1 Github

https://github.com/mariiaelk/ECON-8210-Computational-Methods-in-Economics-HW-1.git

2 Integration

Code which produces results for this section: Q2.m

Tables 1 and 2 provide comparison of 4 methods. We consider different number of points when approximating the function, ranging from 10 to 5000 (denote the number of intervals as N).

	N = 10	N = 100	N = 1000	N = 5000
Midpoint	-17.964	-18.207	-18.21	-18.21
Trapezoid	-18.703	-18.214	-18.21	-18.21
Simpson rule	-18.211	-18.21	-18.21	-18.21
Monte Carlo	-9.407	-14.508	-17.882	-18.671

Table 1: Computed values of the integral for different methods (N indicates number of points or intervals used when approximating the function)

	N = 10	N = 100	N = 1000	N = 5000
Midpoint	0.001	0.005	0.015	0.061
Trapezoid	0.001	0.009	0.026	0.12
Simpson rule	0.001	0.008	0.039	0.177
Monte Carlo	0.008	0.004	0.004	0.005

Table 2: Comparison of speed of different methods, measured in seconds (N indicates number of points or intervals used when approximating the function)

Midpoint, Trapezoid and Simpson rule give the same result of -18.21 (accurate to the third decimal place) starting from N equal to 1000. Simpson rule gives us almost the same value with only 10 intervals used for approximation, indicating that it is very accurate in the example we consider. Midpoint and Trapezoid also produce almost exactly -18.21 with only 100 intervals. If we use only 10 intervals, then they produce an error of 1% and 3% correspondingly. Overall, all three methods give rather accurate results even when we consider a relatively small number of intervals.

Monte Carlo produces much less accurate results. When $N = 10$ it produces an error of almost 50%. Increasing N to 1000 allows us to reduce the error to 2%. But even for $N = 5000$ we still see some discrepancy between -18.21 and Monte Carlo results. So, we could conclude that Monte Carlo produces least accurate results among all methods we consider.

In terms of speed, Simpson rule takes only about 0.001 second to compute the integral for $N = 10$. Midpoint and Trapezoid take less than that, namely, about 0.0006 second. As N increases, the amount of execution increases and the increase is the most substantial for Simpson rule. However, notice that Simpson rule gives accurate results with lower values of N, so there is actually no need to increase N that much when using Simpson rule.

Comparing Monte Carlo and Simpson rule, Monte Carlo takes 8 times more time to compute the integral when $N = 10$. Shifting to $N = 100$ actually reduces the time of execution for Monte Carlo due to Matlab optimization routines. As N increases, the time of execution for Monte Carlo barely changes. For the same N Monte Carlo is superior in terms of speed, however, it comes at a cost in terms of accuracy. Overall, it is more reasonable to compare Monte Carlo with $N = 1000$ with other methods executed using $N = 10$. In such a comparison Monte Carlo does not have any advantage over other methods for our specific example.

3 Optimization: basic problem

Code which produces results for this section: Q3.m

The code allows to compare performance of Newton-Raphson, BFGS, steepest descent and conjugate descent methods when minimizing the objective. The code allows to consider multiple initial guesses, and chooses the lowest attained value across all initial guesses considered. Before executing each method, we obtain gradient and Hessian of the objective as functions using Matlab Symbolic Toolbox. Then these functions are used within the algorithm for each method we consider. For steepest and conjugate descent initial value of the step size is set to be 0.1, but then we solve the following problem at each iteration to obtain the step size α^k*:*

$$\alpha^k = \arg\min_{\alpha} f\left(x^{(k)} + \alpha d^{(k)}\right)$$

where $x^{(k)}$ *is the candidate point and* $d^{(k)}$ *is the direction of the step. To solve this problem, we use Newton-Raphson algorithm. If we do not introduce this step of fining the optimal size step, then steepest and conjugate descent are often unstable (do not converge).*

We consider 7 alternative initial guesses: $(0,0)$, $(0.9,0.9)$, $(1,0)$, $(0,1)$, $(3,3)$, $(5,5)$, $(-5,-3)$ to test the performance of the algorithm. All four algorithms succeed in finding the minimizer $(1,1)$, but are different in terms of stability and speed. Table 3 summarizes how much time it takes the algorithm to run and go over all of the initial guesses (the maximum number of iterations for each guess is set to equal 5000).

	Speed (seconds)
Newton-Raphson	0.027
BFGS	0.002
Steepest descent	0.078
Conjugate descent	0.012

Table 3: Comparison of speed of different methods, measured in seconds

Newton-Raphson algorithm needs the least amount of iterations to converge: the maximum amount of iterations per point is 6, the minimum is 2 (when $(0.9,0.9)$ is used as a guess) and the average is 5. BFGS needs more iterations to converge: the maximum is 502, the minimum is 2 and the average is about 130. Despite higher number of iterations, this algorithm runs very fast: the execution time is 0.002 seconds.

Steepest descent is the slowest and the least stable of all. It converges only for two out of

7 initial guesses, namely, $(0.9, 0.9)$ and $(1, 0)$. In the first case it needs 116 iterations and in the second case it needs 162 iterations. For other initial guesses, algorithm does not converge. So, high execution time is actually the consequence of the fact that the algorithm fails to converge when the maximum number of iterations for each guess is set to equal 5000.

