Price Equilibrium with Saving and Borrowing

Yuqian Zhang (yz6572@nyu.edu)
Yi Shi (ys4451@nyu.edu)

December 2022

Contents

1 Introduction

In this model, we choose the discrete model based on basic dynamic version of Von Neumann's economic growth model and follow the evolution of the model over time. Additionally, to make the model closer to reality, we include one more potential factor: "saving and borrowing" in our model. In this case, we are interested in by adding the saving or borrowing, what changes will we see in price equilibrium, excess demand, and the rate of growth?

Similarly to the basic model mentioned in class, we approach by starting the model with a collection of processes and goods, representing inputs and outputs respectively. Then we add saving and borrowing into the model with interest rate and build new budget constraint and excess demand equations. To relate saving or borrowing with market day, we use a fraction of the intensity of certain processes to represent the saving or borrowing.

By setting the initial condition of the fraction, we will analyze the new model with saving and borrowing under the condition that both saving and borrowing is allowed in the same time and the company can choose to save or borrow based on the budget (which is a bit more realistic than simply adding saving or borrowing separately): On the one hand, when the revenue from the last process is larger than the next process's consumption, we save the money. On the other hand, When the revenue is smaller than or equal to the next process's consumption, we borrow the money.

By plotting the plot of price, intensity and excess demand that change with the market day based on different values of interest rate and saving/borrowing, we compare our model with the basic discrete model to analyze the possible reason for changes.

2 Assumptions and notations

Symbol	Definition
i	process index from 1 to n
j	good index from 1 to m
A	consumption matrix with the size $n \times m$
B	production matrix with the size $n \times m$
t	time
$p_j(t)$	price of one unit of good j at time
$r_i(t)$	intensity for the production period $(t-1, t)$
$e_j(t)$	excess demand for good j

Table 1. Symbols and Definitions

To begin with, we define n processes and m goods and use the $n \times m$ matrix A, B, which are both independent of time t (market day) to characterize the economy of the i-th process, with $A_{ij} > 0$, $B_{ij} \geq 0$. Then, we define the price of good j at time t as $p_j(t)$, which changes with time. Let $r_i(t)$ be the intensity of the i-th process during the production period $(t-1, t)$, and $r_i(t) > 0$.

Thus in the producing period $(t, t+1)$, the consumption of money in the i-th process is $\sum_{j=1}^{m} r_i(t+1)A_{ij}p_j(t)$, and the revenue got from selling the product in the i-th process is $\sum_{j=1}^{m} r_i(t)B_{ij}p_j(t)$.

We define the excess demand for good j as the difference value between production and sales, and we need to minimize the excess demand.

Additionally, we add the interest rate denoted as λ and use the fraction c of $r_i(t)$ to represent the saving or borrowing as $s_i(t) = cr_i(t)$

3 Basic model

3.1 Equations and explanations

The model starts at $t = 0$, but the market day occurs at $t = 1$

For the production period $(t-1, t)$, consumes $r_i(t)A_{ij}$ units of good j at time $t-1$ and produces $r_i(t)B_{ij}$ units of good j at time t.

For a whole production and consumption process including three days $t-1, t$, and $t+1$, the firm produces $r_i(t)B_{ij}$ units of good during the period $(t-1, t)$ and produce $r_i(t+1)A_{ij}$ units of good j in the period $(t, t+1)$

$$\sum_{j=1}^{m} r_i(t+1)A_{ij}p_j(t) = \sum_{j=1}^{m} r_i(t)B_{ij}p_j(t) \tag{1}$$

Thus we get the iterated equation of $r_i(t+1)$:

$$r_i(t+1) = r_i(t)\frac{\sum_{j=1}^{m} B_{ij}p_j(t)}{\sum_{j=1}^{m} A_{ij}p_j(t)} \tag{2}$$

For excess demand:

$$\begin{aligned} e_j(t) &= \sum_{i=1}^{n} r_i(t+1)A_{ij} - \sum_{i=1}^{n} r_i(t)B_{ij} \\ &= \sum_{i=1}^{n} r_i(t)\left(\frac{\sum_{k=1}^{m} B_{ik}p_k(t)}{\sum_{k=1}^{m} A_{ik}p_k(t)}A_{ij} - B_{ij}\right) \end{aligned}$$

Then we have $e_j(t) = E_j(r(t), p(t))$

$$E_j(p) = \sum_{i=1}^{n} r_i(t)\left(\frac{\sum_{k=1}^{m} B_{ik}p_k(t)}{\sum_{k=1}^{m} A_{ik}p_k(t)}A_{ij} - B_{ij}\right) \tag{3}$$

$$\phi(p) = \frac{1}{2}\sum_{j:E_j(p)>0} {E_j}^2(p)$$

goal: find p to minimize $\phi(p)$

4 Model with Saving and Borrowing

4.1 Equations and explanations

saving and borrowing begin at time t

$s_i(t) > 0$ means saving

$s_i(t) < 0$ means borrowing

$s_i(0) = 0$ means no saving or borrowing on the first day

λ interest rate; set in the interval of $(0, 1)$. We pick different number for interest rate to the changes in price, excess demand and rate of growth.

