Experimental Investigation of Quasi-Newton Approaches to a Learning Problem in Electronic Negotiation

by

Paul Meloche

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Management Sciences

Waterloo, Ontario, Canada, 2007

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Abstract. The recent growth in electronic commerce has motivated the development of semi-autonomous negotiation systems capable of implementing multiple negotiations simultaneously. Different approaches have recently been presented in the literature with the aim of providing a solution to this growing market segment. The current thesis presents an examination of optimization approaches for learning the parameters of a time-dependent decision-function that has recently obtained significant interest in the negotiation literature. Twelve different nonlinear optimization variants are evaluated using 800 problems, and the resulting 9600 runs are statistically analyzed on four different performance measures. Potential implications of our analysis are discussed for their possible use in the context of electronic negotiation.

Acknowledgements

I would like to thank my supervisor, Dr. R.P. Sundarraj for his continuous support, both mentally and financially. He has provided me with valuable guidance and encouragement along the way. Without his help and understanding, I could not have completed this thesis. I would also like to thank the readers for their time and guidance.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Motivation

Over the last decade, advancements in information technology have fuelled the growth of electronic commerce (e-commerce) into an essential part of many businesses. With the current growth of the global economy, vendors and purchasers are forced to seek and source creative pricing techniques in order to remain competitive. Previously, a small business owner may have been able to support a self-sustaining business by only selling products locally at a fixed price. However, this simple business model is becoming less effective as global competition begins to rise. The need for a purchaser/vendor to buy/sell products globally using selective pricing techniques has never been greater.

The current thesis considers the situation in which negotiations are conducted via the Internet similar to the methods used by Ozro Negotiate™ and AuctionBot. Such a negotiation shall henceforth be termed as electronic negotiation (e-negotiation). We focus on developing an efficient technique that allows an individual to negotiate prices for goods bought and sold. In order to make this technique applicable in a real-world situation, it will need to be flexible; it will need to use a minimal amount of computational power and it will also need to work semi-autonomously in real-time. Flexibility is a key issue while searching for an efficient technique. A flexible approach will allow the technique to be applicable to negotiation parties that use significantly different negotiation tactics. For the method to be widely used, it will also need to use no more computational power than what is currently available in a standard desktop computer. Last, the method must be able to complete negotiations without much human

intervention and in real-time. If all of the above criteria are met, an individual will be able to effectively negotiate prices selectively with multiple individuals across the world simultaneously. By selectively pricing the goods bought and sold to each individual, the vendor or purchaser will be able to maximize the profit generated by the purchase/sale of the good.

Research into e-negotiation has been conducted from different perspectives, including Game theoretic approaches (Harsanyi, 1972), Bayesian approach (Zeng and Sycara, 1998) and heuristic approaches (Kim, 2000; Mok and Sundarraj, 2005; Jennings, Faratin et al., 2001). The use of optimization techniques to obtain a solution, which is the current interest in this thesis, is still in its infancy and will be thoroughly examined in this paper. Various nonlinear least-squares optimization techniques will be used and tested to verify their effectiveness in obtaining a robust solution to the negotiation problem.

1.2 Research Goal

In this thesis, we consider a tactic, known as the time-dependent tactic (TDT) that has been used in other works related to electronic negotiation (Faratin, 1998; Deveaux et al., 2001; Mok and Sundarraj, 2005). With TDT, negotiators treat time as an important aspect impacting the value of their offers (Pruitt, 1981). Using a mathematical model of TDT (Faratin et al., 1998), the underlying negotiation parameters are attempted to be learned, by only knowing the price offers that have been made by an opponent in an ensuing negotiation. This learning problem is modeled as a nonlinear least-squares problem, and its solution using optimization algorithms is tested.

The goal of this thesis is to improve upon the previous work performed on learning algorithms for electronic negotiation (Shi, 2005). We consider several

algorithms to improve both the convergence rate, as well as accuracy of the parameter estimation. The aspect of parameter-estimation accuracy is especially important to this thesis.

Several nonlinear least-square algorithms and line search methods will be examined in order to meet our research objective. One class of algorithms suggested in the literature is the quasi-Newton approach. We will compare the effectiveness of variants of the quasi-Newton approach over that of the simpler Gauss-Newton method. We will also examine the effect of pre-processing starting points and the use line search algorithms. This results in a combination of 12 algorithms. We test them all on four different performance measures, and include observations based on statistical analysis.

A successful improvement on all examined parameters will allow the development of a negotiation system that can use nonlinear optimization algorithms in order to improve current negotiation performance. This may also lead to a potential commercial application of an electronic agent that uses these techniques to predict the negotiation behaviour of an opponent at the next iteration of a negotiation.

1.3 Structure

Chapter 2 describes the literature review. It deals with the definition of negotiation and fundamentals of negotiation. It also highlights current trends in electronic commerce, the use of adaptive electronic agents to facilitate electronic negotiation, and provides some real world examples of their potential use.

Chapter 3 deals with the formulation of a time dependent negotiation model with practical uses in electronic negotiation systems. The model is presented and then the learning problem is formulated as a nonlinear least squares model.

Chapter 4 explains the current methods used to solve nonlinear least squares problems. A breakdown of detailed methods used includes pre-processing methods, line search approaches and specific least squares algorithms.

Chapter 5 deals with the experimental design for our tests. The performance measures for the solution approaches are presented and their significance in terms of electronic negotiation is discussed.

Chapter 6 provides detailed statistical analyses examining the effectiveness of the methods used and the significance of the results obtained in the context of electronic negotiation. A brief summary is given to provide insights into the best combination of algorithms to effectively solve the learning problem.

Chapter 7 deals with the possible implication of the methods developed in this work. Conclusions and future work are also discussed.

Chapter 2 Literature Review

This chapter provides an overview of negotiation theory, electronic commerce and current applications of electronic negotiation. The material provided in this chapter serves as a basis for understanding the importance of negotiation in an electronic commerce setting. We then introduce the idea of an electronic agent and discuss the application of agents to electronic negotiation.

2.1 Negotiation

2.1.1 Definition

Negotiation can be defined as (Pruitt, 1981):

> "A process by which a joint decision is made by two or more parties. The parties first verbalize contradictory demands and then move towards agreement by a process of concession making or search for new alternatives"

Negotiation is used in everyday purchasing, pricing and bargaining. Negotiation occurs in the interactions of almost everyone in groups and organizations: Labour bargains with management; managers negotiate with employees, peers and senior management; sales people negotiate with customers; purchasing agents negotiate with suppliers. In today's team-based organizations, negotiation skills become critical, so that teams can work together efficiently (Robbins, 2005).

We next discuss the fundamentals of negotiation.

2.1.2 Fundamentals of Negotiation

One of the main attributes of negotiation is that the parties involved start off with opposing interests and preferences (Pruitt,1981). Each party has a given benefit (henceforth termed *utility*) for a specific outcome. At each iteration of the negotiation, a buyer/seller is expected to make a bid/offer that will decrease his/her utility, in the hope of keeping the negotiation in progress. A concession from each party at each iteration is paramount to ensure that a final agreement is reached. Two-party bargaining can be divided into two types: integrative and distributive (Raiffa, 1982).

Integrative bargaining can be defined as a negotiation situation where there exists more than one final settlement where both parties can emerge victorious (Robbins, 2005). In this situation, informally called a win-win situation, both parties can work together in order to reach a settlement where both parties increase their respective utility. A simple example to illustrate this situation can be explained by labour unions bargaining with management over more health benefits for their employees. The union would like to have health benefits to keep its employees healthy and happy. Management does not want to offer increased health benefits on account of the additional cost, but is also concerned with the downside of a discouraged workforce that may be less productive. If both parties work together towards a common goal, a mutually beneficial outcome is possible. For example, if management works together with the union to find a cost effective way to insure its unionized employees with health benefits, it may lead to a win-win situation where the company gains from higher productivity of the workforce, and the unionized employees gain from having health benefits.

The other form of negotiation is distributive in nature. That is, when a negotiation is between two parties, we will encounter a win-lose (or distributive) situation, when

there is only one negotiation issue (e.g., cost). An example of this situation would be the cash-purchase of a used car from a dealer. The potential buyer of the automobile would be inclined to keep the buying price low, while the potential seller would be inclined to keep the selling price high. Both parties have a respective reservation price, which is the highest (lowest) price that the buyer (seller) is willing to give (take). In general, the reservation price of each negotiation party is not known to the opponent. This in fact is a fundamental attribute in the negotiation scheme. If the reservation price of the seller is higher than the reservation price of the buyer, the negotiation will not conclude. However, if there is a zone of potential agreement (ZOPA), there exists a possibility that the negotiation will reach a final settlement price at which the transaction will be made. In the current thesis, we will assume that the ZOPA always exists and that the reservation price of the negotiation is always fixed prior to the start of the negotiation. An illustration of the above example is given in Figure 1.

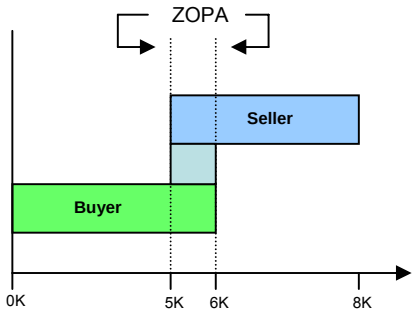

Figure 2.1 Illustration of the zone of potential agreement

In Figure 2.1, the lowest price that the buyer is willing to pay is $0, and the highest price is $6k, which is also the buyer's reservation price. The maximum price at which the wants to sell the car is $8k, which can be viewed as the "best" retail price of the automobile, and the lowest price, the reservation price, at which he is willing to sell, is $5k. The zone of potential agreement in this example is $1k, which is the price difference between the buyer and seller's reservation price for the sale of the automobile. In this case, for every dollar that the seller gains by raising the selling price, the buyer loses, and vice versa. Therefore, distributive bargaining can be looked at as a zero-sum game.

Other than reservation price of the buyer and seller, and the ZOPA of the negotiation, several other negotiation characteristics need to be outlined. One of the most important characteristics of negotiation behaviour is the concession rate of each party engaged in the negotiation. In order to understand how a given negotiators' concession rate affects negotiation, we must first understand the ultimate goal of each negotiator. A bargainers' demand level can be thought of to correspond to the level of benefit (or utility) to the buyer (Pruitt, 1981). For example, two parties negotiating over the price of a given object are only concerned with the final negotiation price. If the negotiation price rises, the benefit increases for the seller and decreases for the buyer. Therefore, in order for the seller to make a concession, he or she must reduce the offer in selling price to increase the level of benefit to the buyer. It is extremely important to make concessions in a negotiation in order to ultimately reach an desired negotiation price. Concessions are generally made in the anticipation that the concession maker will hasten the agreement, will prevent the other party from leaving the negotiation, or to encourage the other party

to make reciprocal concessions (Pruitt, 1981). Therefore, both the size of the concession and the amount of elapsed time between concessions, defined as the concession rate, plays a crucial role in the outcome of the negotiation.

The role of time in a negotiation will have a large effect on both the concession rate and the final outcome. As stated by Raiffa (1982), in negotiations conducted in laboratory settings, subjects show an almost uncanny ability to detect even small ZOPAs, but the smaller the zone, the longer it usually takes them to agree on a solution (Raiffa, 1982). Therefore, it can be inferred from this statement that a negotiation under serious time restrictions would lead to a lower probability of price convergence, ultimately in a case where time is of the essence, a proper concession rate needs to be determined in order to ensure a positive negotiation outcome. In other words, the parties in the negotiation must make concessions quickly enough in order for them to come to an agreement before time runs out.

In the next section, a look at the currently expanding level of commercial activity in e-commerce will motivate the growing need for adaptive, semi-autonomous negotiation.

2.2 Electronic Commerce

2.2.1 Introduction

Electronic commerce is an emerging sector in which business approaches are able to engage with their customers electronically (rather than by phone or in person) in all phases of a business transaction. Electronic commerce have enabled customers to, for example, access product information, select items to purchase, purchase items securely,

and have the purchase settled financially (International Engineering Consortium, 2007). Electronic marketplaces are becoming important players to several industries, because they promise to greatly improve economic efficiency, reduce margins between price and cost, and speed up complicated business deals (Feldman, 2000). Examples of electronic market places include eBay, Equipnet and Officemax where products range from stationary supplies to x-ray generators (eBay, 2007).

Online sales by Canadian companies and government departments grew substantially for the fifth consecutive year in 2004, but e-commerce still accounted for less than 1% of total operating revenues for private businesses (Government of Canada, 2006). The potential for growth in the field of electronic commerce remains strong and is one of the fundamental factors motivating this thesis.

For many firms, e-commerce is one of many steps involved in fully integrating business practices using the Internet. Business-to-Business e-commerce will affect the way that businesses run in several ways, including accelerating business processes, creating transparent markets and redefining market boundaries (Global Reach, 1998). The current reach of the internet will allow markets to be penetrated much faster, enabling the first player in the game to take a significant stake in the business. This puts much emphasis on a company's ability to adapt to the current trends in electronic commerce in order to gain or maintain a valuable market position (Stone, 2005).

Another area in which e-commerce is currently altering the business world is in the globalization of business. Electronic commerce increases the range of services that can be traded internationally (e.g., to include medical, legal, and educational services) and can provide access to markets that were previously closed (Panagariya, 1999). This new-found globalization of business markets will in turn lower transaction and

production costs, facilitate market entry and increase competition by allowing various participants to enter the market who previously were not able due to their location. As a result, this will provide lower prices, increased quality, and provide the creation of new and more diverse products, thereby increasing economic growth and social welfare (Panagariya, 1999).

The introduction of e-commerce has also allowed for the formation of transparent and competitive markets, where differentiation will be essential to survive. Several benefits will be passed off to the consumer, including market knowledge and reduced search costs, even if customers make the final purchase in person (Economist, 2000). Companies will now have to be able to communicate how they are different than their competitors; this can be accomplished by direct comparison of features and price, global delivery, customizing and easy access to worldwide product information.

One area facilitated by e-commerce businesses is that of electronic negotiation (Choi et al., 2001). Electronic negotiation has allowed businesses, such as eBay, to develop fast and efficient ways to deal with multiple customers simultaneously, without undue stress to its labour resources (International Engineering Consortium, 2007). Also, automated negotiation will allow for various techniques (e.g., dynamic pricing) that were previously considered to be difficult for small-scale businesses. This change in the business paradigm has ultimately led to the development of efficient, semi-autonomous systems that can potentially reduce labour-intensive practices to conduct routine activities

The next sections will discuss electronic negotiation and electronic agents, along with a brief description of their fundamental characteristics and current applications.

2.3 Electronic Negotiation

Electronic Negotiations are considered a key component of e-commerce (Sandholm, 1999). They are now playing an increasing role in everyday transactions between businesses, and between customers and businesses (Mahadevan, 2000). For example, eBay boasts more than 220 million registered users selling more than 50,000 categories of merchandise (eBay, 2007). Although electronic negotiation is still in its infancy, there has been a great deal of research that has accompanied it.

Several attempts have been made to define electronic negotiation, and also to define the characteristics that are desirable in electronic negotiations (Rosenschein, 1994; Sandholm, 1999; Lomuscio, Wooldridge et al., 2003). These characteristics include

- *Computational efficiency*: A negotiation mechanism must be computationally efficient.

- *Communication efficiency*: All things being equal, it would be beneficial to have a negotiation mechanism that enables communication among the agents in an efficient way.

- *Individual rationality*: Each individual involved in the negotiation should act rationally and it should be in an individual's best interest to participate in the negotiation. Also, if the utility of a group of individuals should be taken into consideration, the group utility can be obtained by the component of each agent's personal utility.

- *Distribution of computation*: Mechanisms that distribute the computation over the agents involved are preferable to ones in which one server is performing all the computation for the whole system. This is preferred for many reasons, including

the desire to avoid performance bottlenecks and the disruptive effects of a single point of failure.

- Pareto efficiency: A Pareto efficient outcome is one where there is no other possible outcome that could be beneficial to at least one agent without negatively affecting the other agent.

Agents play a key part in electronic negotiation. In the following sections, we will define a negotiation agent and present models in negotiation agents.

2.3.1 Electronic Agents

The emergence of e-commerce has led to the design of online auction programs (Guttman, 1999; Bansal, 2005) as well as software agents capable of negotiating based on several criteria (Lee, Chang et al. 2000; Bichler, Kersten et al. 2003). In general, auctions help establish efficient markets when the goal is to obtain the best price (Beam, 1999), but when the goal is to establish the terms of a transaction, features of a product or service, then negotiation is considered to be advantageous (Gordon Lo, 1999). In negotiation, the agents engage in an iterative and alternating process of bids and offers over time until they converge to a single price acceptable to both.

In order to understand the importance of electronic negotiation agents, we must first understand the definition. The term "agent" has been the subject of much recent debate. Several definitions allow the inclusion of almost all possible objects; others only allow only a limited scope for agents (Faratin, 2000). However, in general, an electronic negotiation agent can be defined as a software-based computer system that can perform certain tasks on behalf of their users. In order for an electronic negotiation agent to be

successful, it must have the following properties. (Guttman, 1999; Cardoso and Oliveira, 2001; Bichler, Kersten et al., 2003; Strobel and Weinhardt, 2003).

- *Autonomous*: agent must be able to make decision by themselves, autonomously, without any direct intervention.
- *Reactive*: agents must be able to understand their environment and respond quickly to any changes.
- *Cooperative*: agents must have a communication interface to interact with other agents or people.
- *Learning*: agents are able to understand the user's preferences and performances as they interact with other agents or humans, so that they can improve the performance over time.
- *Proactive*: Agents are able to act in anticipation to maximize their utility.

Although the above are general properties of agents, there are variations, depending on the task on hand. The next section will deal with the modeling approaches of electronic agents in the context of electronic negotiations.

2.3.2 Modeling Approaches

There are two major modeling approaches for electronic agents: non-learning such as game theoretic methods, and learning approaches such as machine learning and heuristic based methods.

Non-Learning Based Approach: Game-Theoretic Models

Game theory is a branch of economics (Nash, 1950) that provides a formal framework of rational decision making in strategic situations. In a game theory based negotiation, each negotiator must first rank his/her preference for each possible outcome. Each individual must then take into account what the other is likely to do and act accordingly in order to achieve his/her preferred outcome. This formal framework provides clear analyses of various situations and precise results concerning the strategy that a negotiator should use. However, it uses several restrictive assumptions that make it hard to use in real-world negotiations (Wilkenfeld, 1992).

- *Bilateral Negotiation*: Even if multiple agents are present in the negotiation setting, no more than two agents need the same resource.
- *Full Information*: Each agent is aware of all information including the other agent's utility for all possible outcomes over time.
- *Rationality*: All agents behave rationally; each agent attempts to maximize its utility.
- *Commitments Are Kept*: If an agreement is reached, both agents will honour it.
- *No Long Term Commitments*: Each honoured outcome stands alone. An agent cannot commit itself to any future activity other than the current situation.
- *Resource Division Possibilities*: It is assumed that all resources are divisible.
- *No Other Options*: No other possibilities or alternatives to the current negotiation exist. The negotiation must continue until an agreement is reached.

Due to these limiting assumptions, it is often unacceptable to model negotiation behaviour based on game theory alone. However, game-theoretic tactics have been found

useful when applied to negotiation scenarios within the following two key areas (Jennings, Faratin et al., 2001):

1. Game theory is useful to design the appropriate protocol that will govern the interactions between the negotiation participants. In general, a protocol defines the "rules of encounter" between agents. This allows the formation of a framework, or protocol that sets specific constraints and bounds that the negotiation participants are allowed to make.

2. Agents can use game-theoretic models as a benchmark to validate that their current strategy is in their best interest (i.e. utility maximization).

One of the main difficulties with the second point is that the utility maximization problem is difficult, somewhat dampening the effectiveness of this application (Jennings, Faratin et al., 2001). However, several attempts have been made to apply game-theoretic techniques for artificial intelligence purposes, using relaxations to the underlying assumptions (Harsanyi, 1972; Genesereth, 1986; Wilkenfeld, 1992).

Learning-Based Approach: Heuristics and Machine Learning

Another modeling approach is the use of learning based methods including heuristic and machine learning. These methods allow for the correction of the several shortcomings associated with game-theoretic approaches. One of the major advantages is the acknowledgement that although heuristic approaches do not employ the (game-theoretic) optimal solution, the associated computational complexity is often significantly lower (Jennings, Faratin et al., 2001). In many case, the models may be approximation to

the game-theoretic techniques (Genesereth, 1986) or computational models loosely based on the models of negotiation behaviour previously presented (Pruitt, 1981; Raiffa, 1982). These models include (Wilkenfeld, 1992; Faratin, Sierra et al. 1998; Zeng, 1998; Deveaux, 2001; Mok and Sundarraj, 2005; Shi, 2005). In each case, the models attempt to learn from their opponents' behaviour in order to improve their negotiation outcome, which is not the case with Game-Theoretic Models.

Other advantages attributed to heuristics include its realism, since according to research (Pruitt, 1981), people generally tend to base their negotiation strategies on simple heuristics.

