Computational Physics

Project 5

UiO : University of Oslo

Hunter Wilhelm Phillips (*hunterp*)

December 14, 2018

GitHub Repository at

https://github.com/robolux/Computational_Physics

Abstract

This project explores the simulation of financial transactions between financial agents using Monte Carlo methods. Through the selection and advancement of equations, a distribution of income as a function of the income *m* was reduced. The simulation was ran with varying parameters such as savings, nearest neighbor interactions, and former transactions. The plots provided valuable insight into the financial market and allowed for unique distribution types to be observed. The results concluded in the confirmation of a successful Monte Carlo algorithm being ran to simulate the financial transactions among financial agents.

1. Discussion of Problem Statement

In this project, the goal is to model monetary transactions between individuals using Monte Carlo Methods. First, the theory is derived and critical equations are to be selected for use in the algorithm. Second, the program that runs the financial transaction simulation is to be created and verified. Third, a savings rate on the individual level should be added into the algorithmic framework. Fourth, nearest neighbor interactions between individuals is to be integrated into the simulation. Fifth, the probability of future transactions through former transactions is coupled with the nearest neighbors to form the final addition to the algorithm. Plots are to be generated throughout to visualize and analyze the data and conclude in a detailed understanding of financial transactions. Through this, the tasks at hand have been laid out and the theory and methods explaining the solution are described in *Section 2*.

2. Discussion of Theory & Methods

2.1 Monetary Transactions

The base of this project lies in the core equations that govern the chosen representation of monetary transactions. For this simulation, the set of equations used is from [1]. Each agent in the simulation conducts a transaction during the current state. The money that is conserved during the transaction is defined as (1).

$$m_i' + m_j' = m_i + m_j \tag{1}$$

Where i and j is money exchanged in pairs

m' is the amount of money the agents currently have

m is the amount of money the agents had before the transaction

The financial exchange is completed through the use of a random reassignment factor ϵ with (2) and (3) being the result:

$$m_i' = (\epsilon)\left(m_i + m_j\right) \tag{2}$$

$$m_j' = (1 - \epsilon)\left(m_i + m_j\right) \tag{3}$$

To maintain a stable system, (1) adheres to the conservation law with the equilibrium state being given as a Gibbs distribution in (4) and (5).

$$w_m = \beta e^{-\beta m} \tag{4}$$

$$\beta = \frac{1}{\langle m \rangle} \tag{5}$$

Where $\langle m \rangle$ is the expected wealth

Negative wealth is not possible with this model and $m \geq 0$ at all times.

2.2 Pareto Distribution

The distribution of wealth in economics has been widely explored throughout time. Vilfredo Pareto wrote his description of wealth and income distribution through power law probability distributions and derived (6) for the higher end of the wealth spectrum [2].

$$w_m \propto m^{-1-\upsilon} \rightarrow \upsilon \in [1, 2] \tag{6}$$

υ is Pareto's 80-20 rule value

The 80-20 rule championed by Pareto concludes that 20% of the population controls 80% of the wealth.

2.3 Savings and Transactions

The model can be further developed by adding a savings rate λ for the agents. This amount of money is withheld from transactions to ensure the individual maintains minimal economic stability. This new wealth calculation can be seen in (7) and (8)

$$m_i' = \lambda m_i + (\epsilon)\,(1-\lambda)\left(m_i + m_j\right) \tag{7}$$

$$m_j' = \lambda m_j + (1-\epsilon)\,(1-\lambda)\left(m_i + m_j\right) \tag{8}$$

A simple rewrite of (7) and (8) can yield (9), (10), and (11):

$$m_i' = m_i + \delta m \tag{9}$$

$$m_j' = m_j - \delta m \tag{10}$$

$$\delta m = (1-\lambda)\left(\epsilon m_j - (1-\epsilon) m_i\right) \tag{11}$$

2.4 Wealth Neighbors and Former Transactions

It is assumed that wealthy individuals like to complete transactions with other wealthy people. This mindset trickles down the economic framework and a probability can be introduced for more likeminded interactions (12).

$$p_{ij} \propto \left|m_i - m_j\right|^{-\alpha} \tag{12}$$

with α being a value > 0

With individuals also liking to do business with agents they have previously completed financial transactions with, (12) can be trivially modified into (13) to include a higher probability of trading if previous financial transactions have occurred.

$$p_{ij} \propto \left|m_i - m_j\right|^{-\alpha} (c_{ij} + 1)^{\gamma} \tag{13}$$

where c_{ij} is the number of previous transactions between agents

with γ being a value > 0

3. Discussion of Algorithms

3.1 Trading

This is the core function in the Python program providing where the actual transactions take place. The following code snippet contains the overarching looping structure with agent initialization before transactions have occurred.

