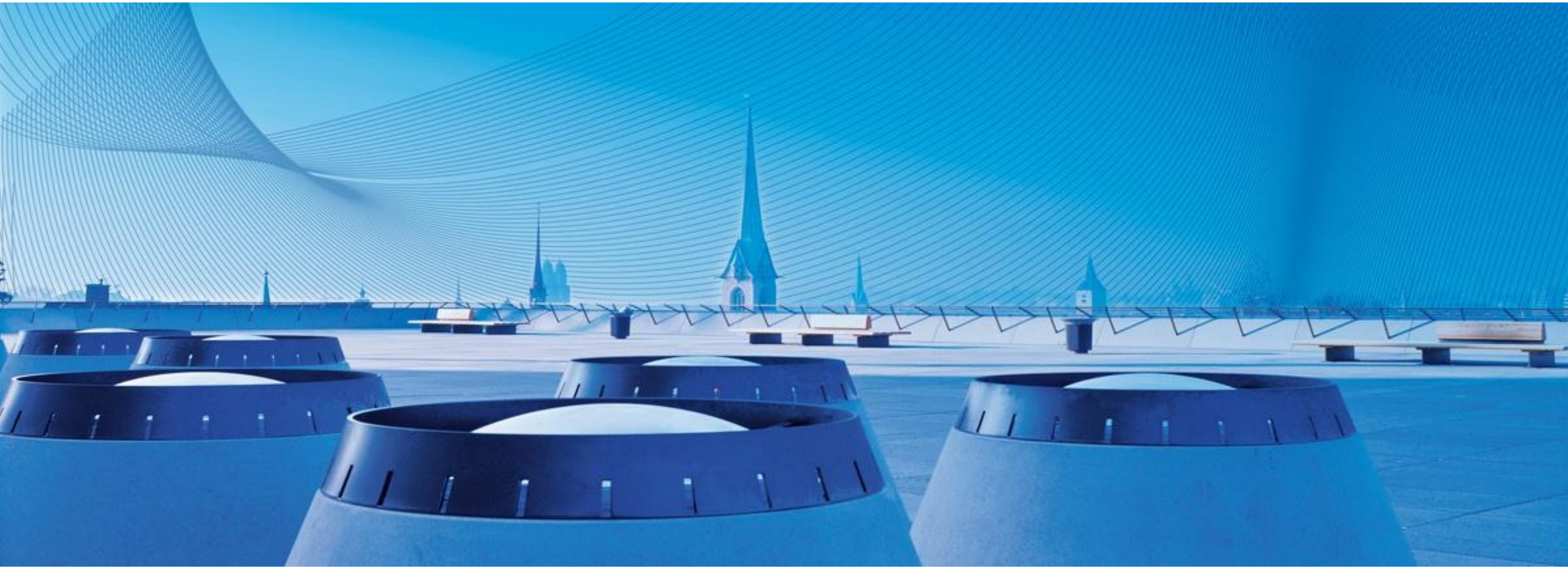
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

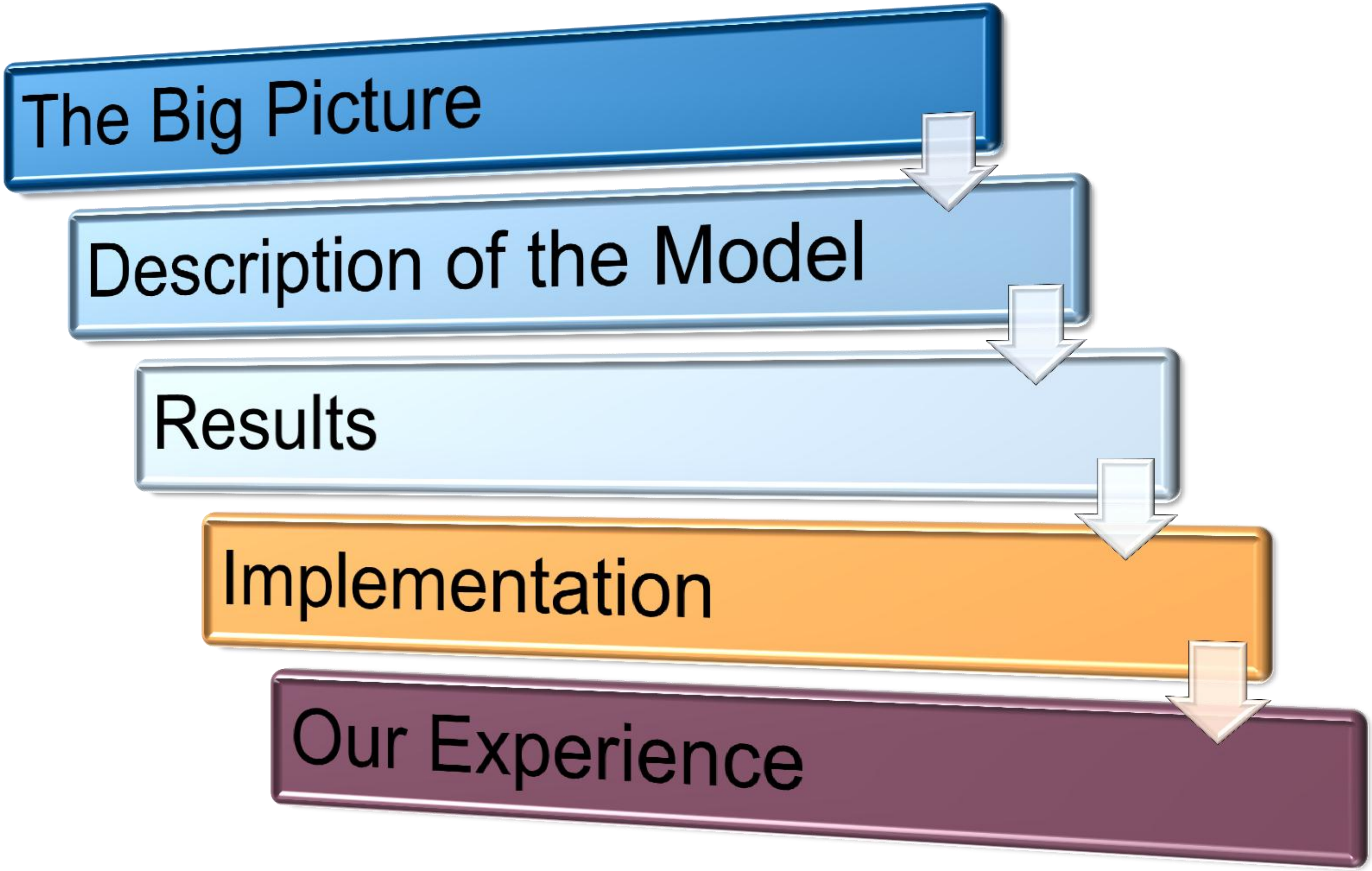
Departement Informationstechnologie
und Elektrotechnik

