NBSIR 83-2749

Programmer's Manual for the Fire Safety Evaluation System Cost Minimizer Computer Program

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Applied Mathematics
Washington, DC 20234

July 1983

Sponsored by
U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Fire Research
shington, DC 20234

partment of Health and Human Services
shington, DC 20201

NBSIR 83-2749

PROGRAMMER'S MANUAL FOR THE FIRE SAFETY EVALUATION SYSTEM COST MINIMIZER COMPUTER PROGRAM

Robert E. Chapman and William G. Hall

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Applied Mathematics
Washington, DC 20234

July 1983

Sponsored by:
U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Fire Research
Washington, DC 20234

and

Department of Health and Human Services
Washington, DC 20201

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

PREFACE

This research was conducted under the sponsorship of the Center for Fire Research and the Department of Health and Human Services by the Operations Research Division, Center for Applied Mathematics, National Engineering Laboratory, National Bureau of Standards.

This report is a product of the Fire and Life Safety Program. This program is a joint Department of Health and Human Services (HHS) and National Bureau of Standards (NBS) effort directed at the development of rational, technically sound solutions to fire safety problems in health care facilities. In addition to the types of work described in this report, the joint HHS/NBS program has produced products in the areas of decision analysis, fire and smoke detection, smoke movement and control, automatic extinguishment, and behavior of institutional and other populations in fire situations.

This study serves as a Programmer's Manual for applying and interpreting the results of the Fire Safety Evaluation System Cost Minimizer (FSESCM) computer program. The mathematical optimization techniques which are the core concept of the FSESCM are used to identify the least-cost means of upgrading health care facilities to compliance with the Life Safety Code. The program uses the "optimal" solution as a reference point from which 10 to 20 compliance strategies based on design considerations are generated. The computer program is intended to be used as a management tool to facilitate the design selection process by providing both information on relative costs and a chance to match common compliance strategies across fire zones.

ABSTRACT

The Fire Safety Evaluation System Cost Minimizer (FSESCM) computer program integrates engineering and economic considerations with a linear programming algorithm which permits the least-cost means of upgrading health care facilities to compliance with the Life Safety Code to be identified. A mathematical discussion of the application problem is used to introduce the basic philosophy behind the computer program. Each routine is described with emphasis on such topics as: (1) purpose; (2) calling sequence; (3) common blocks used; and (4) reports produced. A series of descriptive tables and a glossary are used to define all reports and variables. A discussion of test results and provisions for updating or modifying the source code are also given. The program is written in FORTRAN and complies with the ANSI X3.9-1978 software standard.

Keywords: Building codes; building economics; economic analysis; fire safety; health care facilities; hospitals; life safety; mathematical programming; nursing homes; renovation.

Table of Contents

List of Exhibits

List of Figures

List of Tables

1. INTRODUCTION

1.1 BACKGROUND

The identification of cost-effective levels of fire safety in health care facilities is a major concern to hospital administrators, fire safety engineers and public policy makers. Rising construction and operating costs coupled with more stringent building codes and continuing advances in medical and building technology have complicated the issue, forcing health care facility administrators to carefully assess the alternative means through which they can design, construct or update their facilities.

The National Fire Protection Association (NFPA) has long been recognized as a leader in the development of voluntary codes which establish acceptable fire safety levels. The Life Safety Code[1] is a widely used guide for identifying the minimum acceptable level of fire safety in buildings. Although the code may be thought of as prescriptive since it prescribes fixed solutions for life safety in designated occupancies, performance concepts can be explicitly introduced through a provision which allows for equivalent solutions.

In light of this provision, the National Bureau of Standards' Center for Fire Research, through support from the Department of Health and Human Services, has developed a system, the Fire Safety Evaluation System (FSES), for determining how combinations of several widely accepted fire safety systems could be used to provide a level of safety equivalent to that required in the Life Safety Code.[2] The FSES equivalency methodology which emerged from this effort is particularly attractive since it lends itself to computer optimization techniques. Such optimization techniques should result in improved fire safety in health care facilities because they will resolve many of the differences of opinion surrounding the cost impacts of fire safety in health care facilities in general and the Life Safety Code in particular.

1.2 PURPOSE

The purpose of this report is to present a programmer-oriented description of the economic and engineering considerations that went into the development of a linear programming algorithm which permits the least cost means of achieving compliance to the Life Safety Code to be identified. The Fire Safety Evaluation System Cost Minimizer (FSESCM) computer program discussed in this report is particularly useful because it is based on the equivalency methodology developed by the Center for Fire Research. Since the NFPA has

---

[1] National Fire Protection Association, Code for Safety to Life from Fire in Buildings and Structures, NFPA 101-1981, Quincy, Mass., 1981.

[2] H. E. Nelson and A. J. Shibe, A System for Fire Safety Evaluation of Health Care Facilities, National Bureau of Standards, NBSIR 78-1555, Washington, D.C., 1980.

adopted this equivalency methodology into the Life Safety Code, any solutions from the computer program will be in compliance with the Life Safety Code. Furthermore, since each of the parameters used in the equivalency methodology has a unique value which corresponds to prescriptive compliance, it is possible to quantify the cost savings attributable to a performance based approach, or equivalency methodology, over that of prescriptive compliance. Although the procedure is valid for both new and existing facilities, it is anticipated that its primary use will be in identifying alternative courses of action open to decision makers faced with retrofitting existing facilities.

The computer program uses as its primary input information collected as an integral part of a thorough fire safety evaluation. This information permits the current state of the health care facility to be unambiguously identified. The least-cost or optimal combination of retrofits is identified using the following information:

(1) the current state of the health care facility;

(2) the minimum passing "score" needed to achieve compliance; and

(3) the anticipated costs of each retrofit measure.

The computer program then generates and analyzes a class of alternative retrofits. The optimal combination of retrofits and any alternatives which the program produces, usually between 10 and 20, are then summarized in tabular form and ranked according to cost. By using this approach, health care facility decision makers should have greater flexibility in choosing among retrofit combinations. In particular, by providing alternatives, the decision maker has the opportunity to assess very effectively the impact that non-construction costs would have on the choice of the optimal retrofit combination.

1.3 SCOPE AND APPROACH

The documentation for the FSESCM computer program is divided into two parts, each of which is designed to be self-contained and hence may be read independently. The first part is designed to serve as a User's Manual.[1] The second part, which includes this document, is intended for use as a Programmer's Manual. This approach was taken because most users are not concerned with the internal workings of the program.

Chapter 2 uses a mathematical discussion of the FSESCM package of programs to introduce the basic philosophy behind the algorithm. All data structures are then described.

---

[1] R.E. Chapman and W.G. Hall, User's Manual for the Fire Safety Evaluation System Cost Minimizer Computer Program, National Bureau of Standards, NBSIR (in preparation).

The operational characteristics of the model are described in the third chapter. First, the basic outputs associated with a prespecified data file used in generating test results are described. Emphasis is then placed on an analysis of model test results (e.g., run times experienced, validation activities, and program portability).

The final chapter provides a set of guidelines for obtaining and setting up the model. The transfer medium is first discussed followed by an examination of the contents of the four files provided whenever a request for the source code is made. These files consist of: (1) background information; (2) the FSESCM source code; (3) a data file for initial testing; and (4) an output file for insuring model validity across operating systems. The chapter concludes with a set of provisions for updating or modifying the FSESCM source code.

The report also contains a technical appendix. Appendix A relates the documentation for the model with that outlined in the relevant Federal Information Processing Standards (FIPS) documents. A series of descriptive tables are then used to define all variables and reports. Each subroutine is then described focusing on such topics as: (1) purpose; (2) calling sequence; (3) common blocks used; and (4) reports produced. The FSESCM source code is written in FORTRAN and complies with the guidelines set down in the ANSI X3.9-1978 software standard.[1]

---

[1]American National Standards Institute, American National Standard Programming Language FORTRAN, ANSI X3.9-1978, New York, 1978.

2. FUNCTIONAL DESCRIPTION OF FSESCM

2.1 PROBLEM STRUCTURE

The FSESCM computer program is designed to balance score improvements against the cost of retrofitting so that the least-cost means of achieving compliance to the Life Safety Code can be identified. The core concept behind FSESCM is a mathematical technique known as linear programming. In its usual context linear programming deals with the problem of allocating limited resources among competing activities in an optimal way. At the foundation of any linear programming problem is a mathematical model which describes the problem of concern. In this case, the mathematical model is the Fire Safety Evaluation System. The term "linear" refers to the requirement that all mathematical functions in the model are linear. The term "program" is used in the general sense in that it refers to a plan rather than a computer program *per se*.

In addition to the least cost solution, FSESCM contains a procedure for systematically generating alternative solutions, many of which are close in cost to the optimum. Two groups of alternative solutions are generated to facilitate the design selection process. The first group is based on the input condition of each fire zone. The objective here is to provide an opportunity to force each initial condition to stay in a solution *and* for each potential retrofit to be in a solution. The second group of solutions is based on a prespecified set of design variable qualifiers. The objective here is to insure design compatibility across fire zones. A fire zone is defined as a space separated from all others by floors, horizontal exits, or smoke barriers. Both groups of solutions are generated for each fire zone input. The second group of solutions is used to produce a series of compliance strategies for the entire building within which the key design variable qualifiers are held constant. These solutions are then printed out in ascending order of cost.

The basic methodology behind the FSESCM computer program is summarized in figure 2.1. This figure is a gross-level flowchart of the computer program; it consists of two pages of diagrams. In the discussion which follows, it will be assumed that data on several buildings are being run in a batch mode.[1]

The computations begin with the circular symbol labeled as FSESCM. Prior to the analysis of any data, however, all key variables are first initialized. Background information which includes the facility name and location, the contact person, the type of facility, and a set of cost multipliers is then read in. If the facility type or cost multipliers are out of range, an error message is written and the defective varaible is reset to its default value. These errors are not treated as fatal (i.e., they do not cause the run for this building to abort) since it was rationalized that users would benefit more from diagnostics received later on (most of which are associated with fatal errors) if the program were allowed to continue screening their data. Thus although it is possible that the incorrect problem could be solved, due to a default setting, it is more likely that other errors will be encountered

---

[1]Without loss of generality the term card image could be substituted wherever the word card is used.

later on whose diagnostics will give the user a better idea of where the problem is incorrectly formulated than if the run for the building were aborted after the first block of cards was read. The background information as input or after modification is then printed out for reference and as an aid for comparing the results of different sets of runs for the same building. The fire zone inputs are then read. If an error is encountered at this time, the logic flow is transferred to statement label C on the second page of the flowchart. Immediately after entering the section of logic which begins with statement label C, the program writes an error message which should enable the user to locate the problem. The model then reads over the remaining data for this building, printing out the information on the cards (up to 20) following the one where the error occurred. If this was not the last building in the batch, then the program is transferred back to the logic flow which follows statement label A, where the program attempts to read the background information on the next building. If this was the last building in the batch, then the program stops. If no error was encountered when the fire zone inputs were read, then an input summary is printed out as well as the estimated costs for each retrofit state in Table 4 of the FSES which was deemed feasible.

The control card(s) for the fire zone are then read within the section of logic which follows statement label D on the first page of the flowchart. If an error is encountered on a control card, the logic flow is transferred to statement label C and the operations discussed above are performed. If no error was encountered, then the program will check if a solution is desired. If so, the logic flow is transferred to statement label E on the second page of the flowchart.

The logic associated with the four processing symbols (shown below statement label E) is as follows. First, the linear programming relaxation[1] is solved and stored as a bounding solution and an advanced starting point[2] for all future solutions for this fire zone. The two classes of solutions discussed earlier are then generated. A class of design equivalent solutions for use in matching common retrofit strategies across fire zones is then generated. The program then classifies and sorts all solutions. Once all four processes have been completed, a fire zone summary report is printed out. This report includes data on the fire zone location and a set of information on each solution generated. The logic flow is then returned to statement label D where the next control card is read.

Following the logic flow down to the second decision block beneath statement label D and assuming no solution is desired, the program checks if a modification is required. If so, the modification is read and checked for correctness. If an error is encountered, the logic flow is transferred to statement label C where the operations described earlier are carried out. If no error was encountered, then the logic flow is returned to statement label D where the next control card is read.

---

[1]The formulation associated with the FSES is actually a 0-1 integer programming problem. FSESCM uses a heuristic post-processor to ensure integrality.

[2]The advanced starting point permits all other solutions to be generated with only a fraction of the iterations which would otherwise have been required.

Figure 2.1 Flowchart of the FSESCM Computer Program

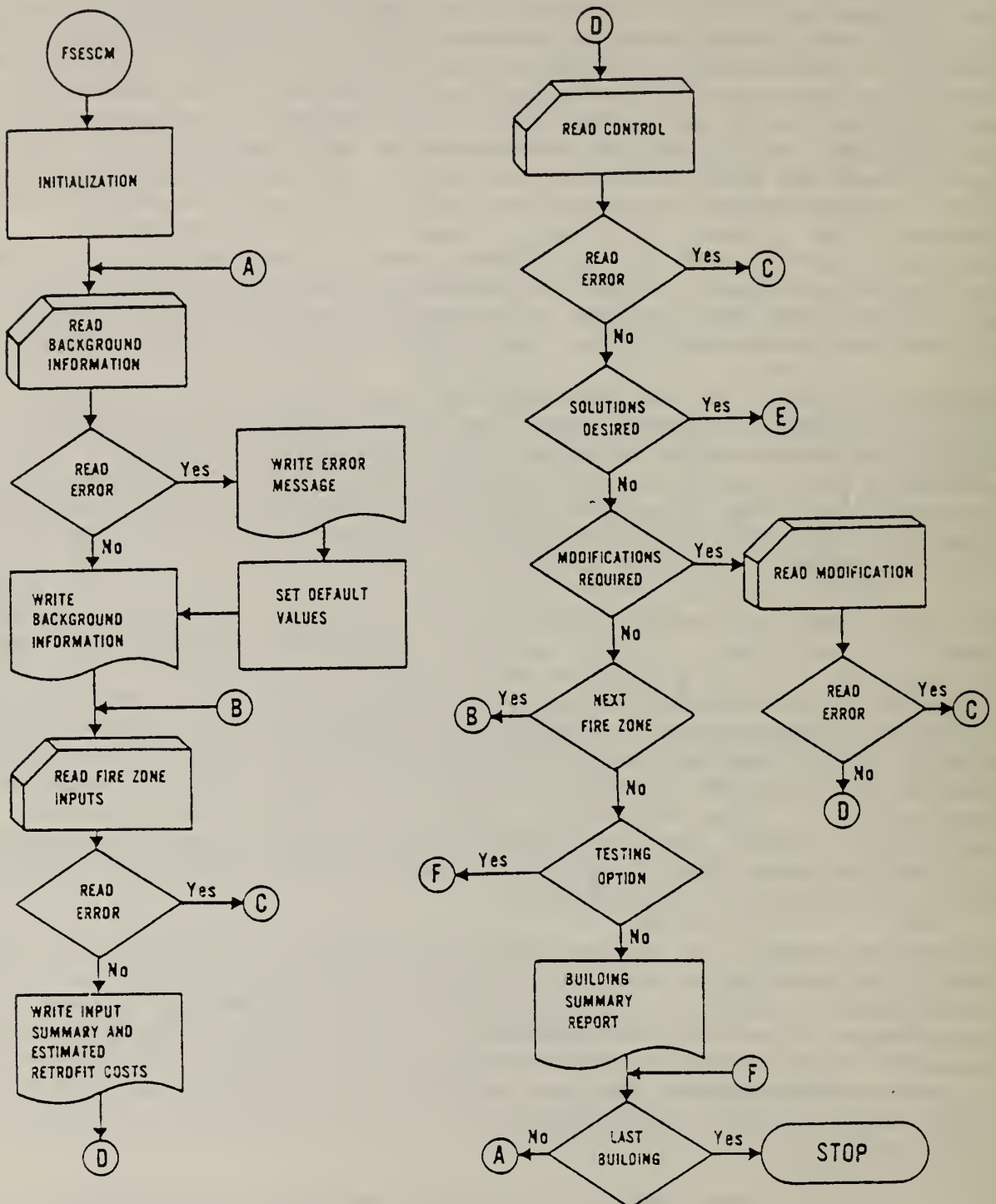
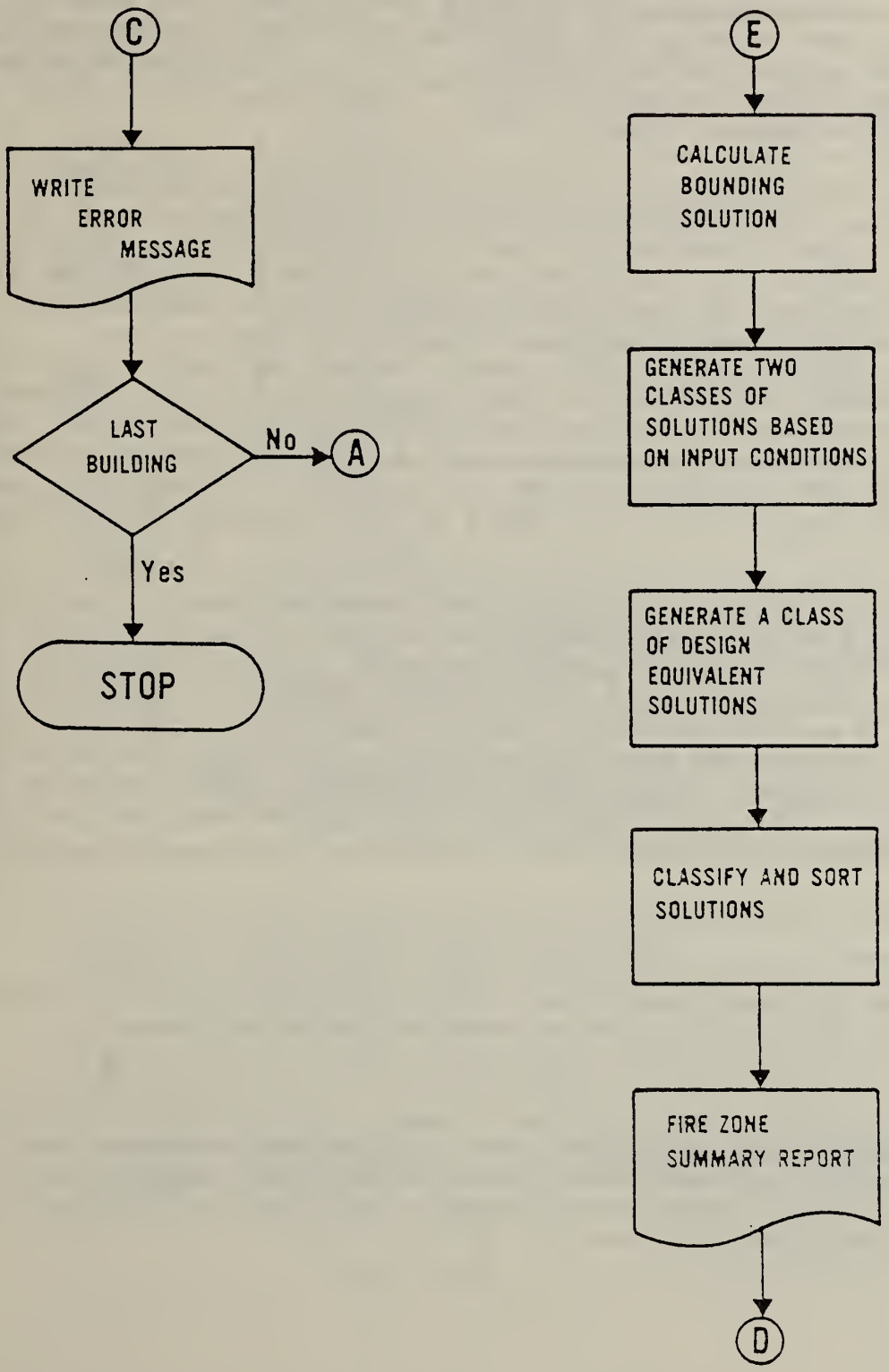


Figure 2.1 Flowchart of the FSESCM Computer Program (continued)



Returning to statement label D, following the logic flow down to the third decision block and assuming no modifications are required, the program checks if all fire zones for this building have been analyzed. If more fire zones are to be analyzed, the logic flow is transferred to statement label B, where data on the next fire zone are read. The program then checks if the testing option has been exercised for this building. The testing option permits users to check their input data for consistency without incurring the costs of generating the various classes of solutions for the first fire zones and then encountering a fatal error on one of the last fire zones.[1] If the testing option for the building has been exercised, then the logic flow skips down to statement label F where the program checks if this is the last building in the batch. If so, the program stops.