Initializing Agents

```
for i in range(0, transaction_count):

    ag_i = random.randint(0, N-1)
    ag_j = random.randint(0, N-1)

    m_i = agents[ag_i]
    m_j = agents[ag_j]

    rn = random.random() # random number
    previous = startt[ag_i, ag_j]
```

One of the most critical parts of the algorithm lies in the logic behind the likelihood of a transaction taking place. This uses the theory we derived to logically produce a valid looping condition.

Transaction Probability

```
varz1 = nmp.power(nmp.absolute(m_i-m_j), -alpha)*nmp.power(previous+1, gamma)

while ((varz1 < rn) | (ag_i == ag_j)):

    ag_i = random.randint(0, N-1)
    ag_j = random.randint(0, N-1)

    m_i = agents[ag_i]
    m_j = agents[ag_j]

    previous = startt[ag_i, ag_j]

    rn = random.random()
```

A random value is then selected and the savings amount calculated. Another key part is the cumulative registering that a transaction took place for future probabilities.

Savings & Successful Transaction Registering

```
epsilon = random.random()

delta_m = (1 - lambdaa) * (epsilon * m_j - (1 - epsilon) * m_i)
agents[ag_i] = agents[ag_i] + delta_m
agents[ag_j] = agents[ag_j] - delta_m

startt[ag_i, ag_j] += 1
startt[ag_j, ag_i] += 1

tempo = agents
cum_1 += tempo          # cumulative
cum_2 += tempo*tempo
```

To test when an equilibrium state has been reached, a criterion had to be developed. It was decided that the squared variance would be computed. This would then be logically compared using variability from [3] and reference computations from [4].

Equilibrium State Logicals

```
if (i % block_length == 0):

    avg_tempo  = cum_1 / block_length
    avg_tempo2 = cum_2 / block_length
    tempo_block = avg_tempo2 - avg_tempo*avg_tempo
    testcond1 = nmp.absolute(avg_tempo_old - avg_tempo) /
    nmp.absolute(avg_tempo_old)
    testcond2 = nmp.absolute(tempo_block - tempo_block_old) /
    nmp.absolute(tempo_block_old)

    if ((testcond1.any() < 0.2686) & (testcond2.any() < 0.4791)):

        break

    else:
        avg_tempo_old = avg_tempo
        tempo_block_old = tempo_block

    cum_1  = 0
    cum_2  = 0
```

3.2 Wrapper

To ease running an extended amount of simulations, a function called wrapper was built to perform the simulation looping, agent averaging, and returning the results.

Wrapper Implementation

```
def wrapper(m_init, N, tran_count, sim_count, lambdaa, alpha, gamma):
    agents = nmp.zeros(N)
    totagents = nmp.zeros(N)
    random.seed(a=None) # start random set
    start = rolex.time()

    for i in range(0, sim_count):
        print('progress: ' + str(i))
        agents.fill(m_init)
        agents = trading(N, tran_count, agents, lambdaa, alpha, gamma)
        totagents += nmp.sort(agents)

    agents = totagents/sim_count
    end = rolex.time()
    tot_time = end - start

    return agents, tot_time
```

3.3 Parametrization

One key aspect was to find a set of equations that successfully parametrize the income distributions. This ended up coming from [1] and allowed for nice and simple statements while plotting in Python. By characterizing the reduced variable, (14) is formed:

$$x = \frac{m}{\langle m \rangle} \tag{14}$$

where m is the money

$\langle m \rangle$ is the average money

Using the associated savings value, , to obtain a forward facing (15):

$$n(\lambda) = 1 + \frac{3\lambda}{1-\lambda} \tag{15}$$

A normalization factor is then needed (16):

$$a_n = \frac{n^n}{\Gamma(n)} \tag{16}$$

where a_n is normalization factor

$\Gamma(n)$ is the gamma function

This combines together to form (17):

$$P_n(x) = a_n x^{n-1} e^{-nx} \tag{17}$$

This allowed for a smooth characterization as seen below in the Python implementation.

