Macroeconomic Theory - Overlapping Generations (OLG) Households in General Equilibrium Framework

Darapheak Tin

Research School of Economics,
Australian National University

Sample Teaching Slides II

Table of Contents

Environment

- Time is discrete.
- Closed economy with a representative competitive firm.
- Competitive market: Prices (w, r) are taken as given by households.
- Overlapping generations (OLG) of households:
  - Each period, a new generation is born and lives for two periods (young $\rightarrow$ old).
  - Young supply inelastic labour (one unit).
  - Old retire.

Household problem

A household earns wage income w in period 1, chooses (c_1, c_2) to maximize lifetime utility:

$$\max_{c_1,\, s,\, c_2} \left\{ \frac{c_1^{1-\sigma}}{1-\sigma} + \beta \frac{c_2^{1-\sigma}}{1-\sigma} \right\}$$

subject to

$$c_1 + s = w,$$
$$c_2 = (1+r)\, s,$$
$$c_1,\ c_2 > 0$$

Firm and pricing

A representative firm with Cobb-Douglas technology. The firm's profit maximization problem is:

$$\max_{K,L}\{AK^{\alpha}N^{1-\alpha} - wN - qK\}$$

and the net interest rate is

$$r = q - \delta, \qquad \delta \in (0,1).$$

Baseline calibration (for simulations later)

Parameter		**Value**
Discount factor	β	0.98
Risk aversion	σ	2
TFP level	A	1
Capital share	α	0.33
Depreciation	δ	0.05

Firm Problem

Firm chooses (K, N) to maximize profit:

$$\max_{K,N} \left\{ AK^{\alpha} N^{1-\alpha} - wN - qK \right\}.$$

FOCs:

$$w = (1-\alpha) AK^{\alpha} N^{-\alpha},$$
$$q = \alpha AK^{\alpha-1} N^{1-\alpha}.$$

Lagrangian Approach - Household Optimization (1/2)

Household problem:

$$L(c_1, c_2, \lambda) = \max_{c_1, c_2, \lambda} \left\{ \frac{c_1^{1-\sigma}}{1-\sigma} + \beta \frac{c_2^{1-\sigma}}{1-\sigma} + \lambda \Big(w - c_1 - \tfrac{c_2}{1+r} \Big) \right\}.$$

First-order conditions (FOCs):

$$\begin{aligned}
\frac{\partial L}{\partial c_1} &: \quad c_1^{-\sigma} - \lambda = 0 \\
\frac{\partial L}{\partial c_2} &: \quad \beta c_2^{-\sigma} - \tfrac{\lambda}{1+r} = 0 \\
\frac{\partial L}{\partial \lambda} &: \quad w - c_1 - \tfrac{c_2}{1+r} = 0
\end{aligned}$$

Combining:

$$c_1 = \lambda^{-\frac{1}{\sigma}}, \qquad c_2 = \lambda^{-\frac{1}{\sigma}} \left(\tfrac{1}{R\beta} \right)^{-\frac{1}{\sigma}}$$

where $R = 1 + r$.

Lagrangian Approach - Household Optimization (2/2)

Plugging into the budget constraint:

$$\lambda^{-\frac{1}{\sigma}} = \frac{w}{\left[1 + \frac{1}{R}\left(\frac{1}{R\beta}\right)^{-\frac{1}{\sigma}}\right]}.$$

Therefore, the optimal consumption-saving rule is

$$c_1 = \frac{w}{\left[1 + \frac{1}{R}\left(\frac{1}{R\beta}\right)^{-\frac{1}{\sigma}}\right]}$$

$$s = w - c_1$$

Lagrangian Approach - Competitive Equilibrium

The CE is characterized by the following system of equations:

$$\lambda^{-\frac{1}{\sigma}} = \frac{w}{\left[1 + \frac{1}{R}\left(\frac{1}{R\beta}\right)^{-\frac{1}{\sigma}}\right]}, \tag{1}$$

$$c_1 = \lambda^{-\frac{1}{\sigma}}, \tag{2}$$

$$c_2 = \lambda^{-\frac{1}{\sigma}}\left(\frac{1}{R\beta}\right)^{-\frac{1}{\sigma}}, \tag{3}$$

$$s = w - c_1, \tag{4}$$

$$N = 1, \tag{5}$$

$$K = s, \tag{6}$$

$$w = (1-\alpha)AK^{\alpha}N^{-\alpha}, \tag{7}$$

$$R = 1 + \alpha AK^{\alpha-1}N^{1-\alpha} - \delta, \tag{8}$$

$$Y = AK^{\alpha}N^{1-\alpha}. \tag{9}$$

Exogenous variables: $\{\beta, \sigma, A, \alpha, \delta\}$.
Endogenous variables: $\{c_1, c_2, s, w, r, K, N, Y\}$.

