FRA-81-12

FRA/ORD-82/08
DOT-TSC-FRA-81-12

PB 82-181587

User's Guide For a Computerized Track Maintenance Simulation Cost Methodology

Richard L. Smith
Allan I. Krauter
Joseph Betor

Shaker Research Corporation
Northway 10 Executive Park
Ballston Lake NY 12019

February 1982
Final Report

This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161.

U.S. Department of Transportation
Federal Railroad Administration

Office of Research and Development
Office of Improved Track Structure Research
Washington DC 20590

01-Track & Structures

NOTICE

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its contents or use thereof.

NOTICE

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the object of this report.

Technical Report Documentation Page

1. Report No. FRA/ORD-82/08	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle USER'S GUIDE FOR A COMPUTERIZED TRACK MAINTENANCE SIMULATION COST METHODOLOGY		5. Report Date February 1982
		6. Performing Organization Code DTS-731
7. Author(s) Richard L. Smith, Allan I. Krauter, and Joseph Betor		8. Performing Organization Report No. DOT-TSC-FRA-81-12
9. Performing Organization Name and Address Shaker Research Corporation* Northway 10 Executive Park Ballston Lake NY 12019		10. Work Unit No. (TRAIS) RR219/R2309
		11. Contract or Grant No. DOT-TSC-1594
12. Sponsoring Agency Name and Address U.S. Department of Transportation Federal Railroad Administration Office of Research and Development Washington DC 20590		13. Type of Report and Period Covered Final Report September 1978-September 1980
		14. Sponsoring Agency Code RRD-12

15. Supplementary Notes

*Under contract to: U.S. Department of Transportation Research and Special Programs Administration Transportation Systems Center Cambridge MA 02142

16. Abstract

This User's Guide describes the simulation cost modeling technique developed for costing of maintenance operations of track and its component structures. The procedure discussed provides for separate maintenance cost entries to be associated with definable track substructures such as rail, cross ties, or ballast. In this manner separate tabulations of maintenance expenditures can be obtained from the computerized technique.

This guide describes the background of the technique as well as provides two examples of the application of the costing procedure. The maintenance costing examples provided illustrate the use of maintenance action diagrams representing the system being modeled. The two example systems involve time-dependent cost estimating and produce costs-by-year for the class of track; component or sub-structure repaired; type of maintenance operation; as well as by several costing sub-categories including labor, material, equipment, delays, scrap, fines, etc.

Although the computer program is tailored specifically for track maintenance cost analysis, user definable flexibility is built into the analysis. Time-dependent aspects of costs, which can vary with track loading MGT, railroad policy, track component quality, and/or Federal Safety Standards, can be entered in the simulation with the aid of user definable functions.

17. Key Words Simulation Cost Model; Life Cycle Costs; Maintenance; Track; Track Standards; Track Maintenance Planning; Computer Simulation.	18. Distribution Statement DOCUMENT IS AVAILABLE TO THE PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VIRGINIA 22161

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 298	22. Price

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

PREFACE

Inadequate return on investment by segments of the domestic railroads is an underlying cause of many problems being faced by the industry. One of the consequences of poor financial conditions is an inadequate level of track maintenance resulting in steadily rising track-related accident rates and a decrease in operational efficiency. One study of deferred track maintenance estimates that 123 million ties and 25,000 track miles of rail are in need of replacement at a cost of 17 billion dollars.

The Federal government has attempted to initiate a variety of programs, regulations, and economic incentives to help the industry respond to these problems. One of these programs is the Improved Track Structures Research Program, sponsored by the Federal Railroad Administration (FRA), Office of Research and Development. This program is exploring alternative approaches to achieve a reduction in track-related accidents; a reduction in track-life cycle costs; and improvements in rail service through quality lower-cost maintainability. This guide is intended to document a simulation cost methodology developed under the program.

This work was conducted for the FRA through the Transportation Systems Center (TSC), Cambridge MA. Robert Smith was the TSC Technical Monitor.

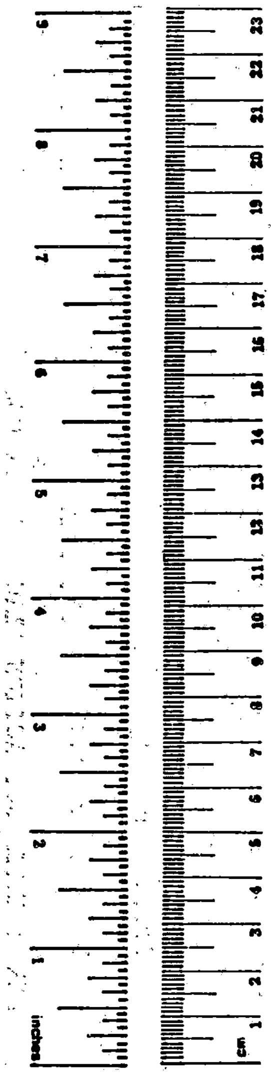
We acknowledge the guidance and cooperation of individuals from two railroads for their assistance in gathering and defining the necessary data for this effort. Robert Tuve, Manager, Quality Control Engineering, Southern Railway System, and Harry Schultz, Assistant Chief Engineer, Delaware and Hudson Railway, contributed significantly.

METRIC CONVERSION FACTORS

Approximate Conversions to Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
in	inches	*2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
		AREA		
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
		MASS (weight)		
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	tonnes	t
		VOLUME		
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
		TEMPERATURE (exact)		
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

*1 in = 2.54 (exactly). For other exact conversions and more detailed tables, see NBS Misc. Publ. 286, Units of Weights and Measures, Price $2.25, SD Catalog No. C13.10:286.



Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
		AREA		
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
		MASS (weight)		
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1000 kg)	1.1	short tons	
		VOLUME		
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
		TEMPERATURE (exact)		
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F

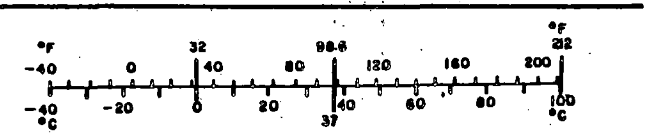


TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

1.0 INTRODUCTION AND OVERVIEW

The actual total cost of maintaining a segment of railroad track is extremely difficult to ascertain. At the point of manufacture a length of rail, a tie, or a ton of ballast has, at a given instant, a fixed determinable cost. But materials of repair are only a fraction of the overall maintenance expenses.

What about delivery costs, installation effort, and the resulting costs of delaying traffic if the process doesn't proceed on schedule? What about next year's cost? Or a more pertinent question:

> WHAT WOULD BE THE COST IMPACT OF SOME PROPOSED MAINTENANCE OPERATION WHICH WOULD ALTER THE STANDARD MAINTENANCE PRACTICE NOW IN USE?

Alterations in a railroad's operating or maintenance procedure can often lead to undesirable economic situations. For example, the conversion to systemwide use of 100-ton hopper cars has increased the rate at which rails sustain fatigue damage.

The task of costing any proposed maintenance practice or policy modification is complicated by:

- The fact that there are almost as many maintenance practices as there are railroads to perform them;
- The constant replacement of system materials;
- The unplanned nature of some maintenance operations; and
- The everchanging demands of business which can alter the physical loading of the track structure themselves.

These problems have prompted several attempts in recent years to get a better grip on what it costs to maintain track. In fact, several of these approaches to track maintenance costing were reviewed recently at a workshop* on the subject of track maintenance. Figure 1-1 is a summary of some of the types of computerized techniques being used today for evaluating track maintenance costs or performance.

*Track Maintenance Planning Workshop, Penn State University, June 1980.

	ORGANIZATION	PROGRAM USE	COMMENTS
DATA BASED MANAGEMENT/REVIEW	Canadian National Railways	Manpower Planning	Data of past maintenance efforts are converted through regression procedures to forecasting formulas for estimating manpower needs by geographic location in the Canadian National Railway System.
	P.A. International	Job Task Planning	Continuous tabulation and updating of projected work for periods of one to four weeks. Identifies through listings the need for people and equipment by regions.
	Conrail	Track Maintenance Management System	Keeps tabs on manhours by maintenance operation during past year. Data from sample 9 miles of track have been kept in detail on a trial basis.
	Southern Railway	Maintenance Planning from track quality indices	Track geometry parameters are measured and correlated with track performance (derailments, rail defects, train delays) and track is identified where scheduled maintenance can be economically justified.
	Bessemer and Lake Erie Railroad	Organization of maintenance operations data	Tabulates and records for future reference track repair histories under 110 different accounting codes.
ECONOMIC SIMULATIONS	Pugh Roberts	Analysis of track related safety options	Uses a collection of mathematical equations that describe the various inter-relationships among the safety operational elements of the railroad.
	Shaker Research	Simulates maintenance actions and tabulate cost of track repair	Allows user to simulate his own maintenance practices and obtain cost accounting outputs with time. Provides for cost comparison simulations under user defined maintenance practice modifications.

FIGURE 1-1. COMPUTERIZED TRACK MAINTENANCE PROGRAMS

This guide presents a methodology for computer simulation of track maintenance procedures and costs. The methodology and FORTRAN IV computer program contained herein (Appendix D) provide for:

- Graphic representation of the system being simulated through the use of a maintenance action diagram;
- Costing proposed alternate maintenance action policies or procedures;
- Listing all annual track maintenance costs; and
- Maintenance cost outputs at any desired instant during the time simulation.

This guide has three purposes. These are to present the simulation costing methodology for track maintenance, to cover the details of the computer program developed, and to give examples of its use.

Section 2 of the report is concerned with introducing the methodology. An introduction is given to the use of maintenance action diagrams for setting up a simulation in graphic form. Also discussed are the types of data needed to perform a simulation as well as costs that are handled (and not handled) by the program in its present state.

Section 3 presents details of the specific data categories under which the program holds cost information.

Section 4 deals with the details of representing maintenance actions in graphic form. Sample maintenance diagrams are covered and alterations of diagrams for costing different maintenance operations and schemes are shown.

Sections 5 and 6 are included in order to give data entry examples and to provide the reader with two example simulations from two separate modeled track maintenance systems. The Appendices contain the computer program, flow charts, cost data examples for test runs, and tabulated output results.

2.0 OVERVIEW OF TRACK MAINTENANCE SIMULATION COST METHODOLOGY

2.1 Track Maintenance as a Cost Simulation Problem

Track maintenance costs can be defined as the annual direct charges for the upkeep of track property and equipment. Track property includes rail, cross ties, ballast, and associated hardware fastenings. Equipment refers to those pieces of machinery used for inspecting, installing, and keeping the track in satisfactory operating condition. In early stages of the work for this contract it became apparent that there are some commonly identifiable aspects to these costs and the associated maintenance operations which are performed by every railroad. It also became apparent that there are as many different ways to perform some of the major tasks of track repair as there are track systems in use. Each railroad has its own maintenance policy which has usually been derived from many years of practical repair operations experience.

In developing a methodology for determining track maintenance costs it was desired that:

- the technique be applicable to a variety of railroads' maintenance of way operations,
- the procedure allow for cost comparison of alternate maintenance operations, either real or imaginary in concept,
- the procedure allow for evaluation of the cost impact of proposed safety standards.

The approach taken in the work centers around a general computerized simulation procedure which can handle any user defined maintenance system. The program uses a standard fourth order iteration technique and a time stepping procedure to simulate defined maintenance actions. As a result, non-linear costing information can be addressed by the procedure. Figure 2-1 is a display which represents the cost information handling of the computer technique developed.

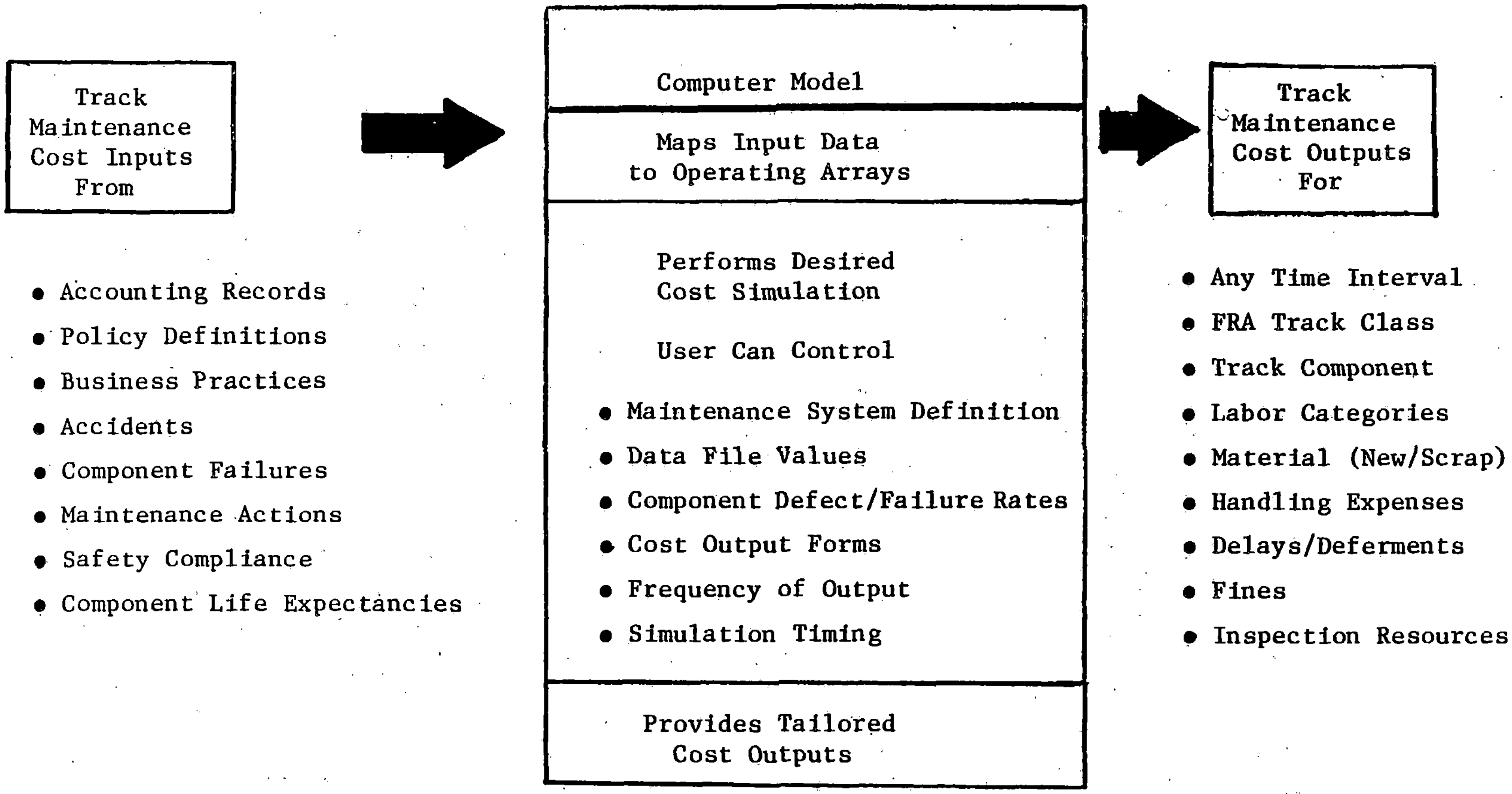


FIGURE 2-1. SIMULATION COST MODEL INFORMATION HANDLING

In any modeling procedure, one must agree in advance to what the model will represent and include. Determining which costs to include as maintenance costs in the modeling process is not simply resolved. Defining what "is" or "is not" a track maintenance cost is often ambiguous and should be decided carefully in advance. An attempt has been made in the present study to define a manageable set of cost input data that will provide a useful set of output information about the modeled track maintenance system.

In the model developed the user is able to include or exclude whatever costs he wants. The present limitations of the simulation cost model are determined in part by the cost information or data available to the program. If a cost item is to be included in the model there must be some data to describe the maintenance expense.

For example purposes, the present technique of track maintenance modeling described in this report has been set up to allow at least for those costs listed below.

Maintenance Cost Items Included in Model Reported

- Labor on repairs of five track structural components. For example purposes the track was defined to consist of the three major substructures or components (referred to by numbers 1 through 5 in computer outputs):

rail	1 - jointed 2 - welded
cross ties	3 - wood 4 - concrete
ballast	5 - ballast

- Present-day material costs of these new components installed during repair.
- Cost to keep electrical isolation and/or contact throughout rail structure.
- Cost of track inspection labor and necessary equipment used in the inspection process.
- Supervisory labor needed in monitoring subcontracted maintenance operations.

- Fines accrued for not keeping the track components within safe standards.
- Delays of trains caused directly by the maintenance repair crews.
- Delays from slow orders imposed as a result of insufficient maintenance upkeep.
- Return costs from scrapping components of system.
- Subcontracted maintenance of the five basic components.
- Delivery costs of transporting new (renewed) components to the repair sites.
- Travel and living associated with getting to and from the repair location if paid by the railroad.
- Cost of heavy equipment needed to perform repairs on basic track structure.
- Cost of fueling or powering certain pieces of equipment such as welders, etc.
- Accident costs.
- Rework or refurbishment costs (for example, conversion of used jointed rail to welded rail).
- Track idle or closure costs.

Simulation cost modeling can address questions of:

- How much does it cost to maintain track?
- What are the cost breakdowns within the maintenance system by defined structural component (i.e., rail, cross ties, or ballast)?
- Where in the maintenance system are the most costly procedures?
- What savings in annual maintenance can be expected if certain operations or policies are altered?
- What will be future maintenance costs if the system work volume or procedures are changed?

Maintenance costs are available for output at any time point in the simulation process. This time frozen "snap-shot" of maintenance repair costs and system rates of repair can be used to compare with past or present accounting records of the railroad system being modeled.

Single simulation time point costs can be printed and broken out into the following categories:

1. Simulation maintenance action diagram path number.
2. Track component (i.e., rail, tie, ballast, etc.).
3. FRA track class (1-6).
4. Major repair block or type (such as rail laying, surfacing, etc. For example purposes, 20 separate types were identified.)
5. Cost code (for the present example consists of about 15 codes including):

   a. Material costs
   b. Equipment costs
   c. Fines
   d. Delays
   e. Scrap return costs
   f. Contracted costs
   g. Delivery costs
   h. Travel and living
   i. Six levels of labor

2.2 Simulation Technique

The Simulation Cost Modeling (SCM) technique consists of a qualitative and quantitative representation of the maintenance actions associated with the track system being modeled. The qualitative representation consists of a maintenance action diagram which is usually constructed by the user and describes the maintenance actions and their relationship to one another. The quantitative representation is the data set associated with the action diagram. Linking the two representations is the computer program. This program quantifies the modeled actions shown in the diagram, implements the associated data set, and provides a selected set of cost outputs.

The computerized time stepping routine used in the simulation of maintenance actions is a fourth order iteration procedure. Being a rate of change structured technique this algorithm takes into account the rate at which the various track components are replaced and the rate at which they naturally generate observable defects requiring maintenance work to be performed. Figure 2-2 shows this time stepping action and information processing scheme for keeping tabs on the condition or quality of the various track components.

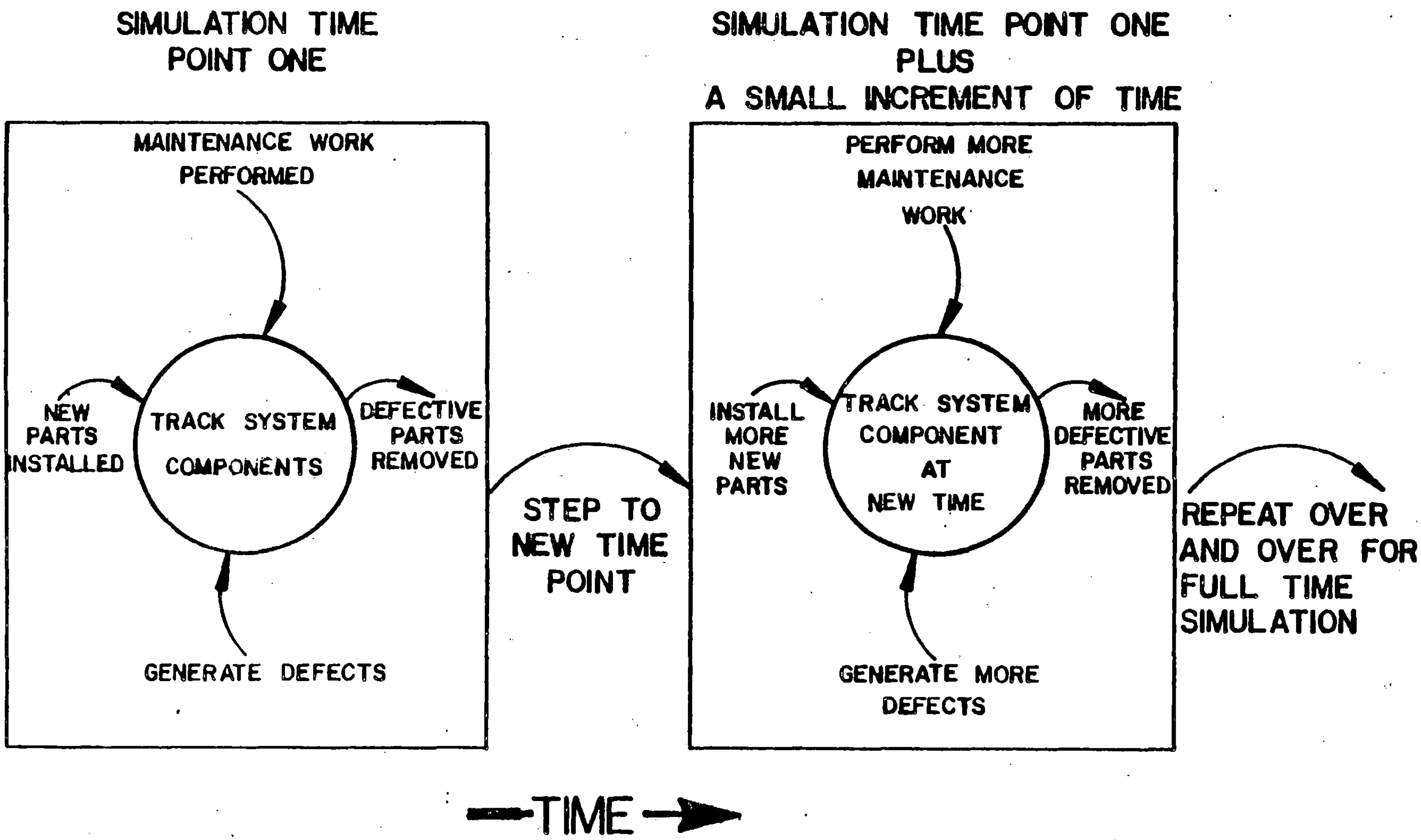


FIGURE 2-2. TIME SIMULATION REPRESENTATION SHOWING VARIOUS ACTIONS AFFECTING THE TRACK SYSTEM COMPONENT QUALITY

2.3 Maintenance Action Diagrams

The representation of the maintenance system takes the form of a pictorial action diagram which describes how the railroad maintains its track. The maintenance action diagram is analagous to a diagram of a water pipe network. Water systems can be represented by drawings which show the various connecting or branching points as well as their distribution pattern. In a similar fashion the cost model maintenance action diagram displays the system being represented. Each of these diagrams has inherent characteristics which must be understood in advance in order to be used. The piping diagram, for example, through a developed notational convention can reveal pipe lengths, sizes used, elevation of layout, valves, and many other features.

The simulation cost model action diagram also carries with it certain special meanings which are discussed here in order that the reader can gain a more thorough understanding of the details in this pictorial view of the maintenance system. Figures 2-3 and 2-4 display some of the shorthand notation used in maintenance action diagramming. All maintenance action diagrams contain paths or lines with arrowheads which are used for showing the association between various maintenance actions or stand for a maintenance action itself. This shorthand method of displaying the maintenance system pictorially is in one sense convenient and in another misleading. It is convenient in that the overall system may be viewed in a glance, and if the notation is understood, can give an impression of the complexity of its structure. This method is misleading from the standpoint that many of the details of the system cannot be contained fully in the display, and may lead the viewer to believe that the model is too simple.

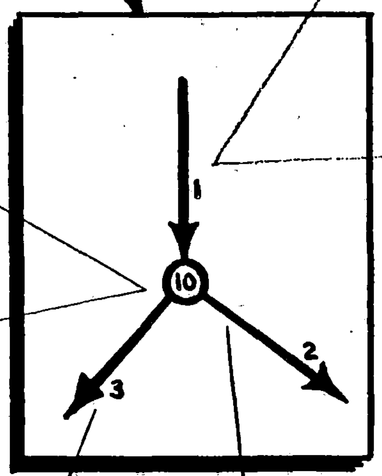
Figure 2-3 shows the numbering technique used in diagramming. The path numbers given on the diagram are used for reference in the computer handling of the cost information input and output. Maintenance costs by path number can be selected by the user.

Diagram numbered circles represent maintenance decision points. These decisions define alternate routes whereby two separate actions are defined

PORTION OF SCHEMATIC DIAGRAM

Paths are used to represent maintenance "actions" or simply to act as connecting links that show action sequencing and hierarchy

Numbered circles represent system maintenance decision points that affect downstream actions



Each path may have many non-zero scalars associated with its action. These scalars define:

- Components that are represented by this path
- Track classes that are represented by this path
- Amount of component on path or amount of component which is being transferred by path
- Rate at which maintenance action is performed
- Labor rate costing of maintenance work
- Quality of components to which work is applied
- Functional operators controlling user defined relationships which can be used to modify system scalars

These numbered split points can also represent alternate maintenance policy actions

This path represents the balance of the components from path "1" not transferred downstream by path "2"

This path represents that portion of the components associated with path "1" transferred downstream to another point in the maintenance system

FIGURE 2-3. MAINTENANCE ACTION DIAGRAM CONCEPTS AND NOTATIONS USED IN NUMBERING PATHS AND SPLIT DECISION POINTS

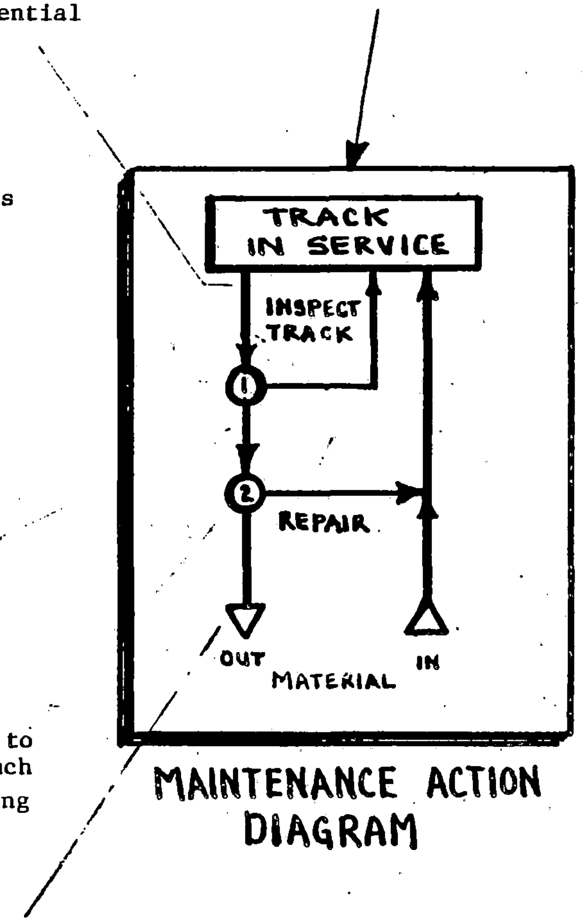


FIGURE 2-4. CONCEPTS AND NOTATIONS USED BY SIMULATION COST METHODOLOGY IN MAINTENANCE ACTION DIAGRAMS

on the track that is conceptually sent beyond this point in the maintenance system. In addition, the maintenance action diagram often has other notational characteristics needed for representing track maintenance. Some of these features are shown in Figure 2-4 and include:

- The "in use" or "in service" portion of the track system,
- The inspection-of-track loop,
- The removal or scrapping of defective system components,
- The act of maintaining the defective track found from a single inspection process, and
- The supplying or bringing of new components to replace those being scrapped from the system.

2.4 Cost Model Limitations

The present maintenance cost model is constrained mainly by the information or data available to be supplied to the program. One could find he does not have sufficient time or resources to collect and enter the data required for a maintenance system whose scope is too inclusive. An attempt has been made in the present study to limit the scope of the required input information while at the same time providing a technique that will yield a useful set of output information about the modeled track maintenance system.

In limiting the scope of the examples used for explaining the model, certain costs have been excluded from the costing procedure. In general, the effects of most of the excluded costs can be obtained, if needed, by operating on the output costs obtained from the simulation cost model. An example would be costs of inflation. Once costs of maintenance have been obtained from the model an inflationary factor might be used to establish final costs including inflation effects.

Although they could be handled by the model, the following cost categories have not been explicitly provided for in the present simulation cost model of the track maintenance system,

- Taxes
- Cost of money, interest
- Cost of inflation
- Costs of equipment storage such as housing and land
- Cost of bridge maintenance
- Cost of signal excitation and transmitting electronics and housing for such
- Failure sensing and detection equipment
- Opportunity costs
- Penalties paid to customers for not delivering goods as a result of track condition or track maintenance practices
- Business cost to customers for not delivering goods on time as a result of shipping delays due to track condition or track maintenance practices
- Loss of business due to unreliable rail service to customers as a result of poor track maintenance.

2.5 Data for Model

The data base required for running a useful cost model is typically very large. Since a large effort in time is needed in collecting data, it was decided that the data coding should:

1. Provide the model user with an annotated description of the data coded;
2. Allow the user to update the data quickly as new (or more reliable) information about the system becomes available; and
3. Let the user add (delete) associations between data elements without restructuring the whole body of data already established.

Data input structuring for use by the computer program are explained in Section 5. Examples of data entry and formats are included in Appendices G and H.

Data here are loosely defined as those numbers needed by the computer program to successfully simulate the maintenance system being analyzed. In general, the data required fall into six separate categories:

1. Computer program control numbers.
2. Cost and performance rates for every maintenance action diagrammed.
3. Initial annual amounts of components handled on each path of the action diagram.
4. Condition (quality) of each component at the start of the simulation.
5. Defect generation rates for each component being handled.
6 Formulas which control maintenance actions.

The computer control numbers are described in detail in Appendix F. Examples for their entry are given also in Appendix F as well as in Appendix G. Control numbers are used by the program to sort incoming data, define the simulation procedure, and set the type of cost outputs, as well as their frequency during the execution of the program.

Control inputs to data files named "IFILE", "RFILE", and "INODE", allow the user to define and run a simulation for any maintenance action diagram as long as it is a closed network of paths similar to those depicted schematically in Figures 2-3 and 2-4. Two examples of these kinds of maintenance action networks are drawn out in detail in Appendix K. The "IFILE" data set defines for the computer the size and complexity of the maintenance network. The "RFILE" contains real numbers used in the computational cycle, the time length of the simulation, and frequency during the cycle of printing the cost outputs. The information contained in "INODE" defines explicitly the path connections of the maintenance action diagram used in the simulation. This file also contains (see Appendix F) path type codes for describing the various allowable path linkage forms that may be desired in general maintenance simulations.

Costs, items 2 through 4 above (including initial component amounts and qualities), are all contained in the path by path cost data entries coded as shown in Appendix G. These data initialize maintenance action diagram path information and contain a commentary entry on the right side. Insertion and/or deletion on a line by line (path by path of the maintenance diagram) basis is allowed for and is convenient for updating the cost data should new information become available.

Track structures such as rail and ties tend to develop defects as they age. These defect generation rates are formed by the program from the Weibull distribution and the data supplied to the "WEIBL" file. This distribution is very general and is well suited for many engineering components which have finite wear-out lives. The attributes of this distribution and its potential for representing cyclic failure modes are discussed in References 9 and 11. Details of data entries to "WEIBL" are included in Appendix F.

Formula for functional relationships of maintenance actions and/or costs can be entered by the user to modify a programmed simulation. In general, any formula can be written for altering costs and/or path elements. For user convenience several families of curves have been programmed which can be used for this purpose. Entry of special parameter values to the "DATA" file will provide many functional operators for the user. Examples of functional use are discussed in Sections 5 and 6 as well as in Appendix E (see explanation of "FUNCTS" subroutine).

2.6 Computer Program Structure

The simulation program designed for implementing the costing methodology is contained in Appendix D. The listing is over 2500 lines in length and has been broken down into 36 separate subroutines. The full subroutine list by name and file number and in alphabetical order is given in Appendix E.

The program is written in standard FORTRAN IV language. The present version has been set up to operate in the batch mode with all run options being selected before running through the control data files "IFILE" and "RFILE". For two example inputs, see Appendices G and H.

3.0 DATA STORAGE CATEGORIES FOR MODELING ANALYSIS

Each path of the maintenance action diagram is intended to portray some element of repair work or service action. In order for the costing information to be of use the analysis must provide for the tabulation of expenses under several different headings.

The computerized cost data are therefore kept in the analysis under separate categories. These categories are, in general, expandable by the user (see Appendices F, G and H). The subcategories used in the model examples discussed here include maintenance expenses by:

- Maintenance Action Diagram Path Number
- Major Repair Block Operation (usually a collection of paths)
- Track Structural Component
- FRA Track Classification
- User Identified Cost Descriptor Codes

3.1 Maintenance Action Paths and Diagrammed Repair Blocks

Cost data for the simulation are entered and keyed to the paths of the maintenance action diagram drawn for the simulation. For convenience the paths of the maintenance action diagram are assigned a number when the drawing is constructed. These paths numbers can be assigned in any way by the user who wishes to perform a simulation. It must be remembered, however, that the program stores the input data in matrix form with one of the matrix subscripts being comprised of the path numbers labeled on the action diagram. Thus, in order to conserve computer space (required matrix size), the path numbering should be sequential starting from one. If path numbering is not sequential (often convenient for inserting paths in the diagram at a future time), this will simply increase the program running time.

In drawing the maintenance action diagram it is convenient to encircle or block out regions of the diagram which represent a single repair operation concept. In this way, the total expenses of this "block" or repair operation might be followed. For an example, see the diagram included in Appendix K. In the second example run for the present study, cost summaries were obtained for twenty separate repair or "block" operations (see Appendix J). The twenty repair (maintenance) operation "blocks" were numbered and included:

1. Laying New Welded Rail (Method 1)
2. Laying New Welded Rail (Method 2)
3. Laying New Welded Rail (same wt. welded)
4. Rail Change Out (jointed)
5. Transpose Rail
6. Transpose Rail (dummy rail method)
7. Track Panel Installation
8. Surfacing
9. Smoothing
10. Timber and Surfacing
11. Tie Renewal
12. Joint Repairs
13. Turn-out Repairs
14. Rail Buildup
15. Rail Grind
16. Ballast Cleaning
17. Brush and Weed Control
18. Ditching
19. Snow/Ice Removal
20. Slides/Washouts

These identified track maintenance repair operations need not remain fixed, but can be modified as required by the user. By expanding the number of paths of the diagram a larger or different set of maintenance repair operations could be handled. Also, if a given track system does not use one or more of the included maintenance practices, then the path data for those operations could simply be zero.

With this process of labeling groups of operations, two or more similar maintenance operations could be compared by diagramming each operation, supplying the separate cost data, and rating the separately tabulated expenses on the outputs from the model. This was done, for example, in the laying of welded rail (methods 1 and 2) above. See output examples of Appendix J.

3.2 Track Components and FRA Track Classifications

For purposes of the examples in this guide, it has been assumed that the track system is comprised of five basic components upon which maintenance repair operations are performed. These five components are:

Major Substructure	Computerized Component Number	Description	Units Used
Rail	1	Jointed	Miles
Rail	2	Welded	Miles
Cross Tie	3	Wood	Miles
Cross Tie	4	Concrete	Miles
Ballast	5	Ballast	Miles

Therefore, the model provides for storing or tabulating data under categories for any one of the above components for each maintenance action path diagrammed.

Furthermore, maintenance actions may be performed on various classes of track. Since different amounts and degrees of maintenance may be needed for each of the different FRA track classifications, cost coding of the present methodology provides for associating expenses with any one of the presently required FRA classifications. Classification of track is primarily by speed of operation where:

	The Maximum Allowable Operating Speed for Freight Trains is —	The Maximum Allowable Operating Speed for Passenger Trains is —
Class 1 track	10 m.p.h.	15 m.p.h.
Class 2 track	25 m.p.h.	30 m.p.h.
Class 3 track	40 m.p.h.	60 m.p.h.
Class 4 track	60 m.p.h.	80 m.p.h.
Class 5 track	80 m.p.h.	90 m.p.h.
Class 6 track	110 m.p.h.	110 m.p.h.

However, several tolerance limits of track geometry must also be held by FRA class (see Reference 10). If data are not available by track class or require other components to be handled, this can be done. By changing two numbers in the "IFILE" data set, the number of track components and track classifications handled by the program can be altered. Dimensioned program arrays may have to be enlarged if more than the above data subcategories are needed for a given cost simulation. For resizing of program arrays, see Appendix F on file structuring.

3.3 Cost Descriptor Codes

Since maintenance expenses are often tabulated in many accounting practices under such headings as materials, labor and equipment, the simulation methodology includes a set of user definable cost data descriptor codes. See codes 30 through 49 of Figure 3-1. Every cost item entered to the program is referenced to these descriptor codes (user account numbers). Summary output costs are then categorized (if desired and selected) under these user defined cost codes (see outputs in Appendices I and J).

DATA DESCRIPTOR CODES

COMPONENTS	00		ISOLATED PARAMETERS	50	Time
	01	Jointed Rail		51	Federal Safety Standards
	02	Welded Rail		52	Speed
	03	Wood Ties		53	Load
	04	Concrete Ties		54	Mgt.
	05	Ballast		55	Accidents
	06			56	Weibull
	07			57	Shipping Rates
	08			58	Income
	09			59	Design Factors
TRACK CLASS	10	Class 0		60	Track Gradient Factors
	11	Class 1		61	Track Curvature Factors
	12	Class 2		62	Weather Factors
	13	Class 3		63	Crossing Factors
	14	Class 4		64	Switch Factors
	15	Class 5		65	Available Manpower
	16	Class 6		66	
	17			67	
	18			68	
	19			69	
PATH INFO	20	Quantity of Component		70	
	21	Quality of Component		71	
	22	Amount of Bad Components		72	
	23	Amount of Good Compnt		73	
	24			74	
	25			75	
	26			76	
	27			77	
	28			78	
	29			79	
COST CATEGORIES	30	Use Quality in Costing	MISC. CODING	80	
	31	Material Costs		81	
	32	Equipment Costs		82	
	33	Fines		83	
	34	Delay Costs		84	
	35	Scrap Return Costs		85	
	36	Contracted Costs		86	
	37	Delivery Costs		87	
	38	Travel and Living		88	
	39			89	
	40	Trackmen Labor		90	
	41	Mechanic Labor		91	
	42	Machine Operator Labor 1		92	
	43	Machine Operator Labor 2		93	
	44	Foremen Labor		94	
	45	Supervisor Labor		95	
	46			96	
	47			97	
	48			98	
	49			99	

FIGURE 3-1. ASSIGNED DATA DESCRIPTOR CODES

4.0 TRACK MAINTENANCE ACTION DIAGRAMS FOR COST MODELING

4.1 Maintenance Action Diagram Structuring

In simulating maintenance costs for the model, the user represents pictorially the way in which a railroad (or the industry as a whole) maintains its track. This schematic diagram consists of a set of maintenance actions which comprise those performed by the typical track maintenance of way department. Furthermore, these actions are linked with connecting paths (or lines) which show with arrows the order in which these actions are carried out. In addition to representing the order in which the actions are done, the diagram also shows the exact order in which the computer carries out its cost computations. The simulation cost modeling procedure calculates the cost for each individual maintenance action represented and thus the total cost to maintain the track system under consideration. Refer to References 12 and 13 for diagrams of other systems.

Track maintenance-related actions chosen for an example simulation are:

- Inspections and other processes which result in a need for maintenance repair work,
- Decisions to carry out maintenance work,
- Maintenance repair operations,
- Maintenance deferments,
- Manufacturing supply of new materials (for instance rail, cross ties, etc.)
- Renew or rework shop operations, and
- Material scrapping.

Each of these maintenance-related actions involves a complex network of tasks performed by several people of a typical maintenance of way staff. The present simulation methodology involves establishing the maintenance policies and/or procedures as they are actually done and in terms of how much (or at what rate) defined work is performed. The various maintenance actions modeled are defined explicitly by the labels on the action diagram; whereas, the rates of doing the work are defined by the data and functions associated with the paths.

Therefore, in order to complete the costing analysis it is essential that cost data for each repair action modeled be identified and available for input. Layed out in diagrammed form the above listed maintenance actions could look as shown in Figure 4-1. This diagram contains the essential features of the maintenance actions listed above. This figure and the more comprehensive one shown in Appendix K are presented for the purpose of demonstrating the simulation costing methodology. Each maintenance action diagram contains numbered paths which connect various points of the action diagram to one another. Each line (path) either represents an element of maintenance repair work or just links two or more actions to one another. During the simulation the calculations proceed as depicted in the diagram in that upstream paths are processed first.

Circled junctions (or nodes) of three connecting paths indicate or represent various decision making processes in the maintenance-related work. For example, circled junction (3) in Figure 4-1 represents an inspector's decision as to whether he accepts the track he has inspected (path 1) and allows its continued use (path 2) or labels it as being in need of repair (path 18) by placing a "slow order" or other restriction on its use.

Junction number (5) in Figure 4-1 represents the decision making process whereby all track identified as being in need of work (path 3) is either passed to the point in the system where it is maintained (path 5) or where work is deferred (path 4).

Areas of the diagram which are shaded represent major repair operations which usually require several paths for their description. The maintenance repair block shown in Figure 4-1 has two incoming and three outgoing paths which pass its boundaries. The connections of these paths are shown and merely represent the overall nature of performing work on track components which have renew (or rework) shops, manufacturing, and scrap yards associated with them.

In Figure 4-1 junctions (6), (7), and (8) represent repair actions of adding new parts to the track system and removing worn or degraded defective parts. Paths 6, 7, 8 and 10 are used to keep tabs on the various rates at which

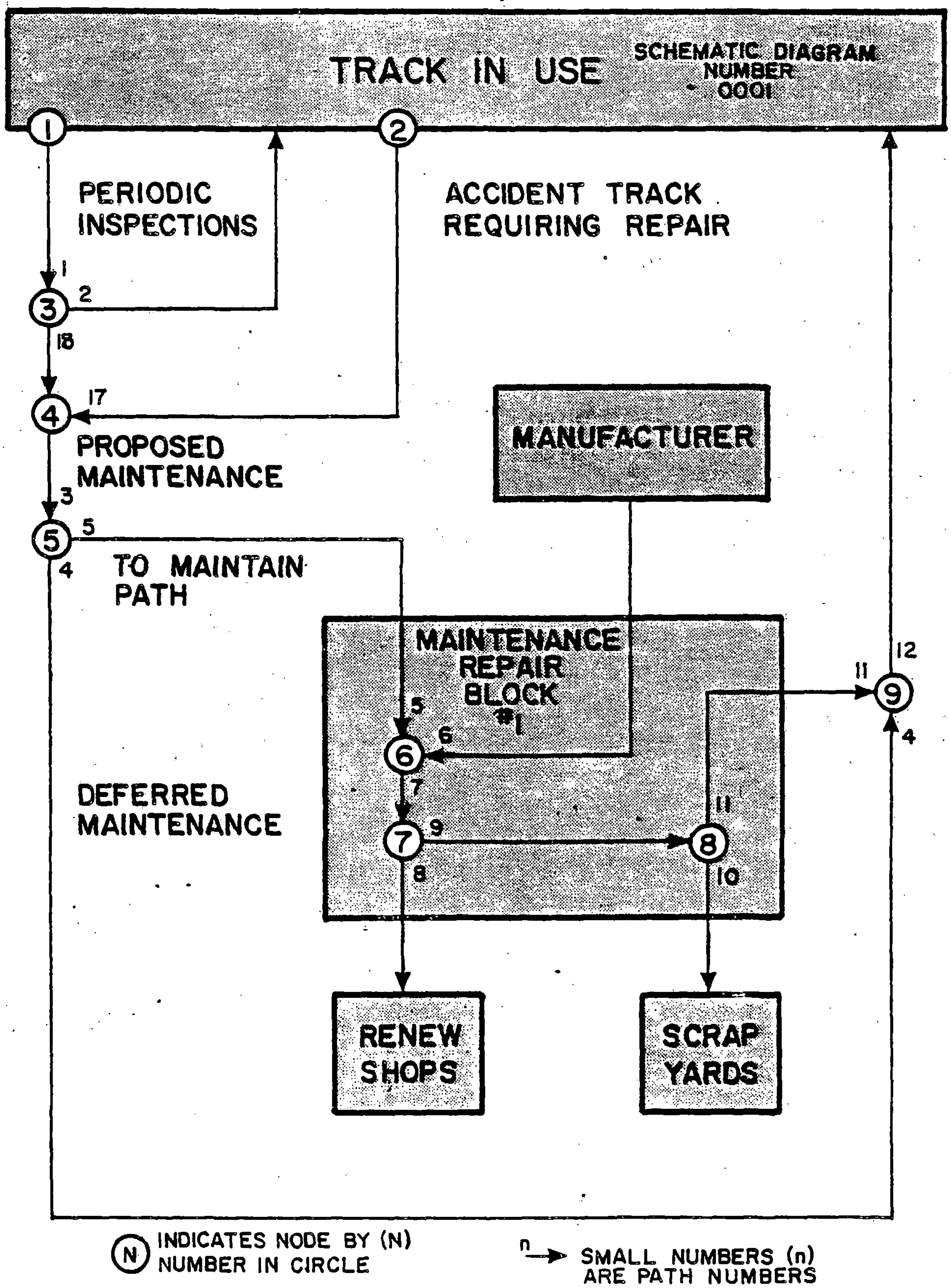


FIGURE 4-1. SIMPLIFIED MAINTENANCE ACTION DIAGRAM OF TRACK USED FOR EXAMPLE RUN NUMBER ONE

the following maintenance costs are expended:

- Path 6 - New parts for materials installed
- Path 7 - Labor, equipment, and other costs
- Path 8 - Costs of converting degraded materials to reusable ones (for example, making old jointed rail into continuous welded rail)
- Path 10 - Scrap return costs from materials removed from use.

Setting out in block form the various types of repair work has several advantages in simulating maintenance costs. One advantage is that each repair block may be as complex or simple in construction as necessary and the internal details of the blocked actions might not be shown at all. Examples might include remote rework shops where components are sent for repair. If the repair rates and costs are known the details of the maintenance work in the facility might be skipped and the applied costs be attached only to the maintenance actions as one lump sum or cost per component repaired. This blocking procedure allows the major maintenance actions to be initially simulated and then updated in more detail at a later time when more detail of the repair operation is learned and/or desired.

4.2 Action Diagramming of Track Inspections and Accidents

The need for maintenance repairs on a given section of track can be generated through inspection reviews of the track or through accidentally occurring incidents. The upkeep and maintenance of track which has been torn up from accidents, washouts, or other environmentally caused events can be costed by the simulation technique. An example of one way to account for expenses of maintenance as a result of accidents is depicted in Figure 4-1 with the linking path number 17. This path is intended for the tabulation of costs required by putting track into acceptable repair after an accident/incident. Data needed for tabulating these repair expenditures can be obtained from FRA reports of annual national reports of all accidents which occur.

Periodic inspections performed by track personnel are made in accordance with schedules set forth in the FRA Track Safety Standards. These inspections are intended to identify potentially unsafe track conditions, as well as yield information about the relative state of deterioration of the track as judged by the people doing the survey. These inspection judgments can often be used for planning or scheduling maintenance work when the track degrades to the limiting condition of compliance with set safety standards or increasing costs.

In complying with either federal inspection standards or with company standards (which can oftentimes be more stringent), unscheduled maintenance repairs must sometimes be performed. Existing standards require repair action be initiated immediately if track is found to be outside the limits of the standard (see Reference 10). In some instances, where repair cannot be performed immediately, a slow-order is placed on that section of track.

Track inspection processes may be diagrammed in a simple way as in Figure 4-1 or in a more complex manner as shown in Figure 4-2. The simplified approach is the single inspection review process practiced by many roads. In such a road the inspector will travel the road on a timely basis (once a day, week, or month) and note various track deficiencies. Within a short time the noted required repairs are made by the maintenance of way staff. Thus, the equivalent of the sent "to repair" operations and the other repair block features of the maintenance system would be accomplished.

Railroads (or specific track structures) may require a more involved inspection process. For example, some roads employ automatic systems for rapid scanning inspections of the rail system; and then back-up the measurements with an on-site personal inspection to establish the exact need or amount of repair. This type of procedure would be more like one of the two or three step inspection processes shown in the diagram of Figure 4-2. Many other examples might be given. Appendix K shows a simulation diagram which has eleven types of inspections, all in the simple one loop arrangement.

The kinds of inspections shown in Appendix K, which include:

1. Federal visual inspections
2. State visual inspections

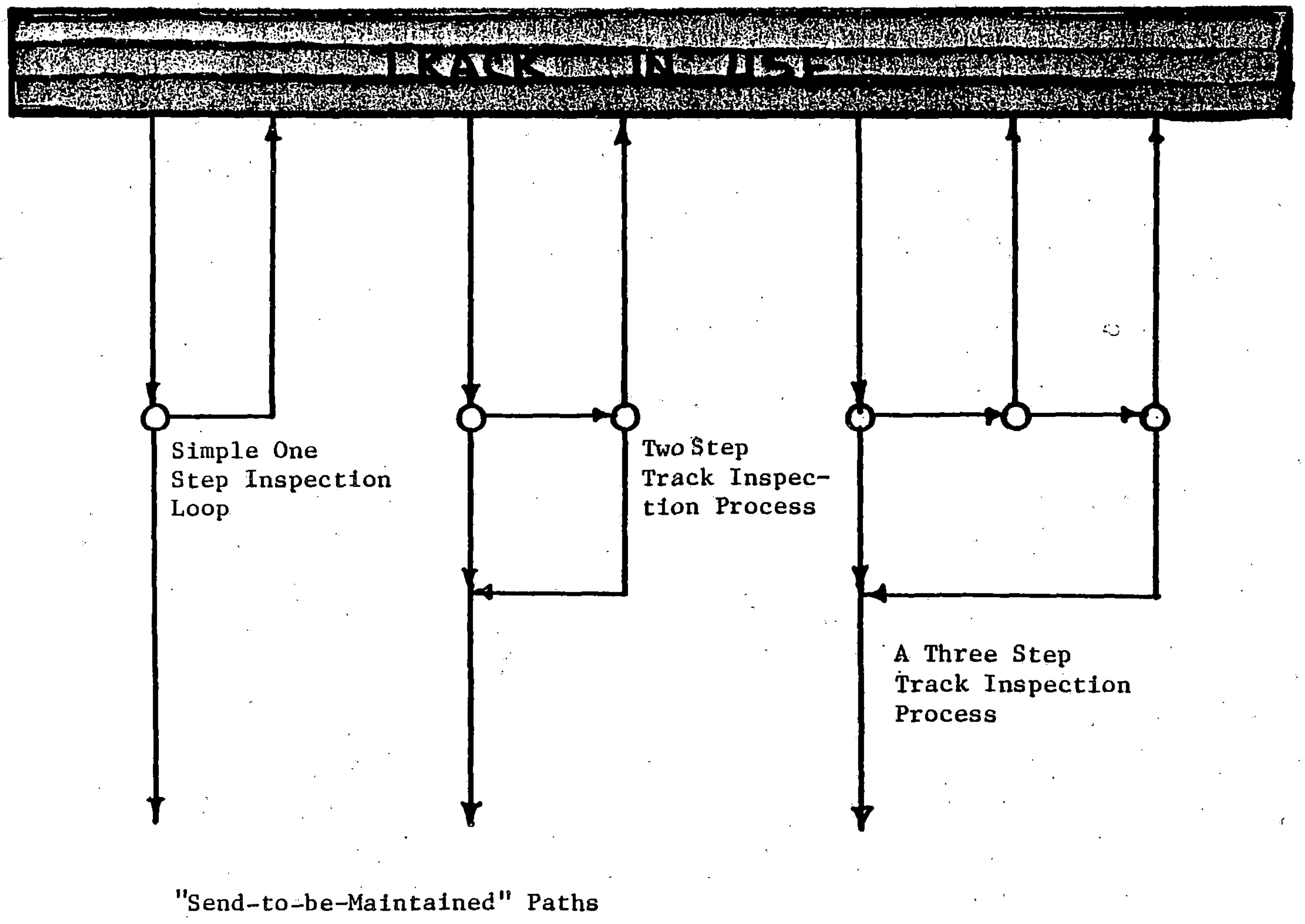


FIGURE 4-2. DIFFERENT WAYS OF MODELING VARIOUS INSPECTION PROCESSES

3. Scheduled track reviews
4. Federal inspection cars
5. Sperry rail cars
6. Sonirail inspection cars
7. Track geometry cars
8. Rail wear cars
9. Track inspected after accidents
10. Track reviewed after weather damages, and
11. Joint inspection cars

are not so much a model of one road's inspection practices as they are a representation of the nation's rail system as a whole. The intent of the inspection portion of the diagram is to account for the costs of inspection. In doing so, each of these represented "inspection loops" could have various amounts of track, ties, or ballast associated with it annually.

Furthermore, each of these "inspection loops" can be separately used to account for separate inspections on all or just some of the six FRA track classes modeled. Thus, each inspection loop may have in the modeled simulation data up to thirty (5x6) separately specified inspection rates. The model allows for separate inspection rates for each of the five components (i.e., jointed rail, welded rail, wood cross ties, concrete cross ties, and ballast) as well as for a separate inspection rate for each of the six classes of track.

4.3 Maintenance Decisions and Deferment Policies

Most maintenance decisions are modeled with diagrammed branching points. These circled points in the maintenance system provide for alternate decisions to be made on the components represented by the paths drawn. These split points could represent managerial actions or policies which might cause, for example, all track inspected and found defective with cracks to be replaced, or up to 5% of the ties in use to be replaced annually, or any other set of conditions to be placed on the repair and up-keep of the track. It is not the intention here to examine all the various maintenance practices of the rail industry. It is sufficient to say that there are many ways to keep a track maintained in such a way as to keep the roadbed structures (rails, ties, etc.) in acceptable operating condition both from a business standpoint as well as from a safety standpoint.

The present simulation cost methodology models maintenance policymaking procedures (if they are known for a given railroad maintenance system) through the use of:

- Simple split points in the maintenance action diagram; or
- Functional equations which describe the policy that causes a given repair operation diagrammed to be executed.

Examples of functions being used in the simulation are given in Section 6.

The deferring of maintenance work can be handled in the present methodology in several different ways. In simulating a specific railroad more than one of these methods may have to be used in order to represent accurately all deferring policies and procedures of that road.

The modeled maintenance system shown previously in Figure 4-1 takes the direct approach of setting aside one path which represents the rate at which maintenance work is delayed for a later period. The costs tabulated on this path would represent the net savings (or loss) from two economic factors. On one hand, there would be a savings since some real amount of maintenance labor is eliminated. On the other hand, repeated maintenance deferment could cause the track structure to degrade to the point where business related revenues may be adversely affected. Degraded track conditions might cause:

- delays in customer deliveries,
- more costly deliveries,
- more frequent train equipment breakdown,
- more expensive repairs when they are finally performed.

Additional approaches to getting an estimate of deferred maintenance costs could be taken with the simulation cost methodology. Although they have not been employed in the costing demonstrations discussed here, these would involve:

1. Setting up several maintenance repair blocks which represented the same repair action but would have costs based on the amount of deferred work accumulated.

2. Use of functional equations which would modify the maintenance costs through the amount of maintenance deferred.

4.4 Manufactured Supplies and Scrap Materials

As mentioned earlier, the present simulation cost methodology is in some ways analogous to the flow of fluid in a complex network of water pipes. Supplying and scrapping of materials in the cost analysis have their counter parts in sources and sinks of the water pipe analogy.

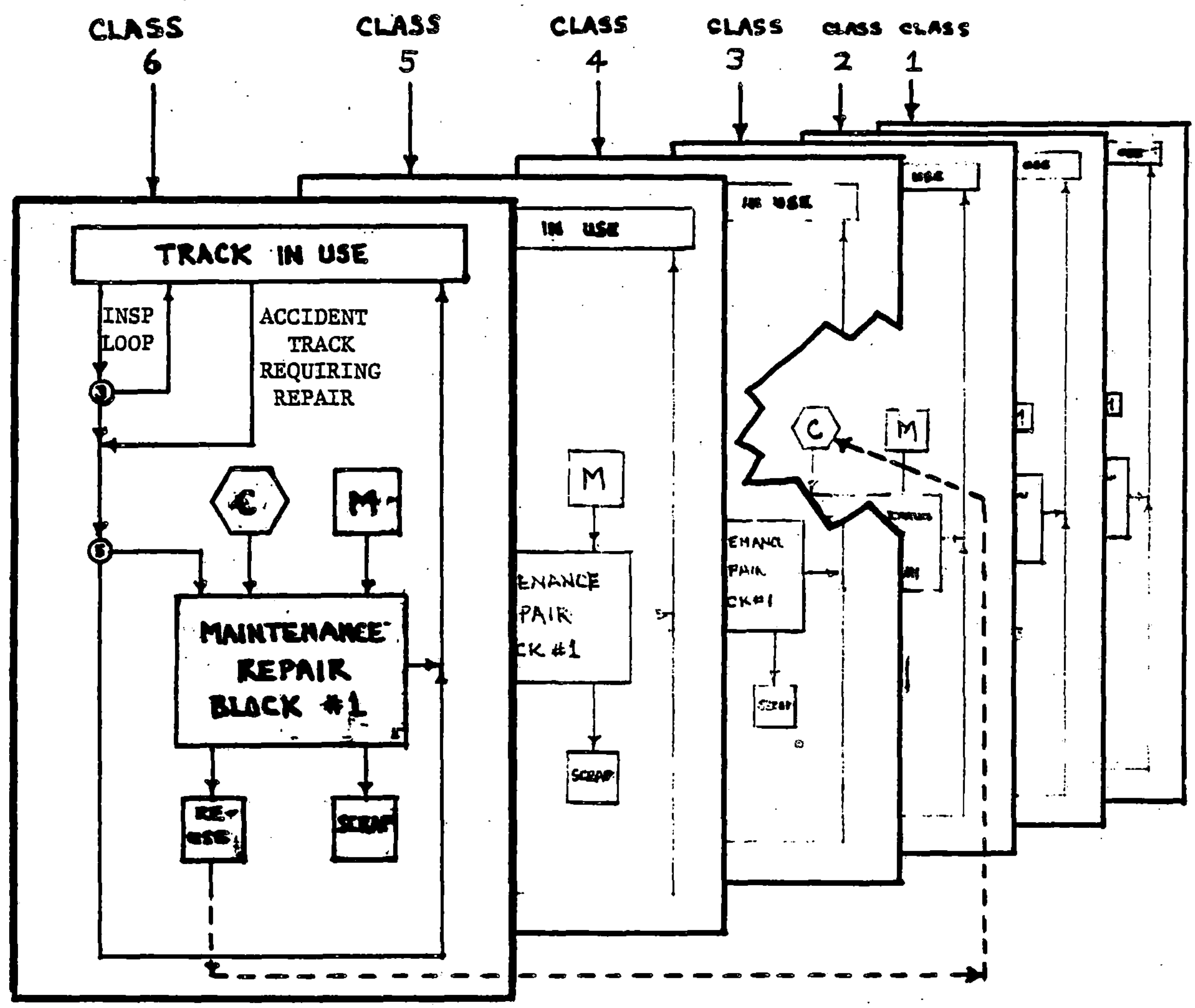
The schematic diagram of the simplified maintenance cost system previously shown in Figure 4-1 has three points associated with material origination or removal. The box labelled "component manufacturer" and its attached path (path 6) is where those purchase costs of new materials would be tabulated. Scrap yard paths (path 10) in this case are used to acknowledge and accumulate material return costs realized from selling materials removed from use (e.g., metals).

The label "renew shop" is used for expenses related to putting degraded components such as jointed rail into a usable condition as continuous welded rail. This procedure involves transportation and delivery costs, as well as shop welding expenses in this simplified maintenance picture.

4.5 Cascading and Reuse of Maintained Materials

If rail cascading is handled in a cost simulation, then some rail which is removed from a higher class (say class 6) could be reused or installed in a part of the rail system represented by a lower class (say class 3 as displayed in Figure 4-3). In this example the rail leaving the class 6 maintenance repair block along the "reuse" path would appear or be the source "cascade" for the rail being installed in the FRA class 3 maintenance block. Figure 4-4 details one way the "Maintenance Repair Block #1" might be layed out for systems that contain rail cascading in their maintenance operations.

This, of course, is only one way to handle cascading of track components through the use of the simulation cost model. Another way might involve the drawing of a single large schematic diagram which had "regions" or



Notation: M = MANUFACTURER'S SOURCE
C = CASCADING REUSE

FIGURE 4-3. CONCEPTUAL LAYOUT OF SIMPLE SCHEMATIC DIAGRAM SHOWING ONE DIAGRAM FOR EACH CLASS OF TRACK

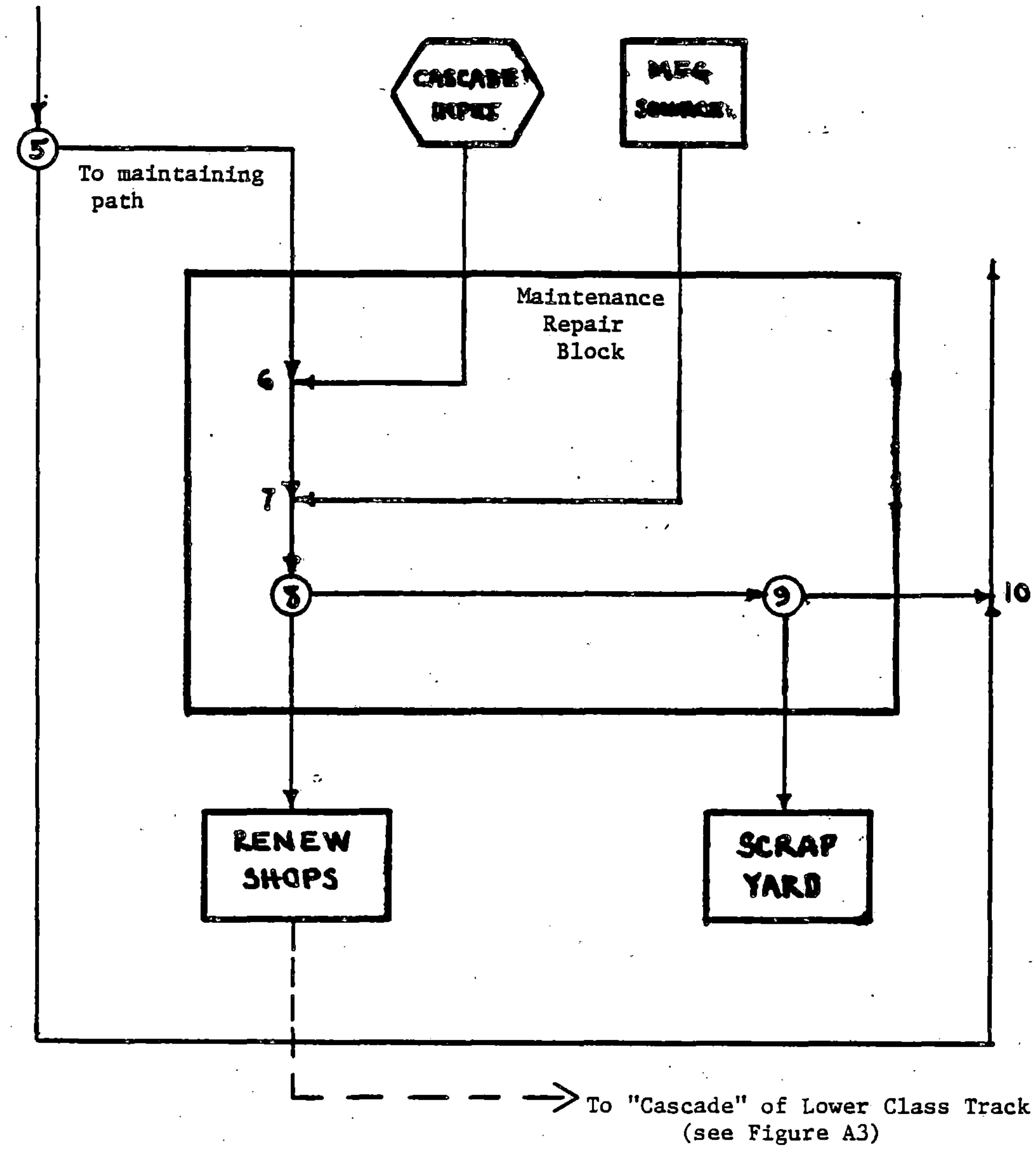


FIGURE 4-4. DETAILS OF A MAINTENANCE REPAIR BLOCK DRAWN TO HANDLE SIMPLE CASCADING OF RAIL

"blocks" of paths which are tailored separately to each class of track maintained. Then the movement of cascaded materials would be followed from "class" to "class" with separately drawn lines or paths.

5.0 TRACK MAINTENANCE COST DATA

A complete set of costing data for the track maintenance cost model will consist of the following information:

1. Annual starting number of components handled on each path of the Maintenance Action Diagram.
2. Quality information on components.
3. Unit cost on each path for the maintenance action cost categories required for the simulation diagrammed.
4. Functional parameters for the above entries needing parametric modifying relationships.

5.1 Model Path Component Quantities

Since the paths of the simulation model represent maintenance actions, these paths may have various amounts of each track component being acted upon. The simulation process requires the annual amount of each component repaired, worked upon, or otherwise being handled by each path at the start of the simulation. The quantity of each path component is entered as shown in Figure 5-1 (see data descriptor code "20").

As in the case of the fluid pipe analogy it is possible to under-define the system flows by specifying the flows within too few pipes of the system. In the case of the cost model, an insufficient data set would require the user to provide more data. A necessary minimum data set provides component quantities for:

1. Each path directly attached to or emanating from every circled split (branch) point in the diagram, and
2. Each manufacturing or scrap path represented.

It is not necessary to specify quantities on paths which emanate from summation points in the diagram unless these paths are connected directly to a branch point.

From these initial quantities the computer program can compute and store all system annual repair rates (flows) for every path diagrammed and can store them for simulation cycling. The subroutine entitled "RAWDAT" uses this particular

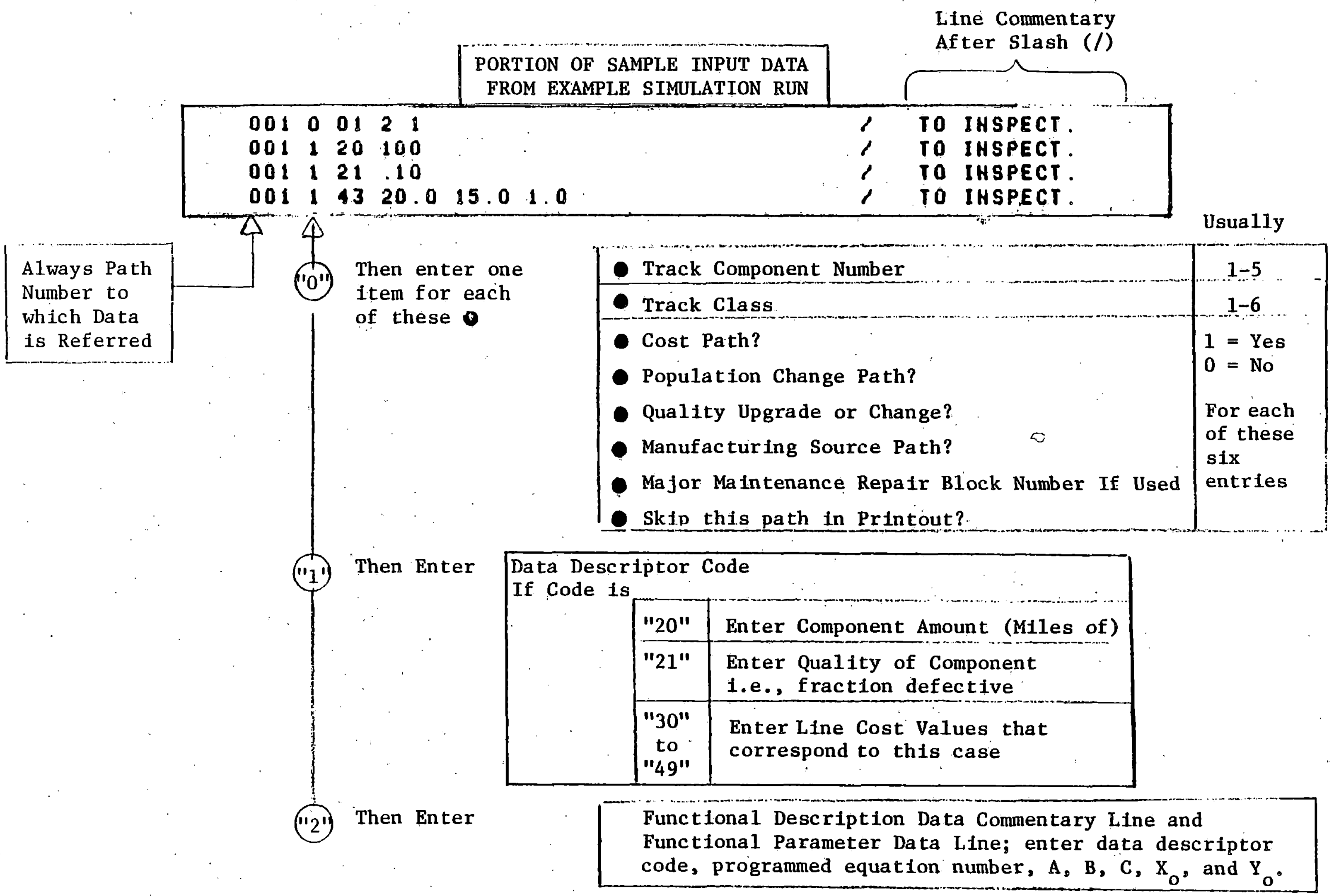


FIGURE 5-1. SAMPLE INPUT DATA DESCRIPTION

data entry format. However, alternate entry formats could be generated if the user wanted to take input data directly from his own computer stored accounting records without retyping the information.

5.2 Cost Data Forms, Sources, and Types

Costs for simulating maintenance work can be found in many forms and can be obtained from many different sources. Such sources as:

- Standard accounting records,
- Policy definitions,
- Business practices,
- Accident records,
- Component failure histories,
- Standard maintenance practices,
- Safety compliance standards, and
- Published reports

can be used to determine various input cost numbers for simulating track maintenance operations.

Cost data, however, must be entered for simulation purposes as shown in Figure 5-1. These initializing costs are all entered directly after codes "30" to "49". Each cost set entered is associated with the data descriptor of the same number listed in Figure 3-1 and detailed further in Appendix G (in particular refer to Figure G-2). Costs are entered on a per unit basis. Each assigned cost designator code has its own unit of measure. If the cost entry is material based then the three cost numbers needed are: number of material units per mile, dollars per material unit, and a multiplying factor (usually one (1.0)). If the cost entry can be based upon hours then the three inputs required for each descriptor are: hours per mile, dollars per hour, and number of people (or a multiplying factor).

The entries in each case provide a unit price rate (dollars per mile as it now stands) for every maintenance action expensed on that path of the diagram. By entering the costs on a unit basis they can be easily scanned at a future time and updated automatically where appropriate with the aid of the computer editing system. For example, if the labor rates (dollars per hour) increase then the data file can be called up and changed as needed.

5.3 Concept of System Component Qualities

Track substructures (i.e., rails, ties, etc.) are considered defective for modeling purposes if some identifiable work needs to be performed on that track in order to bring it within an acceptable level of use. These "acceptable levels of use" are typically defined by some safety standard under which the railroad operates. A set of track structural qualities has been chosen as a numeric way to define the fraction of the track system that needs repair work. Two alternate definitions of quality have been defined and either can be used. One's precise definition need only be made clear when setting or adjusting the numeric parameters in the computer simulation.

At each time step the maintenance simulation model provides for the replacement of defective components with new ones, such as installation of new track for worn out track. The modeling procedure, however, also allows for maintenance practices that put track structures back into a "like-new" user condition without component replacement. Track realignment, surfacing, and brush control operations are examples of this type of maintenance practice.

Track defect rates of growth are analytically generated during the simulation with a set of Weibull* (Refs. 9, 11) defect distributions for each track sub structure modeled. Track system degradation is in the present simulation procedure derived from accumulating defects that are generated from natural causes during a time step, as well as from neglecting to replace all defects that are present in the system at the beginning of the time step (maintenance deferment). Each component is modeled with its own failure rate distribution. The rate at which defects are created can be modeled as either constant or changing with time. In the present examples the Weibull parameters were chosen to keep the defect generation rate constant with time. Data entry examples for failure rate generation of each component for each FRA track class are given in Appendix F (p.F-16) and the Weibull distribution given explicitly in Appendix E (p.E-3).

*The Weibull distribution is an extreme value distribution for the smallest time to failure from a large sample of times from a given failure distribution.

Track component quality (Q) in the present model is defined as a ratio of two measures, the total defective unit amounts (miles) of a given component divided by the total unit amounts (miles) of that component in the system. This measure has the advantage of simply separating components into two defined "states"; good and defective. This ratio measuring concept of quality is displayed in Figure 5-2. As depicted, this particular concept does not portray the defect concept measure of "how bad is it". Since some repairs or levels of repair may require a measure of defect severity, other "quality" or track component condition measures may have to be developed. This would not be difficult as long as the concept of "quality" as defined for the simulation remains consistant throughout the process of data gathering and output interpretation of the results.

One alternate concept of quality which is possible defines it as a frequency of occurrence of defects per unit measure (mile) of that component. This measure has the advantage of indicating the defect severity on the basis of defect density (number/per mile) of the average defect level for that component's population.

One way of obtaining the advantage of either measure of quality; i.e., defect proportion based or frequency based measure, is to adopt a minimum unit length (e.g., 39 ft.) which is associated with each defect. In doing so, each definition of quality can be obtained from the other; i.e.,

$$\left[\text{Defective Proportion Based Quality}\right] = \left[\text{Frequency Based Quality}\right] * \left[\text{A Constant of Proportionality}\right]$$

In other words,

$$\text{Defective Proportion} = \left[\frac{\text{Number of Defects}}{\text{Standard Length, Mi.}}\right]\left[\frac{\left(\frac{\text{Minimum Defect Length, ft.}}{\text{Defect}}\right)}{\frac{5280\text{ ft.}}{\text{mi.}}}\right]$$

*In the present form of the cost simulation methodology, the defect proportion based definition of quality has been used. At the present time, this requires the user to structure his track "quality" data in terms of that definition.

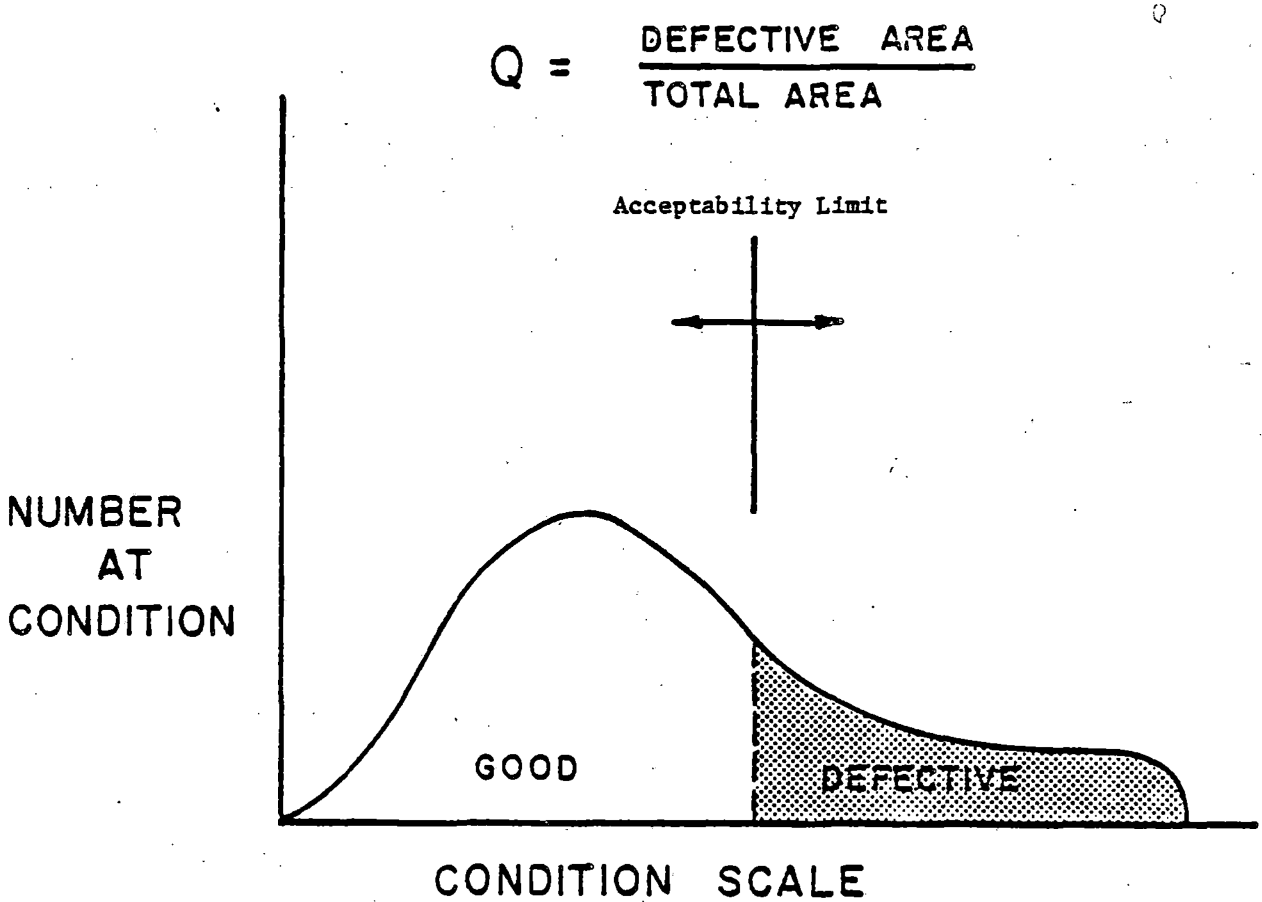


FIGURE 5-2. COMPONENT QUALITY AS A MEASURE OF SYSTEM CONDITION

5.4 Functional Parameters and Their Forms

The majority of cost models developed to date lack a method of evaluating the cost effects of certain parametric alternatives. The cost advantages (or disadvantages) in terms of track maintenance expenditure caused by higher traffic densities, changes in Federal Safety Standards, or an alternate component design, are not usually handled in the traditional cost modeling procedures. The present model addresses these potential variations in the operating system through the identification and use of functional relations.

If a cost or rate of track repair has a known dependence which can be written in terms of either the state variables (component population and component quality) or time, then use can be made of several preprogrammed families of functions which will carry out the definable dependencies during the simulation. Seven families of curves have been already defined (refer to Appendix E). Families of curves; i.e., linear, power, exponential, and others have been programmed for activation and can be "turned-on" through the use of various data codes. Figure 5-3 shows an example of data entries which makes use of a modifying function (number 7, Appendix E). This equation is used in one of the sample computer runs discussed in Section 6.

In order to "activate" the preprogrammed functions the user must enter two lines of data, as shown in Figure 5-3, directly after the data elements (costs or flows or qualities) to be modified by the functional expression. This procedure allows basic data elements to be modified in different functional ways by simply entering new functional parameters rather than new data values themselves. Overriding functions onto base data can be easily "added" or "taken-out" of the analysis by simply putting in or deleting the two data lines required per function applied.