Here, we represent $s_i(t)$ as fraction of $r_i(t)$:

$$s_i(t) = cr_i(t) \tag{4}$$

when c is positive, then s_i is defined as saving, and when c is negative, s_i means borrowing

First of all, we set $c = 0.002$

Budget constraint:

$$s_i(t+1) + \sum_{j=1}^{m} r_i(t+1)A_{ij}p_j(t) = \lambda s_i(t) + \sum_{j=1}^{m} r_i(t)B_{ij}p_j(t) \tag{5}$$

we derive:

$$r_i(t+1) = r_i(t)\frac{\lambda c + \sum_{j=1}^{m} B_{ij}p_j(t)}{c + \sum_{j=1}^{m} A_{ij}p_j(t)} \tag{6}$$

Excess demand: (replace $r_i(t+1)$ with (6))

$$\begin{aligned} e_j(t) &= \sum_{i=1}^{n} r_i(t+1)A_{ij} - \sum_{i=1}^{n} r_i(t)B_{ij} \\ &= \sum_{i=1}^{n} r_i(t)\left(\frac{\lambda c + \sum_{k=1}^{m} B_{ik}p_k(t)}{c + \sum_{k=1}^{m} A_{ik}p_k(t)}A_{ij} - B_{ij}\right) \end{aligned}$$

$$E_j(p) = \sum_{i=1}^{n} r_i(t)\left(\frac{\lambda c + \sum_{k=1}^{m} B_{ik}p_k(t)}{c + \sum_{k=1}^{m} A_{ik}p_k(t)}A_{ij} - B_{ij}\right) \tag{7}$$

We do minimization by

$$\phi(p) = \sum_{j:E_j(p)>0} {E_j}^2(p)$$

we let $c = 0.002$ or $c = -0.002$ according to the change of demand and production

$$\sum_{j=1}^{m} r_i(t) B_{ij} p_j(t) > \sum_{j=1}^{m} r_i(t+1) A_{ij} p_j(t) \tag{8}$$

$$\sum_{j=1}^{m} r_i(t) B_{ij} p_j(t) \leq \sum_{j=1}^{m} r_i(t+1) A_{ij} p_j(t) \tag{9}$$

And our goal is finding p to minimize $\phi(p)$

5 Numerical Method

For the basic model, to avoid the condition that the global minima of ϕ will occur on the boundary of S, we make a change of variables

$p_j = q_j^2$ for $j = 1, ..., m$

where the domain of q is:

$$\bar{S} = \left\{ q \in R^m : \sum_{j=1}^{m} q_j^2 = 1 \right\} \tag{10}$$

Since by the equation 3, we can notice that

$$E_j(\alpha p) = E_j(p) \tag{11}$$

which means that for the excess demand, only the relative prices matter. Thus, we can strictly consider the price in the set

$$S = \left\{ p \in R^m : p \geq 0 \sum_{j=1}^{m} p_j^2 = 1 \right\} \tag{12}$$

Thus, for the basic model, we could give the limitation for the domain (10). Then, we should minimize this function over the unit sphere

$$\bar{\phi}(q) = \frac{1}{2} \sum_{j:E_j(\bar{q})>0} \bar{E_j}^2(q)$$

where

$$\frac{\partial \bar{\phi}}{\partial q_k}(q) = \sum_{j:E_j(\bar{q})>0} E_j\bar{(}q) \frac{\partial \bar{E}_j}{\partial q_k}(q)$$

Then, we calculate the derivative of $\bar{E}_j$

$$\frac{\partial \bar{E}_j}{\partial q_k}(q) = 2q_k \sum_{i=1}^{n} r_i A_{ij} \frac{B_{ik} \sum_{l=1}^{m} A_{il} q_l^2 - A_{ik} \sum_{l=1}^{m} B_{il} q_l^2}{(\sum_{l=1}^{m} A_{il} q_l^2)^2} \tag{13}$$

Then, we applied the gradient descent to get the value p, which can minimize ϕ. In our program, we set the tolerance of ϕ as 0.001 to make it close to 0 as much as possible. Thus, if the price cannot meet the requirement, we shall rerun the program since the price we get is not a global minimizer.

But we shall notice that, for the extended model, we cannot give the price a bound since equation (7) doesn't mean only the relative prices matter.

The derivative of the new excess demand function is

$$\frac{\partial \bar{E}_j}{\partial q_k}(q) = 2q_k \sum_{i=1}^{n} r_i A_{ij} \frac{\lambda c + B_{ik} \sum_{l=1}^{m} A_{il} q_l^2 - A_{ik} \sum_{l=1}^{m} B_{il} q_l^2}{(c + \sum_{l=1}^{m} A_{il} q_l^2)^2} \tag{14}$$

Thus, for the extended model, we won't have the condition that

$$\sum_{j=1}^{m} q_j^2 = 1$$

To solve this problem with no restrictions, we've tried two Matlab build-in method, "fmincon" and "fminsearch'. The first one is based on the gradient descent method, so the only distinction between the numerical model between the basic model and the extended model is there's no limitation for the unit sphere. And the second function is based on the Nelder-Mead simplex algorithm [3]. This algorithm will first sorted all the points such that the value of ϕ for the first point is the lowest and let the last one be the largest. Then calculate all the centroids except the worst point. After that, in each iteration, the algorithm will define how to transform the shape: reflection, expansion, contraction, or shrink contraction. When we reach the requirement, which is 0.01 tolerance in our program, the iteration should be stopped. However, after testing the model for many times, we find that the model will face a problem when $t = 3$ or $t = 4$. We'll illustrate it in the validation part and also take the result we get from the "fminsearch" as one way to validate the model.

6 Validation

6.1 Excess Demand

By the graph, we can check that the excess demand of the basic model is close to the 0 compared with the random number generalized for matrices A and B, in which each number could be nearly dozens.

