ROB-GY 6323
reinforcement learning and optimal control for robotics

Lecture 8
Value and policy iteration

Course material
All necessary material will be posted on Brightspace
Code will be posted on the Github site of the class

https://github.com/righetti/optlearningcontrol

Discussions/Forum with Slack

Contact
ludovic.righetti@nyu.edu
Office hours in person
Wednesday 3pm to 4pm
370 Jay street - room 801

Course Assistant

Armand Jordana
aj2988@nyu.edu
Office hours Monday 1pm to 2pm
Rogers Hall 515

any other time by appointment only

Tentative schedule (subject to change)

Week	Lecture		Homework	Project
1	Intro	Lecture 1: introduction		
2	Trajectory optimization	Lecture 2: Basics of optimization	HW 1	
3		Lecture 3: QPs		
4		Lecture 4: Nonlinear optimal control		
5		Lecture 5: Model-predictive control		
6		Lecture 6: Sampling-based optimal control	HW 2	
7	Policy optimization	Lecture 7: Bellman's principle		
8		Lecture 8: Value iteration / policy iteration	HW 3	Project 1
9		Lecture 9: TD learning - Q-learning		
10		Lecture 10: Deep Q learning	HW 4	
11		Lecture 11: Actor-critic algorithms		
12		Lecture 12: Learning by demonstration	HW 5	Project 2
13		Lecture 13: Monte-Carlo Tree Search		
14		Lecture 14: Beyond the class		
15	Finals week			

Homework 2 is due this week!

Please be concise in your answers
(2-3 sentence per written question should be sufficient)

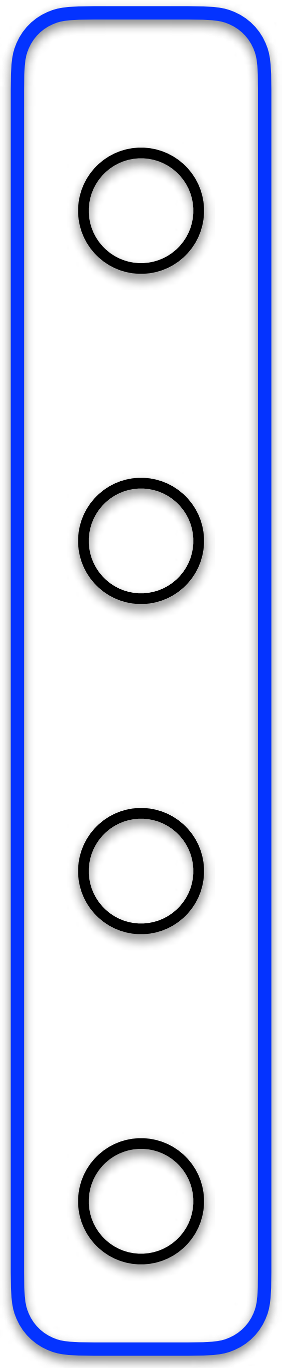
Bellman’s principle of optimality

Dynamic programming

For every initial state x_0, the optimal cost $J^*(x_0)$ of the optimal control problem is equal to $J_0(x_0)$ (i.e. the optimal cost to go from x_0) which is computed backward in time from stage $N-1$ to stage 0 using the following recursion:

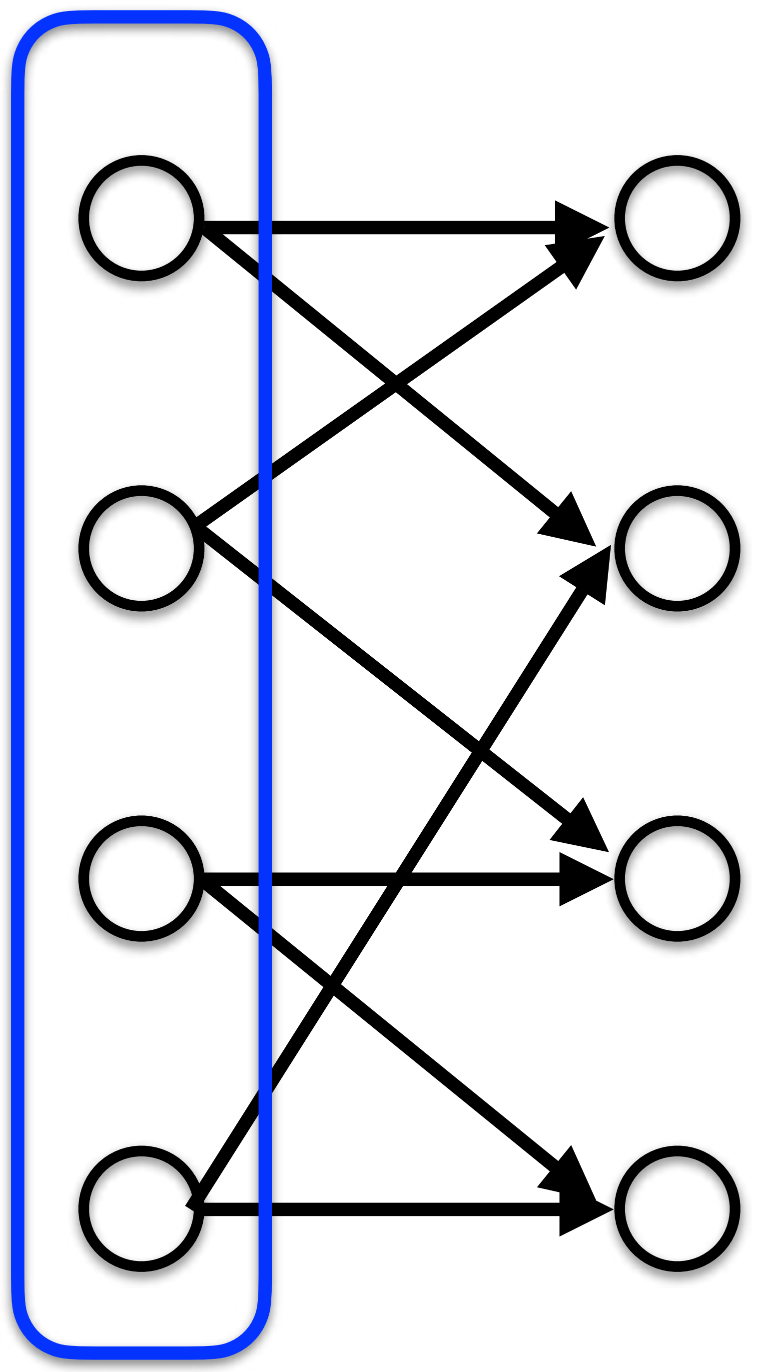
$$
\begin{aligned}
J_N(x_N) &= g_N(x_N) \\
J_k(x_k) &= \min_{u_k} g_k(x_k, u_k) + J_{k+1}(f(x_k, u_k))
\end{aligned}
$$

Furthermore, if the control laws $u_k^* = \mu_k^*(x_k)$ minimize the cost-to-go for each x_k and k, the policy $\pi^* = \{\mu_0^*, \cdots, \mu_{N-1}^*\}$ is optimal.

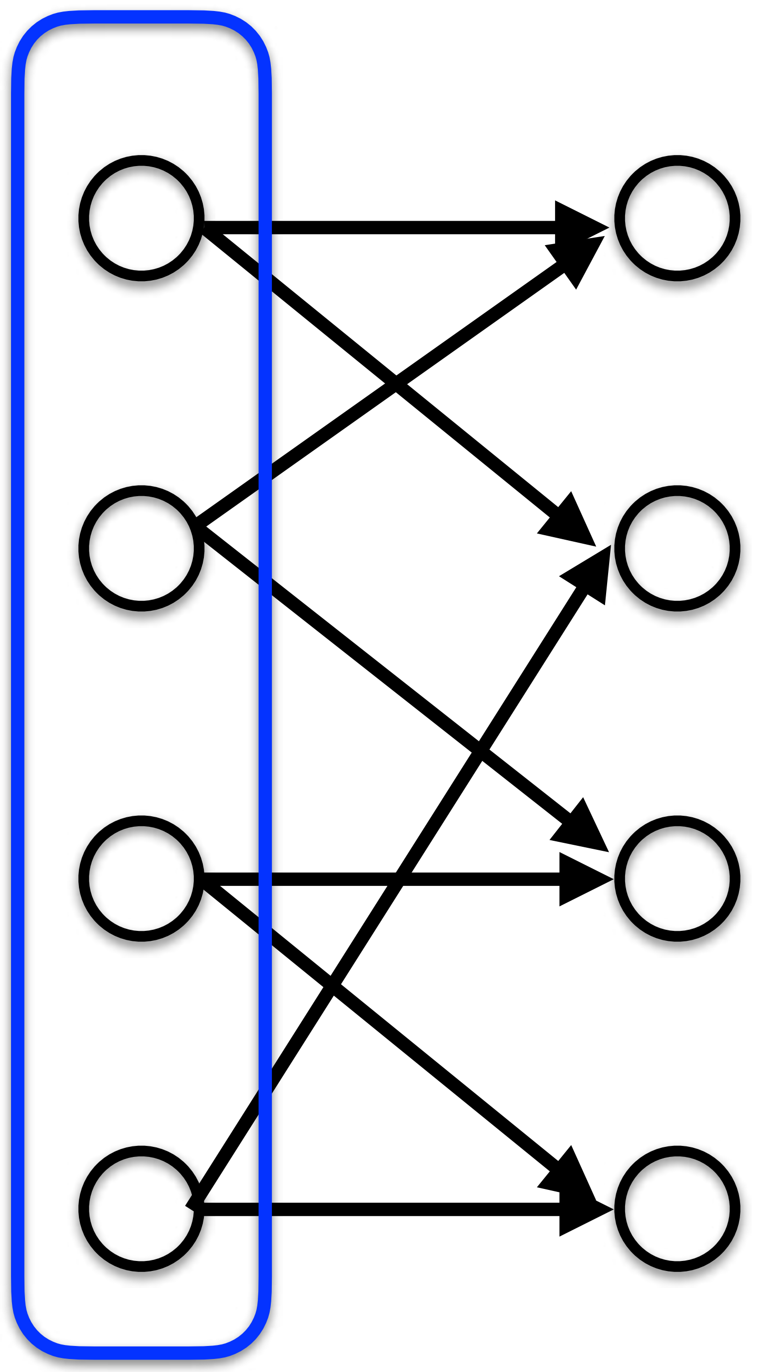


Final States
Stage N

$$J_N(X_N)$$

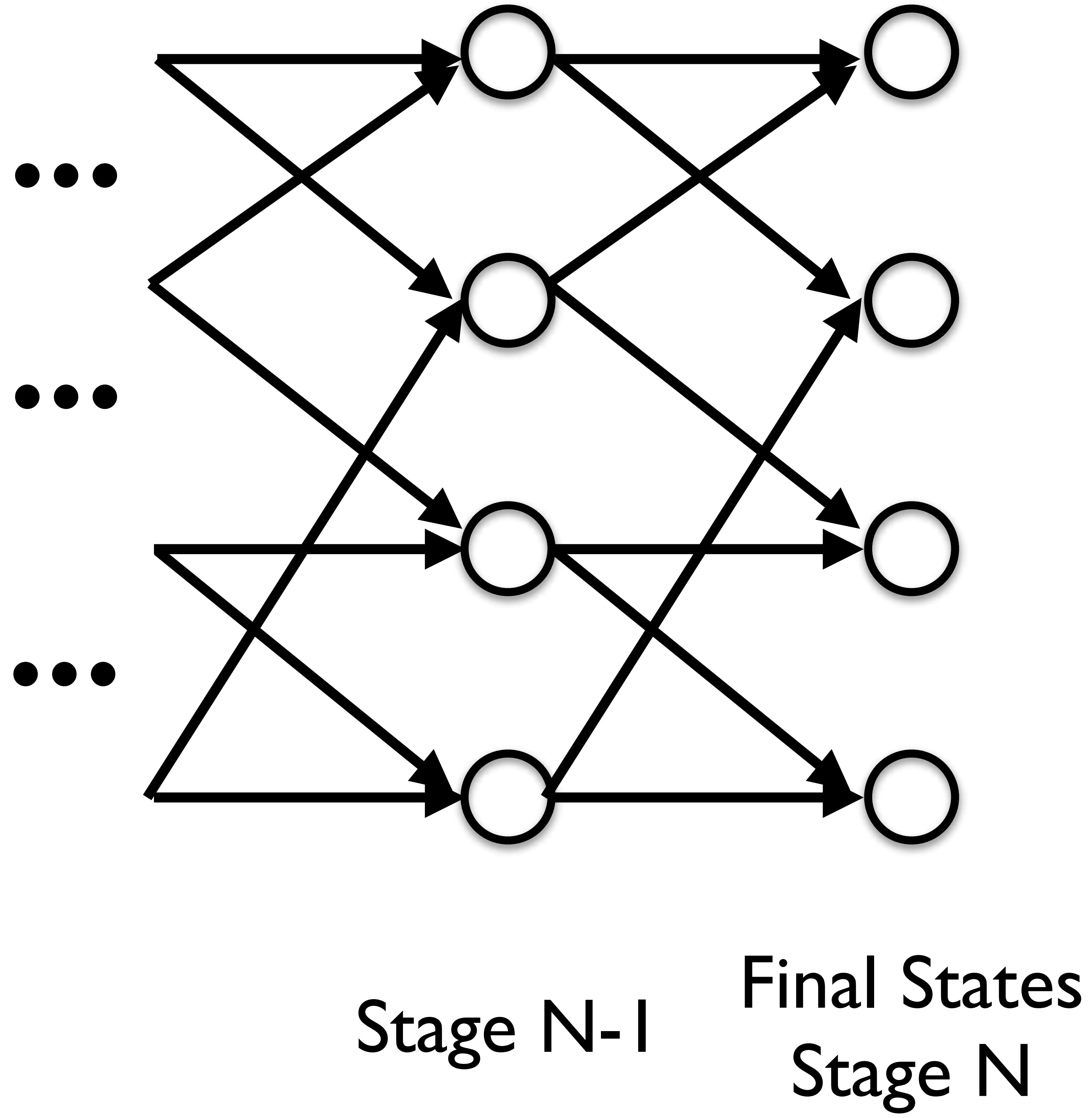


$J_N(X_N)$



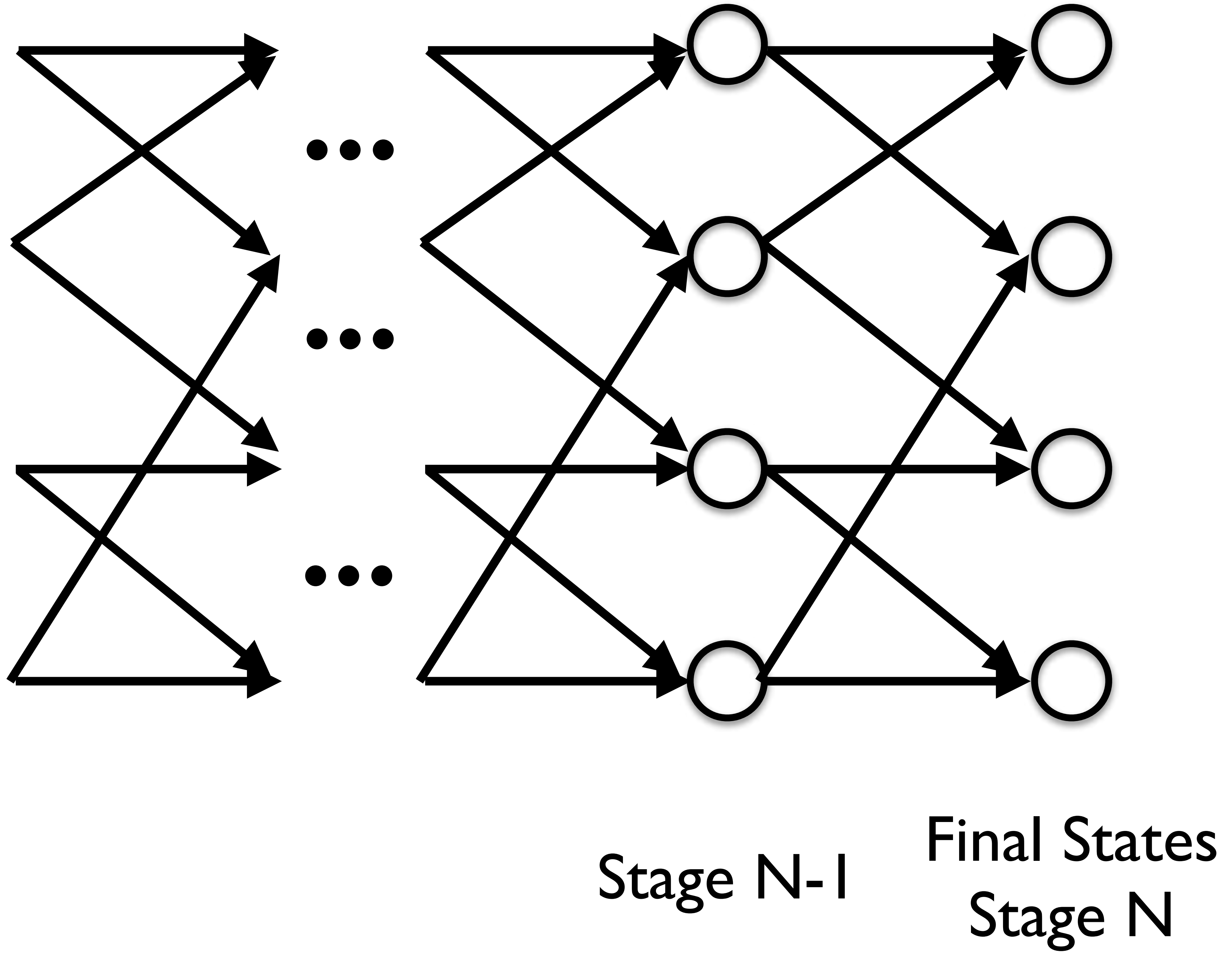
$$J_{N-1}(x_{N-1}) \longleftarrow J_N(X_N)$$

