Solution and Simulation of the Aiyagari-Bewley-Huggett-Imrohoglu Model

Hsiu-Yu Tu

2024/07/28

1 Introduction

This project aims to solve and simulate the Aiyagari-Bewley-Huggett-Imrohoglu horizon model. This model is a cornerstone in the analysis of heterogeneous agents under income uncertainty, capturing key elements of individual decision-making over time. By addressing the intertemporal consumption and savings choices of agents, the model provides insights into the dynamics of income, consumption, and wealth distribution.

This report details the methodology employed to solve and simulate the model, including the implementation of key equations and constraints. We present relevant figures to illustrate the results and provide the source code used for the simulations. Additionally, we discuss potential improvements and new directions for extending the model, aiming to enhance its realism and applicability to real-world economic scenarios.

2 Mathematical derivation

The households seek to maximize their utility over a finite time horizon, represented by the following objective function:

$$\max_{c_t, a_{t+1}} \mathbb{E}\left[\sum_{t=1}^{T} \beta^t \ln(c_t)\right] \tag{1}$$

subject to the budget constraint:

$$c_t + a_{t+1} = (1+r)a_t + y_t \tag{2}$$

$$a_t \geq \underline{a} \tag{3}$$

where log-income $\ln y$ follows an autoregressive process:

$$\ln(y_{t+1}) = \rho \ln(y_t) + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2) \tag{4}$$

To derive the Euler Equation, we set up the Lagrangian for the household's optimization problem:

$$\mathcal{L} = E\left[\sum_{t=1}^{T} \beta^t \ln(c_t) + \lambda_t \left((1+r)a_t + y_t - c_t - a_{t+1}\right)\right] \tag{5}$$

First-order condition with respect to c_t:

$$\frac{\partial \mathcal{L}}{\partial c_t} = \beta^t \frac{1}{c_t} - \lambda_t = 0 \implies \lambda_t = \beta^t \frac{1}{c_t} \tag{6}$$

First-order condition with respect to a_{t+1}:

$$\frac{\partial \mathcal{L}}{\partial a_{t+1}} = -\lambda_t + \lambda_{t+1}(1+r) = 0 \implies \lambda_t = \lambda_{t+1}(1+r) \tag{7}$$

Substituting λ_t:

$$\beta^t \frac{1}{c_t} = \beta^{t+1} \frac{1}{c_{t+1}}(1+r) \tag{8}$$

Rearranging terms, the Euler Equation is:

$$c_{t+1} = \beta(1+r)c_t \tag{9}$$

At $t = 40$:

$$c_{40} + a_{41} = (1+r)a_{40} + y_{40} \tag{10}$$

Given $a_{41} = 0$, we have:

$$\frac{c_{40} - y_{40}}{(1+r)} = a_{40} \tag{11}$$

Continuing backward:

$$c_{39} + a_{40} = (1+r)a_{39} + y_{39} \tag{12}$$

Substituting A_{40}:

$$\frac{c_{39} - y_{39}}{(1+r)} - \frac{c_{40} - y_{40}}{(1+r)^2} = a_{39} \tag{13}$$

Generalizing, we find:

$$a_t = \sum_{k=t}^{40} \frac{c_k - y_k}{(1+r)^{k-t+1}} \Rightarrow a_1 = \sum_{k=1}^{40} \frac{c_k - y_k}{(1+r)^k} = 0 \tag{14}$$

And since $a_1 = 0$ This results in:

$$\frac{c_1 - y_1}{(1+r)} \ldots + \frac{c_{40} - y_{40}}{(1+r)^{40}} = 0 \tag{15}$$

Summing over time and rearranging terms, we get:

$$(1+r)^{39}c_1 - (1+r)^{39}y_1 + (1+r)^{38}c_2 - (1+r)^{38}y_2 + \ldots + (1+r)c_{40} - y_{40} = 0 \tag{16}$$

By (9), we get:

$$c_1(1+r)^{39}\frac{(1-\beta^{40})}{1-\beta} = \sum_{s=1}^{40}(1+r)^{40-s}y_s \tag{17}$$

Solving for C_1^*:

$$c_1^* = \frac{(1-\beta)}{(1-\beta^{40})}\sum_{s=1}^{40}(1+r)^{1-s}y_s \tag{18}$$

Using (9) again:

$$c_t^* = \beta^{t-1}(1+r)^{t-1}\sum_{t=1}^{40}\frac{(1+r)^{1-s}}{(1-\beta^{40})}y_s \tag{19}$$

Simplifying further:

$$c_t^* = \beta^{t-1}\frac{(1-\beta)}{(1-\beta^{40})}\sum_{t=1}^{40}(1+r)^{t-s}y_s \tag{20}$$

3 Methodology

At this part, I try to use mathematical solution to process the optimization.Assume we are at period t in my first framework. The state variables known are y_t (income) and a_t (assets). First, we compute the expected path for income y. The optimal consumption c_t^* is then calculated using the Euler Equation, and backward induction from$a_T = 0$ to $a_T = 0$

$$c_t^* = \beta^{t-1}\frac{(1-\beta)}{(1-\beta^{40})}\sum_{t=1}^{40}(1+r)^{t-s}y_s$$

Hence, we get the equation(20). This equation ensures that the marginal utility of consumption today is equal to the discounted expected marginal utility of future consumption, adjusted by the interest rate r and discount factor β.

Next, we compute the asset level a_t^* from the budget constraint :

$$a_t^* = (1+r)a_{t-1} + y_t - c_t^*$$

If $a_t^* \geq \underline{a}$, we proceed to the next period $t+1$. However, if $a_t^* < \underline{a}$, we set $a_t^* = \underline{a}$ and recompute c_t^* from the budget constraint, ensuring that:

$$c_t^* = (1+r)a_{t-1} + y_t - a$$

This procedure accounts for an occasionally binding constraint on assets, ensuring the solution remains feasible within the given economic constraints. By imposing the lower bound a, we prevent the agent from having an asset level below the specified minimum, thus maintaining the economic feasibility of the model.

3.1 Mean Reversion in AR Process

The log-income $\ln(y)$ follows an autoregressive (AR) process:

$$\ln(y_{t+1}) = \rho \ln(y_t) + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2) \tag{21}$$

There is a mean reversion property for an AR process, particularly when $|\rho| < 1$. This means that the income Y_t will eventually revert to its long-term mean. In this context, the fluctuations in income Y_t will return to a certain mean over the long term, providing some predictability and stability in household income.

