Course 4 — Dynamic Programming, Optimal Growth, and Aiyagari

Short Macroeconomics Course Using Julia

November 27, 2025

Outline

Dynamic Programming I

- There are two alternative representations of the dynamic programming approach: a sequential one and a functional one. They are equivalent.

$$V^*(x_0) = \sup_{\{x_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t U(t, x_t, x_{t+1}) \tag{1}$$

- This is subject to the following constraint:

$$x_{t+1} \in G(x_t), \tag{2}$$

- with x_0 given. Here $G(x)$ is a correspondence, mapping a value into a set. The function $U(x)$ is the instantaneous utility, while β is the discount factor. We distinguish between x_t, a state variable at moment t, since its value is known before t given $G(x_{t-1})$, and x_{t+1}, a control variable at t.

Dynamic Programming II

- The alternative representation, which is actually preferable when solving a dynamic programming problem, is that of a functional equation (FE, hereafter).
  Functional Equation

$$V(x) = \sup_{y \in G(x)} \left[U(x, y) + \beta V(y) \right] \tag{3}$$

- Here, $V(x)$ is the value function. In this formulation, we do not focus on choosing the sequence $\{x_t\}_{t=0}^{\infty}$, but on choosing a policy function through which we can determine x_{t+1} given x_t.

Contraction Mapping

- Under standard assumptions (bounded u, $\beta \in (0,1)$) the Bellman operator is a contraction in sup norm:

$$\|\mathcal{T}V - \mathcal{T}W\|_\infty \leq \beta \|V - W\|_\infty.$$

- Unique fixed point $V^\star$; iterate $V_{k+1} = \mathcal{T}V_k$.

Stochastic DP: Discrete Markov Shocks

- If z takes N_z values with transition matrix P, then

$$V(s,z) = \max_a \left\{ u(s,a,z) + \beta \sum_{z'} P_{zz'} V(s',z') \right\}.$$

Numerical Ingredients

- Grids: linear, log, Chebyshev (nonuniform where curvature bites).
- Interpolation: linear, monotone cubic; projection methods for high accuracy.
- Acceleration: Howard policy iteration, multigrid, Newton steps on Euler equations.

Convergence and Stopping

- Tolerance on value: $\|V_{k+1} - V_k\|_\infty < \epsilon_V$.
- Or policy tolerance: share of states where policy changes $< \epsilon_\pi$.
- Practical: use both, cap iterations.

Planner's Problem (Deterministic)

- Assume a one-sector economy in which there is a representative household. The instantaneous utility function is $u()$ while the discount factor $\beta \in (0,1)$. The representative household maximizes the discounted lifetime utility:

$$\max_{\{c_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u(c_t) \tag{4}$$

- Each period, the household face the following constraint (combining the household budget constraint with the capital stock dynamics):

$$k_{t+1} = f(k_t) + (1-\delta)k_t - c_t \tag{5}$$

- It is further assumed that the initial capital stock k_0 is given while $k_t \geq 0$.
- For this specific model, we can write the Bellman equation as:

$$V(k_t) = \max_{c_t}(u(c_t) + \beta V(k_{t+1})) \tag{6}$$

OGM - Implementation I

- I focus on the optimal growth model and use the following specification for the utility function:

$$u(c) = \frac{c^{1-\sigma} - 1}{1-\sigma} \tag{7}$$

- While the next period k' capital stock can be written as:

$$k' = k^{\alpha} - c + (1-\delta)k \tag{8}$$

- This implies that the Bellman equation becomes now:

$$V(k) = \max_{c} \frac{c^{1-\sigma} - 1}{1-\sigma} + \beta V(k') \tag{9}$$

OGM - Implementation II

- But consumption can be replaced using: $c = k^\alpha + (1-\delta)k - k'$. We rewrite the Bellman equation as:

$$V(k) = \max_{k'} \frac{(k^\alpha + (1-\delta)k - k')^{1-\sigma} - 1}{1-\sigma} + \beta V(k') \tag{10}$$

- For each i point on the grid, we evaluate the value function:

$$V_{i,h} = \frac{(k_i^\alpha + (1-\delta)k_i - k'_h)^{1-\sigma} - 1}{1-\sigma} + \beta V(k_h) \tag{11}$$

VFI Algorithm (discrete controls)