Simulation of Trading in an artificial Stock Market

Nicholas Eyring Youri Popoff



Contents



The Big Picture

Description of the Model

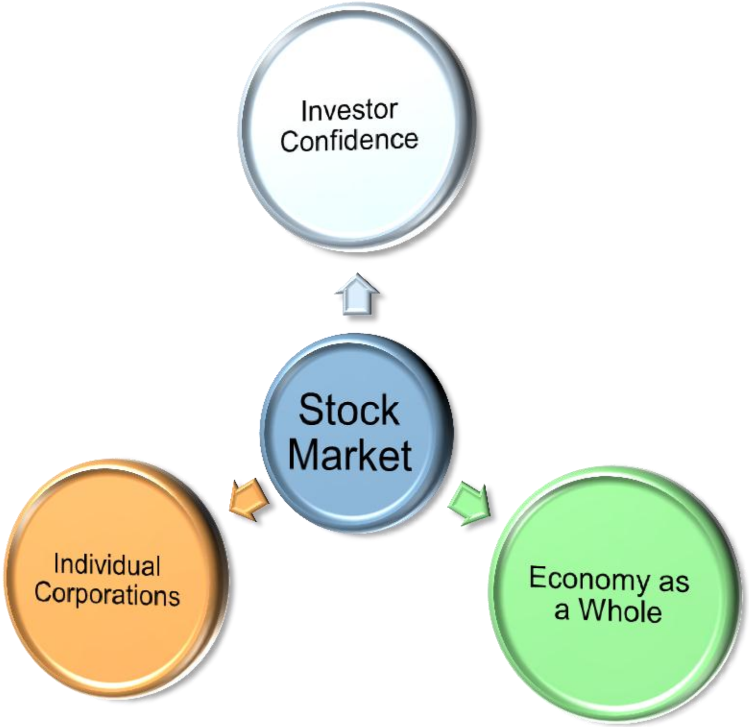
Results

Implementation

Our Experience

The Big Picture – Why do we want a Model

- The financial system is flawed
- A financial crisis has very broad repercussions on the general prosperity of the population
- The stock market can be used as a general indicator of the state of an economy
- The goal : to foresee/prevent financially unfavorable situations by modelling trading on the stock market



Fundamental Research Questions

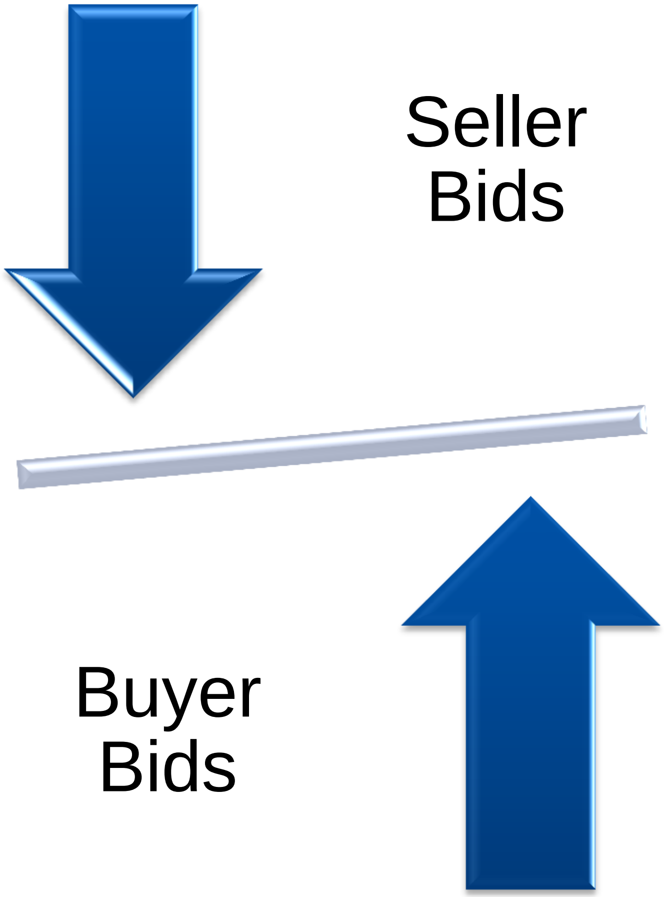
How does past market volatility affect the future price of a stock

To what extent can price regulation be used to stabilize a volatile market

How accurately can financially unfavorable situations be modelled

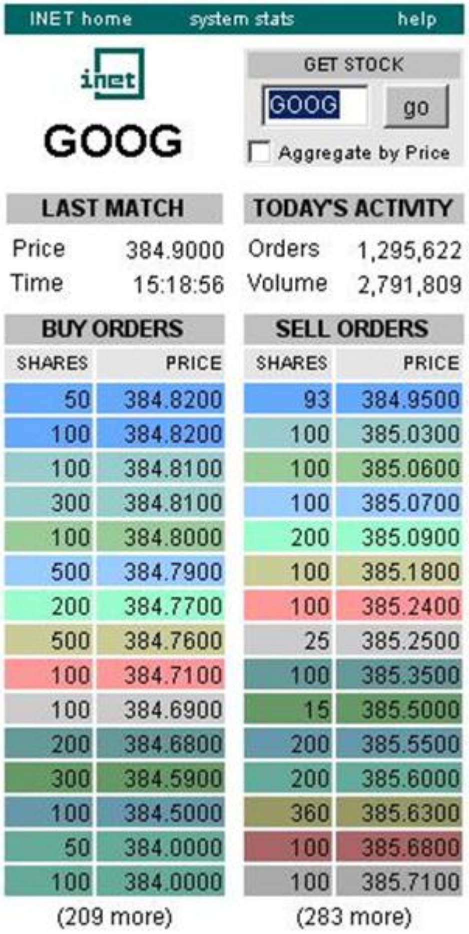
The Model (1)

- One publically traded firm
- Based on paper (Raberto, 2005)
- Agent-based simulation : Traders (agents) interact in the market (playground)
- Fixed number of traders each of which have a specific amount of shares and liquidities which evolve over time
- Limit order book (double auction) mechanism : sellers and buyers bid seperately; price overlap of bids causes a transaction



The Model (2)

- Time is separated into days
- At the end of a day the market closes : all remaining bids are cleared from the books
- Input : initial bid prices (IPO parameters), number of traders, distribution parameters for random variables, etc...
- Output : Average transaction price over time, trader assets over time, value of firm over time, etc...



