Computational Methods HW1

Jack Dunbar

October 31, 2024

1 Integration: Lifetime Utility

We will compute lifetime utility:

$$\int_0^T e^{-\rho t} U\left(1 - e^{-\lambda t}\right) dt,$$

where $T = 100$, $\rho = 0.04$, $\lambda = 0.02$, and $U(c) = -e^{-c}$ using quadrature methods (midpoint rule, trapezoid rule, and Simpson's rule) and a Monte Carlo simulation.

Table 1: Comparison of Integration Methods on a Lifetime Utility Function

	1,000 bins/draws		**100,000 bins/draws**		**10,000,000 bins/draws**	
	Value	Time	Value	Time	Value	Time
Midpoint	-18.1592	0.0030	-18.2090	0.0027	-18.2095	0.1092
Trapezoid	-18.2096	0.0914	-18.2095	0.0206	-18.2095	0.1680
Simpson's	-18.1108	0.0023	-18.2085	0.0187	-18.2095	1.2270
Monte Carlo	-18.1745	0.0314	-18.1625	0.0047	-18.2129	0.1200

Table 1 shows integral values and the compute times (in seconds) for the four integration methods. The trapezoid rule does well for a small number of bins, but takes considerably longer than the other techniques. It also quickly converges with respect to the number of bins, followed by midpoint and Simpson's. Simpson's takes by far the longest for 10,000,000 bins, and that's because it is the only method relying on a for loop. Vectorization would speed Simpson's up considerably, as it did for the midpoint rule. Monte Carlo simulation is quite fast, but doesn't converge even for 10,000,000 draws.

2 Optimization: Rosenbrock Function

We will minimize the following Rosenbrock function:

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

using the Newton-Raphson, BFGS, steepest descent, and conjugate descent methods.

Table 2: Comparison of Minimization Methods on a Rosenbrock Function

	Newton-Raphson	**BFGS**	**Steepest Descent**	**Conjugate Descent**
Time (seconds)	0.0011	0.0027	0.0061	0.0071

Table 2 shows the compute time for each minimization method. For consistency, each method was given the same initial guess of $(0, 0)$, the same tolerance of 10^{-6}, and the same maximum number of iterations of 10,000. Newton-Raphson was quite fast, requiring only 3 iterations. It was followed by BFGS, which required 20. Steepest descent and conjugate descent were significantly slower, and did not even converge in 10,000 iterations. They do, however, converge with a tolerance of 10^{-4}. It seems that methods which depend on the Hessian perform better than those that don't, especially for this function.

3 Endowment Economy: Pareto Optimal Allocations

3.1 Theory

Consider an endowment economy with m goods and n agents. Each agent $i = 1, \ldots, n$ has an endowment $e^i_j > 0$ of each good $j = 1, \ldots, m$. Allocations are denoted $x^i_j > 0$, and each agent has a utility function of the form:

$$u^i(x^i) = \sum_j \alpha_j \frac{(x^i_j)^{1+\omega^i_j}}{1+\omega^i_j},$$

where $x^i = \{x^i_j\}_i$ and $\omega^i_j < 0 < \alpha_j$. Given Pareto weights λ_i, the social planner faces the following problem:

$$\begin{aligned}
&\max_{\{x^i_j\}_{i,j}} \quad \sum_i \lambda_i u^i(x^i) \\
&\text{s.t.} \\
&\sum_i x^i_j = \sum_i e^i_j \qquad \text{for all } j.
\end{aligned}$$

We solve for the optimal allocation numerically.

3.2 Computation

Table 3: PO Allocation with 3 Goods, 3 Agents, and No Heterogeneity

(good, agent)	**1**	**2**	**3**
1	0.82	1.01	1.17
2	0.82	1.01	1.17
3	0.82	1.01	1.17

In Table 3, the only heterogeneity is the Pareto weights placed on agents. Agents with higher indices have higher Pareto weights, and therefore receive more of each good.

