Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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

Project Report

Shoe Resale Market Model from Fashion Trends

Arturas Malinauskas & David Martinez de la Cruz

Zurich
December 2018

Agreement for free-download

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

Arturas Malinauskas David Martinez de la Cruz

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

SHOE RESALE MARKET FROM FASHION TRENDS

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	First name(s):
~~ARTURAS~~ MALINAUSKAS	ARTURAS
MARTÍNEZ DE LA CRUZ	DAVID

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date

Zurich, 8th December 2018

Signature(s)

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

1 Abstract

A unique secondary market has boomed in the past five years: sneakers and other fashionable apparel are being sold outside retail stores for 150-1000 % the retail market price. Demand is high for these items of limited quantity which offer social status and a feeling of uniqueness to their owners. This project simulates this market to understand how its dynamics differ from a traditional stock market. A financial model was created in MATLAB to simulate buyers and sellers of shoes with variable prices and social desirability.

2 Individual contributions

Both team members equally developed the research questions, goals of the project, the final report, and presentation. They also worked together to resolve issues in their individual tasks.

3 Introduction and Motivations

Shoes have acquired a cultural significance in parts of the modern world. Many shoes manufacturers partner with trend setters to create stylish limited quantity shoe models. In recent years, these shoes have been selling out quickly in the retail market and then resold in a secondary market with markups reaching over 1000 % [1]. The sum of these transactions is estimated to be over 1 billion US dollars this year [2].There are many different players in this secondary market. This project seeks to model transactions in the secondary market and identify the best strategy for each type of player.

3.1 Motivation

There is a great pool of research on simulating traditional stock markets [3, 4]. We want to see how applicable these techniques are to a market which is relatively new and has not been studied academically. The resale market for sneakers is also features more diverse players than the stock market. Maximizing financial returns is not the only motivation for some people in the sneaker resale market. The sneakers themselves offer a social value from their rarity and style that many people want to own [1]. We believe it is this desire which creates very large fluctuations in market price above initial retail price. The physical condition of a shoe is also important to its value; pristine shoes sell for more than ones that have been worn more.

This market offers some unique challenges for simulation. The social value of a sneaker is dependant on many things, who is promoting them, the brand, the color, the condition, and its rarity. Like old cars, some sneaker models increase in value simply because they are vintage and viewed as "classics" that defined a brand. These factors are not equally valued in the eyes of the different traders in the market. The traders themselves have different incentives: some want to maximize the stylishness of their sneakers, some want to collect the most interesting sneakers (sometimes not even wearing them), and some are simply seeking to profit as middlemen serving the other groups. Some consumer exhibit behavior that is a combination of the previous factors. The various types of traders create a more complicated and perhaps more interesting trading dynamic than the one seen in the stock market.

3.2 Fundamental Questions

- How does the proportion of sneakers to agents affect price?
- How is price affected by the number of competing sneaker brands?
- Which market parameters are the most impactful on price?
- What behaviors of the sneaker market differ from a standard stock market model?

The first question is important to sneaker manufacturers. They decide the number of sneakers they will make and the initial selling price. The manufacturers want to maximize profit. But they receive an intangible benefit to their brand's reputation and desirability due to excitement and promotion in the resale market [1]. Thus some firms may not sell a shoe for the highest price they could in the present, because they want to create excitement and demand for their brand in the future. Knowing how many sneakers are needed to satisfy a given number of agents would help manufacturers optimize their business.

The second question is important to manufacturers and profiteers. The market has many competitors with limited product differentiation. It is important to understand how competitors may induce price competition.

The third question is important to evaluating price sensitivity. By understanding the most important parameters, the people that are represented by our agents may be able to take information about the real values of those parameters and better inform their own strategies for trading in the market.

The fourth question is important to the academic scope of this work. If there are few differences, it indicates that existing modeling techniques can be applied to a broader array of problem types. If there are significant differences, this opens up new avenues of research to more deeply understand the behavior of non-traditional markets. Appropriately modeling and understanding those different behaviors may offer greater insight into the subtle dynamics of economic systems.

3.3 Expected Results

We expect to find that reducing the ratio of sneakers to agents will increase resale price and create more volatility in the market. Lower supply for given demand should create higher prices in accordance to basic economic theory. If there are too few shoes in the market, the agents may not be able to afford them, which would reduce transaction frequency. This would correspond to a loss of excitement over a specific shoe type.

If the sneaker to agent ratio is constant, but there are more competing brands, the prices should also go down. The model gives preference to owning a greater variety of sneaker types for the collectors and socialites. This should result in prices going down as agents are able to bid on a wider variety of products.

We expect the sneaker to agent ratio to have a larger impact on price than the number of competing brands. This parameter affects the available supply. From an economic perspective the available supply should affect price more than the presence of competition.

There should be some noticeable differences between stock market models and our sneaker market model because our trader agents have behaviors not seen in the stock market. We expect that the different agents see different rates of return based on their preferences for social status or economic value.

4 Description of the Model

This is a complex market with many dynamics, players, and price factors - many of which are not perfectly rational. So we identified the core elements of the model that lent themselves to computational simulation. We built on previous research by Raberto *et al.* [3] which uses an agent based model to simulate a stock market.

In Raberto *et al.*'s model [3], the agents are traders who buy and sell stocks. The sneaker market behaves in a similar way with buyers and sellers. The difference is that non-price value of sneakers is modelled as an additional parameter. Thus, the shoes that are bought and sold are represented by two numbers: a price and a "social status". The social status parameter represents the non-price based appeal of the shoe. The social status quantifies how people in the real world would view the stylishness, quality, and condition of a shoe. The agents respond to the social status in different ways and have different end objectives. The four agents we model are the trendsetters, socialites, collectors, and profiteers.

4.1 Description of Agents

The trendsetters are agents that bring shoes into the marketplace. They represent different shoe brands and create the initial social status. As trendsetters they do not need to buy shoes.

The socialites are agents who buy and sell sneakers to optimize social status. These correspond to people in the real world that buy trending sneakers, wear them for a short time, and sell or exchange them for different sneakers.

The collectors are agents who buy but do not sell any sneakers. They correspond to people in the real world that want to buy the best shoes which will retain value. They take shoes out of the marketplace and often display them in personal collections or shoe racks. In the real world, these collectors may also sell some of their shoes, or buy multiple pairs so they can wear some while keeping another pair in pristine condition. Collectors in the real world often have one to two hundred different pairs of shoes [1]

Finally the profiteers are agents who buy and sell sneakers to maximize monetary profit. These agents ignore the social status. These correspond to people in the real world who have little interest in sneakers themselves, but want to capitalize on high mark ups over retail price.

There are many other types of agents that could have been modelled, but the complexity of their addition did not justify their lesser importance to the market. For example, individuals in the real world may switch roles based on their own incomes and personal impulses. There are counterfeit shoes in the market, scammers who take people's money without sending anything in return, trading bots, and other miscellaneous occurrences that happen less frequently than the core behaviors we modelled.

When modeling the agents, each of them will have different properties which will differ from one type of agent to another. This fields include the social status of the agent, which varies over time; the desire to buy new shoes, defined depending on the agent; and some initial money to start trading.

These agents are classified into buyers and sellers, depending on their type and several factors. Socialites will become sellers if they do not have more money to buy and they have some items, or if they do not want to wear their sneakers anymore. Profiteers however, become sellers when the average value of their assets is higher than the average selling price of the market, trying to maximize their profit. Whenever these two agents sell all their items, they become buyers again.

4.2 Trading Environment

The environment for the agents is a closed market, where the items are released by the trendsetters and then interchanged between the different agents according to their needs and desires. The agents are fixed, meaning they don't change from one type to another and there is no exit or entry of agents into the market. The shoes being traded may depreciate in value to a point where they are no longer sold in the market. But new shoes will be periodically released into the market to ensure demand is met. It is also assumed the agents have supplemental sources of income which is paid out regularly in set periods (representing their real world salary from a job). This ensures that new items in the market are affordable by some people.

The market operates at every time step, which corresponds to a day of trading. Buyers and sellers will decide their individual prices for a day of trading. Buyers that are unable to buy shoes during a trading day will generally increase the price they are willing to pay on the next day. Sellers that are unable to sell a shoe at a price will generally decrease the price they are willing to pay. The time it would take to ship sneakers from buyers to sellers is ignored.

The transactions only occur when certain conditions are met. These conditions depend on the buying price of each buyer and the selling price of each item. The former has to be higher than the latter for the transaction to take place. Additionally, the social status of the item is tied to a random variable. Sneakers with less social status are less likely to be sold in the market.

