July 2014

Comparative Analysis of two Artificial Stock Markets Time Series Properties

Gabriele Visentin

Abstract

Despite the increasing quantity of models proposed in Agent-Based Financial Economics for the replication of real financial time series, little attention has been devouted so far to the actual statistical analysis of the resulting price series. This paper intends to offer a simple layout of the relevant statistical properties and phenomena that an artificial stock market is expected to replicate, emphasizing what deductions can be made from a comparative study of more than one model.

After briefly introducing the reader to the basic notions of financial time series analysis and Agent-Based Financial Economics, two artificial stock market models are presented: the Santa Fe Institute Artificial Stock Market (ASM) model and the Continuous Double Auction (CDA) model.

These models have been implemented in NetLogo and their statistical properties have been computed in R through the NetLogo-R interface provided by the RNetLogo library.

A statistical analysis of the data has been conducted to test the replication of the empirical stylized facts that characterize real financial time series, testing the performance of both models with respect to a benchmark provided by the IBM stock over a suitably long time period.

The models are able to replicate several phenomena of real financial assets to a varying degree of accuracy. The results are exposed in a comparative way, highlighting the differences in performance originating from the two models.

In particular the ASM model seems to replicate more closely industrial stocks with lower excess kurtosis and is more suitable for simulation of daily price and volume time series.

The CDA model, on the other hand, more closely resembles financial indeces and intraday trading time series, while completely lacking the ability to successfully replicate volume-related statistical properties.

1 Introduction

1.1 Stylized empirical facts of financial time series

Fostered by the application of computer-intensive analysis of financial data at ever finer time-scales, over the past two decades a surprising quantity of literature in the fields of statistics and financial mathematics has been devouted to the study of financial time series. Remarkably enough a set of very distinct statistical properties has been found to be shared among the most diverse assets, providing a unique signature for the identification of these processes.

These properties have come to be collectively known as the "stylized empirical facts" of financial time series[1].

Most of the attention has been focused on the statistical properties of the logarithmic returns of prices

$$r(t, \Delta t) = log(P(t + \Delta t)) - log(P(t))$$

with specific emphasis on the time domain analysis of the autocorrelation functions of their various moments and the cross-correlation with other quantities of interest, most notably the volume of transactions.

A general summary of these properties can be outlined as follows:

- **Absence of autocorrelation**
  For $\Delta t > 10$ minutes, the autocorrelation of log-returns vanishes at all positive lags, lending credit to the Efficient Market Hypothesis, postulating that prices follow a random walk process.

- **Leptokurtic distribution**
  The unconditional distribution of log-returns shows a significant positive excess kurtosis, leading to the typical heavy-tailed density with a power-law decay much slower than the exponential decay typical of the Gaussian distribution.

- **Volatility clustering**
  The autocorrelation function of the squared log-returns decays slowly in the time lags, indicating that high (low) volatility tends to be followed by high (low) volatility in the market.

- **Conditional heavy tails**
  Even after accounting for volatility clustering by autoregressive terms through an ARCH or GARCH fit, the residuals follow a leptokurtic distribution.

- **Volume-volatility cross-correlation**
  The time series of volume is positively correlated with volatility and more generally with any moment of the log-returns.

- **Leverage effect**
  The volatility of an asset is negatively correlated with the returns of that asset.

It must be noted that these statistical features are rather peculiar and it should not come as a surprise that immense efforts have been spent in trying to model stochastic processes that could replicate them. In particular a class of processes known as generalized autoregressive heteroskedasticity (GARCH) processes were put forward as good candidates in the 1980's and have been extended over the years to more closely replicate particular financial assets or trading conditions.

Nevertheless the effort to successfully replicate these statistical behaviors in a consistent and robust way is still under way. One promising approach that seeks to explain the emergence of these properties by modeling the actual system that generates them comes from the field of Agent-Based Financial Economics, which has seen a proliferation of different models in the past decade.

1.2 Agent-Based Financial Economics

Agent-Based Financial Economics is the computational modeling of financial markets as dynamic systems of interacting agents. The Agent-Based Model approach [2] allows us to investigate the emergence of macroscopic properties in the marketplace by specifying how single agents form expectations and what market mechanisms are used to implement trading [3].

The edge offered by today's computing power allows us to bridge the gap between the classical theory of rational homogeneous expectations and the every day experience of traders employing heterogeneous and continuously evolving predictions of market behavior [4]. The hurdles of analytical intractability are overcome by the new simulation-based approach, providing a deeper insight into the emergence of complex phenomena.

Several models attempting to replicate the financial markets have been proposed since the 1990's [5] with mixed degrees of success.

This paper focuses on two models: the Santa Fe Institute Artificial Stock Market (ASM) and the Continuous Double Auction Model (CDA). These two models differ enormously in the way agents' expectations are modeled and in the market mechanism regulating trading.

Santa Fe Institute Artificial Stock Market (ASM)[1]

This model was first developed in 1996 [6] by a team of professors affiliated with the Santa Fe Institute. The fundamental insight at the core of this

[1] for a more detailed exposition than is otherwise possible in this paper, consult Leigh Tesfatsion *Detailed notes on the Santa Fe Artificial Stock Market (ASM) Model* at http://www2.econ.iastate.edu/

model is the heterogeneous and dynamic nature of expectations that traders adopt: trading strategies are endogeneously determined by the market and are allowed to change over time through the implementation of a genetic algorithm (GA) [7].

