A COMPUTERIZED SYSTEM FOR ECONOMIC ANALYSIS OF AN ELECTRIC UTILITY'S POWER GENERATION EXPANSION PLANS

A GRADUATE PROJECT

BY

NICOLAS CURIEL, JR.

SUBMITTED TO THE COLLEGE OF SCIENCE AND TECHNOLOGY CORPUS CHRISTI STATE UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE

JUNE, 1982

MAJOR SUBJECT: COMPUTER SCIENCE
A COMPUTERIZED SYSTEM FOR ECONOMIC ANALYSIS OF AN ELECTRIC UTILITY'S POWER GENERATION EXPANSION PLANS

A GENERATION PROJECT

BY

NICOLAS CURIEL, JR.

APPROVED AS TO STYLE AND CONTENT BY:

[Signatures]

(JAHRMAN OF COMMITTEE)

[Signature]

(MEMBER)

[Signature]

(MEMBER)

JUNE, 1982
# TABLE OF CONTENTS

PREFACE  .................................................. iv

Chapter

I. INTRODUCTION ........................................... 1

Scope and Purpose ........................................ 1
Statement of Problems ..................................... 1
Rationale for Project ...................................... 2
Procedure .................................................. 3
Resources .................................................. 5
ROSCOE Introduction ...................................... 6

II. SYSTEM RELIABILITY ..................................... 15

System Operation .......................................... 15
Loss-Of-Load Probability .................................. 16
Load Model Construction ................................... 17
LOLP Calculation .......................................... 35
Input Layout .............................................. 36

III. PRODUCTION COSTING AND FINANCIAL SIMULATION .... 74

Introduction .............................................. 74
Uses of Model ............................................ 75
Model Assumptions ....................................... 77
Utility Cash Flow ......................................... 83
Program Operation Control Codes ....................... 86

IV. FIXED CHARGE RATE CALCULATION .................... 250

Annual Fixed Charges .................................... 250
Revenue Requirements .................................... 251
Input Layout .............................................. 252

V. SUMMARY .................................................. 264

Choice of Analysis Method ................................ 264
Revenue Requirement Method .............................. 266
Input Layout .............................................. 269

LIST OF REFERENCES ....................................... 287
PREFACE

This graduate project links economics and the technology of electric utility power generation. This link is needed when performing economic comparisons of alternate forms of electric energy generation. The function of economic analysis is to estimate how things will turn out. This requires that the economic analysis faithfully represent actual operations, technical as well as financial.

Economic analysis is the process by which the impact of a proposed action on financial results of an organization's operations is measured. Its purpose is to supply one of the bases for management decision.

The author wishes to express his appreciation to Diana Hollingshead for her diligence, competence and patience while preparing the typescript and in performing various editorial tasks. Similar appreciation is expressed to Melissa Matula for her assistance during program development and execution.
CHAPTER I

INTRODUCTION

SCOPE AND PURPOSE

This graduate project partially fulfills the requirement of the Master of Science with Major in Computer Science Degree from Corpus Christi State University.

This graduate project developed a computerized system for economic analysis of an electric utility's power generation expansion plans. Economics of electric utility power generation involves many things from basic accounting principles to electric system reliability. The problem presented here is similar to problems encountered in an actual work environment.

This project produced one of many possible computerized systems for economic analysis of generation expansion plans. This system may be used to analyze corporate and company expansion plans. This system is general in nature and does not attempt to simulate all electric utility procedures to the same degree.

STATEMENT OF PROBLEMS

To provide a computerized system which will perform economic analysis of an electric utility's alternative power generation expansion plans accomplishing the following:
1. Incorporate the revenue requirement method of economic analysis which is widely used in the industry.

2. Provide a comparison of alternative generation expansion plans which are truly comparable based on their long term system reliability.

3. Use capital (investment) costs and annual production costs (fuel, operation, and maintenance) in all analyses.

4. Provide analysis using financial simulation which will reflect the impact of various plans on the company's books.

5. Provide reports by all components as well as a summary report which will show all the data used as input and the preferred plan based on economics.

6. Perform the economic analysis using company computer resources. These resources are listed below.

At least one computer program supports each section of this system. Successful implementation of all computer programs is necessary for proper execution of this system. This project contains the results of the system as well as computer program documentation.