5 Implementation

5.1 Inputs

The inputs required by the user to run this simulation are mainly associated with the size of the simulation. The number of agents to be simulated and the amount of sneakers introduced in the market have to be set. In order to simulate the behaviour of this market, the model should be run with a larger number of agents than items sold. If this is not true the economic system breaks down because the items are no longer limited, everyone could buy one. This is true for the real world: there are many more people interested in buying the limited edition shoes than the available supply of those shoes. The scarcity creates value in all economic systems.

Furthermore, an initial price for the buyers needs to be set. This sets an initial condition and only affects the buying prices in the beginning of the simulation. The buying and selling prices are modified later depending on offers and demand.

The last inputs required are the amount of brands for the sneakers and the duration of the simulation, which is given by the days the user wants to simulate. The former input has to be reasonable according to the amount of agents introduced in the system and the amount of items.

With these inputs set, the model is set to run and simulate the evolution of the model explained in the previous section.

5.2 Outputs

The outputs produced by the program are a set of graphs showing the evolution of different parameters through time as well as some final characteristics that define the agents and their behaviour. These graphs and figures will be discussed in the following section 6.

5.3 Fixed Simulation Parameters

For the simulation to run more parameters than the ones provided in the input routine are needed. These values are set in the code, and they can either have a fixed value during the whole simulation or vary randomly as time goes on. The frequency with which brands release new sneakers to the market is set to be one of those random values calculated in the code, as well as the change in social status of the items through time. The fixed values include for example how much the buying and selling prices are increased or decreased depending on whether the agent performed a transaction or not. These values are weighted with the desire of the buyers to buy the items.

These parameters have been optimized in the code through many iterations so that the model represents the real behaviour of the market and doesn't produce results far from examples we found in our research. This optimization allowed us to better understand the important market dynamics and critically think about why values had to be in certain ranges.

5.4 Code Structure

In the code's first steps, the data structures are created and initialized. There are two main data structures, buyers and sellers, which are then selected to be one of the agents described previously. Initially only trendsetters are sellers, as they are the ones to introduce the sneakers into the market. Then the rest of the agents become potential buyers.

Each buyer is then assigned a set of attributes corresponding to their initial money, social status, and desire to buy items. Sellers on the other hand are assigned assets which are assigned randomly to one brand. A check is made to ensure each brand is represented by a trendsetter. Each brand is then assigned a random social status and initial price, which transfers to the trendsetters belonging to that brand. This way the items get their initial selling prices and social value.

Before running the simulation, some parameters are initialized, such as the increase or decrease in prices depending on the transaction. The code starts the simulation, which performs the purchase of the item, the change in the agents' attributes according to their actions and the update of the items social status as time progresses.

The first lines of the loop for the simulation calculate all the desired information in each iteration, such as the average selling price or the average buying price of each different type of agent. Then the trading operation between the agents occurs. Transactions happen only when the conditions mentioned in subsection 4.2 are fulfilled. With the purpose of approximating reality, the constraint that one buyer can only buy one item per day has been applied into this model.

The next lines in the code (shown in A) vary the different parameters associated with each agent. For buyers, the buying prices are adjusted, the social status is calculated based on the items they own, and the desire changes depending on their agent type. Sellers lower their selling prices if they did not manage to sell one item.

Following these changes, the agents are moved from buyers to sellers and the other way around according to the factors explained before. Both sellers and buyers retain their values for desire, social status, buying price, etc, as they are the same people that move from one side to the other of the market.

Finally, the status of the items is changed in each iteration. Due to depreciation, items lose a random amount of their social status, making it less likely to be bought. This also accounts for the loss of value because of wear. Each brand is also assigned a factor equating to 'vintageness', so some shoe brands increase in social status. However, depreciation tends to be bigger in our time scales and the social status of the shoe decays in time.

Once all of this actions are performed, the new day begins with the transactions between the new buyers and sellers according to their new parameters.

6 Simulation Results and Discussion

To answer the fundamental questions proposed in subsection 3.2, several simulations have been carried out. The different simulations varied the input parameters to test the most influential parameters.

6.1 Nominal Test

We perform a baseline simulation which is used as a comparison for our results. The parameters of the baseline are conservative values that ensure the model is stable. The results from this simulation are our reference for further runs and changes to the input parameters.

The number of agents chosen for this simulation is $N = 1000$ with $n_{items} = 800$ which will be divided into 4 different brands, which will be initialized randomly. The starting price for the buyers in this market is $p = 700$ currency units. This market will be modelled for one year, so the time steps for the iteration will be $t = 350$ days.

The following graphs illustrate the results obtained with this simulation. In *Figure* 1 the average of the buying prices from all the different agents and the global average of the selling price. It can be seen the different strategies followed by each type of agent. The buying prices from the socialites and profiteers do not present much difference from one another. They interchange the roles of sellers and buyers once all the items have been released to the market by the trendsetters. As it can be seen, the socialites are the more active in the market, with a buying price closer to the average selling price.

Meanwhile, the collectors follow a very different trend. These agents buy during the first day of the new items, as this items are the ones worth to collect. This makes them run out of money, and therefore their buying prices decay and start rising, as they earn money through time. When new items are introduced, they use this money in order

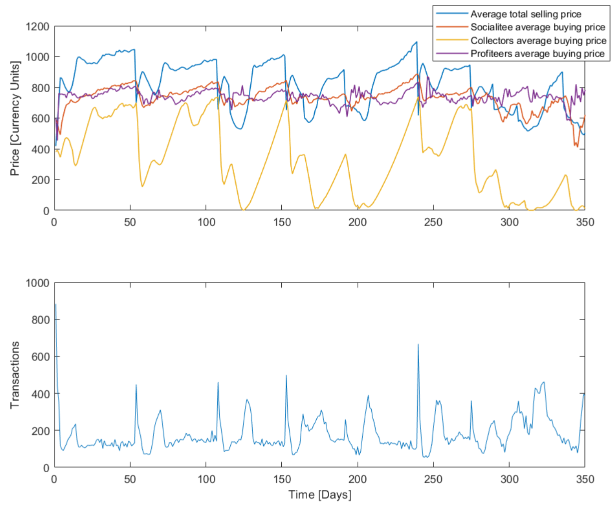


Figure 1: **Top** - Average buying prices from the different buyers and global average price. **Bottom** - Number of transactions each day. Simulation ran with *N=1000*, $n_{items} = 800$, $n_{brands} = 4$ and *t=350 days*

to acquire the brand new released sneakers

Furthermore, the average selling price is higher than the average buying price. This corresponds to the real world, where no seller wants to sell for less of what the paid to buy the item. The only points in which the selling price is lower than the buying price is at very selected days, which correspond with the moments when new items are released. This new items have much lower prices than the value of the sneakers already in the market. It is interesting to note how profiteers increase their buying prices during these periods in order to buy more and therefore sell it for a higher price.

Figure 1 also shows the number of transactions in each day. The peaks shown in this graph correspond to the moments when new items are released and when they are starting to get sold in the market as socialites get tired of them and go looking for different sneakers. As time progresses, more items are in the market and the number of

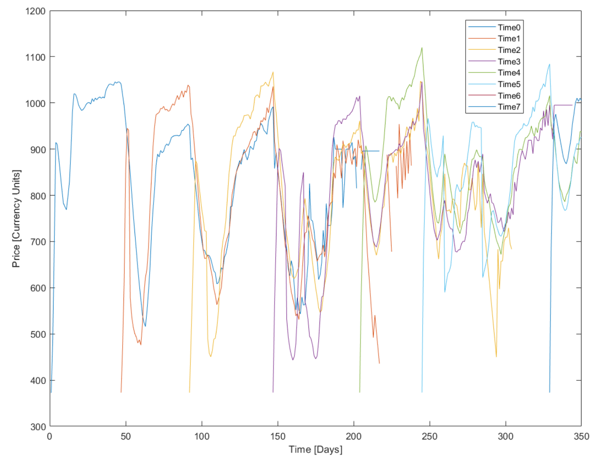


Figure 2: Average selling price of the items introduced in each time period. Time0 being the oldest items in the market and Time7 the newest. Simulation ran with *N=1000*, $n_{items} = 800$, $n_{brands} = 4$, *p=700* and *t=350 days*

transactions starts oscillating at the end of the simulation run. This because there are more items to be interchanged and the agents are more active, trading more goods.

