Project 3: Optimal Consumption and Asset Allocation

Due: November 21, 2025

Objective

The goal of this assignment is to solve a dynamic stochastic optimal control problem. You will extend the ”Stochastic Cake Eating Problem” covered in class to include portfolio choice. Your task is to find the optimal time-varying policies for both consumption (c_t) and asset allocation (α_t) to maximize an agent’s expected lifetime utility, implemented in JAX.

Problem Description

Consider an agent with an initial wealth $x_0 = 10$ who lives for $T = 300$ periods. The agent’s preferences are described by a power utility function:

$$E_0\left[\sum_{t=0}^{T-1} \beta^t \frac{c_t^{1-\gamma}}{1-\gamma}\right]$$

with a time-discount factor $\beta = 0.95$ and risk-aversion coefficient $\gamma = 2$.

In each period t, the agent first decides how much of their wealth x_t to consume, c_t, and how much to save, $s_t = x_t - c_t$.

The agent then decides how to allocate their savings s_t between two assets:

1. **A risk-free asset**, which pays a gross return of $R_f = 1.04$.
2. **A risky asset (stock)**, whose log-return is normally distributed: $\log(R_{s,t+1}) \sim N(\mu_s, \sigma_s^2)$ with $\mu_s = 0.06$ and $\sigma_s = 0.2$.

Let α_t be the fraction of savings invested in the risky asset, where α_t is bounded: $0 \leq \alpha_t \leq 1$. The remaining fraction $(1 - \alpha_t)$ is invested in the risk-free asset.

The agent’s wealth at the beginning of the next period $(t + 1)$ is given by the evolution equation: $x_{t+1} = (x_t - c_t)(\alpha_t R_{s,t+1} + (1 - \alpha_t)R_f)$

The agent’s problem is to choose the policies $\{c_t, \alpha_t\}_{t=0}^{T-1}$ to maximize their expected utility, subject to the wealth evolution and the constraints $0 \leq c_t \leq x_t$ and $0 \leq \alpha_t \leq 1$.

Tasks and Requirements

1. Model Implementation

You must use machine learning implemented through JAX to solve this problem. The key elements must include:

- **Policy Functions:** The agent chooses two values each period. We will approximate the optimal policies as a fraction of wealth, c_t/x_t, and an allocation fraction, α_t. Crucially, these policies can change over time.
- **Time-Varying Parameters:** The ratios for consumption and asset allocation must be time dependent.

- **Wealth Evolution:** The agent's wealth should evolve according to these consumption and investment policies.

2. Training

Train your model to find the parameter vectors Θ_c and Θ_α that maximize the expected utility objective.

3. Final Simulation and Output

Once your model is trained, run a final, large-scale simulation using **1,000,000 (one million)** paths to generate your final results.

- At each time step $t = 0,...,299$, calculate the average consumption-wealth ratio (c_t/x_t) and the average asset allocation (α_t) across all one million paths.
- Save these results to a file named 'results.csv'.
- The file must **not** contain any headers.
- The file must contain exactly three columns:
- **Column 1:** The time index (0,1,2,...,299).
- **Column 2:** The average consumption-wealth ratio for that time t.
- **Column 3:** The average asset allocation for that time t.

Policy on AI Assistance

You are encouraged to use AI tools like ChatGPT, Gemini, or Copilot as assistants to help you with specific coding challenges. For example, using AI for debugging a piece of code, in-depth understanding of a JAX function, or exploring different ways to implement an algorithm is perfectly acceptable.

However, you are **strictly prohibited** from submitting the entire problem description (or significant portions of it) to an AI tool and asking for a complete solution. The goal of this assignment is for you to build and understand the model yourself. Submitting an AI-generated solution as your own will be considered a violation of academic integrity.

Submission

Your submission should consist of two files:

1. **results.csv**: The CSV file as described in Task 3, containing the time index, average consumption-wealth ratio, and average asset allocation.
2. **replication_code.py**: Your complete, runnable JAX code used to train the model and generate the final 'results.csv'.