DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AN INTERACTIVE LIFE CYCLE COST FORECASTING TOOL

THESIS

John A. Gibson, IV
Captain, USAF

AFIT/GOR/ENS/91M-5

REPORT DOCUMENTATION PAGE

Form Approved
OME No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1991	Master's Thesis

4. TITLE AND SUBTITLE
An Interactive Life Cycle Cost Forecasting Tool

5. FUNDING NUMBERS

6. AUTHOR(S)
John A. Gibson, IV, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GOR/ENS-91M-5

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Enhancements were applied to an existing tool designed to estimate life cycle costs. These enhancements provide greater ease of data manipulation, additional options in developing models, and the ability to make modifications to suit the needs of the user. Life cycle costs can be generated quickly and easily. Analysis proved the amount of work done by the computer could be reduced with no loss of information. Additionally, the paths available to the user are shown graphically through flow charts -- aiding in the learning process. The efforts of the user can now be directed towards developing models instead of spending those efforts on deciphering how to make the tool work.

14. SUBJECT TERMS
Life Cycle Cost Models, Computer Programming, Cost Estimating Relationship, Random Number Generation

15. NUMBER OF PAGES
139

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

AFIT/GOR/ENS/91M-5

AN INTERACTIVE LIFE CYCLE COST FORECASTING TOOL

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

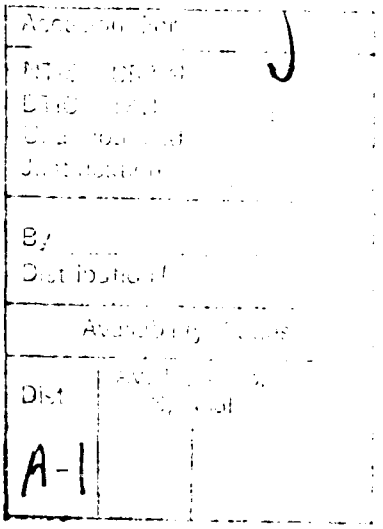
In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

John A. Gibson, IV, M.S.

Captain, USAF



March 1991

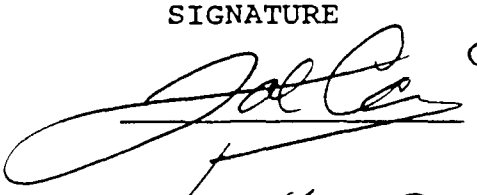
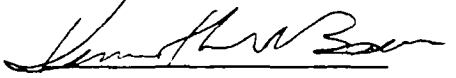
Approved for public release; distribution unlimited

THESIS APPROVAL

STUDENT: John A. Gibson, IV CLASS: GOR-91M

THESIS TITLE: An Interactive Life Cycle Cost Forecasting Tool.

DEFENSE DATE: 25 February 1991

COMMITTEE:	NAME/DEPARTMENT	SIGNATURE
Advisor	Dr Cain/ENS	
Reader	MAJ Bauer/ENS	

Preface

The purpose of this study was to enhance Dr Cain's Monte Carlo simulation for the forecasting of life cycle costs. These enhancements provide greater ease of data manipulation, additional options in developing models, and the ability to make modifications to suit the needs of the user. Life cycle costs can be generated quickly and easily.

I chose this research topic because of my interests in economics and computer programming. I am indebted to Dr Cain for allowing me to work at my own pace and incorporate my ideas into the project.

I would also like to thank Major Morlan who in 15 minutes solved three days of computer debugging -- he didn't realize how much effort he saved me. I would also like to thank Major Bauer and Professor Reynolds for providing me with enough of a statistics background to ensure the rigor required was applied.

My deepest thanks go to my wife Sandy who put up with the never ending procession of late nights and incessant complaining.

John A. "Hoot" Gibson

Table of Contents

List of Figures

List of Tables

Abstract

Enhancements were applied to an existing tool designed to estimate life cycle costs. These enhancements provide greater ease of data manipulation, additional options in developing models, and the ability to make modifications to suit the desires of the user. Life cycle costs can be generated quickly and easily.

Analysis proved the amount of work done by the computer could be reduced with r loss of information. Additionally, the paths available to the user are shown graphically through flow charts -- aiding in the learning process. The efforts of the user can now be directed towards developing models instead of spending those efforts on deciphering how to make the tool work.

AN INTERACTIVE LIFE CYCLE COST FORECASTING TOOL

I. Introduction

Background

Numerous programs within the United States require federal funding in order for them to survive -- one of them being the Department of Defense (DoD). "The acquisition of major weapon systems is a costly and long-term effort, often requiring the commitment of billions of dollars over a period of years" (COEA:2).

The funding necessary for a single weapon system can span more than two decades (like the B-52 bomber and C-141 transport aircraft). This occurs because weapon systems cannot just be bought off the shelf, used for a certain period of time, and then thrown away. Future weapon systems must be tested and modified to assure the American public their money is being spent on projects capable of meeting and suppressing the threat the systems were designed against. Once a weapon system has been procured, additional funding is needed to maintain and operate it. In other words, money is constantly required for the efficient

functioning of a weapon system throughout its entire life, hence the term life cycle cost (LCC). The AF Systems Command Manual 173-1, Cost Estimating Procedures, defines LCC.

> LCC is the government's total cost of owning a system, subsystem, or component over its full life. It includes development, production, operating, and support costs. (AFSCM:5-1)

"Since life cycle costing includes all the phases of a program's life, a more realistic look at the budgetary effect is achieved" (Sumner:3). A weapon system whose anticipated procurement cost is the lowest of all systems competing for selection may not be the cheapest in the long run. If its research and development or maintenance costs are significantly higher than another system's, it may not be the best choice from the cost perspective.

Another important factor deals with the time when money for a weapon system is to be spent. The choice of whether to spend millions of dollars today versus delaying the payment until the following year can be extremely critical in deciding when a weapon system becomes available for use (or available at all). The value of money today is always more than the value of the same dollar amount of money in the future. Therefore, if the expected return on the funding of a weapon system is below an alternative's, and

the price of the weapon system is not anticipated to significantly increase over the following year, then it may be in the best interest of the United States government to delay funding of the weapon system until next year. As Sumner stated,

> Since the stages of a program occur chronologically, LCC allows for the 'timing' of the money spent. Therefore, the timing of money spent by competing programs can be a very significant factor. (Sumner:3)

Objective

The purpose of this research was to enhance Dr Cain's Monte Carlo simulation for the forecasting of life cycle costs. The enhancements include (but are not limited to): providing the user with a choice of distributions in aiding him to estimate the expected life cycle cost of the system in question, modifying the software structure to accommodate better data input, and creating documentation in the form of detailed flow charts. An additional goal of this research was to determine if the number of random numbers currently generated for each random variable and cost estimating relationship (CER) was necessary.

Subobjectives and Methodologies

Following are the study's subobjectives and the methods used to achieve them.

Data Input Modification Modify the program so data can

be changed per element instead of having to reenter the entire data set.

Upon entering data into Dr Cain's program, the only way a change could be made to one or a few of the inputs was to completely reenter the entire data set. This input procedure was modified by first saving the original data set into two external files, and then prompting the user if any changes to one or more of the data needed to be made.

Software Verification and Validation A test model provided by Dr Cain was used to verify and validate the LCC.FOR computer code.

Documentation Develop detailed flow charts to graphically present what the computer code does, and also increase the user-friendliness of the software.

No flow charts existed for Dr Cain's program. Dr Cain felt flow charts would be a good idea to help eliminate the need of having just the computer code available to obtain an understanding of how the program LCC.FOR operates. Detailed flow charts were developed to graphically represent the action taking place in the computer code.

Alternative Distribution Incorporate the triangular distribution to complement the existing beta distribution in generating random variables.

The original program only allowed for using the beta distribution in calculating random variables and CER's. The

addition of the simpler triangular distribution provides the user with an alternative to the somewhat complicated and less intuitive beta distribution.

Random Variable Sampling Reduction The program LCC.FOR currently obtains 750 random samples in estimating the life cycle cost of a system. Determine if any information is lost by reducing the number of samples taken to 500 or 250.

The methodology used in assessing this subobjective was done by first testing whether the data samples taken were normally distributed. The next steps involved comparing the means and variances of runs generated from both sample sizes.

Program Modification Include a list of instructions to the user of the program LCC.FOR so modifications can be made to the program to suit their individual needs.

A list of pertinent changes was presented so any user of the program LCC.FOR could incorporate more cost elements and/or modify the number of samples generated.

II. Literature Review

Introduction

The purpose of this chapter is to review literature pertinent to the forecasting of life cycle costs (LCC's). The areas of literature that must be reviewed for this effort include theoretical life cycle costing methods and random number generation.

Life Cycle Cost Models

As stated earlier, "life cycle costs reflect the cumulative costs of developing, procuring, and operating a system, subsystem, of component over its full life" (COEA:6; AFSCM:5-1). The importance of providing accurate cost estimates over the life of a system cannot be underestimated. Cost is sometimes the only factor that can be directly compared between competing projects. The Program Analysis and Evaluation (PA&E) Office of the Assistant Secretary of Defense states, "Cost estimates are as important as operational effectiveness measures in a cost and operational effectiveness analysis" (COEA:15). The literature suggests there are three types of LCC models (COEA:15; Seldon:23):

1. Estimating by analogy
2. Parametric methodology

3. Engineering, or bottom up estimates

"Estimates by analogy and the parametric methodology are both 'top down' methods, because they examine the program as a whole" (Seldon:16). The engineering estimate is a bottom up method because it is comprised of detailed costs of every component that is contained within the whole program (Seldon:16).

Analogy Estimates

Estimating by analogy is the easiest of the three methods because it requires the least amount of detail. "Analogy estimates are conducted by adjusting the known costs of existing systems similar to the one in question to arrive at cost projections" (COEA:15). In other words, the cost of a current project is related to that of a previous one. Of course, if data are available on more than one past program, all data will be included in the estimate by providing a general relationship on those characteristics (both similar and different) of all systems concerned (Seldon:15-16).

Analogy estimates are usually conducted very early in the development of a future system to try to gauge the approximate order of magnitude of the expected total cost. "By starting the LCC analysis at the conceptual phase, it is possible to ask whether it would be wiser to produce a new system or to modify an older existing one" (Seldon:15). Table 1 contains an example of how analogy estimates are

conducted by comparing a new program for airborne electronic equipment against an already existing system (taken from Seldon's book on Life Cycle Costing: A Better Method of Government Procurement).

If conducted early in the conceptual phase, the method of estimating by analogy provides management with a quick and easy way to determine if the cost of a new project is prohibitive or not. This method allows periodic updates to be made as they occur, with relatively minor effort. "Surprisingly, such estimates are usually accurate if all the significant changes between programs are understood and accounted for" (Seldon:25).

Parametric Methodology

"Parametric cost analysis involves the development and utilization of estimating relationships between historical costs and physical and/or performance characteristics of a system" (AFSCM:5-4; COEA:15; Sumner:7). The system characteristics (such as aircraft speed or the number of maintenance personnel required to support a missile's guidance shop) are commonly referred to as parameters whereas, "the historical costs reflect the impact of system growth, engineering changes, program stretchouts, and any other cost, schedule, or performance difficulties encountered in comparable programs" (AFSCM:5-4). If the characteristics of a current or previous system can be

Table 1

COST ESTIMATE BY ANALOGY
(in millions of $)

Cost Element	Base Program	New Program
Systems engineering and program management	2.0	2.2 (a)
Design	8.0	11.2 (b)
Prototype fabrication and material	1.5	1.1 (c)
Flight and laboratory test	3.0	1.5 (d)
Total	14.5	16.0

Notes:

(a) Systems engineering and program management for the New Program are similar to those of the Base Program; the 10% increase in the New Program is due to larger task of design monitoring.

(b) In the New Program, the additional design task of the moving target indicator adds 20%, and higher performance requirements will require another 20%, for a total of 140% of the cost of the Base Program.

(c) The New Program requires only two prototypes versus three for the Base Program; tooling and other fixed costs were about $300,000 for the Base Program, with each prototype at $400,000.

(d) The New Program requires only one aircraft model qualification; flight test personnel estimate the cost at one-half that of the Base Program.

(Seldon:24)

quantified by some statistical means (linear regression for example), a parametric relationship or cost estimating relationship (CER) can be developed. The purpose of a CER

is to attempt to predict the future based upon information received from past occurrences (MECA:79). A CER accomplishes this by "transforming the problem from one of estimating dollars to one of estimating a more familiar and more accessible variable" (Seldon:25). A major concern with using CER's as a forecasting tool is the amount of confidence that can be placed in estimates derived from the CER.

The DoD Life Cycle Costing Guide For System Acquisitions (taken from Sumner's thesis) lists several advantages associated with using CER's to estimate costs. The advantages are as follows:

1. Cost estimates are based on general system characteristics, no detailed information is necessary;
2. Model is very fast and easy to use;
3. Model is resistant to user bias;
4. Since parametric statistics are used in generating the forecasts, confidence intervals (CI's) can be placed on the forecasts. (Sumner:8)

The concern regarding how much confidence can be placed in CER estimates is addressed by the fourth advantage.

Figure 1 shows an example of a parametric CER model taken from Sumner's thesis. Notice the costs are calculated from general characteristics and some error is associated with each equation.

Engineering or bottom up methodology

The engineering method of estimating the total cost of a system is commonly referred to as the bottom up method. To conduct an engineering estimate, a clear description of the task to be completed must be understood by every level involved in the project. The levels involved then estimate the cost of doing the particular task assigned to it by using specific hardware-to-cost relationships (Seldon:32; Sumner:8-9). The sum of these estimates represent the total cost of the project, hence the term 'bottom up'. "Obviously this takes more information, and indeed the DoD does not recommend this as a method for preliminary work since the level of detail needed is usually obtained after many

COST REPRESENTATION FOR ONE C-130 USING THE PARAMETRIC METHODOLOGY

CER's

Airframe = 200,000 + 75 * X1 + e
Engines = 2,000 + 63 * X2 + e
Electronic = 530 + 200 * X3 + e
Manpower = 300,000 + 400,000 * X4 + e
Operating = 500,000 + 12,000 * X5 + e

where

X1 = airframe weight
X2 = thrust
X3 = number of radios
X4 = number of crewmembers
X5 = yearly flying hours
e = error, iid N(0,MSE)

Figure 1 Parametric CER cost model example (Sumner:9)

crucial decisions (based on cost) have been made" (Sumner:9).

The engineering method is similar to the parametric method because the system is broken down into cost components and CER's may be used to determine the cost of each component (Sumner:8). One main advantage of the engineering method is the costs tend to be very accurate due to the detail provided by each level (Seldon:34; Sumner:9).

Random Number Generation

Overview

The most common method of creating random numbers is the Monte Carlo method. The term Monte Carlo comes from the city located in Monaco famous for its casino. The roulette wheel, a popular game of chance at most casinos, is one of the simplest mechanical devices for the generation of random numbers, hence the name, Monte Carlo. "The Monte Carlo method is a numerical method of solving mathematical problems by means of random sampling" (Sobol:8).

The Monte Carlo method is comprised of a unique feature. In I. M. Sobol's book, The Monte Carlo Method, he states;

> This feature is the simple structure of the computation algorithm. As a rule, a program is prepared to perform only one random trial. This trial is repeated N times, each trial being independent of all others, and the results of all trials are averaged. Thus the Monte Carlo method is sometimes called the 'method of statistical trials'. (Sobol:10)

Sobol continues his discussion of the Monte Carlo method by stating that Monte Carlo is not a good method in solving problems that require a high degree of accuracy. His reason for this is due to the error of the calculations being proportional to the square root of D/N, where D is a constant and N is the number of trials run. Sobol points out that to reduce the error by a factor of 10, 100 additional runs would have to be conducted, which may not be feasible due to time and cost constraints (Sobol:10).

The literature states there are three methods of obtaining random numbers: 1. tables of random numbers, 2. random number generators, and 3. the method of pseudo random numbers (Pritsker.716-717; Sobol:24). A random number table is just that, a list of random numbers. The list of numbers is derived by running an experiment designed to generate a random number, writing it down, and then repeating the experiment. This list is then read into the computer to be used as data for the problem at hand (Pritsker:716). The problem with this method is the amount of memory taken by the list of random numbers, and the additional time it would take to retrieve a number from the computer's memory (Pritsker:716; Sobol:26).

An example of a random number generator would be "to employ a physical device such as a vacuum tube which generates random noise" (Pritsker:716). This would operate

by counting the number of times the noise generated in the vacuum tube exceeded a predetermined threshold over a specified period of time. If the number counted was even, a one would be assigned, whereas if the number counted was odd, a zero would be recorded (Sobol:26). A register in the computer' memory would be designated to receive this collection of zeros and ones to provide a binary result for use when a random number is needed. The problem with this method is the results would not be reproducible, therefore model verification and controlled experimentation would be virtually impossible (Pritsker:716; Sobol:27).

The third, and most widely accepted method of generating random numbers is the method of pseudo-random number Pseudo-random numbers are calculated by means of some formula that simulates the values of a random variable. Pritsker states that the preferred method of generating pseudo random numbers "is to employ a recursive equation which generates the (i + 1)th random number from previous random numbers. Since the sequence of numbers is produced deterministically by an equation, they are not truly random" (Pritsker:717).

Pritsker then lists six properties that are desireable in a pseudo random number generator, which are as follows:

1. The numbers should be uniformly distributed in the interval (0,1).
2. The numbers should be independent and, hence, no

correlation should exist in a sequence of random numbers.

3. Many numbers should be generated before the same number is obtained. This is referred to as the period or cycle length of the generator.
4. A random number sequence should be reproducible. This implies that different starting values or seeds should be permitted to allow different sequences or stream to be generated.
5. The generator should be fast, as many numbers may be required in a simulation.
6. A low storage requirement is preferred.

General Sampling Techniques

There are many different techniques to generate random samples. "Pritsker surveyed over 300 sources, whi' he compiled into the following fundamental approaches for random sample generation" (Pritsker:707):

1. Inverse Transformation
2. Composition
3. Acceptance/Rejection

(All three of these methods are commonly found in the literature.) "The uniform samples derived from these three approaches provide the basic source of randomness in simulation experiments" (Pritsker:707).

Inverse Transformation The inverse transformation method is the simplest and most fundamental technique for generating random samples (Pritsker:708). This method is performed by setting a random number, r, equal to the cumulative density function (cdf), F(x), and then solving for x. An example of this technique using the exponential

cdf, is as follows (where 1/lambda is the mean of the exponential distribution):

$$F(x) = 1 - e^{-(lambda)x}$$

Setting F(x) equal to r and then solving for x yields,

$$x = -(1/lambda)*ln(1 - r)$$

> Hence, if r is uniformly distributed in the range 0 to 1, then x given by the above equation is exponentially distributed with a mean value of 1/lambda. (Pritsker:708)

The problem with this technique is that the cdf in question may not be invertible (Pritsker:709). Therefore this method is not applicable for some commonly used distributions (like the beta distribution).

Composition "The composition method assumes the density function can be written as a weighted sum of component distribution functions with the sum of the weights totaling one" (Pritsker:710). A sample from one of the component distributions must first be taken. This procedure is repeated until a sample from each of the component distributions has been gotten. These samples are then summed in accordance with their respective weighting factors to create one random variable.

Acceptance/Rejection "The acceptance/rejection approach to generating random samples involves generating samples from a distribution and then rejecting some of the samples

in a way that the remaining samples have the desired distribution" (Pritsker:709). The process begins with the generation of two random numbers. "One random number is used to calculate a proposed random sample, x" (Pritsker:709). Let f(x) be the density function from which random deviates are required -- this is where the sample value is inserted (Pritsker:709; Tadikamalla:925). "The key is to find a different function, t(x), whose value returned for a given zero-one number is always larger than the function f(x) at the same point" (Sumner:15). The majorizing function, t(x), must be easy to sample from. The second random number is drawn from a uniform distribution along the (0,1) interval (known as a uniform variate).

If the uniform variate is less than the ratio of the majorizing function value over the pdf value (ie, if U < t(x)/f(x)), then x is accepted as a random variate of the function, f(x) (Tadikamalla:926). Otherwise, x is rejected and the process must be repeated by selecting two new random numbers. As the majorizing function gets closer to the desired function, less values of x are rejected. Therefore the goal of this method is to find a t(x) function as close to the pdf as possible to minimize the amount of work required to generate the desired random numbers.

Summary

Life cycle costing is usually conducted in one or more of

three different models. Estimating by analogy is generally considered the easiest of the three because it requires the least amount of detail. Parametric modeling involves more work than an analogy estimate due to the addition of performance and/or physical characteristics into the CER's. Engineering or bottom up estimates require a great deal of detail and are generally not recommended by the DoD. The key to good LCC estimates is to start them as early in the program's conceptual phase as possible. This allows management to receive a quick and easy overview of what is to be expected while also allowing changes to be made in a simplistic fashion.

The generation of random numbers for work on a digital computer is conducted in one of three ways: by random number tables, by random number generators, or by pseudo random number generation. The preferred method is pseudo random number generation because the results can be replicated and this is the fastest of the three methodologies.

The three most common sampling techniques to produce random samples are the inverse transformation method, the composition method, and the acceptance/rejection method. These methods do not have to be used independently. In most cases, they are combined to ensure the desired sample is achieved.

III. Methodology

Overview

The purpose of this chapter is to acquaint the reader with the general methods used to meet the objective and some of the subobjectives stated in the first chapter (the subobjectives not covered in this chapter will be addressed in Chapter IV). Additionally, the reader will also be acquainted with the two distributions made available in the program LCC.FOR (the Beta and Triangular distributions), and the methodology used to validate and verify the program.

Cost Estimating Relationships

As stated in the literature review, there are three types of LCC models -- of which the parametric relationship or CER was selected to be used by the program LCC.FOR. The CER was chosen because it allows the user to predict the future based upon information received from past occurrences (MECA:79). (An illustration of a CER was shown in Figure 1.)

For a CER to be effective, the variables used to estimate the cost of a future weapon system must have characteristics similar to those of some existing or previously existing system. For example, the future cost of the booster stages to be used in the small intercontinental ballistic missile

(ICBM) may be based upon the costs associated with the boosters used on the Minuteman III or Peacekeeper ICBM weapon system. Because of the close similarities in booster technology, the characteristics demonstrated by existing rocket stages can be used to forecast the cost of future boosters.

A counter example where using an older system may not be practical is the reentry vehicles used on the Minuteman III ICBM versus those used on the Peacekeeper weapon system. The reentry vehicles used on the Minuteman III are the MK-12 and the MK-12A, where the MK-21 is the reentry vehicle used on the Peacekeeper missile. The technological differences between the MK-12/MK-12A and the MK-21 make cost comparisons between the two inadvisable. The MK-12/MK-12A is comprised of vacuum tube electronics -- the MK-21 is entirely solid state. The MK-12/MK-12A conducts a single radar update to determine its location above the earth when reentering the earth's atmosphere -- the MK-21 makes four. Because of these and other significant differences, the MK-12/MK-12A would not represent an acceptable forecasting model to determine the cost of the MK-21.

A CER consists of two types of terms, explanatory variables and beta parameter estimates. The explanatory variables are the Xi terms where the parameter estimates are represented by bi or (beta)i.

In the program LCC.FOR, the explanatory variables are referred to as cost drivers. A person using the program LCC.FOR can assign each Xi to either a constant or to a range of values based upon some form of either the beta or triangular distribution. This is done because sometimes the value of an explanatory variable is known for certain, while at other times the exact value of an explanatory variable is only known to exist within a range of numbers. For example, suppose a CER consists of two cost drivers, X1 and X2. The value of X1 is regarded as a constant equal to 100 units, but the value of X2 can take on any value between 100 and 250 units. The program LCC.FOR needs to know how to assign a value to X2. This is where the beta and triangular distributions come into play.

Beta and Triangular Distributions

Continuing with the example above, an analyst is not likely to know what value to assign the cost driver X2. The expert who builds component X2 is not likely to know the distribution X2 would be best represented by. Together, the analyst can ask the expert a series of questions to determine an educated guess as to the actual value of X2.

The questions needed to be asked by the analyst are:

1. What is the minimum value X2 can attain?
2. What is the maximum value X2 can attain?
3. How is X2 likely to be distributed?

An expert would be expected to know the approximate minimum and maximum values in answering the first two question. But the response to question number three will probably be a blank stare. However, the clever analyst can rephrase question number three to get the needed information -- aided by either the beta or triangular distributions.

Beta Distribution The beta distribution was incorporated into the computer program LCC.FOR because it allows a user to generate a rough model with little or no data. Additionally, "since the beta distribution is defined over a finite interval, it is extremely useful in describing situations which have a finite range" (Pritsker:705). The finite range is explained by the minimum and maximum values provided by the expert. Figure 2 displays nine 'types' of beta distributions (taken from Sumner's thesis). The Beta distribution was reduced to nine pdf's because they represent the input options in the program LCC.FOR. By showing these nine pdf's to the expert, an analyst would be able to get an accurate assessment of how X2 is actually represented.

As shown in Figure 2, the nine beta pdf's give a general account of how a component could be best described. Beta distributions of 'type' 1, 4, and 7 are all skewed left. This means a sample from one of these pdf's is more likely to be closer to the high estimate than to the low estimate.

