MCMC Simulation for Modelling Airline Passenger Choice Behaviour

By

Fajer A. Al-Sayer, B.Sc. M.Sc.

A thesis submitted in fulfilment of the requirements for the Master of Science Degree in Computer Applications

August 2001

School of Computer Applications
Dublin City University
Dublin 9, Ireland

Supervisor: Dr. Alistair Sutherland

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of M. Sc. Is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:  Date: 28/9/2001

Acknowledgements

Thanks to the supervisor Dr. Alistair Sutherland for his continuous assistance and support. Thanks to Dr. Martin Crane for his help and valuable suggestions. A special thanks to Kuwait Airways for giving me the opportunity to continue my studies.

Many thanks to all the following:

My parents and my brothers and sisters for encouraging me and giving me moral support.

My lovely nieces Anood, Baidaa and Noor for being so kind and supportive to their aunt.

My friends back home, Altaf, Huda, Jameelah, Laila, Hana and Mona for keeping in touch with me all the past few years. A very special thanks has to go to my friend Selma in Canada.

My colleagues in the Toastmaster club for helping me to practice public speaking in my spare time.

And last but not least, my friends in Ireland: Vincent Shannon from Malahide, Kay and Jerry Chester from Swords, Kerry and Pete Bedell from Malahide, Heidi and Trevor Sargant from Ballbriggan for treating me as one of the family.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

Abstract

As passengers we would prefer to pay the cheapest fare available for our airline ticket. On the other hand airline companies wish to increase their revenue from its flown tickets. During the booking process of an airline flight, some passengers may arrive early to book their seats, others may decide to book just few days before departure or even on the day of departure. Airlines realise that they have to offer a variety of fares in order to differentiate between different types of passengers. Allocating seats to different fare classes for different types of passengers in such a way that would maximise the airline's revenue requires yield/revenue management systems. There are two main steps in any revenue management system: Forecasting and Optimisation. Accurate prediction of passenger future demand for different fare classes improves the seat allocation recommendations resulting from the optimisation step. The work in this thesis concentrates on studying and analysing the behaviour of different passenger types towards different fare classes. We first formulate a Monte Carlo simulation model for the booking process. The model generates sample booking data for a flight on different fare classes by different types of passengers defined by the characteristics which affect their behaviour. Passenger behaviour is modelled using a customer utility function and a multinomial logit (logistic) model of demand. This sample booking data is then used in a Markov Chain Monte Carlo model in order to estimate the passenger choice parameters used in generating the booking data. These estimated parameters could be used then to classify any new booking data. The MCMC model uses the Metropolis Algorithm for its estimation process. We also examine briefly the computational feasibility of our approach using parallel processing.

CHAPTER 1

INTRODUCTION

Modelling passenger choice behaviour is becoming more and more important in the process of future demand forecasting. Airlines like any service and product provider, realise that it is essential to predict passenger behaviour towards different fare classes on different flights in order to set the future prices and product availability. During the days prior to the departure day of the flight, passengers make their reservations and choose from a set of options of flights and fare classes. Passengers, like any other consumers choosing between a number of available products, have different preferences towards the products on offer. In this research we model the passenger choice behaviour during the flight booking process using a computer simulation model.

Bratly and Scharge (1987) provide the following definition of modelling:

> *A model is a description of some system intended to predict what happens if certain action is taken. Virtually any useful model simplifies and idealizes.....For a model to be useful, it is essential that, given a reasonably limited set of descriptors, all its relevant behaviour and properties can be determined in a practical way: analytically, numerically or by deriving the model with certain (typically random) inputs and observing the corresponding outputs. This process is called simulation.*

While Neelamkavil (1986) gave the following definitions:

> *A model is a simplified representation of a system (or process or theory) intended to enhance our ability to understand, predict, and possibly control the behaviour of the system.... A model adapted for simulation on a computer (i.e. mathematical/logical relations and operational rules built into the computer program) is known as a computer simulation or simply simulation model.... Modelling is the process of establishing*

interrelationships between important entities of a system.....It is almost impossible to understand and isolate all the interrelationships in a real-world system, and one is forced to trade off reality, generality and accuracy for simplicity.... Obviously the ability to build models by selecting the smallest subset of variables which adequately describe the real system is very important and highly desirable quality of a good modeller.... Simplicity is an essential criterion of a good model.

We intend to apply the above ideas to passenger behaviour.

1.1 Background

In order to have a better understanding of the underlying complexity of the airline booking operation, we will give a brief description of the main elements of the booking process.

- **Capacity**: the aircraft type and size determine its capacity, which in its turn determines the configuration of compartments. For example the 378-seats Boeing 747 is physically divided for some airlines as follows:

  * First Class cabin: 12 seats in upper deck and 20 seats in the front of the main body of the aircraft
  * Business Class cabin: 24 seats located between first class and economy cabins
  * Economy Class cabin: 322 seats located in the rear of the aircraft

  The Airbus aircraft A300 type, on the other hand has 235 seats and A310 type has 170 seats. Most aircraft types can also accommodate special seat requests such as carrycots, stretchers, wheelchairs and incubators. When a flight is created in the Reservation System of an airline, an aircraft type is assigned to it. As the day of the flight departure gets closer, the type of aircraft might be changed either to a bigger airplane to accommodate more passengers or to a smaller one if the demand is low.

- **Flight**: There are two types of flights, Single-leg (or Single-sector) flights and Multi-leg flights; a Single-leg flight is the non-stop flight between an origin and a destination. For example the flight number KU301 is a non-stop flight that originates from Kuwait City and ends at its destination in Bombay. A multi-leg flight is a flight between two cities (origin-destination) with one or more stops in intermediate cities. For example, the flight number KU101 starts from Kuwait City stops in London then ends in New York. KU101 is a two-leg flight.

  Flights also consist of segments, a segment is any multi-leg or single-leg city-pair combination on the same flight number. Figure 1.1 illustrates the difference between legs and segments.

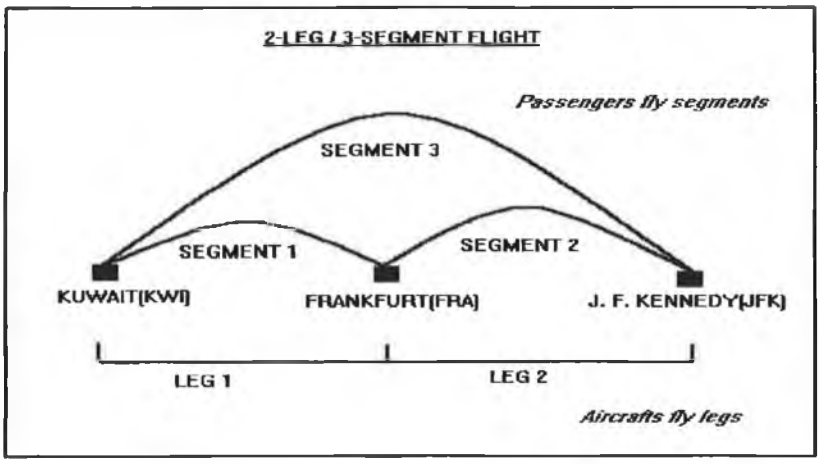


Figure 1.1: The difference between flight leg and flight segment

- **Pricing policy**: in the airline industry different customers are willing to pay different prices for the same product. A product is a flight seat on a particular market and in a particular cabin. Many customers are willing to pay a higher price for additional service features. For example, customers travelling on company business are less price-sensitive and more service-sensitive, on the other hand leisure travellers are willing to sacrifice service features for lower cost. Pilgrims tend to have a lengthy stay at their destination for religious purposes and are price-sensitive. Students prefer low-priced seats but they have limited range of days to choose their flights from due to the timing of the school holidays. There are other types of passengers such as groups, tour operators and labour groups.

  Once a flight is created in the reservation system, it will be assigned price/fare-class structures. There are many fare levels used in the airline business. The Main factors used in setting fare levels are cost, load factors, traffic volume, competitive fare levels, and capacity. Fares are usually changed twice a year after IATA (The International Air Transport Association) conferences in April and October. For example, Kuwait Airways standard IATA based price levels consist of:

  * First class full fares booked in F class
  * Business class full fares booked in J class
  * Economy class full fares booked in Y class
  * High priced excursion fares booked in Y class
  * Lower priced excursion fares booked in Y class

Other fare classes are used for discounted traffic. These discount fare classes (sub-classes) are market-specific and may be used to control point of sale. Discount sub-classes are nested in the reservations system either in parallel or serial nesting structure.

- **The Booking Process**: as best described by Lee (1990), consists of three phases:
  1. The *reservation* phase: when customers request a seat on airline flights and choose from a number of alternatives (flights, classes) that best meet their preferences. The airlines then either accept or reject their requests depending on specific decision policies.
  2. The *cancellation* phase: when passengers who had reserved seats then return later and cancel their reservations during the time before the departure of the flight. Another type of cancellation would occur automatically by the system if the passenger did not comply with the restrictions imposed on their type of booking. For example, when passengers do not purchase their tickets by certain time before departure, the system will automatically cancel their bookings.
  3. The *boarding* phase: which is the actual departure time of the flight when the final number of passengers boarded and travelled may be determined.

- **Passenger Types**: from the airline perspective, there are many types of passengers which are identified by different booking codes in the reservation system. For example,
  - The *confirmed* passengers who hold confirmed bookings and tickets.
  - The *waitlisted* passengers who know that their requested fare classes are full and agree to be waitlisted on them.
  - The *cancelled* passengers who previously reserved seats then cancelled.
  - The *go-show* passengers who arrived on the day of departure and booked seats and travelled.
  - The *no-show* passengers who hold confirmed reservations but did not show up on the departure day for some reason such as illness, difficult road conditions, or late arrival of a connecting flight.
  - The *denied-boarding* who hold confirmed reservation and tickets but were not allowed to board the aircraft either because all seats have been taken by other confirmed passengers due to overbooking, (we will explain

overbooking process in the next paragraph), or perhaps for legal or security reasons.

- **Overbooking**: it is a common practice in the airline reservation operation to allocate more seats to certain fare classes than the physical capacity, by setting overbooking profiles in the reservation system. These profiles allow the system to accept bookings (spaces) greater than the actual number of seats available on the aircraft. As we mentioned above, if airlines did not overbook their flights, and if on the departure day some passengers did not show up then aircraft would depart with empty seats causing loss in revenue. In order to minimise this revenue loss, airlines allow their flights to be overbooked, predicting that these overbooked passengers will be accommodated in the no-show passengers seats. For example, Kuwait Airways flights to Cairo are usually overbooked 80% over the capacity because of the high no-show behavior in the Kuwait/Cairo market. At the same time, airlines must be careful in setting these profiles to avoid overselling their seats and face denied-boarding passengers. These then would be entitled to some type of compensation, such as alternate transportation, a ticket voucher or hotel and meal vouchers. This situation can also cause revenue loss.

1.2 Goal of Thesis

In the past, most work in the area of passenger demand forecasting was concentrated on flight level. Analysis was performed on flight, origin and destination, fare classes and number of booking on each class. One reason for this is the limited detailed passenger data available in the reservation systems. Only recently the airlines enhanced their systems to include more information on their passengers especially with the evolution of the yield management systems which depend on good forecasting techniques in order to recommend what classes should be available and how many seats should be allocated to each class. Even with this information stored and used to predict future passenger behaviour, there were some activities which can not be detected or logged such as when

passengers opt not to book even if their most desired class, according to their preferences, is still available. Using a computer simulation for the booking process enables us to analyze and investigate any situation that may occur such as: what if passengers actually arrive to book on a day before departure but do not book on their most desired class available. Or: what if passengers booked on the second desired class because their most desired class is not available. In this scenario we are actually analyzing the hidden behaviour of the consumers that cannot be reflected in a time series analysis of flights. Also under this analysis we might be able to recognize patterns in different passenger types which will enable us to come up with reasonably identified categories that would define the different passenger types. This could be achieved by analyzing their reaction to different fare classes or what restrictions are placed on these classes.

Monte Carlo simulations can model these different reactions and behaviours, as they depend on a set of conditional probabilities and given parameters. Monte Carlo simulations have been used in the airline yield management systems for testing new techniques. In this thesis we use a particular type of Monte Carlo technique known as the Metropolis algorithm.

The goal of the thesis is to evaluate how well the Metropolis algorithm can be used to estimate models of passenger behaviour. We describe a generic model of passenger behaviour, which predicts how passengers will react to different fares and levels of service on offer. This model involves several variable parameters. Ideally we would like to be able to estimate values for these parameters from real-world data using the Metropolis algorithm. However, before that, we wish to evaluate Metropolis using simulated data.

Our method for evaluating the Metropolis algorithm is as follows:

1) We insert trial values into the model parameters.
2) We generate simulated data from the model using the trial parameters.
3) We then use the Metropolis algorithm to estimate the parameters from the simulated data
4) We evaluate how well the estimated parameters compare with the trial parameters, which were used to generate the data.

1.3 Organization of Thesis

Chapter 2 is a brief description of the Yield management system and its relationship with the airline seat allocation process. Chapter 3 will introduce the basic concept of the Monte Carlo methods and why these methods can be used to model complex systems. In the same chapter we also explain briefly how the Metropolis algorithm can be used to construct Markov Chain(s) in a Monte Carlo simulation model in order to estimate the model parameters. Following these two introductory chapters, our Monte Carlo model of the passenger booking process will be presented in Chapter 4, followed by a detailed description of how to create a Markov Chain using the Metropolis algorithm in Chapter 5. We start Chapter 5 with the building of the Probability Distribution Tables that are required in the mathematical calculations of the likelihood values used by the algorithm. Chapter 6 presents the results of our evaluation of the Metropolis algorithm. For several different cases we generate simulate data. We then use Metropolis to estimate the parameters used to generate the data. We compare the estimated values with the original values. We conclude our work in Chapter 7.

CHAPTER 2

Seat Allocation and Yield Management System

2.1 Introduction

Any business organization attempts to improve its profit while selling its products, whether the products are toothpaste, shoes, cars, hotel rooms or airlines seats. Improving the profit is achieved by reducing the cost and/or increasing the revenue. In order to increase the revenue, business organizations realized that it is essential to use some kind of Yield Management (YM) tools. Airlines were first in implementing YM techniques to enable them to selectively accept or reject bookings in such a way that would maximize their overall revenue. Prior to YM evolution, a group of reservation staff known as the space or inventory controllers were responsible for all flight inventory controls. When a flight is created in the reservation system 11 months before the day of departure, the system keeps track of its inventory which mainly includes the seats allocated in each booking class assigned for the flight and the number of seats sold on each class. Figure 2.1 shows a sample inventory record display from the reservation system. The inventory controller group set the overbooking levels on certain days over the life of the flight, see Figure 2.2. The inventory controllers have to choose from more than 200 different manually determined overbooking profiles, that are stored in the reservation system for each flight. They have to take into consideration the market that the flight serves, the season that the flight operates in and the day of the week that the flight departs on. There is no fixed set of rules for setting these profiles and the decisions are made entirely by the controllers depending on their judgement and experience of the variation in flights' behaviour.

KU 123 6Sep AAA CCC **F**				**ACP**	10	**PRO**	0
INH DC BRG	**NETAU**	**NETSA**	**AU**	**SA**	**PL**	**SNL**	**CNL**
AAA	15	2	18	4	4	0	
BBB	15	11	18	14	4	10	
		J		**ACP**	0	**PRO**	0
INH DC BRG	**NETAU**	**NETSA**	**AU**	**SA**	**PL**	**SNL**	**CNL**
AAA	21	0	24	1	4	0	
BBB	21	18	24	21	4	17	
		Y		**ACP**	15	**PRO**	22
INH DC BRG	**NETAU**	**NETSA**	**AU**	**SA**	**PL**	**SNL**	**CNL**
AAA	21	0	24	1	4	0	
BBB	21	18	24	21	4	17	
		B		**ACP**	15	**PRO**	19
INH DC BRG			**AU**	**SA**	**PL**	**SNL**	**CNL**
AAA			80	71	4	58	
BBB			80	84	4	70	
		H		**ACP**	0	**PRO**	0
INH DC BRG			**AU**	**SA**	**PL**	**SNL**	**CNL**
AAA			90	3	4	0	
BBB			90	48	4	39	
		T		**ACP**	0	**PRO**	0
INH DC BRG			**AU**	**SA**	**PL**	**SNL**	**CNL**
AAA			140	31	4	23	
BBB			140	78	4	66	

.
.

NETAU: net authorization for the prime classes
NETSA: net seats available in the prime classes
ACP: actual percentage overbooking per class
PRO: Profile Table item number which describes the overbooking percentages
AU: Total authorization allocated to each class
SA: total seats available on each class

Figure 2.1: Sample Flight inventory record for two-leg flight KU123 (AAABBB and BBBCCC) with three prime classes F, J, and Y and three subclasses, B, H and T nested within Y class

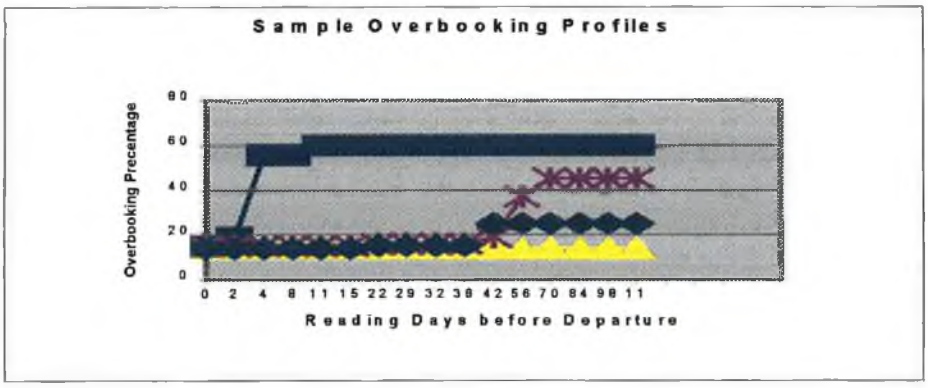


Figure 2.2: Sample Overbooking Profiles

As departure days approach, the controllers have to monitor their flights more frequently adjusting the overbooking profiles as they see best accounting for the current number of bookings, the cancellation rates and the no-shows expected on days of departure from similar past flights. The controllers aim to fill their flights and avoid denied boardings on departure. While monitoring flights the controllers also have to give special attention to the discounted classes on each flight and carefully re-adjust the number of seats allocated to each class in order to maximise the total number of bookings as well as the revenue that would be generated from the fare of each class.

With YM system many of the manually performed tasks can be done automatically such as setting and controlling overbooking profiles for most of the flights allowing the controllers to concentrate on critical flights that require special attention and manual intervention. With the availability of historical flight data that is collected by the YM system more accurate demand forecasts can be determined which in their turn lead to a better estimates for the overbooking profiles. Consistency can be achieved by allowing an automated system to review all flights in the airline network and then set inventory levels for all flights instead of different decisions by different controller. With the help of YM systems which use mathematical optimization and forecasting techniques and algorithms, as well as data management systems, the inventory controllers are

able to make the right decision in setting the seat inventory levels for their flights (Kuwait Airways Yield Management and Pricing, unpublished document 1992). We will describe in details the YM system in section 2.2. Section 2.3 is a brief overview of the literature on YM methods.

2.2 Yield/Revenue management system

2.2.1 History

Yield/Revenue Management systems have developed since the early 1980's in response to the increasing product range which airlines started to offer their customers due to deregulation of the airline industry in 1978. The Civil Aeronautics Board (CAB) was responsible for setting fares for the industry as a whole before deregulation. Since deregulation a number of low-cost carriers, such as EasyJet and Ryanair, started up and proved successful in competing with the major airlines. In order to stay in business, mainstream airlines began to offer seats that would otherwise be empty to low-fare passengers. These low fares had to have some restrictions such as, minimum stay, non-refundable tickets, so that passengers willing to pay higher fares would not be attracted to those offers too.

With the increasing product ranges, most airlines realised that it was essential to use yield management systems in order to maximise the revenue potential that could be generated from these products. American Airlines, one of the leading airlines in developing yield management tools, defined Yield management as the selling of the *right* seats to the *right* customers for the *right* prices at the *right* time. In the airline industry this concept is put into practice via the control and management of seat inventory controls in the reservations system. Inventory controls are set in a way as to maximise airline profitability, given the fare structure and flight schedule. The term *Yield Management* in this context is a wrong term because revenue and profit, not yield, are being maximised. The airline seat after all, is a perishable product because after the flight departs it has no value unless it was occupied and paid for. As a result, Weatherford and Bodily

(1992) proposed to replace the term *yield management* with *Perishable-Asset Revenue Management* (PARM). They defined it to be the *optimal revenue management of perishable asset through price segmentation*. The authors developed a comprehensive taxonomy for the underlying assumptions for general PARM models and identified 14 descriptors that can be used to categorise a range of PARM problems: yield management, overbooking and pricing.

2.2.2 YM objectives

The main objective of the revenue management system is to maximize total passenger revenues and load factors. This can be achieved by the following:

- Balancing the number of low and high fare bookings by making sure that there are always seats available for the higher revenue demand when such demand is expected to avoid selling more seats to low fare customers leading to lower revenue (revenue dilution).
- Using overbooking profiles in such a way that maintain the balance between the number of empty seats when the flight departs (spoilage) by accounting for cancellations and no-shows, and the number of over-sales that leads to denied boarding.
- Providing recommendations when dealing with group bookings.

Two types of factors influence the decisions made by the system:

1. Known factors such as the capacity, fare structure, current number of bookings held, the availability of detailed historical data that provides trends on previous passenger behaviour, and the existence of competitors in the same market.
2. Unknown factors related to the uncertainty in passenger behaviour. Passenger characteristics such as the booking characteristics, i.e. how many and what time would passengers book during the life of the flight, also their cancellation and no-show behaviour. These characteristics are affected by factors such as the season, changes in flight schedule, changes in fares, current events.

Effective revenue management increased revenues tremendously and the payoff from effective seat inventory control also proved to be substantial. For example, Delta Airlines in 1985 estimated that selling just one seat per flight at a full fare rather than a discounted fare can add over $50 million to its annual revenue (see Belobaba 1987b). American Airlines and United Airlines claimed that their annual increase of profit was more than 100 million dollars from their revenue management systems (see YM3). Belobaba and Wilson (1997) using a simulation model showed that the use of a yield management system not only has a positive impact on the airline using it by increasing its revenue, but a negative one on other airlines with no yield management tools that operate in the same competitive market.

2.2.3 YM Components

Figure 2.3 illustrates the relation between the YM system and the Computerized Reservation System (CRS), where the flight inventory control resides (Yield Management Workshop 1996).

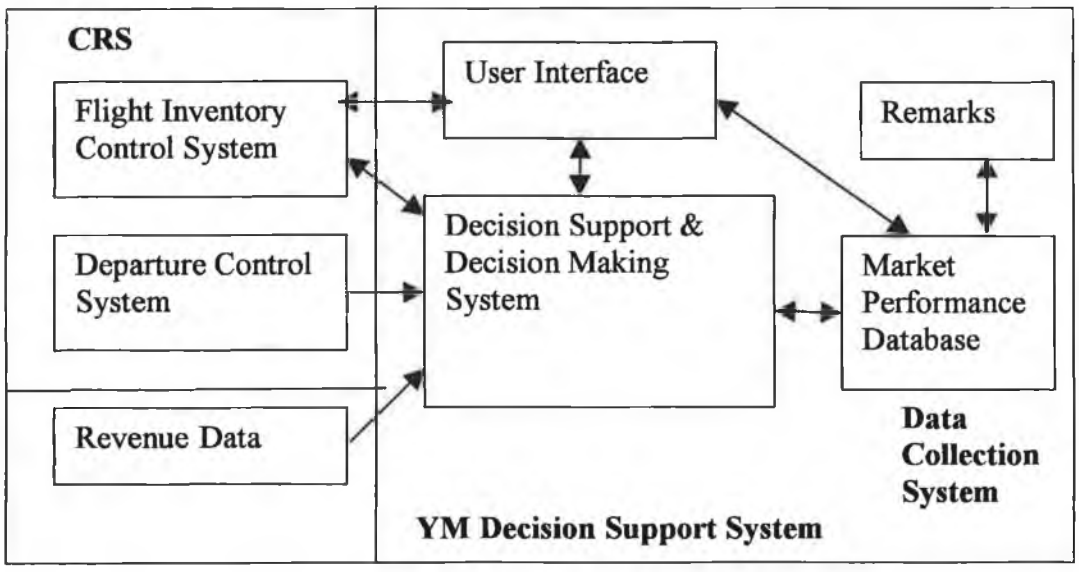


Figure 2.3: YM relation with the Reservation System

The main two modules of any revenue management system are as follows:

1. The **forecasting** module utilizes passengers and operational information that are collected from the reservations, inventory and departure control systems in order to predict passengers behaviour such as demand, no-show rates and cancellation percentages on future flights.
2. The **optimization** module uses the output from the forecasting process with revenue data to determine the effective booking limits and overbooking profiles. It also identifies and highlights critical flights that require special attention from the yield analysts.

Figure 2.4 shows the components of the YM system.

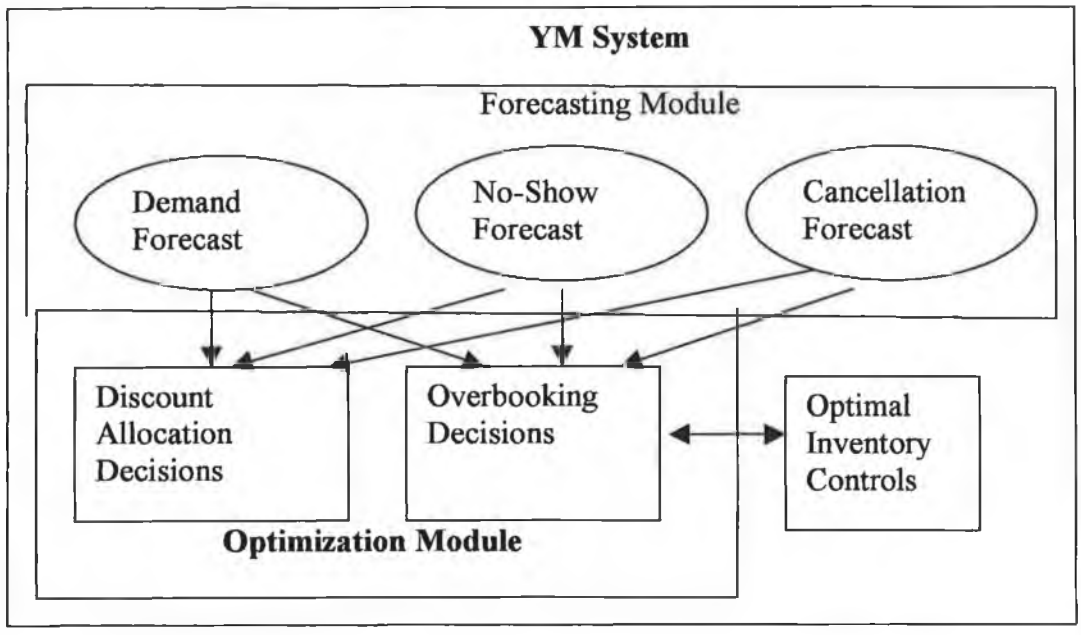


Figure 2.4: YM components

As a result of their success in the airline industry, revenue management systems are currently used in a wide range of businesses such as hotels, car renting, railways and media broadcasting. For more information on revenue management and some existing systems see the following Internet references: YM[1], YM[2], YM[3], YM[4], YM[5], YM[6].

2.3 Previous approaches: A Literature Review

Major airlines, either individually or through operation research groups, developed a variety of yield management methods that best suit their market environment and work well with their reservation systems. Belobaba (1987a, 1989), one of the pioneering researchers in the area of airlines seat allocation management for multiple nested classes, developed a probabilistic decision model based on the Expected Marginal Seat Revenue (EMSR). The EMSR model (and the modified version EMSRb (1992)) is used to set and revise booking limits on each fare class on single-leg flights taking into consideration the stochastic (i.e. *random*) nature of the future demands. By calculating the booking limits on each class the model aims to protect seats for the higher fare class from the next lower fare class. The user decides the points in time (days before departure) during the booking process that the model should run to update its recommendations. These EMSR models are widely used in yield management systems.

Many comparisons were made by researchers and analysts, including Belobaba (1992), between the EMSR algorithm and other optimisation techniques. For example, Brumelle and McGill (1993), Curry (1990), Robinson (1995) and Wollmer (1992), all reported that the EMSR model was only able to calculate the optimal booking limits for the first two highest fare classes in the nesting structure, they either modified the original EMSR or developed their own algorithms to obtain the optimal seat allocation for all nested fare classes.

The seat allocation problem became more complex when airlines started to increase their scheduled flights to many different destinations. They were not interested anymore in maximising the revenue for each flight separately but rather for the whole network, by managing the seat allocation policy on the basis of origin and destination. Figure 2.5 illustrates a sample of a flight network of some Kuwait Airways flights. Seats allocated on different fare classes on KU102 flight (for the North America and Europe market) is directly affected by the KU201 flight to Karachi and KU301 to Bombay (for the Indian Sub-continent market). In order to maximise the overall revenue for the network a balance

between high fare seats and low fare seats has to be analysed carefully to make sure all flights have a minimum number of empty seats.

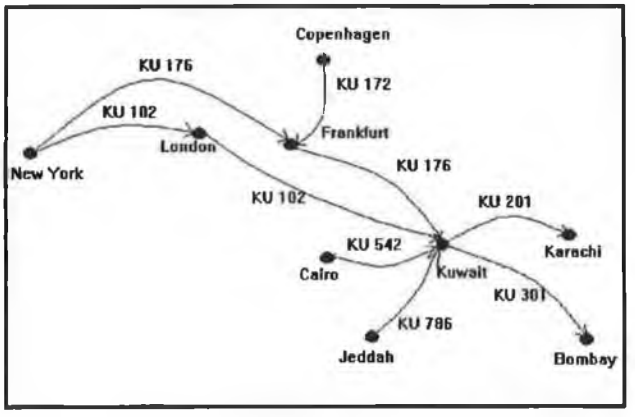


Figure 2.5: Flight network

As a result of this, more work has been done by researchers to extend the control from single-leg techniques to a wider origin-destination optimisation methods. See for example Belobaba (1989), Curry (1990), Dror et al. (1988), Gallego and van Ryzin (1997), Glover et al (1982), Smith and Penn (1988), Vinod (1995), Williamson (1988), and a number of papers appeared in the proceedings of Airline Group of the International Federation of Operational Research Societies (AGIFORS 1996/1997).

In the area of forecasting, Lee (1990) studied the airline reservations forecasting process in detail and developed a probabilistic model that describes the booking process as a stochastic process with requests, reservations, and cancellations interspersed in the time before departure of the flight. He modelled the booking process as an immigration and death process with non-homogeneous requests and cancellation rates. A request or a reservation is considered as a new immigrant to the population of the booked passengers, and a cancellation is considered as a death of an existing member of the population. He introduced a censored Poisson model for modelling the booking process which captures the dynamic nature of

the process and takes into account the censoring of airline booking data from above at the booking limit.

Neural Networks have been used to forecast the no-show rates, Blackley (1993), McGrath (1995), by using attributes from the passenger records held in the reservation system (the historical passenger data stored in databases for statistical analysis varies between airlines) such as: seat request, sales area, type of payments. These studies proved that using passenger data rather than flight data in the forecasting process improves its accuracy. McGrath (1995) reported that using Radial-basis Functions Networks improves show and no-show predictions.

