Simulating Wages with Von Neumann's Model of Economy

Daniel Chin

Dec. 2019

1. Introduction

In this project, I treat labor as a type of goods and use Von Neumann's model of economy to investigate how the evolution of the economy affects the wages.

The research questions that I have some success in answering include:

a) What happens to wages when the economy grows?
b) What happens to wages when the growth rate of the population decreases?
c) What happens to wages when the economy suffers a huge population loss?

2. Model

2.1. Von Neuman's Economy Model

The simulation is a repetition of discrete days.

There are n processes and m goods in the economy. A process takes some input of goods and produces some output of goods, and it takes one day to perform one unit of such transformation. The input and output of all processes are defined respectively in matrix A and matrix B. A and B are both n-by-m matrices.

The state of the economy is described by r, the intensity vector. $r(t)$ is the scale of each process on day t. There are no other states, and goods cannot be preserved between days.

Now, we can compute the supply and demand:

$$supply_j(t) = \sum_{i=1}^{m} r_i(t) B_{i,j}$$

$$demand_j(t) = \sum_{i=1}^{m} r_i(t+1) A_{i,j}$$

Notice that in the $demand$ formula we have $(t+1)$ in place of t. That is because processes demand input goods to prepare for production on the next day.

The $supply$ formula will give us the supply vector for each day given r. The $demand$ formula, however, allows us to compute $r(t+1)$. We will see this later.

After production, the processes trade. They first sell what they have produced at the market prices, p. They earn some revenue this way. Then they spend all the revenue on purchasing the input goods for production on the next day. This means that processes aim at maximizing production. $r(t+1)$ is thus determined.

The last task is to solve for vector p. The requirement for p is that there is no *positive excess demand* for any type of goods. Excess demand for good j:

$$e_j(t) = demand_j(t) - supply_j(t)$$

This means we do not allow the process to require more than how much there is. However, negative excess demand is allowed. This means it is free to dispose goods. Goods with positive excess demand always has price equal to zero.

It turns out there is always at least one solution of p that fulfills the requirement of no positive excess demand. Please see Peskin's notes [1] for the proof using Brouwer's fixed-point theorem.

An interesting observation is that the scale factor for the processes depends on r linealy:

$$\frac{r_i(t+1)}{r_i(t)} = \frac{Revenue_i(t)}{Cost_i(t)} = r_i \frac{\sum_{j=1}^{m} B_{i,j} p_j(t)}{\sum_{j=1}^{m} A_{i,j} p_j(t)}$$

If the prices are in equilibrium, then $\frac{\sum_{j=1}^{m} B_{i,j} p_j(t)}{\sum_{j=1}^{m} A_{i,j} p_j(t)}$ becomes a constant. The economy will grow or shrink exponentially, and the relative intensity between the processed will also be in equilibrium.

2.2. Labor and Wages

This model treats labor as a type of goods, whose prices are what we call wages. There are k different types of labor (people with different trainings offers different types of labor). Let m' denote the number of types of normal goods:

$$m' + k = m$$

A static population model is achieved by adding a constraint on A and B:

$$B_{i,j} = A_{i,j} \text{ for all } i \text{ and all } j \in labor$$

This way, labor is conserved between market days.

The labor columns in A and B simply describe how much labor the processes *require* for one unit of production. The labor is not consumed during the process.

Note that in this static population model labor can still be taken away from the economy when there is an excess supply of labor.

2.3. Population Models

Population is computed by taking the sum of the supply of labor.

$$population_j(t) = \sum_{i=1}^{n} r_i(t) A_{i,j}$$

The population grows or shrinks according to growth rate g:

$$target\_population(t+1) = population(t) \times g$$

g can be higher, lower, or equal to 1.

This mean we can modify the constraint on A and B to

$$B_{i,j} = g \cdot A_{i,j} \text{ for all } i \text{ and all } j \in labor$$

3. Numerical Method

3.1. Setup

In this project I used 10 types of normal goods, 3 types of labor goods, and 3000 processes.