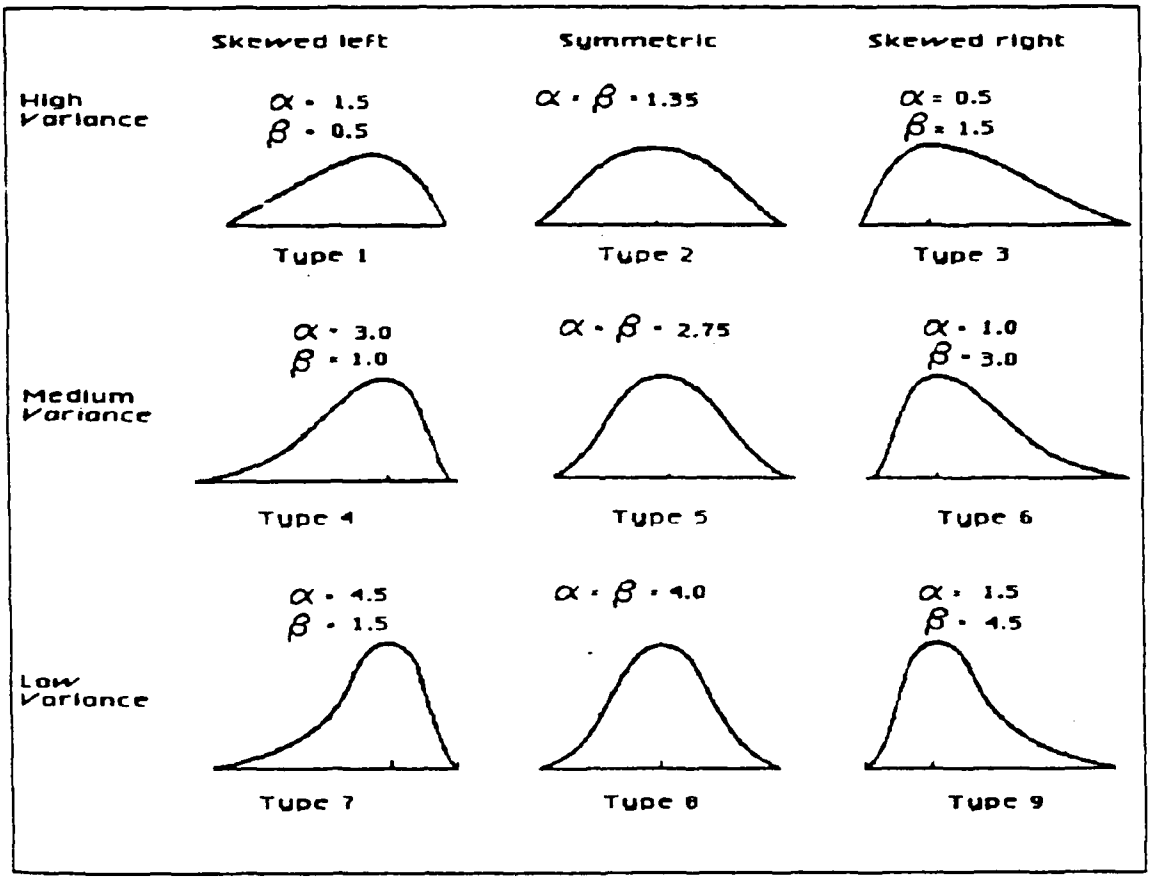


Figure 2 Nine Beta Distributions

The difference between these three 'types' is the amount of variability contained in each of them. For the analyst trying to explain these nine beta distributions to the expert, the analyst could ask the expert if he would expect the actual value of the cost driver to be closer to the low or high estimate he provided earlier. If the expert feels strongly that the actual cost driver would be more likely to be nearer the high estimate than the low, a beta

distribution of 'type' 7 would be assigned to it. If the diagram of the nine beta pdf's is not available, or if the expert is not comprehending the information being explained to him -- the triangular distribution becomes a viable alternative.

Triangular Distribution The triangular distribution operates much like the beta except instead of requiring a 'type', the triangular pdf needs a mode or best guess estimate. The triangular distribution was also incorporated into the computer program because it has the same advantages as those mentioned for the beta distribution -- it also allows a user to generate a rough model in the absence of data. One additional advantage of the triangular distribution is it is more intuitive to those without a background in statistics. No diagrams of nine distributions is required in trying to get information from a non-statistician regarding how a cost driver is likely to be distributed. The only information needed from the expert in this case is:

1. What is the minimum value X2 can attain?
2. What is the maximum value X2 can attain?
3. What is the mode or best guess of the value X2 is likely to be?

This eliminates any possible confusion that may arise in trying to explain the concept of 'variance' to a person not

familiar with statistics. However, the estimation of the variance is the main drawback to the use of the triangular distribution. With only the minimum, maximum, and mode values provided, estimation of the variance produces meaningless results.

Software Verification and Validation

"The software verification and validation processes are concerned with evaluating the performance of a model" (Pritsker:12). In this case, the model is the program LCC.FOR. "Verification means the computer program functions as intended, where validation checks to see if the program is a reasonable representation of the system it was designed to emulate" (Pritsker:12-13). The verification and validation stages were conducted through the use of a test run (provided by Dr Cain and Capt Gibson) consisting of eight cost elements. The algorithms contained in the program's code were compared against the same calculations made by hand. This was done to verify the program LCC.FOR was performing as expected -- both at the nuts and bolts level and the system as a whole. The validation process involved taking the expected value of each of the eight cost elements as they were distributed over time (refer to Table 2 in Chapter IV). The values of the eight cost elements were then placed in the program LCC.FOR to validate that the program yields an accurate assessment of calculating life

cycle cost estimates. It is important to understand that the validation that took place validated a model -- validation of the real world may or may not have been shown. The point here is each cost element is an approximation of the value of some entity -- the entities themselves may not have been an accurate assessment of the real world. However, the program did validate the models presented.

IV. Implementation

Overview

This chapter describes how the subobjectives mentioned in Chapter I were implemented. Specifically, the subobjectives mentioned in this chapter are: 1. how the data input procedures in the program LCC.FOR were modified, 2. detailed results of the verification and validation procedures, 3. instructions regarding how the program LCC.FOR could be modified to meet the individual needs of a user, and 4. the outcomes of the tests conducted to determine if only 250 random samples per cost element and cost driver needed to be obtained versus the current process of getting 750 random variables.

Data Input Modification

The program originally designed to estimate life cycle costs consisted of an impersonal, unforgiving, and tedious method of entering data. The 'old' process was made up of executable code (LCC.EXE) and one external file containing the data (INPUTLCC.DAT). The data file required the user to place the values to be entered in precise positions within the file. No interaction between the main program and user ever took place. If data was improperly aligned, a run time error would occur. This run time error was difficult to

debug because of the large amount of information contained in the data file. Another hinderance in debugging was that there were no messages or labels explaining what the data within the file represented. If data was incorrectly entered, or if a change to one or more values was desired, the user had to physically enter the file INPUTLCC.DAT to make the necessary changes.

The solution to making the 'old' program more personal, more forgiving, and "user friendly" was accomplished as follows: 1. provide the user with prompts from the main program asking for each data point, 2. incorporate checks on the data to ensure the inputs were within an acceptable range, 3. provide the user with an echo check, meaning allow the user to see what she had just entered (so she could be certain the data set contained the intended values), and 4. ask the user if any changes to the data set would be necessary (and the means to correct them), while at the same time keeping the storage locations of the data invisible to the user.

Step one involved asking the user questions regarding each data point to be entered. The user's responses would then be sent to one of two files external to the main program depending on the form of the model being entered (the form of the model refers to either a constant, beta or triangular random variable, or a CER cost element). All

information was stored in the file INPUTLCC.DAT except for some data relating to CER's. Information about the estimated variance of the CER, the covariability of the cost drivers incorporated within the CER, the slope parameters, and the description of how each cost driver was distributed was stored in the file MOREINPT.DAT. This separate file was necessary to simplify data transfers and to eliminate the extra computer code necessary to account for the entering and editing of the data.

The three types of models handled by this program are:

1. constants,
2. random variables from either a beta or triangular distribution,
3. random variables generated by a CER.

The level of complexity increases from constant data to CER information. This means the amount of information required for a CER is greater than for a beta or triangular random variable, which in turn requires more information than a constant cost element. All of the information needed to evaluate the first two models is contained in the file INPUTLCC.DAT, as is most of the CER information. The problem with placing all the CER data into the INPUTLCC.DAT file is: 1) no restriction is placed on the user regarding the order of the models to be input (meaning a constant cost element may be entered before a CER and visa-versa), and 2)

additional bookkeeping procedures would have to be incorporated into the program to maintain the proper placement of data being entered/edited.

The easiest way to approach this was to use the information already provided by the main program. Methodologies already existed within the main program to keep track of data storage. The only change necessary was to modify where the data was being sent. This meant changing file numbers, opposed to developing numerical algorithms and pointers to keep track of data locations.

The second part of the solution involved checking the user's inputs to ensure they were within an acceptable range. For example, many queries presented by the program are in the form of YES/NO questions requiring either a '1' or '2' as a response. If the user entered a value less than '1' or greater than '2', the computer would catch this mistake and ask the user to reenter their response, thus preventing the program from encountering a run time error.

The echo checks in the program occur in one of two places. One echo check takes place immediately after the CER data to be sent to the file MOREINPT.DAT has been entered. The second check occurs after all of the data has been input. The location of the first echo check coincides with why the second data file exists -- for ease and expedition. Also, the availability for corrections to be

made to these inputs only occurs here because this type of data is generally not updated once correctly entered. The second echo check is made available to the user when the program is rerun (refer to the User's Manual located in Appendix A for more information).

The fourth solution ties in closely with the third. Immediately following each echo check, the program asks the user if any changes need to be made. If no corrections are required, the program either continues asking for the remainder of the data set to be entered or begins calculating the LCC of the system. If changes are necessary, the program will ask the user how many changes need to be made. A positive integer response is then followed with queries (equal to the number of changes required), for the index number along the left hand column associated with the value to be updated. For example, assume the user has just entered the estimated variance, variance-covariance matrix, slope parameters, and cost driver cards of a CER consisting of three slope parameters (refer to the User's Manual located in Appendix A for details on how and when these variables are entered). The echo check provided upon completion of entering these values is shown in Table 2. The program then asks:

```
Do you want to edit any of these?
Enter:
1 for YES
2 for NO
```

If a '2' is entered, the program either continues the data acceptance process or begins calculating the LCC of the system. If a '1' is submitted, the program responds with,

> How many values would you like to edit?
> (Enter 0 to exit)

If the user did not want to make any corrections but mistakenly entered a '1' instead of a '2', she can still exit this procedure by entering a '0'. If however, one correction needs to be administered, the user would enter a '1'. The program then asks the user:

> Enter the # of the value to be
> changed (from the left hand column)

Notice the number on the far left of each value entered (see Table 2). This number is what the program is asking the user to enter to represent the value to be changed. Continuing with this example, say the element varcov (2,1) needs to be changed to 0.55. The user would respond by entering the number in the left hand column associated with varcov (2,1) -- in this case a '5'. The program would then state:

> Enter the NEW VARCOV(2,1):

This is where the user would enter the new value for the variance-covariance element (2,1) (by typing .55 or 0.55

then pressing RETURN).

Software Verification and Validation

Verification The verification process was conducted by evaluating whether the following aspects of the program LCC.FOR performed as intended: 1. the error check subroutine ANS, 2. the editing procedures located at both echo checks, and 3. the performance of the subroutines designed to calculate the triangular distribution.

The subroutine ANS was tested while executing the program LCC.FOR. Queries such as, "Enter: 1 for YES, 2 for NO" were responded with numeric values outside of the expected range of '1' to '2'. In every instance the program returned with the statement, "Please enter a number from 1 to 2". Also, the range specified in this second query was appropriate to the data required to be entered.

The editing procedure following the two echo checks was initiated per the instructions provided in the User's Manual (located in Appendix A). Verification that updates incorporated into the database were properly executed were found in two locations. First, if the update made was significant, a noticeable change in the values displayed in the output analysis (refer to Step 24 of the User's Manual) became evident. The second and usually more apparent location occurred when the program was rerun. Selecting the option to review the database verified the new value had

TABLE 2

ECHO CHECK EXAMPLE OF CER DATA CONSISTING OF
3 SLOPE PARAMETERS (NO INTERCEPT TERM)

```
1. variance (s-squared) =  0.99

2. varcov (1,1) =  0.1
3. varcov (1,2) =  0.2
4. varcov (1,3) =  0.3
5. varcov (2,1) =  0.4
6. varcov (2,2) =  0.5
7. varcov (2,3) =  0.6
8. varcov (3,1) =  0.7
9. varcov (3,2) =  0.8
10. varcov (3,3) =  0.9

Press RETURN to continue . . .

11. slope param 1 = 1.1
12. slope param 2 = 2.2
13. slope param 3 = 3.3

Cost Driver for Element # 1
14. LOW value =  1
15. HIGH value = 1
16. Beta TYPE =  1

Cost Driver for Element # 2
17. LOW value =  25
18. HIGH value = 45
19. Beta TYPE =  7

Cost Driver for Element # 3
20. LOW value =  50
21. HIGH value = 50
22. Beta TYPE =  1
```

been accepted by the presence of the new value in the listing (like the one shown in Table 2).

The subroutines TRIAG and TRIAG1 were designed to generate random variables from the triangular distribution.

Each subroutine receives the low, high, and mode values necessary to calculate the triangular distribution from the calling portion of the program. First, the subroutine generates random numbers from the (0,1) uniform distribution. These random numbers are then transformed into values representative of the triangular distribution. Before the transformation can occur however, the "break" point must be determined. The "break" point represents the mode value of the triangular distribution. This value must be obtained because different equations are used depending on whether the random number generated is less or greater than the "break" point.

Proper execution of this subroutine was verified by first comparing the "break" point calculated by the program against a hand calculation of the same value. Once it was established these values matched, the next step was to print out the value of the random number generated and ensure the proper equation was used to produce the triangular distribution estimate. In other words, if the random number generated was greater than the "break" point, the equation designed to calculate this set of circumstances must be selected. Print statements were inserted into the program signifying if the greater than or less than path had been chosen. The subroutine was verified to be functioning properly because each time the random number was

greater/less than the "break" point, the appropriate message was issued.

Validation A test case involving the life cycle cost of a weapon system consisting of eight components was used in attempting to validate the accuracy of the overall program LCC.EXE. The eight components were comprised of two constants, three random variables from the beta distribution, and three random variables generated by CER's. The make-up of the eight components and their hand generated expected life cycle costs are shown in Table 3.

The eight cost elements presented in Table 3 were calculated against a discount rate (annual percentage rate or interest) of 10%. Additionally, each cost element has an individual payment schedule associated with it. The discount rate converts each payment schedule into a present value -- how much it would cost today to pay for the cost element if the payment was a lump sum.

The expected life cycle costs of the two constants were calculated by simply spreading the cost associated with each cost element over their respective payment schedules. The algorithm used in assigning a dollar value to a particular year within each payment schedule was taken from the computer code generated by Dr Cain. This algorithm is located in the subroutine TRAP contained in the program LCC.FOR (refer to Appendix C).

The expected life cycle costs of the cost elements generated by beta random variables were attained in the following manner: 1. refer to the subroutine BETA (located in LCC.FOR) and go to the section of the IF-THEN loop corresponding to the beta 'type' applicable to each cost element, 2. set the PIN and QIN values associated with the beta 'type' in question equal to alpha and beta, respectively, and 3. calculate the expected value using the equation (taken from class notes provided by Dr Cain);

$$\text{Expected Value} = \text{LOW} + (\text{HIGH} - \text{LOW})*(\text{alpha} + 1)/(\text{alpha} + \text{beta} + 2)$$

For example, the expected life cycle cost of cost element #4 (refer to Table 3) was generated as follows:

$$\text{EV (ce \#4)} = 900\text{K} + (2500\text{K} - 900\text{K})*(5.5 + 1)/(5.5 + 2.5 + 2)$$
$$= 900\text{K} + 1600\text{K} * 0.65$$

$$\text{EV (ce \#4)} = 1{,}940{,}000.00$$

Each expected value generated for cost elements 3, 4, and 5 was then subjected to the cost per year algorithm mentioned in the discussion regarding the constant cost elements (for details, refer to the subroutine TKAP contained in the program LCC.FOR).

The expected life cycle costs of the CER's was conducted using the same tools shown in the constant and beta random variable sections, but these tools were applied to different

aspects of the CER's. In each CER shown in Table 3, at least one explanatory variable was not known for certain. This meant the expected value of those explanatory variables had to first be determined. For example, the explanatory variable X2 located in the eighth cost element was distributed by a 'type' 4 beta distribution over a range of 2500 to 6000 units. The algorithm mentioned above was used to calculate the expected value of X2 (see below).

$$EV\ (X2) = 2500 + (6000 - 2500)*(4 + 1)/(4 + 2 + 2)$$

$$= 4687.5$$

Cost elements 6 and 7 represent learning curves (meaning the average cost of each unit decreases as the number of units purchased increases). After determining the expected value of each explanatory variable and taking the inverse natural logarithm of the two equations, this result equalled the expected average cost of purchasing one unit of each respective cost element (AC). Both cost element 6 and cost element 7 required a total buy of 200 units. Thus, the average cost of cost elements 6 and 7 had to be multiplied by 200 to obtain their respective expected life cycle cost.

The same data shown in Table 3 was then put into the computer program LCC.FOR. A total of thirty runs of this data set were generated. Thirty was selected to ensure a reasonable estimate could be established, and because thirty is generally accepted as a significant sample size.

TABLE 3

HAND GENERATED EXPECTED LIFE CYCLE COST OF A WEAPON SYSTEM CONSISTING OF EIGHT COMPONENTS

Cost Element	Type	Dollar Value	Payment Plan	Year Realized	Expected LCC (c)
1	1 (a)	150000	2 (b)	2 (b)	105013.32
2	1	800000	5	1	518393.50
3	2	B:100000 to 300000:2 (d)	9	1	112679.88
4	2	B:900000 to 2500000:7	22	5	439264.37
5	2	B:2500000 to 5000000:4	24	6	885432.91
6	3	(e)	7	4	470850.93
7	3	(f)	17	5	1211617180.00
8	3	(g)	19	0	4646076.99

Notes:

(a) The coding for Type is:
1 = constant
2 = random variable (beta or triangular pdf)
3 = random variable generated by a CER

(b) Component's life cycle (in years).

(c) The expected cost of the respective component.

(d) For components of Type 2, this symbology represents: B or T (for either the Beta or Triangular pdf): the LOW value to the HIGH value: the type of Beta distribution or the MODE of the Triangular distribution.

(e) This cost element is generated by the CER (requiring a total buy of 200 units):

$$\ln(AC) = 9.239961 - .321775*\ln(X1) + .189285*\ln(X2)$$

where, AC = average cost per unit,
Expected value of X1 = 95, X2 = 50

(f) This cost element is generated by the CER (requiring a total buy of 200 units):

$$\ln(AC) = 16.81124283 - .32*\ln(X1) + .15*\ln(X2) + .212*\ln(X3)$$

where, AC = average cost per unit,
Expected value of X1 = 95, X2 = 101, X3 = 68.33

(g) This cost element is generated by the CER:

$$LC = 10000837 + 201.0509491*X1 + 149.3534241*X2$$

where, LCC = life cycle cost
Expected value of X1 = 3000, X2 = 4687.5

Each run was independent of the others because the seeds used during each run were changed (the seeds chosen in each run were based on the computer's clock).

The results of the hand and computer generated expected life cycle costs of a weapon system consisting of eight components are:

HAND CALCULATION = $1,218,794,892.00
COMPUTER CALCULATION = 968,354,333.33

Each computer run generated 750 estimates of the given weapon system's life cycle cost. The output provided by the computer runs was listed in percentile increments. Since the hand calculations were based upon the expected value of the data, the 50th percentile was taken from the thirty computer generated calculations.

The variance of the hand calculation was not obtained, however the variance of the thirty computer runs was. Assuming the data from the thirty runs was normally distributed, a confidence interval was placed around the computer calculation. Using an alpha value of 0.1, the t-value equals 1.699 (taken from a standard t-table where alpha = 0.05 with 29 degrees of freedom). The 90% confidence interval is presented below.

968354333.33 +/- t(1 - alpha/2)*(stdev/(square root of N)
968354333.33 +/- 1.699*(5432038.228/(square root of 30)

968,173,265.4 < Computer calculation < 986,535,401.2

The confidence interval placed around the computer calculation does not contain the hand generated value. This infers the hand generated calculation is not equal to the computer generated calculation -- hence the program LCC.FOR may not be doing what it was intended to do. An error in selecting the 50% may be the reason for the discrepancy -- the hand calculation was best approximated by the 95% in the output analysis of each computer run.

Another possibility for the difference is the manner in which the computer calculates the CER estimates. The program LCC.FOR uses the normal distribution to evaluate the expression:

$$y = bo + b1*X1 + b2*X2 +/- N(0,1)*(s\text{-}sqrd + X'(varcov)X)^{0.5}$$

where N(0,1) represents the normal distribution with a mean of zero and variance of one. The actual value that should have been used is the t distribution. The normal distribution was used because it is much simpler to generate -- one line of code completes the task. The use of the normal opposed to the t is a valid technique if the degrees of freedom are 30 or more. If the degrees of freedom are small (as in the example shown in Table 3), use of the normal distribution can grossly underestimate predictions.

This may have been the case with the computer calculation provided earlier. Notice it is significantly less than the

hand generated calculation -- possibly due to the use of the normal distribution instead of the t distribution.

Random Variable Sampling Reduction

The number of samples generated by the program LCC.FOR is currently 750. A test was conducted to determine if this number could be reduced without losing information. If no information was lost then the benefit of less samples being taken is in the compilation time saved by the computer. The actual amount of calculations saved is not simply, 750 - 250 = 500. In most cases, the number of samples taken exceeds 5000. Therefore, if the 750 could be reduced to 250, the amount of work not having to be accomplished by the computer would be substantial.

The data used in this experiment was the one mentioned earlier containing eight cost elements (refer back to Table 3). Since two of the cost elements represent learning curves, Dr Cain suggested running this experiment at a variety of purchase values. This was accommodated by running tests where the number purchased by the learning curves were set at 95, 125, 150, 175, and 200 units. Each purchase value was run 30 times at 750 samples, 30 times at 500 samples, and 30 times at 250 samples, for a total number of runs equalling 450. Each run was independent from all the others because the seeds used were changed. The output generated after each run was presented in percentile form --

meaning the maximum value was listed, then the value representing 99 percent of the LCC estimates, then 95%, etc, down to the minimum value attained for that particular run. Dr Cain suggested testing the 95th percentile -- which was done along with the minimum and maximum values.

The test was broken into three sections; testing to determine if the data was normally distributed, comparing the mean value of the thirty runs, and comparing the variance of the thirty runs.

Normality Test The Wilk-Shapiro test was conducted on each set of thirty data points to determine if they were normally distributed. The assumptions made for these tests were: 1. the data samples were acquired randomly and 2. the data sets are independent from each other. The hypothesis for each test was:

Ho: data set 1/data set 2 is a normal distribution function with unspecified mean and variance
Ha: data set 1/data set 2 is nonnormal

The decision rule used for these tests was: Fail to accept Ho if the Wilk-Shapiro statistic was less than the 0.05 quantile.

Results: Every test for normality passed the decision rule except for the following:

NUMBER SAMPLES	NUMBER PURCHASED	MIN, MAX OR 95%	WS CALCULATED	WS TABULAR
500	200	MAX	0.9034	0.927
250	150	95%	0.9133	0.927
750	125	95%	0.9023	0.927

Although these three tests did not meet the WS statistic, their result was close enough to be waived.

Comparison of Means The means of each run conducted at 250 and 500 samples was compared against the corresponding run done at 750 samples. The hypothesis test used for these tests was:

Ho: mean (data set 750) = mean (data set 250) or mean (data set 500)
Ha: mean (data set 750) <> mean (data set 250) or mean (data set 500)

The decision rule used was: Fail to accept Ho if the absolute value of the calculated t-statistic was greater than the t-statistic obtained from a table. The tabular t value was obtained at alpha = 0.01 with 58 degrees of freedom.

Results: Every test passed the decision rule. In fact, no test failed at an alpha = 0.1 and only one of the 30 tests failed at an alpha level of 0.2.

Comparison of Variances The hypothesis used in evaluating these tests was:

Ho: var (data set 750) = var (data set 250) or var (data set 500)

Ha: var (data set 750) <> var (data set 250) or var (data set 500)

The decision rule used to determine which hypothesis should be selected was: Using the F-distribution, if the ratio of the sample variances falls between the range of F(alpha/2,n1-1,n2-1) and F(1-alpha/2,n1-1,n2-1), for alpha = 0.01 and n1 = n2 = 30, fail to reject Ho.

Results: all of the tests passed the decision rule except:

1. 250 vs 750 samples with 175 components purchased at the 95% level --the lower bound was exceeded by 0.0274.
2. 250 vs 750 samples with 150 components purchased at the 95% level -- the lower bound was exceeded by 0.2102.
3. 250 vs 750 samples with 95 components purchased at the 95% level -- the lower bound was exceeded by 0.1100.

Based upon the results of these tests, it was concluded a sample size of 250 random samples yields similar results as a sample size of 750 (ie, little or no information is lost by decreasing the number of random samples from 750 to 250).

Program Modification

The program LCC.FOR is currently configured with the following format:

1. 750 LCC samples are acquired,
2. The maximum number of cost elements that can be entered is 20.

If these parameters are not sufficient, changes can be made by making the following changes in the LCC.FOR computer code.

To modify the number of LCC samples taken, the following changes must be made:

LOCATION	LINE #	VARIABLE NEEDING ADJUSTMENT
Main Program	7	A(*,20), LC(*)
Main Program	8	XCER(*,20)
Main Program	12	NRV/*/
Subroutine TRIAG	19	NR = *
Subroutine TRIAG1	5	NR = *
Subroutine BETA	6	NR = *
Subroutine CER	7	NRR = *
Subroutine BETA1	5	NR = *

The line # refers to the line number of the respective routine given under the location heading. Replace the value located where the * is with the number of runs desired. CAUTION: If the number of runs is to be reduced, reduction has only been verified down to 250. Any value inserted less than 250 may not yield statistically acceptable results.

If the number of cost elements needs modification, the following changes must be made:

LOCATION	LINE #	VARIABLE NEEDING ADJUSTMENT
Main Program	7	A(750,*), CE(*,20)
Subroutine TRAP	7	CE(*,20)
Subroutine TRAP	11	MCON(*,100), MRAND(*,100)

The line # refers to the line number of the respective routine given under the location heading. Replace the value located where the * is with the number of cost elements desired.

V. CONCLUSIONS AND RECOMMENDATIONS

GOAL REALIZATION

The purpose of this effort was to enhance Dr Cain's computer simulation for the forecasting of life cycle costs. This was attained. The program LCC.FOR is easy to use and provides quick, accurate estimates of weapon system costs. The addition of the triangular distribution allows those without a solid understanding of distributions, variance in data, and statistics in general the opportunity to comprehend their inputs.

RECOMMENDATIONS FOR FUTURE RESEARCH

Random Variable Sampling Reduction The minimum number of random samples generated by the program LCC.FOR while maintaining statistical integrity is 250. This number can be reduced -- but how far? A reduction is still possible due to the strength shown in the tests done in comparing the means and variances.

A test that could be conducted concerning this topic would be to determine the optimal number of random samples to be acquired for a given set of cost elements. The first item to be checked would be to find the "actual" value of a predetermined quantile. For example, run the same data as entered in Table 3, but let the number of random samples be increased to some large value, say 10,000. This will allow the output to "zero in" on the actual values for the given

data set.

Once the "actual" values are known, a confidence interval could be placed around them. The test would be to reduce the number of random samples taken until the established confidence interval is violated. When the confidence interval is violated, it would be known with certainty that at least that many random samples would have to be taken to ensure statistical integrity is maintained. The procedure would be repeated with various amounts of cost elements to establish the minimum number of random samples to be acquired for each.

Validation Check The analysis regarding the hand and computer generated expected life cycle costs revealed a discrepancy. Was this caused by user error (where the user is this author), using the normal distribution instead of the t distribution, or is there a flaw somewhere in the software? The answer to this is important so that future projects using this program will be as accurate as possible.

APPENDIX 1: USER'S MANUAL

Step 1: Log into the CSC.

Step 2: Copy LCC.EXE and ANAL.SAS from Dr Cain's userid.