$$\mu_{N-1}(x_{N-1})$$



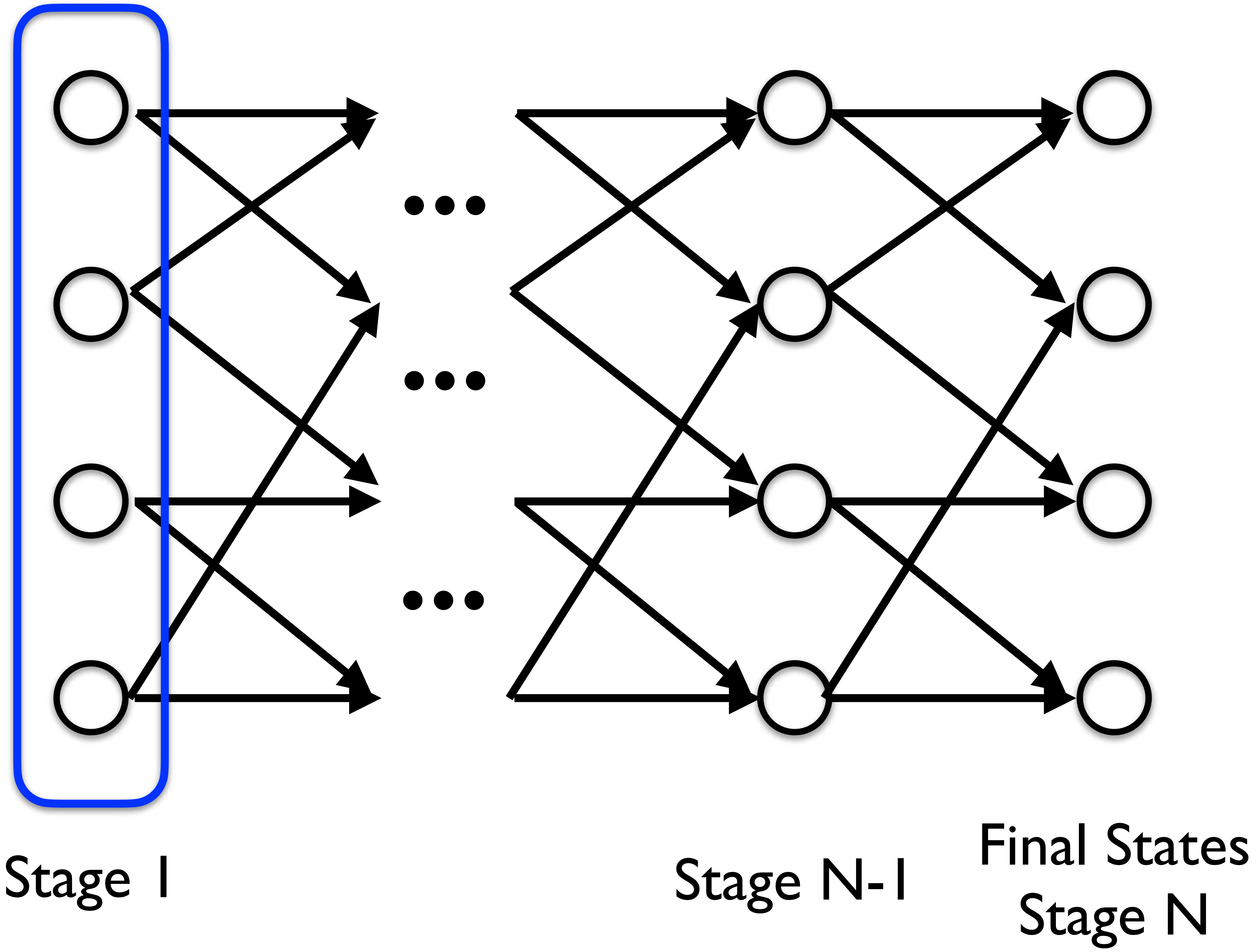
$$\cdots \longleftarrow \quad J_{N-1}(x_{N-1}) \longleftarrow J_N(X_N)$$

$$\mu_{N-1}(x_{N-1})$$



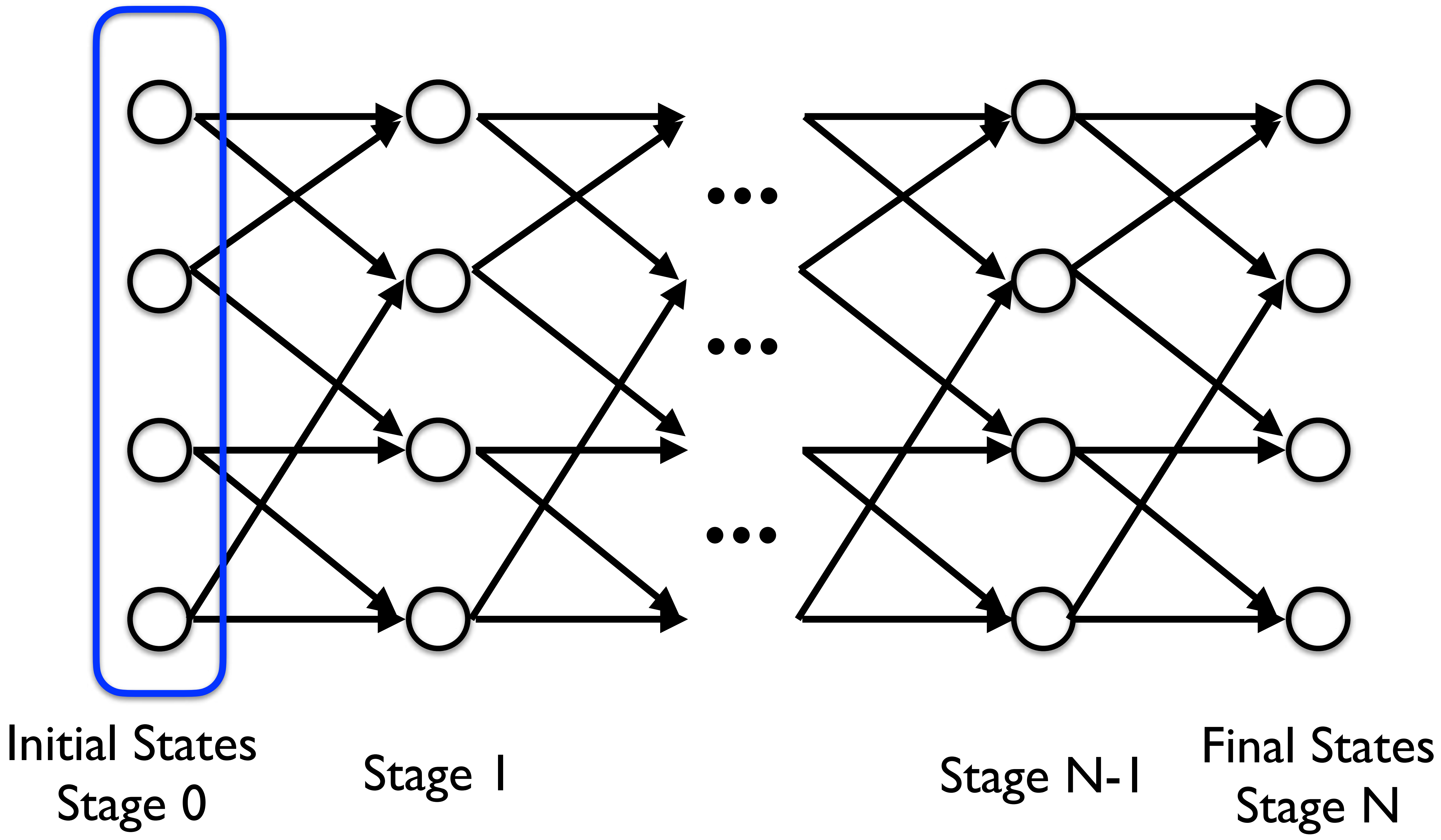
$$\longleftarrow \cdots \longleftarrow \quad J_{N-1}(x_{N-1}) \longleftarrow J_N(X_N)$$

$$\mu_{N-1}(x_{N-1})$$



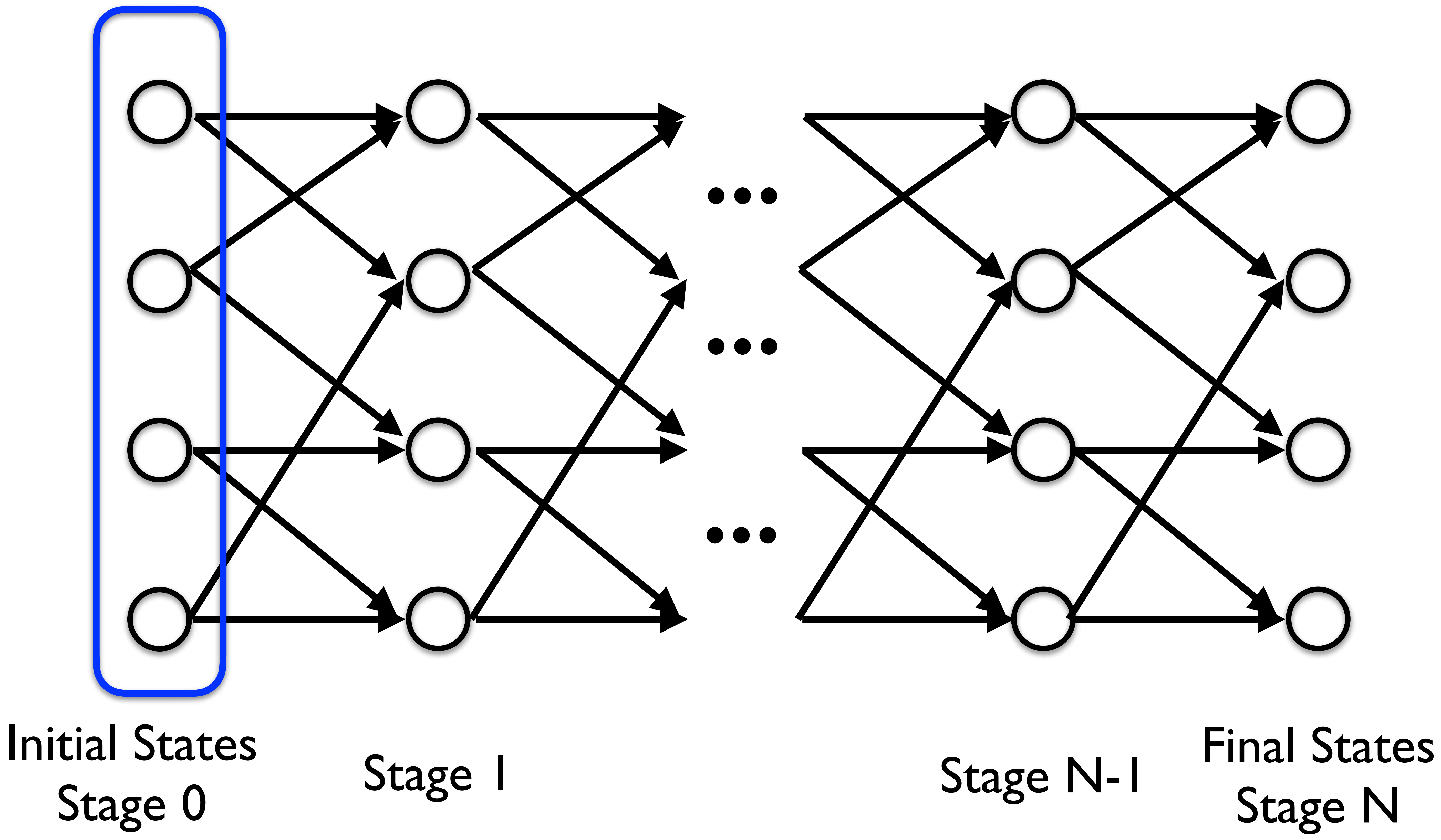
$$J_1(x_1) \quad \longleftarrow \cdots \longleftarrow \quad J_{N-1}(x_{N-1}) \longleftarrow J_N(X_N)$$

$$\mu_1(x_1) \qquad\qquad\qquad\qquad \mu_{N-1}(x_{N-1})$$



$$J_0(x_0) \longleftarrow \quad J_1(x_1) \quad \longleftarrow \cdots \longleftarrow \quad J_{N-1}(x_{N-1}) \longleftarrow J_N(X_N)$$

$$\mu_0(x_0) \qquad \mu_1(x_1) \qquad \qquad \mu_{N-1}(x_{N-1})$$



$\boxed{J_0(x_0)}$ Optimal cost for each initial state

$$\boxed{\mu_0(x_0) \longrightarrow \mu_1(x_1) \longrightarrow \cdots \longrightarrow \mu_{N-1}(x_{N-1})}$$

Globally optimal policy (for every state and stage)

Finite-horizon optimal control (non deterministic)

$$\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n, \boldsymbol{\omega}_n)$$

Uncertainty

Markov Decision Process (MDP)

<u>Markov property</u> knowledge of state n is sufficient to predict n+1
=> there is no need to remember previous states or actions

Note that $\mathbf{x}_n$ is now a random variable.
It is a usual vector if the noise ω_n is 0.

Finite-horizon optimal control (non deterministic)

$$\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n, \boldsymbol{\omega}_n)$$

Uncertainty

$$\cancel{\min_{u_0, u_1, \cdots, u_{N-1}} \sum_{i=0}^{N-1} g_i(x_i, u_i) + g_N(x_N)}$$

$$\min_{\mu_0(x_0), \cdots, \mu_{N-1}(x_{N-1})} \mathbb{E}\left(\sum_{i=0}^{N-1} g_i(x_i, u_i, \omega_i) + g_N(x_N) \right)$$

we now minimize the expected value of the cost
i.e. the "average cost" we can expect to get

Let $\pi^* = \{u_0^*, u_1^*, \cdots, u_{N-1}^*\}$ be an optimal policy for the original optimal control problem and assume that when using π^*, a given state x_k occurs at time k with a positive probability. Consider the subproblem where we reach x_k at time k and wish to minimize the *cost-to-go* from time k to time N

$$\mathbb{E}\left(\sum_{i=k}^{N-1} g_i(x_i, u_i, \omega_i) + g_N(x_N)\right)$$

Then the truncated policy $\{u_k^*, \cdots, u_{N-1}^*\}$ is optimal for this subproblem.

[Taken from Bertsekas, 2005]

For every initial state x_0, the optimal cost $J^*(x_0)$ of the optimal control problem is equal to $J_0(x_0)$ (i.e. the optimal cost to go from x_0) which is computed backward in time from stage $N-1$ to stage 0 using the following recursion:

$$J_N(x_n) = g_N(x_N)$$
$$J_k(x_k) = \min_{u_k} \mathop{\mathbb{E}}_{\omega_k} \left(g_k(x_k, u_k) + J_{k+1}(f(x_k, u_k))\right)$$

where the expectation is taken with respect to the probability distribution of ω_k, which depends on x_k and u_k. Furthermore, if $u_k^* = \mu_k^*(x_k)$ minimizes the cost-to-go for each x_k and k, the policy $\pi^* = \{\mu_0^*, \cdots, \mu_{N-1}^*\}$ is optimal.

[Taken from Bertsekas, 2005]

Dynamic Programming

Dynamic programming => find a global policy to optimize a cost over N stages under a dynamic process

Analytical solutions are typically hard to find (typically only for linear systems / quadratic costs)

Numerical solutions using backward recursion are typically used

The minimization in the backward recursion can be a problem

The curse of dimensionality

For every stage, we need to compute the cost-to-go for every possible states.

