

ROB-GY 6323
reinforcement learning and optimal
control for robotics

Lecture 7
Bellman's principle and dynamic programming

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	<u>Intro</u>	Lecture 1: introduction			
2	<u>Trajectory optimization</u>	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	Project 1	
7	<u>Policy optimization</u>	Lecture 7: Bellman's principle	HW 3		
8		Lecture 8: Value iteration / policy iteration			
9		Lecture 9: TD learning - Q-learning	HW 4		
10		Lecture 10: Deep Q learning			
11		Lecture 11: Actor-critic algorithms	HW 5		
12		Lecture 12: Learning by demonstration			
13		Lecture 13: Monte-Carlo Tree Search	Project 2		
14	Lecture 14: Beyond the class				
15	Finals week				

Homework 2 is due next week!

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

Gradient estimation via Gaussian smoothing

Gaussian smoothing:
$$f_\sigma(x) = \mathbb{E}_{u \sim \mathcal{N}(0, \sigma^2)} [f(x + u)] = \int f(x + u) \frac{e^{-\frac{u^T u}{2\sigma^2}}}{\sqrt{(2\pi)^n}} du$$

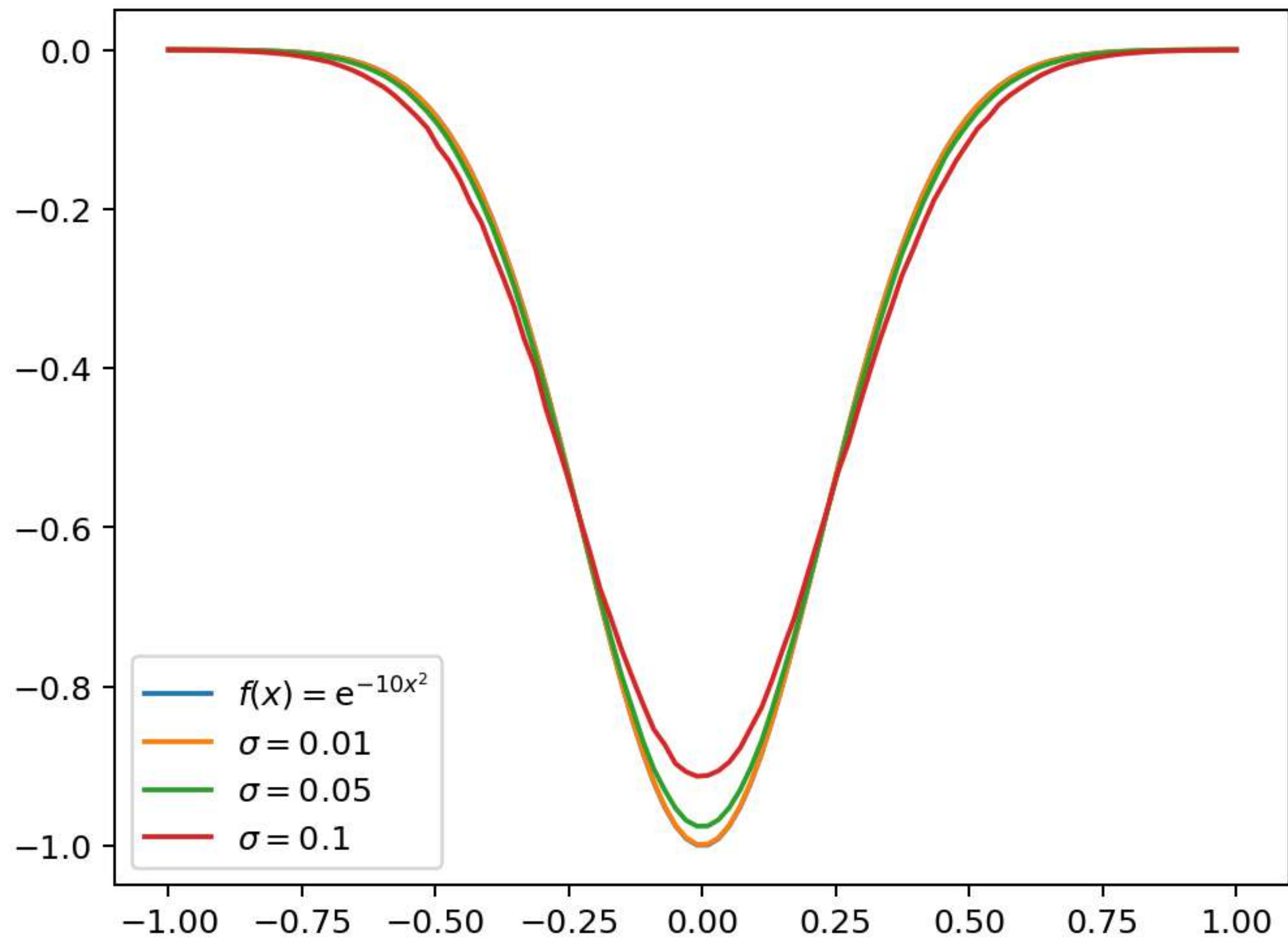
In general $f(x) \neq f_\sigma(x)$

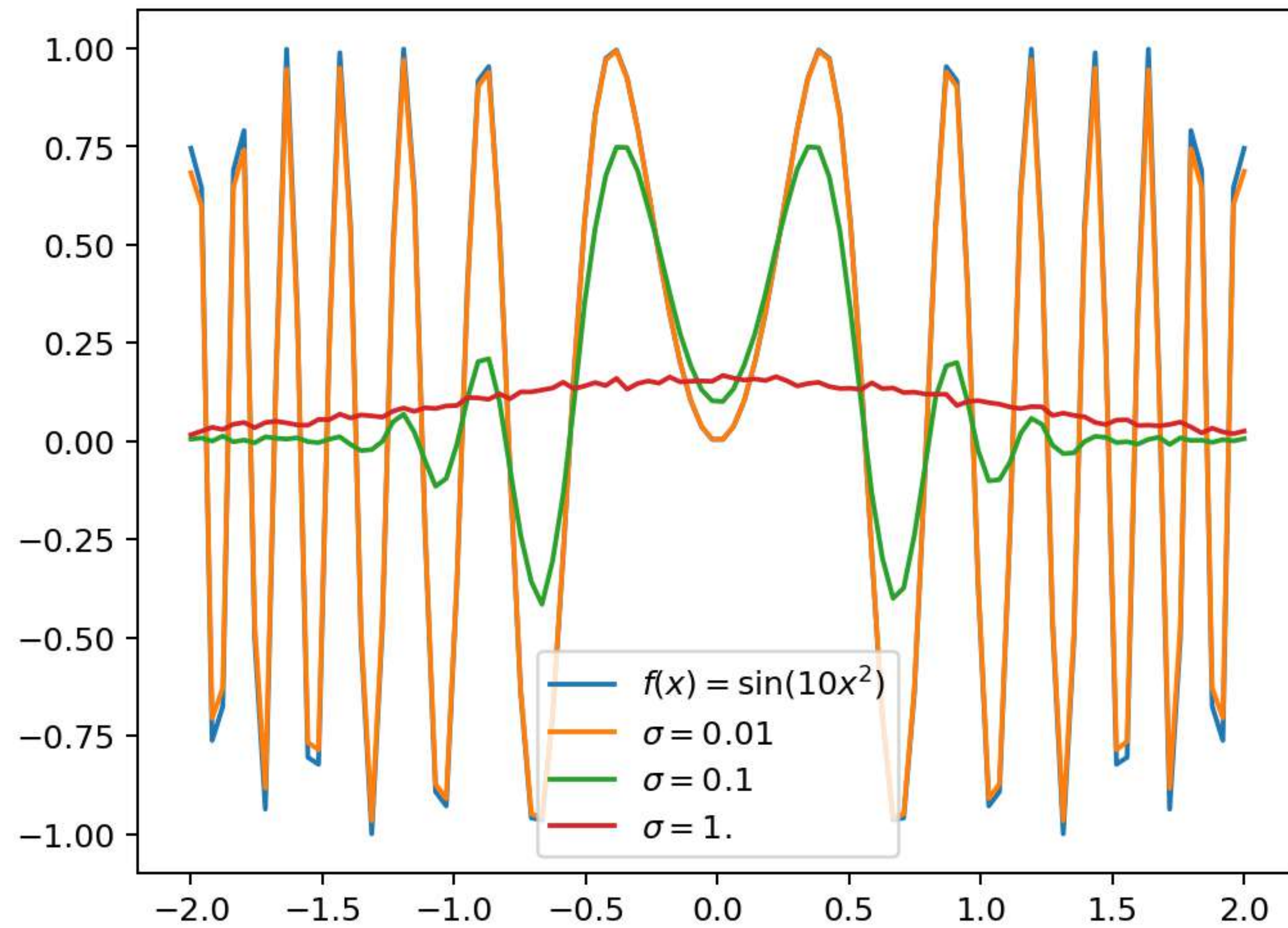
But if $f(x)$ is convex, we have by Jensen's inequality $f(x) \leq f_\sigma(x)$

