ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

Agent Based Simulation of a Financial Market

Samuel Peter, 05-909-122

Zurich
May 2010

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen-Hilsmittel verwenden habe, und alle Stellen, die wörtlich oder sinngemäss aus veröffentlichen Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Arbeit nicht, auch nicht auszugsweise, bereits für andere Prüfungen ausgefertigt wurde.

Samuel Peter

Agreement for free-download

I hereby agree to make my source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, i assure that all source code is written by myself and is not violating any copyright restrictions.

Samuel Peter

Preamble

Me, the author of this report, would like give a few explanations about my background and personal motivations to come up with the chosen topic: As a participant of the master degree program of civil engineering with one of the majors in *Water Resources Management*, I am often confronted with rainfall and discharge data showing typical behavior of power law distribution. There are several links to other different branches in natural sciences concerning the issue of complexity and criticality in many-body systems [1]. In the last two years I have had some insights into the financial community[1] and got more interested into the properties of financial markets inspired by the work of Mandelbrot [9] and Taleb [15]. This course (*Lecture with Computer Exercises: Modelling and Simulating Social Systems with Matlab*©) gave me the unique opportunity to apply the knowledge and deepen the understanding of financial theory.

Absract

An artificial financial market has been modeled based on a microscopic mechanism for price formation of matching demand and supply, ordered by heterogeneous agents. The model assumes a fixed number of agents trading one single asset. The agents differ in the way they judge the current market risk. For liquidity purpose a market maker was introduced who holds theoretically an infinite number of assets and cash. Opinion propagation was modeled with building clusters of traders who show collective behavior in case they get a specific information. The time series produced by this model show the typical properties of real markets, like volatility clustering, leptokurtic shape of the tail of the probability density function (fat tails) of the logarithmic price returns and exponents of the power-law corresponding to very liquid markets.

[1] I am grateful to Fabio Perlini, Lodovico Brioschi and Gabriele Mauri for their bright discussions during dinner time.

Contents

1 Introduction and Motivations

In the last twenty years numerous computer-simulated, artificial financial markets have been built. Several researchers have proposed artificial markets populated with heterogeneous agents endowed with learning and optimization capabilities. These markets exhibit behavior close to real markets but are often too complex. Markets populated with heterogeneous agents characterized by simple trading rules lend themselves better to analytical modeling while still capturing fundamental features of market behavior [12]. This part of research is still in its infancy and evolves very rapidly [3].

The objective of this project was to build an artificial market, based on the compromise between simple generalization and a faithful representation of the known stylized facts of financial time series, i.e. volatility clustering and fat tails int the distribution of short-term returns. The developed market has its basis in the *Genoa market*, described in [12]. Agents trade one single asset. Buy and sell orders[2] are randomly placed by the traders. The key ingredient of the model is the price formation process of the market that is built around a mechanism for matching demand and supply. The realistic assumption of the finiteness of agent's resources is made. For liquidity purpose a market maker is introduced. Groups of traders are forced to act as pessimists or optimists under specific circumstances (what is referred to as opinion propagation).

In a first part the model is described, in a second the implementation in $Matlab^{©}$ is shown, the results of the simulation are shown and further discussed. At the end a short outlook of further improvements and developments is given.

2 Description of the Model

N traders are considered (ith trader denoted with the subscript i). Time is evolving in discrete steps Δt. The current time is denoted with t and the corresponding price is $p(t)$. $C_i(t)$ is the amount of cash, and $A_i(t)$ is the amount of assets hold by the ith trader at time t. At each time step, each trader issues a buy order with probability P_i or a sell order with probability $1 - P_i$.

2.1 Sell and Buy Orders

Assuming the ith trader issues a sell order at time $t + 1$,

$$a_i^s = [r_i A_i(t)] \tag{1}$$

then is the quantity of assets offered for sale by the ith trader and is assumed to be a random fraction of the number of assets hold at time t. r_i in (1) is a random draw

[2] To avoid any confusions, the expressions *buy* and *sell* are to understand from the trader's view, i.e. a *buy* order means that the trader is buying and the client is selling.

from a uniform distribution in the interval [0,1] and the symbol $[x]$ stands for the integer part of x. To each sell order a limit sell price

$$s_i = p(t)LN_i(\mu_s, \sigma_i) \tag{2}$$

is associated, where $LN_i(\mu_s, \sigma_i)$ is a random draw from a lognormal distribution with an average μ_s and a standard deviation σ_i. Sell orders at prices below the limit price cannot be executed.
The traders uncertainty on asset market prices grows in nervous (i.e. volatile) markets [12]. This is expressed in

$$\sigma_i = k\sigma(f_i), \tag{3}$$

where k is a constant factor and f_i is the factor for the calculation of the individual volatility of the ith trader. Each trader shows different risk behavior and will judge the current market condition in a specific way. The individual volatility $\sigma(f_i)$ is evaluated with

$$\sigma_t^2(f_i) = \sum_{j=1}^{m} \alpha_j u_{t-j}^2, \tag{4}$$

where α_j is the weight which is attached to the return $u_t = ln\frac{p(t)}{p(t-1)}$ [16] j days before. The weighing values α_j in the model are set as a *exponentially weighted moving average, EWMA*. In that case it holds $\alpha_{j+1} = e^{-f_i}\alpha_j$. A value e^{-f_i} close to 0 indicates that recent events are of big interest whereas a value close to 1 leads to judgements for which new informations out of daily changes in the market variables have a small impact [6].