Dynamic Programming Approach - Household (1/4)

We turn the household problem into a recursive problem. For the general case, we have:

$$V_t(a_t) = \max_{c_t,\, a_{t+1}} \left\{ \frac{c_t^{1-\sigma}}{1-\sigma} + \beta V_{t+1}(a_{t+1}) \right\}.$$

s.t.

$$\begin{aligned} c_t + \overbrace{a_{t+1}}^{=s_t} &= \overbrace{y_t}^{=w_t} + Ra_t && \text{when young (working life)} \\ c_t + a_{t+1} &= Ra_t && \text{when old (retirement)} \end{aligned}$$

where a_t is asset holding (asset holding today a_t = savings from yesterday s_{t-1}). Key assumptions:

- Born with no asset: $a_1 = 0$
- Terminal conditions:
  $V_{T+1}(a_{T+1}) = 0 \quad \Rightarrow \quad V_T(a_T) = u(\underbrace{(1+r)a_T}_{c_T})$

Dynamic Programming Approach - Household (2/4)

For our specific case where the households live for two period (young and old), the recursive formulation can be expressed as:

$$\textbf{Period 1}: \; V_1(a_1) = \max_{c_1,\, a_2} \left\{ \frac{c_1^{1-\sigma}}{1-\sigma} + \beta V_2(a_2) \right\} \quad \text{s.t. } c_1 + a_2 = y_1 + R \underbrace{a_1}_{=0}$$

$$\textbf{Period 2}: \; V_2(a_2) = \max_{c_2,\, a_3} \left\{ \frac{c_2^{1-\sigma}}{1-\sigma} + \beta \underbrace{V_3(a_3)}_{=0} \right\} \quad \text{s.t. } c_2 + a_3 = \underbrace{y_2}_{=0} + Ra_2$$

Assumptions: $a_1 = 0$ (born with no asset), $y_1 = w$, $y_2 = 0$, $R = 1 + r$, and life ends after two periods so $V_3(a_3) \equiv 0$.

We proceed to solve the household problem by *backward induction*: first period 2 (consume all), then period 1 (choose a_2).

Dynamic Programming Approach - Household (3/4)

The Bellman equation in period 2 is:

$$V_2(a_2) \;=\; \max_{c_2} \frac{c_2^{1-\sigma}}{1-\sigma} \quad \text{s.t. } c_2 = (1+r)a_2.$$

- No income in period 2 ($y_2 = 0$) and no savings in period 3 ($a_3 = 0$)
- Thus, optimal consumption $c_2 = (1+r)a_2$, and the value function is

$$V_2(a_2) = \frac{[(1+r)a_2]^{1-\sigma}}{1-\sigma}$$

- Marginal value function:

$$\frac{\partial V_2(a_2)}{\partial a_2} = (1+r)^{1-\sigma}[a_2]^{-\sigma}$$

Dynamic Programming Approach - Household (4/4)

Given $V_2(a_2)$, Bellman equation in period 1 is:

$$V_1(a_1) = \max_{a_2} \left\{ \frac{(\overbrace{y_1 - a_2}^{c_1})^{1-\sigma}}{1-\sigma} + \beta \overbrace{\frac{[(1+r)a_2]^{1-\sigma}}{1-\sigma}}^{V_2(a_2)} \right\}$$

Taking FOC wrt a_2:

$$-(y_1 - a_2)^{-\sigma} + \beta(1+r)^{1-\sigma}(a_2)^{-\sigma} = 0.$$

Solving the FOC, the optimal plan is:

$$\begin{aligned} a_2 &= \left[\frac{(\beta R)^{1/\sigma}}{R + (\beta R)^{1/\sigma}} \right] y_1 \\ c_1 &= y_1 - a_2 \\ c_2 &= R a_2 \end{aligned}$$

where $R \equiv (1+r)$.