3.2 Occasionally Binding Constraint

When the asset level a_t exceeds the borrowing limit a_{max}, this constraint becomes binding; otherwise, it remains non-binding. The mean reversion property of the AR process implies that the income fluctuations will not be excessively volatile in the long term, thereby preventing the borrowing constraint from being binding for extended periods. This allows us to use the occasionally binding constraint approach. Combining Euler equation with occasionally binding constraint $a_t \geq \underline{a}$

When the constraint is not binding: :

$$U'(C_t) = \beta(1+r)E_t[U'(C_{t+1})]$$

When the constraint is binding:

$$U'(C_t) \geq \beta(1+r)E_t[U'(C_{t+1})]$$

In this scenario, the marginal utility of current consumption may be higher than the present value of the marginal utility of future consumption because the consumer cannot further borrow to smooth consumption.

4 Results & Analyses

For the three different borrowing constraints with 100,000 simulations for each, the model took 148.54 seconds to complete.

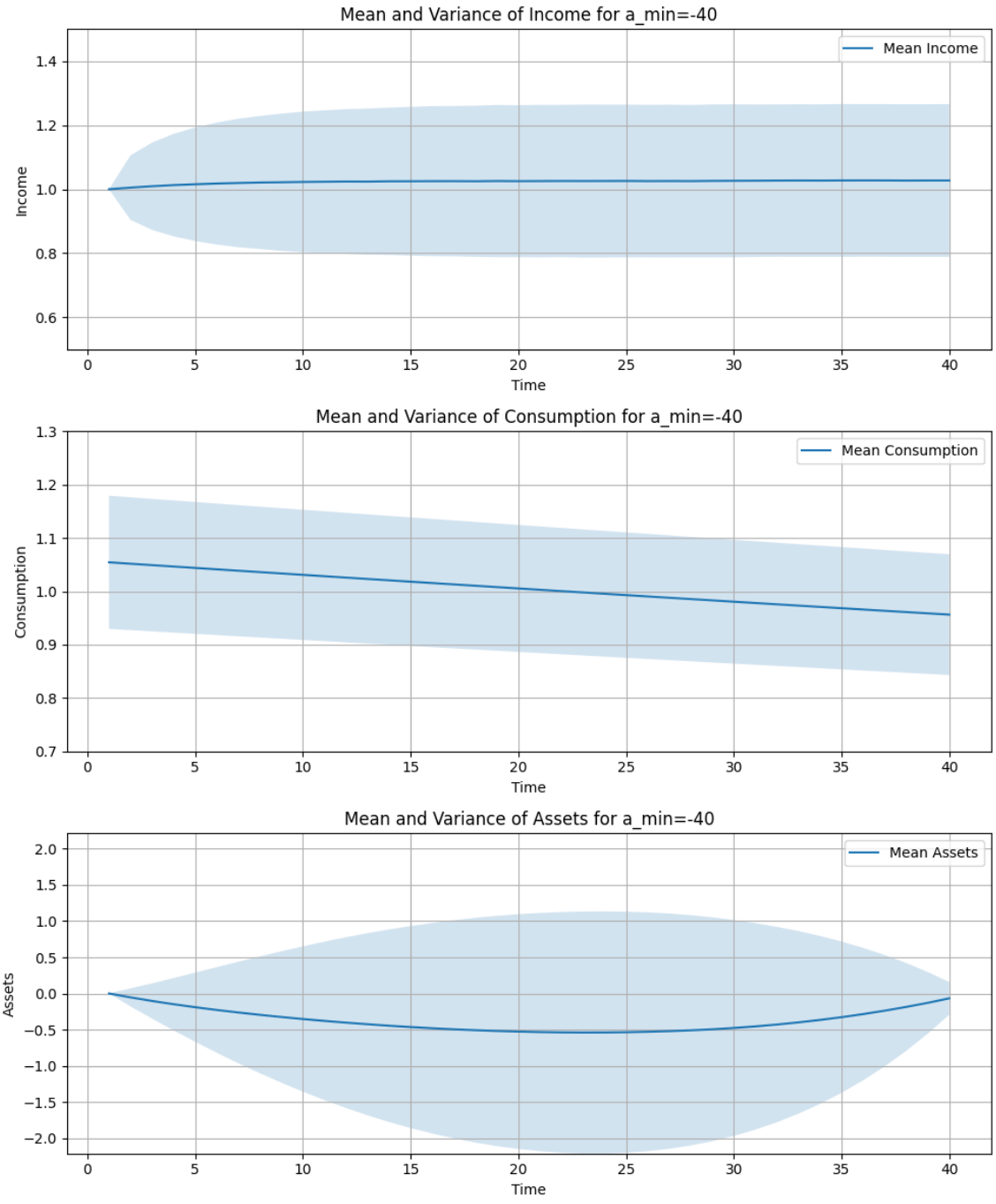

Figure 1: Mean and Variance of Income, Consumption, and Assets for $a_{\min} = -40$

From the Figures 1 and 2, we can observe that in a ABHI model with low borrowing constraints, households have greater flexibility to smooth their consumption over time, resulting in a smoother consumption path. This behavior reflects optimal consumption smoothing, where consumption would generally increase if the income process has a positive drift or remains relatively stable otherwise. Households can borrow against future income, leading to significant negative asset positions, especially in the early periods where borrowing is utilized to maintain consumption. As the time horizon progresses, assets gradually move towards zero to meet the terminal condition$a_{T+1} = 0$. This flexibility allows households to achieve higher utility by better smoothing consumption over time.