The financial assets traded are a riskless asset (a bond) paying a fixed interest r and a risky one (a stock available in 25 shares) issuing dividends at each trading period. The dividend is assumed to be an autoregressive process of order one (AR(1)):

$$d_t = \bar{d} + \rho(d_{t-1} - \bar{d}) + w_t \tag{1}$$

where w_t is a zero-mean Gaussian white noise process of variance σ_w^2. The total number of traders is fixed at 25 and each one of them is endowed with 100 predictors (that is, 100 different ways to compute expectations).

Each predictor consists of

- a *condition*, specifying in what market state it can be used,
- an *action*, specifying how to compute the expectation,
- a *variance*, storing in memory how well the predictor performed in the past.

and poorly performing predictors are thrown away and replaced via a *genetic algorithm*.

Condition

At the beginning of each trading period a market state is computed, summarizing for all the investors what is assumed to be the relevant information at that time.

Bit	Bit inequality
1	$p_t r/d_t > 0.25$
2	$p_t r/d_t > 0.5$
3	$p_t r/d_t > 0.75$
4	$p_t r/d_t > 0.875$
5	$p_t r/d_t > 1$
6	$p_t r/d_t > 1.125$
7	$p_t > MA(5)$
8	$p_t > MA(10)$
9	$p_t > MA(100)$
10	$p_t > MA(500)$
11	fixed at 1
12	fixed at 0

Table 1 - *Market state computation. $MA(n)$ is the moving average process of order n of the price series itself.*

The market state is a string of 12 bits, where each bit is set to one if the corresponding inequality is true, zero otherwise, as explained in table 1.

The bits in the first block are called fundamental bits because they relate the classical fundamental value of the stock to its current price (the infamous P/E price-earnings ratio most traders are familiar with).

The second block consists of technical bits, i.e. those bits that relate the current price to its past trend, as is usually done in technical analysis.

The last two bits are control bits, held at fixed values at all times.

Each predictor's condition is a string of 12 bits that are set at 0, 1 or #. When a predictor's condition matches the market state (# is a wildcard) then that predictor is said to be activated: it becomes a candidate to compute the agent's expectation in that period.

Action

A predictor's action is a vector of two coefficient a and b used to compute the expectation of next period's price and dividend as a linear function of today's price and dividend

$$\mathbb{E}(p_{t+1} + d_{t+1}) = a(p_t + d_t) + b$$

The assumption of a linear expectation here might seem restrictive at first, but it must be noted that by combining all 100 predictors and considering that they are activated at different market states one can easily build complex non-linear expectations in the phase-space of the market by simply gluing together simple linear expecations of this kind.

Variance

Among the active predictors the one with the least variance is chosen for computing that period's expectation[2]. When the price of the following period is calculated and posted, each predictor that was active updates its variance[3] according to the rule

$$\sigma_t^2 = (1 - \theta)\sigma_{t-1}^2 + \theta(p_{t+1} + d_{t+1} - \mathbb{E}(p_{t+1} + d_{t+1}))^2 \tag{2}$$

that is, the variance is a moving average of the squared forecast error at each period the predictor is active. The parameter θ measures the time horizon at which traders judge their predictors. An high θ means the most recent error is given the greatest importance.

[2]this seems to be a contentious point: in [6] it is implied at one point that a statistical combination of the first H active predictors is used for computing the expectation, while LeBaron successively reported that the one with the least variance is actually picked

[3]in [8] is actually explained that this value is stored in memory, but the actual forecast variance is updated only during the genetic algorithm (see next section) and is kept fixed between consecutive implementations

Market Clearing

Agents are assumed to share identical constant absolute risk aversion (CARA) utility functions as a function of their wealth

$$U(W) = \exp -\lambda W$$

As all agents submit expectations and variances of their best predictors the market is cleared imposing a Walrasian (temporary) price equilibrium, by equilizing total demand and total supply. Each agent is assumed to maximize the expected value of his utility function for the next period. Optimal demand[4], as a function of an agent's best predictor's expectation and variance, is

$$x_t(p_t) = \frac{\mathbb{E}(p_{t+1} + d_{t+1}) - p_t(1+r)}{\lambda \sigma_t^2} \tag{3}$$

By solving for p_t the market clearing price can be computed.

Genetic Algorithm

Each predictor has associated a fitness measure, reflecting how well it performs

$$f_t = -(\sigma_t^2 + s) \tag{4}$$

where s is called specificity and is the number of bits set (i.e. different from #) in that predictor's condition. At the end of each trading period every agent implements a genetic algorithm for the evolution of predictors with a fixed probability p_u. Predictors are ranked according to their fitness and the 20% worst predictors are replaced. A new predictor can either come from mutation of one of the remaining predictors or by recombination of two of them. In both cases the parents are chosen by tournament selection and all the parameters of the new predictor are chosen in a variety of ways according to fixed probabilities[5].

