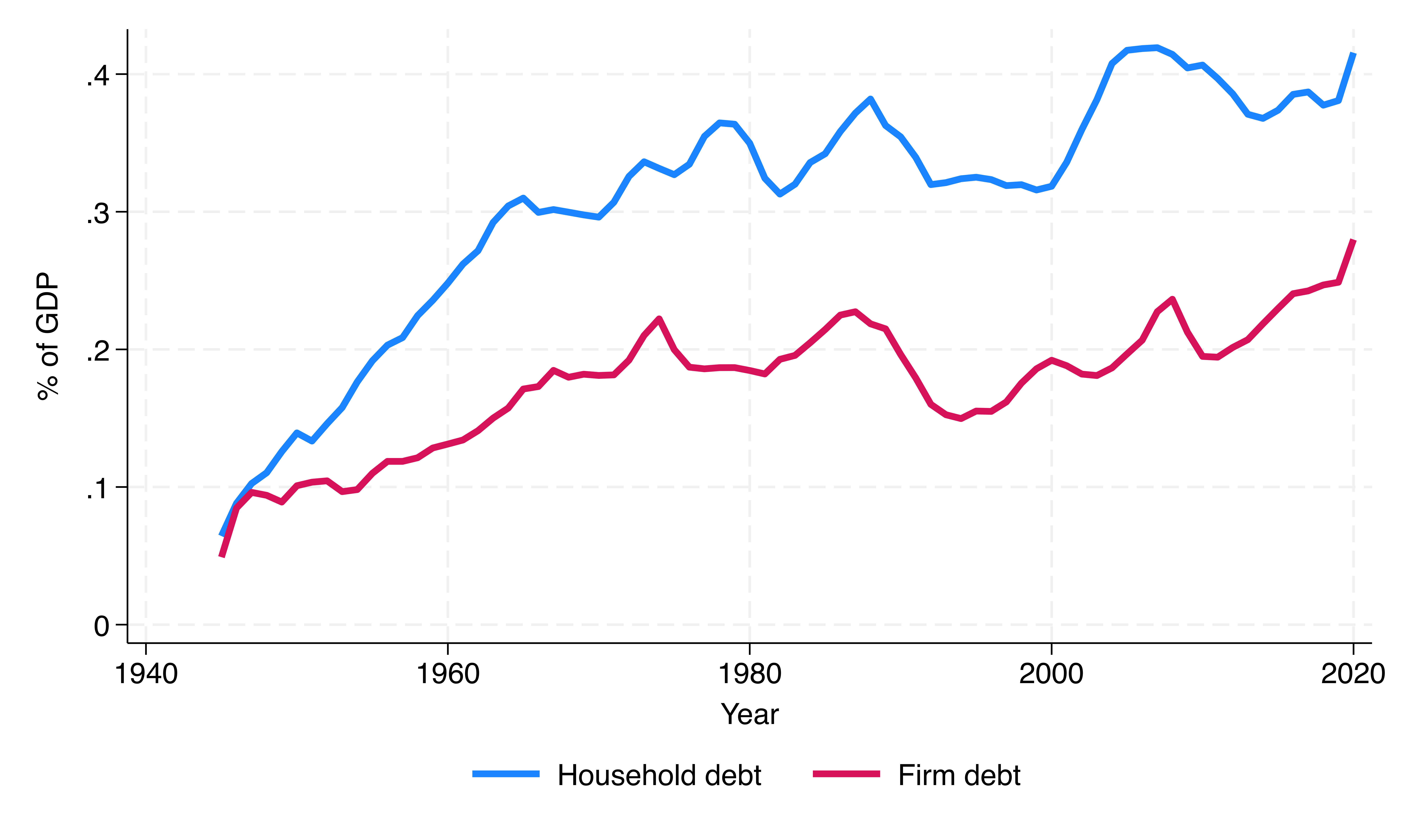
Demand-Side View of Financial Frictions:

Borrowing Constraints and Aggregate Demand

704 Macroeconomics II
Topic 7

Masao Fukui

Household Debt in the US



The Standard Incomplete Market Model

Environment

$$\max_{\{c_{it} \geq 0, a_{it+1}\}} \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_{it})$$

$$\text{s.t.} \quad c_{it} + a_{it+1} = (1+r)a_{it} + y_{it}$$

$$a_{it+1} \geq -\phi$$

- Assume $y_{it} = e_{it} Y$ and e_{it} follows a discrete Markov process ($e_{it} \in \{e_1, \ldots, e_J\}$)
    - Let $y_{\min} \equiv \min_i y_{it} > 0$ and normalize $\mathbb{E}[e_{it}] = 1$ (Y is the aggregate labor income)
- Assume u is strictly increasing and concave, $\lim_{c \to \infty} u'(c) = 0, \lim_{c \to 0} u'(c) = \infty$.
- The key assumption: households only have access to state **non-contingent** asset
- The parameter $\phi \geq 0$ captures the borrowing limit
- For now, assume partial equilibrium: r is exogenously given

Natural Borrowing Limit

- For assets to be state non-contingent, it must be

$$a_{it} \geq -\frac{y_{min}}{r}$$

- Why? State non-contingent $\Leftrightarrow$ households can repay in the worst-case scenario
  - The lifetime budget constraint in the worst case: $y_{is} = y_{min}$ for all $s \geq t$

$$\sum_{s=t}^{\infty} \frac{1}{(1+r)^{s-t}} c_{is} \leq \sum_{s=t}^{\infty} \frac{1}{(1+r)^{s-t}} y_{min} + (1+r)a_{it}$$

  where we used no-Ponzio condition $\lim_{s\to\infty} \frac{1}{(1+r)^{s-t}} a_{is} \geq 0$

  - The maximum repayment HH i can repay is to set $c_{is} = 0$ for all $s \geq t$

$$\Rightarrow \quad (1+r)a_{it} \geq -\sum_{s=t}^{\infty} \frac{1}{(1+r)^{s-t}} y_{min}$$

$$\Leftrightarrow \quad a_{it} \geq -\frac{y_{min}}{r}$$

- This implies $\phi \leq y_{\mathrm{min}}/r$

Household Problem in Recursive Form

$$V(a,y) = \max_{c,\,a' \geq -\phi} u(c) + \beta \mathbb{E} V(a', y')$$

$$\text{s.t.} \quad c + a' = (1+r)a + y$$

- State variables: (a, y). Policy functions: $c(a,y)$, $a'(a,y)$.
- The first-order condition

$$u'(c(a,y)) \geq \beta \mathbb{E} \partial_a V(a'(a,y), y')$$

with equality whenever $a' > -\phi$

- The envelope condition

$$\partial_a V(a,y) = (1+r) u'(c(a,y))$$

- Combining FOC and envelope gives the Euler equation (in sequential notation)

$$u'(c_{it}) \geq \beta(1+r) \mathbb{E}_t u'(c_{it+1})$$

Impossibility of $\beta(1+r) \geq 1$

$$u'(c_{it}) \geq \beta(1+r)\mathbb{E}u'(c_{it+1})$$

Result: If $\beta(1+r) \geq 1$, $c_{it} \to \infty$ almost surely

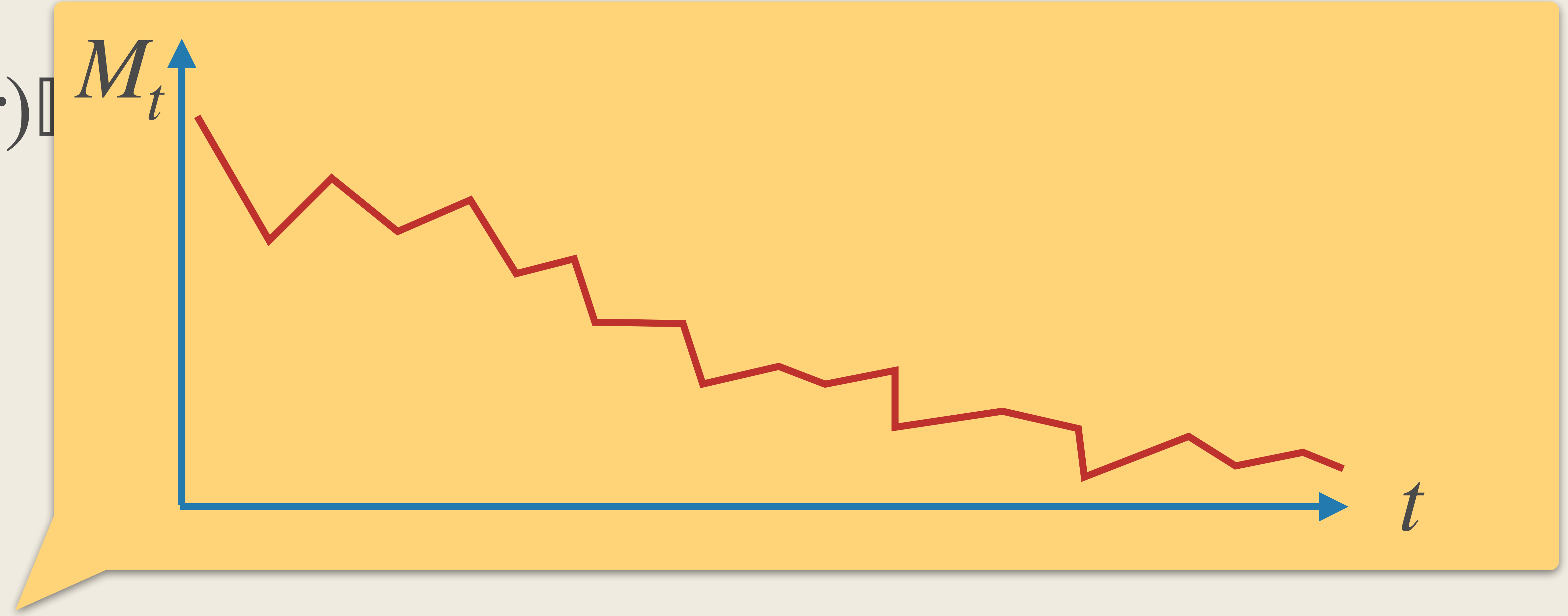
Proof sketch:

1. **Supermartingale convergence theorem**:
   If a stochastic process M_t satisfies $M_t \geq \mathbb{E}_t M_{t+1}$ and is bonded, M_t converges to some random variable $M*$ almost surely.
2. Since $u'(c_{it}) \geq \mathbb{E}u'(c_{it+1})$ and $u'(c_{it})$ is bounded, $u'(c_{it}) \to u'*$.
   i. Could it be $u'* > 0$? Then $c_{it} \to c* < \infty$, but impossible because y_{it} fluctuates
   ii. So $u'* = 0$, which implies $c_{it} \to \infty$

But $c_{it} \to \infty$, which requires $a_{it} \to \infty$, cannot happen in a general equilibrium

Impossibility of $\beta(1+r) \geq 1$

$$u'(c_{it}) \geq \beta(1+r)$$



Result: If $\beta(1+r) \geq 1$, $c_{it} \to \infty$ almost surely

Proof sketch:

1. **Supermartingale convergence theorem**:
   If a stochastic process M_t satisfies $M_t \geq \mathbb{E}_t M_{t+1}$ and is bonded, M_t converges to some random variable M^* almost surely.
2. Since $u'(c_{it}) \geq \mathbb{E}u'(c_{it+1})$ and $u'(c_{it})$ is bounded, $u'(c_{it}) \to u'^*$.
   i. Could it be $u'^* > 0$? Then $c_{it} \to c^* < \infty$, but impossible because y_{it} fluctuates
   ii. So $u'^* = 0$, which implies $c_{it} \to \infty$

