MONTE CARLO SIMULATION OF MONETARY TRANSACTIONS: A SIMPLE MODEL FOR WEALTH DISTRIBUTION

FYS3150: COMPUTATIONAL PHYSICS

TRYGVE LEITHE SVALHEIM
SEBASTIAN G. WINTHER-LARSEN
GITHUB.COM/GREGWINTHER

Abstract. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In id neque elementum, accumsan ligula at, lobortis tortor. Duis elementum pellentesque purus, sit amet euismod diam facilisis volutpat. Nulla facilisi. Mauris quis felis ante. Aliquam ac velit sit amet velit porta condimentum iaculis ut quam. Phasellus pretium libero nec lectus placerat, ac consectetur sem faucibus. Maecenas dictum porta finibus.

Contents

1. Introduction

2. Theoretical Background

2.1. **The simplest model for an economy.** Arguably, the most famous equation in macroeconomics is the "autarky identity"[1]

$$Y = C + G + I, \tag{1}$$

where Y is income, C is consumption, G is government spending and I is investment. The best-know, but not necessarily the best, measure of income Y is the Gross Domestic Product (GDP)[2]. Consumption C is the monetary value of all goods and services purchased in the private sector, while government spending G is the consumption of the government. Finally, investment I is the sum of private and public saving.

Equation 1 is autarkic because all interactions with other economies are excluded from the expression. There are no terms representing exports and captial inflow, for instance. We are dealing with a *closed* economy, alternatively the entire world as a whole. Moreover, let's assume that the economy we are studying is a peaceful anarchy, without a governing authority of any sort, in effect setting $G = 0$. To begin with, we will also forgo the agents the ability to save such that the worth of every individual, or agent, in the economy must be spent at once. Equation 1 is reduced to $Y = C$, everything one agent spends is the income of another.

2.2. **Monetary transactions.** To simulate monetary transactions in our model economy we expand employ the framework introduced in Patriarca et al.[1]. We assume there are N agents that exchange money in pairs (i, j). We assume also that all agents start with the same amount of money $m_0 > 0$. For every period an arbitrary pair of agents are picked at randomly and let them conduct business, id est a transaction takes place between them. Money is conserved during the transaction such that

$$m_i + m_j = m'_i + m'_j, \tag{2}$$

where the right-hand side is the agent i and j's updated wealth and the left-hand side represents the amount of money agents i and j had before the transaction. The exchange is done via a random reassignment factor ϵ, such that

$$m'_i = \epsilon(m_i + m_j) \tag{3}$$

$$m'_j = (1 - \epsilon)(m_i + m_j) \tag{4}$$

No agent will ever have negative wealth, that is $m \geq 0$. Moreover, because of the conservation law in equation 2, the system eventually reaches an equilibrium state given by a Gibbs distribution

$$w_m = \beta e^{-\beta m}, \quad \beta = \frac{1}{\langle m \rangle}, \tag{5}$$

where $\langle m \rangle$ is the expected wealth, for which the arithmetic mean is an unbiased estimator. This implies that after an equilibrium has been reached the majority of agents is left with lower wealth than they had initially and the number of rich agents exponentially decrease.

It is easy to see that the logarithm of the Gibbs measure in equation 5 yields a linear equation

$$\ln(w_m) = \ln(\beta) - \beta m \tag{6}$$

2.3. **Pareto distributions.**

[1] Any introductory text on macroeconomics will give a thorough elaboration, e.g. Gärtner.

[2] GDP = GNP (Gross National Product) in autarky.

2.4. **Transactions and savings.** We are can expand upon the model by introducing a savings rate λ. The savings rate is defined as a fraction of an agent's wealth that does not partake in a transaction for every period. One can gather from the macroeconomic identity in equation 1 that income must still be the same and the transaction law in equation 2 still holds. The updated wealth of agents i and j after a transaction becomes

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

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

one can rewrite these expressions to

$$m_i' = m_i + \delta m$$
$$m_j' = m_j - \delta m$$

where

$$\delta m = (1-\lambda)(\epsilon m_j - (1-\epsilon)m_i) \tag{9}$$

2.5. **Economic inequality and social friction.** "The first historical series of income distribution statistics became available with the publication in 1953 fo Kuznet's monumental Shares of Upper Income Groups in Income and Savings. Kuznet's series dealt with only one country (the United Stetes) over a period of thirty-five years (1913-1948)" (Pikkety & Ganser, 2014, Kindle locations 276-280[3]). Ever since that time there has been wealth and income inequality to a greater or lesser degree. According to the PolitiFact the top 400 richest Americans in 2011 "[had] more wealth than half of all Americans combined"[4].

