Statistics Honours Project

Calibrating an Adaptive Farmer-Joshi Agent-Based Model for Financial Markets

Ivan Jericevich and Murray McKechnie
Supervisor: A/Prof. Tim Gebbie

Department of Statistical Sciences,
University of Cape Town

October 21, 2019

Abstract

We consider the problem of calibrating the Farmer-Joshi agent based model of financial markets using a genetic algorithm and a Nelder-Mead with threshold accepting algorithm in order to replicate prior work. The Farmer-Joshi model [8] is an interesting model for understanding daily trading decisions made from closing auction to closing auction in equity markets, as it attempts to model financial market behaviour without the inclusion of agent adaptation. However, our attempt at calibrating the model has limited success in replicating important stylized facts observed in financial markets, similar to what has been found in other calibration experiments of the model. This leads us to extend the Farmer-Joshi model to include agent adaptation using a Brock-Hommes [3] approach to strategy fitness based on trading strategy profitability. The adaptive Farmer-Joshi model allows trading agents to switch between strategies, favouring strategies that have been more profitable over some period of time determined by a free-parameter determining the profit monitoring time-horizon. The novelty of the Farmer-Joshi model is that the dynamics are driven by trade entry and exit thresholds alone. In the adaptive model we are able to calibrate and recover important stylized facts much more completely by combining the interactions of trade entry levels with trade strategy switching based on profitability. We use this to argue that for low-frequency trading across days, as calibrated to daily sampled data, feed-backs can be accounted for by strategy die-out based on intermediate term profitability; we find that the average trade monitoring horizon is approximately two to three months (or 40 to 60 days) of trading.

Declaration of Authorship

We, Ivan Jericevich and Murray McKechnie, declare that this honours project titled, *Calibrating the Farmer-Joshi Agent-Based Model of Financial Markets* and the work presented in it are our own. We confirm that:

- This work was done wholly or mainly while in candidature for a degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely our own work.
- We have acknowledged all main sources of help.

Signed: _____

Date: _____

Acknowledgements

This dissertation is made possible through the continuous support of our supervisor, Associate Professor Tim Gebbie, and we would like to thank him for the guidance given throughout the various stages of this investigation and for instilling in us a passion for statistical finance research.

Furthermore, we would like to thank members of the Statistical Finance Research Group at UCT - in particular Melusi Mavuso, Lionel Yelibi, and Dr Etienne Pienaar - for their helpful feedback, probing questions, and constructive criticism.

Contents

List of Figures

List of Tables

List of Source Codes

1 Introduction and Literature Review

In what follows, we provide a background and context to the relevant literature that relates to the investigation of the Farmer-Joshi model and its extensions. Thereafter, we formulate the model(s) in a mathematical framework and shed light on two different calibration/optimisation techniques along with our methodology and implementation. Lastly, we gather insights and draw conclusions by presenting the results of two different model calibrations.

Financial market time series have been found to exhibit complex structures which cannot be easily explained by standard aggregate economic models or modelled by simple time series models. It is argued that financial markets exhibit many emergent phenomena, and such phenomena are usually attributed to the interactions and relationships between the agents that make up the system [16], which causes models that do not take into account these interactions to fail to accurately replicate the aggregate behaviour observed in the market. Agent-based models (ABMs), meanwhile, demonstrate the ability to produce realistic simulated system dynamics, comparable to those observed in empirical investigations [30]. Successfully calibrating ABMs to financial time series can allow for inference about the factors determining the price behaviour observed in the real world, provided that parameter estimates are sufficiently robust. ABMs can also help determine the effects of different agent strategies on the resulting financial time series.

ABMs provide a bottom-up approach to modelling the actions and interactions of autonomous agents with the aim of assessing their effect on a complex system. ABMs benefit from not assuming agents are able to solve complex problems in order to determine their rational response, but instead assume that different agents follow simple rules in order to cope with their overly complex environments [14]. These agents are often individual people, but can also be larger groups of people such as a household or a firm. Each agent has their own defined rules that govern how they react to different situations. These rule sets may be simple, only taking one or two pieces of information into account at any one time, but they can also be very complex, with agents taking into account past actions, the actions of others in the market, and their past success. The ABM simulates how these different agents interact with each other over time and shows how these interactions influence the market as a whole [14].

More specifically, the Farmer-Joshi model is an inter-day ABM that uses a market maker based method of price formation to study the price dynamics induced by two commonly used financial trading strategies, namely trend following and value investing together with state dependent threshold strategies [8]. As an extension to the Farmer-Joshi model, we consider the case where agents are allowed to switch strategies probabilistically using a Brock-Hommes [3] approach whereby strategies are favoured according to their profitability over a specified time horizon. Given these two models, this paper aims to show how allowing for switching agents results in improved replication of important empirically observed financial market features known as stylized facts.

1.1 Advantages, Disadvantages and Motivation for the Use of ABMs

Financial markets comprise of many interacting components and whose internal dynamics are highly complex. Assuming that markets process the beliefs and demand of traders, this gives some motivation for using a model which considers price movements as a function of the components in the market. In this regard, ABMs successfully link the micro-level rules of investors behaviour with the macro-behaviour of asset prices in real markets. Another compelling reason for the consideration of ABMs in financial markets is that they do not require assumptions about the market and the distribution of returns. This characteristic is ideal since returns are not normally distributed but instead exhibit excess kurtosis (fat tails) as well as other features that lack good explanations by existing models [21]. Due to the expansive amount of data that is collected on financial markets, there is a particularly great amount of potential for using agent-based models to model financial markets [20]. Overall, ABMs clearly have a number of benefits as a simulation tool to show emergent behaviours, evolution dynamics, and consistent fit with

real-market data. In particular, the Farmer-Joshi model is built from a foundation of realistic trading strategies that is argued to be able to reproduce most of these empirical features [8].

ABMs are, however, usually computationally expensive and so have only in recent years become more popular for modelling as the computational tools available have improved. An additional criticism levelled at agent-based financial markets is that there are too many parameters. Researchers are able not just to move freely through large parameter spaces, but can also change entire internal mechanisms at their discretion in the attempt to fit sets of stylized facts [21]. Furthermore, unlike analytic models, there are still relatively few general principles that one can confidently apply to the construction of different agent-based market models, and most financial market ABMs assume only a small number of assets [21]. ABMs also assume that agents operate inductively (use/learn rules and forecasts that have worked well in the past and improve on them). In other words, ABMs do not consider the case where agents operate deductively - that is, not only looking at past patterns but future patterns as well (e.g. present value analysis, Black-Scholes option pricing formula, etc.) [21]. Lastly, there is the issue of validation. It is difficult to compare the "goodness" of one ABM to another. Such comparisons are usually based on the model's ability to replicate empirical observations - which involves subjective comparisons given the qualitative nature of most stylized facts. It is also usually the case that a unique set of parameters for the calibration of a specific model to a time series does not exist. Without a unique set of parameters allowing us to reproduce the properties of financial time series drawn from a particular market, we cannot argue that introducing and observing changes in the model truly reflects the same changes in the market. Aside from the above points and considering the Farmer-Joshi model specifically, an element missing from its price formation mechanism studied is the risk aversion of the market maker (who is assumed to be risk neutral) which can have profound effects on price formation. Furthermore, the Farmer-Joshi model is shown to suffer from parameter degeneracy - suggesting that stylized fact centric validation may be insufficient.

1.2 Market Microstructure

Market microstructure refers to the details of how exchange occurs within markets and its research concerns the study of the process and outcomes of exchanging assets under explicit trading rules [25]. A major underlying tenet of the agent-based modelling philosophy is that studying every single element is sufficient to understand the system as a whole [32]. While this attempt is able to derive analytical solutions for subsequent analysis, it frequently fails to account successfully for stylized facts, especially in financial markets [32]. This is largely because various models in this framework rely heavily on many unrealistic assumptions, such as market clearing, market convergence to equilibrium prices, perfect information, and rationality, whilst ignoring the emergent characteristics of agents interactions and their diverse strategies [32].

In this implementation of the Farmer-Joshi model, we only consider traders partaking in daily closing auctions. Thus, the model does not face the problem of attempting to model intra-day trader behaviour in a continuous market. While share prices are constantly changing throughout each day, attempting to calibrate ABMs on intra-day prices can cause multiple problems. Models considering intra-day price movements can often suffer from parameter degeneracy even if they are able to recreate the stylized facts of the market, suggesting that the parameters no longer carry the meaning they were intended to carry by the model [30]. This leaves limited ability to make regulatory and structural inferences from the obtained results [30]. Closing auctions differ from normal intra-day trading in that no order matching is done, with their purpose being to provide a transparent closing price for each share at the end of each day [1]. All of the trades that take place at the end of the closing auction execute at the same price [1]. Going into each closing auction, agents in the model only make use of previous closing prices to determine their behaviour. As is described in more detail in section 2, the Farmer-Joshi model only allows agents to submit market orders, with no allowance for other types of order such as limit orders, stop orders, and stop limit orders [8]. With market orders, agents do not attach a price to their order, but instead accept whatever closing

price is determined. This means that at the end of the day all orders are always executed in the model. The Farmer-Joshi model therefore does not attempt to include all elements of the market microstructure found in real financial markets, instead focusing on a specific aspect of exchange in financial markets.

1.3 Existing Approaches to Agent Based Modelling

Many different agent-based models exist in the "econophysics" literature, each with its own set of implicit assumptions and interesting properties, with many of the different models equally being able to replicate well-known stylized facts [3, 27]. An example of one of the earliest heterogeneous agent based models can be found in Zeeman (1974) [36]; other examples include Haltiwanger and Waldmann (1985), Frankel and Froot (1988), DeLong et al. (1990), Dacorogna et al. (1995), Gode and Sunder (1993) [12], Lettau (1997) [22], Brock and Hommes (1998) [3], Kirman (2002) [18], Preis et al. (2006) [31] and Jacob Leal et al. (2015) [19].

Similar to the Farmer-Joshi model, the model by Zeeman (1974) [36] considers heterogeneous agents who adopt either a trend following strategy or a value investing strategy to measure the instability of the market in a stock exchange. Variables C, F and J are used to denote the proportion of the market held by chartists, the proportion of the market held by fundamentalists and the rate of change of an index I respectively. The variable J can be regarded as a dependent variable, depending upon the rate of buying and selling of investors. At the same time there is a feedback because the knowledge of J in turn influences the investors. The question is then proposed: how are the variables C, F and J related? Zeeman then expresses this dynamic relation by an ordinary differential equation using a number of different assumptions and hypotheses regarding agents and the stock exchange.

Gode and Sunder (1993) [12] is a distinctive paper as it uses an agent-based model that assumes that agents do not learn and behave almost completely randomly. The agents make random bids and offers, with only a budget constraint - preventing sellers from offering below cost, and buyers from bidding below the share's redemption value. Gode and Sunder found that with this one constraint, the "zero intelligence" traders allocated the resources at over 97 percent efficiency which is very close to that of humans. These results suggest that the behaviour of rational agents with different rule-sets may be indistinguishable from zero-intelligence traders when using a certain market structure [20]. Thus, it is important that researchers try to distinguish between features that come from learning and adaptation, and features that are a direct result of the market structure itself.

In the paper by Lettau (1997) [22] agents have a choice between buying a risky asset that pays some unknown dividend in the next period (the value of which is randomly drawn from a Gaussian distribution), and a risk-free bond paying zero interest. The paper assumes that the price of the risky asset is set exogenously, and that agents have constant risk aversion preferences. With these assumptions, Lettau [22] is able to observe how the expected optimal behaviour of the agents compares to the behaviour determined by the genetic algorithm used to optimize agents behaviour. Interestingly, the genetic algorithm was found to have a bias in favour of more risky strategies when a finite number of experiments were run. This was due to certain agents getting lucky by taking on risky strategies when the dividends happened to be high. However, as the number of experiments increased, this bias was reduced towards zero, as the chance of getting consistently lucky using a high risk strategy was reduced [22].

More recently, the Preis et al. (2006) [31] and Jacob Leal et al. (2015) [19] models consider different approaches to the agent based simulation of financial markets. In the Jacob Leal et al. model, low frequency agents adopt trading rules based on chronological time and can switch between fundamentalist and chartist strategies. High frequency traders activation is event-driven and depends on price fluctuations. These traders use directional strategies to exploit market information produced by low-frequency traders. In the Preis et al. [31] model, simulations are based on an order book which stores the offers and demands

of the various traders and enables continuous trading. Here agents are called liquidity takers/providers and are differentiated on the basis of their order types (market orders or limit orders respectively). Liquidity providers earn the spread by submitting limit orders around the midpoint between the best bid and the best ask whereas liquidity takers submit market orders.

1.4 Calibration of ABMs

In this paper we use an objective function based frequentist approach to model calibration. More recently, however, ABMs have faced criticism regarding the rigour of commonly used calibration practices. Platt (2019) [28] explores how certain calibration techniques perform compared to each other for their ability to calibrate agent-based models in financial markets. The paper compares the commonly used frequentist inference approaches (such as simulated minimum distance methods) to Bayesian inference approaches, and finds that Bayesian estimation techniques tend to perform best, especially as the number of dimensions in the models is increased [28].

The under-performance of frequentist approaches is argued to be the result of the fact that the objective functions used often lack smoothness and possess many local optimums, making discovery of the global optimum using many traditional methods very difficult [11]. Also, any solution for the optimum is only ever an approximate solution due to the stochastic nature of ABMs [7]. For this reason, the use of heuristic methods can be preferable for ABMs, as a heuristic method can be better at obtaining an approximation of the global optimum. Threshold accepting is a heuristic search method which can be used in conjunction with the Nelder-Mead simplex algorithm to calibrate ABMs. Gilli and Winker [11] present a global optimisation heuristic using this Nelder-Mead with threshold accepting technique. One downside of a heuristic approach, however, is the long computation times required to effectively calibrate the model. To offset this, it is often necessary to resort to very coarse approximations of the objective function, which hinders the quality of the results.

The Farmer-Joshi model has been calibrated in past using both a Genetic algorithm (first introduced by Holland in 1975 [17]) and a Nelder-Mead (first introduced by Nelder and Mead in 1965 [24]) with threshold accepting method [7]. The calibrations by Fabretti make use of daily closing auctions of the S&P 500 Composite index from 2005 to 2007 and the results of the two techniques were compared to one another [7]. It was found that the genetic algorithm was able to find slightly better parameters based off of the objective function used compared to the calibration using the Nelder-Mead with threshold accepting method [7]. In this project, we also make use of both of these calibration methods to calibrate the model.

1.5 Stylized Facts of Financial Markets

The view point of many market analysts has been and remains an event-based approach in which one attempts to "explain" or rationalize a given market movement by relating it to an economic or political event or announcement [6]. From this point of view, one could easily imagine that, since different assets are not necessarily influenced by the same events or information sets, price series obtained from different assets and from different markets will exhibit different properties [6]. Nevertheless, the result of many decades of empirical studies on financial time series indicates that it is the case that if one examines their properties from a statistical point of view, then the seemingly random variations of asset prices do share some quite non-trivial statistical properties [6]. Such properties are known as stylized empirical facts and are frequently used as performance measures for ABMs [6].

Stylized facts are thus obtained by taking a common denominator among the properties observed in studies of different markets and instruments. By doing so one gains in generality but tends to lose in precision of the statements one can make about asset returns [6]. Indeed, stylized facts are usually formulated in terms of qualitative properties of asset returns and may not be precise enough to distinguish among different parametric models. Nevertheless, these stylized facts are so constraining that it is not

easy to exhibit even an ad hoc stochastic process which possesses the same set of properties and one has to go to great lengths to reproduce them with a model [6]. We focus on the following stylized facts that are common to a wide variety of financial assets when determining the success of our calibrations:

1. Absence of autocorrelations of returns. The presence of autocorrelations of returns in financial markets would imply the existence of very simple strategies for making money using only knowledge of past share prices. Given the great incentives people have to make money, it follows that such opportunities would quickly be snatched upon by traders up until the point where the opportunity no longer exists through the cancelling out of autocorrelations by the behaviour of the traders. Thus, the existence of this stylized fact when considering returns over a sufficient period of time makes intuitive sense.
2. Heavy/fat tails for the distribution of returns compared to a normal distribution, resulting in the distribution of returns being leptokurtic.
3. Volatility clustering (first noted by Mandelbrot [23]) as shown by autocorrelations in absolute returns (long memory). Large changes in financial markets returns tend to be grouped together.
4. Gain/loss asymmetry. Large drops in price tend to be observed more frequently than large price increases.

It should also be noted that to properly validate an agent-based model for financial markets, the model should both be able to replicate the stylized facts of financial markets and have parameters that behave in clear ways and do not have insignificant effects on the resulting behaviour of the simulation. It has, however, been argued that the qualitative comparison of stylized facts is an inadequate form of validation and that they demonstrate an inability to detect parameter degeneracies [29].

1.6 Agent Adaptation, Herding and Minority Game

The concept of changes in the number of chartists and fundamentalists is driven by the Friedman hypothesis: "irrational agents will lose money and will be driven out the market by rational agents" in a predator-prey type fashion [9]. An important question in heterogeneous agents modelling is whether "irrational" traders can survive in the market, or whether they would lose money and be driven out of the market by rational investors, who would trade against them and drive prices back to fundamentals, as argued by Friedman [9]. Brock and Hommes' paper [3] presents a tractable form of evolutionary dynamics which they call, Adaptive Belief Systems, in a simple present discounted value (PDV) asset pricing model. According to Brock and Hommes, agents revise their "beliefs" in each period in a boundedly rational way, according to a "fitness measure" such as past realized profits [3]. Their paper shows how an increase in the "intensity of choice" to switch predictors can lead to market instability and the emergence of complicated dynamics for asset prices and returns (which is shown in this paper as well). In both the paper by Brock and Hommes and this one, it is shown that when the intensity of switching is high, asset price fluctuations are indeed characterized by an irregular switching between phases where prices are close to the EMH fundamental price, phases of optimism ("castles in the air") where traders become excited and extrapolate upward trends, and phases of pessimism, where traders become nervous causing a sharp decline in asset prices [3]. A key feature of both our and Brock and Hommes' adaptive belief systems is that this irregular switching is triggered by a rational choice between simple prediction strategies.

While we use a Brock-Hommes approach to agent adaptation, there are other methods also commonly used. Herding is known to be a key source of endogenous fluctuations in asset prices. For the past several decades, finance academics have considered herding behaviour in ABMs, trying to fit ABMs to stylized statistical features salient in empirical data. Herd behaviour was first observed in ant colonies by entomologists. Kirman (2007) [18] develops an ABM to analyse and adapt ants' "recruiting" behaviour to explain agents herding behaviour in financial markets. The similarity between ant societies and financial markets, in terms of herding, allowed Kirman to apply a multi-agent concept to the dynamics in financial markets. This concept defines the probabilities that one agent will follow another agent's opinion ("herding") and

that agents will change their decisions on their own ("self-conversion"). More specifically, Kirman's model considers chartists and fundamentalists who interact and communicate their beliefs on the next period forecast through an epidemiological process. There is a fixed number of agents N where θ_t is the number of agents with a fundamentalist forecast at time t. It is assumed that pairs of agents meet at random and that the probability that the first agent is converted to the opinion of the second one is equal to $(1-\delta)$. Furthermore, each agent can independently change his opinion with probability ξ. Given that the state of the process is summarised by the value of θ_t, the transition probability matrix is given by

$$\Pr[\theta, \theta+1] = \left(1 - \frac{\theta}{N}\right)\left(\xi + (1-\delta)\frac{\theta}{N-1}\right)$$
$$\Pr[\theta, \theta-1] = \frac{\theta}{N}\left(\xi + (1-\delta)\frac{N-\theta}{N-1}\right)$$
$$\Pr[\theta, \theta] = 1 - \Pr[\theta, \theta+1] - \Pr[\theta, \theta-1]$$

After the meetings, the proportion of fundamentalists is equal to θ_t/N (determined by the transition probability matrix). However, agents observe this proportion with error. Agent i observes $k_{i,t} = \theta_t/N + \epsilon_{i,t}$ where $\epsilon_{i,t} \sim N(0, \sigma_\theta^2)$. If agent i observes that $k_{i,t} \geq 0.5$ then he will make a fundamentalist forecast, otherwise he will make a chartist forecast (the proportion of fundamentalist forecasts at time t is $N^{-1}\sum_{i=1}^{N} I_{(k_{i,t}\geq 0)}$). This model is argued to replicate the empirically observed characteristics of daily exchange rate series [18].

Another alternative means for agent adaptation to be considered is the minority game approach (first formulated by Challet and Zhang in 1997 [5]) which is an intuitive model for the behaviour of a group of agents subject to the economic law of supply and demand. In the standard minority game, agents learn and take their decisions to be in minority. Based on supply and demand relation, the winners will make a profit from buying low and selling high, so we can say these agents are value traders and have minority game character. On the other hand, another type of agent will have a majority game character. If the price continues to go up, driven by the majority, this kind of agent will make a profit, so we can say it is a trend trader. This gives the game its "minority" nature; an excess of buyers will force the price of the asset up, consequently the minority of agents who have placed sell orders receive a good price at the penalty of the majority who end up buying at an over-inflated price. In an attempt to learn from their past mistakes the agents constantly update the "profit" or "success" of their strategies and use only the most successful one to make their prediction. The agents of the MG keep a tally of the virtual score U_i's of each of their strategies, $+1$ for a correct prediction and -1 for an incorrect prediction. Asserting that agents are rational and risk averse implies that an agent never plays a strategy that has lost more times than won. In this model, each agent has two strategies: a value strategy and a trend strategy. Both strategies for each agent are used to predict at each time step and scores are given to those strategies. Agents keep a tally of their virtual score $U_{i,c}$ and $U_{i,t}$ for each of their strategies (i.e. score made from value investing (f) and score made from trend following (c)) through time as follows [35]:

$$U_{i,f}(t+1) = U_{i,f}(t) - a_i^I(t) \cdot \text{sign}(A(t+1)) - U_{i,f}(t-H)$$

$$U_{i,c}(t+1) = U_{i,c}(t) + a_i^I(t) \cdot \text{sign}(A(t+1)) - U_{i,c}(t-H)$$

where $a_i^I(t)$ is the action chosen by the ith value investor at time t (1 = buy, -1 = sell) based on the value strategy, $A(t+1) = \sum_{i=1}^{N} a_i(t)$ provides the collective sum of actions from all agents at time t and H represents the horizon for which the strategy records its score. That is to say that only virtual points in the last H steps can be added to $U_i(t)$. Different from value strategies, trend strategies have a positive payoff if $a_i(t)$ and $A(t+1)$ have the same sign. So if a trend follower's action is in the majority in the next time step they gain. However, if a value investor is in the majority at the next time step they lose a point (i.e. if $a_i(t)$ and $A(t+1)$ have the same sign). The best strategy adopted by the ith agent at time t is thus given by $s_i(t) = \underset{s}{\text{argmax}}[U_{i,s}(t)]$ [35]. In this way agents always choose the best strategy

(deterministic), however, one may specify that agents employ their strategy s in a probabilistic fashion as introduced by Cavagna et al. (1999) [4]. Cavagna et al. define the Thermal Minority Game model where agents are allowed a certain degree of stochasticity in their choice of strategy to use at any time step [4] by introducing a "thermal" description which progressively allows stochastic deviations from the "best strategy" rule. An agent will adopt strategy s with probability

$$\phi_t^{s,i} = \frac{e^{U_{i,s}(t)/\Gamma}}{\sum_{s'} e^{U_{i,s'}(t)/\Gamma}}$$

Cavagna et al. terms Γ as "temperature". This decision-making process by agents is specified in more detail in section 2.5.

