ECON 6140

Problem Set 3

Gabe Sekeres

February 12, 2025

Health inequality and mitigation tools

1. If we impose risk-sharing, which follows directly from the social planner caring for each type equally, the utilitarian social planner solves the problem

$$\max_{\{c_t,h_t,x_t,k_{t+1},\pi_t\}} \sum_{t=0}^{\infty} \frac{1}{2}\left((\beta\pi_t)^t + \beta^t\right) u(c_t)$$

subject to

$$\begin{aligned} k_{t+1} &= x_t + (1-\delta)k_t \\ y_t &= Ak_t^{\alpha} \\ y_t &= c_t + h_t + x_t \\ \pi_t &= f\left(\frac{h_t}{y_t}\right) \\ c_t, h_t, x_t, k_{t+1} &\geq 0 \\ k_0 &> 0 \text{ given} \end{aligned}$$

Their Lagrangian is

$$\mathcal{L} = \sum_{t=0}^{\infty} \beta^t \Bigg\{ (\pi_t^t + 1)u(c_t) + \lambda_t\left(x_t + (1-\delta)k_t - k_{t+1}\right) \\ + \theta_t\left(Ak_t^{\alpha} - c_t - h_t - x_t\right) + \mu_t\left(f\left(\frac{h_t}{Ak_t^{\alpha}}\right) - \pi_t\right)\Bigg\}$$

which admits first order conditions

$$0 = (\pi_t^t + 1)u'(c_t) - \theta_t \tag{c_t}$$

$$0 = -\theta_t + \frac{\mu_t}{Ak_t^{\alpha}} f'\left(\frac{h_t}{Ak_t^{\alpha}}\right) \tag{h_t}$$

$$0 = \lambda_t - \theta_t \tag{x_t}$$

$$0 = -\lambda_t + \beta\left[\lambda_{t+1}(1-\delta) + \theta_{t+1}\alpha Ak_{t+1}^{\alpha-1} - \frac{\mu_{t+1}\alpha h_{t+1}}{Ak_{t+1}^{\alpha+1}} f'\left(\frac{h_{t+1}}{Ak_{t+1}^{\alpha}}\right)\right] \tag{k_{t+1}}$$

$$0 = t\pi_t^{t-1}u(c_t) - \mu_t \tag{π_t}$$

Substituting for the Lagrange multipliers in the fourth first order condition, we get

$$(\pi_t^t + 1)u'(c_t) = \beta(\pi_{t+1}^t + 1)u'(c_{t+1})\left[(1-\delta) + \alpha Ak_{t+1}^{\alpha-1} - \frac{\alpha h_{t+1}}{k_{t+1}}\right]$$

In steady state, this becomes

$$1 = \beta \left[(1-\delta) + \alpha A (k^\star)^{\alpha-1} - \alpha h^\star (k^\star)^{-1} \right]$$

and feasibility requires

$$c^\star + h^\star + \delta k^\star = A(k^\star)^\alpha$$

We have two equations for three unknowns ($h^\star$, $k^\star$, and $c^\star$). To get a third equation, we use the first order condition with respect to h and the definition of π, which, in steady state, is

$$\theta = \frac{\mu}{A(k^\star)^\alpha} f'\left(\frac{h^\star}{A(k^\star)^\alpha} \right) \Longrightarrow (f(\cdot)^t + 1) u'(c^\star) = \frac{t f(\cdot)^{t-1} u(c^\star)}{A(k^\star)^\alpha} f'(\cdot)$$

which, substituting from the feasibility constraint, becomes

$$\left[f\left(\frac{h^\star}{A(k^\star)^\alpha} \right)^t + 1 \right] u'(A(k^\star)^\alpha - \delta k^\star - h^\star) = \frac{t f\left(\frac{h^\star}{A(k^\star)^\alpha} \right)^{t-1} u(A(k^\star)^\alpha - \delta k^\star - h^\star)}{A(k^\star)^\alpha} f'\left(\frac{h^\star}{A(k^\star)^\alpha} \right)$$