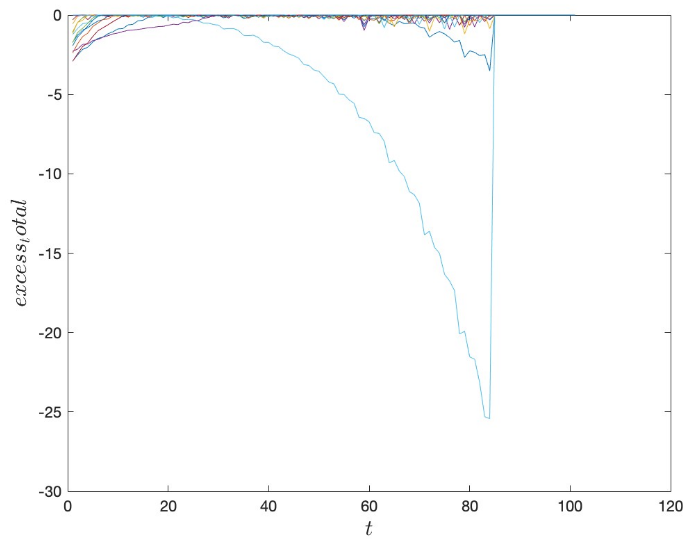


Fig. 1. Excess Demand for the basic model

There's only one or two line drop largely from 0, and we consider it as the error since it is always the line starting far from 0. With time passing, the error will get larger so which will lead to this condition.

We'll see that this condition also holds for the saving-and-borrowing model. By the definition of the equilibrium price vector in Prof. Peskin's lecture notes [2], we shall always get

$$E(p^*) \leq 0$$

Here we check that all the excess demands fit this requirement.

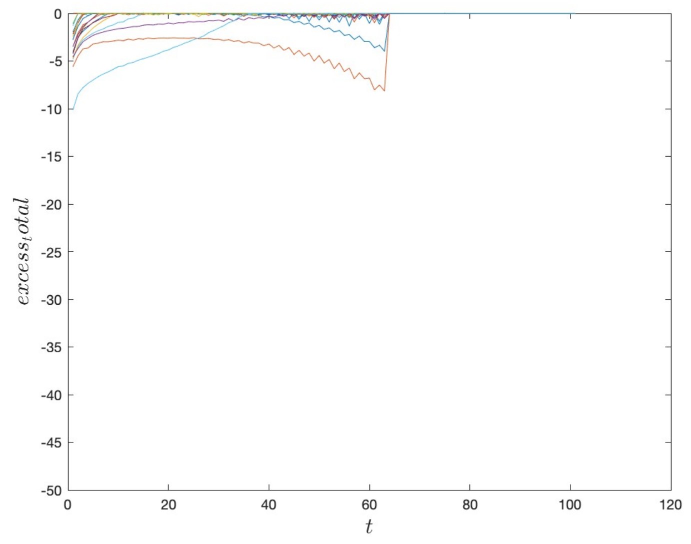


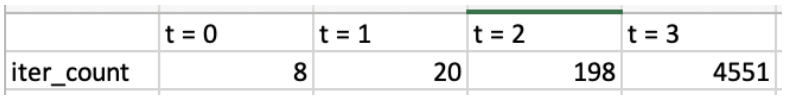
Fig. 2. Excess Demand for the B & S model

We can find that in the first half of the progress, the excess demand is always smaller or equal to 0 and has almost the same trend as the basic model.

However, when t is close to 64, it's hard to find a p to make the excess demand fit our requirement. It may be caused by the unconstrained change of p, and we'll illustrate it in the next part. We also check that no matter in which condition, r is always similar, which also validates this model.

6.2 Two different built-in methods

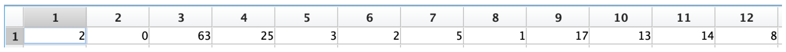
As we said in the previous part, the program cannot find an appropriate value by "fminsearch" method. We count the iteration times, which means how many times the program will finish searching the equilibrium price in one time period. And here's the result for running "fminsearch" for about 20 minutes.



	t = 0	t = 1	t = 2	t = 3
iter_count	8	20	198	4551

Fig. 3. iteration count for fminsearch

We can see that when $t = 3$, the program will run for nearly 5000 times without getting an ideal result. When we apply the unconditional situation to the fmincon method, we'll get pretty good numbers of iterations. (the first line stands for the time period, and the second line means the iteration times)



	1	2	3	4	5	6	7	8	9	10	11	12
1	2	0	63	25	3	2	5	1	17	13	14	8

Fig. 4. iteration count for fmincon

But we can still check the r_{total} for the first three or four time periods and get the following pictures. This is generated by fminsearch