The values in A and B are sampled from the uniform distribution, but with a natural disadvantage to the economy:

$$mean(B) = \frac{mean(A)}{2}$$

After that, the labor constraint is added to update B.

Prices p satisfies $\Sigma p = 1$. Initialization of $p_j = \frac{1}{m}$.

Intensity r is initialized as $r_i = \frac{1}{n}$.

3.2. Timestep

Each day, compute `supply = B .* r;`

Cache `sum_r = sum(r);`

Compute q. See Solving for Prices for procedures.

Update `p = q .^ 2;`

Update `r = sum(supply .* p, 2) ./ sum(A .* p, 2);`

For source code, please see github.com/Daniel-Chin/WageSim/blob/master/aDay.m

3.3. Solving for Prices

I use MATLAB function `fmincon` to minimize

$$\phi(q) = \frac{\sum_{j=1}^{m} \max\left(0, e_j\right)^2}{(\Sigma r)^2}$$

where

$$q_j = \sqrt{p_j}$$

Why ϕ is divided by $(\Sigma r)^2$ will be discussed in Modification of $\boldsymbol{\phi}$.

I use q instead of p because p has sharp boundaries, which may hurt minimization performance.

`fmincon` is given a constraint that q must remain on the unit sphere:

$$\Sigma q_j^2 - 1 = 0$$

The initial guess for `fmincon` is $q(t) = q(t-1)$.

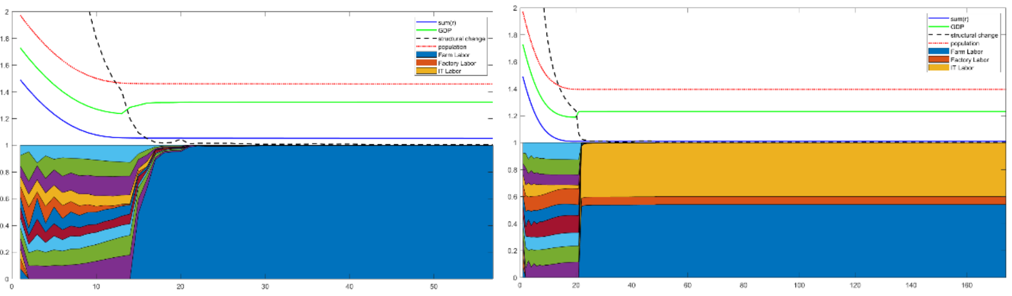
If `fmincon` gives a minimum that is not zero, I randomize q and try again. Randomization of q is based on the uniform distribution on the surface of the unit hypersphere with rejection sampling:[1]

```
while norm(y) > 1
  y = rand(1, dim);
end
```

4. Experiment Results and Analysis

4.1. Static Population

In this section, population growth $g = 1$.



The x axis is time in unit of day.

The lower part ($y \in [0, 1]$) of the graph is prices. Each color is a type of goods.

Population, GDP, and sum(r) are plotted in log-linear. The base of the log is chosen such that all curves will fit within $y \in [1,2]$.

Structural change is $\left\|\frac{d\frac{r}{\|r\|}}{dt}\right\|$. It should be zero when the prices are in equilibrium. Its plot is shifted so that $y = 1$ means structural change $= 0$.

As we can see from the above two runs, at first wages are 0, and GDP is decreasing. But, at some point, the economy starts climbing up and wages become 1, and then GDP stagnates.

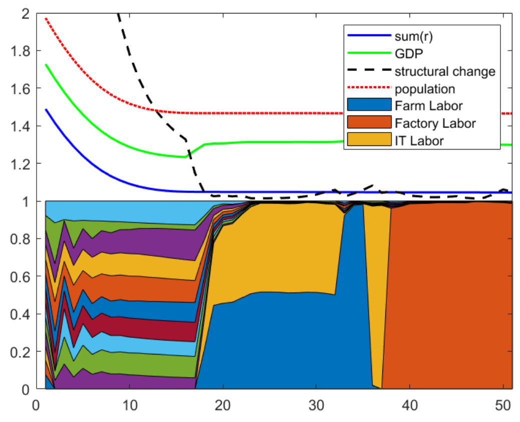
My explanation is as follows. At first, because A and B are set to give the economy a disadvantage, the economy shrinks rapidly. As time go by, worse processes scale down, and better processes survive. This period sees very high structural change. Soon, as long as there are enough good processes to form a working economy, the economy starts to grow. However, because population is limited and labor is required by all processes, the economy eventually becomes labor-bounded and stops growing.