Conjugate descent converges for all 7 guesses. The minimum number of iterations is 204, the maximum number of iterations is 1574 and the average number of iterations is about 540. The execution time is lower than for Newton-Raphson.

Overall, BFGS is superior in terms of speed, and also converges for all of the initial guess we considered.

4 Computing Pareto efficient allocations

Code which produces results for this section: Q4.m. It uses fsolve(.) as a procedure to solve the system of non-linear equations.

First, let's obtain first order conditions of the Pareto problem.

Let j index agents and i index goods. Utility function of agent j:

$$u^j\left(x_j\right) = \sum_{i=1}^{m} \alpha_i \frac{\left(x_j^i\right)^{1+\omega_j^i}}{1+\omega_j^i}$$

Social planner problem:

$$\begin{aligned} &\max_{\{x_j^i\}_{i,j}} \sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_j \alpha_i \frac{\left(x_j^i\right)^{1+\omega_j^i}}{1+\omega_j^i} \\ \text{s.t.} \quad &\sum_{j=1}^{n} x_j^i = \sum_{j=1}^{n} e_j^i \forall i = 1, ..., m \end{aligned}$$

Lagrangian of this problem:

$$\mathcal{L} = \sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_j \alpha_i \frac{\left(x_j^i\right)^{1+\omega_j^i}}{1+\omega_j^i} + \eta_i \sum_{j=1}^{n}\left(e_j^i - x_j^i\right)$$

FOCs:

$$
\begin{aligned}
[x_j^i] : & \lambda_j \alpha_i \left(x_j^i\right)^{\omega_j^i} = \eta_i \\
\Rightarrow & \lambda_j \alpha_i \left(x_j^i\right)^{\omega_j^i} = \lambda_1 \alpha_i \left(x_1^i\right)^{\omega_1^i} \\
\Rightarrow & \frac{\left(x_j^i\right)^{\omega_j^i}}{\left(x_1^i\right)^{\omega_1^i}} = \frac{\lambda_1}{\lambda_j} \forall j = 2, ..., n, \forall i = 1, ..., m \\
[RC] : & \sum_{j=1}^{n} x_j^i = \sum_{j=1}^{n} e_j^i \quad \forall i = 1, ..., m
\end{aligned}
$$

Overall, we get a system of $m \times n$ equations in $m \times n$ unknowns.

We can reduce the dimension of the problem. Notice that

$$
\begin{aligned}
x_j^i &= \left[\frac{\lambda_1}{\lambda_j}\right]^{\frac{1}{\omega_j^i}} \left(x_1^i\right)^{\frac{\omega_1^i}{\omega_j^i}} \\
\Rightarrow \sum_{j=1}^{n} \left[\frac{\lambda_1}{\lambda_j}\right]^{\frac{1}{\omega_j^i}} \left(x_1^i\right)^{\frac{\omega_1^i}{\omega_j^i}} &= \sum_{j=1}^{n} e_j^i \quad \forall i = 1, ..., m
\end{aligned}
$$

Then we have a system of m equations in m unknowns. Notice that $\{\alpha_i\}$ do not affect the results. After obtaining solution for agent 1 $\left\{x_1^i\right\}_{i=1,...,m}$ we can recover consumption of all other agents as well, i.e. we can get the optimal allocation.

Now, we are going to solve this system of equations for 3 cases:

- $m = n = 3$, no heterogeneity in terms of preferences or Pareto weights
- $m = n = 3$, heterogeneity in terms of preferences and Pareto weights
- $m = n = 10$, heterogeneity in terms of preferences and Pareto weights

When there is no heterogeneity, it takes 0.044 seconds for the algorithm to converge. Optimal allocation implies that aggregate endowment is equally divided across agents (initial guess implies that agent 1 consumes all of the aggregate endowment). If we add a fair amount of heterogeneity in terms of preferences and Pareto weights, then execution time increases to 0.05. The number of iterations we need for convergence increases from 1 to 8. Now aggregate endowment is distributed across agents in a non-trivial manner.

When we consider a system with 10 agents, 10 goods and fair amount of heterogeneity, execution time increases to 0.073 seconds. Now the algorithm needs 11 iterations to converge. Still, we are able to solve this system of 10 equations. fsolve function with Trust-Region algorithm chosen as the algorithm to be used handles this problem well. Importantly, we were able to reduce the original system of $n \times m$ equations to just m equations using special properties of Pareto problem.

5 Computing Equilibrium allocations

Code which produces results for this section: Q5.m. It uses fsolve(.) as a procedure to solve the system of non-linear equations.