If the programmed equation has a user selectable dummy variable (see first five equations, Appendix E) then the dependent function can be linked to the independent variable "X" through the data designator codes 50-79 listed in Figure 3-1. Although simulation time (code "50") is the only code presently

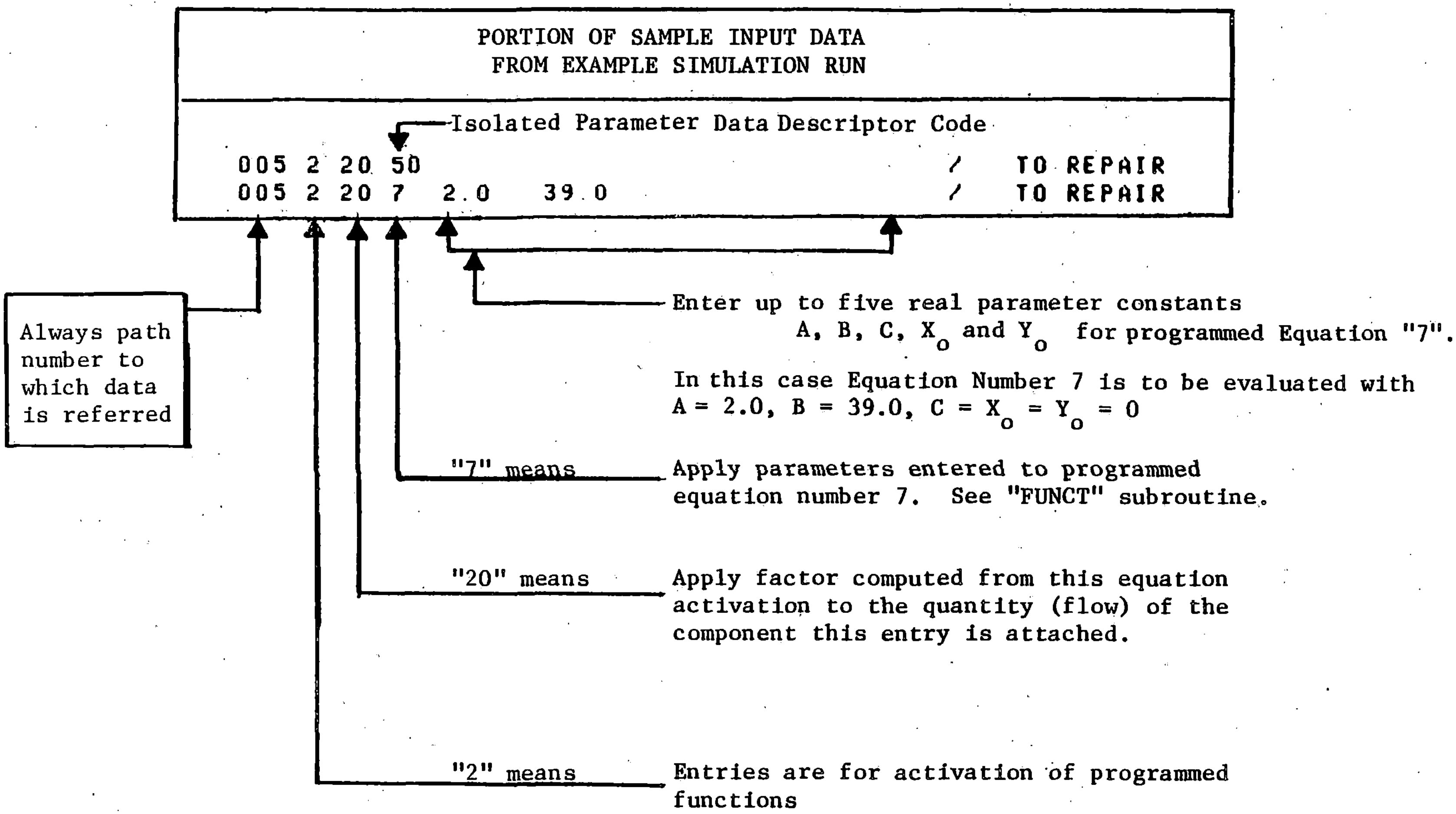


FIGURE 5-3. PORTION OF SAMPLE INPUT DATA FOR USE WITH PROGRAMMED MODIFYING FUNCTIONS

updated during the simulation it is intended that other isolated parameters such as speed, load, millions of gross tons (MGT), etc., could also be step-wise defined as independent variables.

For example, if one wanted to "modify" the annual amount of rail inspected over a ten year period as depicted by the dashed line on the uppermost graph of Figure 5-4, then he would enter the two data lines as shown in Figure 5-5.

The net results to the simulated systems "rail quality" and cumulative maintenance expenditures caused by this "modifying" function on inspection rates is shown in the lower two graphs of Figure 5-4.

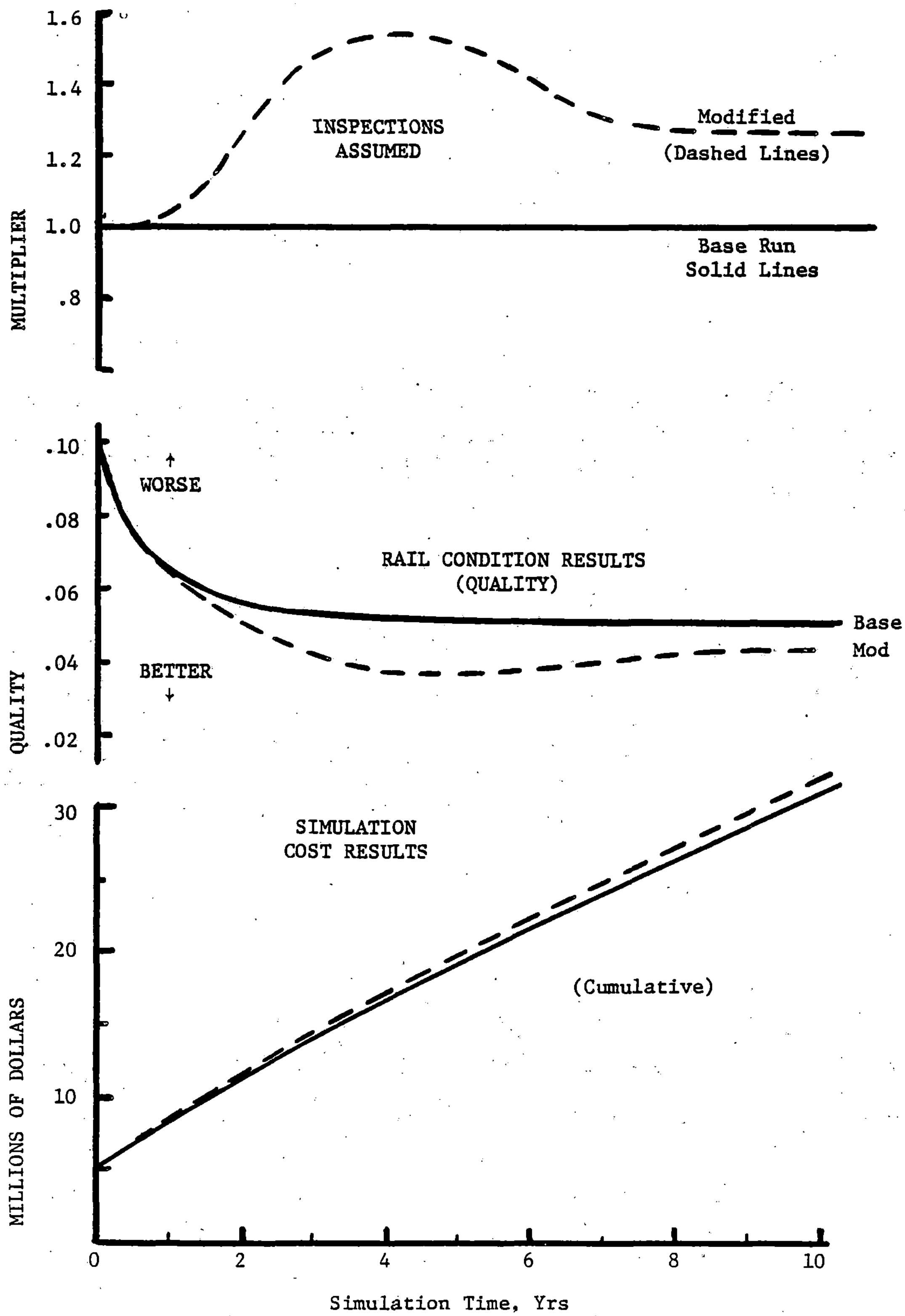


FIGURE 5-4. SIMULATION OF MODIFICATIONS OF INSPECTION RATE MULTIPLIERS AND RESULTING RAIL QUALITIES AND SIMULATION COSTS OBSERVED FROM COMPUTER RUN

PORTION OF SAMPLE INPUT DATA FROM
EXAMPLE SIMULATION RUN

```
011 1 20 5                                  /   GOOD MTRL.
011 0 01 2 1 0 0 0 1                        /   GOOD MTRL.
011 1 20 10                                 /   GOOD MTRL.
011 0 02 2 1 0 0 0 1                        /   GOOD MTRL.
011 1 20 15                                 /   GOOD MTRL.
001 0 01 1 1                                /   TO INSPECT.
001 1 20 700                                /   TO INSPECT.
001 2 20 50                                 /   TO INSPECT.
001 2 20 5  2.0 2.0 1.0 6.0 5.0             /   TO INSPECT.
```

Two data lines, if inserted, will "modify" rail inspections as shown in previous figure over simulation time span. Isolated parameter "50" simulation "time" will be used as independent variable through Equation 5 of FUNCT subroutine

FIGURE 5-5. SAMPLE INPUT DATA FOR MODIFYING TRACK INSPECTIONS WITH TIME

6.0 SAMPLE TRACK MAINTENANCE SIMULATION COST MODEL RUNS

Two example case runs of the computer simulation model were chosen for inclusion in this report. They include:

Example #1. A ten-year time based simulation which shows a comparison of two alternative inspection policies which are known to vary with time, and

Example #2. A test run which depicts a single time point costing (zero year) of a large hypothetical maintenance of way system.

The first example is intended to portray a small railroad whereas the second would be a larger, more complex system, such as that which makes up the full 200,000 miles contained within the United States. These maintenance systems were generated for example purposes; however, run number 2 contains many base line cost elements (see comment line inputs of Appendix H) which were drawn from References 1 through 7.

6.1 Background Information on Example Run #1

A hypothetical Alpha Railroad maintenance system was defined. This system is detailed by the information given in Figures 6-1, 6-2, Appendix G, and the previously presented maintenance action diagram of Figure 4-1. The example run consists of a set of base data plus three overriding functions which pertain to:

1. The amount of track inspected per year (function linked to time of simulation).
2. The number of accidents which can occur (function linked to defects in track).
3. The maintenance capacity of the maintenance of way department after the rail has been identified for repair (function linked to repair resources available at start of simulation).

6.1.1 Inspections

The inspection rate of rail can often be placed on a time basis (see Reference 10). This cycle time is usually on a weekly or monthly basis but can be more often in typical operations. The major expense of

Road Name: ALPHA RAILROAD Schematic Diagram Used: 0001	SYSTEM TOTALS	FRA TRACK CLASSES 1, 2, 3		FRA TRACK CLASSES 4, 5, 6	
		Component 1 Jointed Rail	Component 2 Welded Rail	Component 1 Jointed Rail	Component 2 Welded Rail
Track Systems Size	1000 miles	700 miles	50 miles	100 miles	150 miles
Track Accident Rate (Initial)	.01/mile	.011/mile	.02/mile	.01/mile	None
Track Accident Rate (During Simulation)	(per class only)	[Variable based on rail quality function]	.02/mile	.01/mile	None
Initial Track Quality	.1	.1	.1	.1	.1
Repair/Inspect Policy	100% Defects	100% Defects	100% Defects	100% Defects	100% Defects
Maintenance Deferment Policy Assumed	(per class only)	10%	~16%	10%	0
Maintenance Overload Capacity	(per class only)	[Variable based on simulation function]	Total	Total	Total
Average Component Age	20 years	20 years	20 years	20 years	20 years
Characteristic Component Life	32 years	32 years	32 years	32 years	32 years
Weibull Slope	2	2	2	2	2
Rail Reuse Realized	(per class only)	80%	80%	80%	93%
Rail Scrap Fraction	(per class only)	Net Unused	Net Unused	Net Unused	Net Unused

FIGURE 6-1. DESCRIPTION OF THE RAILROAD USED FOR SIMULATION PURPOSES

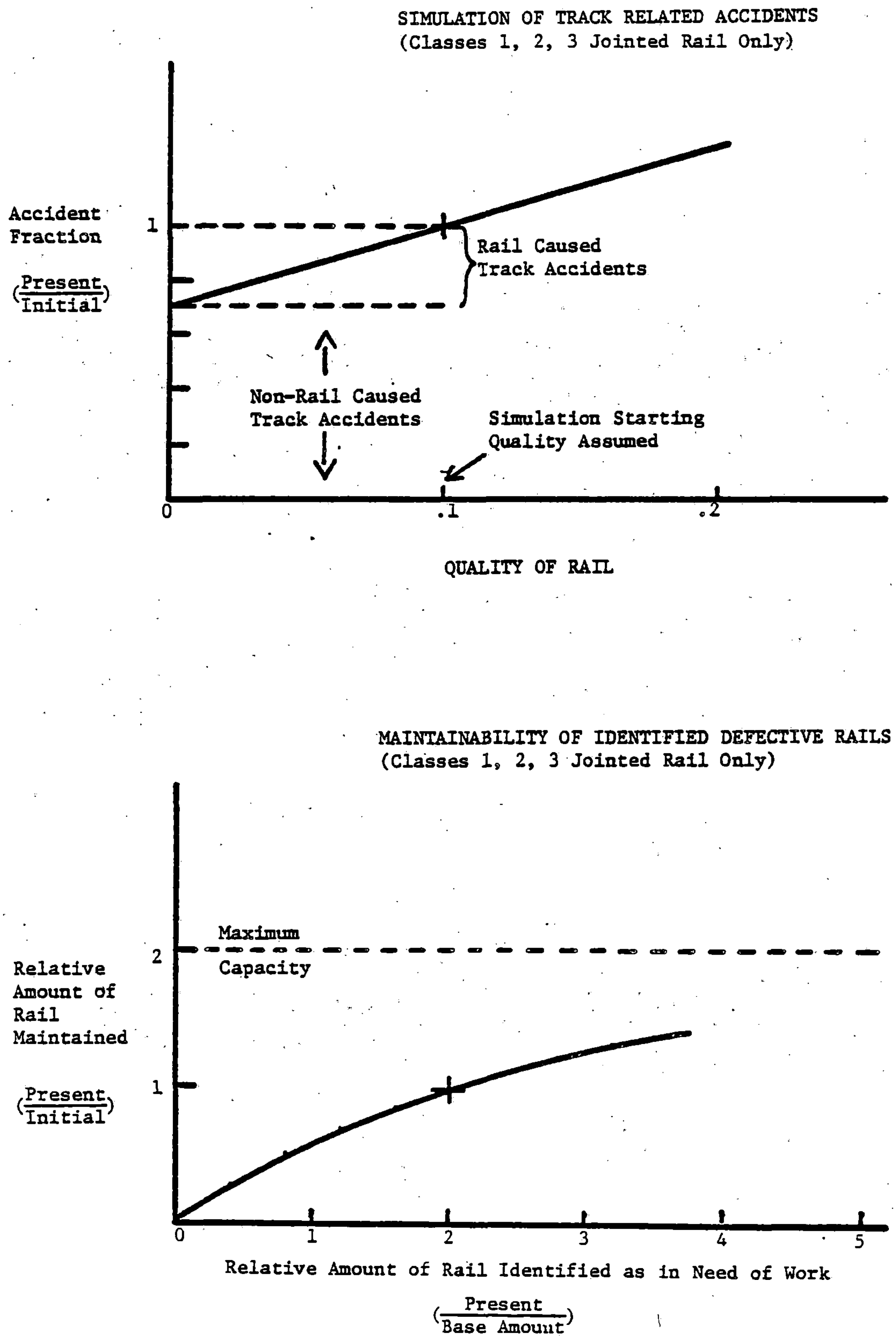


FIGURE 6-2. SHOWING FUNCTIONAL RELATIONS USED IN THE SIMULATION DEMONSTRATION

of inspection is the transportation and labor of performing the task. In some operations the inspector also has "immediate repair" obligations if they can be handled with the small amount of equipment he carries. The present model has two assumptions imposed upon the inspection cycle. First of all, the inspection loop sends toward the maintenance repair block all the "defective" track found during the inspection cycle. Secondly, the amount of track inspected during the simulation of this example is escalated on the schedule shown previously in Figure 5-4 of the last section. This variable amount of inspection based on time in this example, portrays the program's ability to alter maintenance action path flows through functional parameters (see input data for path 1, Appendix G). In addition, this inspection escalation with time follows one of two scenarios expected if certain future plans to rejuvenate the track system are executed, (Reference 8).

In a recent report by Dr. Orringer (8) of the Transportation Systems Center, it has been recognized that different remedial inspection procedures could be implemented in order to rejuvenate substandard track. Two alternate inspection policies were reviewed as new approaches to implementing modifications in standards such as the one outlined above.

These inspections procedures were labeled as the:

1. Interim Revision Plan, and

2. The Defect-Rate-Based Revision Plan

Either of these plans, if implemented, would have a time-varying impact on present day inspection resources. The net expected impact on these resources was depicted graphically in previously shown Figure 5-4 of Section 5.

In general, any formula which generates a simulated inspection procedure could be used. Two examples not exercised in the present program which could be implemented are shown in Figure 6-3. These inspection rates are shown to be linked to the quality of the rail in the system, but in general, could be associated with other variables such as traffic density or average car loading.

Two Non-Exercised Hypothetical
Inspection Policies Which Might Be
Based on Rail Qualities

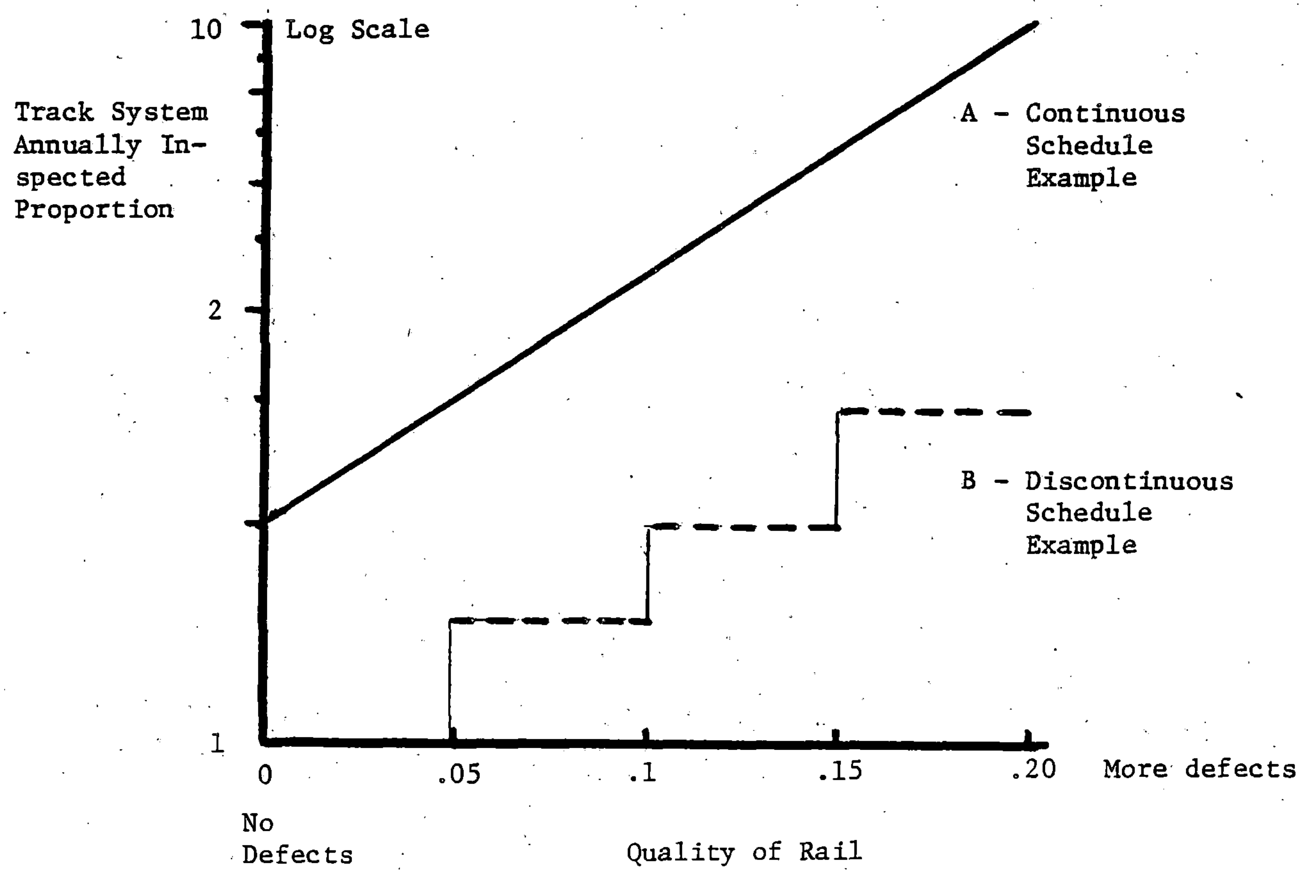


FIGURE 6-3. TWO EXAMPLES OF NON-LINEAR INSPECTION POLICIES WHICH COULD BE LINKED TO RAIL QUALITY

Each is depicted as increasing the amount of inspection as the rail gets worse. In particular, the discontinuous inspection schedule (plot "B", Figure 6-3) would have a tendency to create step like changes in the amount of track repaired, if the track passed toward the maintenance block were based on the amount of inspection performed. This type of simulation would imitate the case where the probability of finding defects in a system would be proportional to the amount of effort spent in reviewing the system.

Other policy concepts might also be added to alter the amount of rail sent toward the maintenance block from the "inspection loop". Such practices of sending (or finding) 50% of all defective rail in the system could be implemented. This type of simulation would represent an automated inspection car that has the feature of only finding some fraction of the defects because of its design limitations.

6.1.2 Accident-Related Track Mileage Repaired

For simulation, the Alpha Railroad was assumed to have accident-associated track and the mileage of that track was assumed to depend on rail quality. This dependence of the accident-associated track mileage was used only for jointed rail in FRA track classes 1-3. The dependence is given in Figure 6-2 as the ratio of the accident-associated track mileage to the initial mileage at the start of the simulation. The figure shows that the accident-associated track miles repaired goes down linearly as the condition of the rail improves (see Reference 14). The accident-associated track mileage for perfect rail condition (quality=0) is that associated with accidents not caused by the track but nevertheless requiring track maintenance.

6.1.3 Maintenance Capacities of Repair Organization

The example run #1 contains a function which provides for varying amounts of track to be deferred during the simulation. The selected function was chosen to reflect the fact that certain maintenance of way organizations have limited resources.

For this example, a simple limiting factor of two (2) over and above the originally defined rail defect repair capacity was chosen. The lower portion of Figure 6-1 shows the simulated association between the amount of rail designated as in need of repair and the amount which is passed along to the maintain path. As the amount of rail needing repair goes up, more rail is then deferred on the argument that the maintenance system is being stretched to more of a limit than it can respond.

6.2 Run Input and Output for Example #1

The input data for example run #1 of the simulation model are the contents of Appendix G. Shown in Appendix I are the outputs which include:

- Program Run Information
  - Data File Used
  - Simulation Description
  - Weibull Parameters Used
  - Active Functions with Parameters
- Annual Maintenance Costs by Simulation Time Point
  - Five Time Points (0-2 years)
- Track Maintenance Cost Summary Table
  - Costs by Designator Codes 31-49
  - Costs by Track Class
  - Costs by Track Component
  - Costs by Major Maintenance Repair Blocks

The program outputs for this simulation period reveal that:

- Maintenance expenditures per year decrease as the time of the simulation proceeds.
- The simulated track condition ultimately improves as time proceeds. The average qualities of each rail type by class drop to separate levels even though they were initialized to the same starting value.

In order to get an impression of how some features of the simulation influence the results, a similar run to the above was made. This alternate run was made with one change in the input data. Equation number five which modifies the inspection cycle was deleted. All other inputs were kept constant.

Deletion of the inspection modifying function provides an output for comparison between the alternate inspection policies assumed. See Figures 6-4 and graphs of previously discussed Figure 5-4 for comparison of results. In comparing the maintenance costs with (modified) and without (base) inspection modifiers it can be noted that:

- The modified Inspection Plan brings the track to a "better condition" more rapidly than the reference base simulation (see first two columns Figure 6-4).
- The modified simulation results in a track condition which is ultimately better than the base simulation maintenance practice (see Figure 5-4).
- The annual cost to maintain the track under the modified simulation conditions is slightly greater than the base maintenance case (see Figure 5-4).
- The total cumulative difference in maintenance expense after ten years with and without the imposed inspection modification is 3.1%.

In summary, these simulation comparison runs with time have shown some of the modeling capabilities of the present simulation cost methodology. The interrelated effects of changing inspection procedures and the realizable quality of the rail and the net associated cost results are shown.

6.3 Sample Run Number 2 Using a Large Data Base File

Maintenance cost operations data obtained from references one through seven were used to generate the elements of the input data. These input data are listed in Appendix H. The data values are keyed to the complex maintenance action diagram #2 shown in Appendix K.

YEAR	SYSTEM RAIL QUALITIES								TRACK MAINTENANCE COSTS				% Increased Cost
	Class 1,2,3 Track				Class 4,5,6 Track				ANNUAL RATES		CUMULATIVE		
	Jointed		Welded		Jointed		Welded						
	Base	Mod	Base	Mod	Base	Mod	Base	Mod	Base Run	Modification	Base Run	Modification	
0	.1000	.1000	.1000	.1000	.1000	.1000	.1000	.1000	5312979	5312979	5312979	5312979	0
.5	.0761	.0759	.0737	.0737	.0737	.0737	.0751	.0751	4009673	4035933	-------	-------	-----
1.0	.0651	.0640	.0585	.0585	.0585	.0585	.0603	.0603	3337871	3404821	8650850	8717800	.70%
1.5	.0597	.0568	.0497	.0497	.0497	.0497	.0516	.0516	2974604	3112344	-------	-------	-----
2.0	.0569	.0509	.0446	.0446	.0446	.0446	.0465	.0465	2773460	2990914	11424310	11708714	2.49%
2.5	.0555	.0456	.0416	.0416	.0416	.0416	.0435	.0435	2660741	2908688	-------	-------	-----
3.0	.0548	.0415	.0399	.0399	.0399	.0399	.0417	.0417	2597199	2819514	14021509	14528228	3.6%
3.5	.0544	.0389	.0389	.0389	.0389	.0389	.0406	.0406	2561299	2744324	-------	-------	-----
4.0	.0542	.0376	.0384	.0384	.0384	.0384	.0400	.0400	2541037	2694137	16562546	17222365	3.98%
4.5	.0541	.0370	.0380	.0380	.0380	.0380	.0396	.0396	2529659	2661725	-------	-------	-----
5.0	.0540	.0369	.0379	.0379	.0379	.0379	.0394	.0394	2523342	2634560	19085888	19856925	4.0%
10	.0539	.0435	.0376	.0376	.0376	.0376	.0391	.0391	2518858	2580861	31675236	32668898	3.1%

FIGURE 6-4. SUMMARY COST MODEL RESULTS WITH AND WITHOUT INSPECTION MODIFYING FUNCTION

This complex diagram is intended to portray a composite of several railroads since this model contains:

- Eleven separate modes of inspection, and
- Twenty individual major maintenance repair operation blocks.

This model might be used to represent, if desired, all railroads within the U.S. as a composite. Although some single railroads could be simulated with the complex diagram shown most roads do not have many of the features incorporated in this complex model.

For example purposes then, some of the data contained in Appendix H was devised for a hypothetical road of 200,000 miles in length and assumed to be comprised of "all" features of the complex diagram. Much of the maintenance gang work laid out in the various major repair blocks (see twenty separate operations of Appendix K) were drawn up from conversations with railroads consulted during the project. Individual manpower performances as well as material quantities and unit costs entered to the data file of Appendix H have been obtained from references 1 through 7 (see commentary of data file).

A summary output of the single time period from this hypothetical maintenance system is given in Appendix J. Notable features of this example run are listed both in the heading of the computer run output, as well as in the summary (last page of Appendix K).

Run assumptions include:

- This system is represented by a maintenance action diagram with about 575 paths and 201 diagram junction points.
- The system has 5 components:

  Jointed Rail
  Welded Rail
  Cross Ties
  Concrete Ties, and
  Ballast

- Due to the complexity of this data set only one FRA track class was assumed.
- No overriding functions were used in this example run to modify any of the initial data values.
- The total mileage was assumed to be roughly split 50-50 between welded and jointed rail.

Output features include:

- Summary costs of maintenance by path number of the modeled maintenance action diagram.
- Materials and labor make up the majority of the maintenance repair costs for this single year's expenses (see cost codes 31 and 40 to 45 of the Track Maintenance Cost Summary Table; last page of Appendix J).
- Rail is the most expensive of the five components to keep up.
- Maintenance repair blocks one through four account for most of the maintenance costs (these represent rail laying operations).

The execution of this program simulation requires about one to two minutes wall clock time, as opposed to the same time for twenty time steps used in example simulation run number 1 discussed previously. This time is for a PRIME 350 run with data being held on a hard disk drive unit. Each of these program simulations have been performed in less time with an altered "PATHIN", "PATOUT" subroutine than that shown in the listing of Appendix D. The altered subroutines used direct addressing schemes, to and from the computer disk which eliminated the required large computational "RARAY" used for storing the data elements during program run.

REFERENCES

1. Danzig, J. C., Hay, W. W., and Reinschmidt, A. J., "Procedures for Analyzing the Economic Costs of Railroad Roadway for Pricing Purposes," Final Report, Vol. 11, 1976, FRA, Washington, D. C.

2. "Yearbook of Railroad Facts," AAR, 1977.

3. "Official Manual of the AAR Interchange Rules," AAR, 1978.

4. Danzig, J. C., Rugg, J. A., Williams, J. H., and Hay, W. W., "Procedures for Analyzing the Economic Costs of Railroad Roadway for Pricing Purposes," Final Report, Vol. I, 1976, FRA, Washington, D. C.

5. "Track Cyclopedia," AAR, 1978.

6. Based on consultation with small railroads.

7. Based on consultation with large railroads.

8. Orringer, O., "Assessment of Impact of Proposed Revisions to Rail Inspection Portions of Track Safety Standards," U.S. Department of Transportation, January 1979.

9. Nelson, W., "Hazard Plotting Methods for Analysis of Life Data with Different Failure Modes," Journal of Quality Technology, Vol. 2, No. 3, July 1970.

10. FRA Safety Standards

11. Collacott, R. A., "Mechanical Fault Diagnosis and Correction Monitoring," J. Wiley & Sons, NY, 1977.

12. Krauter, A. I., and Smith, R. L., "A Methodology for Evaluating the Maintenance of High Speed Passenger Train Trucks," Final Report Contract No. DOT-TSC-1308, Report No. FRA/ORD-78/73 Dec. 1978.

13. Smith, R. L., Krauter, A. I., Betor, J., "User's Manual for the Maintainability Methodology for High Speed Passenger Train Trucks," Final Report, Contract DOT-TSC-1619, April 1980.

14. Pugh-Roberts Associates, Inc., "Preliminary Modeling Activities for a Simulation Study of Track Caused Accidents," Cambridge, Mass., Nov. 1978.

APPENDIX A

OPERATIONS MANUAL

1.0 GENERAL INFORMATION

1.1 Summary

The CONTRL program is designed to simulate the operation of a railroad track maintenance system and compute the changes in track inventory population and costs over a desired time period. The program has the capability of producing a base case for the system and outputting the state of the system at selected time steps.

1.2 Environments

This program was developed for DOT by Shaker Research Corporation under DOT-TSC-1594.

1.3 References

See Section 7.

2.0 OVERVIEW

2.1 Problem and Solution Method

Refer to example program runs 1 and 2 with data explanation in Section 6.

2.2 Program Inventory

2.2.1 Main Program: CONTRL

"CONTRL" is an extremely modular program for calling required subroutines as required through data selections specified.

2.2.2 Subroutines

Refer to Appendices C, D and E which consist of flow charts, listings and further explanations by name.

2.3 File Inventory

CONTROL/6	:	Source Program (Appendix D)
IFLAG	:	Diagnostic control file
IFILE	:	Integer Controls for Program Options
RFILE	:	Real Controls for Program Options
INODE	:	Diagram Path Linkage Designators
RNODE	:	Split Fractions for all junctions Array
FUNCT	:	Function evaluation Parameter Storage
STATE	:	System quantity and qualities
WEIBL	:	Component Failure Rate Distributions
DATA	:	User input cost operations information
OUTPUT	:	Receiving file for output storage.

3.0 DESCRIPTION OF RUNS

3.1 Run Inventory

1. Base Run: Given a set of system flow inputs and node decision parameters, the run will produce the corresponding system flows and costs by path and code.

2. Time Simulation Run: Given a starting population state, a time step, and an ending time, the run will use the Runge-Kutta method to determine the change in state for each time step. The technique requires sufficiently small time steps in order to maintain accuracy.

3.2 Run Progression

In all cases the Base Run is done first. To select other desired run options the control indice values stored in the IFILE and RFILE must be altered. See Appendices F, G, and H for examples.

3.3 Run Description

3.3.1 Control Inputs

The first eight files are read in through subroutine DATAIN on FORTRAN units 8, 9, 12, 13, 14, 15, 17, and 16, respectively. DATA is read in the RAWDAT subroutine on a file unit set from IFILE, so alter the IFILE data accordingly. Output is written to a unit set in IFILE as well.

3.3.2 Operating Information

a. Base Run: see input/outputs of Appendices H and J.

b. Run Time: 2.0 minutes CPU.
   Turn Around Time: 3.5 minutes total on PRIME 350.

c. Operator messages and commands: NONE

d. Trouble Shooting: All control is through IFILE and RFILE.

See example explanations in text and appendices or contact the authors.

APPENDIX B

PROGRAM MAINTENANCE MANUAL

1. General Information

1.1 Summary

This program was written under standard FORTRAN IV conventions and consists of a main program, subroutines called by the main program only, subroutines called by the main program and other subroutines, and subroutines accessed solely from other subroutines. There are no recursive calls. The program is supported by two types of data files distinguished by purpose, those with program control values, the rest with calculation data. The software package of main program, subroutines, and data files is completely self-contained except for the standard FORTRAN functions which must be provided by the user's compiler.

1.2 Environment

This program was developed for DOT by Shaker Research Corporation under DOT-TSC-1594.

1.3 References

"Life Cycle Cost Methodology for the Evaluation of Proposed Track-Related Safety Standards", Smith, R., Krauter, A., Shaker Research Corporation, Ballston Lake, N.Y., Final Report prepared for Department of Transportation/Transportation Systems Center, Contract No. DOT-TSC-1594, February 19, 1979.

2. Program Descriptions

2.1 Program Description - Control

2.1.1 Problem and Solution Method. The problem was to set up a mathematical model to simulate the maintenance system for track. The solution was to set up a diagram of nodes and paths. The paths represent various steps and operations of the maintenance process. The nodes, depending on type, either combine the flows on two incoming paths and send the combined flow down the single outgoing path, or determine how to split up the flow of an incoming path between two outgoing paths. Beginning with the information from some starting path, the program sweeps through the diagram, filling in the flows and qualities for all downstream paths. There are four basic path types, in-use, internodal, manufacture, and scrap. There is only one in-use path, it contains the system population, and quality for all components. Internodal paths represent maintenance operations or movement of components between nodes. Manufacture paths enter the system: the flow on each such path is always set equal the defective portion of the flow in the path it is summed with. The scrap path permits unrepairable units to leave the system. Any path that leaves, enters, or in any way alters the system quality or population, is monitored to keep track of total system change.

A single sweep through the diagram using equilibrium values for the start path, will produce a base run. By taking the system change in state from a sweep, and applying the Runge-Kutta technique, the program simulates operation through time.

2.1.2 Input (See Appendices G and H for listings). Input is from nine data files, read in the following order: IFILE, WEIBL, INODE, RNODE, FUNCT, STATE, RFILE, IFLAG, and DATA. The first eight are read in by the DATAIN subroutine. DATA is read in the subroutine RAWDAT. (NOTE: These file names correspond to those of the sample run provided with this manual. The file names may be changed to suit the user without affecting the program as long as they are read in the proper order.) Files IFILE, RFILE and IFLAG are of the control file type described in Section 1.1. DATA is actually a combination control and cost information with a control data line for every block of flow and costs. Detailed file descriptions and formats are presented separately in Appendix F.

2.1.3 Processing

a. Processing Logic

The main program is faced with two primary tasks. It must first read in, sort, process and store data to be used in the second task. It then must simulate the operation of a track maintenance system with the processed data. To accomplish these goals, the main program acts as a central control area (hence the name CONTRL) calling subroutines as required. The course of action taken is heavily dependent on the first subroutine called, DATAIN. DATAIN reads and stores most of the control variable values used (see Section 2.1.2 and Operations Manual Section 2.2 for greater detail). DATAIN also reads in precalculated population and node split values if base path data is not to be used.

If base data are used, subroutines INTDAT, RAWDAT, and NDVAL are called respectively to clear data structures, to process the base data, and to calculate node split values. This completes the first task, and if the data structures are permanent, the program can be halted at this point and restarted later. (Data structures will be altered to best suit the users machine by simply changing the PATHIN and PATOUT subroutines.)

The second task of actually simulating the system operation is handled by calling the SWEEP rubroutine (called MAIN in early versions of the program). SWEEP handles all aspects of node sweeps, time simulation, and output. In the batch mode version, SWEEP will be the final main program call before CALL EXIT.

b. Data Structures and Linkages

All data, with the exception of control variables, are stored and referenced by the corresponding class-component. Class-component always goes by column, with nodes, paths, and Weibull factors referenced by row. The large real data storage structure for path data may be envisioned as having blocks of columns for each class-component (even though it may be stored as a single sequential file with a calculated positioning).

There is only one instance of a linked or threaded list. In DATA2, FPLACE, and LNKLST functions are stored as read in, in a temporary file. A pointer array is sorted and after the last function for that class-component is read, the pointer array is used to store the function data in the FUNCT array in the proper order.

c. Variables and Constants

Any non-declared names come directly under the control of the IMPLICIT statement that leads the main program and every subroutine. This program was written by several people, with many key subroutines being started before nomenclature conventions were established. This resulted in some cases of the same variable having many different names in different subroutines. Despite this, several important variable names retain their meaning through most of the program.

IMPORTANT VARIABLES AND CONSTANTS

Name	Description
ISTND	Starting node, usually set to 1, but may be set higher for partial sweeps.
LSTND	Last node checked.
ISTPH	Lowest path number. Paths need not be numbered sequentially.
LSTPH	Highest path number.

Name	Description
ISTCL	First class number, like ISTND may be greater than 1 for partial runs.
LSTCL	Last class checked.
ISTCP	First component.
LSTCP	Last component.
LSTFL	Number of flow property columns in a class-component block of RPATH arrays.
ISTCT	Number of columns preceding first cost column in a class-component block (equivalent to LSTFL).
LSTCT	Number of cost columns following first.
IIDM	IPATH array dimension.
IRDM	RPATH array dimension.
IBDM	Class-component block size in RPATH arrays.
NDOUT	Final out path indicator.
NREFP, NRPTH	In-use path number.
IUNIT	Output unit.
IRP	RPATH array reference index for present class-component block. Calculated in RCOLUM subroutine.
IRCOL	STATE, RNODE, and WEIBUL array reference column for current class-component. Calculated in RCOLUM subroutine.
STIME	Simulation start time.
DTIME	Simulation time step.
LTIME	Simulation end time.
LNTHST	Number of STATE array columns.
IP1, IP2, IP3	Path numbers from INODE.
IBRSH	Branch node indicator.
ICDEF	Simple of defective decision indicator.
IREFP	Branch node reference path indicator.
IPDT1, IPDT2, IPDT3	Population change path indicators.
ISGN1, ISGN2, ISGN3	Population increase or decrease indicators.
IQDT1, IQDT2, IQDT3	Quality change path indicator.
IMFG1, IMFG2, IMFG3	Manufacture path indicators.

Name	Description
RCP, RCP1, RCP2, RCP3	Population rate of change.
RCQ, RCQ1, RCQ2, RCQ3	Quality rate of change.
ICOST, ICST1, ICST2, ICST3	Cost calculation indicators.
IPRNT, IPRT1, IPRT2, IPRT3	NORMAL print output indicators.
ICL	Current class.
ICP	Current component.
IIROW	Current node.
CDE	Branch node split value.
PCOST1	Total sweep cost.
DP, DPOP	Change in population.
DB, DBAD	Change in defective pouplation.
USEQ	User quality indicator.
BAD, BAD1, BAD2, BAD3	Defective flow.
QUAL, QUAL1 QUAL2, QUAL3	Flow quality.
FCOST	Function value.
IND1, INDL, I1 I2, I3	Do loop parameters.
J, L. K, etc.	Indexing, etc.

d. Formulas

Formulas and a full explanation of each may be found in Appendix E.

e. Error Handling

The program has built-in logic to avoid errors like divide by zero. Other errors, such as incorrect path numbers, will call the ERRORS subroutine allowing the program to continue. Major errors such as format/mis-match will cause the program to halt.

e. Limits

The program, as set up, cannot handle more than:

30 class-components

300 nodes (path junctions)

600 paths

2 path properties (flow and quality)

19 cost codes per class-component

To exceed these limits, the affected arrays must be expanded. If more path properties are desired, the program will have to be redesigned.

The program is also limited to one run type at a time. To change run type, the user must alter the control values in IFILE and RFILE to the appropriate run set up.

2.1.4 Output

Output is primarily done through subroutines OUTPT and HEADNG. OUTPT is called from many other subroutines to output path flow and costs, system states, and system costs. HEADNG is called from SWEEP prior to any output to write the output file heading. All other output, with one exception in NORMAL, will be diagnostic and in the users version could be commented out. Output device selection will be through variable IUNIT, which is set from the IFILE file. (See Appendices I and J for output examples.)

2.1.5 Interfaces

The program will input data from a maximum of nine different units, and output at most through two more. The base path data file, the diagnostic outputs, and the result outputs are set by values selected by the user and carried in the control file IFILE. The remaining data files are read by the DATAIN subroutine, on constant unit numbers; IFILE on unit 8, WEIBL on unit 9, INODE on Unit 12, RNODE on unit 13, FUNCT on unit 14, STATE on unit 15, RFILE on unit 17, and IFLAG on unit 16.

2.1.6 Arrays

Arrays are dimensioned somewhat oversized to allow for system expansion without changing the program. Most of the arrays can be separated into general categories, those with node oriented data, those with path oriented data, and those with system oriented data. Under the category of path data, some versions of the program have the RARAY and IARAY arrays in COMMON. These two arrays are necessary if no direct access file scheme is used to handle the large quantity of path flow and cost data.

Node Arrays

Name	Description	Where Used
INODE (6,300)	Inode has one column for each node. The first three rows hold the path numbers of the associate paths. The fourth row holds the sum/branch indicator, the fifth row is the decision type, and the sixth row indicates which path is the node reference path on branch paths. Kept in common block I7.	SWEEP, ABNRML, NORMAL, DATAIN, NDVAL
RNODE (30,300)	RNODE holds the flow split fraction for branch nodes. Each column represents the corresponding node with the one row per class-component. In the case of a sum node a zero is stored.	SWEEP, ABNRML, NORML, DATAIN, NDVAL

Path Arrays

Name	Description	Where Used
IPATH, IPATH1, IPATH2, IPATH3 (15)	An IPATH array by any other name would do just as much. Array holds path types and operation indicators.	NORMAL, ABNRML, OUTPT, RAWDAT, DUMMY
RPATH, RPATH1, RPATH 2, RPATH3, PATH (750)	Holds path flow and/or cost data. Array is divided up into equal sized blocks, one for each class-component. The first LSTFL columns of each block holds flow data, or are disregarded	NORMAL, ABNRML, OUTPT, BRANCH, SUMS, PDOT, QDOT, PCOSTS, ADDRCH, CDFUNC, FUNCTS, RAWDAT, DUMMY

Name	Description	Where Used
	when costs only are handled. The remaining columns hold unit costs, or cost function pointer, or path costs, the last when cost only are stored.	
FUNCT (10,10)	Holds path function data. The first three columns hold the path number, function number, and data code respectively. The fourth column records how many FUNCT lines following the current one are associated with the current function, the fifth column points to the location in the RFILE array where the independent variable is stored. The remaining columns are function parameters.	FUNCTS, RAWDAT, DATAIN
VALUE (50)	Array is used to sum and temporarily hold path data during base data processing. Each column corresponds to a data code. Unused codes are left blank. When data for a class-component block is all readin, VALUE is used to fill the appropriate spaces in RPATH.	RAWDAT, DATA1, DATA2, FPLACE, DUMMY
RL, REAL2, REALA REALB, REALC, READD	Buffer arrays into which base/data is first read, prior to sorting.	RAWDAT, DATA1, DATA2, FPLACE

System Arrays

Name	Description	Where Used
STATE (20,30)	STATE Stores the system population and quality for each class-component at the start of and at various points of each time step. It also contains the rates of change computed during the time step. The array can be subdivided into five blocks of four rows each. The first and third row of each block hold the population and quality respectively of the class-components for that point in the time step calculations.	HEADING, SETARR, PDOT, QDOT, WEIBUL, ABNRML, OUTPT, RAWDAT
IFILE (100)	IFILE is the integer control value array. Almost all the constants and	Used in almost all subroutines.

Name	Description	Where Used
	control values used are stored in IFILE. Detailed descriptions of the array can be found in the description of the IFILE file used to fill the array (Section 2.1.3).	
RFILE (100)	RFILE is the real value counterpart to IFILE. Many of the values contained correspond to either time or path functions.	SWEEP, ABNRML, FUNCTS
IFLAG (10)	IFLAG is a diagnostic indicator and control array used in development of the program. IFLAG controls calls to the TRACE subroutine.	BRANCH, SUMS, FUNCTS
IERROR (10)	The array keeps track of the calls to the ERRORS subroutine.	ERRORS
ACCUM (750)	ACCUM sums up system costs by cost coded node sweep.	NORMAL, ACOSTS, OUTPT
REVISN, SCHMDG, DATAFL, PLACE, RUNN, DATES	These arrays hold alphanumeric strings describing various aspects of a run.	DATAIN, RAWDAT, OUTPT, HEADNG.

3.1

a. Processor

The program was developed to run in double precision mode utilizing 64 bits per number. The CPU used was a Prime 350 with 16 bit, 2 byte words. The internal memory required for running is approximately 132 K bytes exclusive of that used for storing the path flow and cost data. The data structures, whether large arrays or direct access files, require close to an additional three megabytes for the maximum size model.

b. Output and Input Files

The input files require about 15 K bytes of memory. The data may be stored in sequential files on disk or magnetic tape without alteration. Card storage require the files to be integrated according to the program reading order.

The output file, for a base run, would require very little disk memory with memory used increasing proportionally when time simulations are run.

c. Data transmission devices

None. The program and data files are non-interactive and are self-contained.

3.2 Support Software

3.2.1 Operating System. The program is run in batch mode and should be compatible with any system equipped to run FORTRAN IV programs.

3.2.2 Compiler/Assembler. Requires a compiler for standard FORTRAN IV.

3.2.3 Other Software. If FORTRAN functions such as DABS( ) are not inserted by the compiler, then a FORTRAN IV math library will also be required.

3.3 Data Base

See Operations Manual, Section 2.3 and Program Maintenance, Section 2.1.2 and Section 4.5.

4. Maintenance Procedures

4.1 Programming Conventions

The program consists of a main program, CONTRL, and 35 subroutines. Subroutines follow the main program in the listing.

Variable and constants follow the IMPLICIT statement and the standard FORTRAN naming conventions. Mnemonic attributes may be found in most names but this should not be assumed to always be true as some names were generated at random.

Strings are kept in double precision arrays, and are used to print out headings for the output. String names are mnemonic in nature, but are not distinguished as strings. The remaining arrays, most of which serve to hold large amounts of data, are similarly mnemonic in nature, i.e., if "node" is part of the name, the data stored are node oriented.

4.2 Verification Procedures

Tested and verified data will be provided with the program along with the corresponding output. After installation and whenever the program becomes suspect, the data can be run to verify integrity of the program.

4.3 Error Correction Procedures

Almost all errors will be data related. Be sure that data values read through free formats are separated by blanks or commas. Carefully check the control file to insure against improper indicator and control parameters. If data are correct, confirm that the data files have been opened on the proper reading unit. If the data are all in order, check the following features of the program:

- Are the arrays too small for the model?
- Are there the correct number of repititions in the loops?
- Are reals and integers being mixed?
- Are all the subroutines present?
- Are all the subroutine arguments present?
- Is the program compiled correctly?

4.4 Special Maintenance Procedures

Data modifications are necessary for different types of runs. Some modifications are specified in the Operations Manual Section 3.3, 3.4 and 3.5.

Program modifications will revolve around four main points, string handling, data functions, I/O units and path flow/cost data structures. Each of these points is machine dependent and must be accounted for before using the program.

Alpha fields are used in DATAIN and RAWDAT to input strings and in OUTPT to write strings. These fields must be altered to conform to the characters per word of the user's machine. The program revision number is assigned to an array in the main program with enclosure by apostrophes. This must also be brought into line.

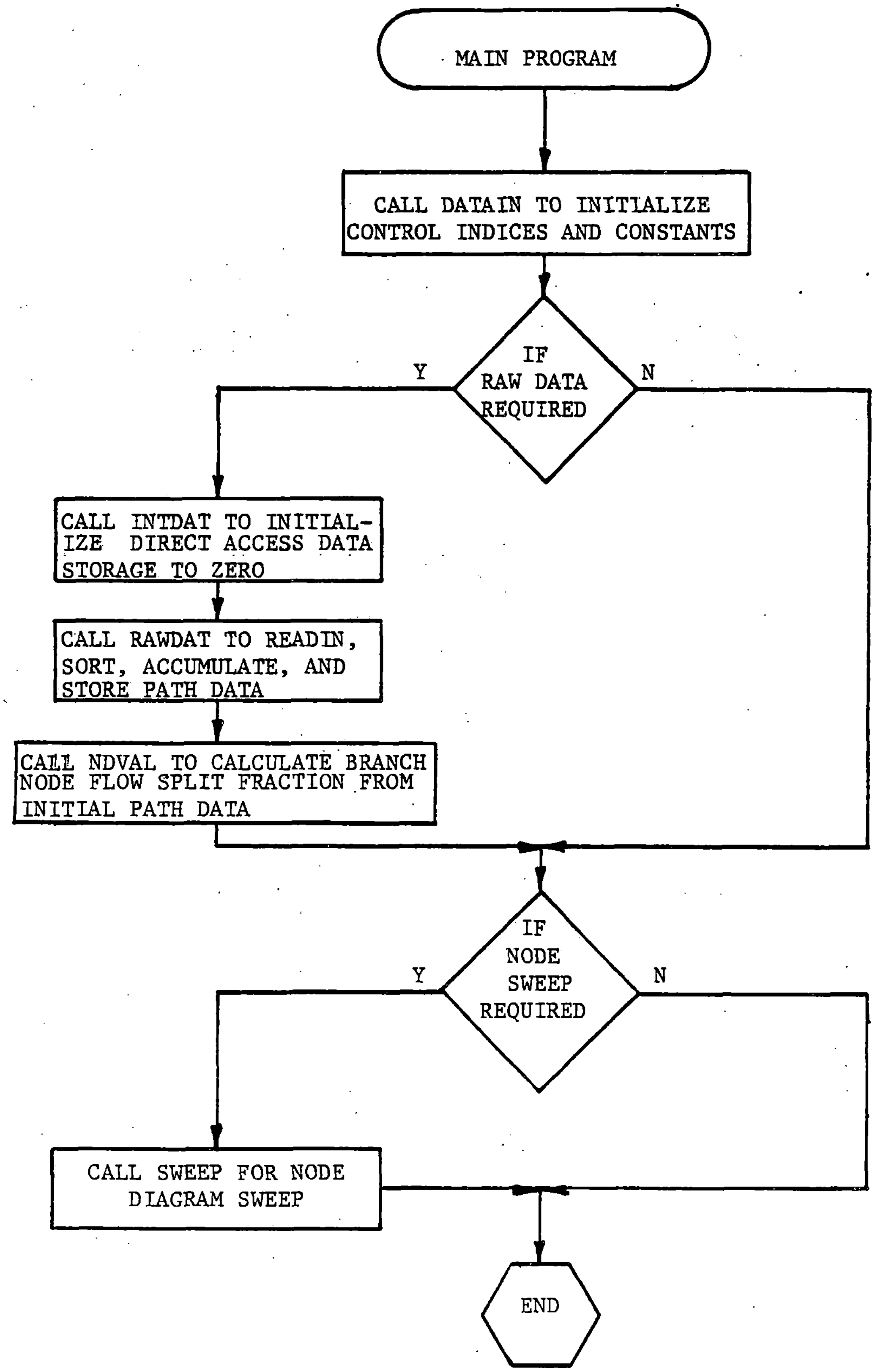
The date function is found in the OUTPT subroutine. In PRIME versions, this will be found as DATE$A. In DEC10 versions, it will be DATE. Variations for other machines will probably be similar. If the users machine does not have a date function, it would be a simple matter to input a date by modifying the DATAIN subroutine to read another alpha line from the IFILE file.

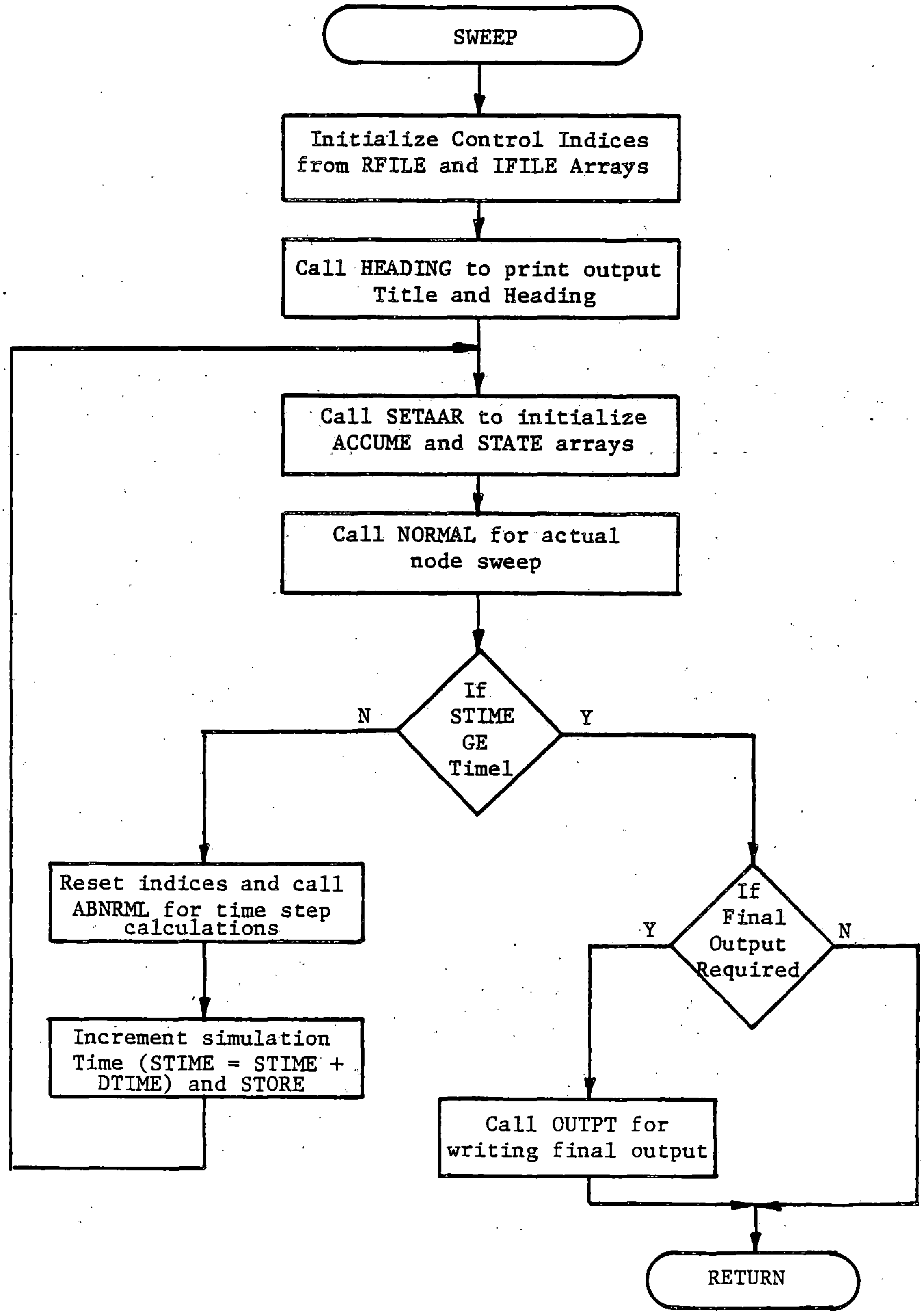
As mentioned previously, Output units are set through the IFILE file, as is the READ unit for the base path data. All other inputs are through the subroutine DATAIN with constants for unit numbers. If these constants aren't compatible with the user's machine, the user can simply change DATAIN READ statements to more suitable units. More extensive modification are necessary when card input is required. In this case the user must consolidate all data files into sequential cards, and alter FORMAT and READ statements accordingly. (NOTE: DATAIN inputs precede RAWDAT input.)

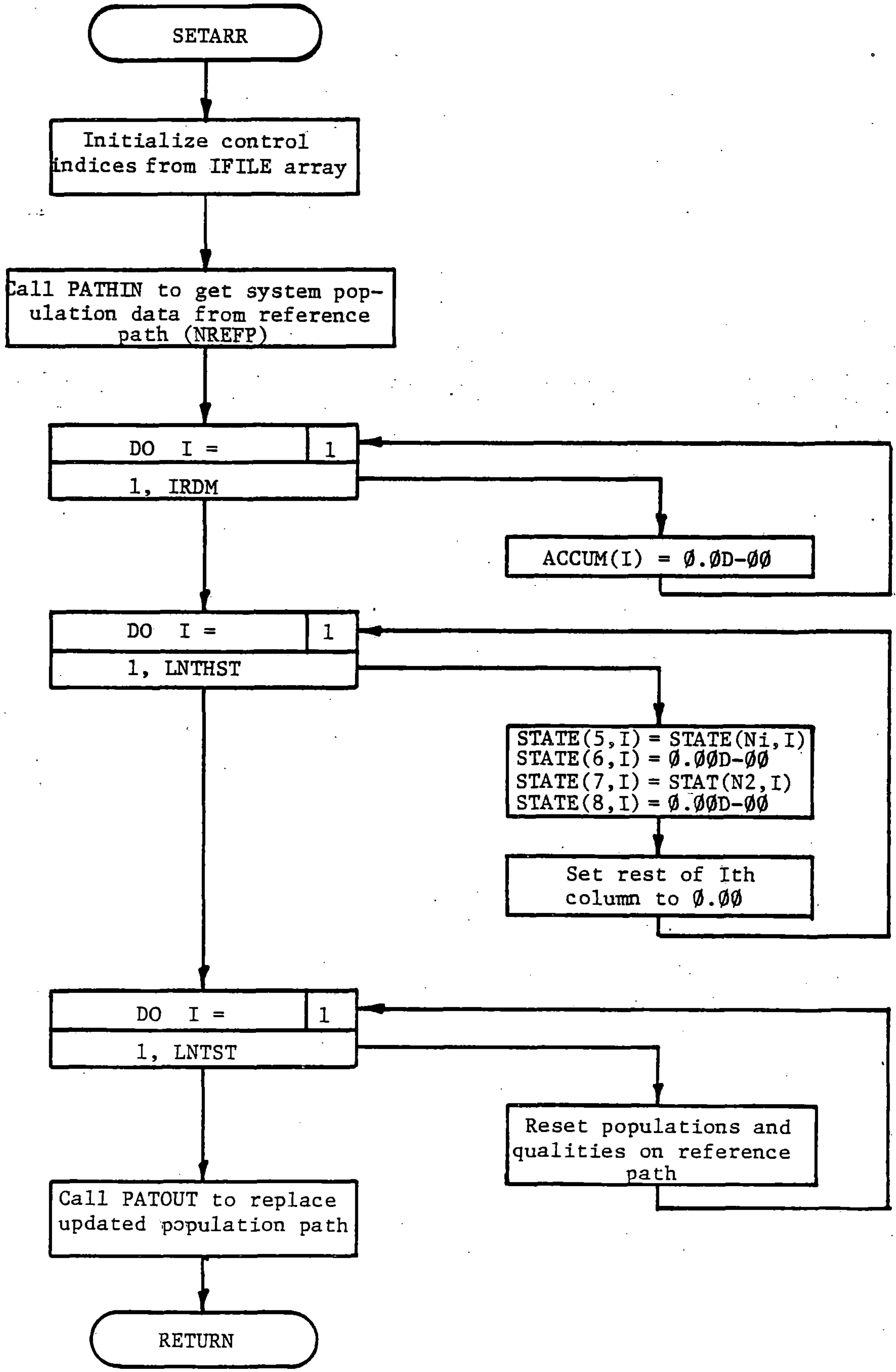
The path flow/cost data structures are dependent upon the machine's ability to handle extremely large arrays. Normally path data would be held in two large array IARAY and RARAY which are dimensioned 15 by 600 and 750 by 600 respectively to allow for the maximum of 600 paths, 30 class-components, and flow/cost blocks of 25 rows each. To handle this, some machines will require special compiling procedures. Some machines will be unable to handle it at all. In the latter instance, a direct access file setup may be able to handle the large amounts of data. As all handling of the data structure is through PATHIN and PATOUT, it is possible to replace these subroutines with two data manipulating subroutines in character with the user's computer. Retain names and arguments in the program but remove RARAY and IARAY from the COMMON block in the main program.

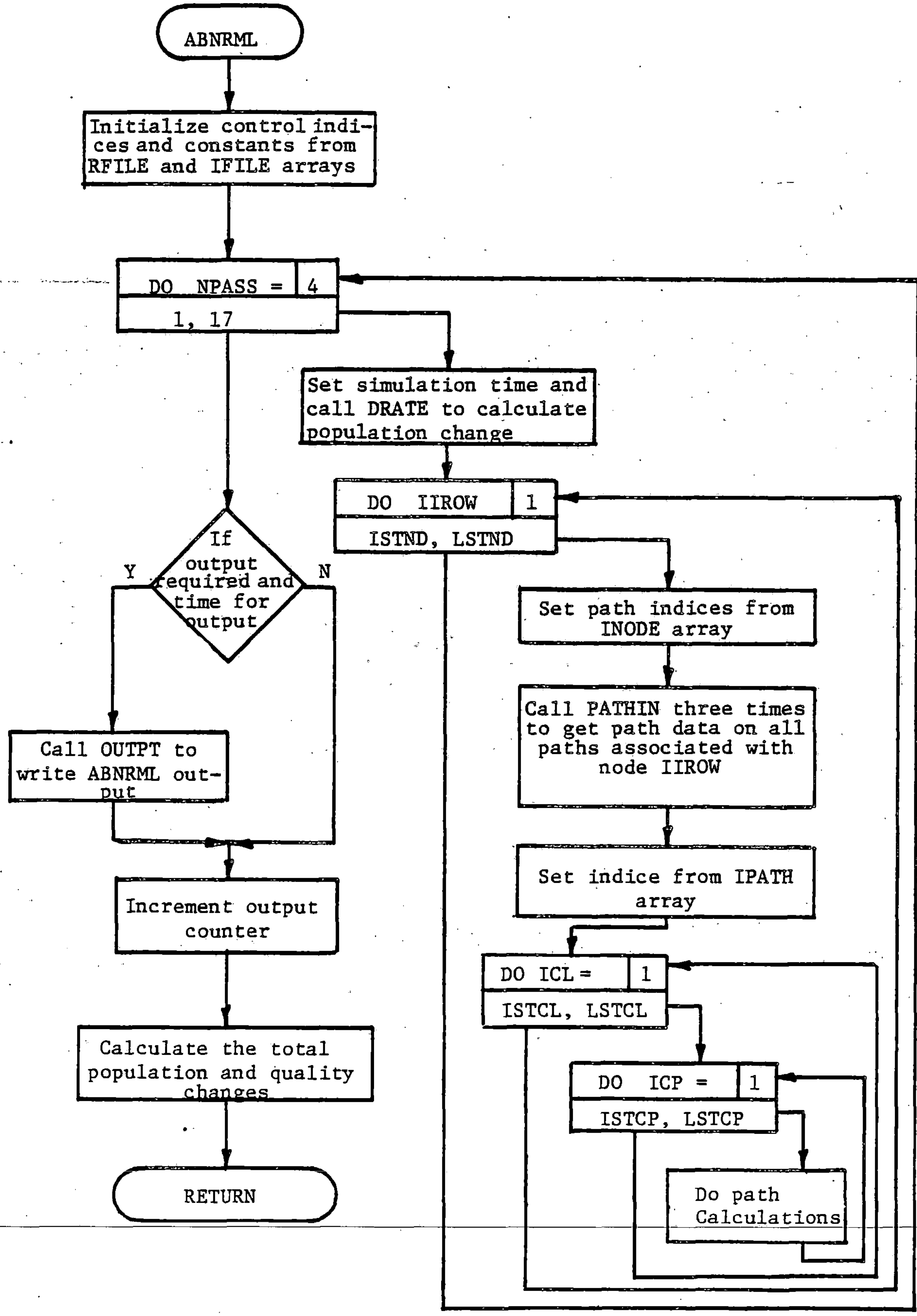
APPENDIX C

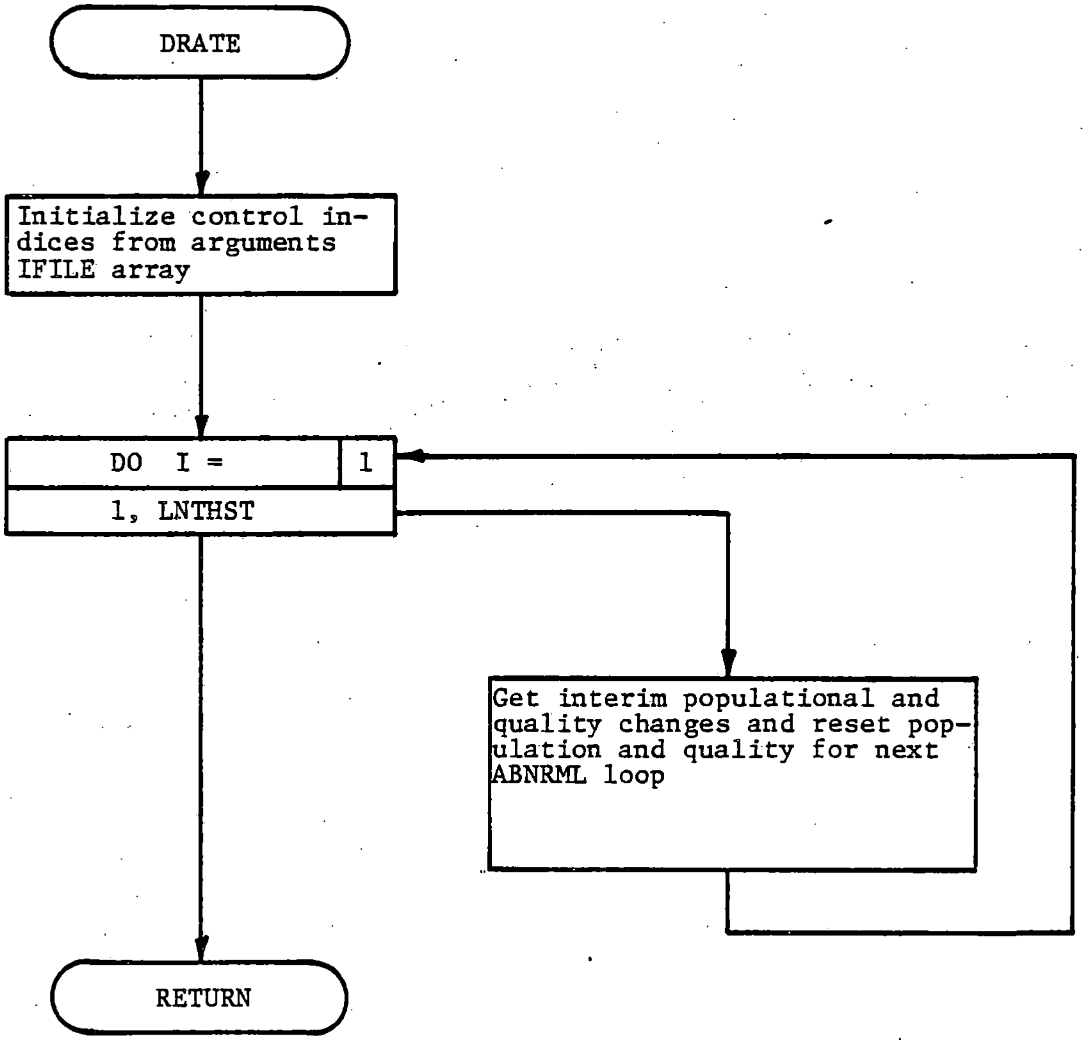
Computer Program Flow Charts

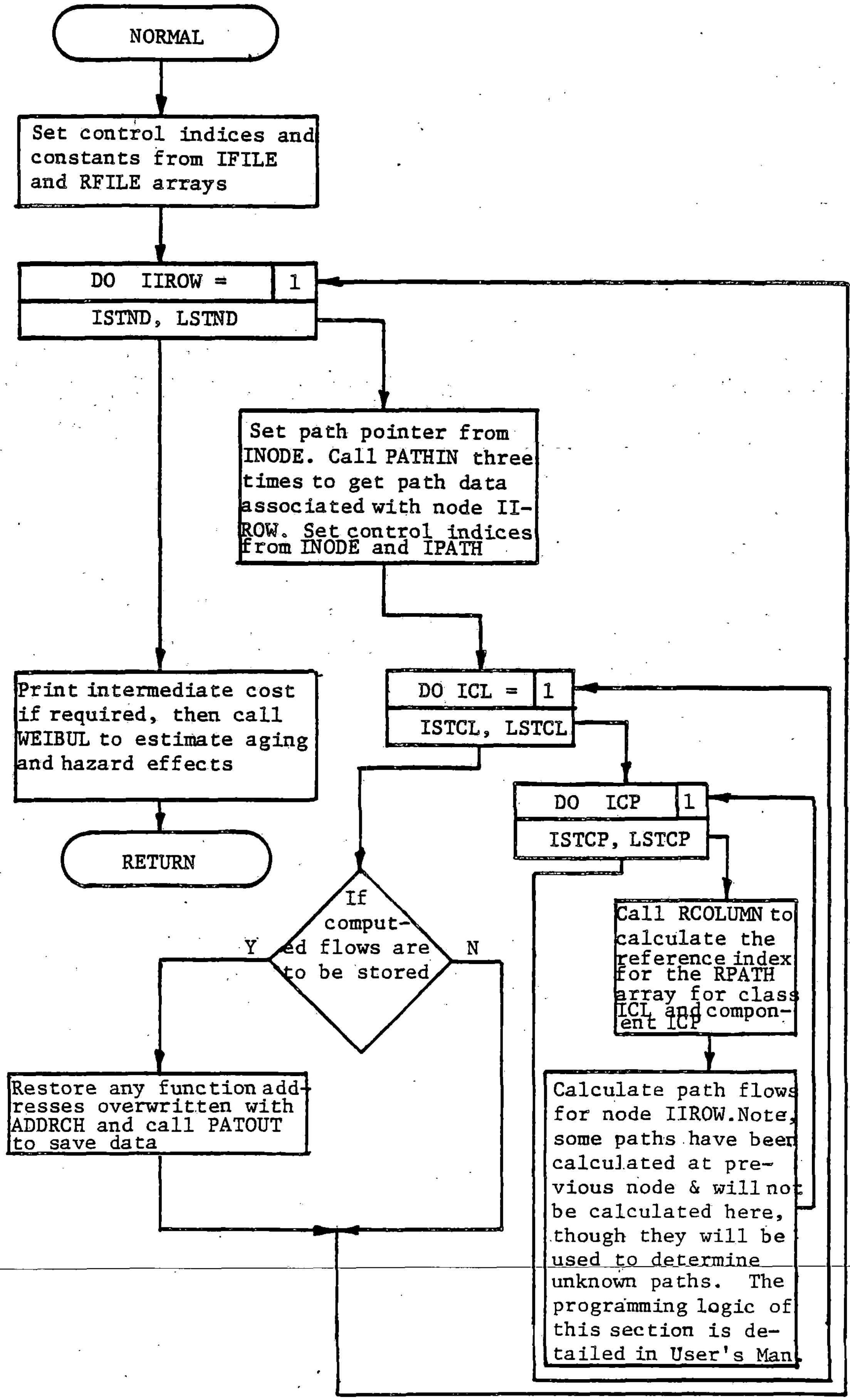

MAIN PROGRAM
CALL DATAIN TO INITIALIZE CONTROL INDICES AND CONSTANTS
IF RAW DATA REQUIRED
Y
N
CALL INTDAT TO INITIALIZE DIRECT ACCESS DATA STORAGE TO ZERO
CALL RAWDAT TO READIN, SORT, ACCUMULATE, AND STORE PATH DATA
CALL NDVAL TO CALCULATE BRANCH NODE FLOW SPLIT FRACTION FROM INITIAL PATH DATA
IF NODE SWEEP REQUIRED
Y
N
CALL SWEEP FOR NODE DIAGRAM SWEEP
END

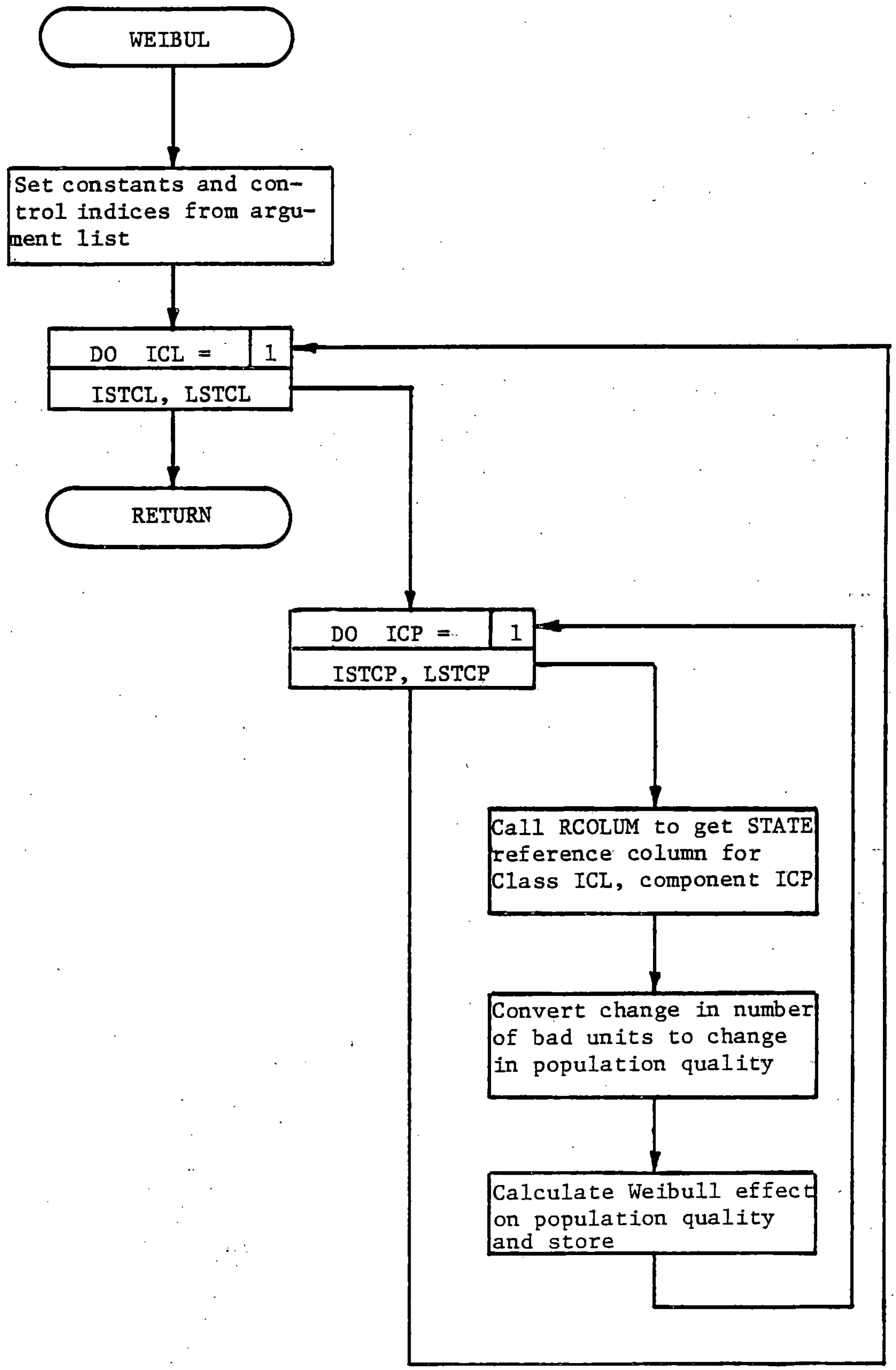

SWEEP
Initialize Control Indices from RFILE and IFILE Arrays
Call HEADING to print output Title and Heading
Call SETAAR to initialize ACCUME and STATE arrays
Call NORMAL for actual node sweep
If STIME GE Time1
N
Y
Reset indices and call ABNRML for time step calculations
Increment simulation Time (STIME = STIME + DTIME) and STORE
If Final Output Required
Y
N
Call OUTPT for writing final output
RETURN

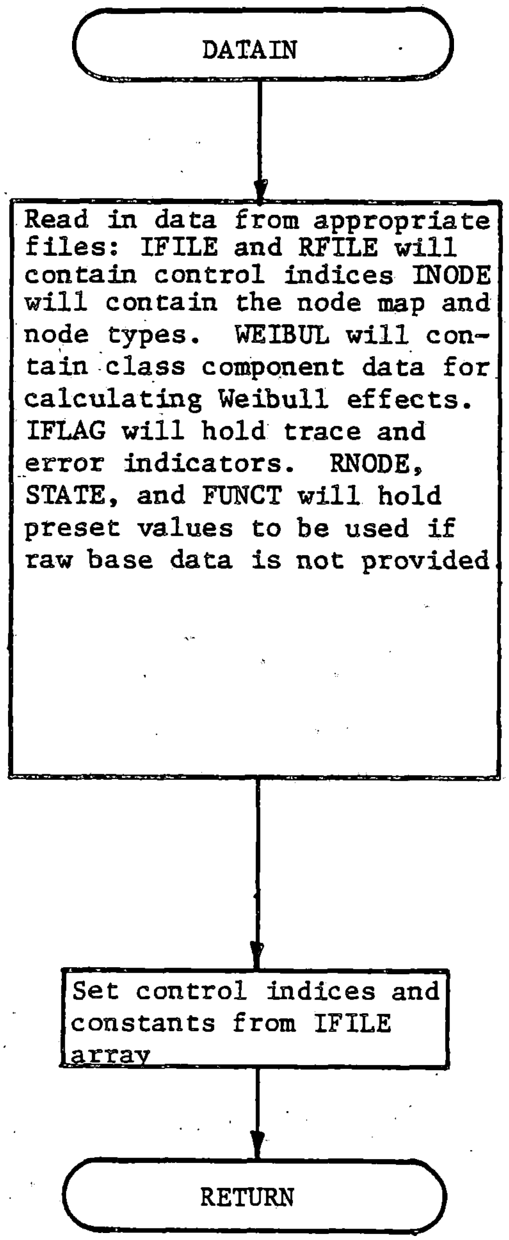

SETARR
Initialize control indices from IFILE array
Call PATHIN to get system population data from reference path (NREFP)
DO I = 1
1, IRDM
ACCUM(I) = Ø.ØD-ØØ
DO I = 1
1, LNTHST
STATE(5,I) = STATE(Ni,I)
STATE(6,I) = Ø.ØØD-ØØ
STATE(7,I) = STAT(N2,I)
STATE(8,I) = Ø.ØØD-ØØ
Set rest of Ith column to Ø.ØØ
DO I = 1
1, LNTST
Reset populations and qualities on reference path
Call PATOUT to replace updated population path
RETURN

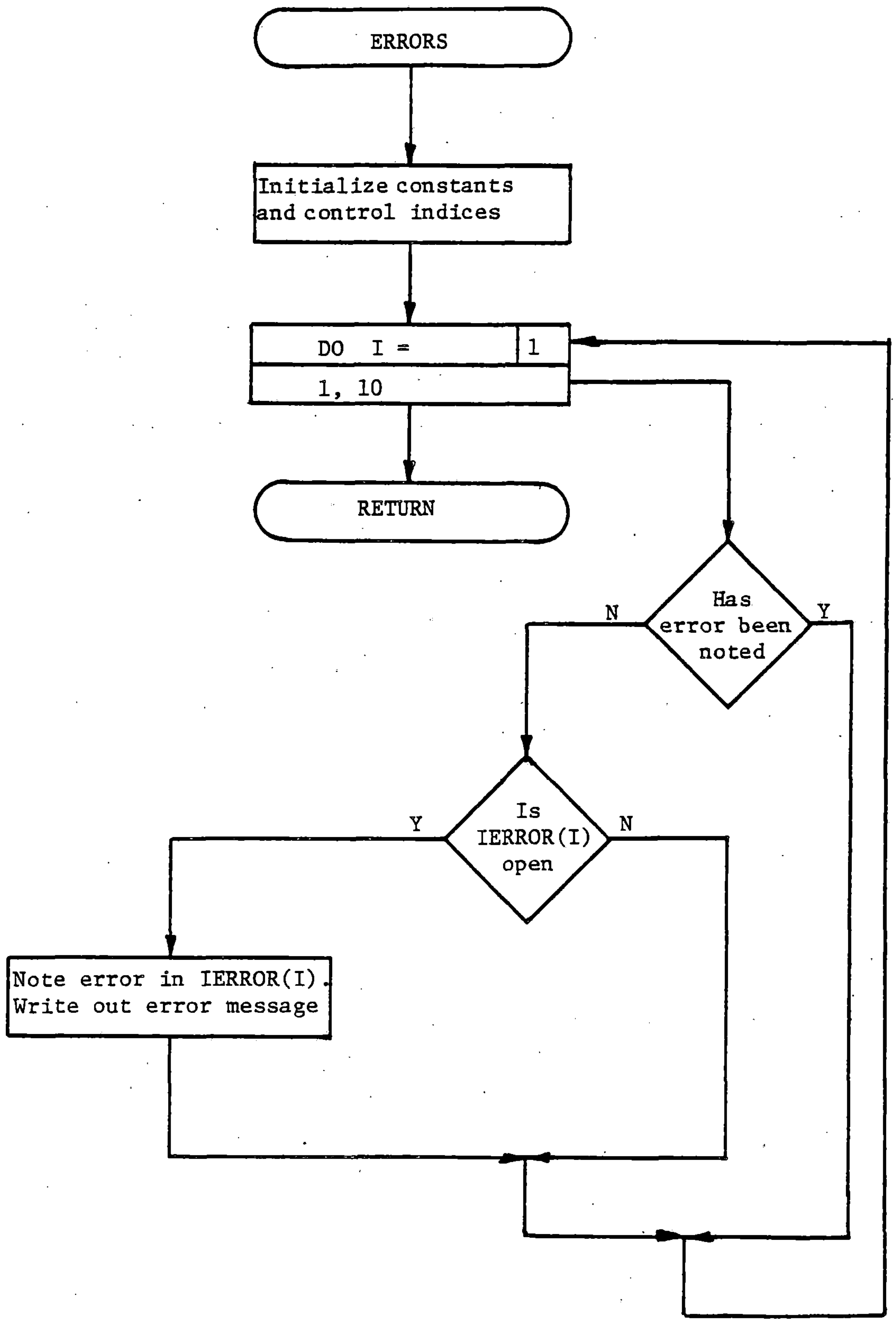

ABNRML
Initialize control indices and constants from RFILE and IFILE arrays
DO NPASS = 1, 17
4
Set simulation time and call DRATE to calculate population change
DO IIROW
1
ISTND, LSTND
If output required and time for output
Y
N
Call OUTPT to write ABNRML output
Set path indices from INODE array
Call PATHIN three times to get path data on all paths associated with node IIROW
Set indice from IPATH array
Increment output counter
DO ICL =
1
ISTCL, LSTCL
Calculate the total population and quality changes
DO ICP =
1
ISTCP, LSTCP
RETURN
Do path Calculations

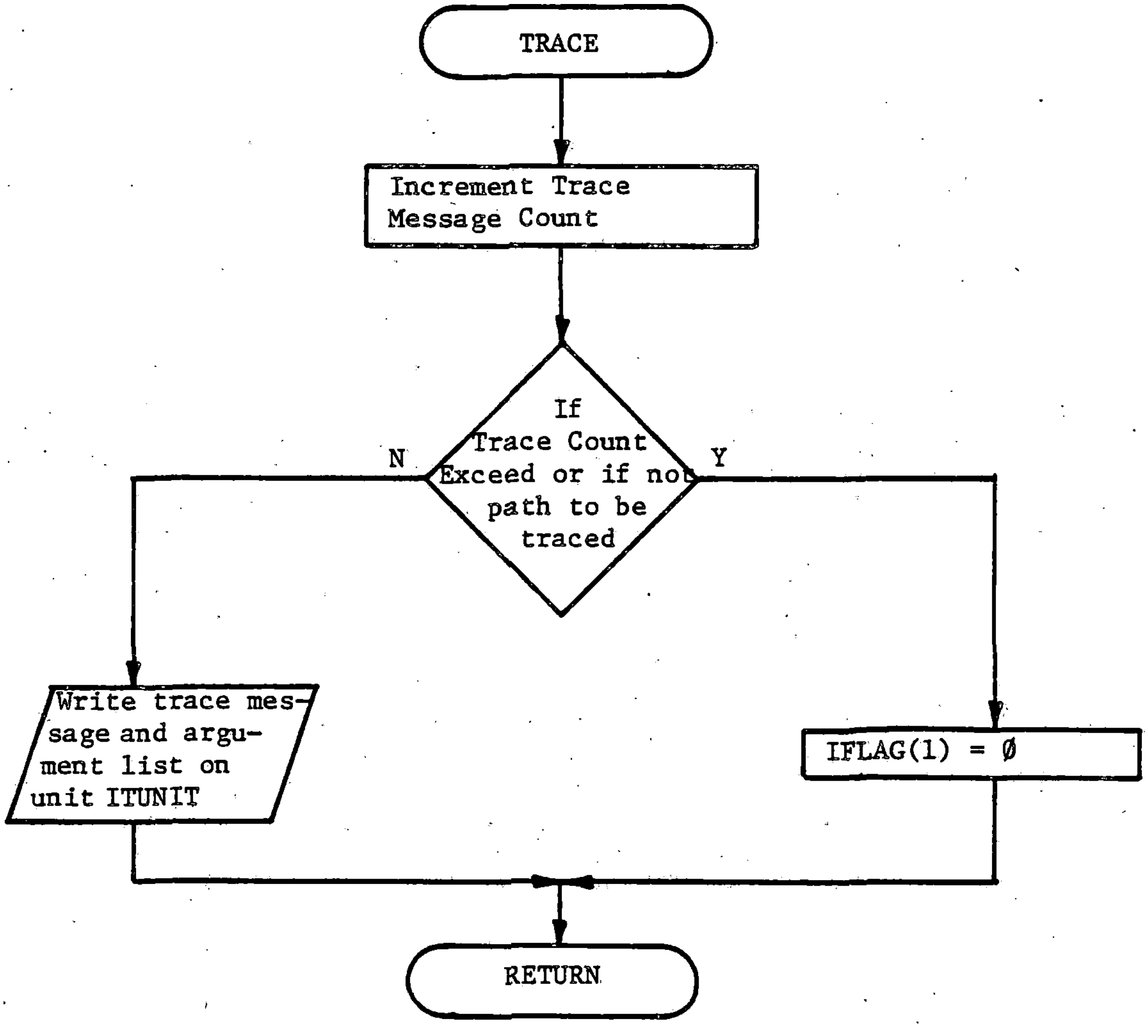

DRATE
Initialize control indices from arguments IFILE array
DO I =
1
1, LNTHST
Get interim populational and quality changes and reset population and quality for next ABNRML loop
RETURN

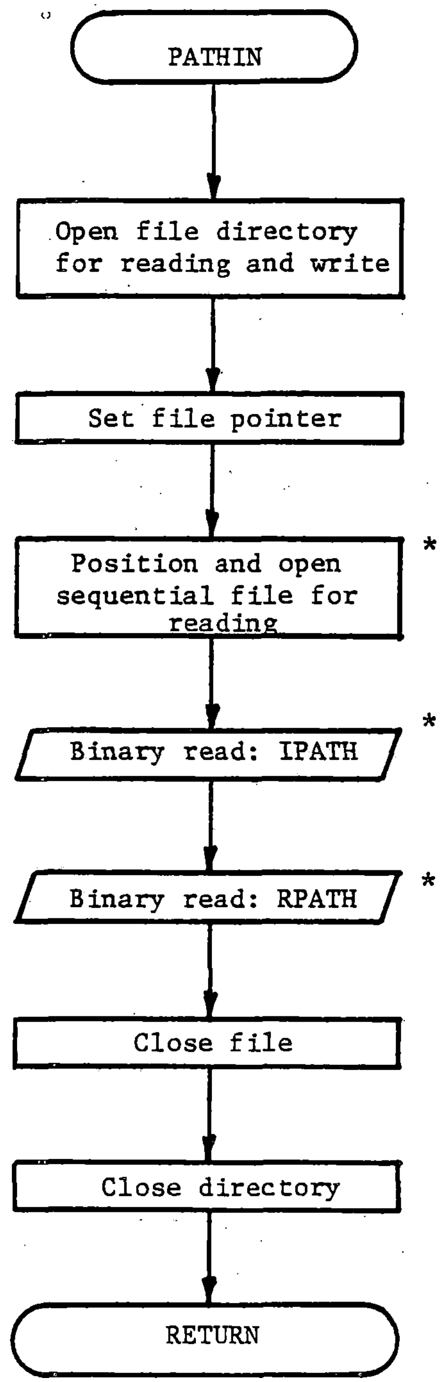

NORMAL
Set control indices and constants from IFILE and RFILE arrays
DO IIROW = 1
ISTND, LSTND
Set path pointer from INODE. Call PATHIN three times to get path data associated with node II-ROW. Set control indices from INODE and IPATH
Print intermediate cost if required, then call WEIBUL to estimate aging and hazard effects
RETURN
DO ICL = 1
ISTCL, LSTCL
DO ICP 1
ISTCP, LSTCP
If computed flows are to be stored
Y
N
Call RCOLUMN to calculate the reference index for the RPATH array for class ICL and component ICP
Restore any function addresses overwritten with ADDRCH and call PATOUT to save data
Calculate path flows for node IIROW. Note, some paths have been calculated at previous node & will not be calculated here, though they will be used to determine unknown paths. The programming logic of this section is detailed in User's Man.


