國立臺灣大學
National Taiwan University

Economy Simulation: Inequality

Documentation

von

Michelle 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:

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

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

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

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

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

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

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

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).

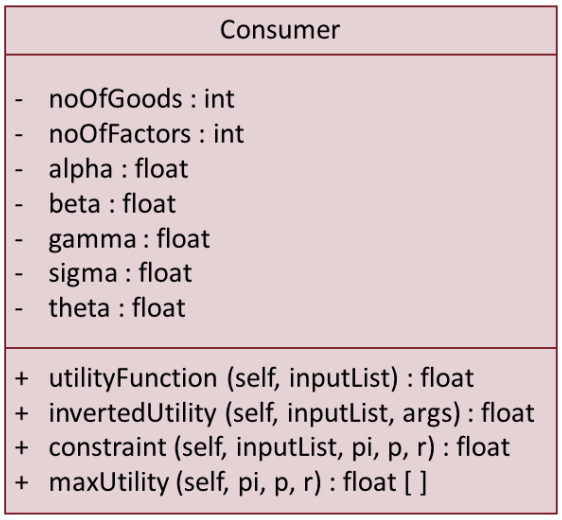

Figure 1 UML Class Diagram Consumer Class

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.