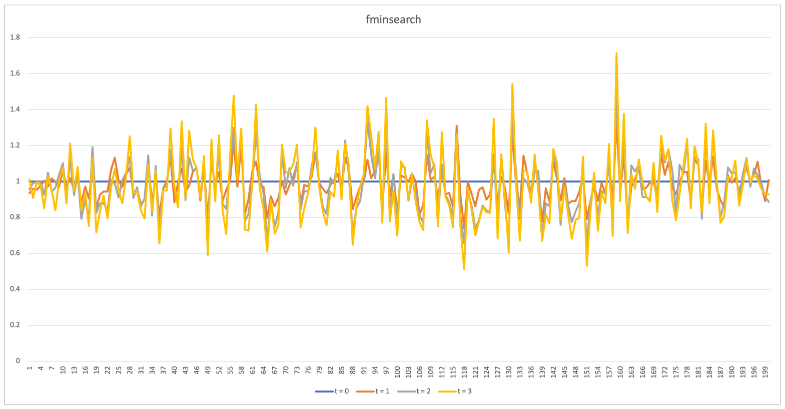


Fig. 5. iteration count for fminsearch

This is generated by fmincon

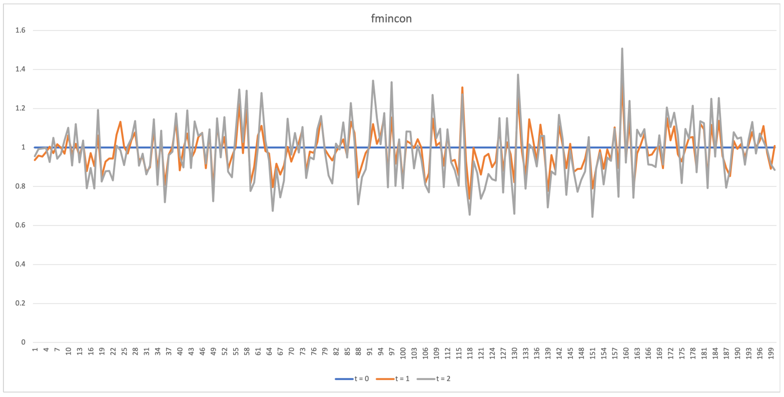


Fig. 6. iteration count for fmincon

We can find that even solving the problem by the different methods, we'll get almost the same r_{total}. This means that "fminsearch" may face some problems without setting the condition in our program so it cannot get the result effectively. But by using "fmincon" without restriction, we can get the same answer in a shorter time, which means it should be a better algorithm for handling our model.

6.3 total price

For the basic model, we make the sum of the prices equal to 1. And here we check in the new model, the sum of p changes instead of being a fixed constraint.

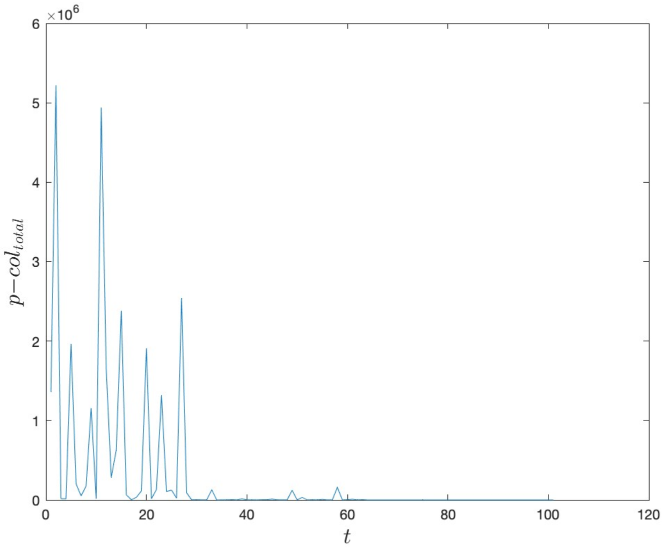


Fig. 7. total price of b&s model

7 Discussion

7.1 the change of inputs of matrices A and B

We changed the inputs of matrices A and B. From purely randomly generated numbers to uniformly distributed numbers from 0 to 1, and normally distributed numbers. The density of price with time will almost not change. It means that the change of numbers won't affect the number of free goods.

However, the log of r will change as we increase the number in A and B, and this will be explained in the following part.

7.2 the rate of log($\sum r$)

In the following pictures, the first one is when the matrices A and B have a relatively high probability of getting larger numbers for each element (abs(0.1 * randn(n,m) + 100)+1e-5). The second picture shows the slope of the log of sum r with data (abs(0.1 * randn(n,m) + 10)+1e-5) generating the matrices. And the third graph is under the condition that abs(0.1 * randn(n,m) + 1)+1e-5, which means each element won't likely to get a large value.