Further details

Some important details for correctly implementing the model have not been included in the preceding paragraphs. In particular all traders are endowned with a default predictor, whose condition consists of all #'s (is matched in any market state) and whose action coefficients are averages of action coefficients of the remaining 99 predictors of that agent, weighted with the inverse of their variances. Furthermore, if a predictor has not

[4]the formula presented is actually optimal demand assuming prices have a Gaussian distribution, which is not the case at hand. Unfortunately the general solution is often analytically intractable and this approximation had to be made for computational flexibility

[5]see [8] for an exhaustive exposition of the GA implementation

been activated for T periods it is forced to mutate its condition. This latter process is known as generalization and is a way of recycling predictors whose conditions are contradictory (e.g. first bit to 0 and second bit to 1) and would otherwise never be activated. The price is evolved for 500 periods upon initilization, during which no trading takes place and the price is set at d_t/r. This is done in order to have a valid MA(500) estimate available at the beginning of the first trading period.

Continuous Double Auction (CDA) Model

As we have seen, in the ASM model a market clearing price is imposed at each period by computing a (temporary) Walrasian equilibrium. As any trader would know this mechanism is highly unrealistic. Investors don't submit their orders synchronously, nor is there a central auctioneer imposing a market clearing condition on aggregate demand and supply.

Instead transactions are processed one at a time by a mechanism known as continuous double auction: buyers and sellers trasmit at any time their orders, specifying at what price they would buy or sell the asset. Order prices are stored in an electronic order book, in which the lowest selling price (ask) and the highest buying price (bid) are kept on top. Their values typically differ, with the ask inevitably higher than the bid, depending on the asset liquidity and the trading conditions.

As a transaction takes place the buyer is charged the ask price, while the seller receives the bid price. The spread between the ask and the bid is pocketed by the clearing house or dealer hosting the transaction.

In the CDA model the agents are Zero-Intelligence (ZI) traders, i.e. traders submitting random ask or bid prices.

At the beginning of a trading day each trader opts out of the market for that day with a fixed propability. The traders remaining pick a random price from a Gaussian distribution centered around the last executed price, with a fixed variance σ^2.

All traders are randomly divided into buyers and sellers, with the probability of being a buyer slightly decreasing as a function of the last executed price. This allows the price time series to be stationary, artificially forcing a bull market when the price is too low and a bear market when the price is too high.

Traders submit their bid or ask prices (depending on whether they are buyers or sellers) one at a time. The submitted prices are then kept in two separate logs, one for buyers and one for sellers, respectively sorted in decreasing and increasing order of price. At each price submission if the possibility arises, a transaction is implemented following the continuous double auction mechanism explained above.

The price time series obtained by recording every single transaction gives a picture of intraday trading and is called the tick prices time series (the

tick being by definition the amount of time separating two successive transactions).

One can also obtain a daily price time series by recording exclusively the closing price of the day, i.e. the last executed price of the day.

These two time series have different statistical properties as will be shown in the last part of the paper.

2 Model implementation

Both models have been implemented in NetLogo and their time series have been analyzed in R through the RNetLogo library. We now briefly expose the main structure of the code.

2.1 NetLogo implementation of the ASM model

Global variables

- `Ninvestors`: number of investors in the market, fixed at 25
- `Nshares`: total number of shares available for trading, fixed at 25
- `Npredictors`: number of predictors for trader, fixed at 100
- `Nmarketbits`: length of the market state bitstring, fixed at 12
- `market-state`: a list of length `Nmarketbits` storing the bits specifying the market state at that time
- `r`: interest rate paied by the riskless asset, fixed at 0.10
- `lambda`: constant absolute risk aversion (CARA) coefficient in the utility function of traders, fixed at 0.5
- `theta`: moving average coefficient in the variance updating formula (2), fixed at 1/75
- `p-u`: probability of implementing the genetic algorithm for predictor evolution, fixed at 1/250
- `p`: probability of implementing recombination among predictors, rather than mutation, fixed at 0.10
- `T`: number of periods regulating the generalization process for inactive predictors, fixed at 4000
- `price-series`: a list storing all prices in the history of the model
- `volume-series`: a list storing the number of transactions per trading period in the history of the model

- `dividend-series`: a list storing the dividend issued at the beginning of each trading period, as in equation (1)

Agent variables

Considering that predictors undergo significant mutations by the GA and are asked to call several reporters, it is convenient to model both traders and predictors as agents in NetLogo. The following is a list of the variables associated with each predictor.