With the Euler equation

$$1 = \beta \left[(1-\delta) + \alpha A (k^\star)^{\alpha-1} - \alpha h^\star (k^\star)^{-1} \right]$$

this is a characterization of $k^\star$ and $h^\star$. However, it cannot be further reduced to remove the t term. Instead, we will impose a steady state, making the assumption that a nontrivial steady state exists in which $\pi_t = \pi \in (0,1)$ for all t. In this case, the planner's objective function becomes, using the properties of geometric series,

$$\max_{\{c,h,x,k,\pi\}} \frac{1}{2} \sum_{t=0}^{\infty} (\beta\pi)^t u(c) + \frac{1}{2} \sum_{t=0}^{\infty} \beta^t u(c) \equiv \max_{\{c,h,x,k,\pi\}} \frac{1}{2} \left(\frac{1}{1-\beta\pi} + \frac{1}{1-\beta} \right) u(c)$$

subject to

$$\begin{aligned} k &= x + (1-\delta)k \\ y &= Ak^\alpha \\ y &= c + h + x \\ \pi &= f(h/y) \\ c, h, x, k &\geq 0 \end{aligned}$$

So the Lagrangian is

$$\mathcal{L} = \frac{1}{2} \left(\frac{1}{1-\beta\pi} + \frac{1}{1-\beta} \right) u(c) + \lambda(x - \delta k) + \theta(Ak^\alpha - c - h - x) + \mu \left(f(h/Ak^\alpha) - \pi \right)$$

which admits first order conditions

$$\begin{aligned} 0 &= \frac{1}{2} \left(\frac{1}{1-\beta\pi} + \frac{1}{1-\beta} \right) u'(c) - \theta && (c) \\ 0 &= -\theta + \frac{\mu}{Ak^\alpha} f'(h/Ak^\alpha) && (h) \\ 0 &= \lambda - \theta && (x) \\ 0 &= -\delta\lambda + \theta\alpha A k^{\alpha-1} - \mu \frac{h}{Ak^{\alpha+1}} f'(h/Ak^\alpha) && (k) \\ 0 &= \frac{\beta}{2} \frac{1}{(1-\beta\pi)^2} u(c) - \mu && (\pi) \end{aligned}$$

Substituting the Lagrange multipliers in the fourth condition, it becomes

$$\delta = \alpha A k^{\alpha-1} - \frac{h}{k} \Longrightarrow h = \alpha A k^{\alpha} - \delta k$$

So from feasibility, we have

$$c = Ak^{\alpha} - (\alpha A k^{\alpha} - \delta k) - \delta k = (1-\alpha) A k^{\alpha}$$

Meaning that consumption is a constant fraction of output. Finally, combining a few conditions, we get that

$$\frac{\beta}{(1-\beta\pi)^2} u((1-\alpha)Ak^{\alpha}) = \left(\frac{1}{1-\beta\pi} + \frac{1}{1-\beta}\right) \frac{Ak^{\alpha}}{f'(\alpha - \delta k / Ak^{\alpha})} u'((1-\alpha)Ak^{\alpha})$$

Which precisely characterizes k in terms of the model primitives once we have functional forms for u and f.

2. The immortal households' problem reduces to the traditional

$$\max_{\{c_t, x_t, k_{t+1}\}} \sum_{t=0}^{\infty} \beta^t u(c_t)$$

subject to

$$\begin{aligned} k_{t+1} &= x_t + (1-\delta)k_t \\ y_t &= Ak_t^{\alpha} \\ y_t &= c_t + x_t \\ c_t, x_t, k_{t+1} &\geq 0 \end{aligned}$$

where we trivially impose that $h_t = 0$ for all t. The Lagrangian is

$$\mathcal{L} = \sum_{t=0}^{\infty} \beta^t \left\{ u(c_t) + \lambda_t \left(x_t + (1-\delta)k_t - k_{t+1}\right) + \theta_t \left(Ak_t^{\alpha} - c_t - x_t\right) \right\}$$

which admits the first order conditions

$$\begin{aligned} 0 &= u'(c_t) - \theta_t && (c_t) \\ 0 &= \lambda_t - \theta_t && (x_t) \\ 0 &= -\lambda_t + \beta\left[\lambda_{t+1}(1-\delta) + \theta_{t+1}\alpha A k_{t+1}^{\alpha-1}\right] && (k_{t+1}) \end{aligned}$$

