國立臺灣大學
National Taiwan University

Economy Simulation: Inequality

Documentation

from

Ricard Milá Bou

Sabrina von Wegerer Elizeche

Michelle Mei-Li Pfister

Department of Economics

National Taiwan University

Professor: Hendrik Rommeswinkel

Table of Contents

1. Introduction

The approach we choose to implement the economy simulation was restructuring and dividing the problem of finding an equilibrium of the economy. We divided the problem in three subproblems, which luckily fit the number of members in our group and assigned one problem to each member. In our group work each member programmed the solution to one problem, as well as provided the documentation to it and we fit the code together in the end to create our economy simulation program.

2. Economic Model

The economy simulation uses the basic general equilibrium model as base to later introduce a public finance question to the simulation. The problem of finding the equilibrium in this economy simulation can be restructured by solving three mostly independent problems: the consumer problem, the producer problem and the excess demand problem.

2.1. Consumer Problem

In order to solve the consumer problem the allocation of goods and factors, which maximizes a consumers utility, must be found. In this problem we assume that the prices for goods and factors are given. The utility a consumer has is described in his utility function, therefore the utility function is the objective that has to be maximized in the consumer problem. Moreover, there is one constraint in the consumer problem: the budget the consumer is able to spend on goods. This budget is the addition of the share of profit a consumer receives and the payment for providing factors a consumer receives. All these information result in the following maximization problem which characterizes the consumer problem:

$$\max_{x_g v_f} \quad u\left(\overrightarrow{x_g}, \overrightarrow{v_f}\right)$$
$$s.t. \quad \overrightarrow{r} \cdot \overrightarrow{v_f} + \pi = \overrightarrow{p} \cdot \overrightarrow{x_g}$$

$\overrightarrow{x_g}$: $vector\ of\ consumption\ of\ all\ goods$
$\overrightarrow{v_f}$: $vector\ of\ factor\ supply\ of\ all\ factors$
$u\left(\overrightarrow{x_g}, \overrightarrow{v_f}\right)$: $utility\ function\ in\ dependency\ of\ the\ factor\ and\ good\ allocation$
π : $share\ of\ profit\ for\ this\ particular\ consumer$
$\overrightarrow{p}$: $price\ vector\ of\ all\ goods$
$\overrightarrow{r}$: $price\ vector\ of\ all\ factors$

2.2. Producer Problem

To solve the producer problem we have to find the allocation of good and factors, which maximizes the producers' profit. With the production function, we obtain the number of goods that can be produced as a maximum of a certain amount of resources and with the minimum cost.
Production Technologies:
Factor f used in the production of good g: r_{gf}
Aggregate amount of good g produced: X^g
Production Function for good g: $X^g = \emptyset(r_{g1,\dots,}r_{gF)}$ or $X^g = \emptyset^g(r_{gF})$
Production Fuction for good g:

$$\emptyset^g(r_{gf}) = \sum_f^f \psi_{gf} \frac{(r_{gf})^{1-\xi_g}}{1-\xi_g}$$

The profit function is the objective that has to be maximized in the producer problem.

$$\max_{x_g v_f} \quad \pi(\overrightarrow{x_g}, \overrightarrow{v_f})$$
$$s.t. \quad \overrightarrow{x} \cdot \overrightarrow{f_g} = \emptyset_g(r_{gf})$$

$\vec{x}$: $vector\ of\ all\ goods\ g$
$\vec{v}$: $vector\ of\ factor\ supply\ of\ all\ factors\ f$

2.3. Market Clearing Conditions

First the two "moving" variables are defined: good price (p) and factor price (r). Their length is given by splitting the inputList. For good prices it goes from 0 to number of Goods, and for the factor prices from number of Goods to number of Goods summed up with number of Factors.
The producer problem follows. By calling the function maxProfit, given p and r, the Producer gives an answer in the format of the array described above. Its answers are summed up to have the total goods produced and factors demanded into SumProd.
Then, the profit number is extracted to distribute it to the Consumers that are also Producers. Taking this into account, the Consumer Problem calls on the maxUtility function and gives its' answer in the format explained above. Its' answers are also summed up to save the total goods demanded and factors provided into SumCon. We return the squared difference between these two. Positivity constraint are added, since we do not want to allow negative prices and then the objective follows, that is, the minimization with the method ='SLSQP' of SciPy and it is saved it into the variable called solution or "sol".