During the first phase, the economy wants to shrink while the population wants to maintain constant. There is fewer and fewer jobs for everyone. As a result, there is an excess supply of labor. Wages are 0, and population decreases. Later, the economy wants to grow while the population wants to stay constant. Labor thus becomes valuable.

[1] The experiment shows that with sphere[13], rejection sampling easily takes thousands of tries.

Wages add up to 1 and normal goods becomes free. Why so extreme? That is because labor is the only finite resource in this simulation. In the real world, a lot of other things are also finite, so normal goods are not free.

We also see that the allocation of wages among three types of labor seems arbitrary.

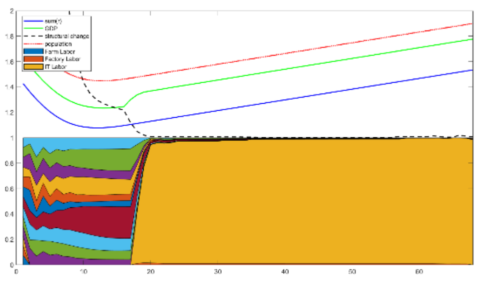


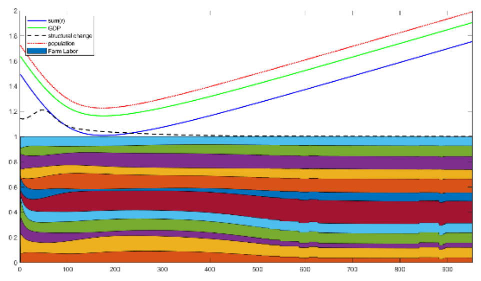
During the above run, the allocation of wages changed multiple times. This usually happens when the prices on the previous day becomes a nonzero local minimum and q is randomized. This means this economy is not at price-equilibrium. It is unclear what caused the local minimum to become nonzero.

4.2. Growing Population

In this section, population growth $g > 1$.

Simulations show that when g is small, there are positive wages, but when g is large, wages are zero.

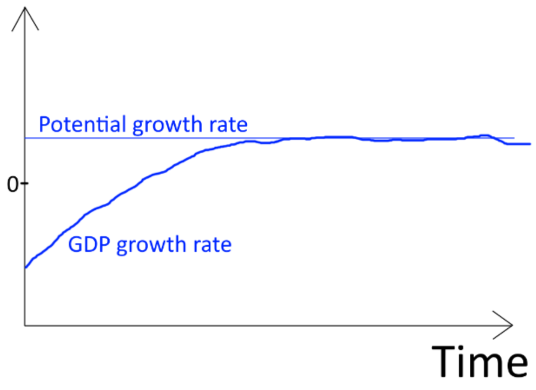




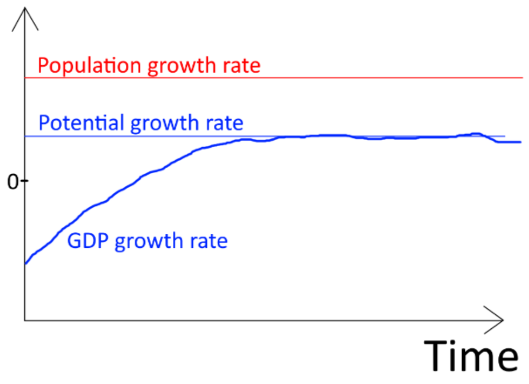
g is small

g is large

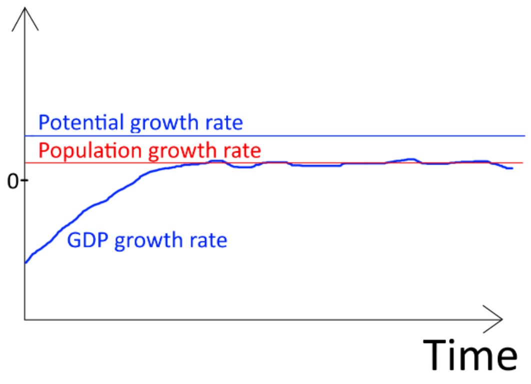
My analysis is as follows. Hypothetically set g to infinity, so the economy is not bounded by labor at all. Observe the growth rate of such an economy.



