UNIVERSITÀ DEGLI STUDI DI TORINO

Fisica Dei Sistemi Complessi

Econophysics

Policy experiments in a macroeconomic model with a turbulent environment

Author:
Carlo Fisicaro

June 13, 2016

Contents

1 Introduction

The economy can be seen as a *complex system*, i.e. a system composed by a huge number of components, each of which can interact with another one in a *non linear way*. Higher the quantity and the variety of relations between the elements of a system and more is its own complexity. Another feature of a complex system is that it can produce an *Emergent Behavior*, i.e. a complex behavior which is not merely the sum over the components of the system. An example is the performance of financial markets. Although one can predict and understand the behavior of individual investors in *microeconomy*, it's impossible to forecast, given the knowledge of individual traders, the performance of the *macroeconomy*.
There is also another approach to the problem. It is called Dynamical Stochastic General Equilibrium (DSGE) which takes inspiration from the theory of general equilibrium. Unlike traditional macroeconometric forecasting models, DSGE should not be, at least in theory, vulnerable to the Lucas Critique. Indeed, DSGE models start from microeconomic principles of constrained decision-making, instead of just taking as given observed correlations (take a look to [4] for a deeper discussion).
Fagiolo and Roventini [1] argue that DSGE models have theoretical, empirical and political economy issues which prevent them to be useful in policy analysis. What they suggest is to turn to Agent Based Computational Economics (ACE) or Agent Based Models, thanks to which empirical validation could be done by looking at the number of "stylized facts" the model is able to replicate, possibly replacing assumptions in a modular way. This would not be possible in DSGE models. An Agent Based Model can also be used to study different strategies available for banks, gaining some insight to distinguish between strategies that may be appropriate and that warrant further empirical investigation, and those that do not.
At the base of my simulation there is the one developed by Pangallo [2]. I even want to emphasize that my work draws many ideas from [3].
I simulate an economic system which contains a network of banks and customers. The agents aren't positioned in a physical space, they in fact "live" within a strategy space. In order to understand the dynamics of how a bank moves inside this space, the differences between "deliberate" (planned) strategies and "emergent" (learning) strategies are discussed.
The strategy space has the property of being even a turbulent environment, i.e. the customers can have a *Brownian motion* along this space. As men-

tioned above, if the business world is viewed as being complex, it is inappropriate to consider models developed under paradigms of equilibrium, stability and linearity to produce an analysis of a turbulent environment. Just as non-linear mathematics is required when studying turbulence within physics, we too adopt models and tools that capture the non linearity of this environment. The main difference from [3] is that I study both macro-variables such as GDP as well as banking strategies (e.g. "Stay Still" Strategy) also simulating monetary policy actions taken by a Central bank and, to simplify my simulation, I don't consider the "Customer Center" strategy.
The rest of this work is structured as follows : Section 2 describes the theoretical model without a specific reference to any programming language. Section 3 describes its implementation in Netlogo[1]: I include the code of the most important parts. In Section 4 I explore the parameter space and perform some policy experiments in a turbulent environment using two different banking strategies, by making use of Mathematica[2] and its package Mathematica Link[3].

[1]https://ccl.northwestern.edu/netlogo/
[2]https://www.wolfram.com/mathematica/
[3]https://ccl.northwestern.edu/netlogo/docs/mathematica.html

2 The model

There are five kind of agents. R&D firms produce capital-goods (machinery) and perform R&D in order to improve the performance of the goods they sell and to cut their production costs. Product firms make final goods using the capital-goods they bought from R&D firms and sell them to their customers. Workers work in the firms; they consume only a fraction of their wage and deposit the other fraction in a bank with a certain probability, obtaining a percentage of the deposited money, fixed by the respective bank. Banks pay a fraction of their deposits to the Central bank obtaining from it a percentage of the deposited money, and grant loans to R&D firms, product firms and workers fixing the interest rate that each customer has to pay towards the respective bank. Central bank grant loans to banks, deciding the interest rate that each debtor bank has to pay, and fixes several parameters that correspond to monetary policy choices I will discuss below. I also impose the Central bank to occupy a stationary position in the strategy space. R&D firms, product firms and workers/consumers also have a *satisfaction level* that I have set inversely proportional to the distance between the customer and the bank in the strategy space. If *satisfaction level* $<$ *satisfaction cut-off*[4], customers search for a new bank. Finally, if *customers rationality level* $= 10$, all potential customers will choose the nearest bank; otherwise if *customers rationality level* $= 0$, they will choose a random one. In order to introduce turbulence into the model, I assume the customers either stay still[5] or take a random walk along the space, i.e. within a turbulent environment.

2.1 R&D firms

As described in [2]

> [...] a number $I = |\mathcal{I}|$ of R&D firms is instantiated and stays in the market until the end of the simulation. Every R&D firm has some attributes: lists containing its workers and its customers (the product firms that bought at least one piece of machinery on previous time step), an integer $M_i(n)$ representing its liquid assets, a real number $f_i(n)$ concerning its market share (the number of orders that firm received with respect to the total of transactions that took place in the

[4] model's parameter which indicates the minimum satisfaction level of the customers
[5] i.e. in a non-turbulent environment

> R&D firms - product firms market), a tuple (A_i^n, B_i^n), i.e. the two parameters representing the level of technology of each firm at time step n. The cost of producing one machine for firm $i \in \mathcal{I}$ is $c_i(n) = 1/B_i(n)$: $c_i(n)$ is the wage payed to firm's workers. By adopting this transaction mechanism no money disappears, all the costs of production are transferred to workers. The highest $B_i(n)$, the lowest is the cost the R&D firm faces. Every R&D firm makes k machines any time step, where $k = \max\{S_i(n-1)+5, S_i(n-1)(1+0.05)\}$ is adaptive with respect to past sales. The price chosen by any R&D firm to sell its machines is $p_i(n) = (1+\mu)c_i(n)$: μ is the markup, constant across R&D firms: they are also homogeneous with respect to the fraction ν of their liquid assets they spend in R&D, adding up to $RD_i(n) = \nu M_i(n))$; all the rest of $M_i(n)$ is used to make the k machines[6]. The innovation process is as follows: the R&D firm gets access to a discovery through a draw from a Bernoulli distribution, whose parameter is $\theta_i(n) = 1 - e^{-\zeta RD_i(n)}$. Then $A_i^{new}(n) = A_i(n)(1 + x_i^A(n))$, $B_i^{new}(n) = B_i(n)(1 + x_i^B(n))$, where $x_i^A(n), x_i^B(n) \sim \Gamma(\alpha, \beta)$, i.e. are drawn from a Gamma distribution with parameters α (shape) and β (rate) and support $[0, \infty)$. The only thing left to R&D firms is to update their list of customers: they start from the list of the product firms that made at least one purchase on the previous time step $(HC_i(n-1))$, and they add $\gamma HC_i(n-1)+1$ product firms chosen randomly (under the constraint they were not previous customers). γ is a proxy for the degree of perfect information in the market, and this assumption captures the insight that the more powerful firms have also the more advertising power.

In addition to the attributes mentioned above, in my model every R&D firm has: a real number $\tilde{d}_i(n)$ concerning its debt towards a bank, a real number $\tilde{l}_i(n)$ representing the amount of the loan for which the client applies and a real number $\tilde{s}_i(n)$ which indicates its satisfaction level[7].

