DSE 2023 Summer School Lausanne

Lecture 1: Numerical Dynamic Programming

John Stachurski

2023

Topics

- Standard DP algorithms (VFI, HPI, OPI)
- State-dependent discounting

Code in

- Julia
- Python / Numba
- Python / NumPy
- Python / JAX

Based on discussion in Ch. 6 of https://dp.quantecon.org/

Example: Inventory Management

Consider an inventory model with Bellman equation

$$v(y) = \max_{a \in \Gamma(y)} \left\{ r(y, a) + \beta \sum_{y'} v(y') R(y, a, y') \right\}$$

- $y \in \{0, \ldots, K\}$ is current inventory
- a is current inventory order (units of stock)
- $r(y, a)$ is current profits
- $R(y, a, y') =$ prob. next state is y' given y, a

In the code, reward (profit) $r(y, a)$ is affected by a fixed cost per order

- hence investment is lumpy (S-s dynamics)

One limitation: **constant discounting**

For this model we would expect

$$\beta = \frac{1}{1 + \text{interest rate}}$$

And interest rates vary over time

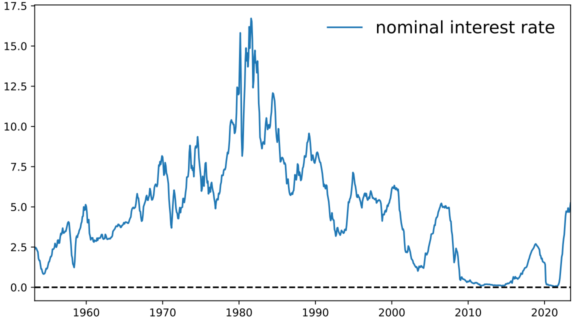


Figure: Nominal US interest rates

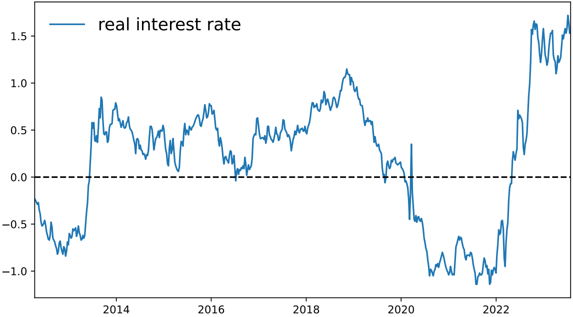


Figure: Real US interest rates

Let's shift to a setting with state-dependent discounting

Example. Discount time t reward r_t via

$$\text{current value } = \mathbb{E}\, \beta_1\, \beta_2 \, \cdots \, \beta_t\, r_t$$

Strategy:

1. study computation of lifetime values under state-dependent discounting
2. learn how to optimize these values across feasible policies

Let's start with step 1:

Valuation with non-constant discounting

Let $(X_t)_{t \geqslant 0}$ be a state process

- the discount factor varies with X_t
- profits / payoffs vary with X_t

To construct the state process, we

- let X be finite and let P be a Markov matrix on X:

$$P \geqslant 0 \quad \text{and} \quad \sum_{x' \in \mathsf{X}} P(x, x') = 1$$

- let $(X_t)_{t \geqslant 0}$ be P-Markov on X:

$$\mathbb{P}\{X_{t+1} = x' \mid X_t = x\} = P(x, x')$$

Consider the **discount factor process**

$$\delta_t := \prod_{i=1}^{t} b(X_{i-1}, X_i)$$

- Example. $b \equiv \beta \in (0,1)$ implies $\delta_t = \beta^t$ (const. discounting)
- Assume $b > 0$

Let A be the **discount operator**

$$A(x, x') := b(x, x') P(x, x')$$

Let

$$\rho(A) = \max_{\lambda \in \sigma(A)} |\lambda|, \qquad \sigma(A) := \text{ eigenvalues of } A$$

Let $\mathbb{R}^{\mathsf{X}}$ = all real-valued functions on X

Theorem. If $h \in \mathbb{R}^{\mathsf{X}}$ and $\rho(A) < 1$, then

$$v(x) := \mathbb{E}_x \sum_{t=0}^{\infty} \delta_t \, h(X_t) \text{ is finite for all } x \in \mathsf{X}$$

