Fuzzy and Multi-resolution Data Processing For Advanced Traffic and Travel Information

A thesis submitted in partial fulfilment of the
requirements of The Nottingham Trent University
for the degree of Doctor of Philosophy

Evgeny Agafonov

February 2003

Fuzzy and Multi-resolution Data Processing For Advanced Traffic and Travel Information

Abstract

Solutions to transportation problems have been extensively studied and developed over the last few decades. Although a significant progress has been made, a constant growth of the number of cars on roads cause new problems which cannot be dealt with by just controlling traffic signals. This implies that alternative ways of managing and controlling traffic need to be developed. A promising solution is to utilise combined information on current traffic and public transport situation and to employ multi-modal traffic optimisation. This solution requires the availability of very accurate real-time information. Information on travel time across an urban road network is one of the most important operational characteristics of the traffic system. It has been shown that existing algorithms for travel time estimation do not provide satisfactory accuracy and possess several disadvantages that render them inapplicable for estimation of travel time in volatile urban traffic. In this research, a problem of accurate travel time estimation is addressed and a novel low-level data processing methodology is presented.

Applications developed in the theoretical framework of real-time traffic systems require a suitable software environment. Many software packages have been developed to date for data processing, analysis and modelling. Unfortunately, most of them are either too general and are computationally burdensome, or are too specific and cannot be applied to solving a problem from a very specialised domain. As Transportation Research is a relatively small niche, the software development was traditionally accomplished on the project-to-project basis and, as a consequence, much time was spent on re-writing software for specific needs of a particular project. In this research, a general framework for processing traffic/travel data has been developed and its functionality has been demonstrated on a specific traffic/transportation problem.

Acknowledgments

I would like to express my gratitude to Prof. Andre Bargiela for his guidance and continual encouragement throughout this work. I am grateful to my second supervisor, Prof. David Applebaum for his extremely useful comments and suggestions on the theoretical part of this work.

Special thanks go to Dr Taha Osman and Dr. Evtim Peytchev for their help in the beginning and throughout the project, Dr Andrew Argile for his humour and other friends and colleagues at the School of Computing and Mathematics.

The financial support of the Nottingham Trent University is gratefully acknowledged.

Table of Content

Chapter 1. Introduction 1
1.1. Overview 1
1.2. Aims of the research 3
1.3. Original contribution 4
1.4. Organisation of the thesis 4

Chapter 2. Transportation Research and the Role of Data Processing . . . 7
2.1. Introduction 7
2.2. The transportation system 8
2.3. Modelling and simulation of transportation systems 10
2.3.1. Modelling paradigms: macro- and microscopic models 10
2.3.2. Macroscopic modelling: problems and results 12
2.3.2.1. Macroscopic traffic flow theory 13
2.3.2.2. Macroscopic area-wide transport system models 24
2.3.2.2.1. Trip generation 25
2.3.2.2.2. Trip distribution 26
2.3.2.2.3. Trip timing 26
2.3.2.2.4. Modal choice 27
2.3.2.2.5. Trip assignment 27
2.3.2.2.6. Origin-destination matrix 27
2.3.2.3. Conclusion on macroscopic modelling 29
2.3.3. Microscopic modelling: problems and solution 29
2.4. The role of data processing and analysis in traffic modelling 31

Chapter 3. Data and Information 33
3.1. Introduction 33
3.2. Data, Information and Research Process 34
3.2.1. Research process 34
3.2.2. Relationships between data and information 37
3.2.3. Data and information in transportation research 38

3.3. SCOOT UTMCS and Traffic Data 40
3.3.1. SCOOT UTMCS 40
3.3.2. SCOOT principles 41
3.3.3. SCOOT data 44
3.3.4. Retrieving SCOOT messages 47
3.4. Uncertain information 51
3.4.1. Introduction 51
3.4.2. Measures of uncertainty 52
3.5. Conclusions 54

Chapter 4. The Data Processing Environment 56
4.1. Introduction 56
4.2. Review of the software tools for data processing 58
4.2.1. The databases 58
4.2.2. Data processing tools for traffic projects 61
4.2.3. Other data processing, modelling and analysis software packages . 63
4.2.3.1. VisSim 63
4.2.3.2. MATLAB and Simulink 65
4.2.4. Conclusions 65
4.3. Requirements 66
4.3.1. Modularity 66
4.3.2. Extensibility 67
4.3.3. Efficiency 67
4.3.4. Concurrency 68
4.3.5. Reliability and fault tolerance 68
4.3.6. Universality 68
4.3.7. User-friendliness 69
4.4. Analysis and design 69
4.4.1. Use cases study 69
4.4.1.1. Edit a model 71
4.4.1.2. Run simulation 71
4.4.1.3. Retrieve the results 72

4.4.1.4. Edit library of modules 72
4.4.2. Functional modules 73
4.4.3. Inter-module communication and data typing 75
4.4.4. The system components 76
4.5. Implementation 77
4.5.1. Choice of platform 78
4.5.2. Package architecture 82
4.5.3. Package `edu.tntu.ism.traffic` 82
4.6. Graphical User Interface 85
4.7. Examples 87
4.7.1. General purpose modules 87
4.7.2. Task-specific modules 89
4.8. Evaluation of the efficiency 90
4.9. Conclusions 92

Chapter 5. Travel time estimation 93
5.1. Introduction 93
5.2. Travel time estimation problem 95
5.3. Review of the existing methodologies 96
5.4. Travel time estimation in urban links 103
5.4.1. The urban link model 104
5.4.2. The methodology of estimation of travel time with the lowest level of resolution 106
5.4.3. Simplified traffic model and mathematical justification of the travel time estimation methodology 117
5.4.4. Applicability of the simplified traffic model 146
5.5. Conclusions 147

Chapter 6. Empirical Study of Urban Traffic 148
6.1. Introduction 148
6.2. Pre-processing of SCOOT data 149
6.2.1. The core pre-processing software 149

6.2.2. Pre-processing of M14 and M19 messages 152
6.2.3. SCOOT message pre-processing module for the data processing environment 154
6.3. Use of the processing software in traffic analysis 155
6.4. Implementation of the travel time estimation procedure 156
6.5. Empirical study of travel time on urban links 158
6.6. Urban Travel Time components 167
6.7. Conclusion 170

Chapter 7. Conclusions and further research 171
7.1. Concluding remarks 171
7.2. Further research 172

References 175

Appendix A. DataTaker and DataGiver interfaces.
Appendix B. Example screens of the data processing environment.
Appendix C. Examples of implementation of modules for the DPE.

List of Figures

Figure 2.1. Typical speed-concentration relationship 15
Figure 2.2. Typical speed-flow curve 16
Figure 2.3. Flow-concentration relationship with operational regions A, B and C 16
Figure 3.1. Induction and deduction research processes 36
Figure 3.2. SCOOT detectors infrastructure 42
Figure 3.3. SCOOT detector data flow 42
Figure 3.4. Generation of M19 messages 46
Figure 3.5. Hierarchy of applications and forwarding of the terminal IO streams 48
Figure 3.6. Unix daemon processes 49
Figure 3.7. Architecture of data acquisition procedure 50
Figure 3.8. Architecture of data acquisition procedure with pseudo-terminals 51
Figure 4.1. The traffic database structure 59
Figure 4.2. Basic use cases and actors 70
Figure 4.3. 'Edit a model' use case diagram 71
Figure 4.4. 'Run simulation' use case diagram 71
Figure 4.5. 'Edit library of modules' use case diagram 72
Figure 4.6. Bean-like environment for functional modules 73
Figure 4.7. Modules use case diagram 73
Figure 4.8. Structure of a module wrapper 74
Figure 4.9. Illustration of the data typing system 76
Figure 4.10. The system's architecture 77
Figure 4.11. GUI classes architecture 86
Figure 5.1 Definition of travel time 95
Figure 5.2. Urban link model and travel time components 105
Figure 5.3. Merging upstream and shifted downstream time series 108
Figure 5.4. Extracting pairs of neighbouring events that come from different time series 108
Figure 5.5. Combinations of events ahead of the current event at k 109
Figure 5.6. Illustration of the concept of dependent and independent pairs 111
Figure 5.7. Architectures of pair of event arrival times extracted by the algorithm 113
Figure 5.8. Plot of the function (5.27) when there is only one event in each of the time series 114
Figure 5.9. The link model: a roundabout between two measuring points A and B allows vehicles to leave and join traffic flow 118
Figure 5.10. Exponentially distributed random variables S_N and S_M compete to be closer to time point t 120
Figure 5.11. Structure of the considered traffic processes 121
Figure 5.12. Illustration of possible outcomes for the next event after currently chosen one 125
Figure 5.13. Illustration of the resulting process and procedure of choosing the pair of events 126
Figure 5.14. Structure of probability distribution function of random variable ξ_1 135
Figure 5.15. Structure of probability distribution function of variables Δ^2, Δ^4 and Δ^{5a} 137
Figure 5.16. The events of types 1 and 3 have swapped their roles 141
Figure 5.17. A typical plot of function (5.78) 142
Figure 5.18. Schematic view of $y(t)$ 144

Figure 5.19. Single solution of equation (5.82)..... 144
Figure 6.1. Mansfield SCOOT-controlled region..... 148
Figure 6.2. The look of the SCOOT messages pre-processing module for the data processing environment..... 154
Figure 6.3. A fragment of a model that uses a file reader and SCOOT data pre-processor..... 154
Figure 6.4. Distribution of headways (taken from N60411M1)..... 155
Figure 6.5. Example of traffic profiling module output visualised by Plot module. Data used is taken for the N60411M1 detector for 16th January 1999, Saturday (red) and 19th January 1999, Tuesday (blue) and aggregated over 2 minutes intervals..... 156
Figure 6.6. Travel time estimation module..... 157
Figure 6.7. Cost functions calculated for different times of day. Link N60331F1-N60421B1, 19 February, Tuesday, 1999..... 161
Figure 6.8. Histogram of the time differences produced by the algorithm that correspond to the minimum of the cost function..... 162
Figure 6.9. Cost functions for the 'turn-right' type of a link: N60311K1 – N60321C1, 19th February (Tuesday), 1999. Time windows interval is 2 hours..... 164
Figure 6.10. Cost functions for the 'right-turn' type of a link: N60311K1 – N60321C1, 19th February (Tuesday), 1999. All day cost function..... 165
Figure 6.11. Plot of travel time estimates for the 'right-turn' type of a link: N60311K1 – N60321C1, 19th February (Tuesday), 1999..... 165
Figure 6.12. Cost functions of unrelated series. A link: N60311K1 – N60321C1, 19th February (Tuesday), 1999. Calculated over all day..... 166
Figure 6.13. Cost functions of link N60331F1-N60421B1 in a bigger scale. Several minimums point to the similarities due to cyclic nature of the series. The global minimum points to the "best match" between the series. Time shift varies from –4 minutes to 4 minutes. The global minimum is reached at time shift equal 24.25 seconds..... 167
Figure 6.14. The urban link model and travel time components..... 167
Figure 6.15. Histograms for T_{PC} travel time component estimates (link N60331F-N60421B1, four days data)..... 169
Figure B.1. A model implementing the discrete approximation of a step function.. B-5
Figure B.2. Simulation of the Lissajou figures..... B-6
Figure B.3. Three random variables and histograms demonstration..... B-7
Figure B.4 A model implementing the headways analysis procedure: visualisation is done via Histogram, Plot and Display..... B-8
Figure B.5 Class Library editor screen..... B-9
Figure B.6 SCOOT profiles demonstration: counter profile and occupancy profile B-10
Figure B.7 SCOOT profiles for different days: red – weekend (sat), blue – working day (tue) (aggregation over 2 minutes)..... B-11
Figure B.8 SCOOT travel time estimation routine is in progress..... B-12

List of Tables

Table 2.1. Principle traffic parameters..... 12
Table 4.1. Interfaces of `edu.tntu.ism.traffic` package..... 83
Table 4.2. Data object classes of `edu.tntu.ism.traffic` package..... 84
Table 4.3. Exception classes of `edu.tntu.ism.traffic` package..... 84
Table 4.4. System core classes..... 85
Table 4.5. General purpose modules: visualisation..... 88
Table 4.6. Results of performance tests for the first group of models..... 91
Table 4.7. Results of performance tests for SCOOT data processing model..... 91
Table 5.1. Types of pairs classified by the values of indicator functions..... 125
Table 5.2. Probabilities for different types of pairs to occur..... 130
Table 6.1. Parameters of the travel time estimation module..... 157
Table 6.2. Values of travel time point estimates..... 162

Chapter 1

Introduction

1.1 Overview

Initial purpose of the project was to investigate the feasibility of application of advances in information theory and soft computing to traffic and transportation systems. While the investigation of the transportation systems progressed on, it was discovered that there are a few niches lacking development, which are meant to be the basis of a fuzzy approach to traffic data analysis. Therefore these problems have been addressed in this research and the results can be considered to form a basis for further research leading to the application of the fuzzy techniques to traffic data processing and analysis.

Analysis of data has long been the basis of many applied areas of science. This is also true of new disciplines, such as Transportation Research, where data analysis provides the necessary foundation for modelling of transportation systems. It can be argued that knowledge about real-life systems is represented in the form of models that explain the system. In this sense modelling can be seen as an activity leading to improved understanding of the principles of the systems functionality. Deterministic and statistical modelling of systems were the only modelling techniques for centuries. However, it has long been known that not all real-world systems can be adequately modelled using deterministic or statistical approaches. These are systems with such number of interactive objects that analytical description of them becomes far too complex or systems, whose objects exhibit non-deterministic but not purely random character. In fact, most of real-world systems have such properties. Using traditional methods it is especially difficult to model systems with relatively large number of objects that themselves exhibit complex non-deterministic behaviour, such as most of the social systems. Transportation systems are such systems with complex interaction between society, technology and nature. The need for a solution was increasingly understood in the middle of the twentieth century and a number of new research directions were initiated. New modelling approaches questioned the applicability of the apparatus of pure mathematics to real-life problems and sought alternative ways. The important understanding of the crucial relationship

between the notions of information and complexity provided the motivation for the development of the new fields of science. The relationship between information and uncertainty has been known from the first works on information theory, which were carried out within the framework of probability theory. Further research has revealed that uncertainty, in fact, is a multi-dimensional concept [Klir, Folger, 1988], [Bargiela, 1998]. The fundamental modelling principles of minimum and maximum entropy have been generalised into the principles of uncertainty, the principles that allow a better understanding of systems' complexity and are the tools to handle it. Several independently developed areas based on biologically-inspired analogies have now joined in one broad area, which is now called Computational Intelligence [Pedrycz, 1998]. Fuzzy systems, neural networks, evolutionary algorithms are all branches of the new discipline. Many applied disciplines, being merely engineering areas, lack certain connections with the cutting edge of the science. Transportation research has traditionally been such a discipline. While much effort has been put to solving the transportation problems using traditional modelling techniques, a little has been done in application of soft computing paradigms, even though the limitations of the pure-mathematical approach have long been known.

In Transportation Research, data was normally used to validate mathematical models developed for certain aspects of the transportation systems, such as traffic flows, traffic assignment models, signalling control, etc. The above models served in transportation research for decades, but in recent years they also highlighted the fact that, in the context of continually increasing number of cars, the traffic management need to adopt a broader context of travel mode decisions rather than just traffic signal optimisation. This caused the need for development of alternative ways of managing traffic in order to utilise fully the available recourses of road and public transport infrastructure. The prominent solution to this seems to be in the development of new generation of intelligent information systems, which will be able to give a new scope for the improvement of the existing traffic control strategies as well as provide additional level of control by influencing the decision making process of the road users [Peytchev, 1999]. This new level of control, on the trip planning level, is a promising approach and is being extensively researched at the present time.

The noteworthy growth and development of the computing technology gives new possibilities in improved utilisation of road and public transport infrastructure and the optimisation of the common 'good' to all road users (urban sustainability). The new approach requires the availability of information on traffic situation at the most accurate level. Several well-known operational characteristics of a transportation system can be used as optimisation criteria. Travel time seems to be the most important among them as it is a natural measure of service provided by a road infrastructure. Although travel time, as an operational parameter of a traffic system, has been researched extensively for highway traffic, it is not well-researched in urban traffic systems. The problem of travel time estimation from traffic data is addressed in this research as a methodology that allows accurate estimation of travel times at the resolution of the data.

The development of computing hardware has been accompanied by a rapid growth in sophistication of computer software. Many software packages have been developed to date for data processing, analysis and modelling. Unfortunately, most of them are either too general and are difficult to apply to a specific problem, or are too specific and cannot be applied to solving a problem from a very specialised domain. As Transportation Research is a relatively small niche, the software development remains to be accomplished on the project-to-project basis and, as a consequence, much time is being spent on re-writing software for specific needs of a particular project. In this research, a general framework for processing traffic/travel data has been developed and its functionality has been demonstrated on a specific traffic/transportation problem.

1.2 Aims of the research

Following the above overview, the aims of this research project are given as follows.

- To investigate the problems of multi-resolution data processing and analysis within the area of Transportation Research
- To identify feasible solutions to traffic problems via utilising available traffic information
- To investigate informative content of traffic data provided by the SCOOT Urban Traffic Management and Control System (UTMCS)
- To develop a general framework for data pre-processing

- To develop necessary software modules, which will facilitate the theoretical research into the content of traffic data
- To develop new algorithms for multi-resolution data processing and information extraction

1.3 Original contribution

During the course of this research study, original contribution to the state-of-the-art of Transportation and Computing research have been made as follows.

- A problem of automation of real-time data retrieval from SCOOT has been solved by adapting standard services of the operating system and the IP in a novel way. The programme uses the standard Unix services, such as telnet or rlogin, in order to connect to SCOOT via a TCP/IP network, and pseudo-terminal capabilities of a Unix system to simulate terminal based connection to the SCOOT system. The software facilitates a range of research directions related to real-time traffic information systems.
- Link travel time, as an operational characteristic, has been researched in a context of urban transportation system. A novel methodology for accurate travel time estimation has been developed.
- A formal mathematical proof of the developed methodology for travel time estimation has been presented for a simplified traffic model using the theory of stochastic processes.
- A general software framework for data-processing, system modelling and simulation has been developed. The software has several distinguishing features that lift it up above similar commercial products. Openness, extensibility, modularity, high portability, visual pleasantness of the environment can greatly contribute to the efficiency of research since the researcher will have more time to concentrate on essential problems rather than technical detail of data pre-processing.

1.4 Organisation of the thesis

This thesis consists of 7 chapters which are as follows.

Chapter 1 is this chapter and gives an introduction to the scope of problems and the area in which this project has been carried out. It also gives a list of the aims of the research and the achieved original contribution to the current state-of-the-art.

Chapter 2 presents an introduction to Transportation Research from historical perspective. Several definitions, including the definition of transportation system, are given. Chapter 2 also reviews standard research methodologies used in transportation research area up to date. An insight to the complexity issues in the area is presented and the role of data processing in transportation research is outlined.

Chapter 3 investigates the notion of information and relationships between data, information and knowledge. Transportation systems are considered from the point of information flows and informative content at some abstract level and important operational characteristics of transportation systems are identified. Since traffic data used in this research has been provided by the SCOOT system, Chapter 3 presents an overview of functionality and features of this system. An informative content of traffic data provided by the SCOOT system is studied. Chapter 3 discusses technical issues of the connectivity to the SCOOT system and problems of automatic real-time data collection and develops a solution to these problems. Also, Chapter 3 touches the modern advances in the information theory and discusses the benefits of its application to traffic modelling.

Chapter 4 is devoted to the development of the data processing environment, the software suite which is aimed at simplifying data study. A brief review of existing software packages for data analysis and system modelling and simulation is presented and disadvantages of the packages are discussed. Then follows a presentation of the set of requirements, which the data processing environment should satisfy in order to improve the state of the area. The software system is designed using OOP methodology. The design follows the consideration of the available options of hardware and software platforms and concludes with choosing the Java platform. Using the Java technology, a data processing environment is developed and illustrated by a set of examples. Assessment of the efficiency of the software is given at the end of the chapter.

Chapter 5 is devoted to the development of a novel methodology for link travel time estimation in urban areas. The chapter presents a model of a typical urban link and discusses the differences between urban and highway links. A literature review shows that there is a lack of development of travel estimation algorithms for urban traffic systems. Following the literature review, the chapter develops a novel methodology which can be used to accurately estimate link travel time in urban areas. The methodology does not require aggregation of traffic data and thus preserves as much information in the data as possible. The methods of the methodology also have several advantages, such as robustness to under- and overcounting of vehicles, instability of occupancy profiles, etc., which are considered to have strong influence on the results of the previously developed methods. The methodology is proved formally using the theory of stochastic processes for a simplified traffic model.

Chapter 6 presents results of application of the developed software environment and travel time estimation methodology to a real-life transportation system. The presented results clearly show the feasibility of the developed methodology, which can be used to accurately estimate median of the link travel time and allows estimation of the probability distribution of the link travel time as random variable.

Chapter 7 contains final remarks and conclusions on the project and outlines further research direction, exploration of which is facilitated by the results of this research study.

Chapter 2

Transportation Research and the Role of Data Processing

2.1 Introduction

Transportation Research has emerged on the junction between many applied and theoretical disciplines in response to the tremendous growth of the number of vehicles in the 1950-1960. The advances in automobile industry and affordability of the vehicles at the time created what can now be characterised as transportation revolution. Since the existing road infrastructure has not been designed to accommodate such volumes of traffic, the situation has quickly created problems which have not been faced before: traffic jams, congestions, delays as well as environmental issues related to the growing fleet of cars - noise and pollution. Solution of the problems required systematic and thorough attack on the major problem faced. Gradually, integrated and balanced programmes of research were initiated in the developed countries such as the USA, the UK and Australia and then followed by many other countries. Since the first traffic and transportation research programmes have been carried out, the area of transportation research grew up into a field that involves co-operation between engineers, theoretical and applied mathematicians, sociologists and experts in many other fields. Transportation research now includes many sub-fields that study different types of transport, such as air or road transportation and areas that focus on certain aspects of transport-society interactions, such as economical issues, planning, environmental problems etc.

Generally there are three approaches to dealing with the traffic problems. The first approach involves creating new road infrastructure to increase the capacity of the existing road networks. This is the most expensive of all approaches as it requires involvement of land reconstruction procedures, engineer planning and significant labour expenses. It also has the strongest impact on environment which normally is not desirable. On the other hand it might be the only possible solution of the problem due to inapplicability or the limitations of the other techniques. Nevertheless in some cases building new roads might not be feasible due to specific nature of the environment, such as urban areas, and thus other techniques are required. One such technique is to manage existing infrastructure in a way that optimises its use. The idea behind this approach is to use available traffic

control systems but optimise this control in order to achieve desired effect (reduction of delays, minimisation of lengths of queues, etc.). This technique has evolved from a control performed by a human (normally a policeman controlled, monitored and optimised a single junction) to a complex multi-layered schemes of control and optimisation [Peytchev, 1999]. The third approach is obviously to decrease traffic volumes by affecting road users decisions on trip planning, namely to motivate the road users to use alternative modes of transport such as public transport or provide guidance and travel decision support. This third approach has not been widely researched and used in practice and therefore is a promising alternative to the first two.

Modern traffic planning, management and optimisation methodologies would normally employ all of the three approaches in order to solve traffic problems with the maximum efficiency and it is apparent that all of them require detailed knowledge of the behaviour of the system. It is also obvious that transportation itself and the influence it has on day-to-day life is very complex. It is unthinkable to give the precise answers to all transportation problems. Thus, transportation research, as well as many other applied disciplines, uses the systems approach in order to deal with the complexity of the interaction processes. The systems approach has been developed as a tool for dealing with complexity of real world and is based on decomposition of the problem into a hierarchy of independent sub-problems that are simpler to solve and then assemble the solved sub-components into the whole to obtain the solution of the original problem. This approach has shown to be very effective where direct solution is impossible due to the complexity of the underlying processes. In this chapter, several main transportation problems and the methodologies of their solution utilising the term 'transportation system' will be reviewed.

2.2 The transportation system

According to Hall [Hall, 1962], a system "is a set of objects with relationships between the objects and between their attributes". Transportation systems are enormously complex socio-technical, which has been acknowledged by many researchers [Kosonen 1999, Richmond 1998]. The systems involve interaction of many objects from single vehicles, pedestrians to communities, from geography to economics, and processes such

as technological processes, operations, planning, management, and many other processes. It is therefore clear that understanding of the behaviour of such systems is a difficult task and, due to its socially-stochastic nature, it cannot be accomplished with a precise deterministic result.

The systems approach essentially relies on modelling. Modelling provides the means by which a system can be described in a formal language and standard methods of analysis can then be applied to investigate the properties of the model. The results of the analysis are then said to correspond to the real systems subject to the adequateness of the model. Several models can be developed for the same system depending on what phenomena of the system are of interest or depending on precision with which the results are required.

In order to model a system, as it follows from Hall's [1962] definition, the system can normally be split down into a set of independent (or assumedly independent) objects and relationships between them that describe the way the objects interact with one another. It is also implicit in Hall's definition that objects themselves can be expressed as systems, which can be modelled as sub-objects and relationships between them thus creating a hierarchy of systems or sub-systems that constitute the initial system. A bigger depth of the hierarchy would normally increase the precision of the model but it would also increase the complexity of the model. To find the golden middle is one of the most important problem in modelling and is highly task-specific. The rules of trade-off between precision and complexity of a model are expressed as principles of maximum and minimum entropy, which will be studied in the chapter devoted to information.

From the systems theory perspective, the transportation research can be defined as research discipline that studies the transportation systems. There is no exact definition, though, what a transportation system is. Informally, a transportation system can be seen as the transfer of people or/and goods, using transport (vehicles, trains, aircrafts, etc.), between geographically separate places [Steenbrick, 1974]. The collective movement of transport between those places is called traffic. For the movement, the transport infrastructure is used. Further, the transport infrastructure can be divided into two components:

1. the fixed objects such as highways, secondary roads, city arterials, streets, railways and so on, with certain characteristics;

2. the organisational system necessary to ensure that the infrastructure is well used.

It is clear from this definition, that the objects of the transportation system are transportation infrastructure and the users of that infrastructure. It can also include other influencing factors and objects, such as pedestrians, weather, etc. The definition, although simple, does not reflect the complexity of the system. It should be noted that the system consists of a multitude of independent (although restricted by constraints and relationships) dynamic and static objects, in which the relationships themselves are tremendously complex multi-dimension stochastic functions. In addition, the transportation systems are part of higher level socio- and economical systems and experience strong influence from the other peer systems. It therefore becomes apparent that the transportation system cannot be modelled in a precise manner. In the following section standard transportation problems and methods used to solve the problems will be reviewed.

2.3 Modelling and Simulation of Transportation Systems

Modelling is the basic method by which systems are studied. Basically, a model is the formal description of a system, not necessarily precise, used for studying the system. Since transportation research is seen to be area of research, or discipline, that studies transportation systems, it has been the primary task of the discipline to build appropriate, or *adequate*, models of transportation systems. The form of models may vary widely depending on the purpose of modelling and phenomena to be modelled. Since modelling in transportation research forms the core of the discipline, the most important results will be reviewed in the following sections.

2.3.1 Modelling paradigms: macro- and microscopic models

Broadly speaking, there are two approaches to modelling systems. First approach has been to identify the objects of the system and try to establish the internal relationships and processes within the system in an explicit form, normally by mathematical equations. For example, a system of two material bodies with no external influence are governed by the Newtonian mechanics, which will form the core relationships between the bodies and describe the processes (motion) in an explicit deterministic form. Problems of such kind

have been pre-dominating in the history of science up to the nineteenth century. Another example, taken from [Klir et al, 1989], is motion of gas molecules in the system consisting of about 10^{23} them. Such system could not be studied in deterministic terms of Newtonian mechanics. Nevertheless, since the elements of the system, though numerous, exhibit similar stochastic character, average statistical properties and relationships between them can be derived and thus the behaviour of the system is adequately modelled. Description of such systems requires developed statistical and probabilistic apparatus. Therefore the second methodology of modelling systems is often referred to as the probabilistic approach.

According to [Weaver, 1948], two examples above describe two poles of systems complexity. The first pole, called *organised simplicity*, is represented by systems for which adequate models can be identified and the models contain a small number of variables. Two mass bodies is the typical example of such a system. The second pole, called *disorganised complexity*, is represented by systems with very large number of variables yet with high degree of randomness. Such systems are best modelled by the statistical mechanics and the second example is a typical system of that sort.

Unfortunately, most of real-life systems, as are transportation systems, do not fall into the two classes of systems described above. Such systems fall into the class of systems of *organised complexity*. As Klir states, methodology of treating such systems is undeveloped in the sense that neither analytical nor statistical methods are adequate and, therefore, new methodology needs to be developed.

Getting back to modelling paradigms, macroscopic modelling utilises the analytical and statistical methods in order to derive explicit description of the model in a form of relationships between variables that form the system. Although methodology is highly developed it represents a challenge to apply these methods to modelling transportation systems since such systems fall into neither the class of organised simplicity nor disorganised complexity. Nevertheless, first attempts to deal with transportation systems have been made, with certain degree of success, from the perspective of macroscopic models. Microscopic modelling is a very powerful method for dealing with systems, and it allows modelling of the systems of organised complexity with satisfactory accuracy. In contrast to the macroscopic modelling, microscopic models model individual objects of

the systems and relationships between them, which are normally simplified to a form of one-to-one relationships. For example, individual vehicles are objects of many microscopic models of traffic flow. A typical microscopic traffic flow model would consist of a number of individual dynamical objects (vehicles, pedestrians, cyclists, etc.) and road infrastructure modelled as a static graph. In order to obtain the insight on the dynamics of the modelled system, simulation is used. Simulation is essentially an iterative procedure that involves a large number of elementary calculations. This is the main reason why microscopic modelling started to be researched into with appearance of computers that are able to cope with such volumes of calculation. In the modern days, when very powerful computers are widely available, microscopic simulation becomes more and more attractive as it is normally a more precise reflection of real traffic processes. Even further, researchers are now looking into even lower level of models, the "nanoscopic" models, that take insight into individual vehicles and driver behaviour [Kosonen, 1999].

Microscopic and macroscopic models are both used in transportation research but the problems they are used for are normally different. Macroscopic models naturally provide the tool for large-scale planning problems, such as traffic assignment problem, finding equilibrium in the demand-supply problem and problems alike. Microscopic models can be used in modelling any scale transportation networks and provide information that can be used to optimise traffic at microscopic level (for instance, signals control). These two modelling approaches will be reviewed in detail in the following sections.

2.3.2 Macroscopic modelling: problems and results

Macroscopic modelling is involved in various levels of investigation and analysis of transportation systems. The main results formed several research areas devoted to solve particular problems or derivation of general relationships between standard parameters, recognised within transportation systems. Main parameters, or variables, normally involved in transportation modelling are summarised in Table 2.1.

Table 2.1. Principle traffic parameters

Traffic Parameter	Description
Volume	Traffic volume, vehicles or other objects (pedestrians, cyclists) per unit of time

Speed	Average speed of traffic flow
Delay	Delay (enforced delay)
Queue	Length of a queue (at a traffic light)
Concentration (density)	Vehicles per unit of length
Generation	Trips per person, activity unit by vehicle type
Parking supply	Number of spaces, costs per unit of time and per unit of space, maximum allowable duration
Parking demand	Occupancy, mean duration, turnover rate, parking volume
Safety	Number of accidents by severity class, number of traffic conflicts
Environmental factors	Noise, air pollution, vibration
Energy	Fuel consumed per unit of travel

2.3.2.1 Macroscopic traffic flow theory

Knowledge of the characteristics of the traffic stream provides the means for understanding traffic behaviour in many situations, such as overtaking, queuing, turning, crossing and merging. These characteristics, in conjunction with travel speed, traffic volumes and traffic density, are the major determinants of road capacity in a given traffic environment. A knowledge of traffic behaviour and parameters provides the means for quantitative formulation of warrants for traffic control systems, planning, policy making and other decision making processes.

As has been mentioned, the ultimate objective of transportation research is to provide the engineer with the methods and techniques of transportation systems optimisation. It is apparent therefore that before a system can be optimised, a rational description of its phenomena must be obtained. First steps towards these objectives have been made to

study traffic flow as the most apparent characteristic of a transportation system. Traffic flow was first studied using available, at the time traditional techniques such as analytical deterministic or stochastic modelling, which are now called macroscopic models.

Traffic flow is used to describe collective movement of discrete units across network infrastructure (in broad sense). The infrastructure is considered to be static and described in terms of topology, geometric features and other static parameters. In the transportation, the discrete units are normally road users such as vehicles, pedestrians, cyclists. Each unit is under control of a human operator and since the operators act independently, the collective traffic exhibits a random behaviour. Macroscopic models are not interested in modelling individual behaviour of single units but rather processes generated by their collective interaction. There are three main components of road systems: the driver, the vehicle and the environment (including road infrastructure). These three components interact with each other. Consequently the moving traffic flow has characteristics that are quite different to those of the individual elements. Several most important results will further be reviewed. The review is based on [Drew, 1968] and [Taylor et al, 1996].

Many results in macroscopic modelling make use of the fundamental equation of the traffic stream, that relate mean velocity (space speed) u, mean flow (rate of flow) q and mean density (concentration) k to each other and is expressed in the form

$$q=ku \tag{2.1}$$

If two of the above variables are known, the third is uniquely determined. Mean volume is the easiest parameter to measure. It can be determined by counting the vehicles $N(x,T)$ that pass a particular point x on the road over the time period T. Then q can be found as

$$q = \frac{N(x,T)}{T} \tag{2.2}$$

The other two parameters can be identified using either specific indirect methods or measured directly. For example, the density can be determined by observing the number of vehicles on the road section X at time t, for instance by taking an aerial photograph of the road at that time and counting the number of vehicles, say $N(X,t)$. Then

$$k = \frac{N(X,t)}{X} \tag{2.3}$$

Speed measurements can either be collected by traditional surveys, which result in giving a set of space speed measurements, or, if appropriate equipment is available, for instance double inductive loop detectors, spot speed measurements can be acquired in real time. Several methods have also been developed to estimate speed from the measurements taken from single loop detectors [Hall, 1988, 1989], [Pushkar et al, 1994], [Coifman, 2001], [Dailey, 1999].

Equation (2.1) has long been the basis of travel time estimation until recently it was shown to have several flaws. Since travel time as the parameter of a transportation system is of paramount importance, other algorithms that do not rely on speed estimates obtained from equation (2.1) have been and are still being developed. This research project addresses the problem of travel time estimation in Chapter 5. Currently available methods will be reviewed and a novel methodology of accurate travel time estimation using only single inductive loop measurements will be presented. The developed methodology has advantageous distinguishing features that make it a better choice for studying urban traffic.

The deterministic approach began with a suitable algebraic equation to explain the flow-concentration relationship. Greensheilds [1934] chose to plot speed u against concentration k for one lane of traffic. A typical plot of speed-concentration relationship is given on the following Figure 2.1.

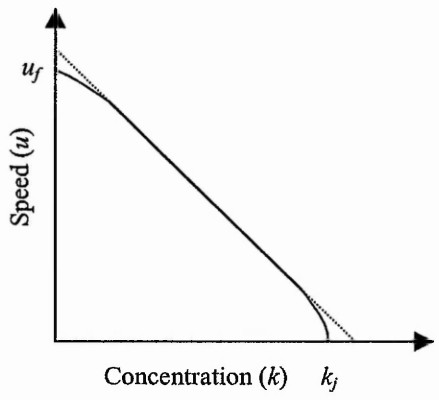


Figure 2.1. Typical speed-concentration relationship

Figure 2.2 shows the fundamental relationship between mean speed and traffic volume. The relationship defines a feasible region for traffic flow under different combinations of volume and speed.

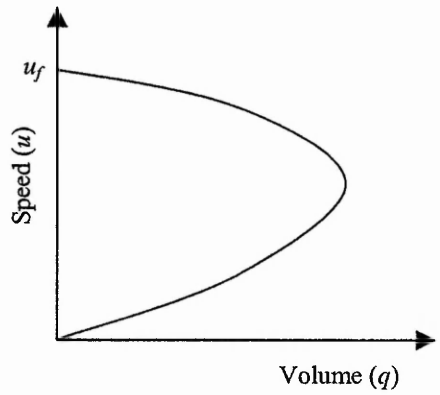


Figure 2.2. Typical speed-flow curve

The curve defines the limits to this region. The two arms of the curve represent different traffic flow conditions. The upper arm is for freely flowing traffic with a significant proportion of the vehicles travelling at their desired speeds. This proportion decreases as the nose of the envelope is approached. The lower arm represents congested flow conditions, when most of the traffic is strongly influenced by the vehicles in front of them. The speed-flow curve is widely used in determining the design capacity of a road, through the use of "levels of service".

The last characteristic curve of the fundamental relationship (2.1) is the volume-concentration relationship. Figure 2.3 shows a typical volume-concentration curve.

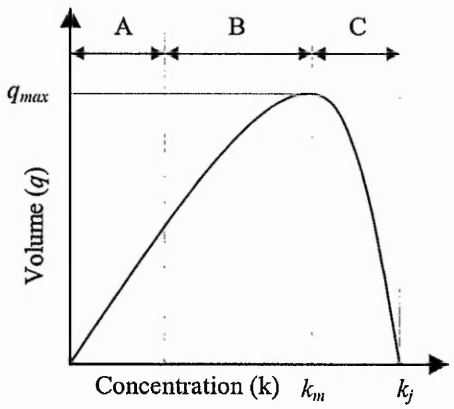


Figure 2.3. Flow-concentration relationship with operational regions A, B and C

The left arm of the curve represents free flow traffic condition, while the right arm is for congested flows. Thus density is the traffic parameter that can provide a full description of the state of a traffic stream. The curve of Figure 2.3 is split into three regions. These regions are for light, freely flowing traffic (region A), moderate but stable, freely flowing traffic (region B), and unstable, forced flows under congested conditions (region C). Region B is the one sought in traffic design since flows in this region achieve an optimum efficiency while utilising all available infrastructure.

First attempts to find the deterministic relationship that binds the above parameters have been made using curve fitting technique. The problem of fitting a curve to a set of points objectively is essentially the problem of estimating the true parameters of the curve. To do so the method of least squares is normally used. A common simplifying assumption is that the curve can be taken to be linear. This assumption is reasonable for most densities apart from the very small and very large densities.

Another method is the boundary-condition approach. The basis of this method is the differentiation of the fundamental equation (2.1) with respect to k. The result is

$$\frac{\mathrm{d}q}{\mathrm{d}k} = k\frac{\mathrm{d}u}{\mathrm{d}k} + u \tag{2.4}$$

Differentiation of the fundamental equation of state $q=f(k)$ with respect to k yields the wave velocity equation. Setting the wave velocity equation equal to zero yields the optimum concentration k_m, which is that value of k for which flow is a maximum. Since a general relationship exists between q and u, the k corresponding to $\mathrm{d}q/\mathrm{d}k=0$ can be set equal to k_m. Thus

$$0 = k_m\frac{\mathrm{d}u}{\mathrm{d}k} + u \tag{2.5}$$

Separating the variables and integrating both sides results in

$$\ln u = -\frac{k}{k_m} + C \tag{2.6}$$

Application of the boundary condition at k=0 and u=u_f enables one to equate C to ln u_f, yielding

$$k = k_m \ln \frac{u_f}{u} \tag{2.7}$$

Substitution (2.7) into the general equation (2.1) gives

$$q = uk_m \ln \frac{u_f}{u} \tag{2.8}$$

Edie [1961] obtained the same result by differentiating (2.1) with respect to u. Edie's equation (2.8) represents an effort to make the macroscopic q-u-k relations more accurate for traffic densities less then optimum. He suggested that the conventional forms still be used for densities greater than optimum, giving rise to a "discontinuous" form of the steady-state surface. This theory as modified for non-congested traffic may help to describe quantitatively the sudden change of state occurring in traffic stream going from a relatively free-flowing condition to a crawling stop-and-go condition and back again.

Several attempts have been made to explain the relationships between the three fundamental parameters using analogy between traffic flow and other physical phenomena. One such physical analogy applied to model traffic flow was the heat-flow analogy. Consider the problem of one-dimensional heat flow in a long, slender insulated rod satisfying the differential equation

$$a^2 \frac{\partial p}{\partial t} + \frac{\partial^2 p}{\partial x^2} = 0 \tag{2.9}$$

where p is temperature; x is distance; t is time and $a^2 = sk/c$, where c is coefficient of conductivity of material; s is specific heat and k is density.

The boundary conditions are statements of the initial conditions at the ends of the rod at any time and the initial conditions throughout the rod at time zero. What is usually desired is a description of the temperature at any time and place along the rod or, in other words, a solution of the differential equation in the form

$$p(x,t) = f(L,a,x,t) \tag{2.10}$$

On the other hand if a continuous record in time was kept of the temperature at various points along the rod one could solve for a and thus determine some property of the material such as its conductivity c.

In application of the heat-flow theory to traffic, a single traffic lane acts as a long, slender insulated rod (controlled access and no opportunity for lane change). If p in (2.9) is allowed to assume the role of some parameter related to such conventional traffic variable as speed, concentration and flow, the solution (2.10) provides the means for evaluating some property of the highway. All that needs to be done is measure the traffic characteristics along the stretch of highway defined by L. The heat equation (2.9) is, in study of partial differential equations of physics, called the equation of diffusion. It is satisfied by the concentration p of any substance which penetrates a porous solid by diffusion. Harr and Leonards [1961] postulated that the movements of traffic resulted from a motivating "pressure potential" analogous to p. In their solution the parameter was then eliminated (since it has no physical significance in the traffic phenomenon), leaving vehicular speed as a function of some properties of the highway and thus affording a means of rating various geometric features as a curve or grade. It is interesting to note that several variations of equation (2.9) have been studied in classical physics. For a rod in which heat is being generated at a constant rate, while the lateral surface is insulated, the heat equation takes the form

$$a^2 \frac{\partial p}{\partial t} + \frac{\partial^2 p}{\partial x^2} = c \tag{2.11}$$

where c is a positive constant. Reddy [1966] has suggested equation (2.11) as a basis for evaluation geometric features of the road rather than (2.9). In this manner the theory was

generalised to account for the effects of vehicles entering or leaving the section of roadways such as in the vicinity of entrance and exit ramps.

Another physical analogy that had been applied to modelling traffic flow is the fluid-flow analogy, or hydraulics. Consider the following equation of motion [after Greenberg, 1959], which expresses the acceleration of the traffic stream at a given place and time as

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{-c^2}{k}\frac{\partial k}{\partial x} \tag{2.12}$$

This equation states that a driver adjusts his velocity at any instance in accordance with the traffic conditions about him as expressed by $k^{-1}(\partial k/\partial x)$. If traffic is thinning out (negative $\partial k/\partial x$) the driver accelerates (positive $\mathrm{d}u/\mathrm{d}t$) and vice versa.

Equation (2.12) has the form of the equation of motion of a one-dimensional compressible fluid with a concentration k and a fluid velocity u. Fluid mechanics is based on the principle of conservation of mass. When this principle is stated mathematically as an equation, it is called the equation of continuity. Considering traffic flow as a conserved system, the change in the number of vehicles on a length of road $\mathrm{d}x$ in an interval of time $\mathrm{d}t$ must equal the difference between the number of vehicles entering the section at x and the number of vehicles leaving the section at $x+\mathrm{d}x$. If the number of vehicles on the length $\mathrm{d}x$ at time t is $k\mathrm{d}x$ and the number of vehicles entering in time $\mathrm{d}t$ at x is expressed as $q\mathrm{d}t$, the conservation of vehicles can be expressed analytically as

$$kdx - \left(k - \frac{\partial k}{\partial t}dt\right)dx = qdt - \left(q + \frac{\partial q}{\partial x}dx\right)dt \tag{2.13}$$

Making use of $q=ku$ yields the equation of continuity for traffic flow

$$\frac{\partial k}{\partial t} + \frac{\partial (ku)}{\partial x} = 0 \tag{2.14}$$

Taking the derivative of the product in the second term yields

$$\frac{\partial k}{\partial t} + u\frac{\partial k}{\partial x} + k\frac{\partial u}{\partial x} = 0 \tag{2.15}$$

It is well established in the theory of traffic flow that vehicular velocity varies inversely with the concentration of vehicles

$$u=f(k) \tag{2.16}$$

And as a consequence of (2.16)

$$\frac{\partial u}{\partial k} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial k} = \frac{du}{dk} = u' \tag{2.17}$$

Solving for $\partial u/\partial x$ from (2.17) and substituting in (2.15), one obtains the following equation of continuity of single-lane vehicular traffic flow

$$\frac{\partial k}{\partial t} + (u + ku')\frac{\partial k}{\partial x} = 0 \tag{2.18}$$

If a more generalised equation of motion is desired than (2.12), an exponent of proportionality n may be utilised:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -c^2 k^n \frac{\partial k}{\partial x} \tag{2.19}$$

Making similar calculation as for the (2.12), one obtains the generalised equation of motion

$$\frac{\partial k}{\partial t} + (u + \frac{c^2 k^n}{u'})\frac{\partial k}{\partial x} = 0 \tag{2.20}$$

The nontrivial solution of (2.18) and (2.20) is obtained by equating the quantities within the brackets

$$(u')^2 = c^2 k^{(n-1)} \tag{2.21}$$

Finally, because of the inverse relation between velocity and concentration,

$$u' = -ck^{(n-1)/2} \tag{2.22}$$

Greenberg [1959] has solved (2.22) for n=-1 obtaining

$$u = c \ln\left(\frac{k_j}{k}\right) \tag{2.23}$$

The solution of (2.22) for $n > -1$ is as follows:

$$u = \frac{-2c}{n+1} k^{(n+1)/2} + C_1 \tag{2.24}$$

where the constant of integration is to be evaluated by the boundary conditions inherent in the vehicular velocity-concentration relationship. Thus, since no movement is possible at jam concentration k_j,

$$C_1 = \frac{2c}{n+1} k_j^{(n+1)/2} \qquad n>-1 \tag{2.25}$$

and

$$u = \frac{2c}{n+1}\left[k_j^{(n+1)/2} - k^{(n+1)/2}\right] \quad n>-1 \tag{2.26}$$

Similarly, the implication exists that a driver is permitted his free speed, u_f only when there are no other vehicles on the highway (k=0). Therefore,

$$u_f = \frac{2c}{n+1} k_j^{(n+1)/2} \qquad n>-1 \tag{2.27}$$

and the constant of proportionality takes on the following physical significance:

$$c = \frac{(n+1)u_f}{2k_j^{(n+1)/2}} \quad n>-1 \tag{2.28}$$

Substitution of (2.28) in (2.26) yields the generalised equations of state

$$u = u_f\left[1-\left(\frac{k}{k_j}\right)^{(n+1)/2}\right] \qquad n>-1 \tag{2.29}$$

$$q = ku = ku_f\left[1-\left(\frac{k}{k_j}\right)^{(n+1)/2}\right] \quad n>-1 \tag{2.30}$$

Differentiation of (2.30) with respect to k equated to zero gives the optimum concentration k_m, which is that concentration yielding the maximum flow of vehicles:

$$\frac{dq}{dk} = \left[1-\frac{(n+3)}{2}\left(\frac{k}{k_j}\right)^{(n+1)/2}\right]u_f = 0$$

$$k_m = \left(\frac{n+3}{2}\right)^{-2/(n+1)} k_j \qquad n>-1 \tag{2.31}$$

Substituting (2.31) in (2.29), one obtains the optimum velocity

$$u_m = \frac{n+1}{n+3}u_f \qquad n>-1 \tag{2.32}$$

The maximum flow of vehicles of which the highway lane is capable (capacity) is obtained from the product of (2.31) and (2.32)

$$q_m = \left[\frac{n+1}{(1/2)^{2/(n+1)}(n+3)^{2/(n+1)+1}}\right]u_f k_j, \; n>-1 \tag{2.33}$$

Some special cases of (2.29) through (2.33) have proven to be of significance. Greenshields' [1934] linear model is obtainable by setting n=1, and Drew [1966] has discussed the case for n=0.

2.3.2.2 Macroscopic area-wide transport system models

While traffic flow theory studies traffic flows on individual roadways, area-wide theory studies transportation networks with many individual flows. Major changes to a traffic system, such as introduction of a new traffic management scheme, may have effects on traffic movement and traffic impacts at locations far away from the sites where the changes are implemented. The optimum passage of vehicles through a network may be sought by coordinating the traffic signal timings at successive intersections so that, for a given direction of flow, vehicles released through one intersection will arrive at the downstream intersection just as its signals turn to green for that traffic stream. For these reasons, amongst others, traffic analysts need to consider the area wide, or network, aspects of traffic flow and traffic demand. According to [Taylor et al, 1996], traffic network analysis is important for three main reasons:

1) urban traffic control must take a network perspective. A modern control system such as British SCOOT [Hunt et al, 1981], Australian SCATS [Sims et al, 1979], Spanish CARS [Grau and Barceló, 1992], attempts to coordinate signal timings at the intersection along routes through the network, so the traffic progression can be expedited. The specific objective is to allow the smooth passage of platoons of vehicles along major routes, with a minimum of stops, and the preservation of the platoons. The capacity of the intersection is a property of the intersection alone (although the cycle time might be set by capacity considerations at some nearby 'critical' intersection). Achieving the best utilisation of the available capacity requires coordination between intersections, so that platoons on intersecting roads arrive in sequence, to use the green times on the respective approaches;
2) changes to the signal timing and geometry for different movements at one intersection (i.e. changes to the transport infrastructure *supply*) may alter the pattern of traffic usage (the travel *demand*) of the intersection and the surrounding network;
3) urban transport systems are multi-modal, with consideration given to other modes of transport such as public transport, cyclists and pedestrians. In addition, the destination

choices of travellers may be made from amongst a set of alternatives. Therefore, it is important to know what people's travel needs and patterns are, how much usage will be made of all the modes, how destination choice will depend on the relative accessibility of alternative destinations, what impact congestion will have on the choice of mode and route and the timing of trips, and what interactions will be observed in different parts of the network.

Traffic network analysis aims to describe or forecast the distribution of traffic flows over a road network in a given time period or over a set of time periods. Modelling is usually undertaken in one of two main stages: in the description of an existing traffic system, where the aim would be to calibrate or evaluate a model, and in the synthesis of a future or alternative system (e.g. a proposed traffic management plan), where a calibrated model would be applied to test the performance or impact of the future or proposed system. Analysis and modelling may be applied in a series of steps, with the following five steps, and the basic question each seeks to answer, being [after Taylor et al, 1996]:

(1) trip generation – shall I travel?
(2) trip distribution – where shall I go?
(3) trip timing – when shall I travel?
(4) modal choice – how shall I travel?
(5) trip assignment – which way shall I go?

The sequence given by steps (1), (2), (4) and (5) forms the traditional 'four-step' procedure which has been applied in transport planning for many years. Step (3), regarding the timing of trips, was traditionally ignored but becomes important when flows over time of day are considered. This information is especially needed for Advanced Traveller Information Systems (ATIS), which are now studied extensively. Therefore, several problems naturally arise from the above steps and are described in the following sub-sections.

2.3.2.2.1 Trip generation

Trip generation seeks to define the numbers of trips which start (originate) at one site and finish at another (a trip is defined as the one way movement from an origin to a destination and there are two trip ends associated with each trip). A narrow but practical

definition of trip generation is useful in traffic impact analysis in that it is the measured level of traffic activity associated with a given site, development or land use. Trip production (P_i) is the number of trips which start (are produced) at a given site (i). Trip attraction (A_j) is the number of trips finishing (attracted to) at site j.

Trip productions from residential areas are usually modelled in terms of average trip rates, or by use of 'category analysis' tables, which classify the trip rates by household variables such as household size, income, car ownership, etc. Trip attractions are usually modelled by regression relationship, with the independent variables selected from the physical or economic characteristics of the land use.

2.3.2.2.2 Trip distribution

Trip distribution follows trip generation as it is concerned with forming linkage between the trip productions (P_i) and trip attractions (A_j) to form the trip matrix T_{ij}, the number of trips that will go from i to j. This matrix is called the Origin-Destination trip matrix or O-D matrix. The origin-destination matrix plays a fundamental role in studies of wide-area transport systems and wide-area traffic control. As a consequence, a considerable amount of interest has been attracted to the problem of estimation of O-D matrix. A review of the methods of obtaining O-D data is given in [Taylor et al., 1996] and mathematical treatment of O-D estimation and related traffic assignment problems is given in [Steenbrick, 1974]. Due to its importance, a review of methods for origin-destination matrix estimation will be given in a separate section.