GOOG

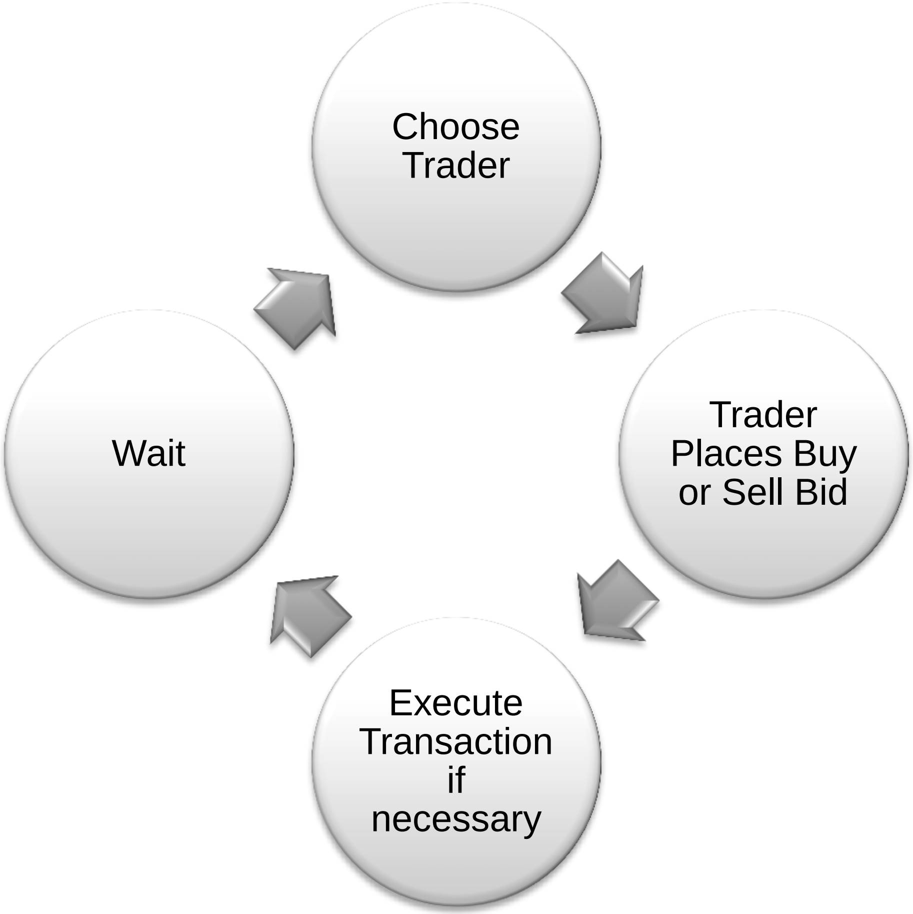
LAST MATCH	
Price	384.9000
Time	15:18:56

TODAY'S ACTIVITY	
Orders	1,295,622
Volume	2,791,809

BUY ORDERS		SELL ORDERS	
SHARES	PRICE	SHARES	PRICE
50	384.8200	93	384.9500
100	384.8200	100	385.0300
100	384.8100	100	385.0600
300	384.8100	100	385.0700
100	384.8000	200	385.0900
500	384.7900	100	385.1800
200	384.7700	100	385.2400
500	384.7600	25	385.2500
100	384.7100	100	385.3500
100	384.6900	15	385.5000
200	384.6800	200	385.5500
300	384.5900	200	385.6000
100	384.5000	360	385.6300
50	384.0000	100	385.6800
100	384.0000	100	385.7100
(209 more)		(283 more)	

Source: ece.cmu.edu

Sequence of Events – When the market is open


Choose Trader
Trader Places Buy or Sell Bid
Execute Transaction if necessary
Wait

- The waiting time follows an exponential distribution
- A trader is chosen based on a uniform distribution
- The trader will place either a buy or sell order with a predetermined probability
- The price of the order follows a Gaussian distribution with the price of the most competitive valid order as the mean
- The number of shares ordered is uniformly distributed between 1 and the trader’s maximum

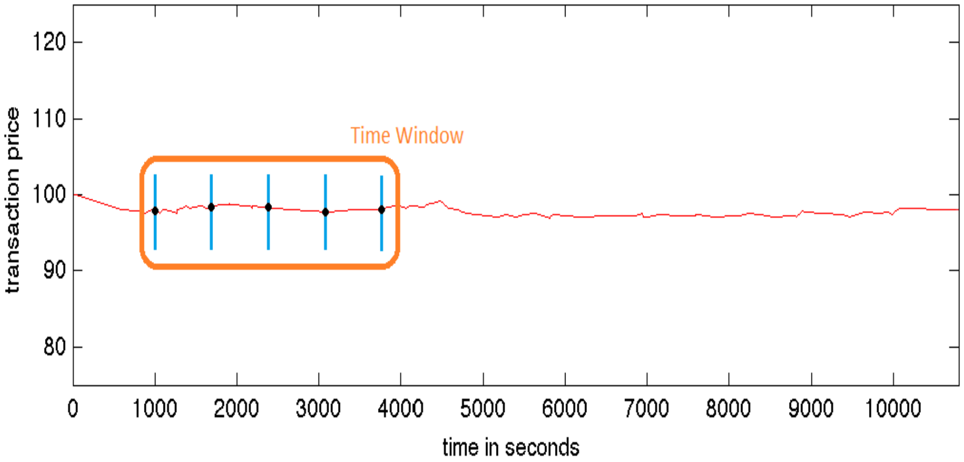
Measuring Market Volatility

- Rate of Return (ROR) : ratio of money gained or lost on an investment relative to the amount of money invested (%)
- Logarithmic returns :

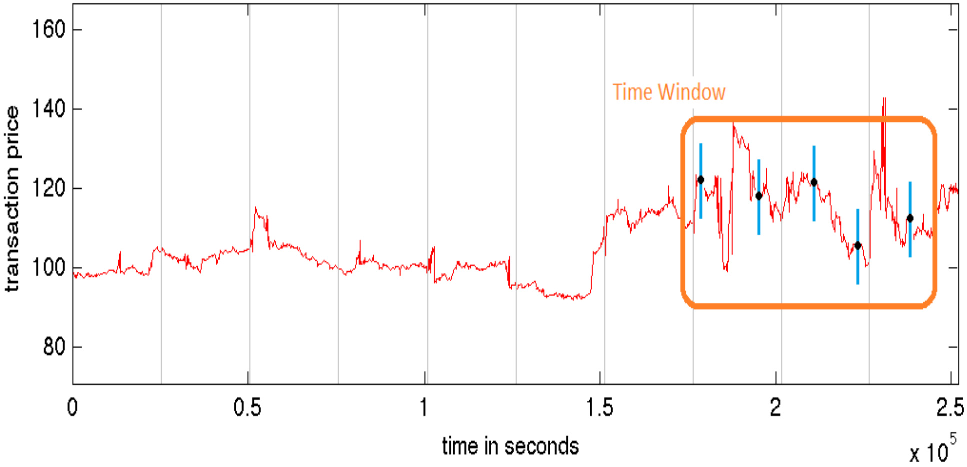
$$r_{log} = 100 \cdot \ln\left[\frac{V_{final}}{V_{initial}}\right]$$

