Agent Walras

Stefano Pietrosanti*

September 22, 2012

Introduction

The aim of this work is to give an agent based interpretation of the walrasian General Equilibrium[1] problem. This issue will be faced through the use of Netlogo, a simple program which is a common tool in Agent Based Modelling[2]. With such a device, I will try to model the case of a market of one good, bought and sold by n agents. The equilibrium price's dynamic is defined by the excess demand function through a *tâtonnement* process.

This is a simplified version of the modern reshaping of the way to see the market completely formalized for the first time by Leon Walras in Sections II to VI of his *Éléments d'économie politique pure.* It is easy to find a lot of descriptions of such excess demand approach. In explaining it, I will follow step by step *Microeconomics, Principles and Analysis* by F. Cowell (2005), adopted in the Microeconomic course of the Master Degree in Economics of the University of Turin.

The excess-demand approach

An excess demand function for the good i is so defined:

$$E_i(p) := x_i(p) - q_i(p) - R_i \tag{1}$$

where $x_i(.)$ and $q_i(.)$ are respectively the aggregate demand and the aggregate supply for the i-th element of the vector of goods, R_i is a possible pre-existent stock of the good and p is a given vector of prices, one for each good.

Under this specification, an equilibrium price vector p^* for the market is identifiable *via* the following conditions:

$$\left\{ E_i(p^*) \leq 0; \quad p_i^* \geq 0; \; p_i^* E_i(p^*) = 0 \right. \tag{2}$$

*I would like to thank Ludovico Russo, Oriana Cesari and Giorgio Martini for the useful discussions and advices; obviously all the errors remaining are my responsibility.

[1] GE from now on

[2] ABM from now on

Which, in plain words, state that - at the end of the market process - there must be no excess demand, no negative prices and that any good for which an excess supply is determined must be a free good[3].

Such an E vector of excess demand has some properties, grounded on the assumption of "fully informed, rational, non-satiated agents in a private ownership economy" (Cowell, 2005, p.159). These are:

1. Homogeneity of degree zero
2. Walras's law:

this last is formally stated as the orthogonality between any emerging vector of prices and the consequent vector of excess demand:

$$pE(p) = 0 \tag{3}$$

the first one who called it Walras's Law is Lange (1942), referring to the §116 - Lecture XI of the fourth edition of the Walras's *Éléments.*

From the first, it follows that we can normalize the prices so that $\sum_{i=1}^{k} p_i = 1$; from the second, that the excess demand vector has k-1 "degree of freedom", in fact, knowing $E_1, ..., E_{k-1}$ we also know E_k:

$$E_k = -\frac{1}{p_k} \sum_{i=1}^{k-1} p_i E_i(p) \tag{4}$$

Under such formalization, an equilibrium in the market

EXISTS

if there is no infinite aggregate supply of any one of the goods, so if $E(.)$ is bounded below, and if $E(.)$ is a continuous function[4]

IS UNIQUE

if the Weak Axiom of Revealed Preference holds at an aggregate level, so if there are no income effect which could "disturb" the aggregate behavior of the excess demand function, generating inconsistencies that may lead to multiple equilibria conditions

IS STABLE

if no trading occurs till the equilibrium. This which may seem an odd statement, is at the same time the weak *and* the fascinating point of the walrasian model. Weak, because

[3] A good whose price is null

[4] which is true under strict concavity of each production function and strict quasi-concavity of each utility function

it means to immagine an *auctioneer*'s mechanism to reach stability; fascinating, because it leave space to interpretation, reasoning and further modelling.

Walras formalized the thought experiment of a market where an auctioneer shouts a certain price, then registers if a positive or negative excess demand does emerge and consequently try another higher or lower price till the equilibrium is reached and exchange is allowed. Following Cowell (2005, p. 165), a linear *tâtonnement* process is so defined:

$$\frac{dp_i\left(t\right)}{dt} = \begin{cases} \alpha_i E_i\left(p\left(t\right)\right) & if\, p_i\left(t\right) \geq 0 \\ 0 & otherwise \end{cases} \tag{5}$$

and defined the notion of distance of $p\left(t\right)$ from the equilibrium $p*$ as:

$$\triangle(t) := \sqrt{\sum_{i=1}^{k} \left[p_i\left(t\right) - p_i^*\right]^2} \tag{6}$$