2.3.3 Current Applications

In human negotiations, two or more parties bargain with one another to determine the price or other transaction terms (Fisher, 1981). In an automated negotiation, software agents engage in broadly similar processes to achieve the same end (Jennings, Faratin et al., 2001). As previously explained, autonomous electronic negotiation agents have several key properties that enable them to negotiate without much human intervention. This makes them extremely important in several areas of e-commerce, since it allows for the reduction of the costly human component. These advantages have been the major drive in the development of negotiation agents for commercial purposes.

A good example of a practical software agent is AuctionBot. The Michigan Internet AuctionBot is a flexible, scalable, and robust auction server that supports both software and human agents (Wurman, 1998). Although this system is currently not in use for commercial applications, it has been successfully used to create an online market for used

textbooks. The server is capable of managing many simultaneous auctions by separating the interface from the core auction procedures. This clever design provides a responsive interface and tolerates system and network disruptions, but necessitates careful timekeeping procedures to ensure temporal accuracy (Wurman, 1998). It also enables users a web-based access to their accounts, and presents an organized view of their bids, the auctions in which they are currently involved, and their past transactions

Several other negotiation systems have also been developed in order to exploit the benefits of electronic agents in a commercial setting; MIT Kasbah marketplace is one of them. Kasbah is an electronic marketplace in which users can create an autonomous agent capable of buying or selling a product (Chavez, 1996). The agent configuration includes some behaviour rules, including the maximum time allowance for the negotiation, the desired price interval and the price suggestion function (de Paula, Ramos et al., 2001). The price suggestion function can be chosen as a linear, quadratic or cubic with respect to time. Unlike AuctionBot, Kasbah is able to perform Merchant Brokering as well as negotiation (Guttman, 2000). The buyer or seller is therefore in control of the desired negotiation strategy, and hence, the concession rate of the electronic agent. This freedom allows an individual to select the negotiation parameters as he/she sees fit, and allows him/her to properly leverage the agent.

Another similar negotiation system, also developed by the MIT Media Lab, is Tête-À-Tête. Unlike Kasbah which is aimed at individuals (Guttman, 2000), this system is geared at retail sales. Also unlike Kasbah and AuctionBot, Tête-À-Tête has the ability to function as a product brokering agent. Product Brokering allows the retrieval of information to help determine what to buy. This encompasses the evaluation of product alternatives based on consumer-provided criteria (Guttman, 2000). Since Tête-À-Tête

deals with the retail sales, unlike most other agents that generally only compete over price, it co-operatively negotiates across multiple terms of a transaction (Wang, 2004), making it extremely useful in commercial settings. The "shopping agent" follows an argumentative style of negotiation with its agents and uses the previously defined evaluation constraints provided by the user in the product brokering stages as dimensions of a multi-attribute utility function (Wang, 2004). The utility associated with the current customer's position is then used to rank merchant offers correspondingly.

In all, several attempts have been made in order to integrate electronic negotiation and auction agents in commercial settings. The first international Trading Agent Competition in 2000 challenged its entrants to design an automated trading agent that was capable of bidding in simultaneous online auctions for complementary and substitutable goods (Stone, 2005). Twenty-two entrants from around the world competed in ten games. Agents were compared on several criteria including their bidding strategy, allocation strategy (i.e. weight given to negotiation factor such as price or quality), special approaches, and team motivations. The large involvement of the teams and the apparent success of the competition provide a strong indication of where the future of e-commerce lies; in the hands of electronic auction and negotiation agents.

Another main application of electronic negotiation could be its implementation in the practice of dynamic pricing. Dynamic pricing refers to the charging of different prices for different quantities of a product, at different times, to different customer groups or in different markets, when these price differences are not justified by cost differences (Salvatore, 2001). It is believed that one of the major changes that will be brought about by agent-mediated e-commerce is that dynamic pricing and personalization of offers will become the norm for many goods and customers (He, Jennings et al. 2003). Online

consumers differ in their purchasing preferences and, therefore, a seller's profit can be increased by charging two different prices for the same good from price-insensitive and price-sensitive consumers (Dasgupta, 2003). A good example of dynamic pricing is given by the operations of the company priceline.com. Priceline allows buyers to name the price they are willing to pay for flights, hotels and mortgages, cars and groceries (Salvatore, 2001). If their price bid is not expectable, Priceline will either reject their bid, or provide a less costly alternative that meets their price needs. This pricing technique was previously difficult to implement due to the fact that it would involve too much labour capital in order to set individual prices for each customer. Now with the pervasiveness of internet access, and the induction of electronic negotiation agents, prices can be individually set according to the maximum price each customer is willing to pay; ultimately allowing retailers to charge specific consumers more for goods and services than the price at which they would initially be marketed.

2.4 Our contribution

So far, we have discussed how heuristic techniques often resemble human behaviour, and as a result, are used in a number of electronic negotiation systems. Mathematical models based on these techniques are scarce and are just beginning to emerge.

In this thesis, we assume that the only issue in the negotiation is *price* and that a negotiator makes price-offers by following the Time-dependent tactic (TDT), introduced in section 1.2 and formulated mathematically in the next chapter. The agent then seeks to estimate (i.e., *learn*) the underlying mathematical parameters of the negotiator's TDT

model, by knowing only the price-offers received thus far in the negotiation. Our goal in the thesis is to determine robust and flexible algorithms for this learning problem. Note that the learning can be used to improve the outcome of the agent, although this aspect is outside the scope of this thesis.

The use of classical optimization techniques for the aforementioned problem has been given by Shi (2005). Shi considers a mathematical functional form of a common negotiation heuristic, known as the time-dependent tactic, that has been incorporated in a number of electronic negotiation systems. As mentioned in section 1.2, he uses optimization to estimate the parameters of the function, based on the increasing price offers received from his opponent. While such an approach was not proposed before, Shi's results suggests the potential for improvement, in terms of algorithmic convergence and accuracy of parameter estimation. Thus, this thesis will tackle the same problem using different line search and nonlinear least squares methods than those previously attempted. The solution approach applied in this paper uses a more complex set of optimization algorithms, namely quasi-Newton algorithms, in order to estimate a negotiator's underlying TDT parameters. This solution approach was chosen since it does not make the assumptions that are inherent with the Gauss-Newton based methods previously applied. These assumptions may have resulted in lower rate of convergence and poorer parameter estimation accuracy, and thus motivate this thesis.

Chapter 3 Problem Formulation

3.1 Introduction

The current chapter deals with the formulation of the time dependent tactic presented by Faratin (Faratin, Sierra et al., 1998). We formulate the learning problem as a nonlinear least squares model. These methods will be presented as well as potential solution approaches using nonlinear programming.

3.2. Time-dependent tactics functions

As previously explained in section 2.1.2, time plays an extremely important role in the concession rate as well as the final outcome of the negotiation. Therefore, the use of the time dependent tactic function seems to be a natural selection when it comes to model negotiations. Several different time-dependent tactical functions have been proposed for use in negotiation agents. See references for more detail (Wilkenfeld, 1992; Faratin, Sierra et al., 1998; Deveaux 2001; Da-Jun and Liang-Xian, 2002; Fatima, Wooldridge et al., 2002; Papamichail and Papamichail, 2003; Mok and Sundarraj, 2005).

In a negotiation process, several factors can affect the concession rates of the participants. However, time pressure is the most relevant. Levels of demand and concession rates are functions of time pressure and the amount of time that has elapsed since the beginning of the negotiation (Pruitt, 1981).

The time dependent tactic presented in this thesis is based on human negotiation behaviour, which can generally be divided into two categories; boulware and conceder behaviour (Pruitt, 1981; Raiffa, 1982).

An agent that exhibits boulware behaviour concedes slowly at the beginning of the negotiation, only incrementing the price by a small amount at each turn of the negotiation. As the time in the negotiation begins to reach the maximum allowable time, the negotiator begins to concede more rapidly. This can be explained by the cost associated with not reaching an agreement in the deal. A good example of this behaviour is a contractor that needs to purchase lumber in order to build houses. If the contractor has already agreed to build the houses, he/she must purchase wood in order to begin the construction. The contractor must reach an agreement in time with the lumber seller (or sellers) in order to build the houses. If not, the contractor may have to break the contracts, loose the chances of gaining profits, and possibly incur a fine for not meeting the contract provisions. At the beginning of the negotiation, the contractor may attempt to only concede slowly due to financial constraints, or possibly to avoid image loss. However, as the negotiation progresses, the cost associated with not reaching an agreement becomes apparent, the contractor begins to concede more rapidly in order to avoid costly contract infringement.

The other type of negotiation behaviour is conceder behaviour. A negotiator exhibiting conceder behaviour concedes rapidly at the beginning of the negotiation and reaches his/her reservation price early on in the bargaining process. This behaviour is often exhibited by negotiators that are pressed for time and want to resolve the negotiation rapidly. Also, rapid concessions at the beginning of the negotiation encourage the opponent to stay in the negotiation, and limits the risk that an agreement will not be reached. Figure 3.1 below plots both negotiation behaviours of a buyer as a function of the number of turns in the negotiation.

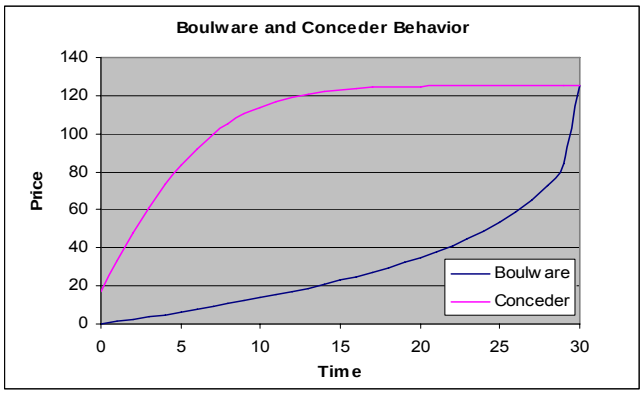

Figure 3.1 Boulware and Conceder Behaviors Price Offers of a Buyer

The next section will deal with the time-dependent tactic function proposed by Faratin.

3.3 The Faratin Model

Based on the negotiation behaviour models proposed in section 3.2, Faratin proposed a mathematical model that incorporates and quantifies these models as well as their assumptions. See Equations 3-1, 3-2.

$$P(t) = P_{\min} + e^{\left(1-\frac{t}{T_{mac}}\right)^{\beta} \ln(k)} \left(P_{\max} - P_{\min}\right) \qquad \text{Buyer} \qquad (3\text{-}1)$$

$$P(t) = P_{\min} + \left(1 - e^{\left(1-\frac{t}{T_{\max}}\right)^{\beta} \ln(k)}\right)\left(P_{\max} - P_{\min}\right) \qquad \text{Seller} \qquad (3\text{-}2)$$

where

t is time (number of turns) in the interval $[0,T_{max}]$
$P(t)$ is the value of the negotiating issue proposed by an agent at time t
P_{min} is the minimum price value
P_{max} is the maximum price value
k is a constant that determines the value of the price in the first offer
T_{max} is the time limit (maximum number of turns) proposed by both agents

β is a constant that determines the degree of convexity of the negotiation function

It must be noted that for the buyer (eqn 3-1), the utility decreases as the price rises (or number of turns increases) and for the seller (eqn 3-2), the utility rises as the price increases (or number of turns increases). This is to be expected, since price is the sole issue determining utility in this set of equations.

In order to fully understand the significance of the Faratin function, we must take a close look at each of the five parameters since their values define the negotiation behaviour used by the agent. The interval of P_{min} and P_{max} defines the price range that the agent is willing to negotiate within. If an agreement is not found within the bound of T_{max}, the negotiation will end. The value of k, given a previously defined interval of $P_{min,}$ and $P_{max,}$, determines the first offer in the negotiation. This can be demonstrated by setting the value of t in equation 3-1 to zero. The resulting formula $P(0) = P_{min} + k (P_{max} - P_{min})$ demonstrates that the value of k must be in the interval of [0,1], where a higher value of k will result in a larger initial offer and a smaller value of k will result in a smaller offer. The value of β determines the convexity of the function. A value of β in the interval of [0,1] results in boulware behaviour, whereas, a value larger than one results in conceder behaviour. A plot of the Faratin buyer model (eqn 3-1) as a function of β is given below.

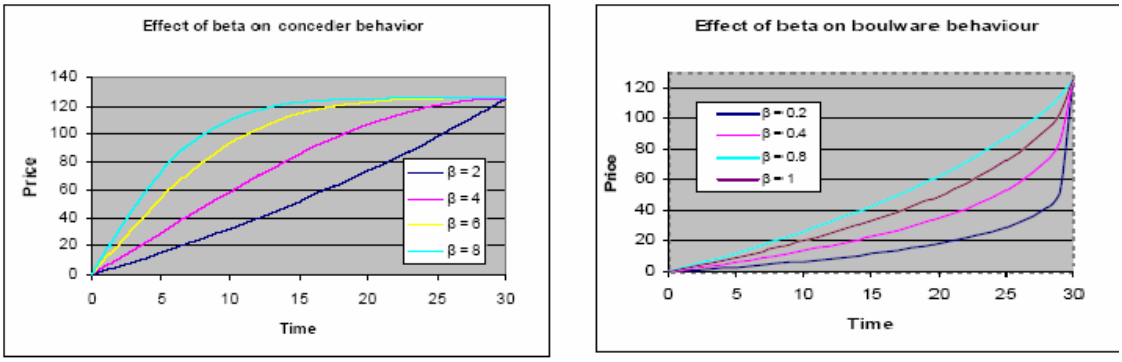

Figure 3.2 Effect of β on the Faratin Buyer model

The next section deals with the formulation of the Faratin model as a nonlinear least squares problem.

3.4 Nonlinear least squares optimization

The ultimate goal of formulating the Faratin buying agent function (eqn 3-1) into a nonlinear least squares function is to obtain an estimate of the parameters [$P_{\min}$, $P_{\max}$, $T_{\max}$, β, k]. The benefit of having the function in this form is the current availability of robust techniques that can be used to solve these equations.

3.4.1 Learning as a Nonlinear Least Squares Problem

To understand the formulation of the learning problem (Shi, 2005), consider the following nonlinear least-squares problem (Fletcher and Xu, 1987):

Find a local minimum x*

$$f(x) = \frac{1}{2}\sum_{i=1}^{n}[r_i(x)]^2 = \frac{1}{2}r(x)^T r(x) \qquad \left(x \in R^n\right) , \tag{3-3}$$

where $r_i(x)$ is a residual term resulting from the difference of $f_i(x^*)$ and $f_i(x)$. Redefining equation 3-1 gives:

$$\hat{P}_t = f(t, \theta), \tag{3-4}$$

where $\theta = [P_{min}, P_{max}, T_{max}, \beta, k]$ is a parameter vector of the five parameters in equation 3-1. If we then substitute the value of the residual r(x) in equation 3-3:

$$r_i(\theta) = P_t - f(\theta, t), \tag{3-5}$$

where P_t is the actual price given by the opponent of the negotiation at time t, we obtain:

$$f(x) = \frac{1}{2} \sum_{i=1}^{n} [P_t - f(\theta, t)]^2 \tag{3-6}$$

Thus, the minimization of equation 3-6 by using nonlinear optimization techniques can provide us with a way to obtain an estimation of parameter vector θ. Therefore, learning the parameters of the time-dependent tactic function is now in the form of an unconstrained nonlinear optimization model as a function of time. We solve this optimization problem once five (i.e., t = 5) offers have been made.

The effectiveness of each solution approach, in terms of price estimation, can then be tested using the following sum of squared errors function. While other metrics can be used, SSE has is generally considered superior owing to its properties (see Shi, 2005 for more details).

$$SSE(\theta) = \sum_t \left[P_t - \hat{P}_t \right]^2 = \sum_t \left[P_t - f(t, \theta) \right]^2 \tag{3-7}$$

The next chapter will deal with the various methods used to solve nonlinear optimization problems and their relevance to the work proposed in this thesis.

Chapter 4 Solution Approach

When attempting to solve a nonlinear optimization problem, several considerations need to be made before an appropriate method is chosen. Methods vary greatly in terms of their computational complexity, accuracy and effectiveness. Also, a method that performs well on one function may perform poorly on another function. It is therefore necessary to analyse algorithms thoroughly, before making recommendations for real-world implementation. In general, nonlinear optimization techniques can be broken down into two major categories: *invariable (or exact) methods*, or *numerical approximations*. Exact solution methods can be beneficial since they provide an optimal solution to the problem at hand. With this class of methods, the first partial derivative of the function (eqn 3-7) with respect to each of the five parameters needs be obtained at each time interval. Each of these derivatives then needs to be set to zero and then the system of equations needs to be solved at each time interval. However, when the function under consideration is large and cumbersome (as is the case with eqn 3-1), it would require too much computation in order to obtain a solution in the time frame provided in an electronic negotiation. Therefore, in contrast to exact methods, numerical approximation methods are generally used to solve nonlinear optimization problems when the function provided appears to be computationally complex, assuming that the approximations obtained are accurate and fast enough.

Several numerical methods are available that all provide various results based on the function being solved. The prominent ones, shown in the figure below (Scales, 1985), can be classified into small residual and large residual types.

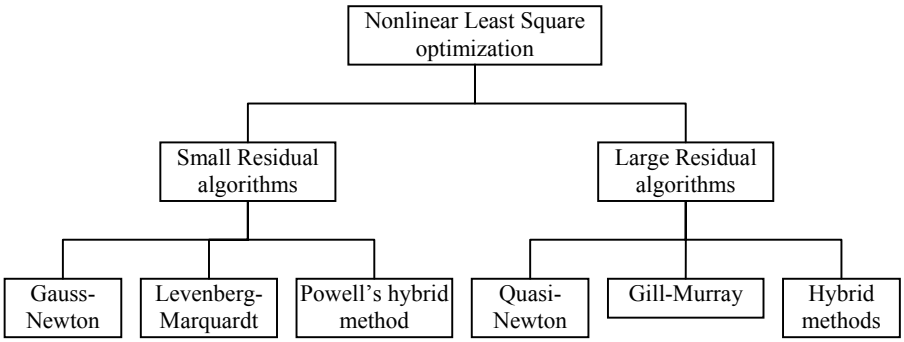

Figure 4.1 Numerical methods for solving nonlinear Least Squares problems

Small residual algorithms can be seen as methods that ignore the residual term in nonlinear least-squares function. This greatly simplifies the computational complexity involved, since the Hessian matrix is not calculated; the Hessian is simply ignored. In contrast, large residual algorithms, the residual term in nonlinear least square function is approximated. This method involves much more computational power since the residual term involving the Hessian matrix is often quite complex.

In the next section, the framework of quasi-Newton methods will be developed, and its relevance to this work explained.

4.1 Framework for Algorithms

For general unconstrained minimization problem where the Hessian matrix is available or computable, Newton's method can be used with great accuracy (Yabe and Takahashi, 1991). This method constructs a sequence of vectors θ_i such that:

$$\theta_{i+1} = \theta_i + \alpha_i d_i , \tag{4-1}$$

where α_i is a scalar steplength and d_i is the direction of the search that satisfies the Newton equation 4-2:

$$\nabla^2 f(\theta_i) d = -\nabla f(\theta_i) \tag{4-2}$$

For sum of squares of nonlinear functions, the gradient vector and Hessian matrix have special forms that are respectively given by:

$$\nabla f(\theta_i) = J(\theta_i)^T r_i(\theta_i), \text{ and} \tag{4-3}$$

$$\nabla^2 f(\theta_i) = J(\theta_i)^T J(\theta_i) + \sum r_i(\theta_i) \nabla^2 r_i(\theta_i), \tag{4-4}$$

where $r(\theta)$, the residual term, is given by (3-5), and $J(\theta_i)$, the Jacobian of (3-1) is as follows (recall from Chapter 3 that *n* is the number of offers received):

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1(t,\boldsymbol{\theta})}{\partial P_{\min}} & \frac{\partial f_1(t,\boldsymbol{\theta})}{\partial P_{\max}} & \frac{\partial f_1(t,\boldsymbol{\theta})}{\partial T_{\max}} & \frac{\partial f_1(t,\boldsymbol{\theta})}{\partial \beta} & \frac{\partial f_1(t,\boldsymbol{\theta})}{\partial k} \\ \vdots & & & & \vdots \\ \frac{\partial f_n(t,\boldsymbol{\theta})}{\partial P_{\min}} & \frac{\partial f_n(t,\boldsymbol{\theta})}{\partial P_{\max}} & \frac{\partial f_n(t,\boldsymbol{\theta})}{\partial T_{\max}} & \frac{\partial f_n(t,\boldsymbol{\theta})}{\partial \beta} & \frac{\partial f_n(t,\boldsymbol{\theta})}{\partial k} \end{bmatrix}$$

Therefore, the method of solving for the decent direction using Newton's method can be presented in the following form:

$$d = -\left(J(\theta_i)^T J(\theta_i) + \sum_{i=1}^{T} r_i(\theta_i) \nabla^2 r_i(\theta_i) \right)^{-1} \left(J^T(\theta_i) r(\theta_i) \right) \tag{4-5}$$

For the Faratin function (3-1), the above expression is computationally complex to evaluate. It is in such cases that quasi-Newton methods are recommended (Scales, 1985). In order to simplify the notation in equation 4-4, the following equation is presented:

$$\nabla^2 f(\theta_i) = J(\theta_i)^T J(\theta_i) + \sum r_i(\theta_i) \nabla^2 r_i(\theta_i) = J(\theta_i)^T J(\theta_i) + A(\theta_i) \,, \tag{4-6}$$

where $A(\theta_i)$ is an approximation as given below given below.

$$A(\theta_i) \approx \sum_{i=1}^{n} r_i(\theta_i) \nabla^2 r_i(\theta_i) \tag{4-7}$$

Therefore, the ultimate goal of quasi-Newton methods is to find a way to calculate the term $A(\theta_i)$ using a minimal amount of computational power, while obtaining close approximations. The next section expands on this framework.