2.2 Product firms

As written by Pangallo [2]:

[6]if $p_i(n) \cdot k > M_i(n)$ only a fraction of the machines is produced, if $p_i(n) \cdot k < M_i(n)$ the firm saves money for next time step

[7]I put $\tilde{s}_i(n) = 1/\tilde{d}_i(n)$, where $d_i(n)$ represents the euclidean distance between a bank and its i-th R&D customer

[...] a number $J = |\mathcal{J}|$ of product firms is instantiated and stays in the market until the end of the simulation. Every product firm has some attributes as R&D firms: a list of workers, market share $f_j(n)$ (which is now the number of goods that product firm sold with respect to the total of transactions that took place in the product firms - consumers market) and liquid assets $M_j(n)$. Peculiar to product firms are a list representing capital stock (a list of h machines whose values are A_i^n's since B_i^n's only have an effect on the price of the machine and thus can be forgotten), and an integer number $N_j(n)$ (inventories). The product firm can use a machine to produce only one good at any time step. If the desired level of production (to be explained later on) $Q_j(n)$ is such that not all machines are to be used, the product firm starts from the ones with a higher $A_i(n)$. Actually the price it has to pay to produce a consumption good is $c_j(A_i^n, n) = 1/A_i^n$, so A_i^n can be regarded as a measure of the quality of the machine. The desired level of production is again decided on the basis of an adaptive rule: $Q_j^D(n) = 10 + S_j(n-1) + N_j^D(n) - N_j(n-1)$. The desired level of inventories is $N_j^D(n) = 0.1S_j(n-1)$: this way product firms adjust their production to cope with the demand. The 10 in the beginning of the formula is to let them recover from a demand crisis. The problem of product firms is the way they should allocate their liquid assets between production of goods and investments in machinery. Lacking an explanation of this mechanism in the original model, I propose the following strategy: the largest part of liquid assets are allocated to produce $Q_j^D(n)$ a minor part[8] is invested to buy up to $\max\{1, 0.2 \cdot h\}$ new machines[9]. This is also needed because machines have to be replaced after 20 time steps. Finally, if some money is left, it is used to complete the production $Q_j^D(n)$, up to $Q_j(n) \leq Q_j^D(n)$). If some money is still left, it is kept for next time step. Product firms decide which machines to buy according to the following protocol: they check which R&D firms have them in their customer list, they rank R&D firms according to the lowest $p_i(n) + c_j(n)(A_i^n, n)$ (sum of cost and quality), they buy as many machines as they planned by choosing randomly one of the R&D firms in the list. A

[8]the exact amount depends on markup: if investment is larger then profits, firms end up running out of their liquidity, and the result is that the whole economy collapses

[9]in Dosi et al. (2010) in footnote 12 the authors mention a fixed maximum threshold for the capital growth rate

> draw from a $\Gamma(1.2, 0.66)$ distribution, combined with a "floor" function, defines the position in the R&D firms list: it is more likely to choose the most competitive, but sometimes a less competitive R&D firm may be chosen. This prevents, or at least slows down, the formation of a monopolistic market. So far I focused on production plans of product firms. Let's now have a look at their selling strategy. Prices are $p_j(n) = (1+\mu)c_j(n)$. Since product firms' goods are heterogeneous in price, they start selling them from the cheapest ones. Firms profits are $\pi_j(n) = \sum_{soldItems} p_j \cdot D_j(n) - \sum_{producedItems} c_j(n) \cdot Q_j(n)$: profits may be negative, but additional production is kept in inventories, and less production will occur next time step. The level of product firm's liquid assets is finally updated: $M_j(n+1) = M_j(n) + \pi_j(n)$. Here $M_j(n)$ is what is left from production and investment.

In addition to the attributes mentioned above, as for R&D firms, in my model every product firm has: a real number $\tilde{d}_j(n)$ concerning its debt towards a bank, a real number $\tilde{l}_j(n)$ representing the amount of the loan for which it applies and a real number $\tilde{s}_j(n)$ which indicates the satisfaction level.

2.3 Workers/Consumers

As described in [2]:

> [...] a number $N = |\mathcal{N}|$ of workers/consumers is instantiated and stays in the market until the end of the simulation. Every worker has one attribute (i.e. the amount of liquid assets) and he can be employed or unemployed. Labor supply is inelastic: $L^s = N$. Labor demand is adaptive. Firms not only need liquid assets and, in the case of product firms, machinery, but they need also workers. I assume that $1/B_i^n$ workers are needed to produce one unit of output by the R&D firms, $1/A_i^n$ by the product firms. This reflects the technological improvement. At the end of each time step firms hire as many workers as it would be needed to produce the desired amount of output. If next time step the firms need even more workers, they hire them; on the other hand, if they need to fire them, they can fire them with a successful Bernoulli trial with parameter ω. This parameter captures the exibility of the labor market, and it can be tuned by the government. Firms decide randomly which workers they hire/fire. However there is a problem with this approach. It turned out while implementing

the simulation that a realistic labor market would require a balance between technological improvement (i.e. productivity) and level of production: these two quantities should scale together. Actually, if production increases faster than productivity, all workers are to be hired; if it scales slower, most workers are to be fired. Since I wanted to study the effects of several policies on GDP, I devised a simplified labor market imposing a "natural rate of unemployment" in the code, by normalizing labor need of firms to about 90% of labor force. This arbitrary choice allows both to show which firms hire the most workers and to keep some unemployed consumers who can benefit from the policies that target them. All the previously described features of the labor market stand also with this simplification (firms still hire and fire workers). As I already mentioned, all production costs $c_i(n)$, $c_j(A_i^n, n)$ are used in wages. Each firm computes its total costs and divide them equally among its employed workers. The average wages are fixed (since total liquid assets are fixed) but growth is apparent from the sharp increase in production. Consumers spend a fraction $(1 - s)$ of their income in consuming goods, and keep a fraction s to increase their liquid assets. When they are unemployed, they consume half of their liquidity to buy goods. To capture imperfect information also in the product firms - consumers market, consumers choose the best price within $J/10$ firms chosen randomly (as if they randomly checked their prices).

In addition to the attributes mentioned above, as for R&D firms and product firms, in my model every Worker or Consumer has: a real number $\tilde{d}_{n'}(n)$[10] concerning its debt towards a bank, a real number $\tilde{l}_{n'}(n)$ representing the amount of the loan for which it applies and a real number $\tilde{s}_{n'}(n)$ which indicates the satisfaction level.

2.4 Banks

A number $P = |\mathcal{P}|$ of banks is instantiated and stays in the market until the end of the simulation. Every bank has some attributes: the amount of liquid assets $M_p(n)$, a real number $\tilde{d}_p(n)$ concerning its debt towards another bank, a real number $\tilde{l}_p(n)$ representing the amount of the loan for which it applies, a real number $\tilde{s}_p(n)$ which indicates the satisfaction level, a real number

[10] I use n' to distinguish the n-th time step from the Workers/Consumers label

$\tilde{x}_p(n)$ standing for the amount of deposits at the p-th bank, a real number $\tilde{d}_p(n)|_{CentralBank}$ concerning its debt towards the Central bank, a real number $\tilde{l}_p(n)|_{CentralBank}$ representing the amount of the loan for which it applies and a boolean variable $\iota(n)$. If liquid assets is greater than liquid assets of other banks, $\iota(n) = True$; otherwise $\iota(n) = False$. The leader bank is the one with $\iota(n) = True$. In the simulation I define several types of interest rates, two of which are fixed by banks: η_1 represents the loan's interest rate a debtor must pay to the creditor bank whereas η_3 is the deposit's interest rate a bank must pay to its customers[11], whose financial returns are $\pi_{n'}(n) = \eta_3 \cdot \tilde{d}_{n'}(n)$; while the others are fixed by the Central bank: re symbolizes the banks' reserve ratio i.e. the percentage of deposits banks are required to keep as cash, η_2 represents the loan's interest rate a debtor bank must pay to the Central bank whereas η_4 is the deposit's interest rate the Central bank must pay to its customers (only commercial banks), whose financial returns are $\pi_p(n) = \eta_4 \cdot \tilde{d}_p(n)$. In reality the banks' reserve ratio should be equal to $re_0 + re_L$, where re_0 is the compulsory reserve ratio (fixed by the Central bank) while re_L acts for the reserve ratio (settled by each singular bank). To simplify the problem I assume that $re_L = 0$ then $re = re_0$, and η_1 has the same value for each bank, ditto for η_2, η_3 and η_4.

