RISK ANALYSIS IN CONSTRUCTION

by

Saleh Bashir Awad Gaderbouh

A thesis submitted for the degree of

Master of Science

Department of Civil Engineering

University of Glasgow

January 1993

ACKNOWLEDGEMENTS

This work reported in this thesis was carried out in the Department of Civil Engineering University of Glasgow.

The author wishes to express his special thanks and appreciation to his supervisor Dr. K. J. Williams, for his valuable guidance and constant encouragement throughout this work. Thanks should also go to him for undertaking much of the 'graphic' work in PERTRA1 and PETRA2.

The author is also grateful to professor Muir Wood (Cormack professor and Head of the Department) and professor D. Green (previous Head of the Department) for allowing the use of the Department facilities.

The author is grateful to his fellow researchers Dr. Zhang, Dr. H. Jasem, Dr. Mesavi, Dr. O. Famiyesin, M. Abdulkader, A. Bensalem, K. Belkhair, A. Hamidon, George Frangopoulos, Giovainni Schlipoti, P. Addison and X. Leng for their helpful discussions and constant encouragement.

The author is grateful to Ras Lanuf Oil and Gas Processing Company and to Jawaby Oil Company for the financial support during the course.

Finally my thanks reserved for my Mother, my Father, my Sister my Brothers, my Wife Taisser and my Daughters Hind and Orayda for their patience and constant encouragement throughout my studies.

SUMMARY

The work reported in this thesis centres around the development of a risk analysis computer software package which operates in conjunction with PERTMASTER ADVANCE, a widely-used commercial project planning software package.

The risk analysis technique used herein is the so-called 'Monte Carlo simulation technique' whereby a number of project simulations, each of which makes use of a set of project data values (such as task durations) chosen randomly from within specified probability distributions, is carried out with a view to producing a particular solution (such as the project duration) defined by a single probability distribution. In such a way, the degree of uncertainty associated with the particular solution, which results from the individual degrees of uncertainty associated with each of the data values on which such solution is based, can be determined.

The software package comprises the following programs:

PERTRA1 – an interactive program which abstracts specific project data from a datafile generated by PERTMASTER ADVANCE.

PERTRA2 – an interactive program which facilitates the keyboard input of task duration probability distribution data.

PERTRA3 — a 'number—crunching' program which carries out the risk analysis on the basis of data generated by PERTRA1 and PERTRA2. The results of the analysis are written to a user—specified datafile in such a form as to enable subsequent graphical presentation thereof by widely available commercial 'graphics' software packages.

All three programs are written in FORTRAN 77 and can be run on any IBM—compatible personal computer using the MS—DOS version 3.3 or higher.

TABLE OF CONTENTS

CONTENTS PAGE NO

ACKNOWLEDGMENTS i

SUMMARY ii

TABLE OF CONTENTS iv

LIST OF FIGURES xiii

LIST OF TABLES xviii

CHAPTER 1: INTRODUCTION TO RISK ANALYSIS 1

1.1 Introduction 1

1.2 Sources of Uncertainties 3

1.2.1 Accuracy of the Data 3

1.2.2 Inherent Uncertainties 5

1.3 Risk Analysis 6

1.3.1 Risk Analysis Techniques 7

1.3.1.1 Sensitivity Analysis 7

1.3.1.2 Probability Analysis 9

CHAPTER 2: NETWORK ANALYSIS 12

2.1 Historical Background to Critical Path Analysis 12

2.2 The Network Model 14

2.2.1 Activity on Arrow Networks 14

2.2.2 Activity on the Node Networks 16

2.2.3 A Comparison Between the Two Networks 17

2.3 The Analysis of Simple Networks 18

2.3.1 The Forward Pass 19

2.3.2 The Backward Pass 20

2.4 Criticality and Float 21

2.5 Improvement to Realism 22

2.5.1 Complex Links 23

2.5.2 Project Close– down 26

2.5.3 Window Times 27

2.5.4 Calendar Dates 27

2.5.5 Seasonal Variations 28

CHAPTER 3: CASH FLOW FORECASTING 30

3.1 The Need for Cash Flow Forecasting 30

3.2 The Requirement for a Forecasting System 32

3.2.1 The Data Needed for Cash Flow Calculations 33

3.3 Capital Lock– up 39

3.3.1 Factors Affect Capital Lock– up 42

3.3.1.1 Margin 42

3.3.1.2 Retention 43

3.3.1.3 Claims 43

3.3.1.4 Front–end Rate Loading 45

3.3.1.5 Overmeasurment 46

3.3.1.6 Back–end Rate Loading 47

3.3.1.7 Delay in Receiving Payments from clients 47

3.3.1.8 Delay in Baying Labour, Plant Hires, Material Supplies and Subcontractors 49

3.3.1.9 Inflation 49

3.4 Internal Rate of Return (IRR) 52

CHAPTER 4: THE ANALYSIS OF UNCERTAINTIES 54

4.1 The Two Approaches 54

4.1.1 The Statistical Method 55

4.1.1.1 Interfering Critical Paths 58

4.1.2 Simulation Techniques 61

4.2 Monte Carlo Simulation 62

4.3 The Difficulties of Simulation 63

4.3.1 The Necessary Number of Simulations 64

4.3.2 Data Storage 64

4.3.3 Distribution Calculation 65

4.4 Triangular Probability Distribution 67

4.4.1 Calculation of the Parameters 67

4.5 Random Selection of Task Duration 71

4.5.1 Equiprobability Bands 72

4.5.2 Direct Calculation Method 73

4.5.3 Comparison 74

4.5.4 Selection of Task Duration Using the Direct Calculation Method 75

CHAPTER 5: THE COMPUTER PROGRAMS 76

5.1 Introduction 76

5.2 PERTRA1: Data reading and initialisation 78

5.2.1 Subroutine HEADER 80

5.2.2 Subroutine READEXP 81

5.2.2.1 Subroutine ABTASK 83

5.2.2.2 Subroutine ABTRD 85

5.2.2.3 Subroutine ABLINK 86

5.2.2.4 Subroutine ABRES 89

5.2.2.5 Subroutine ABSUBRES 90

5.2.2.6 Subroutine ABCAL 91

5.2.3 Subroutine DIMCHK 91

5.2.4 Subroutine CALWKS 92

5.2.5 Subroutine TCOSTS 93

5.2.6 Subroutine RESCOST 95

5.2.7 Subroutine LSORT 105

5.2.8 Subroutine QUIT 109

5.3 PERTRA2: Interactive input of task duration Data 110

5.3.1 Subroutine HEADER 113

5.3.2 Subroutines FDISPLAY, FILTER, TDISPLAY 114

5.3.2.1 Subroutine FDISPLAY 114

5.3.2.2 Subroutine FILTER 116

5.3.2.3 Subroutine TDISPLAY 118

5.3.3 Subroutine QDISPLAY 119

5.3.4 Input of Task Duration Data 121

5.3.5 PERTRA2 Output 122

5.4 PERTRA3: Risk Analysis and Cash Flow Calculations 123

5.4.1 Calendar Subroutines and Functions 127

5.4.2.1 Subroutine CALEND 127

5.4.2.2 Function IDOTW 129

5.4.2 Subroutine ANALYS 129

5.4.2.1 Function LNKDRN 135

5.4.2.2 Function ISADD 135

5.4.2.3 Function ISDEDC 135

5.4.3 Subroutine TRIDIS 136

5.4.4 Subroutine DSELECT 137

5.4.4.1 Function RAND 137

5.4.5 Subroutine PCOSTS 138

5.4.6 Subroutine PSORT 140

5.4.7 Subroutine PRSTAT 140

5.4.8 Subroutine CASTAT 140

5.4.9 Subroutine RRSTAT 140

5.4.10 Subroutine RRSORT 141

5.4.11 Subroutine FREQUA 141

5.4.12 Subroutine FRQIRR 142

5.4.13 Subroutine FRQCAP 142

CHAPTER 6: PROGRAM OPERATION 143

6.1 PERTRA1 143

6.1.1 Screen Displays 143

6.1.1.1 Data Input Line 144

6.1.1.2 Screen Display Table 145

6.1.1.3 Message Line 145

6.1.2 Operational Details 146

6.1.2.1 Initialisation 146

6.1.2.2 Reading of Data File 148

6.1.2.3 Input of Working Week Data 150

6.1.2.4 Input of Cost Component Data 152

6.1.2.5 Generation of Data Files and Program Termination 156

6.2 PERTRA2 157

6.2.1 Screen Displays 157

6.2.2 Operational Details 157

6.2.2.1 Program Introduction 157

6.2.2.2 Reading Task Duration Data 160

6.2.2.3 Data Input 160

6.2.2.4 Input of Tasks with VARIABLE Durations 162

6.2.2.4.1 Filtration Parameters 163

6.2.2.4.2 Filtration Parameters Logic 168

6.2.2.4.3 Input of Optimistic and Pessimistic Durations Factors 172

6.2.2.5 Input of Tasks with INVARIABLE Durations 175

6.2.2.6 Updating of Task Data File and Program Termination 177

6.3 PERTRA3 178

6.3.1 Program Operation 178

6.3.2 Data Input 178

6.3.2.1 Random Number Series Type 178

6.3.2.2 Number of Simulations 179

6.3.3 Data Output 179

6.3.3.1 One Simulation Run Data Output 179

6.3.3.2 Multiple Simulation Runs Output 180

CHAPTER 7: CONCLUSIONS AND RECOMMENDATIONS 184

7.1 Conclusion 184

6.2 Recommendations 185

REFERENCES 187

APPENDICES 189

Appendix A Contents of PMA– Generated Export File 189

Appendix B Samples of Data Files Gnenrated by PERTRA1 200

Appendix B.1 Sample of the General Data Written to the File FLNAME.GEN 200

Appendix B.2 Sample of the Task Data Written to the File FLNAME.TSK 201

Appendix B.3 Sample of the Link Data Written to the File FLNAME.LNK 202

Appendix B.4 Sample of the Cost Data Written to the File FLNAME.CST 203

Appendix C Program PERTRA1 and Subroutines 204

Appendix C.1 Program PERTRA1 204

Appendix C.2 Subroutine HEADER 208

Appendix C.3 Subroutine DIMCHK 212

Appendix C.4 Subroutine READEXP 215

Appendix C.5 Subroutine RECCOM 221

Appendix C.6 Subroutine ABTASK 222

Appendix C.7 Subroutine ABTRD 224

Appendix C.8 Subroutine ABLINK 226

Appendix C.9 Subroutine ABRES 228

Appendix C.10 Subroutine ABSUBRES 229

Appendix C.11 Subroutine ABCAL 230

Appendix C.12 Subroutine CALWKS 231

Appendix C.13 Subroutine TCOSTS 235

Appendix C.14 Subroutine QUIT 244

Appendix C.15 Subroutine RESCOST 248

Appendix C.16 Subroutine SRSWOP 250

Appendix C.17 Subroutine LSORT 251

Appendix C.18 Subroutine LSWOP 253

Appendix C.19 Subroutine CHRCNV 254

Appendix C.20 Subroutine CREAD 255

Appendix D Program PERTRA2 and Subroutines 257

Appendix D.1 Program PERTRA2 257

Appendix D.2 Subroutine HEADER 260

Appendix D.3 Subroutine QDISPLAY 264

Appendix D.4 Subroutine FDISPLAY 271

Appendix D.5 Subroutine FILTER 282

Appendix D.6 Subroutine TDISPLAY 284

Appendix D.7 Subroutine CHRCNV 292

Appendix D.8 Function INTCHR 293

Appendix D.9 Subroutine CREAD 294

Appendix E Sample of the Updated Task Data Written to the File FLNAME.TSK 296

Appendix F Program PERTRA3 and Subroutines 297

Appendix F.1 Program PERTRA3 297

Appendix F.2 Subroutine CALEND 304

Appendix F.3 Function IDOTW 306

Appendix F.4 Subroutine ANALYS 307

Appendix F.5 Function LNKDN 321

Appendix F.6 Function ISADED 322

Appendix F.7 Function ISDEDC 323

Appendix F.8 Subroutine TRIDIS 324

Appendix F.9 Subroutine DSELECT 325

Appendix F.10 Function RAND 326

Appendix F.11 Subroutine PCOSTS 327

Appendix F.12 Subroutine PSORT 334

Appendix F.13 Subroutine PRSTAT 335

Appendix F.14 Subroutine CASTAT 337

Appendix F.15 Subroutine RRSTAT 339

Appendix F.16 Subroutine RRSORT 341

Appendix F.17 Subroutine FREQUA 342

Appendix F.18 Subroutine FRQIRR 344

Appendix F.19 Subroutine FRQCAP 346

Appendix F.20 Function IDAY 348

Appendix G One Run Data Output Written to the File PERTRA3.RES 349

Appendix H Multiple Runs Output Written to the File PERTRA3.RES 350

LIST OF FIGURES

CHAPTER 1:

FIG. (1.1) Sensitivity Analysis Diagram (Spider Diagram) 8

FIG. (1.2) Probability Distribution 11

FIG. (1.3) Cumulative Probability Distribution 11

CHAPTER 2:

FIG. (2.1) Activity on Arrow Network 15

FIG. (2.2) Activity on Node Network 16

FIG. (2.3) Complex Links

(a) The Fundamental Network 24

(b) Precedence Network 24

FIG. (2.4) The Seasonal Variations in the Construction Productivity in the U.K. 29

CHAPTER 3:

FIG. (3.1) Project Pre– tender Program 34

FIG. (3.2) Project Cost– Revenue Pattern 37

FIG. (3.3) Project Cash Flow Diagram 40

FIG. (3.4) The Effect of Main Capital Lock– up 44

FIG. (3.5) The Effect of Front End Loading 46

FIG. (3.6) Effect of Client Payment Delay 48

FIG. (3.7) Project Cash Flow Chart with 8% Inflation 51

FIG. (3.8) IRR Calculation Flow Chart 53

CHAPTER 4:

FIG. (4.1) The Beta Distribution 57

FIG. (4.2) A Unique Critical Path 60

FIG. (4.3) Interfering Critical Paths 60

FIG. (4.4) Zone of Equal Probability 62

FIG. (4.5) Triangular and Beta Ditribution Compared 66

FIG. (4.6) Distribution Parameters 67

FIG. (4.7) Distribution Parameters Calculation Flow Chart 71

FIG. (4.8) Equi– Probability Bands 72

FIG. (4.9) Direct Calculation 73

FIG. (4.10) Selection of Task Duration Using Direct Calculation 76

CHAPTER 5:

FIG. (5.1) Program PERTRA1 Flow Chart 79

FIG. (5.2) Flow Chart for the Subroutine HEADER 80

FIG. (5.3) Subroutine READEXP Flow Chart 82

FIG. (5.4) Subroutine ABTASK Flow Chart 84

FIG. (5.5) Subroutine ABTRD Flow Chart 85

FIG. (5.6) Subroutine ABLINK Flow Chart 88

FIG. (5.7) Subroutine ABRES Flow Chart 89

FIG. (5.8) Subroutine ABSUBRES Flow Chart 90

FIG. (5.9) Subroutine CALWKS Flow Chart 92

FIG. (5.10) Subroutine TCOSTS Flow Chart 94

FIG. (5.11) Resource Hierarchy 95

FIG. (5.12) Program PERTRA2 Flow Chart 112

FIG. (5.13) Subroutine HEADER Flow Chart 113

FIG. (5.14) Subroutine FDISPLAY Flow Chart 117

FIG. (5.15) Subroutine TDISPLAY Flow Chart 118

FIG. (5.16) Subroutine QDISPLAY Flow Chart 120

FIG. (5.17) Program PERTRA3 Flow Chart 125

FIG. (5.18) Subroutine ANALYS Flow Chart 134

FIG. (5.19) Subroutine TRIDIS Flow Chart 136

FIG. (5.20) Subroutine PCOSTS Flow Chart 139

FIG. (5.21) Subroutine PRSTAT Flow Chart 141

FIG. (5.22) Subroutine FRQIRR Flow Chart 142

CHAPTER 6:

FIG. (6.1) Programs Screen Displays 144

FIG. (6.2) File Name Input Screen Display 147

FIG. (6.3) File Name Confirmation Screen Display 147

FIG. (6.4) Data Abstraction 1 Screen Display 148

FIG. (6.5) Data Abstraction 2 Screen Display 149

FIG. (6.6) Working Week Input Screen Display 150

FIG. (6.7) Working Week Confirmation Screen Display 151

FIG. (6.8) Cost Components Data Screen Display 153

FIG. (6.9) Cost Components Data Screen Display (2) 153

FIG. (6.10) Cost Components Data Screen Display (3) 154

FIG. (6.11) Cost Components Data Continuation Screen Display (1) 154

FIG. (6.12) Cost Components Data Continuation Screen Display (2) 155

FIG. (6.13) Termination Screen Display 156

FIG. (6.14) Program Introductory Screen 1 158

FIG. (6.15) Program Introductory Screen 2 158

FIG. (6.16) Program Introductory Screen 3 159

FIG. (6.17) Reading Data Screen 160

FIG. (6.18) Task Duration Data Screen 1 161

FIG. (6.19) Task Duration Data Screen 2 161

FIG. (6.20) Task Duration Data Screen 3 162

FIG. (6.21) First Filter Parameters Screen 1 163

FIG. (6.22) First Filter Parameters Screen 2 164

FIG. (6.23) First Filter Parameters Screen 3 165

FIG. (6.24) Second Filter Parameters Screen 1 166

FIG. (6.25) Second Filter Parameters Screen 5 167

FIG. (6.26) Filters Confirmation Screen 168

FIG. (6.27) Filters Logic Screen 169

FIG. (6.28) Filters Logic Confirmation Screen 170

FIG. (6.29) Sample of Tasks with VARIABLE Durations Display 171

FIG. (6.30) Optimistic Durations Factor Input Screen 172

FIG. (6.31) Pessimistic Durations Factor Input Screen 173

FIG. (6.32) Fctors Confirmation Screen 174

FIG. (6.33) Filtered Tasks Display Screen 175

FIG. (6.34) Coding Screen Display 176

FIG. (6.35) Program Termination Screen Display 177

FIG. (6.36) Project Durations Frequency Diagram 181

FIG. (6.37) Project Durations Cumulative Frequency Diagram 181

FIG. (6.38) Internal Rate of Return (IRR) Frequency Diagram 182

FIG. (6.39) IRR Cumulative Frequency Diagram 182

FIG. (6.40) CAPTIM Frequency Diagram 183

FIG. (6.41) CAPTIM Cumulative Frequency Diagram 183

LIST OF TABLES

TABLE (3.1)	Project Direct Cost Breakdown	34
TABLE (3.2)	Project Summary	35
TABLE (3.3)	Cost/Revenue Summary	36
TABLE (3.4)	Project Cash Flow	38
TABLE (3.5)	Cash Flow for Frontend Loaded Project	45
TABLE (3.6)	Inflationary Project Cash Flow	51

CHAPTER 1

INTRODUCTION TO RISK ANALYSIS

1.1 Introduction

Experience of many projects indicates poor performance in terms of achieving time and cost targets. Many cost and time overruns are attributable to either unforeseen events, which may or may not have been anticipated by an experienced project manager, or foreseen events for which uncertainty was not appropriately accommodated. It is suggested that a significant improvement to project management performance may result from greater attention to the whole process of risk management[9].

Risk and uncertainties do not occur only on large projects[9]. While size can be one of the major causes of risk, other factors include the complexity, speed of construction and location of the project.

Various authors have attempted to distinguish between risk and uncertainty and also between risk and speculative risk. However, in the practice of construction risk management, such distinctions are usually unnecessary and may even be unhelpful[9].

Risk and uncertainty are inherent features of all construction projects and it is suggested that the following are useful basic concepts:[9]

1. Risks and uncertainties are associated with specific events or activities which can be individually identified.

2. A risk event implies that there is a range of outcomes of the event, with each outcome having a probability of occurrence.

3. Some risks offer only the prospect of adverse consequences (loss), for example: structural collapse, bankruptcy, war, sea or flood damage; these may be of a low or high probability but are generally of high impact.

4. Many common construction risks offer the prospect of either loss or gain, for example: productivity of labour and plant, variation, inflation. These are typically of high probability but may be of low or high impact.

5. Because of the individual nature of construction projects, there is usually insufficient objective data to calculate the probability of occurrence of specific outcomes of risk events; some degree of subjective judgement is usually required.

Risks and their effects should be considered in all key decision points throughout the project and by all the parties involved in the decision making process.

1.2 Sources of Uncertainty

Construction is a process which takes place in conditions of extreme uncertainty. Since each construction project, to all intents and purposes, is unique, the planner of such a project is faced with problems which have never before occurred in exactly the same form in which they are presented to him, and he is thus obliged to make forecasts of actions and performance which rely on limited data. This data will become available to him as the work progresses, and the continuous flow of data is a prime reason for regarding the planning and control functions of management as two phases of the same operation. The continual flow of data brings the possibility of continual addition of detail and results in project plans[6].

The planner can not, in the face of the uncertainty of the data available to him, simply abandon decision making. The detailed data, on which he is dependent, is generated only when the action is begun, and the action will not start if all decision are postponed until all is clear and certain. The planner must at least formulate a strategy for the project. In most cases he must do significantly more than this and provide, for example, cost estimates of an accuracy sufficient for the purpose of tendering. The planner is faced with the worrying task of basing important decisions on what is, at best, incomplete data[6].

1.2.1 Accuracy of the Data

In most circumstances, the planner of a construction project can increase the confidence with which the data may be regarded by initiating further study of the problem. He could, for example, commission extra site investigation studies or obtain more detailed information about the supplies of materials or labour in the area of the site. However, this gathering of extra data may be difficult to justify since it is expensive and time consuming. Investment in data gathering is subject to severely diminishing returns because some uncertainties are inherent in the construction process itself and, as accuracy in other data areas is increased, these inherent uncertainties begin to swamp the estimate. The cost of pre–project data gathering is an important constraint for the planner. Under the competitive tendering system, there is a one in six chance that a given tender will be successful. For five tenders out of six, therefore, the planning work is likely to be abortive and the extra costs incurred thereby can only be recovered by increasing the overheads included in the next tender. If pre–project planning costs are too high, overheads will increase with respect to those of competitors and the probability of success will fall correspondingly, thus increasing the problem. The planner must, therefore, aim to optimise the accuracy/cost equation and be prepared to work with data less than certain[6].

In addition to the cost constraints on pre–project planning, there is a severe constraint upon the time available for such since the competitive tendering system allows very little time for the gathering of additional data. For few projects would the planner be allowed more than twelve weeks to familiarise himself with the project and the area on which the site is located. In these circumstances the planner must work with little more information than that provided by the Client (or the Engineer on his behaf), and contained in the contract documents, together with specialised knowledge of his craft that he has gained through experience of similar projects[6].

Obviously the situation concerning the gathering of data is less difficult if the project is being planned in an environment different from that of the construction industry competitive tendering regime. Time and cost constraints are less severe in these circumstances but they are still there, appearing as the problem, familiar in other design disciplines, of knowing when to 'stop thinking and start doing'. Even in this favourable environment, detailed deterministic data is not available concerning every parameter for, as has been remarked, some uncertainty is an inherent part of the construction process[6].

1.2.2 Inherent Uncertainties

Although the quality of most data can be increased by the gathering of more detailed information, the value of such an exercise is limited by the inherent and irreducible uncertainty of some of the project parameters.

The presence of one uncertain component in a sum prevents the elimination of uncertainty from the total and reduces the data enrichment value of other components. In these circumstances there is a level of accuracy which it is futile to exceed. Therefore, it is possible to increase the accuracy of some of the data available to the planner but not the accuracy of all the data and these inherent uncertainties affect the whole policy of pre—project planning and cost estimation[6].

The most obvious source of inherent uncertainty in Great Britain[6] is the weather. This is difficult to predict in the short term, impossible in the long term and, although the planner can reflect broad expectations in his plans, the unpredictability of detail must somehow be reflected in his plans. Similarly the effects of erratic materials supply, labour unrest, and plant breakdown must be included in the plans, even though the existence or extent of these events cannot be known *a priori*.

In view of the presence of uncertainty, and the difficulty of its removal, the planner must be aware of its effects. He may know the effects and chose to ignore them, but this should be a decision consciously made and not one made by default[6].

1.3 Risk Analysis

Many risks and uncertainties are quantifiable in terms of their effect on cost, time or revenue. Such risks can be analysed by measuring their effects on the parameters used to assess project or contract viability (profitability indicators)[9].

The common principle behind any technique for the analysis of risk is to allow a range of values to the input data. For example, durations and costs for specific activities, or estimated revenues accruing from the project, are not given single values, but are considered over a range of values within which the decision maker believes they are likely to lie. An extension of this principle is to apply some statistical probability to any values used, so as to weight the possible outcomes to a more likely result[9].

However, for many construction risks there is insufficient objective data upon which calculations can be made of the probability of occurrence of outcomes. This leads some practitioners to be sceptical of the value of any form of risk analysis since, by implication, the results must be dependent on some degree of subjectivity regarding the input data[9].

1.3.1 Risk Analysis Techniques

Techniques for risk analysis have been theoretically established for a number of years, but their practical application to construction projects has been limited. Two of these are Sensitivity Analysis and Probability Analysis.

1.3.1.1 Sensitivity Analysis

Sensitivity analysis[9] seeks to place a value on the effect of a variation of a single variable parameter within a project by analysing that effect on the project plan. It is the simplest form of risk analysis.

Uncertainty and risk are reflected by defining a likely range of variation for each component of the original base case estimate. In practice, such an analysis is only done for those variables which have a high impact on cost, time or economic return and to which the project will be most sensitive. The effect of change of each of these variable on the final cost or time criteria is then assessed in turn, across the assumed ranges.

If the effects of changes to several parameters are examined, a so called 'spider' diagram[9] (see Figure 1.1) constitutes the most useful method of presenting the results such that the most 'critical' variables may be readily identified.

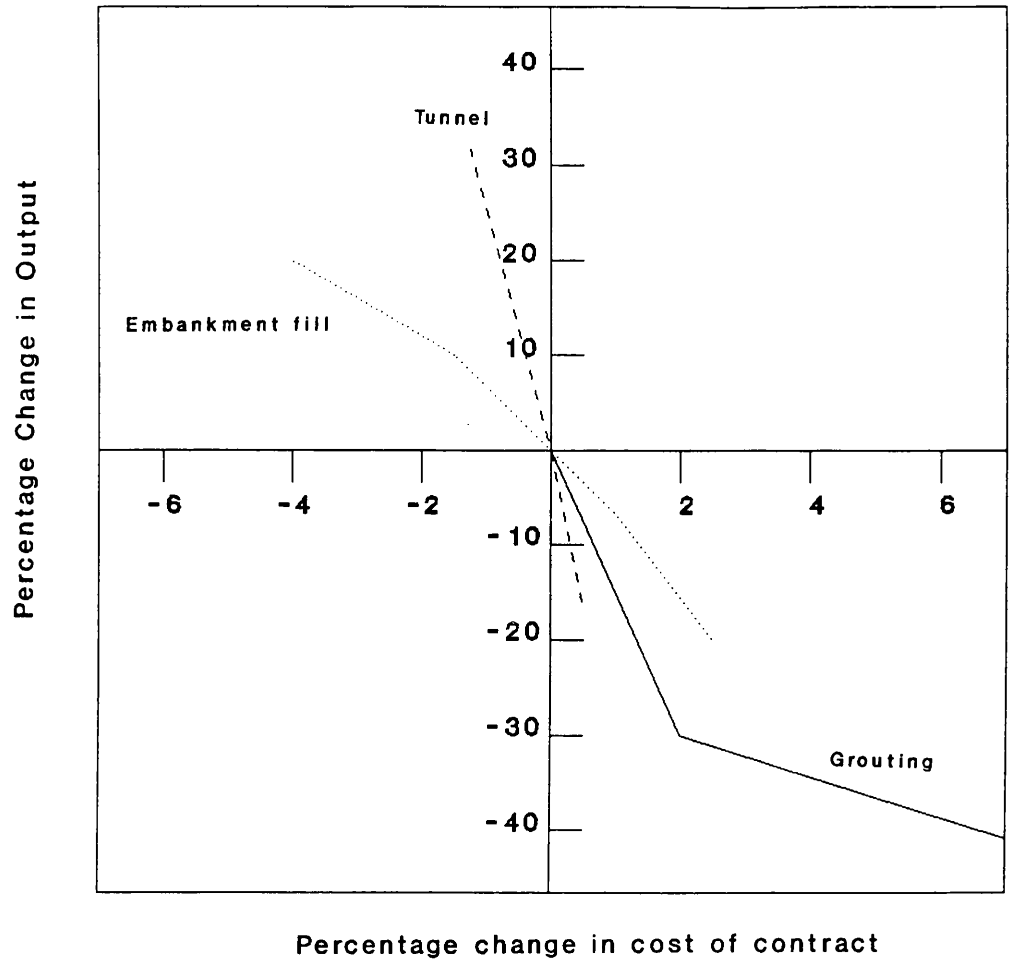


Figure 1.1 Sensitivity Analysis Diagram (Spider Diagram)

One weakness of sensitivity analysis is that the variables are treated individually[9]. This leads to severe limits on the extent to which combinations of variables can be assessed directly from the data. A further weakness of such an analysis is that the sensitivity diagram gives no indication of the anticipated probability of occurrence of any event.

In spite of these weaknesses it has been found that considerable benefits may derive from the use of this simple tool. Such benefits[9] includes:

(a) the powerful impact on management of the realization that there is a range of possible outcomes for a project;

(b) enhanced reality of the decision– making process, despite the increased complexity of the information on which such decisions are based;

(c) the ability to compare the robustness of projects to specific uncertainties;

(d) the relative importance of each variable becoming immediately apparent, thus highlighting those areas which would most benefit from any attempts to reduce or control uncertainty, or those areas which require further development work;

1.3.1.2 *Probability Analysis*

Probability analysis is a more sophisticated form of risk analysis: it overcomes the limitations of sensitivity analysis by specifying a probability distribution of each variable and then considering situations where any (or all) of these variables can change their initial values at any given time[9].

Defining the probability of occurrence of any specific value of a variable may be a difficult problem: not only does every project have many unique features, but political, commercial and fiscal environments change quickly. Nevertheless, it has proved[9] possible to make tentative estimates of probability distributions and ranges.

Essentially, a distribution profile is allocated to the range which has been defined for the variable. A number of profiles are possible, but simple ones are advocated in the absence of statistical data.

The problem of assessing how risks can occur in combination is usually overcome by using a sampling approach, running the analysis a number of times taking random values of each variable (for example the Monte Carlo technique)[9].

The outcome of this full analysis is a range over which the final solutions could lie, and the probability of achieving such solutions is often shown diagrammatically. Examples of such are shown in **Figures** 1.2[9] and 1.3[9].

Depending on the network model being used, and the results required, the analysis can either be time–only, cost–only or, in view of the inherent relationship between both parameters, integrated (time and cost).

Probability analysis has had some notable successes in terms of its predictive ability and consequent assistance to managers in decision making[9].

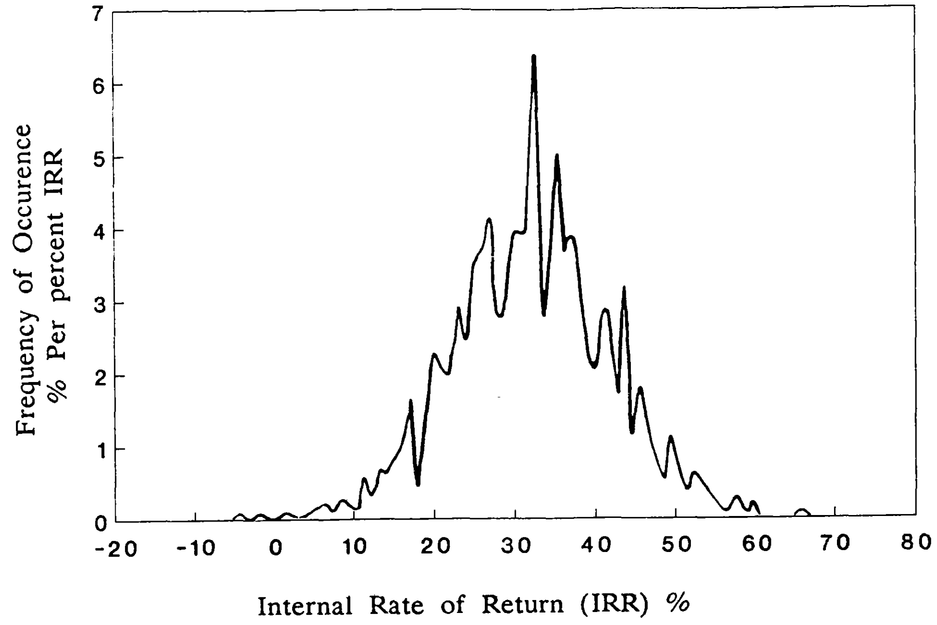


Figure 1.2 Probability Distribution

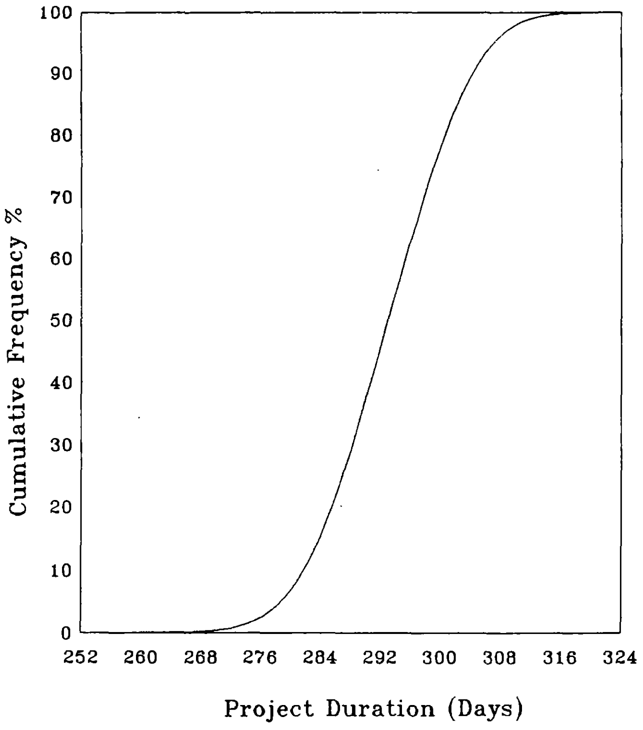


Figure 1.3 Cumulative Probability Diagram

CHAPTER 2

NETWORK ANALYSIS

2.1 Historical Background to Critical Path Analysis (CPA)

Before the advent of CPA, probably the best- known way of trying to plan was by means of a bar or Gantt chart and, although this is extremely useful in many cases, it suffers from inability to show the inter- relationship between the various activities. Thus, it is not possible to deduce from a Gantt chart that, say activity *X* must be completed before activity *I* can be started, or that a delay between activity *Y* and activity *Z* is permissible but not essential. In small projects this is not serious, as the planner can remember the various links between activities, but in large projects such feats of memory are impossible and the Gantt charting technique is then of very limited value[10].

The middle of the 1950s saw an explosion in interest in this problem. In Great Britain[10], the Operational Research Section of the Central Electricity Generation Board investigated the problems concerned with the overhaul of generating plant a – task of considerable complexity which was increasing in importance as new high- performance plant was being brought into service. By 1957, the O.R. Section had devised a technique which consisted essentially of identifying the 'longest irreducible sequence of events' and, using this technique, the time taken to carry out an experimental overhaul at a power station was reduced to 42% of the previous average time for the same work. Continuing to work upon these lines, the overhaul time was subsequently further reduced by 1960 to 32% of the previous average time. The rather clumsy name: 'longest irreducible sequence of events' was soon replaced by the name, 'major sequence' and it was pointed out, for example, that delays in the 'major sequence' would delay completion times, but that difficulties elsewhere need not necessarily involve extensions of total time. This work of the O.R. group was not made public, although comprehensive reports were circulated internally which foreshadowed much later work carried out elsewhere.

At much the same time, work was being undertaken in the U.S.A. and, in early 1958, the U.S. Navy Special Projects Office[9] set up a team to devise a means of dealing with the planning and subsequent control of complex work. This investigation was known as Program Evaluation Research Task, which gave rise (or possibly derived from) the code name PERT. By February 1958, Dr. C.E. Clark, a mathematician in the PERT team, presented the early notions of arrow—diagramming, doubtless deriving from his study of graphics. This early work was rapidly polished and, by July 1958, the first report, *PERT: Summary Report Phase I*[10], was published. By this time, the full title of the work had become *Program Evaluation and Review Technique* and the value of the technique seemed well established. By October 1958, it was decided to apply PERT to the Fleet Ballistic Missiles Programme[10], where it was credited with saving two years in the development of the Polaris missile.

Similar development work was being undertaken elsewhere, for example: in the U.S. Air Force[10] under the code name PEP. Also in 1958, the E.I. du Pont de Nemours Company[10] used a technique called the Critical Path

Method (CPM) to schedule and control a very large project and, during the first complete year's use of CPM, it was credited with saving the company $1.0 million. Subsequent use underlined the basic simplicity and extraordinary usefulness of this method, and by 1959, Dr. Mauchly[10], who had worked with Du Pont project, set up an organization to solve industrial problems using Critical Path Method.

Since 1958, considerable work has been carried out, mainly in the United States of America, in consolidating and improving these techniques.

2.2 *The Network Model*

Two systems of representation of networks are used in project management: the first is known as 'Activity on Arrow', examples being the well-known CPM and PERT programs; the second is known as 'Activity on Node', and PDM is an example of this.

2.2.1 *Activity on Arrow Networks*

Activity on Arrow networks, see **Figure 2.1**, were first used in the late 1950s when the two research teams coincidentally invented the very similar techniques CPM and PERT[6].

PERT and CPM are each based on the simplification of the fundamental network which occurs if the events linked by 'zero time' arrows are combined. If this is done, the arrows joining the nodes can be said to represent activities while the nodes continue to represent events, although now the events are, in fact, combinations of two or more simple events.

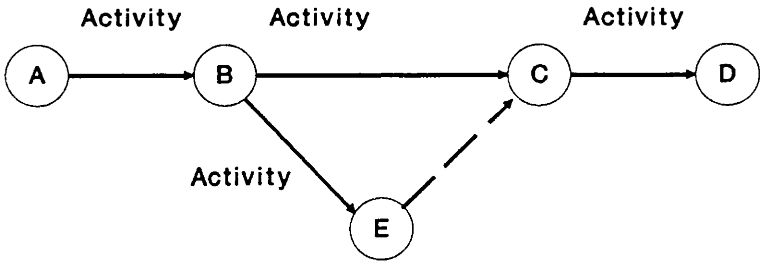


Figure 2.1 Activity on Arrow Network.

The simplification has, however, been bought at the price of some ambiguity for the nodes representing more than one event. If, as is sometimes the case, it is necessary to limit the events at the node, then the 'zero time' arrows of the fundamental network must be re–introduced. Thus to represent the exact situation the 'zero time' arrows (called dummy arrows), must be shown in the network as ilustrated by the 'broken' arrow in **Figure 2.1**. As the complexity of networks increases so does the proportion of dummy arrows in the network, robbing the diagram of some of the simplicity which was won by combining events.

2.2.2 Activity on Node Networks

A further simplification of the fundamental network entails combining the two events which mark the beginning and end of an activity, and the arrow which represent the activity itself. This combination leads to a technique known as the Precedence Diagram Method (PDM) which was introduced in the 1960s[6]. PDM is used in the construction industry to an increasing extent, although the ten-year lead by the Activity on Arrow diagrams has built a considerable 'brand loyalty' for the older techniques. A section from an Activity on Node Network is shown in **Fig.2.2.**

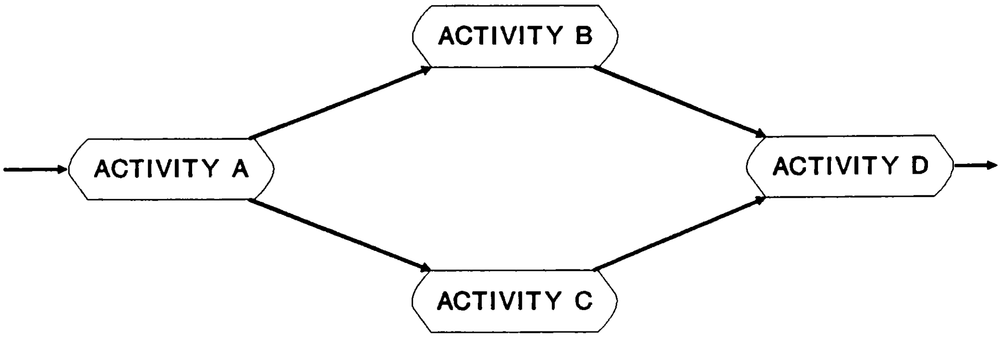


Figure 2.2 Activity on Node Network.

The simplification of the network, brought by the change to precedence diagrams, is not bought at the cost of ambiguity in the diagram, for conventionally the nodes of a precedence network are drawn in the elongated form which makes it possible to visualise the left-hand end of the node as the start of the activity and the right–hand end as the finish. Thus arrows will enter from the left and leave from the right. There is no need for dummy arrows or equivalent in precedence networks, for only one arrow, that presenting the carrying out of the task, can ever occur between the beginning and the end of the task.

2.2.3 A Comparison Between *the Two Networking Systems*

Although CPM and PERT have established a firm hold within the construction industry planning environment, there are strong arguments in favour of a change to the use of PDM. These arguments derive from the lack of ambiguity in PDM and from the relative ease of the use of PDM within computer programs[6].

It has been pointed out above that the CPM simplification of the original network introduces ambiguities. These ambiguities can be overcome by the use of dummy arrows but this destroys some of the simplifications and introduces a potentially dangerous dual status of arrows, some representing both logic and activities while others, the dummies, representing logic alone. Whilst the presence of dummy arrows may not be troublesome in simple networks, in complex networks they will dominate the diagram and will become troublesome when the analysis of the diagram is carried out. The precedence diagram, on the other hand, utilises only one type of arrow, that representing the logical link between activities, the activity arrow having been combined into the node.

The specification of the network for computer usage is much simpler if PDM, rather than CPM, is used. In CPM, the nodes present rather ephemeral events and the activities are defined as linking these. Thus, to code the activities the operator of a computer program must first draw out the network, numbering the nodes, and then use this network during the input stage of the program. With PDM this is not so. In order completely to define the network the user must merely have a list of activities and be in a position to specify which activities are dependent on or precedent to other activities. No prior drawing of network is necessary.

Thus although there are two networking systems available for use in the construction industry, all the advantages, other than the familiarity of the industry with CPM, favour change to the widespread adoption of PDM. All the subsequent discussion and the subroutines developed for this research work will be concentrated, therefore, on the use of the precedence diagram as the most useful model of project.

2.3 *The Analysis of Simple Networks*

It has been argued that a project is a series of activities and the model which has been developed is such that the sequence of activities can be predicted mathematically by forming the activities into a network. The next concern is to develop a mathematical procedure whereby the network can be analysed.

All the activities in the network could take place at any time but, given a particular start date of the project, there is a limit to how early each activity commence. The first part of the analysis of networks is concerned with the identification of the earliest possible time for each activity in the network, this operation being known as the forward pass.

If a forward pass is carried out for the whole network the, earliest possible time for the last activity will be obtained, and, if the aim is to complete the whole project in the shortest possible time, then this early time of the final activity will also be the latest permissible time. The calculation of these latest permissible times forms the second part of the analysis and is called the 'backward pass'.

2.3.1 *The Forward Pass*

In order to calculate the earliest possible time of an activity all the activitiess which must, in accordance with the logic of the project, precede it must be considered and allowance must be made for the durations which are assigned to any of the network arrows. In the case of the precedence network, where all the arrows have zero duration, the start time of an activity will be set to the latest of the finish times of the preceding activities.

As this process continually looks backward to the precedents, it is most efficient to carry out the calculations systematically, working from the start activity, which has no precedents, to the end activity which has every event in the network as its precedent. In this way the calculation is progressive and cumulative: an activity times being determined only when all its precedents have been themselves assigned an early time.

By convention, the upper part of the precedence network node is used to show the early dates: the upper left showing the earliest start of an activity the upper right showing the earliest finish. Starting from the first activity in the network, setting each start time to the latest precedent finish time and each finish time to the start time plus the duration and the values are the earliest possible times for the various activities i.e. for, the Activity on Arrow networks, the earliest possible times (EET) by which each event can be reached is as follows:

EET(succeeding event) = EET(preceding event) + Activity Duration

Where more than one activity leads into an event, the EET is taken to be the <u>largest</u> value of event time computed from any of the routes leading into that event.

For the Activity on Node model, the ealiest start time (ES) of an activity is the <u>highest</u> earliest finish times (EF) of preceding activities and

EF = ES + Activity Duration

2.3.2 *The Backward Pass*

If the end activity of the project is to take place at the earliest possible time, then the times of all other activities in the network which affect it have a time which they must not exceed. To calculate this latest permissible time a process similar to that developed for the forward pass is required. In this case, however, the analysis is concerned not with the activities between the start of the project and the activity being considered, but with those between the activity and the end of the project.

Once again the calculation proceeds in an orderly fashion, but in this case the progression is backwards from the finish activity which have no dependent activities to start, and hence the name 'backward pass'. For, Activity on Arrow networks, the calculation of latest event times **(LET)** is carried out as follows:

LET(preceding event) = LET(succeeding event) − Activity Duration

where more than one activity leads from an event, the **(LET)** is taken to be the smallest value of event time computed using any of the routes leading back into that event.

For the Activity on Node model, the latest finish time of an activity **(LF)** is the lowest latest start time **(LS)** of their succeeding activities and

LS = LF − Activity Duration

2.4 *Criticality and Float*

The forward and backward pass through the network produce for each activity a range of times at which the activity can take place without jeopardising the completion date of the project. The limits of this range are the earliest possible activity times, produced by the forward pass procedure, and the latest permissible time, the product of the backward pass.

In some cases these two times will coincide, and the earliest possible time will also be the latest permissible time, and then any slip will delay the completion of the project. These activities are known as the critical activities, and the path through the network containing them is called the 'critical path'.

Activities away from the critical path have a range of possible times, and this range is known as float. Thus float, the difference between the early and the late times of an activity, is a measure of the tolerance within the network, and comparisons of the float of activities can help the manager of a project to assess priority. In complex projects where, for example, expensive resources are to be shared, the ability to delay the start of one activity, or through the sharing of a common resource to extend its duration, can be of major importance to the manager. Criticality and float make such decision possible.

2.5 Improvements to Realism

The validity of the analysis can be increased by making the model of the project on which the planner is working as realistic as possible. The aim of the developer of all interactive design programs should be to reduce to a minimum the mental gymnastics which the designer has to perform as he takes the presented results and assimilates them prior to deciding his next action[6].

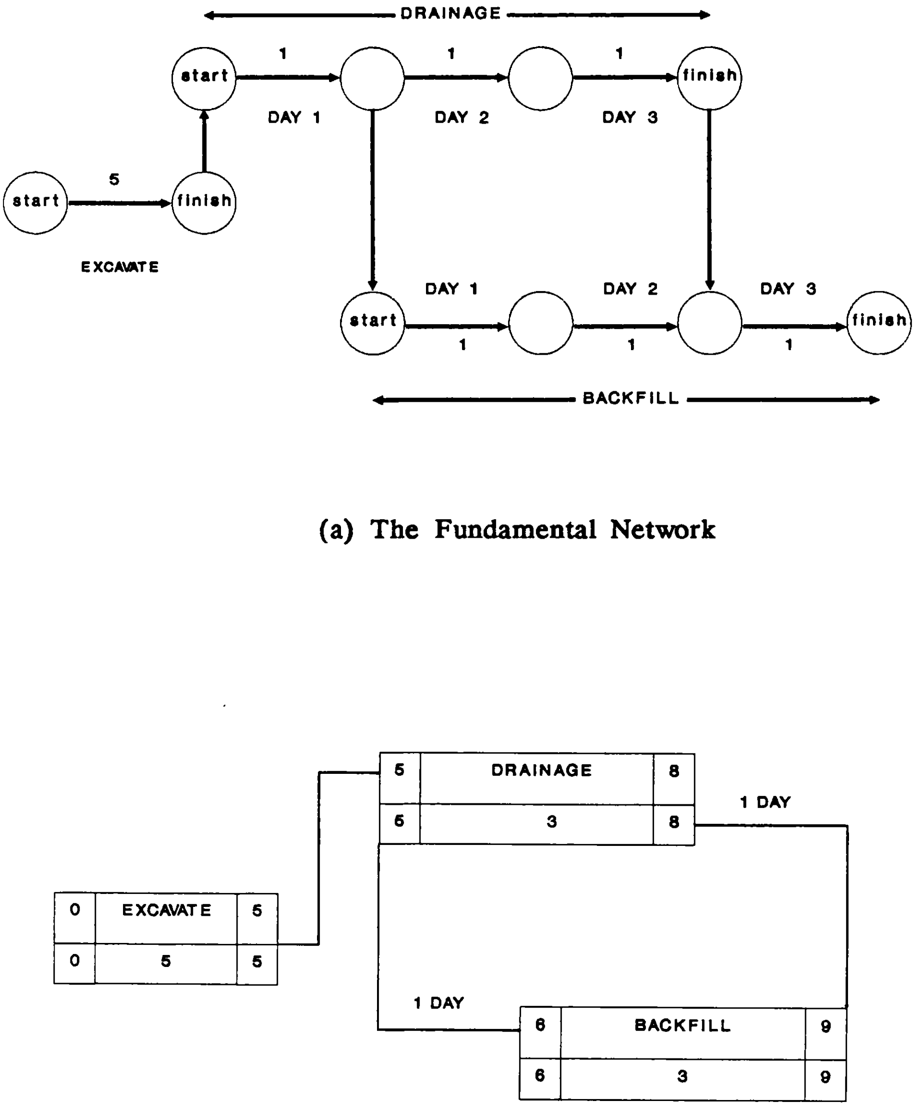
Five techniques will be described, most of them being possible in the non– computer environment when used individually. Their use in combination would face the manual analyst with a complexity such that, at the least, the analysis would lose the elegance which makes hand calculation of network pleasant and that, in the worst case, would force the planner to make mistakes.

2.5.1 Complex Links

One of the advantages of the precedence diagram method is that the specification of links, which are more complex than the direct link, is possible. These complex links, which, together with their equivalents in the fundamental network model, are shown in **Figure 2.3**[6], enable the planner to describe relationships which can only otherwise be described by the splitting of activities, an exercise which would result a significant increase in the complexity of the network. The complex links can be 'leads', which specify that the start of an activity is in some way dependent on the start of its precceeding activity, or 'lags', which specify a relation between the finish of an activity and the finish of its preceding activity.

Complex links allow the planner to use larger activities than would be possible otherwise, and so reduce the complexity of networks both for the analyst and for the user of the results. Their use should be limited, however, for it is tempting for the planner to form 'omnibus' activities which contain a large number of sub–activities and which, because they are amorphous, make detail control impossible[6].

Experience shows that, in a typical contruction network, 10 to 15% of the links in a network may be complex. In some cases, however, particularly in the building industry, wide use is made of very large activities and up to 80% of the links may be complex (with corresponding control difficulties)[6]. The incorporation of complex links into the analysis is not difficult. Referring to **Figure 2.3** it can be seen that, in the forward pass, the start of activity 'BACKFILL' is tied not to the completion of the previous activity but to the partial completion thereof.



(a) The Fundamental Network

(b) The Precedence Network.

Figure 2.3 Complex Links

Thus 'BACKFILL' can start after one day's work has been done on 'DRAINAGE', and thus its start is tied to the start of 'DRAINAGE'. Similarly, the 'lag' link tying the start of the last day's work on 'BACKFILL' to the completion of drainage. In each case the usual forward and backward progression through the network is used as the basis of calculation.

Two points should be noted, the first of which is that, If the lag link in **Figure 2.3** were not present there would be no tie between the finish dates of the activities and the rather strange situation would occur in which the earliest possible completion of the 'DRAINAGE' was day 8, but the latest permissible date was day 9, while the start of the activity remained critical. Consideration of the fundamental network explains this, for the float occurs within the second or third day's work on the 'DRAINAGE' activity leaving the first day's work as critical. Secondly, the delay between the start of the 'DRAINAGE' and the start of the 'BACKFILL' is not absolute. If the first day's drainage work is interrupted, then the delay will be extended.

Complex links are one of the strengths of the precedence network method in that they enable the planner to represent the network in the way that he pictures it (that, for example, the 'BACKFILL' starts one day after the start of the pipelaying) rather than forcing him to make rather artificial divisions in a continuous operation. Whilst they should be available as an option in all PDM programs, their use should be carefully monitored if difficulties in the control phase of the project are to be avoided.

2.5.2 Project Closedown

It is very unusual for projects to continue without a break other than the usual stoppages for weekends; even the busiest project has to accommodate national holidays. When these stoppages occur they obviously lengthen the duration of the current activities.

One way of dealing with this is to construct plans in terms of project days, the project day number being equal to the number of working days since the beginning of the project. This approach is widely used in the non– computer environment but has disadvantages. The major disadvantage is that the construction of a separate project calendar divorces the project activities from the outside world and makes interrelation between them more difficult for the planner and the manager[6]. A second problem caused by this approach is that such an analysis method can not handle those activities which are unaffected by close own; examples of these include the curing of concrete and the consolidation of an embankment under surcharge, i.e. those which continue whether there are men on site or not. An alternative to the project calendar approach which avoids these difficulties, but at a cost of extending the analysis time, is to consider the effect of closedown within the analysis itself. As the duration of each activity is being incorporated into the network (i.e. the calculation of finish times in the forward pass and start times in the backward pass) the location of the activity in time is checked to see if it coincides with a close– down period. If it does then the duration is extended by a time equal to the length of the close– down.

2.5.3 Window Times

A very frequent constraint on the planner of the projects is the requirement that an activity should fall within the specified period. Examples of this are the 'possession times' provided by the railways for the carrying out of work near their tracks and the 'weather windows' which have dictated the launch and positioning of off-shore structures for the oil industry. This type of constraint is easily built into the basic analysis method for the forward and backward pass merely by setting the event times prior to analysis for the forward and backward passes respectively to the start and finish of the time window for the activity concerned and then only altering the event times if they are to be increased (in the forward pass) or reduced (in the backward pass).

Experience shows that very rarely are project plans free from constraints of this type[6]. The absence of the facility automatically to include them in the program seriously inhibits the usefulness of planning programs, but their inclusion is both easy and cheap.

2.5.4 Calendar Dates

The tying of project events to calendar dates has already been shown to be important. The major advantage of the provision of calendar dates is that it removes the need for the planner to carry two calendars in his head, making the use of output much more easy. It is particularly valuable where the project is tied to external events as when, for example, window times are being used. The provision of calendar dates is one of the essential features of a useful planning program.

Although the use of calendar dates is of great help to the planner, this help is necessary only at the output stage. There is, therefore, merit in delaying consideration of calendar dates until output data are being presented and in using a project calendar within the analysis.

2.5.5 Seasonal Variations

The construction industry in the U.K. is particularly sensitive to adverse weather conditions, and this sensitivity must be reflected by the planner as he positions activities in time and estimates activity durations using forecasts of production. Where this adjustment is made, it is made after the analysis of the network by extending those activities which have been shown to occur during the winter months and reducing the times of those which fall in the summer. Many Contractors have curves of productivity which act as guides for the planner and the site manager; one such is shown in **Figure 2.4**[6]. These curves correct not only for the effects of bad weather but also for the shorter working hours of the northern winter and the effect on the morale of the labour force due to working in unpleasant conditions.

A computer technique which allows for the seasonal variations to production is obviously of benefit, for all activities can then be estimated as if the work were to be carried out at those times of the year (April and September) when an average production can be assumed. Like the provision of window times, this facility is very easily built into the analysis method.

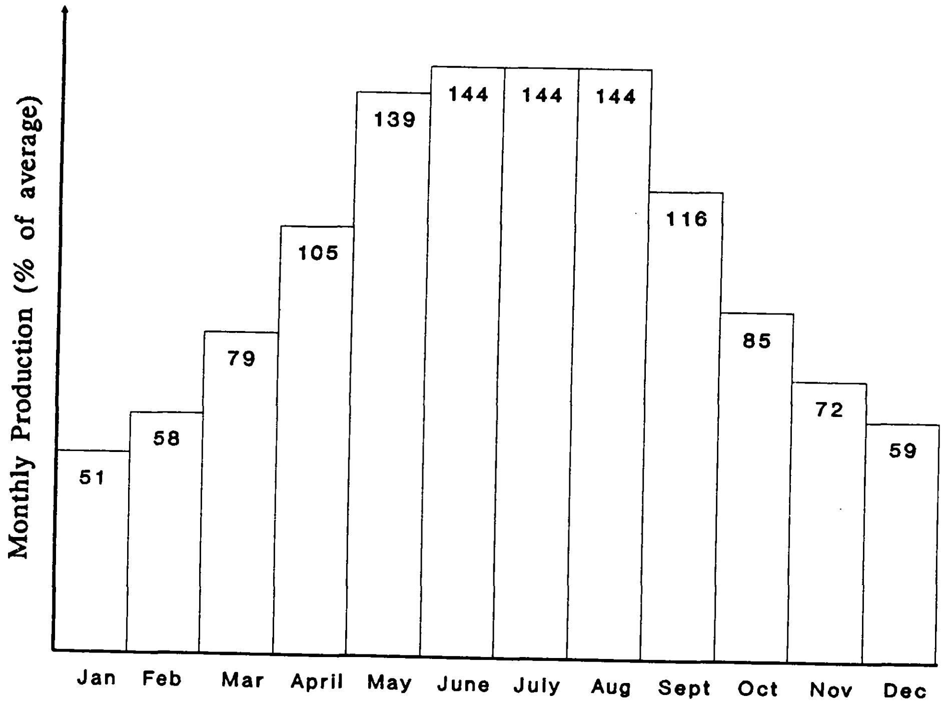


Figure 2.4 The Seasonal Variation in Construction Productivity in the UK.

The ability to adjust times automatically is advantageous not only to the planner of projects but also to those responsible for the control phase and, perhaps most lucratively, those concerned with the costing of disruption and variations. This technique cannot of course be used in the absence of a routine to calculate calendar dates. It consist simply of the identification of the month in which the activity is programmed to take place and the application of a suitable factor to the stored duration. It is a simple but remarkably effective addition to the traditional analysis, but one which cannot be made without the help of the computer.

CHAPTER 3

CASH FLOW FORECASTING

3.1 The Need for Cash Flow Forecasting

Each year the construction industry usually experiences a proportionally greater number of bankruptcies than do other industries[4]. One of the final causes of bankruptcy is a shortage of cash resources combined with a failure to convince financial institutions and other creditors that this shortage is only temporary. This sort of situation comes about for a variety of reasons, the most usual being a tendency to concentrate more on the relative magnitudes of cash inflows and outflows, i.e. profit, than on the relative timings thereof.

A cash flow is the transfer of money into or out of the company. The timing of cash flow is important. There is always, whatever the industry, a time lag between incurring the production or manufacturing costs of an article and receiving the cash payment for the finished product.

In the context of the construction industry, even though the contractor is paid at regular intervals throughout the project and not in a lump sum at the end, the contractor is required to carry the cost for somewhere between one and two months owing to the fact that work completed during any given month is not actually paid for until the end of the following month.

It is worth mentioning that, even if the contractor has credit facilities with suppliers of plant and materials, there are no credit facilities for paying labour, so that, the contractor will always have to carry a proportion of the cost until such time as he's paid.

It is this time lag between costs and receipts which is responsible for the most common economic feature of construction project. Namely that, particularly with a tight profit margin, most projects don't break even until they are virtually completed.

Together with the risks associated with construction, this is probably the reason why the construction industry experiences more bankruptcies in any given year than the majority of other industries.

It can be conclude that, irrespective of the size of the company, the need to forecast cash requirements is vital so that provisions can be made for these difficult times before they occur.

Contractors who undertake cash flow forecasting do so at two levels: Project level—— which is carried out at the tendering stage, for single projects—— and company level, carried out at regular intervals, which involves the aggregation of the cash flows for all active projects[4].

These two types of forecasts require different treatments. In the foremer case, the estimator has all the project details at the estimating stage and, because the forecast applies only to one project, the estimator can produce a carefully calculated forecast based on these details by allocating bill items to 'activities' on the pre– tender 'bar chart' or 'network'. This creates a direct link between the estimators build– up for each item and the pre– tender construction program and allows the production of value versus time and cost versus time curves from which 'cash in' and 'cash out' can be calculated. This calculation is too detailed to be repeated for every project, every quarter or every month, so contractors often devise short cuts when undertaking the company or divisional cash flow forecasts, such shortcuts allowing cash flow forecasts to be made more regularly than just the one detailed forecast at the estimating stage.

3.2 The Requirement for a Forecasting System

Cash flow forecasting is strongly advisable and, for it to be meaningful, must be done regularly. It therefore follows that, for forecasting to be done regularly, the method must be simple and yet accurate enough for the purpose. It is essential, therefore, to reduce the data required for cash flow forecasting to the minimum possible compatible with reliable forecasts and to streamline the necessary calculations[4].

Cash flow forecasting is carried out for the following reasons:

(a) To ensure that sufficient cash is available to meet demands.

(b) To provide a reliable indicator to financing institutions, that, advances made can be repaid according to an agreed programme.

(c) To determine the cost of financing .

(d) To ensure that cash resources are fully utilised to the benefit of the company.

3.2.1 The Data Needed for Cash Flow Calculations

In construction companies, the most appropriate approach is to calculate cash flows on a project basis (which is the immediate concern of this research project) and to aggregate the cash flows from all projects and head office to form the overall company cash flow. This could be structured into divisions or areas for larger companies. The data required for a project are:

1. The magnitude and timing of incurred costs (cash outflow).

i.e. the costs incurred by the contractor in carrying out the project, being based on the estimator's cost build up and the pre-tender programme.

2. The magnitude and timing of receipts (cash inflow).

i.e. the payments by the client for the work completed, being based on incurred costs, plus the appropriate profit margin, and the pre-tender programme.

The compilation of a project cash flow is illustrated by considering a small construction project, the pre-tender programme and estimated financial breakdown of which are shown in **Figure 3.1**[17] and **Tables 3.1**[17] and **3.2**[17].

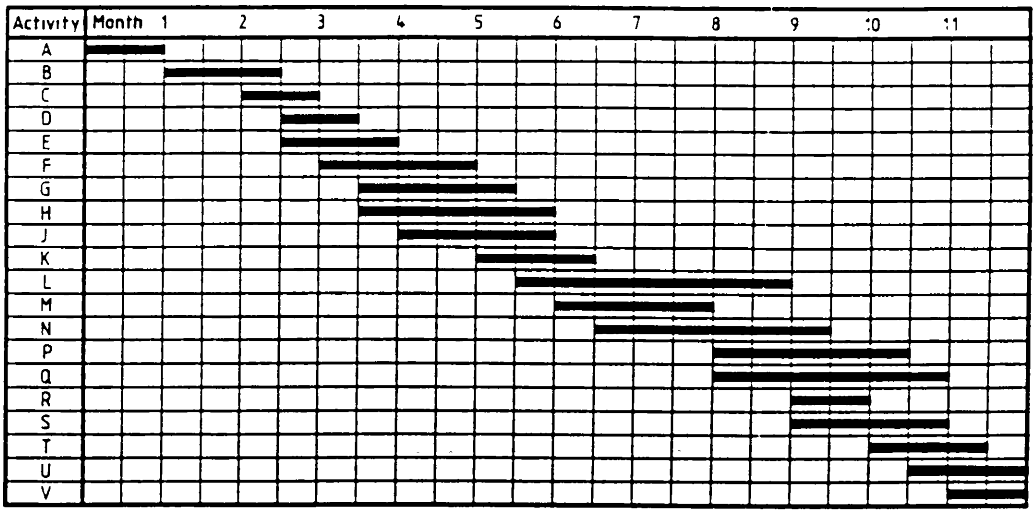


Figure 3.1 Project Pre– tender Programme.

ACTIVITY	LABOUR	MATERIALS	PLANT	SUB-CONT.	TOTAL
A	17,000	9,450	11,250	-	37,700
B	12,150	12,780	47,550	-	72,480
C	3,240	1,800	2,250	-	7,290
D	13,580	7,560	9,100	-	30,240
E	11,340	9,450	17,010	-	37,800
F	14,680	8,200	9,100	-	31,980
G	18,720	11,680	16,400	-	46,800
H	20,450	11,350	13,600	-	45,400
J	5,500	-	9,460	43,200	58,160
K	10,980	6,120	7,290	-	24,390
L	28,420	18,970	47,460	-	94,850
M	42,100	23,400	28,000	-	93,500
N	12,600	6,360	18,960	-	37,920
P	-	10,500	-	48,100	58,600
Q	19,530	10,890	13,050	-	43,470
R	5,670	3,150	3,780	-	12,600
S	4,760	2,880	3,320	-	10,960
T	4,950	2,610	3,150	-	10,710
U	5,400	12,900	11,400	-	29,700
V	3,870	2,070	2,610	-	8,550
TOTALS	254,940	172,120	274,740	91,300	793,100

Table 3.1 Project Direct Cost Breakdown.

Labour Total	£ 254,940
Materials Total	£ 172,120
Plant Total	£ 274,740
Subcontractors Total	£ 91,300
Direct Costs	£ 793,100
General Overheads	£ 81,800
Construction Costs	£ 874,100
Profit (@ 5% margin)	£ 43,705
Tender Sum	£ 917,805

Table 3.2 Project summary.

Based on the pre–tender programme and the financial breakdown, the cumulative cost and revenue pattern is shown in **Table 3.3**[17] and **Figure 3.2**[17].

ACTIVITY	MONTH											
	1	2	3	4	5	6	7	8	9	10	11	12
A	37,700	-	-	-	-	-	-	-	-	-	-	-
B	-	48,320	24,160	-	-	-	-	-	-	-	-	-
C	-	-	7,290	-	-	-	-	-	-	-	-	-
D	-	-	15,120	15,120	-	-	-	-	-	-	-	-
E	-	-	12,600	25,200	-	-	-	-	-	-	-	-
F	-	-	-	15,990	15,990	-	-	-	-	-	-	-
G	-	-	-	11,700	23,400	11,700	-	-	-	-	-	-
H	-	-	-	9,080	18,160	18,160	-	-	-	-	-	-
J	-	-	-	-	29,080	29,080	-	-	-	-	-	-
K	-	-	-	-	-	16,260	8,130	-	-	-	-	-
L	-	-	-	-	-	13,550	27,100	27,100	27,100	-	-	-
M	-	-	-	-	-	-	46,750	46,750	-	-	-	-
N	-	-	-	-	-	-	6,320	12,640	12,640	6,320	-	-
P	-	-	-	-	-	-	-	-	23,440	23,440	11,720	-
Q	-	-	-	-	-	-	-	-	14,490	14,490	14,490	-
R	-	-	-	-	-	-	-	-	-	12,600	-	-
S	-	-	-	-	-	-	-	-	-	5,480	5,480	-
T	-	-	-	-	-	-	-	-	-	-	7,140	3,570
U	-	-	-	-	-	-	-	-	-	-	9,900	19,800
V	-	-	-	-	-	-	-	-	-	-	-	8,550
OVERHEADS	6,750	6,750	6,750	6,750	6,750	6,750	6,750	6,750	6,750	6,750	6,750	6,750
CUMULATIVE COST	44,450	99,520	165,440	249,280	342,660	438,160	533,210	626,450	710,870	779,950	835,430	874,100
CUMULATIVE REVENUE	46,673	104,496	173,712	261,744	359,793	460,068	559,871	657,773	746,414	818,949	877,202	917,805

Table 3.3 Cost/Revenue Summary.

The cost/revenue data has been compiled subject to the following assumptions:

1. The costs associated with each activity are evenly spread over the duration of the activity.

2. Overheads are evenly spread over the project duration.

3. Each activity carries the same profit margin.

4. Revenue is earned immediately each item of work is completed

The cost and revenue curves in **Figure** 3.2[17] represent the commitment of the contractor, to pay the necessary costs, and the client, to pay for the completed items of work, they do not represent the actual monetary transactions required to produce the project cash flow.

The project cash flow as given in **Table** 3.4[17] is compiled in accordance with the following assumptions in regard of receipts and incurred costs:

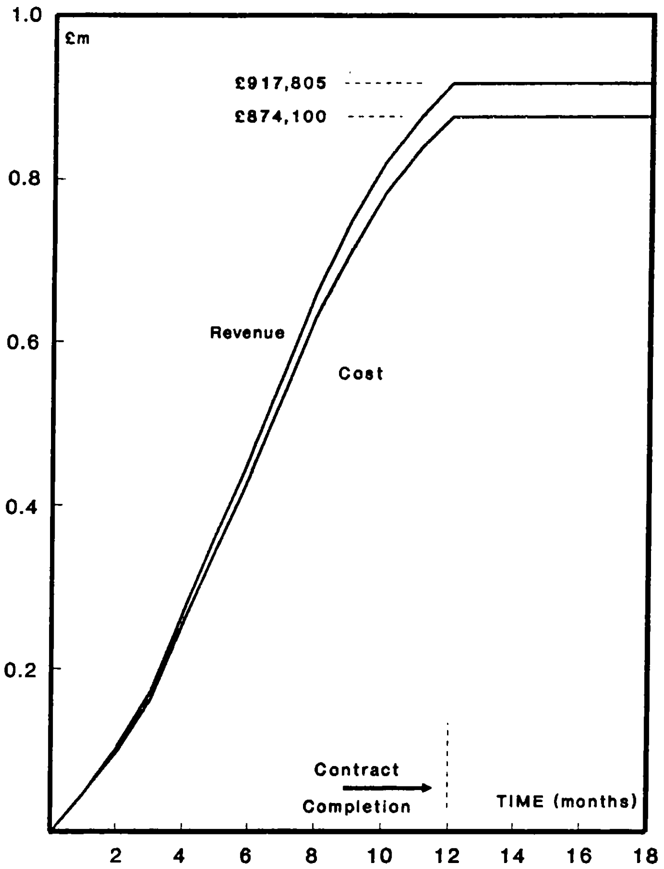


Figure 3.2 Project Cost– Revenue Pattern.

