Department of Computing, Imperial College London

Inferring Tennis Match Progress from In-Play Betting Odds

Author:
Xinzhuo Huang

Supervisor:
Dr. William Knottenbelt

Second Marker:
Dr. Jeremy Bradley

June 22, 2011

Acknowledgements

I would like to thank my supervisor Dr. William Knottenbelt not only for the support and guidance he has given me but also his constant enthusiasm for this work. I would also like to thank Dr. Jeremy Bradley for agreeing to be my second marker.

Abstract

Online tennis betting has become hugely popular, especially in-play betting where trading occurs whilst an event is taking place. During matches, millions of pounds in bets are routinely matched on the online betting exchange, Betfair, alone. An obstacle faced by both human traders as well as automated trading software is the reliability and availability of live match scores on which to base trading decisions. The aim of this work is to determine whether and to what extent it is possible to infer tennis match score purely from the analysis of live betting odds.

We derive a hierarchical Markov tennis model that enables us to calculate each player's expected match-winning probability from any point in a match. By comparing this probability to that implied by odds from the Betfair market, we detect when points are scored. Therefore, when run from start to finish of a match, the current score should always be known to our program. Testing our software against real matches shows that this idea is not only possible but in fact capable of deducing the score of entire sets with few errors. Much enhancement is still needed before the software is suitable for replacing traditional sources of scoring information, however this could be the basis for future work.

Contents

1 Introduction

1.1 Online Sports Betting

Wagering on the outcomes of sports matches has always been popular but never more so than now. This is largely to do with the growth of the online betting market, booming in recent years, as focus increasingly shifts to the internet. The global online betting market value is estimated to reach over $7.8 billion by 2012, implying annual growth exceeding 11% over the past decade [1].

The landscape of online sports betting consists of two types of websites, traditional bookmakers and betting exchanges. Traditional bookmakers act as market makers. That is, they accept wagers on outcomes of events and maintains spreads on the two sides of bets in order to ensure itself profit. On the other hand, betting exchanges provide its customers with the ability to offer and take bets with each other, acting as a middle man. There is usually little or no restriction on the size or odds of a bet that can be offered on an exchange, only requiring that one or more opposing customers are willing to match it. An exchange usually profits from commission on net winnings and so the odds offered, not affected by bookmaker spreads, are truer reflections of probability of outcomes.

1.2 In-Play Trading on Exchanges

A feature that is readily provided by exchanges is in-play betting where trading is facilitated whilst an event is taking place. Instead of relying on pre-match estimations, bettors can make decisions based on live information of an event. In this way, odds placed on the market change quickly as participants react to progression of a match. The rate and amount by which the odds fluctuate in-play far exceeds that of a pre-match market and so gives rise to far greater potential profits. Traditional bookmakers sometimes also provide in-play betting, but this is generally more restricted due to larger risk involved for the bookmaker. In light of volatile price movements, a bookmaker may decide to widen the spread to hedge its risk, making the odds less attractive than on an exchange.

Similarly to financial markets, the aim of a trader is to buy low and sell high except here in the tennis market, the odds are traded instead of financial instruments. For example, if a trader speculates that a player will worsen in performance later on, they could back them immediately with the aim of laying them later when the odds change such that it requires an amount less than than the profit to cover the original liability. Thus if the trader is successful, it is possible to offset all liabilities regardless of match outcome and so guaranteeing profit.

1.3 Betfair

Betfair[3] was the world's first and now largest internet betting exchange. It has grown rapidly since its launch in 2000 and claims to have over 3 million clients and a turnover of more than $50 million per week. Across its products, Betfair processes more transactions on an average day than all European stock markets combined. In fact at its inception, the founders set out to create a new way of betting modelled on stock exchanges. People are allowed to buy and sell, or back and lay as it is referred to with regards to betting, the outcomes of sporting events.

A second reason why Betfair is interesting to us is because it provides users with an API allowing connection of software to its exchanges. This allows users to easily obtain market information as well as place bets through their own programs. Indeed a number of commercial software have been developed with the aim of aiding traders make profit on the exchange, many of them perform automated trading.

1.4 The Tennis Market

Tennis has become one of the most heavily traded sports online and particularly attractive for in-play traders. Horse racing and football have traditionally been popular among sports gamblers, however in recent years tennis has seen an enormous increase in bet volume, particularly on exchanges [2]. The final match of Wimbledon 2008 was a perfect illustration of just how popular trading on tennis has become. During that one match between Roger Federer and Rafael Nadal, a total of £50 million worth of bets were matched on Betfair alone. More recently in the French Open 2011 final, over £40 million was matched, a figure routinely reached in Grand Slam finals and other high profile matches.

From a trading point of view, tennis is a unique sport. The sequence of points are played within fixed intervals, usually no more than half a minute with rallies lasting roughly 10 seconds. As points are gained and lost regularly, in-play traders offer odds that vary with equal frequency. As we have mentioned, the greater the amplitude and frequency of odds changes, the more opportunity there is to profit. Another attraction lies in the availability of matches. Professional tennis is played in eleven months of a year with four major grand slam events; Wimbledon, the French, US and Australian Open as well as a series of prestigious tournaments including those of the Association of Tennis Professional (ATP) tour. This combination of profiting opportunity and availability of matches are the main contributing factors for the growing popularity of tennis trading.

1.4.1 Tennis Trading

It is estimated that the majority in-play tennis bets placed on Betfair are due to traders as opposed to recreational gamblers. That is, bets are placed following strategies with the aim of closing trades out so that profit is ensured regardless of outcome. Although success of trading still requires fortunate speculation, there are many strategies which are claimed to maximise success rate.

Scalping is the strategy of trading short term odds fluctuations. The idea is to make a large number of small profits. For instance, backing a player before a game point then

laying after the point is scored at a lower price. Another strategy is to back the favourite should they under-perform at the start of a match as it is often the case that they regain control later.

The reader may already be able to see that in-play tennis betting is situational, that is, prediction of odds movements and implementation of strategies depend on the state and score of the current match. This information usually comes from televised broadcasts, web streams or live scoring websites. Ideally, this information would be instant however it is often not the case. For instance, matches which are transmitted by satellite are often delayed by at least 5 seconds, potentially disadvantaging the trader. This idea is supported by findings presented in Brown (2010)[11]. Other times, matches may not be broadcast at all or unavailable to be viewed.

Automated trading is also used in conjunction with in-play trading although at present there is no known reliable way for programs to determine match score independent of human input. Only rudimentary methods such as screen-scraping scoring websites exist, which are neither practical nor reliable. Because of this, trading bots can only be set to perform trades in response to triggers caused purely by odds movements without taking match progress or score into account as a human would. Although various studies have been carried out on modelling of tennis matches, prediction of winning probabilities, even analysis of in-play tennis odds on Betfair, there has been no successful work undertaken on inferring the score of a match from betting odds.

1.5 Our Goals

Easton and Uylangco (2010)[12] provides point-by-point comparisons of a player's match-winning probability described by a mathematical model to that implied by betting odds. Figure 1, taken from Easton and Uylangco (2010), show that there is a strong correlation between the two probabilities implying high level of market efficiency. Even on inspection of peaks and troughs of the graph by eye, we can make reasonable estimations about when each player gains and loses games. This is the basis of our idea and provides us with the question we wish to answer; namely, **whether or not it is possible to infer tennis scores through analysis of betting odds and if so, how and to what degree of effectiveness it can be achieved**.

Our ultimate aim is to create a piece of software that is capable of determining the score of tennis matches, in real-time, through nothing but analysis of live in-play online betting odds of the corresponding match. At the time of writing, there has been no successful work carried out to this intention so our work will be as much about determining whether the ideas are viable as it is about producing a working product. Our motivations stem equally from both academic interest and the potential for more intelligent automated trading algorithms, which are aware of the current state and score of a match, that in turn may have wider implications for the online sports betting community.

Aside from the primary objective of score inference our work may produce other results of interest. Our idea will compare predicted match-winning probabilities, according to a stochastic tennis model, against that observed on the Betfair market testing the accuracy and assumptions of our model. In developing algorithms to intelligently recognise point scoring, we study the manner in which the market behaves in response to events throughout a match. Parts of the score inference program that we build may also be desirable as stand-

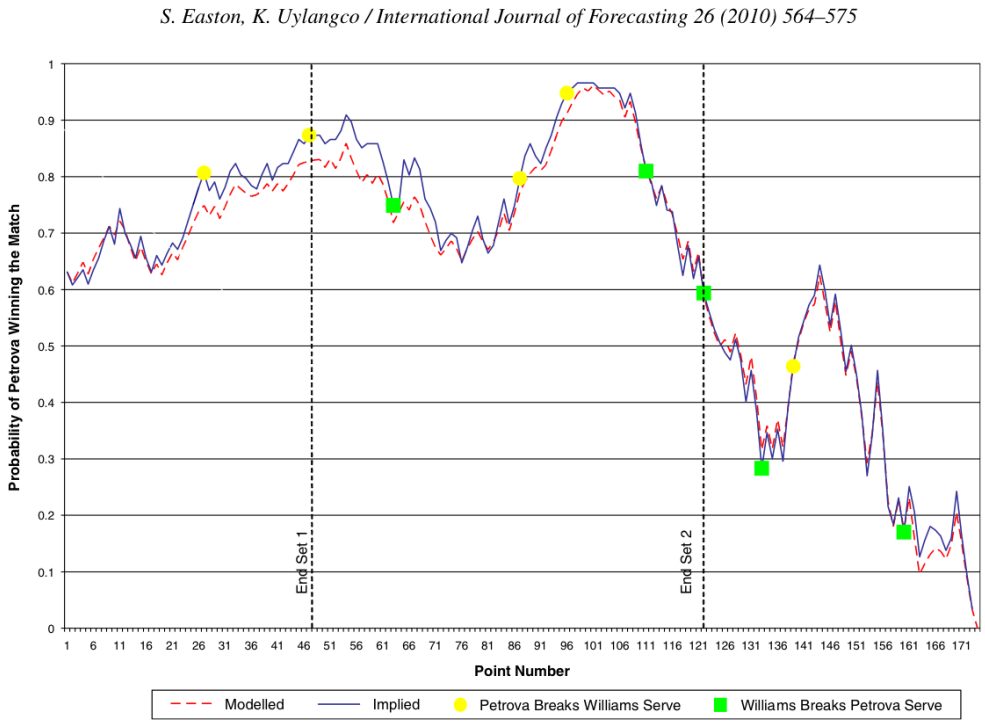


Figure 1: Probability of Petrova winning the match: Petrova vs. Williams (2007).

alone applications themselves, for instance, the engine that calculates expected match-winning probability is a useful tool in itself.

1.6 High Level System Overview

The core of our idea is to compare match-winning probabilities implied by odds from Betfair to expected match-winning probabilities according to a mathematical model of a tennis match, on a point-by-point basis. By continuously analysing implied and expected probabilities, we decide whether a player has scored based on a number of algorithms that we develop. If entirely successful, we would be able to detect whenever points are gained or lost, from beginning to end of a match thus knowing the score at all time. Expected match-winning probabilities will come from calculations based on a hierarchical Markov model of a tennis match. We can split the project into its three main parts:

1. Deriving a **hierarchical Markov tennis model** to create a **match-winning probability calculator** (section 3).

2. Accessing odds information from the Betfair market and processing it to **obtain implied match-winning probabilities**.

3. Developing heuristics including analysing expected and implied probabilities to **detect scoring and prevent erroneous inferences** (section 5).

1.6.1 Match-Winning Probability Calculator

The tennis match model that we create is based on the ideas of Lui (2001) and Klaassen and Magnus (2003), mentioned in section 2.1, linking four variables:

1. Point-winning probability of player 1 on serve
2. Point-winning probability of player 2 on serve
3. Current score
4. Match-winning probability of either player

We use this model to calculate match-winning probabilities corresponding to variations of the other three parameters; the two point-winning probabilities and current score. However, unlike in past studies where this is the stopping point, our aim is to compare these predicted probabilities to that implied by in-play betting odds. In other words, we aim to reverse the process to try to infer current score through the other three parameters.

We model every possible scoring in a match as a discrete state of a Markov chain, with appropriate probabilities of moving between them. In order to significantly reduce the number of states, we use a hierarchical structure separating the levels of *game*, *set* and *match* then link them in a recursive manner. Upon completion of the model, we will be able to perform steady-state analysis to find probabilities of reaching the winning states from any other, i.e. obtain match-winning probabilities.

1.6.2 Score Inference Analysis

We develop algorithms to decide when point scoring occurs upon comparison of expected and market-implied match-winning probabilities. Since betting is a human activity involving some speculation and psychological bias, the process of inferring scores is not exact. That is to say the bets being placed on the market may not always follow expectation exactly and so there are not precise methods that will be successful in all cases. However we develop heuristics to recognise patterns in odds changes and market behaviour to determine occurrences of point scoring. The range of heuristics we will use include the following:

- Calculating threshold probabilities corresponding to next possible scores and detecting crossings (section 5.2).
- Recalibration of point-winning probabilities in accordance to market changes (section 5.3).
- Recognising and ignoring large fluctuations after scoring (section 5.4).
- Averaging odds during periods of fluctuation (section 5.5).
- Recognising some large odds changes as signs of possible scoring (section 5.6).

A diagrammatic summary of the system overview is shown in figure 2. See appendix C for a more technical UML diagram.

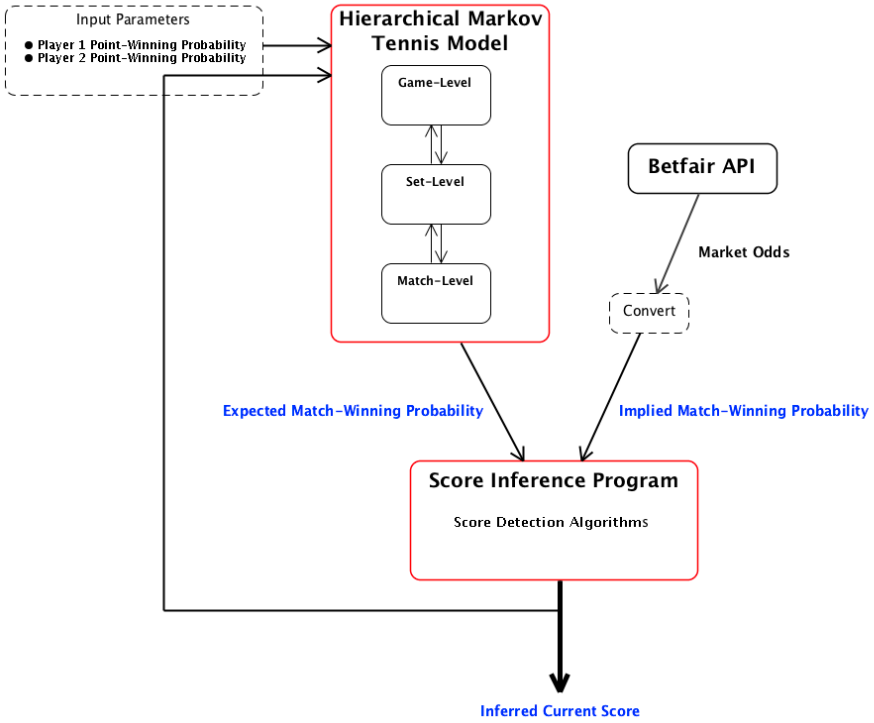


Figure 2: High Level System Diagram.

2 Contributions

Although the process of inferring tennis score from betting odds has not been attempted, there have been studies on tennis match modelling, determining point-winning probabilities and analysis of tennis betting odds that are relevant to our work. The key contributions of this work are now summarised below, with explicit references in later sections where applicable.

Before proceeding, the reader should be aware of at least the basics of tennis scoring structure and rules. A detailed description of tennis rules can be found in appendix A, but for now, it is sufficient to know that in order to win a match, a player must win a number of sets. To win a set, a player must win a number of games, which consists of a sequence of points played with the same player serving. Service alternates between the players after each game. We will see shortly that this game-set-match structure is the key in enabling

tennis to be modelled easily.

2.1 On Tennis Modelling

Klaassen and Magnus (2001)[13] showed, by analysing 90,000 points played at Wimbledon over 3 years, that although points scored in tennis are not independent, the deviations from this feature is small. As such, approximations in modelling under the assumption still provides good results in many cases.

Lui (2001)[14] offers a method of modelling tennis matches using a hierarchy of discrete time Markov chains, which we will formally introduce in section 3. The key assumption is that points are scored independently which allows all possible scores to be represented as a state in the Markov chain. Furthermore, the recursive structure of tennis scoring is exploited to greatly reduce the total number of states. The model, however, assumes the probability of a player winning points are the same whether they are serving or receiving a game. This is typically not true in tennis as significant advantage rests with the server. We will use Lui's idea as the basis to develop our own hierarchical Markov model but extend it to take differences in each player's service games into account.

Klaassen and Magnus (2003) link the service game point-winning probabilities of each player to their match-winning probabilities using a method that appears to be similar, though they do not fully explain it. They suggest that the point-winning probabilities may be derived from a combination of official player rankings and subjective judgement.

Barnett, Brown and Clarke (2003)[16] achieves something similar using complex formulas within Microsoft Excel. There are however two main extensions to the previous modelling ideas, improving upon it. Firstly, apart from the two point-winning parameters, an additional four are used; probabilities of missing the first serve and point-winning probabilities on the second serve for each player. This improvement comes at the cost of additional complexity. Including second serves would mean doubling the states in our Markov chains, therefore we decide that this should only be done if our first model is insufficiently accurate. The second revision to the model is allowing for players who are ahead in a set to increase their probability of winning the set, departing from the independent point scoring assumption. Dependence between points violates the Markov property, explained in section 3, however we may still incorporate this idea in our algorithms instead of the model itself.

Two notable papers that both unified earlier works are Newton and Keller (2005) [17] and O'Malley (2008)[18]. Both present explicit derivation and some analysis of what is now known as the tennis formula. The tennis formula describes probability of winning a match given probabilities of individual player winning points, equivalent to the idea presented in Lui (2001). The attraction in the tennis formula stems from it's succinctness, however it becomes more complex when we wish to compute winning probabilities from within a match rather than from the start. Since we require flexible calculations corresponding to any possible score, we remain with the idea of using Markov chains as in Lui (2001).

2.2 On Probabilities of Winning Points on Serve

The hierarchical Markov tennis model reduces calculation of match-winning probability to knowledge of point-winning probabilities of each player, therefore service game point-winning probabilities are key parameters to our program. We discuss how we estimate these parameters themselves in section 3.6. Klaassen and Magnus (2003) uses a combination of ranking data and subjective judgement whereas Barnett, Brown and Clarke (2003) use historical statistics to do this. Results of both papers suggest that their respective methods are sufficient in producing reasonably accurate results when predicting real matches.

Klaassen and Magnus (2001) concluded, through analysing 258 mens and 223 womens matches that the average of the sum of two players service game point-winning probabilities are 1.29 for men and 1.12 for women. For example in a mens match, if we knew that player 1 has 0.6 probability of winning points on player 1's serve, then the probability of player 2 winning a point on player 2's serve would most be approximately $1.29 - 0.6 = 0.69$. We approach its use with caution as it relies on the assumption of low distribution variance of the sums of point-winning probability pairs although Newman and Aslam (2009)[19] showed that a player's point-winning probability varies from match to match and can be modelled as Gaussian distributed random variables with relatively low variance.