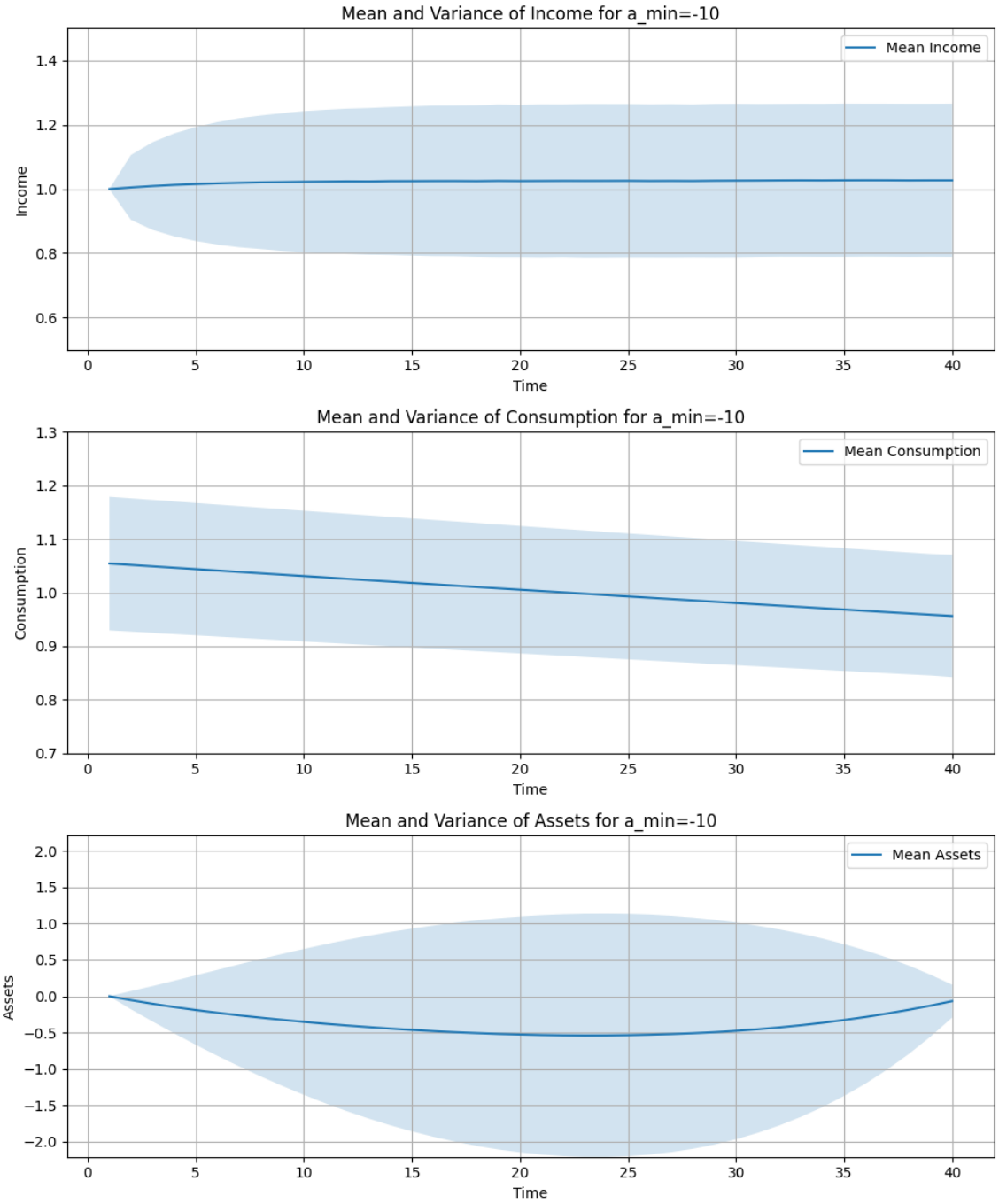

Figure 2: Mean and Variance of Income, Consumption, and Assets for $a_{\min} = -10$

For $a_{\min} = -10$, the mean and variance of income, consumption, and assets show similar trends but with less borrowing compared to $a_{\min} = -40$.

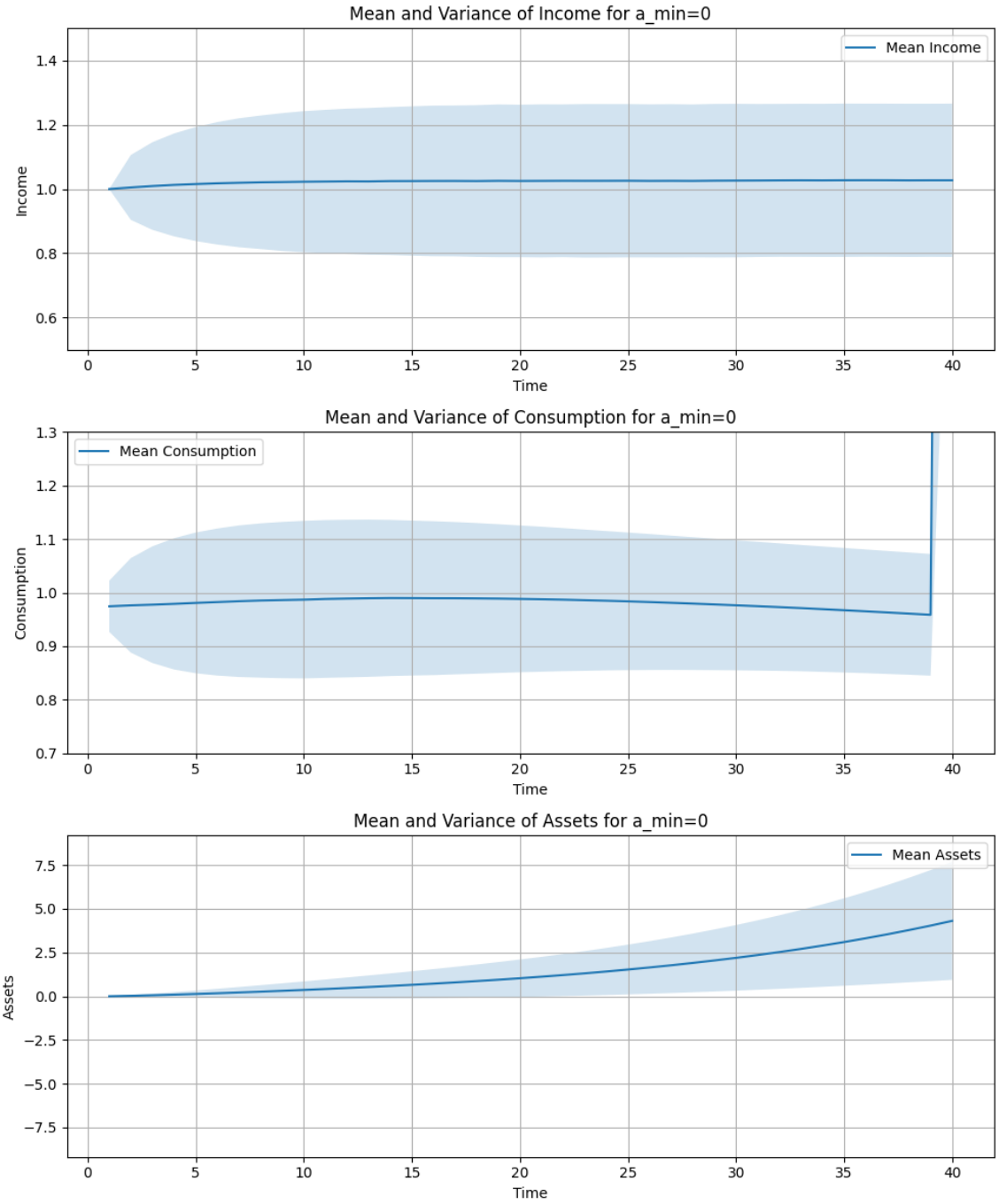

Figure 3: Mean and Variance of Income, Consumption, and Assets for $a_{\min} = 0$

When $a_{\min} = 0$, the agent cannot borrow at all. This scenario shows the highest mean asset levels, as the agent saves to buffer against income shocks. The variance of assets increases over time, reflecting accumulated savings. However, the existence of remaining assets at $t = 40$ suggests that the optimal consumption path under occasionally binding constraints may not fully achieve the best saving strategy. This indicates potential inefficiencies in the saving and consumption decisions. I suggests the reason these phenomena occur is that, under my original decision rule, agents tend to borrow money to maintain their consumption level for optimal utility. However, in this case, agents are unable to borrow. Consequently, in later periods, when income sometimes exceeds the optimal consumption level, agents save the excess rather than withdrawing it to consume.

