Solving a Model of Assortative Matching with Large Firms

Cristina Lafuente

Julia Faltermeier

David Pugh

Python

- The solver has been built using python™
- Some very useful libraries currently expanding, like [quantecon](#) – see the GitHub page of the project for more info! – make the code intuitive and easy to program.
- Much of the credit of the following goes to [David Pugh](#), who started this project and now keeps updating it with quantecon functions.

The Setup

- There are 3 basic classes of objects we use:
  - Input class: like workers or firms, it needs a distribution, boundaries, and name. It contains functions for symbolic and numeric pdf and cdf distributions.
  - Model class: Needs two inputs, a production function (symbolic), a set of parameters for the production function and a type of assortativity (it will be check while solving). It contains all cross-derivatives, expressions for the wages and profits, numeric and symbolic. It uses them and the cdfs from inputs to build the system of differential equations.
  - Solver class: Which branches out into Shooting or Collocation. It contains the code to carry out the integration of the system and store the results.

A Shooting Solver

- Intuitively, the way the solver works is the following:
- We know from the solution that if PAM holds, the highest type worker matches with the highest type firm. The only bit we need to guess is the size of this top firm: $\bar{x} - \bar{y} - \theta^{guess}$
- Once we have those three, we can integrate one step the system of differential equations:

$$\theta'(x) = \frac{H(x)F_{yl} - F_{xr}}{F_{lr}} \; ; \; \mu'(x, \theta(x)) = \frac{H(x)}{\theta(x)}$$

- Which will give us the next combination of $x - \mu(x, \theta(x)) - \theta(x)$
  - We do not need to integrate for the wages, as when we have both other two variables (θ and y) it can be calculated as F_l.

A Shooting Solver

- We keep doing this until one of three things happen:
  - We run out of workers! – feasibility condition failed → update guess and start again
  - We run out of firms! – market clearing failed (-wage) → update guess and start again
  - Last firm matches with last worker. All firms and workers matched with positive wages → success!

A Shooting Solver

- A faster way of solving this problem is to approximate the two set of differential equations with polynomials – orthogonal collocation.
- This method is faster because we can use simple, ready implemented functions to search for the best coefficients that satisfy the optimality, feasibility and market clearing conditions.
- The initial guess in this case would be the solution from the shooting solver of $\mu^*(x)$, $\theta^*(x)$
- Work in progress to be done this week – ask David for details.

Generating distributions

- The shooting solver delivers matched vectors of $x - \mu(x) - \theta(x) - w(x)$
- We don't know a priori the distribution of $\theta(x)$ or $w(x)$, but from the distribution of x we can do a simple change of variables to get the distribution of θ and w.
- Unfortunately, this requires to invert these functions (so we can calculate $H_w(x(\theta))$ for example), something that is complicated if the function $x(\theta)$ is non-monotonic.
- A possible solution to this is just calculate the moments of the distribution of θ and w, which do not requite to evaluate the pdf.

Objective function

- In the end, the idea is to get an objective function that contains three (or more) equations that measure the distance between what the model gives and what we observe in the data.
- For example, we can have [mean(θ), std(θ), mean(w), std(w)], or [distance of θ distributions, distance of w distributions, distance of π distributions].
- Once we have that, apply a simple(x) minimization routine to iterate in search for the best parameters.
- When collocation is ready to use, we only need to use shooting the first time round, and use the previous parameters results as new initial guesses.

To be continued!