ACTIVITY		MONTH													
		1	2	3	4	5	6	7	8	9	10	11	12	13	18
A	L	17,000	-	-	-	-	-	-	-	-	-	-	-	-	-
	M	-	9,450	-	-	-	-	-	-	-	-	-	-	-	-
	P	-	11,250	-	-	-	-	-	-	-	-	-	-	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	L	-	8,100	4,050	4,260	-	-	-	-	-	-	-	-	-	-
	M	-	-	8,520	16,650	-	-	-	-	-	-	-	-	-	-
	P	-	-	31,700	-	-	-	-	-	-	-	-	-	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
C	L	-	-	3,240	-	-	-	-	-	-	-	-	-	-	-
	M	-	-	-	1,800	-	-	-	-	-	-	-	-	-	-
	P	-	-	-	2,250	-	-	-	-	-	-	-	-	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
D	L	-	-	6,790	6,790	3,780	-	-	-	-	-	-	-	-	-
	M	-	-	-	3,780	4,550	-	-	-	-	-	-	-	-	-
	P	-	-	-	4,550	-	-	-	-	-	-	-	-	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
E	L	-	-	3,780	7,560	-	-	-	-	-	-	-	-	-	-
	M	-	-	-	3,150	6,300	-	-	-	-	-	-	-	-	-
	P	-	-	-	5,670	11,340	-	-	-	-	-	-	-	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
F	L	-	-	-	7,340	7,340	-	-	-	-	-	-	-	-	-
	M	-	-	-	-	4,100	4,100	-	-	-	-	-	-	-	-
	P	-	-	-	-	4,550	4,550	-	-	-	-	-	-	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
G	L	-	-	-	4,680	9,360	4,680	-	-	-	-	-	-	-	-
	M	-	-	-	-	2,920	5,840	2,920	-	-	-	-	-	-	-
	P	-	-	-	-	4,100	8,200	4,100	-	-	-	-	-	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
H	L	-	-	-	4,090	8,180	8,180	-	-	-	-	-	-	-	-
	M	-	-	-	-	2,270	4,540	4,540	-	-	-	-	-	-	-
	P	-	-	-	-	2,270	5,440	5,440	-	-	-	-	-	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
J	L	-	-	-	-	2,750	2,750	-	-	-	-	-	-	-	-
	M	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	P	-	-	-	-	-	4,730	4,730	-	-	-	-	-	-	-
	S	-	-	-	-	-	21,600	21,600	-	-	-	-	-	-	-
K	L	-	-	-	-	-	7,320	3,660	-	-	-	-	-	-	-
	M	-	-	-	-	-	-	4,080	2,040	-	-	-	-	-	-
	P	-	-	-	-	-	-	4,860	2,430	-	-	-	-	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
L	L	-	-	-	-	-	4,060	8,120	8,120	8,120	-	-	-	-	-
	M	-	-	-	-	-	-	2,710	5,420	5,420	5,420	-	-	-	-
	P	-	-	-	-	-	-	6,780	13,560	13,560	13,560	-	-	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
M	L	-	-	-	-	-	-	21,050	21,050	-	-	-	-	-	-
	M	-	-	-	-	-	-	-	11,700	11,700	-	-	-	-	-
	P	-	-	-	-	-	-	-	14,000	14,000	-	-	-	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
N	L	-	-	-	-	-	-	21,100	4,200	4,200	2,100	-	-	-	-
	M	-	-	-	-	-	-	-	1,060	2,120	2,120	1,060	-	-	-
	P	-	-	-	-	-	-	-	3,160	6,320	6,320	3,160	-	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
P	L	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	M	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	P	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	S	-	-	-	-	-	-	-	-	-	23,440	23,440	11,720	-	-
Q	L	-	-	-	-	-	-	-	-	6,510	6,510	6,510	-	-	-
	M	-	-	-	-	-	-	-	-	-	3,630	3,630	3,630	-	-
	P	-	-	-	-	-	-	-	-	-	4,350	4,350	4,350	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R	L	-	-	-	-	-	-	-	-	-	5,670	-	-	-	-
	M	-	-	-	-	-	-	-	-	-	-	3,150	-	-	-
	P	-	-	-	-	-	-	-	-	-	-	3,780	-	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
S	L	-	-	-	-	-	-	-	-	-	2,380	2,380	-	-	-
	M	-	-	-	-	-	-	-	-	-	-	1,440	1,440	-	-
	P	-	-	-	-	-	-	-	-	-	-	1,660	1,660	-	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
T	L	-	-	-	-	-	-	-	-	-	-	3,300	1,650	-	-
	M	-	-	-	-	-	-	-	-	-	-	-	1,740	870	-
	P	-	-	-	-	-	-	-	-	-	-	-	2,100	1,050	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
U	L	-	-	-	-	-	-	-	-	-	-	1,800	3,600	-	-
	M	-	-	-	-	-	-	-	-	-	-	-	4,300	8,600	-
	P	-	-	-	-	-	-	-	-	-	-	-	3,800	7,600	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
V	L	-	-	-	-	-	-	-	-	-	-	-	3,870	-	-
	M	-	-	-	-	-	-	-	-	-	-	-	-	2,070	-
	P	-	-	-	-	-	-	-	-	-	-	-	-	2,610	-
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OVERHEADS		6,750	6,750	6,750	6,750	6,750	6,750	6,750	6,750	6,750	6,750	6,750	6,750	-	-
CUMULATIVE COST		23,750	59,300	124,130	202,650	283,660	376,400	479,840	573,330	652,030	734,280	800,690	851,300	874,100	-
GROSS RECEIPTS		-	46,673	57,824	69,216	88,032	98,049	100,276	99,803	97,902	88,641	72,534	58,254	40,604	-
RETENTION		-	-2,334	-2,891	-3,461	-4,402	-4,902	-5,014	-4,530	-	-	-	-	+13,767	+13,767
CUMULATIVE RECEIPTS		-	44,339	99,272	165,027	248,657	341,804	437,065	532,338	630,240	718,881	791,415	849,669	904,040	917,8074

Table 3.4 Project Cash Flow.

(a) Receipts from the Client

1. Work completed in any given month is certified for payment at the end of that month and paid for at the end of the following month.
2. 5% retention is deducted from each monthly valuation, up to a cumulative maximum of 3% of the tender total.
3. 50% of retention is to be refunded on completion.
4. 50% of retention is to be refunded on expiry of the maintenance period of 5 months.

(b) Incurred Costs

1. Labour paid as required.
2. Payment for plant and materials subject to one month's credit.
3. Subcontractors are paid in monthly 'single payment' increments immediately following receipts of the appropriate certified payment from the client.

3.3 Capital Lock–up

The negative cash flow in the early stages of projects represents 'locked–up' capital which must either be supplied from the contractor's cash reserves or borrowed from financial institutions. If the required capital is borrowed the interest payable should be charged to the project; if the required capital is supplied from the contractors reserves the company is being deprived of its interest—earning capability and should therefore charge the project accordingly. A measure of the interest payable is represented by the area between the cash outflow and cash inflow curves, given the units **CAPTIM** (capital x time). **Figure** 3.3[17]. shows the cash flow diagram for the project together with the calculated negative captim.

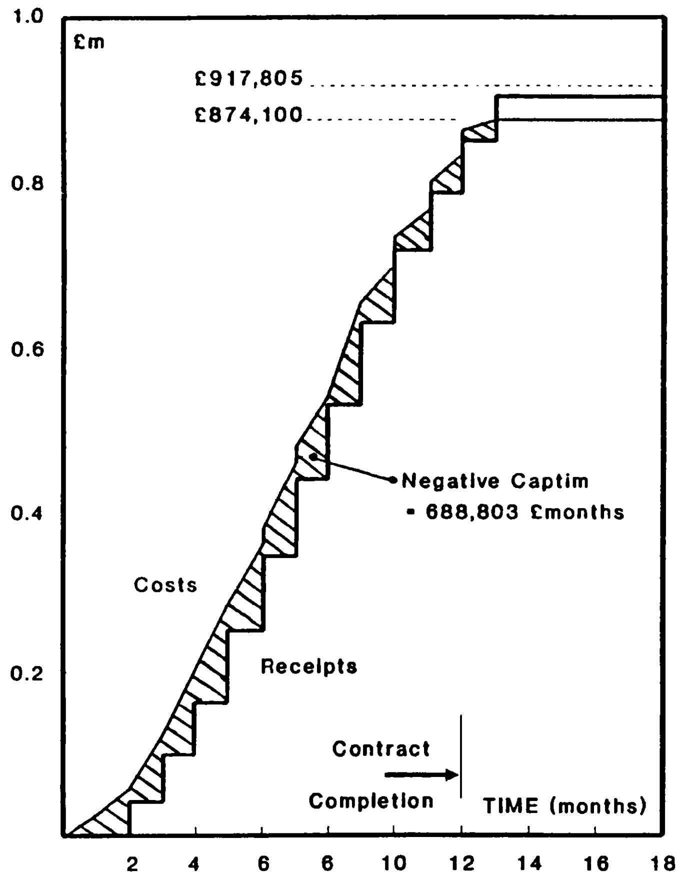


Figure 3.3 Project Cash Flow Diagram.

This compound measure, say 688,803 £ x month, represents the volume of borrowing from the extremes of £688,803 for one month to £1 for 688,803 months.

A simple interest calculation is all that is required to convert captim into an interest charge, if it is assumed that the required finance is borrowed at an annual interest rate of 13½%, then:

$$\text{Chargeable interest} = 688{,}803 \times \left[(1.135)^{1/12} - 1\right] = £7{,}307$$

where $\left[(1.135)^{1/12} - 1\right]$ is the monthly equivalent of 13½% p.a.

If the required margin is to be maintained, the interest charges may be passed on to the client in one of the following forms:

(a) An additional lump in the BoQ Adjustment item.

(b) A general increase of item rates.

(c) A combination of both.

While if the tender sum is remain unchanged by absorbing the interest charges as additional overhead costs, the margin is reduced thus:

$$\text{Total cost} = £874{,}100 + £7307 = £881{,}407$$

$$\text{Effective Margin} = \frac{\text{Tender Sum - Total Cost}}{\text{Total Cost}}$$

$$= \frac{917{,}807 - 881{,}407}{881{,}407} \times 100\%$$

$$= 4.13\%$$

The maximum required cash resource can be determined directly from the project cash flow, which is equal to £136,265 at the end of month eight.

Interest calculation are usually based on the negative captim only, assuming that the cash released by the project does not earn interest, or the interest earned by the positive captim is subtracted from the interest paid on the negative captim. It is likely that the interest paying rate will be different and probably greater than the interest earning rate. The use of this captim measure enables the effect of interest charges to be evaluated.

3.3.1 Factors Affecting Capital Lock–up

The factors that affect the capital lock– up for an individual project or contract are as follows.

3.3.1.1 Margin

The margin, whether profit margin or contribution (profit plus head office overheads) margin, is amongst the most important because it determines the excess over costs and is this excess that control the capital lock– up. Quite simply, the larger the margin, the less capital that is locked up: conversely, the smaller the margin, the more capital that is locked up in the contract.

The margin that should be used in calculating contract cash flows should be the 'effective' margin that is being achieved at the time of executing the contract. This effective margin is neither the margin included at the tender stage nor the margin achieved at the end of the contract when all claims are settled. For example if the tender panel included a margin of 9% and due to variations and other client interference the cost rose by 4%, this would reduce the 'effective' margin to only 5%. Even if the contractor recovered another 6% in claims, making his overall achieved margin at the end of the contract 10%, the 'effective' margin at the time of executing the contract and thus determining the cash flow, would still be only 5%.

Most contractors use the tender margin when calculating contract cash flows and this leads to an optimistic forecast. Although the margin is an important factor in determining a contract's cash flow the tender margin is usually chosen for market reasons rather than cash flow reasons. The effects on the project under consideration are shown in **Figure 3.4**[17].

3.3.1.2 Retention

In the U.K. the system of retention simply reduces the effective margin during the execution of the contract and the effect of retentions can be included in the calculations as shown. In times of very low margins the retention can reduce the effective margin to zero or less. As retentions are fairly standard in public sector contracts there is little scope for negotiation to reduce retention and improve cash flows.

3.3.1.3 Claims

As explained above under the section of margins, claims can return a contract to its original intended level of profit. However, as the settlement of claims is normally subject to some delay the actual settlement does not improve a contract's cash flow and the circumstances given rise to a genuine claims are likely to worsen the contract's cash flow. The settlement of claims is, of course, important to the company's cash flow and therefore it is important that claims are settled as quickly as possible.

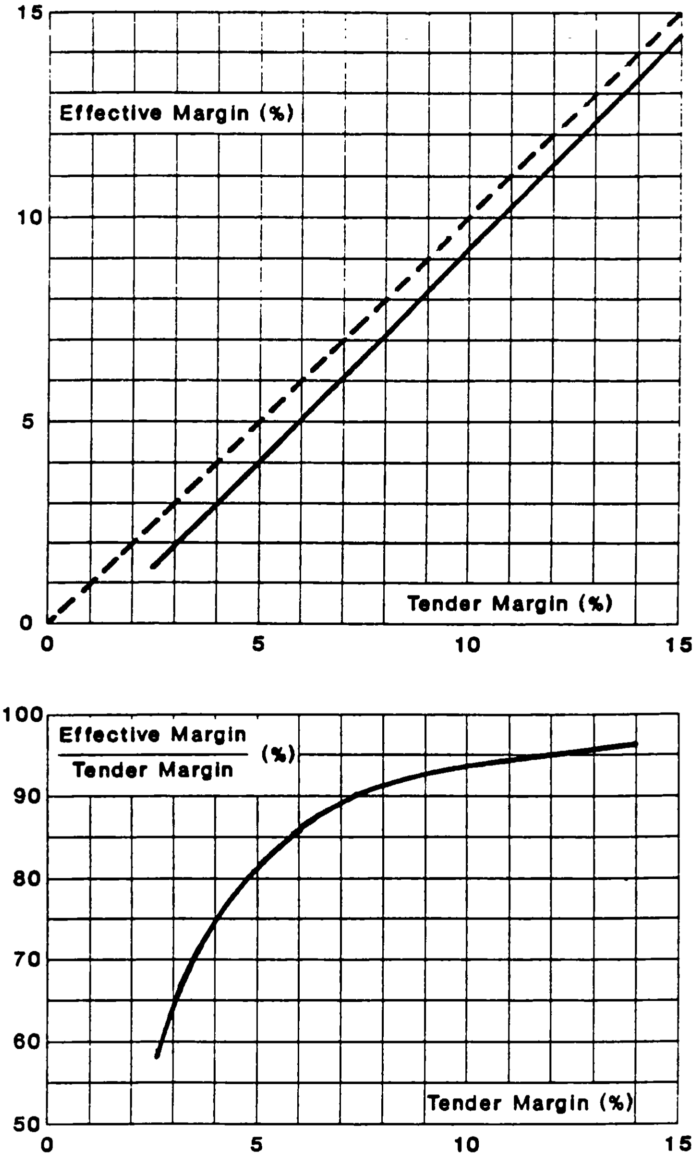


Figure 3.4 The Effect of Margin on Capital Lock—up.

3.3.1.4 Front– end Rate Loading

Front end rate loading is the device whereby the earlier items in the bill carry a higher margin than the later items. This has the effect of improving the effective margin in the early stages of the contract while keeping the overall margin at a competitive level. It is in these early stages that capital lock up is at its worst. The degree to which front end rate loading can be done depends on the client's awareness.

The effects on project under consideration is shown in **Table 3.5**[17] and **Figure 3.5**[17] in which the overall tender margin of £43,705 **(Table 3.2)** is carried 'pro– rata' by activities A to J **(Figure 3.1)** which are completed in the first six months of the project.

	MONTH													
	1	2	3	4	5	6	7	8	9	10	11	12	13	18
CUMULATIVE COSTS	23,750	59,300	124,130	202,850	283,860	378,400	479,840	573,330	652,030	734,280	800,890	851,300	874,100	–
VALUE	–	48,929	60,811	72,950	93,000	103,673	102,603	96,060	93,240	84,420	69,080	56,480	38,870	–
RETENTION	–	-2,446	-3,041	-3,848	-4,660	-5,184	-5,129	-3,436	–	–	–	–	+13,767	+13,767
CUMULATIVE RECEIPTS	–	46,483	104,253	173,656	261,906	360,393	457,767	549,381	842,621	727,041	798,121	851,601	904,036	917,806

Table 3.5 Cash Flow for Front end Loaded Project.

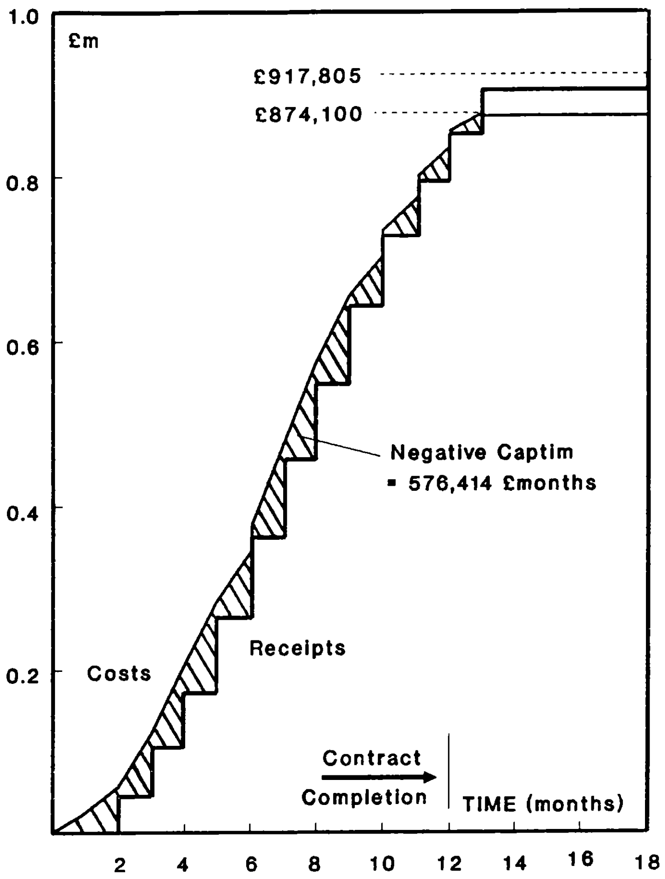


Figure 3.5 The Effect of Front End Loading.

3.3.1.5 Overmeasurement

Overmeasurement is the device whereby the amount of work certified in the early months of a contract is greater than the amount of work done. This is compensated for in later measurements. Thus, overmeasurement has the same effect as front end rate loading; it improves the 'cash in' in the early stages and reduces the capital lock– up.

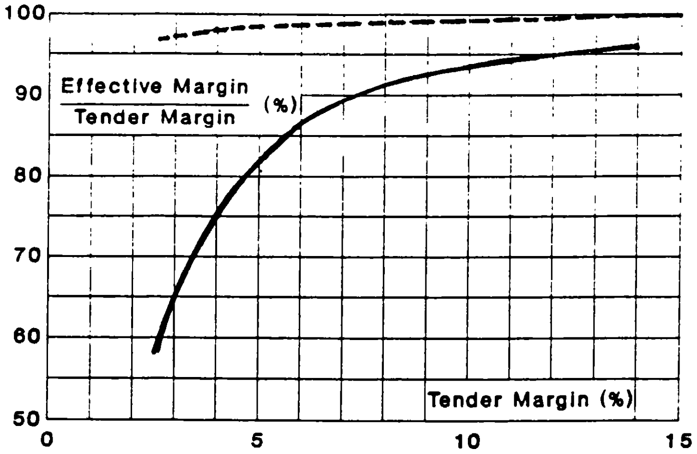
3.3.1.6 Back—end Rate Loading and Undermeasurement

Back end rate loading is the opposite to front end rate loading. Back end rate loading is the device whereby the later items in the bill carry a higher margin than the earlier items. Undermeasurement is the situation whereby the amount of work certified in the early months of a contract is less than the amount actually done. Both these devices have the effect of increasing the capital lock—up and, in most circumstances, these situations are not sought by contractors. However in index—linked price adjustment contracts these devices enhance the monies recovered from price fluctuations in times of inflation. If inflation is high in comparison to the cost of money the contractor may get a better return by funding a larger capital lock—up and gaining on price fluctuations rather than minimising the capital lock—up in the more traditional way. This circumstances existed in the United Kingdom in 1975 and 1976[4].

Thus, only when inflation is very high, which is generally difficult to predict, and borrowing limitations are not critical, can the approach of back end rate loading be countenanced at the tender stage. However, monthly over or undermeasurement gives a contractor a flexibility of policy over the duration of a contract such that changes can be made quickly in response to the predicted movements in the inflation indices and money markets. The effectiveness of this policy also depends on the client's awareness.

3.3.1.7 Delay in Receiving Payment from Client

The time between interim measurement, issuing the certificate and receiving payment is an important variable in the calculation of cash flows. Although monies out goes to many destinations, e.g. labour, plant hirers, materials suppliers and subcontractors, the monies in comes only from one source (the client). Thus any increase in the delay in receiving this money delays all the income for the contract with a resulting increase in the capital lock–up. The time allowed for this payment is specified in the contract and normally interest is charged if payment is late. This may act as incentive to the client but if he is slow in paying it is the contractor who has to find the cash. The effect of delays on receiving payments on the project under consideration is shown in **Figure 3.6**[17].



--- Payment immediately following certification

—— Payment one month after certification

Figure 3.6 Effect of Client Payment Delay.

3.3.1.8 Delay in Paying Labour, Plant Hirers, Materials Suppliers and Subcontractors

The time interval between receiving goods or services and paying for these is the credit the contractor receives from his suppliers. A one week delay is normal in paying labour and any thing between three to six weeks is normal in paying plant hirers and material suppliers. Any increase in these would reduce the capital required to fund a contract. However these items evolve as part of the normal commercial trading arrangements and any increase in these times may undermine commercial confidence in the company. Hence, these factors are not usually seen as suitable for controlling capital lock– up in a contract.

3.3.1.9 Inflation

The effects of inflation on the project cash flow under consideration is shown in Table 3.6[17] and Figure 3.7[17], which have been compiled using the following data:

1. Tender Margin: 5% at current prices.

2. Inflation Rate: 8% p.a.

3. Interest Rate: 13½% p.a.

4. Increased costs reimbursed using the CPF clause in the I.C.E. conditions of contract.

The parameters in **Table 3.6** have been calculated as follows[17]:

1. Average Cost Increase Factor for costs incurred the n^{th} month,

$$(ACIF)_n = \left\{ \frac{(1.08)^{n/12} - (1.08)^{(n-1)/12}}{2} \right\}$$

2. Actual costs incurred during the n^{th} month

   $= (Cost)_n$ x $(ACIF)_n$

3. Price Fluctuation Factor for work carried out during the n^{th} month,

$$(PFF)_n = 0.9 \left\{ (1.08)^{n/12} - 1 \right\}$$

4. Payment of increased costs incurred for work completed during the n^{th} month,

   $= (Value)_n$ x $(PFF)_n$.

The calculated financial effects are as follows:

Tender Sum: Remains unchanged.

Interest on borrowed capital (@ 13½% p.a.): £7743

Total Cost: £912,047 + £7743 = £919,790

$$\text{Effective Margin} = \frac{935,015 - 919,790}{919,790} = 3.61\%$$

Maximum Cash Requirement: £143,607 at the end of month eight.

	MONTH													
	1	2	3	4	5	6	7	8	9	10	11	12	13	18
COST INCREMENT	23,760	35,550	64,830	78,520	81,010	92,740	103,440	93,490	78,700	82,260	66,410	50,610	22,800	-
AV COST INCREASE FACTOR	1 00322	1.00967	1.01617	1.02271	1 02929	1 03591	1 04257	1.04928	1.05603	1.06283	1.06967	1.07666	1 08347	-
ACTUAL CUMULATIVE COSTS	23,826	59,720	125,599	205,902	289,285	385,355	493,198	591,295	674,405	761,823	832,860	887,344	912,047	-
VALUE	46,673	57,824	69,216	88,032	96,049	100,274	99,803	97,902	88,641	72,534	58,264	40,604	-	-
PFF	0 00579	0.01162	0.01748	0.02334	0 02933	0.03631	0.04133	0.04738	0.05348	0 05961	0.06579	0.07200	-	-
INCREASED COSTS	270	672	1,210	2,056	2,878	3,641	4,126	4,639	4,741	4,324	3,833	2,923	-	-
RETENTION	-2,334	-2,891	-3,461	-4,402	-4,802	-5,014	-4,630	-	-	-	-	+13,767	-	+13,767
CUMULATIVE RECEIPTS	-	44,609	100,214	167,179	252,864	348,887	447,888	547,066	649,627	743,009	819,867	881,964	939,248	953,016

Table 3.6 Inflationary Project Cash Flow.

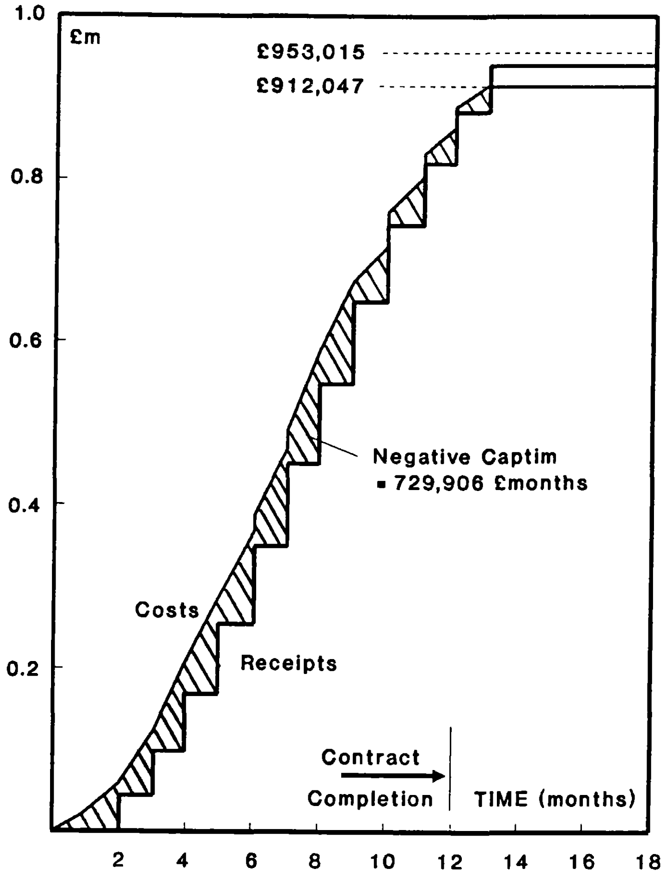


Figure 3.7 Project Cash Flow Chart with 8% Inflation.

3.4 Internal Rate of Return (IRR)

The internal rate or return of a project is the discount rate that make a serious of cash flows both negative and positive, costs and receipts, discount net, Net Present Value, equal to zero, Mathematically defined:

$$\sum_{t=0}^{n} \frac{A_t}{(1+r)^t} = 0$$

where A_t = cash flows in and out

r = internal rate of return

t = time period

n = project duration

In order to find the IRR, it is necessary to establish the value of the interest that will just equate the present worth of all of the future cash flows both positive and negative considered over the full period of the project.

The IRR can only be determined by using trail and error. This achieved by estimating a value of r, the required IRR, the present worth of the future net cash flow, is then calculated using this value, it is obviously saves time and calculation if the initial guess is reasonably close to the true rate of return.

If the resulting calculation gives a positive answer, then a second higher rate is chosen. This will smaller discount factor and hence a smaller discounted cash flow. If the net answer is now negative, the actual rate of return can be by interpolation between the two assumed rates. If the second answer is still positive, a further assumption of a higher discount rate needs to be made and the calculation repeated until two interest rates are reached that give one negative net summation and one positive net summation.

Interpolation between interest rates is carried out ass though the relationship between the interest factors and the present worth is linear in form. This is not in fact the case and therefore a small but not overly significant error occurs in the final interest rate calculated by linear interpolation. The closer are the two interest rates between which interpolation will take place, the smaller will tend to be the error. Using the computer in calculation the IRR under this research will overcome this problem and that's by using the method illustrated in the flow chart in Figure 3.8.

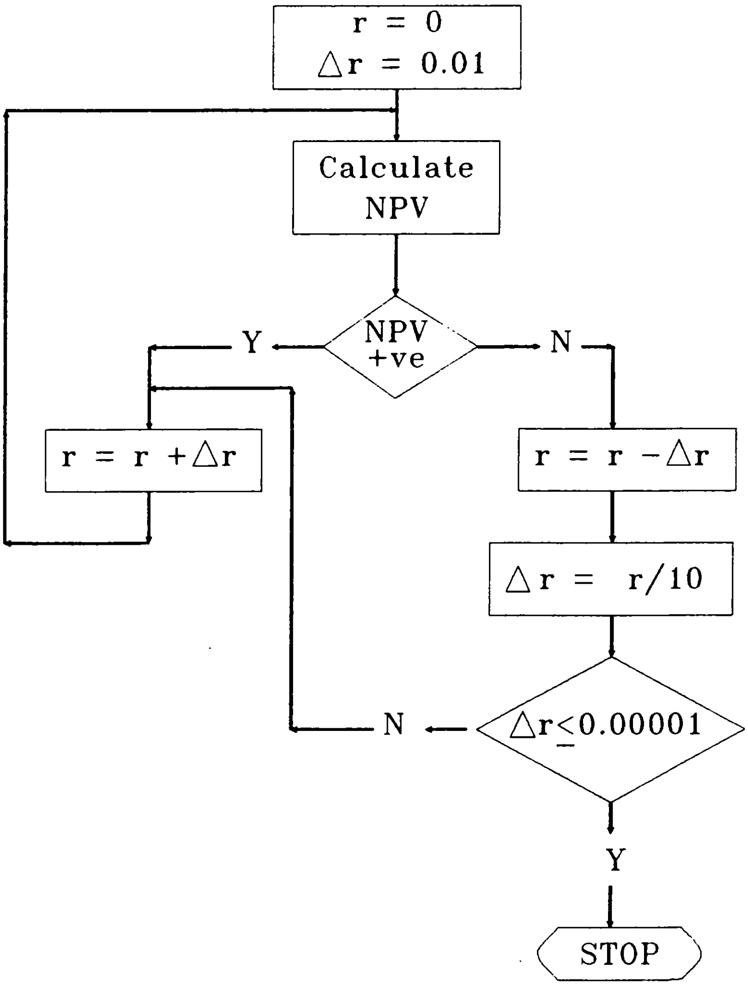


Figure 3.8 IRR Calculation Flow Chart

CHAPTER 4

THE ANALYSIS OF UNCERTAINTIES

Chapter 1 has shown that in some circumstances the analysis of uncertainty as part of the general analysis process is desirable and that, at the least, the confidence of the planner in his results is increased if he has available the means to apprise the risk. This chapter will review the two techniques which have been developed to handle the analysis of uncertainties, and will concentrate on finding solutions to the difficulties facing the chosen method for the development of the computer programs for this research project.

4.1 The Two Approaches

Statistics could be said to be the science of risk, and thus it is natural that a solution to the problem of handling risk should be sought there. The early network analysis packages did not, in general, provide facilities for the handling of uncertainty, but those which did adopted the PERT technique and used elementary statistics and a simple arithmetical process for the analysis.

As been mentioned previously, PERT was devised in the 1950s with one particular project in mind: the research and development required for the production of the Polaris missile[6]. Those responsible for the Polaris project were realistic enough to realise that a deterministic approach to the planning of the project would not be satisfactory. Since the uncertainty involved in such a research– centerd project is so great that it quickly swamps the analysis and makes deterministic forecasts literally incredible. To overcome this difficulty the developers of the PERT system built into their analysis not only a value for the expected duration of each activity but also a parameter representing the expected variation of the duration. These two variables were then carried through the analysis and all results were presented in terms of these parameters.

An alternative approach which, because it relies on the power of modern computer, was adopted rather later than PERT, is the so–called 'Monte–Carlo' simulation[6]. This technique, previously described, uses the power and patience of the computer to analyse the project model many times, each time allowing the project variables to vary randomly within their allotted distribution. The results are stored as a list of recorded values on which statistical operations can be carried out. This technique has the advantage over the PERT technique of flexibility, but it buys this advantage at the price of computer aid.

4.1.1 The Statistical Method

The techniques of statistics can be used to combine uncertain numbers, and these techniques were used by the developers of thr PERT system to enable them to produce useful answers from data which, because of the nature of the research and development task, were very uncertain[6]. Their method relied on the central limit theorem, which shows that the mean and the variance of the sum of a number of uncertain numbers can be easily predicted and that the distribution of the sum tends to be Gussian in form.

Using the central limit theorem it is possible, if the value of the mean and the variance for each component is known, quickly to calculate the mean and the variance of the sum. Knowing the values of the mean and the variance and knowing the form of the resulting distribution, the assessment of risk is trivial.

The central limit theorem holds for any shape of component distribution and thus the approach used in PERT is applicable to the asymmetrical duration distributions for activities which are typically found in project planning.

PERT was built around the'Beta' distribution as the standard for the component distribution. There is, however, no theoretical justification for this choice, merely the pragmatic reason that it provides a wide variety of distribution forms and the calculation associated with it are very simple.

The Beta distribution is of the form:

$$Y = X^a (1 - X)^b$$

which, depending on the values adopted by a and b, produces a range of distributions both skew and symmetrical **(Figure 4.1)**[6].

The beta distribution has the advantages over other distributions of simplicity and flexibility.

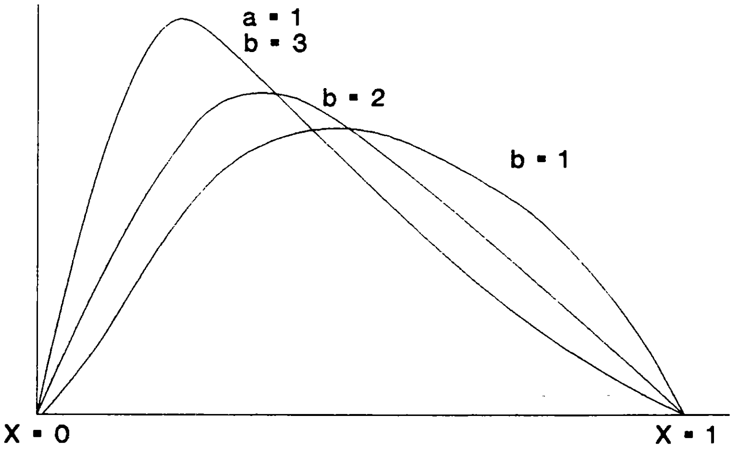


Figure 4.1 The Beta Distribution.

The distributions are defined in PERT not by the specification of statistical parameters, which would be foreign (and frightening) to most planners, but by the specification of three values of duration, the most optimistic estimate, the estimate of the most likely duration, and the most pessimistic estimate. The optimistic and pessimistic estimates have a 10% chance of being optimistic (in the case of the pessimistic values) or pessimistic (in the case of the optimistic values). Denoting these values as *a*, *b* and *c*, the parameters of the activity distribution can be calculated as

$$\bar{X} = \frac{a + 4b + c}{6}$$

$$\sigma = \frac{c - a}{4}$$

Thus it is possible very quickly to move from estimates of duration, which are familiar ground to the planner, to the statistical parameters required for use in the statistical analysis. These parameters can then be used within the analysis of the network which, using the central limit theorem, can merely add the values of the activities on the critical path in order to produce the parameters for the distribution of completion date.

This approach is powerful and allows the planner even without the aid of computer very quickly to produce distributions of event times and thus estimates of risk overrun. The calculations required of the planner are simple and the planner is, for most of the calculation, working with numbers which are significant to him (an important advantage if errors are to be avoided). The approach does, however, have an important difficulty, the handling of parallel critical paths.

4.1.1.1 Interfering Critical Paths

For a simple network with a unique critical path and considerable float elsewhere in the network the simple addition of the values of mean duration and of variance for each activity on the critical path is accurate. Such a situation is shown in **Figure 4.2**[6]. In this simple network there are no circumstances in which path A can become critical; the length of the project is always governed by path B.

Difficulties arise if there are parallel paths which may each be critical in different circumstances. This situation is shown in **Figure 4.3**[6]. In this case it is possible for the network path which is normally critical (path B) to become subcritical because another path (path A) exceeds it. Either of the paths can now be critical and the distribution of completion date must include both possibilities, because duration of less than 36 days can be produced with either A or B critical and each possibility must contribute to the probability of completion within a given time.

It is, of course, possible to handle this problem of interfering critical paths using the methods of classical statistics. For this case the distribution can be shown to be given by[6]

$$f(X) - f_A(X) \int (f_B((0 \longrightarrow X))dX) \qquad \text{A is critical}$$

$$+ f_B(X) \int (f_A((0 \longrightarrow X))dX) \qquad \text{B is critical}$$

but this calculation becomes onerous for large and complex networks where the number of possible interferences is much larger than the two described here.

The difficulty associated with the use of PERT for complex networks dictates the need for an alternative approach in such situations. This does not imply that PERT is not a useful tool, merely that in complex networks it loses the simplicity which is its major strength[6], One alternative to PERT, the 'Monte Carlo simulation', has been available for many years but is becoming increasingly attractive as the cost of computer power, on which it is dependent, is reduced.

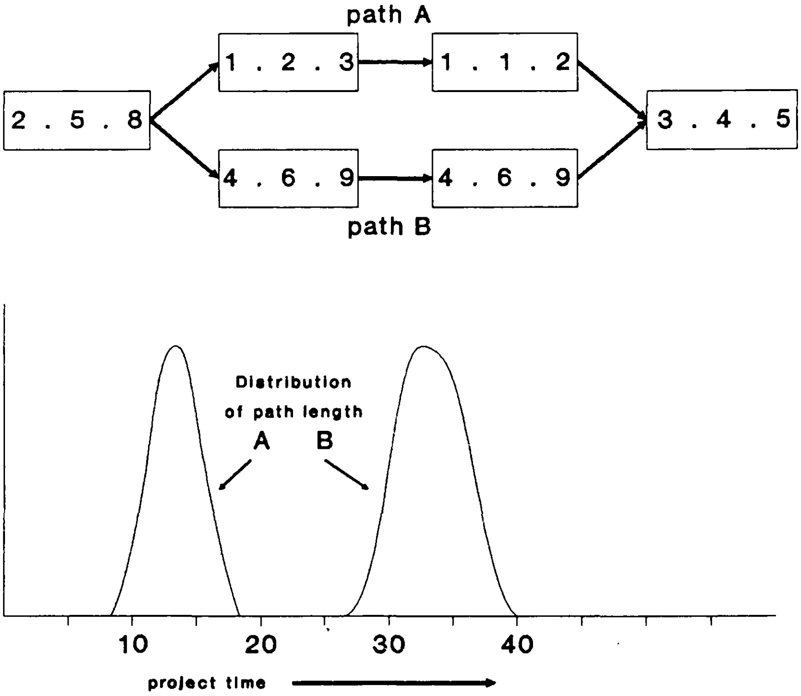


Figure 4.2 A Unique Critical Path.

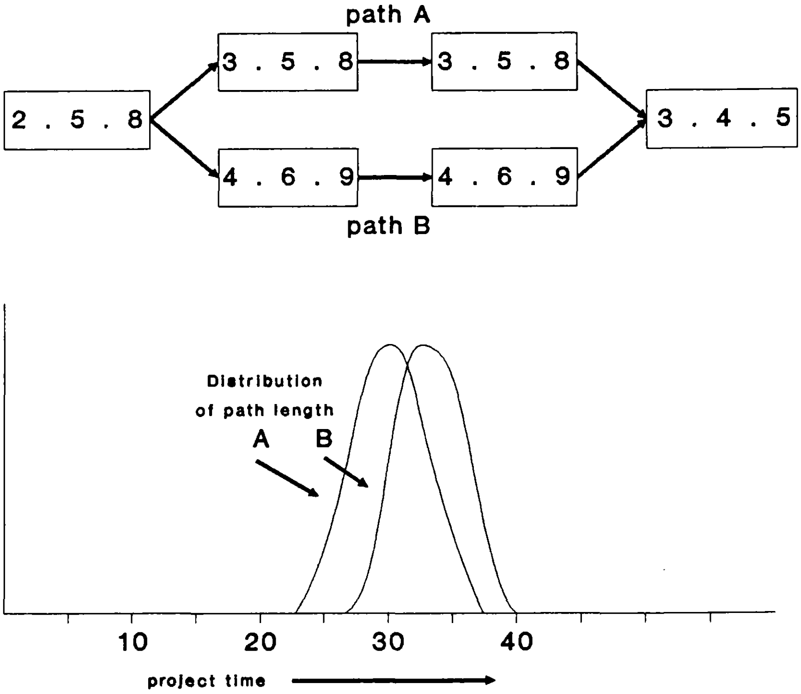


Figure 4.3 Interfering Critical Paths.

4.1.2 Simulation Techniques

The analysis of uncertainty described before relies on carrying the parameters of uncertainty through the analysis and, using the method of statistics, combining them to provide parameters of uncertainty of the result. PERT carries out this calculation using the same method as would have been used in a purely deterministic analysis, but the analysis is done once only and the range of possible answers is represented in the result.

An alternative approach is to create the uncertainty of the result not by carrying uncertainty through a single analysis but by carrying a large number of analyses using data values chosen at random from the original ranges. Thus the distribution comes not from the carrying of uncertainty through a single analysis but from the combination of the results of a large number of deterministic analyses. This alternative technique is known as Monte Carlo simulation.

It is obvious that the distribution produced by a simulation analysis is dependent upon the data values of the component parts. The analysis seeks to model many repetitions of the project and thus must take, as input, values of the variable (in this case duration) which are chosen from a menu of equally probable values going into analysis must consist not only of the choice at random of a value from within the specified range, but must also weight the choice towards the most probable values. This weighted choice is made by dividing the distribution of the variable into equal areas (that is area of equal probability) and then by chosing at random between these areas. **Figure** 4.4[6] shows how, assuming a menu of ten values and a beta distribution, this would be done manually.

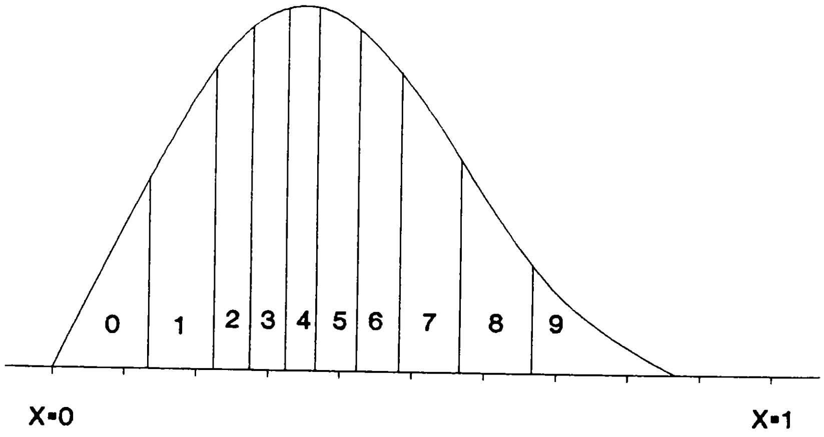


Figure 4.4 Zones of Equal Probability.

When the distribution of each component data value has been specified and divided in this way, a random choice is made between the equiprobable zones for each component and one cycle of the simulation analysis is carried out by allotting a value for the duration of each activity to that which represents the chosen zone. Many cycles are carried out and the results of the analyses are combined to provide distributions for specified events.

4.2 Monte Carlo Simulation

This technique is based on experimentation and simulation and is used in situations where a solution in the form of an equation would be difficult or impossible. The basic steps are:

(a) The range of values of the risk being considered are assessed, together with the probability distribution most suited to each risk.

(b) A value for each risk within its specified range is selected; this value should be randomly chosen and must take account of probability distribution.

(c) The outcome of the project is calculated using the combination of values selected for each one of the risk.

(d) The calculation is repeated a number of times to obtain the probability distribution of the project outcome; the iterations required depends on the number of variables and the degree of confidence required, but typically lies between 100 and 1000[9].

4.3 The Difficulties of Simulation

A simulation analysis is a useful part of an interactive design system only if it is fast. This need for speed must be recognised in the writing of the computer programs and, at times, it may be necessary to sacrifice accuracy on this particular altar. The saving in calculation time can be achieved, without serious loss of accuracy, by reducing the number of simulation cycles and by storing cumulative rather than individual values of data. The third area of saving is the generation of the individual distribution prior to the selection of deterministic duration values. The implication of these recommendations to the development of a computer program must now be considered.

4.3.1 The Necessary Number of Simulations

If the analyst wishes accurately to predict the distribution of any event in the network then he will require a very large number of simulation cycles to produce it. Reducing the number will produce progressively less satisfactory result. If, however, the analyst requires an estimate of mean event time and is satisfied with an assumption of a normal distribution and an estimated value of standard deviation, then the number of simulation cycles can be enormously reduced[6].

4.3.2 Data Storage

Even in large computers the problem of data storage can be troublesome if both the number of events in a network and the number of simulation cycles is high. This problem can be overcome by combining the data as they are produced and carrying forward a cumulative total. This technique, bringing the advantage of reducing the amount of memory required or of increasing the speed of operation of the program, has the disadvantage that it destroys the shape of the distribution and imposes a normal distribution on the result. Previous studies[6] have shown that this is not a serious error at least for events away from the early events near the start of the network.

4.3.3 Distribution Calculation

It has become usual to present the distribution of activity duration by a Beta distribution given by the formula

$$Y = X^a (1 - X)^b$$

There is no theoretical basis for this choice; it is chosen merely because the curve can be moulded, by the selection of the parameters *a* and *b*, to fit the curve to a shape that experienced planners would draw to describe the distribution of durations. The curve can be symmetrical or asymmetrical, flat or sharp. Where manual calculations are being carried out, particularly within the PERT program, the distribution provides easy formula for both mean and standard deviation and is thus convenient.

When the method of analysis is changed to a Monte Carlo simulation, as is being described here, and when the means of doing the work is moved from the manual environment to the computer, then the Beta distribution becomes rather less convenient. No direct use is now made of the mean and standard deviation of the component distributions; instead the centroids of the zones of equal probability must be identified.

Clearly this calculation does not present any difficulty: given the three duration values as is usual in PERT (and assuming a value *b*) the value of *a* can be calculated. Progressive numerical integration of the curve can then be used to produce the positions of the various centroids and these can be used as the menu items from which the random choice is made prior to each simulation cycle. While this calculation is simple, it is also cumbersome and time consuming, especially when it must be repeated for each activity for each simulation cycle. There is merit, therefore, in identifying a distribution which allows the direct calculation of the centroids of equiprobable zones.

It has been proved from previous studies[6] that the results of simulation analyses are insensitive to the shape of the individual activity duration distribution if the number to be combined is greater than two or three, and that this condition applies to all the events in the network apart from those very near the beginning of the forward pass. In view of this insensitivity it seems to be unnecessarily pedantic to insist on the use of a traditional but not theoretically derived beta distribution, for any distribution which approximates to the general shape of the expected duration could be used. When this is considered along with the need for a geometrically simple shape, it seems that a triangular distribution could form a compromise between simplicity, giving fast calculation times, and shape, giving a realistic representation. **Figure 4.5**[6] contrasts the triangular distribution recommended with the beta distribution it replaces.

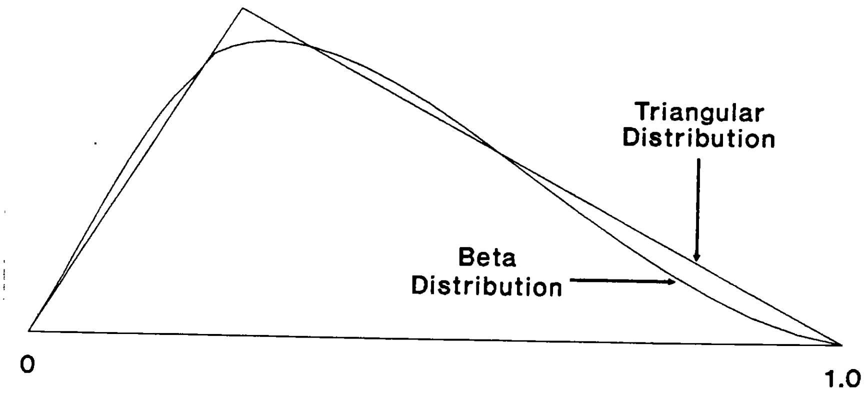


Figure 4.5 Triangular and Beta Distribution Compared.

4.4 Triangular Probability Distribution

4.4.1 Calculation of the Parameters

From **Figure 4.6** f_o, f_m and f_p are the probability values at the durations d_o, d_m and d_p respectively.

Known parameters:

Optimistic duration d_o

Most likely duration d_m

Pessimistic duration d_p

d_o and d_p are chosen such that there is respectively a 99% and 1% probability of exceedence.

Overall area = 1.0

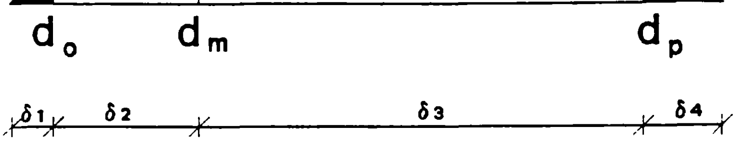


Figure 4.6 Distribution Parameters.

$$\tfrac{1}{2}f_m(\delta_1+\delta_2+\delta_3+\delta_4) = 1.0 \qquad (1)$$

Shaded area = 0.01

$$\tfrac{1}{2}\delta_1 f_o = 0.01 \qquad (2)$$

$$\tfrac{1}{2}\delta_4 f_p = 0.01 \qquad (3)$$

Linearity

$$\frac{f_o}{\delta_1} = \frac{f_m}{\delta_1+\delta_2} \qquad f_o = \frac{f_m\delta_1}{\delta_1+\delta_2} \qquad (4)$$

$$\frac{f_p}{\delta_4} = \frac{f_m}{\delta_3+\delta_4} \qquad f_p = \frac{f_m\delta_4}{\delta_3+\delta_4} \qquad (5)$$

from equation (2)

$$f_o = \frac{0.02}{\delta_1}$$

Substitute in equation (4)

$$\frac{0.02}{\delta_1} = \frac{f_m \delta_1}{\delta_1 + \delta_2}$$

$$f_m \delta_1{}^2 - 0.02\delta_1 - 0.02\delta_2 = 0 \qquad (6)$$

Equation (6) is a quadratic which may be solved for δ_1 from:

$$\delta_1 = \frac{0.02 \pm \sqrt{(0.02)^2 + 0.08 f_m \delta_2}}{2f_m} \qquad (7)$$

But

$$\sqrt{0.02^2 + 0.08 f_m \delta_2} \quad > \quad 0.02 \text{ for all } f_m, \delta_2{}^*$$

Since δ_1 must be > 0 for all cases:*

$$\delta_1 = \frac{0.02+\sqrt{0.04+0.08f_m\delta_2}}{2f_m} \tag{8}$$

From equation (3) $f_p = \frac{0.02}{\delta_4}$

Substitute in equation (5)

$$f_m\delta_4 - 0.02^2\delta_4 - 0.02\delta_3 = 0 \tag{9}$$

Using similar reasoning as above, this yields

$$\delta_4 = \frac{0.02+\sqrt{0.04+0.08f_m\delta_3}}{2f_m} \tag{10}$$

Calculation of the distribution parameters is carried out in accordance with the flow chart in **Figure 4.7**

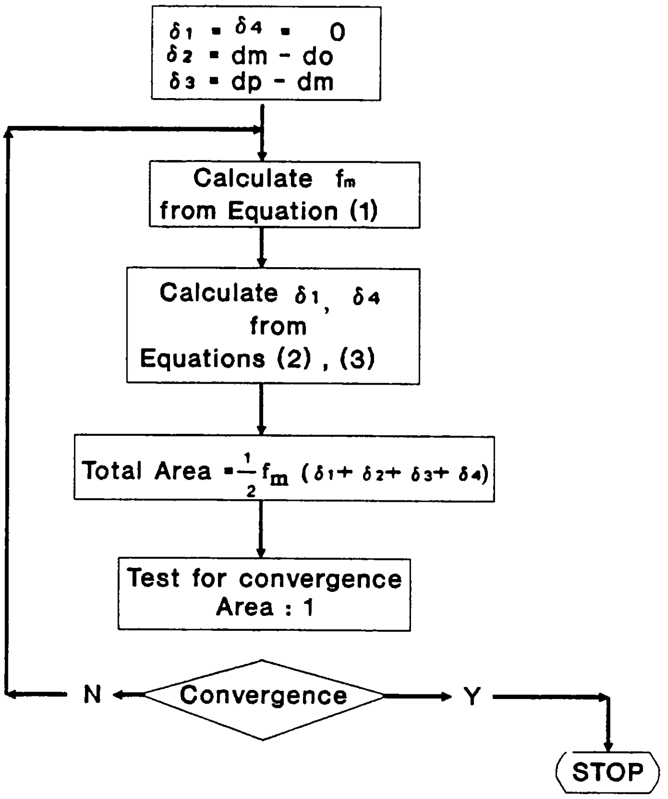

δ1 = δ4 = 0
δ2 = dm - do
δ3 = dp - dm
Calculate fm from Equation (1)
Calculate δ1, δ4 from Equations (2) , (3)
Total Area = 1/2 fm (δ1+ δ2+ δ3+ δ4)
Test for convergence Area : 1
N
Convergence
Y
STOP

Figure 4.7 Distribution Parameters Calculation Flow Chart

4.5 Random Selection of Task Duration

The essence of so–called 'Monte Carlo' simulation technique is that, for each task, a duration is randomly selected from the range of durations defined by the appropriate probability distribution, with the selection being weighted towards the most probable value.

There are two approaches by which this can be achieved, both of which make use of a pseudo random number generator which outputs a series of uniformly distributed random numbers within a specified range.

The two approaches are detailed hereunder for the particular case of a triangular probability distribution. It may be assumed that the principles remain the same for any distribution provided that the parameters of that distribution are known.

4.5.1 Equi–probability Bands

The overall zone of probability (i.e. the distribution area) is subdivided into N zones of equal probability as shown in **Figure 4.8**

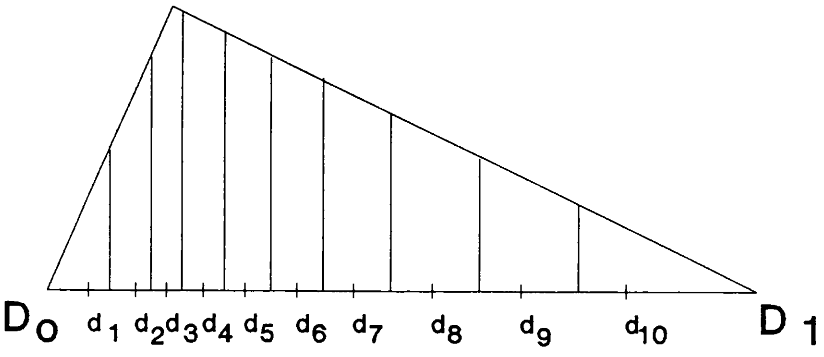


Figure 4.8 Equi–probability Bands

For convenience, each equi–probability band is associated with a single duration (d_i ,i = 1,N) which is taken to be the duration value corresponding to the band centroid.

Owing to the non linear native of the equations used to determine the band limits and centroidal values, the band parameters are evaluated progressively, using the N^th^ band parameters as a basis for evaluating the (N+1)th band parameters.

In any given instance, the task duration d_i is selected on the basic of a uniformly distributed random in the range $1 \leqslant i \leqslant N$.

4.5.2 Direct Calculation Method

Consider the triangular probability distribution shown in **Figure 4.9**

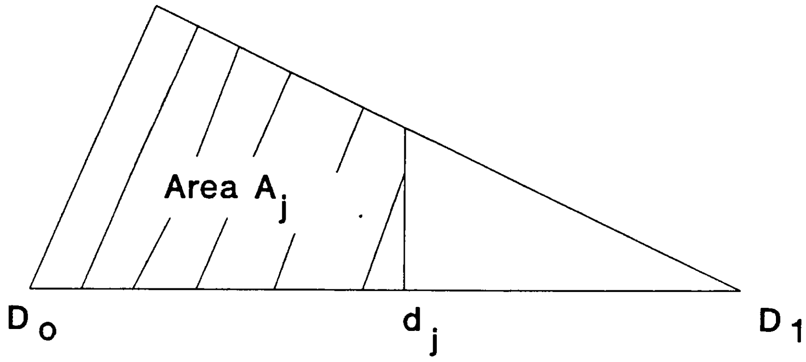


Figure 4.9 Direct Calculation

For a randomly selected duration d in the range $D_0 \leqslant d \leqslant D_1$:

$$P(\,d \leqslant d_j\,) = A_j$$

where $A_j = 0$ for $d_j = D_0$

$A_j = 1$ for $d_j = D_1$

Thus a given value of A_j in the range $0 \leqslant A_j \leqslant 1$ may be associated with a value of d_j in the range $D_0 \leqslant d_j \leqslant D_1$.
A few calculations shoe that if values of Aj are randomly selected from a uniformly distributed series of values $0 \leqslant A_j \leqslant 1$, the resultant distribution of d_j is of the form shown.

In any instance, therefore, the task duration may be calculated directly from a uniformly distributed random number in the range $0.0 \leqslant A \leqslant 1.0$

4.5.3 Comparision

By comparing the two approaches, it has been found that:

In terms of computational requirement the 'Equiprobability' band approach has two major disadvantages

(a) A significant amount of calculation is required to define the bands for each task.

(b) Once defined, the centroidal duration for each task require storage for subsequent use. This require significant memory space.

In terms of flexibility the Equiprobability band approach precludes the use of integer tasks duration. Thus given rise to a number of disadvantages:

(1) Operations with floating point numbers are substantially slower than operations with integers.

(2) Fractional tasks durations introduces unnecessary complications into the 'Cash Flow' computations which would not arise with integer task durations.

(3) In reality, although a task may very well be completed in, say 10.30 days it is unlikely that the succeeding task will commence until the start of day 12. Thus, the duration of the task is effectively expanded to 11 days i.e an integer duration.

The above leads to select the 'Direct calculation' method for the development of the computer subroutine required.

4.5.4 Selection of Task Duration Using the Direct Calculation Method

From **Figure 4.10**

Overall area = 1

$$S_0 = \frac{f_m}{D_m - D_0} \quad ; \quad S_1 = \frac{f_m}{D_m - D_1}$$

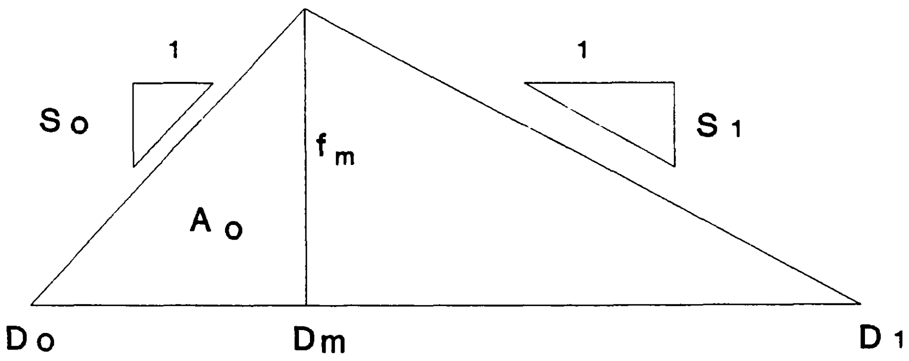


Figure 4.10 Selection of Task Duration Using The Direct Calculation

Case 1 $(0 \leqslant A \leqslant A_0)$

$$d_j = D_0 + \frac{2A}{\sqrt{S_0}}$$

Case 2 $(A_0 < A \leqslant 1)$

$$d_j = D_1 - \frac{2(1-A)}{\sqrt{S_1}}$$

CHAPTER 5

THE COMPUTER PROGRAMS

5.1 Introduction

As suggested in chapter 4, the analysis of uncertainty in a large and complex project is best carried out by performing a large number of project simulations, based on data values (such as activity durations) chosen at random from within specified probability distribution, resulting in a range of possible solutions (such as project durations) defined by a single probability distribution.

In order to apply this technique, known as 'Monte Carlo' simulation, the suite of PERTRA programs was developed to operate on the basis of project data (task durations, logical dependencies, resource usage etc.) output, in the form of a text file, by PERTRMASTER ADVANCE – one of the most widely used commercial software packages for construction planning.

The PERTRA suite comprises three separate programs as follows:

1. PERTRA1: Data Reading and Initialisation;
2. PERTRA2: Interactive Input of Task Duration Data;
3. PERTRA3: Risk Analysis and Cash Flow Calculations;

All three programs are written in FORTRAN 77 and, in order to utilise the graphic support provided thereby, PERTRA1 and PERTRA2 compiled using some of the 'C' language graphics subtoutines.

The programs will now be discussed in turn.

5.2 PERTRA1– Data Reading and Initialisation

The essence of the program is as follows:

1. Reading of selected data from a user–specified file containing project data output by PERTRMASTER ADVANCE.

2. Initialisation of this data to convert it to a form suitable for use by PERTRA2 and PERTRA3.

3. Creation of intermediate data files for use by PERTRA2 and PERTRA3.

The program flow chart is shown in **Figure 5.1** and the program listing is given in **Appendix C.1**.

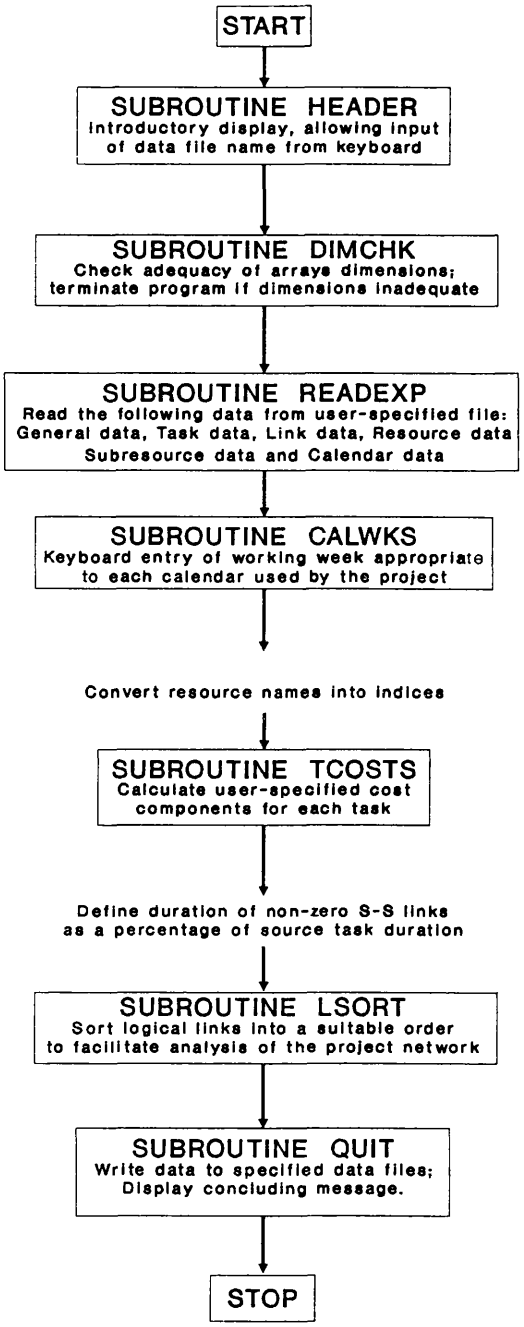

START
SUBROUTINE HEADER
Introductory display, allowing input of data file name from keyboard
SUBROUTINE DIMCHK
Check adequacy of arrays dimensions; terminate program if dimensions inadequate
SUBROUTINE READEXP
Read the following data from user-specified file: General data, Task data, Link data, Resource data Subresource data and Calendar data
SUBROUTINE CALWKS
Keyboard entry of working week appropriate to each calendar used by the project
Convert resource names into indices
SUBROUTINE TCOSTS
Calculate user-specified cost components for each task
Define duration of non-zero S-S links as a percentage of source task duration
SUBROUTINE LSORT
Sort logical links into a suitable order to facilitate analysis of the project network
SUBROUTINE QUIT
Write data to specified data files; Display concluding message.
STOP

Figure 5.1 PERTRA1 Flow Chart

The main program -routines, subroutines and functions are now discussed:

5.2.1 Subroutine HEADER

The purpose of this subroutine is to generate an introductory screen display which provides general information, pertaining to the use of PERTRA1, and to enable keyboard input of the PERTMASTER export data file name.

The above is carried out as shown in the flow chart in **Figure 5.2.** The program listing is given in **Appendix C.2.**

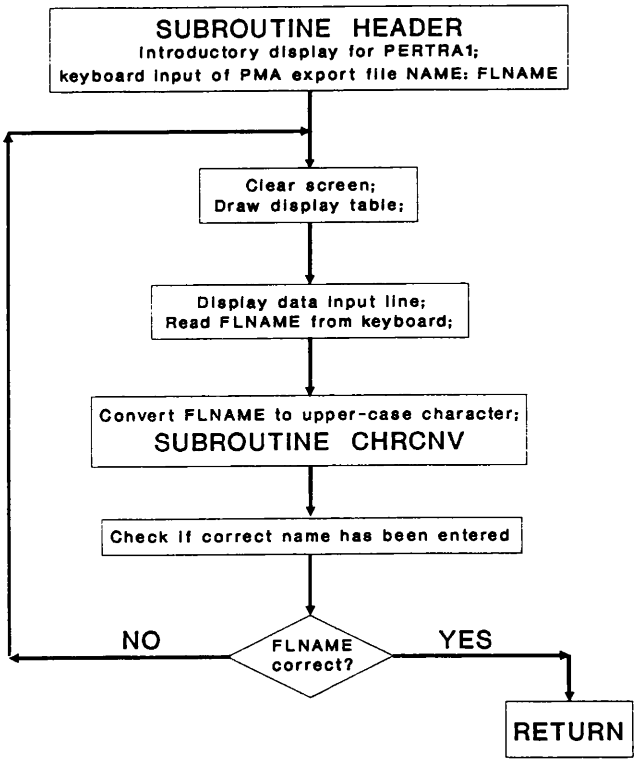

SUBROUTINE HEADER
Introductory display for PERTRA1;
keyboard input of PMA export file NAME: FLNAME
Clear screen;
Draw display table;
Display data input line;
Read FLNAME from keyboard;
Convert FLNAME to upper-case character;
SUBROUTINE CHRCNV
Check if correct name has been entered
NO
FLNAME correct?
YES
RETURN

Figure 5.2 Flow Chart for the Subroutine HEADER.

5.2.2 Subroutine READEXP

This subroutine reads data from the user–specified export file (FLNAME.EXP) generated by PERTMASTER.

The contents of the data export file are formatted as shown in Appendix... The abstraction of data from the file is carried out in accordance with the flow chart shown in **Figure 5.3**. The subroutine listing is given in **Appendix** C.4.

As can be seen in **Appendix A**, the contents of each record in the export file contain a number of parameters which may or may not be required by the PERTRA programs. Each of these parameters is 'comma delimited'. Selective data abstraction, therefore, is carried out on the basis of the 'comma–related' position of the relevant parameters in each record. This is achieved by calling subroutine RECCOM, immediately after reading any given record, which determines the number of commas (NCOMMA) in the record and the positions thereof (ICOMMA(I), I=1,NCOMMA).

Selective data abstraction is carried out by the following subroutines called by subroutine READEXP, with the choice of subroutine being determined by type of record (denoted by the initial character of the record) required to be read.

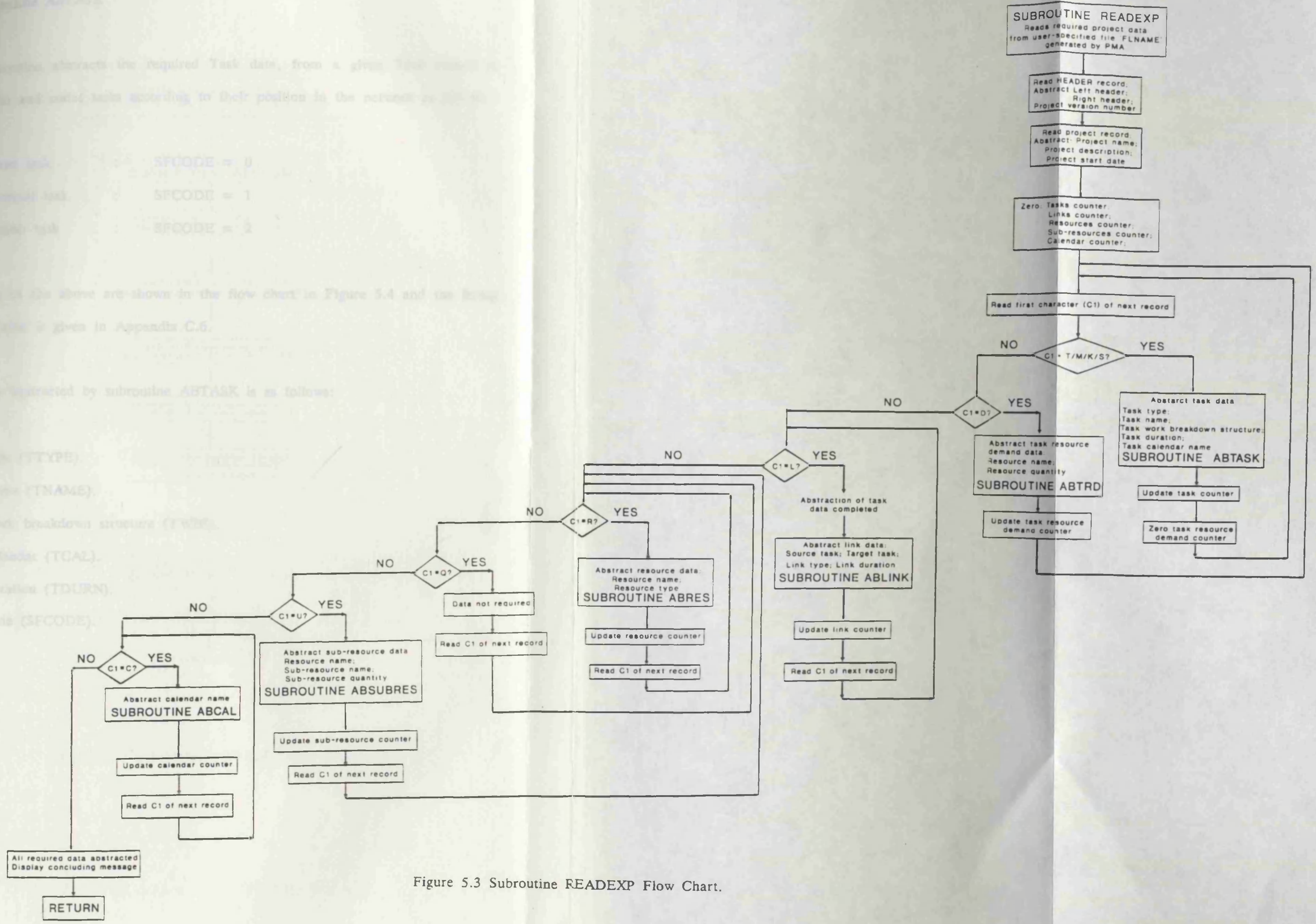


Figure 5.3 Subroutine READEXP Flow Chart.

5.2.2.1 Subroutine ABTASK

This subroutine abstracts the required Task data, from a given Task record in the export file and codes tasks according to their position in the network as follows:

Start task	:	SFCODE = 0
Normal task	:	SFCODE = 1
Finish task	:	SFCODE = 2

The steps of the above are shown in the flow chart in **Figure 5.4** and the listing of the subroutine is given in **Appendix C.6.**

Task data abstracted by subroutine ABTASK is as follows:

1. Task type (TTYPE).
2. Task name (TNAME).
3. Task work breakdown structure (TWBS).
4. Task calendar (TCAL).
5. Task duration (TDURN).
6. Task code (SFCODE).

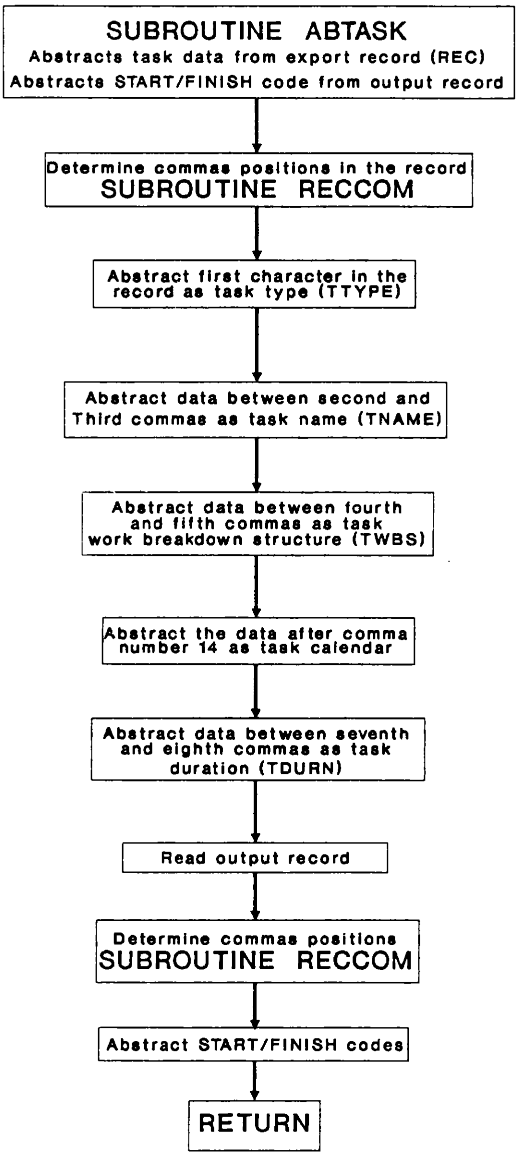

SUBROUTINE ABTASK
Abstracts task data from export record (REC)
Abstracts START/FINISH code from output record
Determine commas positions in the record
SUBROUTINE RECCOM
Abstract first character in the record as task type (TTYPE)
Abstract data between second and Third commas as task name (TNAME)
Abstract data between fourth and fifth commas as task work breakdown structure (TWBS)
Abstract the data after comma number 14 as task calendar
Abstract data between seventh and eighth commas as task duration (TDURN)
Read output record
Determine commas positions
SUBROUTINE RECCOM
Abstract START/FINISH codes
RETURN

Figure 5.4 Subroutine ABTASK Flow Chart.