2.3 On Analysis of Tennis Betting Odds

We have already mentioned Brown (2010) which presents findings that enforce the idea that a subset of the betting population are observing the action before the wider public, possibly due to delays in the television signal, and are betting using this informational advantage. This supports the potential usefulness of a score inference program to in-play traders.

Easton and Uylangco (2010), also already mentioned, offers comparisons between modelled and implied probabilities from in-play odds on Betfair. As is shown in figure 1, predicted and implied probabilities match closely. This is a key property that we rely on for the success of our work.

3

Match-Winning Probability Calculator

The ability to determine the match-winning likelihood is a fundamental part of our method and so at the heart of this project is the engine that performs such calculations. The implementation consists of two parts; an analyser for each tennis level and then a means of combining them. The inputs are point-winning probabilities, at the lowest level, with the parameters of each successive higher level being the result from the proceeding lower one. With this in mind, we derive our model of a tennis match.

3.1 Hierarchical Markov Tennis Model

3.1.1 Assumptions

The main assumption we make in our model is that the points scored are identically distributed and independent. As discussed in the previous section, Klaassen and Magnus (2001) suggests that this should still provide a good approximation. This assumption means that the probability of moving from one state to the next, where states correspond to scores, does not depend on previous information, only the present state. This lends the system to be easily described by a series of discrete-time Markov chains and allows us to perform analysis on their corresponding stochastic matrices. This is the technique described in Lui (2001). Unlike Lui's model, we will take into account the differences in point-winning probability depending on the current server as it is generally true that the server has an advantage.

3.1.2 Introducing Discrete-Time Markov Chains

Here, we give descriptions, definitions, properties and results of Markov chains needed later on. The following is a relatively brief introduction, for more comprehensive information, the reader is invited to refer to Falko Bause [20].

A Markov chain is a type of Markov process, a stochastic process for which the Markov property holds. The intuitive explanation of the Markov property is to say that the future of the process is determined only by the present state. However the process may have evolved to its present state does not influence the future. For the purpose of this project, we only concern ourselves with discrete-time Markov chains. That is, the case where the time spent in a state has a discrete distribution. From here on, when we say Markov chain, we refer to discrete-time Markov chains. The formal definition of a discrete-time Markov chain is as follows:

Definition 1. *The sequence* $\{X_n | n = 0, 1, 2, ...\}$ *is a discrete-time Markov chain provided* $P[X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, ..., X_0 = x_0] = P[X_{n+1} = x_{n+1} | X_n = x_n] \ \forall \ n \in \mathbb{N}$.

The expression on the right-hand side of this equation denotes the probability that the process goes from state x_n to state x_{n+1} when the time parameter is increased from n to $n + 1$.

It is perhaps easiest to understand the concept we require by considering the following well-known problem, named *Gamblers' Ruin*. The description is adapted from Falko Bause [20].

Two gamblers are betting on the outcome of an unlimited sequence of coin tosses. The first gambler always bets heads, which appears with probability p, $0 < p < 1$ *on every toss. The second gambler always bets tails, which appears with probability* $q = 1 - p$. *They start with a total of* C *chips between them. Whenever one gambler wins he has to give the other one chip. The game stops when one gambler runs out of chips (is ruined). Assume the gamblers start with* $C = 3$ *chips between them.*

What is the probability that a gambler is ruined, losing all chips, having started with a single chip?

We represent the above scenario with the Markov chain in figure 3. The labels in each state depict how many chips each gambler has and the set of all states describes all possible combinations of chip distribution. Arcs connecting states represent possible transitions from one state to the next with the respective probabilities. This is a *finite* Markov chain as there are a finite number of states, i.e. 4.

States 0 and 1 are *absorbing states*, where there are no transitions to any other state once they have been reached. These correspond to the cases where one gambler wins and the other is ruined. States 2 and 3 are *transient states*, where there are transitions between them as well as to the *absorbing states*.

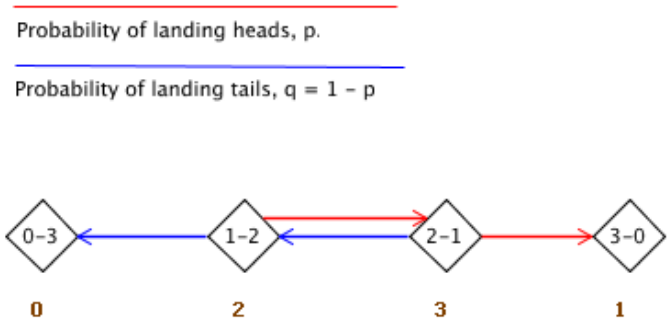


Figure 3: Gamblers' Ruin with 3 chips in total. State with label $i - j$ corresponds to gambler 1 having i chips and gambler 2 having j chips.

We now give a matrix representation of the Markov chain in figure 3. With the states labelled as in the diagram, entry $e_{i,j}$ of our matrix represents the probability of movement from state i to state j in a single step. Note that, for convenience, we refer to the first

row and column of the matrix as the 0^{th} row and column, respectively. Note also that the numbering of states, denoted by brown numbers in figures, is arbitrary.

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ q & 0 & 0 & p \\ 0 & p & q & 0 \end{pmatrix} \tag{1}$$

Upon inspection, we observe that P is in fact of the form:

$$P = \begin{pmatrix} I & 0 \\ R & Q \end{pmatrix} \tag{2}$$

where I is the identity matrix, 0 is the all-zero matrix, R is the matrix describing the movement from the transient to the absorbing states and Q is the matrix describing the movement amongst transient states. We remark that I appears in the top-left as once we reach an absorbing state then we remain there and 0 appears in top-right as it is impossible to move from an absorbing to a transient state. (2) is in fact a general form of any finite discrete-time Markov chains.

Since P represents a single transition in our Markov chain, P^2 represents our system after two transitions. We can generalise this to P^n describing our system after n transitions. Since we wished to know the probability of a gambler being ruined, having started with 1 chip (reaching state 0 from state 2), then we must first calculate P^∞ and look at the $(2, 0)$ entry.

Since the formula for matrix multiplication also applies to matrices written in block form, we can calculate the powers of P in terms of the matrices R and Q:

$$P^2 = \begin{pmatrix} I & 0 \\ R + QR & Q^2 \end{pmatrix}$$

or in general

$$P^n = \begin{pmatrix} I & 0 \\ N_n R & Q^n \end{pmatrix}$$

where $N_n = I + Q + Q^2 + ... + Q^{n-1} = \Sigma_{i=1}^{n} Q^{i-1}$.

Theorem 1. *When $n \to \infty$ then $Q^n \to 0$ and $N_n \to (I - Q)^{-1}$. In particular, the matrix $I - Q$ is invertible.*

It follows from Theorem 1 that

$$\lim_{n\to\infty} P^n = \begin{pmatrix} I & 0 \\ NR & 0 \end{pmatrix} \tag{3}$$

where in our example (1):

$$N = \begin{pmatrix} \frac{1}{1-pq} & \frac{p}{1-pq} \\ \frac{q}{1-pq} & \frac{1}{1-pq} \end{pmatrix}$$

and

$$NR = \begin{pmatrix} \frac{q}{1-pq} & 0 \\ 0 & \frac{p}{1-pq} \end{pmatrix}$$

so

$$\lim_{n\to\infty} P^n = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{q}{1-pq} & 0 & 0 & 0 \\ 0 & \frac{p}{1-pq} & 0 & 0 \end{pmatrix}$$

The $(2, 0)$ entry is $\frac{q}{1-pq}$ telling us that this is the probability of a gambler being ruined having started with 1 chip.

This concludes the introduction to Markov chains. The reader may already be able to see how the above may be applied to modelling various levels of a tennis match. The two gamblers are similar to two tennis players, the chip distribution similar to scoring combinations and the absorbing states in a tennis example will correspond to a player winning.

3.1.3 Modelling the *Game* Level

The three levels of tennis (game, set and match) can be modelled using separate Markov chains. They are linked in such a way that the probability of winning a game will be used in the set level and the probability of winning a set will be used in the match level.

We begin with the lowest level - the game level. Here, we are only concerned with two parameters, p, the probability that the server of the game wins a point and $q = 1 - p$, probability the receiver wins a point. These two probabilities correspond to the likelihood of moving along arcs connecting score states. To simplify calculations, we represent the following pairs of scores by the same state:

- 30 : 30 and 40 : 40 (*deuce*)
- 40 : 30 and *advantage server*
- 30 : 40 and *advantage receiver*

These three states may be visited any number of times before a game is won, similarly to the gambler's ruin situation where the gamblers may win and lose chips for an indefinite

time before being fully ruined. Figure 4 shows the Markov chain encompassing possible evolutions of scores in the game level. We label the absorbing states, where a player wins, as 0 and 1 so that we have the 2×2 identity matrix in the top left of our 17×17 matrix representation of this Markov chain.

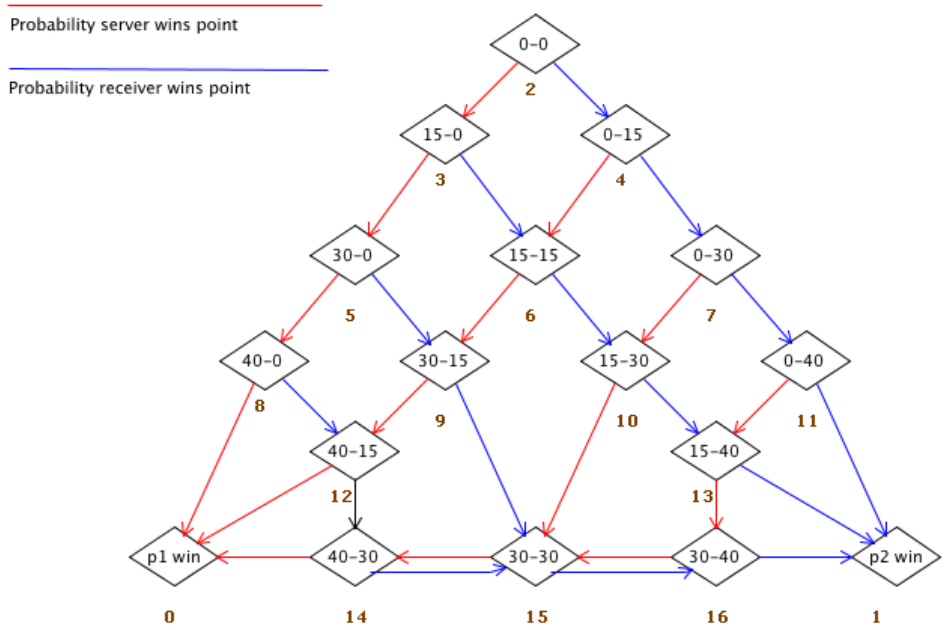


Figure 4: Markov chain showing possible evolution of point scores in a tennis *game*.

3.1.4 Modelling the *Set* Level

The set level, comprising of the most states and with the added complication of the tiebreaker, is the most difficult level to model. Figure 5 shows the possible evolution of game scores in a set. The probabilities corresponding to the arcs are the probabilities of the players winning a game, obtained from analysis of the game level (i.e. probability of moving from state 2 to 0 or 1 in figure 4). Because the rules of tennis dictate that the server alternates between each game, we need at least four parameters to specify the Markov chain: the probabilities of the server winning and losing when each of the two players serve. In figure 5, probabilities corresponding to player 1's service games are denoted by solid lines whilst probabilities corresponding to player 2's service games are shown using dotted lines.

If the score reaches $6-6$ in a set, a tiebreaker game is played to decide the set winner. We account for this scenario by adding two probabilities of moving from the $6-6$ state of figure 5 to the winning, absorbing, states. We develop a separate Markov chain, figure 6, to determine these two probabilities as the tiebreaker has a separate scoring structure of its own. The probabilities of the arcs in figure 6 are the probabilities of winning a point, like in the game level, however this time we need to take into account who is currently serving as service switches many times within a tiebreak game. Similarly to the *deuce* situation at the game level, winning a tiebreak game requires a player to be two points

clear of the opponent so we represent this situation with the relation between the last few states (0, 1, 48, 49, 50, 51, 52 and 53) as in figure 6.

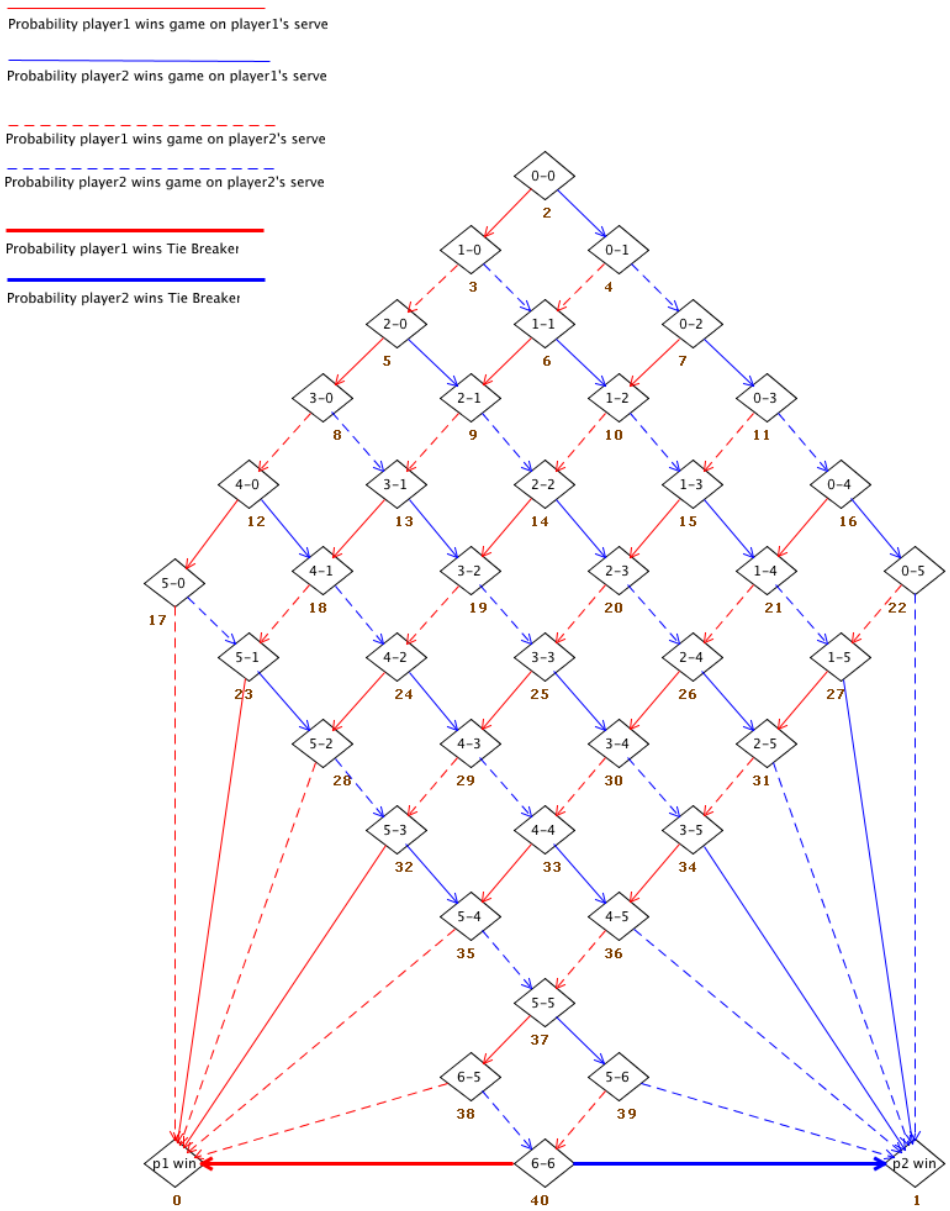


Figure 5: Markov chain showing possible evolution of game scores in a tennis *set*.

3.1.5 Modelling the *Match* Level

Finally, we have the match level. Rules vary with each tournament as to whether the match is played as best-of-three or best-of-five sets. Figure 7 shows the Markov chain of a best-of-five match. The probabilities of the arcs depend on probabilities of each player

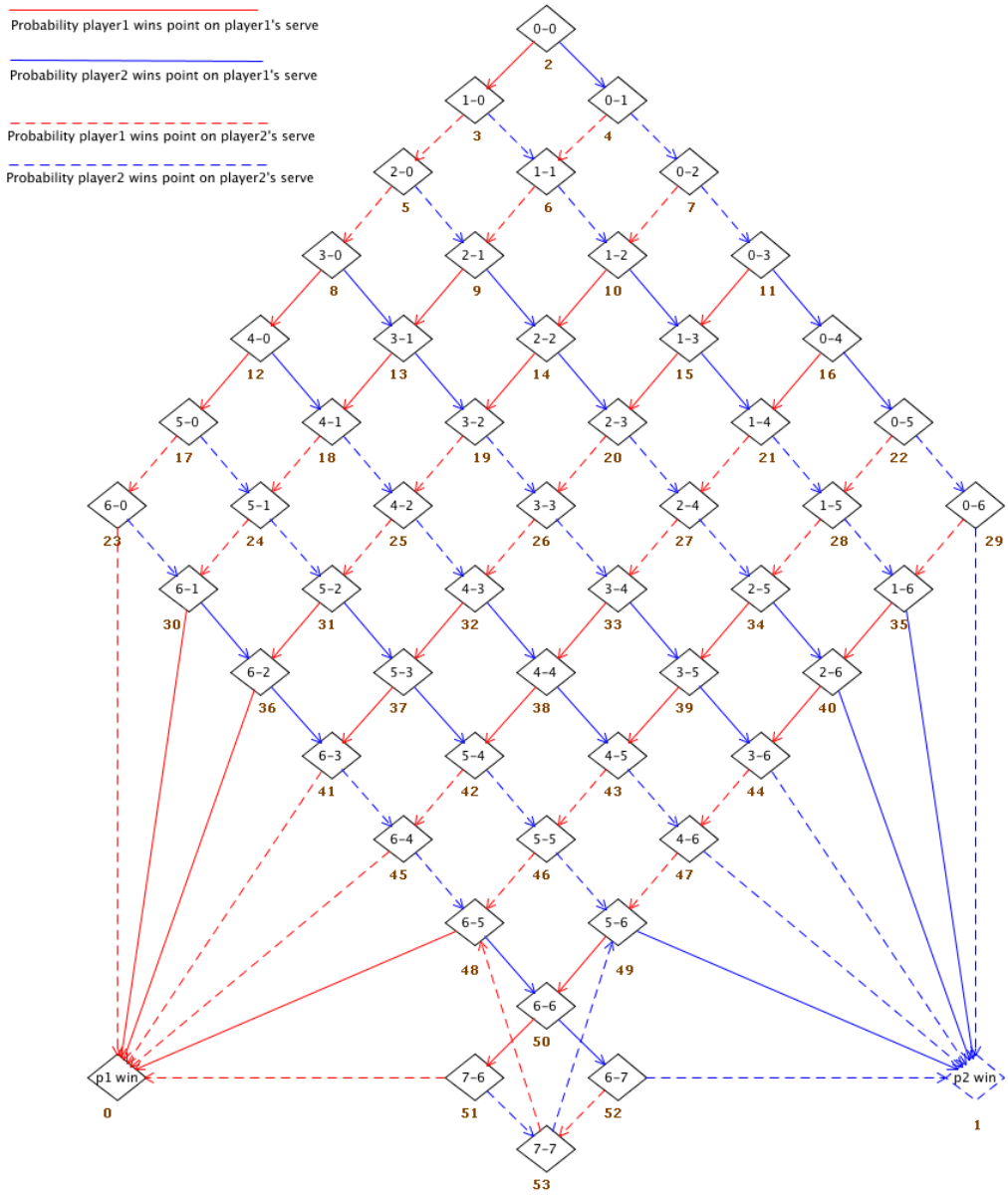


Figure 6: Markov chain showing possible evolution of point scores in a tennis *tiebreaker*.

winning the set level, similarly to the previous levels.

