TSP Lecture - Heterogeanous Agent Models

Jamie Lenney

Dec 2025

Code/slides: **https://github.com/s0840389/TSP_HA**

Objectives

- Motivate incomplete markets
- Understand how income risk and borrowing constraints underpin HA model mechanisms
- Know how to solve a HA model
- Examine the properties of a Hugget model

Further reading

- Ljungqvist and Sargent Recursive Macroeconmic Theory chapters: 17,18
- Auclert, A., Bardóczy, B., Rognlie, M., & Straub, L. (2021). Using the sequence-space Jacobian to solve and estimate heterogeneous-agent models. Econometrica, 89(5), 2375-2408.
- **https://github.com/shade-econ/sequence-jacobian**

Incomplete Markets

What is this?

- Can think of a representative agent model as a model where individuals have written contracts to hedge against all risk
  - They are still exposed to aggregate risk
  - Can still have complete markets models with wealth distribution but all agents consumption would move proportionally in the same way.
- When this is not possible we have incomplete markets
  - e.g. There are not sufficient assets/contracts to trade.
- Incomplete markets creates incentives for self-insurance
  - Saving for a 'rainy day'

Consumption Savings Problem

Households(i) maximize lifetime expected utility:

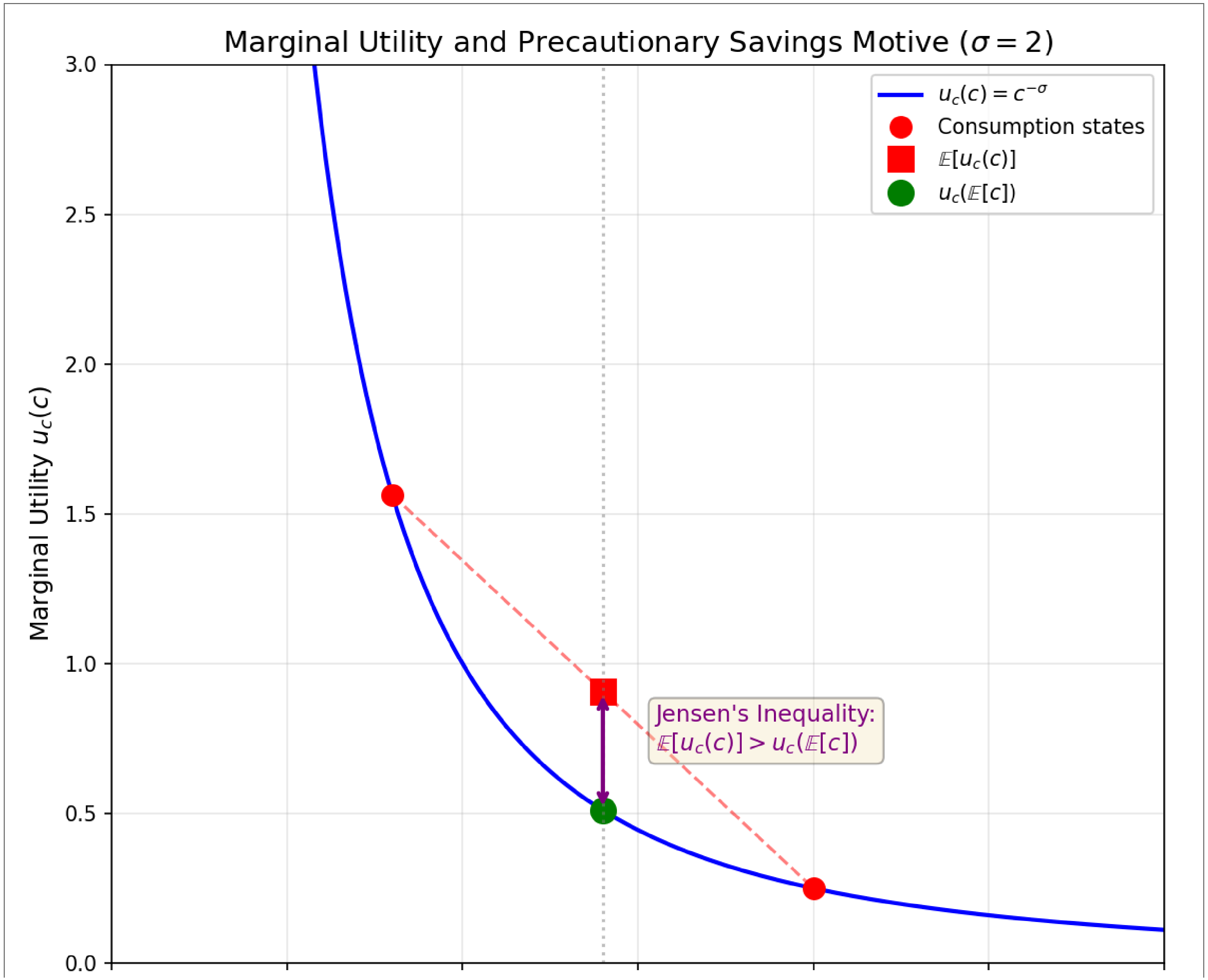
$$\max_{\{c_t\}_{t=0}^{\infty}} \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t)$$

where:

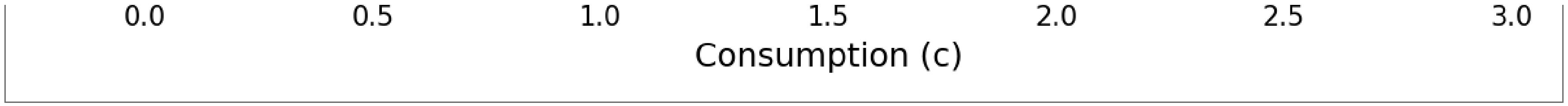
- c_t = consumption at time t
- $\beta \in (0,1)$ = discount factor
- $u(c)$ = instantaneous utility function (typically CRRA $\frac{c^{1-\sigma}}{1-\sigma}$)