WEIBUL
Set constants and control indices from argument list
DO ICL = 1
ISTCL, LSTCL
RETURN
DO ICP = 1
ISTCP, LSTCP
Call RCOLUM to get STATE reference column for Class ICL, component ICP
Convert change in number of bad units to change in population quality
Calculate Weibull effect on population quality and store

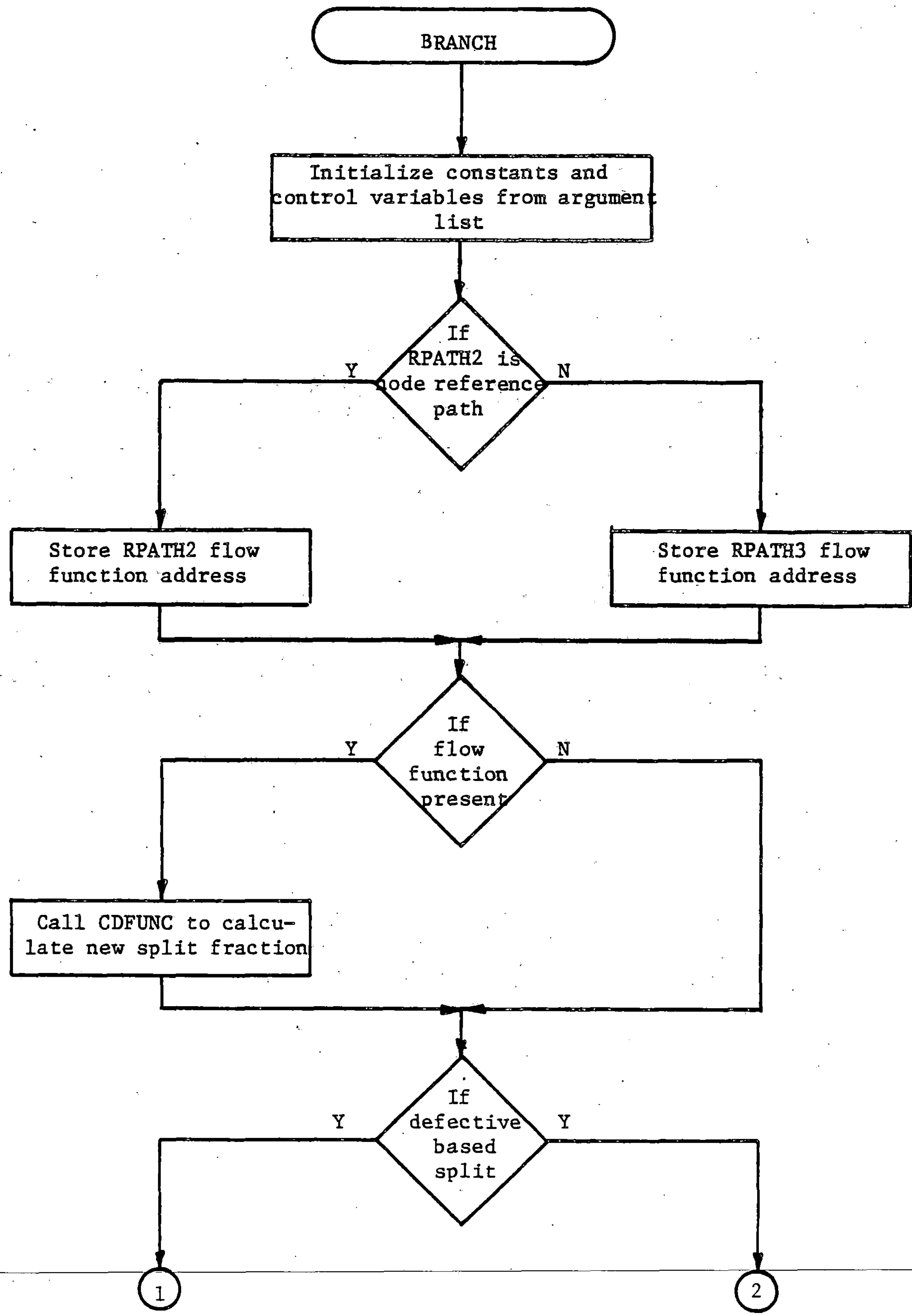

DATAIN
Read in data from appropriate files: IFILE and RFILE will contain control indices INODE will contain the node map and node types. WEIBUL will contain class component data for calculating Weibull effects. IFLAG will hold trace and error indicators. RNODE, STATE, and FUNCT will hold preset values to be used if raw base data is not provided
Set control indices and constants from IFILE array
RETURN

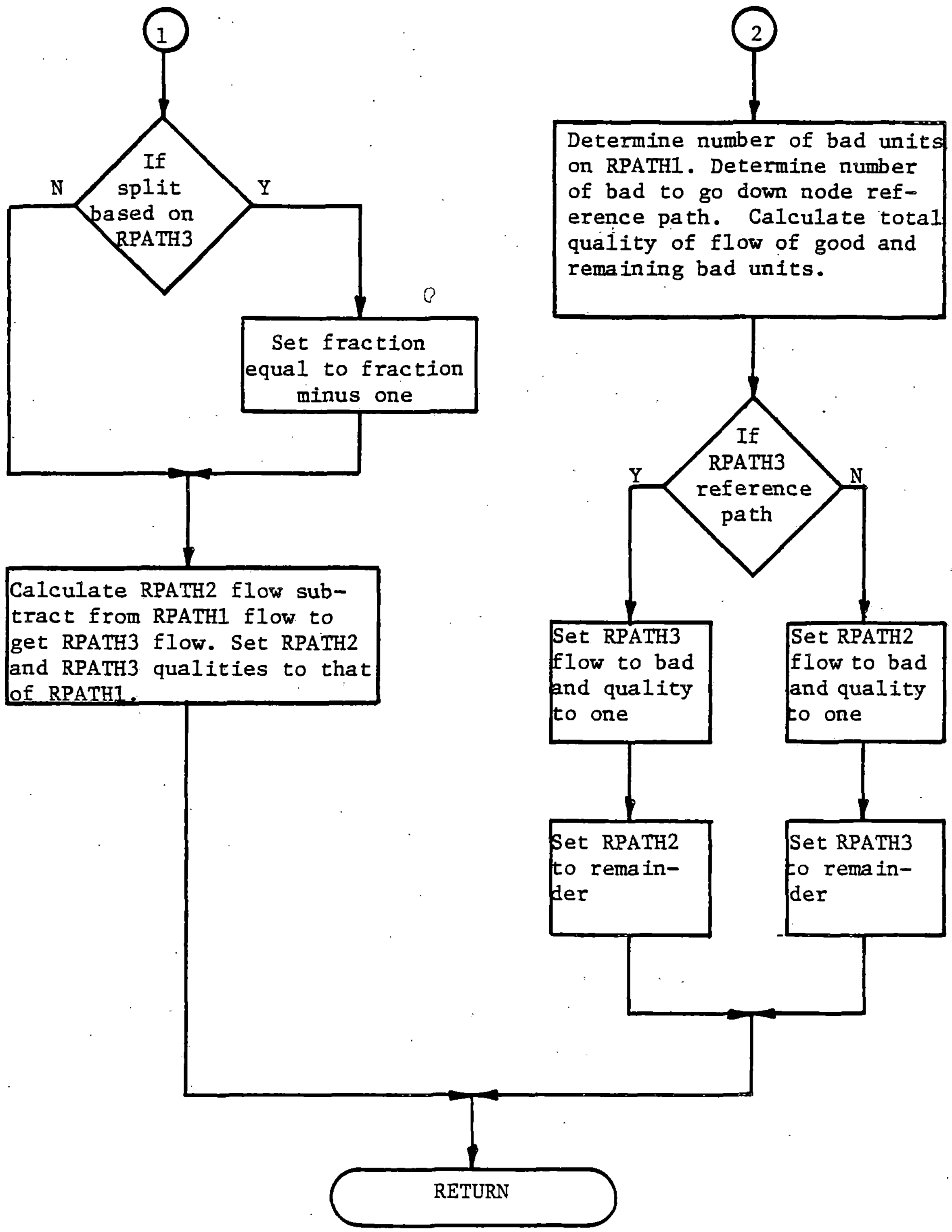

ERRORS
Initialize constants and control indices
DO I =
1
1, 10
RETURN
Has error been noted
N
Y
Is IERROR(I) open
Y
N
Note error in IERROR(I).
Write out error message

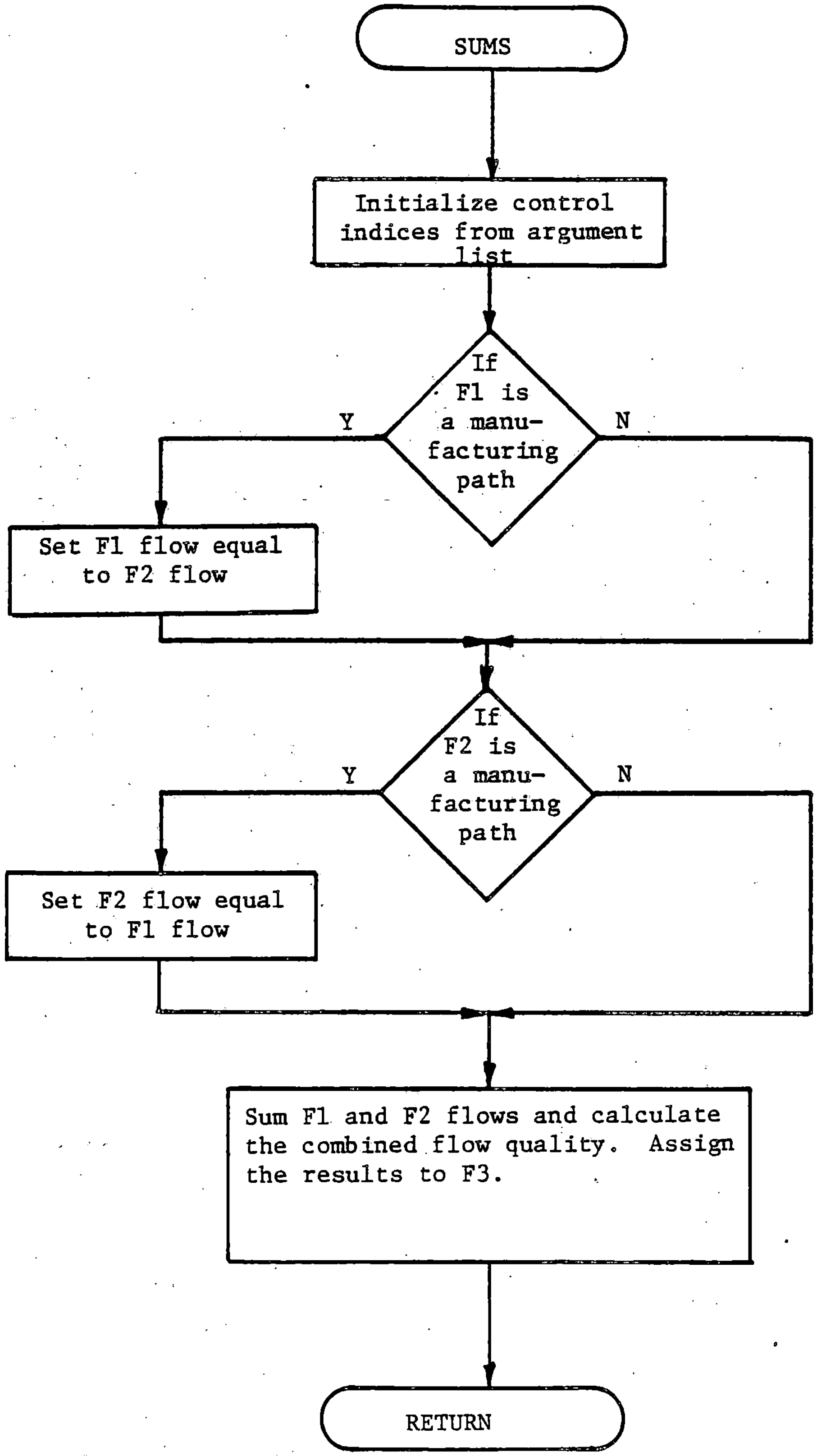

TRACE
Increment Trace Message Count
If Trace Count Exceed or if not path to be traced
N
Y
Write trace message and argument list on unit ITUNIT
IFLAG(1) = Ø
RETURN

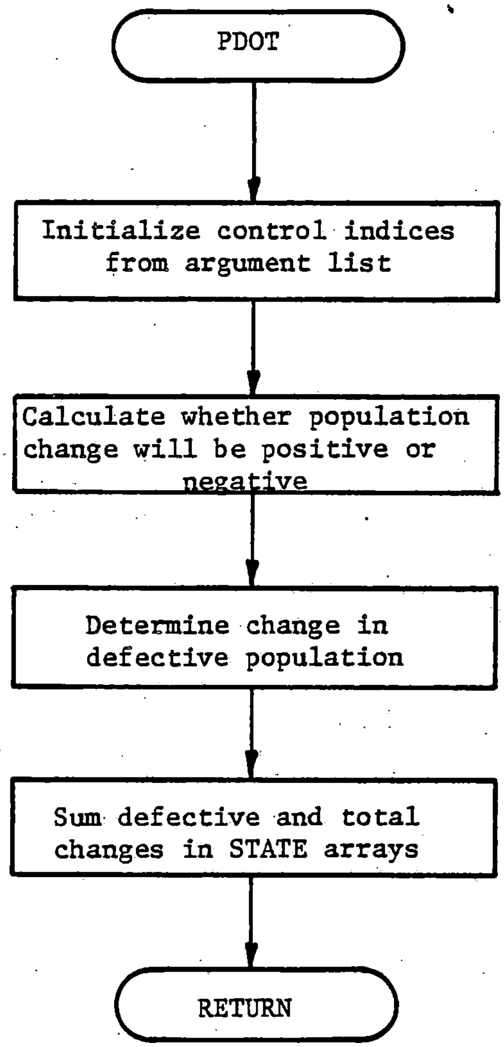


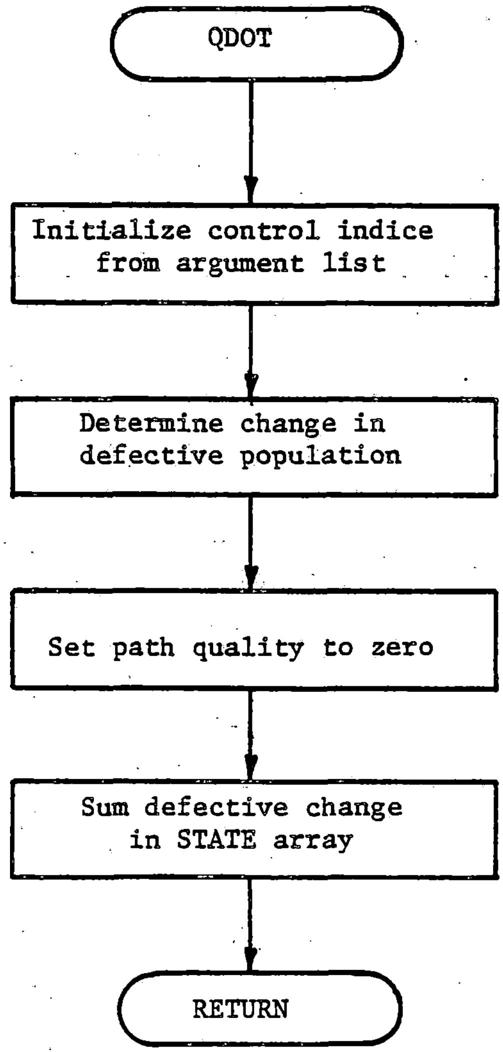
* PATOUT is exactly the same except that these instructions substitute a write for read.

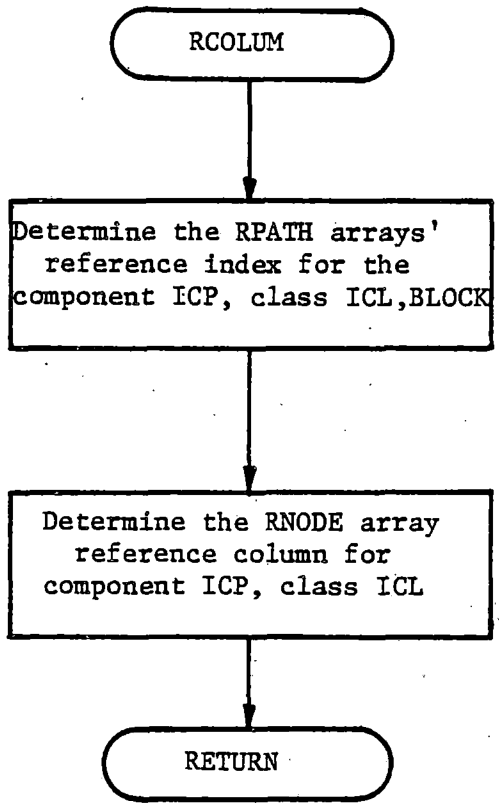

BRANCH
Initialize constants and control variables from argument list
If RPATH2 is node reference path
Y
N
Store RPATH2 flow function address
Store RPATH3 flow function address
If flow function present
Y
N
Call CDFUNC to calculate new split fraction
If defective based split
Y
Y
1
2

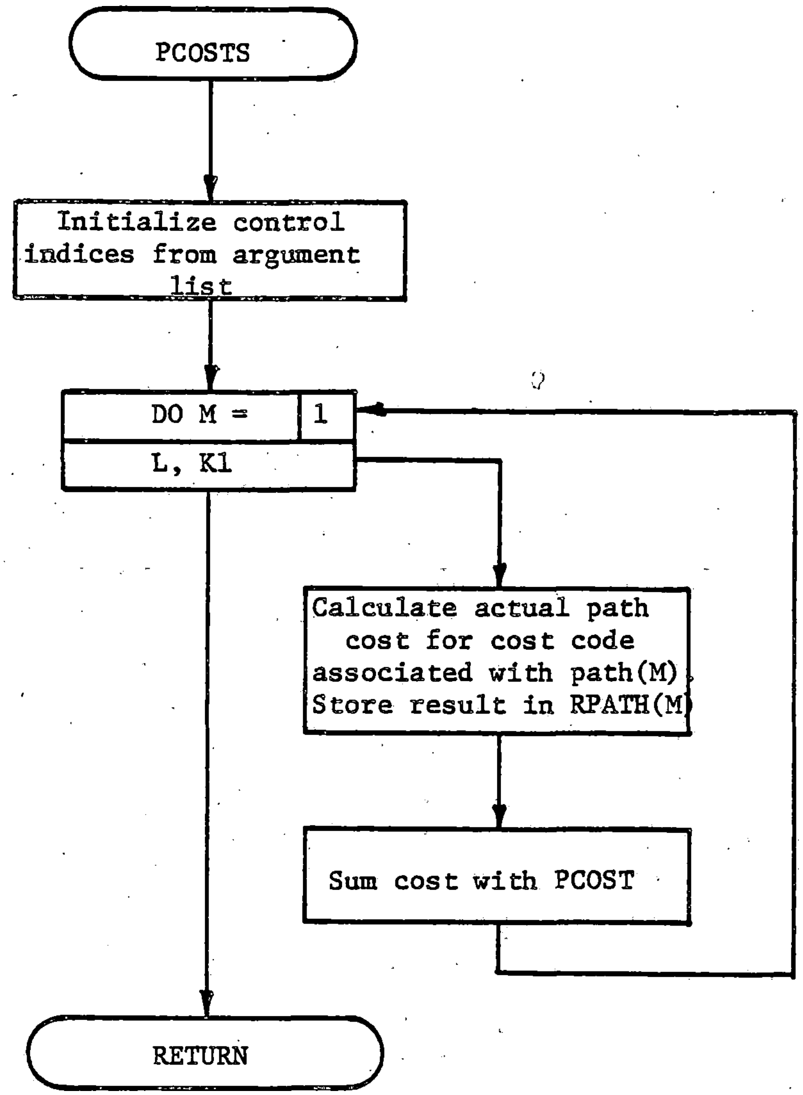

1
If split based on RPATH3
N
Y
Set fraction equal to fraction minus one
Calculate RPATH2 flow subtract from RPATH1 flow to get RPATH3 flow. Set RPATH2 and RPATH3 qualities to that of RPATH1.
2
Determine number of bad units on RPATH1. Determine number of bad to go down node reference path. Calculate total quality of flow of good and remaining bad units.
If RPATH3 reference path
Y
N
Set RPATH3 flow to bad and quality to one
Set RPATH2 flow to bad and quality to one
Set RPATH2 to remainder
Set RPATH3 to remainder
RETURN

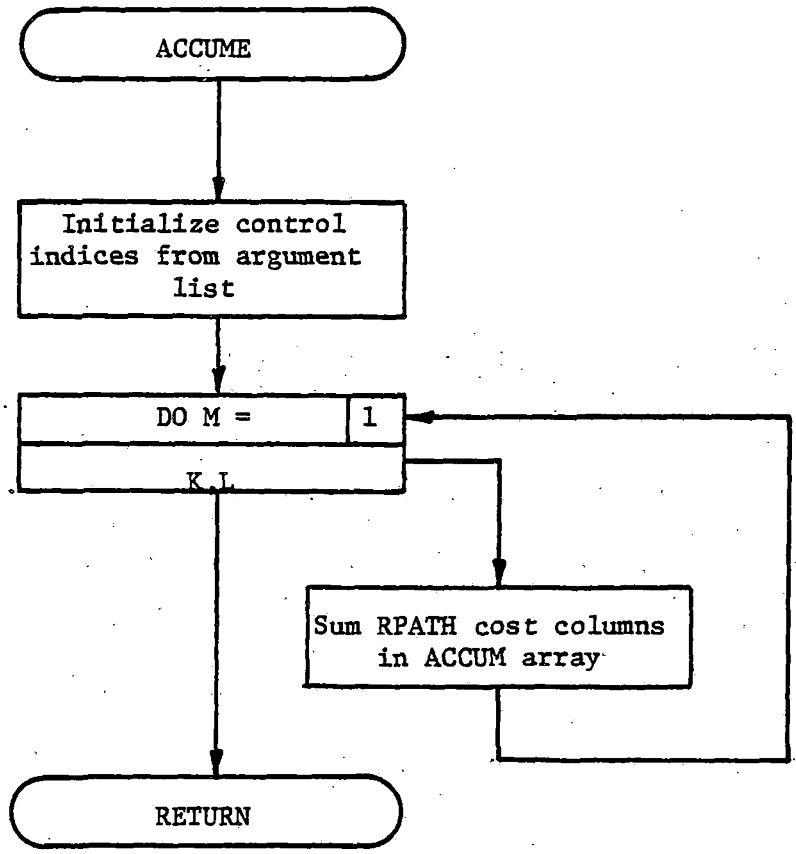

SUMS
Initialize control indices from argument list
If F1 is a manu- facturing path
Y
N
Set F1 flow equal to F2 flow
If F2 is a manu- facturing path
Y
N
Set F2 flow equal to F1 flow
Sum F1 and F2 flows and calculate the combined flow quality. Assign the results to F3.
RETURN

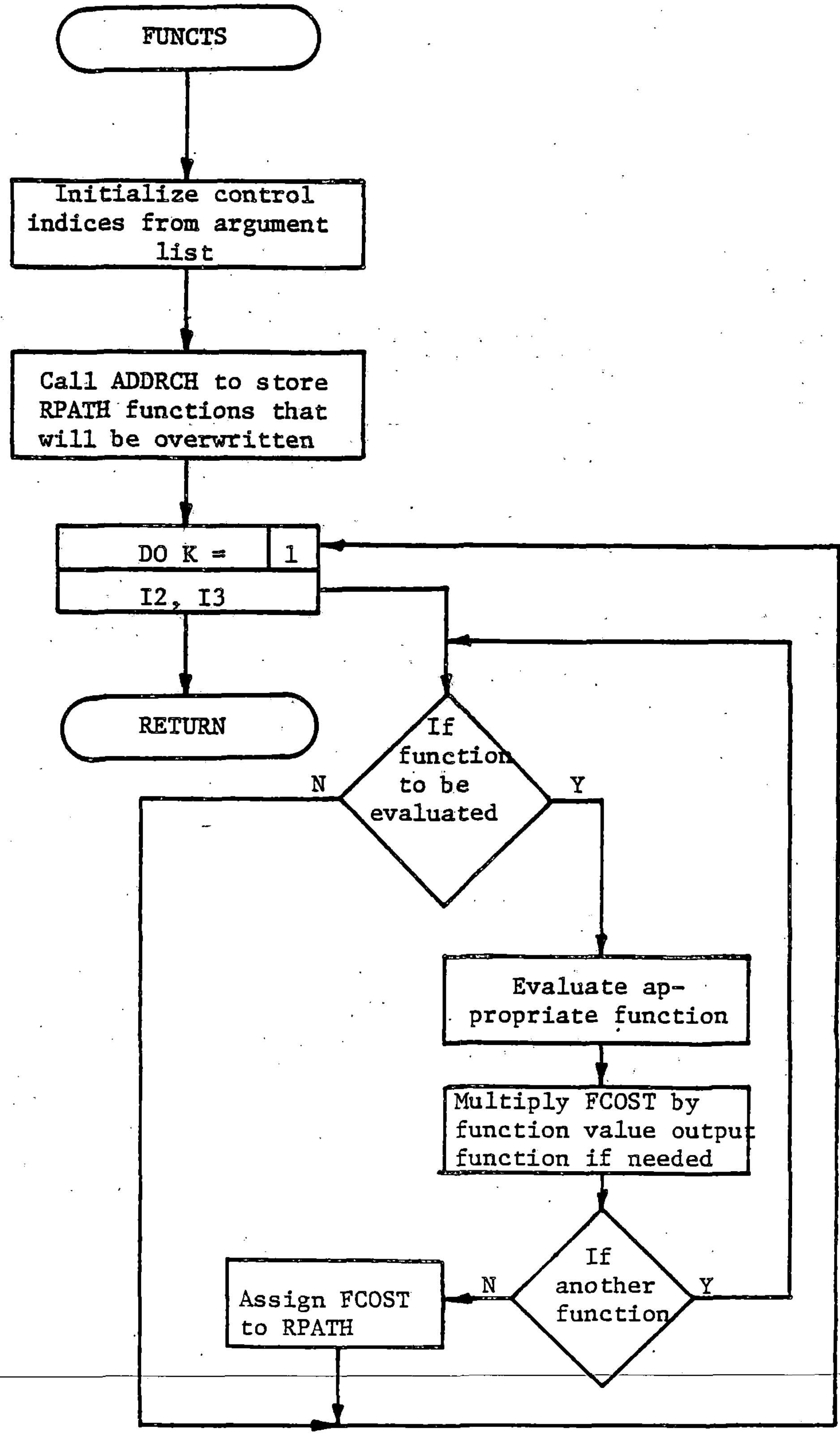

PDOT
Initialize control indices from argument list
Calculate whether population change will be positive or negative
Determine change in defective population
Sum defective and total changes in STATE arrays
RETURN

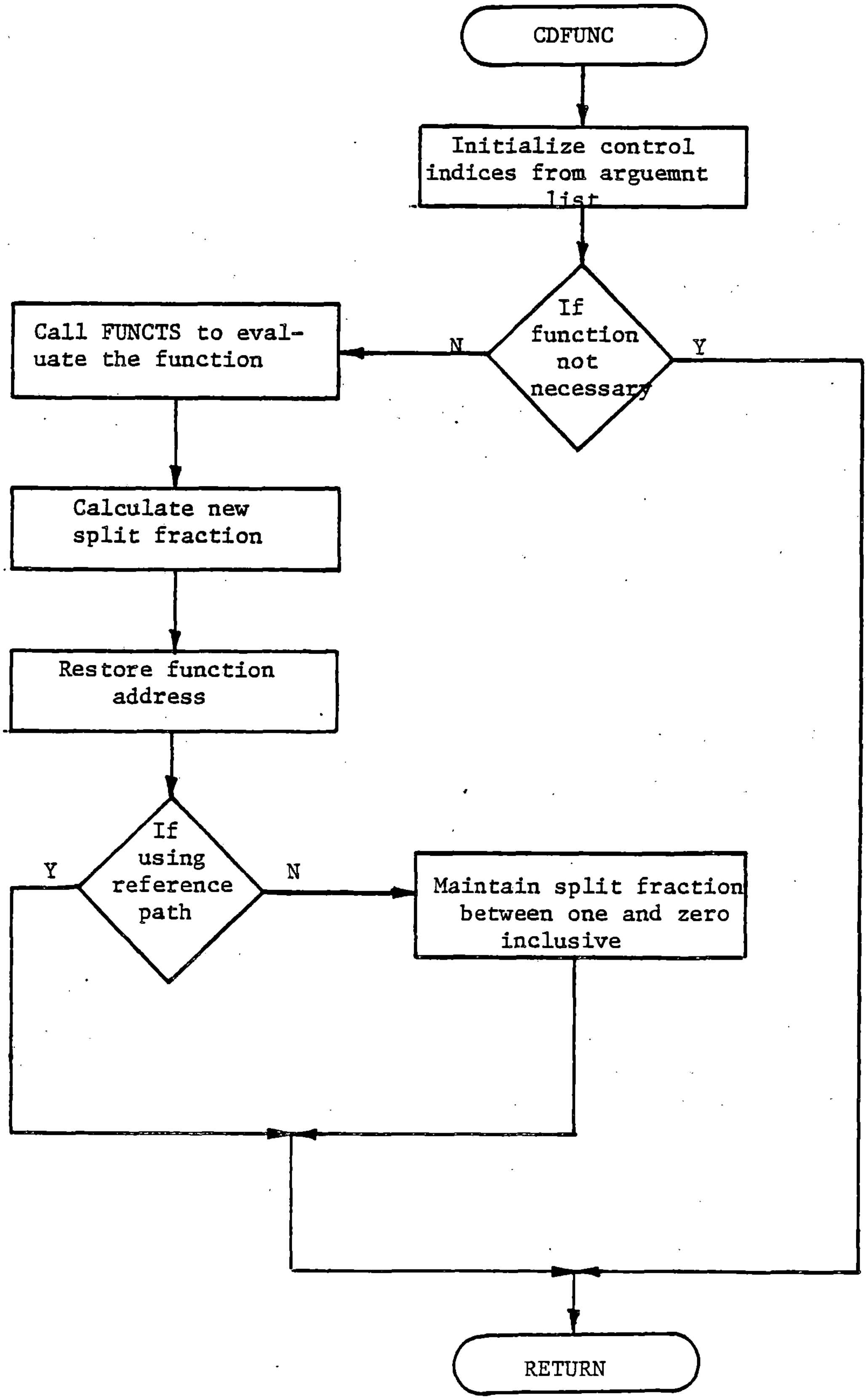

QDOT
Initialize control indice from argument list
Determine change in defective population
Set path quality to zero
Sum defective change in STATE array
RETURN

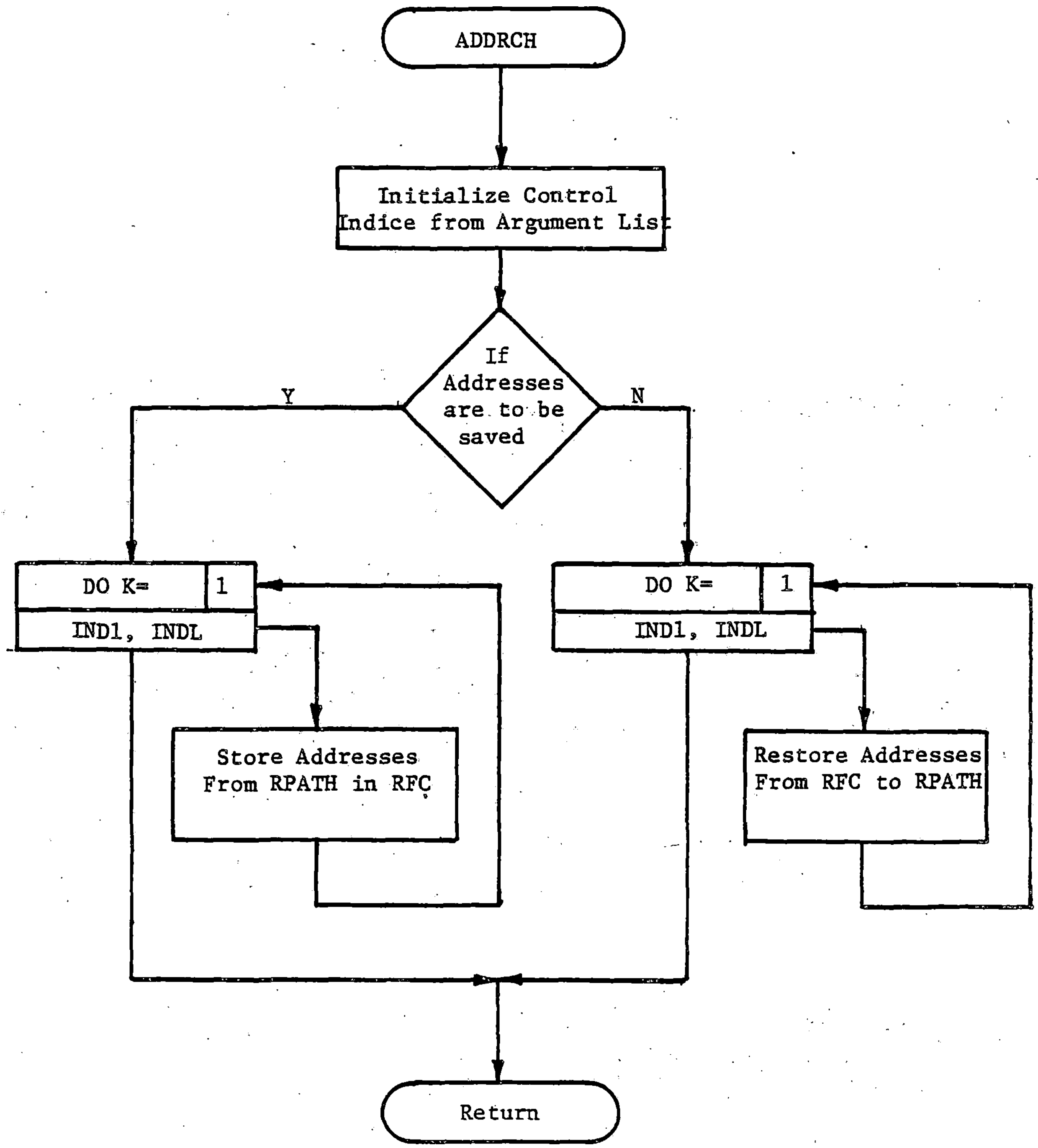

RCOLUM
Determine the RPATH arrays' reference index for the component ICP, class ICL,BLOCK
Determine the RNODE array reference column for component ICP, class ICL
RETURN

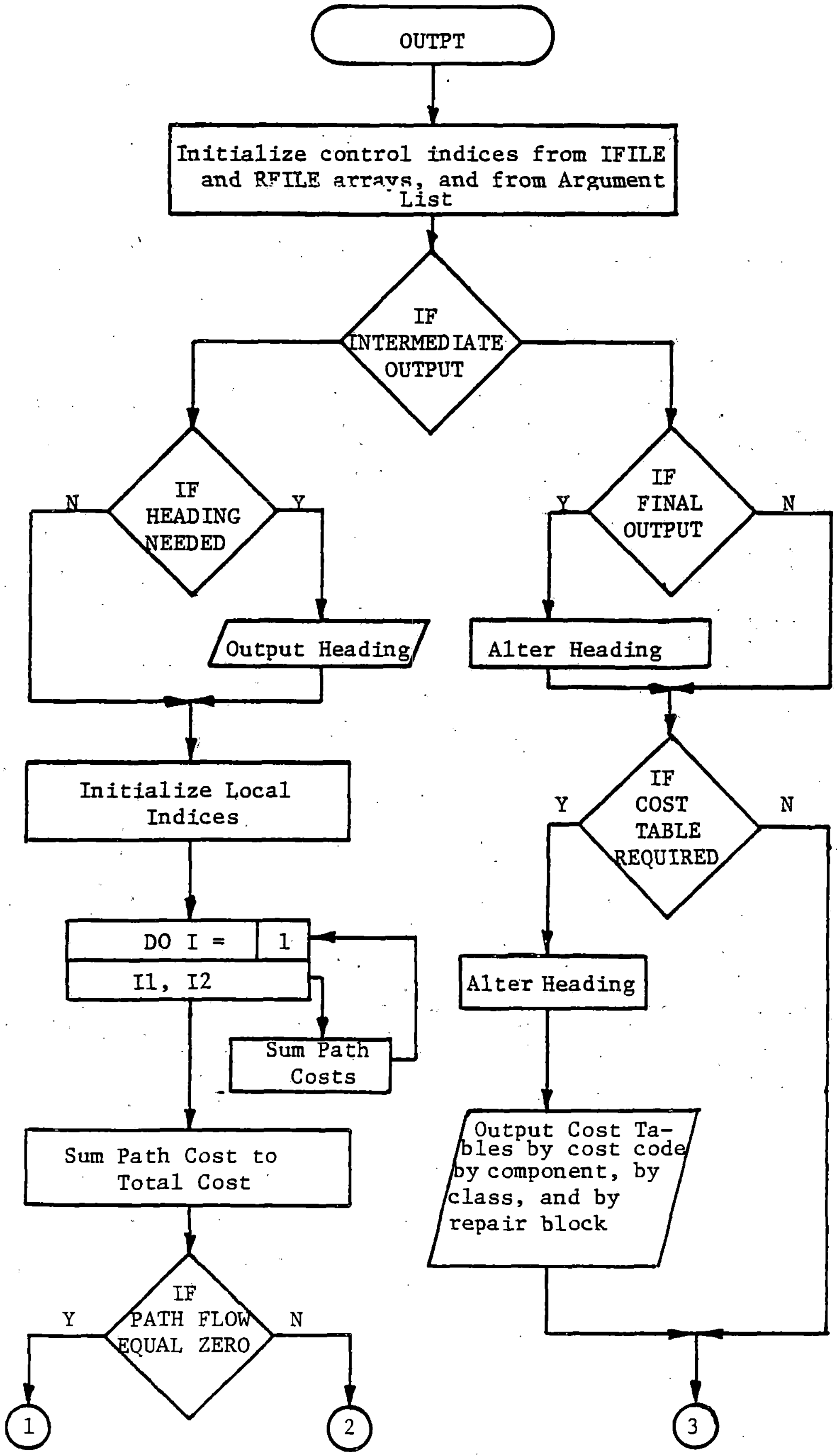

PCOSTS
Initialize control indices from argument list
DO M = 1
L, K1
Calculate actual path cost for cost code associated with path(M). Store result in RPATH(M)
Sum cost with PCOST
RETURN

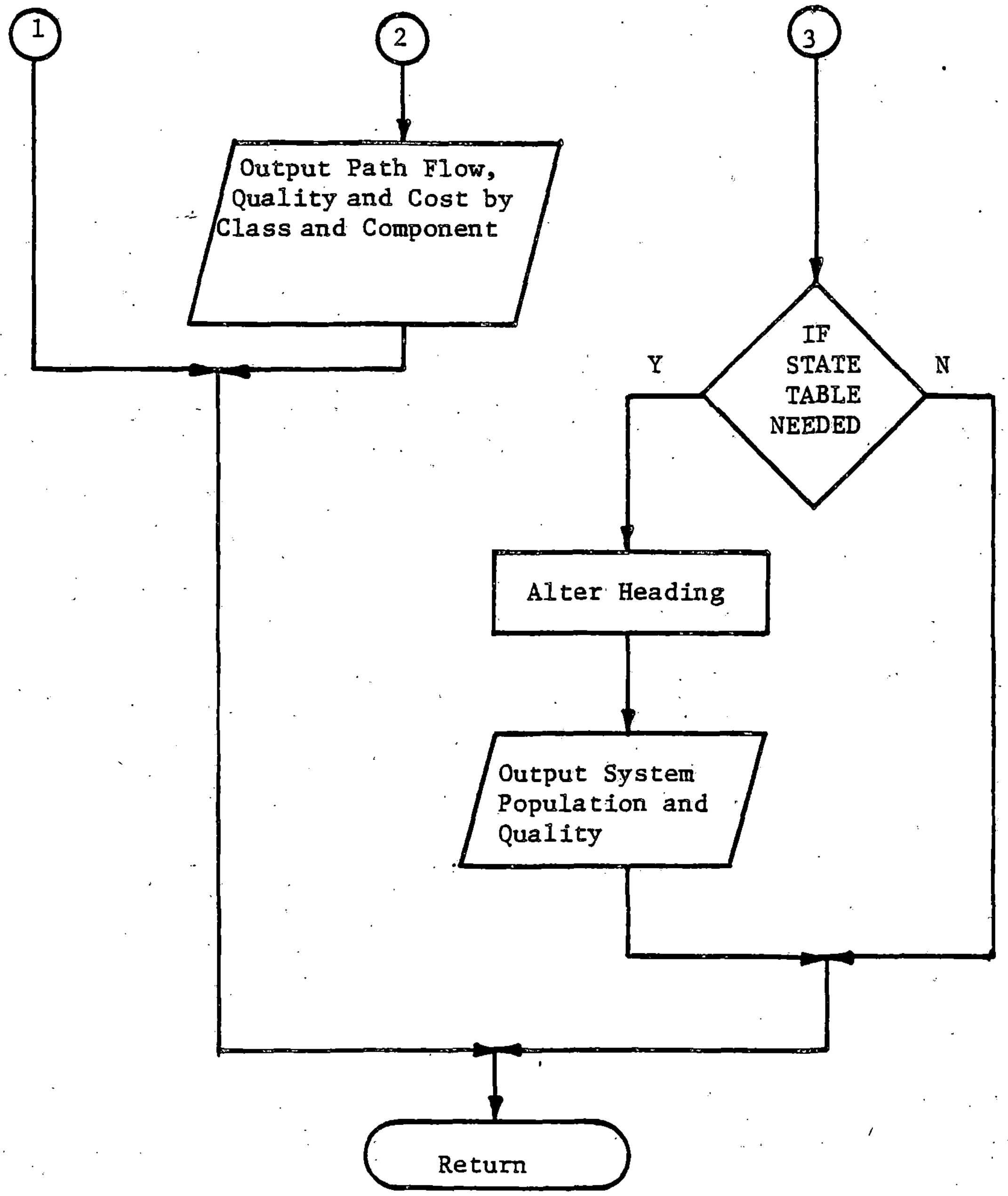

ACCUME
Initialize control indices from argument list
DO M =
1
K,L
Sum RPATH cost columns in ACCUM array
RETURN

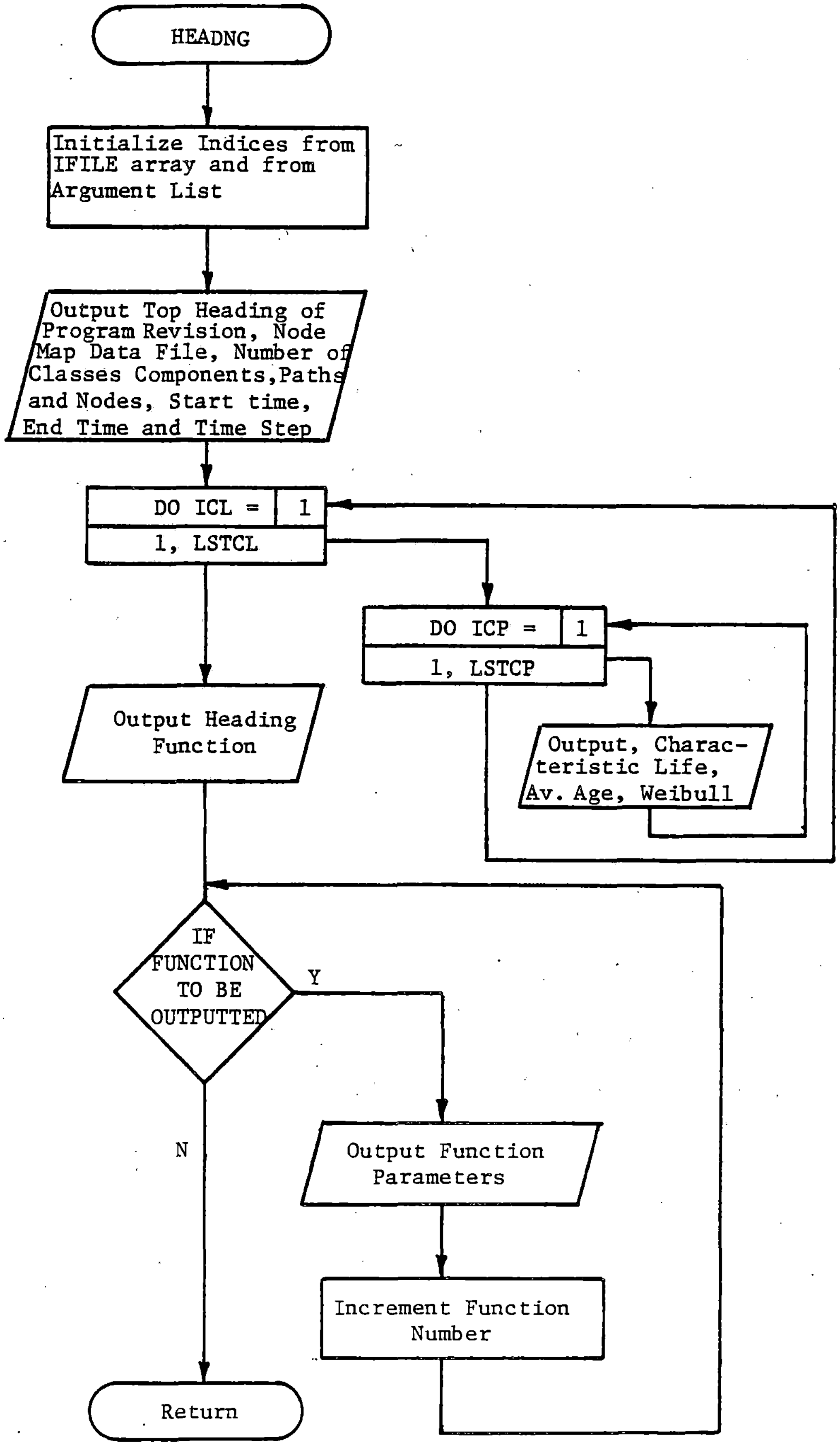

FUNCTS
Initialize control indices from argument list
Call ADDRCH to store RPATH functions that will be overwritten
DO K =
1
I2, I3
RETURN
If function to be evaluated
N
Y
Evaluate appropriate function
Multiply FCOST by function value output function if needed
If another function
N
Y
Assign FCOST to RPATH

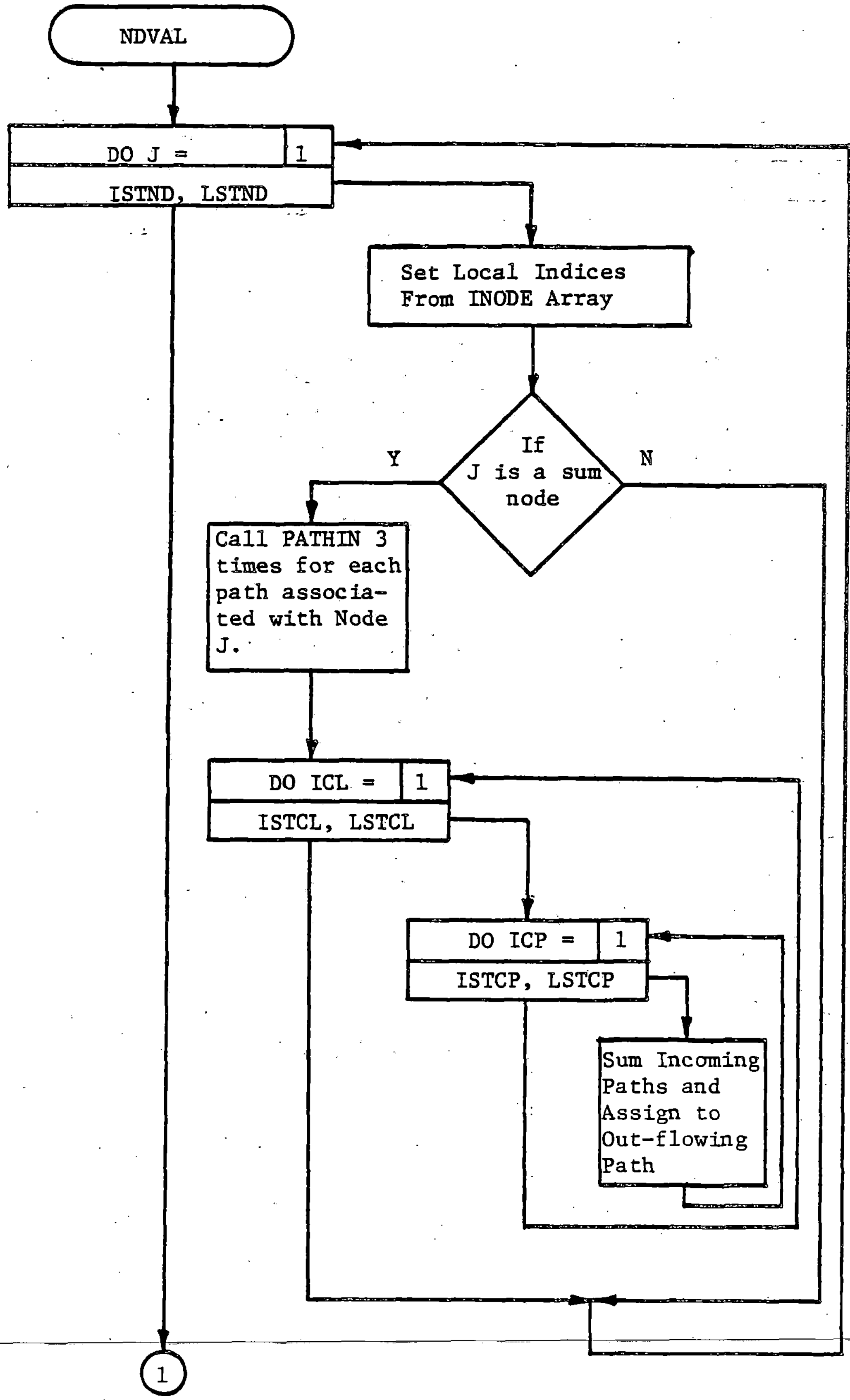

CDFUNC
Initialize control indices from arguemnt list
If function not necessary
N
Y
Call FUNCTS to evaluate the function
Calculate new split fraction
Restore function address
If using reference path
Y
N
Maintain split fraction between one and zero inclusive
RETURN


ADDRCH
Initialize Control
Indice from Argument List
If
Addresses
are to be
saved
Y
N
DO K=
1
IND1, INDL
Store Addresses
From RPATH in RFC
DO K=
1
IND1, INDL
Restore Addresses
From RFC to RPATH
Return

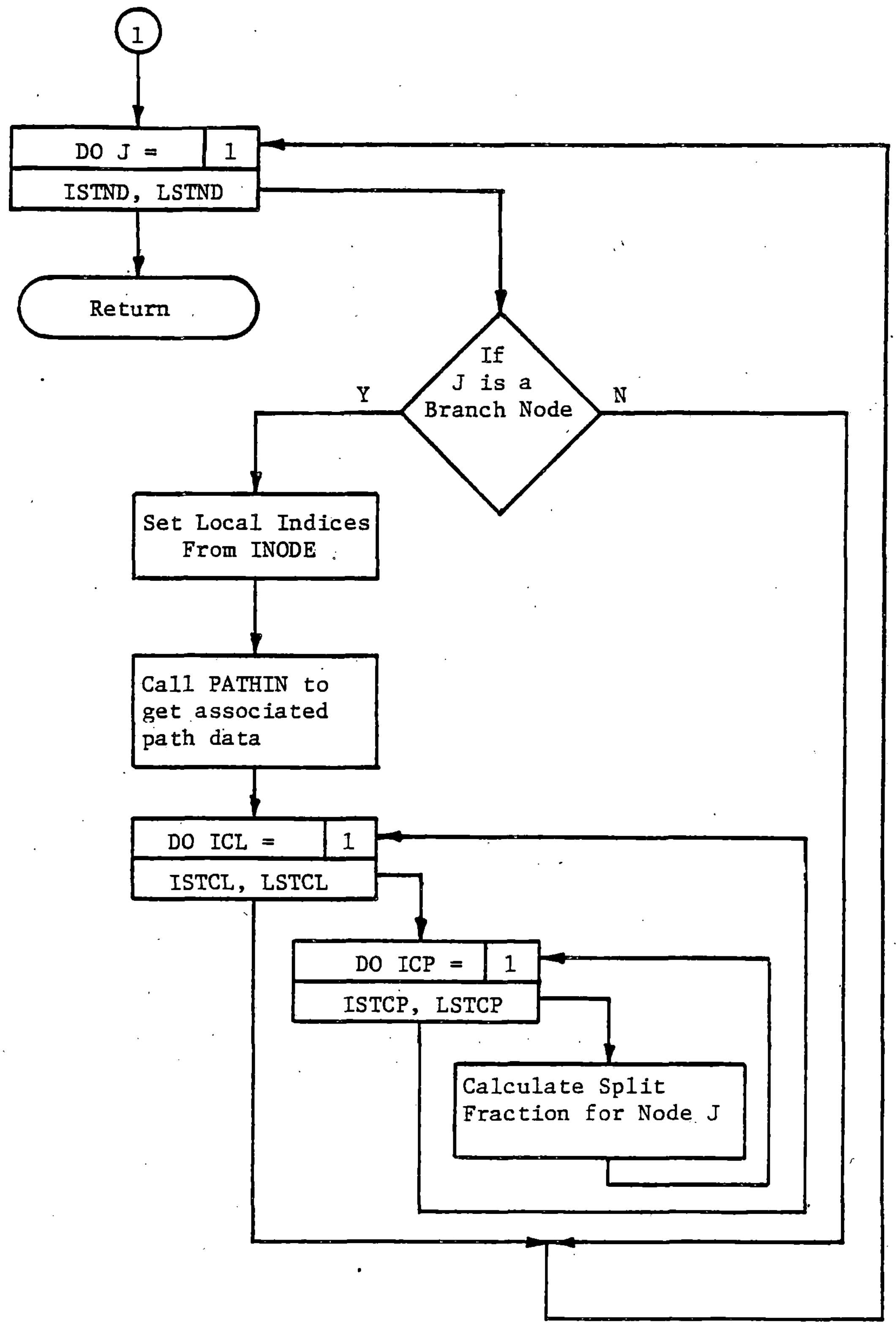

OUTPT
Initialize control indices from IFILE and RFILE arrays, and from Argument List
IF INTERMEDIATE OUTPUT
IF HEADING NEEDED
N
Y
Output Heading
IF FINAL OUTPUT
Y
N
Alter Heading
Initialize Local Indices
DO I =
1
I1, I2
Sum Path Costs
Sum Path Cost to Total Cost
IF PATH FLOW EQUAL ZERO
Y
N
1
2
IF COST TABLE REQUIRED
Y
N
Alter Heading
Output Cost Tables by cost code by component, by class, and by repair block
3

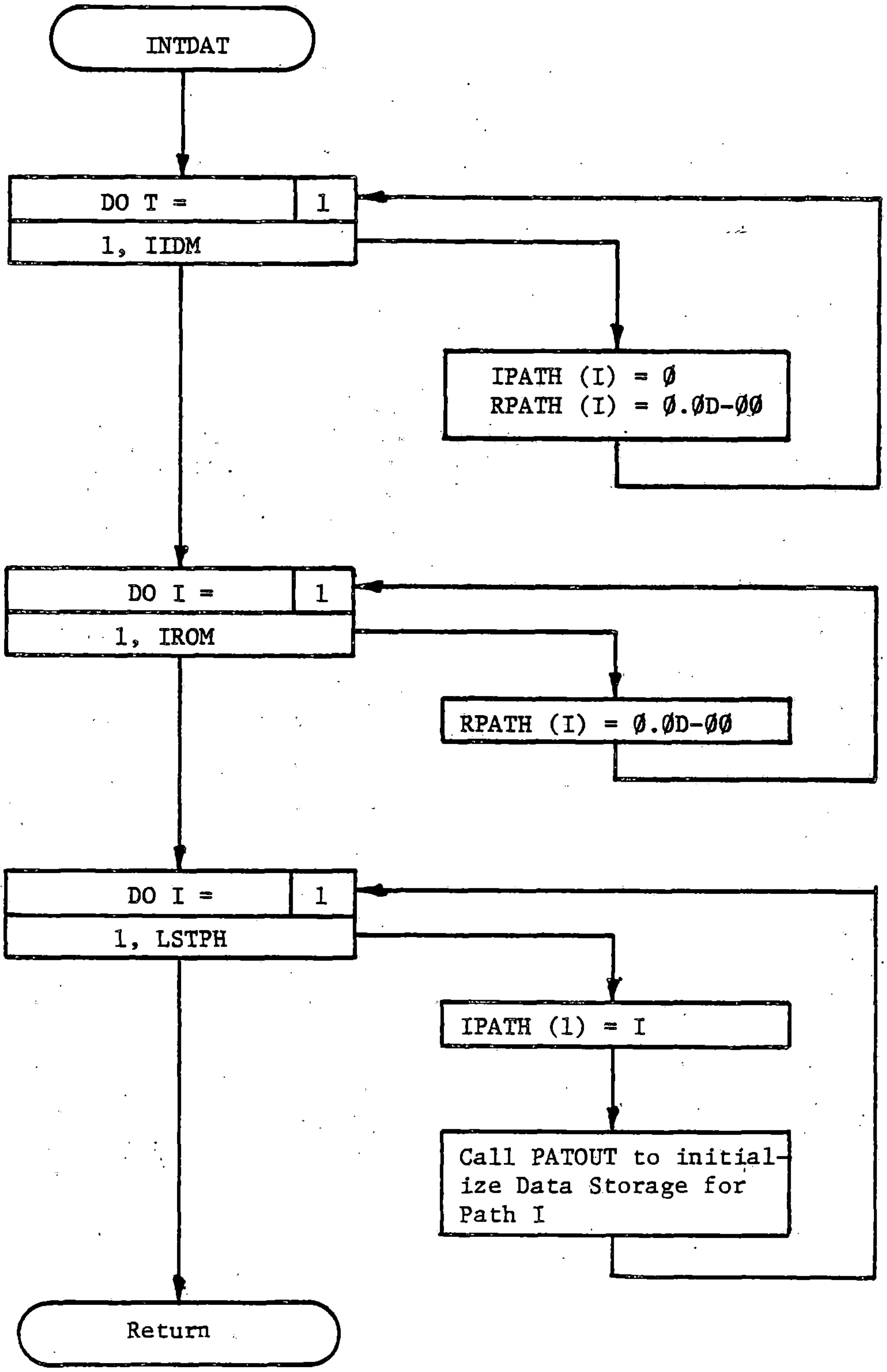

1
2
3
Output Path Flow, Quality and Cost by Class and Component
IF STATE TABLE NEEDED
Y
N
Alter Heading
Output System Population and Quality
Return

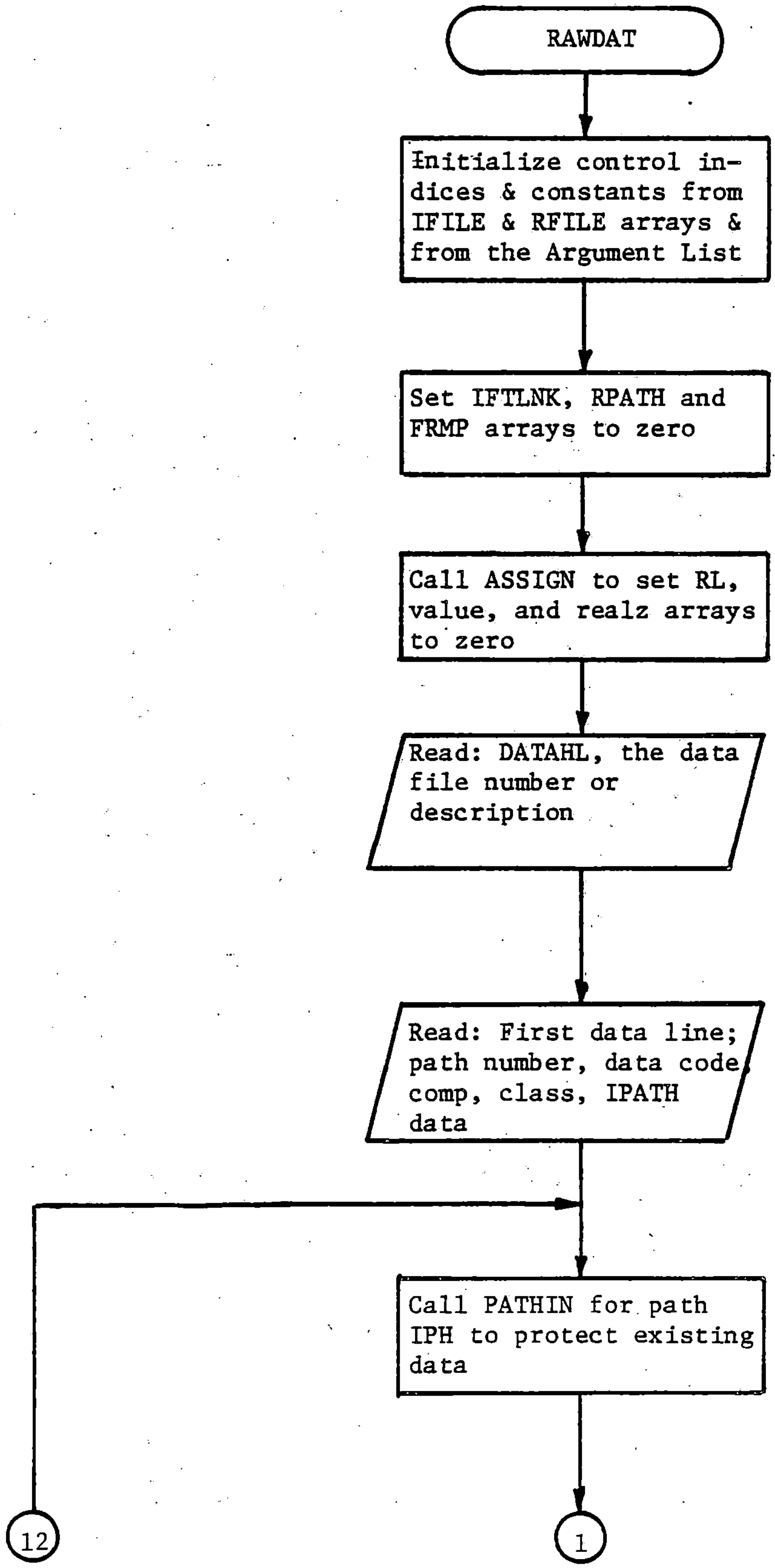

HEADNG
Initialize Indices from IFILE array and from Argument List
Output Top Heading of Program Revision, Node Map Data File, Number of Classes Components, Paths and Nodes, Start time, End Time and Time Step
DO ICL = 1
1, LSTCL
DO ICP = 1
1, LSTCP
Output Heading Function
Output, Characteristic Life, Av. Age, Weibull
IF FUNCTION TO BE OUTPUTTED
Y
N
Output Function Parameters
Increment Function Number
Return

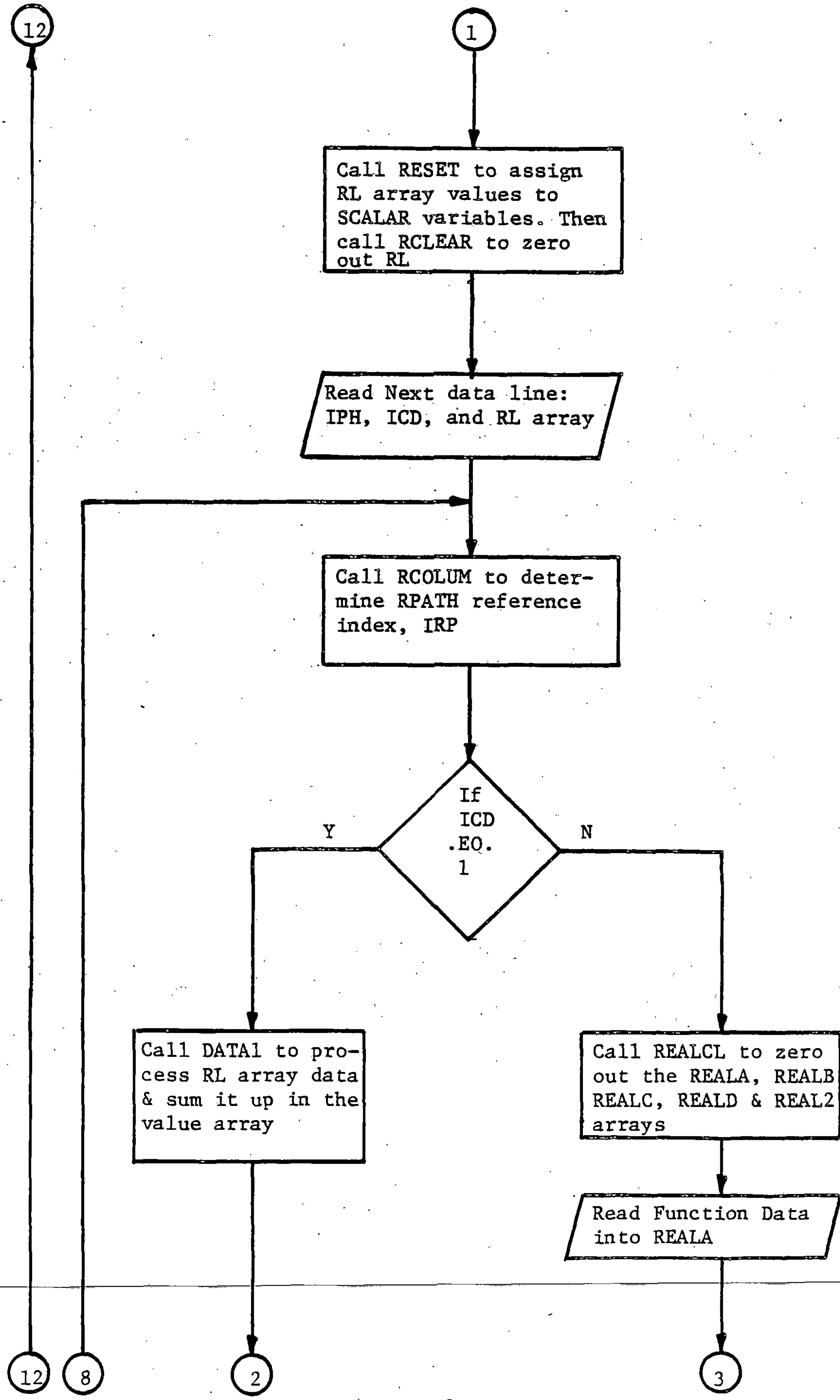


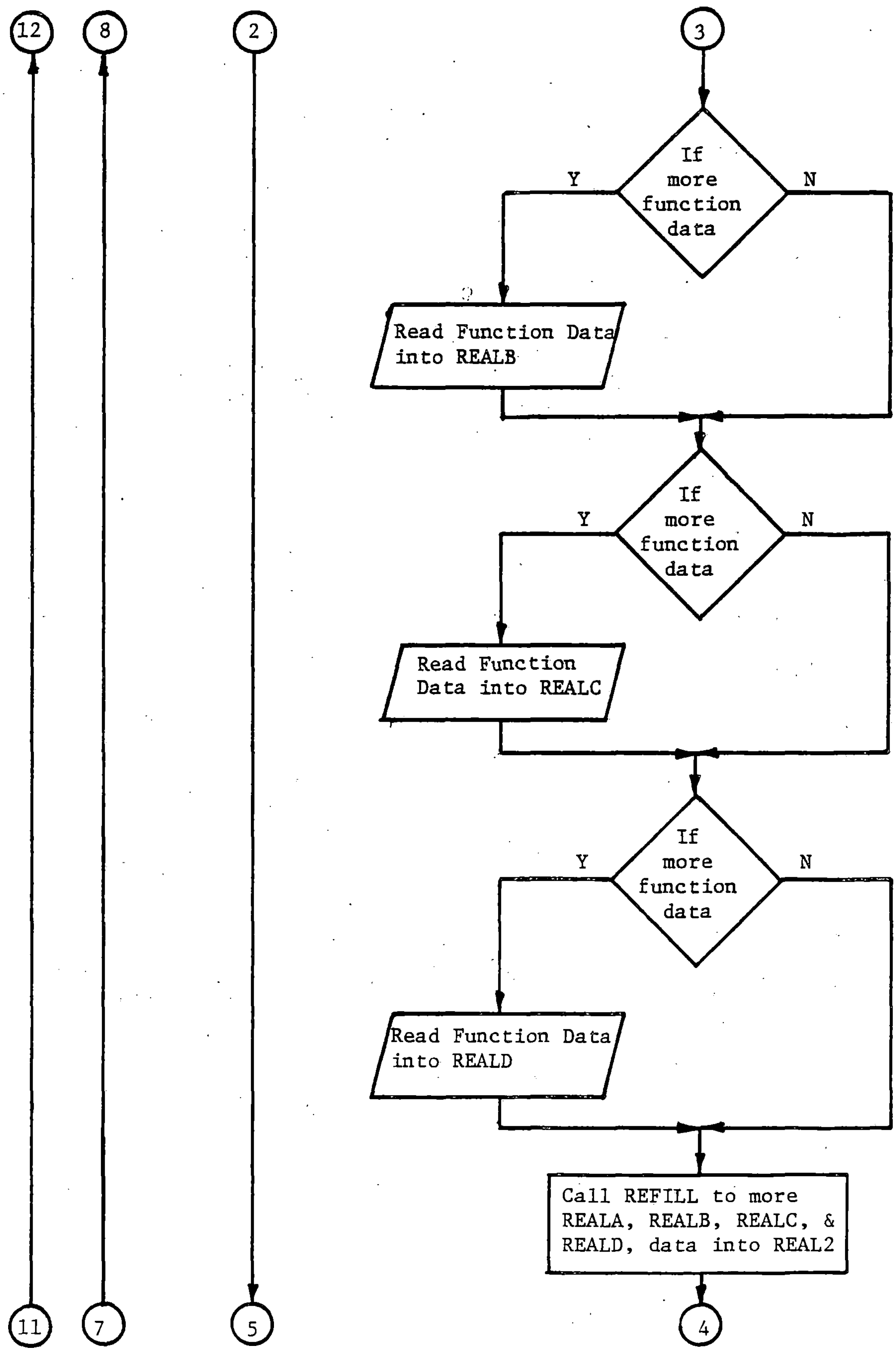
NDVAL
DO J = 1
ISTND, LSTND
Set Local Indices From INODE Array
If J is a sum node
Y
N
Call PATHIN 3 times for each path associated with Node J.
DO ICL = 1
ISTCL, LSTCL
DO ICP = 1
ISTCP, LSTCP
Sum Incoming Paths and Assign to Out-flowing Path
1

