An agent-based approach for a macroeconomic model: the simplified economy of a country

Maurizio Dipierro, Pietro Foini, Giuseppe Ortolano, Daniele Rama

September 25, 2018

Abstract

The report presents an agent-based approach for a macroeconomic model populated by households and firms, which interact through the market of one only consumer good. This work focuses on the observation of some market quantities, including prices, employment rate, total firms profit and many others, aiming for stability and robustness to be found. Simulations over a wide set of parameters shows price stability and a plausible employment rate oscillation.

The second part of this work involves the presence of the government through the agent Public Administration, which interacts with the system purchasing goods and hiring workers. As well as before, the integrated model economy stabilizes after a transient.

1 Introduction

Agent-Based Models are used as a tool for modelling and observing interactions between entities, namely the agents, which perform some kind of actions. In their early developments, the main applications of their usage were attempts to run sociologic and game theory experiments from a computational point of view; the results of those experiments were impressive because such an approach allowed researchers to observe emergent phenomena[6]. Consequently, the techninques took place and covered many scientific areas including biology, ecology and many other kinds of complex systems, as for economics.

Indeed, economic processes could be interpreted as the result of the interaction of multiple entities and this is the reason why they are particularly appropriate for an agents approach. This turns out to be true expecially for macroeconomics models, in which a limited variety of agents are involved

and emergent phenomena are clearly expected. Many economists and complex phenomena researchers have been contributed to the development of this study area such as K. Stieglitz[7], C. Bruun [2] while in recent times CK. Chan [3] and many others.

Modelling a macroeconomic system allows to observe with accuracy the economy of a closed system, such as a country or a city. A market of one or more types of goods could be implemented in order to describe the evolution of a set of quantities (prices, total offer, total demand) which regard all the partecipants, for example including households, firms, finance institution and the government.

This is the aim of the present project: to develop a simplified macroeconomic model able to reproduce a stablility scenario, using a limited number of agents. For the authors, the observed economy takes place in the italian country[1]. The evolution of some national economic indicators, such as the GDP and the deficit is studied.

This work is meant to be the basement of a wider investigation, namely the withdrawal of a country from a political and monetary union; as a matter of fact, this has turned out to be a current study area of increasing academic interest [5] [1] [4].

2 Model

In our model there are three different classes of agents:

- Firms
- Households
- Public Administration

Those agents interact with each other following basic rules of economy. The households, if employed, receive a wage from the firms and consume a fraction of it every cycle. This, in conjunction with the investments from all the agents and the consumption from the unemployed households[2], is added to form the aggregate consumption that define the demand, which is used, with the quantity of goods produced, to determine the market price.

Only one consumer good is produced by all the firms. The production is proportional to the workers of each firm and is tuned at each cycle with hirings or firings, eventually. A continued profit or loss of a firm determines respectively the expansion or bankruptcy of the firm. At a second time a

[1]Some model parameters such as public employment and trade balance value are set equals to the italian values.

[2]Unemployeds receive a benefit.

new agent, representing the Public Administration (PA), is introduced and the results with and without it are compared.

2.1 Agents

In this section the primary variables of each agent are explained, along with their main interaction procedures.

2.1.1 Households

Households attributes initialization are shown in the code below:

```
class Household:

  def __init__(self, is_employed):
    self.is_employed = is_employed
    self.wage = 0
    self.consumption = 0
    self.hasInvested = False
    self.debt = 0
    self.addInvestmentToDemand = False
    self.nInstallmentsleft = 0
```

Listing 1: Households initialization.

Attributes and initialization The leading attribute of the agent class Household is its employment status, namely ***is_employed***.

This is a boolean variable and, if true, the households perceive a wage, unitary and equal for all workers. Otherwise the only revenues of the unemployed households is the unemployment benefit, ***unempBenefit***, which is equal to 30% of the normal wage.

The other households' attributes are:

- ***wage***. The perceived wage equals for all workers and is reduced to 30 % in case of unemployment.
- ***consumption***. Total consumption for the cycle relative to the household.
- ***addInvestmentToDemand***. Each agent has the possibility to invest at a given cycle. This boolean variable records if an investment occurred this cycle and if true adds a constant quantity to the demand.
- ***hasInvested***. A boolean variable that records if the household has any previous debt left to repay.
- ***nInstallmentsleft***. A counter for how many cycles are left until the debt is repaid.

- ***debt***. A real parameter that determines the percentage of wage that is detracted each cycle from the wages of those who invested. Only the interest on the original loan is repaid.

Interactions and functions A household consumes a part of its wage unless it invests. It interacts with the market through its consumptions; the latter is computed using the function **calcConsumption** and its code is shown in appendix A. This is the most important function for the households.