Utility function properties

1. **Strictly increasing**: $u'(c) > 0$
    - More consumption is always preferred
2. **Strictly concave**: $u''(c) < 0$
    - Diminishing marginal utility
    - Induces consumption smoothing
3. **Positive third derivative: Prudence**: $u'''(c) > 0$
    - **Prudence** means the marginal utility function is convex.
    - **Key implication**: Creates a **precautionary savings motive**
        - When facing income risk, agents save more as insurance against bad shocks
        - Expected marginal utility exceeds marginal utility of expected consumption: $\mathbb{E}[u'(c')] > u'(\mathbb{E}[c'])$ (Jensen's inequality)
4. **Inada conditions**:
    - $\lim_{c \to 0} u'(c) = \infty$ ensures $c > 0$ always
    - $\lim_{c \to \infty} u'(c) = 0$ ensures interior solutions



Marginal Utility and Precautionary Savings Motive ($\sigma = 2$)
Marginal Utility $u_c(c)$
3.0
2.5
2.0
1.5
1.0
0.5
0.0
$u_c(c) = c^{-\sigma}$
Consumption states
$\mathbb{E}[u_c(c)]$
$u_c(\mathbb{E}[c])$
Jensen's Inequality:
$\mathbb{E}[u_c(c)] > u_c(\mathbb{E}[c])$


0.0
0.5
1.0
1.5
2.0
2.5
3.0
Consumption (c)

Constraints

Budget Constraint

$$c_t + a_{t+1} = (1 + r_t)a_t + \underbrace{z_t w_t}_{y_t}$$

- a_t = assets at beginning of period t
- r_t = interest rate
- $w_t z_t$ = labor income (stochastic)

Borrowing Constraint

$$a_{t+1} \geq \underline{a}$$

- $\underline{a}$ = borrowing limit
- Common cases: $\underline{a} = 0$ (no borrowing) or $\underline{a} < 0$ (some debt allowed)

Natural borrowing constraint

Under worst case scenario need to be able to afford interest payments

$$\underline{a} = \frac{w_{low} z_{low}}{r_{high}}$$

Income Process

Stochastic Income

Labor productivity (income) follows a Markov process:

$$\log z_t = \rho \log z_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$

Can be discretized into:

- $z_t \in \{z_1, z_2, \ldots, z_N\}$ (finite states)
- Transition matrix Π where $\Pi_{ij} = \Pr(z_{t+1} = z_j | z_t = z_i)$

Key feature: Households face **uninsurable idiosyncratic risk**

- Cannot fully smooth consumption
- Need to self-insure through savings

Bellman Equation

Recursive Formulation

$$V(a,y)=\max_{a'\geq \underline{a}}\left\{u(c)+\beta\mathbb{E}[V(a',y')|y]\right\}$$

subject to:

$$c=(1+r)a+y-a'$$

where:

- $V(a,y)$ = value of having assets a and income y today
- a' = assets tomorrow (choice variable)
- $\mathbb{E}[V(a',y')|y]=\sum_{y'}\Pi(y'|y)V(a',y')$
- **State variables**: (a,y)
- **Choice variables**: (c)

First Order Conditions

Lagrangian

$$\mathcal{L} = u(c) + \beta \mathbb{E}[V(a')] + \lambda(a' - \underline{a})$$

Derivative of $\mathcal{L}$ wrpt to a'

$$0 = -u_c(c) + \beta \mathbb{E}[V_a(a', y')] + \lambda$$

Derivative of $V(a, y)$ wrpt to a

$$V_a(a, y) = \beta(1 + r)\mathbb{E}[V_a(a', y')] + (1 + r)\lambda$$

Putting togethor we have

$$V_a(a, y) = (1 + r)u_c(c)$$

And our Euler equation

$$u_c(c) = \beta \mathbb{E}[(1 + r')u_c(c')] + \lambda$$

Complementary Slackness

1. $\lambda \geq 0$
2. $a' - \underline{a} \geq 0$
3. $\lambda(a' - \underline{a}) = 0$

- Binding constraint ($\lambda > 0$)
  - $u_c(c) = \beta\mathbb{E}[(1 + r')u_c(c')] + \lambda$
  - implies consumption is below what agent would want
- Otherwise
  - $u_c(c) = \beta\mathbb{E}[(1 + r')u_c(c')]$

Value Function Iteration (VFI)

- Goal: solve household problem by iterating on the Bellman equation directly to find the value function and policy functions.
- Core idea
  - For each state (productivity z, assets a), maximize over all possible asset choices a' to find the value function.
  - Bellman equation:

$$V^j(z,a) = max_{a'} u(c) + \beta E[V^{j-1}(z',a')|z]$$

  - find best choice, update value function and continue until it converges $\|V^j - V^{j-1}\| \leq \epsilon$

Value Function Iteration (VFI)

- Algorithm (one backward step)
    1. For each (z, a) state, construct V(z,a,a') value for each possible a' choice on grid
    2. Calculate value: $V = u(c) + \beta E[V_{t+1}(z', a')]$
    3. Find optimal choice: $a* = \max_{a'} V$
    4. Store $V(z, a) = \max V$ and policy $a(z, a) = a*$

Value Function Iteration (VFI)