Dynamic Programming Approach - CE

Competitive equilibrium characterized by:

$$
\begin{aligned}
\overbrace{a_2}^{=s_1} &= \left[\frac{(\beta R)^{1/\sigma}}{R + (\beta R)^{1/\sigma}}\right] y_1, \\
c_1 &= y_1 - a_2, \\
c_2 &= R a_2, \\
N &= 1, \\
K &= a_2, \\
w &= (1-\alpha) A K^{\alpha} N^{-\alpha}, \\
R &= 1 + \alpha A K^{\alpha-1} N^{1-\alpha} - \delta, \\
Y &= A K^{\alpha} N^{1-\alpha}.
\end{aligned}
$$

Exogenous: $\{\beta, \sigma, A, \alpha, \delta\}$
Endogenous: $\{c_1, c_2, a_2, w, R, K, N, Y\}$

Solution Method 1: `Fsolve`

`Fsolve` - `sol_GE_sys_residuals.m` (1/2)

We stack all equilibrium conditions as residuals $F(X) = 0$ and use a nonlinear solver `fsolve` to find the unknowns X:

Unknowns: $X = \left[c_1,\ c_2,\ s,\ w,\ r,\ K,\ N,\ Y\right]^\top$
Parameters: $\{\beta, \sigma, A, \alpha, \delta\}$

Residual system $F(X) = 0$:

$$
\begin{aligned}
&F_1 : \ \beta c_2^{-\sigma} - \frac{c_1^{-\sigma}}{1+r} = 0 && \text{(Euler)}\\
&F_2 : \ c_1 + \frac{c_2}{1+r} - w = 0 && \text{(Budget)}\\
&F_3 : \ s + c_1 - w = 0 && \text{(Savings)}\\
&F_4 : \ w - (1-\alpha)AK^\alpha N^{-\alpha} = 0 && \text{(Firm: wage)}\\
&F_5 : \ r - \alpha AK^{\alpha-1} N^{1-\alpha} + \delta = 0 && \text{(Firm: interest)}\\
&F_6 : \ N - 1 = 0 && \text{(Labour market)}\\
&F_7 : \ K - s = 0 && \text{(Capital market)}\\
&F_8 : \ Y - AK^\alpha N^{1-\alpha} = 0 && \text{(Production)}
\end{aligned}
$$

Fsolve - sol_GE_sys_residuals.m (2/2)

```
function [c_1,c_2,s,w,r,K,N,Y] = sol_GE_sys_residuals(beta,sigma,A,alpha
    ,delta,X0);

options = optimset('Display', 'off'); % Turn off Display
fsolve(@cFOCs_f, X0, options);        % call fsolve

  function F = cFOCs_f(X)            % define the function F with X
        % unpack input arguments
        c_1  = X(1);    c_2  = X(2);
        s    = X(3);
        w    = X(4);    r    = X(5);
        K    = X(6);    N    = X(7);
        Y    = X(8);
        % system of equilibrium equations (residuals)
        F(1)      = beta*c_2^(-sigma) - c_1^(-sigma)/(1+r);
        F(2)      = c_1 + c_2/(1+r)- w;
        F(3)      = s + c_1 - w;
        F(4)      = w - (1-alpha)*A*K^alpha*N^(-alpha);
        F(5)      = r - alpha*A*K^(alpha-1)*N^(1-alpha) + delta;
        F(6)      = N - 1;
        F(7)      = K - s;
        F(8)      = Y - A*K^alpha*N^(1-alpha);
  end
end
```

Fsolve - sol_GE_Fsolve.m

```
% ---- MAIN SCRIPT ---- %
clear; close all; clc

% Parameters (one period = 30 years)
beta  = 0.95^30;
sigma = 2;
A     = 1;
alpha = 0.33;
delta = 1 - (1-0.05)^30;

% Initial guess for endo. vars: [c1 c2 s w r K N Y]
X0 = [0.5, 0.5, 0.2, 0.8, 0.02, 0.2, 1.0, 1.0];

% Solve
[c_1,c_2,s,w,r,K,N,Y] = ...
sol_GE_sys_residuals(beta,sigma,A,alpha,delta,X0);

% Print results
disp('---- Results--------------- ');
disp(['Y   =' num2str(Y)]);
disp(['K   =' num2str(K)]);
disp(['N   =' num2str(N)]);
disp(['R   =' num2str((1+r)^(1/30))]); %annualised gross return
disp(['w   =' num2str(w)]);
disp('------------------------- ');
```

Solution Method 2: Gauss–Seidel (G-S)

Gauss–Seidel Algorithm (Step-by-Step)

Objective: Instead of relying on `fsolve` to solve the entire system of CE conditions, we solve the model by iterating between household and firm decisions until capital converges.