5.2.2.2 Subroutine ABTRD

This subroutine abstracts the following Task Resource Demand data from the Task Resource demand records:

1. Task resource name (CTRES).
2. Task resource quantity (CTRESQ).

The subroutine flow chart is shown in **Figure 5.5** and the listing is given in Appendix C.7.

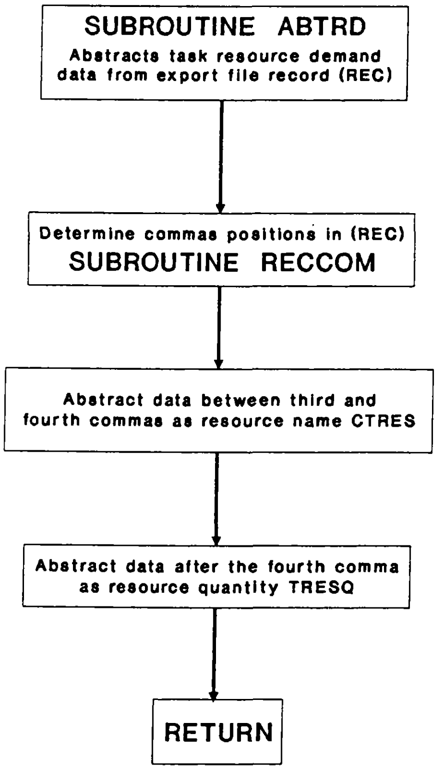

SUBROUTINE ABTRD
Abstracts task resource demand data from export file record (REC)
Determine commas positions in (REC)
SUBROUTINE RECCOM
Abstract data between third and fourth commas as resource name CTRES
Abstract data after the fourth comma as resource quantity TRESQ
RETURN

Figure 5.5 Subroutine ABTRD Flow Chart.

5.2.2.3 Subroutine ABLINK

This subroutine abstracts the following link data from a link record:

1. Source Task Name (CLFROM);

2. Target Task Name (CLTO);

3. Link Type (CLTYPE);

4. Link Duration (LDURN).

As can be seen in **Appendix A**, the first three of the above parameters are alphanumeric variables and require to be abstracted as such. In FORTRAN programs, however, operations involving alphanumeric variables are considerably slower and less efficient than those involving numeric data. It is, therefore, preferably to use the latter wherever possible. This being the case, subroutine ABLINK converts source/target task names to indices and link types to numbers. As an example the Export File link data is shown below:

CLEAR1	SETUP	F–S	0
CLEAR2	FINISH	F–S	0
CMPRESS	SSEPCO	F–F	0
CMPRESS	SSWPCO	F–F	0
CPESC1	CPC1FW	F–S	0
CPESC1	CPC1RF	S–S	2

Converted link data will be as follows:

1	209	1	0
2	40	1	0
3	248	3	0
4	258	3	0
15	6	1	0
15	8	2	2

The subroutine flow chart is shown in **Figure 5.6**, with the listing being given in Appendix C.8.

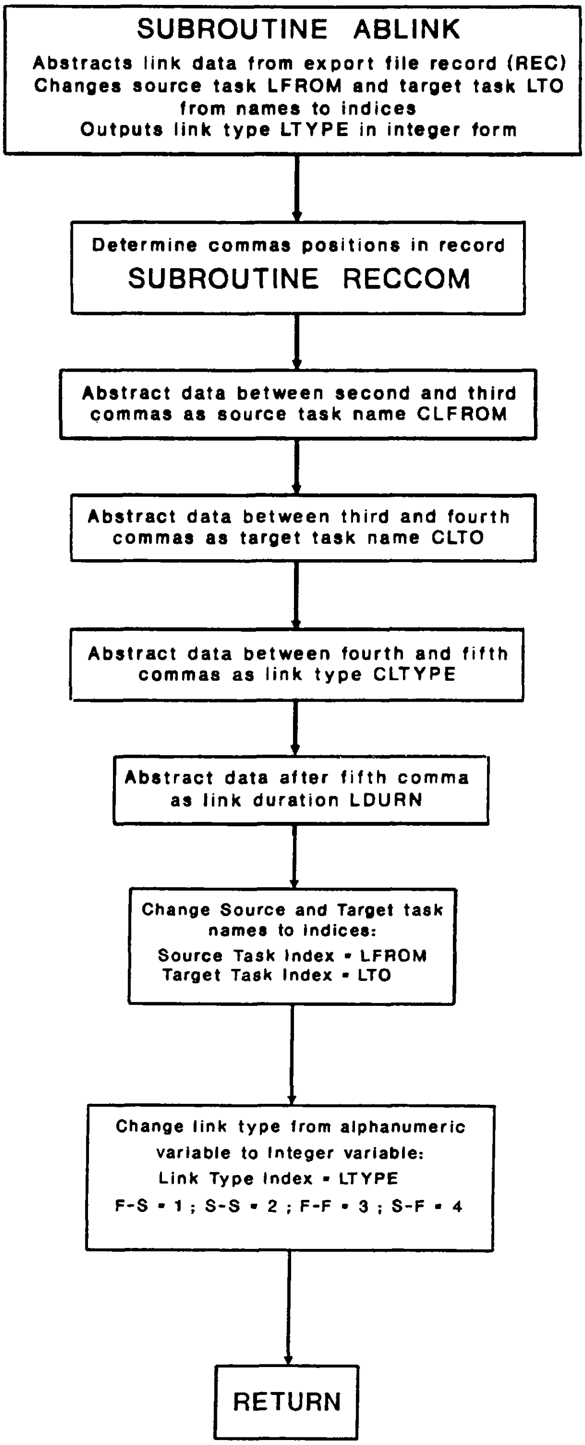

SUBROUTINE ABLINK
Abstracts link data from export file record (REC)
Changes source task LFROM and target task LTO from names to indices
Outputs link type LTYPE in integer form
Determine commas positions in record
SUBROUTINE RECCOM
Abstract data between second and third commas as source task name CLFROM
Abstract data between third and fourth commas as target task name CLTO
Abstract data between fourth and fifth commas as link type CLTYPE
Abstract data after fifth comma as link duration LDURN
Change Source and Target task names to indices:
Source Task Index = LFROM
Target Task Index = LTO
Change link type from alphanumeric variable to Integer variable:
Link Type Index = LTYPE
F-S = 1 ; S-S = 2 ; F-F = 3 ; S-F = 4
RETURN

Figure 5.6 Subroutine ABLINK Flow Chart.

5.2.2.4 Subroutine ABRES

This subroutine abstracts the following resource data from a resource record.

1. Resource Name (RNAME).
2. Resource Type (RTYPE).

The abstraction of the resource data is carried out as illustrated in the subroutine flow chart in **Figure 5.7**. The subroutine listing is given in **Appendix** C.9.

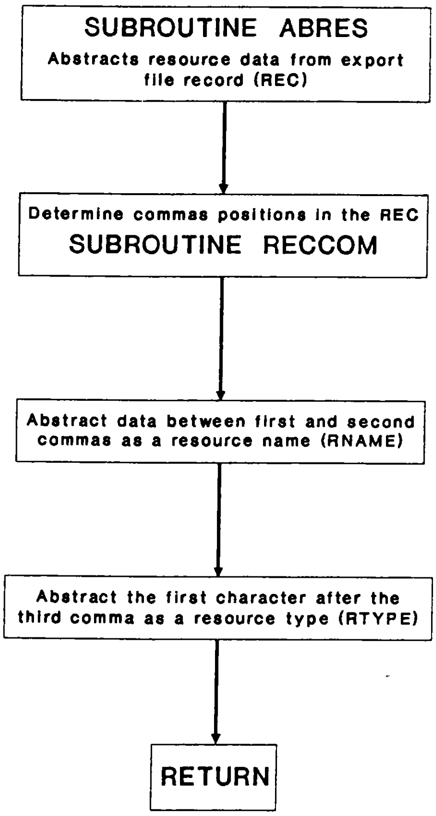

SUBROUTINE ABRES
Abstracts resource data from export file record (REC)
Determine commas positions in the REC
SUBROUTINE RECCOM
Abstract data between first and second commas as a resource name (RNAME)
Abstract the first character after the third comma as a resource type (RTYPE)
RETURN

Figure 5.7 Subroutine ABRES Flow Chart.

5.2.2.5 Subroutine ABSUBRES

Abstracts the following SUB– RESOURCE data from a sub– resource record:

1. Resource Name (CRTOP).
2. Sub– resource Name (CRSUB).
3. Sub– resource Quantity (RSUBQ).

The flow chart of the subroutine ABSUBRES is shown in **Figure 5.8**. Listing of the subroutine is given in **Appendix C.10**.

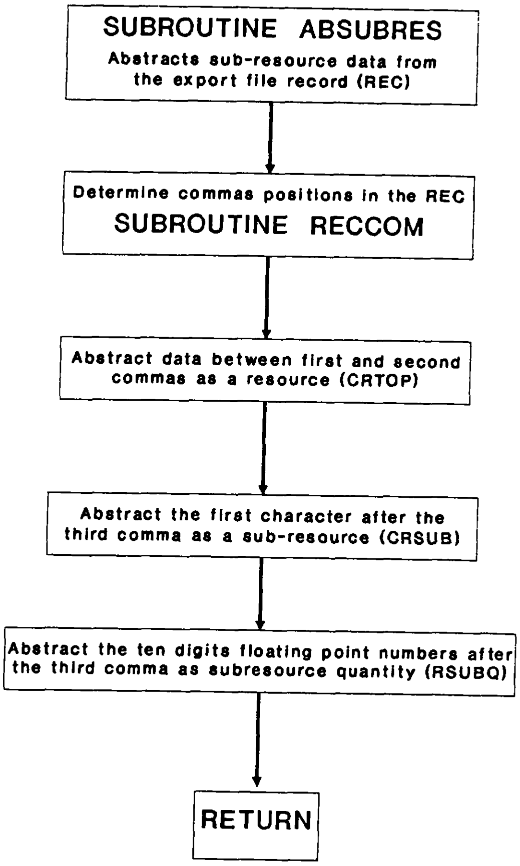

SUBROUTINE ABSUBRES
Abstracts sub-resource data from the export file record (REC)
Determine commas positions in the REC
SUBROUTINE RECCOM
Abstract data between first and second commas as a resource (CRTOP)
Abstract the first character after the third comma as a sub-resource (CRSUB)
Abstract the ten digits floating point numbers after the third comma as subresource quantity (RSUBQ)
RETURN

Figure 5.8 Subroutine ABSUBRES Flow Chart.

5.2.2.6 Subroutine ABCAL

This subroutine abstracts the Calendar Name (CAL) from a calendar record. The listing of the subroutine is given in **Appendix C.10**.

5.2.3 Subroutine DIMCHK

One of the major disadvantages associated with the use of FORTRAN, in the context of reading data files of varying lengths, is the requirement that array dimensions should be specified at the time of compiling the program. In the event that the amount of data, being read by a FORTRAN program from such a file, exceeds the specified capacity of the arrays, however, the program continues to read the 'excess' data and overwrites (to the appropriate extent) the previously-filled arrays. Since no error message is displayed, the user is blissfully unaware of such happening.

To prevent such a possibility occurring in PERTRA1, the subroutine DIMCHK has been included in the program immediately before subroutine READEXP.

Subroutine DIMCHK operates in a similar manner to subroutine READEXP, but reads only the first character of each record, determines its type and accumulates the numbers of each type of record. Should these numbers exceed the specified array dimensions, an error message is displayed and the program is terminated. The subroutine listing is given in **Appendix C.3**.

5.2.4 Subroutine CALWKS

This subroutine determines working week length (ICWK) for each task by facilitating keyboard input of the working week length for each calendar. This is carried out in accordance with the subroutine flow chart in **Figure 5.9.** Listing of the subroutine CALWKS is given in **Appendix C.12.**

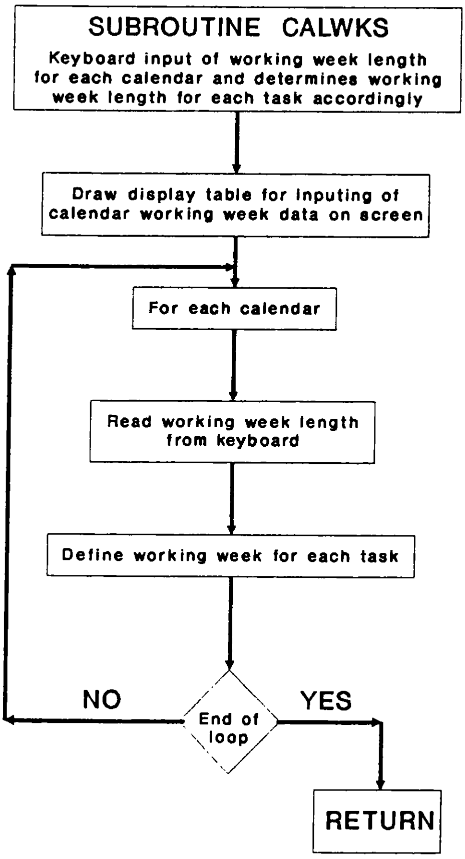

SUBROUTINE CALWKS
Keyboard input of working week length for each calendar and determines working week length for each task accordingly
Draw display table for inputing of calendar working week data on screen
For each calendar
Read working week length from keyboard
Define working week for each task
NO
End of loop
YES
RETURN

Figure 5.9 Subroutine CALWKS Flow Chart.

5.2.5 Subroutine TCOSTS

This subroutine enables keyboard input of cost— component data in order to determine the following cost components for each task in the network:

Labour cost

Material cost

Plant cost daily

Plant cost front end

Plant cost back end

Subcontract cost

Overheads daily

Overheads front end

Task price

For each component, the following data is input from the keyboard:

1. Component Code (e.g. XLC for labour cost).
2. Paymeny Accumulation (Monthly/Weekly)
3. Payment Delay (Days/Weeks/Months).

The subroutine flow chart is shown in **Figure 5.10** and the subroutine listing is given in **Appendix C.13.**

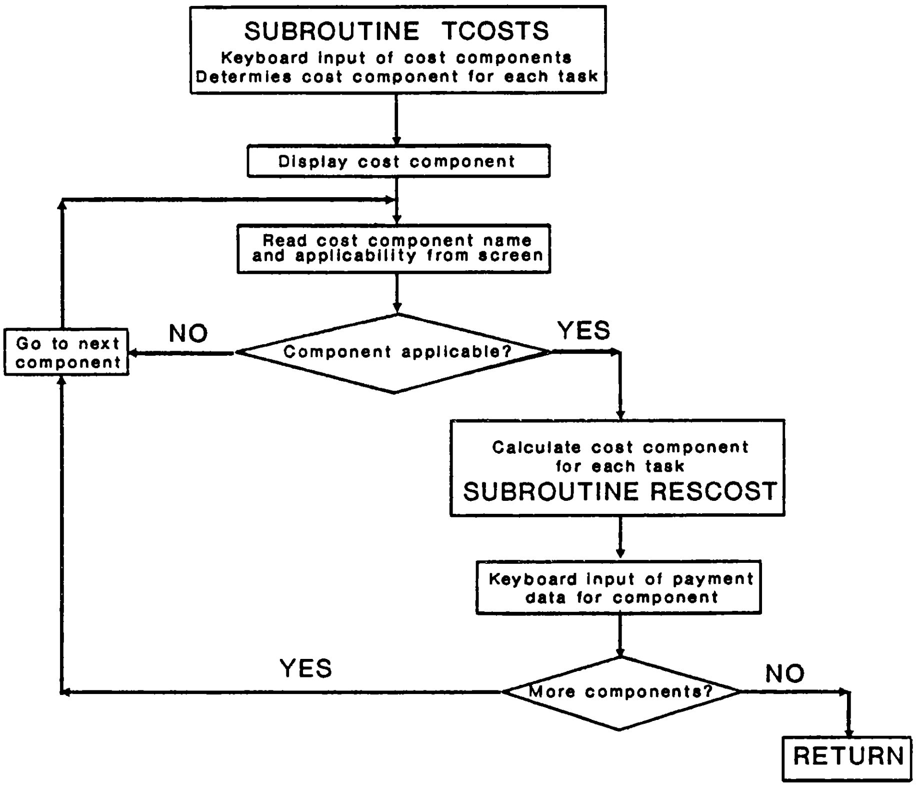


Figure 5.10 Subroutine TCOSTS Flow Chart.

5.2.6 Subroutine RESCOST

One of the noteworthy features of PERTMASTER is the use of a resource 'hierarchy' whereby any given resource may call upon a number of sub–resources which may, themselves, call upon sub–sub–resources and so on. The system is illustrated in **Figure 5.11** for the resource BFG (Backfill Gang) which consists of:

1	Bomag 35	(B35)	@ £75	per day;
2	JCB's	(JCB)	@ £150	per day;
4	Labourers	(LAB)	@ £46.5	per day;
2	7 Ton Tipper Trucks	(TIP)	@ £80	per day;
and each tipper truck make use of a driver		(DRV)	@ £50	per day.

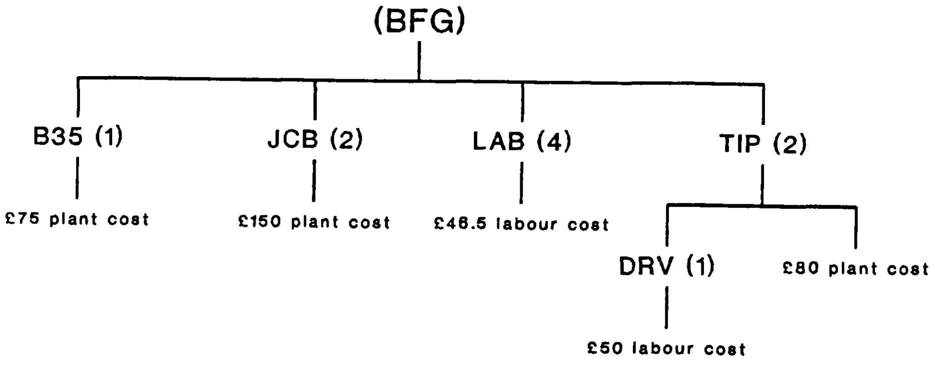


Figure 5.11 Resources Hierarchy.

The hierarchy of the resource BFG will appear in the export file in alphabetic order as follows:

```
R,B35,"Bomag 35",D,0,ALLDAYS
U,B35,XPC,75
     .    .    .    .    .
     .    .    .    .    .
R,BFG,"Backfill Gang",D,0,ALLDAYS
U,BFG,B35,1
U,BFG,JCB,2
U,BFG,LAB,4
U,BFG,TIP,2
     .    .    .    .    .
     .    .    .    .    .
R,DRV,"Driver",D,0,ALLDAYS
U,DRV,XLC,50
     .    .    .    .    .
     .    .    .    .    .
R,JCB,"JCB 3c",D,0,ALLDAYS
U,JCB,XPC,150
     .    .    .    .    .
     .    .    .    .    .
R,LAB,"Labourer",D,0,ALLDAYS
U,LAB,XLC,46.5
     .    .    .    .    .
     .    .    .    .    .
R,TIP,"7 Ton Tipper Truck",D,0,ALLDAYS
U,TIP,DRV,1
U,TIP,XPC,80
     .    .    .    .    .    .    .
     .    .    .    .    .    .    .
```

On the basis of the resource hierarchy, subroutine RESCOST determines the total quantities of a specified cost component used by each 'top- level' resource (where a top- level resourcc is one which is directly utilised by a task). It is then a straight forward matter, by considering each top- level resource used by a given task, to calculate the total quantity of the specified cost component associated with that task.

The operation of subroutine RESCOST is best explained by considering the operations involved in determining, say, the total labour cost component (XLC) associated with Backfill Gang (BFG).

The subresources array generated by subroutine ABSUBRES, corresponding to the export file data previously shown, will be as shown below.

I	RTOP(I)	RSUB(I)	RSUBQ(I)
.	.	.	.
.	.	.	.
.	.	.	.
7	B35	XPC	75
.	.	.	.
.	.	.	.
.	.	.	.
10	BFG	B35	1
11	BFG	JCB	2
12	BFG	LAB	4
13	BFG	TIP	2
.	.	.	.
.	.	.	.
.	.	.	.
17	DRV	XLC	50
.	.	.	.
.	.	.	.
.	.	.	.
25	JCB	XPC	150
.	.	.	.
.	.	.	.
.	.	.	.
28	LAB	XLC	46.5
.	.	.	.
.	.	.	.
.	.	.	.
33	TIP	DRV	1
34	TIP	XPC	80

Foe ease of understanding, the variables RTOP(I) and RSUB(I) have been shown as a containing resource codes as opposed to resource indices, although the latter will be the case in reality.

1. RESCOST identifies each of the above records where RSUB(I) = XLC. The arrays are then re–ordered such that these records are brought to the 'top' of the array, thus:

I	RTOP(I)	RSUB(I)	RSUBQ(I)
1	DRV	XLC	50
2	LAB	XLC	46.5
.	.	.	.
.	.	.	.
.	.	.	.
7	B35	XPC	75
.	.	.	.
.	.	.	.
.	.	.	.
10	BFG	B35	1
11	BFG	JCB	2
12	BFG	LAB	4
13	BFG	TIP	2
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
25	JCB	XPC	150
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
33	TIP	DRV	1
34	TIP	XPC	80

2. RESCOST searches the re–ordered array RSUB (starting from I = 3) for those resources (in this case LAB) in RTOP which have been identified as making direct use of XLC. The arrays are then re–ordered as shown.

I	RTOP(I)	RSUB(I)	RSUBQ(I)
1	DRV	XLC	50
2	LAB	XLC	46.5
3	TIP	DRV	1
4	BFG	LAB	4
.	.	.	.
.	.	.	.
.	.	.	.
7	B35	XPC	75
.	.	.	.
.	.	.	.
.	.	.	.
10	BFG	B35	1
11	BFG	JCB	2
13	BFG	TIP	2
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
25	JCB	XPC	150
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
34	TIP	XPC	80

3. Array RSUB is now searched for subresource which make use of DRV and LAB, namely TIP and BFG respectively. The arrays are sorted accordingly.

I	RTOP(I)	RSUB(I)	RSUBQ(I)
1	DRV	XLC	50
2	LAB	XLC	46.5
3	TIP	DRV	1
4	BFG	LAB	4
5	BFG	TIP	2
.	.	.	.
.	.	.	.

As can be seen, the process terminates when the searching of RSUB reveals no more relevant sub– resources.

4. The array variable RCOST(J) is initialised thus:

RCOST(RTOP) =1 for RTOP = XLC

RCOST(RTOP) =0 for RTOP # XLC

5. For I = 1,5, where I is the number of the re–ordered arrays, the following algorithm is used.

RCOST(RTOP(I)) = RCOST(RTOP(I)) + RCOST(RSUB(I)) x RSUBQ(I)

Thus, the sequence of calculations in this case is as follows:

I= 1 RCOST(DRV) = RCOST(DRV)+ RCOST(XLC) x RSUBQ(1)
= 0 + 1 x 50 = £50

I= 2 RCOST(LAB) = RCOST(LAB)+ RCOST(XLC) x RSUBQ(2)
= 0 + 1 x 46.5 = £46.5

I= 3 RCOST(TIP) = RCOST(TIP)+ RCOST(DRV) x RSUBQ(3)
= 0 + 50 x 1 = £50

I= 4 RCOST(BFG) = RCOST(BFG)+ RCOST(LAB) x RSUBQ(4)
= 0 + 46.5 x 4 = £186

I= 5 RCOST(BFG) = RCOST(BFG)+ RCOST(TIP) x RSUBQ(5)
= 186 + 50 x 2 = £286

Thus, the total labour cost component associated with the Backfill Gang = £286 per day.

Such calculation will be performed for the other cost components associated with BFG.

Continuing the example, the calculations of the total Daily Plant cost (XPC) associated with the same resource BFG will be performed as follows:

1. Rescost identifies each of the records where RSUB(I) = XPC and brought it to the top, thus the arrays will be as follows

I	RTOP(I)	RSUB(I)	RSUBQ(I)
1	B35	XPC	75
2	JCB	XPC	150
3	TIP	XPC	80
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
10	BFG	B35	1
11	BFG	JCB	2
12	BFG	LAB	4
13	BFG	TIP	2
.	.	.	.
.	.	.	.
.	.	.	.
17	DRV	XLC	50
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
28	LAB	XLC	46.5
.	.	.	.
.	.	.	.
.	.	.	.
33	TIP	DRV	1

2. RESCOST searches the re– ordered array RSUB (starting from I= 4) for the resources in RTOP which has been identified as making direct use of XPC (in this case B35, JCB and TIP). The arrays are re– ordered as shown:

I	RTOP(I)	RSUB(I)	RSUBQ(I)
1	B35	XPC	75
2	JCB	XPC	150
3	TIP	XPC	80
4	BFG	B35	1
5	BFG	JCB	2
6	BFG	TIP	2
.	.	.	.
.	.	.	.
12	BFG	LAB	4
.	.	.	.
.	.	.	.
.	.	.	.
17	DRV	XLC	50
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
28	LAB	XLC	46.5
.	.	.	.
.	.	.	.
.	.	.	.
33	TIP	DRV	1

3. Array RSUB is searched (starting from I= 7) for subresources which make direct use of B35, JCB and TIP namely BFG, which in this example will not be found and the process terminates when the search of RSUB revels no more relevant sub– resources.

4. The array variable RESCOST(J) is initialised thus:

RCOST(RTOP) = 1 for RTOP = XPC

RCOST(RTOP) = 0 for RTOP # XPC

5. For I = 1, 6 where I is the number of Swopped arrays, the same algorithm in the previous example is used. Thus the calculations in this case as follows:

I= 1 RCOST(B35) = RCOST(B35)+ RCOST(XPC) x RSUBQ(1)
= 0 + 1 x 75 = £75

I= 2 RCOST(JCB) = RCOST(JCB)+ RCOST(XPC) x RSUBQ(2)
= 0 + 1 x 150 = £150

I= 3 RCOST(TIP) = RCOST(TIP)+ RCOST(XPC) x RSUBQ(3)
= 0 + 1 x 80 = £50

I= 4 RCOST(BFG) = RCOST(BFG)+ RCOST(B35) x RSUBQ(4)
= 0 + 75 x 1 = £75

I= 5 RCOST(BFG) = RCOST(BFG)+ RCOST(JCB) x RSUBQ(5)
= 75 + 150 x 2 = £375

I= 6 RCOST(BFG) = RCOST(BFG)+ RCOST(TIP) x RSUBQ(6)
= 375 + 80 x 2 = £535

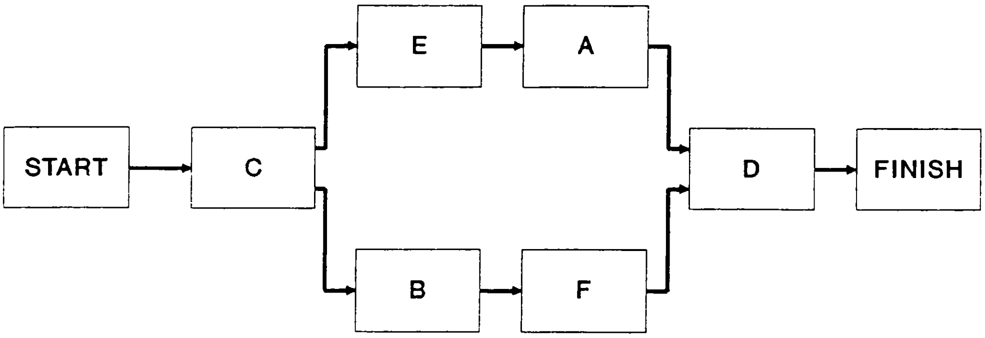
Thus the total Daily Plant cost component associated with the Backfill Gang is £535 per day.

Listing of the subroutine RESCOST is given in **Appendix C.15**.

5.2.7 Subroutine LSORT

Link data abstracted by the subroutine ABLINK are listed in the designated array in alphabetic order and, in order to perform the analysis calculations, this data requires to be arranged into an order suitable for carrying out the forward and backward passes through the network. Arranging the data into the above orders is carried out by the subroutine LSORT and the swopping subroutine LSWOP.

As with the subroutine RESCOST, the working of subroutine LSORT are best explained by the following example. The following network will be used for the explanation:



1. The variables of the link data abstracted by subroutine ABLINK from the export file will be stored in an alphabetic order as shown below:

LFROM(I,1)	LTO(I,1)	LTYPE(I,1)	LDURN(I,1)
A	D	F−S	0
B	F	F−S	0
C	B	F−S	0
C	E	F−S	0
D	FINISH	F−S	0
E	A	F−S	0
F	D	F−S	0
START	C	F−S	0

For ease of understanding, the variables LFROM, LTO and LTYPE have been shown as containing alphanumeric names rather than indices, although the latter will be the case in reality (see section 5.2.2.3).

2. To re−order the link data for the Forward Pass, subroutine LSORT identifies the records where the variable LFROM(I,1) is the first task in the network (having SFCODE=0), using the subroutine LSWOP the arrays are re−ordered such that these records are brought to the 'top', thus:

LFROM(I,1)	LTO(I,1)	LTYPE(I,1)	LDURN(I,1)
START	C	F-S	0
A	D	F-S	0
B	F	F-S	0
C	B	F-S	0
C	E	F-S	0
D	FINISH	F-S	0
E	A	F-S	0
F	D	F-S	0

3. Lsort identifies the relevant target task (in this case: C) and searches the array LFROM(I,1) for all instances where this task appears. As each instance is identified, the relevant link record is brought to the 'top' of the array thus:

LFROM(I,1)	LTO(I,1)	LTYPE(I,1)	LDURN(I,1)
START	C	F-S	0
C	B	F-S	0
C	E	F-S	0
A	D	F-S	0
B	F	F-S	0
D	FINISH	F-S	0
E	A	F-S	0
F	D	F-S	0

4. The process is repeated for each of the target tasks appropriate to c – and so on. The final result is as shown below:

LFROM(I,1)	LTO(I,1)	LTYPE(I,1)	LDURN(I,1)
START	C	F–S	0
C	B	F–S	0
C	E	F–S	0
B	F	F–S	0
E	A	F–S	0
F	D	F–S	0
A	D	F–S	0
D	FINISH	F–S	0

5. The procedure of sorting the (I,2) arrays into 'Backward Pass' order is the exact reverse of that for the 'Forward Pass' order but starting with the last task in the network (SFCODE=2). The final result is as shown below:

LFROM(I,2)	LTO(I,2)	LTYPE(I,2)	LDURN(I,2)
D	FINISH	F–S	0
F	D	F–S	0
A	D	F–S	0
B	F	F–S	0
E	A	F–S	0
C	B	F–S	0
C	E	F–S	0
START	C	F–S	0

The subroutine LSORT listing is given in **Appendix C.17**. and listing of the subroutine LSWOP is given in **Appendix C.18**.

5.2.8 Subroutine QUIT

This subroutine writes data to the following files for later subsequent use by PERTRA2 and PERTRA3:

1. FLNAME.GEN, general data file. See **Appendix B.1**.
2. FLNAME.TSK, tasks data file. See **Appendix B.2**.
3. FLNAME.LNK, link data file. See **Appendix B.3**.
4. FLNAME.CST, cost data file. See **Appendix B.4**.

Following a screen display to that effect the subroutine terminate the operation of PERTRA1. Listing of the subroutine is given in **Appendix C.14**.

5.3 PERTRA2: Interactive Input of Task Duration Data

For the purpose of carrying out the Risk Analysis, the various tasks involved in the project may be categorised as follows:

(a) Tasks with VARIABLE durations:

These are those tasks whose durations are variable in accordance with a probability distribution defined by:

1. 'Optimistic duration': 99% exceedance probability.
2. 'Most likely duration': used in PMA calculations.
3. 'Pessimistic duration': 1% exceedance probability.

(b) Tasks with INVARIABLE durations:

These are those tasks whose durations are invariable and not subject to the probability distribution mentioned above. Milestone and Hammock tasks, by their nature, are both included automatically in this category.

The purpose of PERTRA2, therefore, is to update the information contained in FLNAME.TSK; to include for:

(a) a code which identifies each task as having either a 'VARIABLE' or 'INVARIABLE' duration;

(b) the 'optimistic' and 'pessimistic' durations of each 'VARIABLE' duration task.

Since the purpose of PERTRA2 is essentially the same as that of PERTRA1, namely: the initialisation of data for use by PERTRA3, the operations performed thereby should logically have been included as part and parcel of PERTRA1.

It was felt, however, that such an arrangement would introduce an unnecessary element of rigidity in that it would require the relevant task duration data to be entered, in its entirety, in a single session. Additionally, it would preclude the updating of the task duration data , following an analysis using one particular set of values for 'optimistic' and 'pessimistic' durations, to facilitate a subsequent analysis on the basis of a different set of values thereof.

Consequently, PERTRA2 was designed as a separate module which could be used at any stage in the Risk Analysis process.

A flow chart of the PERTRA2 is shown in **Figure 5.12** and a listing of the program is given in **Appendix D.1**.

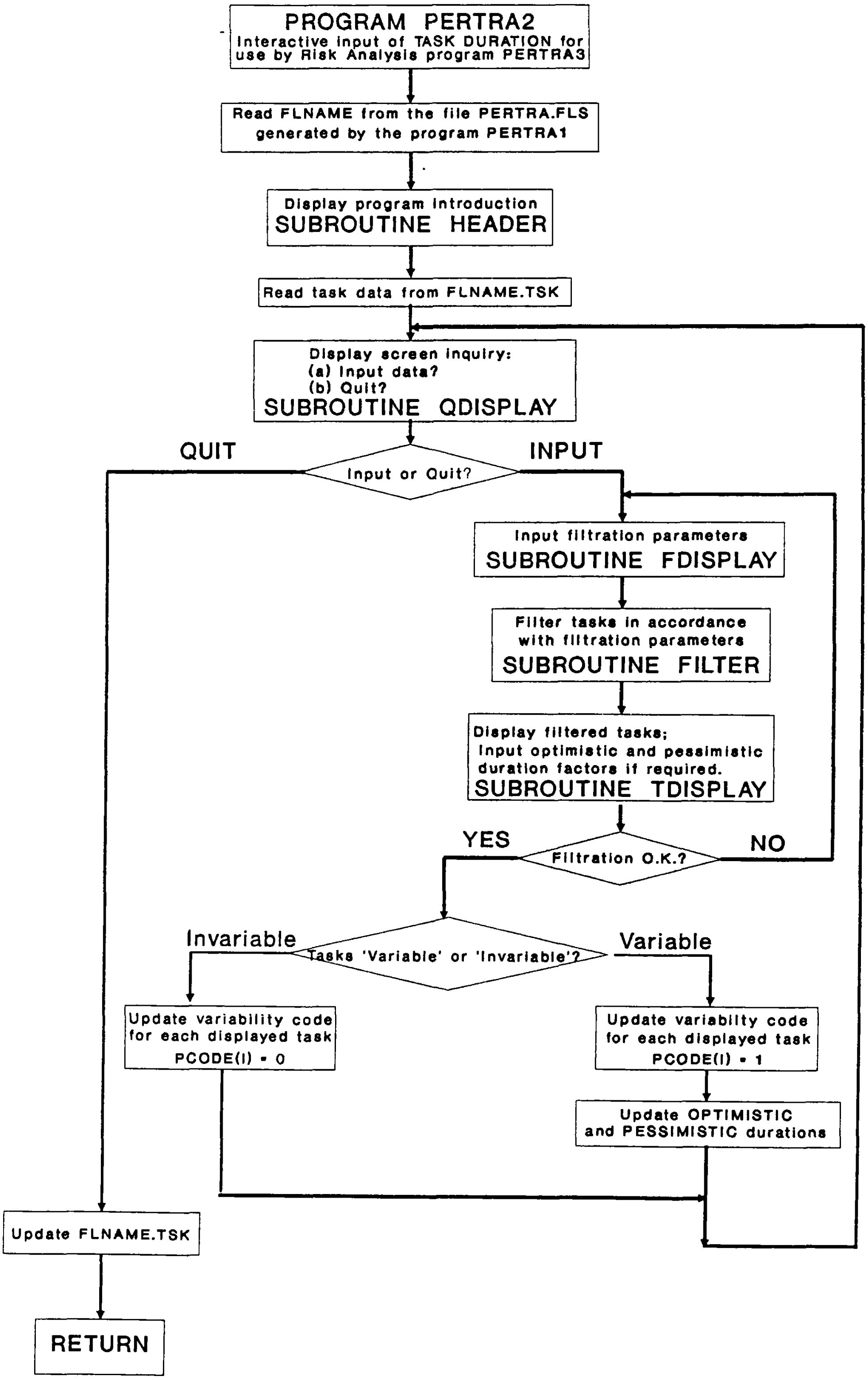

PROGRAM PERTRA2
Interactive input of TASK DURATION for use by Risk Analysis program PERTRA3
Read FLNAME from the file PERTRA.FLS generated by the program PERTRA1
Display program introduction
SUBROUTINE HEADER
Read task data from FLNAME.TSK
Display screen inquiry:
(a) Input data?
(b) Quit?
SUBROUTINE QDISPLAY
QUIT
INPUT
Input or Quit?
Input filtration parameters
SUBROUTINE FDISPLAY
Filter tasks in accordance with filtration parameters
SUBROUTINE FILTER
Display filtered tasks;
Input optimistic and pessimistic duration factors if required.
SUBROUTINE TDISPLAY
YES
Filtration O.K.?
NO
Invariable
Tasks 'Variable' or 'Invariable'?
Variable
Update variability code for each displayed task
PCODE(I) = 0
Update variabilty code for each displayed task
PCODE(I) = 1
Update OPTIMISTIC and PESSIMISTIC durations
Update FLNAME.TSK
RETURN

Figure 5.12 Program PERTRA2 Flow Chart

The main program routines, subroutines and functions are now discussed:

5.3.1 Subroutine HEADER

The purpose of this subroutine is to generate an introductory screen display which provides general information pertaining to the use of PERTRA2.

A flow chart of subroutine HEADER is shown in **Figure 5.13** and a listing of the subroutine is given in **Appendix D.2.**

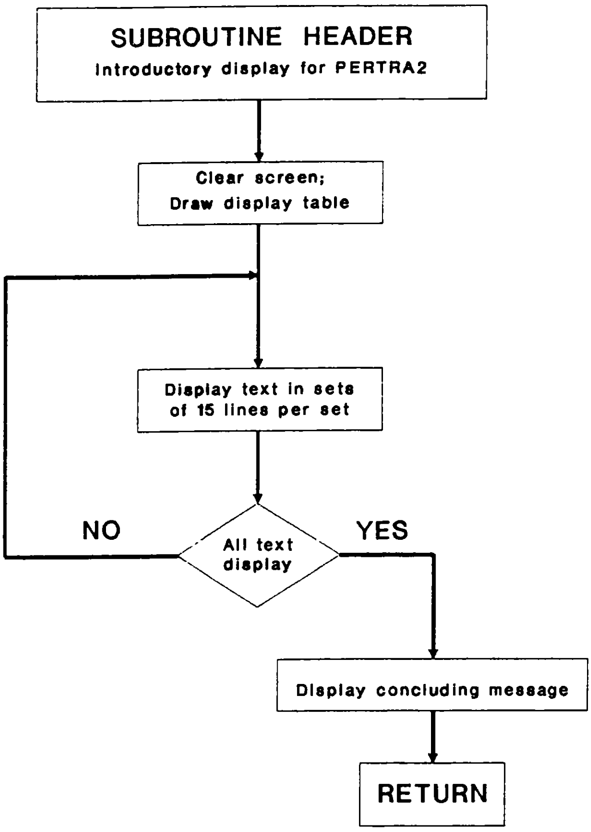

SUBROUTINE HEADER
Introductory display for PERTRA2
Clear screen;
Draw display table
Display text in sets of 15 lines per set
NO
All text display
YES
Display concluding message
RETURN

Figure 5.13 Subroutine HEADER Flow Chart.

5.3.2 Subroutines FDISPLAY, FILTER, TDISPLAY

The input of task duration date (i.e. 'optimistic' and 'pessimistic' durations) can be a somewhat tedious and time consuming exercise if carried out on a 'task by task' basis.

The time spent on the exercise, however, can be much reduced by taking advantages of a particular feature of many construction projects, namely that such projects contain groups of tasks which are identical in both nature and duration. Thus being the case, it may reasonably be inferred that the variability of the duration of the tasks within each such group is identical. If a given group may be identified, therefore, only one set of task duration parameters need to be input from the keyboard and will apply to each task within that group.

The above subroutines provide a facility whereby the overall task list may be 'filtered' to display selected tasks, with the subsequent input of task duration data applying to <u>each</u> of the displayed tasks and <u>only</u> to those tasks.

5.3.2.1 Subroutine FDISPLAY

This subroutine enables the keyboard input of up to six levels of the following four parameters:

CODE (Name, Type or Work Breakdown Structure of the tasks).

POSITION (The position in the CODE which defines the 'start' of the

SELECTION parameter)

OPERAND ('Equal to' or 'Not Equal' to)

SELECTION (A maximum of 9 characters on which selection will be based).

For example, the filtration parameters:

CODE	POSITION	OPERAND	SELECTION
NAME	3	EQ	CONC

will convey an instruction to PERTRA2 to display only those tasks having names which contain the alphanumeric string 'CONC' starting at position 3 therein.

Where more than one level of parameters is specified, FDISPLAY allows the input of a further parameter (AND or OR) which will determine the filtration logic.

If OR is selected, only one of the filtration criteria need be satisfied for a task to be displayed. If AND is selected, all of the filtration criteria require to be satisfied for a task to be displayed.

Amplifying the previous example, the filtration parameters:

CODE	POSITION	OPERAND	SELECTION
NAME	3	EQ	CONC
TYPE	1	NE	H

LOGIC: AND

will result in the display of only the 'non-hammock' tasks whose names contain the alphanumeric string 'CONC' at position 3.

The subroutine flow chart is shown in **Figure 5.14.** Listing of the subroutine is given in **Appendix** D.4.

5.3.2.2 Subroutine FILTER

This subroutine carries out the actual filtration of tasks according to the parameters input via subroutine FDISPLAY. A listing of the subroutine is given in **Appendix** D.5.

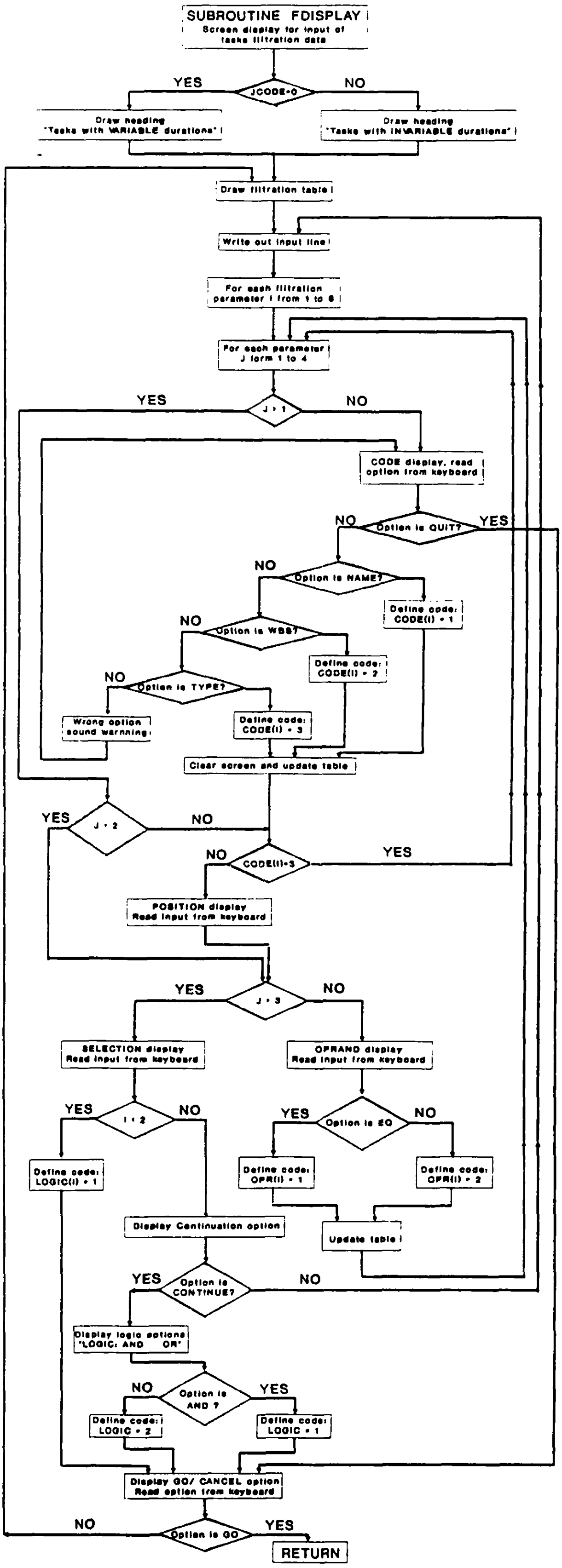
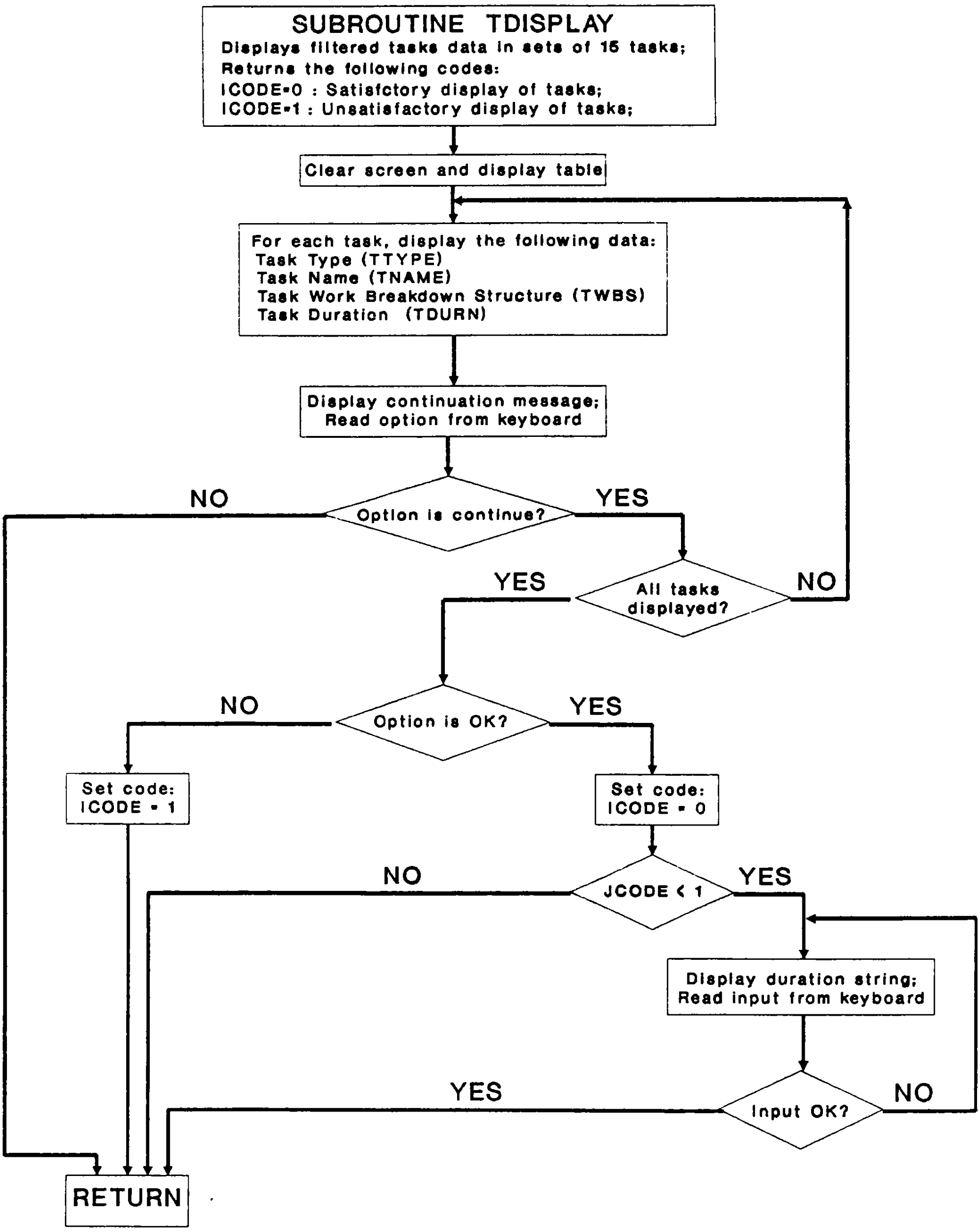

SUBROUTINE FDISPLAY
Screen display for input of tasks filtration data
YES
JCODE=0
NO
Draw heading "Tasks with VARIABLE durations"
Draw heading "Tasks with INVARIABLE durations"
Draw filtration table
Write out input line
For each filtration parameter I from 1 to 6
For each parameter J form 1 to 4
YES
J > 1
NO
CODE display, read option from keyboard
NO
Option is QUIT?
YES
NO
Option is NAME?
Define code: CODE(I) = 1
NO
Option is WBS?
Define code: CODE(I) = 2
NO
Option is TYPE?
Wrong option sound warnning
Define code: CODE(I) = 3
Clear screen and update table
YES
J > 2
NO
NO
CODE(I)=3
YES
POSITION display Read input from keyboard
YES
J > 3
NO
SELECTION display Read input from keyboard
OPRAND display Read input from keyboard
YES
I < 2
NO
YES
Option is EQ
NO
Define code: LOGIC(I) = 1
Define code: OPR(I) = 1
Define code: OPR(I) = 2
Display Continuation option
Update table
YES
Option is CONTINUE?
NO
Display logic options "LOGIC: AND OR"
NO
Option is AND ?
YES
Define code: LOGIC = 2
Define code: LOGIC = 1
Display GO/ CANCEL option Read option from keyboard
NO
Option is GO
YES
RETURN

Figure 5.14 Subroutine FDISPLAY Flow Chart.

5.3.2.3 Subroutine TDISPLAY

This subroutine scrolls the selected tasks to enable the user to check whether the required filtration has been successfully carried out. A flow chart of the subroutine is shown in **Figure 5.15** and a listing of the subroutine is given in **Appendix D.6.**



SUBROUTINE TDISPLAY
Displays filtered tasks data in sets of 15 tasks;
Returns the following codes:
ICODE=0 : Satisfctory display of tasks;
ICODE=1 : Unsatisfactory display of tasks;
Clear screen and display table
For each task, display the following data:
Task Type (TTYPE)
Task Name (TNAME)
Task Work Breakdown Structure (TWBS)
Task Duration (TDURN)
Display continuation message;
Read option from keyboard
NO
Option is continue?
YES
YES
All tasks displayed?
NO
NO
Option is OK?
YES
Set code: ICODE = 1
Set code: ICODE = 0
NO
JCODE < 1
YES
Display duration string;
Read input from keyboard
YES
Input OK?
NO
RETURN

Figure 5.15 Subroutine TDISPLAY Flow Chart

5.3.3 Subroutine QDISPLAY

This subroutine performs a double functions, namely:

(a) To generate the 'input of task duration data' screen display appropriate to the 'variability' (i.e. 'Variable' or 'Invariable') of the tasks displayed by TDISPLAY.

(b) To generate a 'quitting' screen display if the user desires to quit PERTRA2 following the input of task duration data.

The subroutine flow chart is shown in **Figure 5.14** and a listing of the subroutine is given in **Appendix D.3.**

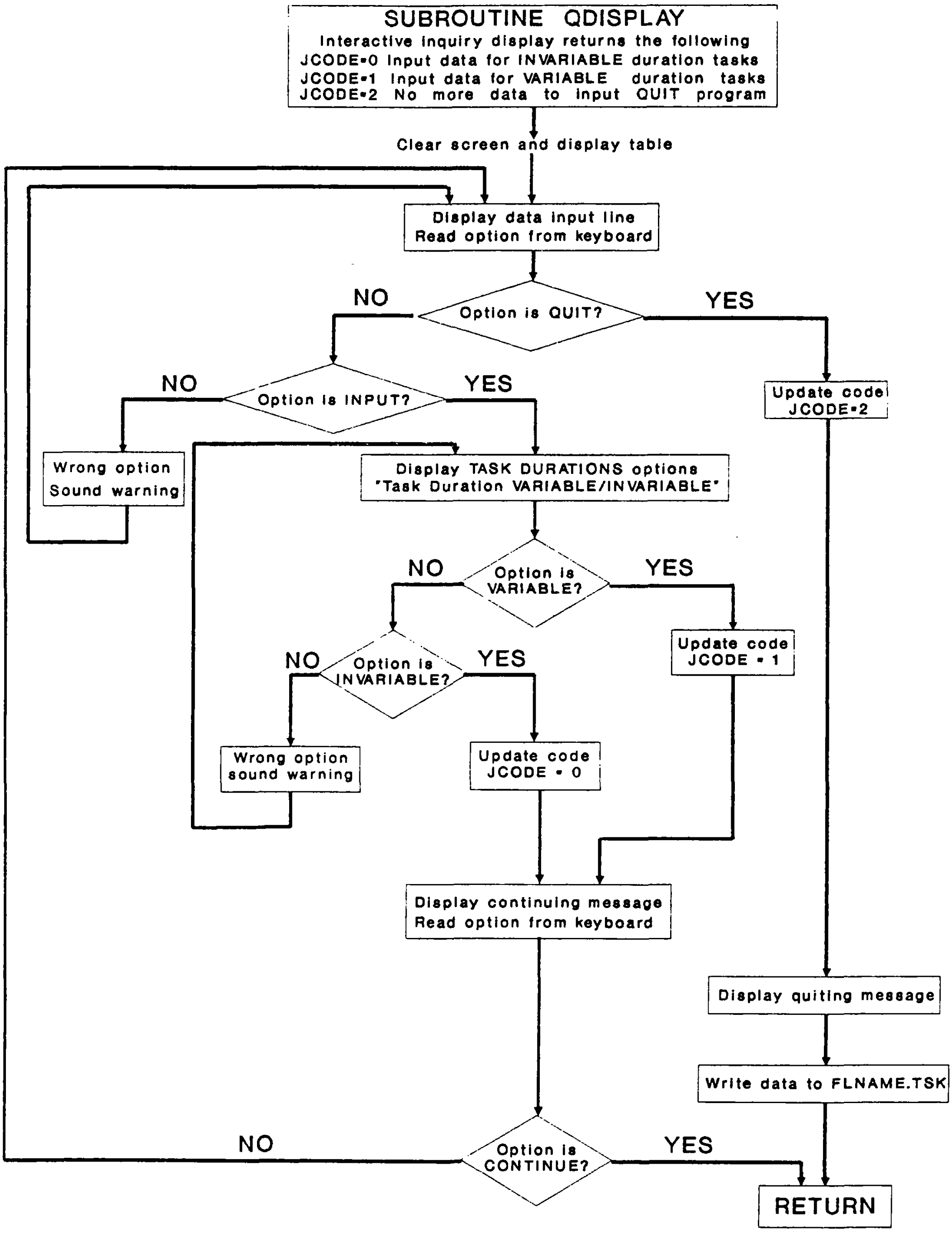

SUBROUTINE QDISPLAY
Interactive inquiry display returns the following
JCODE•0 Input data for INVARIABLE duration tasks
JCODE•1 Input data for VARIABLE duration tasks
JCODE•2 No more data to input QUIT program
Clear screen and display table
Display data input line
Read option from keyboard
NO
Option is QUIT?
YES
NO
Option is INPUT?
YES
Update code
JCODE•2
Wrong option
Sound warning
Display TASK DURATIONS options
'Task Duration VARIABLE/INVARIABLE'
NO
Option is VARIABLE?
YES
NO
Option is INVARIABLE?
YES
Update code
JCODE • 1
Wrong option
sound warning
Update code
JCODE • 0
Display continuing message
Read option from keyboard
Display quiting message
Write data to FLNAME.TSK
NO
Option is CONTINUE?
YES
RETURN

Figure 5.14 Subroutine QDISPLAY Flow Chart.

5.3.4 Input of Task Duration Data.

Task duration data, for VARIABLE duration tasks, is input in the form of factors by which the 'Most Likely Duration' of task or group of tasks, may be multiplied to obtain the 'Optimistic' and 'Pessimistic' durations thereof.

(a) 'Optimistic duration' factor: ranges from 0.00 to 1.00;

(b) 'pessimistic duration' factor: ranges from 1.00 to 1000.00.

The input of task durations using factors have been chosen because of the following reasons:

(a) In reality 'Optimistic' and 'Pessimistic' durations are usually estimated and expressed as a percentage of the 'Most likely' duration (i.e. half, twice etc.).

(b) Allows the input of task durations in groups which is more efficient and less time consuming

5.3.5 PERTRA2 Output.

By running the program PERTRA2 the file FLNAME.TSK will be updated to include the following data for each task:

VARIABILITY CODE (=0 for INVARIABLE; =1 for VARIABLE).

OPTIMISTIC DURATION (In Days).

PESSIMISTIC DURATION (In Days)

An Example of the Updated file is given in **Appendix E.**

5.4 PERTRA3: Risk Analysis and Cash Flow Calculations.

Risk analysis (Time and Cost) using the Monte Carlo simulation technique is the reason behind the development of PERTRA suite of programs of which PERTRA3 is the program which carries out the analysis. The essence of PERTRA3 is as follows:

1. Reading of the data from the files generated by PERTRA1 and PERTRA2.

2. Performance of the following:

   (a) Time analysis of the project network using the Most Likely Durations to calculate the project duration.

   (b) Cost analysis to determine the project IRR and CAPTIM.

3. Determination of the Duration distribution for each task with a VARIABLE duration.

4. Performance of the following for each simulation:

   (a) Random selection of the duration for each task based on the Triangular Probability distribution (see **Section 4.5.2**).

   (b) Performance of the Time Analysis of the project network (based on the selected durations) to determine the project duration.

(c) Performance of the Cost Analysis to determine IRR and CAPTIM.

5. Sorting of project durations, IRR and CAPTIM into rank order and performance of statistical analysis thereon.

6. Writing of data to file PERTRA3.RES.

A flow chart and listing of PERTRA3 is given in **Figure 5.16** and **Appendix F.1**. respectively.

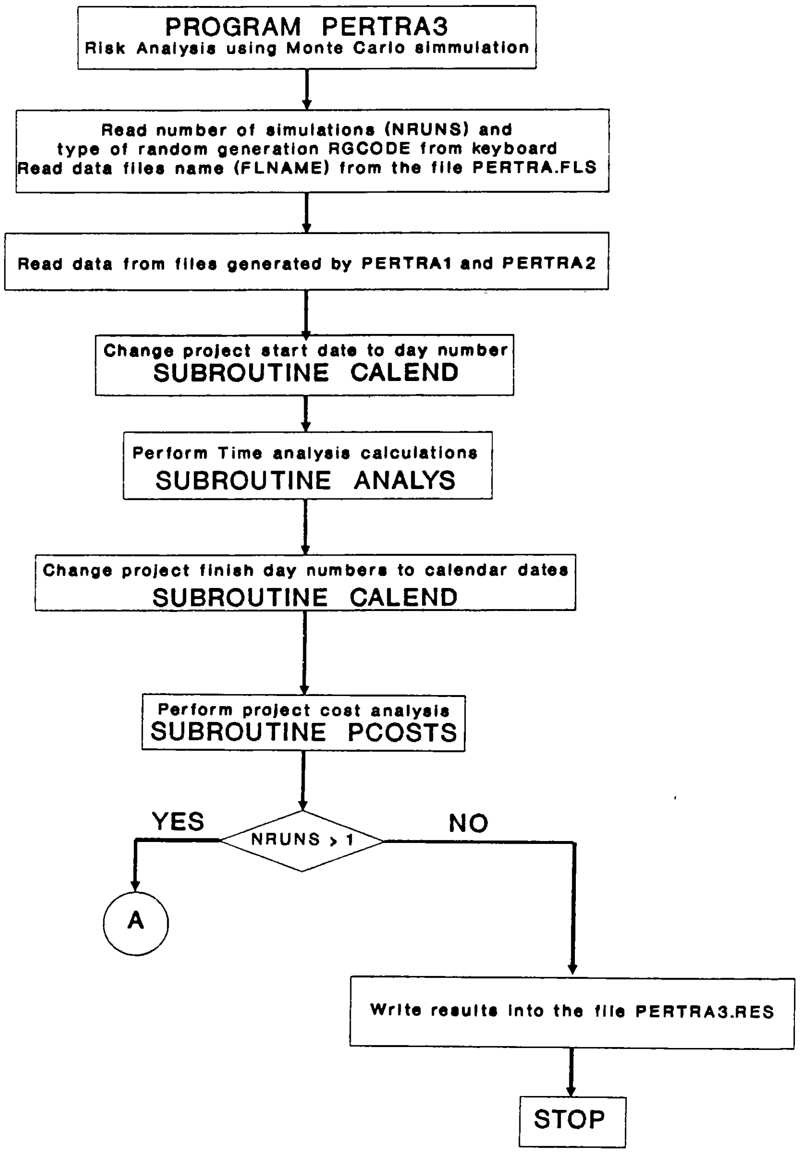

PROGRAM PERTRA3
Risk Analysis using Monte Carlo simmulation
Read number of simulations (NRUNS) and type of random generation RGCODE from keyboard
Read data files name (FLNAME) from the file PERTRA.FLS
Read data from files generated by PERTRA1 and PERTRA2
Change project start date to day number
SUBROUTINE CALEND
Perform Time analysis calculations
SUBROUTINE ANALYS
Change project finish day numbers to calendar dates
SUBROUTINE CALEND
Perform project cost analysis
SUBROUTINE PCOSTS
YES
NRUNS > 1
NO
A
Write results into the file PERTRA3.RES
STOP

Figure 5.16 Program PERTRA3 Flow Chart.

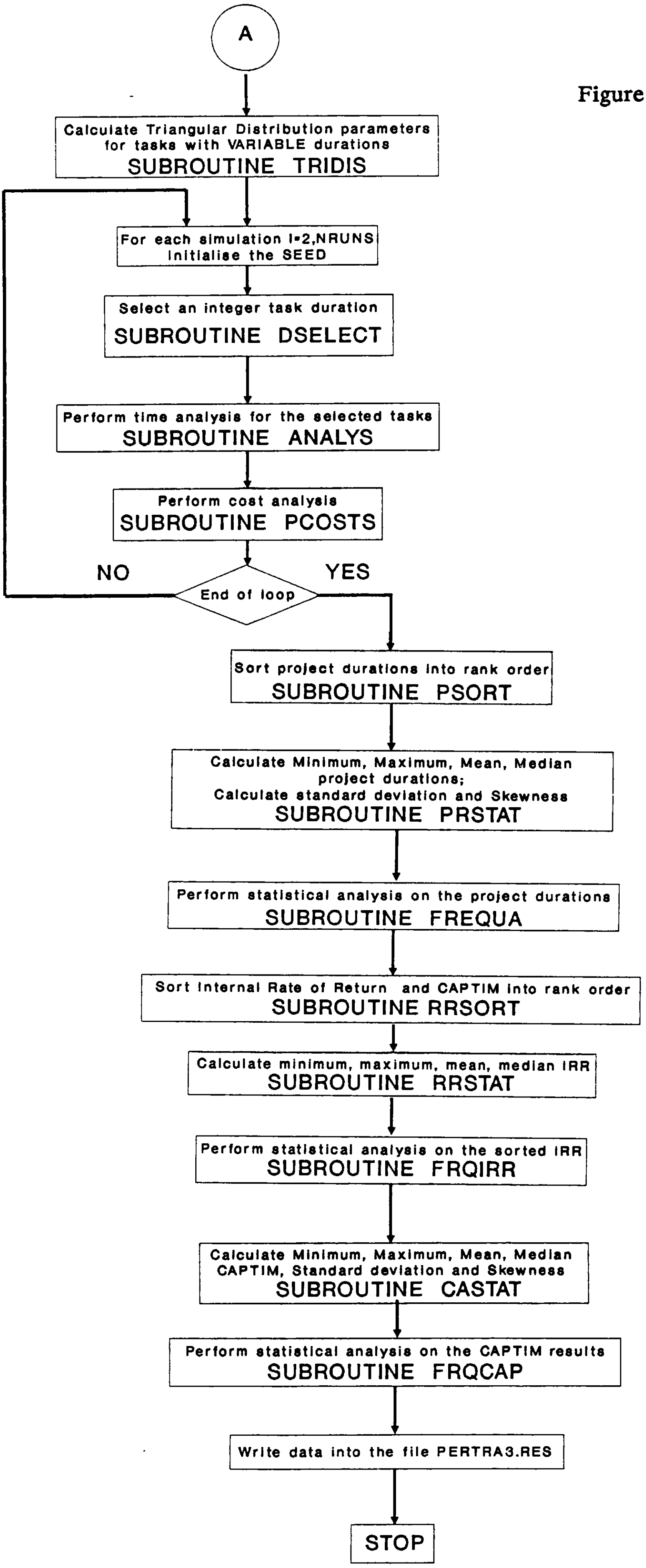

A
Calculate Triangular Distribution parameters for tasks with VARIABLE durations
SUBROUTINE TRIDIS
For each simulation I=2,NRUNS Initialise the SEED
Select an integer task duration
SUBROUTINE DSELECT
Perform time analysis for the selected tasks
SUBROUTINE ANALYS
Perform cost analysis
SUBROUTINE PCOSTS
NO
YES
End of loop
Sort project durations into rank order
SUBROUTINE PSORT
Calculate Minimum, Maximum, Mean, Median project durations;
Calculate standard deviation and Skewness
SUBROUTINE PRSTAT
Perform statistical analysis on the project durations
SUBROUTINE FREQUA
Sort Internal Rate of Return and CAPTIM into rank order
SUBROUTINE RRSORT
Calculate minimum, maximum, mean, median IRR
SUBROUTINE RRSTAT
Perform statistical analysis on the sorted IRR
SUBROUTINE FRQIRR
Calculate Minimum, Maximum, Mean, Median CAPTIM, Standard deviation and Skewness
SUBROUTINE CASTAT
Perform statistical analysis on the CAPTIM results
SUBROUTINE FRQCAP
Write data into the file PERTRA3.RES
STOP

Figure 5.16 (Continue)

5.4.1 Calendar Subroutines and Functions

In order to perform the time analysis of the project network, and to facilitate the performance of cash flow calculations, the following subroutines and functions have been included:

5.4.1.1 Subroutine CALEND

This subroutine converts an alphanumeric date (e.g. 05/DEC/92 – as generated by PERTMASTER) into an integer day number (relating to the 'Julian' day number commonly used in astronomy) and vice–versa, as follows:

(a) Conversion of date to day number:

(i) **Conversion of alphanumeric date to numeric date ID,IM,IY:**

for example: 05/DEC/92 ⇒ ID = 5; IM = 12; IY = 1992

(ii) Conversion of numeric date to day number using the following algorithm:

$$\text{Day Number} = \text{INT}(365.25 \times Y) + \text{INT}(30.6001 \times M) + D$$

where:

$$\begin{aligned} Y &= IY - 1 && \text{if } IM = 1 \text{ or } 2 \\ &= IY && \text{if } IM > 2 \end{aligned}$$

$$\begin{aligned} M &= IM + 13 && \text{if } IM = 1 \text{ or } 2 \\ &= IM + 1 && \text{if } IM > 2 \end{aligned}$$

(b) Conversion of day number to alphanumeric date:

(i) Conversion of day number to numeric date, ID, IM, IY, using the following algorithm:

$$Y = \mathrm{INT}\left[\frac{\text{Day Number} - 122.1}{365.25}\right]$$

$$M = \mathrm{INT}\left[\frac{\text{Day Number} - \mathrm{INT}(365.25 \text{ x } Y)}{30.6001}\right]$$

$$D = \text{Day Number} - \mathrm{INT}(365.25 \text{ x } Y) - \mathrm{INT}(30.6001 \text{ x } M)$$

where:

$$\begin{aligned} IM &= M - 13 && \text{If } M = 14 \text{ , } 15 \\ &= M - 1 && \text{If } M < 14 \end{aligned}$$

$$\begin{aligned} IY &= Y && \text{If } IM > 2 \\ &= Y + 1 && \text{If } IM = 1{,}2 \end{aligned}$$

$$ID = D$$

(ii) Conversion of numeric date to alphanumeric date. For example: 02/09/1992 to 02/SEP/1992

Listing of the subroutine is given in **Appendix F.2**.

5.4.1.2 Function IDOTW

This function calculates the day of the week from the day number using the following algorithm:

Day of the Week = 7 x MOD [(Day Number + 5),7]

The day of the week is represented by integers 0 to 6 where 0 represents Sunday. For ease of calculation the function changes Sunday from 0 to 7. The listing of the function is given in **Appendix F.3.**

5.4.2 Subroutine ANALYS

This subroutine performs the time analysis of the network and calculates the following for each task:

(a) Earliest start day number;

(b) Earliest finish day number;

(c) Latest start day number;

(d) Latest finish day number;

It also calculates Hammock tasks durations and identifies critical tasks in the network.

Calculation of the above task schedules is performed in accordance with the following algorithms:

(a) Calculation of Earliest Start day number (ES):

1. If the link type is F– S;

ES(Succeeding task)= EF(preceding task)+ Link Duration

2. If the link type is S– S;

ES(succeeding task)= ES(preceding task)+ Link Duration

(b) Calculation of Earliest Finish day number (EF):

1. If the link type is F– F;

EF(succeeding task)= EF(preceding task)+ Link Duration

2. If the link type is S– S or F– S;

EF(succeeding task)= ES(succeeding task)+ Duration of succeeding task

(c) Calculation of Latest Start day numbers (LS):

1. If the link type is S– S;

LS(preceding task)= LS(succeeding task)– Link Duration

2. If the link type is F– F or F– S;

LS(preceding task)= LS(succeeding task)– Duration of preceding task

(d) Calculation of Latest Finish day numbers (LF)

1. If the link type is F– F;

LF(preceding task)= LF(succeeding task)– Link Duration

2. If the link type is F– S;

LF(preceding task)= LS(succeeding task)– Link Duration

It should be noted that, since the task and link durations are specified in 'working' days, use of the above algorithms will result in the schedule dates being in the form of 'working' day numbers – which are of strictly limited use schedule– wise where a task's 'working' week differs from the standard 'calendar' week of 7 days.

To ensure that the above dates are in the form of 'actual' day numbers (as opposed to 'working' day numbers), the following procedures and algorithms have been included in the subroutine:

(A) Forward Pass:

Tasks and/or link durations are adjusted to include for 'down time' by adding the number of non—working days (TLADD) given by the following algorithm:

$$\text{TLADD} = \text{INT}\left[\frac{(\text{DURN} + \text{IESDOW} - 2)}{\text{TWK}}\right](7 - \text{TWK})$$

where;

DURN = Task or Link duration;

TWK = Task working days/week;

IESDOW = Link or Task Earliest start day of the week

Subsequently—calculated Earliest Start/Finish is adjusted to ensure that it will fall on a working day by adding the integer ISADD calculated as follows:

If $\quad \text{IESDOW} > \text{TWK};$

$$\text{ISADD} = 8 - \text{IESDOW}$$

(B) Backward Pass:

The task and/or link duration (as the case may be) is adjusted to include for 'down time' by adding the integer TLADD calculated by the following algorithm:

$$\text{TLADD} = \text{INT}\left[\frac{(\text{DURN}-\text{LFDOW}+\text{TWK}-1)}{\text{TWK}}\right](7-\text{TWK})$$

where;

DURN = Task or Link duration;

LFDOW = Task or Link Latest Finish day of the week;

TWK = Task working days/week

Calculated Latest Finish/Start should be adjusted to fall in a working day by subtracting the integer ISDDCT calculated as follows:

If $\quad \text{LFDOW} > \text{TWK};$

$$\text{ISDDCT} = \text{LFDOW} - \text{TWK}$$

NOTE:

Using of the above algorithms is varying in accordance with the type of link and the calculated schedule date, this variation can be seen in the listing of the subroutine ANALYS given in **Appendix F.4**.