- Advantages
  - Simple and robust;
  - No need for invertible Euler equation
  - Can handle non-convexities, occasionally binding constraints
- Limitations / notes
  - Computationally expensive: $O(nZ \times nA^2)$ per iteration
  - Policy functions can be jagged/non-smooth with coarse grids
  - Requires fine grids ($nA = 2000+$) for smooth policies
  - Much slower than other methods (e.g. EGM) for standard models

In [3]:

```
cali={} # calibration

# household
cali['beta']=0.98 # patience
cali['sigma']=2 # risk aversion
cali['eis']=1/cali['sigma'] # intertemporal elasticity of substitution
cali['bc']=-0.2
cali['borwedge']=0.0

cali['rho_z']=0.97
cali['sd_z']=0.6

# aggregate economy
cali['r']=0.04/4
cali['w']=1.0
cali['govT']=0

# discretization
cali['nA']=2000
cali['nZ']=5
cali['amax']=125
```

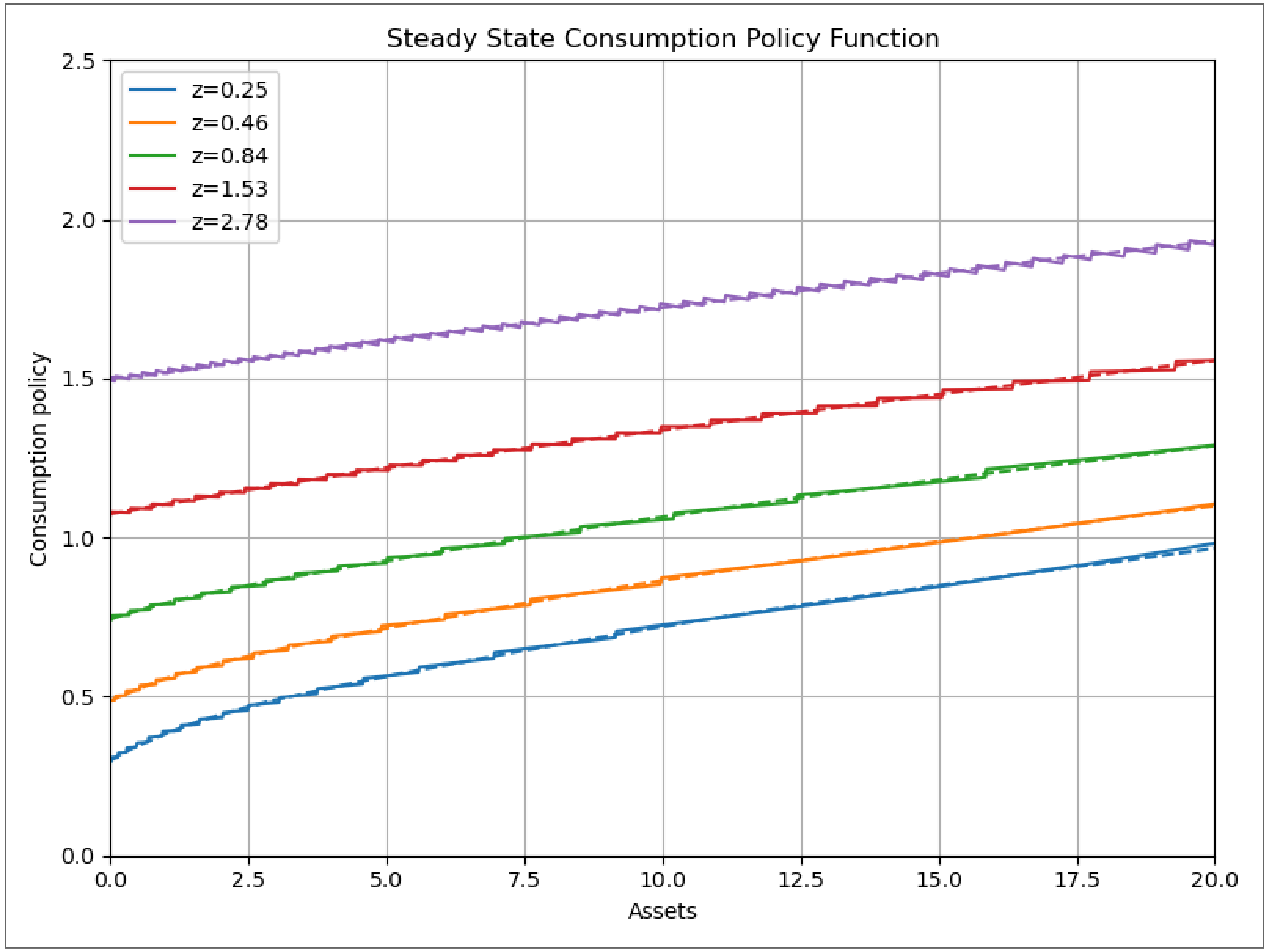
In [4]:

```
## Household problem

def make_grid(bc, amax, nA,nZ,rho_z,sd_z):

    z_grid, pi_z, Pi = grids.markov_rouwenhorst(rho=rho_z, sigma=sd_z, N=nZ)
    a_grid = utilities.discretize.nonlinspace(amax,nA ,1.2, amin = bc) # normalized grid between z

    return z_grid, pi_z, Pi, a_grid


def income(w,r,borwedge,a_grid,z_grid,govT):

    rexpand=r*np.ones_like(a_grid)
    rexpand[a_grid<0]=r+borwedge
    coh = (1+rexpand)*a_grid + w*z_grid[:, np.newaxis]+govT

    return coh


def household_init( r, coh, eis): # initial guess for marginal value function
    Va = (1 + r) * (0.1 * coh) ** (-1 / eis)
```