Step 3: Create the files INPUTLCC.DAT, MOREINPT.DAT, and LCCOST.DAT. This is done by typing (at the $ prompt):

$ CREATE LCCOST.DAT <RETURN>

The system will not respond with any prompt. Pressing <Control 'Z'> at this time creates the file LCCOST.DAT;1 in the user's directory. Repeat this process for the other two files (INPUTLCC.DAT and MOREINPT.DAT).

Step 4: Type (at the $ prompt) 'RUN LCC.EXE <RETURN>'. This executes the file LCC which is designed to calculate the life cycle cost of a weapons system.

Step 5: At this point, a series of questions will be asked by the program LCC requesting the user to provide numerical responses.

WARNING: ALL RESPONSES TO QUERIES FROM THE PROGRAM LCC MUST BE IN NUMERICAL FORM -- EVEN TO ANSWER 'YES/NO' QUESTIONS.

For example, once a data set has been entered and the user wants to make a change to the data on the next run, the program will ask the user:

Do you wish to edit the data?
Enter:
1 for YES
2 for NO

If a 'Y' or 'N' is entered (or any combination of letters), the program will encounter a run time error. The program will stop functioning at this point because only a numeric response is expected. However, if a letter response is entered and the program to aborts, go back to Step 4.

The user should not be concerned with entering a number outside of the specified range (a 1 or 2 in this example). If the range of possible values is exceeded, the computer will repeat its question until an appropriate value is entered. Continuing with this example, if a '3' were entered, the program would respond with:

Please enter a number from 1 to 2

This process would continue until a '1' or '2' was entered by the user.

Step 6: The first question asked by the program is:

Do you wish to access:
1. the last data set entered, or
2. begin a new data set?
Enter a 1 or 2

If the user has never executed this program before, no 'last data set' exists, therefore option '1' is not feasible (yet). In this case the user would enter a '2' and press <RETURN> and proceed to Step 7.

For information regarding what to do if the last data set needs to be edited, go to Step 20.

Step 7: This step and those immediately following mean the user is going to enter a new data set. The first question asked in entering the new data set is:

Enter the number of cost elements:

The user should respond with an integer value from 1 to 20 (the program can only handle a maximum of 20 cost elements as presently configured -- refer to page xx of Capt Gibson's thesis "An Interactive Life Cycle Cost Forecasting Tool", if configuration changes need to be conducted). This value represents the number of cost elements that make up the weapon system.

Step 8: The program then asks the user to enter the life expectancy of the weapon system (in years). This is an integer value representing 1 to 100 years.

Step 9: The discount rate the weapon system is to be evaluated against is the third query in this section. If the discount rate is 9.5%, enter either 0.095 or .095 (the zero preceding the decimal point may be omitted).

Step 10: Step 10 is an iterative process. The number of times it is repeated is equal to the number of cost elements to be entered. The program wants to know what 'type' each cost element is. A cost element can be one of three 'types':

1. a constant or "fixed value" which is assumed to be certain,
2. a random variable modeled by either a Beta or Triangular probability density function (pdf), or
3. a random variable generated by a cost estimating relationship (CER).

If the cost element is a constant, enter a '1' (go to Step 11 for further directions).

If the cost element is a random variable to be modeled by either a Beta or Triangular pdf, enter a '2' (go to Step 12).

If the cost element is a random variable generated by a CER, enter a '3' and proceed to Step 13.

Step 11: Enter the constant cost of the element. For example, if the cost is $25,000, enter 25000. (Go to Step 14.)

Step 12: The first question asked is if this cost element is to be modeled by a Beta or Triangular pdf. If a Beta pdf is desired, enter a '1' and proceed to Step 12a. If the Triangular pdf is desired, enter a '2' and proceed to Step 12b. (Detailed information regarding the Beta and Triangular distributions along with why they were chosen for this program is located on pages 21 to 25 of Capt Gibson's thesis.)

Step 12a: The next three queries regard the LOW value of the Beta distribution, the HIGH value, and the TYPE of Beta distribution to be modeled. For example, if the LOW, HIGH, and TYPE are:

LOW = $10,000
HIGH = $30,000
TYPE = 4 (medium variance with higher probability of values occurring closer to the HIGH end ($30,000) -- refer to Figure 2 in Capt Gibson's thesis for a description of the nine TYPE's of Beta distributions)

The responses to the computer's queries would be:

```
10000   (for LOW)
30000   (for HIGH)
4       (for TYPE -- this must be an integer
        from 1 to 9)
```

When these three values have been entered, proceed to Step

14.

Step 12b: The next three queries regard the LOW value of the Triangular distribution, the HIGH value, and the MODE or best guess of the Triangular distribution to be modeled. For example, if the LOW, HIGH, and MODE are:

LOW = $10,000
HIGH = $30,000
MODE (best guess) = $25,000

The responses to the computer's three queries would be:

10000 (for LOW)
30000 (for HIGH)
25000 (for MODE or best guess)

When these three values have been entered, proceed to Step 14.

Step 13: Because this cost element is a random variable to be generated by a CER, the number of cost drivers (slope parameters) needs to be entered. For example, if the CER to be evaluated is represented by the expression:

$$Y = a0 + a1*X1 + a2*X2 + a3*X3,$$

the number of cost drivers is 3, hence a 3 would be entered (notice the intercept term is not included). Go to Step 13a.

Step 13a: This is where the information regarding the existence of an intercept term asked for. If an intercept term exists in the CER, enter a '1', otherwise enter a '0' (go to Step 13b).

Step 13b: At times the CER in question will not be in a form easily manipulated unless the natural logarithm of both sides is taken. To prevent the user from having to compute these values and have to keep track of which terms need to be transformed, the program performs this task for the user. If a logarithmic transformation is required, the user would respond to the program's query with a '1' -- if no transformation is necessary, enter a '0' (go to Step 13c).

Step 13c: The purchase of a weapon system sometimes involves the buying of more than one particular component making up the weapon system. For example, a Boeing B-52 aircraft requires eight engines. To account for this in the software, the program LCC asks the user if the final cost

estimate of the CER needs to be multiplied by a scalar. If a scalar multiple is required, the user would respond to the program's query by entering a '1' and then going on to Step 13d. If no scalar multiple is required, a '0' should be entered (go to Step 14).

Step 13d: The next three queries represent how many components of the CER need to be purchased. If the value of the scalar multiple is certain, set both the LOW and HIGH values equal to each other. The program will then ask the user how the LOW and HIGH values are to be distributed based upon one of nine forms of the Beta distribution. If the LOW and HIGH values are equal, this information is not used, therefore the user should enter a '1' to the query regarding the TYPE of Beta distribution to be modeled. However, the exact amount of items to be purchased is not always known. In many instances, especially when a weapon system is still in the developmental stage, only a range of the number of components required is known -- hopefully based upon some distribution. This is why the program asks for a LOW value, HIGH value, and TYPE of Beta distribution. For example, if the LOW, HIGH, and TYPE are:

LOW = $10,000
HIGH = $30,000
TYPE = 4 (medium variance with higher probability of values occurring closer to the HIGH end ($30,000))

The responses to the computer's queries would be:

```
10000   (for LOW)
30000   (for HIGH)
4       (for TYPE -- this must be an integer
        from 1 to 9)
```

When these three values have been entered, proceed to Step 14.

Step 14: This section of the program relates to the time phasing of each cost element. Time phasing refers to how the cost of each cost element is to be distributed over time. When the Department of Defense purchases a weapon system, the total cost of the system is not usually paid for in full as soon as the contracts detailing how many will be bought is signed by all involved. In most cases, the cost of the weapon system is spread out over the life of the weapon. For example, if the cost element is a constant with a cost of $50,000, a portion of this cost may occur in increments at first, say over three years (the component's

PHASE-IN period). The next five years of payments could be set at a constant amount (the component's CONSTANT cost per year), while the remaining payments decrease at a steady rate for two years (the component's PHASE-OUT period). The program transforms this information into a trapezoid. If only a PHASE-IN or PHASE-OUT period exists, the shape of the trapezoid becomes a triangle. If only a CONSTANT period exists, the shape of the trapezoid is a rectangle. The point is the program takes all of this into account so the user does not have to. The final piece of information required in this section is -- when do the payments for this particular component begin? The program transforms the initial cost provided by the user into annual costs based upon the discount rate, phase-in, phase-out, and constant cost per year values. Using the data mentioned above, the inputs to the program would be:

```
                    PHASE-IN:  3
                   CONSTANT:  5
                  PHASE-OUT:  2
YEAR WHEN PAYMENTS BEGIN:  8
```

The user must make sure the total of these four values does not exceed the life cycle of the system specified earlier in Step 8 (if the user makes an error here, the computer will catch it and ask the user to check her numbers and to try again).

NOTE: If the first realization of the cost element occurs immediately, enter a '0'. In other words, if the payments for this particular cost element are to start immediately, enter a '0'.

CAUTION: The four time phasing values cannot all be zero. If all four time phasing values are zero, the computer will ask the user to try again.

If this cost element is a random variable estimated by a CER (a '3' was entered at Step 10), proceed to Step 15. If this cost element was either a constant or a random variable modeled by the Beta or Triangular distribution, and more cost elements need to be entered, go back to Step 10. If neither of these two criteria apply, then the user has completed all inputs. This means the program will ask no further questions and the estimated life cycle cost of the weapon system is being executed (refer to Step 24 for output analysis). If changes to the data need to be made, type 'RUN LCC.EXE' when the '$' prompt appears and follow the instructions provided.

Step 15: Every CER has a certain variability associated with it, along with an inherent variance/covariance between the cost drivers included within the CER. Step 15 asks the user to enter the value of the estimated variance (s-squared) of the CER estimate.

CAUTION: If the variance is very small (less than 1e-5) or very large (greater than 1e5), be sure to enter the variance in the form shown below. For example, if the variance is;

0.00000567, at the prompt enter, 5.67e-6, or
2340000000, at the prompt enter, 2.34e9

This will prevent the program from encountering a run time error (go to Step 16).

Step 16: This step is an iterative process dependent on the number of slope parameters plus the intercept term (if applicable). This is where the variance-covariance matrix of the cost drivers is entered. The variance of the intercept term occupies the position (1,1) if it is present. The data for the matrix is requested in row format. This means element (1,1) is asked for first, followed by element (1,2), then element (1,3), until all covariances in the first row have been entered. This process is repeated until all n-squared elements have been entered (where n equals the number of cost drivers plus the intercept (if applicable)). The caution mentioned in Step 15 regarding the input of very small or very large values applies to these entries as well. Proceed to Step 17.

Step 17: This is where the values for the slope parameters are entered. If an intercept term is present, the program will be aware of this and ask for the user to enter it first, followed by the remaining slope parameters.

As mentioned in Step 15, if the value of the intercept or slope parameters is very small or very large, the user must enter the value in accordance with the example shown in Step 15. Proceed to Step 18.

Step 18: Each cost driver is characterized by three queries. Therefore, this step is an iterative process where three inputs are required for each cost driver. Before each cost driver's data can be entered however, the user is asked:

Do you want cost driver # 1 to be modeled
by a Beta or Triangular pdf?
Enter:
1 for YES
2 for NO

If the Beta distribution is selected, go to Step 18a. Go to Step 18b for input instructions regarding the Triangular distribution.

Step 18a. The three queries presented in this step refer to the LOW value of the Beta pdf, the HIGH value of the Beta pdf, and the TYPE of Beta pdf to be modeled. If this cost driver is associated with an intercept term, the program will print the message,

Enter the INTERCEPT values NOW

prior to asking for the LOW value of the Beta pdf. This is important because if as intercept is present, the value to be entered for all three queries is '1.0'. For example, assume an intercept term does exist in the model described below:

$$AC = bo * X1^{b1} * X2^{b2}$$

where,

AC = average cost per unit
X1 & X2 = cost drivers
bo = intercept term
b1 & b2 = slope parameters

This equation can be rewritten as,

$$\ln(AC) = \ln(bo) * X0 + b1 * \ln(X1) + b2 * \ln(X2)$$

This should help clarify why the values of the intercept term in response to the cost driver queries must all be equal to '1.0' -- the X0 term can only take on a value of '1.0'.

CAUTION: The cost drivers representing the X0 term must always equal '1.0', and they can never be altered (without completely reestimating the CER). If an intercept term is present, the program LCC.FOR regards X0 as cost driver #1 even though it is not a cost driver. X1 is treated as cost driver #2 (internally by the program) and X2 is cost driver #3.

Therefore, the three responses for an intercept term would be:

LOW = 1.0
HIGH = 1.0
TYPE = 1.0

If the cost driver in question does not refer to an intercept term, insert the actual LOW, HIGH, and TYPE values. If the value of the cost driver is certain, enter the same number for both the LOW and HIGH queries then enter a '1' for the query regarding the TYPE of Beta distribution to be modeled (the value for the TYPE is unimportant in this instance because the LOW and HIGH terms are equal). If the value of the cost driver is not certain, enter the appropriate LOW and HIGH values along with the TYPE of Beta distribution to be modeled. Continuing the example above, if X1 is assumed to be a constant equal to 100, then the responses to the LOW and HIGH queries would both be 100 (because the value of X1 is certain.) The TYPE of Beta distribution is irrelevant since the LOW and HIGH values are the same, but an integer from 1 to 9 must be entered. Suppose X2 can take on a value between 100 and 250 characterized by a Beta distribution of TYPE 8 (symmetrically distributed with low variability). The responses for this cost driver would be:

LOW = 100
HIGH = 250
TYPE = 8

NOTE: If the natural logarithm of the LOW and HIGH values is required (ie, if a '1' was entered in Step 13b), be sure both the LOW and HIGH values entered by the user are greater than zero. Go to Step 19.

Step 18b: The three queries presented in this step refer to the LOW value of the Triangular pdf, the HIGH value of the Triangular pdf, and the MODE or best guess of the Triangular pdf to be modeled. If this cost driver represents the intercept of the CER in question, the program will print the message,

Enter the INTERCEPT values NOW

prior to asking for the LOW value of the Triangular distribution. This is important because if as intercept is present, the value to be entered for all three queries is '1.0'. For example, assume an intercept term does exist in the model described below:

$$AC = bo * X1^{b1} * X2^{b2}$$

where,

AC = average cost per unit
X1 & X2 = cost drivers
bo = intercept term
b1 & b2 = slope parameters

This equation can be rewritten as,

$$\ln(AC) = \ln(bo) * X0 + b1 * \ln(X1) + b2 * \ln(X2)$$

This should help clarify why the values of the intercept term in response to the cost driver queries must all be equal to '1.0' -- the X0 term can only take on a value of '1.0'.

Therefore, the three responses for an intercept term would be:

LOW = 1.0
HIGH = 1.0
MODE = 1.0

If the cost driver in question does not refer to an intercept term, insert the actual LOW, HIGH, and MODE values. If the value of the cost driver is certain, enter the same number for the LOW, HIGH, and MODE queries. If the value of the cost driver is not certain, enter the appropriate LOW, HIGH, and MODE values. Continuing the example above, if X1 is assumed to be a constant equal to 100, then the responses to the LOW, HIGH, and MODE queries would all be 100 (because the value of X1 is certain.) Suppose X2 can take on a value between 100 and 250 with a best guess of X2 being 150. The responses for this cost driver would be:

LOW = 100
HIGH = 250
MODE (best guess) = 150

NOTE: If the natural logarithm of the LOW, HIGH, and MODE values is required (ie, if a '1' was entered in Step 13b), be sure the LOW, HIGH, and MODE values entered are greater than zero. Go to Step 19.

Step 19: Due to a limitation of the program, editing of the estimated variance (s-squared), variance/covariance matrix, and slope parameters is only possible upon the completion of

each cost element being entered. When the last cost driver card has been submitted for a particular cost element, the program prints the message;

> The estimated variance (s-squared), covariance matrix, slope parameters, and cost driver cards have just been entered -- Do you wish to review or edit any of these?
>
> Enter:
> 1 for YES
> 2 for NO

If no review is desired, the user enters a '2' and the program advances to the next cost element. If this was the last cost driver of the last cost element, no further requests for data will be made by the program -- proceed to Step 24 for a brief explanation of the output analysis.

If the user wishes to review or edit the data from Step 15 on and enters a '1', the program issues a notice then lists the estimated variance (s-squared), and the elements comprising the covariance matrix. If a change is necessary on one or more of these values, write down the number in the left hand column next to the value needing to be modified. The user is instructed to write down the number in the left hand column corresponding to the incorrect parameter because of the large number of parameters that may exist - this will eliminate the user from forgetting which element needed to be changed. For example, if the CER entered contained four cost drivers and an intercept term, a total of 46 elements eligible to be edited would appear on the screen (1 variance estimate, 25 covariance elements, 5 slope parameters, and 15 cost driver cards).

Once the last element of the covariance matrix has been presented, the program issues the message,

> Press RETURN to continue . . .

After reviewing these values and documenting any changes needed to be made, the user would press <RETURN>. This invokes the program to display the slope parameters and cost driver values currently entered in the database. Immediately following the last cost driver value, the program asks the user:

Do you want to edit any of these?

Enter:
1 for YES
2 for NO

If no changes are required the user would enter a '2', meaning the program will advance to the next cost element (if one exists). (Go to either A listed above or to Step XX for output analysis.) If changes ARE required, continue on to Step 19a below.

Step 19a: The program will then ask the user how many changes need to be made. If the user realizes that NO changes are necessary, she would enter a '0'. This will cause the program to act as if a '2' were entered in Step 19. If changes are required, the user would enter an integer value equal to the number of changes desired. For example, if an element in the covariance matrix and one of the cost driver values needed to be changed, the user would enter a '2' in response to the question;

How many values would you like to edit?

The user would then proceed to Step 19b.

Step 19b: The program then asks the user to enter the number of the value to be changed, and follows this query with the statement, "Enter the new value". Continuing with the example shown in Step 19a, the user would enter the following values:

Enter the # of the value to be changed: x
Enter the NEW value: the NEW phase-in value goes here

Enter the # of the value to be changed: y
Enter the NEW value: the NEW phase-out value goes here

The values x and y represent the numbers in the left hand column corresponding to the OLD covariance element and cost driver values, respectively. The order in which these two changes are made is NOT important. The key for the user is she must keep track of the numbers in the left hand column that correspond with the desired updates to be made.

Step 20: This step means the user has selected to either review and possibly edit the last data set entered or just conduct an additional run of the last data set entered. In either case, the first question posed to the user is:

Do you wish to edit the data?

Enter:
1 for YES
2 for NO

If the user selects to not edit the data, the program will proceed to execute the estimated life cycle cost of the weapon system using the same data provided by the last set (refer to Step 24 for output analysis).

If a review or editing of the data is desired, the user would enter a '1' to the question stated above (go to Step 21).

NOTE: The following steps only allow the user to review and edit the previous data set. Cost elements may NOT be added or deleted. If the addition or deletion of a cost element is desired, the user must reenter an entire new data set. This is accomplished by entering a '2' at Step 6.

Step 21: This step asks the user if a change to the expected life of the system needs to be modified. Before asking if a change is necessary, the program first provides a statement stating what the expected life of the system was for the previous data set. If a change is needs to be made, the program will ask the user to enter the new life cycle of the system (go to Step 22).

CAUTION: A change to the life cycle of the system, especially if the number of years is reduced may affect other portions of the data entered -- specifically, the total time of the phase-in, constant, and phase-out should be checked to ensure the new life cycle is not exceeded.

Step 22: This step is similar to Step 21 except in this step the discount rate for the system is presented for possible change. If a modification is desired, be sure to enter the new discount rate as described in Step 9 (go to Step 23).

Step 23: This step is an iterative process dependent upon the number of cost elements entered in the previous data set. The program asks the user if she wants to review each cost element. The cost elements are shown in the same order they were input in the last data set. If the user does not wish to review/edit a particular cost element, a '2' should be entered. This will cause the program to advance to the next cost element. This process is repeated until all of the cost elements have been offered for review/editing.

Once all of the cost elements have been presented for review/editing, the program will begin executing the estimated life cycle cost of the weapon system. If changes have been made, these changes will be incorporated in the program's life cycle cost estimate (go to Step 24 for output analysis).

If review/editing of a cost element is desired, the user would enter a '1' for that particular cost element. The process the program undergoes for this selection is as follows:

A. The 'type' of the cost element is stated (constant, random variable estimated by a Beta or Triangular pdf, or random variable estimated by a CER).
B. Data pertinent to the cost element is listed.
C. The program asks if any changes to this particular cost element is desired. If NO changes are required the user would enter a '2', meaning the program will advance to the next cost element (if one exists). (Go to either A listed above or to Step 24 for output analysis.) If changes ARE required, continue on to step D below.
D. The program will then ask the user how many changes need to be made. If the user realizes that NO changes are necessary, she would enter a '0'. This will cause the program to act as if a '2' were entered in step C. If changes are required, the user would enter an integer value equal to the number of changes desired. For example, if the phase-in and phase-out values needed to be changed for a constant cost element, the user would enter a '2' in response to the question;

How many values would you like to edit?

E. The program then asks the user to enter the number of the value to be changed, and follows this query with the statement, "Enter the new value". Continuing with the example shown in part D, the user would enter the following responses:

Enter the # of the value to be changed: 2
Enter the NEW value: the NEW phase-in value goes here

Enter the # of the value to be changed: 4
Enter the NEW value: the NEW phase-out value goes here

The values 2 and 4 represent the numbers in the left hand column corresponding to the OLD phase-in and phase-out values, respectively. The order in which these two changes are made is NOT important. The key for the user is she must

keep track of the numbers in the left hand column that correspond with the desired updates to be made.

This example refers only to a constant cost element. The entries for the other two possible 'types' of cost elements are similar, but a little more complicated because they contain more options.

NOTE: The 'type' of cost element **cannot** be changed. If the 'type' of cost element requires editing, the entire data set must be reentered.

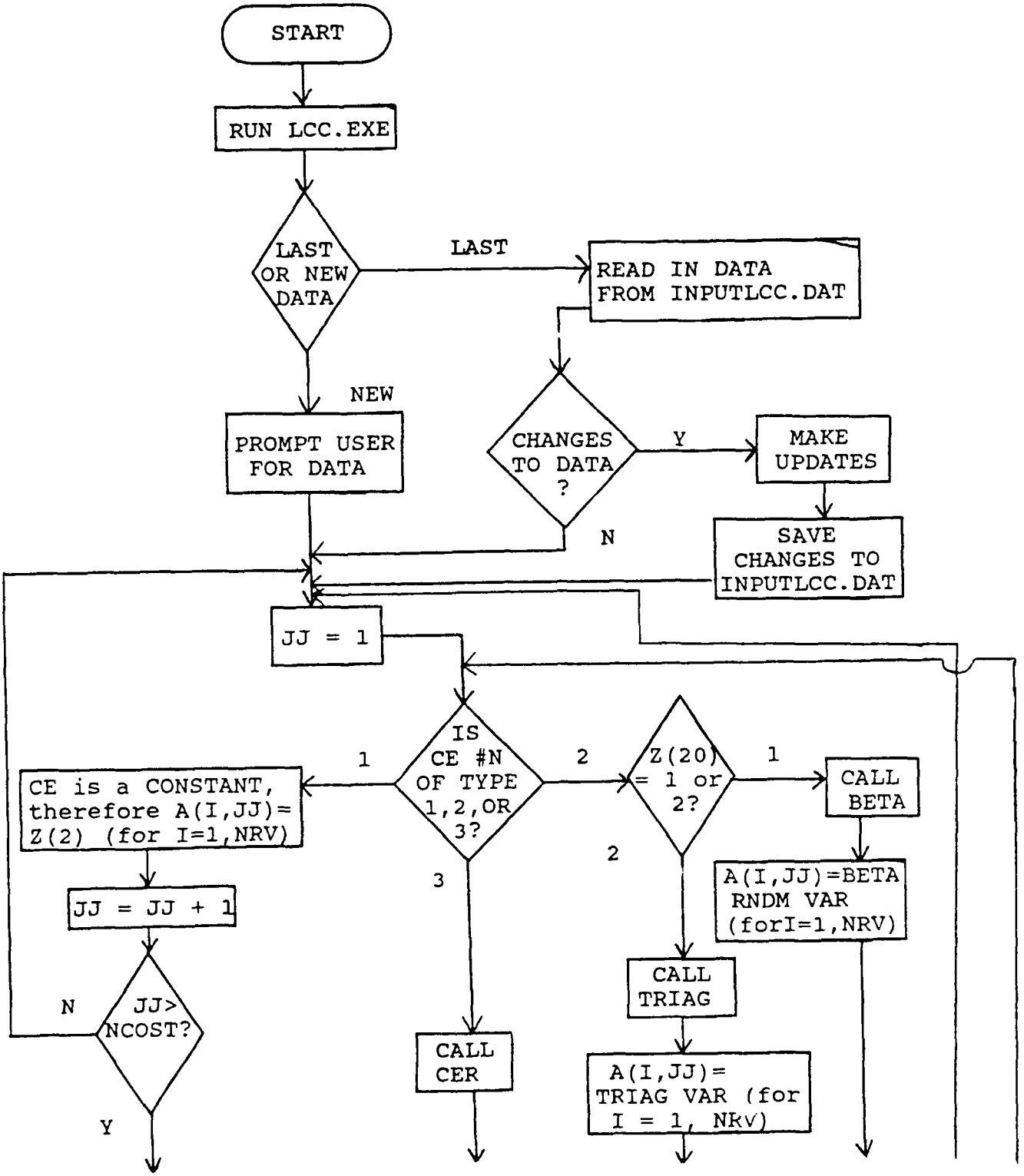
Step 24: This section and those that follow provide a brief explanation in how to obtain an analysis of the data entered into the program LCC.FOR. When the program LCC.EXE has finished running and the '$' prompt returns to the screen, type:

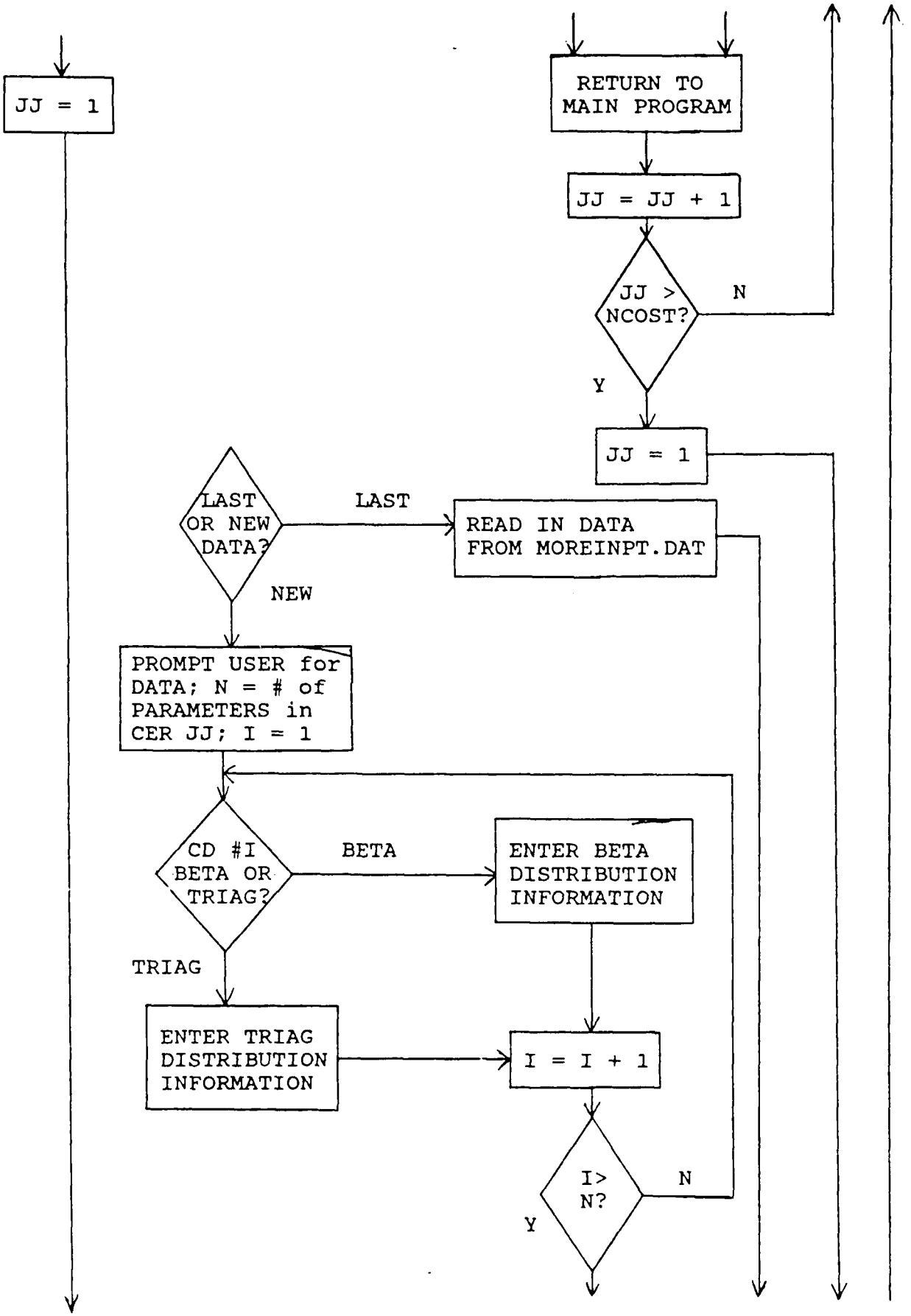
```
$ SAS ANAL
```

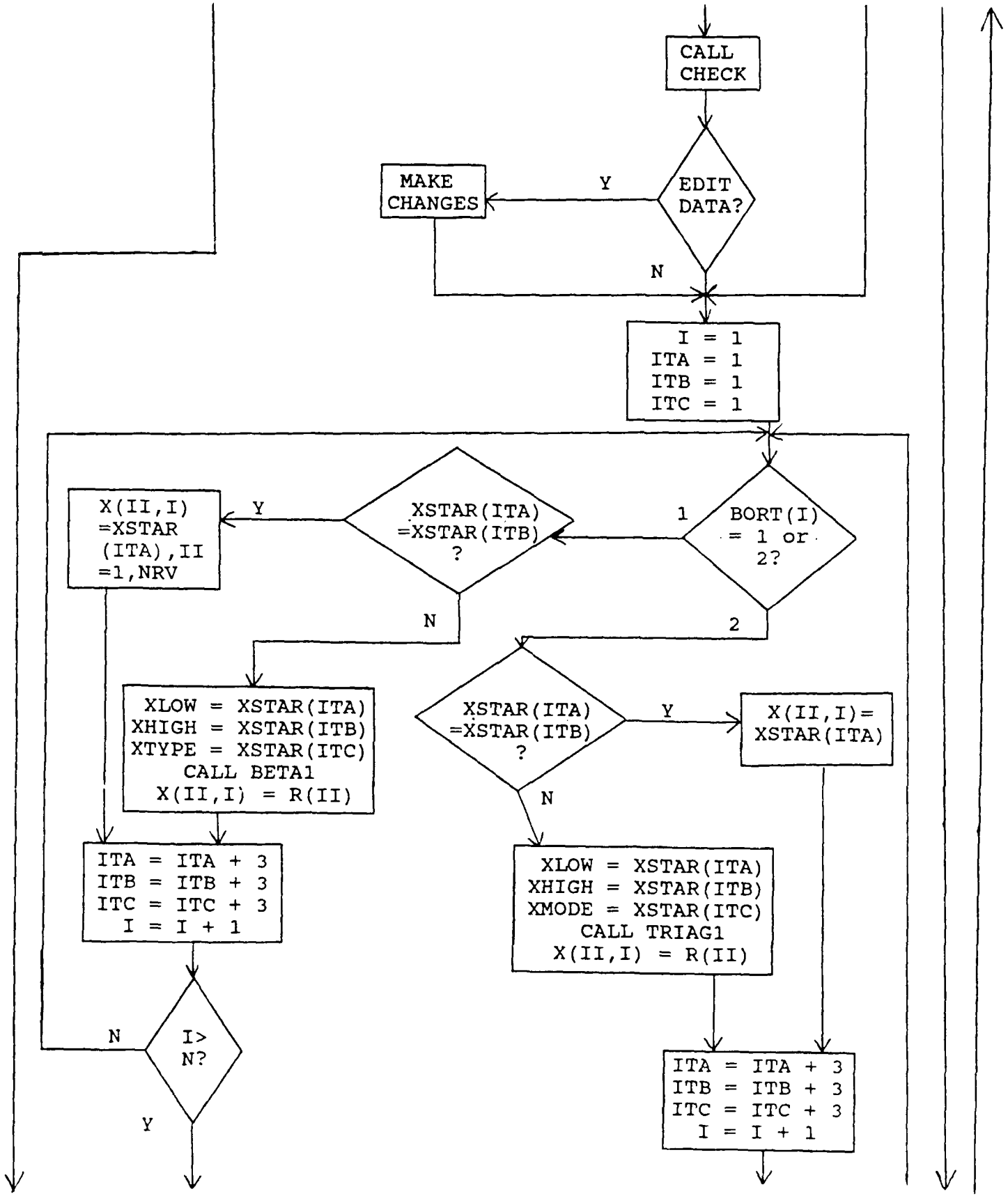
This causes the SAS program ANAL.SAS to be executed. This program will automatically access the file 'LCCOST.DAT' which contains the results calculated by the program LCC.EXE. When the '$' prompt returns, the results from this SAS run will be contained in the file ANAL.LIS.

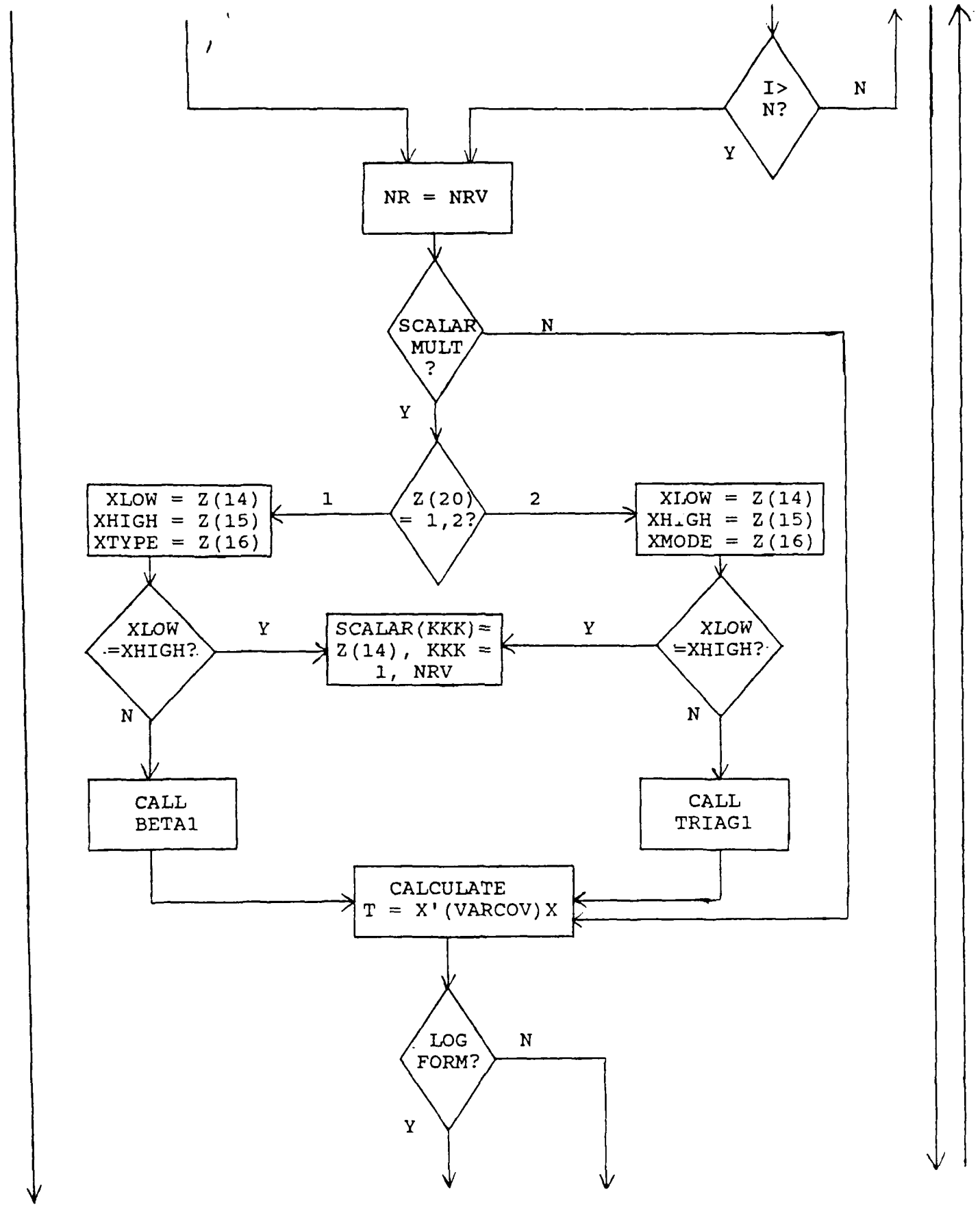
ANAL.LIS contains all of the statistical values necessary to complete an analysis of the estimated life cycle cost of the data provided by the user.

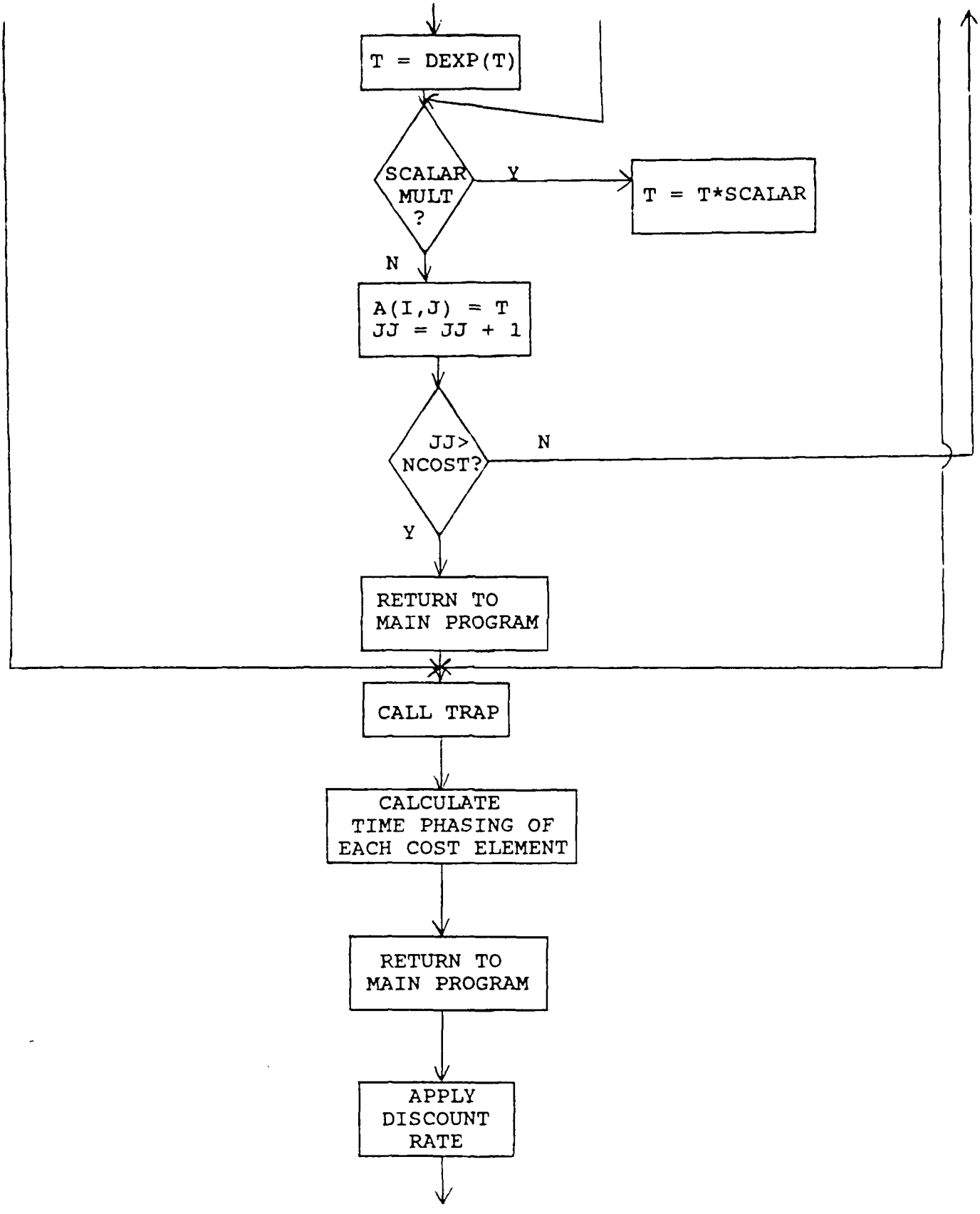
APPENDIX B: FLOWCHART

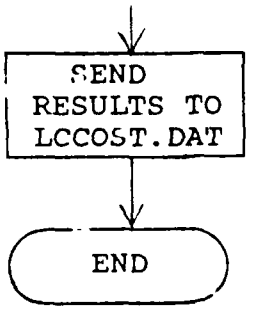



JJ = 1
RETURN TO MAIN PROGRAM
JJ = JJ + 1
JJ > NCOST?
N
Y
JJ = 1
LAST OR NEW DATA?
LAST
READ IN DATA FROM MOREINPT.DAT
NEW
PROMPT USER for DATA; N = # of PARAMETERS in CER JJ; I = 1
CD #I BETA OR TRIAG?
BETA
ENTER BETA DISTRIBUTION INFORMATION
TRIAG
ENTER TRIAG DISTRIBUTION INFORMATION
I = I + 1
I> N?
N
Y


CALL CHECK
EDIT DATA?
Y
MAKE CHANGES
N
I = 1
ITA = 1
ITB = 1
ITC = 1
BORT(I) = 1 or 2?
1
2
XSTAR(ITA) =XSTAR(ITB) ?
Y
X(II,I) =XSTAR (ITA),II =1,NRV
N
XLOW = XSTAR(ITA)
XHIGH = XSTAR(ITB)
XTYPE = XSTAR(ITC)
CALL BETA1
X(II,I) = R(II)
XSTAR(ITA) =XSTAR(ITB) ?
Y
X(II,I)= XSTAR(ITA)
N
XLOW = XSTAR(ITA)
XHIGH = XSTAR(ITB)
XMODE = XSTAR(ITC)
CALL TRIAG1
X(II,I) = R(II)
ITA = ITA + 3
ITB = ITB + 3
ITC = ITC + 3
I = I + 1
I> N?
N
Y
ITA = ITA + 3
ITB = ITB + 3
ITC = ITC + 3
I = I + 1


I> N?
N
Y
NR = NRV
SCALAR MULT ?
N
Y
Z(20) = 1,2?
1
2
XLOW = Z(14)
XHIGH = Z(15)
XTYPE = Z(16)
XLOW = Z(14)
XHIGH = Z(15)
XMODE = Z(16)
XLOW =XHIGH?
Y
SCALAR(KKK)= Z(14), KKK = 1, NRV
Y
XLOW =XHIGH?
N
N
CALL BETA1
CALL TRIAG1
CALCULATE T = X'(VARCOV)X
LOG FORM?
N
Y


T = DEXP(T)
SCALAR MULT ?
Y
T = T*SCALAR
N
A(I,J) = T
JJ = JJ + 1
JJ> NCOST?
N
Y
RETURN TO MAIN PROGRAM
CALL TRAP
CALCULATE TIME PHASING OF EACH COST ELEMENT
RETURN TO MAIN PROGRAM
APPLY DISCOUNT RATE