The first attribute that is checked is ***is_employed***. If $False$ the household consumes its whole benefit. If $True$ the consumption is computed using the following formula:

$$consumption = a + b * (wage - debt) + \epsilon \tag{1}$$

where $a = 0.2$ represents sort of a minimum consumption, $b = 0.6$ is the fraction of the wage which the household chooses to consume, $wage = 1$ and ϵ is a normal distributed variable $N(0, 0.1)$; the ***debt*** is set with the parameter ***pmt.repaymentRate*** which is equal to 0.2 if the household is still to repay a debt, otherwise vanishes.

At every cycle each household has a probability $p = 5\%$ to invest. If an investment occurs a quantity of 5 unity is added to their consumption. Only the interest on the investment is repaid, during 10 cycles, during which the household is prevented from investing again.
The other functions are:

- **setEmploymentStatus** is the function which access the variable ***is_employed*** and sets its value to $True$ if the household is a worker and $False$ otherwise.
- **setInvestmentstatus** is the function that access and sets the value of the variable ***hasInvested*** to $True$ or $False$ depending on whether the household has an open investement or not.

2.1.2 Firms

Firms attributes initialization are shown in the code below:

```
class Firm:

  def __init__(self, nWorkers):
    self.nWorkers = nWorkers
    self.prodPerWorker = pmt.prodPerWorker
    self.firmHealth = 0
    self.hasInvested = False
    self.firmPrice = 0
```

```
self.profit = 0
self.production = 0
self.debt = 0
self.nInstallmentsleft = 0
self.addInvestmentToDemand = False
self.consumptionForInvestment = 0
```

Listing 2: Firms initialization.

Attributes and initialization The characterizing attributes of each firm are its number of workers and its profit. The first one determines its productivity (***production***), which is proportional to ***nWorkers***, while the ***profit*** is computed as the difference between revenues and the firm expenditures.

$$production = prodPerWorker * nWorkers \tag{2}$$

As for the households firms can invest and the variables ***addInvestmentToDemand***, ***hasInvested*** and ***nInstallmentsleft*** work in the same way discussed before but, in this case, the investment results in the creation of a new firm.

The other leading variables which set the firm's behaviour are:

- ***debt***. A real parameter that is detracted each cycle from the profit of the firm who invested. As before, only the interest on the original loan is repaid.
- ***firmHealth***. A counter that goes up by one every cycle there is a positive profit and down by the same amount every time it is negative.
- ***firmPrice***. A variable that states the price different for each firm.
- ***consumptionForInvestment***. Firms investments are not fixed but they depend on how many workers are hired for the new firm created by the investment.

Interactions and functions The productivity of each firm is computed using the function **calcProduction** which follows the equation 2.

As mentioned before, each firm applies a different price which is computed using **setFirmPrice**: given a market price determined as the aggregated demand over the total offer, this function returns a new price obtained adding a random quantity normally distributed around zero and with a variance of 20% of the market price.

Another important function is **checkHealth** that checks whether or not a firm is able to invest: if ***firmHealth***$= 5$ the firm invests. Analogously, if ***firmHealth***$= -2$ the firm goes bankrupt.

An investment from a firm results in the creation of a new firm. The amount of quantity for the investment is related to the number of workers of the incumbent firm; the investment's interests, i.e. the debt, are repaid by the firms at the same rate as it was for the households, reducing the firm profit. This implies that, given a variable investment the number of cycles needed to repay it will vary accordingly.

Those investments are the only source of consumption for the firms, so they fully determine the result of the function **calcConsumption**.

```
def calcConsumption(self):

  if self.addInvestmentToDemand == True:
    consumption = self.consumptionForInvestment
    self.addInvestmentToDemand = False
  else:
    consumption = 0
  return consumption
```

Listing 3: The function **calcConsumption** calculates the consumptions of a firm.

The profit is computed using the **calcProfit** function.

```
def calcProfit(self):

  self.profit = self.production*self.firmPrice - self.nWorkers
*pmt.wage
  if self.hasInvested == True:
    self.profit -= self.debt
    self.nInstallmentsleft -= 1
    if self.nInstallmentsleft == 0:
      self.hasInvested = False
      self.debt = 0

  if self.profit < 0:
    self.firmHealth -= 1
  else:
    self.firmHealth += 1

  return self.profit
```

Listing 4: The function **calcProfit** computes the profit of a firm.

As said before, after this function is computed the ***firmHealth*** counter is updated.

Each cycle the firm plan its future production as follows:

$$P_{t+1} = P_t \left(\frac{D_t}{D_{t-1}} \right) \tag{3}$$

where P is the production and D the demand.