1. **Initial guess:** Pick a starting value for capital $K^{(0)}$ and compute implied market prices: w and R.
2. **Household problem:** Given (w, R), solve for optimal savings s_1 (or equivalently asset holdings a_2).
3. **Update capital:** $K^{(i+1)} = s_1^{(i)}$ or $K^{(i+1)} = a_2^{(i)}$ and recompute market prices using the firm's FOCs based on the updated $K^{(i+1)}$.
4. **Check convergence:** Compare new and old capital levels:
$$\text{Error} = \left| K^{(i+1)} - K^{(i)} \right|$$
5. **Stopping rule:** If error $<$ tolerance (e.g., 10^{-3}), stop. Otherwise, return to step 2.

Intuition: The algorithm iteratively adjusts K until households' saving decisions are consistent with firms' capital demands (i.e., the goods and factor markets clear).

Gauss–Seidel - sol_GE_GaussSeideil_DP_Lagr.m (1/4)

```
clear all; close all
tic
disp('--------- New run ---------');
% Parameter values
beta    = 0.95^30; %0.9324^30; % 0.9324
sigma   = 2;
A       = 1;
alpha   = .33;
delta   = 1 - (1-.05)^30;

% Initials
Kold    = .01;
Nold    = 1;
w       = (1-alpha) * A * Kold^alpha * Nold^(-alpha);
R       = 1 + alpha * A * Kold^(alpha-1) * Nold^(1-alpha) - delta;

% for iteration
error   = 100;
errorv  = 100;
iter    = 0;
itermax = 50;
tol     = .001;
update  = .5;
```

Household - `sol_GE_GaussSeideil_DP_Lagr.m`

Given w, R, the household chooses c_1, c_2, s_1, using one of the three methods (user's choice):

- **Method 0 (Lagrangian):** Solve FOCs analytically.
- **Method 1 (DP with FOC):** Closed-form savings rule from Euler equation obtained through solving DP analytically.
- **Method 2 (DP with Value Function):** Discretise asset space, search for s_1 that maximises V_1.

Gauss–Seidel - sol_GE_GaussSeideil_DP_Lagr.m (2/4)

```
while (iter<itermax)&&(error>tol)
    % [1.] Solving the household problem (3 different methods)
    I_DP = 2;
    %
    if I_DP==0     % Lagrangian using FOCs
       lambda_sig=w/(1+(1/R)*(1/(R*beta))^(-1/sigma));
       c_1 = lambda_sig; c_2 = lambda_sig*(1/(R*beta))^(-1/sigma);
       s_1 = w - c_1;
    elseif I_DP==1 % Dynamic programming using FOCs
       s_1  = (R*beta)^(1/sigma)/(R+(R*beta)^(1/sigma))*w; % a_2 = s_1
       c_1 = w - s_1; c_2 = R*s_1;
    elseif I_DP==2 % Dynamic programming method using value function
       % Asset space (a2 = s1), step size = w/100
       s1v  = [0:w/100:2*w];  % coarser grid
       %s1v = [0:w/1000:2*w]; % finer grid
       V2v = (R*s1v).^(1-sigma)/(1-sigma); % Value over the asset space
       c1v  = w - s1v; % possible choices for consumption 1
       % find value function for V1
       V1v = (c1v>0).*c1v.^(1-sigma)/(1-sigma) + (c1v<=0).*(-10^10) +
            beta*V2v;
       % find the max value of the value function and optimal saving
       [val,pos] = max(V1v);
       s_1 = s1v(pos); % optimal saving/asset
       c_1 = w - s_1;  c_2 = R*s_1;
    end
```