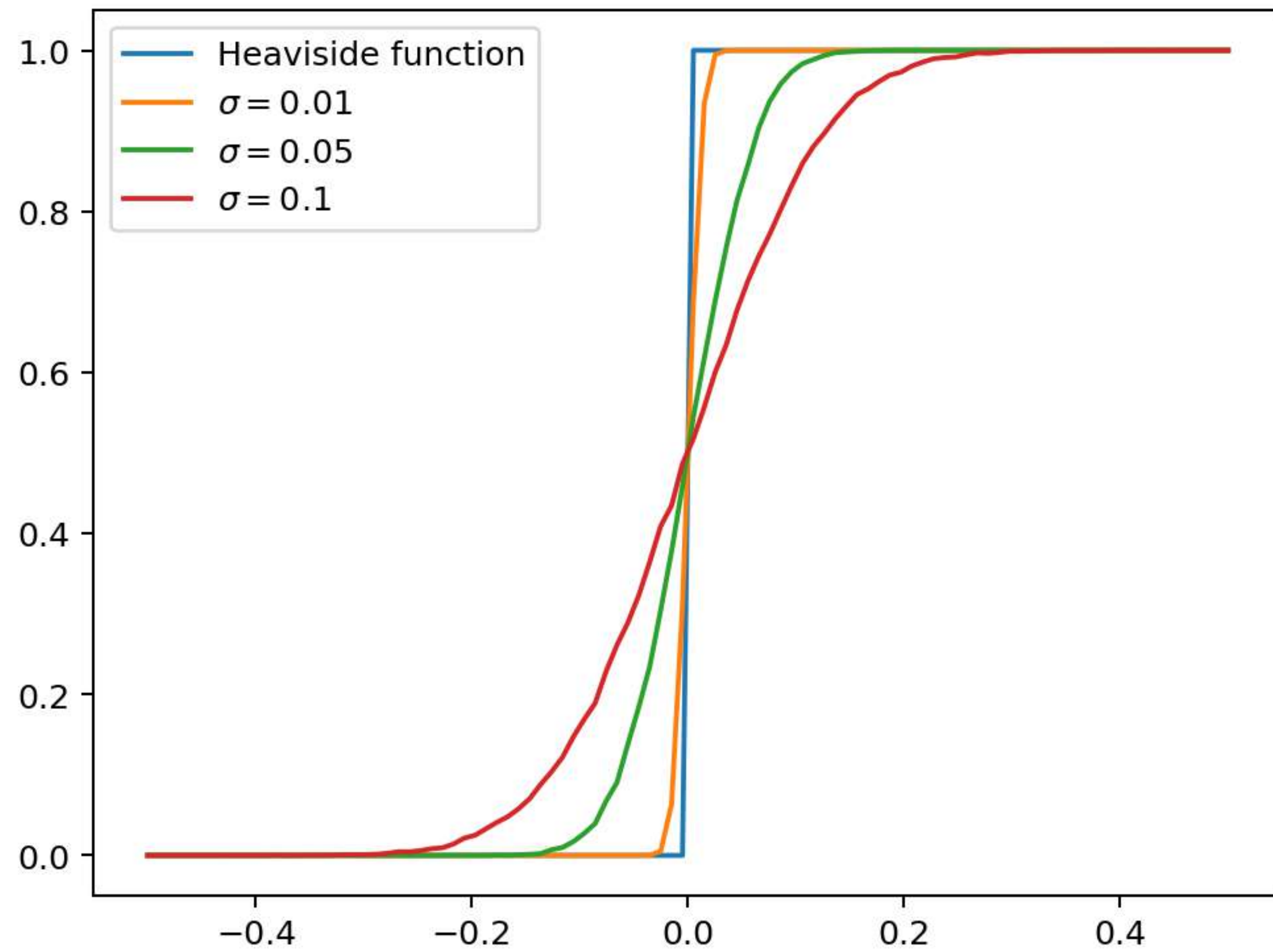
The difference between $f(x)$ and f_σ is however limited and depends on how fast $f(x)$ changes:

If $|f(x) - f(y)| \leq L_0 \|x - y\|$ then $|f_\sigma(x) - f(x)| \leq \sigma L_0 \sqrt{n}$

If $\|\nabla f(x) - \nabla f(y)\| \leq L_1 \|x - y\|$ then $|f_\sigma(x) - f(x)| \leq \frac{\sigma^2}{2} L_1 n$







Gradient estimation via Gaussian smoothing

$$\min_{x \in \mathbb{R}^n} f(x)$$

The gradient of the Gaussian smoothing approximation is

$$\nabla f_\sigma(x) = \mathbb{E}_{u \sim \mathcal{N}(0, \sigma^2)} \left[\frac{f(x + u) - f(x)}{\sigma} u \right] \simeq \frac{1}{N} \sum_{i=1}^N \frac{f(x + u_i) - f(x)}{\sigma} u_i$$

and gradient descent can be implemented by approximating $f_\sigma(x)$ via sampling

Single shooting with sampling based gradient descent

$$\min_{u_0, \dots, u_{N-1}} \sum_{n=0}^{N-1} l_n(f^n(x_0, u_0, \dots, u_{n-1}), u_n) + l_N(f^N(x_0, u_0, \dots, u_{N-1}))$$

Start with a control guess $\bar{u} = [u_0, \dots, u_{N-1}]$, then repeat until convergence

