Computing the Arrow-Debreu Market Equilibrium

August 1, 2007

1 Introduction

We have a market with m agents and n goods. Agent i initially has amount b_{ij} of good j. Agent i achieves utility $a_{ij}x_{ij}$ when he is allocated amount x_{ij} of good j. Thus agent i's total utility is $\sum_j a_{ij}x_{ij}$ when he is allocated $(x_{i1}, \dots, x_{in})$.

A market equilibrium is a set of allocations of goods to agents x_{ij} together with a set of good prices $p \in \mathbf{R}^n$ such that at equilibrium

- all goods are sold at the equilibrium price.
- each agent maximizes his total utility.

It can be shown that a market equilibrium always exists and can be found by solving the following problem

$$\begin{array}{ll} \text{find} & x, \quad p \\ \text{subject to} & \sum_k a_{ik}x_{ik} \geq a_{ij} \sum_k b_{ik} \frac{p_k}{p_j}, \quad \forall\, i, j \\ & \sum_i x_{ij} = \sum_i b_{ij}, \quad \forall j \\ & x_{ij} \geq 0, \quad p_j \geq 0. \end{array} \tag{1}$$

Problem (1) is not a convex optimization problem but can be easily transformed to one by a simple change of variables. Specifically let $p_j = \exp(\psi_j)$. Then problem (1) becomes

$$\begin{array}{ll} \text{find} & x, \quad \psi \\ \text{subject to} & \sum_k a_{ik}x_{ik} \geq a_{ij} \sum_k b_{ik} e^{\psi_k - \psi_j}, \quad \forall\, i, j \\ & \sum_i x_{ij} = \sum_i b_{ij}, \quad \forall j \\ & x_{ij} \geq 0. \end{array} \tag{2}$$

This is a convex feasibility problem and can be solved in a variety of ways. In fact this is a mixed linear-GP.

2 A Trust Region Solution

In this section we describe a simple trust region style method for solving problem (2). This method proceeds in the following way: given an infeasible point (x, ψ), we linearize the first set of constraints in (2) about this point. We allow our next operating point to deviate by a small amount about the previous operating point (the so-called trust region) and update our point accordingly

We first start by reformulating (2) as the following optimization problem

$$
\begin{array}{ll}
\text{maximize} & \min t_i \\
\text{subject to} & t_i \leq \sum_k a_{ik} x_{ik} - a_{ij} \sum_k b_{ik} e^{\psi_k - \psi_j}, \quad \forall\, i, j \\
& \sum_i x_{ij} = \sum_i b_{ij}, \quad \forall j \\
& x_{ij} \geq 0,
\end{array}
\tag{3}
$$

where the optimization variables are x, t, and ψ. Note that this problem is equivalent to (2) since a market equilibrium always exists.

Now, given ψ, we linearize the first set of constraints of (3), which gives the following problem

$$
\begin{array}{ll}
\text{maximize} & \min t_i \\
\text{subject to} & t_i \leq \sum_k a_{ik} x_{ik} - a_{ij} \left(\sum_k b_{ik} e^{\psi_k - \psi_j} (1 + \Delta\psi_k - \Delta\psi_j) - b_{ij} \Delta\psi_j \right), \quad \forall\, i, j \\
& \sum_i x_{ij} = \sum_i b_{ij}, \quad \forall j \\
& \sum_j \Delta\psi_j = 0, \quad \|\Delta\psi\|_\infty \leq \Delta_{\max} \\
& x_{ij} \geq 0,
\end{array}
\tag{4}
$$

with variables x, t, and $\Delta\psi$. We put a constraint in the sum of $\Delta\psi_j$ since the ψ_j's are invariant to scalar shifting. The parameter $\Delta_{\max}$ controls the width of the trust region.

This is an LP and can be solved using an off-the-shelf solver. In fact if a and b are sparse, then we can easily write a custom LP solver that can deal with large instances of this problem.

Given a current operating point (x, ψ) we define the *agent residual* η as

$$
\eta = \min_{i,j} \left(\sum_k a_{ik} x_{ik} - a_{ij} \sum_k b_{ik} e^{\psi_k - \psi_j} \right).
$$

The trust region algorithm for solving problem (2) proceeds as follows

given tolerance $\epsilon > 0$, parameter $\Delta_{\max}$,
initialize: $\psi = 0$
while $\eta < -\epsilon$
  compute $(x, \Delta\psi)$ from (4)
  update:
    $\psi := \psi + \Delta\psi$

We generated a simple numerical example with $m = 30$ $n = 30$ and with a_{ij}, b_{ij} random, uniformly distributed between 0 and 1. We set $\Delta_{\max} = 0.1$ and $\epsilon = 10^{-8}$.