One of the possible reasons for continued wealth and income inequality that has risen in popularity the last few years is the process of wealth concentration. This is a process by which, under certain conditions, newly created wealth concentrates in the possession of already-wealthy individuals or entities. Those who already have wealth have the means to invest in newly created sources and structures of wealth. Piketty argues that the fundamental force for wealth divergence is the usually greater return of capital than economic growth (Piketty & Ganser, 2014 p. 284 Table 12.2[3]).

To incorporate the social rigidities described above, we will make two further additions to the model. These additions build mostly on the work of Goswami and Sen[5]. Firstly, we will make it more likely for two agents to conduct a transaction if they have similar wealth. Second, we will make it more likely for two agents to make a transaction if they have made a transaction before. We define a probability

$$p_{ij} \propto |m_i - m_j|^{-\alpha}(c_{ij}+1)^{\gamma}, \tag{10}$$

where the first factor $|m_i - m_j|^{-\alpha}$ relates to wealth similarity and the second factor $(c_{ij}+1)^{\gamma}$ relates to previous transactions. A relatively large difference between m_i and m_j should translate to a lower probability. The strength of this effect is governed by the exponent α, a larger value decreases the probability of a trade further. The variable c_{ij} is simply the number of times agent i and j have conducted transactions in the past. The number 1 is added in order to ensure that if they have not interacted earlier they can still interact. The strength of this effect is governed by the exponent γ.

3. Algorithm & Implementation

All programming code used in this project is written in C++ and is publicly available at github by following the link on the first page. The program consists mainly of two files, main.cpp and functions.cpp. main.cpp contains the set-up of the

program and is where the number of agents, initial wealth, number of transactions, number of simulations and all parameters (λ, α and γ) are specified. The main part of the program however, is in functions.cpp which in addition to the trade (...) [3] function contains the function output (...) [4] which writes data to file.

3.1. **The trade (...) function.** This function runs a specified number of Monte Carlo cycles, where the transactions take place. For each iteration two random agents are picked, then some logic follows which will determine if the two agents are allowed to go into business. This is a very important part of the program and is therefore listed here:

```
while ((pow(fabs(m_i - m_j), -alpha)*
    pow(previous_transactions+1, gamma) < random_number)
    || (agent_i == agent_j)) {

  // Pick new agents ...
  agent_i = (int) rand() % N;
  agent_j = (int) rand() % N;

  // ... find wealth of these ...
  m_i = agents(agent_i);
  m_j = agents(agent_j);

  // ... update previous transactions ...
  previous_transactions = C(agent_i, agent_j);

  // ... and update the random comparison number
  random_number = (double) rand() / RAND_MAX;
```

One can see that the the argument of the while loop in the program listing above contains equation 10 which is compared against a random number and an equality test that will return True if agents i and j are the same agent. This part of the code makes sure that a new pair of randomly picked agents until we are satisfied that the agents are not the same and that they are close enough in wealth and in previous relations to go forward with a transaction. After everything is all-right and we have a good pair of agents a transaction takes place:

```
// Random value of transaction b/w 0 and 1
double epsilon = (double) rand() / RAND_MAX;

// Transaction takes place, but some is saved (lambda)
double delta_m = (1 - lambda) *
  (epsilon*m_j - (1 - epsilon)*m_i);
agents(agent_i) += delta_m;
agents(agent_j) -= delta_m;

// Register that a transaction has taken place
C(agent_i, agent_j) += 1;
C(agent_j, agent_i) += 1;
```

[3] The functions have several arguments: trade(int N, int no_of_transactions, arma::vec (& agents), double lambda, double alpha, double gamma). To include all the arguments in the text would have decreased readability and they are replaced with dots.

[4] output (...) = output(int N, arma::vec agents, std :: string filename).

The program listing above should be straight-forward. Notice that the transaction is registered in matrix C by incrementing element i, j and j, i by 1.

