control costs. The cost of roof falls must be allocated by a method reflecting both where the roof fall happens and its effect on coal production. The approach selected divides the roof fall cost into roof falls that cause cleanup delays and those that cause production delays. Using this separation, it is possible to structure the simulation program so that the program will calculate the costs due to each type of fall.

It is assumed in using ROCSIM that the pillars have been designed correctly using a pillar design method such as those shown in Appendix A. If the pillars gradually fail over time, adjustments must be made in the opening widths to compensate for the increase in roof span caused by this degradation.

The original cost of installing the bolts is included in the program. The user enters the preferred spacing, but ROCSIM calculates the spacing needed to fill the opening width during the program run. The spacing that ROCSIM uses is shown in figure 4.1. The program then finds the bolt density that is defined as the number of square feet of roof held up by each bolt. The higher this number, the less effective the support will be.

The cut sequencing used by ROCSIM is shown in figure 4.2. The proper depth of cut is found that will give an even number of cuts based on open-
Figure 4.1 - How the Roof Bolt Pattern is Defined
Figure 4.2 - Cut Sequence Used in ROCSIM
ing width and pillar size. Within this section, each cut's location is logged, and is used later to find the location of roof falls as well as their distance from the working face.

Both types of falls must be cleaned up and resupported. The volume of the fall is used to determine the time needed to clean up the fall, and the area of the fall is used to calculate the number of bolts needed to rebolt the area. A diagram showing the assumed shape of the roof fall that ROCSIM uses is shown in figure 4.3.

4.1 Cleanup Costs

Cleanup costs are those costs that occur due to a roof fall that must be cleaned up, but does not cause a halt in production. If a fall is not on a main transportation haulageway, or is more than 200 feet away from the working face, it is considered a cleanup cost. The areas in the mine that are associated with production delay are shown in figure 4.4.

The user of ROCSIM inputs the rate of cleanup activity as well as the per ton cost of the cleanup, so that the cost of cleaning up the roof fall can be calculated. The cost includes the amount of money needed to clean up the fall, which is calculated by determining the weight of a fall multiplied by the per ton cost of cleaning up the fall. The surface area of the roof fall is determined, and this is used to find the resupport cost for the fall. The number of bolts is then calculated, and since the user has specified the installed cost per bolt, a bolting cost for the area is found. Total cost is then calculated, and this amount is stored as an attribute within the record that
Figure 4.3 - Roof Fall Shape Used by ROCSIM

L - Length of roof fall
W - Width of roof fall
T - Thickness of roof fall
Figure 4.4 - Areas that Cause a Production Delay in ROCSIM
contains all the important data about the roof fall.

4.2 Production Delays

The delays caused by falls close to the face or on the main haulage ways are termed production delays. No production is allowed until the fall is cleaned up and resupported. Any time required to clean up the roof fall is deducted from the time available for mining. The cost of the equipment downtime is added to the costs that are incurred by a cleanup delay.

The calculation is the same as used for the cleanup delay, but includes the additional cost of the production downtime. The time needed for resupport is found, and this is added to the cleanup delay. This total downtime is multiplied by the cost of an hour of the production shift, and this is added to the cleanup costs to produce the cost of a production delay. The fall record is stored in a manner similar to the cleanup delay record.
V. MODEL DEVELOPMENT

5.1 Model Design Consideration

To simulate any system a number of steps must be followed. The problem must be defined and the objectives of the simulation outlined. Then model specifications are selected and data from real systems collected. The program must be constructed and then verified and validated. The simulation must be run and the output analyzed. Finally the results of the simulation must be documented and recommendations issued.

The definition of the problem to be solved is a critical task for the modeler. In a ground control situation, the main problem is to achieve the best ground control possible while taking into account the total cost of the plan. The total cost will include the cost of the support and the costs associated with roof falls that happen while using this support.

Total cost is a difficult subject when one is involved in a situation such as roof falls where injury and death can occur. The occurrence of injury is very difficult to simulate because it happens at a fairly low rate. In addition the cost involved due to an injury can range from a relatively small amount to hundreds of thousands of dollars. The cost of a death can exceed a million dollars for an underground coal mine. These facts make it almost impossible to incorporate injuries into a simulation.

5.2 Generating Random Numbers

There are a variety of ways to generate random numbers from within a program. For the purpose of a simulation, the number must come from a
uniform distribution with a range from 0 to 1. To be effective, the numbers must act as if they are from a uniform distribution.

A computer program uses a pseudo-random generator that always produces the same stream of numbers if started with the same seed value. This is a result of the method used to calculate the random number where the previous random number determines what the next random number will be. This is a useful feature because it allows a simulation program to be run with different input parameters, but using the same stream of random numbers so that a comparison can be made between the runs. This takes any random variation out of the comparison process.

The generator used is built into Turbo Pascal. Since it uses 32-bit math it is capable of generating every integer between 1 and more than 4 billion before it repeats (Rolny, 1990). The equation that Turbo Pascal uses is shown in equation 5.1.

\[
\text{seed} = ((\text{Multiplier}) \times \text{seed}) + 1 \mod 4294967296
\]

\[
\text{Multiplier} = 134775813 
\]

The program generates this seed value then divides it to get a random number in the range of 0 to 1.

Many tests were made on the randomness of the random number generator including Chi 1-D, 2-D and 3-D tests. The built in generator would fail on a very rare basis in the higher dimensions, but with the fairly small number of samples that ROCSIM uses, this does not create a problem.
5.3 Generating Random Variates

Once the random number is generated, the next step is to produce random variates from the probability distributions. This is carried out using one of two general techniques, the first is to take the inverse of the function that describes the probability, or if no closed form solution exists for this inverse, an indirect method must be used.

ROCSIM is capable of generating random variates from uniform, triangular, normal, lognormal, gamma, Weibull, exponential and beta distributions. Preferentially the program uses the Weibull, gamma and triangular distributions to generate random variates. Other distributions are included, and can be utilized if another type of distribution is needed. Since the other distributions are optional only the three that are used will be described in detail.

The triangular distribution function is simple to use and to program. It is used when little information is available about the system under study. It is an improvement on the uniform distribution, where all that is known is the lower and upper range of the data, and any value has an equal chance of being selected. The triangular distribution adds a most likely value. An example would be where a person knows that a job is never finished in less than an hour, never takes longer than four hours, but usually it takes two hours. This would be a perfect application for the triangular distribution. The equation for a triangular distribution is shown in equation 5.2. It is simple to invert these equations and solve for \( X \) by setting each of the equations equal to its area. Since these are probability density functions,
the total area equals 1.

\[
    f(x) = \begin{cases} 
        \frac{2(x - a)}{(b - a)(c - a)} & \text{if } a \leq x \leq c \\
        \frac{2(b - x)}{(b - a)(b - c)} & \text{if } c \leq x \leq b 
    \end{cases}
\]

\[a \text{ - lowest value} \]
\[c \text{ - most likely value} \]
\[b \text{ - highest value} \]

(5.2)

A drawing showing the density function is shown in figure 5.1. The solution to this inversion is shown in equation 5.3.

\[
x = \begin{cases} 
    a + \sqrt{r (c - a) (b - a)} & \text{if } r \leq \text{center} \\
    b - \sqrt{(1 - r) (b - c) (b - a)} & \text{if } r > \text{center} 
\end{cases}
\]

\[r = \text{random number } 0 \rightarrow 1 \]
\[\text{center} = \frac{(c - a)}{(b - a)} \]

(5.3)

The actual code can be seen in the math unit of the ROCSIM program in function trian(a,c,b) in Appendix B on page 99.

The Weibull probability distribution is used in ROCSIM to find the time until a roof fall takes place. It can mimic a variety of other distributions depending on the value of its shape parameter. It is commonly used to model the time it takes to complete a task, and the time of failure of a piece of equipment. The Weibull density function is shown in equation 5.4. The
Figure 5.1 - Triangular Density Function

\[ f(x) = \frac{2}{(b - a)} \]
inversion of the Weibull has a closed form solution, which is one of the reasons that it is popular for use in simulations. A graph of the Weibull distribution is shown in figure 5.2. The solution to the inversion is shown in equation 5.5.

\[ f(x) = \alpha \beta^{-\alpha} x^{-\frac{\alpha}{\beta}} \exp\left(\frac{-x}{\beta}\right) \]

\( \alpha \) – shape parameter \hspace{1cm} (5.4)
\( \beta \) – scale parameter

\[ X = \beta (- \ln U)^{\frac{1}{\alpha}} \]

\( U \) – random number 0 \( \rightarrow \) 1

The Weibull is popular due its high speed of execution and versatility within a simulation program.

The most important probability distribution used by ROCSIM is the gamma distribution. It has proven to be very useful in simulating the size of roof falls in coal mines (Patrick, 1978). The problem with the gamma is the difficulty in generating it. The equation for the gamma is shown in equation 5.6.

\[ f(x) = \frac{\beta^{\alpha} x^{\alpha-1}}{\Gamma(\alpha)} \exp\left(-\frac{x}{\beta}\right) \]

\[ \Gamma(\alpha) = \int_0^\infty t^{\alpha-1} \exp(-t) \, dt \]

\( \alpha \) – shape parameter
\( \beta \) – scale parameter

Equation 5.6
Figure 5.2 - Weibull Distributions With Different Shape Parameters
The biggest advantage of the gamma distribution is the wide variety of shapes it can take with different shape and scale parameters. A graph showing the gamma distribution with different shape parameters is shown in figure 5.3. The generation of a random variate from the gamma distribution is fairly complicated. There are two algorithms used, one for $\alpha \leq 1$ and another for $\alpha > 1$. For the $\alpha \leq 1$ case the algorithm used is an acceptance-rejection technique by Ahrens and Dieter (1974). For $\alpha > 1$, a modified acceptance-rejection technique developed by Chang (1977) called the GB algorithm is used. These algorithms are very complex and are shown in the text Simulation Modeling & Analysis (Law and Kelton, 1982). The method of coding these algorithms is shown in Appendix B on page 100 in the function gamma(alpha,beta).

The remaining algorithms used are very similar to these first three, with different coding. Using the built in random number generator and these algorithms, a fairly fast output of random variates is achieved. The additional probability distributions in reserve allow the flexibility to model almost any new situation that might occur within a mine.

5.4 Time Advance Methods

To model the passage of time in a simulation, one of two methods is used to set the internal clock for the simulation. The first method is to advance time by a fixed increment, and the other is to use next event time advance.
Figure 5.3 - Gamma Distributions With Different Shape Parameters
In fixed time advance, the clock is set forward a fixed increment and then the system is scanned to see if any event has occurred within this time. All events that occur within the time increment is considered to have occurred simultaneously. An illustration of this method is shown in fig 5.4. The difficulty is in selecting the proper time increment, so that the simulation can run efficiently while still having the ability to place the event with acceptable accuracy.

The second method is to use next event time advance, also illustrated in fig 5.4. In this method the simulation clock is set to the time of the next event that will occur. The simulation skips over periods when nothing is happening. To use this method, all the scheduled events must be stored in a list in chronological order so that the program can retrieve them. This method has the advantage of not wasting computer time when no events occur within a time interval. It also allows for the more accurate placement of when events occur. This method is used in the rock fall simulation program to store the widely varied spacing of roof fall events.

Simulation normally deal with the next event in either one of two ways: event scheduling or activity scheduling. In event scheduling the next event is taken from an events list, and then sent to the proper routines in the program with no test performed on the activity. In activity scheduling, tests are done on the next event, and then based on these tests another activity is performed until the action is completed. This is then repeated with the next eligible event on the list until the simulation is completed. This is the method ROCSIM uses to deal with events.
Fixed-increment time advance

$e$ - events
$t$ - time

Next-event time advance

(after Law 1991)

Figure 5.4 - Different Time Advance Methods Used by Simulations
5.5 Linked-Allocation List Storage

The storage of lists within a simulation is one of the most critical
tasks of the simulator. If an inefficient storage system is used, the simula-
tion will execute slowly and will not be capable of storing the large quantity
of data needed by the program.