Table 4: PO Allocation with 10 Goods, 10 Agents, and Heterogeneity

(good, agent)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0.83	0.87	0.91	0.95	0.99	1.02	1.06	1.09	1.12	1.15
2	0.84	0.88	0.92	0.95	0.99	1.02	1.06	1.09	1.12	1.15
3	0.84	0.88	0.92	0.96	0.99	1.02	1.05	1.08	1.11	1.14
4	0.85	0.89	0.92	0.96	0.99	1.02	1.05	1.08	1.11	1.13
5	0.86	0.89	0.93	0.96	0.99	1.02	1.05	1.08	1.10	1.13
6	0.86	0.90	0.93	0.96	0.99	1.02	1.05	1.07	1.10	1.12
7	0.87	0.90	0.93	0.96	0.99	1.02	1.05	1.07	1.10	1.12
8	0.87	0.90	0.93	0.96	0.99	1.02	1.04	1.07	1.09	1.11
9	0.87	0.91	0.94	0.97	0.99	1.02	1.04	1.07	1.09	1.11
10	0.88	0.91	0.94	0.97	0.99	1.02	1.04	1.06	1.09	1.11

In Table 4, as before, agents with higher indices are given higher Pareto weights, so they receive more of each good. Endowments aren't interesting from the POV of the social planner since they are free to distribute resources as they see fit. Also, α_j don't affect Pareto optimal allocations since they don't vary across goods and each agent's utility function is additively separable in the goods.

What makes things interesting is ω^i_j, which determines the curvature of each agent's utility function and therefore the dispersion in the allocation across agents. More specifically, it is the way in which ω^i_j varies across goods that matters. In Table 4, goods with higher indices have higher ω^i_j. More curvature means the allocation is less dispersed across agents.

4 Endowment Economy: Equilibrium Prices

4.1 Theory

Now, we decentralize the endowment economy. Given prices p_j, agent i solves the following problem:

$$\max_{\{x^i_j\}_j} \quad \sum_j \alpha_j \frac{(x^i_j)^{1+\omega^i_j}}{1+\omega^i_j}$$

s.t.

$$\sum_j p_j x^i_j = \sum_j p_j e^i_j.$$

Attaching Langrange multiplier μ to the budget constraint, and taking first-order conditions:

$$[x^i_j]: \qquad \alpha_j (x^i_j)^{\omega^i_j} - \mu p_j = 0.$$

Solving for demand as a function of μ:

$$x^i_j = \left(\frac{\mu p_j}{\alpha_j}\right)^{\frac{1}{\omega^i_j}}.$$

To obtain prices, we first plug demand into the budget constraint. This yields an implicit equation for μ. Most of the time when ω_j^i isn't the same across j, we must solve for μ numerically. Regardless, once we have μ, we plug it into demand x_j^i so that it only a function of prices. To obtain equilibrium prices, we substitute demand into each of our market clearing conditions:

$$\sum_i x_j^i = \sum_i e_j^i \qquad \text{for all } j.$$

The result is a nonlinear system of equations for prices, which we solve numerically.

4.2 Computation

Table 5: Equilibrium Prices with 10 Goods, 10 Agents, and Heterogeneity

p_1 (fixed)	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}
1.00	1.20	1.29	1.33	1.31	1.26	1.19	1.11	1.03	0.94

Agents with higher indices are given higher endowments e_j^i, which are constant across goods. High-index goods are simultaneously valued a lot (α_j high) and have more curvature in their component of the utility function (ω_j^i high). As we can see in Table 5, the result is a hump-shaped price vector. Goods with high indices are highly valued, but the curvature and endowment dispersion create demand for low-index goods. The balance of these forces creates the hump-shape.

5 Value Function Iteration

5.1 Social Planner Problem

Written recursively, the social planner's problem is

$$\begin{aligned} V(k, i^-; z) = \max_{c,g,l,k',i} \quad & U(c, g, l) + \beta \mathbb{E}[V(k', i; z', \tau') \mid z] \\ & \text{s.t.} \\ & c + i + g = f(k, l; z), \\ & k' = (1 - \delta)k + h(i, i^-), \end{aligned}$$

where the period utility function is

$$U(c, g, l) = \log c + \gamma \log g - \frac{l^2}{2},$$

the production function is

$$f(k, l; z) = e^z k^\alpha l^{1-\alpha},$$

and the investment function is

$$h(i, i^-) = \left[1 - \psi\left(\frac{i}{i^-} - 1\right)^2\right] i.$$

Attaching Lagrange multipliers λ and μ to the constraints, we obtain first-order conditions:

$$\begin{aligned}
[c]: &\qquad U_c(c,g,l) - \lambda = 0, \\
[l]: &\qquad U_l(c,g,l) + \lambda f_l(k,l;z) = 0, \\
[k']: &\qquad \beta\mathbb{E}[V_k(k',i;z') \mid z] - \mu = 0, \\
[i]: &\qquad \beta\mathbb{E}[V_{i^-}(k',i;z') \mid z] - \lambda + \mu h_i(i,i^-) = 0, \\
[g]: &\qquad U_g(c,g,l) - \lambda = 0.
\end{aligned}$$