Unlike the other types of agents, in my simulation banks have two strategies available: the *"stay still"* strategy, reflecting a *deliberate strategy* on the part of the banks, resulting in a stationary position within the strategy space. The other one is the "*follow the leader*" strategy, representing a more *emergent* strategy. Adopting the latter strategy, non-leader banks try to emulate the leader's one, resulting in the non-leader banks' move near the leader within the strategy space.

Every bank grants loans if and only if $M_p(n) - re_0 \cdot \tilde{x}_p(n) > \sum_{i=1}^{\xi(n)} \tilde{l}_i(n)$, where $\xi(n)$ represents the number of agents which apply for a loan at time step n and $\tilde{l}_i(n)$ stands for the amount of loan for which the i-th agent applies; while the condition above is not satisfied the p-th bank removes a *possible debtor*[12] from its register.

[11]here I mean all agents that deposit money at a bank

[12]an agent which is applying for a loan

2.5 Central bank

Only a Central bank is instantiated and stays in the market until the end of the simulation. It has just one attribute, i.e. a real number $x_{CB}(n)$ standing for the amount of deposits at the Central bank, assuming then that it has always enough money to grant loans. Central bank has several monetary policy's instruments to stimulate growth, which in my simulation come out by varying: η_2, η_4, re_0 and the *quantitative easing* level, i.e. the probability that the Central bank will grant loans to a bank. Central bank is a no-profit entity so it isn't interested in attracting more and more customers, that's why it has a stationary position in the strategy space.

2.6 Public sector

The government levies a tax ϕ on $\pi_j(n)$ and $\pi_p(n)$ (R&D funds, wages and deposits are not taxed). It can spend its funds in unemployment insurance (spending a fraction δ of its funds) or directly buying final goods (spending a fraction $1 - \delta$ of its funds), according to the desired policy.

3 Simulation

NetLogo is an intuitive programming language well suited for modeling complex systems developing over time. Modelers can give instructions to hundreds or thousands of "agents" all operating independently. This makes it possible to explore the connection between the micro-level behavior of individuals and the macro-level patterns that emerge from their interaction[13]. On the other hand, dealing with lists is not trivial; this is why Pangallo [2] first, and then I, decided to provide each agent with links to other agents. The main drawback in that path is speed: indeed, when the number of agents substantially increases, the simulation speed dramatically goes down.
There are five types of agent breeds: `rdFirms`, `productFirms`, `laborForce`, `banks` and `centralBanks`. There are also four link breeds: `supplyRelations` correspond to links between `rdFirms` and `productFirms`, `workRelations` are between firms and `laborForce`, `liquidAssetRelations` place among firms, `laborForce` and `banks` and finally `centralBankRelation` correspond to links between `banks` and `centralBanks`. As mentioned above, patches

[13]https://ccl.northwestern.edu/netlogo/faq.html

rather than a physical space, represent a strategy space which can be extended to an n-dimensional one, called "*strategy hypercube*". I describe below some procedures explaining the less intuitive parts (take a look to Pangallo (2014) for a complementary explanation).

3.1 setup procedure

```
to setup

clear-all
random-seed Seed


set I HowManyRdFirms
set J HowManyProductFirms
set N HowManyWorkersConsumers
set P HowManyBanks
set nu InvestmentInRD
set zeta ChanceNewDiscovery
set alpha 1.2
set beta 1 / TechnologicalDevelopment
set gamma 0.5
set mu Markup
set s SavingRate
set omega FlexibilityLabour
set governmentAssets 0
set fi TaxesLevel
set delta SpendingInUnemploymentAid
set eta1 interestRateRestitution
set eta2 CBRateRestitution
set eta3 interestRateDeposit
set eta4 CBRateDeposit
set ni savings_in_aBank_probability
set re_0 bankReserveRatio
set numberOfInstallments 10
set n_loans 0
set totalProduction 0
```

[...]

```
initial-hire

reset-ticks
end
```

The value of most variables is set in the interface, the seed is fixed in order to allow the repeatability of policy experiments and parameters tuning. The names of the variables are the same used in Section 2, the `numberOfInstallments` is arbitrary. `initial-hire` is a procedure that links every firm with a worker. `reset-ticks` resets the tick counter to zero, sets up all plots, then updates all plots.

3.2 go procedure

```
to go
  if ticks >= 1000 [stop]
  if ticks > 10 and totalProduction < 2 [stop]
  performResearch
  advertiseMachines
  hireAndFire
  produceMachines
  buyMachines
  produceGoods
  buyGoods
  publicSpend
  grantLoan
  depositMoneyToCentralBank
  obtainMoneyByCentralBank
  if turbulence? [move-customers]
  if followLeadBankStrategy? [move-banks]
  ;print totalProduction
  tick
end
```

The second check is to prevent a complete meltdown of the economy, i.e. empty lists and division by zero.

3.3 buyGoods

```
to buyGoods
  ask laborForce [ifelse employed = true [let salary liquidAssets - previousStepLiquidAssets
                                         set expenditure (1 - s) * salary
                                         if random-float 1 < ni [
                                           let nearestBank min-one-of banks [distance myself]
                                           let aBank one-of banks
                                           every ticks [set liquidAssets liquidAssets + (s * salary) * eta3]
                                           ifelse random 10 < customersRationalityLevel [
                                             ask nearestBank [set liquidAssets liquidAssets + s * salary
                                                              set deposits deposits + s * salary
                                                              every ticks [set liquidAssets liquidAssets - (s * salary) * eta3
                                                                           set governmentAssets governmentAssets + (s * salary) * eta3 * fi
                                                                           set liquidAssets liquidAssets - (s * salary) * eta3 * fi]]]
                                            [ask aBank [set liquidAssets liquidAssets + s * salary
                                                        set deposits deposits + s * salary
                                                        every ticks [set liquidAssets liquidAssets - (s * salary) * eta3
                                                                     set governmentAssets governmentAssets + (s * salary) * eta3 * fi
                                                                     set liquidAssets liquidAssets - (s * salary) * eta3 * fi]]]]]
                                        [set expenditure liquidAssets / 2 ]

                   let knownProductFirms n-of (J / 10) productFirms

                   while [any? (knownProductFirms with [not empty? goodsList])] [

                     let chosenProductFirm max-one-of (knownProductFirms with [not empty? goodsList]) [mean goodsList]
                     let localgoodsList sort ([goodsList] of chosenProductFirm)

                     ifelse (1 + mu)*(sum localgoodsList) <= expenditure [
                       set liquidAssets liquidAssets - (1 + mu) * (sum localgoodsList)
                       set expenditure expenditure - (1 + mu) * (sum localgoodsList)
                       ask chosenProductFirm [
                         set pastSales pastSales + length goodsList
                         set liquidAssets liquidAssets + sum goodsList + (1 - fi)* mu * (sum goodsList)
                         set governmentAssets governmentAssets + fi * mu * (sum goodsList)
                         set goodsList []   ]]

                     [let temp 0 let m 0
                       while [expenditure > temp and
                         not empty? localgoodsList] [set temp temp + (1 + mu) * (first localgoodsList)
                         set localgoodsList but-first localgoodsList
                         set m m + 1]
                       if m = 1 or m = 0 [stop]

                       set liquidAssets liquidAssets - (1 + mu) * (sum (sublist (sort([goodsList] of chosenProductFirm)) 0 (m - 1)))
                       set expenditure expenditure - (1 + mu) * (sum (sublist (sort([goodsList] of chosenProductFirm)) 0 (m - 1)))
                       ask chosenProductFirm [
                         set pastSales pastSales + m
                         set liquidAssets liquidAssets + (sum (sublist (sort([goodsList] of chosenProductFirm)) 0 (m - 1))) +
                                                        (1 - fi)* mu * (sum (sublist (sort([goodsList] of chosenProductFirm)) 0 (m - 1)))
                         set governmentAssets governmentAssets + fi * mu *(sum (sublist (sort([goodsList] of chosenProductFirm)) 0 (m - 1)))
                         set goodsList sublist (sort(goodsList)) m (length goodsList)
                       ]]

                   ]


      set previousStepLiquidAssets liquidAssets
      ]
end
```