One possible approach is to use sequential-allocation storage of the
list. In this method the records, which each contain a number of attributes,
are stored in a physical location that is adjacent to the next record in the
list. So if a list were stored using the time of an event, the array would be
arranged so that the first record has the earliest time, the second record has
the next time, and so on. When a new record is added, the entire list must
be rearranged to accommodate the new record. This resorting can become
extremely time consuming with large lists, no matter how efficient the sort-
ing method used.

In ROCSIM, a linked-allocation method of list storage is used. This is
the same method used by major simulation programs due to its efficient use
of storage space and speed in handling a large number of records. In this
method, a new record is stored in the first available storage location, but its
location is shown in the list by using a pointer. This pointer shows the loca-
tion of either the predecessor link (previous record), or it shows the succes-
sor link (next record) in the list. This pointer array is one dimensional, and
it is the only array that must be modified when a new record is added. Each
list has head and tail pointers that show what records is first and last with-
in the list, then the pointer array provides the next record in the list. This
allows the list to be scanned from the head of the list or from the tail, allowing the programmer go through the list in the most efficient manner. The list can be based on any of the attributes of the record, so for example one list can be shown based on the time of a roof fall, and another on the cost of cleaning up the roof fall.

A major advantage of linked-allocation storage is that more than one list can be stored in each array. Since the records can be stored anywhere in the array, records from different lists can be intermingled, with a different pointer array keeping track of the situation. This method allows each list to take up only as much room as needed, with the other storage locations available for the storage of other lists. This master array in ROCSIM is as large as the computer memory will allow for the maximum storage of records.

5.6 Method of Programming

5.6.1 Programming Language

A specialized package such as SLAM or GPSS could be used to simulate the roof falls that occur, but the high purchase price would discourage its use by casual industrial users. Simulation packages are also not designed for the specialized needs of the mining industry. To serve the future end user, a general purpose programming language was selected, allowing ease of program modification, and permitting the free distribution of ROCSIM as an executable file and source code.

Turbo Pascal by Borland was selected because of its widespread use.
Pascal is not as efficient as C, but has the advantage of being much more readable to programmers. Turbo Pascal also has an extremely user friendly interface that makes editing and compiling a program a much simpler task than programming in a language such as Fortran or C. Its high compiling rate allows many programming variations to be tried in a short period of time. This is useful when modification to the program are necessary to better meet a particular user's needs. The source code is provided with the program as well as the executable version. Allowing any programmer familiar with Pascal to modify the program.

A difficulty in using Pascal was that routines originally written in Fortran had to be totally rewritten in Pascal. The list manipulation section of ROCSIM is a heavily reworked version of one written in Fortran (Law and Kelton, 1982). The advantage of this rewrite was that a "top down" programming technique was used. Top down programming is a structured method of programming, designed to make it easier for another programmer to interpret and modify the program. It also produces a much more robust program that is easier to test. The routines used to generate the random variates were developed from scratch using written description of algorithms. A simplified flowchart showing the general layout of ROCSIM is shown in figure 5.5.

The list manipulation section of the program is critical to its execution due to the large number of records that must be stored and manipulated. The list must be stored sequentially to allow the removal of the next roof fall from the event list. It also must be able to store large numbers of roof
Figure 5.5 - Simplified ROCSIM Flowchart
fails so that the simulation can run for an extended period of time. Pascal has the advantage of using dynamic memory allocation to store as many records as the available memory will permit.

5.6.2 User Interface

ROCSIM is a menu driven system that allows easy access to the various modules of the program. An example of the opening screen is shown in figure 5.6. A module can be accessed either by moving the cursor down or by inputting the number of the appropriate selection.

The input section of the program allows the user to interactively fill in the data that the program requires. A user can start with a default data file, or he can use a previously created file and modify any values that are different. A representation of the two input screens is shown in figure 5.7. It is not necessary to input the data sequentially; it can be entered in any order. Once entered it can be changed at any time. After the data file has been modified, the program is then run using the newly created input file.

A status indicator is visible during the execution of the program to provide information on the room width, bolt spacing, number of cuts made, and the hours lost to roof falls. This gives the user confidence that the program is actually executing and provides some information on how the changes in conditions are effecting the frequency of roof falls and the delays that they cause.

Help screens are incorporated into the program to guide the user in running and understanding ROCSIM. By using the F1 key, an appropriate
ROCSIM

Roof-control Optimization Cost SIMulation

1. Input/Edit Data
2. List Input Files
3. Run Program
4. List Output Files
5. Exit Program

INPUT or EDIT data using screen editor

F1 - Help

Figure 5.6 - Opening Menu of ROCSIM
Input the number of SHIFTS that the simulation will run: 500
Input the number of HEADINGS: 9
Input the LOWER BOUND of HEADING WIDTH (ft.): 15.0
Input the UPPER BOUND of HEADING WIDTH (ft.): 25.0
Input the number of INCREMENTS between these two widths: 3
Input the SEAM THICKNESS (ft.): 6.0
Input the number of FEET ADVANCED per cut: 20.0
Input the number of CUTS per shift: 15.0
Input the COST of a production shift ($): 10000
Input the DENSITY of the COAL (PCF): 80.0
Input the TONS of ROOF FALL that can be cleaned in 1 hr.: 5.0
Input the COST of CLEANING a TON of ROOF FALL MATERIAL: 500.

Input the PILLAR LENGTH (ft.): 80
Input the PILLAR WIDTH (ft.): 80
Input the BOLT LENGTH (ft.): 5.0
Input the COST per BOLT including installation ($/bolt): 15.0
Input the BEGINNING spacing between bolts (ft.): 3.5
Input the MAXIMUM spacing between bolts (ft.): 3.5
Input the ROW spacing (ft.): 4.0

Figure 5.7 - Input Screens for ROCSIM
help screen appears to guide the user in that particular part of the program. This context sensitive help is useful in explaining the reasoning behind sections of the program and entry values. Explanations of cut sequencing and the method of determining production and cleanup delays are also provided. Help screens allow the addition of useful information; information that is current and available without turning back to the manual provided with the program. This is also useful when providing a “self contained” copy of the program that can be run without the manual.

5.7 Sensitivity Analysis

ROCSIM is enhanced by the ability to perform a sensitivity analysis, changing one input parameter and running the program to see the effect of the change. This is useful both in running the program and in the early stages of debugging the program and its input parameters.

Sensitivity analysis is a valuable tool for the design engineer because it allows “what if” games to be played with the program. How, for example, will the change from 4-foot to 3.5-foot bolt spacing affect the overall cost of a support plan? This allows the true cost of an incremental change in a design to be evaluated. In many cases major changes in some parameters will have minor effects on the overall costs, while some parameters will be very sensitive to relatively minor changes. Sensitivity analysis allows the determination of which parameters are critical in the overall cost of roof fall control.

Using the sensitivity analysis allows a near optimum roof control plan
to be developed by the mine designer. This is possible because, due to legal and technical restraints, there are a limited number of practical opening widths and bolt spacing combinations. In general, true optimization is not possible with a simulation program, but with the small number of feasible alternatives, a near optimum solution can be determined.

5.8 Graphical Output

The output of ROCSIM is purposely kept very simple to facilitate its use with other programs. With the general sophistication of spreadsheet and graphing programs, and the varying requirements of users, it is almost impossible to achieve adequate graphic output within a language such as Pascal. In addition there are the wide variety of output devices, ranging from dot matrix printers, to laser printers, to plotters of all sizes. Each device requires a different graphic driver. By relying on commercial graphics packages the compatibility issue is resolved because most of these programs support a variety of printers and plotters.

The output section of the program provides a detailed listing of each run of the program, as well as a table listing important values that can be used for comparisons between runs. Some of the output files are structured so that a graphing program can easily use them. All the output files are ASCII format, allowing any of these types of programs, as well as spreadsheets, to read in the values for further manipulation, or just for graphic output.

Output is easily customized for use in a contouring package such as
SURFER. SURFER permits contours to be drawn of the size of roof falls as well as factors such as the RRS of the cuts. For example a base map consisting of the RRS values in place can be overlaid with symbols showing the location and size of the roof falls. Since the underlying RRS contours are a constant, only the new roof falls must be posted to produce new maps with the results of different input parameters. Actual digitized mine maps can also be displayed over these contour maps. Using the grid math section of SURFER it is possible to determine the changes in the sizes of roof fall for any particular area due to a change in the input parameters. This also can be shown as a contour or a surface map that will highlight the areas of maximum change.
VI. MODEL VERIFICATION AND VALIDATION

Any simulation model should be verified and validated. Verification ensures that the model will behave in the correct manner over its operational range. Validation is testing to determine whether the model adequately represents the entity that is being simulated. The general steps in the process are shown in figure 6.1.

6.1 Model Verification

There are a number of general techniques that have been developed to verify the correct operation of the simulation model. It is difficult to test a large simulation model all at once, the program must be broken down into smaller segments that can be individually tested. These small segments can be tested by using driver programs that only input the values that the small segments need. These answers can be compared to hand calculated values to insure that for these specific input values, the subprogram is producing the correct output. An example would be to test the output from a gamma distribution function to see if it actually produces a statistically valid output. Similar tests can check segments such as the ones that manage the lists, set bolting patterns, or determine the sequencing of the mining cuts. The cut sequencing section of the program was fairly difficult to code, but the output was tested by plotting the cuts on SURFER, to see if the cuts were all in the proper location.

ROCSIM was constructed with a fairly small main program that uses subprograms and functions to do the majority of the work. This makes de-
Figure 6.1 - Timing and Relationship of Validation and Verification
bugging much simpler than trying to work with a large main program. This also allows the easy addition of new features to the program, because only a new function or subprogram must be verified for correct operation. Care should still be taken to make sure that unexpected conflicts with other sections of the program do not exist.

Verification also includes making sure the program can handle extreme values and error conditions. For example if the event list becomes too large, the simulation program prompts the user to select a smaller time frame for the simulation run. It is of course impossible to try all the possible combinations of input values to see if the program correctly operates under all these varying conditions. Simulations have the additional problem of dealing with random values that can interact with each other in unexpected ways.

6.2 Model Validation

The model was verified to insure that it performed in the proper manner, and the results seemed reasonable. If for example the roof conditions in an area are getting worse, there should be more falls in the area. Similarly, other parameters were varied to see if the results moved in the proper direction. This was accomplished by using the same stream of random numbers, while only one other input variable was changed. The same random numbers were required to remove the influence of “random” variations caused by different streams of numbers.

The main question is whether the model adequately models roof falls.
A critical factor in the program is the percentage of cuts allowed to have a chance of experiencing a roof fall. This factor \( P \) must be used, or every cut will be assigned a roof fall time, and common sense tells us that every cut won’t suffer a roof fall. Currently the program is using a \( P = 0.1 \), but this can be changed if too many roof falls are being experience by the model.
VII. DISCUSSION OF RESULTS

7.1 Analysis of Output

To demonstrate application of the program, ROCSIM was run using varying RRS numbers, with every other variable held constant. This was a hypothetical example to show the effect of varying roof conditions on the overall cost of a mining section. A selection of 3 drill holes were used to input the values of RRS into the program.

Input Data

Number of Shifts to Run 500
Number of Headings 9
Opening Widths 18 → 22 ft.
Seam Thickness 6 ft.
Advance per Cut 20 ft.
Number of Cuts per Shift 15
Cost per Production Shift $10,000
Coal Density 95 pcf
Roof Fall Cleanup per Hour 4 tons
Cost of Cleaning Up 1 Ton of Material $100
Pillar Size 80 X 80 ft.
Resin Bolt Length 5 ft.
Cost per Bolt $18
Roof Bolt Spacing 3.5 → 4.0 ft.
Roof Bolt Row Spacing 4 ft.
The nine heading advance mine layout used in the examples is shown in figure 7.1.