The economics balance in a mathematical way.

$$m = \frac{p_2 - p_1}{x_2 - x_1} \rightarrow \text{Slope equation}$$

$$p - p_1 = m(x - x_1) \rightarrow \text{Demand and supply equation}$$

$$p - p_1 = m(x - x_1)$$
$$p = \left(\frac{p_2 - p_1}{x_2 - x_1}\right) x + \left[\left(\frac{(p_2 - p_1)x_1}{x_2 - x_1}\right) + p_1\right]$$

$\left(\frac{p_2 - p_1}{x_2 - x_1}\right)$=a(supply)=c(demand)

$\left(\frac{(p_2-p_1)x_1}{x_2-x_1}\right) + p_1$=b (supply)=d(demand).

$$p = ax + b \rightarrow Supply$$
$$p = cx + d \rightarrow Demand$$
$$x_{equilibrium} = \frac{b+d}{a+c}$$
$$p_{equilibrium} = ax_{equilibrium} + b$$

2.4. Inequality

To introduce the question of inequality in our model we followed two separate approaches. First, we defined methods with the goal of measuring inequality in the equilibrium state of our economy and secondly, we introduce additional variables that the user can change to see their impact on the inequality of the simulation. The measures we choose are the well-known gini coefficient and the s80/s20 measure.
The Gini coefficient is a measure of inequality of a distribution. It is defined as a ratio with values between 0 and 1: the numerator is the area between the Lorenz curve(see figure 0) of the distribution and the uniform distribution line, the denominator is the area under the uniform distribution line.

$$G = \left|1 - \sum_{k=1}^{n-1} (X_{k+1} - X_k)(Y_{k+1} + Y_k)\right|$$

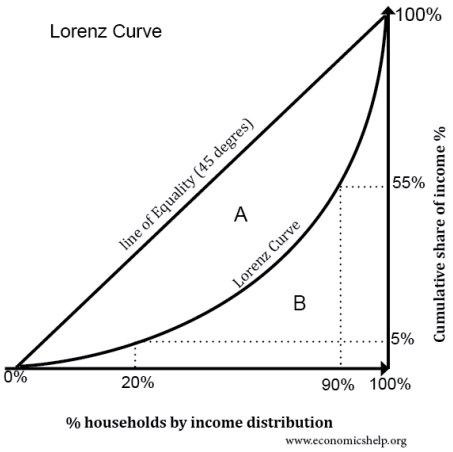

Figure 0:Lorenz Curve

The S80/S20 ratio measures the relative disparity in the distribution of a given order of magnitude. For an income distribution S80/S20 compares the mass of income held by 20% of the richest persons to that held by 20% of the poorest persons.

3. Implementation

The economy simulation is implemented in multiple classes to model an economy based on the Model-View-Controller (MVC) approach. The first package is called ‘Model’ and models the parts of our economy. This contains: a ‘Consumer’ class, modeling the consumer and solving the consumer problem, a ‘Producer’ class, modeling the producer and solving the producer problem and an ‘Economy’ class, modeling the economy and finding prices for which the producer and consumer problem result in a cleared market. The second package is the ‘Controller’ package containing a class called ‘EconomySimulation’, which contains the code carrying out the actual simulation. The third package is the ‘View’ package, which contains a class called ‘Userinput’. In ‘Userinput’ the way in which users of the Program can input

information is determined. For a more specific description on each class please refer to the following subsections.

3.1. Consumer Class

This class models a consumer of the economy. A consumer will be defined through their preferences and consumer behaviour. In order to be able to define consumer with different preferences, each consumer object will model one consumer and will be given values to define their utility function. In the following the instance variables and functions are described more closely (see *Figure 1*).