SEND
RESULTS TO
LCCOST.DAT
END

APPENDIX C: FORTRAN SOURCE CODE

Overview

The subroutines ANS, CLRSCR, TRIAG, TRIAG1, and CHECK were written by Captain Gibson. All other subroutines, including the main program were designed by Dr Cain, though portions of Dr Cain's code was modified by Captain Gibson (specifically, sections of the main program and the subroutine CER were modified by Captain Gibson).

```
0001
0002     C      PROGRAM LCC.FOR WHICH CALCULATES THE LIFE CYCLE
                COST OF A
0003     C      WEAPONS SYSTEM. NOTE: THE IMSL SUBROUTINES MUST BE
                ATTACHED
0004     C      TO THE OBJECT DECK TO GET AN EXECUTABLE FILE.
0005            PROGRAM LCC
0006            LOGICAL found, lastset, ceredit, got
0007            DIMENSION A(750,20), COST(100), CE(20,20),
                STORE(100), LC(750)
0008           1,BET(20), XSTAR(60), XCER(750,20), bort(20),
                super(20,505)
0009            REAL*8 LC
0010            COMMON /AA/ Z(20)
0011            COMMON /AB/ COSTC(100),COSTR(100)
0012            DATA NRV/750/
0013
0014     C      TO CHANGE TO A DIFFERENT NUMBER OF LCC SAMPLES
                CHANGE THE
0015     C      FOLLOWING STATEMENTS IN THE PROGRAM:
0016     C      IN THE DATA STATEMENT BELOW CHANGE NRV NUMBER, IN
                SUBPROGRAMS
0017     C      BETA, CER, AND BETA1 CHANGE THE PARAMETER CARD AT
                THE BEGINNING
0018     C      OF EACH PROGRAM. ALSO CHANGE THE VALUES OF MATRIX A
                AND
0019     C      MATRIX XCER AND THE VALUES OF VECTOR LC IN THE
                ABOVE DIMENSION
0020     C      STATEMENT.
0021     C      A(NRV,20) HOLDS THE NRV SAMPLES OF EACH OF 20 COST
                ELEMENTS.
0022     C      TO HANDLE MORE THAN 20 COST ELEMENTS A(NRV,20) MUST
                BE
0023     C      EXPANDED AS MUST THE ROW DESIGNATOR OF CE(*,20).
0024     C      Z(20) IS THE ARRAY CONTAINING THE COST ELEMENT
                CARD.
0025     C      NOT ALL OF THE VALUES ARE CURRENTLY USED.
```

```
0026    C     CE(*,20): THE * ROWS REFERS TO THE NUMBER OF COST
              ELEMENTS
0027    C     PRESENT, THE 20 COLUMNS TO Z(20). IF YOU WANT 30
              COST
0028    C     ELEMENTS THEN YOU MUST MAKE THE FOLLOWING CHANGES:
0029    C     A(NRV,30),CE(30,20). IN SUBROUTINE TRAP CE(*,20)
              MUST BE CHANGED
0030    C     TO CONFORM EXACTLY TO CE(*,20) IN THE MAIN PROGRAM
              AND MCON(*,100)
0031    C     AND MRAND(*,100) MUST ALSO BE CHANGED (WHERE * IS
              DESIRED NUMBER
0032    C     OF COST ELEMENTS).
0033
0034    C     The files INPUTLCC.DAT and MOREINPT.DAT contain the
              data saved
0035    C     from the previous data set entered.  LCCOST.DAT
              contains the
0036    C     output of the 750 lcc samples attained by this
              program.
0037
0038          OPEN(01,FILE='INPUTLCC.DAT',STATUS='OLD')
0039          OPEN(02,FILE='MOREINPT.DAT',STATUS='OLD')
0040          OPEN(05,FILE='LCCOST.DAT',STATUS='OLD')
0041          REWIND 01
0042          REWIND 02
0043          REWIND 05
0044          CALL CLRSCR
0045          numcer = 0
0046          ceredit = .false.
0047          do 9 i = 1,20
0048          do 9 j = 1,505
0049             super(i,j) = 0.0
0050       9  continue
0051
0052    c     This section begins a long series of user prompts
              regarding
0053    c     either the last data set entered, or the new set to
              be entered.
0054
0055          print*
0056          print*,'Do you wish to access:'
0057          print*,'1. the last data set entered, or'
0058          print*,'2. begin a new data set?'
0059          print*,'Enter a 1 or 2'
0060          read*, oldnew
0061          asked = oldnew
0062          min = 1
0063          max = 2
0064          found = .TRUE.
0065          CALL ANS(min,max,asked,found)
0066          oldnew = asked
0067          lastset = .FALSE.
```

```
0068          /
0069      c     Lastset is initially set to False meaning the user
                will input a
0070      c     new data set.  If the user wants to review the last
                data set
0071      c     entered, lastset will be set to True (see line 86).
                The
0072      c     purpose of this is to ensure that if changes are
                made to the
0073      c     last data set, the changes will be saved at the
                appropriate time
0074      c     and no duplicate information is saved.
0075
0076            CALL CLRSCR
0077            IF (oldnew .EQ. 1) THEN
0078
0079      c     . . . retrieve the last data set entered from the
                   file
          c        INPUTLCC.DAT
0080      c        and put the data in the array CE(I,J) after
                   getting the
0081      c        number of cost components in the system, life
                   cycle of the
0082      c        system (in years), and the discount rate.
0083
0084               read(01,59)NCOST,NYEARS,DISCOUNT
0085       59      FORMAT(I3,1X,I3,1X,F4.3)
0086               DO 22 I=1,NCOST
0087               DO 23 J=1,20
0088                  READ(01,58)CE(I,J)
0089         58       FORMAT(F18.4)
0090         23    continue
0091         22    continue
0092               do 164 i = 1,ncost
0093                  if (ce(i,1) .eq. 3.0) then
0094                     numcer = numcer + 1
0095                     n = ifix(ce(i,10) + ce(i,11))
0096                     do 160 iii = 1, (n*n + 5*n + 1)
0097                        if (iii .le. (n*n + n + 1)) then
0098                           read(02,162) super(numcer,iii)
0099       162                 format(f31.16)
0100                     else if (iii .gt. (n*n+n+1) .and. iii .le.
                        (n*n+4*n+1)) then
0101                           read(02,161) super(numcer,iii)
0102       161                 format(f15.8)
0103                        else
0104                           read(02,163) super(numcer,iii)
0105       163                 format(f5.3)
0106                        endif
0107       160            continue
0108                  else
0109                  endif
```

```
0110      164    continue
0111             lastset = .TRUE.
0112             numcer = 0
0113             print*
0114
0115      c      The following user prompt may be the last one
                 the user sees.
0116      c      If the user does not wish to edit the data, no
                 further
0117      c      questions will appear and the $ prompt will show
                 up meaning
0118      c      the program has completed its run.  If data
                 viewing is
0119      c      desired, the user would enter a 1 to the
                 following prompt.
0120      c      A series of questions asking if the current data
                 needs to
0121      c      be modified would then appear.
0122
0123             print*,'Do you wish to edit the data?'
0124             print*,'Enter:'
0125             print*,'1 for YES'
0126             print*,'2 for NO'
0127             read*, yorn
0128             asked = yorn
0129             CALL ANS(min,max,asked,found)
0130             yorn = asked
0131             print*
0132             IF (yorn .EQ. 1) THEN
0133                print 721,NYEARS
0134      721       FORMAT(' The life cycle of this system
                    is',I4,' years.')
0135                print*,'Do you wish to change this?'
0136                print*,'Enter:'
0137                print*,'1 for YES'
0138                print*,'2 for NO'
0139                read*,yorn
0140                asked = yorn
0141                CALL ANS(min,max,asked,found)
0142                yorn = asked
0143                print*
0144                IF (yorn .EQ. 1) THEN
0145                   print*,'Enter the new life cycle of this
                       system:'
0146                   print*,'(the maximum is 100 years)'
0147                   read*,NYEARS
0148
0149      c            CAUTION:  A change to the life cycle of the
                       system,
0150      c            especially if the number of years is reduced
                       may
0151      c            affect other portions of the data entered --
```

```
0152    c           specifically, the total time of the phase in,
0153    c           constant, and phase out should be checked to
                    ensure
0154    c           the new life cycle is not exceeded.
0155
0156                ELSE
0157                ENDIF
0158                print 722,DISCOUNT
0159          722   FORMAT(' The discount rate for this system is
                    ',F4.3)
0160                print*,'Do you wish to change this?'
0161                print*,'Enter:'
0162                print*,'1 for YES'
0163                print*,'2 for NO'
0164                read*,yorn
0165                asked = yorn
0166                CALL ANS(min,max,asked,found)
0167                yorn = asked
0168                IF (yorn .EQ. 1) THEN
0169                   print*,'Enter the new discount rate for this
                       system:'
0170                   print*,'(for example, 9% would be entered as
                       0.09)'
0171                   read*,DISCOUNT
0172                ELSE
0173                ENDIF
0174                print*
0175                DO 30 ieye = 1, NCOST
0176
0177    c           In this section each of the cost elements become
                    available
0178    c           for review.  The sequence of the elements
                    matches how the
0179    c           data was originally entered.
0180
0181                   DO 31 j=1,20
0182                      Z(j)=CE(ieye,j)
0183             31    continue
0184                   if (Z(1) .eq. 3.0) numcer = numcer + 1
0185                   print 32, ieye
0186             32    FORMAT(' Do you wish to review cost element
                       #',I3)
0187                   print*,'Enter:'
0188                   print*,'1 for YES'
0189                   print*,'2 for NO'
0190                   read*, yorn
0191                   asked = yorn
0192                   min = 1
0193                   max = 2
0194                   CALL ANS(min,max,asked,found)
0195                   yorn = asked
0196                   print*
```

```
0197                IF (yorn .EQ. 1) THEN
0198                   print 33, ieye
0199        33         FORMAT(' COST ELEMENT NUMBER ',I3)
0200                   print*
0201                   IF (Z(1) .EQ. 1) THEN
0202
0203     c                the cost element in question is a
                          constant
0204
0205                      print 34, Z(2)
0206        34            FORMAT(' 1. is a CONSTANT =',F16.2)
0207                      print*,'Only the CONSTANT value can be
                          changed here -'
0208                      print*,'The TYPE of CE CANNOT be
                          changed unless a'
0209                      print*,'NEW data set is entered!'
0210                      print 35, Z(6)
0211        35            FORMAT(' 2. whose PHASE IN period is
                          ',F3.1,' years')
0212                      print 36, Z(7)
0213        36            FORMAT(' 3. This CE is constant for
                          ',F4.1,' years')
0214                      print 37, Z(8)
0215        37            FORMAT(' 4. The PHASE OUT period is
                          ',F3.1,' years')
0216                      print 38, Z(9)
0217        38            FORMAT(' 5. This CE time period begins
                          in year ',F4.1)
0218                      print*
0219                      print*,'Would you like to change any of
                          these?'
0220                      print*,'Enter:'
0221                      print*,'1 for YES'
0222                      print*,'2 for NO'
0223                      read*, yorn
0224                      asked = yorn
0225                      CALL ANS(min,max,asked,found)
0226                      yorn = asked
0227                      IF (yorn .EQ. 1) THEN
0228                        print*,'How many values would you
                            like to edit?'
0229                        print*,'(Enter 0 to exit)'
0230                        read*, numchanges
0231                        asked = numchanges
0232                        min = 0
0233                        max = 5
0234                        CALL ANS (min,max,asked,found)
0235                        numchanges = asked
0236                        IF (numchanges .GT. 0) THEN
0237
0238     c          If changes are to be made, the new values are
                   read into
```

```
0239    c          some form of the counter, jay.  The reason why
                   jay's value
0240    c          is inconsistent is to make sure the number of
                   the area
0241    c          needed to be changed will coincide with where
                   the data
0242    c          actually came from.  For example, if the Phase
                   Out period
0243    c          needs to be modified, the user would enter the
                   number 4.
0244    c          However the number 4 does not match up with
                   where the value
0245    c          for Phase Out is stored in the array Z.  That's
                   why in this
0246    c          case the user's input of 4 must have an
                   additional 4 added
0247    c          to it.
0248
0249                              DO 39 k= 1, numchanges
0250      666                       print*
0251                                print*,'Enter the # of the value
                                  to be'
0252                                print*,'changed (ie, a 1,2,3,4,
                                  or 5)'
0253                                print*,'(Enter a 0 to exit)'
0254                                read*, jay
0255                             IF (jay .LT. 0 .OR. jay .GT. 5) THEN
0256                                GO TO 666
0257                                ELSE IF (jay .EQ. 1) THEN
0258                                   jay = jay + 1
0259                                   print*
0260                                   print*,'Enter the NEW value:'
0261                                   read*, Z(jay)
0262                          ELSE if (jay .ge. 2 .and. jay .le. 5) then
0263                                   jay = jay + 4
0264                                   print*
0265                    print*,'Be sure the life cycle of this system,'
0266                         print 80,NYEARS
0267       80                FORMAT( F3.1,' is not exceeded.')
0268                                   print*,'Enter the NEW value:'
0269                                   read*, Z(jay)
0270                             else
0271                             ENDIF
0272
0273    c             put the new data in its proper place (below)
0274
0275                                CE(ieye,jay)=Z(jay)
0276       39                     continue
0277                           ELSE
0278                           ENDIF
0279                        ELSE
0280                        ENDIF
```

```
0281                    ELSE IF (Z(1) .EQ. 2) THEN
0282
0283     c      the cost element in question is a random variable
0284
0285       444             IF (Z(20) .EQ. 1.0) THEN
0286
0287     c      CAUTION:  If the Low or High value for either the
                beta or
0288     c      triangular distributions is changed, make sure the
                new
0289     c      Low/High value is indeed lower/higher than the
                corresponding
0290     c      High/Low value since no check is provided here in
                the code.
0291
0292                 print*,'1. is a RV modeled by a beta pdf --'
0293                 print*,'THIS CANNOT BE CHANGED HERE - YOU MUST'
0294                        print*,'ENTER A NEW DATA SET!'
0295                        print 40, Z(3)
0296          40          FORMAT(' 2. LOW value of the Beta distro
                          is ',F15.2)
0297                        print 41, Z(4)
0298          41          FORMAT(' 3. HIGH value of the Beta distro
                          is ',F15.2)
0299                        print 42, Z(5)
0300          42          FORMAT(' 4. The TYPE of Beta being
                          modeled is ',F3.1)
0301                      ELSE
0302                       print*,'1. is a RV modeled by a
                           triangular pdf --'
0303                 print*,'THIS CANNOT BE CHANGED HERE - YOU MUST'
0304                        print*,'ENTER A NEW DATA SET!'
0305                        print 70, Z(3)
0306          70          FORMAT(' 2. LOW value of the Triang
                          distro is',F15.2)
0307                        print 71, Z(4)
0308          71        FORMAT(' 3. HIGH value of the Triang distro
                        is',F15.2)
0309                        print*,'4. MODE or EXPECTED value of
                            the Triang'
0310                        print 72, Z(5)
0311          72            FORMAT('    distro is',F15.2)
0312                      ENDIF
0313                      print 43, Z(6)
0314          43          FORMAT(' 5. The PHASE IN period is
                          ',F4.1,' years')
0315                      print 44, Z(7)
0316          44          FORMAT(' 6. This CE is constant for
                          ',F4.1,' years')
0317                      print 45, Z(8)
0318          45          FORMAT(' 7. The PHASE OUT period is
                          ',F4.1,' years')
```

```
0319        /               print 46, Z(9)
0320       46               FORMAT(' 8. This CE time period begins
                            in year ',F4.1)
0321                        print*
0322              print*,'Would you like to change any of these?'
0323                        print*,'Enter:'
0324                        print*,'1 for YES'
0325                        print*,'2 for NO'
0326                        read*, yorn
0327                        asked = yorn
0328                        min = 1
0329                        max = 2
0330                        CALL ANS(min,max,asked,found)
0331                        yorn = asked
0332                        IF (yorn .EQ. 1) THEN
0333                          print*,'How many values would you
                              like to edit?'
0334                          print*,'(Enter a 0 to exit)'
0335                          read*, numchanges
0336                          asked = numchanges
0337                          min = 0
0338                          max = 8
0339                          CALL ANS(min,max,asked,found)
0340                          numchanges = asked
0341                          IF (numchanges .GT. 0) THEN
0342                            DO 47 k = 1, numchanges
0343                              print*
0344              print*,'Enter the # of the value to be changed'
0345                              print*,'(ie, enter a 1 - 8)'
0346                              read*, jay
0347                              IF (jay .EQ. 1) THEN
0348           print*,'THIS CANNOT BE CHANGED HERE!  YOU CAN ONLY'
0349         print*,'CHANGE THIS BY ENTERING A WHOLE NEW DATA SET.'
0350                                print*
0351                                GO TO 444
0352                              ELSE
0353                                jay = jay + 1
0354                                print*,'Enter the NEW value:'
0355                                read*, Z(jay)
0356                              ENDIF
0357                              CE(ieye,jay) = Z(jay)
0358       47                   continue
0359                          ELSE
0360                          ENDIF
0361                        ELSE
0362                        ENDIF
0363                    ELSE
0364
0365     c      The cost component in question is a cost
                estimating
0366     c      relationship.
0367
```

```
0368      555      print*,'1. This CE is a RV generated by a CER'
0369                 print*,'   THIS CANNOT BE CHANGED HERE
                     - a NEW data'
0370               print*,'   set must be entered to change
                   the CE type.'
0371                 print 48, Z(10)
0372       48        FORMAT(' 2. The # of SLOPE parameters
                     is ',F3.1)
0373                 print*,'3. If an INTERCEPT term exists,
                     a 1 will'
0374                 print*,'   appear, otherwise a 0 will
                     be used to'
0375                 print*,'   represent NO INTERCEPT term
                     -- NOTE:'
0376                 print*,'   If this is changed, you must
                     RE-ESTIMATE'
0377                 print*,'   the ENTIRE CER!'
0378                 print 49, Z(11)
0379       49        FORMAT('    INTERCEPT?: 'F3.1)
0380                 print*,'4. If CER was estimated by
                     taking logs, a'
0381                 print 50, Z(12)
0382       50        FORMAT('    1 will appear, a 0 if not -
                     LOGS?: ',F3.1)
0383                 print*,'5. If the final estimate is to
                     be multiplied'
0384                 print*,'   by a scalar, a one will
                     appear, otherwise'
0385                 print 51, Z(13)
0386       51        FORMAT('    a 0 will appear - SCALAR
                     MULT?: ',F3.1)
0387                 print 52, Z(14)
0388       52        FORMAT(' 6. The LOW value of the SCALAR
                     is',F7.1)
0389                 print 53, Z(15)
0390       53        FORMAT(' 7. The HIGH value of the
                     SCALAR is',F7.1)
0391                 print*
0392                 print*,'Press RETURN to continue . . .'
0393                 read*
0394                 if (Z(20) .eq. 1.0) then
0395                    print 54, Z(16)
0396       54           FORMAT(' 8. The TYPE of Beta distro
                        is ',F3.1)
0397                 else if (Z(20) .eq. 2.0) then
0398                    print 8, Z(16)
0399        8           format(' 8. The MODE of the SCALAR
                        is',F15.2)
0400                 else
0401                 endif
0402                 print 55, Z(6)
0403       55      FORMAT(' 9. The PHASE IN period is ',F4.1)
```

```
0404                      print 56, Z(7)
0405        56          FORMAT(' 10. This CE is a constant for
                        ',F4.1,' years')
0406                      print 57, Z(8)
0407        57            FORMAT(' 11. The PHASE OUT period is
                          ',F4.1,' years')
0408                      print 61, Z(9)
0409        61          FORMAT(' 12. This CE time period begins in
                        year ',F4.1)
0410                      print*,'13. Cost Driver changes?'
0411                      print*
0412                print*,'Would you like to change any of these?'
0413                      print*,'Enter:'
0414                      print*,'1 for YES'
0415                      print*,'2 for NO'
0416                      read*, yorn
0417                      asked = yorn
0418                      min = 1
0419                      max = 2
0420                      CALL ANS(min,max,asked,found)
0421                      yorn = asked
0422                      IF (yorn .EQ. 1) THEN
0423                        print*,'How many values would you
                            like to edit?'
0424                        print*,'(Enter a 0 to exit):'
0425                        read*, numchanges
0426                        asked = numchanges
0427                        min = 0
0428                        max = 13
0429                        CALL ANS(min,max,asked,found)
0430                        numchanges = asked
0431                        IF (numchanges .GT. 0) THEN
0432                          DO 62 k = 1, numchanges
0433                            print*
0434                          print*,'Enter the # of the value to
                              be changed:'
0435                            print*,'(ie, enter a 1 - 13)'
0436                            read*, jay
0437                            asked = jay
0438                            CALL ANS(min,max,asked,found)
0439                            jay = asked
0440                            IF (jay .EQ. 1) THEN
0441                 print*,'THIS CANNOT BE CHANGED HERE! YOU CAN
                     ONLY CHANGE'
0442                 print*,'THIS BY ENTERING A NEW DATA SET.'
0443                              print*
0444                              GO TO 555
0445                            ELSE IF (jay .GT. 1 .AND. jay
                                .LT. 9) THEN
0446                              jay = jay + 8
0447                              print*
0448                              print*,'Enter the NEW value:'
```

```
0449                              read*, Z(jay)
0450                            ELSE IF (jay .ge. 10 .and. jay
                                .lt. 13) then
0451                              jay = jay - 3
0452                              print*
0453                              print*,'Enter the NEW value:'
0454                              read*, Z(jay)
0455                            ELSE IF (jay .eq. 13) then
0456                              n = ifix(Z(10) + Z(11))
0457                              istart = (n*n + n + 2)
0458                              kk = 1
0459                              print*
0460                              do 999 iii = 1,n
0461                                 print 27, iii
0462       27                        format(' Cost Driver #',i3)
0463                                 print*
0464                  if (Z(12) .eq. 1.0 .and. kk .gt. 1) then
0465                     rlowout = exp(super(numcer,istart))
0466                     highout = exp(super(numcer,istart+1))
0467                     print 834, kk, rlowout
0468      834            format(I3,'.  LOW = ',f15.2)
0469                     print 835, kk+1, highout
0470      835            format(I3,'. HIGH = ',f15.2)
0471                  else
0472                     print 90, kk, super(numcer,istart)
0473       90                     format(I3,'.  LOW = ',f15.2)
0474                     print 91, kk+1, super(numcer,istart+1)
0475       91                     format(I3,'. HIGH = ',f15.2)
0476                   endif
0477            if (super(numcer,(n*n+4*n+1+iii)) .eq. 1.0) then
0478               print 92, kk+2, super(numcer,istart+2)
0479       92                     format(I3,'. TYPE = ',f7.1)
0480               else if (super(numcer,(n*n+4*n+1+iii)) .eq.
                   2.0) then
0481                    if (Z(12) .eq. 1.0 .and. kk .gt. 1) then
0482                      rmodeout = exp(super(numcer,istart+2))
0483                      print 836, kk+2, rmodeout
0484      836             format(I3,'. MODE = ',f15.2)
0485                    else
0486                      print 93, kk+2, super(numcer,istart+2)
0487       93             format(I3,'. MODE = ',f15.2)
0488                             print*
0489                    endif
0490                             else
0491                             endif
0492                             print*
0493                             kk = kk + 3
0494                             istart = istart + 3
0495      999                  continue
0496                         print*
0497                         print*,'How many cost drivers
                             need editing?'
```

```
0498                          print*,'(enter 0 to exit)'
0499                          read*, ichange
0500                          asked = ichange
0501                          min = 0
0502                          max = 3*n
0503                          CALL ANS(min,max,asked,found)
0504                          ichange = asked
0505                          if (ichange .gt. 0) then
0506                            ceredit = .true.
0507                            super(numcer,505) = n
0508                            do 97 jjj = 1, ichange
0509                             print*
0510                             print*,'Enter the # of the
                                 value to be'
0511                             print 98, max
0512       98                    format(' changed (ie, a 1
                                  -',I3,')')
0513                             read*, jaycd
0514                             asked = jaycd
0515                             min = 1
0516                             CALL ANS(min,max,asked,found)
0517                             jaycd = asked
0518                             print*
0519                             print*,'Enter the NEW value'
0520                             read*, XSTAR(jaycd)
0521                      if (Z(12) .eq. 1.0) then
0522                         ll = 1
0523                         got = .false.
0524                         m = 1
0525       573               if (got .eq. .false.) then
0526                 if (jaycd .gt. ((m-1)*n) .and. jaycd .lt.
                     (m*n+1)) then
0527                     got = .true.
0528                 else
0529                     m = m + 1
0530                     ll = ll + 1
0531                 endif
0532                 go to 573
0533                        else
0534                        endif
0535                              if (Z(11) .eq. 0.0) then
0536                                 ibeg = 1
0537                              else
0538                                 ibeg = 4
0539                              endif
0540                  if (jaycd .ge. ibeg .and. (n*n+4*n+1+ll)
                      .eq. 2.0) then
0541                     XSTAR(jaycd) = LOG(XSTAR(jaycd))
0542                    super(numcer,(n*n+n+1+jaycd)) =
                       XSTAR(jaycd)
0543                else if (MOD(jaycd,3) .ne. 0) then
0544                     XSTAR(jaycd) = LOG(XSTAR(jaycd))
```

```
0545                    super(numcer,(n*n+n+1+jaycd)) =
                        XSTAR(jaycd)
0546                 else
0547                 endif
0548                       else
0549                             super(numcer,(n*n+n+1+jaycd)) =
                                 XSTAR(jaycd)
0550                       endif
0551       97                    continue
0552                          else
0553                          endif
0554                          ELSE
0555                          ENDIF
0556                          CE(ieye,jay) = Z(jay)
0557       62                 continue
0558                       ELSE
0559                       ENDIF
0560                    ELSE
0561                    ENDIF
0562                 ENDIF
0563              ELSE
0564              ENDIF
0565       30    continue
0566             ELSE
0567             ENDIF
0568          ELSE
0569
0570    C     OLDNEW equals 2, ie new data set is to be entered.
              In this
0571    c     section the user receives a series of prompts
              regarding the
0572    c     status of the cost components to be entered.  The
              value of
0573    c     lastset remains False for this section.
0574
0575          print*
0576          print*,'You are entering a new data set.  If an
              incorrect value'
0577          print*,'is entered, continue entering the remaining
              values.'
0578          print*,'Changes may be made by rerunning this
              program and then'
0579          print*,'selecting the option to edit the data.'
0580          print*
0581          print*,'Enter the number of cost elements: '
0582          read*,NCOST
0583          print*,'Enter the life cycle of the system (in
              years): '
0584          read*,NYEARS
0585          print*,'Enter the discount rate (ie, if the
              discount rate:'
0586          print*,'is 9%, enter .09 or 0.09)'
```

```
0587          read*,DISCOUNT
0588
0589     c    Initialization of all arrays is conducted below by
              setting
0590     c    them equal to zero.
0591
0592          DO 1 I=1,NRV
0593          LC(I)=0.0
0594        1 CONTINUE
0595          DO 2 I=1,NCOST
0596          DO 2 J=1,20
0597             CE(I,J)=0.0
0598        2 CONTINUE
0599          DO 5 J=1,NCOST
0600          DO 5 I=1,NRV
0601             A(I,J)= 0.0
0602        5 CONTINUE
0603          DO 6 J=1,100
0604             COST(J)= 0.0
0605        6 CONTINUE
0606
0607          J=0
0608          DO 100 II=1,NCOST
0609
0610     c    Begin loop to enter the new cost elements.
0611
0612          DO 7 I = 1,20
0613             Z(I)= 0.0
0614        7 CONTINUE
0615          J=J+1
0616          CALL CLRSCR
0617          print*
0618          print 222,II
0619      222 FORMAT(' What type is cost component',I3,'?')
0620          print*,'1. a constant?'
0621          print*,'2. a random variable modeled by either'
0622          print*,'   a Beta or Triangular pdf?'
0623          print*,'3. a random variable generated by a CER?'
0624          print*
0625          print*,'Enter a 1, 2, or 3'
0626          read*, Z(1)
0627          asked = Z(1)
0628          min = 1
0629          max = 3
0630          CALL ANS(min,max,asked,found)
0631          Z(1) = asked
0632          print*
0633          if (Z(1) .EQ. 1) then
0634
0635     c    The question asked of the user here relate to this
              cost
0636     c    component being a constant.  Notice the only
```

```
            information
0637    c       needed is the cost.
0638
0639                print 225,II
0640          225   FORMAT(' Enter the constant cost of
                    component',I3,': ')
0641                print*,'(If the cost is $25,000 then enter
                    25000)'
0642                read*, Z(2)
0643              ELSE IF (Z(1) .EQ. 2) THEN
0644
0645    c         These questions reflect that this cost component is
                  a random
0646    c         variable.
0647
0648           print*,'Do you wish this component to be modeled by:'
0649                print*,'   1. a Beta pdf, or'
0650                print*,'   2. a Triangular pdf'
0651                print*,'Enter a 1 or 2: '
0652                read*, answer
0653                asked = answer
0654                min = 1
0655                max = 2
0656                CALL ANS(min,max,asked,found)
0657                answer = asked
0658                Z(20) = answer
0659
0660    c         Z(20) is assigned here as the response to whether
                  this cost
0661    c         component is to be modeled as either a beta or
                  triangular
0662    c         distribution.
0663
0664                IF (ANSWER .EQ. 1) THEN
0665           81      print*
0666                   print*,'Enter the LOW value of the BETA
                       distribution: '
0667                   print*,'(For example, if the LOW value of the
                       BETA'
0668                   print*,'is $25,000, enter 25000.)'
0669                   read*, Z(3)
0670                   print*
0671                   print*,'Enter the HIGH value of the BETA
                       distribution: '
0672                   read*, Z(4)
0673                   IF (Z(3) .GT. Z(4)) THEN
0674                      print*
0675                      print*,'Your LOW Beta value exceeds the
                          HIGH Beta'
0676                      print*,'value -- check your numbers and
                          try again.'
0677                      GO TO 81
```

```
0678                ELSE
0679                ENDIF
0680                print*
0681                print*,'Enter a 1-9 corresponding to the Type
                    # of the'
0682               print*,'BETA distribution for this component: '
0683                read*, Z(5)
0684              ELSE
0685         82      print*
0686                print*,'Enter the LOW value of the TRIANG
                    distribution:'
0687                print*,'(For example, if the LOW value of the
                    TRIANG'
0688                print*,'distro is $25,000, enter 25000.)'
0689                read*, Z(3)
0690                print*
0691                print*,'Enter the HIGH value of the TRIANG
                    distribution:'
0692                read*, Z(4)
0693                IF (Z(3) .GT. Z(4)) THEN
0694                   print*
0695                   print*,'Your LOW value of the TRIANG
                       distro exceeds'
0696                   print*,'the HIGH value -- check your
                       numbers and'
0697                   print*,'try again.'
0698                   GO TO 82
0699                ELSE
0700                ENDIF
0701                print*
0702           print*,'Enter the MODE or BEST GUESS of the TRIANG'
0703                print*,'distribution: (ie, enter a value
                    between the'
0704                print*,'LOW and HIGH values)'
0705                read*, Z(5)
0706                if (Z(5) .lt. Z(3) .or. Z(5) .gt. Z(4) ) then
0707                   print*
0708                   print*,'Your MODE value is either less
                       than the LOW'
0709                   print*,'value or greater than the HIGH
                       value --'
0710                   print*,'check your numbers and try again.'
0711                   go to 82
0712                else
0713                endif
0714             ENDIF
0715           ELSE IF (Z(1) .EQ. 3) THEN
0716
0717       c      This cost component is a cost estimating
                 relationship.
0718
0719              print*
```

```
0720            print*,'Enter the number of of cost drivers in
                the CER.'
0721            print*,'(ie, the number of SLOPE parameters --
                BUT DO:'
0722            print*,'NOT include the intercept term (if
                applicable)):'
0723            read*, Z(10)
0724            print*
0725            print*,'Enter a 1 if an intercept term IS
                present --'
0726            print*,'enter a 0 is NO intercept term is
                present: '
0727            read*, Z(11)
0728            asked = Z(11)
0729            min = 0
0730            max = 1
0731            CALL ANS(min,max,asked,found)
0732            Z(11) = asked
0733            print*
0734            print*,'Enter a 1 if the CER was estimated by
                taking'
0735            print*,'logs (Base e) -- enter a 0 if log
                estimation'
0736            print*,'is NOT desired: '
0737            read*, Z(12)
0738            asked = Z(12)
0739            CALL ANS(min,max,asked,found)
0740            Z(12) = asked
0741            print*
0742            print*,'Enter a 1 if the final cost estimate is
                to be'
0743            print*,'multiplied by a scalar.  For example, if
                the CER'
0744            print*,'describes the cost of buying a radio,
                but you intend'
0745            print*,'to purchase 50 to 60 radios, enter a 1,'
0746            print*,'otherwise enter a 0: '
0747            read*, Z(13)
0748            asked = Z(13)
0749            CALL ANS(min,max,asked,found)
0750            Z(13) = asked
0751            IF (Z(13) .EQ. 1) THEN
0752               print*
0753          print*,'Do you wish this scalar to be modeled by:'
0754               print*,' 1. a Beta pdf, or'
0755               print*,' 2. a Triangular pdf'
0756               print*,'Enter a 1 or 2:'
0757               read*, Z(20)
0758               min = 1
0759               max = 2
0760               asked = Z(20)
0761               CALL ANS(min,max,asked,found)
```

```
0762                Z(20) = asked
0763                print*
0764                print*,'Enter the LOW value of the scalar: '
0765                read*, Z(14)
0766                print*
0767                print*,'Enter the HIGH value of the scalar
                    (if the'
0768                print*,'value of the scalar is certain, set
                    both the'
0769                print*,'LOW and HIGH values equal): '
0770                read*, Z(15)
0771                print*
0772                if (Z(20) .eq. 1.0) then
0773                   print*,'Enter the type of Beta
                       Distribution:'
0774                   print*,'(ie, a 1 - 9)'
0775                else if (Z(20) .eq. 2.0) then
0776                print*,'Enter the MODE or BEST GUESS of the'
0777                   print*,'Triangular distribution:'
0778                else
0779                endif
0780                read*, Z(16)
0781             ELSE
0782             ENDIF
0783          ELSE
0784          ENDIF
0785
0786     C      CHECK FOR TIME PHASING OF THE COST ELEMENT.
0787
0788          print*
0789          print*,'The next three values relate to the Time
             Phasing of'
0790          print*,'the cost element.  Be sure the total length
             of your'
0791          print*,'PHASE-IN, CONSTANT COST, and PHASE-OUT does
             not exceed'
0792          print 226,NYEARS
0793        226 FORMAT(' the total LIFE CYCLE you specified earlier
             (',I3,').')
0794         67 print*
0795          print*,'Enter the component phase-in period (in
             years): '
0796          read*, Z(6)
0797          print*
0798         print*,'Enter # of years cost element is a constant'
0799          print*,'cost per year: '
0800          read*, Z(7)
0801          print*
0802          print*,'Enter the component phase-out period (in
             years): '
0803          read*, Z(8)
0804          print*
```

```
0805          print*,'Enter the time period when the element
              starts: '
0806          print*,'(NOTE: if the 1st realization of this
              element is in'
0807          print*,'year zero then enter 0.)'
0808          read*, Z(9)
0809
0810    c     The check below ensures all of the time phasing
              values are NOT
0811    c     equal to zero and that their sum does not exceed
              the life
0812    c     cycle of the system.
0813
0814          totalzs = Z(6) + Z(7) + Z(8) + Z(9)
0815          IF(Z(6) .EQ. 0.0 .AND. Z(7) .EQ. 0.0
              .AND.Z(8).EQ.0.0) THEN
0816         print*,'Check the Time Phasing values again, they'
0817             print*,'CANNOT ALL BE ZERO!'
0818             GO TO 67
0819          ELSE IF (totalzs .GT. NYEARS) THEN
0820             print*,'You have exceeded the life cycle of the
                 system --'
0821             print*,'please check your numbers and try
                 again.'
0822             GO TO 67
0823          ENDIF
0824
0825    c     Assign the Z(I) values to their correct position in
              the array
0826    c     CE(I,J) for later use within the program.
0827
0828          DO 16 I=1,20
0829             CE(J,I)=Z(I)
0830       16 CONTINUE
0831
0832    c     If cost element J is a "constant" then cost element
              J in
0833    c     A(I,J) will be the same for all NRV values of
              A(I,J)
0834
0835          IF(Z(1) .EQ. 1.0) THEN
0836             DO 25 I=1,NRV
0837                A(I,J)=Z(2)
0838       25    CONTINUE
0839
0840    c     If cost element J is a "random variable" then
              determine
0841    c     if the random variable is from either the beta or
              triang
0842    c     distribution and make the corresponding subroutine
              call.
0843
```

```
0844          ELSE IF(Z(1) .EQ. 2.0) THEN
0845             IF (Z(20) .EQ. 1.0) THEN
0846                CALL BETA(J,A,NRV,NCOST)
0847             ELSE IF (Z(20) .EQ. 2.0) THEN
0848                CALL TRIAG(J,A,NRV,NCOST)
0849             ELSE
0850             ENDIF
0851
0852     c       If cost element J is a "CER" then AAA and AA are
                 calculated
0853     c       before calling the subroutine CER.  AAA sums the
                 number of
0854     c       slope parameters plus the intercept term (if
                 applicable).
0855     c       AA is 3 times AAA for later use with the array
                 XSTAR.
0856
0857             ELSE IF(Z(1) .EQ. 3.0) THEN
0858               AAA=Z(10) + Z(11)
0859               AA=3.0 * AAA
0860               CALL CER (J, A, NRV, NCOST, BET, AAA, XSTAR, AA,
                   XCER, LASTSET,
0861              1numcer,super)
0862             ELSE
0863             ENDIF
0864          100 CONTINUE
0865             ENDIF
0866
0867     c       Store the just input data set into the file
                 INPUTLCC.DAT.
0868
0869             REWIND 01
0870             write(01,101) NCOST,NYEARS,DISCOUNT
0871          101 FORMAT(I3,1X,I3,1X,F4.3)
0872             DO 102 I = 1,NCOST
0873                DO 103 J = 1,20
0874                   write(01,104) CE(I,J)
0875          104      FORMAT(F18.4)
0876          103   continue
0877          102 continue
0878
0879     c       If the last data set was used, the following
                 statements call
0880     c       all of the same subroutines as shown above.  They
                 are called
0881     c       here to make sure any changes made to the data have
                 been
0882     c       saved before the changed data is used in
                 calculating the NRV
0883     c       costs of the system.
0884
0885             IF (lastset .EQ. .TRUE.) THEN
```

```
0886              nc = 0
0887              DO 130 iredo = 1, NCOST
0888
0889        c     Put the data from the array CE(I,J) into the
                  array Z(I).
0890        c     This is done NCOST times -- once for each cost
                  element.
0891
0892                 DO 131 k = 1, 20
0893                    Z(k) = CE(iredo,k)
0894          131    continue
0895              IF (Z(1) .EQ. 1.0) THEN
0896
0897        c     Put the value of the constant into the array A.
0898
0899                 DO 132 i = 1, NRV
0900                   A(i,iredo) = Z(2)
0901          132    continue
0902              ELSE IF (Z(1) .EQ. 2.0) THEN
0903
0904        c     Cost element iredo is a random variable generated by
                  either the
0905        c     Beta or Triangular distribution, depending on the
                  value of Z(20). 0906
0907                 IF (Z(20) .EQ. 1.0) THEN
0908                    CALL BETA (iredo,A,NRV,NCOST)
0909                 ELSE
0910                    CALL TRIAG (iredo,A,NRV,NCOST)
0911                 ENDIF
0912              ELSE
0913                 nc = nc + 1
0914                 numcer = nc
0915                 AAA = Z(10) + Z(11)
0916                 AA = 3.0 * AAA
0917                 CALL CER (iredo, A, NRV, NCOST, BET, AAA,
                     XSTAR,AA,XCER,
0918             1LASTSET,numcer,super)
0919                 ENDIF
0920          130 continue
0921              ELSE
0922              ENDIF
0923              if (ceredit .eq. .true.) then
0924                 rewind 02
0925                 do 850 i = 1, numcer
0926                    n = ifix(super(i,505))
0927                    do 851 j = 1, (n*n+n+1)
0928                       write(02,852) super(i,j)
0929          852          format(f31.16)
0930          851       continue
0931                    do 853 j = (n*n+n+2), (n*n+4*n+1)
0932                       write(02,854) super(i,j)
0933          854          format(f15.8)
```

```
0934      853        continue
0935                 do 855 j = (n*n+4*n+2), (n*n+5*n+1)
0936                    write(02,856) super(i,j)
0937      856           format(f5.3)
0938      855        continue
0939      850     continue
0940           else
0941           endif
0942
0943     C     TO CALCULATE THE LIFE CYCLE COST OF THE SYSTEM THE
0944     C     ELEMENTS OF MATRIX A(NRV,20) MUST BE DISTRIBUTED
                OVER
0945     C     TIME. SUBROUTINE TRAP ACCOMPLISHES THIS.
0946
0947     C     JJ IS THE RANDOM VARIABLE NUMBER WE ARE CURRENTLY
0948     C     WORKING ON, SO JJ=1 => OBSERVATION # 1 OR IN THE
0949     C     CASE OF A COST ELEMENT WHICH IS A RANDOM VARIABLE
0950     C     JJ=1 => RANDOM SAMPLE #1.
0951
0952           JJ = 1
0953           CALL TRAP(A,NRV,NCOST,CE,JJ,NYEARS)
0954
0955     C     COSTC(100) CONTAINS THE TOTAL OF THE CONSTANT COST
0956     C     ELEMENTS DISTRIBUTED OVER 100 YEARS.
0957     C     COSTR(100) CONTAINS THE TOTAL OF THE COST ELEMENTS
0958     C     WHICH ARE RANDOM COST COMPONENTS MODELED AS BETA
0959     C     DISTRIBUTIONS.
0960
0961     C     TRANSFER COSTC(100) TO PROTECTED STORAGE
                [STORE(100)]
0962     C     SINCE WE NEED IT REPEATEDLY.
0963
0964           DO 105 I=1,NYEARS
0965              STORE(I)=COSTC(I)
0966        105 CONTINUE
0967
0968     C     ADD COSTC(I) TO COSTR(I) FOR EACH YEAR.
0969
0970           DO 110 I=1,NYEARS
0971              COST(I)=COSTC(I) + COSTR(I)
0972        110 CONTINUE
0973
0974     C     CALCULATE THE LIFE CYCLE COST FOR OBSERVATION # 1
0975
0976           T1=0.0
0977           X=0.0
0978           DO 120 I=1,NYEARS
0979              T1 = T1 + 1.0
0980              X=X +(COST(I)/((1.0 + DISCOUNT)**T1))
0981        120 CONTINUE
0982           LC(1)=X
0983
```

```
0984     C      CALCULATE THE LIFE CYCLE COST FOR OBSERVATION # 2
                THRU
0985     C      NRV
0986
0987            DO 140 I1 = 2, NRV
0988               JJ = JJ + 1
0989               CALL TRAP(A,NRV,NCOST,CE,JJ,NYEARS)
0990               DO 125 I = 1, NYEARS
0991                  COST(I) = STORE(I) + COSTR(I)
0992       125     CONTINUE
0993
0994     C      CALCULATE LIFE CYCLE COST: OBSERVATIONS #2 THRU #
                NRV
0995
0996               T1 = 0.0
0997               X = 0.0
0998               DO 127 I = 1, NYEARS
0999                  T1 = T1 + 1.0
1000                  X = X+(COST(I)/((1.0 + DISCOUNT)**T1))
1001       127     CONTINUE
1002               LC(I1) = X
1003       140 CONTINUE
1004
1005     c      Place the values calculated for the NRV LC(I)
                values
1006     c      into the file LCCOST.DAT.
1007
1008            WRITE(05,151) (LC(I),I=1,NRV)
1009       151 FORMAT(8(2X,F19.0))
1010            STOP
1011            END
```