The major differences between runs are opening width, bolt spacing and the RRS values for the roof rock. The opening width and roof bolt spacing are fixed, so the RRS value is the critical difference during each simulation run. Changing the RRS value causes the parameters used by the probability distributions for fall size and time to change and will affect when and where the roof fall will occur. For example, if the fall happens at a later time, a cleanup delay is more likely than a production delay.

In addition to raw values for RRS, the time before support is placed and the location of the cut is developed and become part of the assigned RRS values. Thus vulnerable areas with a larger time before the support is placed will have a lower RRS value than those with better values for location and time of support. In figure 7.2 a contour map showing RRS values is overlaid by symbols that show the location of roof falls occurring during a simulation run. The symbols are coded to indicate whether the fall resulted in a production or a cleanup delay. The size of the symbol is proportional to the relative cost of the roof fall. It can be clearly seen that in the center of the mine, where the RRS values are at their lowest, the highest concentration of expensive falls occurred. The falls in this region also tended to cause production delays rather than less disruptive cleanup delays.

For a comparison of ROCSIM runs using different RRS values, two were completed that were identical except for the input RRS values. The RRS values for the respective runs were as follows,
Figure 7.1 - Mine Layout Used in Example Runs of ROCSIM
Figure 7.2 - Contour Map of RRS Values and Symbols Showing Location, Type and Cost of Roof Falls from One Run

Symbol size is proportional to the cost of the roof fall.
Run 1

<table>
<thead>
<tr>
<th>Y-Cord (ft.)</th>
<th>RRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.40</td>
</tr>
<tr>
<td>2000</td>
<td>-1.50</td>
</tr>
<tr>
<td>5000</td>
<td>-1.30</td>
</tr>
</tbody>
</table>

Run 2

<table>
<thead>
<tr>
<th>Y-Cord (ft.)</th>
<th>RRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.40</td>
</tr>
<tr>
<td>2000</td>
<td>1.50</td>
</tr>
<tr>
<td>5000</td>
<td>1.30</td>
</tr>
</tbody>
</table>

The output for the runs is shown in tables 7.1. This table shows a summary of the information that ROCSIM produces. Graphs showing each run as well as a comparison between the runs are represented in figures 7.3 to 7.7. In figures 7.3 and 7.4, the cleanup, production and total delays for the low RRS cases and high RRS cases are shown respectively. Figures 7.5 through 7.7 show the percentage change in each case with the high RRS run being the baseline for the calculation. In the case of cleanup delay, there is a major increase in the delay hours ranging from 17% to 34%. Production delays actually decreased in many cases due to the timing of the falls, with the changes ranging from -55% to +15%. The change in total delays ran from 17% to 26%.

Analysis of the output shows the total number of roof falls is similar between the two runs, with the increase being only about 6%, but the size and cost of the falls increased significantly with the worsening roof conditions. In the case of 18-foot opening width and 3.5-foot bolt spacing, the cost of the average production delay roof fall went up by 24.2% to $2,679. The cost of cleanup roof falls increased by 17.2% to $478. The overall cost of total roof falls went up by 18.5% to $355,564. There was a small drop in the number of production delay falls, which fell by 13.7%. This is a reflection of falls happening outside of the areas that are defined as production delays.
Table 7.1 - A Summary of Output from Run 1 and Run 2 of ROCSIM

For 500 shifts of simulation time.
In a 9 heading section, whose opening width varies from 18.00 to 22.00 ft.
With a seam thickness of 6.00 ft.
With an advance of 20.00 ft., and an average of 15 cuts per shift.

### Output from run 1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>18.00</td>
<td>3.50</td>
<td>3.50</td>
<td>14.00</td>
<td>74.25</td>
<td>429.51</td>
<td>4.43</td>
<td>0.47</td>
<td>188.7</td>
</tr>
<tr>
<td>18.00</td>
<td>3.75</td>
<td>3.75</td>
<td>15.00</td>
<td>81.11</td>
<td>632.13</td>
<td>3.56</td>
<td>0.60</td>
<td>188.5</td>
</tr>
<tr>
<td>18.00</td>
<td>4.00</td>
<td>4.00</td>
<td>16.00</td>
<td>90.76</td>
<td>647.81</td>
<td>3.55</td>
<td>0.64</td>
<td>188.1</td>
</tr>
<tr>
<td>20.00</td>
<td>3.50</td>
<td>3.50</td>
<td>14.00</td>
<td>134.64</td>
<td>690.44</td>
<td>3.99</td>
<td>0.72</td>
<td>206.6</td>
</tr>
<tr>
<td>20.00</td>
<td>3.75</td>
<td>3.75</td>
<td>15.00</td>
<td>131.50</td>
<td>697.58</td>
<td>3.99</td>
<td>0.71</td>
<td>206.7</td>
</tr>
<tr>
<td>20.00</td>
<td>4.00</td>
<td>4.00</td>
<td>16.00</td>
<td>133.77</td>
<td>855.81</td>
<td>3.20</td>
<td>0.80</td>
<td>206.7</td>
</tr>
<tr>
<td>22.00</td>
<td>3.50</td>
<td>3.50</td>
<td>14.00</td>
<td>130.49</td>
<td>693.92</td>
<td>4.35</td>
<td>0.64</td>
<td>227.5</td>
</tr>
<tr>
<td>22.00</td>
<td>3.75</td>
<td>3.75</td>
<td>15.00</td>
<td>152.82</td>
<td>776.07</td>
<td>3.63</td>
<td>0.74</td>
<td>226.2</td>
</tr>
<tr>
<td>22.00</td>
<td>4.00</td>
<td>4.00</td>
<td>16.00</td>
<td>161.02</td>
<td>862.94</td>
<td>3.64</td>
<td>0.80</td>
<td>225.9</td>
</tr>
</tbody>
</table>

### Output from run 2

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>18.00</td>
<td>3.50</td>
<td>3.50</td>
<td>14.00</td>
<td>68.63</td>
<td>333.81</td>
<td>4.42</td>
<td>0.40</td>
<td>189.2</td>
</tr>
<tr>
<td>18.00</td>
<td>3.75</td>
<td>3.75</td>
<td>15.00</td>
<td>102.93</td>
<td>440.41</td>
<td>3.55</td>
<td>0.56</td>
<td>187.4</td>
</tr>
<tr>
<td>18.00</td>
<td>4.00</td>
<td>4.00</td>
<td>16.00</td>
<td>112.27</td>
<td>462.01</td>
<td>3.55</td>
<td>0.60</td>
<td>187.1</td>
</tr>
<tr>
<td>20.00</td>
<td>3.50</td>
<td>3.50</td>
<td>14.00</td>
<td>114.68</td>
<td>567.43</td>
<td>3.99</td>
<td>0.60</td>
<td>207.5</td>
</tr>
<tr>
<td>20.00</td>
<td>3.75</td>
<td>3.75</td>
<td>15.00</td>
<td>127.39</td>
<td>518.92</td>
<td>3.99</td>
<td>0.61</td>
<td>207.1</td>
</tr>
<tr>
<td>20.00</td>
<td>4.00</td>
<td>4.00</td>
<td>16.00</td>
<td>211.27</td>
<td>561.03</td>
<td>3.20</td>
<td>0.86</td>
<td>202.6</td>
</tr>
<tr>
<td>22.00</td>
<td>3.50</td>
<td>3.50</td>
<td>14.00</td>
<td>110.51</td>
<td>564.89</td>
<td>4.34</td>
<td>0.53</td>
<td>229.2</td>
</tr>
<tr>
<td>22.00</td>
<td>3.75</td>
<td>3.75</td>
<td>15.00</td>
<td>129.94</td>
<td>613.53</td>
<td>3.63</td>
<td>0.61</td>
<td>227.4</td>
</tr>
<tr>
<td>22.00</td>
<td>4.00</td>
<td>4.00</td>
<td>16.00</td>
<td>149.06</td>
<td>608.05</td>
<td>3.63</td>
<td>0.65</td>
<td>226.5</td>
</tr>
</tbody>
</table>
Figure 7.3 - Cleanup, Production and Total Delays in Run 1
Figure 7.4 - Cleanup, Production and Total Delays in Run 2

Run 2 - High RRS, better roof conditions

C. D. - Cleanup Delays
P. D. - Production Delays
T. D. - Total Delays
(X,X') is bolt spacing
Figure 7.5 - Changes in Cleanup Delays Between the Two Runs Caused by Worsening Roof Conditions
Figure 7.6 - Changes in Production Delays Between the Two Runs Caused by Worsening Roof Conditions
Figure 7.7 - Changes in Total Delays Between the Two Runs Caused By Worsening Roof Conditions
In other words, the falls are occurring after the face advances 200 feet past the roof falls.

An analysis of all the predicted roof falls from each run shows the falls from run 2 are larger and more costly than run 1. This 20% difference is a reflection of the increase in roof control cost caused by deteriorating roof conditions. This 20% increase understates the cost disadvantage because of the fact that possible injuries, death, and equipment destruction caused by roof falls are not included in the costs. The only items that are included are the actual cost to clean up the fall, resupport cost, and the cost of loss of production caused by cleanup delays.

The majority of the costs are for cleanup, because the number of cleanup delays outnumber the production delays by about 6 to 1 in terms of lost hours. Production delays make up for their smaller number by being much more expensive than cleanup delays, due to the penalty of stopping all mining activity. The average cost of cleaning up a production delay will be approximately five times higher than the cost incurred while cleaning up the average cleanup fall. The total costs of the cleanup portion is about 1.25 times higher than the production delay costs. In actuality, this probably overstates the case because cleanup delays could be handled more efficiently than production delays. They could be scheduled in groups at a convenient time, and many of the falls would require no cleanup if they were in the return air and did not interfere with ventilation or block escape ways.

Since the production delays are the greater problem, due to their high cost and potential to cause production loss, equipment damage and injury,
they should be looked at more closely than the cleanup delays. Areas where major production delays occur would justify a greater effort in roof bolting. Looking at the location of these delays, and studying their size, can give some indication of the difficulties that these falls will present when the area is mined.

During an individual run, the size and cost of the roof falls increases with the increase in roof bolt spacing and room width. Comparing the roof falls of the run that has the smallest bolt spacing and lowest room width with the run having the largest bolt spacing and highest room width provides a good example for selecting appropriate mining methods. Total cost of the roof falls went from $355,564 to $715,492 and the number of falls increased from 477 to 595. The average cost of cleanup falls increased from $478 to $769, and production fall cost went from $2,735 to $3,420. The greater number of falls and increased cost of each fall accounts for the large difference in the total cost. A judgment would be required if the decrease in bolting cost and the increase in productivity caused by the wider rooms would offset the near doubling of the cost of increased roof fall activity.

7.2 Variation of the P Parameter

The P parameter is the area of roof that can experience roof fall activity. It is necessary because without its use, all the roof will at one time experience a fall in the simulation. The variation of the P parameter allows the modeler an almost linear control in the amount of roof fall activity. A comparison of results using P values ranging from 0.10 to 0.25 is shown in table
7.2 a and 7.2 b.