Buy orders are generated in a symmetrical way with respect to sell orders. $c_i = r_i C_i(t)$ denotes the amount of cash the ith trader issues, which is a random fraction of his available cash at time t. The associated limit price for buy orders is $b_i = p(t)LN_i(\mu_b, \sigma_i)$. The quantity of assets ordered to buy is given by $a_i^b = [\frac{c_i}{b_i}]$. Buy orders at prices that are higher than the limit price cannot be executed.

The averages for the random lognormal draw are defined as $\mu_s = -\frac{\Delta\mu}{2}$ and $\mu_b = \frac{\Delta\mu}{2}$, where $\Delta\mu$ is the spread on the market for limit prices. This implies the fact, that for a buy order the trader is likely to be willing to pay more than $p(t)$, conversely for sell order the trader is likely to offer the asset at a lower price than $p(t)$. In other words the traders place the orders in such a way that the chance of the order being executed is increased [12].

2.2 Price Formation Process

In figure 1 the demand and supply curve are shown out of a simulation run. The clearing price is set as the intersection of demand and supply.
Let's assume that the traders have issued U buy orders and V sell orders. Then the

demand function at time $t+1$ is defined as

$$D_{t+1}(p) = \sum_{u|b_u \geq p}^{U} a_u^b, \tag{5}$$

representing the total amount of assets that would be bought at price p. The supply function is defined in the similar way representing the number of assets that would be sold at price p as

$$S_{t+1}(p) = \sum_{v|s_v \leq p}^{V} a_v^s. \tag{6}$$

The clearing price p_c is the price at which the demand and supply curve cross. Due

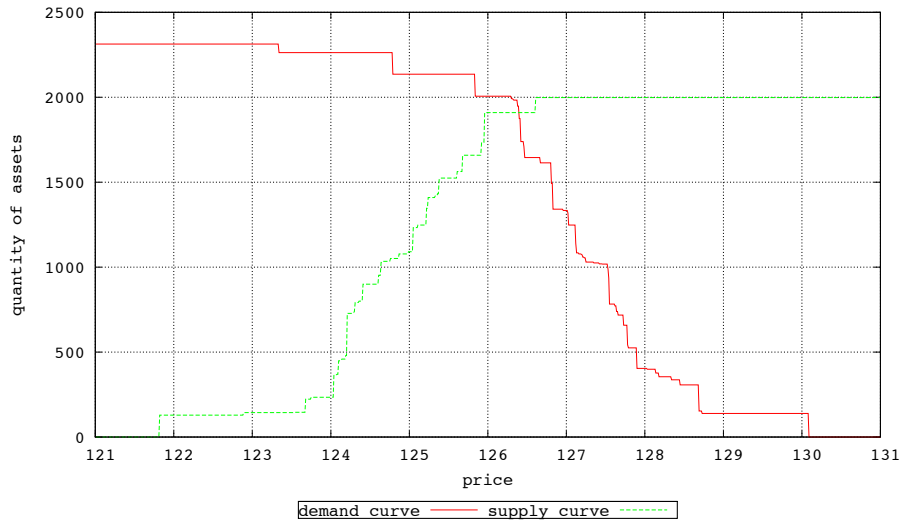


Figure 1: Example of demand curve and supply curve evaluated for price formation process at any simulation step.

to the fact that $D(p)$ and $S(p)$ are step functions, generally the amount assets $S(p_c)$ and $D(p_c)$ are not exactly the same. In order to satisfy all compatible orders, the difference between the two quantities is absorbed by the market maker who holds theoretically an infinite number of assets and amount of cash. Orders, whose limit prices are not compatible with the value of the new price are discarded [11]. After each formation of a new price, all compatible orders are executed, i.e. the cash and number of assets of the traders and the market maker are updated.

2.3 Opinion Propagation among Traders

The model described so far, where the traders are absolutely independent, produces a normal distribution of the returns, i.e. the log-price behavior is a random walk [13].

It is well known that distributions of real market returns show fat tails following a Pareto-Levi distribution [10, 3]. This was modeled by introducing clusters of traders sharing the same opinion, i.e. being optimist or pessimist respectively. At each simulation step two traders are randomly chosen to build a bond with a probability P_a and sharing the same opinion from now on. If a link is established between two traders belonging to different clusters, the two clusters merge. In this way clusters of traders gradually take shape. In addition, at each time step a cluster is randomly chosen with probability P_c. All traders belonging to this cluster receive an inside information to buy (for optimists) or to sell (for pessimists) with a probability of 0.5. In the case they are told to buy, they invest 90% of their actual cash, conversely if they are told to sell, they place 90% of the assets they hold. After such an aggregate order was placed, all links between the traders belonging to the chosen cluster are deleted and the traders act as before [11]. In this way the opinion formation process is described in the model.

3 Implementation

In this section the implementation in *Matlab*© of the model described above is explained. The conceptional design of the model is shown in figure 2. The same architecture can be seen in the code of the m-file *main.m* in figure 3. All the mentioned subroutines will be described further, except the analysis of the generated data, what *gnuplot*©[3] was used for.

