

Dynamic Programming

Chapter 5: Markov Decision Processes

Thomas J. Sargent and John Stachurski

2024

Topics

- Introduction to Markov decision processes
- Lifetime value
- Optimality
- Value function iteration
- Howard policy iteration
- Optimistic policy iteration
- Applications
- Refactoring dynamic programs

Markov Decision Processes (MDPs)

A class of well-behaved dynamic programs

- broad enough to encompass many economic applications
- includes optimal stopping problems as a special case
- admits a clean, powerful theory

A foundation stone for

- reinforcement learning
- artificial intelligence
- operations research, etc.

States and Actions

We take as given

1. a finite set X called the **state space** and
2. a finite set A called the **action space**

We study a controller who, at each integer $t \geq 0$

1. observes the current state $X_t \in X$
2. responds with an action $A_t \in A$

Her aim is to maximize

$$\mathbb{E} \sum_{t \geq 0} \beta^t r(X_t, A_t) \quad \text{given } X_0 = x_0$$

Actions restricted by a **feasible correspondence** Γ

- $\Gamma(x)$ is a nonempty subset of A for each $x \in X$
- interpretation: $\Gamma(x)$ = actions available in state x

Given Γ , we set

$$G = \{(x, a) \in X \times A : a \in \Gamma(x)\}$$

- called the set of **feasible state-action pairs**

Reward $r(x, a)$ is received at feasible state-action pair (x, a)

A **stochastic kernel** from G to X is a map $P: G \times X \rightarrow \mathbb{R}_+$ satisfying

$$\sum_{x' \in X} P(x, a, x') = 1 \quad \text{for all } (x, a) \text{ in } G$$

Interpretation

- next period state x' is drawn from distribution $P(x, a, \cdot)$

Now let's put it all together:

Given X and A , an **MDP** is a tuple $\mathcal{M} = (\Gamma, \beta, r, P)$ where

1. Γ is a nonempty correspondence from $X \rightarrow A$
2. β is a constant in $(0, 1)$
3. r is a function from G to \mathbb{R}
4. P is a stochastic kernel from G to X

In what follows,

- β is called the **discount factor**
- r is called the **reward function**

MDP dynamics:

```
 $t \leftarrow 0$   
input  $X_0$   
while  $t < \infty$  do  
    observe  $X_t$   
    choose action  $A_t$  from  $\Gamma(X_t)$   
    receive reward  $r(X_t, A_t)$   
    draw  $X_{t+1}$  from  $P(X_t, A_t, \cdot)$   
     $t \leftarrow t + 1$   
end
```

The **Bellman equation** is

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

Key idea: reduce infinite horizon problem to a two period problem

— “Bellman’s principle of optimality”

In the two period problem, the controller trades off

1. current rewards and
2. expected discounted value from future states

Example. Rust (1987 ECMA) examined an engine replacement problem for a bus workshop

At each t , the supervisor decides whether or not to replace a bus engine

Replacement is costly but delaying risks unexpected failure

We consider a version with binary action A_t

- $A_t = 1 \implies$ state resets to some fixed **renewal state** \bar{x}
 - e.g., mileage resets to zero when an engine is replaced
- $A_t = 0 \implies$ state updates according to $Q \in \mathcal{M}(\mathbb{R}^X)$
 - e.g., mileage increases stochastically when not replaced

Current reward is $r(x, a)$ at state x and action a

- $x \in X =$ some finite set (e.g., possible mileage values)

The discount factor is $\beta \in (0, 1)$

The Bellman equation is

$$v(x) = \max \left\{ r(x, 0) + \beta \sum_{x' \in X} v(x') Q(x, x'), r(x, 1) + \beta v(\bar{x}), \right\}$$
$$= \max \{ \text{value of action 0, value of action 1} \}$$

Ex. Show that the renewal problem can be framed as an MDP

Proof: We set

- $X =$ as above
- $A = \{0, 1\}$
- $\Gamma(x) = A$ for all $x \in X$
- $r =$ as above

We define

$$P(x, a, x') := a\mathbb{1}\{x' = \bar{x}\} + (1 - a)Q(x, x')$$

You can check that P is a stochastic kernel from G to X

Ex. Show that the renewal problem can be framed as an MDP

Proof: We set

- $X =$ as above
- $A = \{0, 1\}$
- $\Gamma(x) = A$ for all $x \in X$
- $r =$ as above

We define

$$P(x, a, x') := a\mathbb{1}\{x' = \bar{x}\} + (1 - a)Q(x, x')$$

You can check that P is a stochastic kernel from G to X

By construction, $\mathcal{M} = (\Gamma, \beta, r, P)$ is an MDP

Moreover,

$$\begin{aligned} v(x) &= \max_{a \in \Gamma(x)} \left\{ r(x, a) + \beta \sum_{x' \in \mathbf{X}} v(x') P(x, a, x') \right\} \\ &= \max_{a \in \{0,1\}} \left\{ r(x, a) + \beta \left[av(\bar{x}) + (1-a) \sum_{x' \in \mathbf{X}} v(x') Q(x, x') \right] \right\} \\ &= \max \left\{ r(x, 1) + \beta v(\bar{x}), r(x, 0) + \beta \sum_{x' \in \mathbf{X}} v(x') Q(x, x') \right\} \end{aligned}$$

We have recovered the renewal Bellman equation on slide 11

Example. Cake eating, where

$$W_{t+1} = R(W_t - C_t) \quad (t = 0, 1, \dots)$$

- investing d dollars today returns Rd next period
- $C_t, W_t \geq 0$ are current consumption and wealth
- (W_t) restricted to finite grid $W \subset \mathbb{R}_+$

The agent seeks to maximize $\mathbb{E} \sum_{t \geq 0} \beta^t u(C_t)$ given $W_0 = w$