Parametrization Implementation

```
lambdaa = 0.9
x = agents / np.mean(agents)
n = 1 + (3*lambdaa)/(1-lambdaa)
an = (n**n) / gamma(n) # gamma function
P = an*(x**(n-1))*np.exp(-n*x)

plt.loglog(x,P,'r')
```

4. Discussion of Results

4.1 System Specifications

A few notes about the system running the algorithms before analyzing the results should be helpful in determining performance between different computers as seen in *Table 1*.

Operating System	Mac OS X
Processor	i7 - 2.2 GHz
Memory	8 GB 1600 MHz DDR3
Hard drive	128 GB SSD

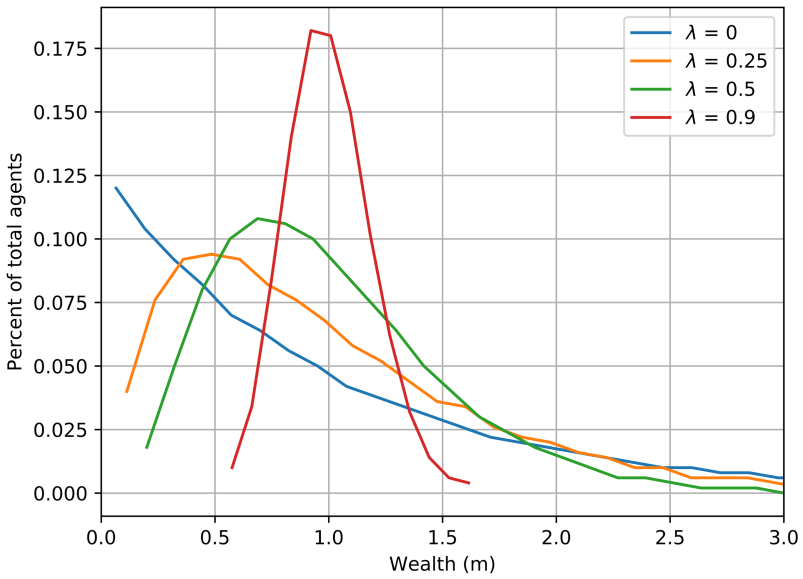
Table 1: System Specifications

With the conditions under test being established we can now dig deeper into the data recorded.

4.2 Simulation of Transactions with Savings

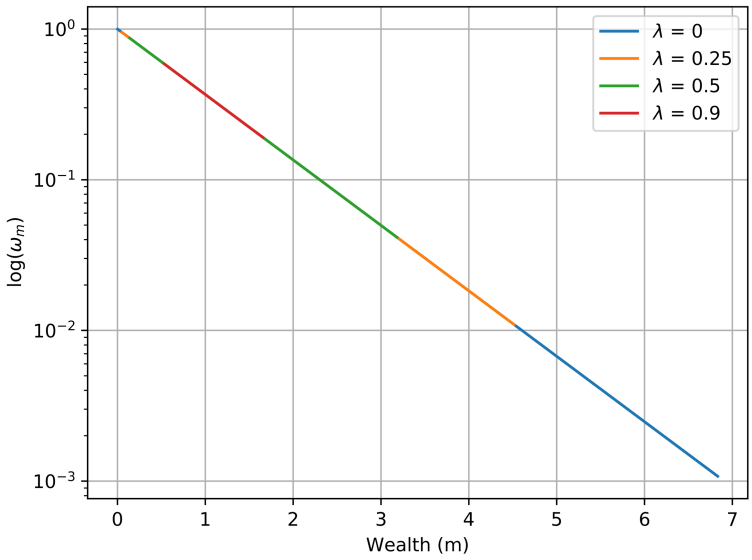
The first test of the project was to simulate a standard case of financial transactions with the algorithm that was developed. For a better viewpoint, the savings rates were varied to show the spectrum of results. When $\lambda = 0$ it indicates that there is no saving occurring. The results can be seen in *Figure 1*.