The subroutine flow chart is shown in **Figure 5.17**.

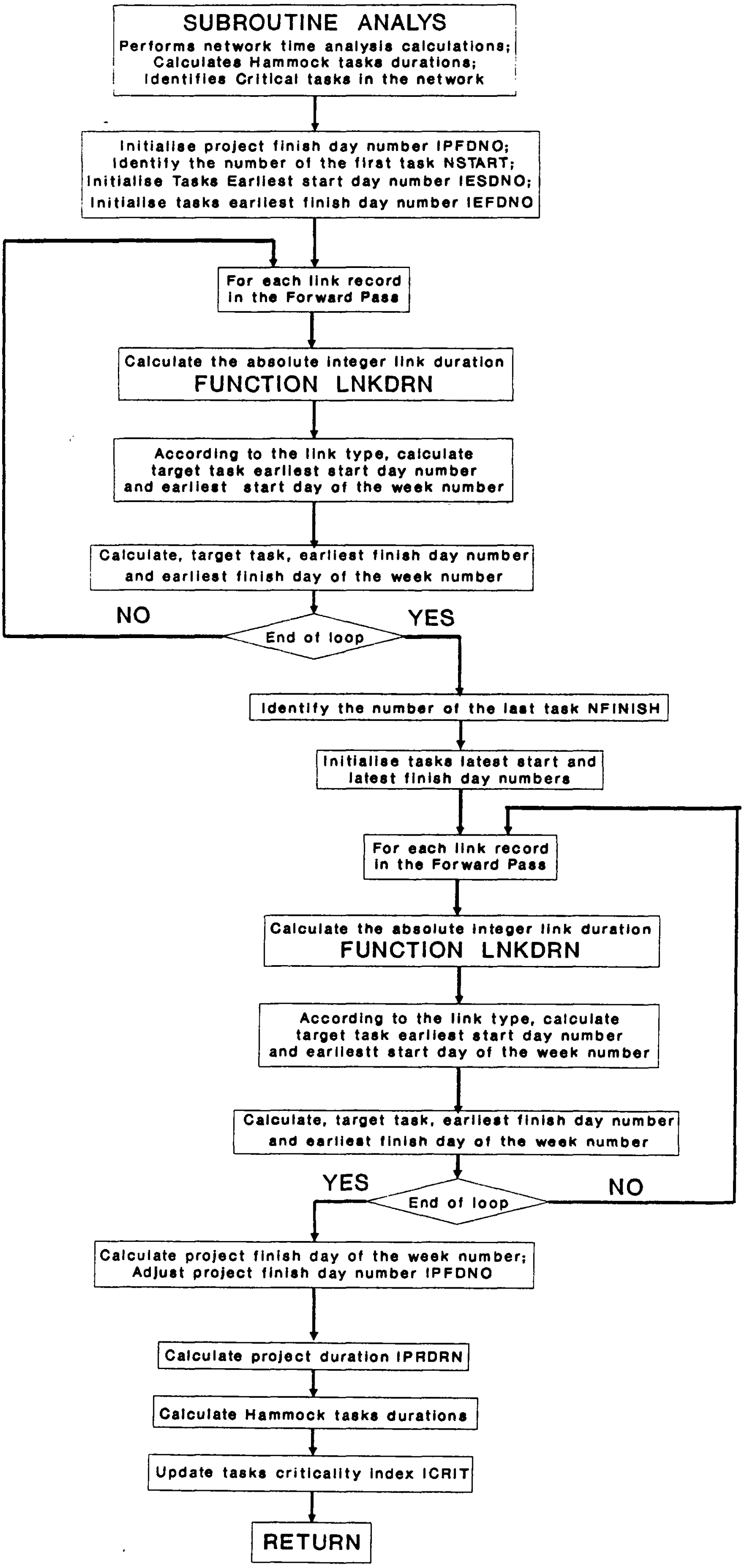

SUBROUTINE ANALYS
Performs network time analysis calculations; Calculates Hammock tasks durations; Identifies Critical tasks in the network
Initialise project finish day number IPFDNO; Identify the number of the first task NSTART; Initialise Tasks Earliest start day number IESDNO; Initialise tasks earliest finish day number IEFDNO
For each link record in the Forward Pass
Calculate the absolute integer link duration FUNCTION LNKDRN
According to the link type, calculate target task earliest start day number and earliest start day of the week number
Calculate, target task, earliest finish day number and earliest finish day of the week number
NO
YES
End of loop
Identify the number of the last task NFINISH
Initialise tasks latest start and latest finish day numbers
For each link record in the Forward Pass
Calculate the absolute integer link duration FUNCTION LNKDRN
According to the link type, calculate target task earliest start day number and earliestt start day of the week number
Calculate, target task, earliest finish day number and earliest finish day of the week number
YES
End of loop
NO
Calculate project finish day of the week number; Adjust project finish day number IPFDNO
Calculate project duration IPRDRN
Calculate Hammock tasks durations
Update tasks criticality index ICRIT
RETURN

Figure 5.17 Subroutine ANALYS Flow Chart

5.4.2.1 Function LINDRN

Since the link duration in the case of S– S link type is given always by a percentage of the preceding task, this function is called by the subroutine ANALYS to change the (S–S) link type duration from a percentage to an absolute Integer duration. It also changes all other type of link duration to Integer durations. Listing of the function is given in **Appendix F.5**.

5.4.2.2 Function ISADD

This function is called by the subroutine ANALYS to calculate the number of non– working days, week– ends, to be added to the task Earliest start day number in order to prevent it from falling in a non– working day. Listing of the function is given in **Appendix F.6**.

5.4.2.3 Function ISDEDC

It is called by the subroutine ANALYS to calculate the number of non– working days to be deducted in order to prevent the calculated Latest Finish day number of falling in a non– working day. Listing of the function is given in **Appendix** F.7..

5.4.3 Subroutine TRIDIS

Calculates the triangular distribution parameters for any given task with VARIABLE duration. It performs the calculation in accordance with the algorithms (1), (2), (3) in Section 4.4.1. The subroutine flow chart is shown in **Figure 5.18**. Listing of the subroutine is given in **Appendix F.8**.

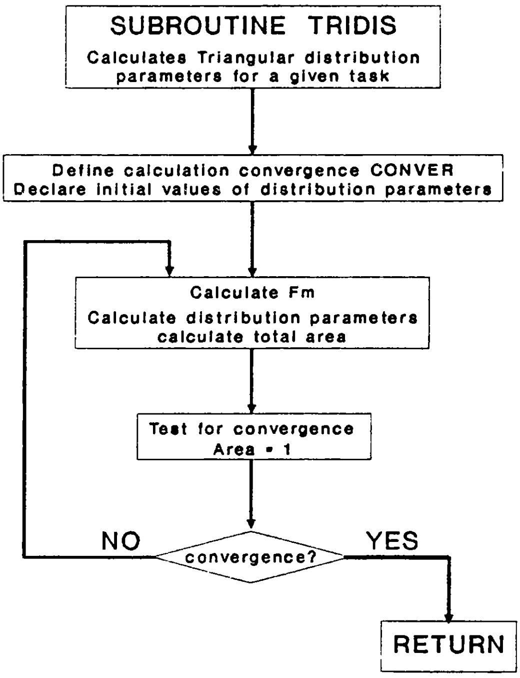

SUBROUTINE TRIDIS
Calculates Triangular distribution parameters for a given task
Define calculation convergence CONVER
Declare initial values of distribution parameters
Calculate Fm
Calculate distribution parameters
calculate total area
Test for convergence
Area = 1
NO
convergence?
YES
RETURN

Figure 5.18 Subroutine TRIDIS Flow Chart.

5.4.4 Subroutine DSELECT

Randomly selects an Integer task duration on the basis of the duration distribution parameters calculated by the subroutine TRIDIS. Listing of the subroutine is given in **Appendix F.9**.

5.4.4.1 Function RAND

This function is called by the subroutine DSELECT to return a random real number, calculated on the basis of a 'seed', in the range 0.0 to 1.0. The 'seed' requires initialisation prior to the function being called.

If a Repeatable series of random numbers is required, the 'seed' initialised in the main program should be a fixed number. If a non-repeatable series of random numbers is required, the seed is initialised by using the variable IHUN, hundred of a second, in the MICROSOFT FORTRAN compiler built-in function GETTIM(IHR,IMIN,ISEC,IHUND) and the 'seed' initialised should range from 0.0 < SEED < 259200. Listing of the function is given in **Appendix F.10**.

5.4.5 Subroutine PCOSTS

This subroutine performs the second part of the overall analysis: Cost Analysis. This is carried out as follows:

1. By distributing the tasks cost components on the project working duration.

2. Calculates the daily costs and adjust them according to the method of payment intervals and payment delays read from the cost file (FLNAME.CST) generated by the program PERTRA1.

3. Based on the above modified data, it calculates the project CAPTIM and Internal Rate of Return.

Steps of carrying out the above are shown on the subroutine flow chart in **Figure 5.19**. Listing of the subroutine is given in **Appendix F.11**.

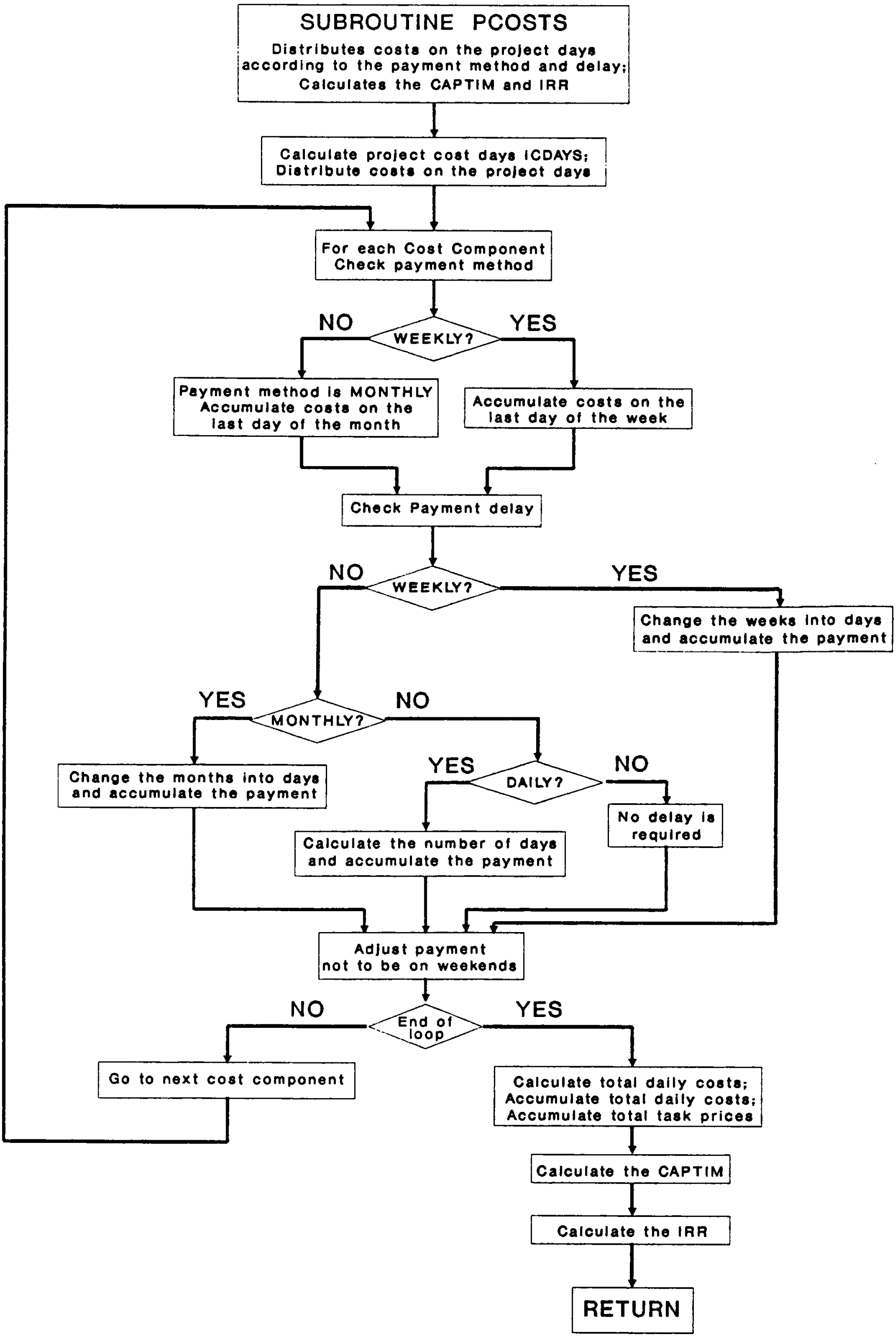

SUBROUTINE PCOSTS
Distributes costs on the project days according to the payment method and delay; Calculates the CAPTIM and IRR
Calculate project cost days ICDAYS; Distribute costs on the project days
For each Cost Component Check payment method
NO
WEEKLY?
YES
Payment method is MONTHLY Accumulate costs on the last day of the month
Accumulate costs on the last day of the week
Check Payment delay
NO
WEEKLY?
YES
Change the weeks into days and accumulate the payment
YES
MONTHLY?
NO
Change the months into days and accumulate the payment
YES
DAILY?
NO
No delay is required
Calculate the number of days and accumulate the payment
Adjust payment not to be on weekends
NO
End of loop
YES
Go to next cost component
Calculate total daily costs; Accumulate total daily costs; Accumulate total task prices
Calculate the CAPTIM
Calculate the IRR
RETURN

Figure 5.19 Subroutine PCOSTS Flow Chart.

5.4.6 Subroutine PSORT

Sorts the project durations, resulting from each 'simmulation run', into rank order in order to perform statistical analysis on the results. Results are shown in Appendix.. Listing of the subroutine is given in **Appendix F.12.**

5.4.7 Subroutine PRSTAT

Performs statistical analysis on the project durations after sorting them in rank order as shown in the flow chart in **Figure 5.20.** Listing of the subroutine is given in **Appendix F.13.**

5.4.8 Subroutine CASTAT

Performs the same steps of subroutine PRSTAT for the sorted CAPTIM results. Listing of the subroutine is given in **Appendix F.14.**

5.4.9 Subroutine RRSTAT

Performs the same steps of subroutines PRSTAT and CASTAT for the sorted IRR results. Listing of the subroutine is given in **Appendix F.15.**

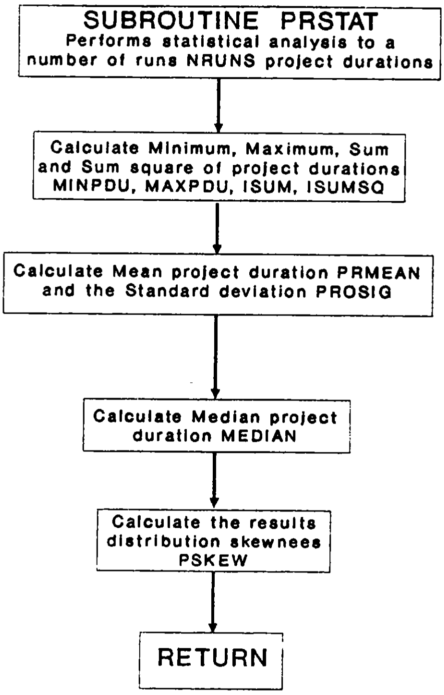

SUBROUTINE PRSTAT
Performs statistical analysis to a number of runs NRUNS project durations
Calculate Minimum, Maximum, Sum and Sum square of project durations MINPDU, MAXPDU, ISUM, ISUMSQ
Calculate Mean project duration PRMEAN and the Standard deviation PROSIG
Calculate Median project duration MEDIAN
Calculate the results distribution skewnees PSKEW
RETURN

Figure 5.20 Subroutine PRSTAT Flow Chart.

5.4.10 Subroutine RRSORT

Sorts the values of Internal Rate of Return (IRR) and CAPTIM, as calculated by each 'simmulation run',into rank order. Listing of the subroutine is given in **Appendix F.16**.

5.4.11 Subroutine FREQUA

Divides the number of runs of project durations into bands, calculate the frequency and cumulative frequency for each band and determine the bands midpoints to allow presenting the results in a graphical mode. Listing is given in **Appendix F.17**.

5.4.12 Subroutine FRQIRR

Performs the same steps as subroutine FREQUA, but, for the number of runs of IRR. The flow chart of the subroutine is shown in **Figure 5.21**. Listing of the subroutine is given in **Appendix F.18**.

5.4.13 Subroutine FRQCAP

Performs the same steps of subroutines FREQUA and FRQIRR for the CAPTIM results. Listing of the subroutine is given in **Appendix F.19**.

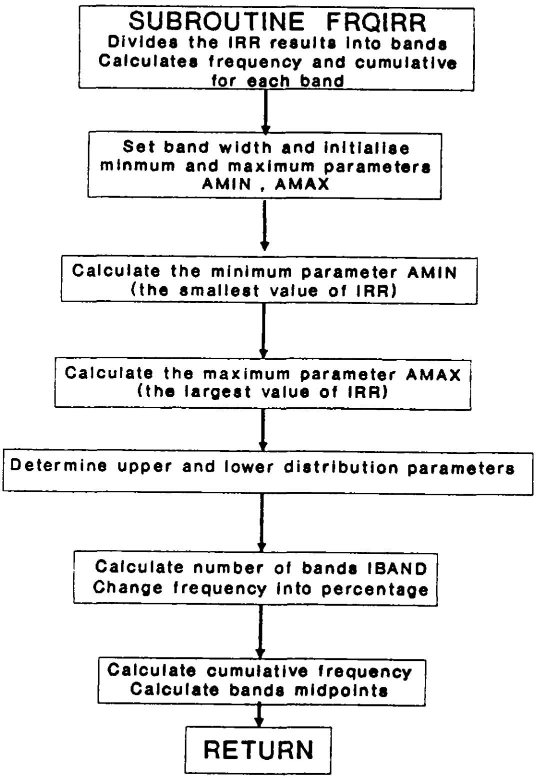

SUBROUTINE FRQIRR
Divides the IRR results into bands
Calculates frequency and cumulative for each band
Set band width and initialise minmum and maximum parameters AMIN , AMAX
Calculate the minimum parameter AMIN (the smallest value of IRR)
Calculate the maximum parameter AMAX (the largest value of IRR)
Determine upper and lower distribution parameters
Calculate number of bands IBAND
Change frequency into percentage
Calculate cumulative frequency
Calculate bands midpoints
RETURN

Figure 5.21 Subroutine FRQIRR Flow Chart.

CHAPTER 6

PROGRAM OPERATION

6.1 PERTRA1

6.1.1 Screen Displays

With a view to ensuring that the program is user–friendly and straightforward to operate – two of the prime requirements of any computer program – PERTRA1 generates a number of screen displays to facilitate the interactive input of data from the keyboard and to keep the user abreast of exactly what the program is doing at any given time. Each screen display is of the form shown in **Figure 6.1**

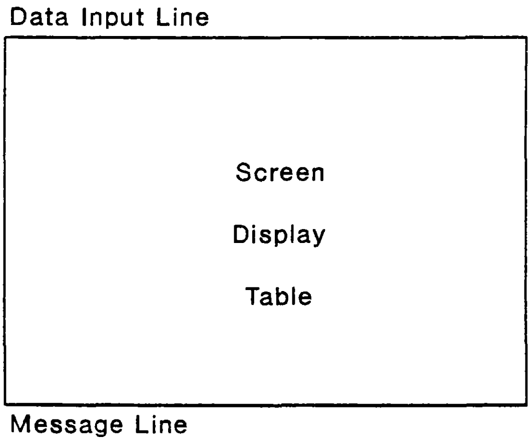


Figure 6.1 Programs Screen Display

6.1.1.1 Data Input Line

In cases where alphanumeric data is required to be input, the data input line will display details of the data required and will contain a pair of brackets [–   ], containing a flashing cursor, indicating the maximum permissible length of the input string. Mistakes may be rectified in the normal way by using the 'backspace' key.

If the input string exceeds the maximum permissible length, a warning bleep is sounded.

In cases where the required input consists of selection of a number of options, the data input line will display a menu of available options highlighting the first character of each option. Selection of the required option is carried out by pressing the required key.

6.1.1.2 Screen Display Table

The screen display table has two functions, namely:

(a) To display introduction or termination information when entering or 'quitting' the program.

(b) To display, in tabular form (for clarity), data which has been input by the user.

6.1.1.3 Message Line

The message line displays either of two types of message:

(a) A message briefing the user (when appropriate) as to what the program is doing at any given time.

(b) A messages containing information (when appropriate) pertaining to the input of data by the user.

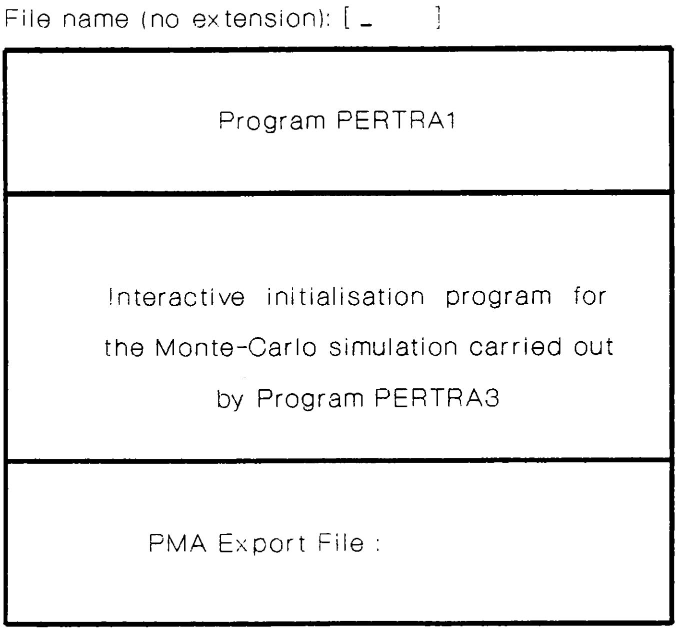
6.1.2 Operational Details

6.1.2.1 Initialisation

Program operation is commenced, in the usual manner, by typing the program name **PERTRA1**.

The resultant screen display – shown in **Figure 6.2** – indicates for the user to input the name (with no extension) of the PMA–generated export file which the program is required to read.

The input name is displayed as shown in **Figure 6.3** (which assumes the input file name to be **BRIDGE**) and the user is given the option of confirming the file name or repeating the exercise in the event of an error in the input. The appropriate options are displayed in the data input line, selection of which is made by typing **Y** or **N**.



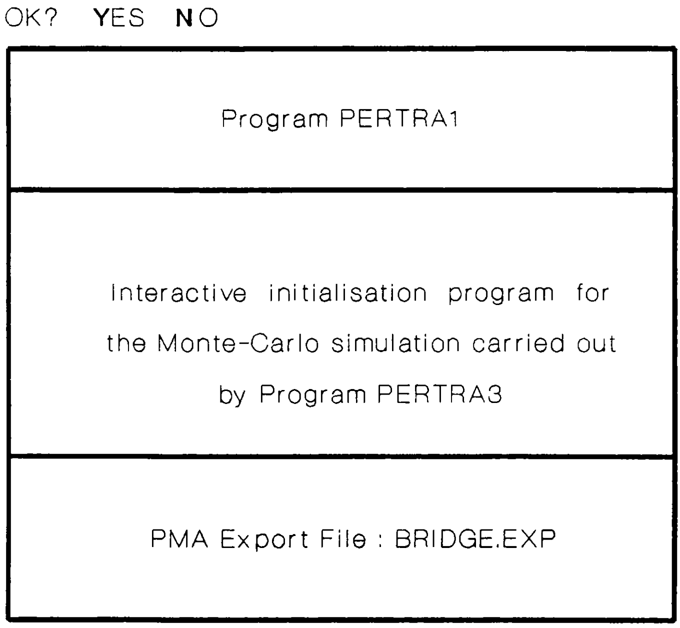
File name (no extension): [_]

Program PERTRA1

Interactive initialisation program for the Monte-Carlo simulation carried out by Program PERTRA3

PMA Export File :

Figure 6.2 File Name Input Screen Display



OK? YES NO

Program PERTRA1

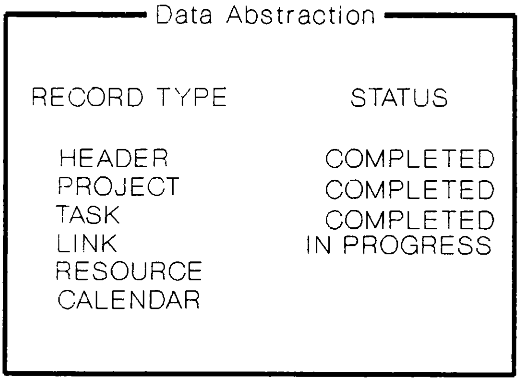
Interactive initialisation program for the Monte-Carlo simulation carried out by Program PERTRA3

PMA Export File : BRIDGE.EXP

Figure 6.3 File Name Confirmation Screen Display.

6.1.2.2 Reading of Data File

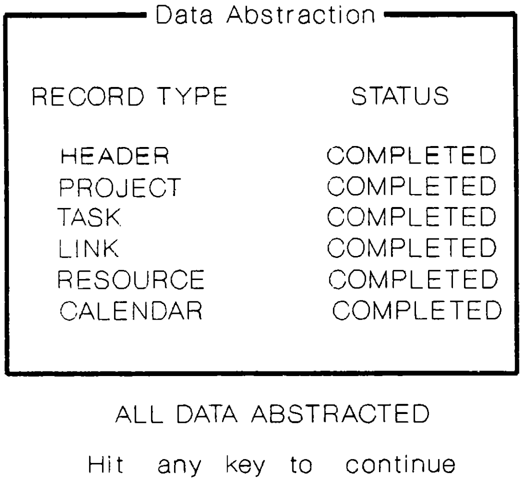
Typing the letter **Y** will result in the display of the data abstraction screen as shown in **Figure 6.4**. The status of the data abstraction will be changing on the screen as the program finishes the relevant data reading.



```
──────── Data Abstraction ────────

RECORD TYPE              STATUS

   HEADER               COMPLETED
   PROJECT              COMPLETED
   TASK                 COMPLETED
   LINK               IN PROGRESS
   RESOURCE
   CALENDAR
```

Figure 6.4 Data Abstraction 1 Screen Display

When all data is abstracted, a message (ALL DATA ABSTRACTED) will appear on the message line and the screen display will be as in **Figure 6.5**. At this stage the user is requested to hit any key for continuation.

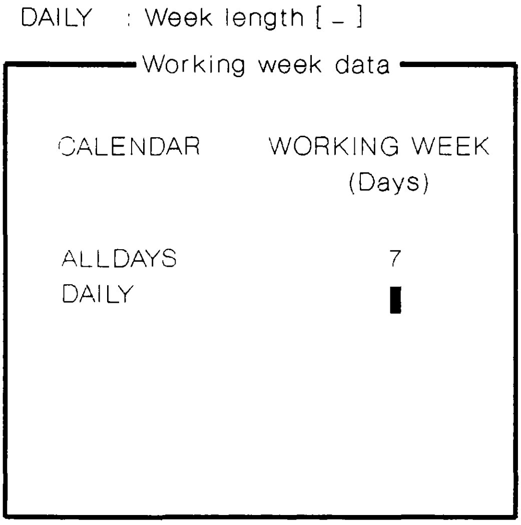


```
                Data Abstraction

RECORD TYPE                  STATUS

  HEADER                   COMPLETED
  PROJECT                  COMPLETED
  TASK                     COMPLETED
  LINK                     COMPLETED
  RESOURCE                 COMPLETED
  CALENDAR                 COMPLETED

           ALL DATA ABSTRACTED
         Hit  any  key  to  continue
```

Figure 6.5 Data Abstraction 2 Screen Display

6.1.2.3 Input of Working Week Data

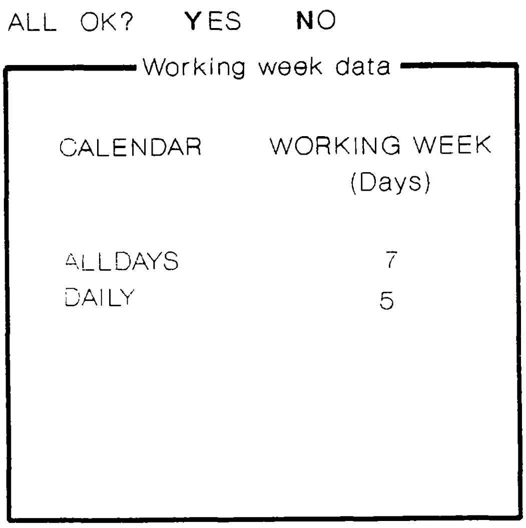
Hitting any key will result the display of the working week data screen (see **Figure 6.6**). The number of working days for each displayed calendar should be input from the keyboard.



```
DAILY   : Week length [ _ ]
         Working week data

CALENDAR        WORKING WEEK
                   (Days)

ALLDAYS              7
DAILY
```

Figure 6.6 Working Week Input Screen Display

Finishing the- input of all such data, the user is given the option of confirming the displayed data or repeating the exercise in the event of an error in such. These options are given in the confirmation screen shown in Figure 6.7.



ALL OK? YES NO

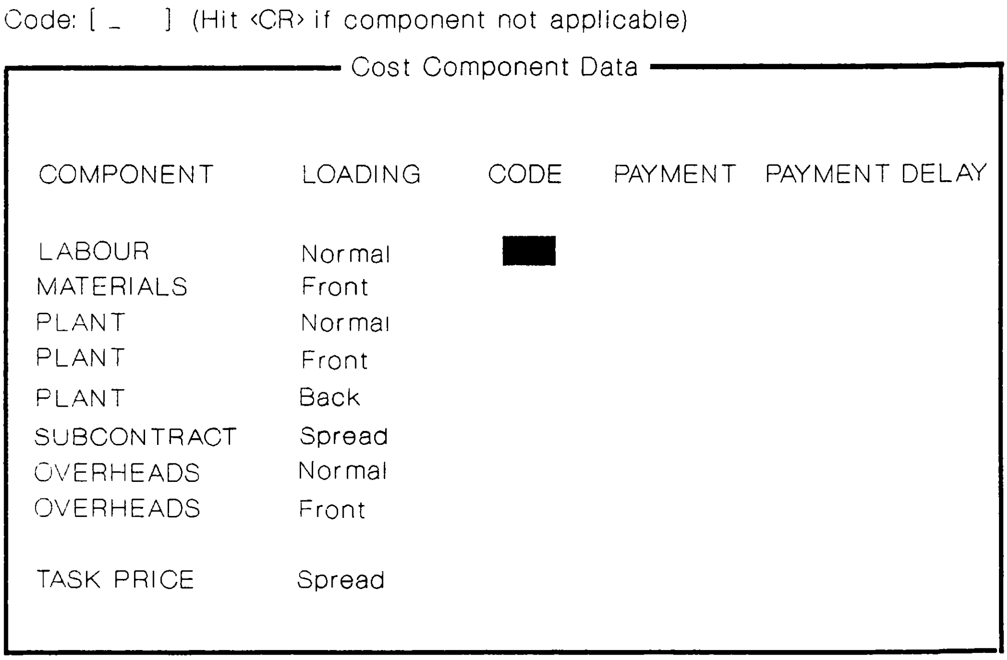
Working week data

CALENDAR	WORKING WEEK (Days)
ALLDAYS	7
DAILY	5

Figure 6.7 Working Week Confirmation Screen Display

6.1.2.4 *Input of .Cost Component Data*

Confirmation of the correctness of working- week data will result in the cost component data screen display as shown in **Figure 6.8.**

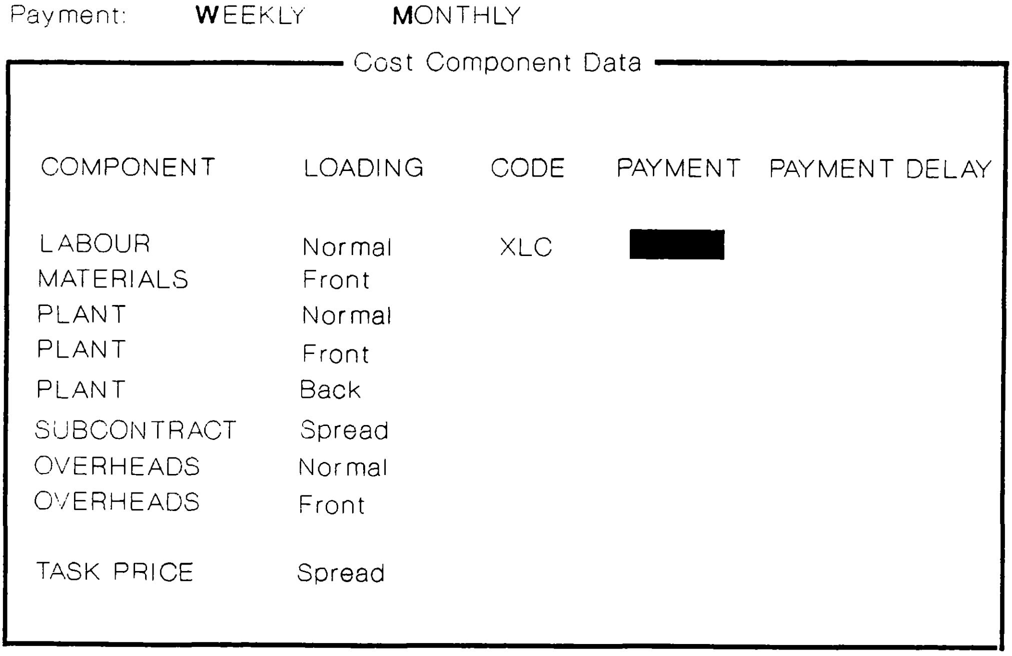


```
Code: [ _    ] (Hit <CR> if component not applicable)
──────────────── Cost Component Data ────────────────

 COMPONENT        LOADING     CODE    PAYMENT   PAYMENT DELAY

 LABOUR           Normal      █
 MATERIALS        Front
 PLANT            Normal
 PLANT            Front
 PLANT            Back
 SUBCONTRACT      Spread
 OVERHEADS        Normal
 OVERHEADS        Front

 TASK PRICE       Spread
```

Figure 6.8 Cost Components Data Screen Display

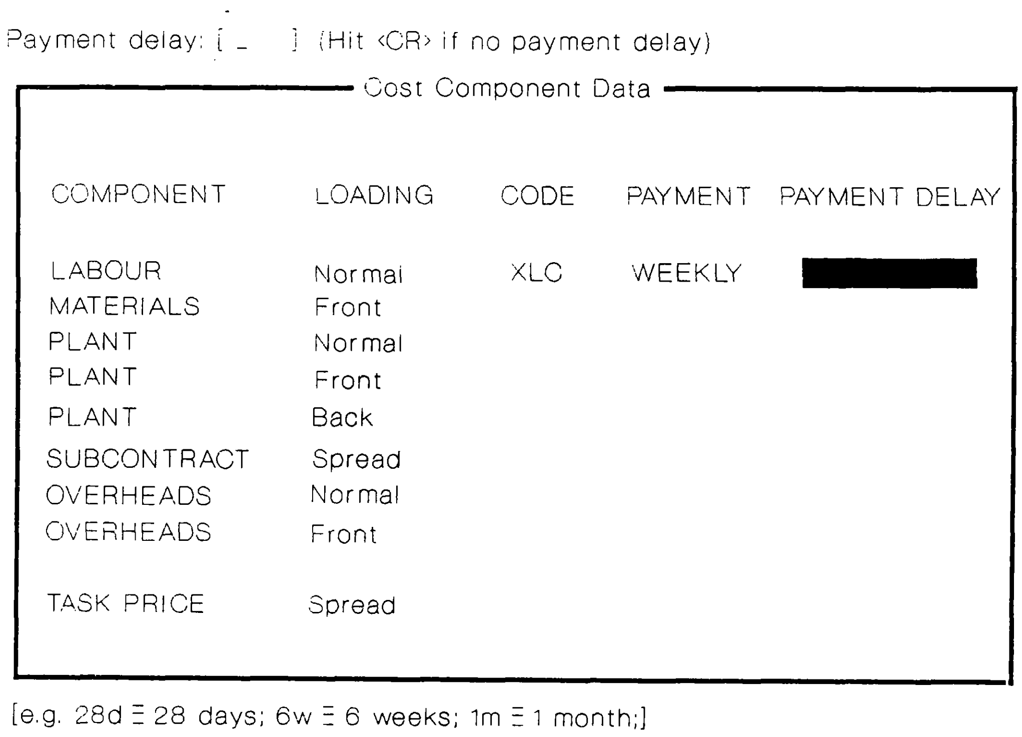
At this stage- the user is requested to input the cost component code as defined by **PMA** or hitting the **<CR>** key if the cost component is not applicable. If the input code is not defined, a message will appear in the message line informing the user that the cost component code is not found and should try again to input the correct code.

Screens display for the input of the **CODE**, **PAYMENT** (weekly or monthly) and **PAYMENT DELAY** (days or weeks or months) are shown in **Figure 6.9**, **Figure 6.10** and **Figure 6.11** respectively.



```
Payment:      WEEKLY        MONTHLY

                        Cost Component Data

 COMPONENT          LOADING        CODE     PAYMENT   PAYMENT DELAY

 LABOUR             Normal         XLC      ████
 MATERIALS          Front
 PLANT              Normal
 PLANT              Front
 PLANT              Back
 SUBCONTRACT        Spread
 OVERHEADS          Normal
 OVERHEADS          Front

 TASK PRICE         Spread
```

Figure 6.9 Cost component Data Screen Display (2)



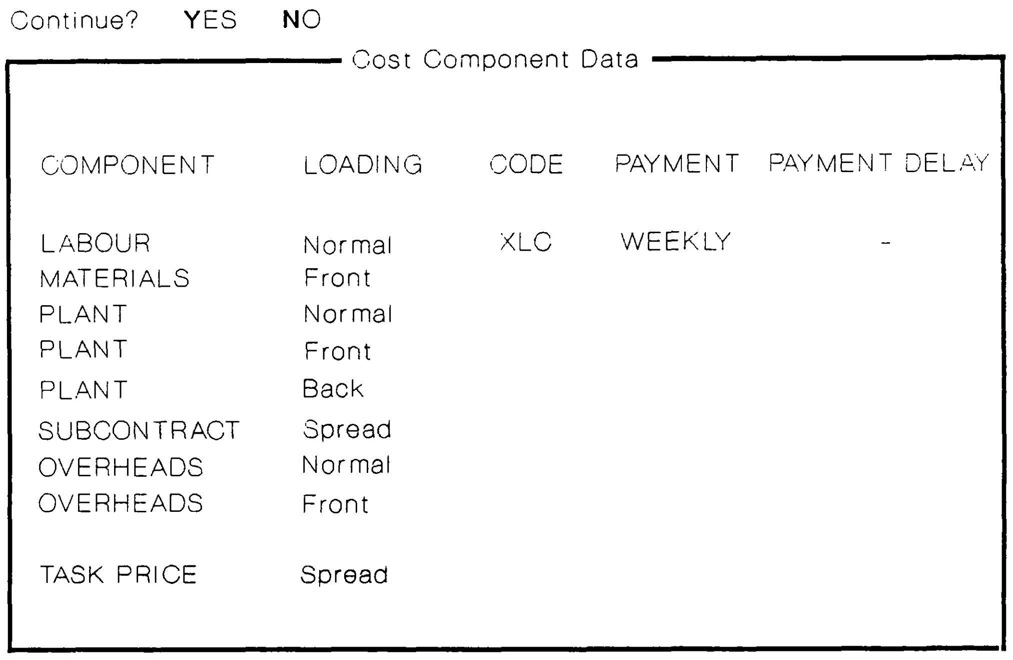
Payment delay: [_] (Hit <CR> if no payment delay)

Cost Component Data

COMPONENT	LOADING	CODE	PAYMENT	PAYMENT DELAY
LABOUR	Normal	XLC	WEEKLY	[redacted]
MATERIALS	Front			
PLANT	Normal			
PLANT	Front			
PLANT	Back			
SUBCONTRACT	Spread			
OVERHEADS	Normal			
OVERHEADS	Front			
TASK PRICE	Spread			

[e.g. 28d ☰ 28 days; 6w ☰ 6 weeks; 1m ☰ 1 month;]

Figure 6.10 Cost Component Data Screen Display (3)



Continue? YES NO

Cost Component Data

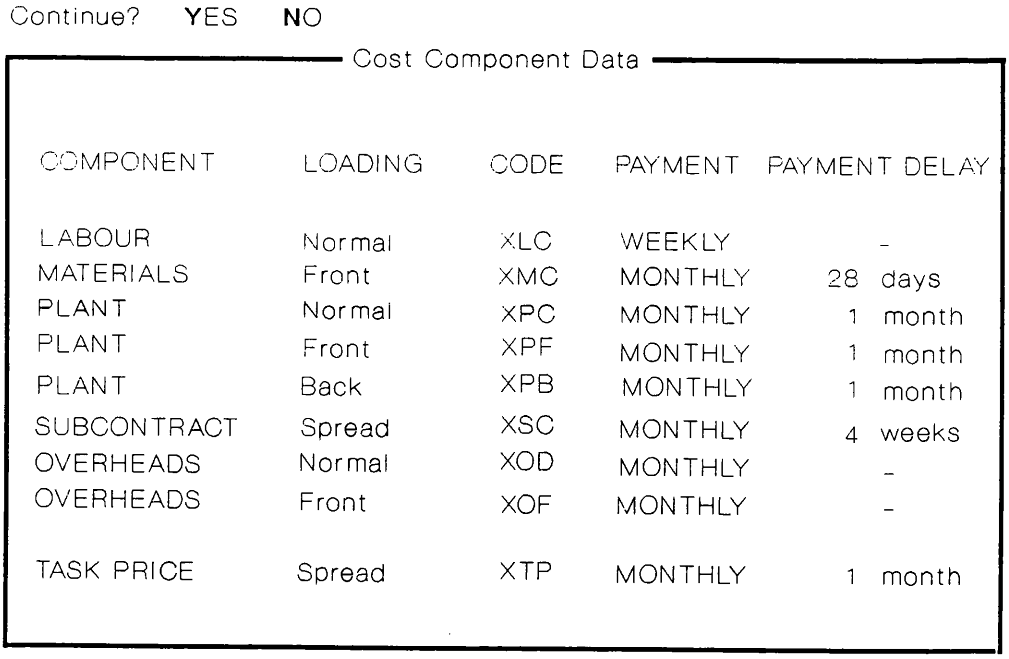
COMPONENT	LOADING	CODE	PAYMENT	PAYMENT DELAY
LABOUR	Normal	XLC	WEEKLY	-
MATERIALS	Front			
PLANT	Normal			
PLANT	Front			
PLANT	Back			
SUBCONTRACT	Spread			
OVERHEADS	Normal			
OVERHEADS	Front			
TASK PRICE	Spread			

Figure 6.11 Cost Component Data Continuation Screen Display (1).

By finishing the data input of a cost component as shown in **Figure 6.11**, the user is given the option of confirming the accuracy of the data input for the relevant cost component or repeating the exercise in the event of any error in the input. Selection of the option is made by typing **Y** or **N**.

The same process is continued for the other components bearing in mind that, for the cost components **PLANT** (**FRONT** and **BACK**) and **OVERHEADS** (**FRONT**), the user will not have the option for the input of their **PAYMENT** and **PAYMENT DELAY** as they will be assumed and displayed as the input of the **NORMAL** cost component for each of them.

Finishing the above, a continuation screen as in **Figure 6.12** will be displayed given the user the option to continue or repeat the exercise once again. Continuation options is made by typing **Y** or **N**.



Continue? YES NO

Cost Component Data

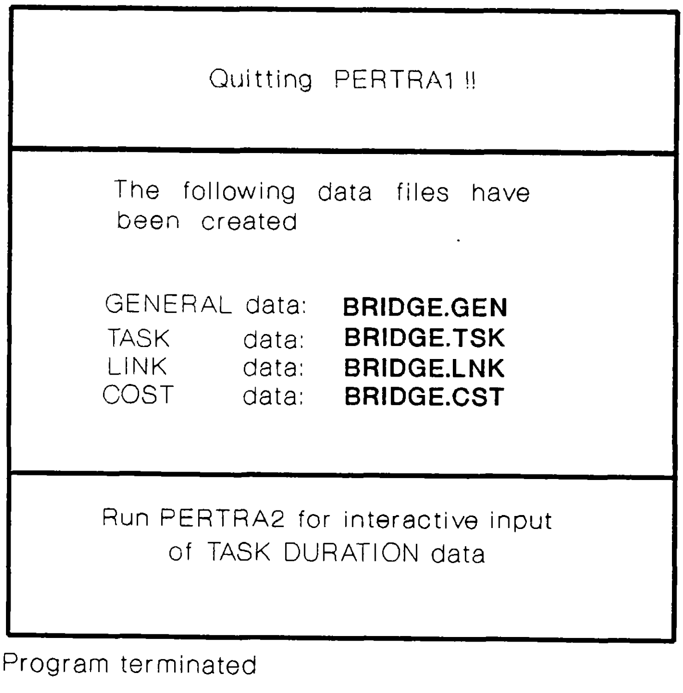
COMPONENT	LOADING	CODE	PAYMENT	PAYMENT DELAY
LABOUR	Normal	XLC	WEEKLY	-
MATERIALS	Front	XMC	MONTHLY	28 days
PLANT	Normal	XPC	MONTHLY	1 month
PLANT	Front	XPF	MONTHLY	1 month
PLANT	Back	XPB	MONTHLY	1 month
SUBCONTRACT	Spread	XSC	MONTHLY	4 weeks
OVERHEADS	Normal	XOD	MONTHLY	-
OVERHEADS	Front	XOF	MONTHLY	-
TASK PRICE	Spread	XTP	MONTHLY	1 month

Figure 6.12 Cost Component Continuation Display (2).

6.1.2.5 Generation of Data Files and Program Termination

Typing the option Y will result in the 'termination' screen display shown in **Figure 6.13** which will show the data files written and to be used by the programs **PERTRA2** and **PERTRA3**.

When all the data is written to the relevant files a termination message will appear in the message line.



```
                 Quitting PERTRA1 !!

        The following data files have
        been created

       GENERAL data:  BRIDGE.GEN
       TASK    data:  BRIDGE.TSK
       LINK    data:  BRIDGE.LNK
       COST    data:  BRIDGE.CST

       Run PERTRA2 for interactive input
           of TASK DURATION data

Program terminated
```

Figure 6.13 Termination Display Screen

6.2 PERTRA2

6.2.1 Screen Displays

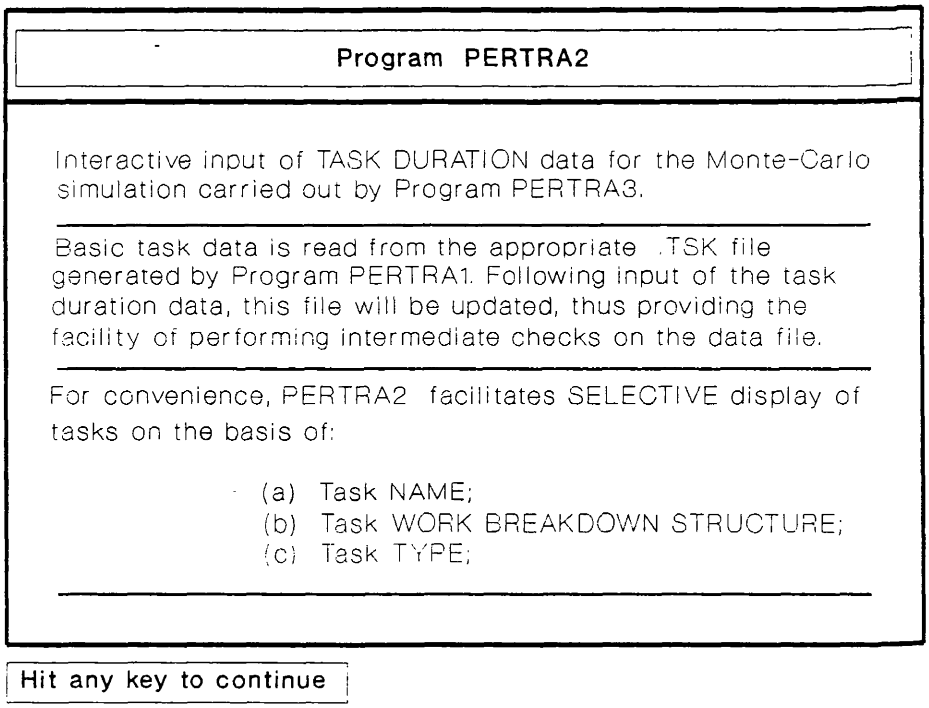
PERTRA2 generates the same screen types as generated by the program PERTRA1 (see **Section 6.1.1**).

6.2.2 Operational Details

6.2.2.1 Program Introduction

Program operation is commenced, in the usual manner, by typing the program name **PERTRA2**.

The resultant is the 'introductory' screen shown in **Figures 6.14 to 6.16** which displays information about PERTRA2.



Program PERTRA2

Interactive input of TASK DURATION data for the Monte-Carlo simulation carried out by Program PERTRA3.

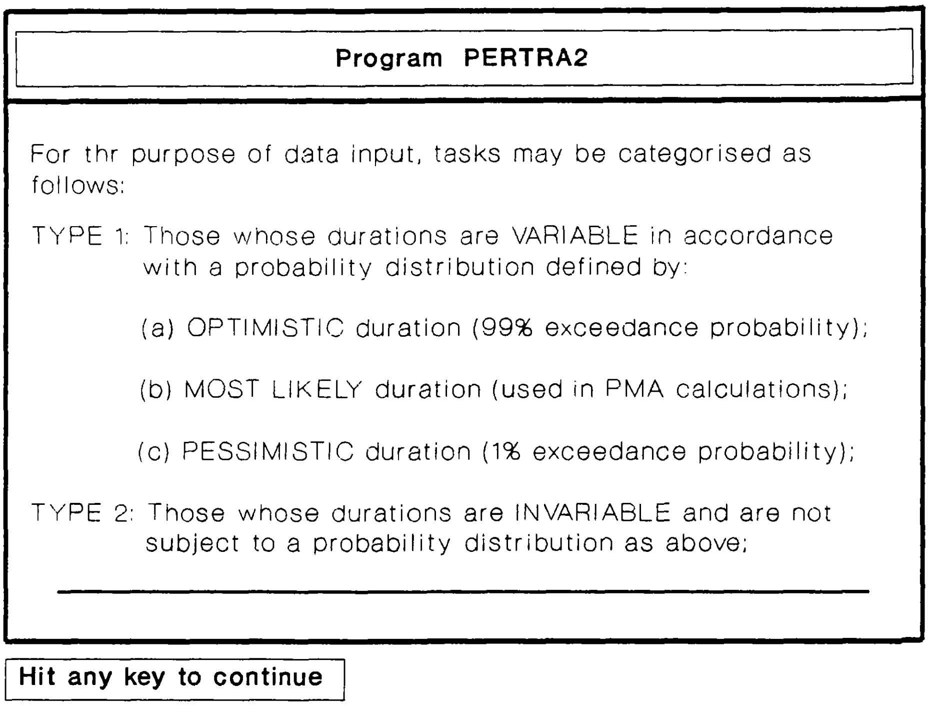
Basic task data is read from the appropriate .TSK file generated by Program PERTRA1. Following input of the task duration data, this file will be updated, thus providing the facility of performing intermediate checks on the data file.

For convenience, PERTRA2 facilitates SELECTIVE display of tasks on the basis of:

(a) Task NAME;
(b) Task WORK BREAKDOWN STRUCTURE;
(c) Task TYPE;

Hit any key to continue

Figure 6.14 Program Introductory Screen 1



Program PERTRA2

For thr purpose of data input, tasks may be categorised as follows:

TYPE 1: Those whose durations are VARIABLE in accordance with a probability distribution defined by:

(a) OPTIMISTIC duration (99% exceedance probability);

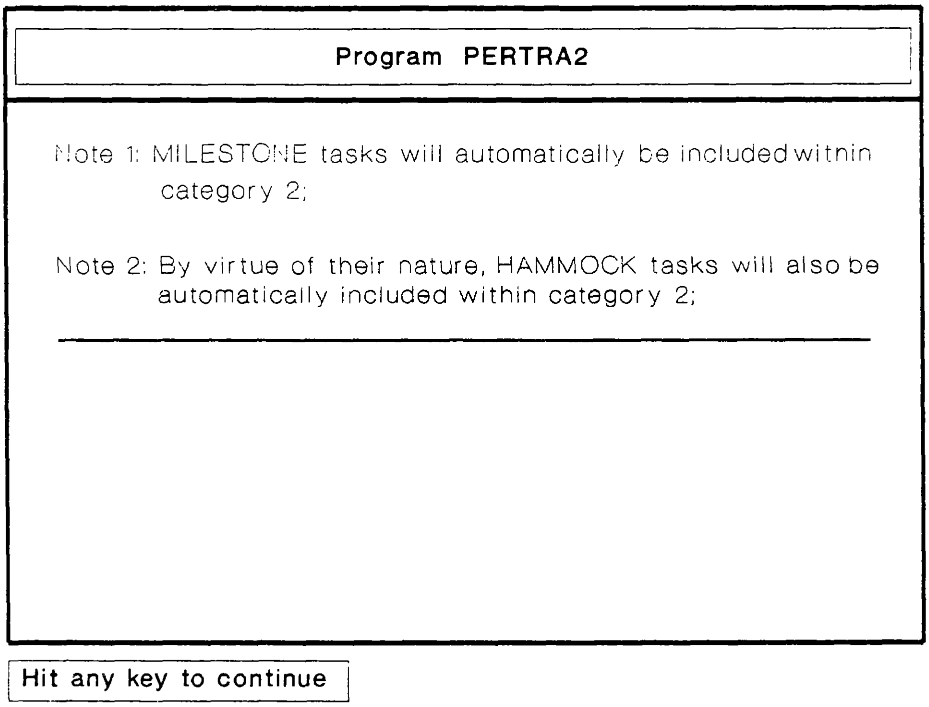
(b) MOST LIKELY duration (used in PMA calculations);

(c) PESSIMISTIC duration (1% exceedance probability);

TYPE 2: Those whose durations are INVARIABLE and are not subject to a probability distribution as above;

Hit any key to continue

Figure 6.15 Program Introductory Screen 2



Program PERTRA2

Note 1: MILESTONE tasks will automatically be included witnin category 2;

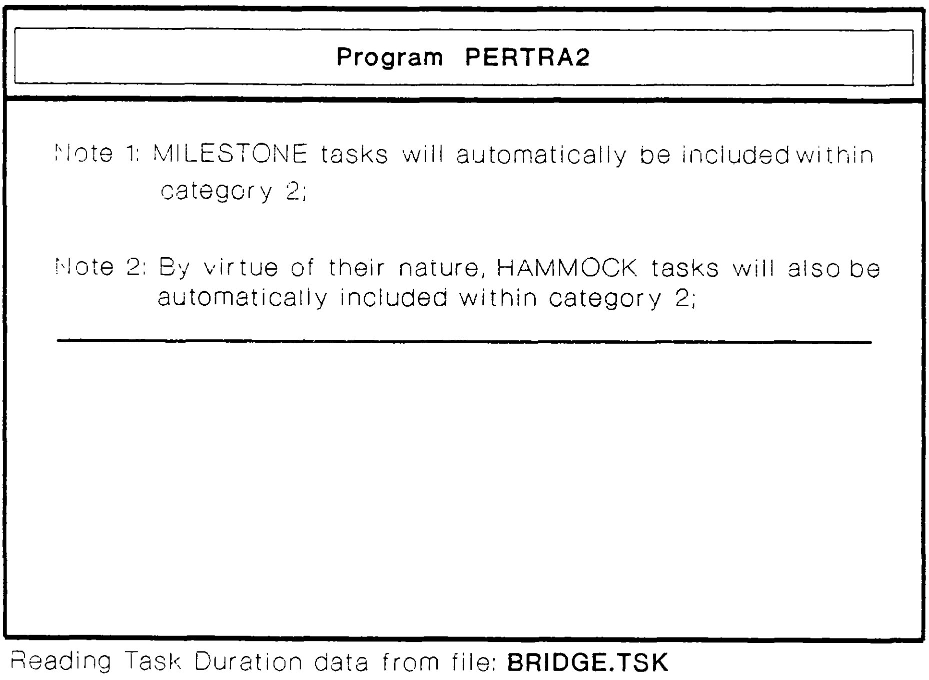
Note 2: By virtue of their nature, HAMMOCK tasks will also be automatically included within category 2;

Hit any key to continue

Figure 6.16 Program Introductory Screen 3

6.2.2.2 Reading Task Duration Data

Hitting any key after the program introduction result in the display of reading data screen as shown in **Figure 6.17** which is briefing the user of what the program is doing at that stage.

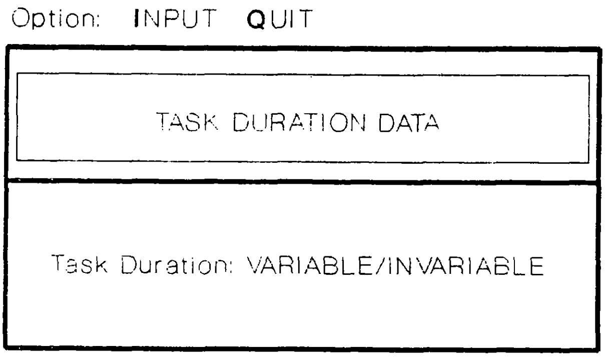


```
                     Program PERTRA2

Note 1: MILESTONE tasks will automatically be included within
        category 2;

Note 2: By virtue of their nature, HAMMOCK tasks will also be
        automatically included within category 2;
____________________________________________________________

Reading Task Duration data from file: BRIDGE.TSK
```

Figure 6.17 Reading Data Screen

6.2.2.3 Data Input

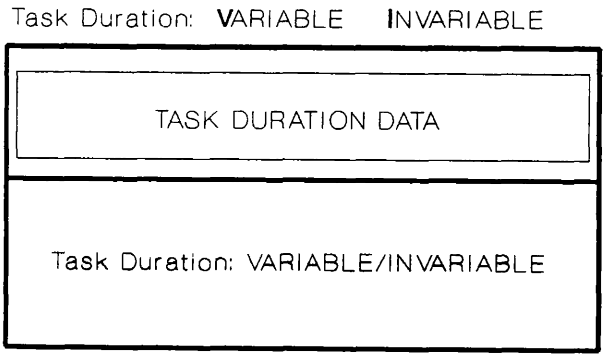
Automatically the program will display the task duration data screen as shown in **Figure 6.18** giving the user the option to input the task duration data or quit the program. The appropriate option is selected by typing the letter **I** or **Q**. Selection of the **INPUT** option results in the screen display shown in **Figure 6.19** giving the user the option to select the type of task duration to input **(VARIABLE/INVARIABLE)**. The option is selected in the usual manner by typing the one of the highlighted letters **V** or **I**.


Option: INPUT QUIT

TASK DURATION DATA

Task Duration: VARIABLE/INVARIABLE

Figure 6.18 Task Duration Data Screen 1


Task Duration: VARIABLE INVARIABLE

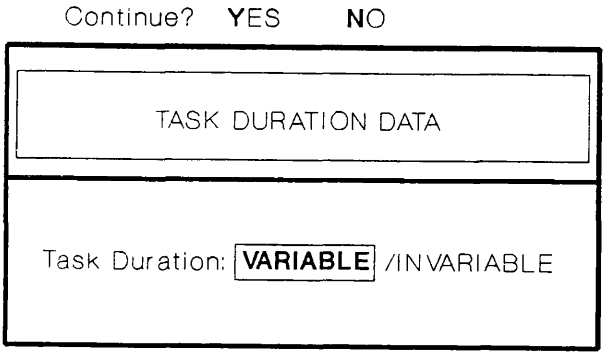
TASK DURATION DATA

Task Duration: VARIABLE/INVARIABLE

Figure 6.19 Task Duration Data Screen 2

6.2.2.4 Input of- Tasks with VARIABLE Durations

Selecting the option V will result in the display of the screen shown in **Figure 6.20** giving the user the option of confirming the input of task duration data for tasks with **VARIABLE** durations (see **Section 5.3.(a)**)by typing the letter Y or going back to the previous screen by typing the letter N and repeating the exercise again.



Continue? YES NO

TASK DURATION DATA

Task Duration: [✓ VARIABLE] /INVARIABLE

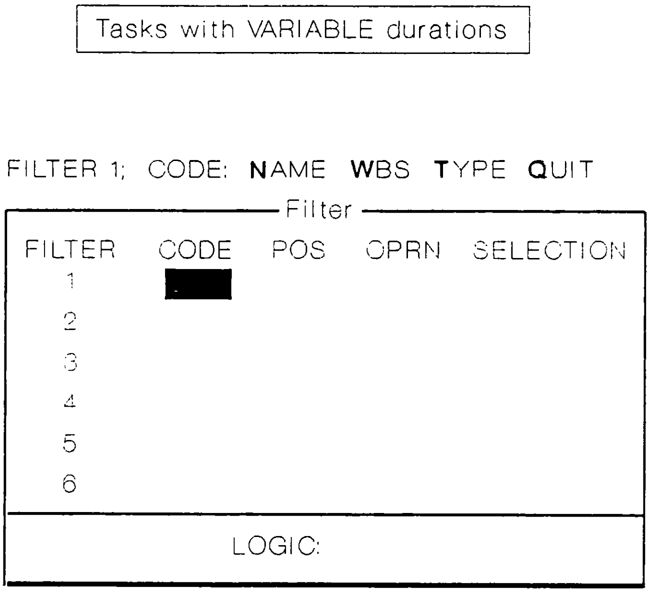
Figure 6.20 Tasks Duration Screen 3

6.2.2.4.1 Filtration Parameters

Continuing will result in the display of the first filtration parameters screen as shown in **Figure 6.21**. The user is given the option of selecting one of the following filtration parameters (see **Section 5.3.2**):

1. Task Name (**NAME**);
2. Task Type (**TYPE**);
3. Task Work Breakdown Structure (**WBS**);

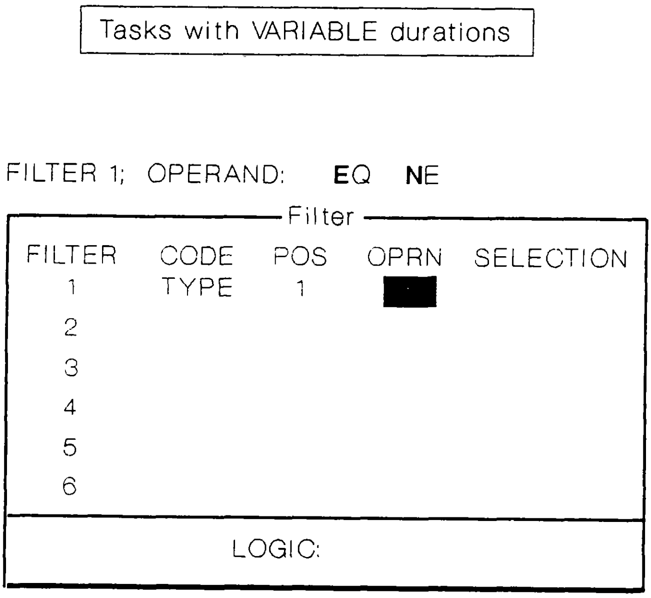
or the option of quitting the input of data (**QUIT**).



```
Tasks with VARIABLE durations

FILTER 1; CODE: NAME  WBS  TYPE  QUIT
                 Filter
FILTER   CODE   POS   OPRN   SELECTION
  1
  2
  3
  4
  5
  6

               LOGIC:
```

Figure 6.21 First Filter Parameters Screen 1

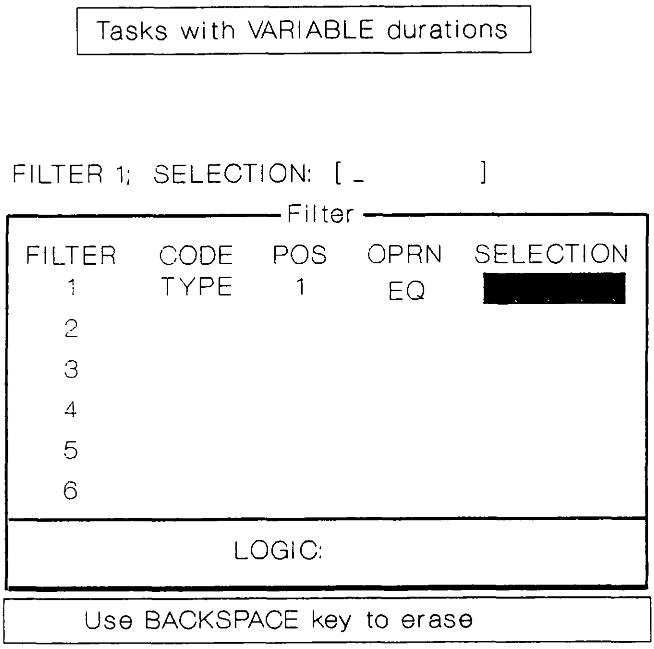
In case of selection of the code **TYPE** , the user will not be required to input the parameter **POS** as, the program will automatically display, as shown in **Figure 6.22,** the number 1 highlighting the third filtration parameter **OPERAND** giving the user the options equal (EQ) or not equal (NE).



```
Tasks with VARIABLE durations

FILTER 1;  OPERAND:   EQ   NE
------------------- Filter -------------------
FILTER   CODE   POS   OPRN   SELECTION
   1     TYPE    1     █
   2
   3
   4
   5
   6
----------------------------------------------
               LOGIC:
```

Figure 6.22 First Filter Parameters Screen 2

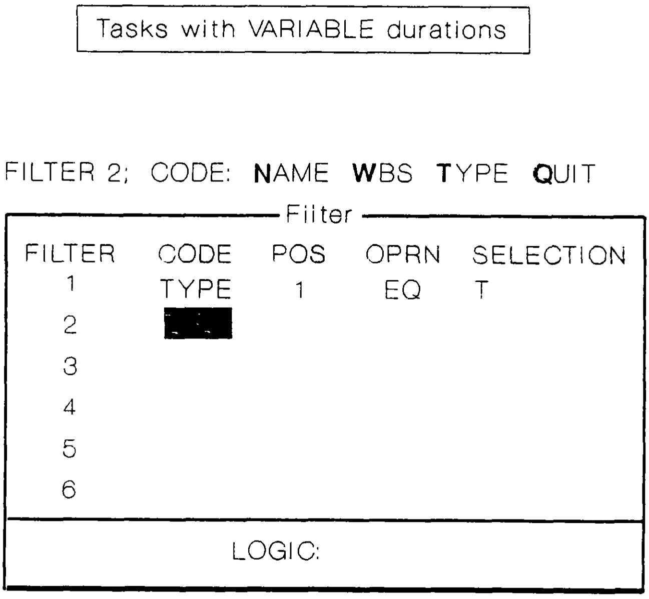
Selection of -the option **EQ** by typing the highlighted letter E will result in the display of the **SELECTION** as shown in **Figure 6.23** and the user is required to input the type of tasks to be selected. Typing mistakes can be corrected by using the backspace key as indicated in the message line.



```
Tasks with VARIABLE durations

FILTER 1; SELECTION: [_          ]
                    Filter
FILTER   CODE   POS   OPRN   SELECTION
   1     TYPE    1     EQ
   2
   3
   4
   5
   6

               LOGIC:

Use BACKSPACE key to erase
```

Figure 6.23 First Filter Parameters screen 3

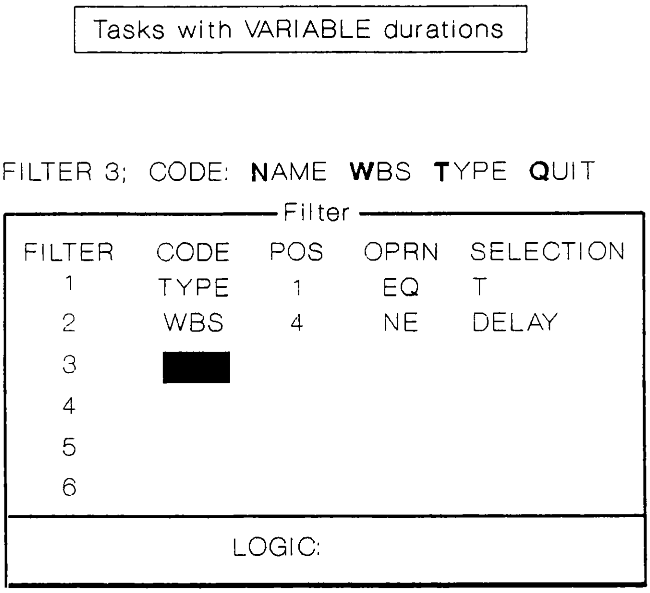
Typing the task type, for example, T and hitting the <CR> key will result in the display of the second filter parameters screen shown in **Figure 6.24**.



```
                Tasks with VARIABLE durations


          FILTER 2; CODE: NAME WBS TYPE QUIT
          ----------------- Filter -----------------
           FILTER   CODE   POS   OPRN   SELECTION
             1      TYPE    1     EQ    T
             2
             3
             4
             5
             6
          ------------------------------------------
                        LOGIC:
```

Figure 6.24 Second Filter Parameters Screen 1

Repeating the same exercise, for example, selection of all tasks with work breakdown structure, starting from the fourth character, not equal to DELAY will result in the display of the screen shown in **Figure 6.25**.



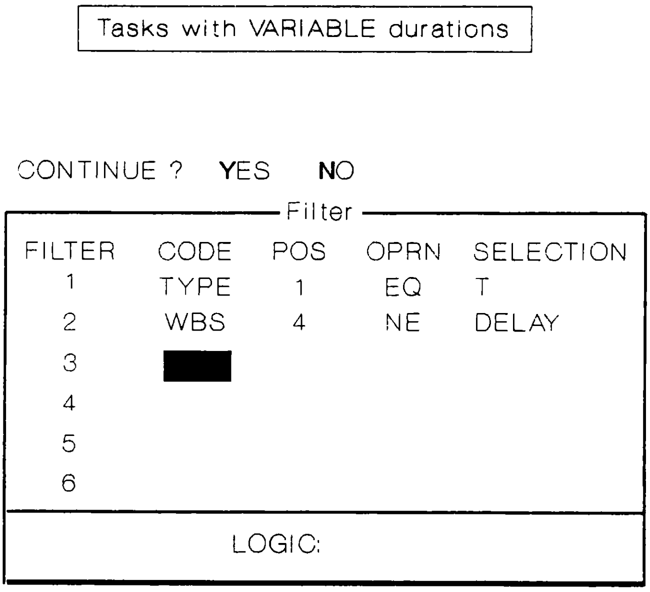
```
          Tasks with VARIABLE durations


FILTER 3;  CODE:  NAME  WBS  TYPE  QUIT
 ------------------ Filter ------------------
  FILTER   CODE   POS   OPRN   SELECTION
     1     TYPE    1     EQ    T
     2     WBS     4     NE    DELAY
     3     █
     4
     5
     6
 --------------------------------------------
                  LOGIC:
```

Figure 6.25 Second Filter Parameters Screen 5

Following the input of the requisite number of filtration parameters, up to a maximum of six, the **QUIT** option should be chosen to allow the user to input the logic parameters.

