國立臺灣大學
National Taiwan University

Economy Simulation: Inequality

Documentation

from

Ricard Milà 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:

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

$\overrightarrow{x_g}$: $vector\ of\ consumption\ of\ all\ goods$

$\overrightarrow{v_f}$: $vector\ of\ factor\ supply\ of\ all\ factors$

$u\left(\overrightarrow{x_g}, \overrightarrow{v_f}\right)$: $utility\ function\ in\ dependency\ of\ the\ factor\ and\ good\ allocation$

π : $share\ of\ profit\ for\ this\ particular\ consumer$

$\vec{p}$: $price\ vector\ of\ all\ goods$

$\overrightarrow{r}$: $price\ vector\ of\ all\ factors$

2.2. Producer Problem

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 'Economy-Simulation', 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*).

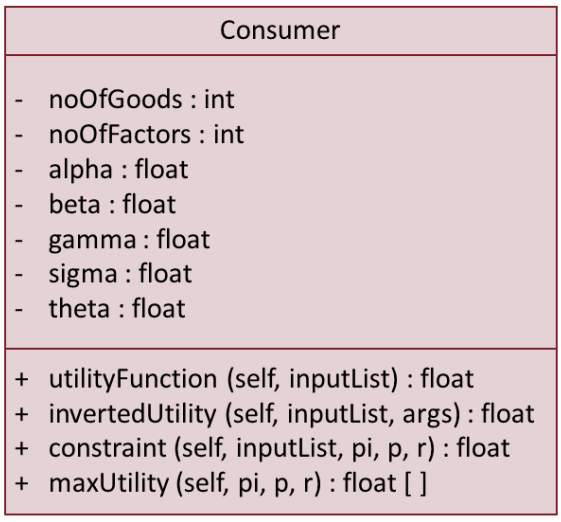

Figure 1: UML Class Diagram Consumer Class

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

To solve the producer problem we have to found the allocation of good and factors, which maximizes the producer profit.
With the production function, we obtain the number of goods that can be produced as a maximum with a certain amount of resources and with the minim cost.
Y = f(L,K)

- Production Technologies
  - Factor f used in the production of good g: rgf
  - Aggregate amount of good g produced: X^g • Production Function for good g:
  - $X^g = \varphi^g(rg1,...,rgF)$ or $X^g = \varphi^g(rgf)$

  - Production Fuction for good g:

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

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

max $\pi(\overrightarrow{x_g}, \overrightarrow{v_g})$

$x_g v f$

$s.t.\ \overrightarrow{x_g} f g = \emptyset g\ (r\ f g)$

$\overrightarrow{x_g}$:$vector\ of all\ goods\ g$

$\overrightarrow{v_g}$:$vector\ of factor supply of all factors\ f$

This class models a producer of the economy. A producer is someone who creates and supplies goods or services. Producers combine labor and capital—called factor inputs—to create—that is, to output—something else.

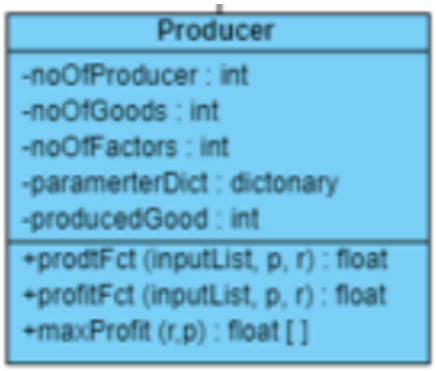

Figure 2 Producer Class Diagram

3.2.1. Production function (Self, inputList,p,r)

This instance function takes the parameters self inputList, p, r. The constraint will contain an allocation of goods and factors and will give back the amount of the total of good. In this case, we calculate the total production of one good.

3.2.2. Profit function (Self, inputList,p,r)

This function take the total of the production and define the total profit, considering the different variables Xhg, Vhg and the price for goods and factors.

3.2.3. Inverted profit function

This instance function 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. Maximization profits (r,p)

The profit-maximizing firm chooses both inputs and outputs so as to maximize the difference between total revenue and total cost. In this instance function maximizes the profit of a producer, the profit they are allocated with, and p and r, the prices for goods and factors.

3.3. Economy Class

3.4.

3.5. UserInput Class

The class UserInput contains all communication with the User. The class UserInput has two methods: introduction ( ) and inputValues ( ) (see Figure 2).

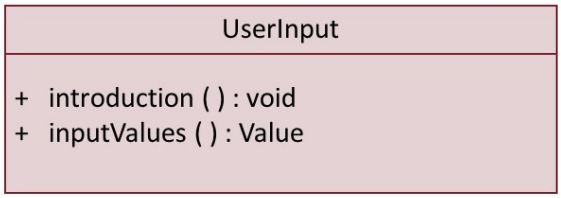

Figure 2: UML Class Diagram UserInput Class

3.5.1. 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.

3.5.2. 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.

3.6. 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 3).

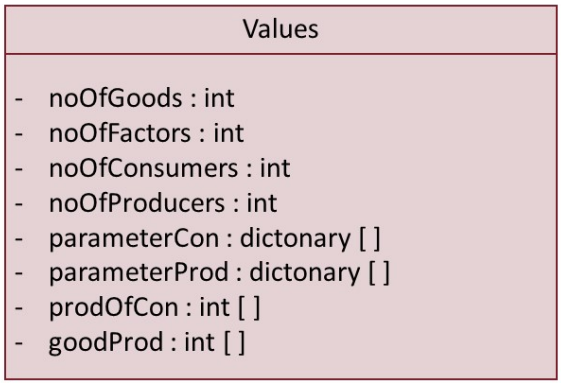

Figure 3 : UML Class Diagram Values Class

3.6.1. 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.

3.7. 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 4). 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.

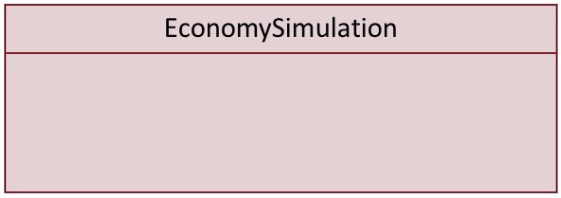

Figure 4 : UML Class Diagram EconomySimulation Class