2.3.2.2.3 Trip timing

Modern transport and traffic planning is concerned with the timing of journeys during the hours of day, in terms of the duration of periods of peak demand, the opportunities for spreading peaks over longer time periods to lessen congestion levels and the influences of congestion levels on an individual's ability to change the times at which trips are made through constraints imposed by non-travel activities undertaken by that person or by interaction between the members of a household. Trip timings is normally undertaken by considering the origin-destination matrix being a function of time, thus yielding different O-D matrices for each individual time t.

2.3.2.2.4 Modal choice

Modal choice analysis is used to predict the number of travellers who will use the different modes of transport available in the area. The most reliable models of modal choice attempt to predict the probability that a given person will choose a particular mode for a given journey. This probability will depend on the socio-economic characteristics of the individual, the characteristics of the available transport modes (e.g. fares, travel time, waiting time, reliability, comfort), and the proximity of the mode to the origin and destination of the individual's trip. The economic theory of maximum consumer utility [Hensher and Johnson, 1981] forms the basis of modern modal choice modelling.

2.3.2.2.5 Trip assignment

Trip assignment modelling is used to determine the route choice of drivers, and to build up the flows on each link in the network. So the question is simply to find a feasible solution to the network constraints. It is obvious that there are many solutions to this general assignment, and it is possible to apply further criteria for the assignment. There are three commonly used strategies for assignment modelling. These are Wardrop's principles [Wardrop, 1952] and Jewell's principle [Jewell, 1967].

2.3.2.2.6 Origin-destination matrix

As it has been mentioned in the previous sections, the origin-destination matrix plays an important role in describing properties of wide-area transportation system models. It has a wide spectre of applications including long-term planning, traffic systems design, real-time traffic signal control and Advanced Traveller Information Systems (ATIS). A variety of methods for O-D matrix estimation have been developed over the last years. The methods are normally utilised in one of the three ways:

1) O-D matrix is constructed by direct observation using traffic surveys or questionnaire surveys [Thompson, 1989]
2) O-D matrix is synthesised from observed flows on the link in the network; or
3) O-D matrix is modelled using, for instance, some 'gravity' model of trip distributions [Wilson, 1967], [Holm, 1976]

Construction of O-D matrices from direct observation is a rather expensive and laborious method, therefore more and more often methods that construct O-D matrices from observed traffic flows are being used. Since modern traffic control systems are being installed in more places, the cheapest data for O-D matrix construction is the traffic counters data that can be obtained from the most traffic control systems, such as SCOOT. Methods of O-D matrix synthesis from observed flows can also be divided to static methods and dynamic methods. Traditionally, static estimation methods were based on gravity model methods [Wilson, 1967], [Holm, 1976], which reduced estimation problem of O-D matrices to calibration problems with small number of unknown parameters. Van Zuzlen [et al., 1980] extended the gravity model to entropy-maximising technique. Another type of static estimation methods developed in recent decades is to include additional observation information (traffic counts, parking surveys, etc.) and then combine it with a priori knowledge of O-D matrices. Cascetta [1984] introduced generalised least squares approach to estimate O-D matrices from both survey data and traffic counts. Mather [1983] proposed to use the Bayesian approach to the O-D matrix estimation problem which results in functional forms equivalent to the generalised least squares method. Bell [1991] presented an algorithm to improve generalised least squares estimates by using inequality constraints.

Over the last two decades much interest has been given to the problem of the dynamic O-D estimation. Cremer and Keller [1987] and Nihan and Davis [1987] proposed recursive algorithms to estimate O-D matrices. Chang and Wu [1994] addressed a problem of time-varying O-D matrices estimation and developed a recursive estimation procedure for O-D estimation. Van der Zijp and Hamerslag [1996] developed an improved Kalman filter for O-D estimation problem. Spiess [1990] addressed a computational complexity of the O-D estimation problem and proposed a gradient-based algorithm that is able to cope with relatively large transport networks. More recently Florian and Chen [1995] proposed an iterative method of adjusting the O-D matrix that does not require the explicit route definition, which allows handing the exponential growth of the number of O-D pairs with the growth of the network. More recent developments were concerned with the O-D matrices estimation in the presence of uncertainty [Li, De Moor, 2002]

2.3.2.3 Conclusion on macroscopic modelling

The above review touches only a few angles of transportation research problems and methods. But it is clear that macroscopic modelling paradigm cannot give adequate answers to several questions. What is the behaviour of spot speeds of individual vehicles in traffic flow? How does headway depend on speed of the flow? How are congestions built up after an accident? There are many other questions which a macroscopic model may find difficult to answer. After digital computers became widely available, a new direction of modelling has started to be extensively developed, namely the microscopic modelling. The following section will give a brief review of the advances in microscopic modelling, the scope of problems it deals with and solutions it provides.

2.3.3 Microscopic modelling: problems and solution

In contrast to macroscopic modelling, whose purpose is to describe a system by a set of 'macroscopic' parameters, such as flow, density, speed of flow, etc., that are normally aggregated versions of some lower lever parameters, microscopic modelling is trying to model individual objects of the system and thus is capable of providing information on those lower level parameters.

In transportation, typical objects of microscopic models are vehicles, cyclists, pedestrians, etc. Road infrastructure is provided to the model as static data. Since real objects of a transportation system have very complex relationships, a microscopic model of the system normally takes into account only pair-wise, or one-to-one, relationships between the objects. Another distinguishing feature of microscopic models is that they are simulation models, that is the modelling is accomplished via simulation. Simulation is a process of updating the model's state with time and, since simulation is normally performed on a digital computer, is a cyclic numerical procedure that calculates the state of each object of the model according to its relationships (or interaction) with the other objects.

According to [Peytchev, 1999], functions of microscopic models can vary substantially. The simulation model can be a basic module for traffic queues and traffic density

predictions in the real-time traffic control or, alternatively, can work off-line and evaluate control strategies, which subsequently will be executed on the field according to the current simulation.

In transportation microscopic modelling, the principle of modelling of the one-to-one relationships between the objects led to the development of the car-following models that lie in the basis of many microscopic models of traffic flow. The first car-following models were proposed by [Reushel, 1950], [Pipes, 1953]. The basic idea of the first models was that the acceleration of a vehicle is directly proportional to the speed differential between it and the vehicle in front. This simple model had several flaws. The model does not include any reference to the spacing between the vehicles, but a driver can be expected to be more responsive to changes in speed if the leading vehicle is only ten meters in front, rather than 100 meters. Gipps [1981, 1986] proposed improved car-following model that closely mimics observed driver behaviour in congested traffic. The more recent models allowed greater accuracy in predicting vehicle movement at the microscopic level developing more complex models. From the point of adequacy of the car-following models it can be noted here that car-following model is not enough to define all possible movements of a vehicle, since in free flowing traffic the vehicle is not in the car-following state and thus has to be modelled differently. Kosonen [1999] developed a rule-based model that is capable of taking into account the possibility for a vehicle to fall into a very critical area where emergency type of actions can take place, the effects of traffic signals, speed limits, and other traffic signs.

Most of the microscopic traffic models are developed as a software tool that facilitates certain direction of research, traffic planning and other activities related to traffic. The tools can broadly be classified as being developed for 1) urban traffic 2) motorway traffic and 3) combined traffic models. A typical model for simulation of urban traffic is HUTSIM developed in Helsinki University of Technology, Finland [Kosonen, Kokkinen, 1992], [Kosonen, Niittymäki, 1995], [Kosonen, 1999]. One of the models for simulation of motorway traffic is AUTOBAHN developed by Benz Consult GmbH, Germany [Benz, 1994], [Benz, 1995], [Benz, 1996]. An example of combined (urban and motorway) traffic models is AIMSUN2 developed in Universitat Politècnica de Catalunya, Barcelona, Spain [Barcelo, Ferrer, 1994], [Barcelo et al, 1995], [Barcelo et al, 1996].

2.4 The role of data processing and analysis in traffic modelling

According to [Bargiela, 1998], systems modelling is broadly applicable in the context of two types of problem-solving activities: system inquiry and system design. In system inquiry, a model is developed for the purpose of understanding and controlling some aspects of reality, making adequate predictions or retrodictions and utilising these capabilities for various ends. In system design, a model is developed for the purpose of analysing the behaviour of the relevant objects before they are actually deployed so that desirable criteria are satisfied within given constraints. It is clear that in both cases data is the medium of information transmission. In system inquiry a model is built based upon some observed behaviour of the modelled system. The observations are normally made in form of measurements of the systems' parameters of interest. In system design, the results of modelling come in the form of data, which is then analysed in order to make decision regarding the models adequacy and relevance.

In the context of transportation system, it is clear that macroscopic modelling is of the system enquiry activity. It is therefore apparent that traffic data form the basis of all developed traffic models and all models have been developed by studying the data. Once a model has been developed, it cannot be applied to real-life problem without validation, which is again a process of comparing the results of modelling with real system's parameters, which are represented in data.

Microscopic modelling can be considered to be of the system design activity. Microscopic models represent systems that do not have to exist yet can be studied by the analysis of the results of modelling. Model validation is a necessary stage in microscopic modelling, which is essentially a process of tuning the model using real-system measurements in order to increase its adequacy to real-life systems. Once a model has been built, it provides information about the modelled system in a variety of forms of data.

Besides modelling, real-time traffic data is used to provide information on many spot-parameters of a real traffic system. Real time data processing, analysis and mining are

finding more and more applications in modern traffic information and advice systems, such as ATIS.

Concluding the above discussion, it is difficult to overestimate the importance of data processing and analysis in Transportation Research. Data processing and analysis is the basis of the most research activities within the area. Unfortunately, despite the apparent importance of data processing, this problem has been given a little attention by researchers. Most of projects carried out in the area of Transportation Research that dealt with traffic data employed individual, normally in-house made tools and methods of data processing that are suitable for a particular project only. Therefore a significant amount of time is spent over and over again for development the tools for particular projects. This problem has been addressed in this research project and as a consequence a flexible and convenient software tool has been developed.

Chapter 3
Data and Information

3.1 Introduction

The problem of understanding the relationships between the process of learning and basic information (data) has long been of a major interest. Aristotle studied the research process nearly three thousand years ago. He suggested that knowledge can be created by applying logic to observations in order to generate conclusions. The methods of logical analysis that he developed still form the basis of current scientific analysis, having been developed and given mathematical structure in the last hundred years. Aristotle realised that necessary data could come from observations of physical experiments and that the conclusions derived are based upon those data.

In the complex modern world, scientists are not the only people who collect data. Almost every aspect of modern social life associated with some or another sort of data being collected, analysed and stored. Several examples of activities that involve collecting and analysing data and the reasons behind the need for data are as follows [after Goodman, 1995-1996]

- Market research. Enormous amounts of data are collected by commercial organisations about buying intentions of customers; these surveys are widely used in the social and political arenas.
- Propaganda. Some data are collected to convince other people of the rightness of someone's view (as well as group's, organisation's, government's, etc.). Most propaganda that involves real data is based on processing and presenting raw data in a way that suits a particular message. This category could also include instances of scientific fraud in which data is falsified or misrepresented to convince a scientist's peer group of the correctness of one's work.
- Belief justification. Many people seek data that support the views they already hold.
- Decision support. In industry and government it is now expected that decisions be based on careful analysis of data. Relevant data about the area to be affected are

collected and collated; this would then form one of the key foundations of the decision to be made.

- Science. There are many fields of science where collecting and interpreting data is a critical part of the process of research.

In general, data collection is accomplished in order to *establish the parameters of a system*. Data represent measurements of the system that can be obtained by one way or another depending on the system. Data is normally regarded as the source of empirical knowledge but data itself does not constitute the knowledge. In order to extract knowledge from data, it first needs to be *transformed* into *information*. It is therefore clear that knowledge is based upon information which, in turn, is based upon data.

3.2 Data, Information and Research Process

3.2.1 Research processes

The purpose of any data collection is to gain knowledge about a system. The process of gaining new knowledge about system is conventionally called *research process*. This section will describe two known research processes as they have been developed throughout the history of science. The review is based on the lectures by [Goodman, 1995-1996].

Two fundamental research processes were identified by Aristotle nearly three thousand years ago. Aristotle argued that knowledge begins with experience and thus the first approach, called *induction*, is the necessary "leap" from unorganised experimental experience to conclusions and generalised knowledge about the system. Induction has been the prevalent research method until almost 19^{th} century and still forms the basis of scientific analysis. Induction proceeds through three main stages as follows.

1) Collection of relevant empirical data through observation. This process generates a body of facts that, over an extended period of time, accumulates to a point where generalisations can be made

2) Summarisation of collected data. The volume of data involved will require careful ordering and classification of the data.
3) Generalisation about the system from the collected data. Organised summaries of the raw data provide the basis for the interpretation process (*exploratory analysis*) that will produce the generalisations ("laws") that are being sought.

Although induction has been used for centuries, it has several serious flaws. Firstly, selection of relevant data is uncertain in a sense that different researchers may collect different data and thus the same phenomenon can be explained by different (based on different data) 'laws'. Secondly, the induction does not specify the selection of collection method ("experimental design"), e.g. in what circumstances, how and what data is to be collected. Also, there are no criteria for deciding exactly what technique should be used to summarise and analyse data. And most critically, the generalisations produced by induction are not really general in the proper sense; they are tied to the specific data that is available and the researcher who has created them. A different researcher can draw different conclusions from the same data and more data may lead to different conclusion.

The alternative process – deduction – has also been described by Aristotle, but it is applicable only when a discipline has reached a certain stage on its evolution. It requires an established body of strong generalisations (created inductively) upon which it can build. Deduction consists of four steps as follows.

1) Identification of the object to be investigated. Deduction starts with a body of 'theories' that are widely accepted to be 'true'. In this context 'true' means that there is substantial amount of evidence to support the theories and little or no evidence exists that contradicts them. The theories are always in some way incomplete, and the aim of research is to attempt to 'fill in' these gaps.
2) Formulation of a reasonable hypothesis and definition of how to test it. The hypothesis is a statement that is a) consistent with, and based upon, current theory but contains ideas that are not accepted as part of that theory; b) it is formulated in such a way that it can generate predictions that are capable of being shown to be either true or false, or in other words, the hypothesis must be *testable*; c) the statement leads to the formulation of an *experiment* that will generate data that can be used to test the hypothesis.

3) Carrying out an experiment. If experiments are problematic in induction, they are critical in deduction. The necessity to collect exactly the required data in order to test a hypothesis drives scientists to strive for the maximum level of control over their data collection. The success of an experiment is measured in terms of whether it collected the right amount of the right type of data to allow the hypothesis to be tested.
4) Testing the hypothesis. The testing of a hypothesis is performed by comparison of predictions based upon the hypothesis with the outcome of the experiment. If these predictions are found to be incorrect, the hypothesis is being rejected. On the contrary, if the predictions are found to have been correct, the hypothesis is said to have been verified.

Unlike induction, the deductive process has no specific role for subjectivity of conclusions ('inspiration'). It is accepted now that the analysis of experimental results and the testing of a hypothesis is a fundamentally 'mechanical' process. Therefore, two scientists should, in principle, generate equivalent results from the same experiment and interpretation in this context is limited to deciding whether or not specific prediction have been supported and the hypothesis is true or false. The induction and deduction research processes are summarised in Figure 3.1.

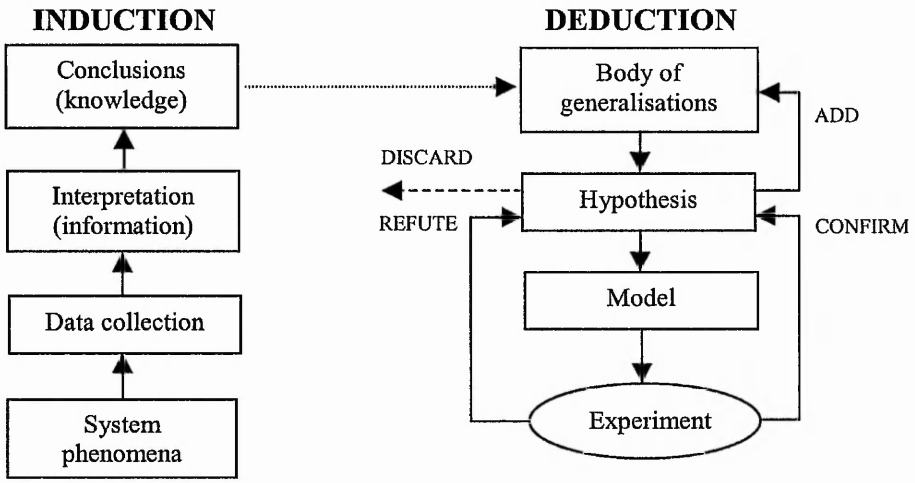


Figure 3.1. Induction and deduction research processes

Thus, a way from data to knowledge lies though these two stages. In first instance, when there is no knowledge available about the system of interest, induction is used to create

the necessary 'ground', or a body of facts. After that, the body of facts are extended and enriched using the deduction process. As a consequence, an established theory that describes the behaviour of the system of interest is created.

Once a theory is established, it contains enough 'confirmed' hypotheses, which are basically the models of phenomena of the system, and having a set of models that describe the system, it becomes possible to predict or influence (control) the processes of the real system. Thus, scientific knowledge can be defined as the understanding, in the form of models, of the phenomena of real-life systems.

3.2.2 Relationships between data and information

In the previous section, a review of fundamental research processes has been given. Let us get back to Figure 3.1. It can be observed from the induction diagram, what differs an entity called *data* from an entity called *information*. The hierarchy of the inductive research processes, emphasising the difference between the stages of the process, can be expressed as the following cliché

> "*Data is not information,*
> *Information is not knowledge,*
> *Knowledge is not understanding,*
> *Understanding is not wisdom*" -- Cliff Stoll & Gary Schubert

It has been widely recognised that concept of data and information are relative [Ahmed et al, 1999]. That is, data is a raw material for information, and information is obtained by processing data. Using an economical analogy, processing of data can be seen as addition of a 'value' to data. But information of a certain level can in turn be raw material for a higher level information, for which previous level information will be data, etc. Therefore, there is no absolute information or absolute data, these concept gain their common notion only in certain context. For example, measurements of the direction of wind, taken from individual points is raw data, based on which the direction of the air stream in bigger scale is information. Further, the information on directions of individual streams of air are data for the information about the cyclone, etc. From these arguments,

the term of *information granularity* can be defined as the level (relative) of the information within the hierarchy of informational entities that describe particular system.

The following section will establish a system within the area of transportation research, which will further investigate various level of information and data flows that are used in modelling the system. Particular informational granules (entities of information) of interest of this study will be identified and the importance of understanding of their behaviour (or underlying phenomena) emphasised.

3.2.3 Data and information in transportation research

In Chapter 2, the main directions in the road transport research has been discussed and several systems (or sub-systems) have been presented. Since transportation research is a relatively young discipline, it is possible to observe the 'classical' way of its development from inductive formation of the knowledge base to the iterative deductive development of certain areas of interest.

In this study, traffic systems are of main interest. The main object of such systems is traffic and this phenomenon of the transportation system is to be studied. Traffic has been of primary interest since transportation research has emerged as an individual scientific discipline. Historically, traffic analysis began with the studies of traffic data, which were collected in a variety of forms and techniques. Some classical traffic survey techniques are on-site observers filling in specifically prepared 'forms', probe vehicles cruising across the area of interest or video surveys. The body of facts eventually led to forming theories, or models, of traffic applied with various success (see Chapter 2).

Currently, the main source of traffic data are Traffic Control Systems, which allow automatic routine data collection. The data provided by the Traffic Control Systems is much cheaper than those collected using classical survey techniques. Another positive feature of automatically collected data is in its continuity (the collection is performed continuously without interrupts). Unfortunately, it is often of worse quality than survey data and also most difficult to process. Normally, Traffic Control Systems are designed to perform one kind of task and the measurement equipment (inductive loop detectors) are

designed for that kind of specific task. As a consequence, the data may be of a poor quality for other kinds of tasks.

In recent years, significant interest in the transportation research has been addressed to development of traffic information systems, or ATIS (Advanced Traveller Information System). This is caused mainly by the fact, that the traditional traffic control and management strategies have reached their limits and are unable to optimise further constantly increasing traffic and as a consequence, alternative ways of improving traffic situation are being sought. Providing current traffic information to the drivers is one such way and is thought to have a significant potential.

Summarising previous thoughts, let us identify the types of information that can affect drivers' decision regarding the planned journey. It is conventionally accepted that the most important information is the expected travel time from the driver's start to the destination. In this context travel time across an area is *information* since it cannot be measured directly but has to be *extracted* from data. Another example of traffic information is the availability of alternative modes of transport (e.g. public transport). In the latter case, public transport timetables can be considered to be data and the availability of the transport at a particular place and time is information, since the arrival of the transport can be affected by various factors of the current traffic situation and thus cannot be obtained from timetables but has to be *predicted* from the traffic data.

There are many important information entities that have been identified within the traffic systems. Investigating each of them requires substantial work. The problem of travel time estimation in urban areas has been one of the less investigated areas and therefore this research will focus on the development of the appropriate methodology for estimation of this important operational parameter of a traffic system. The knowledge of current travel time can be used in solving many practical problems, such as estimation of the origin-destination matrices, assessing uncertainty associated with estimation of a journey time and for Advanced Traveller Information systems. A review of the problems that can be solved once travel time is estimated with required precision and uncertainty associated is measured, will be put forward in the conclusion of this thesis as directions of further research.

3.3 SCOOT UTMCS and Traffic Data

In this project, a SCOOT system's data has been used to carry out experimental study. SCOOT system is a typical representative of modern generation Urban Traffic Control systems and is widely used in the UK and some other countries. A brief introduction into SCOOT principles and functionality will be given in the following section. A presentation of the procedure of retrieval of data from a SCOOT system follows the description of data, its format and features. Then a relationship between data and information will be discussed and the process of transformation one to the other will be studied in greater detail. The review is based on the following sources [Hunt et al, 1981] and [McDonald and Hounsell, 1991].

3.3.1 SCOOT UTMCS

SCOOT (Split Cycle and Offset Optimisation Technique) [Hunt et al, 1981] is a demand responsive urban traffic control system. Research into SCOOT was initiated by the Transport and Road Research Laboratory (TRRL) in 1973. In 1975, preliminary field studies were carried out in Glasgow by a team from the TRRL, Ferranti, GEC and Plessey with assistance from Strathclyde Regional Council. Upon successful completion of the research stage, development of the system began by the Department of Transport and Industry and the three traffic companies in 1976. TRRL continued research into SCOOT and in Spring 1979 carried out a full scale trial of the system in Glasgow. In this trial SCOOT was found to produce average journey times 6% lower than those produced by a set of newly-prepared fixed time TRANSYT plans [Hunt et al, 1981].

As part of the development project a SCOOT system, functionally equivalent to the Glasgow system, was installed in Coventry in 1979. A five week evaluation of the system was carried out in Spring 1980. In this evaluation SCOOT was found to produce average journey times 5.5% lower than those produced by a set of newly-devised fixed time plans [Hunt et al. 1981].

As a result of these successful evaluations, a decision was made to market SCOOT commercially. The first commercial system was installed in Maidstone in 1983. Since

then, the use of SCOOT has spread and the system is now in operation in over forty large towns and cities in the UK and overseas.

3.3.2 SCOOT principles

The basic philosophy of SCOOT is to obtain the optimum traffic signal settings, and that any changes must be implemented with minimum disturbance to traffic movement. In contrast to SCOOT's predecessor TRANSYT, which used pre-calculated fixed timing plans calculated over historical traffic profiles, SCOOT measures current traffic conditions in real time and constructs optimised plans for those conditions. In order to optimise traffic more efficiently, SCOOT attempts to predict particular traffic patterns into the future. To do so a traffic model is required and such a model forms the heart of a SCOOT system. The SCOOT traffic model produces predictions for a short time into the future to give an estimate of traffic demand for the next cycle of the traffic lights and further ahead for cycle time changes. In order to obtain information about traffic flow, a SCOOT system employs an infrastructure of single-loop inductive detectors that are embedded in the roadway. These detectors usually cover one or two lanes, and are usually sited 10 to 15 meters downstream of the upstream junction on a link. Figure 3.2 demonstrates the organisation of the SCOOT's detectors infrastructure. The SCOOT traffic model is based on data collected four times a second. Figure 3.3 shows the processing of detector data collected in 0.25 second intervals into the SCOOT model. This data is processed and used to modify the Cyclic Flow Profiles – one for each link, divided into four intervals. The traffic predicted to be crossing the downstream stop line in each interval is stored in these profiles.

The model also includes all the traffic stop lines. Every stop line is controlled by nodes and signal stages. The use of red-green information at the stop lines and traffic volumes in the profiles allows prediction of the queues that will form for a given stage length. The model contains a representation of traffic demand (cyclic flow profiles), stages, stops and delays.

Congestion is directly measured from the detector. If the detector is placed beyond the normal end of queue in the street it is never covered by stationary traffic, except the case

when congestion occurs. This enables the detector output to show congestion. If any four seconds data shows traffic standing on the detector for the whole interval then this is recorded. The number of intervals of congestion in any cycle is recorded.

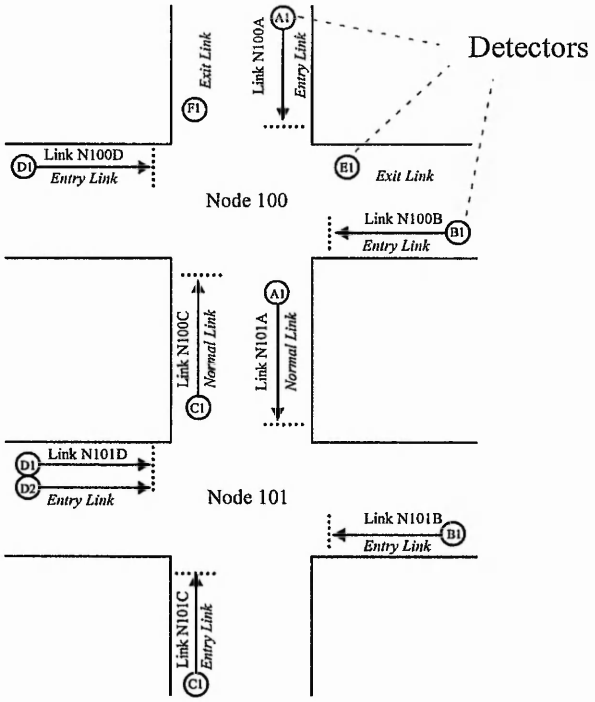


Figure 3.2. SCOOT detectors infrastructure

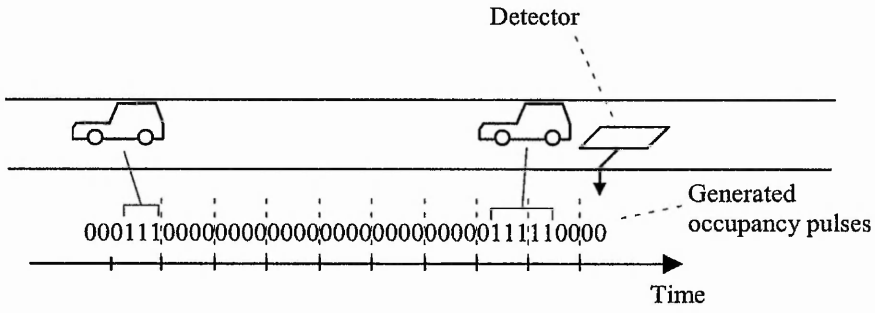


Figure 3.3. SCOOT detector data flow

The purpose of the traffic model is to predict the effect on street of different signal timings. Three separate optimisers use this ability to optimise splits, offsets and cycle time.

The split optimiser works on the traffic model just before each state change. It considers the effect of advancing, retarding or holding the stage change and the effect this has on the green duration. The test it uses is the degree of saturation of all links controlled by the node. The degree of saturation is defined as the ratio of the average flow to the maximum flow which can pass a stop line. The ratio of the demand flow to the maximum possible discharge flow in SCOOT terms is the ratio of the demand of the cyclic flow profile to the demand of the discharge rate (Saturation Occupancy) multiplied by the duration of the effective green time. The split optimiser will try to minimise the maximum degree of saturation on links approaching the node. If the average degree of saturation over a five minute period is greater than 90% then cycle time will increase to give more capacity at the critical node. This has been shown to minimise delay. If congestion is present on the approach to the node then this must be taken into account by the split optimiser. To enable this the proportion of the previous cycle that was congested is included with the degree of saturation used by the optimiser when making its decision. The congestion term will enable a congested link to tend to obtain more green time whatever the degree of saturation shown in the model, depending on the congestion importance factor set for this link.

The offset optimiser optimises the difference in stage start time between nodes. The cyclic flow profiles are used by the offset optimiser to predict the queue length throughout the cycle. Once a cycle the offset optimiser predicts the queue lengths for all the links upstream and downstream of a particular node. The effect of 'moving' the nodes 'time' forward or backward by a small amount is predicted for these links. As the time of arrival of traffic at the stop line is shown in the profiles, these predictions can be used to minimise the stops and delays in the mini-area. The congestion on a link is also used in the offset optimiser so that the congested link is given priority over links without congestion. The degree of priority is related to the degree of congestion.

The cycle optimiser usually runs every 5 minutes. At this time it will work out the degree of saturation at all the stop lines for each node. If any are above 90% saturated the minimum practical cycle time is increased by a small fixed step. If all are below 90% then the minimum practical cycle time is reduced by a small fixed step. The optimiser considers all cycle times from the highest minimum practical cycle time of the critical node up to the maximum region cycle time operating at that time. There is provision

within SCOOT for the optimiser to run twice as often if a trend of rising or falling flows has been established. To reduce delays at very lightly loaded junctions the cycle optimiser will double cycle time of these junctions if the delay is reduced in the network by this action. Because this can show reduction in delay if the cycle time is changed by large amounts, the cycle optimiser is the only optimiser that looks at the effect of large changes. However large changes in cycle time are very disruptive so SCOOT does not make the change in one step. The change made will be in small steps but the direction of movement will be chosen by reference to consideration of a larger change. The cycle time optimiser operates on a region of nodes that have progression between them.

3.3.3 SCOOT data

SCOOT functionality is not limited to producing only signal control. Many useful parameters and information regarding the current state of the system can be obtained in the form of *messages*. Access to SCOOT is normally accomplished via terminal either connected to the SCOOT computer or remotely via *rlogin* or *telnet* protocols. SCOOT starts printing messages to the requesting terminal as a result of a **MESS** command having been given.

There are 5 types of SCOOT messages

1) "M" messages are associated with model information
2) "C" messages are associated with cycle times
3) "S" messages are associated with splits
4) "O" messages are associated with offsets
5) "W" messages are associated with detectors and warnings

All SCOOT messages have common "header" part followed by individual message's content. The header contains the following fields

<DAY> <TIME> <TYPE> <LINK>[<DETECT>]

where

<DAY>	is day of week ('Mo', 'Tu', 'We', 'Th', 'Fr', 'Sa', 'Su')
<TIME>	is time of the event that corresponds to the message (HH:MM:SS)
<TYPE>	is the type of message (i.e., C01, W23, M14, ...)

<LINK> is the name of link the message corresponds to (e.g. N60331H)
<DETECT> is the detector number within the link (1, 2, rarely 3)

The <DETECT> field presents in the detector-related messages only, such as M19.

Two message types, M14 and M19, have been extensively used in this project and will be discussed in detail. For the information on format and content of other types of messages one can refer to the SCOOT User Guide.

Messages of M14 type carry information on the current state of SCOOT traffic model and control output for the period between current and previous message. M14 messages are generated every 4 seconds and have, in addition to the header, the following format

IVL <aa> **OCC** <bbbb> **LQ** <cccc> **BQ** <dddd> **EB** <eeee> **LIT** <ffff>

where

LINK	is a link name (or name of inductive loop)
IVL	is current interval within cycle
OCC	is occupancy value arriving at stop line (LPU/interval) (-1 if link is faulty)
LQ	is length of queue currently modelled (LPU)
BQ	is position of back of queue (LPU)
EB	exit blocked flag (1=block)
LIT	is four bits giving the state of signals in previous four seconds (1=green,0=red)

Here is given a fragment of SCOOT output flow:

```
                                                ...
Tu 09:27:16 M14  N60431E  IVL 0018  OCC   0  LQ -252  BQ   0  EB0  LIT 1111
Tu 09:27:16 M14  N60431F  IVL 0019  OCC  10  LQ   43  BQ  43  EB0  LIT 0000
Tu 09:27:16 M14  N60431G  IVL 0018  OCC   0  LQ -163  BQ   0  EB0  LIT 1111
Tu 09:27:16 M14  N60442B  IVL 0019  OCC   2  LQ   13  BQ  49  EB0  LIT 1111
Tu 09:27:16 M14  N60442C  IVL 0019  OCC  -1  LQ    0  BQ   0  EB0  LIT 0000
Tu 09:27:16 M14  N60442I  IVL 0202  OCC   0  LQ    0  BQ   0  EB0  LIT 1110
Tu 09:27:17 M14  N60311K  IVL 0219  OCC   0  LQ    0  BQ   0  EB0  LIT 0000
                                                ...
```

LPU stands for Link Profile Unit and is the SCOOT's traffic flow measurement unit. An extensive study of the accuracy of measurement of traffic flow has been carried out by several researchers [Carden et al, 1989], [Backer et al, 1991], [Evans, 1995], [Peytchev, 1999]. The main characteristic under analysis has been LPU-to-vehicle conversion factor (LPUCF) defined as a ratio of flow in LPU to the number of vehicles in the flow. It has been reported that LPUCF remains relatively stable over time and does not change much when calculated for peak and off-peak traffic (about 5% variability). It has been found

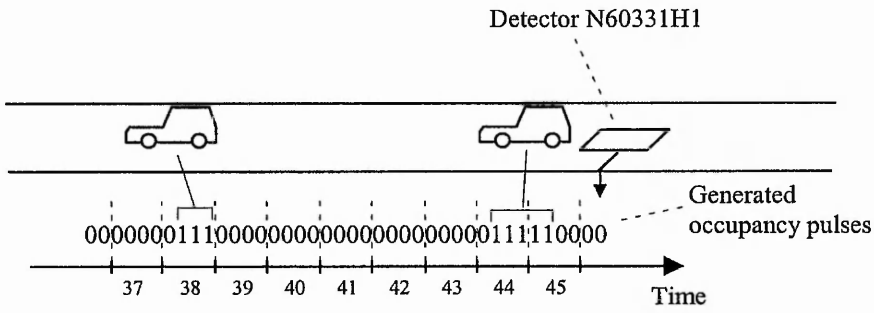
that the average LPUCF is 17 LPU per vehicle although LPU generated for an individual vehicle can change substantially [Peytchev, 1999]. Since accuracy of estimation of various traffic characteristics is the main concern of this study, a decision has been made not to use LPU measurements for the derivation of the characteristics. Another point to be mentioned is that the minimal time granularity of LPU measurements is 4 seconds, which means that traffic measured in LPUs is being averaged over this period of time and this is an additional source of uncertainty which will be inherited by the estimates based upon the LPU measurements.

Although LPU traffic measurements carried in M14 messages will not be used, a useful information in M14 is the signalling stages. Knowledge of signalling stages in urban traffic modelling and simulation is a prerequisite for a success of the study.

M19 messages represent detector occupancy measurements taken every quarter of second and encoded into a string of four bits. Thus, the frequency of M19 messages is once a second. M19-specific information has the following format:

DETECTOR STATE <VALUE>

The generation of M19 messages can be illustrated using previously given Figure 3.3. Considering Figure 3.3 and occupancy flow generated over the loop detector (let us name it detector 1 in link N60331H, or N60331H1), a set of M19 messages generated by SCOOT will be given below.




```
Sa 08:26:38  M19  N60331H1 DETECTOR STATE 0111
Sa 08:26:39  M19  N60331H1 DETECTOR STATE 0000
Sa 08:26:40  M19  N60331H1 DETECTOR STATE 0000
Sa 08:26:41  M19  N60331H1 DETECTOR STATE 0000
Sa 08:26:42  M19  N60331H1 DETECTOR STATE 0000
Sa 08:26:43  M19  N60331H1 DETECTOR STATE 0000
Sa 08:26:44  M19  N60331H1 DETECTOR STATE 0111
Sa 08:26:45  M19  N60331H1 DETECTOR STATE 1100
```

Figure 3.4. Generation of M19 messages

Messages M19 are the lowest level messages (no low pressing has been applied) and with the lowest time granularity. They allow detection of vehicle arrival events with accuracy ±¼ of a second.

M19 messages do not distinguish free travelling vehicles and vehicles that stop over the detector (during congested periods). A convention therefore has been made to consider all sequences of pulses longer than 8 to be related to either a stopped vehicle over the detector or that the detector is faulty. Consequently, since there are at most 8 pulses collected in any occupancy detection event and taking the average vehicle's length for 5 meters, the slowest speed is $5m/(8 \cdot 0.25s)=2.5m/s$ and the fastest speed $5m/(1 \cdot 0.25s)=20m/s$. Then, a series of all possible speed readings are {2.5, 2.8, 3.3, 4, 5, 6.6, 10, 20} *m/s*. This demonstrates the poor quality of spot-speed estimates that are based upon SCOOT single-loop detector measurements. In Chapter 5, a methodology of accurate travel time estimates based on single-loop measurements will be developed. The developed methodology can then be used for a much more accurate space speed estimation based only on single loop detectors' readings.

3.3.4 Retrieving SCOOT messages

Before data becomes available for processing it has to be retrieved from the SCOOT system. SCOOT system provides its users with an interactive interface, which is developed in a form of a command-line shell. As has been already mentioned in the previous section, a user accesses the interface either from a SCOOT terminal or remotely. Current version of SCOOT operating system supports IP protocol and standard remote-access services such as *telnet* or *rlogin*.

The interface needs interaction and does not provide automatic data retrieval process. In order to automate data retrieval process, a user-simulation software is required. Such software will send commands to the system and collect the output. Such software can then be used as an intermediate layer between SCOOT and DIME shared memory system [Argile et al, 1996] or send data to either a file or to an IP socket.

Several terminal simulation software are available that allow connecting to SCOOT system from a remote host, such as MS Window's standard HyperTerminal application. Unfortunately most of them do not have capability to control the data flows of the connection without presence of a human operator. The specificity of SCOOT operation (such as daily resets and initialisations) makes it necessary to monitor its state and provide necessary commands in order to keep data flow uninterrupted. The lack of scripting capabilities in most of the terminal simulation software led to a situation, when the researcher involved in data collection had to monitor the data collection process and change the terminal output files on a daily basis.

A solution, proposed in this study, is to use stream based communication applications, such as Unix's telnet or rlogin, and using piping capabilities of a Unix system to redirect the programs' standard input and output streams to another application, which would then monitor the incoming data and send commands according to the results of the monitoring. Not all terminal-based applications can be controlled via standard input and output streams that are offered by the Unix piping system. Some applications use so called addressed cursors and standard terminal operations such as control characters, colours etc. A set of capabilities that are supported by a terminal device is called a terminal protocol. Examples of such terminal protocols are ANSI, VT100 and other terminal protocols. In Unix, a terminal protocol support is inherited from the underlying device driver (device such as a video card, a dumb terminal connected via serial port, etc.) and then passed to all applications that use terminal capabilities. It can be said, that terminal device input and output channels are being forwarded from the system to the first terminal application and then throughout the hierarchy of applications run one from another (see Figure 3.5).

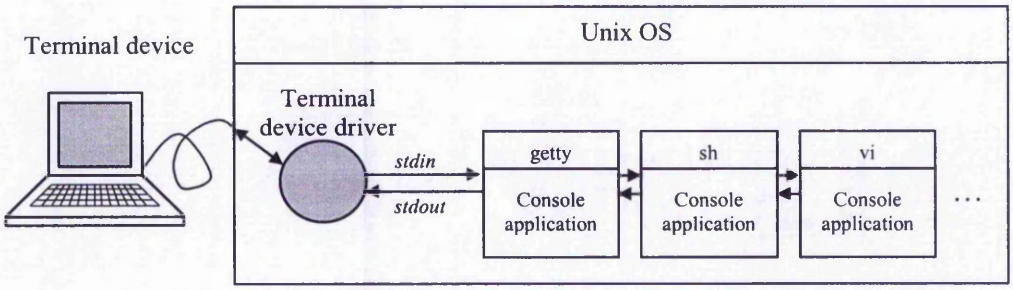


Figure 3.5. Hierarchy of applications and forwarding of the terminal IO streams

Unix *daemons* are processes that are run in the background mode to perform a certain task and do not require interactive operation with the user. Therefore daemons do not have an associated terminal device and have their standard input or output streams closed or redirected to files (see Figure 3.6).

The application that is supposed to use rlogin or telnet in order to connect to a remote SCOOT system, further referred to as 'controller', is by its nature a daemon process. It does not need a terminal device of its own unless it is controlled directly from a console (which is not a good thing to do anyway).

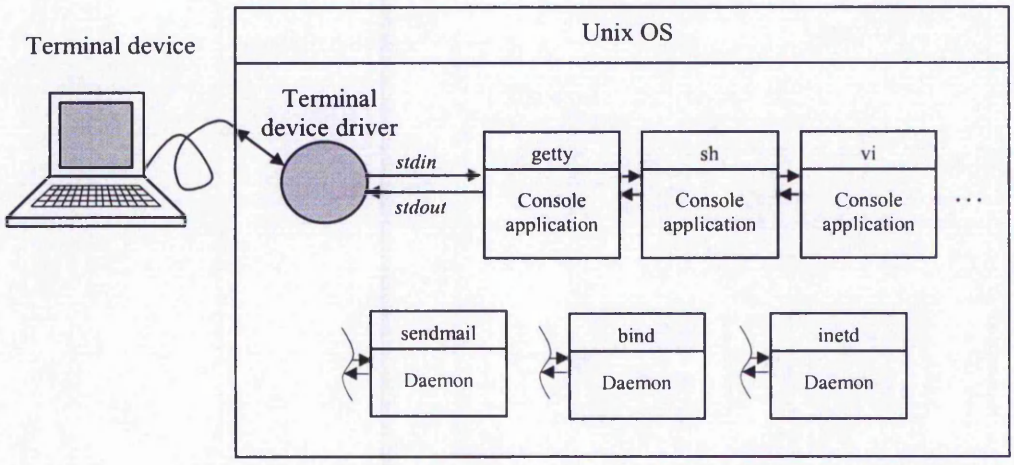


Figure 3.6. Unix daemon processes

Since the controller is a daemon and does not have a terminal device associated with it, the pipes created to communicate with the intermediate application are ordinary streams without support of the terminal capabilities (see Figure 3.7).

Many terminal I/O based application use advanced terminal capabilities which require support for underlying hardware terminal functions. If the application process has been started with the standard input and output streams initialised with ordinary streams (such as file streams, sockets or pipes) the application may cease to work due to errors while initialising terminal. SCOOT interactive interface is such an application. Although the absence of hardware terminal functions is not a fatal error for most applications (but of

course, most applications become useless because of the chaos in its output), SCOOT interface considers it a fatal error and terminates immediately. Therefore, although a scheme of data acquisition with controller as shown on Figure 3.7 works for standard input/output based applications (such as *sh*, *awk*, *cat*, etc.) it does not work for SCOOT interface.

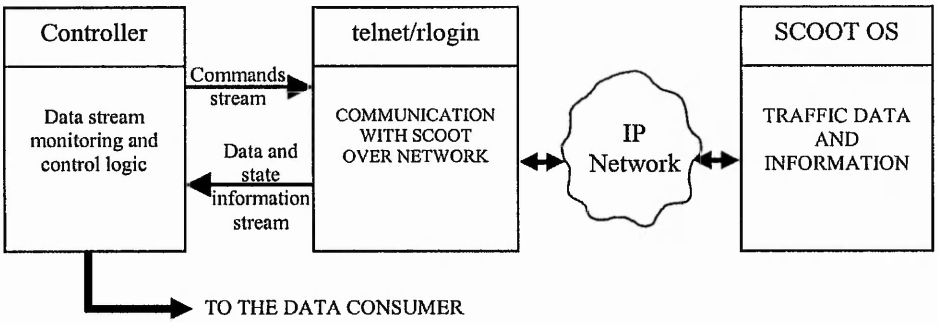


Figure 3.7. Architecture of data acquisition procedure

A solution for this problem has long been implemented in all Unix systems and is called "pseudo-terminals". Pseudo-terminals are bi-directional communication channels that can be used to connect two processes or a process group to another process. What pseudo-terminals provide is a lot like two pipes, but with a support of hardware terminal interface. The hardware terminal support is implemented as "dummy", that is it just an interface with no real functionality since no underlying hardware is available. Nevertheless, this implementation is sufficient for the terminal based application to function properly. Figure 3.8 demonstrates the application of pseudo-terminals to the data acquisition procedure.

From Figure 3.8 it is clear that for the 'telnet/rlogin' part of the scheme, the standard input and output look exactly as if they were associated with a hardware terminal and, thus, an application on a remote location (SCOOT interface) can use standard hardware terminal functions and therefore operate properly.

A software product realising the above scheme has been designed and implemented in the C language on a Unix platform (Solaris). It has then been successfully used to simulate the interactions needed to operate SCOOT in a way that is required for a flawless and

non-intermittent data flow. The controller application has been used to retrieve real-time SCOOT data into DIME system [Argile et al, 1996], which then allows multiple remote application to access the data in a consistent and efficient way.

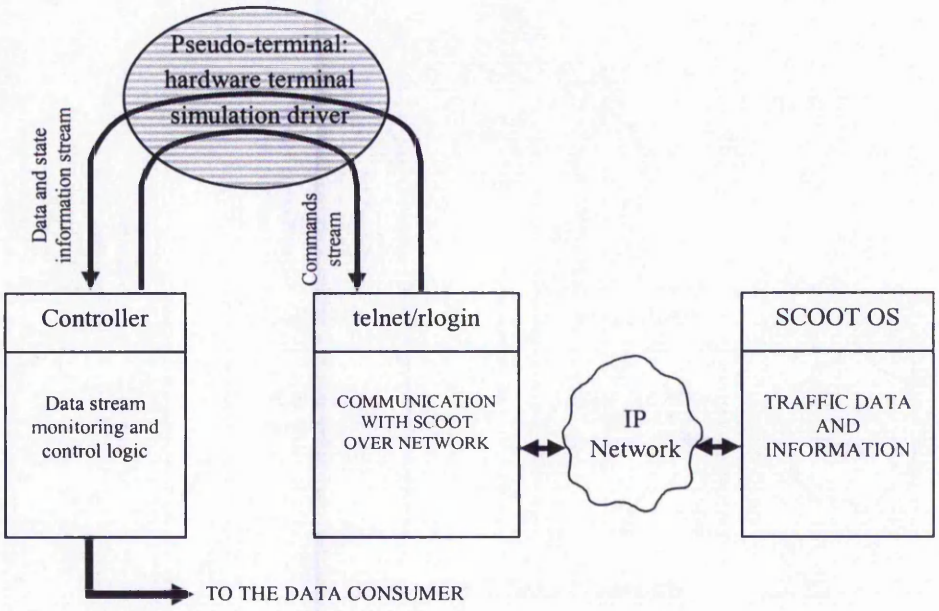


Figure 3.8. Architecture of data acquisition procedure with pseudo-terminals

3.4 Uncertain information

3.4.1 Introduction

It is now well recognised that the uncertainty associated with an aspect of a physical reality is a reflection of the lack of precise information about this reality. That is to say the more information that is known about a system, the less uncertainty is associated with our knowledge of the system. It has been shown [Bargiela, 1998] that most of models of real-life process are affected by uncertainty that appeared to be a multi-dimensional concept. Depending on the mathematical framework employed, uncertainty may be distinct in terms of one or more of the five types: entropy, dissonance, confusion, nonspecificity and fuzziness.

A deterministic model that captures precisely all aspects of a system is certain. That is to say it can answer any question about the system with 100% accuracy. However, real-life systems are of a tremendous complexity in deterministic terms and therefore can hardly, if at all, be described by a deterministic model. The complexity of real-life system makes deterministic approach impossible to be applied for real-life system. On the other hand, a system can be described by an uncertain model where some insignificant information has not been included into the model, and further by a model that does not contain any information about the system (let us call such a model an empty model). An empty model unlike a deterministic one is fully uncertain and on the other hand as simple as empty set from the set theory.

The two extreme poles show the relationship between informational contents and model complexity it causes. Models of real-life systems are neither of these poles. Firstly, it is hardly possible to obtain all the information about a system and, secondly, the purpose of modelling is to extract relevant information about a system and avoid unimportant details. An important problem to find a compromise between complexity and uncertainty has been of main interest of the systems theory. The solution has been found in application of the principles of minimum and maximum entropy [Jaynes, 1979], [Christensen, 1981], [Williams, 1980], [Klir and Folger, 1988] in the framework of the probabilistic information theory.

Nowadays a concept of uncertainty has been broadened [Bargiela, 1998], [Klir and Wierman, 1999], [Wang and Klir, 1992], [Shafer, 1976] and is considered as a multi-dimension entity. The fundamental principles of minimum and maximum entropy have been generalised and became the principles of minimum and maximum of uncertainty. This causes the need for developing new modelling methodologies which take the multidimensional nature of uncertainty into account.

3.4.2 Measures of uncertainty

Probabilistic uncertainty has long been the only known type of uncertainty. Since it represents an intuitive concept of 'chance' it has been found in the works of Pascal where he considered uncertainty associated with placing bets in gambling games. The

presence of probabilistic uncertainty in traffic modelling can be found in the example of travel time estimation. Travel time depends on a number of factors, which can naturally be assumed as being random variables. Since it is possible to perform many experiments with travel time estimation, the statistics on the variable can be collected and then using well-developed statistical apparatus, analysed. In such case, a probability distribution can be estimated by frequency distribution. The probabilistic analysis of the travel time estimation procedure will be given in Chapter 5 and empirical study of travel time distributions in Chapter 6.

Two other types of uncertainty have been discovered by [Shafer, 1976] in which the additivity of the probability measure was questioned. The two measures were called plausibility and belief and measure dissonance and confusion types of uncertainty accordingly [Klir, Folger, 1989]. The presence of dissonance and confusion uncertainties can be found in the procedures that involve interaction of concurrent bodies of evidence. For example, a procedure of turning movements estimation is affected by those kinds of uncertainty and therefore any model based on the turning movements inherits this uncertainty. A discussion on turning movements estimation problem will be given in Chapter 7.

A type of uncertainty called *nonspecificity* is complementary to probability and puts an emphasis on the process of selection of an alternative from the set of all available alternatives. This ambiguity is totally resolved when one of the alternatives is selected [Bargiela, 1998]. A procedure of establishing the correspondence between origin and destination of particular sets of events experiences such an ambiguity. Therefore a travel time estimation procedure, turning movements estimation procedure and others are affected by this sort of uncertainty.

Another type of uncertainty, vagueness, is associated mainly with objects that have "fuzzy" boundaries or it is difficult to draw precise distinction between such objects. The theory of fuzziness is now well developed and includes a number of disciplines such as fuzzy sets, possibility theory, fuzzy systems, fuzzy calculus and so on [Zadeh, 1965], [Dubois and Pride, 1980], [Dubois and Pride, 1985], [Rosenfield, 1971], [Pedrycz, 1983]. It has been recognised that the fuzzy approach can be adequate even in crisp systems where precise information is not required or cannot be obtained. In the case of traffic

modelling, one deals with measurements, which are by their nature crisp numbers. Nevertheless, there are many other factors affecting traffic that cannot be measured precisely but can be formulated in fuzzy terms and included into a model and, as a consequence, increase its credibility.

Also a model of traffic can be developed on the basis of fuzzy numbers and fuzzy relationships, where systems' parameters are fuzzy numbers and relationships define their interaction and dependencies. For example, the estimated travel time can be seen as a fuzzy number whose membership function is the frequency distribution. This idea will be investigated further in Chapter 6.