If we cannot find an analytical solution for a state that takes on real values, we need to discretize => exponential grows with dimension of states

Linear systems with quadratic costs (LQR)

$$\min_{x_n, u_n} \frac{1}{2} \sum_{n=0}^{N-1} x_n^T Q x_n + u_n^T R u_n + x_N^T Q x_N$$

$$\text{subject to } \; x_{n+1} = A x_n + B u_n$$

$$x_0 = x_{init}$$

Compute backward (from N to 0):

$$P_N = Q$$

$$K_n = (R + B^T P_n B)^{-1} B^T P_n A$$

$$P_{n-1} = Q + A^T P_n A - A^T P_n B K_n$$

K_n are called "feedback gains"

The cost to go at stage n is $J_n(x_n) = x_n^T P_n x_n$

The optimal policy is $\mu_n^*(x_n) = -K_n x_n$

Linear - Quadratic Regulator (LQR) => the non-deterministic case

$$\min \mathbb{E}\left(\sum_{i=0}^{N-1}\left(x_i^T Q x_i + u_i^T R u_i\right) + x_N^T Q_N x_N\right)$$

Quadratic cost

Subject to $x_{n+1} = A_n x_n + B_n u_n + \omega_n$

Linear dynamics

where ω_n has zero mean and finite variance

The optimal policy is the same as in the deterministic case! $\mu_n^*(x_n) = -K_n x_n$

The value function $J_0(x_0) = x_0^T P_0 x_0 + \sum_{n=0}^{N-1} \mathbb{E}(\omega_n^T P_{n+1} \omega_n)$

The actual cost differs depending on the disturbance

Infinite horizon problems

$$\lim_{N\to\infty} \quad \min_{u_n} \sum_{n=0}^{N-1} l_n(x_n, u_n)$$

We are often interested in infinite horizon problems
Find policies that work "all the time"

Can we do that?

This is what we call <u>infinite horizon</u> optimal control

In general the value function (the optimal cost) might go to infinity and special care is needed

LQR for infinite horizon control

For the LQR problem where A,B,Q and R are constant over time

$$\min \sum_{i=0}^{\infty} \left(x_i^T Q x_i + u_i^T R u_i \right)$$

Subject to $x_{n+1} = Ax_n + Bu_n$

It can be shown that $K_n = -(B^T P_{n+1} B + R)^{-1} B^T P_{n+1} A$

$$P_n = Q + A^T P_{n+1} A + A^T P_{n+1} B K_n$$

both converge when $n \to -\infty$

it means that $\lim_{n \to -\infty} K_n = K$ $\qquad \lim_{n \to -\infty} P_n = P$

LQR for infinite horizon control

For the LQR problem where A,B,Q and R are constant over time

$$\min \sum_{i=0}^{\infty} \left(x_i^T Q x_i + u_i^T R u_i \right)$$

Subject to $x_{n+1} = Ax_n + Bu_n$

$$\lim_{N \to \infty} \qquad P = Q + A^T PA + APB(B^T PB + R)^{-1} B^T PA$$

With constant feedback gains $K = -(B^T PB + R)^{-1} B^T PA$

Example

$$\min \sum_{i=0}^{N-1} \left(x_i^T Q x_i + u_i^T R u_i\right) + x_N^T Q x_N$$

Subject to $x_{n+1} = 2x_n + u_n$

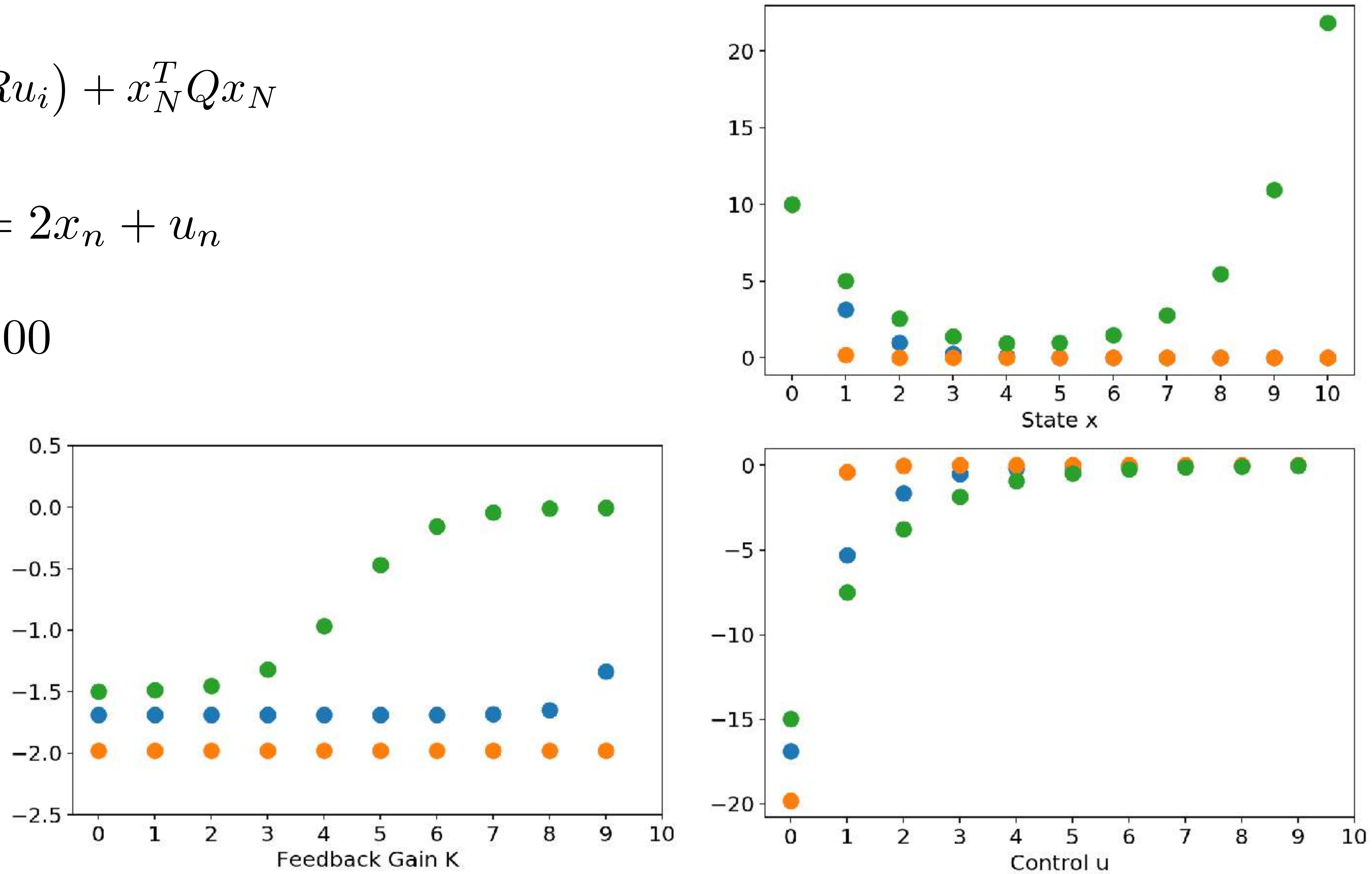
$N = 10 \qquad x_0 = 100$

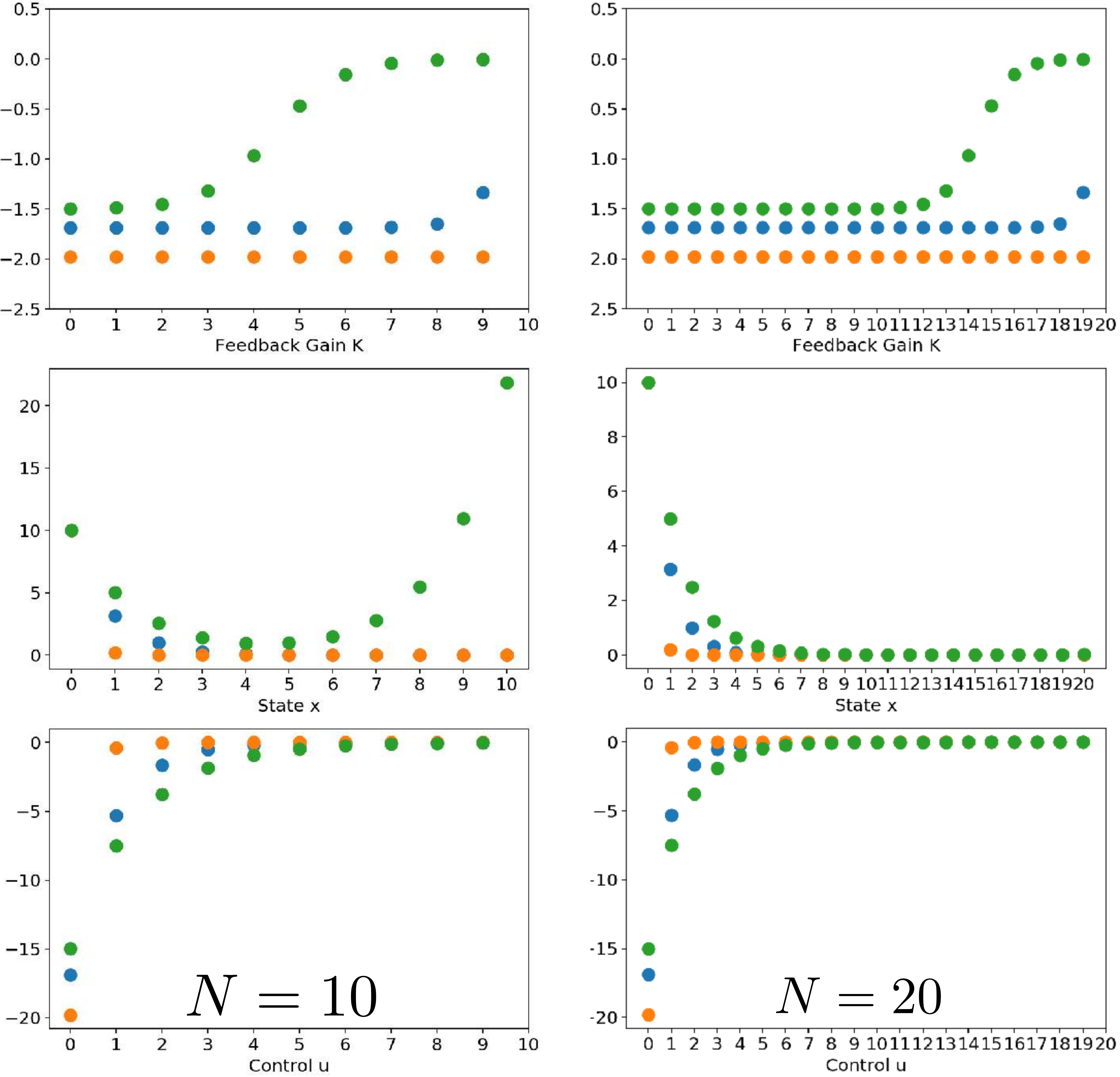
$Q = 2 \;\; R = 1$

$Q = 100 \;\; R = 1$

$Q = 1 \;\; R = 1000$

increase control cost leads to smaller gains and control but stabilization does not occur for N=10





Effect of the horizon N

$Q = 2 \; R = 1$

$Q = 100 \; R = 1$

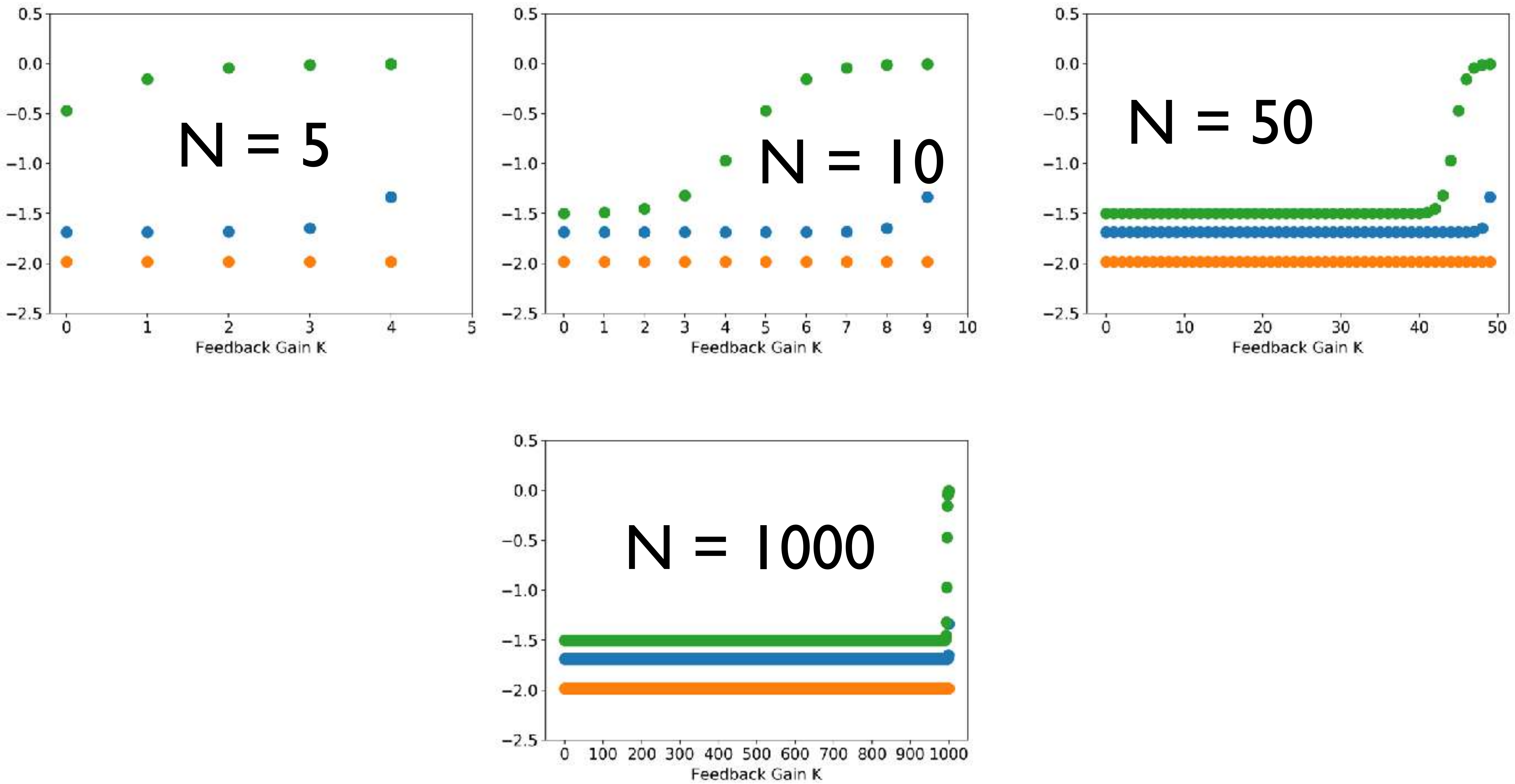
$Q = 1 \; R = 1000$

gains seem constant for early stages

increasing N seems to lead to more stable behavior even with low control

is it always true?

LQR for infinite horizon control



it means that $$\lim_{n \to -\infty} K_n = K \qquad \lim_{n \to -\infty} P_n = P$$

LQR for infinite horizon control

because of the convergence to K and P, the optimal control and the cost-to-go become stationary: they are the constant for every stage of the problem

this leads to a very elegant solution given by solving the following algebraic Riccati equation

$$P = Q + A^T P A - A^T P B (B^T P B + R)^{-1} (B^T P A)$$
$$K = -(B^T P B + R)^{-1} B^T P A$$

both the cost to go and the feedback gain are independent of n (i.e. constant for all stages)

$$u_n = K x_n \qquad J^*(x_0) = x_0^T P x_0$$

In the LQ case - dynamic programming leads to

$$P_n = Q \;\; + A^T P_{n+1} A \;\; + A^T P_{n+1} K_n$$

$$K_n = -(B^T P_{n+1} B \;\; + R \;\;)^{-1} B^T P_{n+1} A$$

Discrete-time Riccati equation

$\lim_{N \to \infty}$

P and K converge

1) global stationary value function $J(x) = x^T P x$
2) global stationary policy $\pi(x) = K x$

Do the policy and value function also converge in other cases?