Since in our model the productivity factor $prodPerWorker = 1$, the offer equals the number of workers required and the rounded[3] difference $P_{t+1} - P_t = \Delta$ is exactly the number of workers a firm has to hire (or to fire) in order to meet the production plan.

The other functions are:

- **setInvestmentStatus** is the function that, if a firm's investment occurs, access and sets the value of ***hasInvested*** and ***addInvestmentToDemand*** to $True$. Moreover, it takes as argument the number of workers of the new firm to be created[4] in order to compute the number of installments that are left to be repaid.
- **getNumWorkers** returns the number of workers of the firm when is called.
- **updateNumWorkers** changes the number of workers of the in case of hiring or firing.

2.1.3 Public Administration

Public Administration is the agent reflecting the action of the government in the system; this intervention results in the purchasing goods from the market and in the hiring of a given number of households, which is constant during the model's evolution.

The introduction of this agent allows to observe a certain number of new quantity which are useful to evaluate the health of a nation like the deficit, the public debt and the GDP.

Moreover, for the implementation of the PA a new variable is needed for the agent Households, namely the households' ***savings***; this variable represents the fraction of the wage a household chooses not to consume.

Households' savings are fundamental to calculate both the deficit of a nation, which is the difference between the total savings and the public expenditure, and the public debt, which results in the cumulutive deficit over the whole simulation.

```
class PubAdm:

  def __init__(self, nWorkers):
    self.nWorkers = nWorkers
    self.purchasings = 0
    self.PublicExpenditure = 0
```

[3] To the closest integer.

[4] Due to the investment.

```
self.deficit = 0
```

Listing 5: Public Administration initialization.

Attributes and initialization As mentioned before, the number of public employees is fixed in the model and in this case ***nWorkers***= 27% of the initial number of workers[5]. The other main attributes are:

- ***purchasing***. A real variable that stands for the quantity of goods purchased by the PA; this value obviously determines the total demand to raise.
- ***PublicExpenditure***. Every cycle the PA purchases goods and pays its workers. The sum of these two quantity results in the ***PublicExpenditure***
- ***deficit***. A real variable which represents the value of deficit updated at each cycle.

Interactions and functions In order to compute the Public Administration consumptions, the function **calcConsumption** has been implemented: this function returns the value of ***purchasing*** evaluated as:

```
def calcConsumption(self):

    wages = self.nWorkers*pmt.wage
    self.purchasings = wages - random.randint(0, round(pmt.
percPurchasingsPA*wages))
    return self.purchasings
```

Listing 6: The function **calcConsumption** computes the consumptions of the Public Administration.

where $percPurchasingsPA$ is a parameter of the model which equals 0.1.

Another main function is **calcDeficit** which evaluates the deficit, as stated before, and a control index, namely ***ControlMaastr***, calculated as the ratio of the deficit to the GDP. For the european countries, this index must not exceed the 3% as stated in the Maastricht Treaty. The other functions are:

- **getNumWorkers** returns the number of workers of the public employees when is called.
- **calcPublicExpenditure** is the function that computes the sum between purchasings of the Public Administration and the wages of the public employees.

[5]The italian public employment is 27% of the total italian workforce, **REFERENCE NEEDED**

2.2 Run

In this section the computational details of the simulation workflow are given in terms of system's initialization and functions involved.

2.2.1 Model initialization

The leading parameters of the model are the number of firms, the number of households and the number of cycles the simulation will perform. These three numbers are requested to the user who runs the file *start.py* and are used to generate the macroeconomic enviroment.

The enviroment generation follow a meticulous schedule: at first firms are created and, immediately, a certain number of workers are connected to each of them. As a matter of fact, every firm is identified through an index which allows the link between firm and its own workers. The total number of workers is obtained deducting the unemployment rate percentage from the number of households. At the initialization stage, the paramater that control this rate is ***pmt.unemploymentRate*** which is fixed[6] ≈ 0.10;

With the initial parameters mentioned so far the averge number of workers per firm is computed, namely ν. Thus, each firm is provided with a number of workers equals to ν plus a perturbating term randomically extracted from -10 to 10, in order to vary the distribution of employees per firm.

In this step, the variable ***is_employed*** of every workers is obviously set to $True$ and remains $False$ for the unemployed.