- Single-period vs. Multiperiod
- Take standard deviation of multiple single-period return values (σ_T)
- Stable market : small deviations
- Volatile market : large deviations

Stable Market :



Volatile Market :



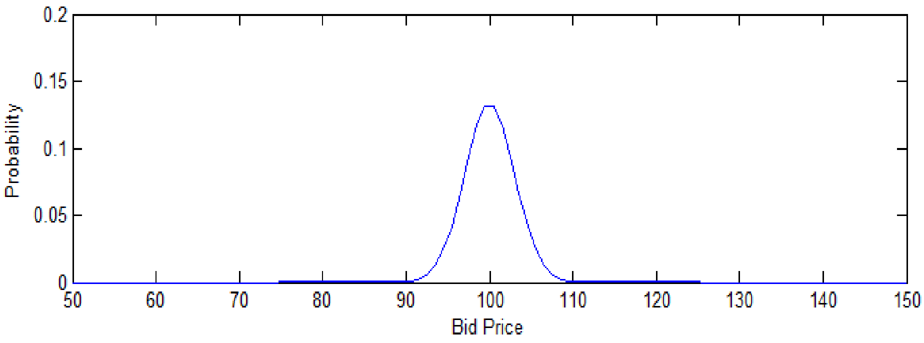
Taking past Market Volatility into Account

- Volatility feedback occurs solely in the prices of new bids
- The principle : if the transaction price has fluctuated in the past traders are less sure of the true value of the stock
- As a result, prices tend to deviate more from the latest bid values
- We can achieve this by modifying the variance of the Gaussian distribution that the new bid prices are based on :

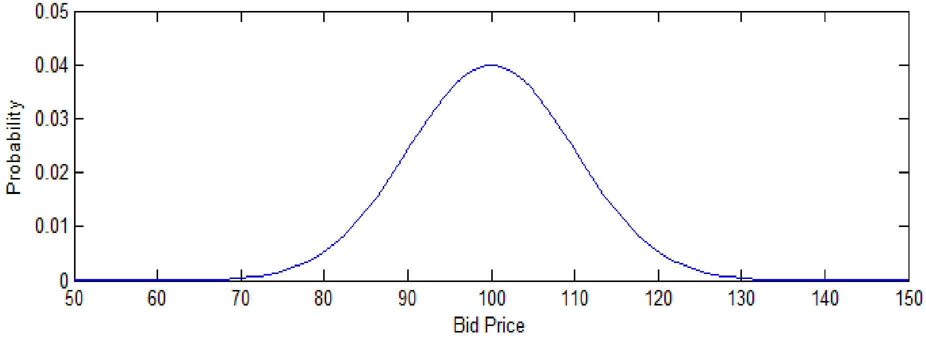
$$\sigma_{new} = k \cdot \sigma_T \,, k = const.$$

$$\sigma = \alpha \cdot \sigma_{new} + (1 - \alpha) \cdot \sigma \,, \alpha \in]0,1]$$

Low Volatility (Small Variance) :

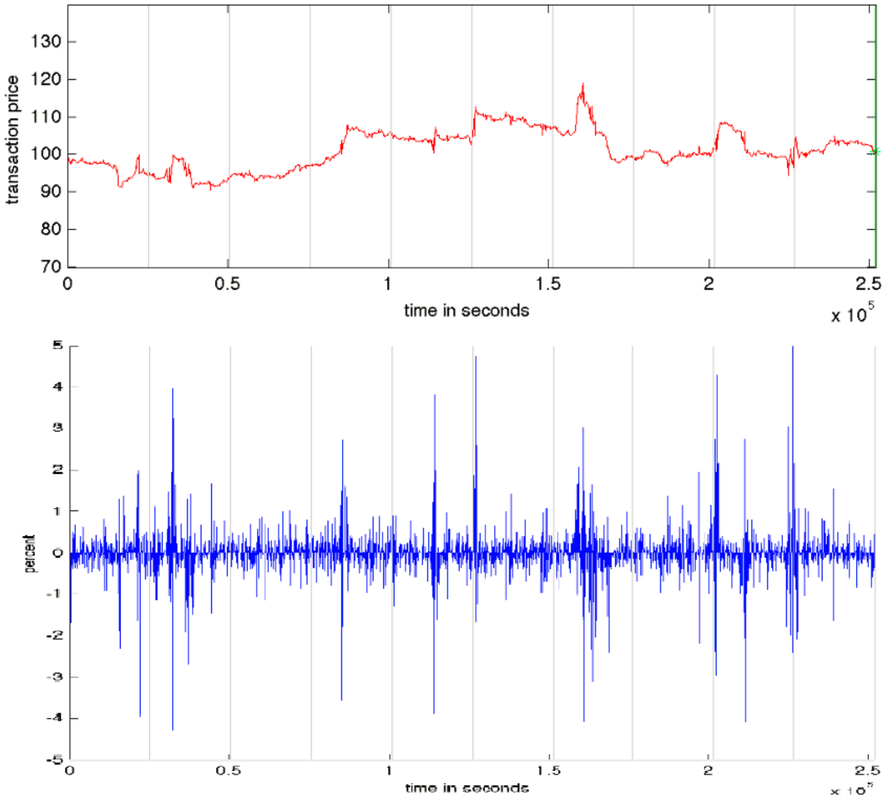


High Volatility (Large Variance) :