- `owner`: integer number specifying the `who` number of the owner of the predictor
- `condition`: list of length `Nmarketbits` specifying the predictor's condition
- `action`: list of two floating point numbers specifying a and b coefficients in the predictor's action
- `var` : floating point number specifying the predictor's variance σ_t^2
- `actual-var`: floating point number recording the moving average of the squared mean forecast error for variance updating, as in equation (2)
- `fitness`: floating point number specifying the predictor's fitness as in equation (4)
- `default`: boolean variable detailing whether the predictor is default or not
- `active`: boolean variable detailing whether the predictor has been activated in that trading period
- `last-active`: boolean variable detailing whether the predictor had been activated in the previous trading period
- `last-active-at`: integer number storing last time the predictor was activated
- `best-predictor`: boolean variable detailing whether the predictor is that trading period's agent's best predictor

Procedures

- `initialize-predictors`: initializes all predictors' variables when `setup` is called

- `update-default-predictor`: initializes and udpates all default predictors' variables at each trading period
- `generate-market-state`: computes that period's market state and stores it in `market-state`
- `mutate`: procedure in turtle-own context called by a parent predictor for generation of a mutated offspring predictor
- `recombination`: procedure called with two parent predictors as mandatory arguments for generation of a recombined offspring predictor
- `mutate-for-inactivity`: procedure for condition mutation in turtle-own context called by predictors who have not been activated for `T` periods

Reporters

- `market-state-comparison`: reporter for comparison between a predictor's `condition` and `market-state`. Returns a boolean value depending on the outcome of the comparison
- `specificity`: reporter computing specificity of a predictor's condition for fitness updating
- `tournament`: reporter returning one predictor selected by tournament selection among predictors in a pool
- `moving-average`: reporter for moving average computation on the `price-series` list

2.2 NetLogo implementation of the CDA model

The code structure is fairly simple, with a standard `setup` procedure to initialize the model and a `go` procedure to run all transactions in a trading day.

Global variables

- `nRandomAgents`: integer number specifying the number of traders in the market, fixed at 1000
- `logB`: a list storing all submitted bid prices, sorted in decreasing order. The first price is the bid at that time
- `logS`: a list storing all submitted ask prices, sorted in increasing order. The first price is the ask at that time
- `exePrice`: a list storing all executed tick prices in the history of the model

- `dailyPrice`: a list storing all daily prices (last tick price of each day) in the history of the model
- `volume`: a list storing the number of transactions per day
- `passLevel`: floating point number between 0 and 1 (set by the observer) detailing the probability for traders of opting out of the market for the day, fixed at 0.15

Agent variables

- `pass`: a boolean variable specifying whether the trader has opted out of trading for the day.
- `buy`: a boolean variable specifying whether the trader is a buyer.
- `sell`: a boolean variable specifying whether the trader is a seller.
- `price`: a floating point number specifying the trader's bid/ask price.

2.3 Data retrieval interface from NetLogo to R

There are several ways to interface NetLogo and R scripts for statistical real-time analysis of the simulation data[9]. It is possible both to embed R in NetLogo by *R-Extension* or *Rserve-Extension* and to embed NetLogo in R by the *RNetLogo* library.

The latter method is the one applied here. The script for data retrieval is straighforward: by using the functions detailed in the RNetLogo documentation it's possible to launch NetLogo from R, to execute commands and report data in an Rdata format, all from an R terminal.

- `library(RNetLogo)`: loads the RNetLogo library functions in the R workspace
- `NLStart(path)`: launches NetLogo from R terminal, `path` is a string storing the (absolute or relative) path to the local directory where NetLogo has been installed (the directory where NetLogo.jar is saved)
- `NLLoadModel(path)`: loads a NetLogo model into R, `path` is a string storing the (absolute or relative) path to the local directory where the NetLogo model has been saved
- `NLCommand('setup')`: executes the `setup` command in the NetLogo model
- `NLDoCommand(n, 'go')`: executes the `go` command in the NetLogo model `n` times

- `NLReport(variable)`: retrives any variable from the NetLogo model and saves it as a dataframe R object, `variable` is a string corresponding to the NetLogo variable name

3 Time series analysis of data

A total of 5 simulations have been run for each model.

The ASM simulations have been run for 100,000 trading periods each to allow sufficient learning and the next 10,000 periods have been recorded.

In the CDA case no learning is involved, so the first 10,000 trading periods of each simulation have been immediately recorded.