The growth rate will stabilize at what I call the potential growth rate, R. R is determined by the normal goods part of A and B and not related to labor usage. Now, let us bring back the population constraint.



$g > R$

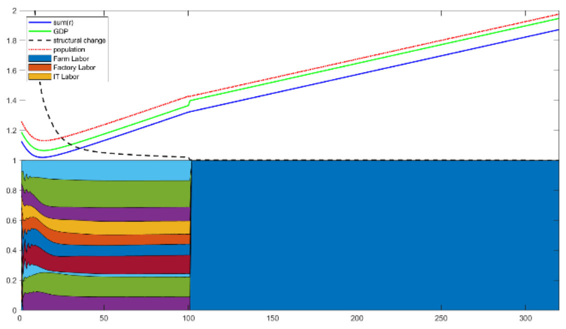


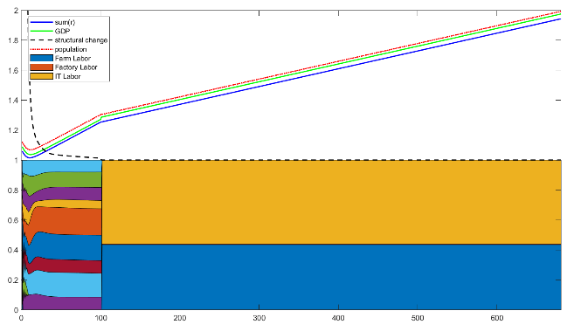
$g < R$

The final GDP growth rate will be either bounded by the potential growth rate or the population growth rate. The only time that wages can be positive is when $g < R$ and after the growth rate catches up with the population growth. That is when population wants to grow but the economy wants to grow faster. At all other times, there is loss of population and wages are zero, because population wants to grow faster than the economy.

4.3. Changing Growth Rate during a Simulation

We can change g and update A and B in the middle of a simulation run. This can be used to model discoveries of contraception technologies or family planning movements.





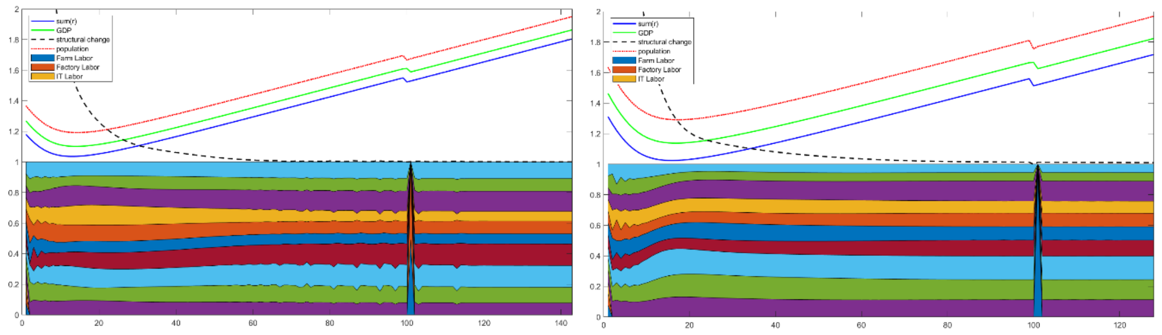
Here, initially $g > R$. At $t > 100$ we decrease g so that $g < R$.

Wages immediately becomes positive. Notice that the GDP growth rate also decrease.

4.4. Sudden Population Loss

We can alter the supply of labor on a particular day. This can be used to represent epidemics or wars.