4.2 Algorithmic Details

The quasi-Newton framework calls for computation of a proper descent direction based on an approximation of equation 4-7. Of the several methods that have been proposed for this purpose, we shall employ two prominent ones, namely, the structured and factorized methods. An overview of these methods is given in section 4.2.1.

Further, as given in (eqn 4-1), one will have to compute an appropriate step size α as well as a suitable starting point θ_0. Algorithms used for these two steps are given in section 4.2.2 and 4.2.3, respectively.

4.2.1 Structured and Factorized Quasi-Newton Methods

The two main methods that will be examined in this work are Structured and Factorized quasi-Newton Methods. Structured methods including the generalized quasi-Newton method (Luksan, 1996), Bartholomew-Biggs (Bartholomew-Biggs, 1977) and the Dennis-Walsh-Gay (DWG) method (Dennis, 1981) are all robust algorithms for large

and small residual problems (Yabe and Takahashi, 1991). The update formulas below are adapted from Yabe and Takahashi (1991).

A generalized update scheme for A is given by the following equation (θ: is suppressed for convenience)

$$A_{i+1} = A_i + \left(\frac{1}{v_i^T s_i} \right) (y_i - A_i s_i) v_i^T \tag{4-8}$$

where

$$v_i = (J_{i+1} - J_i)^T r_{i+1} \tag{4-9}$$

$$s_i = \theta_{i+1} - \theta_i \tag{4-10}$$

$$y_i = \nabla f(\theta_{i+1}) - \nabla f(\theta_i) \tag{4-11}$$

The Bartholomew-Biggs update is given by the following equation:

$$A_{i+1} = \beta_i A_i + \frac{(v_i - \beta_i A_i s_i)(v_i - \beta_i A_i s_i)^T}{(v_i - \beta_i A_i s_i)^T s_i} \tag{4-12}$$

$$\beta_i = \frac{r^T{}_{i+1} r_i}{r_i^T r_i} \tag{4-13}$$

The Dennis, Gay and Welsh (DWG) update is given by the following:

$$A_{i+1} = \beta_i A_i + \frac{(v_i - \beta_i A_i s_i) y_i^T + y_i (v_i - \beta_i A_i s_i)^T}{s_i^T y_i} - \frac{s_i (v_i - \beta_i A_i s_i)}{(s_i^T y_i)^2} y_i y_i^T \tag{4-14}$$

$$\beta_i = \min\left(\left| \frac{s_i^T v_i}{s_i^T A_i s_i} \right| , 1 \right) , \tag{4-15}$$

where

$$r_i = P_t - f(\theta_i, t) \tag{4-16}$$

In general, for quasi-Newton methods, it is desirable for the approximation matrix parameter to be positive definite. This is to ensure that a descent direction for the objective function is obtained at each iteration. However, for structured quasi-Newton methods, it is not clear how to construct an updating formula for $A(\theta_{i+1})$ such that the matrix $J_i^T J_i + A(\theta_i)$ is always positive definite (Yabe and Takahashi,1991). In order to overcome this difficulty, factorized quasi-Newton methods have been proposed, including the use of modified Cholesky and QR decomposition (Bjorke, 1996). Once the matrix has been decomposed using either technique, the problem has the following form (Yabe and Takahashi, 1991):

$$\left(J_i + L_i\right)^T \left(J_i + L_i\right) d_i = -J_i^T r_i \quad , \tag{4-17}$$

where the matrix L_i is an m x n correction matrix to the Jacobian matrix such that $L_i^T J_i + J_i L_i + L_i^T L_i$ is the i-th approximation to the second part of the Hessian matrix of (4-5).

The next section deals with the line search algorithms used in order to ensure a proper step size in the descent direction.

4.2.2 Selecting the Step-size Parameter

Several methods are available to estimate the optimal step size. Since this estimation needs to be calculated at each iteration, it is important to have a method that does not involve too much computational complexity, yet provides a near optimal solution. In order to gain a full understanding of the line search algorithms proposed, it is beneficial to have another look at equation 4-1.

$$\theta_{i+1} = \theta_i + \alpha_i d_i \tag{4-18}$$

Given that the previous value of θ_i has been calculated and the descent direction was obtained using equation 4-5, a value of α_i must be selected at each iteration in order to ensure an optimal step size. Several line search algorithms have been proposed by previous researchers. One algorithm proposed by Shi (2005), referred to in this paper as general step size parameter selection, was to set the value of α_i in equation 4-1 to 1. When the descent direction was calculated and the new parameter values of θ_i were obtained, only the parameters that were found within a predefined bound were updated, and the other parameters were left unaltered. This approach to a line search algorithm may be problematic, since it involves a deviation from the optimal direction of descent. In order to avoid this issue, an attempt was made to use all of the updated parameters, while varying the value of α_i to ensure that the step size in the descent direction was optimal to equation 3-7. We employ the Golden section search algorithm and another algorithm, known as "backaway" algorithm, that aims to keep the updated parameter values within certain feasible limits.

Golden Section Search Algorithm

The Golden Section Search (Scales, 1985) is a robust algorithm that can achieve results with sufficient accuracy in a minimal amount of iterations. This algorithm is a well-know univariate optimization method that allows for a minimum value to be determined by constantly reducing the search interval by a factor $\tau \approx 0.6180$, where τ satisfies the following quadratic equation.

$$\tau^2 + \tau - 1 = 0 \tag{4-19}$$

The Golden section search is illustrated in the following diagram.

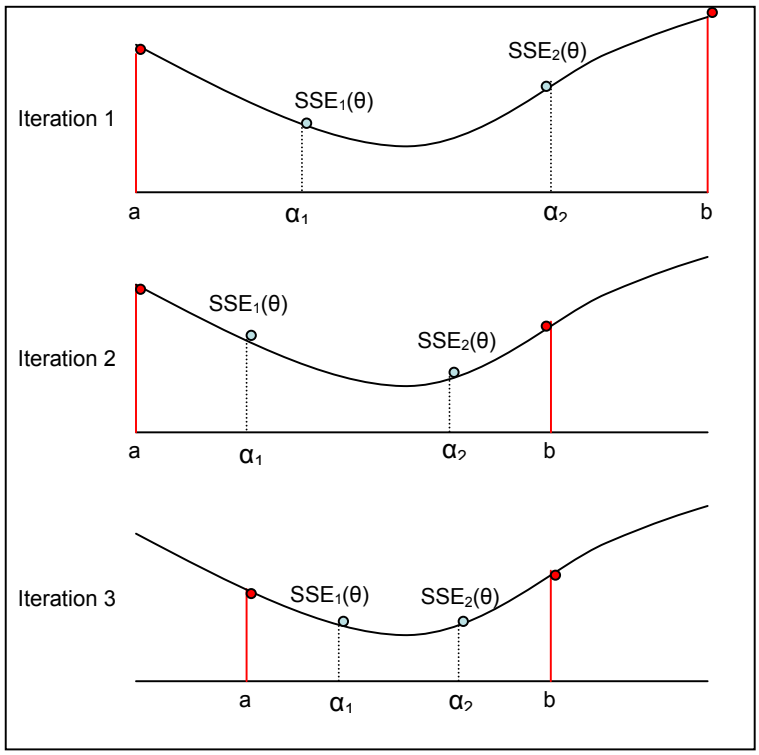

Figure 4.2 Golden Section Search algorithm

The search section starts with two points *a* and *b,* and is then progressively reduced at each iteration by a factor of τ. The traditional Golden section search is for univariate optimization, and so, in order to adapt it to our case, equation 3-7 was modified to take the following form.

$$SSE(\theta) = \sum_{t}\left[\hat{P}_t - f(t, \theta_* = \theta + \alpha * d)\right]^2, \qquad (4\text{-}20)$$

Therefore, the optimal step size was selected by finding a value of α that minimizes (4-20) and hence, minimizes the sum of squared error between the actual and estimated time dependent negotiation function.

The modified algorithm can then be given by the pseudo code in Table 4.1.

```
Initialize  a and b, to 0 and 1;
            τ to 0.6180
          α1= a +  τ* ( b-a)
          α2= a +  τ* ( b-a)
            SSE1(θ, α1 ); SSE2(θ, α2 )

       While b-a > tolerance
         If ( SSE1(θ) > SSE2(θ)  )

              then:  a = α1
                     α1 = α2 ; SSE1(θ)  = SSE2(θ)
                     α2 = a + τ ( b-a)
                     SSE2(θ)  = SSE(θ) (α2)

         Else (SSE1(θ) < SSE2(θ)  )

              Then:  b = α2
                     α2 = α1 ; SSE2(θ)  = SSE1(θ)
                     α1 = a + (1-τ )( b-a)
                     SSE1(θ) = SSE(θ) (α1)
        End If
       End While
```

Table 4.1 Pseudo-code for Golden Section Search

Back-away algorithm

In order to ensure that all the parameters were used to update the value of θ_i, it was necessary to derive a method that would allow the direction of deepest decent to be followed while ensuring that all parameters remained within their predefined bounds. The pseudo code for the proposed algorithm is presented below in Table 4.2.

$\theta_{i+1} = \theta_i + \alpha_i d_i$
θ_L are the lower bound parameters
θ_U are the upper bound parameters

For j=1 to.m

$$\theta_1(j) = \theta_0(j) + \alpha_0(j) * d_0$$

Initialize k = 1

While ($\theta_{i+1}(j) > \theta_U(j)$ **OR** $\theta_{i+1}(j) < \theta_L(j)$

then $\alpha_{i+1}(j) = \alpha_i(j)$ **/ k**

k = k ^{2}

until ($\theta_{i+1}(j) > \theta_U(j)$ **OR** $\theta_{i+1}(j) < \theta_L(j)$

End While

Table 4.2 Pseudo-code for the back-away algorithm

As a brief explanation, if a given parameter was not within the pre-defined bounds, the step size was divided by k (set to 2 for the first iteration). The squared value of k was increased until the updated step size was within with the lower and upper bounds. This allowed decent direction to be maintained while assuring that the parameters remained within their pre-defined bounds.

4.2.3 Pre-processing

In order to ensure a successful attempt is made to locate a minimum value of a function, it is often necessary to begin with a proper starting point. We refer to this setup as pre-processing.

Two main forms of pre-processing are commonly used before the implementation of a Gauss-Newton or quasi-Newton algorithms; Pattern search and Exploratory search. The Hooke-Jeeves algorithm is a hybrid search method that applies aspects of both forms of search without the need for either first or second order derivative information. Hence,

it serves as a robust method to achieve a suitable starting point before the implementation of a Newton-based method.

Hooke-Jeeves

The Hooke-Jeeves algorithm is implemented by defining an initial starting point $\theta_{0,}$ a step size δ, and a set of n orthogonal unit vectors e_i. The first part of the algorithm is the implementation of the exploratory search. Starting from the initial point θ_0, each coordinate direction, j, is explored using the following equation

$$\theta_{i,j_{+1}} = \theta_{i,j} + \delta * e_i \tag{4-21}$$

The functional value of $\theta_{i,\ j+1}$ is then compared to that corresponding to $\theta_{i,\ j}$ in the function being evaluated. If $f\left(\theta_{i,j+1}\right) < f\left(\theta_{i,j}\right)$, then point $\theta_{i,\ j+1}$ is accepted; if the reverse holds, the same increment is subtracted.

$$\theta_{i,j_{+1}} = \theta_{i,j} - \delta * e_i \tag{4-22}$$

In either case, the new value of $\theta_{i,j+1}$ is accepted as long as the function under evaluation has improved. If no improvement is found in the search in either direction, the initial value of $\theta_{i,j}$ will remain unchanged. This procedure will repeat until all of the m search directions have been explored.

The next step in the Hooke-Jeeves algorithm is the implementation of the pattern search. The pattern search first compares the values of the function under evaluation using the starting point, and the resulting point, θ_1, from the exploratory search. If

$f(\theta_1) < f(\theta_0)$, then the pattern search is made along the direction presented in the equation below.

$$y_0 = \theta_1 + \alpha(\theta_1 - \theta_0), \tag{4-23}$$

where, α is the acceleration factor used to size the magnitude of the pattern search. The procedure of the exploratory search is then repeated starting with the new point obtained.

However, if the new found point does not satisfy $f(\theta_1) < f(\theta_0)$, the step magnitude δ is reduced by a factor of *div* using the following equation.

$$\delta = \delta / div \tag{4-24}$$

The Hooke-Jeeves algorithm then continues and concludes when δ is smaller than a predefined small value *tol*.

In terms of the current work, the Hooke-Jeeves algorithm is used as a pre-processor to the initial starting point. This step is then followed by a Newton-based algorithm to find the descent direction. The step size in the decent direction is then defined by the value obtained in the from the parameter selection.

Chapter 5 Experimental Design

5.1 Parameter Generation

In order to measure the effectiveness of a given combination of optimization techniques, a uniform set of starting points and actual points is needed. In the current thesis, a set of 200 starting points and 200 actual points (corresponding to the negotation parameters in θ) are generated using the following equation.

$$\theta(i)=LowerBound(i)+rand(i)*(UpperBound(i)-LowerBound(i)), \quad (5\text{-}1)$$

where rand(i) is a random value within the interval of (0,1). The values of the lower and upper bounds of each parameter are given in the following table.

Parameter	Range
P_{min}	[100, 250]
P_{max}	[300, 600]
T_{max}	[20, 40]
β	[0, 10]
k	[0,1]

Table 5.1 Upper and Lower bounds of test parameters

Without loss of generality, since the current thesis deals with negotiation from the buyer's perspective, we use equation 3-1 to generate bidding prices.

$$P(t) = P_{\min} + e^{\left(1-\frac{t}{T_{mac}}\right)^{\beta}\ln(k)}\left(P_{\max} - P_{\min}\right) \quad (5\text{-}2)$$

The actual parameters are used to generate an array of prices that represent the actual price-curve of the buyer. The number of price-offers generated varies from 4 to 10; see also section 5.3. The starting parameters are applied to equation 5-2 to generate

the starting prices of the buyer. The performance of the algorithm is monitored by using equation 3-7.

$$SSE(\theta) = \sum_t \left[P_t - \hat{P}_t \right]^2 = \sum_t \left[P_t - f\left(t, \theta\right) \right]^2 , \tag{5-3}$$

where P_t is the actual price at time t and $f(t, \theta)$ is the starting price at time t. Since the price vector of the actual prices remains constant, the change in the value of the standard square error is only a function of the changing value of the estimated parameters.

The minimization process is illustrated by the following figure, where SSE_0 through SSE_2 determine the sum of squared error for iteration 0 through 2.

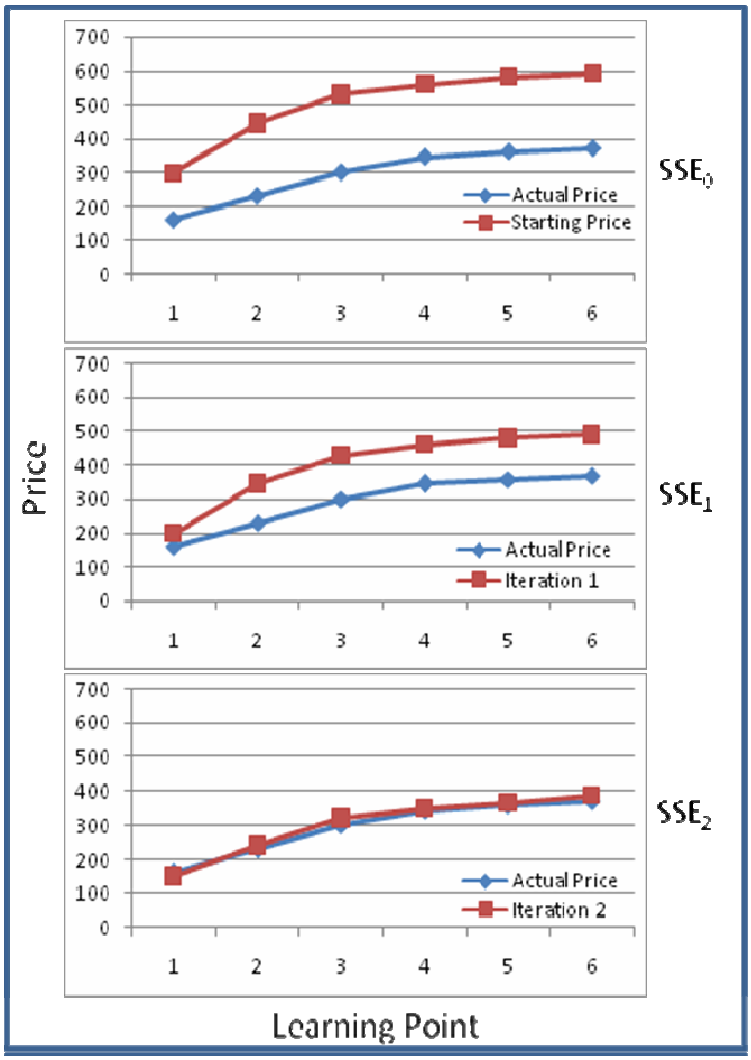

Figure 5.1 Sum of squared errors at iteration 0 through 2

As illustrated in Figure 5.1, the sum of squared errors is minimized at each iteration where $SSE_0 > SSE_1 > SSE_2$. The iterative process is ended when the value of SSE is below a predetermined tolerance level or when the process has reached the maximum iteration number of 30.

The next section deals with the different combinations of algorithms that were applied in order to minimize equation 5-3.

5.2 Overall combinations of algorithms

We follow the experimental framework in Shi (2005), thereby permitting a comparison of the different methods for the problem. All combinations of quasi-Newton methods and line search algorithms are presented in Figure 5.2

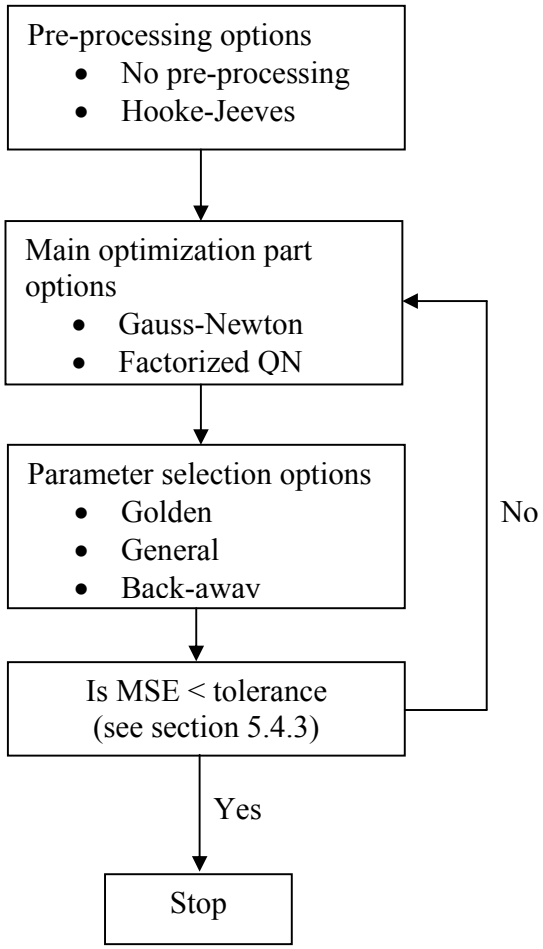

Figure 5.2 Combinations of algorithms tested

After several attempts, it was found that both polynomial interpolation and the Golden section search method were not viable options since they increased computational time and did not introduce an appreciable gain to the final results. Therefore, their results

are not included in chapter 6. Also, only the generalized structured and factorized quasi-Newton methods had any success in terms of solving equation 3-7. Both the Bartholomew-Biggs and Dennis-Gay-Welsh updates introduced problems including rank deficiency and singularity of the approximation matrices. These problems caused equation 3-6 not to converge due to computational errors. As a result, only the generalized quasi-Newton update in its structured and factorized forms is presented in the experimental results in chapter 6.

An updated table of all of the possible permutations of pre-processing, parameter selection and optimization algorithms is presented below.

Generalized QN Methods	Hook-Jeeves	Parameter Selection	Algorithms Combinations
GN	No	GEN	GN_GEN
		BA	GN_BA
	Yes	GEN	HJ_GN_GEN
		BA	HJ_GN_BA
Structured	No	GEN	SQN_GEN
		BA	SQN_BA
	Yes	GEN	HJ_SQN_GEN
		BA	HJ_SQN_BA
Factorized	No	GEN	FQN_GEN
		BA	FQN_BA
	Yes	GEN	HJ_FQN_GEN
		BA	HJ_FQN_BA

LEGEND

GN: Gauss-Newton method
SQN: Structured quasi-Newton method
FQN: Factorized quasi-Newton method
BA: Back-away parameter update
GEN: General parameter update

Table 5.2 Combinations of optimization methods

5.3 Definition of Convergent and Divergent Cases

Each of combination of pre-processing, optimization algorithm and parameter selection will be evaluated using following characteristics. The number of actual prices generated, henceforth called *number of learning points* (NoLP), is 4, 6, or 10.