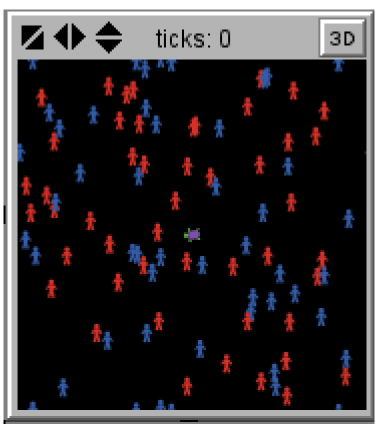
is possible to show (Cowell, 2005, p.166) that it decreases over time, so that the system is stable under the above mentioned assumption.

In the following, I will try to do an AB model of a simplified case from such framework.

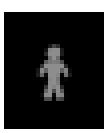
The program

The setting

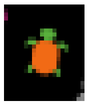
I created a simple Netlogo's world, the auction market for the case of a single good:



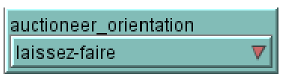
where bidders, represented by:

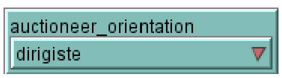


and endowed with a certain stock of the good, face the auctioneer, represented by a turtle setted at the center of the screen:



The bidder's color represents its state in the market: if the color is blue, the bidder desires a smaller amount of good than the one it owns, if the color is red, the other way round. A grey color stands for a bidder whose desired amount is exactly equal to its stock. At each turn (each tick) the auctioneer shouts the price, registers the consequent excess demand and then updates the price till the equilibrium - hopely - is reached. I created a *chooser* to define if the auctioneer stops or not the market when the excess demand function hits the zero:





a "dirigiste" auctioneer will stop the bidders when the excess demand hits the zero, even if the price for which it happens is not a stable one; a pro "laissez-faire" auctioneer will not stop the bargainings instead, letting us see the complete time behavior of the system, so if it reaches an equilibrium prices and mantains it through time.

The procedures

My program has two main parts: the "setup" procedure, in which I created the agents and setted the starting values of the variables, and the "go" procedure, which shows two sub-procedures: the "price-shout" and the "update" of the system. We can start from the setup:

```
create-bidders n-of-bidders

  [set pre_check_owned_amount random-normal endowments endowments_variance

     if pre_check_owned_amount >= 0
       [ set owned_amount pre_check_owned_amount
       ]
     if pre_check_owned_amount < 0
       [ set owned_amount 0
       ]

   set sum_owned_amount (sum_owned_amount + owned_amount)

   set shape "person"
   setxy random-xcor random-ycor
```

```
  set pre_check_desired_amount random-normal desire desire_variance
  set desired_amount (pre_check_desired_amount - owned_amount)
     if desired_amount > 0
      [set color red
      ]
     if desired_amount = 0
       [set color grey
      ]
     if desired_amount < 0
       [set color blue
      ]
  set sum_desired_amount (sum_desired_amount + desired_amount)
 ]
```

While creating our bidders, I defined two random assignment of stocks and desires to our agents, through the "pre_check_owned_amount" and the "pre_check_desired_amount" variables. Each one is a random normal, whose mean and variance can be defined by the user.

After having averted some possible inconsistencies, as a negative assignment for the stocks, I aggregated the stocks over the agents with the "sum_owned_amount" variable. On the other side, the true desire of the i-th agent is defined by the difference between its random assignment and its stock, this way I identified it as a net demader - red color agent - a neutral observer - grey color agent - or a net supplier - blue color agent. Then I aggregated over the agents the real desired amounts.

```
create-auctioneer 1

  [set price initial_price
   set effective_demand (sum_desired_amount - price_sensitivity * price)
   set effective_supply ((sum_owned_amount / 2) + price_sensitivity * price)
   if effective_supply <= sum_owned_amount

     [ set effective_supply effective_supply
     ]

   if effective_supply > sum_owned_amount

     [ set effective_supply sum_owned_amount
     ]

   if price = 0
     [ set effective_supply 0
     ]

   set excess_demand (effective_demand - effective_supply)
   set shape "turtle"
   setxy 0 0
  ]
```

