ECON 714. Quant Macro-Econ Theory

Homework I

Ji Hwan Kim *

November 1, 2021

1 Github

My Github Repo address is `https://github.com/jihwankim94/ECON714.git`.

2 Integration

I compute

$$\int_0^T e^{-\rho t} u(1 - e^{-\lambda t}) dt$$

for $T = 100$, $\rho = 0.04$, $\lambda = 0.02$, and $u(c) = -e^{-c}$ using quadrature (Midpoint, Trapezoid, and Simpson rule) and a Monte Carlo. I constructed 10^1, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, 10^8 grids to obtain the integral value. Table 1 and 2 show the results and there are two points I want to comment on.

1. Midpoint, Trapezoid, and Simpson rules yield -18.2095254 for the large number of grids, while Monte Carlo does not. One thing to note is that Simpson rule already gives -18.2095254 for $N = 10^3$. This is because it has small errors relative to Midpoint and Trapezoid rules. To sum up, Simpson rule is the most accurate, while Monte Carlo is the least accurate.

*University of Pennsylvania. Email: jhkim94@sas.upenn.edu.

2. Simpson rule has the slowest running time for the large number of grids. The reason is that Simpson rule requires a lot of information. Midpoint rule is faster than Trapezoid rule and Trapezoid rule is faster than Simpson rule for the same reason. Monte Carlo has the fastest running time.

Table 1: Integral Value using Quadrature and a Monte Carlo

	Integral Value			
N	Midpoint	Trapezoid	Simpson	Monte Carlo
10^1	-17.96441999	-18.70274754	-18.21052918	-30.15572733
10^2	-18.20703949	-18.21449754	-18.2095255	-20.06409763
10^3	-18.20950054	-18.20957513	-18.2095254	-18.3412443
10^4	-18.20952515	-18.2095259	-18.2095254	-18.28211574
10^5	-18.2095254	-18.2095254	-18.2095254	-18.32942212
10^6	-18.2095254	-18.2095254	-18.2095254	-18.20563374
10^7	-18.2095254	-18.2095254	-18.2095254	-18.21465679
10^8	-18.2095254	-18.2095254	-18.2095254	-18.20790829

Table 2: Computing Time

	Computation Time (sec.)			
N	Midpoint	Trapezoid	Simpson	Monte Carlo
10^1	0.002398645	0.000774171	0.000892866	0.044984383
10^2	0.000168065	0.000052532	0.000063017	0.001963316
10^3	0.00020665	0.000314757	0.000449459	0.003211492
10^4	0.001556	0.00285381	0.003755782	0.002408901
10^5	0.010972782	0.021583188	0.028453577	0.012854599
10^6	0.09312751	0.164548436	0.237195491	0.015075394
10^7	0.879312823	1.775283869	2.598623999	0.155163995
10^8	8.867145365	17.10344608	24.88859687	1.5087364

3 Optimization: basic problem

I use the Newton-Raphson, BFGS, steepest descent, and conjugate gradient method to solve:

$$\min_{x,y} 100(y-x^2)^2 + (1-x)^2$$

where $f(x,y) = 100(y-x^2)^2 + (1-x)^2$ is the Rosenbrock function. It is easy to see that f is minimized at $x = y = 1$ and f has a minimum value 0 at that coordinate. Table 3 and 4 show the results and there are two points I want to comment on.

1. All methods yield the same results. As I expected, f has a minimum value 0 at $x = y = 1$.

2. Conjugate descent method has faster running time than steepest descent method. BFGS has the fastest running time.

Table 3: Solutions

	Solutions			
	Newton-Raphson	BFGS	Steepest Descent	Conjugate Descent
x	1	1.000000017	1.000010415	0.99998986
y	1	1.000000033	1.00002088	0.999979673
f	0	5.83174e-16	1.08724e-10	1.03043e-10

Table 4: Computing Time

Computation Time (sec.)			
Newton-Raphson	BFGS	Steepest Descent	Conjugate Descent
0.009062364	0.008808875	0.280758814	0.06094677