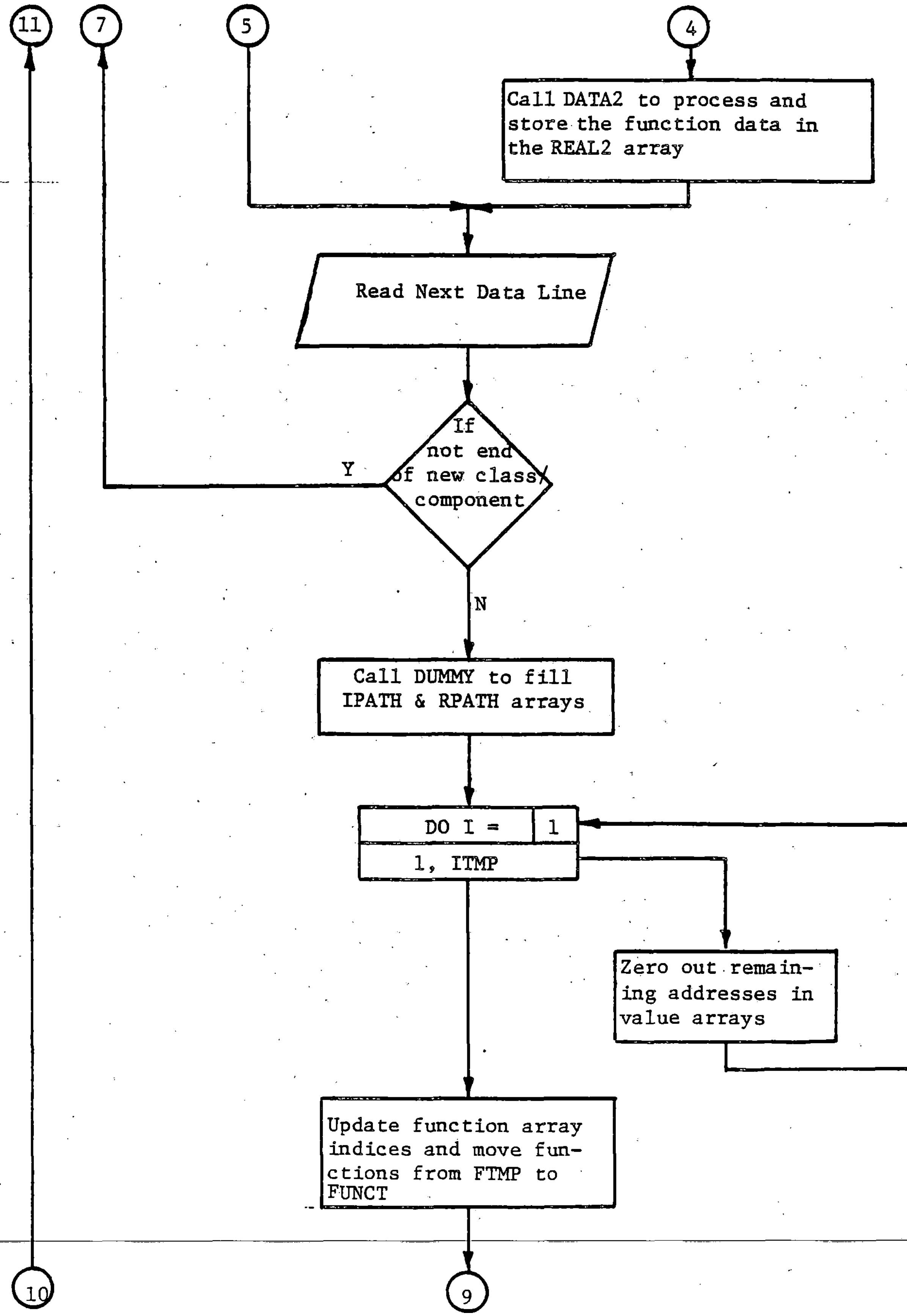

1
DO J = 1
ISTND, LSTND
Return
If J is a Branch Node
Y
N
Set Local Indices From INODE
Call PATHIN to get associated path data
DO ICL = 1
ISTCL, LSTCL
DO ICP = 1
ISTCP, LSTCP
Calculate Split Fraction for Node J

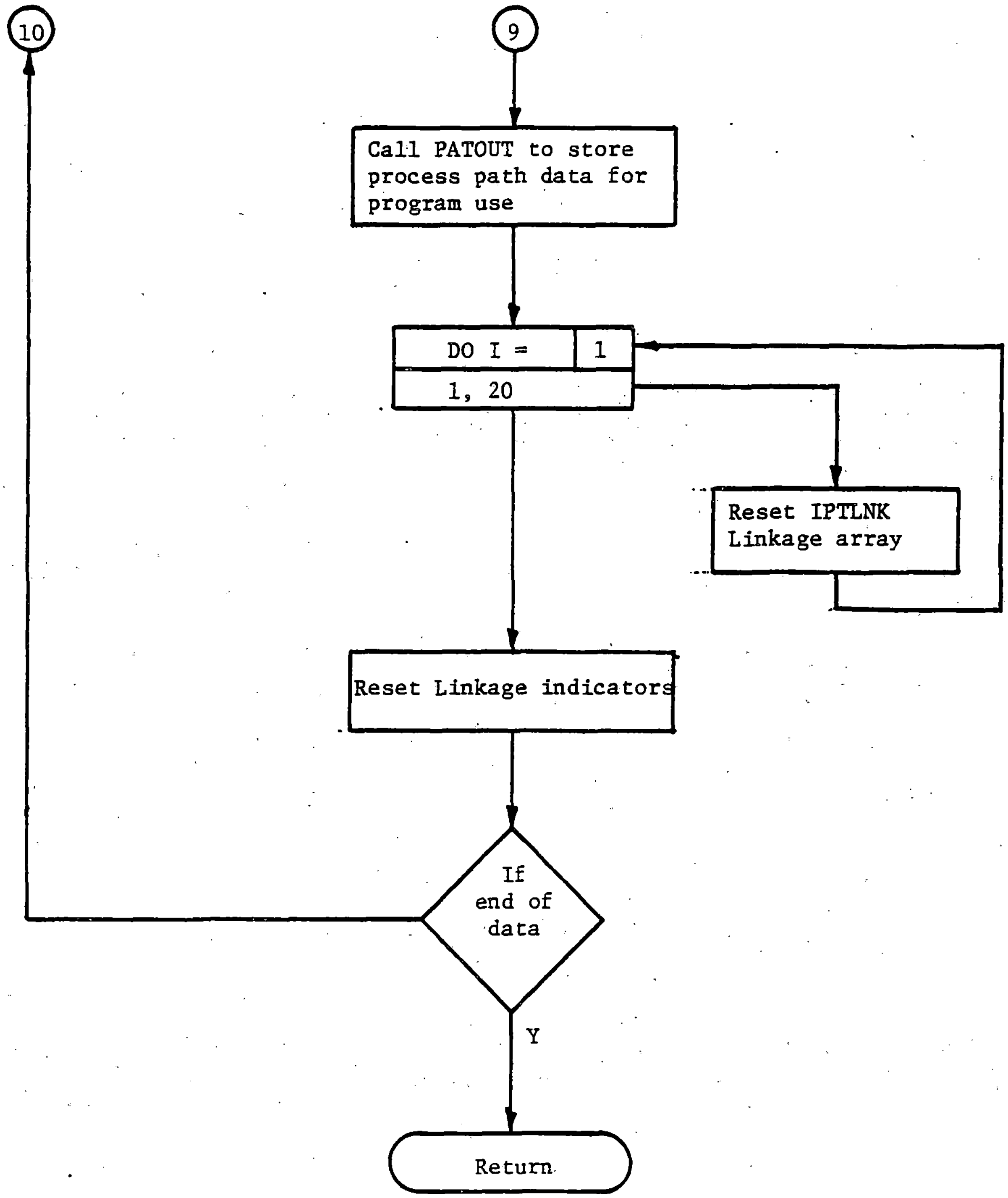

INTDAT
DO T =
1
1, IIDM
IPATH (I) = Ø
RPATH (I) = Ø.ØD-ØØ
DO I =
1
1, IROM
RPATH (I) = Ø.ØD-ØØ
DO I =
1
1, LSTPH
IPATH (1) = I
Call PATOUT to initial-
ize Data Storage for
Path I
Return


RAWDAT
Initialize control indices & constants from IFILE & RFILE arrays & from the Argument List
Set IFTLNK, RPATH and FRMP arrays to zero
Call ASSIGN to set RL, value, and realz arrays to zero
Read: DATAHL, the data file number or description
Read: First data line; path number, data code, comp, class, IPATH data
Call PATHIN for path IPH to protect existing data
12
1


12
1
Call RESET to assign RL array values to SCALAR variables. Then call RCLEAR to zero out RL
Read Next data line: IPH, ICD, and RL array
Call RCOLUM to determine RPATH reference index, IRP
If ICD .EQ. 1
Y
N
Call DATA1 to process RL array data & sum it up in the value array
Call REALCL to zero out the REALA, REALB REALC, REALD & REAL2 arrays
Read Function Data into REALA
12
8
2
3


12
8
2
3
If more function data
Y
N
Read Function Data into REALB
If more function data
Y
N
Read Function Data into REALC
If more function data
Y
N
Read Function Data into REALD
Call REFILL to more REALA, REALB, REALC, & REALD, data into REAL2
11
7
5
4


11
7
5
4
Call DATA2 to process and store the function data in the REAL2 array
Read Next Data Line
If not end of new class/ component
Y
N
Call DUMMY to fill IPATH & RPATH arrays
DO I =
1
1, ITMP
Zero out remaining addresses in value arrays
Update function array indices and move functions from FTMP to FUNCT
10
9


10
9
Call PATOUT to store process path data for program use
DO I =
1
1, 20
Reset IPTLNK Linkage array
Reset Linkage indicators
If end of data
Y
Return

APPENDIX D

Computer Program Listing

```
C******************************************************************
C                                                                *
C                        PROGRAM CONTROL                         *
C                                                                *
C******************************************************************
C                                                                *
C CONTROL IS A RAIL ROAD TRACK MAINTENANCE COST MODEL DESIGNED   *
C BY SHAKER RESEARCH CORPORATION FOR THE D.O.T.. THE ORIGINAL    *
C VERSION ONE WAS AN INTERACTIVE PROGRAM PRESENTING OPTIONS TO   *
C THE USER AND ACTING UPON SUBSEQUENT INPUT. VERSIONS THREE      *
C THROUGH SIX ARE BATCH MODE PROGRAMS, CONTROLLED BY THE IFILE   *
C AND RFILE DATA FILES. BOTH IFILE AND RFILE INCLUDE DESCRIP-    *
C TIONS FOR THE CONTROL VALUES CONTAINED                         *
C                                                                *
C******************************************************************
C                                                                *
C                           PROGRAM HISTORY                      *
C                                                                *
C  ORIGINAL VERSION ............. R. SMITH, N. LIPSTEIN, J. BE-*
C                                 TOR    SHAKER  07/79 - 02/80 *
C  CONTRL/2 ..................... R. SMITH, J.BETOR   SHAKER   *
C                                 02/80 - 03/80                *
C  CONTRL/3 ..................... R. SMITH, J. BETOR  SHAKER   *
C                                 06/80                        *
C  CONTRL/4 ..................... R. SMITH, J. BETOR  SHAKER   *
C                                 06/80                        *
C  CONTRL/5 ..................... R. SMITH, J. BETOR  SHAKER   *
C                                 06/80 - 07/80                *
C  CONTRL/6 ..................... R. SMITH, J. BETOR  SHAKER   *
C                                 07/16/80                     *
C                                                                *
C******************************************************************
C******************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C//////////////////////////////////////////////////////////////////
C SET UP COMMON AND DIMENSION ARRAYS.
C//////////////////////////////////////////////////////////////////
      COMMON/I1/IERROR(10)
      COMMON/I2/IFILE(100)
      COMMON/I3/IFLAG(10)
      COMMON/I7/INODE(6,300)
      COMMON/I8/IARAY(15,600)
C
      COMMON/R1/ACCUM(750)
      COMMON/R2/FUNCT(10,10)
      COMMON/R3/RNODE(30,300)
      COMMON/R4/STATE(20,30)
      COMMON/R5/RFILE(100)
      COMMON/R6/WEIBL(3,30)
      COMMON/R7/REVISN(8),SCHMDG(8),DATAFL(8)
```

```
      COMMON/R8/RARAY(750,600)
      DIMENSION RPATH(750),IPATH(15)
      DIMENSION PLACE(10),RUNN(10)
C////////////////////////////////////////////////////////////////////
C INITIALIZE CONTROL INDICES AND CONSTANTS.
C////////////////////////////////////////////////////////////////////
      REVISN(8) = '0006      '
      CALL DATAIN(ISTND,LSTND,ISTPH,LSTPH,ISTCP,LSTCP,
     *ISTCL,LSTCL,LSTFL,ISTCT,LSTCT,IIDM,IRDM,IBDM,NDOUT,
     *IUNIT,IDCODE,NREFP,PLACE,RUNN)
C//////////////////////////////////////////////////////////////////////
C CHECK IFILE FOR REQUIRED OPERATIONS.
C//////////////////////////////////////////////////////////////////////
      IF(IFILE(41) .EQ. 0) GO TO 100
         CALL INTDAT(LSTPH,IIDM,IRDM)
         CALL RAWDAT(LSTCP,LSTCL,IDCODE,IIDM,IRDM,IBDM,ISTCT,LSTCT)
         CALL HDVAL(IRDM,IIDM,ISTND,LSTND,ISTCL,LSTCL,ISTCP,
     *   LSTCP,IBDM,NREFP)
100   IF(IFILE(42) .EQ. 0) GO TO 9999
         CALL SWEEP(ISTND,LSTND,ISTCP,LSTCP,ISTCL,LSTCL,
     *   LSTFL,ISTCT,LSTCT,IIDM,IRDM,IBDM,IUNIT,
     *   RUNN,PLACE)
9999  CALL EXIT
      END
PROGRAM SIZE:   PROCEDURE - 000214    LINKAGE - 006113    STACK - 000012
0000 ERRORS [<.MAIN.>FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE SWEEP(ISTND,LSTND,ISTCP,LSTCP,
     *ISTCL,LSTCL,LSTFL,ISTCT,LSTCT,IIDM,IRDM,
     *IBDM,IUNIT,RUNN,PLACE)
      IMPLICIT REAL*8(A-H,O-Z)
C*******************************************************************
C THIS SUBROUTINE COMPUTES SYSTEM COSTS AND FLOWS                   *
C THROUGH TIME BY CALLING THE SUBROUTINE NORMAL                     *
C FOR GIVEN TIME STEPS.                                             *
C                                                                   *
C              ARGUMENT LIST                                        *
C         SEE IFILE AND RFILE FOR MORE                              *
C         COMPLETE DESCRIPTIONS.                                    *
C                                                                   *
C   ISTND  - STARTING NODE                                          *
C   LSTND  - LAST NODE                                              *
C   ISTCP  - INITIAL COMPONENT                                      *
C   LSTCP  - LAST COMPONENT                                         *
C   ISTCL  - INITIAL CLASS                                          *
C   LSTCL  - LAST CLASS                                             *
C   LSTFL  - NUMBER OF FLOW COLUMNS                                 *
C   ISTCT  - NUMBER OF COLUMNS FROM REFERENCE TO FIRST COST COLUMN  *
C   LSTCT  - NUMBER OF COST COLUMNS FOLLOWING FIRST                 *
C   IIDM   - DIMENSION OF IPATH                                     *
C   IRDM   - DIMENSION OF RPATH ARRAYS                              *
C   IBDM   - DIMENSION OF CLASS/COMPONENT BLOCK IN RPATH ARRAY      *
C   IUNIT  - FORTRAN OUTPUT UNIT DEVICE NUMBER                      *
C   RUNN   - RUN IDENTIFICATION                                     *
C   PLACE  - USER IDENTIFICATION                                    *
C*******************************************************************
C
      COMMON/I1/IERROR(10)
      COMMON/I2/IFILE(100)
      COMMON/I3/IFLAG(10)
      COMMON/I7/INODE(6,300)
C
      COMMON/R1/ACCUM(750)
      COMMON/R2/FUNCT(10,10)
      COMMON/R3/RNODE(30,300)
      COMMON/R4/STATE(20,30)
      COMMON/R5/RFILE(100)
      COMMON/R6/WEIBL(3,30)
C
      DIMENSION RPATH(750),RUNN(10),PLACE(10)
C//////////////////////////////////////////////////////////////////
C INITIALIZE LOOP PARAMETERS AND LOOP INDEPENDENT CONTROL INDICES
C//////////////////////////////////////////////////////////////////
      STIME  = RFILE(1)
```

```
          DTIME   = RFILE(2)
          TIMEL   = RFILE(3)
          N1      = IFILE(31)
          N2      = IFILE(32)
          LNTHST  = IFILE(24)
          NDOUT   = IFILE(29)
          IFMT    = IFILE(30)
          RFILE(50) = RFILE(1)
C/////////////////////////////////////////////////////////////////////
C INITIALIZE ACCUME AND STATE ARRAYS AND BEGIN TIME LOOP.
C////////////////////////////////////////////////////////////////////
          CALL HEADNG(IUNIT,DTIME,TIMEL,STIME,LSTCP,LSTCL,LSTPH,LSTND,
         *           IBDM)
100       CALL SETARR(IHEAD,N1,N2,IUNIT,IRDM,IBDM,IIDM)
          CALL NORMAL(ISTND,LSTND,ISTCP,LSTCP,ISTCL,
         *LSTCL,ISTCT,LSTCT,IIDM,IRDM,IBDM,IUNIT,
         *RUNN,PLACE,IHEAD,DTIME,STIME)
          IF(STIME .GE. TIMEL)GO TO 150
          N1 = IFILE(33)
          N2 = IFILE(34)
          CALL ABNRML(ISTND,LSTND,ISTCP,LSTCP,ISTCL,LSTCL,IIDM,
         *IRDM,IBDM,IUNIT,RUNN,PLACE,DTIME,STIME,ISTCT,LSTCT)
          STIME = STIME + DTIME
          RFILE(50) = STIME
          GO TO 100
C/////////////////////////////////////////////////////////////////////
C CALL OUTPT FOR FINAL PRINT OUT IF INDICATED.
C/////////////////////////////////////////////////////////////////////
150       IF(NDOUT .EQ. 1)CALL OUTPT(NDOUT,NODE,IP,ICL,ICP,RPATH,IRDM,
         *IIDM,RUNN,IFMT,PLACE,ARGN,IUNIT,ISTCT,LSTCT,ISTCP,LSTCP,
         *ISTCL,LSTCL,IBDM,STIME,PCOST1)
          RETURN
          END
PROGRAM SIZE:   PROCEDURE - 000322    LINKAGE - 006007    STACK - 000070
0000 ERRORS [<SWEEP >FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE SETARR(IHEAD,N1,N2,IUNIT,IRDM,IBDM,IIDM)
      IMPLICIT REAL*8(A-H,O-Z)
C*****************************************************************
C THIS WILL RESET THE VALUES IN THE ACCUM AND STATE ARRAYS FOR *
C USE WITH NORMAL AND ABNRML.                                   *
C                                                               *
C                          ARGUMENT LIST                        *
C                 (SEE IFILE AND RFILE DESCRIPTIONS)            *
C                                                               *
C     IHEAD    - INDICATOR OF HEADING FOR NORMAL OUTPUT.        *
C     N1       - ROW IN STATE TO SET PRESENT POPULATION FROM.   *
C     N2       - ROW IN STATE TO SET PRESENT QUALITY FROM.      *
C    IRDM    - DIMENSION OF RPATH AND ACCUM                     *
C    IUNIT   - FORTRAN OUTPUT UNIT DEVICE NUMBER                *
C    IBDM    - CLASS/COMPONENT BLOCK SIZE                       *
C    IIDM    - IPATH DIMENSION                                  *
C*****************************************************************
      COMMON/I2/IFILE(100)
      COMMON/R1/ACCUM(750)
      COMMON/R4/STATE(20,30)
C
      DIMENSION RPATH(750),IPATH(15)
C/////////////////////////////////////////////////////////////////
C INITIALIZE ARRAYS AND VARIABLES.
C/////////////////////////////////////////////////////////////////
      NREFP  = IFILE(13)
      LNTHST = IFILE(24)
      CALL PATHIN(NREFP,IPATH,IIDM,RPATH,IRDM)
C/////////////////////////////////////////////////////////////////
C RESET ACCUM ARRAY.
C/////////////////////////////////////////////////////////////////
      DO 100 I = 1,IRDM
         ACCUM(I) = 0.00
100   CONTINUE
C/////////////////////////////////////////////////////////////////
C RESET STATE ARRAY.
C/////////////////////////////////////////////////////////////////
      DO 300 I = 1,LNTHST
            STATE(5,I) = STATE(N1,I)
            STATE(6,I) = 0.00
            STATE(7,I) = STATE(N2,I)
            STATE(8,I) = 0.00
         DO 200 K = 9,20
            STATE(K,I) = 0.00
200      CONTINUE
300   CONTINUE
      IHEAD = 1
```

```
C/////////////////////////////////////////////////////////////////////
C INITIALIZE IN USE PATH ON DISK.
C/////////////////////////////////////////////////////////////////////
      DO 500 I = 1,LNTHST
         IRP = (I-1)*IBDM + 1
         IRP2 = IRP + 1
         RPATH(IRP)  = STATE(5,I)
         RPATH(IRP2) = STATE(7,I)
         CALL PATOUT(NREFP,IPATH,IIDM,RPATH,IRDM)
500   CONTINUE
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000223    LINKAGE - 005763    STACK - 000042
0000 ERRORS [<SETARR>FTN-REV16.6]
```

```
C***************************************************************
C***************************************************************
C***************************************************************
      SUBROUTINE ABNRML(ISTND,LSTND,ISTCP,LSTCP,ISTCL,LSTCL,
     *IIDM,IRDM,IBDM,IUNIT,RUNN,PLACE,DT,STIME,ISTCT,LSTCT)
      IMPLICIT REAL*8(A-H,O-Z)
C***************************************************************
C ABNRML WILL MAKE THREE TIME SIMULATION SWEEPS THROUGH THE NO-*
C DAL ARRAY TO INTERPRET POPULATION AND QUALITY TRENDS. ABNRML *
C IS A CONDENSED FORM OF THE SUBROUTINE NORMAL WITH NO COST     *
C COMPUTATION AND EXTRANEOUS COMMENTS AND SUBROUTINE CALLS E-  *
C LIMINATED.                                                   *
C                                                              *
C                          ARGUMENT LIST                       *
C                (SEE IFILE & RFILE FOR MORE DETAIL)           *
C                                                              *
C        ISTND  - FIRST NODE                                   *
C        LSTND  - LAST NODE                                    *
C        ISTCP  - FIRST COMPONENT                              *
C        LSTCP  - LAST COMPONENT                               *
C        ISTCL  - FIRST CLASS                                  *
C        LSTCL  - LAST CLASS                                   *
C        IIDM   - DIMENSION OF IPATH                           *
C        IRDM   - DIMENSION OF RPATH                           *
C        IBDM   - DIMENSION OF CLASS/COMPONENT BLOCK           *
C        IUNIT  - FORTRAN PRINT DEVICE NUMBER                  *
C        RUNN   - RUN NUMBER                                   *
C        PLACE  - GROUP RUNNING PROGRAM                        *
C        DT     - TIME STEP IN YEARS                           *
C        STIME  - CURRENT SIMULATION TIME                      *
C        ISTCT  - NUMBER OF COLUMNS FROM REFERENCE TO FIRST COST*
C                 COLUMN                                       *
C        LSTCT  - NUMBER OF COLUMNS FROM FIRST TO LAST COST CO- *
C                 LUMN                                         *
C***************************************************************
      COMMON/I2/IFILE(100)
      COMMON/I3/IFLAG(10)
      COMMON/I7/INODE(6,300)
C
      COMMON/R1/ACCUM(750)
      COMMON/R2/FUNCT(10,10)
      COMMON/R3/RNODE(30,300)
      COMMON/R4/STATE(20,30)
      COMMON/R5/RFILE(100)
      COMMON/R6/WEIBL(3,30)
C
      DIMENSION IPATH1(15),IPATH2(15),IPATH3(15)
      DIMENSION RUNN(10),RPATH1(750),RPATH2(750),RPATH3(750)
      DIMENSION RPATH(750),PLACE(10)
C///////////////////////////////////////////////////////////////
```

```
C SET CONTROL INDICES.
C////////////////////////////////////////////////////////////////
      IOSIM  = IFILE(27)
      OUTLMD = RFILE(6)
      IFMTS  = IFILE(28)
      TIMSTP = 0.00
      NRPTH  = IFILE(13)
      DTMP   = 0.5
      LNTHST = IFILE(24)
C
C////////////////////////////////////////////////////////////////
C BEGIN TIME SIMULATIONS.
C////////////////////////////////////////////////////////////////
      DO 2000 NPASS = 9,17,4
         DTMP1 = DTMP * DT
         CALL DRATE(DTMP1,NPASS)
         RFILE(50) = STIME + DTMP1
         DO 1900 IIROW = ISTND,LSTND
            IFLAG(4) = IIROW
            IP1      = INODE(1,IIROW)
            IP2      = INODE(2,IIROW)
            IP3      = INODE(3,IIROW)
            CALL PATHIN(IP1,IPATH1,IIDM,RPATH1,IRDM)
            CALL PATHIN(IP2,IPATH2,IIDM,RPATH2,IRDM)
            CALL PATHIN(IP3,IPATH3,IIDM,RPATH3,IRDM)
C////////////////////////////////////////////////////////////////
C SET NODE DEPENDENT INDICES.
C////////////////////////////////////////////////////////////////
            IBRSH = INODE(4,IIROW)
            ICDEF = INODE(5,IIROW)
            IREFP = INODE(6,IIROW)
C////////////////////////////////////////////////////////////////
C SET PATH DEPENDENT INDICES FOR THIS NODE.
C////////////////////////////////////////////////////////////////
           IPDT1 = IABS(IPATH1(11))
           IPDT2 = IABS(IPATH2(11))
           IPDT3 = IABS(IPATH3(11))
           ISGN1 = IPATH1(11)
           ISGN2 = IPATH2(11)
           ISGN3 = IPATH3(11)
           IQDT2 = IPATH2(12)
           IQDT3 = IPATH3(12)
           IMFG1 = IPATH1(13)
           IMFG2 = IPATH2(13)
           IMFG3 = IPATH3(13)
C////////////////////////////////////////////////////////////////
C LOOP THROUGH CLASSES
C////////////////////////////////////////////////////////////////
           DO 1800 ICL = ISTCL,LSTCL
C////////////////////////////////////////////////////////////////
```

```
C LOOP THROUGH COMPONENTS.
C//////////////////////////////////////////////////////////////
                 DO 1700 ICP = ISTCP,LSTCP
                    CALL RCOLUM(ICL,LSTCL,ICP,LSTCP,IRP,IRCOL,IBDM)
                    CDE = RNODE(IRCOL,IIROW)
C//////////////////////////////////////////////////////////////
C CALCULATE FLOWS AND CHANGES.
C//////////////////////////////////////////////////////////////
                    IF(IPDT1 .EQ. 1)CALL PDOT(IRP,IRCOL,RPATH1,
     *                                IRDM,DTMP,ISGN1,NPASS)
                    IF(IBRSH .EQ. 1)CALL BRANCH(ICDEF,IREFP,CDE,
     *                                IRP,RPATH1,RPATH2,RPATH3,IRDM,
     *                                IBDM,NPASS,NRPTH,IP1)
                    IF(IPDT2 .EQ. 1)CALL PDOT(IRP,IRCOL,RPATH2,
     *                                IRDM,DTMP,ISGN2,NPASS)
                    IF(IQDT2 .EQ. 1)CALL QDOT(IRP,IRCOL,RPATH2,
     *                                IRDM,DTMP,ISGN2,NPASS)
                    IF(IBRSH .EQ. 0)CALL SUMS(IMFG1,IMFG2,IRP,
     *                                RPATH1,RPATH2,RPATH3,IRDM)
                    IF(IPDT3 .EQ. 1)CALL PDOT(IRP,IRCOL,RPATH3,
     *                                IRDM,DTMP,ISGN3,NPASS)
                    IF(IQDT3 .EQ. 1)CALL QDOT(IRP,IRCOL,RPATH3,
     *                                IRDM,DTMP,ISGN3,NPASS)
1700             CONTINUE
C//////////////////////////////////////////////////////////////
1800          CONTINUE
C//////////////////////////////////////////////////////////////
1900       CONTINUE
           CALL WEIBUL(ISTCP,ISTCL,LSTCP,LSTCL,NPASS,IBDM,DT)
           TIMSTP = DTMP + TIMSTP
           DTMP   = TIMSTP
C//////////////////////////////////////////////////////////////
2000     CONTINUE
         IF(RFILE(5) .LT. OUTLMD) GO TO 2010
         RFILE(5) = 0.0
         IF(IOSIM .EQ. 3)CALL OUTPT(IOSIM,NODE,IP,ICL,ICP,RPATH,
     *                  IRDM,IIDM,RUNN,IFMTS,PLACE,ARGN,IUNIT,
     *                  ISTCT,LSTCT,ISTCP,LSTCP,ISTCL,LSTCL,IBDM,
     *                  STIME,PCOST1)
2010     RFILE(5) = RFILE(5) + 1.0
         DO 2100 I = 1,LNTHST
            RCP1 = STATE(6,I)
            RCQ1 = STATE(8,I)
            RCP2 = STATE(10,I)
            RCQ2 = STATE(12,I)
            RCP3 = STATE(14,I)
            RCQ3 = STATE(16,I)
            RCP4 = STATE(18,I)
            RCQ4 = STATE(20,I)
            RCPF = (RCP1 + 2.0*RCP2 + 2.0*RCP3 + RCP4)/6.000
```

```
            RCQF = (RCQ1 + 2.0*RCQ2 + 2.0*RCQ3 + RCQ4)/6.000
            STATE(17,I) = STATE(5,I) + (DT * RCPF)
            STATE(19,I) = STATE(7,I) + (DT * RCQF)
 2100   CONTINUE
        RETURN
        END
PROGRAM SIZE:   PROCEDURE - 001067    LINKAGE - 027716    STACK - 000100
0000 ERRORS [<ABNRML>FTN-REV16.6]
```

```
C***************************************************************
C***************************************************************
C***************************************************************
      SUBROUTINE DRATE(DT,N1)
      IMPLICIT REAL*8(A-H,O-Z)
C***************************************************************
C DRATE WILL CALCULATE STATES FOR USE IN ABNRML.               *
C THESE                                                         *
C                                                               *
C                         ARGUMENT LIST                         *
C                                                               *
C     DT     -   THE CURRENTLY USED TIME STEP                   *
C     N1     -   THE STATE ARRAY REFERENCE ROW.                 *
C                                                               *
C***************************************************************
      COMMON/I2/IFILE(100)
      COMMON/R4/STATE(20,30)
C///////////////////////////////////////////////////////////////
C INITIALIZE CONTROL INDICES.
C///////////////////////////////////////////////////////////////
      LNTHST = IFILE(24)
      N3     = N1 + 2
      NDP    = N1 - 3
      NDQ    = N1 - 1
C///////////////////////////////////////////////////////////////
C SET EXTRAPOLATED POPULATION.
C///////////////////////////////////////////////////////////////
      DO 100 I = 1,LNTHST
         RCP = STATE(NDP,I)
         RCQ = STATE(NDQ,I)
         STATE(N1,I) = STATE(5,I) + DT * RCP
         STATE(N3,I) = STATE(7,I) + DT * RCQ
100   CONTINUE
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000113    LINKAGE - 000047    STACK - 000024
0000 ERRORS [<DRATE >FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE NORMAL(ISTHD,LSTHD,ISTCP,LSTCP,ISTCL,LSTCL,
     *ISTCT,LSTCT,IIDM,IRDM,IBDM,IUNIT,RUNN,PLACE,IHEAD,DT,
     *STIME)
      IMPLICIT REAL*8(A-H,O-Z)
C****************************************************************
C     NORMAL SCHEMATIC DIAGRAM CALCULATION                      *
C     SWEEPS THRU NODE ARRAY ROW BY ROW                         *
C     PICKS UP THREE PATH FILES AT A TIME                       *
C     COMPUTES FLOWS QUALITIES AND COSTS                        *
C     STORES INFO BACK INTO PATH FILES                          *
C                                                               *
C                ARGUMENT LIST                                  *
C    (SEE IFILE AND RFILE FOR COMPLETE DESCRIPTIONS)            *
C                                                               *
C    ISTHD  - STARTING NODE OF SWEEP                            *
C    LSTHD  - FINAL NODE OF SWEEP                               *
C    ISTCP  - FIRST COMPONENT CODE                              *
C    LSTCP  - LAST COMPONENT CODE AND NUMBER OF COMPONENTS      *
C    ISTCL  - FIRST CLASS CODE                                  *
C    LSTCL  - LAST CLASS CODE AND NUMBER OF CLASSES             *
C    IIDM   - IPATH DIMENSION SIZE                              *
C    IRDM   - RPATH DIMENSION SIZE                              *
C    IBDM   - RPATH CLASS/COMPONENT BLOCK SIZE                  *
C    IUNIT  - FORTRAN PRINT DEVICE CODE                         *
C    RUNN   - PROGRAM RUN NUMBER                                *
C    PLACE  - PERSON OR GROUP RUNNING PROGRAM                   *
C    IHEAD  - INTERMEDIATE PRINTOUT HEADING INDICATOR           *
C    DT     - MAIN TIME LOOP INCREMENT                          *
C    STIME  - PRESENT SIMULATION TIME                           *
C****************************************************************
C
      COMMON/I1/IERROR(10)
      COMMON/I2/IFILE(100)
      COMMON/I3/IFLAG(10)
      COMMON/I7/INODE(6,300)
C
      COMMON/R1/ACCUM(750)
      COMMON/R2/FUNCT(10,10)
      COMMON/R3/RNODE(30,300)
      COMMON/R4/STATE(20,30)
      COMMON/R5/RFILE(100)
C
      DIMENSION IPATH1(15),IPATH2(15),IPATH3(15)
      DIMENSION RUNN(10),RPATH(750),RPATH1(750),RPATH2(750),
     *          RPATH3(750)
      DIMENSION PLACE(10),RFC1(750),RFC2(750),RFC3(750)
C/////////////////////////////////////////////////////////////////
```

```
C GET CONTROL INDICES.
C///////////////////////////////////////////////////////////////
      ICOST  = IFILE(35)
      NPASS  = IFILE(19)
      NRPTH  = IFILE(13)
      IOUT   = IFILE(15)
      PCOST1 = 0.0
      IVAL   = 0
C///////////////////////////////////////////////////////////////
C CHECK FOR PRINT TIME.
C///////////////////////////////////////////////////////////////
      IPRNT  = 0.0
      IF(RFILE(24) .LT. RFILE(25)) GO TO 155
      IPRNT  = IFILE(25)
      RFILE(24) = 0.0
155   RFILE(24) = RFILE(24) + 1.00
C///////////////////////////////////////////////////////////////
C BEGIN NODE SWEEP.
C///////////////////////////////////////////////////////////////
C     START NORMAL CALCULATIONS
      DO 1900  IIROW  = ISTND,LSTND
1707     FORMAT('NODE:',I4)
         IFLAG(4) = IIROW
         IP1 = INODE(1,IIROW)
         IP2 = INODE(2,IIROW)
         IP3 = INODE(3,IIROW)
         CALL  PATHIN(IP1,IPATH1,IIDM,RPATH1,IRDM)
         CALL  PATHIN(IP2,IPATH2,IIDM,RPATH2,IRDM)
         CALL  PATHIN(IP3,IPATH3,IIDM,RPATH3,IRDM)
         IF(IP1 .NE. IPATH1(1))CALL ERRORS(IP1)
         IF(IP2 .NE. IPATH2(1))CALL ERRORS(IP2)
         IF(IP3 .NE. IPATH3(1))CALL ERRORS(IP3)
C///////////////////////////////////////////////////////////////
C SET NODE CALL INDICES.
C///////////////////////////////////////////////////////////////
         IBRSH = INODE(4,IIROW)
         ICDEF = INODE(5,IIROW)
         IREFP = INODE(6,IIROW)
C///////////////////////////////////////////////////////////////
C SET PATH DEPENDENT CALL INDICES FOR THIS NODE.
C///////////////////////////////////////////////////////////////
        IMFG1  =  IPATH1(13)
        IMFG2  =  IPATH2(13)
        IMFG3  =  IPATH3(13)
        ICST1  =  IPATH1(10) * IMFG1 *  ICOST
        ICST2  =  IPATH2(10) * (IBRSH + IMFG2) * ICOST
        IF(ICST2 .GT. 0)ICST2 = 1
        ICST3  =  IPATH3(10) * ICOST
        IFCT1  =  ICST1
        IFCT2  =  ICST2
```

```
            IFCT3  =  ICST3
            IACM1  =  ICST1
            IACM2  =  ICST2
            IACM3  =  ICST3
            IPDT1  =  IABS(IPATH1(11))
            IPDT2  =  IABS(IPATH2(11))
            IPDT3  =  IABS(IPATH3(11))
            ISGN1  =  IPATH1(11)
            ISGN2  =  IPATH2(11)
            ISGN3  =  IPATH3(11)
            IQDT1  =  IPATH1(12)
            IQDT2  =  IPATH2(12) * IBRSH
            IQDT3  =  IPATH3(12)
            RFC1(1) = 0.0
            RFC2(1) = 0.0
            RFC3(1) = 0.0
            IPRT1  =  IABS(IPRNT * IMFG1) * (1 - IPATH1(14))
            IPRT2  =  IABS(IPRNT * (IMFG2 + IBRSH)) * (1 - IPATH2(14))
            IF(IPRT2 .GT. 0) IPRT2 = 1
            IPRT3  =  IABS(IPRNT) * (1 - IPATH3(14))
C////////////////////////////////////////////////////////////////
C DATA CHECKS.
C////////////////////////////////////////////////////////////////
            IF(IBRCH  .GT.  1)CALL  ERRORS(1000)
            IF(ICDEF  .GT.  1)CALL  ERRORS(1000)
            IF(IREFP  .GT.  1)CALL  ERRORS(1000)
C
            IF(IABS(ICST1) .GT. 1)CALL  ERRORS(1100)
            IF(IABS(ICST2) .GT. 1)CALL  ERRORS(1100)
            IF(IABS(ICST3) .GT. 1)CALL  ERRORS(1100)
            IF(IABS(IPDT1) .GT. 1)CALL  ERRORS(1100)
            IF(IABS(IPDT2) .GT. 1)CALL  ERRORS(1100)
            IF(IABS(IPDT3) .GT. 1)CALL  ERRORS(1100)
            IF(IABS(IQDT1) .GT. 1)CALL  ERRORS(1100)
            IF(IABS(IQDT2) .GT. 1)CALL  ERRORS(1100)
            IF(IABS(IQDT3) .GT. 1)CALL  ERRORS(1100)
            IF(IABS(IMFG1) .GT. 1)CALL  ERRORS(1100)
            IF(IABS(IMFG2) .GT. 1)CALL  ERRORS(1100)
            IF(IABS(IMFG3) .GT. 1)CALL  ERRORS(1100)
C
C////////////////////////////////////////////////////////////////
             DO 1800  ICL = ISTCL,LSTCL
C
C
C
C////////////////////////////////////////////////////////////////
                DO 1700  ICP = ISTCP,LSTCP
                   CALL RCOLUM(ICL,LSTCL,ICP,LSTCP,IRP,IRCOL,IBDM)
                   CDE = RNODE(IRCOL,IIROW)
C////////////////////////////////////////////////////////////////
```

```
C CALLS FOR PATH 1
C///////////////////////////////////////////////////////////////
C                 IF(IBRSH .EQ. 1)CALL BRANCH(ICDEF,IREFP,CDE,IRP,
C    *                              RPATH1,RPATH2,RPATH3,IRDM,IBDM,
C    *                              NPASS,NRPTH,IP1)
                  IF(IBRSH .EQ. 0)CALL SUMS(IMFG1,IMFG2,IRP,RPATH1,
     *                              RPATH2,RPATH3,IRDM)
                  IF(IFCT1 .EQ. 1)CALL FUNCTS(IRP,ISTCT,LSTCT,
     *                              RPATH1,IRDM,IP,RFC1,IBDM,NPASS,
     *                              IUNIT,IVAL,ICP,ICL)
                  IF(IPDT1 .EQ. 1)CALL PDOT(IRP,IRCOL,RPATH1,IRDM,
     *                              DT,ISGN1,NPASS)
C                 IF(IQDT1 .EQ. 1)CALL QDOT(IRP,IRCOL,RPATH1,IRDM,
C    *                              DT,ISGN1,NPASS)
                  IF(ICST1 .EQ. 1)CALL PCOSTS(IRP,ISTCT,LSTCT,
     *                              RPATH1,PCOST,RPATH,IRDM)
                  IF(IACM1 .EQ. 1)CALL ACCUME(IRP,ISTCT,LSTCT,RPATH,
     *                                          IRDM)
                  IF(IPRT1 .EQ. 1)CALL OUTPT(NDOUT,IIROW,IP1,ICL,
     *                              ICP,RPATH1,IRDM,IIDM,RUNN,IHEAD,
     *                              PLACE,ARGN,IUNIT,ISTCT,LSTCT,
     *                              ISTCP,LSTCP,ISTCL,LSTCL,IBDM,
     *                              STIME,PCOST1)
C///////////////////////////////////////////////////////////////
C CALLS FOR PATH 2
C///////////////////////////////////////////////////////////////
                  IF(IBRSH .EQ. 1)CALL BRANCH(ICDEF,IREFP,CDE,IRP,
     *                              RPATH1,RPATH2,RPATH3,IRDM,IBDM,
     *                              NPASS,NRPTH,IP1)
C                 IF(IBRSH .EQ. 0)CALL SUMS(IMFG1,IMFG2,IRP,RPATH1,
C    *                              RPATH2,RPATH3,IRDM)
                  IF(IFCT2 .EQ. 1)CALL FUNCTS(IRP,ISTCT,LSTCT,
     *                              RPATH2,IRDM,IP,RFC2,IBDM,NPASS,
     *                              IUNIT,IVAL,ICP,ICL)
                  IF(IPDT2 .EQ. 1)CALL PDOT(IRP,IRCOL,RPATH2,IRDM,
     *                              DT,ISGN2,NPASS)
                  IF(IQDT2 .EQ. 1)CALL QDOT(IRP,IRCOL,RPATH2,IRDM,
     *                              DT,ISGN2,NPASS)
                  IF(ICST2 .EQ. 1)CALL PCOSTS(IRP,ISTCT,LSTCT,
     *                              RPATH2,PCOST,RPATH,IRDM)
                  IF(IACM2 .EQ. 1)CALL ACCUME(IRP,ISTCT,LSTCT,RPATH,
     *                                          IRDM)
                  IF(IPRT2 .EQ. 1)CALL OUTPT(NDOUT,IIROW,IP2,ICL,
     *                              ICP,RPATH2,IRDM,IIDM,RUNN,IHEAD,
     *                              PLACE,ARGN,IUNIT,ISTCT,LSTCT,
     *                              ISTCP,LSTCP,ISTCL,LSTCL,IBDM,
     *                              STIME,PCOST1)
C///////////////////////////////////////////////////////////////
C CALLS FOR PATH 3
C///////////////////////////////////////////////////////////////
```

```
C                  IF(IBRSH .EQ. 1)CALL BRANCH(ICDEF,IREFP,CDE,IRP,
C       *                               RPATH1,RPATH2,RPATH3,IRDM,IBDM,
C       *                               NPASS,NRPTH,IP1)
C                  IF(IBRSH .EQ. 0)CALL SUMS(IMFG1,IMFG2,IRP,RPATH1,
C       *                               RPATH2,RPATH3,IRDM)
                   IF(IFCT3 .EQ. 1)CALL FUNCTS(IRP,ISTCT,LSTCT,
        *                               RPATH3,IRDM,IP,RFC3,IBDM,NPASS,
        *                               IUNIT,IVAL,ICP,ICL)
                   IF(IPDT3 .EQ. 1)CALL PDOT(IRP,IRCOL,RPATH3,IRDM,
        *                               DT,ISGN3,NPASS)
C                  IF(IQDT3 .EQ. 1)CALL QDOT(IRP,IRCOL,RPATH3,IRDM,
C       *                               DT,ISGN3,NPASS)
                   IF(ICST3 .EQ. 1)CALL PCOSTS(IRP,ISTCT,LSTCT,
        *                               RPATH3,PCOST,RPATH,IRDM)
                   IF(IACM3 .EQ. 1)CALL ACCUME(IRP,ISTCT,LSTCT,RPATH,
        *                                          IRDM)
                   IF(IPRT3 .EQ. 1)CALL OUTPT(NDOUT,IIROW,IP3,ICL,
        *                               ICP,RPATH3,IRDM,IIDM,RUNN,IHEAD,
        *                               PLACE,ARGN,IUNIT,ISTCT,LSTCT,
        *                               ISTCP,LSTCP,ISTCL,LSTCL,IBDM,
        *                               STIME,PCOST1)
 1700              CONTINUE
 1800         CONTINUE
C/////////////////////////////////////////////////////////////////
C PUT PATH INFO BACK ON DISK
C/////////////////////////////////////////////////////////////////
              RFC2(1) = 1.0
              RFC3(1) = 1.0
              IF(IOUT .EQ. 1)CALL ADDRCH(RPATH2,RFC2,IRP,ISTCT,
        *                                    LSTCT,IRDM)
              IF(IOUT .EQ. 1)CALL PATOUT(IP2,IPATH2,IIDM,RPATH2,
        *                                    IRDM)
              IF(IOUT .EQ. 1)CALL ADDRCH(RPATH3,RFC3,IRP,ISTCT,
        *                                    LSTCT,IRDM)
              IF(IOUT .EQ. 1)CALL PATOUT(IP3,IPATH3,IIDM,RPATH3,
        *                                    IRDM)
C/////////////////////////////////////////////////////////////////
 1900    CONTINUE
         IF(IPRT3 .EQ. 1) WRITE(IUNIT,352) PCOST1
 352     FORMAT(1X,64(1H-),/,45X,8HTOTAL  $,I12)
         PCOST1 = 0.0
         CALL WEIBUL(ISTCP,ISTCL,LSTCP,LSTCL,NPASS,IBDM,DT)
         RETURN
         END
PROGRAM SIZE:    PROCEDURE - 002074    LINKAGE - 051306    STACK - 000100
0000 ERRORS [<NORMAL>FTN-REV16.6]
```

```
C****************************************************************
C****************************************************************
C****************************************************************
      SUBROUTINE WEIBUL(ISTCP,ISTCL,LSTCP,LSTCL,NPASS,IBDM,DT)
      IMPLICIT REAL*8(A-H,O-Z)
C****************************************************************
C WEIBUL IS USED TO COMPUTE AND APPLY THE WEIBULL DISTRIBUTION *
C TO THE CHANGE IN SYSTEM QUALITY.                               *
C                                                                *
C                    ARGUMENT LIST                               *
C                                                                *
C          ISTCP  - THE FIRST COMPONENT                          *
C          ISTCL  - THE FIRST CLASS NUMBER                       *
C          LSTCP  - THE NUMBER OF COMPONENTS                     *
C          LSTCL  - THE NUMBER OF CLASSES                        *
C          NPASS  - THE STATE ARRAY REFERENCE ROW                *
C          IBDM   - CLASS/COMPONENT BLOCK SIZE                   *
C          DT     - TIME SIMULATION INCREMENT                    *
C****************************************************************
      COMMON/R4/STATE(20,30)
      COMMON/R6/WEIBL(3,30)
C
      N2 = NPASS + 1
      N3 = NPASS + 2
      N4 = NPASS + 3
C////////////////////////////////////////////////////////////////
C LOOP THROUGH CLASSES AND COMPONENTS.
C////////////////////////////////////////////////////////////////
      DO 100 ICL = ISTCL,LSTCL
         DO 50 ICP = ISTCP,LSTCP
            CALL RCOLUM(ICP,LSTCL,ICL,LSTCP,IRP,IRCOL,IBDM)
C////////////////////////////////////////////////////////////////
C CONVERT CHANGE IN BAD TO CHANGE IN QUALITY.
C////////////////////////////////////////////////////////////////
            QNOW   = STATE(N3,IRCOL)
            POP    = STATE(NPASS,IRCOL)
            DBAD   = STATE(N4,IRCOL)
            DPOP   = STATE(N2,IRCOL)
            IF(POP .EQ. 0.0) GO TO 50
            DQ     = DBAD/POP - QNOW*DPOP/POP
C////////////////////////////////////////////////////////////////
C CALCULATE WEIBULL EFFECT ON QUALITY.
C////////////////////////////////////////////////////////////////
            ANOW   = WEIBL(1,IRCOL)
            WBSLP  = WEIBL(2,IRCOL)
            CHLF   = WEIBL(3,IRCOL)
            IF(CHLF .EQ. 0.0) CHLF = 1.0D-10
            HZRD   = (WBSLP/CHLF) * (ANOW/CHLF)**(WBSLP-1.0)
            DQ     = DQ + HZRD*(1.0 - QNOW)
            STATE(N4,IRCOL) = DQ
```

```
50       CONTINUE
100   CONTINUE
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000257    LINKAGE - 000111    STACK - 000046
0000 ERRORS [<WEIBUL>FTN-REV16.6]
```

```
C******************************************************************
C******************************************************************
C******************************************************************
      SUBROUTINE DATAIN(ISTND,LSTND,ISTPH,LSTPH,ISTCP,LSTCP,
     *                  ISTCL,LSTCL,LSTFL,ISTCT,LSTCT,IIDM,IRDM,
     *                  IBDM,NDOUT,IUNIT,IDCODE,NREFP,PLACE,
     *                  RUNN)
      IMPLICIT REAL*8(A-H,O-Z)
C******************************************************************
C  DATAIN READS DATA INTO ARRAYS FROM DATA FILES THAT HAVE BEEN*
C OPENNED FOR READING PRIOR TO RUNNING THE PROGRAM. IT THEN IN-*
C ITIALIZE SEVERAL CONTROL VARIABLES WITH VALUES THAT HAVE BEEN*
C READ INTO THE IFILE ARRAY.                                   *
C                                                              *
C                     ARGUMENT   DESCRIPTION                   *
C                (SEE RFILE AND IFILE FOR DETAIL)              *
C                                                              *
C                ISTND  - FIRST NODE                           *
C                LSTND  - LAST NODE                            *
C                ISTPH  - FIRST PATH                           *
C                LSTPH  - LAST PATH                            *
C                ISTCP  - FIRST COMPONENT                      *
C                LSTCP  - LAST COMPONENT                       *
C                ISTCL  - FIRST CLASS                          *
C                LSTCL  - LAST CLASS                           *
C                ISTCT  - FIRST COST                           *
C                LSTCT  - LAST COST                            *
C                IIDM   - DIMENSION OF PATH INTEGER NUMBERS    *
C                IRDM   - DIMENSION OF PATH REAL NUMBERS       *
C                IBDM   - DIMENSION OF THE COST-COMPONENT BLOCK*
C                IUNIT  - FORTRAN UNIT DEVICE NUMBER           *
C                IDCODE -                                      *
C******************************************************************
      COMMON/I1/IERROR(10)
      COMMON/I2/IFILE(100)
      COMMON/I3/IFLAG(10)
      COMMON/I7/INODE(6,300)
C
      COMMON/R1/ACCUM(750)
      COMMON/R2/FUNCT(10,10)
      COMMON/R3/RNODE(30,300)
      COMMON/R4/STATE(20,30)
      COMMON/R5/RFILE(100)
      COMMON/R6/WEIBL(3,30)
      COMMON/R7/REVISN(8),SCHMDG(8),DATAFL(8)
      DIMENSION RUNN(10), PLACE(10)
C//////////////////////////////////////////////////////////////////
C READ IN DATA FROM FILES.
C//////////////////////////////////////////////////////////////////
      READ(8,199) (REVISN(J),J=1,7)
```

```
      READ(8,199) RUNN
      READ(8,199) PLACE
      READ(8,200) (IFILE(J),J=1,100)
      READ(9,300) ((WEIBL(M,I),M=1,3),I=1,30)
      READ(12,199) (SCHMDG(I),I=1,8)
      READ(12,199)(DATAFL(I),I=1,8)
      READ(12,500) ((INODE(I,M),I=1,6),M=1,220)
      READ(13,600) ((RNODE(I,M),I=1,30),M=1,220)
      READ(14,700) ((FUNCT(M,I),I=1,10),M=1,10)
      READ(17,840)(RFILE(J),J=1,100)
      READ(16,900) (IFLAG(I),I=1,10)
C///////////////////////////////////////////////////////////////////////////
C FORMATS.
C///////////////////////////////////////////////////////////////////////////
199   FORMAT(10A8)
200   FORMAT(100(I3,/))
300   FORMAT(30(3D10.2,/))
400   FORMAT(2(5I3,/))
500   FORMAT(300(6(I3,1X)/))
600   FORMAT(900(5D10.4,/))
700   FORMAT(10(10D5.4,/))
800   FORMAT(120(5D10.4,/))
840   FORMAT(100(D5.4,/))
900   FORMAT(10(I3,/))
C///////////////////////////////////////////////////////////////////////////
C READ IN CONTROL VARIABLE VALUES.
C///////////////////////////////////////////////////////////////////////////
      ISTND  = IFILE(1)
      LSTND  = IFILE(2)
      ISTPH  = IFILE(3)
      LSTPH  = IFILE(4)
      ISTCP  = IFILE(5)
      LSTCP  = IFILE(6)
      ISTCL  = IFILE(7)
      LSTCL  = IFILE(8)
      ISTCT  = IFILE(9)
      LSTCT  = IFILE(10)
      IIDM   = IFILE(11)
      IRDM   = IFILE(12)
      NREFP  = IFILE(13)
      IBDM   = IFILE(16)
      LSTFL  = IFILE(14)
      IUNIT  = IFILE(21)
      IDCODE = IFILE(40)
C
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000760    LINKAGE - 000142    STACK - 000110
0000 ERRORS [<DATAIN>FTN-REV16.6]
```

```
C**************************************************************
C**************************************************************
C**************************************************************
      SUBROUTINE ERRORS(IER)
C
      COMMON/I1/IERROR(10)
      COMMON/I2/IFILE(100)
      COMMON/I3/IFLAG(10)
C
      ITUNIT = IFILE(23)
100   FORMAT(17HERROR LOCATED AT ,10I5/)
      N = 0
200   DO 400  I = 1,10
          IF(N .EQ. 1)GO TO 400
          IF(IERROR(I) .EQ. 0)GO TO 300
          GO TO 400
300       IERROR(I) = IER
          WRITE(ITUNIT,100) IERROR(I)
          N = 1
400   CONTINUE
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000102    LINKAGE - 000037    STACK - 000020
0000 ERRORS [<ERRORS>FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE TRACE(NFMT,I1,I2,I3,I4,I5,
     *                 R1,R2,R3,R4,R5,R6,R7,R8,R9,R10)
C
C     PRINTS SELECTABLE SETS OF DIAGNOSTICS DURING PROGRAM EXECUTION
C     IFLAG() ARRAY VALUES ARE CONTROL SELECTORS
C
C          (2) = 1 MEANS CRT OUTPUT
C          (3) = AVAILABLE
C          (4) = PATH NUMBER NOW ON IN THIS PASS
C          (6) = AVAILABLE
C          (7) = AVAILABLE
C          (8) = AVAILABLE
C          (9) = NUMBER OF LINES PRINTED UP TO THIS PASS
C         (10) = MAXIMUM ALLOWED LINES OF PRINTOUT
C
      IMPLICIT REAL*8(A-H,O-Z)
C
      COMMON/I2/IFILE(100)
      COMMON/I3/IFLAG(10)
C
      IFLAG(9) = IFLAG(9) + 1
      IF((IFLAG(10).NE.IFLAG(9)).OR.(IFLAG(4).NE.IFLAG(5)))GO TO 50
      IFLAG(1) = 0
      GO TO 999
C
50    ITUNIT = IFILE(23)
      WRITE(ITUNIT,75)NFMT
      WRITE(ITUNIT,1010)
      WRITE(ITUNIT,100)I1,I2,I3,I4,I5
      WRITE(ITUNIT,125)R1,R2,R3,R4,R5
      WRITE(ITUNIT,150)R6,R7,R8,R9,R10
75    FORMAT(19H TRACE NUMBER IS - ,I4,//)
100   FORMAT(5I3,/)
125   FORMAT(5(1PD10.3)/)
150   FORMAT(5(1PD10.3)///)
C
1010  FORMAT(51H SUB BRANCH - SIMPLE SPLIT DECISION VALUES            ,//)
1030  FORMAT(53H SUB BRANCH - DEFECTIVE BASED DECISION VALUES         ,//)
1050  FORMAT(53H SUB BRANCH - AFTER QUALITY RESET CHECK               ,//)
1060  FORMAT(53H SUB SUMS   - VALUES OF GOOD AND BAD QUALITY          ,//)
1114  FORMAT(53H SUB QDOT   - VALUES OF QUALITY RATES OF CHANGE       ,//)
1122  FORMAT(53H SUB PDOT   - VALUES OF STATE MATRIX                  ,//)
1540  FORMAT(53H SUB FUNCTS - VALUES OF FUNCTS MATRIX                 ,//)
1560  FORMAT(53H SUB ACCUME - VALUES OF 1ST TEN PATH ITEMS            ,//)
1580  FORMAT(53H SUB ACCUME - VALUES OF TEN ACCUMS BEFORE SUMMING     ,//)
1590  FORMAT(53H SUB ACCUME - VALUES OF TEN ACCUMS AFTER SUMMING      ,//)
C
```

```
(0878)  999   RETURN
(0879)        END
PROGRAM SIZE:   PROCEDURE - 000777    LINKAGE - 000047    STACK - 000074
0000 ERRORS [<TRACE >FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE PATHIN(IFILE,IPATH,IIDM,RPATH,IRDM)
       IMPLICIT REAL*8(A-H,O-Z)
      COMMON/I8/IARAY(15,600)
      COMMON/R8/RARAY(750,600)
      DIMENSION IPATH(15), RPATH(750)
      DO 100 I = 1,IIDM
         IPATH(I) = IARAY(I,IFILE)
100   CONTINUE
      DO 200 I = 1,IRDM
         RPATH(I) = RARAY(I,IFILE)
200   CONTINUE
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000073    LINKAGE - 000027    STACK - 000034
0000 ERRORS [<PATHIN>FTN-REV16.6]
```

```
C****************************************************************
C****************************************************************
C****************************************************************
      SUBROUTINE PATOUT(IFILE,IPATH,IIDM,RPATH,IRDM)
       IMPLICIT REAL*8(A-H,O-Z)
      COMMON/I8/IARAY(15,600)
      COMMON/R8/RARAY(750,600)
      DIMENSION IPATH(15), RPATH(750)
      DO 100 I = 1,IIDM
         IARAY(I,IFILE) = IPATH(I)
100   CONTINUE
      DO 200 I = 1,IRDM
         RARAY(I,IFILE) = RPATH(I)
200   CONTINUE
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000075    LINKAGE - 000027    STACK - 000034
0000 ERRORS [<PATOUT>FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE BRANCH(J,NODI,CD,I,RPATH1,RPATH2,RPATH3,IRDM,
     *                  IBDM,NPASS,NRPTH,IPH)
      IMPLICIT REAL*8(A-H,O-Z)
C**********************************************************************
C BRANCH WILL DETERMINE THE OUT GOING FLOWS AND QUALITIES ON A BRANCH*
C NODE. A BRANCH NODE WILL INVOLVE A SIMPLE OR A DEFECTIVE DECISION. *
C DEPENDING UPON THE TYPE OF DECISION THE CALCULATIONS WILL BE WORKED*
C BY ONE OF TWO BLOCKS. EACH BLOCK REQUIRES A NODE SPLIT FRACTION, TO*
C DETERMINE THE REFERENCE PATH FLOW. THE FRACTION IS SENT FROM NORMAL*
C AND THEN CDFUNC IS CALLED TO ALTER THE FRACTION AS NECESSARY. AFTER*
C FLOWS AND QUALITIES ARE COMPUTED, A REWORK CHECK IS MADE. IF THIS  *
C IS THE CASE, QUALITY IS SET TO ZERO ON THE APPROPRIATE PATHS.      *
C                                                                    *
C                         ARGUMENT LIST                              *
C     (SEE IFILE AND RFILE FOR COMPLETE DESCRIPTIONS)                *
C                                                                    *
C     1         J      = INDICATOR OF SIMPLE OR DEFECTIVE DECISION   *
C     2         NODI   = REFERENCE PATH INDICATOR                    *
C     3         CD     = NODE SPLIT FRACTION                         *
C     4         I      = REFERENCE COLUMN INDEX OF FLOW              *
C     5         RPATH1= NODE INCOMING PATH NUMBERS                   *
C     5         RPATH2= NODE OUTGOING PATH NUMBERS                   *
C     6         RPATH3= NODE OUTGOING PATH NUMBERS                   *
C     7         IRDM   = DIMENSION OF PATH REAL NUMBERS VECTOR       *
C**********************************************************************
      COMMON/I2/IFILE(100)
      COMMON/I3/IFLAG(10)
C
      COMMON/R5/RFILE(100)
C
      DIMENSION RPATH1(750),RPATH2(750),RPATH3(750)
C/////////////////////////////////////////////////////////////////////
C INITIALIZE CONTROL INDICES AND VARIABLES.
C/////////////////////////////////////////////////////////////////////
      K = I + 1
      I3 = I + 2
      L = I + 4
      ADDRSS = 0.0
      CDTEMP = CD
C////////////////////////////////////////////////////////////////////
C COMPUTE PROPER NODE SPLIT FRACTION AND GO TO APPROPRIATE BLOCK.
C////////////////////////////////////////////////////////////////////
      IF(NODI .EQ. 0) ADDRSS = RPATH2(I3)
      IF(NODI .EQ. 1) ADDRSS = RPATH3(I3)
      IF((ADDRSS .LT. 10.0**-47) .AND. (ADDRSS .GE. 10.0**-50))
     *CALL CDFUNC(I,RPATH1,IRDM,CD,IBDM,NPASS,NRPTH,IPH,ADDRSS)
150   IF(J .GT. 1)CALL ERRORS(1005)
```

```
      IF(J .EQ. 1)GO TO 200
C/////////////////////////////////////////////////////////////////////
C SIMPLE SPLIT DECISION.
C/////////////////////////////////////////////////////////////////////
      IF(NODI .EQ. 1)CD = 1.00 - CD
      RPATH2(I)=RPATH1(I)*CD
      RPATH3(I)=RPATH1(I)-RPATH2(I)
      IF(RPATH1(3) .LT. 0.0)CALL ERRORS(1010)
      RPATH2(K)=RPATH1(K)
      RPATH3(K)=RPATH1(K)
      IF(IFLAG(1) .EQ. IFLAG(4))CALL TRACE(1010,
     *        I,             J,          K,         L,        IRDM,
     *          CD,RPATH1(I), RPATH1(K), RPATH1(L),RPATH2(I),
     *RPATH2(K),RPATH2(L), RPATH3(I), RPATH3(K),RPATH3(L))
      GO TO 300
C/////////////////////////////////////////////////////////////////////
C DEFECTIVE BASED BRANCH DECISION.
C/////////////////////////////////////////////////////////////////////
200   BAD1  = RPATH1(I) * RPATH1(K)
      GOOD1 = RPATH1(I) - BAD1
      BAD2  = BAD1  * CD
      GOOD2 = 0.
      ALL2 = BAD2 + GOOD2
      BAD3  = BAD1  - BAD2
      GOOD3 = GOOD1 + GOOD2
      ALL3 = BAD3 + GOOD3
      IF(ALL2 .EQ. 0.)GO TO 235
      IF(ALL2 .LE. 0.0)CALL ERRORS(1020)
      QUAL2 = BAD2/(BAD2 + GOOD2)
      GO TO 240
235   QUAL2 = 0
240   IF(ALL3 .EQ. 0)GO TO 245
      IF(ALL3 .LE. 0.0)CALL ERRORS(1030)
      QUAL3 = BAD3/(BAD3 + GOOD3)
      GO TO 250
245   QUAL3 = 0
250   IF(NODI .EQ. 1)GO TO  260
      RPATH2(I) = ALL2
      RPATH3(I) = ALL3
      RPATH2(K) = QUAL2
      RPATH3(K) = QUAL3
      GO TO 275
260   RPATH2(I) = ALL3
      RPATH3(I) = ALL2
      RPATH2(K) = QUAL3
      RPATH3(K) = QUAL2
275   IF(IFLAG(1) .EQ. IFLAG(4))CALL TRACE(1030,
     *   I,        J,         K,         L,       IRDM,
     *  CD,     BAD1,     GOOD1,     BAD2,     GOOD2,
     *BAD3,    GOOD3,    QUAL2,    QUAL3,         0.0)
```

```
C/////////////////////////////////////////////////////////////////////////
C RESET QUALITY TO ZERO IF REWORK INDICATOR = 1
C/////////////////////////////////////////////////////////////////////////
300   IF(RPATH2(L) .GE. 1)RPATH2(K)=0.0
      IF(RPATH2(L) .LT. 0)CALL ERRORS(1040)
      IF(RPATH3(L) .GE. 1)RPATH3(K)=0.0
      IF(RPATH3(L) .LT. 0)CALL ERRORS(1050)
      IF(IFLAG(1) .EQ. IFLAG(4))CALL TRACE(1050,
     *         I,         J,         K,         L,      IRDM,
     *        CD,RPATH1(I), RPATH1(K), RPATH1(L),RPATH2(I),
     *RPATH2(K),RPATH2(L), RPATH3(I), RPATH3(K),RPATH3(L))
      CD = CDTEMP
999   RETURN
      END
PROGRAM SIZE:   PROCEDURE - 001115    LINKAGE - 000133    STACK - 000110
0000 ERRORS [<BRANCH>FTN-REV16.6]
```

```
C****************************************************************
C****************************************************************
C****************************************************************
      SUBROUTINE SUMS (IMFG1,IMFG2,I,F1,F2,F3,IRDM)
C
      IMPLICIT REAL*8(A-H,O-Z)
C*****************************************************************
C SUMS WILL TAKE THE FLOWS AND QUALITIES OF THE IN COMING PATHS OF*
C A SUM NODE AND COMBINE THEN TO PRODUCE THE FLOW AND QUALITY OF  *
C OUT GOING NODE.                                                 *
C                                                                 *
C                     ARGUMENT LIST                               *
C                                                                 *
C         I       - RPATH REFERENCE COLUMN                        *
C         IMG1    - INDICATOR THAT RPATH1 IS THE MANUFACTURING PATH *
C         IMG2    - INDICATOR THAT RPATH2 IS THE MANUFACTURING PATH *
C         F1      - RPATH1, INCOMING PATH REAL DATA               *
C         F2      - RPATH2, INCOMING PATH REAL DATA               *
C         F3      - RPATH3, OUT GOING PATH REAL DATA              *
C         IRDM    - RPATH DIMENSION                               *
C*****************************************************************
      COMMON/I3/IFLAG(10)
      DIMENSION F1(750),F2(750),F3(750)
C///////////////////////////////////////////////////////////////////
C SET CONTROL INDICES.
C///////////////////////////////////////////////////////////////////
      K = I + 1
      L = I + 2
C///////////////////////////////////////////////////////////////////
C IF AN INCOMING PATH IS A MANUFACTURING PATH SET IT'S FLOW EQUAL TO
C THAT OF THE OTHER INCOMING PATH.
C///////////////////////////////////////////////////////////////////
      IF(IMFG1 .EQ. 1)F1(I) = F2(I)
      IF(IMFG2 .EQ. 1)F2(I) = F1(I)
C///////////////////////////////////////////////////////////////////
C CALCULATE THE COMBINED FLOWS AND QUALITIES OF F1 AND F2.
C///////////////////////////////////////////////////////////////////
      BAD1  = F1(I) * F1(K)
      GOOD1 = F1(I) - BAD1
      BAD2  = F2(I) * F2(K)
      GOOD2 = F2(I) - BAD2
      BAD3  = BAD1  + BAD2
      GOOD3 = GOOD1 + GOOD2
      FLOW3 = BAD3 + GOOD3
C
      IF(FLOW3 .LT. 0.0)CALL ERRORS(1060)
C
      IF(FLOW3 .EQ. 0.0)GO TO 275
      QUAL3 = BAD3 / (BAD3 + GOOD3)
      GO TO 250
```

```
275    QUAL3 = 0.0
C/////////////////////////////////////////////////////////////////////////
C ASSIGN COMBINED FLOW AND QUALITY TO OUT GOING PATH.
C/////////////////////////////////////////////////////////////////////////
250    F3(I) = FLOW3
       F3(K) = QUAL3
       IF(IFLAG(1) .EQ. IFLAG(4))CALL TRACE(1060,
      *    I,          0,       K,          L,       IRDM,
      * 0.0,       BAD1,   GOOD1,      BAD2,      GOOD2,
      *BAD3,      GOOD3,     0.0,     QUAL3,        0.0)
C
       RETURN
       END
PROGRAM SIZE:   PROCEDURE - 000257    LINKAGE - 000075    STACK - 000046
0000 ERRORS [<SUMS  >FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE PDOT(I,IRCOL,RPATH,IRDM,DT,ISGN,NPASS)
      IMPLICIT REAL*8(A-H,O-Z)
C*********************************************************************
C PDOT IS CALLED FOR PATHS THAT LEAVE OR ENTER THE SYSTEM AND RE-   *
C SULT IN SYSTEM POPULATION CHANGES. THERE IS ALSO A SYSTEM QUALI-  *
C TY CHANGE WHICH IS CAUSED BY THE REMOVAL OF DEFECTIVE COMPONENT   *
C AND THE ADDITION OF GOOD. PDOT KEEPS TRACK OF THESE CHANGES AND   *
C STORES THEM IN THE STATE ARRAY FOR LATER USE.                     *
C                                                                   *
C                      ARGUMENT LIST                                *
C      (SEE IFILE AND RFILE FOR COMPLETE DESCRIPTIONS)              *
C                                                                   *
C      1         I     = REFERENCE COLUMN INDEX OF REAL VALUES      *
C      2         IRCOL = COLUMN INDEX OF REAL MODE REFERENCE POSITION*
C      3         RPATH = VECTOR CONTAINING PATH VALUES              *
C      4         IRDM  = DIMENSION OF PATH REAL NUMBERS             *
C      5         DT    = TIME STEP                                  *
C      6         ISGN  = CHANGE SIGN                                *
C      7         NPASS = STATE ARRAY REFERENCE ROW                  *
C*********************************************************************
      COMMON/I2/IFILE(100)
      COMMON/I3/IFLAG(10)
C
      COMMON/R4/STATE(20,30)
      COMMON/R5/RFILE(100)
C
      DIMENSION RPATH(750)
C/////////////////////////////////////////////////////////////////////
C SET CONTRL INDICES.
C/////////////////////////////////////////////////////////////////////
      J = I + 1
      N2 = NPASS + 1
      N3 = NPASS + 3
C///////////////////////////////////////////////////////////////////
C CALCULATE TOTAL AND BAD POPULATION CHANGES.
C///////////////////////////////////////////////////////////////////
      DP = RPATH(I) * DBLE(FLOAT(ISGN))
      DB = DP * RPATH(J)
      STATE(N2,IRCOL) = STATE(N2,IRCOL) + DP
      STATE(N3,IRCOL) = STATE(N3,IRCOL) + DB
      IF(IFLAG(1) .EQ. IFLAG(4))CALL TRACE(1122,
     *          I,       J,      IRCOL,      IROW,      ICOL,
     *         DB, RPATH(I), RPATH(J),        DG,        DP,
     *         DT,      STATE(2,IRCOL),     0.0D 00,  0.0D 00,  0.0D 00)
C
      RETURN
      END
```

```
PROGRAM SIZE:    PROCEDURE - 000175     LINKAGE - 000061     STACK - 000050
0000 ERRORS [<PDOT  >FTN-REV16.6]
```

```
C*************************************************************
C*************************************************************
C*************************************************************
      SUBROUTINE QDOT(I,IRCOL,RPATH,IRDM,DT,ISGN,NPASS)
      IMPLICIT REAL*8(A-H,O-Z)
C****************************************************************
C QDOT WILL CALCULATE THE CHANGE IN THE SYSTEM DEFECTIVE POPULATION*
C FOR PATHS THAT ENTAIL A CHANGE IN QUALITY. THE VALUE OBTAINED IS *
C THEN USED TO UPDATE THE CHANGE IN BAD IN THE STATE ARRAY FOR LA- *
C USE.                                                             *
C                                                                  *
C                         ARGUMENT LIST                            *
C       (SEE IFILE AND RFILE FOR COMPLETE DESCRIPTIONS)            *
C                                                                  *
C     1         I      = REFERENCE COLUMN INDEX OF REAL VALUES     *
C     2         IRCOL  = COLUMN INDEX OF REAL MODE REFERENCE POSITION*
C     3         RPATH  = VECTOR CONTAINING PATH VALUES             *
C     4         IRDM   = DIMENSION OF PATH REAL NUMBERS            *
C     5         DT     = TIME STEP                                 *
C     6         ISGN   = CHANGE SIGN                               *
C     7         NPASS  = STATE ARRAY REFERENCE ROW                 *
C****************************************************************
      COMMON/I2/IFILE(100)
      COMMON/I3/IFLAG(10)
C
      COMMON/R4/STATE(20,30)
      COMMON/R5/RFILE(100)
C
      DIMENSION RPATH(750)
C///////////////////////////////////////////////////////////////////
C SET CONTRL INDICES.
C///////////////////////////////////////////////////////////////////
      J = I + 1
      N2 = NPASS + 1
      N3 = NPASS + 3
C/////////////////////////////////////////////////////////////////
C CALCULATE BAD POPULATION CHANGES.
C/////////////////////////////////////////////////////////////////
      DB = RPATH(I)*RPATH(J)
      RPATH(J) = 0.00
      N3 = NPASS + 3
      STATE(N3,IRCOL) = STATE(N3,IRCOL) - DB
C
      IF(IFLAG(1) .EQ. IFLAG(4))CALL TRACE(1114,
     *         I,          J,     IRCOL,       IROW,      ICOL,
     *        BG,        DBG,         B,   RPATH(I),  RPATH(J),
     *        DB,         DT,      DQDT,    0.0D 00,    0.0D 00)
C
      RETURN
      END
```

```
PROGRAM SIZE:   PROCEDURE - 000145    LINKAGE - 000065    STACK - 000044
0000 ERRORS [<QDOT  >FTN-REV16.6]
```

```
C*************************************************************
C*************************************************************
C*************************************************************
      SUBROUTINE RCOLUM (ICL,LSTCL,ICP,LSTCP,IRP,IRCOL,IBDM)
      IMPLICIT REAL*8(A-H,O-Z)
C*************************************************************
C SETS COLUMN(IRP) INDEX VALUE FOR REAL PATH VECTORS         *
C                                                            *
C  ARGUMENT        DESCRIPTION                               *
C       1          ICL   = PRESENT CLASS INDEX               *
C       2          LSTCL = LAST CLASS INDEX                  *
C       3          ICP   = PRESENT COMPONENT INDEX           *
C       4          LSTCP = LAST COMPONENT INDEX              *
C       5          IRP   = REFERENCE COLUMN OF RPATH         *
C       6          IRCOL = REFERENCE COLUMN OF RNODE AND STATE *
C       7          IBDM  = DIMENSION OF A SINGLE COST-COMPONENT *
C                          BLOCK                             *
C*************************************************************
      IRP = IBDM*(LSTCP*(ICL-1)+(ICP-1))+1
      IRCOL = (LSTCP*(ICL-1)+(ICP-1))+1
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000030    LINKAGE - 000020    STACK - 000040
0000 ERRORS [<RCOLUM>FTN-REV16.6]
```

```
C****************************************************************
C****************************************************************
C****************************************************************
      SUBROUTINE PCOSTS(I,J,K,PATH,PCOST,RPATH,IRDM)
      IMPLICIT REAL*8(A-H,O-Z)
C****************************************************************
C THIS SUBROUTINE WILL COMPUTE PATH COSTS BY COMPONENT AND SAVE*
C THEM IN THE RPATH ARRAY.                                      *
C               ARGUMENT DESCRIPTION                            *
C     1         I    = REFERENCE COLUMN INDEX                   *
C     2         J    = COLUMN INDEX VALUE OF ISTCT              *
C     3         K    = COLUMN INDEX VALUE OF LSTCT              *
C     4         PATH= VECTOR CONTAINING PATH REAL NUMBER VALUES *
C     5         IRDM = DIMENSION OF PATH REAL NUMBERS VECTOR    *
C     6         RPATH= VECTOR CONTAINING PRESENT COSTS          *
C     7         IRDM = DIMENSION OF COST RATES VECTOR           *
C****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION RPATH(750),PATH(750)
C
      L   = I+J
      I1  = I+1
      I3  = I+5
      K1  = L+K
      USEQ = PATH(I3)
      FLOW= PATH(I)
      QUAL= PATH(I1)
C
      DO 100 M = L,K1
          COST = FLOW*QUAL*PATH(M)*USEQ
     *            +FLOW*PATH(M)*(1.0 - USEQ)
          RPATH(M)=COST
          PCOST = PCOST+COST
100   CONTINUE
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000127    LINKAGE - 000047    STACK - 000044
0000 ERRORS [<PCOSTS>FTN-REV16.6]
```

```
C****************************************************************
C****************************************************************
C****************************************************************
      SUBROUTINE ACCUME(I,J,LSTCT,RPATH,IRDM)
C*******************************************************************
C ACCUM SUMS UP THE COSTS OF ALL THE COMPONENTS FOR EACH COST CODE  *
C AND STORES THE COMPUTED VALUES IN THE ACCUM ARRAY FOR LATER USE.  *
C                                                                   *
C                       ARGUMENT DESCRIPTION                        *
C              (SEE IFILE AND RFILE FOR COMPLETE DESCRIPTION)       *
C      1              I = REFERENCE COLUMN INDEX VALUE              *
C      2              J = COLUMN INDEX VALUE OF ISTCT               *
C      3              LSTCT = COLUMN INDEX VALUE OF LSTCT           *
C      4              RPATH = VECTOR CONTAINING PRESENT PATH COSTS  *
C      5              IRDM  = DIMENSION OF PATH REAL NUMBERS VECTOR *
C*******************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      COMMON/R1/ACCUM(750)
      COMMON/I3/IFLAG(10)
      DIMENSION RPATH(750)
C/////////////////////////////////////////////////////////////////////
C SET CONTROL INDICES.
C/////////////////////////////////////////////////////////////////////
      K = I+J
      L = K+LSTCT
      IF(IFLAG(1) .EQ. IFLAG(4))CALL TRACE(1560,
     *        I,        J,        K,        L,      IRDM,
     * ACCUM(1),ACCUM(2),ACCUM(3),ACCUM(4), ACCUM(5),
     * ACCUM(6),ACCUM(7),ACCUM(8),ACCUM(9),ACCUM(10))
C////////////////////////////////////////////////////////////////////
C BEGIN SUMMATION LOOP.
C////////////////////////////////////////////////////////////////////
      DO 100  M=K,L
         ACCUM(M) = ACCUM(M) + RPATH(M)
100   CONTINUE
C
C
C
      IF(IFLAG(1) .EQ. IFLAG(4))CALL TRACE(1580,
     *        I,        J,        K,        L,      IRDM,
     *RPATH(1),RPATH(2),RPATH(3),RPATH(4), RPATH(5),
     *RPATH(6),RPATH(7),RPATH(8),RPATH(9),RPATH(10))
C
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000232    LINKAGE - 000073    STACK - 000054
0000 ERRORS [<ACCUME>FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE FUNCTS(IRP,ISTCT,LSTCT,RPATH,IRDM,IP,RFC,IBDM,
     *                  NPASS,IUNIT,IVAL,ICP,ICL)
      IMPLICIT REAL*8(A-H,O-Z)
C*****************************************************************
C FUNCTS WILL FETCH INFORMATION STORED IN THE FUNCT ARRAY BY US-*
C ING ADDRESSES STORED IN RPATH. THIS INFORMATION IS USED TO IN-*
C DICATE WHICH COST FUNCTIONS ARE NEEDED TO CALCULATE THE VALUES*
C FOR THE COST CODE IN THE CURRENT PATH. THE VALUES ARE THEN PUT*
C IN THE CORRESPONDING RPATH COST COLUMNS. THE ADDRESSES IN THE *
C COST COLUMNS ARE STORED BY  ADDRCH BEFORE BEING OVER WRITTEN  *
C                                                               *
C                      ARGUMENT LIST                            *
C              (SEE RFILE AND IFILE FOR DE-                     *
C               SCRIPTIONS)                                     *
C                                                               *
C      1          IRP     = REFERENCE COLUMN INDEX IN RPATH     *
C      2          ISTCT  = COLUMN  OF  FIRST COST ITEM IN RPATH *
C      3          LSTCT  = LAST COST INDEX COLUMN VALUE         *
C      4          RPATH  = PATH VALUES                          *
C      5          IRDM   = DIMENSION OF RPATH VECTOR            *
C      6          IP     = PATH NUMBER PRESENTLY COMPUTING ON   *
C      7          RFC    = ARRAY FOR STORING RPATH FUNCT ADDRESSES *
C      8          IBDM   = CLASS/COMPONENT BLOCK SIZE           *
C      9          NPASS  = STATE REFERENCE ROW                  *
C     10          IUNIT  = OUTPUT DEVICE                        *
C     11          IVAL   = OUTPUT INDICATOR                     *
C     12          ICP    = PRESENT COMPONENT                    *
C     13          ICL    = PRESENT CLASS OF OUTPUT                *
C*****************************************************************
C
      COMMON/I2/IFILE(100)
      COMMON/I3/IFLAG(10)
C
      COMMON/R2/FUNCT(10,10)
      COMMON/R4/STATE(20,30)
      COMMON/R5/RFILE(100)
C
      DIMENSION RPATH(750),RFC(750)
C
C///////////////////////////////////////////////////////////////////////
C SET CONTRL INDICES.
C///////////////////////////////////////////////////////////////////////
      I2 = IRP + ISTCT
      I3 = I2  + LSTCT
      N2 = NPASS + 2
C/////////////////////////////////////////////////////////////////////////
C STORE FUNCTION ADDRESSES TO BE USED.
```

```
C/////////////////////////////////////////////////////////////////////
      CALL ADDRCH(RPATH,RFC,IRP,ISTCT,LSTCT,IRDM)
C////////////////////////////////////////////////////////////////////
C LOOP THROUGH RPATH COST COLUMS FOR FUNCT ADDRESSES.
C////////////////////////////////////////////////////////////////////
      DO 5000  K=I2,I3
          IF((RPATH(K) .GT. 10.0**-47) .OR.
     *    (RPATH(K) .LT. 10.0**-50))GO TO 5000
C////////////////////////////////////////////////////////////////////
C SET FUNCTION VARIABLES AND CONSTANTS.
C////////////////////////////////////////////////////////////////////
          FCOST = 1.
          IRW   = IDINT(RPATH(K)*(10.0**50)+0.1)
200       ILINE = IDINT(FUNCT(IRW,2)+0.1)
          AO    = FUNCT(IRW,5)
          IA    = IDINT(AO+0.1)
          X     = RFILE(IA)
          A     = FUNCT(IRW,6)
          B     = FUNCT(IRW,7)
          C     = FUNCT(IRW,8)
          XO    = FUNCT(IRW,9)
          YO    = FUNCT(IRW,10)
C////////////////////////////////////////////////////////////////////
C GO TO APPROPRIATE FUNCTION FORMULA.
C//////////////////////////////////////////////////////////////////////
          GO TO (1010,1020,1030,1040,1050,1060,1070,1080,1090),ILINE
C////////////////////////////////////////////////////////////////////////
C THIS PROGRAM BLOCK RESERVED FOR USER DEFINED EQUATIONS.
C///////////////////////////////////////////////////////////////////////
C////////////////////////////////////////////////////////////////////////
C FUNCTION 1:
C////////////////////////////////////////////////////////////////////////
1010      COST = A * X + B
          GO TO 2000
C////////////////////////////////////////////////////////////////////////
C FUNCTION 2:
C////////////////////////////////////////////////////////////////////////
1020      COST = A * ((X - XO)**B) + YO
          GO TO 2000
C/////////////////////////////////////////////////////////////////////////
C FUNCTION 3:
C/////////////////////////////////////////////////////////////////////////
1030      COST = A * (DEXP(B * (X - XO))) + YO
          GO TO 2000
C/////////////////////////////////////////////////////////////////////////
C FUNCTION 4:
C/////////////////////////////////////////////////////////////////////////
1040      COST = (B * (X**(B - 1)) * (DEXP(-(X/A)**B)))/(A**B)
          GO TO 2000
C/////////////////////////////////////////////////////////////////
```

```
C FUNCTION 5:
C////////////////////////////////////////////////////////////////////
1050       Z1 = (DEXP(X-B)-DEXP(B-X))/(DEXP(X-B)+DEXP(B-X))
           Z2 = (DEXP(X-X0)-DEXP(X0-X))/(DEXP(X-X0)+DEXP(X0-X))
           Z3 = (DEXP(-B) - DEXP(B))/(DEXP(-B) + DEXP(B))
           Z4 = (DEXP(-X0) - DEXP(X0))/(DEXP(-X0) + DEXP(X0))
           COST = (A*(1+Z1) + C*(1-Z2) + Y0)
           COST = COST/(A*(1+Z3) + C*(1-Z4) + Y0)
           GO TO 2000
C////////////////////////////////////////////////////////////////////
C FUNCTION 6:
C////////////////////////////////////////////////////////////////////
1060       IRC = (IRP + IBDM -1)/IBDM
           QP = STATE(N2,IRC)
           COST = A * QP/B + C
           GO TO 2000
C////////////////////////////////////////////////////////////////////
C FUNCTION 7:
C////////////////////////////////////////////////////////////////////
1070       F1I = RPATH(IRP)
           COST = A * F1I/(F1I + A * B)
           GO TO 2000
C/////////////////////////////////////////////////////////////////////////////
C REMAINING AVAILBLE FOR OTHER FUNCTIONS.
C/////////////////////////////////////////////////////////////////////////////
1080        COST = E*8.0
            GO TO 2000
C
1090        COST = F*9.0
            GO TO 2000
C/////////////////////////////////////////////////////////////////////////////
C TAKE PRODUCT OF ALL COST FUNCTIONS ASSOCIATED WITH CURRENT COST CODE.
C/////////////////////////////////////////////////////////////////////////////
C
C
2000        FCOST = COST*FCOST
            IF(IVAL .EQ. 1)WRITE(IUNIT,2100)IP,ICL,ICP,ILINE,ICODE,
     *                          IA,X,FCOST
2100       FORMAT(2X,I3,4X,I2,5X,I2,4X,I2,5X,I2,5X,I2,5X,I2,4X,PD10.3,
     *           2X,PD10.3)
            IF(FUNCT(IRW,4) .EQ. 0.0)GO TO 4000
            IRW = IDINT(FUNCT(IRW,4))
            GO TO 200
4000        IF(IFLAG(1) .EQ. IFLAG(4))CALL TRACE(1540,
     *                I,        IRW,      K,  ILINE,      IP,
     *      FUNCT(IRW,1),  FCOST,       A,        B,        C,
     *                D,        E,      F,RPATH(K),   COST)
            RPATH(K) = FCOST
5000     CONTINUE
C
```

```
(1443)          RETURN
(1444)          END
PROGRAM SIZE:     PROCEDURE - 001256     LINKAGE - 000226     STACK - 000104
0000 ERRORS [<FUNCTS>FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE CDFUNC(IRP,RPATH1,IRDM,CD,IBDM,NPASS,NRPTH,IPH,
     *                   ADDRSS)
      IMPLICIT REAL*8(A-H,O-Z)
C*****************************************************************
C THIS SUBROUTINE WILL CALCULATE THE NODE SPLIT FRACTION         *
C WHEN CALLED.                                                   *
C                                                                *
C             ARGUMENT LIST                                      *
C     IRP    - THE INDEX OF THE FIRSRT COLUMN OF FLOW            *
C     RPATH1 - THE INCOMING NODE PATH VALUE                      *
C     IRDM   - THE ARRAY DIMENSION FOR THE RPATHS                *
C     CD     - THE NODE SPLIT FRACTION TO BE CALCULATED          *
C*****************************************************************
      DIMENSION RPATH1(750),RFC1(750)
      I2 = IRP + 1
      I3 = IRP + 2
      RFC1(1) = 0.0
      IVAL    = 0
      ADTEMP  = RPATH1(I3)
      RPATH1(I3) = ADDRSS
C/////////////////////////////////////////////////////////////////
C CHECK IF FUNCTION IS APPLICABLE.
C/////////////////////////////////////////////////////////////////
      IF(RPATH1(IRP) .EQ. 0.00) GO TO 250
C/////////////////////////////////////////////////////////////////
C COMPUTE TEMPORARY CD VALUE.
C/////////////////////////////////////////////////////////////////
      CALL FUNCTS(IRP,2,0,RPATH1,IRDM,1,RFC1,IBDM,NPASS,IUNIT,
     *            IVAL,ICP,ICL)
      FLOWR = RPATH1(IRP) * RPATH1(I3) * CD
      CD = FLOWR/RPATH1(IRP)
      RPATH1(I3) = ADTEMP
C/////////////////////////////////////////////////////////////////
C CHECK FOR CD VALUES GREATER THAN ONE AND LESS THAN ZERO
C/////////////////////////////////////////////////////////////////
      IF(IPH .EQ. NRPTH) GO TO 250
      IF(CD .GT. 1.00)CD = 1.00
      IF(CD .LT. 0.00)CD = 0.00
250   RETURN
      END
PROGRAM SIZE:  PROCEDURE - 000171    LINKAGE - 005734    STACK - 000050
0000 ERRORS [<CDFUNC>FTN-REV16.6]
```

```
C***************************************************************
C***************************************************************
C***************************************************************
      SUBROUTINE ADDRCH (RPATH,RFC,IRP,ISTCT,LSTCT,IRDM)
      IMPLICIT REAL*8(A-H,O-Z)
C***************************************************************
C  THIS SUBROUTINE WILL RECORD THE POSITIONS OF ADDRESSES IN   *
C  THE FLOW COLUMNS OF RPATH AND WHEN DESIRED RESET THOSE COL- *
C  UMNS BACK TO ADDRESSES. RFC(1) MONITORS THE RECORD AND RE-  *
C  SET OPERATIONS. IF EQUAL TO 1 IT WILL RESET, IF 0 , RECORD. *
C                                                              *
C                      ARGUMENT VALUES                         *
C                                                              *
C           RPATH - PATH VALUES                                *
C           IFC   - ARRAY TO  HOLD COLUMN POSITIONS AND MONITOR *
C                   THE RESET                                  *
C           ISTCT - FIRST CTOST COLUMN                         *
C           IRP   - POSITION OF FIRST COLUMN                   *
C           LSTCT - NUMBER OF FLOW COLUMNS FOLLOWING ISTCT     *
C***************************************************************
      DIMENSION RPATH(750),RFC(750)
C///////////////////////////////////////////////////////////////
C INITIALIZE INDICES AND DETERMINE IF RECORD OR RESET.
C///////////////////////////////////////////////////////////////
      IND1 = IRP + ISTCT
      INDL = IND1 + LSTCT
      IF(RFC(1) .EQ. 0.)GO TO 250
C///////////////////////////////////////////////////////////////
C RESET ADDRESSES IN RPATH.
C///////////////////////////////////////////////////////////////
      DO 100 K=IND1,INDL
       IF(RFC(K) .EQ. 0.)GO TO 100
       RPATH(K) = RFC(K)
100    CONTINUE
      GO TO 300
C///////////////////////////////////////////////////////////////
C RECORD ADDRESSES FROM RPATH.
C///////////////////////////////////////////////////////////////
250   DO 275 K=IND1,INDL
       IF((RPATH(K) .GT. 10.0**-47) .OR. (RPATH(K) .LT.
     *     10.0**-50))GO TO 270
       RFC(K) = RPATH(K)
       GO TO 275
270      RFC(K) = 0.
275    CONTINUE
300   RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000130    LINKAGE - 000030    STACK - 000036
0000 ERRORS [<ADDRCH>FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE OUTPT(IOSRCE,NODE,IP,ICL2,ICP2,RPATH,IRDM,IIDM,
     *RUNN,IFMT,PLACE,ARGN,IUNIT,ISTCT,LSTCT,ISTCP,LSTCP,
     *ISTCL,LSTCL,IBDM,STIME,PCOST1)
      IMPLICIT REAL*8(A-H,O-Z)
C*******************************************************************
C OUTPT GIVES THE USER A NUMBER OF DIFFERENT OUTPUT OPTIONS FOR    *
C USE WITH THE TIME RUN SUBROUTINES, MAIN AND NORMAL. OUTPT IS     *
C DIVIDED INTO TWO BLOCKS. THE FIRST WILL FORMAT AND OUTPUT THE    *
C INTERMEDIATE FLOW AND COST VALUES DURING TIME RUNS. THE SECOND   *
C WILL OUTPUT FINAL VALUES ACCORDING TO ONE OF SEVERAL FORMAT      *
C OPTIONS.                                                         *
C                                                                  *
C                       ARGUMENT LIST                              *
C                                                                  *
C        IOSRCE   - INDICATOR OF FINAL OR INTERMEDIATE OUTPUT      *
C        NODE     - NODE FROM WHICH PATH ORIGINATES                *
C        IP       - PATH WORKING ON                                *
C        ICL2     - CLASS OF TRACK                                 *
C        ICP2     - COMPONENT                                      *
C        RPATH    - PATH DATA ARRAY                                *
C        IRDM     - RPATH DIMENSION                                *
C        RUNN     - PROGRAM RUN NUMBER                             *
C        IFMT     - TABLE FORMAT INDICATOR                         *
C        PLACE    - PERSON OR GROUP RUNNING PROGRAM                *
C        ARGN     - OPEN FOR USE                                   *
C        IUNIT    - FORTRAN PRINT UNIT CODE                        *
C        ISTCT    - FIRST COST COLUMN MINUS THE REFERENCE          *
C        LSTCT    - NUMBER OF COST COLUMNS FOLLOWING THE FIRST     *
C        ISTCP    - FIRST COMPONENT CODE                           *
C        LSTCP    - LAST COMPONENT  CODE                           *
C        ISTCL    - FIRST CLSASS CODE                               *
C        LSTCL    - LAST CLASS CODE                                *
C        IBDM     - RPATH CLASS/COMPONENT BLOCK SIZE               *
C        STIME    - PRESENT SIMULATION TIME                        *
C*******************************************************************
      COMMON/I2/IFILE(100)
      COMMON/R1/ACCUM(750)
      COMMON/R4/STATE(20,30)
      COMMON/R5/RFILE(100)
      DIMENSION RPATH(750),RUNN(10),PLACE(10),DATE(10)
      DIMENSION BLOCK(50),IPATH(15)
      NBLOCK = IFILE(38)
      LSTPH = IFILE(4)
      CALL DATE$A(DATE)
C//////////////////////////////////////////////////////////////////////
C CHECK FOR TYPE OF PRINTOUT AND GO TO APPROPRIATE FORMAT BLOCK
C//////////////////////////////////////////////////////////////////////
```

```
      GO TO (1000,50,1000),IOSRCE
C/////////////////////////////////////////////////////////////////////////
C INTERMEDIATE PRINT OUT.
C/////////////////////////////////////////////////////////////////////////
50    IF(IFMT .EQ. 0)GO TO 300
      CSTSTT = 5H COST
      SUMHED = 1H
      NREFP = IFILE(13)
      IGRPH = IFILE(22)
      WRITE(IUNIT,100)CSTSTT,SUMHED,(RUNN(I),I=1,2),(DATE(I),I=1,2),
     *(PLACE(K),K=1,3),STIME
100   FORMAT(5(/),1H1,21X,18HTRACK MAINTENANCE ,2A8,/,1X,64(1H-),
     */,19H  RUN NUMBER      : ,2A8,/,19H  DATE            : ,2A8,
     */,19H  PLACE           : ,3A8,//,19H  SIMULATION TIME: ,
     *F8.2,/,1X,64(1H-))
      WRITE(IUNIT,145)
145   FORMAT(1X,64(1H-),/,8X,10HPOPULATION,9X,5HCLASS,9X,
     *          6HCOMPO-,9X,7HQUALITY,/,
     *       8X,8H  IN USE,26X,4HNENT,/,1X,64(1H-))
      DO 146 ICL = 1,LSTCL
         DO 147 ICP = 1,LSTCP
            CALL RCOLUM(ICL,LSTCL,ICP,LSTCP,IRP,IRCOL,IBDM)
            WRITE(IUNIT,148) STATE(5,IRCOL),ICL,ICP,
     *                          STATE(7,IRCOL)
148         FORMAT(8X,F10.2,11X,I1,13X,I2,11X,F7.5)
147      CONTINUE
146   CONTINUE
      WRITE(IUNIT,150)
150   FORMAT(1X,64(1H-),/,2X,4HNODE,2X,5HPATH ,2X,5HCLASS,2X,
     *        6HCOMPO-,3X,7HQUALITY,4X,10HPATH FLOWS,2X,
     *11HMAINTENANCE,/,22X,5HONENT,15X,10HMILES/YEAR,4X,
     *7HDOLLARS,/,1X,64(1H-))
      IFMT = 0
300   CALL RCOLUM(ICL2,LSTCL,ICP2,LSTCP,IRP,IRCOL,IBDM)
      IRP2 = IRP + 1
      IUSEQ = IRP + 5
      USEQ = RPATH(IUSEQ)
      I1 = IRP + ISTCT
      I2 = I1  + LSTCT
      DO 325 I=I1,I2
         PCOST = RPATH(IRP)*RPATH(I)*(1.0-USEQ) +
     *           RPATH(IRP)*RPATH(IRP2)*RPATH(I)*USEQ + PCOST
325   CONTINUE
      PCOST1 = PCOST + PCOST1
      IF(RPATH(IRP) .EQ. 0.0)GO TO 2000
      WRITE(IUNIT,350)NODE,IP,ICL2,ICP2,RPATH(IRP2),RPATH(IRP),
     *                PCOST
      PCOST = 0.0
350   FORMAT(2X,I3,4X,I3,5X,I1,6X,I2,5X,F8.6,2X,F11.2,3X,
     *I10)
```

```
      GO TO 2000
C///////////////////////////////////////////////////////////////////////
C ABNRML AND FINAL STATE PRINT FORMATS.
C///////////////////////////////////////////////////////////////////////
1000  SUMHED = 1H
      IF(IOSRCE .EQ. 3)GO TO 1100
      SUMHED = 7HSUMMARY
1100  IF(IFMT .EQ. 2)GO TO 2500
C//////////////////////////////////////////////////////////////////////
C FINAL COST TABLE FORMAT.
C//////////////////////////////////////////////////////////////////////
      CSTSTT = 5H COST
      WRITE(IUNIT,100)CSTSTT,SUMHED,(RUNN(I),I=1,2),(DATE(I),I=1,2),
     *(PLACE(I),I=1,3),STIME
      WRITE(IUNIT,1244)
      I2 = ISTCT + LSTCT
      DO 1250 K = ISTCT,I2
          DO 1200 ICL = ISTCL,LSTCL
              DO 1150 ICP = ISTCP,LSTCP
                  CALL RCOLUM(ICL,LSTCL,ICP,LSTCP,IRP,IRCOL,IBDM)
                  J = IRP+K
                  ACCMED = ACCMED + ACCUM(J)
1150          CONTINUE
1200      CONTINUE
          ICODE = K+25
          WRITE(IUNIT,1204)ICODE,ACCMED
1204      FORMAT(7X,I2,10X,F12.2)
          CTOTAL = CTOTAL + ACCMED
          ACCMED = 0.0
1250  CONTINUE
1244  FORMAT(1X,32(1H-),/,6X,5HCOST ,10X,11HMAINTENANCE,/,
     *5X,4HCODE,14X,4HCOST,/,1X,32(1H-))
1254  FORMAT(4X,14HANNUAL TOTAL :,1X,F12.2,///)
      WRITE(IUNIT,1254)CTOTAL
      CTOTAL = 0.0
      WRITE(IUNIT,1304)
1304  FORMAT(1X,32(1H-),/,6X,5HCLASS,10X,11HMAINTENANCE,/,
     *24X,4HCOST,/,1X,32(1H-))
      DO 1450 ICL = ISTCL,LSTCL
         DO 1425 ICP = ISTCP,LSTCP
             CALL RCOLUM(ICL,LSTCL,ICP,LSTCP,IRP,IRCOL,IBDM)
             I1 = IRP + ISTCT
             I2 = I1 + LSTCT
             DO 1400 K = I1,I2
                 ACCMED = ACCMED + ACCUM(K)
1400         CONTINUE
1425     CONTINUE
           WRITE(IUNIT,1204)ICL,ACCMED
           CTOTAL = CTOTAL + ACCMED
           ACCMED = 0.0
```

```
1450  CONTINUE
      WRITE(IUNIT,1254)CTOTAL
      CTOTAL = 0.0
      WRITE(IUNIT,1454)
1454  FORMAT(1X,32(1H-),/,6X,5HCOMP-,10X,11HMAINTENANCE,/,
     *6X,5HONENT,14X,4HCOST,/,1X,32(1H-))
      DO 1600 ICP = ISTCP,LSTCP
         DO 1550 ICL = ISTCL,LSTCL
            CALL RCOLUM(ICL,LSTCL,ICP,LSTCP,IRP,IRCOL,IBDM)
            I1 = IRP + ISTCT
            I2 = I1 + LSTCT
            DO 1500 K = I1,I2
               ACCMED = ACCMED + ACCUM(K)
1500        CONTINUE
1550     CONTINUE
         WRITE(IUNIT,1204)ICP,ACCMED
         CTOTAL = CTOTAL + ACCMED
         ACCMED = 0.0
1600  CONTINUE
      WRITE(IUNIT,1254)CTOTAL
      CTOTAL = 0.0
      DO 4500 I = 1,LSTPH
         CALL PATHIN(I,IPATH,IIDM,RPATH,IRDM)
        IRP = 1
         MB = IPATH(8)
4050     IF((IRP .GT. IRDM) .OR. (MB .EQ. 0)) GO TO 4500
         I2 = IRP + 1
         I3 =IRP + ISTCT
         I4 = I3 + LSTCT
         I5 = IRP + 5
         DO 4100 K = I3,I4
            BLOCK(MB) = RPATH(IRP)*RPATH(K)*RPATH(I2)*RPATH(I5)+
     *                  RPATH(IRP)*RPATH(K)*(1.0-RPATH(I5))+
     *                  BLOCK(MB)
4100     CONTINUE
         IRP = IRP + IBDM
         GO TO 4050
4500  CONTINUE
      WRITE(IUNIT,4550)
4550  FORMAT(1X,32(1H-),/,6X,6HREPAIR,12X,5HBLOCK,/
     *        7X,5HBLOCK,13X,4HCOST,/,1X,32(1H-))
      DO 4650 K = 1,NBLOCK
         WRITE(IUNIT,4600)K,BLOCK(K)
4600     FORMAT(8X,I2,9X,F10.2)
         BLOCK(K) = 0.00
4650  CONTINUE
C/////////////////////////////////////////////////////////////////////////
C STATE TABLE FORMAT.
C/////////////////////////////////////////////////////////////////////////
2500  IF(IFMT .EQ. 1)GO TO 2000
```

```
           CSTSTT = 5HSTATE
           WRITE(IUNIT,100)CSTSTT,SUMHED,(RUNN(I),I=1,2),(DATE(I),I=1,2),
          *(PLACE(I),I=1,3),STIME
           WRITE(IUNIT,2550)
     2550  FORMAT(11X,
          *53HSYSTEM   DEFECT   SYSTEM   DEFECT    SYSTEM    DEFECT,/,
          *1X,36H TRK CPT QUANTITY FRACTION QUANTITY ,
          *28HFRACTION  RATE  OF  RATE  OF,/,
          *1X,32H CLS  #  AT START AT START   NOW,6X,3HNOW,6X,
          *16HGROWTH    CHANGE,/,1X,64(1H-))
           DO 2700 ICL = ISTCL,LSTCL
              DO 2600 ICP = ISTCP,LSTCP
                 CALL RCOLUM(ICL,LSTCL,ICP,LSTCP,IRP,IRCOL,IBDM)
                 DO 1123 NTMP = 5,17,4
                    NTMP2 = NTMP + 1
                    NTMP3 = NTMP + 2
                    NTMP4 = NTMP + 3
                    WRITE(IUNIT,2654)ICL,ICP,STATE(1,IRCOL),STATE(3,IRCOL),
          *         STATE(NTMP,IRCOL),STATE(NTMP3,IRCOL),STATE(NTMP2,IRCOL),
          *         STATE(NTMP4,IRCOL)
     1123        CONTINUE
     2654        FORMAT(3X,I1,2X,I1,2X,PD8.2,1X,PD8.2,1X,PD8.2,1X,
          *      PD8.2,1X,PD9.2,1X,PD9.2)
     2600     CONTINUE
     2700  CONTINUE
     2000  SUMHED = 8H
           RETURN
           END
PROGRAM SIZE:   PROCEDURE - 003616    LINKAGE - 000552    STACK - 000130
0000 ERRORS [<OUTPT >FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE HEADNG(IUNIT,DT,TIMEL,STIME,LSTCP,LSTCL,
     *                  LSTPH,LSTND,IBDM)
      IMPLICIT REAL*8(A-H,O-Z)
C*****************************************************************
C THIS SUBROUTINE WILL PRINT OUT A HEADING AT THE TOP OF EACH   *
C RUN OUTPUT. IT WILL BE CALLED BY MAIN ONLY, AND IS CALLED ON-*
C LY OUNCE PER RUN.                                              *
C                                                                *
C                         ARGUMENT LIST                          *
C            (SEE RFILE AND IFILE FOR DESCRIPTIONS)              *
C                                                                *
C              IUNIT  - FORTRAN OUTPUT DEVICE NUMBER             *
C              DT     - TIME INCREMENT                           *
C              TIMEL  - FINAL SIMULATION TIME                    *
C              STIME  - START SIMULATION TIME                    *
C              LSTCP  - NUMBER OF COMPONENTS                     *
C              LSTCL  - NUMBER OF CLASSES                        *
C              LSTPH  - NUMBER OF PATHS                          *
C              LSTND  - NUMBER OF NODES                          *
C              IBDM   - CLASS/COMPONENT BLOCK SIZE               *
C*****************************************************************
      COMMON/I2/IFILE(100)
      COMMON/R2/FUNCT(10,10)
      COMMON/R4/STATE(20,30)
      COMMON/R6/WEIBL(3,30)
      COMMON/R7/REVISN(8),SCHMDG(8),DATAFL(8)
      LSTPH = IFILE(4)
C/////////////////////////////////////////////////////////////////
C PRINT HEADINGS
C/////////////////////////////////////////////////////////////////
      WRITE(IUNIT,1000)(REVISN(I),I=1,8),(SCHMDG(I),I=1,8),
     *                  (DATAFL(I),I=1,8),LSTCL,LSTCP,LSTPH,
     *                  LSTND,STIME,TIMEL,DT
1000  FORMAT(3(1X,8A8,/),6X,25HNUMBER OF TRACK CLASSES :,
     *       28X,I2,/,
     *       6X,28HNUMBER OF TRACK COMPONENTS :,25X,I2,/,
     *       6X,28HPATHS IN SCHEMATIC DIAGRAM :,24X,I3,/,
     *       6X,28HNODES IN SCHEMATIC DIAGRAM :,24X,I3,/,
     *       6X,32HTIME SIMULATION BEGINS AT YEAR :,17X,
     *       F6.2,/,
     *       6X,30HTIME SIMULATION ENDS ON YEAR :,19X,F6.2,/,
     *       6X,30HTIME INCREMENT SIZE IN YEARS :,19X,F6.2,//,
     *       18X,31HWEIBULL DISTRIBUTION PARAMETERS,/,
     *       1X,64(1H-),/,
     *       6X,5HCLASS,5X,9HCOMPONENT,6X,8HAVG. AGE,5X,
     *       4HBETA,6X,6HC-LIFE,/,1X,64(1H-))
      DO 1100 ICL = 1,LSTCL
```

```
            DO 1200 ICP = 1,LSTCP
               CALL RCOLUM(ICL,LSTCL,ICP,LSTCP,IRP,IRCOL,IBDM)
               WRITE(IUNIT,2000)ICL,ICP,(WEIBL(I,IRCOL),I=1,3)
2000           FORMAT(8X,I1,11X,I1,11X,F6.2,5X,F5.2,6X,F6.2)
1200        CONTINUE
1100     CONTINUE
         WRITE(IUNIT,3000)
3000     FORMAT(//,15X,37HFUNCTIONS AND SHAPE PARAMETERS ACTIVE,/,
     *         1X,64(1H-),/,2X,4HPATH,2X,5HFUNC.,6X,1HA,9X,1HB,9X,
     *         1HC,9X,2HX0,7X,2HY0,/,1X,64(1H-))
         I = 1
1150     IF(FUNCT(I,1) .EQ. 0.0)GO TO 6000
         WRITE(IUNIT,4000)(FUNCT(I,J),J=1,2),(FUNCT(I,K),K=6,10)
4000     FORMAT(3X,I3,3X,I2,4X,5(PD9.2,1X))
         I = I + 1
4001     FORMAT(I2)
         GO TO 1150
6000     RETURN
         END
PROGRAM SIZE:   PROCEDURE - 001137    LINKAGE - 000066    STACK - 000050
0000 ERRORS [<HEADNG>FTN-REV16.6]
```

```
C***************************************************************
C***************************************************************
C***************************************************************
      SUBROUTINE NDVAL(IRDM,IIDM,ISTND,LSTND,ISTCL,LSTCL,ISTCP,
     *LSTCP,IBDM,NREFP)
      IMPLICIT REAL*8(A-H,O-Z)
C***************************************************************
C     PROGRAM NDVAL CALCULATES NODE SPLIT FRACTIONS FOR SIMPLE *
C     AND DEFECTIVE BASED BRANCH DECISIONS AND STORES THESE VA-*
C     LUES IN RNODE DATA FILE.                                 *
C                                                              *
C                    ARGUMENT LIST                             *
C             SEE IFILE AND RFILE FOR DETAIL                   *
C                                                              *
C         IRDM   - RPATH DIMENSION                             *
C         IIDM   - IPATH DIMENSION                             *
C         ISTND  - FIRST NODE                                  *
C         LSTND  - LAST NODE                                   *
C         ISTCL  - FIRST CLASS                                 *
C         LSTCL  - LAST CLASS                                  *
C         ISTCP  - FIRST COMPONENT                             *
C         LSTCP  - LAST COMPONENT                              *
C         IBDM   - CLASS/COMPONENT BLOCK SIZE                  *
C***************************************************************
      COMMON/I2/IFILE(100)
      COMMON/I3/IFLAG(10)
      COMMON/I7/INODE(6,300)
C
      COMMON/R1/ACCUM(750)
      COMMON/R2/FUNCT(10,10)
      COMMON/R3/RNODE(30,300)
      COMMON/R4/STATE(9,20)
      COMMON/R5/RFILE(100)
C
      DIMENSION RPATH1(750),RPATH2(750),RPATH3(750)
      DIMENSION IPATH1(15),IPATH2(15),IPATH3(15)
C///////////////////////////////////////////////////////////////
C FILL IN EMPTY PATHS BELOW SUM NODES.
C///////////////////////////////////////////////////////////////
C
      DO 1000 J = ISTND,LSTND
C
         IF1 = INODE(1,J)
         IF2 = INODE(2,J)
         IF3 = INODE(3,J)
         IF(INODE(4,J) .EQ.1)GO TO 1000
         CALL PATHIN(IF1,IPATH1,IIDM,RPATH1,IRDM)
         CALL PATHIN(IF2,IPATH2,IIDM,RPATH2,IRDM)
         CALL PATHIN(IF3,IPATH3,IIDM,RPATH3,IRDM)
         DO 850  ICL = ISTCL,LSTCL
```

```
                DO 900    ICP = ISTCP,LSTCP
                  CALL RCOLUM(ICL,LSTCL,ICP,LSTCP,IRP,IRCOL,IBDM)
                  IF(RPATH3(IRP) .GT. 0.0)GO TO 900
                  K = IRP + 1
C
                  RPATH3(IRP) = RPATH1(IRP)+RPATH2(IRP)
                  IF(RPATH3(IRP) .EQ. 0.00)GO TO 900
                  QUAL1 =RPATH1(IRP)*RPATH1(K)
                  QUAL2 = RPATH2(IRP)*RPATH2(K)
                  RPATH3(K) = (QUAL1+QUAL2)/RPATH3(IRP)
                  CALL PATOUT(IF3,IPATH3,IIDM,RPATH3,IRDM)
900             CONTINUE
850         CONTINUE
1000    CONTINUE
C//////////////////////////////////////////////////////////////////////
C DETERMINE SPLIT FRACTIONS FOR BRANCH NODES.
C//////////////////////////////////////////////////////////////////////
        DO 1200 J = ISTND,LSTND
          IF(INODE(4,J) .EQ. 0)GO TO 1200
          IF1 = INODE(1,J)
          IF2 = INODE(2,J)
          IF3 = INODE(3,J)
          CALL PATHIN(IF1,IPATH1,IIDM,RPATH1,IRDM)
          CALL PATHIN(IF2,IPATH2,IIDM,RPATH2,IRDM)
          CALL PATHIN(IF3,IPATH3,IIDM,RPATH3,IRDM)
          DO 1100 ICL = ISTCL,LSTCL
             DO 1150 ICP = ISTCP,LSTCP
                  CALL RCOLUM(ICL,LSTCL,ICP,LSTCP,IRP,IRCOL,IBDM)
                  K = IRP+1
                  IF(INODE(5,J) .EQ. 1)GO TO 800
C
                  IF(RPATH1(IRP) .EQ. 0.0) GO TO 700
                  RNODE(IRCOL,J) = RPATH2(IRP) / RPATH1(IRP)
                  IF(INODE(6,J) .EQ. 1)
     *             RNODE(IRCOL,J) = RPATH3(IRP)/RPATH1(IRP)
                  GO TO 1148
C
700               RNODE(IRCOL,J) = 0.0
                  GO TO 1148
800               BAD1 = RPATH1(IRP)*RPATH1(K)
                  IF(BAD1 .EQ. 0.0)GO TO 700
                  RNODE(IRCOL,J) = RPATH2(IRP)/BAD1
                  IF(INODE(6,J) .EQ. 1)RNODE(IRCOL,J) = RPATH3(IRP)/
     *                                                  BAD1
1148              IF(RNODE(IRCOL,J) .LT. 0.0) RNODE(IRCOL,J) = 0.00
                  IF((IF1 .NE. NREFF) .AND. (RNODE(IRCOL,J) .GT. 1.0))
     *              RNODE(IRCOL,J) = 1.0
1150          CONTINUE
1100        CONTINUE
1200    CONTINUE
```

```
(1932)          RETURN
(1933)          END
PROGRAM SIZE:    PROCEDURE - 000622    LINKAGE - 021621    STACK - 000054
0000 ERRORS [<NDVAL >FTN-REV16.6]
```

```
C****************************************************************
C****************************************************************
C****************************************************************
      SUBROUTINE INTDAT(LSTPH,IIDM,IRDM)
      IMPLICIT REAL*8(A-H,O-Z)
C****************************************************************
C INTDAT WILL CLEAR THE DIRECT ACCESS FILES PRIOR TO READING IN*
C THE DATA.                                                     *
C                                                               *
C                              ARGUMENT LIST                    *
C                    (SEE RFILE AND IFILE FOR DETAIL)           *
C                                                               *
C       LSTPH  - NUMBER OF PATH IN MODEL                        *
C       IIDM   - DIMENSION OF IPATH                             *
C       IRDM   - DIMENSION OF RPATH                             *
C****************************************************************
      DIMENSION IPATH(15),RPATH(750)
      DO 100 I = 1,IIDM
         IPATH(I) = 0
         RPATH(I) = 0.00
100   CONTINUE
      DO 200 I = IIDM,IRDM
         RPATH(I) = 0.00
200   CONTINUE
      DO 250 I = 1,LSTPH
         IPATH(1) = I
         CALL PATOUT(I,IPATH,IIDM,RPATH,IRDM)
250   CONTINUE
      RETURN
      END
PROGRAM SIZE:  PROCEDURE - 000100   LINKAGE - 005732   STACK - 000024
0000 ERRORS [<INTDAT>FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE RAWDAT(LSTCP,LSTCL,IDCODE,IIDM,IRDM,IBDM,ISTCT,
     *LSTCT)
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/I2/IFILE(100)
C
      COMMON/R2/FUNCT(10,10)
      COMMON/R4/STATE(20,30)
      COMMON/R5/RFILE(100)
      COMMON/R7/REVISN(8),SCHHDG(8),DATAFL(8)
      DIMENSION REALA(12),REALB(9),REALC(6),REALD(3),REAL2(12),
     *FTMP(10,10)
      DIMENSION IPATH(15),IFTLNK(20)
      DIMENSION RPATH(750),VALUE(50),RL(14)
C
      NPATHR = IFILE(13)
      NCOSTS = IFILE(17)
      IRUNIT = IFILE(22)
      ITUNIT = IFILE(23)
      DO 7 I=1,20
        IFTLNK(I) = I+1
        RPATH(I) = 0.0
7     CONTINUE
      DO 9 I = 1,10
         DO 8 K = 1,10
            FTMP(I,K) = 0.00D-01
8        CONTINUE
9     CONTINUE
      NXTAV = 1
      ISTLNK = 1
      ITMP = 0
      ICOUNT = 0
C
      ITRACE = 0
C      WRITE(ITUNIT,1000)ITRACE
      CALL ASSIGN(RL,NCOSTS,VALUE,REAL2,IFLAG)
C      WRITE(ITUNIT,565)VALUE(1),VALUE(9)
565   FORMAT(2(1PD10.3,1X),'VALUES IN RAWDAT')
      ITRACE = 1
C      WRITE(ITUNIT,1000)ITRACE
1000  FORMAT(I1)
C
C
      READ(IRUNIT,83)(DATAFL(K),K=1,8)
83    FORMAT(8A8)
      READ(IRUNIT,*)IPH,ICD,RL(1),RL(2),RL(3),RL(4),RL(5),RL(6),
     *         RL(7),RL(8),RL(9),RL(10)
```

```
C
      CALL PATHIN(IPH,IPATH,IIDM,RPATH,IRDM)
      CALL RESET(RL,ICP,ICL,IND,ITYPE,ICST,IPDT,IQDT,IMFG,IRPB,
     *           IPRT)
      IRP = 1
      ITRACE = 2
C      WRITE(ITUNIT,1000)ITRACE
C
10    CALL RCLEAR(RL)
      ITRACE = 3
C      WRITE(ITUNIT,1000)ITRACE
C
C
C
C
C
      READ(IRUNIT,*)IPH,ICD,RL(1),RL(2),RL(3),RL(4),RL(5),
     *          RL(6),RL(7),RL(8),RL(9),RL(10),RL(11),
     *          RL(12),RL(13),RL(14)
C
15    CALL RCOLUM(ICL,LSTCL,ICP,LSTCP,IRP,IRCOL,IBDM)
      IF(ICD .EQ. 1)GO TO 20
      IF(ICD .EQ. 2)GO TO 30
      IF(ICD .EQ. 3) GO TO 40
C
C      WRITE(ITUNIT,565)VALUE(1),VALUE(9)
20    CALL DATA1(RL,VALUE,NCOSTS,ISTCT,IPH)
      ITRACE = 4
C      WRITE(ITUNIT,1000)ITRACE
      GO TO 50
C
30    IFLAG = 1
      CALL REALCL(REALA,REALB,REALC,REALD,REAL2)
      ITRACE = 5
C      WRITE(ITUNIT,1000)ITRACE
      READ(IRUNIT,*)IPH,ICD,IFCODE,IFNUMB,REALA(1),REALA(2),REALA(3),
     *          REALA(4),REALA(5),REALA(6),REALA(7),REALA(8),
     *          REALA(9),REALA(10),REALA(11),REALA(12)
C
      IF(ICD .EQ. 9)READ(IRUNIT,*)IPH,ICD,IFCODE,IFNUMB,REALB(1),
     *                        REALB(2),REALB(3),REALB(4),REALB(5),
     *                        REALB(6),REALB(7),REALB(8),REALB(9)
C
      IF(ICD .EQ. 9)READ(IRUNIT,*)IPH,ICD,IFCODE,IFNUMB,REALC(1),
     *               REALC(2),REALC(3),REALC(4),REALC(5),REALC(6)
C
      IF(ICD .EQ. 9)READ(IRUNIT,*)IPH,ICD,IFCODE,IFNUMB,REALD(1),
     *                        REALD(2),REALD(3)
      CALL REFILL(REALA,REALB,REALC,REALD,REAL2)
      ITRACE = 6
C      WRITE(ITUNIT,1000)ITRACE
```

```
C
C         WRITE(ITUNIT,37)(REAL2(I2),I2=1,12)
37       FORMAT(1X,2(6(PD8.3,1X),/)/)
         CALL DATA2(RL,IFCODE,NCOSTS,REAL2,ITMP,ICOUNT,
     *        IDCODE,VCODE,IPH,IFNUMB,ISTCT,FTMP,IFTLNK,NXTAV,
     *               ISTLNK,RPATH,IRP)
         ITRACE = 7
C         WRITE(ITUNIT,1000)ITRACE
C
         GO TO 50
C
C
C
C40       CALL DATA3(IPH,RL,LSTCP,LSTCL,ICODE,VALUE,NCOSTS,
C    *           IPATH,IIDM,RPATH,IRDM,IDCODE,ICL,ISTCL,ICP,ISTCP,ISTCT)
40         ITRACE = 8
C         WRITE(ITUNIT,1000)ITRACE
C
C
C
C
50       CALL RCLEAR(RL)
         ITRACE = 9
C         WRITE(ITUNIT,1000)ITRACE
C
500      READ(IRUNIT,*)ICK,ICD,RL(1),RL(2),RL(3),RL(4),RL(5),
     *              RL(6),RL(7),RL(8),RL(9),RL(10),RL(11),
     *              RL(12),RL(13),RL(14)
C
C         WRITE(ITUNIT,1000)ITRACE
         IF((ICK .EQ. 999) .OR. (ICD .EQ. 0))GO TO 60
C         WRITE(ITUNIT,1000)ITRACE
         GO TO 15
C
C
C
C         WRITE(ITUNIT,2000)ITRACE
2000     FORMAT(I2)
60       CALL DUMMY(IRP,IPH,ITYPE,IFCODE,IDCODE,ISTCT,VALUE,
     *               NCOSTS,IPATH,IIDM,RPATH,IRDM,ICST,IPDT,
     *               IQDT,IMFG,IRPB,LSTCT,FTMP,NPATHR,IRCOL,
     *               IPRT,ITMP)
         ITRACE = 11
C         WRITE(ITUNIT,2000)ITRACE
C
C
C         WRITE(ITUNIT,57)(IPATH(I),I = 1,15)
57       FORMAT('POST DUMMY ',I3)
C         WRITE(ITUNIT,58)IPH,(RPATH(I),I = 1,20)
C58       FORMAT(I3,2X,5(2X,5(1PD10.3),/))
58       FORMAT('POST N1/XVALUE CHANGE ',I3)
```

```
            N = ISTLNK
            ICNT = ICOUNT+ITMP
            IF(ITMP .EQ. 0)GO TO 850
            DO 875 N = 1,ITMP
               N1 = IDINT(FTMP(N,3))
C               WRITE(ITUNIT,2000)N1
C               WRITE(ITUNIT,2000)ICOUNT
               VALUE(N1) = 0.0
C               WRITE(ITUNIT,200)(FTMP(N,K),K=1,10)
200            FORMAT(2(5(1PD10.3,1X),/),/)
875         CONTINUE
            N = ISTLNK
806         ICOUNT = ICOUNT + 1
            DO 808 I = 1,10
               FUNCT(ICOUNT,I) = FTMP(N,I)
808         CONTINUE
            N = IFTLNK(N)
            IF(ICOUNT .LT. ICNT) GO TO 806
850         ITMP = 0
            CALL PATOUT(IPH,IPATH,IIDM,RPATH,IRDM)
            ITRACE = 12
            DO 857 I = 1,20
               IFTLNK(I) = I+1
               RPATH(I) = 0.0
857         CONTINUE
            NXTAV = 1
            ISTLNK = 1
C            WRITE(ITUNIT,2000)ITRACE
            IF(ICK .EQ. 999)GO TO 100
            ICT = ICOUNT
            IPH = ICK
            CALL RESET(RL,ICP,ICL,IND,ITYPE,ICST,IPDT,IQDT,IMFG,IRPB,
           *           IPRT)
            CALL ASSIGN(RL,NCOSTS,VALUE,REAL2,IFLAG)
            CALL PATHIN(IPH,IPATH,IIDM,RPATH,IRDM)
            GO TO 10
C
C
C
100         RETURN
            END
PROGRAM SIZE:   PROCEDURE - 002232    LINKAGE - 007601    STACK - 000050
0000 ERRORS [<RAWDAT>FTN-REV16.6]
```

```
C******************************************************************
C******************************************************************
C******************************************************************
      SUBROUTINE ASSIGN(RL,NCOSTS,VALUE,REAL2,IFLAG)
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION REAL2(12)
      DIMENSION VALUE(NCOSTS),RL(14)
C
      DO 100  I = 1,14
100      RL(I) = 0.0
C
C
      DO 200  I = 1,NCOSTS
         VALUE(I) = 0.000 00
200   CONTINUE
C
C
      DO 300  I = 1,12
300      REAL2(I) = 0.0
C
C
C
C
C
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000066    LINKAGE - 000024    STACK - 000032
0000 ERRORS [<ASSIGN>FTN-REV16.6]
```

```
C***********************************************************
C***********************************************************
C***********************************************************
      SUBROUTINE RESET(RL,ICP,ICL,IND,ITYPE,ICST,IPDT,IQDT,IMFG,
     *                 IRPB,IPRT)
C
      REAL*8 RL(14)
C
      ICP = IFIX(RL(1))
C
      ICL = IFIX(RL(2))
C
      IND = IFIX(RL(3))
C
      ITYPE = IFIX(RL(4))
      ICST = IFIX(RL(3))
      IPDT = IFIX(RL(4))
      IQDT = IFIX(RL(5))
      IMFG = IFIX(RL(6))
      IRPB = IFIX(RL(7))
      IPRT = IFIX(RL(8))
C
C
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000071    LINKAGE - 000027    STACK - 000056
0000 ERRORS [<RESET >FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE RCLEAR(RL)
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION RL(14)
C
      DO 100  I = 1,13
         RL(I) = 0.0
100   CONTINUE
C
C
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000025    LINKAGE - 000023    STACK - 000016
0000 ERRORS [<RCLEAR>FTN-REV16.6]
```

```
C************************************************************
C************************************************************
C************************************************************
      SUBROUTINE REFILL(REALA,REALC,REALB,REALD,REAL2)
      IMPLICIT REAL*8(A-H,O-Z)
C****************************************************
C THIS SUBROUTINE WILL TAKE DATA STORED IN REALA,REALB
C REALC AND REALD AND PATCK IT INTO REAL2 TO BE USED IN
C DATA2.
C****************************************************
      DIMENSION REALA(12),REALB(9),REALC(6),REALD(3)
      DIMENSION REAL2(12)
C
      I = 1
50    REAL2(I) = REALA(I)
      I = I+1
      IF(I .EQ. 13)GO TO 200
      IF(REALA(I) .NE. 0.0)GO TO 50
C
      IF(REALB(1) .EQ. 0.0)GO TO 200
      IB = 1
75    REAL2(I) = REALB(IB)
      I = I+1
      IF(I .EQ. 13)GO TO 200
      IB = IB+1
      IF(IB .EQ. 10)GO TO 80
      IF(REALB(IB) .NE. 0.0)GO TO 75
C
80    IF(REALC(1) .EQ. 0.0)GO TO 200
      IC = 1
100   REAL2(I) = REALC(IC)
      I = I+1
      IF(I .EQ. 13)GO TO 200
      IC = IC+1
      IF(IC .EQ. 7)GO TO 120
      IF(REALC(IC) .NE. 0.0)GO TO 200
C
120   IF(REALD(1) .EQ. 0.0)GO TO 200
      ID = 1
125   REAL2(I) = REALD(ID)
      I = I+1
      IF(I .EQ. 13)GO TO 200
      ID = ID+1
      IF(ID .EQ. 4)GO TO 200
      IF(REALD(I) .NE. 0.0)GO TO 125
200   RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000214    LINKAGE - 000031    STACK - 000032
0000 ERRORS [<REFILL>FTN-REV16.6]
```

```
C************************************************************
C************************************************************
C************************************************************
      SUBROUTINE DATA1(RL,VALUE,NCOSTS,ISTCT,IPH)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION RL(14),RVAL(24),VALUE(50)
C
C
C      WRITE(ITUNIT,1)RL(1)
1     FORMAT('RL',1X,1PD10.3)
      ITRACE = 55
C      WRITE(ITUNIT,777)ITRACE
777   FORMAT('ITRACE',1X,I3)
      ICOST =  IDINT(RL(1))
C
C      WRITE(ITUNIT,2)ICOST
2     FORMAT('ICOST',1X,I2)
      ITRACE = 56
C      WRITE(ITUNIT,777)ITRACE
      IF((ICOST .GE. 30) .AND. (ICOST .LE. 49))GO TO 200
      VALUE(ICOST) =VALUE(ICOST)+RL(2)
C
      ITRACE = 57
C      WRITE(ITUNIT,777)ITRACE
      GO TO 500
C
200   VALUE(ICOST) = VALUE(ICOST)+(RL(2)*RL(3)*RL(4))
C
      WASH = RL(2)*RL(3)*RL(4)
      IF(VALUE(20) .LE. 0.0000)GO TO 500
      WASH = WASH * VALUE(20)
      ITRACE = 58
C      WRITE(ITUNIT,777)ITRACE
500   RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000143    LINKAGE - 000201    STACK - 000040
0000 ERRORS [<DATA1 >FTN-REV16.6]
```

```
C****************************************************************
C****************************************************************
C****************************************************************
      SUBROUTINE DATA2(RL,IFCODE,NCOSTS,REAL2,ITMP,ICOUNT,
     *                 IDCODE,VCODE,IPH,IFNUMB,ISTCT,FTMP,
     *                 IFTLNK,NXTAV,ISTLNK,RPATH,IRP)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION REAL2(12),FTMP(10,10),RL(14),RPATH(750),IFTLNK(20)
C
C THIS SUBROUTINE WILL READ IN VALUES OTO BE STORED IN
C THE FUNCT ARRAY, STORE THEM IN A TEMPORARY ARRAY AND
C STORE THE FUTURE FUNCT ADDRESS BACK IN RPATH.
C                ARGUMENT LIST
C   1 RL      = ARRAY
C   2 IFCODE =
C   3 REAL2   = ARRAY HOLDING FUNCT DATA
C   4 ITMP    = # OF LINES IN TEMPORARY ARRAY
C   5 ICOUNT = # OF LINES CURRENTLY STORED IN FUNCT
C   6 IDCODE = LAST COST CODE
C   7 IPH     = PATH PRESENTLY ON
C   8 IFUMB   = FUNCTION #
C   9 VCODE   = X FUNCTION #
C  10 ISTCT   = # OF FLOW COLUMNS
C  11 FTMP    = TEMPORAY ARRAY FOR FUNCT VALUES
C  12 IFTLNK = FORWARD LINK ARRAY FOR FTMP
C  13 NXTAV   = NEXT AVAILABLE SPACE IN FTMP
C  14 ISTLNK = FIRST LINE OF LINKED ARRAY
C  15 RPATH   = ARRAY HOLDING FLOW AND COST VALUES AND ADDRESSES
C  16 IRP     = REFERENCE COLUMN OF RPATH
C**************************************************************
C
1000  FORMAT(I2)
C
      L = 0
C
      IF((IFCODE .GE. 20) .AND.(IFCODE .LE.21))GO TO 110
      GO TO 310
110   J = IRP
      M = IFCODE + 1
      DO 300  I = 20,21
         IF(IFCODE .EQ. I)GO TO 250
         GO TO 275
250      L = J+2
275      J = J + 1
300   CONTINUE
C
      GO TO 500
C
310   J = IRP + ISTCT
C      WRITE(ITUNIT,22)M
```

```
22      FORMAT('M',2X,I3)
C        WRITE(ITUNIT,21)IDCODE,NCOSTS
21      FORMAT('IDCODE',1X,I2,1X,'NCOSTS',1X,I2)
        DO 400  I = 31,IDCODE
           IF(IFCODE .NE. I)GO TO 375
           L = J
375       J = J + 1
400     CONTINUE
C
C        WRITE(ITUNIT,23)L
23      FORMAT('L',2X,I2)
C
C
500     VCODE = RL(2)
C        WRITE(ITUNIT,501) VCODE
501     FORMAT(' VCODE: ',I3)
        CALL FPLACE(ICOUNT,ITMP,IPH,IFNUMB,IFCODE,REAL2,FTMP,NXTAV,
       *VCODE,IFTLNK,ISTLNK,RPATH,L)
C
C
        RETURN
        END
PROGRAM SIZE:   PROCEDURE - 000215    LINKAGE - 000032    STACK - 000076
0000 ERRORS [<DATA2 >FTN-REV16.6]
```

```
C*************************************************************
C*************************************************************
C*************************************************************
      SUBROUTINE FPLACE(ICOUNT,ITMP,IPH,IFNUMB,IFCODE,REAL2,FTMP,NXTAV,
     *VCODE,IFTLNK,ISTLNK,RPATH,L)
      IMPLICIT REAL*8(A-H,O-Z)
C
C THIS SUBROUTINE WILL INPUT FUNCTION DATA INTO THE LINKED LIST
C FTMP TO BE HELD UNTIL IT IS TO BE READ INTO THE FUNCT FILE.
C    ICOUNT IS THE NUMBER OF LINES CURRENTLUY IN THE FUNCT FILE
C    ITMP - THE NUMBER OF LINES CURRENTLY FILLED IN FTMP
C    IPH  - THE CURRENT PATH NUMBER
C    IFNUMB - THE FUNCTION NUMBER OF THIS DATA
C    IFCODE - THE COST CODE
C    REAL2  - ARRAY HOLDING FUNCTION CONSTANTS
C    FTMP   - TEMPORARY HOLDING ARRAY FOR FUNCT
C    NXTAV  - THE NEXT AVAILABLE SPACE IN FTMP
C   VCODE   - X FUNCTION #
C    IFTLNK - FORWARD LINK ARRAY
C    FSTLNK - POINTS TO THE FIRST DATA LINE IN FTMP
C    RPATH  - ARRAY HOLDING THE FLOW AND COST DATA
C    L      - COLMN IN RPATH CORRESPONDING TO CURRENT COSTCODE
C
      DIMENSION FTMP(10,10),REAL2(12),IFTNK(20),RPATH(750)
C
C      WRITE(ITUNIT,999)REAL2(1),REAL2(2),REAL2(3),REAL2(4)
999   FORMAT(4(F2.0,1X))
C
C      WRITE(ITUNIT,997)FVAL
997   FORMAT('THIS IS FVAL IN FPLACE',F5.0)
      FTMP(NXTAV,1) = DBLE(FLOAT(IPH))
C
      FTMP(NXTAV,2) = DBLE(FLOAT(IFNUMB))
      FTMP(NXTAV,3) = DBLE(FLOAT(IFCODE))
      FTMP(NXTAV,5) = VCODE
      K = 1
      DO 100  I = 6,10
         FTMP(NXTAV,I) = REAL2(K)
         K = K + 1
100   CONTINUE
      CALL LNKLST(IFTLNK,ICOUNT,ITMP,FTMP,NXTAV,RPATH,L,ISTLNK)
C
C
C
C      WRITE(ITUNIT,94)(FTMP(ITMP,K),K=1,10)
94    FORMAT(' FTMP-',/,2(5(PD10.3,1X),/),/)
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000220    LINKAGE - 000061    STACK - 000066
0000 ERRORS [<FPLACE>FTN-REV16.6]
```

```
C*************************************************************
C*************************************************************
C*************************************************************
      SUBROUTINE LNKLST(IFTLNK,ICOUNT,ITMP,FTMP,NXTAV,RPATH,L,ISTLNK)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION IFTLNK(20)
      DIMENSION FTMP(10,10),RPATH(750)
C THIS SUBROUTINE IS DESIGNED TO INSERT FUNCTION DATA IN
C A LINKED TEMPORARY ARRAY. WHIN THE DATRA IS INSERTED THE FUTURE
C FUTURE FUNCT ARRAY ADDRESS IS PLACED IN THE CORRESPONDING
C RPATH COLUMN. WHEN ALL THE FUNCT DATA FOR A PARTICULAR PATH-
C CLASS- COMPONEBNT IS INSERTED THE ARRAY IS THEN FED INTO THE
C FUNCT FILE AND CLEARED FOR THE NEXT RUN.
C     IFLNK  -ARRAY HOLDING THE FORWARD LINKS
C     NXTAV  - NEXT AVAILABLE SPACE
C     FTMP   - TEMPORARY ARRAY FOR FUNCT DATA
C    RPATH   - ARRAY HOLDING COST AND FLOW DATA
C    L       - COLUMN IN RPATYH WHERE FUNCT ADDRESS IS STORED
C    ICOUNT  - NUMBER OF DATA LINES IN FUNCT FILE
C    ITMP     - NUMBER OF DATA LINES IN FTMP
C********************************************************************
      ITMP = ITMP+1
50    FORMAT(' L-',I2,1X,'RPATH(L)-',PD10.3)
C      WRITE(ITUNIT,75)ITMP,ICOUNT
75    FORMAT(' ITMP-',I2,1X,'ICOUNT-',I2)
      LINK = ISTLNK
      NEWLK = NXTAV
      IDONE = 0
100   IF(IDONE .EQ. 1)GO TO 500
      IF(LINK .EQ. NEWLK) GO TO 400
      IF(FTMP(LINK,3) .GT. FTMP(NEWLK,3))GO TO 300
      IF(FTMP(LINK,3) .NE. FTMP(NEWLK,3))GO TO 200
      IF(FTMP(LINK,4) .EQ. 0.0)FTMP(LINK,4) = DBLE(FLOAT(ICOUNT+ITMP))
200   LLINK = LINK
      LINK = IFTLNK(LINK)
      GO TO 100
300   IDONE = 1
      NXTAV = IFTLNK(NEWLK)
      IFTLNK(NEWLK) = LINK
      IF(LINK .EQ. ISTLNK)GO TO 350
      IFTLNK(LLINK) = NEWLK
      GO TO 375
350   ISTLNK = NEWLK
375   IF((RPATH(L) .GT. 10.0**-47) .OR. (RPATH(L) .LT. 10.0**-50))
     *RPATH(L) = DBLE(FLOAT(ICOUNT+ITMP))*10.0**-50
      GO TO 100
400   IDONE = 1
      NXTAV = IFTLNK(NEWLK)
      IF((RPATH(L) .GT. 10.0**-47) .OR. (RPATH(L) .LT. 10.0**-50))
     *RPATH(L) = DBLE(FLOAT(ICOUNT+ITMP))*10.0**-50
```

```
          GO TO 100
C500       WRITE(ITUNIT,550)(IFTLNK(K),K=1,20),NXTAV,ISTLNK
C550       FORMAT(' IFTLNK',/,2(10(I3,1X),/),/,'NXTAV-',I2,' IST-',I2)
C          WRITE(ITUNIT,50)L,RPATH(L)
500       RETURN
          END
PROGRAM SIZE:    PROCEDURE - 000364    LINKAGE - 000036    STACK - 000054
0000 ERRORS [<LNKLST>FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE DUMMY(IRP,IPH,ITYPE,IFCODE,IDCODE,ISTCT,VALUE,
     *                 NCOSTS,IPATH,IIDM,RPATH,IRDM,ICST,IPDT,
     *                 IQDT,IMFG,IRPB,LSTCT,FTMP,NPATHR,IRCOL,
     *                 IPRT,ITMP)
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/R4/STATE(20,30)
      DIMENSION IPATH(IIDM)
      DIMENSION RPATH(750),VALUE(50),FTMP(10,10)
C
C     DO 30  I = 3,4
C        RPATH(I) = 0.0
C30    CONTINUE
C
      IPATH(1) = IPH
C     IPATH(2) =
C     IPATH(3) =
C     IPATH(4) =
      IPATH(5) = ITYPE
      IPATH(8) = IRPB
C     IPATH(9) =
      IPATH(10) = ICST
      IPATH(11) = IPDT
      IPATH(12) = IQDT
      IPATH(13) = IMFG
      IPATH(14) = IPRT
C     IPATH(15) =
C
C
      IF(ITMP .LE. 0) GO TO 100
      DO 875 N = 1,ITMP
         N1 = IDINT(FTMP(N,3))
C         WRITE(ITUNIT,2000)N1
C         WRITE(ITUNIT,2000)ICOUNT
          IF((N1 .EQ. 20) .OR. (N1 .EQ. 21)) GO TO 875
         FTMP(N,6) = VALUE(N1)+ FTMP(N,6)
875   CONTINUE
C
C      WRITE(ITUNIT,19)IRP
19    FORMAT('IRP IN DUMMY',1X,I1)
C      WRITE(ITUNIT,43)
43    FORMAT('VALUE IN DUMMY')
C      WRITE(ITUNIT,44)(VALUE(I),I = 20,49)
44    FORMAT(5(1PD10.3,1X))
C       WRITE(ITUNIT,42)
C        WRITE(ITUNIT,44)(RPATH(I),I=1,20)
C      WRITE(ITUNIT,10)RPATH(3),RPATH(4)
```

```
10     FORMAT(2(1PD10.3,1X),'RPATH 3,4')
100    IF(NPATHR .NE. IPH)GO TO 21
       STATE(1,IRCOL) = VALUE(20)
       STATE(3,IRCOL) = VALUE(21)
21     K = 20
       J = IRP + 1
C       WRITE(ITUNIT,54)J
54     FORMAT(' K-',I3)
       DO 200  I = IRP,J
          RPATH(I) = VALUE(K)
          VALUE(K) = 0.00
C          WRITE(ITUNIT,64)K
64       FORMAT(' K-',I2)
          K = K + 1
200    CONTINUE
C
C       WRITE(ITUNIT,10)RPATH(3),RPATH(4)
       J = IRP + ISTCT - 1
C       WRITE(ITUNIT,54)J
       K = 30
       M = J + LSTCT + 1
       DO 400  I =J,M
          IF((RPATH(I) .LT. 10.0**-47) .AND. (RPATH(I) .GE. 10.0**
     *    -50)) GO TO 300
C          WRITE(ITUNIT,64)K
          RPATH(I) = VALUE(K)
          VALUE(K) = 0.00
300       K = K + 1
400    CONTINUE
C
C       WRITE(ITUNIT,10)RPATH(3),RPATH(4)
C
C       WRITE(ITUNIT,43)
C       WRITE(ITUNIT,44)(VALUE(I),I = 1,20)
C
C       WRITE(ITUNIT,42)
42     FORMAT('RPATH IN DUMMY')
C       WRITE(ITUNIT,44)(RPATH(I),I = 1,20)
       RETURN
       END
PROGRAM SIZE:   PROCEDURE - 000436    LINKAGE - 000054    STACK - 000122
0000 ERRORS [<DUMMY >FTN-REV16.6]
```

```
C*****************************************************************
C*****************************************************************
C*****************************************************************
      SUBROUTINE REALCL(REALA,REALB,REALC,REALD,REAL2)
C
      DIMENSION REALA(12),REALB(9),REALC(6),REALD(3),REAL2(12)
C
      DO 100  I = 1,12
         REALA(I) = 0.0
100   CONTINUE
C
      DO 200  I = 1,9
         REALB(I) = 0.0
200   CONTINUE
C
      DO 300  I = 1,6
         REALC(I) = 0.0
300   CONTINUE
C
      DO 400  I = 1,3
         REALD(I) = 0.0
400   CONTINUE
      DO 500  I = 1,12
         REAL2(I) = 0.0
500   CONTINUE
C
      RETURN
      END
PROGRAM SIZE:   PROCEDURE - 000117    LINKAGE - 000025    STACK - 000032
0000 ERRORS [<REALCL>FTN-REV16.6]
```

