Workshop 5: Numerical optimization

FIE463: Numerical Methods in Macroeconomics and Finance using Python

Richard Foltyn
NHH Norwegian School of Economics

February 13, 2025

See GitHub repository for notebooks and data:

https://github.com/richardfoltyn/FIE463-V25

Exercise 1: Consumption-leisure choice

In the lecture we analyzed how a household optimally chooses consumption and labor supply. In this exercise, we revisit this setting, imposing different assumptions on the household's utility function.

Assume that the household chooses *consumption* c and *leisure* ℓ in order to maximize

$$\begin{aligned} \max_{c,\,\ell} \; & \Big\{ \log(c) + \psi \log(\ell) \Big\} \\ \text{subject to } \; & c = a + (1-\ell)w \\ & c \geq 0 \\ & 0 \leq \ell \leq 1 \end{aligned}$$

The second line represents the budget constraint, where a is the level of assets and w is the wage rate. The last line imposes that leisure has to be between 0 and 1, where 1 represents the entire time endowment of the household. Consequently, for a given leisure choice, the household supplies the remaining time $1-\ell$ in the labor market and earns $(1-\ell)w$ in labor income.

Analytical solution

To solve the problem, we set up the Lagrangian,

$$\mathcal{L} = \log(c) + \psi \log(\ell) + \lambda \Big(a + (1-\ell)w - c \Big)$$

which gives us the first-order conditions

$$\left.\begin{aligned} \frac{\partial \mathcal{L}}{\partial c} &= \frac{1}{c} - \lambda &&= 0 \\ \frac{\partial \mathcal{L}}{\partial \ell} &= \frac{\psi}{\ell} - \lambda w &&= 0 \end{aligned}\right\} \implies c = \frac{\ell w}{\psi}$$

This condition characterizes the optimal allocation of consumption and leisure for a given wage rate. Substituting this back into the budget constraint, we can solve for optimal leisure:

$$\ell^* = \frac{\psi}{1+\psi} \frac{a+w}{w}$$

The optimal consumption level follows by plugging the above expression into the optimality condition we derived earlier.

$$c^* = \frac{1}{1+\psi}(a+w)$$

Numerical solution

Instead of solving this problem analytically, we can use a numerical solver to obtain the optimal choices for a given set of parameters.

1. Write a function `util()` which takes leisure ℓ and parameters as arguments and returns the corresponding utility level.

   ```
   def util(l, psi, a, w):
       """
       Compute utility for given leisure choice and parameters.
       """
   ```

   Use the budget constraint to obtain consumption for a given leisure choice.

2. Assume that the model is parametrized by $a = 0$, $w = 1$ and $\psi = 1$. Use the univariate minimizer `minimize_scalar()` from the module `scipy.optimize` and solve for the optimal leisure choice. Use the `util()` function you defined previously as the objective.

   Hint: Recall that you are using a *minimizer* to solve a maximization problem, hence you need to flip the sign of your objective function, e.g., by using a `lambda` expression.

   Hint: Use the `bounds` argument to `minimize_scalar()` to restrict the set of admissible values for ℓ.

3. You want to make the above implementation more flexible so you can experiment with different parameter values. Write a function `solve_cons_leisure()` which takes *only* the parameters as arguments and returns a tuple containing the optimal consumption and leisure choices:

   ```
   def solve_cons_leisure(psi, a, w):
       """
       Solve for the optimal consumption/leisure choice for given parameters.
       """
   ```

   For this part, you should just reuse the code you wrote earlier by wrap it in a function.

4. You are interested in how the optimal leisure and consumption choices depend on the utility weight on leisure, ψ. Use the function you wrote in the last part to evaluate optimal consumption and leisure for a grid of 21 points for ψ that is uniformly spaced on the interval $[0, 2]$.

   Create a graph with two panels, one for consumption and one for leisure, plotting the optimal choices against the values of ψ. What is the intuition for the slope of c and ℓ as ψ changes?

5. Since this problem admits an analytical solution, compute the *exact* optimal choices using the equations from above for each value on the ψ grid you used in the previous part.

   Adapt your graph to show both the numerical and the analytical solution. Use a legend to clearly label what is being plotted.