Before this experiment, I expected the wages to suddenly increase and gradually decrease. The population loss will increase the quantity of resources per person in the short term. The economy will immediately start to shrink and eventually restores equilibrium. During the process, wages will decrease and approach the original level. However, the experiment proved me wrong:



On $t = 100$, I halved the population. The increase of wages lasted for only one day. This is because the model does not allow goods to be preserved. On $t = 100$, there is a huge excess supply of normal goods, and they evaporate from the economy before $t = 101$.

5. Insights on the Numerical Method

The above are the results of the simulations. In this section I will share some discoveries regarding the numerical method, for those who are interested.

5.1. Invisible Hand to Compute Prices

Instead of minimizing the ϕ function, I tried to use the Invisible Hand Method to find p.[2] The Invisible Hand Method increases prices when there is excess demand and decreases prices when there is excess supply, similar to real life. It is as if we were doing a gradient descend on p, but using equation 16 from Professor Peskin's Note [1], instead of the gradient of ϕ.

$$\frac{dp_j}{d\tau} = e_j$$

The above is a simplification. The full implementation involves some regularization and normalization: see github.com/Daniel-Chin/WageSim/blob/master/invisibleHand.m

5.1.1. First Algorithm

Denote stride length as α.

$$p_j(\tau + 1) = p_j(\tau) + \alpha \cdot E_j(p_j(\tau))$$

[2] I tried that because initially the way I modelled population breaks one of the assumptions for Walras's Law and the minimization method unfortunately became unavailable.

If a type of goods changes from excess supply to excess demand, call it an “edge cross”. If there is no edge cross, increase $\alpha \leftarrow 1.1 \times \alpha$. If there is edge cross, decrease $\alpha \leftarrow 0.5 \times \alpha$, and the stride is *not* taken (i.e. cancel the p update).

In practice, this algorithm is frequently stuck with two types of goods competing near $e_j = 0$. For example, when goods 1 has $e = 0$, goods 2 has $e > 0$, but raising the price of goods 2 will increase e of goods 1. A visual analogy would be: gradient descend in a “V”-shaped valley. The goal is to reach the lower end of the valley. But because each time we trigger an edge cross we do not take the stride, and when we are near the edge the gradient always point towards the edge, we end up stuck in one place.

5.1.2. Second Algorithm

I modify the first algorithm. Now, the strides are taken even when there is edge cross.

The second algorithm works most of the time. However, occasionally the process goes into a loop. Please see video youtu.be/Jkt58LF5nkU.

To solve that problem, I tell the program randomize p and try again whenever a loop is detected. Interestingly, no matter how we randomize p, the Invisible Hand Method always falls into the same loop.

5.1.3. Result

The Invisible Hand Method does not work reliably for this project. I eventually use minimization. See Acknowledgement.

5.2. Stiffness Problem

In many runs of the simulation, the prices of goods have the problem of oscillating with higher and higher magnitude. The period of the oscillation is two days. When that happens, the economy usually stops growing.

The oscillation happens when the intensity vector responds to processes’ performance too stiffly. For example, goods 1 may turn out to be cheap on day 1, so the processes specializing in producing goods 1 scale down. That leads to a shortage of goods 1 on day 2, so the processes specializing in producing goods 1 will scale back up. This is a stiffness problem.

In order to solve the stiffness problem, I have tried four methods: stick r, inertia, lowering sparseness, and increasing the number of processes.

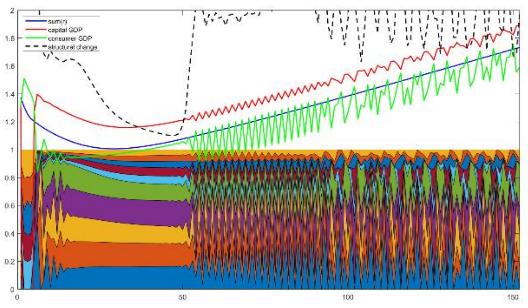
5.2.1. Sticky r Method

This method slows down the movement of r. If $r(t-1)$ is the intensity vector for the previous day and $r'(t)$ is the intensity vector we would get using the vanilla simulation, then the sticky r method sets