We now have a complete model of a tennis match using four interlinked Markov chains and can write down their corresponding matrices (of sizes 16×16, 41×41, 44×44 and 11×11). We note that a potential problem we may encounter in developing a score inference program is false-positive inferences caused by odds changing due to faults of service rather than points being scored. So far, we have not considered first and second services in our Markov chain. However if we discover that we need to include such cases then we could modify our Markov chains by adding extra states corresponding to first service faults. It may well be the case that the current model is sufficient for the purpose of score inference,

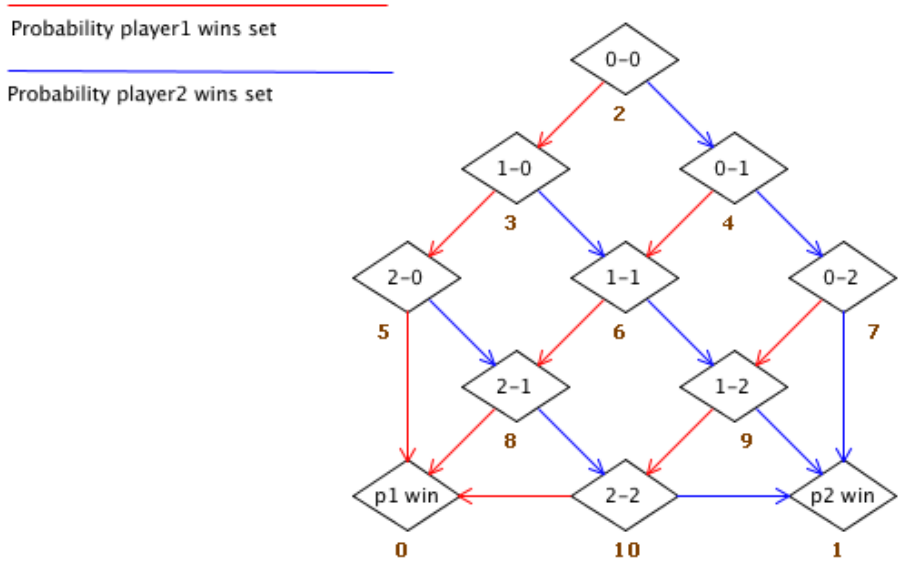


Figure 7: Markov chain showing possible evolution of set scores in a tennis *match*.

which we will not know until testing, so we proceed to implementation

3.2 Implementing a Single-Level Analyser

We wish to use Theorem 1 and its result, (3), on the match level Markov chain to find the probability of each player winning a match by recursively applying the same method to the lower levels in order to obtain probabilities of the arcs. To do so, we must compute the likelihood of each player probability of winning at each level i.e. the probabilities of reaching each of the absorbing states of the Markov chains.

Section 3 describes how to solve a generic Gambler's Ruin problem. This is analogous to our tennis problem and so we apply the same method to our models. We build our analyser to parse Markov chain data structures, creating corresponding matrices of the form in equation 2 and follow the method to calculate the matrix $\lim_{n\to\infty} P^n$ (equation 3). The entries $(i, 0)$ or $(i, 1)$ of $\lim_{n\to\infty} P^n$ gives us the probabilities of player 1 or player 2 winning, respectively, given the current score is that which corresponds to state i. Additionally, we can also compute the probability of score of state i becoming score of state j after n points e.g. from $0 - 15$ to $40 - 30$ in five points. This can be done by selecting entry (i, j) from P^n.

3.3 Linking All Levels

Input parameters to the single-level analyser are point-winning probabilities. These are used in the lowest level to compute game-winning probabilities which can be, in turn, used to initialise the Markov chain matrices of the set-level and similarly for the match

level. This gives a means of calculating match-winning probability from before a match commences. However, we also require match-winning probabilities from an arbitrary score, i.e. in-play. This is more complex as we need to take additional factors into account. Consider the following situation:

In a five set match, the score is 1 set each, 4 games to 1, and no points scored in the current game. The probabilities of moving from state 2 to 3, 4 to 6, 8 to 0 etc (i.e. player 1 winning a set) are all identical however it is different to that of from 6 to 8. This is because the third set is currently in-play and so the probability, instead of being that of from state 2 to 0 of the set-level, is in fact that of from state 18 to 0 (reflecting the current games score of $4 - 1$). This is denoted in figure 8.

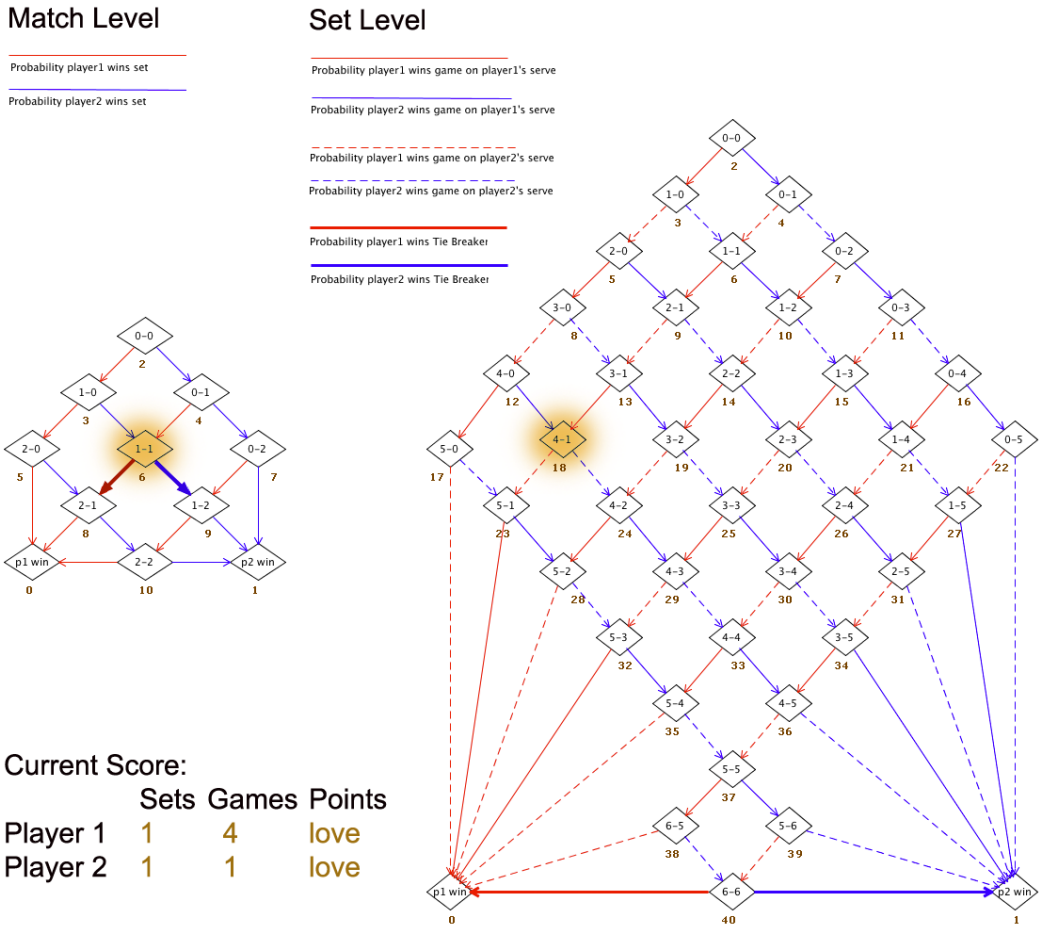


Figure 8: Markov chains of match and set levels. **Orange** highlights states of current score. Thicker **red arrow** and **blue arrow** on the match-level Markov chain denote the arcs with probabilities different to the others due to the scoring scenario.

Additionally, where it was previously sufficient to only have one set-level model, we now need two. Separate ones are required for when each player serves the first game in set because we keep track of who is serving the current game. Therefore we need to adjust the Markov chain matrices accordingly.

Pseudo-code of our recursive algorithm that calculates match-winning probabilities from

any point within a match, is given in 1. We call the function with the following parameters:

probOfWinning($p1Serve$, $p2Serve$, *match-level*, *score*)

where

- $p1Serve$ and $p2Serve$ are the probabilities of each player winning points on their own serve, obtained previously by some other means.
- *match-level* is the recursion level of the initial call to the recursive function.
- *score* is the current score of the match

At each level of recursion, the final step calculates the winning probability at that level using a single-level analyser, the workings of which is described in 3.2. Our implementation uses a data structure to hold current score information and uses methods that translates between current score and state numbers to index the Markov chain matrices.

3.4 Sample of Match-Winning Probability Calculations

We can now calculate match-winning probabilities from any point in a match. A small selection of results is given in figure 9 showing match-winning probabilities in a three-set match given various parameters.

	Point-Winning Probability of Each Player On Serve (p, q)					
Scenario	**(0.60, 0.57)**	**(0.65, 0.54)**	**(0.56, 0.56)**	**(0.55, 0.60)**	**(0.58, 0.62)**	**(0.60, 0.65)**
Pre-Match	0.653	0.549	0.500	0.255	0.303	0.264
1-0, 0-0	0.843	0.971	0.750	0.552	0.597	0.598
1-0, 5-1	0.871	0.977	0.795	0.610	0.441	0.603
1-1, 4-3	0.891	0.968	0.841	0.732	0.762	0.835
0-1, 0-0	0.364	0.688	0.250	0.109	0.133	0.113

Figure 9: Match-winning probabilities corresponding to a range of point-winning probabilities and match score scenarios. Note, the score column is given in the format *(Set-Score, Game-Score)*.

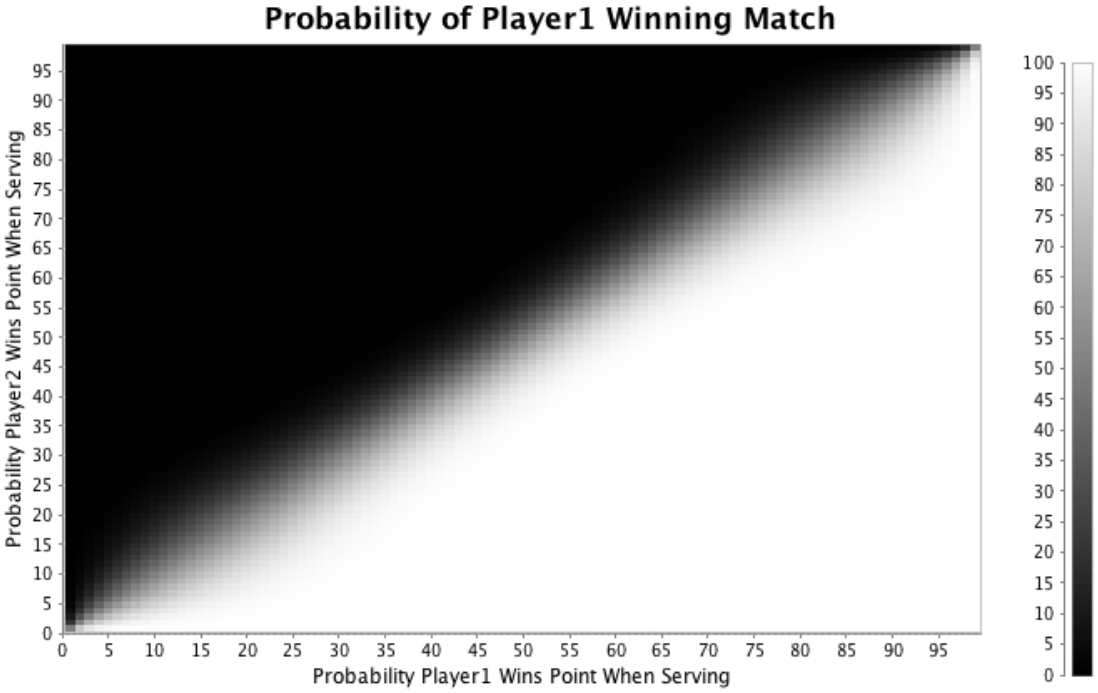
Figure 10a was created by plotting results from our program, using a pre-match *current score* parameter. Similarly, we can produce visualisations corresponding to other scoring situations such as shown in figure 10b, 10c, 10d and 10e, when player 1 is behind in the final set of a three set match.

3.5 Ensuring Correctness of the Match-Winning Probability Calculator

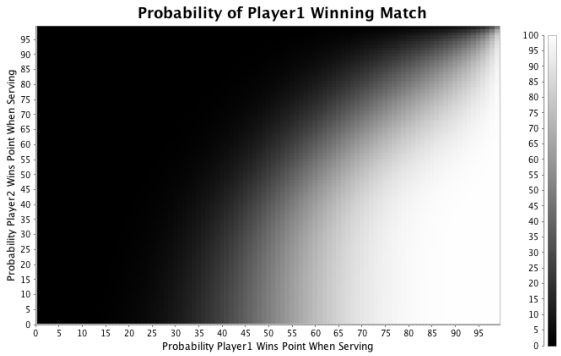
Algorithms used in the latter part of the project, to infer match score, depend heavily on knowing the correct expected match-winning probability. Therefore, it is crucial that the results produced by our calculator are accurate with respect to our model. We use two methods to validate our program:

Algorithm 1 *double* ← probOfWinning (double $p1Serve$, double $p2Serve$, Level *recursion-level*, CurrentScore s)

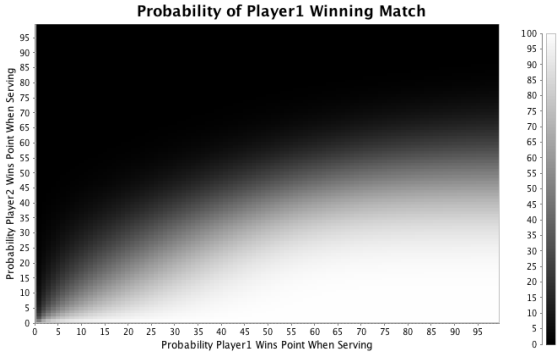
```
1: if recursion-level is game-level then
2:     MarkovChain m = game-level Markov chain;
3:     // initialise probabilities of moving between states of m
4:     for all states of m do
5:         probServerWinsPoint = p1Serve;
6:         probReceiverWinsPoint = 1 − p1Serve;
7:         // note that at the game level the server is always player 1, whoever is serving
           is taken care of at the set-level recursion call
8:     end for
9:     return Analyser.probabilityToAbsorbingStateFromCurrentScore(s, m);
10: end if

11: if recursion-level is set-level then
12:     MarkovChain m = set-level Markov chain;
13:     // initialise probabilities of moving between states of m
14:     for all states of m do
15:         if p1 serving then
16:             probP1WinsGame = probOfWinning(p1Serve, p2Serve, game-level,
                start);
17:             probP2WinsGame = 1 − probP1WinsGame;
18:         end if
19:         if p2 serving then
20:             probP2WinsGame = probOfWinning(p2Serve, p1Serve, game-level,
                start);
21:             probP1WinsGame = 1 − probP2WinsGame;
22:         end if
23:     end for
24:     //modify m according to current score
25:     if p1 serving then
26:         probP1WinsGameFromCurrent = probOfWinning(p1Serve, p2Serve,
            game-level, s);
27:         probP2WinsGameFromCurrent = 1 − probP1WinsGameFromCurrent;
28:     end if
29:     if p2 serving then
30:         probP2WinsGameFromCurrent = probOfWinning(p2Serve, p1Serve,
            game-level, s);
31:         probP1WinsGameFromCurrent = 1 − probP2WinsGameFromCurrent;
32:     end if
33:     return Analyser.probabilityToAbsorbingStateFromCurrentScore(s, m);
34: end if

35: if recursion level is match-level then
36:     MarkovChain m = match-level Markov chain;
37:     // initialise probabilities of moving between states of m
38:     for all states do
39:         probP1WinsSet = probOfWinning(p1Serve, p2Serve, set-level, start);
40:         probP2WinsSet = 1 − probP1WinsSet;
41:     end for
42:     //modify m according to current score
43:     probP1WinsSetFromCurrent = probOfWinning(p1Serve, p2Serve, set-level,
        s);
44:     probP2WinsSetFromCurrent = 1 − probP1WinsSetFromCurrent;
45:     return Analyser.probabilityToAbsorbingStateFromCurrentScore(s, m);
46: end if
```



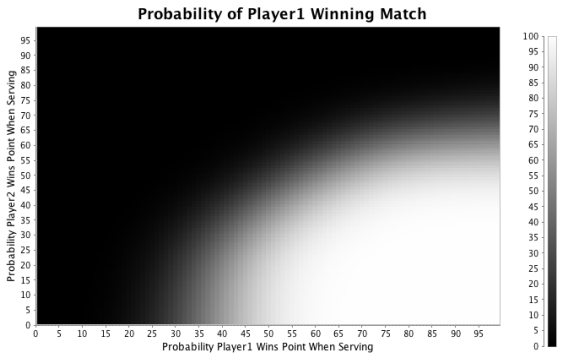
(a) Sets: $0-0$, Games: $0-0$. (Pre-Match)



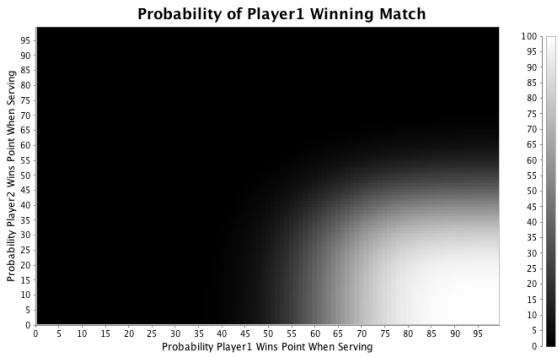
(b) Sets: $0-1$, Games: $4-5$. player 1 to serve.



(c) Sets: $0-1$, Games: $4-5$. player 2 to serve.



(d) Sets: $0-1$, Games: $1-3$. player 1 to serve.



(e) Sets: $0-1$, Games: $0-5$. player 1 to serve.

Figure 10: Charts relating each player's point-winning probabilities and player 1's match-winning probabilities in a three-set match. Axis labelled in percentages. Darker means lower probability to win.

- Compare our results to that in other works on tennis probability.
- Build a tennis match simulator then compare results given by our calculator to that of the simulator, under various parameters.

3.5.1 Comparison to Results in Other Works

Both O'Malley (2008) and Liu (2001) present a selection of data, relating point and match-winning probabilities, under different match scenarios. Liu (2001) shows pre-match match-winning probabilities of both three and five set matches without taking into account of tiebreakers. O'Malley (2008) presents in-play match-winning probabilities, in particular those of making a come-back from at least a break behind in a last set with the use of tiebreakers shown in figure 11.