Exercise 2: Consumption & labor supply with commuting

Recall the consumption & labor supply problem we studied in the lecture: the individual has preferences over consumption and leisure given by

$$u(c, h) = \frac{c^{1-\gamma} - 1}{1 - \gamma} - \psi \frac{h^{1+1/\theta}}{1 + 1/\theta}$$

where $c \geq 0$ is the amount consumed, and $h \geq 0$ (for "hours") is the amount of labor the individual wishes to work (the remaining time is then consumed as leisure). The individual does not like working, and therefore the amount of hours worked enters as a disutility term in the utility function. Note that for

the special case of $\gamma = 1$, the utility from consumption becomes $\log(c)$, the natural logarithm of c, so that

$$u(c,h) = \log(c) - \psi \frac{h^{1+1/\theta}}{1+1/\theta}$$

Unlike in the lecture, we assume that the individual can either work part-time for $h < H$ hours in their neighborhood for a lower wage w_ℓ or commute to the city to work full-time for $h \geq H$ hours at a higher wage w_h. However, commuting to the city incurs a fixed commuting cost κ. The budget constraint is therefore given by

$$c = \begin{cases} a + w_\ell \cdot h & \text{if } h < H \\ a + w_h \cdot h - \kappa & \text{if } h \geq H \end{cases}$$

This is an example of a problem that is harder to solve analytically because of the discountinuous jump at the choice $h = H$. As you will see, this discontinuity can also cause problems for numerical solvers.

To get you started, use the following function to evaluate the utility for given choices and parameters (it's the same one we wrote in the lecture):

[12]:
```python
import numpy as np

def util(c, h, gamma, psi, theta):
    """
    Compute the utility of a given consumption/labor supply choice.

    Parameters
    ----------
    c : float or array
        Consumption level.
    h : float or array
        Hours worked.
    gamma : float
        Relative risk aversion parameter.
    psi : float
        Weight on disutility of labor.
    theta : float
        Labor supply elasticity.

    Returns
    -------
    u : float or array
        Utility value.
    """

    # Consumption utility
    if gamma == 1:
        # Log utility
        u = np.log(c)
    else:
        # General CRRA utility
        u = (c**(1-gamma) - 1) / (1-gamma)

    # add disutility of labor
    u -= psi * h**(1 + 1/theta) / (1 + 1/theta)

    return u
```

In the remainder of this exercise, you're asked to find the optimal choices using grid search and two different SciPy minimizer. Use the following parameters for these tasks:

[13]:
```python
# Parameters
a = 0           # initial assets
w_l = 0.75      # low wage (neighborhood)
w_h = 3         # high wage (city)
```

```
gamma = 1.0     # Relative risk aversion
psi = 2.0       # weight on disutility of labor
theta = 0.5     # labor supply elasticity
H = 1.0         # cut-off for part-time work
kappa = 2.0     # commuting cost
```

Tasks

1. Write a function with the signature

   ```
   def util_h(h, gamma, psi, theta, H, kappa, a, w_l, w_h):
       """
       Compute utility for given labor choice and parameters.
       """
   ```

   which uses `util()` to return the utility associated with a hours choice `h`. Use the budget constraint to obtained the implied consumption level.

   Hint: To get a vectorized function (which can deal with `h` arguments that are arrays), use `np.where()` instead of an `if` statement when dealing with the two cases in the budget constraint.

2. Solve the problem using grid search, evaluating utility on a grid of 10,000 points for h on the interval $[0.1, 2]$. Plot utility as a function of h for the values on the hours grid.

3. Use the scalar minimizer `minimize_scalar()` and the function `util_h()` you wrote to locate the optimum. Do you get a similar result as with grid search?

4. Lastly, use `util_h()` and a derivative-based minimizer implemented by `minimize()` to find the optimum.

   Hint: Call this minimizer as follows:

   ```
   res = minimize(
       lambda x: -util_h(x, gamma, psi, theta, H, kappa, a, w_l, w_h),
       x0 = x0,
       method='L-BFGS-B',
       bounds=((0, None),)
   )
   ```

   This selects the `'L-BFGS-B'` algorithm, a derivative-based method which also supports bounds. We impose these using the `bounds=((0, None),)` argument which imposes the $h \geq 0$ lower bound.

   You need to specify an initial guess `x0`: experiment with `x0=0.75` and `x0=1.1`. Do these yield the same result?

   Hint: The `minimize()` returns a result object where the optimum is stored in the `x` attribute, which in this case is an array of length 1.

Exercise 3: Consumption-savings with uncertain income

We have studied the two-period consumption savings problems several times now, but so far, income in the second period was deterministic. This setting can be extended to uncertain income, a feature that is at the core of modern macroeconomics.

Consider a household which maximizes *expected* lifetime utility by choosing period-1 consumption c_1 and savings s,

$$\begin{aligned} &\max_{c_1,\, s} \; \Big\{ u(c_1) + \beta \mathbb{E}\big[u(c_2)\big] \Big\} \\ \text{s.t.} \quad & c_1 + s = a \\ & c_2 = (1+r)s + y_2 \\ & c_1 \geq 0,\; s \geq 0 \end{aligned}$$

Per-period utility $u(c)$ is the CRRA utility function given by

$$u(c) = \begin{cases} \frac{c^{1-\gamma}}{1-\gamma} & \text{if } \gamma \neq 1 \\ \log(c) & \text{if } \gamma = 1 \end{cases}$$

where γ is the RRA coefficient and $\log(\bullet)$ denotes the natural logarithm.