Bellman equation:

$$v(w) = \max_{0 \leq w' \leq Rw} \left\{ u \left(w - \frac{w'}{R} \right) + \beta v(w') \right\}$$

This model can be framed as an MDP with W as the state space

- action is $A_t = W_{t+1} = \text{savings}$
- action space is also W
- feasible correspondence is

$$\Gamma(w) = \{a \in W : a \leqslant Rw\}$$

- since $A_t = R(W_t - C_t)$, current reward is

$$r(w, a) = u(w - a/R) \quad (a \in \Gamma(w))$$

- stochastic kernel is $P(w, a, w') = \mathbb{1}\{w' = a\}$

Example. Discrete time optimal stopping problems can be framed as MDPs

- See the text for details

This is important from a theoretical perspective

- illustrates the generality of MDPs

But expressing optimal stopping problems as an MDP requires an extra state variable

- status $S_t = \mathbb{1}\{\text{already stopped}\}$

This is why we treated them separately (Ch. 4)

Policies

Time travel is forbidden

- A_t cannot depend on X_{t+j} for any $j \geq 1$

While A_t can depend on the history

$$H_t := (A_0, X_0, \dots, A_{t-1}, X_{t-1}, X_t),$$

for MDPs, conditioning on X_t is sufficient

In other words,

- A_t depends only on X_t

Below we write $A_t = \sigma(X_t)$ and call σ a **policy function**

A **feasible policy** is a

$$\sigma \in A^X \text{ such that } \sigma(x) \in \Gamma(x) \text{ for all } x \in X$$

- Let $\Sigma :=$ the set of all feasible policies

Choosing $\sigma \in \Sigma \implies$

respond to state X_t with action $A_t := \sigma(X_t)$ at all $t \geq 0$

If $A_t := \sigma(X_t)$ at all $t \geq 0$, then

$$X_{t+1} \sim P(X_t, \sigma(X_t), \cdot) \quad \text{for all } t \geq 0$$

Thus, X_t is P_σ -Markov for

$$P_\sigma(x, x') := P(x, \sigma(x), x') \quad (x, x' \in \mathbf{X})$$

- Fixing a policy “closes the loop” in the state dynamics
- Solving an MDP means choosing a Markov chain!

Rewards

Under the policy σ , rewards at x given by $r(x, \sigma(x))$

Let

- $r_\sigma(x) := r(x, \sigma(x))$
- $\mathbb{E}_x := \mathbb{E}[\cdot \mid X_0 = x]$

Now

$$\mathbb{E}_x r(X_t, A_t) = \mathbb{E}_x r_\sigma(X_t) = \sum_{x'} r_\sigma(x') P_\sigma^t(x, x') = (P_\sigma^t r_\sigma)(x)$$

The **lifetime value of σ** starting from x is

$$\begin{aligned} v_\sigma(x) &:= \mathbb{E}_x \sum_{t \geq 0} \beta^t r_\sigma(X_t) \\ &= \sum_{t \geq 0} \mathbb{E}_x [\beta^t r_\sigma(X_t)] \\ &= \sum_{t \geq 0} \beta^t (P_\sigma^t r_\sigma)(x) \end{aligned}$$

Since $\beta < 1$, the spectral radius of βP is < 1 , so

$$v_\sigma = \sum_{t \geq 0} (\beta P_\sigma)^t r_\sigma = (I - \beta P_\sigma)^{-1} r_\sigma$$

Policy Operators

How should we compute v_σ given σ ?

We saw above that

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

- Computationally helpful when X is small
- Problematic for large problems

Example. If $|X| = 10^6$, then $I - \beta P_\sigma$ is $10^6 \times 10^6$

Matrices of this size are difficult invert—or even store in memory

Another way to compute v_σ : use the **policy operator**

$$(T_\sigma v)(x) = r(x, \sigma(x)) + \beta \sum_{x' \in \mathcal{X}} v(x') P(x, \sigma(x), x')$$

- Defined at all $v \in \mathbb{R}^{\mathcal{X}}$
- Analogous to T_σ for the optimal stopping problem

In vector notation, we can write

$$T_\sigma v = r_\sigma + \beta P_\sigma v$$

- T_σ is order-preserving on $\mathbb{R}^{\mathcal{X}}$ — why?

Ex. Show that T_σ is a contraction of modulus β on \mathbb{R}^X

For any v, w in \mathbb{R}^X we have

$$\begin{aligned}|T_\sigma v - T_\sigma w| &= \beta |P_\sigma v - P_\sigma w| \\&= \beta |P_\sigma(v - w)| \\&\leq \beta P_\sigma |v - w| \\&\leq \beta P_\sigma \|v - w\|_\infty \mathbb{1} \\&= \beta \|v - w\|_\infty \mathbb{1}\end{aligned}$$

Now use $|a| \leq |b|$ implies $\|a\|_\infty \leq \|b\|_\infty$

Ex. Show that T_σ is a contraction of modulus β on \mathbb{R}^X

For any v, w in \mathbb{R}^X we have

$$\begin{aligned} |T_\sigma v - T_\sigma w| &= \beta |P_\sigma v - P_\sigma w| \\ &= \beta |P_\sigma(v - w)| \\ &\leq \beta P_\sigma |v - w| \\ &\leq \beta P_\sigma \|v - w\|_\infty \mathbb{1} \\ &= \beta \|v - w\|_\infty \mathbb{1} \end{aligned}$$