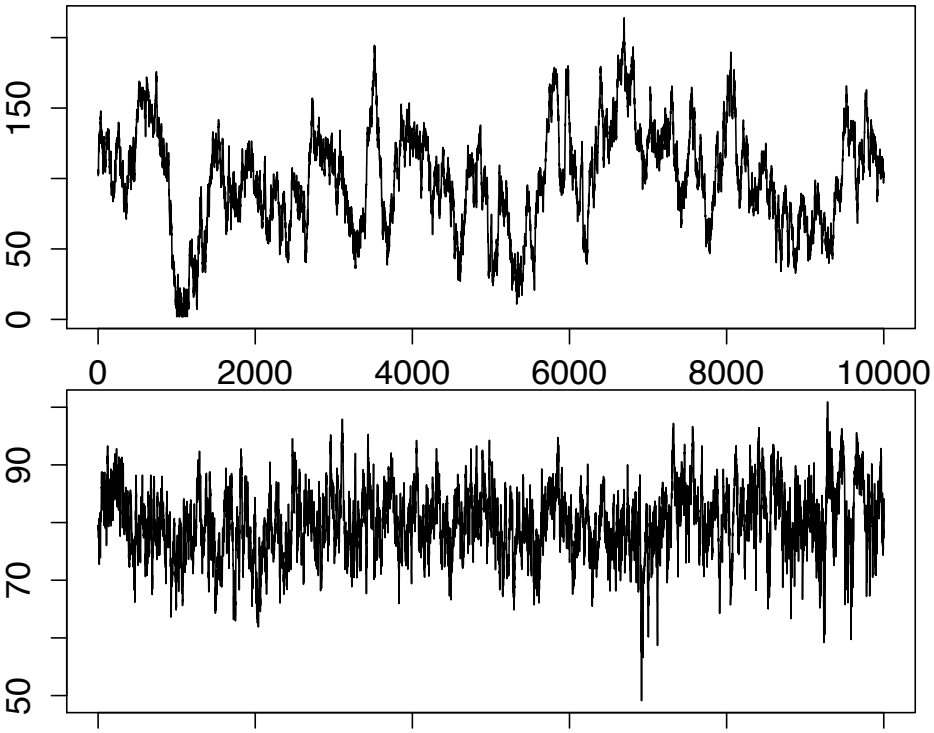

Figure 1 - *CDA daily-price (top) and ASM price-series (bottom) sample paths*

The `price-series` and `volume-series` time series for the ASM and the `exePrice`, `dailyPrice` and `volume` time series for the CDA model have been analysed using R in a comparative way in order to establish the hypothetical replication of the empirical stylized facts of financial time series.

Sample paths of the price time series from one simulation of each model can be seen in figure 1.

All relevant statistics have been compared with a "benchmark stock" represented by the IBM historical quotes from January 1964 to mid-July 2014.

The log-returns autocorrelation functions, shown in figure 2, vanish at positive time lags, as is typical of all financial assets.

It should be noted that a vanishing autocorrelation function doesn't imply independence, as the random walk hypothesis seems to suggest.

If log-returns were truly independent then any non-linear function of them would still display a vanishing autocorrelation function[1].

Unfortunately the situation is not quite that simple.

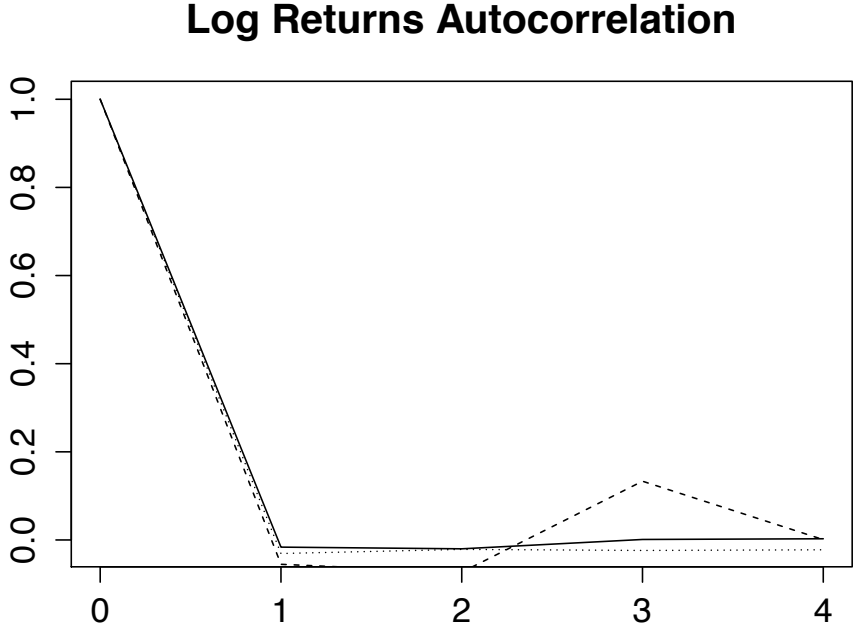

Figure 2 - *solid line = IBM, dashed line = CDA, dotted line = ASM*

As figure 3 implies the autocorrelation of the squared log-returns, which is one of the numerous volatility measures commonly used, displays the typical slow decay in time associated with volatility clustering.

Both models seem to successfully replicate this non-trivial feature, emphasizing how the price time series that have been generated, far from being simple white noise processes, can in fact display complex non-linear dependence in their log-returns.