To address this, I employ a numerical approach to iteratively simulate the consumption and asset policy functions. This method allows for the depiction of the dynamic behavior of agents under these constraints, ensuring that the consumption path better aligns with theoretical optimality while considering the impact of borrowing limits and other occasionally binding constraints.

4.1 Variance of Income, Consumption, and Assets

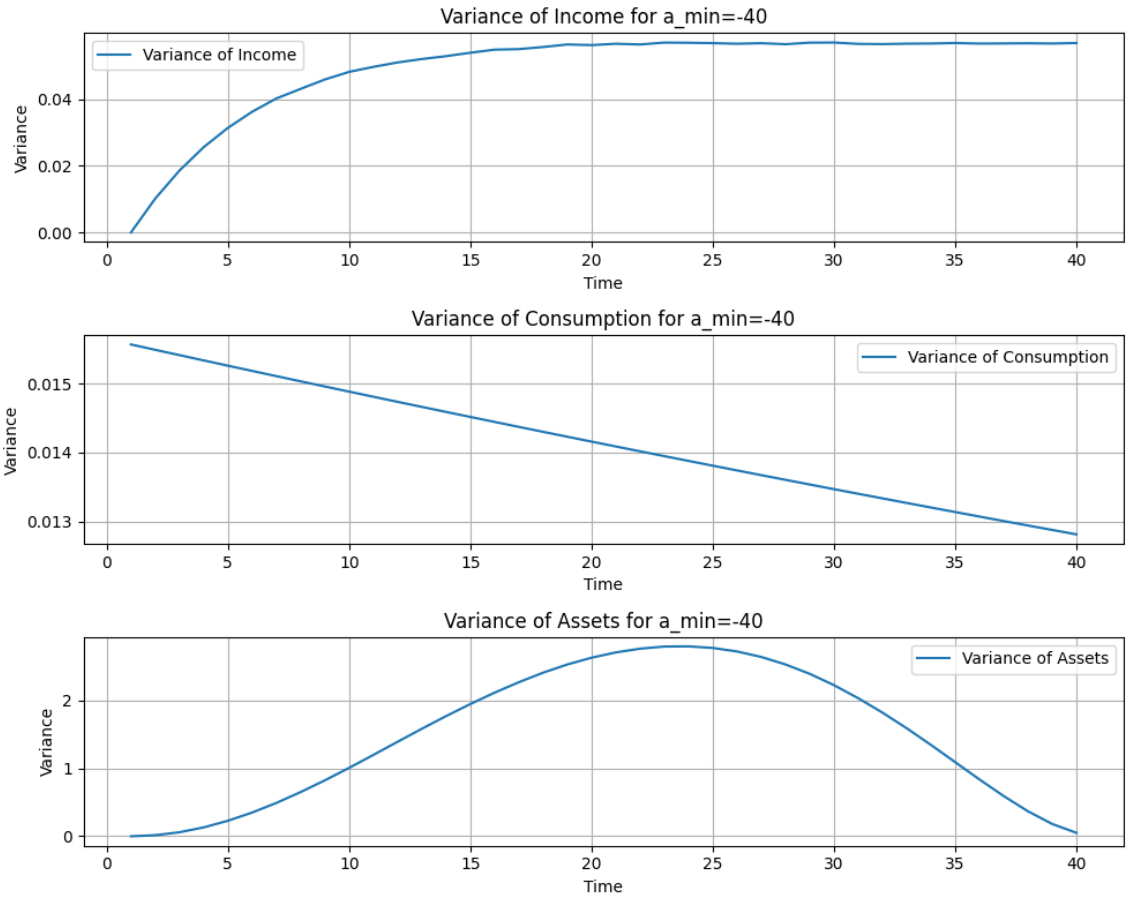

Figure 4: Variance of Income, Consumption, and Assets for $a_{\min} = -40$

The variance plots for $a_{\min} = -40$ highlight the dynamics of income, consumption, and assets. The variance of income increases initially but stabilizes. The variance of consumption decreases slightly but remains steady. Asset variance increases initially as borrowing accumulates but decreases as the agent approaches the terminal period.