On the other hand, a strong argument can be made for refraining from allowing for agent adaptation in ABMs. For example, it would be desirable if the original Farmer-Joshi model were able to replicate the stylized facts, as the authors argue that traders do not constantly consider different strategies and change between them as is often the case in adaptive agent models [8]. Ideally Farmer and Joshi wish to be able to replicate observed price behaviour from the internal mechanisms of agents' strategies and the different ways in which the two strategies are activated [8]. By including adaptive agents in the model, we are adding an explicit model definition in order to better replicate the stylized facts but at the expense of possibly reducing the realism of the agent behaviour, even if such behaviour could be argued to be more rational than simply sticking to a predefined strategy.

1.7 Ergodicity

A short note on the property of ergodicity of financial market time series is made here and is supplementary to the explanation of the implementation of the Farmer-Joshi model. First consider two types of averaging: ensemble averaging and time averaging. Here we will consider discrete processes. The finite-ensemble average of the quantity z at a given time t is

$$\langle Y(t^*)\rangle_N = \frac{1}{N}\sum_{i=1}^{N} Y_i(t^*)$$

where $Y_i(t)$ represents a particular realization of $Y(t)$ and N is the number of realizations. The subscript indicates that it is a function of N. If Y only changes at $T = \frac{\Delta t}{\delta t}$ discrete times $(t+\delta t,\, t+2\delta t, \ldots)$, then the finite-time average of the quantity $Y(t)$ is given by

$$\overline{Y}_i(t) = \frac{1}{T\delta t}\sum_{r=1}^{T} Y_i(t + r\delta t)$$

where T is the length of the time series and t is the start date. An observable process Y is called ergodic (mean ergodic) if its expectation value is constant in time and its time average converges to this value with probability one [26].

$$\lim_{\Delta t\to\infty} \frac{1}{\Delta t}\sum_{r=1}^{T} Y_i\left(t + r\frac{\Delta t}{T}\right) = \lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N} Y_i(t^*)$$
$$\overline{Y}_i = \langle Y(t^*)\rangle$$

The left-hand side is a function of the realization universe i whilst the right-hand side is a function of time t. Figure 1 demonstrates the property of ergodicity by showing that averaging a stochastic process over time or over the ensemble are completely different operations which can rarely be interchanged.

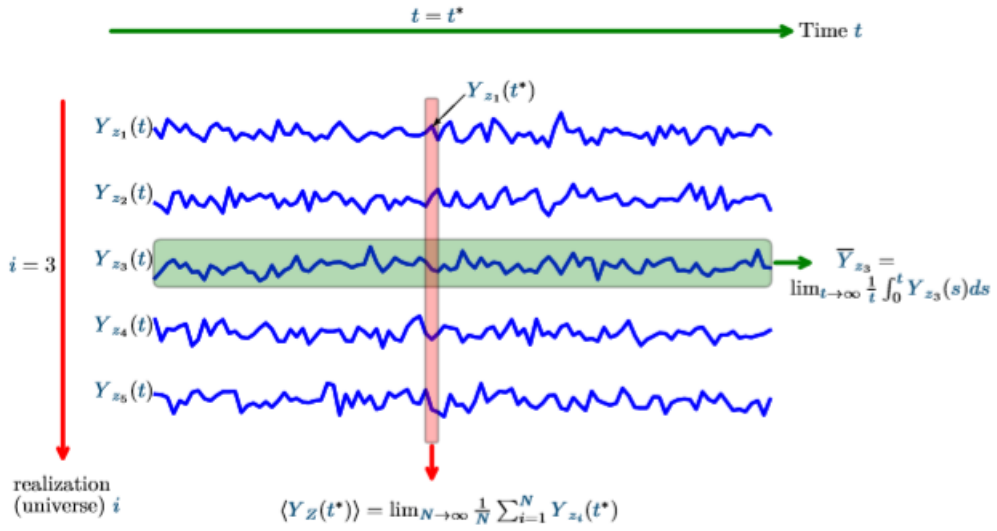

Figure 1: Ergodic property visualization

It should be noted that exchange rate time series are not ergodic. However, it was noted by Daniel Bernoulli (1954) [2] that applying a specific transformation to a price time series (say, Y) can make it ergodic. This transformation is given by

$$\begin{aligned}\Delta \log(Y_t) &= \log(Y_t) - \log(Y_{t-1}) \\ &= \log\left(\frac{Y_t}{Y_{t-1}}\right) \\ &= \log(R_t)\end{aligned}$$

Therefore log returns are ergodic and $\langle \Delta \log(Y)\rangle$ (ensemble) specifies what happens to Y over time.

2 Model Formulation

In what follows we describe the Farmer-Joshi agent based model that is to be calibrated, based off the original paper [8]. The Farmer-Joshi model is important to consider as it presents a simplified framework through which to develop an understanding of the basic structure of financial ABMs. The model considers three different agents, namely: chartists (trend followers), fundamentalists (value investors), and a risk neutral market maker in order to aggregate demand. Demand of individual agents is expressed in terms of orders. The two most common types of orders are market orders and limit orders. The model studies only market orders. Agents observe the market process and information about securities and initiate trades based on these observations. More specifically, during each of T discrete simulation days, the following occurs:

1. The trader agents observe the most recent prices $(P_t, P_{t-1}, \ldots, P_{t-d})$ and the information I_t and demands to buy or sell certain quantities of an asset to a risk-neutral market maker by submitting an order ω_t^i at the end of the day.
2. The market maker then fills the requested orders at a newly determined market price (P_{t+1}) based on a closed-form equation aggregating trader agent demands.

The actions of each trader agent are determined by the strategies which they adopt. Each strategy induces price dynamics that characterize its signal processing properties. Each simulation consists of N trader agents enumerated by index i. Letting $p_t = \log P_t$, at a single time step t the ith trader agent sets their desired demand for the asset, defined as the quantity they intend to buy or sell, according to

$$\omega_t^i = x_t^i - x_{t-1}^i \tag{1}$$

where $x_t^i = x_t^i(p_t, p_{t-1}, \ldots, I_t)$ is the position at time t of the ith trader which is set according to the agent's associated strategy represented by the function x_t^i.

2.1 Market Maker

The market maker bases the price formation only on the net order:

$$\omega_t = \sum_{i=1}^{N} \omega_t^i \tag{2}$$

The market maker determines the price for the net order using the market impact function which relates the net of all orders at any given time to prices. In this way, buying drives the price up, and selling drives it down. The fact that orders, positions and strategies are anonymous gives motivation for basing price formation only on the net order. The market impact function is

$$P_{t+1} = P_t \exp \frac{\omega_t}{\lambda} \tag{3}$$

where λ is called the liquidity parameter. Letting $p_t = \log P_t$ and adding a noise term ζ_{t+1}, equation (3) becomes

$$p_{t+1} = p_t + \frac{\omega_t}{\lambda} + \zeta_{t+1} \tag{4}$$

where ζ_{t+1} is drawn from $N(0, \sigma_\zeta)$. The addition of the random term ζ_{t+1} can be interpreted in one of two ways. It can be thought of as corresponding to "noise traders", or "liquidity traders", who submit orders at random [8]. Alternatively, it can be thought of as corresponding to random changes in the price, for example, random information that affects the market makers price setting decisions [8].

2.2 Trend Followers

Trend followers invest based on the belief that price changes have inertia (so their positions are positively correlated with recent price changes). A trend strategy takes a positive (long) position if prices have recently been going up, and a negative (short) position if they have recently been going down. Hence, we set the ith trend follower's position at time t according to

$$x_{t+1}^i = c^i \cdot \text{sign}(p_t - p_{t-d^i}) \tag{5}$$

where c^i is a positive constant and d^i is the time lag of the ith agent. In this case, the model assume chartists care only about the direction of the change from the lagged price to the current price and not about the magnitude of this change. In this way, the magnitude of the ith agent's position at any point in time will remain constant at c^i with the direction of the position being determined by $\text{sign}(p_t - p_{t-d^i})$ at any point in time t. Trend strategies overall amplify the noise in prices. They reinforce and magnify the ups and downs of price movements, introducing extra volatility and irrational valuations of the security which make information aggregation and dissemination more difficult.

2.3 Value Investors

Value investors, on the other hand, make a subjective assessment of the value of a stock in relation to its price. They believe that their perceived value may not be fully reflected in the current price, and that future prices will move towards their perceived value. Hence, their decisions are based on price deviations from a perceived fundamental asset value. They attempt to make profits by taking positive (long) positions when they think the market is undervalued and negative (short) positions when they think the market is overvalued. Hence, we set the ith value investor's position at time t according to

$$x_{t+1}^i = c^i \cdot \text{sign}(v_t^i - p_t) \tag{6}$$

where $c^i > 0$ is a constant proportional to the trading capital and v_t^i is the logarithm of an investor's fundamental value perception at time t. Similar to trend followers, this assumes fundamentalists care only about the direction of the change from the current price to the perceived value and not about the magnitude of this change. In this way, the magnitude of the ith agent's position at any point in time will remain constant at c^i with the direction of the position being determined by $\text{sign}(v_t^i - p_t)$ at any point in time t. The model assumes the logarithm of the fundamental value follows a random walk given by

$$v_{t+1}^i = v_t^i + \eta_{t+1} \tag{7}$$

where η_{t+1} is a normal, IID noise process with mean μ_η and standard deviation σ_η. For the purposes of this paper it does not matter how individual agents form their opinions about value. We take the estimated value as an exogenous input, and focus on the response of prices to changes in it [8].

2.4 State-Dependent Threshold Strategies

A concern with simple position-based value/trend strategies is excessive transaction costs. Trades are made whenever the mispricing changes. If positions were changed at every simulation step, this would equate to excessive transaction costs in a real-world setting, making the model incredibly unrealistic. To ameliorate this problem and reduce trading frequency strategies, a threshold can be used for entering a position and another threshold for exiting. In this way not all fundamentalist/chartist trader agents are active in all simulation steps, but rather activate (enter or exit positions) only when a certain threshold of mispricing ($m_t^i = p_t - v_t^i$ for fundamentalists or $m_t^i = p_{t-d^i} - p_t$ for chartists) is realized. By only entering a position when the mispricing is large, and only exiting when it is small, the goal is to trade only when the expected price movement is large enough to beat transaction costs.

The model thus assumes that each trader agent has an associated position entry threshold, T^i, drawn from $U(T_{min}, T_{max})$, and an associated position exit threshold, τ^i, drawn from $U(\tau_{min}, \tau_{max})$. A short position $-\omega^i$ is entered when the mispricing exceeds a threshold T^i ($m_t^i > T^i$) and exited when it goes below a threshold τ^i ($m_t^i < \tau^i$). Similarly, a long position ω^i is entered when the mispricing drops below a threshold $-T^i$ ($m_t^i < -T^i$) and exited when it exceeds $-\tau^i$ ($m_t^i > -\tau^i$). A trader who is very conservative about transaction costs, and wants to be sure that the full return has been extracted before the position is exited, will take $\tau^i < 0$. However, others might decide to exit their positions earlier, because they believe that once the price is near the value there is little expected return remaining. We can simulate a mixture of the two approaches by making $\tau_{min} < 0$ and $\tau_{max} > 0$. However, to be a sensible value strategy, a trader would not exit a position at a mispricing that is further from zero than the entry point. Therefore τ_{min} should not be too negative, so we should have $-T^i < \tau^i < T^i$ (so that the exit threshold is between the entry threshold for a long and short position) and $\tau_{min} \leq T_{min}$. Thus, this strategy depends on its own position as well as the mispricing, as shown in figures 2 and 3. Finally, c^i is chosen so that $c^i = a(T^i - \tau^i)$ where a is a positive constant called the scale parameter for capital assignment.

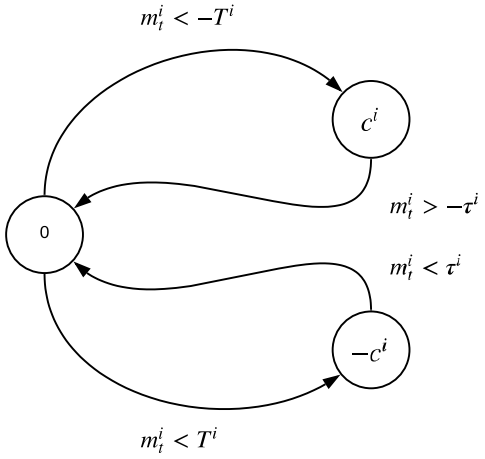

Figure 2: State dependent threshold strategy flowchart

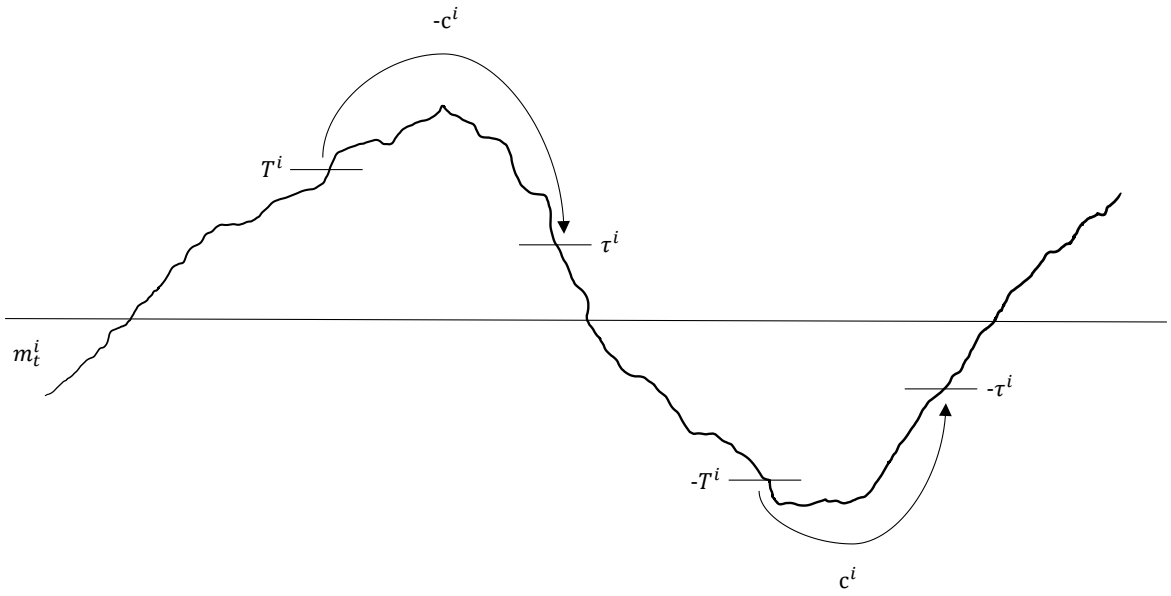

Figure 3: State dependent threshold strategy visualisation

The strategies mentioned are only a few of the strategies actually used in real markets. But they are known to be widely used, and understanding their influence on prices provides a starting point for more realistic behavioural models. To implement the standard Farmer-Joshi model, the above formulation is all that is required, however, we consider a digression from this model by considering an additional strategy adopted by all agents along with the above strategies. This added strategy is detailed in the section below and will be encapsulated in the model which we shall call the adaptive Farmer-Joshi model.

2.5 Agent Adaptation

So far we have dealt with "zero intelligence" [12] traders. The number of traders in the system and the strategy they use remains constant throughout the simulation. The Farmer-Joshi model is novel in this respect as it attempts to recreate the fluctuating volatility found in real markets using solely the different

activation rules defined by the thresholds for each type of trader. The authors of the Farmer-Joshi model take issue with the speed at which agents change strategies in models that involve agent adaptation [8]. This simplicity has allowed us to analyse more easily the interactions between agents and the effects of entry and exit thresholds on price behaviour, but this is at the expense of not being able to reproduce the stylized facts as we find in section 5.1. Therefore, in this adaptive model, we allow for the inclusion of agent adaptation. Despite the argument that it is unrealistic to assume traders making daily decisions on which strategy to take, we argue for this adaptive model for two reasons:

1. Firstly, the act of agents switching from one strategy to another in the model does not have to be seen as agents changing their strategy. While our description of the adaptive model and its internal dynamics imply this, the model can also be seen as a simplification of the situation where an old agent "dies out", and a new agent enters the market at the same time with a different strategy. In real financial markets we know it to be the case that the set of traders in a market is not held constant, but is in fact regularly changing.
2. Secondly, while allowing for traders to switch strategies with such regularity may be unrealistic, it is also unrealistic to expect traders to continue trading throughout the period measured no matter what their profits or losses are. Models necessarily have to make simplifications about the real world, with their goal being to simplify the least relevant aspects of the real world while more completely capturing the aspects that are most relevant to the behaviour we observe.

It should also be noted that trend trading is one of the most important factors leading to excess volatility, even to bubble activation and breaking. So, in order to replicate clustered volatility, we need to include a strategy which allows for the inclusion of more trend followers during periods of the simulation to amplify noise. By the nature of their strategies, trend followers induce positive short-term autocorrelations and value investors induce negative short-term autocorrelations.

In a real securities market, investors do not have perfect information, nor do they have perfect ability to profit themselves by handling the information. So investors have no incentive to persist in a certain trading type. They must make decisions based on inductive thinking; do their best to adapt to the evolving market. So, every agent should be able to act as a trend follower or value investor in different situations based on her own decisions. This agent, who can choose their trading type, is a kind of more intelligent agent. Therefore, we include an adaptation method whereby each agent is equipped with both a value and a trend strategy, and they act in order to try maximise their financial performance by considering the relative profitability of each strategy.

In the standard model, value investors observe fundamental information and do not try to extract information from price trends; on the other hand, trend followers attempt to identify price trends to profit by trend chasing and do not care about the fundamentals. Interaction of the two groups of traders results in the so-called over-reaction and under-reaction. This is the heterogeneous agent paradigm, in which each agent is endowed with unperfected intelligence and so the trading type (value trading or trend trading) of traders is fixed and determined by exogenous factors. In our new model, each agent is endowed with both value and trend strategies, and agents can adapt themselves to the evolving system by changing trading type between value trading and trend trading. We demonstrate in section 5.2 that when agents frequently change their trading style to adapt themselves to the evolving system, the aggregate outcome of the system is more similar to a real market in terms of the stylized facts produced. In an attempt to learn from their past mistakes the agents constantly update the "profit" of their strategies and favour using the more profitable strategy over the less profitable strategy.

The number of traders in the system at any point in time is fixed, N, with the number of available strategies being $2N$. We denote the ith agent's fundamentalist and chartist strategy position at time t by $x_t^{f,i}$ and $x_t^{c,i}$ respectively. Throughout the implementation we keep track of which strategy each agent is adopting. Furthermore, we keep track of the theoretical profit made from each agent's trend following

and value investing strategy (even if they are not currently implementing the given strategy). Profit each day is calculated from the difference between the today's closing log price and yesterday's closing log price, and the (theoretical) position that would have been taken over the past day for the given strategy. Traders accumulate profits over the H most recent days. More formally:

$$\pi_t^{s,i} = \sum_{k=t-H+1}^{t} (x_{k-1}^{s,i}(p_k - p_{k-1})) \tag{8}$$

where

$s =$ the strategy adopted by the ith agent

$\pi_t^{s,i} =$ the profit earned from strategy s adopted by the ith agent at time t

$H =$ the positive real lag parameter for which investors keep track of their profits (exogenous)

We thus include the extra parameter H, whose value is chosen in the optimisation routine. After every day each agent's profit is updated based on the price and the positions they took. By keeping track of accumulated profits, agents are able to switch strategies according to which one would have earned them more profit in the recent past. The mechanism through which this done is through a probability of switching as introduced by Cavagna et al [4]. In order to analyse the long time dynamics of agent adaptation, it is useful to get rid of the discrete characteristics of the game, such as the "always play the best strategy" rule. Instead, one can introduce a probabilistic strategy selection rule that favours well performing strategies. Our method is to impose that agent i adopts their strategy s with a probability that is based on accumulated profit:

$$\phi_t^{c,i} = \frac{e^{\pi_t^{c,i}/\Gamma}}{e^{\pi_t^{c,i}/\Gamma} + e^{\pi_t^{f,i}/\Gamma}} \tag{9}$$

$$\phi_t^{f,i} = 1 - \phi_t^{c,i} \tag{10}$$

where

$\Gamma =$ positive real parameter for intensity of switching

$\phi_t^{s,i} =$ ith agent's probability of adopting strategy $s \in \{\text{fundamentalist, chartist}\}$ at time t

The intensity of switching parameter (Γ) may be interpreted as how sensitive the mass of traders is to differences in profitability across trading strategies.

3 Model Calibration

3.1 Objective Function

For the problem of calibrating the chosen models, the objective function is required to take into account the stylized facts of financial data. A good objective function can be constructed by making use of a set of k moments that best describe the statistical features of financial data [34]. In this regard, searching for the best set of parameters (calibration) is a matter of matching empirical moments with the moments from simulated data. The approach used for estimating parameters will therefore be the method of simulated moments. So, the objective function is chosen to be a combination of estimation errors on moments and statistics. The problem to be solved is thus a minimization problem:

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} f(\boldsymbol{\theta}) \tag{11}$$

where $\boldsymbol{\theta}$ is the vector of parameters, $\boldsymbol{\Theta}$ is the space of feasible parameters, and f is the objective function. Since some properties may be more useful/important in replicating stylized facts, an important issue

in the construction of the objective function relates to how each of these moments and statistics are weighted in the objective function. Secondly, it cannot be assumed that these moments are distributed independently from each other which gives motivation for the use of a weight matrix taking into account the joint distribution of moments [34].

Denoting by $\boldsymbol{m^e} = [m_1^e, \ldots, m_k^e]'$ and $[\boldsymbol{m^s} \mid \boldsymbol{\theta}] = [m_1^s, \ldots, m_k^s]'$ the vector of empirical moments and statistics of real and simulated data, respectively, the method of moments requires that

$$E[[\boldsymbol{m^s} \mid \boldsymbol{\theta}]] = \boldsymbol{m} \implies E[[\boldsymbol{m^s} \mid \boldsymbol{\theta}] - \boldsymbol{m}] = 0 \tag{12}$$

Using the empirical moments, $\boldsymbol{m^e}$, as an estimate for the true moments, $\boldsymbol{m}$, and calculating the expectation using the arithmetic average we have that the estimation error is defined as the average deviation of simulated moments from empirical moments [15]:

$$G(\boldsymbol{\theta}) = \frac{1}{I} \sum_{i=1}^{I} (\boldsymbol{m_i^e} - [\boldsymbol{m_i^s} \mid \boldsymbol{\theta}]) \tag{13}$$

where I is the number of replications/simulations used in estimating the average moments and statistics that the given parameters produce. A high value of I is preferable because it reduces the objective function variance and reduces the chance of abnormally high fitness values from sub-optimal parameters, however this is at the cost of computational time. Denoting by $\boldsymbol{W}$ the $k \times k$ matrix of weights of moments and statistics, the objective function is defined as the quadratic function

$$f(\boldsymbol{\theta}) = G(\boldsymbol{\theta})' \boldsymbol{W} G(\boldsymbol{\theta}) \tag{14}$$

Parameters are estimated through minimization of the sum of squares of the deviations of simulated moments from empirical moments. According to Heij et al. [15], the matrix $\boldsymbol{W}$ is given by the inverse of the covariance matrix of the distribution of moments ($Var^{-1}[\boldsymbol{m^e}]$) [15]. This selected weight matrix takes the uncertainty of estimation associated with $\boldsymbol{m^e}$ into account and assigns larger weights to moments associated with lower uncertainty and vice-verse. The obtained variance can be used to weight each statistic in such a way that their contribution is equally balanced in the determination of the function value.

The weight matrix $\boldsymbol{W}$ is estimated by applying a moving block bootstrap to the time series with a window of size b (explained in more detail in section 4). We create 10 000 bootstrapped samples from which we calculate each of their sets of moments and statistics to be used in the calculation of the covariance matrix of $\boldsymbol{m^e}$. In other words, for each block/window we re-sample with replacement until we obtain 10 000 samples, each of length equal to the length of the original data set. The moments and statistics are then calculated on each of these samples such that we can obtain the covariance between them. The motivation for the use of a moving block bootstrap is that we cannot assume the selected moments are independently distributed. As stated in section 1.5, financial market time series do exhibit significant auto-correlations for absolute log returns. To keep this property in the bootstrap samples, we must sample blocks of data at a time instead of individual daily returns.