Figure 1: Financial Transaction Simulation with varying saving rates (λ) with N=500, 1000 simulations, $m_0 = 1$



It can be seen that there is a decrease in the number of agents as the wealth increases for minimal savings cases. When the savings amount is increased it can be seen that the curve peaks at the initial amount of money, in this case, 1. These are the results expected, and by taking a look at figure 1 in [1] it could be effectively perceived as an identical copy. To additionally verify that everything else is working correctly, a plot of *log* (w_m) as function of *m* was generated as seen in *Figure 2*.

Figure 2: Financial Transactions y-axis log plot with varying saving rates (λ) **with N=500, 1000 simulations, $m_0 = 1$**



The linear result is as expected and verifies that the distribution being shown is correct and a verifiable result.

4.3 Parametrization

Using the parametrization identified in *Section 3.3* the following plots were generated showcasing the close match they provided. *Figures 3 & 4* were selected to show two drastically different cases with savings rates on opposite ends of the spectrum.

Figure 3: Extracted Parametrization and Simulated Financial Transactions with N=500, 1000 simulations, $m_0 = 1$, and $\lambda=0.9$

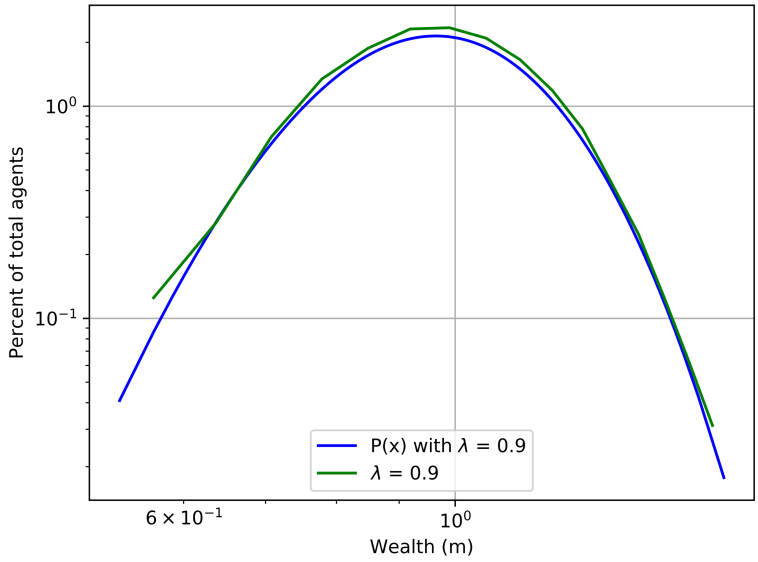
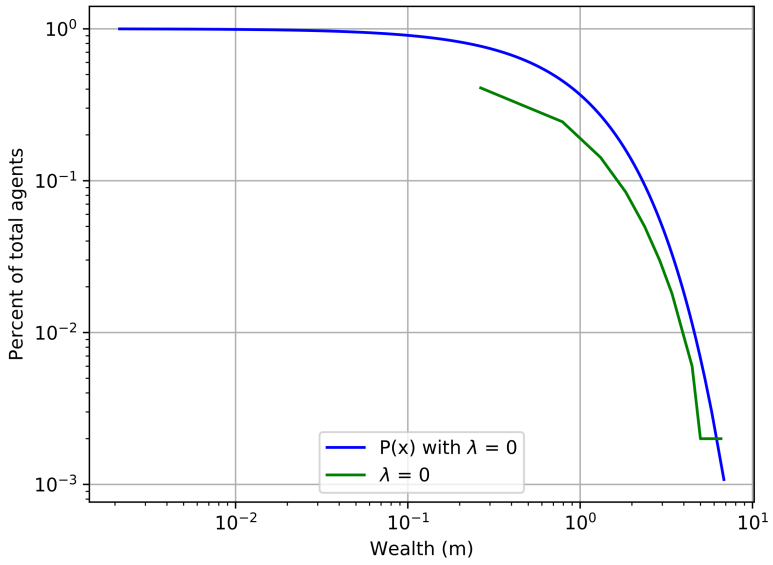


Figure 4: Extracted Parametrization and Simulated Financial Transactions with N=500, 1000 simulations, $m_0 = 1$, and $\lambda=0$