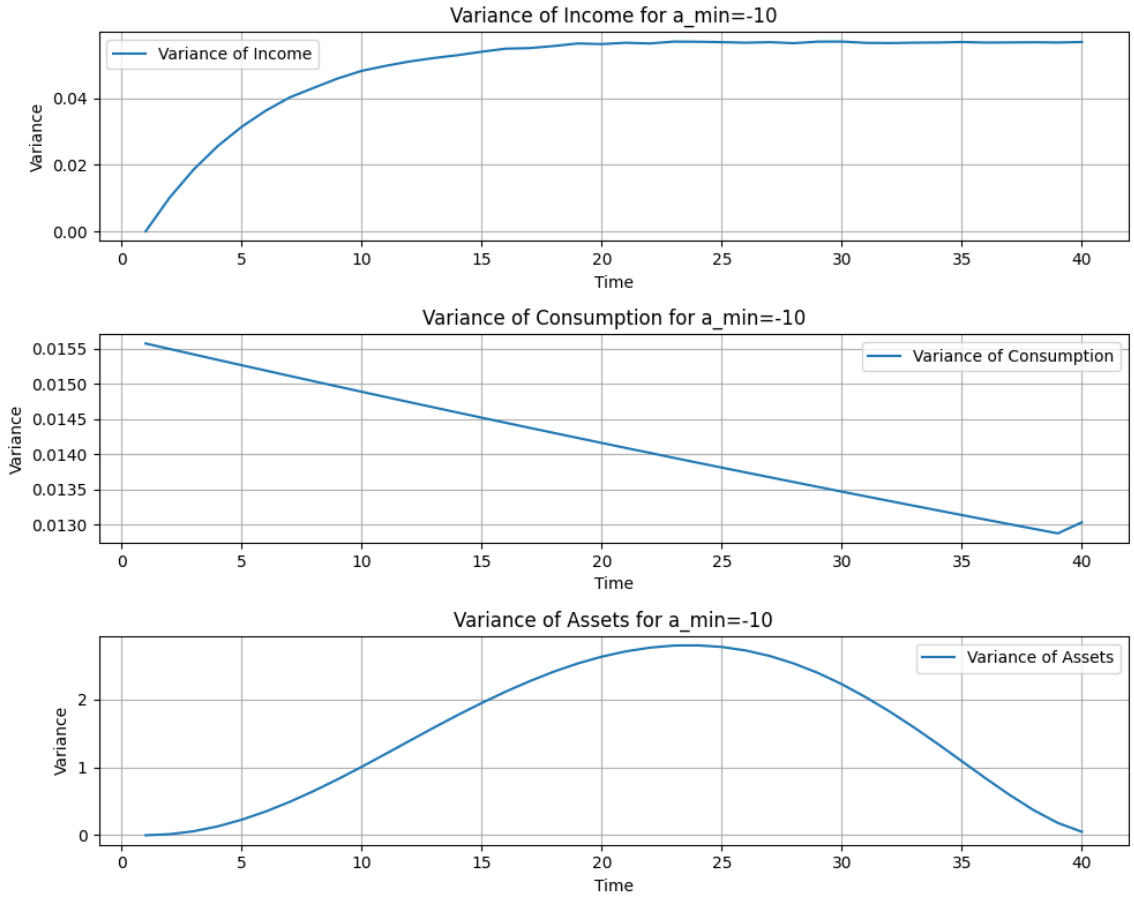

Figure 5: Variance of Income, Consumption, and Assets for $a_{\min} = -10$

For $a_{\min} = -10$, the variance of consumption and assets shows similar patterns with $a_{min} = -40$.We observe that the paths of consumption and assets under the two different minimum constraints are quite similar. This similarity suggests that the constraint is low enough to allow agents to smooth their consumption by borrowing without hitting the borrowing limit. In other words, the less stringent constraint provides more flexibility for agents to optimize their consumption and savings decisions, thus maintaining a relatively stable consumption path even in the presence of income fluctuations.

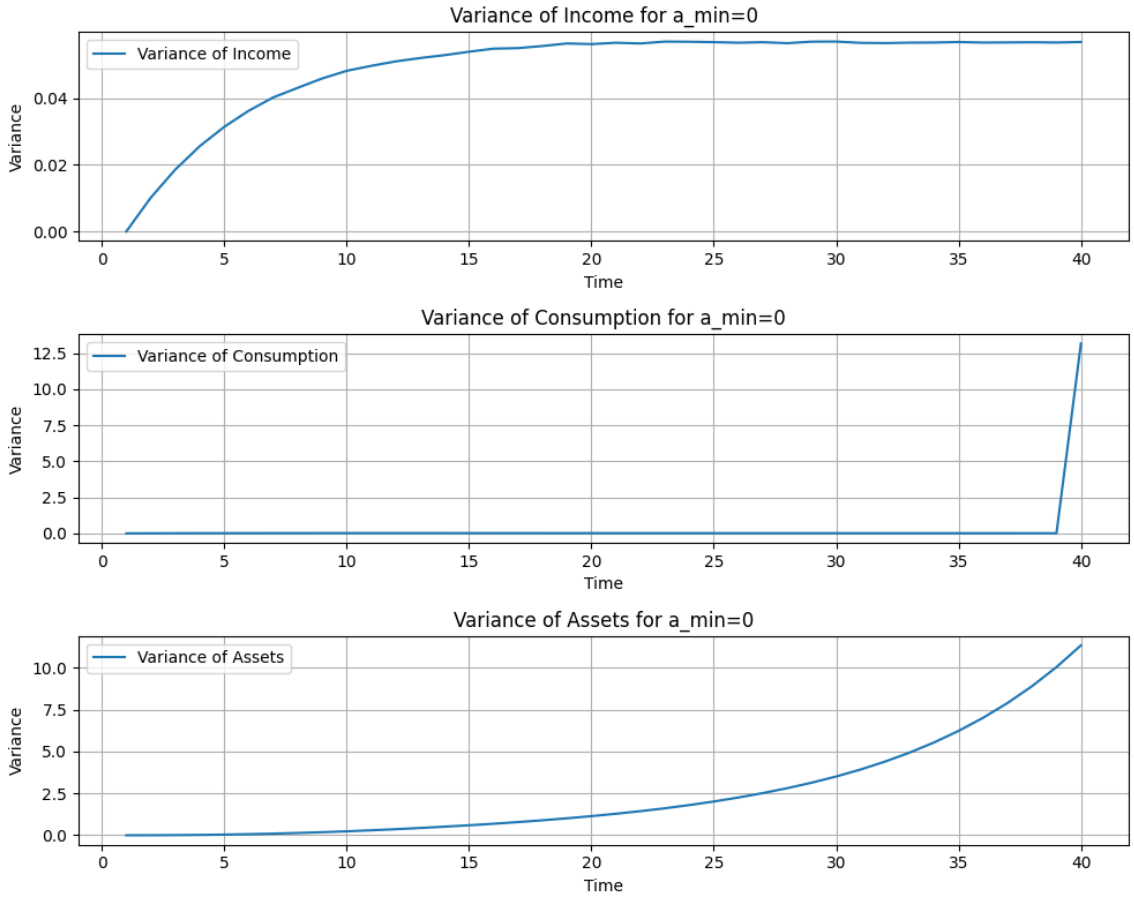

Figure 6: Variance of Income, Consumption, and Assets for $a_{\min} = 0$

With $a_{\min} = 0$, the variance of assets is the highest due to the accumulation of savings without borrowing. The variance of consumption increases significantly towards the end of the period due to the inability to smooth consumption through borrowing.

4.2 Numerical Optimization with Budget Constraints

To solve the issue above, I employed numerical methods to determine the optimal consumption path over a finite horizon. The approach involves solving a constrained optimization problem to maximize the household's utility. Specifically, I used Python's Scipy package with the Sequential Least-Squares Quadratic Programming (SLSQP) algorithm to minimize the utility function while ensuring the budget constraint, non-negative consumption, and a minimum asset constraint are satisfied. It is a value iteration process to find the optimal consumption path, and it took 518.62 seconds to finish the stimulation for $a_{min} = 0$.