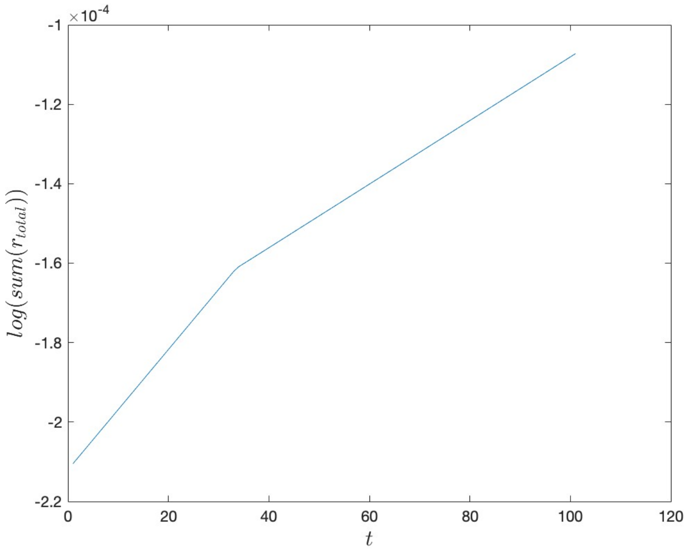


Fig. 8. basic model: rate of log when A&B large

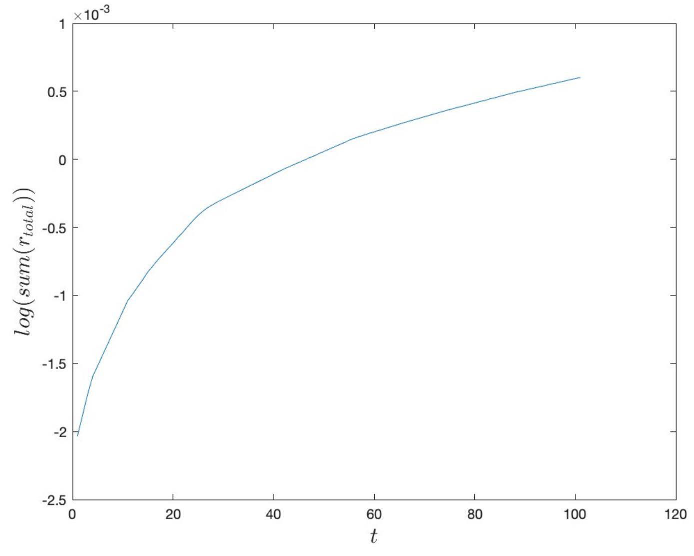


Fig. 9. basic model: rate of log when A&B small

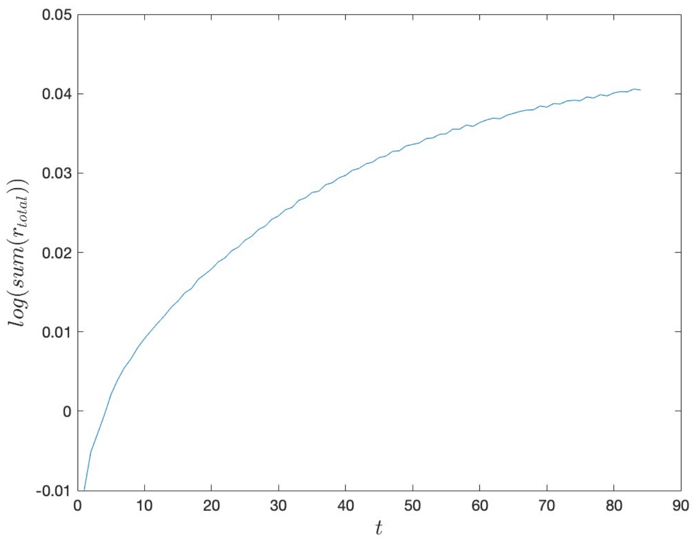


Fig. 10. basic model: rate of log when A&B medium

We test the relatively larger and smaller values for several times, and we find that the slope of log r will approach a larger number (0.5 rather than 0.4) if A and B are larger. But if A and B are too large, there'll be a problem with the program – the process will stop very quickly (only about 20-30 rounds of time t).

We also change the number of λ with $\lambda = 0.01$, $\lambda = 0.02$, $\lambda = 0.3$, and $\lambda = 0.5$. But the slope is almost the same. So the change of λ won't affect the slope of log $\sum$r significantly.

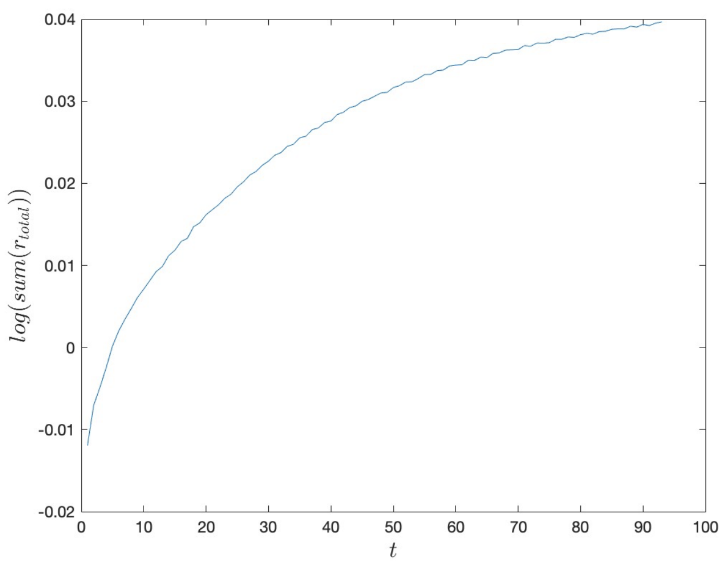


Fig. 11. basic model: rate of log for 0.02

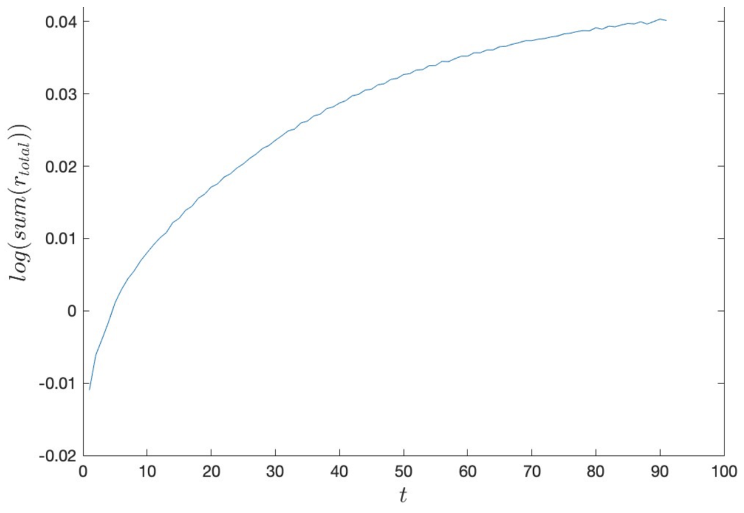


Fig. 12. basic model: rate of log for 0.05

7.3 Prices

The price of the basic model and the borrowing-and-saving model is also different. We can see that the p_{total} is relatively small since we normalize it, which can be considered as giving the price a restriction.