The function performing these actions is **buildObjects**:

```
def buildObjects(self):

    self.nu = round((self.nHouseholds/self.nFirms) * (1 - pmt.
    unemploymentRate))

    for i in range(self.nFirms):
      aFirm = Firm(self.nu + random.randint(-10, 10))
      self.firmsDict[self.idFirm] = aFirm
      self.nTotWorkers += askAgent(aFirm, Firm.getNumWorkers)

      hList = []
      for j in range(askAgent(aFirm, Firm.getNumWorkers)):
        hList.append(Household(is_employed=True))
      self.householdsDict[self.idFirm] = hList
      self.idFirm += 1
        self.nUnemployed = self.nHouseholds - self.nTotWorkers

    hList = []
```

[6]The italian unemployment rate is about the 10% (TheGlobalEconomy.com, 2018)

```
for i in range(self.nUnemployed):
    hList.append(Household(is_employed=False))
self.householdsDict[0] = hList
```

Listing 7: The function **buildObjects** generates the whole macroeconomic enviroment.

When the Public Administration agent is integrated in the model, the principal difference that occurs consist in the number of workers available for firm to reduce; actually, a certain percentage of workers, controlled by the parameter ***pmt.publicEmploymentRate*** $= 0.27$, is assigned to the Public Administration.

2.3 Model actions

After the creation of the enviroment the model performs the following scheduled actions, in due order; all these actions represents the development of an unique cycle:

1. At first, the total supply, P, of goods in the market is calculated summing the production of each firm; in order to do this, the function **calcProduction** is called.
2. The total demand, D, is computed summing the single consumptions of each household and of each firm; as mentioned before, every cycle households automatically consume and invest eventually (with a given probability); rather, firms can consume only through investments (which occurs after a certain period of positive profit); these action are computed for both the agents Households and Firms through the corresponding two functions **calcConsumption**. At each cycle the total demand is saved in memory.
   - 2.a With the PA involved the total demand D is increased by the public comnsumptions, the ***purchasings***; this happens through the function **calcConsumption**.
3. Using the quantities computed so far, the market price is computed as $marketPrice = D/P$; the function involved is **setMarketPrice**.
   - 3.a With the PA integration **setMarketPrice** takes into account the raise of the demand caused by the public consumptions.
4. Given the market price, every firm applies its own price through the function **setFirmPrice**; as said before, this results in a certain distribution of prices in the market.
5. The trade happens and every firm calculates its profit with the function **calcProfit** and the total profit is evaluated summing each firm's profit.
6. At this point the health status of each firm is examinated through

checkFirmsHealth; this function saves the indexes related to both the firms which can invest and the firms which are going bankrupt.

7. Firms which are going bankrupt are removed from the market through the function **removeFirms**: this removal happens passing the indexes mentioned before as argument of the function; consequently, the ***is_employed*** status of each former worker is set to $False$.
8. As a consequence of a firm's investment a new firm could be generated[7]. This procedure happens through the function **addFirms** which takes the indexes of the virtuous firms as argument; then, a new firm is created only if there are enough unemployeds.
9. After the first cycle, every firm plans its production for the following time step. This is performed with the function **planProductionandHireFire** which returns the integer Δ; referring to the section 2.1.2, if Δ is negative the firm needs to fire workers in order to match the planned production; instead, if Δ is positive, the firm needs an extra workforce and hires: anyway, the hiring happens only if there are enough unemployeds;

Henceforth the following steps regards the model integrated with the Public Administration.

10. Finally, a certain number of quantities regarding the status of the system are computed: at first the public expenditure is evaluated and it is used to compute the GDP. As a matter of fact, the GDP is calculated as $GDP = D_h + D_f + pubExp + tradeBalance$ where D_h is the demand determined from the households purchasings, D_f the firms' demand and the trade balance is the difference between the amount of exported and imported goods in an economic system; in this model $tradeBalance$ is chosen[8] as 3% of the GDP.

    Another economic indicator of the system which is taken into account and computed is the deficit, calculated as the difference between the total households' savings and the public expenditure.

    Then, the ratio of the deficit to the GDP is observed in order to realize wheter the system is acceptable for the rules of the Maastricht Treaty or not.

    The last relevant quantity which is computed is the public debt and it is calculated adding the current deficit to the previous cumulative deficit.

The set of all of the previous steps forms an entire cycle; the whole simulation

[7]New firms are created following exactly the same procedure of the model initialization. Therefore the incumbent firms will have about the same number of workers of the starting ones.

[8]See https://www.theglobaleconomy.com/Italy/Trade_balance/

lasts the number of cycles which the user has requested.

3 Results

In this section the effectiveness of the model is evaluated and some results are presented. The section is structured in two main parts: the first one regards the outcomes of the simulations through a well defined set of parameters, which can be found in table[9] 2; the second one presents the sensitivity analysis of the model to the above mentioned parameters. In both the subsections comparisons between the presence and the absence of the Public Administration agent are given.