Only in the last few years researchers realised that they have to put more thought into modelling the passenger choice behavior in more realistic way. That means if there is more than one class available at any point in time during the booking process, passengers might either choose their most desired class (according to their preferences: price, service, times, booking restrictions....) or not book at all. When the most desired class is full (closed) then passengers might choose between booking on the second desired class (vertical recapture or buy up or buy down) or book on another flight (horizontal recapture) or not to book at all and so on.

Passenger Origin-Destination Simulator (PODS), which is the passenger purchase behaviour simulation developed by Hoperstad at Boeing, AGIFORS 1996/1997, demonstrated the importance of passenger choice behaviour on the revenue management system recommendations. In a very comprehensive survey of 40 years of research in the area of forecasting, overbooking, seat inventory control and pricing, provided by McGill and van Ryzin (1999), the authors highlighted the importance of further research in the area of the behaviour of different passenger types towards the change in fare products that are on offer.

Anderson (1998) suggested three approaches to improve the Scandinavian Airlines System (SAS) origin-to-destination seat allocation system in order to incorporate the buy-up (vertical recapture) demand and the horizontal recapture (on other SAS flight) demand. He developed a discrete choice model that calculates the probability of buying up when the most desired class is not

available, the probability of selecting another flight on the same class and the probability of loosing the passenger to another carrier or through cancellations. The model was based on the choice theory and *multinomial logit function* (MNL), which is also the method used in our model as we will explain in Chapter 4. Bitran et al. (1998) developed a general dynamic pricing stochastic model which aims to maximise the revenue for retail chain with multiple stores and variety of perishable products. Although the model had only one choice parameter, price, it successfully modelled the arrival rate of customers and their willingness to pay for products.

Talluri and van Ryzin (2000) developed a discrete choice *dynamic program* (DP) algorithm that captures the consumer choice behaviour by also using the MNL, while deriving the optimal policies for the seat allocation problem. The authors in their paper say: "*...while many attempts have been made to understand the impact of choice behavior on traditional yield management methods and to develop simple heuristics that partially capture buy-up and buy-down behavior, to date there is no methodology that directly and completely addresses the problem.*"

They combined the probability of arrival (or no arrival) with the probability of choosing a fare class according to its utility value to different passenger types. Their choice-DP model assumed a fixed arrival rate for all time periods of the booking process. In order to run the optimization algorithm, estimations for the arrival rates and the choice parameters were obtained by applying the *expectation maximization* (EM) method of Dempster (Dempster et al 1977). There are some similarities between our work and the work conducted in this paper.

CHAPTER 3

Markov Chain Monte Carlo Method

3.1 Introduction

Monte Carlo (MC) methods are stochastic techniques based on the use of random numbers and probability to investigate problems. Any technique that utilizes sequences of random numbers to perform statistical simulation could be called a Monte Carlo method. The name Monte Carlo came from the capital city of Monaco which is known as a center for gambling. Although the basic techniques have been used for centuries, only in the 1940's Metropolis, von Neumann and Ulam called these methods Monte Carlo methods while working on the Manhattan project because of the similarity of statistical simulations to the game of chance, the core basis of gambling.

MC methods have been used in a wide range of fields such as nuclear physics, molecular computation, traffic control, weather forecasting and others. Some of the Seat Allocation Forecasting and Optimization techniques, mentioned in the previous chapter, were tested by MC simulations (Lee 1990). Another application area is medicine and medical related fields. A discrete MC method was used in modelling the interaction between the HIV virus and the immune system (Mannion 2001). This work illustrates that MC algorithms are suitable when the field of application heavily depends on randomness and the behaviour of this type of problem can be modelled by probability distributions. Probabilistic parameters were used for the mutation of the virus and for the mobility of the 4 types of cells used in the model.

There are many reasons behind the wide application of MC methods especially in the last several decades:

(1) Its ability to examine complex systems by random sampling from the probability density functions (pdf's) that describe the behaviour of the physical (or mathematical) system

(2) The availability of high speed supercomputers allows research into problems that may otherwise be computationally intractable

(3) The availability of efficient pseudo-random number generators which MC simulation requires

(4) The method can be used in a parallel environment, where the simulation can run on more than one processor simultaneously with different random number sequences, then all outputs can be combined to get a final result

The major two components of any MC method are the probability distribution (see Spiegel et al, 2000) and random number generation. The use of probability theory in the field of statistics helped statisticians and scientists in many fields to draw valid conclusions and make reasonable decisions on the basis of analysis of data. There are many different interpretations of the concept of "probability". In this thesis we assume that probability measures the uncertainty as to whether a particular event will occur or not in any random experiment. It takes a value between 0 and 1 representing the strength of the belief in the occurrence of an event. If we are certain that the event will occur, then the probability of this event is 1. On the other hand, if we are certain that the event will never occur, then its probability is 0. For airline passengers, this can be applied by assuming that business travellers have probability 0 of arriving 6 months before departure date of a specific flight to book seats. Whereas tourist passengers might start arriving 6 months (or even earlier) before the departure in order to book seats in the low fare classes while available so their probability of arrival has a value greater than 0. We will present the different probability distributions of different types of passenger arrival in more detail in chapter 4 and 5.

Most MC models are based on Bayesian probabilities, which we will explain in section 3.2. The Markov Chain Monte Carlo (MCMC) method (see Gilks et al 1996) is a Monte Carlo technique using Markov chains. A Markov chain is the process of generating a sequence of random states, say $\{X_0, X_1, X_2....\}$ in such a way that after time t, the next state X_{t+1} does not depend on the history of the

chain $\{X_0, X_1, ..., X_{t-1}\}$ and only depends on the current state X_t. The transition from one state to the next is decided by the conditional probability $P(X_{t+1} \mid X_t)$ which determines whether to accept this random variable X_{t+1} or to generate another one.

We mentioned above that the MC methods are stochastic, so in order to ensure this property, we use a random number generator to generate uniformly distributed random numbers between 0 and 1. We use the built-in function rand() in MS Visual C++ to generate sequences of random numbers.

3.2 Bayesian Probability Theory

Most Monte Carlo applications are based on Bayesian Theory (see Spiegel et al, 2000). This probability theory is subjective and depends on our state of knowledge. We mentioned earlier that probability is a measure of uncertainty as to whether a particular event will or will not occur. If we have no prior knowledge regarding the event this will affect our measure of probability. However, if we receive more information or if we actually know something about this specific event or its properties then this will either increase or decrease our belief in it taking place. For example, if a passenger wants to book a seat on a flight and has a choice between two available classes with different fares, you would assign an equal probability (0.5) to each class. However, if you knew that this passenger is going on a business trip and the ticket will be paid by his company, then the probability of booking in the high fare class will increase and the other probability will decrease. In other words, we can say that the probability of passengers booking in high fare class (A) given they are business men (B) is higher than the probability of any passenger booking in high fare class. This is known as the conditional probability of A given B denoted by *P(A|B),* and it is the basic concept of Bayesian Theory.

Bayesian Theory has the following two axioms:

1. $P(A)+P(\overline{A})=1$ where $\overline{A}$ is "not A"

2. $P(A \wedge B)=P(A \mid B)P(B)$ where $A \wedge B$ means "A and B"

We know that

$$P(A \wedge B)=P(B \wedge A)$$

Then

$$P(A \mid B)P(B)=P(B \mid A)P(A)$$

Or

$$P(A \mid B)=\frac{P(B \mid A)P(A)}{P(B)}$$

which is the Bayes Theorem.

Bayes Theorem can be used to assess the probability of a hypothesis, H, being true given some data D and can be represented as follows:

$$P(H \mid D)=\frac{P(D \mid H)P(H)}{P(D)}$$

$P(H \mid D)$ is the probability of the hypothesis after receiving more information represented by some data and known as the "***posterior***"

$P(D \mid H)$ is the probability of the data given the hypothesis and known as the "***likelihood***"

$P(H)$ is the probability of the hypothesis before receiving any information and known as the "***prior***"

$P(D)$ is the probability of the observed data

Notice that our belief in the hypothesis increases or decreases as we observe more data. So if we have more than one hypothesis that could explain the data, we can use Bayes Theorem to assess which hypothesis is better in explaining the given data. In other word, we need to find the posteriors for each hypothesis:

$$P(H_1 \mid D) = \frac{P(D \mid H_1)P(H_1)}{P(D)}$$

$$P(H_2 \mid D) = \frac{P(D \mid H_2)P(H_2)}{P(D)}$$

Each hypothesis represents a "model". Notice that $P(D)$ is the same in both equations and can be ignored, as it is a constant of proportionality. If we have more than two hypotheses and each one is defined by some parameters denoted by the vector θ_i, then the Bayes Theorem can be written as:

$$P(\theta_i \mid D) \propto P(D \mid \theta_i)P(\theta_i)$$

This says that the posterior probability of the hypothesis i (defined by its parameters θ_i) given some data D is proportional to the likelihood multiplied by the prior.

When $P(\theta_i)$ is the same for all hypotheses i.e. uniformly distributed over θ then

$$P(\theta_i \mid D) \propto P(D \mid \theta_i)$$

This means that the posterior probabilities are proportional to the likelihood.

Bayesian Theory is the basis of most algorithms used in MC simulations for its ability to improve the measure of the probability towards the most probable level when used in many trials. It can be used to classify new data. For example, it is widely applied in the field of pattern recognition. One of the main areas of pattern recognition is image analysis and image classification. Ripley and Sutherland (1990), applied the theory to images of spiral galaxies in order to classify (any) new galaxy image, and Ripley (1990) also applied the same theory to nematode recognition. The basic idea of both those applications is to first assign a prior distribution over a class of possible models P(S), using spatial stochastic

processes described by Markov random fields. Then for an observed new image Z we calculate the probability distribution P(Z|S), the likelihood. Then any analysis on S can be based on the posterior distribution P(S|Z) over the whole class of possible models. This posterior is given by P(S|Z) α P(Z|S) P(S). (for other MCMC applications in image analysis, see Chapter 21 in Gilks et al 1996)
We could use real data to fit our model. The data consists of booking curves for large number of flights. But initially we will use simulated data. Our technique is based on the Bayesian approach. We have a model of passenger behaviour, which involves several parameters. In Chapter 4 we generate sample booking data from the model, based on certain input parameters that describe the behaviour of different passenger types. Then we can evaluate the posterior probability of the model parameters given the sample data, by formulating an algorithm to calculate the likelihood of the sample data given random combinations of parameter values, using the appropriate distribution functions that describe the problem. We will explain how this process can be performed in Chapter 5.

3.3 Metropolis Algorithm

When the model that describes the problem under analysis is represented by a large number of parameters it becomes difficult to train it to classify new data. It would be too difficult to build the posterior distribution $P(\theta \mid D)$ over the parameters. In order to do that we would need to try all different combinations of possible parameter values and calculate the probability of each combination, and this is too complicated and time consuming. The solution is to generate random samples from the posterior distribution. If the sample is large enough then it can be used as an approximation to the posterior distribution. The sample consists of a large number of parameter vectors θ. The frequency with which a particular value of θ occurs in the sample is proportional to its posterior probability. Thus, the most probable values of θ will occur most often and the least probable values will be the most rare.

This is basically the Markov Chain Monte Carlo (MCMC) method which generates samples of the parameters from a distribution that is similar to the posterior distribution by running Markov chain(s) for a long time. Starting with a random combination of parameters, the next combination is generated from the current one. After the algorithm has run for a long time, the complete set of parameter combinations represents the posterior distribution. The main advantage of constructing the Markov chain is that the next combination of parameters only depends on the current combination of parameters and is not affected by the starting combination, i.e. the history of the chain. One way to construct a Markov chain is by using the Metropolis Algorithm. If we have D as the data and θ are the parameters of the model, from Bayesian theory, the posterior probability $P(\theta \mid D)$ is proportional to the likelihood $P(D \mid \theta)$ multiplied by $P(\theta)$, which can be calculated from the model. We omitted $P(\theta)$ because we assume the prior of all the combinations of parameters are equally likely, so $P(\theta)$ is constant, (refer to section 3.2.)

The Metropolis algorithm is presented in the following steps:

1. Choose an arbitrary starting combination of parameters θ_1
2. Calculate $\ln(P(D \mid \theta_1)$
3. Change one parameter at random by adding to it a random number to get a new combination of parameters θ_2
4. Calculate $\ln(P(D \mid \theta_2)$
5. If $\ln(P(D \mid \theta_2) > \ln(P(D \mid \theta_1)$ then add θ_2 to the accepted combinations and replace θ_1 with θ_2
6. If not, then if $\exp(\ln(P(D \mid \theta_2) - \ln(P(D \mid \theta_1)) \geq$ uniformly distributed random number in the interval [0-1], then add θ_2 to the accepted combinations and replace θ_1 with θ_2
7. Repeat from step 3

Step 6 allows the algorithm to generate combinations that are less probable than the current one. If this step were not present the algorithm would simply climb up to the local maximum nearest to the starting combination and remain there. Step 6 allows the algorithm to move "downhill" and hence explore the whole parameter space. The length of time that it spends in a particular region of the parameter space is proportional to the posterior probability of the combinations in that region.

After a certain number of iterations the Metropolis algorithm should reach an "equilibrium" state, and after that the accepted list of combinations θ_i should represent the posterior distribution $P(\theta \mid D)$.

Our goal in this thesis is to use the Metropolis algorithm to estimate the parameters of a model describing passenger behaviour in the airline industry.

CHAPTER 4

Booking Process Simulation Model

4.1 Introduction

In this chapter we build a model that simulates the passenger choice behaviour in booking an airline seat. As we mentioned in chapter 1, there are different types of travellers, for example, there are business travellers who are more interested in comfortable seats on a flight (especially in a long trip) than how much the seat costs. On the other hand, there are tourist travellers who are more sensitive toward the price they are paying for their seats than what restrictions that are placed on their type of fares.

The airlines are faced with the problem of how to divide the aircraft into different (physical or logical) sections with a different fare associated with each section in order to get the best revenue out of their flights. If they assign more seats to the service-sensitive travellers and fewer seats to the price-sensitive ones, they might not get enough demand to fill the aircraft (if there are more requests from, say, tourist or student travellers.) On the other hand if there are more seats assigned to the cheaper class and the demand is high enough to fill the flight, but there is also unnoticed demand for the more (slightly) expensive class that would have filled the flight with a reasonable increase in the overall revenue.

There are many factors that influence traveller choice behaviour such as:

* the price they are paying for the seat
* the services attached with the price
* the time of departure and arrival and also the actual departure date

Another factor affecting the traveller decision during the booking process is whether this booking is part of an itinerary of other flights that the passenger needs to connect with. For example, for students studying abroad away from

home, there are certain periods that they can travel, such as Christmas or summer holidays. So the airlines must take into consideration the high demand of students during these periods. At the same time students are price sensitive passengers which means that the demand for the low fare class would be high during those periods. If students have to connect with another flight in order to get to their destinations on a certain day and time, this would have a great effect on the decision of which flight to choose.

The booking process is the most essential part of every airline's reservation system. It takes place in real-time during the life of every flight. It starts when the flight becomes available for booking to the airline's sales staff, travel agents or any one that can access the reservation system such as over the Internet. In order to simulate the behaviour of this large and complex operation and in order to capture the various behaviour of different passenger types (arrival and acceptance to book), we built a Monte Carlo simulation model.

4.2 Airline passengers as customers

In order to understand and analyse the airline passenger behaviour we need to refer to the marketing research in the area of consumer behaviour. Consumer behaviour is the process of choosing a product or a service among a set of alternatives. Ajzen and Fishbein (1980) in their book "*Understanding Attitudes and Predicting Social Behavior*" wrote:

> *The mounting interest in consumer behavior can be attributed in part to the desire of business firms to obtain a competitive advantage by basing their marketing decisions on information about the factors that determine the consumers' preferences among products. At the same time, consumers have organized to express their dissatisfaction and demand political action to ensure, among other things, higher standards of quality and safety, lower prices, and better services.*

> *Needs, motives, or desires are assumed to influence the information a person seeks about a product, as well as her attention to, and perception of, the product's attributes....The product attributes or functions are assumed to be judged in relation to the person's needs by means of certain evaluative criteria, and this process presumably results in the formation of an attitude which ultimately influences intention and purchase behavior.*

In the airline business, it is known that the most important criteria that influence the decision of a traveller while purchasing an airline ticket on a specific class on a specific flight are the price/fare and the services/comfort associated with this price. There are other factors that affect the decision-making process such as the time of the flight (departure and arrival time), and the day of departure (weekday or weekend). For example, business passengers tend to prefer an early flight departure on Monday morning, say at 7 a.m. to attend a meeting, say at 10 a.m., instead of arriving late the night before, and weekend days might have more demand from leisure passengers.

Consumer choice analysis is a well studied area in the field of marketing, see for example Wright (1975; Ajzen and Fishbein (1980); Grether and Wilde (1984); and Engel et al (1986). According to Engel et al (1986) the Fishbein Model is the most well known multi-attribute model used by marketers in consumer choice analysis. The Fishbein Model is based on the following statement as mentioned in Ajzen and Fishbein (1980):

> *... a person's attitude toward an object is a function of his salient beliefs that the object has certain attributes and his evaluations of these attributes. In the context of consumer behaviour the object is typically a product or a brand within a product class. An estimate of attitude toward a product or brand is obtained by multiplying, for each attribute, belief strength and attribute evaluation and then summing these products across all salient attributes.*

Symbolically the attitude can be related to the consequence and belief through the following expression, as shown in Engel et al (1986):

$$A_o = \sum_{i=1}^{n} b_i e_i$$

where:

A_o = attitude toward the object

b_i = strength of the belief that the object has attribute i

e_i = the evaluation of attribute i

n = the number of salient attributes.

And as defined in Engel et al (1986):

> *The model therefore proposes that attitude toward a given object (e.g. brand) is based on the summed set of beliefs about the object's attributes weighted by the evaluation of these attributes.*

In order to calculate the attitude or the desirability of a passenger toward a fare class using the above equation, we need first to define each fare class by its attributes. These attributes will only consist of the cost and the comfort associated with each fare class. We will also have a set of attributes defining the characteristics of the decision-maker, i.e. the passenger. We will explain in more detail how we apply the above equation to obtain the desirability value of each fare class by different passengers later in this chapter.

In the rest of this thesis we always use the word "class" to mean a "fare class".

4.3 The Model

4.3.1 Model assumptions

In order to build a simplified basic model, we assume the following (at the moment these assumptions are not realistic, they are merely to test the technique):

(1) The model represents the booking process on a single flight, but later we use 100 flights

(2) The booking process on this flight starts 6 months (day180) before the departure day (day 0) i.e. booking period is 181 days. Each day of the booking process is divided into 10 intervals. In each interval we assume either no arrival of any passenger type or at most one arrival of each type, i.e. we would have maximum of 10 possible arrivals of each passenger type in each day. The value of maximum 10 passenger was chosen arbitrarily for this experiment. An actual average number of arrival could be determined by conducting some sort of a survey.

(3) The flight consists of three classes, a high fare class, a medium fare class and a low fare class. The three classes are defined by the following attributes: number of seats assigned to each class; the cost (fare) of each class; and the comfort value associated with the cost on each class. See Table 4.1

(4) There are three types of travellers, a business, a tourist and a student. The characteristics of each type are defined by a set of parameters. There are two sets of parameters for each passenger type used in this model, the first one represents the arrival parameters of the passenger which are used to calculate the probability of arrival of a passenger at any given day during the booking process. The other set of parameters is used to calculate the desirability (utility) of each class for each type of passengers. These desirability values will be used to calculate the probability of accepting a booking by the passenger. At the moment we assume all members of a given type are identical. All business travellers will have the same value

of parameters, the same applies to the other types of passengers. See table 4.2

(5) For each type of passengers, the following applies:

* If the most desired class is full and the passenger is willing to accept a booking on the next desired class, this is known as "Vertical recapture" of the passenger on the same flight.

* If the most desired class is available and the passengers are not willing to book on this class, (for example, even with the best combination of price and comfort, the passengers might not think this is good enough for them in order to accept the booking) then either (a) the passengers are willing to book on another flight on the same desired class in which case they are considered as "Horizontal recaptures", or (b) they are not willing to book on this airline and are considered as "lost" passengers. (At the moment we treat these two situations together as we are dealing with one flight only)

* If all classes are full and there are still passengers arriving then these passengers are either lost or willing to book on another flight. We also treat them together as Horizontal recapture or lost due to full flight.

4.3.2 Input data

There are two categories of input data:

1. Fare class information:

    Number of seats

    Price or fare of the class

    Comfort value associated with the price

2. Passenger information:

    Arrival parameter

    Belief or desirability parameters

Table 4.1, 4.2 provide sample values for all the parameters used in this model with business passenger denoted by (bus) tourist by (tour) and student by (std) explanation and detail will be provided shortly.

	High fare class	Medium fare class	Low fare class
$seats_c$ = no. of seats	20	100	150
$\cos t_c$ = class fare	1200	800	500
$comf_c$ = class service	800	600	300

Table 4.1 A sample of class parameters

	bus	tour	std
Arrival parameters:			
a_t = first day of arrival	-30	-180	-60
b_t = last day of arrival (departure day)	0	0	0
c_t = a point between a and b which represents the first day of definite arrival until the day of departure	-7	-30	-14
s_t = scaling value	15	90	30
Belief parameters:			
p_{t1} = comfort importance rating	0.9	0.1	0.4
p_{t2} = cost importance rating	-0.1	-0.9	-0.3
p_{t3} = mean value	-400	-400	-400

Table 4.2 A sample of passenger parameters

4.3.3 Calculation of the Desirability/Utility value of classes

The passenger behaviour model is a discrete choice model where we have a discrete finite set of choices or alternatives to choose from, for example, if we have one flight then the set of alternatives will be the different classes available on this flight. If we have more than one flight with the same origin and destination on the same day, then the set of choices will include the classes of each flight as well as the time attributes of each flight. Each alternative in the choice set is characterised by a set of attributes, which are likely to affect the choice of the individual, i.e. the traveller, such as the fare of a class and the comfort or the services associated with each class. Similarly, the decision-makers/travellers have their attributes that define their characteristics and these differ from one passenger type to the other. These attributes will be used to calculate the desirability or the utility of each class in the choice set.

We choose the fare of the class $cost_c$ and a comfort value associated with each class $comf_c$ to be the class attributes. For the comfort attribute we assume a high level of services associated with the high fare class, i.e. as the price increases the level of comfort increases. (refer to Table 4.1).

The attributes defining the traveller type t (for t = business, tourist, student) consist of the following parameters (refer to Table 4.2):

The first parameter p_{t1}, defines the importance rating (belief) of the services (comfort) associated with the fare class. As we explained earlier this varies between different types of passenger and we decided on a scale between [0, 1], where 0 indicates that comfort is absolutely not important for this individual where as 1 indicates that comfort is a very important factor.

The second parameter p_{t2} defines the importance rating of the price of the class and we chose a scale between [-1, 0]. The value -1 indicates that the passenger is very sensitive to the price, which causes a negative effect on the desirability, where as the value 0 indicates that the passenger is absolutely not worried about the price of the class.

The third parameter of the passenger belief parameters, p_{t3} is required to calculate the probability of the passenger accepting to book on the desired class. We will explain how this parameter is used shortly.

Applying the Fishbein equation (refer to section 4.2) we get an overall value of desirability on each class for each type of passenger as follows:

Denote the desirability by *desr*, then for each class c and passenger type t with parameters p_{t1} (comfort evaluation) and p_{t2} (fare evaluation),

$$desr_{t,c} = comf_c * p_{t1} + cost_c * p_{t2} \qquad (4.1)$$

However, Bierlaire (1997), wrote:

> *The concept of utility associated with the alternatives plays an important role in the context of discrete choice model....the complexity of human behavior suggests that the choice model should explicitly capture some level of uncertainty.*

Also, Ben-Akiva and Lerman (1985) stated:

> *The development of Probabilistic Choice Theories arose from the need to explain experimental observations of inconsistent and non-transitive preferences. In choice experiments individuals have been observed not to select the same alternative in repetitions of the same choice situations. Moreover, by changing choice sets, violations of the transitive preferences assumption are also observed. A probabilistic choice mechanism was introduced to explain these behavioral inconsistencies....it can be used to capture the effects of unobserved variations among decision makers and unobserved attributes of the alternatives.*

Both references suggested the use of Random Utility Model to capture the uncertainty, where the utility is modelled as a random variable and has the following formula:

$$U_a^i = V_a^i + \xi_a^i$$

where:

U_a^i is the utility that individual i is associated with alternative a

V_a^i is the deterministic component of the utility

ξ_a^i is the stochastic or random component

and, the choice probability of any alternative a being selected by person i from the choice set C_i is equal to the probability that the utility of alternative a, U_a^i, is greater than or equal to the utilities of all other alternatives in the choice set. This can be written as follows:

$$P(a|C_i) = \Pr(U_a^i \geq U_b^i, \forall b \in C_i)$$

At the moment we have not yet implemented the above equation into our system. Currently classes are sorted in descending order of desirability and we assume that there is no variation between members of the same type, e.g. all business passengers have identical parameters. We further assume that any passenger will either opt for the most desired class available or accept no class at all. As Ben-Akiva and Lerman (1985) make clear in the above quotation, these assumptions are not realistic because human beings are inconsistent in their behaviour. We would try to resolve these issues in future work (refer to Chapter 7.)

After sorting the desirability in descending order, then if there are seats available in the most desired class we calculate the probability of this passenger accepting to book on this class. If there are no seats available in the most desired class, we check if there are seats in the next desired class and so on. So passengers never

book other classes if the most desired class is available. This is a consequence of assuming no variation in passengers.

4.3.4 Calculation of the probability of accepting to book on the desired class

We choose the Logistic or Logit (distribution) function to calculate the probability of accepting the desired class. The Logistic function has an S-shape curve (see Figure 4.1) centred at zero, and is continuous of the range 0 to 1. It has the following formula:

$$F(X) = \frac{1}{1+e^{-X}}$$

When X is large and negative the function is close to zero. As X approaches zero, the function is at 0.5. As X tends to infinity, the function gets close to 1.
As we mentioned above, the function is centred at X=0, so in order to shift it along the x axis we can subtract a value representing the mean from X. And we can also stretch the function to give it different S-shape curves (shallow or steep), by dividing X by a value, as follows:

$$F(X) = \frac{1}{1+e^{-(X-m)/z}}$$

Figure 4.1 shows different curves all with m (mean) = 400 to shift from zero and different z (stretch) values to illustrate how the shape of the function changes with different values of z.

We apply this function to calculate the probability of acceptance p_acc, where X is the calculated total desirability for each class from the previous step, $desr_{c,t}$, the mean m is the third parameter p_{t3}:

$$p\_acc_{t,c}(desr_{t,c}) = \frac{1}{1+e^{-(desr_{t,c}-p_{t3})/z}} \tag{4.2}$$

We absorb z into p_{t1} p_{t2} and p_{t3} since it is a linear scaling parameter.

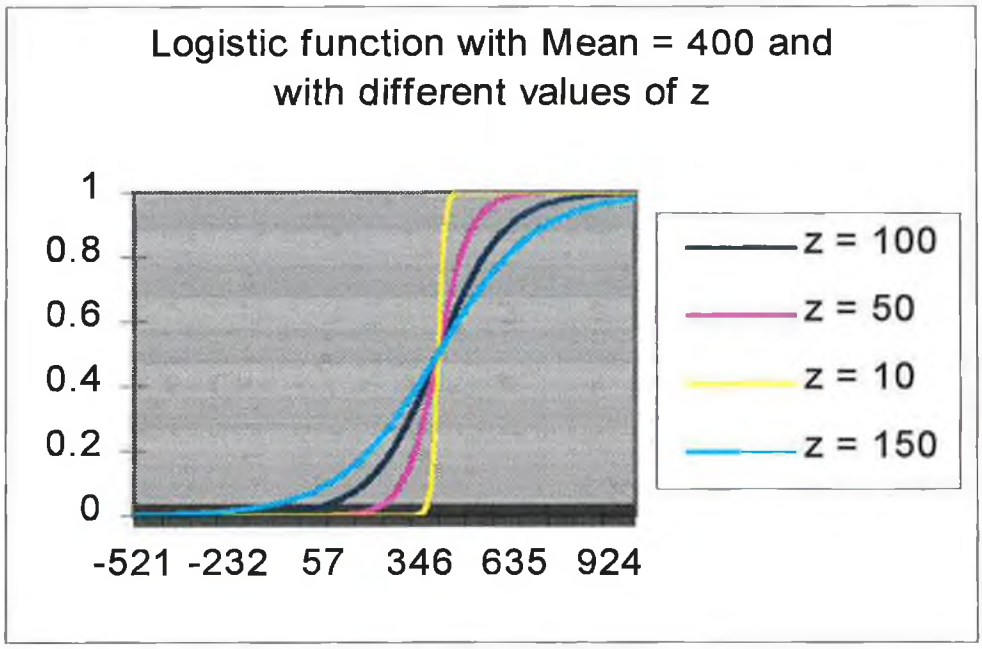


Figure 4.1: Logistic Function

Sample values of calculated desirability and probability of acceptance from one run of the model are shown in Table 4.3. We decided to use this function to calculate the probability of accepting the desired class because it gives a natural explanation of the effect of each class attribute. When cost is equal to 0, i.e. a free ticket, this would be very desirable (very close to 1). As cost increases, desirability would at first remain high but eventually will get close to 0 when cost is too high, i.e. cost has a negative effect on desirability. The function would have

an opposite effect on comfort. When comfort is completely absent, the class would be very undesirable (close to 0). As comfort increases, the desirability would increase until it gets close to 1.

We then compare the result from this function with a uniformly distributed random number between 0 and 1. If the result is greater than or equal to the random number then we assume that the passenger is willing to accept the desired class then a seat will be booked for the passenger.

type 1:		
class: 0 classlabel: 0	desr: 1.8	p_acc: 0.858149
class: 1 classlabel: 1	desr: 0.00199997	p_acc: 0.5005
class: 2 classlabel: 2	desr: 2.98023e-008	p_acc: 0.5
type 2:		
class: 0 classlabel: 2	desr: 2.35	p_acc: 0.912934
class: 1 classlabel: 1	desr: 1.172	p_acc: 0.763506
class: 2 classlabel: 0	desr: -0.2	p_acc: 0.450166
type 3:		
class: 0 classlabel: 0	desr: 3.6	p_acc: 0.973403
class: 1 classlabel: 1	desr: 1.23	p_acc: 0.773819
class: 2 classlabel: 2	desr: 0.55	p_acc: 0.634136

Table 4.3: Sample calculated desirability and probability of acceptance for three passenger types and three classes

4.3.5 Calculation of the probability of passenger arrival

The function we used to model the probability of arrival of a passenger in each interval of the 10 time intervals in the day for each passenger type, p_arr_t, is a simple triangular function of time (see Figure 4.2, 4.3, 4.4). This is not realistic, and just for the sake of the experiment only:

$$p\_arr_t(d \le a_t) = 0$$

$$p\_arr_t(a_t \le d \le c_t) = \frac{2(d - a_t)}{(b_t - a_t)(c_t - a_t)} \qquad (4.3)$$

$$p\_arr_t(c_t < d \le b_t) = 1$$

where

d is the current time

a_t is first day on which passenger type t starts to arrive

b_t is the end of the booking process (i.e. departure day). b_t is always 0

c_t is a point between a_t, b_t on which the rate of arrival levels out

We assume that between time c_t and time of departure, b_t, a passenger will certainly arrive to book a seat every day as we get closer to the day of departure for any type of passengers. Notice that p_arr is assumed to be the same for all the 10 time intervals of the same day d.