The introduction of new items to the market can be observed in *Figure* 2. This graph also shows how newer items have a higher selling price, as they are more demanded. The last two graphs show also how the selling price of the items introduced in one period increases during that period until new sneakers are introduced. This is because items start coming out of the market as they have a social status below a certain threshold, or they are property of the collectors, being also out of the market.

Another interesting result to look at how the different brands evolve through time. This brands are affected by the depreciation and by a random 'vintageness' factor, which is equal for each brand but it is created randomly every iteration. The result of this behavior can be seen in *Figure* 3. This graph shows how competitive this market is,

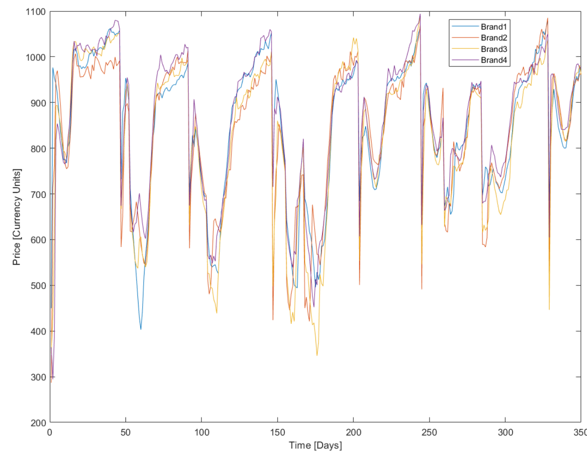


Figure 3: Average selling price per brand. Simulation ran with N=*1000*, $n_{items} = 800$, $n_{brands} = 4$, *p=700* and *t=350 days*

where the four different brands have a similar buying price. However, the best-selling brand changes from time period to time period. This reflects the reality, as different designs and models from the same brand have a bigger or smaller impact on society. When looking through a sort period of time, it can be seen how 'vintageness' affects randomly to the brands, as their selling prices oscillates within days.

The results of these transactions in the market can be seen in *Figure* 4, where the net profit of the profiteers can be seen. This profit is calculated as the difference of what they paid for the items and what they sold it for. When the market closes, the value of each profiteer's inventory is added, as this is money they have invested. However, this leads to some error, as for profiteers who are still buyers, they are selling it for a lower price and therefore losing money, which would not be the behavior of the modelled profiteer.

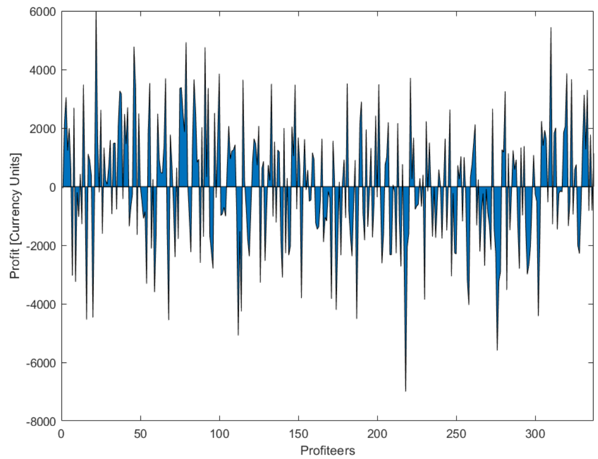
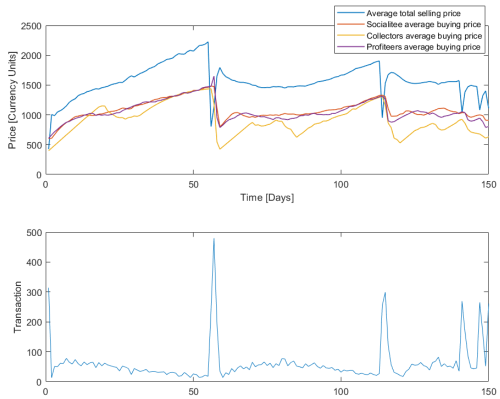


Figure 4: Profit made by each profiteer at the end of the simulation. Current inventory is added. Simulation ran with *N=1000*, $n_{items} = 800$, $n_{brands} = 4$, *p=700* and *t=350 days*

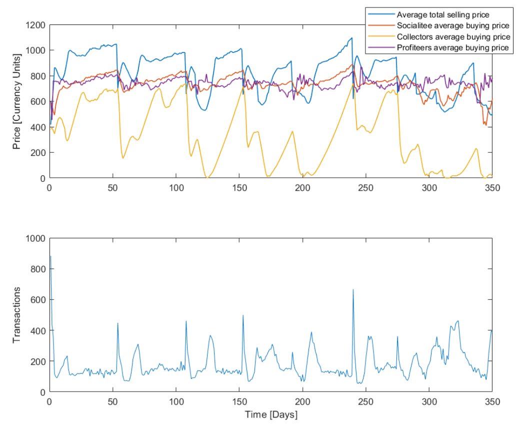
These results from the nominal inputs can be now used to analyze how the outputs of the market vary when changing the inputs and therefore answer the fundamental questions.

6.2 Effect of Ratio of Sneakers to Agents

To answer the first question, the following simulation was done. The number of agents was kept constant, *N=1000*, the number of brands $n_{brands} = 4$ and the initial price *p=700* were the same and the brands were initialized randomly, but the number of items was reduced from 800 to $n_{items} = 200$. This way, there are not enough items to satisfy all the agents and the competition between them increases. Additionally, the simulation lasted only *t=150* days, as the difference can already be seen in this short period of time and the evolution of the market in time would follow the same trends as in the beginning of it. Similar simulations were performed where the 10:2 ratio of agents to items was retained, but their magnitude varied, using *N=8000* and $n_{items} = 1600$



(a) **Top** - Average buying prices from the different buyers and global average price. **Bottom** - Number of transactions each day. Simulation ran with *N=1000*, n_{items} = 200, n_{brands} = 4, *p=700* and *t=150 days*



(b) **Top** - Average buying prices from the different buyers and global average price. **Bottom** - Number of transactions each day. Simulation ran with *N=1000*, n_{items} = 800, n_{brands} = 4 and *t=350 days*

Figure 5: Comparison between the different values of prices and number of transactions for the nominal case and the low amount of items case

for a larger number of agents and using *N=100* and n_{items} = 20 to simulate fewer agents. The overall behavior was the same in these cases, differing only as expected due to the random parameters.

The first results of this simulation can be seen in *Figure* 5. These graph shows the comparison between the nominal and this case of the evolution of the average of the buying and selling prices. Comparing these graph with *Figure* 1, it can clearly be observed that the prices are much higher when the number of items decreases. As said in the expected results, this is what should happen in an environment where supply is strictly limited and the demand is high. The number of transactions is also displayed. This value is significantly lower, as the number of items available is lower and their prices are much higher, meaning not everyone can afford it.

This difference is more clearly seen in *Figure* 6. The average trans-

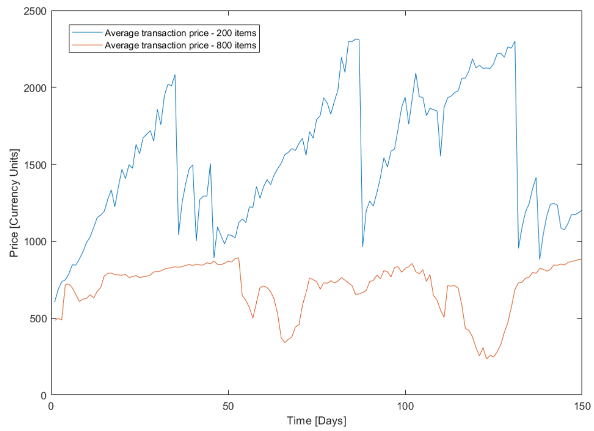


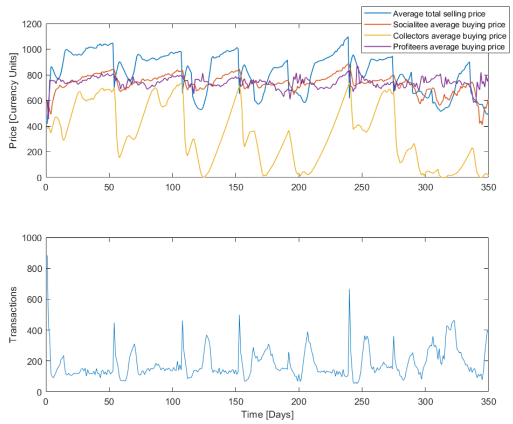
Figure 6: Comparison of average transaction prices per day for both cases: n_{items} = 800 and n_{items} = 200. Simulation ran with *N=1000*, n_{items} = 200, n_{brands} = 4, *p=700* and *t=150 days*

action prices of each day are represented in this graph for both cases, nominal and low amount of items.