If the testing option has not been exercised, then this is the last fire zone for the building under consideration and the building summary report is output. This report is divided into two parts. The first part shows for each design classification: the design variable qualifiers; the total cost for the building; and the cost of prescriptive compliance for the building. The second part shows for each fire zone: the fire zone location; the post-retrofit name for each building safety feature; any surplus over the safety requirement; and the cost to comply for that fire zone. The program then checks if this is the last building in the batch. If it is not, then the logic flow is transferred to statement label A where the program attempts to read the background information on the next building. If this is the last building in the batch, then all buildings have been analyzed and the program stops.

The FSES consists of seven worksheets. FSES Tables 1 through 3 (exhibit 2.1) and 6 (exhibit 2.4) use existing service activities at the facility to determine the level of general safety required to be equivalent to the 1981 edition of the Life Safety Code. FSES Table 4 (exhibit 2.2) gives the scores associated with each building safety feature/state pair.[2] FSES Table 5 (exhibit 2.3) provides the means for calculating performance against each of four safety requirements (containment, extinguishment, people movement, and general safety). FSES Table 7 provides a check against compliance with each of the four safety requirements.

---

[1]The solutions from a previous batch run for a particular building are not stored, so it would be necessary to rerun the entire building.

[2]There are 13 building safety features {construction, interior finish (corridors and exits), interior finish (rooms), corridor partitions/walls, doors to corridor, zone dimensions, vertical openings, hazardous areas, smoke control, emergency movement routes, manual fire alarms, smoke detection and alarms, and automatic sprinklers} defined in the FSES. Each feature is subdivided into a set of states.

Table 1. OCCUPANCY RISK PARAMETER FACTORS

RISK PARAMETERS — RISK FACTOR VALUES

1. PATIENT MOBILITY (M)

MOBILITY STATUS	MOBILE	LIMITED MOBILITY	NOT MOBILE	NOT MOVABLE
RISK FACTOR	1.0	1.6	3.2	4.5

2. PATIENT DENSITY (D)

PATIENT	1-5	6-10	11-30	>30
RISK FACTOR	1.0	1.2	1.5	2.0

3. ZONE LOCATION (L)

FLOOR	1ST	2ND OR 3RD	4TH TO 6TH	7TH AND ABOVE	BASE-MENTS
RISK FACTOR	1.1	1.2	1.4	1.6	1.6

4. RATIO OF PATIENTS TO ATTENDANTS (T)

PATIENTS	1-2	3-5	6-10	>11	ONE OR* MORE
ATTENDANT	1	1	1	1	NONE
RISK FACTOR	1.0	1.1	1.2	1.5	4.0

5. PATIENT AVERAGE AGE (A)

AGE	UNDER 65 YEARS AND OVER 1 YEAR	65 YEARS & OVER 1 YEAR & YOUNGER
RISK FACTOR	1.0	1.2

* RISK FACTOR OF 4.0 IS CHARGED TO ANY ZONE THAT HOUSES PATIENTS WITHOUT ANY STAFF IN IMMEDIATE ATTENDANCE

Table 2. OCCUPANCY RISK FACTOR CALCULATION

	M		D		L		T		A		F
OCCUPANCY RISK	☐	×	☐	×	☐	×	☐	×	☐	=	☐

Table 3A. (NEW BUILDINGS)

	F		R
1.0 X	☐	=	☐

Table 3B. (EXISTING BUILDINGS)

	F		R
0.6 X	☐	=	☐

Exhibit 2.2 Worksheet for Determining the Values of Each Building Safety Feature/State Pair

Table 4. SAFETY PARAMETERS VALUES

PARAMETERS	PARAMETERS VALUES						
1. CONSTRUCTION	COMBUSTIBLE				NON-COMBUSTIBLE		
	WOOD FRAME		ORDINARY				
FLOOR OF ZONE	UNPROTECTED	PROTECTED	UNPROTECTED	PROTECTED	UNPROTECTED	PROTECTED	FIRE RESIST.
FIRST	−2	0	−2	0	0	2	2
SECOND	−4	−2	−4	−2	−2	2	4
THIRD	−4	−2	−4	−2	−4	2	4
4TH & ABOVE	−13	−7	−13	−7	−9	−7	4

2. INTERIOR FINISH (Corr. & Exit)	CLASS C	CLASS B	CLASS A		
	−5	0	3		
3. INTERIOR FINISH (Rooms)	CLASS C	CLASS B	CLASS B		
	−3	1	1		
4. CORRIDOR PARTITIONS/WALLS	NONE OR INCOMPLETE	<1/3 HR	>1/3 <1.0 HR	≥1.0 HR.	
	−10 [0]*	0	1 [0]*	2 [0]*	
5. DOORS TO CORRIDOR	NO DOOR	<20 MIN.FR	>20 MIN. FR	>20 MIN. FR & AUTO CLOS.	
	−10	1	1 [0]+	2 [0]+	

6. ZONE DIMENSIONS	DEAD END MORE THAN 100'	DEAD END 50'-100'	DEAD END 30'-50'	NO DEAD ENDS >30' & ZONE LENGTH IS		
				>150'	100'−150'	<100'
	−6 [0]**	−4 [0]**	−2[0]**	−2	0	1
7. VERTICAL OPENINGS	OPEN 4 OR MORE FLOORS	OPEN 2 OR 3 FLOORS	ENCLOSED WITH INDICATED FIRE RESIST.			
			<1 HR.	≥1HR.<2 HR.	≥2 HR.	
	−14	−10	0	2 [0]++	3 [0]++	
8. HAZARDOUS AREAS	DOUBLE DEFICIENCY		SINGLE DEFICIENCY		NO DEFICIENCIES	
	IN ZONE	OUTSIDE ZONE	IN ZONE	IN ADJACENT ZONE		
	−11	−5	−6	−2	0	
9. SMOKE CONTROL	NO CONTROL	SMOKE PARTITION	MECH. ASSISTED SYSTEMS BY ZONE			
	−5[0]***	0	3			

10. EMERGENCY MOVEMENT ROUTES	<2 ROUTES	MULTIPLE ROUTES			
		DEFICIENT CAPACITY	W/O HORIZONTAL EXIT(s)	HORIZONTAL EXIT(s)	DIRECT EXIT(s)
	−8	−2	0	3	5
11. MANUAL FIRE ALARM	NO MANUAL FIRE ALARM		MANUAL FIRE ALARM		
			W/O F.D. CONN.	W/F.D. CONN.	
	−4		1	2	
12. SMOKE DETECTION & ALARM	NONE	CORRIDOR ONLY	ROOMS ONLY	CORRIDOR & HABIT. SPACE	TOTAL SPACE
	0	2	3	4	5
13. AUTOMATIC SPRINKLERS	NONE	CORRIDOR & HABIT. SPACE	TOTAL SPACE		
	0	8	10		

NOTE. * Use [0] when item 5 is −10.
** Use [0] when item 10 is −8.
*** Use [0] in zone with less than 31 patients in existing buildings.
+ Use [0] when item 4 is −10.
++ Use [0] when item 1 is based on first floor zone or on an unprotected type of construction.

Exhibit 2.3 Worksheet for Calculating Containment Safety, Extinguishment Safety, People Movement Safety, and General Safety

Table 5. **INDIVIDUAL SAFETY EVALUATIONS**

SAFETY PARAMETERS	CONTAINMENT SAFETY (S_1)	EXTINGUISHMENT SAFETY (S_2)	PEOPLE MOVEMENT SAFETY (S_3)	GENERAL SAFETY (S_G)
1. CONSTRUCTION				
2. INTERIOR FINISH (Corr. & Exit)				
3. INTERIOR FINISH (Rooms)				
4. CORRIDOR PARTITIONS/WALLS				
5. DOORS TO CORRIDOR				
6. ZONE DIMENSIONS				
7. VERTICAL OPENINGS				
8. HAZARDOUS AREAS				
9. SMOKE CONTROL				
10. EMERGENCY MOVEMENT ROUTES				
11. MANUAL FIRE ALARM				
12. SMOKE DETECTION & ALARM				
13. AUTOMATIC SPRINKLERS			÷ 2 =	
TOTAL VALUE	S_1 =	S_2 =	S_3 =	S_G =

Table 6. MANDATORY SAFETY REQUIREMENTS						
	CONTAINMENT S_a		EXTINGUISHMENT S_b		PEOPLE MOVEMENT S_c	
ZONE LOCATION	New	Exist.	New	Exist.	New	Exist.
FLOOR 1	9	5	6(4)*	3	6(4)*	1
ABOVE OR BELOW FLOOR 1	14	9	8(6)*	5	9(7)*	3

* Use values in parentheses ( ) for hospitals

Table 7. ZONE SAFETY EQUIVALENCY EVALUATION						YES	NO
CONTAINMENT SAFETY (S_1)	less	MANDATORY CONTAINMENT (S_a)	≥0	S_1 □ − S_a □ = C □			
EXTINGUISHMENT SAFETY (S_2)	less	MANDATORY EXTINGUISHMENT (S_b)	≥0	S_2 □ − S_b □ = E □			
PEOPLE MOVEMENT SAFETY (S_3)	less	MANDATORY PEOPLE MOVEMENT (S_c)	≥0	S_3 □ − S_c □ = P □			
GENERAL SAFETY (S_G)	less	OCCUPANCY RISK (R)	≥0	S_G □ − R □ = G □			

The notation for exposition of the model is based on the worksheets. Consider the numbered building safety features of FSES Table 4 as rows, and denote them by i. (Note that although construction contains four rows, only one is appropriate at any given time.) Consider the columns of FSES Table 4 as states, and denote them by j.

The problem then becomes one of minimizing the total cost of compliance, where the objective function is a linear combination of the state variables, X_{ij}, and the transition costs, C_{ij}. In particular, the state variable X_{ij}, takes on a value of 1 if the j^{th} state of the ith building safety feature is in the solution and is 0 otherwise. The transition cost, C_{ij}, associated with the transition from the initial state of the ith building safety feature to the j^{th} state is defined as follows:

C_{ij} = 0 if in the initial state (i.e., $j = j^{i}_{input}$);

= arbitrarily large if j is less than the initial state or is deemed to be nonfeasible on engineering grounds; otherwise

= estimated cost.

The problem may now be expressed algebraically as:

$$\text{minimize} \quad \sum_{i=1}^{13} \sum_{j=j^{i}_{min}}^{j^{i}_{max}} C_{ij}X_{ij}$$

subject to

(1) a generalized upper bound constraint for each building safety feature

$$\sum_{j=j^{i}_{min}}^{j^{i}_{max}} X_{ij} = 1 \quad i = 1,\ldots, 13 \quad \text{where } X_{ij} = 0 \text{ or } 1$$

and

(2) a performance constraint for each safety requirement

$$\sum_{i=1}^{13} \sum_{j=j^{i}_{min}}^{j^{i}_{max}} W_{ik}V_{ij}X_{ij} - Y_k = R_k \quad k = 1, 2, 3, 4$$

where

j^i_{min} = the minimum state number[1] associated with the i^{th} building safety feature;

j^i_{max} = the maximum state number associated with the i^{th} building safety feature;

W_{ik} = a weighting factor for the state value (W_{ik} = 0, 1/2, or 1);

V_{ij} = the state value;

Y_k = a nonegative surplus variable; and

R_k = the safety requirement.

The above formulation indicates that the problem is a binary (0-1) integer program. However, due to the structure imposed on the problem by the 13 generalized upper bound constraints, the integrality requirements may be relaxed to non-negativity constraints. The associated linear programming relaxation is easily solved through application of the revised simplex method.[2] This algorithm was selected both because a non-proprietary version had been developed in house and it had the capability for using an advanced starting basis. The advanced starting basis was particularly important because, for each fire zone, the optimizer is called interatively in order to generate the two sets of alternative solutions.[3] Thus, for each fire zone, it is only necessary to solve the problem from scratch once; all other solutions use the initial bounding (continuous) solution as a starting basis.

---

[1]States are numbered sequentially from 1 to 56 (e.g., construction consists of state numbers 1 through 7, and automatic sprinklers consists of state numbers 54 through 56).

[2]Additional information on this subject may be found in the introductory text by Gass (S.I. Gass, Linear Programming: Methods and Applications (New York, McGraw-Hill Book Company, 1975)). Additional documentation for the revised simplex algorithm used in FSESCM is contained in: W. G. Hall, R. H. F. Jackson and P. B. Saunders, The National Bureau of Standards Linear and Quadratic Programming Subroutines, NBS Report 10695, 1972.

[3]A process known as state forcing is used to generate the alternative solutions. This process systematically forces building safety feature/state pairs into and out of the basis by changing their transition costs. Since changing transition costs operates on the objective function, it does not affect the feasibility of the linear programming relaxation. An advanced starting basis may therefore be used which permits the relaxation to be solved very efficiently. State forcing is particularly valuable because it facilitates the answering of many what-if questions.

It should be noted that the linear programming relaxation does not, in general, produce an all integer solution. Furthermore, the interdependencies defined on FSES Table 4 (e.g., between corridor partitions/walls and doors to the corridor) were not stated in the formulation given above. The program thus contains a heuristic post processor to impose integrality and to assure that any interdependencies do not render the problem infeasible.

Three state-forcing loops within the MAIN program are used to generate the alternate solutions. In the first state-forcing loop, a subset of up to 13 alternate solutions based on the input states of FSES Table 4 is generated. Each building safety feature, in turn, is inhibited from making a transition to any but the input state. The linear programming relaxation is then solved and integerized. The mechanism for generating this subset, for each i, is to set C_{ij} arbitrarily large for all j, $j^i_{min} \leq j \leq j^i_{max}$, $j \neq j^i_{input}$.

In the second state-forcing loop, a subset of up to 56 alternative solutions based on the bounding solution of the linear programming relaxation is generated. Each X_{ij} of the bounding solution is examined in turn. If X_{ij} is unequal to 1, then the C_{ij}'s are adjusted so that the corresponding X_{ij} is forced into the solution. If X_{ij} is equal to 1, then the C_{ij}'s are adjusted so that X_{ij} is inhibited from appearing in the solution.

The solutions so generated are not necessarily distinct. The number of solutions will, of course, depend upon the input states and the number of allowable transitions. These solutions constitute the first of the two classes mentioned earlier.

In the third state-forcing loop, a subset of up to 40 alternative solutions based on design considerations is generated. Six building safety features have their transition states preset to insure design compatibility. The design variable qualifiers used in establishing the 40 design classifications are defined in table 2.1. Each X_{ij} within a building safety feature is examined in turn, however; unlike the two previous cases, all building safety features must be examined before the optimizer is called. If the i^{th} building safety feature is to be preset, then all of the C_{ij}'s are adjusted so that the corresponding X_{ij} is forced into the solution. If the i^{th} building safety feature is not to be preset, then no changes are made to the C_{ij}'s. Once all building safety features have been examined, all appropriate variables have been preset, and if a meaningful transition[1] can take place for each of the six building safety features, the optimizer is called to solve the linear programming relaxation. The solution is then integerized and checked for feasibility should an interdependency come into play.

---

[1]If a preset design variable qualifier, X_{ij}, has a cost, C_{ij}, which is arbitrarily high, then no meaningful transition is possible and the logic jumps to the end of the loop.

2.2 DATA STRUCTURES

There are four types of data referenced in any run of the FSESCM model. Two of these data types are external; their values are determined by entries made by the user. The two remaining are internal; they are associated with the three labeled COMMONS.

Those data required of the user in order to run the model can be divided into two types. The first type is designated as "background information" (see table 2.1). This information covers such items as the name and location of the facility, who to contact if a question arises and the type of building being analyzed. The second type is designated as "specific information" and refers to data which must be input for each fire zone. These data are used to set up the optimization problem. They are summarized in table 2.1. The user also has available a set of options which affect the optimization problem in a variety of ways. Each option and its effect on the solution are described in table 2.2.

All data are designed so that they can be easily and reliably collected at the same time as the fire zone is evaluated. Since some of the states within FSES Table 4 are not associated with a single element, a worksheet was developed which lists those building components requiring treatment in order to move from one state within a building safety feature to another.[1] The information collected on the worksheet closely follows the design of FSES Table 4. The retrofit measures which the data from the worksheet permit the program to consider are summarized in table 2.3. Associated with each potential retrofit (i.e., one which is deemed feasible on engineering grounds) is a set of information on the one or more elements which must be treated to move to a higher state. This information is stored in an "element count matrix." The product of the element count matrix and the element cost matrix, which is initialized in the BLOCK DATA routine, yields the total cost associated with each potential retrofit. In order to address a wide variety of "what-if" questions, the user has the option to modify the costs of a particular retrofit, the score required to pass, or both.

All data under the background information category are read within the MAIN program. These data are always the first seven lines of the first building in the external "user data file." The first six lines are used as an address label as well as a means for identifying a contact person should additional information be required. The data are read as alphanumeric (A40) and stored in the character array HBLDG (CHARACTER*40 HBLDG(6)). The building type (i.e., hospital or nursing home), the building age (i.e., new single story, new multistory, existing single story, existing multistory), and construction cost modifiers are then read. The first two variables are read under an I2 format, whereas the last three are read under an F5.2 format. The construction cost modifiers are used to update the per unit labor and material cost figures initialized in the BLOCK DATA routine to reflect differences in the regional markets for labor services and building materials as well as cost growth. This approach was taken so that it would not be necessary for users to load their own values into the CPUL and CPUM arrays. Both arrays are 13 by 7 by 10.