When $a_t \le d \le c_t$ and in order to get a number between 0 and 1 we multiply the result of this function by a scaling value, s_t. These variables have different values for each type of passenger t, as shown in Table 4.1. We will demonstrate the application of the function using a tourist passenger. We assume that a tourist passenger tends to arrive early during the booking process (6 months prior to departure day) in order to get a seat in the low fare class before it becomes full. So the value of parameter a for the tourist is –180 (180 days before departure).

We also assume that the probability of a tourist arriving to book a seat gradually increases as we get closer to the departure day. After day c = -30 we assume that a tourist will definitely arrive everyday up to the day of departure b = 0, i.e. after c = -30 the probability of an arrival of a tourist passenger is equal to 1. So on day = -150 (i.e. 5 months before departure) the probability of a tourist passenger arriving on this day is 0.0022 (equation 4.2). Multiply the result by $s_t = 90$ is $p\_arr_{tour}(-150) = 0.2$. Figures 4.2, 4.3 and 4.4 shows the values of the function with respect to the time for business, tourist and student passengers respectively.

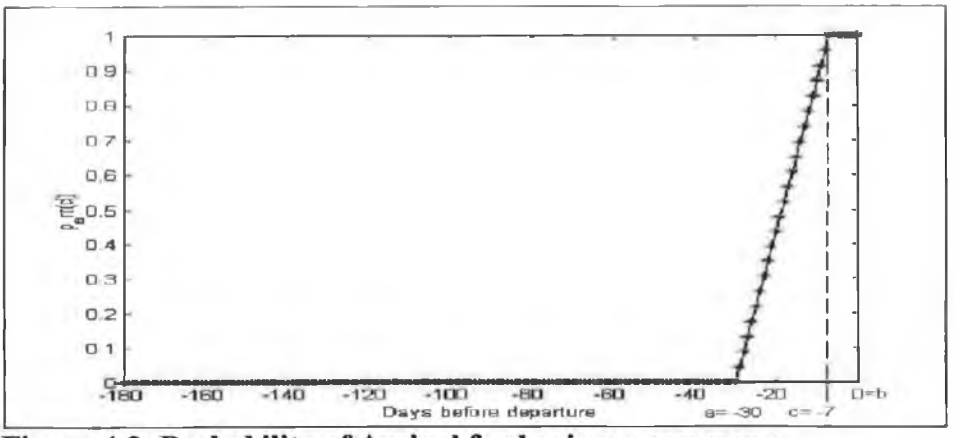


Figure 4.2: Probability of Arrival for business passenger

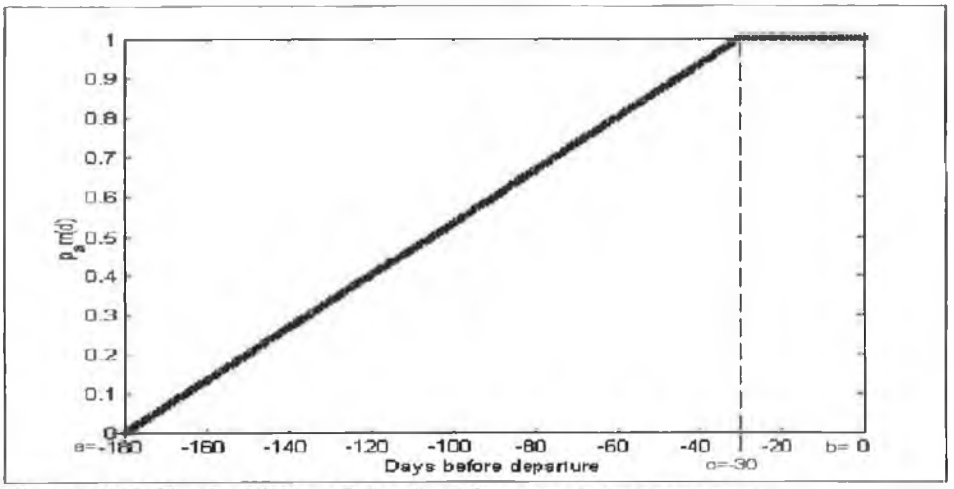


Figure 4.3: Probability of Arrival for tourist passenger

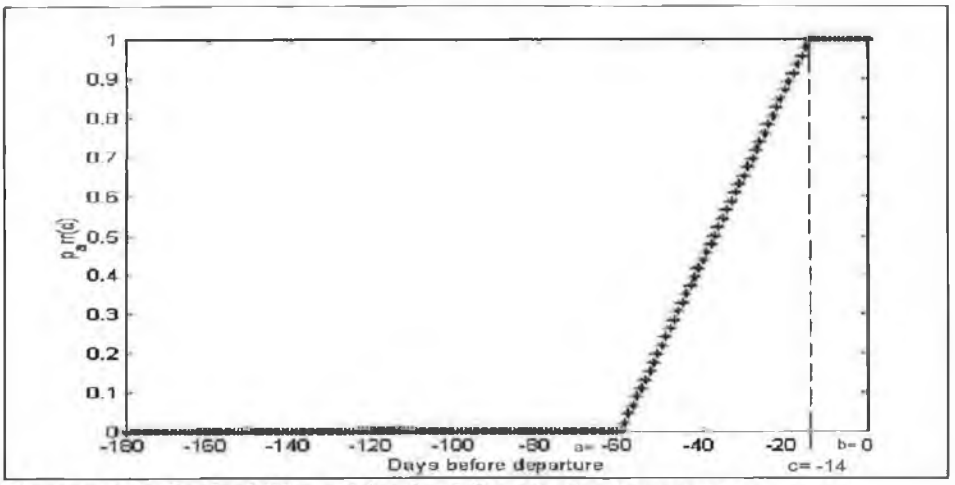


Figure 4.4: Probability of Arrival for student passenger

4.3.6 Booking the passenger on the desired class

We assume that the cost and comfort values and the number of seats in each fare class are known and fixed at the beginning of the booking process.
For each day in the booking process (181 days), for each time interval of the 10 time intervals a day and for each passenger type we check if there is an arrival of this passenger type, (refer to the model flowchart Figure 4.5a in section 4.3.7). We perform the following steps:

1. For each of the ten hours we generate a uniformly distributed random number between 0 and 1, if the calculated probability of arrival *p_arr* from the previous section is greater than the random number then we assume an arrival and continue with step 2. If there is no arrival of any type of passenger on this day then we repeat this process for the next day in the booking period.

2. We then check if there are still seats available on at least one class then continue with step 3. If the flight is full and all seats in all classes have been booked we increment the counter for horizontal or lost passenger due to full flight.

3. If any class is available (refer to Figure 4.5b) and the passenger is willing to accept it then we check if this class is the most desired class, if yes then we increment the counter of total passengers booked on this class and the counter of seats booked by this type of passenger on this class.

4. If this class is not the most desired class and the passenger is willing to accept it then, as well as incrementing the previous counters, we also increment the counter of vertical recapture of passenger.

5. If the passenger is not willing to accept the class we increment the counter of horizontal recapture or lost passenger. At the end of the simulation model we would get the total number of incremental bookings on each class for each day.

4.3.7 Model Flowchart

A flowchart of the proposed model is shown in Figure 4.5a and 4.5b.

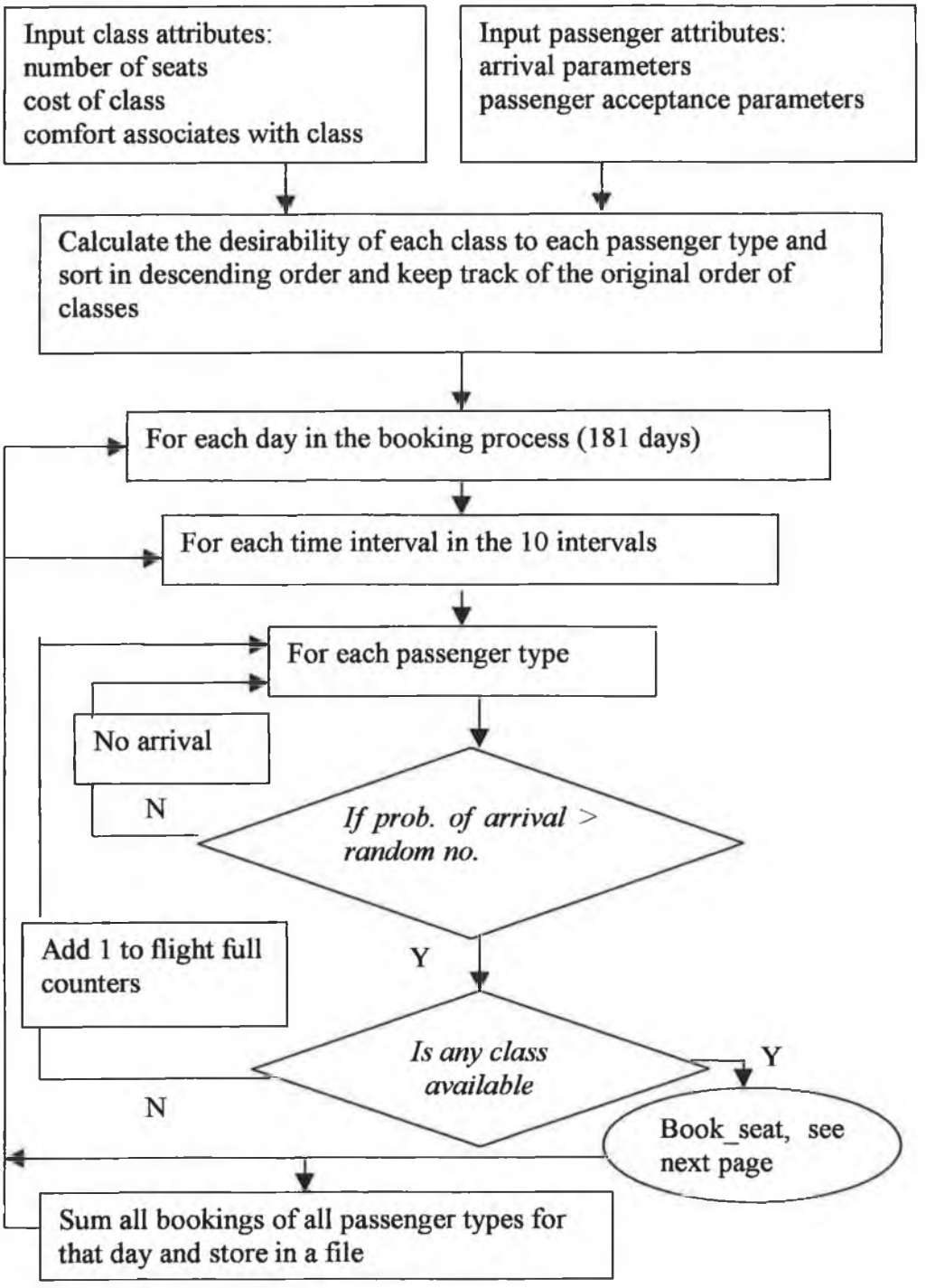


Figure 4.5a: Booking process simulation model

Book_seat:

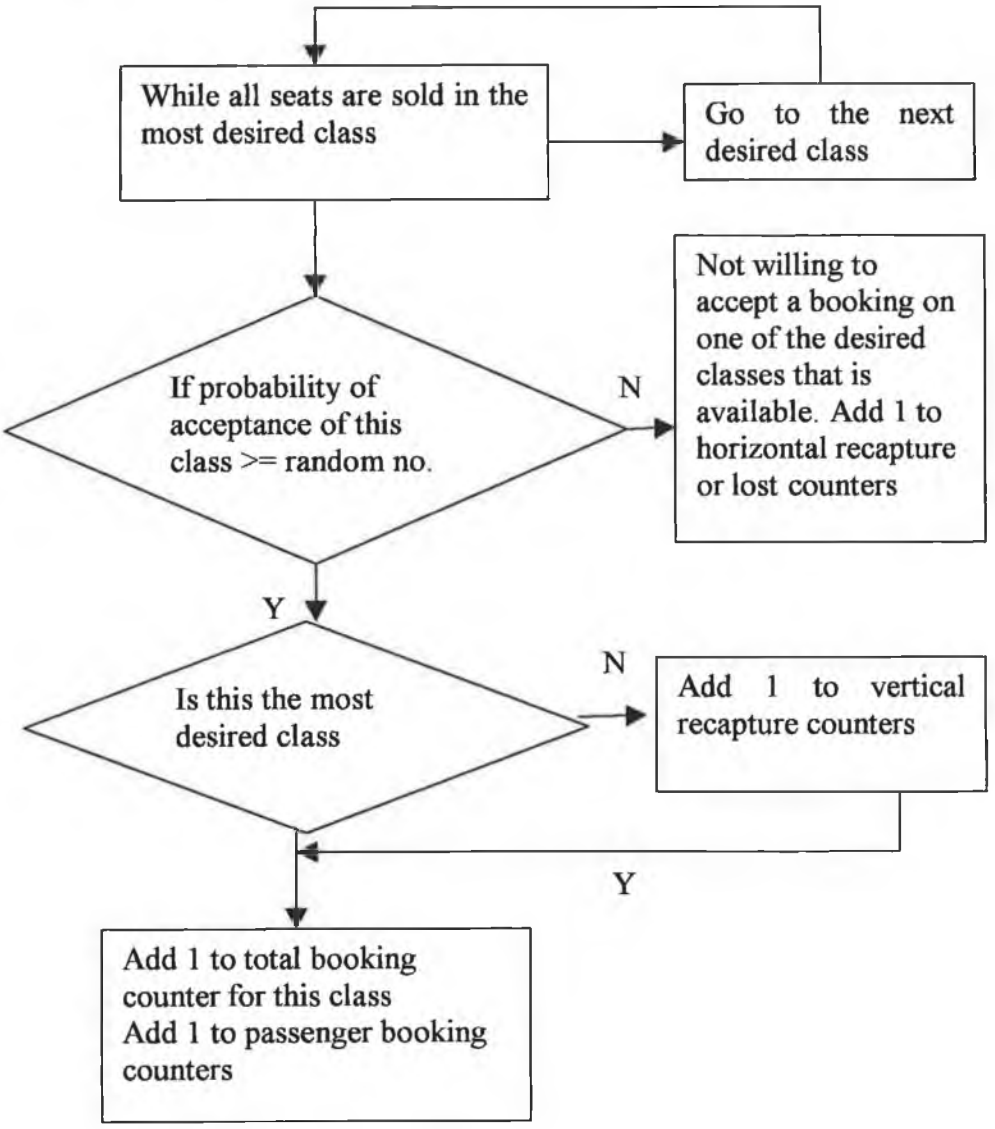


Figure 4.5b: Booking process simulation model

4.4 Generating sample data

For illustration purposes, as shown in Table 4.4, we ran the simulation with the following number of seats assigned to different classes in order to show the contents of the various counters used in the simulation: 20 seats are assigned to the high fare class; 100 seats assigned to the medium fare class; 150 seats assigned to the low fare class. No cancellation allowed on any class. All seats were booked on all classes. The number of business passengers booked on the high fare class is 13 and no business passenger booked on any of the lower classes. There is no vertical recapture of a business passenger either from the high fare class to the low fare class or from the low fare class to the high fare class. There is 1 business passenger who did not accept a booking on the most desired class i.e. horizontal recapture or lost. There are 178 horizontal recapture or lost business passenger due to a full flight. The same explanation can be inferred for the tourist and student passenger counters.

Seats on class: 0:	20
Seats on class: 1:	100
Seats on class: 2:	150
Cancelled bookings on class: 0:	0
Cancelled bookings on class: 1:	0
Cancelled bookings on class: 2:	0
Total seats booked on class: 0:	20
Total seats booked on class: 1:	100
Total seats booked on class: 2:	150
BUS:	
BUS booked on class 0:	13
vertical recapture on class 0:	0
horizontal recapture or lost on class 0:	1
BUS booked on class 1:	0
vertical recapture on class 1:	0
horizontal recapture or lost on class 1:	0
BUS booked on class 2:	0
vertical recapture on class 2:	0
horizontal recapture or lost on class 2:	0
horizontal recapture or lost due to full flight:	178
TOUR:	
TOUR booked on class 0:	0
vertical recapture on class 0:	0
horizontal recapture or lost on class 0:	7
TOUR booked on class 1:	0
vertical recapture on class 1:	0
horizontal recapture or lost on class 1:	0
TOUR booked on class 2:	117
vertical recapture on class 2:	0
horizontal recapture or lost on class 2:	149
horizontal recapture or lost due to full flight:	219
STD:	
STD booked on class 0:	7
vertical recapture on class 0:	7
horizontal recapture or lost on class 0:	0
STD booked on class 1:	100
vertical recapture on class 1:	0
horizontal recapture or lost on class 1:	0
STD booked on class 2:	33
vertical recapture on class 2:	33
horizontal recapture or lost on class 2:	0
horizontal recapture or lost due to full flight:	215

Table 4.4: A sample of model counters

Figure 4.6 shows the sample booking curves for all the fare classes. The low fare class tends to be booked earlier in the booking process, then the medium fare class, and then the high fare class. A sample data booking table is provided in section A.1 of Appendix A.

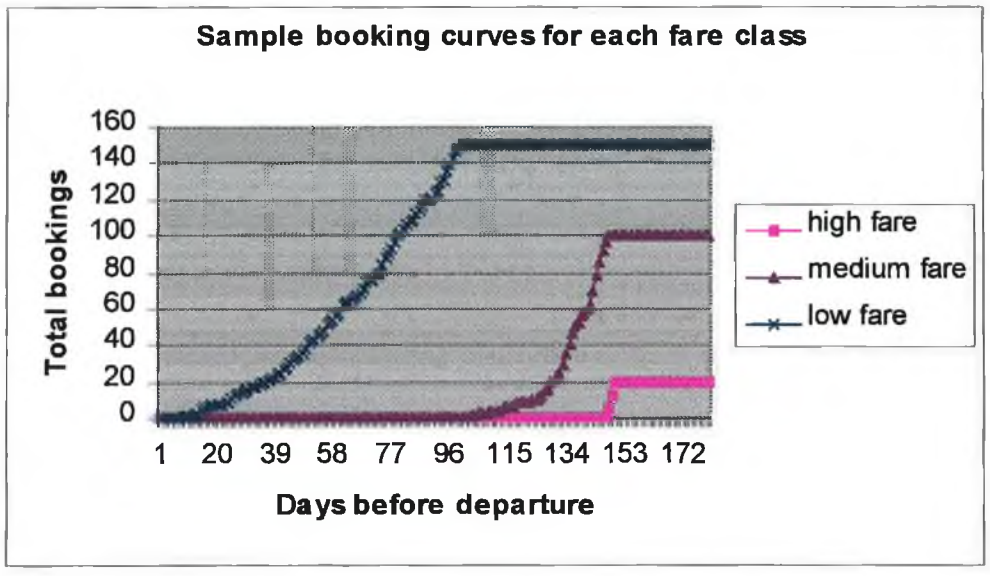


Figure 4.6: Sample booking curves for all classes

The simulation is repeated 1000 times in order to produce minimum and maximum booking values for each class for each passenger type in each day during the booking process (181 days) as well as the mean booking values of the same. Figure 4.7 shows the minimum, maximum and mean booking curves for tourist passengers.

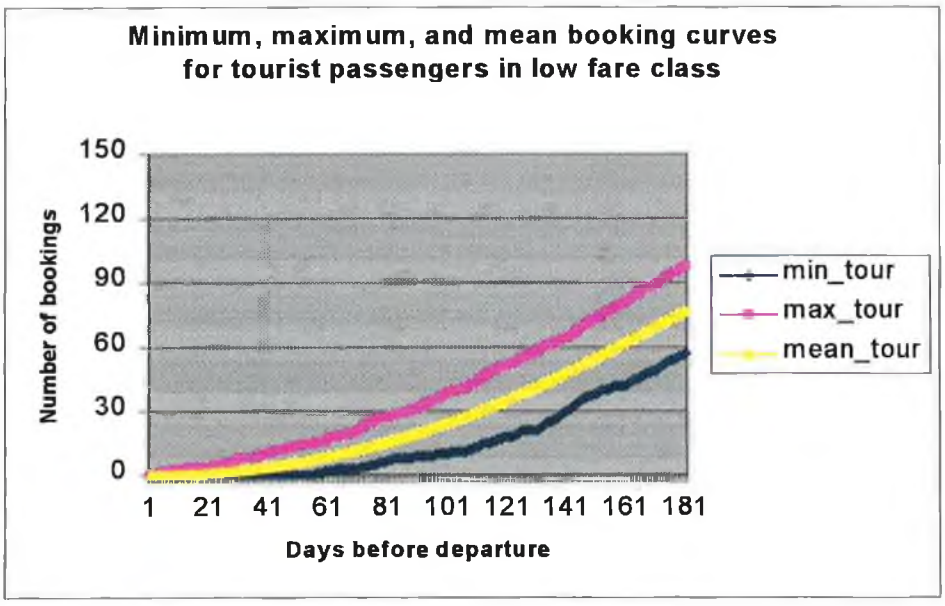


Figure 4.7: Minimum, maximum and mean booking curves for tourist passengers

Now let us illustrate the effect of using different sets of parameters on the booking curves. First we changed the parameters used for the probability of arrival for both types of passenger (refer to Table 4.2 and section 4.3.5).

From the different sets of parameters we changed the first parameter, which is used to represent the first day tourist passengers would likely start arriving, and the third parameter which represents the starting day of definite arrival until the day of departure.

The same was done for the business passengers, Figure 4.8 shows two different mean booking curves for tourist passengers on the low fare class for the first two arrival parameter sets. By decreasing the first parameter, **a,** which is the first day of arrival from -180 to -145 the number of bookings of tourist passengers decreased. Figure 4.9 shows two different mean booking curves for business passengers on the high fare class for the first two parameter sets, we also reduced the first day of arrival from 30 to 20 and that caused the booking curve to go down.

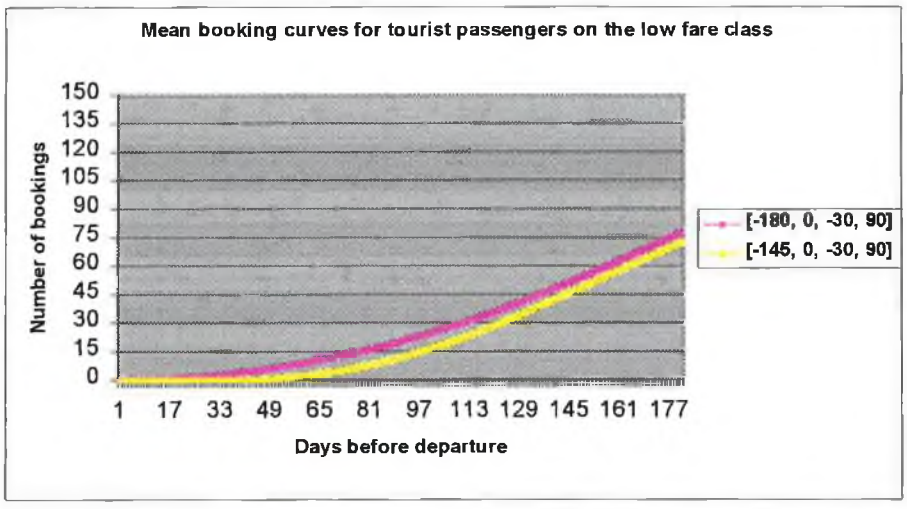


Figure 4.8: Mean booking curves for tourist passengers with different sets of arrival parameters

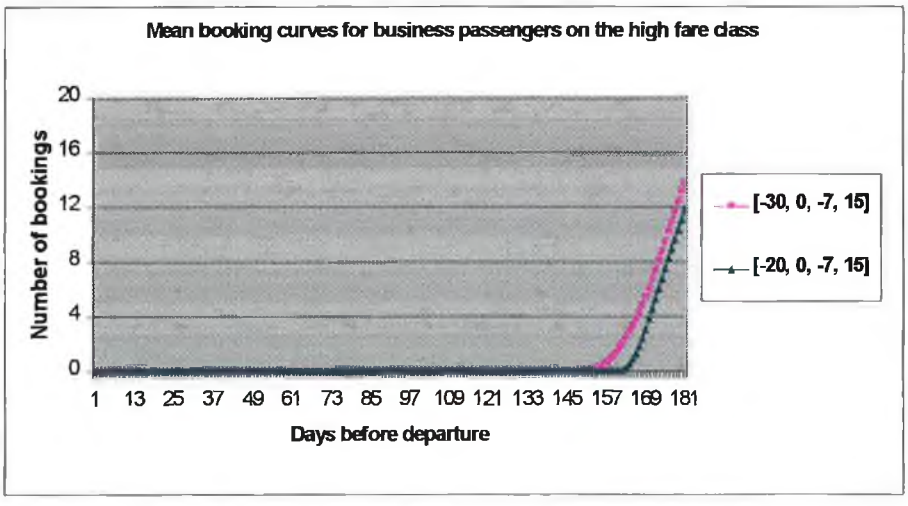


Figure 4.9: Mean booking curves for business passengers with different sets of arrival parameters

For calculating the desirability and the probability of accepting the desired class we changed the parameters of passenger belief for each type of passengers (refer to Table 4.2 and sections 4.3.3, 4.3.4). For tourist passengers we calculate the desirability for the high and low fare class by substituting in equation 4.1 and 4.2. The probability of accepting the high fare class has increased for the tourist passenger from 0.269 to 0.73 by changing the comfort and cost parameters, but also the probability of accepting the low fare class has increased from 0.73 to 0.9 which affected the mean booking curve to go higher from 77 passengers to 95 passengers on the day of departure as shown in Figure 4.10.

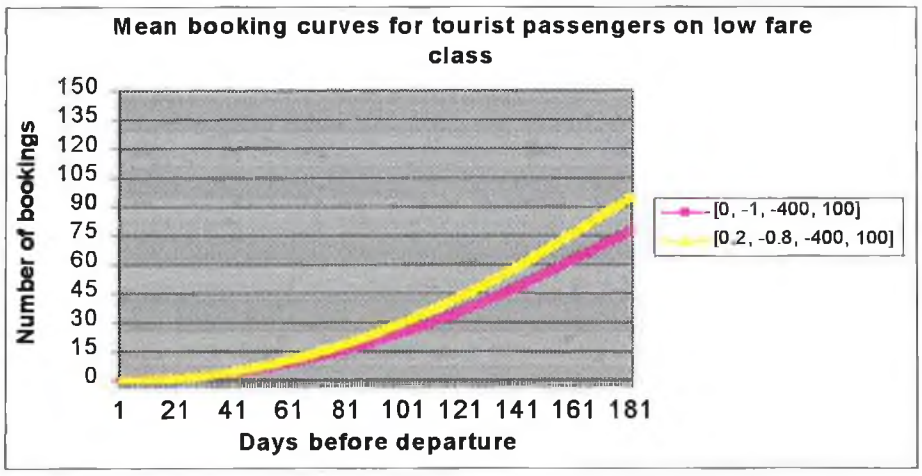


Figure 4.10: Mean booking curves for tourist passengers with different sets of accepting parameters

4.5 Summary

This chapter has developed a probabilistic model for the airline booking process based on Monte Carlo simulation method. The model was able to recognise the different behaviour of different passenger types towards a verity of classes that are available on the flight. We generated sample booking data for the 181 days of the booking process for each class based on simplifying assumptions. We were able to produce a variety of booking curves for the booking period by changing the passenger parameters (characteristics) used in the model. We were also able

to capture some hidden events that are not available in the real-world data. For example, in the reservation systems, there is no record of a passenger non-arrival event or the decision by an arrived customer not to accept the most desired class available and opt for no purchase at all. We showed that the random utility model with discrete choice set works well for passenger choice behaviour and modelled the number of bookings on each class. Vertical recapture was also considered when the most desired class is full and the passenger decides to book on the second or third desired class that is available.

We modelled the arrival behaviour for different passenger as a piecewise linear function. The passenger arrival parameters for each type were assumed to be known. We modelled the probability of accepting to book on a class using the Logistic (Logit) function. The Logistic function performs well in modelling the passenger choice behaviour because its value increases as the desirability of a passenger towards a class increases. We also produced minimum, maximum and mean booking curves from 1000 similar simulated flights. In order to make these booking curves more realistic we really need to take into consideration other realistic factors such as the cancellation and the no-show events.

CHAPTER 5

Fitting the Model to the Data

5.1 Introduction

In Chapter 3 we provided a general description of how to construct a Markov Chain using the Metropolis algorithm in order to draw samples from the posterior distribution of the model, $P(\theta \mid D)$. Sample booking data D was created in Chapter 4, using three different passenger types defined by their parameters (arrival and acceptance parameters). In this chapter we will generate random combinations of the passenger parameters θ (only the accepting parameters) in order to fit the model to the given data D from Chapter 4. This process will be done using the Metropolis algorithm. As we intend to run the Metropolis algorithm for 100,000 iterations in order to get a reasonable collection of parameter combinations that would represent the posterior distribution $P(\theta \mid D)$, we decided to pre-compute all the necessary probabilities needed for our mathematical operations and store them in files to speed up the process.

5.2 Pre-computation of Probability Distributions

In this section we show how to calculate the probability of any given number of bookings on any given class on any given day. This depends on the values of *p_acc* for each of the three passenger types for that class. It also depends on the values of *p_arr* for each of the three passenger types for that day. We have a three-stage procedure. In the first stage we calculate the probabilities of each of the eleven possible numbers of arrivals (from 0 to 10) for each of the three passenger types. We then calculate the probablities of each of the possible number of acceptances from each of the possible number of arrivals for each of the three passenger types for each of eleven different values of *p_acc* (from 0 to 1.0). We then sum the probabilities of all the possible combinations, which could

give rise to a given number of bookings. There are thirty-one possible numbers of bookings on any given day (from 0 to 30).

5.2.1 Creating a table for Probability of Arrival

In chapter 4 we used a triangular function to represent the probability of arrival *p_arr* during one of the ten time intervals of a passenger type in any given day of the booking process (181 days), see section 4.3.5. We also assumed that there are maximum of 10 arrivals of any passenger type in any given day, see section 4.3.1 for model assumptions. In order to calculate the probability of any number of arrivals x from the maximum number of possible arrivals n=10, we used the Binomial distribution, with the probability of occurrence *p_arr* taken from the triangular function. We assume we know in advance what the passenger arrival parameters are: the same parameters are used to generate the booking data, and are the input to the triangular function to give $p\_arr_t(d)$ for each type of passenger on each day. Then *p_arr* will be used in the Binomial distribution to produce the probability distribution of arrival of x [0-10] passengers of type t on each day d.

All possible probabilities of arrivals for each type of passengers on each day were stored in a table called P_arrive. Figures 5.1, 5.2 and 5.3 show the probability distribution for the business passenger arrival on day –26 with p_arr_{bus} = 0.173913, tourist passenger arrival on day -120 with p_arr_{tour} = 0.4 and student passenger arrival on day –40 with p_arr_{std} = 0.434783 respectively. A sample of the table is provided in section A.2 of Appendix A.