6.2.2.4.2 Filtration Parameters Logic

At this stage (see **Figure 6.26**), the user is giving the option of confirming the displayed parameters (**Y**) or repeating the exercise (**N**) in the event of an error being detected.



Tasks with VARIABLE durations

CONTINUE ? YES **NO**

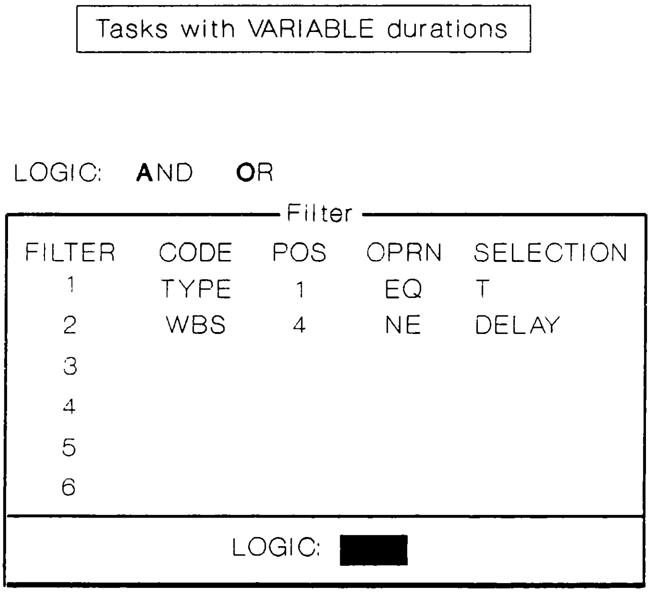
Filter

FILTER	CODE	POS	OPRN	SELECTION
1	TYPE	1	EQ	T
2	WBS	4	NE	DELAY
3				
4				
5				
6				

LOGIC:

Figure 6.26 Filters Confirmation Screen

If **Y** is selected, the logic input screen (see **Figure 6.27**) is displayed, giving the user the option of input **AND** or **OR**. If **AND** is chosen, each of the displayed filtration criteria must be met for a task to be selected. If **OR** is chosen, only one of the displayed criteria must be met for a task to be selected.



Tasks with VARIABLE durations

LOGIC: **AND** **OR**

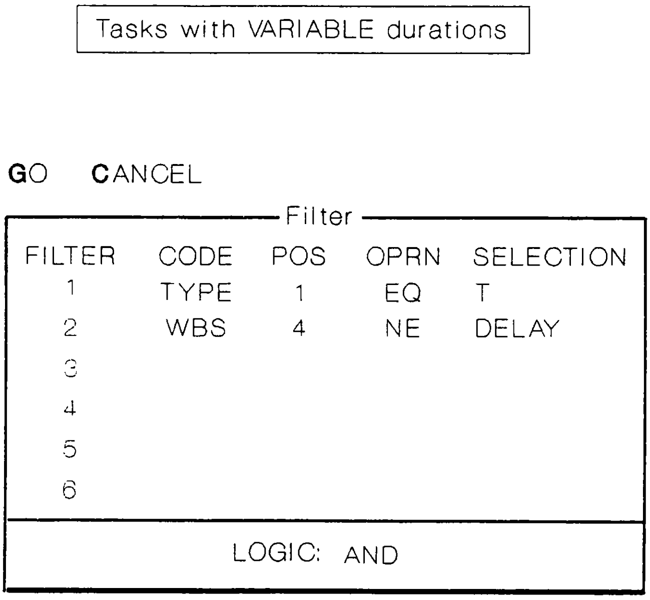
Filter

FILTER	CODE	POS	OPRN	SELECTION
1	TYPE	1	EQ	T
2	WBS	4	NE	DELAY
3				
4				
5				
6				

LOGIC:

Figure 6.27 Filters Logic Screen

For this example, both of the filtration criteria should be met and selection of the option **AND** will result in the screen display shown in **Figure 6.28**.



Tasks with VARIABLE durations

GO CANCEL

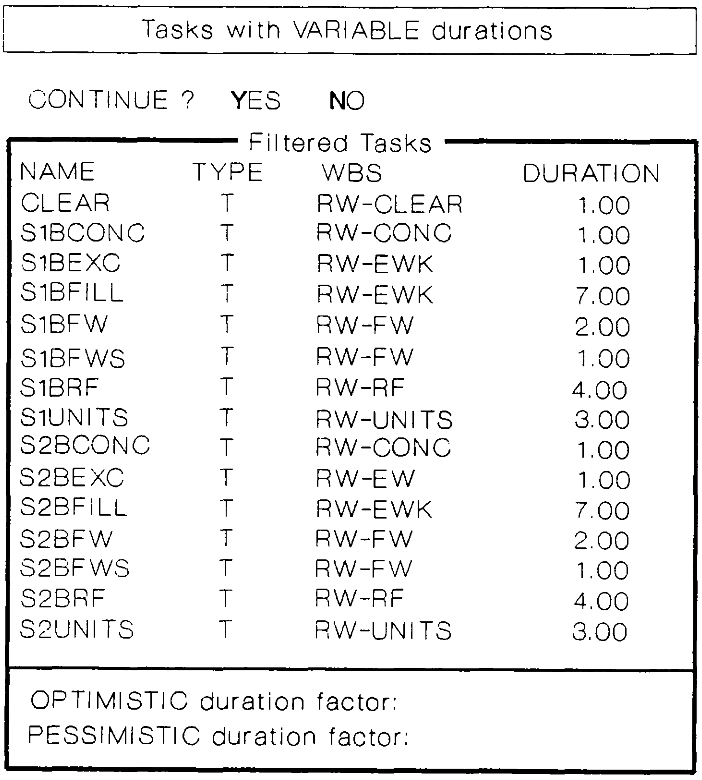
Filter

FILTER	CODE	POS	OPRN	SELECTION
1	TYPE	1	EQ	T
2	WBS	4	NE	DELAY
3				
4				
5				
6				

LOGIC: AND

Figure 6.28 Filters Logic Confirmation Screen

Selection of - the option **CANCEL** will delete the previously selected filtration parameters and logic and the exercise is to be repeated again. Selection of the option **GO** will result in the display of the selected tasks as shown in the sample given in **Figure 6.29**.



Tasks with VARIABLE durations

CONTINUE ? YES NO

Filtered Tasks

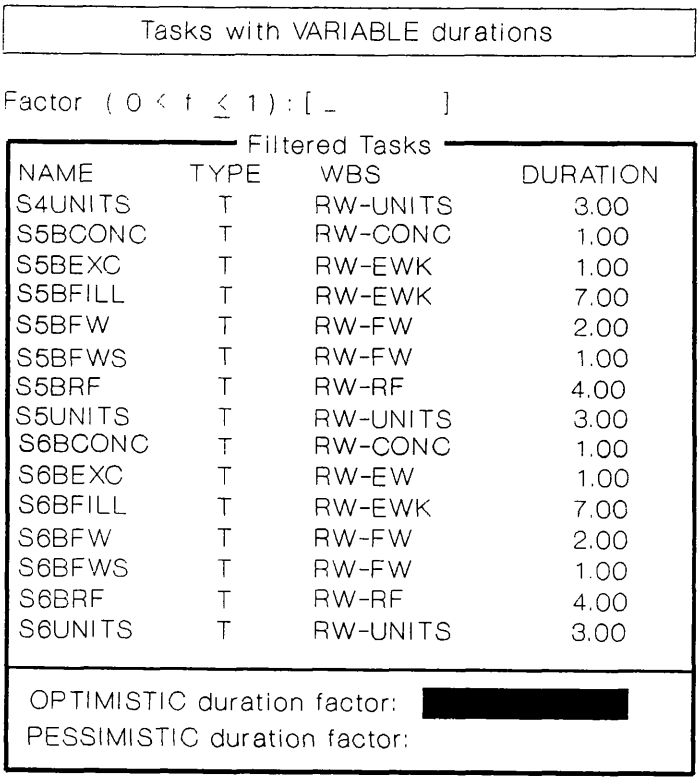
NAME	TYPE	WBS	DURATION
CLEAR	T	RW-CLEAR	1.00
S1BCONC	T	RW-CONC	1.00
S1BEXC	T	RW-EWK	1.00
S1BFILL	T	RW-EWK	7.00
S1BFW	T	RW-FW	2.00
S1BFWS	T	RW-FW	1.00
S1BRF	T	RW-RF	4.00
S1UNITS	T	RW-UNITS	3.00
S2BCONC	T	RW-CONC	1.00
S2BEXC	T	RW-EW	1.00
S2BFILL	T	RW-EWK	7.00
S2BFW	T	RW-FW	2.00
S2BFWS	T	RW-FW	1.00
S2BRF	T	RW-RF	4.00
S2UNITS	T	RW-UNITS	3.00

OPTIMISTIC duration factor:
PESSIMISTIC duration factor:

Figure 6.29 Sample of Tasks With VARIABLE Duration Display

6.2.2.4.3 Input of Optimistic and Pessimistic Durations Factors

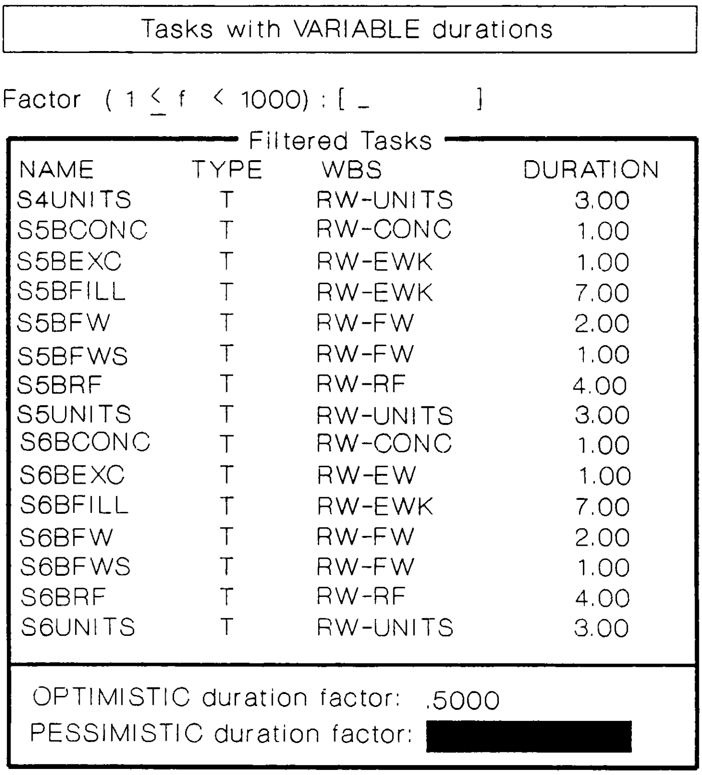
Confirming the accuracy of the displayed tasks will result in the display of the screen shown in **Figure 6.30** and the input of the Optimistic duration factor (see **Section 5.3.4**) will be highlighted. The data input line is showing the range of the factor to input.



```
                 Tasks with VARIABLE durations

Factor ( 0 < f ≤ 1 ) : [ _          ]

                        Filtered Tasks
NAME          TYPE     WBS                DURATION
S4UNITS        T       RW-UNITS             3.00
S5BCONC        T       RW-CONC              1.00
S5BEXC         T       RW-EWK               1.00
S5BFILL        T       RW-EWK               7.00
S5BFW          T       RW-FW                2.00
S5BFWS         T       RW-FW                1.00
S5BRF          T       RW-RF                4.00
S5UNITS        T       RW-UNITS             3.00
S6BCONC        T       RW-CONC              1.00
S6BEXC         T       RW-EW                1.00
S6BFILL        T       RW-EWK               7.00
S6BFW          T       RW-FW                2.00
S6BFWS         T       RW-FW                1.00
S6BRF          T       RW-RF                4.00
S6UNITS        T       RW-UNITS             3.00

 OPTIMISTIC duration factor:
 PESSIMISTIC duration factor:
```

Figure 6.30 Optimistic Duration Factor Input Screen

Typing the Optimistic Duration factor and hitting the <CR> key will result in the display of the screen shown in **Figure 6.31**, indicating the input of the Pessimistic Duration factor and showing the range of the factor in the data input line.



Tasks with VARIABLE durations

Factor (1 ≤ f < 1000) : [_]

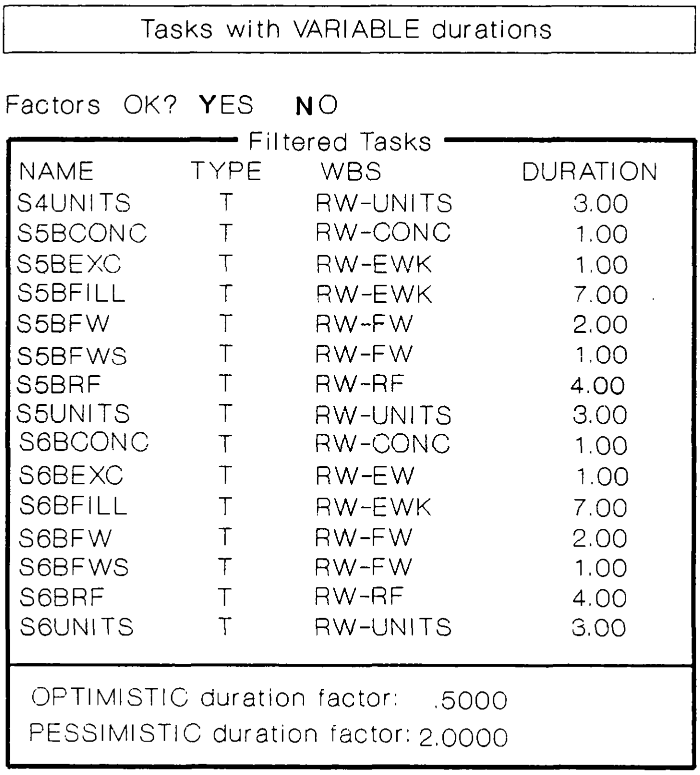
Filtered Tasks

NAME	TYPE	WBS	DURATION
S4UNITS	T	RW-UNITS	3.00
S5BCONC	T	RW-CONC	1.00
S5BEXC	T	RW-EWK	1.00
S5BFILL	T	RW-EWK	7.00
S5BFW	T	RW-FW	2.00
S5BFWS	T	RW-FW	1.00
S5BRF	T	RW-RF	4.00
S5UNITS	T	RW-UNITS	3.00
S6BCONC	T	RW-CONC	1.00
S6BEXC	T	RW-EW	1.00
S6BFILL	T	RW-EWK	7.00
S6BFW	T	RW-FW	2.00
S6BFWS	T	RW-FW	1.00
S6BRF	T	RW-RF	4.00
S6UNITS	T	RW-UNITS	3.00

OPTIMISTIC duration factor: .5000
PESSIMISTIC duration factor:

Figure 6.31 Pessimistic Duration Factor Input Screen

The user is, once again, having the option of confirming the accuracy of the data input as can be seen in the screen display shown in **Figure 6.32**.



Tasks with VARIABLE durations

Factors OK? YES NO

Filtered Tasks

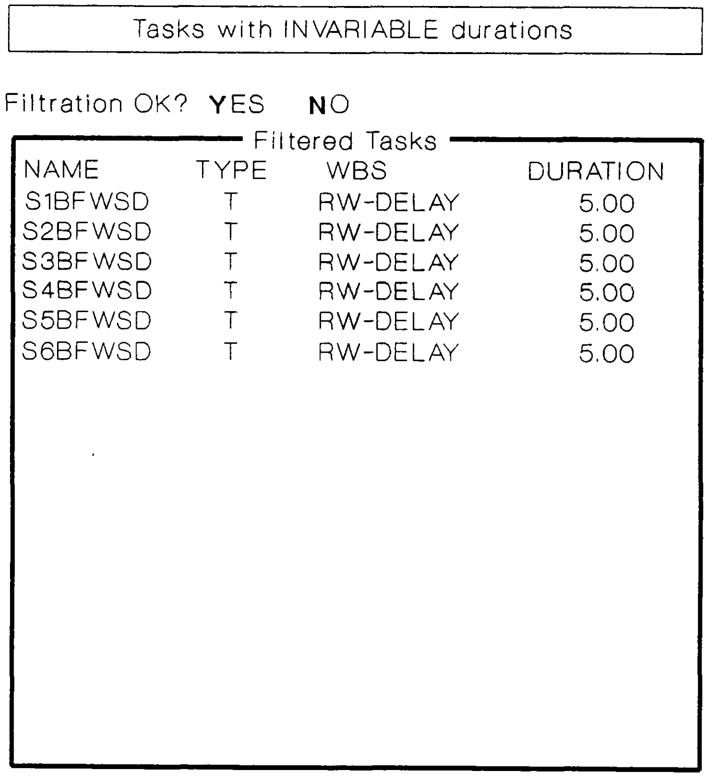
NAME	TYPE	WBS	DURATION
S4UNITS	T	RW-UNITS	3.00
S5BCONC	T	RW-CONC	1.00
S5BEXC	T	RW-EWK	1.00
S5BFILL	T	RW-EWK	7.00
S5BFW	T	RW-FW	2.00
S5BFWS	T	RW-FW	1.00
S5BRF	T	RW-RF	4.00
S5UNITS	T	RW-UNITS	3.00
S6BCONC	T	RW-CONC	1.00
S6BEXC	T	RW-EW	1.00
S6BFILL	T	RW-EWK	7.00
S6BFW	T	RW-FW	2.00
S6BFWS	T	RW-FW	1.00
S6BRF	T	RW-RF	4.00
S6UNITS	T	RW-UNITS	3.00

OPTIMISTIC duration factor: .5000
PESSIMISTIC duration factor: 2.0000

Figure 6.32 Factors Confirmation Screen

6.2.2.5 Input of Tasks with INVARIABLE durations

The selection of the tasks with INVARIABLE durations, for example, concrete curing, consolidation of an embankment under surcharge etc.(see **Section 5.3.(b))**, is carried out in the same manner as for the selection of tasks with variable durations, but the input of optimistic and pessimistic durations factors is not required. The result of the selection exercise is shown in **Figure 6.33**.



Tasks with INVARIABLE durations

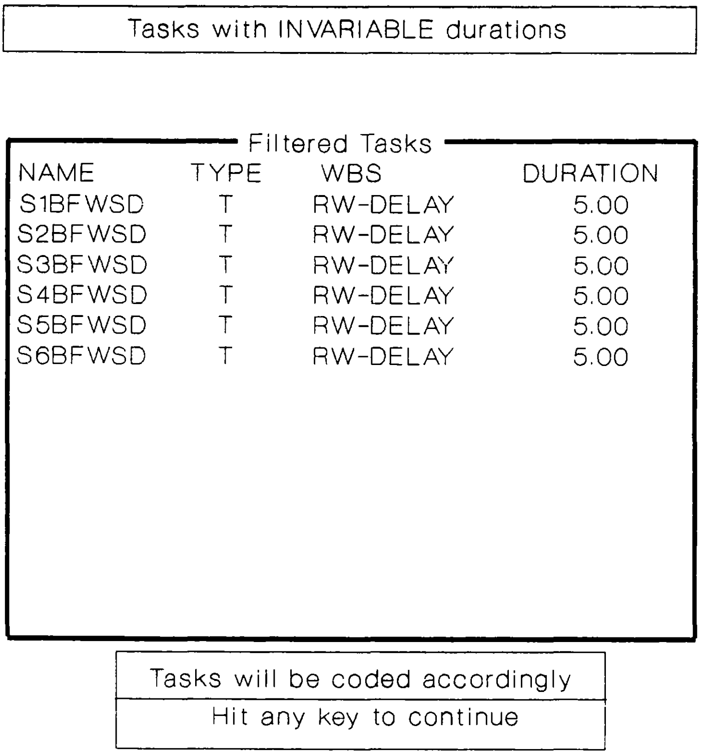
Filtration OK? YES NO

Filtered Tasks

NAME	TYPE	WBS	DURATION
S1BFWSD	T	RW-DELAY	5.00
S2BFWSD	T	RW-DELAY	5.00
S3BFWSD	T	RW-DELAY	5.00
S4BFWSD	T	RW-DELAY	5.00
S5BFWSD	T	RW-DELAY	5.00
S6BFWSD	T	RW-DELAY	5.00

Figure 6.33 Filtered Tasks display screen

Confirming the accuracy of the displayed tasks will result in the display of the screen shown in **Figure 6.34**. The user is required to hit any key and the program will coded all the displayed tasks accordingly.

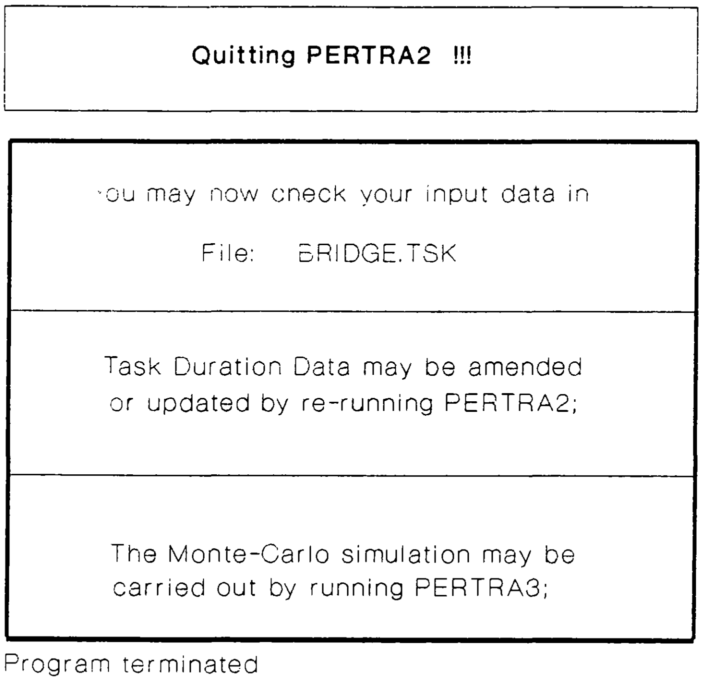


```
Tasks with INVARIABLE durations

                Filtered Tasks
NAME        TYPE     WBS            DURATION
S1BFWSD      T      RW-DELAY          5.00
S2BFWSD      T      RW-DELAY          5.00
S3BFWSD      T      RW-DELAY          5.00
S4BFWSD      T      RW-DELAY          5.00
S5BFWSD      T      RW-DELAY          5.00
S6BFWSD      T      RW-DELAY          5.00

        Tasks will be coded accordingly
           Hit any key to continue
```

Figure 6.34 Coding Screen Display

6.2.2.6 Updating-of Task Data File and Program Termination

Finishing the coding the program automatically will update the Task data file (see **Section 5.3.5**) and the screen shown in **Figure 6.35** will be displayed with a program termination message in the message line.



```
Quitting PERTRA2 !!!

You may now check your input data in

File:    BRIDGE.TSK

Task Duration Data may be amended
or updated by re-running PERTRA2;

The Monte-Carlo simulation may be
carried out by running PERTRA3;

Program terminated
```

Figure 6.35 Program Termination Screen

6.3 PERTRA3

6.3.1 Program Operation

PERTRA3 is considered to be the "number—crunching" program in the suite of the PERTRA programs and its interaction with the user is very limited. For this reason the program was written without the use of the screen displays used in the programs PERTRA1 and PERTRA2.

Program operation is commenced, in the usual manner, by typing the program name **PERTRA3**.

6.3.2 Data Input

6.3.2.1 Random Numbers series Type

The user is required to input the type of random number series to be generated by the program.

If a REPEATABLE series of random numbers is required (see **Ssction 5.4.4.1**), the letter **R** should be typed followed by <CR>. If a NONREPEATABLE series is required, the letter **N** should be typed, followed by <CR>.

In the event of typing any letters other than **R** or **N**, a warning message will be displayed and the correct code should be typed again.

6.3.2.2 Number of Simulations

The next data input required is the number of simulation runs required (see **Section 4.3.1**). The program is capable of performing up to 1000 runs.

Following the input of the above data the program commences the analysis calculations and writes the results in the file **PERTRA3.RES**.

By means of appropriate screen messages, the user is kept abreast of the operations currently being performed by the program.

6.3.3 Data Output

6.3.3.1 One Simulation Run Data Output

If a single simulation run is chosen (see above), the following results will be written to the file PERTRA3.RES.

(a) Project Start Date.

(b) Project Finish Date.

(c) Project Duration (days)

(d) Project Internal Rate of Return (IRR).

(e) Project CAPTIM.

A sample of the results is given in **Appendix G.**

6.3.3.2 Multiple Simulation Runs Output

If more than one simulation run is required, the results will be written into the file PERTRA3.RES as given in the sample in **Appendix H.**

Presenting the results graphically can be carried out using any interactive graphics software. The output shown in **Figures 6.36** to **6.41** was achieved using the software pakage: HARVARD GRAPHICS.

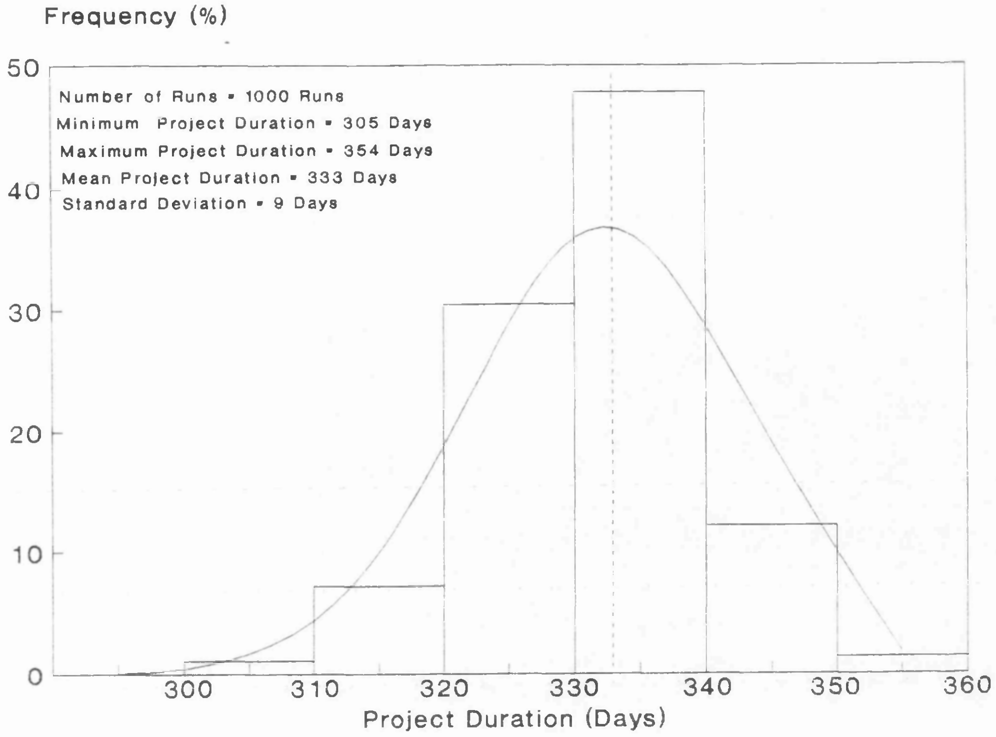


Figure 6.36 Project Durations Frequency Diagram

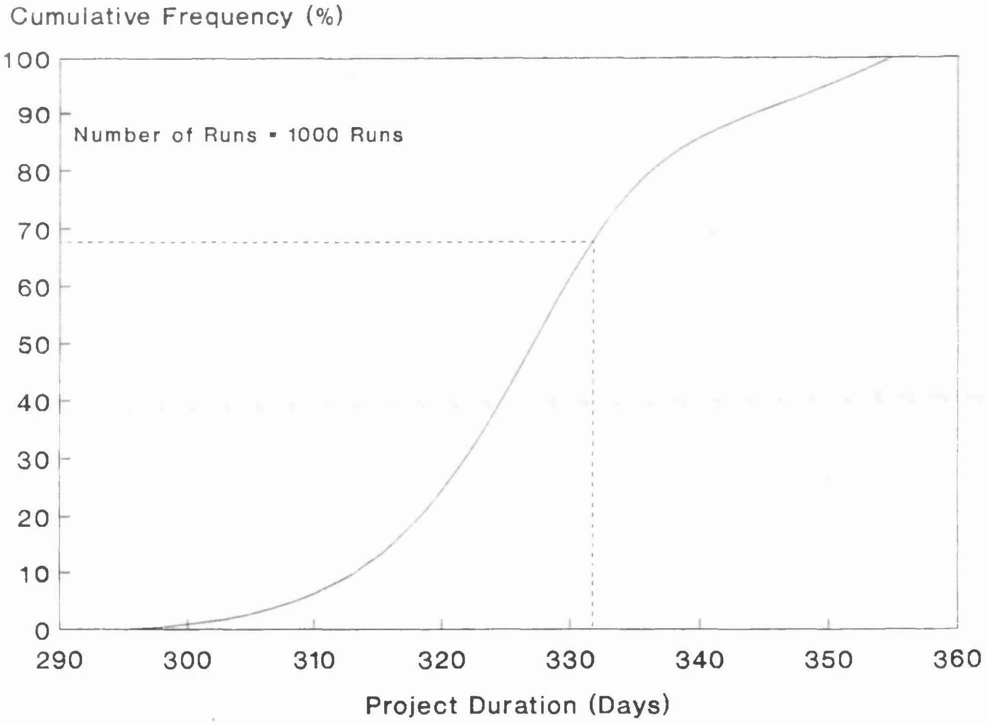


Figure 6.37 Project durations Cumulative Frequency Diagram

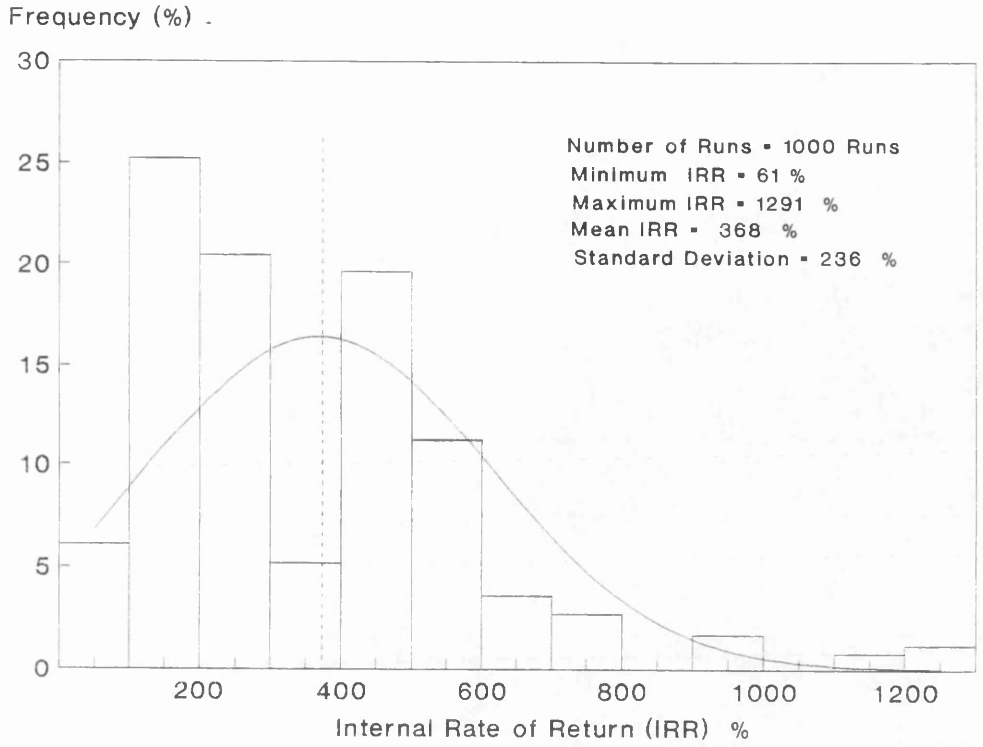


Figure 6.38 Internal Rate of Return (IRR) Frequency Diagram

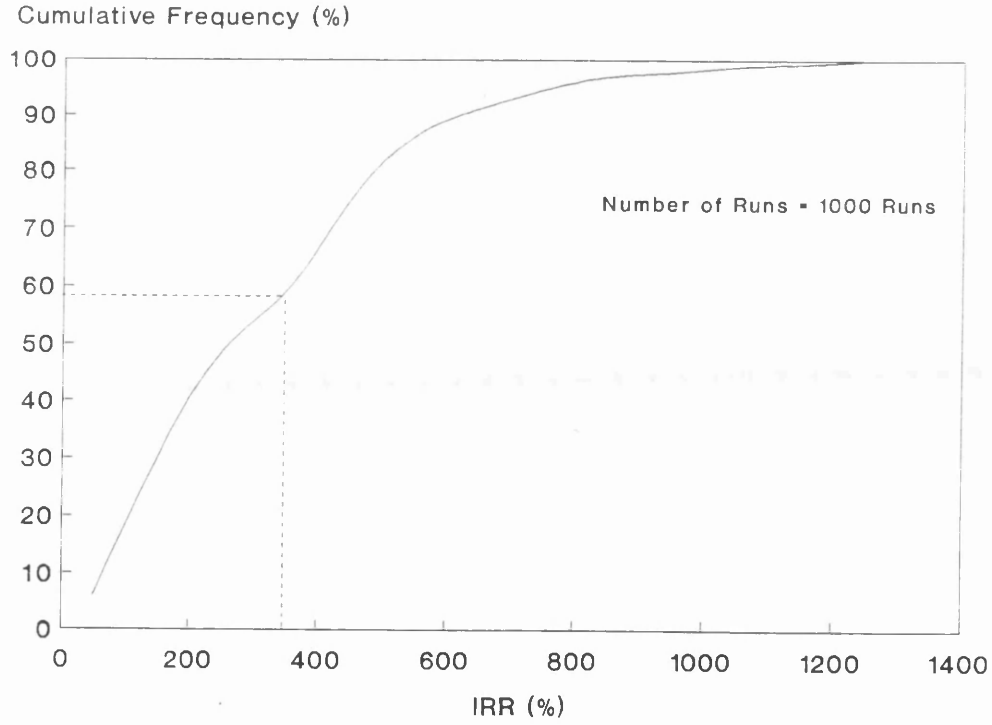


Figure 6.39 IRR Cumulative Frequency Diagram

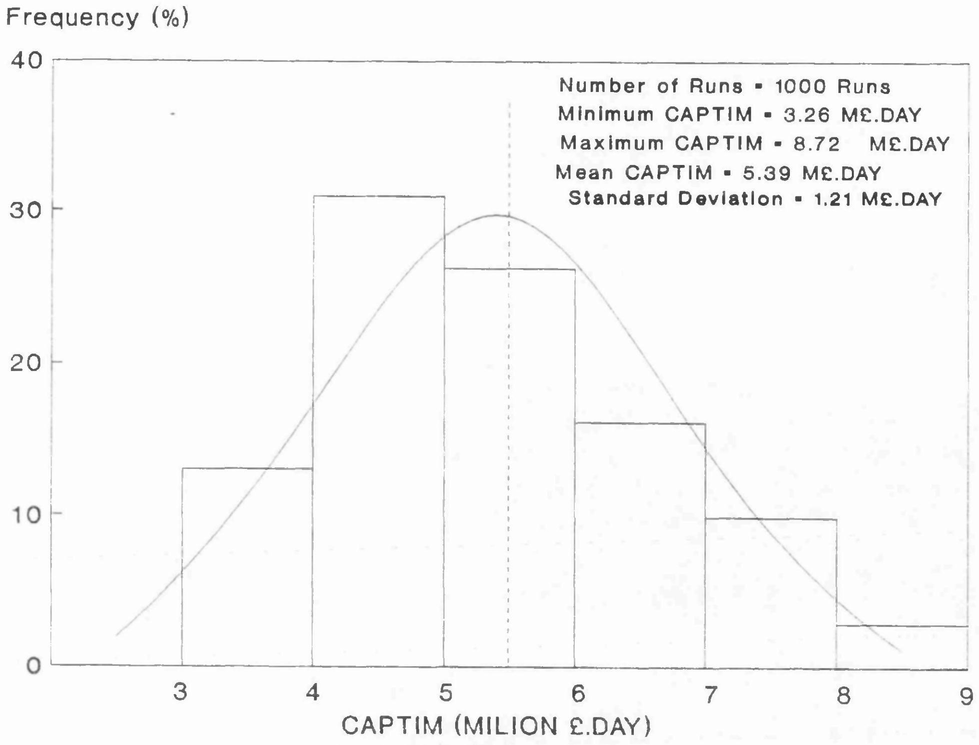


Figure 6.40 CAPTIM Frequency diagram

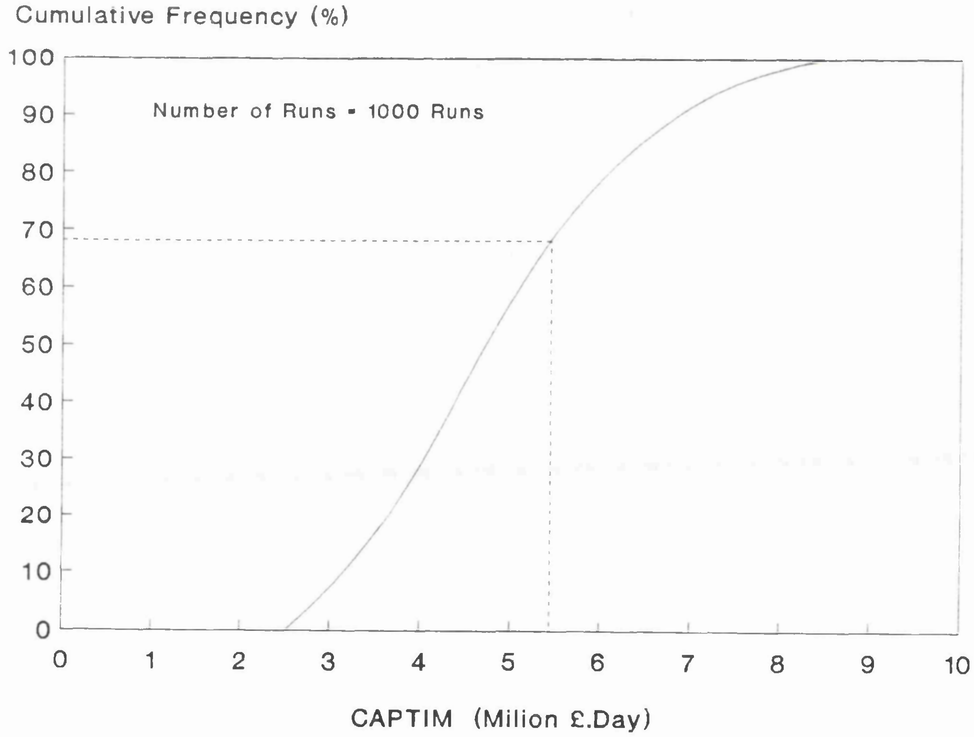


Figure 6.41 CAPTIM Cumulative Frequency Diagram

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

The usual techniques of including the most likely duration in the analysis and producing a single most likely result tends to underestimate the risks in a project. It neglect most of the information that bears in future possibilities. For this reason, risk analysis programs have been developed to report a range of outcomes for the project; including possible worse outcomes as well as desired profitable ones.

By their very nature, risk analysis programs require substantial computing power. In the past, this restricted their use to large organisations and projects. However, the last decade has seen rapid improvements in the capabilities of microcomputers to the extent that it became feasible to develop risk analysis programs for use on personal computers.

The PERTRA programs were therefore developed to quantify the effect of risks on cost and duration of projects. However it should be emphasized that the results are based on estimates to tasks durations (optimistic, pessimistic and most likely) and the accuracy and the confidence in it are very well depend upon the original estimates.

7.2 Recommendations

As the computer results are usually taken to be accurate and the confidence in these results are very high, it should be emphasised here that, especially for the Cost Analysis case, this results should be taken as a guide and the following should be taken into consideration to increase the confidence in these results:

1. The analysis is based <u>only</u> on <u>logical</u> linkage between tasks and does not take into consideration the rescheduling of tasks, for the purpose of maximising the efficiency of resource usage, which usually takes place in reality, therefore a mechanism to consider that should be incorporated in the development of the analysis routines.

2. The costs used by the program are based on the tasks <u>demand</u> for resource while in reality the majority of resource costs are based on the supply thereof. It is there fore recommended that such a facility be included in the program.

3. The inclusion of project close–down and seasonal variations (see Section 2.5.2 and Section 2.5.5).

4. The inclusion of a facility to consider the project maintenance period in the cost analysis subroutines.

5. A facility to take account of the retention money should be included.

6. A facility to consider the effects of the inflation and particularly the rise in the prices during the contract period is to be developed.

7. FORTRAN 77 is a programming language recommended for the "number– crunching" programs, for example PERTRA3, but not for the programs which deals extensively with character operations as the case of PERTRA1 and PERTRA2.

8. FORTRAN programs restricting the user of specifying the arrays dimensions before compiling an linking this limits the program only to the project in hand, therefore, a second thought about the programming language is to be considered.

9. In the case of using the FORTRAN language the compilation of the program PERTRA3 using the compiler SALFORD is recommended because of the advantages of using the graphics support of this compiler which allows the presentation of the results in graphics, on the screen or in hard copy, directly. The reason for not using the above mentioned compiler is its availability at a later stage of this project.

REFERENCES

1. C. CHAPMAN, *Large Engineering Project Risk Analysis*, Transactions on Engineering Management, IEEE, Vol. EM-26, No. 3, August 1979

2. DAVID CORMICAN, *Construction Management Planning and Finance*, Construction Press, 1985

3. BURTON V. DEAN, *Project Management Methods and Studies*, Elsevier science publishers B. V., 1985

4. HARRIS & McCAFFER, *Modern Construction Management*, Collins, 1983

5. R. HAYES, J. PERRY, P. THOMPSON, G. WILLMER, *Risk Management in Engineering Construction*, Implecation for Managers, SERC, December 1986

6. M. J. JACKSON, *Computers in Construction Planning and Control*, Allen, Unwin (Publishers) Ltd., 1986

7. J. KENEDY & A. NEVILL, *Basic Statistical Methods for Engineering and Scientists*, International Textbook Company, Sept. 1968

8. D. J. LEECH, *Economics and Financial Studies for Engineers*, Ellis Horwood Ltd., 1982

9. PERRY AND HAYES, *Risk and its Management in Construction Projects*, Proc. Instn. Civ. Engrs, Part 1, 1985,78, Jun., 499-521

10. K. J. LOCKYER, *An Introduction to Critical Path Analysis*, Pitman Ltd., 1969

11. R. OXLEY and J. POSKITT, *Management Techniques Applied to the Construction Industry*, 4th ed., Collins, 1986

12. ROY PILCHER, *Principles of Construction Management*, Third Edition, Mc Graw– Hill Book Company,

13. T. STRICKLAND and E. GRADY, *Capital Budgeting in Project Evaluation*, Part: 1,2, Hydrocarbon Processing, March, April 1981

14. PETER THOMPSON, *Organization and Economics of Construction*, Mc Graw– Hill, 1992

15. R. WALPOLE, *Probability and Statistics for Engineers and Scientists*, Macmillan, Inc., 1985

16. RICHARD E. WESTNEY, *Managing The Engineering and Construction of Small Projects*, MARCEL DEKKER INC., 1985

17. K. J. WILLIAMS, *Planning/ Work Study Notes for Students*, Department of Civil Engineering, University of Glasgow, 1989

18. G. WILLMER, *Time and Risk Cost Analysis*, Department of Civil Engineering, UMIST, Manchester, 1989

Appendix A: Contents of PMA–Generated Export File.

1. General Format.

The general order of records contained within the export file is as follows:

```
Header Record

Project Record

Task Record                          ┐
Output Record                        │
Task Resource Demand Record 1        │
Task Resource Demand Record 2        ├   TASK 1
    .     .     .     .              │
    .     .     .     .              │
Task Resource Demand Record N        ┘

Task Record                          ┐
Output Record                        │
Task Resource Demand Record 1        │
Task Resource Demand Record 2        ├   TASK 2
    .     .     .     .              │
    .     .     .     .              │
Task Resource Demand Record N        ┘

    .     .     .     .

    .     .     .     .

Task Record                          ┐
Output Record                        │
Task Resource Demand Record 1        │
Task Resource Demand Record 2        ├   TASK N
    .     .     .     .              │
    .     .     .     .              │
Task Resource Demand Record N        ┘

Link Record (Link 1)
Link Record (Link 2)
    .     .     .
    .     .     .
    .     .     .
    .     .     .
    .     .     .
    .     .     .
Link Record (Link N)
```

Resource Definition Record
Resource Demand Record 1
Resource Demand Record 2
. . . .
. . . .
Resource Demand Record N RESOURCE 1
Sub– resource Record 1
Sub– resource Record 2
. . . .
. . . .
Sub– resource Record N

Resource Definition Record
Resource Demand Record 1
Resource Demand Record 2
. . . .
. . . .
Resource Demand Record N RESOURCE 2
Sub– resource Record 1
Sub– resource Record 2
. . . .
. . . .
Sub– resource Record N

. . .

. . .

Resource Definition Record
Resource Demand Record 1
Resource Demand Record 2
. . . .
. . . .
Resource Demand Record N RESOURCE N
Sub– resource Record 1
Sub– resource Record 2
. . . .
. . . .
Sub– resource Record N

Calendar Record
Working Week Record CALENDAR 1

Calendar Record
Working Week Record CALENDAR 2

. . .

. . .

Calendar Record
Working Week Record CALENDAR N

2. Record Format

The contents and format of the data contained within each type of record are as follows, with data required by PERTRA programs being annotated *:

Header Records

H	Record Type
Plan Name	Up to 9 characters
Left Header *	Quote delimited
Right Header *	Quote delimited
Work Breakdown	Up to 12 characters
Plan Type	Single character (A = arrow, P = precedence)
Units Number	(Unused)
Calendar Name	Up to 8 characters
Codes Name	Up to 8 characters (Unused)
Subtime Units	Number between 1 and 80
Version Number *	Number between 0 and 999
Plan Start Date	Date string ("DD/MMM/YY[:SS]")
Plan Finish Date	Date string ("DD/MMM/YY[:SS]")
Time– now	Date string ("DD/MMM/YY[:SS]")

(Note: In date strings [:ss] refers to subtime units if they are in use within the plan. The brackets[ ] refer to an optional item and are not present in the field.)

Example of Header Record:

```
H,"BRIDGE","Construction Management IV","Model Solution","N– NN– ",P,0,"DAILY"
,"",1,0," 7/JUN/92","12/APR/93"," 7/JUN/92
```

Project – defines each sub– project within the plan.

P	Record type
Project Name *	Up to 9 characters
Parent Name	Up to 9 characters
Project Header *	Quote delimited
Project Start Date *	Date string ("DD/MMM/YY[:SS]")
Project Finish Date	Date string ("DD/MMM/YY[:SS]")

Example of Project Record:

```
P,"BRIDGE","","Road over Road Bridge"," 7/JUN/92","12/APR/93"
```

Task – defines a task as an input by the user.

T/M/K/X/S *	Record type – Task, Milestone, Hammock, Sub– project
Project Name	Up to 8 characters
Task Name *	Up to 9 characters – Quote delimited
Task Description	Quote delimited
Work Breakdown	
Structure *	Up to 12 characters

Original Duration *	Unsigned integer
Remaining Duration	Unsigned integer
Preferred Start	Date String ("DD/MMM/YY[:SS]")
Scheduled Start	Date String ("DD/MMM/YY[:SS]")
Scheduled Finish	Date String ("DD/MMM/YY[:SS]")
Actual Start	Date String ("DD/MMM/YY[:SS]")
Actual Finish	Date String ("DD/MMM/YY[:SS]")
Target Start	Date String ("DD/MMM/YY[:SS]")
Target Finish	Date String ("DD/MMM/YY[:SS]")
Calendar Name *	Up to 8 characters
Percent Complete	Unsigned integer as a string

Note: Percent complete only appears if percent complete is set 'Enter'

Example of Task Records:

```
T,"BRIDGE","CLEAR1 ","Clear Site","A– GN–CLEAR","0","2"," 7/JUN/92"," 7/JUN/92
," 8/JUN/92"," 7/JUN/92"," 8/JUN/92"," 7/JUN/92"," 8/JUN/92","DAILY"
```

Output Record – defines the schedule of the task as calculated by PERTMASTER Advance

Z	Record type
Project Name	Up to 9 characters
Task Name	Up to 9 characters – Quote delimited
Calculated Preferred Start	Date String ("DD/MMM/YY[:SS]")

Preferred Finish	Date String ("DD/MMM/YY[:SS]")
Early Start	Date String ("DD/MMM/YY[:SS]")
Early Finish	Date String ("DD/MMM/YY[:SS]")
Late Start	Date String ("DD/MMM/YY[:SS]")
Late Finish	Date String ("DD/MMM/YY[:SS]")
Total Preferred Start Float	Long integer as string – Quote delimited
Total Preferred End Float	Long integer as string – Quote delimited
Free Preferred Start Float	Long integer as string – Quote delimited
Free Preferred End Float	Long integer as string – Quote delimited
Derived Duration	Long integer as string – Quote delimited
Percent Complete	Unsigned integer as string – Quote delimited
Start Task *	Integer as string – 0 = No, 1 = Yes
Finish Task *	Integer as string – 0 = No, 1 = Yes

Example of Output Record:

```
Z,"BRIDGE","CLEAR1"," 7/JUN/92"," 8/JUN/92"," 7/JUN/92"," 8/JUN/92"," 8/JUN/92
"," 9/JUN/92","        1","       1","        0","       0",       2","0      "
,"0","0"
```

Task Resource Demand – gives details of the demand for resources made by the task as defined by the user.

D	Record type
Project Name	Up to 8 characters
Task Name	Up to 9 characters – Quote delimited
Resource Code *	Up to 3 characters
Resource Demand *	Double precision number

Example of Task Resource Demand Records:

```
D,"BRIDGE","CLEAR1","DRT",1
D,"BRIDGE","CLEAR1","LAB",2
D,"BRIDGE","CLEAR1","TIP",1
```

Link – gives details of precedence logic links as defined by the user

L	Record type
Project Name	Up to 9 characters
Preceding Task *	Name 9 characters
Succeeding Task *	Name up to 9 characters
Link Type *	3 characters – See Note below
Link Duration *	Unsigned integer

Note: The link type appears in the file as "S–S, "F–S", "S–F", "F–F".

Example of Link Records:

```
L,"BRIDGE","CLEAR1   ","SETUP    ","F-S","0"
L,"BRIDGE","CLEAR2   ","FINISH   ","F-S","0"
L,"BRIDGE","CMPRESS  ","SSEPCO   ","F-F","0"
```

Resource Definition – defines the resources in use on the plan as defined by the user

R	Record type
Resource Code *	Up to 3 characters
Resource Name	Quote delimited
Resource Type *	Single character (D/F/B/S) – See Note below
Storage Type	Number describing whether limited, unlimited, actual,cost or budget
Resource Calendar	Up to 8 characters

Note: Character corresponds respectively to an integer value. D=Daily, F = Front loading, B = Back loading, and S = Spread.

Example of Resource Definition Records:

```
R,"621 ","Cat 621 Scraper",D,0,"ALLDAYS"
```

Resource Supply – defines the availability for each resource as defined by the user.

S	Record type
Resource Code	Up to 3 characters
Time Offset	Date string("DD/MMM/YY[:SS]")
Supply Quantity	Type defined by 'storage type' above

Subresource Supply – defines demands made by one resource for another in a resource hierarchy as defined by the user.

U	Record type
Resource Code *	Up to 3 characters
Subresource Code *	Up to 3 characters
Resource Demand *	Double precision number

Example of Sub– resource Supply

```
U,"BFG","B35",1
U,"BFG","JCB",2
U,"BFG","LAB",4
U,"BFG","TIP",2
```

Resource Demand – defines the demand for each resource as calculated by PERTMASTER ADVANCE.

Q	Record type
Resource Code	Up to 3 characters
Start of Demand	Date String ("DD/MMM/YY[:SS]")
End of Demand	Date String ("DD/MMM/YY[:SS]")
Demand Level	Double precision number

Example of Resource Demand Records:

```
Q,"BFG","20/JUN/91","20/JUN/91",1
Q,"BFG"," 4/JUL/91"," 4/JUL/91",1
Q,"BFG"," 3/SEP/91"," 3/SEP/91",2
```

Calendar – gives details of the calendars in use on the plan as defined by the user.

C	Record type
Calendar Name *	Up to 9 characters

Example of Calendar Records:

```
C,"ALLDAYS"
C,"DAILY"
```

Working Week – defines the typical working week.

W	Record type
Calendar Name	Up to 9 characters
Working Week spec.	Two integers separated by commas describing start to finish of a working week spec. in subunits.

Example of Working Week Record:

W,DAILY,1,5

Holiday – defines a holiday period within the last mentioned calendar as defined by the user. There will be a list of these records for each calendar.

O	Record type
Calendar Name	Up to 8 characters
Start Date	Date String ("DD/MMM/YY[:SS]")
End Date	Date String ("DD/MMM/YY[:SS]")

Working Day – Defines a work day period within a calendar as defined by the user. There will normally be a list of these records for each calendar.

Y	Record type
Calendar Name	Up to 8 characters
Start Date	Date string ("DD/MMM/YY[:SS]")
End Date	Date string ("DD/MMM/YY[:SS]")

APPENDIX B: Samples of Data Files Generated by PERTRA1

Appendix B.1: Sample of the general dat writen to the file FLNAME.GEN

```
BRIDGE   0                          (1)
Case Study No. 1                    (2)
Road over Road Bridge               (3)
Construction Management IV          (4)
15/APR/91                           (5)
```

Definiton of rows:

(1): Project Nmae; Project Version.

(2): Left Header.

(3): Project description.

(4): Right Header.

(5): Project Start Date.

Appendix B.2: Sample of the Task Data writen to the File FLNAME.TSK

```
 274      (Number of tasks in the project)

(1)  (2)          (3)         (4) (5) (6)  (7)    (8)     (9)

 T CLEAR1    GN-CLEAR        5   1   0   0.00   2.00    0.00

 T CLEAR2    GN-CLEAR        5   1   0   0.00   5.00    0.00

 H CMPRESS   GN-CMPRESS      5   1   0   0.00 136.00    0.00

 T CPC1CO    PR-CONC         5   1   0   0.00   1.00    0.00

 T CPC1D1    PR-DELAY        7   1   0   0.00   1.00    0.00

     .    .    .    .    .    .    .    .    .    .

     .    .    .    .    .    .    .    .    .    .
```

Defintion of Columns are:

(1): Task Type.

(2): Task Name.

(3): Task Work Breakdown Structure.

(4): Number of Working Days/Week

(5): Task Code (Start, Finish or Normal Task)

(6): Dummy Array to be Used Later for Variable/Nonvariable duration Task Codes.

(7): Dummy array to be Used Later for Optimistic Durations.

(8): Task Duration (Most Likely Duration).

(9): Dummy Array to be Used Later for Pessimistic Durations.

Appendix B.3: Sample of Link Data writen to the File FLNAME.LNK

437 (Number of links)

```
(1)  (2) (3)   (4)  (5)  (6) (7)   (8)
268    1  1    .00  158   40  3    .00
268  158  1    .00    2   40  1    .00
  1  209  1    .00  268  158  1    .00
158   40  3    .00  270    2  1    .00
209   49  1    .00   96    2  1    .00
```

Defintion of Columns:

(1): Source Task Number (Forward Pass).

(2): Target Task Number (Forward Pass).

(3): Link Type (Forward Pass).

(4): Link Duration (Forward Pass).

Columns 5 to 8 is the same as the above, but, for the Backward pass.

Appendix B.4: Sample of the Cost Data writen to the file FLNAME.CST

```
(1) (2) (3)  (4)

XLC  W   0    0
XMC  M   M    1
XPC  M   M    1
XPF  M   M    1
XPB  M   M    1
XSC  M   M    1
XOD  M   0    0
XOF  M   0    0
XTP  M   M    1
```

The above four columns defintions is as follows:

(1): Cost component

(2): Payment Interval (W= Weeks, M= Months)

(3): Delay of payment (D= Days, W= Weeks, M= Months, 0= No delay)

(4): Number of Months or Weeks of delay.

```
  (1)      (2)     (3)      (4)     (5)     (6)     (7)     (8)    (9)

143.00     .00  200.00     .00     .00     .00     .00     .00  797.06
238.00     .00  200.00     .00     .00     .00     .00     .00     .00
   .00     .00   40.00     .00     .00     .00     .00     .00     .00
145.00  209.79     .00     .00     .00     .00     .00     .00  381.19
   .00     .00     .00     .00     .00     .00     .00     .00     .00
```

Definition of the above columns is as follows:

(1): Labour Cost.

(2): Material Cost.

(3): Daily Plant Cost.

(4): Front– end Plant Cost.

(5): Back– end Plant Cost.

(6): Subcontract Cost.

(7): Daily Overheads Cost.

(8): Fron– end Overheads Cost.

(9): Task Price.

APPENDIX C: Program PERTRA1 and Subroutines

Appendix C.1 Program PERTRA1

```
c*********************************************************
c**                                                     **
c**                Program PERTRA1.FOR                  **
c**                                                     **
c** Reads project data from user-specified export file  **
c**      FLNAME.EXP generated by PERTMASTER ADVANCE     **
c**                                                     **
c** Writes selected data to files for subsequent use by **
c** Risk Analysis Programs PERTRA2.FOR and PERTRA3.FOR  **
c**                                                     **
c**              GENERAL data: FLNAME.GEN               **
c**              TASK data:    FLNAME.TSK               **
c**              LINK data:    FLNAME.LNK               **
c**              COST data:    FLNAME.CST               **
c**                                                     **
c** Writes FLNAME to file PERTRA.FLS for subsequent use **
c**                                                     **
c*********************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400),TRES(400,10),
     &TRESQ(400,10),TCAL(400),NTRES(400),SFCODE(400),TWK(400),
     &CTRES(400,10),TCOST(400,9),TWBS(400)
      COMMON/LINKS/LFROM(500,2),LTO(500,2),LTYPE(500,2),
     &LDURN(500,2)
      COMMON/RES/RNAME(100),RTYPE(100),RCOST(100)
      COMMON/SUBRES/CRTOP(110),CRSUB(110),RSUBQ(110),RTOP(110),
     &RSUB(110)
      COMMON/CALS/CAL(5),ICWK(5)
      COMMON/PROJ/PHEAD(3),PVER,PNAME,PSTART
      COMMON/COSTS/CCODE(9),CPAY(9),CARR(9),CARRQ(9)
      COMMON/RECORD/REC,ICOMMA(16)
      COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
C
      CHARACTER*1 TTYPE,RTYPE,C1,PVER,CPAY,CARR
      CHARACTER*3 RNAME,CTRES,CRTOP,CRSUB,RCODE,CCODE
      CHARACTER*8 CAL,TCAL,PNAME
      CHARACTER*9 TNAME,PSTART
      CHARACTER*12 FLNAME,TWBS
      CHARACTER*30 PHEAD
      CHARACTER*185 REC
C
      REAL LDURN
      INTEGER SFCODE,TWK,TRES,RTOP,RSUB,CARRQ
C
C...Set TEXT mode
C
      CALL SETMODE(0)
```

```
C
C...Display introduction; Input file name from keyboard
C
      CALL HEADER(FLNAME,NCHRFN)
C
C...Set maximum value of counters according to dimensions
C...Note: Dimensions mut be exactly divisible into 64K
C
      ITMAX=400
      ILMAX=500
      IRMAX=100
      IUMAX=110
      ICMAX=5
      IDMAX=10
C
C...Check adequacy of dimensions
C
      WRITE(6,28)
   28 FORMAT(/,/,' Carrying out preliminary check on dimensions')
      CALL DIMCHK(FLNAME)
      ICODE=0
      WRITE(6,30)IT
   30 FORMAT(/,' No. of TASKS: ',I13)
      IF(IT.GT.ITMAX)THEN
        WRITE(6,40)
   40   FORMAT(' Dimensions exceeded !!')
        IDIM=(IT-MOD(IT,50))+50
        WRITE(6,50)IDIM
   50   FORMAT(' Increase dimensions to ',I3,' and re-compile')
        ICODE=1
      ENDIF
      WRITE(6,60)IDM
   60 FORMAT(/,' Max. No. of TR DEMANDS: ',I3)
      IF(IDM.GT.IDMAX)THEN
        WRITE(6,40)
        IDIM=(IDM-MOD(IDM,5))+5
        WRITE(6,50)IDIM
        ICODE=1
      ENDIF
      WRITE(6,70)IL
   70 FORMAT(/,' No. of LINKS: ',I13)
      IF(IL.GT.ILMAX)THEN
        WRITE(6,40)
        IDIM=(IL-MOD(IL,50))+50
        WRITE(6,50)IDIM
        ICODE=1
      ENDIF
      WRITE(6,80)IR
   80 FORMAT(/,' No. of RESOURCES: ',I9)
      IF(IR.GT.IRMAX)THEN
        WRITE(6,40)
        IDIM=(IR-MOD(IR,10))+10
        WRITE(6,50)IDIM
        ICODE=1
      ENDIF
      WRITE(6,90)IU
  90  FORMAT(/,' No. of SUB-RESOURCES: ',I5)
      IF(IU.GT.IUMAX)THEN
        WRITE(6,40)
        IDIM=(IU-MOD(IU,10))+10
        WRITE(6,50)IDIM
        ICODE=1
      ENDIF
      WRITE(6,100)IC
  100 FORMAT(/,' No. of CALENDARS: ',I9)
      IF(IC.GT.ICMAX)THEN
        WRITE(6,40)
        IDIM=(IC-MOD(IC,5))+5
        WRITE(6,50)IDIM
        ICODE=1
      ENDIF
      IF(ICODE.LT.1)THEN
        WRITE(6,102)
  102   FORMAT(/,/,' ALL DIMENSIONS OK !!')
      ELSE
        GO TO 999
      ENDIF
C
C...Read data from FLNAME
C
      CALL READEXP(FLNAME)
C
C...Determine working week ICWK for each calendar
C
      CALL CALWKS
C
C...Convert TRD resource names into resource indices
C
      DO 410 I=1,IT
        IF(NTRES(I).LT.1)GO TO 410
        DO 400 K=1,NTRES(I)
          DO 390 L=1,IR
            IF(CTRES(I,K).EQ.RNAME(L))THEN
              TRES(I,K)=L
              GO TO 400
            ENDIF
  390       CONTINUE
  400     CONTINUE
  410   CONTINUE
C
C...Convert SR resource names into resource indices
C
      DO 450 I=1,IU
        DO 420 J=1,IR
          IF(CRTOP(I).EQ.RNAME(J))THEN
            RTOP(I)=J
            GO TO 430
          ENDIF
  420     CONTINUE
  430   DO 440 J=1,IR
```

```
          IF(CRSUB(I).EQ.RNAME(J))THEN
            RSUB(I)=J
            GO TO 450
          ENDIF
  440     CONTINUE
  450   CONTINUE
C
C...Calculate cost components for each task
C
      CALL TCOSTS
C
C...Define durations of NON-ZERO S-S LINKS as fractions of
C...SOURCE TASK durations;
C
      DO 600 I=1,IL
        IF(LTYPE(I,1).EQ.2.AND.LDURN(I,1).NE.0.0)THEN
          J=LFROM(I,1)
          LDURN(I,1)=LDURN(I,1)/TDURN(J)
        ENDIF
600     CONTINUE
C
C...Sort links into FORWARD/BACKWARD PASS order
C
      CALL LSORT
C
C...Write data to datafiles; display concluding message;
C
      CALL QUIT(FLNAME,NCHRFN)
  999 END
```

Appendix C.2: Subroutine HEADER

```
      SUBROUTINE HEADER(FLNAME,NCHR)
c***************************************************
c**                                               **
c**              Subroutine HEADER                **
c**                                               **
c**   Introductory display for program PERTRA1    **
c**                                               **
c** Keyboard input of PMA export file name FLNAME **
c**                                               **
c***************************************************
c
      CHARACTER*1 CHR1
      CHARACTER*12 FLNAME
      CHARACTER*42 TABREC(15),BLANK
C
C...Initialise display table strings
C
      DO 20 I=1,15
        TABREC(I)(1:1)=CHAR(186)
        TABREC(I)(42:42)=CHAR(186)
        DO 10 J=2,41
          TABREC(I)(J:J)=' '
  10      CONTINUE
  20    CONTINUE
      BLANK=TABREC(1)
      BLANK(1:1)=' '
      BLANK(42:42)=' '
      TABREC(1)(1:1)=CHAR(201)
      TABREC(1)(42:42)=CHAR(187)
      TABREC(5)(1:1)=CHAR(204)
      TABREC(5)(42:42)=CHAR(185)
      TABREC(11)=TABREC(5)
      TABREC(15)(1:1)=CHAR(200)
      TABREC(15)(42:42)=CHAR(188)
      DO 30 J=2,41
        TABREC(1)(J:J)=CHAR(205)
        TABREC(5)(J:J)=CHAR(205)
        TABREC(11)(J:J)=CHAR(205)
        TABREC(15)(J:J)=CHAR(205)
  30    CONTINUE
      TABREC(3)(14:28)='Program PERTRA1'
      TABREC(7)(3:40)='Interactive initialisation program for'
      TABREC(8)(3:40)='the Monte-Carlo simulation carried out'
      TABREC(9)(13:30)='by Program PERTRA3'
      TABREC(13)(8:23)='PMA Export File:'
C
C...Specify option highlight delay
C
      NDEL=30
```

```
C
C...Define table position
C
      NLTL=6
      NCTL=20
      NLTOP=NLTL-1
      NCTOP=NCTL
C
C...Clear screen; draw table
C
      CALL SCLEAR
      CALL TXTCLR(15,1)
      DO 40 I=1,15
        CALL TMOVETO((NLTL+I-1),NCTL)
        IF(I.LT.2.OR.I.GT.4)THEN
          CALL OUTEXT(TABREC(I))
        ELSE
          CALL OUTEXT(TABREC(I)(1:1))
          CALL TXTCLR(0,6)
          CALL OUTEXT(TABREC(I)(2:41))
          CALL TXTCLR(15,1)
          CALL OUTEXT(TABREC(I)(42:42))
        ENDIF
  40    CONTINUE
C
C...Display FILE NAME input line
C
  50  CALL TXTCLR(15,0)
      CALL TMOVETO(NLTOP,NCTOP)
      CALL OUTEXT(BLANK)
      CALL TMOVETO(NLTOP,NCTOP)
      CALL OUTEXT('File name (no extension): [        ]')
C
C...Read file name from keyboard;
C...convert to upper case if required;
C
      NCHR=8
      FLNAME='            '
      CALL CREAD(NLTOP,(NCTOP+27),FLNAME,NCHR)
      CALL CHRCNV(FLNAME)
C
C...Remove unwanted spaces from FLNAME
C
      FLNAME((NCHR+1):(NCHR+4))='.EXP'
C
C...Update table
C
      CALL TXTCLR(7,1)
      CALL TMOVETO((NLTL+12),(NCTL+24))
      CALL OUTEXT(FLNAME)
```

```
C
C...Display CONTINUATION line; highlight
C...first character of options;
C
      CALL TXTCLR(15,0)
      CALL TMOVETO(NLTOP,NCTOP)
      CALL OUTEXT(BLANK)
      CALL TMOVETO(NLTOP,NCTOP)
      CALL OUTEXT('OK?  YES  NO')
      CALL TXTCLR(4,0)
      CALL TMOVETO(NLTOP,(NCTOP+5))
      CALL OUTEXT('Y')
      CALL TMOVETO(NLTOP,(NCTOP+10))
      CALL OUTEXT('N')
C
C...Hide cursor
C
      CALL TXTCLR(0,0)
      CALL TMOVETO(25,70)
      CALL OUTEXT(' ')
      CALL TMOVETO(25,70)
C
C...Read option from keyboard; convert to upper case if required;
C
  60  CALL KBREAD(0,CHR1)
      CALL CHRCNV(CHR1)
      IF(CHR1.EQ.'Y')THEN
C
C...Option is CONTINUE; highlight option; exit routine
C
        CALL TXTCLR(0,7)
        DO 70 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+5))
          CALL OUTEXT('YES')
  70      CONTINUE
        GO TO 999
      ELSEIF(CHR1.EQ.'N')THEN
C
C...Option is DISCONTINUE; highlight option;
C...out table entry; Input correct file name;
C
        CALL TXTCLR(0,7)
        DO 80 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+10))
          CALL OUTEXT('NO')
  80      CONTINUE
        CALL TXTCLR(0,1)
        CALL TMOVETO((NLTL+12),(NCTL+24))
        CALL OUTEXT(BLANK(1:12))
        GO TO 50
      ELSE
```

```
C
C...Inappropriate character; try again
C
          CALL SPCHAR(1)
          GO TO 60
        ENDIF
C
C...Exit routine
C
  999   CALL TXTCLR(7,0)
        CALL SCLEAR
        RETURN
        END
```

Appendix C.3: Subroutine DIMCHK

```
      SUBROUTINE DIMCHK(FLNAME)
c**************************************************************
c**                                                          **
c**                   Subroutine DIMCHK                      **
c**                                                          **
c** Performs check on adequacy of PERTRA1/PERTRA2 dimensions **
c**                                                          **
c**************************************************************
c
      COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
      CHARACTER*1 C1
      CHARACTER*12 FLNAME
C
C...Initialise control statements
C
      ASSIGN 20 TO LABIF
C
C...Zero array counters
C
      IT=0
      IL=0
      IR=0
      IU=0
      IC=0
      IDM=0
C
C...Open file on unit 3
C
      OPEN(UNIT=3,FILE=FLNAME)
C
C...Skip first two records
C
      READ(3,15)C1
      READ(3,15)C1
C
C...Read 1st character of record
C
  10  READ(3,15)C1
  15  FORMAT(A1)
      GO TO LABIF
C
C...Check if TASK record
C
  20  IF(C1.EQ.'T'.OR.C1.EQ.'M'.OR.C1.EQ.'K'.OR.C1.EQ.'S')THEN
C
C...TASK record; Update counter;
C
        IT=IT+1
C
C...Zero TASK RESOURCE DEMAND counter
C
        ID=0
```

```
C
C...Skip OUTPUT record; Read next record
C
        READ(3,15)C1
        GO TO 10
C
C...Check if TASK RESOURCE DEMAND record
C
      ELSEIF(C1.EQ.'D')THEN
C
C...TASK RESOURCE DEMAND record; Update counter
C
        ID=ID+1
        IF(ID.GT.IDM)IDM=ID
        GO TO 10
      ENDIF
C
C...Check if LINK record
C
  30  IF(C1.EQ.'L')THEN
C
C...LINK record; All TASK type records abstracted;
C...Change label LABIF to avoid unneccessary testing
C
        ASSIGN 30 TO LABIF
C
C...Update counter;
C
        IL=IL+1
        GO TO 10
      ENDIF
C
C...Check if RESOURCE record
C
  40  IF(C1.EQ.'R')THEN
C
C...RESOURCE record; All TASK/LINK records abstracted;
C...Change label LABIF to avoid unneccessary testing;
C
        ASSIGN 40 TO LABIF
C
C...Update counter;
C
        IR=IR+1
        GO TO 10
C
C...Check if (unrequired) RESOURCE DEMAND record
C
      ELSEIF(C1.EQ.'Q'.OR.C1.EQ.'S')THEN
C
C...Unrequired record; Read next record
C
        GO TO 10
```

```
C
C...Check if SUB-RESOURCE record
C
      ELSEIF(C1.EQ.'U')THEN
C
C...SUB-RESOURCE record; Update counter
C
        IU=IU+1
        GO TO 10
      ENDIF
C
C...Check if CALENDAR record
C
   50 IF(C1.EQ.'C')THEN
C
C...CALENDAR record; All TASK/LINK/RESOURCE records abstracted;
C...Change label LABIF to avoid unneccessary testing
C
        ASSIGN 50 TO LABIF
C
C...Update counter;
C
        IC=IC+1
        GO TO 10
      ELSE
C
C...All data abstracted; Close file
C
        CLOSE(3)
      ENDIF
C
C...Exit
C
      RETURN
      END
```