Gauss–Seidel - sol_GE_GaussSeideil_DP_Lagr.m (3/4)

```
    % [.2] Clearing the labor and capital markets
    N      = 1;   % inelastic labor
    Knew   = s_1; % capital

    % Use onvex updating rule for capital for stabiity
    K      = update*Kold + (1-update)*Knew;

    % [.3] Using the firm's FOCs to pin down factor prices
    w      = (1-alpha)*A*K^alpha*N^(-alpha);
    q      = alpha*A*K^(alpha-1)*N^(1-alpha);
    % Interest rate
    r      = q - delta;
    R      = 1 + r;
    % Output
    Y      = A*K^(alpha)*N^(1-alpha);

    % [.4] the convergence condition and updates for next interation
    error = 100*abs(K - Kold)/Kold; % error in percentage
    errorv = [errorv error];
    %
    Kold   = K;
    iter   = iter+1;
end
```

Gauss–Seidel - sol_GE_GaussSeideil_DP_Lagr.m (4/4)

```
disp('---- Results---------------- ');
disp(['Y   =' num2str(Y)]);
disp(['K   =' num2str(K)]);
disp(['N   =' num2str(N)]);
disp(['R   =' num2str(R^(1/30))]); % annualised
disp(['w   =' num2str(w)]);
disp(['error=' num2str(error)]);
disp('-------------------------- ');

toc
```

G–S - Analytical vs. DP (Numerical) Household Sol. (1/2)

	(i) Analytical	(ii) DP, step size $= \frac{w}{100}$	(iii) DP, step size $= \frac{w}{1000}$
Y	0.40049	0.39807	0.40062
K	0.062477	0.061342	0.062540
N	1.00000	1.00000	1.00000
R	1.02860	1.02900	1.02860
w	0.26833	0.26671	0.26842
error	0.00086253	0.00076821	0.00079105
Runtime (s)	0.039171	0.042734	0.051140

- (i) Household problem solved analytically (`I_DP==0/1`)
- (ii) and (iii) solved by value search on discretized asset grid (`I_DP==2`)
- (i) is nice, but many problems have no closed-form solution $\rightarrow$ use (i)/(ii)
- All runs use the same Gauss–Seidel relaxation, tolerance, and production parameters

G–S - Analytical vs. DP (Numerical) Household Sol. (2/2)

Level differences are small but systematic:

- DP with coarser grid (ii) produces slightly lower K and Y relative to analytical (i).
- Finer grid (iii) brings DP back in line with analytical: K and Y essentially match (differences at $< 10^{-4}$).

Prices and wages move accordingly:

- With lower K in (ii), the wage w is a bit lower and R a touch higher (marginal products respond to capital).
- In (iii), w and R revert to analytical values.

Runtime:

- Runtime rises modestly with grid refinement.

Note

Intuition: Coarser grids tend to lead low-asset households to *under-save* due to fewer saving choices. This biases K and Y downward.

Trade-off: Finer grids improve accuracy but increase computational burden (slower).

Solution: A potential solution is using non-uniform grid.

Supplementary Material

Motivation: Why Non-Uniform Grids?

Uniform grids: $a_i = a_{\min} + i \times \Delta a$ (for $i = 0, 1, \ldots, n-1$) use equal spacing. Simple but inefficient when:

- The value or policy function is highly curved near constraints/endpoints (e.g. borrowing limit).
- Optimal savings of low-asset agents can be tiny so coarse uniform grids may not offer this option, forcing them to consume everything (to avoid $c = 0$).

Solution: use non-uniform grids that place more points where accuracy matters (and fewer where it does not), improving accuracy without increasing number of grid points (computational cost).

We consider two discretization methods:

1. **Left-dense grid:** dense near $a_{\min}$ (borrowing constraint).
2. **Chebyshev-Lobatto grid:** dense near both $a_{\min}$ and $a_{\max}$ (Not applicable here).

Back to Main Section

Left-Dense (Power-Transformed) Grid (1/2)

Formula:

$$\mathrm{grid}(t) = a_{\min} + (a_{\max} - a_{\min})\, t^p, \quad t \in [0,1],\ p > 1$$

Code:

```
function grid = left_dense_grid(amin, amax, n, p)
% Left-dense non-uniform grid on [amin, amax]:
% (You need to set p>1 so points cluster near amin)
    t    = linspace(0,1,n);              % uniform in [0,1]
    grid = amin + (amax-amin)*(t.^p);    % convex warp clusters left
    grid(1)=amin; grid(end)=amax;        % exact endpoints
end
```

Intuition:

1. Start with a uniform grid $t = [0, \frac{1}{n-1}, ..., 1]$.
2. Apply a nonlinear stretch $t \mapsto t^p$.
3. Since $f(t) = t^p$ is convex for $p > 1$, points dense near $a_{\min}$ and sparse near $a_{\max}$.