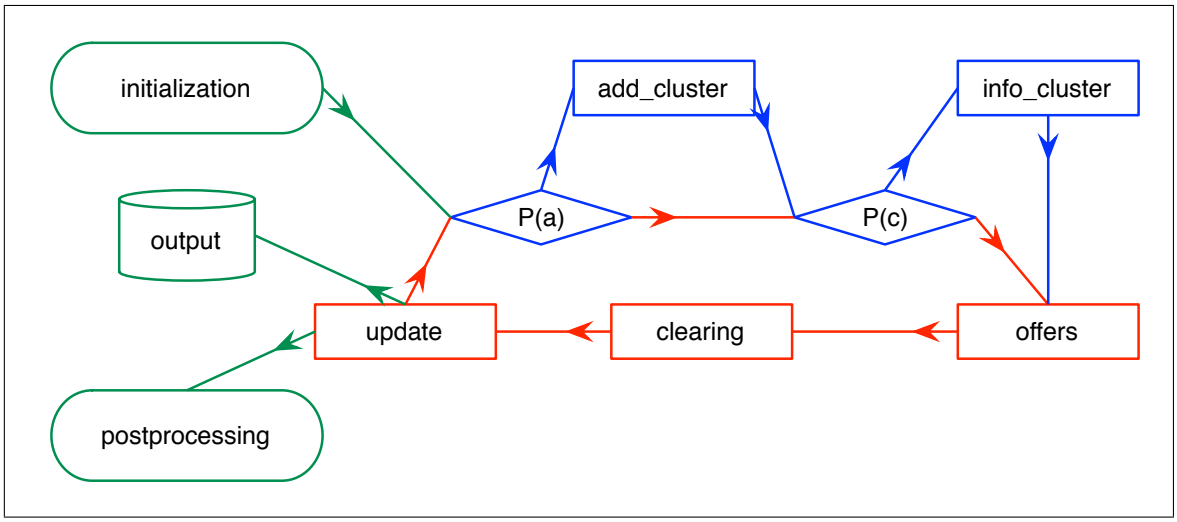


Figure 2: Conceptional scheme of implemented multi-agent model with its subroutines. green: link to the user, red: loop over one simulation step, blue: cluster building process.

[3] `http://www.gnuplot.info`: freely distributed but copyrighted source code.

```
%%%%% number of time steps %%%%%
N_t = 100000;
identifier = 7;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% initialization
run initialization;

% output file
filename = sprintf('N%d_Pa%1.2f_Pc%1.2f_data_%d.txt',N,P_a,P_c,identifier);
datafile = fopen(filename,'w');
fprintf(datafile,'# time\tprice\treturn\tvolatility\tA(marketmaker)\tC(marketmaker)\tcluster\n');

for time=2:N_t+1

    % form bond between two traders
    if rand()<P_a
        run add_cluster;
    end

    % set information to cluster
    if rand()<P_c
        run info_cluster;
    end
    sum_cluster = sum(info);

    % one time step
    run offers;
    run clearing_mm;
    run update;

    % model output
    fprintf(datafile,'%d\t%4.4f\t%1.4f\t%1.4f\t%d\t%d\t%d\n',...
        time,p(time),log_return(time),volatility(time),A_mm(time),C_mm(time),sum_cluster);
end
fclose(datafile);
```

Figure 3: Code of the main m-file, where the different subroutines of the model are put together.

3.1 Model Initialization

In the m-file *initialization.m* (figure 4) all parameters of the model can be defined. These are the number of traders N, the probability of placing a buy order P_i (sell order $1 - P_i$ respectively), the amount of cash and the number of assets hold per trader at initial time, the spread in the average limit prices of buy and sell orders, the time window T for the evaluation of the historical volatility, the minimum and maximum of the factor for the calculation of the exponentially decade of the weighing values α_j in (4), and the parameters for the opinion propagation process P_a and P_c. The initial price is set as the total initial cash in the system divided by the total number of assets. Some arrays with constant size are allocated as well (see figure 4).

3.2 Buying and Selling Offers

In each simulation step the offers for each trader (buy or sell) with the corresponding limit prices are generated in the m-file *offers.m* (figure 5). In a first loop for each trader the standard deviation $\sigma_t(f_i)$ is calculated for the random lognormal draw (see (2)) dependent on the individual risk aversion. In a second loop for each trader the individual offers are generated.

```
% number of traders
N = 100;

% amount of cash per trader at initial time
cash = 10000;

% amount of assets hold per trader at initial time
assets = 100;

% spread in limit prices
spread = 0.02;

% number of bins in price functions
N_p = 1000;

% time window for historical volatility
T = 30;

% factor for exponential decade of volatility dependent decisions
f_max = 0.9;
f_min = 0.5;
f = zeros(N,1);
for i=1:N
    f(i) = f_min+ceil(rand()*f_max*100)/100;
end

% set variables fro clustering
P_a = 0.1;
P_c = 0.05;
ratio = 0.9;
cluster_id = zeros(N,1);
id_free = 1:ceil(N/2);

%initializing other variables
p(1:2,1) = cash/assets;    % initial price
log_return(1:2,1) = 0;  % first value of return
volatility(1:2,1) = 0.02; % initial volatility
C_mm(1:2,1) = 0;    % initial cash for ideal market maker
A_mm(1:2,1) = 0;    % initial assets for ideal market maker
x_vola = T:-1:1;
sigma = zeros(N,1);
weights_vola = exp(-f*x_vola);
mu_buy = spread/2; % expected value of gaussian draw for bids
mu_sell = -spread/2; % expected value of gaussian draw for asks
k = 3.5;    % constant for limit prices
P = 0.5;    % probability to issue a buy order
C(1:N,1) = cash;   % cash at time t
A(1:N,1) = assets;   % assets at time t
info = zeros(N,1);
decision = false(N,1);  % (1->buy, 0->sell)
```

Figure 4: Code of the m-file for initialization, where all parameters of the model have to be set up.