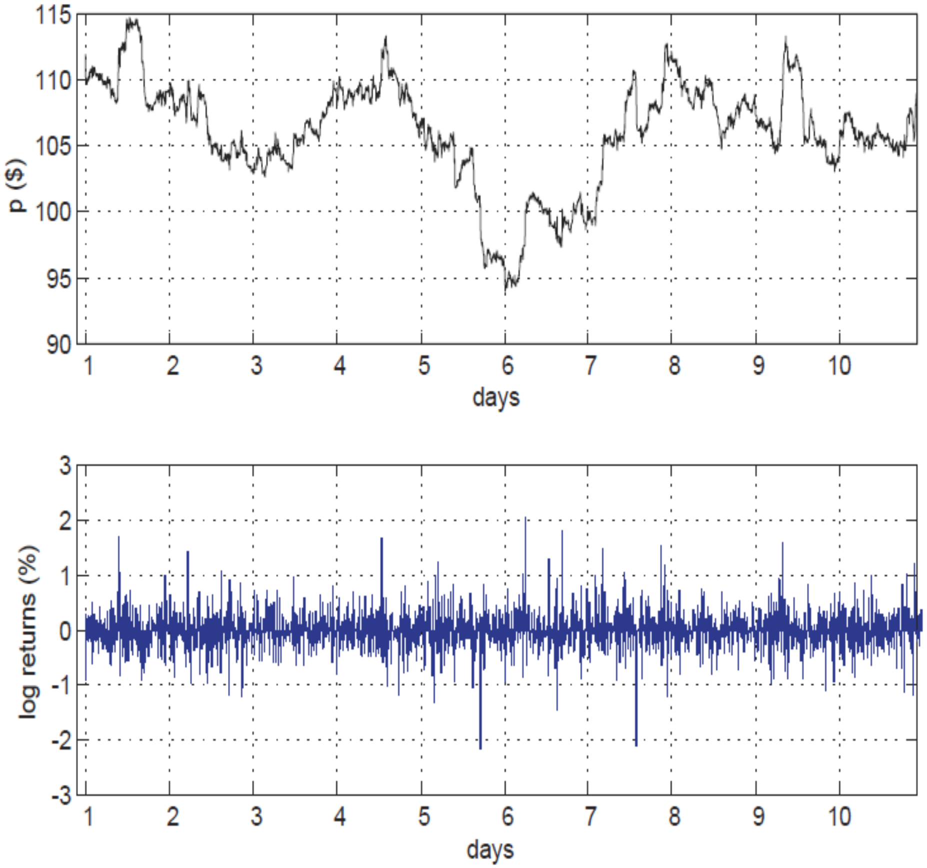


Results – Without Volatility Feedback

Our Results :

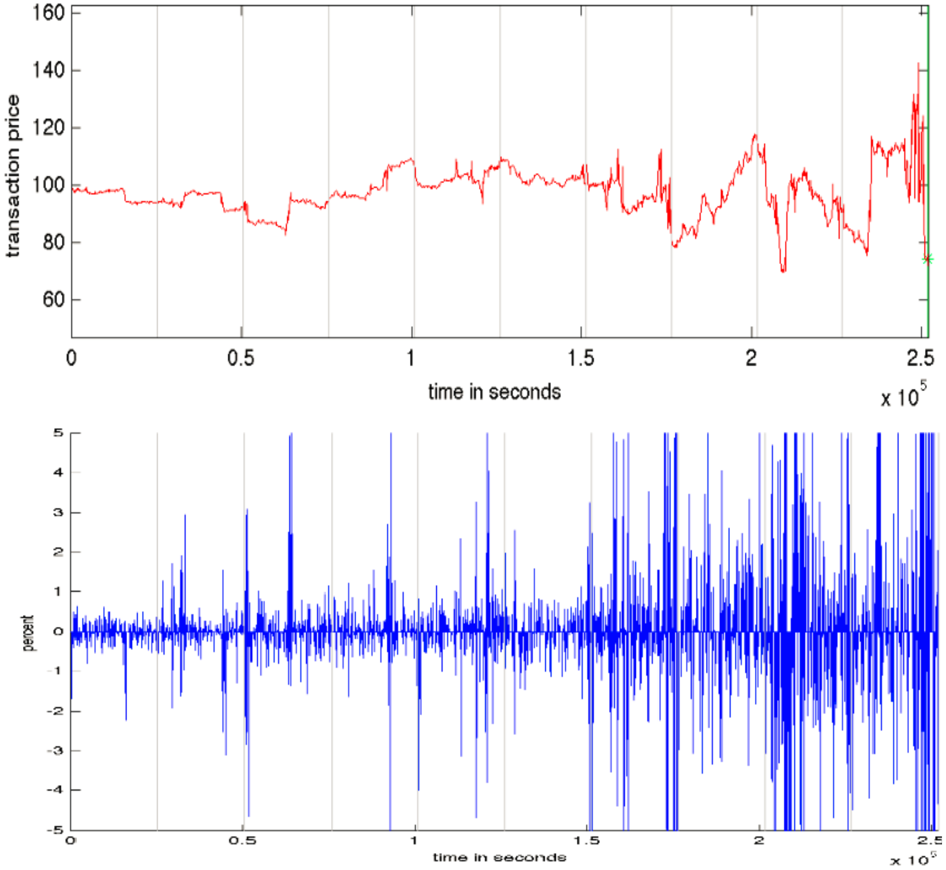


Raberto, 2005 :

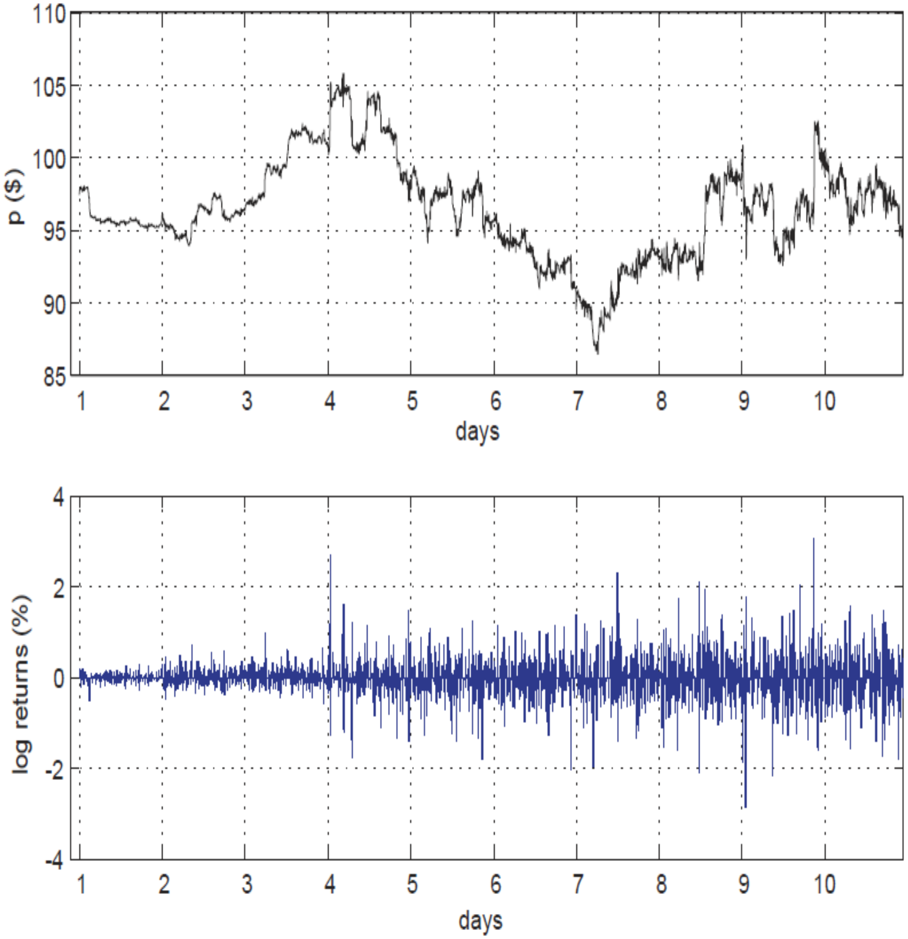


Results – With Volatility Feedback

Our Results :



Raberto, 2005 :