Even by sampling in blocks, however, we are likely to lose some of the long term auto-correlations observed in the real data due to the random way in which blocks are ordered in the bootstrap sample. The smaller the block size, the greater this effect is. A block size of 1, for example, would result in bootstrap samples where any auto-correlations observed are in no way the result of auto-correlations observed in the original time series. On the other hand, choosing a block size that is too big can lead to bad estimates of the variances of the moments that are being calculated. Clearly, a block size equal to the length of the data would result in a variance of 0 for all the moments, as the exact same block would be sampled for each bootstrap sample. Thus, it is important to choose a block size that balances the wish to

preserve the auto-correlations observed in the real world while also allowing for a reasonable estimation of how much each of the moments and statistics measured varies in the real world. We choose a block size of 100, which we believe allows for the bootstrap samples to retain sufficient autocorrelations while allowing for large enough differences between samples. Such a block size has also been used by previous papers that made use of block bootstrapping for financial time series calibration [7, 29].

Finally, a key requirement for the construction of the objective function is the choice of statistics and moments. Our choice of moments and statistics should be robust enough to reflect the properties of financial data and flexible enough to discriminate between different models [34]. A few well-known stylized facts of financial data include fat tails, volatility clustering and persistence in price. Therefore, the following moments and statistics are used in the objective function for being the most representative measures of the stylized facts of financial data. We use a combination of the moments and statistics suggested for use in [34] and the moments used in [7]:

1. Mean of log returns.
2. Standard deviation of log returns.
3. Excess kurtosis relative to the kurtosis of the normal distribution for log returns.
4. Kolmogorov-Smirnov statistic for log returns. This compares the empirical distribution of the actual returns to that of the simulated or bootstrapped returns obtained from the model. The actual returns obtain a K-S statistic of 0 as they are identically distributed to themselves.
5. Simplified Hurst exponent, which represents the scaling properties of the log returns. There are multiple different ways to calculate the Hurst exponent; we choose the simplified Hurst exponent due to its low variance between bootstrap samples of the data.
6. The Geweke and Porter-Hudak (GPH) estimator, which provides a measure of the long-range dependence of the absolute log returns.
7. Augmented Dickey-Fuller (ADF) statistic, which is a measure of the extent of the random walk property of log returns.
8. The sum of the two GARCH(1, 1) parameters. This is used as a measure of short-range dependence. The reason for taking the sum is that this value is much more robust than using either one of the two parameters in the GARCH model alone.
9. The average of the Hill estimator on the right tail of the distribution of returns from the 90th to the 95th percentile. This is a measure of the fat tails on the right-hand side of the return distribution. This statistic is added due to kurtosis having a very high variance, which results in it obtaining a very small weight in the objective function. The Hill estimator, meanwhile, is much more robust - meaning it has a low variance and thus is able to take on a much greater weight in the objective function.

The mean, the standard deviation, the kurtosis, Hill estimator and the K-S statistics are chosen to represent the overall shape of the data distribution [7]. We include the Hill estimator as an additional measure of the tailedness of the distribution over and above kurtosis due to the very low weight kurtosis takes on in the objective function. The Hurst exponent has been largely investigated in the literature because of its appealing feature to synthesize the scaling characteristic as a unique index [7]. Overall, this results in a combination of $k = 9$ moments and statistics being used to characterize the statistical properties of the closing share price data.

Given the above objective function, the parameters of the Farmer-Joshi and adaptive Farmer-Joshi ABM to be estimated are presented in table 1. For the standard Farmer-Joshi model, all but the last two parameters are used, with the number of chartists (N_c) set to equal the number of fundamentalists (N_f). In the adaptive model, the number of traders is given by $N = N_c = N_f$ as each trader has both a fundamentalist and a chartist strategy available.

$\boldsymbol{\theta}$	Description
N_f, N_c	Number of traders of each type
λ	Liquidity parameter
a	Scale parameter for capital assignment
d_{min}, d_{max}	Minimum and maximum time delay for trend followers
μ_η, σ_η	Mean and standard deviation of noise process in v_t
σ_ζ	Noise variance in market maker's price setting decisions
T_{min}, T_{max}	Minimum and maximum threshold for entering positions
τ_{min}, τ_{max}	Minimum and maximum thresholds for exiting positions
v_{min}, v_{max}	Minimum and maximum offset for log of perceived value
Γ	Intensity of switching (adaptive model only)
H	Time Horizon for profit tracking (adaptive model only)

Table 1: Parameters to be estimated in the objective function.

For each chartist, their individual time lag is randomly sampled as an integer between d_{min} and d_{max}, while each fundamentalist's value offset is sampled from a uniform distribution distribution with minimum v_{min} and maximum v_{max}. All traders' entering thresholds are sampled from a uniform distribution between T_{min} and T_{max}, and all traders' exiting thresholds are sampled from a uniform distribution between τ_{min} and τ_{max}.

3.1.1 Stability of the Objective Function

The Farmer-Joshi model contains many stochastic elements, and thus it is possible for different simulations using the same parameter to produce significantly different results. This can lead to the same parameters producing substantially different objective function values when making use of a different set of simulations. One way to reduce this variance is to use a trimmed (truncated) mean of moments instead of a standard mean as suggested in [34]. This allows us to ignore the extreme values obtained from some replications for each of the moments and statistics. To determine the best percentage to trim off of the results from our replications, we estimate the variance of the objective function for different trim percentages as shown in figure 4. These variance estimates are obtained by using a reasonable set of parameters on the standard Farmer-Joshi model to obtain 10 different objective function values each for percentages trimmed equal to $0, 10, 20, \ldots, 90$, and then calculating the sample variance for each percentage trimmed. We observe that setting the percentage trimmed to 70% results in the least variance of the objective function, so we use a 70% trimmed mean to run our final calibration experiments for the standard Farmer-Joshi model. This implies that we ignore the lowest and highest 35% of values obtained for each moment from the replications. The trimmed mean of moments is, however, not used in the adaptive Farmer-Joshi model since the number of replications (I) in this model is kept relatively low compared to the standard model in order to prevent very long computation times[1].

[1]Unlike for the standard Farmer-Joshi model, the speed of the adaptive model was unable to be improved through conducting many simulations at once due to its more complex implementation. Section 4.1 shows the discrepancy in computation times

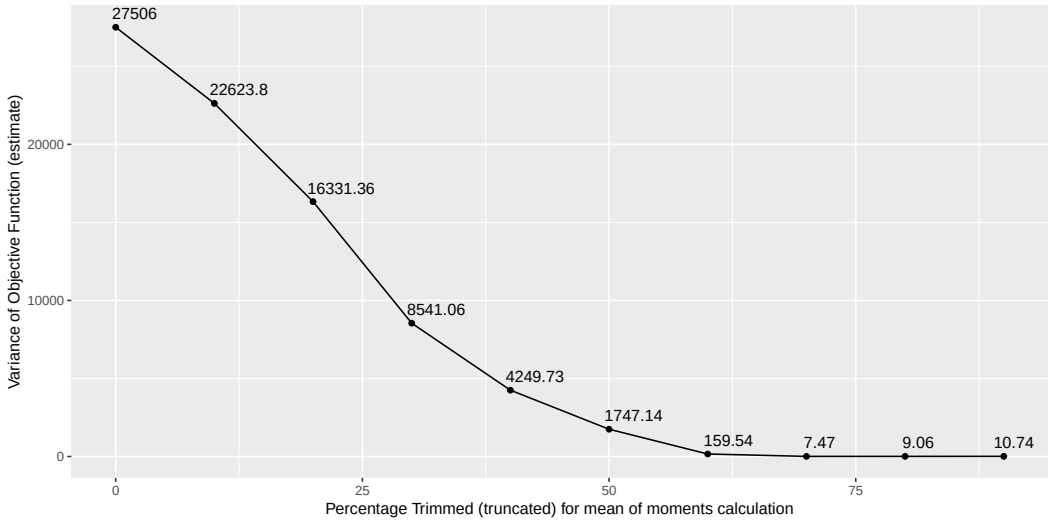

Figure 4: Estimated variance of the objective function for the standard Farmer-Joshi model

3.2 Calibration Methods

Standard local optimisation methods are prone to failure because such an objective function does not always behave well to guarantee a global optimum solution. As we observe in section 5.2.1, the objective function built, when taking into account the statistical properties of data, is not smooth or globally convex. There tend to be many local optima, so it is important to use an optimisation routine that is able to escape these local optima. Furthermore, due to the complexity of the problem and the nature of agent based models, finding a global optimum is very computationally intensive, so any optimisation routine used should be able to efficiently explore the parameter space. Using the methodology proposed in the paper by Gilli and Winker [11], we describe two methods for the global optimisation of the objective function, namely a Nelder-Mead with threshold accepting algorithm and a genetic algorithm.

3.2.1 Nelder-Mead Simplex Algorithm with Threshold Accepting Heuristic

The Nelder-Mead simplex method provides an efficient way of identifying the search direction while threshold accepting helps to avoid local minima. Gilli and Winker provide a number of different ways to combine the two algorithms [11], and variations of this optimisation method have been used for many different ABM calibrations in the literature [7, 27, 30]. This section provides a description of each method separately and then describes how we combine them to solve the problem at hand.

In the Nelder-Mead (NM) simplex algorithm each current solution consists of the vertices of a simplex in the parameter space $\boldsymbol{\Theta}$ [11]. The simplex exists in the parameter space and always has $n+1$ vertices where n is the number of freely moving parameters (the number of dimensions of the parameter space). Each vertex represents a set of values for the freely moving parameters of the model. At each iteration, the algorithm shifts the simplex according to the following process:

1. **Reflection** - The vertex with the worst fitness (labelled x_{n+1}) according to the chosen objective function is reflected through the centroid (x_0) of the remaining vertices so that it represents a new set of values for the parameters. This new vertex is accepted if its fitness is better than the new worst vertex of the new simplex.

$$x_r = x_0 + \alpha(x_0 - x_{n+1}) \qquad \text{for some } \alpha > 0 \tag{15}$$

2. **Expansion** - If the reflection step resulted in the shifted vertex obtaining the best fitness value of all the current vertices, that same vertex is shifted further in the same direction as in the previous

step. The new vertex is then accepted if it obtains an even better fitness than vertex obtained from the reflection.

$$x_e = x_0 + \gamma(x_r - x_0) \qquad \text{for some } \gamma > 1 \tag{16}$$

3. **Contraction** - Should the reflection not have been accepted, the algorithm again takes the worst vertex, but now only shifts it towards the centroid of the remaining vertices. If this results in an improvement in fitness for the vertex, the shift is accepted.

$$x_c = x_0 + \rho(x_{n+1} - x_0) \qquad \text{for some } 0 < \gamma \leq 0.5 \tag{17}$$

4. **Shrinkage** - If none of the above three shifts were accepted, each vertex of the simplex is shifted towards the vertex that has the best fitness (x_1). This step is always accepted, and it results in the simplex shrinking in size.

$$x_i = x_0 + \sigma(x_i - x_1) \qquad \text{for some } 0 < \sigma < 1 \tag{18}$$

The above steps are repeated for whatever number of iterations are specified, or until some other criterion is met. This method appears to be efficient for smooth globally convex functions [11], but in our problem the simplex shrinks rapidly and gets stuck in a region close to the starting point due to the non-convexity of our objective function. To escape this problem, the threshold accepting algorithm is used in conjunction with the Nelder-Mead simplex method.

The standard threshold accepting (TA) algorithm considers only a single point in the parameter space. A single iteration of the threshold accepting algorithm uses a predetermined number of rounds n_R and explores the local structure of the objective function with a fixed number of steps n_S during each round. A single step consists of shifting the value of a randomly chosen parameter to a neighbouring value (however that may be defined). If the new point obtained represents an improvement in fitness it is accepted as the new current solution. However, even if the new point is slightly worse, it will still be accepted as the new current solution provided the drop in fitness does not exceed a predefined threshold. This improves the ability of the algorithm to escape local minima in the objective function. In early rounds, only a coarse approximation of the fitness is obtained in order to reduce computation time. In our calibration, this is achieved by using a low number of replications in the objective function. The high variance of $\tilde{f}$ due to the low number of replications is compensated by these larger values of the threshold parameter τ_r, which is chosen proportionally to the estimated Monte Carlo variance of $\tilde{f}$. As the algorithm proceeds from one round to the next, τ_r is reduced and I is increased allowing for a more accurate estimation of the fitness [11].

Considering now a combination of the two algorithms, at each iteration either the Nelder-Mead search or Threshold Accepting random shift occurs at random - with the Nelder-Mead search having a higher probability of being chosen. Furthermore, instead of working on a single current solution, the TA algorithm is applied to the entire simplex. At each step of the TA algorithm the same randomly chosen parameter is shifted for each vertex of the simplex, with the magnitude of the random shift being dependent on the mean value of the parameter being shifted. Each vertex is shifted by the same magnitude, except if this results in the vertex taking on an impossible value (e.g. a negative number of agents). In such a case the individual vertex is prevented from moving past the relevant bound. At each step the shift of the simplex is accepted if the best vertex from the new simplex is not worse than the best vertex of the old simplex by more than the relevant threshold [11]. When chosen, an iteration of the Nelder-Mead search behaves exactly as described above. The combined algorithm completes once the stopping criterion is met (in our case after a set number of iterations), with the best performing vertex being returned as the calibrated set of parameter values.

While one iteration of the Nelder-Mead search only requires the calculation of fitnesses for a few vertices, the threshold accepting algorithm is much more computationally expensive, requiring $n_S \times n_R$ fitnesses to be calculated for each iteration. Setting n_R and n_S to be too high significantly slows down the algorithm. We set n_S to 3, with $\tau_R = 3 - 1.5(R-1)$. As we have many freely moving parameters (14 in the standard model and 16 in the adaptive model), we set the number of steps in each round to be relatively high, with $n_S = 7$.

3.2.2 Genetic Algorithm

The genetic algorithm can model learning behaviour in many agent-based financial markets [21]. Genetic algorithms are a family of computational models inspired by evolution [33] - that is, the individuals (parameter values) best suited to the environment (objective function) survive and breed while individuals who are inferior become extinct [7]. More broadly, a genetic algorithm is any population-based model that uses selection and recombination operators to generate new sample points in a search space [33]. The use of the genetic algorithm for our problem is appropriate for a couple reasons. Firstly, it is concerned with non-linear optimisation problems where parameters are not independent and where interactions have an effect on the output of the objective function. Secondly, the algorithm is intuitive, robust and well suited to objective functions with many local minima.

An implementation of a genetic algorithm begins with a population of competing chromosomes (members) [33]. One then evaluates these individual members and allocates reproductive opportunities (the likelihood the member has of being used to form the next population) in such a way that those chromosomes which represent a better solution to the target problem are given more chances to reproduce than those chromosomes which are poorer solutions [33].

The simple genetic algorithm that we use consists of the following steps [13]:

1. Generate an initial population of a specified size.
2. Evaluate each member of the population and assign them a fitness value.
3. Apply selection to the current population based on fitness value to create an intermediate population. Members of the previous population with better fitnesses have greater probability of being sampled, and the sampling is done with replacement.
4. With some large probability, apply Local Arithmetic Crossover[2] to each pair of members of the intermediate population.
5. For each member of the updated intermediate population, perform mutation[3] with some small probability.
6. Resulting from the crossover and mutation, we obtain the new population. We repeat the process from step 2 using this new population unless the stopping criterion (required tolerance) has been met. This required tolerance might for example be a maximum number of iterations of the algorithm, or a target fitness level.

The process of moving from the current population to the next population constitutes one generation in the execution of the algorithm [33]. One problem associated with the genetic algorithm is that it is highly computationally intensive. Selecting reasonable bounds for the parameters is important to restrict the genetic algorithm to exploring the parameter values that could plausible lead to good fitness values.

[2]LAC involves taking two members of the population and shifting each parameter for each member towards the equivalent parameter of the other member by some random amount.

[3]Mutation involves shifting the value a single randomly chosen parameter of a member of the population to a random new value within the predefined bounds of the parameter space.

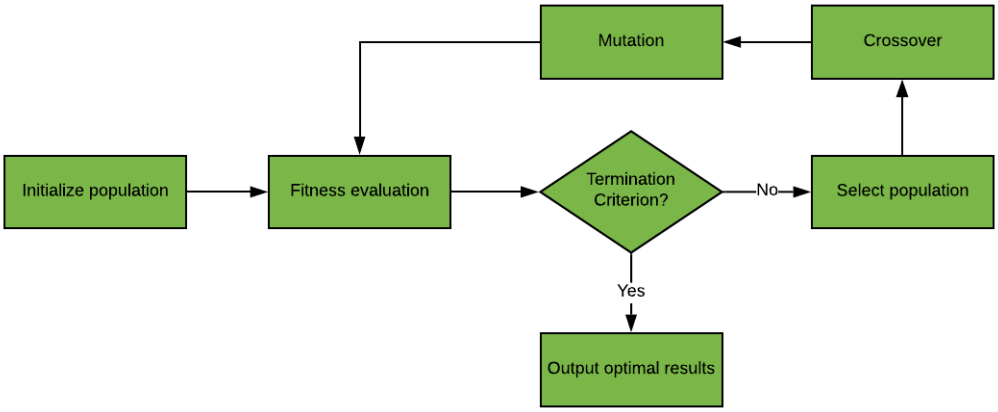

Figure 5: Genetic algorithm flowchart

Since we construct both of the calibration methods to try and maximise the fitness of a given function, we invert the sign of equation (14) so that we turn the minimisation problem into a maximisation problem. It remains that we want our objective function to be as close as possible to zero.

4 Methodology and Implementation

The programming language used to implement this model is `R`[4]. We also make use of multiple packages in `R` for the calculation of each of the moments and statistics, and for the implementation of the genetic algorithm, as shown in the provided `R` code. In terms of hardware requirements, due to the computationally intensive nature of the problem, we make use of RStudio server in conjunction with the Google Cloud Compute Engine for parallel computing[5].

The data used to calibrate the Farmer-Joshi model is Anglo American daily sampled closed price data from 01/01/2005 to 29/04/2016 obtained from the Mendeley website [10]. This results in our simulations being conducted over a period of 2 628 days. The long time-frame used is useful for testing the stability of the model over time. We calibrate the model on an individual stock rather than an index as we argue that in the real world, investors trade individual stocks rather than indices. Simulations only start on the 203rd day (10/20/2005), with the first 202 days of real data being required to initialise the model and provide time-lag data for chartists within the simulation. The data set contains missing values which correspond to weekends and public holidays where the JSE is closed and no trades are made. These days are simply ignored and thus Monday, for example, is considered to be the day after Friday (assuming neither day is a public holiday). When analysing the data, we observed a clear outlier on 12 April 2006, when the share price dropped substantially before immediately jumping back up the following day (as shown in figure 6). We determined the best course of action was to remove this single data point from our data set, rather than using some method to reduce it (such as winsorisation) as the price change was completely abnormal compared to any other price movements.

[4]When running any code involving randomness, we always set the seed for the random number generator to allow for full replication of the results. However, note that we used version 3.6 of **R**, and that versions prior to 3.6 make use of a different random number generator and thus will lead to different results.

[5]Details on how to link RStudio server to Google Cloud's compute engine can be found at https://grantmcdermott.com/2017/05/30/rstudio-server-compute-engine/

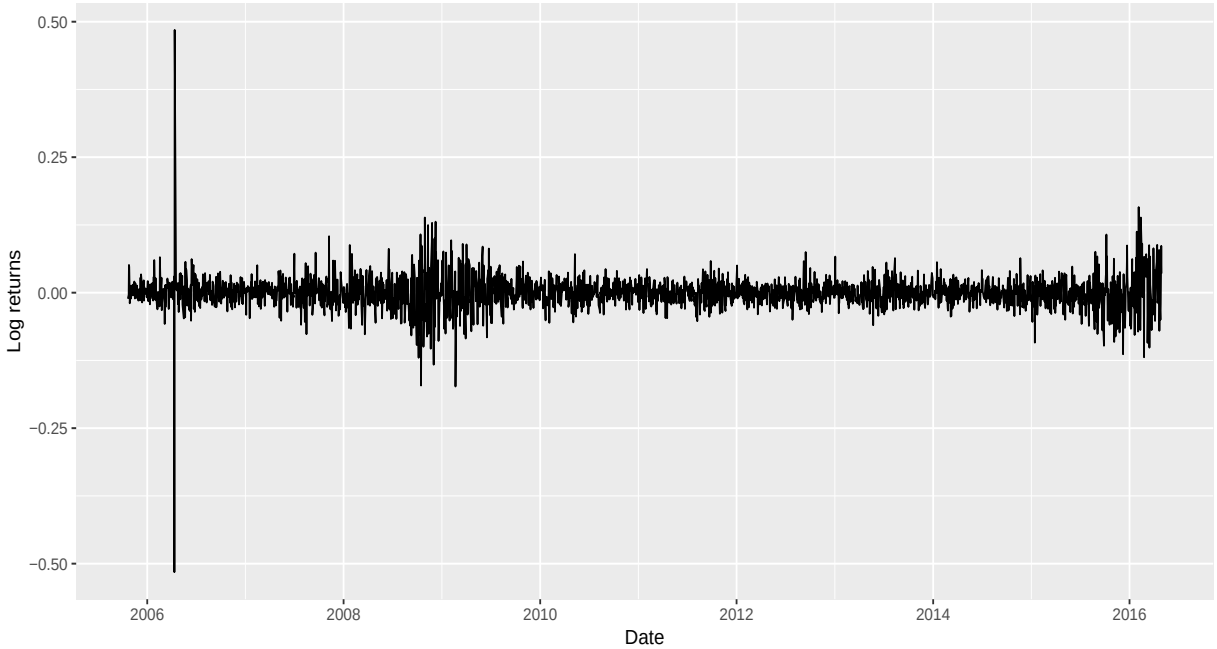

Figure 6: Anglo-American log returns based on daily closing prices

We began by converting the time series to log prices. The simulated data is produced by equation (3). Prior to calibration, we calculated the weight matrix for the objective function using the moving block bootstrap method. The covariance matrix of the empirical moments W^{-1} was obtained using a moving block bootstrap with blocks sizes $b = 100$ and 10000 bootstrap samples. The method we used was as follows:

1. Split the data for the simulation period into $n - 100 + 1$ overlapping blocks of length $b = 100$, with the first block starting at day 1, the second at day 2, etc.
2. Randomly sample blocks with replacement and place them one after each-other until we have a new time series of length 2 628 days - the same length as the simulation period. The last block is truncated to be just 28 days long.
3. Calculate the relevant moments and statistics of this time series.
4. Repeat the previous two steps 10000 times
5. Find the inverse of the covariance matrix of the moments to obtain the weight matrix $\boldsymbol{W}$.

The following genetic algorithm parameters were used to calibrate the models:

- Population size = 60 for standard model; 50 for the adaptive model
- Crossover probability = 0.8
- Mutation probability = 0.1, which reduces linearly to 0.01 starting halfway through the predetermined maximum number of iterations.

For the NMTA algorithm, we made use of the following values for the Nelder-Mead simplex method:

- $\alpha = 1$ (Reflection coefficient)
- $\gamma = 2$ (Expansion coefficient)
- $\rho = 0.5$ (Contraction coefficient)
- $\sigma = 0.5$ (Shrink coefficient)

Table 2 shows the bounds supplied to the genetic algorithm for the parameters that were optimised. The initial parameter values for each of the members/vertices of both the GA and NMTA algorithm calibrations were chosen by random uniform sampling between each of the bounds specified for continuous parameters. Integer parameters were sampled randomly from the integers between the bounds with equal probability. While the members of the GA are always restricted to being within these bounds, the vertices of the NMTA algorithm can shift outside the bounds provided any parameter value obtained is not impossible (such as having a negative time horizon H).