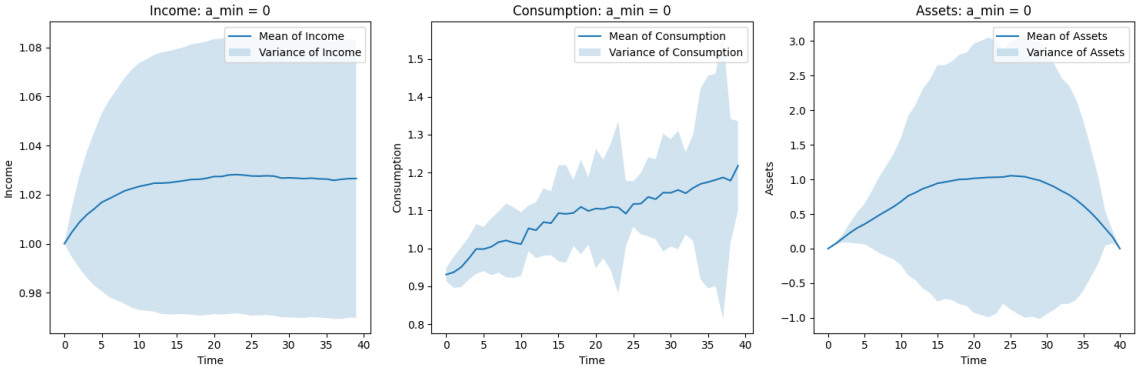

Figure 7: Mean and Variance of Income, Consumption, and Assets for $a_{\min} = 0$ using numerical optimization

The result showing that agents tend to save early and spend later with a no-borrowing constraint aligns with established economic theories, particularly the life-cycle hypothesis, and intertemporal consumption theory. The life-cycle hypothesis suggests that individuals plan their consumption and savings behavior over their lifetime to smooth consumption, saving during their working years to prepare for retirement. Intertemporal consumption theory explains that consumers aim to smooth consumption over time, saving when income is high to buffer against periods of lower income, such as retirement. The presence of a borrowing constraint, which prevents individuals from borrowing against future income, reinforces the need for early savings, and it is also known as precautionary saving.

5 Potential Enhancements & Future Development

In this section, I suggest some possible ways to improve the project and propose new directions for the ABHI model.

5.1 The Expectation of Future Income

The expectation of income serves as an essential basis for the decision-making process of agents. For accuracy and to closely resemble real-world scenarios, considering the unrealized y_{t+1} as $E[y_{t+1}|y_t]$ may be crucial. Below is the derivation.

$$E[\ln(y_{t+1})|y_t] = E[\rho \ln(y_t) + \epsilon_t|y_t]$$

Since $\rho \ln(y_t)$ is deterministic and $E[\epsilon_t|y_t] = 0$:

$$E[\ln(y_{t+1})|y_t] = \rho \ln(y_t)$$

Now, we need to find the conditional expectation of y_{t+1}:

$$E[y_{t+1}|y_t] = E[\exp(\ln(y_{t+1}))|y_t]$$

Using $\ln(y_{t+1}) = \rho \ln(y_t) + \epsilon_t$:

$$E[y_{t+1}|y_t] = E[\exp(\rho \ln(y_t) + \epsilon_t)|y_t]$$

Since ϵ_t is normally distributed, $\exp(\epsilon_t)$ follows a log-normal distribution, and we have:

$$E[\exp(\epsilon_t)] = \exp\left(\frac{\sigma^2}{2}\right)$$

Therefore:

$$E[y_{t+1}|y_t] = \exp(\rho \ln(y_t))E[\exp(\epsilon_t)]$$

$$E[y_{t+1}|y_t] = y_t^{\rho} \exp\left(\frac{\sigma^2}{2}\right)$$

5.2 Value Function & Policy Function

The value function at the terminal period is given by:

$$V_{T+1}(a, y) = 0 \tag{22}$$

For each time period $t = T, T-1, \ldots, 1$:

$$V_t(a, y) = \max_{a'} \left[\ln((1+r)a + y - a') + \beta \mathbb{E}[V_{t+1}(a', y')]\right] \tag{23}$$

where a' is the asset level in the next period, and y' is the income in the next period.

For each state (a, y):

$$\pi_t(a, y) = \arg\max_{a'} \left[\ln((1+r)a + y - a') + \beta \mathbb{E}[V_{t+1}(a', y')]\right] \tag{24}$$

where $\pi_t(a, y)$ is the policy function that determines the optimal asset choice for the next period. To solve the model, we first discretize the state space by creating a grid for assets a and income y. We then initialize the value function by setting $V_{T+1}(a, y) = 0$ for all (a, y). Using backward induction, for each time period $t = T$ to $t = 1$, and for each a and y on the grid, we solve the optimization problem to find $V_t(a, y)$ and $\pi_t(a, y)$. Finally, using forward simulation, given the initial conditions $a_1 = 0$ and $y_1 = 1$, we simulate the paths of a_t, y_t, and c_t using the policy function $\pi_t(a, y)$.

To simulate the paths of income, consumption, and assets, we initialize each path $i = 1, 2, \ldots, 100000$ with $y_1 = 1$ and $a_1 = 0$. For each time period $t = 1, 2, \ldots, T$, we generate income using the process $\ln(y_{t+1}) = \rho \ln(y_t) + \epsilon_t$, where $\epsilon_t \sim N(0, \sigma^2)$. We then compute consumption and savings (c_t, a_{t+1}) based on the policy function, update the state (a_{t+1}, y_{t+1}), and record the income, consumption, and asset data.

5.3 Leveraging CVXPY for Enhanced Model Implementation

CVXPY's ability to handle convex optimization problems makes it an ideal tool for solving the household optimization problem under study. The utility function, which is a logarithmic function, is convex, and the budget constraints are linear, both of which are efficiently managed by CVXPY. Additionally, CVXPY supports dynamic optimization of variables over multiple periods, which aligns perfectly with the multi-period utility maximization problem presented. The library's flexibility in handling constraints, such as the non-negativity of assets, further ensures accurate and efficient solutions. Moreover, CVXPY's compatibility with Python's visualization libraries like Matplotlib allows for the clear and comprehensive visual representation of the model's results, such as the mean and variance of income, consumption, and assets over time. To enhance the model's future development, exploring different parameter settings and scenarios can provide deeper insights. Furthermore, incorporating more complex stochastic processes and constraints can make the model more robust and applicable to a wider range of economic analyses.

5.4 Endogenous Labor Supply

Introducing endogenous labor supply decisions into the ABHI model allows for a more realistic analysis of how individuals adjust their work efforts in response to changes in income risk and borrowing constraints. By incorporating labor supply choices, the model can better capture how people manage income fluctuations and optimize their economic well-being. This enhancement provides insights into how individuals balance work and leisure, respond to policy changes, accumulate wealth, and make life-cycle decisions, ultimately leading to a more comprehensive understanding of household behavior in the face of economic uncertainty.

5.5 Income Distribution Inequality

By considering individual heterogeneity and market imperfections, the ABHI model can reveal how different income groups behave under uncertainty and constraints, and how these behaviors lead to and exacerbate income inequality.

- **Heterogeneous Income Processes:** In the ABHI model, individuals' incomes are typically modeled as stochastic processes (such as AR(1) processes), reflecting real-world variations in income fluctuations and levels across different individuals.