Using Price Regulation to stabilize the Market

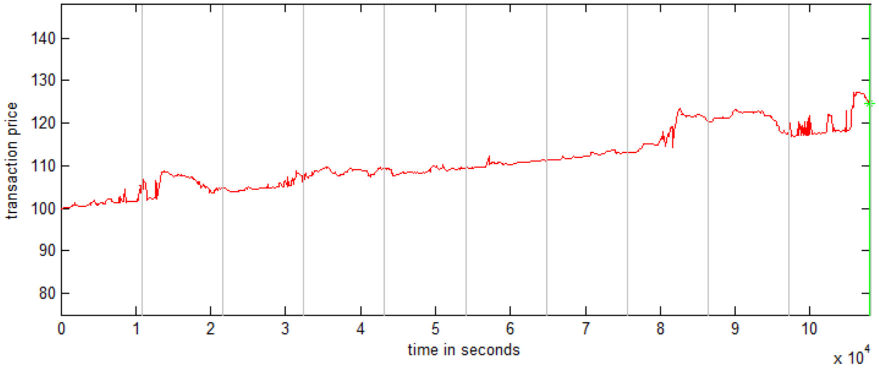
- Combination of price floor/ceiling
- In the model : prices which would be out of bounds are set to the nearest legal limit
- Enable growth (% per day)
- Result : regulation works. Market stays within bounds → stability
- However, no longer a free market → less economic welfare
- Tradeoff between stability and free market

$$\begin{cases} p_c(t) = p_{c0} + \dfrac{g \cdot p_{c0}}{24 \cdot 60 \cdot 60} \cdot t \\ p_f(t) = p_{f0} + \dfrac{g \cdot p_{f0}}{24 \cdot 60 \cdot 60} \cdot t \end{cases}$$

With g = growth (% per day)

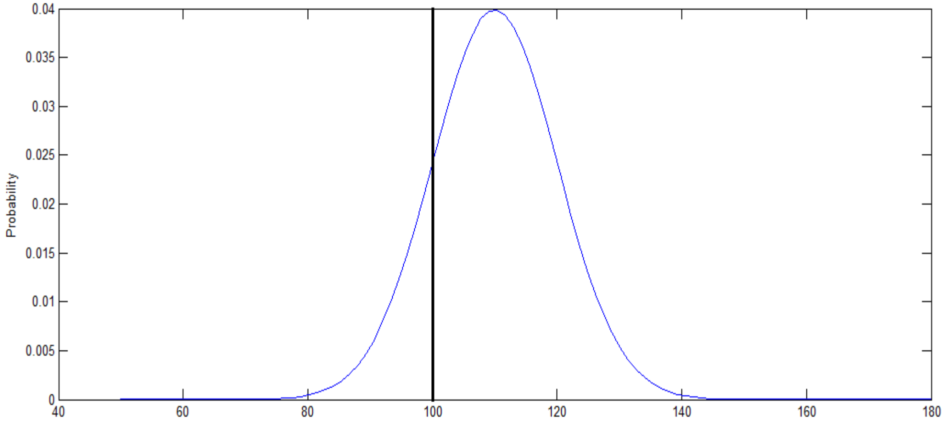
Example :

floor = 100
ceiling = 110
growth = 15% per day

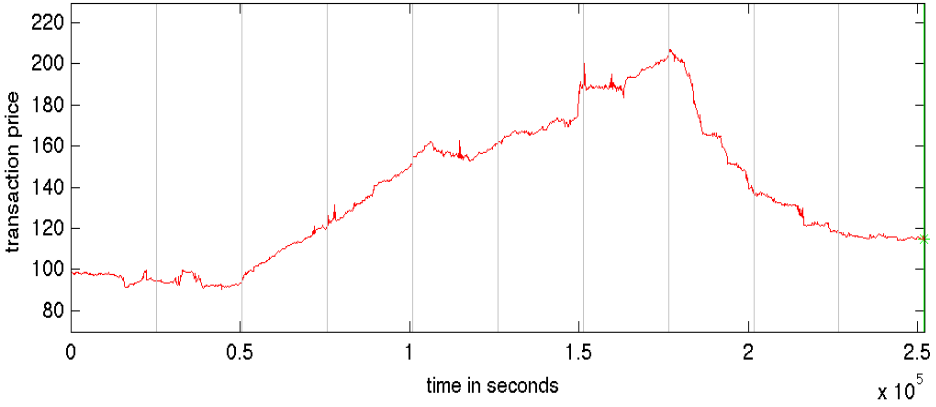


Modelling Financial Bubbles

Shifting the mean :



Result :



- Stock is consistently overvalued → prices explode
- Awareness of overvaluation → prices fall (bubble burst)
- In the model : at some point switch to a non-symmetric probability distribution for bid prices ⇒ prices are much more likely to increase. Later reverse the trend ⇒ prices are extremely likely to fall and keep falling
- Challenge : what causes the switch? When do we do it?
- Too many parameters

Implementation (1)

- Model implemented in 3 main sections:

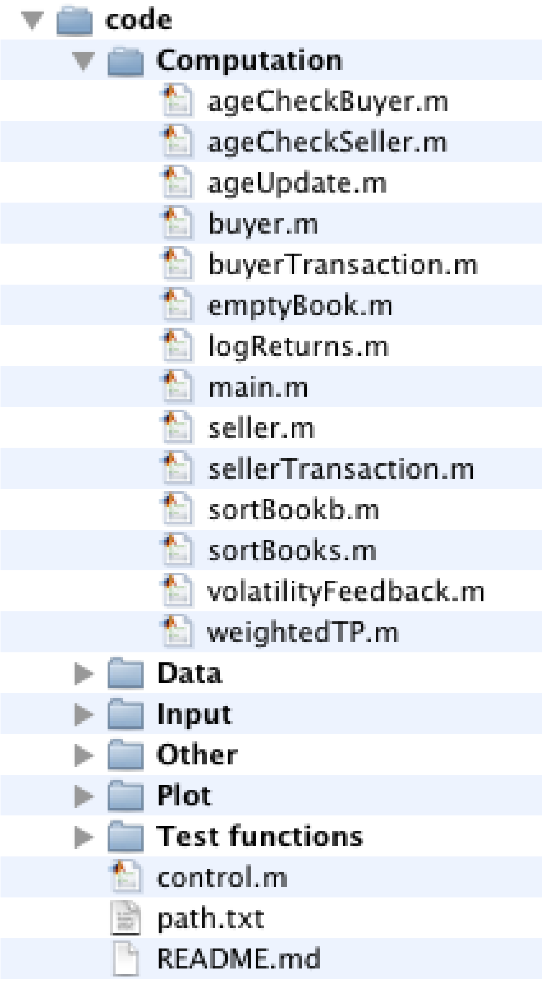
  → input generation

  → computation/simulation

  → result analysis (plot)
- Sections may be called seperately (independent)
- Input generation:

