**Basic Instructions**

To generate, for a given demand function and distribution, a plot of welfare gains from trade, distribution of sales, and distribution of markups, run MAIN.m. To generate welfare gains for all distributions, run WELFARE\_GAINS\_all\_distributions.m. Similarly, to generate a distribution of sales with all distributions plotted on one graph, run SALES\_PERCENTILES\_all\_distributions.m.

**Graphing welfare vs. tau (WELFARE\_GAINS.m)**

**Step 0: Declare parameters.**

* These are defined in PARAMETERS.m.
* The program then loads the distribution (using initDist) based on the value of the string distribution, along with calibrated parameters.

**Step 1: Set the tau range and step size.**

* intervalSize is the multiple of tauInit between successive points on the graph.
* stepsBelow and stepsAbove are the number of steps below and above tauInit that we’ll be graphing (so the tau range is tauInit \* (1 – stepsBelow \* intervalSize, 1 + stepsAbove \* intervalSize)).
  + These are set so that the tau range is roughly 20% above and below tauInit.

**Step 2: Compute the equilibrium for each tau.**

* Loop through every tau and compute the equilibrium, along with the statistics. (See the explanation for solveEquilibrium.m.)

**Step 3: Compute expenditures.**

* Compute the baseline utility (at tauInit).
* Use this utility to compute the expenditures at all taus.

**Step 4: Compute welfare.**

* For **exact welfare gains**, simply divide the baseline expenditure by the expenditure at each tau.
* Compute elasticities over each interval. Use these to compute fractional changes in **welfare ACR** between successive points.
* Use elasticities and rho at baseline to compute changes in **welfare ACDR** between successive points.
* To convert differential changes in welfare ACR and ACDR to total changes, multiple these successive fractional changes, in each direction, starting from the baseline (so that welfare gains are always 1 at the baseline).

**Step 5: Graph the three welfare functions, along with errors.**

**Solving for the two-country symmetric equilibrium (solveEquilibrium.m)**

**Inputs:** tau, p\_star\_init

**Outputs:** equilibrium, statistics

**Step 0: Initialize parameters.** (Use global.)

**Step 1: Run loop to solve equilibrium.**

* Guess initial p\_star (use p\_star\_init).
* Use computePrices to find the price functions. computePrices solves for p\_d and p\_x at a vector of *z* values (z\_vals) at the given p\_star, then interpolates to give continuous functions p\_d(z) and p\_x(z).
* Solve for the two productivity cutoffs using the profit function (and fsolve).
* Integrate to compute the total domestic profits and the total export profits.
* Compute the N implied by the p\_star, z\_d, and z\_x that we have. We now have the four values that are needed to define equilibrium, but they might not satisfy the last equation.
* Integrate to compute the total domestic sales and the total export sales.
* Compute the error based on the last equation. Positive error means that our guess for p\_star was too high, so we adjust downward.
* Repeat until the absolute value of the error is smaller than the tolerance.

**Step 2: Check for valid cutoffs.** If the productivity cutoff is smaller than the lower bound for *z*, the result isn’t valid, since the functions aren’t correctly defined in that region. The program gives an error message.

**Step 3: Store equilibrium values in structure.**

**Step 4: Compute basic statistics.** Store in structure.

More detailed explanations of how each script/function works is provided by in-line comments.