Problem of agent j:

$$\max_{\{x_j^i\}_i} \sum_{i=1}^{m} \alpha_i \frac{\left(x_j^i\right)^{1+\omega_j^i}}{1+\omega_j^i}$$
$$\text{s.t.} \quad \sum_{i=1}^{m} p^i x_j^i = \sum_{i=1}^{m} p^i e_j^i$$

Lagrangian of this problem:

$$\mathcal{L}^j = \sum_{i=1}^{m} \alpha_i \frac{\left(x_j^i\right)^{1+\omega_j^i}}{1+\omega_j^i} + \xi_j \sum_{i=1}^{m} p^i \left(e_j^i - x_j^i\right)$$

FOCs:

$$[x_j^i] : \alpha_i \left(x_j^i\right)^{\omega_j^i} = \xi_j p^i$$
$$[\text{BC}] : \sum_{i=1}^{m} p^i \left(e_j^i - x_j^i\right) = 0$$

Notice that there is a link between Pareto problem from the previous exercise and competitive equilibrium:

$$\frac{\eta_i}{\lambda_j} = \alpha_i \left(x_j^i\right)^{\omega_j^i} = \xi_j p^i$$

Once we have the solution of the Pareto problem, we can set $\eta_i = p_i$ and $\frac{1}{\lambda_j} = \xi_j$. At the same time, we need to find Pareto weights $\{\lambda_j\}$ associated with the competitive equilibrium. We would choose Pareto weights such that budget constraint of each household is satisfied:

$$\sum_{i=1}^{m} \underbrace{p^i}_{=\eta_i} \left(e_j^i - x_j^i\right) = 0 \quad \forall j = 1, ..., n$$
$$\Rightarrow \sum_{i=1}^{m} \eta_i e_j^i = \sum_{i=1}^{m} \eta_i \left(\frac{\eta_i}{\lambda_j \alpha_i}\right)^{\frac{1}{\omega_j^i}} \quad \forall j = 1, ..., n$$

Combine this equations with resource constraints (or equivalently market clearing conditions):

$$\sum_{i=1}^{m} \eta_i e_j^i = \sum_{i=1}^{m} \eta_i \left(\frac{\eta_i}{\lambda_j \alpha_i} \right)^{\frac{1}{\omega_j^i}} \quad \forall j = 1, ..., n$$
$$\sum_{j=1}^{n} \left(\frac{\eta_i}{\lambda_j \alpha_i} \right)^{\frac{1}{\omega_j^i}} = \sum_{j=1}^{n} e_j^i \quad \forall i = 1, ..., m$$

Normalize one of the Pareto weights to 1: $\lambda_1 = 1$ and omit one of the budget constraints of consumers (Walras' law):

$$\lambda_1 = 1$$
$$\sum_{i=1}^{m} \eta_i e_j^i = \sum_{i=1}^{m} \eta_i \left(\frac{\eta_i}{\lambda_j \alpha_i} \right)^{\frac{1}{\omega_j^i}} \quad \forall j = 2, ..., n$$
$$\sum_{j=1}^{n} \left(\frac{\eta_i}{\lambda_j \alpha_i} \right)^{\frac{1}{\omega_j^i}} = \sum_{j=1}^{n} e_j^i \quad \forall i = 1, ..., m$$

We get $m + n - 1$ equations in $m + n - 1$ unknowns: $\{\lambda_2, ..., \lambda_n, \eta_1, ..., \eta_m\}$. Our solution for $\{\eta_1, ..., \eta_m\}$ will give us the vector of prices $\{p_1, ..., p_m : p_i = \eta_i\}$.

Code Q5.m solves this system of equations for $n = m = 3$ and fair amount of heterogeneity in terms of elasticity of substitution parameters, individual endowments and weights of different goods in the utility function of an agent. We use a vector of ones as our initial guess for Lagrange multipliers and prices. Code runs in 0.064 seconds and needs 14 iterations to converge.

6 Value Function Iteration

6.1 Social Planner

Let $\beta = 0.97, \omega^g = 0.2, \chi = 1, \alpha = 0.33, \delta = 0.1, \psi = 0.1$.

We are going to assume that for social planner taxes are exogenous, they are a part of the physical environment over which social planner has no control, so he needs to take this exogenous process into account.

Since taxes are exogenous in this model, government expenditures are determined by the exogenous tax rate, wage and labor. Then we can write the following process for government expenditures that social planner needs to take into account:

$$g_t = \tau_t w_t l_t = \tau_t F_l' (z_t, k_t, l_t) l_t = (1 - \alpha) \tau_t e^{z_t} k_t^{\alpha} l_t^{1-\alpha} = g (z_t, \tau_t, k_t, l_t)$$

Then social planner problem can be written as follows:

$$V^{SP}(z,\tau,k,i) = \max_{c,l,k',i'} \left\{ \log c + \omega^g \log g(\tau,z,k,l) - \frac{\chi}{2} l^2 + \beta \sum_{\tau'} \sum_{z'} \pi(\tau'|\tau)\, \pi(z'|z)\, V^{SP}(z',\tau',k',i') \right\}$$

$$\text{s.t.} \quad c + i' + g(\tau,z,k,l) = e^z k^\alpha l^{1-\alpha}$$

$$k' = (1-\delta)k + \left[1 - \frac{\psi}{2}\left(\frac{i'}{i} - 1\right)^2\right] i'$$

Here k is predetermined level of capital and i is previous period investment.

6.2 Steady State

Q6_2.m contains a function which solves for steady state given parameters of the model.

Since we have distortionary taxation and government expenditures in the utility function of the agent, we do not have the equivalence between social planner formulation in the previous part and recursive competitive equilibrium. So, we need to consider the problem of the household and other parts of competitive equilibrium to solve our model. Looking at the social planner problem in the previous part, we can conjecture that there will be four aggregate states z, τ, K, I and two individual states k, i.