Journal of Quantitative Analysis in Sports, Vol. 4 [2008], Iss. 2, Art. 15

Table 3: Probability of Recovering from a Breakdown to Win a Set

Initial Score	Point Winning Probabilities, (p, q)			
	(0.67, 0.38)	(0.62, 0.33)	(0.645, 0.355)	(0.50, 0.50)
4-5[r]	0.135	0.056	0.089	0.250
3-5	0.116	0.043	0.073	0.125
2-5[s]	0.100	0.034	0.060	0.063
3-4[r]	0.227	0.091	0.150	0.313
2-4	0.199	0.072	0.125	0.188
1-4[s]	0.175	0.057	0.105	0.109
2-3[r]	0.287	0.112	0.188	0.344
1-3	0.255	0.090	0.160	0.227
0-3[s]	0.220	0.070	0.131	0.113

Note: r indicates player receives serve in the next game, s indicates player serves in the next game.

Figure 11: Match-winning probabilities - Results from O'Malley (2008) [18].

Match-winning probability calculated by our program are exactly those of Liu (2001) and O'Malley (2008), for the same scenarios, thus we can be reasonably assured that our calculator works as expected. However, since there is only a small set of match scenarios to check results against, our confidence is limited and we are still cautious of our errors elsewhere. We find another means of obtaining match-winning probabilities to perform further comparisons in more match scenarios.

3.5.2 Building and Comparing Results against a Tennis Match Simulator

Our calculator returns match-winning probabilities according to a hierarchical Markov model but this is not the only method to do so. An alternative would be to simulate tennis matches under desired parameters and record the number of times each player wins over a total number of runs. The proportion of a player's wins compared to total simulation runs gives an estimation of their match-winning probability under those parameters. Since the simulation is probabilistic, outcomes of each run vary but the proportion of a player's wins is expected to converge after a large number of runs. Thus, we use the simulation to obtain

each player's winning proportions, as an estimation for match-winning probabilities, then compare these to that given by our match-winning probability calculator under the same parameters.

The simulator models the scoring structure of a tennis match and takes the same point-winning probability inputs for each player, p and q, as our calculator. Until either wins, the simulator loops with probability p of incrementing player 1's points and $1 - p$ of incrementing player 2's points, during player 1's service game; probability q of incrementing player 1's points and $1-q$ of incrementing player 2's points, during player 2's service game. The algorithm is given in algorithm 2.

Algorithm 2 *double* ← SIMULATEMATCH (double p, double q, int totalRuns)

```
1:  int run = 1
2:  double numMatchesP1Wons = 0
3:  while run ≤ totalRuns do
4:      while match not finished do
5:          if is player 1's service game then
6:              either increment player 1's point with probability p
7:              or  increment player 2's point with probability 1 − p
8:          else
9:              either increment player 2's point with probability q
10:             or increment player 1's point with probability 1 − q
11:         end if
12:     end while

13:     if player 1 won then
14:         numMatchesP1Wons++
15:     end if

16:     return (numMatchesP1Wons/totalRuns)
17: end while
```

Before using the simulation results to validate those from our calculator, we must first be confident that the simulation results are correct themselves. As before, we check the results to those presented in Lui (2001) and O'Malley (2008). Since these are consistent, together with the fact that the implementation is relatively straightforward, we assume that it is accurate. Even if the simulator produces incorrect results, it is unlikely that it would be incorrect in an identical manner to that of our calculator and so we would be prompted to investigate any mistake upon finding inconsistencies.

The probability calculator and the match simulator produce results consistent with each other. Given that we obtained the same match-winning probabilities, across multiple parameters, using independent methods as well as being consistent with results from previous studies, it suggests that our program is correct and so we can move on to the next part of the project and use our calculator with confidence.

An observation to note is the differences in run-times needed to produce results of the two methods. Our calculator parses Markov chain data structures and uses a number of matrix manipulations with matrices of large sizes, nevertheless calculations are done in sub-second time. In contrast, although a single run of the simulation is fast, we require many runs (usually between 10,000 and 100,000) before making an estimation of winning proportions and so on average it takes up to 15 seconds per set of parameters. This proves

the superiority of our calculator method, in terms of time efficiency. It also indicates its potential usefulness in inferring live scores which are continuously updated as well as demonstrating the reason why we could not have used the simulator as our means of obtaining match-winning probabilities, from the start.

3.6 Determining Point-Winning Probability Parameter to Use for a Match

So far, we have developed the method of calculating match-winning probability under the assumption of availability of the parameters. *Current score* is not problematic as if we follow a match from beginning to end, it will always be known. *Point-winning probabilities* on serve, however, is more difficult to determine and we now examine some means to do so.

3.6.1 By Analysis of Historical Player Statistics

Some websites record historical player statistics including results against opponents on various court surfaces as well as further details about those matches. We can analyse this data and combine it with our own judgement to estimate an appropriate point-winning probability for an upcoming match. This is a popular method used in past studies. The major drawback of this is the overhead in obtaining and analysing data of each players of the match we wish to infer scores. A second difficulty is the requirement of subjective judgement in interpreting the data. Furthermore, having to manually deduce and input these parameters is a sacrifice in the degree of automation of our program although it may be a necessary one.

3.6.2 By Assuming a Fixed Value of the Sum of the Two Point-Winning Probabilities

We mentioned in section 2.2 that Klaassen and Magnus (2001) found, through analysis of empirical data, that the sum of two player's service game point-winning probabilities was on average, 1.29 for men and 1.12 for women. To illustrate how this may be useful to us, consider a chart relating each player's point-winning to match-winning probabilities such as figure 10a. Any match-winning probability can arise due to an infinite pair of each player's point winning probabilities, for instance, in the case when match-winning probability is 0.5, i.e. when both players are equally skilled, the point-winning probabilities must be equal; for example $0.1 - 0.1$, $0.2 - 0.2$, $0.99 - 0.99$ etc. In fact, any pair of values on the diagonal line of figure 10a joining the point $0-0$ to $100-100$. Klaassen and Magnus' (2001) result introduces a new constraint that allows us to narrow the previous list of possibilities to a single pair. In our evenly-skilled example, the probabilities would be $0.645 - 0.645$ (as $0.645 + 0.645 = 1.29$) for a men's match and $0.56 - 0.56$ (as $0.56 + 0.56 = 1.12$) for a women's match.

This method is attractive as given the extra constraint and implied match-winning probability at the start of a match, the corresponding pair of point-winning probabilities can be calculated without further input. This method has been used in practice in various

papers including Easton and Uylangco (2010) which show the resulting modelled probabilities closely follow that implied by odds of a real match. However, we still approach its use with caution as it relies on the assumption of low distribution variance of the sums of point-winning probability pairs.

We conclude that our best option is to use a mixture of the two afore mentioned methods. Our program defaults to using the automated fixed-sum method to determine point-winning probabilities upon starting a match, however also allows for manual adjustment of these values should it be needed.

3.7 Comparing Expected and Implied Probabilities of a Real Match

We want match-winning probabilities that we calculate from our model to be as close to that implied by Betfair odds as possible, for the same set of parameters. To test whether this is the case, we record match-winning probabilities implied by Betfair odds along with the score at each point of a full match. The exact means to do so is explained later in 5.1.2. For each point of the match, we use our calculator to work out the expected match-winning probability and overlay this on a graph with the implied probabilities. Each player's point-winning probabilities are obtained by estimation using a mixture of historical statistics and trial such that it equals the pre-match match-winning probability. These point-winning probabilities values are held constant for calculations throughout the match. Note that the resulting graphs are similar to that in Easton and Uylangco (2010) in figure 1, shown earlier, but in greater detail.

3.7.1 Match 1: *Wozniacki vs. Pennetta - Qatar Ladies Open (2011)*

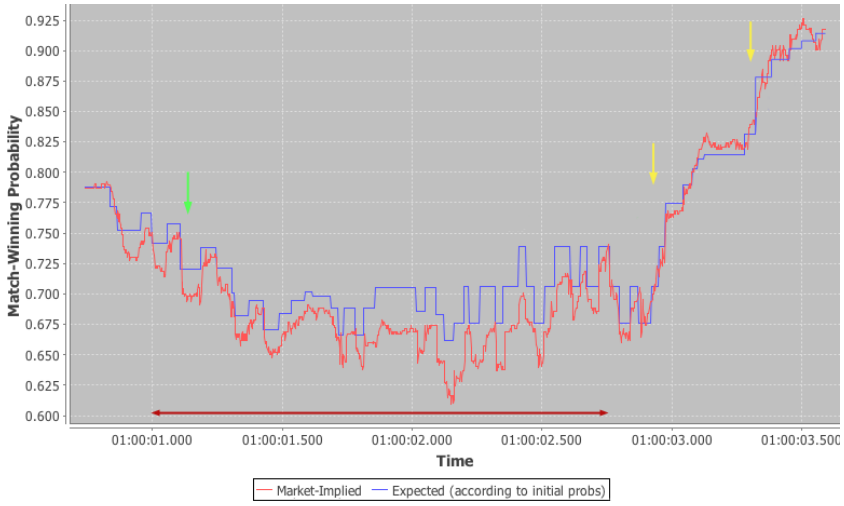


Figure 12: Probability of Wozniacki winning match for first set. **Green arrow** indicates Pennetta breaking Wozniacki's serve, **Yellow arrows** indicate Wozniacki breaking Pennetta's serve. **Red arrow** indicates period of systematic difference between implied and expected probabilities.

We see in figure 12 that the implied and expected match-winning probabilities follow the same overall pattern. Furthermore, it appears that the peaks and troughs corresponding

to individual point scorings also occur in the same manner, albeit with exception that the modelled probability changes at discrete scoring intervals whilst implied probability continues to fluctuate between points.

Wozniacki starts as the favourite with match-winning probability 0.79, point-winning probabilities used for expected match-winning probabilities are *Wozniacki - 0.59, Pennetta - 0.53*. Upon starting, Wozniacki immediately falls behind, the implied match-winning probability falls further than the expected although still keeping to the same patterns, developing a systematic difference between the two. The gap is closed again upon Wozniacki coming back to break Pennetta's serve after a long stand-off at *deuce*. The more than expected drop in implied probability suggests that the market over-reacted to Wozniacki's initial under-performance, only rallying after she regained control of the match. A departure of implied from expected probability, like in this case, could be troublesome for score inference but the systematic nature of the difference suggests that the problem could be overcome by developing a means of detecting such an occurrence. In fact, we encounter and tackle this very problem in section 5.1.2.

3.7.2 Match 2: *Del Potro vs. Soderling - Sony Ericsson Open (2011)*

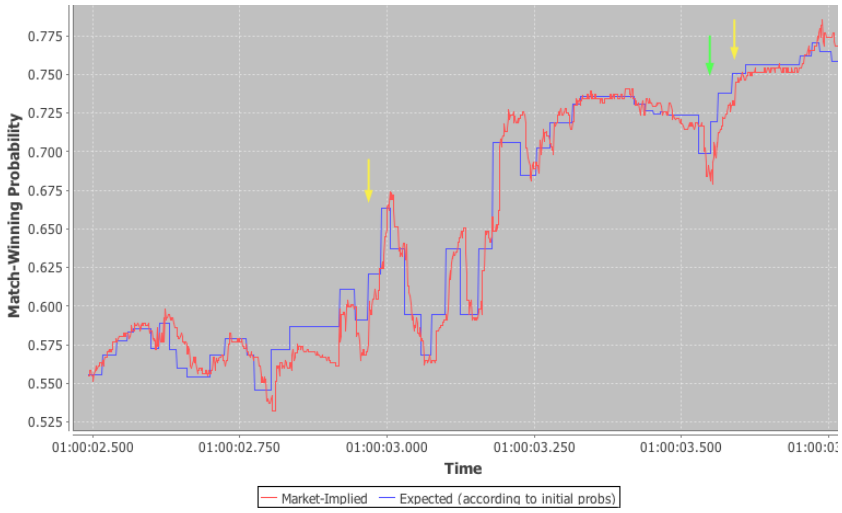


Figure 13: Probability of Del Potro winning match for first set. **Yellow arrows** indicate Soderling breaking Del Potro's serve. **Green arrows** indicates Del Potro breaking Soderling's serve.

Service game point-winning probabilities used as parameters for expected match-winning probability calculations are *Del Potro - 0.6305, Soderling - 0.6195*. Figure 13 shows implied followed expected probability closely, even more so than the previous example. This is encouraging as it shows our hierarchical Markov model appears to be successful in mirroring market probabilities but it is still too early to say whether it does so sufficiently. We can only determine the degree of accuracy needed for score inference through experimentation.

4

Using Data From Betfair

It should be clear by now that we intend to use odds traded on the Betfair exchange to determine market-implied match-winning probabilities. However, we have not yet discussed exactly what odds data is available, which parts are most useful to us, how we obtain the information nor in what manner we will use them. Before we proceed to develop our match-winning probability calculator and full inference program, we addressed these issues in this section.

4.1 About the Betfair API

As part of its Developers Program, Betfair provides a Java API that allows users to access various information relating to its live markets, ranging from match schedules to available odds on an event. We use the Free Access version of the API that has certain limitations to the frequency of calls allowed to be made to its service. For instance, market prices can be polled at a maximum rate of 60 per minute. This is sufficient for our needs as points scored in tennis matches are not more frequent than once every 10 seconds or so, therefore no significant amount of information is lost by only viewing the market at such intervals.

4.2 Tennis Market Information

Betfair's tennis market is divided into sub-markets that can be traded on including match-winner, set-winner, game score, number of aces, etc. Since we are interested in match-winning probabilities, the match-winner market is the most relevant. Within the match-winner market, the following information is available:

- For each of the two players:
  - Offered back prices
  - Offered lay prices
  - Last price matched
  - Volume available at back prices
  - Volume available at lay prices
- For the market as a whole:
  - Timestamp

- Total volume matched
- Market status

Note that we use the words *price* and *odds* interchangeably. Note also that there are separate prices for each player although the relationship between them is simple. Since there are only two players, the probability of one winning is that of the other losing so backing one is equivalent to laying the other. In practice, there is sometimes a small discrepancy between the two values, due to lack of perfect market efficiency. We examine the significance of this to our program shortly, in section 4.3.4.

4.3 Deducing Match-Winning Probability From Market Odds

Betfair presents all odds in decimal format as is favoured by European bookmakers, see appendix B for interpretation guidelines. We can easily convert any given decimal odds to a probability using the method described in appendix B.3. However, since there is a range of market odds data (including back, lay prices and last price matched - for each player), it is not obvious which should be used or how they should be processed in order to obtain a value that implies the match-winning probability. We consider possible choices along with their potential pros and cons for the purpose of score inference.

4.3.1 Choice 1: Average of Best Back and Best Lay Prices

Back and lay odds are the prices currently available to be taken in the market. The difference between the back and lay prices is the spread, which we consider in more detail in section 4.3.5, but for now it can be understood to exist as compensation to traders for taking a risk in the market. It is also the amount which you are certain to lose if you attempted to take opposite positions in the market simultaneously. i.e. both backing and laying the same player at the same time.

At any time, there is a range of back and lay prices offered in the market by different bettors at prices that they are willing to accept. It only makes sense for others in the market to take the best back or best lay price, that is, the price that would give the maximum return compared to the others. If we wanted to use the best back and best lay prices as indicators for match-winning probabilities, it makes sense to take the average and use this mid-point as our value.

The main advantage of this is that the back and lay prices are very responsive to changes in events, such as point scoring. As market participants react to changes, the back and lay prices immediately reflect what prices they are willing to trade at. However, the best back and best lay prices tend to fluctuate much more in comparison to the last priced matched, which we will consider shortly. In practical terms, this means that it is more likely for false-positives to occur in our inference program, that is, erroneously inferring points are scored when odds change is large when the actual cause was fluctuations between points. This is indeed the case when experimenting with the use of back-lay average values in our inference program where many false-positive inferences are made. Figure 14 compares back-lay average to last price matched. **Note that for this graph and all others in the report, we are observing player 1's data.**

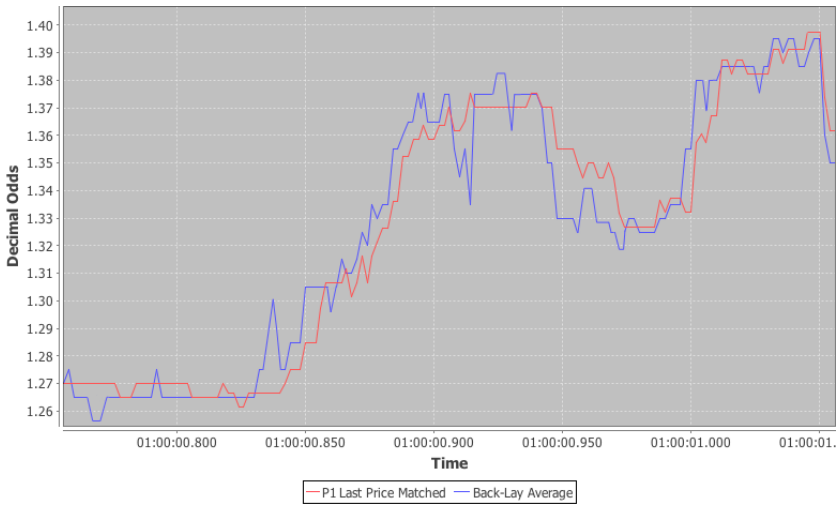


Figure 14: Back-lay average reflects changes in market a step before last price matched, however it also fluctuates more. *(Del Potro vs. Soderling - Sony Ericsson Open 2011)*

4.3.2 Choice 2: Moving Average of Back-Lay Average

In order to smooth out fluctuations, we could continuously calculate an average of a subset of most recent back-lay average values. The idea is that by giving weighting to older values, our program will no longer be as affected by sudden large odds price changes whilst still being able to observe overall movement pattern. The larger the subset used, the smoother our series of values.

It turns out that using a moving average in this way does not work well in practice. The smoothing effect occurs to such an extent that we now have the opposite problem, odds movements now appear to be too subtle. Even with a two point moving average, instances of actual point scoring are often missed. An alternative, the exponential moving average gives greater weighting to the most recent value with each older data point decreaseing in weighting exponentially. This is a slight improvement over a normal moving average, however the problem of over smoothing remains. We conclude that this method is not appropriate for our needs.

4.3.3 Choice 3: Last Price Matched

The last price matched is, as the name suggests, the most recent odds that was traded on the exchange. This is, arguably, a more accurate match-odds indicator than the back-lay average as it shows what was accepted not what is offered. The main drawback of using last price matched is that it is always a step behind the back and lay prices, since back and lay prices are offered first before they can be matched. A related problem is that when market liquidity is low, people may be less willing to take new bets and so the last price matched will be stagnant. These disadvantages are a small cost in avoiding the more serious problem of false-positive score inferences, therefore we will use last price matched as the odds that implies match-winning probability. This choice is supported by experimental results with the program as will be discussed later on.

4.3.4 Using Data from Both Players

We noted in section 4.2 that Betfair has separate odds for the players of a tennis market, each having their own back, lay and last matched prices. The relationship between the two player's odds should be such that if the implied match-winning probability of one player is p then that of the other player is $1 - p$. In theory, this relationship is ensured by traders immediately taking any arbitrage opportunity, that is, the possibility of making risk-free profit by simultaneously backing one player and laying the other. However, due to lack of perfect market efficiency, in practice there is sometimes a small discrepancy between the match-winning probabilities implied by the two player's odds.