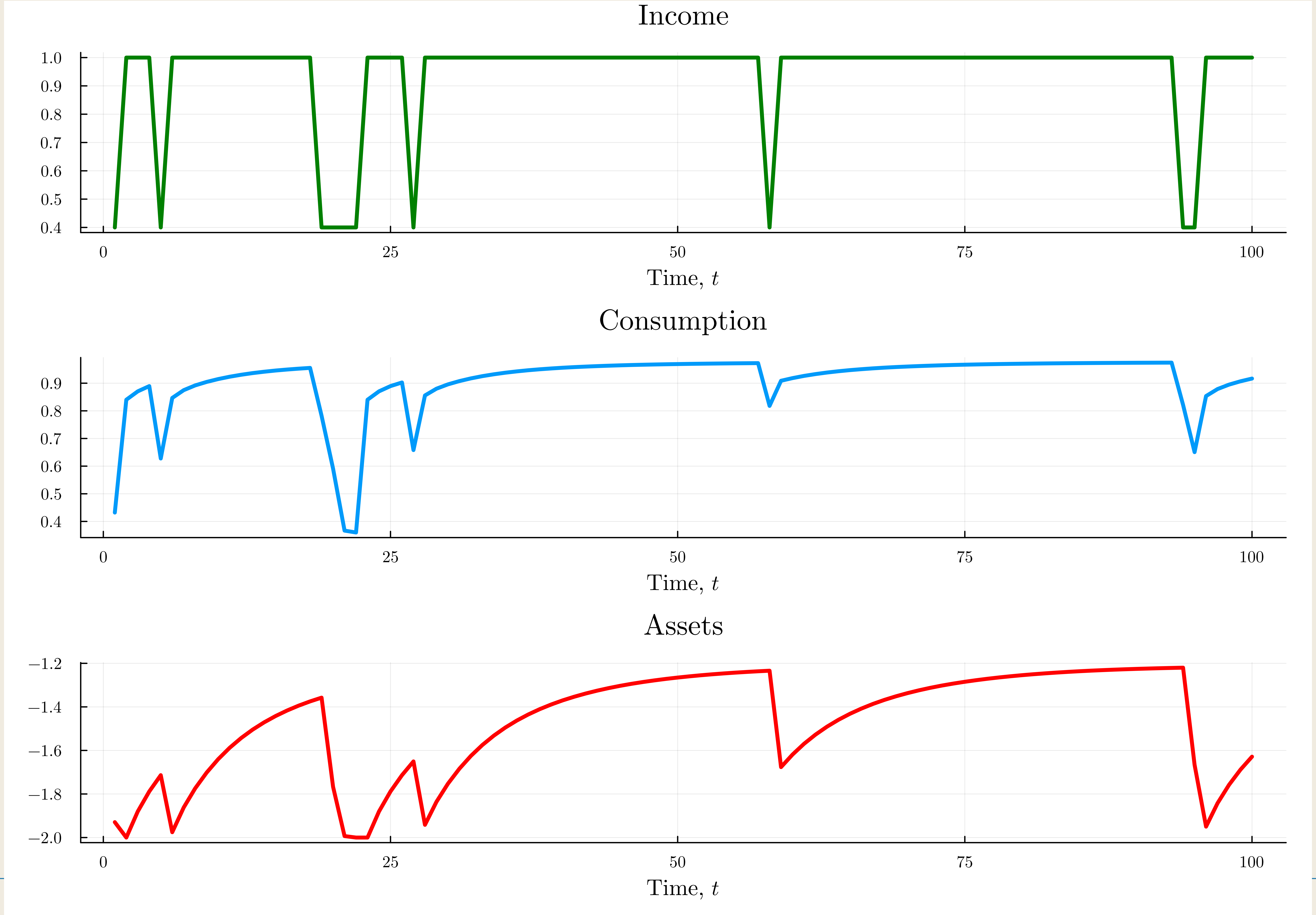
But $c_{it} \to \infty$, which requires $a_{it} \to \infty$, cannot happen in a general equilibrium

Incomplete Market Depresses Real Rate

- So, in order to have well-defined wealth distribution, $(1 + r) < 1/\beta$
- In the steady state of the complete market, $(1 + r) = 1/\beta$
- The incomplete market robustly depresses the real interest rate (again!)
- **Intuition**: incomplete market $\Rightarrow$ precautionary savings
  - But we didn't make any assumption about the third derivative, u'''
  - How can precautionary savings be the intuition?
  - Any globally increasing & concave function must feature $u'''(c) > 0$ as $c \to \infty$

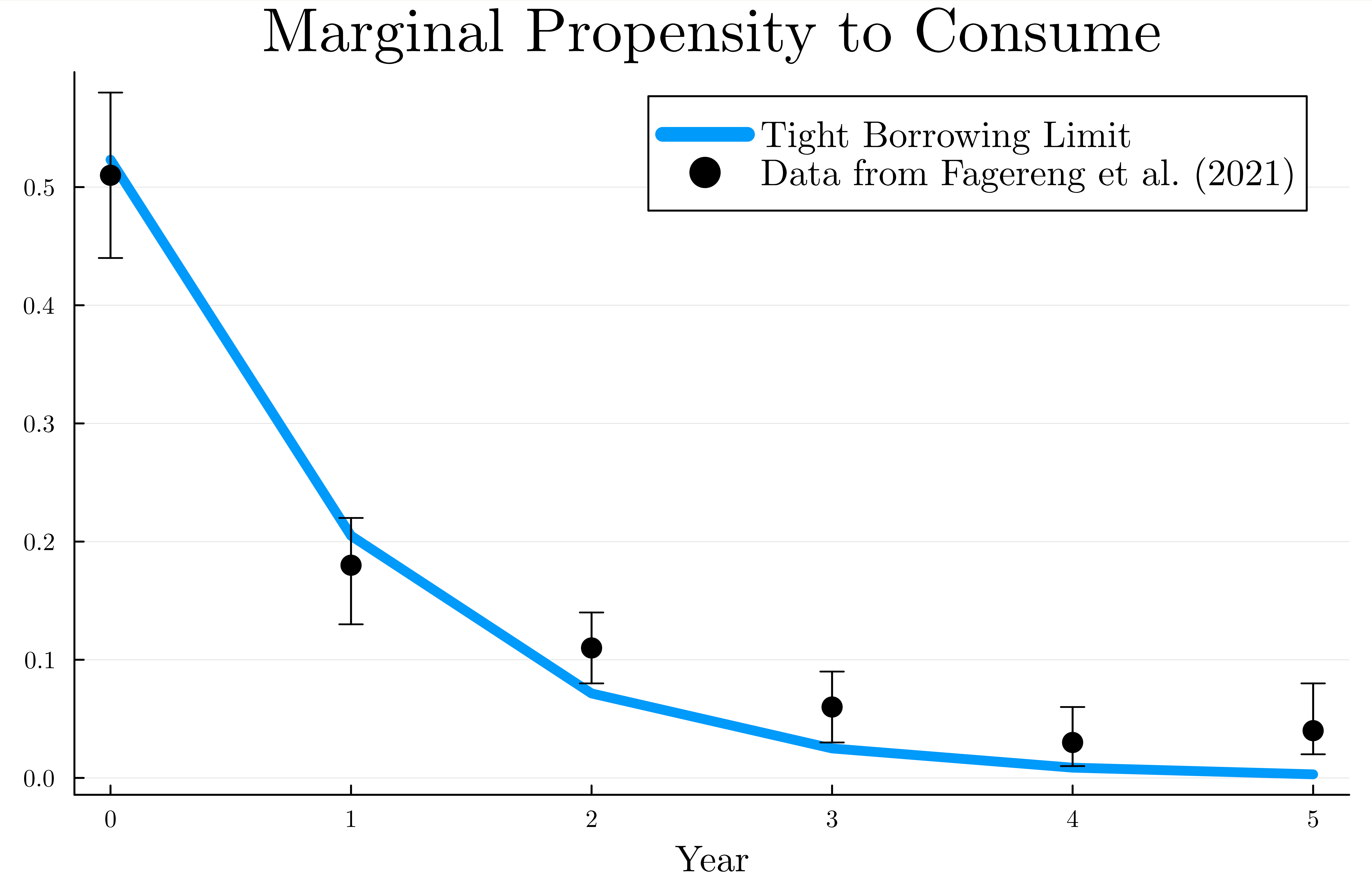
Policy Functions

Simulation

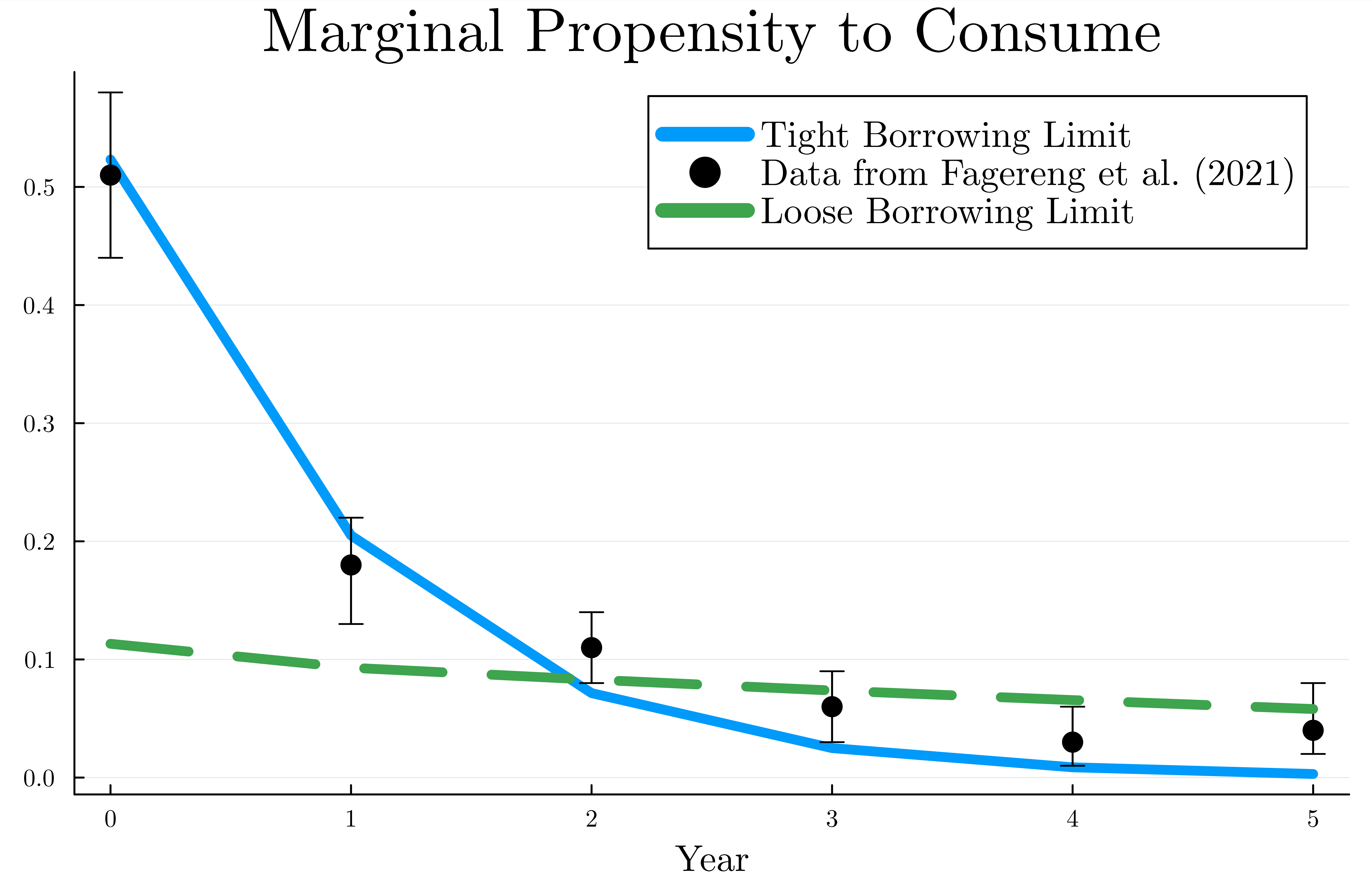


Income
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
25
50
75
100
Time, t
Consumption
0.9
0.8
0.7
0.6
0.5
0.4
0
25
50
75
100
Time, t
Assets
−1.2
−1.4
−1.6
−1.8
−2.0
0
25
50
75
100
Time, t