RATIONALE FOR PROJECT

Today more than ever generation expansion decisions made by electric utilities have tremendous economic consequences. As costs continue to increase regulatory agencies, as well as utility customers, are questioning the economics of generation expansion plans. A utility must be able to show that it has done enough analysis to assure everyone concerned that the right decision was made.
The field of electric utility generation economics has developed over the past twenty-five years from the simple calculation of unit generating costs in mills per KWH to complex analysis of the operation and costs of complex systems of generating units.

For these reasons along with others which became apparent as this project evolved, a system for economic analysis of utility generation expansion plans was developed and will be submitted to company planning engineers. They in turn will use the system as a tool for generation planning.

PROCEDURE

The economic comparison of alternative facilities almost invariably involves alternates whose capital investments and future annual production costs are both different. The problem is to develop for each alternate a single number which can serve as a figure of merit, combining the effects of the cash outlays for capital investment and for annual production expenses. Since the investment has financial implications which extend to the end of the life of the facilities, the analysis must encompass at least a major fraction of that life and, ideally, the whole life.

In some kinds of studies it is possible to make what may be called a direct unit comparison, one without specific reference to the operation or costs of the rest of the generating system. In direct unit analyses, costs are calculated as though the generating system consisted only of the one unit in question.
Direct unit analysis should not be used for evaluation of alternate unit sizes, units of markedly different heat rates or reliability characteristics, or units whose fuel prices or escalation factors are different. In such cases, a total system cost analysis is essential, one which by simulation or other means recognizes that the operation of the system's other generating units will be affected differently by different alternates. Total system cost effect is the ultimate criterion.

The essence of total system analysis is simulation of system operation with respect to both reliability and economy of electric power supply. The simulation is carried out over a period of time which is estimated to include most of the economic impacts of the alternates being studied. A study of not more than twenty years is generally accepted in the industry.

The total system analysis procedure, see Figure 1, is the subject of this graduate project. The major sections of this system are described in detail in the subsequent chapters. Of all the chapters, Number Three contains the most information. The main part of this project, the Production Costing and Financial Simulation Model, is described in great detail in Chapter Three.

Each subsequent chapter starts with introductory material followed by some discussion of the program logic then by a listing of the computer program(s), input and output.
RESOURCES

Company and corporate resources were used for this project. An IBM 3032 with 8,000K bytes of memory was used for all processing. All access was through IBM 3270 type CRT terminals via dedicated telephone lines. The computer operates in an OS/MVS environment. All jobs were submitted in batch mode.

All programs were written in FORTRAN. A few assembler routines were used throughout the project. These were obtained from the company program library and are not included as part of this project. All programs and subroutines were written in IBM FORTRAN and can be used on other systems with minimal revisions to the source code.

There are numerous software packages which can access the IBM 3032 computer. ROSCOE from Applied Data Research, Inc. was selected for use because of its ease of use and fast response time. The following eight pages, taken from the ROSCOE User's Manual, serve as an introduction to ROSCOE.
INTRODUCTION

ROSCOE is a comprehensive software system for online program development with multiple interactive support functions.

This introduction is an overview of the system. It defines terms and provides fundamental information on the topics described more fully in later sections.

The AWS

The AWS (Active Work Space) is a temporary work area in which you enter and edit data. Data is kept in the form of 80-character records. Records (or lines) in the AWS are ordered by a line number which is not part of the data. You can refer to each line either by its line number (absolute address) or by its relative position within the AWS (relative address). ROSCOE always maintains the records in their correct, sequential order.

Part I, Section 3 describes the AWS functions and explains absolute and relative addressing.

The Library

The ROSCOE library is a permanent data storage area. You enter data and edit it in the AWS, then store it in the library. If you want to make changes to the data set after it is in the library, you bring it back into the AWS. The data set remains intact in the library while you are working on a copy in the AWS.

The library is shared by all ROSCOE users, but appears to each user as his own private library. Each user's data sets are uniquely identified by ROSCOE, so that many users can have data sets with the same name. ROSCOE protects the integrity of your data sets by prohibiting other users from deleting or modifying them. Other users cannot even look at your data sets unless they know your key or prefix. Part I, Section 4 describes data set naming conventions and library functions.