Firm's FOCs:

$$w(z,\tau,K,I) = (1-\alpha)\, e^z K^\alpha \left(L^d(z,\tau,K,I)\right)^{-\alpha}$$
$$r(z,\tau,K,I) = \alpha e^z K^{\alpha-1} \left(L^d(z,\tau,K,I)\right)^{1-\alpha}$$

For household problem notice that households do not control government expenditures, and since utility which household gets from government expenditures enters utility function additively, we can drop them from the value function for the household (for the purpose of finding equilibrium).

Problem of the household:

$$V(z,\tau,k,i,K,I) = \max_{c,l,k',i'} \left\{ \log c - \frac{\chi}{2} l^2 + \beta \sum_{\tau'} \sum_{z'} \pi(\tau'|\tau)\, \pi(z'|z)\, V(z',\tau',k',i',K',I') \right\}$$

$$\text{s.t.} \quad c + i' = (1-\tau)\, w(z,\tau,K,I)\, l + r(z,\tau,K,I)\, k$$

$$k' = (1-\delta)k + \left[1 - \frac{\psi}{2}\left(\frac{i'}{i} - 1\right)^2\right] i'$$

$$K' = H^K(z,\tau,K,I)$$

$$I' = H^I(z,\tau,K,I)$$

FOCs:

$$[c]: \frac{1}{c} = \lambda$$

$$[l]: (1-\tau)\, w(z,\tau,K,I)\, \lambda = \chi l$$

$$[k']: \beta \sum_{\tau'} \sum_{z'} \pi(\tau'|\tau)\, \pi(z'|z)\, V_k'(z',\tau',k',i',K',I') - \mu = 0$$

$$[i']: \beta \sum_{\tau'} \sum_{z'} \pi(\tau'|\tau)\, \pi(z'|z)\, V_i'(z',\tau',k',i',K',I') - \lambda + \mu \left[\left[1 - \frac{\psi}{2}\left(\frac{i'}{i} - 1\right)\right] - \psi \frac{i'}{i}\left(\frac{i'}{i} - 1\right)\right] = 0$$

$$V_k'(z,\tau,k,i,K,I) = \lambda r(z,\tau,K,I) + \mu(1-\delta)$$

$$V_i'(z,\tau,k,i,K,I) = \mu\psi\left(\frac{i'}{i} - 1\right)\left(\frac{i'}{i}\right)^2$$

$$\Rightarrow (1-\tau)\, w(z,\tau,K,I)\, \frac{1}{c} = \chi l$$

$$\beta \sum_{\tau'} \sum_{z'} \pi(\tau'|\tau)\, \pi(z'|z) \left[\frac{1}{c'} r(z',\tau',K',I') + \mu'(1-\delta)\right] = \mu$$

$$\beta \sum_{\tau'} \sum_{z'} \pi(\tau'|\tau)\, \pi(z'|z) \left[\mu'\psi\left(\frac{i''}{i'} - 1\right)\left(\frac{i''}{i'}\right)^2\right] = \frac{1}{c} - \mu\left[\left[1 - \frac{\psi}{2}\left(\frac{i'}{i} - 1\right)\right] - \psi\frac{i'}{i}\left(\frac{i'}{i} - 1\right)\right]$$

$$c + i' = (1-\tau)\, w(z,\tau,K,I)\, l + r(z,\tau,K,I)\, k$$

$$k' = (1-\delta)\, k + \left[1 - \frac{\psi}{2}\left(\frac{i'}{i} - 1\right)^2\right] i'$$

This is a system of 5 equations in 5 unknown policy functions: consumption, labor, capital, investment, Lagrange multiplier μ. Denote policy function for variable x as $g^x(z,\tau,k,i,K,I)$.

Consistency:

$$g^{k'}(z,\tau,K,I,K,I) = H^K(z,\tau,K,I)$$
$$g^{i'}(z,\tau,K,I,K,I) = H^I(z,\tau,K,I)$$

Government budget constraint:

$$G(z,\tau,K,I) = \tau w(z,\tau,K,I)\, L^d(z,\tau,K,I)$$

Market clearing:

$$L^d(z,\tau,K,I) = g^l(z,\tau,K,I,K,I)$$
$$g^c(z,\tau,K,I,K,I) + g^i(z,\tau,K,I,K,I) + G(\tau,z,K,I) = e^z K^\alpha \left(L^d(z,\tau,K,I)\right)^{1-\alpha}$$

Now, let's evaluate model's equation at the deterministic steady state ($\tau_{ss} = 0.25, z_{ss} = 0$). Overall, our system is (with $k = K, i = I$):

$$(1-\tau)\, w\,(z,\tau,K,I)\,\frac{1}{c}=\chi l$$

$$\beta\sum_{\tau'}\sum_{z'}\pi\left(\tau'|\tau\right)\pi\left(z'|z\right)\left[\frac{1}{c'}r\left(z',\tau',K',I'\right)+\mu'\left(1-\delta\right)\right]=\mu$$

$$\beta\sum_{\tau'}\sum_{z'}\pi\left(\tau'|\tau\right)\pi\left(z'|z\right)\left[\mu'\psi\left(\frac{i''}{i'}-1\right)\left(\frac{i''}{i'}\right)^{2}\right]=\frac{1}{c}-\mu\left[\left[1-\frac{\psi}{2}\left(\frac{i'}{i}-1\right)\right]-\psi\frac{i'}{i}\left(\frac{i'}{i}-1\right)\right]$$