1. Convergence: The mean squared error between the estimated price and the actual price for NoLP (number of learning points) for (NoLP = 4, 6, 8, or 10) will be evaluated using the following formula:

$$MSE = \frac{\sum_{t=1}^{NoLP} \left(P(t) - \hat{P}(t) \right)^2}{NoLP} , \qquad (5\text{-}4)$$

where P(t) is the actual prices, and $\hat{P}(t)$ is the estimated prices (the number of prices depends on the value of NoLP). A convergent case arises when the chosen algorithm obtains a solution of MSE< 10^{-2} for equation 5-4. If this occurs, the function is said to have converged at a local optimal solution at some point $\theta^* = [P_{min}, P_{max}, T_{max}, \beta, k]$.

2. Divergence: If the algorithm does not converge within 30 iterations, we classify it as a divergent case. That is, no local optimal solution has been found.

5.4 Performance Measures

Each algorithm will be assessed in terms of convergence rate, scaled norm of the difference between the actual and the estimated parameters, the sum of squared errors of the next five learning points and the CPU time.

5.4.1 The convergence rate

The convergence rate is defined as the percentage of converged cases out of the total number of replications (200 in our case) in each learning method. This parameter will be examined closely to determine the effectiveness of a given method in reaching a conclusion to a given negotiation. It should be pointed out that convergence rate, as used herein, does not refer to the rate at which the algorithm converges to a solution.

5.4.2 The scaled norm

The following formula, for the norm of the difference between the actual and the estimated parameters, is used to evaluate the accuracy of the of the optimization algorithm.

$$\overline{norm} = \sqrt{\left(\frac{P_{\min} - \hat{P}_{\min}}{P_{\min}}\right)^2 + \left(\frac{P_{\max} - \hat{P}_{\max}}{P_{\max}}\right)^2 + \left(\frac{T_{\max} - \hat{T}_{\max}}{T_{\max}}\right)^2 + \left(\frac{\beta - \hat{\beta}}{\beta}\right)^2 + \left(\frac{k - \hat{k}}{k}\right)^2} \quad (5\text{-}5)$$

where [P_{min}, P_{max}, T_{max}, β, k] are the actual parameters for the time-dependent model and $\left[\hat{P}_{\min}, \hat{P}_{\max}, \hat{T}_{\max}, \hat{\beta}, \hat{k}\right]$ are the estimates. The scaled norm is an extremely important parameter in this thesis, since it measures the ability of a given set of algorithms to learn from an opponent. Hence, a given set of algorithms with a lower norm would be more successful at predicting an opponent's next bid/offer price.

5.4.3 SSE_N

While the SSE measure indicates the closeness with which the algorithm matches the prices already offered, it does not give any indication as to whether the process can estimate the future moves of the opponent. This estimation of the future is important. Thus, in this work, we use SSE_N, defined as

$$SSE_N = \sum_{i=NoLP+1}^{NoLP+5} \left(P(t) - \hat{P}(\hat{t}) \right)^2 \tag{5-6}$$

In this paper, NoLP =4, 6, 8, or 10 depending on the number of learning points that are being used.

5.4.4 CPU time

Computational time is an important measure to gauge the usefulness of a given set of optimization methods studied in the context of electronic negotiation. Since all electronic negotiations will occur in real time, any method that takes more than a few seconds would not find much practical use. Each method will be evaluated on the basis of the average computation time used in order to obtain convergence. As a result, attempts not resulting in convergence will not be included within this measure.

The next chapter deals with the experimental results obtained from the combination of algorithms presented within this chapter. Inferential statistical analyses will be conducted in order to evaluate the performance measures in both absolute and relative terms.

Chapter 6: Experimental Results

The following analysis on each combination of algorithms was performed using MATLAB 7.0 software package on a Pentium 4, 2.2 GHz CPU with 512MB of RAM. In each of the following sections, the performance measures will be evaluated and compared using relevant statistics.

6.1 Convergence Rate

The convergence rate obtained in this work is an important indication of the effectiveness of a given combination of pre-processing algorithms, optimization algorithms and parameter selection. In the context of electronic negotiation, the convergence rate represents the portion of successful attempts of estimating the negotiation curve of the opponent. Hence, a higher convergence rate would indicate a higher likelihood of gaining valuable information on the behaviour of the negotiation opponent.

The table below provides an overview of the results obtained for the convergence rates of each combination of pre-processing, optimization algorithm and parameter selection, for each set of learning points.

	GN_GEN	HJ_GN_GEN	GN_BA	HJ_GN_BA	SQN_GEN	HJ_SQN_GEN
LP_4	31.00%	35.50%	35.50%	72.00%	39.00%	41.50%
LP_6	21.50%	27.50%	49.50%	52.50%	32.50%	32.00%
LP_8	18.50%	23.50%	50.00%	52.00%	20.00%	30.00%
LP_10	18.00%	24.00%	48.00%	51.50%	22.50%	31.50%
	SQN_BA	**HJ_SQN_BA**	**FQN_GEN**	**HJ_FQN_GEN**	**FQN_BA**	**HJ_FQN_BA**
LP_4	69.00%	72.00%	58.00%	56.00%	73.00%	80.50%
LP_6	66.00%	67.50%	42.50%	40.00%	71.50%	72.50%
LP_8	64.00%	66.00%	41.00%	35.50%	67.50%	67.50%
LP_10	64.00%	64.50%	35.50%	31.00%	64.50%	69.50%

Table 6.1 Convergence Results of different optimization methods

In order to get a proper depiction of the effect that each algorithm has on the rate of convergence, it helps to look at each subclass of methods separately. Since several factors contribute to the convergence of a given method, the performance needs to be compared in three categories: pre-processing, optimization algorithm, and parameter selection. The results are presented in the table below.

	GN_GEN	SQN_GEN	FQN_GEN
LP_4	31.00%	39.00%	56.00%
LP_6	21.50%	32.50%	40.00%
LP_8	18.50%	20.00%	35.50%
LP_10	18.00%	22.50%	31.00%

	HJ_GN_GEN	HJ_SQN_GEN	HJ_FQN_GEN
LP_4	35.50%	41.50%	58.00%
LP_6	27.50%	32.00%	42.50%
LP_8	23.50%	30.00%	41.00%
LP_10	24.00%	31.50%	35.50%

	GN_BA	SQN_BA	FQN_BA
LP_4	35.50%	69.00%	73.00%
LP_6	49.50%	66.00%	71.50%
LP_8	50.00%	64.00%	67.50%
LP_10	48.00%	64.00%	64.50%

	HJ_GN_BA	HJ_SQN_BA	HJ_FQN_BA
LP_4	72.00%	72.00%	80.50%
LP_6	52.50%	67.50%	72.50%
LP_8	52.00%	66.00%	67.50%
LP_10	51.50%	64.50%	69.50%

Table 6.2 Convergence rate of each combination of optimization method

In the current thesis, we have looked at three main types of algorithms: the Gauss-Newton, the structured quasi-Newton and the factorized quasi-Newton. Each algorithm had a different effect on the convergence rates of the functions evaluated, given the same use of pre-processing and parameter selection. It is apparent by comparing algorithms in table 6.2 that there is an improvement in convergence going from the Gauss-Newton to structured quasi-Newton to factorized quasi-Newton method. This pattern seems to hold true regardless of the use of pre-processing, the method used for parameter selection or the number of learning points used. It is therefore evident that structured quasi-Newton methods and Factorized Quasi-Newton methods provide improved rates of convergence for estimating the time dependent negotiation function's parameters. Hence, it can be

concluded that the use of structured and factorized quasi-Newton methods hold an advantage over the Gauss-Newton based methods previously proposed by Shi (2005).

Another pattern that is apparent in table 6.2 is the improvement in convergence obtained by the use of the Hooke-Jeeves pre-processing method. By comparing methods with the same optimization algorithm and parameter selection, methods that used the Hooke-Jeeves pre-processing achieved higher convergence for 22 out of the 24 cases. This conclusion is illustrated in the table below which represents the relative improvement in the convergence rate of a given function by using Hooke-Jeeves pre-processing.

		GN	SQN	FQN
LP_4	BA	102.82%	4.35%	8.22%
LP_6		6.06%	2.27%	1.40%
LP_8		4.00%	3.13%	0.00%
LP_10		7.29%	0.78%	7.75%
LP_4	GEN	14.52%	6.41%	3.57%
LP_6		27.91%	-1.54%	6.25%
LP_8		27.03%	50.00%	15.49%
LP_10		33.33%	40.00%	14.52%

Table 6.3 Relative improvement of convergence using Hooke-Jeeves pre-processing

The results indicate that the Hooke-Jeeves pre-processing method is an integral part in obtaining higher convergence rates while attempting to solve the non-linear optimization function presented in equation 3.7.

The next apparent pattern in table 6.2 is the effect of parameter selection. Parameter selection makes a significant difference in the convergence rate obtained. In each of the 24 cases compared, the use of the back-away algorithm over the use of general parameter selection greatly increased the rate of convergence. This conclusion is

easily drawn by referencing the table below that presents the relative improvement to the convergence rate by the use of Back-away parameter selection.

		GN	SQN	FQN
LP_4	H-J	14.52%	76.92%	30.36%
LP_6		130.23%	103.08%	78.75%
LP_8		170.27%	220.00%	90.14%
LP_10		166.67%	184.44%	108.06%
LP_4	No H-J	102.82%	73.49%	36.21%
LP_6		90.91%	110.94%	70.59%
LP_8		121.28%	120.00%	64.63%
LP_10		114.58%	104.76%	95.77%

Table 6.4 Relative improvement of convergence using the back-away algorithm

The last pattern that presents itself in table 6.2 is the effect of the number of learning points used to estimate the opponents negotiation function. Although the results are not as apparent as with the other three patterns analyzed, there seems to be a pattern that indicates that the convergence rate drops as the number of learning points is increased. This pattern seems slightly more pronounced in cases where the general parameter selection is used as opposed to the back-away algorithm.

In conclusion, the results suggests that the most robust algorithm would be obtained by combining the most effective method suggested above; this would also imply that the least effective would be obtained by combining the least effective methods. This hypothesis is supported by the fact that the highest convergence rate (of 80.5%) was obtained by using Hooke-Jeeves pre-processing coupled with the factorized quasi-Newton method and back-away parameter selection with four learning points, while the

least effective was obtained by using the Gauss-Newton method without pre-processing and the general parameter selection with 10 learning points (18.0%).

Although the convergence rate is a good indication of the effectiveness of a given set of optimization methods, other performance measures also need to be examined to truly reach a definite conclusion. The next section deals with the calculated norms obtained with each combination of pre-processing, optimization algorithm and parameter selection.

6.2 Scaled Norm

The scaled norm is a parameter that is extremely important to the goals of this work. As given in section 5.2, the norm is calculated as the relative difference between estimated parameters and the actual parameter, relative to the actual parameters. Hence, the lower the value obtained for the scaled norm, the more accurately the negotiation parameters of an opponent have been estimated.

In order to obtain a general idea of the magnitude of scaled norms presented in this section, the table below has been provided.

	Pmin	Pmax	Tmax	Beta	k	Scaled Norm
Starting Parameters	**201.5398**	**392.6653**	**22.9402**	**3.0759**	**0.3230**	**0.6086**
Actual Parameters	**172.6017**	**302.8364**	**28.1865**	**3.0525**	**0.6079**	**-**
Norm 1	172.6009	302.8364	28.1861	3.0525	0.6079	0.0000
Norm 2	172.5931	302.8330	28.1643	3.0501	0.6080	0.0011
Norm 3	172.5879	302.8311	28.1547	3.0491	0.6080	0.0016
Norm 4	172.5331	302.8266	28.1456	3.0480	0.6082	0.0022
Norm 5	172.6393	302.8015	28.0528	3.0416	0.6080	0.0059
Norm 6	172.8231	302.8751	27.9701	3.0214	0.6073	0.0129
Norm 7	168.4903	302.3997	26.9111	2.9063	0.6219	0.0738
Norm 8	166.4856	302.2051	26.3608	2.8425	0.6284	0.1064
Norm 9	187.1797	301.1566	26.8774	3.0966	0.5670	0.1185
Norm 10	170.9672	301.4712	25.3245	2.7819	0.6193	0.1365
Norm 11	172.0412	301.0091	23.9785	2.6335	0.6191	0.2037
Norm 12	155.2072	301.2298	23.9255	2.5655	0.6611	0.2572

Table 6.5 Improvement of estimated parameters as the scaled norm is decreased

Table 6.5 contains the results obtained from a given run with 10 learning points. Each Norm presented from Norm 1 to Norm 12 resulted from one convergent case using the same starting and actual parameters. One conclusion that can be drawn from this table is that each convergent case does not necessarily guarantee that the negotiation parameters of an opponent have been estimated accurately. As an example, Norm 12 has converged, however its estimation of the parameters vary from 0.5% to 10%; whereas for Norm 1, all parameters have been estimated within 0.00002% to 0.001%. Therefore, the effectiveness of a given combination of pre-processing, optimization algorithm and parameter selection is also dictated by the ability to minimize the calculated scaled norm.

From the previous section, we make two observations that are going to influence the presentation in this section:

- First, we observe that the methods proposed in this thesis do provide significant improvements in the convergence rates.

- Second, the methods that provide the improvement are Hooke-Jeeves method (in pre-processing), back-away (in parameter selection) and quasi-Newton (in the optimization part).

Thus, in this section, we will study how the scaled-norm measure varies with respect to the three aforementioned algorithms and with respect to convergence rates. Tables 6.6 through 6.9 deal with the optimization algorithms, table 6.10 deals with pre-processing and table 6.11 with the parameter selection element. Finally, we also test to see how the number of learning points influences the norm.

The table below contains a breakdown of the mean and standard deviation of the combined cases for each permutation of pre-processing, optimization algorithm and parameter selection; we present results for all cases as well for convergent cases.

		GEN		HJ_GEN		BA		HJ_BA	
		Mean	STDEV	Mean	STDEV	Mean	STDEV	Mean	STDEV
GN	LP_4	1.5761	0.9103	0.8550	0.8912	1.4591	0.9539	0.9365	0.9172
	LP_6	2.0350	0.4849	1.1135	1.0883	1.6003	0.9632	0.7774	0.9645
	LP_8	2.0619	0.4899	1.8054	0.6285	2.0675	0.4326	0.8508	1.0175
	LP_10	2.0327	0.5020	1.1092	1.1054	1.5820	0.9824	0.8946	1.0282
SQN	LP_4	1.9145	0.5691	0.8978	0.8872	1.1927	0.9387	0.9349	0.8735
	LP_6	1.9619	0.5414	0.8591	0.9947	1.4170	0.9991	0.8165	0.9556
	LP_8	2.0675	1.6426	0.8508	0.9019	1.4082	1.5299	0.8576	0.8574
	LP_10	2.0468	0.4730	0.8651	1.0336	1.5210	0.9891	0.8953	1.0221
FQN	LP_4	1.1927	0.9387	1.2575	0.9400	0.9349	0.8735	0.9365	0.8721
	LP_6	1.4170	1.4170	1.8858	0.5881	0.8165	0.9556	0.8165	0.9439
	LP_8	1.4082	1.0188	1.5299	0.9802	0.8574	0.9871	0.9312	0.9452
	LP_10	1.5210	0.9891	1.6022	0.9590	0.8953	1.0221	0.8653	1.1345

(a) All cases

		GEN		HJ_GEN		BA		HJ_BA	
		Mean	STDEV	Mean	STDEV	Mean	STDEV	Mean	STDEV
GN	LP_4	0.2591	0.2239	0.3411	0.2586	0.3347	0.2562	0.2897	0.2167
	LP_6	0.0109	0.0297	0.0618	0.1268	0.0018	0.0028	0.0019	0.0025
	LP_8	0.0051	0.0088	0.0018	0.0023	0.0017	0.0026	0.0016	0.0022
	LP_10	0.0018	0.0029	0.0008	0.0012	0.0008	0.0011	0.0007	0.0011
SQN	LP_4	0.3303	0.2532	0.3199	0.2725	0.3282	0.2434	0.3050	0.2149
	LP_6	0.1417	0.1644	0.1096	0.1451	0.1002	0.1424	0.1222	0.1671
	LP_8	0.1845	0.2331	0.1085	0.1451	0.0406	0.0595	0.0584	0.0782
	LP_10	0.0896	0.1316	0.0516	0.0919	0.0400	0.0575	0.0742	0.0997
FQN	LP_4	0.3965	0.3157	0.3742	0.2472	0.3638	0.2632	0.3696	0.2498
	LP_6	0.2309	0.2215	0.1928	0.2124	0.1323	0.1746	0.1765	0.2109
	LP_8	0.1078	0.1228	0.0966	0.1171	0.1013	0.1344	0.1167	0.1401
	LP_10	0.1285	0.1522	0.1431	0.1866	0.0717	0.0902	0.0959	0.1294

(b) Only convergent cases

Table 6.6 Means and standard deviations of the scaled norms

In the following table, the norm values obtained using the structured quasi-Newton method and the Gauss-Newton method are compared using p-values at 95% confidence.

	GN versus SQN			
	GEN	GEN_HJ	BA	BA_HJ
LP 4	0.2035	0.2100	0.7444	0.3684
LP 6	0.0263	<0.0001	<0.0001	0.0012
LP 8	0.9045	<0.0001	<0.0001	0.0529
LP 10	0.2037	<0.0001	0.0031	0.0456

(a) All Cases

	GN versus SQN			
	GEN	GEN_HJ	BA	BA_HJ
LP 4	0.2278	0.6208	0.8283	0.5540
LP 6	<0.0001	0.0685	<0.0001	0.0269
LP 8	<0.0001	<0.0001	0.0009	<0.0001
LP 10	0.0002	0.0010	<0.0001	<0.0001

(b) Convergent cases

Table 6.7 ANOVA results comparing structured quasi-Newton with Gauss-Newton

Table 6.7 provides the p-values of GN versus SQN for various combinations of the other components (e.g, column 2 provides the influence of SQN when using Gen parameter update and when HJ pre-processing is used). The results of the *p*-values displayed in table 6.7(a) are mixed, however they seem to indicate that structured quasi-Newton methods improves the calculated norm in 9 out of the 16 variations attempted for all cases. Several resulting p-values in table 6.7(b) suggest that there is a difference between the scaled norms of convergent cases calculated using the SQN method versus those calculated using the GN method. However, table 6.6(b) indicates that smaller norm values are obtained using the GN method. Hence, the opposite effect is apparent for convergent cases. This can possibly be explained by the fact that the use of the SQN optimization algorithm results in more convergent cases. Also, it could be explained by the possibility that the GN optimization algorithm will only result in convergent cases for functions evaluated whose initial starting points result in small calculated norm values. Hence, cases with larger norms that would not have normally converged via the use of the GN optimization algorithm are brought within the range of convergence while using the SQN algorithm. Therefore, in order to properly compare the effect of the optimization algorithms, parameter selection algorithms and pre-processing algorithm on the scaled norm, it is beneficial to only base the analysis on all cases. For the remainder of this section, all results will be presented from all cases and all convergent cases; however, conclusions from statistical analysis will only be drawn from comparison among optimization methods showing all 200 cases.

In the following table, the norm values obtained using the factorized quasi-Newton method and the structured quasi-Newton method are compared using p-values at 95% confidence.

	SQN versus FQN			
	GEN	GEN_HJ	BA	BA_HJ
LP 4	<0.0001	0.0339	0.6738	0.1663
LP 6	<0.0001	0.0004	0.6633	0.6132
LP 8	<0.0001	0.2428	0.9477	0.7109
LP 10	<0.0001	0.8355	0.7699	0.4327

(a) All cases

	SQN versus FQN			
	GEN	GEN –HJ	BA	BA – HJ
LP 4	0.2427	0.0850	0.2453	0.0212
LP 6	0.1450	0.0006	<0.0001	<0.0001
LP 8	0.0013	0.8343	0.0119	<0.0001
LP 10	0.1365	0.0004	0.0018	0.2167

(b) Convergent cases

Table 6.8 ANOVA Results comparing structured quasi-Newton with factorized quasi-Newton

The results of the *p*-values displayed in table 6.8(a) are mixed, however they seem to indicate that the norm calculated using the factorized quasi-Newton methods improves the calculated norm when the general parameter selection is used without Hooke-Jeeves pre-processing (GEN, as shown in column 1).

In the following table, the norm values obtained using the factorized quasi-Newton method and the Gauss-Newton method are compared using p-values at 95% confidence.