- **Initial Wealth Disparities:** Differences in initial wealth among individuals influence their lifetime consumption and savings behaviors, leading to long-term wealth inequality.
- **Budget Constraints:** Individuals must adhere to budget constraints in each period, meaning that consumption and savings cannot exceed their current income and previously accumulated wealth. This constraint dictates their consumption and saving decisions.
- **Asset Accumulation:** Due to market imperfections (e.g., inability to fully insure against income risk), individuals accumulate assets based on expected income fluctuations. These assets serve as self-insurance. However, the ability to accumulate assets varies across income levels, leading to unequal wealth distribution.
- **Income Risk:** The risk and uncertainty individuals face regarding their income, combined with market imperfections (such as borrowing constraints), result in differing consumption and saving behaviors. Higher-income risk generally increases the motivation to save, but lower-income individuals may struggle to accumulate sufficient wealth.
- **Market Imperfections:** In imperfect markets, individuals cannot fully share risks through the market, leading to greater reliance on self-insurance, which further exacerbates wealth inequality.

5.6 Monetary Policy Rules and Redistributive Effects

By adjusting interest rates, policymakers can influence aggregate demand and supply, and the ABHI model helps elucidate the trade-offs involved in these decisions. For instance, increasing interest rates can dampen inflation but may also slow economic growth, while lowering rates can boost consumption and investment but risk higher inflation. Additionally, the model highlights the redistributive effects of interest rate changes. High interest rates benefit savers by providing higher returns on savings, whereas low interest rates favor borrowers by reducing the cost of borrowing. This differential impact can be quantified across different segments of the population, allowing policymakers to assess the broader implications of their monetary policy decisions on income and wealth distribution.

6 Coding Sample

```python
import numpy as np
import matplotlib.pyplot as plt
import time

############################ mathematical ############################
def solve_income(y1, T, rho, sigma): # yt path
    yt = np.empty(T)
    yt[0] = y1
    epsilon_t = np.random.normal(0, sigma, T - 1)
    for t in range(1, T):
        yt[t] = yt[t - 1] ** rho * np.exp(epsilon_t[t - 1])
    return yt

def ct_star(beta, r, yt, t, T): # ct_star path
    numerator = beta ** (t - 1) * (1 - beta)
    denominator = (1 - beta ** T)
    sum_term = sum((1 + r) ** (t - s) * yt[s - 1] for s in range(1, T + 1))
    return numerator / denominator * sum_term

def decision_process(beta, r, yt, T, a_min):
    at = np.zeros(T)
    ct_star_values = np.zeros(T)

    for t in range(1, T):
        ct_star_values[t - 1] = ct_star(beta, r, yt, t, T)
```

```
        at[t] = (1 + r) * at[t - 1] + yt[t - 1] - ct_star_values[t - 1]
        if at[t] < a_min:
            at[t] = a_min
            ct_star_values[t - 1] = (1 + r) * at[t - 1] + yt[t - 1] - a_min

    ct_star_values[T - 1] = (1 + r) * at[T - 1] + yt[T - 1]
    return at, ct_star_values

def utility(ct_star, beta, T):
    return np.sum([beta ** (t + 1) * np.log(ct_star[t]) for t in range(T)])

# Parameters
r = 0.05
beta = 0.95
T = 40
a_min_values = [-40, -10, 0]
rho = 0.90
sigma = 0.1
y1 = 1
n_simulations = 100000

start_time = time.time()

# Simulation
yt_simulations = [solve_income(y1, T, rho, sigma) for _ in range(n_simulations)]
results = {a_min: {'yt': yt_simulations, 'ct': [], 'at': []} for a_min in
↪  a_min_values}

for a_min in a_min_values:
    for yt in yt_simulations:
        at, ct = decision_process(beta, r, yt, T, a_min)
        results[a_min]['ct'].append(ct)
        results[a_min]['at'].append(at)

# Calculate means and variances
mean_results = {a_min: {'yt': [], 'ct': [], 'at': []} for a_min in a_min_values}
var_results = {a_min: {'yt': [], 'ct': [], 'at': []} for a_min in a_min_values}

for a_min in a_min_values:
    mean_results[a_min]['yt'] = np.mean(results[a_min]['yt'], axis=0)
    mean_results[a_min]['ct'] = np.mean(results[a_min]['ct'], axis=0)
    mean_results[a_min]['at'] = np.mean(results[a_min]['at'], axis=0)
    var_results[a_min]['yt'] = np.var(results[a_min]['yt'], axis=0)
    var_results[a_min]['ct'] = np.var(results[a_min]['ct'], axis=0)
    var_results[a_min]['at'] = np.var(results[a_min]['at'], axis=0)

# Plotting Mean and Variance
for a_min in a_min_values:
    plt.figure(figsize=(10, 12))

    x_values = range(1, T + 1)

    # Finding y-axis limits for consistent and symmetric scaling
    y_max = max(np.max(mean_results[a_min]['yt'] +
    ↪  np.sqrt(var_results[a_min]['yt'])),
                np.max(mean_results[a_min]['ct'] +
                ↪  np.sqrt(var_results[a_min]['ct'])),
                np.max(mean_results[a_min]['at'] +
                ↪  np.sqrt(var_results[a_min]['at'])))
```

```
82
83      y_min = min(np.min(mean_results[a_min]['yt'] -
        ↪  np.sqrt(var_results[a_min]['yt'])),
84                  np.min(mean_results[a_min]['ct'] -
                    ↪  np.sqrt(var_results[a_min]['ct'])),
85                  np.min(mean_results[a_min]['at'] -
                    ↪  np.sqrt(var_results[a_min]['at'])))
86
87      y_limit = max(abs(y_max), abs(y_min))
88
89      plt.subplot(3, 1, 1)
90      plt.plot(x_values, mean_results[a_min]['yt'], label='Mean Income')
91      plt.fill_between(x_values, mean_results[a_min]['yt'] -
        ↪  np.sqrt(var_results[a_min]['yt']),
92                       mean_results[a_min]['yt'] +
                         ↪  np.sqrt(var_results[a_min]['yt']), alpha=0.2)
93      plt.title(f'Mean and Variance of Income for a_min={a_min}')
94      plt.xlabel('Time')
95      plt.ylabel('Income')
96      plt.ylim(0.5, 1.5)
97      plt.legend()
98      plt.grid(True)
99
100     plt.subplot(3, 1, 2)
101     plt.plot(x_values, mean_results[a_min]['ct'], label='Mean Consumption')
102     plt.fill_between(x_values, mean_results[a_min]['ct'] -
        ↪  np.sqrt(var_results[a_min]['ct']),
103                      mean_results[a_min]['ct'] +
                         ↪  np.sqrt(var_results[a_min]['ct']), alpha=0.2)
104     plt.title(f'Mean and Variance of Consumption for a_min={a_min}')
105     plt.xlabel('Time')
106     plt.ylabel('Consumption')
107     plt.ylim(0.7, 1.3)
108     plt.legend()
109     plt.grid(True)
110
111     plt.subplot(3, 1, 3)
112     plt.plot(x_values, mean_results[a_min]['at'], label='Mean Assets')
113     plt.fill_between(x_values, mean_results[a_min]['at'] -
        ↪  np.sqrt(var_results[a_min]['at']),
114                      mean_results[a_min]['at'] +
                         ↪  np.sqrt(var_results[a_min]['at']), alpha=0.2)
115     plt.title(f'Mean and Variance of Assets for a_min={a_min}')
116     plt.xlabel('Time')
117     plt.ylabel('Assets')
118     plt.ylim(-y_limit, y_limit)
119     plt.legend()
120     plt.grid(True)
121
122     plt.tight_layout()
123     plt.show()
124
125 # Plotting Variance
126 for a_min in a_min_values:
127     plt.figure(figsize=(10, 8))
128
129     plt.subplot(3, 1, 1)
130     plt.plot(x_values, var_results[a_min]['yt'], label='Variance of Income')
131     plt.title(f'Variance of Income for a_min={a_min}')
132     plt.xlabel('Time')
```

```
    plt.ylabel('Variance')
    plt.legend()
    plt.grid(True)

    plt.subplot(3, 1, 2)
    plt.plot(x_values, var_results[a_min]['ct'], label='Variance of Consumption')
    plt.title(f'Variance of Consumption for a_min={a_min}')
    plt.xlabel('Time')
    plt.ylabel('Variance')
    plt.legend()
    plt.grid(True)

    plt.subplot(3, 1, 3)
    plt.plot(x_values, var_results[a_min]['at'], label='Variance of Assets')
    plt.title(f'Variance of Assets for a_min={a_min}')
    plt.xlabel('Time')
    plt.ylabel('Variance')
    plt.legend()
    plt.grid(True)

    plt.tight_layout()
    plt.show()

end_time = time.time()
print(f"Runtime: {end_time - start_time} seconds")
```

```
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt
from joblib import Parallel, delayed
import time

##########################  numerical  ############################

def solve_income(y1, T, rho, sigma): #income_path
    yt = np.empty(T)
    yt[0] = y1
    epsilon_t = np.random.normal(0, sigma, T - 1)
    for t in range(T-1):
        yt[t+1] = yt[t]**rho * np.exp(epsilon_t[t])
    return yt

def utility(ct, beta, T):
    return -np.sum([beta ** (t + 1) * np.log(ct[t]) for t in range(T)])

def budget_constraint(ct, yt, r, T):
    at = np.zeros(T + 1)
    for t in range(T):
        at[t + 1] = (1 + r) * at[t] + yt[t] - ct[t]
    return at[-1]  # Ensure terminal condition a_T = 0

def non_negative(ct): #constraint for positive consumption
    return ct

def non_negative_assets(ct, yt, r, a_min, T): #constraint for sufficient assets
    at = np.zeros(T + 1)
    for t in range(T):
        at[t + 1] = (1 + r) * at[t] + yt[t] - ct[t]
    return at[1:] - a_min