FOCs 1 and 5 gives us our first intratemporal optimality condition:

$$U_c(c,g,l) - U_g(c,g,l) = 0.$$

Plugging FOC 1 into 2, we obtain our second intratemporal optimality condition:

$$U_l(c,g,l) + f_l(f,l;z)U_c(c,g,l) = 0.$$

The envelope theorem gives us:

$$\begin{aligned}
V_k(k',i;z') &= \lambda' f_k(k',l';z') + \mu'(1-\delta), \\
V_{i^-}(k',i;z') &= \mu' h_{i^-}(i',i).
\end{aligned}$$

Using the first result, we combine FOC 1 with 3:

$$\beta\mathbb{E}[U_c(c',g',l')f_k(k',l';z') + \mu'(1-\delta)] - \mu = 0.$$

Imposing the deterministic steady state and solving for μ:

$$\beta[U_c(c,g,l)f_k(k,l;z) + \mu(1-\delta)] - \mu = 0 \implies \mu = \frac{\beta U_c(c,g,l)f_k(k,l;z)}{1-\beta(1-\delta)}.$$

Plugging this result, FOC 1, and the second envelope result into FOC 4, we obtain a steady state Euler equation:

$$\frac{\beta f_k(k,l;z)}{1-\beta(1-\delta)}[\beta h_{i^-}(i,i) + h_i(i,i)] - 1 = 0.$$

Combined with market clearing and the law of motion for capital, we have five equations that characterize the steady state:

$$\begin{aligned}
U_c(c,g,l) - U_g(c,g,l) &= 0, \\
U_l(c,g,l) + f_l(f,l;z)U_c(c,g,l) &= 0, \\
\frac{\beta f_k(k,l;z)}{1-\beta(1-\delta)}[\beta h_{i^-}(i,i) + h_i(i,i)] - 1 &= 0, \\
f(k,l;z) - c - i - g &= 0, \\
(1-\delta)k + h(i,i) - k &= 0,
\end{aligned}$$

where

$$
\begin{aligned}
U_c(c,g,l) &= \frac{1}{c}, \\
U_g(c,g,l) &= \frac{\gamma}{g}, \\
U_l(c,g,l) &= -l, \\
f_l(k,l;z) &= (1-\alpha)e^z \left(\frac{k}{l}\right)^{\alpha}, \\
f_k(k,l;z) &= \alpha e^z \left(\frac{k}{l}\right)^{\alpha-1}, \\
h_i(i,i^-) &= \left[1-\psi\left(\frac{i}{i^-}-1\right)^2\right] - \frac{2\psi i}{i^-}\left[\frac{i}{i^-}-1\right], \\
h_{i^-}(i,i^-) &= 2\psi\left(\frac{i}{i^-}\right)^2\left(\frac{i}{i^-}-1\right).
\end{aligned}
$$

5.2 Decentralized Steady State

5.2.1 Theory

The recursive problem of the HH is:[1]

$$
\begin{aligned}
v(k,i^-;z,\tau) = \max_{c,l,k',i} \quad & U(c,g,l) + \beta\mathbb{E}[v(k',i;z',\tau') \mid z,\tau] \\
\text{s.t.} \quad & \\
& c + i = (1-\tau)wl + rk, \\
& k' = (1-\delta)k + h(i,i^-).
\end{aligned}
$$

Attach Lagrange multipliers λ and μ to the constraints, we obtain first-order conditions:

$$
\begin{aligned}
[c]: &\quad U_c(c,g,l) - \lambda = 0, \\
[l]: &\quad U_l(c,g,l) + \lambda(1-\tau)w = 0, \\
[k']: &\quad \beta\mathbb{E}[v_k(k',i;z',\tau') \mid z,\tau] - \mu = 0, \\
[i]: &\quad \beta\mathbb{E}[v_{i^-}(k',i;z',\tau') \mid z,\tau] - \lambda + \mu h_i(i,i^-) = 0.
\end{aligned}
$$

Plugging FOC 1 into 2, we obtain our intratemporal optimality condition:

$$
U_l(c,g,l) + (1-\tau)wU_c(c,g,l) = 0.
$$

The envelope theorem gives us:

$$
\begin{aligned}
v_k(k',i;z',\tau') &= \lambda' r' + \mu'(1-\delta), \\
v_{i^-}(k',i;z',\tau') &= \mu' h_{i^-}(i',i).
\end{aligned}
$$

[1] There should also be state variables and laws of motion for aggregate capital and aggregate lagged investment, but I didn't realize this while writing my solution.