In [5]:

```
def household_vfit_init( r, coh, eis): # initial guess for marginal value function
    V = (0.1 * coh)**(1-1/eis)/(1-1/eis)  # utility level
    return V

@het(exogenous='Pi', policy='a', backward='V', backward_init=household_vfit_init)
def household_vfit(V_p, a_grid, r, coh, beta, eis,nZ):
    """Single backward iteration step using endogenous gridpoint method for households with CRRA u

    Parameters
    ----------
    V_p     : array (nE, nA), expected  value function next period
    a_grid   : array (nA), asset grid
    z_grid   : array (nE), productivity grid
    r        : scalar, ex-post real interest rate
    w        : scalar, wage
    beta     : scalar, discount factor
    eis      : scalar, elasticity of intertemporal substitution

    Returns
    ----------
    V : array (nE, nA), value function today
```

Consumption Functions



Endogenous Gridpoint Method (EGM)

- Goal: solve household Euler equation without repeated root-finding at every asset grid point — faster and smoother than brute-force VFI.
- Core idea
  - Use Euler equation to compute optimal consumption on the *next-period* grid of assets (endogenous grid), then recover the implied current assets.
  - For CRRA utility: $u_c(c) = c^{-\sigma}$.
  - Euler gives $u_c(c_t) = \beta(1+r)E[u_c(c_{t+1})]$ ⇒ $c_t = [\beta(1+r)E[u_c(c_{t+1})]]^{\frac{-1}{\sigma}}$

Endogenous Gridpoint Method (EGM)

- Algorithm (one backward step)
  1. For each (z,a') Compute expected marginal utility next period:
     $m = \beta E[(1+r')u'(c(z',a'))]$.
     - $m(z,a') = \beta \sum_{z'} \Pi(z'|z)(1+r')u'(c(z,a'))$
  2. Invert marginal utility to get c at each next-period asset grid point: $c(z,?) = m^{\frac{-1}{\sigma}}$.
  3. Compute implied current assets: $a = \frac{a'+c-y}{1+r}$
     - given I choose a', when i have z, it implies I have a today
  4. Interpolate policy ($a \to c$) from the endogenous a_t back onto the exogenous asset grid.
  5. Enforce borrowing constraint: where implied a' < $\underline{a}$ set $a' = \underline{a}$ and recompute $c = (1+r)a + y - a'$.

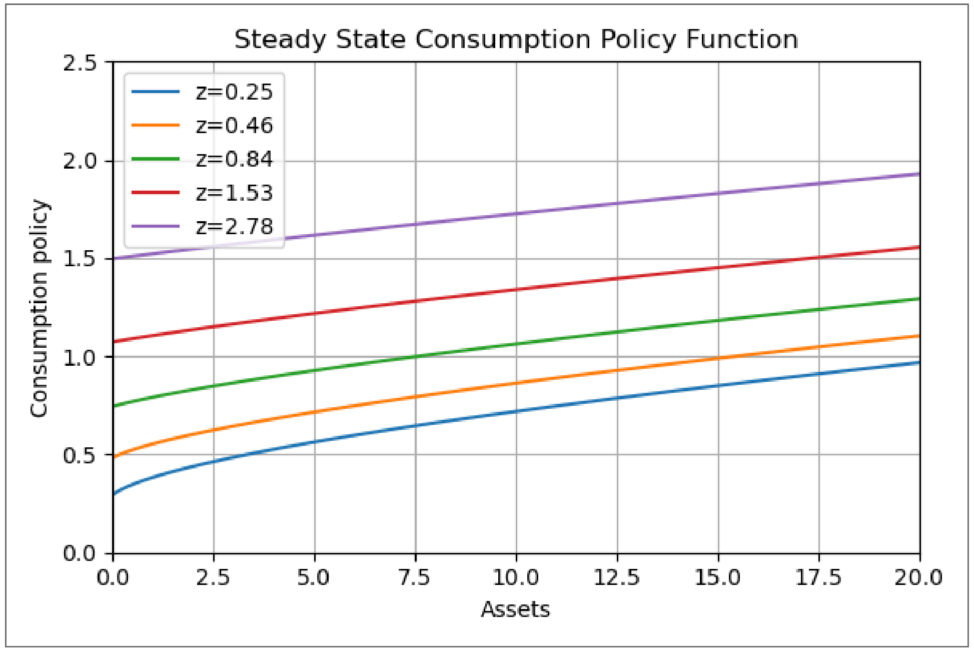
Endogenous Gridpoint Method (EGM)