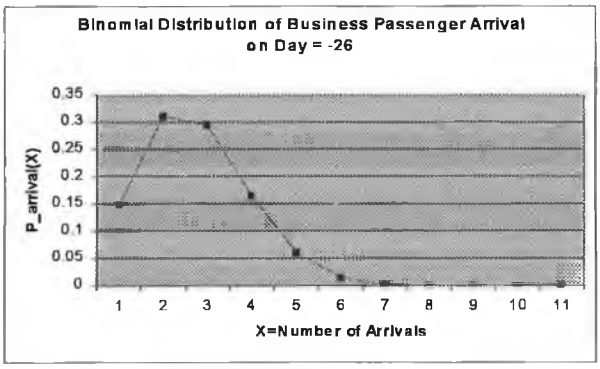


Figure 5.1: Probability Distribution for the business passenger arrivals on day –26

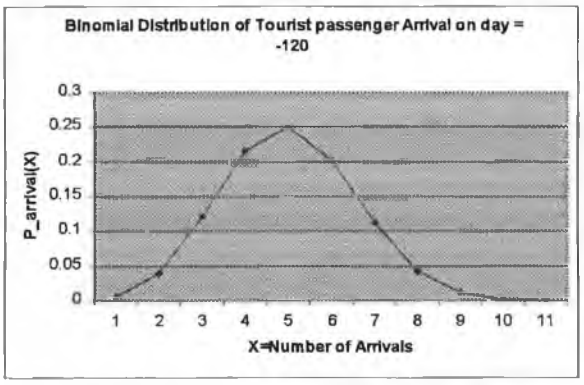


Figure 5.2: Probability Distribution for the tourist passenger arrivals on day –120

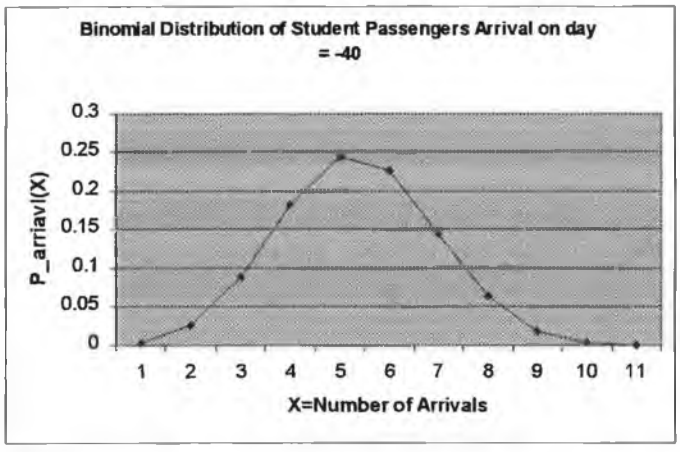


Figure 5.3: Probability Distribution for the student passenger arrivals on day –40

5.2.2 Creating a table for Probability of Acceptance

The Binomial distribution can also be used to model the behaviour of the passenger's willingness to accept. In section 4.3.4 of the previous chapter we used the logistic function to calculate the probability of accepting a booking according to desirability value, and we generated a number between 0 and 1 representing the probability of acceptance for each type of passenger in each class. When we have one arrival then we need to calculate the probability of either the passenger accepting a booking or not. When we have two arrivals then we have three possible outcomes: either no acceptance of a booking or one acceptance or two acceptances. This can be repeated up to 10 maximum arrivals. In order to create a table the probability of acceptance *p_acc* is discretised into 11 values starting from *p_acc*=0 to *p_acc*=1 with intervals of 0.1. For each value of *p_acc* and for each value of possible arrivals *x* we will calculate the probability of accepting bookings from *y* passengers.

All possible probabilities of acceptance from all possible arrivals for each value of *p_acc* were stored in a table called P_accept. Notice that this table can be used for any type of passengers, it only depends on the value of *p_acc* which is derived from the passenger belief parameters (refer to section 4.3.6 and equation 4.2). Figure 5.4 shows the probability distribution for accepting to book when *p_acc* = 0.8 and number of maximum arrivals n = 10. A sample of the table is provided in section A.3 of Appendix A.

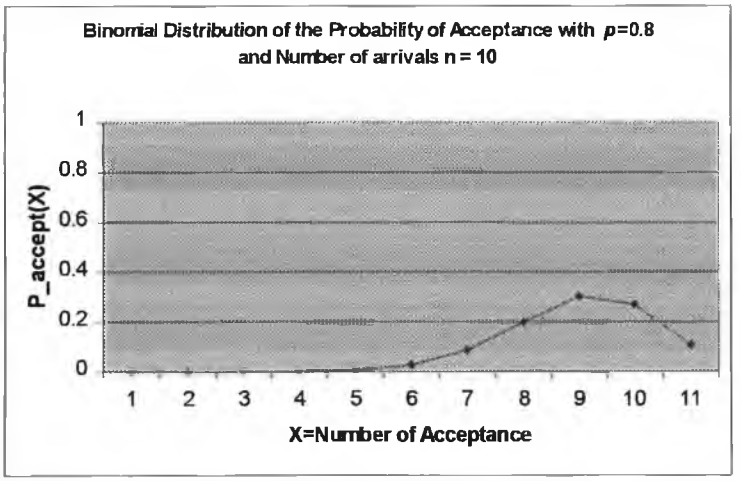


Figure 5.4: Probability Distribution for all possible number of acceptances when p = 0.9 and maximum arrivals n = 10

5.2.3 Creating a table for Total Probability of Bookings

The sample booking data that the simulation model generates consists of the total number of bookings on each fare class on each day of the booking process (refer to section 4.4 and section A.1 of Appendix A). On any day, the only data available is the total number of booking in each fare class. We actually do not know who booked this seat, was it a tourist or student passenger or perhaps a business passenger. We also do not know how many people arrived and did not book at all. For example, if we have 12 bookings on any given day, then these bookings might have come from one of the following combinations:

	10 business, 2 tourist, 0 student
or	10 business, 1 tourist, 1 student
or	10 business, 0 tourist, 2 student
or	9 business, 3 tourist, 0 student
or	9 business, 2 tourist, 1 student
or	9 business, 1 tourist, 2 student
.	
and so on.	

Notice that for each of the above cases, the number of bookings does not mean the total number of arrivals, because the data only presents the total number of acceptances on that day on each class. The number of arrivals might be at least the number of bookings or any number lying between the number of bookings and the maximum number of arrivals (30). On each day, for each number of possible bookings in the range [0-30] we want to calculate the overall probability of all the possible passenger combinations that might have produced this number of bookings.

In order to calculate the probability of any number of bookings b on any given day d for each passenger type t we must multiply the probability of at least x arrivals (taken from the table of probabilities of arrival) by the probability of b acceptances out of x arrivals (taken from the table of probabilities of acceptance). Then we sum over all possible arrivals x, $b \le x \le 10$

$$p\_book_{t,d}(p\_acc,b) = \sum_{x=b}^{10} p\_arr_{t,d}(x) * p\_acc(x,b) \qquad (5.1)$$

Remember that the probability of x acceptances depends on p_acc in the interval [0-1] (refer to section 5.2.2). In order to build a table that covers all possible situations we need to calculate the overall probabilities for all possible combinations of p_acc associated with each type t for each day d.

If we have 3 passenger types then we have p_acc_1, p_acc_2, p_acc_3.

On any given day d we have the total number of bookings (on each class) b_d in the range [0-30].

If the number of booking of each passenger type is x_1, x_2, x_3, then $b_d = x_1 + x_2 + x_3$

We calculate and sum the probabilities of each of the combinations of x_1, x_2, x_3 which sum to b_d.

For each day, for the three types of passenger with *p_acc* combinations starting from [0, 0, 0] to [1, 1, 1], and for each number of total bookings [0-30], *P_tot* will be calculated and stored in Table P_allb. Figure 5.5 illustrates the steps taken to create the table. A sample of the table is shown in section A.4 of Appendix A. This table will be used in every Metropolis iteration (100000 iterations) to calculate the likelihood of all the booking data (181 days in all classes) given the random combination of passenger belief parameters generated for this iteration. The reason we decided to pre-compute all possible cases and store them in a table is to reduce the run time of the algorithm. We actually reduced the run time up to 20% from the original version of the MCMC model. A layout of the binary file that contains the table and the equations involved in calculating the correct read-location from the file is provided in section A.5 of Appendix A.

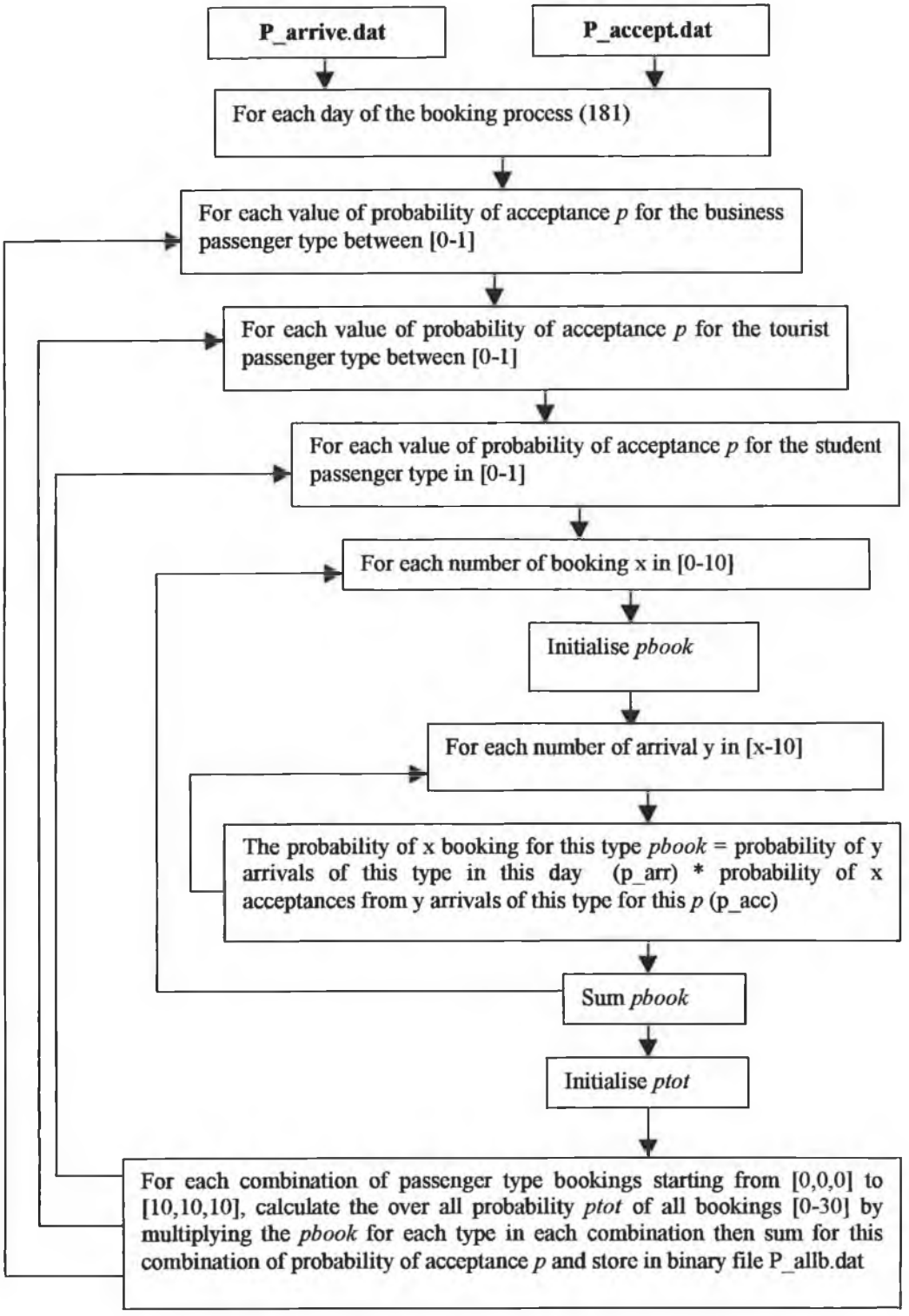


Figure 5.5: Pre-computation of overall probabilities

5.3 Estimation of Model parameters using Metropolis Algorithm

In Chapter 4 we built an MC simulation model to generate a sample data for one flight for each day of the booking process (181 days) on each fare class. This simulated data will be used to estimate the model parameters (only the passenger belief parameters) using the Metropolis algorithm (refer to section 3.3 in Chapter 3). By running the algorithm for a large number of iterations (100,000), i.e. constructing a Markov chain, we would expect the algorithm to converge and reach an equilibrium state after a number of iterations m<100,000. When the algorithm terminates, the random sample of parameters combinations that were accepted should represents the posterior distribution $P(\theta \mid D)$.

5.3.1 Start the algorithm with an arbitrary starting point

Recall that there are three passenger belief parameters, p_{t1} in the range [0-1]; p_{t2} in the range [-1 – 0]; p_{t3} in the range [-500 - -300] (refer to Table 4.2). The algorithm will be started with any values for the three parameters within the range shown above for all passenger types t. The choice of the starting values should not affect the result of the Metropolis algorithm in the MCMC model since it 'forgets' its initial state after it runs long enough.

5.3.2 Change one parameter at random

Generating a random number [1-3] determines which passenger type (3 types) to change its parameter. Another random number [1-3] will be generated to determine the parameter that has to be changed for this passenger type. Finally a random number will be generated and added to the chosen parameter value, keeping it within its valid range (refer to section 5.3.1).

5.3.3 Calculate the likelihood

In each iteration i we will calculate the likelihood of the sample data given the current parameter combination θ_i. For each day of the booking process d and for

each class c, first we find how many bookings occurred $b_{d,c}$. We have already calculated the probability of acceptance of each passenger type t to each class c, $p\_acc_{t,c}$, (refer to section 5.3.1 and 5.3.2).

For each class c we must obtain a set of $p\_acc_{t,c}$ for each passenger type as follows:

If we have 3 passenger types then for each class we should have the set or index $Index_c = [p\_acc_{t1,c}, p\_acc_{t2,c}, p\_acc_{t3,c}]$. If this class is the most desired available class to the passenger type t, then accept p_acc for this type, otherwise p_acc for this type on this class is 0. Given the day d and the total bookings on class c and the set of p_acc values for all passenger types on class c, we obtain the overall likelihood of this event $P\_tot(b_{d,c} \mid \theta_i)$ from the Table P_allb.

For example if $Index_{c1}$ = [0.8, 0, 0] means that only passenger type t_1 has class c_1 at its most desired class.

Recall that the p_acc values that were used to create the Table P_accept were in the range [0-1] with intervals of 0.1 (refer to section 5.2.2 and section A.3 of Appendix A). However, the p_acc that was calculated for each parameter combination can be any value in the interval [0-1] (refer to section 4.3.3 and 4.3.4) i.e. not just one decimal point. So we used linear interpolation (see Van Loan 1999)

5.3.4 Using the Metropolis Algorithm

In the previous section we were able to calculate the likelihood of the total bookings on each class given a random combination of parameters θ_i. In order to calculate the likelihood of all the bookings on all classes and all days, we need to multiply all the $P\_tot(b_{d,c} \mid \theta_i)$ values obtained from Table P_allb in section 5.2.3, $L(D \mid \theta_i) = \prod P\_tot(b_{d,c} \mid \theta_i)$. Instead of doing that we calculate the natural logarithm of each $P\_tot(b_{d,c} \mid \theta_i)$ and sum over all classes and all days of the booking process to obtain the log_likelihood as follows:

If we have 3 classes calculate:

$$Log\_like_{d1} = \sum_{c=1}^{3} \ln(P\_tot(b_{d1,c} \mid \theta_i)) \tag{5.3}$$

Then for all days of the booking process (181) and for this iteration i of the Metropolis algorithm calculate:

$$Log\_like_i = \sum_{d=0}^{180} \ln(P\_tot(b_d \mid \theta_i)) \tag{5.4}$$

In order to compare the likelihood between the current iteration i and the previous iteration $i-1$ we calculate:

$$Max(Log\_like_i, Log\_like_{i-1}) = \text{TRUE} \tag{5.5}$$

If the parameter combination θ_i satisfies the condition in equation (5.5), then repeat the next iteration of the Metropolis algorithm with the parameter combination $\theta_{i+1} = \theta_i$, i.e. the chain moves to the next state.
If not then perform the following test:

$$\text{If } (\exp(Log\_like_i - Log\_like_{i-1}) \geq \text{random no. from U[0-1]}) \tag{5.6}$$

If the parameter combination θ_i satisfies the condition in inequality (5.6), then repeat the next iteration of the Metropolis algorithm with the parameter combination $\theta_{i+1} = \theta_i$, i.e. the chain moves to the next state. This test protects the algorithm from getting trapped in local minima. Otherwise repeat the algorithm with θ_{i-1}. After a certain number of iterations, say n, the Markov chain $\{\theta_0,.,.\theta_n,..,\theta_m\}$ will converge and parameter combinations accepted after iteration

n will be, approximately, samples from the posterior distribution of the model $P(\theta \mid D)$.

5.4 Summary

Before creating the tables for the pre_computed probabilities we faced difficulties with the time taken by the Metropolis algorithm to run (100,000 iterations). So we saved time by first creating a table that contains the probabilities of all possible number of arrivals for each type of passenger. Secondly we created a table for the probabilities of all possible acceptances from all possible arrivals. Then we used these two tables to calculate the total probability for each possible passenger parameter combination and stored in a table. This table is used in each iteration of the algorithm and the algorithm simply reads the required total probabilities in order to calculate the likelihood. We developed a model that creates a Markov Chain that contains samples of the parameter combinations that represent the probability distribution of the model parameters given the data. We used the Metropolis algorithm to construct the chain.

CHAPTER 6

Result Analysis

6.1 Introduction

In Chapter 4 we saw how to generate synthetic data from known parameters. In this chapter we use our MCMC method to estimate the parameters from the data. Our purpose is to test our MCMC algorithm. We wish to see how accurately we can determine the parameters used to generate the data. We conduct a series of tests starting with the simplest possible situation- one passenger type booking on one fare class – and then increase the complexity by introducing more passenger types and more classes.

We construct a Markov chain using the Metropolis algorithm and the simulated booking data D. The chain contains a sequence of random parameter combinations for the passenger belief parameters $\{\theta_0, \theta_1, \theta_2, ...\}$. After a sufficiently long number of iterations the chain converges reaching an equilibrium state. We run the algorithm for 100,000 iterations and output to a file the accepted combination after every 10,000 iterations. This random sample of parameter combinations that were accepted represents the posterior distribution $P(\theta \mid D)$. A point in 3D space can represent the three parameters in each combination. Plotting all combinations of the chain using MATLAB 5.2 we get their distribution in relation to the correct combination used to generate the data.

In section 6.2 we first test the algorithm with simulated data produced from one passenger type booking in one class. In other words we assume that only one passenger type arrives and there is only one class available. We make the number of seats in this class artificially large so that it does not fill up. Then we repeat the same experiment twice more with the same passenger type booking on a different

class each time. Each of the three previous experiments will produce a different plane in parameter space. Then we plot the three planes and examine the area of intersection. This area should include the original parameter combination, which was used to generate the data – so this provides a test of our algorithm.
In section 6.3 we test the algorithm for the case of one passenger type having the choice of booking on two classes. Section 6.4 will give the results of a run for the same passenger type having the choice of booking on three classes together. In section 6.5 we introduce a different passenger type and examine the results of the algorithm for this passenger type and the previous three classes. We then run the algorithm for the previous two passenger types and the same three classes in section 6.6. Finally we add a third passenger type in section 6.7 and examine the results of the algorithm on three passenger types and three classes.

The reason why we expect to have a plane in 3D from each run is because we are dealing with three parameters p_{t1}, p_{t2} and p_{t3} for passenger belief. The first two parameters come from equation (4.1) of the desirability (refer to section 4.3.4):

$$desr_{t,c} = comf_c * p_{t1} + cost_c * p_{t2}$$

The result of this equation is then used to calculate the probability of acceptance using equation (4.2) (refer to section 4.3.5):

$$p\_acc_{t,c}(desr_{t,c}) = \frac{1}{1+e^{-(desr_{t,c}-p_{t3})/z}}$$

Notice that we subtract the third parameter p_{t3} from the desirability $desr_{t,c}$, (z is kept constant through all the experiments). We can rewrite equation (4.1) as follows:

$$desr_{t,c} = comf_c * p_{t1} + cost_c * p_{t2} \qquad (6.1)$$

If the Metropolis algorithm is working correctly it should generate parameter combinations that produce the same probability of acceptance $p\_acc_{t,c}$, i.e. that have similar desirability $desr_{t,c}$. With $\cos t_c$ and $comf_c$ remaining constant during the execution of the experiment and keeping only the points or parameters that generate the same $desr_{t,c}$, then equation (6.1) represents the equation of a plane in a 3D space whose dimensions are p_{t1}, p_{t2} and p_{t3}. This plane should include the original parameter combination, which was used to generate the data.

6.2 One passenger type and one class

6.2.1 One passenger type and one class (class1) for one flight

First we test the algorithm with one passenger type booking on one class for one flight. Figure 6.1 shows the output from the Metropolis algorithm: a plane in 3D representing the posterir distribution over the passenger parameters for this type. The plane does include the original parameter combination, as it should do. The parameters used for this run are shown in Table 6.1. The probability of acceptance for this test is $p\_acc_{t,c} = 0.8$. Remember all points on the plane generate the same $p\_acc_{t,c}$. The thickness of the plane is determined by the amount of data available to the algorithm. We will see later that when we run the algorithm with data from 100 flights instead of data from one flight the degree of uncertainty reduces resulting in thinner planes.

Correct parameters (data parameters):	$p_{t1} = 0.7$	$p_{t2} = -0.8$	$p_{t3} = -400$
The arbitrary starting point:	$p_{t1} = 0.5$	$p_{t2} = -0.2$	$p_{t3} = -450$
Class1 parameters:	$seats_c = 500$	$\cos t_c = 600$	$comf_c = 800$

Table: 6.1 Parameters used for one passenger type and one class (class1)

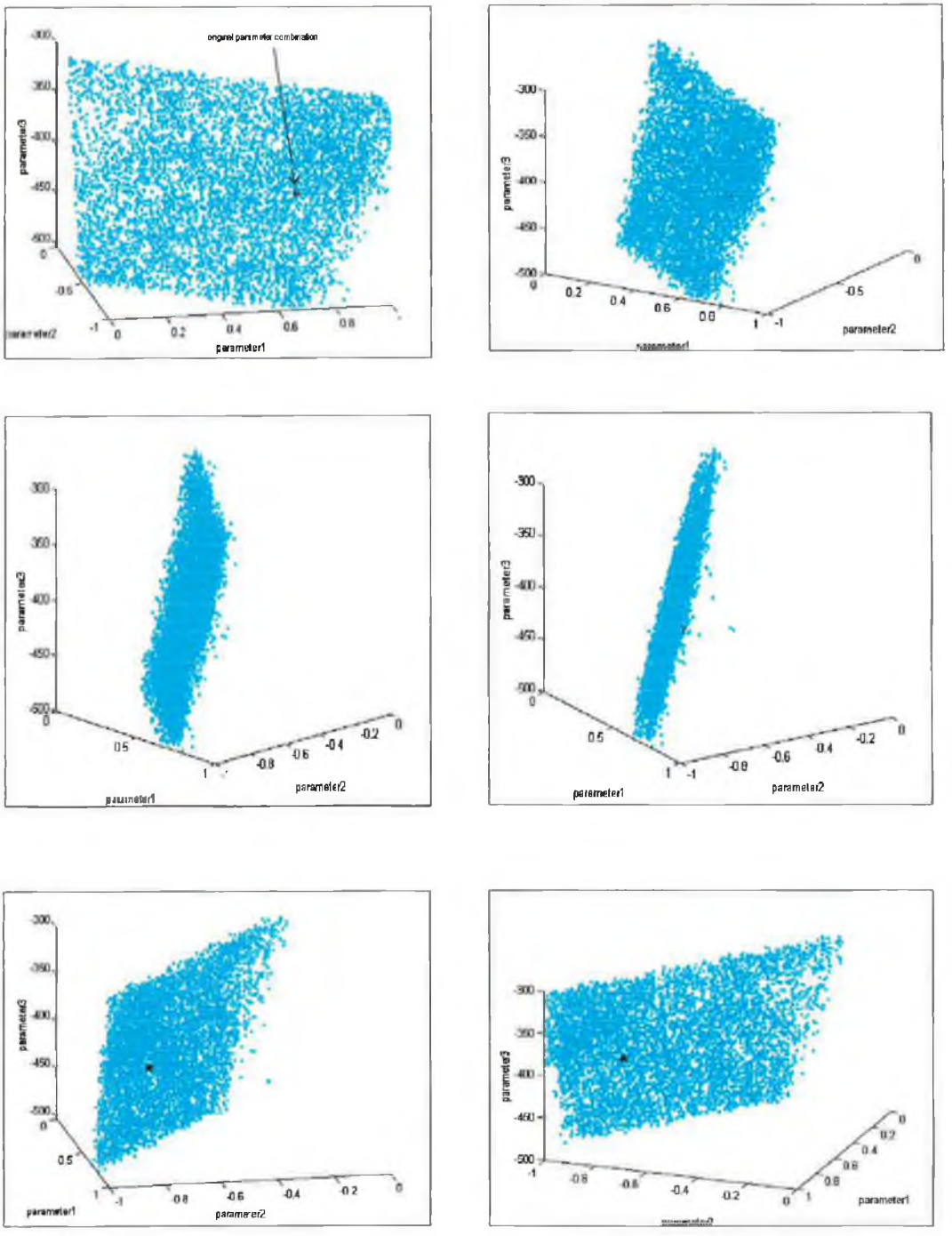


Figure 6.1: A plane viewed from different angles for one passenger type booking on one class with $p\_acc_{t,c}$ = 0.8. The original parameter combination lies in the plane as it should do

6.2.2 One passenger type and another class (class2) for one flight

Then we test the algorithm for another class with the same type of passenger for one flight. Table 6.2 contains the parameters used for this run. The probability of acceptance for this test is $p\_acc_{t,c}$ = 0.5005. Figure 6.2 shows another distribution of this type for another class with a different $p\_acc_{t,c}$. The $\cos t_c$ and $comf_c$ values used in this run are not realistic. They have been contrived to produce a plane that is orthogonal to the previous plane in order that the area of intersection between the two planes is as small as possible.

Correct parameters (data parameters): p_{t1} = 0.7		p_{t2} = -0.8	p_{t3} = -400
The arbitrary starting point:	p_{t1} = 0.5	p_{t2} = -0.2	p_{t3} = -450
Class2 parameters:	$seats_c$ = 500	$\cos t_c$ = -400	$comf_c$ = 150

Table: 6.2 Parameters used for one passenger type and another class (class2)

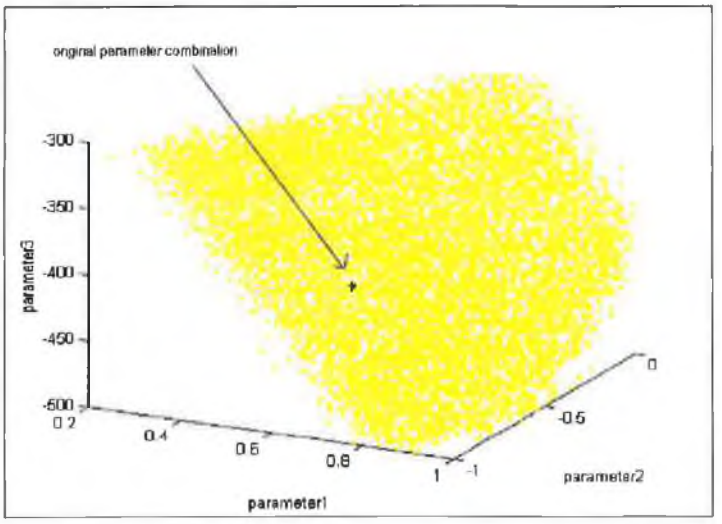


Figure 6.2: One passenger type booking on another class with $p\_acc_{t,c}$ = 0.5005

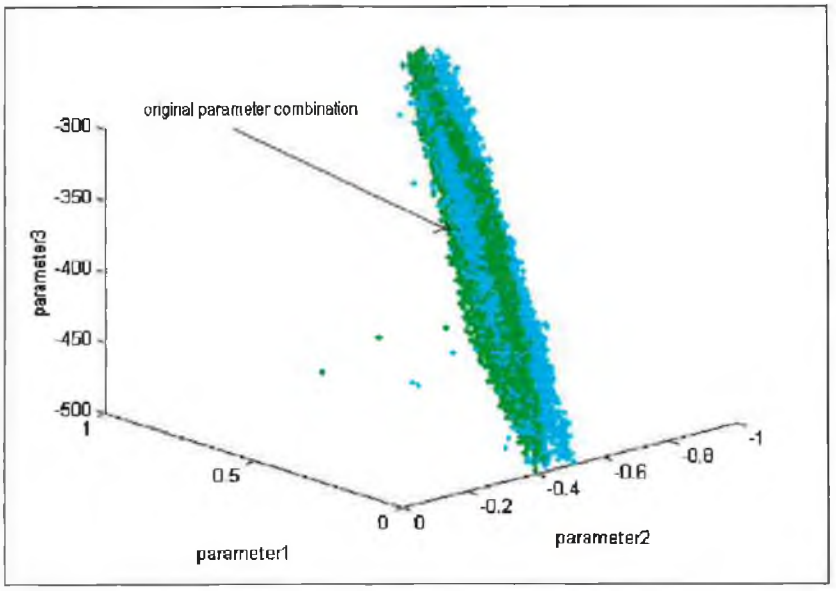


Figure 6.3: Two planes representing two different classes for one passenger type with large area of intersection between them

For comparison Fig. 6.3 shows the result when we run the algorithm with different $\cos t_c$ and $comf_c$ values which produce planes which are nearly parallel. We notice that the two planes representing the two different classes intersect in a large area indicating a high degree of uncertainty in our estimate of the original parameter combination.

6.2.3 One passenger type and another class (class3) for one flight

Then we test the algorithm for a third class keeping the passenger type the same and also for one flight only. Table 6.3 contains the parameters used for this run. The probability of acceptance for this test is $p\_acc_{t,c} = 0.5$. Again $\cos t_c$ and $comf_c$ have been contrived to produce a plane that is orthogonal to the previous two. See Figure 6.4.