$$c+i'=(1-\tau)\, w\,(z,\tau,K,I)\, l+r\,(z,\tau,K,I)\, k$$

$$k'=(1-\delta)\, k+\left[1-\frac{\psi}{2}\left(\frac{i'}{i}-1\right)^{2}\right]i'$$

$$w\,(z,\tau,K,I)=(1-\alpha)\, e^{z}K^{\alpha}\left(L^{d}(z,\tau,K,I)\right)^{-\alpha}$$

$$r\,(z,\tau,K,I)=\alpha e^{z}K^{\alpha-1}\left(L^{d}(z,\tau,K,I)\right)^{1-\alpha}$$

$$L^{d}\,(z,\tau,K,I)=g^{l}\,(z,\tau,K,I,K,I)$$

$$G\,(z,\tau,K,I)=\tau w\,(z,\tau,K,I)\, L^{d}\,(z,\tau,K,I)$$

Notice that in the deterministic steady state $i''=i'=i, k'=k$. Our system becomes:

$$[1]\,(1-\tau_{ss})\, w_{ss}\frac{1}{c_{ss}}=\chi l_{ss}$$

$$[2]\beta\left[r_{ss}+(1-\delta)\right]=1$$

$$[3]\mu_{ss}=\frac{1}{c_{ss}}$$

$$[4]c_{ss}+i_{ss}=(1-\tau)\, w_{ss}l_{ss}+r_{ss}k_{ss}$$

$$[5]i_{ss}=\delta k_{ss}$$

$$[6]w_{ss}=(1-\alpha)\, k_{ss}^{\alpha}\,(l_{ss})^{-\alpha}$$

$$[7]r_{ss}=\alpha k_{ss}^{\alpha-1}\,(l_{ss})^{1-\alpha}$$

$$[8]G_{ss}=\tau_{ss}w_{ss}l_{ss}$$

$$[9]y_{ss}=(k_{ss})^{\alpha}\,(l_{ss})^{1-\alpha}$$

Let's replace household budget constraint [4] with resource constraint, and also omit [3]. Redefine all variables so that they are relative to steady state value of output (except for prices and y_{ss}):

$$[1]\,(1-\tau_{ss})\,w_{ss}\frac{1}{c_{ss}}=\chi\,(y_{ss})^2\,l_{ss}$$

$$[2]\beta\,[r_{ss}+(1-\delta)]=1$$

$$[3]c_{ss}+i_{ss}+G_{ss}=1$$

$$[5]i_{ss}=\delta k_{ss}$$

$$[6]w_{ss}=(1-\alpha)\left(\frac{k_{ss}}{l_{ss}}\right)^{\alpha}$$

$$[7]r_{ss}=\alpha\left(\frac{k_{ss}}{l_{ss}}\right)^{\alpha-1}\Rightarrow\frac{k_{ss}}{l_{ss}}=\left(\frac{r_{ss}}{\alpha}\right)^{\frac{1}{\alpha-1}}$$

$$[8]G_{ss}=\tau_{ss}w_{ss}l_{ss}$$

$$[9]1=\left(\frac{k_{ss}}{l_{ss}}\right)^{\alpha}l_{ss}$$

From [2] we know r_{ss}. Then from [7] we know $\frac{k_{ss}}{l_{ss}}$. Then we know w_{ss} from [6] and l_{ss} from [9]. Knowing l_{ss}, we also know k_{ss} and hence i_{ss}. Then [8] gives is G_{ss} and [3] gives us c_{ss}. Also, [1] gives us y_{ss}. Q6_2.m implements this steps to find steady state of the model.

6.3 Value Function Iteration with a Fixed Grid

Q6_3.m contains a function which solves for value function of the model.

Simplifying our dynamic system from before, we get:

$$[1]\,(1-\tau)\,(1-\alpha)\,e^{z}k^{\alpha}\,(l)^{-\alpha}\,\frac{1}{c}=\chi l$$

$$[2]\beta\sum_{\tau'}\sum_{z'}\pi\left(\tau'|\tau\right)\pi\left(z'|z\right)\left[\frac{1}{c'}\alpha e^{z'}\left(k'\right)^{\alpha-1}\left(l'\right)^{1-\alpha}+\mu'\,(1-\delta)\right]=\mu$$

$$[3]\beta\sum_{\tau'}\sum_{z'}\pi\left(\tau'|\tau\right)\pi\left(z'|z\right)\left[\mu'\psi\left(\frac{i''}{i'}-1\right)\left(\frac{i''}{i'}\right)^{2}\right]=\frac{1}{c}-\mu\left[\left[1-\frac{\psi}{2}\left(\frac{i'}{i}-1\right)\right]-\psi\frac{i'}{i}\left(\frac{i'}{i}-1\right)\right]$$

$$[4]c+i'+G\left(\tau,z,K,I\right)=w\left(z,K,I\right)l+r\left(z,K,I\right)k=\alpha e^{z}k^{\alpha}l^{1-\alpha}+(1-\alpha)e^{z}k^{\alpha}l^{1-\alpha}$$

$$\Rightarrow c+i'+\tau\alpha e^{z}k^{\alpha}l^{1-\alpha}=\alpha e^{z}k^{\alpha}l^{1-\alpha}+(1-\alpha)e^{z}k^{\alpha}l^{1-\alpha}$$

$$\Rightarrow c+i'=(1-\tau)\,\alpha e^{z}k^{\alpha}l^{1-\alpha}+(1-\alpha)e^{z}k^{\alpha}l^{1-\alpha}=\left[(1-\tau)\,\alpha+(1-\alpha)\right]e^{z}k^{\alpha}l^{1-\alpha}$$

$$[5]k'=(1-\delta)\,k+\left[1-\frac{\psi}{2}\left(\frac{i'}{i}-1\right)^{2}\right]i'$$