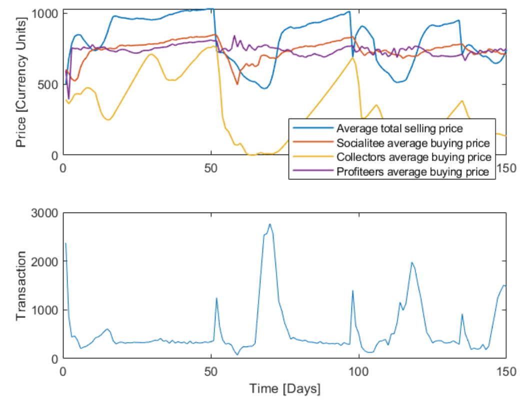
6.3 Effect of Number of Brands on the Price

This simulations investigates how the agents behave when there are many brands to choose from with different attributes. In this case n_{brands} = 10 and all other parameters are the same as the nominal case. This implies that the simulation cannot be with 100 agents any longer, as this would lead to too few trendsetters and some brands would end up with no representation in the market and would not release their items. Therefore, the number of agents is set to be *N=2500*. As the number of agents increases, so has to do the number of items, or the model would have the same behavior as in the last study. Thus, the number of items in this study is n_{items} = 2000. The rest of parameters are the same as the study in subsection 6.2.

The following figures show the behavior of the market under these



(a) **Top** - Average buying prices from the different buyers and global average price. **Bottom** - Number of transactions each day. Simulation ran with *N=1000*, n_{items} = 200, n_{brands} = 4, *p=700* and *t=150 days*

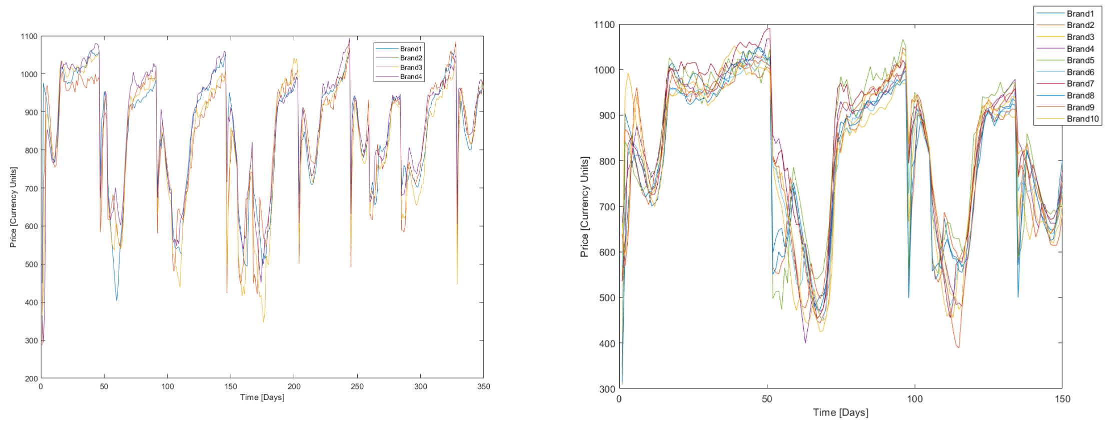


(b) **Top** - Average buying prices from the different buyers and global average price. **Bottom** - Number of transactions each day. Simulation ran with *N=2500*, n_{items} = 2000, n_{brands} = 10 and *t=150 days*

Figure 7: Comparison between the different values of prices and number of transactions for the nominal case and the high number of brands case

conditions. *Figure* 7 illustrates the comparison between the nominal case and the case discussed in this section. The prices seen in *Figure* 8b are very similar to the ones obtained for the nominal case. This is implies that the amount of brands does not have a big impact on the price, a result that was expected as the ratio of items and agents has been kept constant. Competition between brands is not a big factor to take into account for the agents.

However, and as shown in *Figure* 8, there is a bigger variance between the prices for each brand. This is also expected, as more brands would lead to more options when buying and the depreciation factor and 'vintageness' feature affects differently each brand and shoe. More difference would be seen if there was a bigger difference between brands, but this would involve modelling several parameters of the brands, which is not the objective of this work and it would increase



(a) Average selling price per brand. Simulation ran with *N=1000*, n_{items} = 800, n_{brands} = 4, *p=700* and *t=350 days*

(b) Average selling price per brand. Simulation ran with *N=2500*, n_{items} = 2000, n_{brands} = 10, *p=700* and *t=150 days*

Figure 8: Comparison between the selling prices for each brand for the nominal case and the high number of brands case

greatly the complexity of the model.

6.4 Most Influential Parameters

Once these factors have been analyzed, it is clear to see that the amount of items released in the market is a great way to modify the behavior of it. This is of great value to the sneakers brands, which can adapt the amount of items released in order to increase the selling price but without selling any item. Having a surplus of items in the market would also lead to a different behavior than the one presented in this approach, but it would not have a negative impact on the consumers, as the market would be saturated with items and there would be no competition between the agents in order to achieve their desires.

The initial price of the buyers would have a minor effect in the beginning of the simulation, as it would change the initial amount of assets bought if it was below the selling price, which would not represent the reality, since socialites, profiteers and collectors want to buy first hand items for their own goals. Nevertheless, the influence would decrease as time goes by, reaching the same results explained in this report.

Another important parameters that have an effect in the market are more intrinsic parameters such as the price variation of the items depending if they are being sold or bought, depreciation and 'vintageness', inflation if the simulation time was over years, etc. However, modifying these parameters would change the behavior of the model, making it less approximate to reality. Therefore, even though these values affect largely the model, they are not values that should be changed from one simulation to another.

6.5 Difference between Stock Market and Retail Shoe Market

The big difference between these two markets is the effect of personal value of the items and their intangible value. This value is what is accounted for in the social status term in this model presented, where the value of the shoe not only depends on the time it has been in the market, but who owns it and who released and made that sneaker. Therefore, the shoe retail market has much more factors that could influence a potential buyer or seller.

This is what makes it much more difficult to model and simulate, as it needs to take into account parameters that cannot be expressed by any number or analytic expression.

7 Summary and Outlook

The initial results of our simulation show interesting behavior worthy of further study. There are many potential refinements to the code that could increase its accuracy and computational efficiency. Further research should also be done to incorporate real world data into the simulation. There is a wealth of useful data about consumers in the market on various social media platforms.

7.1 Limitations and Simplifications

Many simplifications had to be made to create this model of the sneaker resale market. The attributes of a variety of different products and people had to be simplified. Fashion is a very subjective topic, which makes objective modeling difficult. Several arbitrary parameters must be made to attempt to model human preferences.

Due to processing power, we only simulated up to 8000 agents. There are vastly more people active in this market, on the order of millions. And these people do not trade in the market everyday. Only a few are active daily, others seasonally, and others only on special occasions when they have a surplus of disposable income.

As previously mentioned, counterfeit goods and scammers are active in this market but were not simulated. They drain peoples' available funds. The need to be cautious in the marketplace creates barriers to rapid transactions.

The code itself could be significantly optimized. It was written in small iterations, so many parameters were added as their behavior was understood. This caused our data structures to be fragmented in computer memory which increase computation time.

Further development of this simulation should experiment to find data structuring that is optimal for holding many parameters than can be exchanged between agents that have different attributes. API tools could also be used to access social media and shoe trading websites to qualify actual market activity. This could improve the ability to identify relevant market parameters and tune existing ones to behave like the real market.

7.2 Conclusion

8 References

[1] L. Chow. You See Sneakers, These Guys See Hundreds Of Millions In Resale Profit, October 2014. URL `https://fivethirtyeight.com/features/you-see-sneakers-these-guys-see-hundreds-of-millions-in-resale-prof`

[2] L. Steinberg. The Profitable Hidden Sneaker Market, September 2018. URL `https://www.forbes.com/sites/leighsteinberg/2018/09/17/the-profitable-hidden-sneaker-market/#319df41c5925`.

[3] M. Raberto, S. Cincotti, S. M. Focardi, and M. Marchesi. Agent-based simulation of a financial market. *Physica A Statistical Mechanics and its Applications*, October 2001.