---

[1]The worksheet is described in detail in the User's Manual.

Table 2.1 Input Data Requirements for the FSESCM Computer Program

Major Heading	Item	Purpose
Background Information	Facility Name and Location	Mailing label for the finished run.
	Contact Person and Telephone Number	Who to talk to for more information.
	Building Type	Specifies columns of FSES Table 6.*
	Cost Multipliers	Adjusts relative costs for cost growth over time and for regional price differentials in labor and material.
Specific Information	Fire Zone Location	Specifies the row of FSES Table 6.* Affects the scores of construction and vertical opening states.
	Number of Patients	Affects the score of the smoke control state.
	Risk Parameters	Determines the value of the General Safety requirement. See FSES Table 1 (exhibit 2.1).
	Building Safety Feature Numbers	Determines the current scores for all four safety attributes. Constrains the retrofit options to those states with a higher state value.
	Element Counts	Shows minimum number of deficient elements which must be treated to move to a higher state. Entered for each building safety feature.
	Options	Tells program what to do or look for next. See table 2.3.

*FSES Table 6 (exhibit 2.4) defines the mandatory safety requirements for containment, extinguishment and people movement.

Table 2.2 User Options Available with the FSESCM Computer Program

Option	Purpose
SOLVE	Causes an optimum solution for the fire zone input to be generated. A set of retrofit strategies which satisfy all of the requirements of the Life Safety Code as well as several building design criteria is also generated.
CHANGE	Adjusts the state transition cost to the value specified by the user.
REQUIR	Increases a safety requirement by a percentage specified by the user.
NEXT	Tells the program to look for the data on the next fire zone or the next building.
LAST	Signals that all data for the building under study have been analyzed and that the solutions for the total building should be output.
TEST	Checks all input data for consistency.
FINAL	Tells the program to stop; all data for the run have been output.

Table 2.3 Retrofit Measures Considered by the FSESCM Computer Program

Building Safety Feature		Retrofit
1.	Construction	Resheath walls and partitions
		Protect columns
		Protect beams
		Protect decking
2.	Interior Finish (Corridors and Exits)	Install drywall
		Coat walls with retardant
		Coat ceilings with retardant
		Install carpet
3.	Interior Finish (Rooms)	Install drywall
		Coat walls with retardant
		Coat ceilings with retardant
4.	Corridor Partitions/Walls	Install partition slab to slab
		Extend existing partitions to slab
		Replace see-through panels and frames
		Replace see-through panels only
		Install drywall
5.	Doors to Corridor	Replace doors and frames
		Replace doors only
		Replace latch
		Replace view panel
		Install closers
6.	Zone Dimensions	Install cross connection
		Install stairway
		Install smoke partition
7.	Vertical Openings	Frame and sheath
		Sheath only
		Install doors and frames
		Install doors only
8.	Hazardous Areas	Install sprinklers
		Install Class B door
		Install drywall
9.	Smoke Control	Install smoke partition
10.	Emergency Movement Routes	Install exit stairway
		Install emergency lighting
		Install horizontal exit
11.	Manual Fire Alarm	Install control panel
		Install pull station
		Connect to fire department
12.	Smoke Detection and Alarm	Install smoke detectors
13.	Automatic Sprinklers	Install wet system
		Install dry system

All data under the specific information category begin with card eight and continue until a LAST card is encountered. Data are read in sequence, a single fire zone at a time. Solutions are generated for each fire zone. Five of the 13 building safety features require special treatment; they are handled as subroutines. All other data is read within the MAIN program. Data on the zone location and number of patients are first read in according to an I4 format. Information on FSES Table 1 is then read in according to an I3 format; it is used to calculate the general safety requirement. Information on the existing state for each of the 13 building safety features is then read in according to an I3 format. Since a regression in score is not permitted, these values set lower limits on DO loops. With the exception of construction, zone dimensions, hazardous areas, smoke detection and alarm, and automatic sprinklers, all data are read according to an I6 format. Each line of the file corresponds to a potential retrofit. The number of entries on the line ranges from 1 to 8; critical element counts follow closely the retrofit measures defined in table 2.3.

Construction and smoke detection and alarm use the same I6 format as in the MAIN program but have more complicated costing procedures than allowed in the MAIN's general purpose routine. The three other building safety features which require special handling also require special formats. The zone dimension feature requires information on the length of the fire zone to be entered according to an I6 format. This information is used to calculate the number of smoke partitions which must be installed in order to reduce the zone's dimensions to the levels defined in FSES Table 4. Hazardous areas require special treatment because any of three complicating factors can occur. Some hazardous areas within the fire zone may have: (1) a double deficiency; (2) a single deficiency but lack sprinklering; or (3) a single deficiency with sprinklering present. Since the state value is determined by the worst-case condition, a mixture of hazardous area types may exist. By separately accounting for these complicating factors, it becomes possible to unambiguously estimate the transition costs to all higher states. Qualifiers for each of the three complicating factors are read in according to an I3 format. Sprinklers require qualifiers for both sprinkler type and water supply. The sprinkler type qualifiers are: (1) wet exposed; (2) wet concealed; (3) dry exposed; and (4) dry concealed. The water supply qualifiers are: (1) adequate; (2) not adequate; and (3) unknown. Both sets of qualifiers are read according to an I3 format.

The option/control cards defined in table 2.2 are read within the MAIN program. They are composed of an alphanumeric portion and an integer portion. The command itself is read according to an A6 format. If the user is exercising the CHANGE option, then the building safety feature number/state number designators are read in according to an I3 format. The user-defined transition cost in dollars is then read according to an I7 format. If the user is exercising the REQUIR option, then both the percentage change and the safety requirement to which that change is to be applied are read according to an I3 format. The integer portion for all other commands has no meaningful interpretation and should therefore be left blank.

The internally stored data are made available to the program through the SETUPA, SETUPB, and RESET COMMONs. The SETUPA and SETUPB labeled COMMONs are associated with the BLOCK DATA routine. This routine initializes all variables in the SETUPA and SETUPB COMMONs. Once initialized, regardless of the number of facilities analyzed, all SETUPA and variables remain constant throughout the run. The variables contained in the SETUPA COMMON are listed in alphabetical order in table 2.4. Dimensions, variable type, and purpose are also given in the table. The variables contained in the SETUPB COMMON are described in table 2.5.

The RESET COMMON is associated with subroutine INSETS. The INSETS subroutine initializes (or reinitializes) the values of all major work spaces. It is designed to clear the working spaces so that no information from a previous building could influence the calculations for the one being analyzed currently. INSETS is called in four ways:

(1) initially;

(2) after each building has been analyzed;

(3) if the output of the revised simplex (RVSMPX) subroutine was other than normal; and

(4) if an error in the runstream for the building under analysis was encountered.

The variables contained in the RESET COMMON are described in table 2.6.

Table 2.4 "SETUPA" COMMON Storage

Variable	Data Type	Contents
CPUL(13,7,10)	REAL	Labor cost per unit
CPUM(13,7,10)	REAL	Materials cost per unit
G(13,7)	REAL	FSES Table 4 for print out
IEC1(40)	INTEGER	Design classification qualifiers
IEC5(40,5)	INTEGER	Design classification flags
IST4(56,2)	INTEGER	Simplex/Table 4 crosswalk
IT1M(5)	INTEGER	FSES Table 1: maximum state number
IT4A(7)	INTEGER	FSES Table 4: construction state number
IT4B(7)	INTEGER	FSES Table 4: construction states in increasing order
IT4M(13)	INTEGER	FSES Table 4: maximum state number
IT4S(13,2)	INTEGER	Simplex variable limits
IT8(5)	INTEGER	FSES Table 4: hazardous area states in increasing order
JPR(13,8)	INTEGER	FSES Table 4: descriptive state numbers
LCV(13)	INTEGER	Length of cost vector
T1(5,5)	REAL	FSES Table 1
T48A(5)	REAL	FSES Table 4: hazardous area increasing values row
T4A(4,7)	REAL	FSES Table 4: construction row
T4B(4,7)	REAL	FSES Table 4: construction row in increasing values
T6A(3,2,2)	REAL	FSES Table 6 for hospitals
T6B(3,2,2)	REAL	FSES Table 6 for nursing homes

Table 2.5 "SETUPB" COMMON Storage

Variable	Data Type	Contents
HLAB(56)	CHARACTER*7	State number labels
HOUT(40)	CHARACTER*45	Design classification labels
HPAR(18)	CHARACTER*30	FSES Table 4 labels
HT1(10)	CHARACTER*8	FSES Table 1 labels

Table 2.6 "RESET" COMMON Storage

Variable	Data Type	Cotents
C(56)	REAL	State transition cost
IECT(44)	INTEGER	Design classification solution counts
IPAC(100,22)	INTEGER	Integerized solution storage
ISRT(100,22)	INTEGER	Sorted solution storage
ISTAT(13)	INTEGER	State numbers/flags
ISTK(500,22)	INTEGER	Stacked solution storage
L(150)	INTEGER	Simplex general purpose vector
X(150)	REAL	Simplex solution vector
Y(150)	REAL	Working solution vector
Z(150)	REAL	Prototype solution vector

3. OPERATIONAL CHARACTERISTICS

3.1 MODEL OUTPUTS

The information printed out by the FSESCM computer program fits into three major output classifications, which are summarized in table 3.1. Each output classification is described briefly in the text which follows. The three classifications are: (1) background information; (2) reference data and solutions; and (3) design solutions.

Background Information

The first class of outputs reports includes a title page and FSES Tables 1, 6 and 4.

Title Page

The title page report also serves as a mailing label and identifies the appropriate staff member to contact in the event that a problem is encountered in analyzing the facility.

FSES Tables

FSES Tables 1, 6 and 4 are included in order to show how the occupancy risk factors are used to calculate the General Safety requirement, what values for Containment, Extinguishment and People Movement Safety are required for equivalence, and the full range of state values which contribute toward fire zone safety.

Reference Data and Solutions

The second class of output reports are generated for each fire zone. These outputs consist of a summary of all data input for the fire zone, the estimated costs of moving from the input state to each potential retrofit, and all distinct solutions generated for the fire zone.

Input Summary

The input summary provides the user with a concise statement of the data used in setting up the problem for solution. It provides the user an opportunity to check the correctness of any values input as well as a means of differentiating among several runs for the same fire zone. This report shows the location of the fire zone, the number of patients and the appropriate set of occupancy risk factors for the fire zone under study. Data on each of the 13 building safety features are then printed out. These data show the input and prescriptive state and the number of elements which must be upgraded in order to move to a higher state.

Table 3.1 Outputs of the FSESCM Computer Program

Output Classification	Report Name	Information Reported
Background Information	Title Page	Facility Name and Address, Contact Person, Telephone Number
	FSES Tables	Table 1 (Occupancy Risk), Table 6 (Safety Requirements) Table 4 (Building Safety Features)
Reference Data and Solutions	Input Summary	Fire Zone Location, Patient Count, Table 1 (State Number), Table 4 (Input, Prescriptive, Element Counts)
	Summary of Estimated Retrofit Costs	Fire Zone Location, Time/Regional Cost Modifiers, Table 4 (Input, Prescriptive, Transition Costs), Prescriptive Compliance Cost
	Fire Zone Summary	Fire Zone Location, For Each Solution by Design Classification (Post-Retrofit State Name for Each Building Safety Feature, Surplus Over Each Safety Requirement, Cost to Comply for that Fire Zone)
Design Solutions	Total Building Summary	For Each Design Classification (Design Variable Qualifiers, Total Cost for the Building, Prescriptive Compliance Cost for the Building) For Each Fire Zone (Fire Zone Location, Post-Retrofit State Name for Each Building Safety Feature, Surplus Over Each Safety Requirement, Cost to Comply for That Fire Zone)

Summary of Estimated Retrofit Costs

The costs of moving from the input state to all potential retrofits are then presented. This output includes the location of the fire zone and three types of construction cost modifiers. The modifiers are needed because all costs used within the FSESCM computer program reflect the cost of installing a particular retrofit in the Washington, D. C. area during the summer of 1981. The modifiers permit the user to adjust not only for regional price differences in the markets for labor services and building materials but also for cost growth over time. Through reference to modifiers it is not necessary to change the entries in the CPUL and CPUM arrays. All three of these cost factors are readily available from construction industry publications. Each of the 13 building safety features are then output. The data show the input and prescriptive state and the estimated cost of going to each potential retrofit. The cost of prescriptive compliance for the fire zone is also shown as a basic reference point.

Fire Zone Summary Report

This report shows each distinct solution generated as a line of output. In order to easily identify a particular solution and for ease in differentiating among solutions, the state name is printed beneath each of the 13 building safety feature column headings. The state names closely resemble and hence can be easily matched to the labels in Table 4 of the FSES. The order in which the solutions are output is based on the 40 design classifications listed in table 2.4. All solutions are ranked from least costly to most costly within a design classification; so if more than one solution was generated, the least costly is printed first followed by the second, until all solutions for that design classification are output. The program then outputs the solution(s) for the next design classification for which at least one solution was generated until the list is exhausted. The prescriptive compliance solution is then output. Three other groups of solutions are also output. They are: (1) solutions which have no deficiencies in hazardous areas but do not belong to one of the design classifications; (2) solutions which have a single deficiency in a hazardous area; and (3) solutions which have a double deficiency in a hazardous area.

Design Solutions

The third class of output reports consists of the best solutions by design classifications for the entire building. The design classification solutions are generated and stored for each fire zone. Once all data on the fire zones have been input and analyzed, all solutions are screened. The ones which match the prespecified set of design variable qualifiers are identified. If every fire zone input has at least one solution which was identified as a member of the design classification under consideration, then a solution for the entire building is generated.

Total Building Summary Report

This report gives the design variable qualifiers, the total cost of retrofitting the building for this design classification and the total cost of prescriptive compliance for the building under study. The prescriptive solution serves as a bench mark for comparison. The design classification solutions are printed out in ascending order of estimated retrofit cost for the entire building to facilitate comparison among competing design alternatives. In order to facilitate the identification of each solution, the state names for each of the 13 building safety features are printed out as are the surpluses and retrofit cost for each fire zone. Each fire zone takes up one line in the printout. If one or more fire zones did not contain this design class, no printout for the entire building is generated. Should the user wish such a retrofit, it would be necessary to synthesize it from the individual fire zone printouts.

Six exhibits are used to illustrate the type of output produced by the FSESCM model. The titles of the exhibits are designed for easy cross reference to the output reports listed in table 3.1. Each exhibit has two parts. In part A, the output report as produced by the model is shown. Part B contains a brief description of the purpose of the report and the data output. The routines, either MAIN or a subroutine, from which the report is generated are identified for reference purposes. The outputs shown in the exhibits are based on a test case example. A line-by-line description of this output for this case example may be found in the User's Manual.

The Title Page (exhibit 3.1) and FSES Tables (exhibit 3.2) output reports are generated once for each building immediately after the background information is read into the character array HBLDG. The Input Summary (exhibit 3.3) and the Summary of Estimated Retrofit Costs (exhibit 3.4) reports are generated for each fire zone prior to optimization. The Fire Zone Summary Report (exhibit 3.5) is generated once all three state forcing loops have been completed. The Total Building Summary Report (exhibit 3.6) is generated for each building after all fire zones for the building have been analyzed.

Exhibit 3.1 Sample Title Page

Part A: Printout

NORTH-WEST GENERAL HOSPITAL
MAIN BUILDING
ATTN: ROBERT E. CHAPMAN
101 RESEARCH BLVD.
MONTGOMERY, MARYLAND 20000
TEL: (301)-921-3855

Exhibit 3.1 Sample Title Page

Part B: Discussion

The title page is produced through a call to subroutine PRTBAC. The character array HBLDG is passed as an argument. The subroutine prints the contents of HBLDG (i.e., the mailing label and the contact person). The carriage returns within subroutine PRTBAC also serve as a separator between runs for different facilities. Subroutine PRTBAC is called immediately after it has read into HBLDG the facility ID, name of the contact person and facility address.

OCCUPANCY RISK FACTORS

1.	PATIENT MOBILITY	1.0	1.6	3.2	4.5	
2.	PATIENT DENSITY	1.0	1.2	1.5	2.0	
3.	ZONE LOCATION	1.1	1.2	1.4	1.6	1.6
4.	RATIO OF PT TO AT	1.0	1.1	1.2	1.5	4.0
5.	AVERAGE AGE	1.0	1.2			

MANDATORY SAFETY REQUIREMENTS FOR HOSPITALS

	CONTAINMENT SA		EXTINGUISHMENT SB		PEOPLE MOVEMENT SC	
ZONE LOCATION	NEW	EXIST	NEW	EXIST	NEW	EXIST
FIRST FLOOR	9.0	5.0	4.0	4.0	4.0	1.0
ABOVE 1ST FLOOR	14.0	9.0	6.0	6.0	7.0	3.0

PARAMETER VALUES

1.	CONSTRUCTION							
	FIRST FLOOR	-2.0	.0	-2.0	.0	.0	2.0	2.0
	SECOND FLOOR	-7.0	-2.0	-4.0	-2.0	-2.0	2.0	4.0
	THIRD FLOOR	-9.0	-7.0	-9.0	-7.0	-7.0	2.0	4.0
	4TH & ABOVE	-13.0	-7.0	-13.0	-7.0	-9.0	-7.0	4.0
2.	INTERIOR FINISH CORR AND EXIT	-5.0	.0	3.0				
3.	INTERIOR FINISH ROOMS	-3.0	1.0	3.0				
4.	CORRIDOR PARTITIONS AND WALLS	-10.0	.0	1.0	2.0			
5.	DOORS TO CORRIDOR	-10.0	.0	1.0	2.0			
6.	ZONE DIMENSIONS	-6.0	-4.0	-2.0	-2.0	.0	1.0	
7.	VERTICAL OPENINGS	-14.0	-10.0	.0	2.0	3.0		
8.	HAZARDOUS AREAS	-11.0	-5.0	-6.0	-2.0	.0		
9.	SMOKE CONTROL	-5.0	.0	3.0				
10.	EMERGENCY MOVEMENT ROUTES	-8.0	-2.0	.0	3.0	5.0		
11.	MANUAL FIRE ALARM	-4.0	1.0	2.0				
12.	SMOKE DETECTION AND ALARM	.0	2.0	3.0	4.0	5.0		
13.	AUTOMATIC SPRINKLERS	.0	8.0	10.0				

Exhibit 3.2 Sample FSES Tables

Part B: Discussion

This report is generated by the MAIN program. The labels for FSES Table 1 stored in the character array HT1 are first printed followed by the occupancy risk factors stored in the T1 array. Next, depending on the value of ITYPE, the mandatory safety requirements for the problem under analysis are output. The value of ITYPE determines whether this is a hospital (T6A) or a nursing home (T6B). FSES Table 4 is then printed out; it consists of three parts. First, the labels stored in the character array HPAR are printed out. Next, the state values for the four rows associated with the construction building safety feature stored in array T4A are printed out. Finally, the state values associated with the remaining 12 building safety features, which are stored in the G array, are printed out.