4 Computing Pareto efficient allocations [1]

Consider the following social planner problem.

$$\max_{x_j^i \geq 0} \sum_{j=1}^{n} \lambda_j u_j(x) = \max_{x_j^i \geq 0} \sum_{j=1}^{n} \lambda_j \left(\sum_{i=1}^{m} \alpha_j \frac{{x_j^i}^{1+w_j^i}}{1+w_j^i} \right)$$
$$\text{subject to } \sum_{j=1}^{n} x_j^i = \sum_{j=1}^{n} e_j^i \quad \forall i$$

for Pareto weights $\lambda = (\lambda_1, ..., \lambda_n) > 0$.

The first order conditions are

$$\lambda_1 \alpha_1 {x_1^i}^{w_1^i} = \mu_i$$
$$\lambda_j \alpha_j {x_j^i}^{w_j^i} = \mu_i$$

Combining yields

[1] There are two different folders named $m = n = 3$ and $m = n = 10$. They contain the same matlab codes though.

$$x^i_j = \left(\frac{\alpha_1\lambda_1}{\alpha_j\lambda_j}\right)^{\frac{1}{w^i_j}} {x^i_1}^{\frac{w^i_1}{w^i_j}}$$

I use the resource constraint

$$\sum_{j=1}^{n} e^i_j = \sum_{j=1}^{n} x^i_j$$

$$\Rightarrow \sum_{j=1}^{n} e^i_j = x^i_1 + \sum_{j=2}^{n} \left(\frac{\alpha_1\lambda_1}{\alpha_j\lambda_j} {x^i_1}^{w^i_1}\right)^{\frac{1}{w^i_j}}$$

I solve the resulting system of nonlinear equations to solve for the social planner's problem.

4.1 $m = n = 3$

I compute first the case where all the agents have the same parameters and social weights. To be specific, I set $\alpha_j = 1$ for all j, $\lambda_j = \frac{1}{n} = \frac{1}{3}$ for all j, $w^i_j = -2$ for all j and i, and $e^i_j = 1$ for all j and i. All agents consume 1 unit of each good. In other words, they consume what they are endowed with.

Then I add a fair degree of heterogeneity.[2]

Case 1. I consider heterogeneity in endowment. I set

$$e = \begin{bmatrix} e^1_1 & e^2_1 & e^3_1 \\ e^1_2 & e^2_2 & e^3_2 \\ e^1_3 & e^2_3 & e^3_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$$

All consumers consume the same amount of all goods since there is no heterogeneity in utility functions and the total endowments.

$$x = \begin{bmatrix} x^1_1 & x^2_1 & x^3_1 \\ x^1_2 & x^2_2 & x^3_2 \\ x^1_3 & x^2_3 & x^3_3 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

What if we add heterogeneity in social weights? I set

[2] The values of some parameters do not change unless otherwise noted for each case below.

$$\lambda = \begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

Agents with higher λ_j consumes more goods than agents with lower λ_j. Each agent consumes the same amount of all goods since the total endowments of all goods are 6.

$$x = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 \\ x_2^1 & x_2^2 & x_2^3 \\ x_3^1 & x_3^2 & x_3^3 \end{bmatrix} = \begin{bmatrix} 1.4471 & 1.4471 & 1.4471 \\ 2.0465 & 2.0465 & 2.0465 \\ 2.5064 & 2.5064 & 2.5064 \end{bmatrix}$$

Case 2. I consider heterogeneity in α_js. I set

$$\alpha = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

Agents with higher α_j consumes more than agents with lower α_j. Each agent consumes the same amount of all goods since the total endowments of all goods are 3.

$$x = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 \\ x_2^1 & x_2^2 & x_2^3 \\ x_3^1 & x_3^2 & x_3^3 \end{bmatrix} = \begin{bmatrix} 0.7235 & 0.7235 & 0.7235 \\ 1.0232 & 1.0232 & 1.0232 \\ 1.2532 & 1.2532 & 1.2532 \end{bmatrix}$$

I can observe the same result if I add heterogeneity in α_j instead of heterogeneity in social weights in case 1. This is because I can consider both α_js and λ_js as the coefficients of the CRRA utilities in the objective function.