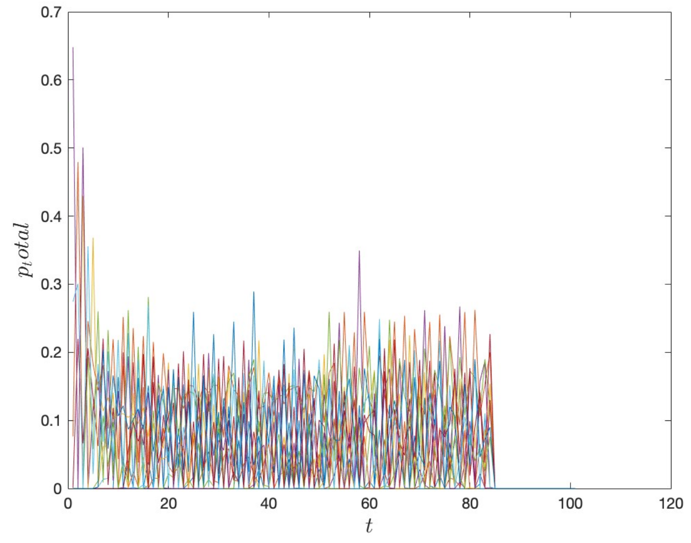


Fig. 13. price of the basic model

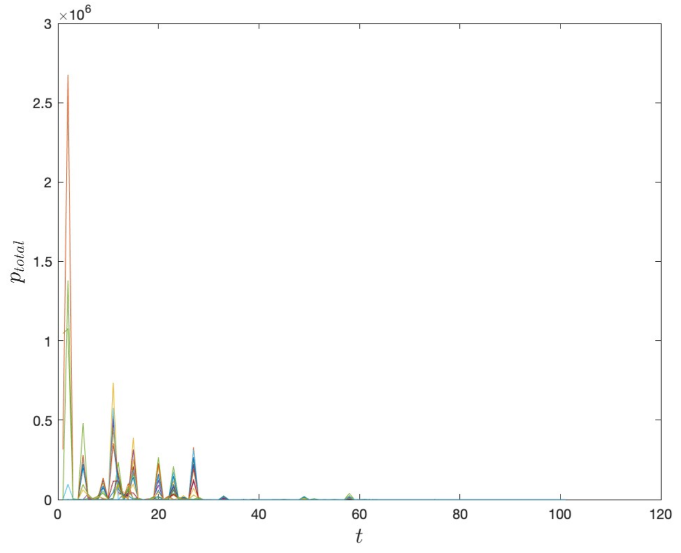


Fig. 14. price of the B&S model

However, in the new model, there are no limits in rules for borrowing or saving. Also, there's no limitation on the price. As a result, the price will have an extremely great change in a short time, and that may also be one reason for the collapsed market as t goes larger.

7.4 sum of prices

The sum of the prices is shown in the following graphs, we can see that for the basic model, it's always equal to 1 and in the extended model, it'll change dramatically. 6.3

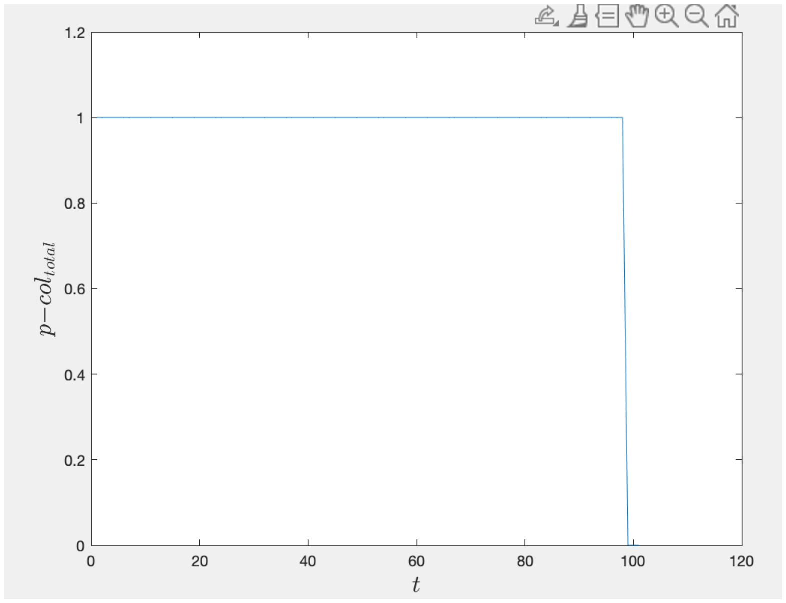
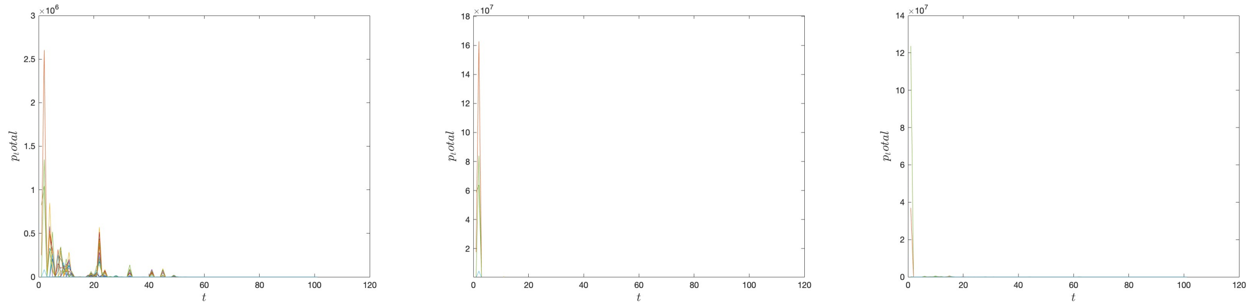


Fig. 15. total price for the basic model

We also find that the price will change periodically when both saving and borrowing are allowed in the model. About every two or three rounds of t, the total price will go down and then rise immediately. It may stand for the cyclic period of the market. Also, we find that if we don't give any restriction of the borrowing or saving amount, and allow the borrowing and saving to happen at the same time. The price will be extremely high than that in the basic model. That may lead to the collapse of the market so that when t grows larger, it'll be extremely hard to find a proper p to keep the market operating as usual.