The last part of the trade (...) function is not particularly important for the quality of the data that a simulation will provide, but vastly increased the speed of the program. This last functionality breaks the for loop within the trade (...) function when a steady state is reached. Listed in its entirety here:

```
double var = arma::var(agents);
cumVarBlock += var;
cumVar2Block += var*var;

// Enter here at the correct place: end of block
if (i % blockSize == 0) {
  // Variance of a block "averaged" over the block
  double avgVar  =  cumVarBlock / blockSize;
  double avgVar2 = cumVar2Block / blockSize;

  // Variance of a block of variance
  double varVarBlock = avgVar2 - avgVar*avgVar;

  // Check if variance is low enough
  if ((fabs(avgVarBlockOld - avgVar) / fabs(avgVarBlockOld)
          < 0.2) &&
    (fabs(varVarBlock - varVarBlockOld) / fabs(
       varVarBlockOld) < 0.5)) {

    // Telling a user that an equilibrium has been reached
    std::cout << "Equilibrium reached at transaction no. "
       << i << std::endl;

    // Done! For now.
    break;

    } else {
      avgVarBlockOld = avgVar;
      varVarBlockOld = varVarBlock;
    }
  // Reset cumulative variance
  cumVarBlock  = 0;
  cumVar2Block = 0;
}
```

This code section does some calculations on the results after a certain amount of transactions have taken place, an amount predetermined by blockSize. The average variance of the block is computed as well as the average square variance of the block size. Then the "variance of the variance" is computed. These measures are compared against the measures from the previous block. The relative size of the average variance must be less than 20% and the relative size of the variance of the variance must be less than 50% of the measures from the previous block. If both these logical statements evaluate to True, the for loop of the trade (...) function is broken, terminating the current simulation. We found that for most simulations an equilibrium is reached quite quickly, usually between 40000 and 120000 transactions have taken place between $N = 1000$ agents. This speeds up

the time for a simulation to finish tremendously, compared with the initially, and rather naively, picked 10^7 iterations. In order to find correct values blockSize as well as when the variance measures are "low enough" the behaviour of the variance measures were meticulously studied through several trial runs of the function.

3.2. **Example main().** The parameters of the simulation is set up in the main() function. When all parameters are chosen the following for loop is almost everything that is needed

```
// Start simulations
for (int i = 0; i < simulations; i++) {

  // Print progress
  std::cout << "Simulation no. " << i << std::endl;

  // Assign intitial wealth to all agents
  agents.fill(m0);

  // Trade!
  trade(N, transactions, agents, lambda, alpha, gamma);

  // Sort and add equilibrium wealth to total
  totagents += arma::sort(agents);
```

One can see that what must be specified is the number of simulations (simulations), initial wealth (m0), number of agents (N), number of transactions (transactions), an array containing the wealth of every agent (agents), savings rate (lambda), parameter governing weight of wealth similarity (alpha) and parameter governing weight of past transactions (gamma).

4. Results

Figure 1 shows plot of different wealth distributions after 1000 simulations for $N = 500$ agents. We see that that without savings, the distribution shows a decreasing number of agents as wealth increases. As the savings rate λ is allowed to increase, the distribution moves to the right and eventually peaks at the initial wealth $m_0 = 100$ when $\lambda = 0.9$. This plot, were it not for the colors, could be an exact copy of figure 1 in Patriarca et al.[1]. Without savings the distribution appears as a inverse exponential function (e^{-x}) which is the same class of function as the Gibbs distribution (equation 5). However, these traits disappear from the wealth distributions once savings are allowed.

A further study of the wealth distributions from figure 1 is visualized in figure 2. Figure 2a shows the same graphs as in figure 1, but on with logarithmic scales on both axes. Figure 2b the wealth values evaluated by equation 5 and plotted with logarithmic y-axis. Observe that all all the wealth distributions give straight lines. This makes sense, as the logarithm of the Gibbs distribution yields a first order expression (equation 6).

Figure 3a shows two different Pareto distrobutions plotted against a distribution at $\lambda = 0$.