APPENDIX E

Program Subroutines

PROGRAM SUBROUTINE INDEX

BY LINE NUMBER		ALPHABETICAL	
Line	Title	Title	Line
78	SWEEP	ABNRML	223
161	SETARR	ACCUME	1250
223	ABNRML	ADDRCH	1491
379	DRATE	ASSIGN	2158
414	NORMAL	BRANCH	915
658	WEIBUL	CDFUNC	1448
712	DATAIN	DATA1	2272
809	ERRORS	DATA2	2307
831	TRACE	DATAIN	712
883	PATHIN	DRATE	379
899	PATOUT	DUMMY	2483
915	BRANCH	ERRORS	809
1029	SUMS	FUNCTS	1296
1092	PDOT	FPLACE	2379
1142	QDOT	HEADNG	1766
1192	RCOLUM	INTDAT	1937
1214	PCOSTS	LNKLST	2427
1250	ACCUME	NDVAL	1835
1296	FUNCTS	NORMAL	414
1448	CDFUNC	OUTPUT	1538
1491	ADDRCH	PATHIN	883
1538	OUTPT	PATOUT	899
1766	HEADNG	PCOSTS	1214
1835	NDVAL	PDOT	1092
1937	INTDAT	QDOT	1142
1967	RAWDAT	RAWDAT	1967
2158	ASSIGN	RCLEAR	2210
2185	RESET	RCOLUM	1192
2210	RCLEAR	REALCL	2574
2225	REFILL	REFILL	2225
2272	DATA1	RESET	2185
2307	DATA2	SETARR	161
2379	FPLACE	SUMS	1029
2427	LNKLST	SWEEP	78
2483	DUMMY	TRACE	831
2574	REALCL	WEIBUL	658

This appendix consists of a program index of all subroutines by line numbers of the listing in Appendix D and a detailed explanation of selected subroutines. Although each subroutine has explanatory comments in the program listing, as well as in Appendix C (flow charts), some routines require more explanation. In particular, a full description of subroutine FUNCTS is included in this appendix. FUNCTS provides for all user implemented functional relationships needed for modifying cost or flow values. At the present time seven (7) families of functions have been entered to the program and are available to the user (see equations below). These functions are activated by entering appropriate input data to the data file such as that shown in Appendix G.

All functions make use of the operating array FUNCT (10,10) which is assembled from user supplied numbers during the data read in cycle of the program. The FUNCT array has five columns for control operation and five columns for numeric shaping of the functions programmed. The various columns 1 through 10 have the meanings shown in Figure E-1.

Assume that the amount of track from accidents being maintained is to be modified as a linear function of current rail quality. Figure E-2 shows the various steps in implementing this function by using program function number 6 which is a linear operating function using the present or current state quality. Figure E-2 shows the original graphic form of the desired function, its associated equation, the input data for activating it, and two program outputs associated with its use. The two outputs of the program are; the prompting message to the user at the top of the run output, and then the "flow" outputs from each time point of the simulation.

The prompting message lists the path and functions that are operating on those paths so that the user knows when functional "modifiers" are operating on the base data for those paths.

Seven functional families of equations are programmed for use. In order that a specific functional equation is from any of the seven forms programmed, the user must supply input data values for A, B, C, X_o and Y_o. See example at end of this appendix on how these values can generate different functions. Family of equation form by programmed function number in Subroutine FUNCTS:

1. Linear $F(X) = A \cdot X + B$

2. Power $F(X) = A \cdot (X-X_o)^B + Y_o$

3. Exponential $F(X) = A \cdot (Exp(B (X-X_o))) + Y_o$

4. Weibull Probability Density* $F(X) = \frac{(B \cdot X^{B-1})}{A^B} \cdot (Exp(-(\frac{X}{A})^B))$

5. Hyperbolic Tangent $F(X) = A (1+Z_1) + C(1-Z_2) + Y_o$

where Z_1 and Z_2 are explicitly given in Subroutine "FUNCTS"

6. Linear on Quality $F(Q) = A \cdot Q/B + C$

7. Explicit on Flow $F(f_1) = A(\frac{f_1}{f_1+A \cdot B})$

where

X = a dummy variable, usually simulation time in years during a simulation run. Independent variable selectable, see Figure E-1 column 5.

A, B, C, X_o, Y_o = Constants input by user (See example at the end of this appendix).

Q = Quality of component at time step during simulation.

f_1 = Path flow of component during simulation on path designated by user.

---

*This results from Weibull cumulative distribution which is defined as

$$f(x) = 1-Exp[-(\frac{X}{A})B]$$

where A>0, B>0, X≥0

The failure rate H(X) is given by $H(X) = \frac{B}{A^B}(X^{B-1})$

(FUNCT ARRAY Rows Presently Limited to 10, Expandable Indefinitely by User, See Program Common Block R2)

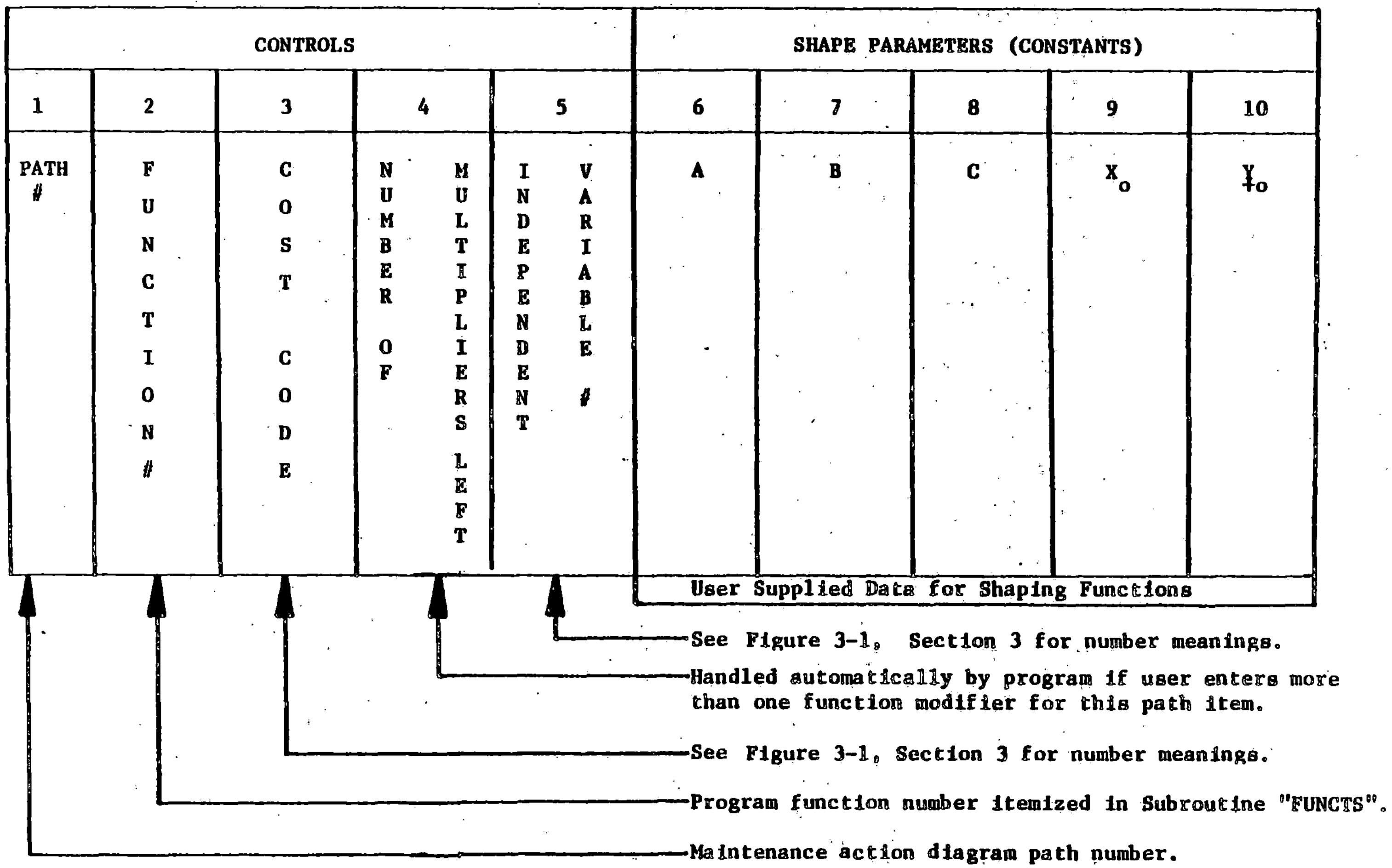
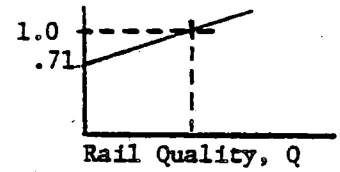


FIGURE E-1. FUNCT ARRAY SPECIFICATIONS BY COLUMN NUMBER

DESCRIPTION	FORM
Graphic Display of Function to be Used in Model to Modify Accident "Flow" on Path 17.	Accident Track Multiplier 
Linear Equation of Above Graphic Display. See Function 6 of "FUNCTS" Subroutine	$\begin{bmatrix}\text{Accident}\\ \text{Track}\\ \text{Multiplier}\end{bmatrix} = \frac{A}{B} Q + C$ A = .29, B = .1, C = .71
Input Data for Activating and Describing Function to be Used in Modifying Track Flow with Track Quality. See data in Appendix G.	Path # ... "Flow Code" 017 2 20 50 017 2 20 6 0.29 0.1 0.71 Function # ... Parameters A, B, C

Tabular Output at Beginning of Computer Run Prompting User That Function 6 is Active on Path 17 as well as Listing Shape Parameters Used.

FUNCTIONS AND SHAPE PARAMETERS ACTIVE

PATH	FUNC.	A	B	C	XO	YO
5	7	2.000 00	3.900 01	0.000-01	0.000-01	0.000-01
17	6	2.900-01	1.000-01	7.100-01	0.000-01	0.000-01

Sample Program Output "Flows" Along Path 17 With and Without Function 6 Operating Over the Ten Year Simulation Period

Time Yr	0	.5	1.0	1.5	2.0	2.5	3.0 . . . 10
Flow* No Function	8.00	6.09	5.21	4.78	4.56	4.44	4.38. . . 4.31
Flow* With Function Modifier on Quality	8.00	5.67	4.68	4.22	3.99	3.87	3.81. . . 3.74

*Component 1 of Class 1, 2, 3 Track

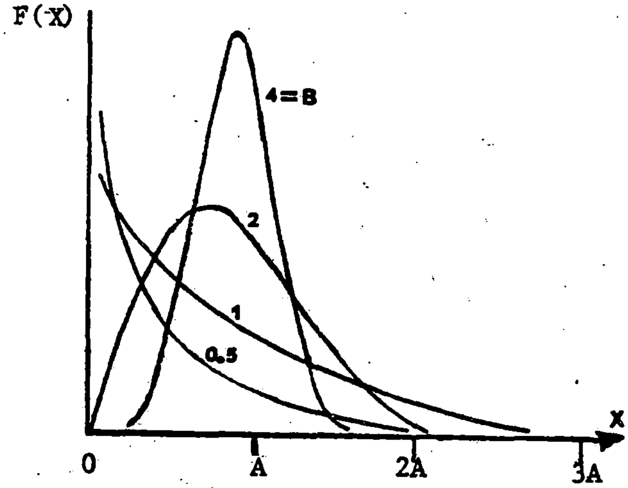
FIGURE E-2. SAMPLE APPLICATION OF A SINGLE FUNCTION TO MODIFY THE AMOUNT OF TRACK FROM TRACK ACCIDENTS BEING MAINTAINED DURING SIMULATION MAINTENANCE ACTION DIAGRAM EXAMPLE #1, PATH 17

In a similar manner, any other path may have functional "modifiers" operating on their flows or costs. One must merely generate the form of the functional dependence and activate the "modifier" function on the path component flows and/or costs known to be affected by the relation. If two or more functional "modifiers" are known to affect the path data then each is entered separately. During the simulation, then both functions will "modify" the base information carried by taking the product of the two functions.

Actually "any" number of functions may be used to modify the path data as long as the input shape parameters can be given to prescribe the relationships used in the simulation modeling.

As an example of how the shape parameters can influence the nature of the programmed family of curves consider the displays of Figure E-3. The upper plot of this figure demonstrates the curve shape obtained from programmed Equation #4 by setting parameter B equal to .5, 1, 2 and 4, respectively. The plot is made for all dummy variable values of "X" such that $0 \leq X \leq 3A$ where A is one of the parameters in programmed Equation #4.

This type of programmed formulation can be used to produce scalar outputs which may be known to depend in general on some "state variable" or "time" during the simulation. A collection of mechanical components whose rate of failure sometimes depends upon the age of the components can often be described by a distribution such as the Weibull.

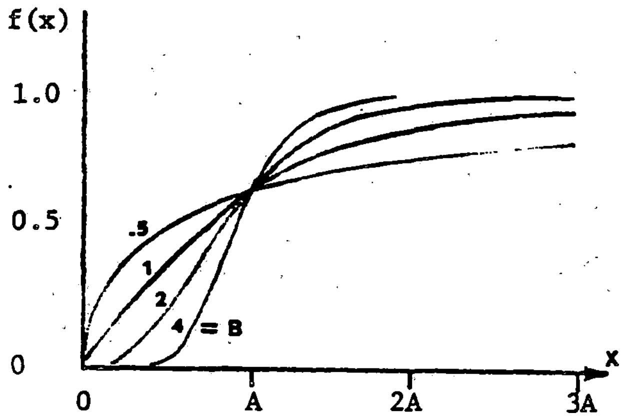


WEIBULL PROBABILITY DENSITY

Programmed Equation #4
for

B = .5, 1, 2, 4

X = 0 . . . 3A

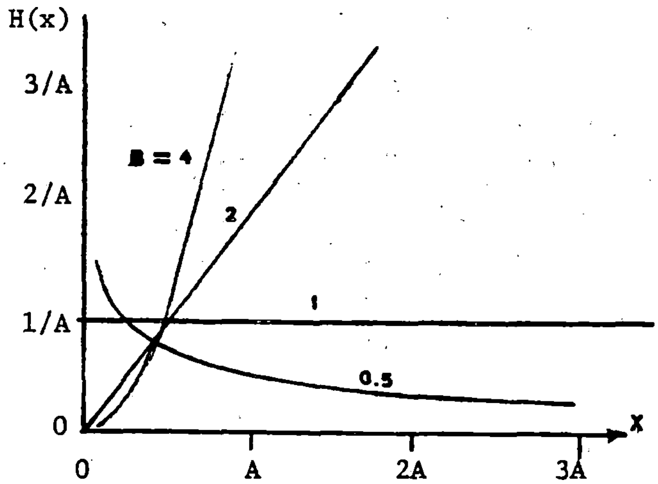


WEIBULL CUMULATIVE DISTRIBUTION

for

B = .5, 1, 2, 4

X = 0 . . . 3A



FAILURE RATE FUNCTION

H(x)

for

B = .5, 1, 2, 4

X = 0 . . . 3A

FIGURE E-3. DIFFERENT FUNCTIONAL FORMS OF WEIBULL DISTRIBUTIONS RESULTING FROM NUMERIC ADJUSTMENT OF SHAPE PARAMETERS A AND B

APPENDIX F

Computer Program Control and
Operating Data File Structuring

This appendix describes program array structuring and element by element descriptions needed for understanding their operation and use.

COMMON INTEGER ARRAYS:

PROGRAM LOCATION AND ARRAY NAME	DESCRIPTION
COMMON/I1/IERROR(10)	Used for program diagnostics. The ten elements of this vector are used in subroutine ERRORS for printing and tracing typical errors resulting from possible clerical mistakes.
COMMON/I2/IFILE(100)	A user file whose various values define various other program array lengths, as well as input/output controls. Each vector element of this file used by the program has a variable "name" used in the program. The full length of this data file with variable names and their meanings is listed at the end of this Appendix.
COMMON/I3/IFLAG(10)	Used for program diagnostics. These ten values control the printed error messages selected by subroutine TRACE. Setting IFLAG(1) = 0 will normally disable or shut this option off.
COMMON/I7/INODE(6,300)	Up to a maximum of 300 nodes. This array carries six numeric values which correspond to each of the various path branching or sum points (nodes) of the maintenance action diagram. The six columns contain the following information.

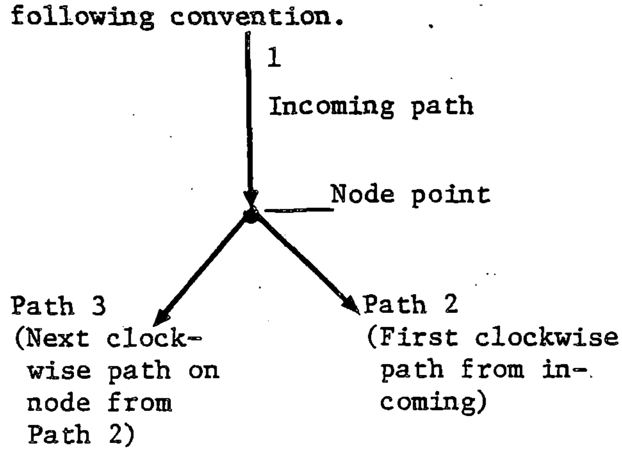
Column	Numberic Value is:
1.	Incoming path number to node.
2.	In or out path number depending upon whether path is a summing node or branch (splitting) node, respectively.

3. Outgoing path number of corresponding node.

4. Zero (0) if a sum node, One (1) if a branch node.

5. Zero (0) if split fraction of path flow based on total incoming flow. One (1) if split fraction of path flow is based on defective portion of incoming flow to the node.

6. This column only has meaning for branching nodes (one path in-two out). Action diagrams branch nodes are always referred to with the following convention.



If the numeric value in column six is zero (0) then path 2 is the reference path. If it is one (1) then path 3 is the reference path.

The reference path flow given in the input data is used at the beginning of the program execution to compute the node split fraction automatically by the program. See RNODE array values.