so the Euler equation is

$$u'(c_t) = \beta u'(c_{t+1})\left[(1-\delta) + \alpha A k_{t+1}^{\alpha-1}\right]$$

which in steady state becomes

$$1 = \beta\left[(1-\delta) + \alpha A k^{\alpha-1}\right] \Longrightarrow k = \left(\frac{\frac{1}{\beta} - (1-\delta)}{\alpha A}\right)^{\frac{1}{\alpha-1}}$$

Meanwhile, the mortal households solve

$$\max_{\{c_t, x_t, k_{t+1}, h_t, \pi_t\}} \sum_{t=0}^{\infty} (\beta\pi_t)^t u(c_t)$$

subject to

$$\begin{aligned} k_{t+1} &= x_t + (1-\delta)k_t \\ y_t &= Ak_t^\alpha \\ y_t &= c_t + h_t + x_t \\ \pi_t &= f(h_t/y_t) \\ c_t, h_t, x_t, k_{t+1} &\geq 0 \end{aligned}$$

As in part (1), the first order conditions will require taking the derivative with respect to π_t, so in order to find a steady state we again assume that one exists, in which $\pi_t = \pi \in (0,1)$ for all t. The problem becomes

$$\max_{\{c,h,x,k,\pi\}} \frac{1}{1-\beta\pi} u(c)$$

subject to

$$\begin{aligned} k &= x + (1-\delta)k \\ y &= Ak^\alpha \\ y &= c + h + x \\ \pi &= f(h/y) \\ c, h, x, k &\geq 0 \end{aligned}$$

So the Lagrangian is

$$\mathcal{L} = \frac{1}{1-\beta\pi} u(c) + \lambda(x - \delta k) + \theta(Ak^\alpha - c - h - x) + \mu(f(h/Ak^\alpha) - \pi)$$

which admits first order conditions

$$\begin{aligned} 0 &= \frac{u'(c)}{1-\beta\pi} - \theta && (c) \\ 0 &= -\theta + \frac{\mu}{Ak^\alpha} f'(h/Ak^\alpha) && (h) \\ 0 &= \lambda - \theta && (x) \\ 0 &= \lambda\delta + \theta\alpha Ak^{\alpha-1} + \mu \frac{h}{Ak^{\alpha+1}} f'(h/Ak^\alpha) && (k) \\ 0 &= \frac{\beta}{(1-\beta\pi)^2} u(c) - \mu && (\pi) \end{aligned}$$

As in the social planner's problem, we have that

$$\delta = \alpha Ak^{\alpha-1} - \frac{h}{k} \implies h = \alpha Ak^\alpha - \delta k$$

and from feasibility, we have that

$$c = Ak^\alpha - (\alpha Ak^\alpha - \delta k) - \delta k = (1-\alpha)Ak^\alpha$$

So as above, consumption is a constant fraction of output. Finally, combining conditions, we have that k is precisely characterized by

$$\frac{\beta}{1-\beta\pi} u(c) = u'(c) \frac{Ak^\alpha}{f'(h/Ak^\alpha)}$$

3. In the social planner's problem, the health investment to output ratio is

$$\frac{h}{y} = \frac{\alpha A k^\alpha - \delta k}{A k^\alpha} = \alpha - \frac{\delta k}{A k^\alpha}$$

Meanwhile in the competitive equilibrium, the immortal households invest 0 and the mortal households invest $\alpha A k^\alpha - \delta k$. Assuming that the fraction of mortal households is $\gamma \in (0,1)$, we have that the health investment to output ratio at some total capital level k is

$$\frac{(1-\gamma)\cdot 0 + \gamma \cdot h}{y} = \gamma\alpha - \frac{\gamma \delta k}{A k^\alpha}$$

which is a fixed proportion γ of the health investment in the social planner's problem. Intuitively, this is because the immortal households do not care about the investment in health at all.