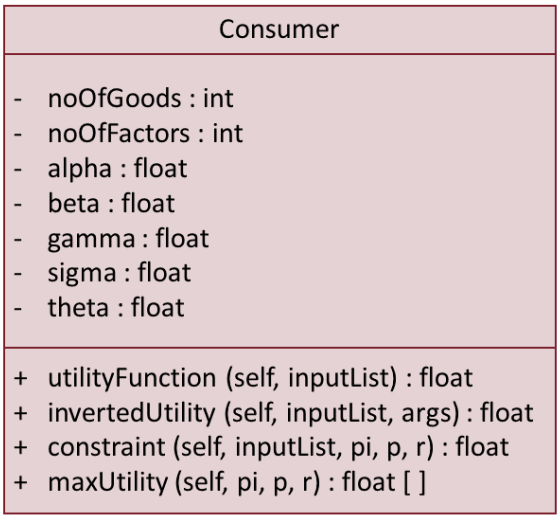

Figure 1: UML Class Diagram Consumer Class

3.1.1. Instance Variables

The instance variable noOfGoods and noOfFactors are used to pass on to the consumer object how many goods and factors in the economy exists. The remaining instance variables (alphy, beta, gamma, sigma, theta) are used to define the utility function of the consumer.

3.1.2. utilityFunction(self, inputList) : float

This instance function takes the parameters self and inputList. inputList is an array that has the length of number of goods plus number of factors. It will contain an allocation of goods and factors and will give back the amount of utility the consumer has by consuming this allocation. The return value which represents the utility for the specific allocation will be a float.

3.1.3. invertedUtility(self, inputList, args) : float

This instance function will simply give back the utility value for a specific allocation times (-1). So that instead of a maximization problem, a minimization problem can be solved using the Scipy minimize function.

3.1.4. constraint(self, inputList, pi, p, r) : float

The instance function constraints models the constraints of the consumer problem. Here the budget restriction is modelled. To satisfy the restriction the outcome of this function has to equal zero. The parametres given are the inputList, which represents a specific allocation of goods and factors, pi, which represents the amount of profit the consumer is allocated with and p and r which are the prices for goods and factors.

3.1.5. maxUtility(self, pi, p, r) : float [ ]

This instance function maximizes the utility of a consumer in dependency on pi, the profit they are allocated with, and p and r, the prices for goods and factors. The solution is returned in a float array, representing the optimal allocation of goods and factors for this specific consumer object. To maximize the utility the package 'scipy.optimize' is used.

3.2. Producer Class

This class models a producer of the economy. The producer has a function of production and tries to maximize the profit. The production function relates physical output of a production process to physical inputs or factors of production. It is a mathematical function that relates the maximum amount of output that can be obtained from a given number of inputs. The production function, therefore, describes a boundary or frontier representing the limit of output obtainable from each feasible combination of inputs. In the following the instance variables and functions are described more closely (see *figure 2*).

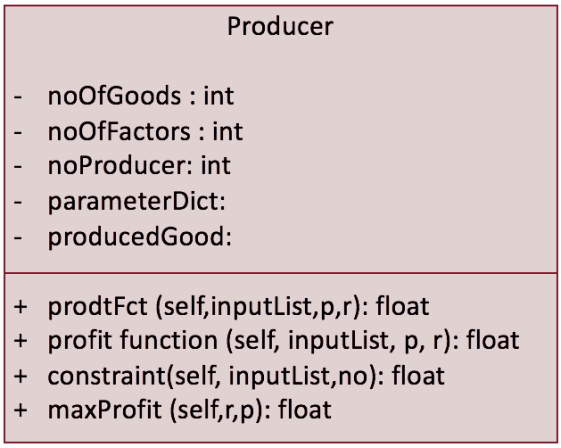

Figure 2: UML Class Diagram Producer Class

3.2.1. Instance Variables

The instance variable noOfGoods and noOfFactors are used to pass on to the consumer object how many goods and factors in the economy exist. We use a parameterDictonary that contains all parameters that define the production function:

$$F_i = \sum noOfgoods + X_i + P_{si}$$

3.2.2. Production function (self, inputList, p, r)

This instance function takes the parameters self and inputList, p and r inputList, is an array that has the length of number of goods plus number of factors. It will contain an allocation of goods and factors and will give back the amount of supply minus number of goods.

3.2.3. Profit function (self, inputList, p, r)

This instance function takes the parameters self, inputList, p and r. inputList is as an array that has the length of number of goods plus "p" and number of factors plus "r". After this function calculates total profit this function returns the value of the total profit.

Finally, we invert the total profit and this will simply give back the profit value for a specific allocation times (-1). So that instead of a maximization problem, a minimization problem can be solved using the Scipy minimize function.