Now use $|a| \leq |b|$ implies $\|a\|_\infty \leq \|b\|_\infty$

Ex. Show that v_σ is the unique fixed point of T_σ in \mathbb{R}^X

Proof: Since $\beta < 1$, we have

$$v = T_\sigma v \iff v = r_\sigma + \beta P_\sigma v$$

$$\iff v = (I - \beta P_\sigma)^{-1} r_\sigma$$

$$\iff v = v_\sigma$$

Hence

$$v \text{ is a fixed point of } T_\sigma \iff v = v_\sigma$$

Ex. Show that v_σ is the unique fixed point of T_σ in \mathbb{R}^X

Proof: Since $\beta < 1$, we have

$$v = T_\sigma v \iff v = r_\sigma + \beta P_\sigma v$$

$$\iff v = (I - \beta P_\sigma)^{-1} r_\sigma$$

$$\iff v = v_\sigma$$

Hence

$$v \text{ is a fixed point of } T_\sigma \iff v = v_\sigma$$

Greedy Policies

Fix $v \in \mathbb{R}^X$

A policy σ is called **v -greedy** if

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

for all $x \in X$

Ex. Prove: at least one v -greedy policy exists in Σ

Proof: Immediate because $\Gamma(x)$ is finite and nonempty at all x

Greedy Policies

Fix $v \in \mathbb{R}^X$

A policy σ is called **v -greedy** if

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

for all $x \in X$

Ex. Prove: at least one v -greedy policy exists in Σ

Proof: Immediate because $\Gamma(x)$ is finite and nonempty at all x

The Bellman Operator

Recall: the Bellman equation is

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

The **Bellman operator** for \mathcal{M} is the self-map on \mathbb{R}^X defined by

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

By construction, $Tv = v \iff v$ satisfies the Bellman equation

Ex. Prove: σ is v -greedy if and only if

$$T_{\sigma} v = Tv$$

Proof: σ is v -greedy if and only if

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

This is equivalent to

$$r(x, \sigma(x)) + \beta \sum_{x'} v(x') P(x, \sigma(x), x') = (Tv)(x) \quad \forall x \in X$$

Ex. Prove: σ is v -greedy if and only if

$$T_{\sigma} v = Tv$$

Proof: σ is v -greedy if and only if

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

This is equivalent to

$$r(x, \sigma(x)) + \beta \sum_{x'} v(x') P(x, \sigma(x), x') = (Tv)(x) \quad \forall x \in X$$

Ex. Prove that, for all $v \in \mathbb{R}^X$,

$$Tv = \bigvee_{\sigma \in \Sigma} T_{\sigma} v$$

Proof:

Fix $v \in \mathbb{R}^X$

For any v -greedy $\sigma \in \Sigma$, we have $T_{\sigma} v = Tv$

Also, for all $\sigma \in \Sigma$ and $x \in X$,

$$(T_{\sigma} v)(x) = r(x, \sigma(x)) + \beta \sum_{x'} v(x') P(x, \sigma(x), x') \leq (Tv)(x)$$

$$\therefore Tv = \bigvee_{\sigma} T_{\sigma} v$$

Ex. Prove that, for all $v \in \mathbb{R}^X$,

$$Tv = \bigvee_{\sigma \in \Sigma} T_{\sigma} v$$

Proof:

Fix $v \in \mathbb{R}^X$

For any v -greedy $\sigma \in \Sigma$, we have $T_{\sigma} v = Tv$

Also, for all $\sigma \in \Sigma$ and $x \in X$,

$$(T_{\sigma} v)(x) = r(x, \sigma(x)) + \beta \sum_{x'} v(x') P(x, \sigma(x), x') \leq (Tv)(x)$$

$$\therefore Tv = \bigvee_{\sigma} T_{\sigma} v$$

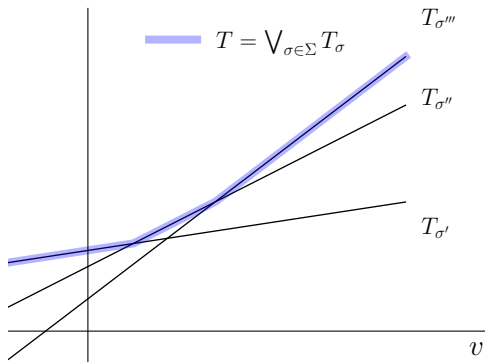


Figure: Visualization in one dimension

Ex. Prove: T is a contraction of modulus β on \mathbb{R}^X

Proof: Recall that, for finite S and any $f, g \in \mathbb{R}^S$,

$$\left| \max_{s \in S} f(s) - \max_{s \in S} g(s) \right| \leq \max_{s \in S} |f(s) - g(s)|$$

Hence, for any v, w in \mathbb{R}^X , we have

$$\begin{aligned} |(Tv)(x) - (Tw)(x)| &= \left| \max_{\sigma \in \Sigma} (T_\sigma v)(x) - \max_{\sigma \in \Sigma} (T_\sigma w)(x) \right| \\ &\leq \max_{\sigma \in \Sigma} |(T_\sigma v)(x) - (T_\sigma w)(x)| \end{aligned}$$

$$\therefore \|Tv - Tw\| \leq \max_{\sigma \in \Sigma} \|T_\sigma v - T_\sigma w\| \leq \beta \|v - w\|_\infty$$

Ex. Prove: T is a contraction of modulus β on \mathbb{R}^X

Proof: Recall that, for finite S and any $f, g \in \mathbb{R}^S$,

$$\left| \max_{s \in S} f(s) - \max_{s \in S} g(s) \right| \leq \max_{s \in S} |f(s) - g(s)|$$

Hence, for any v, w in \mathbb{R}^X , we have

$$\begin{aligned} |(Tv)(x) - (Tw)(x)| &= \left| \max_{\sigma \in \Sigma} (T_\sigma v)(x) - \max_{\sigma \in \Sigma} (T_\sigma w)(x) \right| \\ &\leq \max_{\sigma \in \Sigma} |(T_\sigma v)(x) - (T_\sigma w)(x)| \end{aligned}$$

$$\therefore \|Tv - Tw\| \leq \max_{\sigma \in \Sigma} \|T_\sigma v - T_\sigma w\| \leq \beta \|v - w\|_\infty$$