```

```python
def run_simulation(y1, r, beta, T, rho, sigma, a_min, c0):
    yt = solve_income(y1, T, rho, sigma)
    constraints = [
        {'type': 'eq', 'fun': lambda ct: budget_constraint(ct, yt, r, T)}, # Budget constraint
        {'type': 'ineq', 'fun': lambda ct: non_negative(ct)}, # ct > 0
        {'type': 'ineq', 'fun': lambda ct: non_negative_assets(ct, yt, r, a_min, T)} # at > a_min
    ]

    result = minimize(utility, c0, args=(beta, T), method='SLSQP', constraints=constraints,
                      bounds=[(0, None) for _ in range(T)], options={'maxiter': 100, 'ftol': 1e-6})

    if result.success:
        optimal_c = result.x
        optimal_a = np.zeros(T + 1)
        for t in range(T):
            optimal_a[t + 1] = (1 + r) * optimal_a[t] + yt[t] - optimal_c[t]
        return yt, optimal_c, optimal_a
    else:
        return None

# Parameters
r = 0.05
beta = 0.95
T = 40
rho = 0.9
sigma = 0.1
y1 = 1
a_min = -10

# Initial guess
c0 = np.ones(T)

n_simulations = 100000

start_time = time.time()

yt_simulations = np.zeros((n_simulations, T))

for i in range(n_simulations):
    yt_simulations[i, :] = solve_income(y1, T, rho, sigma)

mean_yt = np.mean(yt_simulations, axis=0)
var_yt = np.var(yt_simulations, axis=0)

results = Parallel(n_jobs=-1, backend="loky")(
    delayed(run_simulation)(y1, r, beta, T, rho, sigma, a_min, c0) for _ in range(n_simulations))

# Filter out failed optimizations
results = [res for res in results if res is not None]

if results:
    yt_simulations = np.array([res[0] for res in results])
    ct_simulations = np.array([res[1] for res in results])
    at_simulations = np.array([res[2] for res in results])
    print("Utility: ", utility(np.mean(ct_simulations, axis=0), beta, T))
```

```
90
91      mean_ct = np.mean(ct_simulations, axis=0)
92      var_ct = np.var(ct_simulations, axis=0)
93      mean_at = np.mean(at_simulations, axis=0)
94      var_at = np.var(at_simulations, axis=0)
95
96      # Plotting
97      plt.figure(figsize=(15, 5))
98
99      plt.subplot(1, 3, 1)
100     plt.plot(mean_yt, label='Mean of Income')
101     plt.fill_between(range(T), mean_yt - var_yt, mean_yt + var_yt, alpha=0.2,
        ↪  label='Variance of Income')
102     plt.xlabel('Time')
103     plt.ylabel('Income')
104     plt.title(f'Income: a_min = {a_min}')
105     plt.legend()
106
107     plt.subplot(1, 3, 2)
108     plt.plot(mean_ct, label='Mean of Consumption')
109     plt.fill_between(range(T), mean_ct - var_ct, mean_ct + var_ct, alpha=0.2,
        ↪  label='Variance of Consumption')
110     plt.xlabel('Time')
111     plt.ylabel('Consumption')
112     plt.title(f'Consumption: a_min = {a_min}')
113     plt.legend()
114
115     plt.subplot(1, 3, 3)
116     plt.plot(mean_at, label='Mean of Assets')
117     plt.fill_between(range(T + 1), mean_at - var_at, mean_at + var_at, alpha=0.2,
        ↪  label='Variance of Assets')
118     plt.xlabel('Time')
119     plt.ylabel('Assets')
120     plt.title(f'Assets: a_min = {a_min}')
121     plt.legend()
122
123     plt.tight_layout()
124     plt.show()
125
126 end_time = time.time()
127 runtime = end_time - start_time
128 print(f"Finished in {runtime:.2f} seconds")
```