Left-Dense (Power-Transformed) Grid (2/2)

Numerical illustration:

t	t^1 (linear)	t^2	t^3
0.00	0.00	0.00	0.00
0.25	0.25	0.06	0.02
0.50	0.50	0.25	0.13
0.75	0.75	0.56	0.42
1.00	1.00	1.00	1.00

Parameter choice and effects:

p	Spacing pattern	Typical use
1	uniform	baseline (equal spacing)
> 1	dense near $a_{\min}$	capture curvature near borrowing bound
$0 < p < 1$	dense near $a_{\max}$	capture behaviour at high wealth

Chebyshev-Lobatto Grid (1/2)

Goal: Create dense grid near both ends $[a_{\min}, a_{\max}]$.

Formula:

$$\xi_j = \cos\left(\pi \cdot \frac{j}{n-1}\right), \quad j = 0, 1, \ldots, n-1$$

$$a_j = \frac{a_{\min} + a_{\max}}{2} + \frac{a_{\max} - a_{\min}}{2}\,\xi_j.$$

Code:

```
function grid = cheb_lobatto_grid(amin, amax, n)
% ChebyshevâLobatto nodes on [amin, amax].
% (points bunch/dense near both ends)
    j   = 0:(n-1);
    xi  = cos(pi * j / (n-1));           % nodes on [-1,1]
    centre = 0.5*(amin+amax);
    g   =  centre + 0.5*(amax-amin)*xi;
    grid = sort(g,'ascend');             % ascending order
    grid(1)=amin;   grid(end)=amax;      % pin exact endpoints
end
```

Chebyshev-Lobatto Grid (2/2)

Let $n = 5$ and interval $[0, 10]$.

Step 1: equally spaced angles

$$\theta_j = j\frac{\pi}{4}, \quad j = 0, \dots, 4.$$

Step 2: take cosines

$$\xi = [1,\ 0.707,\ 0,\ -0.707,\ -1].$$

Step 3: map to $[0, 10]$

$$a = 5 + 5\xi = [10,\ 8.535,\ 5,\ 1.465,\ 0].$$

Step 4: sort ascending $\Rightarrow$ $[0,\ 1.465,\ 5,\ 8.535,\ 10]$.

Observation: short gaps near 0 and 10, wide gaps in the middle.

Why cosine creates endpoint clustering?

Cosine basics:

$$x = \cos(\theta), \quad \theta \in [0, \pi]$$

gives $x \in [-1, 1]$.

If we take equal angular steps in θ from $\theta_i = \{0, \frac{1}{n-1}\pi, \cdots, \frac{j-1}{n-1}\pi, \pi\}$, the corresponding x's are **not equally spaced**. In fact, $x(\theta_i)$ will be dense near both endpoints -1 and 1 and sparse around the centre.

Consider derivative:

$$\frac{dx}{d\theta} = -\sin\theta$$

- Near $\theta = 0$ or π, $\sin\theta \approx 0 \Rightarrow x$ changes slowly, causing points to **crowd near** ± 1.
- In the middle ($\theta \approx \pi/2$), $\sin\theta = 1$, causing x to change quickly and therefore points spaced wider.

Gauss–Seidel - sol_GE_GaussSeideil_DP_Lagr_v1.m

```
while (iter<itermax)&&(error>tol)
    ...
    ...
    elseif I_DP==2 % Dynamic programming method using value function

            % DIFFERENT DISCRETIZATION METHODS
            % --------------------------------
            % [a.] Uniform
            %s1v = make_grid('uniform', 0, 2*w, 201);
            %s1v = make_grid('uniform', 0, 2*w, 2001);

            % [b.] Growing grid (left dense)
            s1v = make_grid('left_dense', 0, 2*w, 201);
            %s1v = make_grid('left_dense', 0, 2*w, 2001);

            % [c.] Chebyshev (dense near both endpoints)
            % (Not suitable for this problem)
            %s1v = make_grid('cheb', 0, 2*w, 201);

            % ----------------------------------------
            V2v = (R*s1v).^(1-sigma)/(1-sigma);
            ...
            ...
            ...
        end
```