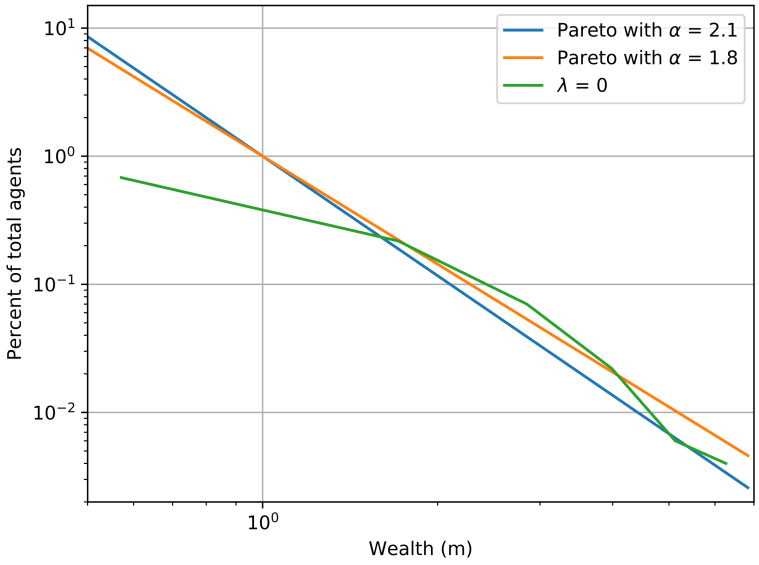


It can be seen the parametrization holds true to the curves quite nicely. It appears to work better with the higher savings rate, possibly due to a tendency towards parabolic structures in its underlying mathematical framework.

4.4 Pareto Power Law Tails

To see the high end of distributions with respect to their power law tails, *Figure 5* was generated.

Figure 5: Pareto Power Law Tails and Simulated Financial Transactions with N=500, 1000 simulations, $m_0 = 1$, and $\lambda=0$



The initial positional gap can be observed between the simulated curve and power law tails. This gap closes when the wealth is increased, showing higher accuracy with the high-end of the distribution. This was the result expected and validated that the power law tails were accurate with greater wealth.

4.5 Addition of Nearest Neighbor Interactions

With nearest neighbor interactions added the simulation was ran again with varying amounts of agents and values of α. The key here that is expected is that wealthier individuals will only desire to complete transactions with other people. This thought is expanded across the complete realm of economics with respect to financial transactions. The results can be seen in *Figures 6 & 7.*

Figure 6: Financial Transactions Simulation with neighbor interactions (α) with 1000 simulations, m_0 = 1, λ=0, and N=500 (left) and N=1000 (right)

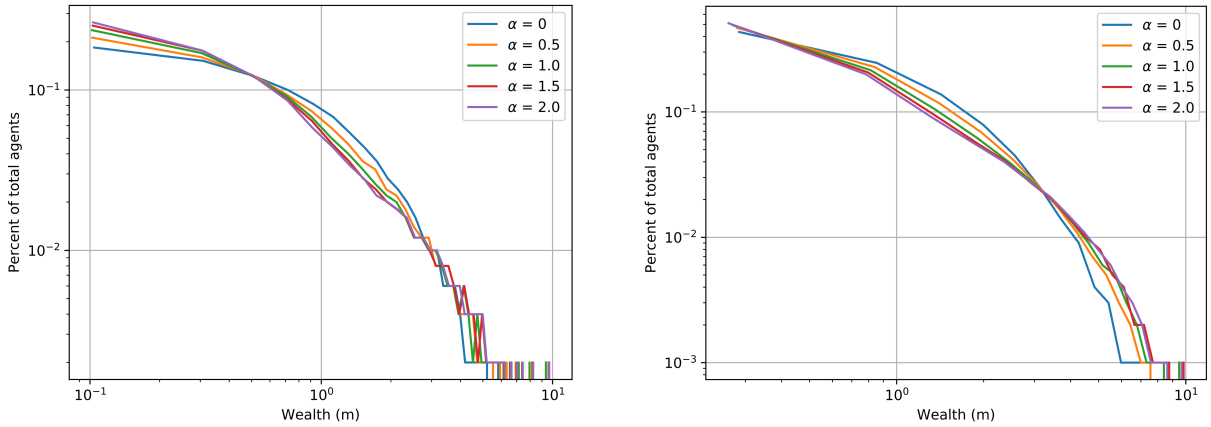
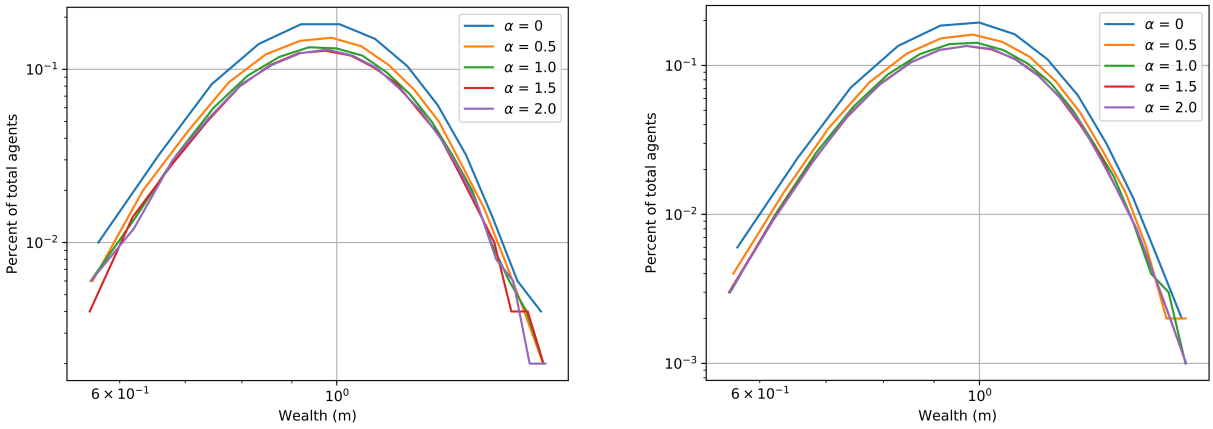