Using the first result, we combine FOC 1 with 3:

$$\beta\mathbb{E}[U_c(c', g', l')r' + \mu'(1-\delta)] - \mu = 0.$$

Imposing the deterministic steady state and solving for μ:

$$\beta[U_c(c, g, l)r + \mu(1-\delta)] - \mu = 0 \implies \mu = \frac{\beta U_c(c, g, l)r}{1-\beta(1-\delta)}.$$

Using the this result, FOC 1, and the second envelope result in FOC 4, we obtain a steady state Euler equation:

$$\frac{\beta r}{1-\beta(1-\delta)}[\beta h_{i^-}(i, i) + h_i(i, i)] - 1 = 0.$$

The static problem of the firm gives us:

$$\begin{aligned} r &= f_k(k, l; z), \\ w &= f_l(k, l; z). \end{aligned}$$

And so the government's budget constraint becomes:

$$g = \tau f_l(k, l; z)l.$$

Thus we have five equations that characterize the steady state:

$$\begin{aligned} U_l(c, g, l) + (1-\tau) f_l(k, l; z)U_c(c, g, l) &= 0, \\ \frac{\beta f_k(k, l; z)}{1-\beta(1-\delta)}[\beta h_{i^-}(i, i) + h_i(i, i)] - 1 &= 0, \\ (1-\tau) f_l(k, l; z)l + f_k(k, l; z)k - c - i &= 0, \\ (1-\delta)k + h(i, i) - k &= 0, \\ \tau f_l(k, l; z)l - g &= 0. \end{aligned}$$

Now, we turn to the computer to solve for the steady state.

5.2.2 Computation

Table 6: Deterministic Steady State with $z = 0$ and $\tau = 0.25$

	Social Planner Steady State	**Decentralized Steady State**
c	1.48	0.85
g	0.30	0.25
l	1.04	0.93
k	5.98	3.70
i	0.60	0.37
f_l/w	0.38	0.42
f_k/r	0.10	0.13

Table 6 shows that distortionary taxes result in lower consumption, government spending, labor, capital, and investment. The marginal productivities of labor and capital (wages and interest rates) are higher since factors are under-utilized.

5.3 Value Function Iteration with a Fixed Grid

5.3.1 Algorithm

Conceptually, my code proceeds as follows:[2]

1. Begin by computing initial guesses for wages and interest rates, policy functions, and the value function based on deterministic steady states. Guesses for policy functions and the value function will vary across productivities and tax rates. Guesses for wages and interest rates will both vary productivities and tax rates as well as across the capital grid.

2. Begin with a guess for wages and interest rates.

3. Given wages and interest rates, solve for the household's value function via value function iteration:

$$
\begin{aligned}
v^k(k, i^-; z, \tau) = \max_{c,l,k',i} \quad & U(c, g, l) + \beta \mathbb{E}[v^{k-1}(k', i; z', \tau') \mid z, \tau] \\
\text{s.t.} \quad & \\
& c + i = (1-\tau) w l + r k, \\
& k' = (1-\delta) k + h(i, i^-).
\end{aligned}
$$

   Stop when the value function converges.

4. Compute the wages and interest rates consistent with household (and firm) decisions. If wages and interest rates have converged, you are done. Otherwise, go back to step 2 with the new wages and interest rates as your guess.

5.3.2 Results

My code takes $\approx$ 40 seconds to converge with 0.1 tolerances (for both wages/interest rates and value functions) on a 5x5 capital-lagged investment grid. There are many opportunities to improve efficiency, some of which are noted in my code.

That said, the results are pretty sensible. Figure 1 shows that the value function increases in productivity, capital, and lagged investment. It decreases in taxes, as expected. Figure 2 shows that the capital policy function is increasing in productivity, capital, and lagged investment. Figure 3 shows that wages increase in productivity and capital. Wages also decrease in lagged investment, which is less obvious.

[2] As noted before: There should also be state variables and laws of motion for aggregate capital and aggregate lagged investment, but I didn't realize this while writing my solution. The household should also take these as given.