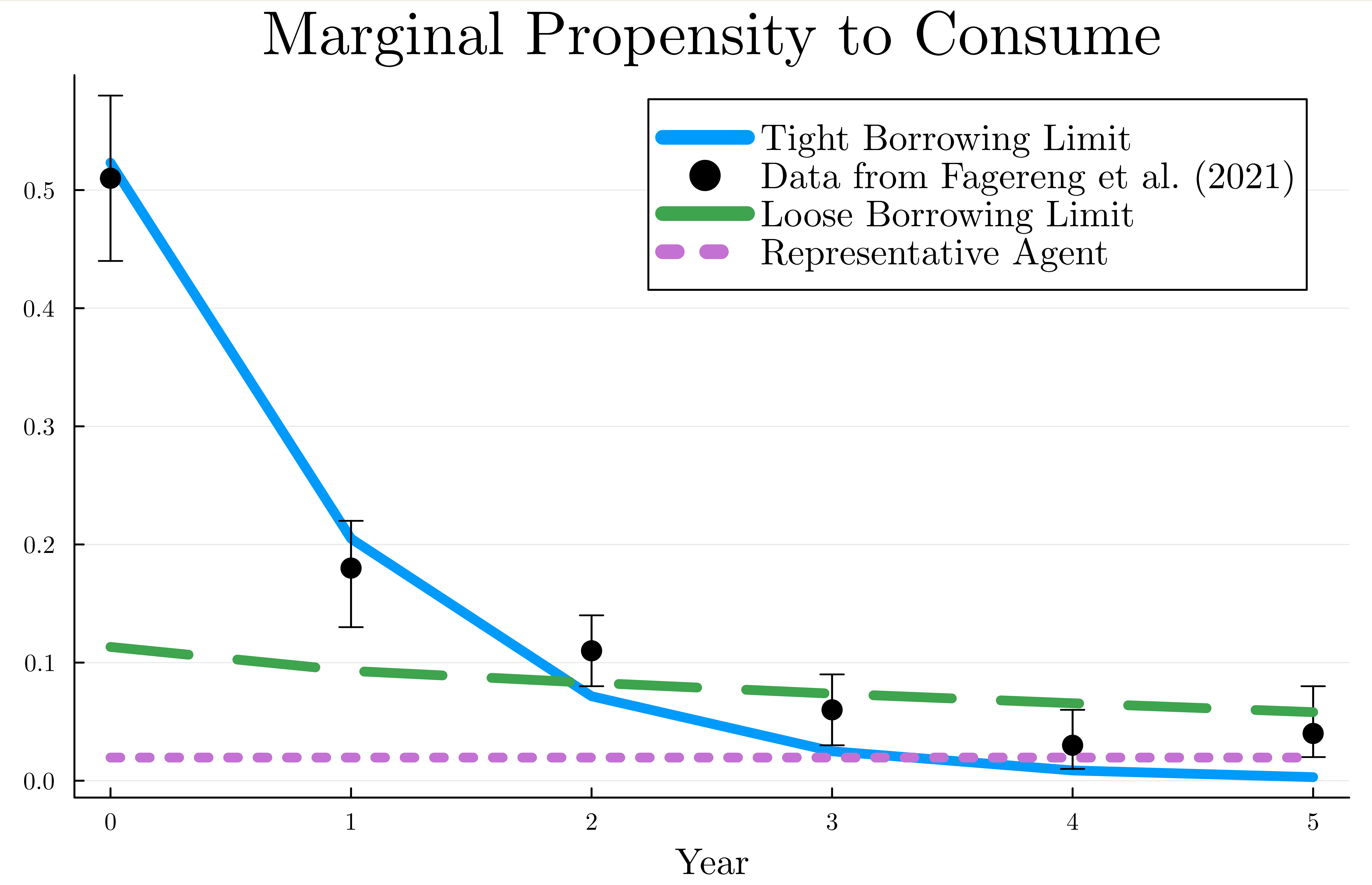
Marginal Propensity to Consume



Marginal Propensity to Consume



Marginal Propensity to Consume

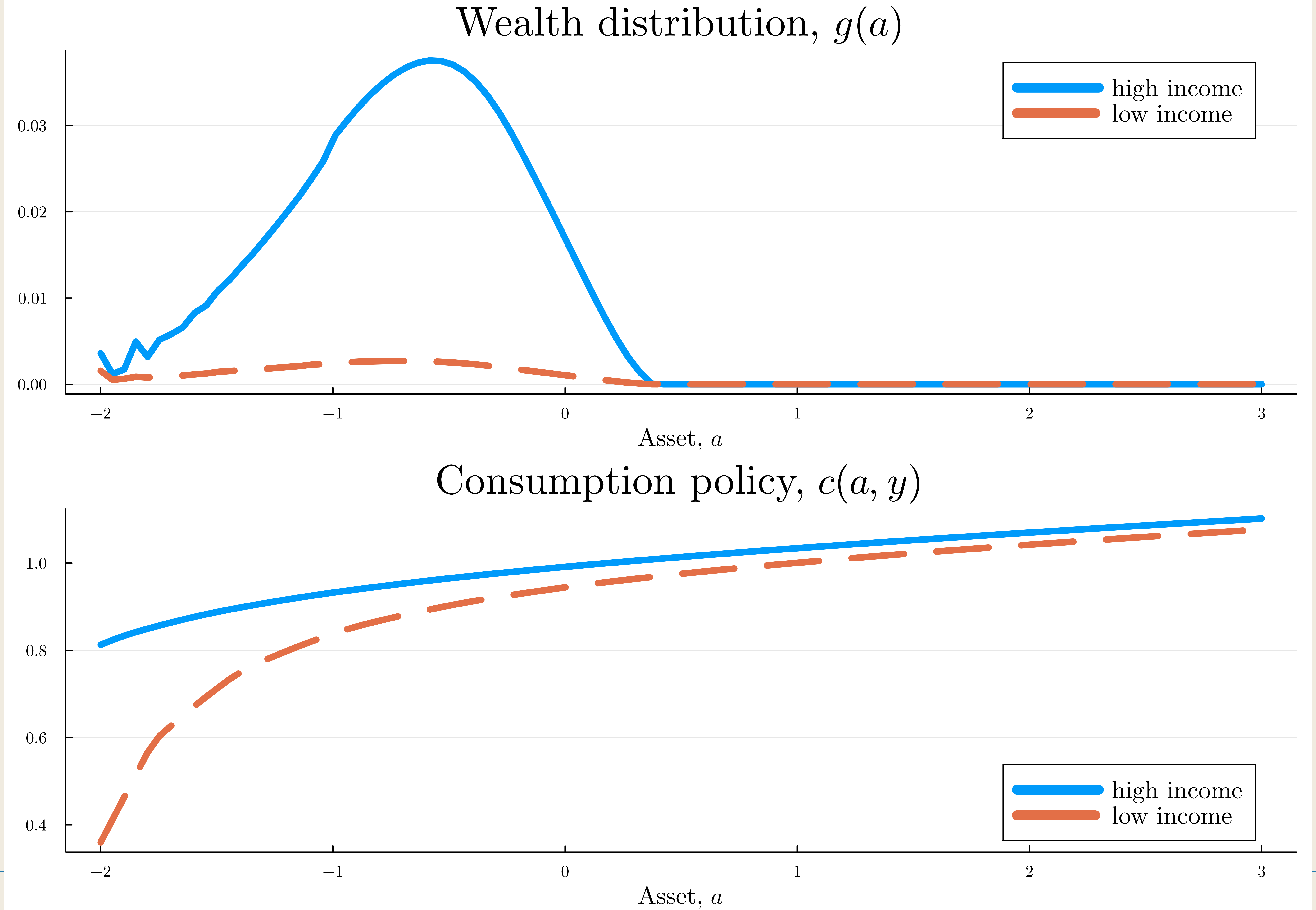


Stationary Wealth Distribution

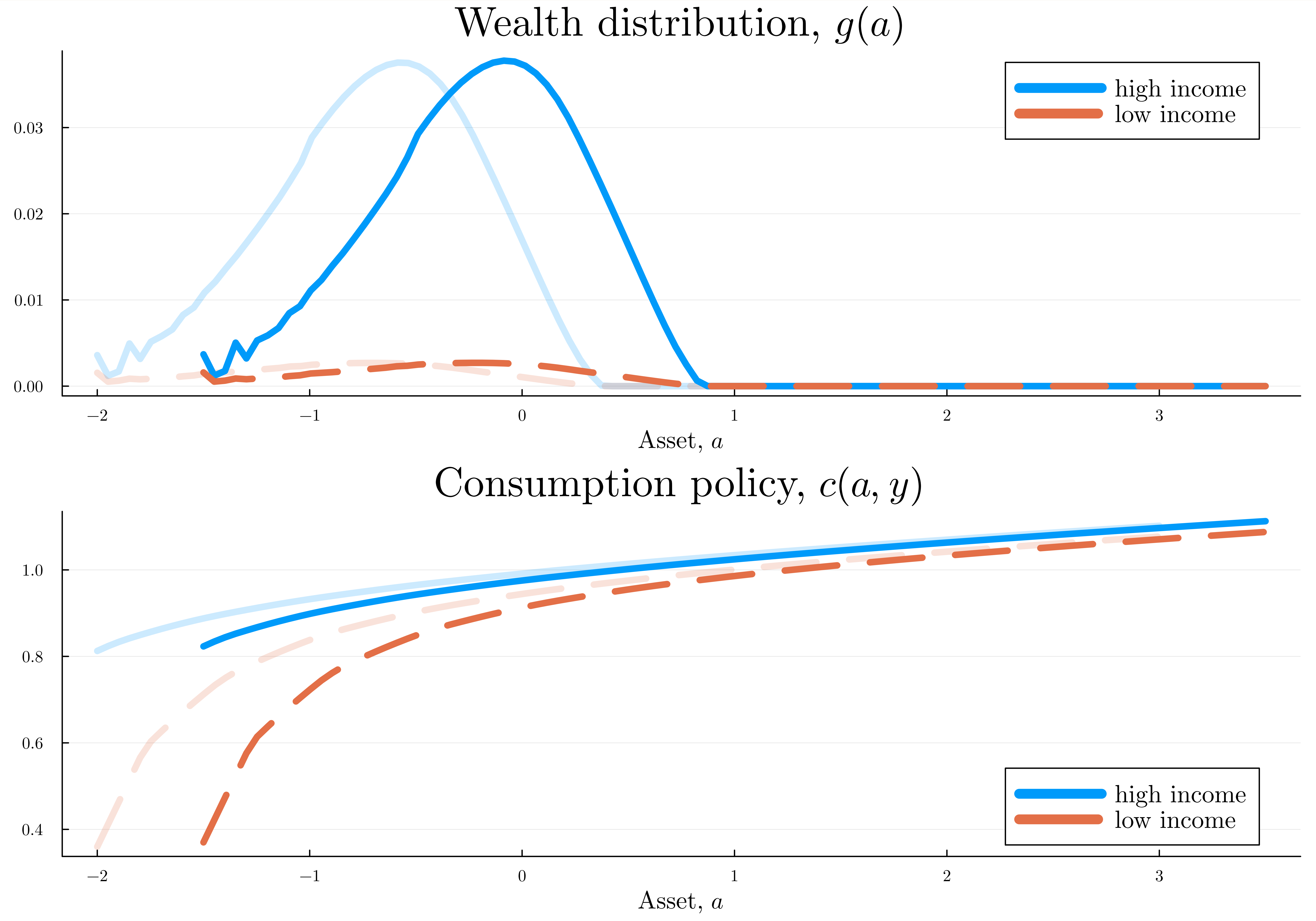
Convergence of Wealth Distribution

- Now consider an economy consisting of a continuum of households
- Let $\mu_t(A, y_j)$ be the measure of households with income y_j and asset $a \in A$ at time t
- Start from some income and asset distribution $\mu_0(A, y_j)$
- Using Prob$(y_i \mid y_j)$ and $a'(a, y_j)$, compute $\mu_1(A, y_j)$ and repeat $\mu_t(A, y_j) \to \mu_{t+1}(A, y_j)$
- Questions:
  1. Is there an invariant distribution such that $\mu^*(A, y_j) \equiv \mu_t(A, y_j) = \mu_{t-1}(A, y_j)$?
  2. If it exists, is the invariant distribution unique?
  3. Do we converge to the invariant distribution, $\lim_{t \to \infty} \mu_t(A, y_j) = \mu^*(A, y_j)$?
- The answers are all **yes** if $\beta(1 + r) < 1$ and RRA, $-u''(c)c/u'(c)$, is bounded above
- TA session will cover the result in detail