Exhibit 3.3 Sample Input Summary

Part A: Printout

INPUT SUMMARY

FLOOR 3 ZONE 1 PATIENTS 280 TABLE 1 INDICES 4 3 2 1 1

	BUILDING SAFETY FEATURE	INPUT	PRESC							
1.	CONSTRUCTION	NCOM•R	NCOM•R	C•WF•U	C•WF•P	C•OR•U	C•OR•P	NCOM•U	NCOM•P	NCOM•R
				•••••	•••••	•••••	•••••	•••••	•••••	0
2.	INTERIOR FINISH CORR AND EXIT	AAAA	BBBB	CCCC	BBBB	AAAA				
				•••••	•••••	0				
3.	INTERIOR FINISH ROOMS	CCCC	BBBB	CCCC	BBBB	AAAA				
				0	0	0				
				0	0	0				
				0	700	700				
4.	CORRIDOR PARTITIONS AND WALLS	GE 1HR	1/3-1H	NON/1N	LT1/3H	1/3-1H	GE 1HR			
				•••••	•••••	•••••	0			
5.	DOORS TO CORRIDOR	NODOOR	GE 20M	NODOOR	LT 20M	GE 20M	GE20AC			
				0	0	49	49			
				0	0	0	0			
				0	0	0	0			
				0	0	0	0			
				0	28	28	28			
				0	49	49	49			
				0	0	0	0			
				0	0	0	0			
6.	ZONE DIMENSIONS	DED•30	NO+100	DED100	DED•50	DED•30	NO+150	NO+100	NO-100	
				•••••	•••••	0	0	0	0	
				•••••	•••••	0	0	0	0	
				•••••	•••••	0	9	9	9	
				•••••	•••••	0	0	7	7	
				•••••	•••••	0	0	0	0	
				•••••	•••••	0	0	0	0	
				•••••	•••••	0	0	0	0	
7.	VERTICAL OPENINGS	ENC-1H	ENC-2H	OP•GE4	OP•2-3	ENC-1H	ENC-2H	ENC+2H		
				•••••	•••••	0	0	0		
				•••••	•••••	0	0	0		
				•••••	•••••	0	7	7		
				•••••	•••••	0	0	0		

Exhibit 3.3 Sample Input Summary

Part B: Discussion

The purpose of this report is to summarize the data used in setting up the problem for solution. Portions are generated by the MAIN program and portions are generated by the five special purpose subroutines (CCONST, CZODIM, CHAZAR, CSMOKE, CSPRNK). In addition to its use as a summary report, it is also useful in distinguishing among runs when a sensitivity analysis is performed. The first descriptive line shows the location of the fire zone (IFLOR, IZONE), the number of patients (IPATS), and the state numbers for FSES Table 1 (IT1X) as read by the MAIN program. Each of the 13 building safety features are then listed by number, I, and label, HPAR(I). The two columns to the right of the building safety feature name show the input and prescriptive state names (HLAB(M1), HLAB(M2) where M1 = j^{i}_{input} and M2 = j^{i}_{presc}). Up to 7 columns are then used as labels (HLAB(J), J=j^{i}_{min}, j^{i}_{max}) to show the number of elements which must be upgraded in order to move to higher state. The element counts correspond to the contents of the arrays IECCON (construction) IECZDM (zone dimensions), IECHAZ (hazardous areas), IECSMO (smoke detection and alarm), IECSPR (sprinklers), and INELT (otherwise). The actual element counts consist of from 1 to 8 rows depending on the building safety feature and its condition. If the building safety feature was input in the highest state, then only one row is used regardless of the number of elements which can be treated. If the building safety feature was not in the highest state, then each row corresponds to a particular element. In interpreting the values, it is necessary to note that the symbol ****** indicates that the state whose name appears above it not a permissible retrofit. This may result because the state is below the input and hence would result in a regression in score or it is precluded on engineering grounds. The states which are permissible have either a value of zero or some positive integer recorded. The second page of the printout consists of data on building safety features 8 through 13. These data are not reproduced here since their interpretation is the same as given above.

SUMMARY OF ESTIMATED RETROFIT COSTS

FLOOR 3 ZONE 1 COST GROWTH FACTOR 1.15 LOCATION MODIFIERS: LABOR 1.00 MATERIALS 1.00

	BUILDING SAFETY FEATURE	INPUT	PRESC							
1.	CONSTRUCTION	NCOM*R	NCOM*R	C*WF*U	C*WF*P	C*OR*U	C*OR*P	NCOM*U	NCOM*P	NCOM*R
	ESTIMATED RETROFIT COST			******	******	******	******	******	******	0
2.	INTERIOR FINISH CORR AND EXIT	AAAA	BBBB	CCCC	BBBB	AAAA				
	ESTIMATED RETROFIT COST			******	******	0				
3.	INTERIOR FINISH ROOMS	CCCC	BBBB	CCCC	BBBB	AAAA				
	ESTIMATED RETROFIT COST			0	604	644				
4.	CORRIDOR PARTITIONS AND WALLS	GE 1HR	1/3-1H	NON/1N	LT1/3H	1/3-1H	GE 1HR			
	ESTIMATED RETROFIT COST			******	******	******	0			
5.	DOORS TO CORRIDOR	NODOOR	GE 20M	NODOOR	LT 20M	GE 20M	GE20AC			
	ESTIMATED RETROFIT COST			0	5233	39043	39043			
6.	ZONE DIMENSIONS	OED*30	NO+100	OE0100	DE0*50	OE0*30	NO+150	NO+100	NO-100	
	ESTIMATED RETROFIT COST			******	******	0	186300	207230	207230	
7.	VERTICAL OPENINGS	ENC-1H	ENC-2H	OP*GE4	OP*2-3	ENC-1H	ENC-2H	ENC+2H		
	ESTIMATED RETROFIT COST			******	******	0	5635	6440		
8.	HAZARDOUS AREAS	SGL IN	NO OEF	OBL IN	OBLOUT	SGL IN	SGLOUT	NO OEF		
	ESTIMATED RETROFIT COST			******	******	0	******	1610		
9.	SMOKE CONTROL	PARTTN	PARTTN	NO CTL	PARTTN	MECZON				
	ESTIMATED RETROFIT COST			******	0	******				
10.	EMERGENCY MOVEMENT ROUTES	OEFCAP	W/OH*E	LT2RTS	OEFCAP	W/OH*E	H*EXIT	DIRECT		
	ESTIMATED RETROFIT COST			******	0	0	49106	******		
11.	MANUAL FIRE ALARM	FO CON	FO CON	NO ALR	W/OCON	FO CON				
	ESTIMATED RETROFIT COST			******	******	0				
12.	SMOKE DETECTION AND ALARM	NONE	NONE	NONE	CORDOR	ROOMS	CORHAB	TTLZON		
	ESTIMATED RETROFIT COST			0	14168	42504	64615	122575		
13.	AUTOMATIC SPRINKLERS	NONE	NONE	NONE	CORHAB	TTLBLO				
	ESTIMATED RETROFIT COST			0	137023	222348				

THE COST OF PRESCRIPTIVE COMPLIANCE FOR THIS FIRE ZONE IS: 254122

Exhibit 3.4 Sample Summary of Estimated Retrofit Costs

Part A: Printout

Exhibit 3.4 Sample Summary of Estimated Retrofit Costs

Part B: Discussion

The purpose of this report is to summarize the cost data used in generating the bounding solution (i.e., the solution to the continuous linear programming problem). This report is generated through a call to subroutine PCOSTS. The first descriptive line shows the location of the fire zone (IFLOR, IZONE), and the three construction cost modifiers (CMODT, CMODL, CMODM). As in the previous report, each of the 13 building safety features are then listed by number, I, and label, HPAR(I). The two columns to the right of the building safety feature name show the input and prescriptive state names (HLAB(M1), HLAB(M2) where M1 $= j^{i}_{input}$ and M2 $= j^{i}_{presc}$). Up to 7 columns are then used as labels (HLAB(J), J $= j^{i}_{min}, j^{i}_{max}$) to show the transition costs which must be incurred in order to move to a higher state. The estimated values of the transition costs (C(J), J = 1,56) are stored temporarily in the L vector prior to output so they are consistent with an integer format. In interpreting the values, it is necessary to note that the symbol ****** indicates that the state whose name appears above it is not a permissible retrofit; it is assigned an arbitrarily high value to prevent its occurrence. The cost of prescriptive compliance (IPRES) is also output.

Exhibit 3.5 Sample Fire Zone Summary Report

Part A: Printout

FIRE ZONE SUMMARY REPORT

FLOOR 3 ZONE 1

CONSTR	FINISH CON&EX	FINISH ROOMS	CORDOR PT&WLS	BUILDING SAFETY FEATURE NAME DOORS TO COR	ZONE DIMENS	VERTIC OPENNG	HAZARD AREAS	SMOKE CONTROL	EMERGE MOVMNT	MANUAL ALARM	SMOKE DET&ALM	SPRNKLR	CON S1	SURPLUS EXT S2	MOV S3	GEN SG	COST TO COMPLY
NCOM*R	AAAA	BBBB	GE 1HR	LT 20M	NO*100	ENC-1H	NO DEF	PARTTN	H*EXIT	FD CON	NONE	NONE	1	0	3	10	242844
NCOM*R	AAAA	BBBB	GE 1HR	LT 20M	DED*30	ENC-1H	NO DEF	PARTTN	H*EXIT	FD CON	NONE	NONE	1	0	1	8	56544
NCOM*R	AAAA	BBBB	GE 1HR	LT 20M	NO*150	ENC-2H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	NONE	3	0	0	7	199374
NCOM*R	AAAA	BBBB	GE 1HR	LT 20M	NO*100	ENC-1H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	NONE	1	0	0	7	214669
NCOM*R	AAAA	BBBB	GE 1HR	LT 20M	NO*100	ENC-1H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	NONE	1	0	1	8	214669
NCOM*R	AAAA	BBBB	GE 1HR	LT 20M	OED*30	ENC-2H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	NONE	3	0	0	7	13074
NCOM*R	AAAA	AAAA	GE 1HR	LT 20M	OED*30	ENC-2H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	NONE	5	0	0	9	13115
NCOM*R	AAAA	CCCC	GE 1HR	LT 20M	DED*30	ENC*2H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	NONE	0	0	1	4	13273
NCOM*R	AAAA	BBBB	GE 1HR	LT 20M	DED*30	ENC*2H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	NONE	4	0	1	8	13875
NCOM*R	AAAA	BBBB	GE 1HR	GE 20AC	DED*30	ENC-1H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	NONE	3	0	0	7	41249
NCOM*R	AAAA	BBBB	GE 1HR	GE 20M	OED*30	ENC-2H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	NONE	1	0	1	6	46805
NCOM*R	AAAA	AAAA	GE 1HR	LT 20M	NO*100	ENC-1H	NO DEF	PARTTN	H*EXIT	FD CON	CORHAB	NONE	3	4	7	16	307499
NCOM*R	AAAA	AAAA	GE 1HR	LT 20M	OED*30	ENC-1H	NO DEF	PARTTN	H*EXIT	FD CON	CORHAB	NONE	3	4	5	14	121199
NCOM*R	AAAA	AAAA	GE 1HR	LT 20M	NO*150	ENC-1H	NO DEF	PARTTN	W/OH*E	FD CON	CORHAB	NONE	3	4	2	11	258394
NCOM*R	AAAA	AAAA	GE 1HR	LT 20M	OED*30	ENC-1H	NO DEF	PARTTN	W/OH*E	FD CON	CORHAB	NONE	3	4	2	11	72095
NCOM*R	AAAA	CCCC	GE 1HR	LT 20M	NO*100	ENC-1H	NO DEF	PARTTN	H*EXIT	FD CON	NONE	CORHAB	5	8	7	14	380065
NCOM*R	AAAA	CCCC	GE 1HR	LT 20M	DED*30	ENC-1H	NO DEF	PARTTN	H*EXIT	FD CON	NONE	CORHAB	5	8	5	12	195763
NCOM*R	AAAA	CCCC	GE 1HR	LT 20M	NO*150	ENC-1H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	CORHAB	5	8	2	9	330956
NCOM*R	AAAA	CCCC	GE 1HR	LT 20M	OED*30	ENC-1H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	CORHAB	5	8	2	9	144656
NCOM*R	AAAA	AAAA	GE 1HR	LT 20M	NO*100	ENC-1H	NO DEF	PARTTN	H*EXIT	FD CON	TTLZON	NONE	3	5	8	17	365459
NCOM*R	AAAA	AAAA	GE 1HR	LT 20M	OED*30	ENC-1H	NO DEF	PARTTN	H*EXIT	FD CON	TTLZON	NONE	3	5	6	15	179159
NCOM*R	AAAA	AAAA	GE 1HR	LT 20M	NO*150	ENC-1H	NO DEF	PARTTN	W/OH*E	FD CON	TTLZON	NONE	3	5	3	12	316354
NCOM*R	AAAA	AAAA	GE 1HR	LT 20M	DED*30	ENC-1H	NO DEF	PARTTN	W/OH*E	FD CON	TTLZON	NONE	3	5	3	12	130054
NCOM*R	AAAA	CCCC	GE 1HR	LT 20M	NO*100	ENC-1H	NO DEF	PARTTN	H*EXIT	FD CON	NONE	TTLBLD	7	10	8	16	464588
NCOM*R	AAAA	CCCC	GE 1HR	LT 20M	DED*30	ENC-1H	NO DEF	PARTTN	H*EXIT	FD CON	NONE	TTLBLD	7	10	6	14	278288
NCOM*R	AAAA	CCCC	GE 1HR	LT 20M	NO*150	ENC-1H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	TTLBLD	7	10	3	11	415483
NCOM*R	AAAA	CCCC	GE 1HR	LT 20M	DED*30	ENC-1H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	TTLBLD	7	10	3	11	229183
NCOM*R	AAAA	BBBB	GE 1HR	GE 20M	NO*100	ENC-2H	NO DEF	PARTTN	W/OH*E	FD CON	NONE	NONE	4	0	3	11	254122
NCOM*R	AAAA	AAAA	GE 1HR	LT 20M	DED*30	ENC-1H	NO DEF	PARTTN	W/OH*E	FD CON	CORDOR	NONE	3	2	0	9	21646
NCOM*R	AAAA	BBBB	GE 1HR	LT 20M	DED*30	ENC*2H	NO DEF	PARTTN	DEFCAP	FD CON	CORDOR	NONE	4	2	1	8	28048
NCOM*R	AAAA	AAAA	GE 1HR	LT 20M	DED*30	ENC-1H	NO DEF	PARTTN	W/OH*E	FD CON	ROOMS	NONE	3	3	1	10	49984
NCOM*R	AAAA	AAAA	GE 1HR	NODOOR	DED*30	ENC*2H	NO DEF	PARTTN	H*EXIT	FD CON	CORHAB	CORHAB	2	12	2	13	260220
NCOM*R	AAAA	BBBB	GE 1HR	LT 20M	DED*30	ENC-1H	SGL IN	PARTTN	W/OH*E	FD CON	NONE	CORHAB	3	2	2	7	143652
NCOM*R	AAAA	AAAA	GE 1HR	LT 20M	OED*30	ENC-1H	SGL IN	PARTTN	W/OH*E	FD CON	CORDOR	CORHAB	5	4	3	11	157860
NCOM*R	AAAA	BBBB	GE 1HR	LT 20M	DED*30	ENC-1H	SGL IN	PARTTN	W/OH*E	FD CON	NONE	TTLBLD	5	4	3	9	228177

Exhibit 3.5 Sample Fire Zone Summary Report

Part B: Discussion

This report shows all of the distinct solutions generated for the fire zone under study; it is generated through a call to subroutine PRTZON. Upon entry to subroutine PRTZON, the ISRT matrix contains all distinct solutions sorted by design classification. A given row of the ISRT matrix, K, contains all data required to characterize the solution. The first descriptive line shows the fire zone location (ISRT(K,14) and ISRT(K,15). Each of the 13 building safety features are then listed followed by the 4 safety requirements. The estimated cost to comply is presented in the last column. The name of the post retrofit state (HSTAT(I), I=1,13) for each of the 13 building safety features are listed beneath the appropriate column heading. Each solution takes up one line of the output. The surpluses over the required score for containment, extinguishment, people movement and general safety are then recorded (ISRT(K,16),..., ISRT(K,19)). The order in which the solutions are output is based on the 40 design classifications (see table 2.1). All solutions are ranked and printed out in ascending order of cost (ISRT(K,22)). The program then outputs the solution(s) for the next design classification for which at least one solution was generated until the list is exhausted. The prescriptive solution is then output. Solutions which do not fit a design classification are then ranked and printed out according to whether they have no deficiencies, a single deficiency, or a double deficiency in hazardous areas.

TOTAL BUILDING SUMMARY REPORT

THE DESIGN VARIABLE QUALIFIERS FOR THIS CLASS ARE AS FOLLOWS

HAZARD AREAS	CONSTR	ZONE DIMENS	EMERGE MOVMNT	SMOKE DT+ALM	SPRNKLR
NO DEF	RESIST	DEO•3O	NO H•E	INPUT	INPUT

THE TOTAL COST OF RETROFITTING THE BUILDING FOR THIS DESIGN CLASSIFICATION IS: 30839
THE TOTAL COST OF PRESCRIPTIVE COMPLIANCE FOR THIS BUILDING IS: 302527

FL OR	Z N	CONSTR	FINISH COR+EX	FINISH ROOMS	CORDOR PT+WLS	DOORS TO COR	ZONE DIMENS	VERTIC OPENNG	HAZARD AREAS	SMOKE CONTROL	EMERGE MOVMNT	MANUAL ALARM	SMOKE DET+ALM	SPRNKLR	SURPLUS CN S1	SURPLUS EX S2	SURPLUS MV S3	SURPLUS GN SG	COST TO COMPLY
BUILDING SAFETY FEATURE NAME																			
-1	1	NCOM•R	AAAA	AAAA	GE 1HR	LT 20M	NO+15O	ENC-1H	NO DEF	PARTTN	W/DH•E	FD CON	NONE	CORHAB	11	8	2	19	5462
1	1	NCOM•R	AAAA	AAAA	GE 1HR	LT 20M	NO+15O	ENC-1H	NO DEF	PARTTN	W/DH•E	FD CON	NONE	NONE	5	0	0	9	4369
2	1	NCOM•R	AAAA	BBBB	GE 1HR	LT 20M	NO+15O	ENC-2H	NO DEF	PARTTN	W/DH•E	FD CON	NONE	NONE	3	0	0	7	7934
3	1	NCOM•R	AAAA	BBBB	GE 1HR	LT 20M	DEO•3O	ENC-2H	NO DEF	PARTTN	W/DH•E	FD CON	NONE	NONE	3	0	0	7	13074

Exhibit 3.6 Sample Total Building Summary Report

Part B: Discussion

This report shows the solution for the entire building based on a prespecified design classification; it is generated through a call to subroutine PRBLDG. Upon entry to subroutine PRBLDG, the ISTK matrix contains the least-cost solution for each of the 40 design classifications for each fire zone in the building. ISTK(K,20) is then checked to see if a given design classification was generated for all fire zones. If a perfect match occurs, the solution for the entire building is printed. The first descriptive heading shows the design variable qualifiers for this design classification (HOUT). The total cost of retrofitting the entire building to this design classification (KOST) is then given followed by the cost of prescriptive compliance (KPRES). Information on the location of the fire zone (ISTK(K,14) and ISTK(K,15)), the post retrofit state (HSTAT(I), I=1, 13) for each of the 13 building safety features, the surplus over each of the 4 safety requirements (ISTK(K,16),...,ISTK(K,19)), and the estimate cost to comply (ISTK(K,22)) are then printed out. Each fire zone takes up one line of output.