If the results of the simulation show that roof falls are occurring at the correct time, but the number of falls is too great, then the P value can be lowered in proportion to the needed reduction in the number of falls. If the percentage of roof involved in falls is known from the area, then this can be used in program, otherwise some trial and error is necessary to correct the figure so the proper number of roof falls occur.
Table 7.2 a – Comparison of Delays Caused by Varying the P Parameter

<table>
<thead>
<tr>
<th>P = 0.10</th>
<th>Opening Width(ft)</th>
<th>C. D. (3.5')</th>
<th>P. D. (3.5')</th>
<th>T. D. (3.5')</th>
<th>C. D. (3.75')</th>
<th>P. D. (3.75')</th>
<th>T. D. (3.75')</th>
<th>C. D. (4.0')</th>
<th>P. D. (4.0')</th>
<th>T. D. (4.0')</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td></td>
<td>164.6</td>
<td>36.8</td>
<td>201.5</td>
<td>246.7</td>
<td>51.3</td>
<td>297.9</td>
<td>339.1</td>
<td>59.7</td>
<td>398.8</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>310.5</td>
<td>67.5</td>
<td>378.0</td>
<td>247.9</td>
<td>98.4</td>
<td>346.3</td>
<td>307.5</td>
<td>107.0</td>
<td>414.5</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>308.7</td>
<td>71.5</td>
<td>380.2</td>
<td>280.3</td>
<td>77.2</td>
<td>357.4</td>
<td>328.3</td>
<td>99.8</td>
<td>428.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P = 0.15</th>
<th>Opening Width(ft)</th>
<th>C. D. (3.5')</th>
<th>P. D. (3.5')</th>
<th>T. D. (3.5')</th>
<th>C. D. (3.75')</th>
<th>P. D. (3.75')</th>
<th>T. D. (3.75')</th>
<th>C. D. (4.0')</th>
<th>P. D. (4.0')</th>
<th>T. D. (4.0')</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td></td>
<td>255.4</td>
<td>51.7</td>
<td>307.1</td>
<td>389.0</td>
<td>65.8</td>
<td>454.6</td>
<td>459.6</td>
<td>104.8</td>
<td>564.4</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>488.9</td>
<td>97.6</td>
<td>586.5</td>
<td>412.5</td>
<td>163.3</td>
<td>575.8</td>
<td>484.9</td>
<td>124.6</td>
<td>609.5</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>465.5</td>
<td>74.1</td>
<td>539.6</td>
<td>471.4</td>
<td>108.4</td>
<td>579.8</td>
<td>545.5</td>
<td>151.4</td>
<td>696.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P = 0.20</th>
<th>Opening Width(ft)</th>
<th>C. D. (3.5')</th>
<th>P. D. (3.5')</th>
<th>T. D. (3.5')</th>
<th>C. D. (3.75')</th>
<th>P. D. (3.75')</th>
<th>T. D. (3.75')</th>
<th>C. D. (4.0')</th>
<th>P. D. (4.0')</th>
<th>T. D. (4.0')</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td></td>
<td>362.5</td>
<td>59.8</td>
<td>422.3</td>
<td>554.8</td>
<td>105.2</td>
<td>660.0</td>
<td>582.8</td>
<td>104.9</td>
<td>697.7</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>623.7</td>
<td>113.5</td>
<td>737.1</td>
<td>594.2</td>
<td>168.5</td>
<td>762.7</td>
<td>614.3</td>
<td>185.1</td>
<td>799.4</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>677.3</td>
<td>115.4</td>
<td>792.7</td>
<td>695.4</td>
<td>175.3</td>
<td>870.6</td>
<td>701.6</td>
<td>172.7</td>
<td>874.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P = 0.25</th>
<th>Opening Width(ft)</th>
<th>C. D. (3.5')</th>
<th>P. D. (3.5')</th>
<th>T. D. (3.5')</th>
<th>C. D. (3.75')</th>
<th>P. D. (3.75')</th>
<th>T. D. (3.75')</th>
<th>C. D. (4.0')</th>
<th>P. D. (4.0')</th>
<th>T. D. (4.0')</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td></td>
<td>517.1</td>
<td>59.6</td>
<td>576.6</td>
<td>722.1</td>
<td>128.0</td>
<td>850.1</td>
<td>735.8</td>
<td>112.1</td>
<td>847.8</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>843.7</td>
<td>125.5</td>
<td>969.2</td>
<td>722.7</td>
<td>189.9</td>
<td>912.6</td>
<td>790.8</td>
<td>214.1</td>
<td>1004.8</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>982.3</td>
<td>147.0</td>
<td>1129.2</td>
<td>780.6</td>
<td>214.3</td>
<td>994.9</td>
<td>822.5</td>
<td>248.6</td>
<td>1071.1</td>
</tr>
</tbody>
</table>

% change in P from 0.1 - .15

<table>
<thead>
<tr>
<th>Opening Width(ft)</th>
<th>C. D. (3.5')</th>
<th>P. D. (3.5')</th>
<th>T. D. (3.5')</th>
<th>C. D. (3.75')</th>
<th>P. D. (3.75')</th>
<th>T. D. (3.75')</th>
<th>C. D. (4.0')</th>
<th>P. D. (4.0')</th>
<th>T. D. (4.0')</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>155.2%</td>
<td>140.4%</td>
<td>152.4%</td>
<td>157.7%</td>
<td>128.0%</td>
<td>152.6%</td>
<td>135.5%</td>
<td>175.4%</td>
<td>141.5%</td>
</tr>
<tr>
<td>20</td>
<td>157.4%</td>
<td>144.7%</td>
<td>155.2%</td>
<td>166.4%</td>
<td>166.0%</td>
<td>166.3%</td>
<td>157.7%</td>
<td>116.5%</td>
<td>147.1%</td>
</tr>
<tr>
<td>22</td>
<td>150.8%</td>
<td>103.6%</td>
<td>141.9%</td>
<td>168.2%</td>
<td>140.4%</td>
<td>162.2%</td>
<td>166.2%</td>
<td>151.8%</td>
<td>162.8%</td>
</tr>
<tr>
<td>average</td>
<td>154.5%</td>
<td>129.6%</td>
<td>149.9%</td>
<td>164.1%</td>
<td>144.8%</td>
<td>160.4%</td>
<td>153.1%</td>
<td>147.9%</td>
<td>150.5%</td>
</tr>
<tr>
<td>overall avg =</td>
<td>150.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 7.2 b – Comparison of Delays Caused by Varying the P Parameter

<table>
<thead>
<tr>
<th>Opening Width (ft)</th>
<th>C. D. (3.5')</th>
<th>P. D. (3.5')</th>
<th>T. D. (3.5)</th>
<th>C. D. (3.75')</th>
<th>P. D. (3.75')</th>
<th>T. D. (3.75)</th>
<th>C. D. (4.0')</th>
<th>P. D. (4.0')</th>
<th>T. D. (4.0')</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>220.2%</td>
<td>162.2%</td>
<td>209.6%</td>
<td>224.9%</td>
<td>205.3%</td>
<td>221.6%</td>
<td>171.9%</td>
<td>173.7%</td>
<td>172.4%</td>
</tr>
<tr>
<td>20</td>
<td>200.9%</td>
<td>168.2%</td>
<td>195.0%</td>
<td>239.7%</td>
<td>171.3%</td>
<td>220.3%</td>
<td>199.8%</td>
<td>173.1%</td>
<td>192.9%</td>
</tr>
<tr>
<td>22</td>
<td>219.4%</td>
<td>161.5%</td>
<td>208.5%</td>
<td>248.1%</td>
<td>227.1%</td>
<td>243.6%</td>
<td>213.7%</td>
<td>173.1%</td>
<td>204.2%</td>
</tr>
</tbody>
</table>

Average: 213.5%  164.0%  204.4%  237.6%  201.2%  228.5%  195.1%  173.9%  189.9%

Overall avg = 200.9%

% change in P from 0.1 - 0.25

<table>
<thead>
<tr>
<th>Opening Width (ft)</th>
<th>C. D. (3.5')</th>
<th>P. D. (3.5')</th>
<th>T. D. (3.5)</th>
<th>C. D. (3.75')</th>
<th>P. D. (3.75')</th>
<th>T. D. (3.75)</th>
<th>C. D. (4.0')</th>
<th>P. D. (4.0')</th>
<th>T. D. (4.0')</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>314.1%</td>
<td>161.7%</td>
<td>286.2%</td>
<td>292.8%</td>
<td>249.7%</td>
<td>285.4%</td>
<td>217.0%</td>
<td>187.6%</td>
<td>212.6%</td>
</tr>
<tr>
<td>20</td>
<td>271.7%</td>
<td>186.1%</td>
<td>256.4%</td>
<td>291.5%</td>
<td>193.0%</td>
<td>263.5%</td>
<td>257.1%</td>
<td>200.1%</td>
<td>242.4%</td>
</tr>
<tr>
<td>22</td>
<td>318.2%</td>
<td>205.7%</td>
<td>297.1%</td>
<td>278.5%</td>
<td>277.7%</td>
<td>278.3%</td>
<td>250.6%</td>
<td>249.2%</td>
<td>250.2%</td>
</tr>
</tbody>
</table>

Average: 301.3%  184.5%  279.9%  287.6%  240.1%  275.7%  241.6%  212.3%  235.1%

Overall avg = 250.9%

P - Percentage of roof that can experience a roof fall
C. D. - Cleanup Delays (hrs.)
T. D. - Total Delays (hrs.)

Figure in (X.XX') is roof bolt spacing
VIII. CONCLUSIONS AND RECOMMENDATIONS

8.1 Conclusions

Simulation has shown itself to be an invaluable tool for evaluating systems that are too complex for manipulation in any other way. The method appears to be a particularly valuable tool for optimizing coal mine roof support for the following reasons:

1) It permits determination of roof fall frequency by exploring the effects of changes in support, room width and geology through the use of the RRS rating method.

2) Analysis of situations that are not tractable to a deterministic solution are possible.

3) Support requirements can be evaluated without considering safety factors or conservative strength values and design assumptions.

4) A total cost approach accounts for factors that any standard support design will ignore because simulation can find the costs of seemingly random events such as roof falls.

Limitations in the simulation approach lies in several areas:

1) It never specifies where a roof fall will occur. It can only show the areas that have a greater tendency to experience a fall.

2) Finding the parameters for the probability distributions is a critical problem within a simulation. If the parameters are correct, the simulation model will do a good job of representing the real mine, if the parameters are wrong, the results are less useful. The RRS will help with the task of selecting parameters, but a knowledgeable user will
still need to adjust the model to achieve acceptable results.

8.2 Recommendations

It is recommended that to verify the effectiveness of ROCSIM, the program should be used in a number of mines. Modifications will be needed in both the input and output variables depending on the needs of the particular mining operations. The program should be able to deal with an individual mine's geology by selecting the appropriate input parameters for the probability distributions that the program supports. Modifications of the output variables could be made in response to needs for graphical output, and for specific information needs.

Further work is necessary to correlate the RRS (Roof Rating System) with falls within a mine environment. Work on appropriate weighting for the variables in RRS is needed to determine the ranking in importance of its individual sections. With this improvement, RRS could be very useful in finding the probability that a roof fall will take place in a particular area of a mine, and as a system to rank the quality of the roof rock in a given area.
REFERENCES


Mineral Industry, University of Arizona, Tucson, April 7 - 11, pp. 337 - 342.


APPENDIX A

Pillar Design Equations
A.1 Basic Methods of Pillar Design

Many methods have been used to try to determine the likelihood of a roof fall occurring within a specific area of a mine. One approach is to use standard pillar and support design equations to formulate the necessary support parameters. A safety factor can be calculated for the support that the pillar provides and this can be an indication of the ability of the design to support the roof successfully.

To determine the average stress in a pillar the tributary area method is used to calculate the average pillar stress according to using the following formula.

\[ S_p = \gamma H \frac{(w + L)(L + B)}{wL} \]  

\( S_p \) - average pillar stress  
\( H \) - depth below surface  
\( w \) - pillar width  
\( L \) - pillar length  
\( B \) - entry width  

It assumes that all the load from an area extending half the opening width away from the pillar is carried by the pillar.