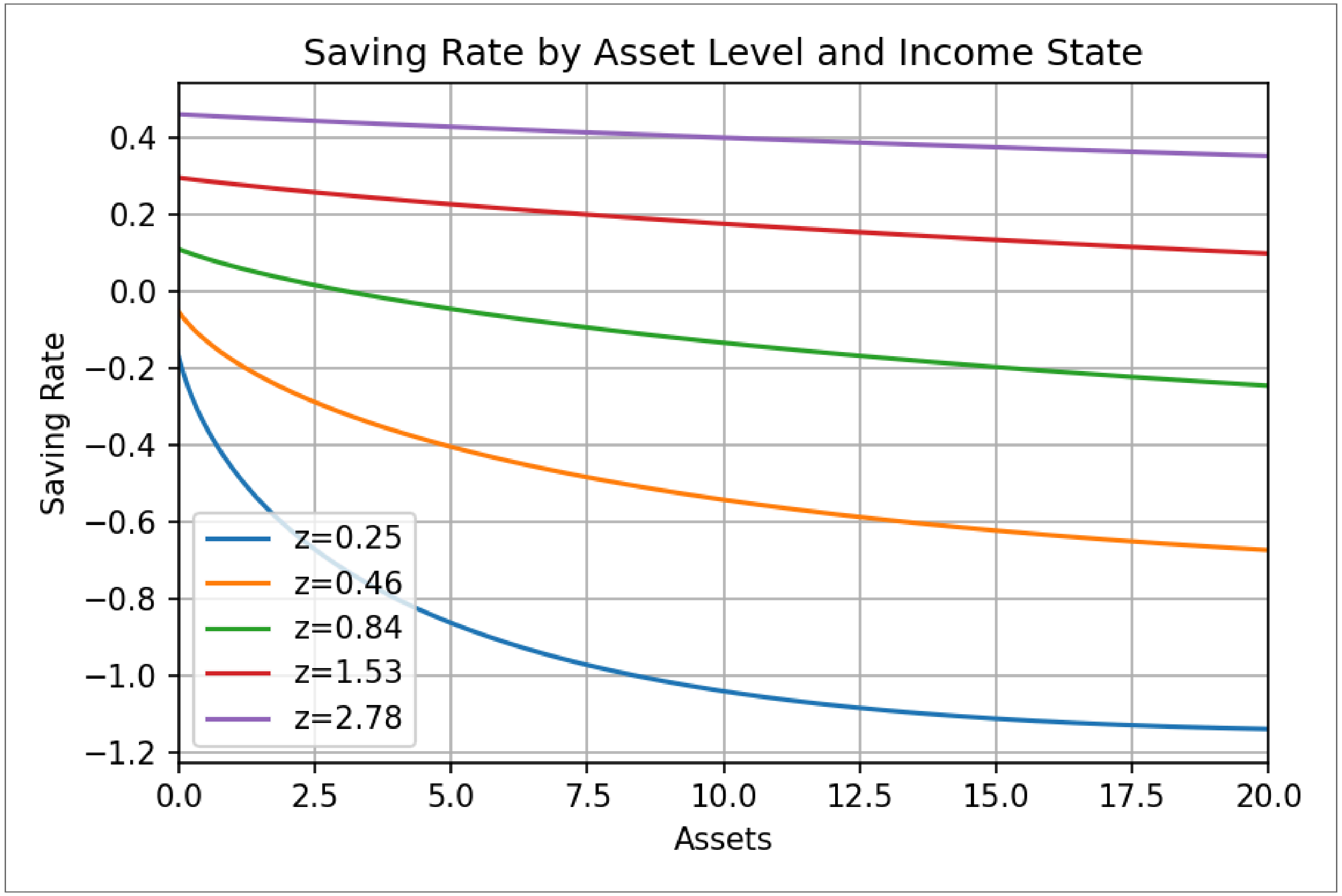
- Advantages
  - No inner-loop maximization; vectorizable and much faster.
  - Produces smooth consumption/savings policies when monotonicity holds.
- Limitations / notes
  - Requires invertible Euler equation (works easily with CRRA).

In [7]:

```
@het(exogenous='Pi', policy='a', backward='Va', backward_init=household_init)
def household(Va_p, a_grid, r, coh, beta, eis,nZ):
    """Single backward iteration step using endogenous gridpoint method for households with CRRA u

    Parameters
    ----------
    Va_p     : array (nE, nA), expected marginal value of assets next period
    a_grid   : array (nA), asset grid
    z_grid   : array (nE), productivity grid
    r        : scalar, ex-post real interest rate
    w        : scalar, wage
    beta     : scalar, discount factor
    eis      : scalar, elasticity of intertemporal substitution

    Returns
    ----------
    Va : array (nE, nA), marginal value of assets today
    c  : array (nE, nA), consumption policy today
    a  : array (nE, nA), asset policy today [a']
    """
```

Consumption Function (EGM)




Saving Rate by Asset Level and Income State
Saving Rate
Assets
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
z=0.25
z=0.46
z=0.84
z=1.53
z=2.78

Forward Iteration: Computing the Distribution

Given policy functions, we need to find the stationary distribution of households over (z, a) states.

Key objects:

- Policy function: $a'(z, a)$ — optimal asset choice
- Transition matrix for income: $\Pi(z'|z)$ — probability of $z \to z'$
- Distribution: $D_t(z, a)$ — mass of households at state (z, a) at time t

Forward iteration equation:

$$D_{t+1}(z', a') = \sum_{z,a} D_t(z, a) \cdot 1[a'(z, a) = a'] \cdot \Pi(z'|z)$$

In words: The mass at (z', a') tomorrow equals the sum of all households who:

1. Choose a' as their savings today
2. Transition from z to z' with probability $\Pi(z'|z)$

Matrix representation: We can write this as a **large transition matrix** Λ where:

$$D_{t+1} = \Lambda D_t$$

where D_t is vectorized over all (z, a) states.