COMMON/I8/IARAY(15,600)

Up to fifteen (15) integer codes are allowed by this array to be attached to each path (up to 600) of the maintenance action diagram of the cost model. Not all of the fifteen have been designated. Those that have are to be associated with the meanings given in the data entry and formsetting procedure of the Appendix G data example.

Common Real Arrays

Common/R1/ACCUM(750)

This array holds the cost summary from all paths. The dimension of this array is to be at least as large as the product of three variables entered by the user in the IFILE ARRAY, i.e., IBDM*LSTCP*LSTCL. See notation in IFILE at the end of this Appendix.

Common/R2/FUNCT(10,10)

Used by the FUNCTS subroutine in all computations requiring formula type relationships involving costs or system path flows known to have explicit dependencies which can be expressed by a formula. Although this array presently has only 10 rows for demonstration purposes it should be dimensioned to have at least as many rows as there would be functional (shape parameter formula related) data lines in the user generated input cost data (See Appendix G for example). The functional shape parameters and numeric control items for this array for each column (for up to ten columns) are explained in detail in Appendix E.

Common/R3/RNODE(30,300)

This array is cleared and filled at the beginning of the program execution without user control from the flow information supplied by the user as shown in the example of Appendix G. The numeric elements of this array are the fractional path flow split values associated with each node. Thus, dimension 300 must be at least as large as the number of nodes (sum and branch) in maintenance action diagram being used in the simulation. The dimension 30 corresponds to the product of the number of track structural components (typically 5) and FRA track classes (typically 6) used in the simulation.

Common/R4/STATE(20,30)

This array is used during the simulation to store computations on the system populations (amount of each component), their rates of change, the system qualities (one for each component and class) and their rates of change. There are five (5) sets of each of these four (4) items stored for the separate Runga Kutta computations. Thus the dimension 20. The thirty (30) dimension corresponds to a separate set of 20 for each track structural component and FRA track class (5 times 6 respectively).

Common/R5/RFILE(100)

This array has two conceptual blocks of information associated with it and only a small portion of this array has been made use of. The first 50 elements were set up to carry certain real time values which could be available in common. The first three elements are simulation starting time (in years), simulation time step in years, and a final reference time point in years which stops the simulation cycle. The final time entered by the user to array element three is usually one time step less than the last time point to be simulated. Array element 4 is not used. Elements five and six give the number of time steps that must pass before an output cost table is printed. Although a full set of costs and/or flows are generated at each time step the user can set the printed output cycle separately with these variables. This condenses the amount of output that must be printed if small time steps are used in the simulation.

The last 50 elements (50 through 99) are intended as simple real value numbers which are defined to be related through the simulation state variable "time" and the fifty variable parameters listed in Figure 3-1 of Section 3.

Common/R6/WEIBL(3,30)

This array holds the user set of Weibull constants used for depicting the amount of track component degradation which can occur naturally. There are three parameters; average age, distribution shape parameter β, and characteristic life of the Weibull distribution. There is allowed a separate distribution for each track structural component and each FRA track class (5 x 6).

The Weibull distribution is general enough to describe component systems which have either an increasing or decreasing failure rate depending upon the value of β. If $\beta=1$ then the exponential distribution results, and the failure rate is constant. The Weibull distribution is used as a time rate of failure descriptive technique and has been found to be useful for describing wear out rates for large collections of components such as bearings. (See References 9 and 11.)

Common/R7/REVISN(8), SCHMGD(8), DATAFL(8)

These arrays are used in passing the program revision number, the maintenance action "schematic diagram" used, and the cost data file number used in the simulation.

The program revision number REVISN(8) is set at the very top of the program and should be renumbered if altered in the future.

Revision '6' was the final Prime Computer version and revision '8' was the final Dec 10 version. The slight differences in the two were alterations for machine compatibility.

Common/R8/RARAY(750,600)

This array is used to hold all cost and flow information associated with each path of the maintenance action diagram. For convenience the data by row number (up to 600 rows in this example) in this array corresponds to the information attached to the path of the same number in the maintenance action diagram used in the simulation. If paths in the action diagram are not numbered sequentially then some space will be wasted. Redimensioning the 600 value of this array for the minimum number of paths used in the simulation will conserve computer space.

The 750 dimension of this array can also be decreased in size to fit the simulation run if space in the computer needs to be conserved. This dimension must be at least as large as the product of program variables IBDM, LSTEL, and LSTCP. This corresponds to the minimum block of numbers (IBDM) needed for each FRA track class (LSTCL), and number of track structural components (LSTCP) used in the simulation. The IBDM minimum data block size must be at least 8 if only one cost code is required to cover costs occurring on any path. For example purposes IBDM was set at 20 and 25 for the data in Appendices G and H, respectively. This allowed for several types of costs (such as labor, materials, scrap, etc.) to be kept separately in the simulation.

Dimension Arrays

Dimension RPATH(750), IPATH(15)

RPATH along with RPATH1, RPATH2, and RPATH3 are vectors used in passing individual real number path costs and flow information between the RARAY and the various program subroutines. The 750 dimension must be at least as large as the product of program variables IBDM*LSTCL*LSTCT. The structure of this vector and its elements are displayed in Figure F-1.

IPATH, IPATH1, IPATH2, and IPATH3 are vectors used in passing integers (typically zero and one type of control indices associated with

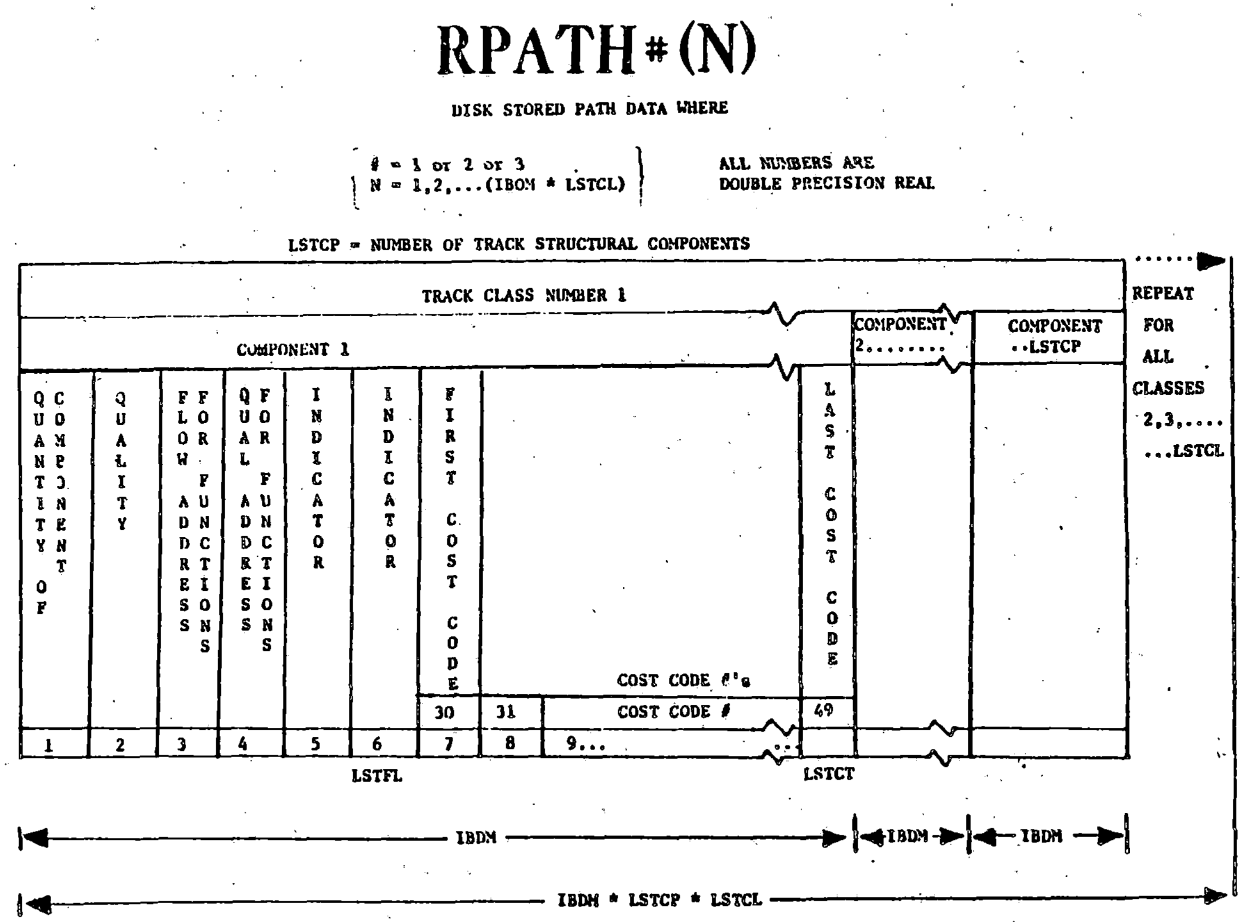


FIGURE F-1. ARRAY STORAGE STRUCTURE FOR PATH DATA BY ARRAY SUBSCRIPT NUMBER the paths) between the IARAY and the various subroutines. The 15 dimension was selected arbitrarily before the actually needed numbers of this kind of integer was fully established. Actually only eight are used during the program execution, (for all indicators I=Yes, O=No) i.e.,

IPATH(1) = The path number of the maintenance action diagram path to which the rest of the array elements are to be associated. Variable IPH.

IPATH(5) = Path type. Variable ITYPE.

IPATH(8) = Maintenance action Repair Block number in which this path resides. This provides cost outputs to be summaried by Repair Block or type. Variable IRPB.

IPATH(10)= An indicator as to whether this path contains costs. This indicator allows the program to skip the cost computations if the path has no costs associated with it. Variable ICST.

IPATH(11)= An indicator as to whether there is a net change in the simulated system population on this path. Source or sink flows such as from manufacturers or scrap yards require this information. Variable IPDT.

IPATH(12)= An indicator showing that the quality of the components on this path are altered from the labor or work applied to them. Variable IQDT.

IPATH(13)= Indicator of manufacturing path. Variable IMFG.

IPATH(14)= Indicator allowing this path to be skipped at printout time during the simulation. This allows printouts of small portions of the maintenance action diagram to be obtained. Variable IPRT.

Dimension Place(10), RUNN(10) Handle heading information for tabular outputs.

Copies of IFILE, RFILE, IFLAG, and WEIBL are listed below. These files along with the input cost data file of Appendix G are the only non-zero files needed to obtain the sample run output file shown in Appendix I.

IFILE Listing
(For Example #1 Run, Appendix I)

```
BASE MOD 2
SHAKER RESEARCH CORP.
  1   1 ISTND    FIRST NODE OF SCHEMATIC TO BEGIN COMPUTATIONS
  9   2 LSTND    LAST NODE OF SCHEMATIC TO STOP
  1   3 ISTPH    FIRST PATH & PATH DATA FILE NUMBER ON DISK
 18   4 LSTPH    LAST PATH & PATH DATA FILE NUMBER ON DISK
  1   5 ISTCP    FIRST COMPONENT NUMBER USED IN COMPUTATIONS
  2   6 LSTCP    LAST COMPONENT NUMBER USED IN COMPUTATIONS
  1   7 ISTCL    FIRST TRACK CLASS TO USE IN COMPUTATIONS
  2   8 LSTCL    LAST TRACK CLASS TO USE IN COMPUTATIONS
  6   9 ISTCT    COLUMN CONTAINING FIRST COST MINUS ONE
 13  10 LSTCT    TOTAL COST COLUMN BLOCK SIZE MINUS ONE
 15  11 IIDM     DIMENSION OF IPATH ARRAY
 80  12 IRDM     DIMENSION OF RPATH ARRAYS
 13  13 NREFP    SCHEMATIC DIAGRAM IN USE REFERENCE PATH NUMBER
  6  14 LSTFL    TOTAL FLOW COLUMN BLOCK SIZE - EQUIVALENT TO ISTCT
  1  15 IOUT     PATH  DATA WRITE TO DISK CONTROL INDICATOR
 20  16 IBDM     NUMBER OF FLOW PLUS COST COLUMNS (BLOCK SIZE)
 50  17 NCOSTS   MAXIMUM NUMBER OF DATA CODES
 44  18 LCTCD    NUMERIC VALUE OF LAST USER DEFINED COST CODE
  5  19 NPASS    STATE BLOCK REFERENCE ROW IN NORMAL
  0  20          AVAILBLE
 10  21 IUNIT    FORTRAN UNIT DEVICE NUMBER FOR OUTPUT PRINTING
 11  22 IRUNIT   FORTRAN UNIT FOR READING IN RAW DATA
  1  23 ITUNIT   FORTRAN UNIT FOR PRINTING DIAGNOSTIC STATEMENTS
  4  24 LNTHST   NUMBER OF COLUMNS IN THE STATE ARRAY
  1  25 IPRNT    INDICATOR FOR NEED OF PATH FLOW/COST PRINTOUT
  0  26          AVAILABLE
  0  27 ISPRT    INDICATOR FOR NEED OF COST/STATE PRINTOUT
  0  28 IFMTS    INDICATES COST/STATE PRINT OPTIONS
  1  29 NDOUT    INDICATES NEED FOR FINAL COST/STATE PRINTOUT
  1  30 IFMT     INDICATES FINAL PRINT OPTIONS
  1  31 N1       STATE POPULATION REFERENCE ROW
  3  32 N2       STATE QUALITY REFERENCE ROW
 17  33 N1       STATE POPULATION UPDATE ROW
 19  34 N2       STATE QUALITY UPDATE ROW
  1  35 ICOST    INDICATOR FOR NEED OF COST COMPUTATIONS
  0  36          AVAILABLE
  0  37          AVAILABLE
  1  38 NBLOCK   NUMBER OF REPAIR BLOCKS IN SYSTEM
  0  39          AVAILABLE
 44  40 IDCODE   LAST COST CODE - EQUIVALENT TO LCTCD
  1  41          RAW DATA PROCESSING INDICATOR
  1  42          MODEL RUN INDICATOR
  0  43          AVAILABLE
  0  44          AVAILABLE
  0  45          AVAILABLE
  0  46          AVAILABLE
  0  47          AVAILABLE
  0  48          AVAILABLE
  0  49          AVAILABLE
  0  50          AVAILABLE
  0  51          AVAILABLE
```

IFILE-Listing, Page 2

0	52	AVAILABLE
0	53	AVAILABLE
0	54	AVAILABLE
0	55	AVAILABLE
0	56	AVAILABLE
0	57	AVAILABLE
0	58	AVAILABLE
0	59	AVAILABLE
0	60	AVAILABLE
0	61	AVAILABLE
0	62	AVAILABLE
0	63	AVAILABLE
0	64	AVAILABLE
0	65	AVAILABLE
0	66	AVAILABLE
0	67	AVAILABLE
0	68	AVAILABLE
0	69	AVAILABLE
0	70	AVAILABLE
0	71	AVAILABLE
0	72	AVAILABLE
0	73	AVAILABLE
0	74	AVAILABLE
0	75	AVAILABLE
0	76	AVAILABLE
0	77	AVAILABLE
0	78	AVAILABLE
0	79	AVAILABLE
0	80	AVAILABLE
0	81	AVAILABLE
0	82	AVAILABLE
0	83	AVAILABLE
0	84	AVAILABLE
0	85	AVAILABLE
0	86	AVAILABLE
0	87	AVAILABLE
0	88	AVAILABLE
0	89	AVAILABLE
0	90	AVAILABLE
0	91	AVAILABLE
0	92	AVAILABLE
0	93	AVAILABLE
0	94	AVAILABLE
0	95	AVAILABLE
0	96	AVAILABLE
0	97	AVAILABLE
0	98	AVAILABLE
0	99	AVAILABLE
0	100	AVAILABLE
0		AVAILABLE

RFILE Listing
(For Example #1 Run, Appendix I)

0.0	1	STIME	STARTING TIME FOR MAIN/NORMAL LOOP
0.1	2	DTIME	TIME STEP FOR MAIN/NORMAL LOOP
1.9	3	TIMEL	FINISH TIME FOR TIME RUN
0.0	4		AVAILABLE
5.0	5		OUTPUT FREQUENCY COUNTER
5.0	6	OUTLMD	OUTPUT FREQUENCY
0.0	7		AVAILABLE
0.0	8		AVAILABLE
0.0	9		AVAILABLE
0.0	10		AVAILABLE
0.0	11		AVAILABLE
0.0	12		AVAILABLE
0.0	13		AVAILABLE
0.0	14		AVAILABLE
0.0	15		AVAILABLE
0.0	16		AVAILABLE
0.0	17		AVAILABLE
0.0	18		AVAILABLE
0.0	19		AVAILABLE
0.0	20		AVAILABLE
0.0	21		AVAILABLE
0.0	22		AVAILABLE
0.0	23		AVAILABLE
5.0	24		NORMAL OUTPUT FREQ.
5.0	25		NORMAL OUTPUT START
0.0	26		AVAILABLE
0.0	27		AVAILABLE
0.0	28		AVAILABLE
0.0	29		AVAILABLE
0.0	30		AVAILABLE
0.0	31		AVAILABLE
0.0	32		AVAILABLE
0.0	33		AVAILABLE
0.0	34		AVAILABLE
0.0	35		AVAILABLE
0.0	36		AVAILABLE
0.0	37		AVAILABLE
0.0	38		AVAILABLE
0.0	39		AVAILABLE
0.0	40		AVAILABLE
0.0	41		AVAILABLE
0.0	42		AVAILABLE
0.0	43		AVAILABLE
0.0	44		AVAILABLE
0.0	45		AVAILABLE
0.0	46		AVAILABLE
0.0	47		AVAILABLE
0.0	48		AVAILABLE
0.0	49		AVAILABLE
0.0	50		AVAILABLE
51.	51		AVAILABLE
0.0	52		AVAILABLE
53.	53		AVAILABLE
54.	54		AVAILABLE

55.	55	AVAILABLE
0.0	56	AVAILABLE
0.0	57	AVAILABLE
0.0	58	AVAILABLE
0.0	59	AVAILABLE
0.0	60	AVAILABLE
0.0	61	AVAILABLE
0.0	62	AVAILABLE
0.0	63	AVAILABLE
0.0	64	AVAILABLE
0.0	65	AVAILABLE
0.0	65	AVAILABLE
0.0	67	AVAILABLE
0.0	68	AVAILABLE
0.0	69	AVAILABLE
0.0	70	AVAILABLE
0.0	71	AVAILABLE
0.0	72	AVAILABLE
0.0	73	AVAILABLE
0.0	74	AVAILABLE
0.0	75	AVAILABLE
0.0	76	AVAILABLE
0.0	77	AVAILABLE
0.0	78	AVAILABLE
0.0	79	AVAILABLE
0.0	80	AVAILABLE
0.0	81	AVAILABLE
0.0	82	AVAILABLE
0.0	83	AVAILABLE
0.0	84	AVAILABLE
0.0	85	AVAILABLE
0.0	86	AVAILABLE
0.0	87	AVAILABLE
0.0	88	AVAILABLE
0.0	89	AVAILABLE
0.0	90	AVAILABLE
0.0	91	AVAILABLE
0.0	92	AVAILABLE
0.0	93	AVAILABLE
0.0	94	AVAILABLE
0.0	95	AVAILABLE
0.0	96	AVAILABLE
0.0	97	AVAILABLE
0.0	98	AVAILABLE
0.0	99	AVAILABLE
0.0	100	AVAILABLE
0.0	EXTRA LINE	AVAILABLE
0.0	EXTRA LINE	AVAILABLE

INODE Listing
(Example Run #1, Appendix I)

```
    SCHEMATIC DIAGRAM NUMBER :
    DATA FILE NUMBER :
13  14   1   1   0   1
 1   2  18   1   1   1
13  14  17   1   1   1
18  17   3   0   0   0
 3   5   4   1   0   0
 5   6   7   0   0   0
 7   9   8   1   1   1
 9  11  10   1   1   1
 4  11  12   0   0   0   ____________ REST OF ARRAY FILLED WITH ZEROS
```

WEIBL Listing
(For Example Run #1, Appendix I)

20.0	20.0	20.0	20.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
2.0	2.0	2.0	2.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
32.0	32.0	32.0	32.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0

NOTE: δ = 20 - fixed average life 20 yrs.

β = 2 - failure rate shape parameter

α = 32 - characteristic life years

For this example, failure rate is assumed constant for each component and given by

$$\frac{\beta}{\alpha}\left(\frac{\delta}{\alpha}\right)^{\beta-1} \cong 4\%$$

APPENDIX G

Input Cost Data Example #1
(For Simplified Track Maintenance Action Diagram #1)

This appendix contains a listing of an example input data file of costs and path flows for the example run output listed in Appendix I. Also included here is an explanation of the file coding structure so the reader may generate a different file for simulation if desired.

An input data coding structure which is compatible with either a computer file or card entry technique has been adopted. Maintenance data in this scheme are coded onto a series of cards or file lines. Each card or file line contains a part of the data required for the cost simulation model. (See listing at end of appendix for a complete data listing of the cost and flow values for example number 1 track maintenance system simulation.)

Ultimately, every data entry to the cost model program is used by the computer to generate a dollar cost for performing every maintenance operation considered. Certain data conventions have been followed in order to place the costing data in a computerized format. Figure G-1 is an overall schematic of the data card entry format assumed. Each card or data entry is read in free field format. Thus, all spaces or commas are read as numeric delimiters separating the numbers entered.

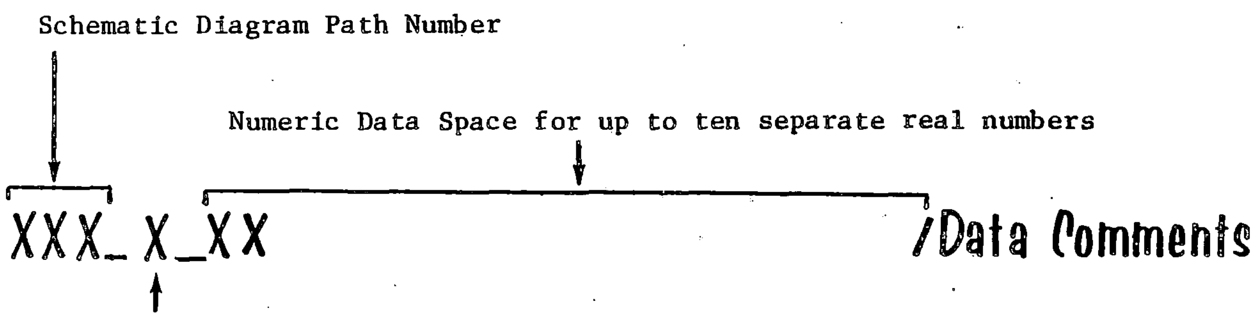
The path number is the first number on each data line. Immediately following the "path number" each data line (card) has a single digit entry (0-9) in column five. This entry refers to the kind of data that follows it on that line. Data beyond column five can be path descriptive numbers, direct path costs and flows, or function shape parameters.

Figure G-1 shows the position of the data entry codes. Codes 3 through 8 are still undefined, but numbers 0-2 and 9 have the following meanings:

0 - Refers to the fact that path information will be found in the numeric data space following this entry.

1 - Indicates that the numeric data that follow on this line comprise a direct cost or flow input set of numbers.

2 - Is used to designate that the numeric data will be function



DATA ENTRY CODE

0 = Refers to overall path information in 'numeric data space'

1 = 'Numeric Data' to be associated with cost or flow code shown

2 = 'Numeric Data' are for functional formulas (shape factors, etc.)

3-8 = Available

9 = Indicates more data on next card (line) for this data entry code.

FIGURE G-1. LAYOUT OF DATA CARD OR LINE FILE ENTRY FORMAT parameters. These shape parameters will be passed during program execution to user defined program equations for function evaluation.

3-8 - Available for definition.

9 - Indicates the numbers for this line (card) exceed the space provided on the card above it; therefore, read the next card and append.

As shown in Figure G-1 each card contains a reference "path number" which is keyed to the maintenance action diagram of the example modeled track maintenance system. In addition, each data card contains a two digit data descriptor located in columns 7 and 8. A list of the data descriptors 30 through 49 are given in Figure G-2. Codes 30 to 49 are intended for the various cost inputs. This set of 20 data subcategories 30-49 can be expanded by user selection if more are needed. This data structure allows each path cost to be entered and/or examined visually. Comments placed after the slash (/) are useful in checking over the data entries and their meanings.

Conventions have been adopted for the data to be entered in the "numeric data space" of each data card. There is a separate entry convention for each data entry code 0 through 2. The following paragraphs discuss the various numbers required for each of these three entry codes.

Entry Code 0 - Path Information

Requires a minimum of two numbers after it. Each number is to be separated by a space or a comma. These are a component number and an FRA track class number.

Entry Code 1 - Cost or Flow Information

Requires between two and five numbers. If the data code descriptor is:

20 - enter starting path component flow after it;

21 - enter starting path component quality after it;

30 through 49 - enter appropriate cost numbers.

	Data Descriptor Code	Description of Data Under This Code	UNITS OF MEASURE: First Numeric Entry Beyond Data Descriptor Code	UNITS OF MEASURE: Second Numeric Entry Beyond Data Descriptor Code	UNITS OF MEASURE: Third Numeric Entry Beyond Data Descriptor Code
PATH FLOW INFO	20	Quantity of Component	Number of Miles Per Year	(No Entry Required)	(No Entry Required)
	21	Quality of Component	Bad Miles÷Total Miles		
	22	Amount of Bad Component	Number of Miles Per Year		
	23	Amount of Good Component	Number of Miles Per Year		
	24				
	25				
	26				
	27				
	28				
	29				
COST CATEGORIES	30				
	31	Material Costs	Items Per Mile	Dollars Per Item	Multiplying Factor
	32	Equipment Costs	Hours Per Mile	Dollars Per Hour	Number of Machines
	33	Fines	Fines Per Mile	Dollars Per Incident	Number of Incidents
	34	Delay Costs	Hours Per Mile	Dollars Per Hour	Multiplying Factor
	35	Scrap Return Costs	Items Per Mile	Dollars Per Item	Multiplying Factor
	36	Contracted Costs	Contracts Per Mile	Dollars Per Contract	Number of Contracts
	37	Delivery Costs	Hours Per Mile	Dollars Per Hour	Multiplying Factor
	38	Travel and Living	Hours Per Mile	Dollars Per Hour	Multiplying Factor
	39		Not Used	Not Used	Not Used
	40	Trackmen Labor	Hours Per Mile	Dollars Per Hour	Number of Men
	41	Mechanic Labor	Hours Per Mile	Dollars Per Hour	Number of Men
	42	Machine Operator Labor 1	Hours Per Mile	Dollars Per Hour	Number of Men
	43	Machine Operator Labor 2	Hours Per Mile	Dollars Per Hour	Number of Men
	44	Foremen Labor	Hours Per Mile	Dollars Per Hour	Number of Men
	45	Supervisor Labor	Hours Per Mile	Dollars Per Hour	Number of Men
	46				
	47				
	48				
	49				

FIGURE G-2. ASSIGNED DATA DESCRIPTORS 20-49 AND THEIR UNITS OF MEASURE FOR CODING

Entry Code 2 - Function Data

Requires two data cards. The first card is not used for computation and contains a set of designator codes for reminder commentary only. The second card with an entry code 2 will carry seven numbers beyond the entry code. These seven numbers will include one designator code, one user defined function number, and up to five shape parameters for that function being activated.

Entry Code 3-8 - Available for definition.

Entry Code 9 - Append this data card entry to the one above.

Descriptive Commentary for Cost Data File for Example Run #1

As explained in Section 6.0, this run was intended to portray a hypothetical small railroad for showing some aspects of the simulation technique and methodology. The railroad and its physical plant (pertinent to this procedure) are fully described in Figure 6-1 of the main body of the report. The values displayed in that figure were contrived simply for this example run; however, the numeric magnitudes of the data selected could very well be found in railroads of this size.

Out of seven hundred miles of track a typical road of today might have the distribution shown between jointed and welded rails (750 to 250, respectively). Accident rates were chosen in a similar fashion based on a national annual average of about 2000 track incidents for the 200,000 miles of track in use, thus .01/mile. Since different accident rates can exist for each FRA track class or type of rail, separate values of .011, .02, .01 and zero accidents per mile of track were entered respectively.

Demonstration of how the track accident rate could vary with a state variable such as "quality" is implemented on only the jointed rail of FRA classes 1, 2, 3 (see Figures 6-1 and 6-2 of the report).

The following tabular commentary is intended to elucidate some of the numeric entries contained in the listing at the end of this appendix.

ANNOTATIONS TO LINE ENTRIES IN LISTING OF
"COST DATA FILE FOR EXAMPLE RUN #1"
CONTAINED AT END OF THIS APPENDIX

Line # Counted From Top	Entry	Description or Explanation	
1		Commentary line passed to run output	
2	014	Action Diagram path number ("in-use" path not actually numbered on Fig. 4-1)	
2	0	The following entries describe this path	
2	01	These entries are for jointed rail (Comp. 1)	
2	1	These entries are for track class 1	
2	0	This path has no cost	
2	0	This path does not change the system population	
2	0	This path does not change the system quality	See
2	0	This path is not from a manufacturing source	Fig.5-1
2	0	Path not in a repair block	
2	1	Skip this path on printout	
2	/ to use	Comments only	
3	014	Action Diagram Path number ("in-use" path not actually numbered on Fig. 4-1)	
3	1	Comment 1 on this	
3	20	Data descriptor code Fig. 3-1 "Quantity of Track" entered next	
3	0	Zero miles of track	
4	014	Action Diagram path number ("in-use" path not actually numbered on Fig. 4-1)	
4	1	Component 1 on this	
4	21	Data descriptor code Fig. 3-1 "Quality of Track" to be entered next	
4	0.0	Quality = 0	
5	012	Action Diagram path number (see Fig. 4-1)	
5	0	Following entries describe this path	
5	01	These entries are for jointed rail (Comp. 1)	
5	1	These entries are for track class 1	
5	1	This path has no cost	

Line # Counted From Top	Entry	Description or Explanation
6	012	Action Diagram path number (see Fig. 4-1)
6	1	Next entry will be a data descriptor code
6	21	Data descriptor code (Fig. 3-1) next entry will be starting "Quality of Track"
6	.0909	Quality of Track = .0909
7	013	Action Diagram path number (see Fig. 4-1)
7	0	Following entries describe path
7	01	Component #1 data
7	1	Track class 1
8	013	Action Diagram path number (see Fig. 4-1)
8	1	Next entry will be data descriptor code
8	20	Data descriptor code (Fig. 3-1) refers to track "quantity" on path
8	700	700 miles of this track (jointed class 1)
9	013	Action Diagram path number (see Fig. 3-1)
9	1	Next entry data descriptor code
9	21	Quality of track just entered
9	.1	Quality = .1
. . . .	. . . .	Similar entries for other types of rail and track classes modeled
19	005	Action Diagram path #5 data entries
19	0	Following data describe path
19	01	Component #1 data (jointed rail)
19	1	Track class 1
19	1	This path could have associated costs
20	005	Path #
20	1	Data descriptor next
20	20	Code indicating rail quantity
20	70	70 miles of this track sent to repair at start of simulation

Line # Counted From Top	Entry	Description or Explanation
21	005	Path #
21	2	This indicates activation data for using functions programmed
21	20	Function applied to "quantity" of track on this path
21	50	Independent variable "time" used in function (see code 50, Fig. 3-1)
22	005	Path #
22	2	Following is function input data
22	20	Function "modifies" rail quantity on this path
22	7	Data to be entered for Equation #7, see Subroutine "FUNCTS"
22	2.0	"A" parameter, See Appendix E
22	39.0	"B" parameter, see Appendix E
.	.	
.	.	
.	.	More inputs for other paths
.	.	
50	004	Path #
50	1	Data descriptor next
50	34	Data descriptor (see Fig. 3-1) costs entered next will be "delay related"
50	1.0	Hours delay per mile of track on this path (see Fig. G-2)
50	25000	Dollars per hour train delay cost
50	1.0	Multiplying factor
.	.	
.	.	
.	.	More inputs for other paths
.	.	
73	008	Path # } See Fig. 5-1
73	0	Following data describe path } See Fig. 5-1
73	02	Component # } See Fig. 5-1
73	1	Track class } See Fig. 5-1
73	1	This path has costs } See Fig. 5-1

Line # Counted From Top	Entry	Description or Explanation	
73	-1	This path reduces the system population	See Fig. 5-1
73	0	No	
73	0	Not a manufacturing path	
73	1	Associate this path with "repair" block 1	
.	.		
.	.		
.	.	More inputs for other paths	
.	.		
128	010	Path #	
128	1	Data Descriptor code next	
128	35	Code "35" relates cost data to be entered with scrap (see Fig. 3-1)	
128	1.0	Items per mile (see Fig. G-2)	
128	-10000	Scrap (negative) dollars per item removed (see Fig. G-2)	
128	1.0	Multiplier on costs (see Fig. G-2)	
.	.		
.	.		
.	.	More entries	
.	.		
221	999	Last data input truncates read operation	
221	0	"999"	

COST DATA FILE FOR EXAMPLE RUN #1

DATA FILE NUMBER : 0001

014 0 01 1 0 0 0 0 0 1	TO USE
014 1 20 0	TO USE
014 1 21 0.0	TO USE
012 0 01 1 1	TO USE
012 1 21 .0909	TO USE
013 0 01 1	IN USE
013 1 20 700	IN USE
013 1 21 .10	IN USE
013 0 02 1	IN USE
013 1 20 50	IN USE
013 1 21 .10	IN USE
013 0 01 2	IN USE
013 1 20 100	IN USE
013 1 21 .10	IN USE
013 0 02 2	IN USE
013 1 20 150	IN USE
013 1 21 .10	IN USE
005 0 01 1 1	TO REPAIR
005 1 20 70	TO REPAIR
005 2 20 50	TO REPAIR
005 2 20 7 2.0 39.0	TO REPAIR
005 1 21 1.0	TO REPAIR
005 0 02 1 1	TO REPAIR
005 1 20 5	TO REPAIR
005 1 21 1.0	TO REPAIR
005 0 01 2 1	TO REPAIR
005 1 20 10	TO REPAIR
005 1 21 1.0	TO REPAIR
005 0 02 2 1	TO REPAIR
005 1 20 15	TO REPAIR
005 1 21 1.0	TO REPAIR
002 0 01 1	RETURN TO USE
002 1 20 630	RETURN TO USE
002 1 21 0.0	RETURN TO USE
002 0 02 1	RETURN TO USE
002 1 20 45	RETURN TO USE
002 1 21 0.0	RETURN TO USE
002 0 01 2	RETURN TO USE
002 1 20 90	RETURN TO USE
002 1 21 0.0	RETURN TO USE
002 0 02 2	RETURN TO USE
002 1 20 135	RETURN TO USE
002 1 21 0.0	RETURN TO USE
003 0 01 1	PROPOSED MAINT.
003 1 20	PROPOSED MAINT.
004 0 01 1 1	POSTPONED
004 1 20 8	POSTPONED
004 1 21 1.0	POSTPONED
004 1 34 1.0 25000 1.0	POSTPONED
004 0 02 1 1	POSTPONED
004 1 20 1	POSTPONED
~~004 1 21 1.0~~	~~POSTPONED~~
004 1 34 1.0 25000 1.0	POSTPONED

```
004 0 01 2 1                             /   POSTPONED
004 1 20 1                               /   POSTPONED
004 1 21 1.0                             /   POSTPONED
004 1 34 1.0 25000 1.0                   /   POSTPONED
004 0 02 2 1                             /   POSTPONED
004 1 34 1.0 25000 1.0                   /   POSTPONED
006 0 01 1 1 1 0 1 1                     /   MFG PATH
006 1 20 70                              /   MFG PATH
006 1 31 1.0 18000 1.0                   /   MFG PATH
006 0 02 1 1 1 0 1 1                     /   MFG PATH
006 1 20 5                               /   MFG PATH
006 1 31 1.0  86000 1.0                  /   MFG PATH
006 0 01 2 1 1 0 1 1                     /   MFG PATH
006 1 20 10                              /   MFG PATH
006 1 31 1.0  45000 1.0                  /   MFG PATH
006 0 02 2 1 1 0 1 1                     /   MFG PATH
006 1 20 15                              /   MFG PATH
006 1 31 1.0  86000 1.0                  /   MFG PATH
008 0 01 1 1 -1 0 0 1                    /   TO RENEW
008 1 20 56                              /   TO RENEW
008 1 21 1.0                             /   TO RENEW
008 0 02 1 1 -1 0 0 1                    /   TO RENEW
008 1 20 4                               /   TO RENEW
008 1 21 1.0                             /   TO RENEW
008 0 01 2 1 -1 0 0 1                    /   TO RENEW
008 1 20 8                               /   TO RENEW
008 1 21 1.0                             /   TO RENEW
008 0 02 2 1 -1 0 0 1                    /   TO RENEW
008 1 20 14                              /   TO RENEW
008 1 21 1.0                             /   TO RENEW
009 0 01 1 1 0 0 0 1                     /   SEPARATE SCRAP
009 1 20 84                              /   SEPARATE SCRAP
009 1 21 .1667                           /   SEPARATE SCRAP
009 0 02 1 1 0 0 0 1                     /   SEPARATE SCRAP
009 1 20 6                               /   SEPARATE SCRAP
009 1 21 .1667                           /   SEPARATE SCRAP
009 0 01 2 1 0 0 0 1                     /   SEPARATE SCRAP
009 1 20 12                              /   SEPARATE SCRAP
009 1 21 .1667                           /   SEPARATE SCRAP
009 0 02 2 1 0 0 0 1                     /   SEPARATE SCRAP
009 1 20 16                              /   SEPARATE SCRAP
009 1 21 .1667                           /   SEPARATE SCRAP
011 0 01 1 1 0 0 0 1                     /   GOOD MTRL.
011 1 20 70                              /   GOOD MTRL.
011 0 02 1 1 0 0 0 1                     /   GOOD MTRL.
011 1 20 5                               /   GOOD MTRL.
011 0 01 2 1 0 0 0 1                     /   GOOD MTRL.
011 1 20 10                              /   GOOD MTRL.
011 0 02 2 1 0 0 0 1                     /   GOOD MTRL.
011 1 20 15                              /   GOOD MTRL.
001 0 01 1 1                             /   TO INSPECT.
001 1 20 700                             /   TO INSPECT.
001 2 20 50                              /   TO INSPECT.
001 2 20 5  2.0 2.0 1.0 6.0 5.0          /   TO INSPECT.
```

```
001 1 21 .10                                   /   TO INSPECT.
001 1 43 20.0 15.0 1.0                         /   TO INSPECT.
001 0 02 1 1                                   /   TO INSPECT.
001 1 20 50                                    /   TO INSPECT.
001 1 21 .10                                   /   TO INSPECT.
001 1 43 20.0 15.0 1.0                         /   TO INSPECT.
001 0 01 2 1                                   /   TO INSPECT.
001 1 20 100                                   /   TO INSPECT.
001 1 21 .10                                   /   TO INSPECT.
001 1 43 20.0 15.0 1.0                         /   TO INSPECT.
001 0 02 2 1                                   /   TO INSPECT.
001 1 20 150                                   /   TO INSPECT.
001 1 21 .10                                   /   TO INSPECT.
001 1 43 20.0 15.0 1.0                         /   TO INSPECT.
007 0 01 1 1 0 0 0 1                           /   SORT MTRL.
007 1 20 140                                   /   SORT MTRL.
007 1 21 .5                                    /   SORT MTRL.
007 1 30 1.0 1.0 1.0                           /   SORT MTRL.
007 1 31 1.0 11785 1.0                         /   SORT MTRL.
007 1 32 1.0 785 1.0                           /   SORT MTRL.
007 1 37 1.0 157 1.0                           /   SORT MTRL.
007 1 38 1.0 314 1.0                           /   SORT MTRL.
007 1 40 1.0 314 1.0                           /   SORT MTRL.
007 1 41 1.0 314 1.0                           /   SORT MTRL.
007 1 42 1.0 942 1.0                           /   SORT MTRL.
007 1 43 1.0 942 1.0                           /   SORT MTRL.
007 1 44 1.0 157 1.0                           /   SORT MTRL.
007 0 02 1 1 0 0 0 1                           /   SORT MTRL.
007 1 20 10                                    /   SORT MTRL.
007 1 21 .5                                    /   SORT MTRL.
007 1 30 1.0 1.0 1.0                           /   SORT MTRL.
007 1 31 1.0 10500 1.0                         /   SORT MTRL.
007 1 32 1.0 700 1.0                           /   SORT MTRL.
007 1 37 1.0 140 1.0                           /   SORT MTRL.
007 1 38 1.0 280 1.0                           /   SORT MTRL.
007 1 40 1.0 280 1.0                           /   SORT MTRL.
007 1 41 1.0 280 1.0                           /   SORT MTRL.
007 1 42 1.0 840 1.0                           /   SORT MTRL.
007 1 43 1.0 840 1.0                           /   SORT MTRL.
007 1 44 1.0 140 1.0                           /   SORT MTRL.
007 0 01 2 1 0 0 0 1                           /   SORT MTRL.
007 1 20 20                                    /   SORT MTRL.
007 1 21 .5                                    /   SORT MTRL.
007 1 30 1.0 1.0 1.0                           /   SORT MTRL.
007 1 31 1.0 20700 1.0                         /   SORT MTRL.
007 1 32 1.0 1380 1.0                          /   SORT MTRL.
007 1 37 1.0 276 1.0                           /   SORT MTRL.
007 1 38 1.0 552 1.0                           /   SORT MTRL.
007 1 40 1.0 552 1.0                           /   SORT MTRL.
007 1 41 1.0 552 1.0                           /   SORT MTRL.
007 1 43 1.0 1656 1.0                          /   SORT MTRL.
007 1 43 1.0 1656 1.0                          /   SORT MTRL.
007 1 44 1.0 276 1.0                           /   SORT MTRL
007 0 02 2 1 0 0 0 1                           /   SORT MTRL.
```

```
007 1 20 30                           /   SORT MTRL.
007 1 21 .5                           /   SORT MTRL.
007 1 30 1.0 1.0 1.0                  /   SORT MTRL.
007 1 31 1.0 1550 1.0                 /   SORT MTRL.
007 1 32 1.0 103 1.0                  /   SORT MTRL.
007 1 37 1.0 20.66 1.0                /   SORT MTRL.
007 1 38 1.0 41.33 1.0                /   SORT MTRL.
007 1 40 1.0 41.33 1.0                /   SORT MTRL.
007 1 41 1.0 41.33 1.0                /   SORT MTRL.
007 1 42 1.0 124 1.0                  /   SORT MTRL.
007 1 43 1.0 124 1.0                  /   SORT MTRL.
007 1 44 1.0 20.66 1.0                /   SORT MTRL.
010 0 01 1 1 -1 0 0 1                 /   TO SCRAP
010 1 20 14                           /   TO SCRAP
010 1 21 1.0                          /   TO SCRAP
010 1 35 1.0 -10000 1.0               /   TO SCRAP
010 0 02 1 1 -1 0 0 1                 /   TO SCRAP
010 1 20 1                            /   TO SCRAP
010 1 21 1.0                          /   TO SCRAP
010 1 35 1.0 -10000 1.0               /   TO SCRAP
010 0 01 2 1 -1 0 0 1                 /   TO SCRAP
010 1 20 2                            /   TO SCRAP
010 1 21 1.0                          /   TO SCRAP
010 1 35 1.0 -10000 1.0               /   TO SCRAP
010 0 02 2 1 -1 0 0 1                 /   TO SCRAP
010 1 20 1                            /   TO SCRAP
010 1 21 1.0                          /   TO SCRAP
010 1 35 1.0 -10000 1.0               /   TO SCRAP
015 0 01 1                            /   RETURN
015 1 20                              /   RETURN
017 0 01 1 1                          /   ACCIDENT PATH
017 1 20 8                            /   ACCIDENT PATH
017 2 20 50                           /   ACCIDENT PATH
017 2 20 6 0.29 0.1 0.71              /   ACCIDENT PATH
017 1 21 1.0                          /   ACCIDENT PATH
017 1 36 1.0 3000  1.0                /   ACCIDENT PATH
017 0 02 1 1                          /   ACCIDENT PATH
017 1 20 1                            /   ACCIDENT PATH
017 1 21 1.0                          /   ACCIDENT PATH
017 1 36 1.0 3000  1.0                /   ACCIDENT PATH
017 0 01 2 1                          /   ACCIDENT PATH
017 1 20 1                            /   ACCIDENT PATH
017 1 21 1.0                          /   ACCIDENT PATH
017 1 36 1.0 3000  1.0                /   ACCIDENT PATH
017 0 02 2 1                          /   ACCIDENT PATH
017 1 36 1.0 3000  1.0                /   ACCIDENT PATH
018 0 01 1                            /   TO REPAIR EVAL.
018 1 20 70                           /   TO REPAIR EVAL.
018 1 21 1.0                          /   TO REPAIR EVAIL.
018 0 02 1                            /   TO REPAIR EVAL.
018 1 20 5                            /   TO REPAIR EVAL.
018 1 21 1.0                          /   TO REPAIR EVAIL.
018 0 01 2                            /   TO REPAIR EVAL.
018 1 20 10                           /   TO REPAIR EVAL.
```

```
018 1 21 1.0                                /   TO REPAIR EVAIL.
018 0 02 2                                  /   TO REPAIR EVAL.
018 1 20 15                                 /   TO REPAIR EVAL.
018 1 21 1.0                                /   TO REPAIR EVAIL.
999 0                                       /   END OF DATA
```

APPENDIX H

Input Data Example #2
(For Comprehensive Track Maintenance Action - Diagram #2)

This Appendix contains file listings

IFILE	-	For Control Indices of Example Run #2
INODE	-	Maintenance Action Diagram Path Numbering Linkages
COST DATA	-	Path Starting Cost and Flow Data

IFILE

PROGRAM REVISION NUMBER :

FOR RUN EXAMPLE #2

SHAKER RESEARCH CORP.

1	1	ISTND	FIRST NODE OF SCHEMATIC TO BEGIN COMPUTATIONS
201	2	LSTND	LAST NODE OF SCHEMATIC TO STOP
1	3	ISTPH	FIRST PATH & PATH DATA FILE NUMBER ON DISK
575	4	LSTPH	LAST PATH & PATH DATA FILE NUMBER ON DISK
1	5	ISTCP	FIRST COMPONENT NUMBER USED IN COMPUTATIONS
5	6	LSTCP	LAST COMPONENT NUMBER USED IN COMPUTATIONS
1	7	ISTCL	FIRST TRACK CLASS TO USE IN COMPUTATIONS
1	8	LSTCL	LAST TRACK CLASS TO USE IN COMPUTATIONS
6	9	ISTCT	COLUMN CONTAINING FIRST COST MINUS ONE
18	10	LSTCT	TOTAL COST COLUMN BLOCK SIZE MINUS ONE
15	11	IIDM	DIMENSION OF IPATH ARRAY
150	12	IRDM	DIMENSION OF RPATH ARRAYS
1	13	NREFP	SCHEMATIC DIAGRAM IN USE REFERENCE PATH NUMBER
6	14	LSTFL	TOTAL FLOW COLUMN BLOCK SIZE - EQUIVALENT TO ISTCT
1	15	IOUT	PATH DATA WRITE TO DISK CONTROL INDICATOR
25	16	IBDM	NUMBER OF FLOW PLUS COST COLUMNS (BLOCK SIZE)
50	17	NCOSTS	MAXIMUM NUMBER OF DATA CODES
49	18	LCTCD	NUMERIC VALUE OF LAST USER DEFINED COST CODE
5	19	NPASS	STATE BLOCK REFERENCE ROW IN NORMAL
0	20		AVAILBLE
10	21	IUNIT	FORTRAN UNIT DEVICE NUMBER FOR OUTPUT PRINTING
11	22	IRUNIT	AVAILBLE
1	23	ITUNIT	AVAILBLE
5	24	LNTHST	NUMBER OF COLUMNS IN THE STATE ARRAY
1	25	IPRNT	INDICATOR FOR NEED OF PATH FLOW/COST PRINTOUT
0	26		AVAILABLE
0	27	ISPRT	INDICATOR FOR NEED OF COST/STATE PRINTOUT
0	28	IFMTS	INDICATES COST/STATE PRINT OPTIONS
1	29	NDOUT	INDICATES NEED FOR FINAL COST/STATE PRINTOUT
1	30	IFMT	INDICATES FINAL PRINT OPTIONS
1	31	N1	STATE POPULATION REFERENCE ROW
3	32	N2	STATE QUALITY REFERENCE ROW
17	33	N1	STATE POPULATION UPDATE ROW
19	34	N2	STATE QUALITY UPDATE ROW
1	35	ICOST	INDICATOR FOR NEED OF COST COMPUTATIONS
0	36		AVAILABLE
0	37		AVAILABLE
20	38	NBLOCK	NUMBER OF REPAIR BLOCKS IN SYSTEM
0	39		AVAILABLE
49	40	IDCODE	LAST COST CODE - EQUIVALENT TO LCTCD
1	41		RAW DATA PROCESSING INDICATOR
1	42		MODEL RUN INDICATOR
0	43		AVAILABLE
0	44		AVAILABLE
0	45		AVAILABLE
0	46		AVAILABLE
0	47		AVAILABLE
0	48		AVAILABLE
0	49		AVAILABLE
0	50		AVAILABLE
0	51		AVAILABLE
.	.		
.	.		
.	.		
0	100		AVAILABLE

INODE FILE

SCHEMATIC DIAGRAM NUMBER : 0002
DATA FILE NUMBER :

1	2	10	1	0	1
1	2	12	1	0	1
1	2	14	1	0	1
1	2	16	1	0	1
1	2	18	1	0	1
1	2	20	1	0	1
1	2	22	1	0	1
1	2	24	1	0	1
1	2	26	1	0	1
1	2	28	1	0	1
1	2	30	1	0	1
10	50	11	1	1	1
12	48	13	1	1	1
30	32	31	1	1	1
16	44	17	1	1	1
18	42	19	1	1	1
20	40	21	1	1	1
22	38	23	1	1	1
24	36	25	1	1	1
26	34	27	1	1	1
28	33	29	1	1	1
27	29	35	0	0	0
25	35	37	0	0	0
23	37	39	0	0	0
21	39	41	0	0	0
19	41	43	0	0	0
17	43	45	0	0	0
14	45	46	0	0	0
46	15	52	1	1	0
11	13	47	0	0	0
47	15	49	0	0	0
49	61	300	1	1	0
61	62	121	1	0	1
121	171	151	1	0	0
201	171	261	0	0	0
261	371	401	1	1	1
371	461	431	1	1	1
62	63	122	1	0	1
122	172	152	1	0	0
172	202	262	0	0	0
262	372	402	1	1	1
372	462	432	1	1	1
462	461	512	0	0	0
63	64	123	1	0	1
123	173	153	1	0	0
203	173	263	0	0	0
263	373	403	1	1	1
373	463	433	1	1	1
463	512	513	0	0	0
64	65	124	1	0	1
124	174	154	1	0	0
204	174	264	0	0	0

264	374	404	1	1	1
374	464	434	1	1	1
464	513	514	0	0	0
65	66	125	1	0	1
125	175	155	1	0	0
205	175	265	0	0	0
265	375	405	1	0	0
375	465	435	1	0	1
465	514	515	0	0	0
66	67	126	1	0	1
126	176	156	1	0	0
206	176	266	0	0	0
266	376	406	1	0	0
376	466	436	1	0	1
466	515	516	0	0	0
67	68	127	1	0	1
127	177	157	1	0	0
207	177	277	0	0	0
277	377	407	1	0	0
377	477	437	1	0	1
477	516	517	0	0	0
68	69	128	1	0	1
128	178	158	1	0	0
208	178	278	0	0	0
278	378	408	1	0	0
378	478	438	1	1	1
478	517	518	0	0	0
69	70	129	1	0	1
129	179	159	1	0	0
209	179	279	0	0	0
279	379	409	1	0	0
379	479	439	1	1	1
479	518	519	0	0	0
70	71	130	1	0	1
130	180	160	1	0	0
210	180	280	0	0	0
280	380	410	1	0	0
380	480	440	1	1	1
480	519	520	0	0	0
71	72	131	1	0	1
131	181	161	1	0	0
211	181	281	0	0	0
281	381	411	1	0	0
381	481	441	1	1	1
481	520	521	0	0	0
72	73	92	1	0	1
92	31	132	0	0	0
132	182	162	1	0	0
212	182	282	0	0	0
282	382	412	1	0	0
382	482	442	1	1	1
482	521	522	0	0	0
73	74	133	1	0	1
133	183	163	1	0	0

213	183	283	0	0	0
283	383	413	1	0	0
383	483	443	1	1	1
483	522	523	0	0	0
74	75	134	1	0	1
134	184	164	1	0	0
214	184	284	0	0	0
284	384	414	1	0	0
384	484	444	1	1	1
484	523	524	0	0	0
75	76	135	1	0	1
135	185	165	1	0	0
215	185	285	0	0	0
285	385	415	1	0	0
385	485	445	1	1	1
485	524	525	0	0	0
76	77	136	1	0	1
136	186	166	1	0	0
216	186	286	0	0	0
286	386	416	1	0	0
386	486	446	1	1	1
486	525	526	0	0	0
77	78	137	1	0	1
137	187	167	1	0	0
217	187	287	0	0	0
287	387	417	1	0	0
387	487	447	1	1	1
487	526	527	0	0	0
78	79	138	1	0	1
138	188	168	1	0	0
218	188	288	0	0	0
288	388	418	1	0	0
388	488	448	1	1	1
488	527	528	0	0	0
79	80	139	1	0	1
139	189	169	1	0	0
219	189	289	0	0	0
289	389	419	1	0	0
389	489	449	1	1	1
489	528	529	0	0	0
80	81	140	1	0	1
140	190	170	1	0	0
220	190	290	0	0	0
290	390	420	1	0	0
390	490	450	1	1	1
490	529	530	0	0	0
170	169	258	0	0	0
258	168	257	0	0	0
257	167	256	0	0	0
256	166	255	0	0	0
255	165	254	0	0	0
254	164	253	0	0	0
253	163	252	0	0	0
252	162	251	0	0	0

251	161	250	0	0	0
250	160	249	0	0	0
249	159	248	0	0	0
248	158	247	0	0	0
247	157	246	0	0	0
246	156	245	0	0	0
245	155	244	0	0	0
244	154	243	0	0	0
243	153	242	0	0	0
242	152	241	0	0	0
241	151	240	0	0	0
419	420	333	0	0	0
418	333	332	0	0	0
417	332	331	0	0	0
416	331	330	0	0	0
415	330	329	0	0	0
414	329	328	0	0	0
413	328	327	0	0	0
412	327	326	0	0	0
411	326	325	0	0	0
410	325	324	0	0	0
409	324	323	0	0	0
408	323	322	0	0	0
407	322	321	0	0	0
406	321	320	0	0	0
405	320	319	0	0	0
404	319	318	0	0	0
403	318	317	0	0	0
402	317	316	0	0	0
401	316	315	0	0	0
300	240	301	0	0	0
301	302	303	1	1	1
303	304	305	1	1	1
305	306	307	1	1	1
307	308	309	1	1	1
315	304	310	0	0	0
310	311	312	1	1	1
312	313	314	1	1	1
313	311	315	0	0	0
309	302	333	0	0	0
333	530	340	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