3.5 Conclusions

In this chapter, a relationship between data and information within the framework of modern research process theory has been established. Transportation research, as a relatively new area of research, has been shown to evolve through the presented stages of theory evolution.

In this research, data collected from a SCOOT system has been used to carry out experimental study as described in the deduction research process. A brief review on the SCOOT functionality and parameters have been given to provide the background on the type of data available from the system. Data formats have been given and several types of SCOOT messages, extensively used in this research, are described in greater detail.

It has been widely recognised that uncertainty associated with knowledge about a system is a multidimensional concept. Modern theory of information has identified several dimensions of uncertainty and outlined directions in which current knowledge can be improved by understanding the associated uncertainty and measuring it in an appropriate way. After uncertainty has been identified and measured it will become possible to master uncertainty in a way that is beneficial in some sense.

Transportation research, as a merely engineering area, has not yet benefited from the development of information theory and there are many open problems to be solved or

current solution improved using the advances in understanding underlying systems complexity and uncertainty.

Chapter 4
The Data Processing Environment

4.1 Introduction

Data is the source of empirical information and information in turn is the source of empirical knowledge. The primary aim of any data analysis is to retrieve or extract the information that the data can provide. Clearly, different sets of data may contain different information as well as the same data can contain various types of information. The nature of both entities has been discussed in the previous chapter and this chapter will concentrate on a practical topic, namely the tool for data processing, analysis, modelling and simulation.

The data in the discussion will be considered being related to traffic and transportation system. This nevertheless will not affect the generality of the tool to be developed, as one of the criteria of the design is the universality meaning that the developed tool should not assume implicitly or explicitly the exact nature of data but to be flexible and deal with data as an abstract entity.

In transportation, many data collection techniques have been developed and exercised over the years [Taylor, 1996]. Traditional data collection techniques involve sophisticated equipment or much resources thus making the collected data rather expensive [Drew, 1968]. Obviously, data produced by different techniques will be different in its format and semantics. This point is just another illustration of the fact that a universal data processing tool must accommodate different types and formats of data even if they are of the same nature.

This project is mainly interested in data that has been routinely collected by a traffic control system (such as SCOOT) and the developed tool will be exercised on this type of data. While the SCOOT data is easily accessible it is also rather difficult to process and analyse. Consequently, satisfying the requirements for the processing of SCOOT data renders the processing tool to be general and applicable to a broad spectrum of

data processing tasks. Additionally, it is desirable that a set of standard analytical and statistical procedures is available to carry out numerical analysis of the properties of data. This chapter is devoted to development and evaluation of such a tool, which provides the basis for this research study. Several examples will be given to demonstrate the functionality and abilities of the developed software environment.

As has been described in the section devoted to SCOOT data, the data has several features that need to be taken into account during specification of the requirements for the processing tool. The features are summarised below:

- Traffic data that is automatically collected by the traffic control system is represented in a form of textual messages
- SCOOT produces high volumes of messages (hundreds of megabytes per day) even for relatively small road networks under its control
- Messages have standard fields and message-specific fields
- There are nearly one hundred types of messages generated by SCOOT (different types of data and information)
- Messages can represent raw measurements of traffic flow as well as the results of internal SCOOT traffic model (control output, higher level traffic flow estimates, etc.)

SCOOT data is redundant in a sense that it is generated constantly regardless of the situation on the road. Messages carry information about non-zero occupancy as well as zero occupancy. SCOOT produces high volume of data but the actual informative content of the data may be small and filtering out dummy messages can significantly reduce the volume of messages that are to be analysed. Therefore it is the first task of the data processing procedure to filter out empty messages and pass on to the further stages of processing only significant data. Although the above features have been recognised for SCOOT data, those features in a sense are the 'worst' case. Many other traffic control systems produce data with similar characteristics and therefore the developed software is believed to be suitable for dealing with data produced by sources other than SCOOT.

There are several ways traffic data can be made available for processing. Firstly, it can be redirected from the traffic control system straight to the processing software. This will facilitate on-line real-time data processing, which can have a number of potential applications. Secondly, data could have been collected over a past period of time and stored for later use. In this case the historical data comes from storage devices. Thirdly data can be generated as a result of simulation process (with the simulator located on either a local or remote system). The fact that data may come from physically different sources implies the design requirement that the data processing environment should be flexible in order to implement data importing gates for different data sources. Several examples of data sources are: files, network sockets, local system processes, keyboard, mouse, etc.

Data analysis normally requires several stages or procedures applied sequentially. One of the first stages would be to remove faulty data or to detect inconsistencies and handle them appropriately. Then data would be analysed statistically. Visualisation of the results is yet another important stage in data analysis. After all, the results of the analysis have to be made available for the other applications or exported as documents. Normally a user would expect all such functionality to be implemented in the processing software or, at least, have the flexibility to extend the in-built functionality by implementing missing functions and integrating them into the system.

Taking into account the above issues the requirements for such a software system will be given in Section 4.3.

4.2 Review of the software tools for data processing

The importance of data processing and analysis has been already emphasised in the previous chapters. This section will give a brief review of approaches to data processing carried out during traffic-related projects.

4.2.1 The databases

Several researchers have proposed to use databases as the standard approach to traffic data storage and handling [Bell, 1992], [Skabardonis et al, 1997], [Cornwell et al.].

Evans [1995] was concerned with processing and storing SCOOT data, which is the same as the data used in this project. Therefore, a brief description of developing a SCOOT database presented in [Evans, 1995] will be discussed.

The motivation behind the use of databases to store SCOOT data was the ability of databases to store a large quantity of data in a compact, easily accessed and centralised location. The databases also ensure that data has a consistent format with a minimum of redundancy.

The architecture of the developed database is shown in Figure 4.1 (adapted from [Evans, 1995])

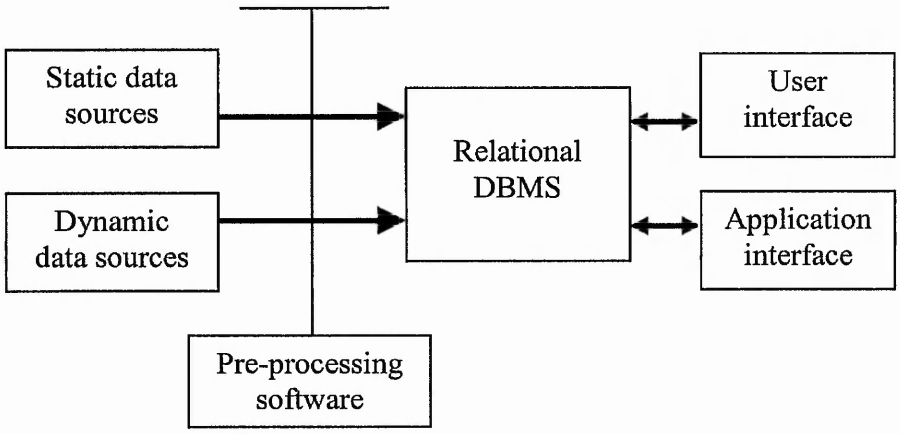


Figure 4.1. The traffic database structure

It is clear from the above figure that data processing is an intermediate stage between data sources and DBMS. Since DBMS do not support custom data processing (unless it is implemented on the level of stored procedures) it has to be implemented as independent layer of the data flow model. For data pre-processing, Evans [1995] uses a suite of computer programmes written in the C and Awk languages. Awk [Aho et al, 1988] is designed to manipulate text files, which contain information that is structured in records and fields and it has been chosen since SCOOT messages have such format.

Databases normally have static structure and design of the structure using the relational algebra is not a trivial task. The COMIS database, presented in Evans [1995], has a

fixed structure that utilises the relationships between SCOOT spatial characteristics such as links, nodes and detectors with two types of messages, namely M16 and M02.

Another system, called ASTRID [Hounsell et al., 1990], was developed by Southampton University as part of a contract funded by the TRRL. The ASTRID system consists of a suite of data collection, pre-processing and analysis programs, that are run on an IBM PC or compatible PC. Within the suite, data collection is carried out using an assembly language program that reads incoming SCOOT messages and logs them onto hard disk. This program runs in the 'background' so that additional tasks can be carried out at the same time as data collection. The messages used in ASTRID are M02, M03, M04 and they provide flow, delay, SCOOT congestion and faults for links, nodes and regions respectively; M16 for stage lengths; C01 for regional cycle time; C30 for degree of saturation of links; W05, W06 provide information on changes in fault status for links. When messages have been logged, they are pre-processed by two FORTRAN programs. One of them splits a file of raw logged messages into a set of sub-files. Each sub-file generally contains one message type for one link, node or region. When producing sub-files, the following additional operations are performed:

1) Unwanted fields are removed from messages;
2) Link flows and delays are scaled by site specific LPUCFs (if these have been specified);
3) Data is aggregated over user-defined time interval

The sub-files produced by the pre-processing stage are subsequently processed by the second FORTRAN program. This program takes one day's worth of message sub-files and uses them to update a set of *trend* and *profile* files. These files form the core of the ASTRID system. Trend and profile files are manipulated for a specified set of links, nodes and regions.

The advantages and disadvantages of using databases in traffic data processing and analysis are apparent. While databases provide a unified structure and interface to the data, there are several drawbacks in the data source-database-application scheme. Briefly they are as follows:

1) In real-time and time-critical applications data is passed through the database causing a significant overhead

2) There is no mechanism in databases that would notify the applications about the arrival of new data. Due to this, applications, that require real-time data have to scan databases continuously, which causes a significant overhead to the whole system
3) Databases are difficult to upgrade or add new relationships

And the advantages of the database approach are also clear:

1) Databases are good when no real-time processing is required, that is for off-line data storage and access.
2) Database Management Systems provide a high level of data safety and consistency, especially when the database is accessed from several remote locations.
3) Databases provide fast random access to any historical data that were stored in it.

In the data processing environment developed within this project, the possibility of using database is open and an interface to a database can easily be implemented thus allowing the user to utilise the advantages provided by the database, but the software is not based on database thus giving the user the flexibility in choosing the working framework that is more suitable for a particular task.

4.2.2 Data processing tools for traffic projects

A set of tools for data processing and analysis have been developed during the Freeway Service Patrol Evaluation project carried out in University of California, Berkeley [Petty, 1995], [Petty et al., 1996]. The suite of programmes called FSP has been developed in order to facilitate the processing and visualisation of the data that was collected during the Freeway Service Patrol Evaluation project. The main function of the tools was to perform diagnostics on the data, generate error reports and make plots of various pieces of data.

The main fsp programme has been written in the C language for UNIX platform and made free for academic use. It takes as its input arguments a file that contains a set of

commands that fsp is to perform, an incident filter file and incident run number. The input of the data into the fsp program is accomplished by placing data files into the specified directories in the file system. The hierarchy of input data directories has to be created. A graphical user interface front-end to the fsp programme, called xfsp, allows the user to select the input data files that are to be processed, start the data fixing procedure, and run fsp to produce output files and browse the content of the output files. xfsp was written in the TCL/TK scripting language and is meant to operate under X-Windows sub-system of UNIX OS.

The data collected during the Freeway Service Patrol Evaluation project was of three categories: the loop detector data, the probe vehicle data and incidents data. The loop data is collected from inductive loops that are embedded in the highway. At a specific location, each lane contains a pair of loops that are spaced about 14 feet apart (double-loop detectors). When a vehicle passes over a loop, the effective inductance in the circuit changes and this results in a change in the current flowing in the loop and detected by the detector using a threshold level. Binary data is generated by sampling the result of the comparison of the current level and threshold at the frequency of 60Hz. For a desired data output period, the fsp programme generates the following information for each interval.

- The number of vehicles that pass over the loop location in that interval
- The fraction of the time in that interval for which the loop is occupied
- The mean speeds in that region during that interval

The fsp programme recognises and tries to fix inaccuracies or inconsistencies in the data that occur due to a variety of reasons, such as

- The loop may not be sensitive enough to detect all the vehicles passing over them and thus may in fact be undercounting. Also, vehicles not in the centre of the lane may be missed, as is the case of vehicles attempting to change lanes. On the other hand, loops may be detecting vehicles that are on neighbouring lane and hence resulting in some overcounting
- The resolution with which the time headways can be measured is 1/60 second. This limits the accuracy of measured vehicle speeds. Further, mismatch in

tuning of a pair of coupled detectors leads to errors that cannot be compensated for.

- The detector thresholds may not be tuned properly and this can result in bias in the measured occupancies.

The developed fsp software does what it was developed for. Unfortunately, there are several problems associated with the use of the software for purposes other than those specified in the Freeway Service Patrol Evaluation project:

- It lacks the generality and should data format or analysis procedures change it will not be able to cope with the changes. It is thus a very specific implementation of data processing software for the very specific problem.
- The input to the programme should be provided in the form of files that contain necessary data and the output of the programme is organised in the form of report files. This makes the use of the software in real-time processing and analysis procedures inconvenient and clumsy.
- Another problem is that since the core programme is written in the C language it is difficult to extend it in the sense of adding new functionality.

The above problems are analysed in this research and taken into account during the development of the data processing environment.

4.2.3 Other data processing, modelling and analysis software packages

There are several commercial software packages that may be considered as candidates for the use as data processing, modelling and simulation tools. Features and capabilities of some of them will be reviewed in the following sections.

4.2.3.1 VisSim™

VisSim is a software program for the modelling and simulation of complex dynamic systems [Visual Solution]. VisSim combines an intuitive drag & drop block diagram interface with a powerful simulation engine. The visual block diagram interface offers a simple method for constructing, modifying and maintaining complex system models.

The simulation engine provides fast and accurate solutions for linear, non-linear, continuous time, discrete time, time varying and hybrid system designs. With VisSim, users can quickly develop software or "virtual" prototypes of systems or processes to demonstrate their behaviour prior to building physical prototypes. Built-in integration tools allow users to communicate seamlessly with MATLAB and Mathcad.

The user builds his system model by selecting predefined blocks from a block library and simply wiring (graphically connecting) the blocks into a diagram. Each block of the diagram performs a mathematical or input/output function. These "blocks" may represent complex algorithms, input variables, or various outputs like graphs, charts, plots or data files. Users can also create custom blocks in C, Fortran or Pascal and add them to the VisSim block library. After the model is configured, a simulation is run and results of the simulation are displayed. The core product, VisSim, is used for general modelling, simulation and control system design applications. VisSim product options include VisSim/Analyze, for frequency domain analysis, and VisSim/Real-Time, for real-time hardware-in-the-loop validation. In addition, there are several extension packages that add to VisSim functionality that is necessary in particular applications such as digital and analogue signal processing, analogue, digital or mixed-mode, end-to-end communication systems, frequency domain analysis and other add-ons.

Due to its efficiency, intuitive block-diagram interface and simplicity of use VisSim has become very popular in engineering circles and is mainly used for solving problems studied in the classical control theory.

One significant disadvantage of VisSim for modelling and simulation of transportation systems is that it only supports numerical data types whereas in the transportation system models variables and parameters may be represented as strings, numbers, logical values or more complex structures such as spatial characteristics of a road network. Another disadvantage is that blocks that form a model of a system work in synchronous mode only and also it is not a multi-threaded product (thus cannot take advantage of parallel processor architectures).

A graphical user interface, similar to VisSim's, has been chosen as a prototype for the data processing environment's user interface as it is the most intuitively understandable

way of representing stages of data processing and transfer between individual processing or analysis functions. Block diagrams are also a common tool for visualisation of control system models.

4.2.3.2 MATLAB™ and Simulink™

MATLAB and Simulink™ are products of The MathWorks, Inc. [The MathWorks]. MATLAB is an integrative environment with its own language and interpreter, which has been especially designed for mathematical calculations. Over the years of its evolution, many packages of functions have been developed and cover almost all topics of applied mathematics including modern theories of neural networks and fuzzy systems. MATLAB has been implemented for the most software and hardware platforms including Sun (Solaris), Hewlett Packard (HP-UX), IBM PC and compatible with MS Windows and Linux and Apple (from version 6.5), which is one of the factors of its popularity.

Simulink is an interactive tool for modelling, simulating, and analysing dynamic, multinomial systems. It lets a modeller build a block diagram, simulate the system's behaviour, evaluate its performance, and refine the design. Simulink integrates seamlessly with MATLAB, providing immediate access to an extensive range of analysis and design tools. Its functionality is similar to VisSim. One of the advantages is that Simulink is built on top of MATLAB and this gives a modeller access to the powerful analysis and graphics capabilities of MATLAB. It also allows writing custom modules on a high-level language of MATLAB whereas VisSim requires its blocks to be written on one of the general purpose languages, such as C or Pascal.

The disadvantages of Simulink are almost the same as VisSim's. The main disadvantage is that it supports several, but all are numerical, data types, which include 32- and 64-bit floating-point, fixed-point and integer. It also has a less intuitive graphical interface.

4.2.4 Conclusions

The problem of data processing as such is rarely considered by the researchers to be worth of special treatment. Since different data requires different processing

procedures, researchers would normally develop a small and very task-specific application that would allow them to transform the source data into some standard representation (such as numerical data: vectors, matrices, etc.) in order to use standard analytical tools to carry out further analysis. This essentially means that the software developed for that particular task cannot be used should source data format or output representation changes. This leads to a very little re-usability of the data processing code, which in general implies extra costs for developing new applications for new data formats or projects. Moreover, a wide range of general-purpose analytical tools creates a problem of migrating from one to another. The migration is a frequent event since there is no a single software system that provides all necessary functionality for every project and problem escalates even further as researchers are often forced to write their own tools and procedures to perform the analysis of the data they require.

4.3 Requirements

Having studied the disadvantages of the existing modelling and data processing and analysis software packages, a set of requirements have been produced that the data processing environment must satisfy. The following sections present the requirements and discussions on their necessity.

4.3.1 Modularity

As it has been mentioned in the introduction, a typical analysis process consists of several stages which data is passed through in order to extract certain pieces of information. Given that the range of functions performed at each stage is vast it is impossible to implement every function in one software system. One possible solution is to require the system to be modular, or in other words all individual functions that can be used for analysis be implemented as functional modules which then can be added into the system and manipulated with. The data processing environment must provide the means by which such functional blocks are integrated into the system and manipulated within it (placed, parameterised, connected, controlled etc.).

4.3.2 Extensibility

Extensibility is basically the consequence of the modularity requirement, which states that the system must be easily extensible by adding new functional modules, which will extend the system's functionality. Modular extensible system has great flexibility in meeting particular analytical requirements of the user. For example, the user can implement and integrate into the data processing system a decision support module that will assist the user in making decision based on the data. Another possibility is the implementation of a control system that produces control output based on data analysis. As a consequence of the extensibility, the developer of the data processing environment is made free from the need to implement many functions into the system since all task-specific functionality can be added to the system in the form of modules.

4.3.3 Efficiency

The system must be efficient enough in order to deal with high volumes of data. Since data processing and analysis can be a continuous real-time process, the efficiency requirement also implies that the computational resources are to be distributed between the functional modules accordingly. For example, as has been mentioned before, high volume of raw data does not necessary imply high volumes of information but extraction of relevant pieces of data can be computationally expensive process. Therefore, the pre-processing stage can take much computational resources but should not overshadow the rest of the analysis process.

In some cases system might not have enough computational power to cope with high volume of data in real time. In such cases the system can only be used as off-line data analyser and cannot be used in real-time decision-making processes.

The computational power can be significantly improved by using parallel or multi-processor hardware. In the case of such hardware being available, the system must utilise the advantages of the resources. This idea is expressed as the following requirement.

4.3.4 Concurrency

This requirement can be considered as a consequence of the efficiency requirement and states that functional modules are executed concurrently in separate threads of executions. In this case the underlying operating system can make use of the available multi-processor hardware resources with the greater flexibility. One drawback of this requirement is that there will be additional computational costs for synchronisation between the processes, which can induce overhead of up to 10% on a single-processor system. Nevertheless, it is a good trade-off when the system is used on a multi-processor hardware.

4.3.5 Reliability and fault tolerance

Since the user can integrate arbitrary code into the system, the system should be robust towards the errors and crashes within the user's code and such crashes within one block should not affect the whole system. This nevertheless should not limit the range of functions available to the user.

Multi-threading feature has its own impact on reliability. That is the functional modules that are run in separate threads of execution are protected against errors that occur within the other modules. A crash within a module can affect the other modules only in a sense that it will stop providing data to the peer modules. This may cause the system (or a set of modules within the system) to cease functioning (put it into a waiting state) but such situation is not a fatal error and the system should be able to recover.

4.3.6 Universality

Data can be received in a variety of formats and representations. Different types of data require different internal types and treated accordingly. For example, numerical data can be represented as real numbers, integer numbers, complex numbers, vectors, matrices and other forms. Some data may be expressed in symbolical form and some data in the form of textual messages. A universal data processing tool must be ready to

accommodate various needs for data types, from the primitive types, such as numbers, to complex types describing sophisticated data structures.

This requirement is not satisfied by most of the commercial software packages that allow the user building system models from basic blocks and carry out simulations in a convenient graphical environment. This is because most of them allow only a limited set of data types, normally numerical types, such as boolean, floating point real or integer numbers, and vectors or matrices of numbers. Such limitation renders the packages useless in solving problems that involve manipulations of heterogeneous objects, as is the case with traffic data processing and analysis processes.

This feature of the data processing system to operate with custom data types and formats will be called universality.

4.3.7 User-friendliness

Graphical user interfaces have become standard de-facto and are expected by the users. Therefore a convenient graphical interface should be provided by the system. It is also desirable that the user be able to manipulate the functional modules, that is, creating instances of the modules, customising module's parameters, placing and wiring them, in a convenient visual environment. As a prototype of the interface, the VisSim's (Section 4.2.3.1) interface has been chosen as it is the most intuitive and flexible.

4.4 Analysis and Design

This section will describe the design of a new software product for data processing and analysis. Taking into account the disadvantages of the existing software tools the new design will follow the specified requirements. The design has been carried out with the object-oriented paradigm in mind.

4.4.1 Use cases study

Many projects begin with use cases, as they provide a good way of obtaining an overall picture of what is happening in the existing system or is planned to happen in the new

system [Bennett et al, 2001]. Use cases diagrams show *use cases* and *actors* and the associations among them. Use cases development approach has been investigated by [Jacobson et al, 1992], [Jacobson et al, 1999], and now forms an important part of UML standard. In Jacobson's approach, use cases were the starting point for the development of a new system. Due to its simplicity and intuitive clarity, this approach has been employed in design of the data processing system.

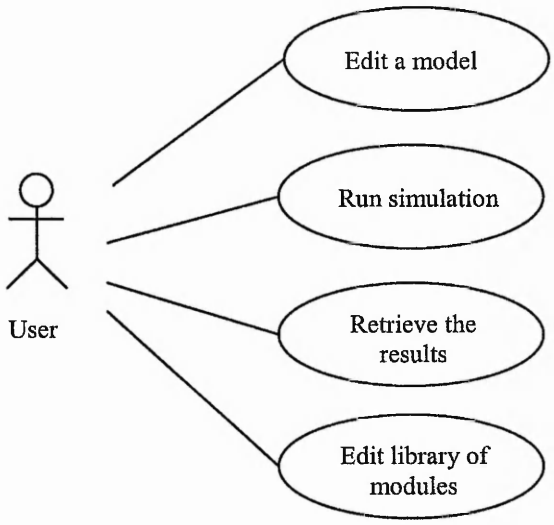


Figure 4.2. Basic use cases and actors

Figure 4.2 provides the basic use cases for the data processing environment. It consists of an actor ("the user"), and four types of activities the user performs with the system.

1) *Edit a model.* This use case includes all activities related to addition or removal of functional modules into/from the current model, placing connections between the modules and updating the modules' parameters according to the requirement of the model.
2) *Run simulation* is the key activity of the system and corresponds to execution of individual functional modules and passage of data between them.
3) *Retrieve the results* activity corresponds to an intermediate or final stages of simulation, when partial or complete results of the simulation become available and the user is able to retrieve them from the system in some conventional format such as tables, graphical plots, etc.

4) *Edit library of modules* activity corresponds to the extension of the system by adding new functional modules or removing the existing ones from the system. This activity represents the extensibility requirement of the system

The following sections develop each use case individually.

4.4.1.1 Edit a model

Edit a model use case can further be developed as given in the following figure.

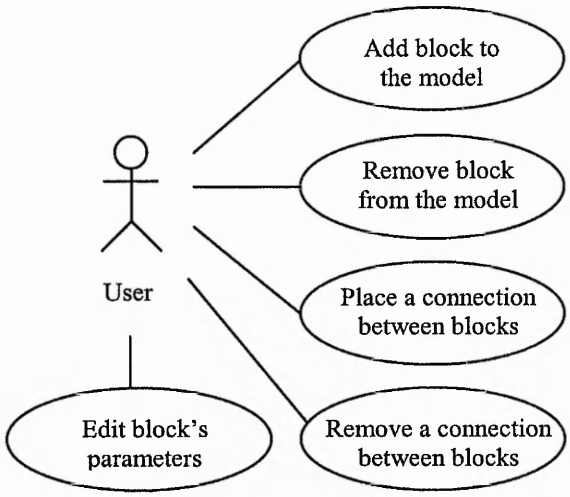


Figure 4.3. 'Edit a model' use case diagram

4.4.1.2 Run simulation

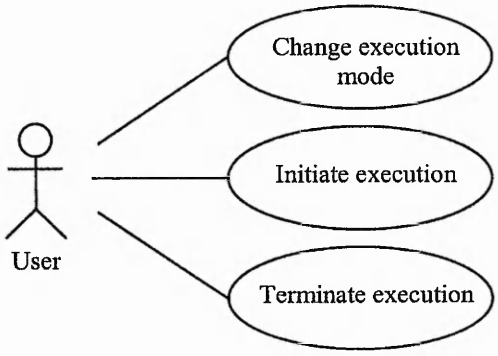


Figure 4.4. 'Run simulation' use case diagram

4.4.1.3 Retrieve the results

Retrieval of the results of simulation is an activity of data transfer between applications or storage devices and is highly dependent on individual functional modules. The data processing environment does not incorporate any centralised facility for result collection and presentation and this functionality must be provided by particular modules. Examples of such modules, such as plots and histograms, will be developed later in the chapter.

4.4.1.4 Edit library of modules

In order to create individual instances of functional modules, the data processing environment needs the information about implementation of the particular modules. This information is stored in the *library of modules*. In order to extend the system with a new module or remove information about a module, a user edits the library accordingly.

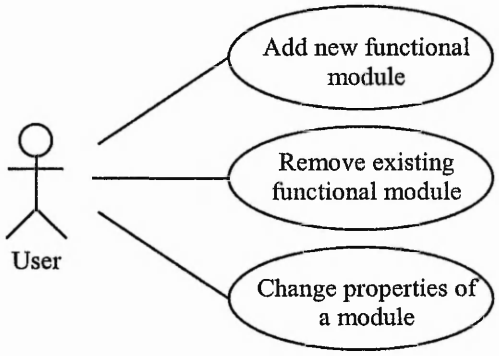


Figure 4.5. 'Edit library of modules' use case diagram

The change of a module's properties is used in a general sense without specifying which properties are to be modified. That means the freedom is left to the implementation to specify the properties and the ways they are changed. An example of such properties is the graphical icon associated with the module's implementation that can be placed on a tools bar for instant access to the module's instance generating factory.

4.4.2 Functional modules

The data processing environment itself does not provide any applied functionality. A design decision has been made to move all applied functionality to pluggable modules. From this standpoint, the data processing environment is the environment in which the functional modules operate. An analogy can be drawn between this architecture and well-known [JavaBeans™ Technology]. In this architecture, a functional module is an independent object connected to the system via an interface or several interfaces. The following figure demonstrates the idea.

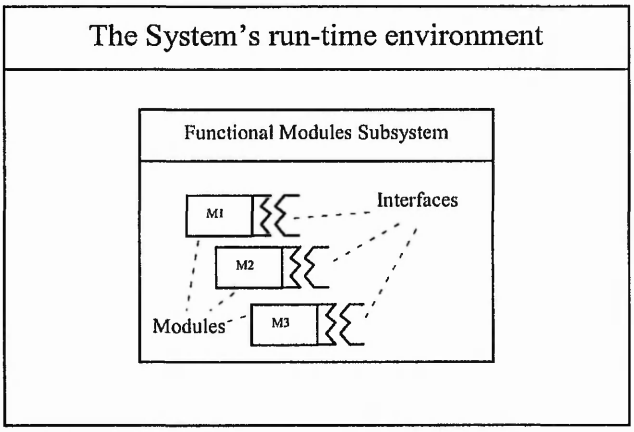


Figure 4.6. Bean-like environment for functional modules

In the above scenario, the functional modules play the role of actors in the use cases diagram, which is given below.

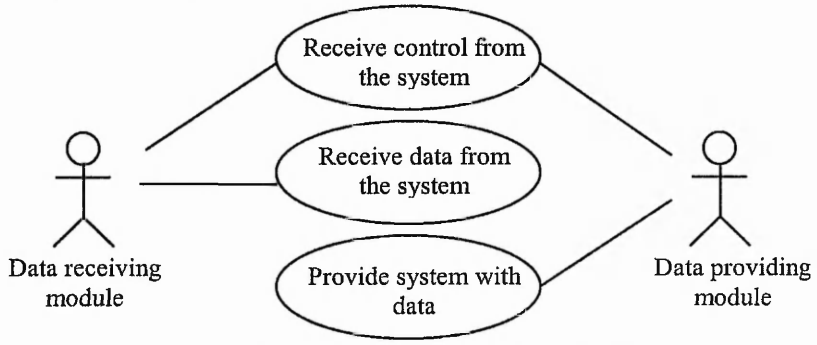


Figure 4.7. Modules use case diagram

From the above use case diagram it is clear that modules can appear in two roles – as data providers and data receivers. Data providing modules produce data according to their internal logic and pass it on to the system. From the system the data is distributed

among the data receiving modules, in order to be processed further. A module can play both roles, as data provider and data consumer, or just one of them. The roles of a module will be determined by interfaces that the module implements. Following from the two roles, there are two interfaces that describe each role. The interfaces will be developed in detail in the section devoted to implementation.

In order to simplify the task of module developer, some common functionality, such as execution control, data passing, errors reporting and recovery, etc., can be implemented in the system in a form of module wrapper (Figure 4.8). The concurrency requirement imposes the need for the modules' code to be executed in a separate thread of execution, independent of the system's thread. A module whose code is executed in a separate thread of execution will be called active module. It is sufficient to implement either of the two modules' roles as active module. A decision has been made to make data providing modules active since it is the most natural way. The data receiving modules' code will be executed from either the system's or a data providing module's thread.

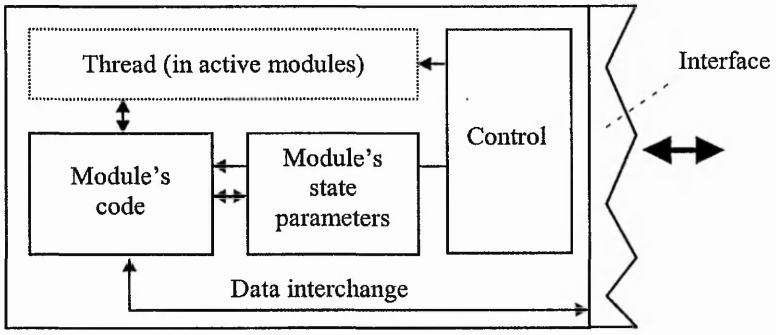
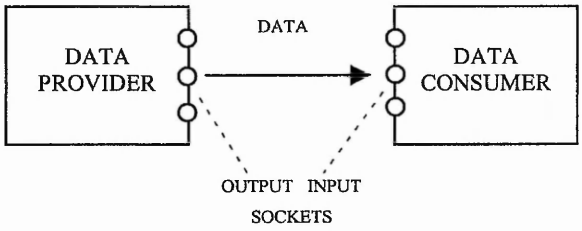


Figure 4.8. Structure of a module wrapper

Thus, the functional modules subsystem of the system consists of a set of instances of modules wrapped in appropriate wrappers as shown in the figure above. The wrappers serve as an intermediate level between the code that implements the module functionality and the system providing necessary functionality for control, monitoring and data interchanging with the module.

4.4.3 Inter-module communication and data typing

The distinguishing feature of the data processing environment from other similar products, as it has been noted before, is the flexible data typing system. Let us consider two functional modules, a data provider and a data consumer.



In order to connect these modules, a data structure that is called c*hannel* is used. A channel consists of two pairs of parameters, namely

Channel = ((data provider, output socket), (data consumer, input socket))

where 'data provider' and 'data consumer' are references to instances of the modules accordingly. When a connection between two modules is made, an instance of channel is being created and registered with the two modules. It is obvious that, only one channel can be registered with a particular socket of the data consuming module but many channels can be associated with a socket of the data providing module.

A set of functional modules, together with a set of channels, form an oriented graph of the model. Workspace of the system is the necessary container where the graph of the current model is kept (Figure 4.10).

In order to facilitate an interchange of data of any abstract type, from the system's standpoint modules interchange objects (instances) of the most abstract type. In the C language such objects can be represented by a pointer with the void type (`void*`), in Java language it is an instance of the basic class `Object`. Since no assumption about data types is done in the system itself, it is the responsibility of the modules to insure correctness of the data types. To do so, a special set of methods must be provided that allow the system to check if a particular data type of a data giver is acceptable to a data taker. Such methods must be implemented in all modules that are supposed to be

participating in data interchange. Since data is provided by a data provider module, the first method must allow the system to retrieve information about the type of data provided by the module. A decision has been made to use a textual string to encode such information, namely the system retrieves data type information as a textual string. The format of the string is not specified but must be known to data consumer, which is to be connected to the data provider. Thus, another method is provided by a data consumer that informs the system if the data type obtained from the data provider is either acceptable or not. The following Figure 4.9 demonstrates the typing system implemented in the data processing environment.

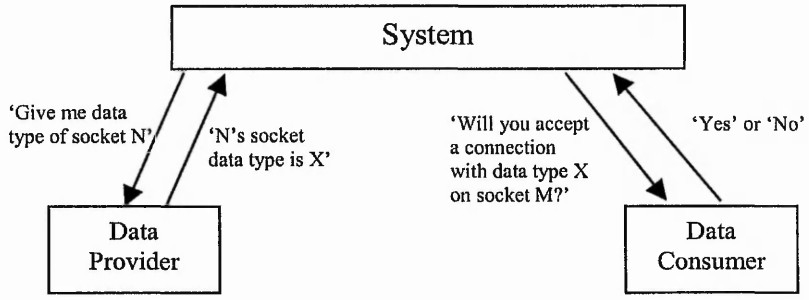


Figure 4.9. Illustration of the data typing system

If the system gets a positive response from the data consumer, a channel is being created and registered with the both modules, creating thereby a data communication link between the two modules.

4.4.4 The system components

From the analysis of the use cases of the system it is clear that the system must be split onto several subsystems, each responsible for certain activity. In this section the sub-components of the system will be identified and modelled.

The obvious components of the system that can be recognised are:

1. The modules environment. This part of the system holds all necessary data structures and infrastructure to keep current model, that is a set of functional modules and links between them and also providing necessary interfaces to access and control the model. This part of the system will be referred to as *workspace*.

2. Library of modules that implement functional modules. This part is a database that keeps records about implementation of functional modules and provides a factory for creating instances of the modules. Additional supplementary information can be kept along with the records.
3. GUI. Graphical user interface is an important part of the system which provides interface for a user to internals of the system and the means to control, edit and perform other operations over the model and the library of classes.

The overall structure of the system is presented in the following Figure 4.10.

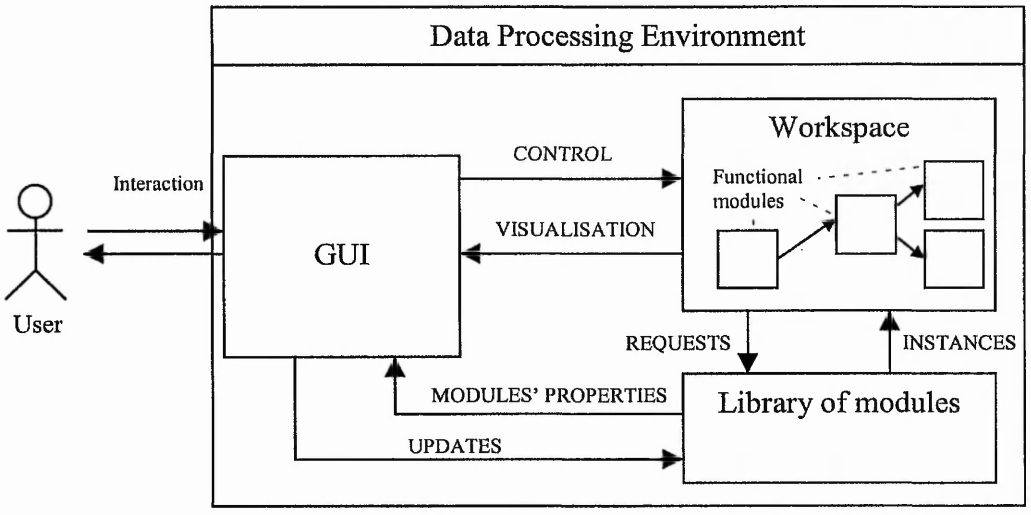


Figure 4.10. The system's architecture

4.5 Implementation

This section will present the implementation of the data processing environment according to the previously given design. First step of the implementation is the choice of the platform under which the software will be developed. A discussion on available options is given in the first part of the section. The following sections will give the overview of the implementation process for the chosen platform.

4.5.1 Choice of platform

Basically, the choice of the platform would consist of three main stages:

1. The choice of the hardware platform. There is a variety of hardware platforms available such as IBM PC or compatible, Sun Workstations, Hewlett Packard, Compaq, etc. The choice of hardware will also affect the choice of the software platform, namely the underlying OS.
2. The choice of the underlying OS. There is a bigger variety of operating systems since several OS can run on the same hardware platform. The most common choice for IBM PC/compatible is Microsoft Windows family of systems (Windows 95, 98, NT, 2000, XP) or Linux (a free UNIX-like operating system). For other hardware platforms (Sun, HP, IBM) there is one or another variant of Unix OS (Solaris, HP-UX, AIX).
3. The choice of the language and development environment for the chosen OS and hardware platform. Among languages the most popular are C/C++, Java, Pascal, ADA, Lisp, Perl, which are available for the most hardware and software platforms.

One exception from the process given above is the Java platform. Java platform has been designed with the aim to create a highly portable and hardware/OS independent environment, which can run on most of the platforms and software developed for it can be run with no modifications on all of the supported platforms. It does not, therefore, require the steps 1 and 2 in the above scheme.

There are two the most common platform groups (hardware and software) which can be referred to as MS Windows based platforms and Unix based platforms. Another platform is unique in some sense, namely the Java platform. These three platform groups will be discussed further.

The first prototype of the data processing software for this project has been implemented in the C language for Microsoft Windows 95™ as a console application. The prototype clearly had several disadvantages and did not meet the specified requirements. Firstly, it is very difficult to integrate new modules into the system. It would normally require writing a C module, compiling it into the object file and then

linking with the whole system. This process assumes that the user is familiar with all the stages and has access to the system in at least compiled form (as a library). Secondly, it is next to impossible to control correctness of the added modules written in the C language, as the modules will be compiled into the byte-code of the underlying hardware platform. The prototype also lacked graphical user interface and concurrency. Clearly, DOS-like console technology is obsolete and therefore is not an acceptable choice.

Microsoft Windows™ operating systems family (MS Windows 95, 98, ME, NT, 2000, XP) are modern systems that provide full-featured graphical window-based interface and multi-tasking capabilities. MS Windows also allow multiple threads of execution within one application and provides the means for dynamical loading and linking of code, namely Dynamically Loaded Libraries (DLL). All these features are sufficient for implementation of the requirements specified for the data processing environment. An advantage of implementation of the system for MS Windows platform is that the Windows system is the most popular in the sector of personal computers, the dominant sector of computing market. Despite its popularity, MS Windows OS has been a target of criticism regarding its unreliability and poor performance for many years.

A family of UNIX-like operating systems, which will further be referred to as Unix OS, is another candidate. Unix has been developed as a server operating system, which implies that the stress during its design was put to security and reliability. From the beginning it was designed to be a multi-user and multi-tasking operating system. Since the development of the IP protocols family, the networking support facilities are integral part of the OS and as a consequence, Unix OS has been the system of choice for Internet and network application servers. Since it has been designed to be a multi-tasking system it has a strong support for semaphores and shared memory system. In addition, X11 sub-system is a powerful and flexible graphical environment that provides all necessary functionality for building comprehensive graphical user interfaces. The main disadvantage of Unix systems is that they are regarded by the users to be difficult (complicated) to use and learn to use and therefore the users of Unix OS are mostly professionals of IT and computing field (such as system programmers and administrators) or academics.

The last platform to be considered is the Java platform developed by Sun Microsystems. It differs from the traditional platforms by that its run-time environment runs on the top of the traditional OS, such as MS Windows or Unix. It provides a portable, interpreted, high-performance, simple, object-oriented programming language, which, together with the run-time environment (Java machine) forms the core of Java platform. The following overview of the Java platform is based on the Java's white paper by [Gosling, McGilton, 1996].

The main design goals of the Java programming language have been

1. To create a **Simple, Object Oriented,** and **Familiar** language. The Java language is *simple* because the fundamental concepts of the Java technology are grasped quickly; programmers can be productive from the very beginning. The Java language has been designed to be *object oriented* from the ground up, provides a clean and efficient object-based development platform. Keeping the Java programming language looking like C++ as far as possible results in it being a *familiar* language, while removing the unnecessary complexities of C++. Having the Java programming language retain many of the object-oriented features and the "look and feel" of C++ means that programmers can migrate easily to the Java platform and be productive quickly
2. The language must be **Robust** and **Secure**. The Java programming language is designed for creating highly *reliable* software. It provides extensive compile-time checking, followed by a second level of run-time checking. Language features guide programmers towards reliable programming habits. The memory management model is extremely simple: objects are created with a `new` operator. There are no explicit programmer-defined pointer data types, no pointer arithmetic, and automatic garbage collection. This simple memory management model eliminates entire classes of programming errors that bedevil C and C++ programmers. The Java language is secure since compiler and run-time system implement several layers of defence against potentially incorrect code. The environment starts with the assumption that nothing is to be trusted, and proceeds accordingly
3. **Architecture Neutral** and **Portable**. Java technology is designed to support applications that will be deployed into heterogeneous network environments. In such environments, applications must be capable of executing on a variety of

hardware architectures. Within this variety of hardware platforms, applications must execute atop a variety of operating systems and interoperate with multiple programming language interfaces. To accommodate the diversity of operating environments, the Java Compiler™ product generates *bytecodes* - an *architecture neutral* intermediate format designed to transport code efficiently to multiple hardware and software platforms. The interpreted nature of Java technology solves both the binary distribution problem and the version problem; the same Java programming language byte codes will run on any platform. The architecture-neutral and portable language platform of Java technology is known as the *Java virtual machine*. It's the specification of an abstract machine for which Java programming language compilers can generate code. Specific implementations of the Java virtual machine for specific hardware and software platforms then provide the concrete realization of the virtual machine. The Java virtual machine is based primarily on the POSIX interface specification--an industry-standard definition of a portable system interface. Implementing the Java virtual machine on new architectures is a relatively straightforward task as long as the target platform meets basic requirements such as support for multithreading.

4. **High Performance**. The Java platform achieves superior performance by adopting a scheme by which the interpreter can run at full speed without needing to check the run-time environment. The *automatic garbage collector* runs as a low-priority background thread, ensuring a high probability that memory is available when required, leading to better performance. Applications requiring large amounts of compute power can be designed such that compute-intensive sections can be rewritten in native machine code as required and interfaced with the Java platform. In general, users perceive that interactive applications respond quickly even though they're interpreted. The overall performance of Java machine has been tremendously improved after introduction of the Just In Time (JIT) Compiler. The JIT compiler works in parallel with a Java application and compiles application's Java bytecode of the classes that are most intensively used into the native code of the underlying platform, thus making the performance of Java applications comparable to similar applications written in the C/C++ language.
5. **Interpreted**, **Threaded**, and **Dynamic**. The *Java interpreter* can execute Java bytecodes directly on any machine to which the interpreter and run-time system have been ported. In an interpreted platform such as Java technology-based system,

the link phase of a program is simple, incremental, and lightweight, thus allowing a much faster application development cycle. The Java platform supports multithreading at the language level with the addition of sophisticated synchronisation primitives: the language library provides the `Thread` class, and the run-time system provides monitor and condition lock primitives. While the Java Compiler is strict in its compile-time static checking, the language and run-time system are *dynamic* in their linking stages. Classes are linked only as needed

Considering the advantages and disadvantages of the above platforms, a decision has been made to use Java platform for the implementation of the data processing environment.

4.5.2 Package architecture

The system has been split into three packages as follows

1. `edu.tntu.ism.traffic` contains the core classes of the system, such as basic data structures, data objects, interfaces, workspace manager, library of classes manager, module wrappers, etc.
2. `edu.tntu.ism.traffic.ui` contains GUI-related classes
3. `edu.tntu.ism.traffic.modules` contains implementation of standard functional modules and examples.

The main package `edu.tntu.ism.traffic` will be overviewed in the following section. Package `edu.tntu.ism.traffic.ui` will be overviewed in the section devoted to GUI and `edu.tntu.ism.traffic.modules` in the section devoted to examples.

4.5.3 Package `edu.tntu.ism.traffic`

The package consists of the core classes of the data processing environment. The classes can be split into several logical groups according to the functionality they implement. The first group consists of the basic data structure classes and interfaces.

Table 4.1 presents interfaces of the first group of classes with description of their function. Table 4.2 provides information on basic data structures and objects used in data interchange process and inter-class communication.

Table 4.1. Interfaces of `edu.tntu.ism.traffic` package

Interface name	Description
`DataGiver`	This interface specifies methods and members that a custom module should implement in order to make it recognisable by the system as data provider
`DataTaker`	This interface specifies methods and members which every custom module should implement in order to be recognised by the system as data receiver
`Indicator`	This is a GUI-related class. Modules that implement this interface support custom look. Interface defines one method which allows the system to retrieve information from the module's class about how it should be rendered on the GUI's working space.
`CommandListener`	Implementing classes can be controlled by sending them commands
`ComplaintListener`	Implementing classes receive information about faults in a functional module
`DataListener`	A class that receives data (`DataMessage`)
`ParameterChangeListener`	An implementing class can be registered for notification of changes in a parameters objects.
`Structured`	By implementing this interface a class states that it has some 'structure' which can eventually change
`StructureChangeListener`	By implementing this interface a class states that it is interested in knowing that some other class (`Structured`) has changed its internal structure

Table 4.2. Data object classes of `edu.tntu.ism.traffic` package

Class name	Description
`CommandEvent`	This class implements a 'command' object. Commands are sent from commanding classes to their `CommandListener`'s
`ComplaintEvent`	This class implements a 'complaint' object. Complaint objects are sent from the module wrappers to `ComplaintListener`'s in case a fault has been detected within a module's core class.
`ParameterChangeEvent`	Instances of this class carry information on a change within a parameter class (`Parameters`). Sent from `Parameters` to `ParameterChangeListener`'s
`StructureChangeEvent`	Information on the change within a `Structured` class. Sent from `Structured` to `StructureChangeListener`'s.
`DataMessage`	This is a wrapper for objects that are passed between the functional modules. Sent from `DataGiver`'s to `DataTaker`'s via `Channel`.

Table 4.3. Exception classes of `edu.tntu.ism.traffic` package

Exception name	Description
`DataGiverDone`	`DataGiverDone` exception is thrown from an instance of `DataGiver` to inform the system that there is no more data to provide
`IncompatibleSocketTypeException`	This exception is thrown when an attempt has been made to connect two modules with incompatible data types
`ParameterDataException`	Thrown from an instance of `Parameters` class when incompatibility between parameter type and data has been detected or attempt has been made to access a missing parameter

Table 4.4. System core classes

Class name	Description
`ClassLibrary`	`ClassLibrary` is a database that keeps records of classes that implement functional modules available to use within the data processing environment and provides necessary methods to access the records and create new instances of modules
`SpaceManager`	`SpaceManager` is a database that keeps the graph of current model and provides necessary functionality to query, update and control the model
`Channel`	`Channel` represents an inter-module communication link
`DataGiverModule`	A wrapper for data providing modules
`DataTakerModule`	A wrapper for data consuming modules
`DummyDataProvider`	A data providing module with no underlying functionality
`DummyDataConsumer`	A data consuming module with no underlying functionality
`UniversalModule`	This module encapsulates the features of data provider as well as data consumer. This module is 'universal' in a sense that it can output as well as input data, and essentially a higher-level wrapper for functional modules
`Parameters`	This class represents a bunch of parameters that can be associated with any object. This is a standard interface, which allows the setting up and tuning a range of objects in a unified manner

4.6 Graphical User Interface

User interface is an important part of any software and plays not the last role in the success of the whole software product. Certain 'unwritten' standards for user interfaces have been developed over years and most of the users will expect the software to conform, to a certain degree, to these standards. It has been mentioned before, some interfaces developed for dynamical system simulation packages showed to be intuitive

and pleasant to the users. Since the data processing environment has similar ideology of functional block-like modelling, choosing such interface as prototype seem to be appropriate.

The user interface for the data processing environment has been developed using JFC [Swing] toolkit of lightweight GUI components. The following pseudo-diagram presents the underlying architecture of the GUI system.

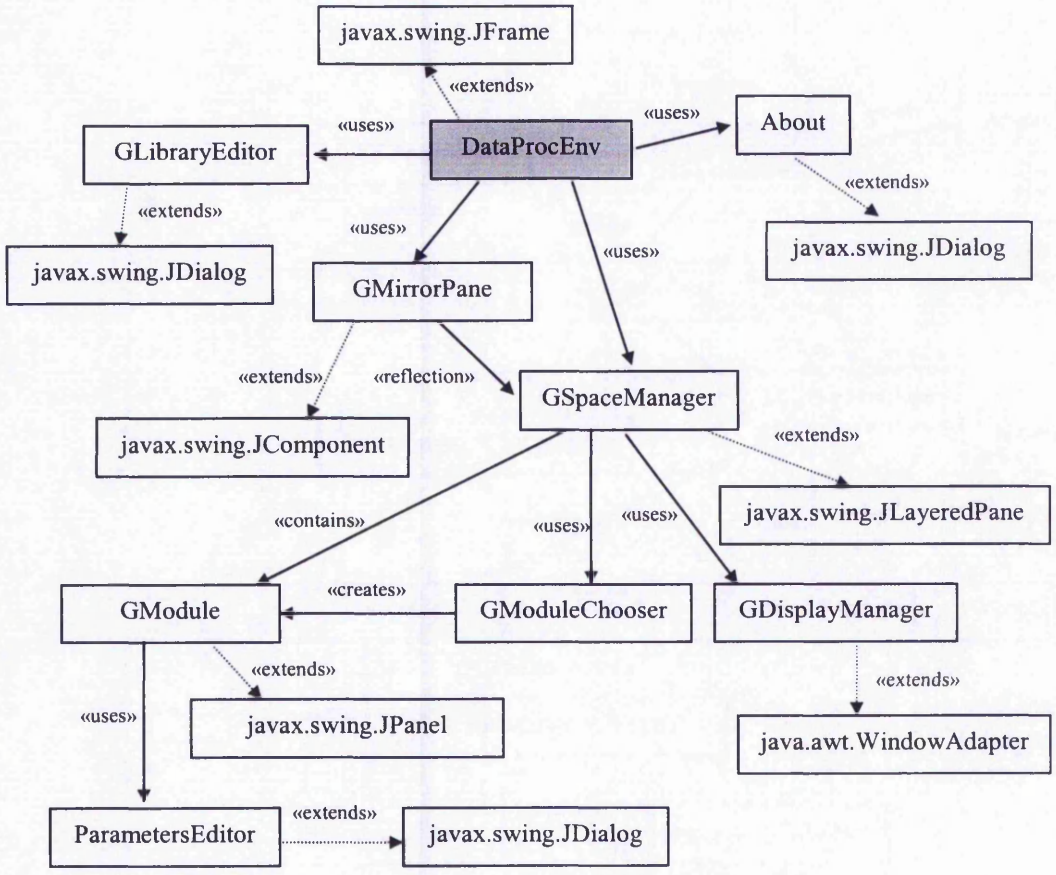


Figure 4.11. GUI classes architecture

Appendix B provides several screenshots of visual appearance of the data processing environment. Main features of the GUI are summarised as follows.