[4] P. Bak, M. Paczuski, and M. Shubik. Price Variations in a Stock Market with Many Agents . *Physica A Statistical Mechanics and its Applications*, December 1997.

A Appendix A

```
function [] = Financial_Model(n_agents, n_items, price, nbrands, rndbrand, timesteps)
%% This Function uses its inputs to run a simulation of a secondary market for sneakers.
%The inputs are:
% n_agents : the total number of agents in the model
% n_items : the total number of shoes in the model
% price : the starting price of a shoe  (maybe shouldnt be fixed here?)
% nbrands : the number of competing shoe brands
% rndbrand : a parameter (0 or 1, to randomize the appeal of the shoes)
% timesteps : the length of time simulated, one time step is one day

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   CAUTION: THE NUMBER OF BRANDS MUST BE LOWER THAN NUMBER OF            %
%   TRENDSETTERS. THERE MUST BE MORE AGENTS THAT ITEMS TO MAKE A          %
%   SENSIBLE ECONOMICA MODEL. OTHERWISE THERE IS NO SCARCITY              %
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
%The outputs are a series of graphs depicting price, profit, and social
%status of the agents under various conditions


%% Data Structure CREATION

close all
agent = struct('money', 0, 'soc_stat', 0, 'items', 0, 'desire', 0, 'mode', 0, 'buysell', 0, 'sprice', 0, 'bprice', 0, 'bought', 0, 'sold', 0, 'keep', 0);
seller = struct('money', 0, 'nitems', 0, 'socstat', 0, 'bprice', 0, 'sold', 0, 'mode', 0, 'initmoney', 0, 'brand', 0, 'gain', 0);
buyer = struct('money', 0, 'nitems', 0, 'socstat', 0, 'desire', 0, 'bprice', 0, 'bought', 0, 'mode', 0, 'initmoney', 0, 'keep', 0, 'bdes', 0, 'gain', 0, 'avgiprice', 0);
brnd = struct('price', 0, 'socstat', 0);

% The brand variable takes into account from which brand is the setter
% selling the shoes. 0 - would be for independent sellers (The socialitees
% and profiteers that are selling something)

% The 'keep' variable is only used by the collector to know if they will sell
% their stuff in order to get the new released items
```

```

%The field initmoney is just only by the profiteers to know when they have
%achieved their goal of maximizing their money

%Mode indicates if the agent is a setter, socialitee, collector or
%profiteer. 1 - set, 2 - soc, 3 - col, 4 - prof

%Buysell indicates if the agent is a seller or a buyer. 0 - Seller 1 -
%Buyer

% The desire parameters measures how much the agent wants to buy the item.
% For socialitees this value will be higher, and for profiteers &
% collectors it will be smaller, but the collectors' value will rise as
% time goes on because items will be more appreciated. For trendsetters
% this value will be 0, since they do not want to buy their own stuff

%% Assigning Roles to each agent
people = repmat(agent, n_agents, 1);
brand = repmat(brnd, nbrands, 1);

n_soc = 0; n_prof = 0; n_coll = 0; n_set = 0; % variable initializations