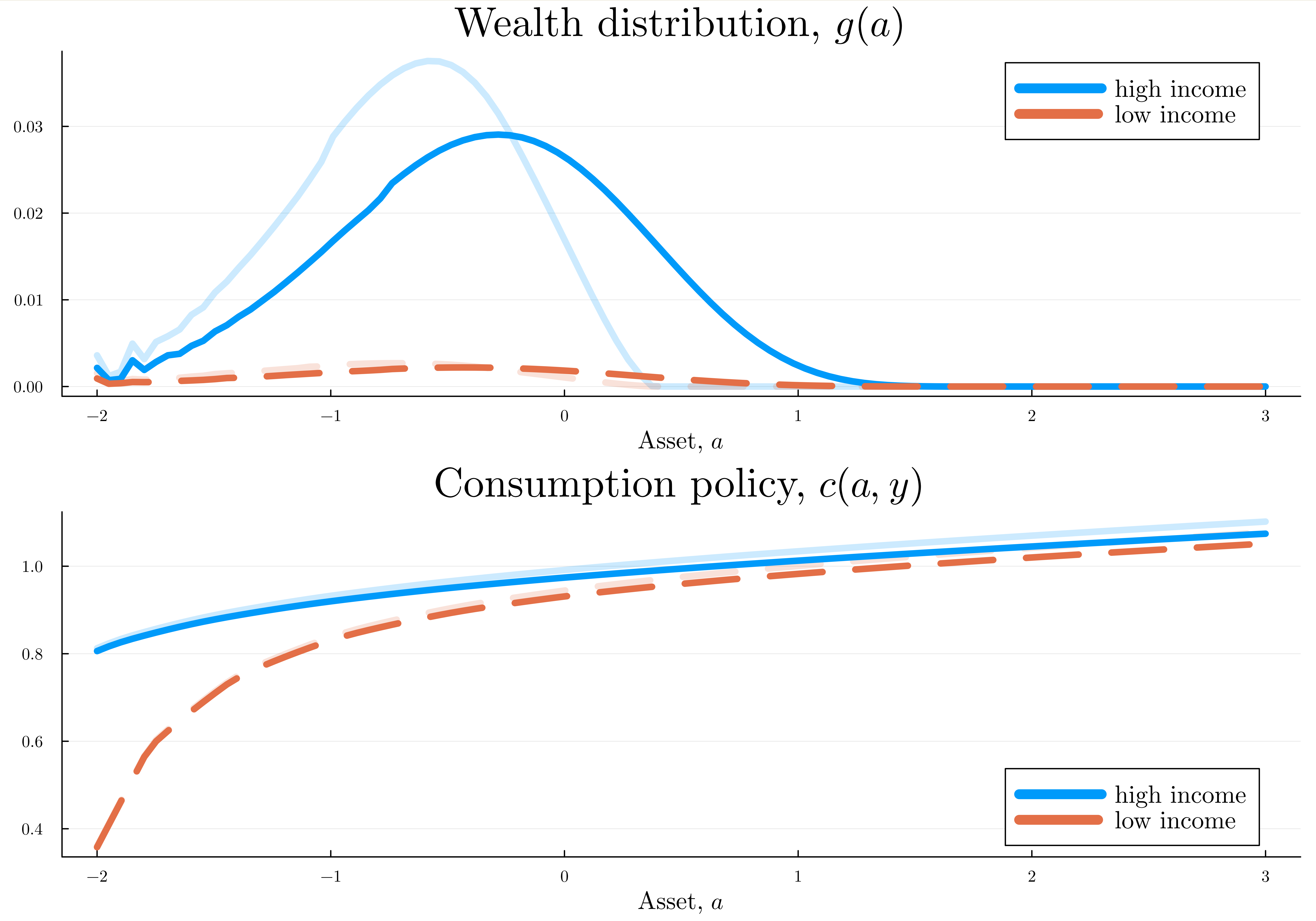
Wealth Distribution



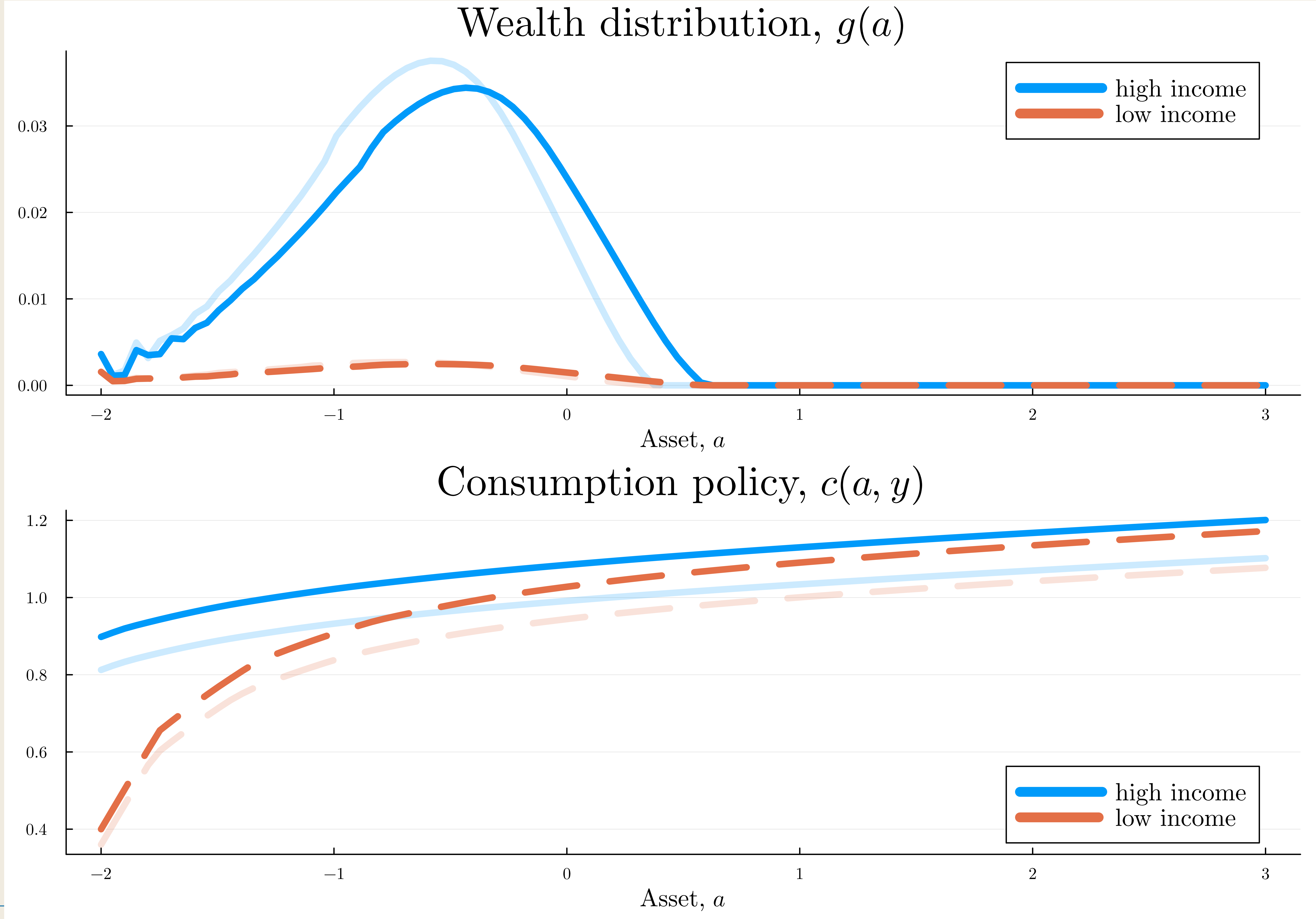
A Reduction in ϕ



An Increase in r



An Increase in Y



Numerically Computing the Standard Incomplete Market Model

Numerical Implementation

- How do we solve the canonical incomplete market model on our computer?
- Two blocks:
  1. Bellman equation $\Rightarrow$ policy functions $a'(a, y)$ and $c(a, y)$
  2. Policy functions $a'(a, y)$ and $c(a, y)$ $\Rightarrow$ stationary distribution

Parameter Settings

```
function set_parameters(;beta = nothing, phi = 2.0)
    # income process
    yg = [0.4; 1.0];
    Ny = length(yg);
    job_finding = 1-exp(-0.4*3);
    job_losing = 1-exp(-0.02*3);
    ytran = [1-job_finding job_finding ; job_losing 1-job_losing ];
    # asset grid
    Na = 100;
    amin = -phi;
    amax = amin + 5.0;
    ag = range(amin,amax,length=Na);
    # risk aversion for utility function
    gamma = 1.0;
    return (
        yg = yg,ag = ag,Ny = Ny,Na = Na, amin = amin, amax = amax,
        tol = 1e-4, gamma = gamma, ytran = ytran, phi = phi, beta = beta
    )
end
```

Solving Bellman Equation

$$V(a_t, y_t) = \max_{c_t, a_t \geq -\phi} u(c_t) + \beta \mathbb{E} V(a_t, y_{t+1})$$

$$\text{s.t.} \quad c_t + a_t = (1 + r) a_{t-1} + y_t$$

- How do we solve the Bellman equation?
  1. Value function iteration or policy function iteration
  2. Endogenous gridpoint method (Caroll, 2007)
- The endogenous gridpoint method is a thousand times faster

Policy Function Iteration

$$u'\big((1+r)a + y - a'_t(a,y)\big) \geq \beta(1+r)\mathbb{E}u'\Big(c_{t+1}\big(a'_t(a,y), y'\big)\Big)$$

- The usual policy function iteration:
  1. Guess policy $c_T(a, y)$
  2. For $t = T-1, T-2, \ldots,$
     1. Given $c_{t+1}(a, y)$, for each (a, y), solve for a'_t that solves the Euler

        $$u'\big((1+r)a + y - a'_t\big) \geq \beta(1+r)\mathbb{E}u'\Big(c_{t+1}\big(a'_t, y'\big)\Big)$$

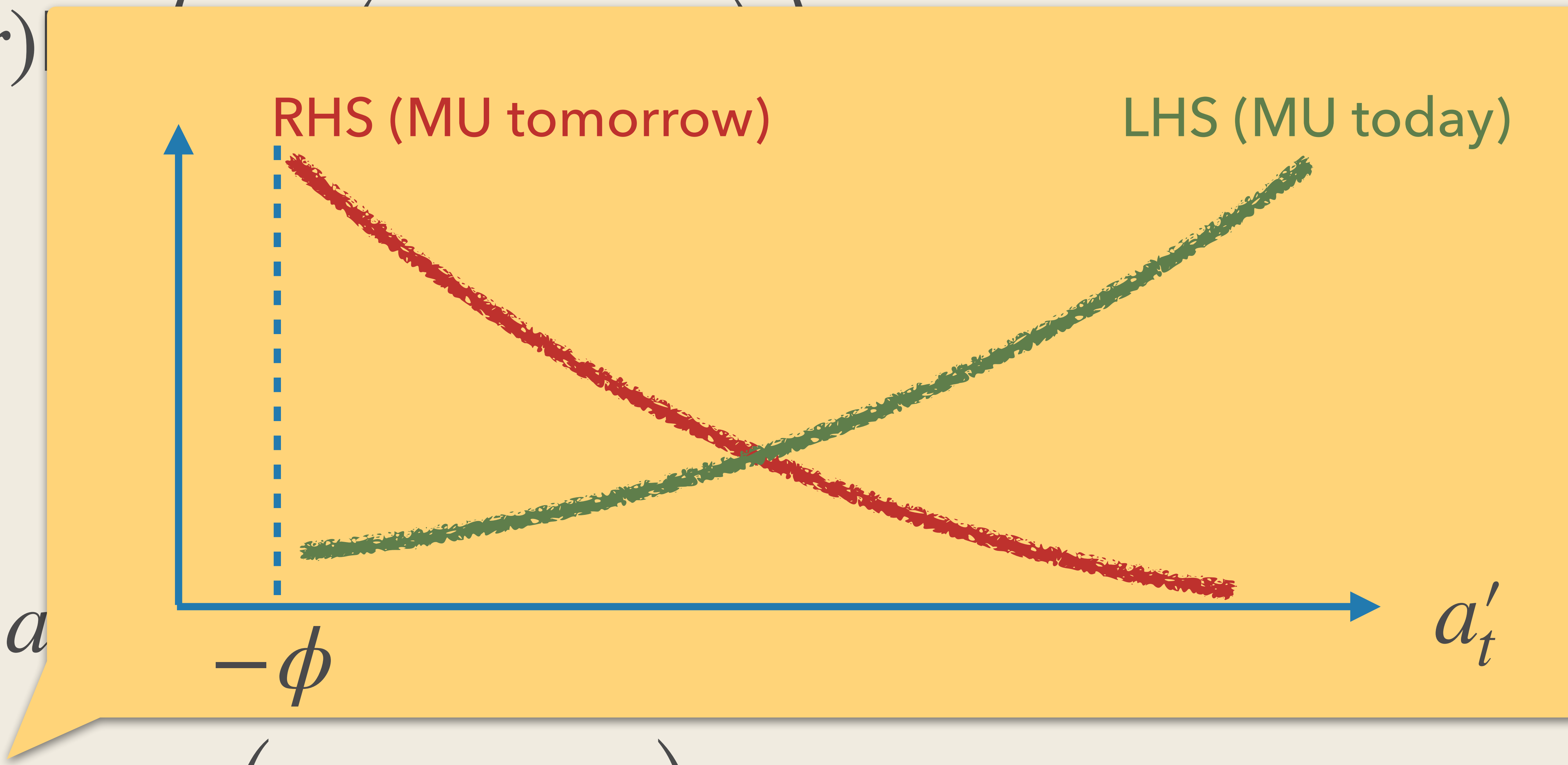
     2. Obtain $c_t(a, y)$ using $a'_t(a, y)$ and the budget constraint:

        $$c_t(a, y) = (1+r)a + y - a'_t(a, y)$$

     3. Repeat until $|c_t(a, y) - c_{t+1}(a, y)|$ small enough

Policy Function Iteration

$$u'\Big((1+r)a+y-a_t'(a,y)\Big) \geq \beta(1+r)$$



- The usual policy function iteration:
  1. Guess policy $c_T(a,y)$
  2. For $t = T-1, T-2, \ldots,$
     1. Given $c_{t+1}(a,y)$, for each (a,y), solve for a

$$u'\Big((1+r)a+y-a_t'\Big) \geq \beta(1+r)\mathbb{E}u'\Big(c_{t+1}\big(a_t', y'\big)\Big)$$