There have been many ways to calculate the strength of a coal pillar used by various researchers over the years. The pillar strength formulas require the strength of a coal specimen of critical size. This is the size where the strength of the coal specimen does not change when the size of the tested sample is increased. To obtain this strength a \( k \) factor is required. \( K \) is
calculated using the formula shown in the Holland-Gaddy pillar strength method in equation 2.4. This k can be converted into \( \sigma_1 \) by using Hustrud's (1976) method for seams of varying thickness as shown in equation A.2.

\[
\sigma_1 = \frac{k}{\sqrt[3]{36}} \quad \text{for } h > 36 \text{ in.}
\]

\[
\sigma_1 = \frac{k}{\sqrt[3]{h}} \quad \text{for } h < 36 \text{ in.}
\]

\( \sigma_1 \) - strength of critical size specimen  
\( k \) - factor from Gaddy equation  
\( h \) - height of seam (in)  

\[\text{(A.2)}\]

Holland (1964) extended the work of Gaddy, and produced a pillar strength equation that takes into account the width vs. the height ratio of the pillar. This is shown in equation A.3.

\[
\sigma_p = \frac{k \sqrt{w}}{h}
\]

\[k = \sigma_c \sqrt{D}\]

\( \sigma_p \) - pillar strength (psi)  
\( \sigma_c \) - uniaxial compressive strength of coal  
\( w \) - pillar width (in)  
\( h \) - pillar height (in)  

\[\text{(A.3)}\]

Obert and Duvall (1967) proposed another system of calculating the strength of pillars which is shown in equation A.4. This was found to be acceptable for pillars with a width to height ratio of 0.25 to 4.0.
\[ \sigma_p = \sigma_1 \left( 0.778 + 0.222 \frac{w}{h} \right) \]

\( \sigma_p \) - pillar strength  
\( w \) - pillar width  
\( h \) - pillar height  
\( \sigma_1 \) - uniaxial strength of cubical specimen  

(B.4)

Bieniawski (1984) proposed an equation he developed from his work in South Africa and in the United States on standing pillars in coal mines. This equation is very similar in form to Obert’s, and is shown in equation B.5.

\[ \sigma_p = \sigma_1 \left( 0.64 + 0.36 \frac{w}{h} \right) \]

\( \sigma_p \) - pillar strength  
\( w \) - pillar width  
\( h \) - pillar height  
\( \sigma_1 \) - strength of cubical specimen > 1m  

(B.5)

Wilson (1977) proposed a yield pillar theory where the pillar core stays intact and is surrounded by yielding material providing confining stress and the equations for both strong and weak roof and floor are shown in equation B.6. This is used to design pillars that will gradually transmit overburden stresses to barrier or abutment pillars.
\[ k = \frac{1 + \sin \phi}{1 - \sin \phi} \]
\[ w = 2(x_b + C) \]

For strong roof and floor
\[ x_b = \frac{h}{F_k} \ln \left( \frac{\gamma h}{p} \right) \]
\[ F_k = \frac{k - 1}{\gamma k} + \frac{(k-1)^2}{k} \tan^{-1} \sqrt{k} \]

Weak rock in roof, seam and floor
\[ x_b = \frac{h}{2} \left[ \left( \frac{\gamma h}{p} \right)^{k^{-1}} - 1 \right] \]  \hspace{1cm} (A.6)

Peak abutment pressure for both cases
\[ \sigma_y = k \gamma h \]

- $x_b$ - width of yield zone
- $w$ - width of pillar
- $C$ - constant
- $h$ - seam height
- $H$ - depth below the surface
- $\gamma$ - unit weight of rock
- $k$ - triaxial stress factor
- $p$ - horizontal restraint at the ribside ($=0.1$ MPa)
APPENDIX B

Listing of the Pascal program ROCSIM

(Roof control Optimization Cost SIMulation)
Program ROCSIM:
(Roof control Optimization Cost SIMulation
Simulates roof fall in a room and pillar coal mine)
uses dos,crn,math,dist;
label 666;
var
check,Overflow:boolean;
IRR,RSL,GW,checkMore:char;
data,data2,data3,data4,data5,data6,DataRS:real;
ahead,behind,head,tail,item,nar,option,row,maxatr,i,j,n,m,pk,ik,k
:integer;
CountProd,CountClean,shifts,headings,CutN,FinalTime,TotalCutN:integer;
counter,NumberRSS,WidthInc,BoltInc,WidthCount,BoltCount:integer;
HeadCuts.getXcutcuts,NoRounds,NumberCuts,width1,bolt1,RSSI:integer;
time,delay1,LowWidth,UpWidth,BegDepth,EndDepth,spacing,SeamThick,Density
:real;
size,advance,FLength,FWidth,NoCuts,distance,width,break,TimeRemain:real;
BoltSpace,RowSpace,BoltCost,BoltLength,TonsCut,CleanUp,CostCleanUp:real;
TotalBoltCost,w1,SumLength,SumWidth,SumThick:real;
TimeUsed,ShiftTime,ProdLoss,CleanTime,ShiftCost,TotalCost:real;
RSS,ProdTime,previous,next,TimeCut,Productivity:real;
BoltDensity,CutBoltCost,NewBoltSpace,BegBoltSpace,EndBoltSpace,x:real;
NewRowSpace,TotalFallBolt,FailBoltingCost,DFCS,TS,QKD,SIR,Location:real;
LargeFall,FailSize,Length,Lengths,Length,EndDist,BegDist,BegRSS,EndRSS
:real;
LargeThick,P,XStart,XCord,YCord,XOrig,YOrig,XAdvance,YAdvance:real;
LocationRSS:real;
BegHour,EndHour,BegMin,EndMin,BegSec,EndSec,Sec100,deltahour,deltamin,
deltasec:word;
linkr,linkp:array[1..25] of integer;
master:array[1..1000,1..9] of real;
trans,tem:array[1..9] of real;
Timealpha,Timebeta,lengtha,lengthb,widtha,widthb,thicka,thickb:
array[1..3,1..3,1..4] of real;
procedure pause;(causes program to pause)
begin
write('Press any key to continue');
repeat until KeyPressed;
end;(procedure pause)

procedure FirstRecord(list:integer);
(Places the record first in a list, 1<=list<=25)
begin
row := nar;
nar := linkr[row];
if nar >= 0 then
  linkp[nar] := 0;
linkr[row] := 0;
head[list] := row;
tail[list] := row;
end;(procedure FirstRecord)

procedure BeforeRecord(list:integer);
(Places a record ahead of the first in a list, 1<=list<=25)
begin
  if isize[list] = 1 then (check for a new list)
    FirstRecord(list)
  else
    begin
      row := nar;(places record in first available space)
nar := linksr[row]; {updates first available space}
if nar >= 0 then
    linkpr[nar] := 0;
ithead := head[list]; {sets successor and predecessor links}
linkpr[ithead] := row;
linksr[row] := ithead;
linkpr[row] := 0;
head[list] := row; {set head of list}
end;
end; {procedure BeforeRecord}

procedure LastRecord(list: integer);
{places a record at the end of the list
with 1<=list<=25 }
begin
    if lsize[list] = 0 then {check to see if list is empty}
        FirstRecord(list)
    else
        begin
            row := nar; {places record in first available location}
            nar := linksr[row]; {finds next available space}
            if nar >= 0 then
                linkpr[nar] := 0;
            itail := tail[list]; {sets predecessor and successor links}
            linkpr[row] := itail;
            linksr[itail] := row;
            linksr[row] := 0;
            tail[list] := row;
        end;
end; { LastRecord}

procedure insert(list: integer);
{inserts a record and adjusts the proceeding and succeeding links
with 1<=list<=25 }
begin
    ahead := links[r[behind]];
    row := nar; {uses first available space}
    nar := linksr[row];
    if nar > 0 then
        linkpr[nar] := 0;
    linkpr[row] := behind; {sets successor and predecessor links}
    linksr[behind] := row;
    linkpr[ahead] := row;
    linksr[row] := ahead;
end; {procedure insert}

procedure fileit (option, list: integer);
{This is the main driving part of the filing system
insert a record into list using option
with 1<=list<=25
option 1:file record in trnsfr[] before first record in list
  2:file record in trnsfr[] after last record in list
  3:file record in trnsfr[] in list in increasing order on lrk[]
  4:file record in trnsfr[] in list in decreasing order on lrk[]
}
begin
    if nar = 0 then {checks to see if master array is full}
        begin
            writeln('Master storage array overflow');
            Overflow := true;
            exit;
        end;
    if (list < 1) or (list > 25) then
        begin
writeIn(list : 3, ' is an improper value for file option.');
halt;
end;
lsizel [list] := lsize[list] + 1;
case option of
{goes to different subroutines depending on value of
option}
1: BeforeRecord(list); {places record first in list}
2: LastRecord(list); {places record last in list}
3: begin {ranks record in increasing order on lrank[]}
item := lrank[list]; {which attribute that the list is ranked on}
row := head[list]; {starts with first record in list}
if lsize[list] = 1 then
FirstRecord(list)
else
begin
while (trnsfr[item] >= master[row, item]) and
{searches for proper place in list to place record}
(tail[list] <= behind) do
begin
behind := row;
row := linksr[behind];
end; {end of while statement}
if tail[list] = behind then
LastRecord(list)
else
begin
if row = head[list] then
BeforeRecord(list)
else
insert(list); {places record in this location}
end;
end;
end; {case option 3}
4: begin {rank record in decreasing order on lrank[]}
item := lrank[list];
row := head[list];
if lsize[list] = 1 then
FirstRecord(list)
else
begin
while (trnsfr[item] <= master[row, item]) and
{searches for proper place in list to place record}
(tail[list] <= behind) do
begin
behind := row;
row := linksr[behind];
end; {end of while statement}
if tail[list] = behind then
LastRecord(list)
else
begin
if row = head[list] then
BeforeRecord(list)
else
insert(list);
end;
end;
end; {case option 4}
else {for improper option}
begin
writeIn(option : 2, ' is an improper value for option');
halt;
end;
end; {improper option}
for item := 1 to maxatr do  (transfers data to master array)
    begin
        master[row, item] := trnsfr[item];
    end;
end;{procedure fileit}

procedure remove(option,list:integer);
{ removes a record from a list and places it in trnsfr[]
  with 1 <= list <= 25
  option 1:Removes first record from list and puts in trnsfr[]
  2:Removes last record from list and puts in trnsfr[] }
begin
  if (list<0) or (list>25) then
      begin
          writeln(list:3,', is an improper value for list.');
          halt;
      end;
  if lsize[list]=0 then
      begin
          writeln('List is empty, can not use remove.');
          halt;
      end;
  lsize[list]:=lsize[list] - 1;{decrements list size}
  if lsize[list]=0 then {checks if only record in list}
      option:=-10;
  case option of
    -10:begin {used if only record in list}
        row:=head[list];
        head[list]:=0;
        tail[list]:=0;
      end; {of option -10}
    1:begin {used to remove first record in list}
        row:=head[list];
        ihead:=linksr[row];
        linkpr[ihead]:=0;
        head[list]:=ihead;
      end; {option 1}
    2:begin {used to remove last record in list}
        row:=tail[list];
        itail:=linkpr[row];
        linksr[itail]:=0;
        tail[list]:=itail;
      end {option 2}
    else {checks for errors in entering option}
      begin
          writeln(option:3,', is an improper value for option.');
          halt;
      end;{error check option}
  end;
  linksr[row]:=narm;
  linkpr[row]:=0;
  narm:=row;
  for i:=1 to maxatr do  {places removed record in trnsfr[]} 
      trnsfr[i]:=master[row,i]
end;{procedure remove}

function CostBolting:real;
(Calculates bolting cost for one cut)
var
  x1,x2:real;
  NoBolts,NoRows:integer;
begin
  x1:=width/BoltSpace;
x2:=advance/RowSpace;
if frac(x1)=0 then (determines if an even number of bolts will go in)
   NoBolts:=round(x1) - 1
else
   NoBolts:=round(x1 + 0.5) - 1;
if frac(x2)=0 then (determines if an even number of bolts will go in)
   NoRows:=round(x2)
else
   NoRows:=round(x2 + 0.5);
NewBoltSpace:=width/(NoBolts + 1);
NewRowSpace:=advance/NoRows;
CostBolting:=NoRows * NoBolts * BoltCost;
BoltDensity:=NewBoltSpace * NewRowSpace;
end;(function CostBolting)

procedure CutLocation(Cut:integer;var Location:real);
{ assigns a X-Coordinate and a Y-Coordinate to a cut and a location using a single cut sequence. }
Locations
1 - heading
2 - beltway
3 - T-intersection
4 - X-intersection
5 - CrossCuts