3.1 The baseline model

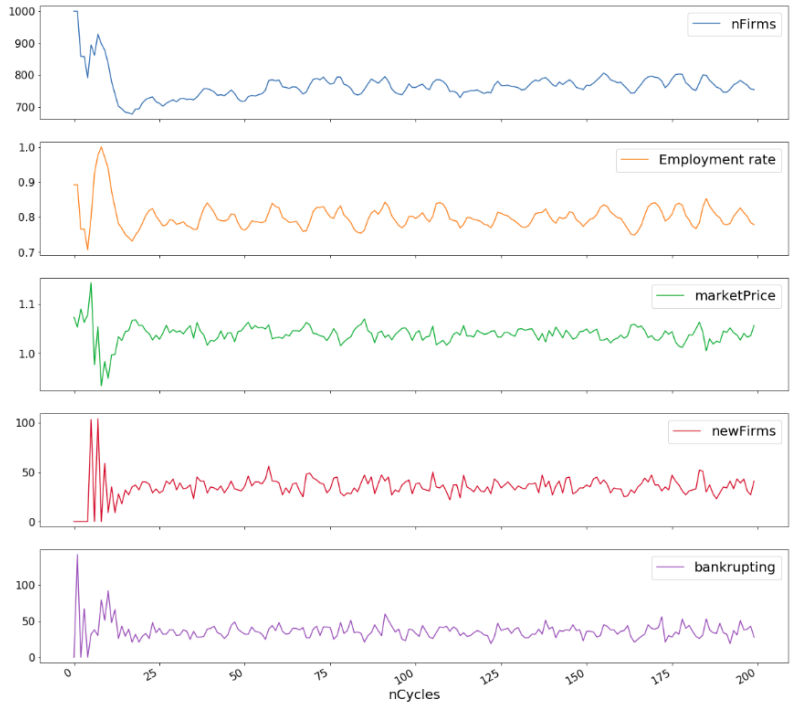


Figure 1: The baseline model.

The following average results arise from simulations with ***nHouseholds*** = 10000, ***nFirms*** = 1000 and over a fixed number of cycle, ***nCycles*** = 200. The latter choice deals with the interpretation of a time step: given that

[9]Appendix G, page 20

firms behaviour involves hirings, firings and production plans, it seems valid to assume a cycle to be equivalent to a quarter. This interpretation does not actually affect the households, which in the model perceive a certain wage and consume a fraction of it: as a matter of fact, one could take this behaviour as an intensive property of a household.

The outcomes of the baseline model, tuned with the parameters in table 2, are shown in figure 1 and in figure 2 for the Public Administration introduction.

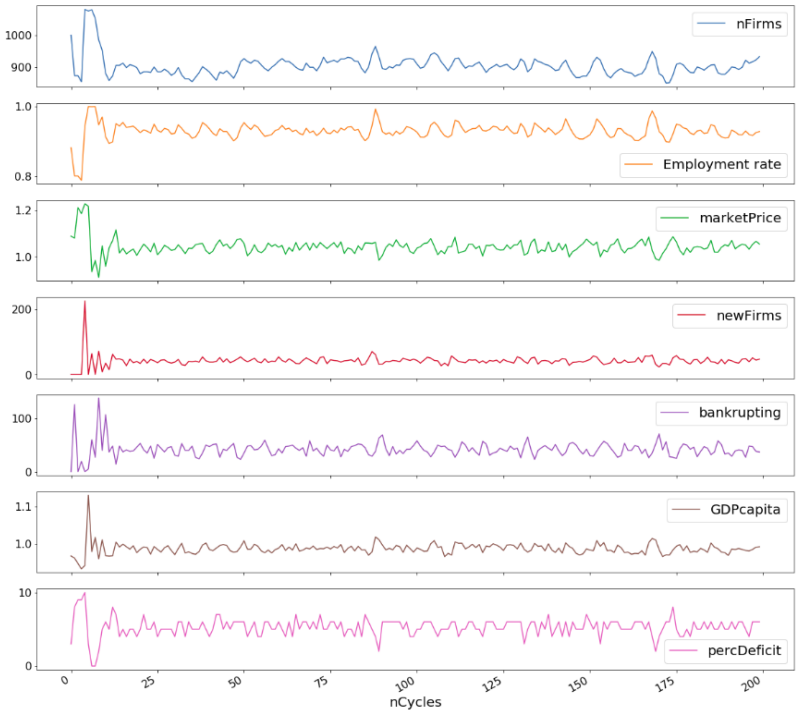


Figure 2: The baseline model integrated with the Public Administration.