Correct parameters (data parameters):	$p_{t1} = 0.7$	$p_{t2} = -0.8$	$p_{t3} = -400$
The arbitrary starting point:	$p_{t1} = 0.5$	$p_{t2} = -0.2$	$p_{t3} = -450$
Class3 parameters:	$seats_c = 500$	$\cos t_c = -114$	$comf_c = 400$

Table: 6.3 Parameters used for one passenger type and another class (class3)

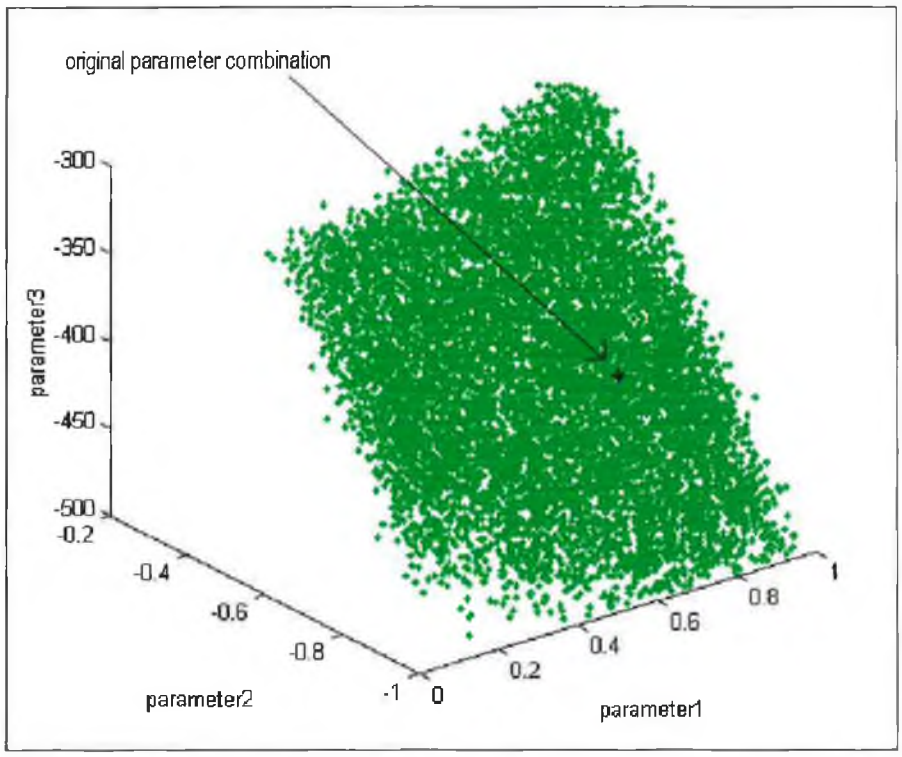


Figure 6.4: One passenger type booking on another class with $p\_acc_{t,c} = 0.5$

6.2.4 Intersection between the planes of the previous classes

Then we plot all the three previous planes as illustrated in Figure 6.5a and 6.5b; they intersect in a region which also includes the correct point that was used to generate the data (0.7, -0.8, -400)

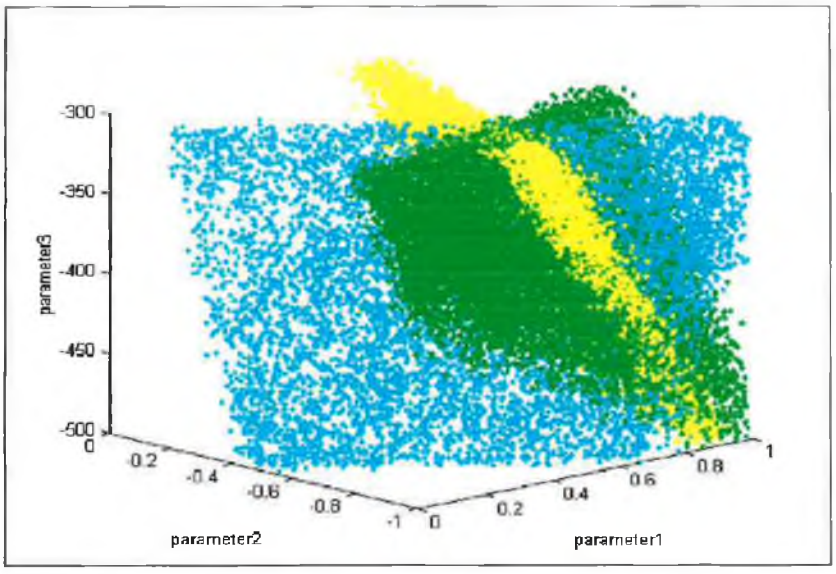


Figure 6.5a: Intersection between the 3 previous planes

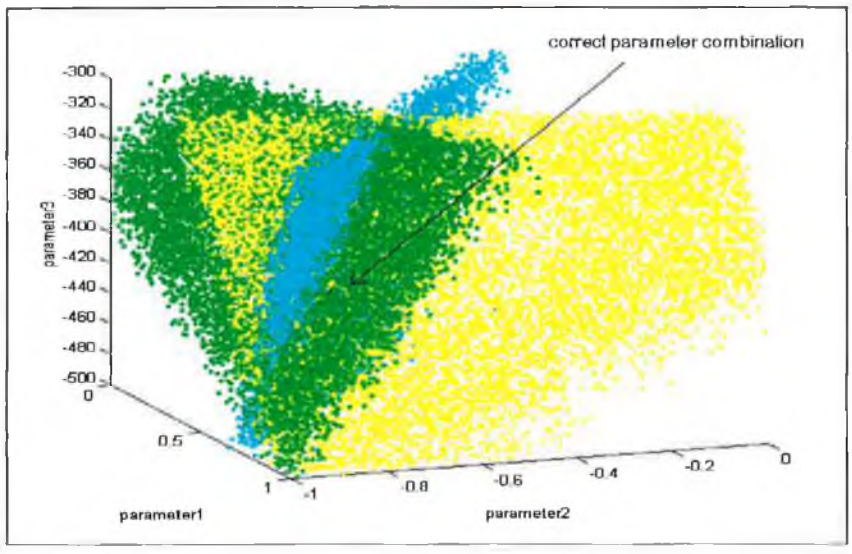


Figure 6.5b: Intersection between the 3 previous planes, different angle

6.3 One passenger type and two classes

In the previous section the passenger only had the choice of booking on one class. We now introduce two classes simultaneously (class1 and class2 from section 6.2.1 and 6.2.2 are used) and allow the same type of passenger used in the previous tests to have the choice of booking in two classes. After the most desired class for this passenger type becomes full then the passenger has the choice either of booking in his second desired class or not. In this case the simulated data includes the effect (influence) of the choice behaviour between two products (or classes). Figure 6.6 shows the booking curve in this case. The booking activity on these two classes should provide us with more information about this passenger type. So running the Metropolis algorithm using this new booking data, the resulting Markov chain should provide more accurate (specific) parameter combinations that better describe this passenger type. Figure 6.7 shows the two planes of the two classes running separately, then Figure 6.8 shows the output of the two classes together in the same run. Notice the location of the output is where the two previous planes from Figure 6.7 intersect. The plot in Figure 6.9 shows the result of the same two classes but run for 100 flights. Using 100 flights reduces the uncertainty, so the distribution is more concentrated but it still contains the original parameter combination. Notice, two classes do not give enough information to pinpoint the original parameter combination exactly.

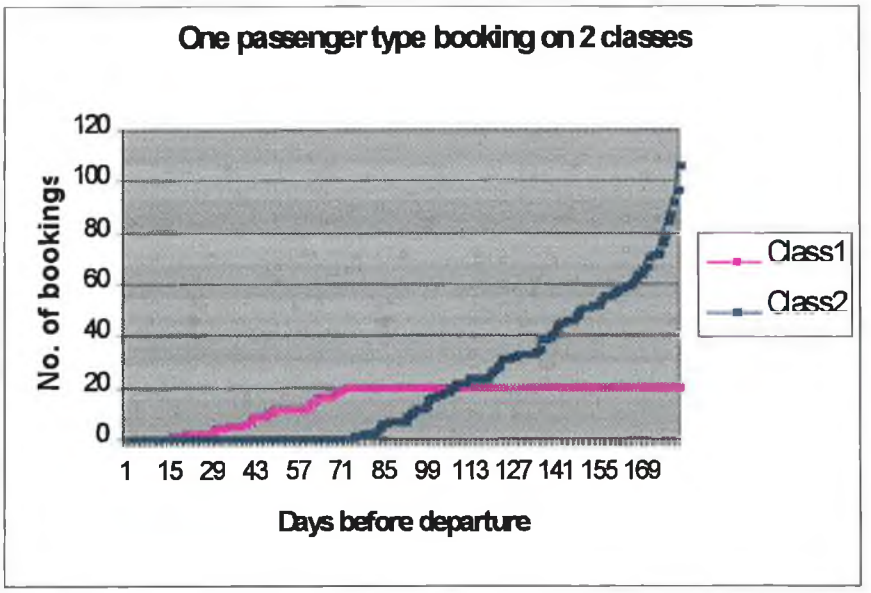


Figure 6.6: Booking curve for one passenger type and two classes

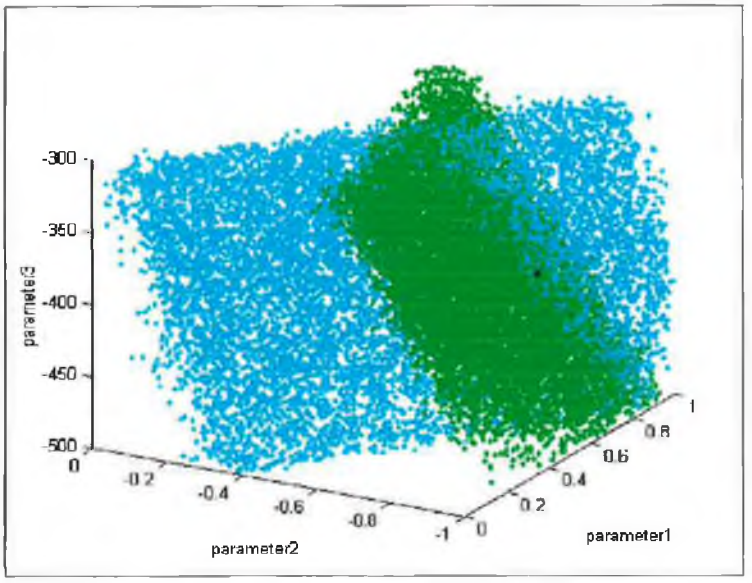


Figure 6.7: Intersection between the runs from one passenger and two classes separately. The original combination is shown as a black dot.

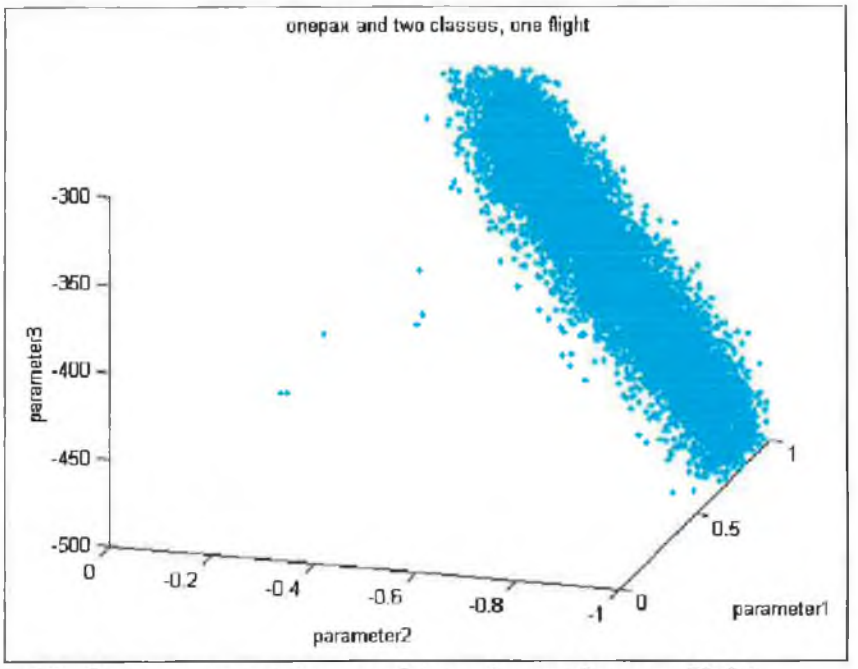


Figure 6.8: One passenger type and two classes for one flight

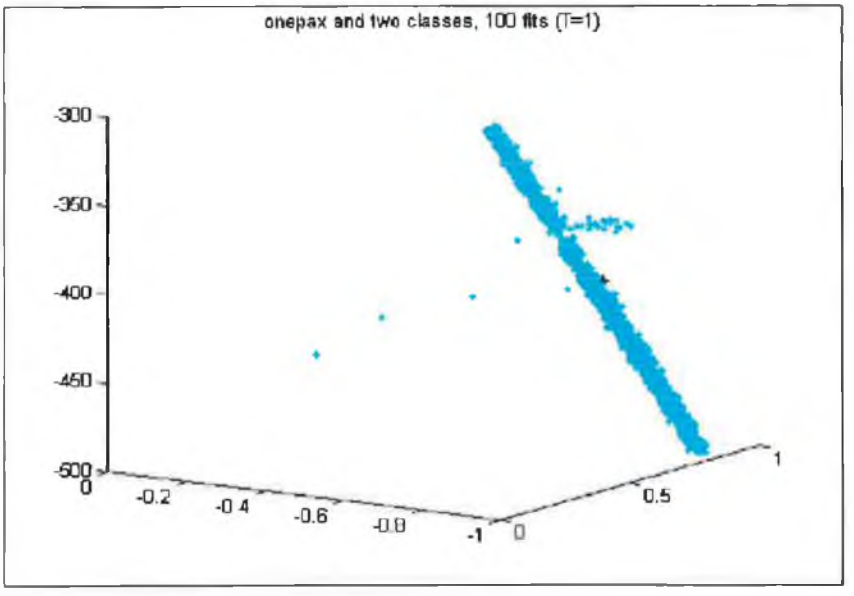


Figure 6.9: One passenger type and two classes for 100 flights. Notice how the spread of the distribution is reduced. The original parameter combination is shown as a black dot.

6.4 One passenger type and three classes

We introduce a third class to give the same passenger type from previous tests the choice between 3 different fare classes (class3 from section 6.2.3 was used). When the most desired class becomes full the passenger has the choice of booking on the second desired class or not to book, and so on, see Figure 6.10. Figure 6.11 shows the posterior distribution for this case. For comparison we repeat Figure 6.5a which shows the planes generated by each class individually. The large blue distribution in Figure 6.11 corresponds to the region of intersection of the three planes in Figure 6.5a.

The red distribution in Figure 6.11 is for 100 flights and the blue distribution is for one flight. Notice that for the 100 flights the distribution is more concentrated. Also notice that the correct parameter combination does not lie in the 100-flight distribution. We think the reason for this is that the correct parameter combination lies on the edge of the region of intersection in Figure 6.5a. The algorithm appears to be biased towards the centre of this region. This problem needs to be addressed in future work.

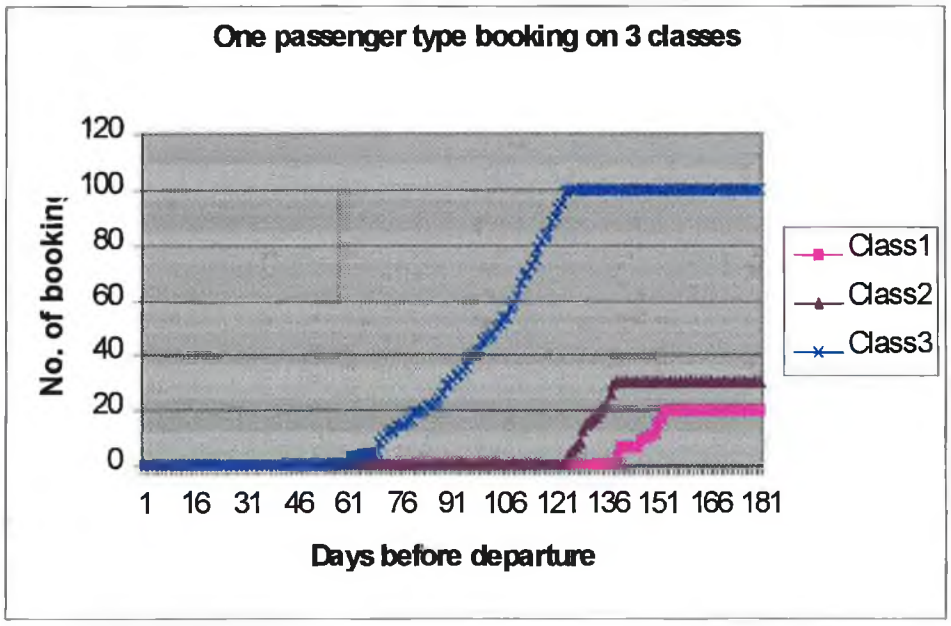


Figure 6.10: Booking curve for one passenger type and three classes

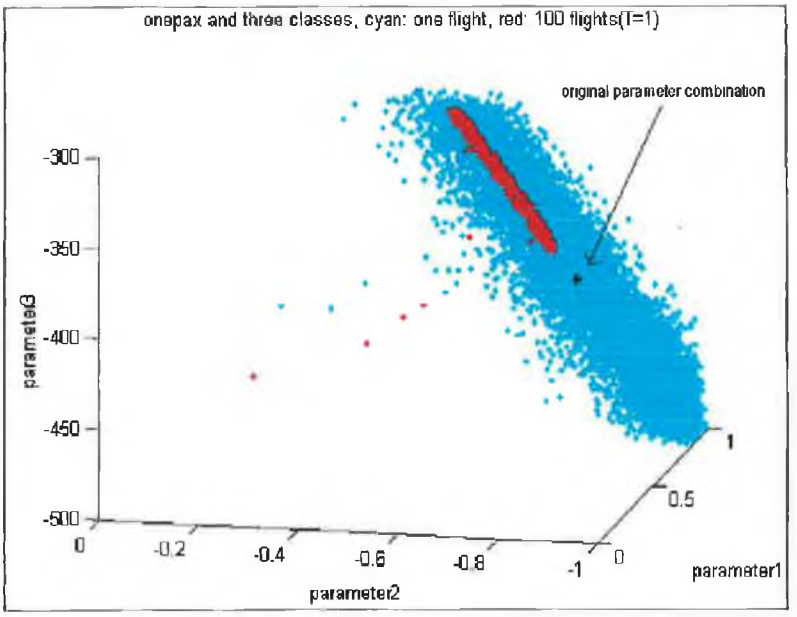


Figure 6.11: One passenger type (type1) with three classes, blue distribution for one flight and red one for the 100 flights

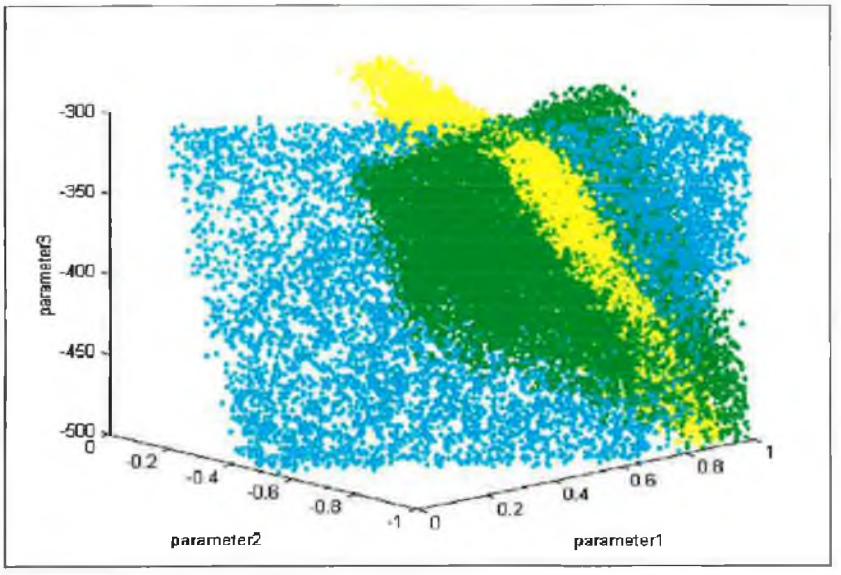


Figure 6.5a (repeated). This shows the planes generated individually by the three classes used to create Figure 6.11 (above).

6.5 Another passenger type and three classes

We change the passenger belief parameters to introduce another passenger type (type 2) and generate new booking data for this passenger type. Then we run the Metropolis Algorithm to produce a different Markov chain representing the distribution of the new passenger type. Table 6.4 contains the parameters used for this run and Figure 6.12 shows the result of the run on data of one flight and 100 flights. Figure 6.13 shows the intersection between the three planes (each for a different class) for this type of passenger. Notice that the area of intersection is the same area covered in Figure 6.12. Figure 6.14 is the same as Figure 6.12 but from different angle.

Correct parameters (data parameters):	$p_{t1} = 0.2$	$p_{t2} = -0.7$	$p_{t3} = -420$
The arbitrary starting point:	$p_{t1} = 0.2$	$p_{t2} = -0.7$	$p_{t3} = -420$
Class1 parameters:	$seats_c = 20$	$\cos t_c = 600$	$comf_c = 800$
Class2 parameters:	$seats_c = 30$	$\cos t_c = -114$	$comf_c = 400$
Class3 parameters:	$seats_c = 100$	$\cos t_c = -400$	$comf_c = 150$

Table: 6.4 Parameters used for another passenger type and three classes

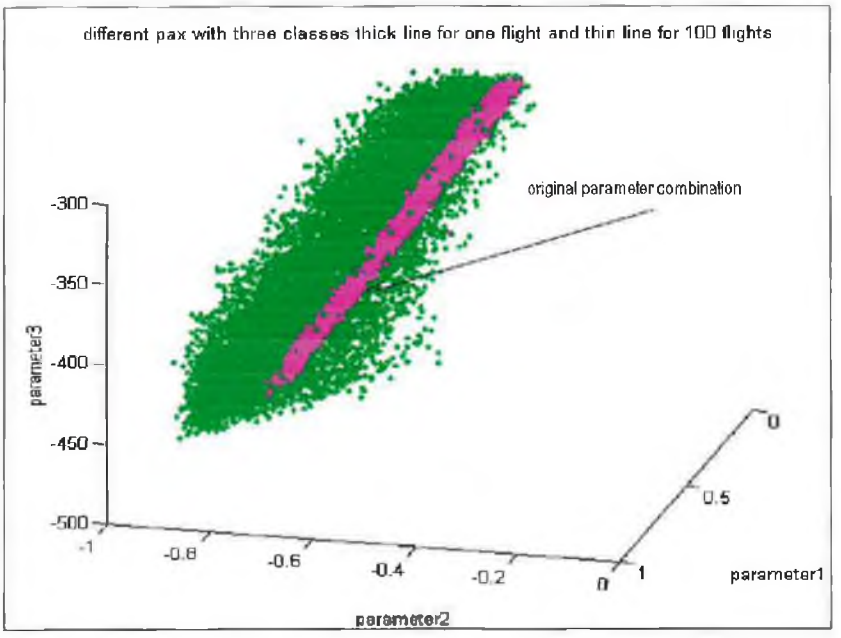


Figure 6.12: Different passenger type (type 2) with three classes for one flight(thick line) and 100 flights (thin line)

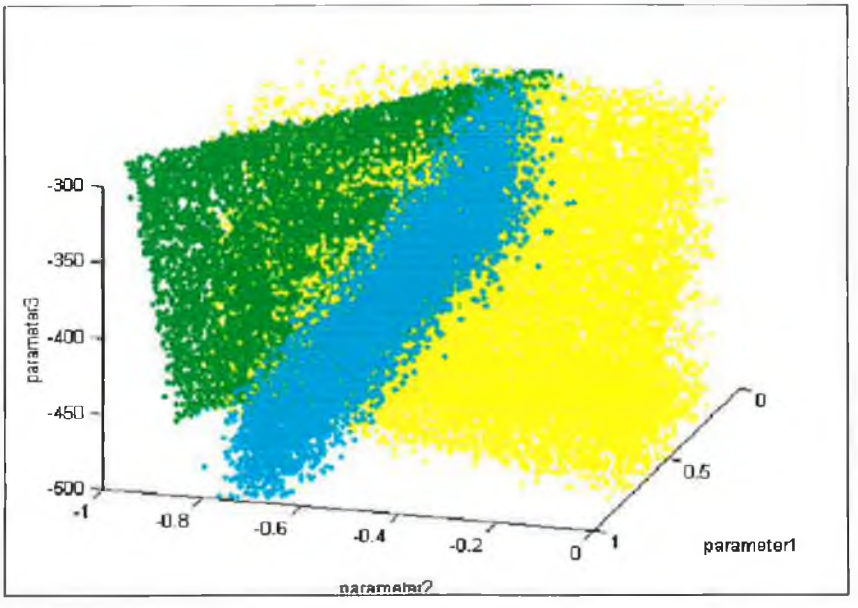


Figure 6.13: Intersection between the runs for a different passenger type (type 2) and three classes separately

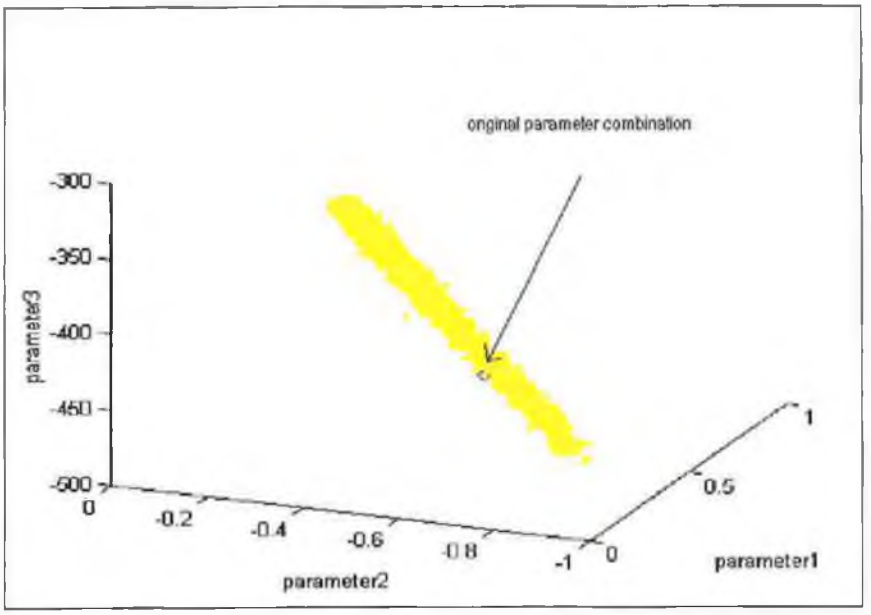


Figure 6.14: Different passenger type (type 2) with three classes-different angle

6.6 Two passenger types and three classes

We create booking data for two passenger types and three classes for 100 flights using the previous two passenger parameters (type 1 and 2) and the previous class attributes, (refer to Tables 6.3 and 6.4). After running the algorithm on this data, the result was very similar to the separate run for each passenger type and the three classes. This indicates that the algorithm is actually recognising different passenger types and keeping only the parameter combinations for each type that have a high likelihood of generating that data. In this run the Algorithm is dealing with six different parameters, three parameters for each type. If the most desired class for each passenger type is different then there is not much of an effect on the different algorithm decisions. However, if the most desired class is the same for both types then the two passenger types are competing for the same class. This will have an effect on the algorithm decision as it has to weigh between the probability of the booking (in case of one booking on this day) being made by the first type or the second type. This effect should not create a major diversion from the correct parameter combination because as the data size increases (100 flights), the pattern of the behaviour of each passenger type can be recognised more accurately. Figure 6.15 shows the previous separate results for each passenger type (same as Figures 6.11 and 6.14), and Figure 6.16 shows the distribution of the passenger parameter combinations for each passenger type from the combined run. Please note Figure 6.16 actually displays six-dimensional data in a three-dimensional space. The output of the Metropolis algorithm consists of combinations of six parameter values: three for each of the two passenger types. The two triplets are actually displayed in the same space, using colour to distinguish the two sets. This implies that each point in the blue distribution actually has a partner in the yellow distribution.

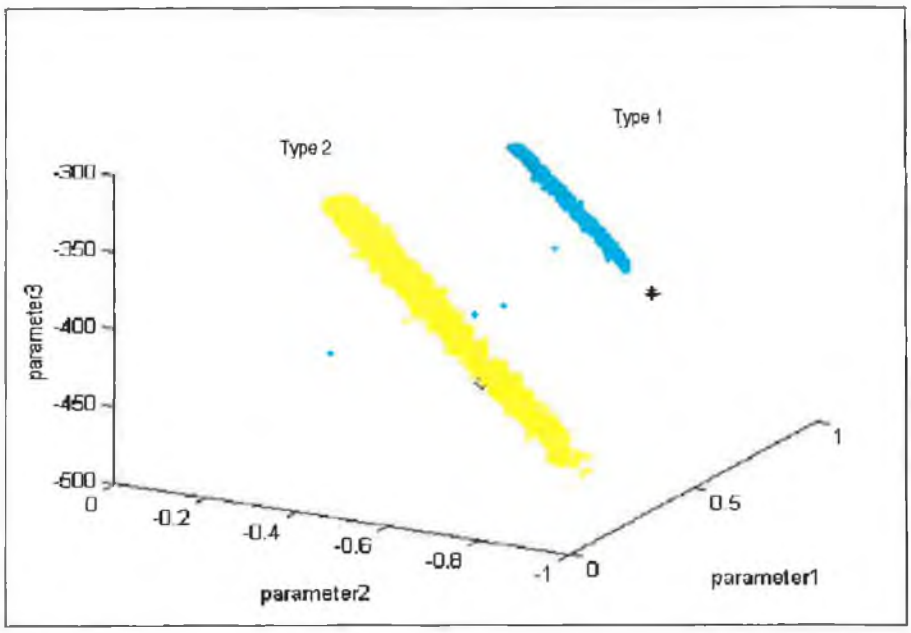


Figure 6.15: Two passenger types (type1 and 2) in separate runs with 3 classes for 100 flights

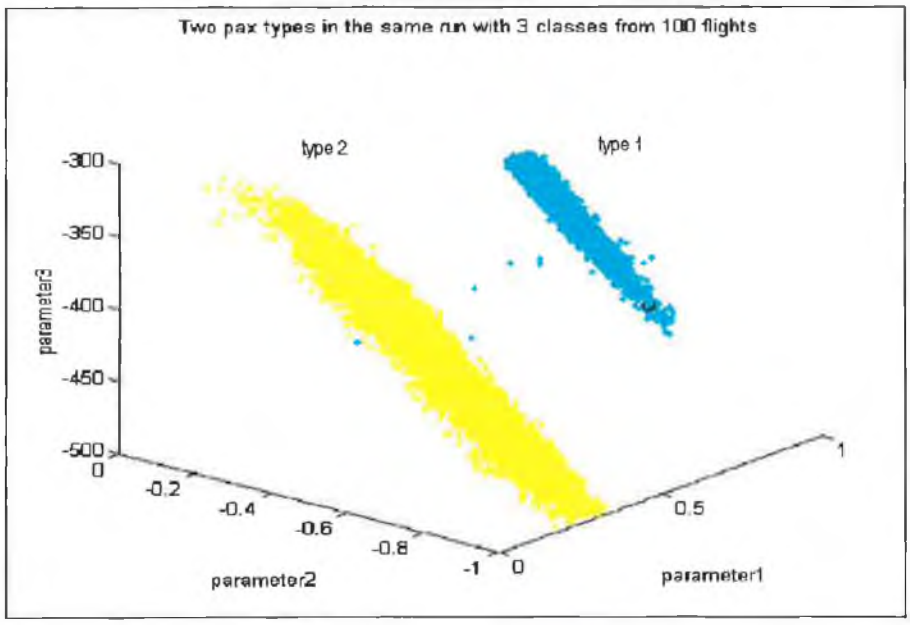


Figure 6.16: Two passenger types with 3 classes from 100 flights. Notice the increase in uncertainty

6.7 Three passenger types and three classes

This time we introduce a third passenger type (type 3) with different belief parameters, as shown in Table 6.5. We first run the algorithm for this passenger type separately and plot the result in Figure 6.17. Then we run the algorithm for the three passenger types and the three classes together and the results are shown in Figure 6.18.