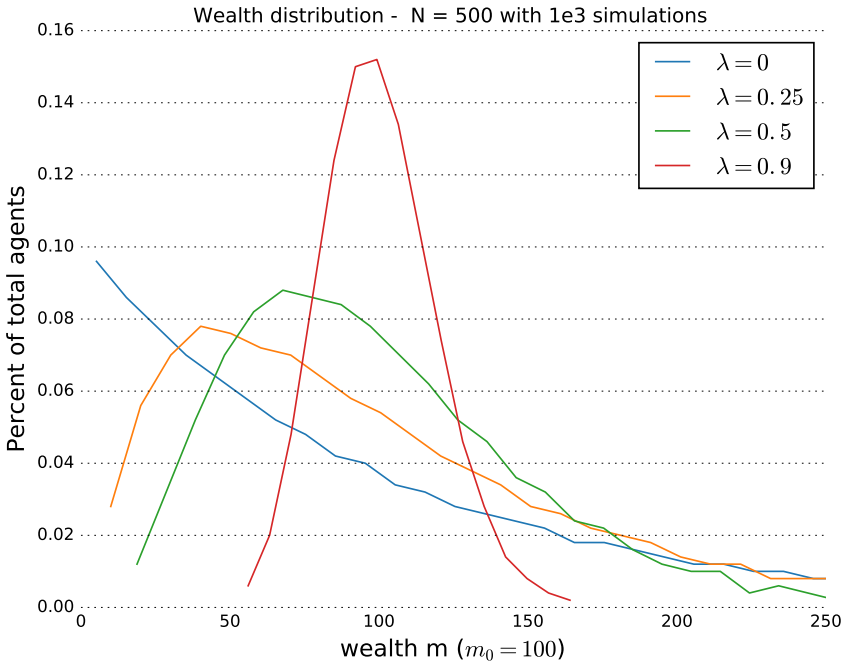
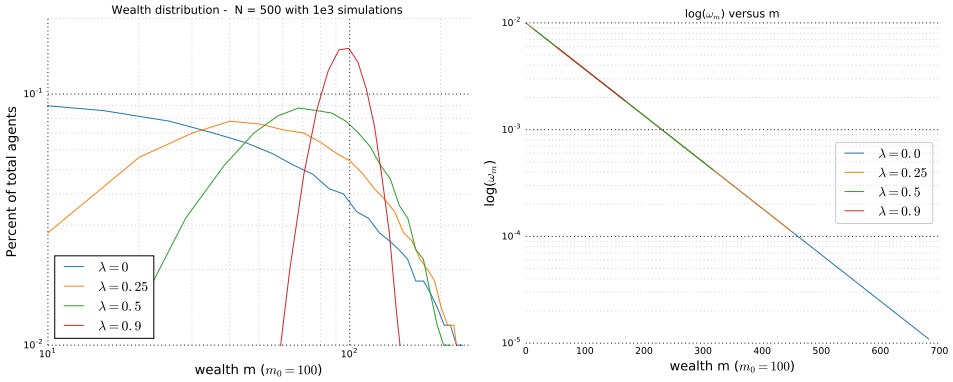


FIGURE 1. Wealth distribution for $N = 500$ agents after 1000 simulations at different savings rates λ



(A) (B)

FIGURE 2. Logarithmic plot wealth distribution from figure 1 (right), and

5. Discussion

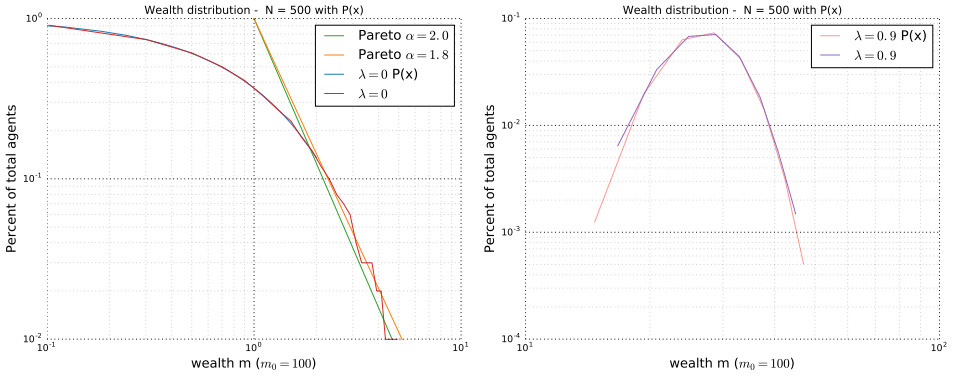
6. Summary Remarks

References

[1] Patriarca, M., Chakraborti, A., & Kaski, K. (2004). Gibbs versus non-Gibbs distributions in money dynamics. *Physica A: Statistical Mechanics and its Applications*, 340(1), 334-339.

[2] Pareto, V. (1897). *Cours d'Économique politique*, Lausanne: Ed. Rouge.

[3] Piketty, T., & Goldhammer, A. (2014). *Capital in the twenty-first century*, Cambridge: Harvard University Press.



(A) (B)

FIGURE 3. Parametrization

[4] Moore, M., (March 7, 2011). The Forbes 400 vs. everybody else. *michaelmoore.com*. Retrieved December 8. 2016.

[5] Goswami, S., & Sen, P. (2014). Agent based models for wealth distribution with preference in interaction. *Physica A: Statistical Mechanics and its Applications*, 415, 514-524.