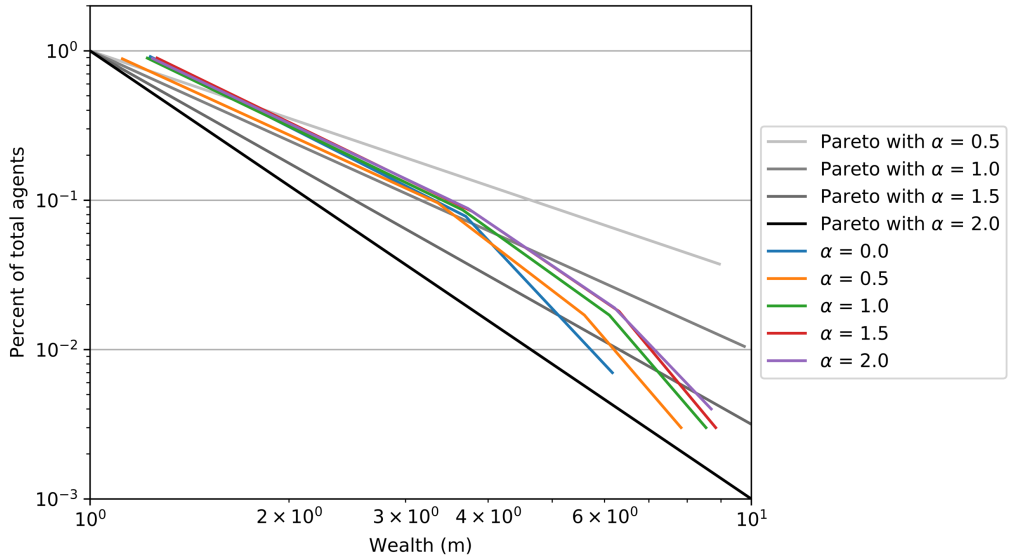
Figure 7: Financial Transactions Simulation with neighbor interactions (α) with 1000 simulations, m_0 = 1, λ=0.9, and N=500 (left) and N=1000 (right)



These results were as expected and you can see the effect that wealth has on neighbor interactions. With the increased number of agents, it can be seen that the conditional convergence or divergence is reached faster and with greater statistical accuracy. Both plots in *Figure 6* a fairly accurate representation of figure 1 in [5].

The Pareto power tails were then plotted with the nearest neighbor transactions and is showcased in *Figure 8*.

Figure 8: Pareto Power Law Tails for Varying α and Simulated Financial Transactions with N=500, 1000 simulations, $m_0 = 1$, and $\lambda=0$



4.6 Addition of Previous Transactions

By adding the ability for former transactions with other individuals to influence the probability of future transactions the results can be further analyzed. The resulting plots can be seen in *Figures 9 & 10*.