Figure 1: Value Function $v(k, i^-; z, \tau)$

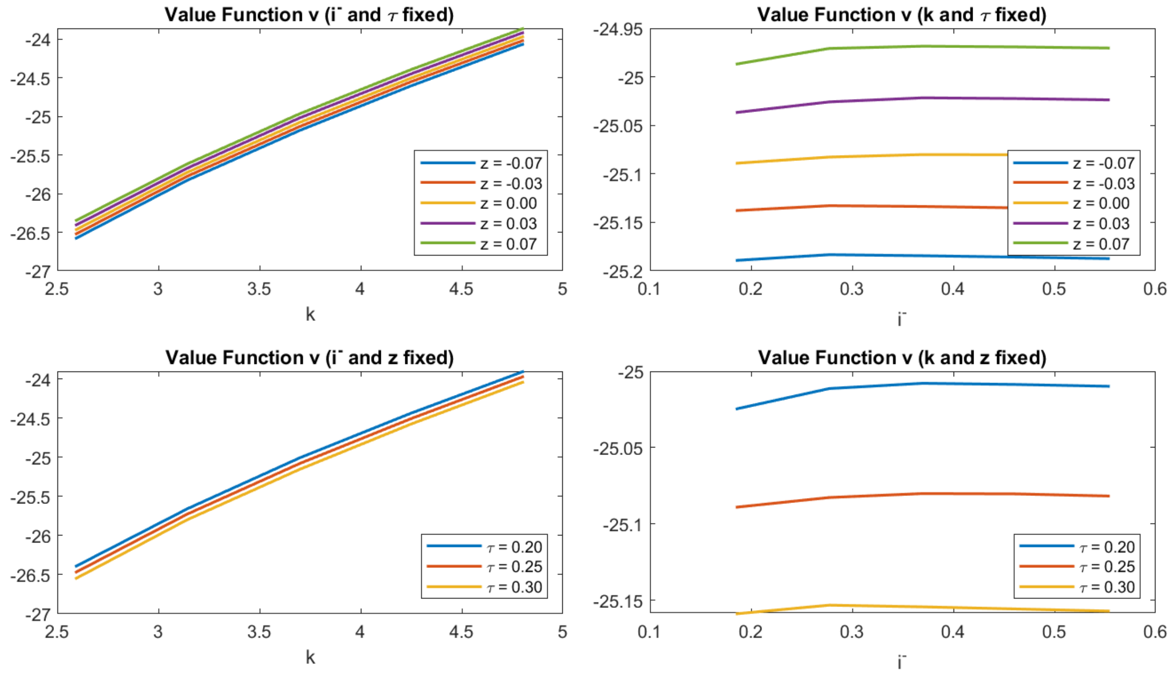

Figure 2: Capital Policy Function $k'(k, i^-; z, \tau)$

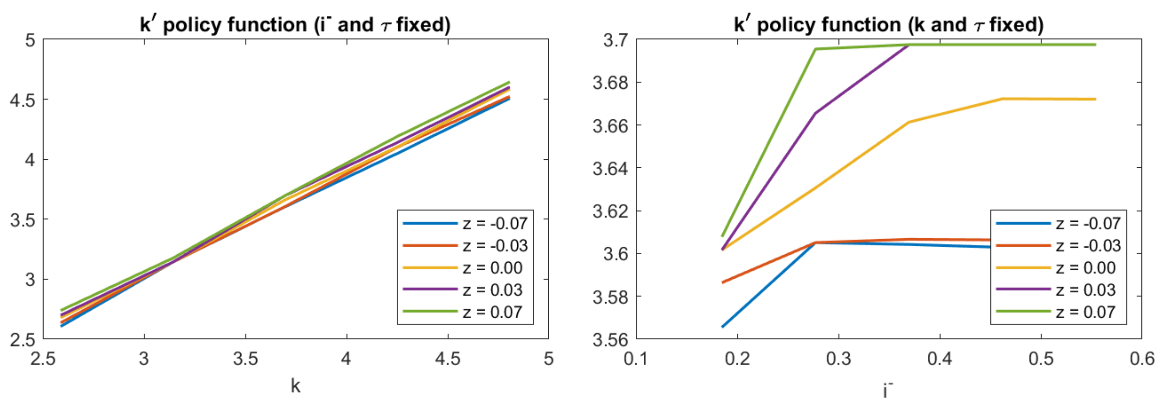

Figure 3: Wages $w(k, i^{-}; z, \tau)$