```

```
while(n_soc<=0 || n_prof<=0 || n_coll<=0 || n_set
    <=nbrands*3)
n_coll = randi([round(n_agents*0.075), round(
    n_agents*0.1)]);
n_soc = randi([round(n_agents*0.4), round(n_agents
    *0.6)]);
n_prof = randi([round(n_agents*0.3), round(
    n_agents*0.6)]);
n_set = n_agents - n_prof - n_soc - n_coll;
end



% Assign the sellers to each brand
if rndbrand == 0 %The prices and social status of
    the brand are not randomly set, they are hard
    coded
    nbrands = 4;
end

sperbrand = zeros(1, nbrands);
while (sperbrand) < 3 % There should be more than
    3 trendsetters for each brand
    for i=1:nbrands
        sperbrand(i) = round(n_set/nbrands);
        exsel = mod(n_set, nbrands);
    end
    for i=1:exsel
        index = mod(i, nbrands);
        sperbrand(index) = sperbrand(index) + 1;
    end
```

```
    for i=1:nbrands-1
        el = round((rand*2-1)*round(n_set/(3*
            nbrands)));
        sperbrand(i) = sperbrand(i) - el;
        sperbrand(i+1) = sperbrand(i+1) + el;
    end
    tset = sum(sperbrand);
    if tset > n_set
        for i=1:tset-n_set
            index = mod(i, nbrands)+1;
            sperbrand(index) = sperbrand(index) - 1;
        end
    end
    if tset < n_set
        for i=1:n_set-tset
            index = mod(i, nbrands)+1;
            sperbrand(index) = sperbrand(index) + 1;
        end
    end
end
% keyboard;

sellers = repmat(seller, n_set, 1);
buyers = repmat(buyer, (n_agents-n_set), 1);

%% Initialize the number of items each agent has
    and the money they have
el_items = 0;
nitems = n_items;

for i=1:n_set % Assign values to the trendsetters
```

```
    people(i).money = (randi([2, 10])/10)*10e6;
    people(i).nitems = round(n_items/n_set) - 1;
    el_items = el_items + people(i).items;
    people(i).soc_stat = randi([80, 100]);
    people(i).desire = 0; % Trendsetters have no desire to buy their own goods
    people(i).mode = 1;
    people(i).buysell = 0;
end

n_items = n_items - el_items;
while n_items > 0
     p = randi([1 n_set]);
     people(p).nitems = people(p).nitems + 1;
     n_items = n_items - 1;
end

for i=n_set+1:n_set+n_soc % Assign values to the socialitees
    people(i).money = (randi([15, 25]))*1e2;
    people(i).nitems = 0;
    people(i).soc_stat = randi([5, 20]);
    people(i).desire = rand()*30+70;
    people(i).mode = 2;
    people(i).buysell = 1;
end

for i=n_set+n_soc+1:n_set+n_soc+n_coll % Assign the values to the collectors
     people(i).money = (randi([15, 25]))*1e2;
     people(i).nitems = 0;
```

```
    people(i).soc_stat = randi([5, 20]);
    people(i).desire = rand()*10+50;
    people(i).mode = 3;
    people(i).buysell = 1;
    people(i).keep = rand();
end

for i=n_set+n_soc+n_coll+1:n_agents % Assign the values to the profiteers
    people(i).money = (randi([15, 25]))*1e2;
    people(i).nitems = 0;
    people(i).soc_stat = randi([5, 20]);
    people(i).desire = rand()*30+70;
    people(i).mode = 4;
    people(i).buysell = 1;
end

% Assign social status to each item

item_index = 1;

for i=1:n_agents
    for j=1:people(i).nitems
        items_soc(item_index) = randi([8, 10])*people(i).soc_stat/10;
        item_index = item_index + 1;
    end
end

%% Put items for sale and make the socialitees, profiteers and collectors buy and sell.
```

```
% Check the number of sellers & buyers and assign the selles & buyers (sellers = setters & buyers = everyone else in
% the beginning)
nsell = 0;
nbuy = 0;
for i=1:n_agents
    if people(i).buysell == 0
        nsell = nsell + 1;
        sellers(nsell).money = people(i).money;
        sellers(nsell).nitems = people(i).nitems;
        sellers(nsell).socstat = people(i).soc_stat;
        sellers(nsell).mode = people(i).mode;
        sellers(nsell).initmoney = people(i).money;
    else
        nbuy = nbuy + 1;
        buyers(nbuy).money = people(i).money;
        buyers(nbuy).nitems = people(i).nitems;
        buyers(nbuy).socstat = people(i).soc_stat;
        buyers(nbuy).mode = people(i).mode;
        buyers(nbuy).desire = people(i).desire;
        buyers(nbuy).keep = people(i).keep;
        buyers(nbuy).bsoc = people(i).soc_stat;
        buyers(nbuy).bdes = people(i).desire;
        buyers(nbuy).initmoney = people(i).money;
    end
end
% Set a price and social status for each brand
    for i=1:nbrands
        brand(i).price = randi([300 600]);
        brand(i).socstat = randi([70 100]);
```

```
        end


% Put the brand to each seller (setter)
h = 0;
w = 0;
for i=1:nbrands
        h = h + sperbrand(i);
      for j=1+w:h
            sellers(j).brand = i;
      end
      w = h;
end

sigmaprice = 0.05;
ks = 2.5; %constant for the standard deviation for
     the selling and buying limit prices
sigma = ks*sigmaprice;
for i=1:n_set
        for j=1:nbrands
              if sellers(i).brand == j
                     for k=1:sellers(i).nitems
                           sellers(i).items(k).price = brand(j)
                               .price*brand(j).socstat/80;
                           sellers(i).items(k).socstat =
                               normrnd(sellers(i).socstat, 0.5);
                           sellers(i).items(k).sold = 0;
                           sellers(i).items(k).time = 0; %This
                               controls when was de object
                               introduced.
                           sellers(i).items(k).brand = sellers(
```

```
                    i).brand;
                end
            end
        end
end

% The probability of a socialitees buying an item depends on the social
% status of the trendsetter that owns the item. Therefore, we need to make
% that the trendsetters with lower social status put their items for sale
% with a lower price so that they get bought

%% Parameters to change something, such as the increase or decrease of the buying prices and selling prices, etc
Pc = 0.7; %probability an item is bought or sold (If a random number is higher than this value, then the item will be bought. If it is lower, it won't be bought
i = 0; % Checks the amount of iterations
lowp = 35; %How much we lower the selling and buying prices of the agents who bought or sold something
highp = 40; %How much we increase the prices in case the agent didn't do anything
newitem = randi([45 60]); %How frequent we insert new items into the model
avgt = zeros(timesteps, round(timesteps/newitem)+1); %Preallocate the matrix containing the
```

```
averages of all the items from one time period
ipt = zeros(timesteps, round(timesteps/newitem)+1); %Preallocate the matrix for the amount of items from one time period
vin = rand(nbrands, 1)*2 + 2; %Get the vintageness of the items once for the first iterations
moneypagent = zeros(3, 1); %Save the amount of money interchanged by each agent. The order is
                                        % 1 - trendsetters, 2 - socialites, 3 -
                                        % profiteers

    for j=1:nbuy %Go through all the buyers and assign them their buy prices
        buyers(j).bprice = price*normrnd(1.01, sigma)*buyers(j).desire/100; %This buying price gets affected by the desire of each agent to buy the good
    end

%% Iterate

while ~isempty(sellers) && ~isempty(buyers) && i<timesteps
    i = i + 1;
        % Introduce new items each 20 timesteps
    if mod(i, newitem) == 0
        for j=1:length(sellers)
            if sellers(j).mode == 1
                sellers(j).nitems = sellers(j).nitems + round(normrnd(nitems/
```

```
                    n_set, 0.1));
                for f=1:nbrands
                    if sellers(j).brand == f
                        for h=1:sellers(j).nitems
                            sellers(j).items(h).price = brand(f).price*brand(f).socstat/80;
                            sellers(j).items(h).socstat = normrnd(sellers(j).socstat, 0.1);
                            sellers(j).items(h).sold = 0;
                            sellers(j).items(h).brand = sellers(j).brand;
                            sellers(j).items(h).time = i/newitem;
                        end
                    end
                end
            end
        end
        newitem = randi([45 60]);
    end

    for j=1:length(buyers)
        buyers(j).bprice = max(buyers(j).bprice, 0);
    end
```

```

    for j=1:nbuy
        buyers(j).bprice = min(buyers(j).money, buyers(j).bprice);
    end

        spriceavg = 0;
    bpriceavgsoc = 0;
    bpriceavgcoll = 0;
    bpriceavgprof = 0;

    % Get the average buying and selling prices
    a = 0;
    c = 0;
    b = 0;

    avgb(i, :) = zeros(1, nbrands);
    ipb(i, :) = zeros(1, nbrands);
    u = 0;
    for j=1:length(sellers)
        for k=1:sellers(j).nitems
            spriceavg = spriceavg +  sellers(j).items(k).price;
            u = u + 1;
            avgt(i, sellers(j).items(k).time+1) = avgt(i, sellers(j).items(k).time+1) + sellers(j).items(k).price;
            ipt(i, sellers(j).items(k).time+1) = ipt(i, sellers(j).items(k).time+1) + 1; %Items per time
            avgb(i, sellers(j).items(k).brand) =
```

```
                avgb(i, sellers(j).items(k).brand) + sellers(j).items(k).price;
            ipb(i, sellers(j).items(k).brand) = ipb(i, sellers(j).items(k).brand) + 1; % Items per brand
        end
    end
    spriceavg = spriceavg/u;
    for j=1:length(buyers)
        if buyers(j).mode == 2
            c = c + 1;
            bpriceavgsoc = bpriceavgsoc + buyers(j).bprice;
        end
        if buyers(j).mode == 3
            a = a + 1;
            bpriceavgcoll = bpriceavgcoll + buyers(j).bprice;
        end
        if buyers(j).mode == 4
            b = b + 1;
            bpriceavgprof = bpriceavgprof + buyers(j).bprice;
        end

    end

    for j=1:size(avgt, 2)
        avgt(i, j) = avgt(i, j)./ipt(i, j);
    end
    for j=1:size(avgb, 2)
```

```
            avgb(i, j) = avgb(i, j)./ipb(i, j);
        end
        bpriceavgsoc = bpriceavgsoc/c;
        bpriceavgprof = bpriceavgprof/b;
        bpriceavgcoll = bpriceavgcoll/a;