Processing Modes

During a ROSCOE terminal session, you will always be in one of four modes:

- Command mode, to enter commands.
- Input mode, to enter data into the AWS from the terminal.
- Editlist mode, to display the AWS for on-screen modification (3270 terminal only).
- Processor mode, to execute ROSPROCs and Monitor routines.

Part I, Section 5 further describes the modes.
**ROSPROCs and Monitor Routines**

ROSPROCs are conversational procedures which you can write to perform a variety of functions: They have access to the terminal, to the AWS, the library, and special ROSPROC data areas. ROSPROCs consist of a set of procedural commands and any ROSCOE command which you might enter at the terminal. Once you have written a ROSPROC and stored it as a library data set, you can execute it with the DO command. Part 1, Section 6 describes ROSPROC facilities.

Your site management will have installed a number of ROSCOE Monitor routines for your use and probably will write additional ones. The Monitor routines supplied by ADR include language and JCL Syntax Checkers and data set management aids. You can execute Monitor routines with the RUN or VERIFY commands (which are synonymous). ADR-supplied Monitor routines are described in the RUS User Guide.

**Storing and Retrieving Data Sets Offline**

Library data sets are stored in a compressed format which is incompatible with normal OS access methods. ROSCOE provides two utilities, ROSDATA and ROSCPONY, for storing and retrieving library data sets in batch.

**Remote Job Entry**

You can submit a job for background execution from the AWS or from the library, using the SUBMIT command.
Remote Job Output

ROSCOE offers three methods for viewing job output at your terminal:

- The Output routine (a ROSCOE Monitor routine), which directly accesses the system queues to display the job's output at the terminal.

- The Post-Processors, which retrieve and format information which you have requested from a job's execution and store it as a library data set.

- The ROSCOE Writer, which retrieves JCL and system messages and stores them as an "output writer data set" on the ROSCOE library.

One or more of these methods may be available to you; check with your management.

Part I, Sections 8 and 9 describe the Post-Processors and Writer, respectively; the Output routine is described in the RUS User Guide.

Terminal Functions

In this manual, we refer to two generic terminal types--3270s and "typewriter devices." Typewriter devices are 2741s and teletypes (i.e., any device which is handled like a teletype).

There are four logical terminal functions--RETURN, ATTN, BACKSPACE, and TAB--which are activated differently from different terminals. Their logical functions are as follows:

RETURN  sends data from the terminal to ROSCOE or transmits a command to ROSCOE.

ATTN    stops the current processing.

BACKSPACE deletes characters from a line before transmitting the line to ROSCOE.

TAB      formats text on a line at the terminal.

Except in Appendices B and C, which present device-dependent information, all discussion in this manual is on the logical, device-independent level.

For 3270s, please note the effect of the following actions when ROSCOE is in the midst of processing a command (such as a LIST):

- ENTER acts as a RETURN (processing continues).

- CLEAR acts as an ATTN (processing stops).

- Entering another command acts as an ATTN.
User Status Commands

ROSCOE has three commands which display user status, AWS status, or time and date. They are STATUS, AWS, and TIME.
LANGUAGE CONVENTIONS

Formal Syntax

All ROSCOE commands have the form:

COMMAND operands

where:

COMMAND represents the command name. The name may be the full form, a short form (a 1-letter abbreviation), or the short form (a 3-letter abbreviation).

operands define the scope of the operations.

In all ROSCOE documentation, we use upper-case alphabetics to represent a word which must be entered to ROSCOE exactly as shown; lower-case alphabetics represent a variable for which you must supply an appropriate value. An underscore indicates the allowable short or alternative form. For example:

FETCH dsn

The command is FETCH; its short form is F; you must supply, in place of dsn, the name of a ROSCOE library data set.

We use brackets to indicate that an operand is optional (nested brackets mean that you can specify one operand only if you specify the other). Slashes indicate any special character (usually used as delimiters of strings). For example:

EDIT [F] /string/ [p [q]]

The command is EDIT; its short form is E. The operand F is optional, however you must specify a character string bounded by delimiters. You may supply p; you may supply q only if you have supplied p.