The remainder of this section will deal with error messages. In order to address the everyday problems of incorrect formatting, recording and sequencing, FSESCM has an elaborate system for edit-checking the values of key input variables. If an error is encountered, then a message is printed out to the user which should help to locate and correct the error. There are two basic types of error messages: (1) those which FSESCM treats as fatal for the building under analysis; and (2) those which result in the setting of default values. A third type of error relates to the message generated within the revised simplex (RVSMPX) subroutine. Since all entries to the constraint matrix are constructed within the MAIN program based on data from the SETUPA COMMON, no errors on input should affect the feasibilty of the application problem. The full range of RVSMPX error messages are included primarily as an aid to debugging should it be necessary to modify the FSESCM source code due to changes in the Life Safety Code or peculiarities of the operating system. The subroutine may also be uncoupled from the model for use as an optimizer in other applications.

The discussion of FSESCM generated error messages which follows is designed to be reasonably comprehensive but should not be considered exhaustive. The diagnostic associated with each error message is presented first. The routine which generated the message is then identified. The action taken by the program and the data output are then given. Recommendations for ways in which the problem may be corrected are also given.

It is important to point out that the discussion which follows is limited to first-order effects. If a non-fatal error (type 2), in which a default value is set, is followed by a fatal error (type 1), then both errors should be analyzed carefully for a possible relationship. In particular, a default setting to correct for an earlier error may result in a sequencing problem which triggers a fatal error. In every case where a fatal error is encountered, there should be sufficient information to easily locate the card which caused the problem. Non-fatal errors which do not require treatment should be subjected to close scrutiny to insure that the interpretation of the problem is not rendered meaningless.

Type 1 Errors

Diagnostic:	NO SATISFACTORY SOLUTIONS ENCOUNTERED
Routine:	MAIN
Action:	Analysis of current building terminates. Subroutine SEARCH is called to locate where data on the next building starts.
Data Output:	None
Remedy:	Output from RVSMPX is abnormal. Check application problem and input data.

Diagnostic:	SAFETY REQUIREMENT OUT OF RANGE
Routine:	MAIN
Action:	Analysis of current building terminates. Subroutine SEARCH is called to locate where data on the next building starts.
Data Output:	None
Remedy:	Check the value of the first integer variable, I1, after the command REQUIR on this control card. I1 must take on a value between 1 and 4.

Diagnostic: USER MOD INCORRECT
Routine: MAIN
Action: Analysis of current building terminates. Subroutine SEARCH is called to locate where data on the next building starts.
Data Output: Command on control card (HIN) plus option qualifiers (I1,I2,I3).
Remedy: Check the value of the command on this control card against those given in table 2.3.

Diagnostic: PARAMETER CARD ___ INCORRECT
Routine: MAIN
Action: Analysis of current building terminates. Subroutine SEARCH is called to locate where data on the next building starts.
Data Output: Building safety feature number as read.
Remedy: Check application problem and input data. An improper setting of the input state for a building safety feature can cause this problem. Check for missing cards.

Diagnostic: THE VALUES ON CARD ___ AFTER THE ERROR ARE: ___ ___
Routine: SEARCH
Action: Analysis of current building terminates. This subroutine searches through the data file for a LAST or FINAL command.
Data Output: Card number in sequence (1-20) and its values in alphanumeric (A6, A72) formats.
Remedy: Use output to locate the card which caused the error. Additional diagnostics will also be available.

Type 2 Errors

Diagnostic: AN ATTEMPT TO RELAX A SAFETY REQUIREMENT IS BEING MADE
Routine: MAIN
Action: None
Data Output: None
Remedy: None is required but it would be advisible to verify that relaxing (reducing) the safety requirement was the intent of the user.

Diagnostic: BUILDING SAFETY FEATURE OUT OT RANGE
Routine: MAIN
Action: The incorrect value is set to a default value of either 1 or 13.
Data Output: None
Remedy: Check the value on the first integer variable, I1, after the command CHANGE on this control card. It must take on a value between 1 and 13.

Diagnostic: RETROFIT STATE OUT OF RANGE
Routine: MAIN
Action: The incorrect value is set to a default value of either 1 or IT4M(I) where I is the building safety feature number.
Data Output: None
Remedy: Check the initial settings of the state variables read into IT4X(I). Check the value of the second integer variable, I2, after the command CHANGE on this control card. Both I2 and IT4X(I) must take on values between 1 and IT4M(I).

Diagnostic: BUILDING TYPE QUALIFIER OUT OF RANGE
Routine: MAIN
Action: The incorrect value is set to a default value of either 1 (hospitals) or 2 (nursing homes).
Data Output: None
Remedy: Check the value on the seventh card for this building in the data file. ITYPE must be either 1 or 2.

Diagnostic: BUILDING AGE QUALIFIER OUT OF RANGE
Routine: MAIN
Action: The incorrect value is set to a default value of either 1 (new single story) or 4 (existing multistory).
Data Output: None
Remedy: Check the value on the seventh card for this building in the data file. IBLDG must be between 1 and 4.

Diagnostic: PARTITION QUALIFIER OUT OF RANGE
Routine: CZODIM
Action: The incorrect value is set to a default value of 0.
Data Output: None
Remedy: Check the value on the first card for the sixth building safety feature (zone dimensions) for this fire zone. IPART must be either 0 or 1.

Diagnostic: DEFICIENCY QUALIFIER OUT OF RANGE
Routine: CHAZAR
Action: The incorrect value is set ot a default value of 0.
Data Output: None
Remedy: Check the value on the first card for the eighth building safety feature (hazardous areas) for this fire zone. Each of three deficiency qualifiers must take on values of either 0 or 1.

Diagnostic:	SPRINKLER DESIGNATION OUT OF RANGE
Routine:	CSPRNK
Action:	The incorrect value is set to a default value of 2 (wet concealed).
Data Output:	None
Remedy:	Check the value on the first card for the thirteenth building safety feature (sprinklers) for this fire zone. The allowable values are 1 through 4.

Diagnostic:	WATER SUPPLY DESIGNATION OUT OF RANGE
Routine:	CSPRNK
Action:	The incorrect value is set to a default value of 3 (unknown).
Data Output:	None
Remedy:	Check the value on the first card for the thirteenth building safety feature (sprinklers) for this fire zone. The allowable values are 1 through 3.

3.2 MODEL TEST RESULTS

The purpose of this section is twofold. First, it provides a description of activities carried out to insure model portability and summarizes run times experienced. Second, it includes a description of the validation activities used to insure that the model does in fact what it should do (as stated in the documentation). The second issue is included because model builders are becoming more aware of the need for a coherent means of validation. Such activities, especially with regard to third party assessments, are summarized in a series of papers dealing with energy models.[1] The discussion of model validation in the latter part of this section will draw on the information contained in the above-mentioned report.

The issue of model portability was addressed in two ways. First, the model was written in FORTRAN for which a nationally accepted standard exists. FSESCM was written in FORTRAN 77 and tested extensively against the ANSI standard.[2] The model is designed to be run on any system which: (1) complies with the X3.9 ANSI standard; and (2) can accomodate an intermediate-sized program. FSESCM complies with the full FORTRAN 77 standard. Compliance with the subset standard would require at a minimum the elimination of the BLOCK DATA routine. Since no testing to the subset standard was carried out,

---

[1] P.B. Saunders, editor, Selected Assessment Strategies Applied to Short-Term Energy Models, National Bureau of Standards, NBSIR 83-2672, March 1983.

[2] American National Standards Institute, American National Standard Programming Language FORTRAN, ANSI X3.9-1978, New York, 1978.

detailed specifications for compliance with the subset standard are not available from the model developers. Compliance with the subset standard may not prove to be a problem, however, since programmers familiar with the host operating system can take advantage of special features which may not be in the standard. Second, the model was tested independently on two types of hardware with different word lengths and memory capabilities. In all cases the solutions produced by the model were identical. In order to increase the likelihood that the model is portable, a test case with solutions for comparison is included with the source code whenever a request for the model is made.

A wide variety of test problems were run on a Sperry Univac 1100/82 and a Perkin-Elmer 3242. Both sets of runs were designed to test for portability and to provide information on the time it takes to analyze: (1) a single fire zone; (2) a single building; and (3) a batch of buildings. In estimating the run time on a machine, it is important to point out that the initial conditions of a building's fire zones must be critically analyzed. If all fire zones are in poor condition, then the optimizer is called more frequently and more alternative solutions are generated. Since the model uses an initial basis for all but the first call to the optimizer for a given fire zone, the subsequent calls will generate solutions quickly. Thus computation time for a particular fire zone should be nearly linear in the number of alternative solutions. The test case building contained four fire zones, two of which could be considered in poor condition.

All preliminary developmental work on the FSESCM model was done on a Sperry Univac 1100/82. This machine is the NBS mainframe computer; it has a 36-bit word. When the model was run on the Univac; it had a memory requirement of 43,000 words. The time to analyze a typical building on this machine ranged from 30 to 60 CPU seconds.

The Perkin-Elmer 3242 is a second generation mini computer; it is used by NBS's Center for Fire Research for a wide variety of applications. The Perkin-Elmer has a 32-bit word. When the model was run on the Perkin-Elmer, it had a memory requirement of 154 kilobytes. Two versions of the model were run on this machine: (1) a standard version which was compiled without options; and (2) an optimized version of the source code. For the first version, run times averaged approximately 3 CPU minutes. For the second version, run times ranged from 40 to 90 CPU seconds per building. Thus, if a mini computer is used and a code optimization option is available on the compiler, it may be wise to use it. This would especially be true if the system were supporting several users and a large number of buildings are to be batched together.

The validation of a complex model aims at demonstrating that the model bears a close resemblance to the physical system. The validation process is, in reality, three separate tasks: (1) technical validity; (2) operational validity; and (3) dynamic validity. Since program documentation provides a basis from which the validity of the model can be assessed, it will be discussed prior to the three major tasks of the validation process.

Documentation

From a model user's point of view, documentation (the written description of the model) is essential if the model is to be useable, useful, and used. Since the abstract model is a mathematical representation whereas the operational FSESCM model appears as a computer code, it is necessary to verify that there exists a unique relationship between the abstract (mathematical) model and the operational model. The relationship between the abstract and operational model can best be understood through reference to section 2.1. There it was shown how FSESCM was related to the FSES developed by NBS's Center for Fire Research. Documentation also requires that portability be established (not just that the program can be run on a variety of machines, but that it produces the desired result). Portability in that sense was the subject of the earlier discussion. Documentation also serves to explain all relevant relationships between inputs, outputs and analysis. The model's documentation also should provide some measure of user friendliness. Complex models often involve subtle techniques which, in the absence of a buffer between the user and the model, could cause frustration and lead to a highly inefficient use of the model. Documentation and an executive code (e.g., extensive edit-checking and message generating capabilities) should serve to shield the user from unnecessary detail without withholding any information which is essential to confidently use the model. The FSESCM model addresses this issue through reference to three reports: (1) an extended executive summary; (2) the User's Manual; and (3) the Programmer's Manual. Each report discusses a particular aspect of the model. These aspects are (1) management; (2) application; and (3) operation and maintenance. Each report is designed to be self contained. Where necessary, connections between the reports are given. The source code provides ample information to users so they can find and correct errors in their file. The source code also contains extensive comments and a glossary of terms for each routine, should it become necessary to make changes to the source code. This subject is covered in great detail in section 4.2.

Technical Validity

Technical validity requires the identification of all model assumptions, including those dealing with data requirements and sources. As a first step, one should identify all stated and implied assumptions, all decision variables, and any hypothesized relationship between variables. This step sheds light on the correspondence between the model and the real world phenomena it attempts to explain. Three types of assumptions may be readily defined. First, the mathematical assumptions include its functional form and the continuity of its relationships (e.g., the linearity of the model). A second type, content assumptions, define all model terms and variables. They should also define the scope and limitations of the model. (This topic will be discussed at the end of the chapter.) The final type, causal assumptions, are concerned with the assumed or hypothesized relationships between terms and variables (e.g., the way the safety scores are calculated).

Ideally, one would like to build a model which would produce true conclusions whenever all of the assumptions are true. To translate such abstract concepts into a form which is concrete and testable, it is necessary to:

(i) determine if the model's calculations are correct and accurate;

(ii) analyze if the logical flow of data and intermediate results are correct and consistent; and

(iii) ensure that variables and relationships have not been omitted.

Establishing the correctness and accuracy of the calculation and solution is directly related to the formulation of the problem. As mentioned earlier, the problem formulation is tied directly to the version of the FSES which was formally adopted into the Life Safety Code. The edit-checking routine which has been built into the model should ensure that infeasible problems do not reach the optimizer. The revised simplex (RVSMPX) routine has been thoroughly tested and used extensively for over ten years. It is therefore unlikely that it would produce incorrect or inaccurate solutions. The integerization routine (INTSOL) combines data from RVSMPX with a straight forward approach to impose integrality.

The second issue relates to the basic philosophy behind FSESCM. Since the objective is to minimize the cost of compliance, where some factors which affect the decision makers choice may not be known, it was necessary to define classes of solutions. This problem is approached by first generating a bounding solution for the fire zone. The bounding solution is then used as a starting basis for all subsequent calls to the optimizer where the alternate solutions are generated. Solutions are then assigned to a design classification and sorted. All solutions for a fire zone are then output. The best solution from each class is saved and the next zone is analyzed. This continues until all fire zones have been analyzed. All fire zones are then screened to see if a match within a design classification exists. If so, these solutions for the entire building are ranked by cost and output.

The third issue relates to the treatment of costing and interdependencies. Since the objective is cost minimization, it is essential that all costs be reasonably correct and accurate. Design specialists worked with the model builders to identify a candidate set of critical elements which span all possible transitions in FSES Table 4. This exercise resulted in a worksheet which is contained in the User's Manual along with directions for completing it and transferring the data to the user's data file. A general purpose or one of five special purpose subroutines, then calculate the transition costs for the building safety feature under consideration. The effect of interdependencies is assessed in subroutine INTSOL. Interdependencies are treated as part of the post-optimality analysis because including them in the application problem could have complicated the process of integerization. As such, larger surpluses may result. A series of tests is used to insure that no solution which violates a constraint (i.e., renders the problem infeasible) due to an interdependency can be passed to the packing routine (TNPSOL).

Operational Validity

Operational validity is concerned with whether or not the model can produce bad answers for proper ranges of parameter values (i.e., the model should be robust in that a user would find it difficult to make the model yield (in terms of the decision maker) an ostensibly wrong answer). Sensitivity analysis is related to, but distinct from, robustness. This technique seeks to systematically vary the values of the model parameters to determine how much (i.e., how sensitive) the solution changes. The state-forcing loops and the user options CHANGE and REQUIR address these issues.

The last aspect of operational validity and the most difficult is implementation validity. Implementation validity is concerned with the extent to which the real world system being modeled will respond in a manner indicated by the recommended solution. This task is difficult because if a decision maker knew how the system would respond to a given change in a parameter or decision variable, there would be considerably less need for a model. Implementation validity has been addressed in an informal matter. Since the model has not been released to the general public (at the time of this writing), it is impossible to state definitively whether its results will merely vindicate engineering judgment or provide genuine insight. Discussions between the model builders and a small group of design professionals, strongly indicate that the model will increase the productivity of the engineering staff by enabling them to carefully weigh the benefits of a large, but well defined, set of solutions.

Dynamic Validity

Dynamic validity is concerned with determining how the model will be maintained during its life cycle so it will continue to be an acceptable representation of the real system. The two aspects associated with dynamic validity are updating and review. Both subjects will be discussed in section 4.2. In updating, the person incorporating the changes needs to be satisfied that the model developers have established a procedure by which information is collected and analyzed to determine if and when model parameters or model structure needs to be changed. It is also important that a process exists by which such changes can be incorporated into the model and disseminated to users. A regular schedule for reviewing the success or failure of the model during its life cycle is also necessary. These reviews should be carried out regularly and should focus on documenting any systematic divergences between the solution predicted and the actual outcomes. The implications and means of accomplishing any proposed model changes should also be commented on.

Program Limitations and Qualifications

The program does have several limitations which may be important in certain instances. First, the costing procedure used in the model is limited to the costs of installing (including any demolition and removal costs) all possible combinations of the fire safety measures defined in FSES Table 4. Consequently, any costs which are not construction related (e.g., lost revenues, future operations and maintenance costs and insurance differentials)

are not included in the procedure. If these costs are deemed sufficiently important, a life-cycle cost analysis of the alternatives which result from this procedure should be performed. The accuracy of the costs presented by the model should be sufficient to discriminate among alternative solutions; however, these costs should not be used as a firm figure for actually carrying out the work.

Second, FSESCM does not contain a procedure for estimating the costs of mechanically assisted smoke control systems. Thus, if a transition to the mechanically assisted by zone state is desired, the user must input a cost estimate via the CHANGE option (see table 2.3). A similar limitation exists for the direct exit state for emergency movement routes. The user can, however, input a cost estimate by using the CHANGE option. If these modifications are deemed important, it would be desirable to prepare a special purpose subroutine to handle each one.

Next, there are two states listed in FSES Table 4 which the model treats as impossible; they are cases where a hazardous area has either a double or single deficiency outside the fire zone. These states are precluded because the entire cost of upgrading the deficiency is allocated to the fire zone where the deficiency occurs. If the costs can be satisfactorily allocated, say through a judicious change in subroutine CHAZAR, then no ambiguity would exist and these states can be analyzed as potential retrofits by the optimizer. No such allocation scheme was, however, evident to the builders of the FSESCM model.

Finally, given the current procedure for insuring the compatibility of a set of designs for the entire facility, a limitation of 10 fire zones for any one building is imposed. The maximum of 10 can be increased however, by merely increasing row dimension of the ISTK matric and resetting the upper limits on its associted do loops.[1] For some specific uses, the way in which alternative solutions are generated may not provide enough flexibility. In such a case, a modification to the program code will be required. Persons wishing to modify the way in which the alternative solutions are generated should carefully follow the directions given in section 4.2.

---

[1]ISTK is contained in the RESET COMMON storage block; it is passed as an argument from the MAIN program to the following subroutines: INSETS; STKSOL; and PRBLDG.