4. If the government bans pesticides, all households will be immortal. The planner's problem becomes (assuming risk-sharing, as before)

$$\max_{\{c_t, x_t, k_{t+1}\}} \sum_{t=0}^{\infty} \beta^t u(c_t)$$

subject to

$$\begin{aligned} k_{t+1} &= x_t + (1-\delta) k_t \\ y_t &= A' k_t^\alpha \\ y_t &= c_t + x_t \\ c_t, x_t, k_{t+1} &\geq 0 \end{aligned}$$

Since there are no longer any returns to investing in health, we remove those conditions and set $h^\star = 0$ in steady state. Observe that this problem is identical to the problem the immortal households solve in the competitive equilibrium. The steady state investment in capital will be

$$k^\star = \left(\frac{\frac{1}{\beta} - (1-\delta)}{\alpha A'} \right)^{\frac{1}{\alpha - 1}}$$

While we did not have a precise closed-form characterization above, we can say that this is strictly larger than the investment in part (1), since in part (1) the social planner invested positive amounts in health.

5. From part (1), we have that in the steady state

$$\frac{\beta}{(1-\beta\pi)^2} u(c) = \left(\frac{1}{1-\beta\pi} + \frac{1}{1-\beta} \right) \frac{y}{f'(h/y)} u'(c)$$

So we have that the value of life is

$$\frac{u(c)}{u'(c)} = \frac{(1-\beta\pi)^2}{\beta} \left(\frac{1}{1-\beta\pi} + \frac{1}{1-\beta} \right) \frac{y}{f'(h/y)}$$

Recall that the investment in health above is a direct function of k, which is defined implicitly by this expression. As the value of life increases, the optimal investment in capital will decrease and the optimal investment in health will increase, as we can see from the rightmost term. This makes implicit sense – as life is more valuable, the social planner is willing to invest more to increase life.

Saving policies: risk and non-homotheticities

1. The farmer's maximization problem can be written as

$$\max_{\{c_t\}} \sum_{t=0}^{\infty} \beta^t \ln(c_t) \text{ s.t. } \sum_{t=0}^{\infty} c_t = x_0$$

   Taking the value function as the attained maximum, we get that starting with endowment x_t in period t,

$$v(x_t) = \max_{c_t} \ln(c_t) + \beta v(x_t - c_t)$$

   so, rewriting in standard notation, we get that

$$v(x) = \max_{c \in [0,x]} \ln(c) + \beta v(x - c)$$

2. I discretized the initial endowment space with 1000 points from 0 to 100, and calculated the value function for the farmer across that grid. The value function is plotted in Figure 1. The code is included below this section. I added an absorbing state where $v(0) = 0$. This will become relevant in part (3) of the problem, for now it causes some strangeness around low levels of the initial endowment but otherwise changes nothing.

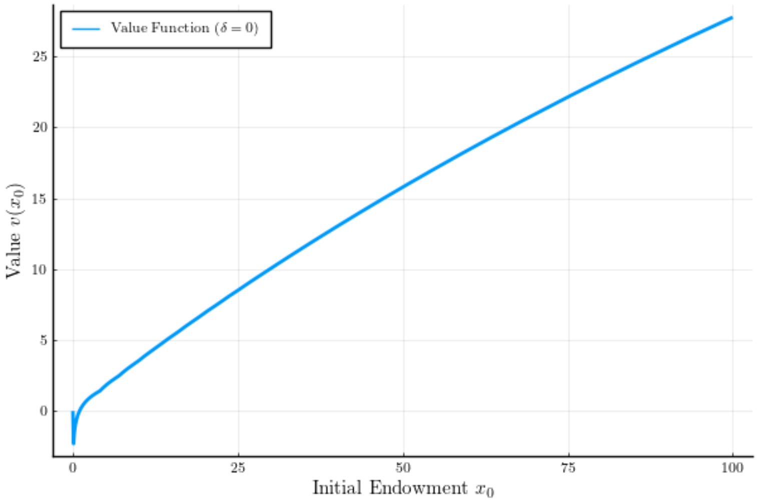

Figure 1: Value to the Farmer (when $\delta = 0$)