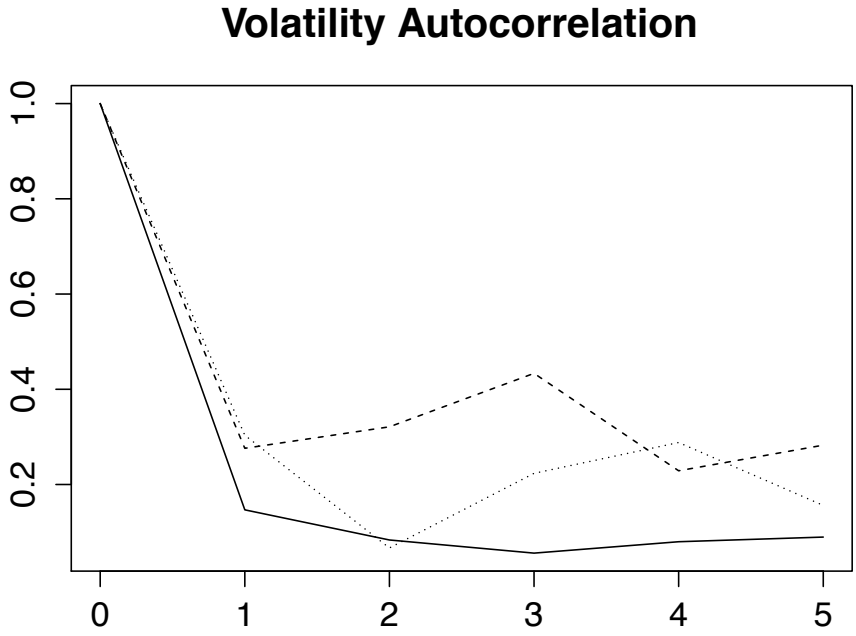

Figure 3 - *solid line = IBM, dashed line = CDA, dotted line = ASM*

In figure 4 are presented the autocorrelation functions of the volume series.

While the ASM model fares pretty well in this validation test, it is clear that the CDA model has a fundamental difficulty in replicating a positive slow decay in the autocorrelation function of volumes.

This is due to the extreme simplicity of the model, which assumes ZI agents as traders.

These agents actually keep trading under any market condition with a fixed probability.

The resulting volume series is pure noise, centered on a constant value.

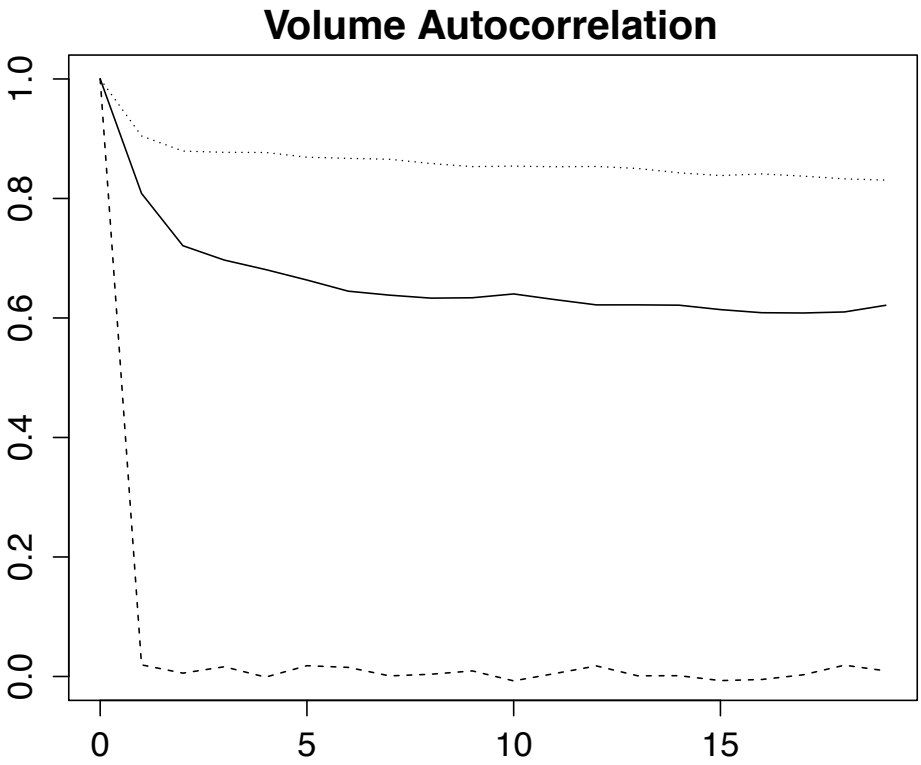

Figure 4 - *solid line = IBM, dashed line = CDA, dotted line = ASM*

This lack of market structure in the CDA model is also evident from the behavior of the volume-volatility cross-correlation function, in figure 5, which is, as expected, insignificantly different from zero at positive time lags.

The ASM model, on the other side, is able to replicate a significant cross-correlation.

As has been noted before [8] this result should be welcomed with considerable surprise: as is clear from equation (3) the demand is proportional to the inverse of the forecast variance σ_t^2, which is actually a proxy for the volatily at time t.

This would suggest that if any correlation exists, it ought to be negative. The ability to replicate a non-trivial positive correlation is therefore a good indication that the ASM model does indeed capture some relevant feature of how real markets work.