3.2.4. Constraint(self, inputList,no):

The instance function constraints models the constraints of the producer problem and

3.2.5. Maximization profits (r,p)

This instance function maximizes the profit of the producers the profit they are allocated with, and p and r, the prices for goods and fact. The solution is returned in a float array, representing the optimal allocation of goods and factors for this specific producer object.

3.3. Economy Class

The object of the Economy class is to find good and factor prices that will make the sum of all the consumer demanded goods and the sum of all supplied factor amounts be the same as the sum of the producer's demanded amounts of factors and supplied goods respectively. The way chosen to do this is to subtract the difference and minimize it up to 0. Another way of saying this, is that this class does the work of the "invisible hand" in the market to adjust the prices in such a manner that there is no excess in supply or demand.
(see *figure 3*).

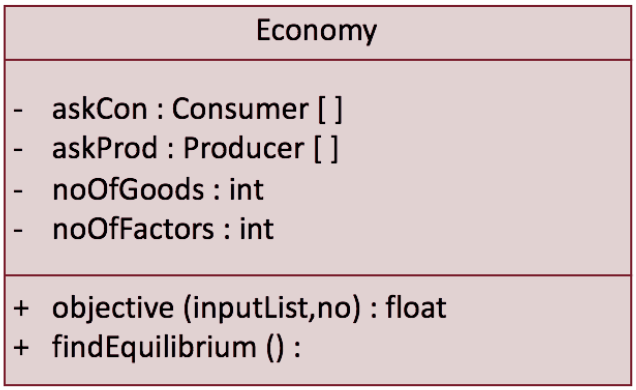

Figure 3: UML Class Diagram Economy Class

3.3.1. Instance Variables

For the objective to work it needs four variables. These are the "answer" of the Producers and Consumers when proposed good and factor prices. When asked, prices of goods and factor costs provided, the producer class gives an output list in form of an array that contains the amount of the good he is willing to provide first, and returns 0 in the places of the goods he is not producing, but still are part of the economy and the consumer decision. Then the amount of factors he demands at that factor price so that he will be able to produce, and at the end, a single profit amount he would make be the transaction successful.
The consumer class, similarly, gives an output list (array), that for given good and factor prices, and the response of the producer about profit, first containing the amount of every good he would consume, and then the amount of every factor he is willing to provide, in order increase his or her budget constraint. So if there are 3 goods produced in the economy, and 8 factors in total, and goods prices and factor prices are provided, and the producer produces the third good,

this class could return [0,0,34,3,4,5,6,7,8,7,0,100] in this format. The consumer, now taking profit (100) into account and good and factor prices, returns [4,5,6,6,7,1,2,3,4,5,6,7]. The consumer's list length is therefore always one item shorter than the producers. In order to be able to split these answers, we need to know the number of Goods and number of Factors.

3.3.2. getIncome(self, p,r)

This method takes any prices for goods and factors, ideally though the prices in which equilibrium is reached in the economy. The prices in which equilibrium is reached in the economy can be obtained by the user by using the findEquilibirum function of the class Economy. The method then returns an array of the income of each consumer of the economy.

3.3.3. calculateGini (self, valueArray)

This method calculates the gini coefficient for a given array. In our case the array of incomes should be given to this method, the gini coefficient will then be returned by the method.

3.3.4. s80_s20 (self, valueArray)

This method calculates the s80/s20 measure for a given array. In our case the array of incomes should be given to this method, the s80/s20 will then be returned by the method.

3.4. UserInput Class

The class UserInput contains all communication with the User. The class UserInput has two methods: introduction ( ) and inputValues ( ) (see Figure 4).

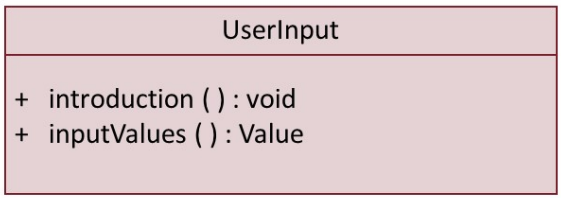

Figure 4: UML Class Diagram UserInput Class

3.4.1. Introduction ( ) : void

The method introduction contains no arguments and shall serve as introduction to the user. It will print out an explanation on the console, that introduces the functionalities and limitations of this economy simulation to the user and gives the user instructions on how to use this economy simulation. The method introduction has no return value.