θ	Lower bound	Upper bound
N_f, N_c	40	240
λ	0	15
a	0	1
d_{min}	1	100
d_{max}	d_{min}	$d_{min} + 100$
μ_η	-0.01	0.01
σ_η	0	0.05
σ_ζ	0	0.05
T_{min}	0	1
T_{max}	T_{min}	$T_{min} + 1$
τ_{min}	-1	0
τ_{max}	τ_{min}	$\tau_{min} + 1$
v_{min}	-0.5	0
v_{max}	v_{min}	$v_{min} + 1$
H	1	100
Γ	0	1

Table 2: Parameter bounds set for the GA search and NMTA initialisation.

As shown in table 2, we allowed close to all of the parameters of the model to move freely due to our uncertainty about what the optimal values of any of the parameters may be. One restriction is that we set the number of fundamentalists to be equal to the number of chartists for the standard model. From experiments of allowing different constant ratios of chartists to fundamentalists, we did not find any improvement in the results of the model.

With regards to the initial positions of the agents at the start of each simulation, it is important that one does not assume that none of them have a position going into day one. Otherwise, we might observe an unrealistic amount of trading on day one of the simulation as agents enter positions they should have already been in, which would strongly affect the simulated closing price. For example, if there is a strong positive trend in the share price leading into the start of the simulation period, one would expect many of the chartists to already be in a position, and it is important that our model accounts for this. For this reason, we set the initial agent positions one day prior to the start of the simulation (which we call time $t = 0$), based on the closing prices observed from the data in the prior days ($t \leq -1$). When we start the simulation, we still use the observed closing price from the data at time $t = 0$. This way, initialising agent positions to positions they should already be in does not affect the price within the simulation.

We set the initial value v_{-1} to be the observed closing price two days prior to the start of the simulation. Each fundamentalist then determines their own initial value v^i_{-1} by offsetting v_{-1} by their individual value offset, and uses this to take their initial positions the day before the start of the simulation ($t = 0$). The

fundamentalists' perceived values are then updated at $t = 0$ according to equation (7), and this is the perceived value they hold going into day one of the simulation period.

It should be noted that simulations are path dependent and that there is randomness in the search routine of the genetic algorithm. For this reason, to obtain confidence intervals for parameters, we considered two methods. One method is to consider the distributions of calibrated parameters across different realizations of a price path. Since we only ever have one price path realization, we apply the moving block bootstrap method to the log returns to obtain bootstrapped price path samples[6]. We would thus calibrate the model on each of these bootstrapped samples to obtain many sets of parameters upon which confidence intervals can be calculated. This method, however, produced parameters which varied greatly from another and showed no convergence to a set of globally optimal parameters. This may be indicative of parameter degeneracies that exist in the model, but may also be due to the nature of the block bootstrapped price paths differing too substantially in behaviour between each-other.

We therefore resort to using an alternative method. We run the GA and NMTA algorithm multiple times on a single price path (the actual data). This gives an indication of the variability in the optimal parameters that is due to the instability in the objective function as well as the randomness between each independent run of the search routine. The bounds for the confidence intervals of the ith parameter are calculated as $\overline{\theta}_i \pm t^{\alpha} \frac{\sigma(\theta_i)}{\sqrt{n}}$ where $\overline{\theta}_i$ is the sample mean for parameter i, $\sigma(\theta_i)$ is the standard deviation over the sample, n is the sample size, and t^{α} is the $\alpha = 2.5\%$ critical value for the t-distribution.

To obtain confidence intervals for moments and statistics we simply ran 1000 simulations using the parameters that resulted in the lowest objective function value for each of the two calibration methods and took the 2.5 percentile and the 97.5 percentile as the lower and upper bounds respectively. In this case the variation in the moments is solely dependent on the random price paths that are simulated.

4.1 Computational Efficiency

Simulation and calibration of the Farmer-Joshi model requires a substantial amount of computation, so it is useful to test which parts of our implementation take the most time to be computed. Using the `microbenchmark` package in `R`, we compare the computation time for each of the functions used in our objective function when considering a 2628 day simulation. Even though the absolute times measured depend mostly on the computer being used, the relative times are mainly what we are interested. These tests were conducted at the same time with as few background processes as possible running on the computer in order to standardise the conditions. Some functions can handle multiple Farmer-Joshi simulations at once, so we calculate their computation time on our chosen number of 50 replications, and then divide this by 50 to get their effective time for a single replication so that they can be compared to the remaining functions that scale proportionally to the number of replications.

[6] Since price is not ergodic, we apply the moving block bootstrap on log returns and then convert back to price by applying a cumulative sum with a random initial price

Function	Computation time for 1 replication	
Simulation (standard model)	125.709	(Effective time when $I = 50$)
Simulation (adaptive model)	3535.000	
Mean	0.003	(Effective time when $I = 50$)
Standard Deviation	0.097	(Effective time when $I = 50$)
Kurtosis	0.181	
KS-test	2.441	
Hurst Exponent	35.882	
GPH	53.400	
ADF	2.335	
GARCH	4.834	
Hill	355.546	

Table 3: Computation times (in milliseconds) for the components of the objective function

From table 3 we can easily compare the difference in simulation time for the standard Farmer-Joshi model versus the adaptive Farmer-Joshi model. For the standard Farmer-Joshi model, our code is able to run multiple simulations at once (provided each simulation uses the same set of parameters) by making use of matrices to speed up computation. Here a matrix of price paths is returned where each column refers to the price path of a different simulation. This allows us to simulate all of the replications required for the objective function at once - significantly reducing the effect that adding more replications has on run-time. The implementation of the adaptive Farmer-Joshi model, however, is unable to conduct multiple simulations at once due to the model's added complexity. This means that - unlike for the standard model - run-time scales proportionally to the number of replications. This is why we find there to be a significantly longer computation time required to calculate the objective function value for the adaptive model compared to the standard model. To mitigate the effect of extra computation time, we lower the number of of replications from 50 to 30 for the adaptive model.

Table 3 also shows that the calculation of the Hill estimate takes a significantly long time. Finding another robust but faster way to test for fat tails would help to drastically improve computation time when calibrating the standard Farmer-Joshi model, as the calculation of the Hill estimator currently takes up a clear majority of the computation time.

4.2 Adaptive Agents Switching Mechanics

With the addition of allowing agents to switch strategies in the adaptive Farmer-Joshi model comes an added complexity with regards to the details of how exactly agents transition from one strategy to another. One method would be to force agents to always liquidate their position they held under the previous strategy, and then from there to decide whether or not to take a position under the new strategy (by checking whether or not the mispricing under the new strategy meets one of their entry thresholds). This is effectively equivalent to agents dying out and new agents entering. This method, however, slightly contradicts the purpose of the entry and exit thresholds, which is to reduce transaction costs. Therefore, we instead implement a system where an agent who is currently holding a position and is switching strategies will still hold their current position if the relevant exit thresholds under the new strategy is not met. The details of this behaviour for the ith agent are illustrated by the made-up example in table 4. The values in the last three columns (the positions) are determined by the values observed in the prior columns, as described below the table.

			$m_t^{f,i}$	$m_t^{c,i}$		Position ($x_t^{s,i}$)		
Day	T^i	τ^i	$p_t - v_t^i$	$p_{t-d^i} - p_t$	Strategy (s)	Fundamentalist	Chartist	Actual
1	1	-0.5	0.1	-0.2	Chartist	0	0	0
2	1	-0.5	0.4	-1.2	Chartist	0	c^i	c^i
3	1	-0.5	0.8	-0.2	Fundamentalist	0	c^i	0
4	1	-0.5	1.1	-0.1	Fundamentalist	$-c^i$	c^i	$-c^i$
5	1	-0.5	-0.6	-0.3	Chartist	0	c^i	$-c^i$
6	1	-0.5	-0.1	-0.6	Chartist	0	c^i	0

Table 4: Agent switching behaviour visualisation

1. Initially, the agent chooses a chartist strategy. On day 1, the entry thresholds of $T^i = 1$ for taking a short position and $-T^i = -1$ for taking a long position are both not met for each strategy. So, no position is taken by the agent.
2. On day 2, there is a strong enough trend for the agent to take a long position of c^i under the chartist strategy ($m_2^{c,i} < -T^i$).
3. On day 3, the agent decides to switch strategies to a fundamentalist strategy. Because $p_3 - v_3^i > -\tau^i$, the agent exits the position they took under the chartist strategy.
4. On day 4, the agent - still using their fundamentalist strategy - enters a short position as $p_4 - v_4^i > T^i$.
5. The agent switches back to a chartist strategy on day 5, but because $p_{5-d^i} - p_5 \not< \tau^i$, the agent still holds onto their previous position of $-c^i$. This is despite the theoretical positions for each strategy had the agent not adapted between strategies both being different to the actual position taken.
6. Lastly, on day 6, the agent's exit threshold is met under the chartist strategy, as $p_{6-d^i} - p_6 < \tau^i$, so they exit their position. However, the theoretical position of the agent had they always remained a chartist stays as c^i as $p_{6-d^i} - p_6 \not> -\tau^i$, so the exit threshold is not met.

Figure 7 shows an overview of our implementation of the adaptive Farmer-Joshi model and the method used for calibrating the model.

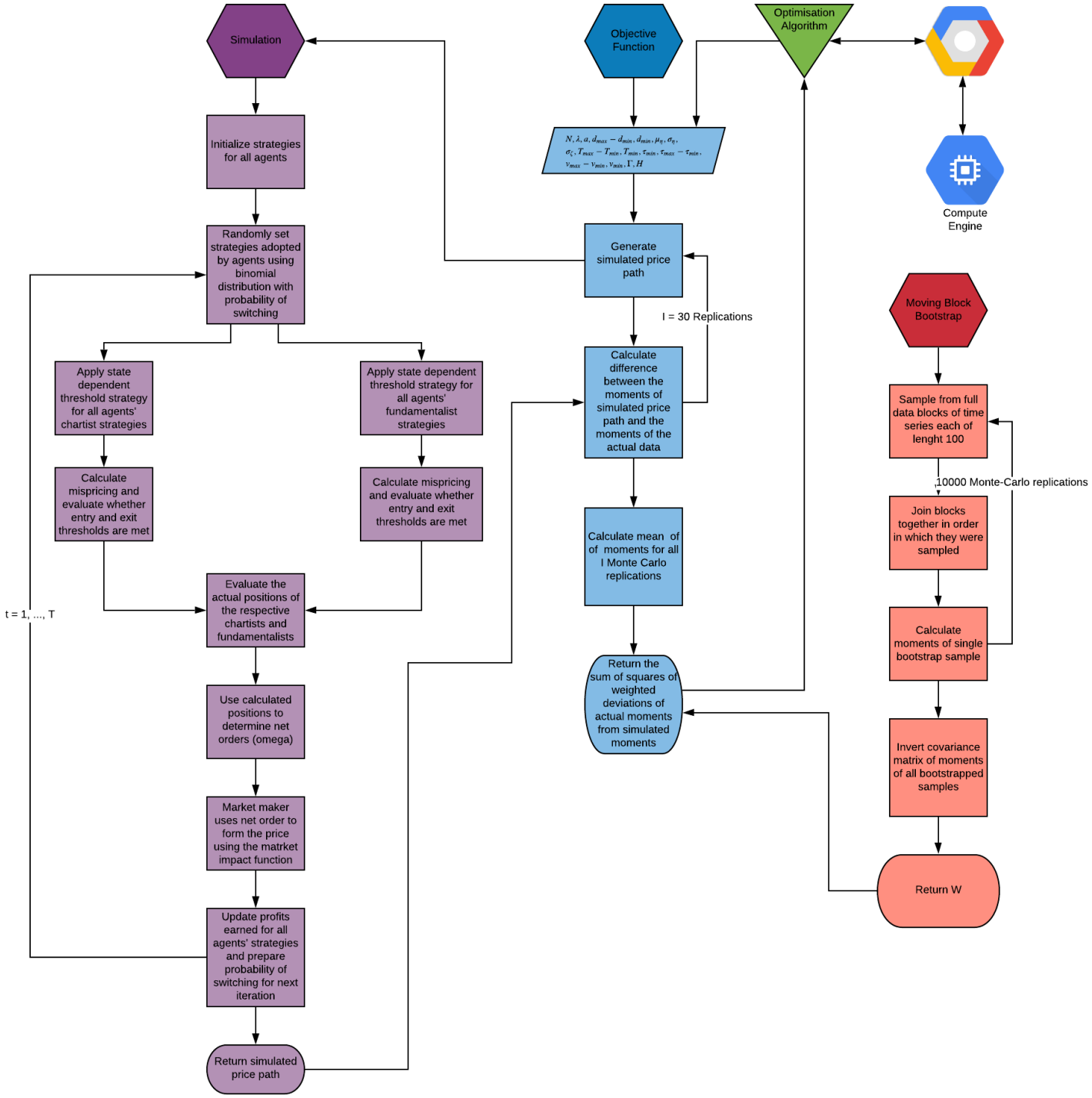

Figure 7: Flowchart visualisation of the adaptive Farmer-Joshi model

4.3 Testing and Replication of Prior Literature

We test if our implementation is harmonious with what has been done in the past by replicating as closely as possible a prior calibration attempt of the Farmer-Joshi model by Fabretti [7]. The paper calibrates the model using S&P 500 closing price data over a two year period from 2005 to 2007[7] [7]. By comparing our results to the results obtained in this prior calibration, we can get a good indication of if there are any differences in between our implementation of the Farmer-Joshi model and the implementation by Fabretti. One challenge that was encountered along the way, however, was that the exact implementation of the Farmer-Joshi model is not easily identifiable. In other words, our implementation of the Farmer-Joshi model may differ from that of the original paper and the version implemented by Fabretti. For this reason, this section is devoted to a comparison of simulation results in an attempt to identify differences in model implementations.

For this experiment, we only compare the moments and statistics shown in the paper, which are the following [7]:

1. Mean of logarithmic price
2. Standard deviation of logarithmic price
3. Kurtosis of logarithmic price
4. Kolmogorov-Smirnov statistic on logarithmic price
5. Hurst exponent on logarithmic returns

There are multiple different methods used to calculate the Hurst exponent that can result in slightly different values being obtained. Fabretti makes use the generalised Hurst exponent [7]. In our implementation, we instead use the simplified Hurst exponent as described in section 3.1.

For this test we make use of slightly different position equations to those described in sections 2.2 and 2.3. These different equations are based off of the equations used by Fabretti [7]. The position equation for the ith chartist is given as follows:

$$x^i_{t+1} = c^i(p_t - p_{t-d_i}) \tag{19}$$

with the position equation for the ith fundamentalist given as:

$$x^i_{t+1} = c^i(v^i_t - p_t) \tag{20}$$

This means that traders no longer only have three possible positions that they can be in at any time, as their position also depends on the exact mispricing at the time that the agent's entry threshold is met. This change does not affect the regularity with which trading takes place. Agents still only enter a position when an entry threshold is met, and they do not change that position until the relevant exit threshold is met, at which point they completely liquidate their position. We do not use these position equations in our calibration attempts later on, as we believe the position equations described in sections 2.2 and 2.3 more closely follow the description of the method for using entering and exiting thresholds as described in the original paper by Farmer and Joshi.

Fabretti does not describe how initialisation of the model is conducted with regards to determining the initial price trend. We use price data prior to the start of the simulation in order to initialise the trend for different traders.

[7] We obtain this data from *Yahoo Finance*

We conduct 1000 simulations using the best parameter values obtained by Fabretti (which were obtained from the genetic algorithm optimisation) and then calculate the moments and statistics for each simulation and construct 95% Monte-Carlo confidence intervals. If our implementation is the same as Fabretti's, we should expect to see these confidence intervals align very closely with those obtained by Fabretti. This comparison is shown in table 5.

Moments and statistics	$CI^{95\%}$	$CI^{95\%}_{Fabretti}$
Mean	[6.9333; 7.20719]	[6.97420; 7.31320]
Standard deviation	[0.01975; 0.09023]	[0.02030; 0.09010]
Kurtosis	[-1.51839; 0.90736]	[1.4970; 3.8355]
Kolmogorov-Smirnov statistic	[0.22659; 1]	[0.2034; 0.9920]
Hurst exponent	[0.45512; 0.59557]	[0.4129; 0.5583]

Table 5: Moments and statistics on simulated data using parameters obtained by Fabretti[7]

Noticeably, the upper bound for mean of logarithmic price that our implementation obtains is significantly lower than the upper bound obtained by Fabretti. However, the values for remaining moments very closely resemble each other, giving little indication of significant differences between the models. The difference in the mean of log price could easily be due to a difference in how we initialise the trend and initial value perceptions for the traders, and does not necessarily indicate any differences between the models themselves.

Choosing one simulation on the parameters at random, we plot figures 8 through 12. In figure 8, we observe that our simulated price path stays well within the confidence intervals for simulated price paths obtained by Fabretti [7]. We also observe an absence of fat tails in figure 10 and an absence of volatility clustering in figure 12, as was observed by Fabretti [7]. Again, these results do not show any clear differences between the two models.

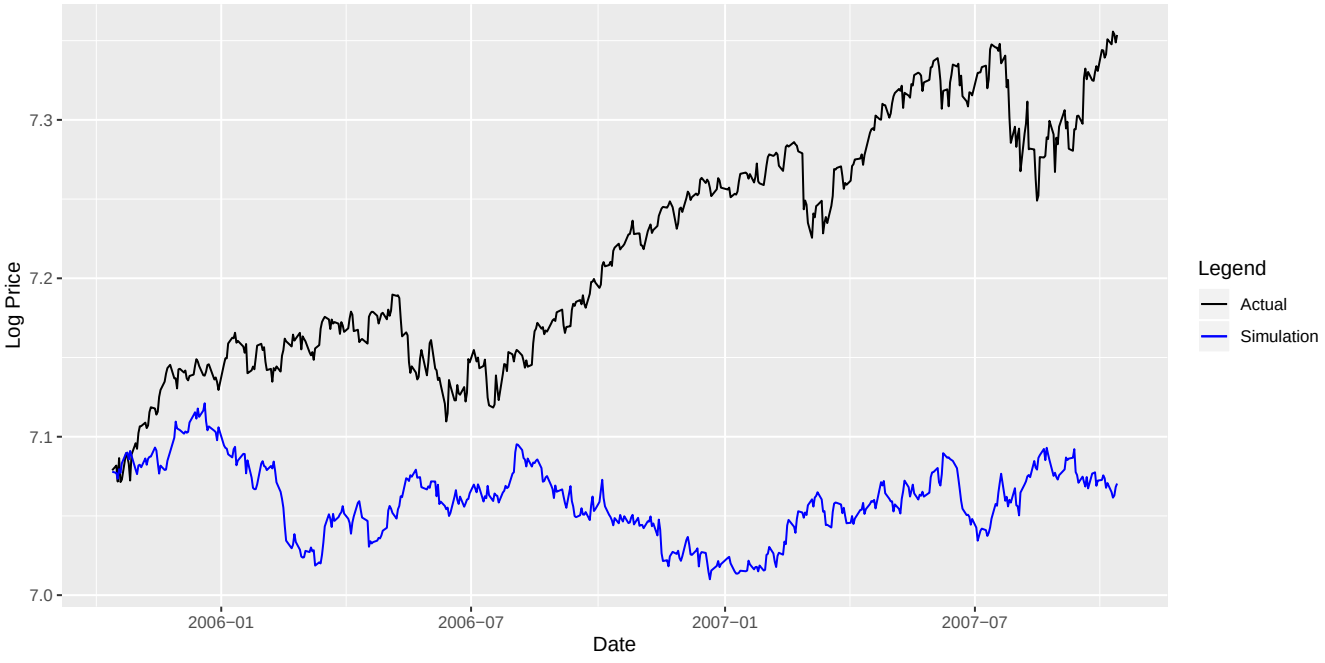

Figure 8: Observed and simulated closing log price paths (for S&P 500 index)

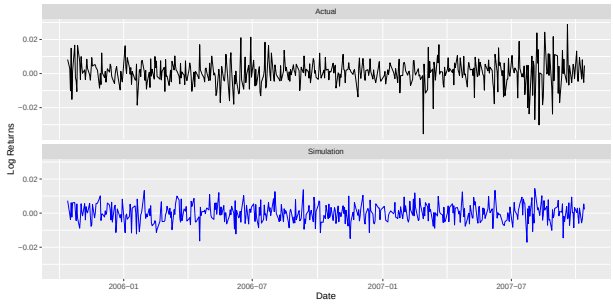

Figure 9: Log returns paths

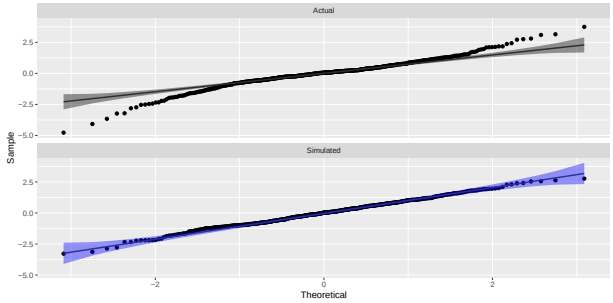

Figure 10: Normal probability plots

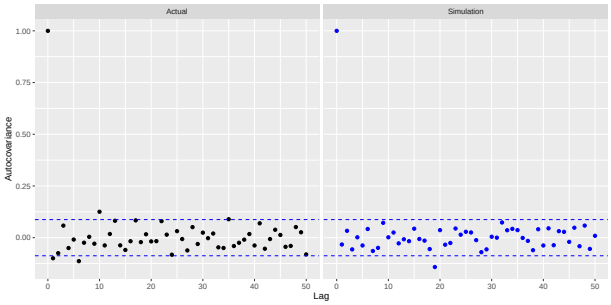

Figure 11: Autocorrelation of log returns

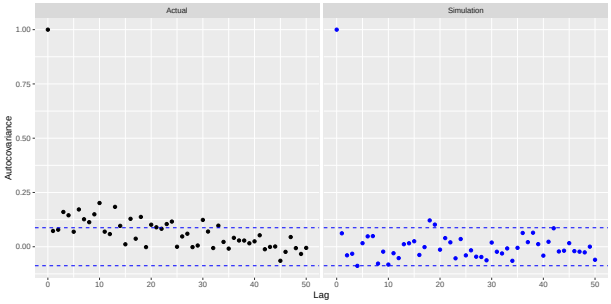

Figure 12: Autocorrelation of absolute log returns

5 Results and Analysis

In what follows, we present the results of our calibration experiments. We do this by demonstrating whether or not our implementations of both models are capable of reproducing a number of well-known stylized facts of log return time series which are currently used to validate agent based models of financial markets. One of the most well-known stylized facts is the empirically observed fact that the distribution of log returns are leptokurtic and fat tailed with a greater probability of extreme negative events than extreme positive events (contrary to traditional branches of financial mathematics). Furthermore absolute log returns exhibit significant short-term autocorrelations which slowly decay as the time lag increases. Although these are the most well-known stylized facts, they are by no means exhaustive. We also further validate the models by considering the range of parameters that result in similar behaviour. A large range indicates that changing the parameter has an unclear effect on the resulting behaviour of the model, suggesting that changing it may just be adding noise to the model.