4. SOFTWARE EXCHANGE

4.1 TRANSFER MEDIUM AND CONTENTS OF FILES PROVIDED

The preferred transfer medium for the model is 9-track magnetic tape. All tapes provided by NBS will be written in either ASCII or EBCDIC. The request for the source code should specify which bit configuration is required for the user's operating system. All tapes provided by NBS will be unlabelled. The 2400 foot real will be recorded with a density of 1600 frames per inch (FPI). Since one of the files recorded on the tape is the test case output, a logical record length of 132 characters is used throughout. A fixed block length of 1320 characters is also used. Table 4.1 summarizes the information on magnetic tapes.

Table 4.1 Specifications for Transfer of 9-Track Unlabelled Magnetic Tapes

Characteristic	Specification
Bit configuration	ASCII or EBCDIC
Density	1600 FPI
Logical record length	132 characters
Block length	1320 characters

Software exists at NBS for producing either ASCII or EBCDIC punched card code. NBS will therefore provide a deck of punched cards if such a request is made. Since the deck will consist of only the source code and data files discussed below, punched cards should only be used when no compatible magnetic tape equipment is available on the user's operating system.

Whenever a request for the FSESCM model is made, the tape sent to the requestor will contain four files. These files contain: (1) background information; (2) the FSESCM source code; (3) a test case "User Data File"; and (4) the test case output file.

The first file contains a table of contents of the tape. It describes what is in each file and how the three remaining files relate to each other. It will also contain information on what portions of the model have been updated and the reason for the change since the publication of the User's Manual and Programmer's Manual.

The second file contains the FSESCM source code. The MAIN program and all subroutines are written into a single element. The file name is FSESCM; the element name is OPTI. The programmer is free to choose which compiler options are to be used. As noted earlier, it may be advisible to exercise the code optimization option if such an option is available on the host system.

The third file is a test case "User Data File". It contains a known correct set of data for use in initial testing on the host system. Once these tests have been performed, it is recommended that this file be provided to users so that they can check out their knowledge of the model with a known set of input data. Similarly, users can duplicate the data file and through an editor

introduce errors to test their troubleshooting skills. This file is thoroughly described in the User's Manual where it is used to illustrate how the model operates. Guidelines for constructing the file based on the FSESCM worksheets are also given.

The last file contains a complete set of solutions to the problem defined by the test case data. The primary purpose of this file is to insure model validity across operating systems. Two major points of comparison between this file and that generated by running the test case file are:

(1) the order and characterization of the solutions contained in the Total Building Summary Report; and

(2) the transition costs for each fire zone, especially with regard to their relationship with the Fire Zone Summary Report.

4.2 PROVISIONS FOR UPDATING OR MODIFYING THE SOURCE CODE

Although the procedure described in this report focuses on the 1981 edition of the Life Safety Code, it is natural to expect that some changes will occur as the code is periodically revised. Any changes which occur in future editions of the Life Safety Code will thus entail changes to the FSESCM computer program. The nature of the change will determine the ease with which program modifications can be made. At this time three types of changes are envisioned. In an increasing order of complexity these changes are: (1) increasing or decreasing one or more of the state values; (2) adding a new state to a given building safety feature; (3) adding a new building safety feature. (Although shortcuts may be possible, decisions to delete a state or a building safety feature would be of a similar level of complexity as those changes denoted as (2) and (3), respectively.) A fourth type of change is also possible; it relates to the way in which the design classifications are defined and generated. The complexity of this type of change is similar to the levels associated with types (1) and (2), respectively.

The design of the FSESCM model was kept deliberately simple. Where, in the opinion of the model developers, the potential for change existed, the clarity of the logic flow dominated all other considerations. This may have resulted in some reduction in computational efficiency, as indicated by the run time statistics from section 3.2, but since much of the packing and unpacking of solutions involves integer operations the tradeoff between efficiency and clarity seems justified. In the discussion which follows, it should be kept in mind that the model has already undergone one major change. This change was initiated because a preliminary version of the model had been coded to the 1973 edition of the Life Safety Code. This version of the Life Safety Code was used by Nelson and Shibe to illustrate the potential for the FSES. Since the 1981 edition of the Life Safety Code adopts a variant of the Nelson-Shibe FSES as an alternative to prescriptive compliance, it was decided to adapt the model to the official version. The incorporation of the 1981 edition of the Life Safety Code, along with a number of recommendations by design professionals, required the model builders to exercise almost the full spectrum of changes just described. The guidelines which follow can thus be considered reasonably comprehensive but should not be viewed as exhaustive.

Given a change in the Life Safety Code, the programmer is faced with two courses of action. The first would be to obtain an updated version of the model directly from NBS. If the changes in the Life Safety Code are substantial, this approach would seem preferable. The second would require a programmer to incorporate all model changes on the user's system. Should this approach be adopted, the information in tables 4.2 and 4.3 should be studied carefully prior to making any changes. Table 4.2 lists all major arrays and shows for each type of change whether that variable will be affected (yes), may be affected (?), or will not be affected (no). Table 4.3 provides the same information for each subroutine. This information is supplemented by the glossary of terms (appendix A.2) where all subroutines within which a given array appears are listed. It is important to note that the changes in the RVSMPX routine are limited to the sizes of arrays as defined in the DIMENSION statement. No other changes should be required. In RVSMPX, as in all other subroutines, the arrays are fully dimensioned. This approach was taken so that operating system debugging options could be employed as an integral part of the change to insure that such inconsistencies as attempting to reference a non-existent element in an array due to an incorrect setting on a DO loop could be easily detected.

In interpreting the information in tables 4.2 and 4.3, two points should be kept in mind. First, the modular design of the FSESCM model reduces most subroutines to 100 lines or less. Thus areas where changes are needed (or where errors occur) should be easy to find. Five routines are well over 100 lines; they are: (1) MAIN; (2) BLOCK DATA; (3) RVSMPX; (4) INTSOL; and (5) STKSOL. It is worth noting however that the changes to be made in the BLOCK DATA routine should be apparent from the context and any changes to RVSMPX are minimal. The three remaining routines will require some preplanning and attention to detail. Second, the responses in the table serve to focus attention on general topics. For certain specific cases, the changes may be significantly more complex than those of a general nature. This is why so many arrays and subroutines have question marks (?) associated with them indicating that a change may be required. In approaching these problem areas, as well as for the general case, one should pay special attention to:

(i) ranges on DO loops;

(ii) tests for branching based on logical IF statements, and

(iii) resetting of values within an array.

Referring now to table 4.2, the first column shows the variable name and current size, the second column its type, and the third through sixth columns characterize the nature of the change.[1]

---

[1]All information contained in tables 4.2 and 4.3 are with respect to FSES Tables 4 through 6. If modifications to FSES Tables 1 through 3 are required, then the problem is much simpler. all data for this situation relate to: HT1; IT1M; IT1X; and T1. The only routines which may require changes are therefore MAIN and BLOCK DATA.

The first type of change, modifying one or more state values, is the easiest to incorporate. Changes will definately have to be made to the G matrix, the analogue of FSES Table 4. However, a number of other changes may be required should this modification affect the computation of one or more of the safety requirements (FSES Table 6). Modifying a state value may also change the ordering for the states for building safety feature 1 or 8. It is also necessary to check if any interdependencies among building safety features are affected and if so adjust them to reflect the new value(s).

If a state is added (subtracted), then all changes described earlier would be in effect, there would also be a need to redimension certain arrays. Examples of arrays which require redimensioning include: A; C; HLAB; and IST4. The indices of certain DO loops will also be affected. For example, the index on the second state-forcing loop in the MAIN program will have to be increased (decreased). If a new interdependency comes into play (or an existing one requires modification), then logic must be incorporated to insure that all penalties or bonuses are properly recorded.

The addition (deletion) of a building safety feature is the most complicated change envisioned. This modification affects FSES Tables 4, 5, and 6. It incorporates all changes discussed so far. It also affects the first two state forcing loops and may have a significant impact on the third (design classifications). Many arrays will have to be redimensioned, including: IPAC; ISRT; and ISTK. The use of a debugging option is recommended.

A modification of the way the design classifications are generated is limited to several well-defined areas of seven subroutines. As a first step, the major arrays (HOUT, IEC1 and IEC5) in the BLOCK DATA routine must be reconstructed. Depending on the outcome of this exercise, the third state-forcing loop in the MAIN program will have to be revised. The assignment of solutions to the various design classifications made in subroutine DESCLS will have to incorporate this new logic. The same is true for the way in which individual fire zones are sorted (SRTSOL), printed in the Fire Zone Summary Report (PRTZON), and stacked (STKSOL). Finally, the logic for sorting and printing the Total Building Summary Report (subroutine PRBLDG) will have to be modified.

Table 4.3 summarizes the information from the previous text and table 4.2 showing how it affects each of the routines. As mentioned earlier, the nature of the change will dictate how closely the routine must be scrutinized prior to any attempt to modify the code.

Table 4.2 Source Code Modification for Key Variables

ARRAY SPECIFICATION		TYPE OF CHANGE			
NAME OF ARRAY	TYPE OF ARRAY	STATE VALUE	ADD/ SUBTRACT STATE	ADD/ SUBTRACT BUILDING SAFETY FEATURE	DESIGN CLASSIFICATION
A(18,61)	REAL	NO	YES	YES	NO
B(19,19)	REAL	NO	NO	YES	NO
C(56)	REAL	NO	YES	YES	NO
CPUL(13,7,10)	REAL	NO	YES	YES	NO
CPUM(13,7,10)	REAL	NO	YES	YES	NO
CPUT(13,7,10)	REAL	NO	?	YES	NO
G(13,7)	REAL	YES	YES	YES	NO
HLAB(57)	CHARACTER*7	NO	YES	YES	NO
HOUT(40)	CHARACTER*45	NO	?	?	YES
HPAR(18)	CHARACTER*30	NO	NO	YES	NO
HSTAT(13)	CHARACTER*30	NO	NO	YES	NO
IEC1(40)	INTEGER	NO	?	?	NO
IEC5(40,5)	INTEGER	NO	?	?	YES
IECCON(4,7)	INTEGER	NO	?	?	NO
IECDBL(3,5)	INTEGER	NO	?	?	NO
IECHAZ(3,5)	INTEGER	NO	?	?	NO
IECNOS(3,5)	INTEGER	NO	?	?	NO
IECSMO(5,5)	INTEGER	NO	?	?	NO
IECSPR(7,3)	INTEGER	NO	?	?	NO
IECT(44)	INTEGER	NO	?	?	YES
IECYES(3,5)	INTEGER	NO	?	?	NO
IECZDM(7,6)	INTEGER	NO	?	?	NO
INELT(7,10)	INTEGER	NO	?	?	NO
IPAC(100,22)	INTEGER	NO	?	YES	?
ISRT(100,22)	INTEGER	NO	?	YES	?
IST4(56,2)	INTEGER	NO	YES	YES	NO
ISTAT(13)	INTEGER	NO	NO	YES	NO
ISTK(500,22)	INTEGER	NO	?	YES	?
IT4A(7)	INTEGER	NO	?	?	NO
IT4B(7)	INTEGER	?	?	?	NO
IT4M(13)	INTEGER	NO	YES	YES	NO
IT4S(13,2)	INTEGER	NO	YES	YES	NO
IT4X(13)	INTEGER	NO	NO	YES	NO
IT4XA(13)	INTEGER	NO	NO	YES	NO
IT8(5)	INTEGER	?	?	?	NO
JPR(13,8)	INTEGER	NO	?	YES	NO
KOST(41)	INTEGER	NO	?	?	YES
LCU(13)	INTEGER	NO	?	YES	NO
T48A(5)	REAL	?	?	?	NO
T4A(4,7)	REAL	?	?	?	NO
T4B(4,7)	REAL	?	?	?	NO

Table 4.2 Source Code Modification for Key Variables (Continued)

ARRAY SPECIFICATION		TYPE OF CHANGE			
NAME OF ARRAY	TYPE OF ARRAY	STATE VALUE	ADD/ SUBTRACT STATE	ADD/ SUBTRACT BUILDING SAFETY FEATURE	DESIGN CLASSIFICATION
T6A(3,2,2)	REAL	?	?	YES	NO
T6B(3,2,2)	REAL	?	?	YES	NO
TEST(7,12)	REAL	NO	NO	?	NO
X(150)	REAL	NO	YES	YES	NO
Y(150)	REAL	NO	YES	YES	NO

Table 4.3 Source Code Modifications by Routine

	TYPE OF CHANGE			
NAME OF ROUTINE	STATE VALUE	ADD/SUBTRACT STATE	ADD/SUBTRACT BUILDING SAFETY FEATURE	DESIGN CLASSIFICATION
MAIN	?	YES	YES	YES
BLOCK DATA	YES	YES	YES	YES
CCONST	NO	YES	YES	NO
CHAZAR	NO	YES	YES	NO
COSEST	NO	YES	YES	NO
COSMOD	NO	?	YES	NO
CSMOKE	NO	YES	YES	NO
CSPRNK	NO	YES	YES	NO
CZODIM	NO	YES	YES	NO
DESCLS	NO	?	YES	YES
INSETS	NO	YES	YES	NO
INTSOL	?	YES	YES	NO
PCOSTS	NO	YES	YES	NO
PRBLDG	NO	YES	YES	YES
PRESOL	NO	YES	YES	NO
PRTBAC	NO	NO	NO	NO
PRTZON	NO	YES	YES	YES
RVSMPX	NO	YES	YES	NO
SEARCH	NO	NO	NO	NO
SRTSOL	NO	?	YES	YES
STKSOL	NO	YES	YES	YES
TNPSOL	NO	YES	YES	NO

APPENDIX A PROGRAM DOCUMENTATION

A.1 RELATIONSHIP TO THE FEDERAL INFORMATION PROCESSING STANDARDS

The three reports which describe the FSESCM model are patterned after recommendations given in the Federal Information Processing Standards (FIPS) Publication 38[1] and NBS Special Publication 500-73[2]. The information contained in these documents permits one to define four general classes of publications. These four classes are associated with the programming and test stages of the software life cycle as defined in FIPS Publication 38. The four general classes of publications are:

(1) Management Summary Manual;
(2) User's Manual;
(3) Programmer's Manual; and
(4) Analyst's Manual.

The first manual is designed as a management tool. It provides the information necessary to assess the model's input requirements (including time, money, and other resources) and the usefulness of the model's results. The Management Summary Manual focuses on how the model can facilitate the decision making process rather than the specifics of how to set up and run the model. The FSESCM equivalent of a Management Summary Manual is A Cost-Conscious Guide to Fire Safety in Health Care Facilities.[3]

The second manual is designed as a reference document for a nonprogramming model user. Information contained in this manual is similar to the first but with increased emphasis on detail. In-depth discussions of the following topics are included: the model's logical structure; the input data requirements; the results produced by the model; and the use of the model's results. The FSESCM equivalent of such a document is User's Manual for the Fire Safety Evaluation System Cost Minimizer Computer Program.[4]

---

[1]National Bureau of Standards, Guidelines for Documentation of Computer Programs and Automated Data Systems, FIPS Publication 38, 1976.

[2]National Bureau of Standards, Computer Model Documentation Guide, NBS Special Publication 500-73, 1981.

[3]Robert E. Chapman, A Cost-Conscious Guide to Fire Safety in Health Care Facilities, National Bureau of Standards, NBSIR 82-2600, 1982.

[4]Robert E. Chapman and William G. Hall, User's Manual for the Fire Safety Evaluation System Cost Minimizer Computer Program, National Bureau of Standards, NBSIR (in preparation).

The third and fourth documents are designed for use by programmers and analysts, respectively. The third manual provides guidelines for maintaining and modifying the model. These guidelines should be of sufficient detail to enable the programmer to understand the operation of the model and to trace through it for debugging, for making modifications, and for determining if and how the model can be converted to other computer systems. The fourth manual differs from the third in that its emphasis is on the model's functional structure, the algorithms used, and the techniques employed for model verification and validation. This report is the FSESCM equivalent of the Programmer's and Analyst's Manuals described above.

A.2 VARIABLE DEFINITION

This section serves as a glossary of terms. It is designed to be both informative and a source of information should the code require updating or modification. The glossary is arranged in alphabetical order by variable name. Each array is dimensioned as it is currently in the model. The purpose of the array is then given. The rows and columns of each array are then defined. Associated with each variable is a list of all subroutines which reference it.

A(18,61) — Purpose: A is the constraint matrix augmented by a right-hand-side column and an objective function row. The first 13 rows are generalized upper bound constraints with right hand sides of 1. Rows 14 through 17 are greater than or equal to constraints associated with the safety requirements. The last row contains the transition costs. Columns 1 through 56 are the states of the system. Columns 57 through 60 are surplus variables associated with rows 14 through 17. Column 61 is the right hand side. The A matrix is constructed within the MAIN program.
Routines: MAIN, RVSMPX, INTSOL, PRESOL.

B(19,19) — Purpose: Upon exit from RVSMPX, the B matrix contains: the inverse of the matrix of the current basic columns; the negative of the current values of the dual variables (row 18 columns 1 through 17); the current values of the basic variables (column 19 rows 1 through 17); the negative of the value of the real objective function B(18,19); and the negative of the value of the artificial objective function (B(19,19)).
Routines: MAIN, RVSMPX.

C(56) — Purpose: This vector contains the estimated transition cost for each of the 56 possible retrofit states. It is used to reinitialize the objective function row of the A matrix during the state-forcing loops. It is used extensively for costing and checking.
Routines: MAIN, INSETS, CCONST, CZODIM, CHAZAR, CSMOKE, CSPRNK, COSEST, PCOSTS, INTSOL, STKSOL.

CMODL — Purpose: This input variable is used to update the labor costs contained in the CPUL array to reflect regional variations in the demand for labor.
Routines: MAIN, COSMOD, PCOSTS.

CMODM — Purpose: This input variable is used to update the material costs contained in the CPUM array to reflect regional variations in the demand for building materials.
Routines: MAIN, COSMOD, PCOSTS.

CMODT — Purpose: This input variable is used to update the per unit costs in the CPUL and CPUM arrays to refect cost growth over time.
Routines: MAIN, COSMOD, PCOSTS.

CPUL(13,7,10) — Purpose: CPUL contains the per unit labor costs prevailing in the Washington, D. C., metropolitan area during the summer of 1981. The leading dimension is the building safety feature number. The middle dimension is the number of the state (rank) according to the appropriate row of FSES Table 4. The last dimension is the critical element which would be retrofitted. All values are initialized within the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA, COSMOD.

CPUM(13,7,10) — Purpose: CPUM contains the per unit material cost prevailing in the Washington, D. C. metropolitan area during the summer of 1981. The dimensions have the same interpretation as CPUL. All values are initialized within the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA, COSMOD.

CPUT(13,7,10) — Purpose: CPUT contains the total per unit cost for each potential retrofit. The values have been adjusted to reflect regional price differences in the labor and materials market as well as cost growth over time. The dimensions of the array have the same interpretation as CPUL and CPUM.
Routines: MAIN, COSMOD, CCONST, CZODIM, CHAZAR, CSMOKE, CSPRNK, COSEST.