Optimality

The **value function** is defined by $v^* := \bigvee_{\sigma \in \Sigma} v_\sigma$

More explicitly,

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

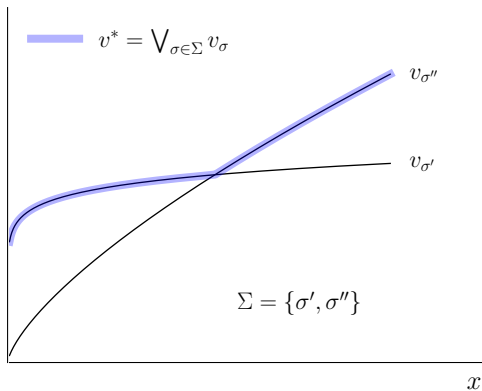
Thus, $v^*(x) = \underline{\text{maximal lifetime value}}$ from state x

A policy $\sigma \in \Sigma$ is called **optimal** if

$$v_\sigma = v^*$$

Thus, σ is optimal \iff lifetime value is maximal at each state

Example. Consider



In this case there is no optimal policy

Indeed, v^* differs from both $v_{\sigma'}$ and $v_{\sigma''}$

Hence no σ satisfies $v^* = v_{\sigma}$

Below we show that such an outcome is **not** possible for MDPs

In other words, an optimal policy always exists

This leads to our next slide. . .

Theorem. If \mathcal{M} is an MDP with Bellman operator T and value function v^* , then

1. v^* is the unique fixed point of T in \mathbb{R}^X
2. A feasible policy is optimal if and only if it is v^* -greedy
3. At least one optimal policy exists

Remark: Point (2) is called **Bellman's principle of optimality**

We prove the theorem below through some exercises

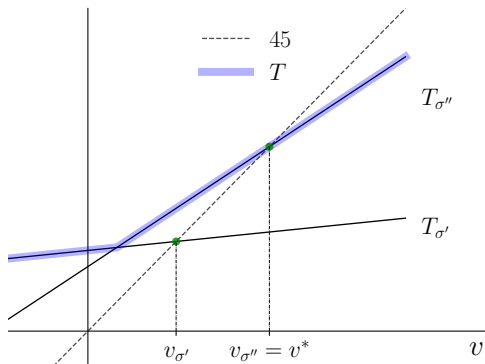


Figure: Illustration of optimality for MDPs

Ex. Show that Bellman's principle of optimality implies existence of an optimal policy

Proof:

Let Bellman's principle of optimality hold

For each $v \in \mathbb{R}^X$, a v -greedy policy exists

Hence a v^* -greedy policy exists

Hence an optimal policy exists

Ex. Show that Bellman's principle of optimality implies existence of an optimal policy

Proof:

Let Bellman's principle of optimality hold

For each $v \in \mathbb{R}^X$, a v -greedy policy exists

Hence a v^* -greedy policy exists

Hence an optimal policy exists

We know that T is a contraction mapping on \mathbb{R}^X

Hence T is globally stable on \mathbb{R}^X with unique fixed point $\bar{v} \in \mathbb{R}^X$

To prove (1) of the above theorem, we have to show $\bar{v} = v^*$

Ex. Show $\bar{v} \leq v^*$

Proof: Let $\sigma \in \Sigma$ be \bar{v} -greedy

Then $T_\sigma \bar{v} = T\bar{v} = \bar{v}$

Hence \bar{v} is a fixed point of T_σ

But the only fixed point of T_σ in \mathbb{R}^X is v_σ

We have shown existence of a σ such that $\bar{v} = v_\sigma$

But then $\bar{v} \leq v^*$ (why?)

Ex. Show $v^* \leq \bar{v}$

Proof: Fix $\sigma \in \Sigma$ and note that

$$T_\sigma v \leq Tv \quad \text{for all } v \in \mathbb{R}^X$$

Moreover, T is order-preserving and globally stable

As we saw in Ch. 2, this implies that the fixed points are likewise ordered

In particular, $v_\sigma \leq \bar{v}$

Hence \bar{v} is an upper bound of $\{v_\sigma\}_{\sigma \in \Sigma}$

Therefore $v^* \leq \bar{v}$

Ex. Show $v^* \leq \bar{v}$

Proof: Fix $\sigma \in \Sigma$ and note that

$$T_\sigma v \leq Tv \quad \text{for all } v \in \mathbb{R}^X$$

Moreover, T is order-preserving and globally stable

As we saw in Ch. 2, this implies that the fixed points are likewise ordered

In particular, $v_\sigma \leq \bar{v}$

Hence \bar{v} is an upper bound of $\{v_\sigma\}_{\sigma \in \Sigma}$

Therefore $v^* \leq \bar{v}$

We have now shown that $v^* = \bar{v}$

Thus, v^* is a fixed point of T in \mathbb{R}^X

But T is globally stable on \mathbb{R}^X

Hence v^* is the only fixed point of T in \mathbb{R}^X

In other words,

$v^* =$ the unique solution to the Bellman equation in \mathbb{R}^X

To prove Bellman's principle of optimality, we use $Tv^* = v^*$

We have

$$\sigma \text{ is } v^*\text{-greedy} \iff T_\sigma v^* = Tv^* \iff T_\sigma v^* = v^*$$

The last statement is equivalent to $v_\sigma = v^*$ (why?)