We calculate the weight matrix by taking the inverse of the covariance matrix for the bootstrapped moments. As a measure of accuracy, we obtain the condition number by finding $||A||_2 \cdot ||A^{-1}||_2$ where A is the covariance matrix for the bootstrapped moments and the subscript 2 refers to the Euclidean norm. The weight matrix $\boldsymbol{W}$ (rounded to 3 decimal places) is estimated to be

$$\begin{bmatrix}
5416161.546 & 363543.535 & 61.278 & 50983.853 & 21613.255 & -1616.466 & -953.207 & 8257.729 & 18342.253 \\
363543.535 & 373821.757 & 214.752 & -2540.696 & 462.448 & -1095.82 & -156.572 & -9152.185 & -15887.062 \\
61.278 & 214.752 & 2.578 & 40.399 & 5.411 & -5.213 & 0.269 & 10.647 & -39.162 \\
50983.853 & -2540.696 & 40.399 & 13909.613 & 379.126 & 10.006 & 10.082 & 386.06 & -64.635 \\
21613.255 & 462.448 & 5.411 & 379.126 & 1404.067 & -9.936 & -3.353 & -15.575 & 68.938 \\
-1616.466 & -1095.82 & -5.213 & 10.006 & -9.936 & 105.597 & 0.401 & -152.06 & -8.612 \\
-953.207 & -156.572 & 0.269 & 10.082 & -3.353 & 0.401 & 1.778 & -2 & -18.912 \\
8257.729 & -9152.185 & 10.647 & 386.06 & -15.575 & -152.06 & -2 & 6646.569 & -44.464 \\
18342.253 & -15887.062 & -39.162 & -64.635 & 68.938 & -8.612 & -18.912 & -44.464 & 2144.721
\end{bmatrix}$$

Condition number = 7196302039

The order of the moments is as stated in section 3.1. This means, for example, that the value in the first row and second column refers to the covariance between mean of log returns and standard deviation of log returns. With this matrix, we can get a very broad indication of which moments and statistics are most important for the objective function by looking at the diagonal elements (the variance of each moment/statistic). A brief inspection of the weight matrix indicates that the mean and standard deviation of returns both seem to be weighted very highly, while the kurtosis and the ADF statistic are of much less importance to the objective function due to their high estimated variance. This implies that these statistics could deviate substantially between the simulations and the actual data and they would still not have a large effect on the fitness value obtained from the objective function. The condition number checks how sensitive the weights are to small changes in the input. This gives an indication of how well conditioned our matrix is and how accurately we are able to invert the covariance matrix to get the weight matrix. The condition number we obtain is on the order of 10^9, which implies we lose about 9 digits of accuracy when inverting the matrix. Given that the software we use stores numbers to 16 decimal places, we consider the matrix to be sufficiently well conditioned for our purposes. As a further test of accuracy, we multiply the weight matrix by the covariance matrix and find that the resulting matrix is equivalent to the identity matrix to 11 decimal places.

5.1 Standard Farmer-Joshi Model

We used both the genetic algorithm (GA) and the Nelder-Mead with Threshold Accepting (NMTA) algorithm to calibrate the standard model. For each calibration using the GA, 10 runs of the genetic algorithm (using different initialisations) were conducted with a minimum of 55 iterations each, with further iterations if necessary up until each calibration achieved an objective function value above -40. The choice of -40 as the cutoff was chosen to be slightly below the best (convergent) results that we had observed for this model when running the calibration for a much larger number of iterations. The cutoff ensures that each calibration results in very similar objective function values - allowing each calibration to be classified as being fairly similarly successful, thereby allowing us to construct confidence intervals for the parameters. For the NMTA algorithm, we conducted 15 separate calibrations where the number of iterations was set to 65, but discarded the 5 calibrations that resulted in the worst fitness values.

For the GA calibrations, this resulted in objective function values ranging from -31 to -39. Some calibrations reached convergence very quickly, while others took over 150 iterations. This implies that the initial parameters selected by the genetic algorithm play a very important role in how successful it is at finding a good solution and not getting stuck in poor local optima. This is despite our method involving periodically resetting the genetic algorithm's population while retaining only the best parameter set. We obtained a slightly greater range of fitnesses from our calibrations using the NMTA algorithm. While the algorithm sometimes resulted in much faster convergence to very good fitness values compared to the GA, certain calibrations got stuck at quite poor fitness values, even when run for many more than 65 iterations. This suggests that the random shift (threshold accepting) portion of the algorithm is occasionally unable to lead the algorithm to escaping sub-par local optima. This being said, by far the greatest improvements in the fitness for each calibration were made when the algorithm chose to use the random shift with threshold accepting method as opposed to the Nelder-Mead method.

As was stated in section 4, our calculation of the parameter confidence intervals is done differently to some other papers in the literature. Many other papers use a method whereby different block bootstrap samples of the share price data are used for each of the calibration runs [27, 28, 29, 30]. This method can be seen as obtaining confidence intervals for the parameters for more general share behaviour by bootstrapping the single observed price path to create new price paths that each have slightly different behaviour to one another. We instead conduct all of our calibrations based off of the same price path - the actual price path observed for Anglo-American stock. As was noted by Fabretti, different calibrations can reach different points differing in values, but with indistinguishable differences in the simulated data [7]. By using the same price path in our method, we instead obtain confidence intervals for which parameter

values best explain the price behaviour observed over the specific period that was measured. Given that we use heuristic optimisation to estimate the parameters due to the stochastic nature of the model, one cannot definitively state that one set of parameters is better than another when their fitnesses are very similar to one another. By obtaining many different parameter values that obtain similar fitnesses, we are able to better define the region of parameter values that can explain the observed data similarly well - an area of study that has in the past been suggested for future research [7].

Table 6 shows the results of our calibration experiments using both optimisation algorithms. The point estimates are the estimates of the parameters obtained by the calibration that resulted in the best fitness for each optimisation algorithm.

θ	θ_{GA}	$\theta_{GA}^{95\%}$	θ_{NMTA}	$\theta_{NMTA}^{95\%}$
N_f, N_c	84	[80;120]	61	[48;203]
λ	12.59549	[9.06162;11.44419]	10.26531	[7.42702;13.40640]
a	0.45737	[0.25131;0.37690]	0.55884	[0.16518;0.68659]
d_{max}	49	[57;80]	64	[44;92]
d_{min}	21	[26;46]	18	[11;39]
μ_η	0.00853	[-0.00501;0.00361]	0.00561	[-0.00496;0.00319]
σ_η	0.04200	[0.02619;0.03708]	0.02176	[0.02705;0.05711]
σ_ζ	0.02032	[0.01896;0.02009]	0.01930	[0.01738;0.02031]
T_{max}	0.96590	[1.00311;1.19918]	1.47698	[1.27609;1.78699]
T_{min}	0.40217	[0.50307;0.67188]	0.50747	[0.46721;0.64138]
τ_{min}	-0.16188	[-0.3251;-0.19622]	-0.13981	[-0.51354;-0.15336]
τ_{max}	0.63070	[0.3515;0.55426]	0.60403	[0.15530;0.76696]
υ_{max}	0.25206	[-0.01738;0.31747]	0.13012	[-0.02491;0.52813]
υ_{min}	-0.27687	[-0.32853;-0.24581]	-0.21514	[-0.32808;-0.08448]
Fitness	-30.54963	[-37.56189;-33.08459]	-22.64955	[-42.19412;-26.26681]

Table 6: Estimates of parameters using GA and NMTA algorithms.

These confidence intervals indicate that many of the parameters can vary substantially from one calibration to another, even with the objective function values obtained being very similar. This indicates the existence of many local optima, each with a similar ability to replicate the moments and statistics found in the actual data. These large confidence intervals unfortunately limit the explanatory power of the model. We do find certain parameters such as μ_η, σ_η and σ_ζ to have narrow confidence intervals, allowing for more precise inference about their values regarding their relevance in the real world. However, inferences about real world behaviour from these parameter estimates should be made with caution, as our implementation of the model is unable to capture some of the most prevalent features of financial markets, as will be shown in section 5.1.1.

In table 7, we observe that our optimal parameters from both calibration methods struggle in particular to replicate the GPH estimator, as the GPH estimator based on the empirical data (in column m^e) falls outside of the confidence intervals obtained from the simulations. For many of the moments, we also observe very wide confidence intervals, indicating a lack of consistency across different simulations. The point values for each of the moments are the moments of a single randomly chosen simulation from each set of parameters for which further analysis is done in the section that follows.

Moments and statistics	$m^s\|\theta_{GA}$	$\text{CI}_{GA}^{95\%}$	$m^s\|\theta_{NMTA}$	$\text{CI}_{NMTA}^{95\%}$	m^e
Mean	0.00025	[-0.00040;0.00150]	0.00112	[-0.00021;0.00133]	-0.00005
Standard deviation	0.02052	[0.0.02096;0.14332]	0.02054	[0.01947;0.13265]	0.02767
Excess Kurtosis	0.26094	[0.08238;14.06729]	1.07280	[0.00489;5.33540]	4.03912
Kolmogorov-Smirnov statistic	0.03997	[0.02589;0.18995]	0.04263	[0.03388;0.16749]	0
Hurst exponent	0.58085	[0.42944;0.59136]	0.58833	[0.44770;0.60538]	0.54487
GPH estimator	0.25637	[0.07374;0.72297]	0.19139	[-0.03422;0.62494]	0.73580
ADF statistic	-0.53381	[-53.73332;-13.33092]	-52.55660	[-53.53643;-16.19956]	-49.42858
GARCH parameters	0.98393	[0.02273;1.31436]	0.33447	[0.00555;1.26677]	0.99339
Hill estimator	0.26949	[0.24877;0.77339]	0.29442	[0.24374;0.53700]	0.42845

Table 7: Moments and statistics on actual data and simulated data using the GA and NMTA optimisation methods.

5.1.1 Comparison of Key Features of Observed and Simulated Data

Figures 13 to 17 are created from the observed log price data, and simulations using both the best performing parameter values from the GA calibrations and best performing parameter values from the NMTA algorithm calibrations. In figure 13 we show the log price paths of a single simulation from each set of parameters, with the shaded areas representing the 95% confidence intervals for each of the simulations[8]. We see that the simulation paths do not closely follow the actual price path, as is to be expected given that the goal of the model is to replicating the overall behaviour of log returns rather than fitting a specific price path. Both sets of confidence intervals almost entirely contain the actual price path. The confidence interval for the parameters obtained from the NMTA algorithm is particularly wide, indicating large discrepancies between simulations, while the confidence interval for the parameters from the GA widens very quickly early in the simulation before stabilising.

Figure 14 shows the comparison between actual logarithmic returns and the simulated logarithmic returns from the individual simulations shown in figure 13. The variance of returns for the simulations are much more consistent through time compared to the observed data, with no clear evidence of volatility clustering in both simulations. In figure 15 we compare the distribution of returns for the actual data and the simulations. We find minimal fat tails in the simulations unlike in the actual data where there are clear fat tails on both ends of the distribution. The simulation using the optimal parameters obtained from the NMTA algorithm does show a small amount of fat tails on the right side of the distribution, yet lacks any fat tails on the left-hand side. This goes against the stylized fact of gain/loss asymmetry (which states that large price drops are more frequent than large price increases), which the actual data does exhibit to a small extent. While the simulations manage to replicate the lack of autocorrelations of log returns that is found in the data (figure 16), they fail to produce anywhere near the same level of autocorrelations of absolute log returns as the actual data as shown in figure 17.

[8]The confidence intervals for the price paths were obtained from Monte-Carlo methods by creating 1000 simulations for each set of parameter values and determining the bounds at each point in time that contain the middle 95% of simulations

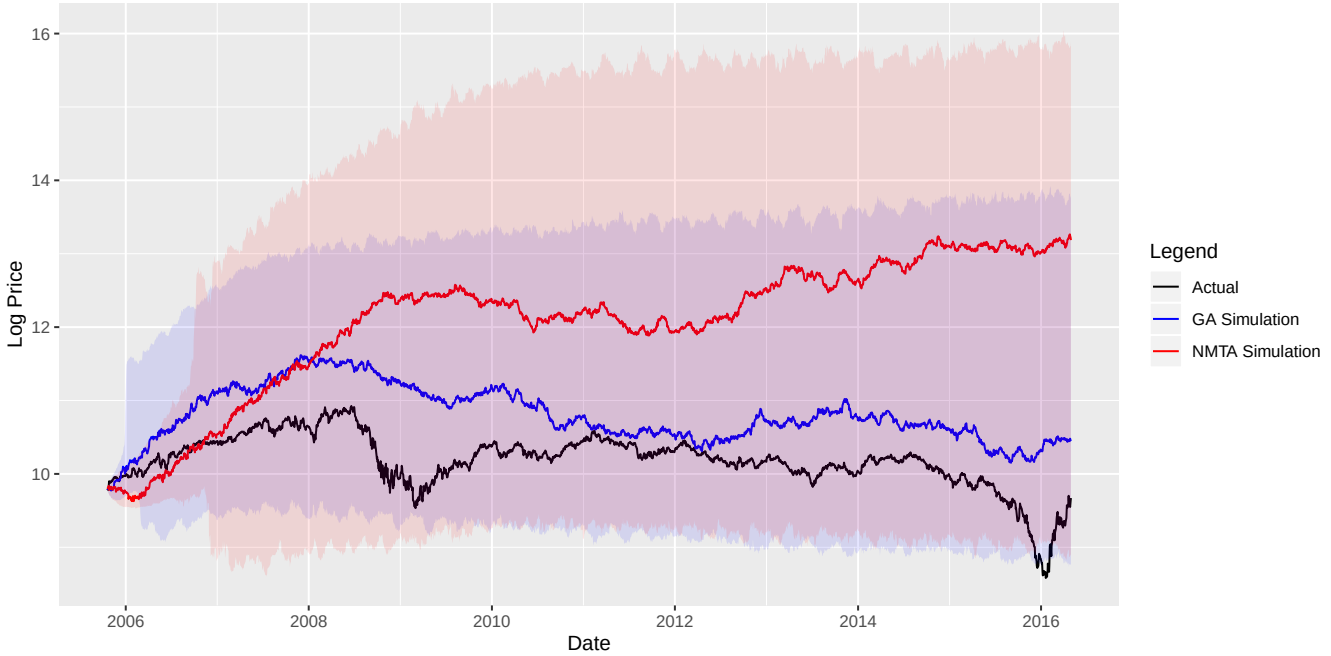

Figure 13: Observed and simulated closing log price paths (for Anglo-American shares)

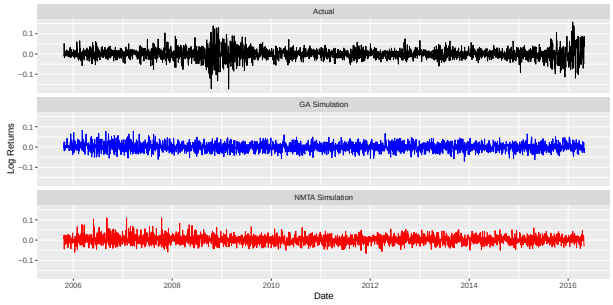

Figure 14: Log return paths

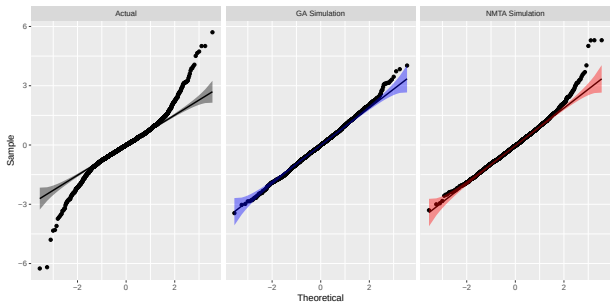

Figure 15: Normal Probability Plots

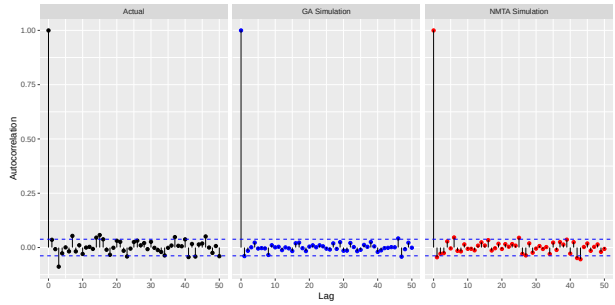

Figure 16: Autocorrelation of log returns

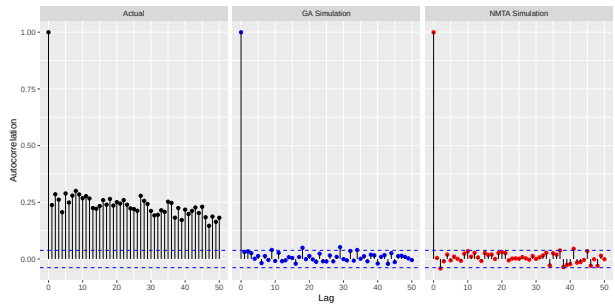

Figure 17: Autocorrelation of absolute log returns

Overall, similar to the results of a previous calibration attempt [7], we are unable to replicate some important stylized facts to a reasonable extent when using the standard Farmer-Joshi model. In the next section, we test the performance of the adaptive Farmer-Joshi model using the same data set.

5.2 Adaptive Farmer-Joshi Model

Table 8 below presents the optimal values and confidence intervals for the parameters as a result of the application of the NMTA and GA algorithms respectively. Confidence intervals are obtained from $n = 11$ optimal sets of parameters for both optimisation algorithms. Similar to the standard model, there appears to be significant differences in the optimal parameters between the two calibrations, indicating that a large range of parameter values are able to result in similar performance. These optimal points are not unique, rather they represent a region of maxima where slight changes in the parameters do not correspond to large differences in the performance of the simulated data (based on the objective function). It is important to note that the value for Γ in the GA calibration is larger than that of the NMTA calibration. Together with this and the fact that the NMTA calibration was less able to replicate the stylized facts (section 5.2.2) gives reason to believe that a larger value for Γ results in a greater degree of switching and therefore a greater allowance for periods where chartist activity dominates. This point was also noted by Brock and Hommes [3] and is made clearer in section 5.2.3.

θ	θ_{GA}	$\theta_{GA}^{95\%}$	θ_{NMTA}	$\theta_{NMTA}^{95\%}$
N	90	[80;121]	81	[101;182]
λ	10.27527	[10.56002;12.62532]	3.26799	[5.47222;12.08974]
a	0.08186	[0.07760;0.11010]	0.01525	[0.01849;0.08230]
d_{max}	87	[88;113]	105	[86;129]
d_{min}	34	[37;53]	90	[30;67]
μ_η	-0.00255	[-0.00211;0.00299]	-0.00596	[-0.00402;0.00282]
σ_η	0.02767	[0.01712;0.03046]	0.04145	[0.01212;0.03439]
σ_ζ	0.01721	[0.01769;0.02103]	0.01897	[0.01399;0.01775]
T_{max}	0.67289	[0.60669;0.98115]	1.414	[0.82567;1.49038]
T_{min}	0.35379	[0.19958;0.47418]	0.60891	[0.28719;0.74903]
τ_{min}	-0.39869	[-0.52035;-0.39311]	-0.52416	[-0.4843;-0.29535]
τ_{max}	0.15383	[0.0979;0.32634]	-0.13256	[-0.00543;0.12641]
v_{max}	0.43789	[0.06897;0.34144]	-0.07734	[0.09896;0.43791]
v_{min}	-0.094	[-0.31161;-0.16015]	-0.38161	[-0.25692;-0.06936]
Γ	0.57012	[0.38411;0.59631]	0.12921	[0.18155;0.53080]
H	45	[43;57]	54	[26;72]
Fitness	-20.64028	[-65.01577;-19.19762]	-16.06043	[-25.41958;-16.87596]

Table 8: Estimates of parameters using GA and NMTA algorithms.

Table 9 below provides confidence intervals for the moments and statistics of simulated data for the GA and NMTA algorithm respectively. Additionally, the estimated moments and statistics from the actual data are included as well (m^e). The moments of simulated log returns from both the GA and NMTA calibrations closely match each other as well as that of the actual returns (with the GA calibration having a slightly closer resemblance) with the exception of excess kurtosis. The GA calibration results in greater excess kurtosis than the NMTA calibration. Hence, for this calibration, one should expect to find that log returns would be more leptokurtic and fat tailed than the NMTA calibration. Lastly, the confidence intervals seem to encapsulate most of the empirical moment values.

Moments and statistics	$m^s\|\theta_{GA}$	$\text{CI}_{GA}^{95\%}$	$m^s\|\theta_{NMTA}$	$\text{CI}_{NMTA}^{95\%}$	m^e
Mean	-0.00003	[-0.00082;0.00055]	-0.00011	[-0.00096;0.00054]	-0.00005
Standard deviation	0.02594	[0.02102;0.02741]	0.02389	[0.02035;0.02589]	0.02767
Excess Kurtosis	6.79172	[0.46553;5.62495]	2.67784	[0.45883;2.88263]	4.03912
Kolmogorov-Smirnov statistic	0.01675	[0.01637;0.04111]	0.02322	[0.01637;0.04492]	0
Hurst exponent	0.54975	[0.47187;0.56053]	0.55758	[0.48541;0.57422]	0.54487
GPH estimator	0.35897	[0.14932;0.60416]	0.43850	[0.10793;0.60692]	0.73580
ADF statistic	-55.72727	[-55.40820;-48.46334]	-51.56574	[-55.05093;-47.79636]	-49.42858
GARCH parameters	0.95984	[0.94076;0.97653]	0.95906	[0.92315;0.97948]	0.99339
Hill estimator	0.39393	[0.26899;0.38734]	0.34828	[0.26658;0.37113]	0.42845

Table 9: Moments and statistics on actual data and simulated data using the GA and NMTA optimisation methods.

5.2.1 Objective Surfaces

It was then tested how the fitness of the simulation varied (according to the objective function) when allowing certain parameters to change. In each of the graphs that follow, 14 of the model parameters are set to their optimal values while the remaining two parameters vary within certain bounds. Thereafter, the objective function for each combination of parameter values are calculated and plotted. The graphs give an indication of how sensitive the fitness of the model is to changes in each parameter. While in some cases the fitness values can be rather haphazard, in each of the graphs a broad trend can be seen, indicating which areas in the parameter space tend to produce the highest fitness values. Some of the objective surfaces appear quite rough with the maximum not being clearly distinguishable (d_{min}, $d_{max} - d_{min}$ in figure 19), whereas others appear fairly smooth and a unique maximum is clearly distinguishable (N in figure 18). There also appears to be certain parameters for which a wide range of values result in optimal performance, but deviation from this range results in a sudden drop in fitness (H and Γ in figure 20).

As mentioned in section 1.1, these results are indicative of some parameter degeneracies where some parameters are calibrated successfully whilst others have no clear unique optimal value for multiple independent calibration experiments. For example, in figure 18 the unique optimal value for N is clear whereas there appears to be no unique optimal value for a. Interestingly, the minimum threshold for exiting a position seem to have minimal effect on the objective function value, while the selection of the maximum entering threshold signicantly influences the fitness value returned by the objective function.