Using last price matched from both players and averaging the implied match-winning probabilities results in the odds movements appearing to be smoother, making our program less affected by the false-positive score inferences caused by sudden odds changes. The downside is that we observe less sharp movements even when they are genuinely caused by point scoring. It is our choice, however, that avoiding false-positives outweighs the disadvantage. Figure 15 compares average last price matched from both players to that of single player.

From here on, whenever we refer to implied match-winning probability, we mean the average of the probabilities given by the last price matched of both players.

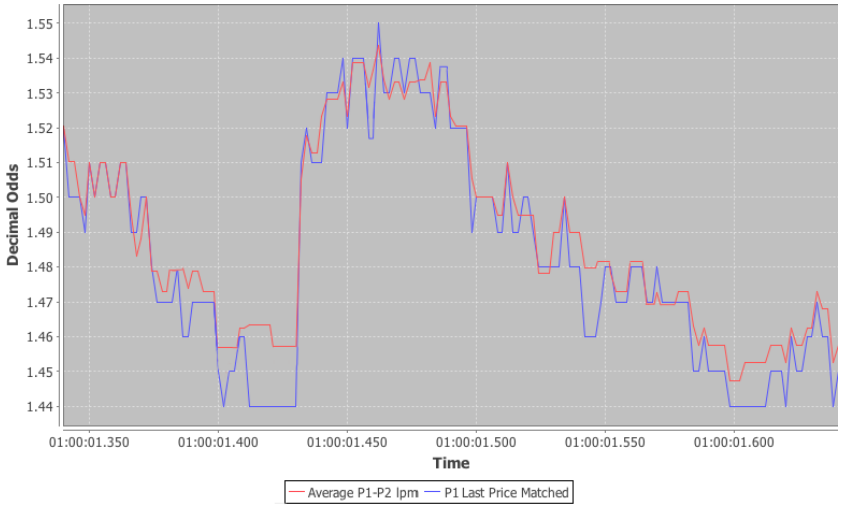


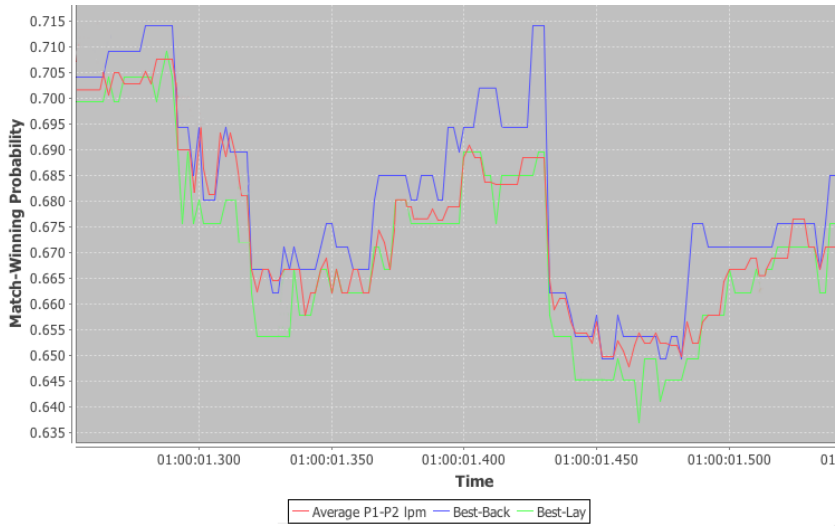
Figure 15: Average of last price matched from both players is smoother, fluctuates less, than from a single player. *(Del Potro vs. Soderling - Sony Erricsson Open 2011)*

4.3.5 Using the Spread as Indicator of Uncertainty

The difference between the best back and best lay prices reflects the level of risk traders incur in taking a position in the market. This is analogous to the spread between the bid and ask prices for stock traded on financial exchanges. One protects oneself from inability to trade out, taking the opposite position at a later time, by offering or accepting safer prices. Therefore the greater the perceived difficulty in doing so the wider the spread. It follows that we can use the relative size of the spread to infer the degree of uncertainty

in the market indicating the level of reliability of the current odds as a match-winning probability indicator.

(a) Smaller Back-Lay difference.



(b) Greater Back-Lay difference.

Figure 16: The spread, difference between best back and best lay prices, could be used to reflect market certainty about the outcome of a match. *(Nadal vs. Djokovic & Wawrinka vs. Federer - BNP Paribas Open 2011)*

4.4 Recording and Replay Matches

So far, we can access information on active markets through Betfair (i.e. matches currently taking place), however we cannot review earlier data. This is a problem when developing our inference analyser as we will need to compare various heuristics against constant sets of data in order to determine their successfulness, in other words, reviewing a single match multiple times. Therefore, it is essential to be able to record a match for play-back at a

later time. It would also increase efficiency if we were able to manipulate the replays such as fast-forward and skipping to specific parts of a match of interest.

To implement a replay feature, we record live market information to comma-separated value (csv) files [6]. We record an additional line in our csv file each time our program obtains market information from Betfair so that once the match is complete, we have an update by update record of the in-play market status. Market data is written in pairs of lines, one corresponding to each player, with a new pair written each time Betfair is polled. Figure 17 is a sample from such a file. To review the match, we parse the file, read data from each line in the same manner as if it were streamed live from Betfair. Additionally, it is also straightforward to fast forward, rewind and skip to particular parts of the match by selecting the line number or order of reading accordingly. For the writing and reading of the csv files, we use opencsv [7], an open source csv parsing library for java. A list of matches for which csv files were recorded throughout the course of the project is given in appendix D.

Timestamp	Inplay delay	Market status	Selection ID	Selection name	BP1	BV1	LP1	LV1	Total matched	LPM
163	0	ACTIVE	2249834	Juan-Martin Del Potro	1.82	758.49	1.84	533	153708.76	1.83
163	0	ACTIVE	2263597	Robin Soderling	2.18	285.53	2.2	86.53	50179.25	2.2
164	0	ACTIVE	2249834	Juan-Martin Del Potro	1.82	758.49	1.84	533	153708.76	1.83
164	0	ACTIVE	2263597	Robin Soderling	2.18	285.53	2.2	86.53	50179.25	2.2
165	0	ACTIVE	2249834	Juan-Martin Del Potro	1.82	758.49	1.84	533	153708.76	1.83
165	0	ACTIVE	2263597	Robin Soderling	2.18	285.53	2.2	86.53	50179.25	2.2
166	0	ACTIVE	2249834	Juan-Martin Del Potro	1.82	758.49	1.84	533	153708.76	1.83
166	0	ACTIVE	2263597	Robin Soderling	2.18	285.53	2.2	86.53	50179.25	2.2
167	0	ACTIVE	2249834	Juan-Martin Del Potro	1.82	758.49	1.84	533	153708.76	1.83
167	0	ACTIVE	2263597	Robin Soderling	2.18	285.53	2.2	86.53	50179.25	2.2

Figure 17: Example of recorded csv file. *(Del Potro vs. Soderling - Sony Ericsson Open 2011)*

4.5 Visualising Market Data

The data that we receive and record are sets of numbers which are fed into our program. We can plot information such as prices or volumes, obtained both live from Betfair or read from a csv file, against a timestamp to see a history of the market data. Visualising price and other data movement patterns helps in understanding market behaviour and so is crucial when developing our program. We use JFreeChart [8], an open source graphing library to plot such graphs. In fact, all graph figures in this article are examples of its use.

5

Inferring Score From Live Odds Feed

Now that we have a tennis probability calculator as well as the ability to pull market data from Betfair, we integrate them to develop the inference program, the second of the two core parts of the project. We have already discussed the basic idea of inferring score from odds: input each player's point-winning and match-winning probabilities to the probability calculator in order to work out the current score.

It should be clear that we rely on the market behaving in a rational way. That is to say, as a trivial example, if a player performs well then the odds for backing them should shorten, diminishing the profit and so implying greater probability of winning the match, and vice versa. Moreover, it would make inference easier if the changes in odds correspond exactly to what is mathematically implied. Although we know that market prices tend to follow what is expected, as shown by Easton and Uylangco (2010) in figure 1, it would be naive to assume that they follow it exactly. Betting is a human activity and so inevitably will be affected by psychological biases and a degree of randomness. We hope that, given a large number of market participants and sufficient market liquidity, the unexpected effects will be insignificant in the majority of cases. When unexpected market behaviour occurs, we need to find ways of dealing with them, depending on how troublesome they are.

We approach the development of the inference analyser in an iterative manner. We propose a range of strategies for inference, implement it then test their effectiveness on match recordings. Since we do not know how much and in what way market behaviour differs from that which we expect, it is difficult to foresee what is required for the entire process. Instead, we adapt and develop new algorithms to overcome flaws exposed in each iteration. In order to test our program after each implementation cycle to determine whether improvements were made, we first define a set of criteria against which we measure successfulness as well as methods to do so.

5.1 Criteria and Methods of Testing

5.1.1 Measurements of Correctness

Our goal is for our program to be able to infer scores from the start to finish of a tennis match and so one measurement of successfulness could be how many points it infers correctly before making a mistake. This is intuitively reasonable as the more points for which the analyser can infer correctly, the closer it is to the ideal goal. We use this as one quantitative measurement although it is not the only one. Longer duration of correct inference is an indication of successfulness, however it may not give the whole picture of

what is going on. For example, it could be that in two versions of the program, where one is a modification of the other, the second makes a mistake sooner but apart from this particular mistake it would have inferred the rest of the match more correctly than the first version. In such cases, it is not trivial to say whether the modification had improved the program and so judgement, upon analysis of results, is needed. It is also important to perform testing on multiple sets of data as methods which seem to work well on one set may be particular to that set of data and ineffective on others.

5.1.2 Manual Testing

To recognise when the program makes an inference mistake, we must know what the actual score of the match is at any time. One way we could do this is to also record scores of matches for which we are recording Betfair data. During testing against replays, we could then compare inferred scores to that which was recorded. Furthermore, recording the times of scoring along with graphs of Betfair data would allow us to easily identify and analyse price movement patterns triggered by point scoring.

There are a number of websites that claim to have live-updating of tennis match scores. In reality, the *live* scores of the vast majority of these have a substantial lag or inconsistent rate of update. For instance, it is often the case that the websites indicate scoring up to half a minute after the Betfair odds movements have suggested scoring had occurred and it is not uncommon either for updates not to occur for many minutes then all at once. Of all the websites, the most reliable for scoring updates has a consistent lag of around 15 seconds. Paired with graphs of live Betfair data, it is not too difficult to determine exactly when points are scored. Thus, for a means to compare inferred scores to actual scores, we first perform a screen recording of our live data graph along with the website's live scores whilst a match is taking place, as in figure 18, then play it back at a later time while also running our analyser. We remark that in fact, one of the potential applications of this project could be to completely or partially replace the dependence on other less reliable methods of obtaining scores such as those websites that we use here in testing our program.

The method of confirming correctness of the inference analyser described above is entirely manual in that it relies on comparing actual score, shown on the recording, to inferred score, from the program running on replay data, by eye on a point by point basis. This may appear somewhat crude but it is definitely necessary in understanding the workings of the analyser. For instance, viewing price movements on the graph as the inference program run with replay data can give important insight into how and why some methods are successful and others not. We will see examples shortly.

5.1.3 Automated Testing Framework

Although manually following the program whilst each score is inferred is essential, it is still useful to have a means of efficiently testing the program in an automated way since following an entire match is very time consuming. Sometimes we merely wish to know whether a mistake is made at all and if so, at what point in the match does it occur so that we can focus our attention there, thus improving efficiency in debugging. For this purpose, we implement a testing framework for our program. As well as for comparing various inference strategy implementations, we also use the framework to rapidly run the program

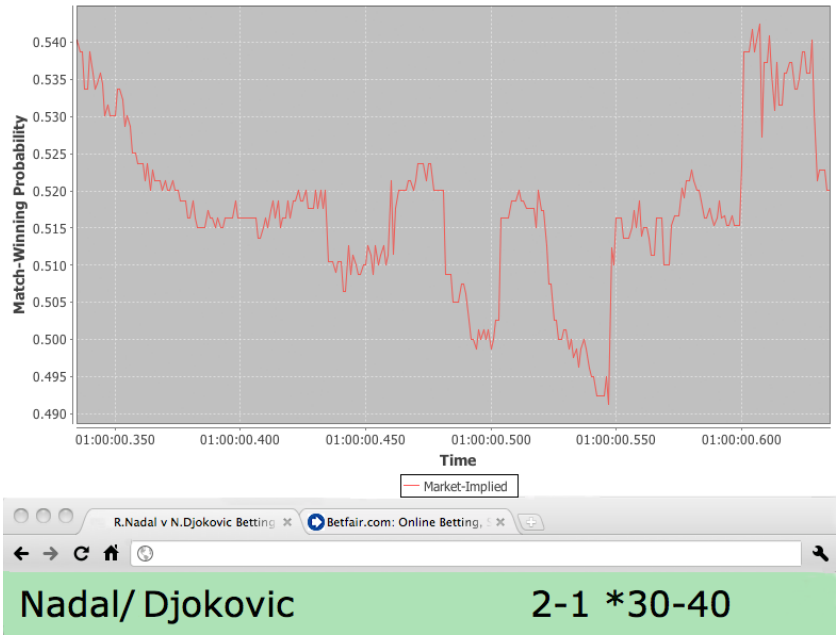


Figure 18: Performing screen recording of match-winning probabilities implied by Betfair odds together with live scores. *(Nadal vs. Djokovic - Sony Ericsson Open 2011)*

with multiple parameters and compare results. This is particularly useful in relation to testing a range of player point-winning probabilities, discussed shortly in section 5.1.4.

We modify the recorded csv file 4.4 to include a scoring column with entries denoting when a player scores. This can be done either through the GUI at the time of replay recording or by editing the csv file, post match, against recorded scoring data.

We build the framework to run the inference analyser at maximum speed checking times of inferred scoring against that of the actual score, written earlier to the csv file. If, at a particular time, one source indicates that scoring occurs and the other does not then we know that an erroneous inference was made. In such a case, we either ask the test to terminate, to allow us to investigate the cause, or to automatically correct the erroneous inferred score to actual score and continue testing. In the end, the test reports the number of times and at which points a correction was needed. Note that we allow a predefined discrepancy between the time of inferred scoring and actual scoring as the timing is not expected to be exact.

By testing the program in such a way, we reduce the time it takes to find points of failure of errors of the program, with respect to one set of match data, from around 3 hours (if we were to follow a whole match in real-time) to around one minute. Note that this is at the cost of the additional overhead of inputting the actual scoring to the csv file before we can use the framework. However, this is still beneficial especially when we may require to test the program multiple times for many reasons including that described below.

5.1.4 Testing with Multiple Point-Winning Parameters

Since there is no sure or straightforward way of obtaining values for player point-winning probabilities, it would be difficult for us to be confident in their reliability, no matter how they are determined. Consequently, if the program makes an inference mistake, we cannot differentiate whether it was truly due to the inference strategies being used or due to incorrect estimation of these input parameters. The only way to be sure would be to re-run the program under different point-winning probability parameters.

If the current implied match-winning probability is 0.5, then this may correspond to an infinite number of player 1 and player 2 point-winning probabilities e.g. 0.2, 0.3, 0.5 etc. In fact, any value as long as both players have the same. One such method we suggested in section 3.6 for solving this problem is to introduce an additional constraint such as a fixed sum of the two probabilities but this itself is difficult. For testing purposes, we want to run the inference simulation under a range of point-winning parameters, near to that which we estimated. To do this, we repeat the test varying the value of the fixed sum of the two point-winning probabilities, in intervals of a predefined step, after each run.

5.2 Heuristic 1: Calculating Thresholds and Detecting Crossings

A single probability corresponds to many possible scorings in a match as multiple situations have the same likelihood of occurring. However, given the current score, there are only two possible new scorings that could immediately arise, specifically, either player 1 gains a point or player 2 gains a point. Our inference analyser should rely on detection of when a player scores given that it already knows what the current score is. Thus, if we run the program from the start of the match then it should be able to detect point scoring and keep track of the score as the match progresses. The question is now how to recognise point scoring events occurring by viewing the market data. Assume that at any given point in the match we know the following:

- Each player's point-winning probability
- Current score of the match
- Match-winning probability (Implied by latest match-odds as discussed in 4.3)

We use our probability calculator, discussed in section 3.3, to compute the match-winning probability if player 1 were to score from the current situation and the match-winning probability if player 2 were to score from the current situation. Assuming that the point-winning probability parameters are accurate, we now have the two possible values that the implied match-winning probabilities could change to depending on which player actually scores. It is intuitive that these two probabilities will be either side of the current implied match-winning probability. We set these two values as thresholds for which we continuously test the latest implied match-winning probability against, until one is crossed in which case we infer that the corresponding player had scored. By updating the current score to what was just inferred and using the latest implied match-winning probability, we calculate a new pair of threshold values and repeat the process. This is shown in figure 19.

The pseudo-code for such an algorithm is given in Algorithm 3. The threshold is defined to be exactly the match-winning probability if a player was to win a point from the cur-

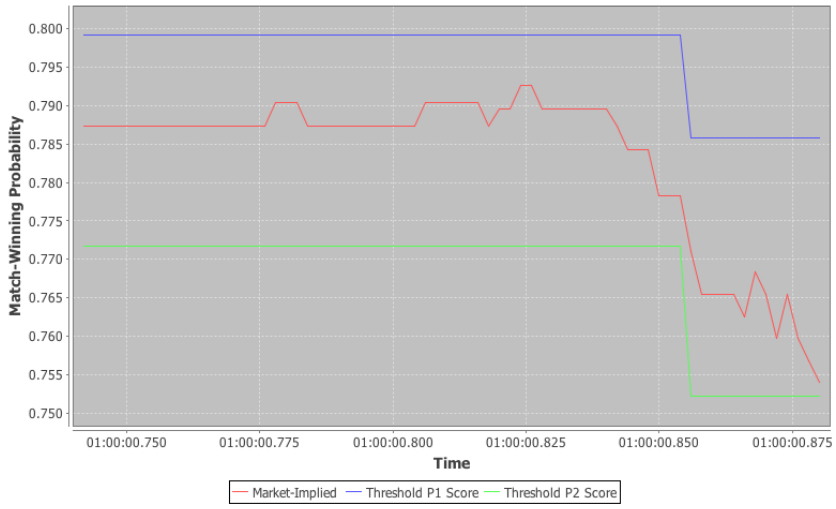


Figure 19: When implied match-winning probability exceeds a threshold, we infer that a point was scored. New thresholds are then calculated according to new score. *(Nadal vs. Djokovic - Sony Ericsson Open 2011)*

rent score. This could be changed to include an additional percentage error, adjusting the threshold for cases when the market is found to be under or over-reacting to point scorings. If the market is over responsive, that is if implied match-winning probabilities are changing more than expected after each point, we could widen the gap between the threshold values for a stricter requirement before inferring scoring. Conversely, if the market is less responsive, we could narrow the threshold gap to infer scoring under smaller change of implied probabilities.