PROGRAM SECTIONS

ENTRY POINTS

Address	Type	Name	References
0-00000000		LCC	5#

VARIABLES

Address	Type	Name	References			
2-0002956C	R*4	AA	859=	860A	916=	917A
2-00029568	R*4	AAA	858=	859	860A	915=
			916	917A		
2-00029564	R*4	ANSWER	652=	653	657=	658
			664			
2-0002950C	R*4	ASKED	61=	65A	66	128=

			129A	130	140=	141A
			142	165=	166A	167
			191=	194A	195	224=
			225A	226	231=	234A
			235	327=	330A	331
			336=	339A	340	417=
			420A	421	426=	429A
			430	437=	438A	439
			500=	503A	504	514=
			516A	517	627=	630A
			631	653=	656A	657
			728=	731A	732	738=
			739A	740	748=	749A
			750	760=	761A	762
2-000294F8	L*4	CEREDIT	6	46=	506=	923
2-00029520	R*4	DISCOUNT	84=	158	171=	587=
			870	980	1000	
2-000294F0	L*4	FOUND	6	64=	65A	129A
			141A	166A	194A	225A
			234A	330A	339A	420A
			429A	438A	503A	516A
			630A	656A	731A	739A
			749A	761A		
**	L*4	GOT	6	523=	525	527=
**	R*4	HIGHOUT	466=	469		
**	I*4	I	47=	49	86=	88
			92=	93	95(2)	592=
			593	595=	597	600=
			601	612=	613	828=
			829(2)	836=	837	872=
			874	899=	900	925=
			926	928	932	936
			964=	965(2)	970=	971(3)
			978=	980	990=	991(3)
			998=	1000	1008(2)=	
2-00029580	I*4	I1	987=	1002		
**	I*4	IBEG	536=	538=	540	
2-00029550	I*4	ICHANGE	499=	500	504=	505
			508			
2-0002952C	I*4	IEYE	175=	182	185	198
			275	357	556	
2-0002955C	I*4	II	608=	618	639	
2-00029524	I*4	III	96=	97	98	100(2)
			101	104	460=	461
			477	480		
2-00029574	I*4	IREDO	887=	893	900	908A
			910A	917A		
**	I*4	ISTART	457=	465	466	472
			474	478	482	486
			494(2)=			
2-00029504	I*4	J	48=	49	87=	88
			181=	182(2)	596=	597

			599=	601	603=	604
			607=	615(2)=	829	837
			846A	848A	860A	873=
			874	927=	928	931=
			932	935=	936	
2-0002953C	I*4	JAY	254=	255(2)	257	258(2)=
			261	262(2)	263(2)=	269
			275(2)	346=	347	353(2)=
			355	357(2)	436=	437
			439=	440	445(2)	
			446(2)=	449	450(2)	
			451(2)=	454	455	556(2)
2-00029558	I*4	JAYCD	513=	514	517=	520
			526(2)	540	541(2)	
			542(2)	543	544(2)	
			545(2)	549(2)		
2-0002957C	I*4	JJ	952=	953A	988(2)=	989A
**	I*4	JJJ	508=			
2-00029534	I*4	K	249=	342=	432=	892=
			893(2)			
**	I*4	KK	458=	464	467	469
			472	474	478	481
			483	486	493(2)=	
2-000294F4	L*4	LASTSET	6	67=	111=	860A
			885	917A		
**	I*4	LL	522=	530(2)=	540	
**	I*4	M	524=	526(2)	529(2)=	
2-00029514	I*4	MAX	63=	65A	129A	141A
			166A	193=	194A	225A
			233=	234A	329=	330A
			338=	339A	419=	420A
			428=	429A	438A	502=
			503A	511	516A	629=
			630A	655=	656A	730=
			731A	739A	749A	759=
			761A			
2-00029510	I*4	MIN	62=	65A	129A	141A
			166A	192=	194A	225A
			232=	234A	328=	330A
			337=	339A	418=	420A
			427=	429A	438A	501=
			503A	515=	516A	628=
			630A	654=	656A	729=
			731A	739A	749A	758=
			761A			
**	I*4	N	95=	96(3)	97(3)	
			100(6)	456=	457(3)	460
			477(3)	480(3)	502	507
			526(2)	540(3)	542(3)	
			545(3)	549(3)	926=	
			927(3)	931(6)	935(6)	
2-00029570	I*4	NC	886=	913(2)=	914	

2-00029518	I*4	NCOST	84= 86 92 175 582= 595 599 608 846A 848A 860A 870 872 887 908A 910A 917A 953A 989A
2-000294FC	I*4	NRV	12D 592 600 836 846A 848A 860A 899 908A 910A 917A 953A 987 989A 1008
2-00029500	I*4	NUMCER	45= 94(2)= 98 101 104 112= 184(2)= 465 466 472 474 477 478 480 482 486 507 542 545 549 860A 914= 917A 925
2-00029530	I*4	NUMCHANGES	230= 231 235= 236 249 335= 336 340= 341 342 425= 426 430= 431 432
2-0002951C	I*4	NYEARS	84= 133 147= 266 584= 792 819 870 953A 964 970 978 989A 990 998
2-00029508	R*4	OLDNEW	60= 61 66= 77
2-0002954C	R*4	RLOWOUT	465= 467
**	R*4	RMODEOUT	482= 483
**	R*4	T1	976= 979(2)= 980 996= 999(2)= 1000
**	R*4	TOTALZS	814= 819
**	R*4	X	977= 980(2)= 982 997= 1000(2)= 1002
2-00029528	R*4	YORN	127= 128 130= 132 139= 140 142= 144 164= 165 167= 168 190= 191 195= 197 223= 224 226= 227 326= 327 331= 332 416= 417 421= 422

ARRAYS

Address	Type	Name	Bytes	Dimensions	References
2-00001770	R*4	A	60000	(750, 20)	7 601= 837= 846A 848A 860A 900= 908A 910A 917A 953A 989A
2-00010B30	R*4	BET	80	(20)	7 860A 917A

2-0001DF60	R*4	BORT	80	(20)	7	
2-00010360	R*4	CE	1600	(20, 20)	7	88=
					93	95(2)
					182	275=
					357=	556=
					597=	829=
					874	893
					953A	989A
2-000101D0	R*4	COST	400	(100)	7	604=
					971=	980
					991=	1000
4-00000000	R*4	COSTC	400	(100)	11	965
					971	
4-00000190	R*4	COSTR	400	(100)	11	971
					991	
2-00000000	R*8	LC	6000	(750)	7	9
					593=	982=
					1002=	1008
2-000109A0	R*4	STORE	400	(100)	7	965=
					991	
2-0001F720	R*4	SUPER	40400	(20, 505)	7	49=
					98=	101=
					104=	465
					466	472
					474	477
					478	480
					482	486
					507=	542=
					545=	549=
					860A	917A
					926	928
					932	936
2-00010C70	R*4	XCER	60000	(750, 20)	7	860A
					917A	
2-00010B80	R*4	XSTAR	240	(60)	7	520=
					541(2)=	542
					544(2)=	545
					549	860A
					917A	
3-00000000	R*4	Z	80	(20)	10	182=
					184	201
					205	210
					212	214
					216	261=
					269=	275
					281	285
					295	297
					299	305
					307	310
					313	315
					317	319

355=	357
371	378
378	381
385	387
389	394
395	397
398	402
404	406
408	449=
454=	456(2)
464	481
521	535
556	613=
626=	627
631=	633
642=	643
658=	669=
672=	673(2)
683=	689=
692=	693(2)
705=	706(4)
715	723=
727=	728
732=	737=
738	740=
747=	748
750=	751
757=	760
762=	765=
770=	772
775	780=
796=	800=
803=	808=
814(4)	815(3)
829	835
837	844
845	847
857	858(2)
893=	895
900	902
907	915(2)

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References			
	ANS	65	129	141	166
		194	225	234	330
		339	420	429	438
		503	516	630	656
		731	739	749	761
	BETA	846	908		

```
        CER                     860       917
        CLRSCR                   44        76       616
        FOR$OPEN                 38        39        40
   R*4  MTH$ALOG                541       544
   R*4  MTH$EXP                 465       466       482
        TRAP                    953       989
        TRIAG                   848       910

0001
0002
0003     c      Subroutine ANS makes sure the range of possible
                inputs to the
0004     c      questions posed to the user is met.  If the user's
                response is
0005     c      outside of the min and max values, the loop is
                continued until
0006     c      an appropriate answer is given.  The logical
                variable FOUND is
0007     c      used as the flag stating whether an acceptable
                response has been
0008     c      attained or not.
0009
0010           SUBROUTINE ANS(min,max,asked,found)
0011         1 IF (asked .LT. min .OR. asked .GT. max) THEN
0012              found = .FALSE.
0013           ELSE
0014              found = .TRUE.
0015           ENDIF
0016           IF (found .EQ. .FALSE.) THEN
0017              print 10,min,max
0018        10    FORMAT(' Please enter a number from',I2,'
                  to',I3)
0019              read*,asked
0020              GOTO 1
0021           ENDIF
0022           RETURN
0023           END


PROGRAM SECTIONS

ENTRY POINTS

    Address  Type  Name                          References

  0-00000000       ANS                             10#

VARIABLES

    Address  Type  Name          References

 AP-0000)00C@ R*4  ASKED         10         11(2)      19=
```

```
 AP-00000010@ R*4  FOUND          10          12=         14=          16
 AP-00000008@ I*4  MAX            10          11          17
 AP-00000004@ I*4  MIN            10          11          17

LABELS

    Address   Label                                 References

  0-00000004  1                                       11#         20
  1-00000000  10'                                     17          18#


0001
0002     C      This subroutine is designed to clear the screen of
                the previous
0003     c      material.  The purpose is so only material
                pertinent to
0004     c      the user is in front of him/her.
0005
0006            SUBROUTINE CLRSCR
0007            DO 1 i = 1, 23
0008               print*
0009          1 continue
0010            RETURN
0011            END

PROGRAM SECTIONS

ENTRY POINTS

    Address  Type  Name                          References

  0-00000000       CLRSCR                           6#

VARIABLES

    Address  Type  Name          Attributes  References

      **     I*4   I                           7=

LABELS

    Address   Label                           References

      **      1                                 7          9#


0001
0002
0003     C      The subroutine Triag is designed to get NRV random
                samples
0004     c      from the triangular distribution.  The seed for
```

```
            rnset is set
0005    c   to zero so that the seed is based upon the clock of
            the
0006    c   computer.  Random numbers are received based on the
            uniform
0007    c   distribution within the range 0,1.  These random
            numbers are
0008    c   then converted to the triangular distribution by
            comparing
0009    c   the random number against the variable FOFX.  FOFX
            represents
0010    c   the cumulative triangular distribution according to
            the data
0011    c   input by the user (contained in Z(3), Z(4), and
            Z(5)).  Z(3)
0012    c   corresponds to the low value of the triag distro,
            Z(4) the high
0013    c   value, and Z(5) to the mode or best guess of the
            triag
0014    c   distribution.
0015
0016        SUBROUTINE TRIAG(J,A,NRV,NCOST)
0017        DIMENSION A(NRV,NCOST)
0018        COMMON /AA/ Z(20)
0019        INTEGER ISEED
0020        PARAMETER (NR=750)
0021        REAL R(NR),FOFX,ALOW,BMID,CHIGH
0022        EXTERNAL RNSET,RNUN
0023        call rnset(0)
0024
0025    c   The call to rnun gets nr uniformly distributed
            values and
0026    c   puts them in the array r.
0027
0028        call rnun(nr,r)
0029        alow = Z(3)
0030        bmid = Z(5)
0031        chigh = Z(4)
0032    c   If the LOW and HIGH values are equal then the value
            of this
0033    c   cost element is known for certain.
0034        if (alow .eq. chigh) then
0035           do 5 i = 1,nr
0036              a(i,j) = alow
0037        5  continue
0038        else
0039    c    FOFX represents the "break" point in the Triangular
            distribution.
0040    c    If the random number generated equals fofx, assign
            A(I,J) to the
0041    c   MODE, otherwise the corresponding equation is
            executed to
```

```
0042     c  /  determine the value of A(I,J).
0043              fofx = ((bmid - alow)/(chigh - alow))
0044              DO 10 i = 1, nr
0045                 IF (r(i) .EQ. fofx) THEN
0046                    A(I,J) = bmid
0047                 ELSE IF (r(i) .LT. fofx) THEN
0048                  A(I,J) = (alow + SQRT(r(i)*(bmid -
                      alow)*(chigh - alow)))
0049                 ELSE
0050                    A(I,J) = (chigh - SQRT((chigh -
                      alow)*(chigh - bmid)*
0051           1(1 - r(i))))
0052                 ENDIF
0053        10    continue
0054           endif
0055           RETURN
0056           END
```

PROGRAM SECTIONS

ENTRY POINTS

Address	Type	Name	References
0-00000000		TRIAG	16#

VARIABLES

Address	Type	Name	References			
**	R*4	ALOW	21	29=	34	36
			43(2)	48(3)	50	
**	R*4	BMID	21	30=	43	46
			48	50		
**	R*4	CHIGH	21	31=	34	43
			48	50(3)		
**	R*4	FOFX	21	43=	45	47
**	I*4	I	35=	36	44=	45
			46	47	48(2)	50(2)
**	I*4	ISEED	19			
AP-00000004@	I*4	J	16	36	46	48
			50			
AP-00000010@	I*4	NCOST	16	17		
AP-0000000C@	I*4	NRV	16	17		

ARRAYS

Address	Type	Name	Bytes	Dimensions	References		
AP-00000008@	R*4	A	**	(*, *)	16	17	36=
					46=	48=	50=
2-00000000	R*4	R	3000	(750)	21	28A	45

```
                                                    47        48        50
  3-00000000  R*4   Z            80   (20)          18        29        30
                                                    31
```

PARAMETER CONSTANTS

Type	Name	References
I*4	NR	20# 21 28 35 44

LABELS

Address	Label	References
**	5	35 37#
**	10	44 53#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
R*4	MTH$SQRT	48 50
	RNSET	22 23
	RNUN	22 28

```
0001
0002
0003     c      This subroutine is the same as TRIAG except it is
                designed
0004     c      specifically for use when a cost driver is a random
                varible.
0005            subroutine triag1(j,xlow,xhigh,xmode,nrv,r)
0006            integer iseed
0007            parameter (nr=750)
0008            real r(nr),fofx,xlow,xhigh,xmode
0009            external rnset,rnun
0010
0011            call rnset(0)
0012            call rnun(nr,r)
0013            if (xlow .eq. xhigh) then
0014               do 5 i = 1,nr
0015                  r(i) = xlow
0016          5    continue
0017            else
0018               fofx = ((xmode - xlow)/(xhigh - xlow))
0019               do 10 i = 1,nr
0020                  if (r(i) .eq. fofx) then
0021                     r(i) = xmode
0022                  else if (r(i) .lt. fofx) then
```

```
0023                    r(i) = (xlow + SQRT(r(i)*(xmode -
                        xlow)*(xhigh - xlow)))
0024                   else
0025                      r(i) = (xhigh - SQRT
                          ((xhigh-xlow)*(xhigh-xmode)*
0026           1(1-r(i))))
0027                    endif
0028         10    continue
0029            endif
0030            return
0031            end
```

PROGRAM SECTIONS

ENTRY POINTS

Address	Type	Name	References
0-00000000		TRIAG1	5#

VARIABLES

Address	Type	Name	References			
**	R*4	FOFX	8	18=	20	22
**	I*4	I	14=	15	19=	20
			21	22	23(2)	25(2)
**	I*4	ISEED	6			
AP-00000004@	I*4	J	5			
AP-00000014@	I*4	NRV	5			
AP-0000000C@	R*4	XHIGH	5	8	13	18
			23	25(3)		
AP-00000008@	R*4	XLOW	5	8	13	15
			18(2)	23(3)	25	
AP-00000010@	R*4	XMODE	5	8	18	21
			23	25		

ARRAYS

Address	Type	Name	Bytes	Dimensions	References	
AP-00000018@	R*4	R	3000	(750)	5	8
					12A	15=
					20	21=
					22	23(2)=
					25(2)=	

PARAMETER CONSTANTS

Type	Name	References

```
   I*4   NR                        7#          8          12          14
                                  19
```

LABELS

Address	Label	References	
**	5	14	16#
**	10	19	28#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References	
R*4	MTH$SQRT	23	25
	RNSET	9	11
	RNUN	9	12

```
0001
0002
0003          SUBROUTINE BETA(J,A,NRV,NCOST)
0004          DIMENSION A(NRV,NCOST)
0005          COMMON /AA/ Z(20)
0006          INTEGER ISEED
0007          PARAMETER (NR=750)
0008          REAL PIN,QIN,R(NR)
0009          EXTERNAL RNBET,RNSET
0010
0011     C    CHECK FOR TYPE OF BETA DISTRIBUTION.  PIN
              corresonds to
0012     c    Alpha whereas QIN corresponds to Beta.
0013
0014          IF(Z(5).EQ.1.0) THEN
0015             PIN=2.5
0016             QIN=1.5
0017          ELSE IF(Z(5).EQ.2.0) THEN
0018             PIN=2.35
0019             QIN=2.35
0020          ELSE IF(Z(5).EQ.3.0) THEN
0021             PIN=1.5
0022             QIN=2.5
0023          ELSE IF(Z(5).EQ.4.0) THEN
0024             PIN=4.0
0025             QIN=2.0
0026          ELSE IF(Z(5).EQ.5.0) THEN
0027             PIN=3.75
0028             QIN=3.75
0029          ELSE IF(Z(5).EQ.6.0) THEN
0030             PIN=2.0
0031             QIN=4.0
```

```
0032            ELSE IF(Z(5).EQ.7.0) THEN
0033               PIN=5.5
0034               QIN=2.5
0035            ELSE IF(Z(5).EQ.8.0) THEN
0036               PIN=5.0
0037               QIN=5.0
0038            ELSE IF(Z(5).EQ.9.0) THEN
0039               PIN=2.5
0040               QIN=5.5
0041            ELSE
0042            ENDIF
0043
0044            CALL RNSET(0)
0045            CALL RNBET(NR,PIN,QIN,R)
0046            DO 10 I=1,NR
0047               A(I,J)=Z(3)+((Z(4)-Z(3))*R(I))
0048         10 CONTINUE
0049            RETURN
0050            END
```

PROGRAM SECTIONS

ENTRY POINTS

Address	Type	Name	References
0-00000000		BETA	3#

VARIABLES

Address	Type	Name	References			
**	I*4	I	46=	47(2)		
**	I*4	ISEED	6			
AP-00000004@	I*4	J	3	47		
AP-00000010@	I*4	NCOST	3	4		
AP-0000000C@	I*4	NRV	3	4		
2-00000BB8	R*4	PIN	8	15=	18=	21=
			24=	27=	30=	33=
			36=	39=	45A	
2-00000BBC	R*4	QIN	8	16=	19=	22=
			25=	28=	31=	34=
			37=	40=	45A	

ARRAYS

Address	Type	Name	Bytes	Dimensions	References	
AP-00000008@	R*4	A	**	(*, *)	3	4
					47=	
2-00000000	R*4	R	3000	(750)	8	45A

						47	
3-00000000	R*4	Z	80	(20)		5	14
						17	20
						23	26
						29	32
						35	38
						47(3)	