Appendix C.4: Subroutine READEXP

```
      SUBROUTINE READEXP(FLNAME)
c*****************************************************
c**                                                 **
c**                Subroutine READEXP               **
c**                                                 **
c** Reads data from user-specified export file FLNAME **
c**                  generated by PMA               **
c**                                                 **
c*****************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400),TRES(400,10),
     &TRESQ(400,10),TCAL(400),NTRES(400),SFCODE(400),TWK(400),
     &CTRES(400,10),TCOST(400,9),TWBS(400)
      COMMON/LINKS/LFROM(500,2),LTO(500,2),LTYPE(500,2),
     &LDURN(500,2)
      COMMON/RES/RNAME(100),RTYPE(100),RCOST(100)
      COMMON/SUBRES/CRTOP(110),CRSUB(110),RSUBQ(110),RTOP(110),
     &RSUB(110)
      COMMON/CALS/CAL(5),ICWK(5)
      COMMON/PROJ/PHEAD(3),PVER,PNAME,PSTART
      COMMON/RECORD/REC,ICOMMA(16)
      COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
C
      CHARACTER*1 TTYPE,RTYPE,C1,PVER
      CHARACTER*3 RNAME,CTRES,CRTOP,CRSUB
      CHARACTER*8 CAL,TCAL,PNAME
      CHARACTER*9 TNAME,PSTART
      CHARACTER*11 STATUS(2)
      CHARACTER*12 FLNAME,TWBS
      CHARACTER*28 TABREC(12)
      CHARACTER*30 PHEAD
      CHARACTER*185 REC
C
      REAL LDURN
      INTEGER SFCODE,TWK,TRES,RTOP,RSUB
C
C...Initialise display table records
C
      DO 510 I=1,12
        TABREC(I)(1:1)=CHAR(186)
        TABREC(I)(28:28)=CHAR(186)
        DO 500 J=2,27
          TABREC(I)(J:J)=' '
  500     CONTINUE
  510   CONTINUE
      TABREC(1)(1:1)=CHAR(201)
      TABREC(1)(28:28)=CHAR(187)
      TABREC(12)(1:1)=CHAR(200)
      TABREC(12)(28:28)=CHAR(188)
      DO 520 J=2,27
        TABREC(1)(J:J)=CHAR(205)
        TABREC(12)(J:J)=CHAR(205)
  520   CONTINUE
      TABREC(1)(7:22)='Data Abstraction'
      TABREC(3)(3:13)='RECORD TYPE'
      TABREC(3)(19:24)='STATUS'
      TABREC(5)(4:9)='HEADER'
      TABREC(6)(4:10)='PROJECT'
      TABREC(7)(4:7)='TASK'
      TABREC(8)(4:7)='LINK'
      TABREC(9)(4:11)='RESOURCE'
      TABREC(10)(4:11)='CALENDAR'
      STATUS(1)='IN PROGRESS'
      STATUS(2)='  COMPLETED'
C
C...Draw display table (BRIGHT WHITE on BLUE)
C
      CALL SCLEAR
      CALL TXTCLR(15,1)
      NLTL=5
      NCTL=27
      DO 530 I=1,12
        CALL TMOVETO((NLTL+I-1),NCTL)
        CALL OUTEXT(TABREC(I))
  530   CONTINUE
C
C...Set text colour for 'STATUS' entries (WHITE on BLUE)
C
      CALL TXTCLR(7,1)
C
C...Initialise control statements
C
      ASSIGN 80 TO LABIF
      ASSIGN 90 TO LABTW
      ASSIGN 130 TO LABLW
      ASSIGN 170 TO LABRW
      ASSIGN 210 TO LABCW
C
C...Zero array counters
C
      IT=0
      IL=0
      IR=0
      IU=0
      IC=0
C
C...Open export file FLNAME on unit 3
C
      OPEN(UNIT=3,FILE=FLNAME)
C
C...Read HEADER record
C
      READ(3,20)REC
  20  FORMAT(A185)
      CALL TMOVETO((NLTL+4),(NCTL+15))
      CALL OUTEXT(STATUS(1))
```

```
C
C...Determine comma positions in record
C
      CALL RECCOM(10)
C
C...Zero character variable
C
      DO 50 I=1,3
        DO 40 J=1,30
          PHEAD(I)(J:J)=' '
 40       CONTINUE
 50     CONTINUE
C
C...Abstract character variables
C
C...Left header
C
      PHEAD(1)=REC((ICOMMA(2)+2):(ICOMMA(3)-2))
C
C...Right header
C
      PHEAD(3)=REC((ICOMMA(3)+2):(ICOMMA(4)-2))
C
C...Project version number
C
      PVER=REC((ICOMMA(10)+1):(ICOMMA(10)+1))
      CALL TMOVETO((NLTL+4),(NCTL+15))
      CALL OUTEXT(STATUS(2))
      CALL SPCHAR(1)
C
C...Read PROJECT record
C
      READ(3,20)REC
      CALL TMOVETO((NLTL+5),(NCTL+15))
      CALL OUTEXT(STATUS(1))
C
C...Determine comma positions in record
C
      CALL RECCOM(5)
C
C...Zero character variable
C
      PNAME='        '
C
C...Abstract character variable
C
C...Project Name
C
      PNAME=REC(4:(ICOMMA(2)-2))
C
C...Project Description
C
      PHEAD(2)=REC((ICOMMA(3)+2):(ICOMMA(4)-2))
```

```
C
C...Project Start Date
C
      PSTART=REC((ICOMMA(4)+2):(ICOMMA(5)-2))
      CALL TMOVETO((NLTL+5),(NCTL+15))
      CALL OUTEXT(STATUS(2))
      CALL SPCHAR(1)
C
C...Read record and abstract 1st character
C
  70  READ(3,20)REC
      C1=REC(1:1)
      GO TO LABIF
C
C...Check if TASK record
C
  80  IF(C1.EQ.'T'.OR.C1.EQ.'M'.OR.C1.EQ.'K'.OR.C1.EQ.'S')THEN
C
C...TASK record; Write screen message on FIRST task record only
C
        GO TO LABTW
  90    CALL TMOVETO((NLTL+6),(NCTL+15))
        CALL OUTEXT(STATUS(1))
        ASSIGN 110 TO LABTW
C
C...Update counter; Abstract data
C
  110   IT=IT+1
        CALL ABTASK
C
C...Zero TASK RESOURCE DEMAND counter and read next record
C
        ID=0
        GO TO 70
C
C...Check if TASK RESOURCE DEMAND record
C
      ELSEIF(C1.EQ.'D')THEN
C
C...TRD record; Update counter; Abstract data and read next record
C
        ID=ID+1
        NTRES(IT)=ID
        CALL ABTRD
        GO TO 70
      ENDIF
C
C...Check if LINK record
C
  120 IF(C1.EQ.'L')THEN
C
C...LINK record; All TASK type records abstracted;
C...Change label LABIF to avoid unneccessary testing
C
        ASSIGN 120 TO LABIF
```

```
C
C...Write screen message on FIRST link record only
C
        GO TO LABLW
  130   CALL TMOVETO((NLTL+6),(NCTL+15))
        CALL OUTEXT(STATUS(2))
        CALL SPCHAR(1)
        CALL TMOVETO((NLTL+7),(NCTL+15))
        CALL OUTEXT(STATUS(1))
        ASSIGN 150 TO LABLW

C
C...Update counter; Abstract data and read next record
C
  150   IL=IL+1
        CALL ABLINK
        GO TO 70
      ENDIF
C
C...Check if RESOURCE record
C
  160 IF(C1.EQ.'R')THEN
C
C...RESOURCE record; All TASK/LINK records abstracted;
C...Change label LABIF to avoid unneccessary testing;
C
        ASSIGN 160 TO LABIF
C
C...Write screen message on FIRST resource record only
C
        GO TO LABRW
  170   CALL TMOVETO((NLTL+7),(NCTL+15))
        CALL OUTEXT(STATUS(2))
        CALL SPCHAR(1)
        CALL TMOVETO((NLTL+8),(NCTL+15))
        CALL OUTEXT(STATUS(1))
        ASSIGN 190 TO LABRW
C
C...Update counter; Abstract data and read next record
C
  190   IR=IR+1
        CALL ABRES
        GO TO 70
C
C...Check if (unrequired) RESOURCE DEMAND record
C
      ELSEIF(C1.EQ.'Q'.OR.C1.EQ.'S')THEN
C
C...Unrequired record; Read next record
C
        GO TO 70
C
C...Check if SUB-RESOURCE record
C
      ELSEIF(C1.EQ.'U')THEN
```

```
C
C...S-R record; Update counter; Abstract data and read next record
C
        IU=IU+1
        CALL ABSUBRES
        GO TO 70
      ENDIF
C
C...Check if CALENDAR record
C
 200  IF(C1.EQ.'C')THEN
C
C...CALENDAR record; All TASK/LINK/RESOURCE records abstracted;
C...Change label LABIF to avoid unneccessary testing
C
        ASSIGN 200 TO LABIF
C
C...Write screen message on FIRST calendar record only
C
        GO TO LABCW
 210    CALL TMOVETO((NLTL+8),(NCTL+15))
        CALL OUTEXT(STATUS(2))
        CALL TMOVETO((NLTL+9),(NCTL+15))
        CALL OUTEXT(STATUS(1))
        ASSIGN 230 TO LABCW
C
C...Update counter; Abstract data and read next record
C
 230    IC=IC+1
        CALL ABCAL
        GO TO 70
      ELSE
C
C...All data abstracted; Close file
C
        CLOSE(3)
      ENDIF
C
C...Display concluding message
C
      CALL TMOVETO((NLTL+9),(NCTL+15))
      CALL OUTEXT(STATUS(2))
      CALL TXTCLR(15,1)
      CALL TMOVETO((NLTL+12),NCTL)
      CALL OUTEXT('     ALL DATA ABSTRACTED    ')
      CALL SPCHAR(1)
      CALL TXTCLR(0,4)
      CALL TMOVETO((NLTL+13),NCTL)
      CALL OUTEXT('   Hit any key to continue  ')
      CALL TXTCLR(0,0)
      CALL TMOVETO(25,70)
      CALL OUTEXT(' ')
      CALL TMOVETO(25,70)
      CALL KBREAD(0,C1)
```

```
C
C...Exit
C
      CALL TXTCLR(7,0)
      CALL SCLEAR
      RETURN
      END
```

Appendix C.5: Subroutine RECCOM

```
      SUBROUTINE RECCOM(NCOMMA)
c**************************************************************
c**                                                          **
c**                   Subroutine RECCOM                      **
c** Determines comma positions in PMA export file record REC **
c**                                                          **
c**             NCOMMA - Number of commas in record          **
c**                                                          **
c**************************************************************
c
      COMMON/RECORD/REC,ICOMMA(16)
      CHARACTER*185 REC
c
      J=1
      DO 10 I=1,185
        IF(REC(I:I).NE.',')GO TO 10
        ICOMMA(J)=I
        J=J+1
        IF(J.GT.NCOMMA)GO TO 20
  10    CONTINUE
c
c...Exit
c
  20  RETURN
      END
```

Appendix C.6: Subroutine ABTASK

```
      SUBROUTINE ABTASK
c*****************************************************
c**                                                 **
c**               Subroutine ABTASK                 **
c**                                                 **
c** Abstracts TASK data from export file record REC **
c**                                                 **
c** Abstracts START/FINISH code from OUTPUT record  **
c**                                                 **
c**          START  task: SFCODE - 0                **
c**          NORMAL task: SFCODE - 1                **
c**          FINISH task: SFCODE - 2                **
c*****************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400),TRES(400,10),
     &TRESQ(400,10),TCAL(400),NTRES(400),SFCODE(400),TWK(400),
     &CTRES(400,10),TCOST(400,9),TWBS(400)
      COMMON/RECORD/REC,ICOMMA(16)
      COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
C
      CHARACTER*1 TTYPE,SCODE,FCODE
      CHARACTER*3 CTRES
      CHARACTER*8 TCAL
      CHARACTER*9 TNAME
      CHARACTER*10 FMT
      CHARACTER*12 TWBS
      CHARACTER*185 REC
C
      INTEGER SFCODE,TWK,TRES
C
C...Determine comma positions in record
C
      CALL RECCOM(14)
C
C...Zero character variables
C
      TNAME(IT)='         '
      TCAL(IT)='        '
      SFCODE(IT)='N'
C
C...Abstract character variables
C
C...Task Type
C
      IF(REC(1:1).EQ.'K')THEN
        TTYPE(IT)='H'
      ELSE
        TTYPE(IT)=REC(1:1)
      ENDIF
```

```
C
C...Task Name
C
      TNAME(IT)=REC((ICOMMA(2)+2):(ICOMMA(3)-2))
C
C...Task Work Breakdown Structure
C
      TWBS(IT)=REC((ICOMMA(4)+2):(ICOMMA(5)-2))
C
C...Task Calendar
C
      TCAL(IT)=REC((ICOMMA(14)+1):(ICOMMA(14)+8))
C
C...Abstract numerical variables
C
C...Task Duration
C
      NX=ICOMMA(6)+1
      NC=ICOMMA(7)-(1)-(NX+1)
      WRITE(FMT,10)NX,NC
   10 FORMAT('(',I2,'X,I',I3,')')
      READ(REC,FMT)IDURN
      TDURN(IT)=REAL(IDURN)
C
C...Read OUTPUT record
C
      READ(3,20)REC
   20 FORMAT(A185)
C
C...Abstract START/FINISH code
C
      CALL RECCOM(16)
      SCODE=REC((ICOMMA(15)+2):(ICOMMA(15)+2))
      FCODE=REC((ICOMMA(16)+2):(ICOMMA(16)+2))
      SFCODE(IT)=1
      IF(SCODE.EQ.'1')THEN
        SFCODE(IT)=0
      ELSEIF(FCODE.EQ.'1')THEN
        SFCODE(IT)=2
      ENDIF
C
C...Exit
C
      RETURN
      END
```

Appendix C.7: Subroutine ABTRD

```
      SUBROUTINE ABTRD
c*****************************************
c**                                     **
c**        Subroutine ABTRD             **
c**                                     **
c** Abstracts TASK RESOURCE DEMAND data **
c**    from export file record REC      **
c**                                     **
c*****************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400),TRES(400,10),
     &TRESQ(400,10),TCAL(400),NTRES(400),SFCODE(400),TWK(400),
     &CTRES(400,10),TCOST(400,9),TWBS(400)
      COMMON/RECORD/REC,ICOMMA(16)
      COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
C
      CHARACTER*1 TTYPE
      CHARACTER*3 CTRES
      CHARACTER*8 TCAL
      CHARACTER*9 TNAME
      CHARACTER*11 NUMB
      CHARACTER*12 TWBS
      CHARACTER*185 REC
C
      INTEGER SFCODE,TWK,TRES
C
C...Determine comma positions in record
C
      CALL RECCOM(4)
C
C...Zero character variables
C
      CTRES(IT,ID)='   '
C
C...Abstract character variables
C
C...Resource Name
C
      CTRES(IT,ID)=REC((ICOMMA(3)+2):(ICOMMA(4)-2))
C
C...Abstract numerical variables
C
C...Resource Quantity
C...Check whether INTEGER or FLOATING POINT number
C
      NUMB=REC((ICOMMA(4)+1):(ICOMMA(4)+11))
      DO 30 I=1,11
        IF(NUMB(I:I).EQ.'.')THEN
```

```
C
C...FLOATING POINT number; Abstract using 'F' format
C
          READ(NUMB,10)TRESQ(IT,ID)
 10       FORMAT(F11.4)
          GO TO 40
        ELSEIF(NUMB(I:I).EQ.' ')THEN

C
C...INTEGER number; Abstract using 'I'
C...format and convert to FP number
C
          READ(NUMB,20)IRESQ
 20       FORMAT(I6)
          TRESQ(IT,ID)=REAL(IRESQ)
          GO TO 40
        ENDIF
 30     CONTINUE
C
C...Exit
C
 40   RETURN
      END
```

Appendix C.8: Subroutine ABLINK

```
      SUBROUTINE ABLINK
c****************************************************************
c**                                                            **
c**                       Subroutine ABLINK                    **
c**                                                            **
c**        Abstracts LINK data from export file record REC     **
c**                                                            **
c**        Outputs SOURCE task (LFROM) and TARGET task (LTO)   **
c**                        as task INDEX                       **
c**                                                            **
c**           Outputs Link Type (LTYPE) in INTEGER form        **
c**                                                            **
c**                    F-S: LTYPE - 1                          **
c**                    S-S: LTYPE - 2                          **
c**                    F-F: LTYPE - 3                          **
c**                    S-F: LTYPE - 4                          **
c**                                                            **
c****************************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400),TRES(400,10),
     &TRESQ(400,10),TCAL(400),NTRES(400),SFCODE(400),TWK(400),
     &CTRES(400,10),TCOST(400,9),TWBS(400)
C
      COMMON/LINKS/LFROM(500,2),LTO(500,2),LTYPE(500,2),
     &LDURN(500,2)
      COMMON/RECORD/REC,ICOMMA(16)
      COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
C
      CHARACTER*1 TTYPE
      CHARACTER*3 CLTYPE,CTRES
      CHARACTER*8 FMT,TCAL
      CHARACTER*9 TNAME,CLFROM,CLTO
      CHARACTER*12 TWBS
      CHARACTER*185 REC
C
      REAL LDURN
      INTEGER SFCODE,TWK,TRES
C
C...Determine comma positions in record
C
      CALL RECCOM(5)
C
C...Zero character variables
C
      CLFROM='         '
      CLTO='         '
C
C...Abstract character variables
C
C...Source task
C
      CLFROM=REC((ICOMMA(2)+2):(ICOMMA(3)-2))
```

```
C
C...Target task
C
      CLTO=REC((ICOMMA(3)+2):(ICOMMA(4)-2))
C
C...Link type
C
      CLTYPE=REC((ICOMMA(4)+2):(ICOMMA(5)-2))
C
C...Abstract link duration as a numerical variable
C
      NX=ICOMMA(5)+1
      J=NX+3
      WRITE(FMT,20)NX
   20 FORMAT('(',I2,'X,I3)')
   21 READ(REC,FMT,ERR=22)IDURN
      GO TO 23
   22 IF(REC(J:J).EQ.'"')THEN
        FMT(7:7)='2'
      ELSE
        FMT(7:7)='1'
      ENDIF
      GO TO 21
   23 LDURN(IL,1)=REAL(IDURN)
C
C...Change SOURCE/TARGET task names to task indices
C
      DO 30 I=1,IT
        IF(CLFROM.EQ.TNAME(I))THEN
          LFROM(IL,1)=I
          GO TO 40
        ENDIF
   30   CONTINUE
   40 DO 50 I=1,IT
        IF(CLTO.EQ.TNAME(I))THEN
          LTO(IL,1)=I
          GO TO 60
        ENDIF
   50   CONTINUE
C
C...Change Link Type to an integer
C
   60 IF(CLTYPE.EQ.'F-S')THEN
        LTYPE(IL,1)=1
        GO TO 70
      ELSEIF(CLTYPE.EQ.'S-S')THEN
        LTYPE(IL,1)=2
        GO TO 70
      ELSEIF(CLTYPE.EQ.'F-F')THEN
        LTYPE(IL,1)=3
        GO TO 70
      ELSE
        LTYPE(IL,1)=4
      ENDIF
```

```
C
C...Exit
C
  70  RETURN
      END
```

Appendix C.9: Subroutine ABRES

```
      SUBROUTINE ABRES
c*********************************************************
c**                                                     **
c**                 Subroutine ABRES                    **
c**                                                     **
c** Abstracts RESOURCE data from export file record REC **
c**                                                     **
c*********************************************************
c
      COMMON/RES/RNAME(100),RTYPE(100),RCOST(100)
      COMMON/RECORD/REC,ICOMMA(16)
      COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
C
      CHARACTER*1 RTYPE
      CHARACTER*3 RNAME
      CHARACTER*185 REC
C
C
C...Determine comma positions in record
C
      CALL RECCOM(5)
C
C...Zero character variables
C
      RNAME(IR)='   '
C
C...Abstract character variables
C
C...Resource name
C
      RNAME(IR)=REC((ICOMMA(1)+2):(ICOMMA(2)-2))
C
C...Resource type
C
      RTYPE(IR)=REC((ICOMMA(3)+1):(ICOMMA(3)+1))
C
C...Exit
C
      RETURN
      END
```

Appendix C.10: Subroutine ABSUBRES

```
      SUBROUTINE ABSUBRES
c*************************************************************
c**                                                         **
c**                  Subroutine ABSUBRES                    **
c**                                                         **
c** Abstracts SUB-RESOURCE data from export file record REC **
c**                                                         **
c*************************************************************
c
      COMMON/SUBRES/CRTOP(110),CRSUB(110),RSUBQ(110),RTOP(110),
     &RSUB(110)
      COMMON/RECORD/REC,ICOMMA(16)
      COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
C
      CHARACTER*3 CRTOP,CRSUB
      CHARACTER*11 NUMB
      CHARACTER*185 REC
C
      INTEGER RTOP,RSUB
C
C...Determine comma positions in record
C
      CALL RECCOM(3)
C
C...Zero character variables
C
      CRTOP(IU)='   '
      CRSUB(IU)='   '
C
C...Abstract character variables
C
C...Resource
C
      CRTOP(IU)=REC((ICOMMA(1)+2):(ICOMMA(2)-2))
C
C...Sub-resource
C
      CRSUB(IU)=REC((ICOMMA(2)+2):(ICOMMA(3)-2))
C
C...Abstract numerical variable
C
C...Sub-resource quantity
C...Check whether INTEGER or FLOATING POINT number
C
      NUMB=REC((ICOMMA(3)+1):(ICOMMA(3)+11))
      DO 30 I=1,11
        IF(NUMB(I:I).EQ.'.')THEN
C
C...FLOATING POINT number; Abstract using 'F' format
C
          READ(NUMB,10)RSUBQ(IU)
  10      FORMAT(F11.4)
```

```
           GO TO 40
         ELSEIF(NUMB(I:I).EQ.' ')THEN
C
C...INTEGER number; Abstract using 'I'
C...format and convert to FP number
C
           READ(NUMB,20)ISUBQ
  20       FORMAT(I6)
           RSUBQ(IU)=REAL(ISUBQ)
           GO TO 40
         ENDIF
  30     CONTINUE
C
C...Exit
C
  40   RETURN
       END
```

Appendix C.11: Subroutine ABCAL

```
       SUBROUTINE ABCAL
c*********************************************************
c**                                                     **
c**                  Subroutine ABCAL                   **
c**                                                     **
c** Abstracts CALENDAR data from export file record REC **
c**                                                     **
c*********************************************************
c
       COMMON/CALS/CAL(5),ICWK(5)
       COMMON/RECORD/REC,ICOMMA(16)
       COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
C
       CHARACTER*8 CAL
       CHARACTER*185 REC
C
C...Zero character variable
C
       CAL(IC)='        '
C
C...Abstract character variable
C
       DO10 I=4,20
         IF(REC(I:I).EQ.'"')GO TO 20
  10     CONTINUE
  20   CAL(IC)=REC(4:(I-1))
C
C...Exit
C
       RETURN
       END
```

Appendix C.12: Subroutine CALWKS

```
      SUBROUTINE CALWKS
c*****************************************
c**         Subroutine CALWKS           **
c**                                     **
c** Keyboard input of working week length **
c**         for each calendar;          **
c**                                     **
c** Determination of working week length  **
c**          for each task;             **
c**                                     **
c*****************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400),TRES(400,10),
     &TRESQ(400,10),TCAL(400),NTRES(400),SFCODE(400),TWK(400),
     &CTRES(400,10),TCOST(400,9),TWBS(400)
      COMMON/CALS/CAL(5),ICWK(5)
      COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
C
      CHARACTER*1 TTYPE,CHR
      CHARACTER*3 CTRES
      CHARACTER*8 CAL,TCAL
      CHARACTER*9 TNAME
      CHARACTER*12 TWBS
      CHARACTER*27 TABREC(12),BLANK
C
      INTEGER SFCODE,TWK,TRES
C
C...Initialise display table records
C
      DO 20 I=1,12
        TABREC(I)(1:1)=CHAR(186)
        TABREC(I)(27:27)=CHAR(186)
        DO 10 J=2,26
          TABREC(I)(J:J)=' '
   10     CONTINUE
   20   CONTINUE
      DO 30 J=2,26
        TABREC(1)(J:J)=CHAR(205)
        TABREC(12)(J:J)=CHAR(205)
        BLANK(J:J)=' '
   30   CONTINUE
      TABREC(1)(1:1)=CHAR(201)
      TABREC(1)(6:22)='Working Week data'
      TABREC(1)(27:27)=CHAR(187)
      TABREC(3)(3:10)='CALENDAR'
      TABREC(3)(14:25)='WORKING WEEK'
      TABREC(4)(17:22)='(Days)'
      TABREC(12)(1:1)=CHAR(200)
      TABREC(12)(27:27)=CHAR(188)
      DO 40 I=1,IC
        J=I+5
        TABREC(J)(3:10)=CAL(I)
```

```
  40    CONTINUE
C
C...Draw display table (BRIGHT WHITE on BLUE)
C
      NLTL=8
      NCTL=27
      NLTOP=NLTL-1
      CALL TXTCLR(7,0)
      CALL SCLEAR
      CALL TXTCLR(15,1)
      DO 50 I=1,12
        CALL TMOVETO((NLTL+I-1),NCTL)
        CALL OUTEXT(TABREC(I))
  50    CONTINUE
C
C...Define option highlight delay
C
      NDEL=30
C
C...Input calendar working week data from keyboard
C
  60  DO 80 I=1,IC
C
C...Highlight entry slot in table (BROWN)
C
        CALL TXTCLR(0,6)
        CALL TMOVETO((NLTL+I+4),(NCTL+18))
        CALL OUTEXT(' ')
C
C...Display input line (BRIGHT WHITE)
C
        CALL TXTCLR(15,0)
        CALL TMOVETO(NLTOP,NCTL)
        CALL OUTEXT(CAL(I)//': Week length [ ]')
        CALL TMOVETO(NLTOP,(NCTL+23))
C
C...Read working week length from keyboard
C
  70    CALL KBREAD(0,CHR)
        ICHR=ICHAR(CHR)
        IF(ICHR.LT.49.OR.ICHR.GT.57)THEN
C
C...Inappropriate character; try again
C
          CALL SPCHAR(1)
          GO TO 70
        ENDIF
C
C...Appropriate character; update table and variable ICWK
C
        CALL TXTCLR(7,0)
        DO 75 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTL+23))
          CALL OUTEXT(CHR)
  75      CONTINUE
```

```
        ICWK(I)=ICHR-48
        CALL TXTCLR(7,1)
        CALL TMOVETO((NLTL+I+4),(NCTL+18))
        CALL OUTEXT(CHR)
  80    CONTINUE
C
C...Display CONTINUATION message;
C...highlight 1st character of options
C
      CALL TXTCLR(15,0)
      CALL TMOVETO(NLTOP,NCTL)
      CALL OUTEXT(BLANK(2:26))
      CALL TMOVETO(NLTOP,NCTL)
      CALL OUTEXT('All OK?  YES  NO')
      CALL TXTCLR(4,0)
      CALL TMOVETO(NLTOP,(NCTL+9))
      CALL OUTEXT('Y')
      CALL TMOVETO(NLTOP,(NCTL+14))
      CALL OUTEXT('N')
      CALL TXTCLR(0,0)
      CALL TMOVETO(25,70)
      CALL OUTEXT(' ')
      CALL TMOVETO(25,70)
C
C...Read options from keyboard
C
  90  CALL KBREAD(0,CHR)
      CALL CHRCNV(CHR)
      IF(CHR.EQ.'N')THEN
C
C...Option is UNSATISFACTORY; Erase table data and try again;
C
        CALL TXTCLR(0,7)
        DO 110 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTL+14))
          CALL OUTEXT('NO')
  110     CONTINUE
        CALL TXTCLR(0,1)
        DO 120 I=1,IC
          CALL TMOVETO((NLTL+I+4),(NCTL+18))
          CALL OUTEXT(' ')
  120     CONTINUE
        GO TO 60
      ELSEIF(CHR.EQ.'Y')THEN
C
C...Option is SATISFACTORY; Continue
C
        CALL TXTCLR(0,7)
        DO 130 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTL+9))
          CALL OUTEXT('YES')
  130     CONTINUE
        GO TO 140
      ELSE
```

```
C
C...Inappropriate character; try again
C
        CALL SPCHAR(1)
        GO TO 90
      ENDIF
C
C...Erase input display line; Display 'initialisation' message;
C
  140 CALL TXTCLR(0,0)
      CALL TMOVETO(NLTOP,NCTL)
      CALL OUTEXT(BLANK(2:26))
      CALL TXTCLR(15,0)
      CALL TMOVETO((NLTL+12),NCTL)
      CALL OUTEXT('Commencing data initialisation')
C
C...Define length of working week for each task
C
      DO 160 I-1,IT
        DO 150 J=1,IC
          IF(TCAL(I).EQ.CAL(J))THEN
            TWK(I)=ICWK(J)
            GO TO 160
          ENDIF
  150     CONTINUE
  160   CONTINUE
C
C...Exit
C
      RETURN
      END
```

Appendix C.13: Subroutine TCOSTS

```
      SUBROUTINE TCOSTS
c*******************************************************
c**                                                   **
c**                Subroutine TCOSTS                  **
c**                                                   **
c**     Keyboard input of cost component data;        **
c**                                                   **
c** Determination of cost components for each task;   **
c**                                                   **
c*******************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400),TRES(400,10),
     &TRESQ(400,10),TCAL(400),NTRES(400),SFCODE(400),TWK(400),
     &CTRES(400,10),TCOST(400,9),TWBS(400)
      COMMON/RES/RNAME(100),RTYPE(100),RCOST(100)
      COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
      COMMON/COSTS/CCODE(9),CPAY(9),CARR(9),CARRQ(9)
C
      CHARACTER*1 TTYPE,RTYPE,CPAY,CARR,CHR
      CHARACTER*3 RNAME,CTRES,CCODE
      CHARACTER*4 FMT,CHR4
      CHARACTER*8 TCAL
      CHARACTER*9 TNAME
      CHARACTER*10 CRDEL
      CHARACTER*12 TWBS
      CHARACTER*54 TABREC(16),BLANK
C
      INTEGER SFCODE,TWK,TRES,CARRQ
C
C...Initialise display table strings
C
      DO 20 I=1,16
        TABREC(I)(1:1)=CHAR(186)
        TABREC(I)(54:54)=CHAR(186)
        DO 10 J=2,53
          TABREC(I)(J:J)=' '
  10      CONTINUE
  20    CONTINUE
      DO 30 J=2,53
        TABREC(1)(J:J)=CHAR(205)
        TABREC(16)(J:J)=CHAR(205)
        BLANK(J:J)=' '
  30    CONTINUE
      TABREC(1)(1:1)=CHAR(201)
      TABREC(1)(18:36)='Cost Component Data'
      TABREC(1)(54:54)=CHAR(187)
      TABREC(3)(3:37)='COMPONENT    LOADING  CODE  PAYMENT'
      TABREC(3)(40:52)='PAYMENT DELAY'
      TABREC(5)(3:21)='LABOUR       Normal'
      TABREC(6)(3:21)='MATERIALS    Front '
      TABREC(7)(3:21)='PLANT        Normal'
      TABREC(8)(3:21)='PLANT        Front '
      TABREC(9)(3:21)='PLANT       Back   '
      TABREC(10)(3:21)='SUBCONTRACT  Spread'
      TABREC(11)(3:21)='OVERHEADS    Normal'
      TABREC(12)(3:21)='OVERHEADS    Front '
      TABREC(14)(3:21)='TASK PRICE   Spread'
      TABREC(16)(1:1)=CHAR(200)
      TABREC(16)(54:54)=CHAR(188)
C
C...Initialise task cost component arrays
C
      DO 50 I=1,IT
        DO 40 J=1,9
          TCOST(I,J)=0.0
  40      CONTINUE
  50    CONTINUE
C
C...Specify option highlight delay
C
      NDEL=30
C
C...Specify table position parameters
C
      NLTL=4
      NCTL=14
      NLTOP=NLTL-1
      NCTOP=NCTL
      NLBOT=NLTL+16
      NCBOT=NCTL
C
C...Draw display table (BRIGHT WHITE on BLUE)
C
      CALL TXTCLR(7,0)
      CALL SCLEAR
      CALL TXTCLR(15,1)
      DO 60 I=1,16
        CALL TMOVETO((NLTL+I-1),NCTL)
        CALL OUTEXT(TABREC(I))
  60    CONTINUE
C
C...Input cost component data from keyboard
C
      DO 240 I=1,9
  65    NLTAB=NLTL+I+3
        IF(I.GT.8)NLTAB=NLTAB+1
C
C...Display CODE input line;
C
        CALL TXTCLR(15,0)
        CALL TMOVETO(NLTOP,NCTOP)
        CALL OUTEXT('Code: [    ] ')
        CALL OUTEXT('(Hit <CR> if component not applicable)')
```

```
C
C...Highlight table entry position;
C...Input component code from keyboard;
C
      CALL TXTCLR(0,6)
      CALL TMOVETO(NLTAB,(NCTL+24))
      CALL OUTEXT('   ')
 70   NCHR=3
      CCODE(I)='   '
      CALL CREAD(NLTOP,(NCTOP+7),CCODE(I),NCHR)
C
C...Erase any existing error messages
C
      CALL TMOVETO(NLBOT,NCBOT)
      CALL OUTEXT(BLANK(2:53))
      IF(NCHR.EQ.0)THEN
C
C...Component not applicable; erase input line;
C...Update table and variables accordingly and
C...proceed to next component;
C
        CALL TXTCLR(0,0)
        CALL TMOVETO(NLTOP,NCTOP)
        CALL OUTEXT(BLANK(2:53))
        CALL TXTCLR(7,1)
        CALL TMOVETO(NLTAB,(NCTL+24))
        CALL OUTEXT(' - ')
        CALL TMOVETO(NLTAB,(NCTL+32))
        CALL OUTEXT('-')
        CALL TMOVETO(NLTAB,(NCTL+43))
        CALL OUTEXT('-')
        CCODE(I)='000'
        CPAY(I)='0'
        CARR(I)='0'
        CARRQ(I)=0
        GO TO 205
      ENDIF
C
C...Component applicable; convert code to upper case if required;
C...Search for cost component index
C
      CALL CHRCNV(CCODE(I))
      DO 80 J=1,IR
        IF(RNAME(J).EQ.CCODE(I))THEN
          ICOST=J
          GO TO 90
        ENDIF
 80     CONTINUE
C
C...Cost component index not found; display error message;
C...try again;
C
      CALL TXTCLR(15,0)
      CALL TMOVETO(NLBOT,NCBOT)
      CALL OUTEXT('Component code not found; Try again!!')
      CALL SPCHAR(1)
      CALL TMOVETO(NLTOP,(NCTOP+7))
      CALL OUTEXT('   ')
      GO TO 70
C
C...Cost component code found; Update table (WHITE on BLUE)
C
 90   CALL TXTCLR(7,1)
      CALL TMOVETO(NLTAB,(NCTL+24))
      CALL OUTEXT(CCODE(I))
C
C...Hide cursor while calculating
C
      CALL TXTCLR(0,0)
      CALL TMOVETO(25,70)
      CALL OUTEXT(' ')
      CALL TMOVETO(25,70)
C
C...Calculate appropriate cost component for each task;
C...Avoid calculation if a task has no resource demands;
C
      CALL RESCOST(ICOST)
      DO 110 J=1,IT
        IF(NTRES(J).GT.0)THEN
          DO 100 K=1,NTRES(J)
            L=TRES(J,K)
            TCOST(J,I)=TCOST(J,I)+RCOST(L)*TRESQ(J,K)
 100        CONTINUE
        ENDIF
 110    CONTINUE
C
C...Avoid repetitive input for PLANT and OVERHEADS components
C
      IF(I.NE.4.AND.I.NE.5.AND.I.NE.8)GO TO 115
      IF(CCODE(I-1).EQ.'000')GO TO 115
      CALL TMOVETO(NLTOP,NCTOP)
      CALL OUTEXT(BLANK(2:53))
      CPAY(I)=CPAY(I-1)
      CARR(I)=CARR(I-1)
      CARRQ(I)=CARRQ(I-1)
      CALL TXTCLR(7,1)
      IF(CPAY(I).EQ.'W')THEN
        CALL TMOVETO(NLTAB,(NCTL+30))
        CALL OUTEXT('WEEKLY ')
      ELSE
        CALL TMOVETO(NLTAB,(NCTL+30))
        CALL OUTEXT('MONTHLY')
      ENDIF
      IF(CARR(I).EQ.'0')THEN
        CALL TMOVETO(NLTAB,(NCTL+40))
        CALL OUTEXT('    -      ')
        GO TO 205
      ENDIF
      GO TO 202
```

```
C
C...Blank out existing input line; Display PAYMENT input line
C...Highlight first character of options
C
  115   CALL TXTCLR(15,0)
        CALL TMOVETO(NLTOP,NCTOP)
        CALL OUTEXT(BLANK(2:53))
        CALL TMOVETO(NLTOP,NCTOP)
        CALL OUTEXT('Payment:  WEEKLY  MONTHLY')
        CALL TXTCLR(4,0)
        CALL TMOVETO(NLTOP,(NCTOP+10))
        CALL OUTEXT('W')
        CALL TMOVETO(NLTOP,(NCTOP+18))
        CALL OUTEXT('M')
C
C...Highlight table entry position; Hide cursor
C
        CALL TXTCLR(0,6)
        CALL TMOVETO(NLTAB,(NCTL+30))
        CALL OUTEXT('        ')
        CALL TXTCLR(0,0)
        CALL TMOVETO(25,70)
        CALL OUTEXT(' ')
        CALL TMOVETO(25,70)
C
C...Read option from keyboard; convert to upper case if required;
C
  120   CALL KBREAD(0,CPAY(I))
        CALL CHRCNV(CPAY(I))
        IF(CPAY(I).EQ.'W')THEN
C
C...Option is WEEKLY; Highlight option; Update table; Update CPAY;
C
          CALL TXTCLR(0,7)
          DO 130 M=1,NDEL
            CALL TMOVETO(NLTOP,(NCTOP+10))
            CALL OUTEXT('WEEKLY')
  130       CONTINUE
          CALL TXTCLR(7,1)
          CALL TMOVETO(NLTAB,(NCTL+30))
          CALL OUTEXT('WEEKLY ')
        ELSEIF(CPAY(I).EQ.'M')THEN
C
C...Option is MONTHLY; Highlight option; Update table;
C...Update CPAY;
C
          CALL TXTCLR(0,7)
          DO 140 M=1,NDEL
            CALL TMOVETO(NLTOP,(NCTOP+18))
            CALL OUTEXT('MONTHLY')
  140       CONTINUE
          CALL TXTCLR(7,1)
          CALL TMOVETO(NLTAB,(NCTL+30))
          CALL OUTEXT('MONTHLY')
        ELSE
```

```
C
C...Inappropriate character; try again
C
        CALL SPCHAR(1)
        GO TO 120
      ENDIF
C
C...Display PAYMENT DELAY input line + E.G.line
C
  150 CALL TXTCLR(15,0)
      CALL TMOVETO(NLTOP,NCTOP)
      CALL OUTEXT('Payment delay: [    ] ')
      CALL OUTEXT('(Hit <CR> if no payment delay)')
      CALL TMOVETO(NLBOT,NCBOT)
      CALL OUTEXT('[e.g. 28d '//CHAR(240)//' 28 days; ')
      CALL OUTEXT('6w '//CHAR(240)//' 6 weeks; ')
      CALL OUTEXT('1m '//CHAR(240)//' 1 month;]')
C
C...Highlight table entry position
C
      CALL TXTCLR(0,6)
      CALL TMOVETO(NLTAB,(NCTL+40))
      CALL OUTEXT('          ')
C
C...Read PAYMENT DELAY from keyboard; erase input + E.G.line
C
      NCHR=4
      CHR4='    '
      CALL CREAD(NLTOP,(NCTOP+16),CHR4,NCHR)
      CALL TMOVETO(NLTOP,NCTOP)
      CALL OUTEXT(BLANK(2:53))
      CALL TMOVETO(NLBOT,NCBOT)
      CALL OUTEXT(BLANK(2:53))
      IF(NCHR.EQ.0)THEN
C
C...Payment delay NOT APPLICABLE; Update table;
C...Update CARR and CARRQ accordingly;
C
        CALL TXTCLR(7,1)
        CALL TMOVETO(NLTAB,(NCTL+40))
        CALL OUTEXT('   -      ')
        CARR(I)='0'
        CARRQ(I)=0
        GO TO 205
      ENDIF
C
C...Payment delay APPLICABLE; convert to upper case if required
C...Check suitability of input
C
      CALL CHRCNV(CHR4)
      DO 160 J=1,NCHR
        CHR=CHR4(J:J)
        K=ICHAR(CHR)
        IF(K.GT.47.AND.K.LT.58)GO TO 160
        IF(CHR.EQ.'D'.OR.CHR.EQ.'W'.OR.CHR.EQ.'M')THEN
```

```
            L=J-1
            GO TO 180
          ENDIF
  160     CONTINUE
C
C...Unsuitable input; display error message and try again;
C
        CALL TMOVETO(NLBOT,NCBOT)
        CALL OUTEXT(BLANK(2:53))
        CALL SPCHAR(1)
        CALL TXTCLR(15,0)
        DO 170 M=1,NDEL
          CALL TMOVETO(NLBOT,NCBOT)
          CALL OUTEXT('Inappropriate input; try again!!')
  170     CONTINUE
        GO TO 150
C
C...Suitable input; abstract data
C
  180   WRITE(FMT,190)L
  190   FORMAT('(I',I1,')')
        READ(CHR4,FMT)CARRQ(I)
        CARR(I)=CHR4(J:J)
C
C...Define table display string
C
        CRDEL='          '
        WRITE(CRDEL(1:3),200)CARRQ(I)
  200   FORMAT(I3)
        IF(CARR(I).EQ.'D')THEN
          CRDEL(5:7)='day'
          IF(CARRQ(I).GT.1)CRDEL(8:8)='s'
        ELSEIF(CARR(I).EQ.'W')THEN
          CRDEL(5:8)='week'
          IF(CARRQ(I).GT.1)CRDEL(9:9)='s'
        ELSE
          CRDEL(5:9)='month'
          IF(CARRQ(I).GT.1)CRDEL(10:10)='s'
        ENDIF
C
C...Update table
C
  202   CALL TXTCLR(7,1)
        CALL TMOVETO(NLTAB,(NCTL+40))
        CALL OUTEXT(CRDEL)
C
C...Display CONTINUE? input line
C...Highlight first character of options
C
  205   CALL TXTCLR(15,0)
        CALL TMOVETO(NLTOP,NCTOP)
        CALL OUTEXT('Continue?  YES  NO')
        CALL TXTCLR(4,0)
        CALL TMOVETO(NLTOP,(NCTOP+11))
        CALL OUTEXT('Y')
        CALL TMOVETO(NLTOP,(NCTOP+16))
        CALL OUTEXT('N')
C
C...Hide cursor
C
        CALL TXTCLR(0,0)
        CALL TMOVETO(25,70)
        CALL OUTEXT(' ')
        CALL TMOVETO(25,70)
C
C...Read option from keyboard; convert to upper case if required;
C
  210   CALL KBREAD(0,CHR)
        CALL CHRCNV(CHR)
        IF(CHR.EQ.'Y')THEN
C
C...Option is CONTINUE; Highlight option;
C...Proceed to next component;
C
          CALL TXTCLR(0,7)
          DO 220 M=1,NDEL
            CALL TMOVETO(NLTOP,(NCTOP+11))
            CALL OUTEXT('YES')
  220       CONTINUE
          GO TO 240
        ELSEIF(CHR.EQ.'N')THEN
C
C...Option is DISCONTINUE; Highlight option;
C
          CALL TXTCLR(0,7)
          DO 230 M=1,NDEL
            CALL TMOVETO(NLTOP,(NCTOP+16))
            CALL OUTEXT('NO')
  230       CONTINUE
C
C...Erase table entries and input new entries
C
          CALL TXTCLR(7,1)
          CALL TMOVETO(NLTAB,(NCTL+24))
          CALL OUTEXT(BLANK(25:50))
          GO TO 65
        ELSE
C
C...Inappropriate character; try again
C
          CALL SPCHAR(1)
          GO TO 210
        ENDIF
  240 CONTINUE
C
C...Erase any input or error lines; Display COMPLETION message
C
      CALL TXTCLR(15,0)
      CALL TMOVETO(NLTOP,NCTOP)
      CALL OUTEXT(BLANK(2:53))
      CALL TMOVETO(NLBOT,NCBOT)
      CALL OUTEXT(BLANK(2:53))
      CALL TMOVETO(NLBOT,NCBOT)
      CALL OUTEXT('Completing data initialisation')
C
C...Exit
C
      RETURN
      END
```

Appendix C.14: Subroutine QUIT

```
      SUBROUTINE QUIT(FLNAME,NCHR)
c*******************************************
c**                                       **
c**            Subroutine QUIT            **
c**                                       **
c**   Concluding display for program PERTRA1 **
c**                                       **
c** Writes data to data files for subsequent **
c**       use by PERTRA2 and PERTRA3       **
c**                                       **
c*******************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400),TRES(400,10),
     &TRESQ(400,10),TCAL(400),NTRES(400),SFCODE(400),TWK(400),
     &CTRES(400,10),TCOST(400,9),TWBS(400)
      COMMON/LINKS/LFROM(500,2),LTO(500,2),LTYPE(500,2),
     &LDURN(500,2)
      COMMON/PROJ/PHEAD(3),PVER,PNAME,PSTART
      COMMON/COSTS/CCODE(9),CPAY(9),CARR(9),CARRQ(9)
      COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
C
      CHARACTER*1 TTYPE,PVER,CPAY,CARR
      CHARACTER*3 CTRES,CCODE
      CHARACTER*8 TCAL,PNAME
      CHARACTER*9 TNAME,PSTART
      CHARACTER*12 FLNAME,TWBS
      CHARACTER*30 PHEAD
      CHARACTER*37 TABREC(20)
C
      REAL LDURN
      INTEGER SFCODE,TWK,TRES,CARRQ
C
C...Initialise display table strings
C
      DO 20 I=1,20
        TABREC(I)(1:1)=CHAR(186)
        TABREC(I)(37:37)=CHAR(186)
        DO 10 J=2,36
          TABREC(I)(J:J)=' '
 10       CONTINUE
 20     CONTINUE
      TABREC(1)(1:1)=CHAR(201)
      TABREC(1)(37:37)=CHAR(187)
      TABREC(5)(1:1)=CHAR(204)
      TABREC(5)(37:37)=CHAR(185)
      TABREC(20)(1:1)=CHAR(200)
      TABREC(20)(37:37)=CHAR(188)
      DO 30 J=2,36
        TABREC(1)(J:J)=CHAR(205)
        TABREC(5)(J:J)=CHAR(205)
        TABREC(20)(J:J)=CHAR(205)
 30     CONTINUE
```

```
      TABREC(15)=TABREC(5)
      TABREC(3)(10:28)='Quitting PERTRA1 !!'
      TABREC(7)(5:33)='The following data files have'
      TABREC(8)(5:17)='been created:'
C
C...Define table position
C
      NLTL=3
      NCTL=22
C
C...Clear screen; draw table
C
      CALL TXTCLR(0,0)
      CALL SCLEAR
      CALL TXTCLR(15,1)
      DO 40 I=1,20
        CALL TMOVETO((NLTL+I-1),NCTL)
        IF(I.LT.2.OR.I.GT.4)THEN
          CALL OUTEXT(TABREC(I))
        ELSE
          CALL OUTEXT(TABREC(I)(1:1))
          CALL TXTCLR(0,4)
          CALL OUTEXT(TABREC(I)(2:36))
          CALL TXTCLR(15,1)
          CALL OUTEXT(TABREC(I)(37:37))
        ENDIF
  40    CONTINUE
C
C...Write data to datafiles
C
C...Define GENERAL file
C
      FLNAME((NCHR+1):(NCHR+4))='.GEN'
C
C...Open file on Unit 4; Write GENERAL data
C
      OPEN(UNIT=4,FILE=FLNAME)
      WRITE(4,50)PNAME,PVER
  50  FORMAT(A8,1X,A2)
      WRITE(4,60)(PHEAD(I),I=1,3)
  60  FORMAT(A30)
      WRITE(4,70)PSTART
  70  FORMAT(A9)
      CLOSE(4)
C
C...Update table
C
      CALL TXTCLR(15,1)
      CALL TMOVETO((NLTL+9),(NCTL+5))
      CALL OUTEXT('GENERAL data: ')
      CALL TXTCLR(14,1)
      CALL OUTEXT(FLNAME(1:(NCHR+4)))
```

```
C
C...Define TASKS data file
C
      FLNAME((NCHR+1):(NCHR+4))='.TSK'
C
C...Open file on Unit 4; Write TASKS data;
C
      OPEN(UNIT=4,FILE=FLNAME)
      WRITE(4,80)IT
   80 FORMAT(I4)
      DUM=0.0
      IDUM=1
      WRITE(4,90)(TTYPE(I),TNAME(I),TWBS(I),TWK(I),SFCODE(I),IDUM,
     &DUM,TDURN(I),DUM,I=1,IT)
   90 FORMAT(1X,A1,1X,A9,1X,A12,3I2,3F7.2)
      CLOSE(4)
C
C...Update table
C
      CALL TXTCLR(15,1)
      CALL TMOVETO((NLTL+10),(NCTL+5))
      CALL OUTEXT('TASK    data: ')
      CALL TXTCLR(14,1)
      CALL OUTEXT(FLNAME(1:(NCHR+4)))
C
C...Define LINKS data file
C
      FLNAME((NCHR+1):(NCHR+4))='.LNK'
C
C...Open file on Unit 4; Write LINKS data
C
      OPEN(UNIT=4,FILE=FLNAME)
      WRITE(4,80)IL
      WRITE(4,100)((LFROM(I,J),LTO(I,J),LTYPE(I,J),
     &LDURN(I,J),J=1,2),I=1,IL)
  100 FORMAT(2(3I4,F7.2))
      CLOSE(4)
C
C...Update table
C
      CALL TXTCLR(15,1)
      CALL TMOVETO((NLTL+11),(NCTL+5))
      CALL OUTEXT('LINK    data: ')
      CALL TXTCLR(14,1)
      CALL OUTEXT(FLNAME(1:(NCHR+4)))
C
C...Define COSTS data file
C
      FLNAME((NCHR+1):(NCHR+4))='.CST'
C
C...Open file on Unit 4; Write COSTS data
C
      OPEN(UNIT=4,FILE=FLNAME)
      WRITE(4,110)(CCODE(I),CPAY(I),CARR(I),CARRQ(I),I=1,9)
  110 FORMAT(A3,1X,A1,1X,A1,I4)
```

```
      WRITE(4,120)((TCOST(I,J),J=1,9),I=1,IT)
  120 FORMAT(9F10.2)
      CLOSE(4)
C
C...Update table
C
      CALL TXTCLR(15,1)
      CALL TMOVETO((NLTL+12),(NCTL+5))
      CALL OUTEXT('COST    data: ')
      CALL TXTCLR(14,1)
      CALL OUTEXT(FLNAME(1:(NCHR+4)))
C
C...Write FLNAME to file PERTRA.FLS on Unit 4
C
      OPEN(UNIT=4,FILE='PERTRA.FLS')
      WRITE(4,130)FLNAME(1:NCHR)
  130 FORMAT(A8)
      CLOSE(4)
C
C...Display concluding message
C
      CALL TXTCLR(15,1)
      CALL TMOVETO((NLTL+16),(NCTL+2))
      CALL OUTEXT('Run PERTRA2 for interactive input')
      CALL TMOVETO((NLTL+17),(NCTL+8))
      CALL OUTEXT('of TASK DURATION data')
      CALL TXTCLR(7,0)
      CALL TMOVETO(23,22)
      CALL OUTEXT('Program terminated')
C
C...Exit routine
C
      RETURN
      END
```

Appendix C.15: Subroutine RESCOST

```
      SUBROUTINE RESCOST(ICOST)
c***********************************************************
c**                                                       **
c**                 Subroutine RESCOST                    **
c**                                                       **
c**  Calculates individual cost components of resources  **
c**                                                       **
c**  ICOST - Index of cost component under consideration **
c**                                                       **
c***********************************************************
c
      COMMON/RES/RNAME(100),RTYPE(100),RCOST(100)
      COMMON/SUBRES/CRTOP(110),CRSUB(110),RSUBQ(110),RTOP(110),
     &RSUB(110)
      COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
C
      CHARACTER*1 RTYPE
      CHARACTER*3 RNAME,CRTOP,CRSUB
C
      INTEGER RTOP,RSUB
C
C...Initialise cost array
C
      DO 10 I=1,IR
        RCOST(I)=0.0
  10    CONTINUE
      RCOST(ICOST)=1.0
C
C...Initialise control statement LAB
C
      ASSIGN 35 TO LAB
C
C...Initialise calculation indices
C
      IND=ICOST
      I1=1
      I2=0
C
C...Rearrange SUB-RESOURCE arrays
C
  20  ISTART=I1
      DO 30 I=ISTART,IU
        IF(RSUB(I).EQ.IND)THEN
          CALL SRSWOP(I,I1)
          I1=I1+1
        ENDIF
  30    CONTINUE
C
C...Test (First time only) if cost
C...component exists as a sub-resource
C
      GO TO LAB
  35  IF(I1.LT.2)THEN
C
C...It does not exist; Exit subroutine;
C
        GO TO 50
      ELSE
C
C...It exists; Update control statement
C...to avoid unnecessary testing
C
        ASSIGN 37 TO LAB
      ENDIF
  37  I2=I2+1
      IND=RTOP(I2)
      IF(I2.LT.I1)GO TO 20
C
C...Calculate cost component for each resource
C
      DO 40 I=1,(I2-1)
        J=RTOP(I)
        K=RSUB(I)
        RCOST(J)=RCOST(J)+RCOST(K)*RSUBQ(I)
  40    CONTINUE
C
C...Exit
C
  50  RETURN
      END
```

Appendix C.16: Subroutine SRSWOP

```
      SUBROUTINE SRSWOP(I,I1)
c**************************************************************
c**                                                          **
c**                    Subroutine SRSWOP                     **
c**                                                          **
c** Sub-resource swopping routine used by Subroutine RESCOST **
c**                                                          **
c**************************************************************
c
      COMMON/SUBRES/CRTOP(110),CRSUB(110),RSUBQ(110),RTOP(110),
     &RSUB(110)
C
      CHARACTER*3 CRTOP,CRSUB
C
      INTEGER RTOP,RSUB,TEMP1,TEMP2
C
      TEMP1=RTOP(I1)
      TEMP2=RSUB(I1)
      TEMP3=RSUBQ(I1)
      RTOP(I1)=RTOP(I)
      RSUB(I1)=RSUB(I)
      RSUBQ(I1)=RSUBQ(I)
      RTOP(I)=TEMP1
      RSUB(I)=TEMP2
      RSUBQ(I)=TEMP3
C
C...Exit
C
      RETURN
      END
```

Appendix C.17: Subroutine LSORT

```
      SUBROUTINE LSORT
c*****************************************************
c**                                                 **
c**                Subroutine LSORT                 **
c**                                                 **
c** Rearranges links into the appropriate order for **
c**  the forward/backward pass through the network  **
c**                                                 **
c*****************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400),TRES(400,10),
     &TRESQ(400,10),TCAL(400),NTRES(400),SFCODE(400),TWK(400),
     &CTRES(400,10),TCOST(400,9),TWBS(400)
      COMMON/LINKS/LFROM(500,2),LTO(500,2),LTYPE(500,2),
     &LDURN(500,2)
C
      CHARACTER*1 TTYPE
      CHARACTER*3 CTRES
      CHARACTER*8 TCAL
      CHARACTER*9 TNAME
      CHARACTER*12 TWBS
C
      REAL LDURN
      INTEGER SFCODE,TWK,TRES
C
C...Rearrange links in FORWARD PASS order
C
C...Zero link counter
C
      I1=1
C
C...Search for START tasks
C
      DO 10 I=1,IL
        J=LFROM(I,1)
        IF(SFCODE(J).EQ.0)THEN
          CALL LSWOP(I,I1,1)
          I1=I1+1
        ENDIF
  10    CONTINUE
C
C...Sort remaining links into order
C
      DO 30 I2=1,IL
        ISTART=I1
        IF(ISTART.GE.IL)GO TO 40
        DO 20 I=ISTART,IL
          IF(LFROM(I,1).EQ.LTO(I2,1))THEN
            CALL LSWOP(I,I1,1)
            I1=I1+1
          ENDIF
  20      CONTINUE
  30     CONTINUE
C
C...Set up BACKWARD PASS array
C
  40  DO 50 I=1,IL
         LFROM(I,2)=LFROM(I,1)
         LTO(I,2)=LTO(I,1)
         LTYPE(I,2)=LTYPE(I,1)
         LDURN(I,2)=LDURN(I,1)
  50     CONTINUE
C
C...Rearrange links in BACKWARD PASS order
C
C...Zero link counter
C
      I1=1
C
C...Search for FINISH tasks
C
      DO 60 I=1,IL
         J=LTO(I,2)
         IF(SFCODE(J).EQ.2)THEN
           CALL LSWOP(I,I1,2)
           I1=I1+1
         ENDIF
  60     CONTINUE
C
C...Sort remaining links into order
C
      DO 80 I2=1,IL
         ISTART=I1
         IF(ISTART.GE.IL)GO TO 90
         DO 70 I=ISTART,IL
           IF(LTO(I,2).EQ.LFROM(I2,2))THEN
             CALL LSWOP(I,I1,2)
             I1=I1+1
           ENDIF
  70       CONTINUE
  80     CONTINUE
C
C...Exit
C
  90  RETURN
      END
```

Appendix C.18: Subroutine LSWOP

```
      SUBROUTINE LSWOP(I,I1,J)
c***************************************************
c**                                               **
c**                Subroutine LSWOP               **
c**                                               **
c** Link swopping routine used by Subroutine LSORT **
c**                                               **
c***************************************************
c
      COMMON/LINKS/LFROM(500,2),LTO(500,2),LTYPE(500,2),
     &LDURN(500,2)
      COMMON/COUNTS/IT,ID,IL,IR,IU,IC,IDM
C
      REAL LDURN
      INTEGER TEMP1,TEMP2,TEMP3
C
      TEMP1=LFROM(I,J)
      TEMP2=LTO(I,J)
      TEMP3=LTYPE(I,J)
      TEMP4=LDURN(I,J)
      LFROM(I,J)=LFROM(I1,J)
      LTO(I,J)=LTO(I1,J)
      LTYPE(I,J)=LTYPE(I1,J)
      LDURN(I,J)=LDURN(I1,J)
      LFROM(I1,J)=TEMP1
      LTO(I1,J)=TEMP2
      LTYPE(I1,J)=TEMP3
      LDURN(I1,J)=TEMP4
C
C...Exit
C
      RETURN
      END
```

Appendix C.19: Subroutine CHRCNV

```
      SUBROUTINE CHRCNV(CHRVAR)
c*****************************************************************
c**                                                             **
c**                      Subroutine CHRCNV                      **
c**                                                             **
c** Converts any lower case characters in CHRVAR to upper case **
c**                                                             **
c*****************************************************************
c
      CHARACTER CHRVAR*(*)
C
      NCHR=LEN(CHRVAR)
      DO 10 I=1,NCHR
        N=ICHAR(CHRVAR(I:I))
        IF(N.GT.96.AND.N.LT.123)THEN
          N=N-32
          CHRVAR(I:I)=CHAR(N)
        ENDIF
 10     CONTINUE
C
C...Exit
C
      RETURN
      END
```

Appendix C.20: Subroutine CREAD

```
      SUBROUTINE CREAD(NL,NC,CHRVAR,NCHR)
c**********************************************************
c**                                                      **
c**                 Subroutine CREAD                     **
c**                                                      **
c** Reads a character variable CHRVAR, of maximum length **
c**     NCHR characters, from the keyboard at screen     **
c**                 position NL,NC.                      **
c**                                                      **
c** Returns the actual number of characters NCHR in the  **
c**                    variable.                         **
c**                                                      **
c**              Uses 'C' subroutines.                   **
c**                                                      **
c**********************************************************
c
      CHARACTER CHRVAR*(*)
      CHARACTER*1 CHR
C
      CALL TXTCLR(7,0)
C
C...Move cursor to required position
C
      CALL TMOVETO(NL,NC)
      I=1
  10  IF(I.GT.NCHR)GO TO 30
      IC=NC+I-1
C
C...Read input character (no echo)
C
  20  CALL KBREAD(0,CHR)
      J=ICHAR(CHR)
      IF(J.EQ.13)THEN
C
C...Character is CARRIAGE RETURN;
C...Determine length of character variable; Exit routine;
C
        N=IC-NC
        GO TO 50
      ELSEIF(J.EQ.8)THEN
C
C...Character is BACKSPACE; Delete preceding character;
C...Protect against over-deletion
C
        IC=IC-1
        I=I-1
        IF(I.GT.0)GO TO 25
        I=I+1
        IC=I+1
        CALL SPCHAR(1)
        GO TO 20
  25    CALL TMOVETO(NL,IC)
```

```
        CALL OUTEXT(' ')
        CALL TMOVETO(NL,IC)
        GO TO 20
      ELSEIF(J.EQ.32)THEN
C
C...Character is SPACE; Ignore
C
        CALL SPCHAR(1)
        GO TO 20
      ELSE
C
C...Character is acceptable; Update
C...character variable and continue
C
        N=(IC-NC)+1
        CHRVAR(N:N)=CHR
        CALL OUTEXT(CHR)
      ENDIF
      I=I+1
      GO TO 10
C
C...Hide cursor
C
  30  CALL TXTCLR(0,0)
      CALL TMOVETO(25,70)
      CALL OUTEXT(' ')
      CALL TMOVETO(25,70)
  40  CALL KBREAD(0,CHR)
      J=ICHAR(CHR)
      IF(J.EQ.13)THEN
        GO TO 50
      ELSEIF(J.EQ.8)THEN
        CALL TXTCLR(7,0)
        CALL TMOVETO(NL,IC)
        CALL OUTEXT(' ')
        CALL TMOVETO(NL,IC)
        I=NCHR
        GO TO 10
      ELSE
        CALL SPCHAR(1)
        GO TO 40
      ENDIF
C
C...Exit
C
  50  NCHR=N
      RETURN
      END
```

APPENDIX D: Program PERTRA2 and Subroutines

Appendix D.1: Program PERTRA2

```
c***********************************************************
c**                                                       **
c**                 Program PERTRA2.FOR                   **
c**                                                       **
c**     Interactive input of TASK DURATIONS for use by    **
c**           Risk Analysis Program PERTRA3.FOR           **
c**                                                       **
c**           TDURN(I,1): OPTIMISTIC  duration;           **
c**           TDURN(I,2): MOST LIKELY duration;           **
c**           TDURN(I,3): PESSIMISTIC duration;           **
c**                                                       **
c** Reads initial data from user-specified file: FLNAME.TSK **
c**    generated by initialisation program: PERTRA1.FOR   **
c**                                                       **
c** Appends FLNAME.TSK to include data input from keyboard **
c**                                                       **
c***********************************************************
c
      COMMON/TASKS/TNAME(400),TWBS(400),TTYPE(400),TDURN(400,3),
     &TWK(400),SFCODE(400),FTASK(400),PCODE(400),DFACT(2)

      COMMON/FILDAT/CODE(6),POS(6),OPR(6),SELECT(6),NCHR(6),
     &LOGIC,IFLT,NTEST
C
      CHARACTER*1 TTYPE,YES
      CHARACTER*9 TNAME,SELECT
      CHARACTER*12 TWBS,CVAR,FLNAME
C
      REAL TDURN
      INTEGER FTASK,PCODE,CODE,POS,OPR,NCHR,SFCODE,TWK,IFLT,NTEST
C
C...Read FLNAME from file: PERTRA.FLS on Unit 3
C
      OPEN(UNIT=3,FILE='PERTRA.FLS')
      READ(3,10)FLNAME
   10 FORMAT(A8)
      CLOSE(3)
C
C...Set TEXT mode; display program introduction
C
      CALL SETMODE(0)
      CALL HEADER(FLNAME)
```

```
C
C...Read TASK data from FLNAME on Unit 3
C
      FLNAME(9:12)='.TSK'
      OPEN(UNIT=3,FILE=FLNAME)
      READ(3,*)NTASK
      READ(3,20)(TTYPE(I),TNAME(I),TWBS(I),TWK(I),SFCODE(I),
     &PCODE(I),(TDURN(I,J),J=1,3),I=1,NTASK)
   20 FORMAT(1X,A1,1X,A9,1X,A12,3I2,3F7.2)
      CLOSE(3)
C
C...Set HAMMOCK/MILESTONE task codes set to zero
C
      DO 30 I=1,NTASK
        IF(TTYPE(I).EQ.'H'.OR.TTYPE(I).EQ.'M')PCODE(I)=0
   30   CONTINUE
C
C...Display screen enquiry
C
   40 CALL QDISPLAY(JCODE,FLNAME,NTASK)
C
C...Quit program if so desired; Otherwise continue
C
      IF(JCODE.GT.1)GO TO 999
C
C...Input filtration parameters from keyboard
C
   50 CALL FDISPLAY(JCODE)
C
C...Filter tasks in accordance with filtration parameters
C
      CALL FILTER(NTASK)
C
C...Display filtered tasks on screen
C
      CALL TDISPLAY(ICODE,JCODE)
C
C...If filtration unsatisfactory, input new filtration parameters
C
      IF(ICODE.GT.0)GO TO 50
C
C...Filtration satisfactory; Update the appropriate variables
C
      IF(JCODE.EQ.0)THEN
C
C...INVARIABLE duration tasks; Update task codes PCODE
C
        DO 60 I=1,IFLT
          J=FTASK(I)
          PCODE(J)=0
   60     CONTINUE
      ELSE
```

```
C
C...VARIABLE duration tasks; Update OPTIMISTIC/PESSIMISTIC durations
C
        DO 70 I=1,IFLT
          J=FTASK(I)
          TDURN(J,1)=TDURN(J,2)*DFACT(1)
          TDURN(J,3)=TDURN(J,2)*DFACT(2)
  70      CONTINUE
      ENDIF
C
C...Repeat for next set of tasks
C
      GO TO 40
C
C...Program ends
C
 999  END
```

Appendix D.2: Subroutine HEADER

```
      SUBROUTINE HEADER(FLNAME)
c*******************************************
c**                                       **
c**          Subroutine HEADER            **
c**                                       **
c** Outputs program introductory message **
c**               to screen               **
c**                                       **
c*******************************************
c
      CHARACTER*1 CHR
      CHARACTER*12 FLNAME
      CHARACTER*61 TXTREC(45)
      CHARACTER*65 TABREC(21)
C
C...Initialise TABLE records
C
      DO 20 I=1,23
        TABREC(I)(1:1)=CHAR(186)
        TABREC(I)(65:65)=CHAR(186)
        DO 10 J=2,64
          TABREC(I)(J:J)=' '
  10      CONTINUE
  20    CONTINUE
      TABREC(1)(1:1)=CHAR(201)
      TABREC(1)(65:65)=CHAR(187)
      TABREC(5)(1:1)=CHAR(204)
      TABREC(5)(65:65)=CHAR(185)
      TABREC(21)(1:1)=CHAR(200)
      TABREC(21)(65:65)=CHAR(188)
      DO 30 J=2,64
        TABREC(1)(J:J)=CHAR(205)
        TABREC(5)(J:J)=CHAR(205)
        TABREC(21)(J:J)=CHAR(205)
  30    CONTINUE
      TABREC(3)(24:38)='Program PERTRA2'

C
C...Initialise TEXT records
C
      DO 40 I=1,35
        DO 35 J=1,61
          TXTREC(I)(J:J)=' '
  35      CONTINUE
  40    CONTINUE
      TXTREC(1)(1:43)='Interactive input of TASK DURATION data for'
      TXTREC(1)(45:59)='the Monte-Carlo'
      TXTREC(2)(1:42)='simulation carried out by Program PERTRA3.'
      TXTREC(4)(1:44)='Basic task data is read from the appropriate'
      TXTREC(4)(46:54)='.TSK file'
      TXTREC(5)(1:29)='generated by Program PERTRA1.'
      TXTREC(5)(31:57)='Following input of the task'
      TXTREC(6)(1:46)='duration data, this file will be updated, thus'
      TXTREC(6)(48:60)='providing the'
      TXTREC(7)(1:43)='facility for performing intermediate checks'
      TXTREC(7)(45:61)='on the data file.'
      TXTREC(9)(1:36)='For convenience, PERTRA2 facilitates'
      TXTREC(9)(38:57)='SELECTIVE display of'
      TXTREC(10)(1:22)='tasks on the basis of:'
      TXTREC(12)(15:28)='(a) Task NAME;'
      TXTREC(13)(15:48)='(b) Task WORK BREAKDOWN STRUCTURE;'
      TXTREC(14)(15:28)='(c) Task TYPE;'
      TXTREC(16)(1:43)='For the purpose of data input, tasks may be'
      TXTREC(16)(45:58)='categorised as'
      TXTREC(17)(1:8)='follows:'
      TXTREC(19)(1:42)='Type 1: Those whose durations are VARIABLE'
      TXTREC(19)(44:56)='in accordance'
      TXTREC(20)(9:51)='with a probability distribution defined by:'
      TXTREC(22)(9:31)='(a) OPTIMISTIC duration'
      TXTREC(22)(33:61)='(99% exceedance probability);'
      TXTREC(24)(9:32)='(b) MOST LIKELY duration'
      TXTREC(24)(34:60)='(used in PMA calculations);'
      TXTREC(26)(9:32)='(c) PESSIMISTIC duration'
      TXTREC(26)(34:61)='(1% exceedance probability);'
      TXTREC(28)(1:44)='Type 2: Those whose durations are INVARIABLE'
      TXTREC(28)(46:56)='and are not'
      TXTREC(29)(9:55)='subject to a probability distribution as above;'
      TXTREC(31)(1:42)='Note 1: MILESTONE tasks will automatically'
      TXTREC(31)(44:61)='be included within'
      TXTREC(32)(9:19)='category 2;'
      TXTREC(34)(1:34)='Note 2: By virtue of their nature,'
      TXTREC(34)(36:61)='HAMMOCK tasks will also be'
      TXTREC(35)(9:49)='automatically included within category 2;'
      DO 50 I=1,61
        TXTREC(3)(I:I)=CHAR(196)
        TXTREC(8)(I:I)=CHAR(196)
        TXTREC(15)(I:I)=CHAR(196)
        TXTREC(30)(I:I)=CHAR(196)
        TXTREC(36)(I:I)=CHAR(196)
50      CONTINUE
      DO 70 I=37,45
        DO 60 J=1,61
          TXTREC(I)(J:J)=CHAR(176)
60        CONTINUE
70      CONTINUE
```

```
C
C...Draw table
C
      CALL SCLEAR
      CALL SETWIN(1,1,25,80)
      CALL TXTCLR(15,1)
      DO 80 I=1,21
        CALL TMOVETO(I,8)
        IF(I.LT.2.OR.I.GT.4)THEN
          CALL OUTEXT(TABREC(I))
        ELSE
          CALL OUTEXT(TABREC(I)(1:1))
          CALL TXTCLR(0,6)
          CALL OUTEXT(TABREC(I)(2:64))
          CALL TXTCLR(15,1)
          CALL OUTEXT(TABREC(I)(65:65))
        ENDIF
   80   CONTINUE
C
C...Set text window
C
      CALL SETWIN(6,10,20,71)
C
C...Output text to screen in 15 line sets
C
      DO 90 I=1,45
        IF(MOD(I,15).GT.0)THEN
          CALL OUTEXT(TXTREC(I)//CHAR(10))
        ELSE
          CALL OUTEXT(TXTREC(I))
          CALL SETWIN(1,1,25,80)
          CALL TMOVETO(22,8)
          CALL TXTCLR(0,4)
          CALL OUTEXT(' Hit any key to continue ')
          CALL TXTCLR(0,0)
          CALL TMOVETO(25,75)
          CALL OUTEXT(' ')
          CALL TMOVETO(25,75)
          CALL KBREAD(0,CHR)
          IF(I.EQ.45)GO TO 100
          CALL TXTCLR(15,0)
          CALL TMOVETO(22,8)
          CALL OUTEXT(TABREC(6)(2:64))
          CALL TXTCLR(15,1)
          CALL SETWIN(6,10,20,71)
          CALL TMOVETO(15,62)
          CALL OUTEXT(CHAR(10))
        ENDIF
   90   CONTINUE
```

```
C
C...All text output; Display concluding message and exit routine
C
  100 CALL TXTCLR(7,0)
      DO 110 I=1,8
        J=8-(I-1)
        IF(FLNAME(J:J).NE.' ')GO TO 120
  110   CONTINUE
  120 CALL TMOVETO(22,8)
      CALL OUTEXT('Reading Task Duration data from file: ')
      CALL TXTCLR(15,0)
      CALL OUTEXT(FLNAME(1:J)//'.TSK')
      RETURN
      END
```