Infinite horizon problems

We need time invariant dynamics $f(x, u)$ and costs $l(x, u)$

We are looking for a time-invariant policy $\pi(x, u)$

$$\lim_{N \to \infty} \min_{\pi(x_n)} \sum_{n=0}^{N} l(x_n, \pi(x_n))$$

subject to $\quad x_{n+1} = f(x_n, u_n)$

Infinite horizon problems

In general the sum of costs might diverge

We introduce <u>discounted costs</u> $$\lim_{N\to\infty} \min_{\pi(x_n)} \sum_{n=0}^{N} \alpha^n l(x_n, \pi(x_n))$$

$0 < \alpha < 1$ Discount factor

This will work as long as $l(x_n, \pi(x_n))$ is <u>bounded:</u> $|l(x,u)| < M$

Value iteration: dynamic programming until convergence

1) Start with $J_N(x_N) = g_N(x_N)$

2) Iterate backwards

$$J_n(x_n) = \min_{u_n} \underbrace{g\ (x_n, u_n)}_{\text{instantaneous cost at time n}} + \underbrace{J_{n+1}(f(x_n, u_n))}_{\text{optimal cost of the problem from stage n+1}}$$

$J_n(x_n)$ is called Cost-to-go

π^* is the optimal policy that solves the problem

$$J^*(x_0) = \min_{\mu_0(x_0)\ \cdots\ \mu_{N-1}(x_{N-1})} \left\{ g\ (x_N) + \sum_{k=0}^{N-1} g\ (x_k, \mu_k(x_k), \omega_k) \right\}$$

is called the optimal value function

Value iteration

$$J^*(x) = \lim_{N\to\infty} \min_{\mu(x)} \sum_{n=0}^{N} \alpha^n g(x_n, \mu(x_n))$$

Dynamic Programming algorithm (finite horizon)

$$J_n(x) = \min_{\mu(x)} g(x, u) + \alpha J_{n+1}(f(x, u))$$

Value iteration

For any bounded function $J(x)$, the iteration

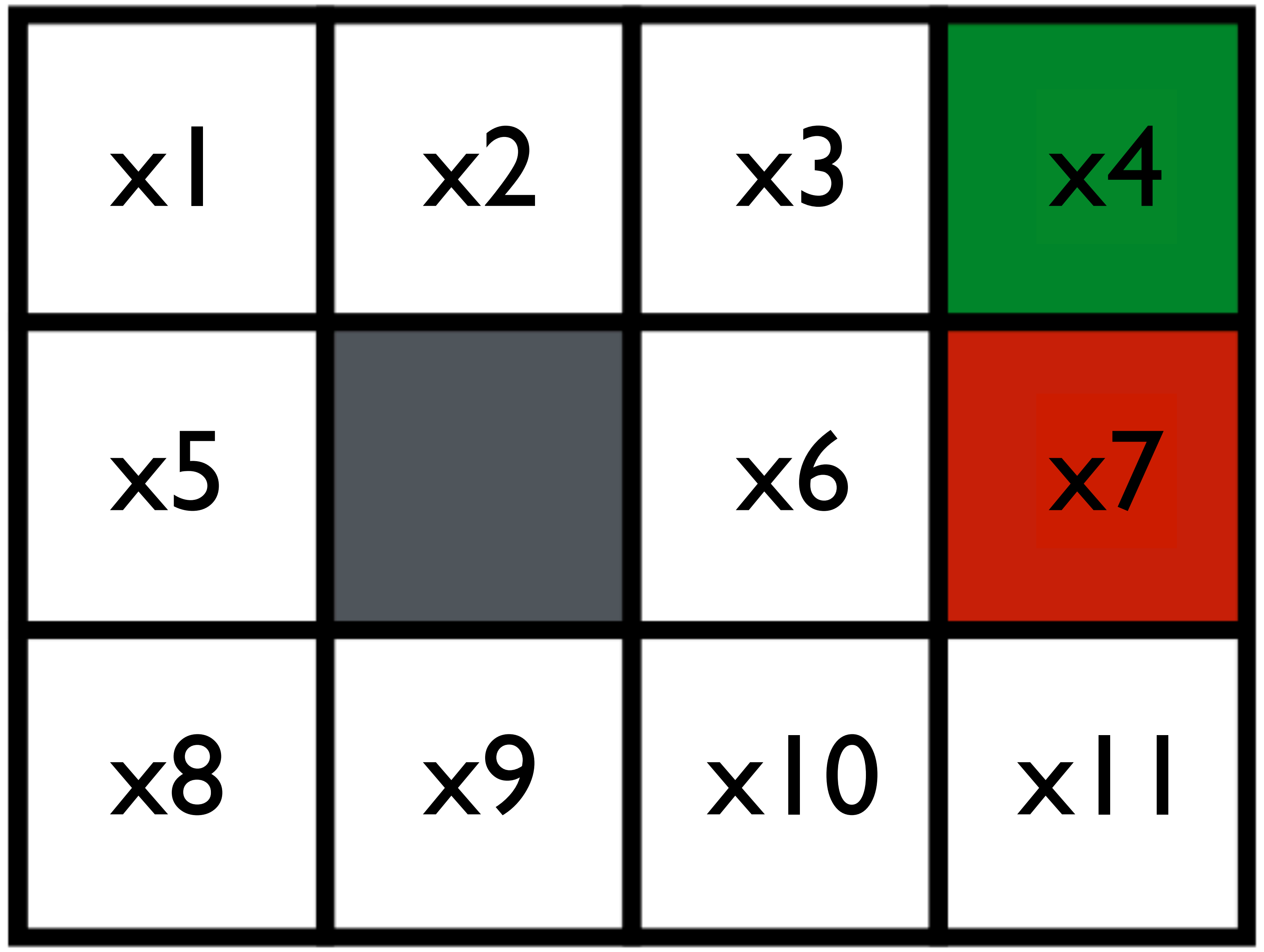
$$J^{n+1}(x) = \min_u g(x,u) + \alpha J^n(f(x,u))$$

with $J^0(x) = J(x)$ converges to the optimal value function, i.e.

$$\lim_{n \to \infty} J^{n+1}(x) = J^*(x)$$

<u>Value iteration algorithm</u>: start from an arbitrary J(x) and iterate $J^{n+1}(x)$

$$J_{n+1}(x) = \min_u l(x,u) + \alpha J_n(f(x,u))$$



Get out of the maze

- Red is bad (+1 cost)
- Green is good (-1 cost)
- Possible actions (N,E,W,S)
- $\alpha = 0.9$

$$J_{n+1}(x) = \min_u l(x,u) + \alpha J_n(f(x,u))$$

0	0	0	-1
0		0	1
0	0	0	0

- Initialize J_0

$$J_{n+1}(x) = \min_u l(x,u) + \alpha J_n(f(x,u))$$

0	0	-0.9	-1.9
0		0	1.9
0	0	0	0

- First iteration of Bellman (we update every state)

$$J_{n+1}(x) = \min_u l(x,u) + \alpha J_n(f(x,u))$$

0	-0.81	-1.71	-2.71
0		-0.81	2.71
0	0	0	0

- 2nd Iteration

$$J_{n+1}(x) = \min_u l(x,u) + \alpha J_n(f(x,u))$$

-0.73	-1.54	-2.44	-3.44
0		-1.54	3.44
0	0	-0.73	0

- 3rd Iteration

$$J_{n+1}(x) = \min_u l(x,u) + \alpha J_n(f(x,u))$$

-1.39	-2.2	-3.1	-4.1
-0.66		-2.2	4.1
0	-0.66	-1.39	-0.66

- 4th Iteration

$$J_{n+1}(x) = \min_{u} l(x,u) + \alpha J_n(f(x,u))$$

-4.15	-4.96	-5.86	-6.86
-3.42		-4.96	6.86
-2.77	-3.42	-4.15	-3.42

- 10th Iteration

$$J_{n+1}(x) = \min_u l(x,u) + \alpha J_n(f(x,u))$$

-7.29	-8.1	-9	-10
-6.56		-8.1	10
-5.9	-6.56	-7.29	-6.56

- 100th Iteration

$$J_{n+1}(x) = \min_u l(x,u) + \alpha J_n(f(x,u))$$

-7.29	-8.1	-9	-10
-6.56		-8.1	10
-5.9	-6.56	-7.29	-6.56

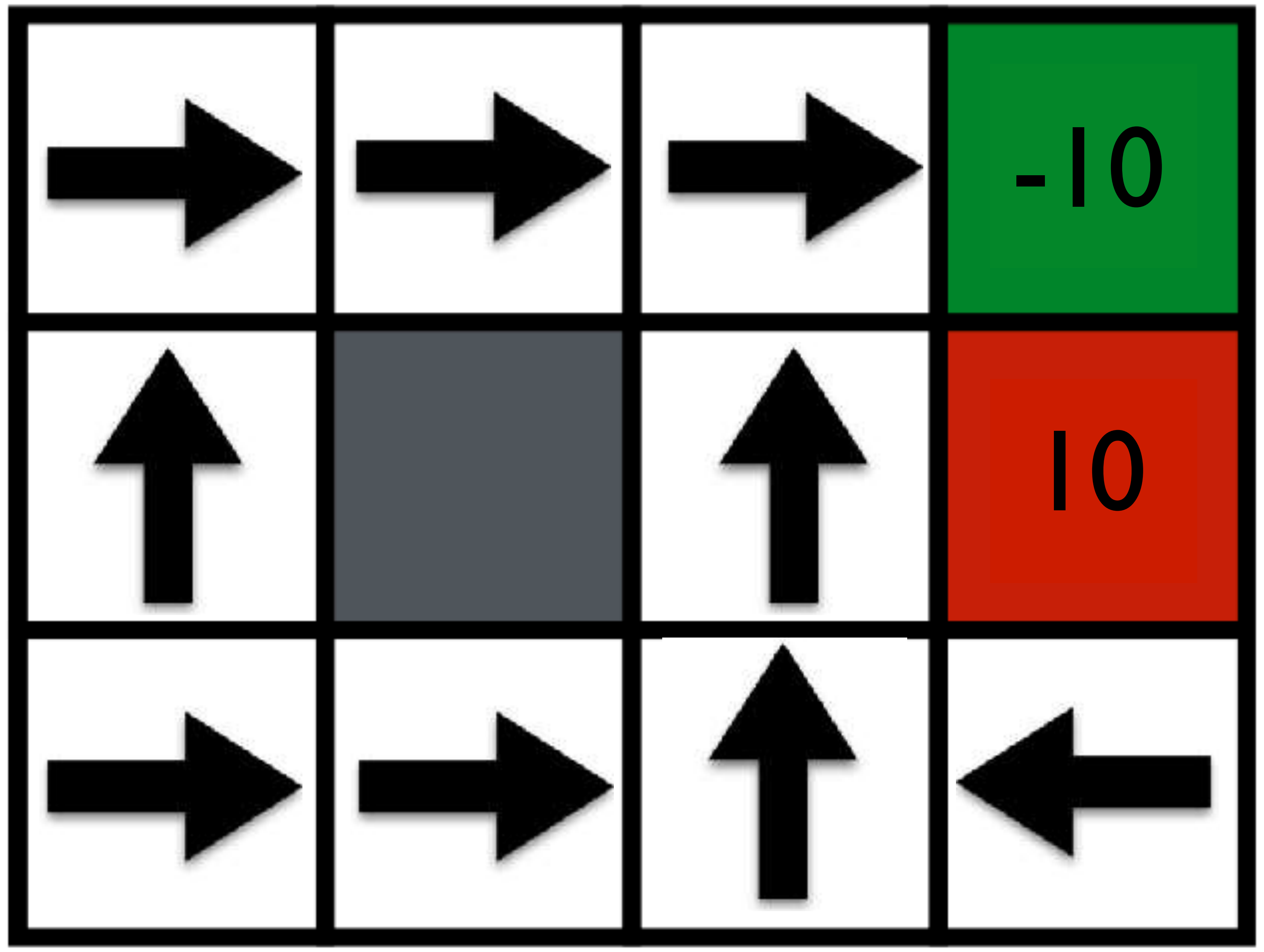
- 1000th Iteration

-7.29	-8.1	-9	-10
-6.56		-8.1	10
-5.9	-6.56	-7.29	-6.56

- 1000th Iteration

We have converged and found the optimal value function

The policy is read out by following the action that creates the lowest next value + current cost