The spending decisions depend on whether the consumer is employed or unemployed. Since both machines and goods have been produced, first difference between current liquid assets and previous step liquid assets is just the worker's wage. Depending on the probability `ni` that a worker deposits its savings in a bank, if the `customerRationalityLevel = 10`, the worker certainly deposits his savings at the nearest bank to himself; otherwise if the `customerRationalityLevel = 0`, he undoubtedly deposits his savings at any bank. `s * salary` is the money saved by the worker. Every time step the worker receives a fraction `(s * salary) * eta3` of the money he has deposited in a bank while `(s * salary) * eta3 * fi` indicates the fraction of his financial returns he has to pay towards the public sector (taxes).

3.4 grantLoan

```
to grantLoan
if ticks > 10 [
  ask (turtle-set rdFirms productFirms laborForce banks) [
    set loan 0.1 * liquidAssets
```

`[...]`

```
        if any? needersOfLoan[
              ask needersOfLoan [ifelse random 10 < customersRationalityLevel
                [create-liquidAssetRelation-with min-one-of other banks [distance myself] [set color pink
                                                                                           set hidden? true]]
                [create-liquidAssetRelation-with one-of other banks [set color pink
                                                                     set hidden? true]]
                                 if breed != banks and satisfaction < satisfactionCutOff [
                                   ask my-liquidAssetRelations [die]
                                   create-liquidAssetRelation-with min-one-of banks [distance myself] [set color pink
                                                                                                       set hidden? true]]]
              let creditorBanks liquidAssetRelation-neighbors with [breed = banks]
              ask creditorBanks  [let bankReserve re_0 * deposits
                                  let pDebtorsOfaBank liquidAssetRelation-neighbors
                                  if liquidAssets - bankReserve < sum [loan] of pDebtorsOfaBank and any? pDebtorsOfaBank
                                     [while [liquidAssets - bankReserve < sum [loan] of pDebtorsOfaBank]
                                            [ask one-of pDebtorsOfaBank [ask my-liquidAssetRelations[die]
                                                                         set pDebtorsOfaBank pDebtorsOfaBank with [self != myself]]]]
                                  let debtorsOfaBank pDebtorsOfaBank
                                  ask my-liquidAssetRelations [set hidden? false]
                                  set n_loans n_loans + count debtorsOfaBank
                                  set liquidAssets liquidAssets - sum [loan] of debtorsOfaBank
                                  ask debtorsOfaBank  [
                                    set liquidAssets liquidAssets + loan
                                    set debt loan
                                    every ticks [ifelse debt > 0[
                                       set liquidAssets liquidAssets - loan / numberOfInstallments - (loan / numberOfInstallments) * eta1
                                       set debt debt - loan / numberOfInstallments]
                                      [ask my-liquidAssetRelations [die]
                                       set debtorsOfaBank debtorsOfaBank with [self != myself]]]]
                                  every ticks [
                                    set liquidAssets (liquidAssets +
                                                     (sum [loan] of debtorsOfaBank) / numberOfInstallments +
                                                     eta1 * (sum [loan] of debtorsOfaBank) / numberOfInstallments)]]]
    ]
  ask (turtle-set rdFirms productFirms laborForce) [
    ifelse any? liquidAssetRelation-neighbors [
      let neighborBankDistance distance one-of liquidAssetRelation-neighbors
      ifelse neighborBankDistance > 0 [set satisfaction 1 / neighborBankDistance]
                                     [set satisfaction 1]]
     [set satisfaction 1000]]]
end
```

I have initialized the `liquidAssets` of each bank to zero, by starting this procedure after 10 time steps from the beginning of the simulation, banks accumulates enough money to grant loans; otherwise if the procedure started when tick counter reports 0, for the duration of the whole simulation the `liquidAssets` of each bank should be equal to zero, too. I needed a function that made the `union` of four different agent-sets but unfortunately in NetLogo there isn't an analog one, hence I write a report procedure at the end of the code to carry out this task. `pDebtorsOfaBank` stands for "possible debtors of a bank", therefore if a bank holds enough money to grant a loan to all of its "possible debtors", they actually become `debtorsOfaBank`. Obviously debtors must settle their own debt plus the quantity `loan * eta1` (interest rate on debt), deferring the payment in a certain number of installments (fixed in the setup procedure), one every tick. Finally, if the distance between a bank and its customer is equal to zero the client's satisfaction level tends to infinity, that's why I fix `satisfaction = 1000` in this case.

3.5 depositMoneyToCentralBank

```
to depositMoneyToCentralBank
   let RE sum [deposits] of banks
   ask banks [set deposits (1 - re_0) * deposits
              every ticks [set liquidAssets liquidAssets + deposits * re_0 * eta4
                           set governmentAssets governmentAssets + deposits * re_0 * eta4 * fi
                           set liquidAssets liquidAssets - deposits * re_0 * eta4 * fi]]
   ask centralBanks [set deposits deposits + RE * re_0]
end
```

Banks deposit a fraction `re_0 * deposits` of their deposits at the Central bank. If `re_0 > 0`, they get a `deposits * re_0 * eta4` quantity; but if `re_0 < 0`, storing money at the Central bank they lose that quantity. At last `deposits * re_0 * eta4 * fi` represents the taxation on financial return.

3.6 obtainMoneyByCentralBank

```
to obtainMoneyByCentralBank
  if ticks > 10 [
  ask centralBanks [let banksNeedersOfLoanByCentralBank n-of ceiling((count banks with [
          debt > 0 and liquidAssets > centralBankLoan + eta2 * centralBankLoan]) * 0.25) banks with [debt > 0 and liquidAssets > centralBankLoan +
                                                                                                     eta2 * centralBankLoan]
          if random-float 1 < quantitativeEasing [
             ask banksNeedersOfLoanByCentralBank [create-centralBankRelation-with one-of centralBanks [set color brown]]
             let debtorsOfCentralBank centralBankRelation-neighbors with [breed = banks]
             ask debtorsOfCentralBank  [set centralBankLoan liquidAssets * 0.1
                                        set liquidAssets liquidAssets + centralBankLoan
                                        set centralBankDebt centralBankDebt + centralBankLoan
                                        every ticks [ifelse centralBankDebt > 0[set liquidAssets liquidAssets -
                                                                                    centralBankLoan / numberOfInstallments -
                                                                                    (centralBankLoan / numberOfInstallments) * eta2
                                                                                set centralBankDebt centralBankDebt -
                                                                                    centralBankLoan / numberOfInstallments]
                                                                                [ask my-centralBankRelations [die]
                                                                                 set debtorsOfCentralBank debtorsOfCentralBank with [self != myself]]]]
                 ]]]
end
```

It's very similar to the "grantLoan" procedure excepted for that Central bank grant loans *only* to banks with a certain `quantitativeEasing` probability.