As the auctioneer is created, it shouts an initial price and then records the initial level of the demand and the supply, defined linearly as:

$$D_0 = \sum_{i=1}^{n} desire_i - b * p_0 \tag{7}$$

$$S_0 = \frac{\sum_{i=1}^{n} stock_i}{2} + b * p_0 \tag{8}$$

This way each player contributes both to the demand and the supply, on the bases of its endowments and desires. I wish to stress that the halving of the aggregate stock is part of the precautions I took to avoid irrational cases, as a greater than the stocks supply; economically, it can be justified by a precautional behavior of the agents with respect to their stock. At the end, the excess demand is defined as:

$$\begin{aligned} E_0 &= \sum_{i=1}^{n} desire_i - b * p_0 - \frac{\sum_{i=1}^{n} stock_i}{2} - b * p_0 \\ \Rightarrow E_0 &= \sum_{i=1}^{n} \left[desire_i - \frac{stock_i}{2} \right] - 2b * p_0 = k - 2b * p_0 \end{aligned} \tag{9}$$

where b is defined in the program as the "price_sensitivity", decided by the user through a *slider*.

Then comes the "go" procedure, which is made by two parts: the proper price-shout and the update of the variables

```
to go

  priceShout
  update

 if auctioneer_orientation = "dirigiste" and ticks > 10
 [if excess_demand * excess_demand_in_t-1 <= 0
     [stop
     ]
 ]

     tick
end
```

the orientation of the auctioneer is stated in its end as a *chooser* that makes the auctioneer stop the market the first time the excess demand hits the zero after the tenth tick.[5]

[5]Just a tumb rule I adopted to let the user see the behavior of the functions even under the "dirigiste" case

```
to priceShout

  ask auctioneer

    [ set price_t-1 price
      set excess_demand_in_t-1 excess_demand
      set price (price_t-1 + p_adj_parameter * excess_demand_in_t-1)

      if price >= 0
       [set price price
       ]

      if price < 0
       [set price 0
       ]
      ]

end
```

Here above the price-shout is thoroughly stated. It consists in asking the auctioneer to record the previous price and the previous value of the excess demand and then adjust the price on this bases, taking care to avoid negative prices. Formally, another time through a linear function for simplicity sake:

$$p_t = \begin{cases} p_{t-1} + a * E_{t-1} & if\, p_{t-1} + a * E_{t-1} \geq 0 \\ 0 & if\, p_{t-1} + a * E_{t-1} < 0 \end{cases} \tag{10}$$

where a is the "p_adj_parameter". This parameter defines how strong is the price correction at each time; once more, it is decided by the user through a *slider*. With (9) and (10) a simplified and discrete version of the Walrasian adjustment method is described.

```
to update

  ask auctioneer

   [ set effective_demand_in_t (sum_desired_amount - price_sensitivity * price)
     set effective_supply_in_t (((sum_owned_amount / 2) + price_sensitivity * price))

      if effective_supply <= sum_owned_amount

       [ set effective_supply effective_supply
       ]
      if effective_supply > sum_owned_amount
       [ set effective_supply sum_owned_amount
       ]

      if price = 0
       [ set effective_supply 0
       ]
    ]

 set excess_demand (effective_demand_in_t - effective_supply_in_t)

;-----------------------------;;  THE END ;) ;;--------------------------------------
end
```

The end is the "update" part, where the auctioneer updates the demand and the supply through the law stated in (7) and (8)[6] and the excess demand following (9).

Some mathematics

We can see how the phenomenon described by our program is led by a finite difference system composed by (9) and (10):

$$\begin{cases} E_t = k - 2b * p_t \\ p_t = p_{t-1} + a * E_{t-1} \end{cases} \tag{11}$$

Substituting the first equation in the second and solving for p_t I got:

$$\begin{gathered} p_t = p_{t-1}\,(1-2ab) + a * k \\ \Rightarrow p_t - p_{t-1}\,(1-2ab) = a * k \end{gathered} \tag{12}$$

its homogeneus associated equation is:

$$p_t = p_{t-1}\,(1-2ab) \tag{13}$$

whose solution is the geometric progression with $(1-2ab)$ as the reason and c as the constant term:

$$p_t = c\,(1-2ab)^t \tag{14}$$

The general solution is the sum of (14) and of a particular solution of (12). So, for $p_t = p_{t-1} = p_h$, we get:

[6] with the previous caveats to avoid economically irrational results

$$\begin{array}{c} p_h * 2ab = a * k \\ \Rightarrow p_h = \frac{k}{2b} \end{array} \tag{15}$$

and the general solution is:

$$p_t = c\,(1 - 2ab)^t + \frac{k}{2b} \tag{16}$$

from this I derived p_0 , then I wrote c as a function of p_0 to complete the solution :

$$\begin{array}{c} p_0 = c + \frac{k}{2b} \Rightarrow c = p_0 - \frac{k}{2b} \\ \Rightarrow p_t = \left(p_0 - \frac{k}{2b}\right)(1 - 2ab)^t + \frac{k}{2b} \end{array} \tag{17}$$

In the following, I will keep on considering the problem on the price side.

Conclusion

From (17) we can understand that the convergence or not of the system to an equilibrium price and level of the excess demand is decided by the $(1 - 2ab)$ term.

We can derive five cases, each of one is confirmed by the results of the simulations. In the following, the five cases are thoroughly stated and each time followed by the plot of the behavior of the system under the associated specification of a and b[7]:

1.
$$(1 - 2ab) \leq -1$$
for which the price does not converge and, under the "laissez-faire" option, keep on oscillating to infinity.

2.
$$-1 < (1 - 2ab) < 0$$
for which the price shows an oscillating, but convergent behavior

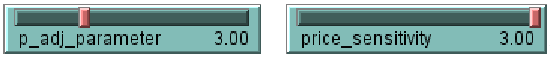
3.
$$(1 - 2ab) = 0$$
for which the price behaves linearly

4.
$$0 < (1 - 2ab) < 1$$
for which the price converges exponentially

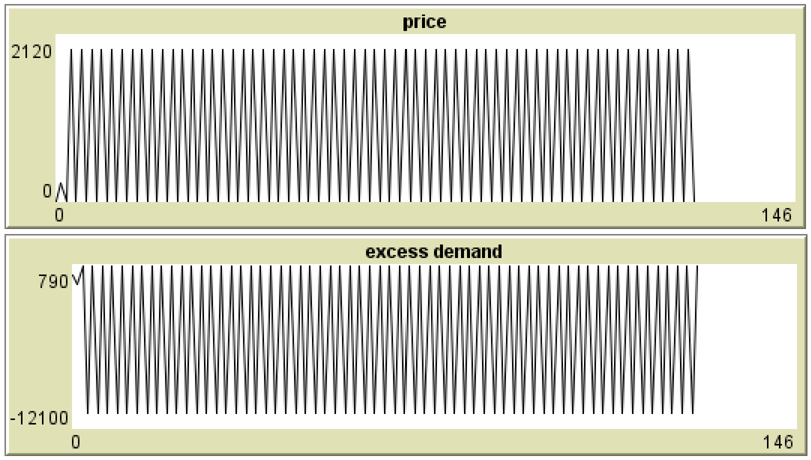
5.
$$(1 - 2ab) > 1$$
which is not really relevant under the simplified perspective I assumed. 5. implies a or b to be negative, which means an auctioneer that adjusts the price in a conterintuitive way, or agents which adjust their demand and supply in a conterintuive way. For this reason, this case is not allowed in the program and no plot is then associated to 5. Thus, the simulation examples follow:

[7] I recall that a is the "price_sensitivity" and b the "p_adj_parameter"

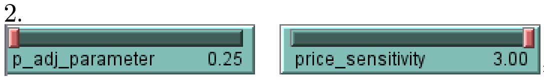
1.