Appendix D.3: Subroutine QDISPLAY

```
      SUBROUTINE QDISPLAY(JCODE,FLNAME,NTASK)
c***********************************************************
c**                                                       **
c**                 Subroutine QDISPLAY                   **
c**                                                       **
c**              Intermediate enquiry display             **
c**                                                       **
c** JCODE = 0: Input data for INVARIABLE duration tasks; **
c** JCODE - 1: Input data for VARIABLE duration tasks;   **
c** JCODE = 2; No more data to input; Quit program;      **
c**                                                       **
c***********************************************************
c
      COMMON/TASKS/TNAME(400),TWBS(400),TTYPE(400),TDURN(400,3),
     &TWK(400),SFCODE(400),FTASK(400),PCODE(400),DFACT(2)
C
      CHARACTER*1 CHR1,TTYPE
      CHARACTER*9 TNAME
      CHARACTER*12 FLNAME,TWBS
      CHARACTER*40 TABREC(22)
C
      INTEGER FTASK,PCODE,SFCODE,TWK
C
C...Specify highlight delay
C
      NDEL=30
C
C...Initialise TABLE records
C
      DO 20 I=1,9
        TABREC(I)(1:1)=CHAR(186)
        TABREC(I)(38:38)=CHAR(186)
        DO 10 J=2,37
          TABREC(I)(J:J)=' '
 10       CONTINUE
 20     CONTINUE
      TABREC(1)(1:1)=CHAR(201)
      TABREC(1)(38:38)=CHAR(187)
      TABREC(5)(1:1)=CHAR(204)
      TABREC(5)(38:38)=CHAR(185)
      TABREC(9)(1:1)=CHAR(200)
      TABREC(9)(38:38)=CHAR(188)
      DO 30 J=2,37
        TABREC(1)(J:J)=CHAR(205)
        TABREC(5)(J:J)=CHAR(205)
        TABREC(9)(J:J)=CHAR(205)
 30     CONTINUE
      TABREC(3)(11:28)='TASK DURATION DATA'
```

```
C
C...Draw table
C
      CALL SCLEAR
      CALL SETWIN(1,1,25,80)
      CALL TXTCLR(15,1)
      NLTL=7
      NCTL=22
      DO 40 I=1,9
        CALL TMOVETO((NLTL+I-1),NCTL)
        IF(I.LT.2.OR.I.GT.4)THEN
          CALL OUTEXT(TABREC(I)(1:38))
        ELSE
          CALL OUTEXT(TABREC(I)(1:1))
          CALL TXTCLR(0,6)
          CALL OUTEXT(TABREC(I)(2:37))
          CALL TXTCLR(15,1)
          CALL OUTEXT(TABREC(I)(38:38))
        ENDIF
  40    CONTINUE
  45  CALL TXTCLR(15,1)
      CALL TMOVETO((NLTL+6),(NCTL+2))
      CALL OUTEXT('Task Duration: ')
      CALL TXTCLR(7,1)
      CALL OUTEXT('VARIABLE/INVARIABLE')
C
C...Display 'OPTION' message with highlights; Hide cursor;
C
      NLTOP=NLTL-1
      NCTOP=NCTL+1
      CALL TXTCLR(15,0)
      CALL TMOVETO(NLTOP,NCTOP)
      CALL OUTEXT('Option:  INPUT  QUIT')
      CALL TXTCLR(4,0)
      CALL TMOVETO(NLTOP,(NCTOP+9))
      CALL OUTEXT('I')
      CALL TMOVETO(NLTOP,(NCTOP+16))
      CALL OUTEXT('Q')
      CALL TXTCLR(0,0)
      CALL TMOVETO(25,75)
      CALL OUTEXT(' ')
      CALL TMOVETO(25,75)
C
C...Read input from keyboard; Convert to upper case if required;
C
  50  CALL KBREAD(0,CHR1)
      CALL CHRCNV(CHR1)
      IF(CHR1.EQ.'I')THEN
C
C...Option is INPUT; Highlight and proceed;
C
        CALL TXTCLR(0,7)
        DO 60 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+9))
          CALL OUTEXT('INPUT')
  60      CONTINUE
      ELSEIF(CHR1.EQ.'Q')THEN
C
C...Option is QUIT; Highlight; Update JCODE
C...Move to 'quitting' routine;
C
        CALL TXTCLR(0,7)
        DO 70 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+16))
          CALL OUTEXT('QUIT')
  70      CONTINUE
        JCODE=2
        GO TO 140
      ELSE
C
C...Inappropriate character; sound warning and try again;
C
        CALL SPCHAR(1)
        GO TO 50
      ENDIF
C
C...Display TASK DURATION message with highlights
C
      CALL TXTCLR(15,0)
      CALL TMOVETO(NLTOP,NCTOP)
      CALL OUTEXT('Task Duration:  VARIABLE  INVARIABLE')
      CALL TXTCLR(4,0)
      CALL TMOVETO(NLTOP,(NCTOP+16))
      CALL OUTEXT('V')
      CALL TMOVETO(NLTOP,(NCTOP+26))
      CALL OUTEXT('I')
      CALL TXTCLR(0,0)
      CALL TMOVETO(25,75)
      CALL OUTEXT(' ')
      CALL TMOVETO(25,75)
C
C...Read input from keyboard; Convert to upper case if required;
C
  80  CALL KBREAD(0,CHR1)
      CALL CHRCNV(CHR1)
      IF(CHR1.EQ.'V')THEN
C
C...Option is VARIABLE; Highlight; Update JCODE
C
        CALL TXTCLR(0,7)
        DO 90 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+16))
          CALL OUTEXT('VARIABLE')
  90      CONTINUE
        JCODE=1
```

```
C
C...Highlight tabular display and proceed
C
          CALL TXTCLR(0,4)
          CALL TMOVETO((NLTL+6),(NCTL+17))
          CALL OUTEXT('VARIABLE')
        ELSEIF(CHR1.EQ.'I')THEN
C
C...Option is INVARIABLE; Highlight; Update JCODE;
C
          CALL TXTCLR(0,7)
          DO 100 M=1,NDEL
            CALL TMOVETO(NLTOP,(NCTOP+26))
            CALL OUTEXT('INVARIABLE')
  100       CONTINUE
          JCODE=0
C
C...Highlight tabular display and proceed
C
          CALL TXTCLR(0,4)
          CALL TMOVETO((NLTL+6),(NCTL+26))
          CALL OUTEXT('INVARIABLE')
        ELSE
C
C...Inappropriate character; sound warning and try again;
C
          CALL SPCHAR(1)
          GO TO 80
        ENDIF
C
C...Display CONTINUE message; Highlight options
C
        CALL TXTCLR(0,0)
        CALL TMOVETO(NLTOP,NCTOP)
        CALL OUTEXT(TABREC(2)(2:37))
        CALL TXTCLR(15,0)
        CALL TMOVETO(NLTOP,NCTOP)
        CALL OUTEXT('Continue?  YES  NO')
        CALL TXTCLR(4,0)
        CALL TMOVETO(NLTOP,(NCTOP+11))
        CALL OUTEXT('Y')
        CALL TMOVETO(NLTOP,(NCTOP+16))
        CALL OUTEXT('N')
        CALL TXTCLR(0,0)
        CALL TMOVETO(25,75)
        CALL OUTEXT(' ')
        CALL TMOVETO(25,75)
C
C...Read input from keyboard; Convert to upper case if required;
C
  110   CALL KBREAD(0,CHR1)
        CALL CHRCNV(CHR1)
        IF(CHR1.EQ.'N')THEN
```

```
C
C...Option is DISCONTINUE; Highlight and start again
C
        CALL TXTCLR(0,7)
        DO 120 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+16))
          CALL OUTEXT('NO')
  120     CONTINUE
        GO TO 45
      ELSEIF(CHR1.EQ.'Y')THEN
C
C...Option is CONTINUE; Highlight and exit routine
C
        CALL TXTCLR(0,7)
        DO 130 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+11))
          CALL OUTEXT('YES')
  130     CONTINUE
        CALL TXTCLR(7,0)
        GO TO 999
      ELSE
C
C...Inappropriate character; sound warning and try again;
C
        CALL SPCHAR(1)
        GO TO 110
      ENDIF
C
C...Quitting routine
C
  140 CALL TXTCLR(7,0)
      CALL SCLEAR
C
C...Initialise display table records
C
      DO 160 I=1,22
        TABREC(I)(1:1)=CHAR(186)
        TABREC(I)(40:40)=CHAR(186)
        DO 150 J=2,39
          TABREC(I)(J:J)=' '
  150     CONTINUE
  160   CONTINUE
      TABREC(1)(1:1)=CHAR(201)
      TABREC(1)(40:40)=CHAR(187)
      TABREC(5)(1:1)=CHAR(200)
      TABREC(5)(40:40)=CHAR(188)
      TABREC(6)(1:1)=CHAR(201)
      TABREC(6)(40:40)=CHAR(187)
      TABREC(12)(1:1)=CHAR(199)
      TABREC(12)(40:40)=CHAR(182)
      TABREC(17)(1:1)=CHAR(199)
      TABREC(17)(40:40)=CHAR(182)
      TABREC(22)(1:1)=CHAR(200)
      TABREC(22)(40:40)=CHAR(188)
      DO 170 J=2,39
```

```
        TABREC(1)(J:J)=CHAR(205)
        TABREC(5)(J:J)=CHAR(205)
        TABREC(6)(J:J)=CHAR(205)
        TABREC(12)(J:J)=CHAR(196)
        TABREC(17)(J:J)=CHAR(196)
        TABREC(22)(J:J)=CHAR(205)
 170    CONTINUE
      TABREC(3)(11:30)='Quitting PERTRA2 !!!'
      TABREC(8)(3:38)='You may now check your input data in'
      DO 180 I=1,8
        J=8-(I-1)
        IF(FLNAME(J:J).NE.' ')THEN
          GO TO 190
        ENDIF
 180    CONTINUE
 190  TABREC(10)(12:(J+21))='File: '//FLNAME(1:J)//'.TSK'
      TABREC(14)(4:36)='Task Duration Data may be amended'
      TABREC(15)(4:36)='or updated by re-running PERTRA2;'
      TABREC(19)(4:36)='The Monte-Carlo simulation may be'
      TABREC(20)(4:34)='carried out by running PERTRA3;'
C
C...Draw 'quitting' display; write data to FLNAME on Unit 4
C
      CALL TXTCLR(15,4)
      NLTL=1
      NCTL=21
      DO 200 I=1,5
        CALL TMOVETO((NLTL+I-1),NCTL)
        IF(I.LT.2.OR.I.GT.4)THEN
          CALL OUTEXT(TABREC(I))
        ELSE
          CALL OUTEXT(TABREC(I)(1:1))
          CALL TXTCLR(0,4)
          CALL OUTEXT(TABREC(I)(2:39))
          CALL TXTCLR(15,4)
          CALL OUTEXT(TABREC(I)(40:40))
        ENDIF
 200    CONTINUE
      CALL TXTCLR(0,0)
      CALL TMOVETO(24,75)
      CALL OUTEXT(' ')
      CALL TMOVETO(24,75)
      OPEN(UNIT=4,FILE=FLNAME)
      WRITE(4,210)NTASK
 210  FORMAT(I4)
      WRITE(4,220)(TTYPE(I),TNAME(I),TWBS(I),TWK(I),SFCODE(I),
     &PCODE(I),(TDURN(I,J),J=1,3),I=1,NTASK)
 220  FORMAT(1X,A1,1X,A9,1X,A12,3I2,3F7.2)
      CLOSE(4)
      CALL TXTCLR(15,1)
      DO 230 I=6,22
        CALL TMOVETO((NLTL+I),NCTL)
        CALL OUTEXT(TABREC(I))
 230    CONTINUE
      CALL TXTCLR(7,0)
      CALL TMOVETO((NLTL+23),NCTL)
      CALL OUTEXT('Program terminated')
C
C...Exit routine
C
  999 RETURN
      END
```

Appendix D.4: Subroutine FDISPLAY

```
      SUBROUTINE FDISPLAY(JCODE)
c******************************************************
c**                                                  **
c**               Subroutine FDISPLAY                **
c**                                                  **
c** Screen display for input of task filtration data **
c**                                                  **
c******************************************************
c
      COMMON/FILDAT/CODE(6),POS(6),OPR(6),SELECT(6),
     &NCHR(6),LOGIC,IFLT,NTEST
c
      CHARACTER*1 CHR,INTCHR
      CHARACTER*9 SELECT,BLANK
      CHARACTER*38 TABREC(11),TOPREC(7)
c
      INTEGER CODE,POS,OPR,NCHR,LOGIC,IFLT,NTEST
C
C...Specify display colours
C
C...Table borders and headings (BRIGHT WHITE)
C
      ICLR1=15
C
C...Table background (BLUE)
C
      ICLR2=1
C
C...Table text (WHITE)
C
      ICLR3=7
C
C...Table location highlight (BROWN)
C
      ICLR4=6
C
C...'Backspace' display (RED)
C
      ICLR5=4
C
C...Specify 'reverse video' delay
C
      NDEL=30
C
C...Initialise display strings
C
      TOPREC(1)='FILTER   ; CODE:  NAME  WBS  TYPE  QUIT'
      TOPREC(2)='FILTER   ; POSITION:  [ ]              '
      TOPREC(3)='FILTER   ; OPERAND:  EQ  NE            '
      TOPREC(4)='FILTER   ; SELECTION:  [         ]     '
      TOPREC(5)='CONTINUE ?  YES  NO                    '
      TOPREC(6)='LOGIC:  AND  OR                        '
      TOPREC(7)='GO  CANCEL                  '
C
C...Initialise table strings
C
      DO 20 I=1,11
        TABREC(I)(1:1)=CHAR(186)
        TABREC(I)(38:38)=CHAR(186)
        DO 10 J=2,37
          TABREC(I)(J:J)=' '
  10      CONTINUE
  20    CONTINUE
      TABREC(1)(1:1)=CHAR(201)
      TABREC(1)(38:38)=CHAR(187)
      TABREC(9)(1:1)=CHAR(199)
      TABREC(9)(38:38)=CHAR(182)
      TABREC(11)(1:1)=CHAR(200)
      TABREC(11)(38:38)=CHAR(188)
      DO 30 I=2,37
        TABREC(1)(I:I)=CHAR(205)
        TABREC(9)(I:I)=CHAR(196)
        TABREC(11)(I:I)=CHAR(205)
  30    CONTINUE
      TABREC(1)(17:22)='Filter'
      TABREC(2)(3:36)='FILTER  CODE  POS  OPRN  SELECTION'
      TABREC(10)(15:20)='LOGIC:'
      DO 60 I=1,6
        TABREC(I+2)(5:5)=INTCHR(I)
  60    CONTINUE
      BLANK='      '
C
C...Define text window
C
      CALL SETWIN(1,1,25,80)
C
C...Define position of display
C
      NLTL=6
      NCTL=22
C
C...Draw appropriate heading (BLACK on CYAN)
C
      CALL TXTCLR(7,0)
      CALL SCLEAR
      CALL TXTCLR(0,3)
      IF(JCODE.EQ.0)THEN
        CALL TMOVETO(1,(NCTL+3))
        CALL OUTEXT(' Tasks with INVARIABLE durations ')
      ELSE
        CALL TMOVETO(1,(NCTL+4))
        CALL OUTEXT(' Tasks with VARIABLE durations ')
      ENDIF
```

```
C
C...Draw filtration table
C
      CALL TXTCLR(ICLR1,ICLR2)
      DO 70 I=1,11
        CALL TMOVETO((NLTL+I-1),NCTL)
        CALL OUTEXT(TABREC(I))
  70    CONTINUE
C
C...Write out input line (BRIGHT WHITE on BLACK)
C
  80  NLTOP=NLTL-1
      NCTOP=NCTL
      DO 160 I=1,6
        NLTAB=NLTL+I+1
        DO 150 J=1,4
          TOPREC(J)(8:8)=INTCHR(I)
          CALL TMOVETO(NLTOP,NCTOP)
          CALL TXTCLR(15,0)
          CALL OUTEXT(TOPREC(J))
          IF(J.GT.1)GO TO 100
C
C...CODE display; Highlight first character of options (RED on BLACK)
C
          NCTAB=NCTL+10
          NB=4
          CALL TXTCLR(4,0)
          CALL TMOVETO(NLTOP,(NCTOP+17))
          CALL OUTEXT('N')
          CALL TMOVETO(NLTOP,(NCTOP+23))
          CALL OUTEXT('W')
          CALL TMOVETO(NLTOP,(NCTOP+28))
          CALL OUTEXT('T')
          CALL TMOVETO(NLTOP,(NCTOP+34))
          CALL OUTEXT('Q')
C
C...Highlight table location
C
          CALL TMOVETO(NLTAB,NCTAB)
          CALL TXTCLR(0,ICLR4)
          CALL OUTEXT(BLANK(1:NB))
C
C...Hide cursor
C
          CALL TXTCLR(0,0)
          CALL TMOVETO(25,70)
          CALL OUTEXT(' ')
          CALL TMOVETO(25,70)
C
C...Read keyboard input; convert to upper case if required
C
  90      CALL KBREAD(0,CHR)
          CALL CHRCNV(CHR)
          IF(CHR.EQ.'N')THEN
```

```
C
C...Option is 'NAME'; Define CODE(I);
C...Highlight option (BLACK on WHITE); Update table;
C
            CODE(I)=1
            CALL TXTCLR(0,7)
            DO 92 M=1,NDEL
              CALL TMOVETO(NLTOP,(NCTOP+17))
              CALL OUTEXT('NAME')
  92          CONTINUE
            CALL TMOVETO(NLTAB,NCTAB)
            CALL TXTCLR(ICLR3,ICLR2)
            CALL OUTEXT('NAME')
          ELSEIF(CHR.EQ.'W')THEN
C
C...Option is 'WBS'; Define CODE(I);
C...Highlight option (BLACK on WHITE); Update table;
C
            CODE(I)=2
            CALL TXTCLR(0,7)
            DO 94 M=1,NDEL
              CALL TMOVETO(NLTOP,(NCTOP+23))
              CALL OUTEXT('WBS')
  94          CONTINUE
            CALL TMOVETO(NLTAB,NCTAB)
            CALL TXTCLR(ICLR3,ICLR2)
            CALL OUTEXT('WBS ')
          ELSEIF(CHR.EQ.'T')THEN
C
C...Option is 'TYPE'; Define CODE(I); Default POSITION is 1
C...Highlight option (BLACK on WHITE); Update table;
C
            CODE(I)=3
            POS(I)=1
            CALL TXTCLR(0,7)
            DO 96 M=1,NDEL
              CALL TMOVETO(NLTOP,(NCTOP+28))
              CALL OUTEXT('TYPE')
  96          CONTINUE
            CALL TMOVETO(NLTAB,NCTAB)
            CALL TXTCLR(ICLR3,ICLR2)
            CALL OUTEXT('TYPE   1')
          ELSEIF(CHR.EQ.'Q')THEN
C
C...Option is 'QUIT'; Highlight option (BLACK on WHITE);
C...Move to 'quitting' routine
C
            CALL TXTCLR(0,7)
            DO 98 M=1,NDEL
              CALL TMOVETO(NLTOP,(NCTOP+34))
              CALL OUTEXT('QUIT')
  98          CONTINUE
            CALL TMOVETO(NLTAB,NCTAB)
            CALL TXTCLR(0,ICLR2)
            CALL OUTEXT(BLANK(1:4))
          GO TO 170
        ELSE
C
C...Inappropriate character; try again
C
          CALL SPCHAR(1)
          GO TO 90
        ENDIF
        GO TO 150
 100    IF(J.GT.2)GO TO 120
        IF(CODE(I).EQ.3)GO TO 150
C
C...POSITION display;
C
        NCTAB=NCTL+17
        NB=1
C
C...Highlight table location
C
        CALL TMOVETO(NLTAB,NCTAB)
        CALL TXTCLR(0,ICLR4)
        CALL OUTEXT(BLANK(1:NB))
C
C...Read data input from keyboard; Update table
C
        CALL TMOVETO(NLTOP,(NCTOP+22))
 110    CALL KBREAD(0,CHR)
        ICHR=ICHAR(CHR)
        IF(ICHR.LT.49.OR.ICHR.GT.57)THEN
          CALL SPCHAR(1)
          GO TO 110
        ELSE
          CALL OUTEXT(CHR)
        ENDIF
        POS(I)=ICHR-48
        CALL TMOVETO(NLTAB,NCTAB)
        CALL TXTCLR(ICLR3,ICLR2)
        CALL OUTEXT(CHR)
        GO TO 150
 120    IF(J.GT.3)GO TO 140
C
C...OPERAND display; Highlight first character
C...of options (RED on BLACK)
C
        NCTAB=NCTL+22
        NB=2
        CALL TXTCLR(4,0)
        CALL TMOVETO(NLTOP,(NCTOP+20))
        CALL OUTEXT('E')
        CALL TMOVETO(NLTOP,(NCTOP+24))
        CALL OUTEXT('N')
```

```
C
C...Highlight table location
C
      CALL TMOVETO(NLTAB,NCTAB)
      CALL TXTCLR(0,ICLR4)
      CALL OUTEXT(BLANK(1:NB))
C
C...Hide cursor
C
      CALL TXTCLR(0,0)
      CALL TMOVETO(25,70)
      CALL OUTEXT(' ')
      CALL TMOVETO(25,70)
C
C...Read keyboard input; convert to upper case if required
C
 130  CALL KBREAD(0,CHR)
      CALL CHRCNV(CHR)
      IF(CHR.EQ.'E')THEN
C
C...Option is 'EQUAL TO'; Define OPR(I);
C...Highlight option (BLACK on WHITE); Update table;
C
       OPR(I)=1
       CALL TXTCLR(0,7)
       DO 132 M=1,NDEL
        CALL TMOVETO(NLTOP,(NCTOP+20))
        CALL OUTEXT('EQ')
 132    CONTINUE
       CALL TMOVETO(NLTAB,NCTAB)
       CALL TXTCLR(ICLR3,ICLR2)
       CALL OUTEXT('EQ')
      ELSEIF(CHR.EQ.'N')THEN
C
C...Option is 'NOT EQUAL TO'; Define OPR(I);
C...Highlight option (BLACK on WHITE); Update table;
C
       OPR(I)=2
       CALL TXTCLR(0,7)
       DO 134 M=1,NDEL
        CALL TMOVETO(NLTOP,(NCTOP+24))
        CALL OUTEXT('NE')
 134    CONTINUE
       CALL TMOVETO(NLTAB,NCTAB)
       CALL TXTCLR(ICLR3,ICLR2)
       CALL OUTEXT('NE')
      ELSE
C
C...Inappropriate character; try again
C
       CALL SPCHAR(1)
       GO TO 130
      ENDIF
      GO TO 150
```

```
C
C...SELECTION display;
C
C...Highlight table location
C
  140     NCTAB=NCTL+27
          NB=9
          CALL TMOVETO(NLTAB,NCTAB)
          CALL TXTCLR(0,ICLR4)
          CALL OUTEXT(BLANK(1:NB))
C
C...Read data input from keyboard; update table
C
          CALL TMOVETO((NLTL+11),NCTL)
          CALL TXTCLR(0,ICLR5)
          CALL OUTEXT('      Use BACKSPACE key to erase      ')
          NCHR(I)=9
          CALL CREAD(NLTOP,(NCTOP+23),SELECT(I),NCHR(I))
          CALL CHRCNV(SELECT(I))
          CALL TMOVETO(NLTAB,NCTAB)
          CALL TXTCLR(ICLR3,ICLR2)
          NB=NB-NCHR(I)
          IF(NB.GT.0)THEN
            CALL OUTEXT(SELECT(I)(1:NCHR(I))//BLANK(1:NB))
          ELSE
            CALL OUTEXT(SELECT(I)(1:NCHR(I)))
          ENDIF
          CALL TMOVETO((NLTL+11),NCTL)
          CALL TXTCLR(7,0)
          CALL OUTEXT('                                      ')
  150     CONTINUE
  160   CONTINUE
C
C...'Quitting' routine; write CONTINUE display (WHITE on BLACK);
C...Highlight first character of options (RED on BLACK)
C
 170  NTEST=I-1
      IF(NTEST.LT.2)THEN
        LOGIC=1
        GO TO 200
      ENDIF
      CALL TMOVETO(NLTOP,NCTOP)
      CALL TXTCLR(15,0)
      CALL OUTEXT(TOPREC(5))
      CALL TXTCLR(4,0)
      CALL TMOVETO(NLTOP,(NCTOP+12))
      CALL OUTEXT('Y')
      CALL TMOVETO(NLTOP,(NCTOP+17))
      CALL OUTEXT('N')
```

```
C
C...Hide cursor
C
          CALL TXTCLR(0,0)
          CALL TMOVETO(25,70)
          CALL OUTEXT(' ')
          CALL TMOVETO(25,70)
C
C...Read keyboard input; convert to upper case if required
C
  180 CALL KBREAD(0,CHR)
      CALL CHRCNV(CHR)
      IF(CHR.EQ.'N')THEN
C
C...Option is 'NO'; highlight option (BLACK on WHITE); Start again
C
        CALL TXTCLR(0,7)
        DO 182 M-1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+17))
          CALL OUTEXT('NO')
  182     CONTINUE
        CALL TXTCLR(ICLR3,ICLR2)
        DO 183 M=1,NTEST
          CALL TMOVETO((NLTL+M+1),(NCTL+8))
          CALL OUTEXT(BLANK//BLANK//BLANK)
  183     CONTINUE
        GO TO 80
      ELSEIF(CHR.EQ.'Y')THEN
C
C...Option is 'YES'; highlight option (BLACK on WHITE); Continue
C
        CALL TXTCLR(0,7)
        DO 184 M-1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+12))
          CALL OUTEXT('YES')
  184     CONTINUE
      ELSE
C
C...Inappropriate character; try again
C
        CALL SPCHAR(1)
        GO TO 180
      ENDIF
C
C...Write LOGIC display (WHITE on BLACK);
C...Highlight first character of options (RED on BLACK)
C
      NLTAB=NLTL+9
      NCTAB=NCTL+21
      NB=3
      CALL TMOVETO(NLTOP,NCTOP)
      CALL TXTCLR(15,0)
      CALL OUTEXT(TOPREC(6))
      CALL TXTCLR(4,0)
      CALL TMOVETO(NLTOP,(NCTOP+8))
      CALL OUTEXT('A')
      CALL TMOVETO(NLTOP,(NCTOP+13))
      CALL OUTEXT('O')
C
C...Highlight table location
C
          CALL TMOVETO(NLTAB,NCTAB)
          CALL TXTCLR(0,ICLR4)
          CALL OUTEXT(BLANK(1:NB))
C
C...Hide cursor
C
          CALL TXTCLR(0,0)
          CALL TMOVETO(25,70)
          CALL OUTEXT(' ')
          CALL TMOVETO(25,70)
C
C...Read keyboard input; convert to upper case if required
C
  190 CALL KBREAD(0,CHR)
      CALL CHRCNV(CHR)
      IF(CHR.EQ.'A')THEN
C
C...Option is 'AND'; Define LOGIC(I);
C...Highlight option (BLACK on WHITE); Update table;
C
        LOGIC=1
        CALL TXTCLR(0,7)
        DO 192 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+8))
          CALL OUTEXT('AND')
  192     CONTINUE
        CALL TMOVETO(NLTAB,NCTAB)
        CALL TXTCLR(ICLR3,ICLR2)
        CALL OUTEXT('AND')
      ELSEIF(CHR.EQ.'O')THEN
C
C...Option is 'OR'; Define LOGIC(I);
C...Highlight option (BLACK on WHITE); Update table;
C
        LOGIC=2
        CALL TXTCLR(0,7)
        DO 194 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+13))
          CALL OUTEXT('OR')
  194     CONTINUE
        CALL TMOVETO(NLTAB,NCTAB)
        CALL TXTCLR(ICLR3,ICLR2)
        CALL OUTEXT('OR ')
      ELSE
```

```
C
C...Inappropriate character; try again
C
        CALL SPCHAR(1)
        GO TO 190
      ENDIF
C
C...Write GO/CANCEL display (WHITE on BLACK)
C...Highlight first character of options (RED on BLACK)
C
 200  CALL TMOVETO(NLTOP,NCTOP)
      CALL TXTCLR(15,0)
      CALL OUTEXT(TOPREC(7))
      CALL TXTCLR(4,0)
      CALL TMOVETO(NLTOP,NCTOP)
      CALL OUTEXT('G')
      CALL TMOVETO(NLTOP,(NCTOP+4))
      CALL OUTEXT('C')
C
C...Hide cursor
C
          CALL TXTCLR(0,0)
          CALL TMOVETO(25,70)
          CALL OUTEXT(' ')
          CALL TMOVETO(25,70)
C
C...Read keyboard input; convert to upper case if required
C
  210 CALL KBREAD(0,CHR)
      CALL CHRCNV(CHR)
      IF(CHR.EQ.'C')THEN
C
C...Option is 'CANCEL'; highlight option (BLACK on WHITE); Start again
C
        CALL TXTCLR(0,7)
        DO 212 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+4))
          CALL OUTEXT('CANCEL')
  212     CONTINUE
        CALL TXTCLR(ICLR3,ICLR2)
        DO 213 M=1,NTEST
          CALL TMOVETO((NLTL+M+1),(NCTL+8))
          CALL OUTEXT(BLANK//BLANK//BLANK)
  213     CONTINUE
        CALL TMOVETO(NLTAB,NCTAB)
        CALL OUTEXT(BLANK)
        GO TO 80
      ELSEIF(CHR.EQ.'G')THEN
```

```
C
C...Option is 'GO'; highlight option (BLACK on WHITE); Continue
C
        CALL TXTCLR(0,7)
        DO 214 M=1,NDEL
          CALL TMOVETO(NLTOP,NCTOP)
          CALL OUTEXT('GO')
  214     CONTINUE
      ELSE
C
C...Inappropriate character; try again
C
        CALL SPCHAR(1)
        GO TO 210
      ENDIF
C
C...Exit
C
      CALL TXTCLR(7,0)
      RETURN
      END
```

Appendix D.5: Subroutine FILTER

```
      SUBROUTINE FILTER(NTASK)
c***************************************************************
c**                                                           **
c**                    Subroutine FILTER                      **
c**                                                           **
c**   Filters NTASK tasks for screen display and data input;  **
c**                                                           **
c**    Performs a maximum of 6 (NTEST) levels of filtration   **
c**    according to specified characteristics (SELECT) of:    **
c**                                                           **
c**                   Task Name (CODE - 1)                    **
c**                   Task WBS  (CODE - 2)                    **
c**                   Task Type (CODE - 3)                    **
c**                                                           **
c**       Filtration parameters are:  LOGIC: 1 for .AND.      **
c**                                          2 for .OR.       **
c**                                                           **
c**                                  OPR:   1 for .EQ.       **
c**                                          2 for .NE.       **
c**                                                           **
c***************************************************************
c
      COMMON/TASKS/TNAME(400),TWBS(400),TTYPE(400),TDURN(400,3),
     &TWK(400),SFCODE(400),FTASK(400),PCODE(400),DFACT(2)

      COMMON/FILDAT/CODE(6),POS(6),OPR(6),SELECT(6),NCHR(6),
     &LOGIC,IFLT,NTEST
C
      CHARACTER*1 TTYPE
      CHARACTER*9 TNAME,SELECT
      CHARACTER*12 TWBS,CVAR,FLNAME
C
      REAL TDURN
      INTEGER FTASK,PCODE,CODE,POS,OPR,NCHR,SFCODE,
     &TWK,IFLT,NTEST
C
C...Set counter for filtered task array FTASK
C
      IFLT=0
C
C...Carry out filtration
C
      DO 90 I=1,NTASK
        DO 70 J=1,NTEST
C
C...Define LH operator in equality/non-equality tests
C
          N1=POS(J)
          N2=N1+(NCHR(J)-1)
          GO TO (10,20,30) CODE(J)
  10      CVAR=TNAME(I)
          GO TO (50,60) OPR(J)
  20       CVAR=TWBS(I)
           GO TO (50,60) OPR(J)
  30       CVAR=TTYPE(I)
           GO TO (50,60) OPR(J)
C
C...Perform appropriate test according to value of OPR
C...Post-test path determined by value of LOGIC
C
C...EQUALITY test
C
  50       IF(CVAR(N1:N2).EQ.SELECT(J)(1:NCHR(J)))GO TO (70,80) LOGIC
           GO TO (90,70) LOGIC
C
C...NON-EQUALITY test
C
  60       IF(CVAR(N1:N2).NE.SELECT(J)(1:NCHR(J)))GO TO (70,80) LOGIC
           GO TO (90,70) LOGIC
  70       CONTINUE
         GO TO (80,90) LOGIC
C
C...Task satisfies filtration criteria; Update filtered task array
C
  80     IFLT=IFLT+1
         FTASK(IFLT)=I
  90     CONTINUE
C
C...Exit
C
       RETURN
       END
```

Appendix D.6: Subroutine TDISPLAY

```
      SUBROUTINE TDISPLAY(ICODE,JCODE)
c*****************************************************
c**                                                 **
c**                Subroutine TDISPLAY              **
c**                                                 **
c** Writes filtered task data to screen in sets of 15 **
c**                                                 **
c** ICODE = 0: Satisfactory display of tasks;       **
c** ICODE = 1: Unsatisfactory display of tasks;     **
c**                                                 **
c** JCODE = 0: Display of INVARIABLE duration tasks; **
c** JCODE = 1: Display of VARIABLE duration tasks;  **
c**                                                 **
c*****************************************************
c
      COMMON/TASKS/TNAME(400),TWBS(400),TTYPE(400),TDURN(400,3),
     &TWK(400),SFCODE(400),FTASK(400),PCODE(400),DFACT(2)

      COMMON/FILDAT/CODE(6),POS(6),OPR(6),SELECT(6),NCHR(6),
     &LOGIC,IFLT,NTEST
C
      CHARACTER*1 TTYPE,CHR1
      CHARACTER*8 CFACT
      CHARACTER*9 TNAME,SELECT,FMT
      CHARACTER*12 TWBS
      CHARACTER*37 TCHAR
      CHARACTER*43 TABREC(21),DBLANK,HEAD
C
      REAL TDURN
      INTEGER FTASK,PCODE,SFCODE,TWK,IFLT,CODE,POS,OPR,NCHR,
     &LOGIC,NTEST
C
C...Specify display colours
C
C...Heading (CYAN)
C
      ICLR1=3
C
C...Table borders and headings (BRIGHT WHITE)
C
      ICLR2=15
C
C...Table background (BLUE)
C
      ICLR3=1
C
C...Table text (WHITE)
C
      ICLR4=7
```

```
C
C...Specify 'reverse video' delay
C
      NDEL=30
C
C...Initialise table strings
C
      IF(JCODE.EQ.0)THEN
        HEAD='      Tasks with INVARIABLE durations      '
      ELSE
        HEAD='       Tasks with VARIABLE durations       '
      ENDIF
      DO 20 I=1,21
        TABREC(I)(1:1)=CHAR(186)
        TABREC(I)(43:43)=CHAR(186)
        DO 10 J=2,42
          TABREC(I)(J:J)=' '
  10      CONTINUE
  20    CONTINUE
      TABREC(1)(1:1)=CHAR(201)
      TABREC(1)(43:43)=CHAR(187)
      TABREC(21)(1:1)=CHAR(200)
      TABREC(21)(43:43)=CHAR(188)
      DO 30 I=2,42
        TABREC(1)(I:I)=CHAR(205)
        TABREC(21)(I:I)=CHAR(205)
  30    CONTINUE
      TABREC(1)(15:28)='Filtered Tasks'
      TABREC(2)(3:41)='NAME       TYPE  WBS           DURATION'
      IF(JCODE.EQ.0)THEN
        TABREC(18)(1:1)=CHAR(200)
        TABREC(18)(43:43)=CHAR(188)
        DO 40 I=2,42
          TABREC(18)(I:I)=CHAR(205)
  40      CONTINUE
      ELSE
        TABREC(18)(1:1)=CHAR(199)
        TABREC(18)(43:43)=CHAR(182)
        DO 50 I=2,42
          TABREC(18)(I:I)=CHAR(196)
  50      CONTINUE
      ENDIF
      TABREC(19)(3:29)='OPTIMISTIC duration factor:'
      TABREC(20)(3:30)='PESSIMISTIC duration factor:'
      DBLANK= '                                           '
C
C...Clear screen
C
      CALL SETWIN(1,1,25,80)
      CALL SCLEAR
C
C...Define table position
C
      NLTL=4
      NCTL=19
```

```
C
C...Define position of display strings
C
      NLTOP=NLTL-1
      NCTOP=NCTL
C
C...Draw header
C
      CALL TMOVETO(1,NCTL)
      CALL TXTCLR(0,ICLR1)
      CALL OUTEXT(HEAD)
C
C...Draw task display table
C
      CALL TXTCLR(ICLR2,ICLR3)
      IREC=18
      IF(JCODE.GT.0)IREC=21
      DO 60 I=1,IREC
        NL=NLTL+I-1
        CALL TMOVETO(NL,NCTL)
        CALL OUTEXT(TABREC(I))
  60    CONTINUE
C
C...Set text window for task scrolling
C
      NLWTL=NLTL+2
      NCWTL=NCTL+2
      NLWBR=NLWTL+14
      NCWBR=NCWTL+37
      CALL SETWIN(NLWTL,NCWTL,NLWBR,NCWBR)
C
C...Scroll filtered tasks in sets of 15
C
      CALL TXTCLR(ICLR4,ICLR3)
      DO 100 I=1,IFLT
        K=FTASK(I)
C
C...Convert task records to CHARACTER string
C
        DO 70 J=1,37
          TCHAR(J:J)=' '
  70      CONTINUE
        WRITE(TCHAR,80)TDURN(K,2)
  80    FORMAT(31X,F6.2)
        TCHAR(1:9)=TNAME(K)
        TCHAR(13:13)=TTYPE(K)
        TCHAR(18:29)=TWBS(K)
        IF(I.EQ.IFLT)THEN
          CALL OUTEXT(TCHAR)
          GO TO 100
        ELSEIF(MOD(I,15).GT.0)THEN
```

```
C
C...Continue until 15th record
C
            CALL OUTEXT(TCHAR//CHAR(10))
            GO TO 100
          ENDIF
C
C...15th record; pause, hide cursor, and display CONTINUE message;
C...Highlight first character of options (RED on BLACK);
C
          CALL OUTEXT(TCHAR)
          CALL SETWIN(1,1,25,80)
          CALL TMOVETO(NLTOP,NCTOP)
          CALL TXTCLR(15,0)
          CALL OUTEXT('Continue?  YES  NO')
          CALL TXTCLR(4,0)
          CALL TMOVETO(NLTOP,(NCTOP+11))
          CALL OUTEXT('Y')
          CALL TMOVETO(NLTOP,(NCTOP+16))
          CALL OUTEXT('N')
          CALL TXTCLR(0,0)
          CALL TMOVETO(25,70)
          CALL OUTEXT(' ')
          CALL TMOVETO(25,70)
C
C...Read input from keyboard; Convert to upper case
C
   90     CALL KBREAD(0,CHR1)
          CALL CHRCNV(CHR1)
          IF(CHR1.EQ.'Y')THEN
C
C...Option is CONTINUE; reset window and continue scrolling tasks;
C
            CALL TXTCLR(0,7)
            DO 92 M=1,NDEL
              CALL TMOVETO(NLTOP,(NCTOP+11))
              CALL OUTEXT('YES')
   92         CONTINUE
            CALL TXTCLR(7,0)
            CALL TMOVETO(NLTOP,NCTOP)
            CALL OUTEXT(DBLANK)
            CALL SETWIN(NLWTL,NCWTL,NLWBR,NCWBR)
            CALL TXTCLR(ICLR4,ICLR3)
            CALL TMOVETO(15,38)
            CALL OUTEXT(CHAR(10))
          ELSEIF(CHR1.EQ.'N')THEN
C
C...Option is DISCONTINUE; Set ICODE and exit routine;
C
            CALL TXTCLR(0,7)
            DO 94 M=1,NDEL
              CALL TMOVETO(NLTOP,(NCTOP+16))
              CALL OUTEXT('NO')
94            CONTINUE
            ICODE=1
          GO TO 999
        ELSE
C
C...Inappropriate character; try again;
C
          CALL SPCHAR(1)
          GO TO 90
        ENDIF
  100   CONTINUE
C
C...Display concluding message;
C...Highlight first character of options (RED on BLACK); Hide cursor;
C
      CALL SETWIN(1,1,25,80)
      CALL TMOVETO(NLTOP,NCTOP)
      CALL TXTCLR(15,0)
      CALL OUTEXT('Filtration OK?  YES  NO')
      CALL TXTCLR(4,0)
      CALL TMOVETO(NLTOP,(NCTOP+16))
      CALL OUTEXT('Y')
      CALL TMOVETO(NLTOP,(NCTOP+21))
      CALL OUTEXT('N')
      CALL TXTCLR(0,0)
      CALL TMOVETO(25,70)
      CALL OUTEXT(' ')
      CALL TMOVETO(25,70)
C
C...Read input from keyboard; Convert to upper case
C
  110 CALL KBREAD(0,CHR1)
      CALL CHRCNV(CHR1)
      IF(CHR1.EQ.'N')THEN
C
C...Option is NOT OK; Set ICODE and exit routine
C
        CALL TXTCLR(0,7)
        DO 112 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+21))
          CALL OUTEXT('NO')
  112     CONTINUE
        ICODE=1
        GO TO 999
      ELSEIF(CHR1.EQ.'Y')THEN
C
C...Option is OK; Set ICODE;
C
        CALL TXTCLR(0,7)
        DO 114 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+16))
          CALL OUTEXT('YES')
  114     CONTINUE
        ICODE=0
      ELSE
```

```
C
C...Inappropriate character; try again;
C
        CALL SPCHAR(1)
        GO TO 110
      ENDIF
C
C...Check if DURATION displays required; If not, exit routine;
C
      IF(JCODE.LT.1)THEN
        CALL TMOVETO(NLTOP,NCTOP)
        CALL TXTCLR(0,0)
        CALL OUTEXT(DBLANK)
        CALL TXTCLR(0,ICLR1)
        CALL TMOVETO((NLTL+18),(NCTL+4))
        CALL OUTEXT(' Tasks will be coded accordingly ')
        CALL TMOVETO((NLTL+19),(NCTL+4))
        CALL TXTCLR(0,4)
        CALL OUTEXT('     Hit any key to continue      ')
        CALL TXTCLR(0,0)
        CALL TMOVETO(25,70)
        CALL OUTEXT(' ')
        CALL TMOVETO(25,70)
        CALL KBREAD(0,CHR1)
        GO TO 999
      ENDIF
C
C...Display DURATION strings and read data from keyboard;
C...Convert to REAL NUMBER if input as INTEGER;
C
  120 DO 180 I=1,2
        CALL TXTCLR(15,0)
        CALL TMOVETO(NLTOP,NCTOP)
        CALL OUTEXT('Factor ')
        IF(I.EQ.1)THEN
         CALL OUTEXT('(0 < f '//CHAR(243)//' 1): ')
         IPOS=21
        ELSE
          CALL OUTEXT('(1 '//CHAR(243)//' f < 1000): ')
          IPOS=24
        ENDIF
        CALL OUTEXT('[         ]')
        CALL TXTCLR(0,6)
        CALL TMOVETO((NLTL+17+I),(NCTL+31))
        CALL OUTEXT('         ')
  125   NCHAR=8
        CALL CREAD(NLTOP,(NCTOP+IPOS),CFACT,NCHAR)
        CALL TMOVETO((NLTL+21),NCTL)
        CALL OUTEXT(DBLANK)
        FMT='      '
        DO 140 K=1,NCHAR
          IF(CFACT(K:K).EQ.'.')THEN
            L=NCHAR-K
            WRITE(FMT,130)NCHAR,L
  130       FORMAT('(F',I1,'.',I1,')')
```

```
            READ(CFACT,FMT)DFACT(I)
            GO TO 160
          ENDIF
  140     CONTINUE
        WRITE(FMT,150)NCHAR
  150   FORMAT('(I',I1,')')
        READ(CFACT,FMT)IDFACT
        DFACT(I)=REAL(IDFACT)
  160   WRITE(CFACT,170)DFACT(I)
  170   FORMAT(F8.4)
C
C...Check if factor within limits
C
        IF(I.EQ.1.AND.DFACT(I).GT.0.0.AND.DFACT(I).LE.1.0)GO TO 17
5
        IF(I.EQ.2.AND.DFACT(I).GT.1.0.AND.DFACT(I).LE.1000.0)GO TO
 175
C
C...Factor out of range; display error message and re-enter
C
        CALL TXTCLR(15,0)
        CALL TMOVETO((NLTL+21),NCTOP)
        CALL OUTEXT('Factor out of range; Try again!!')
        CALL SPCHAR(1)
        CALL TMOVETO(NLTOP,(NCTOP+IPOS))
        CALL OUTEXT('        ')
        GO TO 125
C
C...Duration within limits; update table
C
  175   CALL TMOVETO((NLTL+17+I),(NCTL+31))
        CALL TXTCLR(ICLR4,ICLR3)
        CALL OUTEXT(CFACT(1:8))
  180   CONTINUE
C
C...Check whether input is satisfactory
C
      CALL TMOVETO(NLTOP,NCTOP)
      CALL TXTCLR(0,0)
      CALL OUTEXT(DBLANK)
      CALL TMOVETO(NLTOP,NCTOP)
      CALL TXTCLR(15,0)
      CALL OUTEXT('Factors OK?  YES  NO        ')
      CALL TXTCLR(4,0)
      CALL TMOVETO(NLTOP,(NCTOP+13))
      CALL OUTEXT('Y')
      CALL TMOVETO(NLTOP,(NCTOP+18))
      CALL OUTEXT('N')
      CALL TXTCLR(0,0)
      CALL TMOVETO(25,70)
      CALL OUTEXT(' ')
      CALL TMOVETO(25,70)
```

```
C
C...Read input from keyboard; Convert to upper case
C
  190 CALL KBREAD(0,CHR1)
      CALL CHRCNV(CHR1)
      IF(CHR1.EQ.'N')THEN
C
C...Option is NOT OK; Input new durations
C
        CALL TXTCLR(0,7)
        DO 200 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+18))
          CALL OUTEXT('NO')
  200     CONTINUE
        CALL TXTCLR(ICLR4,ICLR3)
        CFACT(1:8)='        '
        DO 205 I=1,2
          CALL TMOVETO((NLTL+17+I),(NCTL+31))
          CALL OUTEXT(CFACT(1:8))
  205     CONTINUE
        CALL TXTCLR(7,0)
        CALL TMOVETO(NLTOP,NCTOP)
        CALL OUTEXT(DBLANK)
        GO TO 120
      ELSEIF(CHR1.EQ.'Y')THEN
C
C...Option is OK; Exit routine
C
        CALL TXTCLR(0,7)
        DO 210 M=1,NDEL
          CALL TMOVETO(NLTOP,(NCTOP+13))
          CALL OUTEXT('YES')
  210     CONTINUE
      ELSE
C
C...Inappropriate character; try again;
C
        CALL SPCHAR(1)
        GO TO 190
      ENDIF
C
C...Exit routine
C
 999  CALL TXTCLR(7,0)
      RETURN
      END
```

Appendix D.7: Subroutine CHRCNV

```
      SUBROUTINE CHRCNV(CHRVAR)
c****************************************************************
c**                                                            **
c**                    Subroutine CHRCNV                       **
c**                                                            **
c** Converts any lower case characters in CHRVAR to upper case **
c**                                                            **
c****************************************************************
c
      CHARACTER CHRVAR*(*)
C
      NCHR=LEN(CHRVAR)
      DO 10 I=1,NCHR
        N=ICHAR(CHRVAR(I:I))
        IF(N.GT.96.AND.N.LT.123)THEN
          N=N-32
          CHRVAR(I:I)=CHAR(N)
        ENDIF
   10   CONTINUE
C
C...Exit
C
      RETURN
      END
```

Appendix D.8: Function INTCHR

```
      FUNCTION INTCHR(I)
c*************************************************
c**                                            **
c**              Function INTCHR               **
c**                                            **
c** Converts a single digit NUMERIC INTEGER (I) **
c**          into an ASCII CHARACTER           **
c**                                            **
c*************************************************
c
      CHARACTER*1 INTCHR
c
      IF(I.LT.10)THEN
        N=I+48
        INTCHR=CHAR(N)
      ENDIF
      RETURN
      END
```

Appendix D.9: Subroutine CREAD

```
      SUBROUTINE CREAD(NL,NC,CHRVAR,NCHR)
c*******************************************************
c**                                                   **
c**                Subroutine CREAD                   **
c**                                                   **
c** Reads a character variable CHRVAR, of maximum length **
c**    NCHR characters, from the keyboard at screen   **
c**                position NL,NC.                    **
c**                                                   **
c** Returns the actual number of characters NCHR in the **
c**                   variable.                       **
c**                                                   **
c**             Uses 'C' subroutines.                 **
c**                                                   **
c*******************************************************
c
      CHARACTER CHRVAR*(*)
      CHARACTER*1 CHR
C
      CALL TXTCLR(7,0)
C
C...Move cursor to required position
C
      CALL TMOVETO(NL,NC)
      I=1
  10  IF(I.GT.NCHR)GO TO 30
      IC=NC+I-1
C
C...Read input character (no echo)
C
  20  CALL KBREAD(0,CHR)
      J=ICHAR(CHR)
      IF(J.EQ.13)THEN
C
C...Character is CARRIAGE RETURN;
C...Determine length of character variable; Exit routine;
C
        N=IC-NC
        GO TO 50
      ELSEIF(J.EQ.8)THEN
C
C...Character is BACKSPACE; Delete preceding character;
C...Protect against over-deletion
C
        IC=IC-1
        I=I-1
        IF(I.GT.0)GO TO 25
        I=I+1
        IC=I+1
        CALL SPCHAR(1)
        GO TO 20
  25    CALL TMOVETO(NL,IC)
```

```
          CALL OUTEXT(' ')
          CALL TMOVETO(NL,IC)
          GO TO 20
        ELSEIF(J.EQ.32)THEN
C
C...Character is SPACE; Ignore
C
          CALL SPCHAR(1)
          GO TO 20
        ELSE
C
C...Character is acceptable; Update character variable and continue
C
          N=(IC-NC)+1
          CHRVAR(N:N)=CHR
          CALL OUTEXT(CHR)
        ENDIF
        I=I+1
        GO TO 10
C
C...Hide cursor
C
   30   CALL TXTCLR(0,0)
        CALL TMOVETO(25,70)
        CALL OUTEXT(' ')
        CALL TMOVETO(25,70)
   40   CALL KBREAD(0,CHR)
        J=ICHAR(CHR)
        IF(J.EQ.13)THEN
          GO TO 50
        ELSEIF(J.EQ.8)THEN
          CALL TXTCLR(7,0)
          CALL TMOVETO(NL,IC)
          CALL OUTEXT(' ')
          CALL TMOVETO(NL,IC)
          I=NCHR
          GO TO 10
        ELSE
          CALL SPCHAR(1)
          GO TO 40
        ENDIF
C
C...Exit
C
   50   NCHR=N
        RETURN
        END
```

Appendix E: Sample of the Updated Task Data Written to the File FLNAME.TSK

```
 274      (Number of tasks in the project)

(1)  (2)         (3)          (4) (5) (6)  (7)      (8)      (9)

 T CLEAR1     GN-CLEAR         5   1   1  1.00     2.00     4.00
 T CLEAR2     GN-CLEAR         5   1   1  2.50     5.00    10.00
 H CMPRESS    GN-CMPRESS       5   1   0  0.00   136.00     0.00
 T CPC1CO     PR-CONC          5   1   1  5.00     1.00    20.00
 T CPC1D1     PR-DELAY         7   1   0  0.00     1.00     0.00
     .    .    .    .    .    .    .    .    .    .
     .    .    .    .    .    .    .    .    .    .
```

Defintion of Columns are:

(1): Task Type.

(2): Task Name.

(3): Task Work Breakdown Structure.

(4): Number of Working Days/Week.

(5): Task Code (Start, Finish or Normal Task).

(6): Variable/Invariable duration Task Codes (1= Invariable; 0= Variable).

(7): Optimistic Duration (Days).

(8): Most Likely Duration (Days).

(9): Pessimistic Durations (Days).

Appendix F: Program PERTRA3 and Subroutines

Appendix F.1: Program PERTRA3

```
c*******************************************************************
c**                                                               **
c**                     Program PERTRA3.FOR                       **
c**                                                               **
c**       Reads selected data from the files generated by the     **
c**          programs PERTRA1.FOR; PERTRA2.FOR and perform         **
c**       RISK ANALYSIS CALCULATIONS (Monte-Carlo simulation)      **
c**                                                               **
c*******************************************************************
c
      COMMON/FREQU1/ PCFREQ(100),PRBMPT(100),IPBAND(100),PFREQ(100),
     &RCPWPD(100)
      COMMON/FREQU2/CFREQ(100),BANMPT(100),IBAND(100),FREQ(100),
     &ROPWRR(100)
      COMMON/FREQU3/CCFREQ(100),CABMPT(100),ICBAND(100),CAFREQ(100),
     &ROPWCAP(100)
      COMMON/PDIST/DLIM(400,2),FMODE(400)
      COMMON/STATS/MINPDU,MAXPDU,PRMEAN,PSKEW,MEDIAN,PROSIG
      COMMON/STAT2/RRMIN,RRMAX,RRMEAN,RRSKEW,RRMEDN,RRSIG
      COMMON/STAT3/CAPMIN,CAPMAX,CAMEAN,CSKEW,CMEDIAN,CAPSIG
      COMMON/PDCOST/DCOSTS(500,9),COSPAY(500,9),COSDLY(500,9),
     &IDDATE(500),MACDAY(50),TOTCOS(500),ACTOTC(500),ACCTPR(500),
     &CAPTIM(1000),RATERN(1000)
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400,3),SFCODE(400),
     &TWK(400),TCOST(400,9),TWBS(400),PCCODE(400),IDURN(400)
      COMMON/LINKS/LFROM(500,2),LTO(500,2),LTYPE(500,2),LDURN(500,2)
      COMMON/FINSTA/ISTART(400),IFINISH(400),NDAYOW(1000)
      COMMON/HEADS/PHEAD(3)
      COMMON/COSTS/CCODE(9),CPAY(9),CARR(9),CARRQ(9)
      COMMON/CDATES/IDAYSD(400,4),MODATE(400,4),IYDATE(400,4)
      COMMON/DAYNOS/IESDNO(400),IESDOW(400),IDURN2(400),LFDNO(400),
     &LFDOW(400),IDURN3(400),ILDURN(500,2),ICRIT(400),ACRIT(400),
     &IPRDRN(1000),IEFDNO(400),IEFDOW(400),LSDNO(400),LSDOW(400)
      COMMON/STAFIN/NSTART,NFINISH,IPSDNO,IPFDNO,IPSDOW,NLINK,NTASK
c
      CHARACTER*1 TTYPE,RGCODE,CPAY,CARR
      CHARACTER*3 PSMOTH,CCODE,MODATE,PFMOTH
      CHARACTER*8 PNAME
      CHARACTER*9 TNAME
      CHARACTER*12 FLNAME,TWBS
      CHARACTER*30 PHEAD
c
      REAL TDURN,LDURN,TCOST,MEDIAN,ACRIT
      INTEGER SFCODE,TWK,PCCODE,PVER,CARRQ,LFROM,LTO,LTYPE,ICRIT,N
```

```
c
c...Input data from terminal
c
      WRITE(6,10)
 10   FORMAT(/////,30X,'PROGRAM PERTRA3.FOR',//,9X,'Reads selected data
     &from the files generated by Programs',/,9x,'PERTRA1.FOR ; PERTRA2.
     &FOR and perform Risk Analysis calculations')
 20   WRITE(6,30)
 30   FORMAT(///,' Select the type of random generation ',
     &/,' (R) for a Repetable OR (N) for a non-repetable ',\)
      READ(5,40)RGCODE
 40   FORMAT(A1)
c
c...Check random generation code
c
      IF(RGCODE.EQ.'R'.OR.RGCODE.EQ.'r'.OR.RGCODE.EQ.'N'.OR.RGCODE.EQ.'n
     &')GOTO55
      WRITE(6,50)RGCODE
 50   FORMAT(//,' CODE WERE WRITING AS ( ',A1,' )',/,' UNIDINTIFIED CODE
     &',/,' TRY AGAIN')
      GOTO20
c
c...Input number of runs
c
 55   WRITE(6,60)
 60   FORMAT(///,' Number of runs required  ?? ',\)
      READ(5,70)NRUNS
 70   FORMAT(I4)
c
c...Read FLNAME from file: FILES.PRT on unit 4
c
      OPEN(UNIT=4,FILE='PERTRA.FLS')
      WRITE(6,80)
  80  FORMAT(//,' Reading Project Name ')
      READ(4,90)FLNAME
  90  FORMAT(A8)
      CLOSE(4)
c
c...Define GENERAL data file
c
      FLNAME(9:12)='.GEN'
c
c...Open GENERAL data file on unit 4 and read the data.
c
      OPEN(UNIT=4,FILE=FLNAME)
      WRITE(6,100)FLNAME
 100  FORMAT(///,' Reading GENERAL data from file ',A12)
      READ(4,110)PNAME,PVER
 110  FORMAT(A8,1X,I2)
      READ(4,120)(PHEAD(I),I=1,3)
 120  FORMAT(A30)
      READ(4,130)IPSDAY,PSMOTH,IPSYR
 130  FORMAT(I2,1X,A3,1X,I2)
      CLOSE(4)
```

```
c
c...Adjust Project Start Year for day numbers calculations
c
      IPSYR=IPSYR+1900
c
c...Define TASKS data file
c
      FLNAME(9:12)='.TSK'
c
c...Open TASKS data file on unit 4 and read the data
c
      OPEN(UNIT=4,FILE=FLNAME)
      WRITE(6,140)FLNAME
 140  FORMAT(///,' Reading TASKA data from file ',A12)
      READ(4,*)NTASK
      READ(4,150)(TTYPE(I),TNAME(I),TWBS(I),TWK(I),SFCODE(I),PCCODE(I),
     &(TDURN(I,J),J=1,3),I=1,NTASK)
 150  FORMAT(1X,A1,1X,A9,1X,A12,3I2,3F7.2)
      CLOSE(4)
c
c...Define LINKS data file
c
      FLNAME(9:12)='.LNK'
c
c...Open LINKS file on unit 4 and read the data.
c
      OPEN(UNIT=4,FILE=FLNAME)
      WRITE(6,160)FLNAME
 160  FORMAT(///,' Reading LINKS data from file ',A12)
      READ(4,*)NLINK
      READ(4,170)((LFROM(I,J),LTO(I,J),LTYPE(I,J),LDURN(I,J),J=1,2),I=1,
     &NLINK)
 170  FORMAT(2(3I4,F7.2))
      CLOSE(4)
c
c...Define COSTS data file
c
      FLNAME(9:12)='.CST'
c
c...Open Ccosts file on unit 4 and read data.
c
      OPEN(UNIT=4,FILE=FLNAME)
      WRITE(6,180)FLNAME
 180  FORMAT(///,' Reading COSTS data from file ',A12)
      READ(4,190)(CCODE(I),CPAY(I),CARR(I),CARRQ(I),I=1,9)
      READ(4,200)((TCOST(I,J),J=1,9),I=1,NTASK)
 190  FORMAT(A3,1X,A1,1X,A1,I4)
 200  FORMAT(9F10.2)
      CLOSE(4)
c
c...Initilise Criticality Indix
c
      DO210,I=1,NTASK
        ICRIT(I)=0
 210    CONTINUE
```

```
c
c...Set run without probability analysis to run
c...number one (Using Most Likely Task Duration)
c
      N=1
c
c...Change tasks duration to Integers
c
      DO220,I=1,NTASK
        IDURN(I)=INT(TDURN(I,2))
        IF(TDURN(I,2).GT.AINT(TDURN(I,2)))IDURN(I)=IDURN(I)+1
 220    CONTINUE

c
c...Calculate project start day number
c
      CALL CALEND(1,IPSDAY,PSMOTH,IPSYR,IPSDNO)
c
c...Calculate Project Start Day of the Week Number
c
      IPSDOW=IDOTW(IPSDNO)
c
c...Check that the starting date of the project
c...is not a weekend and adjust if so .........
c
      IF(IPSDOW.EQ.7)THEN
c
c...Adjust if it is sunday
c
        IPSDNO=IPSDNO+1
        IPSDOW=1
      ELSEIF(IPSDOW.EQ.6)THEN
c
c...Adjust if it is saturday
c
        IPSDNO=IPSDNO+2
        IPSDOW=1
      ENDIF
c
c...Perform network Time analysis
c
      WRITE(6,230)
 230  FORMAT(///' Performing network TIME ANALYSIS ')
      CALL ANALYS(N)
c
c...Change Project Finish Day Number to Calendar Date
c
      CALL CALEND(2,IPFDAY,PFMOTH,IPFYR,IPFDNO)
c
c...Distribute the costs on the project days,
c...calculate IRR and CAPTIM.................
c
      WRITE(6,250)
 250  FORMAT(///,' Performing network COST ANALYSIS')
      CALL PCOSTS(NSTART,NTASK,ICDAYS,N)
```

```
c
c...Quit the program if so desired; Otherwise continue
c
      OPEN(UNIT=4,FILE='PERTRA3.RES')
      WRITE(4,251)IPSDAY,PSMOTH,IPSYR
      WRITE(4,252)IPFDAY,PFMOTH,IPFYR
      WRITE(4,255)IPRDRN(N)
      WRITE(4,253)CAPTIM(N)
      WRITE(4,254)RATERN(N)
 251  FORMAT(//,1X,'PROJECT START  DATE: ',I2,'/',A3,'/',I4)
 252  FORMAT(//,1X,'PROJECT FINISH DATE: ',I2,'/',A3,'/',I4)
 253  FORMAT(//,1X,'Negative CAPTIM = ',F10.2,' .Day')
 254  FORMAT(//,1X,'Internal Rate of Return (IRR) -',f10.2,' %',//)
 255  FORMAT(//,1x,'Project Duration =',I6,2X,'Days')
      CLOSE(4)
      IF(NRUNS.EQ.1)GOTO900
      WRITE(6,260)
 260  FORMAT(///,' Calculating Triangular distribution parameters')
      DO270,I=1,NTASK
c
c...Ignore Milestones and Hammock tasks
c
        IF(PCCODE(I).EQ.0.OR.TTYPE(I).EQ.'H')GOTO270
c
c...Calculate Triangular distribution Parameters.
c
        CALL TRIDIS(I)
 270    CONTINUE
c
c...Perform the required number of runs, taking into account run no. one
c
      DO400,N=2,NRUNS
        WRITE(6,280)N
 280    FORMAT(///,' Performing run number:',I4)
c
c...Calculate the seed for a repetable random generation
c
        IF(RGCODE.EQ.'R'.OR.RGCODE.EQ.'r')THEN
          SEED=129600.0
        ELSE
c
c...Calculate the seed for non-repetable random generation
c
          CALL GETTIM(IHR,IMIN,ISEC,IHUN)
          SEED=REAL(IHUN*2592+1296)
        ENDIF
c
c...Ignoring Milestones and Hammock tasks
c
        DO290,I=1,NTASK
          IF(PCCODE(I).EQ.0.OR.TTYPE(I).EQ.'H')GOTO290
```

```
c
c...Randomly select  (INTEGER) Tasks durations
c
          CALL DSELECT(I,SEED)
 290      CONTINUE
c
c...Perform network time analysis for the selecte durations
c
        CALL ANALYS(N)
c
c...Distribute the costs on the project days,
c...calculate IRR and CATIM.................
c
        CALL PCOSTS(NSTART,NTASK,ICDAYS,N)
 400    CONTINUE
c
c...Change criticality index int percentages
c
      DO410,I=1,NTASK
        ACRIT(I)=FLOAT(ICRIT(I))/FLOAT(NRUNS)*100.0
 410    CONTINUE
c
c...Write Screen Message
c
      WRITE(6,500)
 500  FORMAT(///,1x,'Performing Statistical Analysis on the Results')
c
c...Sort project durations into rank order
c
      CALL PSORT(NRUNS)
c
c...Calculate minimum,maximum,mean,median project durations and skewnees
c
      CALL PRSTAT(NRUNS)
c
c...Divide the project durations into bands and calculate frequency
c
      CALL FREQUA(NRUNS,NBANDS)
c
c...Sort IRR and CAPTIM into rank order
c
      CALL RRSORT(NRUNS)
c
c...Calculate minimum,maximum,mean,median IRR and skewnees
c
      CALL RRSTAT(NRUNS)
c
c...Calculate number of bands and frequency for the sorted IRR
c
      CALL FRQIRR(NRUNS,INBAND)
c
c...Calculate minimum,maximum,mean,median  CAPTIM and skewnees
c
      CALL CASTAT(NRUNS)
c
```

```
c...Calculate number of bands and frequency for the sorted CAPTIM
c
      CALL FRQCAP(NRUNS,ICNBAND)
c
c...Write the results into the file PERTRA3.RES
c
      WRITE(6,600)
 600  FORMAT(///,1X,'Writing Data to the File PERTRA3.RES',///)
      OPEN(UNIT=3,FILE='PERTRA3.RES')
      WRITE(3,700)NRUNS
 700  FORMAT(//,' Number of runs =',I4)
      WRITE(3,710)
 710  FORMAT(//,2X,'TASK',3X,'TASK',6X,'TASK',3X,'OPTIMISTIC',2X,'MOST L
     &IKELY',2X,'PESSIMISTIC',2X,'CRITICALITY',/,1X,'NUMBER',2X,'NAME',6
     &X,'TYPE',4X,'DURATION',4X,'DURATION',5X,'DURATION',5X,'INDEX (%)')
      WRITE(3,720)(I,TNAME(I),TTYPE(I),TDURN(I,1),TDURN(I,2),TDURN(I,3),
     &ACRIT(I),I=1,NTASK)
 720  FORMAT(2X,I3,4X,A8,3X,A1,7X,F6.2,6X,F6.2,7X,F6.2,7X,F6.2)
      WRITE(3,725)
 725  FORMAT(///,1X,'1. PROJECT DURATIONS')
      WRITE(3,730)MINPDU,MAXPDU,PRMEAN,MEDIAN,PROSIG,PSKEW
 730  FORMAT(//,' Minimum project duration =',I4,4X,'DAYS',/,' Maximum p
     &roject duration =',I4,4X,'DAYS',/,' Mean Project duration    =',F7
     &.2,1X,'DAYS',/,' Median project duration  =',F7.2,1X,'DAYS',//,' S
     &tandard Deviation =',F7.2,1X,'DAYS',/,' Skewness =',F7.4,//)
      WRITE(3,740)
 740  FORMAT(/,5X,'MIDPOINT',5X,'FREQUENCY',3X,'CUM. FREQUENCY',8X,'f(x)
     &',/)
      WRITE(3,750)(PRBMPT(I),PFREQ(I),PCFREQ(I),RCPWPD(I),I=1,(NBANDS-1)
     &)
 750  FORMAT(5X,F7.2,5X,F7.2,8X,F7.2,10X,F7.2)
      WRITE(3,760)
 760  FORMAT(///,1X,'2. INTERNAL RATE OF RETURN (IRR)')
      WRITE(3,770)RRMIN,RRMAX,RRMEAN,RRMEDN,RRSIG,RRSKEW
 770  FORMAT(//,' Minimum IRR =',F12.3,1X,'%',/,' Maximum IRR =',F12.3,1
     &X,'%',/,' Mean    IRR =',F12.3,1X,'%',/,' Median  IRR =',F12.3,1X,
     &'%',//,' IRR Standard Deviation =',F12.5,1X,'%',/,' Skewness =',F9
     &.5,///)
      WRITE(3,740)
      WRITE(3,780)(BANMPT(I),FREQ(I),CFREQ(I),ROPWRR(I),I=1,(INBAND-1))
 780  FORMAT(2X,F10.2,5X,F7.2,8X,F7.2,11X,F7.2)
      WRITE(3,781)
 781  FORMAT(//,1X,'3. Negative CAPTIM')
      WRITE(3,782)CAPMIN,CAPMAX,CAMEAN,CMEDIAN,CAPSIG,CSKEW
 782  FORMAT(//,' Minimum CAPTIM =',F15.2,1X,'.DAY  ',/,' Maximum CAPTI
     &M =',F15.2,1X,'.DAY  ',/,' Mean    CAPTIM =',F15.2,1X,'.DAY  ',
     &/,' Median  CAPTIM =',F15.2,1X,'.DAY  ',//,' CAPTIM Standard Devi
     &ation =',F12.2,1X,'.DAY  ',/,' Skewness =',F9.5,//)
      WRITE(3,740)
      WRITE(3,785)(CABMPT(I),CAFREQ(I),CCFREQ(I),ROPWCAP(I),I=1,(ICNBAND
     &-1))
 785  FORMAT(2X,F15.2,2X,F7.2,8X,F7.2,11X,F7.2)
      CLOSE(3)
 900  STOP
      END
```