Moreover, $I - A$ is bijective and

$$v = \sum_{t \geqslant 0} A^t h$$

$$= (I - A)^{-1} h$$

Sketch of proof: An inductive argument shows that

$$\mathbb{E}_x \, \delta_t \, h(X_t) = (A^t h)(x)$$

Hence

$$v(x) = \mathbb{E}_x \sum_{t=0}^{\infty} \delta_t \, h(X_t) = \sum_{t=0}^{\infty} \mathbb{E}_x \, \delta_t \, h(X_t) = \sum_{t=0}^{\infty} (A^t h)(x)$$

That is, $v = \sum_{t \geqslant 0} A^t h$

By $\rho(A) < 1$ and the Neumann series lemma,

$$\sum_{t \geqslant 0} A^t = (I - A)^{-1}$$

Note that the condition $\rho(A) < 1$ is weak

- necessary as well as sufficient for convergence in many settings
- permits discount factor > 1 with positive probability

The last fact means

- can handle negative interest rates
- can handle models with large preference shocks
    - ZLB literature
    - asset pricing literature

Next steps

1. Obtain general results on DP with state-dependent discounting
2. Apply them to the inventory problem
3. Solve with Python / Julia using different algorithms

Our general results will use the theorem on slide 10

- a condition to ensure that valuations are finite

MDPs with State-Dependent Discounting

We consider a Markov decision process (MDP) with Bellman equation

$$v(x) = \max_{a \in \Gamma(x)} \left\{ r(x,a) + \sum_{x'} v(x')\beta(x,a,x')P(x,a,x') \right\}$$

for all $x \in \mathsf{X}$

- X, A finite state & action spaces
- Γ a nonempty correspondence from $\mathsf{X} \to \mathsf{A}$
- G := all (x, a) s.t. $a \in \Gamma(x)$
- β maps $\mathsf{G} \times \mathsf{X}$ to $(0, \infty)$

Let Σ be the set of **feasible policies**

$$\Sigma := \{\sigma \in \mathsf{A}^{\mathsf{X}} : \sigma(x) \in \Gamma(x) \text{ for all } x \in \mathsf{X}\}$$

Let

- $r_\sigma(x) := r(x, \sigma(x)) =$ rewards under σ
- $P_\sigma(x, x') := P(x, \sigma(x), x') =$ transitions under σ
- $\beta_\sigma(x, x') := \beta(x, \sigma(x), x') =$ discounting under σ

Note that P_σ is Markov dynamics for the state under σ

When it exists, the **lifetime value** v_σ of σ obeys

$$v_\sigma(x) = \mathbb{E}_x \sum_{t=0}^{\infty} \delta_t^\sigma \, r_\sigma(X_t)$$

where

$$\delta_t^\sigma := \prod_{i=1}^{t} \beta_\sigma(X_{i-1}, X_i)$$

and

$$(X_t)_{t \geqslant 0} \text{ is } P_\sigma\text{-Markov with } X_0 = x$$

Let

$$L_\sigma(x, x') := \beta_\sigma(x, x') P_\sigma(x, x')$$

Proposition. If $\rho(L_\sigma) < 1$, then

- $I - L_\sigma$ is bijective
- v_σ is finite and

$$v_\sigma = \sum_{t \geqslant 0} L_\sigma^t v = (I - L_\sigma)^{-1} r_\sigma$$

Proof: Follows directly from the result on slide 10

Let T_σ be the **policy operator** corresponding to σ:

$$T_\sigma \, v := r_\sigma + L_\sigma \, v$$

Note that

$$\begin{aligned} v_\sigma &= (I - L_\sigma)^{-1} r_\sigma \\ &\iff \quad v_\sigma \text{ solves eq. } (I - L)v = r_\sigma \\ &\iff \quad v_\sigma \text{ solves eq. } v = r_\sigma + L_\sigma v \\ &\iff \quad v_\sigma \in \operatorname{fix}(T_\sigma) \end{aligned}$$