var
   passage,place,XCutNo:integer;
begin
   place:= Cut mod Norounds;{Cut number within each cut sequence}
   passage:= place mod headings;{which heading the cut is made}
   if place <= (headings * HeadCuts) then
      {checks if cut is made before X-cuts are begun}
      begin
         if passage = 1 then {checks for first heading}
            begin
               XCord:= XOrig;{sets X-Cord to X-cord of first heading}
               YCord:= YCord + YAdvance;{advance Y-cord one cut}
            end
         else
            begin
               XCord:= XCord + PWidth + Width;{increments X-cord by center to center dist}
               if passage = round((headings -1)/2 +1) then{finds beltway}
                  Location:=2 {beltways}
               else if place > (headings*(HeadCuts-1)) then{checks for intersection}
                  begin
                     if (passage=0) or (passage=1) then
                         Location:=3
                     else
                         Location:=4;
                  end
               else if place = 0 then
                  Location:=5 {last cut of sequence, in cross-cut}
               else
                  Location:=1;
            end
         end
   else {cuts within the cross-cuts}
      begin
         Location:=5; {X-Cuts}
         XCutNo:=(place - (headings * HeadCuts)) mod (headings -1);
         {finds cut number within X-cut sequence}
         if place = (headings * HeadCuts +1) then
            begin
XStart:=XOrig + PWidth + Width - XAdvance;
X Cord:=XStart
end
else
begin
if XCUTNo = 1 then
begin
XStart:=XStart - XAdvance;
X Cord:=XStart;
end
else
X Cord:=X Cord + PWidth + Width;
end;
end;
end;{procedure CutLocation}

procedure CutAdvance;{ finds # of cuts in xcuts and headings and cut advance}
begin
  HeadCuts:=round((PLength + Width)/Advance + 0.5);
  XCutCuts:=round((PWidth/Advance + 0.5));
  YAdvance:=(PLength + Width)/HeadCuts;
  XAdvance:=PWidth/XCutCuts;
  NoRounds:=(headings * HeadCuts) + (headings - 1) * XCutCuts;
end;{procedure CutAdvance}

procedure FallDelay(var ProdT,FallBoltCost:real;var test:boolean);
{ finds cost of roof fall cleanup}
var
  CleanCost,ProdCost:real;
begin
distance:=YCord - trnsfr[4];
if (trnsfr[2]=2) or (distance <= 200) then
  { checks to see if fall occurs within 200 feet of face or if it occurs in a belteay}
  begin
    ProdT:=trnsfr[8];
    ProdCost:=trnsfr[8]/6 * ShiftCost;{cost of lost production time}
    FallBoltCost:=round(trnsfr[5]/BoltDensity) * BoltCost * 1.5;
    trnsfr[9]:=ProdCost + CleanCost + FallBoltCost;
    test:=true;
  end
else
begin
  FallBoltCost:=round(trnsfr[5]/BoltDensity + 0.5) * BoltCost * 1.5;
  ProdT:=0;
  test:=false;
end;
end;{procedure FallDelay}

procedure initparameter;
var
  i,j,k,im:integer;
begin
assign(data6,'para.dat');
reset(data6);
readin(data6);
for i,j:=1 to 3 do
for k:=1 to 3 do
for im:=1 to 4 do
begin
readln(data6,Timealpha[i,j,k,im],timebeta[i,j,k,im],lengtha[i,j,k,im],
lengthb[i,j,k,im],widtha[i,j,k,im],widthb[i,j,k,im],thicka[i,j,k,im],
thickb[i,j,k,im]);
end;
close(data6);
end;{procedure initparameter}

procedure RRSSIn;
{Input and file reading routine for RRS}
begin
if UpCase(filecheck)='Y' then
begin
readln(data4,LocationRRS);
readln(data4,IRR);
readln(data4,DFCS);
readln(data4,GW);
readln(data4,RQD);
readln(data4,SIR);
end
else
begin
writeln;
writeln;
writeln('For the ',pk;3,' RRS calculation.');
write('Input the distance from the beginning of simulation run? ');
readln(LocationRRS);
writeln;
writeln('Input V(Very Good), G(Good), F(Fair), P(Poor), ');
write('or E(Extremely Poor) for immediate roof rock? ');
readln(IRR);
writeln;
write('Input distance in feet to first competent strata? ');
readln(DFCS);
writeln;
write('Input D(Dry), M(Moist), W(Wet), V(Very wet). or F(Flowing water) ');
readln(GW);
writeln;
write('Input the RQD of the roof material (%) ? ');
readln(RQD);
writeln;
write('Input the uniaxial compressive strength of intact rock in psi? ');
readln(SIR);
end;
end;{procedure RRSSIn}

procedure RRSSys(var RRSNo:real;var EntryCheck:boolean);
{ uses Roof Rating System to assign a rating to an area
+2 - Very Good
+1 - Good
0 - Fair
-1 - Poor
-2 - Very Poor }
var
IRRNo,GWNo,DFCSNo,RQDNo,SIRNo:integer;
IRRW,GW,W,DFCSW,RQDW,SIRW:real;
begin
RRSSIn;{Input procedure}
IRRW:=1/7; {weights assigned to each category of the rating}
DFCSW:=1/7;
GWW:=1/7;
RQDW:=1/7;
SIRW:=1/7;
IRRN:=0;DFCNS:=0;GWN:=0;RQDN:=0;SIRN:=0;
EntryCheck:=true;{initializes proper entry variable to true}
if UpCase(IRR) = 'V' then
  IRRN:=2
else if UpCase(IRR) = 'G' then
  IRRN:=1
else if UpCase(IRR) = 'F' then
  IRRN:=0
else if UpCase(IRR) = 'P' then
  IRRN:=-1
else if UpCase(IRR) = 'E' then
  IRRN:=-2
else
  EntryCheck:=false;{test for improper entry}
if DFCS <= 5 then
  DFCNS:=2
else if DFCS <= 10 then
  DFCNS:=1
else if DFCS <= 25 then
  DFCNS:=0
else if DFCS <= 35 then
  DFCNS:=-1
else
  DFCNS:=-2;
if UpCase(GW) = 'D' then
  GWN:=2
else if UpCase(GW) = 'M' then
  GWN:=1
else if UpCase(GW) = 'W' then
  GWN:=0
else if UpCase(GW) = 'V' then
  GWN:=-1
else if UpCase(GW) = 'F' then
  GWN:=-2
else
  EntryCheck:=false;{test for improper entry}
if RQD <= 25 then
  RQDN:= -2
else if RQD <= 50 then
  RQDN:= -1
else if RQD <= 75 then
  RQDN:= 0
else if RQD <= 90 then
  RQDN:= 1
else if RQD <= 100 then
  RQDN:= 2
else
  EntryCheck:=false;{test for improper entry}
if SIR <= 3625 then
  SIRN:=-2
else if SIR <= 7250 then
  SIRN:=-1
else if SIR <= 14500 then
  SIRN:=0
else if SIR <= 36250 then
  SIRN:= 1
else
  SIRN:= 2;
RRSN:=IRRN*IRRW+DFCNS*DFCSW+GWN*GWW+RQDN*RQDW+SIRN*SIRW;
end; {procedure RRSys}
procedure RRSRoutine;[controls the calculation and storage of RRS]
Var
  ProperEntry:boolean;
begin
  for pk:=1 to k do
  begin
    repeat
      RRSys(trnsfr[2],ProperEntry);
      if not ProperEntry then
        writeln('There was a problem in the entry of the RRS data,
            please try again. ')
      until ProperEntry;'(checks for proper input values)
      if not (UpCase(filecheck)='Y') then
        begin
          writeln(data4,LocationRRS:10:1);
          writeln(data4,IRRR);
          writeln(data4,DFCS:6:2);
          writeln(data4,GW);
          writeln(data4,QD:6:2);
          writeln(data4,SIR:10:2);
        end;
        trnsfr[1]:=LocationRRS;
        fileit(3,2);
    end;
    assign(data5,'RRS.out');
    rewrite(data5);
    row:=head[2];
    for i:=1 to lsize[2] do
    begin
      writeln(data5,master[row,1]:10:1,master[row,2]:8:2);
      row:=linksr[row];
    end;
  close(data4);
end;{procedure RRSRoutine}

function RRSNow:real;
Var
  RRSTemp,TSNo,TSW,RSLNo,RSLW,SupportTime:real;
begin
  RSLW:=1/7;
  TSW:=1/7;
  if k = 0 then
    RRSTemp:=0
  else
    begin
      if YCord < EndDist then
        RRSTemp:=BegRRS+((YCord-BegDist)/(EndDist-BegDist))
           *(EndRRS-BegRRS)
      else
        begin
          if lsize[2] > 0 then
            begin
              remove(1,2);
              BegDist:= EndDist;
              EndDist:= trnsfr[1];
              BegRRS:= EndRRS;
              EndRRS:= trnsfr[2];
              RRSTemp:=BegRRS+((YCord-BegDist)/(EndDist-BegDist))
               *(EndRRS-BegRRS);
            end
          else
            RRSTemp:=EndRRS;
SupportTime:=trian(0.5,2,15);
if trnsfr[2]=3 then
  RSLNo:=0
else if trnsfr[2]=4 then
  RSLNo:=-1
else
  RSLNo:=2;
if SupportTime < 2.5 then
  TSNo:=2
else if SupportTime < 5 then
  TSNo:=1
else if SupportTime < 8 then
  TSNo:=0
else if SupportTime < 12.5 then
  TSNo:=-1
else
  TSNo:=-2;
RRSNow:=RRSTemp + TSNo * TSW + RSLNo * RSLW;
end;{RRS Now}

procedure TimeExec;
begin
  deltax:=EndHour - BegHour;
  if BegMin <= EndMin then
    deltax:= EndMin - BegMin
  else
    begin
      deltax:=deltax-1;
      deltax:=EndMin + 60 - BegMin;
    end;
  if BegSec <= EndSec then
    deltasec:=EndSec - BegSec
  else
    begin
      deltasec:=deltasec-1;
      deltasec:=EndSec + 60 - BegSec;
    end;
end;{procedure TimeExec}

procedure restart;{if master storage array overflows}
begin
  Sound(400);{beep on error}
  Delay(100);
  NoSound;
  writeln('The master storage array is full, a smaller value for number');
  writeln('of shifts must be chosen, the current number is , shifts:4);
  writeln('An amount < , round(TotalCutN/(shifts*NoCuts)*shifts):5,' is
  writeln('recommended.'););
  write('Please input a new shift number? ');
  readln(shifts);
  ClrScr;
  rewrite(data2);
  rewrite(data3);
  reset(data4);
  readln(data4,k);
end;{procedure reset}