Case 3. I randomly assigned endowments. Each element in e is randomly drawn from the standard uniform distribution.

$$e = \begin{bmatrix} e_1^1 & e_1^2 & e_1^3 \\ e_2^1 & e_2^2 & e_2^3 \\ e_3^1 & e_3^2 & e_3^3 \end{bmatrix} = \begin{bmatrix} 0.7922 & 0.0357 & 0.6787 \\ 0.9595 & 0.8491 & 0.7577 \\ 0.6557 & 0.9340 & 0.7431 \end{bmatrix}$$

All agents consume the same amount of each good since utility functions and social weights are the same.

$$x = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 \\ x_2^1 & x_2^2 & x_2^3 \\ x_3^1 & x_3^2 & x_3^3 \end{bmatrix} = \begin{bmatrix} 0.8025 & 0.6063 & 0.7265 \\ 0.8025 & 0.6063 & 0.7265 \\ 0.8025 & 0.6063 & 0.7265 \end{bmatrix}$$

4.2 $m = n = 10$

Results and implications are identical to the case $m = n = 3$ except for the computation time. Computing Pareto efficient allocations for $m = n = 10$ needs more time. For example, it takes 0.154 seconds to solve for allocations in the case $m = n = 10$, while it only takes 0.096 seconds to solve for allocations in the case $m = n = 3$.

5 Computing Equilibrium allocations

Given the prices, household j solves

$$\max_{x_j^i \geq 0} u_j(x) = \max_{x_j^i \geq 0} \sum_{i=1}^{m} \alpha_j \frac{{x_j^i}^{1+w_j^i}}{1+w_j^i}$$
$$\text{subject to } \sum_{i=1}^{m} p^i x_j^i \leq \sum_{i=1}^{m} p^i e_j^i$$

The first order necessary conditions are

$$\alpha_j {x_j^1}^{w_j^1} = \mu_j p^1$$
$$\alpha_j {x_j^i}^{w_j^i} = \mu_j p^i$$

and hence

$${x_j^1}^{w_j^1} p^i = {x_j^i}^{w_j^i} p^1 \qquad (1)$$

I use goods market clearing conditions

$$\sum_{j=1}^{n} e_j^i = \sum_{j=1}^{n} x_j^i$$
$$\Rightarrow \sum_{j=1}^{n} e_j^i = \sum_{j=1}^{n} \left(\frac{p^i}{p^1}\right)^{\frac{1}{w_j^i}} x_j^{1\,\frac{w_j^1}{w_j^i}}$$

to solve for Arrow-Debreu equilibrium prices. I normalize $p^1 = 1$. First Welfare Theorem implies that Pareto efficient allocations I obtained in Question 4 and Arrow-Debreu equilibrium allocations are identical. Therefore, I put Pareto efficient allocations into goods market clearing conditions to find the equilibrium prices p^i.

5.1 $m = n = 3$

I compute first the case where there is no degree of heterogeneity. All prices are equal to 1 as expected.

Then I consider the cases where there is a fair degree of heterogeneity.

Case 1. All equilibrium prices are 1 as in the case where there is no degree of heterogeneity. This is because I only change the parameters in the utility functions, while the total endowments of each good remaining unchanged.

Case 2. All equilibrium prices are 1 for the same reason as in Case 1.

Case 3. Equilibrium prices are inversely related to the total endowments. The total endowments of each good are respectively 2.4074, 1.8188, and 2.1796.

$$p = \begin{bmatrix} 1 & 1.7520 & 1.2200 \end{bmatrix}$$

5.2 $m = n = 10$

Results and implications are the same as above but for the computation time.