Correct parameters (data parameters):	$p_{t1} = 0.5$	$p_{t2} = -0.3$	$p_{t3} = -300$
The arbitrary starting point:	$p_{t1} = 0.5$	$p_{t2} = -0.3$	$p_{t3} = -300$
Class1 parameters:	$seats_c = 20$	$\cos t_c = 600$	$comf_c = 800$
Class2 parameters:	$seats_c = 30$	$\cos t_c = -114$	$comf_c = 400$
Class3 parameters:	$seats_c = 100$	$\cos t_c = -400$	$comf_c = 150$

Table: 6.5 Parameters used for a third passenger type and three classes

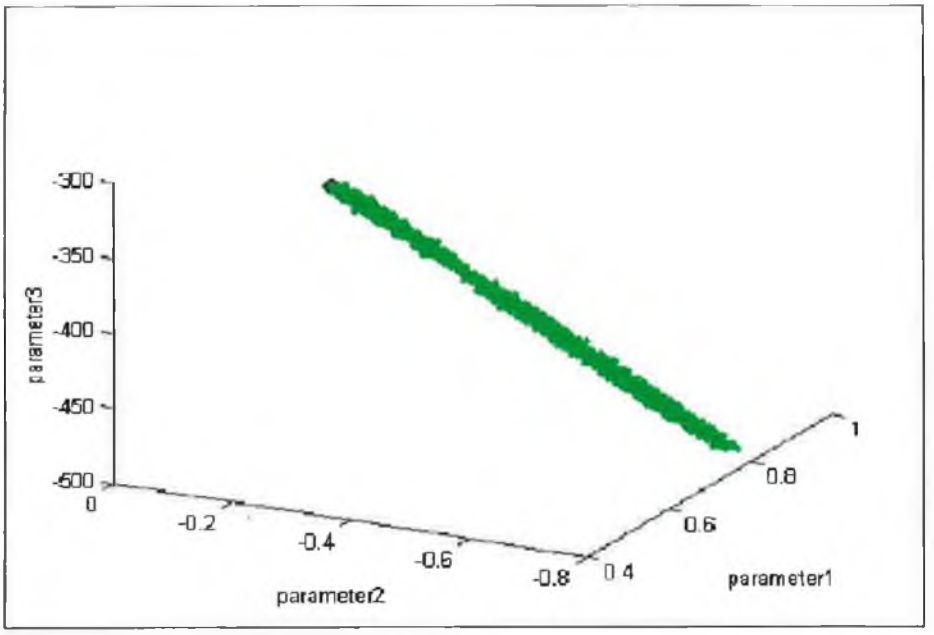


Figure 6.17: A third passenger type (type 3) with 3 classes from 100 flights

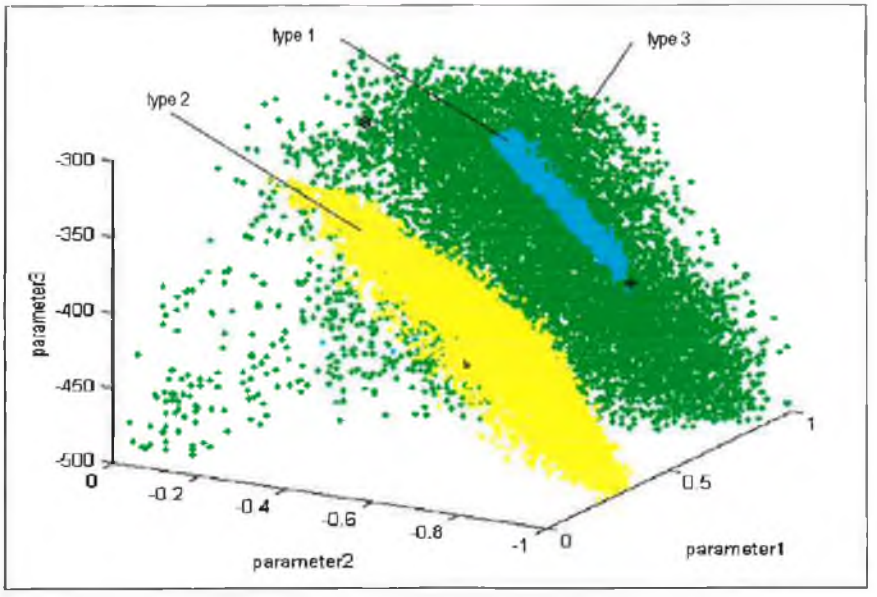


Figure 6.18: Three passenger types with 3 classes from 100 flights

From Figure 6.18 we can see that as we add more passenger types (type 3), the uncertainty increases and the algorithm faces difficulties homing in on the correct set of parameters. This is why the area covered for the third passenger type is still large. We then increased the number of flights to 1000 for the three passenger types, to give the algorithm more data to work with. This decreased the uncertainty in type 2 but type1 and type 3 got trapped in local minima. The result is shown in Figure 6.19. Perhaps the solution to this problem is to use simulated annealing.

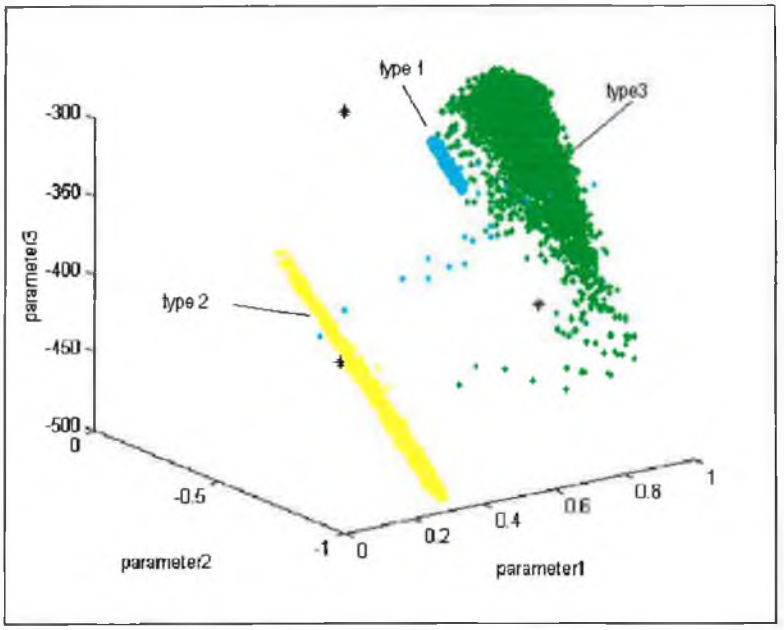


Figure 6.19: Three passenger types with 3 classes from 1000 flights

6.8 Summary

In Chapter 5 we developed a Markov Chain Monte Carlo (MCMC) model using the Metropolis algorithm to fit the model parameters to sample booking data generated by the model developed in Chapter 4. Starting with booking data for one passenger type and one class we were able to construct a Markov Chain that represents the probability distribution of the parameters given this sample data. We then introduced another class to test whether the model can recognise the different behaviour of the passenger towards different classes. The same was repeated with one passenger type and three classes and the algorithm was able to use the additional information given to it to get better results regarding this passenger type. Then we did another experiment with a different passenger type and the previous three classes. We then ran the algorithm on data generated from two passenger types booking on three classes. The last experiment was done on

three passenger types and three classes. In all the previous tests we noticed that although the algorithm was able to distinguish between the different passenger types, as we increased the number of passenger types the level of uncertainty increased resulting in the algorithm facing difficulties homing in on the correct set of parameters. We also tested the benefit of having more data available for the algorithm by generating booking data for 100 flights instead of only one flight, and the resulting distributions from these tests were more concentrated around the correct set of parameters.

CHAPTER 7

Conclusion and Future work

7.1 Conclusion

In this work, we built a Monte Carlo simulation model for the airline booking process based on one flight with different fare classes and different passenger types. We generated sample booking data for the 181 days of the booking process for each class. We were able to produce a variety of booking curves for the booking period by changing the passenger parameters (characteristics) used in the model. We were also able to capture some hidden events that are not available in the real-world data. For example, in the reservation systems, there is no record of a passenger non-arrival event or the decision by an arrived customer not to accept the most desired class available and opt for no purchase at all. We showed that the random utility model with discrete choice set works well for passenger choice behaviour and modelled the number of bookings on each class. Vertical recapture was also considered when the most desired class is full and the passenger decides to book on the second or third desired class that is available.

We modelled the arrival behaviour for different passenger types and represented it with a probability distribution. The passenger arrival parameters for each type were assumed to be known. We modelled the probability of accepting a booking on a class using the Logistic (Logit) function which is one of the random utility models. The Logistic function has an S-shape curve. It performs well in modelling the passenger choice behaviour because its value increases as the desirability of a passenger towards a class increases, i.e. if the desirability towards a class is high then the probability of accepting a booking on this class is also high. We decided to build a simple model of the booking process (one flight, up to three passenger types, and up to three classes). As is always recommended, we began with a simplified version when modelling a complex system. However,

it is also preferable to simulate all the important processes that have a direct and important effect on the decision of the real system.

This sample booking data was then used in a Markov Chain Monte Carlo model in order to estimate the passenger choice parameters used in generating the data. We used the Metropolis algorithm to construct the Markov Chain. In order to compute the log-likelihood required by the algorithm for each iteration and to speed up the execution time, we pre-computed all the necessary probabilities and stored them in tables. When the algorithm reached an equilibrium state after a certain number of iterations, the chain converged and the parameter combinations or the random samples that were chosen did represent the distribution of the parameter combinations given the generated data. These estimated parameters for each passenger type could be used then to classify any new booking data. When we increased the amount of data (100 flights) available to the model we were able to get closer to the most probable set of parameters. However, care had to be taken to avoid getting trapped in local minima.

As we increased the number of passenger types (i.e. increased the number of estimated parameters) and the number of classes, the complexity of the model increased and the algorithm faced difficulties in converging on the right parameter combinations.

We also tested the MCMC model in a parallel processing environment with a cluster of one server and 10 nodes. The 100,000 iterations of the Metropolis algorithm were divided into 10,000 iterations running on each machine (node) creating multiple parallel chains. The results from all machines were combined and produced the same result as the result from all iterations on one PC. The run time was reduced from approximately 216 hours to 30 hours.

7.2 Future work

Further work and research can be conducted in the following areas:

- The MC simulation model can be extended to introduce more classes and other types of passenger, as well as other flights with differing fares and comfort values. We could also add more attributes to the alternative criteria to increase the choice set, such as time of departure, day of the week etc. Also the probability of cancellation can be introduced and incorporated with the overall probability for each day of the booking process in selected classes. This will give more accurate predictions of the actual total number of bookings on departure day. Also the no-show probability can be added in to give an estimate to the number of no-shows that might occur on departure day.
- The parameters used to model the probability of arrival of each passenger type were assumed to be known in our model. We could assume these were unknown and try to estimate them using the MCMC model.
- For each passenger type the belief parameters used in the model were the same for all members of the type. We can introduce variation in these parameters for each type to get more realistic choice behaviour.
- When we introduced a third passenger type we found difficulties in getting a good estimate for the three passenger parameters. The Metropolis Algorithm was not able to distinguish between the most desired class for each passenger without the indirect effect of the other classes. One way to overcome this hurdle is by running the algorithm in a separate time frame for each type. While we know the parameters of arrival for each passenger type indicating the most probable time-frame that this type will arrive to book. So we can run the algorithm for this period which will give us an idea of what the parameter combinations we should keep for this passenger type. For example, tourist passengers arrive and book early during the booking process period, so there is a time-frame when bookings are entirely due to tourist passengers. We could then continue running the algorithm from the day the next type is

expected to arrive until the day the third passenger type start arriving, and keep the accepted parameters and so on.

- For a further test on the algorithm, we could run it using a real booking data and examine whether the estimated parameters of different passenger type can tell us who actually booked this real data.

References

AGIFORS, Airline Group of the International Federation of Operational Research Societies, Reservations & Yield Management Study Group Proceeding, Zurich, 1996 and Montreal, 1997

Anderson, S. E., "Passenger Choice Analysis for Seat Capacity Control: A Pilot Project in Scandinavian Airlines," *Intl. Trans Opl. Res.*, Vol. 5, pp. 47-486, 1998

Ajzen, I. and Fishbein, M., *Understanding Attitudes and Prediction Social Behavior*, Prentice-Hall, Inc., 1980

Belobaba, P., "Air Travel Demand and Airline Seat Inventory Management'" Ph.D. Dissertation, MIT, Cambridge, Mass, 1987a

Belobaba, P., "Airline Yield Management: An Overview of Seat Inventory Control", *Transportation Science*, Vol. 21, pp. 63-73, 1987b

Belobaba, P., "Application of a Probabilistic Decision Model to Airline Seat Inventory Control", *Operations Research*, Vol. 37, No. 2, pp. 183-197, March-April 1989

Belobaba, P., Williamson, E. and Martin, B., "Comparison of Yield Management Methods for Flight Leg and O-D Control" AGIFORS, Reservation and Yield Management Study Group Meeting, Chicago, IL, April 1989

Belobaba, P., "Optimal vs. Heuristic Methods for Nested Seat Allocation", AGIFORS Reservations & Yield Management Study Group, Brussels, May 1992

Belobaba, P. and Wilson, J. L., "Impacts of Yield Management in Competitive Airline Markets", *Journal of Air Transport Management*, Vol. 3, No. 1, pp. 3-9, 1997

Ben_Akiva, M. and Lerman, S. (1985) *Discrete Choice Analysis: Theory and Application to Travel Demand* , (MIT Press, Cambridge, Ma.)

Bierlaire, M., Discrete Choice Models, 1997

http://its.mit.edu/michel/discretChoice/paper.html

Bitran, G. Caldentey, R. abd Mondschein, S. "Coordinating Clearance Markdown Sales of Seasonal Products in Retail Chains", *Operations Research*, Vol. 46, pp. 609-624, 1998

Blackley R., "Comparison Study of Forecasting No-Show Rates", Internal British Airways memo, 1.0, 19 January 1993

Bratly, P, Fox, B. L., Scharge, L. E.,*A Guide to Simulation,* Second Edition, Springer-Verlag, 1987

Brumelle S. L. and McGill J. I., "Airline Seat Allocation with Multiple Nested Fare Classes", *Operations Research*, Vol. 41, No. 1, pp. 127-137, January-February 1993

Curry R. E., "Optimal Airline Seat Allocation with Fare Classes Nested by Origin and Destination", *Transportation Science*, Vol. 24, No. 3, pp. 193-204, August 1990

Dempster, A. P., Laird, N.M. and Rubin, D.B., "Maximum Likelihood From Incomplete Data via the EM Algorithm", *Journal of the Royal Stat. Society*, B, Vol 39, pp. 1-38, 1977

Dror M., Trudeau P. and Ladany S., "Network Models for Seat Allocation on Flights", *Transportation Research-B*, Vol. 22B, No. 4, pp. 239-250, 1988

Engel J. F. Blackwell, R. D. and Miniard, P. W., *Consumer Behaviour,* 6th ed., CBS Publishing Japan Ltd., 1986

Gilks, W.R., Richardson, S. and Spiegelhalter, D.J., "Markov Chain Monte Carlo in Practice", Chapman & Hall, 1996

Gallego G. and van Ryzin G., "A Multiproduct Dynamic Pricing Problem and its Applications to Network Yield Management", *Operations Research,* Vol. 45, No. 1, pp. 24-41, 1997

Glover F., Glover R., Lorenzo J. and McMillan C., "The Passenger Mix Problem in the Scheduled Airline", *Interfaces* Vol. 12, No. 3, pp. 73-79, 1982

Grether, D. and Wilde, L., 'An Analysis of Conjunctive Choice: Theory and Experiments', *Journal of Consumer Research,* Vol. 10 (March), pp. 373-385, 1984

Kheir, N. A., *System Modelling and Computer Simulation*, Marcel Dekker, Inc., 1988

Kuwait Airways Yield Management and Pricing: Operational Review and Action Plan, American Airlines, Decision Technologies, Unpublished document, May 1992

Lee A.O., "Airlines Reservations Forecasting: Probabilistic and Statistical Models of the Booking Process", Ph.D. Dissertation, MIT, Cambridge, Mass, 1990

Mannion, R., "CA and Monte Carlo Models of HIV Infection", M.Sc. Thesis, Dublin City University, Faculty of Computing and Mathematical Sciences, January 2001

McGill, J. I. and van Ryzin, G. J., "Revenue Management Research Overview and Prospects", Trans. Sci., Vol. 33, pp. 233-256, 1999

McGrath, Paul A., "Forecasting Airline Passenger No Show Rates Using Radial-Basis Function Networks", M.Sc. Thesis, University of Brunel, August 1995

Neelamkavil, F., *Computer Simulation and Modelling*, John Wiley & Sons Ltd., 1986

Robert C. P. and Casella G., "Monte Carlo Statistical Methods", Springer-Verlag New York, Inc., 1999

Ripley, B. D. and Sutherland, A. I., "Finding Spiral Structures in Images of Galaxies", Phil. Trans. R. Soc. Lond., A(1990)

Ripley, B. D., "Recognizing Organisms from their Shapes – A case Study in Image Analysis", XVth International Biometrics Conference, Budapest, 1990

Robinson L. W., *"Optimal and Approximate Control Policies for Airline Booking with Sequential Nonmonotonic Fare Classes", Operations Research*, Vol. 43, No. 2, pp. 252-263, March-April 1995

Smith B. C. and Penn C. W., "Analysis of Alternate Origin-Destination Control Strategies", AGIFORS Symposium Proceedings 28, pp. 123-144, 1988

Spiegel, M. R., Schiller J. and Srinivasan, R. A., "SCHAUM'S OUTLINES: Probability and Statistics", Second Edition, McGraw-Hill, 2000

Talluri, K. and van Ryzin, G., "Discrete Choice Model of Yield Management", September 2000

Van Loan, C.F., "Introduction to Scientific Computing, A Matrix-Vector Approach Using MATLAB", Second Edition, Prentice-Hall, 1999

Vinod, B. "Origin and Destination Yield Management", SABRE Decision Technologies, Handbook of Airline Economics, September 1995

Weatherford L. R., Bodily S. E., "A Taxonomy and Research overview of perishable-asset revenue management: Yield Management, Overbooking, and Pricing",*Operations Research*, Vol. 40, No. 5, pp. 831-843, September-October 1992

Williamson E., "Comparison of Optimization Techniques for Origin-Destination Seat Inventory Control", report FTL-R88-2, Flight Transportation Laboratory, MIT, Cambridge, MA 1988

Wollmer R. D., *"An Airline Seat Management Model for a Single Leg Route when Lower Fare Classes Book First", Operations Research*, Vol. 40, No. 1, pp. 26-37, January-February 1992

Wright, P., "Consumer Choice Strategies: Simplifying vs. Optimizing2, *Journal of Marketing Reset*, Vol. XII (February), pp. 60-67, 1975

Yield Management Workshop, Sabre Decision Technology, 1996

Yield Management References from the Internet:

[YM1] http://www.sh-e.com/whatsnew/020100.html

[YM2] http://www.denverpost.com/business/biz1115.htm

[YM3] http://www.dfi.com/txt/pb005.htm

[YM4] http://www.qantas.com.au/company/factfiles/yield.html

[YM5] http://www.siam.org/siamnews/mtc/mtc694.htm

[YM6] http://www.horand-vogel.de/members/moreym.asp

APPENDIX A

A.1 Sample Booking Data

Day	class0	class1	class2
-180	0	0	0
-179	0	0	0
-178	0	0	0
-177	0	0	0
-176	0	0	0
-175	0	0	0
-174	0	0	0
-173	0	0	0
-172	0	0	1
-171	0	0	1
-170	0	0	1
-169	0	0	1
-168	0	0	2
-167	0	0	3
-166	0	0	5
-165	0	0	6
-164	0	0	6
-163	0	0	7
-162	0	0	7
-161	0	0	7
-160	0	0	7
-159	0	0	7
-158	0	0	8
-157	0	0	8
-156	0	0	11
-155	0	0	12
-154	0	0	14
-153	0	0	14
-152	0	0	15
-151	0	0	16
-150	0	0	17
.			
.			
-82	0	0	149
-81	0	0	150
-80	0	0	150
-79	0	0	150
-78	0	1	150
-77	0	1	150
-76	0	2	150
-75	0	2	150

-74	0	2	150
-73	0	3	150
-72	0	4	150
-71	0	4	150
-70	0	4	150
-69	0	4	150
-68	0	5	150
-67	0	6	150
-66	0	6	150
-65	0	6	150
-64	0	6	150
-63	0	8	150
-62	0	8	150
-61	0	8	150
-60	0	8	150
-59	0	8	150
-58	0	10	150
-57	0	10	150

-37	0	78	150
-36	0	86	150
-35	0	91	150
-34	0	97	150
-33	5	100	150
-32	12	100	150
-31	19	100	150
-30	20	100	150
-29	20	100	150
-28	20	100	150
-27	20	100	150
-26	20	100	150
-25	20	100	150
-24	20	100	150
-23	20	100	150
-22	20	100	150
-21	20	100	150
-20	20	100	150
-19	20	100	150
-18	20	100	150
-17	20	100	150
-16	20	100	150
-15	20	100	150
-14	20	100	150
-13	20	100	150
-12	20	100	150

-11	20	100	150
-10	20	100	150
-9	20	100	150
-8	20	100	150
-7	20	100	150
-6	20	100	150
-5	20	100	150
-4	20	100	150
-3	20	100	150
-2	20	100	150
-1	20	100	150
0	20	100	150

A.2 Table for Probability of Arrival (P_arrive)

day: -180											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	1	0	0	0	0	0	0	0	0	0	0
STD	1	0	0	0	0	0	0	0	0	0	0
day: -179											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	0.935298	0.062772	0.001896	3.39E-05	3.98E-07	3.21E-09	1.79E-11	6.88E-14	1.73E-16	2.58E-19	1.73E-22
STD	1	0	0	0	0	0	0	0	0	0	0
day: -178											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	0.874389	0.118161	0.007185	0.000259	6.12E-06	9.93E-08	1.12E-09	8.64E-12	4.38E-14	1.31E-16	1.78E-19
STD	1	0	0	0	0	0	0	0	0	0	0
day: -177											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	0.817073	0.16675	0.015314	0.000833	2.98E-05	7.29E-07	1.24E-08	1.45E-10	1.11E-12	5.02E-15	1.02E-17
STD	1	0	0	0	0	0	0	0	0	0	0
day: -176											

BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	0.763161	0.209085	0.025778	0.001883	9.03E-05	2.97E-06	6.78E-08	1.06E-09	1.09E-11	6.64E-14	1.82E-16
STD	1	0	0	0	0	0	0	0	0	0	0
day: -175											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	0.712471	0.24568	0.038123	0.003506	0.000212	8.75E-06	2.52E-07	4.96E-09	6.41E-11	4.91E-13	1.69E-15
STD	1	0	0	0	0	0	0	0	0	0	0
day: -174											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	0.664833	0.277014	0.05194	0.005771	0.000421	2.10E-05	7.31E-07	1.74E-08	2.72E-10	2.52E-12	1.05E-14
STD	1	0	0	0	0	0	0	0	0	0	0
day: -173											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	0.62008	0.303536	0.066863	0.008728	0.000748	4.39E-05	1.79E-06	5.01E-08	9.20E-10	1.00E-11	4.90E-14
STD	1	0	0	0	0	0	0	0	0	0	0
day: -172											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	0.578057	0.325666	0.082563	0.012404	0.001223	8.27E-05	3.88E-06	1.25E-07	2.64E-09	3.31E-11	1.86E-13

STD	1	0	0	0	0	0	0	0	0	0	0
day: -171											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	0.538615	0.343797	0.09875	0.016809	0.001878	0.000144	7.65E-06	2.79E-07	6.68E-09	9.47E-11	6.05E-13
STD	1	0	0	0	0	0	0	0	0	0	0
day: -45											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	1.00E-10	9.00E-09	3.65E-07	8.75E-06	0.000138	0.001488	0.01116	0.057396	0.19371	0.38742	0.348678
STD	0.019321	0.093491	0.203568	0.262669	0.222421	0.129148	0.052076	0.014399	0.002613	0.000281	1.36E-05
day: -44											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	5.02E-11	4.87E-09	2.13E-07	5.52E-06	9.38E-05	0.001094	0.008852	0.049139	0.179006	0.386425	0.375385
STD	0.01392	0.074239	0.178174	0.253403	0.236509	0.151366	0.067274	0.020503	0.004101	0.000486	2.59E-05
day: -43											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	2.39E-11	2.52E-09	1.19E-07	3.36E-06	6.19E-05	0.000783	0.006877	0.041414	0.163663	0.38328	0.403918
STD	0.009917	0.058137	0.153361	0.239737	0.245937	0.173004	0.084514	0.02831	0.006223	0.000811	4.75E-05

day: -42											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	1.07E-11	1.23E-09	6.39E-08	1.96E-06	3.94E-05	0.000544	0.005216	0.034274	0.147807	0.377729	0.434388
STD	0.006982	0.044887	0.12985	0.222601	0.250426	0.193186	0.103492	0.038018	0.009165	0.001309	8.42E-05
day: -41											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	4.50E-12	5.68E-10	3.23E-08	1.09E-06	2.41E-05	0.000365	0.003846	0.027768	0.131584	0.369499	0.466912
STD	0.004854	0.034155	0.108156	0.20296	0.249941	0.211062	0.123771	0.04977	0.013134	0.002054	0.000145
day: -40											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	1.73E-12	2.43E-10	1.53E-08	5.71E-07	1.40E-05	0.000235	0.002742	0.021936	0.115166	0.358294	0.501612
STD	0.003328	0.025598	0.08861	0.181763	0.244681	0.22586	0.144782	0.06364	0.018358	0.003138	0.000241
day: -39											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	6.05E-13	9.47E-11	6.68E-09	2.79E-07	7.65E-06	0.000144	0.001878	0.016809	0.09875	0.343797	0.538615
STD	0.002248	0.018884	0.071383	0.159897	0.235049	0.236929	0.165851	0.079608	0.025077	0.004681	0.000393
day: -38											
BUS	1	0	0	0	0	0	0	0	0	0	0

TOUR	1.86E-13	3.31E-11	2.64E-09	1.25E-07	3.88E-06	8.27E-05	0.001223	0.012404	0.082563	0.325666	0.578057
STD	0.001495	0.013701	0.056516	0.13815	0.221615	0.243777	0.186218	0.097543	0.03353	0.00683	0.000626
day: -37											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	4.90E-14	1.00E-11	9.20E-10	5.01E-08	1.79E-06	4.39E-05	0.000748	0.008728	0.066863	0.303536	0.62008
STD	0.000977	0.009766	0.043945	0.117188	0.205078	0.246094	0.205078	0.117188	0.043945	0.009766	0.000977
day: -36											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	1.05E-14	2.52E-12	2.72E-10	1.74E-08	7.31E-07	2.10E-05	0.000421	0.005771	0.05194	0.277014	0.664832
STD	0.000626	0.00683	0.03353	0.097543	0.186218	0.243777	0.221615	0.13815	0.056516	0.013701	0.001495
day: -35											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	1.69E-15	4.91E-13	6.41E-11	4.96E-09	2.52E-07	8.75E-06	0.000212	0.003506	0.038123	0.24568	0.712471
STD	0.000393	0.004681	0.025077	0.079608	0.165851	0.236929	0.235049	0.159897	0.071383	0.018884	0.002248
day: -34											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	1.82E-16	6.64E-14	1.09E-11	1.06E-09	6.78E-08	2.97E-06	9.03E-05	0.001883	0.025778	0.209085	0.76316
STD	0.000241	0.003138	0.018358	0.06364	0.144782	0.22586	0.244681	0.181763	0.08861	0.025598	0.003328

day: -33											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	1.02E-17	5.02E-15	1.11E-12	1.45E-10	1.24E-08	7.29E-07	2.98E-05	0.000833	0.015314	0.16675	0.817072
STD	0.000145	0.002054	0.013134	0.04977	0.123771	0.211061	0.249941	0.20296	0.108156	0.034155	0.004854
day: -32											
BUS	1	0	0	0	0	0	0	0	0	0	0
TOUR	1.78E-19	1.31E-16	4.38E-14	8.64E-12	1.12E-09	9.93E-08	6.12E-06	0.000259	0.007185	0.118161	0.874388
STD	8.42E-05	0.001309	0.009165	0.038018	0.103492	0.193186	0.250426	0.222601	0.12985	0.044887	0.006982

day: -10											
BUS	1.43E-09	9.50E-08	2.85E-06	5.07E-05	0.000591	0.00473	0.026279	0.10011	0.250274	0.370777	0.247185
TOUR	0	0	0	0	0	0	0	0	0	0	1
STD	0	0	0	0	0	0	0	0	0	0	1
day: -9											
BUS	2.47E-11	2.60E-09	1.23E-07	3.43E-06	6.31E-05	0.000795	0.006956	0.041738	0.164342	0.383465	0.402638
TOUR	0	0	0	0	0	0	0	0	0	0	1
STD	0	0	0	0	0	0	0	0	0	0	1
day: -8											
BUS	2.41E-14	5.31E-12	5.26E-10	3.08E-08	1.19E-06	3.13E-05	0.000575	0.007225	0.05961	0.291424	0.641134
TOUR	0	0	0	0	0	0	0	0	0	0	1
STD	0	0	0	0	0	0	0	0	0	0	1
day: -7											
BUS	0	0	0	0	0	0	0	0	0	0	1
TOUR	0	0	0	0	0	0	0	0	0	0	1
STD	0	0	0	0	0	0	0	0	0	0	1
day: -6											
BUS	0	0	0	0	0	0	0	0	0	0	1

TOUR	0	0	0	0	0	0	0	0	0	0	1
STD	0	0	0	0	0	0	0	0	0	0	1
day: -5											
BUS	0	0	0	0	0	0	0	0	0	0	1
TOUR	0	0	0	0	0	0	0	0	0	0	1
STD	0	0	0	0	0	0	0	0	0	0	1
day: -4											
BUS	0	0	0	0	0	0	0	0	0	0	1
TOUR	0	0	0	0	0	0	0	0	0	0	1
STD	0	0	0	0	0	0	0	0	0	0	1
day: -3											
BUS	0	0	0	0	0	0	0	0	0	0	1
TOUR	0	0	0	0	0	0	0	0	0	0	1
STD	0	0	0	0	0	0	0	0	0	0	1
day: -2											
BUS	0	0	0	0	0	0	0	0	0	0	1
TOUR	0	0	0	0	0	0	0	0	0	0	1

STD	0	0	0	0	0	0	0	0	0	0	1
day: -1											
BUS	0	0	0	0	0	0	0	0	0	0	1
TOUR	0	0	0	0	0	0	0	0	0	0	1
STD	0	0	0	0	0	0	0	0	0	0	1
day: 0											
BUS	0	0	0	0	0	0	0	0	0	0	1
TOUR	0	0	0	0	0	0	0	0	0	0	1
STD	0	0	0	0	0	0	0	0	0	0	1

A.3 Table for Probability of Acceptance (P_accept)

p_acc=0	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0
p_acc=0.1	1	0.9	0.81	0.729	0.6561	0.59049	0.531441	0.478297	0.430467	0.38742	0.348678
	0	0.1	0.18	0.243	0.2916	0.32805	0.354294	0.372009	0.382638	0.38742	0.38742
	0	0	0.01	0.027	0.0486	0.0729	0.098415	0.124003	0.148803	0.172187	0.19371
	0	0	0	0.001	0.0036	0.0081	0.01458	0.022964	0.033067	0.044641	0.057396
	0	0	0	0	0.0001	0.00045	0.001215	0.002552	0.004593	0.00744	0.01116
	0	0	0	0	0	1.00E-05	5.40E-05	0.00017	0.000408	0.000827	0.001488
	0	0	0	0	0	0	1.00E-06	6.30E-06	2.27E-05	6.12E-05	0.000138
	0	0	0	0	0	0	0	1.00E-07	7.20E-07	2.92E-06	8.75E-06
	0	0	0	0	0	0	0	0	1.00E-08	8.10E-08	3.65E-07
	0	0	0	0	0	0	0	0	0	1.00E-09	9.00E-09
	0	0	0	0	0	0	0	0	0	0	1.00E-10
p_acc=0.2	1	0.8	0.64	0.512	0.4096	0.32768	0.262144	0.209715	0.167772	0.134218	0.107374
	0	0.2	0.32	0.384	0.4096	0.4096	0.393216	0.367002	0.335544	0.30199	0.268435
	0	0	0.04	0.096	0.1536	0.2048	0.24576	0.275251	0.293601	0.30199	0.30199
	0	0	0	0.008	0.0256	0.0512	0.08192	0.114688	0.146801	0.176161	0.201327
	0	0	0	0	0.0016	0.0064	0.01536	0.028672	0.045875	0.06606	0.08808