5.2.1 Results of Use

This strategy does allow our program to detect initial instances of scoring, however success is severely limited. In fact, only the first few points are correctly inferred before a mistake appears. Even if we manually correct the inferred score, another mistake will be made at most two or three points later.

The problem seems to be that when a point is scored, odds often change more than expected beyond the threshold, meaning the expected match-winning probability could now be very different to the actual implied match-winning probability for the given point. As more points are scored, the difference between the expected and implied match-winning probabilities, grows larger. This results in the thresholds becoming increasingly incorrect with respect to the market implied match-winning probability. Consequently, the thresholds may not bound the implied match-winning probability as expected, thus becoming a useless means of scoring detection. In extreme cases, the implied match-winning probability may even be outside of a pair of newly calculated threshold causing an immediate erroneous inference after the previous one. An example is shown in figure 20a.

One could conclude that the more than expected price movements is down to irrational over-reactions of the market, however this may not be the case entirely. The fact that we judge market odds changes to be an over-reaction is relative to what we are expecting and

Algorithm 3 *void* ← INFERENCE ( )

```
1: p = probability player 1 scores on their serve
2: q = probability player 2 scores on their serve
3: s = current score

4: while matched not finished do
5:     m =latest implied match-winning probability for player 1
6:     s1 =score if player 1 wins point from s
7:     s2 =score if player 2 wins point from s
8:     //Threshold1
9:     m1 = implied match-winning probability for player 1 at s1 calculated using p and
       q as parameters
10:    //Threshold2
11:    m2 = implied match-winning probability for player 1 at s2 calculated using p and
       q as parameters

12:    if m ≥ m1 then
13:        //Inferred that player 1 had just scored
14:        s = s1
15:    else if m ≤ m2 then
16:        //Inferred that player 2 had just scored
17:        s = s2
18:    else
19:        //Inferred neither player had scored
20:    end if
21: end while
```

so it is reasonable to challenge the information upon which we are basing our expectations. In particular, our model assumes that the estimation of point-winning probabilities, which the market gives to each player, remains constant throughout the match. This may not be a valid assumption as with new information, in the form of observations of player performance, the market may alter its perception of a player's point-winning ability. With this in mind, we modify our strategy to include periodic re-calibrations of player point-winning probabilities.

5.3 Heuristic 2: Recalibrating Point-Winning Probabilities

We consider how to deduce the market's possible changing opinion of player point-winning probabilities. The difficulty here is almost the same as that of determining the point-winning probabilities at the start of the match 3.6, that we have two unknown variables (point-winning probabilities) to deduce from one equation (the relation linking the unknown variables to the known match-winning probability). The difference in this situation is that with an assumption, we can reduce the problem to solving for one variable at a time. Note that the two point-winning probabilities correspond to when each of the two players serves but also that at any point in a match, only one player is the server. From this, we assume that the market only changes its views on the point-winning probability of the player who is the current server. In practice, the two probabilities are not likely to be entirely independent but their correlation may be assumed insignificant enough for us to ignore.

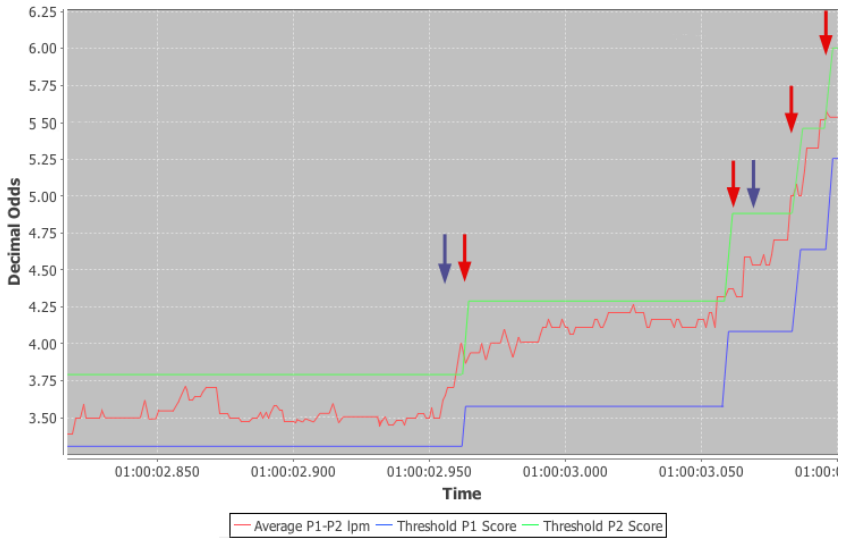
Let

- $p1Serve$ be player 1's point-winning probability on their serve
- $p2Serve$ be player 2's point-winning probability on their serve
- $mwpImplied$ be latest implied match-winning probability for player 1

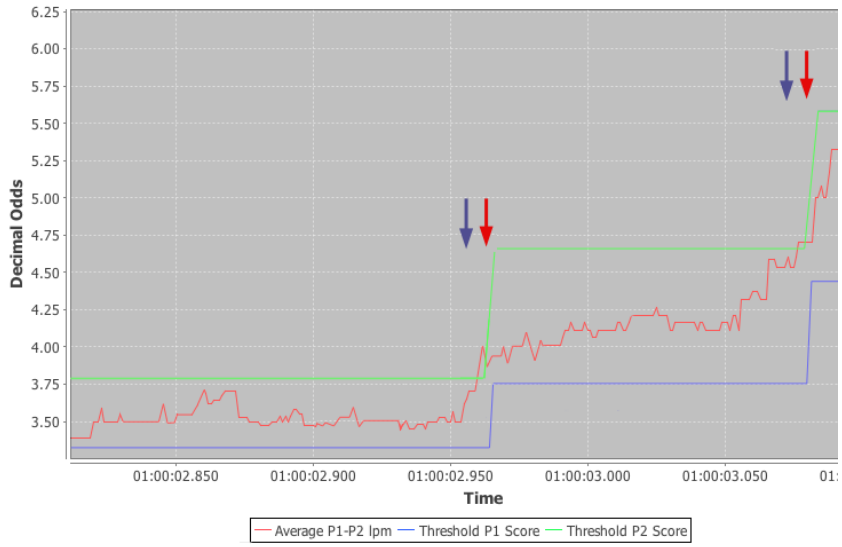
If player 1 is serving then we recalibrate by solving for $p1Serve$ with fixed $p2Serve$ and fixed $mwpImplied$. If player 2 is serving, we recalibrate by solving for $p2Serve$ with fixed $p1Serve$ and fixed $mwpImplied$. We test a range of possible point-winning probabilities by replacing one of $p1Serve$ or $p2Serve$ and calculating corresponding match-winning probabilities using our calculator from section 3. The algorithm loops in this way until a point-winning probability is found that matches the current implied match-winning probability. Note that we choose to use a loop as opposed to solving algebraically since the implementation is simpler and calculations are fast enough to not be of concern. The algorithm is given in Algorithm 4.

5.3.1 Results of Use

The result is that when points are scored, the next threshold values are now calculated using point-winning probabilities in line with the market's perception. This prevents the problem described in 5.2.1 and allows scoring detection by threshold crossing to work correctly far beyond the first few points. See figure 20 for demonstration of this scheme in use. Although errors are no longer caused by incorrect threshold values, sporadical erroneous inferences



(a) *Before* - Inferences become erroneous after short period.



(b) *After* - Correctly identifies scoring.

Figure 20: Identical situation before and after implementing point-winning probability recalibration. **Blue arrows** indicate time of actual point scoring, **red arrows** highlight time of inferred scoring. Erroneous inferences made in (*a*) as thresholds calculated using point-winning probabilities that become increasingly out of sync market views. Problem solved in (*b*) by calculating new thresholds according to recalibrated point-winning probabilities. *(Mayer vs. Berdych - Sony Ericson Open 2011)*

Algorithm 4 *void* ← RECALIBRATEPOINTWINNINGPROB (double *mwpImplied*, Score *currentScore*)

```
1: double pointWinningTemp = 0
2: double mwpTemp = 0
3: double errorOld = 1

4: if player 1 is serving then
5:     while abs(mwpImplied − mwpTemp)≤ errorOld do
6:         errorOld = abs(mwpImplied − mwpTemp)
7:         //Use Algorithm1 to calculate match-winning probability with parameters:
8:         mwpTemp = probOfWinning(pointWinningTemp, p2Serve, match-level,
           currentScore)
9:     end while
10:    p1Serve = pointWinningTemp

11: else
12:    while abs(mwpImplied − mwpTemp)≤ errorOld do
13:        errorOld =abs(mwpImplied − mwpTemp)
14:        //Use Algorithm1 to calculate match-winning probability with the parame-
           ters:
15:        mwpTemp = probOfWinning(p1serve, pointWinningTemp, match-level,
           currentScore)
16:    end while
17:    p2Serve = pointWinningTemp
18: end if
```

still occur for other reasons such as mistaking inter-point odds fluctuations for actual scoring.

5.4 Heuristic 3: Recognising Post-Scoring Odds Fluctuations

Large price fluctuations often occur in the time immediately following a player scoring a point, typically lasting a few seconds before settling down to the expected value. Examples of such occurrences are shown in figure 21. This phenomenon is most prevalent, although not limited to, crucial points of a match such as breakpoints, deuce and tiebreakers. An explanation of these price fluctuations could be initial uncertainty about where the odds should be - one can intuitively understand this as market forces find an equilibrium between backers' and layers' offers.

Whilst not unexpected, the extent to which prices fluctuate is often problematic for our program. It is difficult to distinguish such post-scoring fluctuations from price changes caused by actual scoring, figure 22a shows an example of a false-positive inference in such a situation.

We devise a method that allows our program to recognise this market behaviour. The reason we, as humans, can tell some price movements are due to post-point fluctuations is due to knowledge of very recent point scoring, making another point scoring within such a short time frame impossible, or at least very unlikely. It therefore seems natural to teach our program to use the same method, in particular, by implementing a timer. We

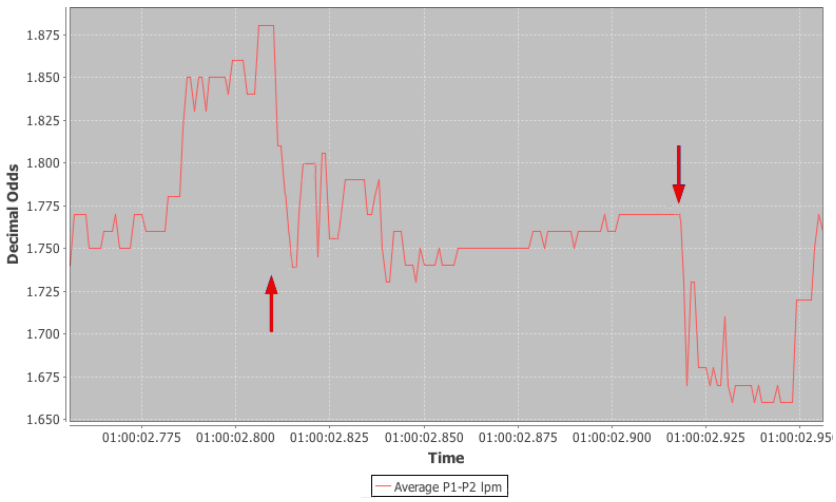


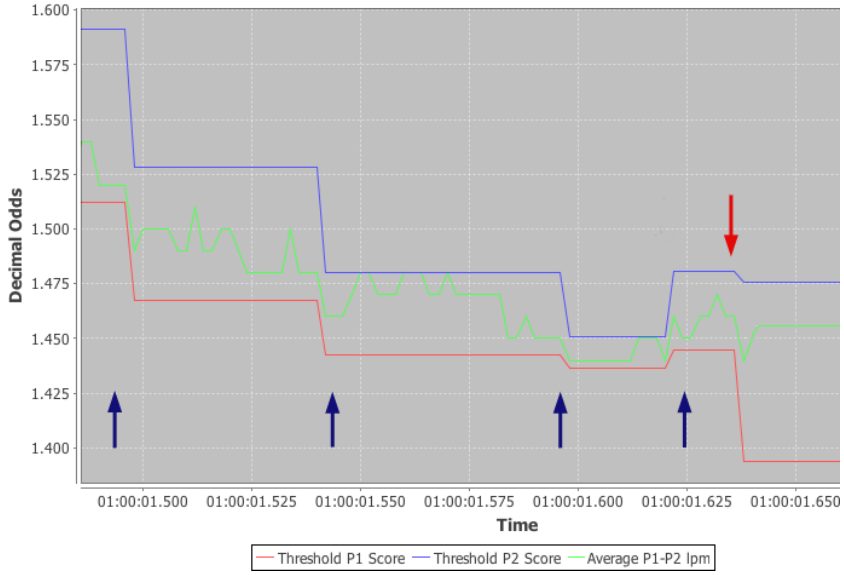
Figure 21: **Red arrows** highlight large odds changes caused by point scoring followed by short periods of smaller odds fluctuations. *(Nadal vs. Djokovic - Sony Ericsson Open 2011)*

start a countdown of our timer each time our program infers that a point was scored and any new crossings of thresholds before the timer reaches zero is attributed to post-scoring fluctuations and is ignored. Normal inference is resumed once the timer fully counts down.

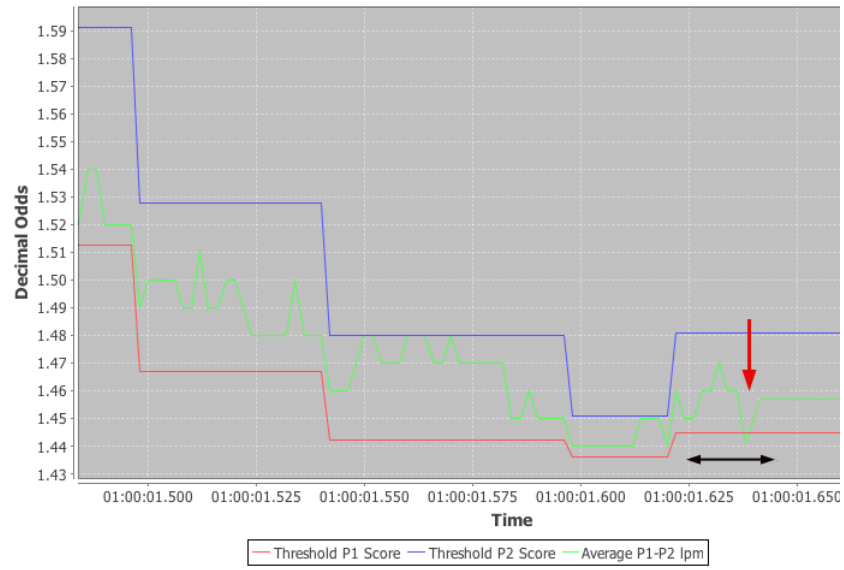
A difficulty in implementing this post-scoring timer method is the choice of countdown duration. A longer countdown ensures that normal inference is not resumed before the post-scoring fluctuation period ends but also increases the chance of missing the next genuine point scoring should it happen within this time. We must find an optimal value that strikes the delicate balance between these two factors. Since there is no official rule in tennis regarding minimum time between each point, a suitable value must be decided through experimentation. We find that the average time between points varies depending on whether the server or the receiver is the next scorer. In particular, an *ace* allows the server to rapidly score the next point whereas it would take the receiver a few extra seconds to do the same. In accordance with this, we determine that a reasonable countdown value for the server is 7 seconds and receiver is 9 seconds.

5.4.1 Results of Use

The post-scoring timer prevents almost all instances of false-inference in the situation described above. Furthermore, we find that even if the next point is scored before the timer runs down, our program can still detect the point correctly once countdown finishes since the threshold will already be crossed when normal inference resumes. This gives us a little leeway in our countdown timer value. However, if two points are scored during countdown then the program would not be able to infer them both.



(a) *Before* - False-positive inference made.



(b) *After* - Fluctuation correctly ignored.

Figure 22: Identical situation before and after implementing scoring timer. **Blue arrows** in (*a*) denote recalculation of thresholds following correct inference of scoring, arrows omitted in (*b*). **Red arrow** in (*a*) highlights an erroneous inference made after odds crossed threshold values but was not caused scoring. This is correctly ignored in (*b*). **Black arrow** denotes duration of countdown. *(Wozniacki vs. Pennetta - Qatar Ladies Open 2011)*

5.5 Heuristic 4: Averaging Odds During Fluctuation

An issue related to post-scoring fluctuations is calculation of thresholds according to incorrect values. After scoring, the program recalibrates point-winning probabilities according to the most recent match-winning probability. As we have seen previously, odds after scoring may be in fluctuation and so the value taken could be an inaccurate estimation of match-winning odds for that point. In turn, the thresholds calculated using these values will also be inaccurate and thus cause incorrect inferences of the next point as is the case in figure 23a. To solve this problem, we record all values of implied match-winning probability whilst prices are still in fluctuation during countdowns after point scoring. Once the timer reaches zero, we use an average of the recorded implied match-winning probabilities for recalibration of point-winning probabilities, which in turn are used for threshold calculation for the next point. This avoids using a single value, which may be inaccurate, for threshold calculations. Examples of use of this method are shown in figure 23b.

5.5.1 Results of Use

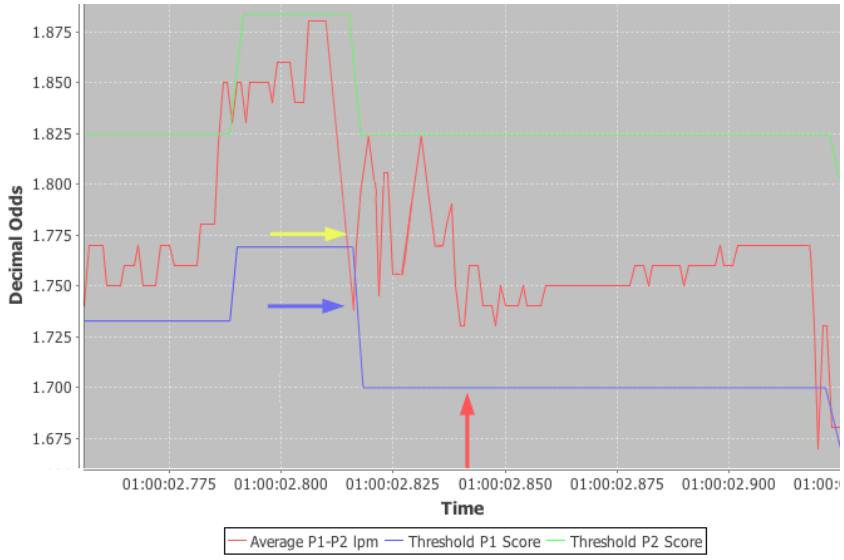
As with the case with many heuristics that we try, the above averaging method solves one problem but causes another. We mentioned in section 5.4.1 that our program may still correctly infer points scored during countdown since if a point was scored, a threshold should already be crossed when normal inference resumes. With the new recalibrating according to an average, this is no longer the case. Changes in implied match-winning probability during countdown are included in calculating an average and so new threshold values would change accordingly. This shows that recalibration according to an average should not always be used. To decide when it should be done, we maintain detection of possible scoring even when the timer is counting down and avoid using this averaging technique should a possible scoring be detected.