1. The main window contains the main menu, workspace, 'mirror' pane and toolbars with shortcuts to the modules creation function
2. A user places modules on the workspace by clicking an icon on one of the toolbars (right, left or bottom) or from 'Module chooser' window. After a module has been

selected it appears as semitransparent and follows the mouse pointer. The module can be permanently placed at the desired location by left click of the mouse. The placement can be cancelled by right click of the mouse.

3. Top toolbar provides buttons for instant access to the control of the module. By clicking a certain button, the user can start, pause or stop model simulation, copy, cut or paste modules stored in clipboard, save current workspace in a file or load a model file into current workspace. Several buttons are provided to change simulation modes: flow execution, delayed execution, step-by-step execution or discarding execution (data is not passed via channels – it is discarded)
4. Library Editor is available from main menu (see Figure B.5 in Appendix B). From the editor a short-cut to creation of a module can be placed on one of the three toolbars. Also, a description of the module can be provided in HTML format for future reference.
5. 'Mirror pane' provides a quick access to modules scattered across the workspace.

More detailed information on implementation of the GUI for data processing environment and API for customisation of the modules' looks and feels is available in the technical documentation provided with the data processing environment.

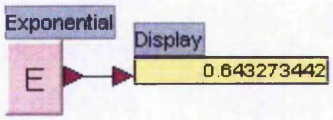
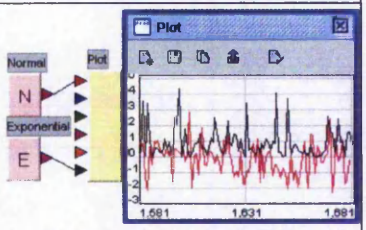
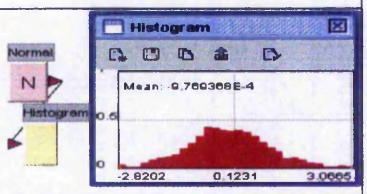
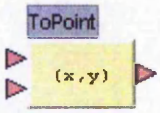
4.7 Examples

Several functional modules have been implemented in order to demonstrate the potential of the developed software. These modules are split into two groups – the general purpose modules and the task-specific modules. These two classes will be overviewed separately.

4.7.1 General purpose modules

Modules that are implemented into package `edu.tntu.ism.traffic.modules` are general purpose modules. The list of the modules is given in the following table.

Table 4.5. General purpose modules: visualisation

Module name (class name)	Description	Visual appearance
`Display`	`Display` implements its own look that is used to print into the incoming numerical data. It has a fixed (1) number of inputs and therefore can be used to monitor only one stream of numerical data	 Display prints the values of exponential random variable
`Plot`	`Plot` draws a plot of the incoming data. It can have up to 20 inputs, which will be drawn on the `Plot`'s display.	 Plot draws exponential and normal random variables
`Histogram`	`Histogram` is similar to Plot but instead of plot of the input data it draws a histogram. It has only one input.	 Histogram of a normal random variables
`ToPoint`	This module converts two streams of numerical data into one stream of two dimensional vectors, that is two inputs x and y are transformed into a pair (x,y). An application can be seen in the simulation of Lissajou figures shown in Figure B.2 of Appendix B.	
`Min`	Receives several streams of numerical data and returns the minimal value among them on its output	
`Max`	Same as `Min` but returns the maximal value among the inputs.	

For demonstrational purposes, in Appendix C, Example 1 presents source code of Display module. In Appendix B, there are shown several models that extensively use the above standard modules to visualise the undergoing processes in the model.

4.7.2 Task-specific modules

In order to show, that the data processing environment offers functionality that is not poorer than the reviewed commercial products for dynamical system modelling, a set of modules for continuous numerical system modelling and simulation has been developed into package `edu.tntu.ism.traffic.modules.math`.

Simulation of continuous systems on a computer always involves a discrete approximation of continuous time. Therefore the modules must operate in a synchronous manner. Commercial packages, such as VisSim and SIMULINK, are essentially systems designed for a synchronous mode of operation.

By contrast, the developed data processing environment developed here, is an asynchronous system. In order to implement a synchronous model, a mechanism must exist to synchronise the modules with one another. Such mechanism has been implemented into the continuous time system simulation package in the form of a module `Controller`. `Controller` does not have inputs or outputs and serve exclusively to synchronise other modules of the package.

The package also provides several abstract classes that implement most of the intermediate functionality. A new module can be created by just extending the abstract class and implementing its function. In Appendix C, Example 2 demonstrates implementation of a module by extension of an abstract class.

The other implemented modules of the package for continuous systems modelling and simulation are: `Const` (a constant), `Derivative`, `Integral`, `SystemTime`, `Gain` (a coefficient), `Sum` (summation), `Product`, `Ratio`, `Sin` (sine), `Exp` (exponent), `Exponential` (exponential random variable), `Normal` (normal random variable), `MathPlot` (extended plot that scales X-axis according to the model's time).

Figure B.1 of Appendix B presents an example of a continuous time system. The system in the figure demonstrates the well-known Fourier transform of the discrete function. Figure B.2 presents the mixed model that uses modules from the continuous time package and general-purpose modules, to simulate and visualise the Lissajou figures. Figure B.3 presents the use of histogram and random variable modules.

4.8 Evaluation of the efficiency

Since one of the requirements that has been set upon the data processing environment is efficiency, it is necessary to test if it is efficient enough to be used for real-time data processing and simulation processes.

Several models have been implemented for the data processing environment using the modules packages described in the previous sections. Also several traffic related models, which will be described in the following section, have been evaluated. The screenshots of the models are given in Appendix B.

A set of tests have been carried out on two PC-compatible machines with Windows 98 system installed. The first PC, PC-1, is AMD Athlon 1GHz processor bases with 256MB of RAM and the second PC, PC-2, is Intel Pentium-II 333 MHz processor based with 32MB of RAM. Two groups of tests have been carried out. The modules used in the first test group were the mathematical simulation models: 'discrete approximation' (Appendix B, Figure B.1) – M-1, the 'Lissajou figures' (Figure B.2) – M-2, 'random variables and histograms' (Figure B.3) – M-3. The second group of tests have evaluated the performance of the system for SCOOT data processing. The model used SCOOT data processing software (which will be discussed in the next chapter) and used two file readers to read files with M14 and M19 messages collected over one full day. The model will be referred to as M-4. The SCOOT data processing model consists of a file reader and SCOOT message pre-processor. The results of the performance evaluation obtained for the first test group is given in the following Table 4.6.

Table 4.6. Results of performance tests for the first group of models

Model	PC-1 with graphics	PC-1 without graphics	PC-2 with graphics	PC-2 without graphics
M-1	55 cps[1]	1698 cps	58 cps	442 cps
M-2	33 cps	1110 cps	38 cps	491 cps
M-3	786 cps	1046 cps	233 cps	293 cps

[1] cps stands for Cycles Per Second (a cycle is one iteration of the simulation process)

The results of the SCOOT processing model evaluation is given in the following

Table 4.7. Results of performance tests for SCOOT data processing model

Model	PC-1 with graphics	PC-1 without graphics	PC-2 with graphics	PC-2 without graphics
M-4	1254 ops[1]	2798 ops	683 ops	757 ops

[1] ops stands for Objects Per Second – object here is a SCOOT data object

The volume of the two files (with one day M14 and M19 messages, collected on 17th January, 1999) is 244,339,391 bytes. Such volume of data has been processed within 1392 seconds, or 23 minutes 12 seconds, with some overhead induced by graphical output for monitoring purposes. During the test, the processing modules have generated over 1.75 million data objects (1.6 million M14 related and 150 thousand M19 related objects). Without graphical output the numbers are more impressive. According to the above Table 4.7, 2798 objects per second has been the ratio of generation of data during the processing. The volume of the two data files has been processed in less than 10 minutes 24 second (624 seconds) which is less than half of the time needed for run with graphical output. A similar data processing routine written in the C language has completed the same processing procedure in 4 minutes. This gives the basis to a claim that the performance of a Java application is quite comparable to a programme of equal functionality written in the C language.

It is clear from the above two tables that graphical output, as might be expected, induces a significant overhead, while performance of the system is quite high without it. Since one day traffic data has been processed in about 10 minutes, the rest of the 23 hours 50 minutes time slice gives a broad room for variety of applications, which will

still be operating in real time. Of course, if a SCOOT-controlled region under consideration gets bigger, these numbers will be different.

4.9 Conclusions

In this chapter, a novel data processing framework for discrete and continuous systems modelling and simulation has been presented. The software has been shown to provide highly extensible environment and has several distinguishing features that make it choice number one for researchers who need to develop their own functional modules and carry out extensive experiments combining different options of data sources and pre-processing routines. It has also been shown to be efficient enough for most traffic and other modelling and simulation applications.

Chapter 6 will present series of experimental studies carried out using the developed software.

Chapter 5
Travel time estimation

5.1 Introduction

The importance of processing and analysis of real data has been emphasised in the previous chapters. Data processing and analysis techniques have been the main tool of investigation of transportation systems. Since transportation systems belong to the class of systems of organised complexity, precise analytical modelling of them is almost impossible. Therefore the majority of models of transportation systems are of empirical character. Empirical models essentially rely on data analysis and represent findings of relationships within the data.

One of the important problems recognised within the transportation research is the problem of travel time estimation from routinely collected data. Travel time is one of the most important operational characteristics of transportation systems and plays an important part in many criteria of transportation system optimisation. Knowledge of travel time, its behaviour and factors that affect it gives traffic control and travel information systems the opportunity to manage traffic demand and supply factors more efficiently. Therefore significant attention has been given to investigation of these traffic and travel characteristics.

Many traffic control strategies implemented into UTMC systems rely heavily on the efficiency and accuracy of travel time estimation. Travel time information is a fundamental parameter in many intelligent traffic and travel information systems and their success highly depends on the accuracy, availability and granularity of travel time estimates. Traditional survey-based travel time estimations are notoriously expensive and inefficient and the applicability of surveys is restricted only to off-line monitoring and analysis. With the development and installation of new generation of Traffic Control Systems that operate on the basis of real-time traffic measurements, it has become possible to carry out travel time estimations using the measurements taken by the control systems. Most of the systems, due to economical reasons, utilise single-loop detectors. It is known, that single loop detectors are incapable of accurate spot speeds estimation due

to their physical limitations. Therefore, much effort has been put into investigation and development of methodologies of travel time estimation that rely only on occupancy measurements, as is the case with single-loop detectors.

In Chapter 2, three fundamental macroscopic characteristics - speed, volume and density, have been identified. It has also been mentioned that of these three characteristics volume is the easiest to measure whereas speed is more difficult. The situation gets worse when there is no equipment available that is capable of measuring speed with adequate accuracy. It is known that double-loop detectors, called "speed traps" are capable of accurate measuring spot speed of individual vehicles, but unfortunately most of the installed traffic control systems are equipped with single loop detectors whose speed estimates are very rough. In addition, spot speed, even measured accurately by double loop detectors, does not necessarily give adequate measure of space speed which is used in the fundamental traffic flow equation. In the case, when speed cannot be measured directly one essentially arrives to the problem of estimation of travel time between consecutive single-loop detectors. The travel time found in this study can then be used to estimate speed.

The apparent importance of travel time estimation led to a strong interest of researchers in solving the problem. Significant progress has been made during the last two decades and the main results of the development of the field will be reviewed in the following sections.

This chapter will review the advances in solution of the problem of travel time estimation from automatically collected data and describe new algorithms that have been developed within this research study. A novel methodology for travel time estimation at the minimal resolution level will be presented and a discussion of several algorithms will be given.

The first part of this chapter gives a definition of travel time that will be used throughout the chapter and presents a brief overview on previous works carried out in the area of travel time estimation. The second part will develop a novel methodology that allows very accurate estimation of travel times and without the need for data aggregation that is used in most procedures developed up to date.

5.2 Travel time estimation problem

Travel time is defined here as the time that is required for a vehicle to traverse between two consecutive measuring points through a part of road infrastructure in condition of non-stopping journey. This general definition does not specify the conditions of the traffic or network topology thus allowing to speak of freeway travel time, urban link travel time or any other types in the same terms.

Basically, as the definition states, there must be some information about traffic processes provided at the entrance point, which will be referred to as upstream point, and exit point, which will be referred to as downstream point. It is normally assumed that the upstream and downstream measuring devices are capable of measuring presence of a vehicle at a moment of time thus providing temporal characteristic of arrival process. This capability forms the basis for most of the estimation procedures. Having such a device is also follows that it can act as a counter. Since counters have been used in many traffic studies, it is also a desired feature.

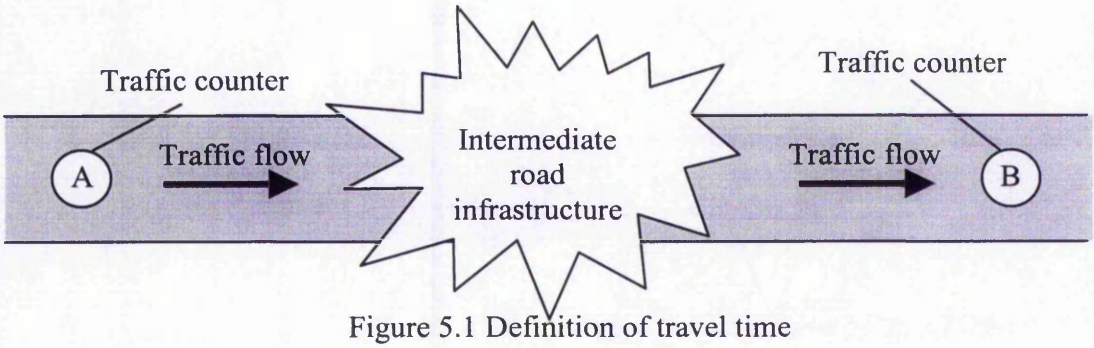


Figure 5.1 Definition of travel time

Figure 5.1 illustrates the definition of travel time. Basically, there may be several paths that lead from point A to point B, which is shown on the figure as intermediate infrastructure. Since travel time is essentially a characteristic of a certain path, the definition becomes ambiguous. In order to overcome this ambiguity, an additional condition has to be added to the definition, namely which path will be considered for the travel time analysis. Therefore, here and thereafter, unless stated otherwise, the shortest (in length) path from point A to point B will be considered for the analysis and be called a link. It is implicitly assumed that the link exists, that is traffic can proceed from A to B. As a result of a link being the shortest path from A to B, most traffic will use the link as

the natural behaviour of drivers. It is nevertheless not mandatory that traffic at point A is the same as at the point B since vehicles are allowed to leave and join the link in-between the points.

5.3 Review of the existing methodologies

Early attempts to estimate link travel time have used the general traffic flow relationship in order to estimate speed [Wardrop, 1952], [Lighthill, Whitham, 1955], [Wirasinghe, 1978], [Hall, 1987], [Hall and Persaud, 1988], [Persaud and Hurdle, 1988], which in turn allows calculation of the travel time since the link length is a constant parameter. The relationship used is

$$speed = \frac{flow}{occupancy \cdot g} \tag{5.1}$$

where $1/g$ is the average effective car length – the sum of the car length and the width of the loop detector in the correct units. The factor g basically converts occupancy to density. In [Hall, Persaud, 1989] and [Pushkar, et al., 1994] the authors investigated the relationship (5.1) and showed that the accuracy of the equation (5.1) is a function of many factors including location and weather. They also provided results suggesting that it is prone to a systematic bias with respect to occupancy.

[Dailey, 1997] attempted to improve the speed estimates by taking into account the stochastic nature of measurements of volume and occupancy. He considers measurements from a traffic management system to be realisations from statistical distributions. To address the variability of the observations he employs extended Kalman filter. Using the filter Dailey derived estimates of the effective car length and speed and then combined them in order to obtain the travel time estimate. The link travel time is derived as a polynomial of the following form.

$$\tau_{i+1} = 2\Delta x\left[\frac{1}{s_{i+1}+s_i} + \frac{(\Delta s_i)^2}{3}\left(\frac{1}{s_{i+1}+s_i}\right)^3 + \frac{(\Delta s_i)^4}{5}\left(\frac{1}{s_{i+1}+s_i}\right)^5 + \ldots\right] \tag{5.2}$$

where τ_{i+1} is the next estimate of the link travel time, Δx is the length of the link, s_{i+1} and s_i are $(i+1)^{th}$ and i^{th} estimates of speed accordingly and $\Delta s_i = s_{i+1} - s_i$.

The disadvantage of the Dailey's filtering approach is that it requires several parameters that must be properly estimated plus the combined disadvantages inherited from the underlying traffic model based on the speed-flow-concentration relationship.

There have been attempts to improve speed estimation upon single loop detectors [Dailey, 1999], [Coifman, 2001]. Coifman's algorithm is based on explicit identification of sources of errors and suggests the way to reduce their impact. In the previous works the authors, while recognising the stochastic nature of the parameters involved in the estimation procedure, did not address the possibility that their probability distributions can change over time. Coifman presented the results of study showing that the mean vehicle length is a function of time of day and showed that variance of the vehicle length is a function of time, too. He observed that during free-flow traffic condition the estimation of the vehicle length is very noisy whereas during congested periods the estimates are more stable. Taking into account those points, Coifman derives a simple procedure that allows improved speed estimation based upon single-loop data.

As has been mentioned before, all speed estimation based algorithms inherit a set of problems from the underlying traffic model based on the fundamental traffic flow relationship such as the dependence on many factors that are difficult to measure (link location, length, weather, etc.). Moreover, in the problem of travel time estimation the estimation of speed is a side effect. Most applications are interested in travel time rather than in speed. Consequently, several methodologies that do not rely on speed estimates have been proposed.

In [Dailey, 1993] the author presents a method of estimation of the mean space speed that is independent of the mean value of the volume. In order to estimate speed of traffic flow he first estimates the mean delay time for propagation between two loops, which is essentially the mean of the link travel time. To estimate the mean of the link travel time Dailey models highway traffic as a continuous flow with some average concentration

about which there is some statistical fluctuation. The value of the concentration at a point on the highway is a function of several parameters:

1. The concentration $\alpha(x, t)$ from an upstream point (the position of an inductive loop located at x_1) that has propagated to this location (the position of an inductive loop located at x_2) at time t.
2. The change in concentration from addition or removal of vehicles [$S(x, t)$]
3. A dispersion factor that represents the lack of rigidity of the statistical fluctuation relative to the mean concentration (b)
4. Uncorrelated fluctuations and loop failure [$n(t)$]

and can be written

$$\alpha(x_2,t) = b\alpha(x_1,t+\tau_0) + \int_{x_1}^{x_2} S(x,t)dx + n(t) \tag{5.3}$$

where τ_0 is the mean time for the traffic to propagate from x_1 to x_2.

The concentration may be represented as the sum of a time-independent mean value and a time-varying fluctuation

$$\alpha(x,t) = \alpha_0 + \delta\alpha(x,t) \tag{5.4}$$

where

$$\alpha_0 = \frac{1}{2}\int_{-T}^{T} \alpha(x,t)dt \tag{5.5}$$

This representation assumes that the traffic is in steady-state flow during the period [$-T,T$], over which $\alpha(x, t)$ is being considered. If the concentration is mean centred, the fluctuating part of the downstream station is written

$$\delta\alpha(x_2,t) = b\delta\alpha(x_1,t+\tau_0) + \int_{x_1}^{x_2} \delta S(x,t)dx + \delta n(t) \tag{5.6}$$

And this is the time series of the traffic flow to be examined. In order to estimate the delay time between two loops the cross-correlation function is used. The cross-correlation function in the given context can be written

$$R_{12}(\tau,T)=\frac{1}{2T}\int_{-T}^{T}\delta\alpha(x_1,t)\delta\alpha(x_2,t-\tau)dt \tag{5.7}$$

Considering the property of the time series and the cross-correlation function, Dailey claims that the point of maximum of the cross-correlation function will correspond to the mean delay between the two series. In order to apply the method to real data he then develops the model in the discrete framework. Then he constructs discrete time series of the traffic volume by summing the occupancy counts over 5 seconds. That is the data is being aggregated by the time intervals of 5 seconds. Then the time series are being centred by constructing

$$V_{ik}=Vol_{ik}-\frac{1}{N}\sum_{j=1}^{N}Vol_{ij} \tag{5.8}$$

where Vol_{ik} is the k^{th} point in the series of length N from the i^{th} station.

The discrete cross-correlation function is then written as

$$\rho(\tau_k,T)=\frac{\sum_{j=1}^{N}V_{1j}V_{2(j+k)}}{\sigma_1\sigma_2}\qquad \text{for } \forall k\in[0,N] \tag{5.9}$$

where

$$\sigma_i=\sqrt{\frac{1}{N}\sum_{k=1}^{N}V_{ik}^2} \tag{5.10}$$

are the standard deviations of the series 1 and 2.

Finally, the link travel time can be found as

$$\tau = \arg\max \rho(\tau_k, T) \tag{5.11}$$

The disadvantages of the cross-correlation approach are apparent. While cross-correlation can lead to accurate estimation of the mean of travel time, it does not provide the information on other characteristics of travel time as random variable. Second disadvantage is the need for aggregation of traffic counts in order to obtain the desired time series, which can lead to the loss of information and worsen the accuracy of the algorithm for certain traffic conditions. Third point discussed in [Petty et al., 1998] is that aggregation level of 5 seconds can be too big for some applications. This, for example, makes the algorithm inherently inaccurate in the situations where the mean travel time is as little as 15 seconds, which is the case with the urban links.

In [Petty et al, 1998] the authors developed a more general model of traffic flow which has certain relationships to the Dailey's cross-correlation approach. Their methodology for estimating travel times between single loop detectors is based on a stochastic model, which suggests practical procedures that lead to accurate travel time estimates and is briefly reviewed below.

The authors assume that during a given interval of time the travel times may be regarded as drawn from the same probability distribution. They then estimate that distribution from the cumulative upstream and downstream arrival processes.

The arrival point process is considered over a time interval $[T_B, T_F]$ and the arrivals are assumed to be exchangeable, that is no distinction is made upon different types of vehicles. The cumulative number of upstream arrivals is denoted by $X(t)$ and the cumulative number of downstream arrivals by $Y(t)$. Denoting the upstream arrival times by σ_i and the travel times by τ_j and summation for all possible σ_i yields

$$dX(t) = \sum_i \delta(t - \sigma_i)dt \tag{5.12}$$

$$dY(t) = \sum_j \delta(t - \sigma_j - \tau_j)dt \tag{5.13}$$

where $\delta(\cdot)$ is Dirac's delta function. The model postulates that conditional on σ_i τ_j has a distribution independent of i, σ_i for $T_B \le \sigma_i \le T_F$. It is following from the assumption that no change of traffic regime is allowed during the time interval under consideration and that the probability distribution does not change with time.

Provided that $f(\cdot)$ denotes the marginal density of τ_i under those assumptions, $p(\cdot)$ denotes the conditional density of σ_i given $T_B \le \sigma_i \le T_F$ and $q(\cdot)$ denotes the density of $\sigma_i+\tau_i$ the arrival time at the downstream detector given $T_B \le \sigma_i \le T_F$. If T_B =-∞ then conditional on X(·), the expected process of downstream arrivals satisfies

$$\begin{aligned} E[dY(t) \mid X] &= \int_{-\infty}^{t} \sum_j \delta(t-\sigma_j-\tau_j) f(\tau_j) d\tau_j dt \\ &= \sum_j f(t-\sigma_j) dt \\ &= \left(\int_{-\infty}^{t} f(t-v) dX(v) \right) dt \end{aligned} \tag{5.14}$$

where $E(\cdot)$ is the expectation operator.

However, T_B=-∞ is unrealisable. On the other hand, if vehicles arrive before T_B, equation (5.14) does not hold without further requirement. The most natural condition is that $0<a \le \tau_i \le b$ which implies that f=0 is outside $[a, b]$. The interval $[a, b]$ will be referred to as the fit window. Following further, for $T_B+b \le t \le T_F+a$, equation (5.14) holds since -∞ can be replaced with T_B. Then (5.14) can be rewritten as

$$q(t) = \int_{T_B}^{t} f(t-v) p(v) dv \qquad \text{for } T_B+b \le t \le T_F+a \tag{5.15}$$

To apply the method to real situation, the processes need to be aggregated in discrete units of length Δ. If $f(\cdot)$, $p(\cdot)$, $q(\cdot)$ are approximated by discrete mass functions f_s, p_s, q_s, where q_s is the mass in the interval [sΔ, (s+1) Δ], the discrete approximation of (5.15) can be written as

$$q_t = \sum_{v=\frac{T_B}{\Delta}}^{\frac{t}{\Delta}} f_{t-v} p_v \tag{5.16}$$

This in turn leads to a natural estimation scheme in which q_t and p_v are replaced with the counters y_t, x_v of arrivals at the downstream and upstream detectors in consecutive intervals and apply a natural measure of discrepancy, least squares, to fit equation (5.16). That is, the estimation of f_s is accomplished by minimising

$$\sum_{t=\frac{T+b}{\Delta}}^{\frac{T_F+a}{\Delta}-1} \left(y_t - \sum_{s=\frac{a}{\Delta}}^{\frac{b}{\Delta}-1} x_{t-s} f_s \right)^2 \rightarrow \min \tag{5.17}$$

over $\{f: f_s \geq 0, \Delta \Sigma f_s = 1\}$.

The equation (5.17) forms the basis of [Petty et al., 1998] scheme. Having estimated the probability distribution f_s the authors proposed to use the mode as the estimation of the travel time as the mean is more sensitive to disturbances.

Several difficulties associated with the above methodology have been recognised by the authors. First essential difficulty is the choice of the parameters a and b. Having carried out empirical study of the effect of choosing different a and b they found that an adaptive choice is required for generally satisfactory results in both congested and uncongested periods. The width of the fit window b-a can be taken as fairly small and fixed since the range of speed of vehicles during homogeneous regimes is not great. But the centre of the window, $(a+b)/2$, has to move. The authors found that using equation (5.1) with a reasonable value of g can work well. Second difficulty is the choice of the 'stationary' periods $[T_B, T_F]$. It is clear that the larger the interval the better precision of the scheme. However, the larger the interval the bigger the risk of getting a transition in regime during this period. The authors state that it is possible to detect the transitions in regime and choose the interval according to the available information. They do not provide the methodology but state that the measure of density can be used as an indicator of the transitions in traffic regime. Third problem is associated with the choice of appropriate

aggregation level Δ. The authors provide a thorough empirical study of the effects of different levels of aggregation on the results of the scheme. Since this problem is important for many other methods and algorithms a brief review will follow.

In order to investigate the effect of aggregation on the accuracy of travel time estimation, the authors use double-loop speed estimates to compare against the estimates produced by the scheme. They define the average distance L between these two estimates as a function of aggregation level Δ:

$$L(\Delta) = \frac{1}{N}\sum_{i=1}^{N} | \hat{\tau}_i^{\Delta} - \tilde{\tau}_i | \tag{5.18}$$

By comparing the values of the above function, the authors found that the best results are given by aggregation levels of 3 seconds and 11 seconds. While 3 seconds aggregation gives results that are a bit worse than those obtained for aggregation of 11 seconds the latter level is surrounded by very poor results provided for 10 and 12 seconds aggregation levels. Later in this chapter a methodology is introduced that does not make use of aggregation and therefore is free of the above problems.

Several authors [Blue et al, 1994], [Dia,1999] proposed to use artificial neural networks as estimators of freeway travel time. The reported results have clearly demonstrated the feasibility of using neural networks for travel time estimation problem. In addition, it was found that neural networks are capable of predicting travel times up to 15 minutes into the future with a high degree of accuracy (93%-95%). The methodology has been tested on sites where travel times are about 5 to 10 minutes, which are too big compared to short urban links and therefore it is unclear if the same results will be obtained for urban links.

5.4 Travel time estimation in urban links

Most of the travel time estimation methods have been developed for freeway traffic. Unfortunately, urban links have several distinguishing features that do not allow direct application of the freeway methods without additional modifications. Firstly, the distance between consecutive junctions is relatively short compared to freeway links. On such a link, vehicle's travel time can be as little as 15 seconds, which makes the methods that

rely on aggregated traffic counters inherently inaccurate. On urban networks one normally deals with sums of many link travel times in order to obtain the full journey travel time and therefore the estimation of individual link travel times must be as accurate as possible. This renders the aggregation-based algorithms an inappropriate choice. Secondly, an urban link normally has several unobserved (no traffic measurements are taken) side-roads, which allow vehicles that entered the link leave the flow unnoticed (no information on their leaving available) or new vehicles join the flow without being counted on the entrance, thus making the distributions of counters on the entrance and the exit of the link different. The second feature of urban links violates the assumptions of the methods that rely on comparison of the entering and exiting distributions, which therefore cannot be used without modifications that take the unmeasured leaving and joining of vehicles into account.

Taking into consideration all above points, a new methodology for link travel time estimation has been developed within this research study. Formal mathematical justification of the methodology in its general form is complicated due to the complex nature of traffic processes. Nevertheless for a simple traffic model it has become possible to carry out rigorous analytical calculations that prove the correctness of the scheme for that model. The study and the results are presented in the following section.

5.4.1 The urban link model

A typical urban link is shown in Figure 5.2. The main feature that distinguishes a typical urban link from freeway links is the presence of a controlled junction in-between the detectors.

The control at the junction is performed in order to allow different traffic flow to use a single resource – the piece of land that belongs to more than one road (the junction). The effect of the control is such that the traffic measured at the point A and the traffic measured at the point B may be completely different and uncorrelated. The situation becomes even worse as at the junction part of traffic from the origin A can turn away from the road and not be measured at the destination B and some traffic from the side

road can join the link *A-B* and be measured at point *B*. These features make the link travel estimation problem a challenge.

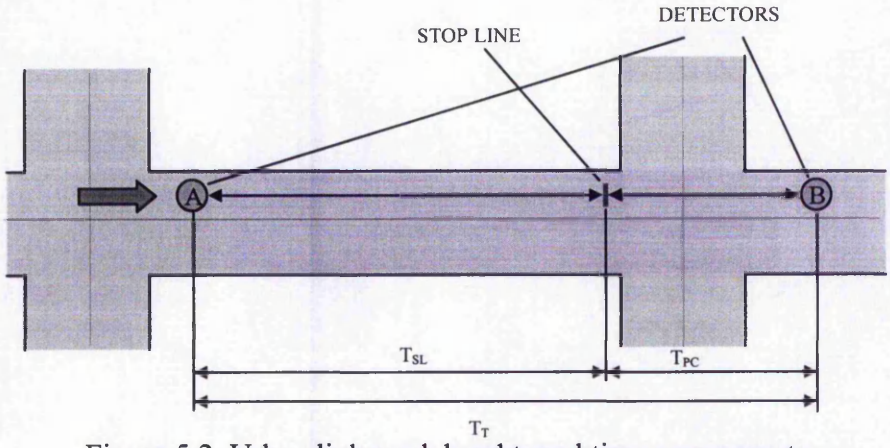


Figure 5.2. Urban link model and travel time components

It is also clear from the figure that total travel time (T_T) on an urban link can be split into two components, which are T_{SL} (SL stands for Stop Line) – travel time from the upstream measurement point up to the stop line and T_{PC} (PC stands for Passage of Crossroad) – travel time from the stop line to the downstream measurement point. The purpose of such division is that the total time may often include the delay induced by the traffic control which contradicts the definition of travel time made in Section 5.2. Therefore T_{SL} becomes the main measure of travel time in urban links. This measure basically provides the information about the link length and typical time that vehicles spend to traverse the link. It is also worth mentioning that some UTMC systems (for instance, SCOOT) use T_{SL} in order to optimise traffic and produce optimal control. The division of total travel time into two components can also be used for estimation of other important characteristics of traffic such as turning movements. This possibility will be investigated later in this thesis.

The necessity of splitting the total travel time across an urban link has been justified. Unfortunately, the components cannot be measured directly in most cases. Therefore, an estimation procedure for each component has to be developed. The estimation procedure for the total travel time is being developed and studied in the following section. In Chapter 6, an empirical study of the other components of urban link travel time will be presented together with the demonstration of the (developed in this chapter) methodology of estimation of the T_T component (see Figure 5.2).

5.4.2 The methodology of estimation of travel time with the lowest level of resolution

The main objective pursued here is the development of a methodology that is suitable for estimation of travel time in urban conditions. The specificity of urban links, as has been emphasised in the previous sections, is that travel time between nodes of a link is short compared to similar freeway links and therefore aggregation of traffic measurements is not desirable. The developed methodology allows carrying out travel time estimations with the resolution of measurement data, thus does not have limitations of previously developed techniques. Another specificity of urban link is the high probability of a vehicle leaving the link or another vehicle joining the link in-between the traffic measurement devices and therefore not being accounted at one of the nodes. This problem has not yet been addressed within the area of travel time estimation research and all developed methodologies assume that traffic flow does not change from uplink to downlink measurement points. The developed methodology is proposed as a heuristic which is intuitively appealing. Unfortunately, the methodology cannot be proved in a general case with mathematical rigour for the reasons that are discussed later in the thesis. Nevertheless, for a simplified model it has become possible to derive precise results, which gave a rationale behind the use of the methodology in its more general form in practice. This section will give a description of the developed algorithm and discuss its possible applications. The following section will present the mathematical proof of the methodology for a simplified model of a link and with certain assumptions regarding traffic flow processes. Then will follow a discussion on applicability of the simplified model used in the proof, a simulation study of the problem and examples of the results obtained for real urban traffic system.

Consider Figure 5.2 that illustrates a typical urban link. Assume that measurements that come from detectors A and B represent time of arrival of a vehicle at that particular detector. Then the set of measurements forms a time series, or a process, of arrival times. Let us denote the upstream time series by $\{A(t), t \geq 0\}$ and the downstream time series by $\{B(t), t \geq 0\}$.

It is further assumed, that traffic forms a free flow from A to B; that is it does not stop at the stop line as shown on the figure. This requirement is necessary since only free

flowing traffic carries information about the travel time between the nodes of the link since stopped traffic has a delay induced by the waiting time at the stop line. It is nevertheless possible to separate these times from the whole time of the travel with a simple pre-processing. Since in urban links traffic is normally controlled by a UTMC system, the problem of filtering the part of traffic flow that corresponds to that, which has been stopped and part that forms free flow will be addressed later.

Since time series $A(t)$ and $B(t)$ represent counting process, they are discrete and countable. Let us enumerate and denote individual arrival times of the above time series by $\{a_i\}$ and $\{b_j\}$. It is intuitively appealing that if a_i is the arrival time of a vehicle at the upstream detector and b_j is the arrival time of the same vehicle at the downstream detector, then subject to the free traffic flow condition, a_i and b_j are related through a relationship of the form

$$b_j = a_i + \varepsilon \tag{5.19}$$

where ε is a stochastic component that represents the delay between arrival of a vehicle at upstream and downstream points and takes into account variation in the speeds of the vehicle and fluctuation of speed with time. It also seems rational that for a short distance, as is the case with urban links, that variable ε be considered to have a symmetrical bell-shaped distribution, such as normal. Then ε can well be characterised by its mean and standard deviation: $\varepsilon = \varepsilon (T_T, \sigma)$. Then the problem of travel time estimation essentially reduces to the problem of estimation of the parameters of random variable $\varepsilon (T_T, \sigma)$. Another point to be mentioned is that vehicles in real traffic tend to keep microscopic structure of the flow over significant periods of time; that is the headways between consecutive vehicles change insignificantly. This feature of traffic lies behind the success of the developed methodology.

Arrival times of downstream process are first being shifted into the past by a time τ; that is the following linear transformation is performed.

$$b^*_j = b_j - \tau \tag{5.20}$$

The choice of parameter τ will be discussed later. Then the upstream time series is combined with the modified downstream time series in a way, that keeps their order. The following Figure 5.3 illustrates the above idea.

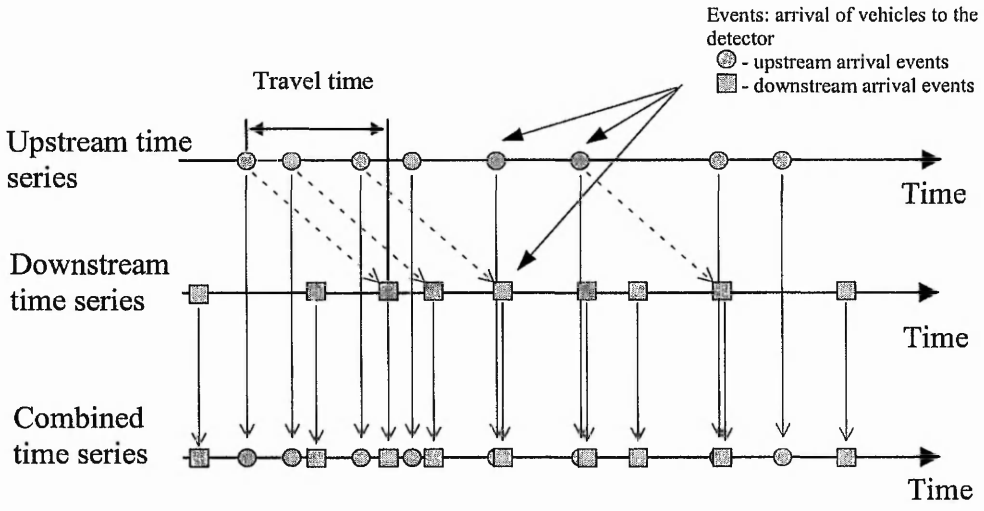


Figure 5.3. Merging upstream and shifted downstream time series

The dashed arrows on the figure point to the related events, that is linking the upstream arrival events to the corresponding downstream arrival events. Next step is, using the combined process, to extract pairs of neighbouring arrival times that are from different sources, that is upstream arrival time is associated with the closest in time downstream arrival. The following Figure 5.4 illustrates the pair extraction procedure.

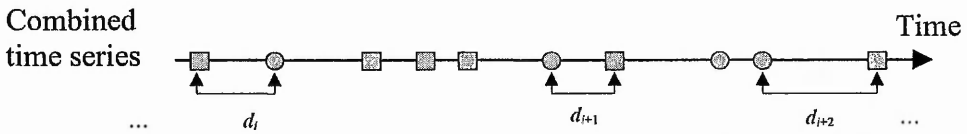


Figure 5.4. Extracting pairs of neighbouring events that come from different time series

As a result, algorithm produces series of pairs of events $P_k=\{a_i, b_j\}$ where a_i is the arrival time of the upstream event and b_j is the arrival time of the downstream event that form the pair. Several methods of extracting pairs P_k from the combined time series can be suggested. The most effective method would be to extract the pairs in such way, that the sum of corresponding time differences d_k is minimal among all possible choices of pairs.

Although this method will yield the result with higher accuracy, it is not efficient. Complexity of the optimal algorithm is $O(N^2)$, where N is the number of elements in the sample of combined time series. A method that chooses pairs using local optimisation has been tested and showed that its accuracy is close to the optimal algorithm but has complexity of $O(N)$, that is it is much more efficient. An algorithm with local optimisation is described below.

After the upstream and downstream times series have been combined into the resulting time series, a step of the algorithm is given as follows. Assume that current position of an event in the resulting time series is k. Then there are six possible configurations of events, characterised by the event arrival times, as illustrated in Figure 5.5.

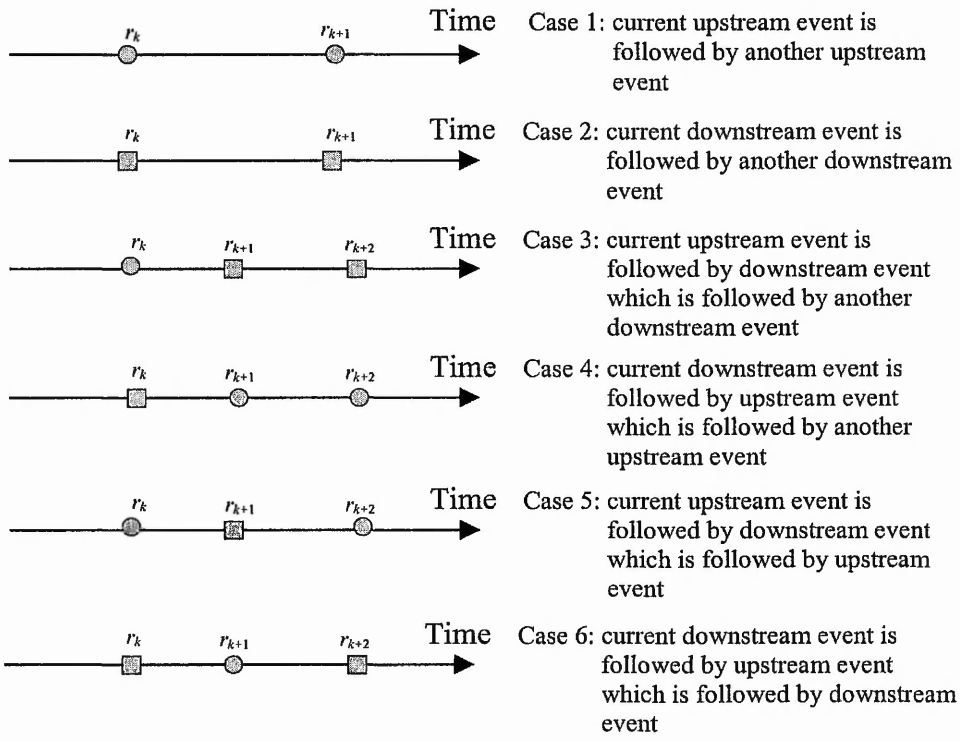


Figure 5.5. Combinations of events ahead of the current event at k

In the cases 1 and 2 above, the algorithm does not produce pairs since the current event at k cannot be associated with the following event since they have the same origin. If case 1 or case 2 is encountered at step k, the algorithm makes event k+1 its current and repeats the procedure. In cases 3 and 4 there are clear pairs formed by events at k and k+1, since event at k+1 cannot form a pair with event at k+2 because of their same origin. Thus, if

case 3 or 4 is encountered, a pair of events at k and k+1 is formed and the algorithm makes its current event for the next cycle the event at k+2. In cases 5 and 6 it is not clear which pair is to be taken, the one formed by events at k and k+1 or events at k+1 and k+2. In these cases, the algorithm forms pairs of such events, whose difference is smaller. That is if r_{k+1}-r_k < r_{k+2}-r_{k+1} then algorithm produces a pair formed by events k and k+1, and advances to k+2 as its current event for the next cycle and produces a pair formed by k+1 and k+2 otherwise, advancing to k+3 as its current event for the next cycle.

The above algorithm ensures that if an event is associated with a pair, it is associated with that pair only. It also performs a local minimisation of the time differences associated with pairs as shown above for cases 5 and 6. It is also clear that the algorithm's complexity is O(N) where N is the number of events in the resulting (merger) time series.

Once a set of pairs P_k has been obtained, an average of time differences between the events of a pair is calculated. It is calculated for currently applied time shift τ and thus being considered a function of τ. Let us denote the arrival time of an upstream event that forms part of pair k by a^k and the arrival time of a downstream event that forms part of the same pair k by b^k. Then

$$F_N(\tau) = \frac{1}{N}\sum_{k=1}^{N}\left|a^k - b^k\right| \tag{5.21}$$

where N is the number of pairs.

All pairs selected by the algorithm can be split into two classes – the ones that are formed by independent upstream and downstream events and those formed by the events that are related through relationship (5.19). Figure 5.6 below illustrates this concept. Assume that the set of independent pairs is denoted by Ω_i and the set of dependent pairs is denoted by Ω_d. Then their cardinalities N_i=$|\Omega_i|$ and N_d=$|\Omega_d|$ satisfy N=N_i+N_d and (5.21) can then be written as the following sum

$$F_N(\tau) = \frac{1}{N}\left(\sum_{k:P_k\in\Omega_i}^{N_i}\left|a^k - b^k\right| + \sum_{l:P_l\in\Omega_d}^{N_d}\left|a^l - b^l\right|\right) =$$

$$\frac{N_i}{N}\left(\frac{1}{N_i}\sum_{k:P_k\in\Omega_i}^{N_i}\left|a^k-b^k\right|\right)+\frac{N_d}{N}\left(\frac{1}{N_d}\sum_{l:P_l\in\Omega_d}^{N_d}\left|a^l-b^l\right|\right)=$$

$$q(\tau)(\overline{X_i}(\tau)+(1-q(\tau))\overline{X_d}(\tau);\quad 0\le q(\tau)\le 1 \tag{5.22}$$

where $\overline{X_i}(\tau)$ is the average of time differences $\{d_i\}$ for independent pairs and $\overline{X_d}(\tau)$ is the average of time differences $\{d_j\}$ for dependent pairs and $q(\tau)=N_i/N$.

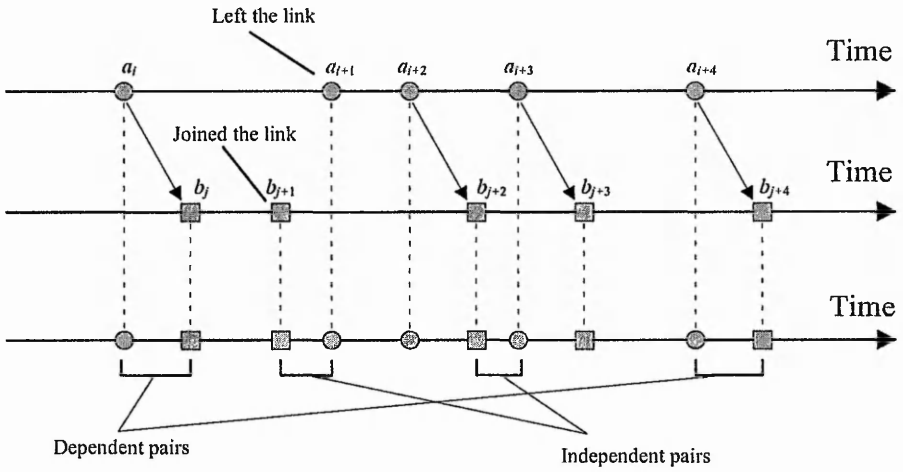


Figure 5.6. Illustration of the concept of dependent and independent pairs

It is difficult to express explicitly the relationship between differences $\{d_i\}$ produced by independent pairs and time shift τ but it can be anticipated that such a relationship will be weaker (or more irregular) than that of differences produced by dependent events. Using (5.19) and (5.20), the relationship between dependent difference d_j and τ can be expressed as

$$d_j(\tau)=b_j^*-a_i=b_j-\tau-a_i=\varepsilon(T_T,\sigma)-\tau \tag{5.23}$$

The properties of this relationship will further be investigated. First of all, let us calculate the expectation of $|d_j(\tau)|$ as a random variable

$$\mathrm{E}\,|\,\varepsilon(T_T,\sigma)-\tau\,|=\int_{-\infty}^{+\infty}|\,x-\tau\,|\,f_\varepsilon(x)dx=\int_{-\infty}^{\tau}(x-\tau)f_\varepsilon(x)dx-\int_{\tau}^{+\infty}(x-\tau)f_\varepsilon(x)dx=$$

$$-2\int_{-\infty}^{\tau} x f_{\varepsilon}(x)dx + \mathrm{E}_{\varepsilon} + \tau(2F_{\varepsilon}(\tau)-1) = \overline{X_d}(\tau) \tag{5.24}$$

In order to obtain the extrema of the function (5.24), its derivative is to be taken

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\overline{X_d}(\tau) = -2\frac{\mathrm{d}}{\mathrm{d}\tau}\int_{-\infty}^{\tau} x f_{\varepsilon}(x)dx + \frac{\mathrm{d}}{\mathrm{d}\tau}\left[2\tau F_{\varepsilon}(\tau) - \tau\right] =$$

$$-2\tau f_{\varepsilon}(\tau) + 2F_{\varepsilon}(\tau) + 2\tau f_{\varepsilon}(\tau) - 1 = 2F_{\varepsilon}(\tau) - 1 \tag{5.25}$$

In the point of extremum the derivative is zero

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\overline{X_d}(\tau) = 0 \quad \Rightarrow \quad 2F_{\varepsilon}(\tau) - 1 = 0\,, \text{ or}$$

$$F_{\varepsilon}(\tau) = \frac{1}{2} \tag{5.26}$$

It is clear from (5.24) that $\overline{X_d}(\tau) \to +\infty$ whenever $\tau \to \pm\infty$, therefore the found extremum is minimum. Consequently, it follows from (5.26) that $\overline{X_d}(\tau)$ reaches its minimum whenever τ is the median of $\varepsilon\,(T_T,\, \sigma)$. If $\varepsilon\,(T_T,\, \sigma)$ is assumed to be normally distributed then its median equals its mode and equals its mean T_T. Therefore, if the mean of independent differences does not change significantly with τ, then it is reasonable to expect (5.22) to reach its minimum in the point $\tau = T_T$.

The above argumentation can be considered being a heuristic, as it cannot be formally proven in its general form. Nevertheless, it is strongly appealing and a number of experiments with real traffic data, which will be presented in Chapter 6, have shown it to be correct. Another point for the correctness of this approach is that when τ approaches T_T, coefficient (1-$q(\tau)$) in (5.22) tends to grow while $q(\tau)$ itself decreases subject to that σ of $\varepsilon(T_T,\, \sigma)$ is smaller than average time gap between consecutive arrival events. This, in turn, makes the second term of the sum (5.22) to have stronger influence on the whole sum and thus minimum in the point $\tau = T_T$ is being stronger emphasised.

Consider now the problem of choosing τ on each step of the algorithm. It is obvious that a scheme of choosing a step of shift on each iteration of the algorithm will depend on the algorithm of pair extraction used. Nevertheless, a general idea of the step-choosing scheme is the same and will be given here.

As has been mentioned before, the algorithm produces series of pairs of arrival times of events that come from different sources, $P_k=\{a_i, b^*_j\}$. There are three possible configurations of event arrival times a_i and b^*_j relative to each other, namely b^*_j follows a_i, b^*_j matches (equals) a_i and a_i follows b^*_j as illustrated in Figure 5.7.

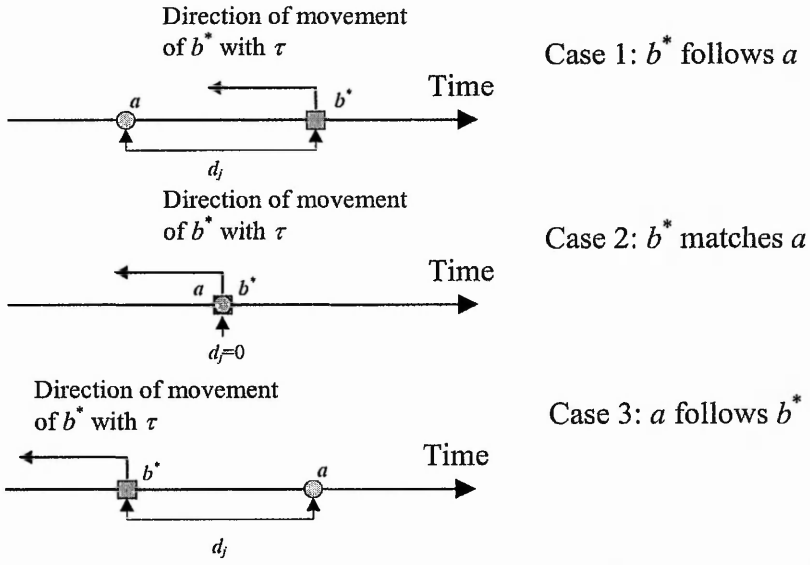


Figure 5.7. Architectures of pair of event arrival times extracted by the algorithm

Then it is clear from Figure 5.7 that there are two classes of pairs that behave differently with changing of τ. Arrival time b^* of the pairs of the first class (case 1 on the figure) will advance towards a linearly with the growth of τ whereas arrival time b^* of the second class (case 2 and case 3) will recede from a also linearly with the growth of τ. Since pairs representing the second case in the picture can be seen as special case of either case 1 or case 3 and it will be associated with the second class of pairs. Thus it can be assumed that whenever b^* of the pair of the first class reaches a, this pair moves from the first class of pairs to the second class of pairs. Assume now that N_1 is the total number of pairs of the first class and N_2 is the total number of pairs of the second class.