	GN versus FQN			
	GEN	GEN_HJ	BA	BA_HJ
LP 4	<0.0001	0.0006	0.9295	0.6542
LP 6	<0.0001	0.0060	<0.0001	0.0051
LP 8	<0.0001	<0.0001	<0.0001	0.0199
LP 10	<0.0001	<0.0001	0.0070	0.0053

(a) All cases

	GN versus FQN			
	GEN	GEN –HJ	BA	BA – HJ
LP 4	0.0142	0.2110	0.2614	0.0040
LP 6	<0.0001	0.0006	<0.0001	<0.0001
LP 8	<0.0001	<0.0001	<0.0001	<0.0001
LP 10	<0.0001	<0.0001	<0.0001	<0.0001

(b) Convergent cases

Table 6.9 ANOVA Results comparing Gauss- Newton with factorized quasi-Newton

The results of the *p*-values displayed in table 6.9(a) indicate that the norm calculated using the factorized quasi-Newton methods improves the calculated norm when the general parameter selection is used with and without Hooke-Jeeves pre-processing.

In terms of the scaled norm in this thesis, there are three parameters that need to be investigated further. As discussed in the previous section, it seems that other components yielding improved convergence are: Hooke-Jeeves pre-processing, back-away parameter selection. Thus we investigate the following:

In the following table, the norm values obtained with and without the use of the Hooke-Jeeves pre-processing are compared using p-values at 95% confidence.

	GN		*SQN*		*FQN*	
	GEN	BA	GEN	*BA*	*GEN*	*BA*
LP 4	<0.0000	0.4228	<0.0001	0.6687	0.4908	0.1605
LP 6	0.4269	<0.0000	<0.0001	0.4059	0.0000	0.9226
LP 8	<0.0000	<0.0000	<0.0001	0.6188	0.2255	0.9437
LP 10	0.1245	0.4960	<0.0001	0.7756	0.4061	0.4061

(a) All Cases

	GN		*SQN*		*FQN*	
	GEN	BA	GEN	*BA*	*GEN*	*BA*
LP 4	0.9953	0.0806	0.3721	*0.6442*	*0.2536*	*0.3025*
LP 6	0.1663	<0.0001	0.2162	*0.5864*	*0.6233*	*0.4240*
LP 8	0.6890	0.6251	0.4913	*0.6602*	*0.6105*	*0.7617*
LP 10	0.4357	0.7451	0.1438	*0.5487*	*0.7654*	*0.2315*

(b) Convergent cases

Table 6.10 ANOVA Results comparing cases with and without Hooke-Jeeves pre-processing

The results of the *p*-values displayed in table 6.10(a) indicate that the norm values obtained with Hooke-Jeeves pre-processing are smaller then those obtained using no pre-processing for SQN while using the general parameter selection. Also, the results suggest that the same pattern holds true for the GN algorithm using both parameter selection methods and the FQN optimization algorithm using the general parameter selection. However, the same observation is not supported for either the SQN or FQN optimization algorithms while using the back-away parameter selection. Overall, it appears that there is sufficient evidence to support that the Hooke-Jeeves pre-processing method may help in reducing the scaled norm obtained while estimating the TDT parameters. However, this result is not constant for each combination of methods analysed.

The next parameter to be analysed is the effect of parameter selection on the scaled norm. As with the previous Hooke-Jeeves pre-processing analysis, all other parameters will be held constant to determine the effect of the parameter selection in isolation.

In the following table, the norm values obtained using the back-away parameter selection and the general parameter selection are compared using p-values at 95% confidence.

	GN		SQN		FQN	
	No HJ	HJ	No HJ	HJ	No HJ	HJ
LP 4	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
LP 6	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
LP 8	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
LP 10	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001

(a) All cases

	GN		SQN		FQN	
	No HJ	HJ	No HJ	HJ	No HJ	HJ
LP 4	0.7541	0.1406	0.6346	0.2448	0.7807	0.8337
LP 6	0.8657	0.0121	0.8396	0.0681	0.8583	0.5543
LP 8	0.1899	0.6460	0.1956	0.0765	0.3680	0.9559
LP 10	0.0679	0.3787	0.5727	0.9639	0.2226	0.7639

(b) Convergent cases

Table 6.11 ANOVA Results comparing the influence of the back-away algorithm

The results of the *p*-values displayed in table 6.11(a) all indicate that the calculated scaled norms significantly improve with the use of back-away parameter selection. Therefore, there is strong reason to believe that the use of the back-away parameter selection can improve the learning of an opponents negotiation parameters.

The next parameter to be tested in table 6.12 concerns the number of learning points used. By observing the mean values of the norm in table 6.6(b) it becomes evident that the mean scaled norm decreases as the number of learning points increases. This seems intuitive since more information regarding an opponent's negotiation behaviour is gained at each exchange of bid-offers between the buyer and seller.

	GEN			BA		
	GN	SQN	FQN	GN	SQN	FQN
No HJ	0.0648	<0.0001	<0.0001	0.0425	<0.0001	<0.0001
HJ	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001

(a) All cases

	GEN			BA		
	GN	SQN	FQN	GN	SQN	FQN
No HJ	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
HJ	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001

(b) Convergent cases

Table 6.12 ANOVA Results comparing the influence of the number of learning points

The results of the *p*-values displayed in Table 6.12(a) and (b) all indicate, with one exception, that the scaled norm estimated decreased by increasing the number of learning points. Hence, the initial assumption is supported. Therefore, the more learning points obtained from an opponent, the lower the value of the scaled norm will be. Hence, the accuracy of the estimation of an opponent's negotiation behaviour can be significantly increased by obtaining more learning points.

The test results of this section bring out a few important observations. First, as discussed in Table 6.5, algorithmic convergence does not necessarily imply an accurate estimation of the parameters. Fortunately, based on the mean results on our test problems, when convergence does occur, the estimations have been generally accurate. Thus, the superior convergence rates of our methods yield improvements on knowing the parameters characterizing the underlying negotiation behaviour of the opponent.

6.3 SSE_N

The next parameter examined is extremely important in the context of electronic negotiation. It examines the ability of a given set of methods to predict the next five price offers given by the buyer. A method with a low SSE_N is a direct measure of its effectiveness to predict the actual future offers of the opponent. Since this parameter is based on the next five moves of an opponent, once it is believed that the negotiation parameters of the opponent have been properly estimated, it is only valid for convergent cases. Hence, the results presented in the section will only represent those obtained from convergent cases.

		GEN		HJ_GEN		BA		HJ_BA	
		Mean	STDEV	Mean	STDEV	Mean	STDEV	Mean	STDEV
GN	LP_4	0.1053	0.1285	0.1209	0.1347	0.1236	0.1433	0.0967	0.1066
	LP_6	0.0001	0.0005	0.0019	0.0083	0.0000	0.0000	0.0000	0.0000
	LP_8	0.0000	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	LP_10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SQN	LP_4	0.1111	0.1271	0.0938	0.1090	0.1255	0.1375	0.1195	0.1347
	LP_6	0.0263	0.0392	0.0173	0.0374	0.0045	0.0078	0.0075	0.0206
	LP_8	0.0410	0.0817	0.0129	0.0263	0.0019	0.0040	0.0032	0.0065
	LP_10	0.0085	0.0173	0.0006	0.0014	0.0011	0.0029	0.0040	0.0097
FQN	LP_4	0.1477	0.1483	0.1777	0.2237	0.1495	0.1635	0.1396	0.1468
	LP_6	0.0391	0.0617	0.0618	0.0873	0.0212	0.0520	0.0357	0.0580
	LP_8	0.0095	0.0170	0.0149	0.0307	0.0100	0.0223	0.0151	0.0258
	LP_10	0.0090	0.0257	0.0158	0.0291	0.0076	0.0144	0.0091	0.0234

Table 6.13 Sum of squared errors over the next five turns for all convergent cases

At a quick glance, one pattern becomes evident in table 6.13. As expected, the more learning points that are obtained, the lower the value of SSE_N will be. This is a direct implication of the lower value obtained for the scaled norm of functions with a higher number of learning points, as previously observed from the results in table 6.12 .

It was not apparent from the results in table 6.10(b) that the norm values obtained using Hooke-Jeeves pre-processing were lower than those obtained without pre-processing. However, one way to further the analysis on the effectiveness of the Hooke-Jeeves algorithm is to test its ability to lower the value of SSE_N obtained.

In the following table, SSE_N values obtained using Hooke-Jeeves pre-processing and those obtained without pre-processing are compared using p-values at 95% confidence.

	GN		SQN		FQN	
	GEN	BA	GEN	*BA*	*GEN*	*BA*
LP 4	<0.0001	0.0463	<0.0001	0.1212	0.1053	0.0386
LP 6	0.0436	<0.0001	<0.0001	0.1686	0.1507	0.4540
LP 8	0.0001	<0.0001	<0.0001	0.1095	0.2114	0.6913
LP 10	0.0133	0.0328	<0.0001	0.0103	0.0402	0.1902

Table 6.14 ANOVA results comparing SSEN with and without Hooke-Jeeves pre-processing

The results of the *p*-values displayed in table 6.14 are mixed. Several cases support the assumption that Hooke-Jeeves pre-processing can lower the SSE_N and several others do not. Hence, it cannot be assured with certainty that the value of SSE_N significantly improves with the addition of a pre-processing stage with 95% confidence for each method. However, since several cases support this preliminary observation, there is reason to believe that the addition of a pre-processing step may be worthwhile step to include as part of our learning algorithm.

6.4 CPU Time

The next parameter analyzed in this work is the computational time used in order to reach convergence in each method. Table 6.15 contains the values of computational time, in seconds, obtained.

		GEN		HJ_GEN		BA		HJ_BA	
		Mean	STDEV	Mean	STDEV	Mean	STDEV	Mean	STDEV
GN	LP_4	0.1062	0.0486	0.1502	0.0733	0.1269	0.0704	0.1649	0.0807
	LP_6	0.1277	0.0546	0.1765	0.1089	0.1583	0.0688	0.1854	0.0876
	LP_8	0.1570	0.0690	0.1912	0.0693	0.1814	0.0805	0.2300	0.1188
	LP_10	0.2130	0.0985	0.2171	0.0641	0.2156	0.0753	0.2489	0.1110
SQN	LP_4	0.1679	0.1003	0.1966	0.1034	0.1610	0.0862	0.2061	0.1007
	LP_6	0.2243	0.1446	0.3273	0.1741	0.3394	0.2161	0.4217	0.2363
	LP_8	0.2588	0.1648	0.3340	0.2100	0.2983	0.1803	0.3498	0.1916
	LP_10	0.3442	0.2284	0.3707	0.2327	0.3711	0.2175	0.4389	0.2555
FQN	LP_4	0.2479	0.1281	0.2364	0.1121	0.2217	0.1128	0.2466	0.1245
	LP_6	0.4289	0.2462	0.5126	0.2462	0.3291	0.1575	0.3213	0.1574
	LP_8	0.3790	0.2087	0.3994	0.1872	0.4461	0.2005	0.4135	0.2061
	LP_10	0.4888	0.2776	0.5679	0.2754	0.5788	0.2623	0.5391	0.2849

Table 6.15 Computational time mean and standard deviations

As would be expected, when the Faratin function is evaluated using more learning point, computational time will be added. This statement is supported by the test by the p-values in table 6.16 below.

	GEN			BA		
	GN	SQN	FQN	GN	SQN	FQN
No HJ	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
HJ	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001

Table 6.16 ANOVA results comparing computational time comparison among cases with different number of learning point selection

The results of the *p*-values displayed in table 6.16 indicate that the computational time is increased as the number of learning points is increased. Hence, the more learning points obtained from an opponent, the longer it will take to estimate an opponent's negotiation parameters.

Another pattern arose that was also expected. It appears by examining the data in table 6.15 that the computational time is increased with increasing complexity of the optimization algorithm. This in turn would suggest that the least computational time

would be needed using the Gauss-Newton algorithm while the most time would be required while using the factorized Quasi-Newton algorithm.

	GEN			BA		
	GN	SQN	FQN	GN	SQN	FQN
No HJ	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
HJ	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001

Table 6.17 ANOVA results comparing computational time comparison among cases with different optimization algorithms

The *p*-values in table 6.17 support the hypothesis that the more computationally complex an algorithm is, the more computation time it will take in order to solve the nonlinear optimization problem presented in equation 3-7.

Although it has been show with 95% confidence that the computational time increases with an increase in the number of learning points and with increasing computational complexity of the optimization algorithm, the improvement in convergence of both of these methods more than compensates for the increased time. By taking the computational time into the context of a real time electronic negotiation, the time difference between an algorithm that can make the computation in 0.1s versus a more complex and accurate algorithm that can be performed in 0.5s is negligible. Hence, in this case, the computational time should not affect the decision of which methods are to be used in a real time electronic negotiation.

6.5 Summary of Results

In summary, this chapter has demonstrated that, as compared to the standard Gauss-Newton method, higher convergence rates for solving equation 3.7 can be obtained by the use of a structured quasi-Newton method. It has also demonstrated that by the use

of factorized quasi-Newton methods, convergence can again be improved over the use of a structured quasi-Newton methods by ensuring that the Hessian approximation matrix is positive definite, hence ensuring a decent direction at each iteration.

The back-away algorithm used in this paper also significantly increased convergence in each case, regardless of the optimization algorithm used and regardless of whether or not parameter pre-processing was performed. By glancing at table 6.2, the effectiveness of the back-away parameter selection method is clearly evident. It introduced relative improvements to convergence in different cases with the lowest improvement being 14.5% percent, and the highest being 220%. It was also determined that by increasing the number of learning points, the rate of convergence will decrease. However, this negative effect was dampened by a decrease in scaled norm, hence resulting in more accurate estimations of an opponent's negotiation behaviour.

Although we have conducted no mathematical analysis concerning the improvements afforded by our methods, the statistical results have demonstrated with 95% confidence the that use of the back-away algorithm, Hooke-Jeeves pre-processing and factorized quasi-Newton methods can improve the learning of an opponent's negotiation parameters. It has also been found that the use of a higher number of learning point will also result in a more accurate estimation of the negotiation parameters. These conclusions are important since they ultimately support the initial motivation for this thesis.

In conclusion, in order to improve the convergence and the accuracy of the estimation of an opponent's negotiation parameters, Hooke-Jeeves pre-processing coupled with a factorized quasi-Newton method and back-away parameter selection should be used. If convergence is more important in the analysis, a smaller number of

learning points should be used. However, if the ultimate goal of the analysis is to accurately estimate the negotiation parameters of your opponent, the more learning points obtained the more accuracy that will be achieved.

The next chapter summarizes the significance of the results obtained within this chapter. Potential applications of the work performed within this thesis will be summarized and future work to possibly improve the results further will be brought forth.

Chapter 7 Conclusion and Implications

This chapter deals with the implication of the results obtained in this work. Possible applications for this solution method will be examined and future work that may contribute to further improvement to the results obtained will be outlined.

7.1 Summary

In this thesis, the TDT has been defined and the estimation of its parameters was formulated into a nonlinear least squares problem. A successful attempt was made which resulted in improved algorithmic convergence and accuracy in TDT parameter estimation. This positive result may have several practical implications in the context of electronic negotiation.

7.2 Implication of results

The results achieved in this work have significantly improved on those obtained by Shi (2005) by increasing the convergence rate from the high 60% range to approximately 80%. Also, the low values of the scaled norm obtained by a number of the methods tested suggest that the nonlinear least squares optimization methods attempted can estimate the negotiation parameters with great accuracy. This in turn implies that the next move made by an opponent in an electronic negotiation can be estimated with reasonable accuracy. This statement is supported by the very small values of the sum of squared errors obtained over the next five moves by an opponent (Table 6.13).

By taking the results in context of an electronic negotiation, a method has been found that can accurately estimate the negotiation parameters of an opponent, and use

these results to accurately predict the value of the next bid/offer price. This estimation can be obtained within a fraction of a second (Table 6.15) on a standard desktop computer, semi-autonomously and without any human interaction. The combination of all of these results indicates that the methods applied in this work may potentially contribute to the design of appropriate decision models underlying an automated negotiation agent. As pointed out by (Bichler and Kersten 2003), decision models constitute only one aspect of negotiation agents and the construction of agents entails rigorous Computer Science techniques (e.g., computational linguistics, artificial intelligence, protocol design etc.).

In the next section, previous research conducted is analysed for its potential use in conjunction with the learning methods presented in this paper. An algorithm is presented that can enable a buyer/seller to apply the learning methods in order to obtain a lower/higher transaction price.

7.3 The benefit of learning

In the current thesis, it has been concluded that one is able to successfully learn an opponents negotiation parameters using various combinations of pre-processing, optimization algorithm and parameter selection. Previous work (Mok and Sundarraj, 2005) has shown that the ability to learn an opponent's negotiation behaviour throughout the process of negotiation can be beneficial in terms of maximizing utility. Their proposed algorithm, the reaction algorithm, consists of three phases: (i) selection of target range; (ii) feasibility check; and (iii) parameter adjustment. The first phase uses the learnt parameters to determine a set of final target offers that would improve the negotiation outcome, as compared to the non-learning outcome. In the feasibility phase, they attempt

to determine if the targets obtained are feasible. Finally, in the parameter adjustment phase, the concession rate (β in this paper) is adjusted in order to reach the final target.

This algorithm was experimentally tested under each of the follow scenarios:

1. The buyer and the seller exhibit boulware behaviour
2. The buyer and the seller exhibit conceder behaviour
3. The buyer exhibits boulware behaviour and the seller exhibits conceder behaviour
4. The buyer exhibits conceder behaviour and the seller exhibits boulware behaviour

The figure below (Mok and Sundarraj, 2005) is a pictorial description of the effect of learning on the resulting transaction price.

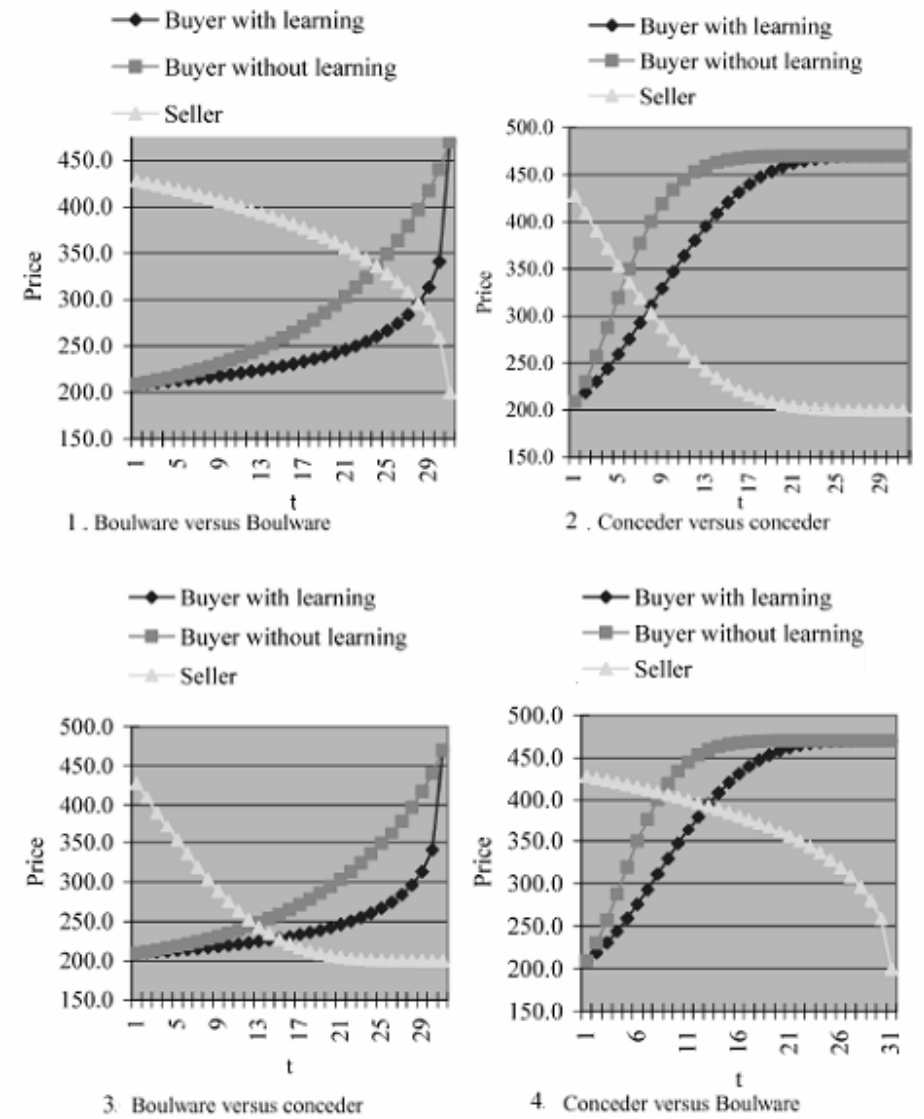

Figure 7.1 Effect of learning on the transaction price

In each of the above outcomes, Mok and Sundarraj have determined that the use of learning algorithms, in conjunction with the reaction algorithm, can benefit the buyer by lowering the final negotiation price (transaction price). This is an extremely important conclusion since it highlights the importance of the work performed in this thesis. The improved learning algorithm, achieved with the use of back away parameter selection, Hooke-Jeeves pre-processing and the factorized quasi-Newton algorithm, can be used in conjunction with the reaction algorithm to benefit the buyer/seller in terms of further reducing lowering/raising the transaction price. The figure below is an example of how accurate leaning can lead to an improved transaction price for a buyer.