3.4.2. InputValues ( ) : Value

The method inputValues will ask the user for all values necessary to create an economy object. These values will then be stored in a Value object and given back as return value. The values necessary to create an economy object include: an array of consumers, an array of producers, no of goods and number of factors in the economy. Hence, all information to create an array of consumers and an array of producers is also needed. Including the parameters to define utility and production function as well as which consumer gets profit from which producer and which good each producer produces.

3.5. Values Class

The Value class contains no methods, but only instance variables. It's sole purpos is to create a datatype in which we can store all information given by the user to create the economy in one variable. The instance varaibles of Value Class are noOfGoods, noOfFactors, noOfConsumers, noOfProducers, parameterProd, parameter Con, prodOfCon and goodProd (see Figure 5).

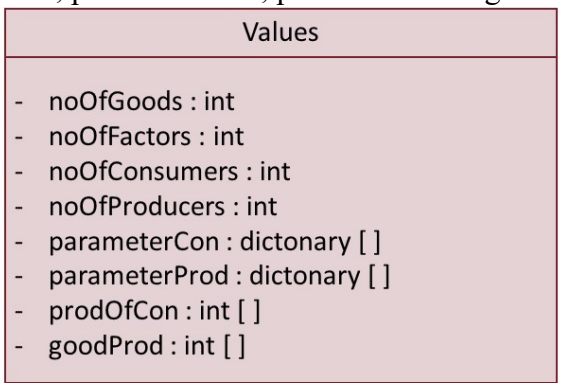

Figure 5: UML Class Diagram Values Class

3.5.1. Instance Variables

Instance Variable	Description
noOfGoods	Defines the number of goods in the economy. Datatype integer.
noOfFactors	Defines the number of factors in the economy. Datatype integer.
noOfConsumers	Defines the number of consumers in the economy. Datatype integer.
noOfProducers	Defines the number of producers in the economy. Datatype integer.
parameterCon	Defines the parameter for the utility function for each consumer. Datatype is a list of dictionaries, each dictionary contains the parameters: alpha, beta, gamma, sigma and theta.
parameterProd	Defines the parameter for the production function for each producer. Datatype is a list of dictionaries, each dictionary contains the parameters: xi and psi.
prodOfCon	Defines the relationship between consumers and producers, more specifically it defines for each consumer, which shares he has of a producer's company. This will later in the economy simulation impact the profit a consumer gets to spend. Datatype is a list of integers, the integer is the number of the producer from which the consumer will receive profit.
goodProd	Defines which producer produces which good. Datatype is a list of integers, the integer is the number which good will be produced by the indexed producer.

3.6. Economy Simulation Class

The EconomySimulation class uses all the previous code in order to run the economy simulation. Therefore, it has no instance variables nor methods (see Figure 5). In the EconomySimulation class the user will first be introduced to the economy simulation by creating a InputUser object and calling on the its introduction ( ) function. Then the user will be asked to give in values by using the inputValue ( ) function of the InputUser class. Afterwards the EconomySimulation class will create consumers, producers and economy objects according to the values given by the user. Lastly, the EconomySimulation class will run the economy simulation and give the user back the measures of inequality measured by the given instance of economy.

3.6.1. Objective(self, Inputlist)

3.6.2. Constraint(self, InputList,no)

3.6.3. FindEquilibrium(self)

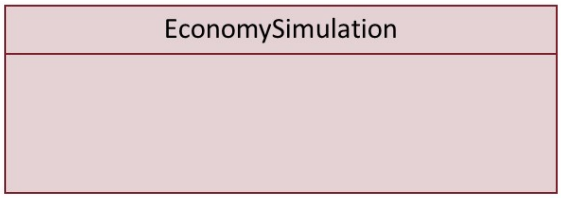

Figure *6*: UML Class Diagram EconomySimulation C

4. Future Outlook

In order to solve the problem of inequality in the economy we have different options:
A progressive tax it can be an option, a progressive tax is that tax by which the tax rate increases as the tax base increases, pursuing a redistributive effect of income or expenses.

The progressive tax tries to reduce the incidence of taxes that people with lower purchasing power must pay, it's often applied to personal income taxes, where people with higher incomes pay a higher percentage of those income in taxes than people who enjoy countless income.