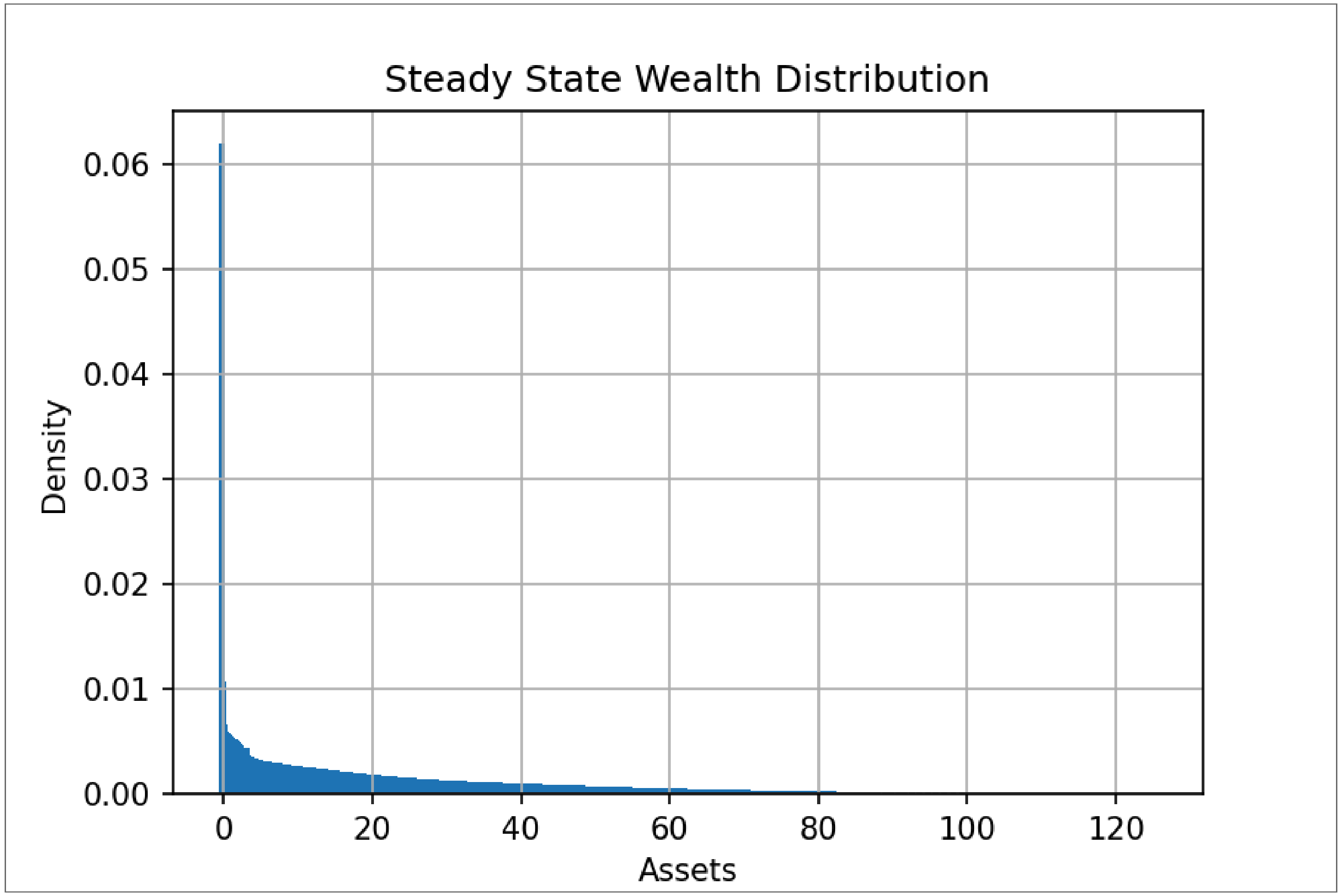
Finding the Ergodic Distribution

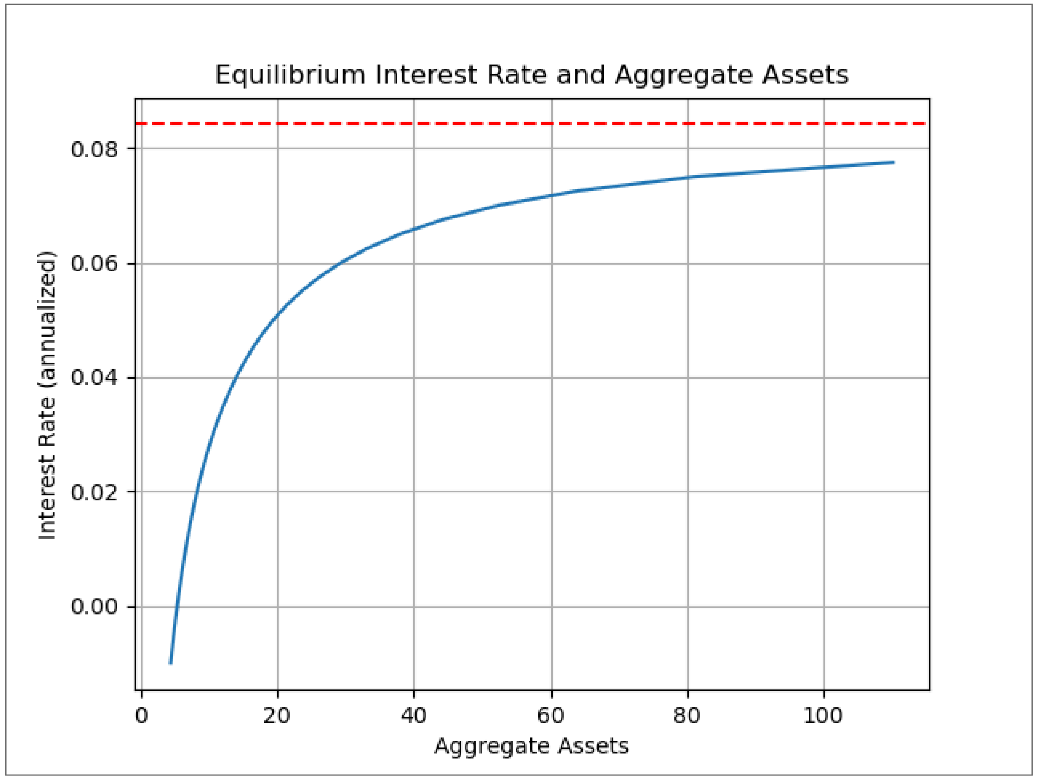
Steady state condition: $D^* = \Lambda D^*$

This is an **eigenvalue problem** with eigenvalue = 1.