Hence

$$\sigma \text{ is } v^*\text{-greedy} \iff v^* = v_\sigma \iff \sigma \text{ is optimal}$$

In other words, Bellman's principle of optimality holds

Algorithms

Previously we used value function iteration (VFI) to solve optimal stopping problems

Here we

1. present a generalization suitable for arbitrary MDPs
2. introduce two other important methods

The two other methods are called

1. Howard policy iteration (HPI) and
2. Optimistic policy iteration (OPI)

Algorithm 1: VFI for MDPs

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

input τ , a tolerance level for error

$\varepsilon \leftarrow \tau + 1$

$k \leftarrow 0$

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

for $x \in X$ **do**

$v_{k+1}(x) \leftarrow (Tv_k)(x)$

end

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

$k \leftarrow k + 1$

end

Compute a v_k -greedy policy σ

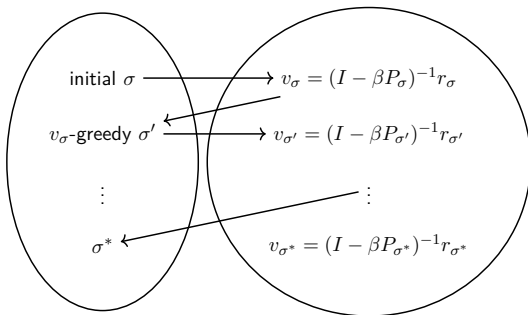
return σ

VFI is

- easy to understand
- easy to implement
- globally convergent
- relatively robust

However, we can often find faster methods with a bit of effort

Howard Policy Iteration



Iterates between computing the value of a given policy and computing the greedy policy associated with that value

Algorithm 2: Howard policy iteration for MDPs

input $\sigma \in \Sigma$

$v_0 \leftarrow v_\sigma$ and $k \leftarrow 0$

repeat

$\sigma_k \leftarrow$ a v_k -greedy policy

$v_{k+1} \leftarrow (I - \beta P_{\sigma_k})^{-1} r_{\sigma_k}$

if $v_{k+1} = v_k$ **then break**

$k \leftarrow k + 1$

return σ_k

Proposition. HPI returns an exact optimal policy in a finite number of steps

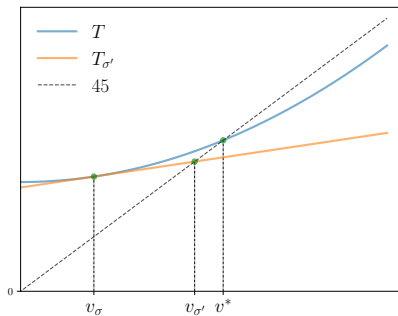
Also, rate of convergence is faster than VFI

In fact HPI is analogous to gradient-based Newton iteration on T

- Details are in the text

In general, for a given fixed point problem,

1. Newton iteration yields a quadratic rate of convergence
2. Successive approximation yields a linear rate of convergence



- σ' is v_{σ} -greedy if $T_{\sigma'}v_{\sigma} = Tv_{\sigma}$
- $v_{\sigma'}$ is the fixed point of $T_{\sigma'}$

Optimistic Policy Iteration

OPI is a “convex combination” of VFI and HPI

Similar to HPI except that

- HPI takes current σ and obtains v_σ
- OPI takes current σ and iterates m times with T_σ

Recall that, for any $v \in \mathbb{R}^X$, we have $T_\sigma^m v \rightarrow v_\sigma$ as $m \rightarrow \infty$

Hence OPI replaces v_σ with an approximation

Algorithm 3: Optimistic policy iteration for MDPs

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

input τ , a tolerance level for error

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

$k \leftarrow 0$

$\varepsilon \leftarrow \tau + 1$

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. For all values of m we have $v_k \rightarrow v^*$

Ex. Show that OPI = VFI when $m = 1$

Proof. Suppose we are at step k with $v_k \in \mathbb{R}^X$

Now we take σ_k to be v_k -greedy and set $v_{k+1} = T_{\sigma_k} v_k$

Since σ_k is v_k -greedy, we also have $T_{\sigma_k} v_k = T v_k$

Hence v_{k+1} is also the next step for VFI

Proposition. For all values of m we have $v_k \rightarrow v^*$

Ex. Show that OPI = VFI when $m = 1$

Proof. Suppose we are at step k with $v_k \in \mathbb{R}^X$

Now we take σ_k to be v_k -greedy and set $v_{k+1} = T_{\sigma_k} v_k$

Since σ_k is v_k -greedy, we also have $T_{\sigma_k} v_k = T v_k$

Hence v_{k+1} is also the next step for VFI

Intuitively, if $m = \infty$, OPI is identical to HPI

This is because $\lim_{m \rightarrow \infty} T_{\sigma_k}^m v_k = v_{\sigma_k}$

Without parallelization, an intermediate value of m is usually better than both

More rules of thumb:

- parallelization tends to favor HPI
- VFI works well when β is small and optimization is cheap

Application: Optimal Inventories

Previously we analyzed S-s inventory dynamics

- our aim was to understand Markov chains

But are such dynamics realistic?

We now investigate whether S-s behavior arises naturally in optimizing model

- firm chooses its inventory path to maximize firm value

We assume for now that the firm only sells one product

Given a demand process $(D_t)_{t \geq 0}$, inventory $(X_t)_{t \geq 0}$ obeys

$$X_{t+1} = f(X_t, A_t, D_{t+1})$$

where

- $f(x, a, d) = (x - d) \vee 0 + a$
- A_t is units of stock ordered this period
- The firm can store at most K items at one time

The state space is $X := \{0, \dots, K\}$

We assume $(D_t) \stackrel{\text{iid}}{\sim} \varphi \in \mathcal{D}(\mathbb{Z}_+)$

Profits are given by

$$\pi_t := X_t \wedge D_{t+1} - cA_t - \kappa \mathbb{1}\{A_t > 0\}$$

- output price $\equiv 1$
- orders in excess of inventory are lost
- c is unit product cost (and unit sales prices = 1)
- κ is a fixed cost of ordering inventory

With $\beta := 1/(1+r)$ and $r > 0$, the value of the firm is

$$V_0 = \mathbb{E} \sum_{t \geq 0} \beta^t \pi_t$$

Expected current profit is

$$r(x, a) := \sum_{d \geq 0} (x \wedge d) \varphi(d) - ca - \kappa \mathbb{1}\{a > 0\}$$