COST DATA

```
     DATA FILE                   0002
171 0 01 1 1 0 0 0 1                  / 1 NEW RAIL
171 1 20 2000                         / 1 NEW RAIL
171 1 21 1.0                          / 1 NEW RAIL
201 0 01 1 1 1 0 1 1                  / 1 MFGS NEW RAIL
201 1 20 2000                         / 1 NEW RAIL
201 1 31 478720 .125 1                / 1 RAIL MATERIAL RF.3
201 1 37 17.9 12.5  7                 / 1 RAIL DELIVERY RF.4
261 0 01 1 1 0 0 0 1                  / 1 PATH INFO NEW RAIL
261 1 20 4000                         / 1 NEW RAIL
261 1 21 0.5                          / 1 NEW RAIL
261 1 37 1.0  70.71  1                / 1 EQUIP DELIVERY RF.4
261 1 38 4.1  17.5 5.7                / 1 T & L RF.4
261 1 42 10.9 12.5  1                 / 1 CRIB AND BROOM RF.1
261 1 32 10.9 16.27 1                 / 1 CRIB AND BROOM RF.4
261 1 43 10.9 15.00 6                 / 1 PULL SPIKES
261 1 32 10.9 1.67  6                 / 1 PULL SPIKES RF.4
261 1 42 10.9 12.50 2                 / 1 GAUGE THREADER
261 1 32 10.9 9.31  2                 / 1 GAUGE THREADER RF.4
261 1 40 10.9  7.50 1                 / 1 LIFT ANC & PLTS RF.6
261 1 40 10.9  7.50 1                 / 1 PLUG & CREOSOTE
261 1 31 24264 .0214 1                / 1 PLUG & CREOS 8/TIE RF.4
261 1 42 10.9 12.50 4                 / 1 CRIBBING MACH
261 1 32 10.9 16.67 1                 / 1 CRIBING MACH
261 1 42 10.9 12.50 2                 / 1 ADZER FACE TIES
261 1 32 10.9 16.27 1                 / 1 ADZER FACE TIES
261 1 42 10.9 12.50 2                 / 1 TIE PLATE LINER
261 1 31 6066 4.52  1                 / 1 TIE PLATE LINER RF.3/4
261 1 32 10.9 13.72 1                 / 1 TIE PLATE LINER
261 1 42 10.9 12.50 1                 / 1 PLACE NEW RAIL
261 1 32 10.9 9.31  2                 / 1 PLACE NEW RAIL
261 1 42 10.9 12.50 1                 / 1 HEAT RAIL
261 1 32 10.9 7.25  1                 / 1 HEAT RAIL RF.4
261 1 31 10.9 .20   1                 / 1 HEAT RAIL
261 1 42 10.9 12.50 6                 / 1 LINER SPIKER
261 1 32 10.9 16.27 1                 / 1 LINER SPIKER
261 1 31 27297 .25  1                 / 1 LINER SPIKER 89/TIE RF.6
261 1 42 10.9 12.50 1                 / 1 TAMPER
261 1 32 10.9 13.72 1                 / 1 TAMPER RF.4
261 1 43 10.9 15.00 4                 / 1 ANCHOR MACH
261 1 32 10.9 13.72 4                 / 1 ANCHOR MACH RF.4
261 1 31 1625 3.0   1                 / 1 ANCHOR MACH RF.5
261 1 40 10.9 7.50  1                 / 1 GREASE MACH
261 1 32 10.9 5.0   1                 / 1 GREASE MACH
261 1 41 10.9 10.00 1                 / 1 RAIL QUENCH
261 1 32 10.9 7.25  1                 / 1 RAIL QUENCH
261 1 41 10.9 10.00 4                 / 1 POWER DRILL RF.4
261 1 32 10.9 16.27 1                 / 1 POWER DRILL
261 1 43 10.9 15.00 2                 / 1 WELD MACHINE
261 1 32 10.9 9.82 1                  / 1 WELD MACHINE RF.4
261 1 42 10.9 12.50 2                 / 1 HEAVY CRIB
261 1 32 10.9 16.27 2                 / 1 HEAVY CRIB RF.4
261 1 31 2060 1 1                     / 1 OTHR TRK MATL RF.4
261 1 45 10.9 20.00 2                 / 1 SUPERVISORS RF.7
```

```
261 1 45 10.9 20.00 1              / 1 ASST SUPER RF.7
261 1 44 10.9 17.50 1              / 1 FORMAN RF.7
261 1 44 10.9 17.50 2              / 1 ASST FORMAN RF.7
261 1 43 10.9  7.50 7              / 1 LABORS RF.7
401 0 01 1 1 0 0 0 1               / 1 PATH INFO
401 1 20 1800                      / 1 TO RENEW RF.4
401 1 21 1.0                       / 1 TO RENEW RF.4
401 1 37 17.9 12.5 7               / 1 TO RENEW RF.1
371 0 01 1 1 0 0 0 1               / 1 TO SCRAP INSP
371 1 20 2200                      / 1 TO SCRAP INSP
371 1 21 0.09091                   / 1 TO SCRAP INSP
431 0 01 1 1 1 0 0 1               / 1 PATH INFO
431 1 20 200                       / 1 TO SCRAP RF.4
431 1 37 .895 12.5 -7              / 1 TO SCRAP
172 0 01 1 1 0 0 0 2               / 2 NEW RAIL
172 1 20 1000                      / 2 NEW RAIL
172 1 21 1.0                       / 2 NEW RAIL
202 0 01 1 1 1 0 1 2               / 2 NEW RAIL
202 1 20 1000                      / 2 NEW RAIL
202 1 37 17.9 12.5 7               / 2 RAIL DELIVERY RF.1
202 1 31 478720 .125 1             / 2 MFGS RAIL RF.3
262 0 01 1 1 0 0 0 1               / 2 NEW RAIL
262 1 20 2000                      / 2 NEW RAIL
262 1 21 0.5                       / 2 NEW RAIL
262 1 37 1.0  70.71  1             / 2 EQUIPMENT DELIVERY RF.4
262 1 38 4.1  17.5   5.7           / 2 T & L RF.4
262 1 43 10.9   15.00  6           / 2 PULL SPIKES
262 1 32 10.9  1.67  6             / 2 PULL SPIKES RF.4
262 1 40 10.9  7.50   1            / 2 REMOVE ANCHORS
262 1 42 10.9  12.50  2            / 2 LIFT OUT OLD RAIL
262 1 32 10.9  9.31  2             / 2 LIFT OUT OLD RAIL RF.4
262 1 40 10.9  7.50  1             / 2 LIFT TIE PLATES
262 1 40 10.9  7.50  1             / 2 DRIVE IN BROKEN SPIKES
262 1 40 10.9  7.50  1             / 2 INSERT WOODEN PLUGS
262 1 31 24264 .0214 1             / 2 INSERT WOODEN PLUGS RF.4
262 1 42 10.9 12.50  2             / 2 FACE OFF TIE
262 1 32 10.9 16.27 1              / 2 FACE OFF TIE
262 1 42 10.9 12.5 1               / 2 BROOM OFF TIES RF.1
262 1 32 10.9 16.27 1              / 2 BROOM OFF TIES RF.4
262 1 42 10.9 12.50 4              / 2 NEW SPIKES/ANCHORS/PLATES RF.6
262 1 31 27297 .25 1               / 2 NEW SPIKES/ANCHORS/PLATES
262 1 31 1625 3.0 1                / 2 NEW ANCHORS RF.5
262 1 31 3033 4.52 1               / 2 NEW PLATES RF.4
262 1 42 10.9 12.50 6              / 2 BEGIN RAIL SET IN
262 1 32 10.9 9.31 2               / 2 BEGIN RAIL SET IN
262 1 40 10.9 7.50 4               / 2 CONTINUE SET-IN
262 1 32 10.9 9.31 1               / 2 CONTINUE SET-IN
262 1 42 10.9 12.50 2              / 2 GAUGE LINER
262 1 32 10.9 16.27 1              / 2 GAUGE LINER
262 1 43 10.9 15.00 4              / 2 ANCHOR RAILS
262 1 32 10.9 13.72 4              / 2 ANCHOR RAILS RF.4
262 1 42 10.9 12.50 6              / 2 END MATCHING PROCESS
262 1 32 10.9 9.31 2               / 2 END MATCHING PROCESS
262 1 42 10.9 12.50 4              / 2 FOLLOW UP SPIKING
```

```
262 1 32 10.9 1.67 4            / 2 FOLLOW UP SPIKING
262 1 43 10.9 15.00 2           / 2 THERMITE WELD
262 1 32 10.9 9.82 1            / 2 THERMITE WELD
262 1 31 2060 1 1               / 2 OTHR TRK MATL RF.4
262 1 45 10.9 20.00 2           / 2 SUPERVISORS RF.7
262 1 45 10.9 20.00 1           / 2 ASST SUPER RF.7
262 1 44 10.9 17.50 1           / 2 FORMAN RF.7
262 1 44 10.9 17.50 2           / 2 ASST FORMAN RF.7
262 1 43 10.9 7.50 10           / 2 LABORS RF.7
402 0 01 1 1 0 0 0 2            / 2 PATH INFO
402 1 20 900                    / 2 TO RENEW RF.4
402 1 21 1.0                    / 2 TO RENEW RF.4
402 1 37 17.9 12.5 7            / 2 TO RENEW RF.1
372 0 01 1 1 0 0 0 2            / 2 TO SCRAP
372 1 20 1100                   / 2 TO SCRAP
372 1 21 0.0909                 / 2 TO SCRAP
432 0 01 1 1 1 0 0 2            / 2 PATH INFO
432 1 20 100                    / 2 TO SCRAP RF.4
432 1 21 1.0                    / 2 SCRAP
432 1 37 .895 12.5 -7           / 2 TO SCRAP
173 0 01 1 1 0 0 0 3            / 3 PATH INFO
173 1 20 1000                   / 3 DELAYS
173 1 21 1.0                    / 3 DELAYS
203 0 01 1 1 1 0 1 3            / 3 PATH INFO
203 1 20 1000                   / 3 RAIL TO SITE
203 1 37 17.9 12.5 7            / 3 RAIL TO SITE
203 1 31 478720 .125 1          / 3 RAIL TO SITE RF.3
263 0 01 1 1 0 0 0 3            / 3 PATH INFO
263 1 20 2000                   / 3 TRAVEL TO SITE
263 1 21 0.5                    / 3 TRAVEL TO SITE
263 1 37 1.0 70.71 1            / 3 TRAVEL TO SITE RF.4
263 1 38 4.1 17.5 5.7           / 3 TRAVEL TO SITE RF.4
263 1 43 10.9 15.00 6           / 3 PULL SPIKES/ANCHOR RF.1
263 1 32 10.9 1.67 6            / 3 PULL SPIKES/ANCHOR RF.4
263 1 42 10.9 12.50 2           / 3 LIFT OUT OLD RAIL
263 1 32 10.9 9.31 2            / 3 LIFT OUT OLD RAIL RF.4
263 1 42 10.9 12.5 4            / 3 LAYOUT SPIKES/ANCHORS
263 1 42 10.9 12.50 6           / 3 SET RAIL
263 1 32 10.9 9.31 2            / 3 SET RAIL RF.4
263 1 40 10.9 7.50 4            / 3 CONTINUE SET-IN
263 1 32 10.9 9.31 2            / 3 CONTINUE SET-IN
263 1 42 10.9 12.50 6           / 3 SPIKE DOWN RAIL
263 1 32 10.9 16.27 1           / 3 SPIKE DOWN RAIL
263 1 31 27297 .25 1            / 3 SPIKE DOWN RAIL RF.6
263 1 43 10.9 15.00 4           / 3 ANCHOR RAIL
263 1 43 10.9 13.72 4           / 3 ANCHOR RAIL RF.4
263 1 31 1625 3.0 1             / 3 ANCHOR RAIL RF.5
263 1 42 10.9 12.50 2           / 3 MATCH END PROCESS
263 1 32 10.9 9.31 2            / 3 MATCH END PROCESS
263 1 43 10.9 15.00 2           / 3 THERMITE WELD
263 1 32 10.9 9.82 1            / 3 THERMITE WELD
263 1 42 10.9 12.50 4           / 3 FOLLOW UP SPIKING
263 1 32 10.9 1.67 4            / 3 FOLLOW UP SPIKING
263 1 31 2060 1 1               / 3 OTHR TRK MATL
```

```
263 1 45 10.9 20.00 2          / 3 SUPERVISORS
263 1 45 10.9 20.00 1          / 3 ASST SUPER
263 1 44 10.9 17.50 1          / 3 FORMAN
263 1 44 10.9 17.50 2          / 3 ASST FORMAN
263 1 43 10.9 7.50  9          / 3 LABORS
403 0 01 1 1 0 0 0 3           / 3 PATH INFO
403 1 20 900                   / 3 RENEW RF.4
403 1 37 17.9 12.5 7           / 3 RENEW
373 0 01 1 1 0 0 0 3           / 3 TO SCRAP PATH
373 1 20 1100                  / 3 TO SCRAP PATH
373 1 21 0.0909                / 3 TO SCRAP PATH
433 0 01 1 1 1 0 0 3           / 3 PATH SCRAP
433 1 20 100.0                 / 3 SCRAP RF.4
433 1 37 .895 12.50 -7         / 3 SCRAP
174 0 01 1 1 0 0 0 4           / 4 JOINTED RAIL
174 1 20 1000                  / 4 JOINTED MILES
204 0 01 1 1 1 0 1 4           / 4 JOINTED RAIL
204 1 20 1000                  / 4 RAIL TO SITE
204 1 37 17.9 12.5 7           / 4 RAIL TO SITE RF.1
204 1 31 478720 .125 1         / 4 RAIL TO SITE RF.3
264 0 01 1 1 0 0 0 4           / 4 JOINTERD RAIL
264 1 20 2000                  / 4 REDISTRIBUTE
264 1 21 0.5                   / 4 REDISTRIBUTE
264 1 40 135 7.50 10           / 4 PULL ONE SECTION RF.6
264 1 32 135 16.27 1           / 4 PULL ONE SECTION
264 1 42 135 12.50 10          / 4 INSERT NEW RAIL
264 1 32 135 16.27 1           / 4 INSERT NEW RAIL
264 1 31 9260 .25 1            / 4 SPIKES RF.6
264 1 31 400 3.00 1            / 4 ANCHOR
264 1 31 541 17.00 1           / 4 BARS
264 1 31 3033 4.52 1           / 4 PLATES
264 1 41 45 10.00 1            / 4 REPLACE SIGNAL RF.6
264 1 31 3600 1 1              / 4 OTHER TRACK MATL
264 1 44 135 17.50 1           / 4 FORMAN RF.6
404 0 01 1 1 0 0 0 4           / 4 TO RENEW
404 1 20 900                   / 4 TO RENEW RF.4
404 1 21 1.0                   / 4 TO RENEW RF.4
404 1 37 17.9 12.5 7           / 4 TO RENEW RF.1
374 0 01 1 1 0 0 0 4           / 4 TO SCRAP PATH
374 1 20 1100                  / 4 TO SCRAP PATH
374 1 21 0.09091               / 4 TO SCRAP PATH
434 0 01 1 1 1 0 0 4           / 4 TO SCRAP
434 1 20 100                   / 4 TO SCRAPRF.4
434 1 37 4.48 12.50 -7         / 4 TO SCRAP
175 0 01 1 1 0 0 0 5           / 5 TRANSPOSE RAIL
175 1 20 1000                  / 5 TRANSPOSE RAIL
175 1 21 1.0                   / 5 TRANSPOSE RAIL
205 0 01 1 1 1 0 0 5           / 5 MFG NEW RAIL
205 1 20 0                     / 5 MFG SOURCE
265 0 01 1 1 0 0 0 5           / 5 TRANSPOSE RAIL
265 1 20 1000                  / 5 TRAVEL TO SITE
265 1 40 135 7.50 5            / 5 LIFT OUT RAIL RF.6
265 1 32 135 16.27 1           / 5 LIFT OUT RAIL
265 1 42 135 12.50 5           / 5 RESET RAIL SECTIONS
```

```
265 1 31 9260 .25 1          / 5 RESET RAIL SECTIONS RF.6
405 0 01 1 1 0 0 0 5         / 5 PATH INFO
405 1 20 0                   / 5 TO RENEW
375 0 01 1 1 0 1 0 5         / 5 PATH INFO
375 1 20 1000                / 5 TO SCRAP PATH
375 1 21 1.0                 / 5 TO SCRAP PATH
435 0 01 1 1 0 0 0 5         / 5 PATH INFO
435 1 20 0                   / 5 TO SCRAP
176 0 01 1 1 0 0 0 6         / 6 TRANSPOSE RAIL
176 1 20 1000                / 6 TRANSPOSE RAIL
176 1 21 1.0                 / 6 TRANSPOSE RAIL
206 0 01 1 1 1 0 0 6         / 6 TRANSPOSE RAIL
206 1 20 0                   / 6 MFG SOURCE
266 0 01 1 1 0 0 0 6         / 6 TRANSPOSE RAIL
266 1 20 1000                / 6 TRAVEL TO SITE
266 1 40 135 7.50 5          / 6 LIFT OUT ONE SIDE RF.6
266 1 32 135 16.27 1         / 6 LIFT OUT ONE SIDE
266 1 42 135 12.50 5         / 6 INSERT DUMMY
266 1 40 135 7.50 5          / 6 PULL SECOND SIDE
266 1 32 135 16.27 1         / 6 PULL SECOND SIDE
266 1 42 135 12.50 5         / 6 INSERT 1ST SECTION
266 1 40 135 7.50 5          / 6 PULL DUMMY
266 1 32 135 16.27 1         / 6 PULL DUMMY
266 1 42 135 12.50 5         / 6 RESET SECOND
266 1 31 9260 .25 1          / 6 SPIKE & ANCHOR RF. 6
406 0 01 1 1 0 0 0 6         / 6 PATH INFO
406 1 20 0                   / 6 TO RENEW
376 0 01 1 1 0 1 0 5         / 6 TO SCRAP PATH
376 1 20 1000                / 6 TO SCRAP PATH
376 1 21 1.0                 / 6 TO SCRAP PATH
436 0 01 1 1 0 0 0 6         / 6 PATH INFO
436 1 20 0                   / 6 TO SCRAP
177 0 01 1 1 0 0 0 7         / 7 TRACK PANEL
177 1 20 0                   / 7 TRACK PANEL
207 0 01 1 1 1 0 1 7         / 7 MFG SOURCE
207 1 20 0                   / 7 DROP PANEL AT SITE
277 0 01 1 1 0 0 0 6         / 7 TRACK PANEL
277 1 20 0                   / 7 TRACK PANEL
407 0 01 1 1 0 0 0 7         / 7 PATH INFO
407 1 20 0                   / 7 TO RENEW
437 0 01 1 1 0 0 0 7         / 7 PATH INFO
437 1 20 0                   / 7 TO SCRAP
178 0 05 1 1 0 0 0 8         / 8 PATH INFO
178 1 20 1000                / 8 MILES SURFACED
208 0 05 1 1 0 0 1 8         / 8 PATH INFO
208 1 20 1000                / 8 MILES SURFACED
208 1 31 370 13.50 1         / 8 NEW BALLAST RF.6
278 0 05 1 1 0 0 0 8         / 8 PATH INFO
278 1 20 2000                / 8 MILES SURFACED
278 1 32 6.6 16.58 1         / 8 REGULATE BALLAST RF.4
278 1 42 6.6 12.50 14        / 8 REGULATE BALLAST RF.6
408 0 05 1 1 0 0 0 8         / 8 PATH INFO
408 1 20 0                   / 8 TO RENEW
378 0 05 1 1 0 0 0 8         / 8 TO SCRAP PATH
```

```
378 1 20 2000                    / 8 TO SCRAP PATH
378 1 21 0.5                     / 8 TO SCRAP PATH
438 0 05 1 1 0 0 0 8             / 8 PATH INFO
438 1 20 1000                    / 8 TO SCRAP
438 1 21 1.0                     / 8 TO SCRAP
478 0 05 1 1 0 0 0 8             / 8 RETURN
478 1 20 1000                    / 8 RETURN
478 1 21 1.0                     / 8 RETURN
179 0 05 1 1 0 0 0 9             / 9 PATH INFO
179 1 20 1000                    / 9 MILES SMOOTHED
209 0 05 1 1 1 0 1 9             / 9 PATH INFO
209 1 20 1000                    / 9 MILES SMOOTHED
279 0 05 1 1 0 0 0 9             / 9 PATH INFO
279 1 20 2000                    / 9 MILES SMOOTHED
279 1 32 6.6 9.32 1              / 9 REGULATOR RF.4
279 1 42 6.6 12.50 5             / 9 SMOOTH BALLAST RF.6
409 0 05 1 1 0 0 0 9             / 9 PATH INFO
409 1 20 0                       / 9 TO RENEW
379 0 05 1 1 0 0 0 9             / 9 TO SCRAP PATH
379 1 20 2000                    / 9 TO SCRAP PATH
379 1 21 0.5                     / 9 TO SCRAP PATH
439 0 05 1 1 0 0 0 9             / 9 PATH INFO
439 1 20 1000                    / 9 TO SCRAP
439 1 21 1.0                     / 9 TO SCRAP
479 0 05 1 1 0 0 0 9             / 9 RETURN
479 1 20 1000                    / 9 RETURN
180 0 03 1 1 0 0 0 10            /10 PATH INFO
180 1 20 1000                    /10 TIMBER & SURFACE MIL
180 1 21 1.0                     /10 TIMBER & SURFACE MIL
180 0 05 1 1 0 0 0 10            /10 TIMBER & SURFACE MILES
180 1 20 1000                    /10 TIMBER & SURFACE MILES
180 1 21 1.0                     /10 TIMBER & SURFACE MILES
210 0 03 1 1 1 0 1 10            /10 PATH INFO
210 1 20 1000                    /10 NEW TIES
210 1 31 370 13.50 1             /10 NEW BALLAST RF.6
210 1 31 1000 15.00 1            /10 NEW TIES
210 1 37 9.09 110 1              /10 DELIVER TIES RF.6
280 0 03 1 1 0 0 0 10            /10 PATH INFO
280 1 20 2000                    /10 TRAVEL TO SITE
280 1 37 1 83.33 1               /10 TRAVEL TO SITE
280 1 38 4.1 17.50 2.2           /10 TRAVEL TO SITE RF.4
280 1 42 11.1 12.50 1            /10 TIE BROOM & CRIB
280 1 32 11.1 16.27 1            /10 TIE BROOM & CRIB
280 1 40 11.1 7.50 2             /10 PULL SPIKES
280 1 32 11.1 1.67 2             /10 PULL SPIKES
280 1 40 11.1 7.50 2             /10 REMOVE ANCHOR
280 1 42 11.1 12.50 1            /10 REMOVE TIES
280 1 32 11.1 5.01 1             /10 REMOVE TIES
280 1 43 11.1 15.00 1            /10 RESET NEW TIES
280 1 32 11.1 12.48 1            /10 RESET NEW TIES
280 1 41 11.1 10.00 2            /10 SPIKE & ANCHOR
280 1 32 11.1 2.78 1             /10 SPIKE & ANCHOR
280 1 31 2000 7.77 1             /10 SPIKE & ANCHOR
280 1 43 11.1 15.00 1            /10 REGULATE BALLAST
```

```
280 1 32 11.1 9.32 1                  /10 REGULATE BALLAST
280 1 43 11.1 15.00 1                 /10 TAMP BALLAST
280 1 32 11.1 7.26 1                  /10 TAMP BALLAST
280 1 42 11.1 12.50 2                 /10 ALIGN TRACK
280 1 32 11.1 16.27 1                 /10 ALIGN TRACK
280 1 42 11.1 12.50 1                 /10 BROOM & SWEEP
280 1 32 11.1 16.27 1                 /10 BROOM & SWEEP
280 0 05 1 1 0 0 0 10                 /10
280 1 20 2000                         /10
410 0 03 1 1 0 0 0 10                 /10 PATH INFO
410 1 20 0                            /10 TO RENEW
410 0 05 1 1 0 0 0 10                 /10 PATH INFO
410 1 20 0                            /10 TO RENEW
380 0 03 1 1 0 0 0 10                 /10 PATH INFO
380 1 20 2000                         /10 TO SCRAP PATH
380 1 21 0.5                          /10 TO SCRAP PATH
380 0 05 1 1 0 0 0 10                 /10 PATH INFO
380 1 20 2000                         /10 TO SCRAP PATH
380 1 21 0.5                          /10 TO SCRAP PATH
440 0 03 1 1 1 0 0 10                 /10 PATH INFO
440 1 20 1000                         /10 TO SCRAP
410 0 05 1 1 0 0 0 10                 /10 PATH INFO
410 1 20 0                            /10 TO RENEW
440 0 05 1 1 1 0 0 10                 /10 PATH INFO
440 1 20 1000                         /10 TO SCRAP
181 0 03 1 1 0 0 0 11                 /11 PATH INFO
181 1 20 1000                         /11 MILES RENEWED
211 0 03 1 1 1 0 1 11                 /11 PATH INFO
211 1 20 1000                         /11 NEW TIES @ 1000/MILE
211 1 37 9.09 110.00 1                /11 DELIVERY RF.6
211 1 31 1000 15.00 1                 /11 NEW TIES
281 0 03 1 1 0 0 0 11                 /11 RENEWED MILES
281 1 20 1000                         /11 RENEWED MILES
281 1 37 1.0 45.09 1                  /11 EQUIP. DELIVERY RF.4
281 1 38 4.1 17.50 2.2                /11 TRAVEL TO SITE
281 1 40 11.1 7.50 2                  /11 PULL SPIKES
281 1 32 11.1 1.67 2                  /11 PULL SPIKES
281 1 42 11.1 12.50 1                 /11 CUT UP TIES
281 1 32 11.1 5.01 1                  /11 CUT UP TIES
281 1 43 11.1 15.00 1                 /11 SET NEW TIES
281 1 32 11.1 7.47 1                  /11 SET NEW TIES
281 1 42 11.1 7.47 1                  /11 INJECT
281 1 32 11.1 4.79 1                  /11 INJECT
281 1 42 11.1 12.50 1                 /11 INSERT PLATES
281 1 32 11.1 2.00 1                  /11 INSERT PLATES
281 1 31 2000 4.52 1                  /11 INSERT PLATES
281 1 41 11.1 10.00 2                 /11 SPIKE
281 1 32 11.1 2.78 1                  /11 SPIKE
281 1 31 9000 .25 1                   /11 SPIKE
281 1 43 11.1 15.00 1                 /11 TAMP
281 1 32 11.1 7.26 1                  /11 TAMP
281 1 43 11.1 15.00 1                 /11 REG BALLAST
281 1 32 11.1 9.32 1                  /11 REG BALLAST
281 1 31                              /11 OTM
```

```
281 1 44 11.1 17.50 1          /11 FOREMAN
281 1 44 11.1 17.50 2          /11 ASST. FOREMAN
411 0 03 1 1 0 0 0 11          /11 PATH INFO
411 1 20 1000                  /11 TO RENEW
381 0 03 1 1 0 0 0 11          /11 PATH INFO
381 1 21 2000                  /11 TO SCRAP PATH
381 1 21 0.5                   /11 TO SCRAP PATH
441 0 03 1 1 1 0 0 11          /11 PATH INFO
441 1 20 1000                  /11 TO SCRAP
441 1 21 1.0                   /11 TO SCRAP
182 0 01 1 1 0 0 0 12          /12 PATH INFO
182 1 20 1000.0                /12 JOINT REPAIRS
212 0 01 1 1 1 0 1 12          /12 PATH INFO
212 1 20 0                     /12 MFG SOURCE
282 0 01 1 1 0 0 0 12          /12 PATH INFO
282 1 20 1000                  /12 REPAIRS @ 1/MILE
282 1 36                       /12 REPAIRS
412 0 01 1 1 0 0 0 12          /12 PATH INFO
412 1 20 0                     /12 TO RENEW
382 0 01 1 1 0 1 0 12          /12 PATH INFO
382 1 20 1000.0                /12 TO SCRAP PATH
442 0 01 1 1 1 0 0 12          /12 PATH INFO
442 1 20 0                     /12 TO SCRAP
183 0 01 1 1 0 0 0 13          /13 PATH INFO
183 1 20 1000                  /13 TURN OUT REPAIRS
183 1 21 1.0                   /13 TURN OUT REPAIRS
213 0 01 1 1 1 0 0 13          /13 PATH INFO
213 1 20 0                     /13 MFG SOURCE
283 0 01 1 1 0 0 0 13          /13 PATH INFO
283 1 20 1000                  /13 REPAIRS @ 1/MILE
283 1 32 8.0 3.92 1            /13 REPAIRS
283 1 31 100 1 1               /13 REPAIRS
283 1 42 8.0 12.50 4           /13 REPAIRS
413 0 01 1 1 0 0 0 13          /13 PATH INFO
413 1 20 0                     /13 TO RENEW
383 0 01 1 1 0 1 0 13          /13 PATH INFO
383 1 20 1000                  /13 TO SCRAP PATH
383 1 21 1.0                   /13 TO SCRAP PATH
443 0 01 1 1 1 0 0 13          /13 PATH INFO
443 1 20 0                     /13 TO SCRAP
184 0 01 1 1 0 0 0 14          /14 PATH INFO
184 1 20 1000                  /14 MILES REBUILT
214 0 01 1 1 1 0 0 14          /14 PATH INFO
214 1 20 0                     /14 MFG SOURCE
284 0 01 1 1 0 0 0 14          /14 PATH INFO
284 1 20 1000                  /14 REPAIRS
284 1 36 100 1 1               /14 REBUILT RF.5
414 0 01 1 1 0 0 0 14          /14 PATH INFO
414 1 20 0                     /14 TO RENEW
384 0 01 1 1 0 1 0 14          /14 PATH INFO
384 1 20 1000                  /14 TO SCRAP PATH
384 1 21 1.0                   /14 TO SCRAP PATH
444 0 01 1 1 1 0 0 14          /14 PATH INFO
444 1 20 0                     /14 TO SCRAP
```

```
185 0 02 1 1 0 0 0 15          /15 PATH INFO
185 1 20 3000                  /15 RAIL GRIND
215 0 02 1 1 1 0 0 15          /15 PATH INFO
215 1 20 0.0                   /15 NULL
285 0 02 1 1 0 0 0 15          /15 PATH INFO
285 1 20 3000                  /15 REPAIRS
285 1 21 1.0                   /15 REPAIRS
285 1 36 .5 200 1              /15 REPAIRS RF.6
415 0 02 1 1 0 0 0 15          /15 PATH INFO
415 1 20 0                     /15 TO RENEW
385 0 02 1 1 0 1 0 15          /15 PATH INFO
385 1 20 3000                  /15 TO SCRAP PATH
385 1 21 1.0                   /15 TO SCRAP PATH
445 0 01 1 1 1 0 0 15          /15 PATH INFO
445 1 20 0                     /15 TO SCRAP
485 0 02 1 1 0 0 0 15          /15 RETURN
485 1 20 3000                  /15 RETURN
485 1 21 0.0                   /15 RETURN
186 0 05 1 1 0 0 0 15          /16 PATH INFO
186 1 20 1000                  /16 MILES CLEANED
186 1 21 1.0                   /16 MILES CLEANED
216 0 05 1 1 1 0 1 15          /16 PATH INFO
216 1 20 1000                  /16 MFG SOURCE
286 0 05 1 1 0 0 0 15          /16 PATH INFO
286 1 20 2000                  /16 REPAIRS
286 1 36 500 1 1               /16 REPAIRS
416 0 05 1 1 0 0 0 16          /16 PATH INFO
416 1 20 0                     /16 TO RENEW
386 0 05 1 1 0 0 0 16          /16 PATH INFO
386 1 20 2000                  /16 TO SCRAP PATH
386 1 21 0.5                   /16 TO SCRAP PATH
446 0 05 1 1 1 0 0 16          /16 PATH INFO
446 1 20 1000                  /16 TO SCRAP
187 0 01 1 1 0 0 0 17          /17 PATH INFO
187 1 20 1000                  /17 BRUSH & WEED MILES
217 0 01 1 1 1 0 0 17          /17 PATH INFO
217 1 20 0                     /17 MFG SOURCE
287 0 01 1 1 0 0 0 17          /17 PATH INFO
287 1 20 1000                  /17 REPAIR ITEMS
287 1 36 175 1 1               /17 REPAIR ITEMS RF.5
417 0 01 1 1 0 0 0 17          /17 PATH INFO
417 1 20 0                     /17 TO RENEW
387 0 01 1 1 0 1 0 17          /17 TO SCRAP PATH
387 1 20 1000                  /17 TO SCRAP PATH
387 1 21 1.0                   /17 TO SCRAP PATH
447 0 01 1 1 1 0 0 17          /17 PATH INFO
447 1 20 0                     /17 TO SCRAP
188 0 01 1 1 0 0 0 18          /18 PATH INFO
188 1 20 0                     /18 DITCHING
218 0 01 1 1 1 0 1 18          /18 PATH INFO
218 1 20 0                     /18 MFG SOURCE
288 0 01 1 1 0 0 0 18          /18 PATH INFO
288 1 20 0                     /18 PATH INFO
418 0 01 1 1 0 0 0 18          /18 PATH INFO
```

```
418 1 20 0                          /18 TO RENEW
448 0 01 1 1 1 0 0 18               /18 PATH INFO
448 1 20 0                          /18 TO SCRAP
189 0 01 1 1 0 0 0 19               /19 PATH INFO
189 1 20                            /19 SNOW & ICE REMOVAL
219 0 01 1 1 1 0 1 19               /19 PATH INFO
219 1 20 0                          /19 MFG SOURCE
289 0 01 1 1 0 0 0 19               /19 PATH INFO
289 1 20 0                          /19 REPAIR ITEMS
419 0 01 1 1 0 0 0 19               /19 PATH INFO
419 1 20 0                          /19 TO RENEW
449 0 01 1 1 1 0 0 19               /19 PATH INFO
449 1 20 0                          /19 TO SCRAP
190 0 01 1 1 0 0 0 20               /20 PATH INFO
190 1 20 0                          /20 SLIDES & WASHOUTS
220 0 01 1 1 1 0 1 20               /20 PATH INFO
220 1 20 0                          /20 BALLAST/TIES/RAIL
290 0 01 1 1 0 0 0 20               /20 PATH INFO
290 1 20 0                          /20 REPAIR ITEMS
420 0 01 1 1 0 0 0 20               /20 PATH INFO
420 1 20 0                          /20 TO RENEW
450 0 01 1 1 1 0 0 20               /20 PATH INFO
450 1 20 0                          /20 TO SCRAP
002 0 01 1 1 0 0 0 0 1              /   DUMPING PATH
002 1 21 0.00                       /   DUMPING PATH
001 0 01 1 1 0 0 0 0 0              /   TRACK IN USE
001 1 20 100000                     /   TRACK IN USE RF.2
001 1 21 .15                        /   TRACK IN USE
001 0 02 1 1 0 0 0 0 0              /   TRACK IN USE
001 1 20 100000                     /   TRACK IN USE RF.2
001 1 21 .04                        /   TRACK IN USE
001 0 03 1 1 0 0 0 0 0              /   TRACK IN USE
001 1 20 200000                     /   TRACK IN USE RF.2
001 1 21 .05                        /   TRACK IN USE
001 0 04 1 1 0 0 0 0 0              /   TRACK IN USE
001 1 20 0                          /   TRACK IN USE
001 0 05 1 1 0 0 0 0 0              /   TRACK IN USE
001 1 20 200000                     /   TRACK IN USE RF.2
001 1 21 .05                        /   TRACK IN USE
016 0 01 1 1 0 0 0 0                /   PATH INFO
016 1 20 200000                     /   FED CAR INSPT
016 1 21 0.15                       /   FED CAR INSPT
017 0 01 1 1 0 0 0 0                /   PATH INFO
017 1 20 1000                       /   FED CAR RESULTS
017 1 21 1.0                        /   FED CAR RESULTS
018 0 01 1 1 0 0 0 0                /   PATH INFO
018 1 20 200000                     /   SPERRY RAIL CAR
018 1 21 0.15                       /   SPERRY RAIL CAR
018 1 36 25 1 1                     /   SPERRY RAIL CAR
019 0 01 1 1 0 0 0 0                /   PATH INFO
019 1 20 1000                       /   SPERRY RAIL CAR
019 1 21 1.0                        /   SPERRY RAIL CAR
020 0 01 1 1 0 0 0 0                /   PATH INFO
020 1 20 200000                     /   SOUND RAIL CAR
```

```
020 1 21 0.15                          /    SOUND RAIL CAR
020 1 36 25 1 1                        /    SOUND RAIL CAR
021 0 01 1 1 0 0 0 0                   /    PATH INFO
021 1 20 1000                          /    SOUND RAIL CAR
021 1 21 1.0                           /    SOUND RAIL CAR
022 0 01 1 1 0 0 0 0                   /    PATH INFO
022 1 20 200000                        /    TRACK GEOM CAR
022 1 21 0.15                          /    TRACK GEOM CAR
022 1 36 25 1 1                        /    TRACK GEOM CAR
023 0 01 1 1 0 0 0 0                   /    PATH INFO
023 1 20 1000                          /    TRACK GEOM CAR
023 1 21 1.0                           /    TRACK GEOM CAR
024 0 01 1 1 0 0 0 0                   /    PATH INFO
024 1 20 200000                        /    RAIL WEAR CAR
024 1 21 0.15                          /    RAIL WEAR CAR
024 1 36 25 1 1                        /    RAIL WEAR CAR
025 0 01 1 1 0 0 0 0                   /    PATH INFO
025 1 20 1000                          /    RAIL WEAR CAR
025 1 21 1.0                           /    RAIL WEAR CAR
010 0 01 1 1 0 0 0 0                   /    PATH INFO
010 1 20 10000                         /    FEDERAL VIS INSP
010 1 21 0.15                          /    FEDERAL VIS INSP
010 1 41 .1 10.00 2                    /    FEDERAL VIS INSP
011 0 01 1 1 0 0 0 0                   /    PATH INFO
011 1 20 1000                          /    FEDERAL VIS INSP
011 1 21 1.0                           /    FEDERAL VIS INSP
012 0 01 1 1 0 0 0 0                   /    PATH INFO
012 1 20 10000                         /    STATE VIS INSP
012 1 21 0.15                          /    STATE VIS INSP
012 1 41 .1 10.00 2                    /    STATE VIS INSP
013 0 01 1 1 0 0 0 0                   /    PATH INFO
013 1 20 1000                          /    STATE VIS INSP
013 1 21 1.0                           /    STATE VIS INSP
014 0 01 1 1 0 0 0 0                   /    PATH INFO
014 1 20 26667                         /    SCHED TRACK REVIEW
014 1 21 0.15                          /    SCHED TRACK REVIEW
014 0 02 1 1 0 0 0 0                   /    PATH INFO
014 1 20 100000.00                     /    SCHED TRACK REVIEW
014 1 21 0.04                          /    SCHED TRACK REVIEW
014 0 03 1 1 0 0 0 0                   /    PATH INFO
014 1 20 80000                         /    SCHED TRACK REVIEW
014 1 21 0.05                          /    SCHED TRACK REVIEW
014 0 04 1 1 0 0 0 0                   /    PATH INFO
014 1 20 0                             /    SCHED TRACK REVIEW
014 1 21 0.15                          /    SCHED TRACK REVIEW
014 0 05 1 1 0 0 0 0                   /    PATH INFO
014 1 20 80000                         /    SCHED TRACK REVIEW
014 1 21 0.05                          /    SCHED TRACK REVIEW
026 0 01 1 1 0 0 0 0                   /    PATH INFO
026 1 20 1000                          /    ACCIDENTS
026 1 21 0.15                          /    ACCIDENTS
026 1 36 25000 1 1                     /    ACCIDENTS RF.8
027 0 01 1 1 0 0 0 0                   /    PATH INFO
027 1 20 150                           /    ACC RESULTS
```

```
027 1 21 1.0                      /    ACC RESULTS
030 0 01 1 1 0 0 0 0              /    PATH INFO
030 1 20 200000                   /    JOINT INSP CAR
030 1 21 0.15                     /    JOINT INSP CAR
030 1 36 10 1 1                   /    JOINT INSP CAR
031 0 01 1 1 0 0 0 0              /    PATH INFO
031 1 20 1000                     /    JOINT RESULTS
031 1 21 1.0                      /    JOINT RESULTS
015 0 01 1 1 0 0 0 0              /    PATH INFO
015 1 20 9150.0                   /    INSPECT RESULTS
015 1 21 1.0                      /    INSPECT RESULTS
015 0 02 1 1 0 0 0 0              /    PATH INFO
015 1 20 4.0D+03                  /    INSPECT RESULTS
015 1 21 1.0                      /    INSPECT RESULTS
015 0 03 1 1 0 0 0 0              /    PATH INFO
015 1 20 4000.0                   /    INSPECT RESULTS
015 1 21 1.0                      /    INSPECT RESULTS
015 0 04 1 1 0 0 0 0              /    PATH INFO
015 1 20 0                        /    INSPECT RESULTS
015 0 05 1 1 0 0 0 0              /    PATH INFO
015 1 20 4000.0                   /    INSPECT RESULTS
015 1 21 1.0                      /    INSPECT RESULTS
061 0 01 1 1 0 0 0 0              /    PATH INFO
061 1 20 10000.0                  /    TO REPAIR
061 1 21 1.0                      /    TO REPAIR
061 0 02 1 1 0 0 0 0              /    PATH INFO
061 1 20 3000.0                   /    TO REPAIR
061 1 21 1.0                      /    TO REPAIR
061 0 03 1 1 0 0 0 0              /    PATH INFO
061 1 20 2000.0                   /    TO REPAIR
061 1 21 1.0                      /    TO REPAIR
061 0 04 1 1 0 0 0 0              /    PATH INFO
061 1 20 0                        /    TO REPAIR
061 0 05 1 1 0 0 0 0              /    PATH INFO
061 1 20 4000.0                   /    TO REPAIR
061 1 21 1.0                      /    TO REPAIR
062 0 01 1 1 0 0 0 0              /    PATH INFO
062 1 20 8000.0                   /    AFTER MAINT 1
062 0 02 1 1 0 0 0 0              /    PATH INFO
062 1 20 3000.0                   /    AFTER MAINT 1
062 1 21 1.0                      /    AFTER MAINT 1
062 0 03 1 1 0 0 0 0              /    PATH INFO
062 1 20 2000.0                   /    AFTER MAINT 1
062 1 21 1.0                      /    AFTER MAINT 1
062 0 04 1 1 0 0 0 0              /    PATH INFO
062 1 20 0                        /    AFTER MAINT 1
062 0 05 1 1 0 0 0 0              /    AFTER MAINT 1
062 1 20 4000                     /    AFTER MAINT 1
062 1 21 1.0                      /    AFTER MAINT 1
121 0 01 1 1 0 0 0 0              /    PATH INFO
121 1 20 2000                     /    TO MAINT #1
121 1 21 1.0                      /    TO MAINT #1
063 0 01 1 1 0 0 0 0              /    PATH INFO
063 1 20 7000                     /    AFTER MAINT 2
```

```
063 0 02 1 1    0                    /   PATH INFO
063 1 20 3000                        /   AFTER MAINT 2
063 0 03 1 1    0                    /   PATH INFO
063 1 20 2000                        /   AFTER MAINT 2
063 0 04 1 1    0                    /   PATH INFO
063 1 20 0                           /   AFTER MAINT 2
063 0 05 1 1    0                    /   PATH INFO
063 1 20 4000                        /   AFTER MAINT 2
122 0 01 1 1    0                    /   PATH INFO
122 1 20 1000                        /   TO MAINT #2
064 0 01 1 1    0                    /   PATH INFO
064 1 20 6000                        /   AFTER MAINT 2
064 0 02 1 1    0                    /   PATH INFO
064 1 20 3000                        /   AFTER MAINT 2
064 0 03 1 1    0                    /   PATH INFO
064 1 20 2000                        /   AFTER MAINT 2
064 0 04 1 1    0                    /   PATH INFO
064 1 20 0                           /   AFTER MAINT 2
064 0 05 1 1    0                    /   PATH INFO
064 1 20 4000                        /   AFTER MAINT 3
123 0 01 1 1    0                    /   PATH INFO
123 1 20 1000                        /   TO MAINT #2
066 0 01 1 1    0                    /   PATH INFO
066 1 20 4000                        /   AFTER MAINT 5
066 0 02 1 1    0                    /   PATH INFO
066 1 20 3000                        /   AFTER MAINT 5
066 0 03 1 1    0                    /   PATH INFO
066 1 20 2000                        /   AFTER MAINT 5
066 0 04 1 1    0                    /   PATH INFO
066 1 20 0                           /   AFTER MAINT 5
066 0 05 1 1    0                    /   PATH INFO
066 1 20 4000                        /   AFTER MAINT 5
125 0 01 1 1    0                    /   PATH INFO
125 1 20 1000                        /   TO MAINT #5
065 0 01 1 1    0                    /   PATH INFO
065 1 20 5000                        /   TO MAINT 4
065 0 02 1 1    0                    /   PATH INFO
065 1 20 3000                        /   AFTER MAINT 4
065 0 03 1 1    0                    /   PATH INFO
065 1 20 2000                        /   AFTER MAINT 4
065 0 04 1 1    0                    /   PATH INFO
065 1 20 0                           /   AFTER MAINT 4
065 0 05 1 1    0                    /   PATH INFO
065 1 20 4000                        /   AFTER MAINT 4
124 0 01 1 1    0                    /   PATH INFO
124 1 20 1000                        /   TO MAINT #4
067 0 01 1 1    0                    /   PATH INFO
067 1 20 3000                        /   AFTER MAINT 6
067 0 02 1 1    0                    /   PATH INFO
067 1 20 3000                        /   AFTER MAINT 6
067 0 03 1 1    0                    /   PATH INFO
067 1 20 2000                        /   AFTER MAINT 6
067 0 04 1 1    0                    /   PATH INFO
067 1 20 0                           /   AFTER MAINT 6
```

```
067 0 05 1 1    0              /    PATH INFO
067 1 20 4000                  /    AFTER MAINT 6
126 0 01 1 1    0              /    PATH INFO
126 1 20 1000                  /    TO MAINT #6
068 0 01 1 1    0              /    PATH INFO
068 1 20 3000                  /    AFTER MAINT 7
068 0 02 1 1    0              /    PATH INFO
068 1 20 3000                  /    AFTER MAINT 7
068 0 03 1 1    0              /    PATH INFO
068 1 20 2000                  /    AFTER MAINT 7
068 0 04 1 1    0              /    PATH INFO
068 1 20 0                     /    AFTER MAINT 7
068 0 05 1 1    0              /    PATH INFO
068 1 20 4000                  /    AFTER MAINT 7
127 0 01 1 1    0              /    PATH INFO
127 1 20 0                     /    TO MAINT #7
069 0 01 1 1    0              /    PATH INFO
069 1 20 3000                  /    AFTER MAINT 8
069 0 02 1 1    0              /    PATH INFO
069 1 20 3000                  /    AFTER MAINT 8
069 0 03 1 1    0              /    PATH INFO
069 1 20 2000                  /    AFTER MAINT 8
069 0 04 1 1    0              /    PATH INFO
069 1 20 0                     /    AFTER MAINT 8
069 0 05 1 1    0              /    PATH INFO
069 1 20 3000                  /    AFTER MAINT 8
128 0 05 1 1    0              /    PATH INFO
128 1 20 1000                  /    TO MAINT #8
070 0 01 1 1    0              /    PATH INFO
070 1 20 3000                  /    AFTER MAINT 9
070 0 02 1 1    0              /    PATH INFO
070 1 20 3000                  /    AFTER MAINT 9
070 0 03 1 1    0              /    PATH INFO
070 1 20 2000                  /    AFTER MAINT 9
070 0 04 1 1    0              /    PATH INFO
070 1 20 0                     /    AFTER MAINT 9
070 0 05 1 1    0              /    PATH INFO
070 1 20 2000                  /    AFTER MAINT 9
129 0 05 1 1    0              /    PATH INFO
129 1 20 1000                  /    TO MAINT 9
071 0 01 1 1    0              /    PATH INFO
071 1 20 3000                  /    AFTER MAINT 10
071 0 02 1 1    0              /    PATH INFO
071 1 20 3000                  /    AFTER MAINT 10
071 0 03 1 1    0              /    PATH INFO
071 1 20 1000                  /    AFTER MAINT 10
071 0 04 1 1    0              /    PATH INFO
071 1 20 0                     /    AFTER MAINT 10
071 0 05 1 1    0              /    PATH INFO
071 1 20 1000                  /    AFTER MAINT 10
130 0 03 1 1    0              /    PATH INFO
130 1 20 1000                  /    TO MAINT #10
130 0 05 1 1    0              /    PATH INFO
130 1 20 1000                  /    TO MAINT #10
```

```
072 0 01 1 1    0                         /    PATH INFO
072 1 20 3000                             /    AFTER MAINT 11
072 0 02 1 1    0                         /    PATH INFO
072 1 20 3000                             /    AFTER MAINT 11
072 0 03 1 1    0                         /    PATH INFO
072 1 20 0                                /    AFTER MAINT 11
072 0 04 1 1    0                         /    PATH INFO
072 1 20 0                                /    AFTER MAINT 11
072 0 05 1 1    0                         /    PATH INFO
072 1 20 1000                             /    AFTER MAINT 11
131 0 03 1 1    0                         /    PATH INFO
131 1 20 1000                             /    TO MAINT #11
073 0 01 1 1    0                         /    PATH INFO
073 1 20 3000                             /    AFTER MAINT 12
073 0 02 1 1    0                         /    PATH INFO
073 1 20 3000                             /    AFTER MAINT 12
073 0 03 1 1    0                         /    PATH INFO
073 1 20 0                                /    AFTER MAINT 12
073 0 04 1 1    0                         /    PATH INFO
073 1 20 0                                /    AFTER MAINT 12
073 0 05 1 1    0                         /    PATH INFO
073 1 20 1000                             /    AFTER MAINT 12
074 0 01 1 1    0                         /    PATH INFO
074 1 20 2000                             /    AFTER MAINT 13
074 0 02 1 1    0                         /    PATH INFO
074 1 20 3000                             /    AFTER MAINT 13
074 0 03 1 1    0                         /    PATH INFO
074 1 20 0                                /    AFTER MAINT 13
074 0 04 1 1    0                         /    PATH INFO
074 1 20 0                                /    AFTER MAINT 13
074 0 05 1 1    0                         /    PATH INFO
074 1 20 1000                             /    AFTER MAINT 13
133 0 01 1 1    0                         /    PATH INFO
133 1 20 1000                             /    TO MAINT #13
075 0 01 1 1    0                         /    PATH INFO
075 1 20 1000                             /    AFTER MAINT 14
075 0 02 1 1    0                         /    PATH INFO
075 1 20 3000                             /    AFTER MAINT 14
075 0 03 1 1    0                         /    PATH INFO
075 1 20 0                                /    AFTER MAINT 14
075 0 04 1 1    0                         /    PATH INFO
075 1 20 0                                /    AFTER MAINT 14
075 0 05 1 1    0                         /    PATH INFO
075 1 20 1000                             /    AFTER  MAINT 14
134 0 01 1 1    0                         /    PATH INFO
134 1 20 1000                             /    TO MAINT #14
076 0 01 1 1    0                         /    PATH INFO
076 1 20 1000                             /    AFTER MAINT 15
076 0 02 1 1    0                         /    PATH INFO
076 1 20 0                                /    AFTER MAINT 15
076 0 03 1 1    0                         /    PATH INFO
076 1 20 0                                /    AFTER MAINT 15
076 0 04 1 1    0                         /    PATH INFO
076 1 20 0                                /    AFTER MAINT 15
```

```
076 0 05 1 1    0                         /    PATH INFO
076 1 20 1000                             /    AFTER MAINT 15
135 0 02 1 1    0                         /    PATH INFO
135 1 20 3000                             /    TO MAINT #15
077 0 01 1 1    0                         /    PATH INFO
077 1 20 1000                             /    AFTER MAINT 16
077 0 02 1 1    0                         /    PATH INFO
077 1 20 0                                /    AFTER MAINT 16
077 0 03 1 1    0                         /    PATH INFO
077 1 20 0                                /    AFTER MAINT 16
077 0 04 1 1    0                         /    PATH INFO
077 1 20 0                                /    AFTER MAINT 16
077 0 05 1 1    0                         /    PATH INFO
077 1 20 0                                /    AFTER MAINT 16
136 0 05 1 1    0                         /    PATH INFO
136 1 20 1000                             /    TO MAINT #16
078 0 01 1 1    0                         /    PATH INFO
078 1 20 0                                /    AFTER MAINT 17
078 0 02 1 1    0                         /    PATH INFO
078 1 20 0                                /    AFTER MAINT 17
078 0 03 1 1    0                         /    PATH INFO
078 1 20 0                                /    AFTER MAINT 17
078 0 04 1 1    0                         /    PATH INFO
078 1 20 0                                /    AFTER MAINT 17
078 0 05 1 1    0                         /    PATH INFO
078 1 20 0                                /    AFTER MAINT 17
137 0 01 1 1    0                         /    PATH INFO
137 1 20 1000                             /    TO MAINT #17
300 0 01 1 1    0                         /    NOT REPAIRED
300 1 20 1150.0                           /    NOT REPAIRED
300 1 21 1.0                              /    NOT REPAIRED
300 0 02 1 1    0                         /    NOT REPAIRED
300 1 20 1000.0                           /    NOT REPAIRED
300 1 21 1.0                              /    NOT REPAIRED
300 0 03 1 1    0                         /    NOT REPAIRED
300 1 20 2000.0                           /    NOT REPAIRED
300 1 21 1.0                              /    NOT REPAIRED
302 0 01 1 1    0                         /    PATH INFO
302 1 20 2000                             /    SLOW ORDER
302 0 02 1 1    0                         /    PATH INFO
302 1 20 1000                             /    SLOW ORDER
302 0 03 1 1    0                         /    PATH INFO
302 1 20 2000                             /    SLOW ORDER
302 0 04 1 1    0                         /    PATH INFO
302 1 20 0                                /    SLOW ORDER
302 0 05 1 1    0                         /    PATH INFO
302 1 20 0                                /    SLOW ORDER
306 0 01 1 1    0                         /    PATH INFO
306 1 20 0                                /    IDLE TRACK
308 0 01 1 1    0                         /    PATH INFO
308 1 20 0                                /    CLOSE TRACK
312 0 01 1 1    0                         /    PATH INFO
312 1 20 3750                             /    NOTED BAD RF.4
313 0 01 1 1    0                         /    PATH INFO
```

```
313 1 20 3000                         /   REWELDED RAIL RF.4
313 1 36                              /   REWELDED RAIL
461 0 01 1 1 0 0 0                    / 1 RETURN
461 1 20 2000                         / 1 RETURN
461 1 21 0                            / 1 RETURN
999 0 01 1 1   0                      /   END OF FILE
```

APPENDIX I

Output Costs From Program Run Example #1

PROGRAM REVISION NUMBER : 0006
SCHEMATIC DIAGRAM NUMBER : 0001
DATA FILE NUMBER : 0001
NUMBER OF TRACK CLASSES : 2
NUMBER OF TRACK COMPONENTS : 2
PATHS IN SCHEMATIC DIAGRAM : 18
NODES IN SCHEMATIC DIAGRAM : 9
TIME SIMULATION BEGINS AT YEAR : 0.00
TIME SIMULATION ENDS ON YEAR : 1.90
TIME INCREMENT SIZE IN YEARS : 0.10

WEIBULL DISTRIBUTION PARAMETERS

CLASS	COMPONENT	AVG. AGE	BETA	C-LIFE
1	1	20.00	2.00	32.00
1	2	20.00	2.00	32.00
2	1	20.00	2.00	32.00
2	2	20.00	2.00	32.00

FUNCTIONS AND SHAPE PARAMETERS ACTIVE

PATH	FUNC.	A	B	C	XO	YO
5	7	2.000D 00	3.900D 01	0.000D-01	0.000D-01	0.000D-01
1	5*	2.000D 00	2.000D 00	1.000D 00	6.000D 00	5.000D 00
17	6	2.900D-01	1.000D-01	7.100D-01	0.000D-01	0.000D-01

*NOTE: This function was also deleted and the full simulation rerun. See Section 6 for comparison discussion with and without the inspection modifying function in operation.

TRACK MAINTENANCE COST

RUN NUMBER : BASE MOD 2
DATE : THU, AUG 14 1980
PLACE : SHAKER RESEARCH CORP.

SIMULATION TIME: 0.00

POPULATION IN USE	CLASS	COMPO-NENT	QUALITY
700.00	1	1	0.10000
50.00	1	2	0.10000
100.00	2	1	0.10000
150.00	2	2	0.10000

NODE	PATH	CLASS	COMPO-ONENT	QUALITY	PATH FLOWS MILES/YEAR	MAINTENANCE DOLLARS
1	1	1	1	0.100000	700.00	210000
1	1	1	2	0.100000	50.00	15000
1	1	2	1	0.100000	100.00	30000
1	1	2	2	0.100000	150.00	45000
2	2	1	1	0.000000	630.00	0
2	18	1	1	1.000000	70.00	0
2	2	1	2	0.000000	45.00	0
2	18	1	2	1.000000	5.00	0
2	2	2	1	0.000000	90.00	0
2	18	2	1	1.000000	10.00	0
2	2	2	2	0.000000	135.00	0
2	18	2	2	1.000000	15.00	0
3	17	1	1	1.000000	8.00	24000
3	17	1	2	1.000000	1.00	3000
3	17	2	1	1.000000	1.00	3000
4	3	1	1	1.000000	78.00	0
4	3	1	2	1.000000	6.00	0
4	3	2	1	1.000000	11.00	0
4	3	2	2	1.000000	15.00	0
5	5	1	1	1.000000	70.00	0
5	4	1	1	1.000000	8.00	200000
5	5	1	2	1.000000	5.00	0
5	4	1	2	1.000000	1.00	25000
5	5	2	1	1.000000	10.00	0
5	4	2	1	1.000000	1.00	25000
5	5	2	2	1.000000	15.00	0
6	6	1	1	0.000000	70.00	1260000
6	7	1	1	0.500000	140.00	1099700
6	6	1	2	0.000000	5.00	430000
6	7	1	2	0.500000	10.00	70000
6	6	2	1	0.000000	10.00	450000
6	7	2	1	0.500000	20.00	276000
6	6	2	2	0.000000	15.00	1290000
6	7	2	2	0.500000	30.00	30994
7	9	1	1	0.166667	84.00	0
7	8	1	1	1.000000	56.00	0
7	9	1	2	0.166667	6.00	0
7	8	1	2	1.000000	4.00	0
7	9	2	1	0.166667	12.00	0
7	8	2	1	1.000000	8.00	0
7	9	2	2	0.062500	16.00	0
7	8	2	2	1.000000	14.00	0
8	11	1	1	0.000040	70.00	0
8	10	1	1	1.000000	14.00	-139972
8	11	1	2	0.000040	5.00	0
8	10	1	2	1.000000	1.00	-9999
8	11	2	1	0.000040	10.00	0
8	10	2	1	1.000000	2.00	-19996
8	11	2	2	0.040005	15.63	0
8	10	2	2	1.000000	0.37	-3749
9	12	1	1	0.102596	78.00	0
9	12	1	2	0.166694	6.00	0
9	12	2	1	0.090942	11.00	0
9	12	2	2	0.040005	15.63	0

TOTAL $ 5312979

TRACK MAINTENANCE COST

RUN NUMBER : BASE MOD 2
DATE : THU, AUG 14 1980
PLACE : SHAKER RESEARCH CORP.

SIMULATION TIME: 0.50

POPULATION IN USE	CLASS	COMPONENT	QUALITY
700.00	1	1	0.07587
50.00	1	2	0.07369
100.00	2	1	0.07369
150.28	2	2	0.07506

NODE	PATH	CLASS	COMPONENT	QUALITY	PATH FLOWS MILES/YEAR	MAINTENANCE DOLLARS
1	1	1	1	0.075873	711.66	213496
1	1	1	2	0.073693	50.00	15000
1	1	2	1	0.073693	100.00	30000
1	1	2	2	0.075056	150.28	45083
2	2	1	1	0.000000	657.66	0
2	18	1	1	1.000000	54.00	0
2	2	1	2	0.000000	46.32	0
2	18	1	2	1.000000	3.68	0
2	2	2	1	0.000000	92.63	0
2	18	2	1	1.000000	7.37	0
2	2	2	2	0.000000	139.00	0
2	18	2	2	1.000000	11.28	0
3	17	1	1	1.000000	5.65	16935
3	17	1	2	1.000000	0.74	2210
3	17	2	1	1.000000	0.74	2210
4	3	1	1	1.000000	59.64	0
4	3	1	2	1.000000	4.42	0
4	3	2	1	1.000000	8.11	0
4	3	2	2	1.000000	11.28	0
5	5	1	1	1.000000	46.38	0
5	4	1	1	1.000000	13.26	331408
5	5	1	2	1.000000	3.68	0
5	4	1	2	1.000000	0.74	18423
5	5	2	1	1.000000	7.37	0
5	4	2	1	1.000000	0.74	18423
5	5	2	2	1.000000	11.28	0
6	6	1	1	0.000000	46.38	834913
6	7	1	1	0.500000	92.77	728694
6	6	1	2	0.000000	3.68	316882
6	7	1	2	0.500000	7.37	51585
6	6	2	1	0.000000	7.37	331621
6	7	2	1	0.500000	14.74	203394
6	6	2	2	0.000000	11.28	970016
6	7	2	2	0.500000	22.56	23306
7	9	1	1	0.166667	55.66	0
7	8	1	1	1.000000	37.11	0
7	9	1	2	0.166667	4.42	0
7	8	1	2	1.000000	2.95	0
7	9	2	1	0.166667	8.84	0
7	8	2	1	1.000000	5.90	0
7	9	2	2	0.062500	12.03	0
7	8	2	2	1.000000	10.53	0
8	11	1	1	0.000040	46.39	0
8	10	1	1	1.000000	9.27	-92749
8	11	1	2	0.000040	3.68	0
8	10	1	2	1.000000	0.74	-7367
8	11	2	1	0.000040	7.37	0
8	10	2	1	1.000000	1.47	-14735
8	11	2	2	0.040005	11.75	0
8	10	2	2	1.000000	0.28	-2819
9	12	1	1	0.222295	59.64	0
9	12	1	2	0.166694	4.42	0
9	12	2	1	0.090942	8.11	0
9	12	2	2	0.040005	11.75	0

TOTAL $ 4035933

TRACK MAINTENANCE COST

RUN NUMBER : BASE MOD 2
DATE : THU, AUG 14 1980
PLACE : SHAKER RESEARCH CORP.

SIMULATION TIME: 1.00

POPULATION IN USE	CLASS	COMPO-NENT	QUALITY
700.00	1	1	0.06402
50.00	1	2	0.05848
100.00	2	1	0.05848
150.49	2	2	0.06034

NODE	PATH	CLASS	COMPO-ONENT	QUALITY	PATH FLOWS MILES/YEAR	MAINTENANCE DOLLARS
1	1	1	1	0.064017	740.07	222020
1	1	1	2	0.058476	50.00	15000
1	1	2	1	0.058476	100.00	30000
1	1	2	2	0.060338	150.49	45147
2	2	1	1	0.000000	692.69	0
2	18	1	1	1.000000	47.38	0
2	2	1	2	0.000000	47.08	0
2	18	1	2	1.000000	2.92	0
2	2	2	1	0.000000	94.15	0
2	18	2	1	1.000000	5.85	0
2	2	2	2	0.000000	141.41	0
2	18	2	2	1.000000	9.08	0
3	17	1	1	1.000000	4.59	13760
3	17	1	2	1.000000	0.58	1754
3	17	2	1	1.000000	0.58	1754
4	3	1	1	1.000000	51.96	0
4	3	1	2	1.000000	3.51	0
4	3	2	1	1.000000	6.43	0
4	3	2	2	1.000000	9.08	0
5	5	1	1	1.000000	37.29	0
5	4	1	1	1.000000	14.67	366800
5	5	1	2	1.000000	2.92	0
5	4	1	2	1.000000	0.58	14619
5	5	2	1	1.000000	5.85	0
5	4	2	1	1.000000	0.58	14619
5	5	2	2	1.000000	9.08	0
6	6	1	1	0.000000	37.29	671257
6	7	1	1	0.500000	74.58	585858
6	6	1	2	0.000000	2.92	251449
6	7	1	2	0.500000	5.85	40933
6	6	2	1	0.000000	5.85	263144
6	7	2	1	0.500000	11.70	161395
6	6	2	2	0.000000	9.08	780922
6	7	2	2	0.500000	18.16	18763
7	9	1	1	0.166667	44.75	0
7	8	1	1	1.000000	29.83	0
7	9	1	2	0.166667	3.51	0
7	8	1	2	1.000000	2.34	0
7	9	2	1	0.166667	7.02	0
7	8	2	1	1.000000	4.68	0
7	9	2	2	0.062500	9.69	0
7	8	2	2	1.000000	8.48	0
8	11	1	1	0.000040	37.29	0
8	10	1	1	1.000000	7.46	-74569
8	11	1	2	0.000040	2.92	0
8	10	1	2	1.000000	0.58	-5846
8	11	2	1	0.000040	5.85	0
8	10	2	1	1.000000	1.17	-11693
8	11	2	2	0.040005	9.46	0
8	10	2	2	1.000000	0.23	-2269
9	12	1	1	0.282370	51.97	0
9	12	1	2	0.166694	3.51	0
9	12	2	1	0.090942	6.43	0
9	12	2	2	0.040005	9.46	0
					TOTAL $	3404821

TRACK MAINTENANCE COST

RUN NUMBER : BASE MOD 2
DATE : THU, AUG 14 1980
PLACE : SHAKER RESEARCH CORP.

SIMULATION TIME: 1.50

POPULATION IN USE	CLASS	COMPO- NENT	QUALITY
700.00	1	1	0.05680
50.00	1	2	0.04967
100.00	2	1	0.04967
150.67	2	2	0.05165

NODE	PATH	CLASS	COMPO- ONENT	QUALITY	PATH FLOWS MILES/YEAR	MAINTENANCE DOLLARS
1	1	1	1	0.056805	799.34	239802
1	1	1	2	0.049674	50.00	15000
1	1	2	1	0.049674	100.00	30000
1	1	2	2	0.051649	150.67	45201
2	2	1	1	0.000000	753.93	0
2	18	1	1	1.000000	45.41	0
2	2	1	2	0.000000	47.52	0
2	18	1	2	1.000000	2.48	0
2	2	2	1	0.000000	95.03	0
2	18	2	1	1.000000	4.97	0
2	2	2	2	0.000000	142.89	0
2	18	2	2	1.000000	7.78	0
3	17	1	1	1.000000	3.98	11925
3	17	1	2	1.000000	0.50	1490
3	17	2	1	1.000000	0.50	1490
4	3	1	1	1.000000	49.38	0
4	3	1	2	1.000000	2.98	0
4	3	2	1	1.000000	5.46	0
4	3	2	2	1.000000	7.78	0
5	5	1	1	1.000000	34.36	0
5	4	1	1	1.000000	15.02	375532
5	5	1	2	1.000000	2.48	0
5	4	1	2	1.000000	0.50	12418
5	5	2	1	1.000000	4.97	0
5	4	2	1	1.000000	0.50	12418
5	5	2	2	1.000000	7.78	0
6	6	1	1	0.000000	34.36	618485
6	7	1	1	0.500000	68.72	539800
6	6	1	2	0.000000	2.48	213599
6	7	1	2	0.500000	4.97	34772
6	6	2	1	0.000000	4.97	223534
6	7	2	1	0.500000	9.93	137100
6	6	2	2	0.000000	7.78	669246
6	7	2	2	0.500000	15.56	16079
7	9	1	1	0.166667	41.23	0
7	8	1	1	1.000000	27.49	0
7	9	1	2	0.166667	2.98	0
7	8	1	2	1.000000	1.99	0
7	9	2	1	0.166667	5.96	0
7	8	2	1	1.000000	3.97	0
7	9	2	2	0.062500	8.30	0
7	8	2	2	1.000000	7.26	0
8	11	1	1	0.000040	34.36	0
8	10	1	1	1.000000	6.87	-68706
8	11	1	2	0.000040	2.48	0
8	10	1	2	1.000000	0.50	-4966
8	11	2	1	0.000040	4.97	0
8	10	2	1	1.000000	0.99	-9932
8	11	2	2	0.040005	8.11	0
8	10	2	2	1.000000	0.19	-1945
9	12	1	1	0.304209	49.38	0
9	12	1	2	0.166694	2.98	0
9	12	2	1	0.090942	5.46	0
9	12	2	2	0.040005	8.11	0
					TOTAL $	3112344

TRACK MAINTENANCE COST

RUN NUMBER : BASE MOD 2
DATE : THU, AUG 14 1980
PLACE : SHAKER RESEARCH CORP.

SIMULATION TIME: 2.00

POPULATION IN USE	CLASS	COMPONENT	QUALITY
700.00	1	1	0.05094
50.00	1	2	0.04458
100.00	2	1	0.04458
150.83	2	2	0.04652

NODE	PATH	CLASS	COMPONENT	QUALITY	PATH FLOWS MILES/YEAR	MAINTENANCE DOLLARS
1	1	1	1	0.050944	890.78	267235
1	1	1	2	0.044582	50.00	15000
1	1	2	1	0.044582	100.00	30000
1	1	2	2	0.046516	150.83	45247
2	2	1	1	0.000000	845.40	0
2	18	1	1	1.000000	45.38	0
2	2	1	2	0.000000	47.77	0
2	18	1	2	1.000000	2.23	0
2	2	2	1	0.000000	95.54	0
2	18	2	1	1.000000	4.46	0
2	2	2	2	0.000000	143.81	0
2	18	2	2	1.000000	7.02	0
3	17	1	1	1.000000	3.50	10487
3	17	1	2	1.000000	0.45	1337
3	17	2	1	1.000000	0.45	1337
4	3	1	1	1.000000	48.88	0
4	3	1	2	1.000000	2.67	0
4	3	2	1	1.000000	4.90	0
4	3	2	2	1.000000	7.02	0
5	5	1	1	1.000000	33.79	0
5	4	1	1	1.000000	15.08	377039
5	5	1	2	1.000000	2.23	0
5	4	1	2	1.000000	0.45	11145
5	5	2	1	1.000000	4.46	0
5	4	2	1	1.000000	0.45	11145
5	5	2	2	1.000000	7.02	0
6	6	1	1	0.000000	33.79	608293
6	7	1	1	0.500000	67.59	530905
6	6	1	2	0.000000	2.23	191704
6	7	1	2	0.500000	4.46	31207
6	6	2	1	0.000000	4.46	200621
6	7	2	1	0.500000	8.92	123047
6	6	2	2	0.000000	7.02	603361
6	7	2	2	0.500000	14.03	14496
7	9	1	1	0.166667	40.55	0
7	8	1	1	1.000000	27.04	0
7	9	1	2	0.166667	2.67	0
7	8	1	2	1.000000	1.78	0
7	9	2	1	0.166667	5.35	0
7	8	2	1	1.000000	3.57	0
7	9	2	2	0.062500	7.48	0
7	8	2	2	1.000000	6.55	0
8	11	1	1	0.000040	33.80	0
8	10	1	1	1.000000	6.76	-67574
8	11	1	2	0.000040	2.23	0
8	10	1	2	1.000000	0.45	-4457
8	11	2	1	0.000040	4.46	0
8	10	2	1	1.000000	0.89	-8914
8	11	2	2	0.040005	7.31	0
8	10	2	2	1.000000	0.18	-1753
9	12	1	1	0.308590	48.88	0
9	12	1	2	0.166694	2.68	0
9	12	2	1	0.090942	4.90	0
9	12	2	2	0.040005	7.31	0

TOTAL $ 2990914

TRACK MAINTENANCE COST SUMMARY

RUN NUMBER : BASE MOD 2
DATE : THU, AUG 14 1980
PLACE : SHAKER RESEARCH CORP.

SIMULATION TIME: 2.00

COST CODE	MAINTENANCE COST
31	2128810.42
32	34963.76
33	0.00
34	399331.18
35	-82700.33
36	13162.15
37	6993.17
38	13986.42
39	0.00
40	13986.42
41	13986.42
42	34576.45
43	406825.09
44	6993.17
ANNUAL TOTAL :	2990914.31

CLASS	MAINTENANCE COST
1	1972324.34
2	1018589.97
ANNUAL TOTAL :	2990914.31

COMP-ONENT	MAINTENANCE COST
1	2083623.55
2	907290.76
ANNUAL TOTAL :	2990914.31

REPAIR BLOCK	BLOCK COST
1	2220938.07

APPENDIX J

Output Costs From Program Run Example #2

PROGRAM REVISION NUMBER :	0006
SCHEMATIC DIAGRAM NUMBER :	0002
DATA FILE	002
NUMBER OF TRACK CLASSES :	1
NUMBER OF TRACK COMPONENTS :	5
PATHS IN SCHEMATIC DIAGRAM :	575
NODES IN SCHEMATIC DIAGRAM :	201
TIME SIMULATION BEGINS AT YEAR :	0.00
TIME SIMULATION ENDS ON YEAR :	0.00
TIME INCREMENT SIZE IN YEARS :	0.00

WEIBULL DISTRIBUTION PARAMETERS

CLASS	COMPONENT	AVG. AGE	BETA	C-LIFE
1	1	20.00	2.00	32.00
1	2	20.00	2.00	32.00
1	3	20.00	2.00	32.00
1	4	20.00	2.00	32.00
1	5	0.00	0.00	0.00

FUNCTIONS AND SHAPE PARAMETERS ACTIVE

PATH	FUNC	A	B	C	XO	YO

1 TRACK MAINTENANCE COST

RUN NUMBER : BASE MOD 2
DATE : TUE, AUG 12 1980
PLACE : SHAKER RESEARCH CORP.

SIMULATION TIME: 0.00

POPULATION IN USE	CLASS	COMPO-NENT	QUALITY
100000.00	1	1	0.15000
100000.00	1	2	0.04000
200000.00	1	3	0.05000
0.00	1	4	0.00000
200000.00	1	5	0.05000

NODE	PATH	CLASS	COMPO-ONENT	QUALITY	PATH FLOWS MILES/YEAR	MAINTENANCE DOLLARS
1	10	1	1	0.150000	10000.00	20000
2	12	1	1	0.150000	10000.00	20000

3	14	1	1	0.150000	26667.00	0
3	14	1	2	0.040000	100000.00	0
3	14	1	3	0.050000	80000.00	0
3	14	1	5	0.050000	80000.00	0
4	16	1	1	0.150000	200000.00	0
5	18	1	1	0.150000	200000.00	5000000
6	20	1	1	0.150000	200000.00	5000000
7	22	1	1	0.150000	200000.00	5000000
8	24	1	1	0.150000	200000.00	5000000
9	26	1	1	0.150000	1000.00	25000000
11	30	1	1	0.150000	200000.00	2000000
12	50	1	1	0.055556	9000.00	0
12	11	1	1	1.000000	1000.00	0
13	48	1	1	0.055556	9000.00	0
13	13	1	1	1.000000	1000.00	0
14	32	1	1	0.145729	199000.00	0
14	31	1	1	1.000000	1000.00	0
15	44	1	1	0.145729	199000.00	0
15	17	1	1	1.000000	1000.00	0
16	42	1	1	0.145729	199000.00	0
16	19	1	1	1.000000	1000.00	0
17	40	1	1	0.145729	199000.00	0
17	21	1	1	1.000000	1000.00	0
18	38	1	1	0.145729	199000.00	0
18	23	1	1	1.000000	1000.00	0
19	36	1	1	0.145729	199000.00	0
19	25	1	1	1.000000	1000.00	0
20	34	1	1	0.000000	850.00	0
20	27	1	1	1.000000	150.00	0
22	35	1	1	1.000000	150.00	0
23	37	1	1	1.000000	1150.00	0
24	39	1	1	1.000000	2150.00	0
25	41	1	1	1.000000	3150.00	0
26	43	1	1	1.000000	4150.00	0
27	45	1	1	1.000000	5150.00	0
28	46	1	1	0.287594	31817.00	0
28	46	1	2	0.040000	100000.00	0
28	46	1	3	0.050000	80000.00	0
28	46	1	5	0.050000	80000.00	0
29	15	1	1	1.000000	9150.00	0
29	52	1	1	0.000002	22667.00	0
29	15	1	2	1.000000	4000.00	0
29	52	1	2	0.000000	96000.00	0
29	15	1	3	1.000000	4000.00	0
29	52	1	3	0.000000	76000.00	0
29	15	1	5	1.000000	4000.00	0
29	52	1	5	0.000000	76000.00	0
30	47	1	1	1.000000	2000.00	0
31	49	1	1	1.000000	11150.00	0
31	49	1	2	1.000000	4000.00	0
31	49	1	3	1.000000	4000.00	0
31	49	1	5	1.000000	4000.00	0
32	61	1	1	1.000000	10000.00	0
32	300	1	1	1.000000	1150.00	0

32	61	1	2	1.000000	3000.00	0
32	300	1	2	1.000000	1000.00	0
32	61	1	3	1.000000	2000.00	0
32	300	1	3	1.000000	2000.00	0
32	61	1	5	1.000000	4000.00	0
33	62	1	1	1.000000	8000.00	0
33	121	1	1	1.000000	2000.00	0
33	62	1	2	1.000000	3000.00	0
33	121	1	2	1.000000	0.00	0
33	62	1	3	1.000000	2000.00	0
33	62	1	5	1.000000	4000.00	0
34	171	1	1	1.000000	2000.00	0
34	151	1	2	1.000000	0.00	0
35	201	1	1	0.000000	2000.00	122812500
35	261	1	1	0.500000	4000.00	210818822
36	371	1	1	0.090909	2200.00	0
36	401	1	1	1.000000	1800.00	2819250
37	461	1	1	0.000001	2000.00	0
37	431	1	1	1.000000	200.00	-15662
38	63	1	1	1.000000	7000.00	0
38	122	1	1	1.000000	1000.00	0
38	63	1	2	1.000000	3000.00	0
38	63	1	3	1.000000	2000.00	0
38	63	1	5	1.000000	4000.00	0
39	172	1	1	1.000000	1000.00	0
40	202	1	1	0.000000	1000.00	61406250
40	262	1	1	0.500000	2000.00	77870973
41	372	1	1	0.090909	1100.00	0
41	402	1	1	1.000000	900.00	1409625
42	462	1	1	0.000000	1000.00	0
42	432	1	1	1.000000	100.00	-7831
43	512	1	1	0.000001	3000.00	0
44	64	1	1	1.000000	6000.00	0
44	123	1	1	1.000000	1000.00	0
44	64	1	2	1.000000	3000.00	0
44	64	1	3	1.000000	2000.00	0
44	64	1	5	1.000000	4000.00	0
45	173	1	1	1.000000	1000.00	0
46	203	1	1	0.000000	1000.00	61406250
46	263	1	1	0.500000	2000.00	47272740
47	373	1	1	0.090909	1100.00	0
47	403	1	1	1.000000	900.00	1409625
48	463	1	1	0.000000	1000.00	0
48	433	1	1	1.000000	100.00	-7831
49	513	1	1	0.000000	4000.00	0
50	65	1	1	1.000000	5000.00	0
50	124	1	1	1.000000	1000.00	0
50	65	1	2	1.000000	3000.00	0
50	65	1	3	1.000000	2000.00	0
50	65	1	5	1.000000	4000.00	0
51	174	1	1	1.000000	1000.00	0
52	204	1	1	0.000000	1000.00	61406250
52	264	1	1	0.500000	2000.00	128453120
53	374	1	1	0.090909	1100.00	0

53	404	1	1	1.000000	900.00	1409625
54	464	1	1	0.000001	1000.00	0
54	434	1	1	1.000000	100.00	-39199
55	514	1	1	0.000001	5000.00	0
56	66	1	1	1.000000	4000.00	0
56	125	1	1	1.000000	1000.00	0
56	66	1	2	1.000000	3000.00	0
56	66	1	3	1.000000	2000.00	0
56	66	1	5	1.000000	4000.00	0
57	175	1	1	1.000000	1000.00	0
58	265	1	1	1.000000	1000.00	18011450
59	375	1	1	0.000000	1000.00	0
60	465	1	1	0.000000	1000.00	0
61	515	1	1	0.000000	6000.00	0
62	67	1	1	1.000000	3000.00	0
62	126	1	1	1.000000	1000.00	0
62	67	1	2	1.000000	3000.00	0
62	67	1	3	1.000000	2000.00	0
62	67	1	5	1.000000	4000.00	0
63	176	1	1	1.000000	1000.00	0
64	266	1	1	1.000000	1000.00	49404350
65	376	1	1	0.000000	1000.00	0
66	466	1	1	0.000000	1000.00	0
67	516	1	1	0.000000	7000.00	0
68	68	1	1	1.000000	3000.00	0
68	68	1	2	1.000000	3000.00	0
68	68	1	3	1.000000	2000.00	0
68	68	1	5	1.000000	4000.00	0
73	517	1	1	0.000000	7000.00	0
74	69	1	1	1.000000	3000.00	0
74	69	1	2	1.000000	3000.00	0
74	69	1	3	1.000000	2000.00	0
74	69	1	5	1.000000	3000.00	0
74	128	1	5	1.000000	1000.00	0
75	178	1	5	1.000000	1000.00	0
76	208	1	5	0.000000	1000.00	4995000
76	278	1	5	0.500000	2000.00	2528856
77	378	1	5	0.500000	2000.00	0
78	478	1	5	0.000000	1000.00	0
78	438	1	5	1.000000	1000.00	0
79	518	1	1	0.000000	7000.00	0
79	518	1	5	0.000000	1000.00	0
80	70	1	1	1.000000	3000.00	0
80	70	1	2	1.000000	3000.00	0
80	70	1	3	1.000000	2000.00	0
80	70	1	5	1.000000	2000.00	0
80	129	1	5	1.000000	1000.00	0
81	179	1	5	1.000000	1000.00	0
82	209	1	5	0.000000	1000.00	0
82	279	1	5	0.500000	2000.00	948024
83	379	1	5	0.500000	2000.00	0
84	479	1	5	0.000000	1000.00	0
84	439	1	5	1.000000	1000.00	0
85	519	1	1	0.000000	7000.00	0

85	519	1	5	0.000000	2000.00	0
86	71	1	1	1.000000	3000.00	0
86	71	1	2	1.000000	3000.00	0
86	71	1	3	1.000000	1000.00	0
86	130	1	3	1.000000	1000.00	0
86	71	1	5	1.000000	1000.00	0
86	130	1	5	1.000000	1000.00	0
87	180	1	3	1.000000	1000.00	0
87	180	1	5	1.000000	1000.00	0
88	210	1	3	0.000000	1000.00	20994900
88	280	1	3	0.500000	2000.00	37034660
88	210	1	5	0.000000	1000.00	0
88	280	1	5	0.500000	2000.00	0
89	380	1	3	0.500000	2000.00	0
89	380	1	5	0.500000	2000.00	0
90	480	1	3	0.000000	1000.00	0
90	440	1	3	1.000000	1000.00	0
90	480	1	5	0.000000	1000.00	0
90	440	1	5	1.000000	1000.00	0
91	520	1	1	0.000000	7000.00	0
91	520	1	3	0.000000	1000.00	0
91	520	1	5	0.000000	3000.00	0
92	72	1	1	1.000000	3000.00	0
92	72	1	2	1.000000	3000.00	0
92	131	1	3	1.000000	1000.00	0
92	72	1	5	1.000000	1000.00	0
93	181	1	3	1.000000	1000.00	0
94	211	1	3	0.000000	1000.00	15999900
94	281	1	3	0.500000	2000.00	27579948
95	411	1	3	0.500000	2000.00	0
97	521	1	1	0.000000	7000.00	0
97	521	1	3	0.000000	1000.00	0
97	521	1	5	0.000000	3000.00	0
98	73	1	1	1.000000	3000.00	0
98	73	1	2	1.000000	3000.00	0
98	73	1	5	1.000000	1000.00	0
99	132	1	1	1.000000	1000.00	0
100	182	1	1	1.000000	1000.00	0
101	212	1	1	0.000000	1000.00	0
101	282	1	1	0.500000	2000.00	0
102	382	1	1	0.000000	2000.00	0
103	482	1	1	0.000000	2000.00	0
104	522	1	1	0.000000	9000.00	0
104	522	1	3	0.000000	1000.00	0
104	522	1	5	0.000000	3000.00	0
105	74	1	1	1.000000	2000.00	0
105	133	1	1	1.000000	1000.00	0
105	74	1	2	1.000000	3000.00	0
105	74	1	5	1.000000	1000.00	0
106	183	1	1	1.000000	1000.00	0
107	283	1	1	1.000000	1000.00	531360
108	383	1	1	0.000000	1000.00	0
109	483	1	1	0.000000	1000.00	0
110	523	1	1	0.000000	10000.00	0

110	523	1	3	0.000000	1000.00	0
110	523	1	5	0.000000	3000.00	0
111	75	1	1	1.000000	1000.00	0
111	134	1	1	1.000000	1000.00	0
111	75	1	2	1.000000	3000.00	0
111	75	1	5	1.000000	1000.00	0
112	184	1	1	1.000000	1000.00	0
113	284	1	1	1.000000	1000.00	100000
114	384	1	1	0.000000	1000.00	0
115	484	1	1	0.000000	1000.00	0
116	524	1	1	0.000000	11000.00	0
116	524	1	3	0.000000	1000.00	0
116	524	1	5	0.000000	3000.00	0
117	76	1	1	1.000000	1000.00	0
117	135	1	2	1.000000	3000.00	0
117	76	1	5	1.000000	1000.00	0
118	185	1	2	1.000000	3000.00	0
119	285	1	2	1.000000	3000.00	300000
120	385	1	2	0.000000	3000.00	0
121	485	1	2	0.000000	3000.00	0
122	525	1	1	0.000000	11000.00	0
122	525	1	2	0.000000	3000.00	0
122	525	1	3	0.000000	1000.00	0
122	525	1	5	0.000000	3000.00	0
123	77	1	1	1.000000	1000.00	0
123	136	1	5	1.000000	1000.00	0
124	186	1	5	1.000000	1000.00	0
125	216	1	5	0.000000	1000.00	0
125	286	1	5	0.500000	2000.00	1000000
126	386	1	5	0.500000	2000.00	0
127	486	1	5	0.000000	1000.00	0
127	446	1	5	1.000000	1000.00	0
128	526	1	1	0.000000	11000.00	0
128	526	1	2	0.000000	3000.00	0
128	526	1	3	0.000000	1000.00	0
128	526	1	5	0.000000	4000.00	0
129	137	1	1	1.000000	1000.00	0
130	187	1	1	1.000000	1000.00	0
131	287	1	1	1.000000	1000.00	175000
132	387	1	1	0.000000	1000.00	0
133	487	1	1	0.000000	1000.00	0
134	527	1	1	0.000000	12000.00	0
134	527	1	2	0.000000	3000.00	0
134	527	1	3	0.000000	1000.00	0
134	527	1	5	0.000000	4000.00	0
140	528	1	1	0.000000	12000.00	0
140	528	1	2	0.000000	3000.00	0
140	528	1	3	0.000000	1000.00	0
140	528	1	5	0.000000	4000.00	0
146	529	1	1	0.000000	12000.00	0
146	529	1	2	0.000000	3000.00	0
146	529	1	3	0.000000	1000.00	0
146	529	1	5	0.000000	4000.00	0
152	530	1	1	0.000000	12000.00	0

152	530	1	2	0.000000	3000.00	0
152	530	1	3	0.000000	1000.00	0
152	530	1	5	0.000000	4000.00	0
171	240	1	2	1.000000	0.00	0
180	325	1	3	0.500000	2000.00	0
181	324	1	3	0.500000	2000.00	0
182	323	1	3	0.500000	2000.00	0
183	322	1	3	0.500000	2000.00	0
184	321	1	3	0.500000	2000.00	0
185	320	1	3	0.500000	2000.00	0
186	319	1	3	0.500000	2000.00	0
187	318	1	1	1.000000	900.00	0
187	318	1	3	0.500000	2000.00	0
188	317	1	1	1.000000	1800.00	0
188	317	1	3	0.500000	2000.00	0
189	316	1	1	1.000000	2700.00	0
189	316	1	3	0.500000	2000.00	0
190	315	1	1	1.000000	4500.00	0
190	315	1	3	0.500000	2000.00	0
191	301	1	1	1.000000	1150.00	0
191	301	1	2	1.000000	1000.00	0
191	301	1	3	1.000000	2000.00	0
192	302	1	1	1.000000	1150.00	0
192	302	1	2	1.000000	1000.00	0
192	302	1	3	1.000000	2000.00	0
196	310	1	1	1.000000	4500.00	0
196	310	1	3	0.500000	2000.00	0
197	311	1	1	0.285714	0.00	0
197	312	1	1	1.000000	4500.00	0
197	311	1	3	0.500000	2000.00	0
198	313	1	1	1.000000	4500.00	0
199	315	1	1	1.000000	4500.00	0
199	315	1	3	0.500000	2000.00	0
200	333	1	1	1.000000	1150.00	0
200	333	1	2	1.000000	1000.00	0
200	333	1	3	1.000000	2000.00	0
201	340	1	1	0.087453	13150.00	0
201	340	1	2	0.250000	4000.00	0
201	340	1	3	0.666667	3000.00	0
201	340	1	5	0.000000	4000.00	0

TOTAL $ 1005067954

TRACK MAINTENANCE COST SUMMARY

RUN NUMBER : Example Run #2
DATE : TUE, AUG 12 1980
PLACE : SHAKER RESEARCH CORP.

SIMULATION TIME: 0.00

COST CODE	MAINTENANCE COST
31	707912137.60
32	39579228.00
33	0.00
34	0.00
35	0.00
36	48575000.00
37	17631170.55
38	3903200.00
39	0.00
40	44442000.00
41	4008000.00
42	99030834.00
43	24285884.00
44	10468500.00
45	5232000.00
46	0.00
47	0.00
48	0.00
49	0.00
ANNUAL TOTAL :	$1005067954.

CLASS	MAINTENANCE COST
1	$1005067954.
ANNUAL TOTAL :	$1005067954.

COMP-ONENT	MAINTENANCE COST
1	893686666.15
2	300000.00
3	101609408.00
4	0.00
5	9471880.00
ANNUAL TOTAL :	$1005067954.

REPAIR BLOCK	BLOCK COST
1	$414305883
2	$62808043.
3	$110080783
4	$191229795
5	$18011450.
6	$49404350.
7	0.00
8	7523356.00
9	948024.00
10	$58029560.
11	$43579848.
12	0.00
13	531360.00
14	100000.00
15	1300000.00
16	0.00
17	175000.00
18	0.00
19	0.00
20	0.00

APPENDIX K

TRACK COST MODEL MAINTENANCE ACTION DIAGRAMS

This appendix contains drawings of all the maintenance action diagrams used for showing examples of the simulation cost methodology. Figure K-1 is a simplified maintenance action diagram for example run number 1. This diagram is to be used along with the input and output data of Appendices G and I, respectively. The costs in Appendices G and I are keyed to the path numbers shown in this Figure K-1.

Figure K-2 is a full maintenance action diagram for example run number 2. This diagram has twenty (20) separate maintenance repair blocks which are individually drawn out, numbered, and shown in Figures K-4 through K-24 of this appendix. Figure K-3 is a blank repair block numbering sheet which can be used to renumber or add paths to the existing diagram as shown in Figure K-2 if needed in the future.

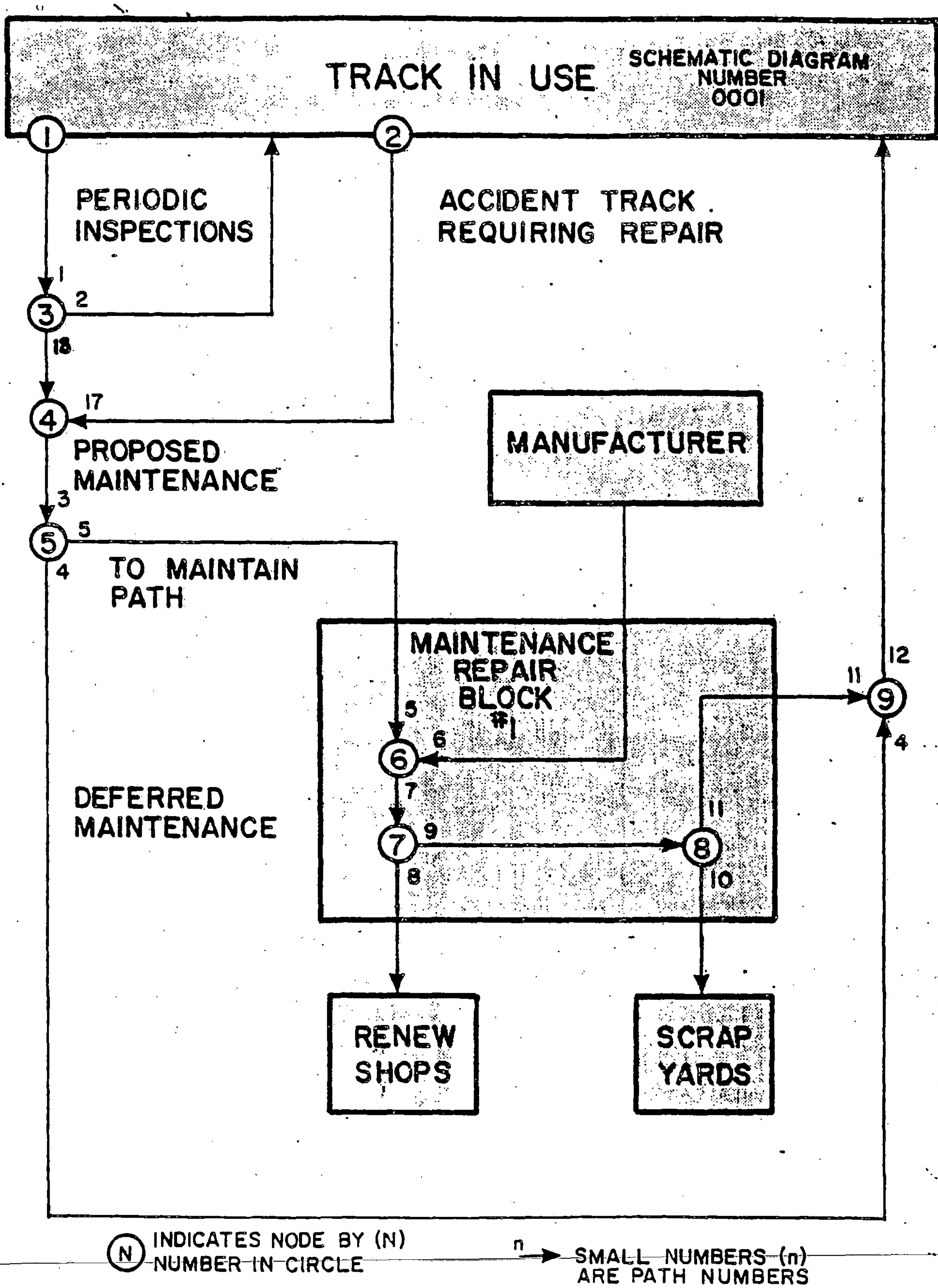


FIGURE K-1. TRACK IN USE, SCHEMATIC DIAGRAM NUMBER 0001

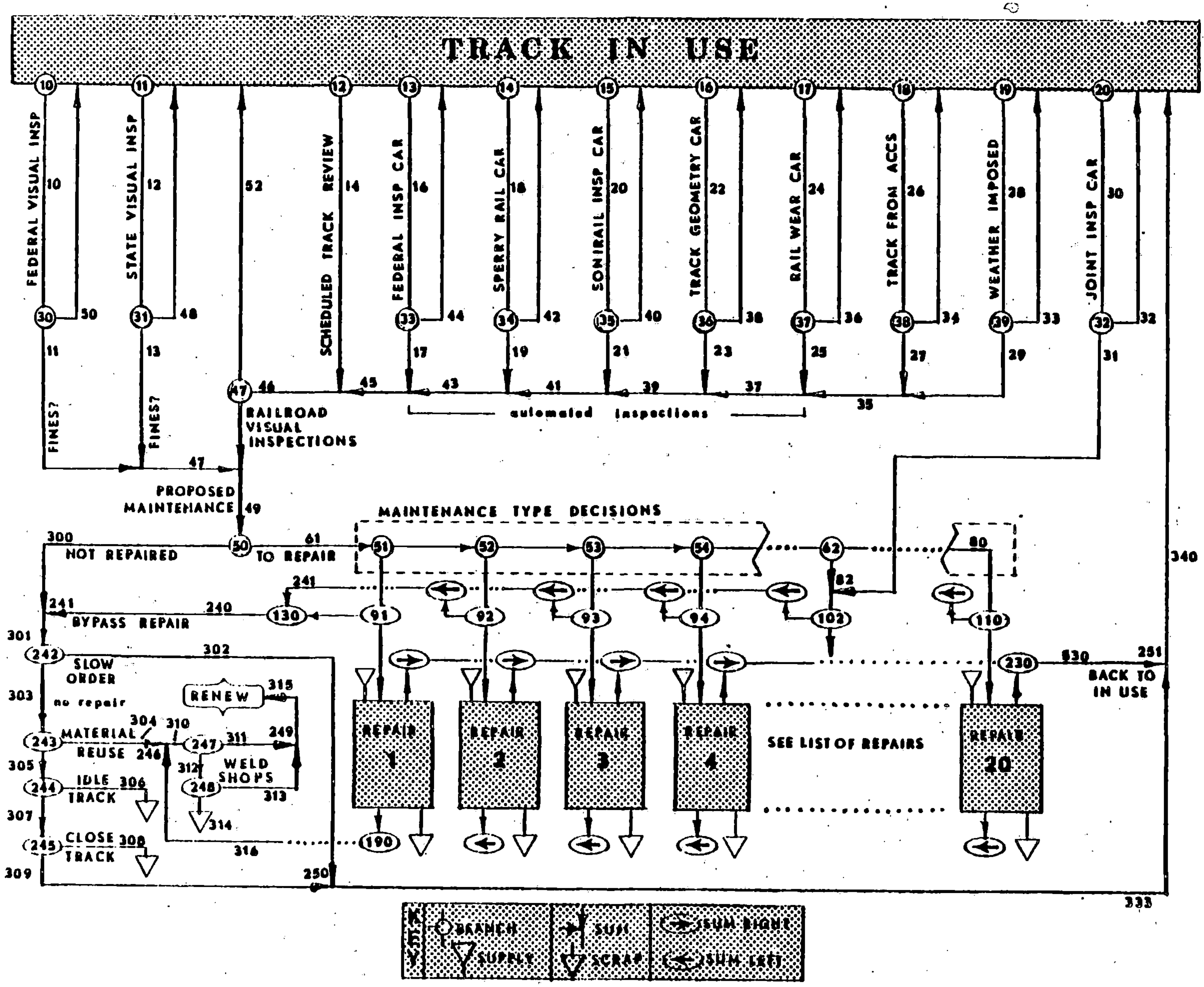


FIGURE K-2. TRACK IN USE

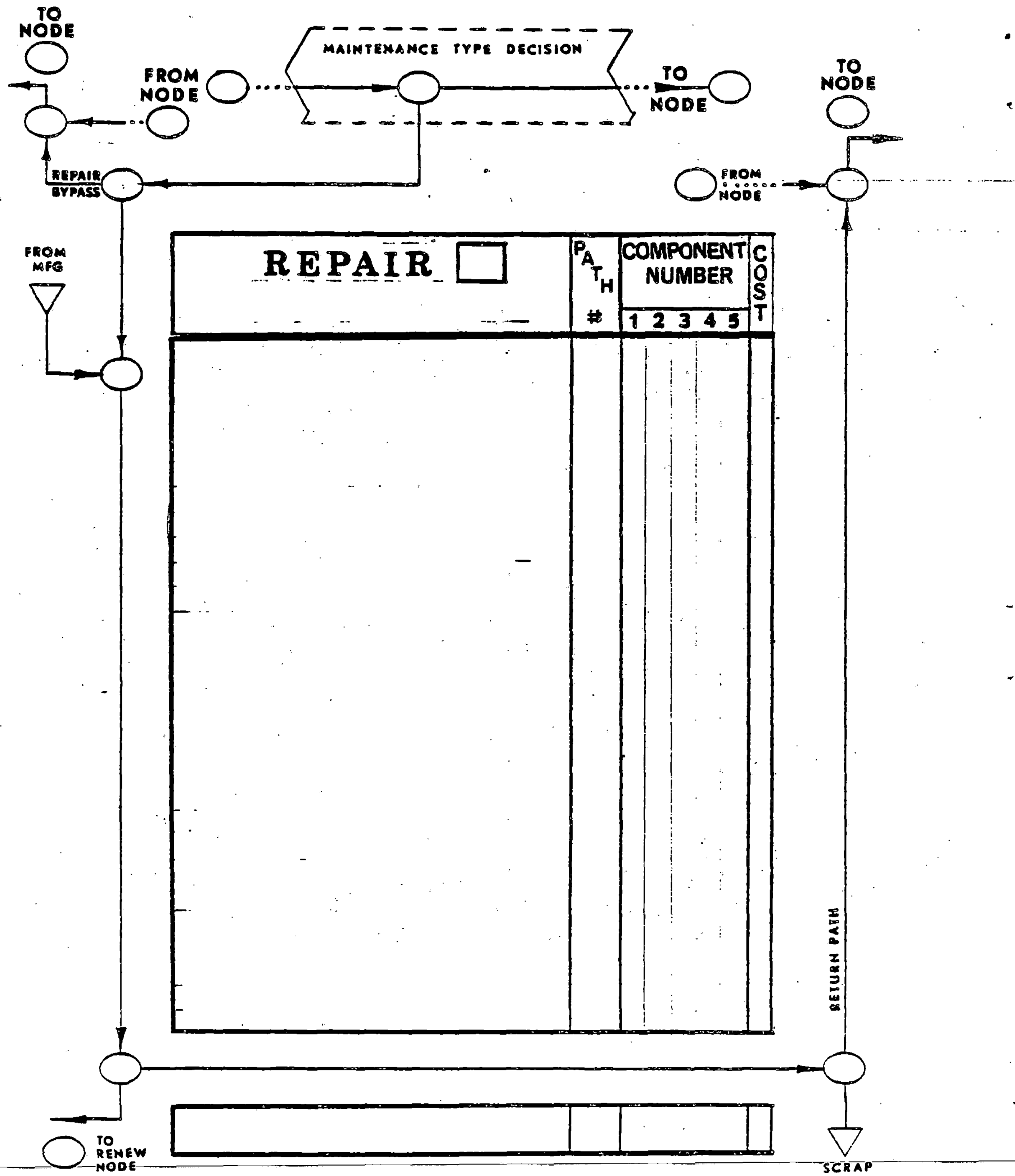
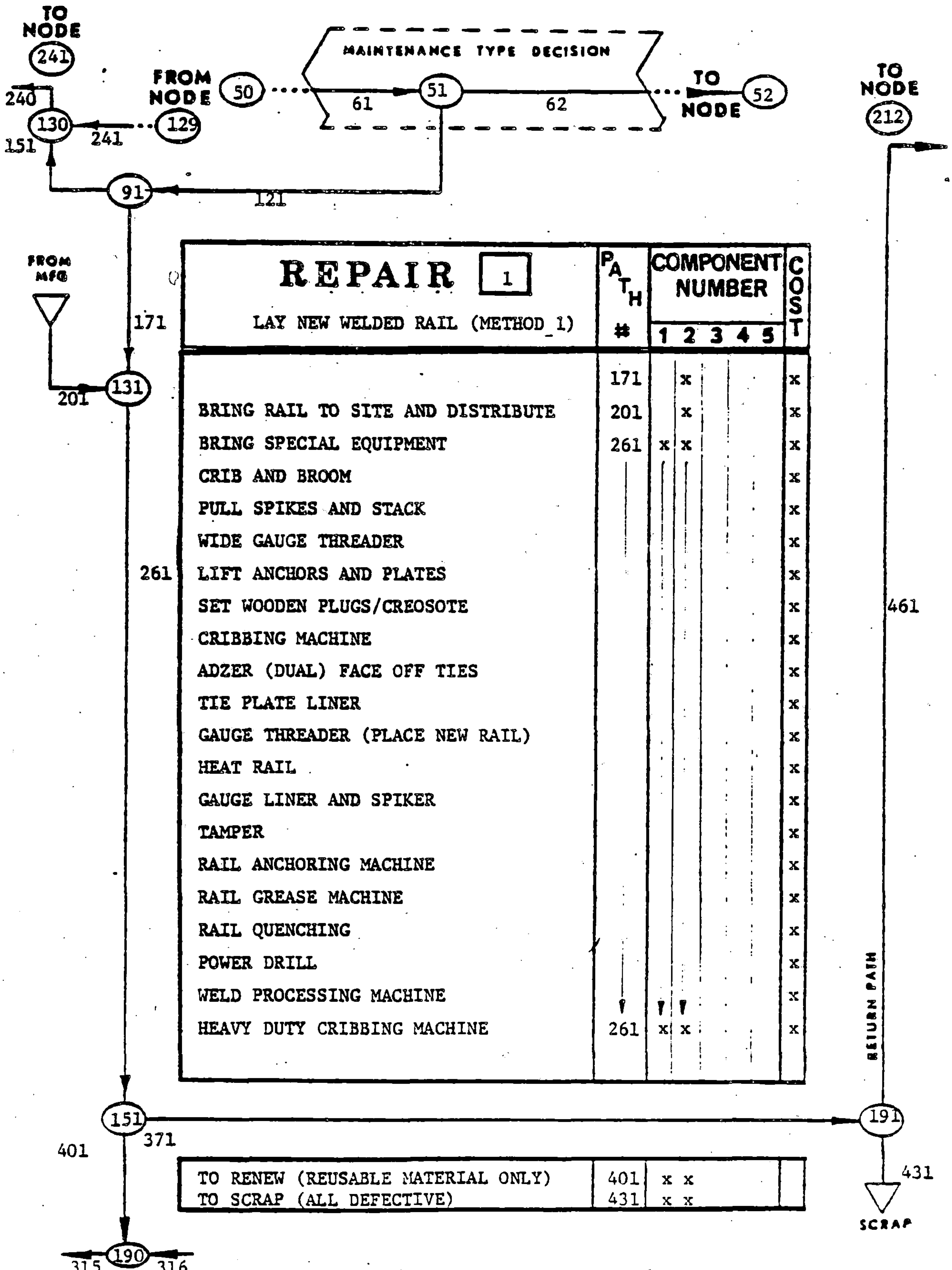