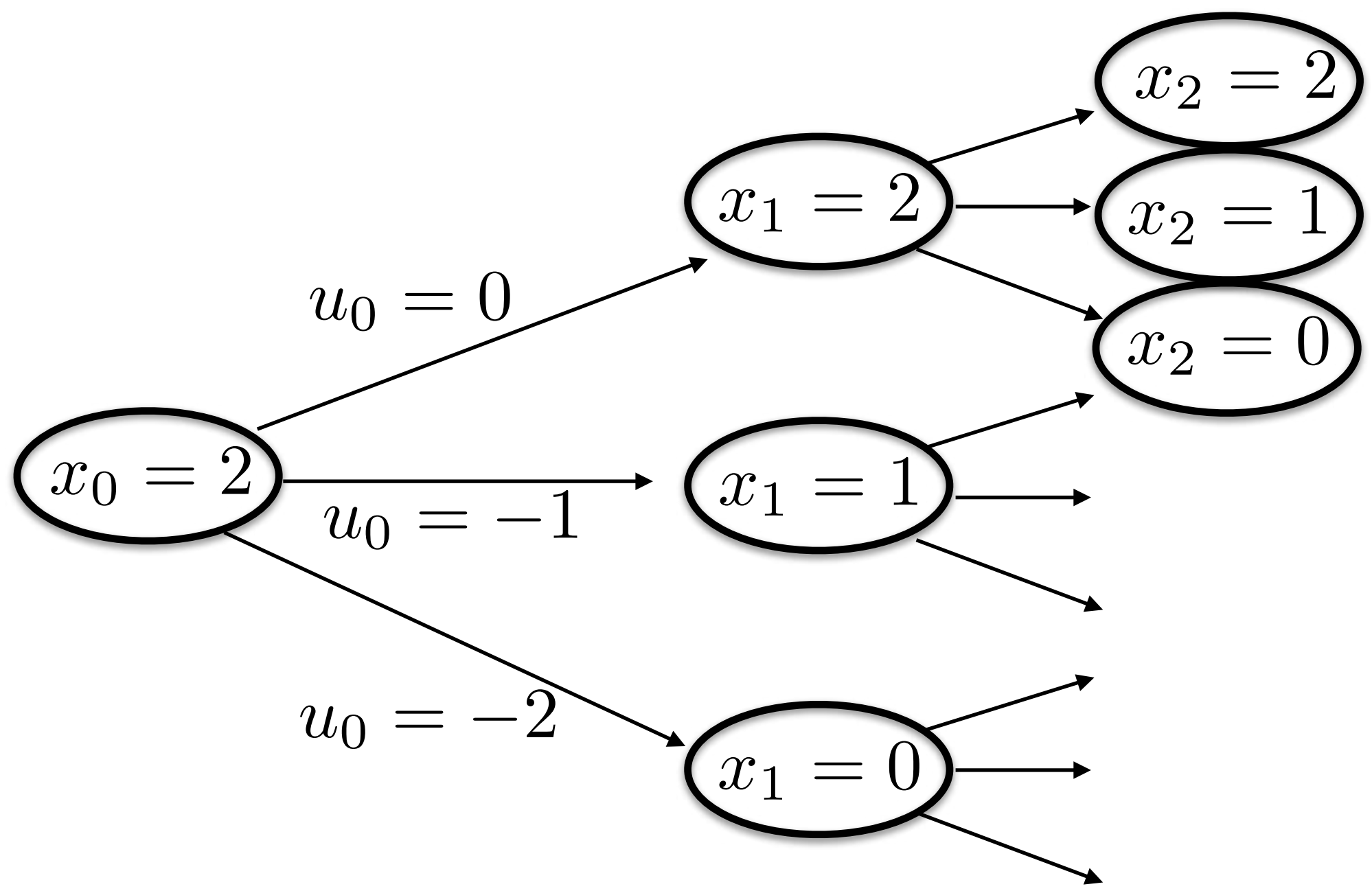
- Sample N trajectories $\bar{u} + \epsilon_n$ where $\epsilon \in \mathcal{N}(0, I)$
- Estimate the gradient of the cost $\nabla l(\bar{u}) \simeq \frac{1}{N\sigma} \sum (f(\bar{u} + \epsilon_n) - f(\bar{u})) \epsilon_n$
- Update $\bar{u} \leftarrow \bar{u} + \alpha \nabla l(\bar{u})$

Very convenient as we can replace the dynamics $f()$ by a simulator!

From trajectories to policies...