3.7 move-customers

```
to move-customers
  ask (turtle-set rdFirms productFirms laborForce) [right random 360
                                                    forward 1]
end
```

When "turbulence" is turned on `rdFirms`, `productFirms` and `laborForce` take a random walk.

3.8 move-banks

```
to move-banks
  ask banks [ifelse liquidAssets + deposits = max[liquidAssets + deposits] of banks [set lead_bank true
                                                                                     set color blue]
                                                                                    [set lead_bank false
                                                                                     set color pink]]
  let leaderBank banks with [lead_bank = true]
  ask leaderBank [let leaderBankNearPatches patches in-radius 5
                  ask other banks [move-to one-of leaderBankNearPatches]]
end
```

If "followLeadBankStrategy" is turned on, `banks` move around the *leader bank* (the blue one).

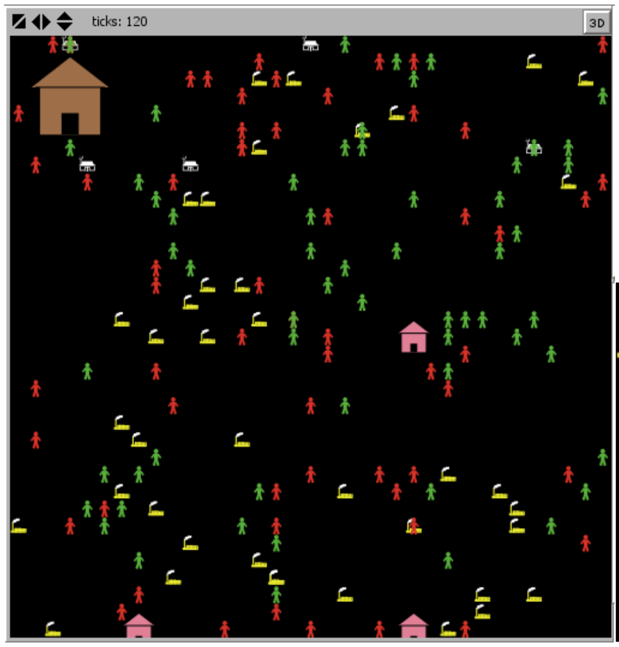
4 Data analysis

The list of benchmark parameters is given in Table 1. With these parameters, a typical image appearing on NetLogo's View is that in Figure 1.

Interface input	Symbol	Value
HowManyRdFirms	I	5
HowManyProductFirms	J	40
HowManyWorkersConsumers	N	120
HowManyBanks	P	3
interestRateRestitution	η_1	0.10
CBRateRestitution	η_2	0.00
interestRateDeposit	η_3	0.00
CBRateDeposit	η_4	-0.03
savings_in_aBank_probability	ϖ	0.8
customersRationalityLevel	none	8
bankReserveRatio	re_0	0.1
quantitativeEasing	none	0.4
satisfactionCutOff	none	0.1
turbulence?	none	off
followLeadBankStrategy?	none	off
InvestmentInRD	ν	0.10
ChanceNewDiscovery	ζ	2.6
TechnologicalDevelopment	α	1.2
TechnologicalDevelopment	β	37
Extension of advertisement	γ	0.5
Markup	μ	0.2
SavingRate	s	0.5
FlexibilityLabour	ω	0.1
TaxesLevel	ϕ	0.75
SpendingInUnemploymentAid	δ	0.1

Table 1 : Benchmark Parameters

If one fixes quantitativeEasing and savings_in_aBank_probability to zero, and switches off turbulence and followLeadBankStrategy; banks and the Central bank don't take part to the simulation. In this scenario, apart an initial pick, an equilibrium pattern emerges (cf. Fig 2), that's why I keep fixed ν, ζ, μ, s, $\omega, \phi, \delta, \eta_1, \eta_3, \varpi$ and `customersRationalityLevel` with the values of Table 1 for the whole duration of the simulation.



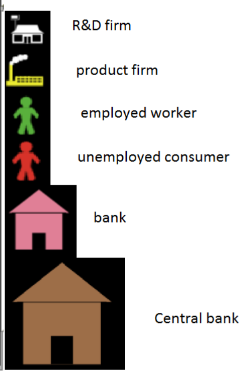
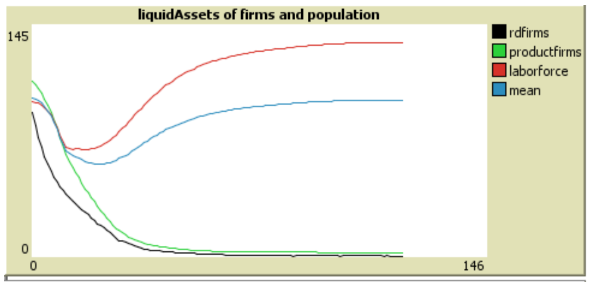
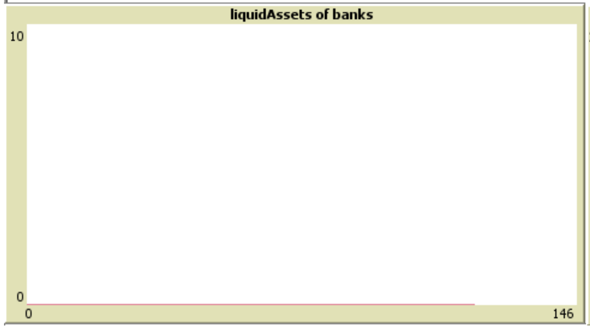


Figure 1: (a) NetLogo' s view (b)NetLogo's world caption





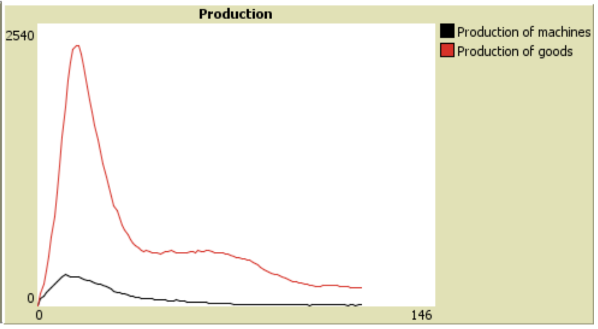


Figure 2: (a) liquid assets of agents (b) level of aggregate production

In order to get better results, I decided to turn to Mathematica. Once installed in Mathematica, the NetLogo package can be loaded with the command `« NetLogo`` ` ``. To start NetLogo you only need to type the command `NLStart[]`, and use the file browser to locate the NetLogo parent directory. Use the command

```
NLLoadModel[ToFileName[{$NLHome, "directory"}, "file_name.nlogo"]]
```

to load the model from the "directory".