  → creates different sets of input parameters according to parameter sweep

  → saves to input files



```
▼ code
    ▼ Computation
        ageCheckBuyer.m
        ageCheckSeller.m
        ageUpdate.m
        buyer.m
        buyerTransaction.m
        emptyBook.m
        logReturns.m
        main.m
        seller.m
        sellerTransaction.m
        sortBookb.m
        sortBooks.m
        volatilityFeedback.m
        weightedTP.m
    ▶ Data
    ▶ Input
    ▶ Other
    ▶ Plot
    ▶ Test functions
      control.m
      path.txt
      README.md
```

Implementation (2)

- Computation/simulation:

→ runs model on specific set of parameters and saves the results

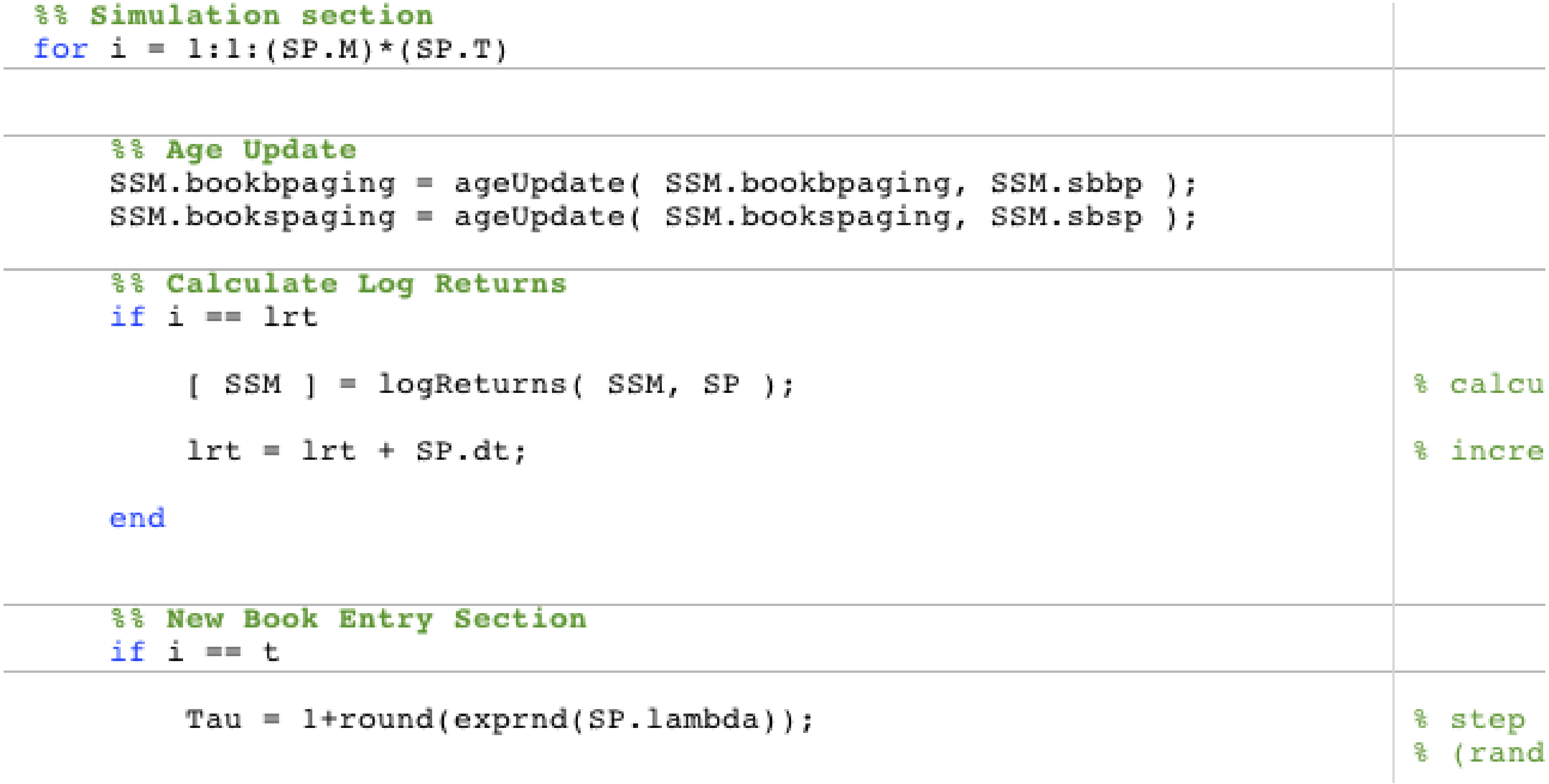
→ *main.m* function contains for-loop which iterates through each simulation second and calls different sub-functions (auction age update, new auction, book emptying, etc...)

- Result analysis:

→ generates plot of the results

→ live plot also available (simulation takes more time)

main.m (sample)

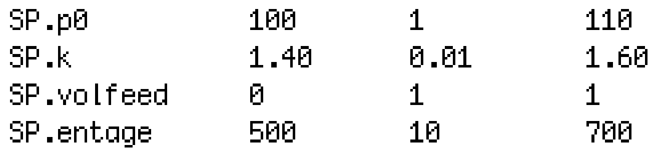


```
%% Simulation section
for i = 1:1:(SP.M)*(SP.T)


    %% Age Update
    SSM.bookbpaging = ageUpdate( SSM.bookbpaging, SSM.sbbp );
    SSM.bookspaging = ageUpdate( SSM.bookspaging, SSM.sbsp );

    %% Calculate Log Returns
    if i == lrt

        [ SSM ] = logReturns( SSM, SP );                                  % calcu

        lrt = lrt + SP.dt;                                                % incre

    end


    %% New Book Entry Section
    if i == t

        Tau = 1+round(exprnd(SP.lambda));                                 % step
                                                                          % (rand
```

Parameter Sweep

- *param.txt* defines the sweep range of the involved parameters
- Every possible combination of parameters is then generated
- Input files are created

→ Example of *param.txt* :



```
SP.p0          100      1        110
SP.k           1.40     0.01     1.60
SP.volfeed     0        1        1
SP.entage      500      10       700
```

