Lydia Teinfalt, 3/25/2025, HW6: Low-level Noise Added to Edgeworth Box Code simulating the real-world unpredictability of bilateral exchange between two agents and 2 goods
specs: Python 3x on Google Collab and Spyder

Background from Homework 5

Consumer (agents) class is instantiated with Cobb-Douglas preferences with α randomly chosen from the list of values = 0.25, 0.33, 0.5 and β = 1 $-\alpha.$ Consumer's initial endowment x_1 is randomly initialized with possible values from 1 to K and x_2 with possible values 1 to L. The parameters, K and L, are global variables representing the dimensions of the Edgeworth box and maximum amounts of goods 1 and goods 2 that can be traded between consumer 1 and consumer 2.

The Population class is instantiated with N agents. The model selects two consumers randomly and determines if a trade is possible. Trade is possible if given the two selected consumers do not have the equal marginal rates of substitutions (MRS) (Foundations of ABM, p. 67)

Cobb-Douglas preferences (see appendix G), i.e.,

$$U^i(x_1^i, x_2^i) = (x_1^i)^{\alpha^i}(x_2^i)^{1-\alpha^i}$$

From this we can compute the marginal rates of substitution:

$$MRS_{12}^i(\boldsymbol{x}^i) = \frac{\frac{\partial U^i(\boldsymbol{x}^i)}{\partial x_1^i}}{\frac{\partial U^i(\boldsymbol{x}^i)}{\partial x_2^i}} = \frac{\alpha^i x_2^i}{(1-\alpha^i)x_1^i}.$$

If exchange is possible, the trade is executed by randomly selected a new $x_1$between consumer 1's x_1 initial endowment and consumer 2's x_1 initial endowment. A new x_2 is derived from a contract curve between the two consumers based on the following formula (1) (Foundations of ABM, p. 76)

Solving this for x_2^i as a function of x_1^i gives

$$x_2^i = \frac{\alpha^j(1-\alpha^i)x_1^i x_2^T}{\alpha^i x_1^T(1-\alpha^j) - x_1^i(\alpha^i - \alpha^j)} \tag{1}$$

Code Introducing Low-Level Noise HW6

```
# Function to execute trade between two consumers by picking a value from the contract curve
# Low level noise added to simulate real-world unpredictability controlled by noise_level
# If the randomly generated value is above or equal to the noise level which is set by default to 0.5,
# then randomly select a y_new value between consumer 1 and 2's endowment2 values instead of a
# value on the contract curve
  def execute_trade(self, consumer1, consumer2,  noise_level=0.5):

    # Pick a random value of x within the range of their endowments
    x_new = random.uniform(consumer1.endowment1, consumer2.endowment1)

    #total endowments of consumers 1 and 2
    x_total = consumer1.endowment1 + consumer2.endowment1
    y_total = consumer1.endowment2 + consumer2.endowment2

    # generate random x between 0 and 1
    # introducing low level of noise into the process
```

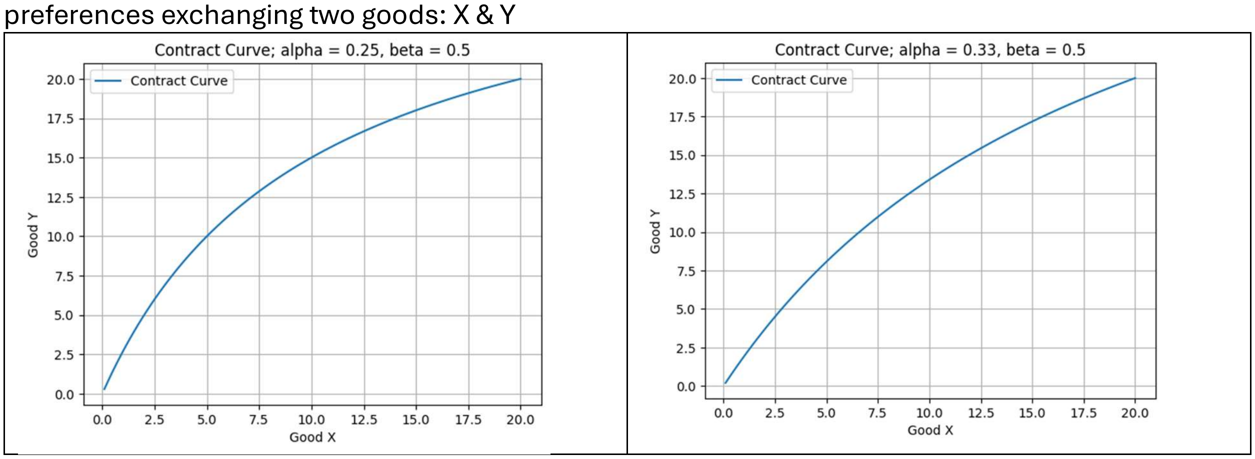
```
    # to simulate real-world unpredictability in trades
    randx = random.rand()
    if randx >= noise_level:
      #print("Random 'noise' variable meets criteria, execute trades",randx)
      y_new= random.uniform(consumer1.endowment2, consumer2.endowment2)
    else:
      # Calculate the corresponding y value on the contract curve
      y_new = (consumer1.beta*(1-
consumer1.alpha)*x_new*y_total)/(consumer1.alpha*x_total*(1-consumer1.beta)-
(x_new*(consumer1.alpha - consumer1.beta)))
```

Results HW 6

Simulation creates a population of N consumers trading two goods and have agents trade until the population reaches Pareto Optimality or the maximum number 1000 trades. Simulation re-run for 45 runs. Columns D and F display the new results of the run with low-level noise code introduced. With 10 agents, the new code shows a lower percentage of experiment runs (out of 45) where Pareto Optimality solution is found and at a lower number of trades required. At 20 agents, the new code resulted in more runs reaching pareto optimality than without the noise but at a cost of a greater average number of trades. For 50 agents, pareto optimality declines dramatically to only 4.44% and a steep rise in cost for number of trades. At 70 and 100 agents, pareto optimality is never reached before the 1000 maximum number of trades is reached first.

A	B	C	D	E	F
Number of Agents	**Number of Runs**	**% Runs with Pareto Optimal Solution**	**With Noise % Run Pareto Optimal Solution**	**Avg Number of Trades**	**With Noise Avg Number of Trades**
10	45	31%	24.44%	691.02	652.04
20	45	27%	35.56%	737.18	792.73
50	45	29%	4.44%	721.55	996.44
70	45	24%	0.00%	768.0	1000.00
100	45	40%	0.00%	629.36	1000.00

Sample contract curves between two agents with different alpha and beta levels Cobb-Douglas preferences exchanging two goods: X & Y



Code repository

https://github.com/lydiateinfalt/CSS610-AgentBasedModelingSimulation-Spring2025/blob/main/EdgeworthBoxNoise.py