The set of feasible order quantities at x is

$$\Gamma(x) := \{0, \dots, K - x\}$$

The Bellman equation is

$$v(x) = \max_{a \in \Gamma(x)} \left\{ r(x, a) + \beta \sum_{d \geq 0} v(f(x, a, d)) \varphi(d) \right\}$$

The model is an MDP $\mathcal{M} = (\Gamma, \beta, r, P)$ with

- state space X and action space $A := X$
- Γ , r and β as above
- stochastic kernel

$$\begin{aligned} P(x, a, x') &:= \mathbb{P}\{f(x, a, D) = x'\} \\ &= \sum_{d \geq 0} \mathbb{1}\{f(x, a, d) = x'\} \varphi(d) \end{aligned}$$

Since \mathcal{M} is an MDP, all optimality results on slide 35 apply

using Distributions

$f(x, a, d) = \max(x - d, 0) + a$ *# Inventory update*

```
function create_inventory_model(;  $\beta=0.98$ ,      # discount factor  
                                 $K=40$ ,          # maximum inventory  
                                 $c=0.2$ ,  $\kappa=2$ , # cost paramters  
                                 $p=0.6$ )      # demand parameter  
     $\phi(d) = (1 - p)^d * p$       # demand distribution  
     $x\_vals = \text{collect}(0:K)$     # set of inventory levels  
    return (;  $\beta$ ,  $K$ ,  $c$ ,  $\kappa$ ,  $p$ ,  $\phi$ ,  $x\_vals$ )  
end
```

"The function $B(x, a, v) = r(x, a) + \beta \sum_{x'} v(x') P(x, a, x')$."

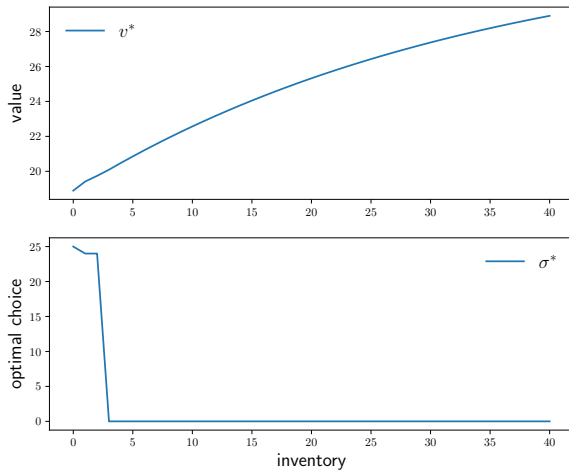
```
function B(x, a, v, model;  $d\_max=100$ )  
    (;  $\beta$ ,  $K$ ,  $c$ ,  $\kappa$ ,  $p$ ,  $\phi$ ,  $x\_vals$ ) = model  
    revenue = sum(min(x, d) *  $\phi(d)$  for d in 0: $d\_max$ )  
    current_profit = revenue -  $c * a - \kappa * (a > 0)$   
    next_value = sum(v[f(x, a, d) + 1] *  $\phi(d)$  for d in 0: $d\_max$ )  
    return current_profit +  $\beta * next\_value$   
end
```

"The Bellman operator."

```
function T(v, model)
    (;  $\beta$ , K, c,  $\kappa$ , p,  $\phi$ , x_vals) = model
    new_v = similar(v)
    for (x_idx, x) in enumerate(x_vals)
         $\Gamma x = 0:(K - x)$ 
        new_v[x_idx], _ = findmax(B(x, a, v, model) for a in  $\Gamma x$ )
    end
    return new_v
end
```

"Get a v-greedy policy."

```
function get_greedy(v, model)
    (;  $\beta$ , K, c,  $\kappa$ , p,  $\phi$ , x_vals) = model
     $\sigma_{\text{star}} = \text{zero}(x\_vals)$ 
    for (x_idx, x) in enumerate(x_vals)
         $\Gamma x = 0:(K - x)$ 
        _, a_idx = findmax(B(x, a, v, model) for a in  $\Gamma x$ )
         $\sigma_{\text{star}}[x\_idx] = \Gamma x[a\_idx]$ 
    end
    return  $\sigma_{\text{star}}$ 
end
```



Ex. Try to replicate these plots

- Use the code given above (or at least the same parameters)
- Use value function iteration

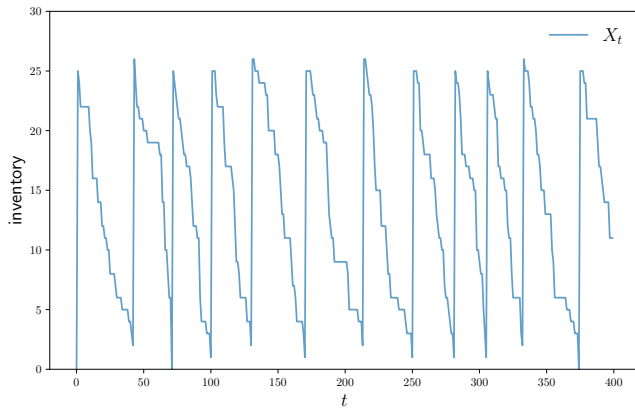


Figure: Optimal inventory dynamics

Optimal Savings with Labor Income

Wealth evolves according to

$$W_{t+1} = R(W_t + Y_t - C_t) \quad (t = 0, 1, \dots)$$

- (W_t) takes values in finite set $W \subset \mathbb{R}_+$
- (Y_t) is Q -Markov chain on finite set Y
- $C_t, W_t \geq 0$