PARAMETER CONSTANTS

Type	Name	References			
I*4	NR	7#	8	45	46

LABELS

Address	Label	References	
**	10	46	48#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References	
	RNBET	9	45
	RNSET	9	44

```
0001
0002
0003     C      SUBROUTINE TRAP : HANDLES THE DISTRIBUTION OF COSTS
                OVER TIME
0004     C      USING TRAPEZOID RULES.
0005     C      JJ IS THE OBSERVATION WE ARE CURRENTLY WORKING ON.
0006     C      JJ=NRV RANDOM VARIABLES IS THE MAX. ALLOWED FOR THE
                SIMULATION.
0007            SUBROUTINE TRAP(A,NRV,NCOST,CE,JJ,NYEARS)
0008            DIMENSION A(NRV,NCOST),CE(20,20),ZZ(100),DEL(100),
0009           1PHASIN(100), PHACON(100),PHAOUT(100),
0010           2COSTC(100),COSTR(100)
0011            DOUBLE PRECISION VALUE,X,Y
0012            REAL*8 MCON(20,100),MRAND(20,100)
0013
0014     C      TO INCREASE THE NUMBER OF COST ELEMENTS ALTER ABOVE
                CE(*,20),
0015     C      MCON(*,100) AND MRAND(*,100). REPLACE * WITH THE
                NUMBER OF DESIRED
0016     C      COST ELEMENTS.
0017
0018            COMMON /AB/  COSTC,COSTR
0019
0020     C      COSTC(100) : COST ACCUMULATOR FOR THE CONSTANT COST
```

```
            ELEMENTS.
0021    C   COSTR(100) : COST ACCUMULATOR FOR THE RANDOM
            COMPONENTS.
0022
0023    C   CLEAR THE ARRAYS COSTC(100) AND COSTR(100). NOTE:
0024    C   THIS ALSO CLEARS THEM IN THE COMMON BLOCK. THE
            FIRST
0025    C   STATEMENT AFTER THE CALL STATEMENT IN THE
            MAINPROGRAM
0026    C   SHOULD BE TO TRANSFER COSTC(100) TO PROTECTED
            STORAGE.
0027
0028        DO 2 I=1,NCOST
0029        DO 2 J=1,100
0030           MCON(I,J)=0.0
0031           MRAND(I,J)=0.0
0032      2 CONTINUE
0033        DO 3 I=1,100
0034           COSTC(I)= 0.0
0035           COSTR(I)= 0.0
0036      3 CONTINUE
0037
0038    C   THE CURRENT VALUE OF EACH COST ELEMENT IS
            DISTRIBUTED
0039    C   OVER TIME. THE NEXT DO STATEMENT ACCOMPLISHES THIS.
0040    C  AFTER THE EXIT POINT THE COST ELEMENTS ARE STORED IN
0041    C   EITHER THE CONSTANT COST ACCUMULATOR [COSTC(100)]
0042    C   OR THE RANDOM COST ACCUMULATOR [COSTR(100)]
0043
0044        DO 500 I=1,NCOST
0045
0046    C   THE CONSTANT COST ELEMENTS PASS THROUGH THE PROGRAM
0047    C   ONLY ONE TIME SINCE ONCE CALCULATED THEY REMAIN THE
0048    C   SAME FOR ALL NRV ITERATIONS.
0049
0050        IF(JJ.EQ.1) GO TO 5
0051        IF(CE(I,1).EQ.1.0) GO TO 500
0052      5 VALUE=A(JJ,I)
0053        A1=CE(I,6)
0054        C1=CE(I,7)
0055        D1=CE(I,8)
0056        START=CE(I,9)
0057        TT=A1 + C1 + D1
0058        N2=IFIX(TT)
0059        NN1=IFIX(A1)
0060        NN2=IFIX(C1)
0061        NN3=IFIX(D1)
0062
0063   C Z1 IS THE % OF THE TOTAL AREA UNDER THE TRAPEZOID IN THE
0064    C   FIRST TRIANGLE( PHASE IN).
0065    C   Z2 IS THE % OF THE TOTAL AREA UNDER THE TRAPEZOID IN
THE
```

```
0066     C    RECTANGULAR REGION ( CONSTANT COST PER YEAR).
0067    C Z3 IS THE % OF THE TOTAL AREA UNDER THE TRAPEZOID IN THE
0068     C    RIGHT HAND TRIANGLE ( PHASE OUT).
0069
0070           Z1= A1/(A1 + D1 +(2.0 * C1))
0071           Z2= C1/(.50 *(A1 +D1+(2.0*C1)))
0072           Z3= D1/(A1 + D1 + (2.0*C1))
0073     C     PHASE IN ALLOCATION.
0074     C     CHECK FOR THE EXISTENCE OF THE FIRST TRIANGLE.
0075
0076           IF(A1.NE.0.0) THEN
0077              B=2.0/A1
0078           ELSE
0079              GO TO 50
0080           ENDIF
0081
0082     C     THE ALLOCATION FOR CONSTANT COST STARTS AT
               STATEMENT # 50
0083
0084           N1=IFIX(A1)
0085
0086     C  IF ALL OF THE AREA IS IN THE TRIANGLE ON THE RIGHT AND
0087     C     THERE IS ONE COST ELEMENT THEN YOU ARE DONE WITH
               THE COST
0088     C     ALLOCATION FOR THIS COST ELEMENT.
0089
0090           IF (Z1 .EQ. 1.0 .AND. A1 .EQ. 1.0) THEN
0091              PHASIN(1) = VALUE
0092              GO TO 300
0093           ELSE IF (A1 .EQ. 1.0 .AND. Z1 .NE. 1.0) THEN
0094              PHASIN(1) = Z1 * VALUE
0095              GO TO 50
0096           ELSE
0097           ENDIF
0098           DO 10 II=1,N1
0099              DD = FLOAT(II)
0100              ZZ(II) = (DD*B)/A1
0101        10 CONTINUE
0102           DO 20 II = 1,N1-1
0103              DD = FLOAT(II)
0104              DEL(II) = 0.50 * (ZZ(II)*DD)
0105        20 CONTINUE
0106           X = DEL(1)
0107           IF (N1-1 .LT. 2) GO TO 26
0108           DO 25 II = 2,N1-1
0109              DEL(II) = DEL(II) - X
0110              X = X + DEL(II)
0111        25 CONTINUE
0112        26 DEL(N1) = 1.0 - X
0113
0114     C     COST COMPONENT MULTIPLIERS
0115
```

```
0116          DO 27 II=1,N1
0117             ZZ(II) = DEL(II) * Z1
0118       27 CONTINUE
0119
0120    C     CALCULATE THE COST DISTRIBUTION FOR THE PHASE IN
              PERIOD
0121    C     AND STORE IN PHASIN(100)
0122
0123          DO 29 II=1,N1
0124             PHASIN(II)= ZZ(II) * VALUE
0125       29 CONTINUE
0126
0127    C     CONSTANT COST PART OF THE TIME HORIZON.
0128    C     CHECK FOR THE EXISTENCE OF THE RECTANGULAR REGION
0129    C     OF THE TRAPEZOID.
0130
0131       50 IF (C1 .EQ. 0.0) GO TO 100
0132
0133    C     DISTRIBUTE COST OVER THE RECTANGULAR REGION.
0134
0135          X = (Z2/C1) * VALUE
0136          N1 = IFIX(C1)
0137          DO 55 II = 1,N1
0138             PHACON(II) = X
0139       55 CONTINUE
0140
0141    C     CONSTANT COST PART OF TRAPEZOID IS COMPLETE.
0142    C     CALCULATE THE PHASEOUT PART OF THE COST ALLOCATION.
0143
0144    C     CHECK FOR THE EXISTENCE OF A PHASE OUT PERIOD.
0145
0146      100 IF (D1 .EQ. 0.0) GO TO 300
0147          B = 2.0/D1
0148          N1 = IFIX(D1)
0149
0150    C     IF ALL OF THE AREA IS IN THE PHASE OUT TRIANGLE AND
0151    C     THERE IS ONE COST ELEMENT YOU ARE DONE WITH THE
              COST
0152    C     ALLOCATION FOR THIS COST ELEMENT.
0153
0154          IF (Z3 .EQ. 1.0 .AND. D1 .EQ. 1.0) THEN
0155              PHAOUT(1) = VALUE
0156              GO TO 300
0157          ELSE IF (D1 .EQ. 1.0 .AND. Z3 .NE. 1.0) THEN
0158              PHAOUT(1) = Z3 * VALUE
0159              GO TO 300
0160          ELSE
0161          ENDIF
0162          DO 110 II = 1,N1
0163             DD = FLOAT(II)
0164             ZZ(II) = (DD*B)/D1
0165      110 CONTINUE
```

```
0166            DO 120 II = 1,N1-1
0167               DD = FLOAT(II)
0168               DEL(II) = .50 * (ZZ(II)*DD)
0169          120 CONTINUE
0170            X = DEL(1)
0171            IF (N1-1 .LT. 2) GO TO 126
0172            DO 125 II = 2,N1-1
0173               DEL(II) = DEL(II) - X
0174               X = X + DEL(II)
0175          125 CONTINUE
0176          126 DEL(N1) = 1.0 - X
0177            DO 127 II = 1,N1
0178               ZZ(II) = DEL(II) * Z3
0179          127 CONTINUE
0180
0181        C     CALCULATE THE COST DISTRIBUTION FOR THE PHASE OUT
0182        C     PERIOD AND STORE IN PHAOUT(100).
0183
0184            J1 = N1
0185            DO 129 II = 1,N1
0186               PHAOUT(II) = ZZ(J1) * VALUE
0187               J1 = J1 - 1
0188          129 CONTINUE
0189
0190        C     STATEMENT #300 FINISHES THE COST ALLOCATION BY
0191        C     TRAPEZOID SECTOR. THE NEXT BLOCK LINKS THE SEGMENTS
0192        C     OF THE TRAPEZOID SEQUENTIALLY AND THE WHOLE
                  STRUCTURE
0193        C     IS SHIFTED TO THE STARTING POINT(DEFINED BY
                  VARIABLE
0194        C     START)
0195
0196          300 CONTINUE
0197            DO 301 II = 1,100
0198               ZZ(II) = 0.0
0199          301 CONTINUE
0200            IF (Z1 .EQ. 1.0) THEN
0201                DO 305 II = 1,NN1
0202                   ZZ(II) = PHASIN(II)
0203          305      CONTINUE
0204                GO TO 350
0205            ELSE
0206                SUM = Z1 + Z2
0207            ENDIF
0208            IF (SUM .EQ. 1.0) THEN
0209               DO 306 II = 1,NN1
0210                  ZZ(II) = PHASIN(II)
0211          306    CONTINUE
0212               J1 = 1
0213                NEND = NN1 + NN2
0214                DO 307 II = NN1+1,NEND
0215                   ZZ(II) = PHACON(J1)
```

```
0216                  J1 = J1 + 1
0217        307       CONTINUE
0218                  GO TO 350
0219              ELSE
0220              ENDIF
0221              SUM = Z1 + Z3
0222              IF (SUM .EQ. 1.0) THEN
0223                 DO 310 II = 1,NN1
0224                    ZZ(II) = PHASIN(II)
0225        310       CONTINUE
0226                  J1 = 1
0227                  NEND = NN1 +NN3
0228                  DO 311 II = NN1+1,NEND
0229                     ZZ(II) = PHAOUT(J1)
0230                     J1 = J1+1
0231        311       CONTINUE
0232                  GO TO 350
0233              ELSE
0234              ENDIF
0235
0236        C     AT THIS POINT A FULL TRAPEZOID MUST EXIST.
0237
0238              DO 315 II=1,NN1
0239                 ZZ(II)=PHASIN(II)
0240        315   CONTINUE
0241              J1 = 1
0242              NEND = NN1 +NN2
0243              DO 316 II = NN1+1,NEND
0244                 ZZ(II) = PHACON(J1)
0245                 J1 = J1+1
0246        316   CONTINUE
0247              J1 = 1
0248              DO 317 II=NEND+1,N2
0249                 ZZ(II)=PHAOUT(J1)
0250                 J1 = J1+1
0251        317   CONTINUE
0252
0253        C     SHIFT THE STRUCTURE TO THE STARTING POSITION.
0254
0255        350   CONTINUE
0256
0257        C     IF COST ELEMENT IS OF CONSTANT COST TYPE PUT IN
                  MCON(20,100)
0258        C     NOTE: 20 COST ELEMENTS BY 100 YEARS FOR THE SYSTEM.
0259
0260              IF (CE(I,1) .EQ. 1.0) THEN
0261                 JSTRT = IFIX(START)
0262                 DO 355 II = 1,N2
0263                    JSTRT = JSTRT + 1
0264                    MCON(I,JSTRT) = ZZ(II)
0265        355       CONTINUE
0266              ELSE
```

```
0267           ENDIF
0268
0269     C     IF COST ELEMENT IS OF RANDOM COST TYPE PUT IN
               MRAND(20,100)
0270     C     NOTE: 20 COST ELEMENT BY 100 YEARS FOR THE SYSTEM.
0271
0272           IF (CE(I,1) .EQ. 2.0 .OR. CE(I,1) .EQ. 3.0) THEN
0273              JSTRT = IFIX(START)
0274              DO 360 II = 1,N2
0275                 JSTRT = JSTRT + 1
0276                 MRAND(I,JSTRT) = ZZ(II)
0277        360   CONTINUE
0278           ELSE
0279           ENDIF
0280        500 CONTINUE
0281
0282     C     ACCUMULATE INTO COSTC(100) AND COSTR(100)
0283
0284           DO 510 J = 1,NYEARS
0285              X = 0.0
0286              Y = 0.0
0287              DO 505 I = 1,NCOST
0288                 X = X + MCON(I,J)
0289                 Y = Y + MRAND(I,J)
0290        505   CONTINUE
0291              COSTC(J) = X
0292              COSTR(J) = Y
0293        510 CONTINUE
0294           RETURN
0295           END
```

PROGRAM SECTIONS

ENTRY POINTS

Address	Type	Name	References
0-00000000		TRAP	7#

VARIABLES

Address	Type	Name	References			
**	R*4	A1	53=	57	59	70(2)
			71	72	76	77
			84	90	93	100
**	R*4	B	77=	100	147=	164
**	R*4	C1	54=	57	60	70
			71(2)	72	131	135
			136			
2-000084E0	R*4	D1	55=	57	61	70

			71	72(2)	146	147
			148	154	157	164
**	R*4	DD	99=	100	103=	104
			163=	164	167=	168
2-000084D8	I*4	I	28=	30	31	33=
			34	35	44=	51
			52	53	54	55
			56	260	264	272(2)
			276	287=	288	289
**	I*4	II	98=	99	100	102=
			103	104(2)	108=	109(2)
			110	116=	117(2)	123=
			124(2)	137=	138	162=
			163	164	166=	167
			168(2)	172=	173(2)	174
			177=	178(2)	185=	186
			197=	198	201=	202(2)
			209=	210(2)	214=	215
			223=	224(2)	228=	229
			238=	239(2)	243=	244
			248=	249	262=	264
			274=	276		
2-000084DC	I*4	J	29=	30	31	284=
			288	289	291	292
**	I*4	J1	184=	186	187(2)=	212=
			215	216(2)=	226=	229
			230(2)=	241=	244	245(2)=
			247=	249	250(2)=	
AP-00000014@	I*4	JJ	7	50	52	
**	I*4	JSTRT	261=	263(2)=	264	273=
			275(2)=	276		
**	I*4	N1	84=	98	102	107
			108	112	116	123
			136=	137	148=	162
			166	171	172	176
			177	184	185	
2-000084E8	I*4	N2	58=	248	262	274
AP-0000000C@	I*4	NCOST	7	8	28	44
			287			
**	I*4	NEND	213=	214	227=	228
			242=	243	248	
2-000084EC	I*4	NN1	59=	201	209	213
			214	223	227	228
			238	242	243	
2-000084F0	I*4	NN2	60=	213	242	
2-000084F4	I*4	NN3	61=	227		
AP-00000008@	I*4	NRV	7	8		
AP-00000018@	I*4	NYEARS	7	284		
2-000084E4	R*4	START	56=	261	273	
**	R*4	SUM	206=	208	221=	222
**	R*4	TT	57=	58		
2-000084D0	R*8	VALUE	1(	52=	91	94

			124	135	155	158
			186			
**	R*8	X	11	106=	109	110(2)=
			112	135=	138	170=
			173	174(2)=	176	285=
			288(2)=	291		
**	R*8	Y	11	286=	289(2)=	292
2-000084F8	R*4	Z1	70=	90	93	94
			117	200	206	221
2-000084FC	R*4	Z2	71=	135	206	
2-00008500	R*4	Z3	72=	154	157	158
			178	221		

ARRAYS

Address	Type	Name	Bytes	Dimensions	References
AP-00000004@	R*4	A	**	(*, *)	7 8
					52
AP-00000010@	R*4	CE	1600	(20, 20)	7 8
					51 53
					54 55
					56 260
					272(2)
3-00000000	R*4	COSTC	400	(100)	8 18
					34= 291=
3-00000190	R*4	COSTR	400	(100)	8 18
					35= 292=
2-00007E90	R*4	DEL	400	(100)	8 104=
					106 109(2)=
					110 112=
					117 168=
					170 173(2)=
					174 176=
					178
2-00000000	R*8	MCON	16000	(20, 100)	12 30=
					264= 288
2-00003E80	R*8	MRAND	16000	(20, 100)	12 31=
					276= 289
2-000081B0	R*4	PHACON	400	(100)	8 138=
					215 244
2-00008340	R*4	PHAOUT	400	(100)	8 155=
					158= 186=
					229 249
2-00008020	R*4	PHASIN	400	(100)	8 91=
					94= 124=
					202 210
					224 239
2-00007D00	R*4	ZZ	400	(100)	8 100=
					104 117=
					124 164

```
                                                    168         178=
                                                    186         198=
                                                    202=        210=
                                                    215=        224=
                                                    229=        239=
                                                    244=        249=
                                                    264         276


0001
0002          SUBROUTINE CER (J, A, NRV, NCOST, B, AAA,
              XSTAR,AA,X,LASTSET,
0003         1numcer,super)
0004          DIMENSION A(NRV,NCOST),B(20),XSTAR(60),X(NRV,20)
0005         1,VARCOV(20,20),bort(20),super(20,505)
0006          DOUBLE PRECISION VAR, RINTER, XLOW, XHIGH, T, TT,
              TTT,XTYPE,xmode
0007          COMMON /AA/ Z(20)
0008          PARAMETER(NRR=750)
0009          REAL R(NRR),RR(NRR),SCALAR(NRR)
0010          INTEGER NR,ISEED
0011          LOGICAL lastset
0012          EXTERNAL RNNOR,RNSET
0013
0014    C     B(AAA) HOLDS UP TO 20 TOTAL PARAMETERS IN THE CER.
0015    C     XSTAR(AA) HOLDS THE LOW,HIGH AND TYPE BETA PDF FOR
0016    C     EACH COST DRIVER. IF THE LOW AND HIGH VALUE ARE THE
              SAME
0017    C     FOR ANY COST DRIVER THEN IT IS ASSUMED THAT COST
              DRIVER IS
0018    C     THE SAME FOR ALL NRV ITERATIONS(ie. A CONSTANT).
0019    C     RINTER= THE RINTERCEPT OF THE CER IF ONE IS
              PRESENT.
0020    C     NOTE: AA= 3*AAA
0021    C     IF A CER CONTAINS MORE THAN 20 PARAMETERS THEN
0022    C     VARCOV(20,20) MUST BE INCREASED AS MUST B(AAA),
0023    C     XSTAR(AA) AND X(NRV,AAA) IN THE MAIN PROGRAM.
0024
0025    C     CLEAR THE ARRAYS.
0026
0027          N = IFIX(AAA)
0028          NN = IFIX(AA)
0029          IF (lastset .EQ. .FALSE.) THEN
0030          var = 0.0
0031          DO 2 I = 1, N
0032             DO 3 II = 1, N
0033                varcov(I,II) = 0.0
0034        3    continue
0035        2 continue
0036          DO 5 I = 1,N
0037             B(I) = 0.0
```

```
0038          5 CONTINUE
0039            DO 6 I = 1,NRV
0040            DO 6 II = 1,N
0041               X(I,II) = 0.0
0042          6 CONTINUE
0043            DO 10 I = 1,NN
0044               XSTAR(I) = 0.0
0045         10 CONTINUE
0046
0047            print*
0048            print*,'Enter the estimated variance (s-squared): '
0049            READ*, VAR
0050
0051     C      READ VARIANCE COVARIANCE MATRIX.
0052     C      NOTE: VARIANCE OF THE INCERCEPT OCCUPIES POSITION
                (1,1)
0053     C      IF AN RINTERCEPT IS PRESENT.
0054     C      VARCOV = VARIANCE COVARIANCE MATRIX OF THE CER.
0055
0056            print*
0057            print*,'Enter the VARIANCE/COVARIANCE matrix'
0058            print*
0059            DO 12 I=1,N
0060            DO 99 II=1,N
0061               print 98,I,II
0062         98    FORMAT(' Enter value (',I2,',',I2,')')
0063               read*, VARCOV(I,II)
0064         99 continue
0065         12 CONTINUE
0066
0067     C      READ THE VALUE OF THE RINTERCEPT IF PRESENT THEN
                THE VALUES OF
0068     C      THE SLOPE PARAMETERS. THE VECTOR B(I) WILL HOLD THE
                VALUES OF
0069     C      ALL OF THE PARAMETERS
0070
0071            IF (Z(11) .EQ. 1.0) THEN
0072               print*
0073               print*,'Enter the value of the Intercept: '
0074               READ*, RINTER
0075               B(1) = RINTER
0076               print*
0077               print*,'Enter the value of the slope
                   parameter(s).'
0078               print*
0079               DO 15 I = 2,N
0080                  print 97,I-1
0081         97       FORMAT(' Enter slope parameter ',I2)
0082                  READ*, B(I)
0083         15    continue
0084            ELSE
0085               print*
```

```
0086              print*,'Enter the value of the slope
                  parameter(s).'
0087              print*
0088              DO 20 I = 1,N
0089                 print 96,I
0090        96       FORMAT(' Enter slope parameter ',I2)
0091                 READ*, B(I)
0092        20    CONTINUE
0093           ENDIF
0094
0095     C     N IS THE NUMBER OF PARAMETERS IN THE CER, INCLUDING
               RINTERCEPT
0096     C     IF ONE IS PRESENT.
0097     C     NN IS 3*N.
0098
0099     C     READ THE VALUES OF XSTAR.
0100     C     NOTE IF AN RINTERCEPT IS PRESENT THE NUMBER  1
               MUST BE
0101     C     THE FIRST ELEMENT TO ENTER AND ITS HIGH AND LOW
               VALUE
0102     C     MUST = 1.0
0103
0104           print*
0105           print*,'COST DRIVER CARDS'
0106           print*
0107           print*,'Each cost driver is characterized by 3
               cards.'
0108        print*,'1st Card = Low value of Beta or Triangular pdf'
0109       print*,'2nd Card = High value of Beta or Triangular pdf'
0110           print*,'3rd Card = Type of Beta pdf or Mode of
               Triang pdf.'
0111           print*,'(If the cost driver is NOT a random
               variable (ie it'
0112           print*,'is a constant) then set the 1st and 2nd
               cards (the'
0113           print*,'LOW and HIGH values of the Beta or Triag
               distribution)'
0114           print*,'equal to the SAME value.  If an intercept
               term IS'
0115           print*,'present, then set the first 3 cards equal
               to 1.0.)'
0116           print*
0117           k = 1
0118           DO 40 I = 1,N
0119              print*
0120
0121     c        Begin the loop to obtain the cost driver data.
0122
0123              print 30, I
0124        30    format(' Do you want cost driver #',I2,' to be
               modeled')
0125              print*,'by a Beta or Triangular pdf?'
```

```
0126              print*,'Enter:'
0127              print*,'1 for BETA pdf'
0128              print*,'2 for TRIANGULAR pdf'
0129              read*, bort(i)
0130              min = 1
0131              max = 2
0132              asked = bort(i)
0133              call ans(min,max,asked,found)
0134              bort(i) = asked
0135              print*
0136              if (bort(i) .eq. 1.0) then
0137                 if (Z(11) .eq. 1.0 .and. i .eq. 1) then
0138                  print*,'Enter the INTERCEPT values NOW, ie'
0139                    print*,'THE NEXT 3 RESPONSES MUST BE 1!'
0140                 else
0141                 endif
0142              print*,'Enter the LOW value of the Beta pdf for'
0143              print 95,I
0144        95    FORMAT(' cost driver #',I2,': (for example, if
                  the')
0145          print*,'LOW number of components to be evaluated is'
0146              print*,'50, enter 50.)'
0147              READ*, XSTAR(k)
0148              print*,'Enter the HIGH value of the Beta pdf
                  for'
0149              print 74,I
0150        74    FORMAT(' cost driver #',I2,':')
0151              read*, XSTAR(k+1)
0152              IF (k .GT. 1 .AND. Z(12) .EQ. 1.0) THEN
0153                 XSTAR(k) = LOG(XSTAR(k))
0154                 XSTAR(k+1) = LOG(XSTAR(k+1))
0155              ELSE
0156              ENDIF
0157              print*,'Enter the TYPE of Beta pdf for cost'
0158              print 75,I
0159        75    FORMAT(' driver #',I2,': (Enter 1.0 -- 9.0)')
0160              read*, XSTAR(k+2)
0161              min = 1
0162              max = 9
0163              asked = xstar(k+2)
0164              CALL ANS(min,max,asked,found)
0165              xstar(k+2) = asked
0166              k = k + 3
0167              print*
0168              else if (bort(i) .eq. 2.0) then
0169                 if (Z(11) .eq. 1.0 .and. i .eq. 1) then
0170                    print*,'Enter the INTERCEPT values NOW.'
0171                 else
0172                 endif
0173            print*,'Enter the LOW value of the Triang pdf for'
0174                 print 31, I
0175        31       format(' cost driver #',I2,': (for example,
```

```
                if the')
0176            print*,'LOW number of components to be
                evaluated is'
0177            print*,'50, enter 50.)'
0178            read*, XSTAR(k)
0179            print*,'Enter the HIGH value of the Triang
                pdf for'
0180            print 32, I
0181       32   format(' cost driver #',I2,':')
0182            read*, XSTAR(k+1)
0183            print*,'Enter the MODE or BEST GUESS of the
                Triang'
0184            print 33, I
0185       33   format(' pdf for cost driver #',I2,':')
0186            read*, XSTAR(k+2)
0187            if (k .gt. 1 .and. Z(12) .eq. 1.0) then
0188               XSTAR(k) = LOG(XSTAR(k))
0189               XSTAR(k+1) = LOG(XSTAR(k+1))
0190               XSTAR(k+2) = LOG(XSTAR(k+2))
0191            else
0192            endif
0193            k = k + 3
0194            else
0195            endif
0196       40 CONTINUE
0197          call check(var,varcov,xstar,n,b,bort)
0198
0199    c     Write the estimated variance, var/cov matrix, slope
0200    c     parameters, and cost driver values to the file
              MOREINPT.DAT
0201
0202          write (02,200) var
0203      200 format (f31.16)
0204          do 201 I = 1,n
0205          do 201 II = 1,n
0206             write (02,202) varcov(I,II)
0207      202    format (f31.16)
0208      201 continue
0209          do 203 I = 1,n
0210             write (02,204) B(I)
0211      204    format (f31.16)
0212      203 continue
0213          k = 1
0214          do 205 I = 1,n
0215             write (02,206) XSTAR(k)
0216      206    format (f15.8)
0217             write (02,80) XSTAR(k+1)
0218       80    format (f15.8)
0219             write (02,81) XSTAR(k+2)
0220       81    format (f15.8)
0221             k = k + 3
0222      205 continue
```

```
0223          do 207 i = 1,n
0224             write(02,208) bort(i)
0225       208   format(f5.3)
0226       207 continue
0227          ELSE IF (lastset .EQ. .TRUE.) THEN
0228
0229     c    The data set entered on the previous run is being
              entered,
0230     c    therefore its values have to be read in from the
              file,
0231     c    MOREINPT.DAT.
0232
0233             iii = 2
0234             var = super(numcer,1)
0235             DO 137 I = 1, N
0236                DO 138 II = 1, N
0237                   varcov(i,ii) = super(numcer,iii)
0238                   iii = iii + 1
0239       138       continue
0240       137    continue
0241             IF (Z(11) .EQ. 1.0) THEN
0242                RINTER = super(numcer,iii)
0243                B(1) = RINTER
0244                iii = iii + 1
0245                DO 141 I = 2, N
0246                   b(i) = super(numcer,iii)
0247                   iii = iii + 1
0248       141       continue
0249             ELSE
0250                DO 143 I = 1, N
0251                   b(i) = super(numcer,iii)
0252                   iii = iii + 1
0253       143       continue
0254             ENDIF
0255             k = 1
0256             DO 145 I = 1, N
0257                XSTAR(k) = super(numcer,iii)
0258                XSTAR(k+1) = super(numcer,iii+1)
0259                XSTAR(k+2) = super(numcer,iii+2)
0260                k = k + 3
0261                iii = iii + 3
0262       145    continue
0263             do 147 i = 1,n
0264                bort(i) = super(numcer,iii)
0265                iii = iii + 1
0266       147    continue
0267          ELSE
0268          ENDIF
0269
0270     C    CHECK TO SEE IF THE COST DRIVER HAS THE SAME LOW
              AND
0271     C    HIGH VALUE. IF SO, THAT VALUE IS THE SAME FOR ALL
```

```
            NRV
0272    C   RUNS. ALL OTHER COST DRIVERS MUST PASS THROUGH THE
0273    C   RANDOM NUMBER GENERATOR FOR THE APPROPRIATE BETA
            PDF.
0274
0275            ITA=1
0276            ITB=2
0277            ITC=3
0278            DO 100 I = 1,N
0279             if (bort(i) .eq. 1.0) then
0280                IF (XSTAR(ITA) .EQ. XSTAR(ITB)) THEN
0281                 DO 50 II = 1,NRV
0282                    X(II,I) = XSTAR(ITA)
0283         50         CONTINUE
0284                ELSE
0285                 XLOW = XSTAR(ITA)
0286                 XHIGH = XSTAR(ITB)
0287                 XTYPE = XSTAR(ITC)
0288                 CALL BETA1(J,XLOW,XHIGH,XTYPE,NRV,R)
0289                 DO 60 II = 1,NRV
0290                    X(II,I) = R(II)
0291         60         CONTINUE
0292                 ENDIF
0293               ITA = ITA+3
0294            ITB = ITB+3
0295            ITC = ITC+3
0296            else if (bort(i) .eq. 2.0) then
0297            if (xstar(ita) .eq. xstar(itb)) then
0298               do 71 ii = 1,nrv
0299                  x(ii,i) = xstar(ita)
0300         71    continue
0301            else
0302            xlow = xstar(ita)
0303            xhigh = xstar(itb)
0304            xmode = xstar(itc)
0305            call triag1(j,xlow,xhigh,xmode,nrv,r)
0306            do 70 ii = 1,nrv
0307               x(ii,i) = r(ii)
0308         70 continue
0309            endif
0310            ita = ita + 3
0311            itb = itb + 3
0312            itc = itc + 3
0313            else
0314            endif
0315       100 CONTINUE
0316
0317    C   DRAW NRV RANDOM SAMPLES FROM NORMAL(0,1).
0318    C   STORE THEM IN VECTOR RR(NRV).
0319
0320            NR = NRV
0321            CALL RNSET(0)
```

```
0322            CALL RNNOR(NR,RR)
0323            IF (Z(13) .EQ. 1.0) THEN
0324               if (Z(20) .eq. 1.0) then
0325                  XLOW = Z(14)
0326                  XHIGH = Z(15)
0327                  XTYPE = Z(16)
0328                  IF (Z(14) .EQ. Z(15)) THEN
0329                     DO 105 KKK = 1,NRV
0330                        SCALAR(KKK) = Z(14)
0331        105           CONTINUE
0332                  ELSE
0333                     CALL BETA1(J,XLOW,XHIGH,XTYPE,NRV,SCALAR)
0334                  ENDIF
0335               else if (Z(20) .eq. 2.0) then
0336                  xlow = Z(14)
0337                  xhigh = Z(15)
0338                  xmode = Z(16)
0339                  if (Z(14) .eq. Z(15)) then
0340                     do 7 kkk = 1,nrv
0341                        scalar(kkk) = Z(14)
0342          7           continue
0343                  else
0344                     CALL TRIAG1(j,xlow,xhigh,xmode,nrv,scalar)
0345                  endif
0346               else
0347               endif
0348            ELSE
0349            ENDIF
0350            DO 300 I = 1,NRV
0351             DO 111 K = 1,N
0352             T = 0.0
0353             DO 110 L = 1,N
0354                T = T + (X(I,L)*VARCOV(L,K))
0355        110  CONTINUE
0356             R(K) = T
0357        111  CONTINUE
0358             T = 0.0
0359             DO 115 II = 1,N
0360                T = T + (R(II)*X(I,II))
0361        115  CONTINUE
0362
0363        C    T IS THE QUADRATIC FORM X'(VARCOV)X
0364
0365             T = VAR + T
0366             T = DSQRT(T)
0367             TT = RR(I)*T
0368             TTT = 0.0
0369             DO 120 II = 1,N
0370                TTT = TTT + (X(I,II)*B(II))
0371        120  CONTINUE
0372             T = TT + TTT
0373
```

```
0374     C     CHECK TO SEE IF THE DEPENDENT VARIABLE IS IN LOG
               FORM
0375     C     (IE. CER WAS ESTIMATED BY TAKING LOG OF BASE E. IF
0376     C     SO TAKE THE ANITLOG OF THE FORECAST.
0377
0378           IF (Z(12) .EQ. 1.0) THEN
0379              T = DEXP(T)
0380           ELSE
0381           ENDIF
0382
0383     C     T IS THE FORECASTED VALUE OF THE CER FOR RANDOM
0384     C     NUMBER # I
0385     C     STORE THE VALUE IN THE A(NRV,NCOST) MATRIX.
0386
0387           IF (Z(13) .EQ. 1.0) THEN
0388              A(I,J) = SCALAR(I)*T
0389           ELSE
0390              A(I,J) = T
0391           ENDIF
0392       300 CONTINUE
0393           RETURN
0394           END
```