3. The farmer has a probability δ of losing the entire endowment each period. Of course, this runs into the immediate problem that $\log(0) = -\infty$, so any expected utility calculation would lead to, in expectation, $-\infty$ utility. Instead, we will manually set $v(0) = 0$, and restrict the range of initial endowments to those where the value should be positive – see above, where we set the minimum endowment at 2.5. With this, the Bellman equation becomes

$$v(x) = \max_{c \in [0,x]} \ln(c) + \beta \left[\delta v(0) + (1 - \delta) v(x - c)\right]$$

   which simplifies to

$$v(x) = \max_{c \in [0,x]} \ln(c) + \beta (1 - \delta) v(x - c)$$

We can rewrite the code to include this value function, and get Figure 2. Note that the new value function is not a linear transformation of the old – consumption patterns shift, putting more weight on the present, which reduces total value more than linearly.

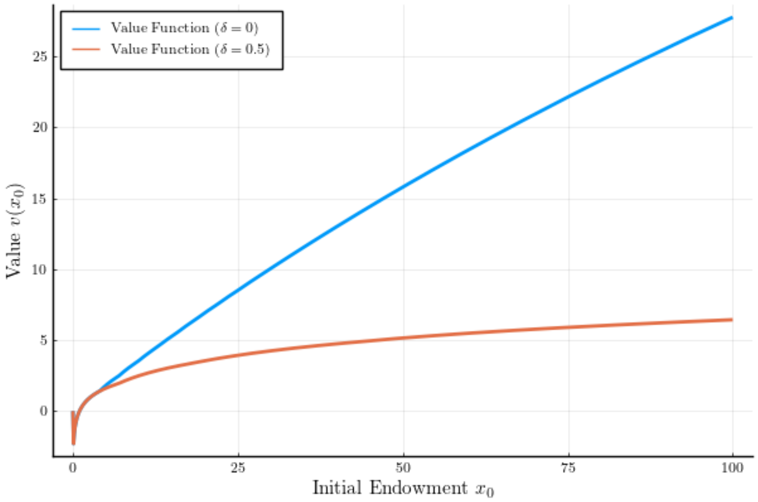

Figure 2: Value to the Farmer when $\delta = 0.5$

4. If we now have a lower bound on the value function $\bar{c} = x_0/100$, we can represent the (non-homothetic) utility function as:

$$u(c) = \begin{cases} \ln(c - \bar{c}) & c > \bar{c} \\ -\infty & c \leq \bar{c} \end{cases}$$

This admits the Bellman Equation

$$v(x) = \max_{c \in [\bar{c}, x]} \ln(c - \bar{c}) + \beta v(x - c)$$

The value functions are plotted in Figure 3. As we can see, there are no large differences at low initial endowment, but at higher levels of the initial endowment the farmer would clearly prefer to consume less than $\bar{c}$ in some periods, so we have further constrained their problem (which necessarily leads to less value).