1. Construct a grid of admissible values for the state variable k of the form: $\mathcal{K} = \{k_1, ..., k_n\}$
2. Propose a starting point for the value function $V_0(x)$ and set an approximation criterion ϵ.
3. Iterate on the grid such that for each $k_i \in \mathcal{K}$ with $(i = 1, .., n)$, we solve for: $V_{j+1}(k_i) = \max_{k'}(u(f(k_i, k'), k_i) + \beta V_j(k'))$
4. Stop if the error is smaller than the criterion ϵ, i.e.: $||V_{j+1}(k) - V_j(k)|| < \epsilon$
5. Determine the solution: $f^*(k) = f(k, k')$ and $V^*(k) = \frac{u(f^*(k),k)}{1-\beta}$

Howard Policy Improvement

- Fix policy $a(s)$ and solve the linear system for V implied by Bellman with fixed choices.
- Alternate: policy evaluation (fast) and improvement (re-optimize).

Julia - VFI I

```
sigma = 1.5;
delta = 0.1;
beta  = 0.95;
alpha = 0.30;
ks    = ((1-beta*(1-delta))/(alpha*beta))^(1/(alpha-1));
csy   = 1-alpha*beta*delta/(1-beta*(1-delta));
dev   = 0.9;
kmin  = (1-dev)*ks;
kmax  = (1+dev)*ks;
nbk   = 1000;
devk  = (kmax-kmin)/(nbk-1);
k     = collect(LinRange(kmin,kmax,nbk));
v0    = zeros(nbk,1);
v0    = ((csy*k.^alpha).^(1-sigma).-1)./((1-sigma)*(1-beta));
v     = zeros(nbk,1);
ik1   = zeros(nbk,1);
```

Julia - VFI II

```
while crit>tol;
for i=1:nbk
imin =  max(ceil(((1-delta)*k[i]-kmin)/devk)+1.0,1);
imax =  min(floor((k[i]^alpha+(1-delta)*k[i]-kmin)/devk)+1.0,nbk);

imin=trunc(Int, imin);
imax=trunc(Int, imax);
c = k[i]^alpha+(1-delta)*k[i].-k[imin:imax];
u = (c.^(1-sigma).-1)/(1-sigma);
(v[i],itmp)= findmax(u+beta*v0[imin:imax]);
ik1[i] = imin-1+itmp;
end;
error=abs.(v[:,1]-v0);
crit= maximum(error);
copy!(v0, v[:,1])
iter= iter+1;
end
```

Diagnostics

- Plot $V(s)$ and $a^{\star}(s)$; check monotonicity.
- Policy residuals (Euler eq.) on simulated paths.
- Sensitivity to grid density.

Main idea

- VFI converges slow, easy to implement.
- Howard improvement algorithm: instead of iterating on the value function, the algorithm iterates on the policy function.

Main idea

1. Start from an initial guess of the control variable $c = f_0(k)$ and compute the implied value function for this initial guess using: $V(k_t) = \sum_{s=0}^{\infty} \beta^s u(f_i(k_{t+s}), k_{t+s})$. Here, we use $k_{t+1} = h(k_t, c_t) = h(k_t, f_i(k_t))$.
2. We compute a new policy rule $c = f_{i+1}(k)$ that verifies: $f_{i+1}(k) \in argmax_c u(c, k) + \beta V(k')$. Here, we use $k' = h(k, f_i(k))$.
3. Using a parameter ϵ as the stopping criteria, we check whether: $||f_{i+1}(k) - f_i(k)|| < \epsilon$. If the condition is not fulfilled, the algorithm goes back to the second step.

Stochastic Dynamic Programming I

We consider the introduction of a random (stochastic) variable $z_t \in \mathcal{Z}$, where $\mathcal{Z} = \{z_1, z_2, ..., z_n\}$. It is assumed that the the set $\mathcal{Z}$ is finite. The utility function becomes now:

$$U(x_t, x_{t+1}, z_t) \tag{12}$$

Stochastic Dynamic Programming II

The random variable z_t is assumed to follow a Markov chain, implying that the current value z_t depends only on the value in the last period z_{t-1}. We could write this formally as:

$$P[z_t = z_j | z_0, ..., z_{t-1}] = P[z_t = z_j | z_{t-1}] \tag{13}$$

Stochastic Dynamic Programming III

The sequential representation

$$V^*(x_0, z_0) = \sup_{\{\tilde{x}[z^t]\}_{t=0}^{\infty}} \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t U(\tilde{x}[z^{t-1}], \tilde{x}[z^t], z_t) \tag{14}$$