PROGRAM SECTIONS

ENTRY POINTS

Address	Type	Name	References
0-00000000		CER	2#

VARIABLES

Address	Type	Name	References			
AP-00000020@	R*4	AA	2	28		
AP-00000018@	R*4	AAA	2	27		
2-00002A28	R*4	ASKED	132=	133A	134	163=
			164A	165		
2-00002A2C	R*4	FOUND	133A	164A		
2-000029F0	I*4	I	31=	33	36=	37
			39=	41	43=	44
			59=	61	63	79=
			80	82	88=	89
			91	118=	123	129
			132	134	136	137
			143	149	158	168
			169	174	180	184
			204=	206	209=	210
			214=	223=	224	235=
			237	245=	246	250=

			251 256= 263= 264
			278= 279 282 290
			296 299 307 350=
			354 360 367 370
			388(2) 390
**	I*4	II	32= 33 40= 41
			60= 61 63 205=
			206 236= 237 281=
			282 289= 290(2) 298=
			299 306= 307(2) 359=
			360(2) 369= 370(2)
**	I*4	III	233= 237 238(2)= 242
			244(2)= 246 247(2)= 251
			252(2)= 257 258 259
			261(2)= 264 265(2)=
**	I*4	ISEED	10
**	I*4	ITA	275= 280 282 285
			293(2)= 297 299 302
			310(2)=
**	I*4	ITB	276= 280 286 294(2)=
			297 303 311(2)=
2-00002A48	I*4	ITC	277= 287 295(2)= 304
			312(2)=
AP-00000004@	I*4	J	2 288A 305A 333A
			344A 388 390
**	I*4	K	117= 147 151 152
			153(2) 154(2) 160 163
			165 166(2)= 178 182
			186 187 188(2) 189(2)
			190(2) 193(2)= 213= 215
			217 219 221(2)= 255=
			257 258 259 260(2)=
			351= 354 356
**	I*4	KKK	329= 330 340= 341
**	I*4	L	353= 354(2)
AP-00000028@	L*4	LASTSET	2 11 29 227
2-00002A24	I*4	MAX	131= 133A 162= 164A
2-00002A20	I*4	MIN	130= 133A 161= 164A
2-000029EC	I*4	N	27= 31 32 36
			40 59 60 79
			88 118 197A 204
			205 209 214 223
			235 236 245 250
			256 263 278 351
			353 359 369
AP-00000010@	I*4	NCOST	2 4
**	I*4	NN	28= 43
2-000029E8	I*4	NR	10 320= 322A
AP-0000000C@	I*4	NRV	2 4(2) 39 281
			288A 289 298 305A
			306 320 329 333A
			340 344A 350

AP-0000002C@	I*4	NUMCER	2	234	237	242
			246	251	257	258
			259	264		
2-000029C0	R*8	RINTER	6	74=	75	242=
			24			
**	R*8	T	6	352=	354(2)=	356
			358=	360(2)=	365(2)	366(2)=
			367	372=	379(2)=	388
			390			
**	R*8	TT	6	367=	372	
**	R*8	TTT	6	368=	370(2)=	372
2-000029B8	R*8	VAR	6	30=	49=	197A
			202	234=	365	
2-000029D0	R*8	XHIGH	6	286=	288A	303=
			305A	326=	333A	337=
			344A			
2-000029C8	R*8	XLOW	6	285=	288A	302=
			305A	325=	333A	336=
			344A			
2-000029E0	R*8	XMODE	6	304=	305A	338=
			344A			
2-000029D8	R*8	XTYPE	6	287=	288A	327=
			333A			

ARRAYS

Address	Type	Name	Bytes	Dimensions	References	
AP-00000008@	R*4	A	**	(*, *)	2	4
					388=	390=
AP-00000014@	R*4	B	80	(20)	2	4
					37=	75=
					82=	91=
					197A	210
					243=	246=
					251=	370
2-00000640	R*4	BORT	80	(20)	4	129=
					132	134=
					136	168
					197A	224
					264=	279
					296	
2-00000690	R*4	R	3000	(750)	9	288A
					290	305A
					307	356=
					360	
2-00001248	R*4	RR	3000	(750)	9	322A
					367	
2-00001E00	R*4	SCALAR	3000	(750)	9	330=
					333A	341=
					344A	388
AP-00000030@	R*4	SUPER	40400	(20, 505)	2	4

					234 237
					242 246
					251 257
					258 259
					264
2-00000000	R*4	VARCOV	1600	(20, 20)	4 33=
					63= 197A
					206 237=
					354
AP-00000024@	R*4	X	**	(*, 20)	2 4
					41= 282=
					290= 299=
					307= 354
					360 370
AP-0000001C@	R*4	XSTAR	240	(60)	2 4
					44= 147=
					151= 153(2)=
					154(2)= 160=
					163 165=
					178= 182=
					186= 188(2)=
					189(2)= 190(2)=
					197A 215
					217 219
					257= 258=
					259= 280(2)
					282 285
					286 287
					297(2) 299
					302 303
					304
3-00000000	R*4	Z	80	(20)	7 71
					137 152
					169 187
					241 323
					324 325
					326 327
					328(2) 330
					335 336
					337 338
					339(2) 341
					378 387

PARAMETER CONSTANTS

Type	Name	References
I*4	NRR	8# 9(3)

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References

```
        ANS                     133       164
        BETA1                   288       333
        CHECK                   197
  R*4   MTH$ALOG                153       154       188       189
                                190
  R*8   MTH$DEXP                379
  R*8   MTH$DSQRT               366
        RNNOR                    12       322
        RNSET                    12       321
        TRIAG1                  305       344


0001
0002
0003          subroutine check(var,varcov,xstar,n,b,bort)
0004          dimension xstar(60), varcov(20,20), B(20), bort(20)
0005          double precision var
0006          logical done
0007          common /aa/ Z(20)
0008
0009          call clrscr
0010          x = 0.0
0011          print*,'The estimated variance (s-squared),
              covariance matrix,'
0012          print*,'slope parameters, and cost driver cards
              have just been'
0013          print*,'entered.'
0014          print*,'Do you wish to review or edit any of
              these?'
0015          print*
0016          print*,'Enter:'
0017          print*,'1 for YES'
0018          print*,'2 for NO'
0019          read*, yorn
0020          min = 1
0021          max = 2
0022          asked = yorn
0023          call ans(min,max,asked,found)
0024          yorn = asked
0025          if (yorn .eq. 1.0) then
0026             print*
0027          print*,'Do not be surprised if some round-off has'
0028          print*,'occurred -- this will happen at times due'
0029          print*,'the size (large or small) of the
              variables.'
0030             print*
0031             print 80, var
0032       80    format('  1. variance (s-squared) = ',F31.16)
0033             print*
0034             k = 2
0035             do 81 I = 1,n
```

```
0036              do 81 II = 1,n
0037                 print 82, k,I,II,varcov(I,II)
0038        82       format(I3,'. varcov (',I2,',',I2,') =
                     ',F31.16)
0039                 k = k + 1
0040        81    continue
0041              print*
0042              print*,'Press RETURN to continue . . .'
0043              read*
0044              k = n*n + 2
0045              if (Z(11) .eq. 1.0) then
0046                 print 83, k, B(1)
0047        83       format(I3,'. intercept =      ',F31.16)
0048                 do 84 I = 2,n
0049                    k = k + 1
0050                    print 85, k, I-1, B(I)
0051        85          format(I3,'. slope param ',I2,' =
                        ',F31.16)
0052        84       continue
0053              else
0054                 do 87 I = 1,n
0055                    print 86, k, I, B(I)
0056        86          format(I3,'. slope param ',I2,' =
                        ',F31.16)
0057                    k = k + 1
0058        87       continue
0059              endif
0060              print*
0061              k = ((n*n) + n + 2)
0062              kk = 1
0063              do 88 I = 1,n
0064                 print 89, I
0065        89       format(' Cost Driver for Element #',I2,':')
0066                 print 90, k,XSTAR(kk)
0067        90       format(I3,'.  LOW value = ',F15.8)
0068                 print 91, k+1,XSTAR(kk+1)
0069        91       format(I3,'. HIGH value = ',F15.8)
0070                 if (bort(i) .eq. 1.0) then
0071                    print 92, k+2,XSTAR(kk+2)
0072        92          format(I3,'.  Beta TYPE = ',F15.8)
0073                 else if (bort(i) .eq. 2.0) then
0074                    print 40, k+2,XSTAR(kk+2)
0075        40          format(I3,'.  MODE value = ',f15.8)
0076                 else
0077                 endif
0078                 print*
0079                 k = k + 3
0080                 kk = kk + 3
0081        88    continue
0082              print*
0083              print*,'Do you want to edit any of these?'
0084              print*,'Enter:'
```

```
0085              print*,'1 for YES'
0086              print*,'2 for NO'
0087              read*, yorn
0088              asked = yorn
0089              call ans(min,max,asked,found)
0090              yorn = asked
0091              if (yorn .eq. 1) then
0092                 print*
0093             print*,'How many values would you like to edit?'
0094                 print*,'(Enter 0 to exit)'
0095                 read*, numchanges
0096                 done = .false.
0097       888       if (done .eq. .false.) then
0098                 if (numchanges .gt. 0) then
0099                    do 93 iii = 1,numchanges
0100        94             print*
0101                       print*,'Enter the # of the value to be'
0102                       print*,'changed (from the left hand
                           column)'
0103                       read*, jjj
0104                       jcount = 1
0105                       ic = 1
0106                       kcount = n*n
0107                       if (jjj .lt. 1 .or. jjj .gt. (n*n + 4*n
                           + 1)) then
0108                          go to 94
0109                       else if (jjj .eq. 1) then
0110                             print*
0111                             print*,'Enter the NEW variance
                                 (s-squared):'
0112                             read*, var
0113                       else if (jjj .gt. 1 .and. jjj .le.
                           (n*n+1)) then
0114        25                   if (ic .le. kcount) then
0115                                isum = ic*n
0116                                if ((jjj-1) .le. isum) then
0117                                   I = jcount
0118                                   II = (jjj - 1 -
                                       (jcount-1)*n)
0119                                   ic = n*n+1
0120                                   go to 25
0121                                else
0122                                   ic = ic + 1
0123                                   isum = ic*n
0124                                   jcount = jcount + 1
0125                                   go to 25
0126                                endif
0127                             else
0128                             endif
0129                             print 8, I,II
0130         8             format(' Enter the NEW
                           VARCOV(',I2,',',I2,')')
```

```
0131                          read*, varcov(I,II)
0132                   else if (jjj .gt. (n*n+1) .and. jjj .le.
0133         1(n*n+n+1)) then
0134                          icount = jjj - (n*n+1)
0135                          if (Z(11) .eq. 1.0) then
0136                             if (icount .eq. 1) then
0137                                print*
0138                                print*,'Enter the NEW
                                   INTERCEPT:'
0139                                read*, B(1)
0140                                rinter = B(1)
0141                             else
0142                                print*
0143                                print 15, icount-1
0144       15                       format(' Enter the NEW
                                   Slope Param',I3)
0145                                read*, B(icount)
0146                             endif
0147                          else
0148                             print*
0149                             print 16, icount
0150       16                    format(' Enter the NEW Slope
                                   Param',I3)
0151                             read*, B(icount)
0152                          endif
0153                       else if (jjj .gt. (n*n + n + 1))
                           then
0154                          icount = jjj - (n*n+n+1)
0155                          print*
0156                          print 33, icount
0157       33                 format(' Enter the NEW XSTAR
                              (',I2,') term:')
0158                          read*, XSTAR(icount)
0159                       else
0160                       endif
0161       93           continue
0162                 else
0163                    done = .true.
0164                 endif
0165                 print*
0166                 print*,'Do you wish to make further changes?'
0167                 print*,'Enter:'
0168                 print*,'1 for YES'
0169                 print*,'2 for NO'
0170                 read*, yorn
0171                 asked = yorn
0172                 call ans(min, max, asked, found)
0173                 yorn = asked
0174                 if (yorn .eq. 1.0) then
0175                    go to  888
0176                 else
0177                    done = .true.
```

```
0178                endif
0179                else
0180                endif
0181             else
0182             endif
0183          else
0184          endif
0185          return
0186          end
```

ENTRY POINTS

Address	Type	Name	References
0-00000000		CHECK	3#

VARIABLES

Address	Type	Name	References
2-00000010	R*4	ASKED	22= 23A 24 88= 89A 90 171= 172A 173
2-00000000	L*4	DONE	6 96= 97 163= 177=
2-00000014	R*4	FOUND	23A 89A 172A
**	I*4	I	35= 37(2) 48= 50(2) 54= 55(2) 63= 64 70 73 117= 129 131
**	I*4	IC	105= 114 115 119= 122(2)= 123
**	I*4	ICOUNT	134= 136 143 145 149 151 154= 156 158
2-00000018	I*4	II	36= 37(2) 118= 129 131
**	I*4	III	99=
**	I*4	ISUM	115= 116 123=
**	I*4	JCOUNT	104= 117 118 124(2)=
2-00000020	I*4	JJJ	103= 107(2) 109 113(2) 116 118 132(2) 134 153 154
**	I*4	K	34= 37 39(2)= 44= 46 49(2)= 50 55 57(2)= 61= 66 68 71 74 79(2)=
**	I*4	KCOUNT	106= 114
**	I*4	KK	62= 66 68 71 74 80(2)=
2-0000000C	I*4	MAX	21= 23A 89A 172A 133

2-00000008	I*4	MIN	20=	23A	89A	172A
AP-00000010@	I*4	N	3	35	36	44(2)
			48	54	61(3)	63
			106(2)	107(3)	113(2)	115
			118	119(2)	123	132(5)
			134(2)	153(3)	154(3)	
2-0000001C	I*4	NUMCHANGES	95=	98	99	
**	R*4	RINTER	140=			
AP-00000004@	R*8	VAR	3	5	31	112=
**	R*4	X	10=			
2-00000004	R*4	YORN	19=	22	24=	25
			87=	88	90=	91
			170=	171	173=	174

ARRAYS

Address	Type	Name	Bytes	Dimensions	References	
AP-00000014@	R*4	B	80	(20)	3	4
					46	50
					55	139=
					140	145=
					151=	
AP-00000018@	R*4	BORT	80	(20)	3	4
					70	73
AP-00000008@	R*4	VARCOV	1600	(20, 20)	3	4
					37	131=
AP-0000000C@	R*4	XSTAR	240	(60)	3	4
					66	68
					71	74
					158=	
3-00000000	R*4	Z	80	(20)	7	45
					135	

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References		
	ANS	23	89	172
	CLRSCR	9		

```
0001
0002
0003           SUBROUTINE BETA1(J,XLOW,XHIGH,XTYPE,NRV,R)
0004           INTEGER ISEED
0005           PARAMETER (NR=750)
0006           REAL PIN,QIN,R(NR)
0007           EXTERNAL RNBET,RNSET
0008
0009     C     CHECK FOR TYPE OF BETA DISTRIBUTION.
0010
```

```
0011          IF (XTYPE .EQ. 1.0) THEN
0012             PIN = 2.5
0013             QIN = 1.5
0014          ELSE IF (XTYPE .EQ. 2.0) THEN
0015             PIN = 2.35
0016             QIN = 2.35
0017          ELSE IF (XTYPE .EQ. 3.0) THEN
0018             PIN = 1.5
0019             QIN = 2.5
0020          ELSE IF (XTYPE .EQ. 4.0) THEN
0021             PIN = 4.0
0022             QIN = 2.0
0023          ELSE IF (XTYPE .EQ. 5.0) THEN
0024             PIN = 3.75
0025             QIN = 3.75
0026          ELSE IF (XTYPE .EQ. 6.0) THEN
0027             PIN = 2.0
0028             QIN = 4.0
0029          ELSE IF (XTYPE .EQ. 7.0) THEN
0030             PIN = 5.5
0031             QIN = 2.5
0032          ELSE IF (XTYPE .EQ. 8.0) THEN
0033             PIN = 5.0
0034             QIN = 5.0
0035          ELSE IF (XTYPE .EQ. 9.0) THEN
0036             PIN = 2.5
0037             QIN = 5.5
0038          ELSE
0039          ENDIF
0040
0041          CALL RNSET(0)
0042          CALL RNBET(NR,PIN,QIN,R)
0043          DO 10 I = 1,NR
0044             R(I) = XLOW+((XHIGH-XLOW)*R(I))
0045       10 CONTINUE
0046          RETURN
0047          END
```

PROGRAM SECTIONS

ENTRY POINTS

Address	Type	Name	References
0-00000000		BETA1	3#

VARIABLES

Address	Type	Name	References
**	I*4	I	43= 44(2)

```
     **       I*4   ISEED          4
AP-00000004@  I*4   J             3
AP-00000014@  I*4   NRV           3
 2-00000000   R*4   PIN           6        12=       15=       18=
                                  21=      24=       27=       30=
                                  33=      36=       42A
 2-00000004   R*4   QIN           6        13=       16=       19=
                                  22=      25=       28=       31=
                                  34=      37=       42A
AP-0000000C@  R*4   XHIGH         3        44
AP-00000008@  R*4   XLOW          3        44(2)
AP-00000010@  R*4   XTYPE         3        11        14        17
                                  20       23        26        29
                                  32       35
```

ARRAYS

```
    Address  Type  Name    Bytes  Dimensions   References

AP-00000018@ R*4   R       3000  (750)          3           6    42A
                                                44(2)=
```

PARAMETER CONSTANTS

```
  Type  Name                          References

   I*4  NR                            5#        6         42        43
```

FUNCTIONS AND SUBROUTINES REFERENCED

```
  Type  Name                                    References

        RNBET                                     7         42
        RNSET                                     7         41
```

```
+-----------------------------------------------+
|          KEY TO REFERENCE FLAGS               |
|   =   - Value Modified                        |
|   #   - Defining Reference                    |
|   A   - Actual Argument, possibly modified    |
|   D   - Data Initialization                   |
|  (n)  - Number of occurrences on line         |
+-----------------------------------------------+
```

COMMAND QUALIFIERS

```
  FORTRAN/LIS/CROSS TT.FOR

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /SHOW=(NODICTIONARY,NOINCLUDE,MAP,NOPREPROCESSOR,SINGLE)
```

```
  /STANDARD=(NOSEMANTIC,NOSOURCE_FORM,NOSYNTAX)
  /WARNINGS=(NODECLARATIONS,GENERAL,NOULTRIX,NOVAXELN)
  /CONTINUATIONS=19  /CROSS_REFERENCE  /NOD_LINES
/NOEXTEND_SOURCE
  /F77  /NOG_FLOATING  /I4  /NOMACHINE_CODE  /OPTIMIZE
/NOPARALLEL
  /NOANALYSIS_DATA
  /NODIAGNOSTICS
  /LIST=GOR91M:[JGIBSON]TT.LIS;37
  /OBJECT=GOR91M:[JGIBSON]TT.OBJ;1

COMPILATION STATISTICS

  Run Time:            9.48 seconds
  Elapsed Time:        11.02 seconds
  Page Faults:         1936
  Dynamic Memory:      1494 pages
```

Bibliography

1. Department of the Air Force. Cost Estimating Procedures. AFSCM 173-1. Washington: HQ AFSC, 17 April 1972.

2. Office of the Assistant Secretary of Defense. Cost and Operational Effectiveness Analysis (COEA) Guidelines. DoD Directive 5000.1. Washington: Government Printing Office, February 1990.

3. Pritsker, A. Alan B. Introduction to Simulation and SLAM II, (third edition). New York: Systems Publishing Corporation, 1986.

4. Seldon, Robert M. Life Cycle Costing: A Better Method of Government Procurement. Boulder, CO: Westview Press, 1979.

5. Sobol, I. M. The Monte Carlo Method. Moscow: Mir Publishers, 1975.

6. Sumner, Capt David L. An Interactive Life Cycle Cost Forecasting Tool. MS thesis, AFIT/GOR/ENS/90M-17. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH.

7. Tadikamalla, Pandu R. "Computer Generation of Gamma Random Variables II," Communications of the ACM, 21:925-928 (November 1978).