     2. Obtain $c_t(a,y)$ using $a_t'(a,y)$ and the budget constraint:

$$c_t(a,y) = (1+r)a + y - a_t'(a,y)$$

  3. Repeat until $|c_t(a,y) - c_{t+1}(a,y)|$ small enough

Policy Function Iteration

$$u'\Big((1+r)a + y - a_t'(a,y)\Big) \geq \beta(1+r)\mathbb{}$$

- The usual policy function iteration:
  1. Guess policy $c_T(a,y)$
  2. For $t = T-1, T-2, \ldots,$
     1. Given $c_{t+1}(a,y)$, for each (a,y), solve for a

$$u'\Big((1+r)a + y - a_t'\Big) \geq \beta(1+r)\mathbb{E}u'\Big(c_{t+1}\big(a_t', y'\big)\Big)$$

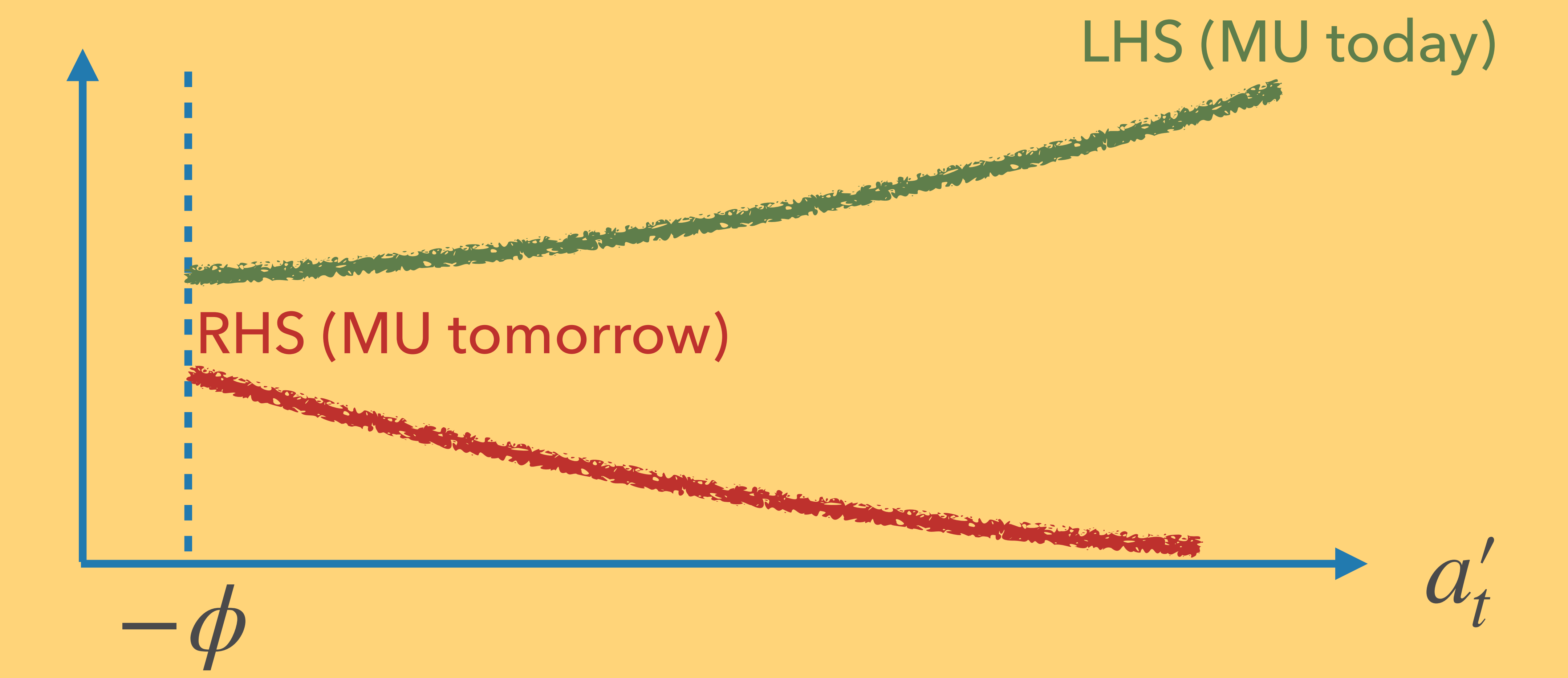
     2. Obtain $c_t(a,y)$ using $a_t'(a,y)$ and the budget constraint:

$$c_t(a,y) = (1+r)a + y - a_t'(a,y)$$

     3. Repeat until $|c_t(a,y) - c_{t+1}(a,y)|$ small enough

Policy Function Iteration

$$u'\Big((1+r)a + y - a_t'(a,y)\Big) \geq \beta(1+r)$$



- The usual policy function iteration:
  1. Guess policy $c_T(a,y)$
  2. For $t = T-1, T-2, \ldots,$
     1. Given $c_{t+1}(a,y)$, for each (a,y), solve for a

$$u'\Big((1+r)a + y - a_t'\Big) \geq \beta(1+r)\mathbb{E}u'\Big(c_{t+1}\big(a_t', y'\big)\Big)$$

     2. Obtain $c_t(a,y)$ using $a_t'(a,y)$ and the budget constraint:

$$c_t(a,y) = (1+r)a + y - a_t'(a,y)$$

     3. Repeat until $|c_t(a,y) - c_{t+1}(a,y)|$ small enough

Endogenous Gridpoint Method

- Policy function iteration is expensive because it involves root-finding

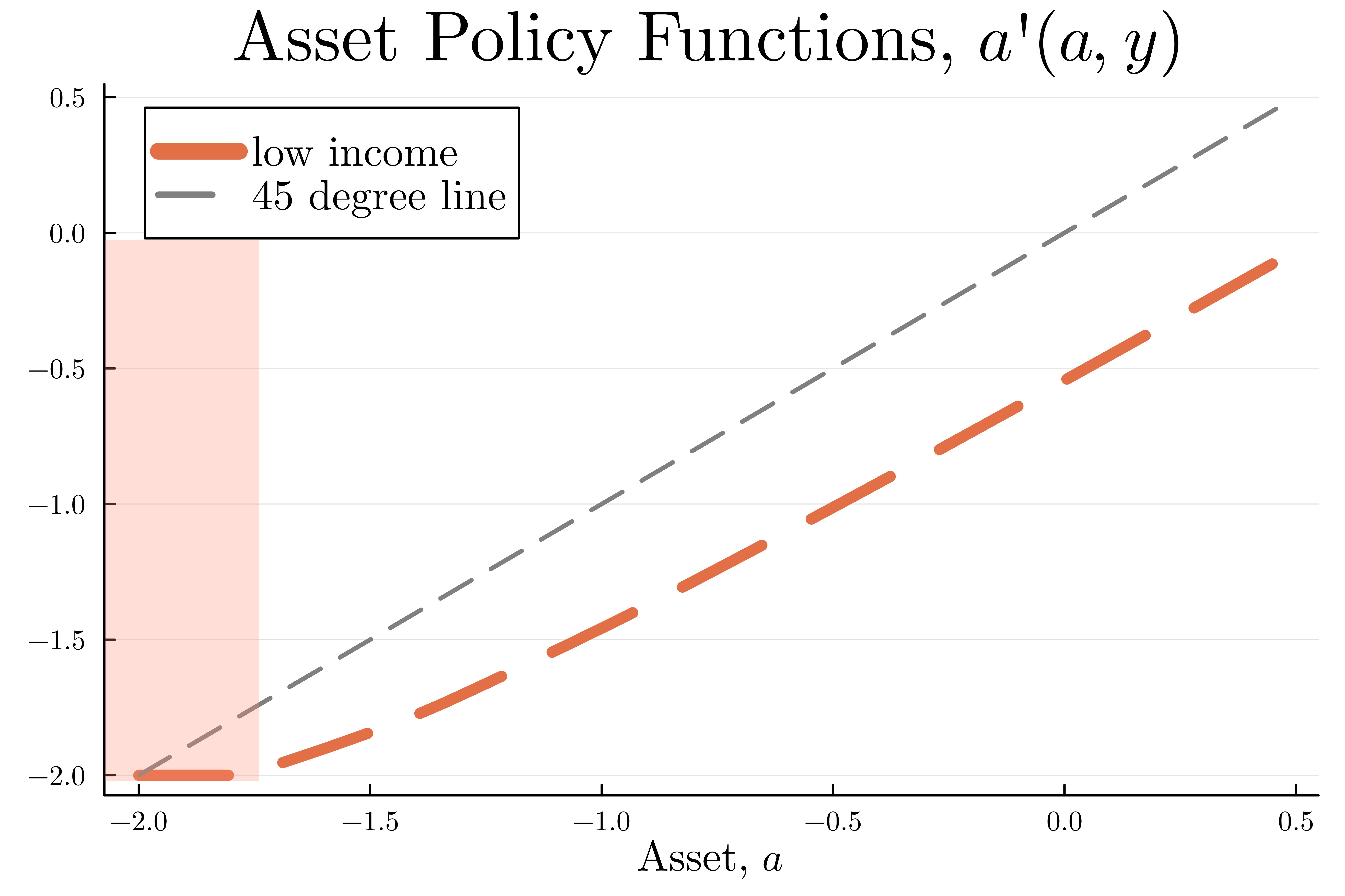
- Key observation: a_t' is not analytical, but a is!

$$u'\big((1+r)a + y - a_t'\big) = \beta(1+r)\mathbb{E}u'\Big(c_{t+1}\big(a_t', y'\big)\Big)$$

$$\Leftrightarrow \quad a = \frac{1}{1+r}\left[u'^{-1}\left(\beta(1+r)\mathbb{E}u'\Big(c_{t+1}\big(a_t', y'\big)\Big)\right) + a_t' - y\right]$$

- Much easier if we ask the reverse question: what is my state when I am saving a'?

Which a' rationalizes a?



Endogenous Gridpoint Method

- For each (a_i', y_j) on the grid $A \equiv [a_1, \dots, a_I]$ and $Y \equiv [y_1, \dots, y_J]$, obtain

$$a_{i,j}^* = \frac{1}{1+r}\left[u^{'-1}\left(\beta(1+r)\mathbb{E}u'\left(c_{t+1}\left(a_i', y'\right)\right)\right) + a_i' - y_j\right]$$

- This gives an updated policy on endogenously determined grid points:

$$a_t'(a_{i,j}^*, y_j) = a_i' \qquad \text{(EGM)}$$

hence its name!