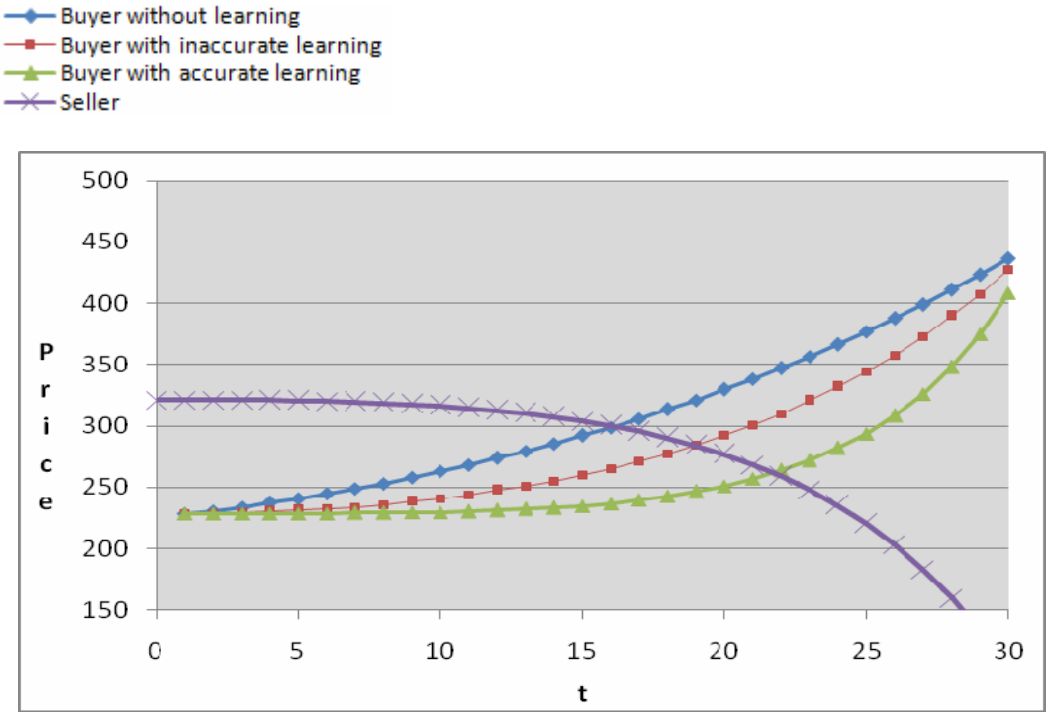

Figure 7.2 Effect of improved learning on the transaction price

In figure 7.2, the buyer with accurate learning is able to significantly lower the price at which he/she obtains the good or service under negotiation. Hence, the ability to structure robust learning algorithm can ultimately result in beneficial results for the user.

The next section deals with future work that may result from this thesis.

7.4 Future Work

Future work that may be applied to further strengthen the conclusion obtained in this thesis can include a series of tests in real-life negotiation settings against actual opponents. The design could involve selecting a large pool of people, and having them negotiate online on a one-on-one basis with an electronic agent. The lowest bid price and the highest offer price of the item under consideration would be predefined by the system in order to set the pricing parameters. Also, the maximum number of iterations allowable to reach an agreement would be set in order to add time pressure to the process. The bids and offers of each respective party would be recorded, at each iteration, in order to obtain an estimate of their negotiation parameters. After the completion of a successful transaction, the buyer and seller would exchange roles, and the same procedure would be followed using a different item. Once the data is collected, each negotiator would be required to participate in four additional negotiations with the automated agent. This would allow the agent to act as a buyer and seller, with and without learning, against a single opponent.

The results of this experiment could be quantified by comparing the normalized transaction price obtained, for each negotiation conducted by the agent, with and without the use of learning. The normalized transaction price of the buyer would be given by the

difference between the median price and the transaction price, divided by the difference between the maximum price and the minimum price. For the seller, the normalized transaction price would be given by the difference between the transaction price and the median price, divided by the difference between the maximum price and the minimum price.

In the first equation, the normalized transaction price is positive when the agent, as a buyer, achieves a transaction price lower than the median price. Likewise, in the second equation, the normalized transaction price of the agent, as a seller, is positive when the transaction price is higher than the median price. In each case, the agent will be awarded with a positive value, normalized by the difference between the set maximum and minimum price of the item under negotiation, when a transaction price that is preferred to the median price is obtained. By comparing the results obtained for cases with and without the benefits of learning, the net effect of learning in an actual negotiation setting can be tested using relevant statistics, with a high degree of confidence. A successful result of this experiment would strengthen the conclusion that the learning methods described in this paper could be used to benefit an agent in a real life electronic negotiation.

The main limitation of this experiment is that the current learning method used in this thesis makes several assumptions that do not allow the opponent to act irrationally, which may be the case in the real-life application described above. In future work, the learning methods presented in this paper can be further tested in situations where the opponent's negotiation behaviour changes dramatically during the life of the negotiation. If a solution is found, it will be able to greatly increase the ability of this type of negotiation strategy to find a place in an e-commerce setting.

Bibliography

Bansal, V. (2005). "Simultaneous Independent Online Auctions with Discrete Bid Increments." Electronic Commerce Research, **5**(2): 181-201.

Bartholomew-Biggs, M. C. (1977). "The Estimation of the Hessian Matrix in Nonlinear least squares problems with non-zero residuals." Mathematical Programming **12**: 67-80.

Beam, C. (1999). "A New Market-based Negotiation Paradigm." http://haas.berkeley.edu/~citm/nego/newnego.html.

Bichler, M and Kersten, G. (2003). "Towards a structured design of electronic negotiations." Group Decision and Negotiation **12**(4): 311-335.

Bjorke, A. (1996). Numerical Methods for Least Squares Problems, SIAM.

Cardoso, H. L. and Oliveira, E. (2001). A platform for electronic commerce with adaptive agents. Agent-Mediated Electronic Commerce III. Current Issues in Agent-Based Electronic Commerce Systems, 3-4 June 2000, Barcelona, Spain, Springer-Verlag.

Chaves, A. M. (1996). Kasbah, An Agent Marketplace for Buying and Selling Goods. Proceedings of the First International Conference on The Practical Application of Intelligent Agents and Multi-agent Technology, London, UK.

Choi S. P. M., Liu, J., and Chan, S. P. 2001. "A Genetic agent-based negotiation system". Computer Networks and ISDN Systems 37, 195–204.

Da-Jun, C. and Liang-Xian, X (2002). A negotiation model of incomplete information under time constraints. AAMAS '02: First International Joint Conference on Autonomous Agents and Multi-Agent Systems, 15-19 July 2002, Bologna, Italy, ACM.

Dasgupta, P. (2003). "Dynamic Consumer Profiling and Tiered Pricing Using Software Agents." Electronic Commerce Research **3**: 277-296.

de Paula, G. E. and Ramos, F. S. (2001). Bilateral Negotiation Model for Agent-Mediated Electronic Commerce. Lecture Notes in Computer Science Volume 2003/2001.

Dennis, J. E. and Walsh R.E (1981). "An Adaptive Nonlinear Least-Squares Algorithm." ACM Transactions on Mathematical Software (TOMS) **7**(3), 23-41.

Deveaux, L. (2001). "Bargaining on an Internet Agent-based Market: Behavioural vs. Optimizing Agents." Electronic Commerce Research **1**(4): 371-401.

eBay, "sneak peak", Retrieved July 18th, 2007, from http://pages.ebay.com/sneakpeek

Economist, The. (2000). "Dotty about dot.commerce? The E-Commerce boom is changing business, for the better [Editorial]." The Economist **354**(8159): 24.

Faratin, P. (2000). "Automated Service Negotiation Between Autonomous Computational Agents." A dissertation submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy of the University of London.

Faratin, P. and Sierra, C. and Jennings, N.R. (1998). "Negotiation decision functions for autonomous agents." Robotics and Autonomous Systems **24**(3-4): 159-82.

Fatima, S. S. and Wooldridge, M (2002). Multi-issue negotiation under time constraints. AAMAS '02: First International Joint Conference on Autonomous Agents and Multi-Agent Systems, 15-19 July 2002, Bologna, Italy, ACM.

Feldman, S. (2000). "Electronic marketplaces." IEEE Internet Computing **4**(4): 93-95.

Fisher, R. U., (1981). Getting to Yes: Negotiating an Agreement without Giving In, Random House.
Fletcher, R. and Xu, C. (1987). "Hybrid methods for nonlinear least squares." IMA Journal of Numerical Analysis **7**(3): 371-89.
Genesereth, M. R. G., (1986). Cooperation without communication. In Proc. of the National Conference on Artificial Intelligence, Philadelphia.
Global Reach, "Electronic Commerce". Retrieved March 18th, 2006, from http://www.glreach.com/eng/ed/art/ecommerce.html
Gordon Lo, G. (1999). Negotiation and Electronic Commerce: Integrating Negotiation Support and Software Agent Technologies. 29th Atlantic Schools of Business Conference (1999)
Government of Canada, "Electronic Commerce". Retrieved February 13th, 2006, from http://www.statcan.ca
Guttman, R. H. (1999). "Agent-mediated Integrative Negotiation for Retail Electronic Commerce." Workshop on Agent Mediated Electronic Trading (AMET'98).
Guttman, R. H. and Maes, P (2000). "Agents as Mediators in Electronic Commerce". Electronic Markets, MIT Media Lab. Cambridge**:** 1-6.
Harsanyi, J. C. (1972). "Generalized Nash solution for 2-person bargaining games with incomplete information." Management Science **18**(5): 50-106.
He, M. H. and Jennings, N. R (2003). "On agent-mediated electronic commerce." IEEE Transactions on Knowledge and Data Engineering **15**(4): 985-1003.
International Engineering Consortium, "Electronic Commerce". Retrieved January 13th, 2007, from http://www.iec.org/online/tutorials/e_commerce
Jennings, N. R. and Faratin. P. (2001). "Automated negotiation: Prospects, methods and challenges." Group Decision and Negotiation **10**(2): 199-215.
Kersten, G.E. and Noronha, S. and Teich, J. (2000). "Are All E-Commerce Negotiations Auctions?" Fourth International Conference on the Design of Cooperative Systems, 1-10Kitti, M. (2004). "Analysis of the Constraint Proposal Method for Two-Party Negotiations." Helsinki University of Technology: 1-21.
Lee, K. J. and Chang, Y.S. (2000). "Time-bound negotiation framework for electronic commerce agents." Decision Support Systems **28**(4): 319-31.
Lomuscio, A. R. and Wooldridge, M. (2003). "A classification scheme for negotiation in electronic commerce." Group Decision and Negotiation **12**(1): 31-56.
Mahadevan, B. (2000). "Business Models for Internet based E-Commerce: An Anatomy." California Management Review **42**(40): 1-33.
Mok, W. and Sundarraj, R. P. (2005). "Learning Algorithms for Single-Instance Electronic Negotiations Using the Time-Dependent Behavioural Tactic." ACM Transactions on Internet Technology (TOIT) **5**(1): 195-230.
Nash, J. (1950). "The bargaining problem." Econometrica **18**: 155-162.
Panagariya, A. (1999). "Electronic Commerce, WTO and Developing Countries." WTO, Electronic Commerce **5**(15): 28.
Panagariya, A. (1999). "Electronic Commerce, WTO, and Developing Countries." WTO, Electronic Commerce **5**(11): 30-31.
Papamichail, G. P. and Papamichail, D. P. (2003). "Towards using computational methods for real-time negotiations in electronic commerce". European Journal of Operational Research. **145:** 232-238.
Pruitt, D. (1981). Negotiation Behavior, Academic Press.

Raiffa, H. (1982). The Art and Science of Negotiation. Cambridge, Mass., Harvard Univesrity Press: 23-27

Robbins, S. P. (2005). Communication, conflict, and Negotiation. Fundamentals of Organizational Behaviour. Toronto, Pearson Prentice Hall**:** 195.

Rosenschein, J. S. (1994). Rules of Encounter: Designing Conventions for Automated Negotiation among Computers. The MIT Press. Cambridge**:** 20-22.

Salvatore, D. (2001). Market Structure and Pricing Practices. Managerial Economics in a Global Economy. New York, Harcourt College Publishers**:** 482-510.

Sandholm, T. (1999). "Automated Negotiation." Communications of the ACM **42**(3): 84-85.

Scales, L. E. (1985.). Introduction to non-linear optimization. London, U.K., Macmillan.

Shi, X. (2005). Comparison of Nonlinear Optimization Methods for Learning Time-dependent Decision Functions in Electronic Negotiations. Management Sciences MASc Thesis,. Waterloo, University of Waterloo**:** 133.

Stone, P. (2005). "The First International Trading Agent Competition: Autonomous Bidding Agents." Electronic Commerce Research and Applications **5**: 229-265.

Strobel, M. and Weinhardt, C. (2003). "The Montreal Taxonomy for electronic negotiations." Group Decision and Negotiation **12**(2): 143-164.

Wang, Y. T. (2004). "PumaMart: a parallel and autonomous agents based internet marketplace." Electronic Commerce Research and Applications **3**: 294-310.

Wilkenfeld, J. K. and Zlotkin, G; (1992). "Multiagent Negotiation Under Time Constraints". Computer Science Technical Report Series. College Park**:** 1-62.

Wurman, P. R. and Walsh, W.E. (1998). The Michigan Internet AuctionBot: A Congurable Auction Server for Human and Software Agents. In Proceedings of the Second International Conference on Autonomous Agents, Minneapolis, MN.

Yabe, H. and Takahashi, T. (1991a). "Factorized quasi-Newton methods for nonlinear least squares problems." Mathematical Programming **51**: 75-100.

Yabe, H. and Takahashi, T. (1991b). "Numerical comparison among structured quasi-Newton methods for nonlinear least squares problems." Journal of the Operations Research Society of Japan **34**(3): 287-305.

Zeng, D. D. and Sycara, K. (1998). "Bayesian learning in negotiation" International Journal of Human-Computer Studies(48): 125-141.

Appendix

Matlab Generate

```
function[ActualPara, NoPara, StartingPoint, StartingPrice, ActualPrice,
LowerBound, UpperBound] =Generate(NoRepl)

LowerBound=[100,300,20,0,0];
UpperBound=[250,600,40,10,1];

NoRepl=1;

% Verify the sizes of lowerbound and upperbound
if size(LowerBound)~=size(UpperBound)
    error('The sizes of the first two parameters in function
GenerateStartingPoint must be the same!')
end

NoPara=size(LowerBound,2);  %record the number of parameters
time=1:10; % number of learning points

% begin to generate starting points under the given conditions
for j=1:NoRepl
    for i=1:NoPara
        ActualPara(i)=LowerBound(i)+rand*(UpperBound(i)-LowerBound(i));
        StartingPoint(i)=LowerBound(i)+rand*(UpperBound(i)-
LowerBound(i));
    end %for


    StartingPrice=faratin(StartingPoint,time);
end %for
```

Matlab Faratin

```
function P=faratin(Para,t)
% evaluation of faratin's function
% Pb=Pmin+exp((1-t/Tmax)^beta*log(K))*(Pmax-Pmin)
P=Para(1)+exp((1-t/Para(3)).^Para(4)*log(Para(5)))*(Para(2)-Para(1));
```

Matlab Quasi-Newton General

```
%counter=0;
%while counter < 30

tolerance=1E-2;
MaximumIteration=30;
NoPara=5;  % number of parameters
time=1:7;
NoTime=size(time,2);  % number of time

iteration=0;
```

```
ConvergeInfor=0;
DivergeInfor=0;

TotalTime=0;
NoFunEval=0;
NoFunEvalAlg=0;

[ActualPara]=Generate1;
[StartingPoint]=Generate1;
ActualPrice=faratin(ActualPara, time);
f=faratin(StartingPoint, time);
r=f-ActualPrice;
SSE=norm(r)^2;
%plot(time, ActualPrice, time, f);
LowerBound=[100,300,20,0,0];
UpperBound=[250,600,40,10,1];

syms h1 h2 h3 h4 h5
for i=1:NoPara
    eval(['SymbolG(',int2str(i),')=h',int2str(i),';']);  % SymbolG(i)=xi
end

for i=1:NoTime
    FuncF(i)=faratin(SymbolG,time(i));  % function F
   % ActualPrice(i)=faratin(ActualPara,time(i)); % actual price value
end
%************************************************

J=jacobian(FuncF.',SymbolG); % matrix D
Jt=transpose(J);
%NoFunEval=NoFunEval+NoFunEvalAlg;
gradient=Jt*r';
Ak=Jt*J;
h1=StartingPoint(1);
h2=StartingPoint(2);
h3=StartingPoint(3);
h4=StartingPoint(4);
h5=StartingPoint(5);
Ak=eval(Ak);
Hess=Ak;
gradient=eval(gradient);
trial=-pinv(Hess);
J=eval(J);
Jt=eval(Jt);
x=StartingPoint;
d=-pinv(Hess)*gradient;
%alpha=0.5*eye(5,5);
alpha=1;
decentd=(alpha*d)';

%%%%%%%%%%%%%%%%%%%
% iteration begins
%%%%%%%%%%%%%%%%%%%

while ((SSE>tolerance)&(iteration<MaximumIteration));
      iteration=iteration+1;  % record iteration number
```

```
      fprintf('.')

    syms alpha1

      alphavect=x+alpha1*d';
      %sizealphavect=size(alphavect)
      F=faratin(alphavect,time) ;
      R=Res(F, ActualPrice);

for i=1:5
 if (x(i)+ decentd(i)>LowerBound(i) & x(i)+ decentd(i)<UpperBound(i));
      x_1(i)=x(i)+decentd(i)
 else
     x_1(i)=x(i)+0.4*decentd(i)
 end
end
         % elseif((x(i)+ decentd(i)<LowerBound(i) | x(i)+
decentd(i)>UpperBound(i)))
          %    x_1=(i)+decentd(i)*0.1

                  for i=1:NoTime
          f(i)=faratin(x_1,time(i));

     end


       h1=x_1(1);
       h2=x_1(2);
       h3=x_1(3);
       h4=x_1(4);
       h5=x_1(5);

        J_1=jacobian(FuncF.',SymbolG);
        Jt_1=transpose(J_1);
        r_1=f-ActualPrice;    % Matrix Y
        J_1=eval(J_1);
        Jt_1=eval(Jt_1);
        gradient_1=(Jt_1)*(r_1)';
        B=Jt_1*J_1;
        SSE=norm(r_1)^2%standard square error

        if SSE>100
            alpha=1;
            else
            alpha=0.2;
        end
    %Calculations%%%%%%%%%%%%%%%%%%%%%%%%%%

    Jtrial=J_1-J;
    sk=x_1-x;
    yk=(gradient_1-gradient)';
    Ak=gen(sk, yk, J_1, Jt_1);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    try,
```

```
         Hess=B+Ak;
         detHess=det(Hess);

      if det(Hess)==0;
         fprintf('s'); % stand for singular
      end

         trial=-pinv(Hess);
         d=-pinv(Hess)*gradient_1;       % minimal norm solution for
singular case
         decentd=(alpha*d)';
         NoFunEval=NoFunEval+1;
     catch,
        fprintf('\nDiverge: Error in calculating matrix B in iteration
%d. However, continue...',iteration)
        ConvergeIndex='diverge'
        EndTime=cputime;

        EndingPoint=x;  % converge ending point
        NormSA=norm((StartingPoint-ActualPara)./ActualPara); % norm
between starting point and actual parameter
        NormEA=norm((EndingPoint-ActualPara)./ActualPara); % norm
between ending point and actual parameter
        SSEPara=norm(EndingPoint-ActualPara)^2;  % unnormalized SSE
between parameters.
        SSERevPara=norm((EndingPoint-ActualPara)./ActualPara)^2; %SSE
between actual parameters and estimated parameters.

       break;
    end



    if SSE<=tolerance
        fprintf('Converge!')
        ConvergeIndex='converge';

        EndingPoint=x;  % converge ending point
        NormSA=norm((StartingPoint-ActualPara)./ActualPara); % norm
between starting point and actual parameter
        NormEA=norm((EndingPoint-ActualPara)./ActualPara); % norm
between ending point and actual parameter
        SSEPara=norm(EndingPoint-ActualPara)^2;  % unnormalized SSE
between parameters.
        SSERevPara=norm((EndingPoint-ActualPara)./ActualPara)^2; %SSE
between actual parameters and estimated parameters.

    elseif iteration==MaximumIteration
        fprintf('Diverge!');
        ConvergeIndex='diverge';
        %TimeElapse=EndTime-StartTime;
        EndingPoint=x;
        NormSA=norm((StartingPoint-ActualPara)./ActualPara); % norm
between starting point and actual parameter
        NormEA=norm((EndingPoint-ActualPara)./ActualPara); % norm
between ending point and actual parameter
```

```
        SSEPara=norm(EndingPoint-ActualPara)^2;  % unnormalized SSE
between parameters.
        SSERevPara=norm((EndingPoint-ActualPara)./ActualPara)^2; %SSE
between actual parameters and estimated parameters.

      end % if

  %changing variables%%%%%%%%%%%%%%%%%%
        x=x_1;
        J=J_1;
        Jt=Jt_1;
        r=r_1;
        gradient=gradient_1;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   end % while

  EndPrice=faratin(EndingPoint, time);
  StartingPrice=faratin(StartingPoint, time);

  plot(time, f, time, ActualPrice);
  subplot(2,1,1); plot(time,ActualPrice, time, StartingPrice);
  subplot(2,1,2); plot(time, ActualPrice, time, EndPrice);
```