Here $z^t = \{z_1, ..., z_t\}$ is the history of z_t up to date t, with z_0 considered as given.
This is subject to the following constraint:

$$\tilde{x}[z^t] \in G(\tilde{x}[z^{t-1}], z_t) \tag{15}$$

Stochastic Dynamic Programming IV

The alternative representation, which is actually preferable when solving a dynamic programming problem, is that of a functional equation. However there are two ways to achieve this. In the simplest form, we could use the expectation operator and write:
Functional Equation

$$V(x,z) = \sup_{y \in G(x)} \left[U(x,y,z) + \beta \mathbb{E}[V(y,z')|z] \right] \tag{16}$$

Julia VFI I - Stochastic

```
sigma    = 1.5;
delta    = 0.1;
beta     = 0.95;
alpha    = 0.30;
p        = 0.9;
PI       = [p 1-p;1-p p];
se       = 0.2;
ab       = 0;
am       = exp(ab-se);
as       = exp(ab+se);
A        = [am as];
nba      = 2;8
```

Julia VFI II - Stochastic

```
ks     = ((1-beta*(1-delta))/(alpha*beta))^(1/(alpha-1));
csy    = 1-alpha*beta*delta/(1-beta*(1-delta));
dev    = 0.9;
kmin   = (1-dev)*ks;
kmax   = (1+dev)*ks;
nbk    = 1000;
devk   = (kmax-kmin)/(nbk-1);
k      = collect(LinRange(kmin,kmax,nbk));
kp     = zeros(nbk, nba);
c      = zeros(nbk,nba);
u      = zeros(nbk,nba);
v      = zeros(nbk,nba);
Tv     = zeros(nbk,nba);
iter   = 1;
crit   = 1;
tol    = 1e-6;
```

Julia VFI III - Stochastic

```
#iterate on Value Function
while crit>tol
for i=1:nbk
for j=1:nba
c         = A[j]*k[i]^alpha+(1-delta)*k[i].-k;
neg       = findall(x -> x <=0.0,c);
c[neg].= NaN;
u[:,j]    = (c.^(1-sigma).-1)/(1-sigma);
u[neg,j].= -1e12;
end;
u_obj = u+beta*(v*PI)
(tv1,dr1) = findmax(u_obj[:,1], dims=1)
(tv2,dr2) = findmax(u_obj[:,2], dims=1)
dr[i,1] = dr1[1]; dr[i,2] = dr2[1];
Tv[i,1] = tv1[1]; Tv[i,2] = tv2[1];
end;
```

Outputs

- Policy $k'(k, z)$; consumption $c(k, z) = zf(k) - k'$.
- IRFs from a TFP shock via simulation of (k_t, z_t).

Acceleration Tips

- Howard improvement: fix policy, re-evaluate V several times.
- Use interpolation for $V(k', z')$ to avoid coarse k' grid.

Environment (Aiyagari)

- Incomplete markets, idiosyncratic income risk y (Markov).
- Household chooses savings a' with borrowing limit $a' \geq \underline{a}$.
- Prices (r, w) are competitive; aggregate K from stationary distribution.

Household Problem I

The problem of the individual is to maximize the discounted lifetime utility function:

$$E_0 \sum_{t=0}^{\infty} \beta^t U(c_t) \tag{17}$$

Here, c_t is the utility function, E_0 is the expectation operator which we already met, β is the discount factor while U is the utility function. The individual face the following constraint:

$$c_t + a_{t+1} = wl_t + (1+r)a \tag{18}$$

Household Problem II

We can now introduce two new variables, $\hat{a}_t$ and z_t and define them as:

$$\hat{a}_t = a_t + \phi \tag{19}$$

$$z_t = wl_t + (1+r)\hat{a}_t - r\phi \tag{20}$$

where z_t can be interpreted as the total resources that are available to an agent at time t. We can rewrite equation (6.21) as:

$$c_t + \hat{a}_{t+1} = z_t, \text{ if } c_t \geq 0, \hat{a} \geq 0 \tag{21}$$

$$z_{t+1} = wl_{t+1} + (1+r)\hat{a}_{t+1} - r\phi \tag{22}$$

The two equation above can be further written in a recursive manner using the approach already outlined in chapter 4. Let us define the optimal value function for the agent with total resources z_t as $V(z_t, b, w, r)$. This is the unique solution to the following dynamic programming problem:

$$V(z_t, b, w, r) = max\{U(z_t - \hat{a}_{t+1}) + \beta \int V(z_{t+1}, b, w, r) dF(l_{t+1})\} \tag{23}$$