3.3 Clearing Price

Out of the buy and sell orders with their according limit prices generated in *offers.m* the demand and supply curves are calculated with (5) and (6) (see m-file *clearing.m* in figure 6). The clearing price is defined at the intersection of the two curves. If there does not exist a matching point, all orders are discarded and a new configuration of orders will be established.

3.4 Update for all Variables

After a matching price for a given configuration was found, all compatible orders are executed. The cash and number of assets hold per trader are updated according to their orders. The difference between the total of assets being bought and sold is absorbed by the market maker. In that way a certain variability in the number of traded assets in the system cannot be avoided. The time series $p(time)$, $log\_return(time)$ and $volatility(time)$ are updated as well.

```
% present values
price = p(time-1);
if time>T
    for i=1:N
        % ARCH-model
        sigma(i) = sqrt(sum(dot(weights_vola(i,:),...
            (log_return(time-T:time-1)).^2))/sum(weights_vola(i,:)));
    end
else
    for i=1:N
        sigma(i) = volatility(time-1);
    end
end

% odering sell / buy offers
a_sell(1:N,1) = zeros(N,1);
s(1:N,1) = zeros(N,1);
c(1:N,1) = zeros(N,1);
b(1:N,1) = zeros(N,1);
a_buy(1:N,1) = zeros(N,1);
for i=1:N
    % trader does not belong to cluster
    if info(i)==0
        if rand()<=P
            decision(i) = 1;
            c(i) = rand()*C(i);
            lognormal_draw = random('logn',mu_buy,k*sigma(i));
            b(i) = price*lognormal_draw;
            a_buy(i) = floor(c(i)/b(i));
        else
            decision(i) = 0;
            a_sell(i) = floor(rand()*A(i));
            lognormal_draw = random('logn',mu_sell,k*sigma(i));
            s(i) = price*lognormal_draw;
        end
    % trader belongs to optimistic cluster
    elseif info(i)==1
        decision(i) = 1;
        c(i) = ratio*C(i);
        b(i) = price;
        a_buy(i) = floor(c(i)/b(i));
    % trader belongs to pessimistic cluster
    elseif info(i)==-1
        decision(i) = 0;
        s(i) = price;
        a_sell(i) = floor(ratio*A(i));
    end
end
```

Figure 5: Code of the m-file which generates the buy and sell orders and their corresponding limit prices for each trader.

3.5 Cluster Process

At each simulation step two traders are randomly chosen with probability P_a and a permanent link is established between them, meaning they interact in a certain way (e.g. they talk to each other or they have both access to the same information, let's say they share a specific opinion about the market behavior). Big clusters are built up by merging different already existing clusters. The corresponding m-file *add_cluster.m* is shown in figure 8. The array *cluster_id(i)* contains the cluster identifier, which the ith trader belongs to, or zero if trader i does not belong to any cluster.

With a probability of P_c each time step a cluster is randomly chosen independent of its size. All traders belonging to this cluster act as optimists or pessimists for one time step. They place order with 90% of their actual cash or number of assets respectively. The probability to buy or to sell is with P_i the same as it is for one single trader. The information for a buy or sell order is stored in the array *info* (see figure 9). After the execution the linked traders are released and get a new risk behavior by allocating a new factor f_i (cp. (3)).

```
% demand and supply curve
max_price = ceil(max([s;b]));
min_price = floor(min([s(s~=0);b(b~=0)]));
dp = (max_price-min_price)/N_p;
demand = zeros(N_p,1);
supply = zeros(N_p,1);
price = zeros(N_p,1);
for i=1:N_p
    p_control = min_price+i*dp;
    price(i) = p_control;
    demand(i) = sum(a_buy(b>=p_control));
    supply(i) = sum(a_sell(s<=p_control));
end

% evaluating clearing price
if sum(demand<supply)==0
    run offers;
    run clearing_mm;
end
if sum(demand==supply)>=1
    index = demand==supply;
    price_clear = mean(price(index));
else
    index = supply<demand;
    price_clear = (price(sum(index))+price(sum(index)+1))/2;
end
```

Figure 6: Code of the m-file which builds the demand and supply curve and figures out the clearing price.

```
% update for price
p(time) = price_clear;
log_return(time) = log(p(time))-log(p(time-1));

% update for traders
for i=1:N
    if (decision(i)==0) && (s(i)<=price_clear)
        A(i) = A(i)-a_sell(i);
        C(i) = C(i)+a_sell(i)*price_clear;
    elseif (decision(i)==1) && (b(i)>=price_clear)
        A(i) = A(i)+a_buy(i);
        C(i) = C(i)-a_buy(i)*price_clear;
    end
    if info(i)~=0
        % new risk aversion
        f(i) = ceil(rand()*f_max*100)/100;
        weights_vola(i,:) = exp(-f(i)*x_vola);
        info(i) = 0;
    end
end

%update for market maker
corr = sum(a_sell(s<=price_clear))-sum(a_buy(b>=price_clear));
A_mm(time) = A_mm(time-1)+corr;
C_mm(time) = C_mm(time-1)-price_clear*corr;

% update for volatility
if time>T
    sigma = std(log_return(time-T:time));
else
    sigma = volatility(time-1);
end
volatility(time) = sigma;
```