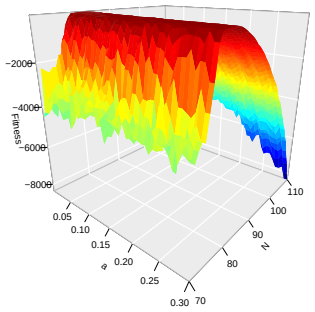

Figure 18: Objective surface (a, N)

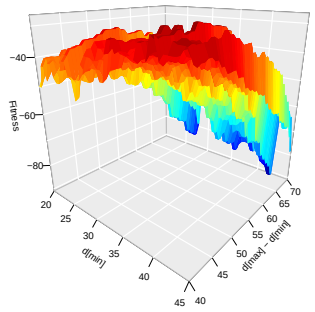

Figure 19: Objective surface $(d_{min}, d_{max} - d_{min})$

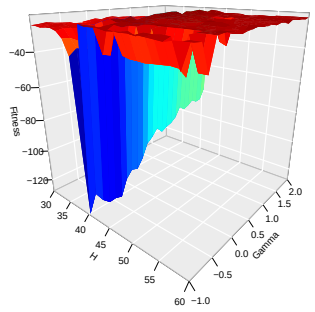

Figure 20: Objective surface (H, Γ)

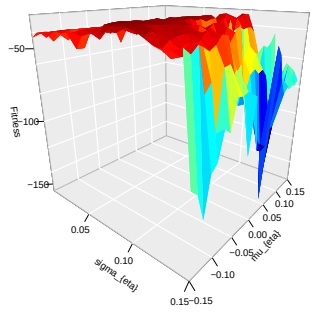

Figure 21: Objective surface (σ_η, μ_η)

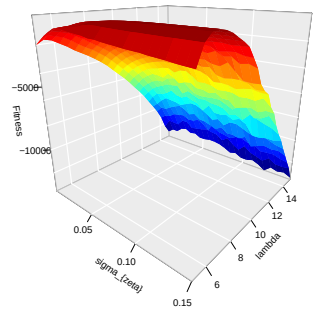

Figure 22: Objective surface (σ_ζ, λ)

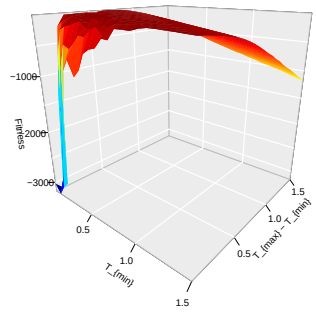

Figure 23: Objective surface $(T_{min}, T_{max} - T_{min})$

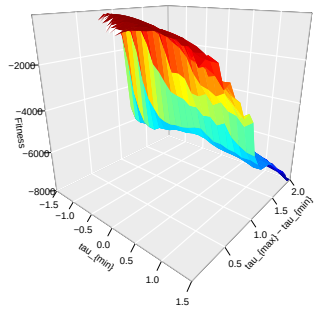

Figure 24: Objective surface $(\tau_{min}, \tau_{max} - \tau_{min})$

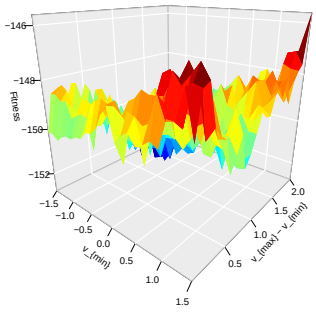

Figure 25: Objective surface $(v_{min}, v_{max} - v_{min})$

5.2.2 Comparison of Key Features of Observed and Simulated Data

Using the obtained best parameter values from each calibration method, a simulation of the Farmer-Joshi model was run. The resulting simulated log prices, along with the confidence intervals for the simulated prices and the actual closing log prices, are shown in figure 26. We observe that the actual price is rarely within the confidence intervals during the first few years. The confidence intervals are much narrower than for the standard model, indicating greater consistency between simulations when using the adaptive model. There is also a more consistent increase in the confidence intervals over time as opposed to the more sudden increases observed from the standard model.

In figure 27, with regard to the adaptive model, it can be seen that the daily log returns exhibit relatively similar variation to the actual log returns, however, areas of clustered volatility are not as prominent as that of the actual data and there are less cases of particularly high and low returns in the simulation compared to the observed returns.

Figure 28 shows the normal probability plots for the daily real and simulated returns. Both calibrations appear to have generated fat tails compared to the normal distribution with that of the NMTA calibration being less prominent. In figure 29 both calibrations exhibit almost no significant autocorrelations of log returns. Lastly, the GA calibration produced highly significant autocorrelations of absolute log returns which decayed at a faster rate than the observations whilst the autocorrelations for the NMTA calibration were less significant but decayed at a slower rate. Based on these results and comparing it to the standard model, it can be concluded that allowing agents to adapt and change over time results in volatility clustering.

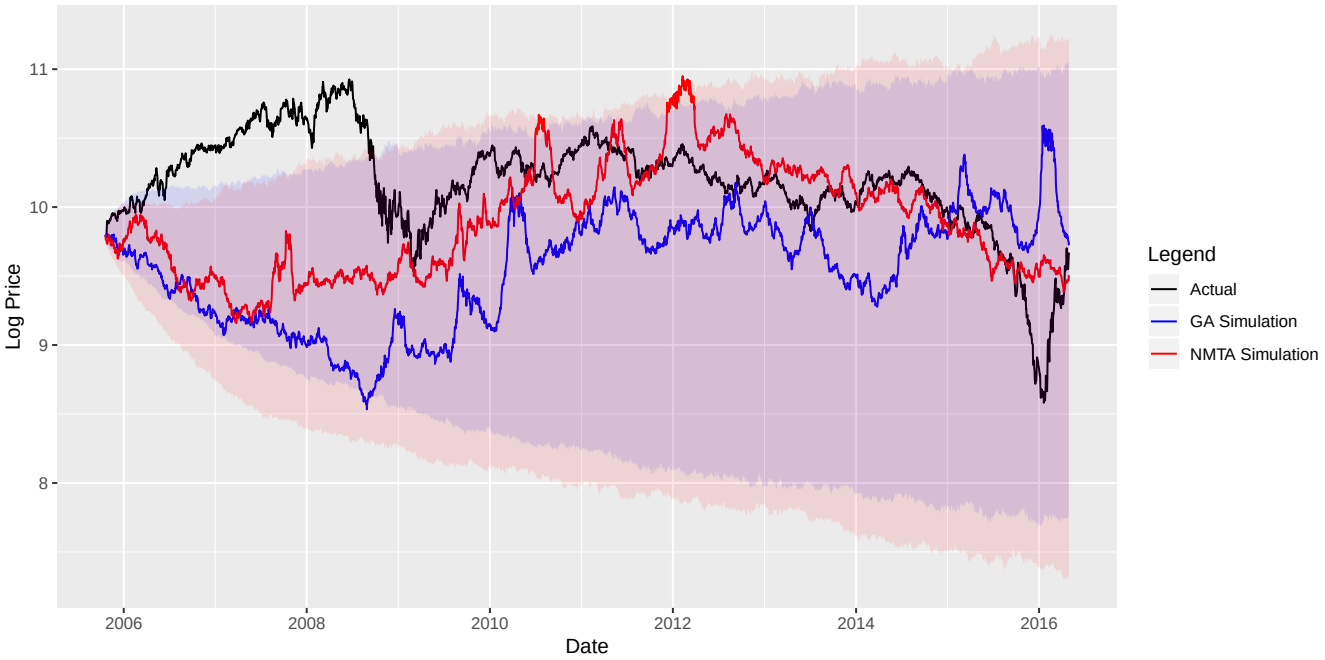

Figure 26: Observed and simulated closing log price paths (for Anglo-American shares)

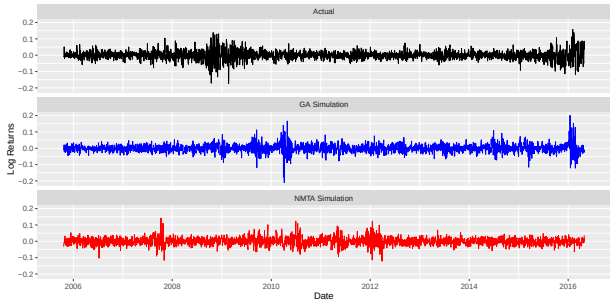

Figure 27: Log return paths

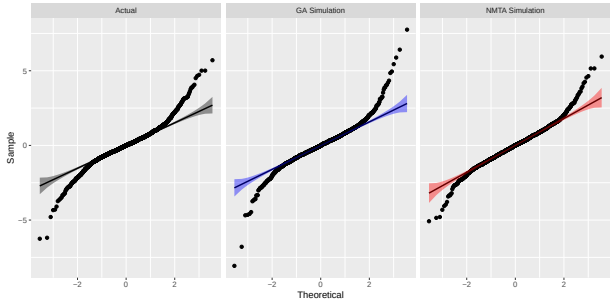

Figure 28: Normal Probability Plots

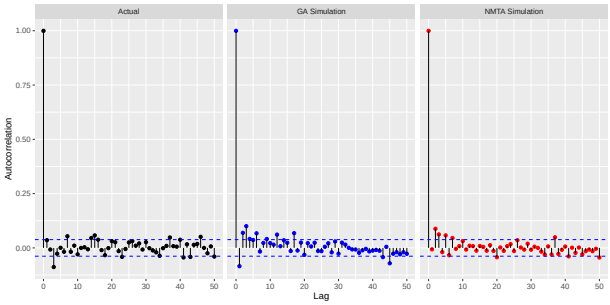

Figure 29: Autocorrelation of log returns

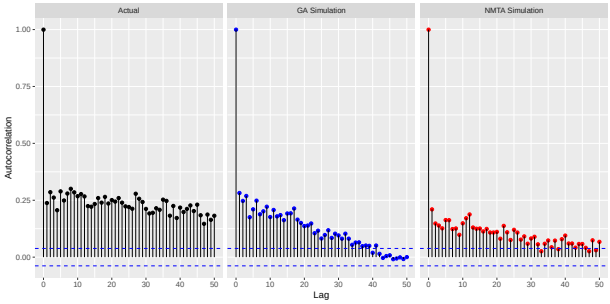

Figure 30: Autocorrelation of absolute log returns

5.2.3 Analysis of Strategy Profitability and Trader Switching Behaviour

Plotting the number of active chartists and fundamentalists at each time step reveals that switching is relatively volatile. To better visualise these movements in the number of agents, we apply a simple moving average to obtain the white line through each plot in figure 31. Note that the large jump in the log returns of the GA calibration just after 2010 (middle of figure 27) corresponds the large peak in the number of chartists that appears at the same time in the plot of the number of chartists (top left of figure 31). A similar observation can be made by looking at the areas of the number of chartists that corresponds to areas of volatility clustering in the plot of log returns for the NMTA calibration (bottom of figures 27 and 31). For this reason, we conclude that a large fraction of fundamentalists tends to stabilise prices, whereas a large fraction of chartists tends to destabilise prices. Furthermore, asset price fluctuations are caused by the interaction between these stabilizing and destabilizing forces.

Lastly, plotting the sum of all intermediate term profits realized by the fundamentalist and chartist strategies that were adopted by agents at each time step provides an explanation for why the chartist effect was more prominent in the GA algorithm compared to the NMTA algorithm. Looking at the left-hand side of figure 32, profit changes for chartists appear to be volatile and much more significant than that of fundamentalists (which is relatively stable). In contrast, the NMTA calibration produces less significant profits. This may explain why volatility clustering is less significant in this calibration as there is less chartist/fundamentalist activity/interaction.

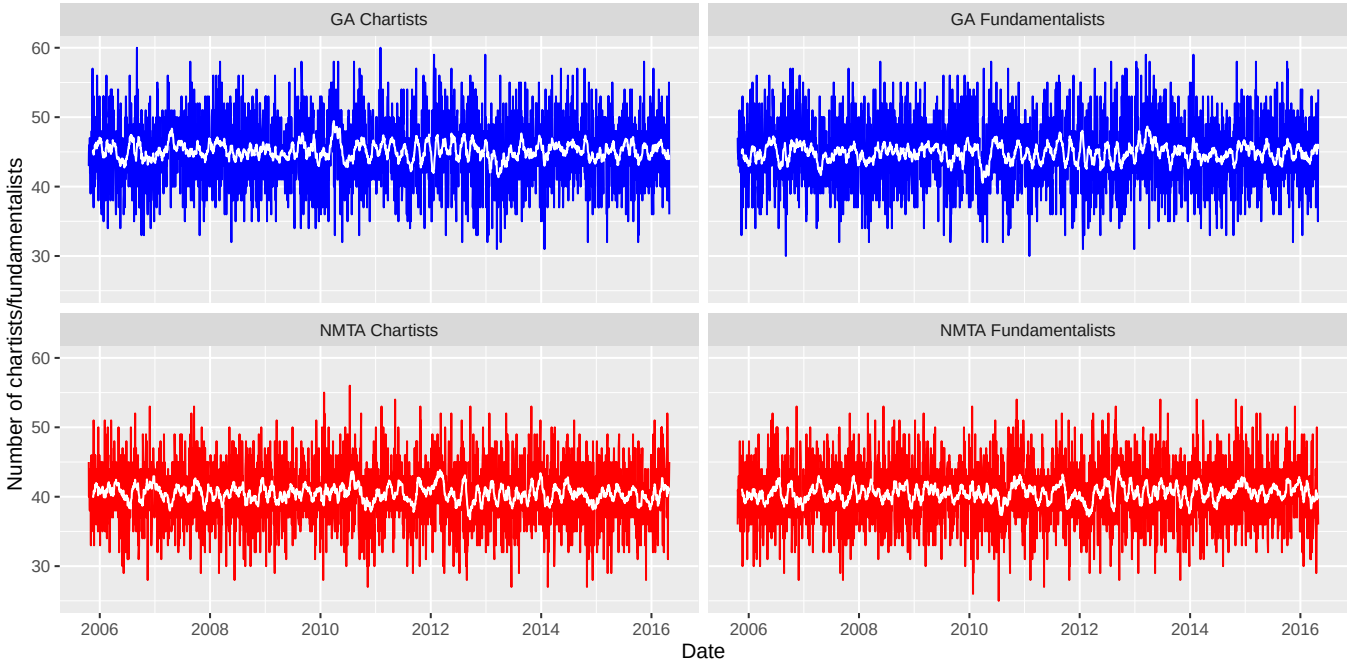

Figure 31: Number of chartists/fundamentalists

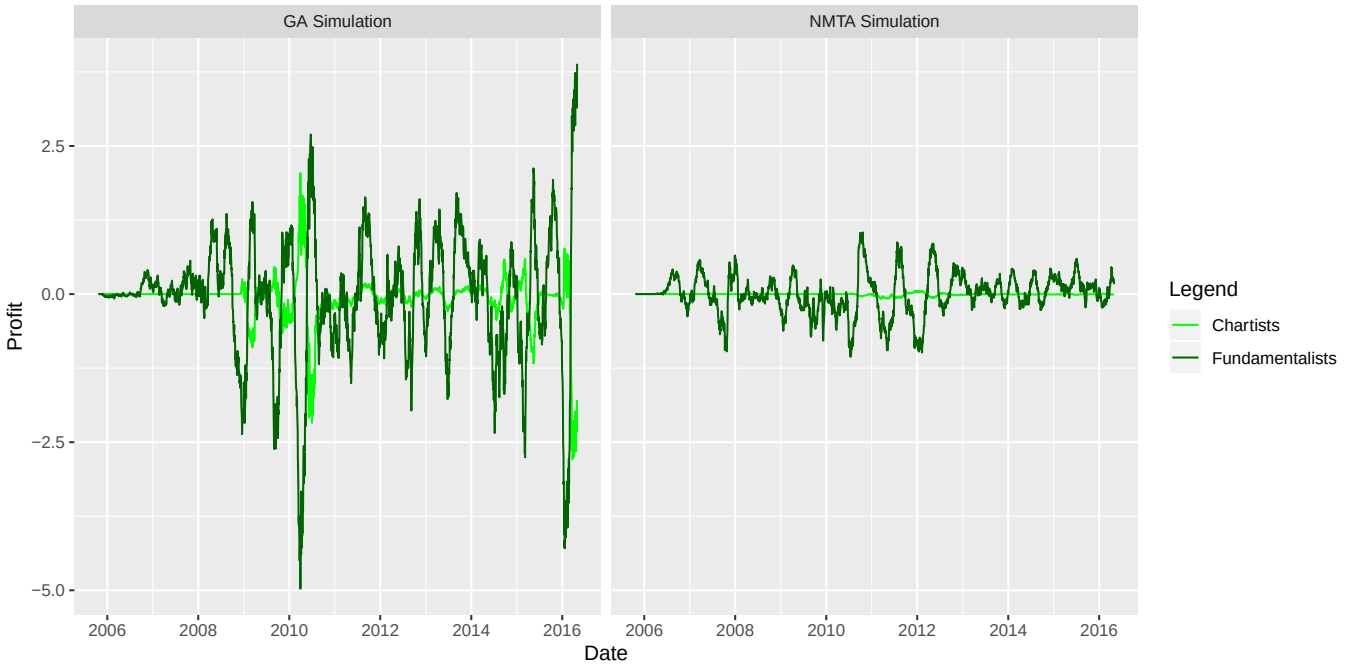

Figure 32: Profit of chartists/fundamentalists

6 Conclusion

A key feature of the Farmer-Joshi model is that the two types of agents are activated through the thresholds adopted by each agent. Farmer and Joshi argued that it was this feature that made it possible for the model to replicate the stylized facts. However, with regards to the standard Farmer-Joshi model, we have not been able to show that the model can replicate common empirical facts observed in financial markets - in particular leptokurtosis and volatility clustering. The less leptokurtic nature of simulated data is likely not due to the optimisation procedures but to the model characteristics or to the number of chartists and fundamentalists being fixed throughout each simulation.

A key feature of the adaptive model is that the fluctuations in the number of chartists and fundamentalists are triggered by a rational choice, based upon maximising profits over a time horizon of approximately two to three months. Allowing for greater trend following activity during periods of the simulation by introducing adaptive agents produced the desired volatility clustering, leptokurtosis and absence of autocorrelation of log returns. The adaptive Farmer-Joshi model leads to periods where chartist activity fluctuates between high and low, with an irregular switching between phases of: close to the "Efficient Market Hypothesis fundamental price" fluctuations, optimism with upward trends, and phases of pessimism with downward trends. The adaptive Farmer-Joshi model therefore does much better at replicating the stylized facts than the standard Farmer-Joshi model. Furthermore, it was found (by looking at the changes in the number and profit of chartists and fundamentalists over time) that a large fraction of fundamentalists tends to stabilise prices whilst a large fraction of chartists tends to destabilise prices. Together with this, a large value for Γ results in a greater degree switching compared to when Γ is small (a similar conclusion is made by Brock and Hommes [3]).

Comparing the two calibration methods, despite getting a better fitness, the optimal parameter values from the Nelder-Mead simplex with threshold accepting optimisation were less able to replicate the stylized facts than those from the genetic algorithm optimisation when we compared two simulations. This may be due to the random choice of simulations used not being representative of the average behaviour. It may also have been due to the the lower number of replications used in the threshold accepting algorithm resulting in slightly worse estimates of the fitness of the parameter sets.

When it comes to the behaviour of the parameters, we observe relatively large confidence intervals, as well as haphazard or flat objective function surfaces, for many of the parameters. This indicates that many parameters of the model do not have a very clear or significant effect on price behaviour. This makes it difficult to make insights about the effect of these parameters on share price behaviour, limiting the explanatory ability of the model. With this in mind, both models exhibit parameter degeneracies - that is, independent calibrations on the same data do not yield similar optimal parameters, while still resulting in a similar ability (or lack thereof) to replicate the stylized facts.

Glossary

calibration A reverse process to regression, where a known observation of the dependent variable (price) is used to determine the corresponding explanatory variables (parameters). 2

centroid The arithmetic mean position of all the points in a multidimensional figure. 17

chromosome A string which defines a set of parameter values. Each chromosome is a possible solution to the optimization problem. 19

contrarian An investment style in which investors purposefully go against prevailing market trends by selling when others are buying, and buying when most investors are selling. 9

double-auction market Allows buyers and sellers to submit prices they deem acceptable to a list. When a match between a buyers price and a sellers asking price is found, the trade proceeds at that price. Trades without matches will not be executed. 9

ergodicity A random process is ergodic if its time average is the same as its average over the probability space (known as its ensemble average). 7

fat tails The unconditional returns of nancial series are not normally distributed. They usually display a distribution with too many observations near the mean, too few in the mid range, and too many in the extreme left and right tails. 1

herding A situation in which market participants react to information about the behaviour of other market agents or participants rather than the behaviour of the market, and the fundamental transactions. Investors follow what they perceive other investors are doing, rather than their own analysis. 5

leptokurtic The condition of a probability density curve to have fatter tails and a higher peak at the mean than the normal distribution. In other words, a distribution with positive excess kurtosis. 29

limit order A type of order to purchase or sell a security at a specified price or better. For buy limit orders, the order will be executed only at the limit price or a lower one, while for sell limit orders, the order will be executed only at the limit price or a higher one. 4

limit order book A record of unexecuted limit orders maintained by the security specialist who works at the exchange. 9

liquidity provider An individual/institution which acts as a market maker in a given asset class. The liquidity provider will act as the both the buyer and seller of a particular asset, thus making a market. They place limit orders on the order books. 4

liquidity taker An individual/institution that places market orders to immediately buy/sell orders sitting on the books. 4

market microstructure A branch of finance concerned with the details of how exchange occurs in markets. While much of economics abstracts from the mechanics of trading, microstructure literature analyzes how specific trading mechanisms affect the price formation process. 2

market order A type of order to buy or sell a security at the best available price in the current market. 4

minority game It is inspired by the El Farol bar problem, which is a simple model that shows how (selfish) players cooperate with each other in the absence of communication. In the minority game, an odd number of players have to choose one of two choices independently at each turn. The players who end up on the minority side win. 6

mispricing The difference between a stock's price and its intrinsic/perceived value. 10

mutation The act of changing a parameter value of a member of the population to some randomly chosen other value. 19

noise trader A term used to describe investors who make decisions regarding buy and sell trades without the support of professional advice or advanced fundamental analysis (whose decisions to buy, sell, or hold are irrational and erratic). 9

parameter degeneracy In this case refers to the phenomenon where different calibration experiments do not result in similar optimal parameter values. 2

parent A set of parameter values in the current or intermediate population before recombination occurs. The best individuals in a generation are more likely to be selected as parents. 19

population A set of strings each representing a different combination of parameter values within the possible parameter space. 19

predator-prey Describes the dynamics of biological systems in which two species interact, one as a predator and the other as prey. Species compete, evolve and disperse for the purpose of obtaining resources (in this case profit). 5

recombination The process of creating offspring from a pair of parent strings by applying crossover to bits. 19

reproduce The process of selection, recombination and mutation which results in a new population. 19

simplex A generalization of the notion of a triangle or tetrahedron to arbitrary dimensions. i.e. an n-dimensional object with flat sides that has only $n+1$ vertices. 4, 17

spread The gap between the bid and the ask prices of a security/asset. 4

state dependent threshold strategy A strategy adopted by chartists and fundamentalists to reduce transaction costs. Agents enter their positions when their mispricing meets their entry threshold, and exit their positions when their mispricing meets their exit threshold. iii, 11

stylized fact Statistical properties that appear to be present in many empirical asset returns (across time and markets). In general, these are observations repeated in so many contexts that they are commonly accepted as empirical truths and set boundaries to which all new hypotheses must conform. 4

volatility clustering The observation, first noted by Mandelbrot [23], that "large changes tend to be followed by large changes, of either sign, and small changes tend to be followed by small changes". While stock returns themselves are relatively uncorrelated, the squares or absolute values of returns are autocorrelated, reflecting a tendency for markets to move from periods of relative quiet to more turbulent periods. Signicant positive autocorrelations for absolute stock returns continue out a year or more, and decay at a rate which is slower than exponential [21]. 15

Appendix

Psuedocode

In this section we use pseudocode to provide an overview of the objective function used in our calibrations of the adaptive Farmer-Joshi model, and a more detailed description of the function used to create simulations of the adaptive model.