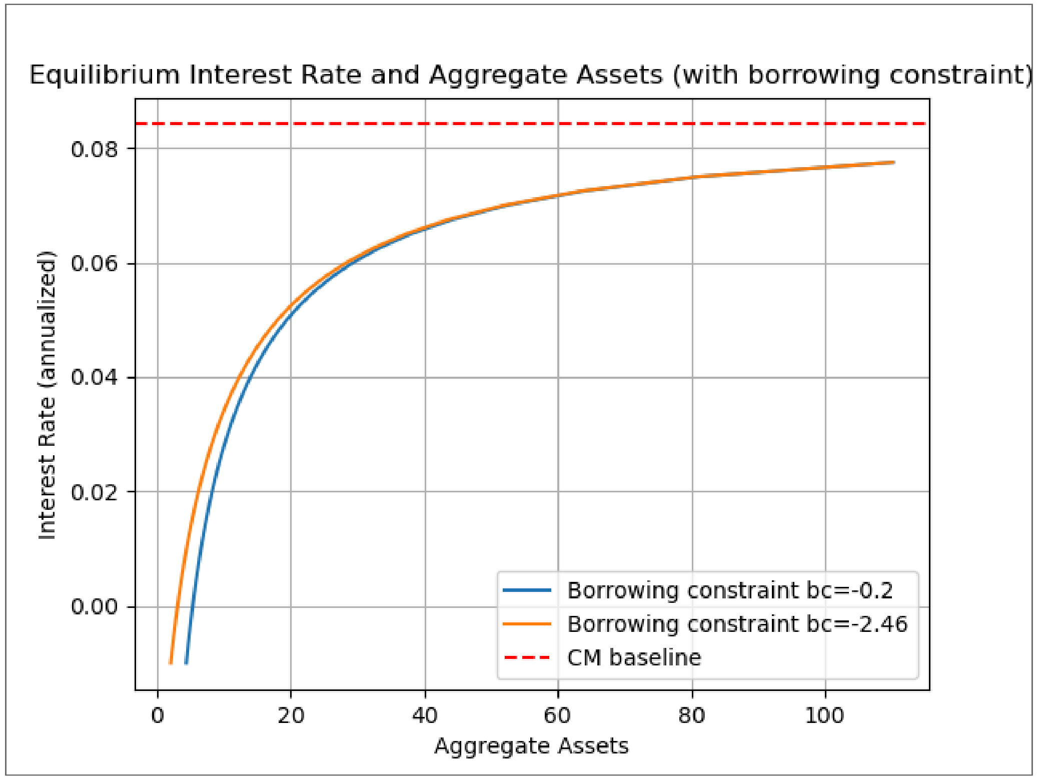
Two approaches:

1. **Power iteration** (iterate until convergence):
    - Start with initial guess D_0 (e.g., uniform distribution)
    - Iterate: $D_{t+1} = \Lambda D_t$
    - Continue until $\|D_{t+1} - D_t\| < \epsilon$
2. **Eigenvalue method** (fast and exact):
    - Find eigenvector of Λ corresponding to eigenvalue = 1
    - $\Lambda v = 1 \cdot v$ where v is the stationary distribution
    - Normalize: $D^* = v / \sum v$


Steady State Wealth Distribution
Density
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100
120
Assets



- Solving the model for different interest rates
- Aggregate asset demand increasing in the interest rate
- The more assets the more we move towards complete markets benchmark



- Loosening borrowing constraints shifts in the asset demand curve

Aggregate Dynamics

- So far considered a model at it's steady state
- Generally we want to think about aggregate dynamics
  - Theoretically under rational expecations this is basiclay impossible
  - But under some reasonable assumptions we have methods that maket this feasible
- With HA models, now common to use peturbation methods around steady state to compute dynamic responses to shocks
  - e.g. SGU method (Bayer et al, 2024) or Sequence Space Jacobian (Auclert et al, 2021)
- All rely on taking derivatives around steady state and solving for policy functions that maintain equlibrium equations.

TFP Shock in a Hugget model

$$Y_t = Z_t \int z_i di$$
$$A_t = B$$
$$\tau_t = r_t B$$
$$\log Z_t = \rho \log Z_{t-1} + \epsilon_t$$

- Fixed level of saving supported by fixed issuance of government bonds B.
- Output fluctuates with aggregate TFP Process
- Interest rate fluctuates to ensure savings supply equals demand
- Government pays for interest payment with a lump sum tax on all households
- Normalise such that $\int z_i di = 1$

In [13]:

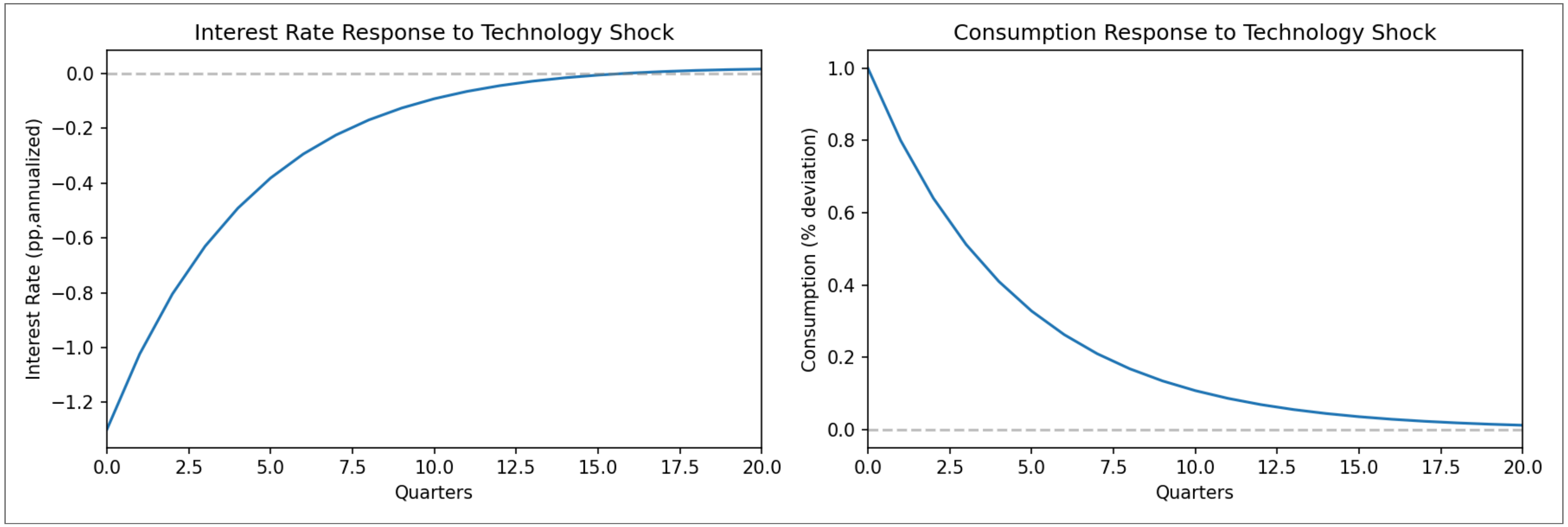
```
@simple
def wage(Z):
    w=Z
    Y=Z
    return w,Y

@simple
def mkt_clearing(A,B,Y,C):
    asset_mkt=A-B
    goods_mkt=C-Y
    return asset_mkt, goods_mkt

@simple
def gov(r,B):
    govT=-r*B
    return govT
```

```
In [14]:
```

```python
cali_full=copy.deepcopy(cali)
cali_full['B']=8
cali_full['Z']=1

hugget_full = create_model([hh,wage,gov,mkt_clearing], name="Hugget")


unknowns_ss = {'r': 0.015}
targets_ss = {'asset_mkt': 0}

ss0_full = hugget_full.solve_steady_state(cali_full, unknowns_ss, targets_ss, solver="hybr")

print('r:',ss0_full['r'])
print('Good_mkt:',ss0_full['goods_mkt'])

# dynamics

Ghugget= hugget_full.solve_jacobian(ss0_full, ['r'], ['asset_mkt'], ['Z'], T=300)
```

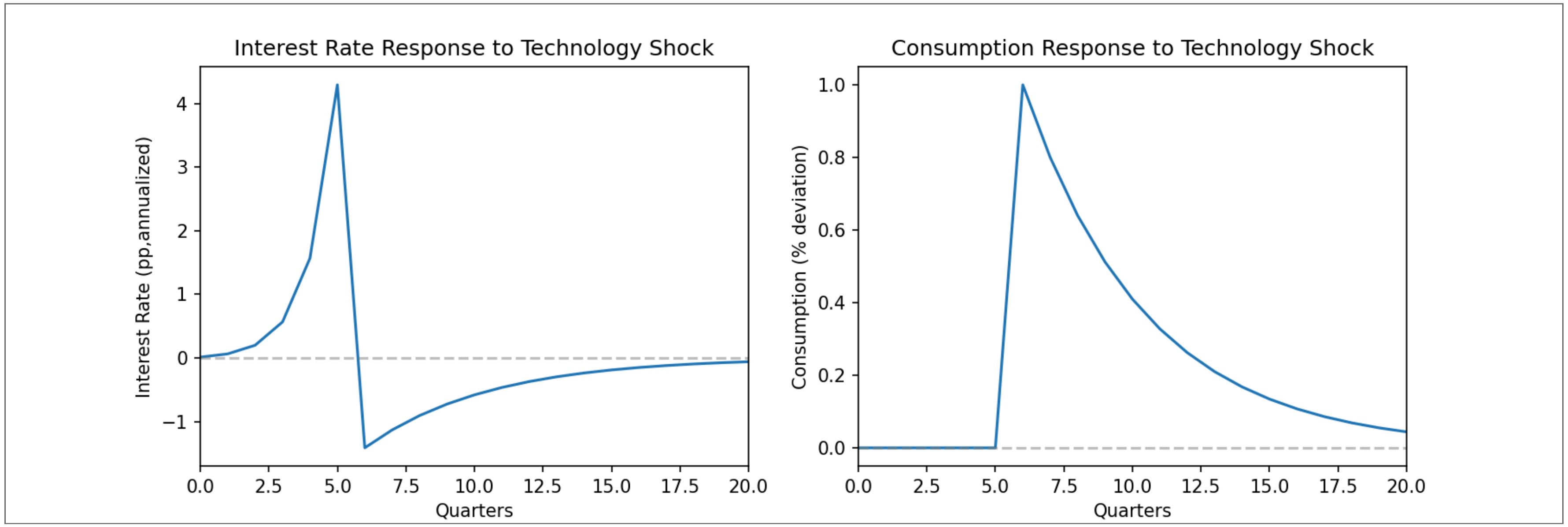
```
r: 0.0038574596889343112
Good_mkt: 5.67532842943308e-09
```

Response to a 1 pct TFP shock



- Nature of shock means today is better than tomorrow
- Households want to save more
- Interest rate has to fall to bring saving demand into equilibrium with supply (r^* falls)

Response to a 1 pct TFP News shock



- Same shock only anticpated 6 quarters in advance
- Nature of shock means that tomorrows is better than today
- Households want to save less (higher r^*)