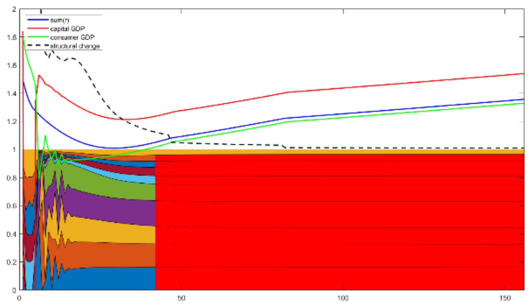
$$r(t) = \lambda \cdot r(t-1) + (1-\lambda) \cdot r'(t)$$

i.e. a weighted average of the old value and the target value.

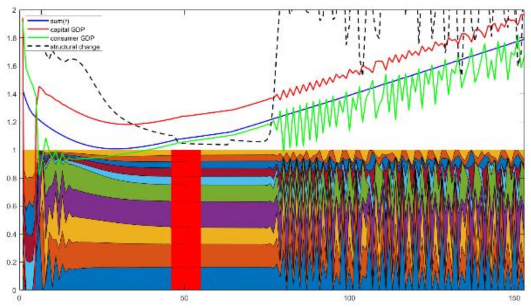
The Sticky r method proves to be effective in delaying the dawning of market oscillations. However, this could just result from the fact that the economy under sticky r *progresses more slowly*. To test the true effect of Sticky r, I change λ during the simulation to see if it is able to “cure” an already oscillating market.



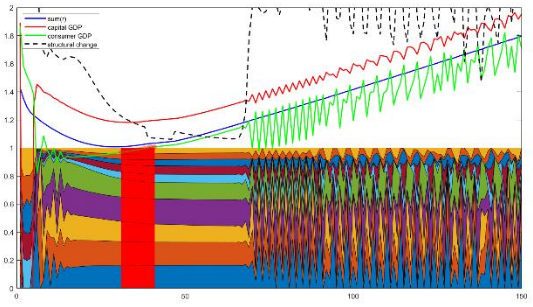
The above is the baseline, i.e. not applying sticky r.



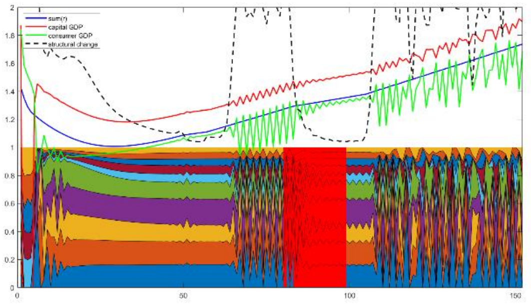
The red area denotes setting $\lambda = 0.5$. As we can see, sticky r gets rid of all oscillation until day 150.



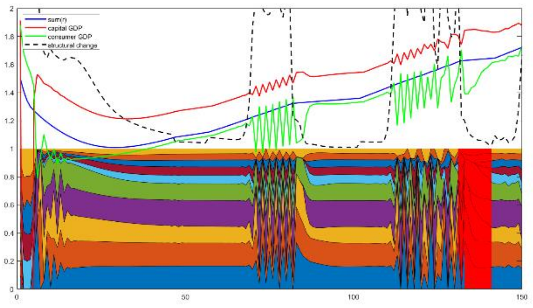
The above plot shows that applying sticky r in the early stage of oscillation stops the oscillation. After setting $\lambda = 0$, the economy stays stable for 20 days, but eventually starts oscillating again. This demonstrate the "curing" effect of sticky r – it is not just postponing the oscillation.



In the above run, I applied sticky r even earlier, before the oscillation occurs. This also delays the appearance of market oscillation.



In the above run, I applied sticky r from day 45 to 55, and day 80 to 100 (marked as red).



In the above run, I applied sticky r from day 45 to 55, day 80 to 100, and day 130 to 140 (marked as red). We see that sticky r indeed corrects the economy. However, its effect is not long-lasting.