Algorithm 1 Objective function

Require: $N, \lambda, a, d_{min}, d_{max}, \mu_\eta, \sigma_\eta, \sigma_\zeta, T_{min}, T_{max}, \tau_{min}, \tau_{max}, v_{min}, v_{max}, \Gamma, H$

Initialize number of Monte-Carlo replications I and matrix of I deviation vectors
for $i = 1, 2, \ldots, 30$ **do**
    Generate a simulated price path from parameters and calculate returns
    Get moments of simulated price path $\mathbf{m_i^s}|\boldsymbol{\theta}$
end for
$\mathbf{G} = \frac{1}{I}\sum_{i=1}^{I}[(\mathbf{m_i^s}|\boldsymbol{\theta}) - \mathbf{m^e}]$
return $-\mathbf{G'WG}$

Algorithm 2 Simulate function (adaptive Farmer-Joshi model)

Require: $N, \lambda, a, d_{min}, d_{max}\mu_\eta, \sigma_\eta, \sigma_\zeta, T_{min}, T_{max}, \tau_{min}, \tau_{max}, v_{min}, v_{max}, \Gamma, H$

Initialization $\forall\ i = 1, \ldots, N$:
$T_i =$ sample $U(T_{min}, T_{max})$ (constant entry threshold)
$\tau_i =$ sample $U(\tau_{min}, \tau_{max})$ (constant exit threshold)
$d_i =$ sample $d_{min} : d_{max}$ (constant time lag)
$c_i = a(T_i - \tau_i)$ (constant capital)
Set p_t to actual observed log prices for all integers t where $-200 \leq t \leq 0$.
$x_{i,0}^c = c_i \times \text{sign}(p_{-1} - p_{-1-d_i})$ (initial chartist position assuming entry threshold is met)
$v_{i,-1} =$ sample $U(v_{min} + p_{-1}, v_{max} + p_{-1})$
$x_{i,0}^f = c_i \times \text{sign}(v_{i,-1} - p_{-1})$ (initial fundamentalist position assuming entry threshold is met)
$m_{i,-1}^f = p_{-1} - v_{i,-1}$
$m_{i,-1}^c = p_{-1-d_i} - p_{-1}$
if $-T_i < m_{i,-1}^c < T_i$ **then**
    $x_{i,0}^c = 0$ (No position)
end if
if $-T_i < m_{i,-1}^f < T_i$ **then**
    $x_{i,0}^f = 0$ (No position)
end if
$\phi_{i,0}^c = 0.5$ (initial probability of being a chartist)
Assign each of N agents a strategy randomly according to $Binom(1, \phi_{i,0})$
Actual positions based on strategy $\forall\ i = 1, \ldots, N$
if Agent is currently using the chartist strategy **then**
    $x_{i,0}^a = x_{i,0}^c$
else
    $x_{i,0}^a = x_{i,0}^f$
end if

for $t = 0, 1, \ldots$ **do**
  Chartists $\forall\ i = 1, \ldots, N$ (including those not currently using chartist strategy)
  $x^c_{i,t+1} = c_i \times \text{sign}(p_t - p_{t-d_i})$ (assuming relevant thresholds are met, which are checked below)
  $m^c_{i,t} = p_{t-d_i} - p_t$
  if $x^c_{i,t} > 0$ & $m^c_{i,t} < -\tau_i$ (was in a long position; will hold if below -tau) **then**
    $x^c_{i,t+1} = x^c_{i,t}$ (No trade)
  end if
  if $x^c_{i,t} > 0$ & $-\tau_i < m^c_{i,t} < T_i$ **then**
    $x^c_{i,t+1} = 0$ (Exit long; don't enter short)
  end if
  if $x^c_{i,t} < 0$ & $m^c_{i,t} > \tau_i$ (was in a short position; will hold if above tau) **then**
    $x^c_{i,t+1} = x^c_{i,t}$ (No trade)
  end if
  if $x^c_{i,t} < 0$ & $-T_i < m^c_{i,t} < \tau_i$ **then**
    $x^c_{i,t+1} = 0$ (Exit short; don't enter long)
  end if
  if $x^c_{i,t} = 0$ & $-T_i < m^c_{i,t} < T_i$ **then**
    $x^c_{i,t+1} = 0$ (No trade)
  end if
  Fundamentalists $\forall\ i = 1, \ldots, N$ (including those not currently using fundamentalist strategy)
  $v_{i,t} = v_{i,t-1}$+sample $N(\mu_\eta, \sigma_\eta)$ (shift each agent's perceived value, where the same shift is applied to each age
  $x^f_{i,t+1} = c_i \times \text{sign}(v_t - p_t)$ (assuming relevant thresholds are met, which are checked below)
  $m^f_{i,t} = p_t - v_{i,t}$
  if $x^f_{i,t+1} > 0$ & $m^f_{i,t} < -\tau_i$ **then**
    $x^f_{i,t+1} = x^f_{i,t}$ (No trade)
  end if
  if $x^f_{i,t+1} > 0$ & $-\tau_i < m^f_{i,t} < T_i$ **then**
    $x^f_{i,t+1} = 0$ (Exit long; don't enter short)
  end if
  if $x^f_{i,t+1} < 0$ & $m^f_{i,t} > \tau_i$ **then**
    $x^f_{i,t+1} = x^f_{i,t}$ (No trade)
  end if
  if $x^f_{i,t+1} < 0$ & $-T_i < m^f_{i,t} < \tau_i$ **then**
    $x^f_{i,t+1} = 0$ (Exit short; don't enter long)
  end if
  if $x^f_{i,t+1} = 0$ & $-T_i < m^f_{i,t} < T_i$ **then**
    $x^f_{i,t+1} = 0$ (No trade)
  end if
  Actual positions based on strategy $\forall\ i = 1, \ldots, N$
  if Agent is currently using the chartist strategy **then**
    $x^a_{i,t+1} = x^c_{i,t+1}$ (assuming relevant thresholds are met, which are checked below)
    $m^a_{i,t} = m^c_{i,t}$
  else
    $x^a_{i,t+1} = x^f_{i,t+1}$ (assuming relevant thresholds are met, which are checked below)
    $m^a_{i,t} = m^f_{i,t}$
  end if

if $x^a_{i,t} > 0$ & $m^a_{i,t} < -\tau_i$ **then**
  $x^a_{i,t+1} = x^a_{i,t}$ (No trade)
end if
if $x^a_{i,t} > 0$ & $-\tau_i < m^a_{i,t} < T_i$ **then**
  $x^a_{i,t+1} = 0$ (Exit long; don't enter short)
end if
if $x^a_{i,t} < 0$ & $m^a_{i,t} > \tau_i$ **then**
  $x^a_{i,t+1} = x^a_{i,t}$ (No trade)
end if
if $x^a_{i,t} < 0$ & $-T_i < m^a_{i,t} < \tau_i$ **then**
  $x^a_{i,t+1} = 0$ (Exit short; don't enter long)
end if
if $x^a_{i,t} = 0$ & $-T_i < m^a_{i,t} < T_i$ **then**
  $x^a_{i,t+1} = 0$ (No trade)
end if
Market Impact (only considers positions that agents actually took)
$\omega_{t+1} = \sum_{i=1}^{N} \{x^a_{i,t+1} - x^a_{i,t}\}$
$p_{t+1} = p_t + \frac{1}{\lambda}\omega_{t+1} + N(0, \sigma_\zeta)$
Update $\forall\ i = 1, \ldots, N$
$\pi^c_{i,t+1} = \sum_{k=t-H+1}^{t} \{x^c_{i,k}(p_{k+1} - p_k)\}$
$\pi^f_{i,t+1} = \sum_{k=t-H+1}^{t} \{x^f_{i,k}(p_{k+1} - p_k)\}$
$\phi^c_{i,t+1} = \frac{e^{\pi^c_{i,t+1}/\Gamma}}{e^{\pi^c_{i,t+1}/\Gamma} + e^{\pi^f_{i,t+1}/\Gamma}}$
Assign each of N agents a new strategy randomly according to $Binom(1, \phi_{i,t})$
end for
return $p_t \quad \forall\ t$ (simulated price path)

R Code

In this section we provide the R code for the main functions used in our calibrations. This includes the code for calculating the weight matrix, the functions used to simulate each of the models ("Simulate" functions), the objective functions, and the function for implementing the NMTA algorithm on the standard model (the NMTA algorithm for the adaptive model is similar except for the addition of two further vertices for the two added parameters - H and Γ).

Listing 1 Moving block bootstrap for estimating $\boldsymbol{W}$

```
b = 100 # Size of window
k = 9 # Number of moments
bootstrap_moments = matrix(NA, nrow = 10000, ncol = k)
cl = makeCluster(detectCores())
registerDoParallel(cl)
bootstrap_moments = foreach(i = 1:10000, .combine = "rbind", .packages = c("
    pracma", "timeDate", "tseries", "fracdiff", "extremefit")) %dopar% {
  set.seed(i)
  index = sample(x = 1:(length(returns) - b + 1), size = ceiling(length(returns)/
      b), replace = TRUE) # Get block indices for bootstrap sample
  boot = c()
  for(j in 1:length(index)) { # Join the blocks together until we create a time
      series of at least length(returns)
    boot = c(boot, returns[(index[j]):(index[j]  + b - 1)])
  }
  boot = boot[1:length(returns)] # Truncate last block if necessary
  c(mean(boot), sd(boot), kurtosis(boot), ks.test(boot, returns)$statistic,
      hurstexp(boot, display = FALSE)$Hs, fdGPH(abs(boot))$d, adf.test(boot, k =
      0)$statistic, sum(garch(boot, trace = F)$coef[2:3]), mean(hill(boot)$hill
      [(0.05 * length(boot)):(0.1 * length(boot))]))
}
stopCluster(cl)
W = solve(cov(bootstrap_moments)) # Inverse of covariance matrix from
    distribution of bootstrapped moments
```

Listing 2 Simulate function (standard Farmer-Joshi model)

```
MarketImpact = function(omega, lambda, p_t, zeta) {
  return(p_t + (1/lambda)*omega + zeta)
}
Simulate = function(N, lambda, a, d_max, d_min, mu_eta, sigma_eta, sigma_zeta,
    T_max, tau_min, v_max, v_min, T_min = 0, tau_max = 0) {
  I = 1000 # Number of replications
  ## Initialization
  tau = matrix(NA, nrow = 2*N, ncol = I) # Each column represents a single Monte-
      Carlo replication and each row is an agent
  p_sim = matrix(NA, nrow = length(price), ncol = I) # Matrix of I vectors of
      simulated prices
  d = sapply(X = rep(x = N, times = I), FUN = sample, x = d_min:d_max, replace =
      TRUE) # Matrix of I vectors of lags for chartists
```

```
  Thresh = sapply(X = rep(x = 2*N, times = I), FUN = runif, min = T_min, max =
      T_max) # Matrix of I vectors of entry thresholds (T) for agents
  for (i in 1:I) {
    tau_constraint = cbind(Thresh[, i], -tau_min) # Used to ensure -tau < Thresh
    tau[, i] = runif(2*N, -colMins(t(tau_constraint)), tau_max) # Exit thresholds
        (tau) for each agent for ith replication
  }
  c_value = a*(Thresh[1:N, ] - tau[1:N, ]) # Matrix of I vectors of c's for
      fundamentalists
  c_trend = a*(Thresh[(N + 1):(2*N), ] - tau[(N + 1):(2*N), ]) # Matrix of I
      vectors of c's for chartists
  p_sim[1:(200 + 2), ] = price[1:(200 + 2)] # The simulated and actual prices are
       the same for this interval so that the lagged price does not fall behind
      our window of data
  v_previous = sapply(X = rep(x = N, times = I), FUN = runif, min = v_min + p_sim
      [201], max = v_max + p_sim[201]) # The initial value for each value investor
       is exogenous
  # Initial mispricings
  m = matrix(price[201], nrow = N, ncol = I, byrow = F) - v_previous # Mispricing
       that each fundamentalist believes there to be
  mc = matrix(price[201 - d], nrow = N, ncol = I, byrow = F) - matrix(price[201],
       nrow = N, ncol = I, byrow = F)
  # Initial positions assuming an entry threshold is met
  x_previous_trend = c_trend*sign(matrix(price[201], nrow = N, ncol = I, byrow =
      TRUE) - matrix(price[201 - d], nrow = N, ncol = I, byrow = FALSE)) # The
      initial positions of chartists
  x_previous_value = c_value*sign(v_previous - matrix(price[201], nrow = N, ncol
      = I, byrow = TRUE)) # The initial positions of fundamentalists
  # Initial positions are only taken if outside of entry thresholds
  x_previous_value[which(m < Thresh[1:N, ] & m > -Thresh[1:N, ], arr.ind = TRUE)]
       = 0 # Initial position is 0
  x_previous_trend[which(mc < Thresh[1:N, ] & mc > -Thresh[1:N, ], arr.ind = TRUE
      )] = 0 # Initial position is 0
  for (t in (200 + 3):(length(price))) {
    ## Chartists
    x_new_trend = c_trend*sign(matrix(p_sim[t - 1, ], nrow = N, ncol = I, byrow =
        TRUE) - matrix(p_sim[as.vector(t - 1 - d + length(price) * matrix(0:(I -
        1), nrow = N, ncol = I, byrow = T))], nrow = N, ncol = I, byrow = FALSE))
        # The positions chartists want to now move to
    ## Fundamentalists
    v_new = v_previous + sapply(X = rep(N, times = I), FUN = rnorm, mean = mu_eta
        , sd = sigma_eta) # The fundamentalists' new value judgements
    m = matrix(p_sim[t - 1, ], nrow = N, ncol = I, byrow = F) - v_new #
        Mispricing that each fundamentalist believes there to be
    mc = matrix(p_sim[as.vector(t - 1 - d + length(price) * matrix(0:(I - 1),
        nrow = N, ncol = I, byrow = T))], nrow = N, ncol = I, byrow = F) - matrix(
        p_sim[t - 1, ], nrow = N, ncol = I, byrow = T)
    x_new_value = c_value*sign(v_new - matrix(p_sim[t - 1, ], nrow = N, ncol = I,
        byrow = TRUE)) # The position they wish to now take (assuming thresholds
        are met for fundamentalists)
```

```
    # State dependent value strategy
    # Fundamentalist was in a long position; will only trade if below tau
    x_new_value[which(x_previous_value > 0 & m < -tau[1:N, ], arr.ind = TRUE)] =
        x_previous_value[which(x_previous_value > 0 & m < -tau[1:N, ], arr.ind =
        TRUE)] # No trade takes place
    x_new_value[which(x_previous_value > 0 & m < Thresh[1:N, ] & m > -tau[1:N, ],
         arr.ind = TRUE)] = 0 # Exits long but does not enter short position as m
        doesn't meet entry threshold
    # Fundamentalist was in a short position; will only trade if above -tau
    x_new_value[which(x_previous_value < 0 & m > tau[1:N, ], arr.ind = TRUE)] =
        x_previous_value[which(x_previous_value < 0 & m > tau[1:N, ], arr.ind =
        TRUE)] # No trade takes place
    x_new_value[which(x_previous_value < 0 & (m > -Thresh[1:N, ]) & (m < tau[1:N,
         ]), arr.ind = TRUE)] = 0 # Exits short but does not enter long position
        as m doesn't meet entry threshold
    # Fundamentalist was taking no position; will only trade if above Thresh or
        below -Thresh
    x_new_value[which(x_previous_value == 0 & m < Thresh[1:N, ] & m > -Thresh[1:N
        , ], arr.ind = TRUE)] = 0 # No trade takes place
    # Chartist was in a long position; will only trade if below tau
    x_new_trend[which(x_previous_trend > 0 & mc < -tau[(N+1):(2*N), ], arr.ind =
        TRUE)] = x_previous_trend[which(x_previous_trend > 0 & mc < -tau[(N+1):(2*
        N), ], arr.ind = TRUE)] # No trade takes place
    x_new_trend[which(x_previous_trend > 0 & mc < Thresh[(N+1):(2*N), ] & mc > -
        tau[(N+1):(2*N), ], arr.ind = TRUE)] = 0 # Exits long but does not enter
        short position as m doesn't meet entry threshold
    # Fundamentalist was in a short position; will only trade if above -tau
    x_new_trend[which(x_previous_trend < 0 & mc > tau[(N+1):(2*N), ], arr.ind =
        TRUE)] = x_previous_trend[which(x_previous_trend < 0 & mc > tau[(N+1):(2*N
        ), ], arr.ind = TRUE)] # No trade takes place
    x_new_trend[which(x_previous_trend < 0 & (mc > -Thresh[(N+1):(2*N), ]) & (mc
        < tau[(N+1):(2*N), ]), arr.ind = TRUE)] = 0 # Exits short but does not
        enter long position as m doesn't meet entry threshold
    # Fundamentalist was taking no position; will only trade if above Thresh or
        below -Thresh
    x_new_trend[which(x_previous_trend == 0 & mc < Thresh[(N+1):(2*N), ] & mc > -
        Thresh[(N+1):(2*N), ], arr.ind = TRUE)] = 0 # No trade takes place
    ## Fundamentalists and chartists
    omega = colSums(x_new_trend - x_previous_trend) + colSums(x_new_value -
        x_previous_value) # The net order of all I sets of agents for the day
    v_previous = v_new # Prepare for the next loop by updating v_previous
    x_previous_value = x_new_value # Prepare for the next loop
    x_previous_trend = x_new_trend # Prepare for the next loop
    p_sim[t, ] = MarketImpact(lambda = lambda, p_t = p_sim[t - 1, ], omega =
        omega, zeta = rnorm(I, 0, sigma_zeta)) # MarketImpact evaluated at a
        vector of parameters
  }
  return(p_sim) # I simulated price paths; First 202 days are real data
}
```

Listing 3 Objective function (standard Farmer-Joshi model)

```
ObjectiveFunction = function(parameters) {
  I = 50 # Number of replications
  p_sim = Simulate(N = ceiling(parameters[1]), lambda = parameters[2], a =
     parameters[3], d_max = ceiling(parameters[5]) + floor(parameters[4]), d_min
     = ceiling(parameters[5]), mu_eta = parameters[6], sigma_eta = parameters[7],
      sigma_zeta = parameters[8], T_max = parameters[9] + parameters[13], tau_min
      = parameters[10], v_max = parameters[11] + parameters[12], v_min =
     parameters[12], T_min = parameters[13], tau_max = parameters[14] +
     parameters[10]) # Ceiling and floor functions act as integer constraints
  sim_returns = diff(p_sim[-c(1:201), ])
  mean_sim = colMeans(sim_returns)
  stdev_sim = colSds(sim_returns)
  kurtosis_sim = apply(X = sim_returns, MARGIN = 2, FUN = kurtosis)
  hurst_sim = numeric(I)
  ks_sim = numeric(I)
  garch_sim = numeric(I)
  gph_sim = numeric(I)
  adf_sim = numeric(I)
  hill_sim = numeric(I)
  for (i in 1:I) {
    if(max(abs(na.omit(sim_returns[, i]))) > 10 | sum(is.na(sim_returns[, i])) >
       0) {
      return(NaN)
    } else {
      hurst_sim[i] = hurstexp(sim_returns[, i], display = FALSE)$Hs
      ks_sim[i] = ks.test(x = sim_returns[, i], y = returns)$statistic
      gph_sim[i] = fdGPH(abs(sim_returns[, i]))$d
      adf_sim[i] = adf.test(sim_returns[, i], k = 0)$statistic
      garch_sim[i] = sum(garch(sim_returns[, i], trace = F)$coef[2:3])
      hill_sim[i] = mean(hill(sim_returns[, i])$hill[(0.05 * length(sim_returns[,
          i])):(0.1 * length(sim_returns[, i]))])
    }
  }
  G = rbind(mean_sim, stdev_sim, kurtosis_sim, ks_sim, hurst_sim, gph_sim,
     adf_sim, garch_sim, hill_sim) - matrix(actual_moments, ncol = I, nrow = k,
     byrow = FALSE)
  G = apply(G, 1, trimmean, percent = 70)
  return(-t(G)%*%W%*%G) # Returns a negative value as GA maximises
}
```