Then for a given τ (5.21) can be expresses in the terms of these two classes of pairs as follows

$$F_N(\tau) = \frac{1}{N}\sum_{k=1}^{N}\left|a^k - b^k\right| =$$

$$\frac{1}{N}\left(\sum_{k\in\Omega_1}^{N_1}\left(b^*_{j_k} - a_{i_k}\right) + \sum_{k\in\Omega_2}^{N_2}\left(a_{i_k} - b^*_{j_k}\right)\right) \tag{5.27}$$

where Ω_1 is a set of indices of pairs that belong to the first class as specified above, Ω_2 is a set of indices of pairs that belong to the second class and i_k, j_k are indices of corresponding arrival times of individual time series. It is important to mention that process of transferring of pairs from class 2 to class 1 is continuous. In order to clarify this statement let us illustrate the transfer process. Assume that upstream and downstream time series consist of only one event arrival time, such as shown on Figure 5.7. Then using (5.20), (5.27) will take the following form

$$F(\tau) = \left| b^* - a \right| = \left| b - \tau - a \right| \tag{5.28}$$

Then it is clear that the above function is continuous piece-wise linear with respect to τ.

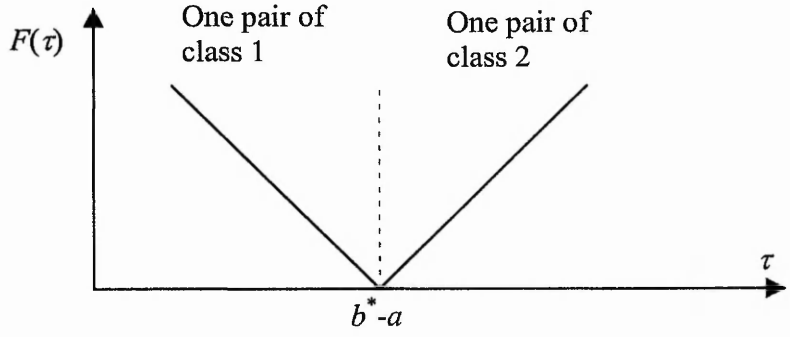


Figure 5.8. Plot of the function (5.27) when there is only one event in each of the time series

It is also clear that a life cycle of a pair from class 1 is to move to class 2. From class 2 pairs disappear, that is the individual arrival times are being recombined and new pair(s) are formed. It is also clear that once the algorithm has performed one of its cycles and formed a set of pairs, then with continuous movement of τ new pairs are born into the

class 1. Therefore, the process of movement of τ (as shown in Figure 5.7) is associated with the process of forming new pairs into class 1, transferring pairs from class 1 to class 2 and destruction (recombination of the forming events) of pairs of class 2.

Assume that the algorithm has produced the set of pairs. Let pair $P_k=\{a^1, b^{*1}\}$ belong to class 1 and its corresponding time difference $|d_k|>0$ and pair $P_m=\{a^2, b^{*2}\}$ belong to class 2 and its corresponding time difference d_m. It is clear, if one moves τ by a small (such that does not affect class property of the pairs P_k and P_m) value $\Delta\tau>0$, the differences will change in such a way that the change induced by moving τ by $\Delta\tau$ will be compensated by both pairs. That is, if for a given $\tau=\tau_1$ $|d_k(\tau_1)|=x_1>0$ then $|d_k(\tau_1+\Delta\tau)|=x_1-\Delta\tau$ (since $x_1>0$ $\Delta\tau$ can always be chosen such, that $x_1-\Delta\tau>0$). Analogically, if for the same $\tau=\tau_1$ $|d_m(\tau_1)|=x_2\geq 0$ then $|d_m(\tau_1+\Delta\tau)|=x_2+\Delta\tau$. Then since these two differences are summed using (5.21), these two terms appear in the sum as

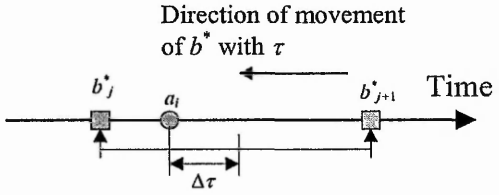
$$\begin{aligned}\Sigma=&\ldots+|d_k(\tau_1+\Delta\tau)|+\ldots+|d_m(\tau_1+\Delta\tau)|+\ldots=\\&\ldots+x_1-\Delta\tau+\ldots+x_2+\Delta\tau+\ldots=\\&\ldots+x_1+\ldots+x_2+\ldots=\\&\ldots+|d_k(\tau_1)|+\ldots+|d_m(\tau_1)|+\ldots=\end{aligned}\qquad(5.29)$$

It is clear from (5.29) that with changing of τ (by appropriately chosen $\Delta\tau>0$) the sum of modules of the two differences does not change. Now let us get back to definitions used in (5.27). It is also clear from (5.29) that the total sum can change with τ (provided that the structure of classes does not change) if and only if $N_1\neq N_2$, that is the pairs of class 1 cannot compensate or be compensated by the pairs of class 2. If $N_1>N_2$, then the sum (5.27) rises by $(N_1-N_2)\Delta\tau$ and $N_1<N_2$, then the sum declines by $(N_2-N_1)\Delta\tau$ for any $\Delta\tau>0$ such that preserve the pairs in their classes. Consequently, it is clear that the sum (5.27) is linear on the intervals where structure of classes of pairs does not change. Let us now investigate the conditions when the structure of the two classes changes. By the change of the structure of the classes will be understood a process of migration pairs from class 1 to class 2 and destruction, and as a consequence formation of new pairs into class 1, of pairs of class 2.

It is obvious that pairs from class 1 migrate into class 2 for such $\Delta\tau$ that makes the corresponding difference equal to zero. Then as it is stated above, if $N_1 > N_2$ then the average (5.27) will decrease until classes change. And since the process of migration is continuous the minimum of the average (5.27) will be in the point when one of the pairs from class1 migrates to class 2. For this case, the advance of τ, $\Delta\tau$, can be chosen such that makes differences of one (or more) pair from class 1 equal to zero, or in other words $\Delta\tau$ can be chosen to be the minimal difference d_i of pairs of class 1.

$$\Delta\tau^1 = \min | d_i | \quad \forall d_i = a_{k_i} - b^*_{l_i}, i \in \Omega_1 \tag{5.30}$$

Second situation when the structure of classes of pairs change is when a pair from class 2 is being destroyed. The situations are illustrated below.



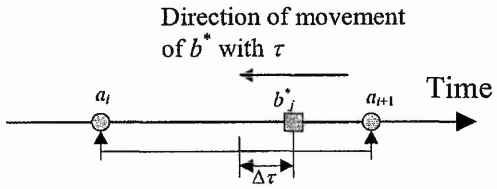
In the case above a pair (a_i, b^*_j) of class 2 will disappear and new pair (a_i, b^*_{j+1}) of class 1 will be formed, if the time shift is advanced as shown in the picture. Among all situations as in the above picture, $\Delta\tau$ must be taken minimal non-zero. For the above illustration, $\Delta\tau$ can be found as

$$\Delta\tau = \frac{b^*_j + b^*_{j+1}}{2} - a_i \tag{5.31}$$

Then if all $\Delta\tau$ that correspond to configuration above are enumerated by index l, the final $\Delta\tau$ for this class of configurations is taken as

$$\Delta\tau^2 = \min \Delta\tau_l \quad \forall l \tag{5.32}$$

Another situation which leads to a destruction of a pair from class 2 is shown in the illustration below.



In the figure above if τ is changed by $\Delta\tau$, then the pair (a_{i+1}, b^*_j) of class 2 will be destructed and a new pair (a_i, b^*_j) of class 1 formed. In order to make the above situation realise, $\Delta\tau$ must be chosen as follows

$$\Delta\tau = b^*_j - \frac{a_i + a_{i+1}}{2} \tag{5.33}$$

Then considering all such structures, minimal $\Delta\tau$ among them must be taken as the increment of τ. Let us denote the minimal $\Delta\tau$ among those that are produced by the above situation by $\Delta\tau^3$. Then summarising the cases that lead to changes of the classes' structure, the minimal among $\Delta\tau^1$, $\Delta\tau^2$ and $\Delta\tau^3$ should be taken in order for the algorithm to encounter all changes in behaviour of the resulting sum (5.27).

$$\Delta\tau_{\text{final}} = \min\{\Delta\tau^1, \Delta\tau^2, \Delta\tau^3\} \tag{5.34}$$

If the next shift is chosen according to above equation (5.34), the algorithm is ensured to discover the global minimum of the cost function (5.27). Nevertheless, there is one serious drawback in the above scheme, namely the step $\Delta\tau_{\text{final}}$ tends to get smaller with the growth of the number of events in the upstream and downstream time series. As a limit, when the upstream and downstream time series are considered infinite, $\Delta\tau_{\text{final}}$ becomes equal to zero. It might therefore be better to use the above scheme of choosing $\Delta\tau$ in cases, when a neighbourhood of the minimum has been identified and exact value of it is sought.

5.4.3 Simplified traffic model and mathematical justification of the travel time estimation methodology

In order to prove that the proposed methodology indeed gives the solution, a simplified model will be formally investigated. The model of the considered link is shown on Figure

5.9. This model simplifies the traditional urban link in a way that there are no control devices installed between the measuring points, and thus traffic is uncontrolled. Nevertheless the model is realistic in a sense that it allows vehicles to leave the link and new vehicles are allowed to join the link at the roundabout located between the detectors. It is worth noticing that there is no additional information available besides the two series of measurements that come from upstream counter A and downstream counter B.

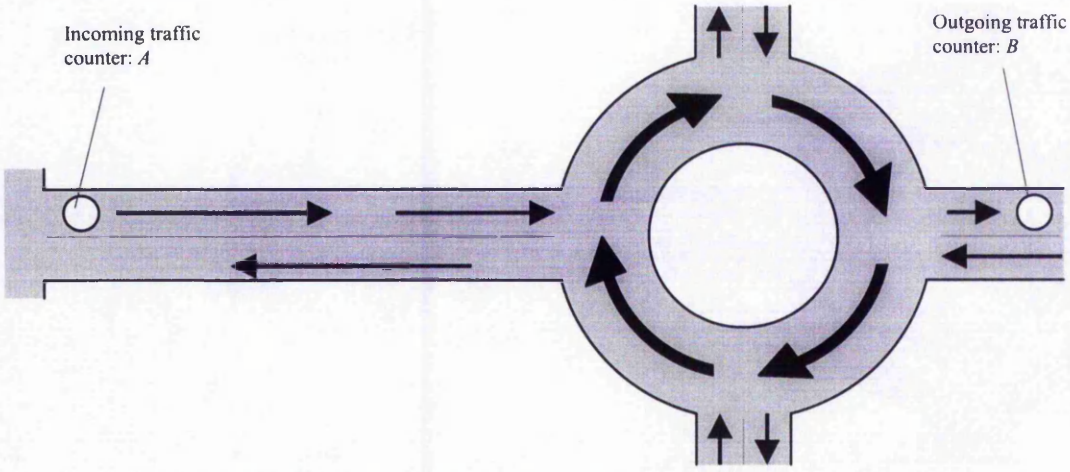


Figure 5.9. The link model: a roundabout between two measuring points A and B allows vehicles to leave and join traffic flow

The model also assumes that traffic flows can be adequately modelled by a homogeneous Poisson process. The applicability of such a model will be discussed later. Another simplification that is necessary to make is that the travel time between the measurement points is constant, that is if a vehicle intends to travel from A to B then it does not move repeatedly in the roundabout and the mean space speed is a constant (spot speed can change). The need for such assumption will be clear later when the mathematical formulation of the processes will be made.

Let us assume that traffic at the counter A forms an arrival process that is the homogeneous Poisson process $N(t)$ with parameter λ and traffic at the counter B forms an arrival process that is the homogeneous Poisson process $M(t)$ with parameter μ. Let us denote arrival times at the counter A by a_i and arrival times at the counter B by b_j. It should be noted here that the above variables and parameters are observed from the traffic counters and thus considered to be available.

It is assumed that the flow of leaving vehicles and the flow of joining vehicles form two independent Poisson processes $N_1(t)$ and $M_1(t)$ with parameters λ_1 and μ_1 accordingly. Then part of the upstream flow that moves to the downstream point forms a Poisson process $L_u(t)$ that is independent of $N_1(t)$ and $M_1(t)$. Let us denote its parameter by α. The part of downstream process that has originated from the upstream will be denoted by $L_d(t)$. Further, the upstream arrival events that correspond to process $L_u(t)$ will incur downstream arrival events with a certain constant delay (travel time). That idea can be expressed in the form of arrival times as follows. Provided travel time T is constant

$$\forall a_i \in L_u(t) \quad \exists b_j \in L_d(t): \; b_j = a_i + T \quad \text{and also} \tag{5.35}$$
$$\forall b_k \in L_d(t) \quad \exists a_l \in L_u(t): \; b_k = a_l + T$$

where a_i or a_l is an event's arrival time that belongs to $L_u(t)$ and b_j or b_k is an event's arrival time that belongs to $L_d(t)$. It then follows that process $L_d(t)$ is $L_u(t)$ shifted in time by a constant T, or $L_d(t) = L_u(t\text{-}T)$. The following results will be required for further analysis.

Lemma 1. Shifting a homogeneous Poisson process $N(t)$ with parameter λ in time by a constant τ does not affect its probabilistic properties.

Proof.

The result of Lemma 1 follows directly from the well-known property of the Poisson process, namely if $\{N(t), t\geq 0\}$ is a homogeneous Poisson process with parameter λ, than process defined as

$$N_\tau(t) = N(\tau + t) - N(\tau)$$

is also a homogeneous Poisson process with parameter λ.

Lemma 2. Given a merger $S(t)$ of two independent homogeneous Poisson processes $N(t)$ with parameter λ and $M(t)$ with parameter μ, that is $S(t) = N(t) + M(t)$. Than an event

randomly chosen from process $S(t)$ will originate from $N(t)$ with probability $\frac{\lambda}{\lambda+\mu}$ and originate from M(t) with probability $\frac{\mu}{\lambda+\mu}$.

Proof.

First of all, let us note that the sum of two independent homogeneous Poisson processes with densities λ and μ, results in homogeneous Poisson process with density $\lambda+\mu$. [Feller, 1968], [Karlin et al, 1975].

Let us choose a random point $t\geq 0$, then the event of process $S(t)$ that follows point t will be called a *randomly chosen* event of process $S(t)$. Consider processes $N(t)$ and $M(t)$ on the same time axis (Figure 5.10).

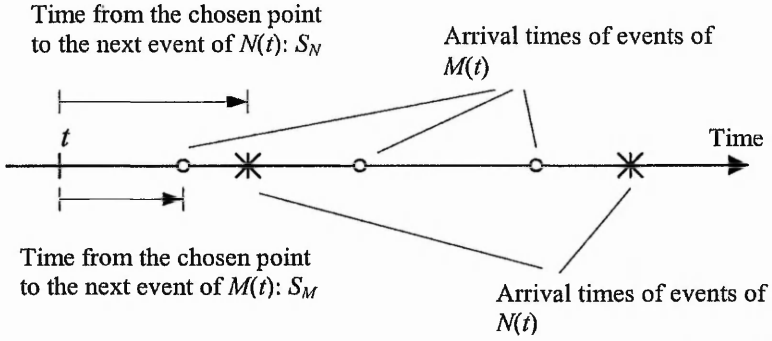


Figure 5.10. Exponentially distributed random variables S_N and S_M compete to be closer to Time point t.

Since interarrival times of processes $N(t)$ and $M(t)$ are distributed according to exponential distribution with parameters λ and μ accordingly and following the memoryless property of exponential distribution random variables S_M and S_N, as shown on the illustration, are distributed according to exponential distribution with parameters λ and μ accordingly. Then provided $t\geq 0$ is chosen randomly, the following event arrival time of process $S(t)$ will correspond to either of the two events of processes $N(t)$ and $M(t)$ that happen after t. Let Q_1 stand for the event that the arrival following t will correspond to $N(t)$ and Q_2 otherwise. Thus, the probability that the first arrival after time t will correspond to $N(t)$ will be equal to probability that the minimum of S_N and S_M equals S_N, or

$$P\{Q_1\} = P\{ \min(S_N, S_M) = S_N \}$$

and

$$P\{Q_2\} = P\{ \min(S_N, S_M) = S_M \} = 1 - P\{Q_1\}$$

The probability $P\{ \min(S_N, S_M) = S_N \}$ can be written in the form of convolution of two independent and exponentially distributed random variables S_N and S_M as

$$P\{Q_1\} = P\{ \min(S_N, S_M)=S_N \} = P\{ S_N<S_M \} = P\{ S_N-S_M<0 \}=$$

$$\int_0^{+\infty} \lambda e^{-\lambda t} \int_t^{+\infty} \mu e^{-\mu x} dx dt = \int_0^{+\infty} \lambda e^{-(\lambda+\mu)t} dt = \frac{\lambda}{\lambda+\mu}$$

and

$$P\{Q_2\} = 1 - P\{Q_1\}=\frac{\mu}{\lambda+\mu}$$

Which completes the proof.

The observable processes $N(t)$ and $M(t)$ can be expressed in terms of the component processes as

$$N(t)= N_1(t)+L_u(t)$$
$$M(t)= M_1(t)+L_d(t)$$

where arrival times of processes $L_u(t)$ and $L_d(t)$ are subject to the conditions (5.35).

Thus, the observable upstream and downstream processes can be considered as being mergers of two independent processes. Figure 5.11 illustrates this idea.

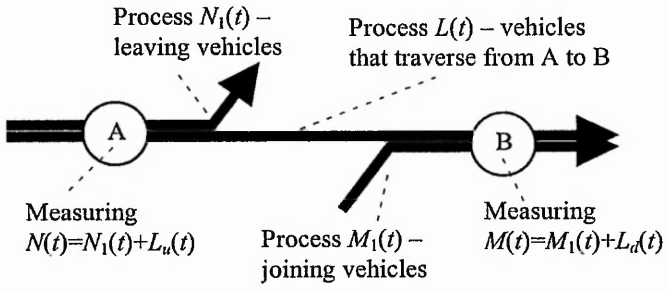


Figure 5.11. Structure of the considered traffic processes

Following from the description, the problem of travel time estimation can be formulated as follows. Given two homogeneous processes $N(t)$ and $M(t)$, and taking into account the above assumption, find the constant T. From the previously described methodology, the algorithmic procedure used to estimate travel time T is as follows.

Algorithm 1.

The downstream process $M(t)$ is being shifted by a time τ. Then upstream and the shifted downstream processes are merged into $R(t, \tau)$

$$R(t,\tau)=N(t)+M(t-\tau) \tag{5.36}$$

One condition applied to the merging procedure is that if the arrival time of an upstream event equals the arrival time of a downstream event, then the upstream arrival time is given a preference and is placed in front of the downstream arrival time. $N(t)$ and $M(t-\tau)$ are Poisson processes but the compound process $R(t)$ is obviously not a Poisson process since the processes $N(t)$ and $M(t)$ are not independent. Let us denote the arrival times of the compound process $R(t)$ by $r_i(\tau)$. In order to distinguish arrival times $r_i(\tau)$ by their original process ($N(t)$ or $M(t)$), let us introduce the indicator function $I(r_i(\tau))$ as

$$I(r_i(\tau))=\begin{cases}0 & \text{if } r_i(\tau) \text{ originates from process } N(t)\\ 1 & \text{if } r_i(\tau) \text{ originates from process } M(t)\end{cases} \tag{5.37}$$

The algorithm extracts series of interarrival times $r_{i+1} - r_i$ as follows.

1. k=1; i=1; Initialisation
2. While $I(r_i(\tau))$=1; i=i+1; Looking for an event from the upstream process
3. if $I(r_{i+1}(\tau))$=0; i=i+1; go to 2; Skip this event if next is not from downstream
4. $\Delta_k(\tau)= r_{i+1}(\tau) - r_i(\tau)$; Save the found interarrival time at k-s position
   k=k+1; i=i+1; go to 2;

As a result, the algorithm produces a series of interarrival times of the resulting process with the feature that they are produced by arrival times of both the upstream and downstream processes. The procedure above is infinite but in the real world it can be stopped when either processes $N(t)$ and $M(t)$ are exhausted, or necessary precision is reached. The main result can then be formulated as the following theorem.

Theorem 1.

Consider the scheme of processes as shown in Figure 5.11. The processes $N_1(t)$, $M_1(t)$, $L_u(t)$ and $L_d(t)$ are Poisson processes with parameters λ_1, μ_1, α and α respectively and parameters of $N(t)= N_1(t)+L_u(t)$ and $M(t)= M_1(t)+L_d(t)$, λ and μ, are related as follows

$$\lambda=\lambda_1+\alpha$$
$$\mu=\mu_1+\alpha$$

Assume that $\lambda>0$ and $\mu>0$ and consider the average of the values $\Delta_k(\tau)$ (a sample of interarrival times of process $R(t,\tau)$) produced by Algorithm 1 and assume that N of such values have been produced. The average is given as

$$F_N(\tau)=\frac{1}{N}\sum_{k=1}^{N}\Delta_k(\tau) \tag{5.38}$$

If N increases infinitely we have

$$F^*(\tau)=\lim_{N\to\infty}F_N(\tau) \tag{5.39}$$

Then, $F_N(\tau)$ converges to a finite limit $F^*(\tau)$ in probability and the travel time T can be found as

$$T=\arg\min F^*(\tau) \tag{5.40}$$

Proof

Assume that $T\text{-}\tau \geq 0$. In order to find the analytical form of function $F^*(\tau)$, it is required to investigate the properties of the interarrival times extracted by the algorithm.

During the first step the algorithm shifts the downstream process by a constant time τ. Lemma 1 states that shifted processes do not change their probabilistic properties unless the shifted process is compared to another dependent process. Therefore, the shift does not have effect on independent processes and this property will be exercised in the proof. The second step of the algorithm is to find an event that is originated from upstream flow. Since processes $L_u(t)$ and $N_1(t)$ that form the observed upstream process $N(t)$ are independent it follows from Lemma 2 that the probability for the algorithm to pick an event that corresponds to part $L_u(t)$ of $N(t)$ is $\frac{\alpha}{\alpha+\lambda_1}=\frac{\alpha}{\lambda}$ and the probability that the next event of upstream process corresponds to part $N_1(t)$ is $\frac{\lambda_1}{\alpha+\lambda_1}=\frac{\lambda_1}{\lambda}$.

Once the algorithm has chosen the next event from the upstream process, it checks if the following event of the merged process $R(t)$ corresponds to the downstream process. There are three possibilities for the next event of process $R(t)$:

1. Next event originates from upstream.
2. Next event originates from downstream from its $L_d(t)$ part
3. Next event originates from downstream from its $M_1(t)$ part

Figure 5.12 illustrates the above outcomes. Let us introduce an extended indicator function in order to distinguish the above combinations.

$$I(r_i(\tau))=\begin{cases}0 & \text{if } r_i(\tau) \text{ originates from process } N_1(t)\\ 1 & \text{if } r_i(\tau) \text{ originates from process } L_u(t)\\ 2 & \text{if } r_i(\tau) \text{ originates from process } M_1(t)\\ 3 & \text{if } r_i(\tau) \text{ originates from process } L_d(t)\end{cases} \tag{5.41}$$

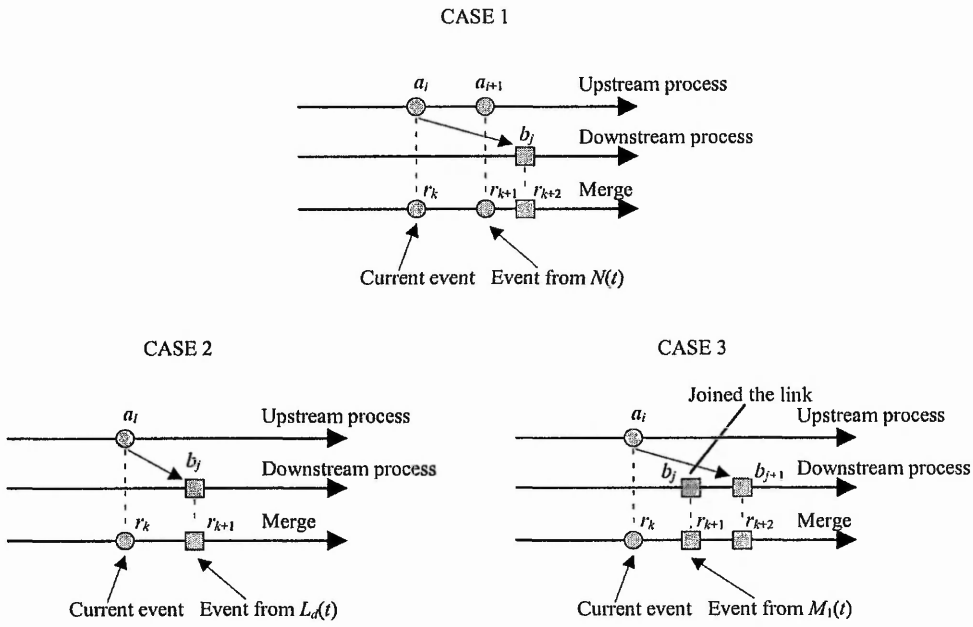


Figure 5.12. Illustration of possible outcomes for the next event after currently chosen one

Thus, the algorithm basically deals with 6 types of pairs of events. Let us represent type of a pair $(r_i(\tau), r_{i+1}(\tau))$ of consequent events of process $R(t)$ by the pair of their indicator functions $(I(r_i(\tau)), I(r_{i+1}(\tau)))$. Then the types of pairs the algorithm deals with are given in the following table

Table 5.1. Types of pairs classified by the values of indicator functions

Type №	Pair of indicator functions	Comment
1	(0, 2)	First event is from $N_1(t)$ and second event is from $M_1(t)$
2	(0, 3)	First event is from $N_1(t)$ and second event is from $L_d(t)$
3	(0, 0), (0, 1)	First event is from $N_1(t)$ and second event is from $N(t)$
4	(1, 2)	First event is from $L_u(t)$ and second event is from $M_1(t)$
5	(1, 3)	First event is from $L_u(t)$ and second event is from $L_d(t)$
6	(1, 0), (1, 1)	First event is from $L_u(t)$ and second event is from $N(t)$

Pairs of type 3 are not distinguished as they are of no interest (skipped by the algorithm). Same applies for pairs of type 6.

The purpose of such distinction of the pairs is that the corresponding time differences form random variables of different types and in order to derive the properties of the sum of them, they need to be investigated separately.

Let us denote an event that the algorithm has chosen an event from $N_1(t)$ by Q_1 and by Q_2 an event that the algorithm has chosen an event from $L_u(t)$. It has already been noted that

$$P\{Q_1\}=\frac{\lambda_1}{\lambda} \quad \text{and} \quad P\{Q_2\}=\frac{\alpha}{\lambda} \tag{5.42}$$

It follows from the independence of processes $N_1(t)$ and $L_u(t)$ that events Q_1 and Q_2 are independent. Assume that event Q_1 has taken place and calculate probabilities for the algorithm to encounter pairs of types 1, 2 and 3.

The following figure is helpful in visualising the undergoing procedure. The circles on the time axis represent events of the merged process and the numbers within the circles show the value of indicator function (the source process) of the event.

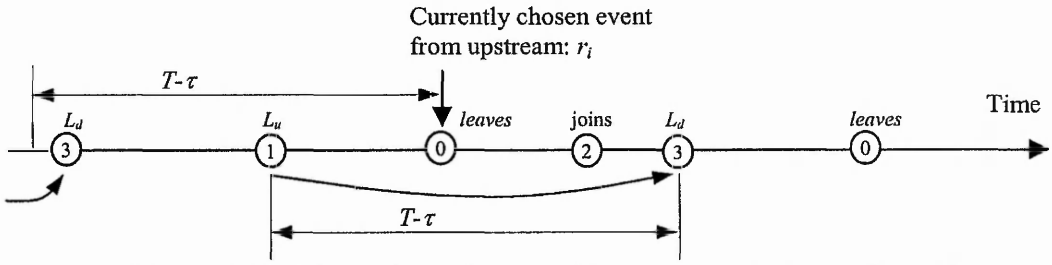


Figure 5.13. Illustration of the resulting process and procedure of choosing the pair of events

On the figure above, arrows emphasise the dependency between co-related events from $L_u(t)$ and $L_d(t)$. The distance between corresponding events from $L_u(t)$ and $L_d(t)$ is T-τ due to the shift τ introduced by the algorithm.

Given the above assumption, the probability that the event following the currently chosen one will be from $M_1(t)$ (indicator function equals 2) can then be calculated as

$$P\{ I(r_{i+1})=2 \mid Q_1 \} = \int_0^{T-\tau} \mu_1 e^{-\mu_1 t} \int_t^{+\infty} (\alpha+\alpha+\lambda_1) e^{-(\alpha+\alpha+\lambda_1)x} dx dt +$$

$$e^{-\alpha(T-\tau)} \int_{T-\tau}^{+\infty} \mu_1 e^{-\mu_1 t} \int_t^{+\infty} (\alpha+\lambda_1) e^{-(\alpha+\lambda_1)x} dx dt \qquad (5.43)$$

The first component of the above sum represents the fact that on the interval $[0, T-\tau]$ four independent exponentially distributed random variables are active. The process $L_d(t)$ has effect at the distance from r_i that does not exceed $T-\tau$, since its generating event from $L_u(t)$ has to be before r_i and not further from r_i then $T-\tau$. The probability of the event that an event originated from $L_u(t)$ lies within the interval $[r_i-(T-\tau), r_i]$ equals $1-e^{-\alpha(T-\tau)}$ (following from properties of exponentially distributed random variable) and thus the second component of the sum (5.43) represent the case when it does not happen. In the case if $L_u(t)$ does not lie within the interval $[r_i-(T-\tau), r_i]$, there is no process $L_d(t)$ acting and the second component of the sum represents this fact by omitting one alpha-parameter (that corresponds to process $L_d(t)$). Bearing in mind that $\lambda=\alpha+\lambda_1$ and $\mu=\alpha+\mu_1$, equation (5.43) can then be calculated as

$$P\{ I(r_{i+1})=2 \mid Q_1 \} = \int_0^{T-\tau} \mu_1 e^{-(\mu+\lambda)t} dt + e^{-\alpha(T-\tau)} \int_{T-\tau}^{+\infty} \mu_1 e^{-(\lambda+\mu_1)t} dt =$$

$$\frac{\mu_1}{\lambda+\mu}(1-e^{-(\lambda+\mu)(T-\tau)}) + e^{-\alpha(T-\tau)} \frac{\mu_1}{\lambda+\mu_1} e^{-(\lambda+\mu_1)(T-\tau)} =$$

$$\frac{\mu_1}{\lambda+\mu}\left(1+\frac{\alpha}{\lambda+\mu_1} e^{-(\lambda+\mu)(T-\tau)}\right) \qquad (5.44)$$

Using the same arguments, the following probability for pairs of type 3 can be calculated

$$P\{ I(r_{i+1})=0 \text{ or } I(r_{i+1})=1 \mid Q_1 \} = \int_0^{T-\tau} \lambda e^{-\lambda t} \int_t^{+\infty} (\alpha+\mu_1) e^{-(\alpha+\mu_1)x} dx dt +$$

$$e^{-\alpha(T-\tau)} \int_{T-\tau}^{+\infty} \lambda e^{-\lambda t} \int_t^{+\infty} \mu_1 e^{-\mu_1 x} dx dt =$$

$$\frac{\lambda}{\lambda+\mu}\left(1+\frac{\alpha}{\lambda+\mu_1} e^{-(\lambda+\mu)(T-\tau)}\right) \qquad (5.45)$$

The probability of an event of choosing a pair of type 2 is different in that that the effect of process $L_d(t)$ lies in the interval $[r_i, r_i+(T-\tau)]$ only due to the previously discussed reasons. Then,

$$P\{ I(r_{i+1})=3 \mid Q_1 \} = \int_0^{T-\tau} \alpha e^{-\alpha t} \int_t^{+\infty} (\alpha + \lambda_1 + \mu_1) e^{-(\alpha+\lambda_1+\mu_1)x} dx dt =$$

$$\frac{\alpha}{\lambda+\mu}(1-e^{-(\lambda+\mu)(T-\tau)}) \tag{5.46}$$

Equations (5.44), (5.45), (5.46) form a partition of the space that consist of outcomes of the experiment of trying the type of the next event that follows the chosen r_i. It can be easily checked that

$$P\{ I(r_{i+1})=2 \mid Q_1 \} + P\{ I(r_{i+1})=3 \mid Q_1 \} + P\{ I(r_{i+1})=0 \text{ or } I(r_{i+1})=1 \mid Q_1 \} = 1$$

as required.

Now, assume that Q_2 has taken place while the algorithm was choosing the upstream event. Then together with the following event r_{i+1} the pair can be of types 4, 5 or 6 as given in the table. Let us calculate the probabilities of the above types.

Consider the outcome that leads to a pair of type 4. Since the current upstream even r_i originates from process $L_u(t)$, it is known that within the interval of length $T-\tau$ there is a related event originated from $L_d(t)$. Therefore r_{i+1} are of the $M_1(t)$ process if and only if they happen before the expected event from $L_d(t)$. It is also known that on the interval $[r_i, r_i+(T-\tau)]$ events from $L_u(t)$, $N_1(t)$ and uncorrelated $L_d(t)$ can take place. Then the probability that the outcome is of type 4 will be

$$P\{ I(r_{i+1})=2 \mid Q_2 \} = \int_0^{T-\tau} \mu_1 e^{-\mu_1 t} \int_t^{+\infty} (\alpha + \lambda_1 + \alpha) e^{-(\alpha+\lambda_1+\alpha)x} dx dt =$$

$$\int_0^{T-\tau} \mu_1 e^{-(\mu_1+\alpha+\lambda_1+\alpha)t} dt = \frac{\mu_1}{\lambda+\mu}(1-e^{-(\lambda+\mu)(T-\tau)}) \tag{5.47}$$

Now consider the outcome that leads to a pair of type 5. In this case there are two possible outcomes. First one is when the events that form a pair are not related. In this case the difference is a random variable. Its distribution will be studied later. The second outcome is when the events that form the pair are related. Then their difference equates to $T-\tau$ is a constant provided that τ is fixed. Let us study the probabilities of the above outcomes.

The events r_i and r_{i+1} will be independent if $r_{i+1}-r_i<T-\tau$. Therefore, the probability of such outcome, denoted by P_1, is

$$P_1\{ I(r_{i+1})=3 \mid Q_2 \}= \int_0^{T-\tau} \alpha e^{-\alpha t} \int_t^{+\infty} (\alpha+\lambda_1+\mu_1) e^{-(\alpha+\lambda_1+\mu_1)x} dx dt =$$

$$\frac{\alpha}{\lambda+\mu}(1-e^{-(\lambda+\mu)(T-\tau)}) \qquad (5.48)$$

The probability for r_i and r_{i+1} that correspond to independent events to be such that $r_{i+1}-r_i = T-\tau$ is zero and thus can be neglected. Therefore, if no events of processes $L_u(t)$, $M_1(t)$, $N_1(t)$, $L_d(t)$ have happened in the interval $[r_i, r_i+T-\tau)$ then r_i and r_{i+1} are dependent and the probability for this event is

$$P_2\{ I(r_{i+1})=3 \mid Q_2 \}= e^{-(\lambda+\mu)(T-\tau)} \qquad (5.49)$$

Together P_1 and P_2 as defined in (5.48) and (5.49) give the total probability for the pair to be of the type 5 and is calculated as

$$P\{ I(r_{i+1})=3 \mid Q_2 \} = P_1\{ I(r_{i+1})=3 \mid Q_2 \} + P_2\{ I(r_{i+1})=3 \mid Q_2 \}=$$

$$=\frac{\alpha}{\lambda+\mu}(1-e^{-(\lambda+\mu)(T-\tau)}) + e^{-(\lambda+\mu)(T-\tau)} =$$

$$=\frac{\alpha}{\lambda+\mu} + \frac{\lambda+\mu_1}{\lambda+\mu} e^{-(\lambda+\mu)(T-\tau)} \qquad (5.50)$$

The last possible outcome for the pair is to be of type 6. In this case the events with arrival times r_i and r_{i+1} are produced by the upstream and thus the algorithm will skip over them. The probability for the pair to be of type 6 equals the probability of an exponentially distributed random variable with parameter λ (upstream) to be greater than another independent exponentially distributed random variable with parameter μ (downstream). Therefore

$$\mathrm{P}\{\, I(r_{i+1})=0 \text{ or } I(r_{i+1})=1 \mid Q_2 \,\} = \int_0^{T-\tau} \lambda e^{-\lambda t} \int_t^{+\infty} \mu e^{-\mu x}\, dx dt =$$

$$\frac{\lambda}{\lambda+\mu}\left(1-e^{-(\lambda+\mu)(T-\tau)}\right) \qquad (5.51)$$

All the above probabilities have been calculated conditional on the occurrence of Q_1 and Q_2. Then the probabilities for the algorithm to encounter a certain type of a pair in a step of its cycle are given as full probabilities

$$\mathrm{P}\{\, \mathrm{Q}_1, A \,\} = \mathrm{P}\{\mathrm{Q}_1\}\, \mathrm{P}\{\, A \mid \mathrm{Q}_1 \,\} \quad \text{and}$$
$$\mathrm{P}\{\, \mathrm{Q}_2, A \,\} = \mathrm{P}\{\mathrm{Q}_2\}\, \mathrm{P}\{\, A \mid \mathrm{Q}_2 \,\}$$

where A is one of the following outcomes $\{\, I(r_{i+1})=0$ or $I(r_{i+1})=1$, $I(r_{i+1})=2$, $I(r_{i+1})=3 \,\}$. Probabilities $\mathrm{P}\{\mathrm{Q}_1\}$ and $\mathrm{P}\{\mathrm{Q}_2\}$ have been given in (5.42).

The probabilities of the pair types that the algorithm follows through can now be calculated and are summarised in Table 5.2.

Table 5.2. Probabilities for different types of pairs to occur

Type №	Pair of indicators $(I(r_i), I(r_{i+1}))$	Full probability
1	(0, 2)	$\frac{\lambda_1}{\lambda}\frac{\mu_1}{\lambda+\mu}\left(1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}\right)$
2	(0, 3)	$\frac{\lambda_1}{\lambda}\frac{\alpha}{\lambda+\mu}\left(1-e^{-(\lambda+\mu)(T-\tau)}\right)$

3	(0, 0), (0, 1)	$\frac{\lambda_1}{\lambda}\frac{\lambda}{\lambda+\mu}\left(1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}\right)$
4	(1, 2)	$\frac{\alpha}{\lambda}\frac{\mu_1}{\lambda+\mu}(1-e^{-(\lambda+\mu)(T-\tau)})$
5	(1, 3)	Both: $\frac{\alpha}{\lambda}\left(\frac{\alpha}{\lambda+\mu}+\frac{\lambda+\mu_1}{\lambda+\mu}e^{-(\lambda+\mu)(T-\tau)}\right)$
5a		Independent: $\frac{\alpha}{\lambda}\frac{\alpha}{\lambda+\mu}\left(1-e^{-(\lambda+\mu)(T-\tau)}\right)$
5b		Dependent: $\frac{\alpha}{\lambda}e^{-(\lambda+\mu)(T-\tau)}$
6	(1, 0), (1, 1)	$\frac{\alpha}{\lambda}\frac{\lambda}{\lambda+\mu}(1-e^{-(\lambda+\mu)(T-\tau)})$

From the description of the algorithm it follows that it extracts only pairs of types 1, 2, 4 and 5 and sends the corresponding difference $r_{i+1}-r_i$ to the resulting series. Thus it is required to calculate the probability with which a certain type of differences goes into the resulting series. Let us denote by A_k, an event that the current type of a pair encountered by the algorithm is k. Then the difference of the pair will go to the resulting series if and only if one of A_1, A_2, A_4 or A_5 has taken place. Then among outcomes A_1, A_2, A_4 or A_5 the probability for the difference of type k to go into the resulting series is given by the conditional probability

$$\mathrm{P}_k = P\left\{A_k \mid \bigcup_{i\in\{1,2,4,5a,5b\}} A_i\right\} = \frac{P\{A_k\}}{P\left\{\bigcup_{i\in\{1,2,4,5a,5b\}} A_i\right\}},\ k=1,2,4,5a,5b \qquad (5.52)$$

Since the A_k's are disjoint, (5.52) can be rewritten as

$$\mathrm{P}_k = \frac{P\{A_k\}}{\sum_{i\in\{1,2,4,5a,5b\}} P\{A_i\}},\ k=1,2,4,5a,5b \qquad (5.53)$$

It can be easily calculated that

$$\sum_{i\in\{1,2,4,5a,5b\}} P\{A_i\} = \frac{\mu}{\lambda+\mu}\left(1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}\right) \tag{5.54}$$

Finally, the calculated probabilities P_k are given as

$$P_1=\frac{\lambda_1\mu_1}{\lambda\mu} \qquad P_2=\frac{\alpha\lambda_1}{\lambda\mu}\frac{1-e^{-(\lambda+\mu)(T-\tau)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}} \qquad P_4=\frac{\alpha\mu_1}{\lambda\mu}\frac{1-e^{-(\lambda+\mu)(T-\tau)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}$$

$$P_{5a}=\frac{\alpha^2}{\lambda\mu}\frac{1-e^{-(\lambda+\mu)(T-\tau)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}} \qquad P_{5b}=\frac{\alpha(\lambda+\mu)}{\lambda\mu}\frac{e^{-(\lambda+\mu)(T-\tau)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}$$

Now consider equation (5.38). It is the average of the elements of the resulting series. As has been mentioned before, the algorithm gives four types of differences as elements of the series. Let us denote these differences by Δ^k where k is its type (as in Table 5.2). The algorithm does not recognise dependent and independent differences of type 5 but they will be distinguished here and thereafter as Δ^{5a} for independent and Δ^{5b} for dependent differences. Thus, the sum (5.38) can be expanded onto a five sub-sums that contain elements only of certain type:

$$F_N(\tau)=\frac{1}{N}\sum_{k=1}^{N}\Delta_k(\tau)=\frac{1}{N}\left(\sum_{k\in\{1,2,4,5a,5b\}}\sum_{i\in\Omega_k}^{|\Omega_k|}\Delta_i^k(\tau)\right) \tag{5.55}$$

where Ω_k is a set of indices that correspond to differences Δ of type k, $k\in\{1,2,4,5a,5b\}$ upper index of the Δ-s shows the type of this difference and $|\Omega_k|=N_k$ is the cardinality of Ω_k. Further

$$F_N(\tau)=\frac{1}{N}\left(\sum_{k\in\{1,2,4,5a,5b\}}\left(\frac{N_k}{N_k}\sum_{i\in\Omega_k}^{N_k}\Delta_i^k(\tau)\right)\right)=\sum_{k\in\{1,2,4,5a,5b\}}\left(\frac{N_k}{N}\left[\frac{1}{N_k}\sum_{i\in\Omega_k}^{N_k}\Delta_i^k(\tau)\right]\right) \tag{5.56}$$

As $N\to\infty$ according to the strong law of large numbers the terms $\frac{1}{N_k}\sum_{i\in\Omega_4}^{N_4}\Delta_i^k(\tau)$ converge with probability 1 to the expectations $\mathrm{E}[\Delta^k(\tau)]$ of the random variables Δ^k, and according to the weak law of large numbers (Bernoulli's theorem), frequencies $\frac{N_k}{N}$ converge (in probability) to the probabilities Pr_k of a certain type of differences to get into the resulting series [Feller, 1968]. It is a well-known fact that if sequence X_N converges to X almost surely (with probability 1) and Y_M converges to Y in probability, then X_NY_M converges to XY in probability. Therefore $F_N(\tau)$ converges to $F^*(\tau)$ in probability and

$$F^*(\tau)=\lim_{N\to\infty}F_N(\tau)=\sum_{k\in\{1,2,4,5a,5b\}}\mathrm{Pr}_k\,\mathrm{E}\left[\Delta_i^k(\tau)\right] \tag{5.57}$$

In order to calculate the expectations of the above random variables, their properties have to be investigated.

The first type of the differences, Δ^1, is produced by r_i and r_{i+1} such that $I(r_i)=0$ and $I(r_{i+1})=2$, that is r_i and r_{i+1} correspond to the events that belong to processes $N_1(t)$ and $M_1(t)$ accordingly. These two processes are independent and also independent of $L_u(t)$ and $L_d(t)$. Let us denote the random variable whose realizations are Δ^1 by ξ_1 and calculate its probability distribution function. Assume that r_i is fixed and $I(r_i)=0$ and consider a random variable χ that represents the time from r_i to the occurrence of an event from either of the processes, that is $\chi=r_{i+1}-r_i$. Then ξ_1 can be considered as being conditional on χ such that it is produced by a pair of type 1 (or $I(r_{i+1})=2$ subject to $I(r_i)=0$). This fact will be denoted as $\chi\in(0,2)$. Formally, it can be written as follows

$$F_{\xi_1}(t)=\Pr\{\chi<t\mid\chi\in(0,2)\}=\frac{\Pr\{\chi<t\cap\chi\in(0,2)\}}{\Pr\{\chi\in(0,2)\}} \tag{5.58}$$

The dividend in the above formula represents the probability that from the given time r_i first arrival event will correspond to the process $M_1(t)$ and the distance from r_i to that event is less than t. Taking into account that process $L_d(t)$ is active only on the interval $[r_i, r_i+T-\tau]$, and does not realise on that interval with probability $e^{-\alpha(T-\tau)}$, that probability

can be calculated as follows (similar arguments to those used in derivation of equation (5.44) can be applied)

$$\Pr\{\chi < t \cap \chi \in (0,2)\} = \int_0^t \mu_1 e^{-\mu_1 x} \int_x^{+\infty} (\lambda_1 + \alpha + \alpha) e^{-(\lambda_1+\alpha+\alpha)y} dy dx =$$

$$\frac{\mu_1}{\lambda + \mu}(1 - e^{-(\lambda+\mu)t}) \qquad t \le T-\tau$$

$$\Pr\{\chi < t \cap \chi \in (0,2)\} =$$

$$\frac{\mu_1}{\lambda+\mu}(1-e^{-(\lambda+\mu)(T-\tau)}) + e^{-\alpha(T-\tau)} \int_{T-\tau}^{t} \mu_1 e^{-\mu_1 x} \int_x^{+\infty} (\lambda_1 + \alpha) e^{-(\lambda_1+\alpha)y} dy dx =$$

$$\frac{\mu_1}{\lambda+\mu}(1-e^{-(\lambda+\mu)(T-\tau)}) + e^{-\alpha(T-\tau)} \frac{\mu_1}{\lambda+\mu_1}(e^{-(\lambda+\mu_1)(T-\tau)} - e^{-(\lambda+\mu_1)t}) \quad t > T-\tau$$

(5.59)

It can be easily seen that the denominator in the equation (5.58) is the same as (5.44), that is (Q_1 represents an event that $I(r_i)=0$)

$$\Pr\{\chi \in (0,2)\} = \Pr\{I(r_{i+1}) = 2 \mid Q_1\} = \frac{\mu_1}{\lambda+\mu}\left(1 + \frac{\alpha}{\lambda+\mu_1} e^{-(\lambda+\mu)(T-\tau)}\right) \tag{5.60}$$

Finally, substitution of (5.59) and (5.60) in (5.58) yields the following probability distribution function for random variable ξ_1

$$F_{\xi_1}(t) = \begin{cases} \dfrac{1-e^{-(\lambda+\mu)t}}{1+\dfrac{\alpha}{\lambda+\mu}e^{-(\lambda+\mu)(T-\tau)}}, & t \le T-\tau \\ \dfrac{1-e^{-(\lambda+\mu)(T-\tau)}}{1+\dfrac{\alpha}{\lambda+\mu}e^{-(\lambda+\mu)(T-\tau)}} + e^{-\alpha(T-\tau)} \dfrac{(\lambda+\mu)(e^{-(\lambda+\mu_1)(T-\tau)} - e^{-(\lambda+\mu_1)t})}{(\lambda+\mu_1)\left(1+\dfrac{\alpha}{\lambda+\mu}e^{-(\lambda+\mu)(T-\tau)}\right)}, & t > T-\tau \end{cases}$$

(5.61)

In order to obtain the expectation of ξ_1, its probability density function is needed. From (5.61) the probability density function of ξ_1 is obtained as

$$f_{\xi_1}(t)=\begin{cases}\dfrac{\lambda+\mu}{1+\dfrac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}e^{-(\lambda+\mu)t} & t\in[0,T-\tau)\\ \dfrac{(\lambda+\mu)e^{-\alpha(T-\tau)}}{1+\dfrac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}e^{-(\lambda+\mu_1)t} & t\in[T-\tau,+\infty)\end{cases} \tag{5.62}$$

An illustration of the above probability density function is given in Figure 5.14. It is worth noticing that the probability density function is continuous in the point of $t=T-\tau$.

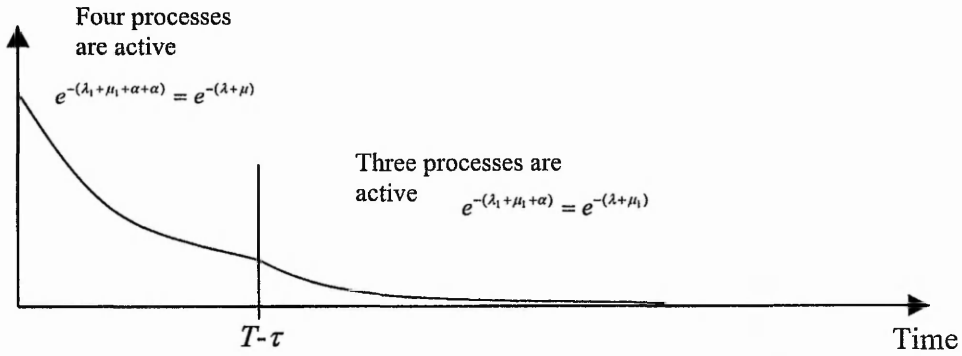


Figure 5.14. Structure of probability distribution function of random variable ξ_1

Having calculated the probability distribution function of the random variable ξ_1 it is possible to calculate its expectation using the definition

$$\mathrm{E}[\xi_1]=\int_{-\infty}^{+\infty}tf_{\xi_1}(t)dt \Rightarrow$$

$$\mathrm{E}[\xi_1]=\frac{1}{1+\dfrac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}\int_0^{T-\tau}(\lambda+\mu)te^{-(\lambda+\mu)t}dt+ \tag{5.63}$$

$$\frac{\dfrac{\lambda+\mu}{\lambda+\mu_1}e^{-\alpha(T-\tau)}}{1+\dfrac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}\int_{T-\tau}^{+\infty}(\lambda+\mu_1)te^{-(\lambda+\mu_1)t}dt$$

The integration can be accomplished using integration by parts. The following result is standard

$$\int kte^{-kt}dt = -e^{-kt}\left(t+\frac{1}{k}\right)+C, \tag{5.64}$$

where C is an arbitrary constant and $k\neq 0$.