Figure 7: Code of the m-file for updating cash and number of assets of all traders and the market maker.

```
% add randomly two traders sharing the same behavior
trader1 = round(rand()*(N-1))+1;
trader2 = round(rand()*(N-1))+1;
if (cluster_id(trader1)==0) && (cluster_id(trader2)==0)
    id = min(id_free);
    cluster_id(trader1) = id;
    cluster_id(trader2) = id;
    id_free(id_free==id) = [];
elseif (cluster_id(trader1)~=0) && (cluster_id(trader2)==0)
    cluster_id(trader2) = cluster_id(trader1);
elseif (cluster_id(trader1)==0) && (cluster_id(trader2)~=0)
    cluster_id(trader1) = cluster_id(trader2);
elseif (cluster_id(trader1)~=0) && (cluster_id(trader2)~=0)
    if cluster_id(trader1)>cluster_id(trader2)
        id = cluster_id(trader1);
        id_old = cluster_id(trader2);
        cluster_id(cluster_id==id_old) = id;
        id_free(length(id_free)+1) = id_old;
    elseif cluster_id(trader2)>cluster_id(trader1)
        id = cluster_id(trader2);
        id_old = cluster_id(trader1);
        cluster_id(cluster_id==id_old) = id;
        id_free(length(id_free)+1) = id_old;
    end
end
```

Figure 8: Code of the m-file that is executed with a probability of P_a: Two traders are randomly chosen and bonded to each other.

```
% set information to specific cluster (-1 -> pessimistic; +1 -> optimistic)
id_occ = [];
count = 0;
for i=1:ceil(N/2)
    if sum(id_free==i)==0
        count = count+1;
        id_occ(count) = i;
    end
end
if isempty(id_occ)~=1
    id = id_occ(floor(rand()*count)+1);
    info(cluster_id==id) = round(rand())*2-1;
    cluster_id(cluster_id==id) = 0;
    id_free(length(id_free)+1) = id;
end
```

Figure 9: Code of the m-file which is executed with a probability of P_c: All traders belonging to a randomly chosen cluster are set to as pessimistic or optimistic .

4 Simulation Results and Discussion

In this section the results are presented from a simulation run over 100'000 simulation steps. The results are shown in form of different time series, the way distribution functions look like, and by having a closer look at the scaling behavior of the price trajectory.

4.1 Parameter and Sensitivity Analysis

First an investigation about the sensitivity of the different parameters is made. The amount of cash and assets in the system can be chosen arbitrarily, since the data analysis is made based on returns. The Parameters that control the fluctuations in the limit prices (k_i, f_i, μ_i) do not influence the time series in a coarse way. They are set either as realistic values or out of literature. The number of traders N would be as large as possible to simulate realistic circumstances of the market. Due to performance lack of *Matlab*© the number of agents could not be chosen as high as it was wished for. The only parameters the model can be calibrated with, are the two parameters controlling the opinion propagation among the traders P_a and P_c. Those are chosen such that the time series and statistical behavior produced by the model represent realistic data. Since these two parameters are dependent on the number of traders, only one configuration with $N = 100$ has been evaluated. The parameter set for the further presentation of the results can be seen in table 1.

N	100	number of traders
$cash$	10'000	initial cash hold by each trader
$assets$	100	initial number of assets hold by each trader (following initial price $p(0) = \frac{cash}{assets} = 100$)
P_i	0.5	probability of place a buy order (sell order), for all traders
k_i	3.5	constant for standard deviation in limit price building, for all traders [12]
$spread$	0.02	spread of the average limit price for buy and sell orders [12]
P_a	0.1	probability that a bond is set between two traders
P_c	0.05	probability that a cluster is chosen to place optimistic or pessimistic orders
f_i	$[0.5, 0.9]$	interval within the individual factors f_i for each trader randomly is chosen[6]

Table 1: Chosen parameters of the simulation run used for presenting the results.

4.2 Timeseries

The below presented simulation run is one out of numerous other simulation runs with the same parameter set. They all show the same behavior and properties. The one used for further discussion is chosen randomly.

The variables used for a time series investigation are the raw price $p(t)$, the logarith-