4.1 Banking strategy

I now investigate the relationship between *turbulence* and the *cut-off satisfaction level* of customers. The test bank, Bank 165, as mentioned above, can adopt only two strategies: "staying still" and "following the leader". Let's analyze them, here it is a little piece of Mathematica code to plot the "average profit of Bank 165", calculated on a cycle of 100 *ticks*, vs. the "cut-off satisfaction level":

```
BankProfitMean[satisfactionCutOff_] := Module[{}, NLCommand["set satisfactionCutOff ", satisfactionCutOff, "setup"] ;
GeometricMean[NLDoReport["go", "[liquidAssets] of bank 165", 100]]
   ];
satisfactionCutOffs = Table[satisfactionCutOff, {satisfactionCutOff, 0, 0.7, 0.1}]
NLCommand["no-display"];
NLCommand["set turbulence?", false]
Data = Map[BankProfitMean, satisfactionCutOffs];
NLCommand["set turbulence?", true]
DataTurbulence = Map[BankProfitMean, satisfactionCutOffs];
```

4.1.1 Stay still strategy

As you can see (Fig 3), the "turbulence-off" and the "turbulence-on" curves have a very similar trend, only at *cut-off satisfaction level* $= 0.7$ there is a substantial difference between the two ones. In [3] the same thing happens at *cut-off satisfaction level* $= 700$, therefore the results I obtained, which suggest that the "Stay Still Strategy" isn't very liable to a turbulent environment, are very similar to his ones.

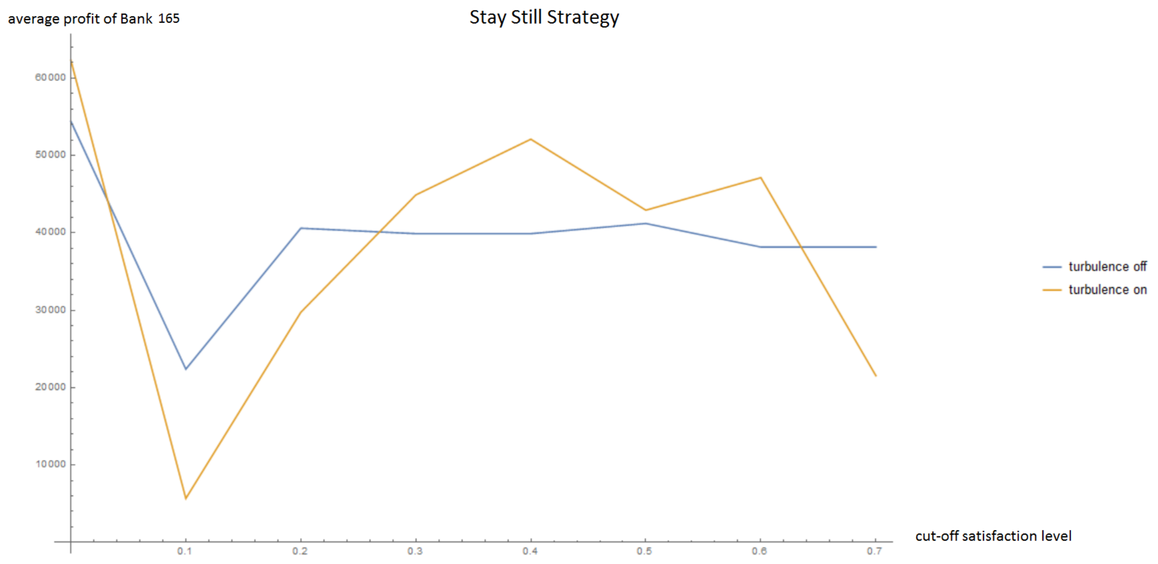


Figure 3: "Stay still" strategy

4.1.2 Follow leader strategy

In NetLogo simulation I point out the leader bank, setting its own color on blue (Fig 4).

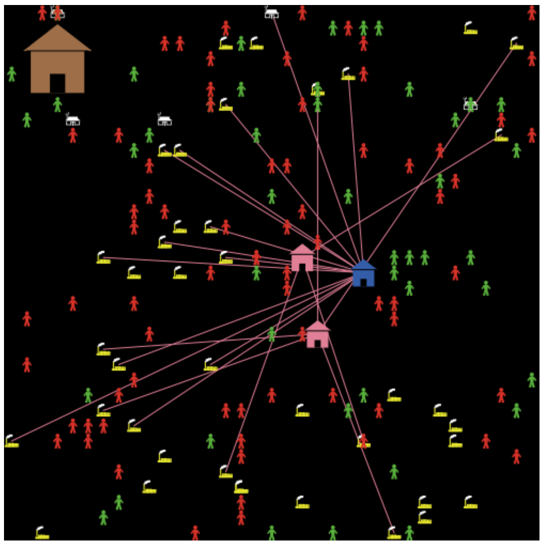


Figure 4: leader bank's view in NetLogo

As shown in Fig 5, the “turbulence-off” and the “turbulence-on” curves, except at *cut-off satisfaction level* = 0.7 where the two ones seem to tend to the same value, now have a quite different trend.

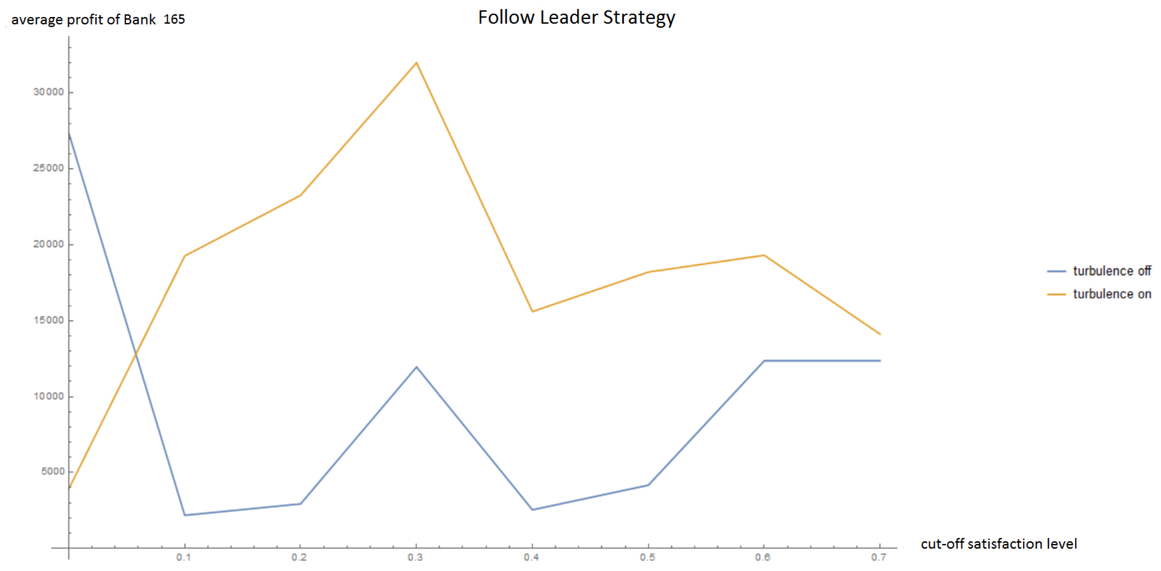


Figure 5: “Follow Leader” strategy

The “follow leader strategy” i.a. seems to obtain better results with low values of *cut-off satisfaction level* and *turbulence* switched off, the reversed situation occurs for *cut-off satisfaction level*’s greater values. Finally, one can notice that all of the features I pointed out are presented in [3].