- Interpolate (EGM) to obtain the policy on the original grid:

$$a_t(a_i, y_j) = \begin{cases} \text{linearly interpolate (EGM)} & \text{if } a \geq a_{1,j}^* \\ -\phi & \text{if } a < a_{1,j}^* \end{cases}$$

where we used that for $a < a_{1,j}^*$, the borrowing constraint must be binding

Outer Loop for EGM

```
function solve_policy_EGM(param, beta, r)
    @unpack Na, Ny,tol = param
    c_pol_old = ones(Na,Ny)
    c_pol_new = 100*ones(Na,Ny)
    a_pol_new = []
    iter = 0;
    while maximum(abs.(c_pol_new .- c_pol_old)) > tol
        c_pol_old = c_pol_new
        c_pol_new, a_pol_new = Euler_iteration_once(param,c_pol_old,beta,r)
        iter += 1
    end
    return (
        c_pol = c_pol_new,
        a_pol = a_pol_new,
    )
end
```

Inner Problem

```
function Euler_iteration_once(param,c_pol_old, beta, r)
    @unpack Na, Ny, ag, yg,ytran,amin = param

    uprime_future = uprime_fun(param,c_pol_old)
    a_today_unconstrained = zeros(Na,Ny)
    for (ia,a_future) in enumerate(ag)
        uprime_future_y =  uprime_future[ia,:]
        Euler_RHS = beta.*(1.0 .+r).*(ytran*uprime_future_y)
        c_today_unconstrained = uprime_inv_fun(param,Euler_RHS)
        a_today_unconstrained[ia,:] =  (c_today_unconstrained .+ a_future .- yg)./(1.0 .+r)
    end

    a_pol_new = zeros(Na,Ny)
    c_pol_new = zeros(Na,Ny)
    for (iy,y_today) in enumerate(yg)
        ainterp = LinearInterpolation(a_today_unconstrained[:,iy],ag,
                   extrapolation_bc=Interpolations.Flat())
        a_pol_new[:,iy] = ainterp.(ag)
        a_pol_new[a_pol_new[:,iy] .< amin,iy] .= amin
        c_pol_new[:,iy] = (1 .+r).*ag .+ y_today .- a_pol_new[:,iy]
    end
    return c_pol_new, a_pol_new
end
```

Obtaining Stationary Distribution

- The second block is solving for the stationary distribution over (a, y)
- Let $\mathbf{\Pi}$ be $(I \times J) \times (I \times J)$ transition matrix
- The distribution $\boldsymbol{g}_t$ evolves

$$\boldsymbol{g}_{t+1} = \Pi^T \boldsymbol{g}_t$$

- In the steady state, $\boldsymbol{g}_{t+1} = \boldsymbol{g}_t \equiv \boldsymbol{g}$, so that

$$\left[\mathbf{I} - \mathbf{\Pi}^T\right] \boldsymbol{g} = \mathbf{0}$$

- Together with $\sum_{i,j} g_{i,j} = 1$, solving this system of equations would give $\boldsymbol{g}$
- How do we construct $\mathbf{\Pi}$?

Allocating on Grid

- We construct $\mathbf{\Pi}$ using $a'(a, y)$ and Prob$(y' \mid y)$
- The issue is that $a'(a, y)$ is not necessarily on the grid
- Properly allocate on the grid:

$$\text{Prob}(a_j \mid (a, y)) = \mathbb{I}_{a'(a,y) \in [a_{j-1}, a_j]} \frac{a'(a, y) - a_{j-1}}{a_j - a_{j-1}} + \mathbb{I}_{a'(a,y) \in [a_j, a_{j+1}]} \frac{a_j - a'(a, y)}{a_{j+1} - a_j}$$

- Unbiased in the aggregate
- This is referred to as "non-stochastic simulation" (Young, 2010)

Solve SS Distribution by Matrix Inversion

```
function solve_ss_distribution(param,Bellman_result)
    @unpack Na, Ny = param
    @unpack a_pol = Bellman_result
    transition_matrix = construct_transition_matrix(param,a_pol)

    Matrix_to_invert = I - transition_matrix'
    Matrix_to_invert[end,:] = ones(Na*Ny)

    RHS = zeros(Na*Ny)
    RHS[end] = 1.0;
    ss_distribution = Matrix_to_invert\RHS
    @assert sum(ss_distribution) ≈ 1.0
    ss_distribution = reshape(ss_distribution,Na,Ny)
    return ss_distribution
end
```

Construct Transition Matrix

```
function construct_transition_matrix(param, a_pol)
    @unpack Na, Ny, yg, ytran,ag = param
    transition_matrix = zeros(Na*Ny,Na*Ny)
    for ia = 1:Na
        for iy = 1:Ny
            index_ia_iy = compute_index_ia_iy(param,ia,iy)
            a_next = a_pol[ia,iy]
            left_grid,right_grid,left_weight,right_weight = find_nearest_grid(ag,a_next)
            for iy_next = 1:Ny
                index_ia_iy_next = compute_index_ia_iy(param,left_grid,iy_next)
                transition_matrix[index_ia_iy,index_ia_iy_next] += left_weight*ytran[iy,iy_next]

                index_ia_iy_next = compute_index_ia_iy(param,right_grid,iy_next)
                transition_matrix[index_ia_iy,index_ia_iy_next] += right_weight*ytran[iy,iy_next]
            end
        end
    end
    transition_matrix = sparse(transition_matrix)
    @assert sum(transition_matrix,dims=2) ≈ ones(Ny*Na)
    return transition_matrix
end
```

Construct Transition Matrix

```
function construct_transition_matrix(param, a_pol)
    @unpack Na, Ny, yg, ytran,ag = param
    transition_matrix = zeros(Na*Ny,Na*Ny)
    for ia = 1:Na
        for iy = 1:Ny
            index_ia_iy = compute_index_ia_iy(param,ia,iy)
            a_next = a_pol[ia,iy]
            left_grid,right_grid,left_weight,right_weight = find_nearest_grid(ag,a_next)
            for iy_next = 1:Ny
                index_ia_iy_next = compute_index_ia_iy(param,left_grid,iy_next)
                transition_matrix[index_ia_iy,index_ia_iy_next] += left_weight*ytran[iy,iy_next]

                index_ia_iy_next = compute_index_ia_iy(param,right_grid,iy_next)
                transition_matrix[index_ia_iy,index_ia_iy_next] += right_weight*ytran[iy,iy_next]
            end
        end
    end
    transition_matrix = sparse(transition_matrix)
    @assert sum(transition_matrix,dims=2) ≈ ones(Ny*Na)
    return transition_matrix
end
```

Always work with a sparse matrix!!

General Equilibrium

General Equilibrium

- We now endogenize interest rate r by moving to general equilibrium
- Assume bonds are in zero net supply so that

$$\int a d\mu = 0$$

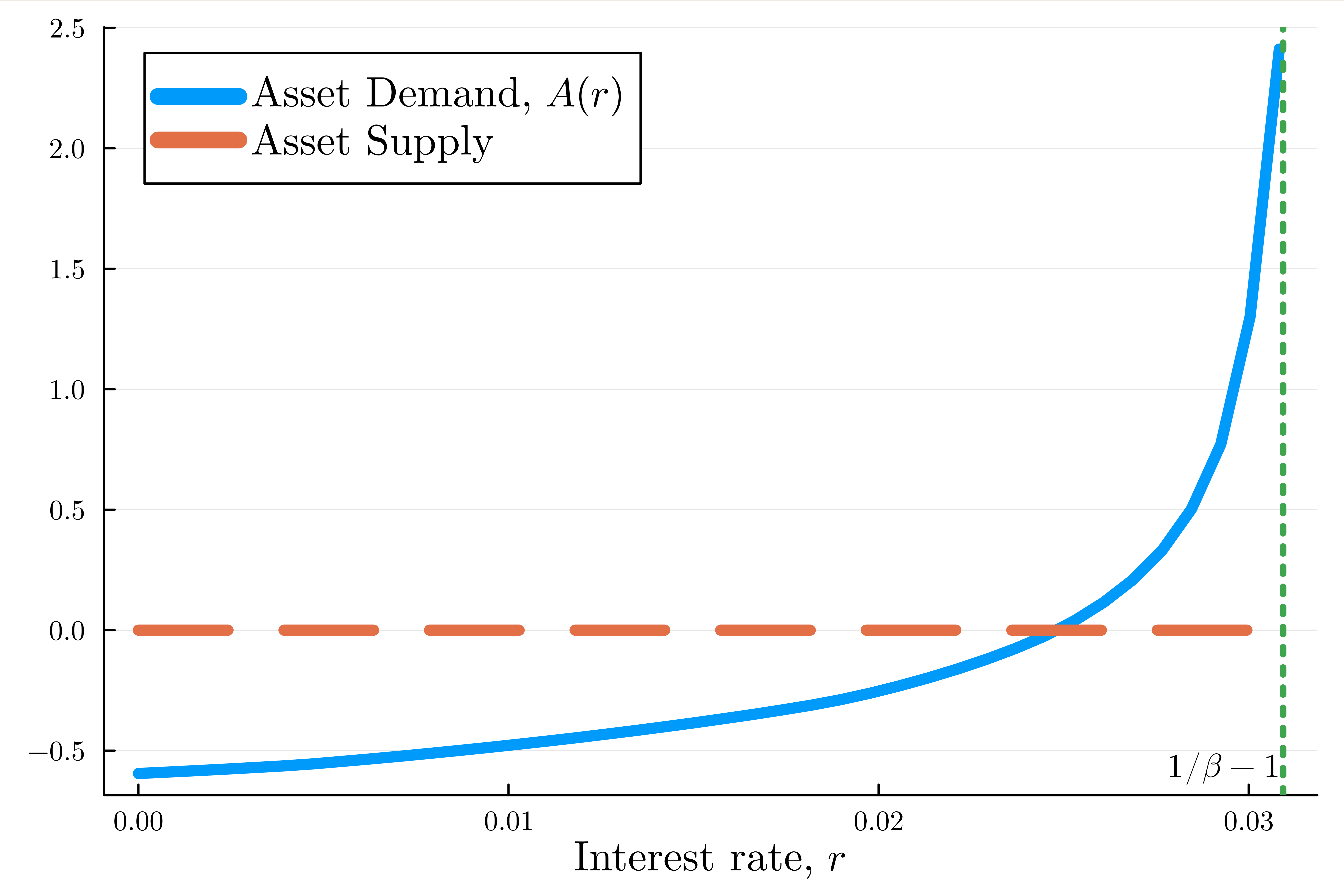
- Steady-state (recursive) equilibrium: $\{c(a,y), a'(a,y), V(a,y), \mu(a,y)\}$ and r such that
  1. Given r, $\{c(a,y), a'(a,y), V(a,y)\}$ solve household's Bellman equation
  2. $\{\mu(a,y)\}$ satisfies