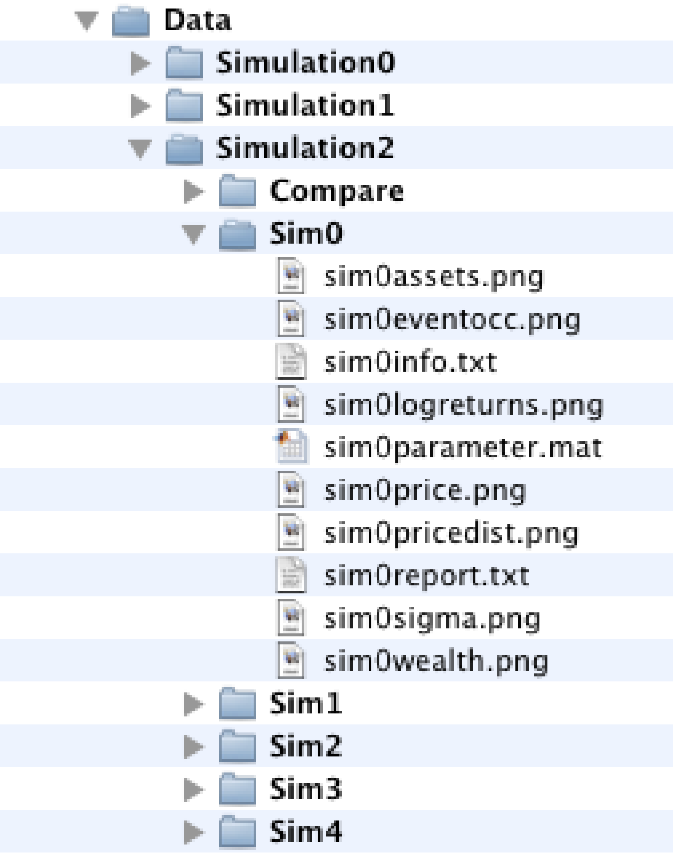
Automatic File Structure

- *control.m*:

  → generates file and folder structure using bash commands

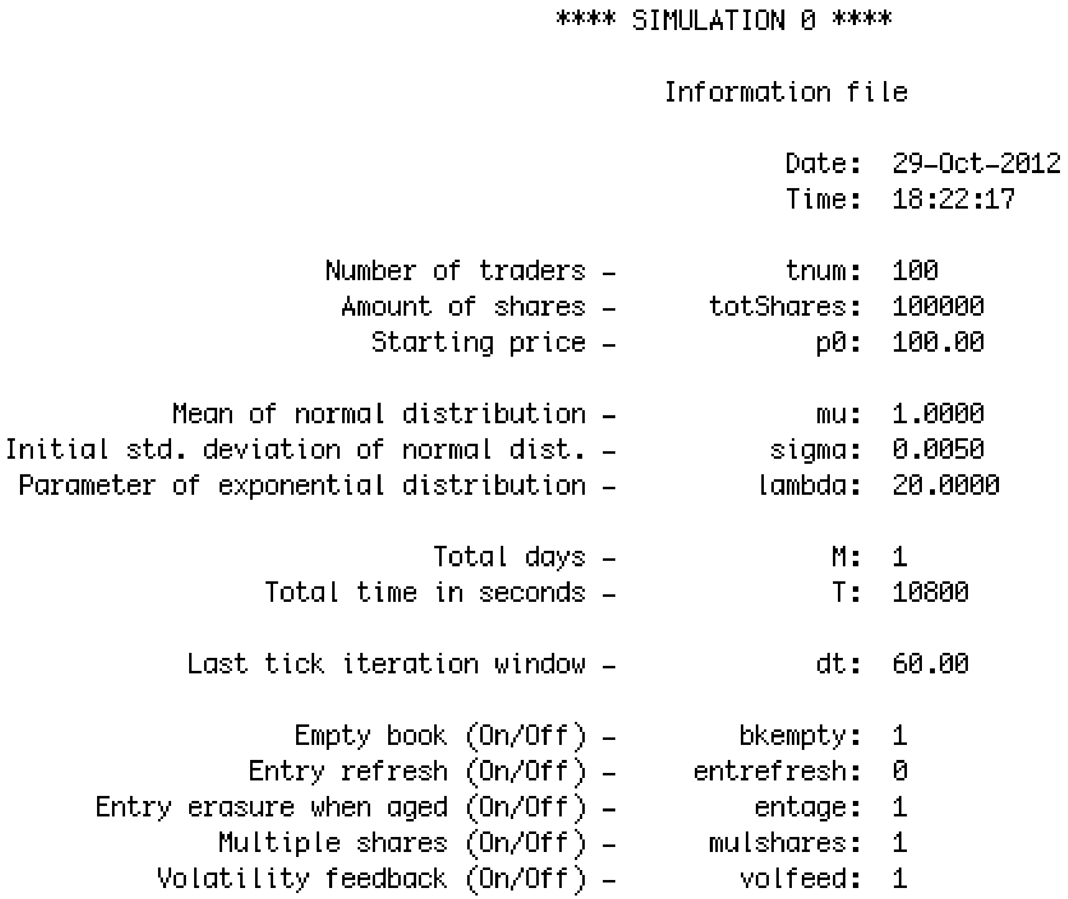
  → executes parameter sweep according to chosen parameters (*param.txt*) and creates input files

  → runs simulation and calls plot functions
- *info-files* and *log-files* created for each simulation
- *“Compare folder*” with graphs to be compared



```
Data
    Simulation0
    Simulation1
    Simulation2
        Compare
        Sim0
            sim0assets.png
            sim0eventocc.png
            sim0info.txt
            sim0logreturns.png
            sim0parameter.mat
            sim0price.png
            sim0pricedist.png
            sim0report.txt
            sim0sigma.png
            sim0wealth.png
        Sim1
        Sim2
        Sim3
        Sim4
```

Information File (*sim0info.txt*)



```
                                    **** SIMULATION 0 ****

                                           Information file

                                                   Date:  29-Oct-2012
                                                   Time:  18:22:17

                     Number of traders -           tnum:  100
                      Amount of shares -      totShares:  100000
                        Starting price -             p0:  100.00

           Mean of normal distribution -             mu:  1.0000
Initial std. deviation of normal dist. -          sigma:  0.0050
 Parameter of exponential distribution -         lambda:  20.0000

                            Total days -              M:  1
                 Total time in seconds -              T:  10800

            Last tick iteration window -             dt:  60.00

                   Empty book (On/Off) -        bkempty:  1
                Entry refresh (On/Off) -     entrefresh:  0
      Entry erasure when aged (On/Off) -         entage:  1
              Multiple shares (On/Off) -      mulshares:  1
          Volatility feedback (On/Off) -        volfeed:  1
```

Our Experience - Why we didn’t hand it in

- Time management

  → debugging can be VERY time consuming!

  → eg. ~12h : parameter is too big by a factor of 100

Problems we had to overcome

- Model badly described in paper

  → led to a lot of testing and bouncing ideas off eachother
- Very unclear concerning implementation of the model

  → eg. parameter choice
- Hard to control the model due to large number of random events

  → First simulated without random variables
- Things got complicated very quickly

  → Started with the bare minimum and slowly added to it

Things we did well

- MatLab Script/Function hierarchy/organization
  → keep track of the big picture
- Commenting almost every line of code
  → easier to find your way around a complex model
- Using existing MatLab functions
- Using data structures (eg. MatLab structs)
  → system state matrices as function inputs/outputs
  → grouping important output variables together
  → simplifies data analysis

Thanks for listening!