Production I

The Aiyagari model also features a standard production sector. A representative firm produces a final good employing capital K and labor l. The production function is givem by:

$$Y_t = F(K_t, L_t) \tag{24}$$

The profit maximization leads to the typical first-order conditions:

$$\begin{aligned} F_K(K, L) &= r + \delta \\ F_L(K, L) &= w \end{aligned} \tag{25}$$

Equilibrium Conditions I

A stationary recursive equilibrium for the problem defined above consists in a value function $V : S \to \mathcal{R}$, policy functions for households $g : S \to \mathcal{R}$ and $c : S \to \mathcal{R}_+$, the choice of the representative firm in terms of capital stock K and labor L, the prices w and r and a stationary measure λ^* on S such that the following hold.

Equilibrium Conditions II

1. For a given set of prices (r, w), the policy functions (g, c) solve the household problem while V is the associated value function.
2. For a given set of prices (r, w), the representative firm chooses optimally the inputs K and L.
3. Factor prices r, w equal their marginal productivity as in equation (6.33);
4. The labor market is in equilibrium: $L = \int l d\lambda^*(a, l)$.
5. The asset market is in equilibrium: $K = \int g(a, l) d\lambda^*(a, l)$.
6. The economy's resource constraint is verified:

$$\int c(a, l) d\lambda^*(a, l) + \delta K = F(K, L) \tag{26}$$

7. For all Borel sets $A \times B$ on S, the distribution λ^* is stationary:

$$\lambda^*(A \times B) = \int 1_{g(a,l) \in A} Q(B, l) d\lambda^*(a, l)) \tag{27}$$

Algorithm (Fixed Prices)

1. Discretize assets grid $\{a_i\}$ and income states $\{y_j\}$ with Markov Π.
2. Solve VFI for policy $a'(a_i, y_j)$.
3. Compute stationary distribution (histogram simulation or transition matrix).
4. Guess r; set $w = F_w(K, L)$ from firm (or exogenous w).
5. Solve household and get implied $K(r)$.
6. Update r via bisection until $K(r)$ matches firm's K.

Julia Sketch: Grids and Prices

```
#  set parameter values
sigma  = 1.50;              # risk aversion
beta   = 0.98;              # subjective discount factor
prob   = [ .8 .2; .5 .5]; # prob(i,j) = probability (s(t+1)=sj | s(t) =
si)
delta  = 0.97;              # 1 - depreciation
A      = 1.00;              # production technology
alpha  = 0.25;              # capital's share of income
theta  = 0.05;              # non-rental income if unemployed is
theta*wage
Kstart = 10.0;              # initial value for aggregate capital stock
g      = 0.20;              # relaxation parameter
#   form capital grid
maxkap = 20;                        # maximum value of capital grid
inckap = 0.025;                     # size of capital grid increments
nkap   = trunc(Int,maxkap/inckap+1);  # number of grid points: make it
```

Julia Sketch: VFI at Fixed Prices

- loop to find fixed point for aggregate capital stock

```
liter   = 1;
maxiter = 50;
toler   = 0.001;
metric  = 10.0;
K = Kstart;
Kold= K;
wage=1.0;
rent=1.0;
println("ITERATING ON K");
println("");
println("Iter    metric                       meanK
Kold");
```

Stationary Distribution: Simulation

```
function markov(T,n,s0,V);
r,c   = size(T);
v1,v2 = size(V);
#rand('uniform');
#using Distributions
X=rand(Uniform(0,1), n-1,1)
state=zeros(2,99);
chain=[];
s=zeros(r,1);
s[s0]=1
cum=T*triu(ones(size(T)));
ppi  =[];
state[:,1]=s
for k=1:length(X);
#k=1
state[:,k]=s;
```

Outputs and Checks

- Simulate life history.
- Capital & income distribution.

Summary

- Dynamic programming: fixed-point and numerical building blocks.
- VFI: robust with constraints; acceleration via Howard and interpolation.
- Optimal growth: DP; then Aiyagari with GE loop.