As it can be seen, after a transient period, the main economic quantities stabilize. In table 1 some average values of the main quantities are reported; using these values it can be established that households experiment more wealth when the Public Administration agent is involved; as a matter of fact, the employment rate increases and so does the number of firms. The rise of the employment rate is an expected consequence which could be easily explained with the public employment, which is fixed and then represents a sort of offset; meanwhile, the higher average number of firms could be understood as the consequence of more distributed wealth.

In both models the number of active firms stabylizes due to the balance

Observed quantities			
	Initial	Without PA	With PA
Firms	1000	761.90 ± 36.13	883.69 ± 22.45
Employment Rate	0.90	0.80 ± 0.02	0.93 ± 0.02
Market Price	//	1.04 ± 0.01	1.04 ± 0.02
New Firms	//	36.10 ± 6.32	40.30 ± 7.53
Failed Firms	//	35.44 ± 7.10	40.00 ± 9.97
GDP per capita	//	//	0.99 ± 0.01
Deficit/GDP	//	//	5.20 ± 0.95

Table 1: Initial and average quantities observed in the baseline model.

between incumbent firms and firms going bankrupt per cycle[10]. For clearness sake it is useful to remind that the circumstance for a firm to fail or for a new firm to be created are both controlled by the ***firmHealth*** counter[11]; every time this counter equals -2, the correspondent firm fails, meanwhile whenever ***firmHealth*** $= 5$ the firm invests and, eventually, a new firm is created. Afterward, it seems more likely for a firm to go bankrupt due to this asymmetry; nevertheless this balance results and remains true reducing the ***goBankrupt*** value to -1.

The explanation for this occurence is the following: every cycle the firms tune their future production increasing or decreasing it; this happens through both the functions **planProduction** and **planProductionAndHireFire**[12] which take into account the current total demand and the previous one in order to establish the future production. This precaution makes harder for a firm to have negative profit and causes the number of firms to stabilyze.

3.2 Sensitivity analysis

The model turns out to be robust to the variation of the initial parameters in table 2 giving reliable expected outcomes. For instance, reducing the wages forces the demand (and the market price) to decrease, resulting in less distributed wealth[13], while the wage rise ends up with the opposite circumstance. As stated before, the model attempts to reproduce the italian economy through setting some parameters, such as unemployment rate and public employment rate, equals to the italian ones. Nevertheless, the model

[10]See table 1

[11]See section 2.1.2.

[12]See the appendices B and F for more details.

[13]Setting the wage to 0.5 reduces the GDP to a mean value of 0.7 in the PA model and increases the deficit over GDP ratio to 17%.

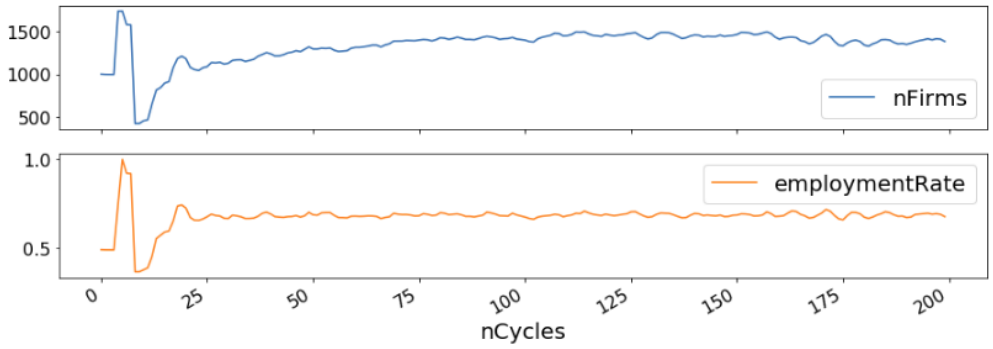


Figure 3: Setting the initial unemployment rate to 0.5 does not affect the main outcomes of the model.

shows robustness over a wide set of those parameters, allowing to explore and to reproduce many different initial condition. Thoroughly, any initial unemployment rate from 0.1 to 0.5 leads to prosperity after a transient of $15 - 20$ cycles, in which the more the unemployds, the more the incumbent firms.

3.2.1 Welfare policies

As mentioned before, the model considers an unemployment benefit which is set to 30% the wage. An interesting outcome has been observed: progressively reducing this percentage forces the economy to collapse within fifty cycles as shown in figure 4a.