By the results on slide 10,

$$\rho(L_\sigma) < 1 \implies \operatorname{fix}(T_\sigma) = \{v_\sigma\}$$

Fact. T_σ is **globally stable** on $\mathbb{R}^{\mathsf{X}}$ when $\rho(L_\sigma) < 1$:

$$\forall\, v \in \mathbb{R}^{\mathsf{X}}, \qquad T_\sigma^k v \to v_\sigma \quad \text{as} \quad k \to \infty \tag{1}$$

- See proof in `https://dp.quantecon.org/`

Hence we have three ways to compute v_σ:

1. $v_\sigma = \sum_{t \geqslant 0} L_\sigma^t r_\sigma$
2. $v_\sigma = (I - L_\sigma)^{-1} r_\sigma$
3. via (1)

We need a uniform bound, for all $\sigma \in \Sigma$:

Proposition. If $\exists$ an L such that

- $L_\sigma \leqslant L$ for all $\sigma \in \Sigma$
- $\rho(L) < 1$

then, for all $\sigma \in \Sigma$, the unique fixed point of T_σ in $\mathbb{R}^{\mathsf{X}}$ is

$$v_\sigma = (I - L_\sigma)^{-1} r_\sigma$$

Proof: Immediate from

1. $0 \leqslant A \leqslant B$ implies $\rho(A) \leqslant \rho(B)$
2. $0 \leqslant L_\sigma \leqslant L$ and $\rho(L) < 1$

What we have achieved

- found a condition for finite lifetime values across all policies
- found methods to compute these lifetime values

Yet to do

- show how to maximize lifetime values across $\sigma \in \Sigma$
- implement numerically

Defining optimality

Suppose the stated condition holds

- $\exists$ matrix L with $\rho(L) < 1$ and $L_\sigma \leqslant L$ for all σ

Then v_σ is finite for all σ

Hence we can define the **value function**

$$v^*(x) := \max_{\sigma \in \Sigma} v_\sigma(x) \qquad (x \in \mathsf{X})$$

A policy σ is called **optimal** if $v_\sigma = v^*$

The **Bellman operator** takes the form

$$(Tv)(x) = \max_{a \in \Gamma(x)} \left\{ r(x,a) + \sum_{x'} v(x') \beta(x,a,x') P(x,a,x') \right\}$$

Given $v \in \mathbb{R}^{\mathsf{X}}$, a policy σ is called v-**greedy** if

$$\sigma(x) \in \operatorname*{argmax}_{a \in \Gamma(x)} \left\{ r(x,a) + \sum_{x'} v(x') \beta(x,a,x') P(x,a,x') \right\}$$

for all x in X

Proposition. If there exists a linear operator L on $\mathbb{R}^{\mathsf{X}}$ such that

$$\rho(L) < 1 \quad \text{and} \quad \beta(x, a, x')P(x, a, x') \leqslant L(x, x')$$

for all $(x, a) \in \mathsf{G}$ and $x' \in \mathsf{X}$, then

1. T_σ is globally stable on $\mathbb{R}^{\mathsf{X}}$ with unique fixed point v_σ
2. $v_\sigma = (I - L_\sigma)^{-1} r_\sigma$
3. T is globally stable on $\mathbb{R}^{\mathsf{X}}$ with unique fixed point v^*
4. a policy $\sigma \in \Sigma$ is optimal if and only if it is v^*-greedy
5. at least one optimal policy exists

Algorithms

Now we have

- conditions for optimality
- general DP optimality results

Next we need algorithms

Let's consider

- value function iteration (VFI)
- Howard policy iteration (HPI)
- optimistic policy iteration (OPI)