G(13,7) — Purpose: G is the program analogue of FSES Table 4. The rows of G correspond to the building safety feature numbers in FSES Table 4. The columns correspond to the state number (rank) within a given row of FSES Table 4.
Routines: MAIN, BLOCK DATA.

HBLDG(6)
CHARACTER*40 — Purpose: This input variable is used as a mailing label and provides information of whom to contact should problems arise. The rows refer to: (1) the name of the facility; (2) the name of the building; (3) the name of the contact; (4) the street address; (5) the city, state and zip code; and (6) the telephone number.
Routines: MAIN, PRTBAC.

HFIND
CHARACTER*6

Purpose: This variable contains the first six characters on on a card read after a fatal error has occurred. The value is checked against the LAST and FINAL commands to locate the last card for the building or the batch.
Routine: SEARCH

HIN
CHARACTER*6

Purpose: This input variable is used for declaring which options are to be exercised.
Routines: MAIN.

HLAB(56)
CHARACTER*7

Purpose: This array contains the state names for each of the 56 possible retrofits.
Routines: MAIN, BLOCK DATA, CCONST, CZODIM, CHAZAR, CSMOKE, CSPRNK, PCOSTS, PRTZON, PRBLDG.

HOUT(40)
CHARACTER*45

Purpose: This array contains the design variable qualifiers for each of the 40 design classifications. All values are initialized within the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA, PRBLDG.

HPAR(18)
CHARACTER*30

Purpose: This array contains the labels for the rows (building safety features) shown in FSES Table 4.
Routines: MAIN, BLOCK DATA, CCONST, CZODIM, CHAZAR, CSMOKE, CSPRNK, PCOSTS, PRTZON, PRBLDG.

HSTAT(13)
CHARACTER*7

Purpose: This array is initialized from the settings of HLAB which correspond to the state within each of the 13 building safety features which is in the solution.
Routines: PRTZON, PRBLDG.

HT1(10)
CHARACTER*8

Purpose: This array contains the labels for FSES Table 1. All values are initialized within the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA.

HTEST
CHARACTER*72

Purpose: This variable is used to output information on the contents of those cards which immediately follow a fatal error.
Routines: SEARCH.

IB(5)
CHARACTER*6

Purpose: This character array is used for outputting information from the RVSMPX routine.
Routines: RVSMPX.

IBLDG

Purpose: This input variable specifies whether the building is: (1) new single story; (2) new multistory; (3) existing single story; or (4) existing multistory.
Routines: MAIN.

IEC1(40)

Purpose: This array contains the integer value of each design classification's set of design variable qualifiers. This value is used in testing, assigning and matching design equivalent solutions within and among fire zones.
All values are initialized within the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA, DESCLS, SRTSOL, STKSOL, PRBLDG.

IEC5(40,5) Purpose: This array contains a set of flags which are used to check if a perfect match between a solution and a design classification exists. The five columns correspond to: (1) construction; (2) zone dimensions; (3) emergency movement routes; (4) somke detection and alarm; and (5) sprinklers. All values are initialized within the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA, DESCLS.

IECCON(4,7) Purpose: This array is used to store the inputs for the critical element counts (rows) requiring treatment in order to retrofit one of the construction states (columns).
Routines: CCONST.

IECDBL(3,5) Purpose: This array is used to store the input values for the critical element counts (rows) in high hazard areas requiring treatment in order to move to a higher state (columns).
Routines: CHAZAR.

IECHAZ(3,5) Purpose: This array is used to store the sum across all three possible input categories for each critical element (row) in hazardous areas. The column dimension refers to the state within this building safety feature.
Routines: CHAZAR.

IECNOS(3,5) Purpose: This array is used to store the input values for the critical elements counts (rows) for non-sprinklered single deficiency hazardous areas. The column dimension refers to the state within this building safety feature.
Routines: CHAZAR.

IECSMO(5,5) Purpose: This array is used to store the input values for all critical element counts for smoke detection and alarm (the twelfth building safety feature). The column dimension refers to the state within this building safety feature.
Routines: CSMOKE.

IECSPR(7,3) Purpose: This array is used to store the input values for the critical element counts (rows) associated with automatic sprinkler systems. The column dimension refers to the state within the building safety feature.
Routines: CSPRNK.

IECT(44) Purpose: This array is used to store the number of distinct solutions for each of the 40 design classifications, the prescriptive compliance solution, and the three types of solutions which do not fit one of the above cases.
Routines: MAIN, INSETS, DESCLS, SRTSOL, PRTZON.

IECYES(3,5) Purpose: This array is used to store the input values for the critical element counts (rows) for sprinklered single deficiency hazardous areas. The column dimension refers to the state within the building safety feature.
Routines: CHAZAR.

IECZDM(7,6) Purpose: This array is used to store the input values for the critical element counts (rows) associated with zone dimensions (the sixth building safety feature). The column dimension refers to the state within the building safety feature.
Routines: CZODIM.

IFLOR Purpose: This input variable records the floor number of the fire zone currently under analysis.
Routines: MAIN, PCOSTS, INTSOL, TNPSOL, PRESOL.

INELT(7,10) Purpose: INELT is used to store the input values for the critical element counts (columns) associated with each building safety feature for which a special purpose routine was not written. The row dimension refers to the state within the building safety feature.
Routines: MAIN, COSEST

INV Purpose: INV is a switch that allows the user to provide RVSMPX with an initial basis. If INV is non-zero, an initial basis is expected. If INV is zero, the subroutine will start from scratch. Except for the first call to RVSMPX, an initial basis is supplied.
Routines: MAIN, RVSMPX.

IP(5) Purpose: IP stores the numbers of the variables which are entering and leaving the basis. The generation of a basis print is governed by the settings in the L vector.
Routines: RVSMPX.

IPAC(100,22) Purpose: IPAC is a multipurpose array. It is used for intermediate storage of solutions generated by the INTSOL routine. Each row of IPAC is a distinct solution. Columns 1 through 13 contain the number of the post retrofit state within each building safety feature, columns 14 and 15 contain the floor number and zone number columns 16 thorough 19 contain the surplus over each of the four safety requirements. Column 22 contains the post-retrofit cost. Columns 20 and 21 are left blank until entry into the DESCLS and SRTSOL routines. Upon exit they are design classification number and the rank according to cost within that design classification respectively.
Routines: MAIN, INSETS, INTSOL, TNPSOL, PRESOL, DESCLS, SRTSOL, STKSOL.

IPART Purpose: This input serves as a partition qualifier. If the input value is 0 then partitions can not be installed to reduce the length of the fire zone. If the input value is 1, then partitions can be installed. The number of partitions is dependent on the length of the fire zone, LZONE.
Routines: MAIN, CZODIM, INTSOL

IPATS — Purpose: This input variable tells how many patients are in the fire zone under consideration. It is used in calculating the score for the no control state of the ninth building safety feature (smoke control).
Routines: MAIN

IPRES — Purpose: This variable is used to store the cost of the prescriptive compliance solution.
Routines: MAIN, PCOSTS, PRESOL, STKSOL.

ISRT(100,22) — Purpose: ISRT contains the solutions for each fire zone sorted within each design classification. Each solution occupies one row. The columns have exactly the same meaning as for IPAC.
Routines: MAIN, INSETS, SRTSOL, PRTZON, STKSOL.

IST4(56,2) — Purpose: IST4 provides a crosswalk between the number of the simplex variable (row J, column 1) and the row of FSES Table 4 (row J, column 2). All data are initialized in the BLOCK DATA Routine.
Routines: MAIN, BLOCK DATA.

ISTAT(13) — Purpose: ISTAT is a multipurpose array. Within MAIN, CCONST, CZODIM, CHAZAR, CSMOKE, CSPRNK, ISTAT is used as a flag for input errors. Within TNPSOL, ISTAT is used to set flags to identify the prescriptive compliance solution.
Routines: MAIN, INSETS, CCONST, CZODIM, CHAZAR, CSMOKE, CSPRNK, INTSOL, TNPSOL.

ISTK(500,22) — Purpose: ISTK contains the best solutions for each fire zone for each design classification which had one or more solutions. Solutions as stored in blocks by fire zone. The columns of ISTK have the same interpretation as for IPAC and ISRT.
Routines: MAIN, INSETS, STKSOL, PRBLDG

IT1M(5) — Purpose: IT1M defines the maximum state number for each of the five occupancy risk factors shown in FSES Table 1. These values are used to check against the inputs IT1X and as default values should the maximum be exceeded. All values are initialized in the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA.

IT1X(5) — Purpose: IT1X contains the input values of the five occupancy risk parameters for FSES Table 1.
Routines: MAIN

IT4A(7) — Purpose: IT4A provides a crosswalk between T4A (the construction portion of FSES Table 4) and T4B (the construction portion of FSES Table 4 arranged as an increasing function of score). It is used as a crosswalk between IT4X and IT4XA in setting up the simplex tableau. All values are initialized in the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA.

IT4B(7)

Purpose: IT4B provides a crosswalk between T4B (the construction portion of FSES Table 4 arranged as an increasing function of score) and T4A (the construction portion of FSES Table 4). All values are initialized in the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA, PRTZON.

IT4M(13)

Purpose: This array contains the maximum state number within each of the 13 building safety features. It is used as a check against the input state numbers (IT4X). All values are initialized in the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA.

IT4S(13,2)

Purpose: IT4S provides a crosswalk between the simplex tableau (A) and FSES Table 4. The rows of IT4S correspond to the 13 building safety features. The first column of IT4S contains the first simplex variable number for a given building safety feature. The second column contains the last simplex variable number for a given building safety feature. All values are initialized in the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA, PCOSTS, INTSOL, TNPSOL,PRESOL, PRTZON, PRBLDG.

IT4X(13)

Purpose: IT4X stores the input state numbers for each building safety feature. The value input should reflect the current condition of that building safety feature.
Routines: MAIN, CCONST, CZODIM, CHAZAR, CSMOKE, CSPRNK, PCOSTS.

IT4XA(13)

Purpose: This array is used to define the input states for each of the 13 building safety features when setting up the simplex tableau. IT4XA differs from the user input array IT4X in that all state numbers are arranged so that their scores are monotonically increasing.
Routines: MAIN, INTSOL, PRESOL, DESCLS.

IT8(5)

Purpose: IT8 provides a crosswalk between IT4X and IT4XA for hazardous areas. All values are initialized in the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA, PRTZON.

ITYPE

Purpose: This input variable defines whether the facility under analysis is a hospital (ITYPE = 1) or a nursing home (ITYPE = 2). It is later used to define whether T6A or T6B should be used and which rows of the prescriptive requirement matrix (JPR) should be referenced.
Routines: MAIN.

IZONE

Purpose: This input variable qualifies the zone under analysis. It is essential if more than one fire zone exists on a floor.
Routines: MAIN, PCOSTS, INTSOL, TNPSOL, PRESOL.

JPR(13,8)

Purpose: JPR defines the number of the prescriptive compliance state for each of the 13 building safety features. The first four columns are for hospitals; the last four columns are for nursing homes. All values are initialized in the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA, CCONST, CZODIM, CHAZAR, CSMOKE, CSPRNK, PCOSTS, PRESOL, DESCLS.

JPRES

Purpose: This variable defines the column of the JPR matrix which contains the prescriptive compliance state numbers.
Routines: MAIN, CCONST, CZODIM, CHAZAR, CSMOKE, CSPRNK, PCOSTS, PRESOL, DESCLS.

K1, K2, K3

Purpose: The interpretation of these variables is limited to the loop where critical element counts are input. Within that loop, K1 is the length of the cost vector for the building safety feature under analysis, K2 is the number of the state above the one input, and K3 is the maximum state number. This information is passed to the COSEST subroutine where transition costs are estimated.
Routine: MAIN, COSEST.

KOST(41)

Purpose: KOST contains the cost of retrofitting the entire building to each of the 40 design classifications; the last row contains the cost of prescriptive compliance. KOST is used to govern the order in which design classifications are output.
Routines: PRBLDG.

KPRES

Purpose: This variable is used to store the cost of prescriptive compliance. It is used as a baseline against which the savings due to the FSES can be compared.
Routines: MAIN, PCOSTS, STKSOL, PRBLDG.

KQP

Purpose: KQP is a switch which allows the RVSMPX subroutine to solve quadratic programming problems. KQP is preset within the MAIN program to indicate that the application problem is linear.
Routines: MAIN, RVSMPX.

L(150)

Purpose: L is a multipurpose array. Its major purpose is to preset switches which define the constraint structure and output options for the RVSMPX subroutine. Values are preset within the MAIN program. Within subroutine PCOSTS, L is used to store the integerized values of the retrofit costs. Within subroutines INTSOL, TNPSOL and SRTSOL, L is used for integer manipulations.
Routines: MAIN, INSETS, PCOSTS, RVSMPX, INTSOL, TNPSOL, SRTSOL.

LCV(13)

Purpose: LCV records the length of the cost vector for each of the 13 building safety features. It is used for reading in the critical element counts. All values are initialized in the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA.

LZONE

Purpose: LZONE as input determines the length of the fire zone. It is used to calculate how many partitions would be needed to reduce the overall zone length.
Routines: CZODIM.

MA

Purpose: MA is the row dimension of the simplex tableau A. It is preset within the MAIN program.
Routines: MAIN, RVSMPX.

MB

Purpose: MB is the row dimension of the basis inverse matrix B. It is preset within the MAIN program.
Routines: MAIN, RVSMPX.

MT

Purpose: MT is the number of rows of information in the A matrix. MT is preset within the MAIN program.
Routines: MAIN, RVSMPX.

NT

Purpose: NT is the number of columns in the A matrix. NT is preset within the MAIN program.
Routines: MAIN, RVSMPX.

PR(5)

Purpose: PR contains the values of the basic variables within the Basis Print Report of subroutine RVSMPX.
Routines: RVSMPX.

T1(5,5)

Purpose: T1 contains values of the occupancy risk factors defined in FSES Table 1. All values are initialized in the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA.

T48A(5)

Purpose: T48A contains the scores for the five hazardous area states in increasing order. The crosswalk from FSES Table 4 and T48A and vice versa is made through reference to IT8. All values are initialized in the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA.

T4A(4,7)

Purposes: T4A contains the scores for the four floor categories associated with the first building safety feature (construction). T4A is identical with that portion of FSES Table 4 and is used in the output of the appropriate FSES tables. All values are initialized within the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA.

T4B(4,7)

Purpose: T4B contains the same type of information as T4A but with its states reorganized so that their values are monotonically increasing. IT4A is used as a crosswalk between T4A and T4B. IT4B is used as a crosswalk between T4B and T4A. All values are initialized within the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA.

T6A(3,2,2)

Purpose: This array contains the safety requirements for hospitals. T6A(I,J,K) may be interpreted as follows. K=1 implies a first floor fire zone; K=2 implies above the first floor. J=1 implies new; J=2 implies an existing facility. I=1 implies containment; I=2 implies extinguishment; and I=3 implies people movement. All variables are initialized in the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA.

T6B(3,2,2)

Purpose: This array contains the safety requirements for nursing homes. All indices have the same meaning as under T6A. All variables initialized in the BLOCK DATA routine.
Routines: MAIN, BLOCK DATA

TEST(7,12)

Purpose: TEST is used to store intermediate data during the integerization process.
Routines: INTSOL.

TOLP

Purpose: TOLP is used in the construction of epsilon (EPS) and capital epsilon (CEPS) in the RVSMPX routine. These variables are used to test tolerances during optimzation. TOLP is preset in the MAIN program.
Routines: MAIN, RVSMPX.

X(150)

Purpose: X is the solution vector which results from the call to RVSMPX. Upon exit from RVSMPX the vector X contains: the terminal values of the variables in the original problem in X(1) through X(56); the surplus over each of the four safety requirements in X(57) through X(60); the artificial variables for the 17 equations in X(61) through X(77); and the negative value of the objective function in X(78). After the first call, X has its values read into Z. Prior to each call in the state-forcing loops, X is reinitialized with the values from Z so that an initial basis is provided to RVSMPX.
Routines: MAIN, INSETS, RVSMPX, INTSOL.

Y(150)

Purpose: Y is a multipurpose working solution vector. The solution stored in X is read into Y, all manipulations (testing, integerizing, packing) are then based on Y.
Routines: MAIN, INSETS, RVSMPX, INTSOL, TNPSOL.

Z(150)

Purpose: Z contains the optimal solution to the continuous linear program. It is used to restore the X vector so that RVSMPX may be entered with an initial basis.
Routines: MAIN, INSETS, PRTZON.

A.3 DESCRIPTION OF THE SUBROUTINES

This section provides a brief description of all subroutines in the model. For the MAIN program, a more detailed description by functional block is given in chapter 2. Similarly, the specification program (BLOCK DATA) associated with the initialization of all variables in the SETUPA and SETUPB COMMON blocks is described in section 2.2.

The section is arranged in alphabetical order by subroutine name. Each subroutine is described on a single program summary sheet. This summary sheet includes: (a) the name of the subroutine; (b) the call statement; (c) a narrative description; (d) the calling routines (e) the called routines; (f) the commons referenced; and (g) any messages generated.[1] The information provided on the summary sheets in conjunction with the model flowchart shown in chapter 2 should facilitate the programmer's task of effectively maintaining the model. The interactions among subroutines which are explicitly stated on the summary sheets should also assist the programmer in updating or making any modifications to the source code dictated by user needs or peculiarities of the operating system.

It is important to note that some of the output listed under the messages generated section of the revised simplex (RVSMPX) program summary sheet would not normally occur. This is because the MAIN program contains an elaborate system of edit checking. In particular, all entries to the constraint matrix are constructed within the MAIN program based on data from the SETUPA COMMON block, so errors in input would not affect the feasibility of the application problem. However, input errors could affect the objective function and hence render the problem meaningless. The complete RVSMPX program was incorporated into FSESCM because, should it become necessary to revise the source code (e.g., due to a major change in the Life Safety Code), ill-defined problems could be sent to the optimizer. Having the full range of error messages should therefore facilitate finding any flaws in logic introduced by the changes. The program may also be uncoupled from the model for use as an optimizer in other applications.

---

[1]Throughout this section the term messages generated refers to an error condition. Consequently, those subroutines which produce output reports under normal circumstances will not reference these reports under the heading of messages generated. Reference to the output report will be made in the narrative section, however. Each report is described in detail in section 3.1.

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: CCONST -- Costing for CONSTruction

CALL STATEMENT: CCONST(CPUT,JPR,C,IT4X,ISTAT,HPAR,HLAB,JPRES)

DESCRIPTION:

This subroutine reads and prints the critical element counts and estimates the cost of installing retrofits which affect construction. It is entered each time data on a fire zone is input.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: CHAZAR — Costing for HAZARdous areas

CALL STATEMENT: CHAZAR(CPUT,JPR,C,IT4X,ISTAT,HPAR,HLAB,JPRES)

DESCRIPTION:

This subroutine reads and prints the critical element counts and estimates the cost of installing retrofits which affect hazardous areas. It is entered each time data on a fire zone is input.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: See Attachment

PROGRAM SUMMARY SHEET (ATTACHMENT)

SUBROUTINE NAME: CHAZAR

MESSAGES GENERATED:

Deficiency qualifier out of range. The allowable values are:

0. None present (e.g., Double, Single--Not sprinklered, and Single--Sprinklered), and

1. Aggregated count.

The values input were:

Double ____, Single--Non sprinklered ____, Single--Sprinklered ____

The run continues using a default value of 0 for the qualifiers incorrectly input.