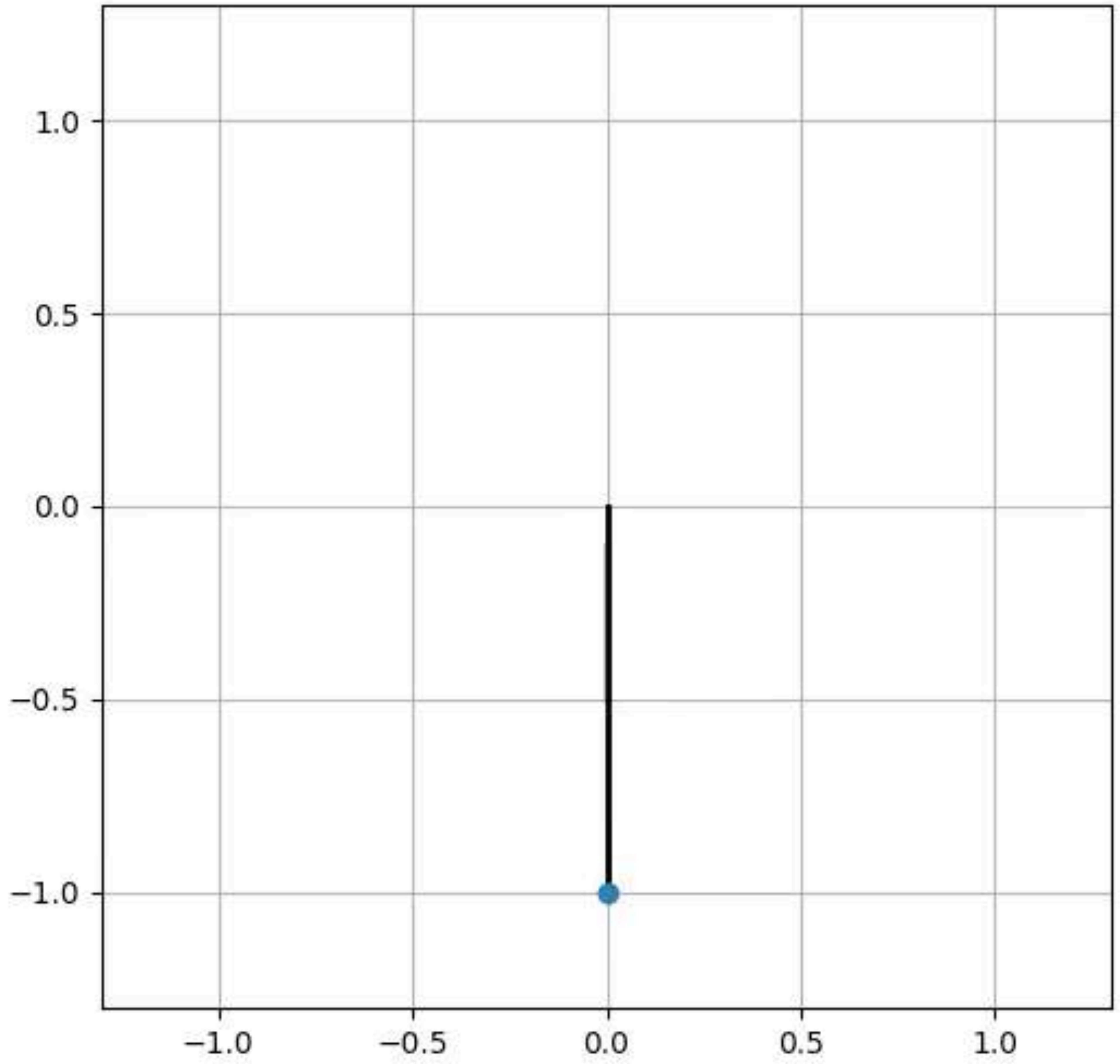


- 1000th Iteration

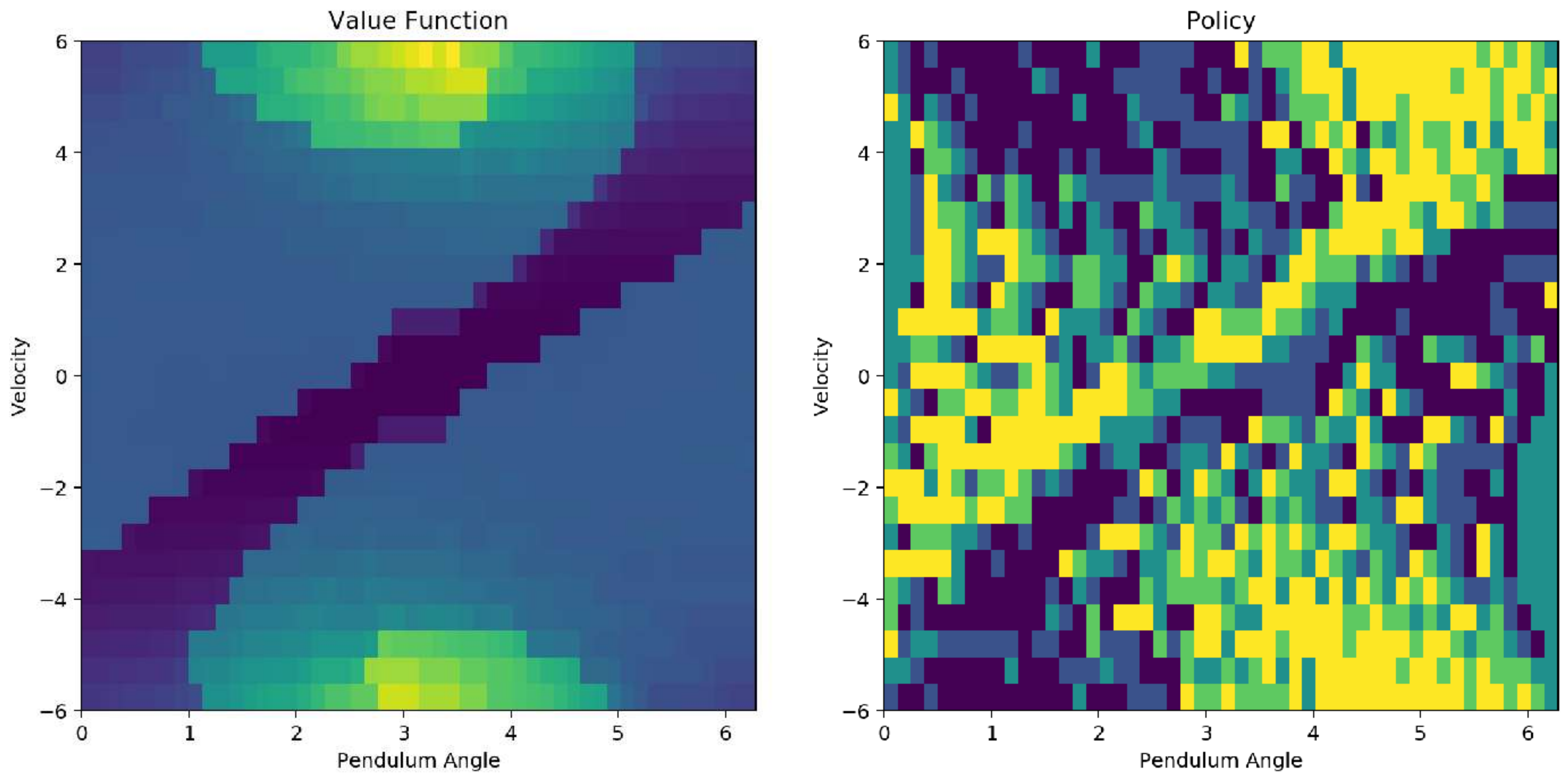
We have converged and found the optimal value function

The policy is read out by following the action that creates the lowest next value

VI on inverted pendulum



VI on inverted pendulum



Policy evaluation: how good is a policy?

Policy evaluation

We would like to know the total cost of using a certain stationary policy $\mu(x)$

$$J_\mu(x) = \lim_{N\to\infty} \sum_{n=0}^{N} \alpha^n g(x_n, \mu(x_n))$$

Since the sum is infinite, we can rewrite as

$$\begin{aligned} J_\mu(x_0) &= g(x_0, \mu(x_0)) + \lim_{N\to\infty} \sum_{n=1}^{N} \alpha^n g(x_n, \mu(x_n)) \\ &= g(x_0, \mu(x_0)) + \alpha J_\mu(f(x_0, \mu(x_0))) \end{aligned}$$

Policy evaluation

For an arbitrary $J(x)$,

$$J_\mu^{n+1}(x) = g(x, \mu(x)) + \alpha J_\mu^n(f(x, \mu(x))$$

is the cost of using policy $\mu(x)$ for the $N = n + 1$ optimal control problem with cost $(\sum_{n=0}^{N-1} \alpha^n g(x_n, \mu(x_n))) + \alpha^N J(x_N)$ and dynamics $x_{n+1} = f(x_n, u_n)$

when $n \to \infty$, the contribution of the final cost will converge to 0 and we can expect that $\lim_{n\to\infty} J_\mu^n$ converges then to $J_\mu(x)$, independently of the initial $J(x)$

Policy evaluation algorithm I

For any bounded function $J(x)$, the iteration

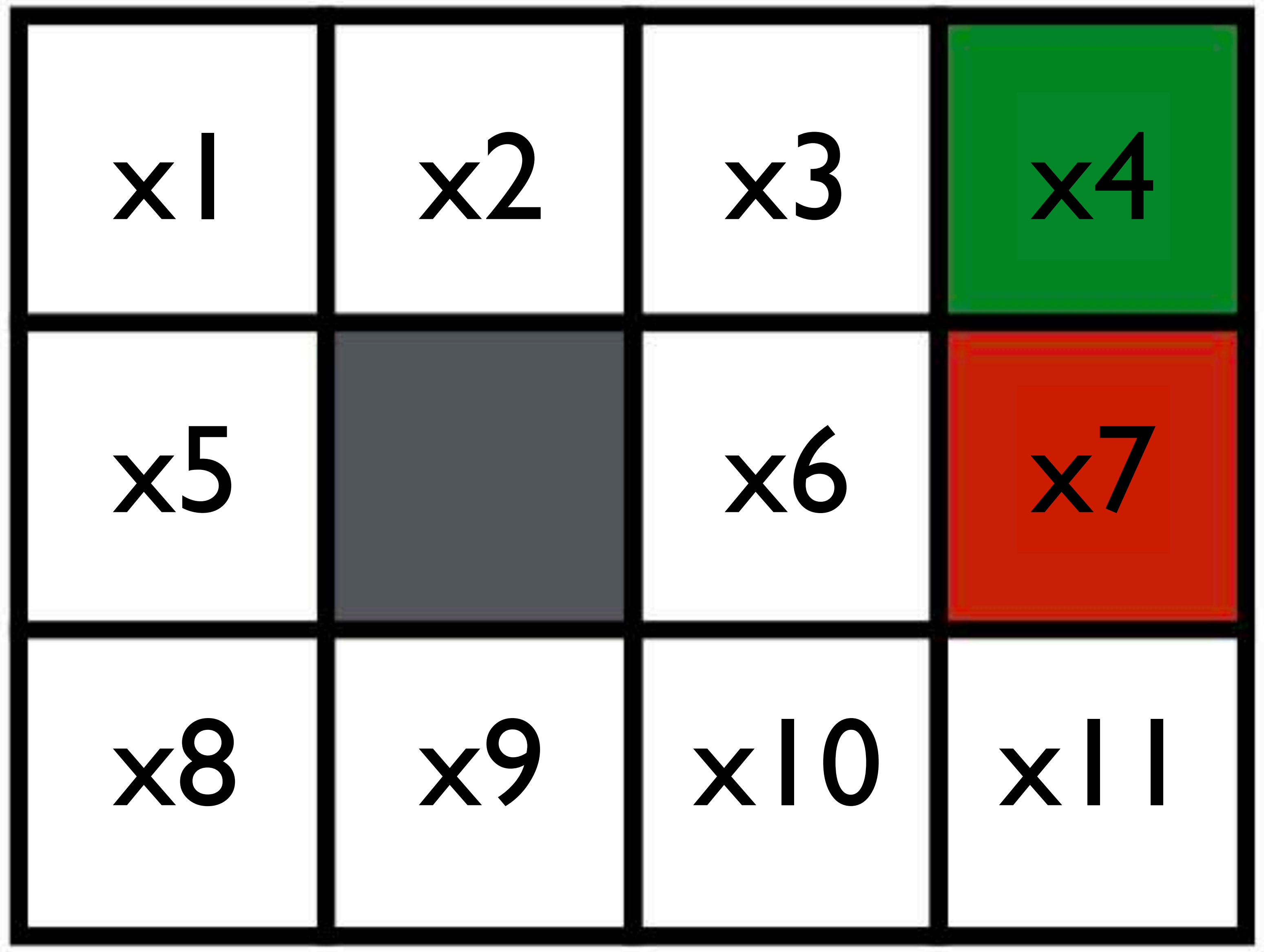
$$J_\mu^{n+1}(x) = g(x, \mu(x)) + \alpha J_\mu^n(f(x, \mu(x)))$$

converges to the total cost of the stationary policy $\mu(x)$, i.e.

$$\lim_{n\to\infty} J_\mu^n(x) = J_\mu(x)$$

Policy evaluation algorithm: start from an arbitrary J(x) and iterate $J_\mu^{n+1}(x)$

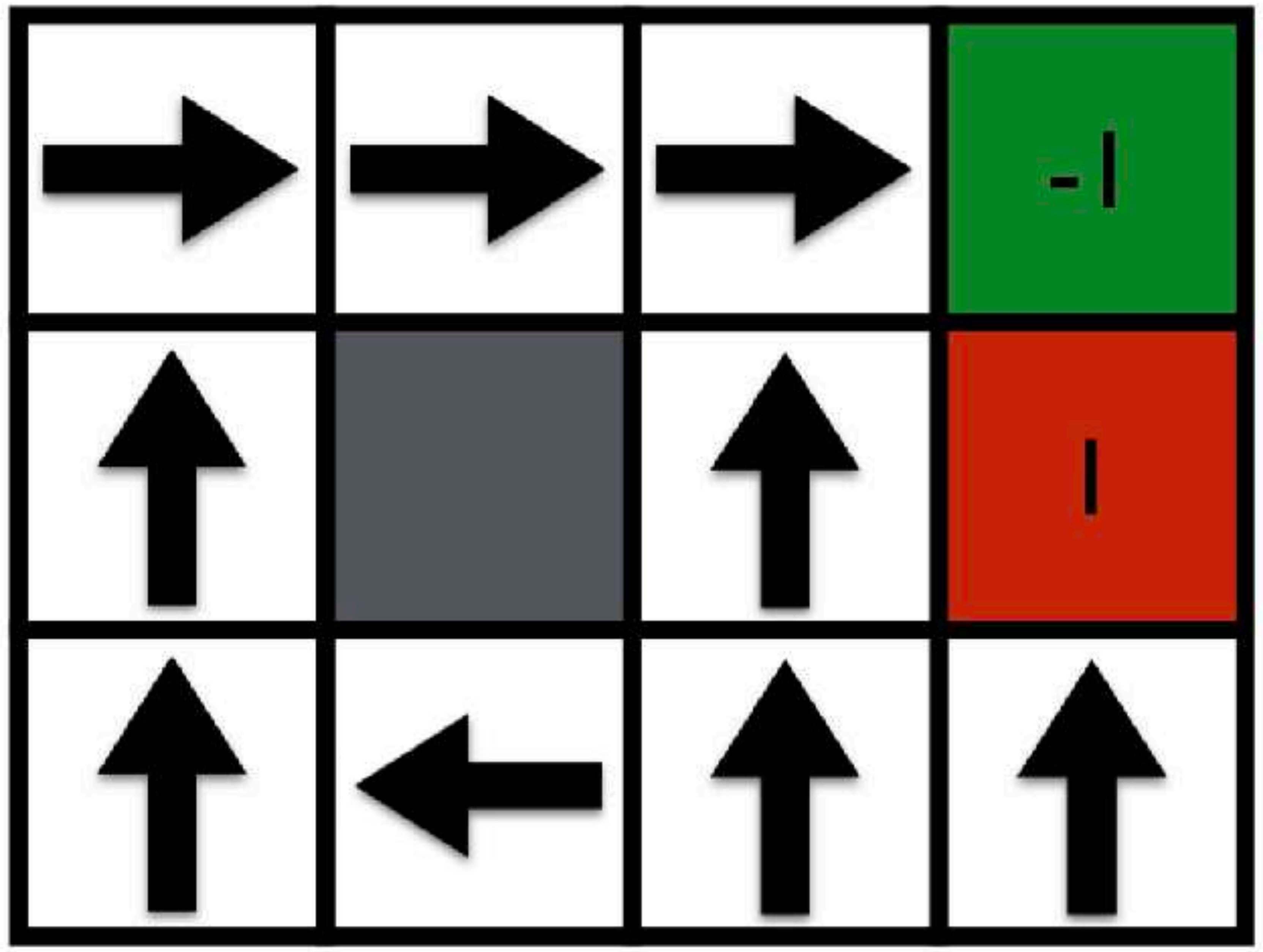
Policy evaluation: example



Get out of the maze

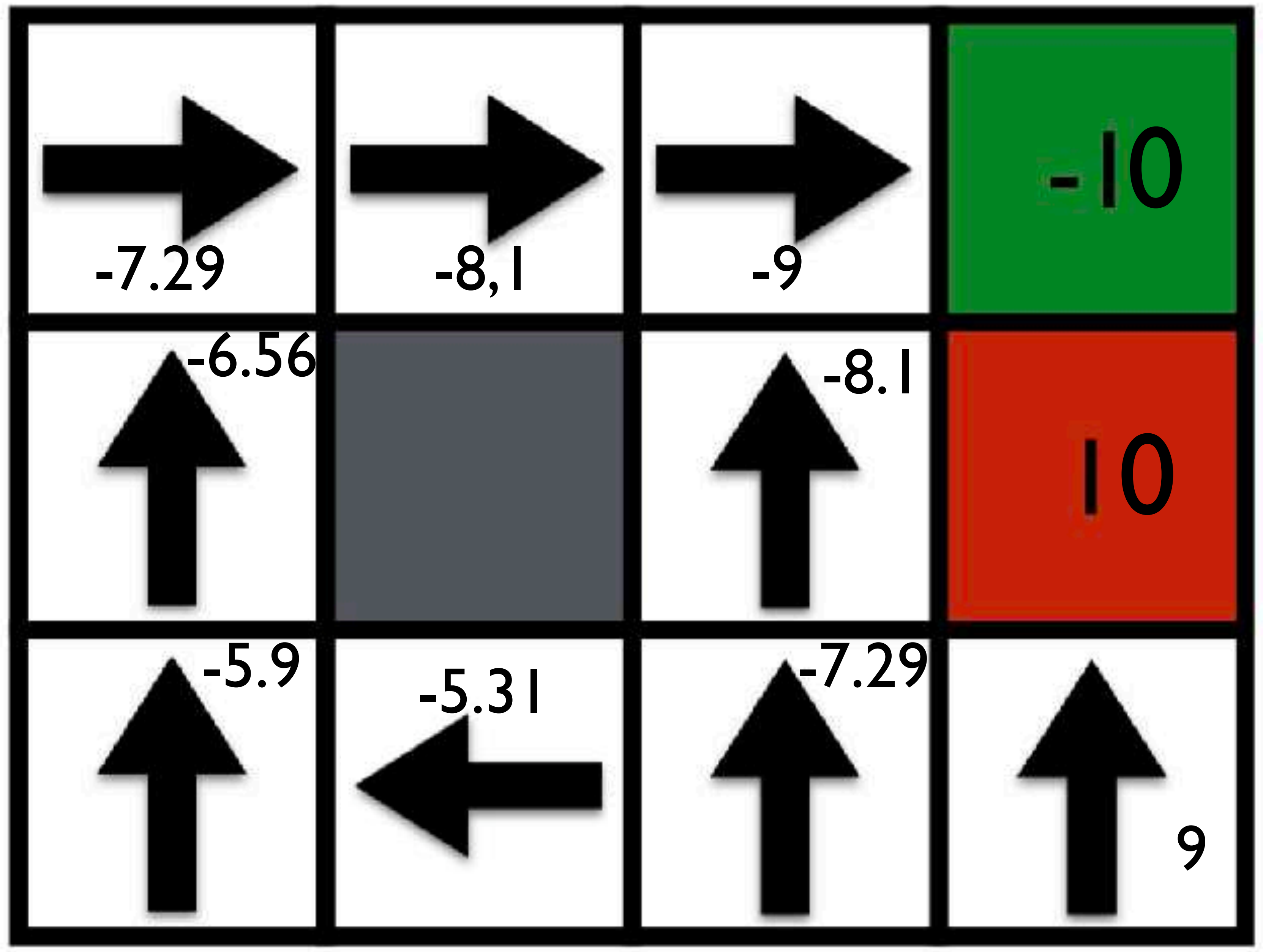
- Red is bad (+1 cost)
- Green is good (-1 cost)
- Possible actions (N,E,W,S)
- $\alpha = 0.9$

Given a policy find its value function



Get out of the maze

- Red is bad (+1 cost)
- Green is good (-1 cost)
- Possible actions (N,E,W,S)
- $\alpha = 0.9$