Algorithm 1: VFI

input $v_0 \in \mathbb{R}^{\mathsf{X}}$

input τ, a tolerance level for error

$\varepsilon \leftarrow +\infty$

$k \leftarrow 0$

while $\varepsilon > \tau$ **do**

$v_{k+1} \leftarrow T v_k$

$\varepsilon \leftarrow \|v_k - v_{k+1}\|_\infty$

$k \leftarrow k + 1$

end

Compute a v_k-greedy policy σ

return σ

Algorithm 2: HPI

input $\sigma_0 \in \Sigma$, an initial guess of σ^*

$k \leftarrow 0$

$\varepsilon \leftarrow +\infty$

while $\varepsilon > 0$ **do**

  $v_k \leftarrow (I - L_{\sigma_k})^{-1} r_{\sigma_k}$

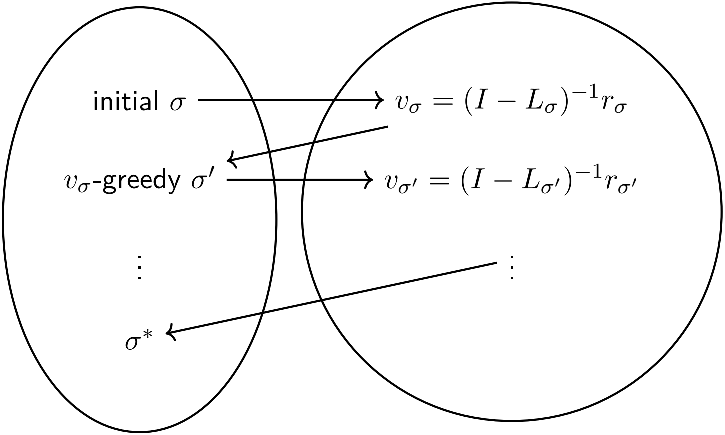
  $\sigma_{k+1} \leftarrow$ a v_k-greedy policy

  $\varepsilon \leftarrow \mathbb{1}\{\sigma_k \neq \sigma_{k+1}\}$

  $k \leftarrow k + 1$

end

return σ_k


initial σ
$v_\sigma = (I - L_\sigma)^{-1} r_\sigma$
v_σ-greedy σ'
$v_{\sigma'} = (I - L_{\sigma'})^{-1} r_{\sigma'}$
$\vdots$
$\vdots$
σ^*

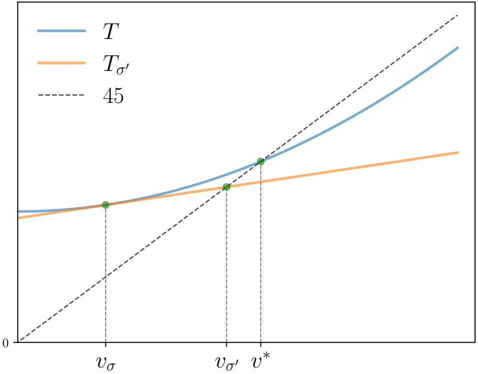


Figure: HPI as a version of Newton's method

Algorithm 3: OPI

input $v_0 \in \mathbb{R}^{\mathsf{X}}$, an initial guess of v^*

input τ, a tolerance level for error

input $m \in \mathbb{N}$, a step size

$k \leftarrow 0$

$\varepsilon \leftarrow +\infty$

while $\varepsilon > \tau$ **do**

  $\sigma_k \leftarrow$ a v_k-greedy policy

  $v_{k+1} \leftarrow T_{\sigma_k}^m v_k$

  $\varepsilon \leftarrow \|v_k - v_{k+1}\|_\infty$

  $k \leftarrow k + 1$

end

return σ_k

Proposition. Under the stated condition, VFI, HPI and OPI all converge

Moreover, HPI converges to an exact optimal policy in finitely many steps

For details and proofs see Ch. 6 of https://dp.quantecon.org/

Back to the inventory problem

Replace β with $\beta(Z_t)$ where $(Z_t)_{t \geqslant 0}$ is Q-Markov on Z

The Bellman equation becomes

$$v(y, z) = \max_{a \in \Gamma(y)} \left\{ r(y, a) + \beta(z) \sum_{z', \, y'} v(y', z') Q(z, z') R(y, a, y') \right\}$$

Given $\sigma \in \Sigma$, the **policy operator** is

$$(T_\sigma \, v)(y, z) = r(y, \sigma(y, z)) + \beta(z) \sum_{z', \, y'} v(y', z') Q(z, z') R(y, \sigma(y, z), y')$$

Proposition Let

$$L(z, z') := \beta(z) Q(z, z')$$

If $r(L) < 1$, then all of the optimality results on slide 24 apply

Ex. Check the details

- See Ch. 6 of https://dp.quantecon.org/ if you get stuck

Code that solves the model can be found at

- https://github.com/QuantEcon/dse_2023

Illustrates

- Python vs Julia
- Numba vs NumPy vs JAX
- Power of GPUs — if you have one