        avg(i, 1) = spriceavg;
        avg(i, 2) = bpriceavgsoc;
        avg(i, 3) = bpriceavgcoll;
        avg(i, 4) = bpriceavgprof;
        iter(i) = i;

        % Compare buying & selling prices and determine whether a purchase is
        % made or not based on a probability
        avgsoldprice(i) = 0;
        ntrans(i) = 0;
        for k=1:nsell % loop over all the sellers and see if any buying price is below the associated selling price
            for j=1:nbuy % Loop over the buyers & as soon as someone buys from the seller, stop the iteration
                for  h=1:sellers(k).nitems
                    if (buyers(j).bprice>=sellers(k).items(h).price && rand()>Pc && buyers(j).bought == 0)
                        if rand()<(sellers(k).items(h).socstat/100)
                            ntrans(i) = ntrans(i) + 1;
                            if buyers(j).mode ~= 3 % If
```

```
the buyer is not a collector, everything works fine
% If there is a purchase, remove the item from the seller &
% put it in the buyer. Subtract the money used for buying
% from the buyer account & put it in the seller account
sellers(k).nitems = sellers(k).nitems - 1;
buyers(j).nitems = buyers(j).nitems + 1;
soldprice = (buyers(j).bprice+sellers(k).items(h).price)/2;
buyers(j).money = buyers(j).money - soldprice;
sellers(k).money = sellers(k).money + soldprice;
buyers(j).items(buyers(j).nitems).socstat = normrnd(sellers(k).items(h).socstat, 1);
```

```
buyers(j).items(buyers(j).nitems).brand = sellers(k).items(h).brand;
buyers(j).items(buyers(j).nitems).time = sellers(k).items(h).time;
if buyers(j).mode ~= 4
  buyers(j).items(buyers(j).nitems).price = sellers(k).items(h).price + (rand()+1)*buyers(j).items(buyers(j).nitems).socstat*4;
end
if buyers(j).mode == 4
    buyers(j).items(buyers(j).nitems).price = soldprice + (rand()+1)*buyers(j).items(buyers(j).nitems).socstat*2;
end
buyers(j).bought = 1; % Set a flag so that that buyer doesn't buy anymore this
```

```
iteration
            sellers(k).sold = 1; % Set a flag to know that the seller has sold
            sellers(k).items(h) = [];

        avgsoldprice(i) = avgsoldprice(i) + soldprice;
        if sellers(k).mode == 1
            moneypagent(1) = moneypagent(1) + soldprice;
        end
        if sellers(k).mode == 2
            moneypagent(2) = moneypagent(2) + soldprice;
        end
        if sellers(k).mode == 4
            moneypagent(3) = moneypagent(3) + soldprice;
            sellers(k).gain = sellers(k).gain + soldprice;
        end
        if buyers(j).mode == 4
            buyers(j).gain = buyers
```

```
(j).gain - soldprice;
                end
                break;
            end
            if buyers(j).mode == 3 &&
               sellers(k).items(h).
               socstat >= 40
                    % If there is a
                      purchase, remove
                      the item from the
                       seller &
                % put it in the buyer.
                  Subtract the money
                  used for buying
                % from the buyer
                  account & put it in
                  the seller account
                sellers(k).nitems =
                  sellers(k).nitems -
                  1;
                buyers(j).nitems =
                  buyers(j).nitems + 1;
                soldprice = (buyers(j).
                  bprice+sellers(k).
                  items(h).price)/2;
                buyers(j).money =
                  buyers(j).money -
                  soldprice;
                sellers(k).money =
                  sellers(k).money +
                  soldprice;
```

```
buyers(j).items(buyers(j).nitems).socstat = normrnd(sellers(k).items(h).socstat, 1);
buyers(j).items(buyers(j).nitems).brand = sellers(k).items(h).brand;
buyers(j).items(buyers(j).nitems).time = sellers(k).items(h).time;
if buyers(j).mode ~= 4
  buyers(j).items(buyers(j).nitems).price = sellers(k).items(h).price + (rand()+1)*buyers(j).items(buyers(j).nitems).socstat*5; %this code is whack
end
if buyers(j).mode == 4
    buyers(j).items(buyers(j).nitems).price = soldprice + (rand()+1)*buyers(j).items(buyers(j).nitems).socstat
```

```
*3;
                end
                buyers(j).bought = 1; % Set a flag so that that buyer doesn't buy anymore this iteration
                sellers(k).sold = 1; % Set a flag to know that the seller has sold
                sellers(k).items(h) = [];
            avgsoldprice(i) = avgsoldprice(i) + soldprice;
            if sellers(k).mode == 1
                moneypagent(1) = moneypagent(1) + soldprice;
            end
            if sellers(k).mode == 2
                moneypagent(2) = moneypagent(2) + soldprice;
            end
            if sellers(k).mode == 4
                moneypagent(3) = moneypagent(3) + soldprice;
                sellers(k).gain =
```

```
sellers(k).gain + soldprice;
                    end
                    if buyers(j).mode == 4
                        buyers(j).gain = buyers(j).gain - soldprice;
                    end
                        break;
                    end
                end
                %buyers(j).bought = 1; % This says that the buyer didnt buy the thing because of its socials status and not because of the money
            end
          end
      end
  end
    avgsoldprice(i) = avgsoldprice(i)/ntrans(i);

%% Adjust the price in each agent depending on whether they bought or sold or not
    for j=1:length(sellers)
        for k=1:sellers(j).nitems
            if sellers(j).items(k).sold == 0
                sellers(j).items(k).price = sellers(j).items(k).price - 70;
                if sellers(j).mode == 4
                    sellers(j).items(k).price = max(
```

```
                    sellers(j).bprice, sellers(j).
                    items(k).price);
                  % Profiteers won't sell it lower
                     than what they bought it
                  % for, they don't want to lose
                    money
                end
            end
        end
    end

    for j=1:length(buyers)
        buyers(j).socstat = buyers(j).bsoc;

        for k=1:buyers(j).nitems
            buyers(j).socstat = buyers(j).socstat +
                0.2*buyers(j).items(k).socstat;
        end
    end
    for j=1:length(sellers)
         if sellers(j).mode ~= 1
             sellers(j).socstat = sellers(j).bsoc;
            for k=1:sellers(j).nitems
                sellers(j).socstat = sellers(j).
                   socstat + 0.2*sellers(j).items(k).
                   socstat;
            end
         end
    end

%% Updating Desire of agents
```

```
% The desire of the socialitees decreases as
    the have more socstat. But
% in increases when they have less social
    status. For the profiteers,
% the desire goes with the difference between
    their initial money and
% their money now
for j=1:length(buyers)
    if buyers(j).mode == 2 %Only if they are
        socialitees
        buyers(j).desire = buyers(j).bdes + (
            buyers(j).bsoc - buyers(j).socstat)
            *0.9;
    end
    if buyers(j).mode == 4 %Only the
        profiteers
        buyers(j).desire = buyers(j).bdes - (
            buyers(j).money - avgsoldprice(i))
            *0.01;
    end
    if buyers(j).mode == 3 %Collectors
        buyers(j).desire = buyers(j).bdes + (
            buyers(j).bsoc - buyers(j).socstat)
            *0.1;
    end
    buyers(j).desire = max(buyers(j).desire,
        0);
    buyers(j).desire = min(buyers(j).desire,
        100);
end
for j=1:length(sellers)
```

```
            if sellers(j).mode == 2 %Only if they are socialitees
                sellers(j).desire = sellers(j).bdes + (sellers(j).bsoc - sellers(j).socstat)*0.9; %Their desire lowers when they have a high social status and viceversa, it increases when they have a lower social status
                sellers(j).desire = max(sellers(j).desire, 0);
                sellers(j).desire = min(sellers(j).desire, 100);
            end
            if sellers(j).mode == 4 %Only the profiteers
                sellers(j).desire = sellers(j).bdes - (sellers(j).money - sellers(j).initmoney)*0.01; %Same as socialitees but
                sellers(j).desire = max(sellers(j).desire, 0);
                sellers(j).desire = min(sellers(j).desire, 100);
            end
        end

        for j=1:length(buyers)
            buyers(j).avgiprice = 0;
        end
        for j=1:length(buyers)
            if buyers(j).mode == 4
```

```
                for k=1:length(buyers(j).items)
                    buyers(j).avgiprice = buyers(j).avgiprice + buyers(j).items(k).price;
                end
                buyers(j).avgiprice = buyers(j).avgiprice/length(buyers(j).items);
            end
        end

%% Conversion of buyers and sellers
        % Change all the buyers that do not have enough money (based on the avg
        % selling price) and make them sellers. Collectors do not become
        % sellers, as they don't want to sell.
        k = 0; %k indicates the amount of buyers that have to be deleted
        svindex = []; %Saves the indices that have to be deleted
        nsell = length(sellers);
        nbuy = length(buyers);
        for j=1:nbuy
            if (buyers(j).nitems > 0 && buyers(j).mode == 4 && buyers(j).avgiprice > 1.1*avgsoldprice(i))
                k = k + 1;
                sellers(nsell+k).money = buyers(j).money;
                sellers(nsell+k).socstat = buyers(j).socstat;
```

```
            sellers(nsell+k).nitems = buyers(j).nitems;
            sellers(nsell+k).gain = buyers(j).gain;
            sellers(nsell+k).bprice = buyers(j).bprice; %This is so sellers remember the price they bought the item for
            sellers(nsell+k).items = buyers(j).items;
            for h=1:buyers(j).nitems
                sellers(nsell+k).items(h).sold = 0;
            end
            sellers(nsell+k).mode = buyers(j).mode;
            sellers(nsell+k).initmoney = buyers(j).initmoney;