Matlab Quasi-Newton General Update

```
function Ak=gen(sk, yk, J_1, Jt_1);

Ak=pinv(sk)*(yk-sk*Jt_1*J_1);
```

Matlab Quasi-Newton DGW

```
tolerance=1E-2;
MaximumIteration=30;
NoPara=5;  % number of parameters
time=1:5;
NoTime=size(time,2);  % number of time

iteration=0;
ConvergeInfor=0;
DivergeInfor=0;

TotalTime=0;
NoFunEval=0;
NoFunEvalAlg=0;

[ActualPara]=Generate1;
[StartingPoint]=Generate1;
ActualPrice=faratin(ActualPara, time);
f=faratin(StartingPoint, time);
r=f-ActualPrice;
SSE=norm(r)^2;
```

```
%plot(time, ActualPrice, time, f);
LowerBound=[100,300,20,0,0];
UpperBound=[250,600,40,10,1];

syms h1 h2 h3 h4 h5
for i=1:NoPara
    eval(['SymbolG(',int2str(i),')=h',int2str(i),';']);  % SymbolG(i)=xi
end

for i=1:NoTime
    FuncF(i)=faratin(SymbolG,time(i));  % function F
   % ActualPrice(i)=faratin(ActualPara,time(i)); % actual price value
end
%***********************************************

J=jacobian(FuncF.',SymbolG); % matrix D
Jt=transpose(J);
%NoFunEval=NoFunEval+NoFunEvalAlg;
gradient=Jt*r';
Ak=Jt*J;
h1=StartingPoint(1);
h2=StartingPoint(2);
h3=StartingPoint(3);
h4=StartingPoint(4);
h5=StartingPoint(5);
Ak=eval(Ak);
Hess=Ak;
gradient=eval(gradient);
trial=-pinv(Hess);
J=eval(J);
Jt=eval(Jt);
x=StartingPoint;
d=-pinv(Hess)*gradient;
%alpha=0.5*eye(5,5);
alpha=0.5;
decentd=(alpha*d)';

%%%%%%%%%%%%%%%%%%%
% iteration begins
%%%%%%%%%%%%%%%%%%%

while ((SSE>tolerance)&(iteration<MaximumIteration));
         iteration=iteration+1;  % record iteration number
      fprintf('.')

    syms alpha1

      alphavect=x+alpha1*d';
      %sizealphavect=size(alphavect);
      alpha_lb=0;
      alpha_ub=3;

      F=faratin(alphavect,time) ;

         R=Res(F, ActualPrice);
```

```
% end %---LOOP STOP

   %disp(alpha1)
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
         if (x+ decentd >LowerBound & x+ decentd<UpperBound);
             x_1=x+decentd;
        fprintf('no adj')
         else
             decentd=decentd./6;
             x_1=x+decentd
         fprintf('adjust')
         end % if
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  for i=1:NoTime
       f(i)=faratin(x_1,time(i));
       NoFunEval=NoFunEval+1;
  end


    h1=x_1(1);
    h2=x_1(2);
    h3=x_1(3);
    h4=x_1(4);
    h5=x_1(5);

     J_1=jacobian(FuncF.',SymbolG);
     Jt_1=transpose(J_1);
     r_1=f-ActualPrice;    % Matrix Y
     J_1=eval(J_1);
     Jt_1=eval(Jt_1);
     gradient_1=(Jt_1)*(r_1)';
     B=Jt_1*J_1;
     SSE=norm(r_1)^2;%standard square error


 %Calculations%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 Jtrial=J_1-J;
 sk=x_1-x
 skt=transpose(sk);
 yk=(gradient_1-gradient)'
 ykt=transpose(yk);
 beta=(r_1*r')/(r*r')

 if beta > 1;
     beta=1;
 end
 disp(beta)
 v=(J_1-J)'*r_1';
 G=(v-beta*Ak*sk');
 Gt=transpose(G);
 Ak_1=DGW(beta,G,Ak,yk,ykt,Gt,skt);
 pause
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 try,
```

```
         Hess=B+Ak;
         detHess=det(Hess);

      if det(Hess)==0;
         fprintf('s'); % stand for singular
      end

         trial=-pinv(Hess);
         d=-pinv(Hess)*gradient_1;       % minimal norm solution for
singular case
         decentd=(alpha*d)';
         NoFunEval=NoFunEval+1;
     catch,
        fprintf('\nDiverge: Error in calculating matrix B in iteration
%d. However, continue...',iteration)
        ConvergeIndex='diverge'
        EndTime=cputime;
        %TimeElapse=EndTime-StartTime;
        EndingPoint=x;  % converge ending point
        NormSA=norm((StartingPoint-ActualPara)./ActualPara); % norm
between starting point and actual parameter
        NormEA=norm((EndingPoint-ActualPara)./ActualPara); % norm
between ending point and actual parameter
        SSEPara=norm(EndingPoint-ActualPara)^2;  % unnormalized SSE
between parameters.
        SSERevPara=norm((EndingPoint-ActualPara)./ActualPara)^2; %SSE
between actual parameters and estimated parameters.

infor=[StartingPoint,EndingPoint,SSE,iteration,NormSA,NormEA,NoFunEval];
       break;
    end



    if SSE<=tolerance
        fprintf('Converge!')
        ConvergeIndex='converge';
        EndTime=cputime;
        %TimeElapse=EndTime-StartTime;
        EndingPoint=x;  % converge ending point
        NormSA=norm((StartingPoint-ActualPara)./ActualPara); % norm
between starting point and actual parameter
        NormEA=norm((EndingPoint-ActualPara)./ActualPara); % norm
between ending point and actual parameter
        SSEPara=norm(EndingPoint-ActualPara)^2;  % unnormalized SSE
between parameters.
        SSERevPara=norm((EndingPoint-ActualPara)./ActualPara)^2; %SSE
between actual parameters and estimated parameters.

infor=[StartingPoint,EndingPoint,SSE,iteration,NormSA,NormEA,NoFunEval];
    elseif iteration==MaximumIteration
        fprintf('Diverge!');
        ConvergeIndex='diverge';
        EndTime=cputime;
        %TimeElapse=EndTime-StartTime;
        EndingPoint=x;
```

```
        NormSA=norm((StartingPoint-ActualPara)./ActualPara); % norm
between starting point and actual parameter
        NormEA=norm((EndingPoint-ActualPara)./ActualPara); % norm
between ending point and actual parameter
        SSEPara=norm(EndingPoint-ActualPara)^2;  % unnormalized SSE
between parameters.
        SSERevPara=norm((EndingPoint-ActualPara)./ActualPara)^2; %SSE
between actual parameters and estimated parameters.

infor=[StartingPoint,EndingPoint,SSE,iteration,NormSA,NormEA,NoFunEval];
      end % if

  %changing variables%%%%%%%%%%%%%%%%%%
        x=x_1;
        J=J_1;
        Jt=Jt_1;
        r=r_1;
        gradient=gradient_1;
        Ak=Ak_1;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   end % while

  EndPrice=faratin(EndingPoint, time);
  StartingPrice=faratin(StartingPoint, time);

  plot(time, f, time, ActualPrice);
  subplot(2,1,1); plot(time,ActualPrice, time, StartingPrice);
  subplot(2,1,2); plot(time, ActualPrice, time, EndPrice);
  %counter=counter+1;
 %fprintf('[%d]',counter)
 %end
%end%countrer
```

Matlab Quasi-Newton DGW Update

```
function Ak_1=DGW(beta,G,Ak,yk,ykt,Gt,skt)

Ak_1=beta*Ak+(G*yk+ykt*Gt)/(skt*yk)-((skt*Gt)/(skt*yk)^2)*ykt*yk
```

Matlab Biggs

```
%counter=0;
%while counter < 30

tolerance=1E-2;
MaximumIteration=30;
NoPara=5;  % number of parameters
time=1:5;
NoTime=size(time,2);  % number of time
```

```
iteration=0;
ConvergeInfor=0;
DivergeInfor=0;

TotalTime=0;
NoFunEval=0;
NoFunEvalAlg=0;

[ActualPara]=Generate1;
[StartingPoint]=Generate1;
ActualPrice=faratin(ActualPara, time);
f=faratin(StartingPoint, time);
r=f-ActualPrice;
SSE=norm(r)^2;
%plot(time, ActualPrice, time, f);
LowerBound=[100,300,20,0,0];
UpperBound=[250,600,40,10,1];

syms h1 h2 h3 h4 h5
for i=1:NoPara
    eval(['SymbolG(',int2str(i),')=h',int2str(i),';']);  % SymbolG(i)=xi
end

for i=1:NoTime
    FuncF(i)=faratin(SymbolG,time(i));  % function F
   % ActualPrice(i)=faratin(ActualPara,time(i)); % actual price value
end
%************************************************

J=jacobian(FuncF.',SymbolG); % matrix D
Jt=transpose(J);
%NoFunEval=NoFunEval+NoFunEvalAlg;
gradient=Jt*r';
Ak=Jt*J;
h1=StartingPoint(1);
h2=StartingPoint(2);
h3=StartingPoint(3);
h4=StartingPoint(4);
h5=StartingPoint(5);
Ak=eval(Ak);
Hess=Ak;
gradient=eval(gradient);
trial=-pinv(Hess);
J=eval(J);
Jt=eval(Jt);
x=StartingPoint;
d=-pinv(Hess)*gradient;
%alpha=0.5*eye(5,5);
alpha=1;
decentd=(alpha*d)';

%%%%%%%%%%%%%%%%%%%%
% iteration begins
%%%%%%%%%%%%%%%%%%%%

while ((SSE>tolerance)&(iteration<MaximumIteration));
```

```
      iteration=iteration+1;  % record iteration number
      fprintf('.')

    syms alpha1

      alphavect=x+alpha1*d';
      %sizealphavect=size(alphavect);
      alpha_lb=0;
      alpha_ub=3;

      F=faratin(alphavect,time) ;

         R=Res(F, ActualPrice);

      %disp(alpha1)
      for i=1:NoPara
            if (x(i)+ decentd(i)>LowerBound(i) & x(i)+
decentd(i)<UpperBound(i));
                x_1(i)=x(i)+decentd(i);
            else
                x_1(i)=x(i);
            end % if
      end %for

     for i=1:NoTime
          f(i)=faratin(x_1,time(i));
          NoFunEval=NoFunEval+1;
     end


       h1=x_1(1);
       h2=x_1(2);
       h3=x_1(3);
       h4=x_1(4);
       h5=x_1(5);

        J_1=jacobian(FuncF.',SymbolG);
        Jt_1=transpose(J_1);
        r_1=f-ActualPrice;    % Matrix Y
        J_1=eval(J_1);
        Jt_1=eval(Jt_1);
        gradient_1=(Jt_1)*(r_1)';
        B=Jt_1*J_1;
        SSE=norm(r_1)^2;%standard square error


    %Calculations%%%%%%%%%%%%%%%%%%%%%%%%%%

    Jtrial=J_1-J;
    sk=x_1-x;
    skt=transpose(sk)
    yk=(gradient_1-gradient)';
    ykt=transpose(yk)
    beta=(r_1*r')/(r*r')
    v=(J_1-J)'*r_1';
```

```
    G=(v-beta*Ak*sk')
    Gt=transpose(G);
    Ak_1=biggs(G,Gt,sk,Ak,beta)

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    try,
         Hess=B+Ak;
         detHess=det(Hess);

      if det(Hess)==0;
         fprintf('s'); % stand for singular
      end

         trial=-pinv(Hess);
         d=-pinv(Hess)*gradient_1;       % minimal norm solution for
singular case
         decentd=(alpha*d)';
         NoFunEval=NoFunEval+1;
     catch,
        fprintf('\nDiverge: Error in calculating matrix B in iteration
%d. However, continue...',iteration)
        ConvergeIndex='diverge'
        EndTime=cputime;
        %TimeElapse=EndTime-StartTime;
        EndingPoint=x;  % converge ending point
        NormSA=norm((StartingPoint-ActualPara)./ActualPara); % norm
between starting point and actual parameter
        NormEA=norm((EndingPoint-ActualPara)./ActualPara); % norm
between ending point and actual parameter
        SSEPara=norm(EndingPoint-ActualPara)^2;  % unnormalized SSE
between parameters.
        SSERevPara=norm((EndingPoint-ActualPara)./ActualPara)^2; %SSE
between actual parameters and estimated parameters.

infor=[StartingPoint,EndingPoint,SSE,iteration,NormSA,NormEA,NoFunEval];
       break;
    end



    if SSE<=tolerance
        fprintf('Converge!')
        ConvergeIndex='converge';
        EndTime=cputime;
        %TimeElapse=EndTime-StartTime;
        EndingPoint=x;  % converge ending point
        NormSA=norm((StartingPoint-ActualPara)./ActualPara); % norm
between starting point and actual parameter
        NormEA=norm((EndingPoint-ActualPara)./ActualPara); % norm
between ending point and actual parameter
        SSEPara=norm(EndingPoint-ActualPara)^2;  % unnormalized SSE
between parameters.
        SSERevPara=norm((EndingPoint-ActualPara)./ActualPara)^2; %SSE
between actual parameters and estimated parameters.

infor=[StartingPoint,EndingPoint,SSE,iteration,NormSA,NormEA,NoFunEval];
```

```
    elseif iteration==MaximumIteration
        fprintf('Diverge!');
        ConvergeIndex='diverge';
        EndTime=cputime;
        %TimeElapse=EndTime-StartTime;
        EndingPoint=x;
        NormSA=norm((StartingPoint-ActualPara)./ActualPara); % norm
between starting point and actual parameter
        NormEA=norm((EndingPoint-ActualPara)./ActualPara); % norm
between ending point and actual parameter
        SSEPara=norm(EndingPoint-ActualPara)^2;  % unnormalized SSE
between parameters.
        SSERevPara=norm((EndingPoint-ActualPara)./ActualPara)^2; %SSE
between actual parameters and estimated parameters.

infor=[StartingPoint,EndingPoint,SSE,iteration,NormSA,NormEA,NoFunEval];
      end % if

  %changing variables%%%%%%%%%%%%%%%%%
        x=x_1;
        J=J_1;
        Jt=Jt_1;
        r=r_1;
        gradient=gradient_1;
        Ak=Ak_1;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   end % while

  EndPrice=faratin(EndingPoint, time);
  StartingPrice=faratin(StartingPoint, time);

  plot(time, f, time, ActualPrice);
  subplot(2,1,1); plot(time,ActualPrice, time, StartingPrice);
  subplot(2,1,2); plot(time, ActualPrice, time, EndPrice);
```

Matlab Biggs Update

```
function Ak_1=biggs(G,Gt,sk,beta,Ak)

Ak_1=beta*Ak+(G*Gt)/(sk'*Gt)
```

Matlab Alpha Select

```
function x_1=alpha_select(decentd, x);
x
LowerBound=[100,300,20,0,0];
UpperBound=[250,600,40,10,1];
counter=0;
q=[0 0 0 0 0];

while sum(q)<5;
```

```
counter=(1+counter);

  for i=1:5
   if(x(i)+ decentd(i)/counter <UpperBound(i) & x(i)+ decentd(i)/counter
>LowerBound(i) );
  q(i)=1;
   else
  q(i)=0 ;
   end%if
  end%for

  if sum(q)==5
    x_1=x+ decentd./counter;
  break
  end
end %while
```

Matlab Alpha Range

```
function[alpha,min_E]=alpha_range(alpha_lower,alpha_upper,lowest_possibl
e_alpha,E,alpha1,SSE);

    steps=[0.5,0.1]
end
for i=1:length(steps);
    incr=steps(i);

[alpha,min_E]=check_range_alpha(alpha_lower,incr,alpha_upper,E,alpha1);
    if(alpha==lowest_possible_alpha);
        alpha_upper=alpha +incr;
        alpha_lower=alpha;
    else
        alpha_upper=alpha;
        lower_alpha=alpha-incr;
         upper_alpha=alpha+incr;

[l_alpha,l_min_E]=check_range_alpha(lower_alpha,incr,lower_alpha,E,alpha
1);

[u_alpha,u_min_E]=check_range_alpha(upper_alpha,incr,upper_alpha,E,alpha
1);
        if(abs(l_min_E)<abs(u_min_E));
             alpha_lower=lower_alpha;
            alpha_upper;
           %fprintf('\n Lower alpha:%f\t Upper alpha: %f\t at step size:
%f\t \n',alpha_lower,alpha_upper,incr);
         else
            alpha_lower=alpha_upper;
            alpha_upper=upper_alpha;
            %fprintf('\n Lower alpha:%f\t Upper alpha: %f\t at step
size: %f\t \n',alpha_lower,alpha_upper,incr);

        end
```

```
    end

end
```

Matlab Check Alpha Range

```
function[alpha,min_E]=check_range_alpha(alpha_lower,incr,alpha_upper,E,a
lpha1);
      range=alpha_lower:incr:alpha_upper;% generating values for alphal
      range=range';% creating a column array
      min_E=0.0;
      alpha=alpha_lower;
      fid=fopen('e_cal.txt','A'); % opens file result.txt in appen mode
      %fprintf('\n current step size: %f',incr);
      for i=1:size(range),;
          alpha_1=range(i);%alpha_one
          EE=subs(E,alpha1, alpha_1);% replace alpha1 with the value of
alpha_1

          if i==1
              min_E=abs(EE); % initialize min_E with first value
              alpha=alpha_1;

          else
              if abs(EE) < min_E;
                  min_E=abs(EE);
                  alpha=alpha_1;
              end
          end
      fclose(fid);
```

Stats

```
function [NormEA, SSERevPara]=Stats(Endpoint, StartingPointv, time,
ActualParav)
    EndPrice=faratin(Endpoint, time);
    StartingPrice=faratin(StartingPointv, time);
    NormSA=norm((StartingPointv-ActualParav)./ActualParav); % norm
between starting point and actual parameter
    NormEA=norm((Endpoint-ActualParav)./ActualParav); % norm between
ending point and actual parameter
    SSEPara=norm(Endpoint-ActualParav)^2;  % unnormalized SSE between
parameters.
    SSERevPara=norm((Endpoint-ActualParav)./ActualParav)^2; %SSE between
actual parameters and estimated parameters.
```

Cholesky Matrix Decomposition

```
function [L,D,E,pneg]=mchol(G)
%
%  n gives the size of the matrix.
%
n=size(G,1);
%
%  gamma, zi, nu, and beta2 are quantities used by the algorithm.
%
gamma=max(diag(G));
zi=max(max(G-diag(diag(G))));
nu=max([1,sqrt(n^2-1)]);
beta2=max([gamma, zi/nu, 1.0E-15]);
%
%  Initialize diag(C) to diag(G).
%
C=diag(diag(G));
%
%  Loop through, calculating column j of L for j=1:n
%

L=zeros(n);
D=zeros(n);
E=zeros(n);

for j=1:n,
    bb=[1:j-1];
    ee=[j+1:n];

    %
    %  Calculate the jth row of L.
    %
    if (j > 1),
        L(j,bb)=C(j,bb)./diag(D(bb,bb))';
    end;
    %
    %  Update the jth column of C.
    %
    if (j >= 2),
        if (j < n),
            C(ee,j)=G(ee,j)-(L(j,bb)*C(ee,bb)')';
        end;
    else
        C(ee,j)=G(ee,j);
    end;
    %
    % Update theta.
    %
    if (j == n)
        theta(j)=0;
```

```
    else
        theta(j)=max(abs(C(ee,j)));
    end;
    %
    %  Update D
    %
    D(j,j)=max([eps,abs(C(j,j)),theta(j)^2/beta2]');
    %
    % Update E.
    %
    E(j,j)=D(j,j)-C(j,j);

    ind=[j*(n+1)+1 : n+1 : n*n]';
    C(ind)=C(ind)-(1/D(j,j))*C(ee,j).^2;


end;

ind=[1 : n+1 : n*n]';
L(ind)=1;

%  if needed, find a descent direction.
%
if ((nargout == 4) & (min(diag(C)) < 0.0))
    [m,col]=min(diag(C));
    rhs=zeros(n,1);
    rhs(col)=1;
    pneg=L'\rhs;
else
  pneg=[];
end;


return
```