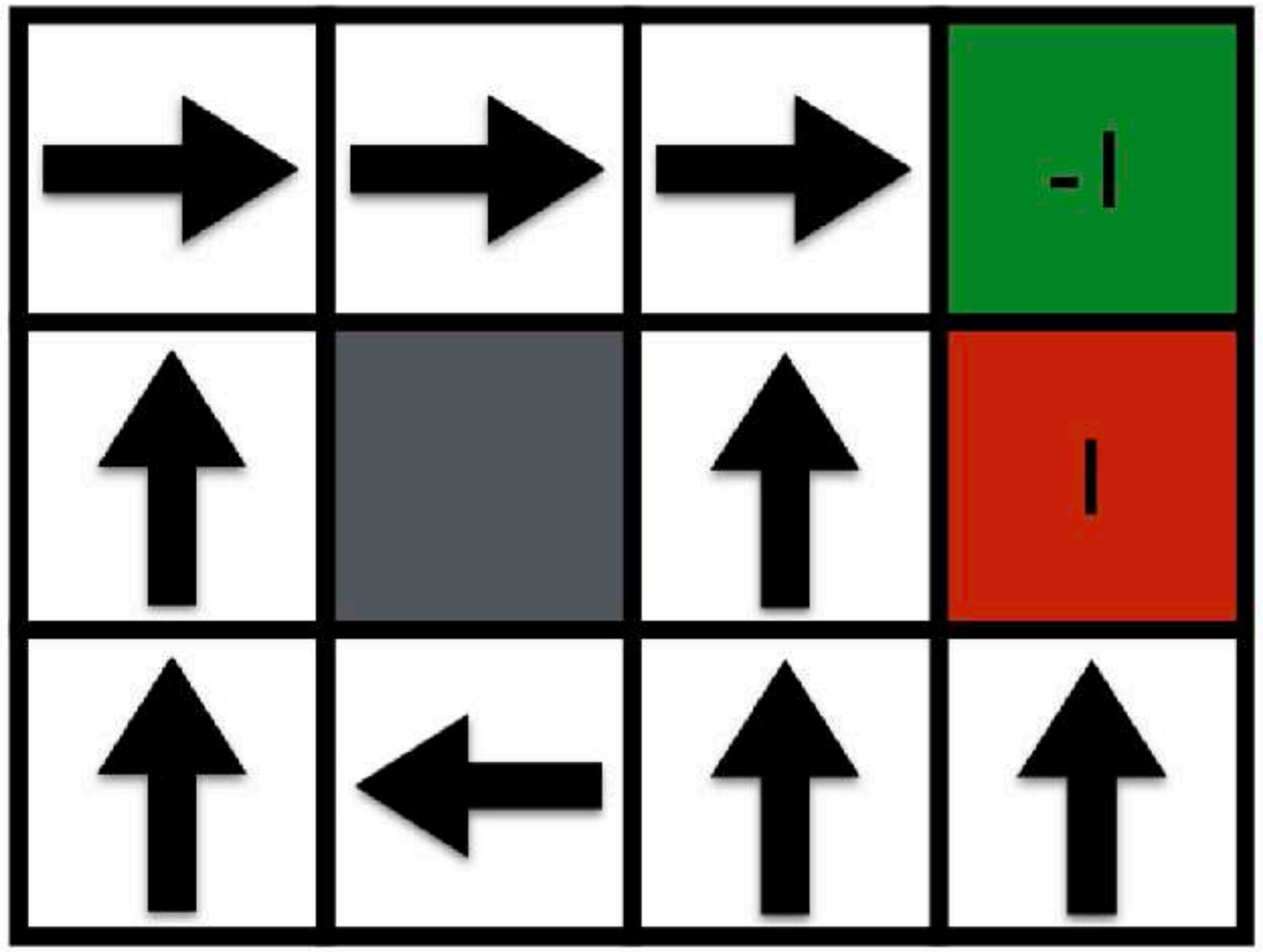


Compute policy cost

Policy evaluation 2

For finite number of states, it is possible to evaluate the policy without iterating - just through solving a linear equation

Given a policy find its value function



The policy can be evaluated by solving the following

$$(I - \alpha A)J_\mu = \bar{g}$$

$A =$

```
[[0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]]
```

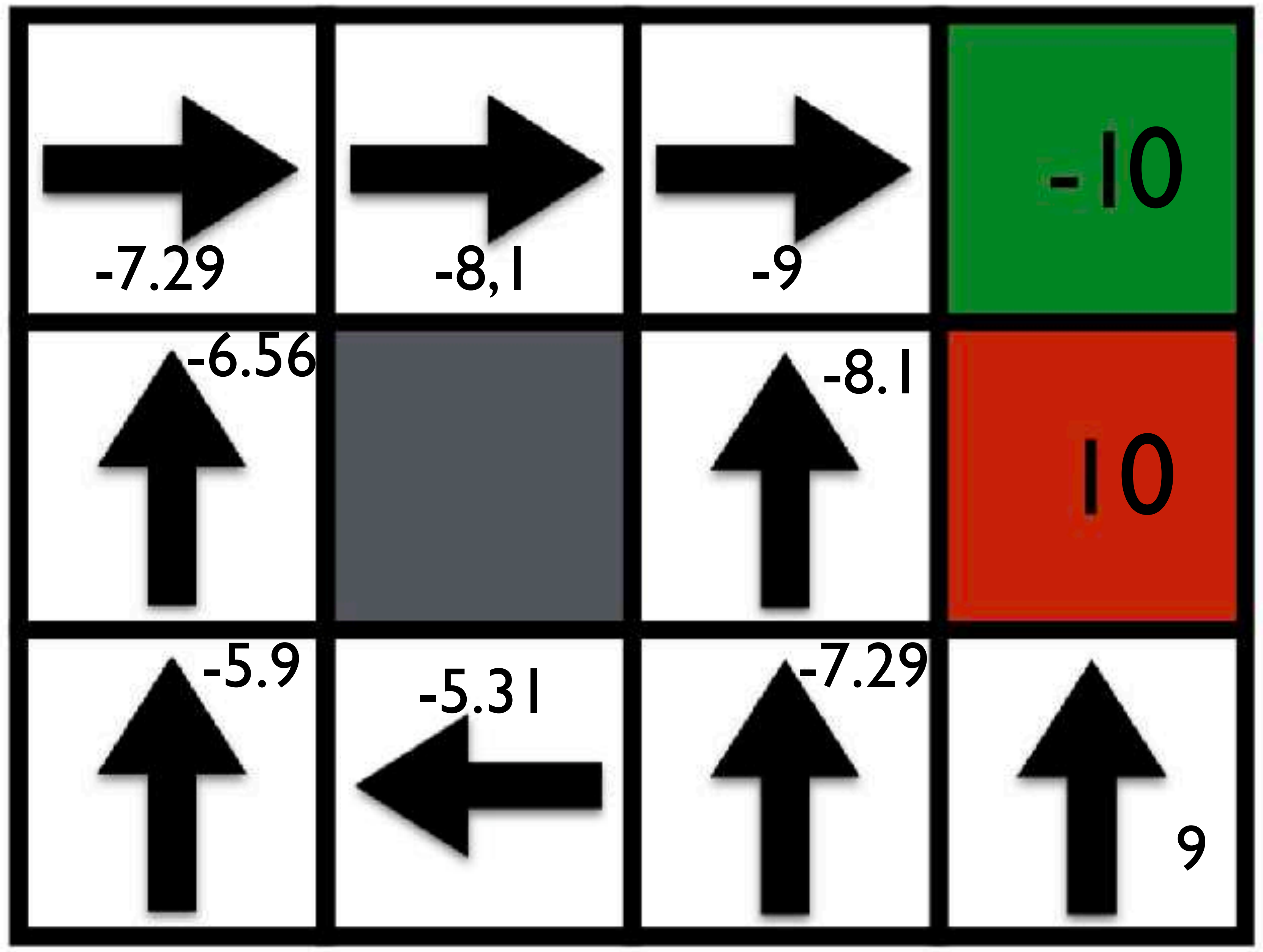
$\bar{g} =$

```
[[ 0]
 [ 0]
 [ 0]
 [-1]
 [ 0]
 [ 0]
 [ 1]
 [ 0]
 [ 0]
 [ 0]
 [ 0]]
```

$J_\mu = (I - \alpha A)^{-1}\bar{g} =$

```
[[ -7.29    ]
 [ -8.1     ]
 [ -9.      ]
 [-10.      ]
 [ -6.561   ]
 [ -8.1     ]
 [ 10.      ]
 [ -5.9049  ]
 [ -5.31441]
 [ -7.29    ]
 [  9.      ]]
```

Compute policy cost



Policy evaluation 2

Write the possible states as x_i from 1 to N (i.e. we have N states)

We can write the dynamics $f(x, \mu(x))$ with a matrix A that sends a given state to its next state

For example, the transition from x_1 can be written

$$x_{next} = f(x_1, \mu(x_1)) = [0 \ \cdots \ 1 \ \cdots] \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}$$

where $[0 \ \cdots \ 1 \ \cdots]$ simply selects the new states from the vector of states. The row vectors defining the transition from each state compose A

Policy evaluation 2

Let $\bar{g}$ the vector of costs for all the states, i.e. $\bar{g} = \begin{pmatrix} g(x_1, \mu(x_1)) \\ g(x_2, \mu(x_2)) \\ \vdots \\ g(x_N, \mu(x_N)) \end{pmatrix}$

J_μ can also be written as a vector $\begin{pmatrix} J_\mu(x_1) \\ J_\mu(x_2) \\ \vdots \\ J_\mu(x_N) \end{pmatrix}$

$J_\mu(f(x, \mu(x)))$ can be written using the matrix A of transitions such that we get the following linear equation

$$J_\mu = \bar{g} + \alpha A J_\mu$$

Policy evaluation 2

A policy μ can be evaluated by solving the linear equation

$$(I - \alpha A)J_\mu = \bar{g}$$

which is very easy to compute!
(no need to use the recursion to evaluate a policy)

Policy Evaluation Algorithm II

Given a policy μ :

Solve $(I - \alpha A)J_{\mu} = \bar{g}$

Policy Iteration algorithm: optimal policy through policy evaluation

Start with an initial guess for the policy μ_0

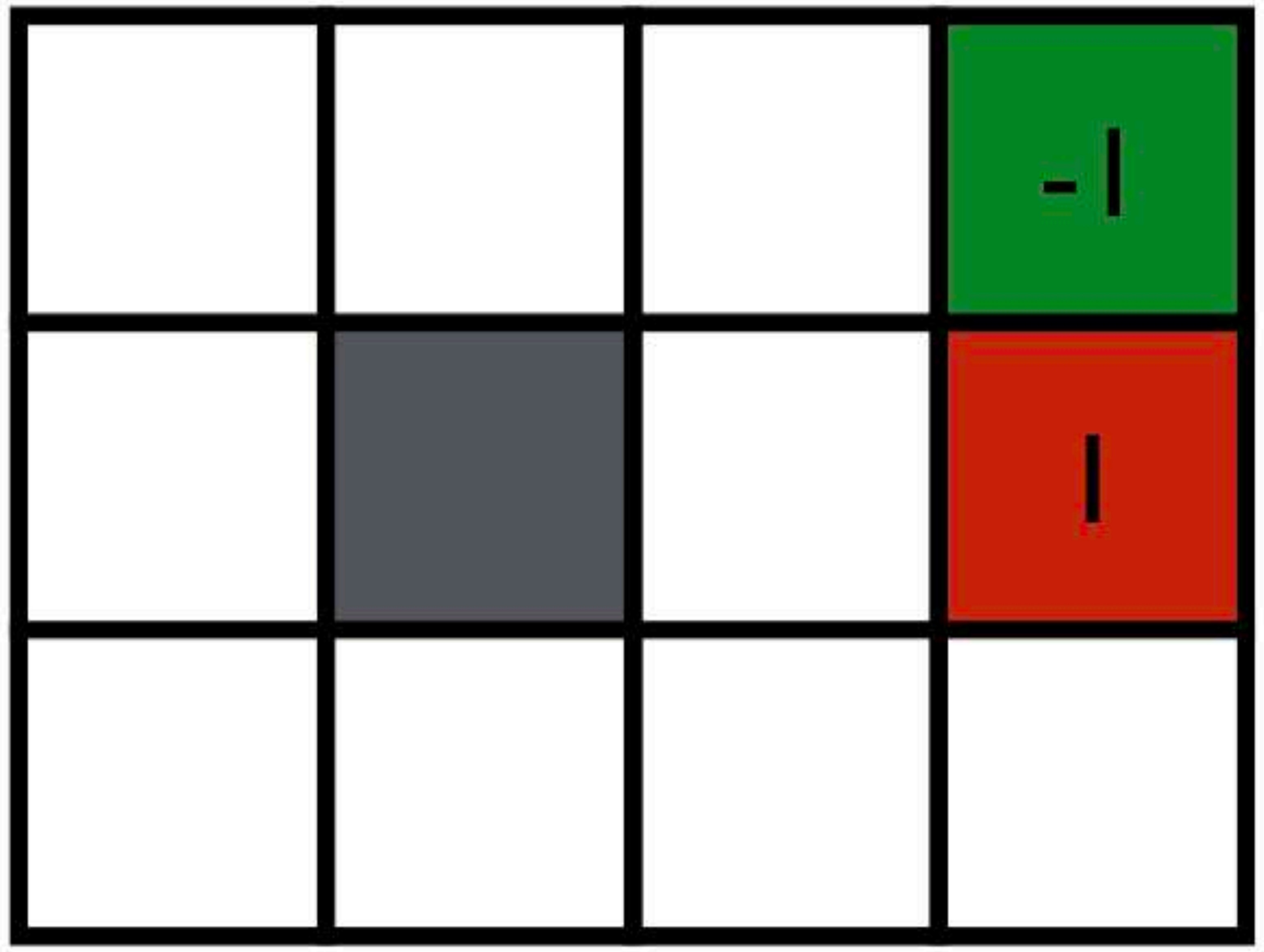
1. Policy evaluation step:
   Compute $J_{\mu_n}(x)$ using the policy evaluation algorithm

2. Policy update step:
   Update the policy using

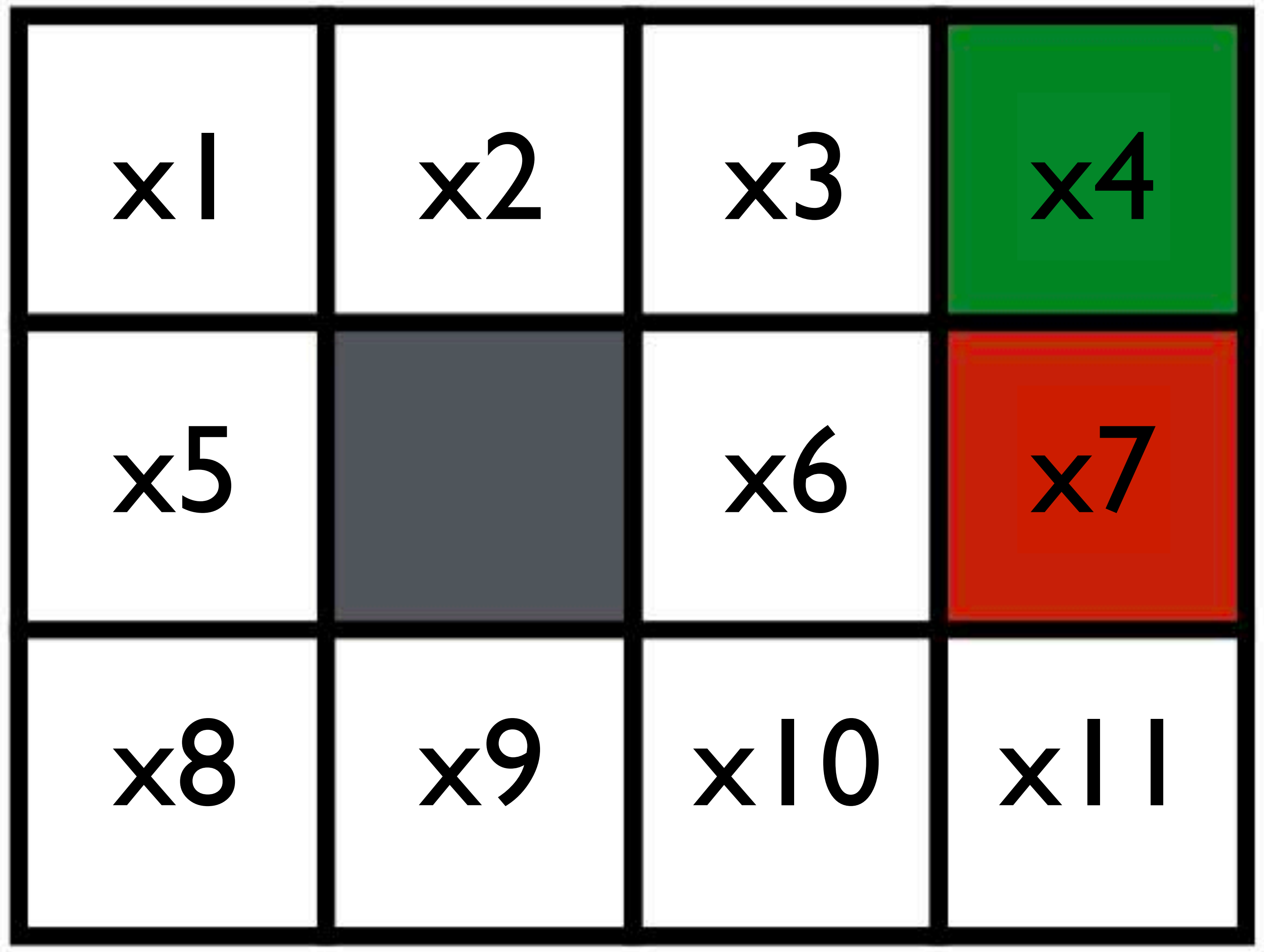
$$\mu_{k+1} = \arg\min_u g(x,u) + \alpha J_{\mu_k}(f(x,u))$$