procedure DataIn;{Data input procedure}
begin
  Assign(data,'fall.dat');
assign(data4, 'RRS.dat');
assign(dataRRS, 'RRSCon.dat');
rewrite(dataRRS);
write('Are you using a previously made data file, (Y/ N) ');
readln(filecheck);
if UpCase(filecheck)='Y' then
  begin
    reset(data);
    reset(data4);
    readln(data4, k);
    readln(data, shifts);
    readln(data, headings);
    readln(data, LowWidth, UpWidth);
    readln(data, WidthInc);
    readln(data, SeamThick);
    readln(data, advance, NoCuts);
    readln(data, ShiftCost);
    readln(data, Density);
    readln(data, CleanUp);
    readln(data, CostCleanUp);
    readln(data, PLength, FWidth);
    readln(data, BoltLength, BoltCost);
    readln(data, BegBoltSpace, EndBoltSpace);
    readln(data, RowSpace, BoltInc);
    close(data);
  end
else
  begin
    rewrite(data);
    rewrite(data4);
    write('Input the number of shifts that the simulation will run? ');
    readln(shifts);
    writeln(data, shifts:3);
    write('Input the number of headings? ');
    readln(headings);
    writeln(data, headings:2);
    write('Input the lower and upper bound of heading width(ft.)? ',
         LowWidth, UpWidth);
    writeln(data, LowWidth:6:1, UpWidth:6:1);
    write('Input the number of increments between these two widths? ');
    readln(WidthInc);
    writeln(data, WidthInc:4);
    write('Input the seam thickness(ft.)? ',
         SeamThick);
    readln(data, SeamThick);
    writeln(data, SeamThick:7:2);
    write('Avg. number of feet advanced per cut and number of cuts per shift? ');
    readln(advance, NoCuts);
    writeln(data, advance:6:3, NoCuts:5:1);
    write('Input the Cost of a production shift? ');
    readln(ShiftCost);
    writeln(data, ShiftCost:7:2);
    write('Input the density of the coal (pcf)? ',
         Density);
    readln(Density);
    writeln(data, Density:6:2);
    write('Input the tons of roof fall that can be cleaned up in one hour? ');
    readln(CleanUp);
    writeln(data, CleanUp:7:2);
    write('Input the cost of cleaning up a ton of roof fall material? ')};
readin(CostCleanUp);
writeln(data,CostCleanUp:7:2);
write('Input pillar length and width(ft.)? ');
readin(Length,PWidth);
writeln(data,Length:6:1,PWidth:6:1);
write('Input bolt length and cost per bolt including installation? ');
readin(BoltLength,BoltCost);
writeln(data,BoltLength:6:1,BoltCost:8:2);
write('Input the beginning spacing between bolts and the maximum spacing between bolts? ');
readin(BegBoltSpace,EndBoltSpace);
writeln(data,BegBoltSpace:5:1,EndBoltSpace:5:1);
write('Input the row spacing and the number of increments between bolt spacing? ');
readin(RowSpace,BoltInc);
writeln(data,RowSpace:5:1,BoltInc:4);
close(data);
write('How many RRS (Roof-Rock Ratings) have to be calculated? ');
readin(k);
writeln(data,k:K:3);
end;
end;

procedure DataOutInt:[Does the heading for the output]
begin
assign(data2,'fall.out');
assign(data3,'graph.out');
rewrite(data2);
rewrite(data3);
writeln(data3,'For ','shifts:4,' shifts of simulation time.');
writeln(data3,'In a ','headings:2,' heading section, whose opening width varies from ','LowWidth:4:2,' to ','UpWidth:4:2,' ft.);
writeln(data3,'With a seam thickness of ','SeamThick:4:2,' ft.);
writeln(data3,'With an advance of ','advance:4:2,' ft., and an average of ','NoCuts:3:0,' cuts per shift.');
writeln(data3);
writeln(data3,'(ft) (ft) ft^2/bolt (hrs)','','$/ton','$/ton t/hr');
end;

procedure DataOut:[Outputs the data]
begin
writeln('WidthCount = ','WidthCount:3','BoltCount = ','BoltCount:3');
writeln('TotalBoltCost = ','TotalBoltCost:6:2','TotalCutN = ','TotalCutN:4');
writeln('Total Time = ','deltahour:2',';','deltamin:2',';','deltasec:2');
writeln(data2,'Width = ','width:5:2','BoltSpace = ','BoltSpace:5:2','Bolt spacing = ','BoltDensity:4:2');
writeln(data2,'Size[1] = ','Size[1]:4');
writeln(data2,'CountProd = ','CountProd:5',' CountClean = ','CountClean:5');
writeln(data2,'Theoretical Cuts ','Shifts*NoCuts:7:0',' Total CutN = ','TotalCutN:8');
writeln(data2,'Production delays = ','ProdLoss:8:2',' Cleanup time = ','CleanTime:8:2',' Total cost = ','TotalCost:8:1');
writeln(data2,'Total Bolting Cost = ',TotalBoltCost:10:1,' Total Fall
Bolting = ',TotalFallBolt:8:1);
if lsize[1] > 0 then
  writeln(data2,'Avg. Length = ',SumLength/lsize[1]:4:2,' ft,
  Avg. Width = ',SumWidth/lsize[1]:4:2,
  ft, Avg. thick = ',SumThick/lsize[1]:4:2,' ft');
writeln(data2,'Biggest Fall Size = ',LargeFall:6:1,' ft^3,
  with L = ',LargeLength:4:2,' ft, W = ',LargeWidth:4:2,' ft
  and T = ',LargeThick:4:2,' ft');
writeln(data2,'Total Time = ',deltahour:2,':',deltamin:2,':',deltasec:2);
{ row=head[1];
  for i:=1 to lsize[1] do
  begin
  writeln(data2,master[row,1]:8:2,master[row,2]:6:0,master[row,3]:8:0,
  master[row,4]:8:0,master[row,5]:8:0,master[row,6]:5:1,
  master[row,7]:7:0,master[row,8]:5:1,master[row,9]:5:0);
  behind:=row;
  row:=linksr[behind];
  end; }
writeln(data2);
writeln(data3,width:5:2,BoltSpace:8:2,NewBoltSpace:8:2,BoltDensity:10:2,
  ProdLoss:11:2,CleanTime:9:2,TotalBoltCost/(TonsCut*TotalCutN):8:2
  ,TotalCost/(TonsCut*TotalCutN):9:2,(TotalCutN*TonsCut/(Shifts*8))
  :9:1);
end;{procedure DataOut}

procedure initlk;
  {initializes program}
var list:integer;
begin
  FillChar(master,SizeOf(master),0);{initializes the various arrays to 0}
  FillChar(head,SizeOf(head),0);
  FillChar(tail,SizeOf(tail),0);
  FillChar(lsize,SizeOf(lsize),0);
  FillChar(lrank,SizeOf(lrank),0);
  FillChar(linkpr,SizeOf(linkpr),0);
  FillChar(transfr,SizeOf(transfr),0);
  for row:=1 to 1000 do {sets the successor links to row + 1}
    linksr[row]:=row + 1;
  linksr[1000]:=0;
  SumLength:=0;
  SumWidth:=0;
  SumThick:=0;
  TimeRemain:=0;
  Next:=0;
  time:=0;
  TimeRemain:=0;
  CountProd:=0;
  CountClean:=0;
  TotalCost:=0;
  TotalFallBolt:=0;
  TotalBoltCost:=0;
  CutN:=0;
  TotalCutN:=0;
  ProdLoss:=0;
  CleanTime:=0;
  LargeFall:=0;
  NumberCuts:=0;
  XORig:=0;
  YORig:=0;
  delayT:=0;
  nar:=1;{sets the first available space in master[] to row = 1}
lr ank[25]: = 1;
lrank[1]: = 1;
lrank[2]: = 1;
Overflow:=false;
maxatr:=9; {maximum number of attributes in list}
end;{procedure initk}

begin {START OF MAIN PROGRAM}
{Attributes
1 - time of fall (hrs.)
2 - location (ex. X-cut)
3 - X-Coordinate(ft)
4 - Y-Coordinate(ft)
5 - area of roof fall (ft^2)
6 - thickness of fall (ft)
7 - volume of fall (ft^3)
8 - delay caused by falls (hours)
9 - cost of roof fall

Lists
25 - event list, ranked by time
1 - fall list, ranked by time
2 - RRS information

TextBackground(blue);
TextColor(yellow);
ClrScr;
Randomize; {provides for a new string of random numbers}
DataIn; {goes to Input procedure}
666:DataOutInt; {writes heading on output and reentry point for
error condition}
initk;{initialization}
iniparameter;
RR SRoutine; {finds RRS values for mine}
Width:=LowWidth;
for WidthCount:=1 to WidthInc do {start of width do loop}
begin
BoltSpace:=BegBoltSpace;
CutAdvance;{finds # of cuts in xcuts and headings and cut advance}
for BoltCount:=1 to BoltInc do {start of bolt do loop}
begin