Hooke-Jeeves

```
function
StartingPointv_HJ=H_J(StartingPointv,ActualPrice,LowerBound,UpperBound,t
ime, NoPara)
%--------------------------------
f_sp=faratin(StartingPointv,time);
r_sp=f_sp-ActualPrice;
SSE_sp=norm(r_sp)^2;


x0=StartingPointv';
k=0.2;
div=2;
e=[100,0,0,0,0;
    0,100,0,0,0;
    0,0,2,0,0;
    0,0,0,0.2,0;
```

```
    0,0,0,0,0.2];
alpha=1;
delta=0.00001;
x1=x0;

initial_HJ=1;
step1=1;
step2=1;
step3=1;
step4=1;
NoIter=0;
fail=0;

while ((step1 | step2 | step3 | step4 |initial_HJ) & ~fail) % or

    NoIter=NoIter+1;
    if NoIter>200
        fail=1;
        fprintf('f');
    end;
    x=x1;
    initial_HJ=0;
    for i=1:NoPara % size(x0,1)
        x_old=x;
        y_old=faratin(x_old,time);
        r_old=y_old-ActualPrice;
        SSE_old=norm(r_old)^2;
        x_new = x_old + k * e(:,i);     % update point
        y_new=faratin(x_new,time);
        r_new=y_new-ActualPrice;
        SSE_new=norm(r_new)^2;


        if SSE_new >= SSE_old

         x_new = x_old - k * e(:,i); % update point
         y_new=faratin(x_new,time);
         SSE_new=norm(r_new)^2;
         k = -k;

         if SSE_new >= SSE_old
                x = x_old;
           else
                x = x_new;
           end %if
          else
          x = x_new;
        end %if
     end %for

    f_temp1=faratin(x,time);
    r_temp1=f_temp1-ActualPrice;
    SSE_temp1=norm(r_temp1)^2;

    f_temp2=faratin(x0,time);
    r_temp2=f_temp2-ActualPrice;
```

```
    SSE_temp2=norm(r_temp2)^2;
 %-------------------------------------------------------------------------
------------

if SSE_temp1<SSE_temp2
        step1=1;
        x1 = x + alpha * (x - x0);
        x0 = x;

else
      step1 = 0;

       if x1~=x0
           x1 = x0;
           step2 = 1;
       else
           step2 = 0;
            k=k/div;
            if abs(k)>=delta
              step3 = 1;
            else
              step3 = 0;
            end %if
         end %if
    end %if
xt=x;
StartingPointv_HJ_t=x';
for i=1:NoPara
    if StartingPointv_HJ_t(i)<LowerBound(i)
       StartingPointv_HJ_t(i)=LowerBound(i);
    elseif StartingPointv_HJ_t(i)>UpperBound(i)
        StartingPointv_HJ_t(i)=UpperBound(i);
    end%if
end%for
if StartingPointv_HJ_t ~= xt';
     x1=StartingPointv_HJ_t';
     step4=1;
else
    step4=0;
    x1=StartingPointv_HJ_t';
end
end %while
StartingPointv_HJ=x1';
```

Directory

```
Generate_run;

for n=0:3

  switch n
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      case 0

          Parameters_4;
          Jac_4;
          algo=0;
          conind=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     for i=0:11

        switch i

        case 0
        for Q=1:NoRepl;
          GN_B2;
        end

        LP_4_conind_GN_B2=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_4_conind_GN_B2,2,'H4')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,2,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,1,'A3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,1,'B3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,1,'C3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,1,'D3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,1,'E3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,29,'P3')
        conind=0;

        clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

            case 1
        for Q=1:NoRepl;
          GN_B2_HJ;
        end
```

```
        LP_4_conind_GN_B2_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_4_conind_GN_B2_HJ,4,'H4')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,4,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,3,'A3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,3,'B3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,3,'C3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,3,'D3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,3,'E3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,29,'W3')

        conind=0;
        clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

            case 2
        for Q=1:NoRepl;
          GN_B3;
        end

        LP_4_conind_GN_B3=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_4_conind_GN_B3,6,'H4')

xlswrite('C:\MATLAB701\work\All_Results',LP_4_conind_GN_B2_HJ,6,'H4')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,5,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,5,'A3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,5,'B3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,5,'C3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,5,'D3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,5,'E3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,29,'AD3')

        conind=0;

           clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1


            case 3
        for Q=1:NoRepl;
          GN_B3_HJ;
        end

        LP_4_conind_GN_B3_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_4_conind_GN_B3_HJ,8,'H4')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,8,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,7,'A3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,7,'B3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,7,'C3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,7,'D3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,7,'E3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,29,'AK3')
```

```
        conind=0;

            clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

            case 4
        for Q=1:NoRepl;
          QNG_B2;
        end

        LP_4_conind_QNG_B2=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_4_conind_QNG_B2,10,'H4')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,10,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,9,'A3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,9,'B3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,9,'C3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,9,'D3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,9,'E3')
         xlswrite('C:\MATLAB701\work\All_Results',Endpointt,29,'AR3')

        conind=0;

           clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

             case 5
        for Q=1:NoRepl;
          QNG_B2_HJ;
        end

        LP_4_conind_QNG_B2_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_4_conind_QNG_B2_HJ,12,'H4')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,12,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,11,'A3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,11,'B3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,11,'C3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,11,'D3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,11,'E3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,29,'AY3')

        conind=0;

            clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1


            case 6
        for Q=1:NoRepl;
          QNG_B3;
        end
        LP_4_conind_QNG_B3=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_4_conind_QNG_B3,14,'H4')
```

```
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,14,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,13,'A3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,13,'B3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,13,'C3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,13,'D3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,13,'E3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,29,'BF3')


        conind=0;
           clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

            case 7
        for Q=1:NoRepl;
          QNG_B3_HJ;
        end
        LP_4_conind_QNG_B3_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_4_conind_QNG_B3_HJ,16,'H4')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,16,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,15,'A3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,15,'B3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,15,'C3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,15,'D3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,15,'E3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,29,'BM3')

        conind=0;

          clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

            case 8
        for Q=1:NoRepl;
          QNGC_B2;
        end
        LP_4_conind_QNGC_B2=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_4_conind_QNGC_B2,18,'H4')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,18,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,17,'A3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,17,'B3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,17,'C3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,17,'D3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,17,'E3')
         xlswrite('C:\MATLAB701\work\All_Results',Endpointt,29,'BT3')

        conind=0;
           clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

            case 9
        for Q=1:NoRepl;
          QNGC_B2_HJ;
        end
```

```
        LP_4_conind_QNGC_B2_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_4_conind_QNGC_B2_HJ,20,'H4')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,20,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,19,'A3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,19,'B3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,19,'C3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,19,'D3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,19,'E3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,29,'CA3')

        conind=0;
        clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

             case 10
        for Q=1:NoRepl;
          QNGC_B3;
        end

        LP_4_conind_QNGC_B3=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_4_conind_QNGC_B3,22,'H4')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,22,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,21,'A3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,21,'B3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,21,'C3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,21,'D3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,21,'E3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,29,'CH3')

        conind=0;

        clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

             case 11
        for Q=1:NoRepl;
          QNGC_B3_HJ;
        end

        LP_4_conind_QNGC_B3_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_4_conind_QNGC_B3_HJ,24,'H4')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,24,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,23,'A3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,23,'B3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,23,'C3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,23,'D3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,23,'E3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,29,'CO3')

        conind=0;
         clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        end
     end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      case 1

          Parameters_6;
          Jac_6;
          algo=0;
          conind=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     for i=0:11

        switch i

            case 0
        for Q=1:NoRepl;
          GN_B2;
        end

        LP_6_conind_GN_B2=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_6_conind_GN_B2,2,'H5')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,2,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,1,'G3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,1,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,1,'I3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,1,'J3')
        xlswrite('C:\MATLAB701\work\All_Results',iteration,1,'K3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,30,'P3')

        conind=0;

      clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1
             case 1
        for Q=1:NoRepl;
          GN_B2_HJ;
        end

        LP_6_conind_GN_B2_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_6_conind_GN_B2_HJ,4,'H5')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,4,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,3,'G3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,3,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,3,'I3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,3,'J3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,3,'K3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,30,'W3')
        conind=0;
       clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

             case 2
```

```
        for Q=1:NoRepl;
          GN_B3;
        end

        LP_6_conind_GN_B3=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_6_conind_GN_B3,6,'H5')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,6,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,5,'G3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,5,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,5,'I3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,5,'J3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,5,'K3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,30,'AD3')
        conind=0;
        clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

             case 3
        for Q=1:NoRepl;
          GN_B3_HJ;
        end

        LP_6_conind_GN_B3_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_6_conind_GN_B3_HJ,8,'H5')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,8,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,7,'G3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,7,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,7,'I3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,7,'J3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,7,'K3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,30,'AK3')
        conind=0;
       clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

           case 4
        for Q=1:NoRepl;
          QNG_B2;
        end

        LP_6_conind_QNG_B2=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_6_conind_QNG_B2,10,'H5')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,10,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,9,'G3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,9,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,9,'I3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,9,'J3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,9,'K3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,30,'AR3')
        conind=0;
         clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1
```

```
            case 5
        for Q=1:NoRepl;
          QNG_B2_HJ;
        end

        LP_6_conind_QNG_B2_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_6_conind_QNG_B2_HJ,12,'H5')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,12,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,11,'G3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,11,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,11,'I3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,11,'J3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,11,'K3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,30,'AY3')
        conind=0;
   clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

            case 6
        for Q=1:NoRepl;
          QNG_B3;
        end

        LP_6_conind_QNG_B3=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_6_conind_QNG_B3,14,'H5')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,14,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,13,'G3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,13,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,13,'I3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,13,'J3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,13,'K3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,30,'BF3')
        conind=0;
       clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

           case 7
        for Q=1:NoRepl;
          QNG_B3_HJ;
        end
        LP_6_conind_QNG_B3_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_6_conind_QNG_B2_HJ,16,'H5')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,16,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,15,'G3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,15,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,15,'I3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,15,'J3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,15,'K3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,30,'BM3')
        conind=0;
  clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

            case 8
```

```
        for Q=1:NoRepl;
          QNGC_B2;
        end

        LP_6_conind_QNGC_B2=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_6_conind_QNGC_B2,18,'H5')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,18,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,17,'G3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,17,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,17,'I3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,17,'J3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,17,'K3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,30,'BT3')
        conind=0;
 clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

             case 9
        for Q=1:NoRepl;
          QNGC_B2_HJ;
        end

        LP_6_conind_QNGC_B2_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_6_conind_QNGC_B2_HJ,20,'H5')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,20,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,19,'G3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,19,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,19,'I3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,19,'J3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,19,'K3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,30,'CA3')
        conind=0;
 clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

             case 10
        for Q=1:NoRepl;
          QNGC_B3;
        end

        LP_6_conind_QNGC_B3=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_6_conind_QNGC_B3,22,'H5')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,21,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,21,'G3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,21,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,21,'I3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,21,'J3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,21,'K3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,30,'CH3')
        conind=0;
      clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1
```

```
            case 11
          for Q=1:NoRepl;
            QNGC_B3_HJ;
          end

          LP_6_conind_QNGC_B3_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_6_conind_QNGC_B3_HJ,24,'H5')
          xlswrite('C:\MATLAB701\work\All_Results',NoRepl,23,'H3')
          xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,23,'G3')
          xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,23,'H3')
          xlswrite('C:\MATLAB701\work\All_Results',timer1,23,'I3')
          xlswrite('C:\MATLAB701\work\All_Results',Qt,23,'J3')
          xlswrite('C:\MATLAB701\work\All_Results',iterationt,23,'K3')
          xlswrite('C:\MATLAB701\work\All_Results',Endpointt,30,'CO3')
          conind=0;
      clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

          end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        case 2

            Parameters_8;
            Jac_8;
            algo=0;
            conind=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       for i=0:11

          switch i

              case 0
          for Q=1:NoRepl;
            GN_B2;
          end

          LP_8_conind_GN_B2=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_8_conind_GN_B2,2,'H6')
          xlswrite('C:\MATLAB701\work\All_Results',NoRepl,2,'H3')
          xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,1,'M3')
          xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,1,'N3')
          xlswrite('C:\MATLAB701\work\All_Results',timer1,1,'O3')
          xlswrite('C:\MATLAB701\work\All_Results',Qt,1,'P3')
          xlswrite('C:\MATLAB701\work\All_Results',iterationt,1,'Q3')
          xlswrite('C:\MATLAB701\work\All_Results',Endpointt,31,'P3')
          conind=0;
 clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

              case 1
```

```
        for Q=1:NoRepl;
          GN_B2_HJ;
        end

        LP_8_conind_GN_B2_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_8_conind_GN_B2_HJ,4,'H6')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,4,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,3,'M3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,3,'N3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,3,'O3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,3,'P3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,3,'Q3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,31,'W3')
        conind=0;
 clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

             case 2
        for Q=1:NoRepl;
          GN_B3;
        end

        LP_8_conind_GN_B3=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_8_conind_GN_B3,6,'H6')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,5,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,5,'M3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,5,'N3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,5,'O3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,5,'P3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,5,'Q3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,31,'AD3')
        conind=0;
 clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

             case 3
        for Q=1:NoRepl;
          GN_B3_HJ;
        end

        LP_8_conind_GN_B3_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_8_conind_GN_B3_HJ,8,'H6')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,8,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,7,'M3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,7,'N3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,7,'O3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,7,'P3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,7,'Q3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,31,'AK3')
        conind=0;
       clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1
```

```
            case 4
        for Q=1:NoRepl;
          QNG_B2;
        end

        LP_8_conind_QNG_B2=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_8_conind_QNG_B2,10,'H6')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,10,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,9,'M3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,9,'N3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,9,'O3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,9,'P3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,9,'Q3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,31,'AR3')
        conind=0;
     clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

            case 5
        for Q=1:NoRepl;
          QNG_B2_HJ;
        end

        LP_8_conind_QNG_B2_HJ=conind


xlswrite('C:\MATLAB701\work\All_Results',LP_8_conind_QNG_B2_HJ,12,'H6')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,12,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,11,'M3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,11,'N3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,11,'O3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,11,'P3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,11,'Q3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,31,'AY3')
        conind=0;
         clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1


            case 6
        for Q=1:NoRepl;
          QNG_B3;
        end
        LP_8_conind_QNG_B3=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_8_conind_QNG_B3,14,'H6')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,14,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,13,'M3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,13,'N3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,13,'O3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,13,'P3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,13,'Q3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,31,'BF3')
        conind=0;
```

```
      clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

            case 7
        for Q=1:NoRepl;
          QNG_B3_HJ;
        end
        LP_8_conind_QNG_B3_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_8_conind_QNG_B3_HJ,16,'H6')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,16,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,15,'M3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,15,'N3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,15,'O3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,15,'P3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,15,'Q3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,31,'BM3')
        conind=0;

        clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

            case 8
        for Q=1:NoRepl;
          QNGC_B2;
        end

        LP_8_conind_QNGC_B2=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_8_conind_QNGC_B2,18,'H6')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,18,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,17,'M3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,17,'N3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,17,'O3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,17,'P3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,17,'Q3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,31,'BT3')
        conind=0;
 clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

            case 9
        for Q=1:NoRepl;
          QNGC_B2_HJ;
        end

        LP_8_conind_QNGC_B2_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_8_conind_QNGC_B2_HJ,20,'H6')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,20,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,19,'M3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,19,'N3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,19,'O3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,19,'P3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,19,'Q3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,31,'CA3')
```

```
        conind=0;
    clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

             case 10
        for Q=1:NoRepl;
          QNGC_B3;
        end

        LP_8_conind_QNGC_B3=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_8_conind_QNGC_B3,22,'H6')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,21,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,21,'M3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,21,'N3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,21,'O3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,21,'P3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,21,'Q3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,31,'CH3')
        conind=0;
       clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

             case 11
        for Q=1:NoRepl;
          QNGC_B3_HJ;
        end

        LP_8_conind_QNGC_B3_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_8_conind_QNGC_B3_HJ,24,'H6')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,24,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,23,'M3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,23,'N3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,23,'O3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,23,'P3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,23,'Q3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,31,'CO3')
        conind=0;
 clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

        end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      case 3

          Parameters_10;
          Jac_10;
          algo=0;
          conind=0;
```

```
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     for i=0:11

         switch i

               case 0
         for Q=1:NoRepl;
           GN_B2;
         end

         LP_10_conind_GN_B2=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_10_conind_GN_B2,2,'H7')
         xlswrite('C:\MATLAB701\work\All_Results',NoRepl,2,'H3')
         xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,1,'S3')
         xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,1,'T3')
         xlswrite('C:\MATLAB701\work\All_Results',timer1,1,'U3')
         xlswrite('C:\MATLAB701\work\All_Results',Qt,1,'V3')
         xlswrite('C:\MATLAB701\work\All_Results',iterationt,1,'W3')
         xlswrite('C:\MATLAB701\work\All_Results',Endpointt,32,'P3')
         conind=0;
 clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

              case 1
         for Q=1:NoRepl;
           GN_B2_HJ;
         end

         LP_10_conind_GN_B2_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_10_conind_GN_B2_HJ,4,'H7')
         xlswrite('C:\MATLAB701\work\All_Results',NoRepl,4,'H3')
         xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,3,'S3')
         xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,3,'T3')
         xlswrite('C:\MATLAB701\work\All_Results',timer1,3,'U3')
         xlswrite('C:\MATLAB701\work\All_Results',Qt,3,'V3')
         xlswrite('C:\MATLAB701\work\All_Results',iterationt,3,'W3')
         xlswrite('C:\MATLAB701\work\All_Results',Endpointt,32,'W3')
         conind=0;
         clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

              case 2
         for Q=1:NoRepl;
           GN_B3;
         end

         LP_10_conind_GN_B3=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_10_conind_GN_B3,6,'H7')
         xlswrite('C:\MATLAB701\work\All_Results',NoRepl,6,'H3')
         xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,5,'S3')
         xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,5,'T3')
         xlswrite('C:\MATLAB701\work\All_Results',timer1,5,'U3')
         xlswrite('C:\MATLAB701\work\All_Results',Qt,5,'V3')
```

```
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,5,'W3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,32,'AD3')
        conind=0;
        clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

             case 3
        for Q=1:NoRepl;
          GN_B3_HJ;
        end

        LP_10_conind_GN_B3_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_10_conind_GN_B3_HJ,8,'H7')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,8,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,7,'S3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,7,'T3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,7,'U3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,7,'V3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,7,'W3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,32,'AK3')
        conind=0;
          clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

             case 4
        for Q=1:NoRepl;
          QNG_B2;
        end

        LP_10_conind_QNG_B2=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_10_conind_QNG_B2,10,'H7')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,10,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,9,'S3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,9,'T3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,9,'U3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,9,'V3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,9,'W3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,32,'AR3')
        conind=0;
          clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

             case 5
        for Q=1:NoRepl;
          QNG_B2_HJ;
        end

        LP_10_conind_QNG_B2_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_10_conind_QNG_B2_HJ,12,'H7')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,12,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,11,'S3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,11,'T3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,11,'U3')
```

```
        xlswrite('C:\MATLAB701\work\All_Results',Qt,11,'V3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,11,'W3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,32,'AY3')
        conind=0;
       clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

            case 6
        for Q=1:NoRepl;
          QNG_B3;
        end
        LP_10_conind_QNG_B3=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_10_conind_QNG_B3,14,'H7')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,14,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,13,'S3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,13,'T3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,13,'U3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,13,'V3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,13,'W3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,32,'BF3')
        conind=0;
        clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

            case 7
        for Q=1:NoRepl;
          QNG_B3_HJ;
        end

        LP_10_conind_QNG_B3_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_10_conind_QNG_B3_HJ,16,'H7')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,16,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,15,'S3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,15,'T3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,15,'U3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,15,'V3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,15,'W3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,32,'BM3')
        conind=0;
     clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

            case 8
        for Q=1:NoRepl;
          QNGC_B2;
        end

        LP_10_conind_QNGC_B2=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_10_conind_QNGC_B2,18,'H7')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,18,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,17,'S3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,17,'T3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,17,'U3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,17,'V3')
```

```
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,17,'W3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,32,'BT3')
        conind=0;
         clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1
iterationt_1 timer1 timer_1 Endpointt Endpointt_1

             case 9
        for Q=1:NoRepl;
          QNGC_B2_HJ;
        end

        LP_10_conind_QNGC_B2_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_10_conind_QNGC_B2_HJ,20,'H7'
)
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,20,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,19,'S3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,19,'T3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,19,'U3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,19,'V3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,19,'W3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,32,'CA3')
        conind=0;
   clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

             case 10
        for Q=1:NoRepl;
          QNGC_B3;
        end

        LP_10_conind_QNGC_B3=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_10_conind_QNGC_B3,22,'H7')
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,22,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,21,'S3')
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,21,'T3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,21,'U3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,21,'V3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,21,'W3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,32,'CH3')
        conind=0;
 clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1

             case 11
        for Q=1:NoRepl;
          QNGC_B3_HJ;
        end

        LP_10_conind_QNGC_B3_HJ=conind

xlswrite('C:\MATLAB701\work\All_Results',LP_10_conind_QNGC_B3_HJ,24,'H7'
)
        xlswrite('C:\MATLAB701\work\All_Results',NoRepl,24,'H3')
        xlswrite('C:\MATLAB701\work\All_Results',NormEA_1,23,'S3')
```

```
        xlswrite('C:\MATLAB701\work\All_Results',SSERevPara_1,23,'T3')
        xlswrite('C:\MATLAB701\work\All_Results',timer1,23,'U3')
        xlswrite('C:\MATLAB701\work\All_Results',Qt,23,'V3')
        xlswrite('C:\MATLAB701\work\All_Results',iterationt,23,'W3')
        xlswrite('C:\MATLAB701\work\All_Results',Endpointt,32,'CO3')
        conind=0;
 clear NormEA_1 NormEA SSERevPara_1 SSERevPara Qt Qt_1 iterationt_1
timer1 timer_1 Endpointt Endpointt_1
        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  end
end
```