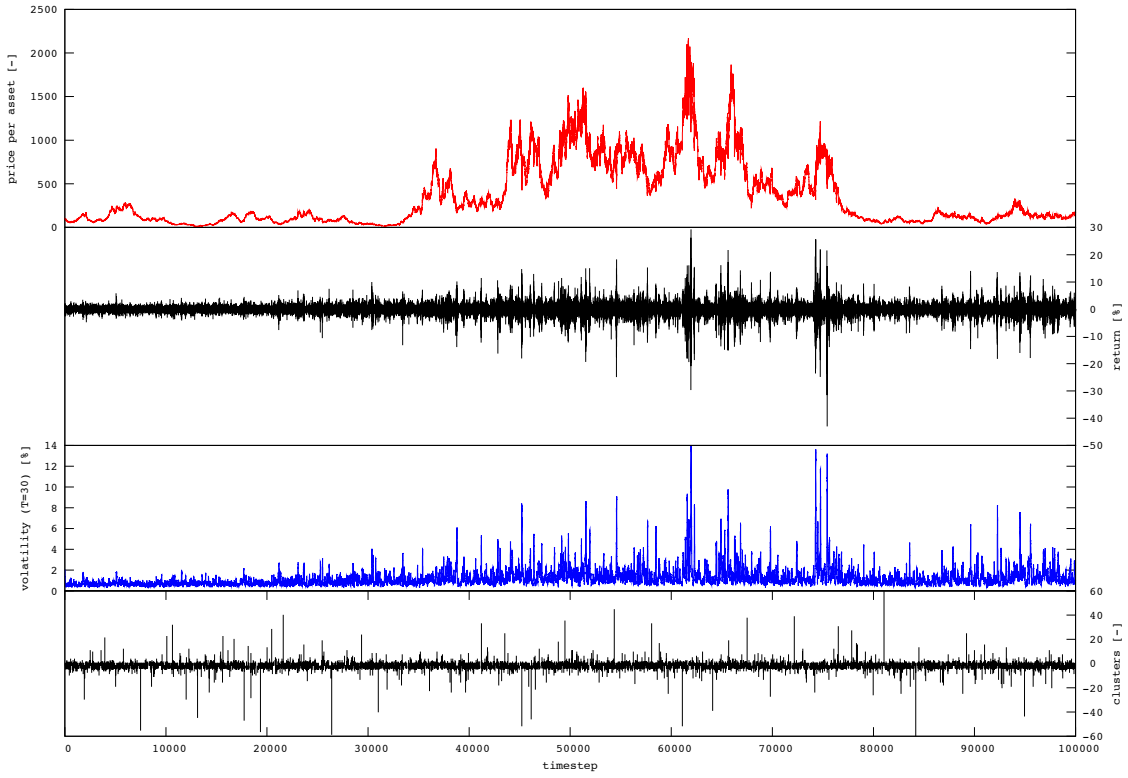


Figure 10: Simulation run over 100'000 time steps modeling 100 traders: the time series show the price evolution, the logarithmic returns, the simple historical volatility and the number of traders belonging to executed cluster (from top to down).

mic return $u_t = ln\frac{p(t)}{p(t-1)}$, the simple historical volatility measured in a time window $T = 30$ (cp. (4)) and the number of agents belonging to the activated clusters (see figure 10). The system evolves over time into different regimes. At the beginning (up to the first 20'000 simulation steps) the system relaxes into a regime where the measured volatility is very small and the observed returns look like gaussian white noise [14], although several big clusters have been activated. Between simulation steps 20'000 and 40'000 several periods with high volatility can be observed, i.e. volatility

clusters. This means that several time steps with a higher (lower) volatility tend to be followed by time steps again showing high (low) volatitilty [7]. The next 40'000 simulation steps show high fluctuations in the price trajectory. These big events are embed into clusters of high volatility and bring real data to mind as they are shown in [14, 9]. After that turbulent period the system shifts back to more calm region before some bigger volatility clusters can be seen again at the end of the 100'000 simulated time steps.
It has to be mentioned that over the whole time series only one variable stays of the same manner, namely the size of the activated pool of agents sharing the same optimistic (pessimistic) opinion. Thus it can be said the existence of the opinion propagation among traders is necessary for the system to reproduce realistic time series, but the activation of one single big cluster of traders is not responsible for the outliers in the price trajectory.

4.3 Statistical Analysis of Simulated Data

For all statistical investigations a few simulation steps at the beginning are not considered because there might be some abnormal properties due to the initialization. The system is assumed to be relaxed after 1'000 time steps.
First the frequency distribution of the logarithmic return was evaluated. Figure 11

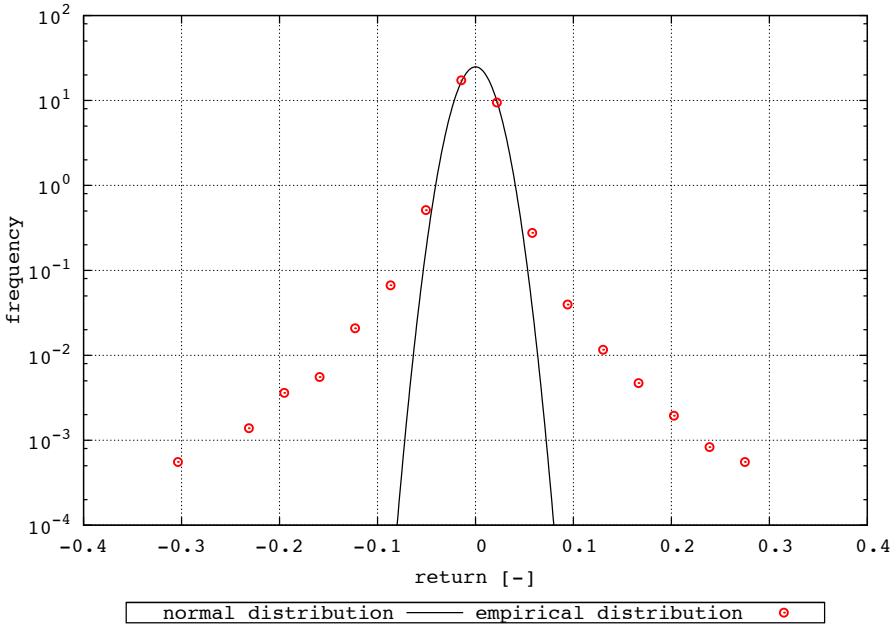


Figure 11: Comparison of the sample distribution with the according normal distribution.

shows the comparison between the empirical data and the theoretical distribution generated by a random walk having the same mean and standard deviation like the