Figure 9: Financial Transactions Simulation with Previous Transaction History (γ) with 1000 simulations, $m_0 = 1$, $\lambda=0$, N=1000, and $\alpha = 1$ (left) and $\alpha = 2$ (right)

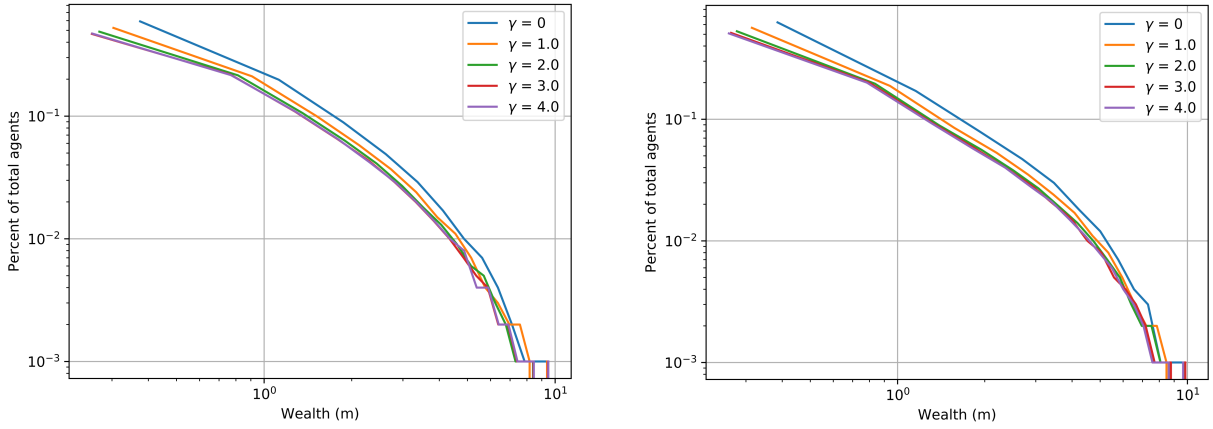
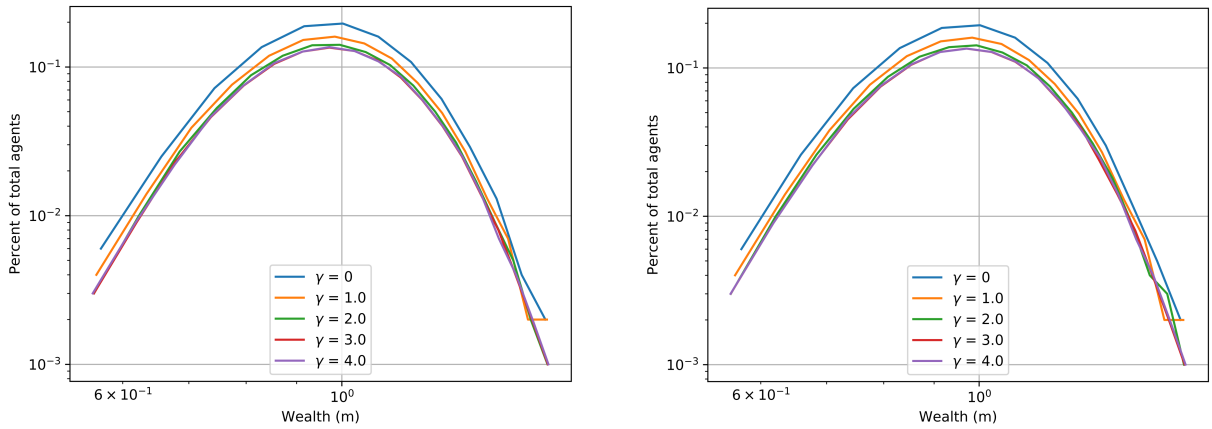
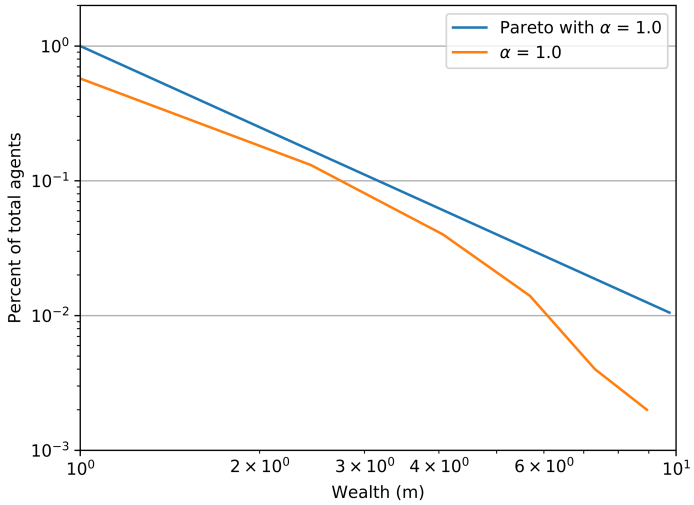