7.5 interest rate

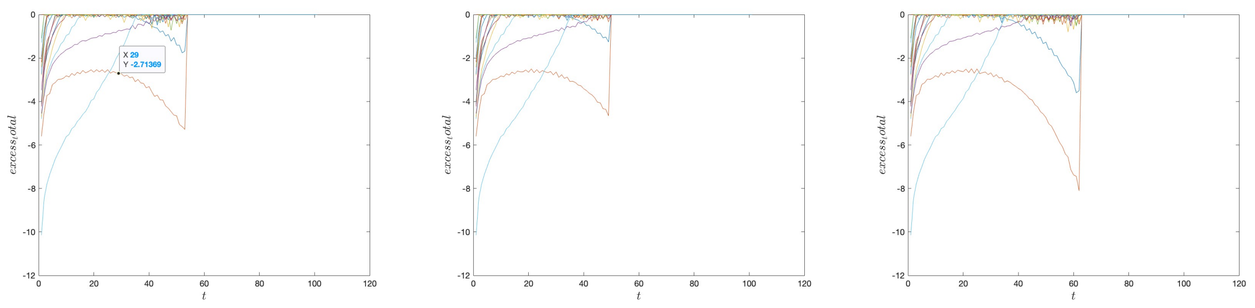
In the new model that allows saving and borrowing, when we change the interest rate (choose the value:0.02, 0.2, 0.5), and fix the inputs of matrices A and B at the level of (abs(0.1 * randn(n,m) + 1)+1e-5), we see a significant difference in the price.



(a) (b) (c)

Fig. 16. total price when (a)$\lambda = 0.02$ (b) $\lambda = 0.2$ (c) $\lambda = 0.5$

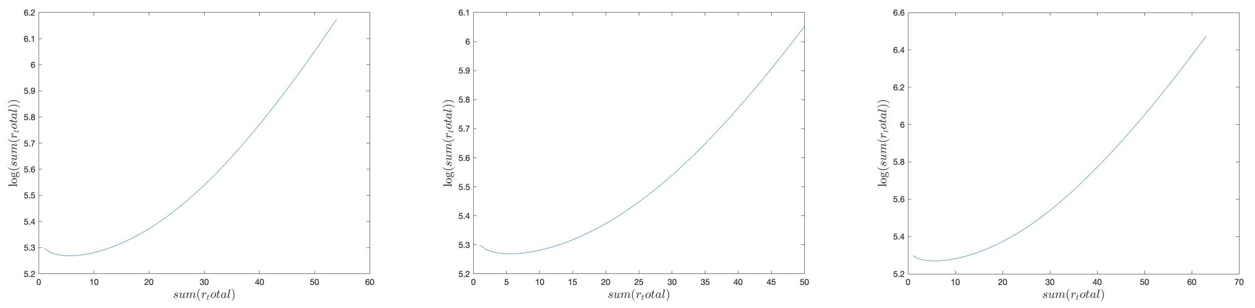
For the total price, we see bigger summit of fluctuations (Figure 16) when λ increases, and those fluctuations tend to be more obvious at the beginning of market day (and can be extremely large) and more free goods appear when time passes.



(a) (b) (c)

Fig. 17. excess demand when (a)$\lambda = 0.02$ (b) $\lambda = 0.2$ (c) $\lambda = 0.5$

As for the excess demand, the fluctuation tends to become bigger with market day growing but the interest rate generally won't have obvious effect on the pattern of excess demand (figure 17).



(a) (b) (c)

Fig. 18. log sum of r when (a)$\lambda = 0.02$ (b) $\lambda = 0.2$ (c) $\lambda = 0.5$

Additionally, the log sum of r always have similar increase pattern and value no matter how λ changes (figure 18). As a result, in some ways, the growth rate and excess demand won't be largely affected by the interest rate but the total price will be influenced, as larger interest rate leads to extreme price.

8 Summary and Conclusion

We first test the basic model and compare it with the new model including the condition of saving and borrowing. We find that the market tends to face a problem as t goes larger if we don't give any restriction on the price p. Finally, we compare several important variables' changing given different values for other constants. We analyze the rate of log($\sum r$), prices' changing, the change of the sum of prices in one period, and what will happen when we change the interest rate (but also keep the interest rate similar to the real world).

References

[1] Von Neumann J: A Model of General Economic Equilibrium. The Review of Economic Studies 13(1): 1-9, 1945

[2] Charles S.Peskin: Notes on Economic Growth And Price Equilibrium, 2019

[3] srjoglekar246: Nelder-Mead Optimization, Jan 16, 2016.
https://codesachin.wordpress.com/2016/01/16/nelder-mead-optimization/