5.6 Heuristic 5: Recognising Large Odds Change as Scoring

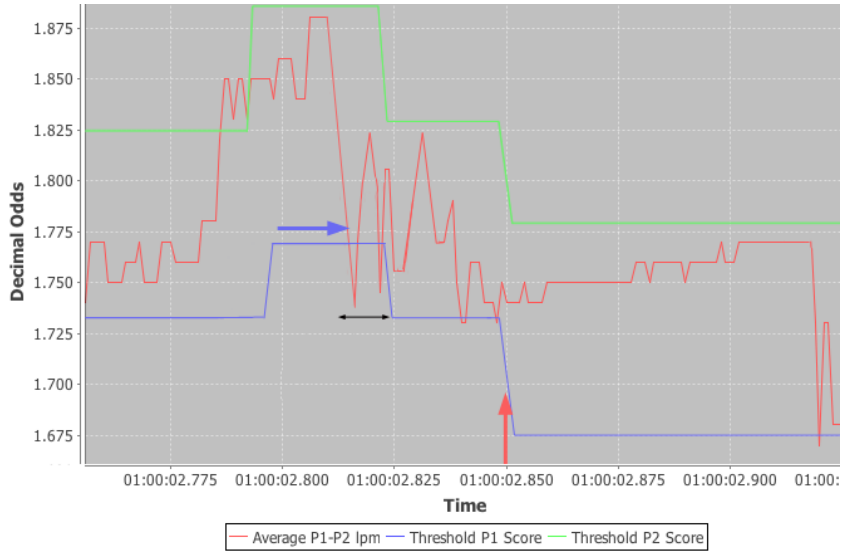
We have implemented heuristics to suppress errors in different scenarios, however the only indicator of scoring that we use is still by detection of threshold crossing. Although this works in most cases, we observe a situations in which it is unsuccessful, shown in figure 24a. We can see that a large, sharp and sustained change in odds occurs in response to point scoring yet the program fails to detect this as the odds is close but not reaching the threshold value. In response to this, one might suggest making the thresholds narrower, for example by allowing a degree of error between implied match-winning odds and the threshold, but we find this relaxation of criteria causes many false-positives elsewhere. An alternative solution that does not affect the current threshold crossing scheme is to implement a separate detection method that recognises large movements as point scoring.

5.6.1 Defining a *Large* Change

We explore how to define a *large* change in odds. The difference between threshold values varies depending on the current score therefore it is clear that *large* cannot be a constant value. Instead, we define it as a percentage of the difference between current threshold



(a) *Before* - Threshold calculated according to a value in fluctuation.



(b) *After* - Threshold calculated according to average value over fluctuation period.

Figure 23: Identical situations before and after implementing score-timer and countdown-averaging. **Red arrow** denotes time of scoring, undetected in (*a*) but detected in (*b*). **Yellow arrow** denotes average odds value over fluctuation period. **Blue arrows** denotes value of odds used in calculation of next threshold values. **Black arrow** highlights time over which average of odds is taken for subsequent threshold calculation. *(Nadal vs. Djokovic - Sony Ericsson Open 2011)*

we

values. If two consecutive implied match-winning probabilities differ by this large amount then we infer that a point was scored. This is illustrated in figure 24b. In order to avoid inferring random spikes as scoring, we add an additional constraint requiring that the odds to remain around the new value after a large change before we conclude that the cause was a point scoring. Large movements due to random spiking usually return to around the original value after the initial change and so will be ignored. See figure 25 for an example.

5.6.2 Detecting Large Change Over Many Points

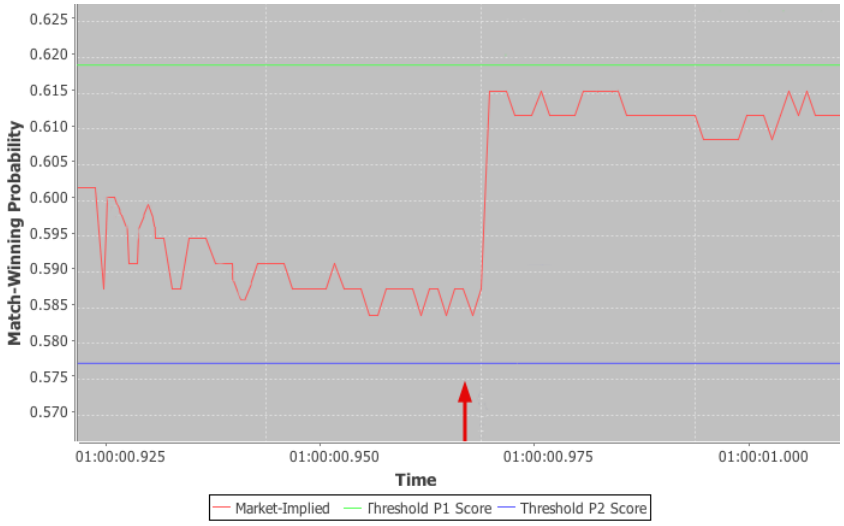
The above heuristic only considers the difference between two consecutive odds updates. A large change that is spread over three or more points may not be recognised as each consecutive change may be considered small. Figure 26 shows a case situation in which this is a weakness. We extend the previous scheme to also detect changes for any number of updates as long as the change is in one direction, i.e. monotonically increasing or decreasing. In this way, consistent changes of odds can be detected regardless of the rate it occurs.

We currently define a *large* change to be one which exceeds a percentage of the difference between threshold values. For instance, we can decide that if odds change in an amount over 50% of the threshold difference then a point has been scored as is the case in figure 24b. Although sometimes successful, the flaw in this idea is that the relative volatility in odds varies between points and so the appropriate definition of large should also change. Consider two situations in figure 27. For whatever reason, odds in situation figure 27 oscillate less between points than in figure 27b so we should accept a relatively smaller change in odds as being sufficiently large to be deemed to be caused by point scoring. This implies that we need a measurement of relative odds volatility between points.

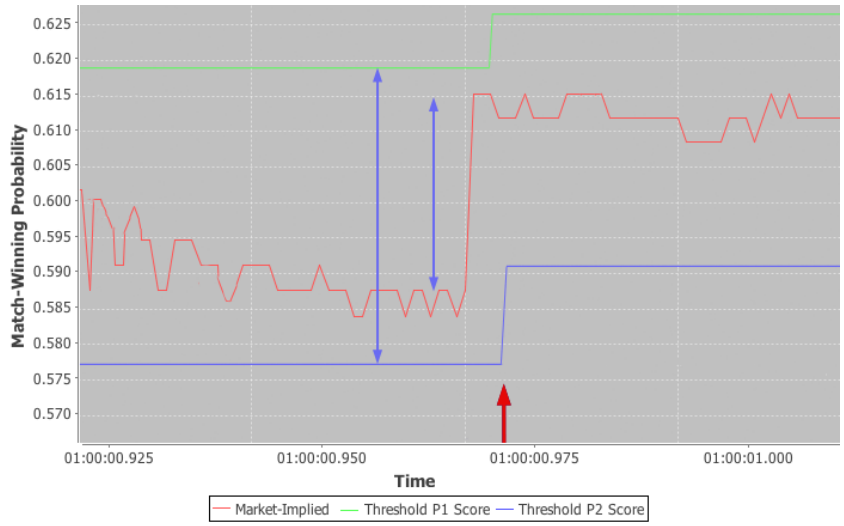
We record a list of odds values since the last point scoring then the larger the spread between the values, the greater the volatility. We use the interquartile range of the list of values, the difference between the third and first quartiles, as the measure of spread. The interquartile range is suitable over other statistical measures, such as range, mean or standard deviation as it is more robust, that is to say it is less affected by a small number of outliers. We can now dynamically specify the *large* percentage depending on the expected level of fluctuation. In periods of high volatility, we require a relatively large change in odds before we deem it to be caused by scoring and the opposite in periods of low volatility.

5.6.3 Results of Use

We find that recognition of large odds changes as scoring makes a huge contribution to our program by detecting almost all scorings that the threshold crossing method misses. In fact, we estimate that out of all inferences made 70% is detected by threshold crossing and 30% by large odds change recognition.



(a) *Before* - Fails to detect scoring as threshold not crossed.



(b) *After* - Correctly identifies scoring.

Figure 24: Identical situation before and after implementing large odds change detection. **Red arrows** indicate time of point scoring and subsequent odds change. **Blue arrows** in (*b*) denote the threshold difference and maximum odds change which are compared. In (*b*), the program correctly detecting scoring shown by the subsequent re-calculation of thresholds. *(Wozniacki vs. Pennetta - Qatar Ladies Open 2011)*

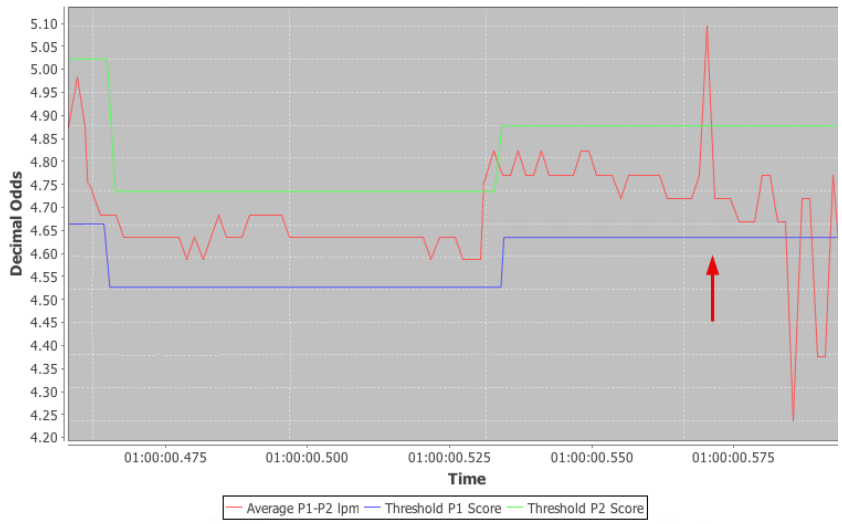
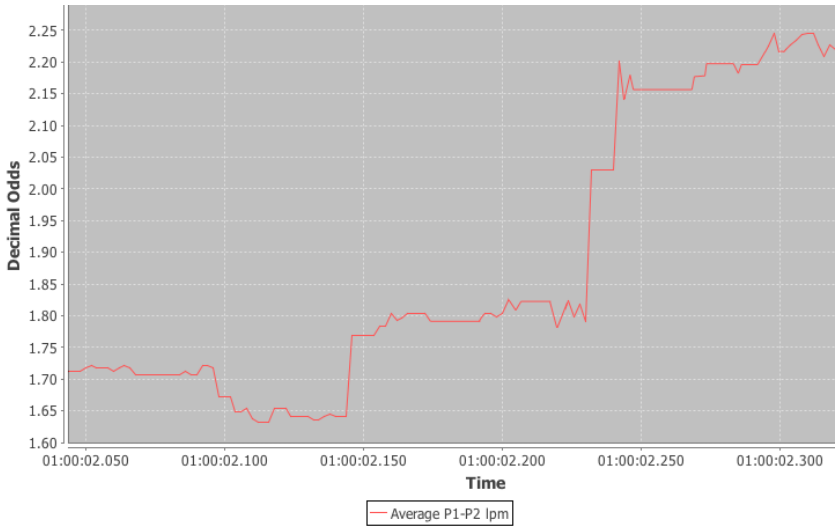
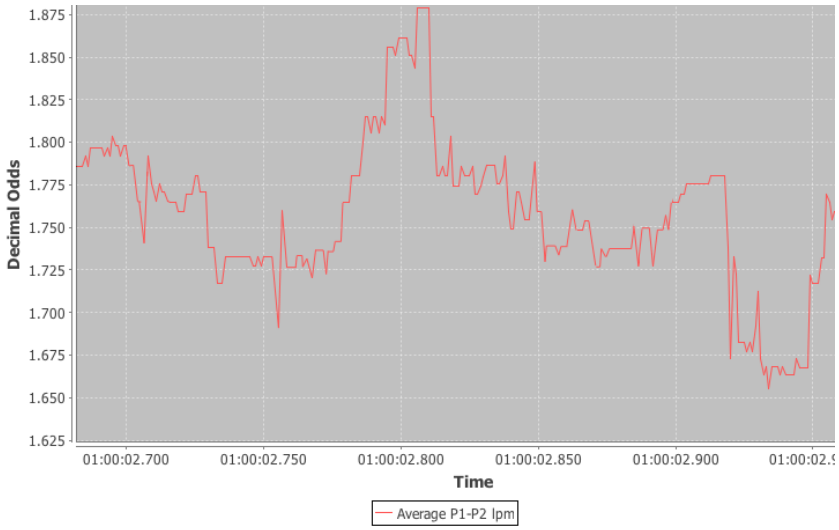


Figure 25: **Red arrow** highlights odds spike due to random fluctuation which is correctly ignored. *(Peng vs. Bartoli - Qatar Ladies Open 2011)*

Figure 26: **Red arrow** highlights large change in odds over a period and undetected by our large movement heuristic as consecutive changes are small. *(Ljubicic vs. Verdasco - French Open 2011)*



(a) Mild odds fluctuation between point scoring.



(b) High odds fluctuation between point scoring.

Figure 27: Matches, and sometimes different periods within a match, are characterised by varying amounts of odds volatility. *(Wozniacki vs. Pennetta - Qatar Ladies Open 2011 & Troicki vs. Djockovic - Sony Ericsson Open 2011)*

6

Case Studies

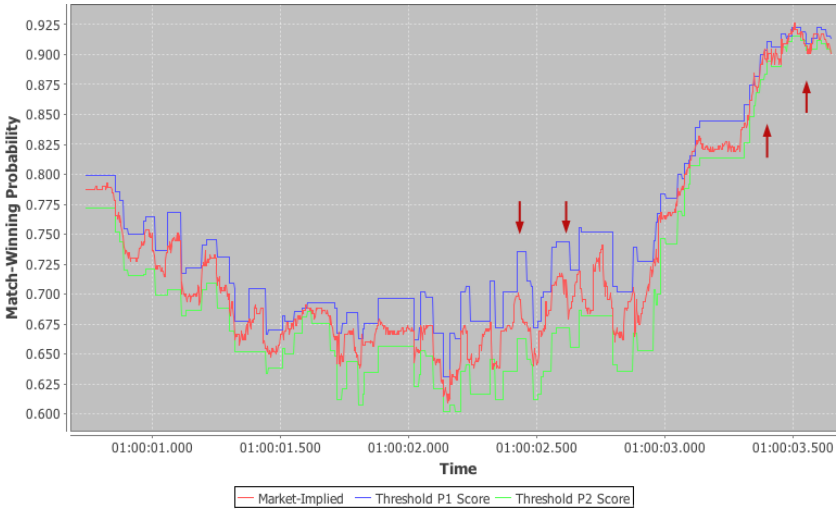
We now show the full score inference software in use although there is some difficulty in presenting our work as we wish to demonstrate a process rather than a set of data. We run the program against two real matches. If any mistakes in inference are made, we manually correct it and continue running. The matches are chosen as examples that give typical results, in comparison to the numerous matches tested, with regards to successfulness and limitations of our program. The matches chosen are the same as those analysed in section 3.7. We present graphs plotting implied-match winning probability with scoring probability thresholds, as described in section 5.2, so that the reader can see when scoring was inferred by noting times of threshold recalculation. Any errors made are also explicitly noted.

6.0.4 Match 1: *Wozniacki vs. Pennetta - Qatar Ladies Open (2011)*

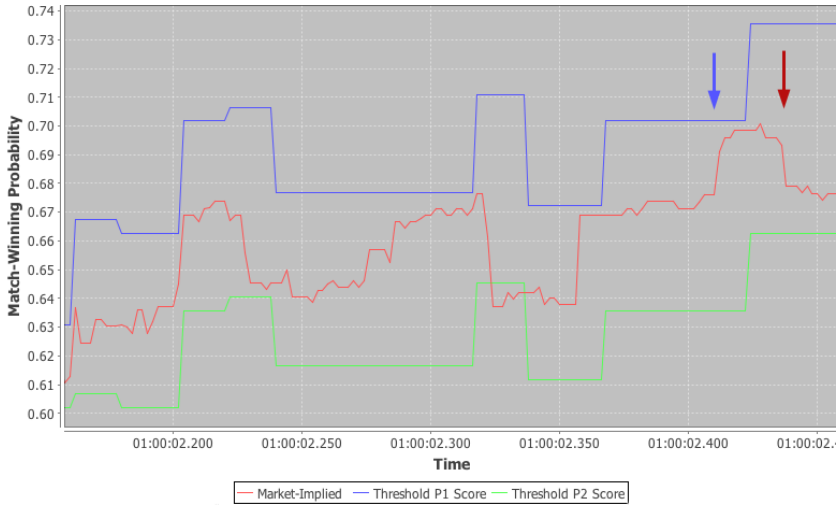
Caroline Wozniacki played Flavia Pennetta in the quarter finals of the Qatar Ladies Open 2011. Wozniacki was favourite to win with pre-match odds of 1.27, implying a winning probability just below 0.79. We run our software for the first set, as shown in figure 28a with the key highlights as follows:

- The program automatically determines service game point-winning probabilities using method described in section 3.6 with the option of manual adjustment that we do not use. Probabilities assigned are *Wozniacki* - 0.59*, Pennetta* - 0.53.
- Upon starting the match, Wozniacki immediately loses her first service game and her implied-match winning probability drops. Time of break shown in figure 12. For the next period, Wozniacki's under-performance drives her market-implied match-winning probability below that of her expected, according to the starting point-winning parameters.
- Recalibration of point-winning parameters after each point, as described in section 5.3, prevents many false inferences in this time. By this stage, probabilities assigned are *Wozniacki* - 0.576*, Pennetta* - 0.53. If not for recalibration, many false-positives inferences for Pennetta scoring would have occurred as the market over-estimates her good performance with regards to the starting point-winning probabilities.
- No errors made for the first 35 points played, until end of fourth set.
- First error is an undetected point scoring due to it occurring too quickly after the previous point, perhaps from an *ace*. The fluctuation timer, described in section 5.4, was counting down. This can be seen in more detail in figure 28b.

- Wozniacki regains control and dominates the second half of the set breaking Pennetta twice.
- Inference errors occur towards the end of the set when Wozniacki's match-winning probability is very high, around 0.9, due to the difference between each point only affecting match-winning probability slightly.
- Wozniacki wins first set $5-2$ after 75 points.



(a) First set of match.



(b) Close-up of the first error in part (*a*) - scoring occurs too close together to be detected.