	0	0	0	0	0	0.00032	0.001536	0.004301	0.009175	0.016515	0.026424
	0	0	0	0	0	0	6.40E-05	0.000358	0.001147	0.002753	0.005505
	0	0	0	0	0	0	0	1.28E-05	8.19E-05	0.000295	0.000786
	0	0	0	0	0	0	0	0	2.56E-06	1.84E-05	7.37E-05
	0	0	0	0	0	0	0	0	0	5.12E-07	4.10E-06
	0	0	0	0	0	0	0	0	0	0	1.02E-07
p_acc=0.3	1	0.7	0.49	0.343	0.2401	0.16807	0.117649	0.082354	0.057648	0.040354	0.028248
	0	0.3	0.42	0.441	0.4116	0.36015	0.302526	0.247063	0.19765	0.15565	0.121061
	0	0	0.09	0.189	0.2646	0.3087	0.324135	0.317652	0.296475	0.266828	0.233474
	0	0	0	0.027	0.0756	0.1323	0.18522	0.226895	0.254122	0.266828	0.266828
	0	0	0	0	0.0081	0.02835	0.059535	0.097241	0.136137	0.171532	0.200121
	0	0	0	0	0	0.00243	0.010206	0.025005	0.046675	0.073514	0.102919
	0	0	0	0	0	0	0.000729	0.003572	0.010002	0.021004	0.036757
	0	0	0	0	0	0	0	0.000219	0.001225	0.003858	0.009002
	0	0	0	0	0	0	0	0	6.56E-05	0.000413	0.001447
	0	0	0	0	0	0	0	0	0	1.97E-05	0.000138
	0	0	0	0	0	0	0	0	0	0	5.90E-06
p_acc=0.4	1	0.6	0.36	0.216	0.1296	0.07776	0.046656	0.027994	0.016796	0.010078	0.006047
	0	0.4	0.48	0.432	0.3456	0.2592	0.186624	0.130637	0.08958	0.060466	0.040311
	0	0	0.16	0.288	0.3456	0.3456	0.31104	0.261274	0.209019	0.161243	0.120932
	0	0	0	0.064	0.1536	0.2304	0.27648	0.290304	0.278692	0.250823	0.214991
	0	0	0	0	0.0256	0.0768	0.13824	0.193536	0.232243	0.250823	0.250823
	0	0	0	0	0	0.01024	0.036864	0.077414	0.123863	0.167215	0.200658
	0	0	0	0	0	0	0.004096	0.017203	0.041288	0.074318	0.111477
	0	0	0	0	0	0	0	0.001638	0.007864	0.021234	0.042467
	0	0	0	0	0	0	0	0	0.000655	0.003539	0.010617
	0	0	0	0	0	0	0	0	0	0.000262	0.001573
	0	0	0	0	0	0	0	0	0	0	0.000105

p_acc=0.5	1	0.5	0.25	0.125	0.0625	0.03125	0.015625	0.007813	0.003906	0.001953	0.000977
	0	0.5	0.5	0.375	0.25	0.15625	0.09375	0.054688	0.03125	0.017578	0.009766
	0	0	0.25	0.375	0.375	0.3125	0.234375	0.164063	0.109375	0.070313	0.043945
	0	0	0	0.125	0.25	0.3125	0.3125	0.273438	0.21875	0.164063	0.117188
	0	0	0	0	0.0625	0.15625	0.234375	0.273438	0.273438	0.246094	0.205078
	0	0	0	0	0	0.03125	0.09375	0.164063	0.21875	0.246094	0.246094
	0	0	0	0	0	0	0.015625	0.054688	0.109375	0.164063	0.205078
	0	0	0	0	0	0	0	0.007813	0.03125	0.070313	0.117188
	0	0	0	0	0	0	0	0	0.003906	0.017578	0.043945
	0	0	0	0	0	0	0	0	0	0.001953	0.009766
	0	0	0	0	0	0	0	0	0	0	0.000977
p_acc=0.6	1	0.4	0.16	0.064	0.0256	0.01024	0.004096	0.001638	0.000655	0.000262	0.000105
	0	0.6	0.48	0.288	0.1536	0.0768	0.036864	0.017203	0.007864	0.003539	0.001573
	0	0	0.36	0.432	0.3456	0.2304	0.13824	0.077414	0.041288	0.021234	0.010617
	0	0	0	0.216	0.3456	0.3456	0.27648	0.193536	0.123863	0.074318	0.042467
	0	0	0	0	0.1296	0.2592	0.31104	0.290304	0.232243	0.167215	0.111477
	0	0	0	0	0	0.07776	0.186624	0.261274	0.278692	0.250823	0.200658
	0	0	0	0	0	0	0.046656	0.130637	0.209019	0.250823	0.250823
	0	0	0	0	0	0	0	0.027994	0.08958	0.161243	0.214991
	0	0	0	0	0	0	0	0	0.016796	0.060466	0.120932
	0	0	0	0	0	0	0	0	0	0.010078	0.040311
	0	0	0	0	0	0	0	0	0	0	0.006047
p_acc=0.7	1	0.3	0.09	0.027	0.0081	0.00243	0.000729	0.000219	6.56E-05	1.97E-05	5.90E-06
	0	0.7	0.42	0.189	0.0756	0.02835	0.010206	0.003572	0.001225	0.000413	0.000138
	0	0	0.49	0.441	0.2646	0.1323	0.059535	0.025005	0.010002	0.003858	0.001447
	0	0	0	0.343	0.4116	0.3087	0.18522	0.097241	0.046675	0.021004	0.009002
	0	0	0	0	0.2401	0.36015	0.324135	0.226895	0.136137	0.073514	0.036757
	0	0	0	0	0	0.16807	0.302526	0.317652	0.254122	0.171532	0.102919

	0	0	0	0	0	0	0.117649	0.247063	0.296475	0.266828	0.200121
	0	0	0	0	0	0	0	0.082354	0.19765	0.266828	0.266828
	0	0	0	0	0	0	0	0	0.057648	0.15565	0.233474
	0	0	0	0	0	0	0	0	0	0.040354	0.121061
	0	0	0	0	0	0	0	0	0	0	0.028248
p_acc=0.8	1	0.2	0.04	0.008	0.0016	0.00032	6.40E-05	1.28E-05	2.56E-06	5.12E-07	1.02E-07
	0	0.8	0.32	0.096	0.0256	0.0064	0.001536	0.000358	8.19E-05	1.84E-05	4.10E-06
	0	0	0.64	0.384	0.1536	0.0512	0.01536	0.004301	0.001147	0.000295	7.37E-05
	0	0	0	0.512	0.4096	0.2048	0.08192	0.028672	0.009175	0.002753	0.000786
	0	0	0	0	0.4096	0.4096	0.24576	0.114688	0.045875	0.016515	0.005505
	0	0	0	0	0	0.32768	0.393216	0.275251	0.146801	0.06606	0.026424
	0	0	0	0	0	0	0.262144	0.367002	0.293601	0.176161	0.08808
	0	0	0	0	0	0	0	0.209715	0.335544	0.30199	0.201327
	0	0	0	0	0	0	0	0	0.167772	0.30199	0.30199
	0	0	0	0	0	0	0	0	0	0.134218	0.268435
	0	0	0	0	0	0	0	0	0	0	0.107374
p_acc=0.9	1	0.1	0.01	0.001	0.0001	1.00E-05	1.00E-06	1.00E-07	1.00E-08	1.00E-09	1.00E-10
	0	0.9	0.18	0.027	0.0036	0.00045	5.40E-05	6.30E-06	7.20E-07	8.10E-08	9.00E-09
	0	0	0.81	0.243	0.0486	0.0081	0.001215	0.00017	2.27E-05	2.92E-06	3.65E-07
	0	0	0	0.729	0.2916	0.0729	0.01458	0.002552	0.000408	6.12E-05	8.75E-06
	0	0	0	0	0.6561	0.32805	0.098415	0.022964	0.004593	0.000827	0.000138
	0	0	0	0	0	0.59049	0.354294	0.124003	0.033067	0.00744	0.001488
	0	0	0	0	0	0	0.531441	0.372009	0.148803	0.044641	0.01116
	0	0	0	0	0	0	0	0.478297	0.382638	0.172187	0.057396
	0	0	0	0	0	0	0	0	0.430467	0.38742	0.19371
	0	0	0	0	0	0	0	0	0	0.38742	0.38742
	0	0	0	0	0	0	0	0	0	0	0.348678

p_acc=1	1	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0
	0	0	1	0	0	0	0	0	0	0	0
	0	0	0	1	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	0	0
	0	0	0	0	0	0	1	0	0	0	0
	0	0	0	0	0	0	0	1	0	0	0
	0	0	0	0	0	0	0	0	1	0	0
	0	0	0	0	0	0	0	0	0	1	0
	0	0	0	0	0	0	0	0	0	0	1

A.4 Table for Over all Probability of Bookings

day: -179

[0,0,0] 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0

[0,0,0.1] 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0

[0,0,0.2] 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0

[0,0,0.3] 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0

[0,0,0.4] 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0

[0,0,0.5] 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0

[0,0,0.6] 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0

[0,0,0.7] 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0

[0,0,0.8] 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0

[0,0,0.9] 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0

[0,0,1] 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0

0 0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 0 0 0 0
0 0 0 0 0 0

[0,0.1,0] 0.993353 0.00662677 1.98936e-005 3.53899e-008 4.13158e-011 3.30748e-014 1.83871e-017 7.00929e-021
1.75349e-024 2.5995e-028 1.73415e-032 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[0,0.1,0.1] 0.993353 0.00662677 1.98936e-005 3.53899e-008 4.13158e-011 3.30748e-014 1.83871e-017 7.00929e-021
1.75349e-024 2.5995e-028 1.73415e-032 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[0,0.1,0.2] 0.993353 0.00662677 1.98936e-005 3.53899e-008 4.13158e-011 3.30748e-014 1.83871e-017 7.00929e-021
1.75349e-024 2.5995e-028 1.73415e-032 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[0,0.1,0.3] 0.993353 0.00662677 1.98936e-005 3.53899e-008 4.13158e-011 3.30748e-014 1.83871e-017 7.00929e-021
1.75349e-024 2.5995e-028 1.73415e-032 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[0,0.1,0.4] 0.993353 0.00662677 1.98936e-005 3.53899e-008 4.13158e-011 3.30748e-014 1.83871e-017 7.00929e-021
1.75349e-024 2.5995e-028 1.73415e-032 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[0,0.1,0.5] 0.993353 0.00662677 1.98936e-005 3.53899e-008 4.13158e-011 3.30748e-014 1.83871e-017 7.00929e-021
1.75349e-024 2.5995e-028 1.73415e-032 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[0,0.1,0.6] 0.993353 0.00662677 1.98936e-005 3.53899e-008 4.13158e-011 3.30748e-014 1.83871e-017 7.00929e-021
1.75349e-024 2.5995e-028 1.73415e-032 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[0,0.1,0.7] 0.993353 0.00662677 1.98936e-005 3.53899e-008 4.13158e-011 3.30748e-014 1.83871e-017 7.00929e-021
1.75349e-024 2.5995e-028 1.73415e-032 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[0,0.1,0.8] 0.993353 0.00662677 1.98936e-005 3.53899e-008 4.13158e-011 3.30748e-014 1.83871e-017 7.00929e-021
1.75349e-024 2.5995e-028 1.73415e-032 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[0,0.1,0.9] 0.993353 0.00662677 1.98936e-005 3.53899e-008 4.13158e-011 3.30748e-014 1.83871e-017 7.00929e-021
1.75349e-024 2.5995e-028 1.73415e-032 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[0,0.1,1] 0.993353 0.00662677 1.98936e-005 3.53899e-008 4.13158e-011 3.30748e-014 1.83871e-017 7.00929e-021
1.75349e-024 2.5995e-028 1.73415e-032 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[0,0.2,0] 0.986746 0.0131742 7.91507e-005 2.818e-007 6.58411e-010 1.05487e-012 1.17364e-015 8.95394e-019
4.48295e-022 1.33005e-025 1.77577e-029 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0

[0,0.2,0.1]	0.986746	0.0131742	7.91507e-005	2.818e-007							
4.48295e-022	1.33005e-025	1.77577e-029	0	0							
0	0	0	0	0	0	0					
[0,0.2,0.2]	0.986746	0.0131742	7.91507e-005	2.818e-007							
4.48295e-022	1.33005e-025	1.77577e-029	0	0							
0	0	0	0	0	0	0					
[0,0.2,0.3]	0.986746	0.0131742	7.91507e-005	2.818e-007							
4.48295e-022	1.33005e-025	1.77577e-029	0	0							
0	0	0	0	0	0	0					
.											
.											
.											
[1,0.8,0.3]	0.947928	0.0508273	0.0012264	1.75356e-005							
2.91446e-017	3.4727e-020	1.86203e-023	0	0							
0	0	0	0	0	0	0					
[1,0.8,0.4]	0.947928	0.0508273	0.0012264	1.75356e-005							
2.91446e-017	3.4727e-020	1.86203e-023	0	0							
0	0	0	0	0	0	0					
[1,0.8,0.5]	0.947928	0.0508273	0.0012264	1.75356e-005							
2.91446e-017	3.4727e-020	1.86203e-023	0	0							
0	0	0	0	0	0	0					
[1,0.8,0.6]	0.947928	0.0508273	0.0012264	1.75356e-005							
2.91446e-017	3.4727e-020	1.86203e-023	0	0							
0	0	0	0	0	0	0					
[1,0.8,0.7]	0.947928	0.0508273	0.0012264	1.75356e-005							
2.91446e-017	3.4727e-020	1.86203e-023	0	0							
0	0	0	0	0	0	0					
[1,0.8,0.8]	0.947928	0.0508273	0.0012264	1.75356e-005							
2.91446e-017	3.4727e-020	1.86203e-023	0	0							
0	0	0	0	0	0	0					
[1,0.8,0.9]	0.947928	0.0508273	0.0012264	1.75356e-005							
2.91446e-017	3.4727e-020	1.86203e-023	0	0							
0	0	0	0	0	0	0					

6.58411e-010 1.05487e-012 1.17364e-015 8.95394e-019
0 0 0 0 0 0 0
0 0 0 0
6.58411e-010 1.05487e-012 1.17364e-015 8.95394e-019
0 0 0 0 0 0 0
0 0 0 0
6.58411e-010 1.05487e-012 1.17364e-015 8.95394e-019
0 0 0 0 0 0 0
0 0 0 0

1.64543e-007 1.05872e-009 4.73067e-012 1.44946e-014
0 0 0 0 0 0 0
0 0 0 0
1.64543e-007 1.05872e-009 4.73067e-012 1.44946e-014
0 0 0 0 0 0 0
0 0 0 0
1.64543e-007 1.05872e-009 4.73067e-012 1.44946e-014
0 0 0 0 0 0 0
0 0 0 0
1.64543e-007 1.05872e-009 4.73067e-012 1.44946e-014
0 0 0 0 0 0 0
0 0 0 0
1.64543e-007 1.05872e-009 4.73067e-012 1.44946e-014
0 0 0 0 0 0 0
0 0 0 0
1.64543e-007 1.05872e-009 4.73067e-012 1.44946e-014
0 0 0 0 0 0 0
0 0 0 0
1.64543e-007 1.05872e-009 4.73067e-012 1.44946e-014
0 0 0 0 0 0 0
0 0 0 0

[1,0.8,1] 0.947928 0.0508273 0.0012264 1.75356e-005 1.64543e-007 1.05872e-009 4.73067e-012 1.44946e-014
2.91446e-017 3.4727e-020 1.86203e-023 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[1,0.9,0] 0.941594 0.0568367 0.00154385 2.48507e-005 2.62508e-007 1.90147e-009 9.56473e-012 3.29913e-014
7.46785e-017 1.00172e-019 6.0466e-023 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[1,0.9,0.1] 0.941594 0.0568367 0.00154385 2.48507e-005 2.62508e-007 1.90147e-009 9.56473e-012 3.29913e-014
7.46785e-017 1.00172e-019 6.0466e-023 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[1,0.9,0.2] 0.941594 0.0568367 0.00154385 2.48507e-005 2.62508e-007 1.90147e-009 9.56473e-012 3.29913e-014
7.46785e-017 1.00172e-019 6.0466e-023 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[1,0.9,0.3] 0.941594 0.0568367 0.00154385 2.48507e-005 2.62508e-007 1.90147e-009 9.56473e-012 3.29913e-014
7.46785e-017 1.00172e-019 6.0466e-023 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[1,0.9,0.4] 0.941594 0.0568367 0.00154385 2.48507e-005 2.62508e-007 1.90147e-009 9.56473e-012 3.29913e-014
7.46785e-017 1.00172e-019 6.0466e-023 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[1,0.9,0.5] 0.941594 0.0568367 0.00154385 2.48507e-005 2.62508e-007 1.90147e-009 9.56473e-012 3.29913e-014
7.46785e-017 1.00172e-019 6.0466e-023 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[1,0.9,0.6] 0.941594 0.0568367 0.00154385 2.48507e-005 2.62508e-007 1.90147e-009 9.56473e-012 3.29913e-014
7.46785e-017 1.00172e-019 6.0466e-023 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[1,0.9,0.7] 0.941594 0.0568367 0.00154385 2.48507e-005 2.62508e-007 1.90147e-009 9.56473e-012 3.29913e-014
7.46785e-017 1.00172e-019 6.0466e-023 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[1,0.9,0.8] 0.941594 0.0568367 0.00154385 2.48507e-005 2.62508e-007 1.90147e-009 9.56473e-012 3.29913e-014
7.46785e-017 1.00172e-019 6.0466e-023 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[1,0.9,0.9] 0.941594 0.0568367 0.00154385 2.48507e-005 2.62508e-007 1.90147e-009 9.56473e-012 3.29913e-014
7.46785e-017 1.00172e-019 6.0466e-023 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
[1,0.9,1] 0.941594 0.0568367 0.00154385 2.48507e-005 2.62508e-007 1.90147e-009 9.56473e-012 3.29913e-014
7.46785e-017 1.00172e-019 6.0466e-023 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0

[1,1,0] 0.935298 0.0627717 0.00189579 3.39291e-005 3.98496e-007 3.20937e-009 1.79495e-011 6.88379e-014
1.7325e-016 2.58389e-019 1.73415e-022 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
[1,1,0.1] 0.935298 0.0627717 0.00189579 3.39291e-005 3.98496e-007 3.20937e-009 1.79495e-011 6.88379e-014
1.7325e-016 2.58389e-019 1.73415e-022 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
[1,1,0.2] 0.935298 0.0627717 0.00189579 3.39291e-005 3.98496e-007 3.20937e-009 1.79495e-011 6.88379e-014
1.7325e-016 2.58389e-019 1.73415e-022 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
[1,1,0.3] 0.935298 0.0627717 0.00189579 3.39291e-005 3.98496e-007 3.20937e-009 1.79495e-011 6.88379e-014
1.7325e-016 2.58389e-019 1.73415e-022 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
[1,1,0.4] 0.935298 0.0627717 0.00189579 3.39291e-005 3.98496e-007 3.20937e-009 1.79495e-011 6.88379e-014
1.7325e-016 2.58389e-019 1.73415e-022 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
[1,1,0.5] 0.935298 0.0627717 0.00189579 3.39291e-005 3.98496e-007 3.20937e-009 1.79495e-011 6.88379e-014
1.7325e-016 2.58389e-019 1.73415e-022 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
[1,1,0.6] 0.935298 0.0627717 0.00189579 3.39291e-005 3.98496e-007 3.20937e-009 1.79495e-011 6.88379e-014
1.7325e-016 2.58389e-019 1.73415e-022 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
[1,1,0.7] 0.935298 0.0627717 0.00189579 3.39291e-005 3.98496e-007 3.20937e-009 1.79495e-011 6.88379e-014
1.7325e-016 2.58389e-019 1.73415e-022 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
[1,1,0.8] 0.935298 0.0627717 0.00189579 3.39291e-005 3.98496e-007 3.20937e-009 1.79495e-011 6.88379e-014
1.7325e-016 2.58389e-019 1.73415e-022 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
[1,1,0.9] 0.935298 0.0627717 0.00189579 3.39291e-005 3.98496e-007 3.20937e-009 1.79495e-011 6.88379e-014
1.7325e-016 2.58389e-019 1.73415e-022 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
[1,1,1] 0.935298 0.0627717 0.00189579 3.39291e-005 3.98496e-007 3.20937e-009 1.79495e-011 6.88379e-014
1.7325e-016 2.58389e-019 1.73415e-022 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

A.5 Reading from the binary file P_allb

* *P_acc* for each passenger type in the range [0-1]

  Max_index = maximum of P_acc = 11

  If NOTYPES = 3 ➔ *Index* = [Max_index, Max_index, Max_index]

  Index will be in the range [0, 0, 0] to [10, 10, 10]

* Number of bookings in the range of [0 - Max_arr*NOTYPES]

  If maximum arrival for each type = 10 ➔ Max_arr = 30

  Number of total booking in the range [0-30]

  Max_booking = Max_arr * NOTYPES+1 ➔ 31

* Each *P_tot* value is stored in 4 bytes

* *Index* = [ind1, ind2, ind3]

(1) To point to the start of any day *d*

$$\text{Pointer}_1 = d * Max\_index^{NOTYPES} * Max\_booking * 4$$

For three types of passenger, number of bytes for each day is: $Max\_index^{NOTYPES} * Max\_booking * 4 = 1331 * 31 * 4 = 165044$

e.g. for $d = 0$,

$\text{Pointer}_1 = 0 * 1\,65044 = 0$

(2) To move to the starting location of any day (i.e. *bookings* = 0), given *Index* = [ind1, ind2, ind3]

$Pointer_2 = Pointer_1 + (ind1*Max\_index*Max\_index + ind2*Max\_index + ind3)*(Max\_booking*4)$

e.g. for *d* = 0 and *Index* = [9, 0, 0]

[ ] = 0 + (9 * 11 * 11 + 0 * 11 + 0) * (31 * 4)

= 135036

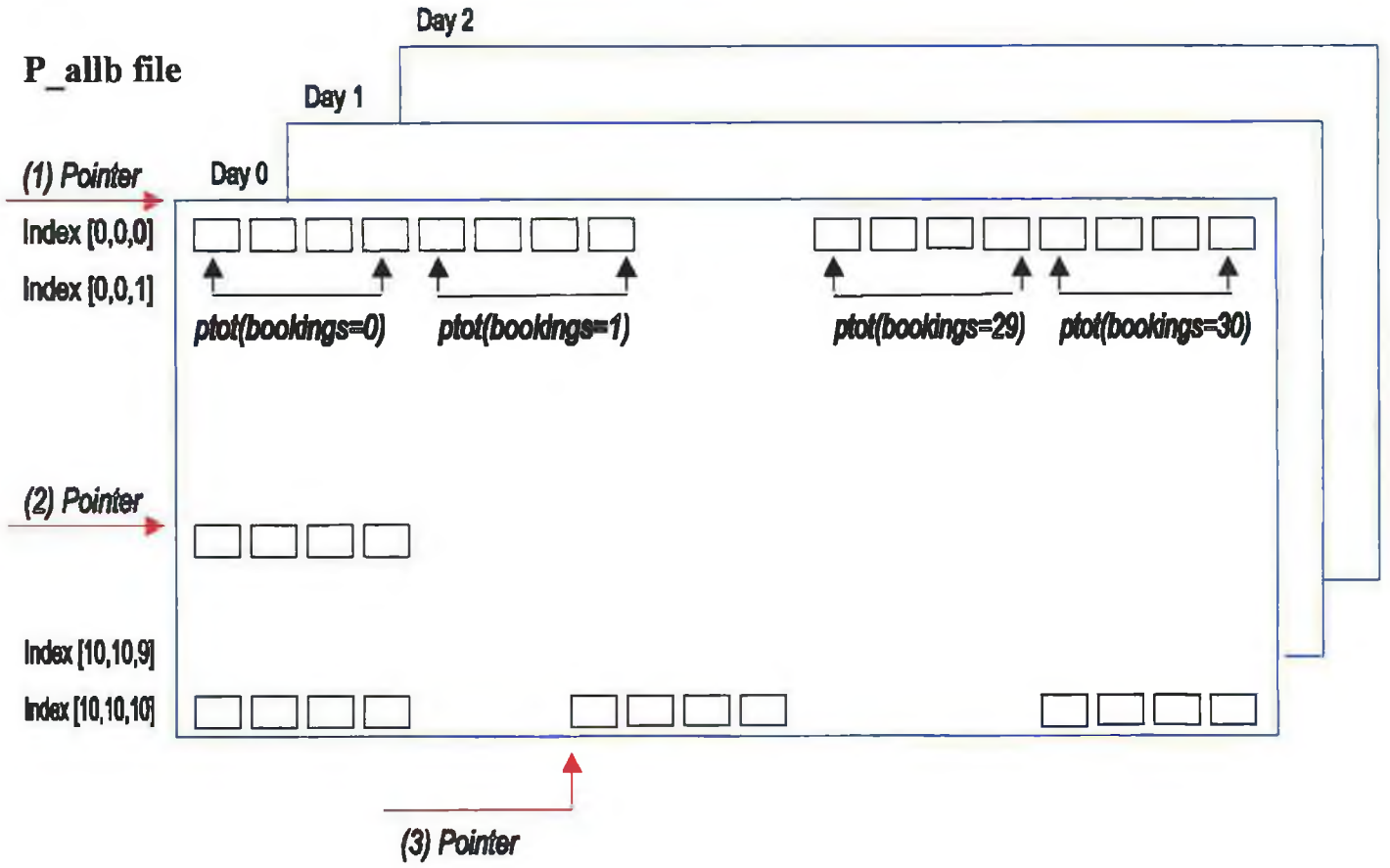
(3) To move to any location within a day and *Index*, given the number of booking, *bookings*

[ ] = [ ] + (*bookings* * 4)

e.g. for *d* = 0 and *Index* = [9, 0, 0] and *bookings* = 3

[ ] = 135036 + (3 * 4)

= 135048


P_allb file
Day 2
Day 1
Day 0
(1) Pointer
Index [0,0,0]
Index [0,0,1]
ptot(bookings=0)
ptot(bookings=1)
ptot(bookings=29)
ptot(bookings=30)
(2) Pointer
Index [10,10,9]
Index [10,10,10]
(3) Pointer