More complete and specific information on ROSCOE language conventions is given in the next two sections; symbols and special notations are described in Part II, Section 1.4.
Command Language Character Sets

The characters that you may use in ROSEOE commands fall into six categories, each of which is described below:

Separators: Comma (,) and space

Reserved character: Asterisk (*)

Alphabets: a-z
A-Z
@ # $

Numerics: 0-9

Special characters: Any symbols which appear on a standard keyboard and which are not part of any of the four sets defined above. Note that the following are not special characters:

, # * $ @

Binary characters: Any EBCDIC bit combinations which cannot be entered from a standard keyboard without multi-punches or control modifications and any which are not part of any of the five sets defined above.

Use of the Character Sets

The rules for the use of the above characters sets are as follows:

- Separators

  -- Separate the operands in a command. The following command forms perform identical functions:

  LIST PROG1,50,900
  LIST PROG1 50 900

  -- Separate the command from its operands. If you use the long form of a command (e.g., DELETE) or its alternative form (e.g., DEL), a separator is optional; if you use the short form (e.g., L), a separator is required. The following command forms perform identical functions:

  LIST 10
  LIST10
  L,10
  L 10
- Reserved Character

An asterisk (*) is a reserved character which has one of three meanings, depending upon the ROSCOE command with which you use it.

-- It refers to the name of the last data set fetched, as in:

   UPDATE *

-- It refers to the name of the ROSPROC currently in execution, as in:

   DO * 400

-- It refers to the relative address of a line in the AWS, as in:

   DELETE *

Relative addressing is fully described in Part I, Section 3.3.2.

Another reserved character is the "double comma" (,,), which you can use only once in a command as an operand to represent the last line in the AWS. For example:

   LIST 5,,

This form means list the contents of the AWS beginning at line 5 and continuing to the end.

Explanation of Notation

The special symbols that are used as operands, or to surround operands in ROSCOE commands are described below:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ]</td>
<td>the enclosed is optional.</td>
</tr>
<tr>
<td>/</td>
<td>any special character. It may be used to delimit a string, in which case the special character you use must not appear within the string.</td>
</tr>
<tr>
<td>{ }</td>
<td>the enclosed operands are alternatives from which you must choose one and only one.</td>
</tr>
<tr>
<td>*</td>
<td>is a ROSCOE reserved character which has one of three meanings, depending upon the command with which it is used: It may refer to the name of the last data set fetched, the name of the ROSPROC currently in execution, or the current position in the AWS.</td>
</tr>
<tr>
<td>cond</td>
<td>is a condition: EQUAL, UNEQUAL, HIGH, LOW, or SET.</td>
</tr>
<tr>
<td>dsn</td>
<td>is the name of a ROSCOE library data set.</td>
</tr>
</tbody>
</table>
is an increment value used for determining line numbers in the AWS.

is the length of a string or field.

represent a range of lines in a library data set: \( m \) is the number of the line at which a library operation will begin; \( n \) is the number of the line at which the operation will end.

If you omit \( n \), the operation is limited to line \( m \). However, if you omit \( n \) in a DO, IF or CALL command, the operation begins at line \( m \) and continues to the end of the data set.

is the destination of an operation. It can be represented in any one of the following ways:

- \( " \) indicating the last line of the AWS.
- \( T \) indicating the first (top) line of the AWS.
- \( B \) indicating the last (bottom) line of the AWS.
- \( * \) indicating the current line of the AWS.
- \( v \) indicating a specific line number in the AWS.

If you omit \( o \), the default destination is the last line of the AWS.

represent a range of lines in the AWS: \( p \) is the number of the line at which the AWS operation will begin (it can be a line number or the positional expression); \( q \) is the line at which the operation will end.

If \( p \) is a line number, \( q \) is also a line number. If \( p \) is a positional expression, \( q \) is a line count.

If you omit \( q \), the operation is limited to line \( p \).

If you omit \( p \) and \( q \), a default range of lines is assumed: The default range is the entire AWS if the mode is TOP (i.e., SET MODE TOP); the default range is from the current position to the end if the mode is POS (i.e., SET MODE POS).

represent a range of columns: \( s \) is the number of the column at which an operation will begin; \( e \) is the number of the column at which an operation will end. Both \( s \) and \( e \) must be in the range 1-80.

represent the starting column and the length.

represents any sequence of characters; blanks are valid.

is a numeric.
Fig. 1 TOTAL SYSTEM ANALYSIS PROCEDURE