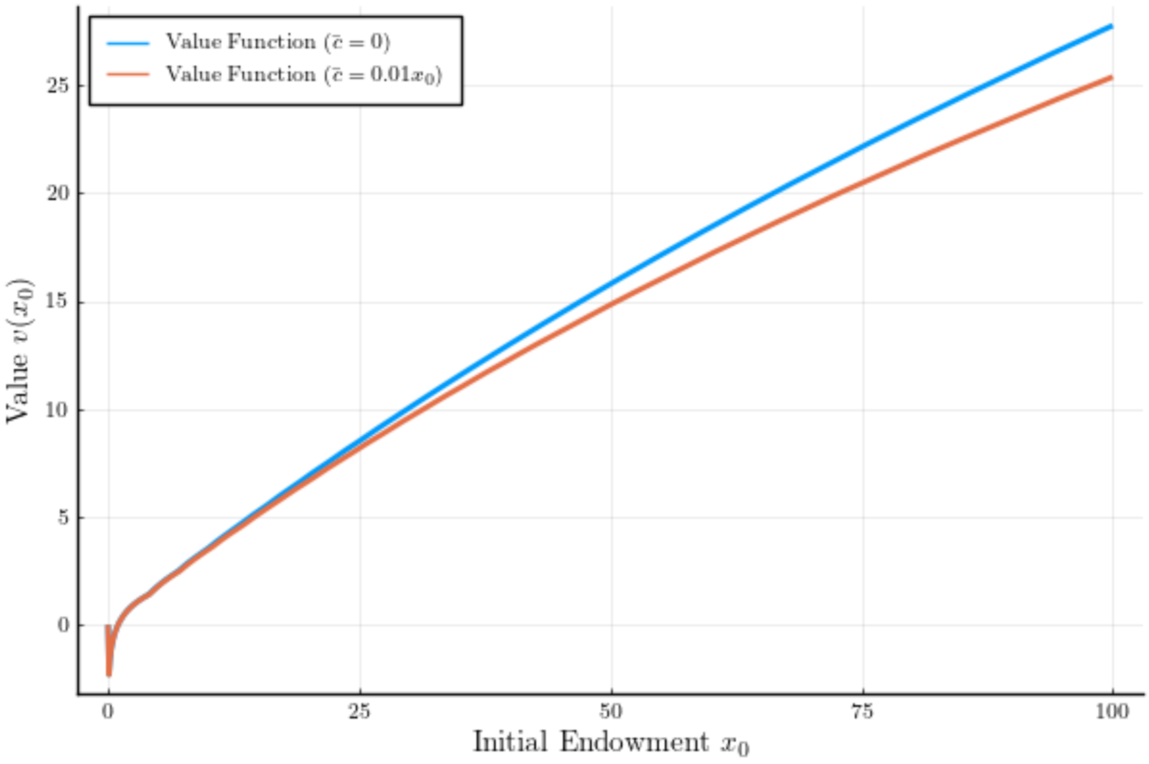

Figure 3: Value to the Farmer when $\bar{c} = 0.01x_0$

Code. The Julia code I used to complete this problem is below. It runs entirely in approximately 0.98 seconds.

```
using Plots, LaTeXStrings, PyPlot

@time begin

# Make the plots look pretty
pyplot()
PyPlot.rc("text", usetex=true)
PyPlot.rc("font", family="serif")
PyPlot.matplotlib.rcParams["mathtext.fontset"] = "cm"

# Set parameters
beta    = 0.98
N       = 1000
x_min   = 0.0
x_max   = 100.0
x_grid  = range(x_min, x_max, length = N)

# VFI settings
max_iter    = 500
tol         = 1e-4
```

```
# Settings for the consumption grid (per state)
M           = 200
eps         = 1e-6

# Utility function
u(c) = log(c)

# Linear interpolation
function interp_value(x, x_grid, V)
    # If x is at or below the minimum grid point, return V[1]
    if x <= first(x_grid)
        return V[1]
    # If x is above the maximum grid point, return the last value
    elseif x >= last(x_grid)
        return V[end]
    else
        # Find the index i such that x_grid[i] <= x < x_grid[i+1]
        i = searchsortedlast(x_grid, x)
        # Linear interpolation between x_grid[i] and x_grid[i+1]
        x_low, x_high = x_grid[i], x_grid[i+1]
        V_low, V_high = V[i], V[i+1]
        return V_low + (V_high - V_low) * (x - x_low) / (x_high - x_low)
    end
end

# Bellman operator
function bellman_operator(V, x_grid, beta; M = 200, eps = 1e-6, delta = 0.0)
    V_new = similar(V)
    for (i, x) in enumerate(x_grid)
        # At x = 0 (or very near 0), define v(0)=0 directly.
        if x < eps
            V_new[i] = 0.0
        else
            # Create a grid for consumption c from a tiny positive number to x
            c_grid = range(eps, stop = x, length = M)
            # Compute the candidate value for each consumption c
            # (Here, next cake is x - c, and we use interpolation for v(x-c))
            candidate_vals = [ u(c) + beta * (1 - delta) * interp_value(x - c,
                x_grid, V) for c in c_grid ]
            # The updated value function at state x is the maximum over
               candidates
            V_new[i] = maximum(candidate_vals)
        end
    end
    return V_new
end

# Value Function Iteration
function vfi(x_grid, beta; tol = 1e-4, max_iter = 500, M = 200, eps = 1e-6,
   delta = 0.0)
    V = zeros(length(x_grid))          # initial guess: v(x)=0 everywhere
    err = tol + 1.0
    iter = 0
    while err > tol && iter < max_iter
        V_new = bellman_operator(V, x_grid, beta; M = M, eps = eps, delta =
           delta)
        err = maximum(abs.(V_new .- V))
        V = V_new
        iter += 1
    end
    println("Converged in $iter iterations with error = $err")
    return V
end

# Initial problem
V_init = vfi(x_grid, beta; tol = tol, max_iter = max_iter, M = M, eps = eps,
   delta = 0.0)