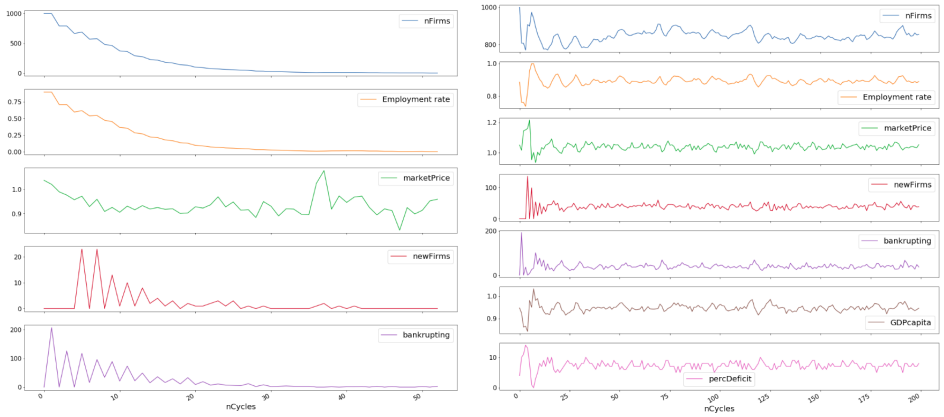
Such an occurrence could be easily explained: the vanishing benefits determines the total demand to decrease; this has a direct consequence on the firms status which, in order to tune their production, automatically fire workers, triggering a chain reaction.

As a matter of fact, the firing results in an additional decrease of the total demand due to the growing number of people who are not working.

On the other hand, when the Public Administration is involved, its presence acts as a lifebelt for the whole system as shown in 4b.

Indeed, the total demand is influenced by the Public Administration in two different ways: firstly, the PA consumes a certain amount[14] of goods every cycle, determining the demand to rise; secondly, PA workers enjoy a permanent contract given that PA can nor hire nor fire and this established a positive offset into the demand; those two occurences makes the economy to last.

[14]Namely, the ***purchasings***.



(a) Model without PA. **(b)** Model with PA

Figure 4: The first plot shows the absence of unemployment benefit makes the economy to collapse within 50 cycles, without the PA; the second one exhibits the economy to last with the PA aid.

4 Conclusions

The main aim of the work was to present a macroeconomic model able to reproduce the economy of a country with reliable characteristics. This has been accomplished through an agent based approach, with the agents being households and firms at first and the public administration at a second time. The main achievement of this model is that it represents a closed economic system which depends entirely on the (bounded) rationality of the agents.

The simulation experiments show stability in production and pricing, with better results after the integration of the public administration.

Though, the model is a baseline for further improvements: as a matter of fact it could be expanded with the involvement of other types of agents such as banks and, at the same time, a labour market could be taken into account.

A calcConsumption

```
def calcConsumption(self):

    if self.is_employed == True:
      if self.hasInvested == False:
        p = random.uniform(0, 1)
        if p < pmt.pInvestHousehold:
          self.hasInvested = True
          self.debt = pmt.repaymentRate
          self.nInstallmentsleft = pmt.
nInstallmentsHouseholds
          self.addInvestmentToDemand = True
    if self.is_employed == True:
      self.wage = pmt.wage
      if self.hasInvested == True:
        self.wage -= self.debt
        self.nInstallmentsleft -= 1
        if self.nInstallmentsleft == 0:
          self.hasInvested = False
          self.debt = 0
      self.consumption = pmt.a + pmt.b*self.wage + random.
normalvariate(0, 0.1)
      if self.consumption > self.wage:
        self.consumption = self.wage
        if self.addInvestmentToDemand == True:
          self.consumption += pmt.
demandForInvestmentHouseholds
          self.addInvestmentToDemand = False
    else:
      self.consumption = pmt.unempBenefit
      self.debt = 0
      self.wage = 0
    return self.consumption
```

Listing 8: Function that computes the consumptions of a household.

B planProduction

```
def planProduction(self, previousDemand, actualDemand):
  if self.production == 0:
    self.deltaWorkforce = 0
  else:
    plannedProduction = self.production * (float(actualDemand)
  / previousDemand)
    plannedWorkforce = round(plannedProduction / self.
prodPerWorker)
    self.deltaWorkforce = plannedWorkforce - self.nWorkers
```

```
return self.deltaWorkforce
```

Listing 9: Function that plane the production of a firm.

C checkFirmsHealth

```
def checkFirmsHealth(self):
  idxToDel = []
  newFirms = []
  for key, aFirm in self.firmsDict.items():
    if askAgent(aFirm, Firm.checkHealth) == -1:
      idxToDel.append(key)
    if askAgent(aFirm, Firm.checkHealth) == 1:
      newFirms.append(key)
  return (idxToDel, newFirms)
```

Listing 10: Function that checks the firm health returning the bankrupting condition or the investment condition.