$$\mu(a, y_j) = \sum_i \mu(a'^{-1}(a,y), y_i) \text{Prob}(y_j | y_i)$$

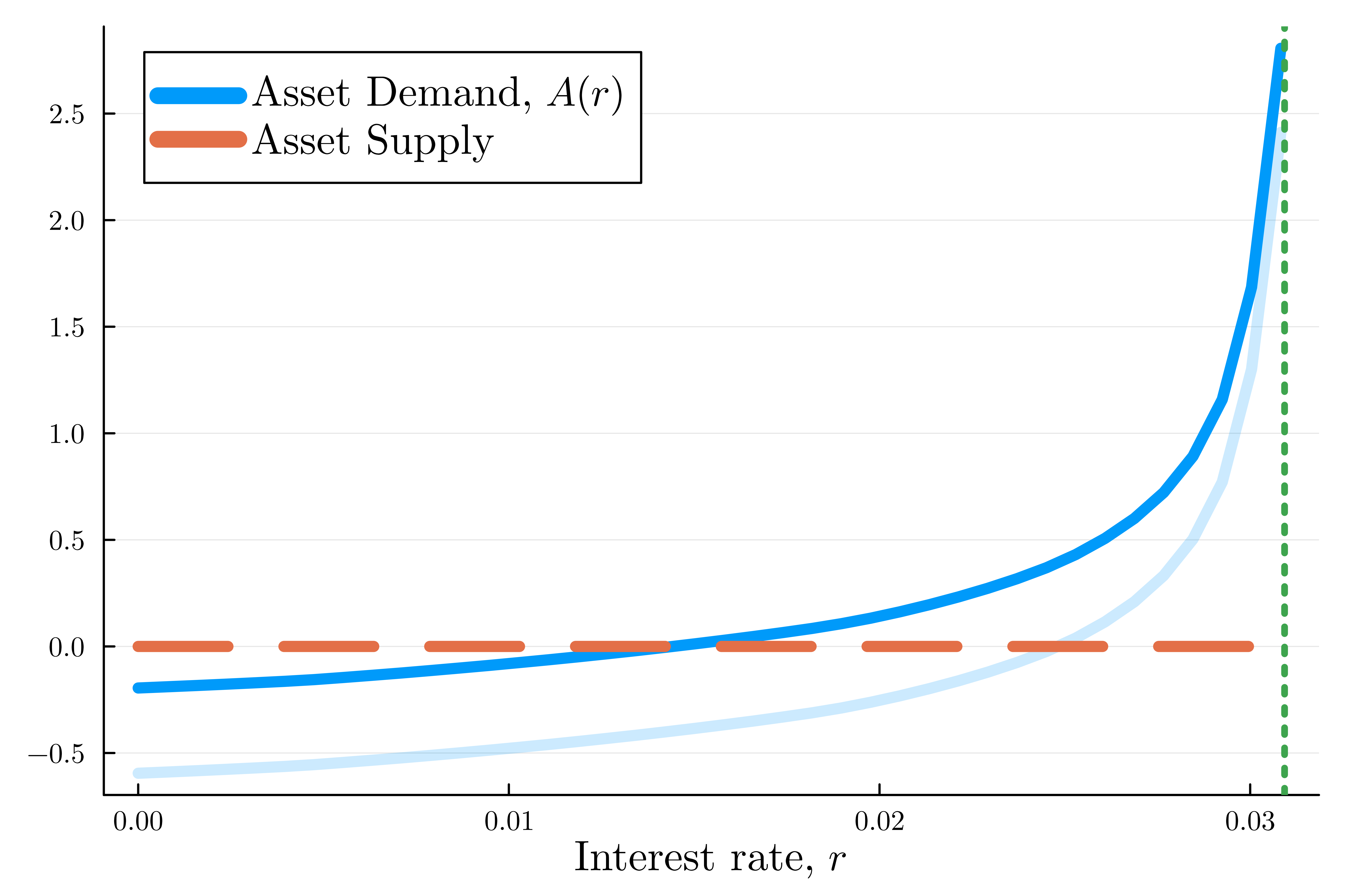
  3. Asset market clears:

$$\int a d\mu = 0$$

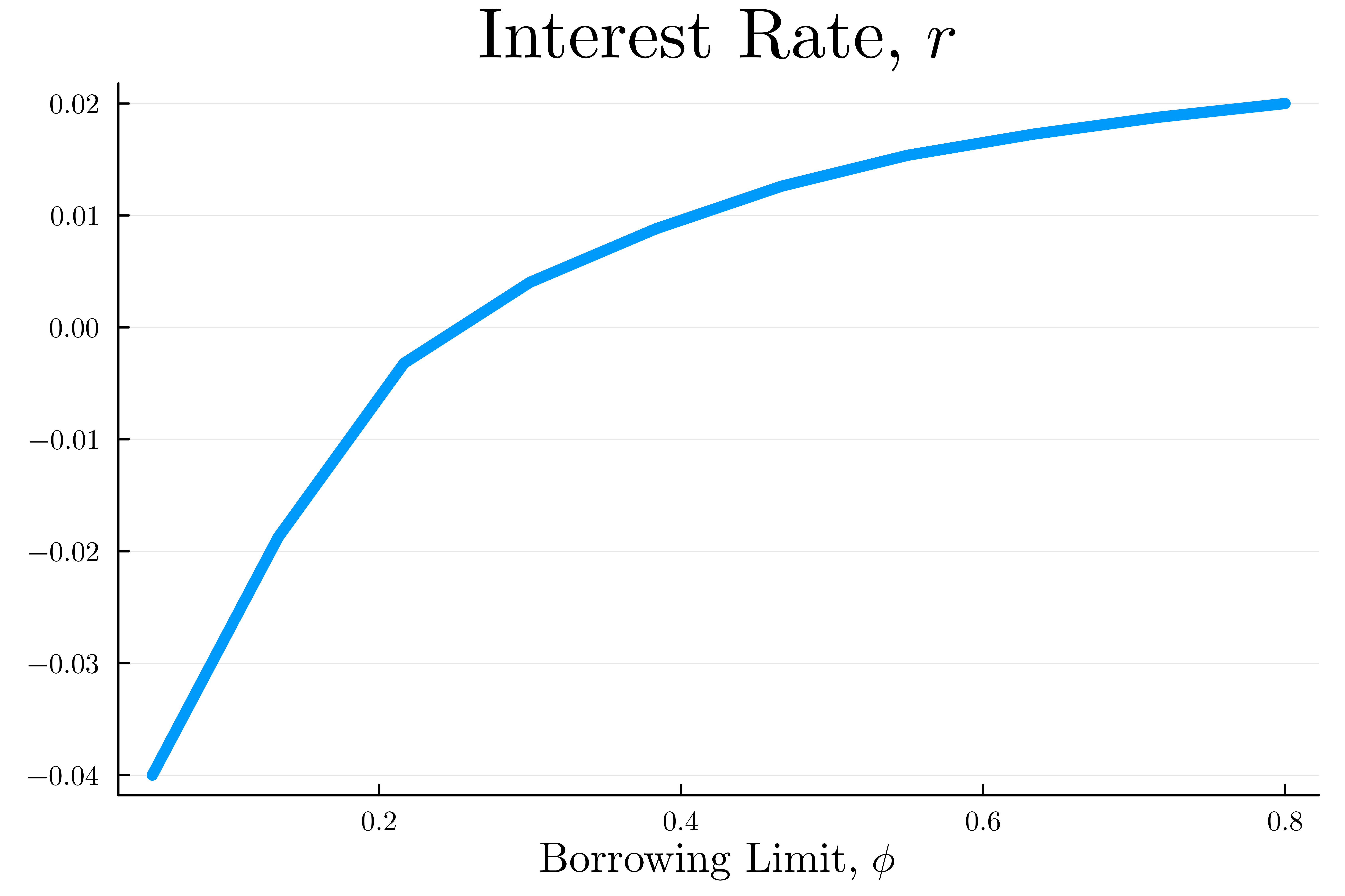
General Equilibrium



A Reduction in Borrowing Limit ϕ

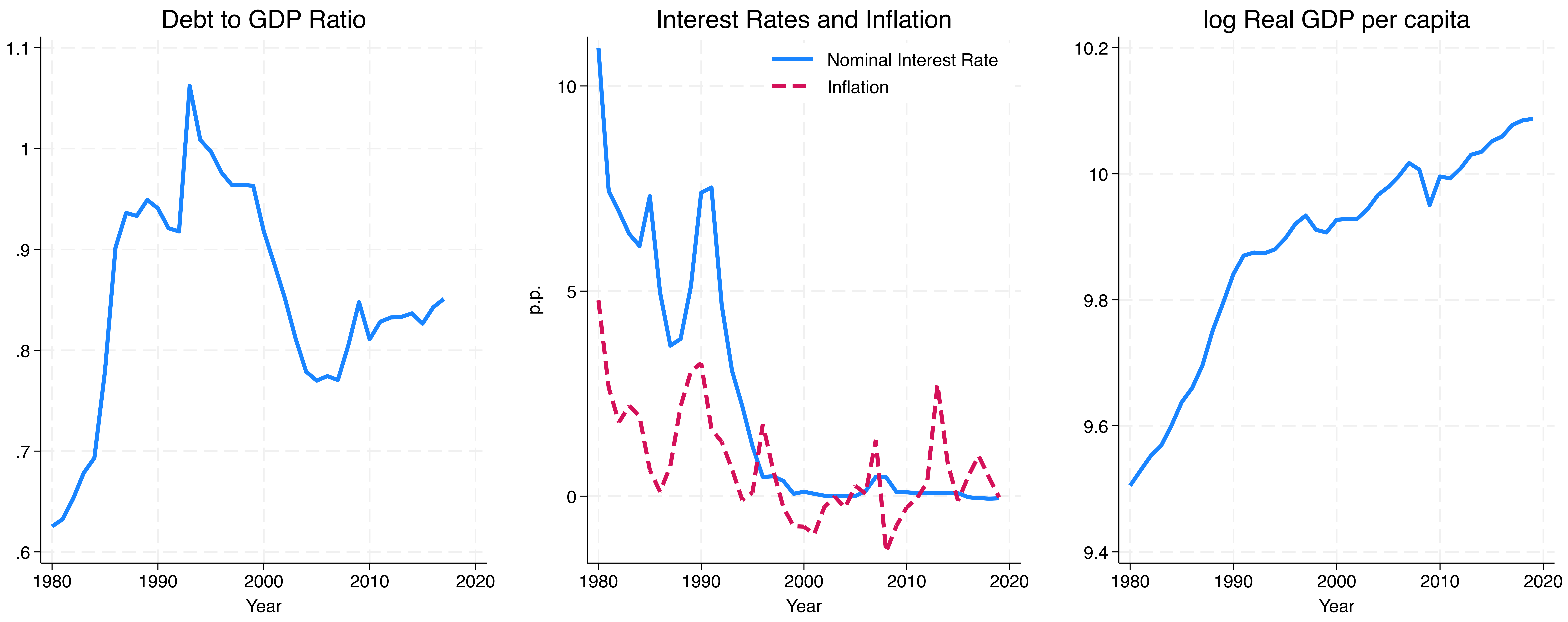


Tightening Borrowing Limit Depresses r

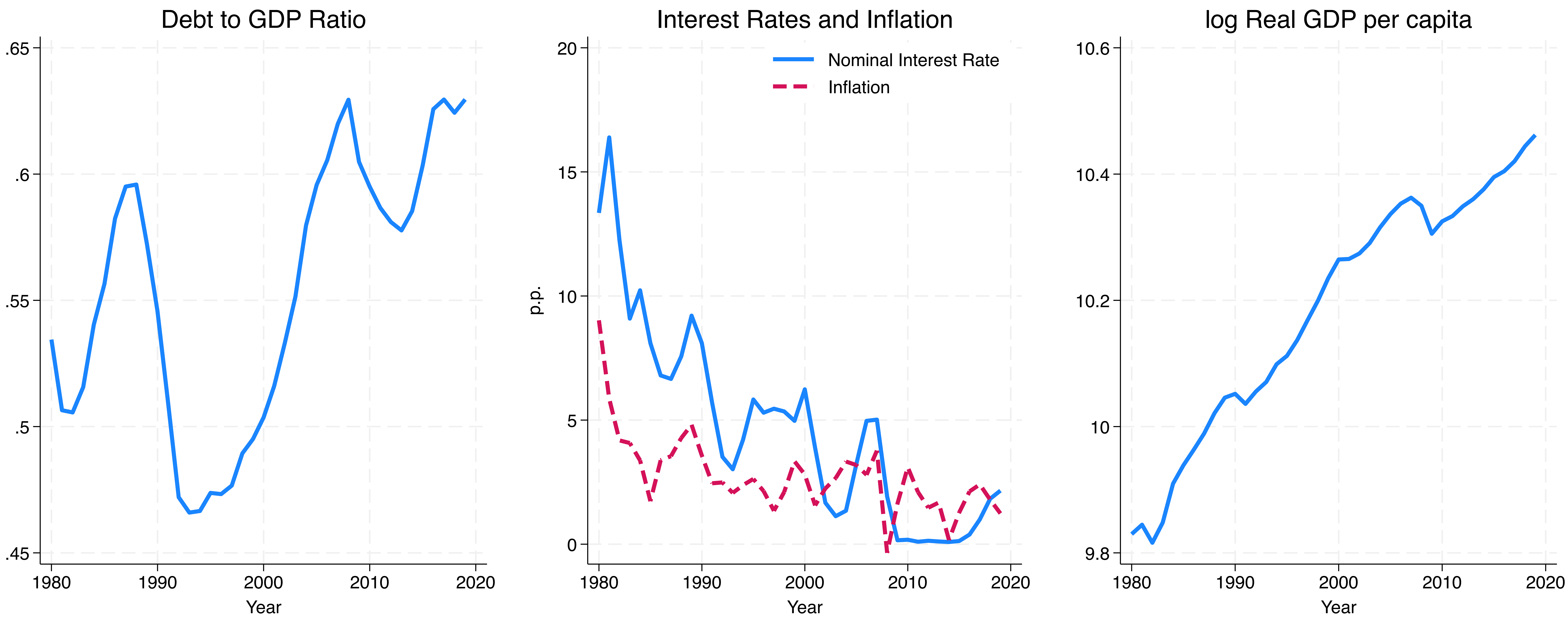


Borrowing Constraints and Aggregate Demand

Japan


Debt to GDP Ratio
1.1
1
.9
.8
.7
.6
1980
1990
2000
2010
2020
Year
Interest Rates and Inflation
Nominal Interest Rate
Inflation
p.p.
10
5
0
1980
1990
2000
2010
2020
Year
log Real GDP per capita
10.2
10
9.8
9.6
9.4
1980
1990
2000
2010
2020
Year