Remember that our household problem is not identical to our social planner problem, so when we do value function iteration we have 6 states (4 aggregate and 2 individual), and we need to guess on labor demand and laws of motion for capital and investment. Alternatively, we could guess on current prices and law of motion for prices.

In order to reduce the dimensionality of our problem, let's try to write a problem of an agent similar to social planner such that this problem will give us FOCs identical to the equations above

but would not involve prices:

- We need to distort labor just like in the household problem
- We must not distort capital (in terms of FOC which involves rate of return)
- We must ensure that resource constraint implied by RCE is satisfied

Consider the following problem:

$$\tilde{V}(z,\tau,k,i) = \max_{c,l,k',i'} \left\{ \log c - \frac{1}{1-\tau}\frac{\chi}{2}l^2 + \beta \sum_{\tau'}\sum_{z'} \pi(\tau'|\tau)\,\pi(z'|z)\,\tilde{V}(z',\tau',k',i') \right\}$$

$$\text{s.t.} \quad c + i' + \tilde{g} = e^z k^\alpha l^{1-\alpha}$$

$$k' = (1-\delta)k + \left[1 - \frac{\psi}{2}\left(\frac{i'}{i} - 1\right)^2\right] i'$$

FOCs:

$$[c]: \frac{1}{c} = \lambda$$

$$[l]: -\frac{1}{1-\tau}\chi l + \lambda\left[(1-\alpha)\,e^z k^\alpha l^{-\alpha}\right] = 0 \Rightarrow (1-\tau)(1-\alpha)\,e^z k^\alpha l^{-\alpha}\frac{1}{c} = \chi l$$

$$[k']: \beta \sum_{\tau'}\sum_{z'} \pi(\tau'|\tau)\,\pi(z'|z)\,\tilde{V}_k(z',\tau',k',i') + \lambda\left[\alpha e^z k^{\alpha-1} l^{1-\alpha}\right] - \mu = 0$$

$$[i']: \beta \sum_{\tau'}\sum_{z'} \pi(\tau'|\tau)\,\pi(z'|z)\,\tilde{V}_i(z',\tau',k',i') - \lambda + \mu\left[\left[1 - \frac{\psi}{2}\left(\frac{i'}{i} - 1\right)\right] - \psi\frac{i'}{i}\left(\frac{i'}{i} - 1\right)\right] = 0$$

$$\tilde{V}_k(z,\tau,k,i) = \lambda\left[\alpha e^z k^{\alpha-1} l^{1-\alpha}\right] + \mu(1-\delta)$$

$$\tilde{V}_i(z,\tau,k,i) = \mu\psi\left(\frac{i'}{i} - 1\right)\left(\frac{i'}{i}\right)^2$$

$\Rightarrow$

$$[1]\,(1-\tau)(1-\alpha)\,e^z k^\alpha l^{-\alpha}\frac{1}{c} = \chi l$$

$$[2]\beta \sum_{\tau'}\sum_{z'} \pi(\tau'|\tau)\,\pi(z'|z)\left[\frac{1}{c'}\left[\alpha e^{z'}\left(k'\right)^{\alpha-1}\left(l'\right)^{1-\alpha}\right] + \mu'(1-\delta)\right] + \frac{1}{c}\left[\alpha e^z k^{\alpha-1} l^{1-\alpha}\right] = \mu$$

$$[3]\beta \sum_{\tau'}\sum_{z'} \pi(\tau'|\tau)\,\pi(z'|z)\left[\mu'\psi\left(\frac{i''}{i'} - 1\right)\left(\frac{i''}{i'}\right)^2\right] = \frac{1}{c} - \mu\left[\left[1 - \frac{\psi}{2}\left(\frac{i'}{i} - 1\right)\right] - \psi\frac{i'}{i}\left(\frac{i'}{i} - 1\right)\right]$$

$$[4]c + i' = e^z k^\alpha l^{1-\alpha} - \tilde{g}$$

$$\tilde{g}: \left[(1-\tau)\,\alpha + (1-\alpha)\right] e^z k^\alpha l^{1-\alpha} = e^z k^\alpha l^{1-\alpha} - \tilde{g}$$

$$\Rightarrow \tilde{g} = \left[1 - (1-\tau)\,\alpha - (1-\alpha)\right] e^z k^\alpha l^{1-\alpha} = \tau\alpha e^z k^\alpha l^{1-\alpha}$$

$$[5]k' = (1-\delta)k + \left[1 - \frac{\psi}{2}\left(\frac{i'}{i} - 1\right)^2\right] i'$$

Here $\tilde{g}$ is like a state contingent transfer $\tilde{g} = \tilde{g}(z, \tau, k, i)$. So, we get identical first order conditions meaning we would get the same policy functions for this problem and for original RCE.