The household maximizes

$$\mathbb{E} \sum_{t \geq 0} \beta^t u(C_t)$$

The model is an MDP with state space $X := W \times Y$

- action = gross savings = next period wealth
- The feasible correspondence is

$$\Gamma(w, y) = \{s \in W : s \leq R(w + y)\}$$

- current reward is

$$r(w, y, s) = u(w + y - s/R) \quad (\text{utility of consumption})$$

- stochastic kernel is

$$P((w, y), s, (w', y')) = \mathbb{1}\{w' = s\}Q(y, y')$$

Bellman operator:

$$(Tv)(w, y) =$$

$$\max_{w' \in \Gamma(w, y)} \left\{ u(w + y - w'/R) + \beta \sum_{y' \in Y} v(w', y') Q(y, y') \right\}$$

The policy operator for given $\sigma \in \Sigma$ is

$$(T_\sigma v)(w, y) =$$

$$u(w + y - \sigma(w, y)/R) + \beta \sum_{y' \in Y} v(\sigma(w, y), y') Q(y, y')$$

How to solve $v_\sigma = (I - \beta P_\sigma)^{-1} r_\sigma$?

Set

$$P_\sigma((w, y), (w', y')) := \mathbb{1}\{\sigma(w, y) = w'\} Q(y, y')$$

and

$$r_\sigma(w, y) := u(w + y - \sigma(w, y)/R)$$

Either

- reshape for matrix algebra (compress (w, y) to single index)
- use linear operator routines

How to solve $v_\sigma = (I - \beta P_\sigma)^{-1} r_\sigma$?

Set

$$P_\sigma((w, y), (w', y')) := \mathbb{1}\{\sigma(w, y) = w'\} Q(y, y')$$

and

$$r_\sigma(w, y) := u(w + y - \sigma(w, y)/R)$$

Either

- reshape for matrix algebra (compress (w, y) to single index)
- use linear operator routines

```
using QuantEcon, LinearAlgebra, IterTools
```

```
function create_savings_model(; R=1.01,  $\beta$ =0.98,  $\gamma$ =2.5,  
                             w_min=0.01, w_max=20.0, w_size=200,  
                              $\rho$ =0.9, v=0.1, y_size=5)  
    w_grid = LinRange(w_min, w_max, w_size)  
    mc = tauchen(y_size,  $\rho$ , v)  
    y_grid, Q = exp.(mc.state_values), mc.p  
    return (;  $\beta$ , R,  $\gamma$ , w_grid, y_grid, Q)  
end
```

```
"B(w, y, w', v) = u(R*w + y - w') +  $\beta \sum y' v(w', y') Q(y, y')."$ 
```

```
function B(i, j, k, v, model)  
    (;  $\beta$ , R,  $\gamma$ , w_grid, y_grid, Q) = model  
    w, y, w' = w_grid[i], y_grid[j], w_grid[k]  
    u(c) = c^(1- $\gamma$ ) / (1- $\gamma$ )  
    c = w + y - (w' / R)  
    @views value = c > 0 ? u(c) +  $\beta$  * dot(v[k, :], Q[j, :]) : -Inf  
    return value  
end
```

"The Bellman operator."

```
function T(v, model)
    w_idx, y_idx = (eachindex(g) for g in (model.w_grid, model.y_grid))
    v_new = similar(v)
    for (i, j) in product(w_idx, y_idx)
        v_new[i, j] = maximum(B(i, j, k, v, model) for k in w_idx)
    end
    return v_new
end
```

"The policy operator."

```
function T_σ(v, σ, model)
    w_idx, y_idx = (eachindex(g) for g in (model.w_grid, model.y_grid))
    v_new = similar(v)
    for (i, j) in product(w_idx, y_idx)
        v_new[i, j] = B(i, j, σ[i, j], v, model)
    end
    return v_new
end
```

```
include("finite_opt_saving_0.jl")

"Compute a v-greedy policy."
function get_greedy(v, model)
    w_idx, y_idx = (eachindex(g) for g in (model.w_grid, model.y_grid))
    σ = Matrix{Int32}(undef, length(w_idx), length(y_idx))
    for (i, j) in product(w_idx, y_idx)
        _, σ[i, j] = findmax(B(i, j, k, v, model) for k in w_idx)
    end
    return σ
end
```

```

"Get the value  $v_\sigma$  of policy  $\sigma$ ."
function get_value( $\sigma$ , model)
    # Unpack and set up
    (;  $\beta$ , R,  $\gamma$ , w_grid, y_grid, Q) = model
    w_idx, y_idx = (eachindex(g) for g in (w_grid, y_grid))
    wn, yn = length(w_idx), length(y_idx)
    n = wn * yn
    u(c) =  $c^{(1-\gamma)} / (1-\gamma)$ 
    # Build  $P_\sigma$  and  $r_\sigma$  as multi-index arrays
    P_σ = zeros(wn, yn, wn, yn)
    r_σ = zeros(wn, yn)
    for (i, j) in product(w_idx, y_idx)
        w, y, w' = w_grid[i], y_grid[j], w_grid[σ[i, j]]
        r_σ[i, j] = u(w + y - w'/R)
        for j' in y_idx
            P_σ[i, j, σ[i, j], j'] = Q[j, j']
        end
    end
    # Reshape for matrix algebra
    P_σ = reshape(P_σ, n, n)
    r_σ = reshape(r_σ, n)
    # Apply matrix operations --- solve for the value of  $\sigma$ 
    v_σ = (I -  $\beta$  * P_σ) \ r_σ
    # Return as multi-index array
    return reshape(v_σ, wn, yn)
end

```

"Value function iteration routine."

"Howard policy iteration routine."