The US


Debt to GDP Ratio
.65
.6
.55
.5
.45
1980
1990
2000
2010
2020
Year
Interest Rates and Inflation
20
15
10
5
0
p.p.
Nominal Interest Rate
Inflation
1980
1990
2000
2010
2020
Year
log Real GDP per capita
10.6
10.4
10.2
10
9.8
1980
1990
2000
2010
2020
Year

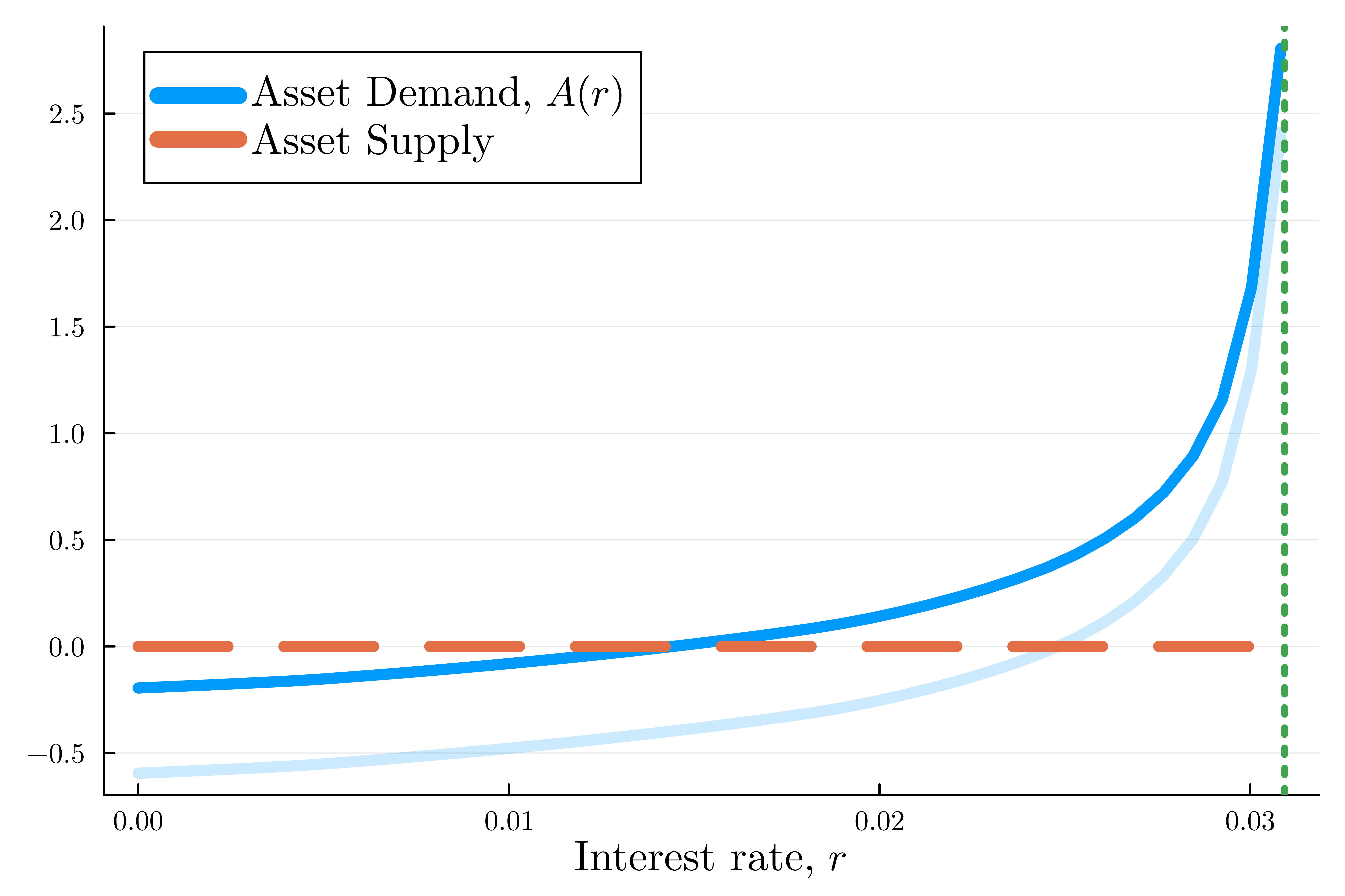
Demand Determined Equilibrium

- Now suppose that r is exogenously given
  - Assume sticky prices in the background
  - Monetary policy sets $r \equiv i - \pi$

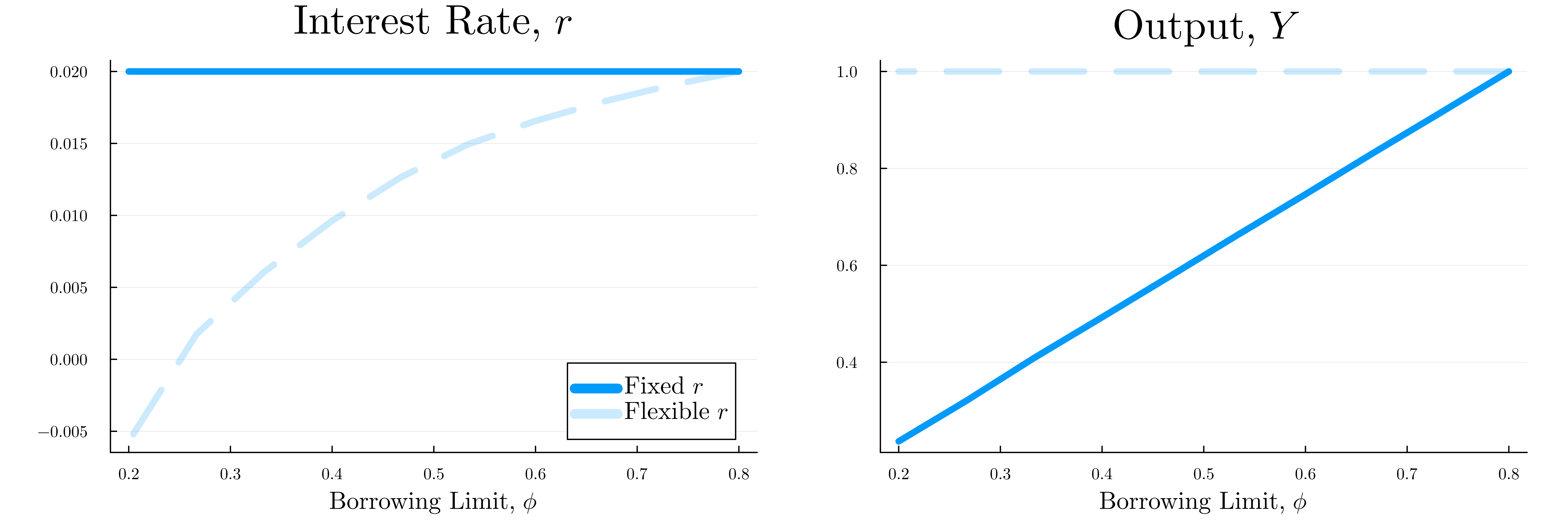
- Given r, steady-state *demand-determined* equilibrium consists of $\{c(a,y), a'(a,y), V(a,y), \mu(a,y)\}$ and Y such that

1. Given Y, $\{c(a,y), a'(a,y), V(a,y)\}$ solve household's Bellman equation
2. $\{\mu(a,y)\}$ satisfies
$$\mu(a, y_j) = \sum_i \mu(a'^{-1}(a,y), y_i)\text{Prob}(y_j | y_i)$$
3. Asset market clears:
$$\int a d\mu = 0$$

A Reduction in Borrowing Limit ϕ



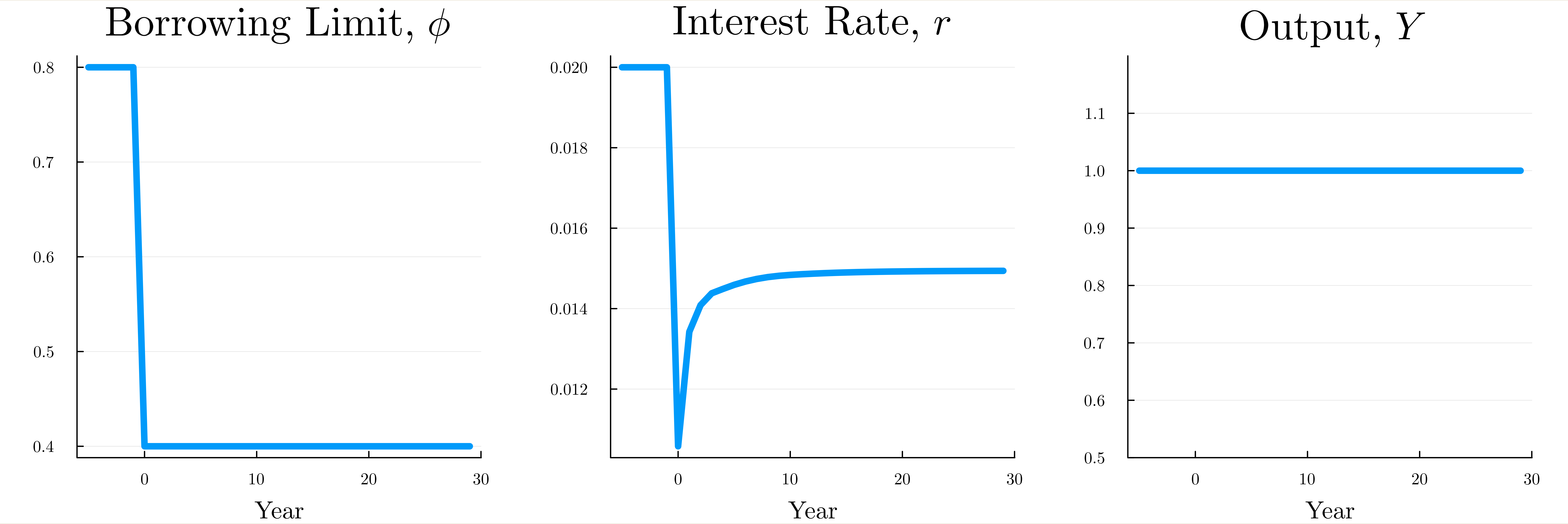
Secular Stagnation?



Short-Run Impact of Credit Crunch

– Guerrieri and Lorenzoni (2017)

Flexible Interest Rate



Rigid r

Borrowing Limit, ϕ
0.8
0.7
0.6
0.5
0.4
0
10
20
30
Year
Interest Rate, r
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
10
20
30
Year
Output, Y
1.0
0.8
0.6
0.4
0.2
0
10
20
30
Year

Sequence Space Jacobian Method

- When I was a PhD student, obtaining the transition dynamics took hours
- With the recent advancements, now it takes me less than a second
- See Auclert, Bardóczy, Rognlie, & Straub (2021)
    - I also have Julia implementation on the GitHub page (co-written with Aru)

Appendix:
Nomarilizing Asset

Normalizing Asset

- Computationally, often convenient to rewrite

$$V_t(a_t, y_t) = \max_{c_t, a_{t+1} \geq -\phi_t} u(c_t) + \beta \mathbb{E}_t V_{t+1}(a_{t+1}, y_{t+1})$$

$$\text{s.t.} \quad c_t + a_{t+1} = (1 + r_{t-1})a + y_t$$

$$\Downarrow$$

$$V^{\phi}(a_t^{\phi}, y_t) = \max_{c_t, a_{t+1}^{\phi} \geq 0} u(c_t) + \beta \mathbb{E}_t V_{t+1}^{\phi}(a_{t+1}^{\phi}, y_{t+1})$$

$$\text{s.t.} \quad c_t + a_{t+1}^{\phi} - \phi_t = (1 + r_{t-1})a^{\phi} - (1 + r)\phi_{t-1} + y_t$$

were $a_t^{\phi} \equiv a_t + \phi_t$ is the asset level relative to the borrowing limit