Hence, we can solve for policy functions and value function of this problem (different value function is not a problem since we are not interested in welfare analysis).

Q6_3.m implements value function iteration for the case when there are no tax shocks (tax shocks can be added to the model in the same manner as productivity shocks).

In the absence of tax shock our problem looks as follows:

$$\begin{aligned}
\tilde{V}(z,k,i) = \max_{l,i'} \Bigg\{ & (1-\beta)\left[\log\left[e^z k^\alpha l^{1-\alpha} - i' - \tilde{g}(z,k,i)\right] - \frac{1}{1-\tau}\frac{\chi}{2}l^2\right] + \\
& + \beta \sum_{z'} \pi(z'|z)\, \tilde{V}\left(z', k'\left(k,i,i'\right), i'\right) \Bigg\} \\
\text{s.t.} \quad & k'\left(k,i,i'\right) = (1-\delta)k + \left[1 - \frac{\psi}{2}\left(\frac{i'}{i} - 1\right)^2\right] i'
\end{aligned}$$

The algorithm works as follows. At iteration j:

1. Guess $\tilde{g}_j(z,k,i), \tilde{V}_j(z,k,i)$. At initial step $\tilde{g}_0(z,k,i) = \tau\alpha e^z k^\alpha l_{ss}^{1-\alpha}, \tilde{V}_j(z,k,i) = \log[c_{ss}] - \frac{\chi}{2}\frac{1}{1-\tau}l_{ss}^2$.
2. For each value of i' in the grid, find optimal labor $l(z,k,i,i')$ using first order condition for labor:

$$\begin{aligned}
& \frac{1}{1-\tau}\chi l = \frac{1}{e^z k^\alpha l^{1-\alpha} - i' - \tilde{g}(k,i)}(1-\alpha)e^z k^\alpha l^{-\alpha} \\
\Rightarrow & \frac{1}{1-\tau}\chi l^2 = \frac{(1-\alpha)e^z k^\alpha l^{1-\alpha}}{e^z k^\alpha l^{1-\alpha} - i' - \tilde{g}(k,i)} \\
\Rightarrow & \frac{1}{1-\tau}\chi l^2 = \frac{(1-\alpha)\frac{\tilde{g}(z,k,i)}{\tau\alpha}}{\frac{\tilde{g}(z,k,i)}{\tau\alpha} - i' - \tilde{g}(z,k,i)} \\
\Rightarrow & l = \left(\frac{(1-\alpha)(1-\tau)\frac{1}{\chi}\frac{\tilde{g}(z,k,i)}{\tau\alpha}}{\frac{\tilde{g}(z,k,i)}{\tau\alpha} - i' - \tilde{g}(z,k,i)}\right)^{\frac{1}{2}}
\end{aligned}$$

3. To find optimal $i'(z,k,i)$, do the following steps:
   (a) For fixed i' compute $k'(k,i,i')$ using law of motion for capital.
   (b) Use linear interpolation to obtain $\tilde{V}(z,' k'(k,i,i'), i')$ for $k'(k,i,i')$ off the grid.
   (c) Compute expected continuation value for given (z,k,i,i')
   (d) Compute the objective for given (z,k,i,i')
   (e) Find optimal $i'(z,k,i)$
   (f) Find optimal $l(z,k,i)$
4. Compute $\tilde{g}_{j+1}(z,k,i) = \tau\alpha e^z k^\alpha l(z,k,i)^{1-\alpha}, \tilde{V}_{j+1}(z,k,i)$

5. Check convergence of $\tilde{g}_{j+1}(z,k,i)$, $\tilde{V}_{j+1}(z,k,i)$, i.e. compute distance between $\tilde{g}_j(z,k,i)$, $\tilde{V}_j(z,k,i)$ and $\tilde{g}_{j+1}(z,k,i)$, $\tilde{V}_{j+1}(z,k,i)$ correspondingly. If convergence was not achieved (distance between functions measured using sup norm exceeds tolerance level), then move to the new iteration with a new guess $\tilde{g}_{j+1}(z,k,i)$, $\tilde{V}_{j+1}(z,k,i)$.

At each iteration $\tilde{g}_j(z,k,i)$, $\tilde{V}_j(z,k,i)$, $l(z,k,i,i')$ and other functions are stored as matrices to exploit the ability of Matlab to efficiently perform operations over matrices.

For a grid with 100 points for capital and 50 points for investment the algorithm converges in 1321 iterations which takes 135 seconds.

Figure 1 and 2 show value functions which we obtain.