Using (5.64), it follows from (5.63) that

$$\mathrm{E}[\xi_1] = \frac{1}{1+\dfrac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}\left(-e^{-(\lambda+\mu)t}\left(t+\frac{1}{\lambda+\mu}\right)\right|_0^{T-\tau} + \frac{\dfrac{\lambda+\mu}{\lambda+\mu_1}e^{-\alpha(T-\tau)}}{1+\dfrac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}\left(-e^{-(\lambda+\mu_1)t}\left(t+\frac{1}{\lambda+\mu_1}\right)\right|_{T-\tau}^{+\infty} \tag{5.65}$$

After elementary calculation and simplifications, the final form of the resulting expectation of ξ_1 can be found as

$$\mathrm{E}[\xi_1] = \frac{\dfrac{1}{\lambda+\mu}+\dfrac{\alpha}{\lambda+\mu_1}\left[T-\tau+\dfrac{1}{\lambda+\mu}+\dfrac{1}{\lambda+\mu_1}\right]e^{-(\lambda+\mu)(T-\tau)}}{1+\dfrac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}} \tag{5.66}$$

Now consider differences Δ^2, Δ^4, Δ^{5a}. Despite the fact that these differences are produced by events that belong to different Poisson processes, they can be considered as realisations of the same random variable that will be denoted by ξ_2. This result is implied by the fact that events, whose arrival times produce the above differences belong to either of the processes $L_u(t)$ or $L_d(t)$ (see Figure 5.13, events of types 1 and 3). For example, consider events that generate differences of the 2nd class. The indicator functions for such a pair of events are $I(r_i)=0$ and $I(r_{i+1})=3$. Due to the relationship between the processes $L_u(t)$ and $L_d(t)$, it is known, that at the time $r_{i+1}-T+\tau$ there is an event from $L_u(t)$ that has

taken place. Therefore, the event from $N_1(t)$ that happened at time r_i and produced the difference cannot be located into the past further then r_{i+1}-T+τ. From this argument it follows that all differences of the second type do not exceed the value of T-τ. Now consider time interval $[r_i;\ r_{i+1}$-T+$\tau]$. On this interval four independent exponentially distributed random variables are acting. Thus, random variable ξ_2 can be seen as being exponentially distributed but with the restriction that its values are not greater than T-τ. Therefore, its probability density function is the head of the exponential density cut at the point T-τ. The following figure illustrates such a distribution.

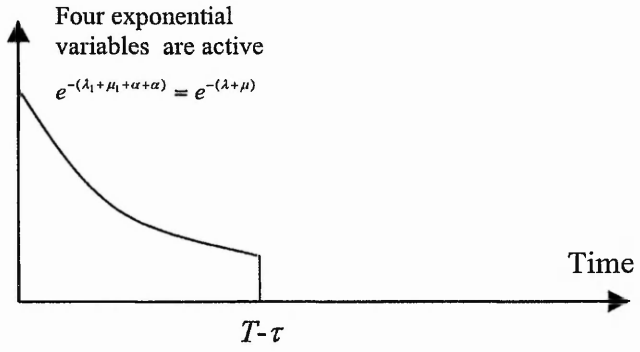


Figure 5.15. Structure of probability distribution function of variables Δ^2, Δ^4 and Δ^{5a}.

It can also be shown that random variables that produce differences Δ^4 and Δ^{5a} have the same form of distribution function. Since ξ_2 is exponentially distributed but conditional on not being greater than T-τ, it can be defined via a proper exponentially distributed random variable as having a conditional probability distribution function as follows

$$F_{\xi_2}(t) = \Pr\{\psi < t \mid \psi \le T - \tau\} \tag{5.67}$$

where ψ is an exponentially distributed random variable with parameter λ+μ.

Then, following from the definition of conditional probability

$$\Pr\{\psi < t \mid \psi \le T - \tau\} = \frac{\Pr\{\psi < \min(t, T - \tau)\}}{\Pr\{\psi < T - \tau\}} =$$

$$\frac{1-e^{-(\lambda+\mu)t}}{1-e^{-(\lambda+\mu)(T-\tau)}} \quad \forall t \in [0, T-\tau) \tag{5.68}$$

Finally, the random variable that corresponds to differences Δ^2, Δ^4, Δ^{5a} has the following probability density function

$$f_{\xi_2}(t) = \begin{cases} \dfrac{\lambda+\mu}{1-e^{-(\lambda+\mu)(T-\tau)}} e^{-(\lambda+\mu)t} & t \in [0, T-\tau) \\ 0 & \text{everywhereelse} \end{cases} \tag{5.69}$$

To calculate the expectation of ξ_2 equation (5.64) can be used. Using the definition of the expectation of a random variable and the integration formula (5.64), one obtains

$$\mathrm{E}[\xi_2] = \int_{-\infty}^{+\infty} t f_{\xi_2}(t) dt \Rightarrow$$

$$\mathrm{E}[\xi_2] = \frac{1}{1-e^{-(\lambda+\mu)(T-\tau)}} \int_0^{T-\tau} (\lambda+\mu) t e^{-(\lambda+\mu)t} dt =$$

$$\frac{1}{1-e^{-(\lambda+\mu)(T-\tau)}} \left[-e^{-(\lambda+\mu)t} \left(t + \frac{1}{\lambda+\mu} \right) \right] \Bigg|_0^{T-\tau}$$

And finally

$$\mathrm{E}[\xi_2] = \frac{\dfrac{1}{\lambda+\mu} - \left(T-\tau+\dfrac{1}{\lambda+\mu} \right) e^{-(\lambda+\mu)(T-\tau)}}{1-e^{-(\lambda+\mu)(T-\tau)}} \tag{5.70}$$

The last type of the differences left to be considered is Δ^{5b}. The differences of this type correspond to the events of the processes $L_u(t)$ and $L_d(t)$ that are bound by the relationship given in (5.35). Therefore, the arrival time r_i that corresponds to an event of process $L_u(t)$ and arrival time r_{i+1} that corresponds to the related event of process $L_d(t)$ are bound by the relationship that follows from (5.35):

$$r_{i+1} = r_i + T\text{-}\tau$$

Therefore, the differences Δ^{5b} are not random variables but a constant

$$\Delta^{5b} = r_{i+1} - r_i = T\text{-}\tau, \tag{5.71}$$

Thus, all components for the equation (5.56) have been identified. The final analytical limit form of the average $F_N(\tau)$, as given in (5.39), of the time series Δ produced by the algorithm is given as

$$F^*(\tau) = \sum_{k\in\{1,2,4,5a,5b\}} \mathrm{Pr}_k \mathrm{E}\left[\Delta_i^k(\tau)\right] = \mathrm{E}_{\xi_1}\mathrm{P}_1 + \mathrm{E}_{\xi_2}\sum_{k\in\{2,4,5a\}}\mathrm{P}_k + (T-\tau)\mathrm{P}_{5b} \tag{5.72}$$

The first component of the sum is

$$\mathrm{E}_{\xi_1}\mathrm{P}_1 = \frac{\dfrac{1}{\lambda+\mu} + \dfrac{\alpha}{\lambda+\mu_1}\left[T-\tau+\dfrac{1}{\lambda+\mu}+\dfrac{1}{\lambda+\mu_1}\right]e^{-(\lambda+\mu)(T-\tau)}}{1+\dfrac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}\frac{\lambda_1\mu_1}{\lambda\mu} \tag{5.73}$$

It can be shown, that

$$\sum_{k\in\{2,4,5a\}}\mathrm{P}_k = \frac{\alpha(\lambda+\mu_1)}{\lambda\mu}\frac{1-e^{-(\lambda+\mu)(T-\tau)}}{1+\dfrac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}$$

Therefore, the second component of the sum (5.72) is

$$\mathrm{E}_{\xi_2}\sum_{k\in\{2,4,5a\}}\mathrm{P}_k = \frac{\dfrac{1}{\lambda+\mu}-\left(T-\tau+\dfrac{1}{\lambda+\mu}\right)e^{-(\lambda+\mu)(T-\tau)}}{1-e^{-(\lambda+\mu)(T-\tau)}}\frac{\alpha(\lambda+\mu_1)}{\lambda\mu}\frac{1-e^{-(\lambda+\mu)(T-\tau)}}{1+\dfrac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}} =$$

$$\frac{\alpha(\lambda+\mu_1)}{\lambda\mu}\frac{\frac{1}{\lambda+\mu}-\left(T-\tau+\frac{1}{\lambda+\mu}\right)e^{-(\lambda+\mu)(T-\tau)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}} \tag{5.74}$$

And the last component of the sum (5.72) is simply

$$(T-\tau)\mathrm{P}_{5b}=(T-\tau)\frac{\alpha(\lambda+\mu)}{\lambda\mu}\frac{e^{-(\lambda+\mu)(T-\tau)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}} \tag{5.75}$$

Substitution of equations (5.73), (5.74), (5.75) in (5.72) yields

$$F^*(\tau)=\frac{\frac{1}{\lambda+\mu}+\frac{\alpha}{\lambda+\mu_1}\left[T-\tau+\frac{1}{\lambda+\mu}+\frac{1}{\lambda+\mu_1}\right]e^{-(\lambda+\mu)(T-\tau)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}\frac{\lambda_1\mu_1}{\lambda\mu}+$$

$$\frac{\alpha(\lambda+\mu_1)}{\lambda\mu}\frac{\frac{1}{\lambda+\mu}-\left(T-\tau+\frac{1}{\lambda+\mu}\right)e^{-(\lambda+\mu)(T-\tau)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}+$$

$$(T-\tau)\frac{\alpha(\lambda+\mu)}{\lambda\mu}\frac{e^{-(\lambda+\mu)(T-\tau)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}$$

After certain simplifications $F^*(\tau)$ can be obtained as

$$F^*(\tau)=\frac{\frac{1}{\lambda+\mu}+\frac{\alpha}{\lambda\mu}\left[\frac{\lambda\mu}{\lambda+\mu_1}(T-\tau)+\frac{\lambda_1\mu_1}{\lambda+\mu_1}\left(\frac{1}{\lambda+\mu_1}+\frac{1}{\lambda+\mu}\right)-\frac{\lambda+\mu_1}{\lambda+\mu}\right]e^{-(\lambda+\mu)(T-\tau)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}} \tag{5.76}$$

The above calculation have been conducted with the assumption that $T-\tau \geq 0$. Since the algorithm does not know T a priori, it can choose τ such that $T-\tau<0$. Assume now that $T-\tau<0$ and calculate $F^*(\tau)$.

Since $T-\tau<0$ the dependency picture on the Figure 5.13 will change as follows

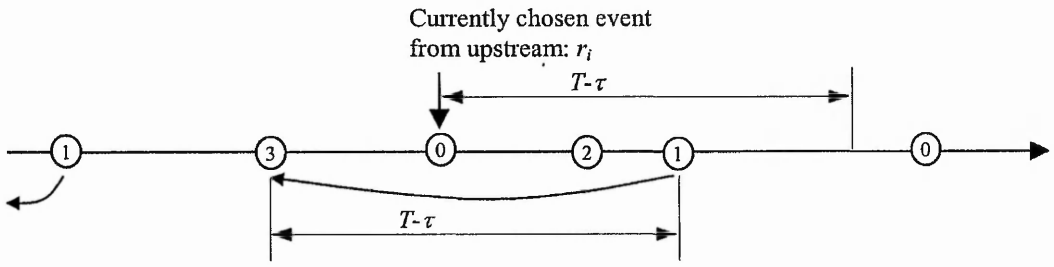


Figure 5.16. The events of types 1 and 3 have swapped their roles

It is clear from comparison of Figure 5.13 and Figure 5.16 that the dependency between events of processes $L_u(t)$ and $L_d(t)$ became backward from $L_u(t)$ to $L_d(t)$. That means that every event of $L_u(t)$ will have a correspondent event of $L_d(t)$ but in the past, located away by time $T-\tau$. Since the algorithm looks ahead for a candidate for the pair, the currently chosen event's type will not affect the algorithm's choice, that is the following events of $M(t)$ will be known to be independent of the currently chosen event of $N(t)$. Therefore, a separate consideration of $N_1(t)$ and $L_u(t)$, as it has been done for the case of $T-\tau \geq 0$, is not required.

Analogous to the previously considered case of the area of activeness of the process $L_d(t)$ from a randomly chosen time point t provided that $T-\tau \geq 0$, it is process $L_u(t)$ now that is limited in the area of $[t, t+T-\tau]$ from a randomly chosen time point t. Similarly to previous case of $L_d(t)$, if an event of $L_u(t)$ is removed further than $T-\tau$ from the point t, it will be known that there will be an event of $L_d(t)$ lying at the distance $T-\tau$ into the past from the event of $L_u(t)$, and thus masking out the event from $L_u(t)$.

Taking into account the above arguments, the pairs of events taken out by the algorithm will be identically distributed random variables with the probability density function shown on Figure 5.14 and thus have the mean function given by equation (5.66) with the only difference that $T-\tau$ must be replaced with $\tau-T$.

Since there is only one type of random variables the algorithm will be discovering in the case of T-τ<0, the equation (5.39) will take the following form

$$F^*(\tau)=\frac{\frac{1}{\lambda+\mu}+\frac{\alpha}{\lambda+\mu_1}\left[\tau-T+\frac{1}{\lambda+\mu}+\frac{1}{\lambda+\mu_1}\right]e^{-(\lambda+\mu)(\tau-T)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(\tau-T)}} \tag{5.77}$$

Finally, (5.76) and (5.77) together give the full solution for the limit version of the cost function (5.38) as follows

$$F^*(\tau)=\begin{cases}\dfrac{\frac{1}{\lambda+\mu}+\frac{\alpha}{\lambda\mu}\left[\frac{\lambda\mu}{\lambda+\mu_1}(T-\tau)+\frac{\lambda_1\mu_1}{\lambda+\mu_1}\left(\frac{1}{\lambda+\mu_1}+\frac{1}{\lambda+\mu}\right)-\frac{\lambda+\mu_1}{\lambda+\mu}\right]e^{-(\lambda+\mu)(T-\tau)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(T-\tau)}}, & \tau\le T \\ \dfrac{\frac{1}{\lambda+\mu}+\frac{\alpha}{\lambda+\mu_1}\left[\tau-T+\frac{1}{\lambda+\mu}+\frac{1}{\lambda+\mu_1}\right]e^{-(\lambda+\mu)(\tau-T)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(\tau-T)}}, & \tau>T\end{cases} \tag{5.78}$$

A typical plot of the above function is given in the following Figure 5.17.

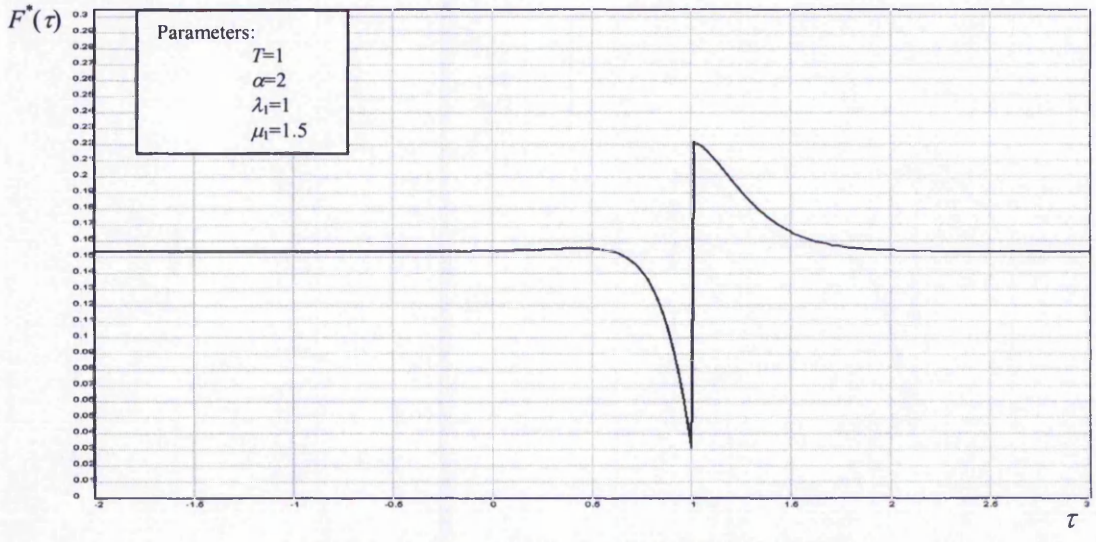


Figure 5.17. A typical plot of function (5.78)

It has therefore been shown that the limit (in probability) (5.39) exists and is finite and its analytical form has been identified in the form of the equation (5.78). In order to prove the last statement of the theorem it is required to show that (5.78) has a single global minimum at the point of $T-\tau=0$, or $\tau = T$.

Consider again cases $T-\tau\geq 0$ and $T-\tau<0$ separately. Assume first that $T-\tau\geq 0$. In order to investigate the properties of the function (5.78), let us introduce the following parameters.

$$a_1 = \frac{1}{\lambda+\mu},\ a_2 = \frac{\alpha}{\lambda+\mu_1},\ a_3 = \frac{\alpha}{\lambda\mu}\left[\frac{\lambda+\mu_1}{\lambda+\mu} - \frac{\lambda_1\mu_1}{\lambda+\mu_1}\left(\frac{1}{\lambda+\mu_1}+\frac{1}{\lambda+\mu}\right)\right],$$

$$k = \lambda+\mu \qquad \text{and} \qquad t = T-\tau \tag{5.79}$$

Assume that $\alpha>0$. It is obvious that $a_1>0$ and $a_2>0$ as it follows from the definitions of λ and μ. Let us show that a_3 is also greater than zero, e.g. $a_3>0$. First let us notice that $\frac{\alpha}{\lambda\mu}>0$, therefore it is sufficient to show that the term in the square brackets is greater than zero.

$$\frac{\lambda+\mu_1}{\lambda+\mu} - \frac{\lambda_1\mu_1}{\lambda+\mu_1}\left(\frac{1}{\lambda+\mu_1}+\frac{1}{\lambda+\mu}\right) > 0 \quad \Rightarrow \tag{5.80}$$

$$\frac{(\lambda+\mu_1)^3}{(\lambda+\mu_1)^2(\lambda+\mu)} - \frac{\lambda_1\mu_1(2\lambda+\mu+\mu_1)}{(\lambda+\mu_1)^2(\lambda+\mu)} > 0 \quad \Rightarrow$$

$$(\lambda+\mu_1)^3 - \lambda_1\mu_1(2\lambda+\mu+\mu_1) > 0 \quad \Rightarrow$$

$$(\lambda+\mu_1)^3 - (\lambda-\alpha)\mu_1(2\lambda+2\mu_1+\alpha) > 0 \quad \Rightarrow$$

$$(\lambda+\mu_1)^3 - 2\lambda^2\mu_1 - 2\lambda\mu_1^2 - \alpha\lambda\mu_1 + 2\alpha\lambda\mu_1 + 2\alpha\mu_1^2 + \alpha^2\mu_1 > 0 \quad \Rightarrow$$

$$\lambda^3 + 3\lambda^2\mu_1 + 3\lambda\mu_1^2 + \mu_1^3 - 2\lambda^2\mu_1 - 2\lambda\mu_1^2 + \alpha\lambda\mu_1 + 2\alpha\mu_1^2 + \alpha^2\mu_1 > 0 \quad \Rightarrow$$

$$\lambda^3 + \lambda^2\mu_1 + \lambda\mu_1^2 + \mu_1^3 + \alpha\lambda\mu_1 + 2\alpha\mu_1^2 + \alpha^2\mu_1 > 0$$

Obviously, the above statement is true. It is therefore shown that $a_3>0$.

Using (5.79), the function (5.78) can be rewritten in the following form.

$$f(t) = \frac{a_1 + (a_2 t - a_3)e^{-kt}}{1 + a_2 e^{-kt}} \tag{5.81}$$

where $a_1>0$, $a_2>0$, $a_3>0$, $k>0$ and $t\geq 0$.

Using elementary calculus and transformations, it can be shown that the condition $f'(t)=0$ leads to the following equation.

$$e^{-kt} = \frac{k}{a_2} t - \left(\frac{1}{a_2} + \frac{ka_3}{a_2^2} + \frac{ka_1}{a_2} \right) \tag{5.82}$$

Let us denote by $y(t)$ the right part of the above equation. Then $y(t)$ is a linear function. The condition $t=0$ yields $y(0)=-\left(\frac{1}{a_2} + \frac{ka_3}{a_2^2} + \frac{ka_1}{a_2} \right)<0$, and the condition $y(t)=0$ yields $t=\frac{1}{k} + \frac{a_3}{a_2} + a_1 > 0$. Therefore, the function $y(t)$ can be represented as in Figure 5.18.

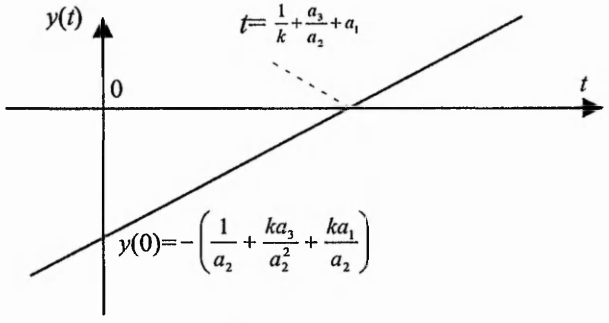


Figure 5.18. Schematic view of $y(t)$

It is now obvious, that since e^{-kt} is a monotonous decreasing function for $t\geq 0$, equation (5.82) has a single solution. It is illustrated in Figure 5.19.

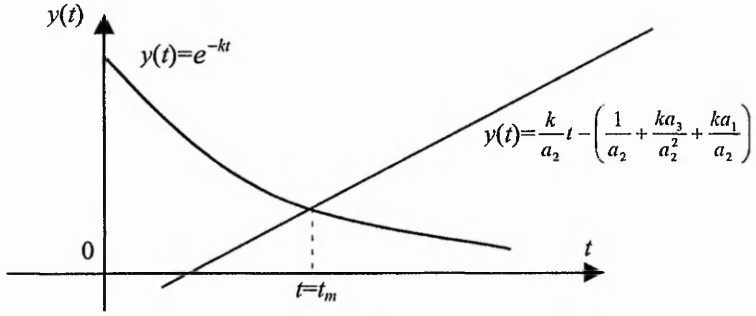


Figure 5.19. Single solution of equation (5.82).

It has therefore been shown that function (5.78) has a single extremum at $t=t_m$ or, equally, $\tau=\tau_m$, where $\tau_m=T-t_m$ on the interval $T-\tau\geq 0$. It can be easily shown by checking the sign of the derivative $f'(t)$ that the extremum is maximum. Since $F^*(t)$ is increasing on the interval $(-\infty,\tau_m)$ and decreasing on the interval $(\tau_m, T]$, it is required to show that $F^*(-\infty)\geq F^*(T)$. It is clear that

$$F^*(T)=\frac{\frac{\lambda+\mu_1}{\lambda+\mu}-\frac{\alpha(\lambda+\mu_1)}{\lambda\mu}\left[\frac{\lambda+\mu_1}{\lambda+\mu}-\frac{\lambda_1\mu_1}{\lambda+\mu_1}\left(\frac{1}{\lambda+\mu_1}+\frac{1}{\lambda+\mu}\right)\right]}{\lambda+\mu}$$

, and

$$\lim_{\tau\to-\infty}F^*(\tau)=\frac{1}{\lambda+\mu}$$

Then,

$$\frac{\frac{\lambda+\mu_1}{\lambda+\mu}-\frac{\alpha(\lambda+\mu_1)}{\lambda\mu}\left[\frac{\lambda+\mu_1}{\lambda+\mu}-\frac{\lambda_1\mu_1}{\lambda+\mu_1}\left(\frac{1}{\lambda+\mu_1}+\frac{1}{\lambda+\mu}\right)\right]}{\lambda+\mu}\leq\frac{1}{\lambda+\mu}\quad\Rightarrow$$

$$\frac{\lambda+\mu_1}{\lambda+\mu}-\frac{\alpha(\lambda+\mu_1)}{\lambda\mu}\left[\frac{\lambda+\mu_1}{\lambda+\mu}-\frac{\lambda_1\mu_1}{\lambda+\mu_1}\left(\frac{1}{\lambda+\mu_1}+\frac{1}{\lambda+\mu}\right)\right]\leq 1\quad\Rightarrow$$

$$1-\frac{\alpha}{\lambda+\mu}-\frac{\alpha(\lambda+\mu_1)}{\lambda\mu}\left[\frac{\lambda+\mu_1}{\lambda+\mu}-\frac{\lambda_1\mu_1}{\lambda+\mu_1}\left(\frac{1}{\lambda+\mu_1}+\frac{1}{\lambda+\mu}\right)\right]\leq 1\quad\Rightarrow$$

$$-\frac{\alpha}{\lambda+\mu}-\frac{\alpha(\lambda+\mu_1)}{\lambda\mu}\left[\frac{\lambda+\mu_1}{\lambda+\mu}-\frac{\lambda_1\mu_1}{\lambda+\mu_1}\left(\frac{1}{\lambda+\mu_1}+\frac{1}{\lambda+\mu}\right)\right]\leq 0 \qquad (5.83)$$

The term in the square brackets has been already shown to be greater than zero (inequality (5.80)), therefore, the above inequality (5.83) holds. It is important to notice that the equality (5.83) holds if and only if $\alpha=0$ (which is a trivial case).

Now, assume that $T-\tau<0$. To prove the theorem it is sufficient to show that

$$F^*(\tau)=\frac{\frac{1}{\lambda+\mu}+\frac{\alpha}{\lambda+\mu_1}\left[\tau-T+\frac{1}{\lambda+\mu}+\frac{1}{\lambda+\mu_1}\right]e^{-(\lambda+\mu)(\tau-T)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(\tau-T)}}\geq\frac{1}{\lambda+\mu} \qquad (5.84)$$

$$\frac{\frac{1}{\lambda+\mu}+\frac{\alpha}{\lambda+\mu_1}\left[\tau-T+\frac{1}{\lambda+\mu}+\frac{1}{\lambda+\mu_1}\right]e^{-(\lambda+\mu)(\tau-T)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(\tau-T)}} \geq \frac{\frac{1}{\lambda+\mu}\left(1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(\tau-T)}\right)}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(\tau-T)}} \Rightarrow$$

$$\frac{\frac{1}{\lambda+\mu}+\frac{\alpha}{\lambda+\mu_1}\left[\tau-T+\frac{1}{\lambda+\mu}+\frac{1}{\lambda+\mu_1}\right]e^{-(\lambda+\mu)(\tau-T)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(\tau-T)}} - \frac{\frac{1}{\lambda+\mu}\left(1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(\tau-T)}\right)}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(\tau-T)}} \geq 0 \Rightarrow$$

$$\frac{\frac{1}{\lambda+\mu}+\frac{\alpha}{\lambda+\mu_1}\left[\tau-T+\frac{1}{\lambda+\mu}+\frac{1}{\lambda+\mu_1}\right]e^{-(\lambda+\mu)(\tau-T)} - \frac{1}{\lambda+\mu} - \frac{1}{\lambda+\mu}\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(\tau-T)}}{1+\frac{\alpha}{\lambda+\mu_1}e^{-(\lambda+\mu)(\tau-T)}} \geq 0$$

$\Rightarrow$

$$\frac{\alpha}{\lambda+\mu_1}\left[\tau-T+\frac{1}{\lambda+\mu}+\frac{1}{\lambda+\mu_1}-\frac{1}{\lambda+\mu}\right]e^{-(\lambda+\mu)(\tau-T)} \geq 0 \quad \Rightarrow$$

$$\frac{\alpha}{\lambda+\mu_1}\left[\tau-T+\frac{1}{\lambda+\mu_1}\right]e^{-(\lambda+\mu)(\tau-T)} \geq 0, \tag{5.85}$$

which is obvious ($\tau>T$). The equality in (5.85) holds if and only if $\alpha=0$.

It has been shown that function (5.78) has a single global minimum at the point of $\tau=T$, provided that $\alpha>0$. In the case $\alpha=0$, function (5.78) becomes a constant, which is a special case and is of no interest. The proof is complete.

5.4.4 Applicability of the simplified traffic model

The model described above makes two main assumptions regarding traffic flow and travel time as a random variable. First, the processes involved in the calculations have to be homogeneous Poisson processes. The adequateness of modelling road traffic as a Poisson process started to be investigated in the fifties [Gerlough, 1955], [Gerlough and Barnes, 1971]. It has been shown that light highway traffic reasonably fits into the

Poisson process model. Many other models of traffic have been proposed [Cowan, 1975]. Unfortunately, it is well known from the theory of stochastic processes, that the Poisson process is the only renewal process with the property that a sum of two independent Poisson processes is also a Poisson process. Derivation of closed analytical results for the other renewal processes is cumbersome, if at all possible [Karlin, 1975].

The second assumption is that the travel time of vehicles traversing the distance from *A* to *B* is constant. This requirement was necessary in order to keep the downstream process being Poisson. In case of a non-constant travel time, the convolution of a gamma-distributed arrival times at point A with a non-constant time (which would require a normal-like distribution) will yield a distribution that is certainly not gamma distribution and therefore, the downstream fails to be Poisson. This assumption is also reasonable in very short distances, as is the case with urban links.

5.5 Conclusions

In this chapter, a review of the current state of the art in solving the problem of travel time estimation from traffic data has been reviewed. A conclusion has been made that most algorithms have certain disadvantages for the application in urban areas. Following the review, a problem of travel time estimation for an urban link has been formulated and novel methodology of solving the problem has been developed. The methodology has been proven analytically for a simplified traffic model using the theory of stochastic processes.

The following Chapter 6 will present the results of empirical study of a real urban network based on the automatically collected traffic data and using the software and theoretical methodology developed during the course of this study.

Chapter 6
Empirical Study of Urban Traffic

6.1 Introduction

Previous chapters have developed necessary tools (Chapter 4 has developed the software environment for system modelling and simulation and Chapter 5 has developed a novel methodology of travel time estimation on urban areas) in order to facilitate the empirical study of real-life traffic system. In this chapter, such an empirical study of urban traffic will be carried out.

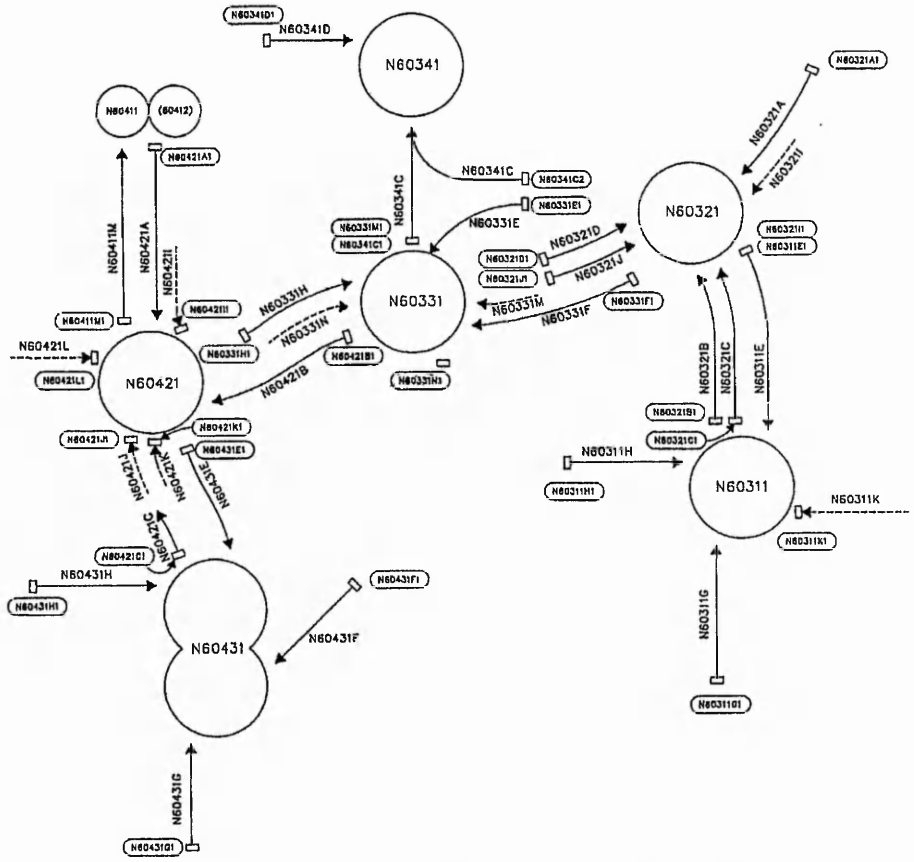


Figure 6.1. Mansfield SCOOT-controlled region

Chapter 3 has introduced the SCOOT system since it provides data for the current study. The SCOOT system used in this study is installed in a town of Mansfield, UK. The

SCOOT controlled region map is given on Figure 6.1. References to the links on the map will be made throughout this chapter.

The structure of this chapter is as follows. In the first section, necessary software modules will be developed in order to facilitate the SCOOT data processing and analysis. Then follows a set of demonstrations on the use of developed data processing environment with integrated modules for SCOOT data processing and analysis. Several standard data analysis procedures will be carried out and results presented. Section 6.4 will be solely devoted to the implementation of the travel time methodology developed in Chapter 5. Empirical study of travel times using Mansfield SCOOT data will be given in Section 6.5. The study of travel times in urban links will be finalised in Section 6.6 with the estimation of its components as given in Chapter 5, Section 5.5.1. Finally, the chapter will be closed with conclusions.

6.2 Pre-processing of SCOOT data

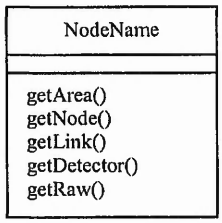
Raw SCOOT messages in the form of textual strings are not very convenient format for carrying out data analysis. It is more desirable that messages are processed into higher level data structures and the informative fields of the messages are properly parsed and filled in to the structure. Moreover, low level messages are often redundant in a sense that they are generated but do not carry useful information (filled in with dummy data). The two message types, described in Chapter 3, M14 and M19 are such messages. Therefore, it would be beneficial to filter out empty messages while creating proper information stream of useful messages to the data analysis algorithm.

Even though this study uses only two types of SCOOT messages, it is a good practice to develop a pre-processing software in such a way, that it would allow, in principle, processing any other kind of messages. Such pre-processing software has been developed as a functional module for the data processing environment and will be described here.

6.2.1 The core pre-processing software

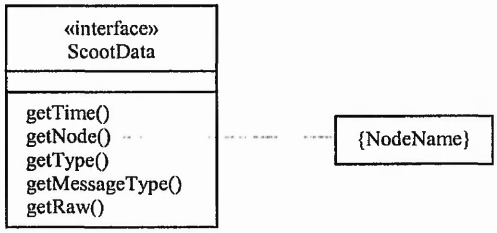
The SCOOT messages pre-processing software will be developed as a package named, `modules.scoot.data`.

Since all SCOOT messages have spatial attributes, such as the node, link and detector information, the first data structure to be defined is the one that describes such spatial attribute. Such structure has been designed and initially called `NodeName`. The following model represents the `NodeName` class.



From the model of the `NodeName` class, it is clear that the spatial paratemer of the SCOOT messages have the following attributes: Area, Node, Link and Detector, as follows from the corresponding methods.

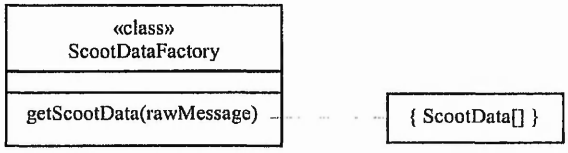
The basic data structure that will be associated with a SCOOT message is named `ScootData`. `ScootData` is implemented as a Java interface and describes a set of methods that all implementing classes should have. The methods of `ScootData` interface are designed to reflect the common fields that all SCOOT messages possess. The following model of the interface describes its structure.



Since `ScootData` is an interface, it is an abstraction of the real SCOOT data objects. There is no requirement that data objects correspond to SCOOT messages since some SCOOT messages might produce data of different nature. An example of such message is M14 which carries SCOOT modelling results in the form of predictions of the lengths of the queues and accompanied signal control outputs, which can be considered as independent piece of information. Based on this argument, a decision to split M14 messages into two SCOOT data objects has been made. The detail of implementation will be presented later in this section.

The minimal implementation of the `ScootData` interface is called `ScootMessage`. `ScootMessage` is used to parse the header of any SCOOT message and leave the type-specific information of a message in its raw form.

To satisfy the requirement for the pre-processing core to be able to process any kind of SCOOT message, a standard Java technique of using a *class factory* has been employed. A class factory is normally a class, which is used to create instances of other classes based on the given specification. The class factory technique extensively uses interfaces and abstract classes mechanisms. A class factory has been implemented to create instances of SCOOT data objects according to the availability of implementation of the classes for particular messages types. In case when no implementation is available, the most general form of a SCOOT data object, implemented by the `ScootMessage` class is used. The following diagram gives an outlook on the SCOOT class factory structure.



When `getScootData(rawMessage)` method is invoked, it parses the `rawMessage` into an instance of the `ScootMessage` class, from which the type of the message can be obtained (as a string in the form of 'M14', 'W03', etc.). After the type of the message has been obtained, the factory will try to locate a class with the following name `ScootData<TYPE>`, for example, `ScootDataM14` for M14 messages. In case if such class is not found, the factory will return the `ScootMessage` object, since no further processing can be done. If `ScootData<TYPE>`, has been found, the factory will invoke a static method called `process`, which is specified as follows

```
ScootData [] process(ScootMessage msg, HashMap store)
                    throws WrongTypeException, BadDataException;
```

The above method of the `ScootData<TYPE>` class will know how to parse that type of message into an instance of itself or other related SCOOT data objects. Since it is possible that one raw message is processed into more than one instance of SCOOT data

objects, the `process` method returns an array of such objects. `WrongTypeException` and `BadDataException` are thrown if the process has been invoked for a different type of message or the format of the message is violated.

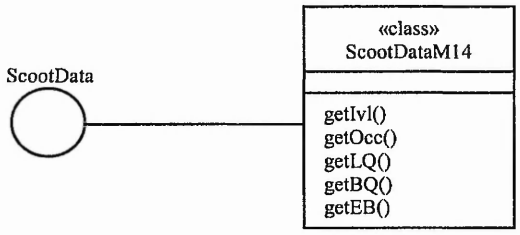
It follows from the use of factory as described above, that there is no requirement to implement `ScootData<TYPE>` classes for all possible types of messages since basic pre-processing is implemented as a `ScootMessage` class. On the contrary, it is easy to add the functionality for the system to process the required message type by simply creating a class with the name of the form `ScootData<TYPE>` and implement the `process` method as given above.

The above scheme has been implemented, tested and shown to be both flexible and efficient.

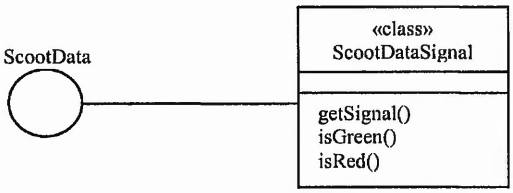
6.2.2 Pre-processing of M14 and M19 messages

Pre-processing class factories have been implemented for the two message types, M14 and M19, which will be used in further analysis.

M14 processing factory, `ScootDataM14`, produces two objects of SCOOT data, namely the SCOOT modelling data object and SCOOT control data object. The SCOOT modelling data is returned as a dynamically created instance of `ScootDataM14`, with the fields filled with data parsed from the raw message. The following model presents an instance of `ScootDataM14` class.

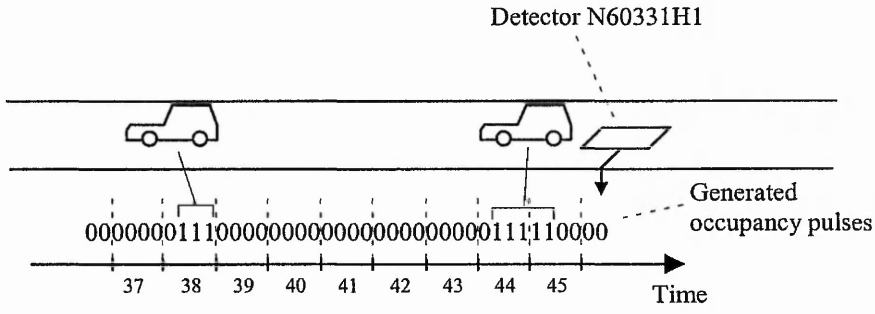


In addition M14 processing factory creates instances of `ScootDataSignal` class, which represents the changes in signalling stages at a particular link.



`ScootDataSignal` provides methods (as shown above) that allow checking the current signalling stage at a link.

M19 processing factory, `ScootDataM19`, searches non-zero occupancy strings in the flow of occupancy pulses generated by a particular detector. Let us get back to Figure 3.4

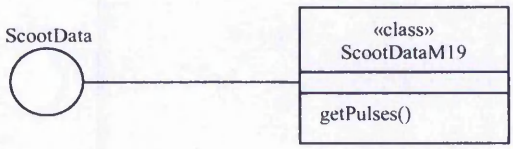



```
Sa 08:26:38  M19  N60331H1 DETECTOR STATE 0111
Sa 08:26:39  M19  N60331H1 DETECTOR STATE 0000
Sa 08:26:40  M19  N60331H1 DETECTOR STATE 0000
Sa 08:26:41  M19  N60331H1 DETECTOR STATE 0000
Sa 08:26:42  M19  N60331H1 DETECTOR STATE 0000
Sa 08:26:43  M19  N60331H1 DETECTOR STATE 0000
Sa 08:26:44  M19  N60331H1 DETECTOR STATE 0111
Sa 08:26:45  M19  N60331H1 DETECTOR STATE 1100
```

The M19 factory will receive a flow of messages as shown above and try to reconstruct the string of pulses as pointed in the figure under "Generated occupancy pulses". Having constructed a continuous string of pulses, the factory will extract strings that contain continuous strings of 1's, as 111 and 11111 on the figure, and create the instances of `ScootDataM19` initialised with the time of a string of 1's appearance and the number of pulses in the string. Thus, from the example above two M19 data objects will be created:

Time	№ of pulses
08:26:38.25	3
08:26:44.25	5

The model of M19 data class is given below.



6.2.3 SCOOT message pre-processing module for the data processing environment

A module for pre-processing SCOOT messages has been implemented as part of `modules.scoot.data` package and shown in the following Figure 6.2.

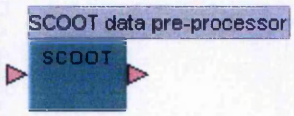


Figure 6.2. The look of the SCOOT messages pre-processing module for the data processing environment

The module accepts SCOOT messages in their raw textual form on its input and produces a stream of `ScootData` objects that correspond to the parsed messages. In order to process the textual messages, a SCOOT data processing factory, as described in Section 6.2.1, is used.

The raw messages can be obtained from various sources such as files, network, keyboard, or directly from SCOOT as described in Section 3.3.4. A typical scheme of using a file reader and SCOOT pre-processing modules is shown on the following Figure 6.3.

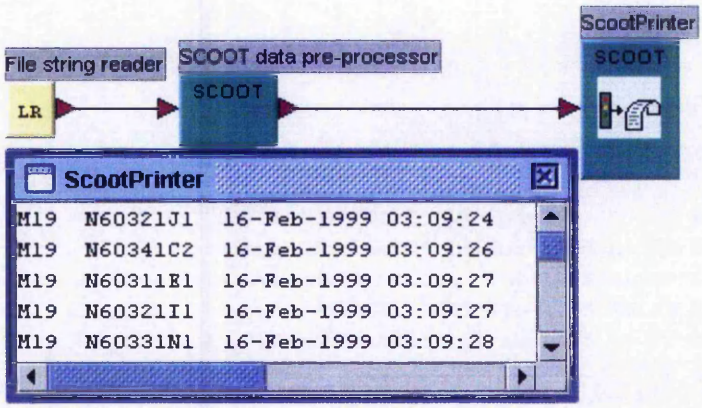


Figure 6.3. A fragment of a model that uses a file reader and SCOOT data pre-processor

6.3 Use of the processing software in traffic analysis

The developed software for data traffic processing can be used in many standard traffic analysis procedures. As examples, two procedures – headways study and traffic profiles will be presented here.

Headway is the time gap between successive vehicles in a traffic stream. Analysis of headways is used in many transportation decision making processes, such as saturation flow and traffic signals design [Adams, Hummer, 1993], or gap acceptance studies [Troutbeck, 1993].

A module for extraction of headways from traffic data stream has been developed and tested. As an example of a module implementation, full source code of the module is given in Appendix C, Example 3. Appendix B, Figure B.4 shows the screen of the data processing environment with a model that implements headway data extraction and visualisation using plot, histogram and display modules. From the headways plot, it is noticeable how traffic control affects formation of platoons. Platoons of cars correspond to the low little changing areas on the plot and peaks represent a delay induced by the change of the signalling stage at the intersection. A typical distribution of headways is given on the following Figure 6.4.

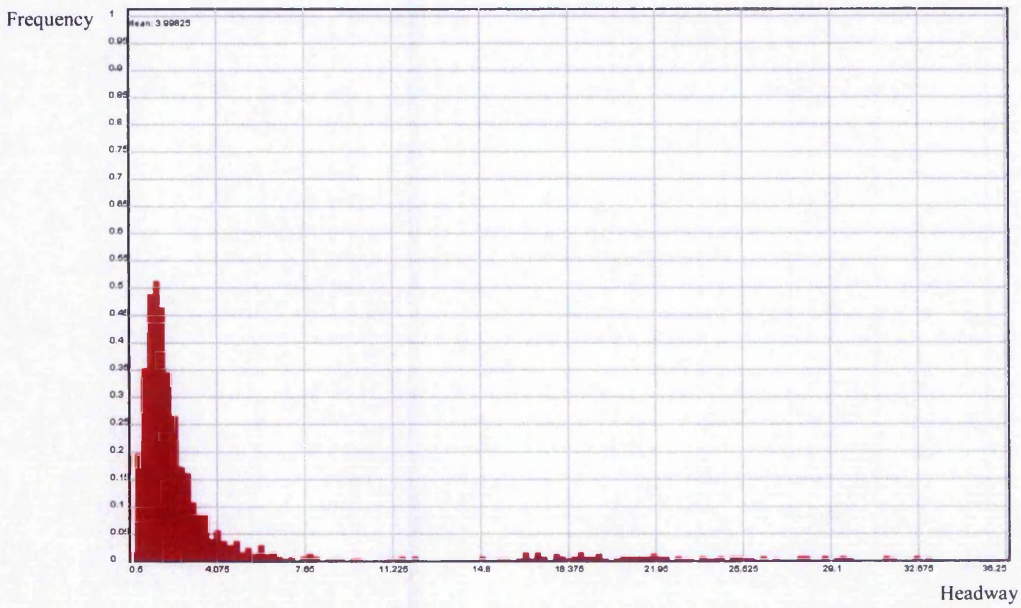


Figure 6.4. Distribution of headways (taken from N60411M1)

Traffic profiles also play an important role in traffic control strategies. For example, SCOOT can use historical traffic profile data to optimise traffic signals in case if detector hardware gave a fault. Also, traffic profiles give a global outlook on changes in traffic volumes depending on season, weather and other factors.

A module for construction of traffic profiles using SCOOT data has been developed (see example screen of the data processing environment with profiling module in Appendix B, Figure B.6 and Figure B.7).

Typical traffic profiles for weekend (red) and working day (blue) are shown in the following Figure 6.5.

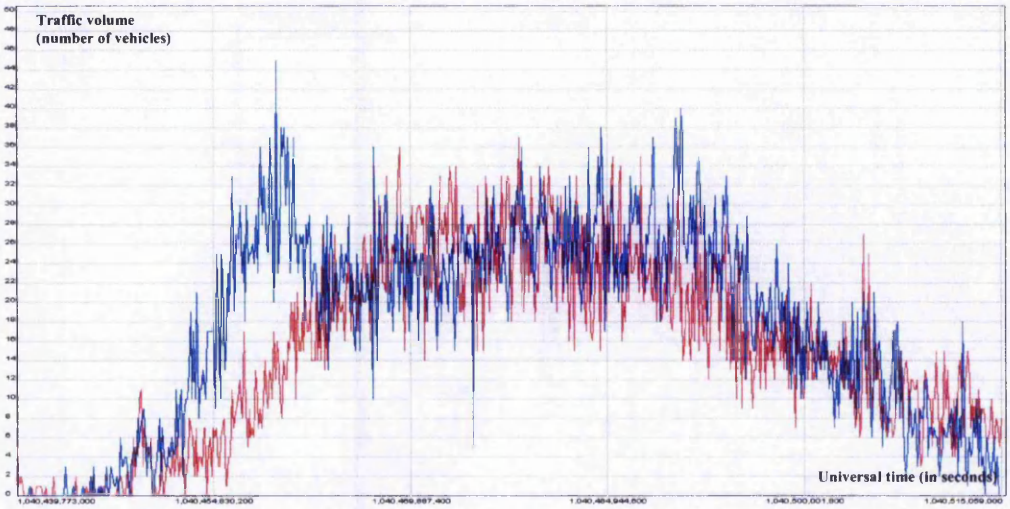


Figure 6.5. Example of traffic profiling module output visualised by Plot module. Data used is taken for the N60411M1 detector for 16th January 1999, Saturday (red) and 19th January 1999, Tuesday (blue) and aggregated over 2 minutes intervals.

Other modules for traffic analysis can be developed as required in particular application.

6.4 Implementation of the travel time estimation procedure

Travel time estimation procedure has been implemented as a functional module to the data processing environment. It has one input, as shown in Figure 6.6, and three outputs.

A stream of `ScootData` objects is received on the input and three types of information regarding travel time estimation are sent out from the three outputs.

Figure 6.6. Travel time estimation module

The following parameters are used within the travel time estimation module.

Table 6.1. Parameters of the travel time estimation module

Parameter name	Type	Description
UPSTREAM	String	Upstream detector name
DOWNSTREAM	String	Downstream detector name
TW	long	Time window within which the module will collect input vectors (in seconds)
DS	boolean	Do stops for every new time window (true) or not (false)
CS	boolean	Take signalling stages on the uplink into account (true) or do not (false)
LSB	long	Left boundary for time shifts (in milliseconds)
RSB	long	Left boundary for time shifts (in milliseconds)
STEP	long	Step of the shift (in milliseconds)

The scheme of the module's functionality is as follows. Once the parameters have been set up, a module is ready to carry out travel time estimation. The first two parameters, the upstream and downstream detector names are necessary parameters. Considering Figure 6.1, examples of couples of detectors can be given as N60311G1-N60321B1, N60321B1-

N60331F1, N60331F1-N60421B1, etc. One module estimates travel time for just one couple of upstream-downstream detectors. Time window (TW) is the time interval during which input data will be collected into two input vectors of event arrival times (refer to Chapter 5 for details on the requirements for the travel estimation procedure). If CS parameter is set to 'true' then the module will discard the downstream arrival events during the red phase of the signals on the uplink. No discarding is done if CS is set to 'false'. After the flow of SCOOT data objects is initiated on the input of the module, the module starts filtering M19 objects that correspond to upstream or downstream detectors and building up the upstream and downstream arrival events vectors. After the arrival time vectors have been built over a period of time equal TW, the input is blocked and the algorithm of travel time estimation is started. The algorithm implemented as described in Section 5.4.2 of Chapter 5 is a local optimisation algorithm.

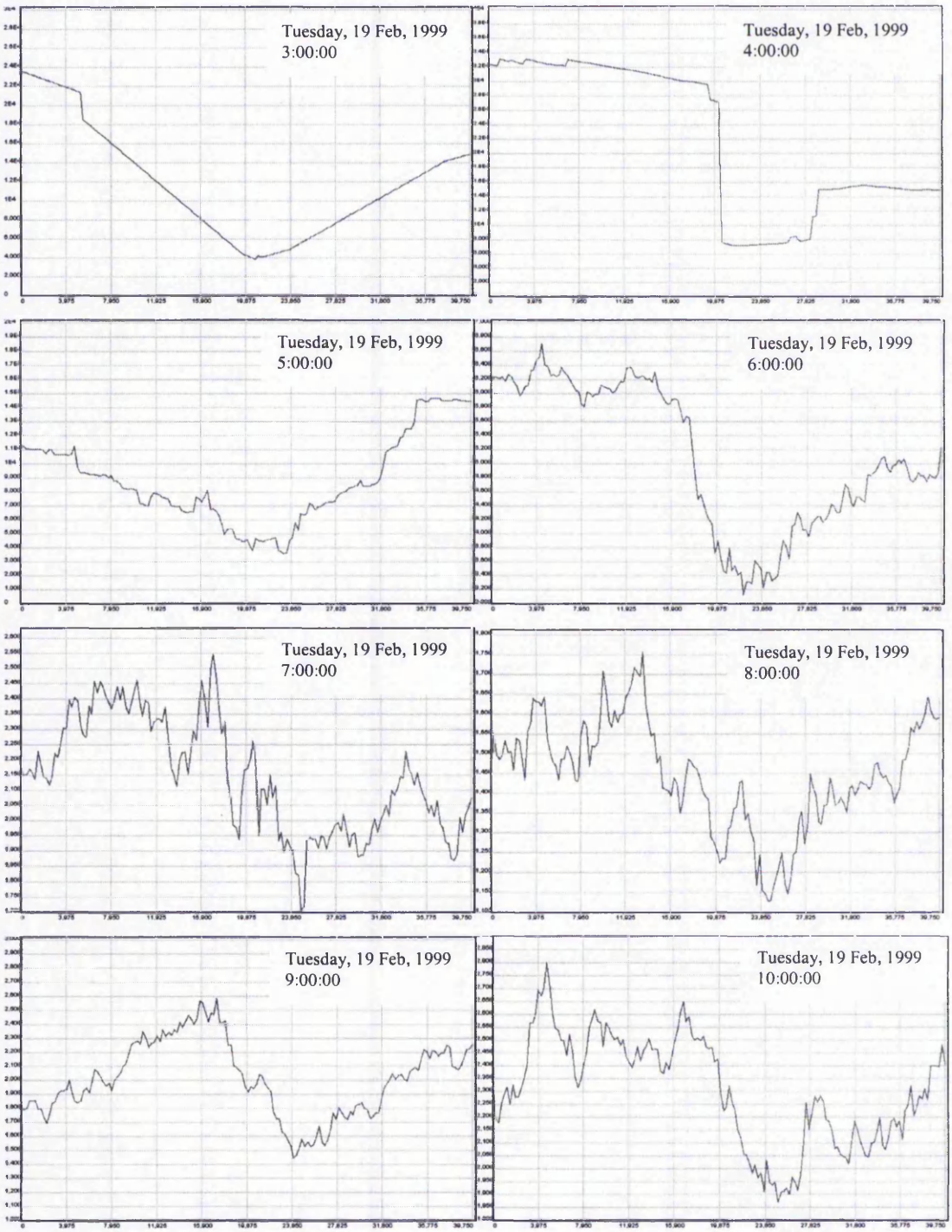
The algorithm produces three types of results. The first result is the cost function of time shift. The function is represented as an array of two-dimensional points (x,y) where x is the time shift and y is the value of cost function for that shift. The argument x which minimises the function is the estimation of the median of travel time variable, as described in Section 5.4.2, and is the point estimation of the link travel time. This is the second type of result. The third result is the series of time differences produced by the algorithm for the shift that minimises the cost function. This series can be used for estimation of travel time probability distribution or other characteristics.