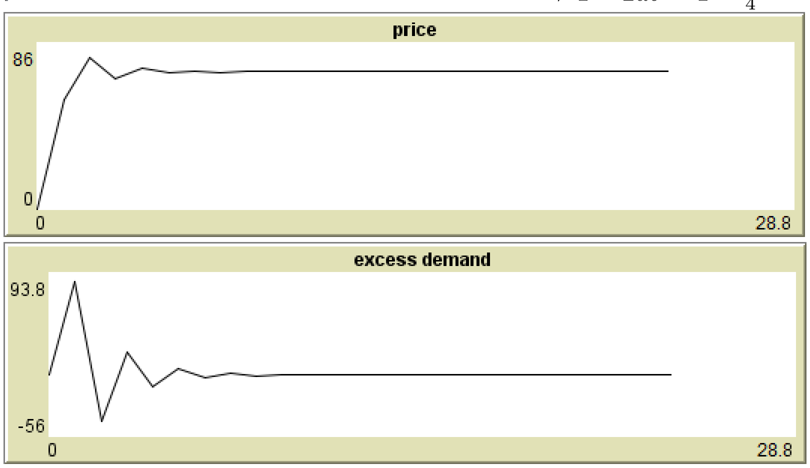
$\Rightarrow 1 - 2ab = 1 - 18 = 17$



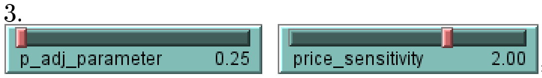
2.



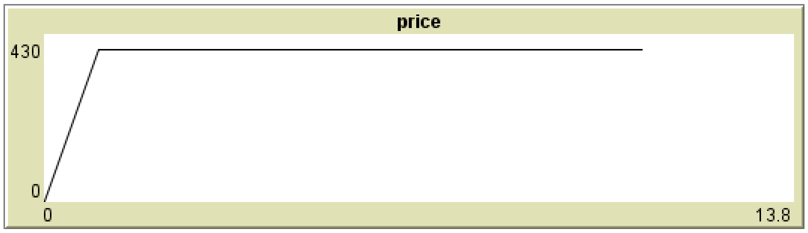
$\Rightarrow 1 - 2ab = 1 - \frac{6}{4} = -\frac{1}{2}$

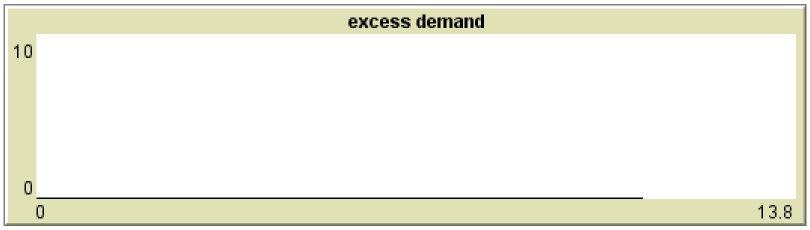


3.

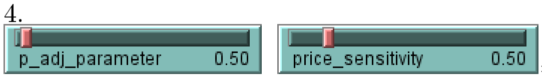


$\Rightarrow 1 - 2ab = 1 - 1 = 0$

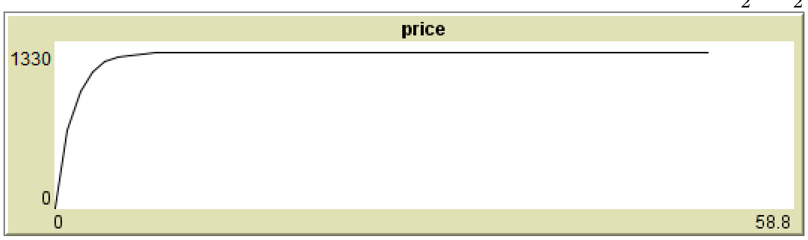


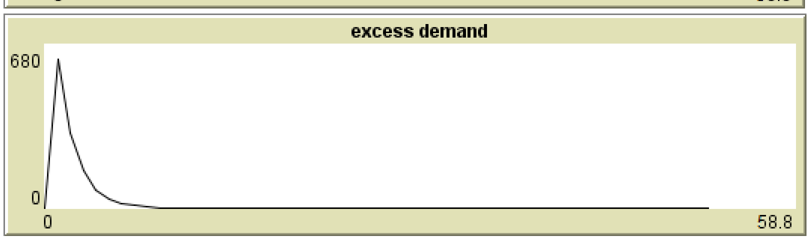


4.



$\Rightarrow 1 - 2ab = 1 - \frac{1}{2} = \frac{1}{2}$





Summing up, we can draw at least a conclusion: in a discrete framework, for a low precision correction system, the decision of the auctioneer to stop the bargaing and allow the exchange is crucial, because no equilibrium price is stable.