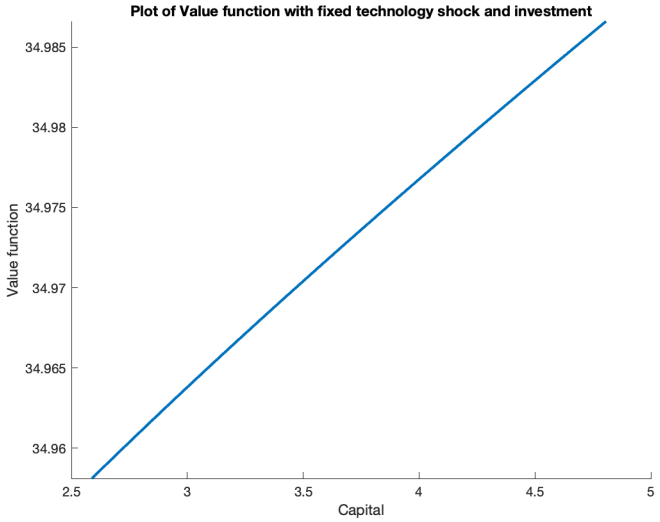

Figure 1: Value function as a function of capital for $z = 0$ and investment fixed at the center of the grid

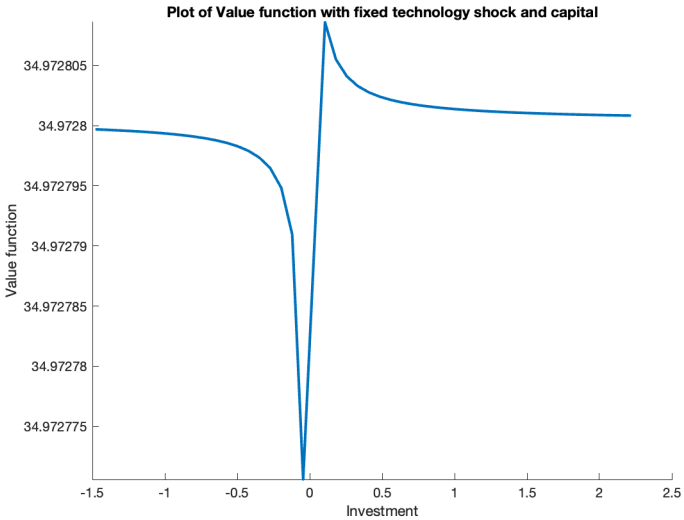

Figure 2: Value function as a function of investment for $z = 0$ and capital fixed at the center of the grid

Figure 3 shows policy functions for consumption.

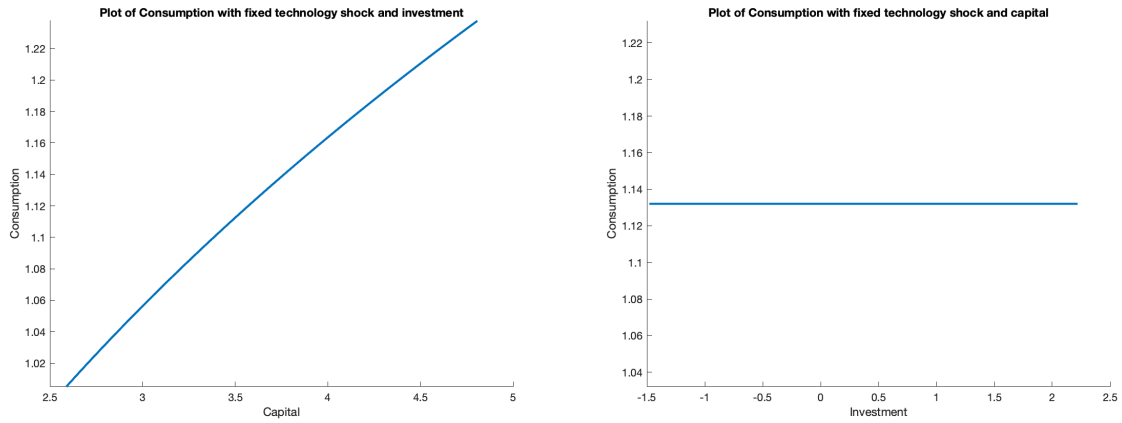

Figure 3: Policy function for consumption for $z = 0$

Figure 4 shows policy functions for labor.

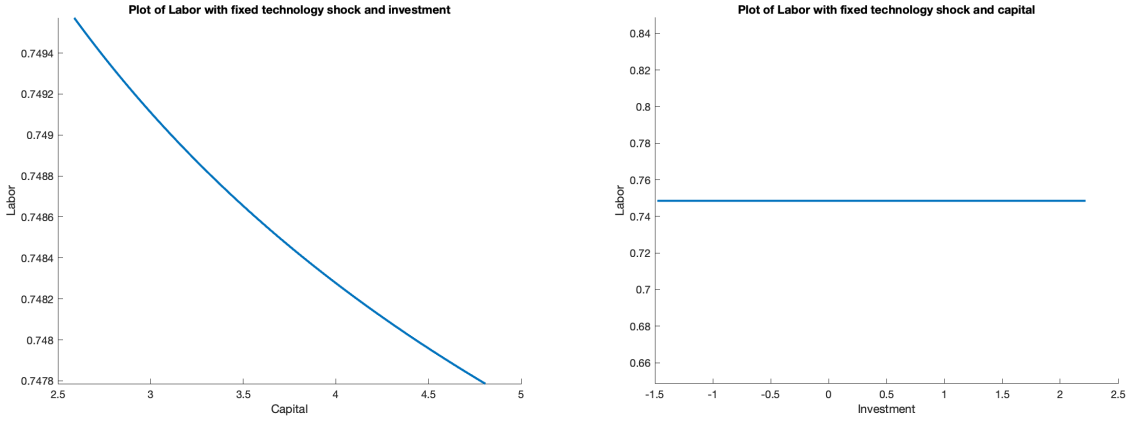

Figure 4: Policy function for labor for $z = 0$