Figure 1 shows the agent residual versus algorithm iteration for an instance of this problem.

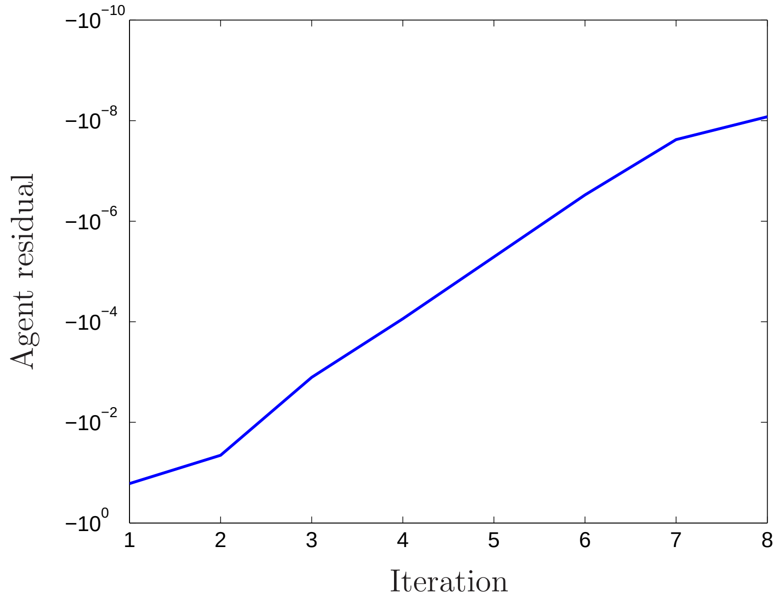

Figure 1: Agent residual versus trust region method iteration.

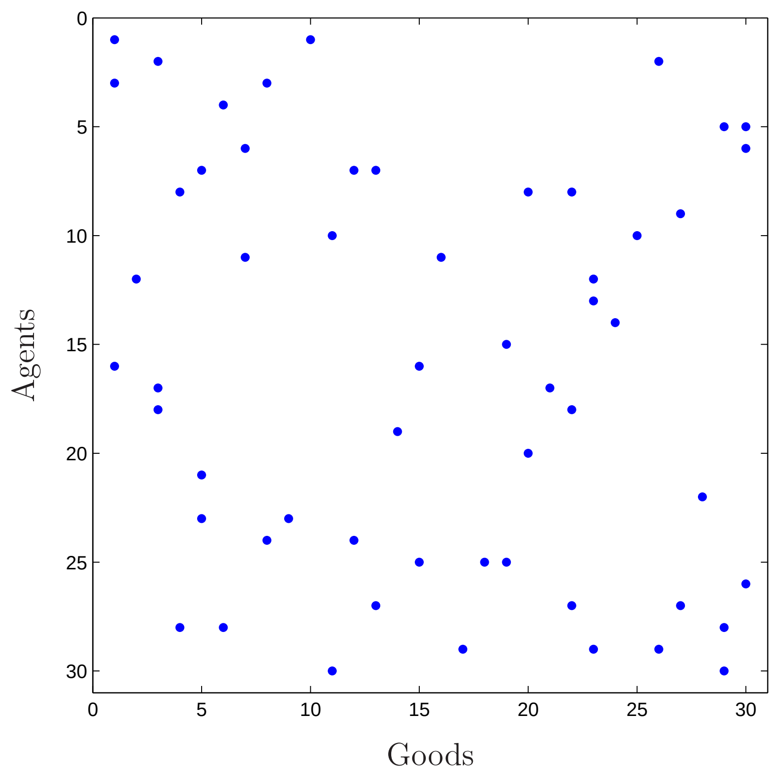

Figure 2: Sparsity pattern of final solution.

Figure 2 shows the sparsity pattern of the final solution x. Interestingly, only 56 out of 900 good assignments are nonzero.

XXX A word of caution: this method will oscillate if $\Delta_{\max}$ is too large. There are ways around this: we could expand $\exp(x)$ as a quadratic rather than just linearizing, or we could iteratively decrease the trust region width.