In practice, I avoid using sticky r whenever possible. Here are some problems with sticky r. First of all, it introduces an extra parameter of the economy, λ. Von Neumann's model is very elegant partly because it requires little "magic numbers". Introducing more parameters makes researchers cherry-pick before they even realize it. Secondly, I am afraid that the economy may somehow learn to use sticky r to cheat for resources. Thirdly, sticky r slows down the evolution of the economy, making every simulation run require more time.

5.2.2. Inertia Method

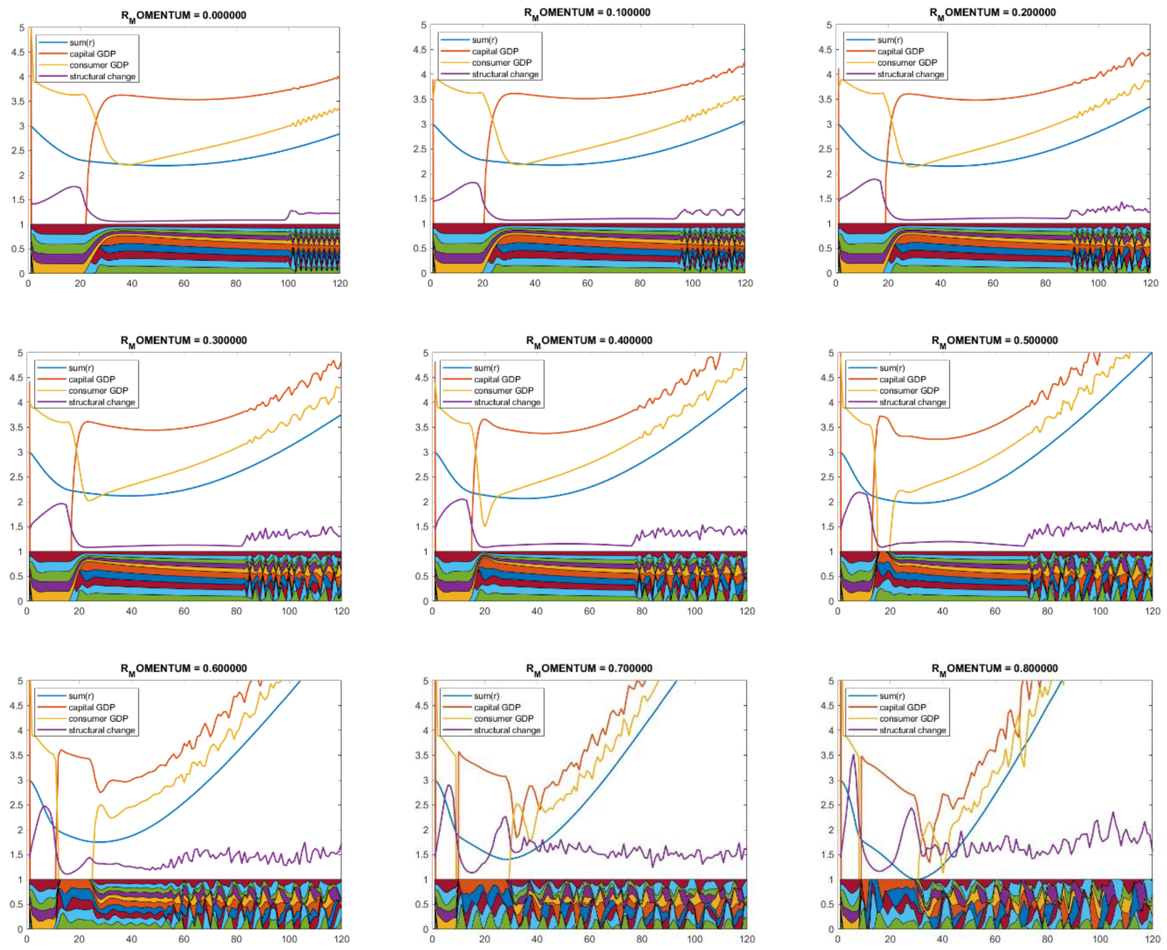
As is explained earlier, market oscillation happens because the economy's response to price changes is too stiff. For example, $r_i(t)$ is too small, so $r_i(t+1)$ becomes very large, but what r_i really just wants to be in the middle. It repeats, and we see an oscillation with period $=\ 2$ days.