initk;{initializes the variables}
reset(data5);
GetTime(BegHour,BegMin,BegSec,Sec100);
for i:=1 to k do {reads RRS values back into list[2]}
begin
readln(data5, trnsfr[1], trnsfr[2]);
fileit(2, 2);
end;
if lszie[2] > 0 then {takes first value of RRS from list[2]}
remove(1, 2);
X'Cord:=Y'Orig;
BegDist:=0;{initializes values for determination of RRS}
EndDist:=trnsfr[1];
BegRBS:=trnsfr[2];
EndRBS:=BegRBS;
RR S:=BegRBS;
CutBoltCost:=CostBolting;{calls CostBolting Procedure}
writeln('For width = ',width:4:2,' Bolt Spacing =
','NewBoltSpace:4:2);
writeln('Bolting Cost/cut = ',CutBoltCost:7:2);
TonsCut:=SeamThick * width * advance * Density/2000;
if width <= 18 then
    widthI:=1
else if width <= 22 then
    widthI:=2
else
    widthI:=3;
if BoltDensity <= 14 then
    BoltI:=1
else if BoltDensity <= 16 then
    BoltI:=2
else
    BoltI:=3;
FinalTime:=shifts * 8;
TimeCut:=8/NoCuts;
repeat (Top of sample generating loop that runs until
    time => FinalTime)
    if Lsize[25]=0 then {checks to see if event list is empty}
        begin
        previous:=next;
        next:=FinalTime;
        ProdTime:=0;
        end
    else
        begin
        previous:=next;
        repeat {Removes falls from events list until a production delay}
        Remove[1,25];
        FallDelay(ProdTime,FallBoltingCost,check);
        if check then {sees if a production delay occurred}
            begin
            ProdLoss:=ProdLoss + trnsfr[8];
            Inc(CountProd);
            end
        else {for cleanup delays}
            begin
            CleanTime:=CleanTime + trnsfr[8];
            Inc(CountClean);
            end;
        TotalCost:=TotalCost + trnsfr[9];
        TotalFallBolt:=TotalFallBolt + FallBoltingCost;
        fileit(2,1);{places fall from events list to list[1]}
        until check or (Lsize[25] = 0);{end of repeat until loop}
    if check then
        next:=trnsfr[1];{sets clock to next fall}
    if (Lsize[25] = 0) and not check then
        next:=FinalTime; {sets clock to end of simulation time}
    end;
    TimeRemain:=TimeRemain + ProdTime;
if TimeRemain >= (next-previous) then {checks if cleanup time is greater
    than time to next fall event}
    begin
    CutN:=0;
    TimeRemain:=TimeRemain - (next - previous);
    end
else
    begin
    CutN:=round((next - previous - TimeRemain)/TimeCut);
    TimeRemain:=0;
    end;
counter:=1;{Initializes loop counter}
while counter <= CutN do {top of the cut generation loop}
    begin
    P:=0.2; {The probability of cut having a roof fall}
{
if (NumberCuts mod 50)=0 then
  writeln(YCord:6:0,RRS:6:2,timealpha[widthI,boltI,RRSI]:6:2,
    timebeta[widthI,boltI,RRSI]:6:2,length[widthI,boltI,RRSI]:6:2,
    lengthb[widthI,boltI,RRSI]:6:2,thickness[widthI,boltI,RRSI]:6:2,
    thicknessb[widthI,boltI,RRSI]:6:2,thicka[widthI,boltI,RRSI]:6:2,
    thickb[widthI,boltI,RRSI]:6:2);}
Inc(NumberCuts);{counter for number of cuts made}
CutLocation(NumberCuts,transf[2]);{finds cut location}
  if random <= P then {see if a fall is possible}
    begin
      RRS:=RRSNow;
      if (BoltCount=1) and (WidthCount=1) then
        writeln(dataRRS,XCord:10:1,YCord:10:1,RRS:10:2);
      if RRS < -1 then
        RRSI:=1
      else if RRS < 0 then
        RRSI:=2
      else if RRS < 1 then
        RRSI:=3
      else
        RRSI:=4;
      transf[1]:=weibull(timealpha[widthI,boltI,RRSI],
        timebeta[widthI,boltI,RRSI])*24+(counter/CutN)
        * (next=previous) + time;
      if transf[1] < FinalTime then
        {checks to see if fall occurs within simulation time}
        begin
          transf[3]:=XCord;
          transf[4]:=YCord;
          w:=gamma(width[widthI,boltI,RRSI],widthb[widthI,boltI,RRSI]);
          {width of roof fall}
          if w > width then {keeps width of fall < width
          of opening}
            w:=width;
          l:=gamma(length[widthI,boltI,RRSI],lengthb[widthI,boltI,RRSI]);
          {length of roof fall}
          transf[5]:=w * l;{area of roof fall}
          transf[6]:=gamma(thicka[widthI,boltI,RRSI],thickb[widthI,boltI,RRSI]);
          {thickness of roof fall}
          FallSize:=w * l * transf[6];
          if FallSize > LargeFall then {saves largest roof fall}
            begin
              LargeFall:=FallSize;
              LargeLength:=l;
              LargeWidth:=w;
              LargeThick:=transf[6];
            end;
          transf[7]:=FallSize;{volume of roof fall}
          transf[8]:=(transf[7]*155/2000)/CleanUp;
          {time needed to clean up}
          if transf[7] > 10 then {check for small falls}
            begin
              fileit(3,25);{places fall in events list}
              if OverFlow=true then
                begin
                  restart;{inputs a new value for number of shifts}
                  goto 666 {goes back to top of program}
                end;
              SumLength:=SumLength+w;
              SumWidth:=SumWidth+l;
              SumThick:=SumThick+transf[6];
            end;
          end;
        end;
    end;
}
end;
if (master[head[25],1] < next) and (Lsize[25] > 0) then
  (checks if any falls from cuts in progress happen
  before thenext scheduled event)
begin
  Remove(1,25);
  FallDelay(ProdTime,FallBoltingCost,check);
  if check then
    begin
      ProdLoss:=ProdLoss + trnsfr[8];
      {sums total production delays}
      Inc(CountProd);
    end
  else
    begin
      CleanTime:=CleanTime + trnsfr[8];
      {sums total cleanup time}
      Inc(CountClean);
    end;
  TotalCost:=TotalCost + trnsfr[9];
  if check and (Lsize[1]>0) then
    begin
      for m:= 1 to maxatr do
        temp[m]:=trnsfr[m];
        {stores the next roof fall to happen in a}
        {temporary array}
      remove(2,1);
      {places last record removed from event list}
      {into trnsfr[]}
      FallDelay(ProdTime,FallBoltingCost,check);
      {subtracts times that were previously added to}
      {production delays and to cleanup delays}
      if check then
        begin
          ProdLoss:=ProdLoss - trnsfr[8];
          CountProd:=CountProd - 1;
        end
      else
        begin
          CleanTime:=CleanTime - trnsfr[8];
          CountClean:=CountClean - 1;
        end;
      TotalCost:=TotalCost - trnsfr[9];
    end;
TotalFallBolt:=TotalFallBolt - FallBoltingCost;
filei(3,25);{puts fall back at top of}
{events list}
for i:=1 tc maxatr do
  trnsfr[i]:=temp[i];{replaces fall in trnsfr[] }
filei(2,1);{places fall in list no. 1}
next:=trnsfr[1];
CutN:=round((next-previous-TimeRemain-
trnsfr[8])/TimeCut);
end
else
  begin
    if check then
      begin
        next:=trnsfr[1];
        CutN:=round((next-previous-TimeRemain)/TimeCut);
      end;
    filei(2,1)
  end;
end;{swap top of events list with new fall if statement}
end;(if random < P statement)
    Inc(counter);(increments counter for CutN do loop)
end;(end of CutN do loop)
time:=next;
    TotalCutN:=TotalCutN + CutN;(sum of total cuts during simulation)
until time >= FinalTime;(End of generation loop)
    GetTime(EndHour,EndMin,EndSeg,Sec100);
    TimeExec;(finds time run took)
    TotalBoltCost:=TotalCutN * CutBoltCost + TotalFallBolt;
    DataOut;(Outputs the data)
    If BoltInc > 1 then
        BoltSpace:=BoltSpace + (EndBoltSpace - BegBoltSpace)/(BoltInc - 1);
        {increments bolt spacing}
end;(end of bolt do loop)
if WidthInc > 1 then
    width:=width + (UpWidth - LowWidth)/(WidthInc - 1);
    {increments opening width}
end;(end of width do loop)
close(data2);(close output files)
close(data3);
close(data5);
close(dataRRS);
pause;(holds output on screen)
    TextBackground(black);(resets screen conditions)
    TextColor(white);
    ClrScr;
end.(program ROCSIM)
unit Dist {contains probability distributions used by ROCSIM};
interface
uses math;
function uniform(a,b:real):real;
function trian(a,m,b:real):real;
function normal(mu,sigma:real):real;
function lognorm(mu,sigma:real):real;
function gamma(alpha,beta:real):real;
function weibull(alpha,beta:real):real;
function expon(beta:real):real;
function beta(alpha1, alpha2,a,b:real):real;
implementation

function uniform(a, b : real) : real;
{provides a sample time on a uniform distribution
 a - lowest value
 b - highest value }
BEGIN
  uniform := random * (b - a) + a;
END;{fuction uniform}

function trian(a,m,b:real):real;
{returns a value from a transformed triangular distribution
 a - lowest value
 m - most likely value
 b - highest value }
VAR
center,r:real;
BEGIN
  center := (m - a) / (b - a);{determines what side of the distn. to use}
  r := random;
  IF r <= center THEN
    trian := a + sqrt(r * (m - a) * (b - a))
  ELSE
    trian := b - sqrt((b - m) * (b - a) * (1 - r))
  END;{function triangular}

function normal(mu,sigma:real):real;
{returns a value from a normal distribution
 mu - mean of normal curve
 sigma - standard deviation }
VAR
v1,v2,d,r:real;
BEGIN
  REPEAT
    v1 := (2 * random) - 1;
    v2 := (2 * random) - 1;
    d := sqrt(v1) + sqrt(v2)
  UNTIL d < 1;
  r := v1 * sqrt((-2 * ln(d)) / d);
  normal := mu + (r * sigma);
END;{function normal}

function lognorm(mu,sigma:real):real;
{returns a value from a lognormal distribution
 mu - mean of the lognormal distribution
 sigma - standard deviation of the lognormal distribution }
var
    mean, SD, y: real;
beging
    mean := ln(sqrt(mu)/sqrt(sigma + sqrt(mu)));
    SD:=sqrt(ln((sqrt(sigma)+sqrt(mu))/sqrt(mu)));
    y:=normal(mean,SD);
    lognorm:=exp(y);
end; {function lognorm}

function expon(beta:real):real;
{returns values from an exponential distribution}
beging
    expon := -beta*ln(random);
end; {function expon}

function gamma(alpha, beta:real):real;
{returns a value from a gamma distribution
  taken from Simulation Modeling and Analysis p. 256-257
  alpha - shape parameter
  beta - scale parameter}
var
    p, y, u1, u2, v, z, w, a, q, theta, d, b, test, gamma: real;
    finished: boolean;
begin
    if alpha < 1 then
        begin
            finished := false;
            repeat
                b := (exp(1)+alpha)/exp(1);
                p := b*random;
                if p <= 1 then
                    begin
                        y := power(p, 1/alpha);
                        u2 := random;
                        if u2 <= exp(-y) then
                            begin
                                gamma := y;
                                finished := true;
                            end;
                    end
                else
                    begin
                        y := -ln((b-p)/alpha);
                        u2 := random;
                        if u2 <= power(y, alpha-1) then
                            begin
                                gamma := y;
                                finished := true;
                            end;
                    end;
            until finished;
        end
    else if alpha = 1 then
        gamma := expon(1)
    else
        begin
            a := power((2*alpha-1), -0.5);
            b := alpha-ln(4);
            q := alpha+1/a;
            theta := 4.5;
            d := l+ln(theta);
        end;
finished:=false;
repeat
  u1:=random;
  u2:=random;
  v:=a*ln(u1/(1-u1));
  y:=alpha * exp(v);
  z:=sqr(u1)*u2;
  w:=b+q*v-y;
  test:=w+d-theta*z;
  if test >= 0 then
    begin
      gamma1:=y;
      finished:=true;
    end
  else
    begin
      if w >= ln(z) then
        begin
          gamma1:=y;
          finished:=true;
        end;
    end;
  until finished;
end;
{function gamma}

function weibull(alpha,beta:real):real;
{ alpha - shape parameter
  beta - scale parameter }
var
  u:real;
begin
  u:=random;
  weibull:=beta*(power(-ln(u),1/alpha));
end;{function weibull}

function beta(alpha1,alpha2,a,b:real):real;
{returns a value from a beta distribution
  alpha1,alpha2 - shape parameters
  a - lower range of interval
  b - upper range of interval }
var
  y1,y2,x:real;
begin
  y1:=gamma(alpha1,1);
  y2:=gamma(alpha2,1);
  x:=y1/(y1 + y2);
  beta:=a + (b - a) * x;
end;{function beta}
end.[Unit Dist]
UNIT Math {math routines used by ROCSIM};
INTERFACE
USES crt;
function tan(x: real) : real;
function cot(x: real) : real;
function power(x, n : real) : real;
function arcsin(y:real):real;
function arccos(y:real):real;
function StandDev(n:integer;avg,sumX2:real):real;
function norm(mu, sigma, x:real):real;
function integration(mu, sigma, lower, upper:real):real;
CONST
    pi=3.141592653589793;
IMPLEMENTATION

function tan(x : real) : real;
{ Note that x must be in radians }
Begin
    if cos(x) = 0.0 then exit;
    tan := sin(x)/cos(x);
End; {function tan}

function cot(x : real) : real;
{x must be in radians }
Begin
    if sin(x) = 0.0 then exit;
    cot := cos(x)/sin(x);
End; {function cot}

function arcsin(y:real):real;{output in radians}
begin
    arcsin:=arctan(y/sqrt(1-sqr(y)));
end; {function arcsin}

function arccos(y:real):real;{output in radians}
begin
    arccos:=pi/2 - arctan(y/sqrt(1-sqr(y)));
end; {function arccos}

function power(x, n : real) : real;
var y : real;
{ Function to raise x to the nth power
  n can be any value. 
  x can be any positive value or negative value when n is an actual
  integer value (only expressed in real). Any invalid input will
  result in a value of zero. }
begin
    if x = 0.0 then
        begin
            power := 0.0;
            exit;
        end;
    if (x > 0.0) or (Frac(n) = 0.0) then
        y := exp(n*ln(abs(x)))
    else
        begin
            }
writeln('Error, power is returned as zero');
y := 0.0;
end;
power := y;
if (x < 0.0) and (int(n/2.0)*2.0 <> n) then power := - y;
end; { of function Power }

function StandDev(n:integer;avg,sumX2:real):real;
{ finds standard deviation given the number of values,
  avg. of the values and the sum of the values squared }
var
  variance:real;
begin
  variance:=(1/(n-1)) * (sumX2 - n * sqr(avg));
  StandDev:=sqrt(variance);
end; { function StandDev }

function norm(mu,sigma,x:real):real;
{ returns y value from a normal dist }
begin
  norm:=(1/(sigma*sqr(2*pi))) * exp(-0.5*sqr((x-mu)/sigma));
end; { function norm }

function integration(mu,sigma,lower,upper:real):real;
{ finds area under normal curve using trapezoidal rule }
var
  pieces,i:integer;
  delta,esum,psum,y:real;
begin
  pieces:=200; { increase for greater accuracy }
  psum:=0;
  delta:=(upper - lower)/pieces;
  esum:=norm(mu,sigma,lower) + norm(mu,sigma,upper);
  for i:=1 to pieces-1 do
    begin
      y:=lower + i * delta;
      psum:=psum + norm(mu,sigma,y);
    end;
  integration:=(esum + 2 * psum) * delta/2;
end; { function integration }

END. { end of Unit MATH }
VITA

Richard Fraher was born on February 27, 1958, in Great Falls, Montana. He graduated with a B. S. in Civil Engineering with a Mining Option from the University of Wyoming in 1988. 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. He has worked at a variety of jobs in hard rock mines, trona mines and coal strip mines in Colorado and Wyoming.