"Optimistic policy iteration routine."

```
function optimistic_policy_iteration(model; tolerance=1e-5, m=100)
    v = zeros(length(model.w_grid), length(model.y_grid))
    error = tolerance + 1
    while error > tolerance
        last_v = v
         $\sigma$  = get_greedy(v, model)
        for i in 1:m
            v = T_ $\sigma$ (v,  $\sigma$ , model)
        end
        error = maximum(abs.(v - last_v))
    end
    return get_greedy(v, model)
end
```

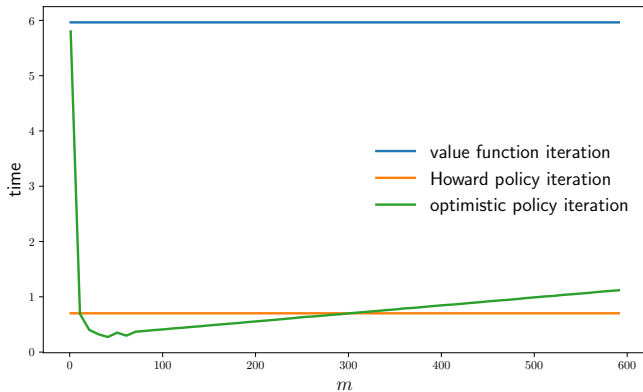


Figure: Timings for alternative algorithms

Investment with Adjustment Costs

A monopolist faces an inverse demand function of the form

$$P_t = a_0 - a_1 Y_t + Z_t$$

- a_0, a_1 are positive parameters
- Y_t is output
- P_t is price and
- the demand shock Z_t follows

$$Z_{t+1} = \rho Z_t + \sigma \eta_{t+1}, \quad \{\eta_t\} \stackrel{\text{iid}}{\sim} N(0, 1)$$

Current profits are given by

$$\pi_t := P_t Y_t - c Y_t - \gamma (Y_{t+1} - Y_t)^2$$

- $\gamma(Y_{t+1} - Y_t)^2$ represents adjustment costs
- rapid changes to capacity are expensive when γ large

Objective: maximize value

$$\mathbb{E} \sum_{t=0}^{\infty} \beta^t \pi_t$$

- $\beta = 1/(1+r)$, where $r > 0$ is a fixed interest rate

Building intuition: If $\gamma = 0$ then no intertemporal trade-off

Therefore maximize current profit each period:

$$\max_{Y_t} \{P_t Y_t - c Y_t\} = \max_{Y_t} \{(a_0 - a_1 Y_t + Z_t) Y_t - c Y_t\}$$

Ex. Show that the maximizer is

$$\bar{Y}_t := \frac{a_0 - c + Z_t}{2a_1}$$

On the other hand, if γ is very large then $(Y_t)_{t \geq 0}$ should be almost constant

Thus, we expect the following:

- $\gamma \approx 0 \implies Y_t$ tracks \bar{Y}_t closely
- γ large $\implies Y_t$ will be smoother than \bar{Y}_t

In short,

more adjustment costs \implies smoother time path for $(Y_t)_{t \geq 0}$

Implementation as an MDP

Let $Y \subset \mathbb{R}_+$ be a grid containing output values

We discretize (Z_t) using Tauchen's method:

(Z_t) is Q -Markov on finite set $Z \subset \mathbb{R}$

The state space $X := Y \times Z$

The action space is Y

The feasible correspondence is $\Gamma(x) = Y$ for all x

- choice of output is not restricted by the state

The set Σ is all $\sigma: Y \times Z \rightarrow Y$

The current reward function is current profits:

$$r(y, z, \hat{y}) = (a_0 - a_1 y + z - c)y - \gamma(\hat{y} - y)^2$$

- \hat{y} = action = choice of next-period output

The stochastic kernel is

$$P((y, z), \hat{y}, (y', z')) = \mathbb{1}\{y' = \hat{y}\}Q(z, z')$$

Now the problem defines an MDP

- all of the optimality theory for MDPs applies

Optimal Investment

The Bellman and policy operators for this problem are

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

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

On \mathbb{R}^X , both are

- order-preserving
- contractions of modulus β

A v -greedy policy is a $\sigma \in \Sigma$ that obeys

$$\sigma(y, z) = \operatorname{argmax}_{y' \in Y} \left\{ r(y, z, y') + \beta \sum_{z' \in Z} v(y', z') Q(z, z') \right\}$$

By our results for MDPs

- v^* -greedy policies = optimal policies
- OPI / HPI / VFI converge

Implications for output can be studied by

1. generating a Q -Markov chain $(Z_t)_{t=1}^T$
2. simulating optimal output via $Y_{t+1} = \sigma^*(Y_t, Z_t)$

```
using QuantEcon, LinearAlgebra, IterTools
include("s_approx.jl")
```

```
function create_investment_model(;
    r=0.04,                    # Interest rate
    a_0=10.0, a_1=1.0,        # Demand parameters
    γ=25.0, c=1.0,            # Adjustment and unit cost
    y_min=0.0, y_max=20.0, y_size=100, # Grid for output
    ρ=0.9, v=1.0,              # AR(1) parameters
    z_size=25)                 # Grid size for shock
    β = 1/(1+r)
    y_grid = LinRange(y_min, y_max, y_size)
    mc = tauchen(z_size, ρ, v)
    z_grid, Q = mc.state_values, mc.p
    return (; β, a_0, a_1, γ, c, y_grid, z_grid, Q)
end
```

The aggregator B is given by

$$B(y, z, y') = r(y, z, y') + \beta \sum_{z'} v(y', z') Q(z, z').$$

where

$$r(y, z, y') := (a_0 - a_1 * y + z - c) y - y * (y' - y)^2$$

```
function B(i, j, k, v, model)
    (;  $\beta$ , a_0, a_1,  $\gamma$ , c, y_grid, z_grid, Q) = model
    y, z, y' = y_grid[i], z_grid[j], y_grid[k]
    r = (a_0 - a_1 * y + z - c) * y -  $\gamma$  * (y' - y)^2
    return @views r +  $\beta$  * dot(v[k, :], Q[j, :])
end
```