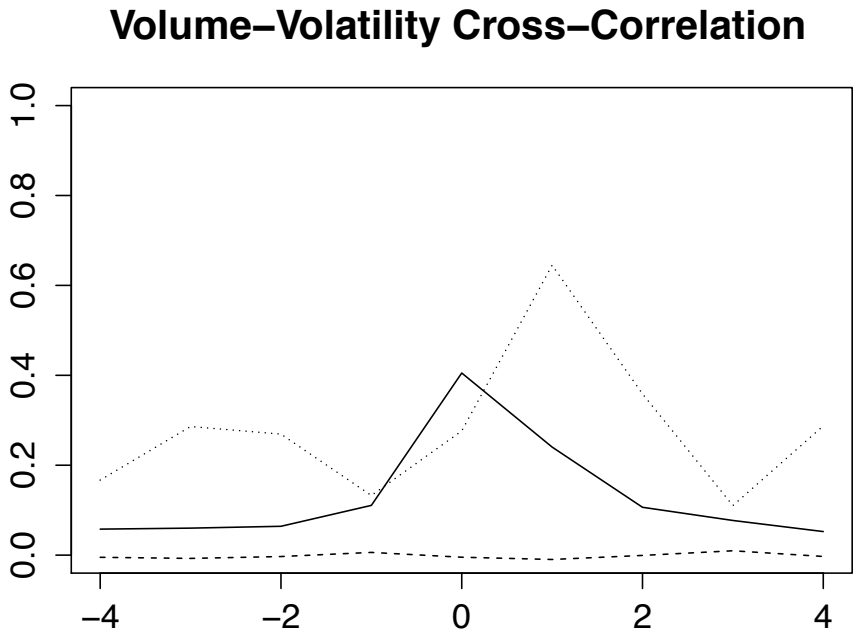

Figure 5 - *solid line = IBM, dashed line = CDA, dotted line = ASM*

Another interesting phenomenon that seems to be at least moderately replicated by these models is the leverage effect.

As shown in figure 6 the cross-correlation functions between volatility and returns present a significant negative correlation at some non-negative time lags.

Despite the highly "bumpy" profile of the functions, due to lack of sufficient samples, and striking disagreements with the IBM benchmark at low time lags, a significant negative trend can be observed at most positive time lags.

Given the extreme difficulty in replicating these third order statistical properties, this result can indeed be considered satisfactory.

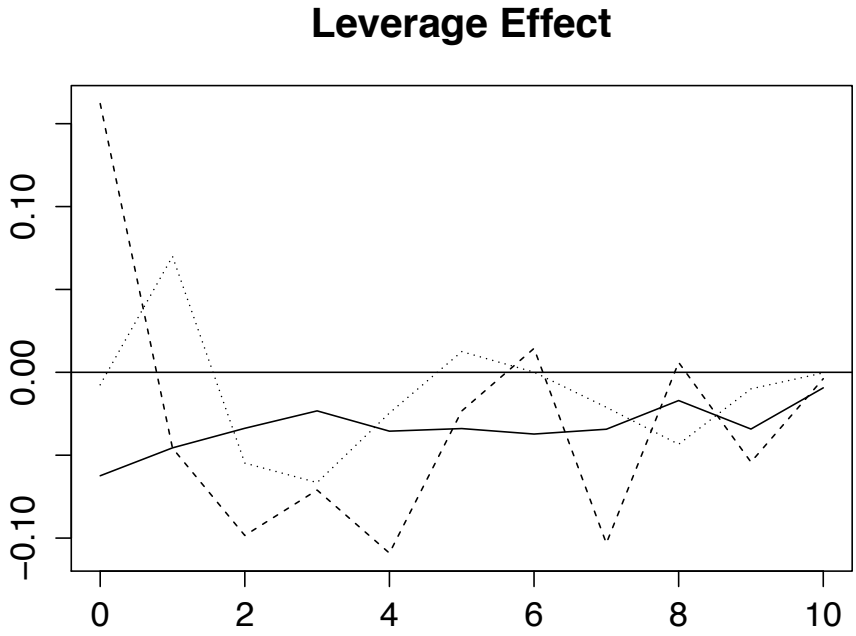

Figure 6 - *solid line = IBM, dashed line = CDA, dotted line = ASM*

In table 1 are reproduced some summary statistics for the log-returns time series of both models, compared with the benchmark IBM time series.

Statistics	IBM	ASM	CDA
ρ_1	-0.016	-0.019 ± 0.012	-0.026 ± 0.065
ρ_1^2	0.15	0.26 ± 0.13	0.44 ± 0.06
Excess kurtosis	12.79	5.04 ± 2.86	73.58 ± 21.68
Rejection of no-ARCH	100%	100%	100%
Excess kurtosis in ARCH(1) residuals	553.24	81.97 ± 56.77	2034 ± 1329

Table 1 - *Summary statistics of log-returns time series, averaged over 5 simulations of each model. Uncertainties are given by the standard deviations*

In the first row the autocorrelation coefficient of log-returns at the first time lag is computed. It is insignificantly different from zero, as was clear from figure 2.

The ρ_1^2 coefficient is the autocorrelation coefficient at the first time lag for the squared log-returns time series. As volatility clustering demands, it is significantly positive, despite higher deviations in the ASM case among the 5 simulations performed.

The excess kurtosis is a measure of how leptokurtic is the log-returns unconditional distribution. From both cases a positive excess kurtosis is evidence of a heavy tail distribution. In the ASM case the kurtosis is not extremely pronounced, while in the CDA case it is both positive and large. Different financial assets show different excess kurtoses. In particular the CDA case is compatible with stock indeces, like the S&P500, while the ASM model is closer to typical industrial stocks, like BMW.DE.

A test for heteroskedasticity has been performed, on the lines of the ARCH test proposed by Engle[10]. Squared log-returns have been fit to an ARCH model on two lags and all simulations reject the null-hypothesis, which means that all time series display typical ARCH effects at a confidence level of 95%.

The last row shows excess kurtosis in the distribution of the residuals of an ARCH(1) fit of squared log-returns. This phenomenon, known as conditional heavy tails, shows that the leptokurtic distribution is remarkably robust even after volatility clustering has been accounted for.