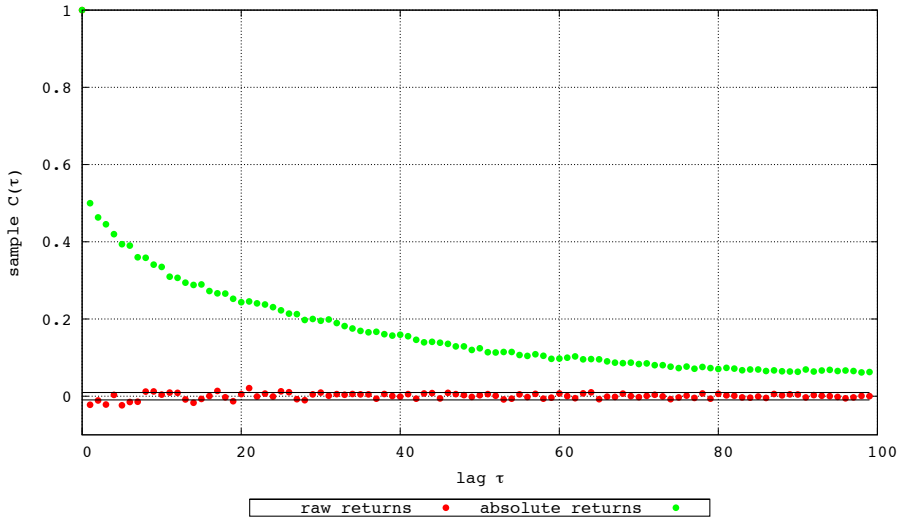


Figure 12: Autocorrelation function $C(\tau)$ of time lag τ for the raw returns and the absolute returns. The black lines represent the noise levels computed as $\pm\frac{3}{\sqrt{M}}$, where M is the length of the time series (100'000) [3].

empirical data (efficient market hypothesis, [14]). It clearly can be seen that the data generated by the model produces the so called *fat tails*. Even six-sigma events (referring to events which are six times the standard deviation) occur quite frequently (around 0.05) which would only occur once in three millions time steps based on the assumption of a gaussian distribution.

Figure 12 shows the autocorrelation function (ACF) of the time series of the returns (raw and absolute) produced by the model. The price trajectory is meant to follow a Markov Process, i.e. the future development of the price is only dependent on its current value. This is also embed in the theory of weak market efficiency, where the current price reflects all the available informations [6]. The sample ACF of the raw returns decades very rapidly to zero and fluctuates around the noise level. The few data points of lag 1-7 show a slight anticorrelation, which was found in data of real markets as well [3]. One cannot find significant correlation over any time lags. On the other hand the sample ACF of the absolute values of the returns shows the presence of long-range correlations with a very slow exponential decay. Taking the absolute returns as a measure of volatility, one can point out again the existence of the volatility clustering [12].

4.4 Scaling Behavior and Self Affinity

As there has been a big discussion ongoing over the last two centuries about self organization, criticality and scaling properties in financial markets [8, 9, 14, 1], the data generated by the model is investigated in order to detect some simple scaling properties.

In Figure 13 the probability that a specific return u_i was exceeded during one

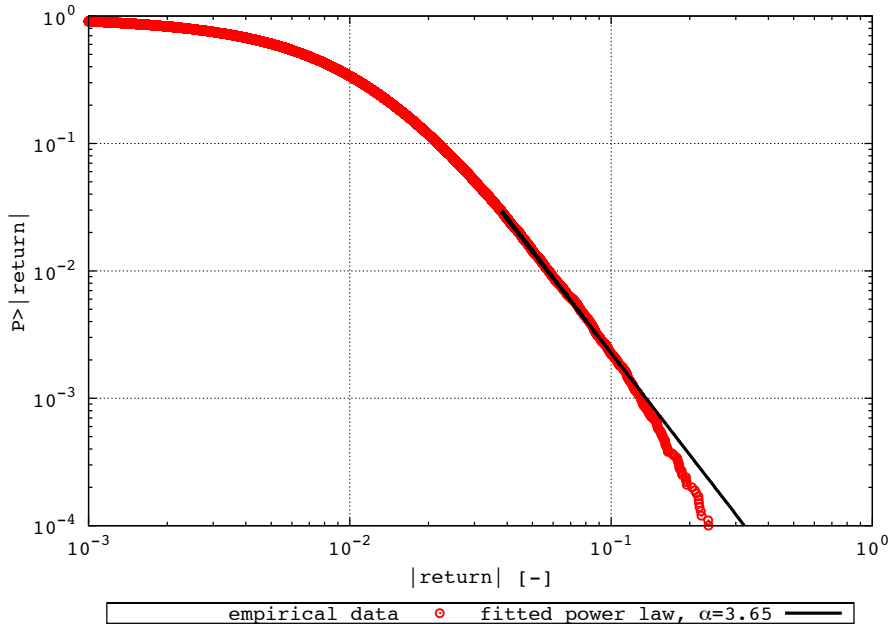


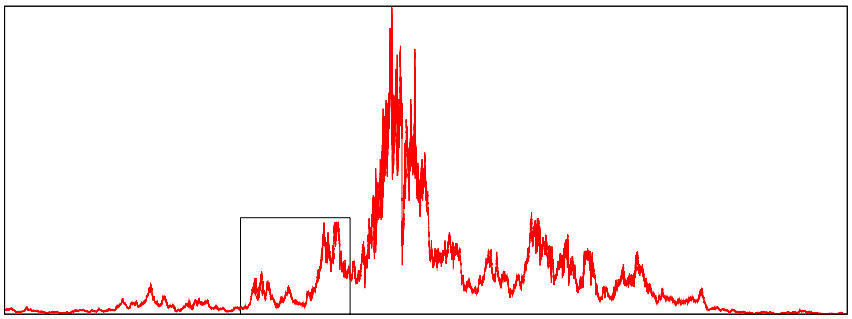
Figure 13: Probabilty $P(|u_i|)$ that a specific absolute return $|u_i|$ is exceeded.

simulation run is plotted in double logarithmic scale. Mathematically the power law is defined as

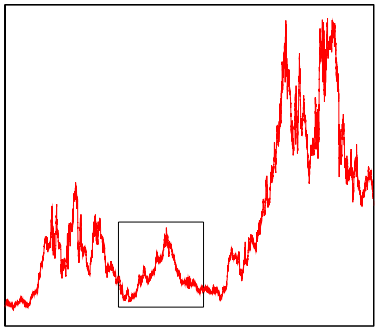
$$P(|u_i| > x) \propto x^{-\alpha}, \tag{7}$$

where α is a constant parameter of the distribution known as the *exponent* or *scaling parameter* [4]. In literature the values for the exponent varies between 2.5 and 4 [8, 12, 5, 3]. The data from the model on hand exhibits an exponent $\alpha = 3.65$, estimated with the method of most likelihood above a minimum return around 4%.