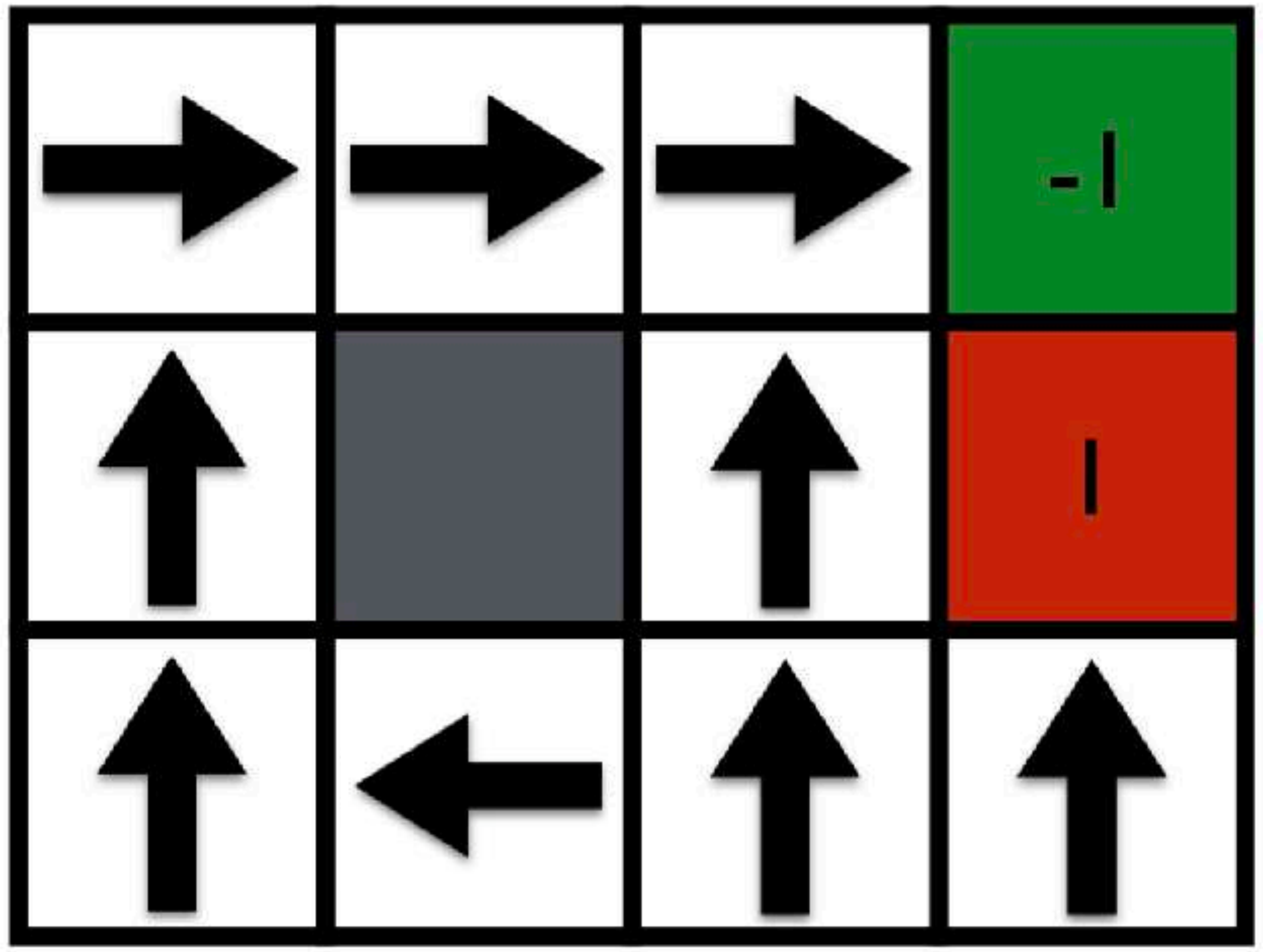
Iterate until convergence (happens in a finite number of iteration)