Figure 10: Financial Transactions Simulation with Previous Transaction History (γ) with 1000 simulations, $m_0 = 1$, λ=0, N=1000, and $\alpha = 1$ (left) and $\alpha = 2$ (right)



The same tightening convergence can be observed with larger α value. This corresponds with figures 5 and 6 from [5]. The power tail was then plotted as seen in *Figure 11*.

Figure 11: Pareto Power Law Tails and Simulated Financial Transactions with N=500, 1000 simulations, $m_0 = 1$, $\gamma = 2$, and λ=0



There appears to be a slight gap between the tail and simulated curve. This could be due to a myriad of possibilities, but most likely is just a situational zone. There were a lot of parameters that had to be altered to produce the previous transactions and neighbor interactions for these particular cases. This is noted to be a large computational time with the simulations running for over 8 hours. If 10000 simulations were to be ran, it would take an estimated week to complete. This is why the lower end of 1000 simulations was chosen, to ease the load on the hardware.

5. Conclusion

Through this project, a successful Monte Carlo simulation of financial transactions with respect to wealth was created. A program was developed in Python to simulate the system and produce results for analysis. The system proved to be reliable across a gamut of test conditions with an array of situations being tested. The parametrizations and Pareto power law fits verified the literature's content being analyzed. With the addition of savings, neighbor interactions, and previous transaction history, the model was formed into a formidable simulation that represented real-world situations. The Gibbs distribution was observed in almost all cases, which is expected for example: with a low number of high wealth individuals. This project concludes in a comprehensive understanding of financial transactions and how to use the Monte Carlo method coupled with varying parameters to simulate it effectively.

6. Future Work and Thoughts

The reason I chose this project was due to the fact that I could get to experiment with Monte Carlo methods again, since it was my favorite part of the course. I know that Norway is a socialist democracy and the exchanging of money is different than my home country, the United States. I feel like it would be really interesting to somehow model the difference of financial transactions among a more open democracy such as the United States, versus the socialist nature of Norway's financial market. I assume it would be fairly hard to find good mathematical foundations for comparison, but still in theory the concept is intriguing.

As I end this course, I just want to say thank you to everyone involved in putting on this class. Your effort is out of this world and you have expanded my perspectives greatly. Not being a Physics major has made some of the content challenging in this course, but the teachings made everything so much clearer. With my mechanical engineering degree, I hope to be able to implement the computational knowledge I gained in this class and fuse it into robotic designs. Thanks again for an incredible course and I wish everyone involved a wonderful break.

7. References

[1] Patriarca, Marco, et al. “Gibbs versus Non-Gibbs Distributions in Money Dynamics.” Physica A: Statistical Mechanics and Its Applications, vol. 340, no. 1-3, 2004, pp. 334–339.

[2] Pareto, Vilfredo. “Cours D'Économie Politique.” The Annals of the American Academy of Political and Social Science, vol. 9, no. 3, 1897, pp. 128–131.

[3] Persons, Warren M. “The Variability in the Distribution of Wealth and Income.” The Quarterly Journal of Economics, vol. 23, no. 3, 1909, p. 416.

[4] KovacÆevic, Milorad S, and David A Binder. “Variance Estimation for Measures of Income Inequality and Polarization ± The Estimating Equations Approach.” Journal of Official Statistics, vol. 13, no. 1, 1997, pp. 41–58.

[5] Goswami, Sanchari, and Parongama Sen. “Agent Based Models for Wealth Distribution with Preference in Interaction.” Physica A: Statistical Mechanics and Its Applications, vol. 415, 2014, pp. 514–524.