Figure 28: Market-Implied probability of Wozniacki winning match, with threshold crossings. **Red arrows** indicate points of erroneous inference. **Blue arrow** in part (*b*) indicates the last correct inference before the missed scoring. *(Wozniacki vs. Pennetta - Qatar Ladies Open 2011)*

In summary, our program inferred the scores of the first set of the match, making only four errors in over 75 points (8 games including a 10 point *deuce*). The longest consecutive

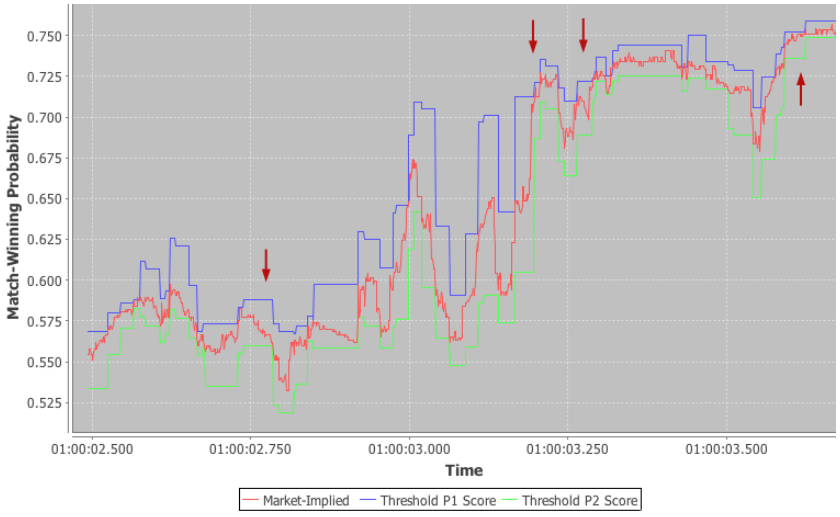
period without errors was 35 points. Each heuristic we developed in section 5 was used many times in the process. A weakness that was exposed is when one player denominates the match, their match-winning probability is very high such that any change in score only alters it slightly such that our program has difficulty distinguishing fine changes from market fluctuations between points.

6.0.5 Match 2: *Del Potro vs. Soderling - Sony Ericsson Open (2011)*

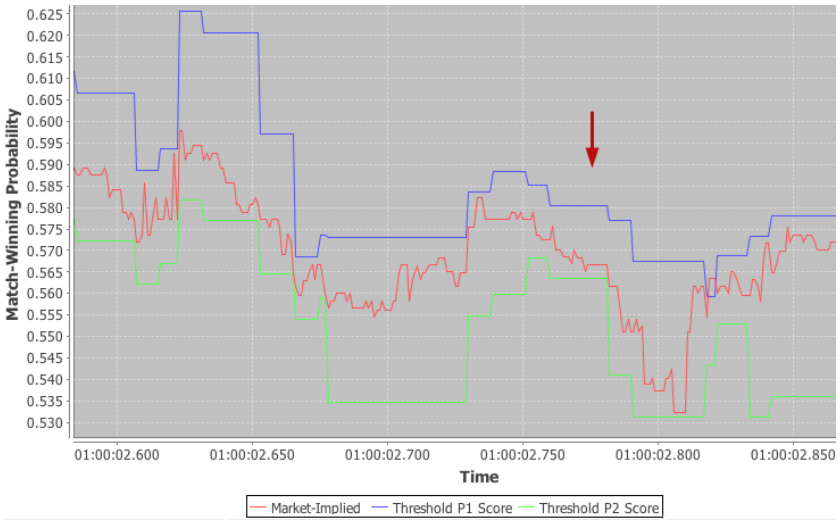
Juan Martin Del Potro played Robin Soderling in the second round of the Sony Ericsson Open 2011. Graph of implied match-winning probability for Del Potro and thresholds shown in figure 26.

- Pre-match odds 1.80 imply match-winning for Del Potro of 0.56. Automatically assigned point-winning probabilities *Del Potro - 0.6305, Soderling - 0.6195.*
- The first missed inference occurs after 10 points in the third game. This was caused by market odds not changing as much as expected, not crossing threshold nor triggering detection via large change in odds.
- The next period, including Soderling breaking Del Potro, was accurately inferred. Large change in odds were expected for each scoring and so are easily detected when they occur.
- The following period of lower expected change in odds are more difficult to distinguish from inter-point fluctuations and so cause some errors.
- Del Potro wins the match $5 - 2$ after 60 points.

Slightly more errors were made in a similar number of points. Again, all heuristics developed in section 5 were used. Errors occurred due to similar reasons to before; when expected odds between points are small, there is difficulty in detecting scoring and more chance of false-positive inference. Additionally, there was a greater amount of fluctuation between points than in the previous match that may have been caused by more traders in the market, greater volume traded and greater liquidity, compare figure 29b to 28b. This caused odds movement to seem more continuous as they blurred together rather than change in discrete steps, making it harder to distinguish separate point scoring. Greater inter-point fluctuation also requires large change in odds for scoring detection using method described in section 5.6 and so some were missed due to this reason.



(a) First set of match.



(b) Close-up of the first error in part (*a*) - discrete consecutive scorings unclear.

Figure 29: Market-implied probability of Del Potro winning match. **Red arrows** indicate points of erroneous inference. *(Del Potro vs. Soderling - Sony Ericsson Open 2011)*

7 Evaluation

7.1 Hierarchical Markov Model and Match-Winning Probability Calculator

In section 3, we derived a stochastic tennis model using a hierarchy of Markov chains defined by two point-winning probability parameters. This model enabled us to calculate match-winning probabilities, from any point in a match, which formed the foundation of scoring detection heuristics used later in 5.2.

In section 3.1.5, we discussed the idea of extending our model to include second services by introducing additional states but rejected this for reasons of complexity. During testing of our inference program, however, we did speculate that some false-positive inferences may have been caused by market reactions to first service faults rather than point scoring. It is possible that these errors may have been prevented if our model took second services into account, however we hypothesise that it is equally likely that this would have introduced inference errors in other cases as additional estimated parameters would be needed to define our model. Further research could be done to investigate this.

Although developed for use in our score inference program, we suggest that the match-winning probability calculator may be desirable as a standalone application or part of other systems. For instance, it could be used on future work in tennis modelling, integrated into other software or even as a tool in aiding traders to make decisions on the market.

7.2 Score Inference From Live Feeds

In section 5, we developed ideas for score inference. We first implemented the fundamental idea of calculating scoring thresholds then made refinements and additions to enhance our program. We discovered that the heuristics used with the aim of solving a particular problem generally did not work in all cases of the targeted behaviour and would sometimes even cause errors in other situation. This was often inevitable as we found that the market did not always react in an identical manner given the same situation nor across different matches. However, we found that inference errors were generally rare overall and that the heuristics worked well, as seen in section 6.

The score inference software can be enhanced by implementing additional heuristics in scoring detection, false-positive inference prevention as well as other ideas, forming basis for much future work. For instance, we suggest:

- Account for non-independence in probability of winning points, for example, adjusting for psychological advantage when ahead in a set.
- Use variance of each player's point-winning probability in threshold calculations.
- Develop measure of certainty for when scoring is inferred i.e. confidence that detection decision is correct.
- Use additional market information such as volume of bets and fully range of back and lay odds offered.

7.3 Conclusion

We set out to investigate the extent to which it is possible to infer the score of a tennis match from live betting odds. Our work has proven that this is not only possible but in fact can be done correctly such that an entire set can be inferred with very few errors. Despite this success, much more work is still needed before the software would be suitable for replacing traditional sources of scoring information or integration with automated trading applications. Overall, we believe our contribution is valuable to the research community in answering the above question, offering insight into this area and provides precedence for further research.

A

Rules of Tennis

A tennis-singles match is contested between two players. A game is finished when one player receives at least 4 points with a two point margin. The values of the first three points earned by a player are 15, 15, and 10, respectively. When a player wins the first point, the score is 15:0 (or 0:15). When the scores are a tie at 40−all or above, it is called a deuce. If the server scores or loses the point after deuce, it is advantage to the server or receiver, respectively. If the server wins the next point following 40:0, 40:15, 40:30, or advantage server then he takes the game.

A set is finished when one player wins at least 6 games with a two−game margin. If the score reaches 6−all, a tiebreaker game will be played to decide who takes the set. Thus, the possible winning scores in a set are: 6:0, 6:1, 6:2, 6:3, 6:4, 7:5, 7:6. Service of each game alternates between the players, the winner of the current set starts service of the next set.

A tiebreak is won when a player reaches 7 points and has a two point lead, it is possible for a tiebreak game to last indefinitely. The player who was first to serve in the set begins serving in a tiebreak, after the first point the service alternates every two points.

A match consists of three or five sets, played as the best of three or five. Women's matches are always best of three whereas men's differ depending on tournament rules.

The server of the first game is decided by a coin toss. Thereafter, each successive game is served by the other player. During a tiebreaker, the player who would normally serve the point serves once, then service alternates every two points.

For more detailed description of tennis rules and regulations, refer to the International Tennis Federation rule book [9].

B

Explanation of Betting Odds

B.1 Fractional Odds

Fractional odds quote the total that will be paid out to the bettor should he win, relative to his stake. For example, odds of 3/1 imply that if you bet £10 then you stand to win an additional £30, making the net profit £40. Conversely, odds of 1/3 mean you stand to win £10 on a £60 stake.

B.2 Decimal Odds

In contrast to fractional odds which are conventional in the UK, decimal odds are favoured in continental Europe. Decimal odds differ from fractional odds in that they quote the total amount that will be paid out to the bettor including the initial stake. Fractional odds of 3/1 would be $1 + 3/1 = 4$ and fractional odds of 1/3 would be $1 + 1/3 = 1.333$.

B.3 Conversion Between Decimal Odds and Winning Percentage

It is more convenient for us to work with percentages rather than odds. Algorithm 5 demonstrates how to convert from decimal odds to percentage.

Algorithm 5 *double* ← DECIMALODDSTOPERCENTAGE (double *odds*)

1: double $n = odds - 1$
2: int $dp = n$'s number of decimal places
3: double $numerator = n * 10^{dp}$
4: double $denominator = 10^{dp}$
5: double $newDenominator = numerator + denominator$
6: return $(1 - numerator/newDenominator) * 100$

Converting from probability to decimal odds is equally more simple, shown in Algorithm 6.

Algorithm 6 *double* ← PERCENTAGETODECIMALODDS (double *perc*)

1: double $numerator = 1 - (perc/100)$
2: double $denominator = perc$
3: return $numerator/newDenominator + 1$

C

Program Structure

Figure 30 shows a simplified UML diagram of the main components of our system. The two core components that perform probability calculations and score inference are highlighted in red, development and implementation of these parts will be explained in detail in sections 3 and section 5, respectively. Implied match-winning probability parameter can be taken from one of two different parts of our program; live Betfair market or recorded replay data, this is explained in section 4.4.

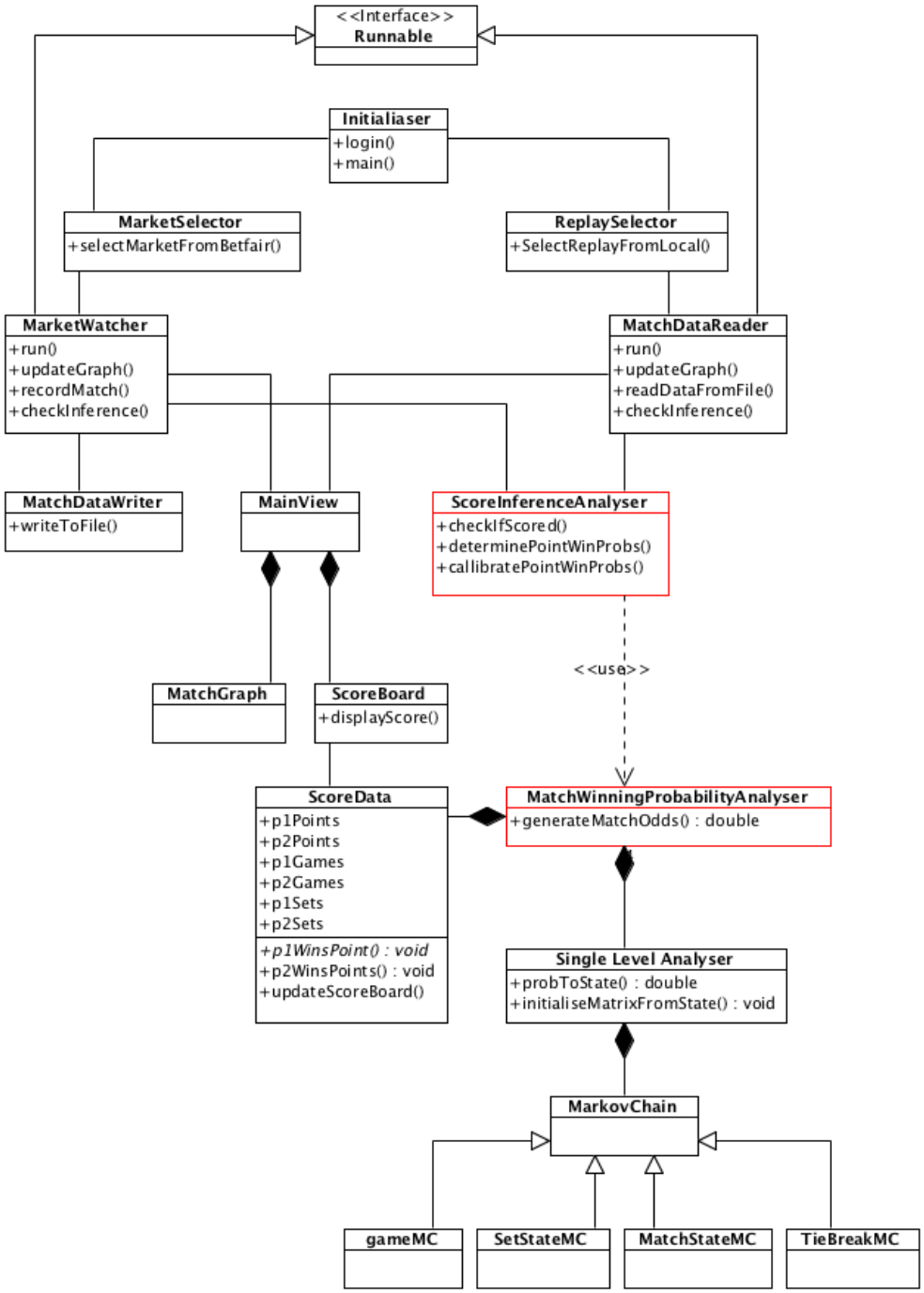


Figure 30: Simplified UML class diagram of software design.

D

Tennis Matches Recorded in csv Files

The following is a list of matches for which csv files were recorded, either in part or in entirety.

Men's

- Troicki vs. Djokovic - *Australian Open 2011* (Set 1)
- Federer vs. Djokovic - *Australian Open 2011* (Set 1)
- Nadal vs. Del Potro - *BNP Parisbas Open 2011* (Full)
- Nadal vs. Djokovic - *BNP Parisbas Open 2011* (Full)
- Isner vs. Anderson - *Sony Ericsson Open 2011* (Set 1)
- Wawrinka vs. Federer - *Sony Ericsson Open 2011* (Set 1)
- Mayer vs. Berdych - *Sony Ericsson Open 2011* (Set 1)
- Nadal vs. Djokovic - *Sony Ericsson Open 2011* (Full)
- Del Potro vs. Soderling - *Sony Ericsson Open 2011* (Full)
- Ferrer vs. Devvarman - *Sony Ericsson Open 2011* (Full)
- Ljubicic vs. Verdasco - *French Open 2011* (Set 1)
- Gasquet vs. Bellucci - *French Open 2011* (Full)
- Nadal vs. Soderling - *French Open 2011* (Set 1)
- Nadal vs. Federer - *French Open 2011* (Set 1)

Women's

- Wozniacki vs. Pennetta - *Qatar Ladies Open* (Set 1)
- Wozniacki vs. Sharapova - *Qatar Ladies Open* (Set 1)
- Wozniacki vs. Azarenka - *Qatar Ladies Open* (Set 1)
- Jankovic vs. Schiavone - *French Open 2011* (Full)
- Bartoli vs. Dulko - *French Open 2011* (Set 1)
- Jankovic vs. Schiavone - *French Open 2011* (Set 1)
- Bartoli vs. Schiavone - *French Open 2011* (Set 1)

References

[1] H2 Gambling Capital Consultants - Gambling Statistics and Reports (accessed 2011). http://www.h2gc.com/h2data_and_reports.php

[2] SportingIndex (accessed 2010). http://www.sportingindex.com/extra-spread/guide/betting-market-overview.htm

[3] Betfair (accessed 2010). www.Betfair.com.

[4] William Hill (accessed 2010). www.WilliamHill.com.

[5] Bet365 (accessed 2010). www.Bet365.com.

[6] Wikipedia - Comma-Separated Values Files (accessed 2011). http://en.wikipedia.org/wiki/Comma-separated_values

[7] Opencsv (accessed 2011) http://opencsv.sourceforge.net/

[8] JFreeChart (accessed 2011) http://www.jfree.org/jfreechart/

[9] International Tennis Federation, Tennis Rules (accessed 2011) http://www.itftennis.com/abouttheitf/rulesregs/rules.asp

[10] Magnus, J.R. & Klaassen, F.J.G.M., 2000. "How to reduce the service dominance in tennis? Empirical results from four years at Wimbledon", *Open Access publications from Tilburg University*, urn:nbn:nl:ui:12-84108, Tilburg University.

[11] Brown, Alasdair, 2011. "Evidence of In-Play Insider Trading on a U.K. Betting Exchange", *Applied Economics*, issn:0003-6846. http://www.informaworld.com/10.1080/00036846.2010.537644.

[12] Easton, Stephen & Uylangco, Katherine, 2010. "Forecasting outcomes in tennis matches using within-match betting markets," *International Journal of Forecasting*, Elsevier, Vol. 26(3), pages 564-575, July.

[13] Magnus, J.R. & Klaassen, F.J.G.M., 2001. "Are Points in Tennis Independent and Identically Distributed? Evidence From a Dynamic Binary Panel Data Model", *Journal of The American Statistical Association*, Vol.96, pages 500-509. http://www1.fee.uva.nl/pp/klaassen/index_files/iid.pdf.

[14] Yaping. Lui, 2001. "Random Walks in Tennis". http://www.math-cs.cmsu.edu/~mjms/2001.3/Yliuten.pdf.

[15] Magnus, J.R. & Klaassen, F.J.G.M., 2003. "On the Probability of Winning a Tennis Match", *Medicine and Science in Tennis*, Vol.8, pages 10-11. http://www1.fee.uva.nl/pp/klaassen/index_files/ArticleMST.pdf.

[16] T. Barnett, A. Brown, S. Clarke (2003). "Developing a Tennis Model That Reflects Outcomes of Tennis Matches". strategicgames.com.au/8mcs.pdf

[17] Paul K. Newton, Joseph B. Keller, 2005. "Probability of Winning at Tennis I. Theory and Data", *Studies in Applied Mathematics*, Vol.114 (3), pages 241-269, April.

[18] James. A. O'Malley, 2008. "Probability Formulas and Statistical Analysis in Tennis", *Journal of Quantitative Analysis in Sports*, Vol.4, Iss.2, Article 15. http://www.bepress.com/jqas/vol4/iss2/15.

[19] Paul K. Newton & Kamran Aslam, 2009. "Monte Carlo Tennis: A Stochastic Markov Chain Model", *Journal of Quantitative Analysis in Sports*, Berkeley Electronic Press, Vol. 5(3). http://www.bepress.com/jqas/vol5/iss3/7.

[20] F. Bause, 2002. Stochastic Petri Nets (2nd Edition). http://ls4-www.informatik.uni-dortmund.de/QM/MA/fb/publication_ps_files/bause_kritzinger_spn_book_screen.pdf.

[21] "Probability Models for Tennis Scoring Systems", *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, Vol 37 pages 63-75.

[22] Jackson, D., and K. Mosurski. 1997. "Heavy defeats in tennis: Psychological momentum or random effect?", *Chance*, Vol.10(2), page 27.