The savings s generate a gross return $(1+r)$ in the second period. Additionally, the household receives stochastic income y_2. For simplicity, we assume that y_2 fluctuates symmetrically around its mean and can take on only two realizations:

$$y_2 = \begin{cases} \overline{y} + \epsilon & \text{with probability } \frac{1}{2} \\ \overline{y} - \epsilon & \text{with probability } \frac{1}{2} \end{cases}$$

You can easily verify that $\mathbb{E}[y_2] = \overline{y}$ and $\text{Var}(y_2) = \epsilon^2$ so that the parameters $\overline{y}$ and ϵ govern the mean and the volatility of period-2 income, respectively.

Analytical results

In general, it is not possible to solve such a problem analytically unless we are willing to impose simplifying assumptions (e.g., setting $\gamma = 1$, but even then the optimal s^* is characterized by a tedious quadratic equation).

Nevertheless, we can use the Euler equation for this problem,

$$u'(c_1) = \beta(1+r)\mathbb{E}\big[u'(c_2)\big]$$

to solve the problem numerically using root-finding. For this purpose, we use the functional form assumption for $u(\bullet)$ and plug in the period-specific budget constraints into the Euler equation to obtain,

$$(a-s)^{-\gamma} = \beta(1+r)\mathbb{E}\Big[\big((1+r)s + y_2\big)^{-\gamma}\Big] \tag{3.1}$$

which is a nonlinear equation in a single unknown, s.

Numerical solution

In the remainder of the exercise, you are asked to solve this problem numerically using both a minimizer and a root-finder approach. You should use the following implementation of per-period utility $u(\bullet)$:

[26]:
```
import numpy as np

def util(c, gamma):
    """
    Compute per-period utility for given consumption.

    Parameters
    ----------
    c : float or array
        Consumption level.
    gamma : float
```

```
        Relative risk aversion parameter.

    Returns
    -------
    u : float or array
        Utility value.
    """
    if gamma == 1:
        # Log preferences
        u = np.log(c)
    else:
        # General CRRA preferences
        u = c**(1-gamma) / (1-gamma)
    return u
```

Moreover, assume the following parameters for the tasks below:

[27]:
```
# Parameters
ybar = 0.5          # average period-2 income
epsilon = 0.2       # standard deviation of period-2 income
a = 1.0             # initial assets
r = 0.04            # interest rate
gamma = 2.0         # relative risk aversion
beta = 0.96         # discount factor
```

Tasks

1. Define a function `util_life()` with the following signature which should return the expected life-time utility for a given savings choice s:

   ```
   def util_life(s, beta, gamma, a, r, ybar, epsilon):
       """
       Evaluate lifetime utility for given savings choice and parameters.
       """
   ```

2. Plot this function against a grid of candidate savings levels s. You should define a savings grid with reasonale bounds for this task.

3. Write a function `solve_minimize()` with the following signature which should use `minimize_scalar()` from `scipy.optimize` to find the optimal choices for given parameters.

   ```
   def solve_minimize(beta, gamma, a, r, ybar, epsilon):
       """
       Solve consumption-savings problem using a minimizer.
       """
   ```

   Your function should return the tuple (c_1^*, s^*, U^*), i.e., the optimal consumption in period 1, optimal savings, and the maximized lifetime utility U^*.

   Run the minimizer for the given parameters and report the results.

4. The model we are studying features a *precautionary savings motive*: an increase in risk induces households to increase their savings.

   Investigate this behavior by computing the solution for a grid of ϵ defined on the interval $[0, \overline{y}]$. Note that varying epsilon only increases the variance of income shocks in period 2 but leaves the mean $\mathbb{E}y_2$ unaffected.

   Plot optimal savings against ϵ for both $\gamma = 1$ and $\gamma = 2$ in a single plot. Why are the responses different?

5. Lastly, find the optimal choices using a root-finder. To this end, rewrite the Euler equation (3.1) as

$$f(s) = \left(a - s\right)^{-\gamma} - \beta(1+r)\mathbb{E}\left[\left((1+r)s + y_2\right)^{-\gamma}\right]$$

which returns the Euler equation error for savings s. The optimal choice needs to satisfy $f(s^*) = 0$, so you need to find the root of f.

   1. Write a function `euler_err()` with the following signature which implements f and returns the Euler equation error:

```
def euler_err(s, beta, gamma, a, r, ybar, epsilon):
    """
    Compute the Euler equation error for a given savings choice.
    """
```

   2. Plot this function against the savings grid used earlier to verify that this function indeed has a root somewhere.
   3. Use the root-finder `root_scalar()` to locate this root (choose the method `brentq` and specify an initial `bracket`).
   4. Verify that this approach yields the same result as the minimizer.