# Plot the computed value function.
plot(x_grid, V_init, label = L"Value Function ($\delta = 0$)", xlabel = L"
   Initial Endowment $x_0$", ylabel = L"Value $v(x_0)$", lw = 2, legend = :
   topleft)
savefig("macro_hw3_code/value_function.png")


# Probability 0.5 of losing endowment
V_delta = vfi(x_grid, beta; tol = tol, max_iter = max_iter, M = M, eps = eps,
   delta = 0.5)

# Plot the computed value function.
plot!(x_grid, V_delta, label = L"Value Function ($\delta = 0.5$)", xlabel = L"
   Initial Endowment $x_0$", ylabel = L"Value $v(x_0)$", lw = 2, legend = :
   topleft)
savefig("macro_hw3_code/value_function_delta.png")


# Define subsistence level
c_bar = x_grid ./ 100

# Bellman operator with subsistence utility
function bellman_operator_subsistence(V, x_grid, beta; M = 200, eps = 1e-6,
   delta = 0.0, c_bar = x_grid ./ 100)
    V_new = similar(V)
    for (i, x) in enumerate(x_grid)
        # At x = 0 (or very near 0), define v(0)=0 directly.
        if x < eps
            V_new[i] = 0.0
        else
            # Create a grid for consumption c from a tiny positive number to x
            c_grid = range(c_bar[i], stop = x, length = M)
            # Compute the candidate value for each consumption c
            # (Here, next cake is x - c, and we use interpolation for v(x-c))
            candidate_vals = [ u(c - c_bar[i]) + beta * (1 - delta) *
               interp_value(x - c, x_grid, V) for c in c_grid ]
            # The updated value function at state x is the maximum over
               candidates
            V_new[i] = maximum(candidate_vals)
        end
    end
    return V_new
end

# Value Function Iteration with subsistence utility
function vfi_subsistence(x_grid, beta; tol = 1e-4, max_iter = 500, M = 200,
   eps = 1e-6, delta = 0.0, c_bar = x_grid ./ 100)
    V = zeros(length(x_grid))          # initial guess: v(x)=0 everywhere
    err = tol + 1.0
    iter = 0
    while err > tol && iter < max_iter
        V_new = bellman_operator_subsistence(V, x_grid, beta; M = M, eps = eps
           , delta = delta, c_bar = c_bar)
        err = maximum(abs.(V_new .- V))
        V = V_new
        iter += 1
    end
    println("Converged in $iter iterations with error = $err")
    return V
end

# Perform value function iteration
V_base = vfi(x_grid, beta; tol = tol, max_iter = max_iter, M = M, eps = eps,
   delta = 0.0)
V_subs = vfi_subsistence(x_grid, beta; tol = tol, max_iter = max_iter, M = M,
   eps = eps, delta = 0.0, c_bar = c_bar)

# Plot the computed value function.
plot(x_grid, V_base, label = L"Value Function ($\bar{c}=0$)", xlabel = L"
   Initial Endowment $x_0$", ylabel = L"Value $v(x_0)$", lw = 2, legend = :
   topleft)
plot!(x_grid, V_subs, label = L"Value Function ($\bar{c}=0.01x_0$)", xlabel =
   L"Initial Endowment $x_0$", ylabel = L"Value $v(x_0)$", lw = 2, legend = :
   topleft)
savefig("macro_hw3_code/value_function_subsistence.png")

end
```