Listing 4 Simulate function (adaptive Farmer-Joshi model)

```
MarketImpact = function(omega, lambda, p_t, zeta) {
  return(p_t + (1/lambda)*omega + zeta)
}
```

```
Profit = function(p_cur, p_prev, x_prev) {
  x_prev*(p_cur - p_prev) # Difference between trade value today and the value of the order placed yesterday which comes into effect today
}
Simulate = function(price = price, N, lambda, a, d_max, d_min, mu_eta, sigma_eta, sigma_zeta, T_max, tau_min, v_max, v_min, Gamma = 10^(-9), H = 2, T_min = 0, tau_max = 0) {
  ## Initialization
  p_sim = numeric(length(price)) # Vector of simulated prices
  p_sim[1:(200 + 2)] = price[1:(200 + 2)] # The simulated and actual prices are the same for this interval so that the lagged price does not fall behind our window of data
  # Tracking
  N_c = numeric(length(price)) # Keep track of number of active chartists at each time point
  N_f = numeric(length(price)) # Keep track of number of active fundamentalists at each time point
  pi_c = numeric(length(price)) # Keep track of profit for chartists at each time point
  pi_f = numeric(length(price)) # Keep track of profit for fundamentalists at each time point
  # Data frame keeping track of all agents' strategies
  agents = data.frame(d = sample(size = N, x = d_min:d_max, replace = TRUE), # Vector of lags for chartists
                      Thresh = runif(N, T_min, T_max), # Vector of entry thresholds
                      tau = NA, c_value = NA, c_trend = NA,
                      v_previous = runif(N, min = v_min + p_sim[201], max = v_max + p_sim[201]), # The initial value for each investor is exogenous
                      v_new = NA, x_previous_value = NA, x_previous_trend = NA, x_previous_actual = numeric(N),
                      x_new_value = NA, x_new_trend = NA,  x_new_actual = numeric(N),
                      m_value = NA, m_trend = NA, m_actual = numeric(N),
                      profit_value = rep(0, times = N), profit_trend = rep(0, times = N), # No profit initially
                      full_profit_value = rep(0, times = N), full_profit_trend = rep(0, times = N),
                      phi = rep(0.5, times = N), # phi = probability of being a chartist
                      chartist = as.logical(rbinom(N, size = 1, prob = 0.5)), # Randomly initialize strategies for each agent based on binomial distribution with probability 0.5
                      prev_day_chartist = NA,
                      fundamentalist = NA, prev_day_fundamentalist = NA)
  agents$tau = runif(N, -colMins(t(cbind(agents$Thresh, -tau_min))), tau_max) # Used to ensure -tau < Thresh. Vector of exit thresholds
  agents$c_value = a*(agents$Thresh - agents$tau) # Vector of c's for all agents' value investing strategies
```

```
  agents$c_trend = agents$c_value # The ith agent has the same c for both his
     strategies (remains constant throughout)
  # Get mispricing for both fundamentalists and chartists
  agents$m_value = p_sim[201] - agents$v_previous # Mispricing that each
     fundamentalist believes there to be
  agents$m_trend = p_sim[201 - agents$d] - p_sim[201] # Mispricing that each
     chartist believes there to be
  # Set their initial positions as if the thresholds have been met
  agents$x_previous_trend = agents$c_trend*sign(p_sim[201] - p_sim[201 - agents$d
     ]) # The initial positions for all agents' trend following strategies
  agents$x_previous_value = agents$c_value*sign(agents$v_previous - p_sim[201]) #
      The initial positions for all agents' value investing strategies
  # Check that thresholds have been met. Agents only take the positions above if
     they are outside one of the entry thresholds
  agents$x_previous_value[which(agents$m_value < agents$Thresh & agents$m_value >
      -agents$Thresh)] = 0 # No trade takes place
  agents$x_previous_trend[which(agents$m_trend < agents$Thresh & agents$m_trend >
      -agents$Thresh)] = 0 # No trade takes place
  agents$fundamentalist = !agents$chartist # If they're not currently a chartist
     then they're a fundamentalist
  agents$prev_day_chartist = agents$chartist
  agents$prev_day_fundamentalist = agents$fundamentalist
  past_profits_trend <- matrix(0, nrow = N, ncol = H)
  past_profits_value <- matrix(0, nrow = N, ncol = H)
  agents$x_previous_actual[which(agents$chartist)] = agents$x_previous_trend[
     which(agents$chartist)]
  agents$x_previous_actual[which(agents$fundamentalist)] = agents$
     x_previous_value[which(agents$fundamentalist)]
  # In what follows, both strategies for each agent is updated as normal but only
      their specific strategy for that time is used to calculate omega
  for (t in (200 + 3):(length(price))) {
    ## Chartists
    agents$x_new_trend = agents$c_trend*sign(p_sim[t - 1] - p_sim[t - 1 - agents$
       d]) # The positions chartists want to now move to. Update all agents'
       trend following strategies
    agents$m_trend = p_sim[t - 1 - agents$d] - p_sim[t - 1]
    ## Fundamentalists
    agents$v_new = agents$v_previous + rnorm(1, mean = mu_eta, sd = sigma_eta) #
       The fundamentalists' new value judgements. Update all agents' value
       investing strategies
    agents$m_value = p_sim[t - 1] - agents$v_new # Mispricing that each
       fundamentalist believes there to be
    agents$x_new_value = agents$c_value*sign(agents$v_new - p_sim[t - 1]) # The
       position they wish to now take (assuming thresholds are met for
       fundamentalists)
    ## State dependent value strategy
    # Fundamentalist was in a long position; will only trade if below tau
    agents$x_new_value[which(agents$x_previous_value > 0 & agents$m_value < -
       agents$tau)] = agents$x_previous_value[which(agents$x_previous_value > 0 &
        agents$m_value < -agents$tau)] # No trade takes place
```

```
61      agents$x_new_value[which(agents$x_previous_value > 0 & agents$m_value <
            agents$Thresh & agents$m_value > -agents$tau)] = 0 # Exits long but does
            not enter short position as m doesn't meet entry threshold
62      # Fundamentalist was in a short position; will only trade if above -tau
63      agents$x_new_value[which(agents$x_previous_value < 0 & agents$m_value >
            agents$tau)] = agents$x_previous_value[which(agents$x_previous_value < 0 &
             agents$m_value > agents$tau)] # No trade takes place
64      agents$x_new_value[which(agents$x_previous_value < 0 & agents$m_value > -
            agents$Thresh & agents$m_value < agents$tau)] = 0 # Exits short but does
            not enter long position as m doesn't meet entry threshold
65      # Fundamentalist was taking no position; will only trade if above Thresh or
            below -Thresh
66      agents$x_new_value[which(agents$x_previous_value == 0 & agents$m_value <
            agents$Thresh & agents$m_value > -agents$Thresh)] = 0 # No trade takes
            place
67      ## State dependent trend strategy
68      # Chartist was in a long position; will only trade if below tau
69      agents$x_new_trend[which(agents$x_previous_trend > 0 & agents$m_trend < -
            agents$tau)] = agents$x_previous_trend[which(agents$x_previous_trend > 0 &
             agents$m_trend < -agents$tau)] # No trade takes place
70      agents$x_new_trend[which(agents$x_previous_trend > 0 & agents$m_trend <
            agents$Thresh & agents$m_trend > -agents$tau)] = 0 # Exits long but does
            not enter short position as m doesn't meet entry threshold
71      # Chartist was in a short position; will only trade if above -tau
72      agents$x_new_trend[which(agents$x_previous_trend < 0 & agents$m_trend >
            agents$tau)] = agents$x_previous_trend[which(agents$x_previous_trend < 0 &
             agents$m_trend > agents$tau)] # No trade takes place
73      agents$x_new_trend[which(agents$x_previous_trend < 0 & agents$m_trend > -
            agents$Thresh & agents$m_trend < agents$tau)] = 0 # Exits short but does
            not enter long position as m doesn't meet entry threshold
74      # Chartist was taking no position; will only trade if above Thresh or below -
            Thresh
75      agents$x_new_trend[which(agents$x_previous_trend == 0 & agents$m_trend <
            agents$Thresh & agents$m_trend > -agents$Thresh)] = 0 # No trade takes
            place
76      ## Actual strategy
77      agents$x_new_actual[which(agents$chartist)] = agents$x_new_trend[which(agents
            $chartist)]
78      agents$m_actual[which(agents$chartist)] = agents$m_trend[which(agents$
            chartist)]
79      agents$x_new_actual[which(agents$fundamentalist)] = agents$x_new_value[which(
            agents$fundamentalist)]
80      agents$m_actual[which(agents$fundamentalist)] = agents$m_value[which(agents$
            fundamentalist)]
81      # Agent was in a long position; will only trade if below tau
82      agents$x_new_actual[which(agents$x_previous_actual > 0 & agents$m_actual < -
            agents$tau)] = agents$x_previous_actual[which(agents$x_previous_actual > 0
             & agents$m_actual < -agents$tau)] # No trade takes place
83      agents$x_new_actual[which(agents$x_previous_actual > 0 & agents$m_actual <
            agents$Thresh & agents$m_actual > -agents$tau)] = 0 # Exits long but does
            not enter short position as m doesn't meet entry threshold
```

```
    # Fundamentalist was in a short position; will only trade if above -tau
    agents$x_new_actual[which(agents$x_previous_actual < 0 & agents$m_actual >
        agents$tau)] = agents$x_previous_actual[which(agents$x_previous_actual < 0
         & agents$m_actual > agents$tau)] # No trade takes place
    agents$x_new_actual[which(agents$x_previous_actual < 0 & agents$m_actual > -
        agents$Thresh & agents$m_actual < agents$tau)] = 0 # Exits short but does
        not enter long position as m doesn't meet entry threshold
    # Fundamentalist was taking no position; will only trade if above Thresh or
        below -Thresh
    agents$x_new_actual[which(agents$x_previous_actual == 0 & agents$m_actual <
        agents$Thresh & agents$m_actual > -agents$Thresh)] = 0 # No trade takes
        place
    ## Fundamentalists and chartists
    omega = sum(agents$x_new_actual - agents$x_previous_actual) # The net order
        of all agents for the day. Select only those agents' specific strategies
    ## Market maker
    p_sim[t] = MarketImpact(lambda = lambda, p_t = p_sim[t - 1], omega = omega,
        zeta = rnorm(1, 0, sigma_zeta))
    ## Update adaptive probability, profit of each agent's strategy and which
        strategy each agent is adopting
    agents$profit_value = Profit(p_cur = p_sim[t], p_prev = p_sim[t - 1], x_prev
        = agents$x_previous_value)
    agents$profit_trend = Profit(p_cur = p_sim[t], p_prev = p_sim[t - 1], x_prev
        = agents$x_previous_trend)
    past_profits_trend = cbind(agents$profit_trend, past_profits_trend[, 1:(H-1)
        ])
    past_profits_value = cbind(agents$profit_value, past_profits_value[, 1:(H-1)
        ])
    agents$full_profit_trend = rowSums(past_profits_trend)
    agents$full_profit_value = rowSums(past_profits_value)
    agents$prev_day_chartist = agents$chartist
    agents$prev_day_fundamentalist = agents$fundamentalist
    ## Tracking
    N_c[t-1] = sum(agents$chartist)
    N_f[t-1] = sum(agents$fundamentalist)
    pi_c[t] = sum(agents$full_profit_trend)
    pi_f[t] = sum(agents$full_profit_value)
    agents$phi = (exp(agents$full_profit_trend/Gamma))/(exp(agents$
        full_profit_trend/Gamma) + exp(agents$full_profit_value/Gamma)) # Gamma is
         the intensity of switching parameter
    agents$chartist = as.logical(sapply(X = agents$phi, FUN = rbinom, n = 1, size
        = 1)) # Choose strategies based on each agent's probability of switching
    agents$fundamentalist = !agents$chartist
    ## Update positions
    agents$v_previous = agents$v_new # Prepare for the next loop by updating
        v_previous for all agents
    agents$x_previous_value = agents$x_new_value # Prepare for the next loop by
        updating x_previous_value for all agents
    agents$x_previous_trend = agents$x_new_trend # Prepare for the next loop by
        updating x_previous_trend for all agents
```

```
    agents$x_previous_actual = agents$x_new_actual # Prepare for the next loop by
        updating x_previous_actual for all agents
  }
  return(data.frame(p_sim = p_sim, N_c = N_c, N_f = N_f, pi_c = pi_c, pi_f = pi_f
      )) # Single simulated price path
}
```

Listing 5 Objective function (adaptive Farmer-Joshi model)

```
ObjectiveFunction = function(parameters) {
  I = 30 # Number of replications
  G = matrix(NA, nrow = k, ncol = I) # Initialise G
  for (i in 1:I) {
    set.seed(i) # Ensure there is a different seed for each replication
    p_sim = Simulate(price = price, N = ceiling(parameters[1]), lambda =
        parameters[2], a = parameters[3], d_max = ceiling(parameters[5]) + floor(
        parameters[4]), d_min = ceiling(parameters[5]), mu_eta = parameters[6],
        sigma_eta = parameters[7], sigma_zeta = parameters[8], T_max = parameters
        [9] + parameters[15], tau_min = parameters[10], v_max = parameters[11] +
        parameters[12], v_min = parameters[12], Gamma = parameters[13], H =
        ceiling(parameters[14]), T_min = parameters[15], tau_max = parameters[10]
        + parameters[16])$p_sim # Ceiling and floor functions act as integer
        constraints
    returns_sim = diff(p_sim[-c(1:201)])
    # Controlling errors from poor initialisations
    if(max(abs(na.omit(returns_sim))) > 10 | sum(is.na(returns_sim)) > 0) {
      return(NaN)
    } else {
      moments = Moments(returns_sim, returns)
      mean_sim = moments[1]
      stdev_sim = moments[2]
      kurtosis_sim = moments[3]
      ks_sim = moments[4]
      hurst_sim = moments[5]
      gph_sim = moments[6]
      adf_sim = moments[7]
      garch_sim = moments[8]
      hill_sim = moments[9]
    }
    G[, i] = c(mean_sim, stdev_sim, kurtosis_sim, ks_sim, hurst_sim, gph_sim,
        adf_sim, garch_sim, hill_sim) - Moments(returns, returns)
  }
  G = matrix(apply(G, 1, mean), nrow = k, ncol = 1)
  return(-t(G)%*%W%*%G) # Fitness
}
```

Listing 6 NMTA function (standard Farmer-Joshi model)

```
threshold_accepting = function(vertices, fitnesses) {
  # This function implements the random shift step of the NMTA algorithm
  tau = c(3, 1.5, 0) # Thresholds for each round
  n = length(fitnesses) - 1 # Number of parameters
  for(i in 1:3) { # Each round we reduce tau and increase the number of
     replications
    for(j in 1:7) { # Within each round we shift up to 7 parameters
      direction = sample(size = 1, x = 1:n) # Determine which parameter to shift
      magnitude = runif(n = 1, min = -0.5, max = 0.5) * mean(vertices[direction,
         ]) # Determine how much to shift it by
      # Get new candidate solutions
      temp_vertices = vertices
      temp_vertices[direction, ] = temp_vertices[direction, ] + magnitude
      # Restrict shifts to possible values only
      temp_vertices[1, ] = pmax(1, temp_vertices[1, ])
      temp_vertices[2, ] = pmax(0.001, temp_vertices[2, ])
      temp_vertices[3, ] = pmax(0, temp_vertices[3, ])
      temp_vertices[4, ] = pmax(0, temp_vertices[4, ])
      temp_vertices[4, ] = pmin(100, temp_vertices[4, ])
      temp_vertices[5, ] = pmax(1, temp_vertices[5, ])
      temp_vertices[5, ] = pmin(90, temp_vertices[5, ])
      temp_vertices[7, ] = pmax(0, temp_vertices[7, ])
      temp_vertices[8, ] = pmax(0.00000001, temp_vertices[8, ])
      temp_vertices[9, ] = pmax(0, temp_vertices[9, ])
      temp_vertices[10, ] = pmin(0, temp_vertices[10, ])
      temp_vertices[11, ] = pmax(0, temp_vertices[11, ])
      temp_vertices[14, ] = pmax(0, temp_vertices[14, ])
      # Get their fitnesses
      temp_fitnesses = foreach(k = 1:(n+1), .combine = "c", .export = ls(
         globalenv()), .packages = list.of.packages) %dorng% {
        ObjectiveFunction(c(temp_vertices[, k], 15 + 5 * i))
      }
      # Replace the old solutions with the new ones if the best solution from the
         new set is good enough
      if(max(temp_fitnesses) > max(fitnesses) - tau[i]) {
        vertices = temp_vertices
        fitnesses = temp_fitnesses
      }
    }
  }
  return(rbind(vertices, fitnesses))
}

Calibrate_NMTA = function(maxiter, I) { # 'I' is the number of replications
  # Get starting vertices and fitnesses
  initials = foreach(k = 1:15, .combine = "cbind", .export = ls(globalenv()), .
     packages = list.of.packages) %dorng% {
```

```
    # Randomly create parameter values
    vertex = c(runif(1, 40, 240), # N_f, N_c
               runif(1, 0, 15), # liquidity
               runif(1, 0, 1.5), # capital assignment
               runif(1, 0, 100), # d_max - d_min
               runif(1, 0.01, 90), # d_min
               runif(1, -0.01, 0.01), # mu_eta
               runif(1, 0, 0.05), # var_eta
               runif(1, 0, 0.05), # var_zeta
               runif(1, 0, 1), # T_max - T_min
               runif(1, -1, 0), # tau_min
               runif(1, 0, 1), # v_max - v_min
               runif(1, -0.5, 0), # v_min
               runif(1, 0, 1), # tau_max - tau_min
               runif(1, 0, 1)) # T_min
    fitness = ObjectiveFunction(c(vertex, I))
    while(fitness == -100000) { # Ensure that simulation does not result in crazy
        behaviour
      vertex = c(runif(1, 40, 240), runif(1, 0, 15), runif(1, 0, 1.5), runif(1,
          0, 100), runif(1, 0.01, 90), runif(1, -0.01, 0.01), runif(1, 0, 0.05),
          runif(1, 0, 0.05), runif(1, 0, 1), runif(1, -1, 0), runif(1, 0, 1),
          runif(1, -0.5, 0), runif(1, 0, 1), runif(1, 0, 1))
      fitness = ObjectiveFunction(c(vertex, I))
    }
    c(fitness, vertex)
  }
  vertices = initials[-1, ]
  fitnesses = initials[1, ]
  # Sort from best to worst
  vertices = vertices[, order(fitnesses, decreasing = T)]
  fitnesses = sort(fitnesses, decreasing = T)
  # Keep track of best ever result
  best_ever_iteration = 0
  best_ever_fitness = fitnesses[1]
  best_ever_vertex = vertices[, 1]
  # Print progress
  progress = c(0, fitnesses[1], vertices[, 1])
  print(progress)
  # Set the calibration parameters
  alpha = 0.15 # Probability of choosing threshold accepting section
  sigmaNM = 0.5 # Shrinkage parameter
  rhoNM = 0.5 # Contraction parameter
  alphaNM = 1 # Reflection parameter
  gammaNM = 2 # Expansion parameter
  for (r in 1:maxiter) {
    if (runif(1, 0, 1) < alpha) { # Threshold accepting
      new_parameters = threshold_accepting(vertices, fitnesses)
      vertices = new_parameters[1:(14), ]
      fitnesses = new_parameters[15, ]
    }
```

```
    else {
      # Reflect the worst point to the other side of the average of each of the
          other parameters
      centroid = rowMeans(vertices[, -15])
      reflected_point = centroid + alphaNM * (centroid - vertices[, 15])
      # Restrict shifts to possible values only
      reflected_point[1] = max(1, reflected_point[1])
      reflected_point[2] = max(0.001, reflected_point[2])
      reflected_point[3] = max(0, reflected_point[3])
      reflected_point[4] = max(0, reflected_point[4])
      reflected_point[4] = min(100, reflected_point[4])
      reflected_point[5] = max(1, reflected_point[5])
      reflected_point[5] = min(90, reflected_point[5])
      reflected_point[7] = max(0, reflected_point[7])
      reflected_point[8] = max(0.00000001, reflected_point[8])
      reflected_point[9] = max(0, reflected_point[9])
      reflected_point[10] = min(0, reflected_point[10])
      reflected_point[11] = max(0, reflected_point[11])
      reflected_point[14] = max(0, reflected_point[14])
      ref_fitness = ObjectiveFunction(c(reflected_point, I))
      # Check if new solution is in between best and worst fitnesses
      if(ref_fitness < fitnesses[1] & ref_fitness > fitnesses[14]) {
        vertices[, 15] = reflected_point
        fitnesses[15] = ref_fitness
      }
      # Check if new parameters are new best solution; if so try improve further
          by moving more in that direction
      else if(ref_fitness > fitnesses[1]) {
        # Expand
        expanded_point = centroid + gammaNM * (reflected_point - centroid)
        # Restrict shifts to possible values only
        expanded_point[1] = max(1, expanded_point[1])
        expanded_point[2] = max(0.001, expanded_point[2])
        expanded_point[3] = max(0, expanded_point[3])
        expanded_point[4] = max(0, expanded_point[4])
        expanded_point[4] = min(100, expanded_point[4])
        expanded_point[5] = max(1, expanded_point[5])
        expanded_point[5] = min(90, expanded_point[5])
        expanded_point[7] = max(0, expanded_point[7])
        expanded_point[8] = max(0.00000001, expanded_point[8])
        expanded_point[9] = max(0, expanded_point[9])
        expanded_point[10] = min(0, expanded_point[10])
        expanded_point[11] = max(0, expanded_point[11])
        expanded_point[14] = max(0, expanded_point[14])
        exp_fitness = ObjectiveFunction(c(expanded_point, I))
        # Check if the solution is even better now
        if(exp_fitness > ref_fitness) {
          vertices[, 15] = expanded_point
          fitnesses[15] = exp_fitness
```

```
136         } else { # If not then just use the original improvement
137           vertices[, 15] = reflected_point
138           fitnesses[15] = ref_fitness
139         }
140       }
141       # Check if new parameters are still the worst. If so, try contracting
142       else if(ref_fitness < fitnesses[14]) {
143         contracted_point = centroid + rhoNM * (vertices[, 15] - centroid)
144         con_fitness = ObjectiveFunction(c(contracted_point, I))
145         # Check if contracting managed to actually get a better result than the
              original worst parameter
146         if(con_fitness > fitnesses[15]) {
147           vertices[, 15] = contracted_point
148           fitnesses[15] = con_fitness
149         } else {
150           # If none of the above helped, shrink everything towards best solution
151           vertices = vertices[, 1] + sigmaNM * (vertices - vertices[, 1])
152           fitnesses = c(fitnesses[1], foreach(k = 2:15, .combine = "c", .export =
                ls(globalenv()), .packages = list.of.packages) %dorng% {
153             ObjectiveFunction(c(vertices[, k], I))
154           })
155         }
156       }
157     }
158     # Sort from best to worst and try again
159     vertices = vertices[, order(fitnesses, decreasing = T)]
160     fitnesses = sort(fitnesses, decreasing = T)
161     if(fitnesses[1] > best_ever_fitness) { # Keep track of best ever result
162       best_ever_iteration = r
163       best_ever_fitness = fitnesses[1]
164       best_ever_vertex = vertices[, 1]
165     }
166     progress = c(r, fitnesses[1], vertices[, 1])
167     names(progress) = c("Iteration", "fitness", "N", "lambda", "a", "damx - dmin"
          , "dmin", "mu_eta", "sigma_eta", "sigma_zeta", "T_max - T_min", "tau_min",
          "v_max - vmin", "v_min", "T_min", "tau_max - tau_min")
168     print(progress)
169   }
170   return(c(best_ever_iteration, best_ever_fitness, best_ever_vertex))
171 }
```

References

[1] *Jse auction process*, https://www.jse.co.za/grow-my-wealth/jse-auction-process.

[2] Daniel Bernoulli, *Exposition of a new theory on the measurement of risk*, Econometrica **22** (1954), no. 1, 23–36.

[3] William A. Brock and Cars H. Hommes, *Heterogeneous beliefs and routes to chaos in a simple asset pricing model*, Journal of Economic Dynamics and Control **22** (1998), no. 8-9, 1235–1274.

[4] Andrea Cavagna, Juan P. Garrahan, Irene Giardina, and David Sherrington, *Thermal model for adaptive competition in a market*, Physical Review Letters **83** (1999), no. 21, 4429.

[5] D. Challet and Y. C. Zhang, *Emergence of cooperation and organization in an evolutionary game*, Physica A: Statistical Mechanics and its Applications **246** (1997), no. 3-4, 407–418.

[6] Rama Cont, *Empirical properties of asset returns: Stylized facts and statistical issues*, Quantitative Finance **1** (2001), 223–236.

[7] Annalisa Fabretti, *On the problem of calibrating an agent based model for financial markets*, Journal of Economic Interaction and Coordination **8** (2013), no. 2, 277–293.

[8] J Doyne Farmer and Shareen Joshi, *The price dynamics of common trading strategies*, Journal of Economic Behavior & Organization **49** (2002), no. 2, 149–171.

[9] Milton Friedman and Marilyn Friedman, *The case for flexible exchange rates. in: Essays in positive economics*, University of Chicago Press, 1953.

[10] Michael Gant and Tim Gebbie, *Jse top 40 constituents daily price relatives 2003-2018 (bloomberg)*, Mendeley Data, v1, 2019.

[11] Manfred Gilli and Peter Winker, *A global optimization heuristic for estimating agent based models*, Computational Statistics & Data Analysis **42** (2003), no. 3, 299–312.

[12] Dhananjay K. Gode and Shyam Sunder, *Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality*, Journal of Political Economy **101** (1993), no. 1, 119–137.

[13] David E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, 1st ed., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[14] Lynne Hamill and Nigel Gilbert, *Agent-based modelling in economics*, John Wiley & Sons, 2015.

[15] Christiaan Heij, Paul de Boer, Philip H. Franses, Teun Kloek, Herman K. van Dijk, et al., *Econometric methods with applications in business and economics*, Oxford University Press, 2004.

[16] Francis Heylighen, *Complexity and self-organization*, Encyclopedia of Library and Information Sciences **3** (2008), 1215–1224.

[17] John H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, University of Michigan Press, 1975.

[18] Alan Kirman and Gilles Teyssiere, *Microeconomic models for long memory in the volatility of financial time series*, Studies in Nonlinear Dynamics & Econometrics **5** (2002), no. 4, 1–23.

[19] Sandrine J. Leal, Mauro Napoletano, Andrea Roventini, and Giorgio Fagiolo, *Rock around the clock: An agent-based model of low-and high-frequency trading*, Journal of Evolutionary Economics **26** (2016), no. 1, 49–76.

[20] Blake LeBaron, *Agent-based computational finance: Suggested readings and early research*, Journal of Economic Dynamics and Control **24** (2000), no. 5-7, 679–702.

[21] ———, *Agent-based computational finance*, Handbook of Computational Economics **2** (2006), 1187–1233.

[22] Martin Lettau, *Explaining the facts with adaptive agents: The case of mutual fund flows*, Journal of Economic Dynamics and Control **21** (1997), no. 7, 1117–1147.

[23] Benoit B. Mandelbrot, *The variation of certain speculative prices*, The Journal of Business **40** (1967), no. 4, 393–413.

[24] John A. Nelder and Roger Mead, *A simplex method for function minimization*, The Computer Journal **7** (1965), no. 4, 308–313.

[25] M O'Hara, *Market microstructure theory*, Wiley, 1997.

[26] Ole Peters and Alexander Adamou, *Ergodicity economics*, WordPress, 2018.

[27] Donovan F. Platt, *The calibration of financial agent-based models*, Master's thesis, University of the Witwatersrand, 2017.

[28] ———, *A comparison of economic agent-based model calibration methods*, 2019, pp. 1–32.

[29] Donovan F. Platt and Tim J. Gebbie, *The problem of calibrating an agent-based model of high-frequency trading.*

[30] ———, *Can agent-based models probe market microstructure?*, Physica A: Statistical Mechanics and its Applications **503** (2018), 1092–1106.

[31] Tobias Preis, Sebastian Golke, Wolfgang Paul, and Johannes J. Schneider, *Multi-agent-based order book model of financial markets*, EPL (Europhysics Letters) **75** (2006), no. 3, 510–516.

[32] L Wang, Kwangwon Ahn, C Kim, and C Ha, *Agent-based models in financial market studies*, Journal of Physics: Conference Series **1039** (2018), no. 1, 012022.

[33] Darrell Whitley, *A genetic algorithm tutorial*, Statistics and Computing **4** (1994), no. 2, 65–85.

[34] Peter Winker, Manfred Gilli, and Vahidin Jeleskovic, *An objective function for simulation based inference on exchange rate data*, Journal of Economic Interaction and Coordination **2** (2007), no. 2, 125–145.

[35] Chi Ho Yeung and Yi-Cheng Zhang, *Minority games*, Encyclopedia of Complexity and Systems Science (2009), 5588–5604.

[36] E Christopher Zeeman, *On the unstable behaviour of stock exchanges*, Journal of Mathematical Economics **1** (1974), no. 1, 39–49.