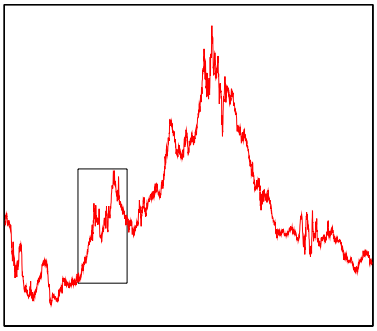
Referring to the work of Mandelbrot [9] the basis of the scaling behavior is the self affinity. In the case of financial data one can look at different time scales. This is shown in figure 14(a) -(e) where an investigation by eye can be established. It clearly shows that the way the absolute price curves look like, stays at the same manner for a wide range of the time scale. Without knowing the scales of the axis one is not capable to detect the true scale.



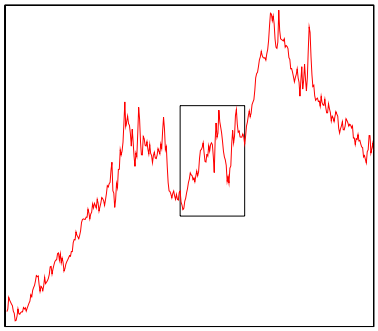
(a) whole simulation period (100'000 time stpes)



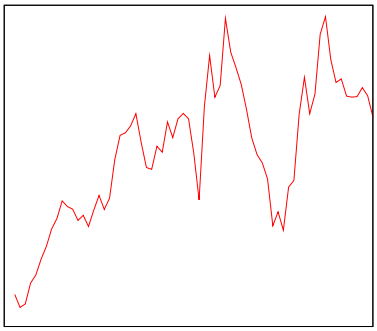
(b) 13'000 simulation steps



(c) 3'000 simulation steps



(d) 400 simulation steps



(e) 70 simulation steps

Figure 14: Different time windows of the asset price. The rectangle indicates the window of the next time scale. The axis are not labeled for comparing purpose.

5 Outlook

Since the model has been implemented in *Matlab*©, the number of agents that could be simulated is limited up to 100 due to performance lack. To investigate bigger systems one is forced to implement the model into a more powerful programming language (e.g. C++). A further amplification would be the option pricing based on the price trajectories the model produces and comparing it with the traditional pricing model introduced by Black & Scholes [2].

References

[1] P. Bak. *how nature works, the science of self-organized criticality*. Copernicus, Springer-Verlag, New-York, 1996.

[2] F. Black and S. Myron. The pricing of the options and corporate liabilities. *The Journal of Political Ecnonmy*, 81(3):637–654, 1973.

[3] J.-P. Bouchaud and M. Potters. *Theory of Financial Risks: From Statistical Physics to Risk Management*. Cambridge University Press, Cambridge, UK, 2000.

[4] A. Clauset, C. R. Shalizi, and M. Newman. Power-law distributions in empirical data. *SIAM Review*, 51:661–703, 2009.

[5] X. Gabaix, P. Gopikrishnan, V. Plerou, and H. E. Stanley. A theory of power-law distributions in financial market fluctuations. *Nature*, 423:267–270, May 2003.

[6] J. C. Hull. *Options, Futures, and other Derivatives*. New Jersey: Pearson Education, 7th edition, 2008.

[7] T. Lux and M. Marchesi. Volatility clustering in financial markets: A micro-simulation of interactive agents. Technical report, Departement of Economics, University of Bonn, 1998.

[8] T. Lux and M. Marchesi. Scaling and criticality in a stochastic multi-agent model of a financial market. *Nature*, 397:498–500, February 1999.

[9] B. B. Mandelbrot and R. L. Hudson. *Fraktale und Finanzen*. Piper Verlag GmbH, München, 2. edition, 2005.

[10] R. N. Mantegna and H. E. Stanley. *An Introduction to Econophysics. Correlations and Complexity in Finance*. Cambridge University Press, Cambridge, UK, 2000.

[11] M. Marchesi, S. Cincotti, S. M. Focardi, and M. Raberto. Development and testing of an artificial stock market. *Modelli Dinamici in Economia e Finanza*, September 2000.

[12] M. Raberto, S. Cincotti, S. M. Focardi, and M. Marchesi. Agent-based simulation of a financial market. *Physica A*, 299:319–327, 2001.

[13] R. U. Seydel. *Tools for Computational Finance*. Springer Verlag, Berlin, 3. edition, 2006.

[14] D. Sornette. *Why Stock Markets Crash. Critical Events in Complex Financial Systems*. Princeton University Press, Princeton, New Jersey, 5. edition, 2004.

[15] N. N. Taleb. *Narren des Zufalls, die verborgene Rolle des Glücks an den Finanzmärkten und im Rest des Lebens*. Wiley-VCH Verlag GmbH & Co., Weinheim, 1. edition, 2008.

[16] P. Wilmott. *Paul Wilmott on Quantitative Finance*, volume 1. John Wiley & Sons Ltd., West Sussex, 2000.