4.1.3 Considerations

From what I pointed out above it’s clear that in a *non-turbulent* environment (turbulence switched off), the *Stay still* strategy is the most suitable one; this means that a bank’s manger should adopt a *Deliberate* strategy in such situation. Conversely in a *turbulent* environment (turbulence switched on), the *follow leader* seems to be the most appropriate strategy; hence, in such framework, a bank’s manger should opt for an *Emergent* strategy.

4.2 Monetary policy parameters

I now study the effect of monetary policy parameters' variation on GDP. In order to carry out this task I've used Mathematica to plot the *total production* time series varying,on a cycle of 100 *ticks* and with turbulence switched on, the values of the parameters: `quantitativeEasing` (Fig 6), η_2 (Fig 7), η_4 (Fig 8) and re_0 (Fig 9). To obtain better results I've also increase the agents' number within the simulation fixing: $I = 6$, $J = 45$, $N = 250$, $P = 3$. Here it is a little piece of Mathematica code:

```
NLCommand["set turbulence?", true]
NLCommand["set followLeadBankStrategy?", false]
TotalProductionTimeSeries[quantitativeEasing_] :=
  Module[{},
   NLCommand["set quantitativeEasing ", quantitativeEasing, "setup"];
   NLDoReportWhile["go", "totalProduction", "ticks < 100"]
   ];
quantitativeEasings = Table[quantitativeEasing, {quantitativeEasing, 0, 1, 0.2}]
NLCommand["no-display"];
totalProductionData = Map[TotalProductionTimeSeries, quantitativeEasings];
  (* create a color for each austerity *)
numColors = Length[quantitativeEasing];
quantitativeEasingColors =
  Table[Blend[{ {1, Yellow}, {numColors, Red}}, n], {n, numColors}];

(* makes each run equal length *)
maxSteps = Max[Length /@ totalProductionData];
completedData = (PadRight[#, maxSteps, Last[#]] &) /@ totalProductionData;
```