Appendix F.2 Subroutine CALEND

```
      SUBROUTINE CALEND(ICODE,IDAY,MONTH,IYEAR,IDAYNO)
c**************************************************************
c**                                                          **
c**                 Subroutine CALEND                        **
c**    ICODE=1;  Changes Project Start Date into Day Number  **
c**    ICODE=2;  Changes Project Finish Day  Number  into    **
c**              Calendar Date                               **
c**                                                          **
c**************************************************************
c
      CHARACTER*3 MONTH
c
      PARAMETER(A=365.25,B=30.6001,C=122.1)
c
c...Check ICODE and perform Calculation Accordingly
c
      IF(ICODE.EQ.1)THEN
c
c...Change the Month from Character Format to Integer
c
        IF(MONTH.EQ.'JAN')THEN
          IMONTH=1
        ELSEIF(MONTH.EQ.'FEB')THEN
          IMONTH=2
        ELSEIF(MONTH.EQ.'MAR')THEN
          IMONTH=3
        ELSEIF(MONTH.EQ.'APR')THEN
          IMONTH=4
        ELSEIF(MONTH.EQ.'MAY')THEN
          IMONTH=5
        ELSEIF(MONTH.EQ.'JUN')THEN
          IMONTH=6
        ELSEIF(MONTH.EQ.'JUL')THEN
          IMONTH=7
        ELSEIF(MONTH.EQ.'AUG')THEN
          IMONTH=8
        ELSEIF(MONTH.EQ.'SEP')THEN
          IMONTH=9
        ELSEIF(MONTH.EQ.'OCT')THEN
          IMONTH=10
        ELSEIF(MONTH.EQ.'NOV')THEN
          IMONTH=11
        ELSEIF(MONTH.EQ.'DEC')THEN
          IMONTH=12
        ENDIF
```

```
c
c...Calculate the Day Number
c
        IF(IMONTH.GE.1.AND.IMONTH.LE.2)THEN
          IYEAR1=IYEAR-1
          IMONTH1=IMONTH+13
        ELSE
          IYEAR1=IYEAR
          IMONTH1=IMONTH+1
        ENDIF
        IDAYNO=INT(A*IYEAR1)+INT(B*IMONTH1)+IDAY
c
c...ICODE is G.T. One; Changing from Day Number to Date is Required
c
      ELSE
c
c...Calculate the approximate year and month
c
        IYEAR2=INT((IDAYNO-C)/A)
        IMTH2=INT((IDAYNO-INT(A*IYEAR2))/B)
c
c...Calculate the exact day of the date
c
        IDAY=IDAYNO-INT(A*IYEAR2)-INT(B*IMTH2)
c
c...Calculate the exact month of the date
c
        IF(IMTH2.LT.14)THEN
          IMONTH=IMTH2-1
        ELSE
          IMONTH=IMTH2-13
        ENDIF
c
c...Calculate the exact year
c
        IF(IMONTH.LE.2)THEN
          IYEAR=IYEAR2+1
        ELSE
          IYEAR=IYEAR2
        ENDIF
```

```
c
c...Change the month from number to name
c
        IF(IMONTH.EQ.1)THEN
          MONTH='JAN'
        ELSEIF(IMONTH.EQ.2)THEN
          MONTH='FEB'
        ELSEIF(IMONTH.EQ.3)THEN
          MONTH='MAR'
        ELSEIF(IMONTH.EQ.4)THEN
          MONTH='APR'
        ELSEIF(IMONTH.EQ.5)THEN
          MONTH='MAY'
        ELSEIF(IMONTH.EQ.6)THEN
          MONTH='JUN'
        ELSEIF(IMONTH.EQ.7)THEN
          MONTH='JUL'
        ELSEIF(IMONTH.EQ.8)THEN
          MONTH='AUG'
        ELSEIF(IMONTH.EQ.9)THEN
          MONTH='SEP'
        ELSEIF(IMONTH.EQ.10)THEN
          MONTH-'OCT'
        ELSEIF(IMONTH.EQ.11)THEN
          MONTH='NOV'
        ELSEIF(IMONTH.EQ.12)THEN
          MONTH='DEC'
        ENDIF
      ENDIF
      RETURN
      END
```

Appendix F.3: Function IDOTW

```
      INTEGER FUNCTION IDOTW(IDAYNO)
C*********************************************************
C**                                                     **
C**     FUNCTION RETURNS THE DAY OF THE WEEK NUMBER     **
C**               FROM THE DAY NUMBER                   **
C**                                                     **
C**      IDAYNO - DATE DAY NUMBER                       **
C**      IDOTW  - DATE DAY OF THE WEEK NUMBER           **
C**                                                     **
C*********************************************************
C
      M=IDAYNO+5
      IDOTW=MOD(M,7)
      IF(IDOTW.EQ.0)IDOTW=7
      RETURN
      END
```

Appendix F.4: Subroutine ANALYS

```
      SUBROUTINE ANALYS(N)
c**********************************************************************
c**                                                                  **
c**                       SUBROUTINE ANALYS                          **
c**                                                                  **
c**      Performs forward pass to determine tasks Earliest Start     **
c**      day numbers (IESDNO) , Earliest  Start  day of the week     **
c**      numbers (IESDOW), Earliest finish day numbers  (IEFDNO)     **
c**      and Earliest  Finish  day of the week numbers  (IEFDOW)     **
c**                                                                  **
c**      Performs backward pass to calculate Latest  Finish day      **
c**      numbers (LFDNO), Latest finish day  of the week number      **
c**      (LFDOW), Latest Start day number  (LSDNO)  and  latest      **
c**            start day of the week numbers (LSDOW)                 **
c**                                                                  **
c**       Calculate Hammock tasks durations and Identify             **
c**          Critical tasks in the network (ICRIT)                   **
c**                                                                  **
c**         NSTART  - Number of the first task in the network        **
c**         NFINISH - Number of the last  task in the network        **
c**         IPSDNO  - Project start day number                       **
c**         IPSDOW  - Project start day of the week number           **
c**         NLINK   - Number of links                                **
c**         NTASK   - Number of tasks                                **
c**         N       - Run number                                     **
c**                                                                  **
c**********************************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400,3),SFCODE(400),
     &TWK(400),TCOST(400,9),TWBS(400),PCCODE(400),IDURN(400)
      COMMON/LINKS/LFROM(500,2),LTO(500,2),LTYPE(500,2),LDURN(500,2)
      COMMON/DAYNOS/IESDNO(400),IESDOW(400),IDURN2(400),LFDNO(400),
     &LFDOW(400),IDURN3(400),ILDURN(500,2),ICRIT(400),ACRIT(400),
     &IPRDRN(1000),IEFDNO(400),IEFDOW(400),LSDNO(400),LSDOW(400)
      COMMON/STAFIN/NSTART,NFINISH,IPSDNO,IPFDNO,IPSDOW,NLINK,NTASK
      CHARACTER*1 TTYPE
      CHARACTER*9 TNAME
      CHARACTER*12 TWBS
c
      REAL LDURN
      INTEGER SFCODE,TWK,PCCODE
c
c...Initialise project finish day number (IPFDNO)
c
      IPFDNO=0
c
c...Search for START task number (NSTART)
c
      DO10,I=1,NTASK
        IF(SFCODE(I).EQ.0)NSTART=I
  10    CONTINUE
```

```
c
c...Assign, project start, day number  (IPSDNO) and  day of
c...the week number (IPSDOW) to tasks (IESDNO),(IEFDNO) and
c...(IESDOW), (IEFDOW) respectively
c
      DO20,I=1,NTASK
        IESDNO(I)=IPSDNO
        IEFDNO(I)=IPSDNO
        IESDOW(I)=IPSDOW
        IEFDOW(I)=IPSDOW
  20    CONTINUE
c
c...Set the number of non-working days
c...(IDURN2) for the start task to zero
c
      IDURN2(NSTART)=0
c
c...Initialise repeating loop counter
c
  30  K=0
c
c...Carry out forward pass to determine tasks
c...(IESDNO), (IESDOW) and (IEFDNO), (IEFDOW)
c
      DO50,I=1,NLINK
        IFROM=LFROM(I,1)
        ITO=LTO(I,1)
c
c...Ignore Hammock tasks
c
        IF(TTYPE(IFROM).EQ.'H'.OR.TTYPE(ITO).EQ.'H')GOTO50
c
c...Identify the forward pass for later use
c
        M=1
c
c...Calculate the absolute Integer link durations
c
        ILDURN(I,1)=LINKDN(I,M,IFROM)
c
c...For (F-S) link type
c
        IF(LTYPE(I,1).EQ.1)THEN
c
c...Find out the addition days, 0 or 1 (Because of using day numbers)
c
          IADD=SFCODE(IFROM)
c
c...Calculate the, temporary, early start day number
c...(ITIME) and early start day of the week number (ITEMP)
c
          ITIME=IEFDNO(IFROM)+IADD
          ITEMP=IDOTW(ITIME)
```

```
c
c...Using the Function (ISADED), Calculate
c...the additional non working days
c
          ISADD=ISADED(ITO,ITEMP)
c
c...Find out the early start day of the week number of the link duration
c
          IESLD=ITIME+ISADD
          IDOWL=IDOTW(IESLD)
c
c...Calculate the additional non working days to the link duration
c
          ILADD=(INT((ILDURN(I,1)+IDOWL-2)/TWK(IFROM)))*(7-TWK(IFROM))
c
c...Recalculate (ITIME), (ITEMP)
c
          ITIME=IESLD+ILDURN(I,1)+ILADD
          ITEMP=IDOTW(ITIME)
c
c...Calculate the additional number of non-working days
c
          ISADD=ISADED(ITO,ITEMP)
c
c...Adjust  (ITIME)
c
          ITIME=ITIME+ISADD
c
c...For (S-S) link type
c
        ELSEIF(LTYPE(I,1).EQ.2)THEN
c
c...Calculate the additional number of
c...non-working days to the link duration
c
          ILADD=(INT((ILDURN(I,1)+IESDOW(IFROM)-2)/TWK(IFROM)))*(7-TWK(
     &IFROM))
c
c...Calculate (ITIME), (ITEMP)
c
          ITIME=IESDNO(IFROM)+ILDURN(I,1)+ILADD
          ITEMP=IDOTW(ITIME)
c
c...Calculate the additional number of non-working days
c
          ISADD=ISADED(ITO,ITEMP)
c
c...Adjust (ITIME)
c
          ITIME=ITIME+ISADD
c
c...For (F-F) or (S-F) link types
c
        ELSE
```

```
c
c...No calculation for these type of links
c
        ITIME=IESDNO(ITO)
      ENDIF
c
c...Compare (ITIME) with succeeding task (IESDNO) and divert the program
c
      IF(ITIME.LE.IESDNO(ITO))GOTO40
c
c...Update the repeating loop counter
c
      K=K+1
c
c...Assign ITIME to the succeeding task (IESDNO)
c
      IESDNO(ITO)=ITIME
c
c...Update project finish day number (IPFDNO)
c
      IF(ITIME.GT.IPFDNO)IPFDNO=ITIME

c
c...Calculate the early start day of the week number  (IESDOW)
c... and the non working days (IDURN2) for the succeeding task
c
      IESDOW(ITO)=IDOTW(ITIME)
      IDURN2(ITO)=(INT((IDURN(ITO)+IESDOW(ITO)-2)/TWK(ITO)))*(7-TWK(IT
     &O))
c
c...For each type of link, calculate tasks (IEFDNO),(IEFDOW)
c
c...For 'F-S' or 'S-S' link types
c
 40   IF(LTYPE(I,1).EQ.1.OR.LTYPE(I,1).EQ.2)THEN
c
c...Calculate number of non-workingdays to be subtracted
c...(Because of the use of day numbers)
c
        ITNADD=2-SFCODE(ITO)
c
c...Using the earliest start day number calculate
c...the temporary earliest finish day number (ITIME2)
c
        ITIME2=IESDNO(ITO)+IDURN(ITO)+IDURN2(ITO)-ITNADD
c
c...For 'F-F' link type
c
      ELSEIF(LTYPE(I,1).EQ.3)THEN
c
c...Calculate the additional non working days to the link duration
c
        ILADD=(INT((IEFDOW(IFROM)+ILDURN(I,1)-2)/TWK(ITO)))*(7-TWK(ITO
     &))
```

```
c
c...Calculate the temporary earliest finish day number(ITIME2)
c...and  earliest  finish  day  of the  week  number  (ITEMP2)
c
          ITIME2=IEFDNO(IFROM)+ILDURN(I,1)+ILADD
          ITEMP2=IDOTW(ITIME2)
c
c...Check if any more days to be added and calculate it
c
          ISADD=ISADED(ITO,ITEMP2)
c
c...Adjust (ITIME2)
c
          ITIME2=ITIME2+ISADD
c
c...For 'S-F' link type
c
        ELSE
c
c...Calculate the additional non-working days to the link duration
c
          ILADD=(INT((IESDOW(IFROM)+ILDURN(I,1)-2)/TWK(ITO)))*(7-TWK(ITO
     &))
c
c...Calculate (ITIME2)
c
          ITIME2=IESDNO(IFROM)+ILDURN(I,1)+ILADD-1
        ENDIF
c
c...Compare (ITIME2) with succeeding task (IEFDNO); Divert
c... program to store the greatest and calculate (IEFDOW)
c
        IF(ITIME2.LE.IEFDNO(ITO))GOTO50
        IEFDNO(ITO)=ITIME2
        IEFDOW(ITO)=IDOTW(ITIME2)
   50   CONTINUE
c
c...If counter updated; Repeat the loop
c
       IF(K.GT.0)GOTO30
c
c...Identify FINISH task number (NFINISH)
c
       DO60,I=1,NTASK
         IF(SFCODE(I).EQ.2)NFINISH=I
   60   CONTINUE
```

```
c
c...Initialise tasks latest finish day numbers (LFDNO), latest start
c...day numbers (LSDNO), latest finish day of the week numbers (LFDOW)
c...and latest start day of the week numbers (LSDOW)
c
      DO70,I=1,NTASK
        LFDNO(I)=IPFDNO
        LSDNO(I)=IPFDNO
        LFDOW(I)=IDOTW(IPFDNO)
        LSDOW(I)=IDOTW(IPFDNO)
   70   CONTINUE
c
c...Initialise repeating loop counter
c
   80 K=0
c
c...Carry out backward pass to calculate tasks
c...(LFDNO), (LFDOW) and (LSDNO), (LSDOW)
c
      DO100,I=1,NLINK
        IFROM=LFROM(I,2)
        ITO=LTO(I,2)
c
c...Ignore Hammock tasks
c
        IF(TTYPE(IFROM).EQ.'H'.OR.TTYPE(ITO).EQ.'H')GOTO100
c
c...Name the backward pass
c
        M=2
c
c...Using the function (LINKDN) calculate the Integer link durations
c
        ILDURN(I,2)=LINKDN(I,M,IFROM)
c
c...Select the link type and perform calculations; For (F-S) link type
c
        IF(LTYPE(I,2).EQ.1)THEN
c
c...Identify the number of days to deducted
c...(Because of using day numbers)
c
          IDDCT=SFCODE(IFROM)
c
c...Calculate temporary latest finish day number
c...(ITIME) and day of the week number (ITEMP)
c
          ITIME=LSDNO(ITO)-IDDCT
          ITEMP=IDOTW(ITIME)
c
c...Using the Function (ISDEDC), Calculate
c...the number of days to be subtarcted
c
          ISDDCT=ISDEDC(IFROM,ITEMP)
```

```
c
c...Adjust (ITIME) and Recalculate (ITEMP)
c
          ITIME=ITIME-ISDDCT
          ITEMP=IDOTW(ITIME)
c
c...Calculate the non-working days for the link duration
c
          ILDDCT=(INT((ILDURN(I,2)-ITEMP+TWK(IFROM)-1)/TWK(IFROM)))*(7-T
     &WK(IFROM))
c
c...Recalculate (ITIME) and (ITEMP)
c
          ITIME=ITIME-ILDURN(I,2)-ILDDCT
          ITEMP=IDOTW(ITIME)
c
c...Calculate the number of non-working days to be deducted
c
          ISDDCT=ISDEDC(IFROM,ITEMP)
c
c...Adjust (ITIME)
c
          ITIME=ITIME-ISDDCT
c
c...For (F-F) link type
c
        ELSEIF(LTYPE(I,2).EQ.3)THEN
c
c...Calculate the non-working days of the link duration
c
          ILDDCT=(INT((ILDURN(I,2)-LFDOW(ITO)+TWK(ITO)-1)/TWK(ITO)))*(7-
     &TWK(ITO))
c
c...Calculate (ITIME) and (ITEMP)
c
          ITIME=LFDNO(ITO)-ILDURN(I,2)-ILDDCT
          ITEMP=IDOTW(ITIME)
c
c...Calculate the additional non-working days to be subtracted
c
          ISDDCT=ISDEDC(IFROM,ITEMP)
c
c...Adjust the temporary latest finish day number (ITIME)
c
          ITIME=ITIME-ISDDCT
c
c...For (S-S) or (S-F) link types
c
        ELSE
c
c...NO calculation for this types of link is required
c
          ITIME=LFDNO(IFROM)

        ENDIF
```

```
c
c...Compare (ITIME) with the stored (LFDNO) and divert the program
c
        IF(ITIME.GE.LFDNO(IFROM))GOTO90
c
c...Update  (LFDNO) and calculate (LFDOW)
c
        LFDNO(IFROM)=ITIME
        LFDOW(IFROM)=IDOTW(ITIME)
c
c...Update Repeating loop counter
c
        K=K+1
c
c...Calculation of latest start day number  (LSDNO)
c...and latest start day of the week number (LSDOW)
c
c...For 'F-S' or 'F-F' link types
c
  90    IF(LTYPE(I,2).EQ.1.OR.LTYPE(I,2).EQ.3)THEN
c
c...Calculate the non-working days
c
          IDDUCT=(INT((IDURN(IFROM)-LFDOW(IFROM)+TWK(IFROM)-1)/TWK(IFROM
     &)))*(7-TWK(IFROM))
c
c...Calculate additional days  1 or 0 (because of using day numbers)
c
          ITNADD=SFCODE(IFROM)
c
c...Using task (LFDNO); Calculate the temporary latest start
c...day number (ITIME2)  and day of the week number (ITEMP2)
c
          ITIME2=LFDNO(IFROM)-IDURN(IFROM)-IDDUCT+ITNADD
          ITEMP2=IDOTW(ITIME2)
c
c...Ckeck if any non-working days to be added and calculate it
c
          ISADD=ISADED(IFROM,ITEMP2)
c
c...Adjust the temporary earliest start day number
c
          ITIME2=ITIME2+ISADD
c
c...For a 'S-S' link type
c
        ELSEIF(LTYPE(I,2).EQ.2)THEN
c
c...Calculate the number of non-working days
c
          IDDUCT=(INT((ILDURN(I,2)-LSDOW(ITO)+TWK(IFROM)-1)/TWK(IFROM)))
     &*(7-TWK(IFROM))
```

```
c
c...Calculate (ITIME2) and (ITEMP2)
c
          ITIME2=LSDNO(ITO)-ILDURN(I,2)-IDDUCT
          ITEMP2=IDOTW(ITIME2)
c
c...Calculate the number of non-working days
c
          ISDDCT=ISDEDC(IFROM,ITEMP2)
c
c...Adjust the temporary latest start day number (ITIME2)
c
          ITIME2=ITIME2-ISDDCT
c
c...For 'S-F' link type
c
        ELSE
c
c...Calculate the number of non-working days
c
          IDDUCT=(INT((ILDURN(I,2)-LFDOW(ITO)+TWK(IFROM)-1)/TWK(IFROM)))
     &*(7-TWK(IFROM))
c
c...Calculate (ITIME2)
c
          ITIME2=LFDNO(ITO)-ILDURN(I,2)-IDDUCT+1
        ENDIF
c
c...Compare the temporary with the stored (LSDNO) and divert program
c
        IF(ITIME2.GE.LSDNO(IFROM))GOTO100
c
c...Store (ITIME2) as precceding task (LSDNO) and calculate (LSDOW)
c
        LSDNO(IFROM)=ITIME2
        LSDOW(IFROM)=IDOTW(ITIME2)
 100    CONTINUE
c
c...Counter updated; Repeat the loop
c
      IF(K.GT.0)GOTO80
c
c...Counter equal to zero (not updated)
c
c...Calculate the project finish day of the week number
c
      IPFDOW=IDOTW(IPFDNO)
c
c...Project finish day is the first day of the week
c
      IF(IPFDOW.EQ.1)THEN
```

```
c
c...Deduct the number of non-working days to calculate the
c...exact project finish day and day of the week numbers
c
        IPFDNO=IPFDNO-(8-TWK(NFINISH))
        IPFDOW=IDOTW(IPFDNO)
c
c...Project finish day is not the first day in the week
c
      ELSE
c
c...Calculate the exact project finish day and day of the week numbers
c
        IPFDNO=IPFDNO-1
        IPFDOW=IDOTW(IPFDNO)
      ENDIF
c
c...Calculate the project duration (IPRDRN) in days
c
      IPRDRN(N)=IPFDNO-IESDNO(NSTART)+1
c
c...Initialise (IEFDNO) for hammock tasks
c
      DO110,I=1,NTASK
c
c...Select hammock task
c
        IF(TTYPE(I).NE.'H')GOTO110
        IEFDNO(I)=IPFDNO
 110    CONTINUE
c
c...Calculation of HAMMOCK tasks durations
c
      DO140,I=1,NTASK
c
c...Select the hammock task
c
        IF(TTYPE(I).NE.'H')GOTO140
        DO120,J=1,NLINK
c
c...Using forward pass; Select the precceding task
c
          IF(LTO(J,1).NE.I)GOTO120
          IFROM=LFROM(J,1)
          ITO=LTO(J,1)
c
c...Identify the forward pass number
c
          M=1
c
c...Calculate the absolute Integer link duration
c
          ILDURN(J,1)=LINKDN(J,M,IFROM)
```

```
c
c...Perform calculations for 'S-S' link type
c
      IF(LTYPE(J,1).EQ.2)THEN
c
c...Calculate the non-working days to be added to the link duration
c
        ILADD=(INT((ILDURN(J,1)+IESDOW(IFROM)-2)/TWK(IFROM)))*(7-TWK
     &(IFROM))
c
c...Calculate tempeorary Earliest start day number
c
        ITIME=IESDNO(IFROM)+ILDURN(J,1)+ILADD
c
c...Perform calculations for 'F-S' link type
c
      ELSEIF(LTYPE(J,1).EQ.1)THEN
c
c...Calculate the tempeorary earliest start day number
c
        ITSDNO=IESDNO(IFROM)+IDURN(IFROM)+IDURN2(IFROM)
        ITSDOW=IDOTW(ITSDNO)
        ISADD=ISADED(I,ITSDOW)
        IESLD=ITSDNO+ISADD
        IOWLIN=IDOTW(IESLD)
c
c...Calculate the additional days to the link duration
c
        ILAD=(INT((ILDURN(J,1)+IOWLIN-2)/TWK(IFROM)))*(7-TWK(IFROM))
c
c...Recalculate the tempeorary earliest start day number
c
        ITIME=ITSDNO+ILDURN(J,1)+ILAD
      ENDIF
c
c...Calculate the temperoray earliest start day of the week number
c
        ITEMP=IDOTW(ITIME)
c
c...Check if any days to be added and calculate
c
        ISADD=ISADED(ITO,ITEMP)
c
c...Adjust (ITIME)
c
        ITIME=ITIME+ISADD
c
c...Calculate the Earliest Start  day  number (IESDNO)
c...and Earliest start day of the week number (IESDOW)
c
        IF(ITIME.LE.IESDNO(I))GOTO120
        IESDNO(I)=ITIME
        IESDOW(I)=IDOTW(ITIME)
 120  CONTINUE
```

```
c
c...Using the forward pass; Select the succeeding task
c
        DO130,K=1,NLINK
          IF(LFROM(K,1).NE.I)GOTO130
          IFROM=LFROM(K,1)
          ITO=LTO(K,1)
          M=1
c
c...Calculate the Integer link duration.
c
          ILDURN(K,1)=LINKDN(K,M,IFROM)
c
c...Perform calculations for 'F-F' link type
c
          IF(LTYPE(K,1).EQ.3)THEN
c
c...If the succeding task is 'FINISH' task
c
           IF(SFCODE(ITO).EQ.2)THEN
c
c...Adjust the Earliest finish day number
c
            ITEMP2=IEFDNO(ITO)-1
            ITEMP3=IDOTW(ITEMP2)
            ISDDCT=ISDEDC(ITO,ITEMP3)
            ITEMP2=ITEMP2-ISDDCT
            IDOWT=IDOTW(ITEMP2)
c
c...Calculate the additional non working days to the link duration
c
            IDDUCT=(INT((ILDURN(K,1)-IDOWT+TWK(ITO)-1)/TWK(ITO)))*(7-TWK
     &(ITO))
c
c...Calculate the tempeorary earliest finish day number
c...and earliest finish day of the week number
c
            IEFDNT=ITEMP2-ILDURN(K,1)-IDDUCT
            IEFDWT=IDOTW(IEFDNT)
c
c...For any other type of succeeding task
c
           ELSE
c
c...Calculate the addtional non working days to link duration
c
            IDDUCT=(INT((ILDURN(K,1)-IEFDOW(ITO)+TWK(ITO)-1)/TWK(ITO)))*
     &(7-TWK(ITO))
c
c...Calculate the tempeorary earliest finish day number
c...and earliest finish day of the week number
c
            IEFDNT=IEFDNO(ITO)-ILDURN(K,1)-IDDUCT
            IEFDWT=IDOTW(IEFDNT)
           ENDIF
```

```
c
c...For 'F-S' link type
c
        ELSEIF(LTYPE(K,1).EQ.1)THEN
c
c...Calculate tempeorarily earliest finish day number
c...and  earliest  finish  day  of  the week   number
c
          ITIME=IESDNO(ITO)-1
          ITEMP=IDOTW(ITIME)
c
c...Check if any number of days to be subtracted and calculate it
c
          ISDDCT=ISDEDC(IFROM,ITEMP)
c
c...Adjust earliest finish day and day of the week numbers
c
          IEFDNT=ITEMP-ISDDCT
          IEFDWT=IDOTW(IEFDNT)
c
c...calculate the additional number of non working days
c
          IDDUCT=(INT((ILDURN(K,1)-IEFDWT+TWK(IFROM)-1)/TWK(IFROM)))*(
     &7-TWK(IFROM))
c
c...Recalculate tempeorarily day and day of the week numbers
c
          IEFDNT=IEFDNT-ILDURN(K,1)-IDDUCT
          IEFDWT=IDOTW(IEFDNT)
        ELSE
          IEFDNT=IEFDNO(IFROM)
          IEFDWT=IDOTW(IEFDNT)
        ENDIF
c
c...Check if any non working days to be subtarcted and calculate it
c
          ISDDCT=ISDEDC(IFROM,IEFDWT)
c
c...Calculate (IEFDNO), (IEFDOW)
c
          IEFDNT=IEFDNT-ISDDCT
          IF(IEFDNT.GE.IEFDNO(IFROM))GOTO130
          IEFDNO(I)=IEFDNT
          IEFDOW(I)=IDOTW(IEFDNO(I))
 130    CONTINUE
c
c...Calculate the hammock task total duration
c...(including non working days)
c
        IHTDRN=IEFDNO(I)-IESDNO(I)+1
c
c...Calculate the number of non working days
c
        IDURN2(I)=(INT((IHTDRN+IESDOW(I)-1)/7))*(7-TWK(I))
```

```
c
c...Calculate Hammock task duration
c
          IDURN(I)=IHTDRN-IDURN2(I)
 140   CONTINUE
c
c...Update criticality Index (ICRIT)
c
      DO150,I=1,NTASK
c
c...Ignore Hammock tasks
c
        IF(TTYPE(I).EQ.'H')GOTO150
c
c...Calculate the float
c
        IFLOAT=LFDNO(I)-(IESDNO(I)+IDURN(I)+IDURN2(I))+1
c
c...Check if task is critical
c
        IF(IFLOAT.LE.0)ICRIT(I)=ICRIT(I)+1
 150   CONTINUE
c
c...Exit
c
      RETURN
      END
```

Appendix F.5: Function LINKDN

```
      INTEGER FUNCTION LINKDN(I,M,IFROM)
c***********************************************************************
c**                                                                   **
c**                      FUNCTION LINKDN                              **
c**                                                                   **
c**        Changes (S-S) link duration to absolute durations          **
c**        Changes all types of link  durations to  Integers          **
c**                                                                   **
c***********************************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400,3),SFCODE(400),
     &TWK(400),TCOST(400,9),TWBS(400),PCCODE(400),IDURN(400)
      COMMON/LINKS/LFROM(500,2),LTO(500,2),LTYPE(500,2),LDURN(500,2)
c
      CHARACTER*1 TTYPE
      CHARACTER*9 TNAME
      CHARACTER*12 TWBS
c
      REAL TDURN,LDURN,TCOST
      INTEGER SFCODE,TWK,PCCODE
c
c...Change (S-S) link durations from percentage to absolute
c
      IF(LTYPE(I,M).EQ.2)THEN
         TEMP=LDURN(I,M)*FLOAT(IDURN(IFROM))
c
c...Change link duration to Integer
c
         LINKDN=INT(TEMP)
         IF(TEMP.GT.AINT(TEMP))LINKDN=LINKDN+1
      ELSE
c
c...Change other types of link durations to Integers
c
         LINKDN=INT(LDURN(I,M))
         IF(LDURN(I,M).GT.AINT(LDURN(I,M)))LINKDN=LINKDN+1
      ENDIF
c
c...Exit
c
      RETURN
      END
```

Appendix F.6: Function ISADED

```
      INTEGER FUNCTION ISADED(I,IDOWNO)
c***********************************************************************
c**                                                                   **
c**                         FUNCTION ISADED                           **
c**         Returns the number of non-working days (weekends)         **
c**         to be added to the tasks durations to calculate the       **
c**        Ealiest start day number of a succeeding task(IESDNO)      **
c**                                                                   **
c***********************************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400,3),SFCODE(400),
     &TWK(400),TCOST(400,9),TWBS(400),PCCODE(400),IDURN(400)
c
      CHARACTER*1 TTYPE
      CHARACTER*9 TNAME
      CHARACTER*12 TWBS
c
      REAL TDURN
      INTEGER SFCODE,TWK,PCCODE
c
c...Calculate the additional number of days
c
      IF(IDOWNO.GT.TWK(I))THEN
        ISADED=8-IDOWNO
      ELSE
        ISADED=0
      ENDIF
      RETURN
      END
```

Appendix F.7: Function ISDEDC

```
      INTEGER FUNCTION ISDEDC(I,IDOWNO)
c***********************************************************************
c**                                                                   **
c**                      FUNCTION ISDEDC                              **
c**         Returns the number of non-working days (week-ends)        **
c**        to be deducted from the tasks durations to calculate       **
c**     the Latest Finisht day number of a precceeding task(LFDNO)    **
c**                                                                   **
c***********************************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400,3),SFCODE(400),
     &TWK(400),TCOST(400,9),TWBS(400),PCCODE(400),IDURN(400)
c
      CHARACTER*1 TTYPE
      CHARACTER*9 TNAME
      CHARACTER*12 TWBS
c
      REAL TDURN
      INTEGER SFCODE,TWK,PCCODE
c
c...Calculate the additional number of days
c
      IF(IDOWNO.GT.TWK(I))THEN
        ISDEDC=IDOWNO-TWK(I)
      ELSE
        ISDEDC=0
      ENDIF
      RETURN
      END
```

Appendix F.8: Subroutine TRIDIS

```
      SUBROUTINE TRIDIS(I)
c*********************************************************************
c**                                                                 **
c**                      SUBROUTINE TRIDIS                          **
c**                                                                 **
c**  Calculates triangular distribution parameters for the Ith task **
c**                                                                 **
c*********************************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400,3),SFCODE(400),
     &TWK(400),TCOST(400,9),TWBS(400),PCCODE(400),IDURN(400)
      COMMON/PDIST/DLIM(400,2),FMODE(400)
c
      CHARACTER*1 TTYPE
      CHARACTER*9 TNAME
      CHARACTER*12 TWBS
c
      REAL TDURN,TCOST
      INTEGER SFCODE,TWK,PCCODE
c
c...Define calculation convergence criterion
c
      CONVER=1.0E-6
c
c...Declare initial values of distribution parameters.
c
      DEL1=0.0
      DEL2=TDURN(I,2)-TDURN(I,1)
      DEL3=TDURN(I,3)-TDURN(I,2)
      DEL4=0.0
c
c...Update values of distribution parameters.
c
   10 FMODE(I)=2.0/(DEL1+DEL2+DEL3+DEL4)
      IF(DEL2.GT.0.0)THEN
        DEL1=(0.02+SQRT(0.02**2+0.08*FMODE(I)*DEL2))/(2.0*FMODE(I))
      ENDIF
      DEL4=(0.02+SQRT(0.02**2+0.08*FMODE(I)*DEL3))/(2.0*FMODE(I))
      PTOTAL=0.5*FMODE(I)*(DEL1+DEL2+DEL3+DEL4)
      IF((ABS(1.0-PTOTAL)).GT.CONVER)GOTO10
c
c...Calculations have converged; Define duration limits
c
      DLIM(I,1)=TDURN(I,1)-DEL1
      DLIM(I,2)=TDURN(I,3)+DEL4
c
c...Exit
c
      RETURN
      END
```

Appendix F.9: Subroutine DSELECT

```
      SUBROUTINE DSELECT(I,SEED)
c***********************************************************************
c**                                                                   **
c**                        SUBROUTINE DSELECT                         **
c**                                                                   **
c**       Randomly selects an INTEGER task duration IDURN on          **
c**       the basis of the duration  distribution parameters          **
c**                calculated by Subroutine TRIDIS                    **
c**                                                                   **
c***********************************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400,3),SFCODE(400),
     &TWK(400),TCOST(400,9),TWBS(400),PCCODE(400),IDURN(400)
      COMMON/PDIST/DLIM(400,2),FMODE(400)
c
      CHARACTER*1 TTYPE
      CHARACTER*9 TNAME
      CHARACTER*12 TWBS
c
      REAL TDURN,TCOST
      INTEGER SFCODE,TWK,PCCODE
c
c...Calculate RH slope of distribution
c
      S1=FMODE(I)/(DLIM(I,2)-TDURN(I,2))
c
c...Check whether duration can be less than modal duration
c
      DLEFT=TDURN(I,2)-DLIM(I,1)
      IF(DLEFT.GT.0.0)THEN
c
c...It can be; Calculate LH slope and area
c
        S0=FMODE(I)/DLEFT
        A0=0.5*DLEFT**2*FMODE(I)
c
c...It can't be; Set LH area to zero
c
      ELSE
        A0=0.0
      ENDIF
c
c...Randomly select an integer duration
c
      A=RAND(SEED)
      IF(A.LT.A0)THEN
        DURN=DLIM(I,1)+SQRT(2.0*A/S0)
      ELSE
        DURN=DLIM(I,2)-SQRT(2.0*(1.0-A)/S1)
      ENDIF
      IDURN(I)=INT(DURN)
      IF(DURN.GT.AINT(DURN))IDURN(I)=IDURN(I)+1
```

```
c
c...Exit
c
      RETURN
      END
```

Appendix F.10: Function RAND

```
      FUNCTION RAND(SEED)
c************************************************************************
c**                                                                    **
c**                       FUNCTION RAND                                **
c**                                                                    **
c**        Returns a random REAL number in the range 0. to 1.0         **
c**                                                                    **
c**          Requires initialisation by  0.0 < seed < 259200           **
c**                                                                    **
c************************************************************************
c
      PARAMETER(IA=7141,IC=54773,IM=259200)
c
      SEED=REAL(MOD(INT(SEED)*IA+IC,IM))
      RAND=SEED/REAL(IM)
      RETURN
      END
```

Appendix F.11: Subroutine PCOSTS

```
      SUBROUTINE PCOSTS(NSTART,NTASK,ICDAYS,N)
c***********************************************************************
c**                                                                   **
c**                        SUBROUTINE PCOSTS                          **
c**                                                                   **
c**         Distribute tasks costs on the days of the project         **
c**         Calculates project daily costs and tasks prices           **
c**         Accumulate and arrange cost and tasks prices and          **
c**     calculates the CAPTIM and Internal rate of return (IRR)       **
c**                                                                   **
c***********************************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400,3),SFCODE(400),
     &TWK(400),TCOST(400,9),TWBS(400),PCCODE(400),IDURN(400)
      COMMON/DAYNOS/IESDNO(400),IESDOW(400),IDURN2(400),LFDNO(400),
     &LFDOW(400),IDURN3(400),ILDURN(500,2),ICRIT(400),ACRIT(400),
     &IPRDRN(1000),IEFDNO(400),IEFDOW(400),LSDNO(400),LSDOW(400)
      COMMON/FINSTA/ISTART(400),IFINISH(400),NDAYOW(1000)
      COMMON/PDCOST/DCOSTS(500,9),COSPAY(500,9),COSDLY(500,9),
     &IDDATE(500),MACDAY(50),TOTCOS(500),ACTOTC(500),ACCTPR(500),
     &CAPTIM(1000),RATERN(1000)
      COMMON/COSTS/CCODE(9),CPAY(9),CARR(9),CARRQ(9)
c
      CHARACTER*1 TTYPE,CPAY,CARR
      CHARACTER*3 CCODE
      CHARACTER*9 TNAME
      CHARACTER*12 TWBS
c
      REAL TDURN,TCOST,X
      INTEGER SFCODE,TWK,PCCODE,CARRQ
c
c...Calculate the maximum payment delay
c
      MAXDLY=0
      DO5,I=1,9
        IF(CARR(I).EQ.'D')THEN
          IF(MAXDLY.LT.CARRQ(I))MAXDLY=CARRQ(I)
        ELSEIF(CARR(I).EQ.'W')THEN
          NDAYS=CARRQ(I)*7
          IF(MAXDLY.LT.NDAYS)MAXDLY=NDAYS
        ELSEIF(CARR(I).EQ.'M')THEN
          NDAYS=CARRQ(I)*30
          IF(MAXDLY.LT.NDAYS)MAXDLY=NDAYS
        ENDIF
  5     CONTINUE
c
c...Add maximum payment delay to the project duration
c...Add 30 Days to cover the payment method.........
c
      ICDAYS=IPRDRN(N)+MAXDLY+30
```

```
c
c...Calculate start (ISTART) and finish
c...(IFINSH) day number for each task
c
      DO10,I=1,NTASK
        ISTART(I)=IESDNO(I)-IESDNO(NSTART)+1
        IFINISH(I)=ISTART(I)+IDURN(I)+IDURN2(I)-1
  10    CONTINUE
c
c...Calculate the day of the week number (NDAYOW)
c...and date (IDDATE) for each day
c
      IDAYNO=IESDNO(NSTART)
      DO20,I=1,ICDAYS
        NDAYOW(I)=IDOTW(IDAYNO)
        IDDATE(I)=IDAY(IDAYNO)
        IDAYNO=IDAYNO+1
  20    CONTINUE
c
c...Find out the monthly accumulation days (MACDAY)
c
      J=1
      DO30,I=2,ICDAYS
        IF(IDDATE(I).NE.1)GOTO30
          MACDAY(J)=I-1
          J=J+1
  30    CONTINUE
c
c...Initialise daily costs arrays
c
      DO50,K=1,9
        DO40,I=1,ICDAYS
          DCOSTS(I,K)=0.0
          COSPAY(I,K)=0.0
          COSDLY(I,K)=0.0
  40      CONTINUE
  50    CONTINUE
c
c...Ditribute tasks daily labour cost on
c...the project days (Normal loading)
c
      DO70,I=1,ICDAYS
        DO60,J=1,NTASK
          IF(NDAYOW(I).GT.TWK(J))GOTO60
          IF(I.GE.ISTART(J).AND.I.LE.IFINISH(J))DCOSTS(I,1)=DCOSTS(I,1)+
     &TCOST(J,1)
  60      CONTINUE
  70    CONTINUE
```

```
c
c...Materials cost (Front loading)
c
      DO90,I=1,ICDAYS
        DO80,J=1,NTASK
          IF(I.EQ.ISTART(J))DCOSTS(I,2)=DCOSTS(I,2)+TCOST(J,2)
  80      CONTINUE
  90    CONTINUE
c
c...Daily plant costs  (Normal loading)
c
      DO110,I=1,ICDAYS
        DO100,J=1,NTASK
          IF(NDAYOW(I).GT.TWK(J))GOTO100
          IF(I.GE.ISTART(J).AND.I.LE.IFINISH(J))DCOSTS(I,3)=DCOSTS(I,3)+
     &TCOST(J,3)
 100      CONTINUE
 110    CONTINUE
c
c...Front-end plant cost (Front loadin)
c
      DO130,I=1,ICDAYS
        DO120,J=1,NTASK
          IF(I.EQ.ISTART(J))DCOSTS(I,4)=DCOSTS(I,4)+TCOST(J,4)
 120      CONTINUE
 130    CONTINUE
c
c...Back-end plant cost (Back loading)
c
      DO150,I=1,ICDAYS
        DO140,J=1,NTASK
          IF(I.EQ.IFINISH(J))DCOSTS(I,5)=DCOSTS(I,5)+TCOST(J,5)
 140      CONTINUE
 150    CONTINUE
c
c...Subcontracts cost (Spread loading)
c
      DO170,I=1,ICDAYS
        DO160,J=1,NTASK
          IF(NDAYOW(I).GT.TWK(J))GOTO160
          IF(TCOST(J,6).EQ.0.0)GOTO160
          IF(I.GE.ISTART(J).AND.I.LE.IFINISH(J))DCOSTS(I,6)=DCOSTS(I,6)+
     &(TCOST(J,6)/IDURN(J))
 160      CONTINUE
 170    CONTINUE
c
c...Daily overheads cost (Normal)
c
      DO190,I=1,ICDAYS
        DO180,J=1,NTASK
          IF(NDAYOW(I).GT.TWK(J))GOTO180
          IF(I.GE.ISTART(J).AND.I.LE.IFINISH(J))DCOSTS(I,7)=DCOSTS(I,7)+
     &TCOST(J,7)
 180      CONTINUE
 190    CONTINUE
```

```
c
c...Front-end overheads cost (Front loading)
c
      DO210,I=1,ICDAYS
        DO200,J=1,NTASK
          IF(I.EQ.ISTART(J))DCOSTS(I,8)=DCOSTS(I,8)+TCOST(J,8)
 200      CONTINUE
 210    CONTINUE
c
c...Task prices (Spread loading)
c
      DO230,I=1,ICDAYS
        DO220,J=1,NTASK
          IF(NDAYOW(I).GT.TWK(J))GOTO220
          IF(TCOST(J,9).EQ.0.0)GOTO220
          IF(I.GE.ISTART(J).AND.I.LE.IFINISH(J))DCOSTS(I,9)=DCOSTS(I,9)+
     &(TCOST(J,9)/IDURN(J))
 220      CONTINUE
 230    CONTINUE
c
c...Accumulate Daily cost according to the payment method
c
      DO320,K=1,9
c
c...Weekly payment
c
      IF(CPAY(K).EQ.'W')THEN
        ACWCOS=0.0
        DO240,I=1,ICDAYS
          IF(NDAYOW(I).NE.5)THEN
            ACWCOS=ACWCOS+DCOSTS(I,K)
          ELSE
            COSPAY(I,K)=ACWCOS+DCOSTS(I,K)
            ACWCOS=0.0
          ENDIF
 240      CONTINUE
      ELSE
c
c...Monthly payment
c
        J=1
        ACMCOS=0.0
        DO250,I=1,ICDAYS
          IF(I.NE.MACDAY(J))THEN
            ACMCOS=ACMCOS+DCOSTS(I,K)
          ELSE
            COSPAY(I,K)=ACMCOS+DCOSTS(I,K)
            J=J+1
            ACMCOS=0.0
          ENDIF
 250      CONTINUE
      ENDIF
```

```
c
c...Arrange Commulative Cost according to the given payment delay
c
      IF(CARR(K).EQ.'0')THEN
c
c...Payment delay not applicable
c
        DO260,I=1,ICDAYS
          COSDLY(I,K)=COSPAY(I,K)
 260      CONTINUE
c
c...Payment delay in days
c
      ELSEIF(CARR(K).EQ.'D')THEN
        IARR=CARRQ(K)
        DO270,I=1,ICDAYS
          J=I+IARR
          COSDLY(J,K)=COSPAY(I,K)
 270      CONTINUE
c
c...Payment delay in weeks
c
      ELSEIF(CARR(K).EQ.'W')THEN
        IARR=CARRQ(K)*7
        DO280,I=1,ICDAYS
          J=I+IARR
          COSDLY(J,K)=COSPAY(I,K)
 280      CONTINUE
c
c...Payment delay in months
c
      ELSE
        IF(CPAY(K).EQ.'W')THEN
          IARR=CARRQ(K)*30
          DO290,I=1,ICDAYS
            J=I+IARR
            COSDLY(J,K)=COSPAY(I,K)
 290        CONTINUE
        ELSE
          J=1
          M=J+CARRQ(K)
          DO300,I=1,ICDAYS
            IF(I.NE.MACDAY(M))GOTO300
            COSDLY(I,K)=COSPAY(MACDAY(J),K)
            J=J+1
            M=M+1
 300        CONTINUE
        ENDIF
      ENDIF
```

```
c
c...Adjust payments not to be in week-ends
c
      DO310,I=1,ICDAYS
      IF(COSDLY(I,K).EQ.0.0)GOTO310
         IF(NDAYOW(I).EQ.6)THEN
           COSDLY(I-1,K)=COSDLY(I,K)+COSDLY(I-1,K)
           COSDLY(I,K)=0.0
         ELSEIF(NDAYOW(I).EQ.7)THEN
           COSDLY(I-2,K)=COSDLY(I,K)+COSDLY(I-2,K)
           COSDLY(I,K)=0.0
         ENDIF
 310     CONTINUE
 320    CONTINUE
c
c...Calculate Tolal cost (TOTCOS)
c
      DO330,I=1,ICDAYS
        TOTCOS(I)=COSDLY(I,1)+COSDLY(I,2)+COSDLY(I,3)+COSDLY(I,4)+
     &COSDLY(I,5)+COSDLY(I,6)+COSDLY(I,7)+COSDLY(I,8)
 330    CONTINUE
c
c...Intialise commulative total cost  (ACTOTC)
c
      DO340,I=0,ICDAYS
        ACTOTC(I)=0.0
 340    CONTINUE
c
c...Accumulate total cost
c
      DO350,I-1,ICDAYS
        ACTOTC(I)=ACTOTC(I-1)+TOTCOS(I)
 350    CONTINUE
c
c...Initilise commulative Task prices (ACCTPR)
c
      DO360,I=0,ICDAYS
        ACCTPR(I)=0.0
 360    CONTINUE
c
c...Accumulate Task Price
c
      DO370,I=1,ICDAYS
        ACCTPR(I)=ACCTPR(I-1)+COSDLY(I,9)
 370    CONTINUE
c
c...Initialise CAPTIM
c
      CAPTIM(N)=0.0
c
c...Calculate the CAPTIM
c
      DO380,I=1,ICDAYS
        CAPTIM(N)=CAPTIM(N)+DIM(ACTOTC(I),ACCTPR(I))
 380    CONTINUE
```

```
c
c...Initialise annual interest rate (X)
c
      X=-1.0
c
c...Set Increasing of annual Interest rate (DELX) to 10%
c
      DELX=0.1
      X=X+DELX
c
c...Initialise Net Present Value (ACCNPV)
c
 390  ACCNPV=0.0
       R=((1.0+X)**(1.0/365.0))-1.0
c
c...Calculate Net Present Value (ACCNPV)
c
      DO400,I=1,ICDAYS
        ACCNPV=ACCNPV+(COSDLY(I,9)-TOTCOS(I))/((1.0+R)**(I-1))
 400    CONTINUE
c
c...Net preseent value is negative
c
      IF(ACCNPV.GT.0.0)THEN
c
c...Increase Annual Intrest rate and Re-calculate (ACCNPV)
c
        X=X+DELX
        GOTO390
      ELSE
c
c...Net Present Value is Positive
c...Deduct the previously seted DELX
c
        X=X-DELX
c
c...Reset DELX to 10% of the previous one
c
        DELX=DELX/10.0
c
c...Recalculate the annual Interest rate
c
        X=X+DELX
      ENDIF
      IF(DELX.LE.1.0E-5)GOTO410
      GOTO390
c
c...Calculate Internal Rate of return
c
 410  RATERN(N)=X*100.0
c
c...Exit
c
      RETURN
      END
```

Appendix F.12: Subroutine PSORT

```
      SUBROUTINE PSORT(NRUNS)
c***********************************************************************
c**                                                                   **
c**                          SUBROUTINE PSORT                         **
c**           Sorts NRUNS project durations into Rank order           **
c**                                                                   **
c***********************************************************************
c
      COMMON/DAYNOS/IESDNO(400),IESDOW(400),IDURN2(400),LFDNO(400),
     &LFDOW(400),IDURN3(400),ILDURN(500,2),ICRIT(400),ACRIT(400),
     &IPRDRN(1000),IEFDNO(400),IEFDOW(400),LSDNO(400),LSDOW(400)
c
c...Sort project durations into rank order
c
      DO20,I=1,(NRUNS-1)
        DO10,J=1,(NRUNS-I)
          K=J+1
          IF(IPRDRN(J).LE.IPRDRN(K))GOTO10
          ITEMP=IPRDRN(J)
          IPRDRN(J)=IPRDRN(K)
          IPRDRN(K)=ITEMP
  10      CONTINUE
  20    CONTINUE
c
c...Exit
c
      RETURN
      END
```

Appendix F.13: Subroutine PRSTAT

```
      SUBROUTINE PRSTAT(NRUNS)
c***********************************************************************
c**                                                                   **
c**                        SUBROUTINE PRSTAT                          **
c**                                                                   **
c**    Calculates Minimum, Maximum, Median, Mean project durations    **
c**                       and Project Skewness                        **
c**                                                                   **
c***********************************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400,3),SFCODE(400),
     &TWK(400),TCOST(400,9),TWBS(400),PCCODE(400),IDURN(400)
      COMMON/DAYNOS/IESDNO(400),IESDOW(400),IDURN2(400),LFDNO(400),
     &LFDOW(400),IDURN3(400),ILDURN(500,2),ICRIT(400),ACRIT(400),
     &IPRDRN(1000),IEFDNO(400),IEFDOW(400),LSDNO(400),LSDOW(400)
      COMMON/STATS/MINPDU,MAXPDU,PRMEAN,PSKEW,MEDIAN,PROSIG
c
      CHARACTER*1 TTYPE
      CHARACTER*9 TNAME
      CHARACTER*12 TWBS
c
      REAL MEDIAN,TDURN,TCOST
c
      INTEGER SFCODE,TWK,PCCODE
c
c...Initialise minimum, maximum, sum and sum square of project durations
c
      MINPDU=1E6
      MAXPDU=0
      ISUM=0
      ISUMSQ=0
c
c...Calculate minimum, maximum, sum and sum square
c
      DO10,I=1,NRUNS
         IF(IPRDRN(I).LT.MINPDU)MINPDU=IPRDRN(I)
         IF(IPRDRN(I).GT.MAXPDU)MAXPDU=IPRDRN(I)
         ISUM=ISUM+IPRDRN(I)
         ISUMSQ=ISUMSQ+IPRDRN(I)**2
  10     CONTINUE
c
c...Calculate Mean project duration and Standard deviation
c
      PRMEAN=FLOAT(ISUM)/FLOAT(NRUNS)
      PROSIG=SQRT((FLOAT(ISUMSQ)-FLOAT(ISUM)**2/FLOAT(NRUNS))/FLOAT(NRUN
     &S-1))
```

```
c
c...Calculate the Median project duration
c
      I=(NRUNS+1)/2
      J=I+1
      IF(MOD(NRUNS,2).LT.1)THEN
        MEDIAN=FLOAT(IPRDRN(I)+IPRDRN(J))/2.0
      ELSE
        MEDIAN=FLOAT(IPRDRN(I))
      ENDIF
c
c...Calculate project durations Skewnees
c
      PSKEW=3.0*(PRMEAN-MEDIAN)/PROSIG
c
c...Exit
c
      RETURN
      END
```

Appendix F.14: Subroutine CSTAT

```
      SUBROUTINE CASTAT(NRUNS)
c************************************************************************
c**                                                                    **
c**                         SUBROUTINE CASTAT                          **
c**                                                                    **
c**     Calculates Minimum, Maximum, Median, Mean CAPTIM               **
c**                      and CAPTIM  Skewness                          **
c**                                                                    **
c************************************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400,3),SFCODE(400),
     &TWK(400),TCOST(400,9),TWBS(400),PCCODE(400),IDURN(400)
      COMMON/DAYNOS/IESDNO(400),IESDOW(400),IDURN2(400),LFDNO(400),
     &LFDOW(400),IDURN3(400),ILDURN(500,2),ICRIT(400),ACRIT(400),
     &IPRDRN(1000),IEFDNO(400),IEFDOW(400),LSDNO(400),LSDOW(400)
      COMMON/STAT3/CAPMIN,CAPMAX,CAMEAN,CSKEW,CMEDIAN,CAPSIG
      COMMON/PDCOST/DCOSTS(500,9),COSPAY(500,9),COSDLY(500,9),
     &IDDATE(500),MACDAY(50),TOTCOS(500),ACTOTC(500),ACCTPR(500),
     &CAPTIM(1000),RATERN(1000)
c
      CHARACTER*1 TTYPE
      CHARACTER*9 TNAME
      CHARACTER*12 TWBS
c
      REAL TDURN,TCOST
c
      INTEGER SFCODE,TWK,PCCODE
c
c...Initialise minimum, maximum, sum and sum square of CAPTIM
c
      CAPMIN=1.0E12
      CAPMAX=0.0
      SUM=0.0
      SUMSQ=0.0
c
c...Calculate CAPTIM minimum, maximum, sum and sum square
c
      DO10,I=1,NRUNS
         IF(CAPTIM(I).LT.CAPMIN)CAPMIN=CAPTIM(I)
         IF(CAPTIM(I).GT.CAPMAX)CAPMAX=CAPTIM(I)
         SUM=SUM+CAPTIM(I)
         SUMSQ=SUMSQ+CAPTIM(I)**2
  10     CONTINUE
c
c...Calculate Mean CAPTIM and Standard deviation
c
      CAMEAN=SUM/FLOAT(NRUNS)
      CAPSIG=SQRT((SUMSQ-SUM**2/FLOAT(NRUNS))/FLOAT(NRUNS-1))
```

```
c
c...Calculate the Median CAPTIM
c
      I=(NRUNS+1)/2
      J=I+1
      IF(MOD(NRUNS,2).LT.1)THEN
        CMEDIAN=(CAPTIM(I)+CAPTIM(J))/2.0
      ELSE
        CMEDIAN=CAPTIM(I)
      ENDIF
c
c...Calculate CAPTIM Skewnees
c
      CSKEW=3.0*(CAMEAN-CMEDIAN)/CAPSIG
c
c...Exit
c
      RETURN
      END
```

Appendix F.15: Subroutine RRSTAT

```
      SUBROUTINE RRSTAT(NRUNS)
c**********************************************************************
c**                                                                  **
c**                       SUBROUTINE RRSTAT                          **
c**                                                                  **
c**    Calculates Minimum, Maximum, Median, Mean for IRR             **
c**                       and IRR Skewness                           **
c**                                                                  **
c**********************************************************************
c
      COMMON/TASKS/TTYPE(400),TNAME(400),TDURN(400,3),SFCODE(400),
     &TWK(400),TCOST(400,9),TWBS(400),PCCODE(400),IDURN(400)
      COMMON/DAYNOS/IESDNO(400),IESDOW(400),IDURN2(400),LFDNO(400),
     &LFDOW(400),IDURN3(400),ILDURN(500,2),ICRIT(400),ACRIT(400),
     &IPRDRN(1000),IEFDNO(400),IEFDOW(400),LSDNO(400),LSDOW(400)
      COMMON/STAT2/RRMIN,RRMAX,RRMEAN,RRSKEW,RRMEDN,RRSIG
      COMMON/PDCOST/DCOSTS(500,9),COSPAY(500,9),COSDLY(500,9),
     &IDDATE(500),MACDAY(50),TOTCOS(500),ACTOTC(500),ACCTPR(500),
     &CAPTIM(1000),RATERN(1000)

c
      CHARACTER*1 TTYPE
      CHARACTER*9 TNAME
      CHARACTER*12 TWBS
c
      REAL TDURN,TCOST
c
      INTEGER SFCODE,TWK,PCCODE
c
c...Initialise minimum, maximum, sum and sum square of IRR
c
      RRMIN=1.0E12
      RRMAX=0.0
      SUM=0.0
      SUMSQ=0.0
c
c...Calculate IRR minimum, maximum, sum and sum square
c
      DO10,I=1,NRUNS
        IF(RATERN(I).LT.RRMIN)RRMIN=RATERN(I)
        IF(RATERN(I).GT.RRMAX)RRMAX=RATERN(I)
        SUM=SUM+RATERN(I)
        SUMSQ=SUMSQ+RATERN(I)**2
 10     CONTINUE
c
c...Calculate Mean Internal Rate of Return and Standard deviation
c
      RRMEAN=SUM/FLOAT(NRUNS)
      RRSIG=SQRT((SUMSQ-SUM**2/FLOAT(NRUNS))/FLOAT(NRUNS-1))
```

```
c
c...Calculate the Median IRR
c
      I=(NRUNS+1)/2
      J=I+1
      IF(MOD(NRUNS,2).LT.1)THEN
        RRMEDN=(RATERN(I)+RATERN(J))/2.0
      ELSE
        RRMEDN=RATERN(I)
      ENDIF
c
c...Calculate IRR Skewnees
c
      RRSKEW=3.0*(RRMEAN-RRMEDN)/RRSIG
c
c...Exit
c
      RETURN
      END
```

Appendix F.16: Subroutine RRSORT

```
      SUBROUTINE RRSORT(NRUNS)
c***********************************************************************
c**                                                                   **
c**                       SUBROUTINE RRSORT                           **
c**                                                                   **
c**    Sorts Number of(NRUNS) Internal rate or return (RATERN),       **
c**                    (CAPTIM) into Rank order                       **
c**                                                                   **
c***********************************************************************
c
      COMMON/PDCOST/DCOSTS(500,9),COSPAY(500,9),COSDLY(500,9),
     &IDDATE(500),MACDAY(50),TOTCOS(500),ACTOTC(500),ACCTPR(500),
     &CAPTIM(1000),RATERN(1000)
c
c...Sort IRR and corresponding CAPTIM into rank order
c
      DO20,I=1,(NRUNS-1)
        DO10,J=1,(NRUNS-I)
          K=J+1
          IF(RATERN(J).LE.RATERN(K))GOTO10
          TEMP=RATERN(J)
          TEMP1=CAPTIM(J)
          RATERN(J)=RATERN(K)
          CAPTIM(J)=CAPTIM(K)
          RATERN(K)=TEMP
          CAPTIM(K)=TEMP1
  10      CONTINUE
  20    CONTINUE
c
c...Exit
c
      RETURN
      END
```

Appendix F.17: Subroutine FREQUA

```
      SUBROUTINE FREQUA(NRUNS,NBANDS)
c***********************************************************************
c**                                                                   **
c**                        SUBROUTINE FREQUA                          **
c**                                                                   **
c**       Calculates number of bands, Frequency and commulative       **
c**       in each band for the number of runs project durations       **
c**                                                                   **
c***********************************************************************
c
      COMMON/DAYNOS/IESDNO(400),IESDOW(400),IDURN2(400),LFDNO(400),
     &LFDOW(400),IDURN3(400),ILDURN(500,2),ICRIT(400),ACRIT(400),
     &IPRDRN(1000),IEFDNO(400),IEFDOW(400),LSDNO(400),LSDOW(400)
      COMMON/FREQU1/PCFREQ(100),PRBMPT(100),IPBAND(100),PFREQ(100),
     &RCPWPD(100)
      COMMON/STATS/MINPDU,MAXPDU,PRMEAN,PSKEW,MEDIAN,PROSIG


c
      IBW=10
c
c...Set minimum and maximum parameters
c
      AMIN=MINPDU
      AMAX=MAXPDU
c
c...Change the band width to real number
c
      BANDW=FLOAT(IBW)
c
c...Determine upper and lower distribution
c
        A1=AMIN-AMOD(AMIN,BANDW)-bandw
        A2=AMAX-AMOD(AMAX,BANDW)+BANDW
c
c...Calculate number of bands and zero each band frequency
c
      NBANDS=(IFIX(A2)-IFIX(A1))/IBW+1
      IPBAND(1)=IFIX(A1)
      DO40,I=2,NBANDS
        IPBAND(I)=IPBAND(I-1)+IBW
        PFREQ(I-1)=0.0
  40    CONTINUE
```

```
c
c...Calculate frequency for each band
c
      DO60,I=1,NRUNS
        DO50,J=1,(NBANDS-1)
          IF(IPRDRN(I).GT.IPBAND(J).AND.IPRDRN(I).LE.IPBAND(J+1))THEN
            PFREQ(J)=PFREQ(J)+1.0
            GOTO60
          ENDIF
 50       CONTINUE
 60     CONTINUE
c
c...Initialise commulative frequency
c
      DO70,I=1,(NBANDS-1)
        PCFREQ(I)=0.0
 70     CONTINUE
c
c...Calculate frequency (%)
c
      DO80,I=1,(NBANDS-1)
        PFREQ(I)=PFREQ(I)*100.0/FLOAT(NRUNS)
 80     CONTINUE
c
c...Calculate commulative frequency (%)
c
      PCFREQ(1)=PFREQ(1)
      DO90,I=2,(NBANDS-1)
        PCFREQ(I)=PCFREQ(I-1)+PFREQ(I)
 90     CONTINUE
c
c...Calculate bands midpoints
c
      DO95,I=1,NBANDS
        PRBMPT(I)=FLOAT(IPBAND(I))+BANDW/2.0
 95     CONTINUE
c
c...Calculate the Rate of Probability with Project Duration (f(x))
c
      A=0.39894228/PROSIG
      DO100,I=1,(NBANDS-1)
        RCPWPD(I)=(A*EXP(-(((PRBMPT(I)-PRMEAN)**2)/(2.0*PROSIG**2))))*10
     &0.00*BANDW
100     CONTINUE
c
c...Exit
c
      RETURN
      END
```

Appendix F.18: Subroutine FRQIRR

```
      SUBROUTINE FRQIRR(NRUNS,INBAND)
c*************************************************************************
c**                                                                     **
c**                        SUBROUTINE FRQIRR                            **
c**                                                                     **
c**      Calculates number of bands, Frequency and commulative          **
c**            in each band for the number of runs IRR                  **
c**                                                                     **
c*************************************************************************
c
      COMMON/PDCOST/DCOSTS(500,9),COSPAY(500,9),COSDLY(500,9),
     &IDDATE(500),MACDAY(50),TOTCOS(500),ACTOTC(500),ACCTPR(500),
     &CAPTIM(1000),RATERN(1000)
      COMMON/STAT2/RRMIN,RRMAX,RRMEAN,RRSKEW,RRMEDN,RRSIG
      COMMON/FREQU2/CFREQ(100),BANMPT(100),IBAND(100),FREQ(100),
     &ROPWRR(100)
c
      IBW=100
c
c...Initialise minimum and maximum parameters
c
      AMIN=RRMIN
      AMAX=RRMAX
c
c...Change the band width to real number
c
      BANDW=FLOAT(IBW)
c
c...Dtermine upper and lower distribution
c
      IF(AMIN.LE.0.0)THEN
        A1=AMIN-AMOD(AMIN,BANDW)-BANDW
      ELSE
        A1=AMIN-AMOD(AMIN,BANDW)
      ENDIF
      A2=AMAX-AMOD(AMAX,BANDW)+BANDW
c
c...Calculate number of bands and zero each band
c
      INBAND=(IFIX(A2)-IFIX(A1))/IBW+1
      IBAND(1)=IFIX(A1)
      DO40,I=2,INBAND
        IBAND(I)=IBAND(I-1)+IBW
        FREQ(I-1)=0.0
  40    CONTINUE
```

```
c
c...Calculate frequency in each band
c
      DO60,I=1,NRUNS
        DO50,J=1,(INBAND-1)
          IF(RATERN(I).GT.IBAND(J).AND.RATERN(I).LE.IBAND(J+1))THEN
            FREQ(J)=FREQ(J)+1.0
            GOTO60
          ENDIF
  50      CONTINUE
  60    CONTINUE
c
c...Initialise commulative frequency
c
      DO70,I=1,(INBAND-1)
        CFREQ(I)=0.0
  70    CONTINUE
c
c...Calculate frequency (%)
c
      DO80,I=1,(INBAND-1)
        FREQ(I)=FREQ(I)*100.0/FLOAT(NRUNS)
  80    CONTINUE
c
c...Calculate commulative fequency (%)
c
      CFREQ(1)=FREQ(1)
      DO90,I=2,(INBAND-1)
        CFREQ(I)=CFREQ(I-1)+FREQ(I)
  90    CONTINUE
c
c...Calculate bands midpoints
c
      DO95,I=1,INBAND
        BANMPT(I)=FLOAT(IBAND(I))+BANDW/2.0
  95    CONTINUE
c
c...Calculate the Rate of Probability with IRR (f(x))
c
      A=0.39894228/RRSIG
      DO100,I=1,(INBAND-1)
        ROPWRR(I)=(A*EXP(-(((BANMPT(I)-RRMEAN)**2)/(2.0*RRSIG**2))))*100
     &.00*BANDW
 100    CONTINUE
c
c...Exit
c
      RETURN
      END
```

Appendix F.19: Subroutine FRQCAP

```
      SUBROUTINE FRQCAP(NRUNS,ICNBAND)
c*********************************************************************
c**                                                                 **
c**                    SUBROUTINE FRQCAP                            **
c**                                                                 **
c**        Calculates number of bands, Frequency and commulative    **
c**              in each band for the number of runs CAPTIM         **
c**                                                                 **
c*********************************************************************
c
      COMMON/PDCOST/DCOSTS(500,9),COSPAY(500,9),COSDLY(500,9),
     &IDDATE(500),MACDAY(50),TOTCOS(500),ACTOTC(500),ACCTPR(500),
     &CAPTIM(1000),RATERN(1000)
      COMMON/FREQU3/CCFREQ(100),CABMPT(100),ICBAND(100),CAFREQ(100),
     &ROPWCAP(100)
      COMMON/STAT3/CAPMIN,CAPMAX,CAMEAN,CSKEW,CMEDIAN,CAPSIG

c
      IBW=1E6
c
c...Initialise minimum and maximum parameters
c
        AMIN=CAPMIN
        AMAX=CAPMAX
c
c...Change the band width to real number
c
      BANDW=FLOAT(IBW)
c
c...Dtermine upper and lower distribution
c
      A1=AMIN-AMOD(AMIN,BANDW)-BANDW
      A2=AMAX-AMOD(AMAX,BANDW)+BANDW
c
c...Calculate number of bands and zero each band
c
      ICNBAND=(IFIX(A2)-IFIX(A1))/IBW+1
      ICBAND(1)=IFIX(A1)
      DO40,I=2,ICNBAND
        ICBAND(I)=ICBAND(I-1)+IBW
        CAFREQ(I-1)=0.0
  40    CONTINUE
```

```
c
c...Calculate frequency in each band
c
      DO60,I=1,NRUNS
        DO50,J=1,(ICNBAND-1)
          IF(CAPTIM(I).GT.ICBAND(J).AND.CAPTIM(I).LE.ICBAND(J+1))THEN
            CAFREQ(J)=CAFREQ(J)+1.0
            GOTO60
          ENDIF
  50      CONTINUE
  60    CONTINUE
c
c...Initialise commulative frequency
c
      DO70,I=1,(ICNBAND-1)
        CCFREQ(I)=0.0
  70    CONTINUE
c
c...Calculate frequency (%)
c
      DO80,I=1,(ICNBAND-1)
        CAFREQ(I)=CAFREQ(I)*100.0/FLOAT(NRUNS)
  80    CONTINUE
c
c...Calculate cumulative fequency (%)
c
      CCFREQ(1)=CAFREQ(1)
      DO90,I=2,(ICNBAND-1)
        CCFREQ(I)=CCFREQ(I-1)+CAFREQ(I)
  90    CONTINUE
c
c...Calculate bands midpoints
c
      DO95,I=1,ICNBAND
        CABMPT(I)=FLOAT(ICBAND(I))+BANDW/2.0
  95    CONTINUE
c
c...Calculate the Rate of Probability with CAPTIM (f(x))
c
      A=0.39894228/CAPSIG
      DO100,I=1,(ICNBAND-1)
        ROPWCAP(I)=(A*EXP(-(((CABMPT(I)-CAMEAN)**2)/(2.0*CAPSIG**2))))*1
     &00.00*BANDW
 100    CONTINUE
c
c...Exit
c
      RETURN
      END
```

Appendix F.20: Function IDAY

```
      INTEGER FUNCTION IDAY(IDAYNO)
c***********************************************************
c**                                                       **
c**     FUNCTION RETURNS AN INTEGER DATE (DAY ONLY)       **
c**         FOR ANY GIVEN DAY NUMBER (DAYNO)              **
c**                                                       **
c***********************************************************
c
      PARAMETER(A=365.25,B=30.6001,C=122.1)
c
c...Calculate the approximate year
c
      IYEAR=INT((IDAYNO-C)/A)
c
c...Calculate the approximate month
c
      IMONTH=INT((IDAYNO-INT(A*IYEAR))/B)
c
c...Calculate the day
c
      IDAY=IDAYNO-INT(A*IYEAR)-INT(B*IMONTH)
c
c...Exit
c
      RETURN
      END
```

Appendix G: One Simulation Run Output Written to the File PERTRA3.RES

```
PROJECT START  DATE:  7/JUN/1993

PROJECT FINISH DATE: 12/APR/1994

Project Duration =    310  Days

CAPTIM = 5332846.00 £.DAY

Internal Rate of Return (IRR) = 562.22 %
```

Appendix H: Sample of the Multiple Runs Output Written to the File PERTRA3.RES

```
Number of runs =1000


TASK     TASK   OPTIMISTIC   MOST LIKELY  PESSIMISTIC   CRITICALITY
NAME     TYPE    DURATION     DURATION     DURATION       INDEX %


AFFWM     T       1.20          4.00         5.60           .00

AUSFWM    T       1.20          4.00         5.60           .00

CLEAR1    T        .60          2.00         2.80         51.90

CLEAR2    T       1.50          5.00         7.00        100.00

CPBPD14   T        .00         14.00          .00           .00

   .      .         .             .            .             .

   .      .         .             .            .             .

   .      .         .             .            .             .

CPFBLD    T        .30          1.00         1.40           .30

CPFCONC   T        .30          1.00         1.40           .30

CPFEX     T        .60          2.00         2.80           .30

CPFRF     T       4.50         15.00        21.00           .30

CPSCAFF   H        .00         46.00          .00           .00

   .      .         .             .            .             .

   .      .         .             .            .             .
```

Note: Criticality index shows the percentage to the number of runs that the task was a critical task.

Appendix H (Continue..)

1. PROJECT DURATIONS

Minimum project duration	=	305	DAYS
Maximum project duration	=	354	DAYS
Mean Project duration	=	332.70	DAYS
Median project duration	=	332.00	DAYS
Standard Deviation	=	8.85	DAYS

Skewness = .2367

MIDPOINT	FREQUENCY	CUM. FREQUENCY	f(x)
295.00	.00	.00	.01
305.00	1.10	1.10	.33
315.00	7.20	8.30	6.09
325.00	30.40	38.70	30.88
335.00	47.80	86.50	43.60
345.00	12.10	98.60	17.15
355.00	1.40	100.00	1.88

Median project duration is the duration lies in the middle of the set of project durations when arranged in oreder of magnitude.

f(x) is the rate of probability with project durations.

Appendix H (Continue..)

2. INTERNAL RATE OF RETURN (IRR)

Minimum IRR = 61.101 %

Maximum IRR = 1291.002 %

Mean IRR = 368.269 %

Median IRR = 289.961 %

IRR Standard Deviation = 236.04650 %

Skewness = .99525

MIDPOINT	FREQUENCY	CUM. FREQUENCY	f(x)
50.00	6.10	6.10	6.81
150.00	25.20	31.30	11.02
250.00	20.40	51.70	14.91
350.00	5.20	56.90	16.85
450.00	19.60	76.50	15.92
550.00	11.20	87.70	12.57
650.00	3.60	91.30	8.29
750.00	3.50	94.80	4.57
850.00	2.70	97.50	2.11
950.00	.00	97.50	.81
1050.00	1.70	99.20	.26
1150.00	.00	99.20	.07
1250.00	.80	100.00	.02

Appendix H (Continue..)

3. Negative CAPTIM

Minimum CAPTIM	=	3255526.00	£.DAY
Maximum CAPTIM	=	8720172.00	£.DAY
Mean CAPTIM	=	5385784.00	£.DAY
Median CAPTIM	=	5097613.00	£.DAY
CAPTIM Standard Deviation	=	1212286.00	£.DAY

Skewness = .71313

MIDPOINT	FREQUENCY	CUM. FREQUENCY	f(x)
2500000.00	.00	.00	1.94
3500000.00	13.00	13.00	9.81
4500000.00	31.90	44.90	25.20
5500000.00	26.20	71.10	32.76
6500000.00	16.10	87.20	21.57
7500000.00	9.90	97.10	7.19
8500000.00	2.90	100.00	1.21