A last interesting feature that agrees with real financial time series is offered by the autocorrelation function of the `exePrice` list in the CDA model (see figure 7).

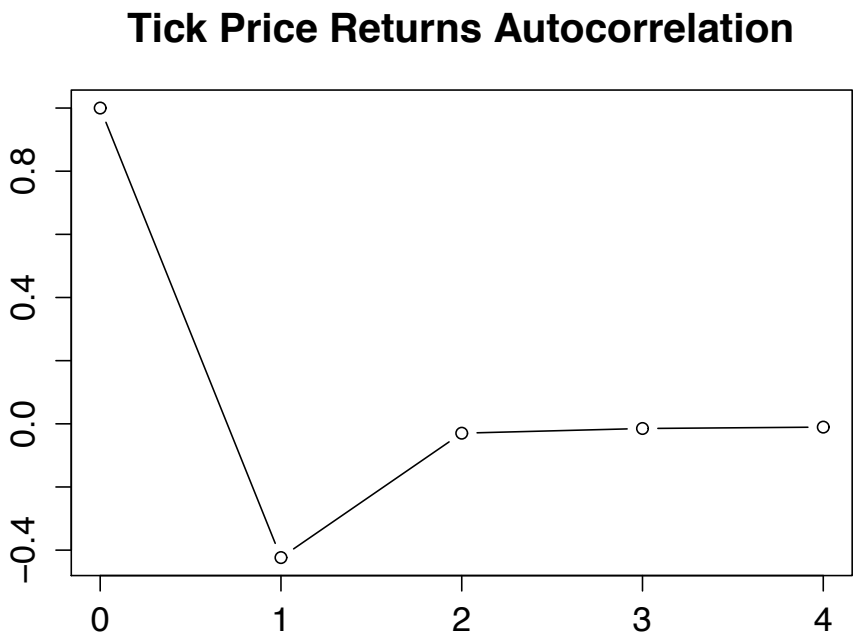

Figure 7 - *Autocorrelation of the tick-prices from the CDA model*

This time series shows the tick prices from one transaction to the next in

the CDA model. The characteristic negative coefficient at the first time lag is typical of fine-scale real financial time series and has come to be known as the *bid-ask bounce* effect[1].

At ever finer scales the price "bounces" between the ask and bid limit prices, typically jumping from one to the other at every transaction. This leads to a strong anticorrelation between two successive prices. This feature is successfully replicated by the CDA model. The ASM model, on the other hand, reproduces only a daily price series and is therefore unable to reproduce intraday trading properties like this one.

4 Conclusions

We have presented a general introduction to the fields of Financial Time Series Analysis and Agent-Based Financial Economics, with particular attention to the problem of replicating real financial statistical features by means of simulated artificial markets.

Two models, the ASM model and the CDA model, have been implemented in NetLogo and their relevant time series have been reported and analysed in R.

A remarkable replication of most real statistical properties has been proved and all differences in performance between the two models have been explained in terms of underlying modeling assumptions, leading to the conclusion that the two models are best suited to replicate different financial assets.

The ASM model presents a rich trading structure, allowing for realistic volume time series and log-returns leptokurtic distributions of positive but low excess kurtosis. This model is therefore best suited to replicate industrial stocks, for which daily time series of both price and volume are relevant.

The CDA model fails at replicating volume-related statistics, but shows large positive excess kurtosis and successful replication of intraday trading phenomena. It is therefore ideal for modeling financial indeces and any financial asset for a which realistic intraday trading time series is particularly needed.

References

[1] Rama Cont, *Empirical Properties of Asset Returns: Stylized Facts and Statistical Issues*. Quantitative Finance, vol. 1 (223-236), 2001

[2] C. M. Macal and M. J. North, *Tutorial on Agent-Based Modeling and Simulation*. Journal of Simulation, vol. 4 (151-162), 2010

[3] Blake LeBaron, *A Builder's Guide to Agent-Based Financial Markets*. Quantitative Finance, vol. 1 (254-261), 2001

[4] Brian Arthur, *Complexity in Economic and Financial Markets*. Complexity, vol. 1, n. 1, 1995

[5] Blake LeBaron, *Agent-Based Computational Finance: Suggested Readings and Early Research*, October 1998

[6] W. Brian Arthur, John H. Holland, Blake LeBaron, Richard Palmer and Paul Tayler, *Asset Pricing Under Endogeneous Expectations in an Artificial Stock Market*, December 1996

[7] Blake LeBaron, *Building the Santa Fe Artificial Stock Market*, June 2002

[8] Blake LeBaron, *Time Series Properties of an Artificial Stock Market*, Journal of Economic Dynamics & Control, n. 23 (1487-1516), 1999

[9] Jan C. Thiele, Winfried Kurth and Volker Grimm, *Agent-Based Modelling: Tools for Linking NetLogo and R*, Journal of Artificial Societies and Social Simulation 15 (3) 8, 2012

[10] Engle, R.F., *Autoregressive conditional heteroskedasticity with estimates of the variance of united kingdom inflation*, Econometrica 50, (987-1007), 1982