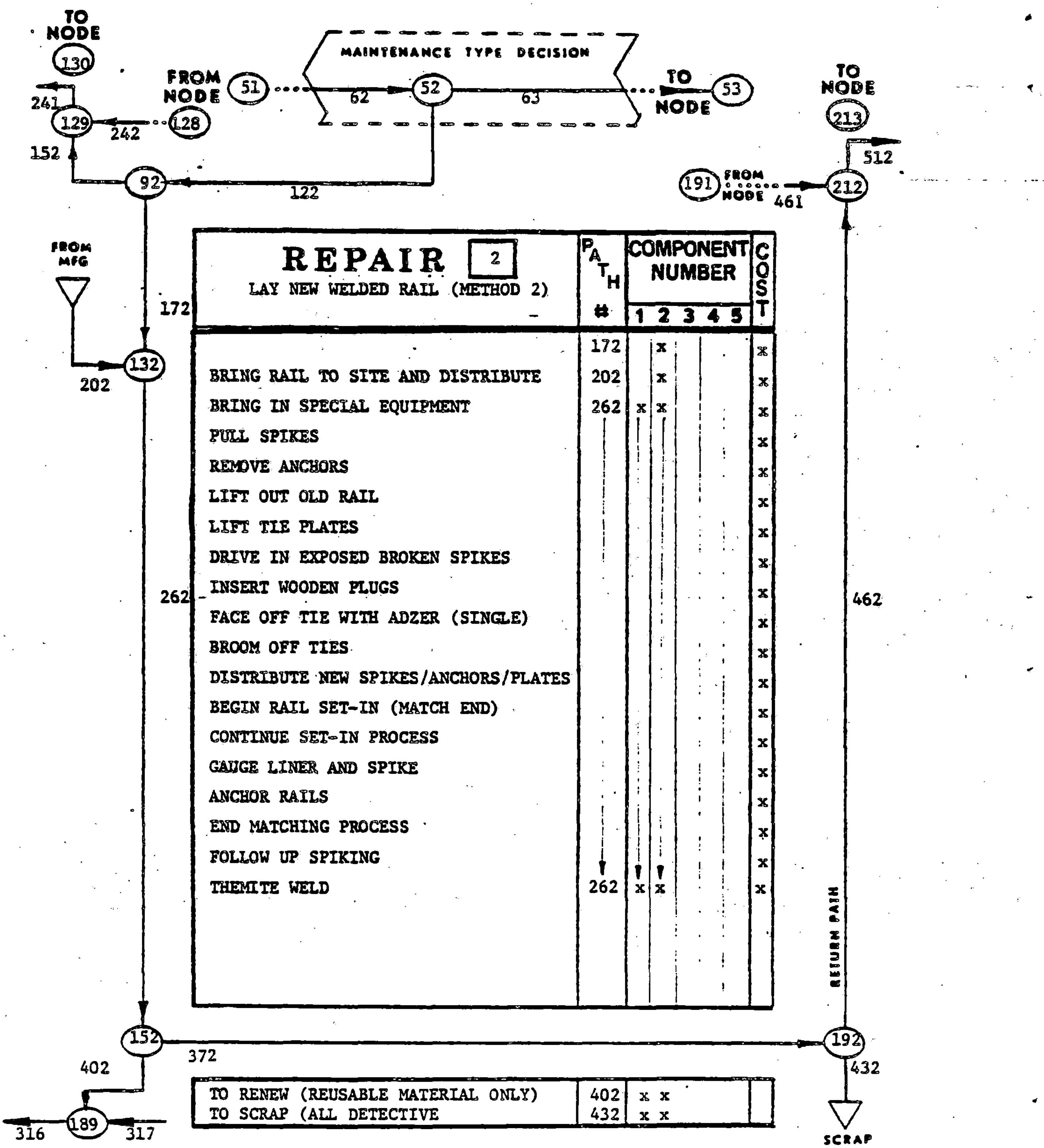
FIGURE K-3. REPAIR OPERATION



REPAIR 1 LAY NEW WELDED RAIL (METHOD 1)	PATH #	COMPONENT NUMBER 1	2	3	4	5	COST
	171		x				x
BRING RAIL TO SITE AND DISTRIBUTE	201		x				x
BRING SPECIAL EQUIPMENT	261	x	x				x
CRIB AND BROOM							x
PULL SPIKES AND STACK							x
WIDE GAUGE THREADER							x
LIFT ANCHORS AND PLATES							x
SET WOODEN PLUGS/CREOSOTE							x
CRIBBING MACHINE							x
ADZER (DUAL) FACE OFF TIES							x
TIE PLATE LINER							x
GAUGE THREADER (PLACE NEW RAIL)							x
HEAT RAIL							x
GAUGE LINER AND SPIKER							x
TAMPER							x
RAIL ANCHORING MACHINE							x
RAIL GREASE MACHINE							x
RAIL QUENCHING							x
POWER DRILL							x
WELD PROCESSING MACHINE							x
HEAVY DUTY CRIBBING MACHINE	261	x	x				x

TO RENEW (REUSABLE MATERIAL ONLY)	401	x	x	
TO SCRAP (ALL DEFECTIVE)	431	x	x	

FIGURE K-4. REPAIR OPERATION 1

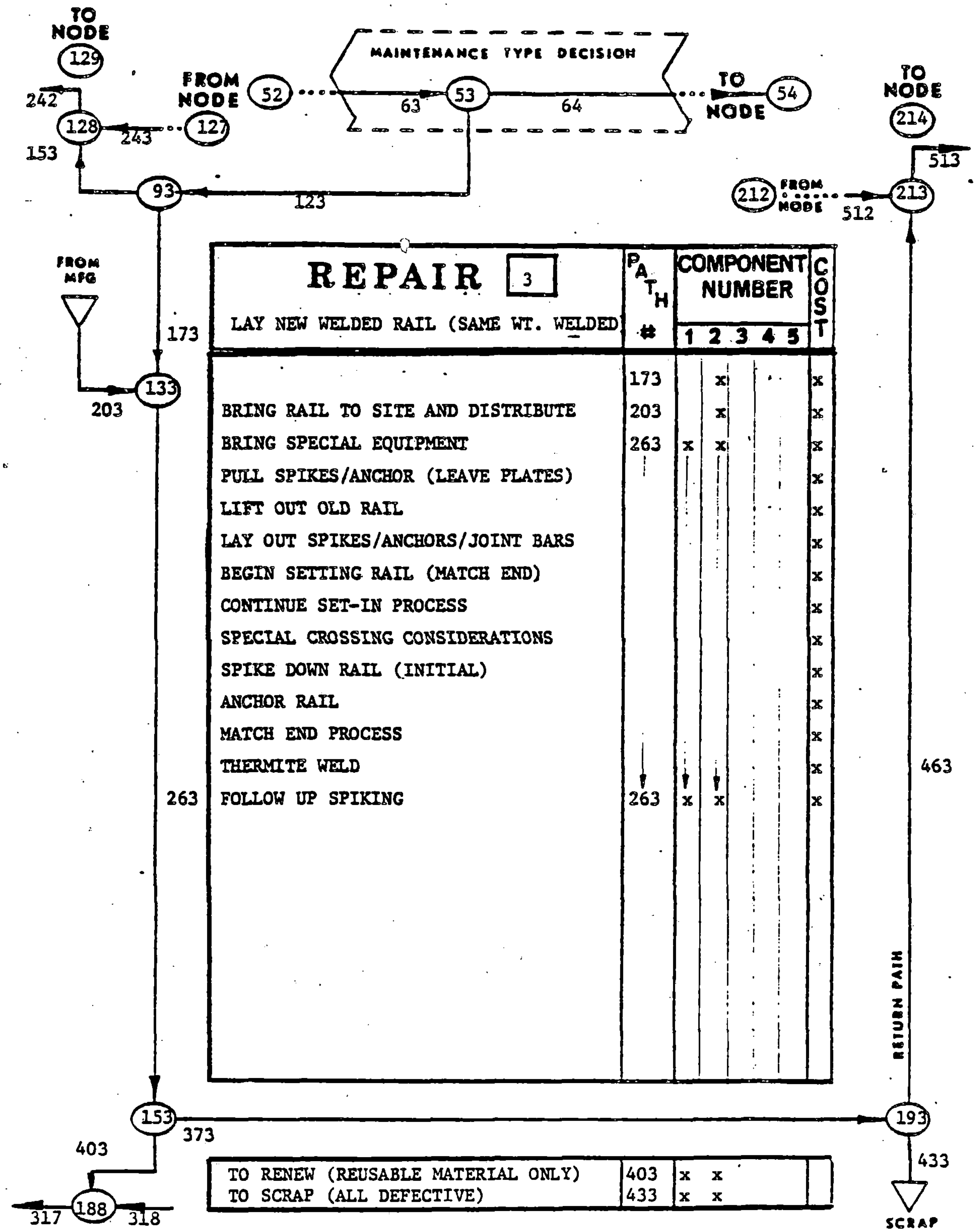


REPAIR 2
LAY NEW WELDED RAIL (METHOD 2)

	PATH #	COMPONENT NUMBER 1	2	3	4	5	COST
	172		x				x
BRING RAIL TO SITE AND DISTRIBUTE	202		x				x
BRING IN SPECIAL EQUIPMENT	262	x	x				x
PULL SPIKES							x
REMOVE ANCHORS							x
LIFT OUT OLD RAIL							x
LIFT TIE PLATES							x
DRIVE IN EXPOSED BROKEN SPIKES							x
INSERT WOODEN PLUGS							x
FACE OFF TIE WITH ADZER (SINGLE)							x
BROOM OFF TIES							x
DISTRIBUTE NEW SPIKES/ANCHORS/PLATES							x
BEGIN RAIL SET-IN (MATCH END)							x
CONTINUE SET-IN PROCESS							x
GAUGE LINER AND SPIKE							x
ANCHOR RAILS							x
END MATCHING PROCESS							x
FOLLOW UP SPIKING							x
THEMITE WELD	262	x	x				x

TO RENEW (REUSABLE MATERIAL ONLY)	402	x	x	
TO SCRAP (ALL DETECTIVE	432	x	x	

FIGURE K-5. REPAIR OPERATION 2



REPAIR 3 LAY NEW WELDED RAIL (SAME WT. WELDED)	PATH #	1	2	3	4	5	COST
	173		x				x
BRING RAIL TO SITE AND DISTRIBUTE	203		x				x
BRING SPECIAL EQUIPMENT	263	x	x				x
PULL SPIKES/ANCHOR (LEAVE PLATES)							x
LIFT OUT OLD RAIL							x
LAY OUT SPIKES/ANCHORS/JOINT BARS							x
BEGIN SETTING RAIL (MATCH END)							x
CONTINUE SET-IN PROCESS							x
SPECIAL CROSSING CONSIDERATIONS							x
SPIKE DOWN RAIL (INITIAL)							x
ANCHOR RAIL							x
MATCH END PROCESS							x
THERMITE WELD							x
FOLLOW UP SPIKING	263	x	x				x

TO RENEW (REUSABLE MATERIAL ONLY)	403	x	x				
TO SCRAP (ALL DEFECTIVE)	433	x	x				

FIGURE K-6. REPAIR OPERATION 3

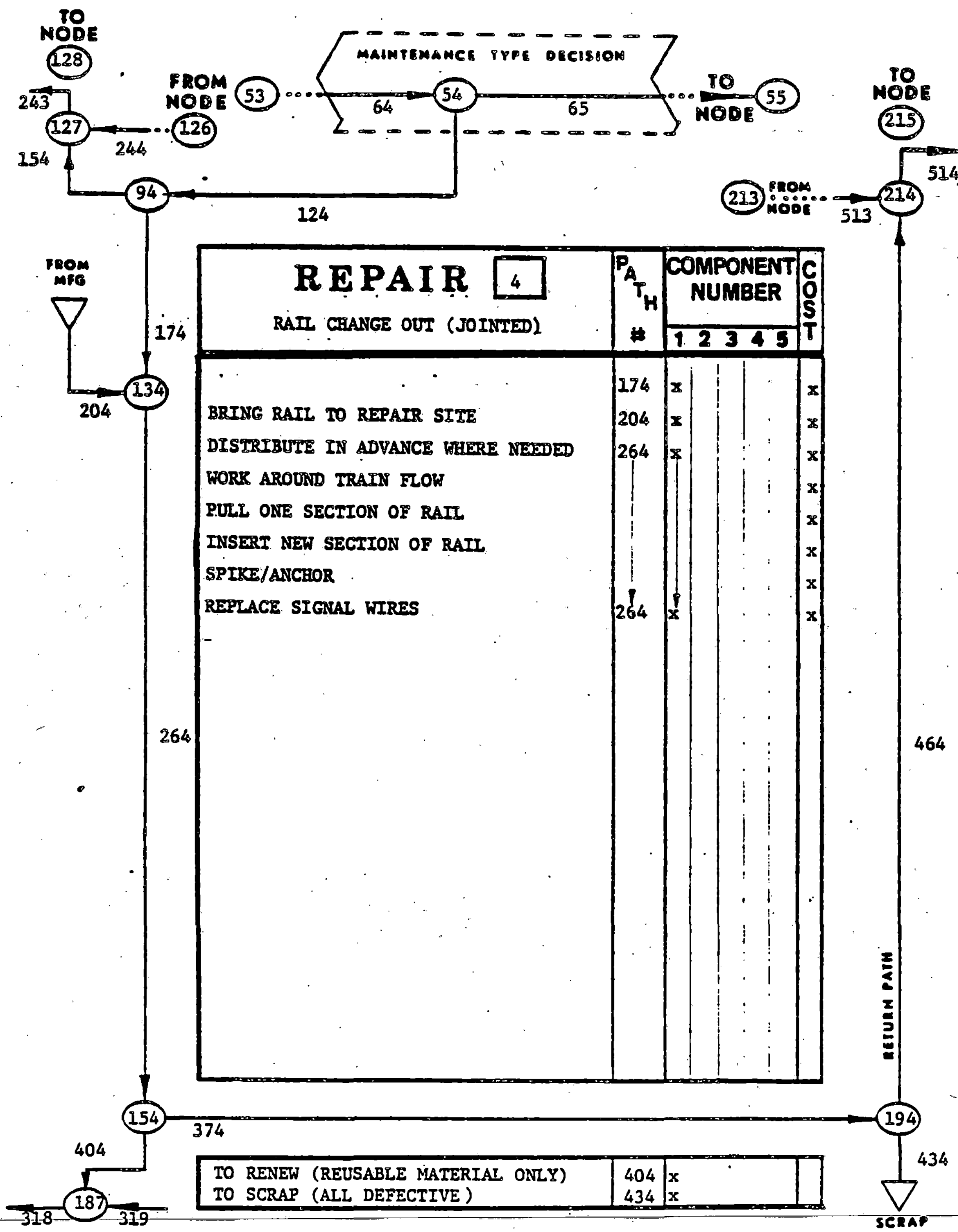
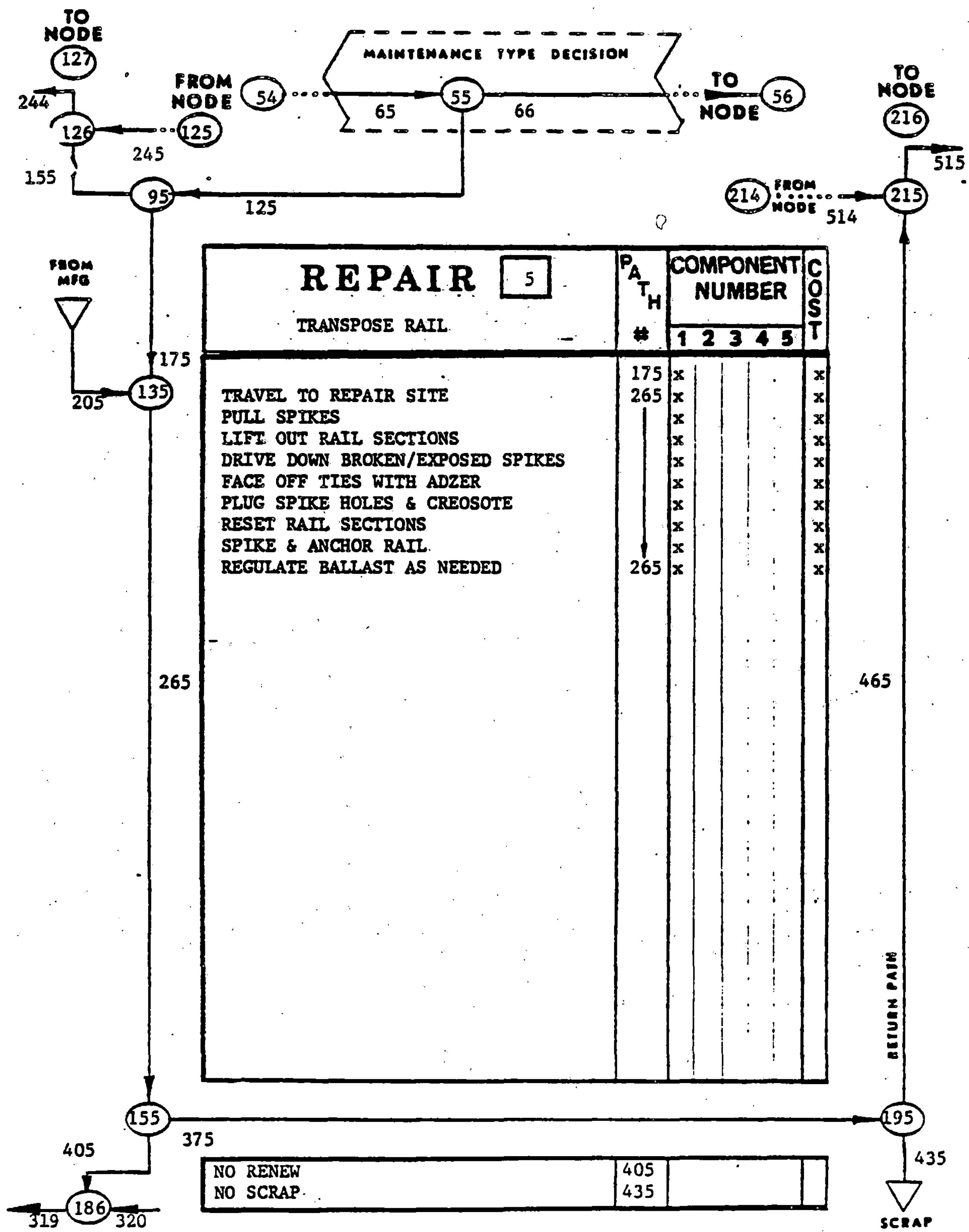


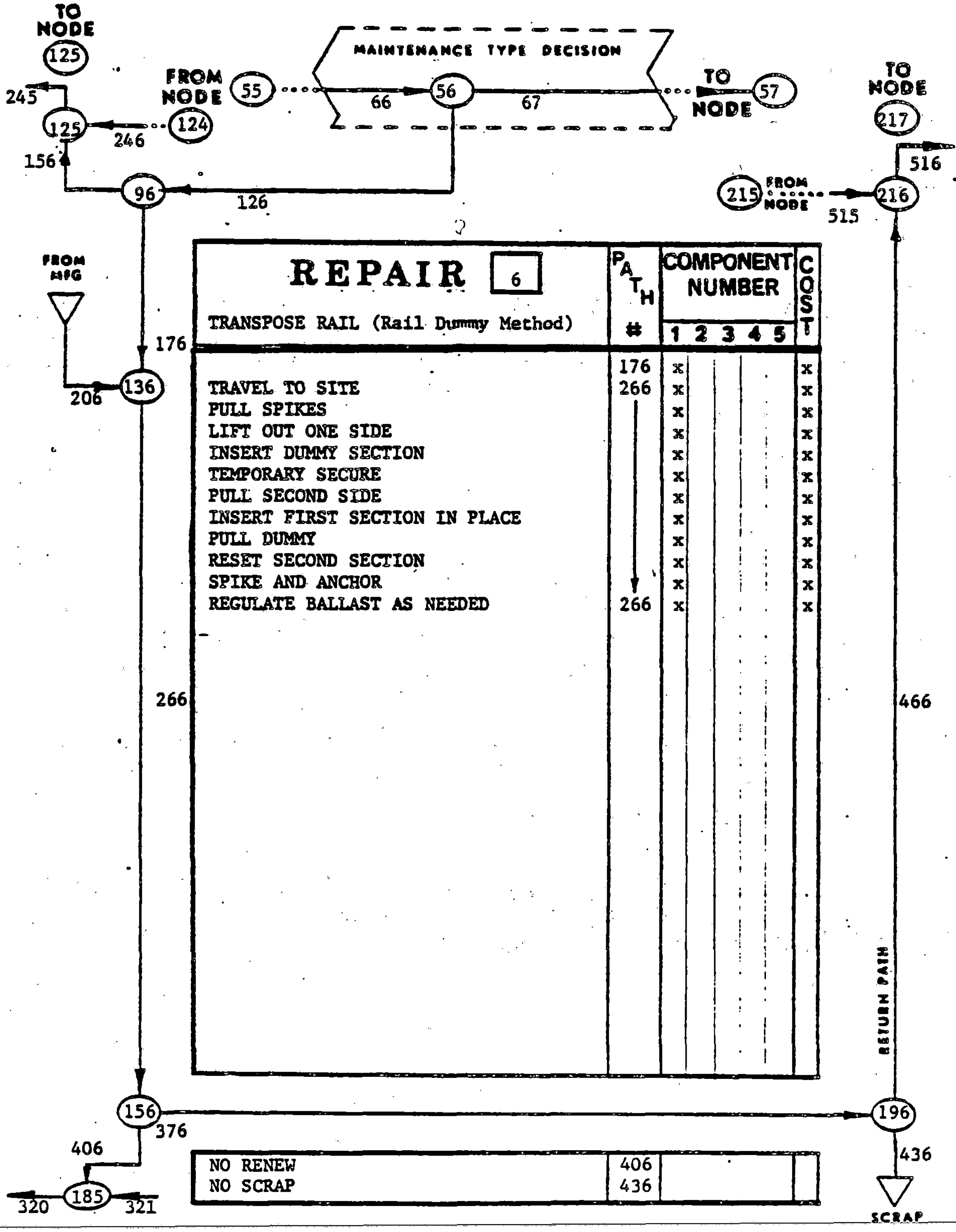
FIGURE K-7. REPAIR OPERATION 4



REPAIR 5 TRANSPOSE RAIL	PATH #	COMPONENT NUMBER 1	2	3	4	5	COST
	175	x					x
TRAVEL TO REPAIR SITE	265	x					x
PULL SPIKES	↓	x					x
LIFT OUT RAIL SECTIONS		x					x
DRIVE DOWN BROKEN/EXPOSED SPIKES		x					x
FACE OFF TIES WITH ADZER		x					x
PLUG SPIKE HOLES & CREOSOTE		x					x
RESET RAIL SECTIONS		x					x
SPIKE & ANCHOR RAIL		x					x
REGULATE BALLAST AS NEEDED	265	x					x

NO RENEW	405		
NO SCRAP	435		

FIGURE K-8. REPAIR OPERATION 5



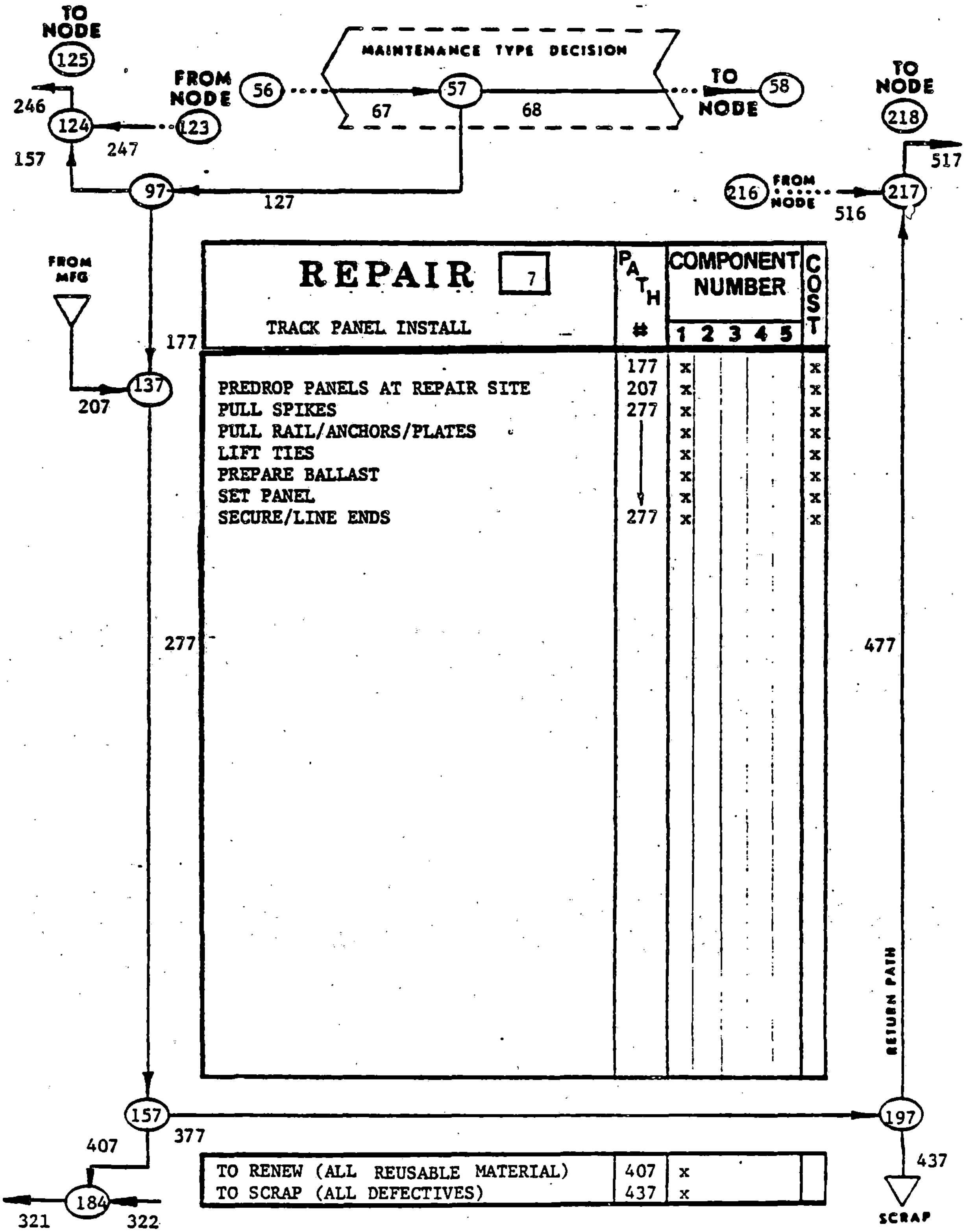
REPAIR 6

TRANSPOSE RAIL (Rail Dummy Method)

	PATH #	COMPONENT NUMBER 1	2	3	4	5	COST
	176	x					x
TRAVEL TO SITE	266	x					x
PULL SPIKES		x					x
LIFT OUT ONE SIDE		x					x
INSERT DUMMY SECTION		x					x
TEMPORARY SECURE		x					x
PULL SECOND SIDE		x					x
INSERT FIRST SECTION IN PLACE		x					x
PULL DUMMY		x					x
RESET SECOND SECTION		x					x
SPIKE AND ANCHOR		x					x
REGULATE BALLAST AS NEEDED	266	x					x

NO RENEW	406		
NO SCRAP	436		

FIGURE K-9. REPAIR OPERATION 6



REPAIR 7 TRACK PANEL INSTALL	PATH #	COMPONENT NUMBER 1	2	3	4	5	COST
	177	x					x
PREDROP PANELS AT REPAIR SITE	207	x					x
PULL SPIKES	277	x					x
PULL RAIL/ANCHORS/PLATES		x					x
LIFT TIES		x					x
PREPARE BALLAST		x					x
SET PANEL		x					x
SECURE/LINE ENDS	277	x					x

TO RENEW (ALL REUSABLE MATERIAL)	407	x	
TO SCRAP (ALL DEFECTIVES)	437	x	

FIGURE K-10. REPAIR OPERATION 7

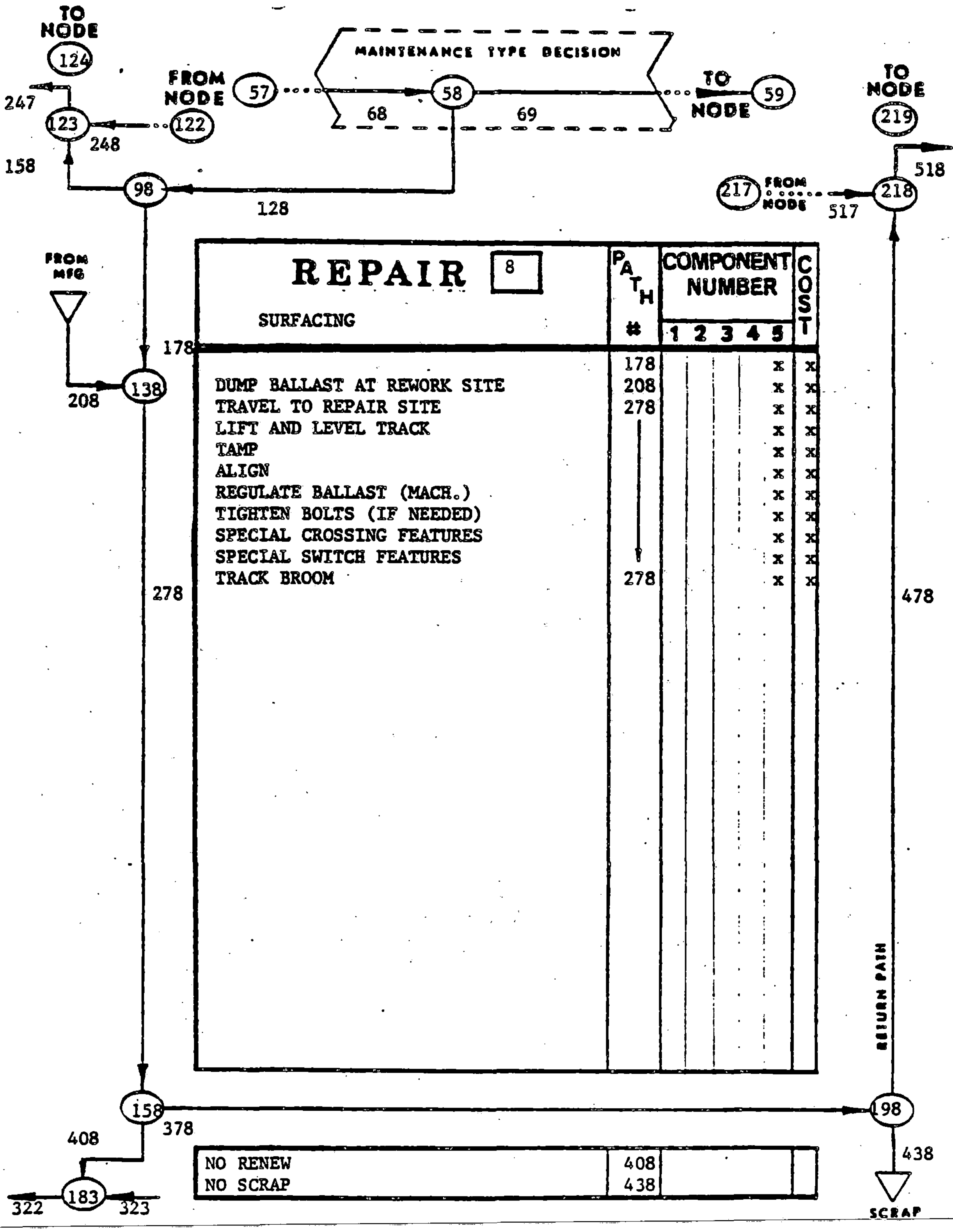


FIGURE K-11. REPAIR OPERATION 8

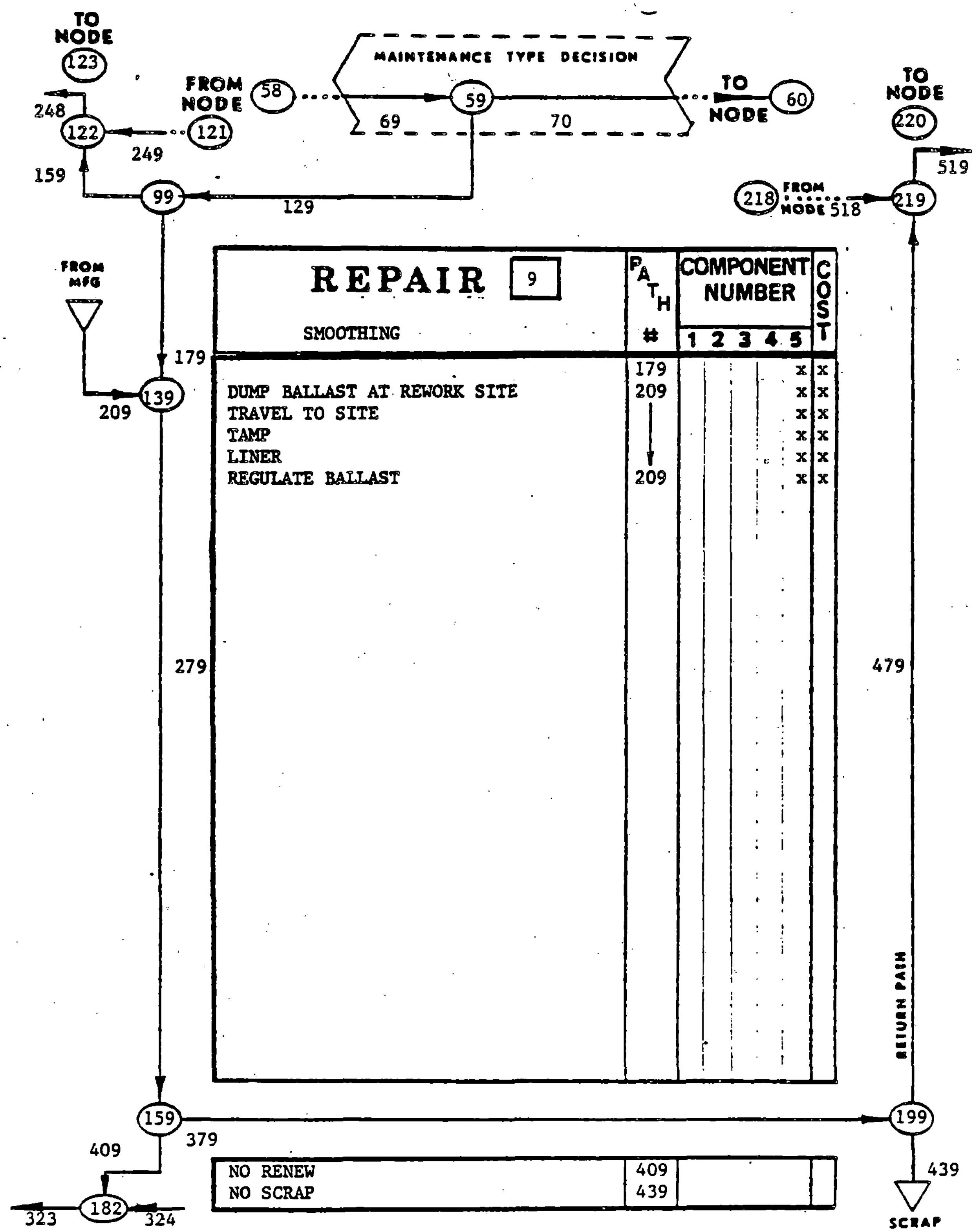
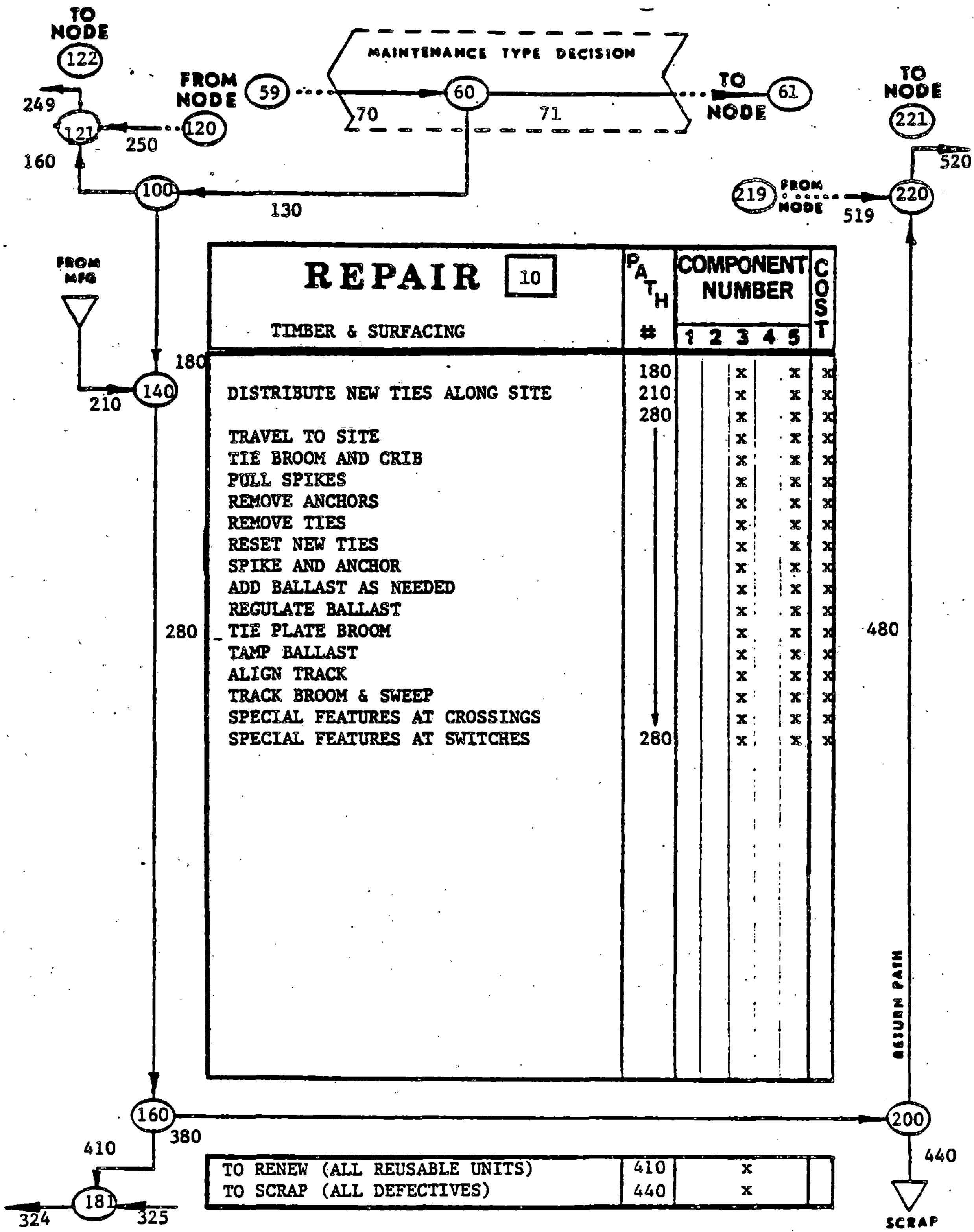


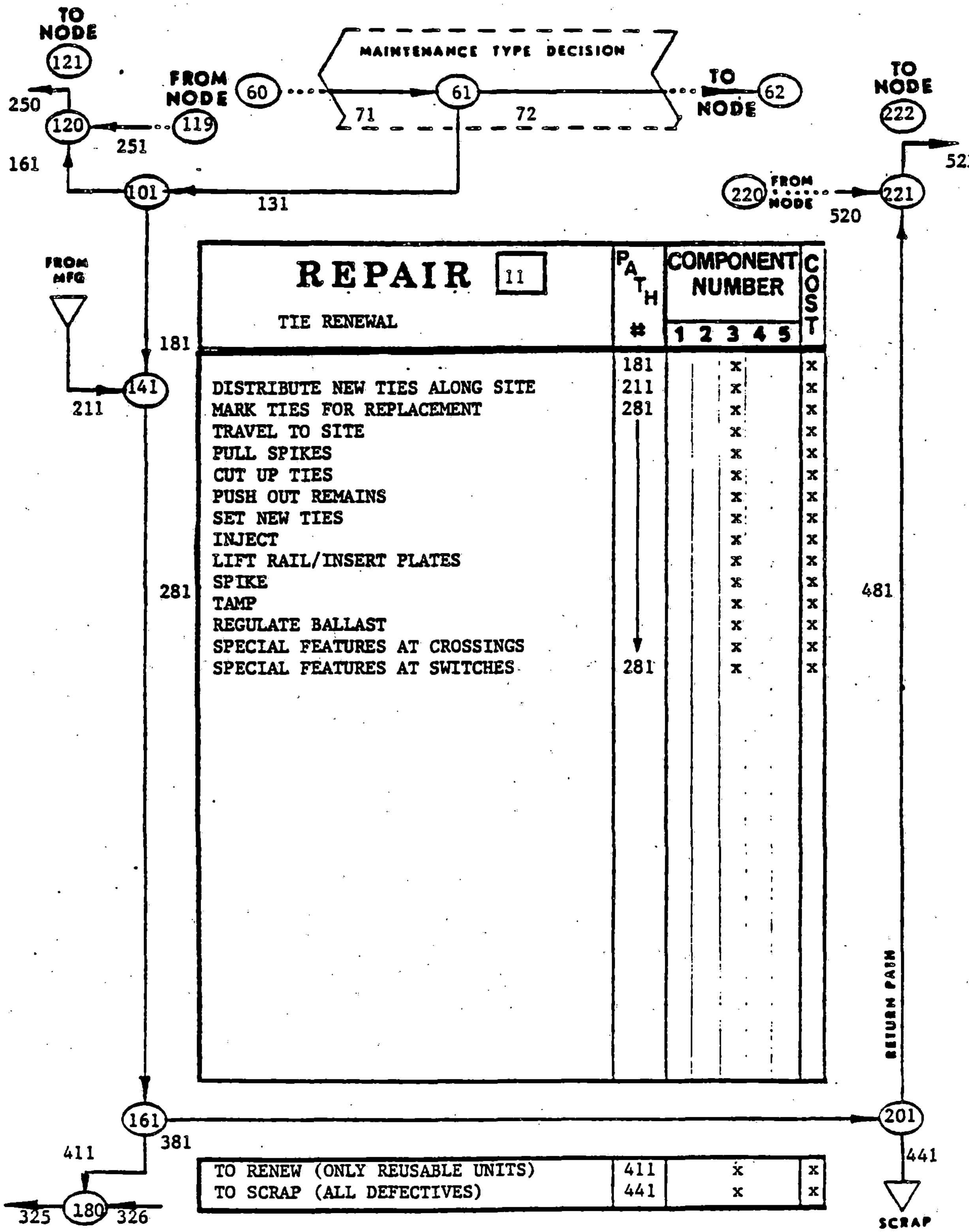
FIGURE K-12. REPAIR OPERATION 9



REPAIR 10 — TIMBER & SURFACING	PATH #	1	2	3	4	5	COST
	180			x		x	x
DISTRIBUTE NEW TIES ALONG SITE	210			x		x	x
	280			x		x	x
TRAVEL TO SITE				x		x	x
TIE BROOM AND CRIB				x		x	x
PULL SPIKES				x		x	x
REMOVE ANCHORS				x		x	x
REMOVE TIES				x		x	x
RESET NEW TIES				x		x	x
SPIKE AND ANCHOR				x		x	x
ADD BALLAST AS NEEDED				x		x	x
REGULATE BALLAST				x		x	x
TIE PLATE BROOM				x		x	x
TAMP BALLAST				x		x	x
ALIGN TRACK				x		x	x
TRACK BROOM & SWEEP				x		x	x
SPECIAL FEATURES AT CROSSINGS				x		x	x
SPECIAL FEATURES AT SWITCHES	280			x		x	x

TO RENEW (ALL REUSABLE UNITS)	410			x			
TO SCRAP (ALL DEFECTIVES)	440			x			

FIGURE K-13. REPAIR OPERATION 10



REPAIR 11 — TIE RENEWAL	PATH #	COMPONENT NUMBER 1	2	3	4	5	COST
	181			x			x
DISTRIBUTE NEW TIES ALONG SITE	211			x			x
MARK TIES FOR REPLACEMENT	281			x			x
TRAVEL TO SITE	↓			x			x
PULL SPIKES				x			x
CUT UP TIES				x			x
PUSH OUT REMAINS				x			x
SET NEW TIES				x			x
INJECT				x			x
LIFT RAIL/INSERT PLATES				x			x
SPIKE				x			x
TAMP				x			x
REGULATE BALLAST				x			x
SPECIAL FEATURES AT CROSSINGS				x			x
SPECIAL FEATURES AT SWITCHES	281			x			x

TO RENEW (ONLY REUSABLE UNITS)	411			x			x
TO SCRAP (ALL DEFECTIVES)	441			x			x

FIGURE K-14. REPAIR OPERATION 11

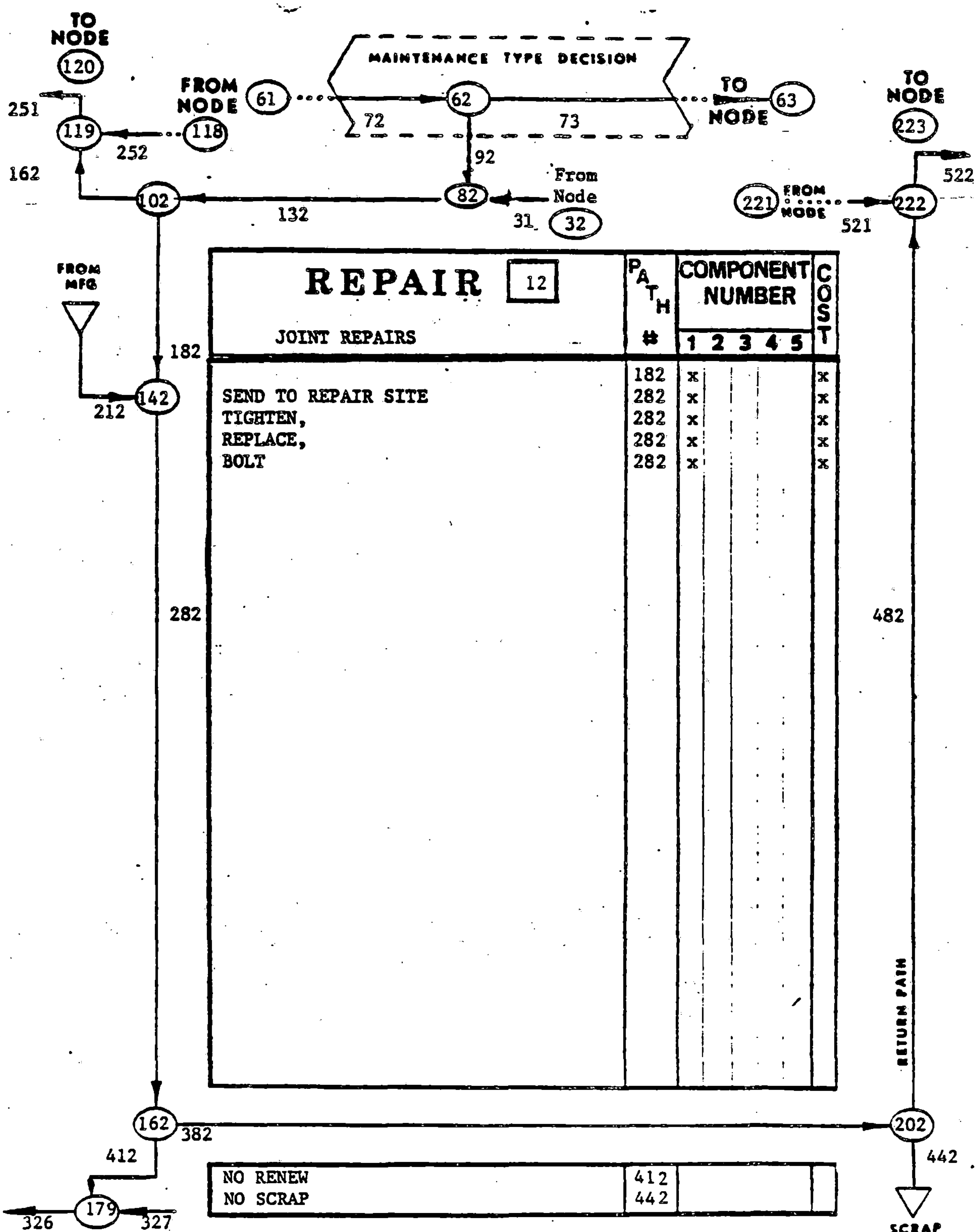


FIGURE K-15. REPAIR OPERATION 12

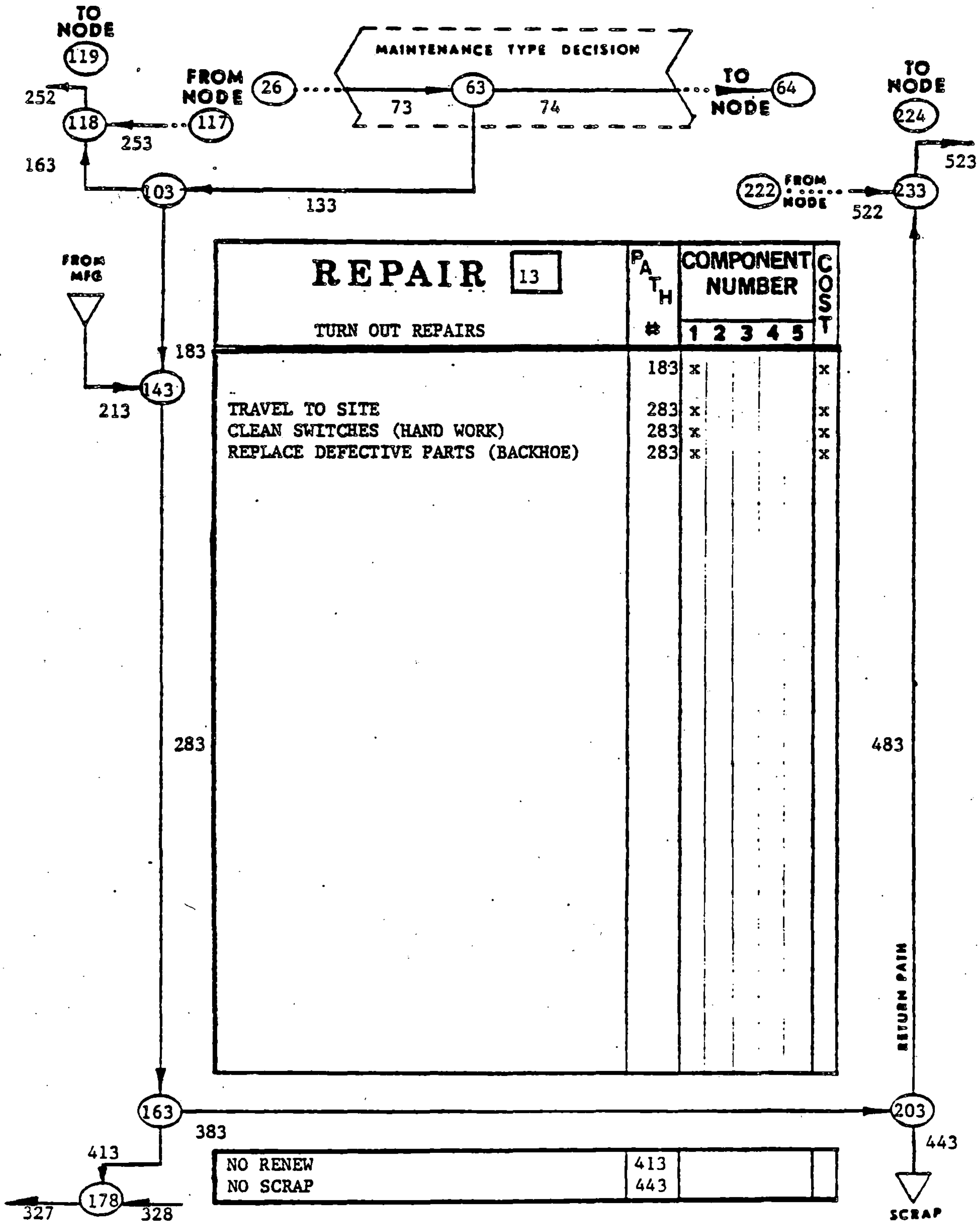
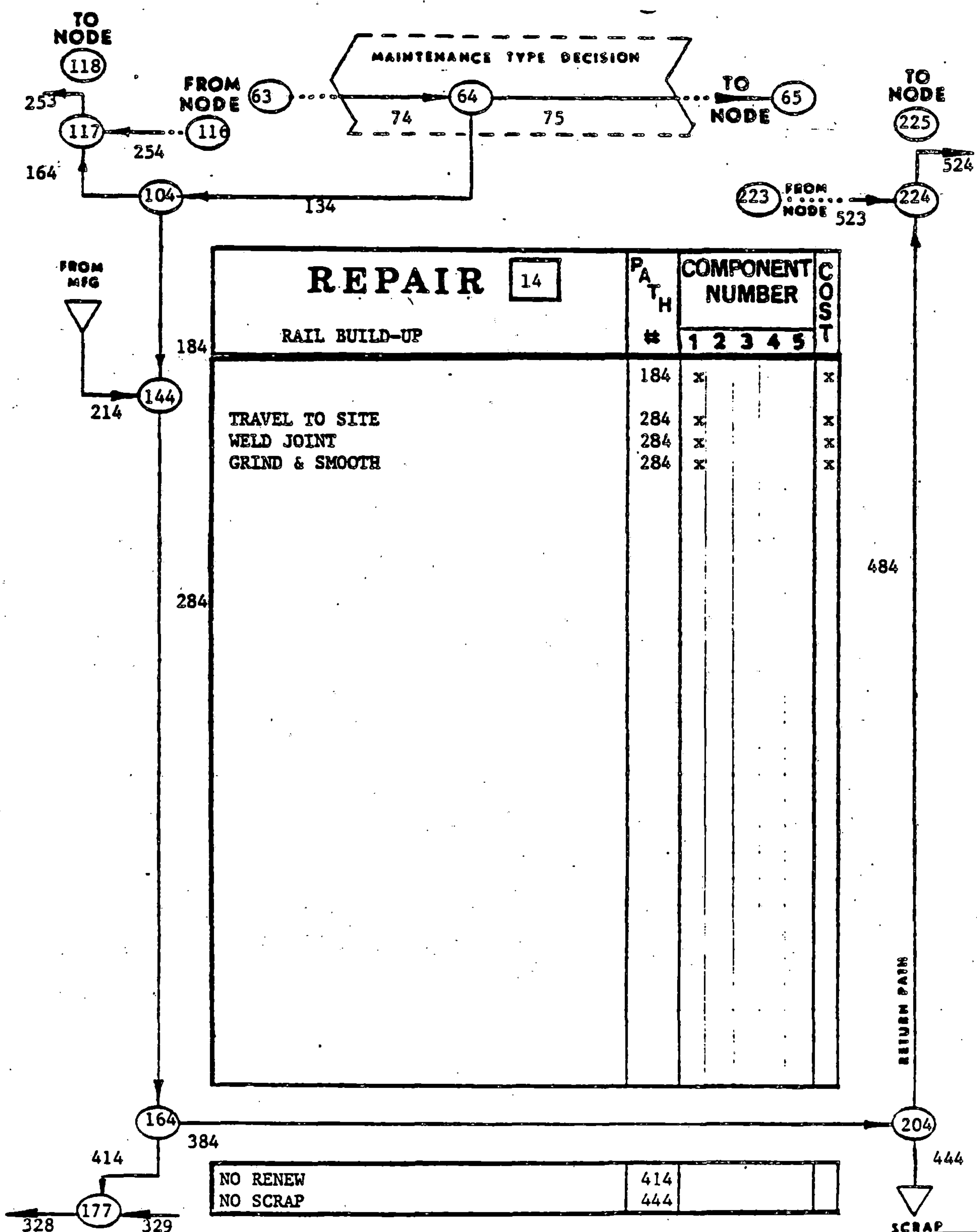


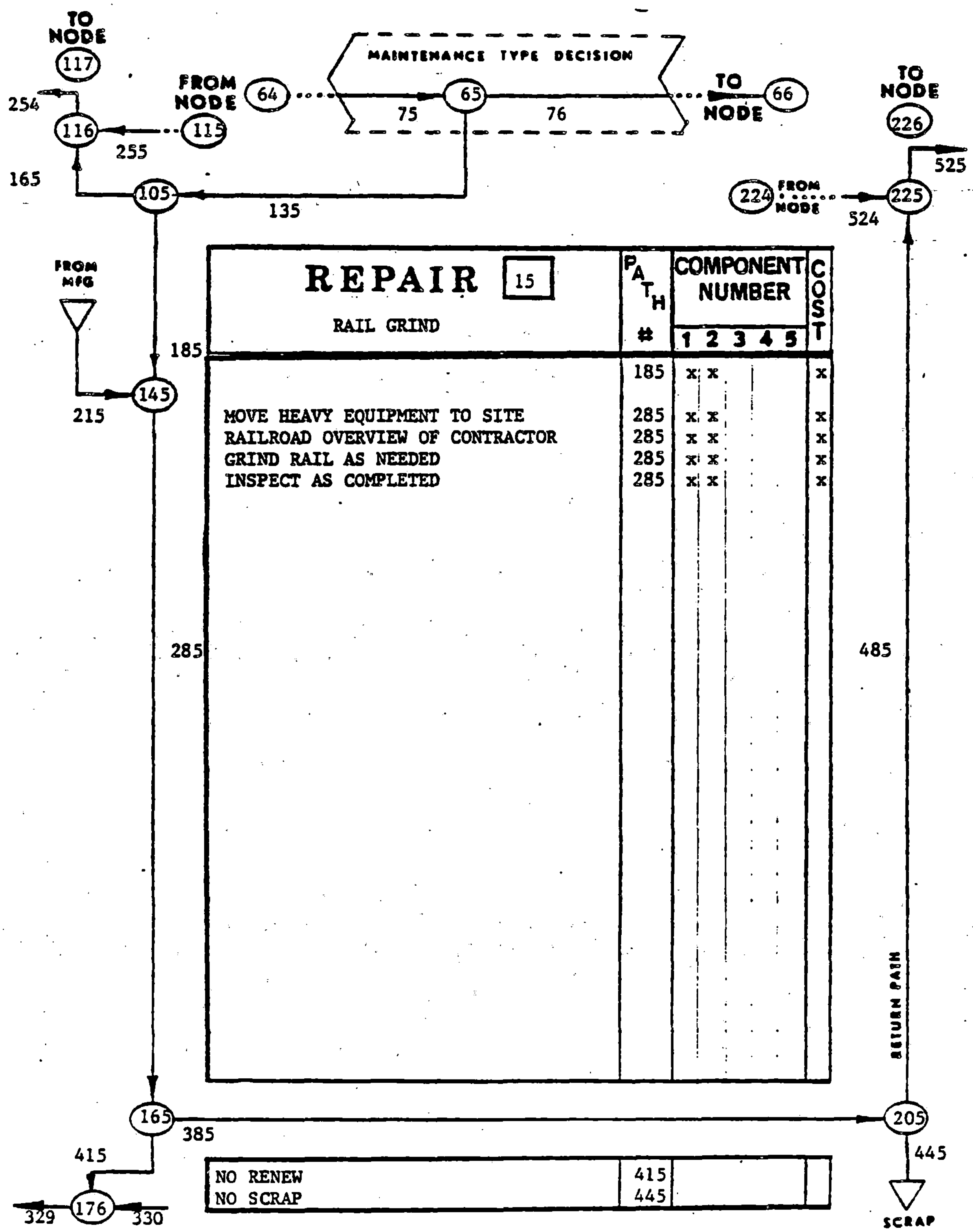
FIGURE K-16. REPAIR OPERATION 13



REPAIR 14 RAIL BUILD-UP	PATH #	COMPONENT NUMBER 1	2	3	4	5	COST
	184	x					x
TRAVEL TO SITE	284	x					x
WELD JOINT	284	x					x
GRIND & SMOOTH	284	x					x

NO RENEW	414		
NO SCRAP	444		

FIGURE K-17. REPAIR OPERATION 14



REPAIR 15 RAIL GRIND	PATH #	COMPONENT NUMBER 1	2	3	4	5	COST
	185	x	x				x
MOVE HEAVY EQUIPMENT TO SITE	285	x	x				x
RAILROAD OVERVIEW OF CONTRACTOR	285	x	x				x
GRIND RAIL AS NEEDED	285	x	x				x
INSPECT AS COMPLETED	285	x	x				x

NO RENEW	415		
NO SCRAP	445		

FIGURE K-18. REPAIR OPERATION 15

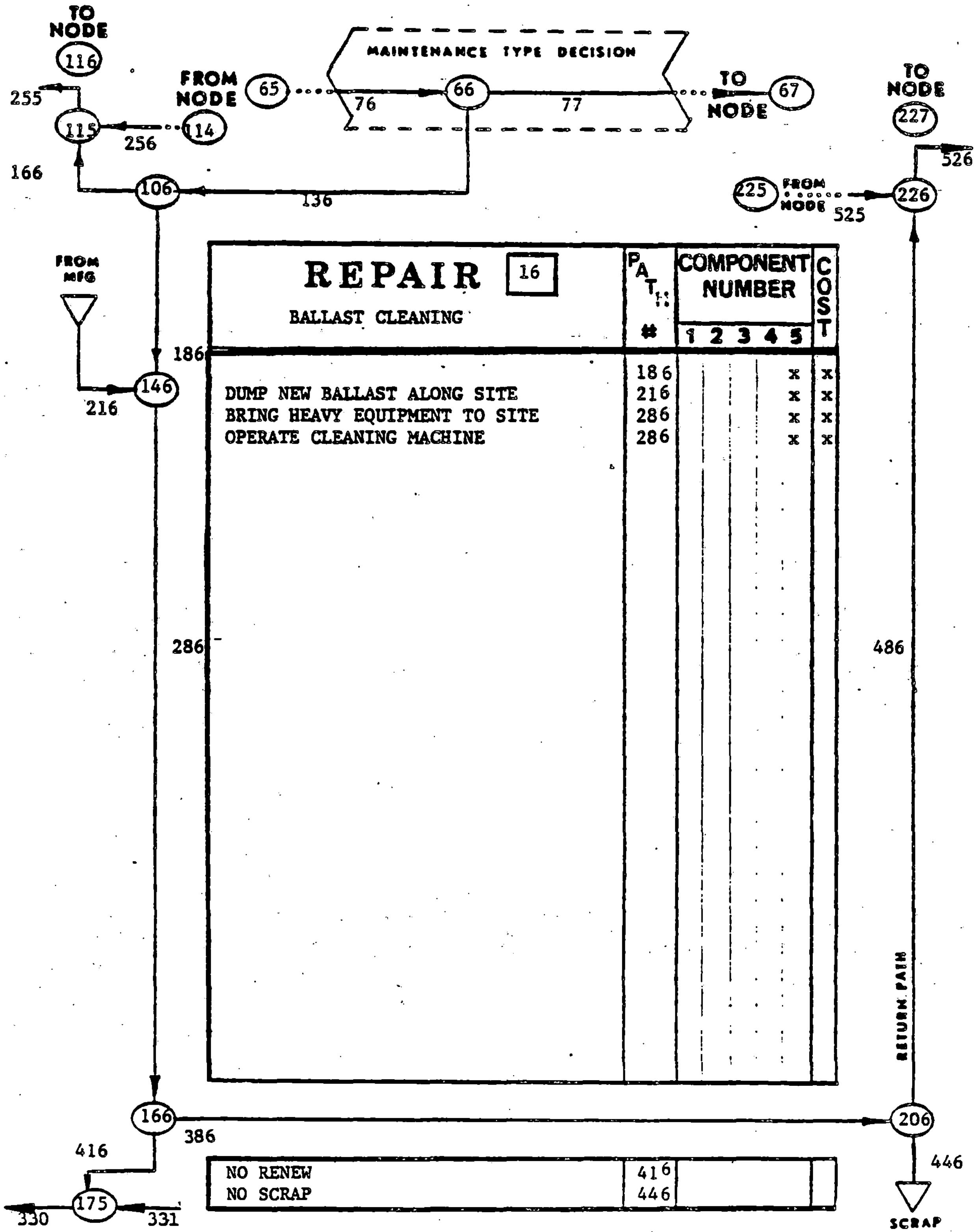


FIGURE K-19. REPAIR OPERATION 16

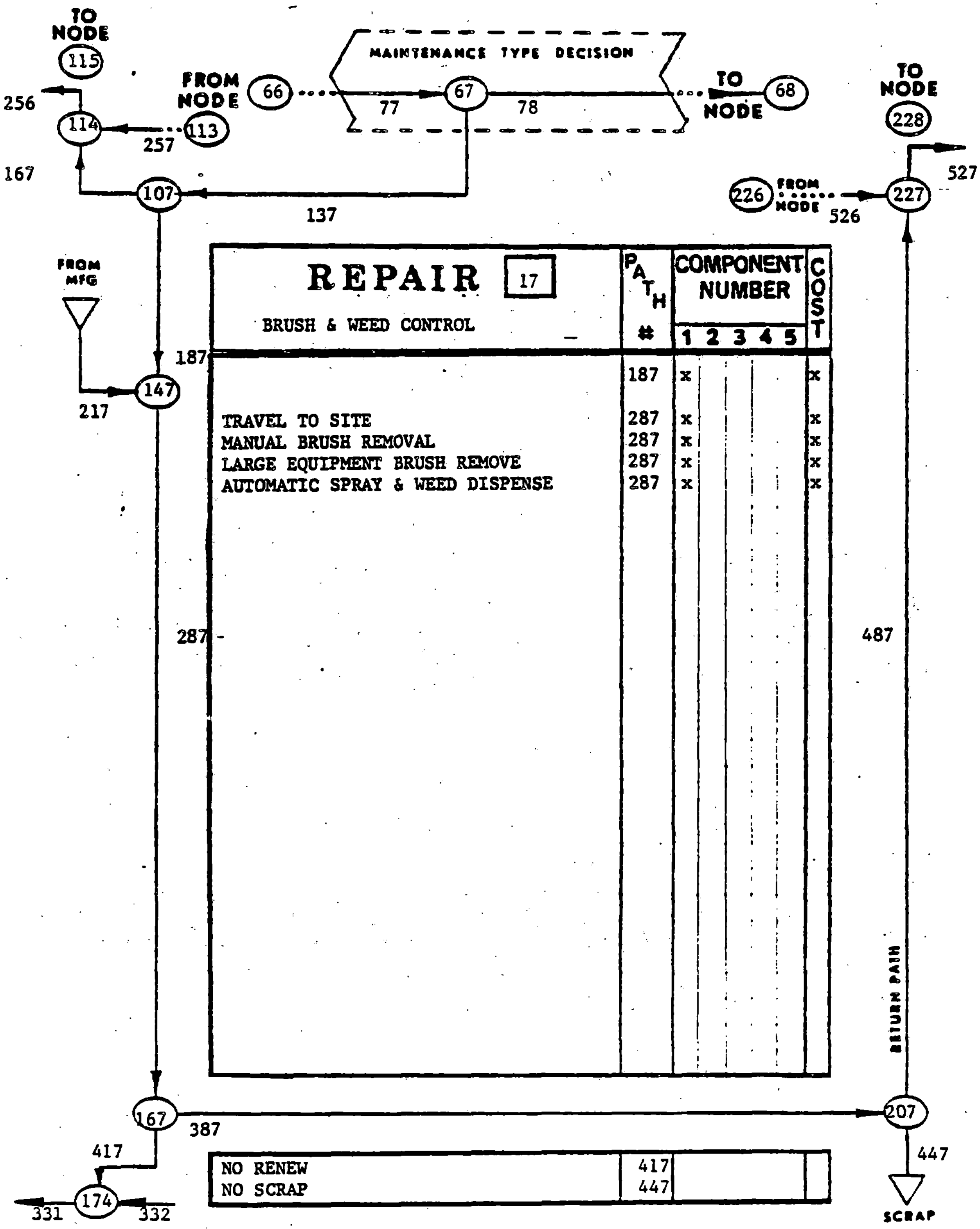


FIGURE K-20. REPAIR OPERATION 17

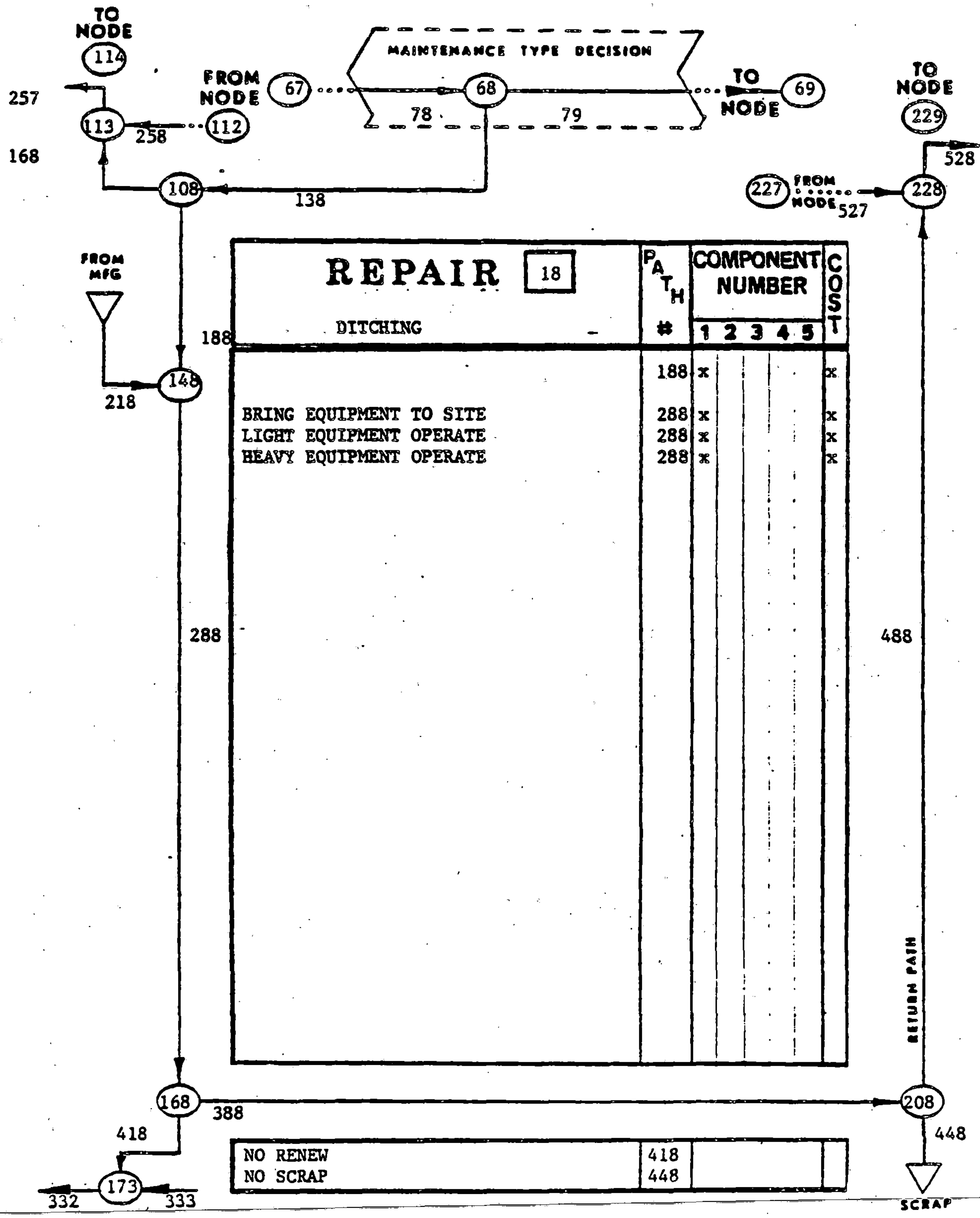
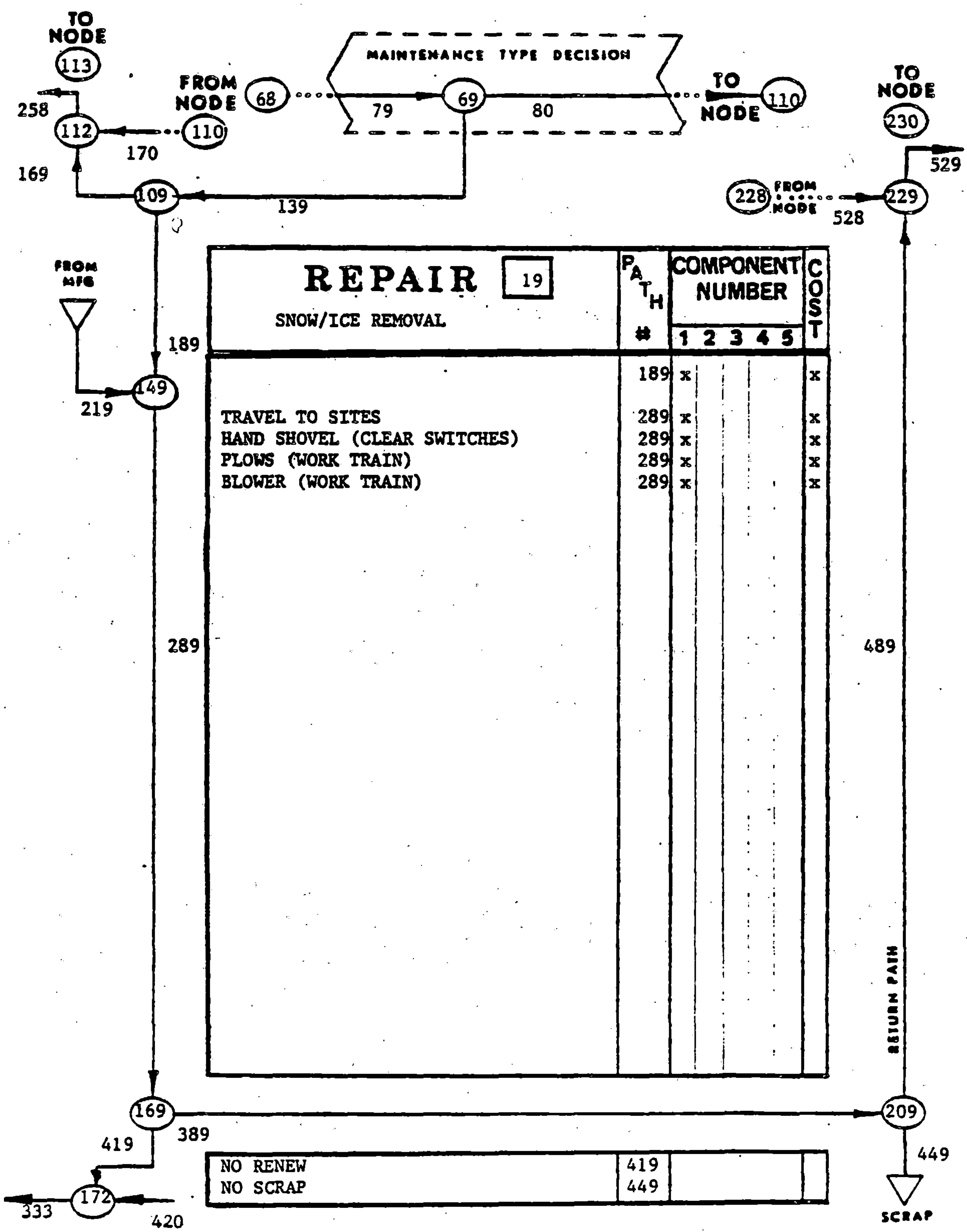


FIGURE K-21. REPAIR OPERATION 18



REPAIR 19 SNOW/ICE REMOVAL	PATH #	COMPONENT NUMBER 1	2	3	4	5	COST
	189	x					x
TRAVEL TO SITES	289	x					x
HAND SHOVEL (CLEAR SWITCHES)	289	x					x
PLOWS (WORK TRAIN)	289	x					x
BLOWER (WORK TRAIN)	289	x					x

NO RENEW	419		
NO SCRAP	449		

FIGURE K-22. REPAIR OPERATION 19

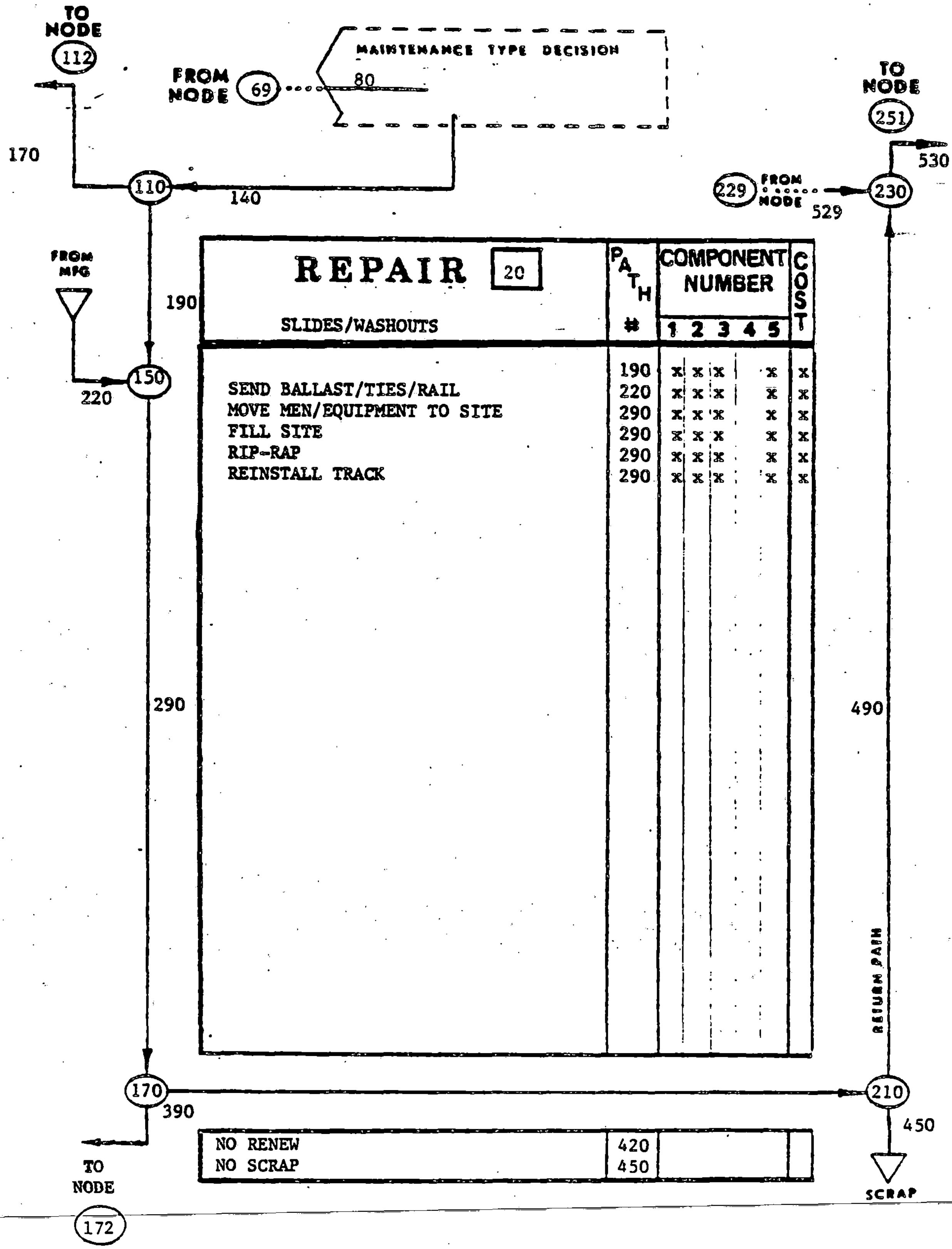


FIGURE K-23. REPAIR OPERATION 20

APPENDIX L

REPORT OF NEW TECHNOLOGY

The work described in this report concerns the application of a methodology, the simulation cost model (SCM), to the economic aspects of maintaining and/or laying of rail, ties, and ballast. Because the work was not concerned with devices, testing, or construction, no inventions were developed. However, the work did result in a methodology which can be applied to economic systems beyond those associated with a single track maintenance system. Given the proper input data most any track maintenance operation can be modeled and simulated by the computer program developed. Alternate systems most appropriately treated by the SCM consist mainly of large sets of individual components. The SCM methodology requires the user to draw a maintenance action diagram and provide an initial set of input data for its paths before running the computer program to obtain a cost simulation output. The output includes a quantitative description of current (present time) annual system operation costs and indicates quantitatively the most costly portions of the system as well as provides a projection of future system operating costs.

55 copies

User's Guide for A Computerized Track Maintenance Simulation Cost Methodology (Final Report), 1982
US DOT, FRA, Richard L smith, Allan I Krauter, Joseph Betor

U.S. Department
of Transportation

**Research and
Special Programs
Administration**

Kendall Square
Cambridge, Massachusetts 02142

Official Business
Penalty for Private Use $300

PROPERTY OF FRA
RESEARCH & DEVELOPMENT
LIBRARY

Postage and Fees Paid
Research and Special
Programs Administration
DOT 513