D removeFirms

```
def removeFirms(self, idxToDel):
  self.firmsDict = {key:self.firmsDict[key] for key in list(
 self.firmsDict.keys()) if key not in idxToDel}
  self.nFirms = len(list(self.firmsDict.keys()))
  print()

  for key in idxToDel:
    for worker in self.householdsDict[key]:
      askAgent(worker, Household.setEmploymentStatus, False)
      askAgent(worker, Household.setInvestmentStatus, False)
      self.householdsDict[0].append(worker)
      self.nUnemployed += 1
      self.nTotWorkers -= 1
    del[self.householdsDict[key]]
```

Listing 11: Function responsible for the removal of a bankrupting firm.

E addFirms

```
def addFirms(self, newFirms):
  self.nNewFirms = 0
  for key in newFirms:
    nNewWorkers = self.nu + random.randint(-10, 10)

    if nNewWorkers <= self.nUnemployed:
      newFirm = Firm(nNewWorkers)
      self.nNewFirms += 1
```

```
        self.idFirm += 1
        self.firmsDict[self.idFirm] = newFirm
        self.nFirms += 1

        self.nTotWorkers += nNewWorkers
        self.householdsDict[self.idFirm] = self.householdsDict
[0][:nNewWorkers]
        for newWorker in self.householdsDict[self.idFirm]:
          askAgent(newWorker, Household.setEmploymentStatus,
True)
          self.nUnemployed -= 1
        self.householdsDict[0] = self.householdsDict[0][
nNewWorkers:]

        askAgent(self.firmsDict[key], Firm.setInvestmentStatus,
nNewWorkers)
```

Listing 12: Function responsible of the integration of a new firm in the market.

F planProductionAndHireFire

```
def planProductionAndHireFire(self):
  for key, aFirm in self.firmsDict.items():
    extraWorkforce = aFirm.planProduction(previousDemand=self.
demand[self.t - 1], actualDemand=self.demand[self.t])

    if extraWorkforce > 0:
      for i in range(extraWorkforce):
        if self.nUnemployed > 0:
          newWorker = self.householdsDict[0][0]
          askAgent(newWorker, Household.setEmploymentStatus,
True)
          self.householdsDict[key].append(newWorker)
          del(self.householdsDict[0][0])
          askAgent(aFirm, Firm.updateNumWorkers, 1)
          self.nTotWorkers += 1
          self.nUnemployed -= 1

    if extraWorkforce < 0:
      for i in range(extraWorkforce):
        newUnemployed = self.householdsDict[key][0]
        askAgent(newUnemployed, Household.setEmploymentStatus,
 False)
        askAgent(newUnemployed, Household.setInvestmentStatus,
 False)
        self.householdsDict[0].append(newUnemployed)
        del(self.householdsDict[key][0])
        askAgent(aFirm, Firm.updateNumWorkers, -1)
        self.nTotWorkers -= 1
```

```
            self.nUnemployed += 1
```

Listing 13: planProductionAndHireFire takes into account the current production and the current profit in order to establish the following time step production through hirings/firings.

G Tables

Model parameters initialization		
Global	**unemploymentRate**	0.1
	repaymentRate	0.2
Firms	**prodPerWorker**	1
	sigmaPrice	0.2
	doInvest	5
	goBankrupt	-2
Households	**wage**	1
	unempBenefit	0.3
	pInvest	0.05
	nInstallments	10
	demandInvestment	5
Public Administration	**publicEmploymentRate**	0.27

Table 2: Set of parameters used in the base model.

References

[1] Alberto Bagnai, Brigitte Granville, and Christian A. Mongeau Ospina. Withdrawal of Italy from the euro area: Stochastic simulations of a structural macroeconometric model. *Economic Modelling*, 64(April):524–538, 2017.

[2] Charlotte Bruun. Agent-Based Keynesian Economics: Simulating a Monetary Production System Bottom-Up. Technical report, 1999.

[3] CK Chan and Ken Steiglitz. An agent-based model of a minimal economy. *Princeton University*, (1994):1–33, 2008.

[4] Domenico Delli Gatti and Saul Desiderio. Monetary policy experiments in an agent-based model with financial frictions. *Journal of Economic Interaction and Coordination*, 10(2):265–286, 2015.

[5] Aristeidis Samitas, Stathis Polyzos, and Costas Siriopoulos. Brexit and financial stability: An agent-based simulation. *Economic Modelling*, (September):1–12, 2017.

[6] Thomas C. Schelling. Dynamic models of segregation†. *The Journal of Mathematical Sociology*, 1(2):143–186, jul 1971.

[7] Ken Steiglitz, Michael L Honig, and Leonard M Cohen. A computational market model based on individual action. Technical report, 1993.