Policy iteration: canonical example



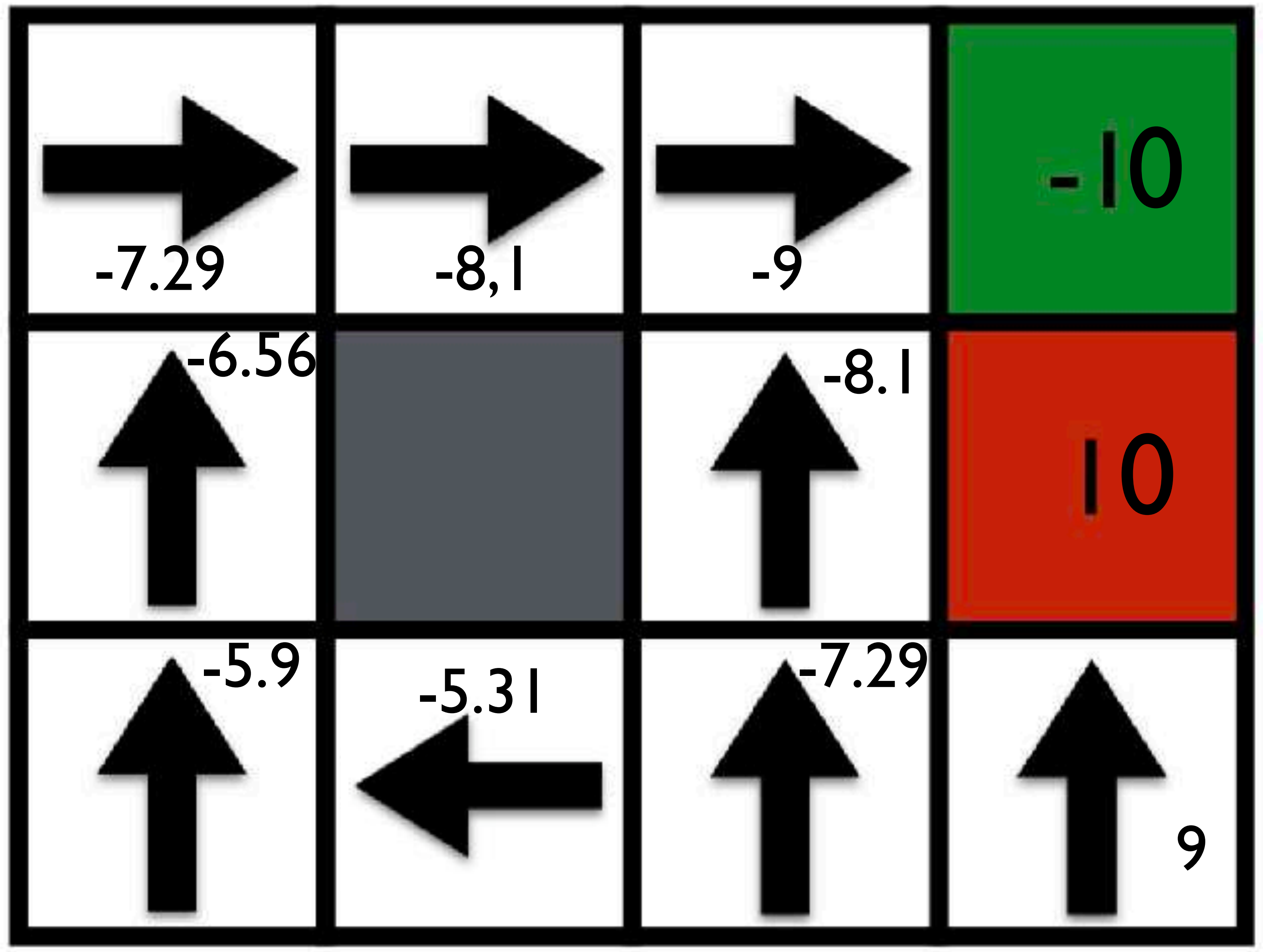
Policy iteration: canonical example





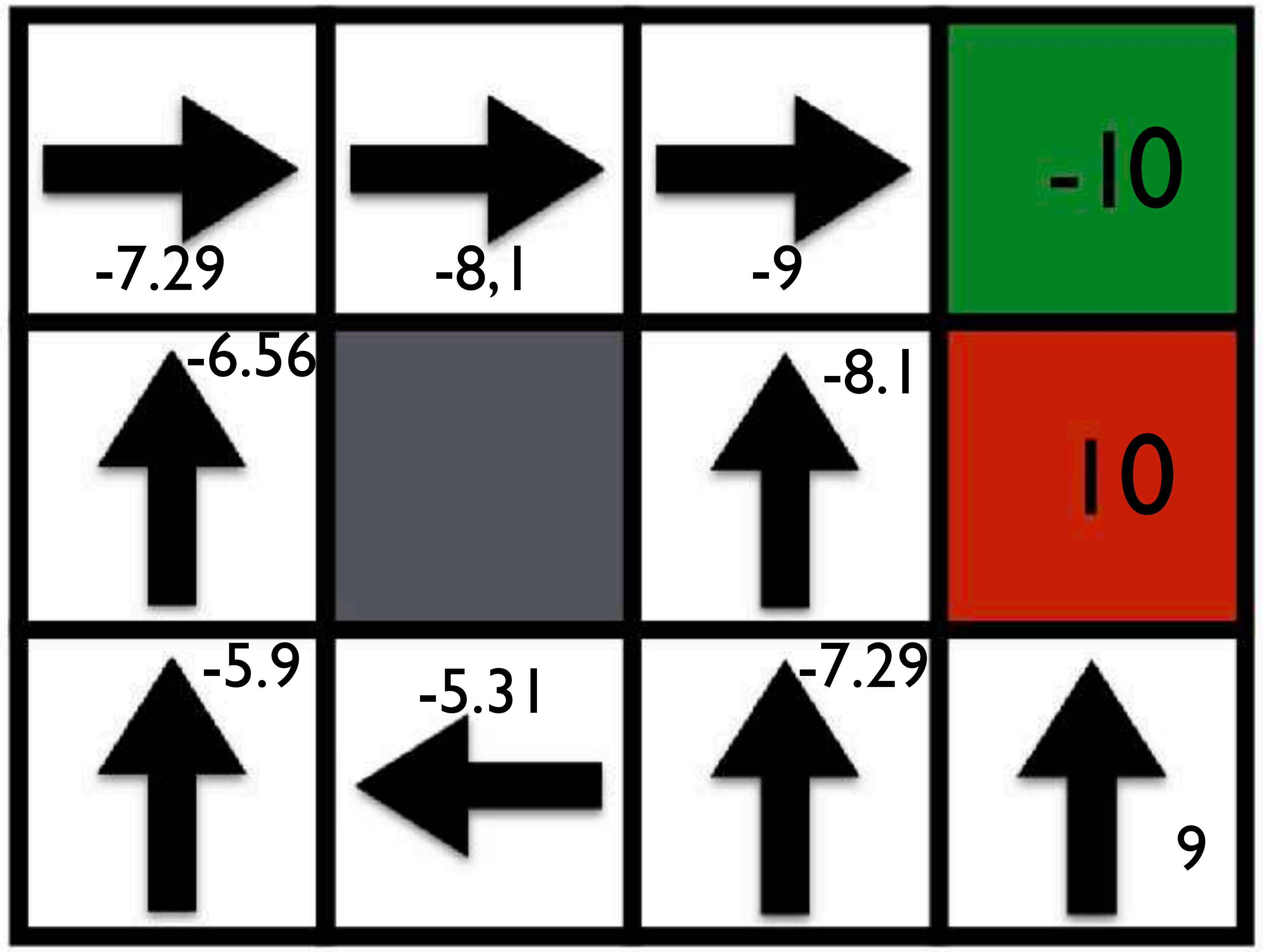
Initialize policy guess

1) Given a policy find its value function



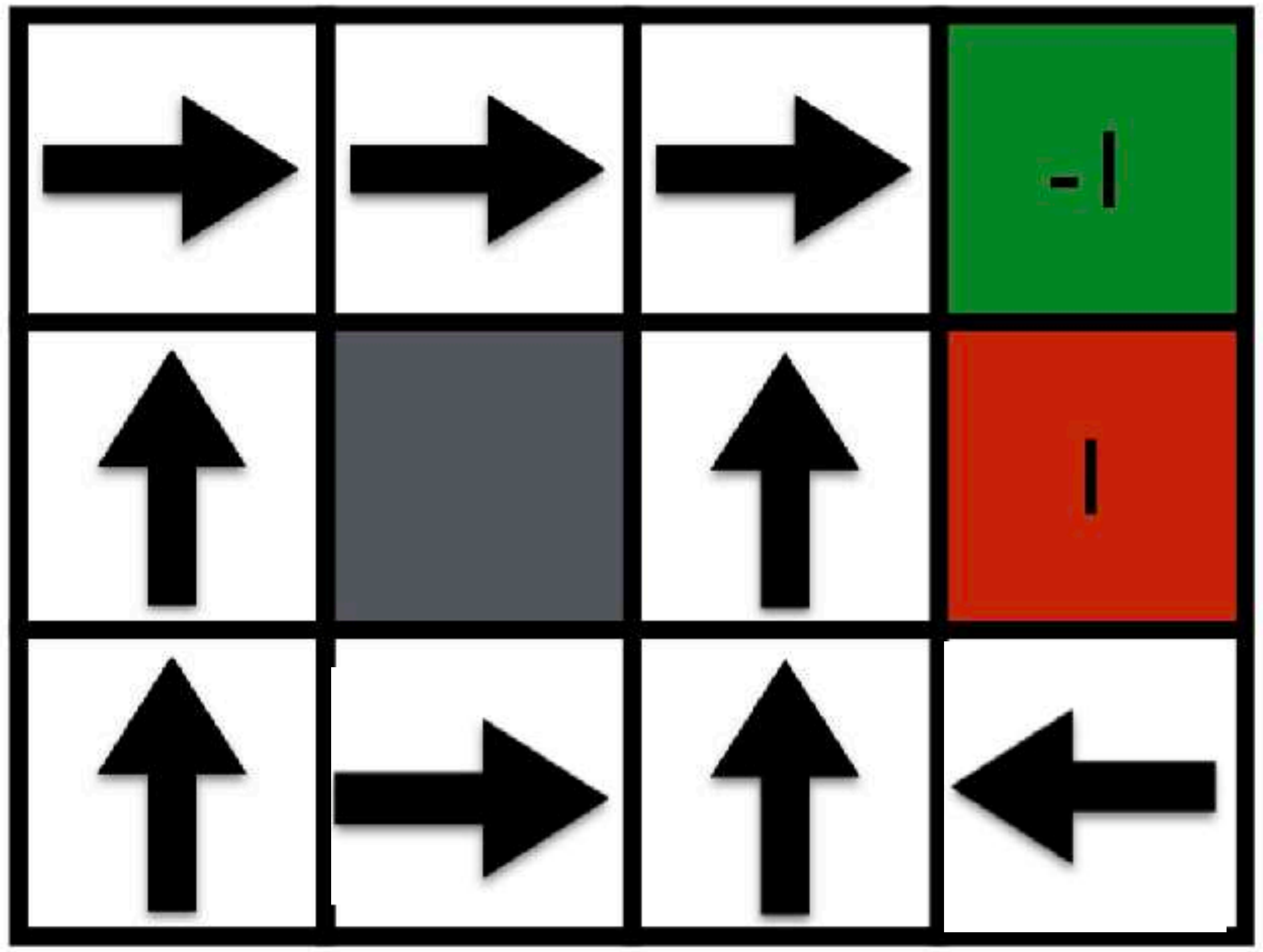
2) Update the policy

$$\mu_{k+1} = \arg\min_u g(x,u) + \alpha J_{\mu_k}(f(x,u))$$

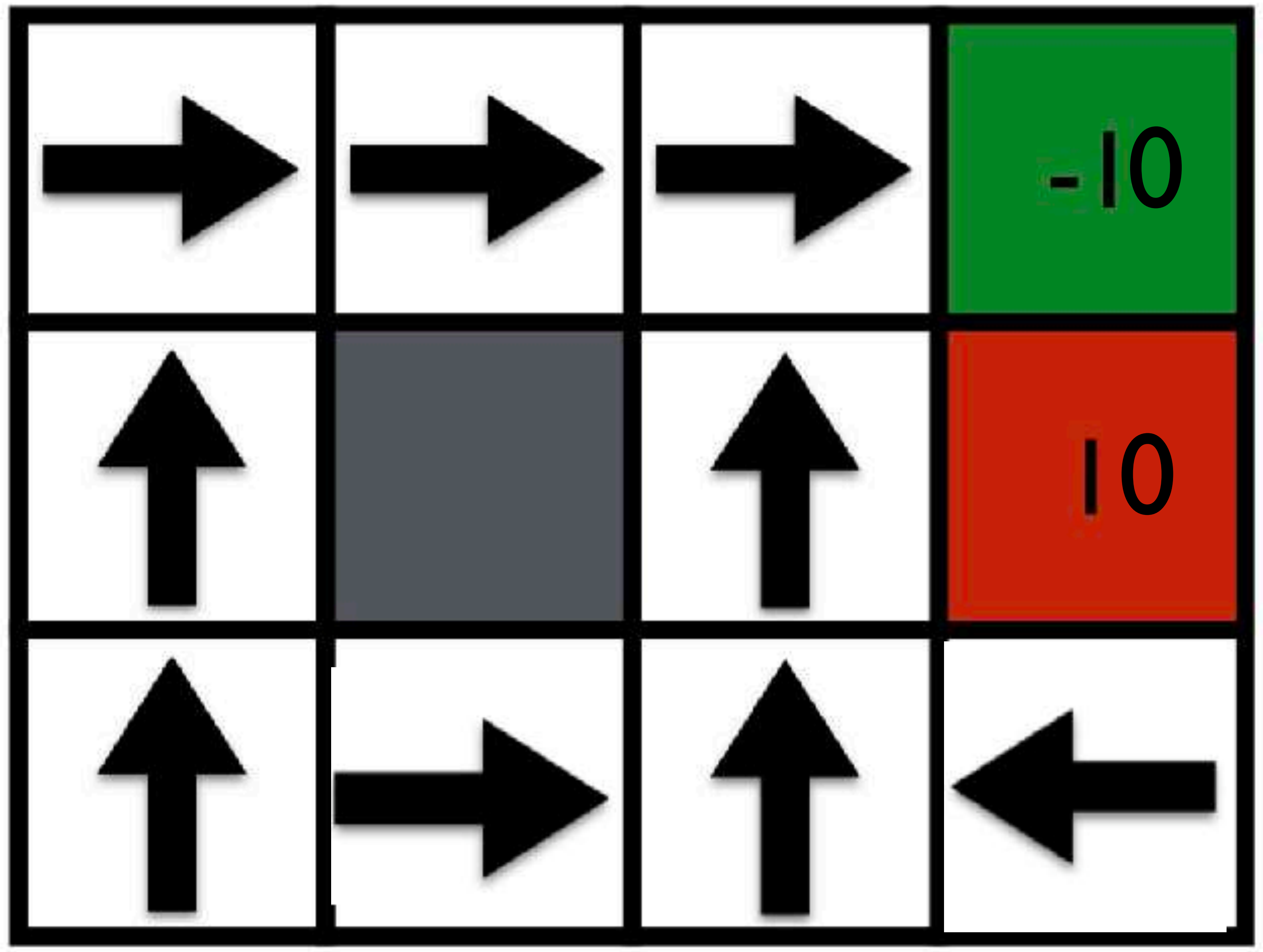


2) Update the policy

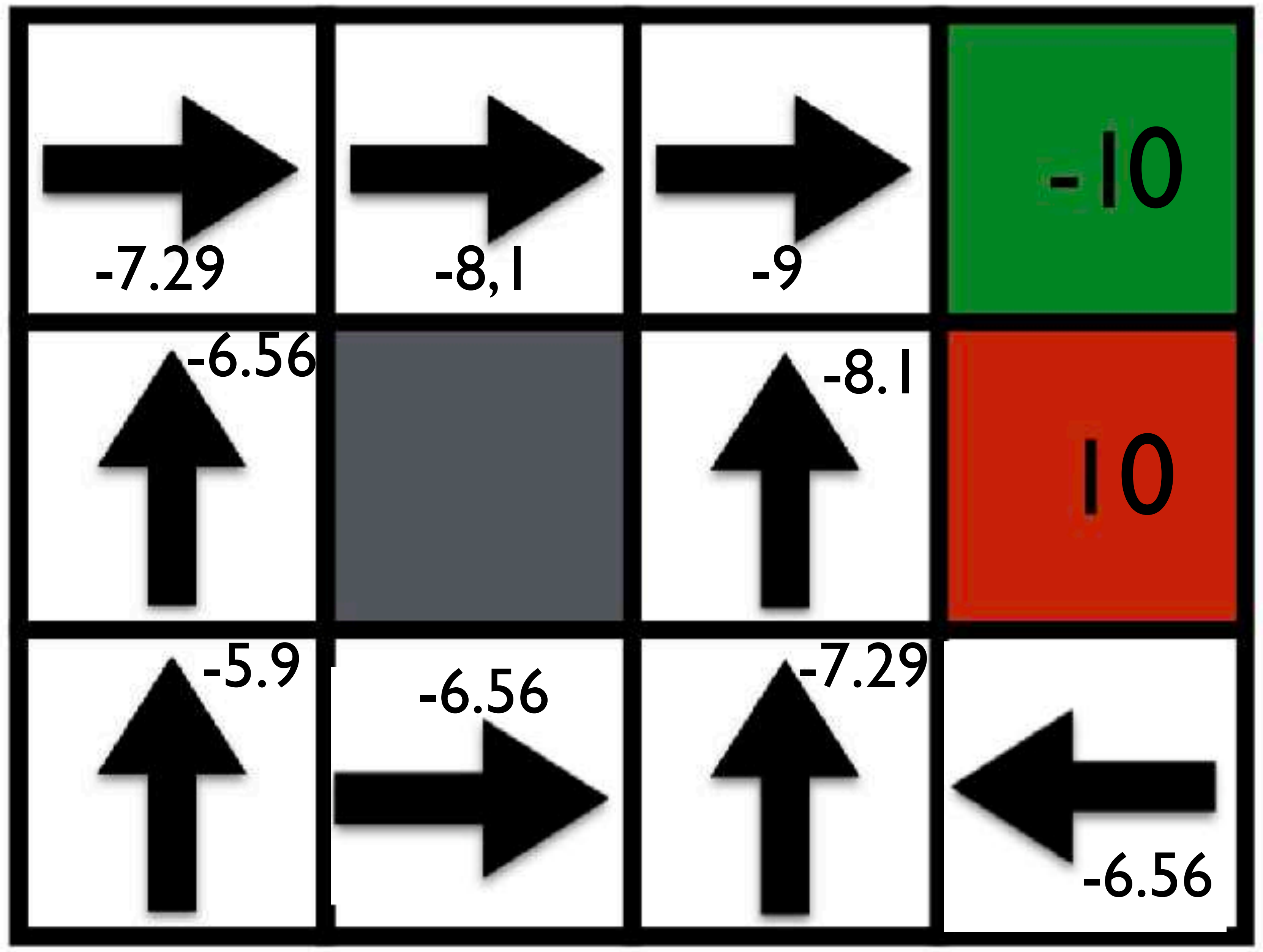
$$\mu_{k+1} = \arg\min g(x,u) + \alpha J_{\mu_k}(f(x,u))$$



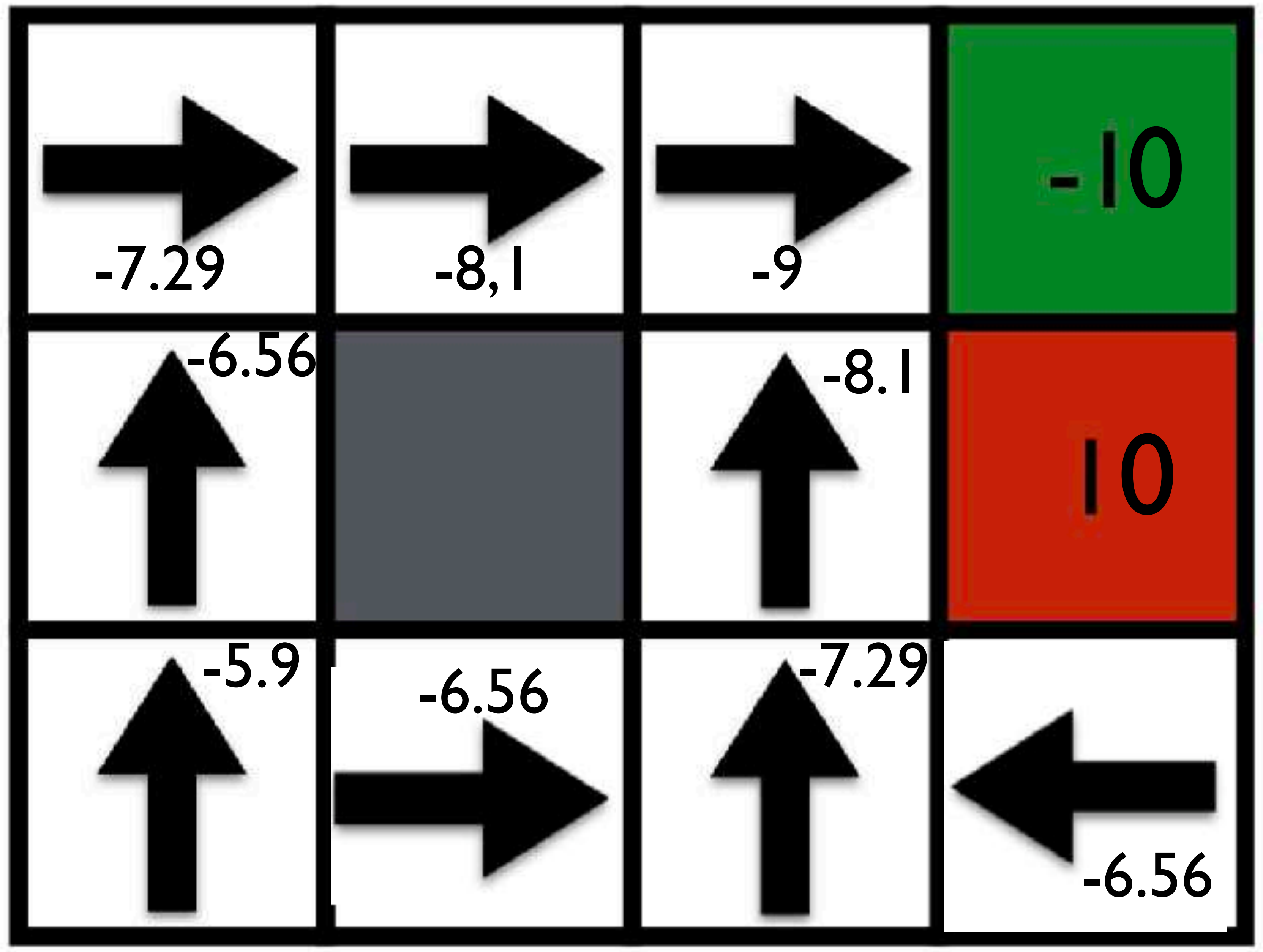
1) compute new policy cost



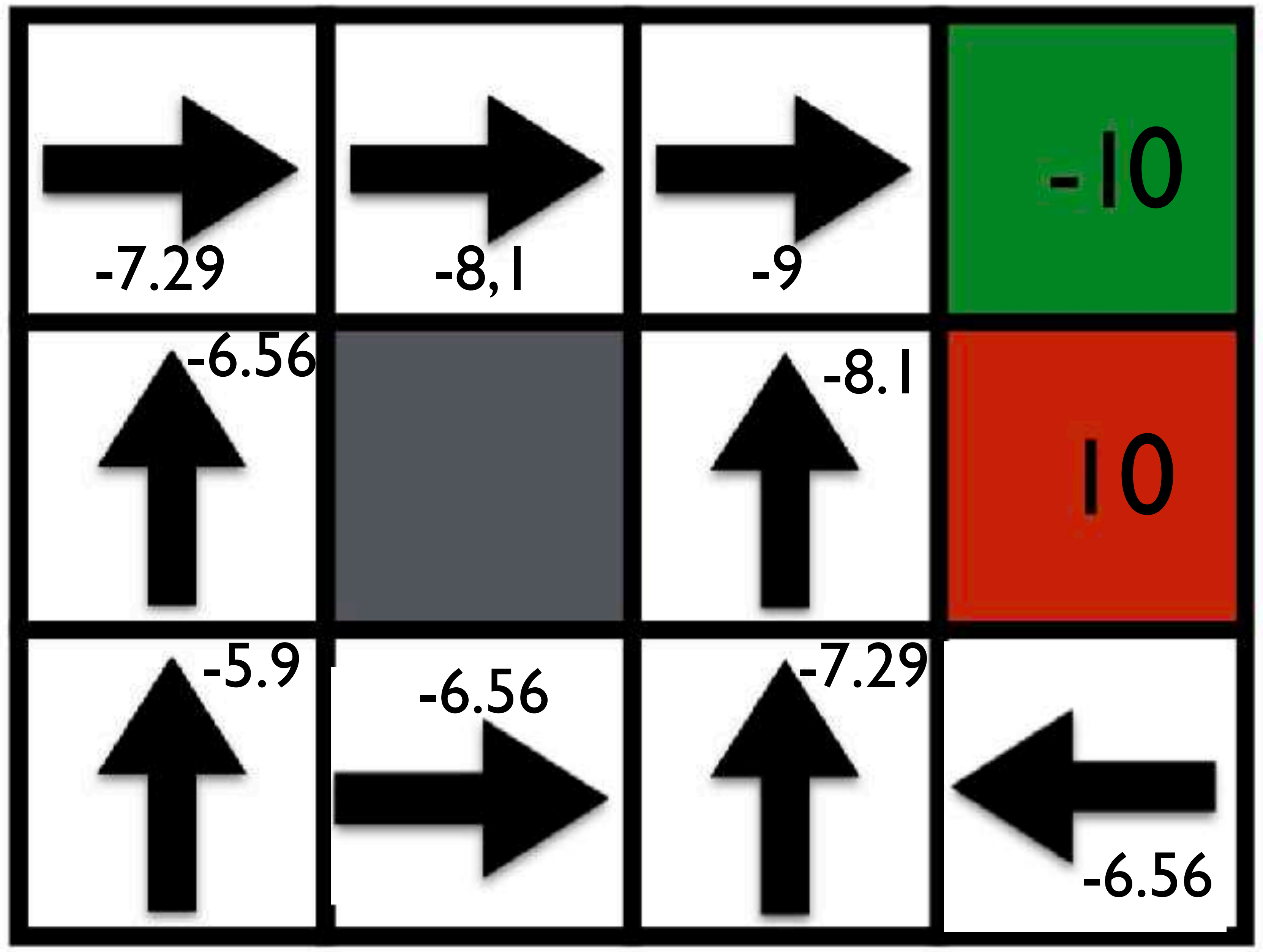
1) compute new policy cost



2) update policy $\mu_{k+1} = \arg\min_u g(x,u) + \alpha J_{\mu_k}(f(x,u))$



No change we found the optimal policy (and the optimal value function)



Policy iteration is guaranteed to converge in a finite number of steps!

Checking all the possible x is not always possible
Can we do something less “perfect” but more practical?

=> reinforcement learning!

Some notations

In the RL literature

States S_n

Control inputs are called actions A_n

Non-deterministic systems are considered

$$x_{n+1} = f(x_n, u_n, \omega_n)$$

So the optimization criteria is often

$$\min_{u_n} \mathbb{E}_{\omega_n} \left(\sum_{n=0}^{N-1} g_n(x_n, u_n) \right)$$

Evaluating a policy through sampling: TD-learning

$$J_\mu(x_t) \stackrel{?}{=} \underline{g(x_t, \mu(x_t)) + \alpha J_\mu(x_{t+1})}$$

current guess of cost for state x_t

actual return following time t

Temporal difference error (TD error)

$$\delta_t = g(x_t, \mu(x_t)) + \alpha J_\mu(x_{t+1}) - J_\mu(x_t)$$

Temporal difference learning

[Sutton 1988]
[Samuel 1959]

TD(0) learning for estimating J_μ

Input: policy to be evaluated μ

Choose a step size $\gamma \in [0, 1]$

Initialize J_μ for all states x

For each episode of length N:

Choose an initial state x_0

Loop for each step of the episode:

Do $\mu(x_t)$

Observe x_{t+1} Compute $g(x_t, \mu(x_t))$

Update $J_\mu(x_t) \leftarrow J_\mu(x_t) + \gamma\delta_t$

using $\delta_t = g(x_t, \mu(x_t)) + \alpha J_\mu(x_{t+1}) - J_\mu(x_t)$