APPENDIX B

B.1 *Sim_data* C program for generating sample data

```
/*sim_data.cpp creates a sample booking data for 181 days of the
booking process with 3 types of passenger and 3 classes*/

#include <iostream.h>
#include <iomanip.h>
#include <fstream.h>
#include <stdlib.h>

#define NOCLASSES 3//number of classes on this flight
#define NOTYPES 3  //number of passenger types
#define Max_arr 10 //Max no. of arrival each type in 10 intervals
#define FLIGHT 100 //number of flights


/*functions prototype*/
void run_simulation(int);
void intialise_variables(int);
float get_desr_class(float [NOTYPES][4], int);
void sort_classes(int [NOTYPES][NOCLASSES], int);
void book_seat(int );
int willing_to_accept(int, int);
float get_random_number(float, float);
float p_arrive(float [NOTYPES][4], int, int);
double prob_of_accept(float);
void sell_seat(int, int);
void vertical_r(int, int);
void openFiles(void);
void closeFiles(void);

/* Output files for analysis */
ofstream outdataFile("threepax3cls100N_data.dat", ios::out);
ifstream inpaxFile("pax_belief_parm.txt", ios::in);
//pax belief parm
ifstream inarrFile("pax_arrival_parm.txt", ios::in);
//pax arrival parm
ifstream inclsFile("pax_3class3_parm.txt", ios::in);//class parm

char names[NOTYPES][15] = {"BUS", "TOUR", "STD"};//pax types*/
int class_seats[NOCLASSES], /* number of seats */
      class_comf[NOCLASSES],
      class_fare[NOCLASSES];
```

```
int pax_seats_booked[NOTYPES][NOCLASSES],
      tot_booked[NOCLASSES],
      seats_cancelled[NOCLASSES],
      pax_ver_rec[NOTYPES][NOCLASSES],
      pax_hor_rec[NOTYPES][NOCLASSES],
      pax_hor_rec_full[NOTYPES];

/*passenger a, b, c parameters for probability of arrival*/
float pax_arrival_parm[NOTYPES][4];
/*passenger attributes for comfort and cost to calculate
desirability*/
float pax_belief_parm[NOTYPES][4];
float desr[NOTYPES][NOCLASSES];
int class_label[NOTYPES][NOCLASSES];
/*array required for calculating data bookings*/
int data_bookings[FLIGHT][NOCLASSES][181];
double p_acc[NOTYPES][NOCLASSES];

int main()
{
  srand(time(0));
  openFiles();

  /*calculate the desiarability of each passenger type to each
class*/
  for(int i=0; i<NOTYPES; i++)
  {
        get_desr_class(pax_belief_parm, i);

        for(int j=0; j<NOCLASSES; j++)
        {
             class_label[i][j] = j;
        }

        sort_classes(class_label, i);
  }

  /*run simulation to get the 100 different data bookings*/
  for (int flt = 0; flt < FLIGHT; flt++)
  {
     run_simulation(flt);
  }
  return 0;
}
```

```
void run_simulation(int flt)
{
  intialise_variables(flt);

  for (int day = -180; day <= 0; day++)
  {
   /*for every hour in 10 hours day*/
   for (int t1 = 1; t1 <= 10; t1++)
   {
    for (int type_pax = 0; type_pax <= NOTYPES-1; type_pax++)
    {
      /*calculate the probability of a pass. arriving in time t1*/
      if (p_arrive(pax_arrival_parm,
      day,type_pax)>get_random_number(1,0))
      {
        /*check if all classes are full*/
        j = 0;
        while (class_seats[class_label[type_pax][j]] ==
        tot_booked[class_label[type_pax][j]] && j <= NOCLASSES-
        1)j++;

        if(j>=NOCLASSES)
        {
          ++pax_hor_rec_full[type_pax];
        }
        else
          book_seat(type_pax);
        }
      }
    }
      /*calculate total bookings for all types on each class for
      each day*/
    for (j=0; j<NOCLASSES; j++)
          data_bookings[flt][j][180+day] += tot_booked[j];
  }

  for(day = -180; day <= 0; day++)
        for (int j=0; j<NOCLASSES; j++)
             outdataFile << data_bookings[flt][j][180+day] << " ";
}
```

```
void book_seat(int type_pax)
{
  /*check if seats available in desired class*/
  int j = 0;
      while (class_seats[class_label[type_pax][j]] ==
      tot_booked[class_label[type_pax][j]] && j < NOCLASSES -1)
      j++;

  /*is passenger willing to accept the desired class?*/
  if (willing_to_accept(type_pax, j))
  {
    /* is this class the most desired? */
    if (j == 0)
       sell_seat(class_label[type_pax][j], type_pax);
    else
    {
       sell_seat(class_label[type_pax][j], type_pax);
       vertical_r(class_label[type_pax][j], type_pax);
    }
  }
  else
  {
    /*seats available in the desired class but not willing to
accept*/
    ++pax_hor_rec[type_pax][class_label[type_pax][j]];
  }
}

void sort_classes(int class_label[NOTYPES][NOCLASSES], int type_pax)
{
 /* sort classes according to desirability */
  float hold;
  int hold1;
  for (int pass = 1; pass < NOCLASSES; pass++)
  {
    for (int i = 0; i < NOCLASSES-1 ; i++)
    {
      if (desr[type_pax][i] < desr[type_pax][i+1])
      {
        hold = desr[type_pax][i];
        hold1 = class_label[type_pax][i];
        desr[type_pax][i] = desr[type_pax][i+1];
        class_label[type_pax][i] = class_label[type_pax][i+1];
        desr[type_pax][i+1] = hold;
        class_label[type_pax][i+1] = hold1;
      }
    }
  }
}
```

```
void intialise_variables(int flt)
{
  for (int i = 0; i < NOCLASSES; i++){
    seats_cancelled[i] = 0.0;
    tot_booked[i] = 0.0;}

  for (int j = 0; j < NOTYPES; j++)
  {
    pax_hor_rec_full[j] = 0.0;
    for (int i = 0; i < NOCLASSES; i++)
    {
      pax_seats_booked[j][i] = 0.0;
      pax_ver_rec[j][i] = 0.0;
      pax_hor_rec[j][i] = 0.0;

    }
  }
   for(int day=0; day<=180; day++)
    for(int j=0; j<NOCLASSES; j++)
            data_bookings[flt][j][day]=0.0;
}

/*calculate desirability for each class for this type of pax*/
float get_desr_class(float b[NOTYPES][4], int type_pax)
{
  float b0=b[type_pax][0];
  float b1=b[type_pax][1];
  float b2=b[type_pax][2];
  float b3=1/b[type_pax][3];

  for (int i = 0; i < NOCLASSES; i++)
  {
        desr[type_pax][i]=((class_comf[i]*b0)+(class_fare[i]*b1)-
    b2)*b3;
  }
  return 0;
}

/*add 1 to total seats booked on desired class*/
void sell_seat(int cls, int type_pax)
{
  ++tot_booked[cls];
  ++pax_seats_booked[type_pax][cls];
}

/*vertical recapture*/
void vertical_r(int cls, int type_pax)
{
  ++pax_ver_rec[type_pax][cls];
}
```

```
/*willing to accept class*/
int willing_to_accept(int type_pax, int cls)
{
  if (p_acc[type_pax][cls] >= get_random_number(1, 0))
    return 1;
  else
    return 0;
}


/*calculate the probability of a passenger will arrive in time t1*/
float p_arrive(float parm[NOTYPES][4], int t1, int type_pax)
{
  float p;
  float p0 = parm[type_pax][0];
  float p1 = parm[type_pax][1];
  float p2 = parm[type_pax][2];
  float p3 = parm[type_pax][3];
  float div = (p1-p0)*(p2-p0);
  div = 1/div;

  if (t1 < p0)
        p = 0;

  else if (t1 <= p2)
       p = 2 * (t1 - p0) * div * p3;

  else
     p = 1;
  return p;
}

/*calculate the probability of a passenger willing to accept using
logistic function*/
double prob_of_accept(float desr)
{
  double p;
  p = 1 / (1 + exp(- desr));
  return p;
}

/*generate random number*/
float get_random_number(float n, float m)
{
  float random = ((n-m) * ((float)rand()/RAND_MAX)) + m;
  return random;
}
```

```
/*open output files*/
void openFiles()
{
  if (!inpaxFile) {
    cerr << "inpaxFile file could not be opened" << endl;
      exit(1);}

  for (int i=0; i<NOTYPES; i++)
    for (int j=0; j<=3; j++)
      inpaxFile >> pax_belief_parm[i][j];

  if (!inarrFile) {
    cerr << "inarrFile file could not be opened" << endl;
      exit(1);}

  for (i=0; i<NOTYPES; i++)
    for (int j=0; j<=3; j++)
      inarrFile >> pax_arrival_parm[i][j];

  if (!inclsFile) {
    cerr << "inclsFile file could not be opened" << endl;
      exit(1);}
  for ( i=0; i<NOCLASSES; i++)
   inclsFile >> class_seats[i] >> class_comf[i] >> class_fare[i];

  if (!outdataFile) {
      cerr << "file could not be opened" << endl;
      exit(1);}
}

/*close output files*/
void closeFiles()
{
  inarrFile.close();
  inpaxFile.close();
  inclsFile.close();
  outdataFile.close();
}
```

B.2 *p_arrive* C program for creating p_arrive table

```
/*P_arrive.cpp to calculate all combinations of probability of
arrival for each passenger type on each day and save all probability
of arrivals in file=p_arrive.dat*/

#include <iostream.h>
#include <iomanip.h>
#include <fstream.h>
#include <stdlib.h>
#include <math.h>

#define NOTYPES 3  //number of passenger types
#define Max_arr 10 //Max no. of arrival for each type in 10
intervals

/*functions prototype*/
float prob_of_arrive(float [][4], int, int);
double binom(int, int, double);
double fact(double);
void openFiles(void);
void closeFiles(void);

/* Output files for analysis */
ofstream outarrFile("p_arriveBST.dat", ios::out);
ifstream inarrFile("threepax_arrival_parm.txt", ios::in);

/*passenger a, b, c parameters for probability of arrival*/
float pax_arrival_parm[NOTYPES][4];

/*probability of cancellation is fixed for each type of pax*/
double p_arr[NOTYPES][181][Max_arr+1];


int main()
{
  openFiles();
  /*calculate probability of arrival for each type of pax for each
day*/

  for (int day = 0; day <= 180; day++)
  {
    for (int i=0; i<NOTYPES; i++)
    {
       for (int Arrive = 0; Arrive <= Max_arr; Arrive++)
       {
            p_arr[i][day][Arrive]=binom(Max_arr, Arrive,
            prob_of_arrive(pax_arrival_parm, -180+day, i));
            outarrFile << p_arr[i][day][Arrive] << " ";
```

```
        }
      }
    }
    closeFiles();
    return 0;
}
```

```
double binom(int x1, int x2, double x3)
{
  double p1 = 0, p11=0.0,p111=0.0;
  double p2 = 0.0;

  p1 = fact(x1)/(fact(x2)*fact(x1-x2));
  p11 = pow(x3, x2);
  p111= pow((1-x3), (x1 - x2));
  p2 = p1 * p11 * p111 ;
  return p2;
}


double fact(double x)
{
  if (x<=0) return 1;

   return x*fact(x-1);
}


/*calculate the probability of a passenger will arrive in time t*/
float prob_of_arrive(float parm[][4], int t1, int type_pax)
{
  float p;
  float p0 = parm[type_pax][0];
  float p1 = parm[type_pax][1];
  float p2 = parm[type_pax][2];
  float p3 = parm[type_pax][3];
  float div = (p1-p0)*(p2-p0);
  div = 1/div;

  if (t1 < p0)
        p = 0;
  else if (t1 <= p2)
       p = 2 * (t1 - p0) * div * p3;
  else
     p = 1;

  return p;
}
```

```
/*open output files*/
void openFiles()
{
  if (!outarrFile) {
      cerr << "file could not be opened" << endl;
      exit(1);
  }

  if (!inarrFile) {
      cerr << "inarrFile file could not be opened" << endl;
      exit(1);
  }

  for (int i=0; i<NOTYPES; i++)
    for (int j=0; j<=3; j++)
      narrFile >> pax_arrival_parm[i][j];
}
/*close output files*/
void closeFiles()
{
  outarrFile.close();
  inarrFile.close();
}
```

B.3 *p_accept* C program for creating p_accept table

```
/*P_accept1.cpp to Calculate from 0 to 10 arrival the prob. of 0 to
10 acceptance for each prob. of an acceptance [0-1] and save in file
p_accept1.dat */

#include <iostream.h>
#include <iomanip.h>
#include <fstream.h>
#include <stdlib.h>
#include <math.h>

#define Max_arr 10 //Max number of arrival for each type in 10
intervals

/*functions prototype*/
void calc_prob(void);
double binom(int, int, double);
double fact(double);
void openFiles(void);
void closeFiles(void);

/* Output files for analysis */
ofstream outaccFile("p_accept.dat", ios::out);

double p_acc[11][Max_arr+1][Max_arr+1];

int main()
{
  openFiles();
  for (double p=0; p<1.01; p+=0.1)
  {
    double p_acc =0;
    for (int Accept=0; Accept <= Max_arr; Accept++)
    {
      for (int Arrive=0; Arrive <= Max_arr; Arrive++)
      {
        if (Arrive >= Accept)
        {
          p_acc = binom(Arrive, Accept, p);
          if (p_acc <= 1.0e-14) p_acc=0;
        }
        else
        {
          p_acc = 0;
        }
        outaccFile << p_acc << " ";
      }
    }
```

```
    }
    closeFiles();
    return 0;

}
```

```
double binom(int x1, int x2, double x3)
{
  double p1 = 0, p11=0.0,p111=0.0;
  double p2 = 0.0;

  p1 = fact(x1)/(fact(x2)*fact(x1-x2));
  p11 = pow(x3, x2);
  p111= pow((1-x3), (x1 - x2));
  p2 = p1 * p11 * p111 ;
  return p2;
}

double fact(double x)
{
  if (x==0) return 1;

   return x*fact(x-1);
}


/*open output files*/
void openFiles()
{
  if (!outaccFile) {
      cerr << "file could not be opened" << endl;
      exit(1);
  }

}

/*close output files*/
void closeFiles()
{
  outaccFile.close();
}
```

B.4 *precomp* C program for creating p_tot table

```
/*precomp2.cpp computes the probabilities of all the 31 (0 bookings
to max 30 bookings) possible events that could occur in every day of
the booking process for different combinations of prob_of_acceptance
for all types of passengers(p_acc=0 to p_acc=1) ie (181x11x11x11x31)
for three types of passengers and stores in p_allb.dat file (41.6MB)
to be read in metropolis program */

#include <iostream.h>
#include <iomanip.h>
#include <fstream.h>
#include <stdlib.h>
#include <math.h>

#define NOTYPES 3     //number of passenger types
#define Max_arr 10    //Max number of arrival for each type in 10
intervals
#define All_arr Max_arr*NOTYPES //Arrival of all types

/*functions prototype*/
void calc_prob1(void);
double calc_prob2(int, int, int);
void openFiles(void);
void closeFiles(void);

/* Input and Output files */
ifstream inarrFile("p_arriveBST.dat");
ifstream inaccFile("p_accept.dat");
ofstream outpallFile("p_allbBST.dat",ios::out|ios::binary);

double p_arr[NOTYPES][181][Max_arr+1];
double p_acc[11][Max_arr+1][Max_arr+1];
double pbook[NOTYPES][Max_arr+1];
float ptot[All_arr+1];

int main()
{
  openFiles();
  calc_prob1();
  closeFiles();
  return 0;
}
```

```
void calc_prob1(void)
{
  for (int day = 0; day <= 180; day++)
  {
    for (int paccb=0; paccb<=10; paccb++)
      for (int paccs=0; paccs<=10; paccs++)
        for (int pacct=0; pacct<=10; pacct++)
        {
         calc_prob2(0, paccb,day);
         calc_prob2(1, paccs,day);
         calc_prob2(2, pacct,day);
         for(int tot=0; tot<All_arr+1; tot++)ptot[tot]=0;

         for(int b=0; b <= Max_arr; b++)
           for(int s=0; s <= Max_arr; s++)
            for(int t=0; t <= Max_arr; t++){
              ptot[t+s+b] += pbook[0][b]*pbook[1][s]*pbook[2][t];}
              outpallFile.write((char *)(&ptot), sizeof(ptot));
        }
  }
  return;
}

double calc_prob2(int type,int pacc, int day)
{
  for (int book=0; book <= Max_arr; book++)
  {
      pbook[type][book]=0;
      for (int narr = book; narr <= Max_arr; narr++)
      {
            pbook[type][book] += p_arr[type][day][narr] *
            p_acc[pacc][book][narr];
      }
  }
  return 0;
}
```

```
/*open output files*/
void openFiles()
{
  int i,day, arr, acc,a ;
  if (!outpallFile) {
      cerr << "file could not be opened" << endl;
      exit(1);
  }

  if (!inarrFile) {
      cerr << "file could not be opened" << endl;
      exit(1);
  }

  for (day=0; day<=180; day++)
      for ( i=0; i< NOTYPES; i++)
        for ( a = 0; a <= Max_arr; a++)
            inarrFile >> p_arr[i][day][a];


  if (!inaccFile) {
      cerr << "file could not be opened" << endl;
      exit(1);
  }

  for ( i=0; i<=10; i++)
      for ( arr = 0; arr <= Max_arr; arr++)
         for ( acc = 0; acc <= Max_arr; acc++)
            inaccFile >> p_acc[i][arr][acc];
}

/*close output files*/
void closeFiles()
{
  outpallFile.close();
  inarrFile.close();
  inaccFile.close();
}
```

B.5 *met_onepax* C program for Metropolis algorithm

```
/*met_onepax.cpp Metropolis Algorithm for one passenger booking data
and 3 classes*/

#include <iostream.h>
#include <iomanip.h>
#include <fstream.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define NOCLASSES 3  //number of classes on this flight
#define NOTYPES 1    //number of passenger types
#define Max_arr 10   //Max number of arrival for each type in 10
intervals
#define All_arr Max_arr*NOTYPES //Arrival of all types
#define FLIGHT 100   //number of flights
#define ITR 100000   //metropolis iteartions

/*functions prototype*/
void create_hash_table(void);
void get_desr_and_pacc(float [NOTYPES][4]);
int good_or_bad_parm(void);
double get_Likelihood(void);
float change_one_parm(float [NOTYPES][4]);
void find_ratio(float [NOTYPES][4], float[NOTYPES][4]);
float get_random_number(float, float);
float one_interp(int, int, double[NOTYPES] , int[NOTYPES], int);
float two_interp(int, int, double[NOTYPES] , int[NOTYPES], int);
float three_interp(int, int, double[NOTYPES], int[NOTYPES], int);
float interpolate(float, float, float);
float get_ptot(int, int, double [NOTYPES][NOCLASSES], int);
void openFiles(void);
void closeFiles(void);


/* Output files for analysis */
ifstream indataFile("diff1pax3cls100N_data.dat",ios::in);
ifstream inpallFile("p_allbMdiff1.dat",ios::in|ios::binary);
ofstream outpaccFile("diff1pax3cls_pacc.dat",ios::out);
ifstream inclsFile("diffpax_3class3_parm.txt", ios::in);

int data_bookings[FLIGHT][NOCLASSES][181];
int class_seats[NOCLASSES], /* number of seats */
    class_comf[NOCLASSES],
    class_fare[NOCLASSES];
int flt_hash_table[181][NOCLASSES][All_arr+1];
double grid_point1[NOTYPES][NOCLASSES];
```

```
double grid_point2[NOTYPES][NOCLASSES];
double index[NOTYPES][NOCLASSES];
double p_acc[NOTYPES][NOCLASSES];
double p_acc_old[NOTYPES][NOCLASSES];
int cls_label[NOTYPES][NOCLASSES];
double loglik_1=0;
double loglik_2=0;
```

```
int main()
{
  ifstream inpaxFile("diff1pax_belief_parm.txt", ios::in);
  ofstream outparmFile("Metdiff1pax3cls100N.dat", ios::out);
  char start[128], finish[128];
  float pax_belief_parm[NOTYPES][4];
  float parm_1[NOTYPES][4], parm_2[NOTYPES][4];

  srand(time(0));
  openFiles();
  if (!inpaxFile) {
    cerr << "inpaxFile file could not be opened" << endl;
    exit(1);
  }

  for (int i=0; i<=NOTYPES; i++)
  {
    for (int j=0; j<=3; j++)
    {
       inpaxFile >> pax_belief_parm[i][j];
    }
  }

  if (!outparmFile) {
      cerr << "file could not be opened" << endl;
      exit(1);
  }
  if (!outpaccFile) {
      cerr << "file could not be opened" << endl;
      exit(1);
  }

  _strtime(start);
  outparmFile << "program started at: " << start << endl << endl;


  /*Metropolis algorithm*/
  for (i = 0; i <= NOTYPES-1; i++)
   for (int j = 0; j <= 3; j++){
       parm_1[i][j] = 0.0;
       parm_2[i][j] = 0.0;}

  //give any arbitrary starting values to the parameters
  parm_1[0][0] = (float)0.5;
  parm_1[0][1] = (float)-0.3;
  parm_1[0][2] = (float)-300;
  parm_1[0][3] = (float)100;

  //output the starting combination of parameters
  outparmFile << "-10 ";
```

```
  for (i = 0; i <= NOTYPES-1; i++){
   for (int j = 0; j <= 2; j++)
     outparmFile << setw(8) << setprecision(3)
     << parm_1[i][j] << " ";}

  /*calculate the desiarability and pacc for each passenger type to
  each class*/
  get_desr_and_pacc(parm_1);
  int this_parm=0;
  this_parm=good_or_bad_parm();
  if (this_parm == 0)
  {
   //calculate the liklehood associated with the starting
parameters
   loglik_1 = get_Likelihood();
   outparmFile << "     " << setw(10)<< setprecision(6)<< loglik_1;
  }


  //Metropolis loop
  for (int k = 0; k < ITR; k++)
  {
    for (i = 0; i <= NOTYPES-1; i++)
     for (int j = 0; j <= 3; j++)
       parm_2[i][j] = parm_1[i][j];

     //select a parameter at random and add a random number to it
     change_one_parm(parm_2);

     /*calculate desiarability and pacc for each type on each class
     with new set of parm.*/
     get_desr_and_pacc(parm_2);
     int this_parm=0;
     this_parm=good_or_bad_parm();
     if (this_parm == 0)
     {
       //calculate the liklehood associated with the new parameters
       loglik_2 = get_Likelihood();

       //find_ratio
       find_ratio(parm_1, parm_2);

       //output the accepted 12 parameters and log_Likelihood
       if(k % 10 == 0)
       {
         outparmFile << k << " ";
         cout << k << " ";
         for (int i = 0; i <= NOTYPES-1; i++)
           for (int j=0; j <= 2; j++)
           {
```

```
                outparmFile << setw(8) << setprecision(3) <<
                parm_1[i][j] << "   ";
              }
            outparmFile << "   " << setw(10)<<
               setprecision(6)<< loglik_1;
          }
        }
    }
    _strtime(finish);
    outparmFile << endl << "program finished at: " << finish << endl;
    inpaxFile.close();
    outparmFile.close();
    outpaccFile.close();
    closeFiles();
    return 0;
}
```

```
int good_or_bad_parm(void)
{
  int this_parm1=0;
  for (int day = 0; day <= 180; day++)
  {
   int j = 0;
   int NOClassless1=NOCLASSES-1;
   //is there a booking on any class apart from the most desired
class
  while

(class_seats[cls_label[0][j]]==data_bookings[0][cls_label[0][j]][day
])
    j++;

  for(int i=j+1; i<NOClassless1+1; i++)
  {
   //bad combination of parm
   if(data_bookings[0][cls_label[0][i]][day] != 0)this_parm1 = 1;}
  }
  return this_parm1;
}
```

```
void create_hash_table(void)
{
  int today_bookings; /*today's booking*/
  //if all classes are full keep the last number of days
  int fullday = -1000;
  int notfull = 0;
  int d=0;
  while (fullday<0)
  {
      notfull=0;
      for (int c = 0; c < NOCLASSES; c++)
      {
        if (class_seats[c] != data_bookings[0][c][d])
              notfull=1;
      }
      if (notfull==0)
              fullday=d;
      d++;
  }
  for (int day = 0; day <= d; day++)
  {
    for (int cls = 0; cls < NOCLASSES; cls++)
      for (int t_b = 0; t_b < All_arr+1; t_b++)
      {
         flt_hash_table[day][cls][t_b] = 0;
      }
    today_bookings=0;
    for (int flt=0; flt < FLIGHT; flt++)
    {
      int j = 0;
      int NOClassless1=NOCLASSES-1;
      while (class_seats[cls_label[0][j]] ==
      data_bookings[flt][cls_label[0][j]][day-1] && j <
      NOClassless1)j++;
      if (class_seats[cls_label[0][j]] !=
      data_bookings[flt][cls_label[0][j]][day])
      {
        int mycls=cls_label[0][j];
        double mypacc=p_acc[0][j];
        /*if first day of booking process then today_bookings equal
        to      todays's booking*/
        if (day>0)
        {
            today_bookings = data_bookings[flt][mycls][day] -
            data_bookings[flt][mycls][day-1];
            flt_hash_table[day][mycls][today_bookings]++;
        }
        else
        {
            today_bookings = data_bookings[flt][mycls][0];
```

```
                flt_hash_table[day][mycls][today_bookings]++;
            }
        }

    } //close flt loop
  }//close day loop
}
```

```
double get_Likelihood(void)
{
  double log_cls[NOCLASSES]={0,0};
  float ptot=0;
  double log_all_data;
  double log_per_cls;
  double dx[NOTYPES]={0};
  log_all_data=0;
  create_hash_table();

  for (int day = 0; day <= 180; day++)
  {
   log_per_cls=0;
   for (int t1 = 0; t1 < NOTYPES; t1++)
   {
     for (int c1 = 0; c1 < NOCLASSES; c1++)
     {
       grid_point1[t1][c1] = 0;
       grid_point2[t1][c1] = 0;
       index[t1][c1]=0;
     }
   }

   for (int cls = 0; cls < NOCLASSES; cls++)
   {
     for (int t_b = 0; t_b < All_arr+1; t_b++)
     {
      if (flt_hash_table[day][cls][t_b] != 0)
      {
        for (int t = 0; t < NOTYPES; t++)
        {
           double mypacc=p_acc[t][cls_label[t][cls]];
           grid_point1[t][cls]=(int(mypacc*10))*0.1;
           double mydiff=mypacc-grid_point1[t][cls];
           if (mydiff >0)
           {
             dx[t]=mydiff;
             grid_point2[t][cls]=grid_point1[t][cls]+0.1;
           }
           else
           {
             grid_point2[t][cls]=grid_point1[t][cls];
           }
        }
        int label[NOTYPES]={0};
        int a=0, b=0;
        for (t = 0; t < NOTYPES; t++)
        {
          //if pacc lies between two pacc's eg. [0, 0.37, 0.8]
          if (fabs(grid_point2[t][cls] - grid_point1[t][cls])>0)
```

```
    {
      a = a+1;
      label[b]=t;
      b=b+1;
    }
}
```

```
        //for this class move the first set of grid_points
        //to index eg. class 0 with only type 0 (bus) passenger
        //pacc [0.9, 0, 0]
        for (t = 0; t < NOTYPES; t++)
        {
           index[t][cls]=grid_point1[t][cls];
        }
        log_per_cls=0;
        if (a==1)
            ptot=one_interp(day, cls, dx, label, t_b);
        else if (a==2)
            ptot=two_interp(day, cls, dx, label, t_b);
        else if (a==3)
            ptot=three_interp(day, cls, dx, label, t_b);
        else
        {
            //no interpolation required
            ptot = get_ptot(day, cls, index, t_b);
        }

        if (ptot>0)
           log_per_cls=log(ptot);

        else log_per_cls= -1.0e99;

        log_all_data +=
        log_per_cls*(double)flt_hash_table[day][cls][t_b];
      }
     } //close flt_table loop
   } //close class loop
  } //close day loop
  return log_all_data;
}
```

```
float one_interp(int day, int cls, double dx[NOTYPES], int
l[NOTYPES], int t_bookings)
{
  index[l[0]][cls] = grid_point1[l[0]][cls];
  float ptot_1 = get_ptot(day,  cls, index,t_bookings);

  index[l[0]][cls] = grid_point2[l[0]][cls];
  float ptot_2 = get_ptot(day, cls, index, t_bookings);

  float lamda = (float)dx[l[0]]/((float)grid_point2[l[0]][cls]-
  (float)grid_point1[l[0]][cls]);

  float lptot=interpolate(ptot_1, ptot_2, lamda);
  return lptot;
}

float two_interp(int day, int cls, double dx[NOTYPES], int
l[NOTYPES], int t_bookings)
{
  index[l[1]][cls] = grid_point1[l[1]][cls];
  float z1 = one_interp(day, cls,dx, l, t_bookings);

  index[l[1]][cls] = grid_point2[l[1]][cls];
  float z2 = one_interp(day, cls,dx, l , t_bookings);

  float mue = (float)dx[l[1]]/((float)grid_point2[l[1]][cls]-
  (float)grid_point1[l[1]][cls]);

  float lptot=interpolate(z1, z2, mue);
  return lptot;
}

float three_interp(int day,int cls, double dx[NOTYPES], int
l[NOTYPES], int t_bookings)
{
  index[l[2]][cls]=grid_point1[l[2]][cls];
  float z1 = two_interp(day, cls, dx, l, t_bookings);

  index[l[2]][cls]=grid_point2[l[2]][cls];
  float z2 = two_interp(day, cls, dx, l, t_bookings);

  float fin = (float) dx[l[2]]/((float)grid_point2[l[2]][cls]-
  (float)grid_point1[l[2]][cls]);

  float lptot=interpolate(z1, z2, fin);
  return lptot;
}

float interpolate(float ptot_1, float ptot_2, float dy)
{
```

```
  float lptot = ((1-dy)*ptot_1) + (dy*ptot_2);
  return lptot;
}
```

```
//find the value of total prob. (ptot) from the binary file
float get_ptot(int day, int cls, double index[NOTYPES][NOCLASSES],
int t_bookings)
{
  double ind[NOTYPES]={0};
  float lptot=0;
  long pointer=0;
  long bookings = (long)t_bookings;
  double i0=0;
  int ii=0,i1=0,i2=0;

  for (int t1 = 0; t1 < NOTYPES;t1++)
  {
      ind[t1] = (index[t1][cls])*10;
  }

  i0 = ind[0];
  ii=ceil(i0);

  /*to point to any location: multiply day number by 11 values for
  each   pacc (11*11*11=1331) by 31 vaulues for all passenger
  arrivals (0-30) by 4 bytes for each value */

  pointer = (long)pow(All_arr+1, NOTYPES);
  pointer = pointer*(long)day*(long)(All_arr+1)*4;
  pointer = pointer + i0*(long)(All_arr+1)*4;

  //add to pointer the value of today_bookings (bookings) * 4 bytes
  pointer = pointer + bookings*4;

  inpallFile.seekg(0);
  inpallFile.seekg(pointer);
  inpallFile.read((char *)(&lptot), sizeof(lptot));
  inpallFile.seekg(0);
  return lptot;
}
```

```
void get_desr_and_pacc(float parms[NOTYPES][4])
{
  float desr[NOTYPES][NOCLASSES];

  //calculate desirability for each passenger type on each class
with
  //new set of parm
  for (int i=0; i<NOTYPES; i++)
  {
    float b0=parms[i][0];
    float b1=parms[i][1];
    float b2=parms[i][2];
    float b3=1/parms[i][3];

    for (int j = 0; j < NOCLASSES; j++)
    {
      desr[i][j] =((class_comf[j]*b0)+(class_fare[j]*b1)-b2)*b3;
    }

    for (j=0; j<NOCLASSES; j++)
    {
        cls_label[i][j] = j;
    }

    //sort classes according to desirability
    float hold;
    int hold1;
    for (int pass = 1; pass < NOCLASSES; pass++)
    {
       for (int j = 0; j < NOCLASSES-1 ; j++)
       {
         if (desr[i][j] < desr[i][j+1])
         {
            hold = desr[i][j];
            hold1 = cls_label[i][j];
            desr[i][j] = desr[i][j+1];
            cls_label[i][j] = cls_label[i][j+1];
            desr[i][j+1] = hold;
            cls_label[i][j+1] = hold1;
         }
       }
    }
  }

  for(i=0; i<NOTYPES; i++)
  {
    for(int j=0; j<NOCLASSES; j++)
    {
      //Calculate p_acc for each passenger type for each class
      p_acc[i][j] = 1 / (1 + exp(-desr[i][j]));
```

```
        }
    }
}
```

```
/*select a parameter at random and add a random number to it*/
float change_one_parm(float parm_2[NOTYPES][4])
{
  int random_0, random_1;
  random_0 = NOTYPES-1; /*only onepax*/
  //selcet one parameter at random[0,1or2]
  random_1 = rand()%3;
  float test = 0.0;
  float random_2 = 0.0;
  switch (random_1)
  {
    //parm_2[0][0] must have a value between 0, 1
    case 0:
        test = parm_2[random_0][0];
        random_2 = get_random_number(0.1, -0.1);
        test += random_2;
        while (test > 1 || test < 0)
        {
            random_2 = get_random_number(0.1, -0.1);
            test += random_2;
        }
        parm_2[random_0][0] = test;
        break;

    //parm_2[0][1] must have a value between -1, 0
    case 1:
      test = parm_2[random_0][1];
      random_2 = get_random_number(0.1, -0.1);
      test += random_2;
      while (test > 0 || test < -1)
      {
          random_2 = get_random_number(0.1, -0.1);
          test += random_2;
      }
      parm_2[random_0][1] = test;
      break;
    //parm_2[0][2] must have a value between -300, -500
    case 2:
      test = parm_2[random_0][2];
      random_2 = get_random_number(20, -20);
      test += random_2;
      while (test > -300 || test < -500)
      {
          random_2 = get_random_number(20, -20);
          test += random_2;
      }
      parm_2[random_0][2] = test;
      break;
  }
  return parm_2[random_0][random_1];
```

```
}
/*generate random number*/
float get_random_number(float n, float m)
{
  float random = ((n-m) * ((float)rand()/RAND_MAX)) + m;
  return random;
}
```

```
/*find_ratio*/
void find_ratio(float parm_1[NOTYPES][4], float parm_2[NOTYPES][4])
{
    if (loglik_2 > loglik_1)
    {
      loglik_1 = loglik_2;
      for (int i = 0; i <= NOTYPES-1; i++)
        for (int j = 0; j <= 3; j++)
          parm_1[i][j] = parm_2[i][j];

    }
    else
    if ((exp((loglik_2 - loglik_1))) >= get_random_number(1, 0))
    {
      loglik_1 = loglik_2;
      for (int i = 0; i <= NOTYPES-1; i++)
        for (int j = 0; j <= 3; j++)
          parm_1[i][j] = parm_2[i][j];

    }
}

/*open output files*/
void openFiles()
{
   if (!indataFile) {
    cerr << "indataFile file could not be opened" << endl;
      exit(1);
  }

  for (int flt=0; flt < FLIGHT; flt++)
    for (int day=0; day<=180; day++)
        for (int i=0; i< NOCLASSES; i++)
            indataFile >> data_bookings[flt][i][day];
  if (!inclsFile) {
    cerr << "inclsFile file could not be opened" << endl;
      exit(1);
  }
  for (int i=0; i<NOCLASSES; i++)
      inclsFile >> class_seats[i]  >> class_comf[i] >>
class_fare[i];

}

/*close output files*/
void closeFiles()
{
  indataFile.close();
  inpallFile.close();
  inclsFile.close();
```

```
    outpaccFile.close();
}
```