Again, using r' to denote the intensity vector we would get using the vanilla simulation, I propose:

$$r_i(t) = r_i'(t) + \mu(r_i(t-1) - r_i(t-2))$$

This simply means to take the movement of r vector from the previous day, multiply by μ, and apply it to the new r. Ideally, when r moves back and forth, this will bring the economy to the middle; when r moves in a direction, this accelerates r's movement!

Unfortunately, that is wishful thinking. I soon realize that the above method behaves much like *inertia* in physics: the economy wants to stay moving at the same speed as it did in the previous day. As we know in Physics, adding mass to an oscillator does not stop the oscillation. Let us see how the inertia method affects the economy. "R$_M$OMENTUM" in the below plots refers to μ:



As we can see, increasing μ only changes the oscillation from period $= 2$ to period > 2.

As a conclusion, the inertia method does not work at all.

5.2.3. Lowering Sparseness

Initially I set B to be a sparse matrix. This is because I want to approximate the reality where processes do not produce a little bit of everything. However, this contributes to the stiffness problem. If a process relies on the sales of only one specific type of goods, then it is very sensitive to the price of that type of goods.

Lowering the variance of B makes the processes more resilient and prices more stable. According to my observation, this is by far the most effective way to reduce stiffness.

5.2.4. Increasing the Number of Processes

I observe that the order of magnitude of the number of processes seems related to the probability of market oscillation appearing before day 600. How this works is not yet clear. I hypothesize that having more processes means a higher chance for more diversified processes to appear, and hence lower stiffness.

5.3. Modification of ϕ

At first, I used this function to minimize excess demand:

$$\phi'(q) = \sum_{j=1}^{m} \max(0, e_j)^2$$

MATLAB minimization function `fmincon` returns `fval` as the minimum value. My program raises a non-zero local minimum alert when `fval` $> 10^{-3}$. I was very confused to see `fval` gradually going up as the simulations ran – and the simulations would always be stuck at around $t = 500$ because every minimum the program found was non-zero. It was as if the economy was evolving to become tricky to simulate. How bizarre!

The mystery is solved when you realize ϕ' is proportional to $(\Sigma r)^2$. As the economy grows, ϕ' grows too! The "non-zero local minima" I found were valid global minima after all, because when the scale of the economy goes to 10^{10}, a little bit of excess demand ≈ 0 is a valid approximation.

To make things consistent, I changed ϕ' to:

$$\phi(q) = \frac{\sum_{j=1}^{m} \max(0, e_j)^2}{(\Sigma r)^2}$$

This time, `fval` stays in the same order of magnitude as the simulation goes on.

6. Future Work

- Dynamic λ. Make the program monitor the oscillation in the economy and automatically adjust λ in the sticky r method.
- AI to replace human labor. Introduce Artificial Intelligence as an alternate input for processes. Investigate the changes in wages of the three types of labors when AI replaces only one type of labor. Does the overall welfare of the people increase? Is there more inequality between professions? What happens when AI replaces all types of labor?
- Introduce finite natural resources. In this writeup, wages often add to 1. Introducing other finite resources may bring us a reasonable split between prices of labor and prices of normal goods.
- Introduce capital goods. For example, under static population, it will be interesting to see the following scenario:

  At the beginning, capital is low, so labor-intensive processes grow.

  Later, capital accumulates, and the population constraint becomes prominent, so the world now favors capital-intensive processes.

7. Acknowledgement

I would like to thank Professor Peskin for his lectures on simulations, his help, and his feedbacks on this project. I would like to thank TA Sun for his insight on labor as conserved goods. Source code and relevant documents can be found here: github.com/daniel-chin/WageSim

8. References

Notes on Economic Growth and Price Equilibrium, by Prof. C. Peskin. www.math.nyu.edu/faculty/peskin/modsim_lecture_notes/economic_growth_and_price_equilibrium.pdf.