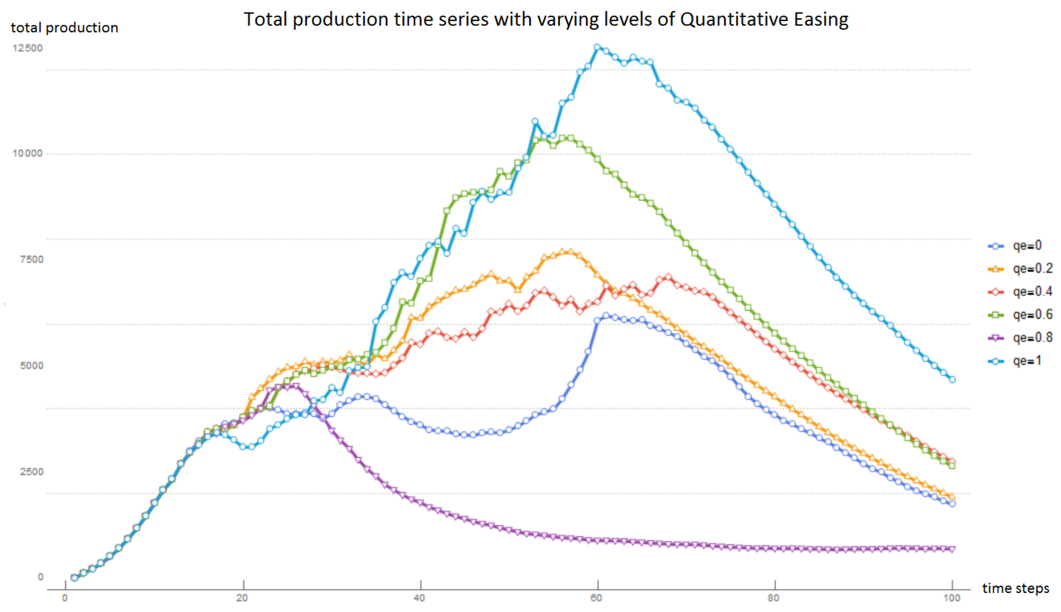


Figure 6: Growth at several Quantitative Easing levels

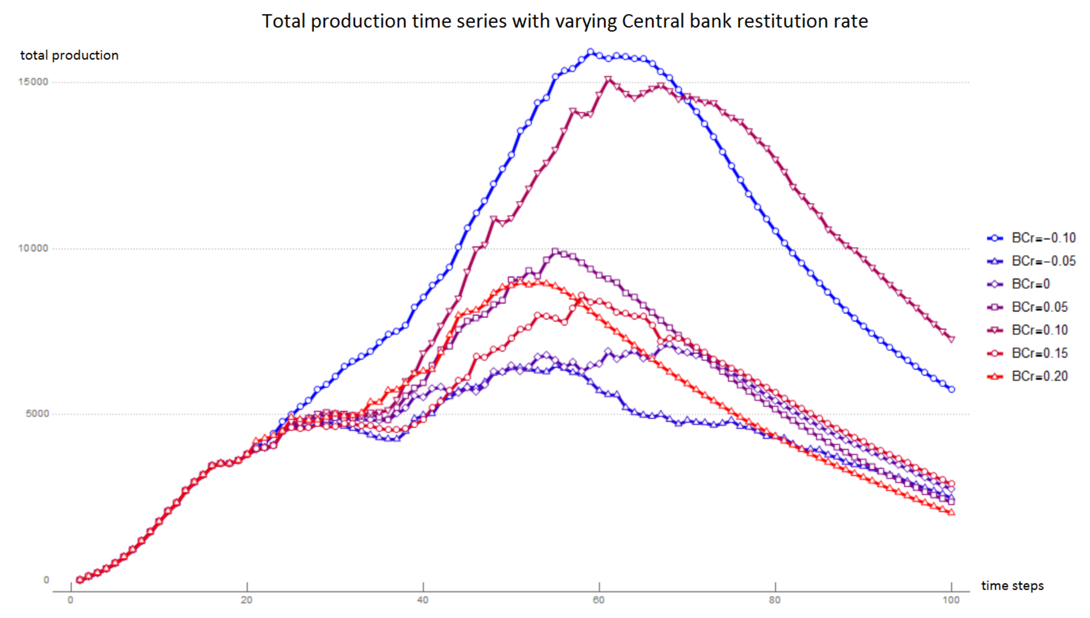


Figure 7: Growth at several Central bank restitution rates

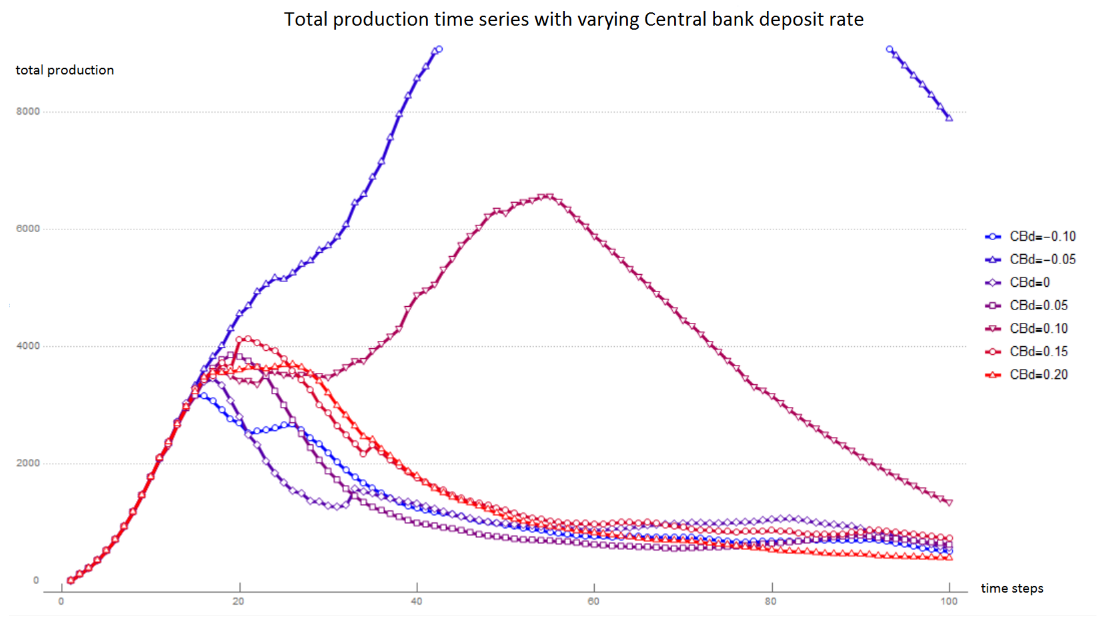


Figure 8: Growth at several Central bank deposit rates

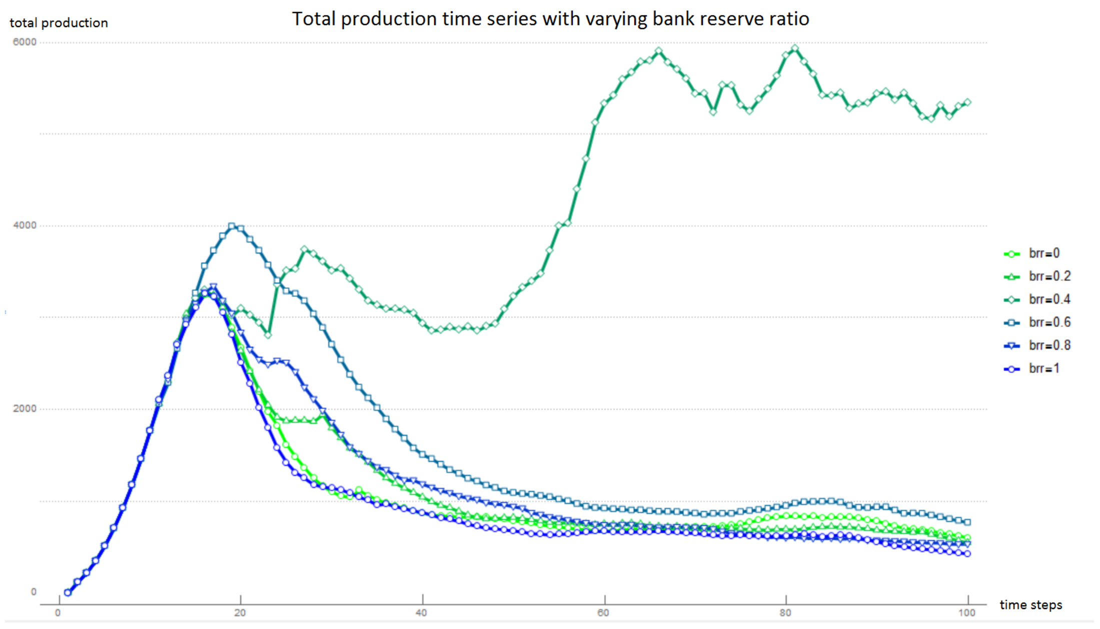


Figure 9: Growth at several Central bank reserve ratios

It seems clear that the total production is maximized by the parameters:

Parameter	Symbol	Value
quantitativeEasing	none	1
CBRateRestitution	η_2	± 0.10
CBRateDeposit	η_4	-0.05
bankReserveRatio	re_0	0.4

The results reflect what one would expect. In particular the double value of the Central bank restitution rate (η_2) said us that, if a percentage of the debt, a bank has against the Central bank, is given by the latter, the Grow increases; but for only a short time compared to the opposite situation where a bank, applying for a loan, has to pay an interest rate to the Central bank (its creditor).

5 Conclusion

In this work I've studied a non equilibrium model, not treatable with analytical tools, displaying the strategy a bank should adopt to maximize its profits

within two different environments and the monetary policy the Central bank should adopt to support growth, without imposing any ex ante consistency requirement but the Central bank's unlimited money supply.
It's interesting to note that the model, set with parameters of Table 1 but turbulence switched on, shows a deep recession lasting approximately nine ticks (Fig 10) after the maximum level of total production the system reaches in the simulation. In particular the sudden decline of growth, as one would expect, corresponds to an instant increase of the mean of banks' liquid assets.

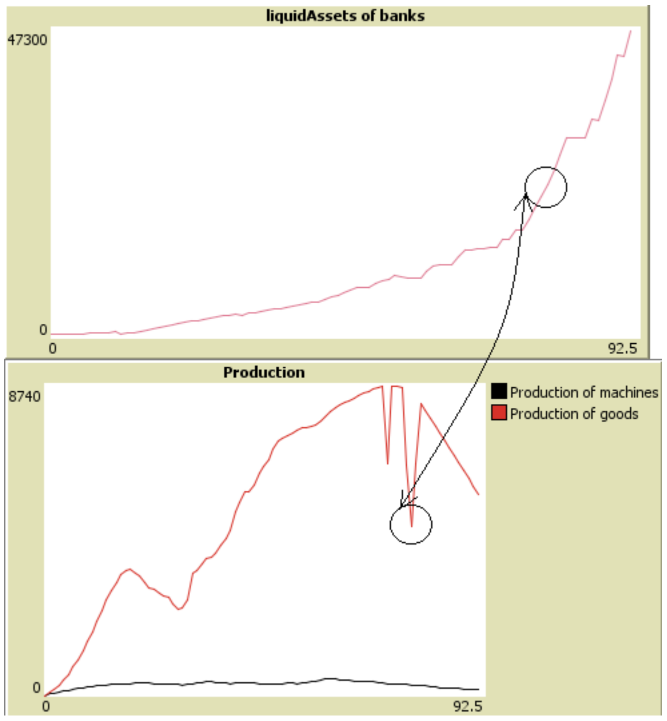


Figure 10: recession scenario

Possible improvements of this model are quite limited by computational capabilities; anyway some aspects such as the limited Central bank's money supply, the presence of more than one Central bank as well as increasing the number of available strategies (i.e. within a *strategy hypercube*) a bank can adopt, could be explored. It could be also interesting to complicate the principles governing the granting of loans and the interactions between agents in the economy.
Finally, in my simulation I introduced significant improvements vis-a-vis [3],

indeed my model suggests the better strategy a bank's manager should adopt even taking into account the Central bank's monetary policy, but turning down a strategy space's dimension (the "Customer Center" strategy described in [3]).

References

[1] Giorgio Fagiolo and Andrea Roventini. Macroeconomic policy in dsge and agent-based models. *Revue de l'OFCE*, (5):67–116, 2012.

[2] Marco Pangallo. Policy experiments in a macroeconomic model with endogenous growth.

[3] Duncan A Robertson. Agent-based models of a banking network as an example of a turbulent environment: the deliberate vs. emergent strategy debate revisited. *Emergence*, 5(2):56–71, 2003.

[4] Julio Rotemberg and Michael Woodford. An optimization-based econometric framework for the evaluation of monetary policy. In *NBER Macroeconomics Annual 1997, Volume 12*, pages 297–361. MIT Press, 1997.