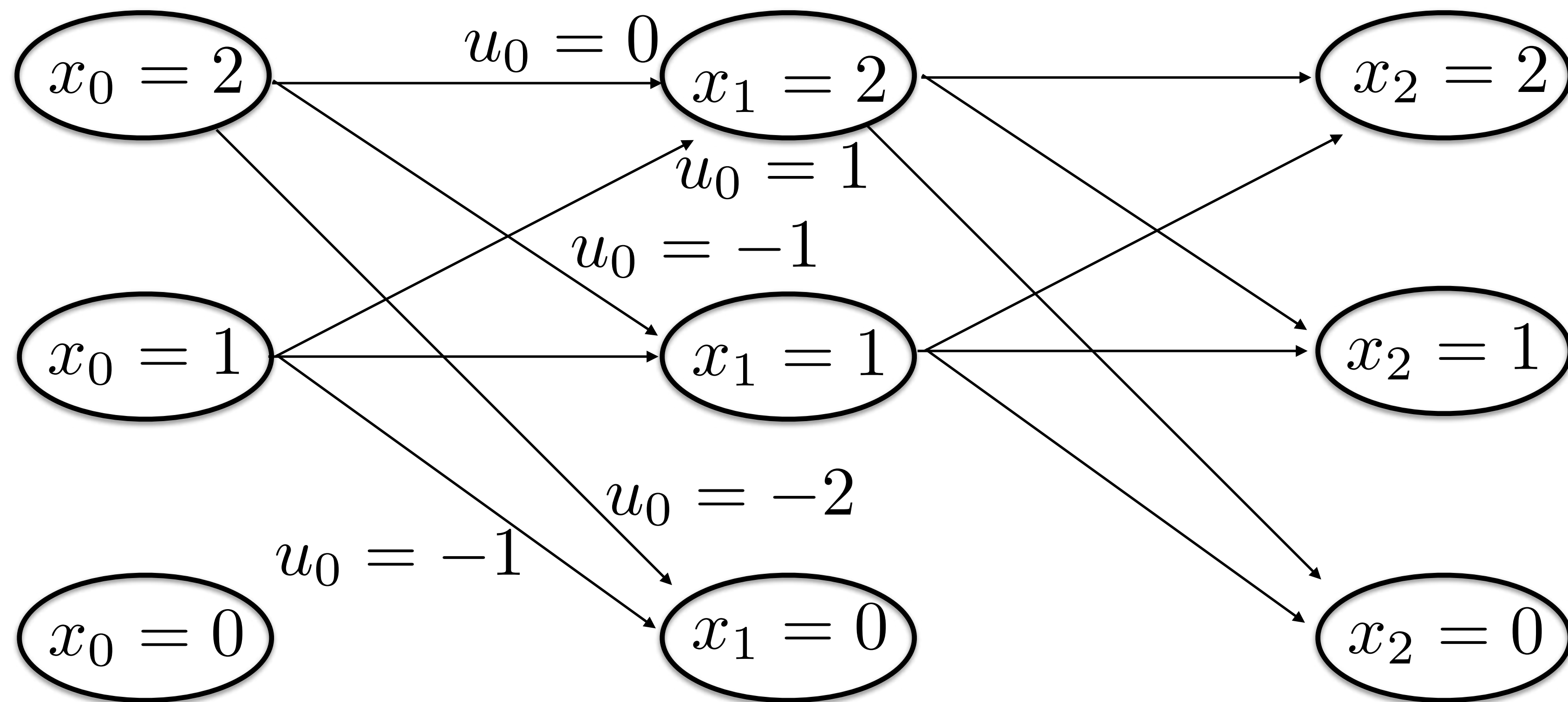
computing the optimal control for every initial state