After the above results have been obtained, the module forward them to the outputs. Thus, the first output corresponds to the cost function, the second corresponds to the series of time differences and the third corresponds to the point estimation of the travel time. Several experiments of link travel time estimation have been carried out for some links of the Mansfield SCOOT controlled region showed in Figure 6.1. The results are outlined in the following section.

6.5 Empirical study of travel time on urban links

The following set of travel time estimates has been obtained for the pair of detectors N60331F1 as the upstream detector and N60421B1 as the downstream detector. The time

window was chosen to be one hour (3600 seconds) and the cost functions were obtained for time shifts from the interval [0, 40] seconds with the step of 0.25 second (lowest resolution of the data). The following set of plots presents the cost functions calculated over different times (one hour window).




Tuesday, 19 Feb, 1999
11:00:00
Tuesday, 19 Feb, 1999
12:00:00
Tuesday, 19 Feb, 1999
13:00:00
Tuesday, 19 Feb, 1999
14:00:00
Tuesday, 19 Feb, 1999
15:00:00
Tuesday, 19 Feb, 1999
16:00:00
Tuesday, 19 Feb, 1999
17:00:00
Tuesday, 19 Feb, 1999
18:00:00

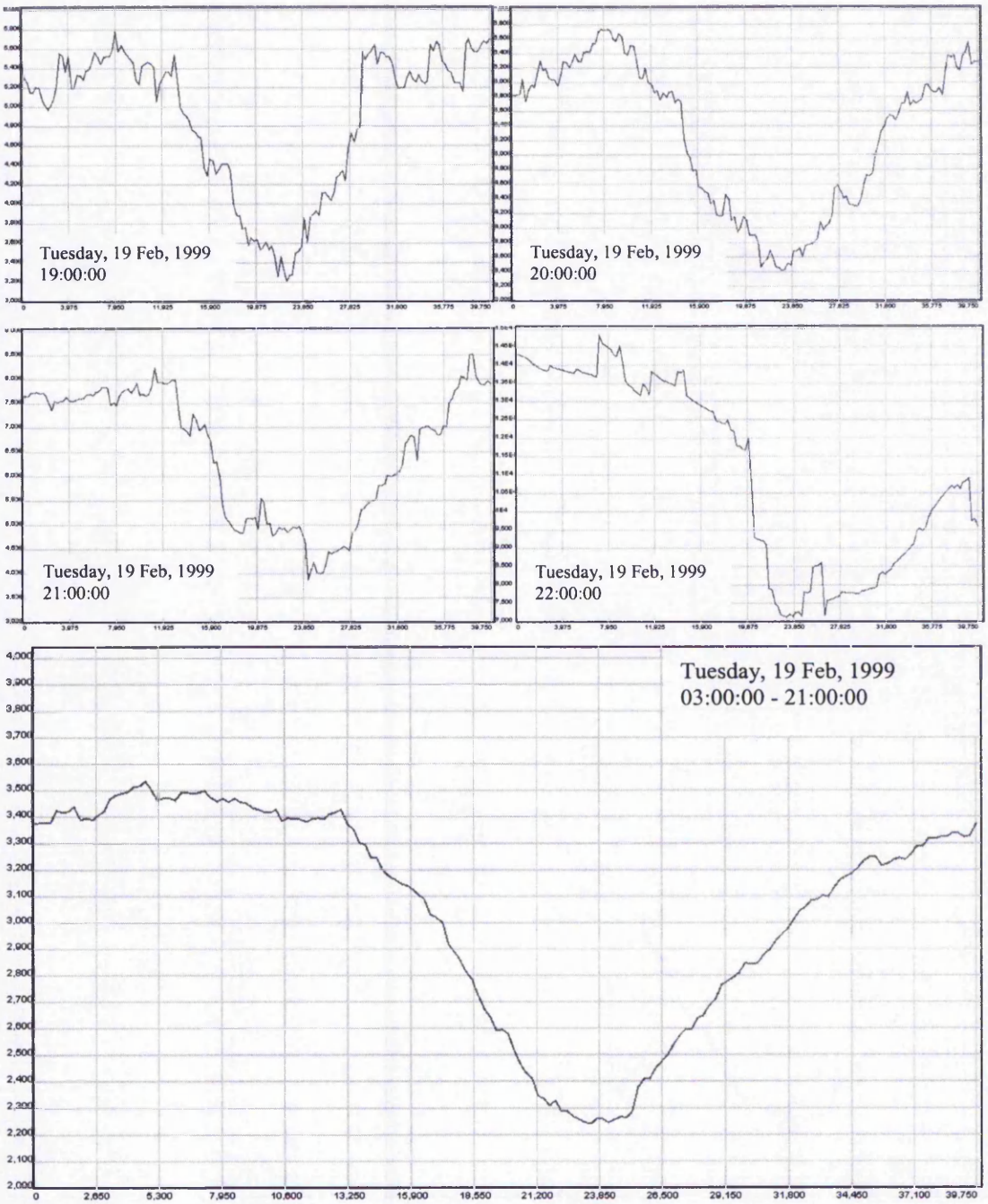
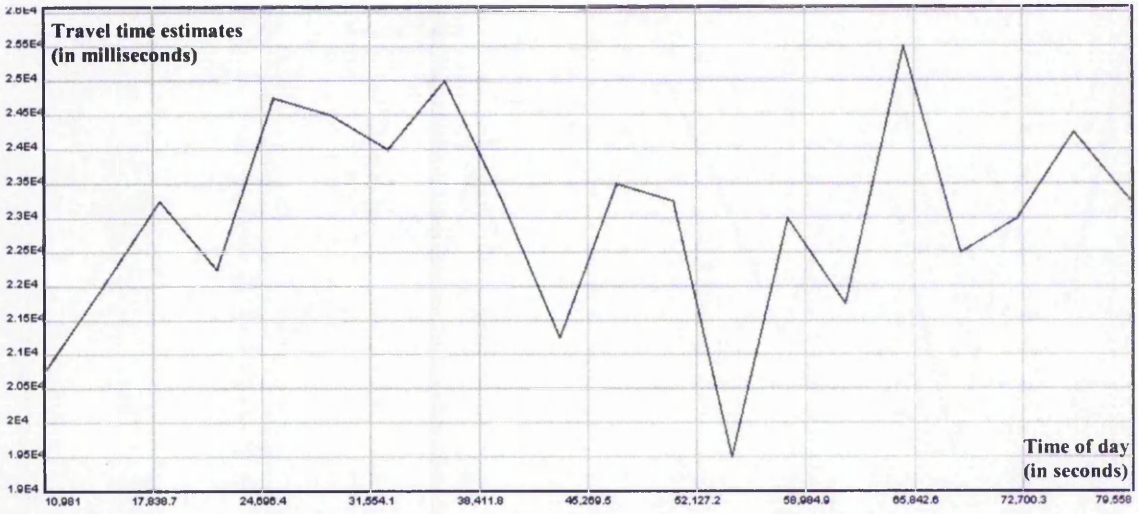


Figure 6.7. Cost functions calculated for different times of day. Link N60331F1-N60421B1, 19 February, Tuesday, 1999.

It is clear from the above plots that the cost functions all have minimum at about 20-25 seconds. The following plot demonstrates the point estimates of travel times that correspond to each plot above.



The values of travel time estimates are given in the following table (in seconds).

Table 6.2. Values of travel time point estimates

20.75	22.00	23.25	22.25	24.75
24.50	24.00	25.00	23.25	21.25
23.50	23.25	19.50	23.00	21.75
25.50	22.50	23.00	24.25	23.25

The set of time differences used in calculation of the cost function can be analysed separately and can provide useful information on the distribution of travel time as a random variable. Figure 6.8 presents the histogram of the series of differences produced by the algorithm.

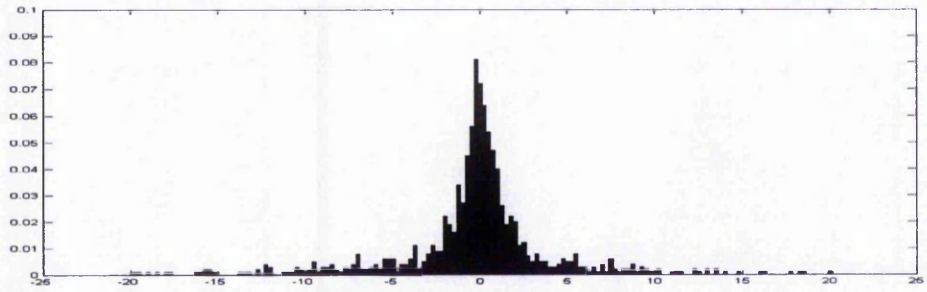
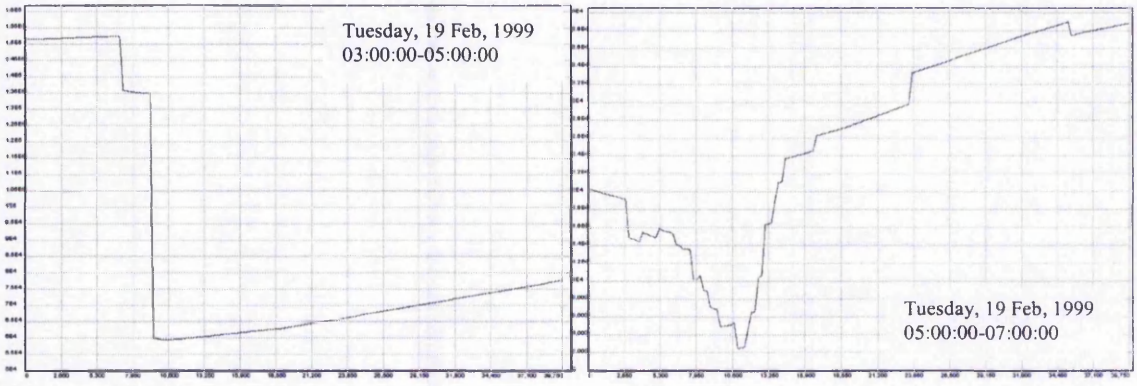


Figure 6.8. Histogram of the time differences produced by the algorithm that correspond to the minimum of the cost function

The mean value of the series of time differences for the above link is -0.0475 and the standard deviation is 6.31. Since the series of time differences contain noise together with travel time corresponding differences (see Chapter 5 for details on related and non-related time differences), the value of the standard deviation is higher than the real deviation of travel time. Thus, it can be said that standard deviation of travel time is not higher than 6.31 for the above case.

From Table 6.2 the mean travel time can be calculated. The result is 23.025 seconds with standard deviation of 1.5 second. This gives another estimate of the real deviation of travel time random variable.

It can be mentioned that the travel time estimation methodology does not make any assumption on the mutual location of the upstream and downstream detectors relatively to each other. Therefore, the methodology can be applied to estimate all types of links, the straight link and the turning link. The following results show the cost functions calculated for a 'turn-right' type of link, namely from N60311K1 to N60321C1 (see the region map in Figure 6.1). The link must have smaller travel time than previously considered link since the upstream detector is placed immediately before the junction (turning vehicles counter). Also, since the upstream detector counts only turning vehicles, the cost functions are expected to be clearer than the ones of the previously considered link. The following plots confirm the above expectations. The cost functions given in Figure 6.9 have been calculated over the period of two hours and using the data collected on the 19th February (Tuesday), 1999.



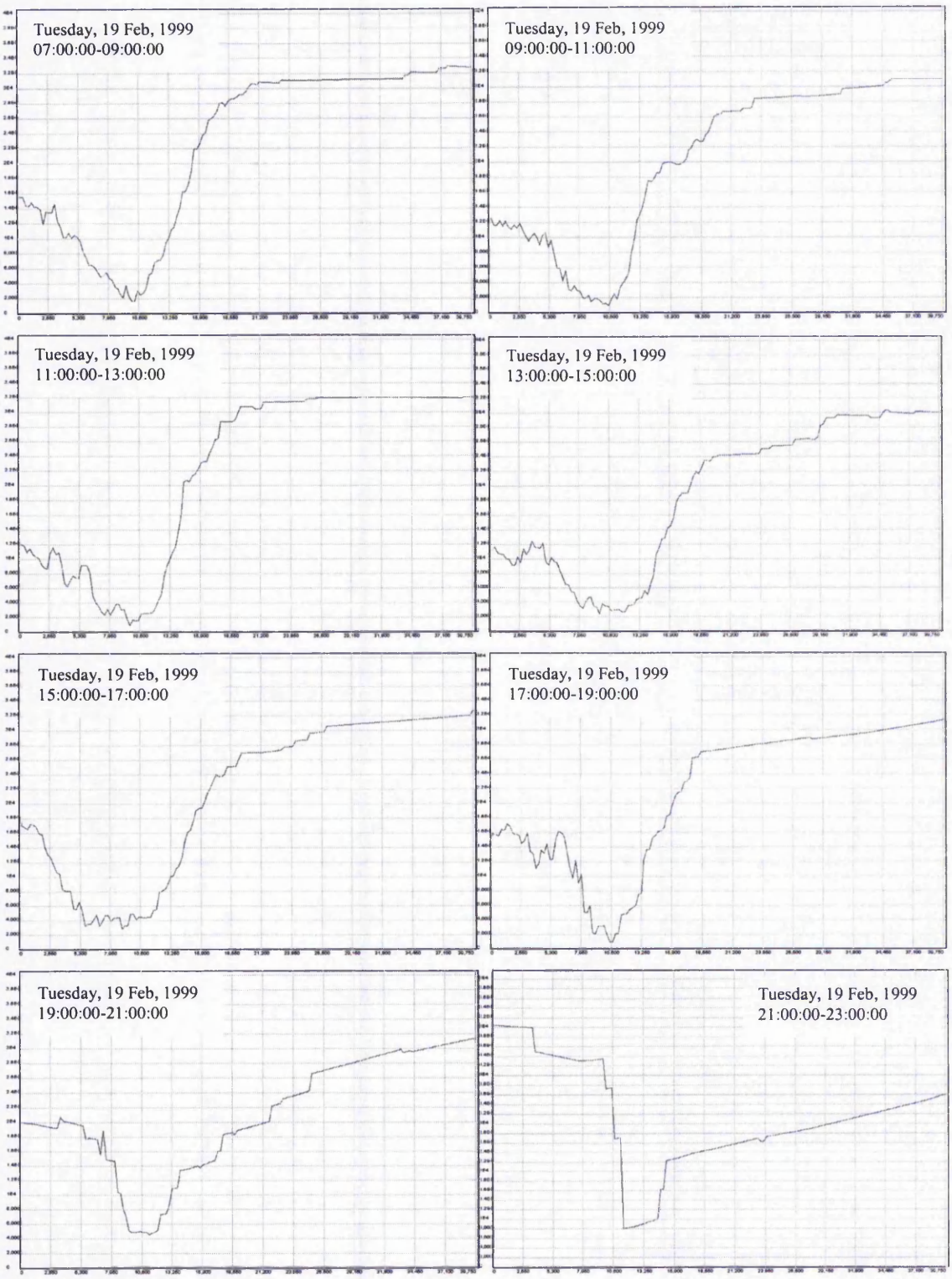


Figure 6.9. Cost functions for the 'turn-right' type of a link: N60311K1 – N60321C1, 19th February (Tuesday), 1999. Time windows interval is 2 hours.

Clearly, the minimum of the functions point to the same neighbourhood of about 10 seconds. The following plot shows the cost function calculated for all the day.

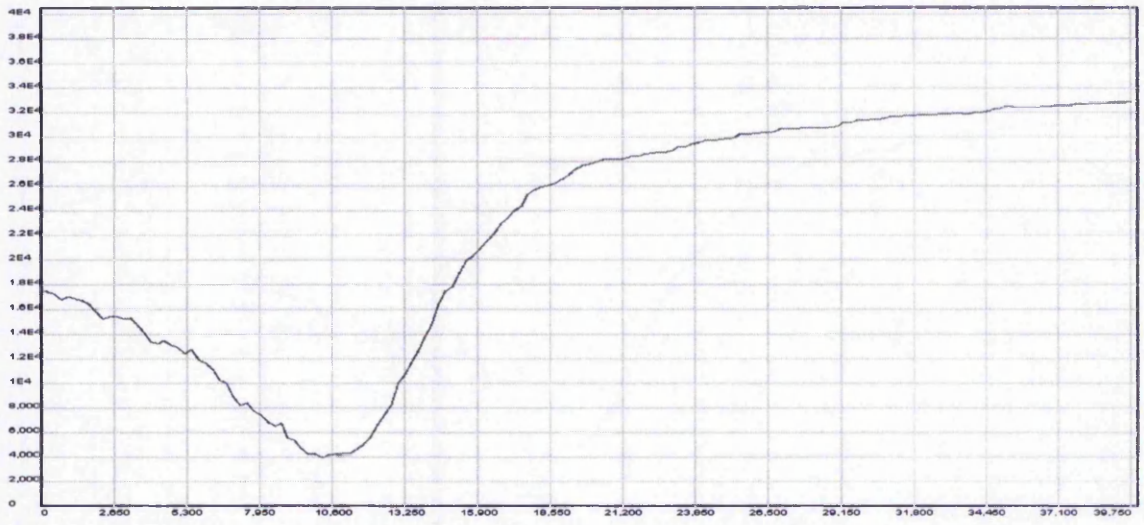


Figure 6.10. Cost function for the 'right-turn' type of a link: N60311K1 – N60321C1, 19th February (Tuesday), 1999. All day cost function.

All estimates of the travel time for the link N60311K1–N60321C1 throughout the day are given in the following plot.

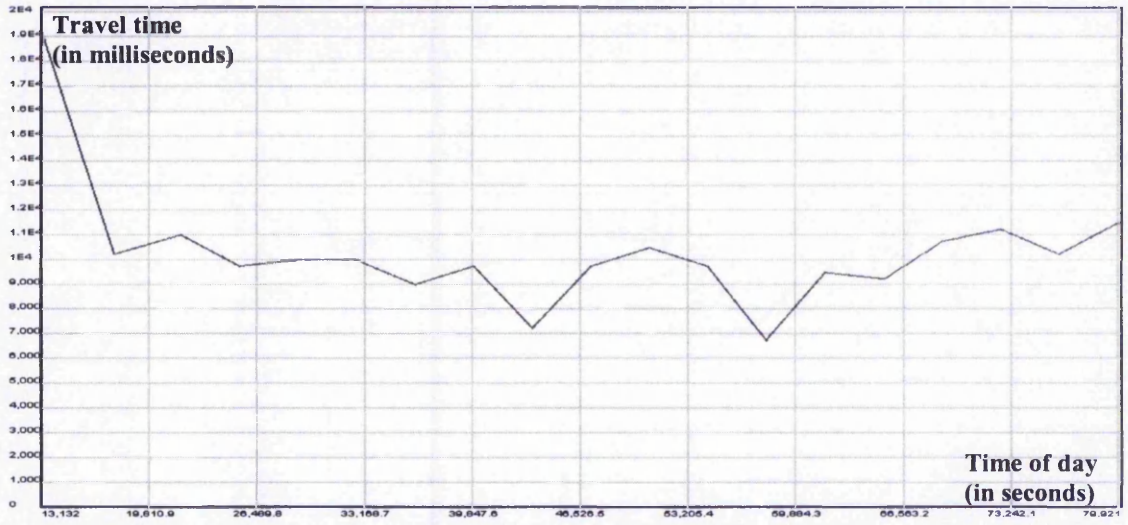


Figure 6.11. Plot of travel time estimates for the 'right-turn' type of a link: N60311K1 – N60321C1, 19th February (Tuesday), 1999.

The values of the travel time estimates are as follows.

19,25 10,25 11,00 09,75 10,00 10,00 09,00 09,75 07,25 09,75
10,50 09,75 06,75 09,50 09,25 10,75 11,25 10,25 11,50

The mean value of the estimates is 10.29 seconds and standard deviation is 2.47 seconds.

Although some similarity can be found in all time series collected from the detectors of the same SCOOT controlled region (since SCOOT operates on the same cycle length in all region), the cost functions calculated for unrelated series are more irregular. The following plot demonstrates a cost function calculated for such a link.

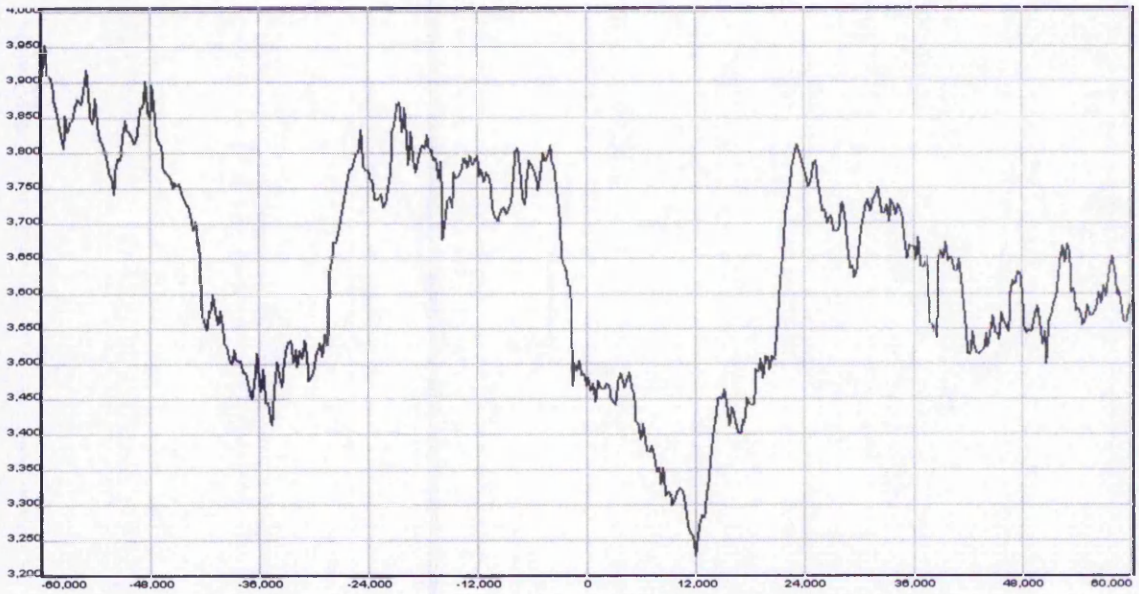


Figure 6.12. Cost function of unrelated series. A link: N60311K1 – N60321C1, 19th February (Tuesday), 1999. Calculated over all day.

Despite the similarities between any two series taken from a region controlled by SCOOT due to its cyclic nature, the global minimum of the cost function points to the travel time if the series are related. The following plot (Figure 6.13) demonstrates the same cost as shown Figure 6.7 but in a bigger scale.

It is clear that the global minimum points to the real travel time, since only this time shift provides the best match between the series even among the similar cyclic profiles.

The model of the travel time estimation procedure used in the above experiments is shown in Figure B.8 of Appendix B.

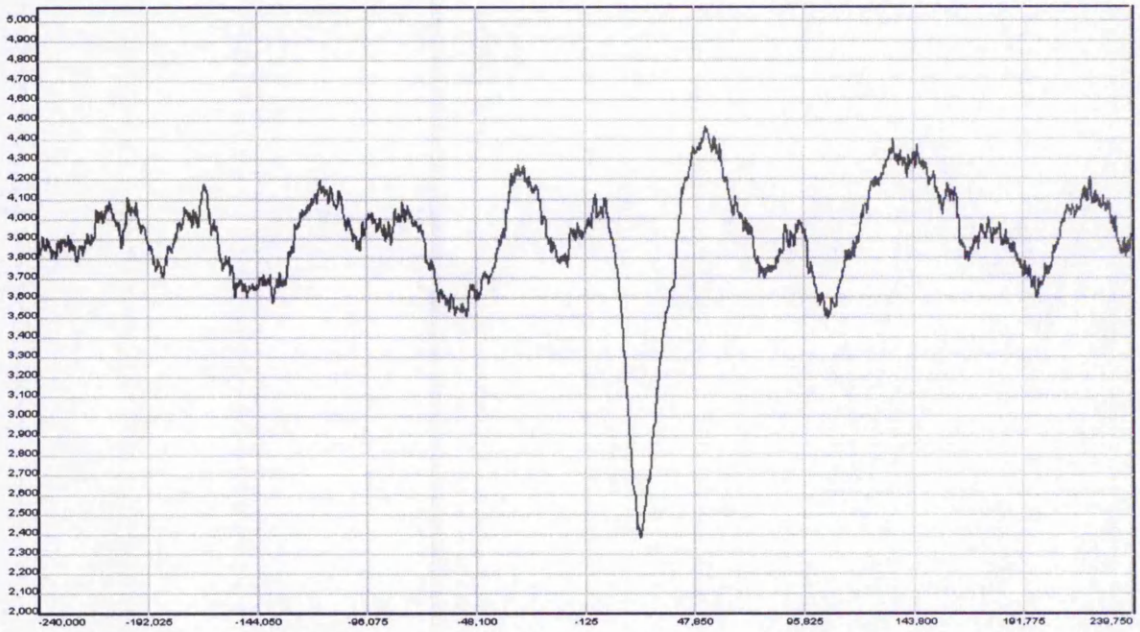


Figure 6.13. Cost function of link N60331F1-N60421B1 in a bigger scale. Several minimums point to the similarities due to cyclic nature of the series. The global minimum points to the "best match" between the series. Time shift varies from –4 minutes to 4 minutes. The global minimum is reached at time shift equal 24.25 seconds.

6.6 Urban Travel Time components

In Chapter 5 a model of an urban link has been given and the link travel time has been defined as a sum of two components, namely the travel time from the upstream detector to the stop line and from the stop line to the downstream detector. The full travel time between two detectors (defined as T_T – see Figure 6.14) has been studied in previous sections.

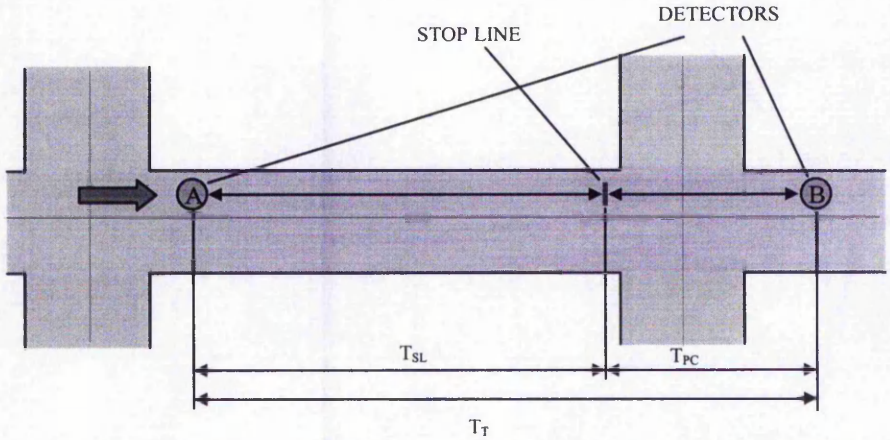


Figure 6.14. The urban link model and travel time components

In some applications one is interested in T_{SL} rather than T_T. An example of such an application is the SCOOT traffic model, which strongly relies on the T_{SL} parameter. T_{SL} is typically provided to SCOOT as a constant at the time of installation. Another use of T_{SL} parameter is the estimation of turning movements that is based on tracking individual vehicles rather than on the analysis of aggregated traffic flows. It is therefore important to estimate this parameter from traffic data.

If only counter data is available it is impossible to distinguish the travel time components. In this case additional information is required. Fortunately, if information on signalling stages at the junction is available it becomes possible to carry out the estimation of T_{PC} parameter. A T_{PC} component estimation scheme based on the knowledge of signalling stages has been developed. The main idea behind the scheme is the correlation of the time of change of the signal from "red" to "green" with the time of arrival of the first vehicle to the downstream detector.

The estimation requires knowledge of signalling times and the knowledge that there has been a queue formed during "red" phase of the traffic light. Signalling time series is available directly from SCOOT (in M14 messages). However, the knowledge regarding the presence of a queue at the stop line during the "red" phase of the signals is uncertain, as there is no information regarding the queue growth process. Lane occupancy on the upstream detector cannot be used as indicator of the queue growth as the travel time between the upstream detector and the stop line is unknown and therefore the moment of time when a vehicle has joined the queue is unknown as well. For example, at the time of arrival of the vehicle at the end of the queue the signalling stage can change and at that moment queue will not exist.

The possible solution to this problem is to collect statistics of estimations under the assumption that during busy hours the presence of a queue is more likely. An algorithm based on this assumption has been developed and is described below.

The basic idea of the algorithm is to collect a set of time differences when signal changes from "red" to "green" and the first car arrives at the downstream detector. After a set of such differences has been collected, it can be analysed statistically. So, the algorithm

consists of two stages. The first is the stage of data collection (time differences) and the second stage is data analysis (extraction of information about T_{PC}). During data collection stage, the algorithm watches the data flow from the SCOOT and extracts relevant pieces of data (differences) from the main data flow. This stage is time-consuming and requires big volume of SCOOT data (not less then one-day data for robust estimation) to be processed as every SCOOT cycle produces not more then one time difference. The data analysis stage involves statistical analysis leading to the identification of the parameter T_{PC}.

The data collected at the first stage necessarily contains some poor estimates representing times measured for vehicles that were not stationary at the time of red-green transition. The set needs to be therefore analysed statistically. The statistical analysis carried by data-processing algorithm, relies on assumption that the queue at a stop line is more common situation than free flow that is not being stopped at the stop line. This assumption appeals to common sense and the fact, that on a busy road normally at least one car is being caught by 'red' signal, which is enough for the algorithm to work properly.

A set of experiments has been carried out on the data collected for four days. Figure 6.15 shows histograms that represent the frequencies of appearance of particular values extracted by the data-collecting algorithm. The histograms clearly show domination of values in the time interval between 2.5 and 7.5 seconds with the maximum frequency at approximately 5 seconds.

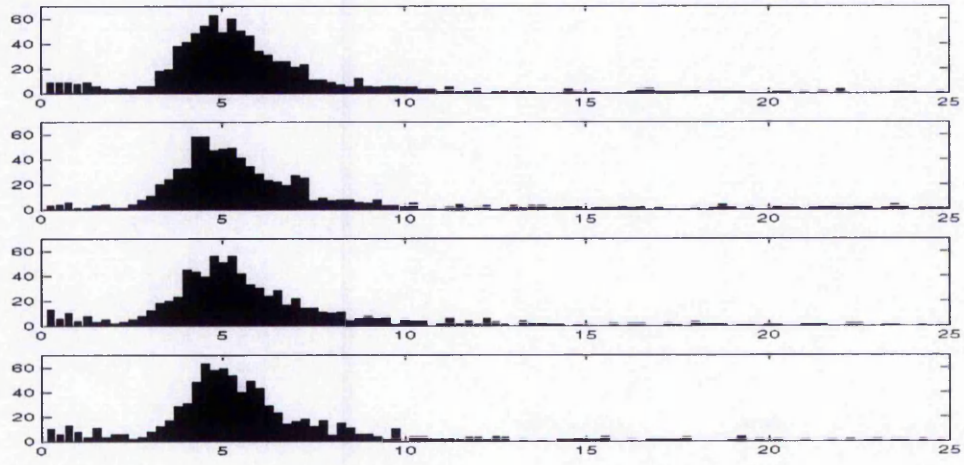


Figure 6.15. Histograms for T_{PC} travel time component estimates (link N60331F-N60421B1, four days data)

Obviously, the above estimates of T_{PC} component cannot be used directly to estimate T_{SL} since T_{PC} and T_T have been estimated for different regimes of traffic flow; T_T is estimated from free-flowing traffic regime and T_{PC} is estimated from stop-and-go traffic regime. In order to translate T_{PC} into equivalent to free-flowing traffic, it has to be corrected taking into account the acceleration of the vehicle from stopped to full-speed moving. This problem requires further investigation and is proposed as an extension of this research.

After both parameters, T_T and T_{PC}, are estimated, the third component, T_{SL}, can be derived using the following relationship (6.1) between the three parameters.

$$T_T=T_{PC}+T_{SL} \tag{6.1}$$

A particular method of obtaining T_{SL} from T_T and T_{PC} will depend on the representation of the estimates of T_T and T_{PC}. For example, T_T and T_{PC} can be represented as fuzzy numbers with membership function estimated from traffic data. Then T_{SL} will be obtained using fuzzy addition.

6.7 Conclusion

In this chapter, SCOOT data processing package has been developed for the data processing environment. A number of traffic data analysis procedures have been carried out as a demonstration of the potential of the developed software.

Travel time methodology has been implemented as a module for the data processing environment. A set of experiments has shown that the methodology indeed works as anticipated. In addition to the point travel time estimation, the algorithm provides additional information, which allows estimation of other useful characteristics of the travel time random variable.

Chapter 7

Conclusions and further research

7.1 Concluding remarks

The main purpose of this research was to investigate techniques for data processing in the area of Transportation Research. Data processing and analysis techniques have been under development for many years but it is apparent that general data analysis is not always sufficient in application to a specific problem.

One particular problem within the area of transportation research has been identified, namely the travel time estimation problem. Although several methodologies have been developed up to date to solve the problem, the solutions are not satisfactory in some applications. As it has been discussed, in modern traffic systems the information about current traffic conditions must be obtained with the maximum possible accuracy and on a real-time basis. Only availability of such information can improve the quality of traffic control. New levels of control, such as trip making decision support, also require most accurate information and assessment of uncertainty associated with such decision making process.

In this research, a novel methodology for travel time estimation has been developed. It is shown to be advantageous over the previously developed techniques and provides travel time estimates with the resolution of traffic data. That means that the accuracy of the estimates is only limited by the underlying measurement hardware. The developed methodology has opened new ways of improving other traffic-related parameters and operational characteristics, such as the origin-destination matrices. The possibilities opened by the methodology developed for the travel time estimation are discussed in Section 7.2.

An investigation of available software tools for data processing and analysis has been carried out. It has shown that most of the available general-purpose data analysis, system modelling and simulation packages do not provide satisfactory functionality to support

real-time traffic data processing. This problem has been addressed with the development of a novel software environment that is free of the identified disadvantages of the existing packages. This software has been successfully used in the simulation study of arrival processes and travel time estimation procedures.

7.2 Further research

Although traffic processes have been investigated extensively, there are still many unsolved problems. This is primarily because neither microscopic nor macroscopic traffic abstractions capture fully the complexity of real-life traffic. Microscopic models deal with modelling detailed behaviour of individual cars and may be considered as numerical methods of solving complex dynamical systems. This makes it impossible to use such models in the context of real-life processes which have undetermined stochastic character. Macroscopic models, on the other hand, deal with aggregated traffic processes as modelling objects. However, the macroscopic level of aggregation frequently renders such models too far removed from reality. They can be used to give approximate view of traffic behaviour, but cannot be used to make accurate short-time predictions of traffic.

The present situation on the roads is such, that traditional approaches of controlling and managing traffic have exhausted their potential. It is apparent, therefore, that new traffic management and control techniques need to be developed. A potential for improvement of the quality of the current traffic management and control seems to be in the use of all available traffic information and utilisation of all layers of control of traffic. The research into the understanding of traffic information has been by this time the use of traditional schemes of data analysis and system modelling. It is now apparent that the transportation research can greatly benefit from the recent advances in the development of information science, such as granular and soft computing, computational intelligence, fuzzy systems and neural networks. Some of these new techniques of handling information have been applied to solving specific transportation problems over two decades ago [Mamdani, Pappis, 1977]. However, there are still many problems solution of which can greatly benefit from the advances in the information science.

In this research, a step towards a better understanding of traffic processes has been made. The developed methodology of travel time estimation allows estimation of travel times with a greater accuracy and also provides additional useful information on traffic patterns. The next step in the development of the methodology will be to investigate the profile of time differences produced by the algorithms of the methodology in order to separate useful travel time measurements and random noise. It is also possible, from the series of time differences, to identify individual vehicles, with some measurable uncertainty, that traverse the link from upstream to downstream detectors. This information will allow tracing individual vehicles across the road network and thus, allow much more accurate estimation of the origin-destination matrices. Assessment of uncertainty associated with such estimation is also facilitated by the developed methodology. Uncertainty, shown to be a multi-dimensional entity, will be quantified not only by its probabilistic appearance (entropy), but also as a confusion, dissonance, nonspecificity and fuzziness. Understanding of uncertainty associated with the information extraction algorithms will enable the tool to manage this uncertainty and use that knowledge for better control of traffic. Summarising the above thoughts, the further research directions can be outlined as follows.

- To investigate the series of time differences produced by the algorithm and develop techniques for extraction of travel time characteristics from the series.
- To investigate the feasibility of the use of the series of time differences for identification of individual vehicles that traverse a link
- To investigate the dimensions of uncertainty associated with the above estimations
- To investigate the ways of reducing the uncertainty or the uses of the knowledge of uncertainty in the decision support system (ATIS)
- Develop novel algorithms based upon the developed travel time estimation methodology for very accurate estimation of the origin-destination matrices.
- Develop a traffic and travel information system that will fully utilise available information on the current state of the traffic processes across whole network, and which will be aimed at providing a new level of control of traffic by affecting trip planning process of individual road users.

This research has also developed a software tool that greatly simplifies a data analysis process. The software can also be used in a range of system modelling activities since it provides a very flexible environment. There are several directions in which the software can be improved and are outlined below.

- Introduction of a hierarchy of modules. That is, a new type of modules will be developed. The modules of the new type will not have a Java implementing class associated with them but will be a system of the generic DPE modules, which can be handled as a module. This hierarchy will greatly reduce the visual complexity of modelled systems.
- Introduction of code editor. Since the Java technology allows run-time linkage of new classes, such classes can be written, compiled and linked into the system right from the system itself. This will eliminate the need for a researcher (or a programmer) to edit and compile the module classes outside the environment.
- At present time, classes cannot be reloaded run-time. This problem requires investigation of the mechanism of Java class loaders and most likely a new class loader will need to be written, which will track the changes of the byte-code of the module classes and re-load the classes whenever it is necessary. This will eliminate the need to restart the system whenever the class of a module has been modified.
- Development of new modules. Several packages will be helpful in the theoretical research outlined above, such as fuzzy reasoning modules, neural network modules, etc.
- A feasibility of the integration of the environment with geographical information systems needs to be investigated. The use of geographical information systems can benefit the traffic analysis and visualisation.

REFERENCES

Adams, J.C., Hummer, J.E., 1993, Effects of u-turns on left-turn saturation flow rate. *Transportation Research Record* 1398, pp 90-100.

Ahmed, S., Blessing, L.T.M., Wallace, K.M., 1999, The relationship between data, information and knowledge based on a preliminary study of engineering designers. *ASME Design Theory and Methodology, Las Vegas, Nevada, USA.*

Aho, A.V., Kernighan, B.W., Weinberger, P.J., 1988, *The Awk Programming Language.* Addison-Wesley, Reading, Massachusetts.

Argile, A., Peytchev, E., Bargiela, A., Kosonen, I., 1996, DIME: A shared memory environment for distributed simulation, monitoring and control of urban areas. *Proc ESS'96,* Genoa, Vol. 1, pp 152-156.

Backer, R,T., Wood, K., 1991, The use of SCOOT detectors to measure traffic flows. *TRRL working paper WP/TO/76, Transport and Road Research Laboratory*, Crowthorne, Berkshire.

Barcelo J. and Ferrer J.L., 1994, Microscopic simulation of vehicle guidance systems with AIMSUN2. *XIIIth Euro Conference, Glasgow.*

Barcelo J., Ferrer J., Grau R, Florian M, Chabini, Le Saux E., 1995, A Route Based Version of the AIMSUN2 Micro-Simulation Model. *Second World Congress on ITS,* Yokohama.

Barcelo J., Ferrer J., Garcia D., Florian M., Le Saux E., 1996, The Parallelisation of AIMSUN2 microscopic simulator for ITS applications. *Third World Congress on ITS,* Orlando.

Bargiela, A., 1998, Uncertainty - A Key For Better Understanding of Systems (Plenary Lecture), *Proc. European Simulation Symposium ESS'98*, pp 11-19.

Bell, M.C., Bennet, L.D., Evans, R.G., 1992, The 'Instrumented city' project: towards an integrated transport database. *Proceedings of the first meeting of the EURO Working Group on Urban Traffic and Transportation*, Landshut, Germany.

Bell, M.G.H., 1991, The estimation of origin-destination matrices by constrained least squares. *Transportation Research B25*, pp 13-22.

Bennett, S., Skelton, J., Lunn, K. *Schaum's Outline of UML*. Schaum's Outlines Series. McGraw-Hill, 2001.

Benz, T., 1994, Traffic Flow Effects of Intelligent Vehicles. *Traffic Technology International.*

Benz, T., 1995, ASIS 4.0 : Flexible Microscopic Traffic Flow Simulation in Motorway and Urban Networks. *Workshop Proceedings "East-West Co-operation in Road Traffic Operations"*, Prague.

Benz, T., 1996, Automatic Distance Keeping in a High Speed Environment - ICC Parameter Design. *Proceedings of the Third ITS World Congress*, Orlando.

Carden, P.J.C, Hounsell, N.B, McDonald, M. 1989, SCOOT model accuracy. *TRRL Contractor Report 153, Transport and Road Research Laboratory,* Crowthorne, Berkshire.

Cascetta, E., 1984, Estimation of trip matrices from traffic counts and survey data: a generalized least squares approach estimator. *Transportation Research B18*, pp 289-299.

Chang, G., Wu, J., 1994, Recursive estimation of time-varying origin-destination flows from traffic counts in feeway corridors. *Transportation Research B,* Vol. 28B, No. 2, pp. 141-160.

Christensen, R., 1981, *Entropy Minimax Sourcebook. Vol. I: General Description. Entropy*. Lincoln, Mass.

Coifman, B., 2001, Improved Velocity Estimation Using Single Loop Detectors. *Transportation Research: Part A*, vol 35, no 10, pp 863-880.

Cornwell, I., Coleman, J. Archer, S., Glen, J. *ITS Integration through a common database*. http://www.mottmac.com/html/02/pdf/its_integration.pdf

Cortes, C., Lavanya, R., Oh, J., Jayakrishnan, R., 2002, General Purpose Methodology for Link Travel Time Estimation Using Multiple Point Detection of Traffic. *Annual meeting of the Transportation Research Board (TRB),* Washington, D.C.

Cowan, R.J., 1975, Useful Headway Models. *Transportation Research*, 9(6), pp 371-375.

Cremer, M., Keller, H., 1987, A new class of dynamic methods for the identification of origin-destination flows. *Transportation Research B21*, pp 117-132.

Dahlgren, J.W., Turner, S.M., Garcia, R.C., 2002, Collecting, Processing, Archiving and Disseminating Traffic Data to Measure and Improve Traffic Performance. *Annual meeting of the Transportation Research Board (TRB),* Washington, D.C.

Dailey, D.J., 1993, Travel time estimation using cross-correlation techniques. *Transportation Research Part B*, 27B (2), pp 97-107.

Dailey, D.J., 1997, Travel time estimates using a series of single loop volume and occupancy measurements. *Transportation Research Board, 76th Annual Meeting, Washington.*

Dailey, D.J., 1999, A statistical algorithm for estimating speed from single loop volume and occupancy measurements. *Transportation Research B*. Vol. 33, pp 313-322.

Dia, H., 1999, Freeway travel time estimation using neural networks. *Processings of the ITSA'99 4th International Conference Smart Solutions at Work,* Adelaide, Australia.

Drew, D.R., 1966, *A study of freeway traffic congestion*. Doctoral dissertation, Texas A&M University, College Station.

Drew, D. R., 1968, *Traffic flow theory and control*. McGraw-Hill, Inc.

Dubois, D., Prade, 1980, H. *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, New York.

Dubois, D., Prade, H., 1985, A review of fuzzy set aggregation connectives. *Information Sciences*, 36, pp 85-121.

Edie, L.C., 1961, Car-following and steady state theory for non-congested traffic. *Operations Research*, vol.9, no.1, pp 66-67.

Evans, R.G., 1995, *The identification and control of recurrent urban traffic congestion using SCOOT data*. PhD thesis, Department of Civil Engineering, University of Nottingham.

Feller, W., 1968, *An introduction to Probability Theory and Its Applications*. Third edition. Volume 1. John Wiley & Sons, Inc.

Florian, M., Chen, Y., 1995, A Coordinate Descent Method for the Bi-level OD Matrix Adjustment Problem. *International Transactions in Operations Research*, Vol. 2, No. 2, pp. 165-175.

Gerlough, D. L., 1955, Use of Poisson distribution in highway traffic. *The Eno Foundation for Highway Traffic Control,* Saugatuck, Conn.

Gerlough, D. L., Barnes, F. C., 1971, The Poisson and other probability distributions in highway traffic. In *Poisson and other distributions in traffic. The Eno Foundation for Transportation*, Saugatuck, Conn.

Goodman, A., 1995-1996, *Introduction to Data Collection and Analysis*. Lecture notes. http://www.deakin.edu.au/~agoodman/sci101/index.html

Gosling, J., McGilton, H., 1996, *The Java Language Environment.* A white paper. http://java.sun.com/docs/white/langenv/index.html.

Grau, R., Babceló, J., 1992, *A review of objective functions in the CARS demand-responsive traffic control system.* Department d'Estadistica i Investigacio Operativa, Secco d'Informatica, Universitat Politecnica de Catalunya, Research Report.

Greenberg, H., 1959, An analysis of traffic flow. *Operations Research*, vol. 7, no.1, pp 79-85.

Greenshields, B. D., 1934, A study in highway capacity. *Highway Research Board, Proc. 14*, pp 448-477.

Hall, A.D., 1962, *A methodology for System Engineering.* D.Van Nostrand Company, Inc. Princeton.

Hall, F.L., 1987, An interpretation of speed-flow-concentration relationships using catastrophe theory. *Transportation Research Part A*, Vol.21A, No. 3, pp.191-201.

Hall, F.L., Persaud, B.N., 1988, Estimation speeds from freeway traffic management systems. *ITE 1988 Compendium of Technical Papers*, pp.166-171.

Hall, F.L., Persaud, B.N., 1989, Evaluation of speed estimates made with single-detector data from freeways traffic managemtnt systems. *Transportation Research Record* 1232, pp 9-16.

Harr, M.E., Leonards, G.E., 1961, A theory of traffic flow for evaluation of the geometric aspects of highways. *Highway Research Board Bulletin*, 308.

Hauer, E., Pagitsas, E., Shin, B.T., 1981, Estimation of turning flows from automatic counts. *Transportation Research Record 795*, pp 1-8.

Hensher, D.A., Johnson, L.W., 1981, *Applied discrete choice modelling*. Croom Helm, New York.

Holm, J., Jensen, T., Nielsen, S.K., 1976, Calibrating traffic models on traffic census results only. *Traffic Eng. Control* 17, 137-140.

Hounsell, N.B., McLeod, F., 1990, ASTRID: Automatic SCOOT Traffic Information System. *TRRL Contractor Report 235, Transport and Road Research Laboratory,* Crowthorne, Berkshire.

Hunt, P.B., Robertson, D.I., Bretherton, R.D., Winton, R.I., 1981, SCOOT - A traffic responsive method of co-ordinating signals. *TRRL Report LR1014. Transport and Road Research Laboratory.*

Jacobson, I., Christerson, M., Jonsson, P., Övergaard, G., 1992, *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley.

Jacobson, I., Booch, G., Rumbaugh, J. 1999, *The Unified Software Development Process*. Addison-Wesley, ACM Press.

JavaBeans Technology™. Sun Microsystems, http://java.sun.com/products/javabeans

Jaynes, E.T., 1979, Prior probabilities. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4, pp 227-241.

Jewell, W.S., 1967, Models for traffic assignments. *Transportation Research* 1, pp 31-46.

Karlin, S., Taylor, H.M., 1975, *A first course in stochastic processes*. Second edition. Academic Press.

Klir, G.J, Wierman, M.J., 1999, *Uncertainty-Based Information: Elements of Generalized Information Theory (Studies in Fuzziness and Soft Computing)*. Springer Verlag.

Klir, G.J., Folger, T.A., 1988, *Fuzzy Sets, Uncertainty and Information*. Prentice Hall, New York.

Kosonen, I., Kokkinen, M., 1992, A new simulation system for traffic signal control evaluation. *ITE 62nd annual meeting,* Compendium of technical papers, Washington D.C., USA, pp 250-254.

Kosonen, I., Niittymäki, J., 1995, Simulation tool for traffic analysis. *High technology in Finland 1995. The Finnish Academy of Technology*. Espoo. Pp 202-203.

Kosonen, I., 1999, *HUTSIM - Urban Traffic Simulation and Control Model: Principles and Applications*. Dissertation for the degree of Doctor of Technology.

Leutzbach, W., 1988, *Introduction to the Theory of Traffic Flow*. Springer-Verlag.

Li, B., De Moor, B., 2002, Dynamic identification of origin-destination matrices in the presence of incomplete observations. *Transportation Research Part B* V. 36, pp 37-57.

Lighthill, M., Whitham, G., 1955, On kinematic waves, a theory of traffic flow on long crowded roads. *Proc. Royal Society, London*, pp 229-317.

Mamdani, E.H., Pappis, C.P., 1977, A fuzzy logic controller for a traffic junction. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-7, pp 707-717.

Mather, M.J., 1983, Inferences on trip matrices from observations on link volumes: a Bayesian statistical approach. *Transportation Research B17*, pp 435-447.

McDonald, M., Hounsell, N.B., 1991, Road Traffic Control: TRANSYT and SCOOT. In *Concise Encyclopaedia of Traffic and Transportation Systems*, ed. M.Papageorgiou, Oxford, Pergamon Press.

Nihan, N, Davis, G., 1987, Recursive estimation of origin-destination matrices from input/output counts. *Transportation Research B21*, 149-163.

Pedrycz, W., 1983, Fuzzy relational equations with generalized connectives and their applications. *Fuzzy Sets and Systems*, 10, pp 185-201.

Pedrycz, W., 1998, *Computational Intelligence: An Introduction*. CRC Press, New York.

Persaud, B.N., Hurdle, V.F. 1988, Some new data that challenges some old ideas about speed-flow relationships. *Transportation Research Record*, 1194, TRB, National Research Council, Washington, D.C., pp 191-198.