            sellers(nsell+k).bsoc = buyers(j).bsoc;
            sellers(nsell+k).bdes = buyers(j).bdes;
            sellers(nsell+k).sold = 0;
            svindex(k) = j; %Store the index of the buyer that has to be deleted
        end
        if (buyers(j).money < avgsoldprice(i)*1.5 && buyers(j).nitems > 0 && buyers(j).mode == 2) || (buyers(j).mode == 2 && rand() < 0.2 && ~isempty(buyers(j).items))
            k = k + 1;
            sellers(nsell+k).money = buyers(j).money;
            sellers(nsell+k).socstat = buyers(j).socstat;
            sellers(nsell+k).nitems = buyers(j).nitems;
```

```
            sellers(nsell+k).gain = buyers(j).gain;
            sellers(nsell+k).bprice = buyers(j).bprice; %This is so sellers remember the price they bought the item for
            sellers(nsell+k).items = buyers(j).items;
            for h=1:buyers(j).nitems
                sellers(nsell+k).items(h).sold = 0;
            end
            sellers(nsell+k).mode = buyers(j).mode;
            sellers(nsell+k).initmoney = buyers(j).initmoney;
            sellers(nsell+k).bsoc = buyers(j).bsoc;
            sellers(nsell+k).bdes = buyers(j).bdes;
            sellers(nsell+k).sold = 0;
            svindex(k) = j; %Store the index of the buyer that has to be deleted
        end
    end

    for j=1:k
        h = svindex(k-j+1);
        buyers(h) = [];
    end

    % Now check if any of the sellers who are not a setter has no items and
    % make them buyers. nsell is updated with the new amount of sellers,
    % but it should not change the buyers who just became sellers, as they
```

```
% would have items, if everything worked correctly
k = 0;
svindex = [];
nsell = length(sellers);
nbuy = length(buyers);
for j=1:nsell
    if sellers(j).nitems == 0 && sellers(j).mode ~= 1
        k = k + 1;
        buyers(nbuy+k).money = sellers(j).money;
        buyers(nbuy+k).nitems = sellers(j).nitems;
        buyers(nbuy+k).socstat = sellers(j).socstat;
        buyers(nbuy+k).mode = sellers(j).mode;
        buyers(nbuy+k).gain = sellers(j).gain;
        buyers(nbuy+k).items = sellers(j).items;
        buyers(nbuy+k).initmoney = sellers(j).initmoney;
        buyers(nbuy+k).bought = 0;
        buyers(nbuy+k).bdes = sellers(j).bdes;
        buyers(nbuy+k).bprice = price*normrnd(1.01, sigma);
        if sellers(j).mode == 2 %The seller is a socialitee
            buyers(nbuy+k).desire = rand()*30+70;

        end
        if sellers(j).mode == 4 %The seller is a
```

```
                    collector
                    buyers(nbuy+k).desire = rand()
                        *20+50;
                end
                buyers(nbuy+k).bsoc = sellers(j).bsoc;
                svindex(k) = j;
            end
        end
        for j=1:k
            h = svindex(k-j+1);
            sellers(h) = [];
        end
        nbuy = nbuy + k;
        nsell = nsell - k;

    for j=1:nbuy
        if buyers(j).bought == 0
            buyers(j).bprice = buyers(j).bprice +
                normrnd(highp*buyers(j).desire, 90)/50;
        else
            buyers(j).bprice = buyers(j).bprice -
                normrnd(lowp/buyers(j).desire, 0.1)*50;
        end
    end

    for j=1:nbuy
        buyers(j).bought = 0; %Restart the bought
            flag for the next iteration
    end
    for j=1:nsell
        sellers(j).sold = 0; % Restart the sold flag
```

```
            for the next iteration
    end
    % Check that at least one seller has items to
       sell & at least one buyer
    % has money to buy

     % Give some money to the people that don't
        have enough money, as if they
     % earned money in their jobs

     for j=1:length(buyers)
         if buyers(j).money < price*2 && buyers(j).
            bought == 0
           buyers(j).money = buyers(j).money + 120;
         end
     end

     %Check if an seller is ready to leave the game
          with the same procedure
     if ~isempty(sellers) && ~isempty(buyers)
          buypos = 0;
          sellpos = 0;
             for j=1:length(sellers)
            if sellers(j).nitems > 0
                sellpos = sellpos + 1;
            end
         end
         for j=1:nbuy
            if buyers(j).money > 0
                buypos = buypos + 1;
            end
```

```
        end
    end

%% Updating Shoe Item Stats
    %In every iteration the social status of the shoe is affected by
    %depretiation and by the vintageness of the shoe (brand)
    %
    if mod(i, 4) == 0
                vin = rand(nbrands, 1)*2 + 2;
    end
    for j=1:length(buyers)
        for k=1:buyers(j).nitems
            buyers(j).items(k).socstat = buyers(j).items(k).socstat - (rand()+1)*3.5; %The social status of the object is affecteed by depretiation
            buyers(j).items(k).socstat = buyers(j).items(k).socstat + (rand()+1)*vin(buyers(j).items(k).brand); %The social status is affected by the vintageness of the brand
        end
    end

    % Delete the items that have a really low social status as nobody is
    % going to buy them
    itdel = zeros(1, 2); %his array keeps the index of the person and the index of the
```

```
item tobe removed. First person, then item
    a = 0;
    for j=1:length(buyers)
        for k=1:length(buyers(j).items)
            if buyers(j).items(k).socstat <= 7
                a = a + 1;
                itdel(a, 1) = j;
                itdel(a, 2) = k;
            end
        end
    end
    for j=1:a
        i1 = itdel(end-j+1, 1); %This variable keeps the buyers from which the item has to be deleted
        i2 = itdel(end-j+1, 2); %This variable keeps the index of the item to be deleted
        buyers(i1).items(i2) = [];
    end
    for j=1:length(buyers)
        buyers(j).nitems = length(buyers(j).items);
    end
    for j=1:length(sellers)
        if sellers(j).mode ~= 1
            for k=1:sellers(j).nitems
                sellers(j).items(k).socstat = sellers(j).items(k).socstat - (rand()+1)*3.5; %The social status of the object is affected by depretiation
```

```
                sellers(j).items(k).socstat = sellers(j).items(k).socstat + (rand()+1)*vin(sellers(j).items(k).brand); %The social status of the item is affected by the vintageness of the brand
            end
        end
    end
    % Delete the items that have a really low social status as nobody is
    % going to buy them
    itdel = zeros(1, 2); %his array keeps the index of the person and the index of the item tobe removed. First person, then item
    a = 0;
    for j=1:length(sellers)
        for k=1:length(sellers(j).items)
            if sellers(j).items(k).socstat <= 7
                a = a + 1;
                itdel(a, 1) = j;
                itdel(a, 2) = k;
            end
        end
    end
    for j=1:a
        i1 = itdel(end-j+1, 1); %This variable keeps the seller from which the item has to be deleted
        i2 = itdel(end-j+1, 2); %This variable keeps the index of the item to be
```

```
            deleted
         sellers(i1).items(i2) = [];
      end
      for j=1:length(sellers)
         sellers(j).nitems = length(sellers(j).items
            );
      end

      %Shift the buyers & the sellers so they
        randomize a little bit

     rndb = randperm(length(buyers));
     buyers = buyers(rndb);
     rnds = randperm(length(sellers));
     sellers = sellers(rnds);
%      buyers = circshift(buyers, 100);
%      sellers = circshift(sellers , 100);

      nbuy = length(buyers);
      nsell = length(sellers);

%% Plotting Useful Parameters
      figure (1);
      hold off;
      plot(iter , avg(:, 1), 'DisplayName', 'Average
        total selling price', 'Linewidth', 1);
      hold on;
      drawnow;
      plot(iter , avg(:, 2), 'DisplayName', '
        Socialitee average buying price', 'Linewidth
        ', 1);
```

```
    hold on;
    drawnow;
    plot(iter, avg(:, 3), 'DisplayName', 'Collectors average buying price', 'Linewidth', 1);
    hold on;
    drawnow;
    plot(iter, avg(:, 4), 'DisplayName', 'Profiteers average buying price', 'Linewidth', 1);
    hold on;
    drawnow;
    legend ('Location', 'northwest');
    title('Average Item Price at Each Time')
    xlabel('Time')
    ylabel('Price')
     figure(2);
    hold off;
    for j=1:floor(i/newitem)+1
        plot(iter, avgt(1:i, j), 'DisplayName', strcat('Time ', num2str(j-1)));
        hold on;
        drawnow;
    end
    legend ('Location', 'northwest');
    figure(3);
    hold off;
    for j=1:(size(avgb, 2))
        plot(iter, avgb(:, j), 'DisplayName', strcat('Brand', num2str(j)));
        hold on;
```

```
        drawnow;
    end
    legend ('Location', 'northwest');
end
figure(6);
hold off;
plot(iter, avgsoldprice, 'DisplayName', 'Average transaction price');
hold on;
legend('Location', 'northwest');
gain = zeros(n_prof, 1);
c = 0;
for i=1:length(buyers)
    if buyers(i).mode == 4
        c = c + 1;
        gain(c) = buyers(i).gain + buyers(i).nitems*avg(end, 1);
    end
end
for i=1:length(sellers)
   if sellers(i).mode == 4
      c = c + 1;
      gain(c) = sellers(i).gain + sellers(i).nitems*avg(end, 1);
   end
end
figure(7);
area(gain);
xlim([0 n_prof]);
xlabel('Profiteers')
ylabel('Net Profit')
```

```
title('The End Profit of Each Profiteer');
figure(5);
area(gain);
xlim([0 n_prof]);
figure(6);
subplot(2, 1, 1);
    hold off;
    plot(iter, avg(:, 1), 'DisplayName', 'Average total selling price', 'Linewidth', 1);
    hold on;
    drawnow;
    plot(iter, avg(:, 2), 'DisplayName', 'Socialitee average buying price', 'Linewidth', 1);
    hold on;
    drawnow;
    plot(iter, avg(:, 3), 'DisplayName', 'Collectors average buying price', 'Linewidth', 1);
    hold on;
    drawnow;
    plot(iter, avg(:, 4), 'DisplayName', 'Profiteers average buying price', 'Linewidth', 1);
    hold on;
    drawnow;
    ylabel('Price [Currency Units]');
    legend;
subplot(2, 1, 2);
    plot(iter, ntrans, 'DisplayName', 'Number of transactions');
```

```
        ylabel('Transaction');
        xlabel('Time [Days]');
end
```