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: COSEST — COSt ESTimation

CALL STATEMENT:

```
CALL COSEST(CPUT,INELT,C,I,I1,K1,K2,K3)
```

DESCRIPTION:

This subroutine is entered each time the data on a building safety feature is read in. The retrofit costs associated with each state in that building safety feature are then estimated. The following working parameters are transferred from the MAIN program:

1. I denotes the building safety feature number;
2. K1 denotes the length of the cost vector;
3. K2 is the state above that input;
4. K3 gives the maximum number of retrofit states, and;
5. I1 is the first simplex variable number.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: COSMOD — COSt MODifier

CALL STATEMENT: COSMOD(CPUT,CPUL,CPUM,CMODT,CMODL,CMODM)

DESCRIPTION:

This subroutine is entered each time data on a new building is input. The purpose of the subroutine is to adjust the component costs stored in the CPUL (per unit labor costs) and CPUM (per unit material costs) matrices to reflect cost growth (CMODT) and regional price differentials (where CMODL captures variations in the prevailing wage and CMODM captures variations in material prices). These adjustments are recommended since all costs in the CPUL and CPUM matrices are based on the rates prevailing in the Washington, D. C. metropolitan area during the summer of 1981. It is recommended that the values of CMODT, CMODL and CMODM be based on data from a construction cost index report or some other recognized source.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: CSMOKE — Costing for SMOKE detection and alarm

CALL STATEMENT: CSMOKE(CPUT,JPR,C,IT4X,ISTAT,HPAR,HLAB,JPRES)

DESCRIPTION:

This subroutine reads and prints the critical element counts and estimates the cost of installing smoke detection and alarm systems. It is entered each time data on a fire zone is input.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: CSPRNK — Costing for Automatic SPRiNKlers

CALL STATEMENT: CSPRNK(CPUT,JPR,C,IT4X,ISTAT,HPAR,HLAB,JPRES)

DESCRIPTION:

This subroutine reads and prints the critical element counts and estimates the cost of installing retrofits which affect sprinkler systems. It is entered each time data on a fire zone is input.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: See Attachment

PROGRAM SUMMARY SHEET (ATTACHMENT)

SUBROUTINE NAME: CSPRNK

MESSAGES GENERATED:

Sprinkler designation out of range. The allowable values are:

1. Wet, exposed;
2. Wet, concealed;
3. Dry, exposed; and
4. Dry, concealed.

The value input was: _____.
The run continues using a default value of 2.

Water supply designation out of range. The allowable values are:

1. Adequate;
2. Not adequate; and
3. Unknown.

The value input was: _____.
The run continues using a default value of 3.

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: CZODIM — Costing for ZOne DIMensions

CALL STATEMENT: CZODIM(CPUT,JPR,C,IT4X,ISTAT,HPAR,HLAB,JPRES, IPART)

DESCRIPTION:

This subroutine reads and prints the critical element counts and estimates the cost of installing retrofits which affect zone dimensions. It is entered each time data on a fire zone is input.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: DESCLS — DESign CLaSsification

CALL STATEMENT: DESCLS(IPAC,IEC5,JPR,IEC1,IECT,IT4XA,JPRES)

DESCRIPTION:

This subroutine attempts to assign each solution to a design classification. If the solution does not fit any of the predetermined design classes, a test flag is set to zero. The subroutine is entered from the MAIN program after the first two sets of alternate solutions have been generated and the class forcing loop of the MAIN program has been completed.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: INSETS — INitialize SETtingS

CALL STATEMENT: INSETS

DESCRIPTION:

This subroutine initializes (or reinitializes) the values of all major work spaces. It is designed to clear the working space so that no information from a previous building could influence the calculations for the one currently being run. This subroutine initializes all settings for integerized, sorted, and stacked solutions for a building. Solution counts by design class, all retrofit costs and the last fire zone's solutions for a building are also set to zero. It is called in four ways:

1. initially;
2. after each building has been solved;
3. if the output of the revised simplex routine was other than normal; and
4. if an error in the runstream was encountered.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: RESET

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: INTSOL — INTegerize SOLution

CALL STATEMENT:

```
INTSOL(A,IPAC,IT4S,TEST,L,X,Y,C,IT4XA,ISTAT,
    IFLOR,IZONE,IPART,X2)
```

DESCRIPTION:

This subroutine is entered each time a new solution has been generated. Upon entry the X vector contains the optimal (continuous LP) solution to the problem being solved. The entries in the X vector are then examined one at a time. Each building safety feature where a split occurs is flagged and counted. If no splits are encountered, the program then checks to see if an interdependency has rendered the problem infeasible. If the problem is still feasible, the TNPSOL subroutine is called and the solution is tested and packed into a row of the IPAC matrix. Control is then returned to the MAIN program. If one or more splits were encountered, an ad hoc integerization procedure is used whereby each building safety feature is used as the first entry, the second entry, etc., in integerizing the solution. The resultant solutions are then checked for feasibility in the event that one or more interdependencies came into play. The TNPSOL subroutine is then called and control is returned to the MAIN program.

CALLED BY: MAIN

CALLS: TNPSOL

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: PCOSTS — Print COSTS

CALL STATEMENT:
```
PCOSTS(JPR,IT4S,IT4X,L,C,HLAB,HPAR,JPRES,IFLOR,
       IZONE,CMODT,CMODL,CMODM,KPRES,IPRES)
```

DESCRIPTION:

This subroutine prints back the estimated retrofit cost for each state within a given building safety feature. In order to facilitate comparisons with any solutions generated as well as for identifying if and/or where a cost needs modification via the 'CHANGE' option, each cost is listed immediately below the state name.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: PRBLDG — PRint BuiLDinG

CALL STATEMENT: PRBLDG(ISTK,IEC1,IT4S,HOUT,HLAB,KPRES)

DESCRIPTION:

This subroutine is entered once all sets of alternate solutions for all fire zones have been generated and printed. The least cost solution for each of the 40 design classifications for each fire zone in the building is stored in the ISTK matrix. ISTK is then checked to see if a given design classification was generated either naturally or artificially for all fire zones. If a perfect match occurs, the solution for the entire building is printed. Each fire zone takes up one line in the printout. In order to facilitate the identification of each solution, the state values taken on by each of the 13 building safety features are printed out as are the surpluses and retrofit cost for each zone. If some fire zone did not contain this class of retrofits, no printout for the entire building is generated. Should the user wish such a retrofit, it would be necessary to synthesize it from the individual fire zone printouts.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: PRESOL — PREscriptive SOLution

CALL STATEMENT:

```
PRESOL(A,IPAC,IT4S,JPR,IT4ZA,JPRES,IFLOR,IPRES,
       IZONE)
```

DESCRIPTION:

This subroutine generates the solution which corresponds to prescriptive compliance to the Life Safety Code. If the prescriptive solution has been excluded from consideration as a viable alternative, then control is returned to the MAIN program. If not, then a solution is generated for later use as a baseline against which the design classifications can be ranked.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: PRTBAC — PRinT BACk

CALL STATEMENT: PRTBAC(HBLDG,ASTAR)

DESCRIPTION:

This subroutine prints an address label for the run which follows. It also serves as a separator between runs for different facilities. It is called immediately after reading in the facility ID, contact person and facility address contained in matrix HBLDG.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: PRTZON — PRinT fire ZONe

CALL STATEMENT: PRTZON(ISRT,IT4S,IT4B,IT8,ITECT,Z,HLAB)

DESCRIPTION:

This subroutine is entered only after all three sets of alternate solutions have been generated. The subroutine first prints a brief summary of the zone. The solutions are then printed out by design classification ranked by retrofit cost. Those solutions which do not fit a design classification are printed out in the following order:

1. strategies with no deficiencies in hazardous areas;
2. strategies with a single deficiency in hazardous areas, and;
3. strategies with a double deficiency in hazardous areas.

In order to uniquely identify each solution, the output state for each building safety feature is printed under the respective column heading. The surplus associated with each performance requirement is then printed under the headings S1, S2, S3, SG. The last column contains the estimate retrofit cost.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: RVSMPX — ReVised SiMPleX

CALL STATEMENT: RVSMPX(A,MA,B,MB,MT,NT,L,X,TOLP,INV,KQP)

DESCRIPTION:

This subroutine solves for the maximum value of a linear objective function subject to a set of linear constraints. Since the objective function is a linear combination of the state variables and the negative of the state transition costs, maximizing it is equivalent to minimizing the expected compliance cost. The problem consists of thirteen generalized upper bound constraints and four performance constraints (containment, extinguishment, people movement, and general). The problem is solved through the use of the revised simplex algorithm. The subroutine is entered from each of the four stages in the optimization loop of the MAIN program.

1. Each time data on a new fire zone has been input.
2. From the first state forcing loop.
3. From the second state forcing loop.
4. From the third state forcing loop.

If the user options 'CHANGE' or 'REQUIR' are exercised, stages 1 through 4 will be repeated.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: See Attachment

PROGRAM SUMMARY SHEET (ATTACHMENT)

SUBROUTINE NAME: RVSMPX

MESSAGES GENERATED:

System error - Computational impossibility.

Epsilon = ____ Capital epsilon = ____
____ Non-zero entries are effectively equal to zero.

Minimum pivot was ____ at iteration ____.

Error - The problem is infeasible. Infeasibility indicated during
re-inversion of the basis matrix.

Real objective function = ____.

Warning — Small pivot element at interation ____ of Phase ____.
Pivot = ____.

Error — The variables specified as comprising an intial solution do not form a basic feasible solution.

End of phase ____. Objective function = ____.
There were ____ iterations.

Phase ____ Iteration ____. Pivot = ____.
Objective function = ____. ____ ____ entered the basis ____ ____ left.

Error — The problem is infeasible. The constraints associated with the artifical variables below are inconsistent. If none appear, numerical difficulties hve been encountered. The largest entry in the objective function row is ____.

PROGRAM SUMMARY SHEET (ATTACHMENT)

SUBROUTINE NAME: RVSMPX

MESSAGES GENERATED (CONTINUED):

Error — The problem is unbounded. The variable X(-) can assume an arbitrarly large value, thereby yielding an arbitrarily large value of the objective function.

The error was detected at iteration — of phase ____. At that time the objective function value was ____ and the following variables were basic.
_ _ _ ··· _ _ _

Warning — Computational inconsistency indicated at end of phase 1. The algorithm will continue with phase 2 but the user is advised to criticize the results.

At least one element of the right hand side column is less than zero. Subroutine terminates.

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: SEARCH — SEARCH for next building

CALL STATEMENT: SEARCH

DESCRIPTION:

This subroutine is entered in two ways:

1. If an error in the runstream was encountered; and
2. If the output of the revised simplex routine was other than normal.

The subroutine is designed to read until the last entry for the building where the problem was encountered is reached. It will then return control to the MAIN program and look for the inputs for the next building. If there are no other buildings the run will be terminated.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED:

The values on card _____ after the error are: _____.

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: SRTSOL — SoRT SOLutions

CALL STATEMENT: SRTSOL(IPAC,IRT,IEC1,IECT,L,X2)

DESCRIPTION:

This subroutine is entered only after all three sets of alternate solutions for each fire zone have been generated. The variable X2, the bound on the objective function is used as the bound on retrofit cost when beginning a sort. The subroutine first ranks each compliance strategy within a given design class by retrofit cost. Solutions which do not belong to a design classification are then ranked in the following order:

1. Strategies with no deficiencies in hazardous areas;
2. Strategies with a single deficiency in hazardous areas, and;
3. Stragegies with a double deficiency in hazardous areas.

All solutions once sorted are stored in the ISRT matrix.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: STKSOL — STacK SOLutions

CALL STATEMENT: STKSOL(ISRT,ISTK,IPAC,IEC1,C,KPRES,IPRES)

DESCRIPTION:

This subroutine is entered only after all three sets of alternate solutions for each fire zone have been generated, sorted and printed. The solutions associated with each design class are screened and the one with the lowest retrofit cost is saved. These solutions are used later in subroutine PRBLDG as a means of synthesizing a retrofit strategy for the entire building. Prior to leaving this subroutine the ISRT and IPAC matrices and reinitialized.

CALLED BY: MAIN

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

PROGRAM SUMMARY SHEET

SUBROUTINE NAME: TNPSOL — Test aNd Pack SOLutions

CALL STATEMENT: TNPSOL(IPAC,IT4S,L,Y,ISTAT,IFLOR,IZONE)

DESCRIPTION:

This subroutine is entered each time a new solution has been generated and integerized. Upon entry, the Y vector contains the integerized solution to the problem being solved. The entries in the Y vector are then examined one at a time and packed into the L vector. The L vector is then used to determine the state number for each building safety feature. The state numbers are then loaded into the ISTAT vector. The resultant state numbers are then checked against the entries in the IPAC matrix. If the same solution has already been found, the current one is discarded and control is returned to the INTSOL routine. If a distinct solution results, then a new row of IPAC is opened up and the solution is stored in it. The value of the objective function, floor and zone IDs and any score surpluses are also stored in that row of the IPAC matrix.

CALLED BY: INTSOL

CALLS: NONE

COMMONS REFERENCED: NONE

MESSAGES GENERATED: NONE

REFERENCES

American National Standards Institute, American National Standard Programming Language FORTRAN, ANSI X3.9-1978, New York, 1978.

Chapman, R.E., A Cost-Conscious Guide to Fire Safety in Health Care Facilities, National Bureau of Standards, NBSIR 82-2600, Washington, D.C., 1982.

Chapman, R.E. and W.G. Hall, User's Manual for the Fire Safety Evaluation System Cost Minimizer Computer Program, National Bureau of Standards, NBSIR (in preparation), Washington, D.C.

Gass, S.I., Linear Programming: Methods and Applications (New York, McGraw Hill Book Company, 1975).

Hall, W.G., R.H.F. Jackson and P.B. Saunders, The National Bureau of Standards Linear and Quadratic Programming Subroutines, National Bureau of Standards, Report 10695, Washington, D.C., 1972.

National Bureau of Standards, Computer Model Documentation Guide, NBS Special Publication 500-73, Washington, D.C., 1981.

National Bureau of Standards, Guidelines for Documentation of Computer Programs and Automated Data Systems, FIPS Publication 38, Washington, D.C., 1976.

National Fire Protection Association, Code for Safety to Life from Fire in Buildings and Structures, NFPA 101-1981, Quincy, Mass., 1981.

Nelson, H.E., and A.J. Shibe, A System for Fire Safety Evaluation of Health Care Facilities, National Bureau of Standards, NBSIR 78-1555, Washington, D.C., 1980.

Saunders, P.B., editor, Selected Assessment Strategies Applied to Short-Term Energy Models, National Bureau of Standards, NBSIR 83-2672, Washington, D.C., 1983.

FEDERAL INFORMATION PROCESSING STANDARD SOFTWARE SUMMARY

01. Summary date: Yr. 83 Mo. 07 Day 19

02. Summary prepared by (Name and Phone): Robert E. Chapman (301) 921-3855

03. Summary action: New [X] Replacement [ ] Deletion [ ]

04. Software date: Yr. 83 Mo. 07 Day 19

05. Software title: FIRE SAFETY EVALUATION SYSTEM COST MINIMIZER (FSESCM)

Previous Internal Software ID:

07. Internal Software ID: NA

06. Short title:

08. Software type: [ ] Automated Data System [X] Computer Program [ ] Subroutine/Module

09. Processing mode: [ ] Interactive [X] Batch [ ] Combination

10. Application area

General		Specific
[ ] Computer Systems Support/Utility	[ ] Management/Business	Fire Safety in Health Care Facilities
[X] Scientific/Engineering	[ ] Process Control	
[ ] Bibliographic/Textual	[ ] Other	

11. Submitting organization and address:

Operations Research Division
Center for Applied Mathematics
National Bureau of Standards
Washington, D. C. 20234

12. Technical contact(s) and phone:

Robert E. Chapman (301) 921-3855

William G. Hall (301) 921-3855

13. Narrative

This program minimizes the cost to hospitals and nursing homes of complying with the Life Safety Code. The model uses the Fire Safety Evaluation System developed by the Center for Fire Research at the National Bureau of Standards as a means for identifying alternatives to prescriptive compliance to the Life Safety Code. A linear programming algorithm is then used to define compliance strategies which, for a given set of constraints, are optimal in terms of minimum cost.

14. Keywords: Building codes; building economics; economic analysis; fire safety; health care facilities; hospitals; life safety; mathematical programming; nursing homes; renovation.

15. Computer manuf'r and model	16. Computer operating system	17. Programing language(s)	18. Number of source program statements
Sperry Univac	1100/82	FORTRAN 77	APPX 4000

19. Computer memory requirements	20. Tape drives	21. Disk/Drum units	22. Terminals
less than 256 K Bytes	U9V/UPS (9 Track 1600 or 6350 FPI)		

23. Other operational requirements

24. Software availability: Available [X] Limited [ ] In-house only [ ]

25. Documentation availability: Available [X] Inadequate [ ] In-house only [ ]

26. FOR SUBMITTING ORGANIZATION USE

Standard Form 185
1974 July
U.S. Dept. of Commerce—NBS
(FIPS. Pub. 30)

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 83-2749	2. Performing Organ. Report No.	3. Publication Date July 1983

4. TITLE AND SUBTITLE

PROGRAMMER'S MANUAL FOR THE FIRE SAFETY EVALUATION SYSTEM COST MINIMIZER COMPUTER PROGRAM

5. AUTHOR(S)

Robert E. Chapman and William G. Hall

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Center for Fire Research
National Engineering Laboratory
National Bureau of Standards
Washington, DC 20234

Department of Health and Human Services
Washington, DC 20201

10. SUPPLEMENTARY NOTES

[X] Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

The Fire Safety Evaluation System Cost Minimizer (FSESCM) computer program integrates engineering and economic considerations with a linear programming algorithm which permits the least-cost means of upgrading health care facilities to compliance with the Life Safety Code to be identified. A mathematical discussion of the application problem is used to introduce the basic philosophy behind the computer program. Each routine is described with emphasis on such topics as: (1) purpose; (2) calling sequence; (3) common blocks used; and (4) reports produced. A series of descriptive tables and a glossary are used to define all reports and variables. A discussion of test results and provisions for updating or modifying the source code are also given. The program is written in FORTRAN and complies with the ANSI X3.9-1978 software standard.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

Building codes; building economics; economic analysis; fire safety; health care facilities; hospitals; life safety; mathematical programming; nursing homes; renovation.

13. AVAILABILITY

[X] Unlimited

[ ] For Official Distribution. Do Not Release to NTIS

[ ] Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

[X] Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

98

15. Price

$11.50