Petty, K. F., 1995, *Freeway Service Patrol (FSP) 1.1: The Analysis Software for the FSP Project*, California PATH Research Report, UCB-ITS-PRR-95-20, Institute of Transportation Studies, University of California, Berkeley.

Petty, K., Noeimi, H., Sanwal, K., Rydzewski, D., Skabardonis, A., Varaiya, P., Al-Deek, H., 1996, The Freeway Service Patrol Evaluation Project: Database Support Programs, and Accessibility. *Transportation Research Part C*, vol. 4, no. 3.

Petty, K., Bickel, P., Jiang, J., Ostland, M., Rice, J., Ritov, Y., Schoenberg, F., 1998, Accurate estimation of travel times from single-loop detectors. *Transportation Research, Part B: Methodological*, 32(1), pp 1-17

Peytchev, E.T. 1999, *Integrative framework for discrete systems simulation and monitoring*. PhD dissertation, The Nottingham Trent University.

Pipes, L.A., 1953, An operational analysis of traffic dynamics. *Application in Physics,* 24, pp 274-281.

Pushkar, A., Hall, F.L., Acha-Daza, J.A., 1994, Estimation of speeds from single-loop freeway travel times in real time from flow measurements. *Journal of Transportation Engineering*, 122, pp. 185-191.

Reddy, M.S., 1966, *Quantitative evaluation of the effect of merging vehicles on freeway operation*. Doctoral dissertation, Texas A&M University, College Station.

Reushel, R. 1950, Fahrzeugbewegungen in der Kolonne bei gleichformig beschleunigtem oder verzogertem. Leihffahrzeug. *Z. Osterr. Ing. Arch. Ver.*, 95, pp 52-62, 73-77.

Richmond, Jonathan E.D., 1998, Simplicity and Complexity in Design for Transportation Systems and Urban Forms, *Journal of Planning Education and Research*, Vol. 17, No. 3, pp 220-230

Rosenfield, A., 1971, Fuzzy groups. *Journal of Math. Analysis and Applications*, 35, pp 512-517.

Shafer, G., 1976, *Mathematical Theory of Evidence*. Princeton University Press.

Sims, A.G., Dobinson, K.W., 1979, SCAT: the Sydney coordinated adaptive traffic system - philosophy and benefits. *Proc. Int. Symp. on Traffic Control Systems* 2B, pp 9-42. Institute of Transportation Studies, University of California, Berkeley.

Skabardonis, A., Noeimi, H., Petty, K., Rydzewski, D., Varaiya, P., Al-Deek, H., 1995, Freeway Service Patrol Evaluation. California PATH Research Report, *UCB-ITS-PRR-95-5*, Institute of Transportation Studies, University of California, Berkeley.

Skabardonis, A., Petty, K. F., Bertini, R. L., Varaiya, P. P., 1997, The I-880 Field Experiment: Analysis of the Incident Data. Paper 970121, *The 76th Annual Meeting Transportation Research Board*, Washington, DC.

Spiess, H., 1990, A Gradient Approach for the OD Matrix Adjustment Problem, *Publication No. 693*, Centre de Recherche sur les Transports, Université de Montréal.

Steenbrick, Peter A., 1974, *Optimization of transport networks*. John Wiley & Sons Ltd.

Swing™ package. Sun Microsystems. http://java.sun.com/products/jfc

Taylor, M.A.P., Young, W., Bonsall, P.W., 1996, *Understanding Traffic Systems: Data, analysis and presentation*. Ashgate Publishing Limited.

The MathWorks, Incorporated. http://www.mathworks.co.uk

Thompson, R.G., 1989, A sampling technique for origin-destination surveys. *Australian Road Research* 19 (3), pp 230-233.

Troutbeck, R.J., 1993, Effect of heavy vehicles at Australian traffic circles and unsignalised intersections. *Transportation Research Record* 1398, pp 54-60.

Van der Zijp, N.J. and Hamerslag, R., 1996, Improved Kalman Filtering Approach for Estimating Origin-Destination Matrices for Freeway Corridors. *Transportation Research Record 1443*, pp.54-64.

Van Zuzlen, H.J., Willumsen, L.G. 1980, The most likely trip matrix estimated from traffic counts. *Transportation Research* B14, pp 281-293.

Visual Solutions, Incorporated. http://www.vissim.com

Wang, Z., Klir, G.J., 1992, *Fuzzy Measure Theory*. Plenum Press.

Wardrop, J., 1952, Some Theoretical Aspects of Road Traffic Research. *Proc. Inst. Civil Engineers*, Part 2, pp. 325-378.

Weaver, W., 1948, Science and Complexity. *American Scientist*, 36, pp 536-544.

Williams, P.M., 1980, Bayesian conditionalisation and the principle of minimum information. *British Journal for the Philosophy of Science*, 31, pp 131-144.

Wilson, A.J., 1967, A statistical theory of spatial distribution models. *Transportation Research* 1, pp 253-269.

Wirasinghe, S., 1978, Determination of traffic delays from shock-wave analysis. *Transportation Research*, 12, pp 343-348.

Zadeh, L. A., 1965, Fuzzy sets. *Information and Control*, 8, pp 338-353.

Zadeh, L.A., 1971, Similarity relations and fuzzy orderings. *Information Sciences*, 3, pp 177-200.

Appendix A

DataTaker and DataGiver interfaces

This appendix provides a source for the DataTaker and DataGiver interfaces which are the basis of any functional module, written for the data processing environment. In order to make a module recognisable by the system, the module's class should implement one of or both of the above interfaces.

```
/*
 * @(#)DataGiver.java 1.0 25/06/2002
 *
 * Copyright 2002 E.Agafonov. All rights reserved.
 * PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 */

package edu.tntu.ism.traffic;

import java.util.*;
import edu.tntu.ism.traffic.*;

/**
 * This interface specifies methods and members that the user's module
 * should implement in order to make it recognisable by the system
 * as a data provider
 *
 * @version 1.0 25/06/2002
 * @author Eugene Agafonov
 *
 * @see DataTaker
 * @see Structured
 * @see Parameters
 * @see DataMessage
 */

public interface DataGiver extends Structured {
    /**
     * Variable in the parameters that specified the module's name
     * (it will be used as a label)
     */
    String GIVER_NAME="Name";

    /**
     * @return number of output sockets of the module
     */
    int giverSocketsNumber();

    /**
     * @return type of the specified socket
     */
    String giverGetSocketType(int socket);

    /**
     * Returns the <code>Parameters</code> object of this
     * <code>DataGiver</code>
     *
     * @return <code>Parameters</code> object
     */
    Parameters giverGetParameters();

    /**
     * Initialisation procedure. An extrnal <code>Parameters</code> object can
     * be passed into the <code>DataGiver</code> so as to use for its own
     * parameters (it should be passed unlocked but the <code>DataGiver</code>
     * can lock it after filling it in)<p>
     *
```

```
     * One of the initialise procedures should be called before the
     * object is engaged into functioning<p>
     *
     * @param external external <code>Parameters</code> object to be
     *        initialised with.
     */
    void initialiseGiver(Parameters external);

    /**
     * This procedure would normally not be called from within the system
     * and is left for backward compatibility only. The typical implementation
     * would look like the following:<p>
     * <blockquote>
     * <pre>
     * public void initialiseGiver(){
     *     initialiseGiver(new Parameters());
     * }
     * </pre>
     * </blockquote>
     */
    void initialiseGiver();

    /**
     * This method is fundamental for the <code>DataGiver</code> module. It will be
     * called by the system in order to retrieve data the module
     * wants to pass further on. The method should return either <code>null</code>
     * if data is not available or an array of <code>DataMessage</code> objects
     * whose elements will correspond to the output sockets of the module
     * as follows:<p>
     * <pre>
     *   [0] -> socket #0
     *   [1] -> spcket #1,
     *       ... etc
     * </pre><p>
     * where [N] represents the elements of the resulting array. <p>
     * The elements of the resulting array must be either null, if
     * the module does not wish to pass any data to the corresponding
     * output, or objects of <code>DataMessage</code> class with data
     * of proper type.
     *
     * @return array of <code>DataMessage</code> objects, each element to
     *         each output socket
     * @exception DataGiverDone thrown when this module has exhausted its data
     *         resources (reached end of file or similar conditions)
     * @exception InterruptedException if reading operation has been
     *         interrupted
     */
    DataMessage [] readNext() throws InterruptedException, DataGiverDone;

    /**
     * Resets the module. A chance to re-set its internal variables
     * and validate the parameters
     */
    void giverReset();

    /**
     * Called upon disposal of the module. Allows to do some final cleaning-up
     * (stop executing threads, closing streams etc)
     */
    void giverDispose();
}

/*
 * @(#)DataGiver.java 1.0 13/02/2002
 *
 * Copyright 2002 E.Agafonov. All rights reserved.
 * PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 */

package edu.tntu.ism.traffic;

/**
 * This interface specifies methods and members which every data
 * consumer (receiver) should implement in order to be recognisable by
 * the system as a data receiver
```

```
 *
 * @version 1.1 12/07/2002
 * @author Eugene Agafonov
 *
 * @see DataTaker
 * @see Structured
 * @see Parameters
 * @see DataMessage
 */

public interface DataTaker extends Structured {
    /**
     * Variable in the parameters that specified the module's name
     * (it will be used as a label)
     */
    String TAKER_NAME="Name";

    int TAKER_NO_COMMAND=-1;     // no commands for the data source
    int TAKER_PAUSE=1;           // pause the data source
    int TAKER_RESUME=2;          // resume data source operations
    int TAKER_DISCARD=3;         // discard mode of the reader... it stops
                                 // forwarding data to this taker but does not
                                 // stop functioning...

    /**
     * Returns the number of input sockets available for the
     * system. The number of input sockets is a structural characteristic
     * and whenever this changes, all registered
     * <code>StructureChangeListener</code>-s should be notified.
     *
     * @return the number of input sockets
     * @see StructureChangeListener
     */
    int takerSocketsNumber();

    /**
     * Returns the type of the specified <code>socket</code> encoded as a
     * String
     *
     * @param socket the socket ID whose type is enquired about
     * @return String object that is the encoded type or null if
     *         socket ID is out of range
     */
    String takerGetSocketType(int socket);

    /**
     * Tests if the specified data type will be accepted on the
     * specified socket.
     *
     * @param socket ID of the socket that is being tested for type
     *        compliance
     * @param type <code>String</code>-encoded type that is tested
     *        for compliance with the socket type
     * @return false if data type is not acceptable or socket ID is
     *        out of range and true otherwise
     */
    boolean takerAcceptable(int socket, String type);

    /**
     * Returns the <code>Parameters</code> object of this
     * <code>DataTaket</code>
     *
     * @return <code>Parameters</code> of the <code>DataTaker</code>
     * @see Parameters
     */
    Parameters takerGetParameters();

    /**
     * Initialisation procedure. An extrnal <code>Parameters</code> object
     * can be passed into the <code>DataTaker</code> so as to use for its
     * own parameters (it should be passed unlocked but the dataGiver can
     * lock it after filling it in)<p>
     *
     * The implementation should make use of this object instead of creating
     * its own in order to make the parameters persistent. Otherwise
     * the restoration of the state (save and load) of the module would
     * not be possible
```

```
     *
     * @param extenal external Parameters object
     */
    void initialiseTaker(Parameters external);

    /**
     * This procedure would not normally be called from within the system
     * and is left for backward compatibility only. The typical implementation
     * would look like the following:<p>
     * <blockquote>
     * <pre>
     * public void initialiseTaker(){
     *     initialiseTaker(new Parameters());
     * }
     * </pre>
     * </blockquote>
     */
    void initialiseTaker();

    /**
     * This is the interface procedure which is used by the system
     * to provide the module with new data
     *
     * this procedure should return as quick as possible
     *
     * @param data <code>DataMessage</code> object containing new data
     * @param socket the socket in which data is sent to
     * @return commands for each socket. The command will be passed through
     *         to the module-source of data
     * @throws InterruptedException
     */
    int [] processNext(DataMessage data, int socket) throws InterruptedException;

    /**
     * Called whenever the system wants the module to reset its state
     */
    void takerReset();

    /**
     * Called before the module is disposed of from the system. Allows
     * the module to do some clean up
     */
    void takerDispose();
}
```

Appendix B

Example screens of the data processing environment

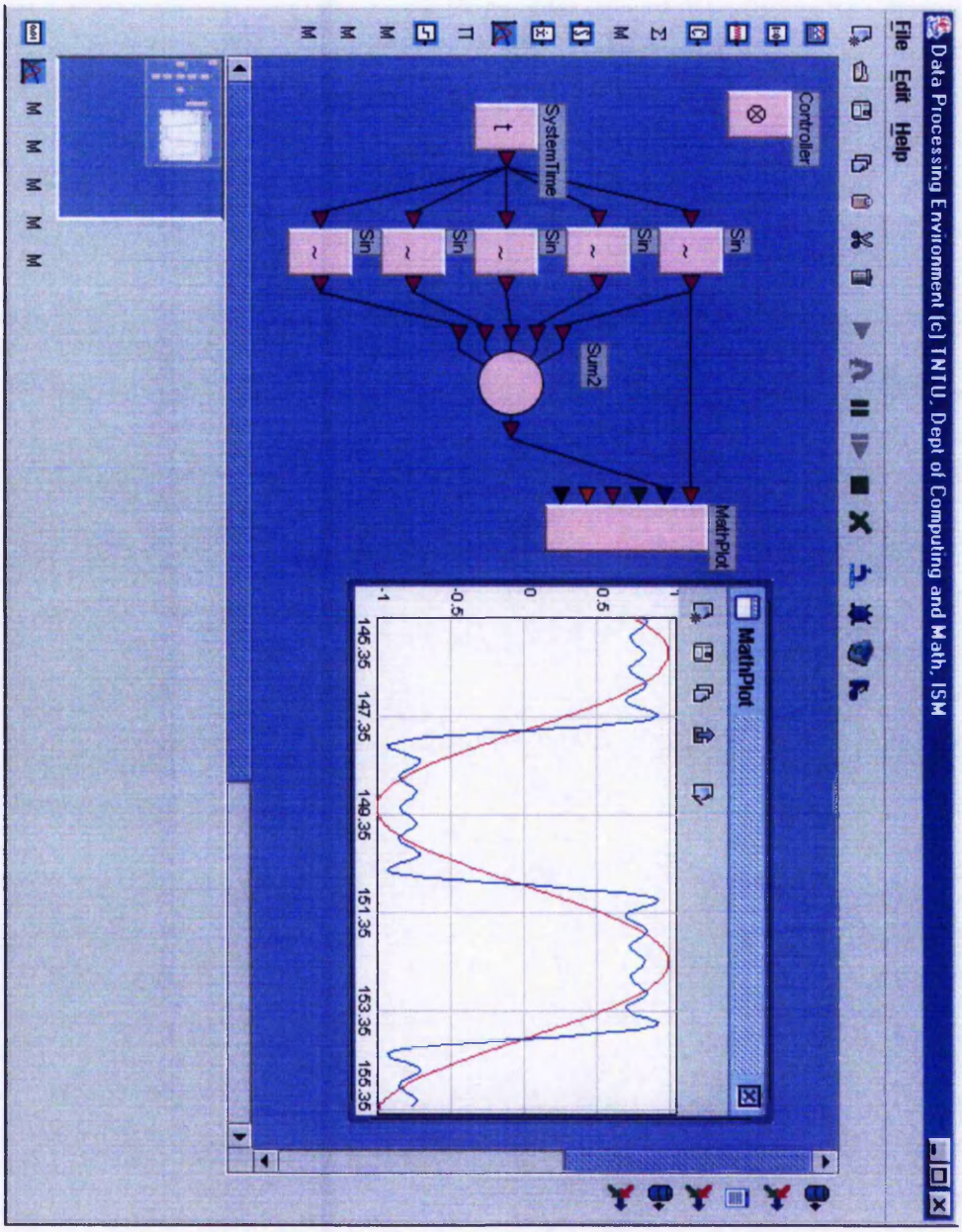


Figure B.1. A model implementing the discrete approximation of a step function

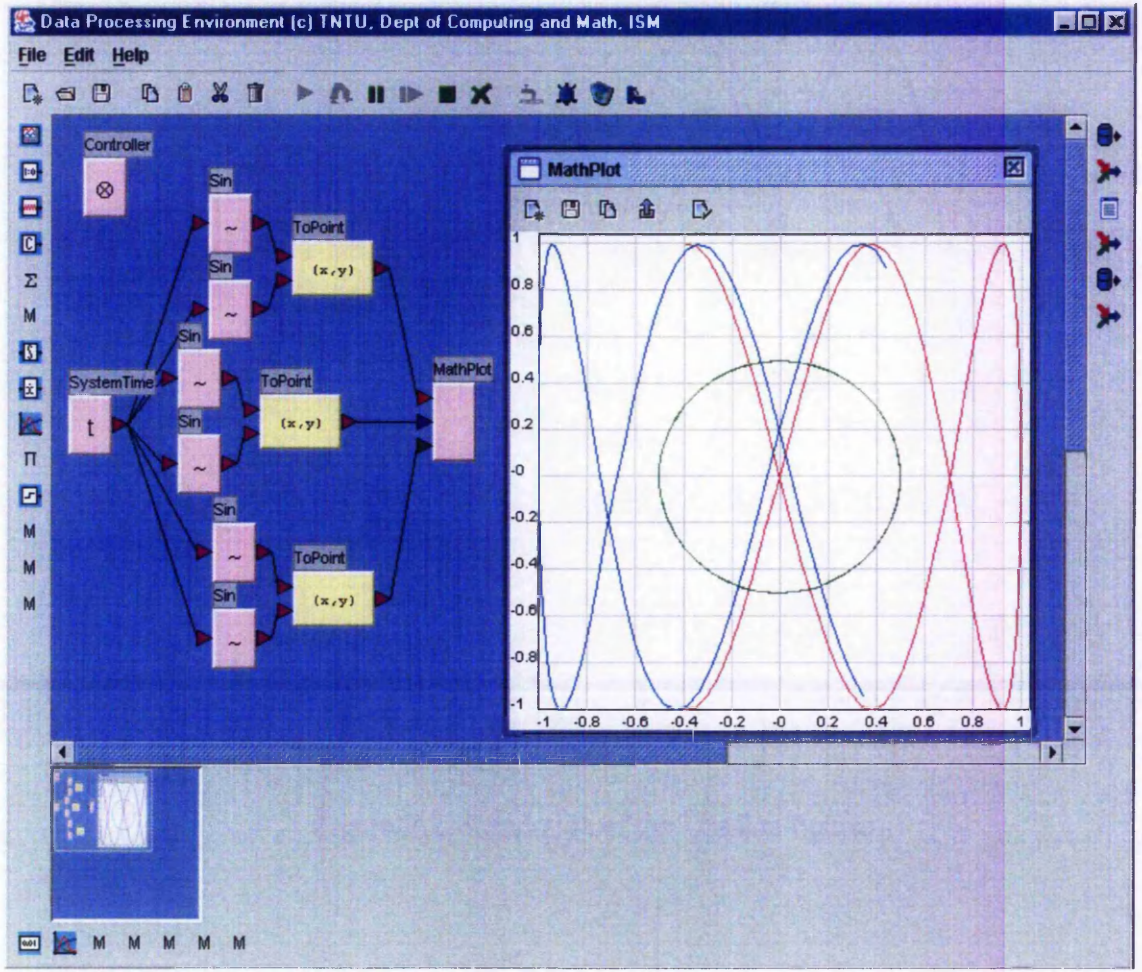


Figure B.2. Simulation of the Lissajou figures

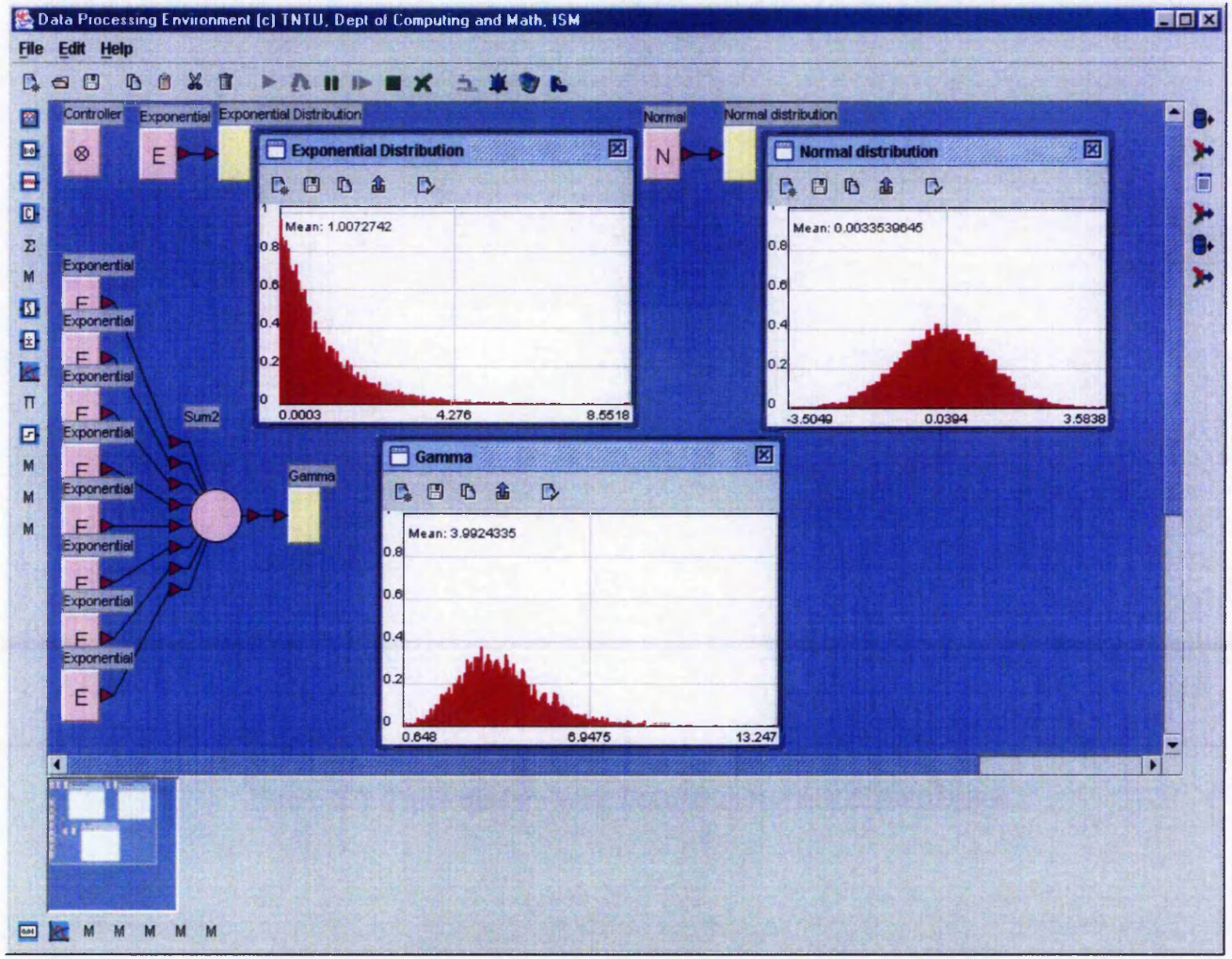


Figure B.3. Three random variables and histograms demonstration

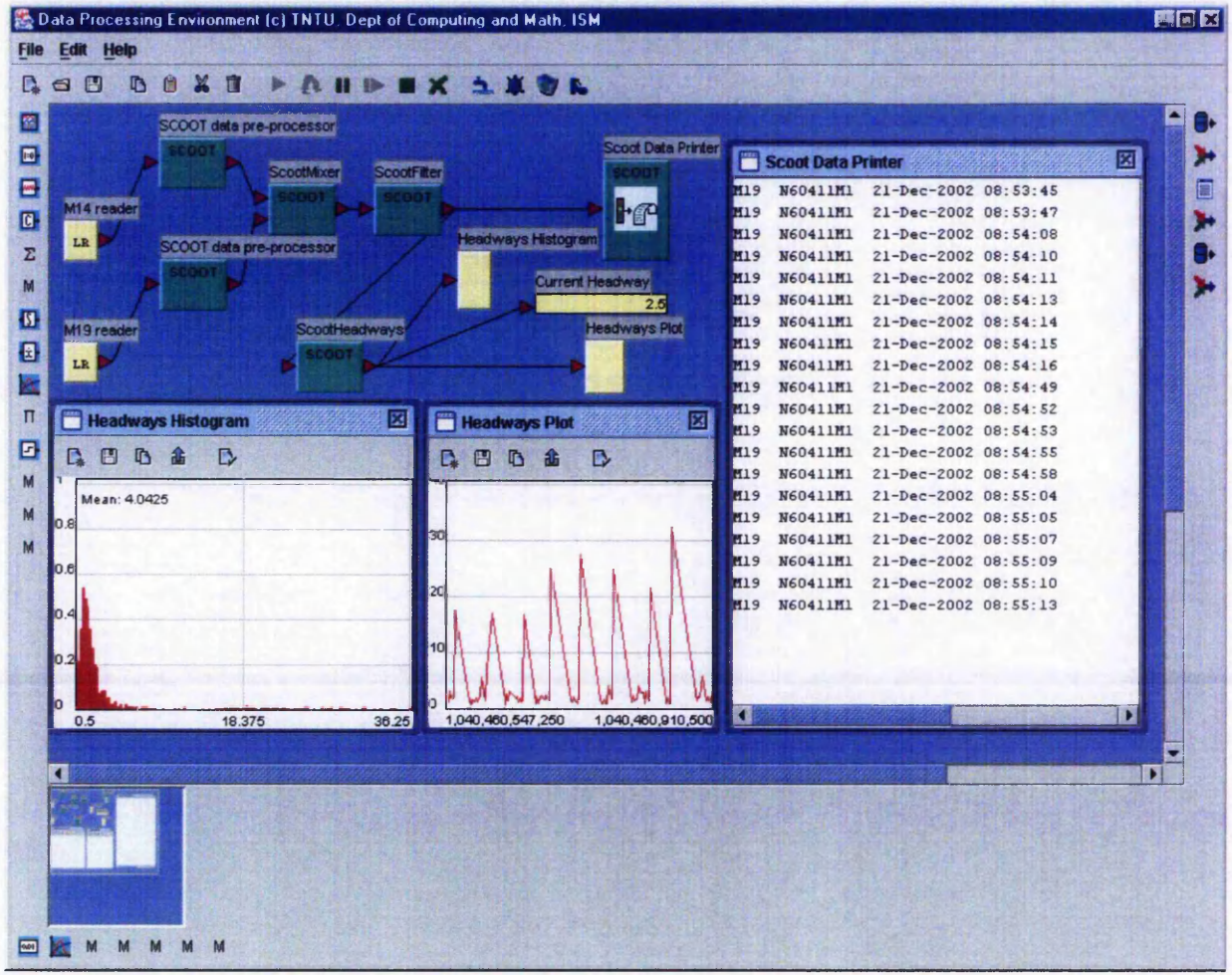


Figure B.4 A model implementing the headways analysis procedure: visualisation is done via Histogram, Plot and Display.

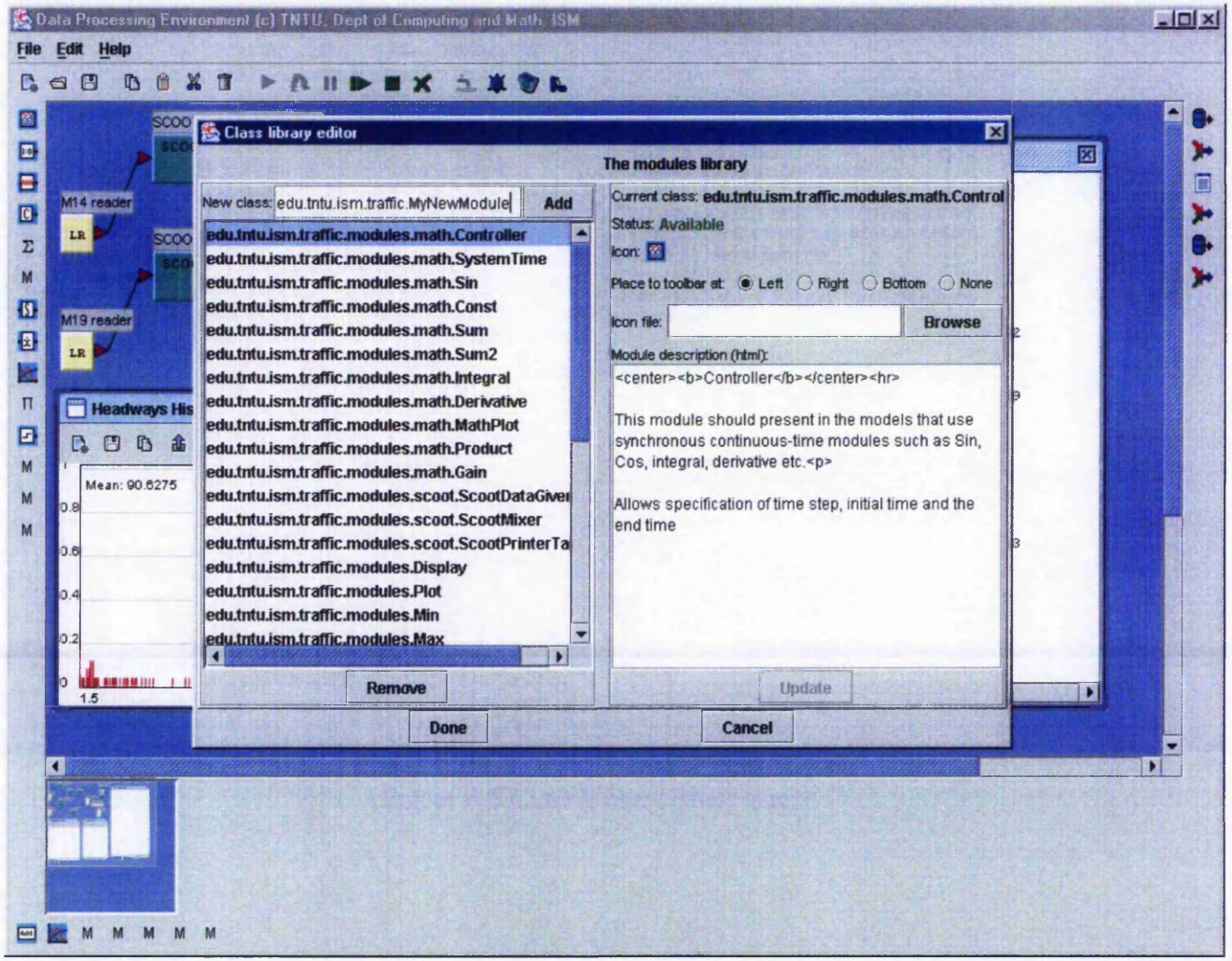


Figure B.5 Class Library editor screen

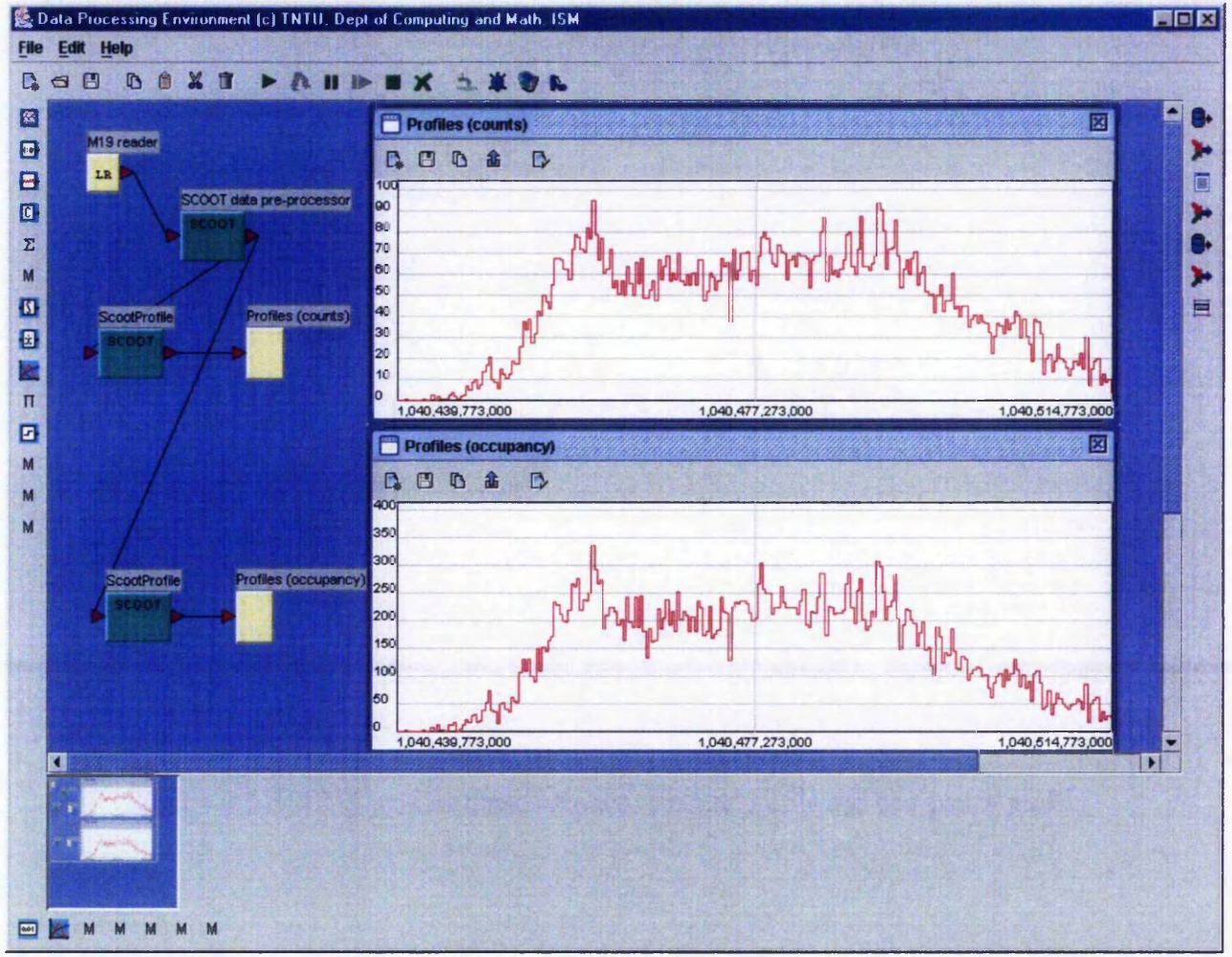


Figure B.6 SCOOT profiles demonstration: counter profile and occupancy profile

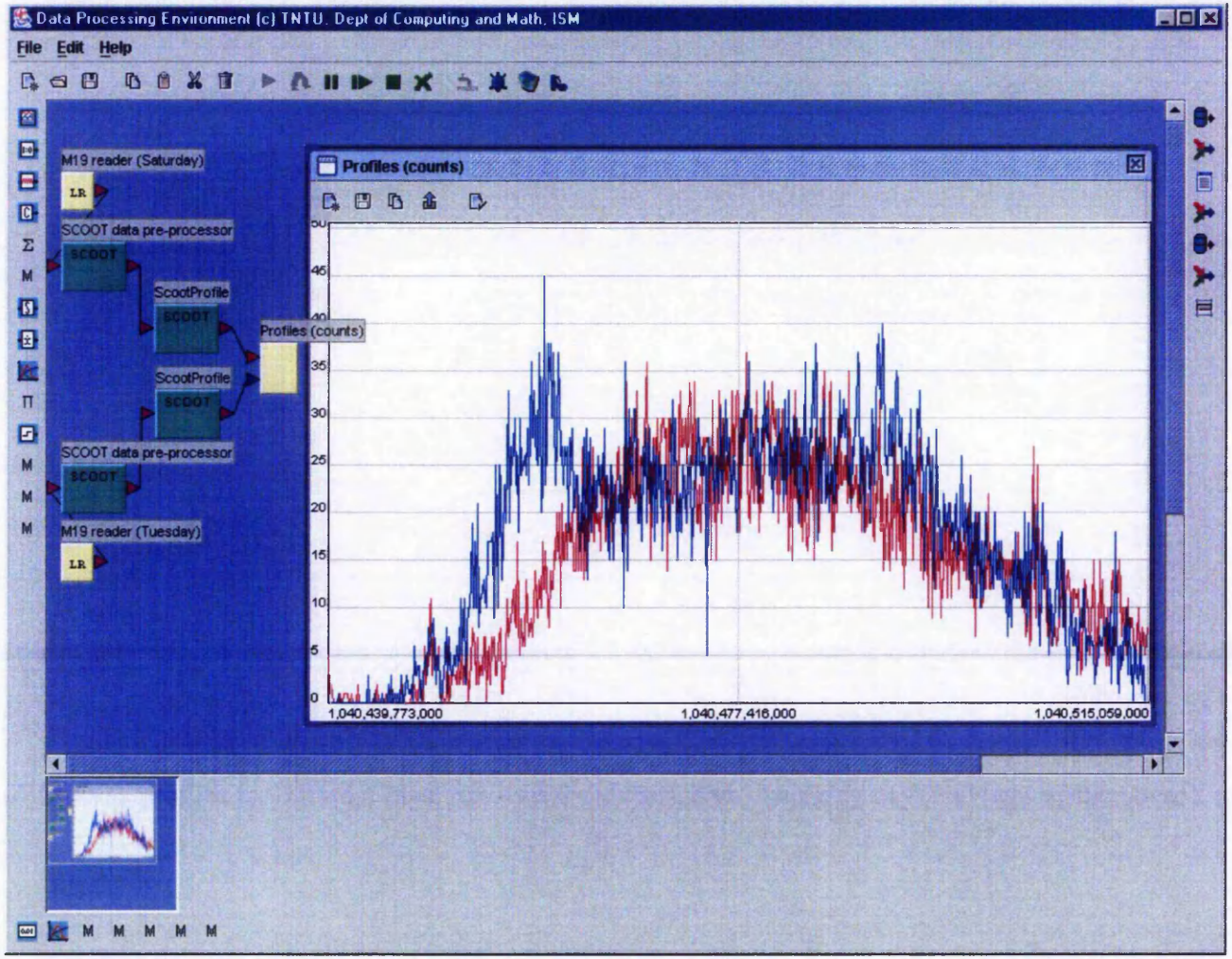


Figure B.7 SCOOT profiles for different days: red – weekend (sat), blue – working day (tue) (aggregation over 2 minutes)

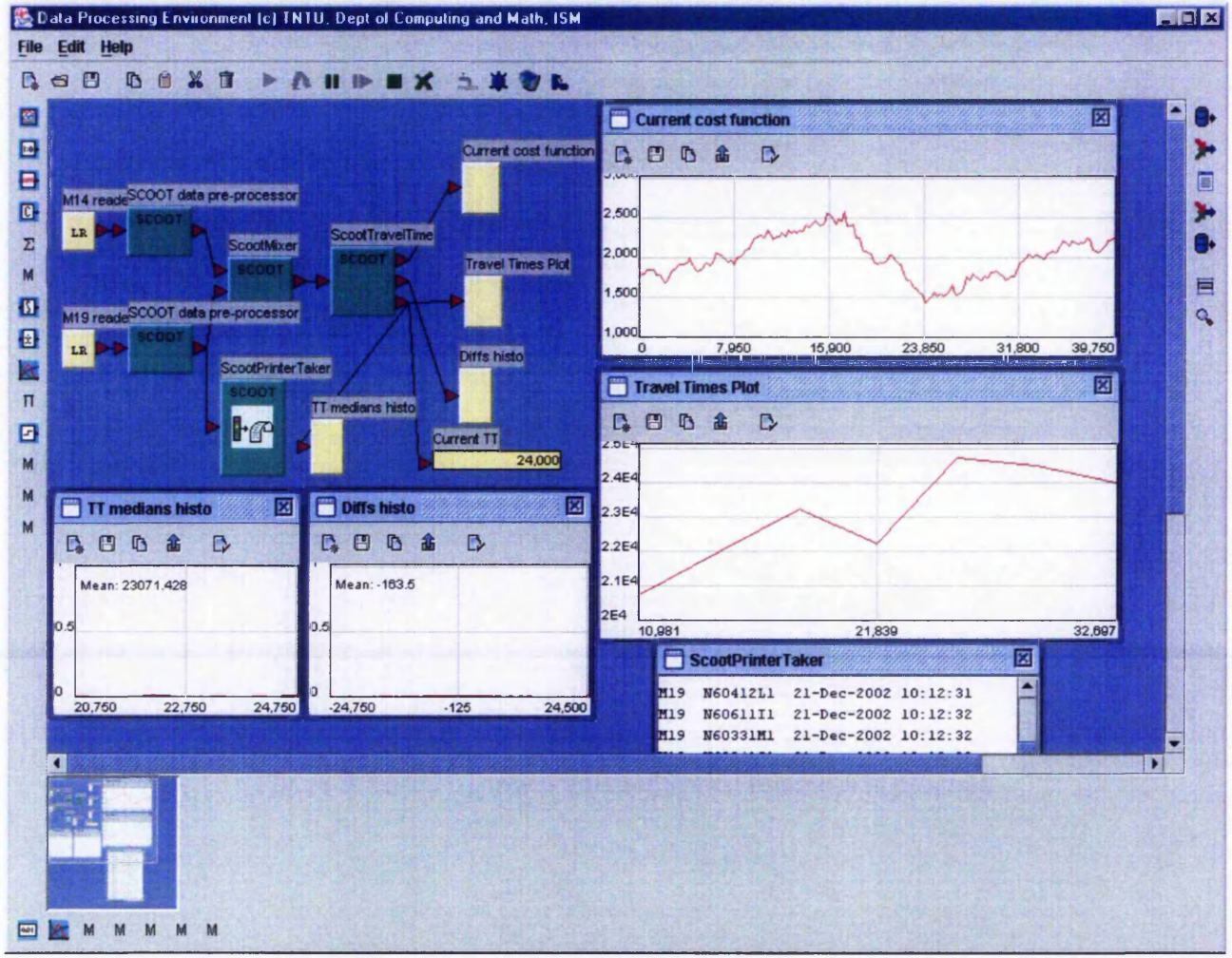


Figure B.8 SCOOT travel time estimation routine is in progress

Appendix C

Examples of implementation of modules for the DPE

Example 1. Display module source

```
/*
 * @(#)Display.java  1.0 22/08/2002
 *
 * Copyright 2002 E.Agafonov. All rights reserved.
 * PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 */

package edu.tntu.ism.traffic.modules;

import java.text.*;
import java.io.*;

import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;

import edu.tntu.ism.traffic.*;
import edu.tntu.ism.traffic.ui.*;

/**
 * Display is a module that prints out the values of the incoming
 * signal to its shape object (see {@link edu.tntu.ism.traffic.ui.ModuleLook#getShape()}
 *
 * @version 1.0 22/08/2002
 * @author Eugene Agafonov
 */

public class Display extends DefaultModuleLook
                          implements DataTaker, Indicator, Runnable {
    final static Font displayFont=new Font("Arial", Font.PLAIN, 10);
    Parameters params;

    protected JLabel label;
    protected double value;

    private NumberFormat formatter = NumberFormat.getInstance();

    private static final int WIDTH=100, HEIGHT=15;

    /**
     * Constructs a new object of Display class
     */
    public Display(){
        super(new JLabel());
        formatter.setMaximumFractionDigits(9);
        label=(JLabel)shape;
        label.setOpaque(true);
        label.setBackground(Color.WHITE);
        label.setBorder(new LineBorder(Color.BLACK,1));
        label.setFont(displayFont);
        label.setHorizontalAlignment(JLabel.RIGHT);
        label.setVerticalAlignment(JLabel.CENTER);
    }

        // the number of sockets for Display is always 1
    public int takerSocketsNumber(){
        return 1;
    }

        // data type we will expect in our socket -
        // the name of Number class
    public String takerGetSocketType(int socket){
```

```
        return Number.class.getName();
    }

        // Check if the proposed to connect data type is acceptable for us
    public boolean takerAcceptable(int socket, String type){
        if (socket!=0) return false;
        try {
            Class c=Class.forName(type);
            return Number.class.isAssignableFrom(c);
        }catch(Exception e){
        }
        return false;
    }

        // return our parameters object
    public Parameters takerGetParameters(){
        return params;
    }

        // necessary initialisation procedure before we become operational
    public void initialiseTaker(Parameters external){
        params=external;
        takerReset();
    }

        // initialisation with our own parameters
    public void initialiseTaker(){
        initialiseTaker(new Parameters());
    }

        // call for us to reset our internal state
    public synchronized void takerReset(){
        value=0.0;
        SwingUtilities.invokeLater(this);
    }

        // disposal procedure - we have nothing to dispose of
    public void takerDispose(){
    }

        // here we get a new piece of data and paint it on our look object
    public synchronized int [] processNext(DataMessage data, int socket){
        if (socket!=0) return null;

        value=((Number)data.getData()).doubleValue();
            // call for repaint
        SwingUtilities.invokeLater(this);
        return null;
    }

        // we are structured but our structure doesn't change
    public void addStructureChangeListener(StructureChangeListener scl){
        // doesn't ever change...
    }
    public void removeStructureChangeListener(StructureChangeListener scl){
        // doesn't ever change...
    }

/*************************************
 *    Extension of the default look    *
 *************************************/

    public Dimension getSize(int orientation){
        return new Dimension(WIDTH, HEIGHT);
    }
    public boolean isSticky(){
        return false;
    }
    public Icon getIcon(){
        return UITools.getIconResource("display.gif");
    }
    public ModuleLook getLook(){
        return this;
    }

        // this is used to repaint our look object. We can only do it from the
        // GUI event dispatching thread
```

```
    public void run(){
        label.setText(formatter.format(value));
    }
}
```

Example 2. The use of abstract classes to simplify modules development process

The package of continuous systems modelling and simulation provides several abstract classes which immensely simplify the development of new modules. In this appendix, a module that implements mathematical **sin(*kt*+*f*)** function by extending an abstract class `UnaryOperator` is presented. It is clear that most of the code is devoted to handling the parameters, whether the real module's functionality is given in a single line of code.

```
/*
 * @(#)Sin.java  1.0 22/08/2002
 *
 * Copyright 2002 E.Agafonov. All rights reserved.
 * PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 */

package edu.tntu.ism.traffic.modules.math;

import edu.tntu.ism.traffic.*;
import edu.tntu.ism.traffic.ui.*;
import javax.swing.*;

/**
 * Mathematical sine module. Extends UnaryOperator and implements its own function
 *
 * @version 1.1 16/09/2002
 * @author Eugene Agafonov
 */

public class Sin extends UnaryOperator {

    private static final String PHASE="Initial phase";
    private static final String COEFF="Coefficient";
    private static final String FREQ="Frequency (omega)";

    private double phase, coeff, freq;

    public Sin() {
        super();
    }

    protected void initParameters(){
        params.addDouble(COEFF, 1.0, true);
        params.addDouble(PHASE, 0.0, true);
        params.addDouble(FREQ, 1.0, true);
    }

    protected void updateParameters(){
        try {
            phase=params.getDouble(PHASE).doubleValue();
        }catch(Exception e){
            phase=0;
        }
        try {
            coeff=params.getDouble(COEFF).doubleValue();
        }catch(Exception e){
            coeff=1.0;
        }
```

```
        try {
            freq=params.getDouble(FREQ).doubleValue();
        }catch(Exception e){
            freq=1.0;
        }
    }

        // this is the functionality of this module
    protected double result(double value){
        return coeff*Math.sin(freq*value+phase);
    }

    private final Icon icon=UITools.getIconResource("mathSin.gif");
    private final ModuleLook look=new MathModuleLook('\u223C', icon);
    public ModuleLook getLook(){
        return look;
    }
}
```

Example 3. Implementation of SCOOT headways module

```
/*
 * @(#)ScootHeadways.java 1.1 28/08/2002
 *
 * Copyright 2002 E.Agafonov. All rights reserved.
 * PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 */

package edu.tntu.ism.traffic.modules.scoot;

import java.io.*;
import java.util.*;

import java.awt.*;
import javax.swing.*;

import edu.tntu.ism.traffic.*;
import edu.tntu.ism.traffic.ui.*;
import edu.tntu.ism.traffic.modules.scoot.data.*;

/**
 *
 * This class implements a module that produces series of headways for
 * certain link based on M19 data. The link name (with detector) should be
 * provided as a parameter
 *
 * @version 1.0 21/12/2002
 * @author Eugene Agafonov
 */

public class ScootHeadways implements
                DataGiver, DataTaker, Indicator, ParameterChangeListener {
        // only one parameter: the name of a detector
    private final static String LINK="Link name (with detector)";

        // for synchronization purpose
    private final Object locker=new Object();

    private Parameters params;

    private long previousTime, headway, time;
    private String linkName;

        // the module has only 1 input socket
    public int takerSocketsNumber(){
        return 1;
    }

        // the data type will be expected to be ScootData.class.getName()
    public String takerGetSocketType(int socket){
        if (socket!=0) return null;
        return ScootData.class.getName();
```

```
}

    // the module only accepts ScootData on its input
public boolean takerAcceptable(int socket, String type){
    if (socket!=0) return false;
    try {
        Class c=Class.forName(type);
        return ScootData.class.isAssignableFrom(c);
    }catch(Exception e){
    }
    return false;
}

    // returns parameters
public Parameters takerGetParameters(){
    return params;
}

    // initialisation procedures
public void initialiseTaker(){
    initialiseTaker(new Parameters());
}

public void initialiseTaker(Parameters external){
    params=external;
    params.addParameterChangeListener(this);
    params.addString(LINK, "", true);
    takerReset();
}

    // reset the internal state of the module
public void takerReset(){
    previousTime=-1;
    headway=-1;
    try {
        linkName=params.getString(LINK);
    }catch(Exception e){
        linkName="";
    }
}

    // We have receive a data object on our input. The data
    // is being checked for being of M19 message and from a
    // detector of interest. Then a headway is being calculated
    // and pushed on to our data providing part
public int [] processNext(DataMessage data, int socket)
                                    throws InterruptedException {
    if (socket!=0) return null;

        // wrong data type
    if ( !(data.getData() instanceof ScootDataM19)) return null;
    ScootDataM19 m19=(ScootDataM19)data.getData();

        // check if this is from our link...
    if (!m19.getNode().getRaw().equals(linkName)) return null;

    if (previousTime>0){
        synchronized(locker){
            while(headway>=0) locker.wait();

            headway=m19.getTime()-previousTime;
            time=previousTime;
            locker.notify();
        }
    }
    previousTime=m19.getTime();

    return null;
}

public void takerDispose(){
    params.removeParameterChangeListener(this);
}
```

```
    /*************************************************************************/
    /*                          data providing part                          */
    /*************************************************************************/

        // send current headway out as Float value (in seconds)
    public DataMessage [] readNext() throws InterruptedException, DataGiverDone {
        DataMessage [] res=null;

        synchronized(locker){
            while(headway<0) locker.wait();
            res=new DataMessage[] {
                new DataMessage(this, new Float(headway/1000.0f),
                                        Number.class.getName(), time)
            };
            headway=-1;
            locker.notify();
        }

        return res;
    }

        // the module has only 1 output
    public int giverSocketsNumber(){
        return 1;
    }

        // the output data type is numerical (object of Number class)
    public String giverGetSocketType(int socket){
        if (socket!=0) return null;
        return Number.class.getName();
    }

        // paramteres
    public Parameters giverGetParameters(){
        return params;
    }

        // these are redundant, since initialisation will be done in
        // receiving part
    public void initialiseGiver(){
    }
    public void initialiseGiver(Parameters external){
    }
    public void giverReset(){
    }
    public void giverDispose(){
    }


        // out look is an instance of ScootModuleLook object
    private final ModuleLook look =
                new ScootModuleLook(UITools.getIconResource("scootheadways.gif"));
    public ModuleLook getLook(){
        return look;
    }

        // structure of the module doesn't change
    public void addStructureChangeListener(StructureChangeListener scl){
    }
    public void removeStructureChangeListener(StructureChangeListener scl){
    }
    protected void fireStructureChangeEvent(StructureChangeEvent e){
    }

        // ParameterListener: if a change in the parameters occur
        // the module resets its state according to the parameters
    public void parameterChanged(ParameterChangeEvent pce){
        takerReset();
    }
}
```