9 Appendix

Here’s the code for the price equilibrium.

```
global n m A B r_total r_count;
n = 100;        %process
m = 15;         %good
lambda = 0.3;   %interest rate

%here’s the initial variables randomly generated
%%A and B are positive
A = abs(0.1 * randn(n,m) + 10)+1e-5;                  %consumption matrix
B = abs(0.1 * randn(n,m) + 10)+1e-5;                  %production matrix
r0 = ones(n,1);                                       %intensity r
q0 = 10 + randn(m,1);
q0 = q0/sqrt(sum(q0.^2));

%set up for simulation
t_max = 100;                          %max timestep
q_total = zeros(m,t_max + 1);         %record q
p_total = zeros(m,t_max + 1);         %record p
excess_total = zeros(m,t_max + 1);    %record excess demand
r_total = zeros(n,t_max + 2);         %record intensity
r_total(:,1) = r0;
r_count = 1;
iter_count = zeros(1, t_max + 1);     % count the iteration to get the proper

fun = @Minimization;            %objective function to minimize

%nonlicon = @nlcon;              %nonlinear constraint
nonlicon = [];
%options for the nonlicon, uncommanded for the basic mode
% options = optimoptions('fmincon',"EnableFeasibilityMode",
true,'SpecifyObjectiveGradient',true, ...
%     'MaxIterations', 3e3, 'MaxFunctionEvaluations',
3e4, "SubproblemAlgorithm","cg",'Display','iter');

for t = 0:t_max
    if ((t+1) == 1)
        q_equi = fmincon(fun, q0, [], [], [], [], [], [], nonlicon, options)
        %minimize the objective function with equilibiurm price in the prev
    else
        q_equi = fmincon(fun, q_total(:,t), [], [], [], [], [], [], nonlico
        %minimize the objective function
    end
```

```
validResult = false;                          %check
%minexcessdemand = 100;%initialize current min of excess
for itr = 1 : 10
    while fun(q_equi) > 0.01 %tolerance, need to take q again
        iter_count(1,t+1)=iter_count(1,t+1)+1;
        %q0 = 10 + 3*randn(m,1);
        q0 = 10 + randn(m,1);                    %choose a number for q0
        q0 = q0/sqrt (sum(q0.^2));
        q_equi = fmincon(fun, q0, [], [], [], [],
        [], [], nonlicon, options);
        %minimize the objective function
        %q_equi = fminsearch(fun,q0);            %with fminsearch method
        disp(ExcessDemand(q_equi.^2));           %dispaly E
        %disp(fun(q_equi));
        %disp('r is');
        %disp(r_total(t+1));
    end
    p_equi = q_equi;                             %equilibrium price
    %record
    q_total(:,t+1) = q_equi;
    p_total(:,t+1) = p_equi;
    excess = ExcessDemand(q_equi);
    if(all(excess < .01))
    %check: excess every good < 0
          validResult = true;
          break;
    end
end


if(~validResult)
%report the error if the model cannot work well
      disp("invalid result, there's an error");
      return;
end

%r(t+1)
r_total(:,(t+1)+1) = (r_total(:,t+1) .*
((B*(q_equi.^2)) ./ (A*(q_equi.^2)))) ;
% eliminate small variables
for count = 1:n         %r
    if (r_total(count,(t+1)+1)) < 1e-2 && (r_total(count,t+1)) > 0 ||
    r_total(count,(t+1)+1) < 0
        r_total(count,(t+1)+1) = 0;
        end
    end

    for count = 1:m         %E
        if (excess(count) < 1e-2 && excess(count) > 0)
        || (excess(count) > -1e-2 && excess(count) < 0)
            excess_total(count,t+1) = 0;
        else
            excess_total(count,t+1) = excess(count);
        end
    end
    r_count = r_count + 1;      %alternative referencing t

    %the choose of con (saving/borrowing) depends on the previous market
    %condition

    if (r_total(:,t+1) .* (B*(q_equi.^2))
    > r_total(:,(t+1)+1) .* A*(q_equi.^2))
        %con = repmat(0.002 .* ((B*(q_equi.^2))./(A*(q_equi.^2))),200,1);
        %%another way of chooing c
        con = 0.002;
    else
        %con = repmat(-0.002 .* ((B*(q_equi.^2))./(A*(q_equi.^2))),200,1);
        con = -0.002;
    end
end

toc

%Functions used above:
%excess demand
function [E, grad_E] = ExcessDemand(q)
    global m A B r_total r_count lambda con;
    %excess demand function
    E = ( transpose(r_total(:,r_count)) *
    ((repmat((lambda * con + B*(q.^2))
    ./ (con + A*(q.^2)), 1, m) .* A - B) )).';
    %gradient of excess demand function
    grad_E = 2.*q.*((transpose(r_total(:,r_count))
    * ((A.*( ((B.*repmat((con + A*(q.^2)), 1, m))...
        -(A.*repmat((lambda * con + B*(q.^2)), 1, m)))
        ./ (repmat((con + A*(q.^2)).^2, 1, m)) )))))';
end

%minimize
```

```
function [phi, grad] = Minimization(q)
    global m A B r_total r_count lambda con;
    E = ( transpose(r_total(:,r_count))
    * ( repmat((B*(q.^2)) ./ (A*(q.^2)), 1, m) .* A - B ) ).';
    grad_E = 2.*q.*((transpose(r_total(:,r_count))
    * ((A.*( ((B.*repmat((con + A*(q.^2)), 1, m)) ...
        -(A.*repmat((lambda * con + B*(q.^2)), 1, m)))
        ./ (repmat((con + A*(q.^2)).^2, 1, m)) )))))';

    phi = 0;
    grad = zeros(m,1);
    for e_j = 1:m
    %take the sum of E>0
        if E(e_j) > 0
            phi = phi + E(e_j);
        end
    end

    %gradient descent
    if nargout > 1
        for k = 1:m
            if (E(k) > 0)
                grad(k) = grad(k) + (E(k) * (grad_E(k))') ;
            end
        end
    end
end

% for the nonlinear constrain
% uncommanded if for the basic model
% function [c,ceq] = nlcon(q)
%     c = [];
%     ceq = sum(q.^2)-1;
% end
```