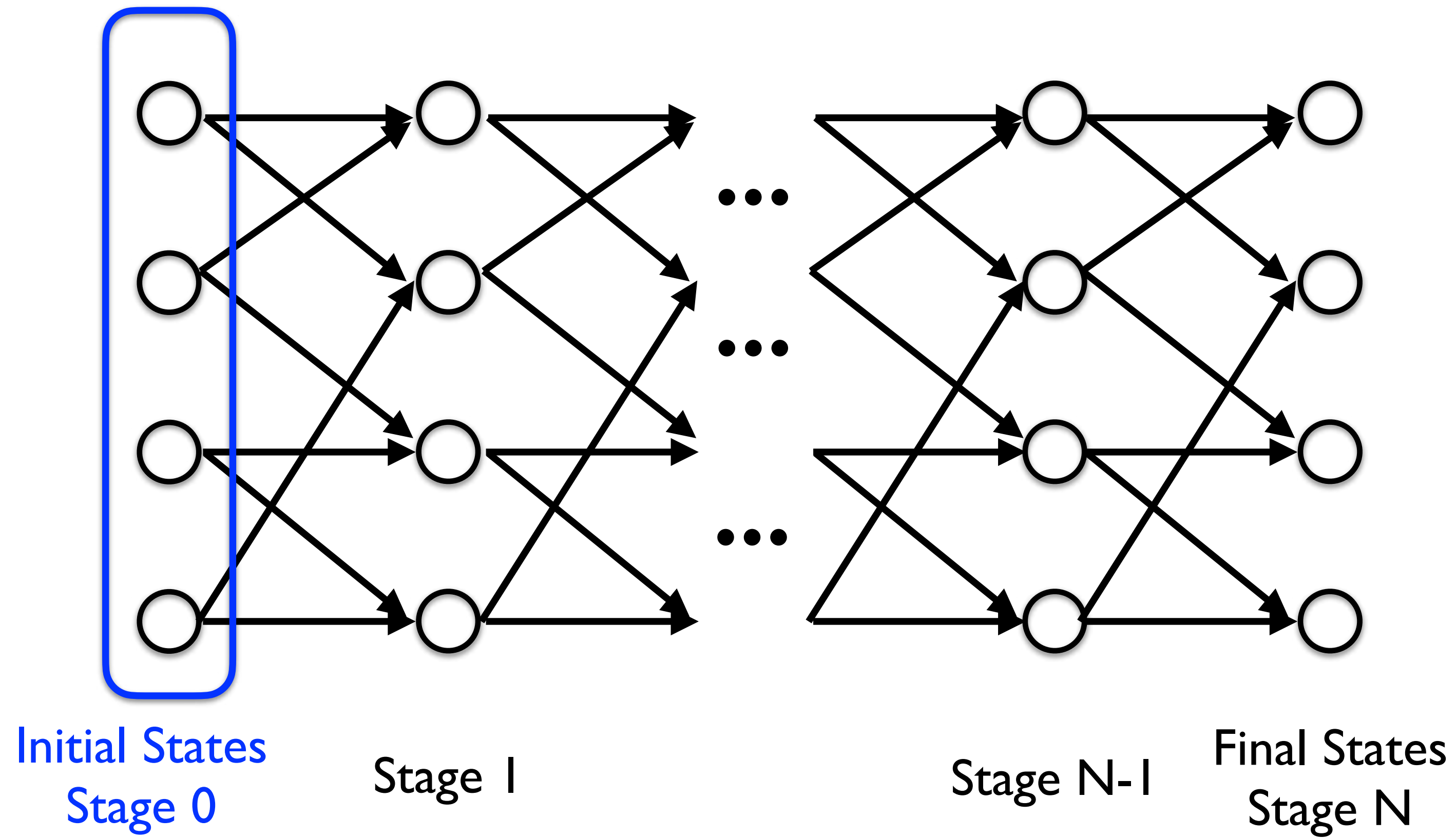
Example



$$x_0 = 1$$

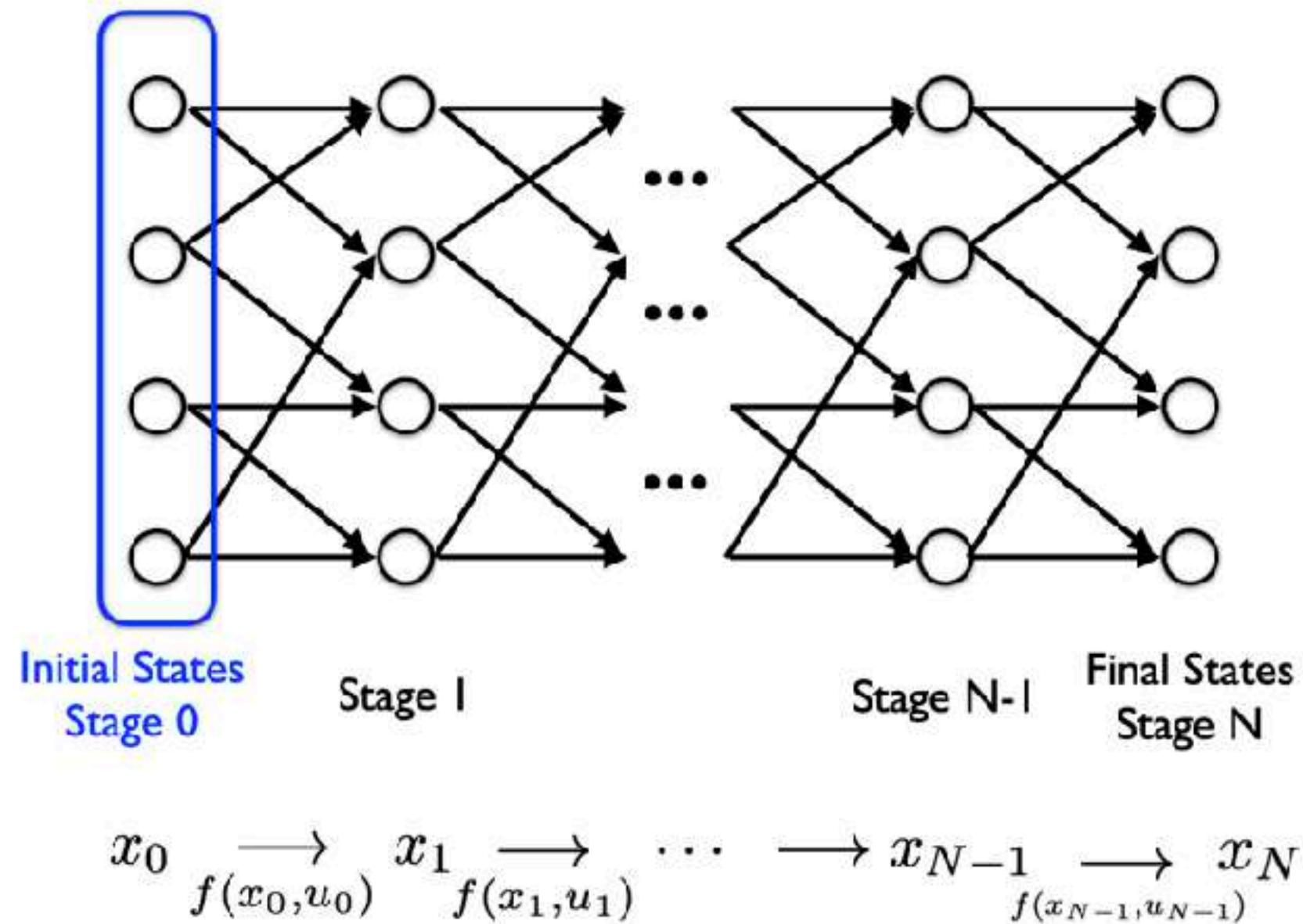
$$x_0 = 0$$





$$x_0 \xrightarrow{f(x_0, u_0)} x_1 \xrightarrow{f(x_1, u_1)} \cdots \xrightarrow{\quad} x_{N-1} \xrightarrow{f(x_{N-1}, u_{N-1})} x_N$$

Control law and control policy



A **control law** $\mu_k(x_k)$ for stage k is a function that maps every state x_k to a control action u_k , i.e. it is a function that tells us what to do at stage k .

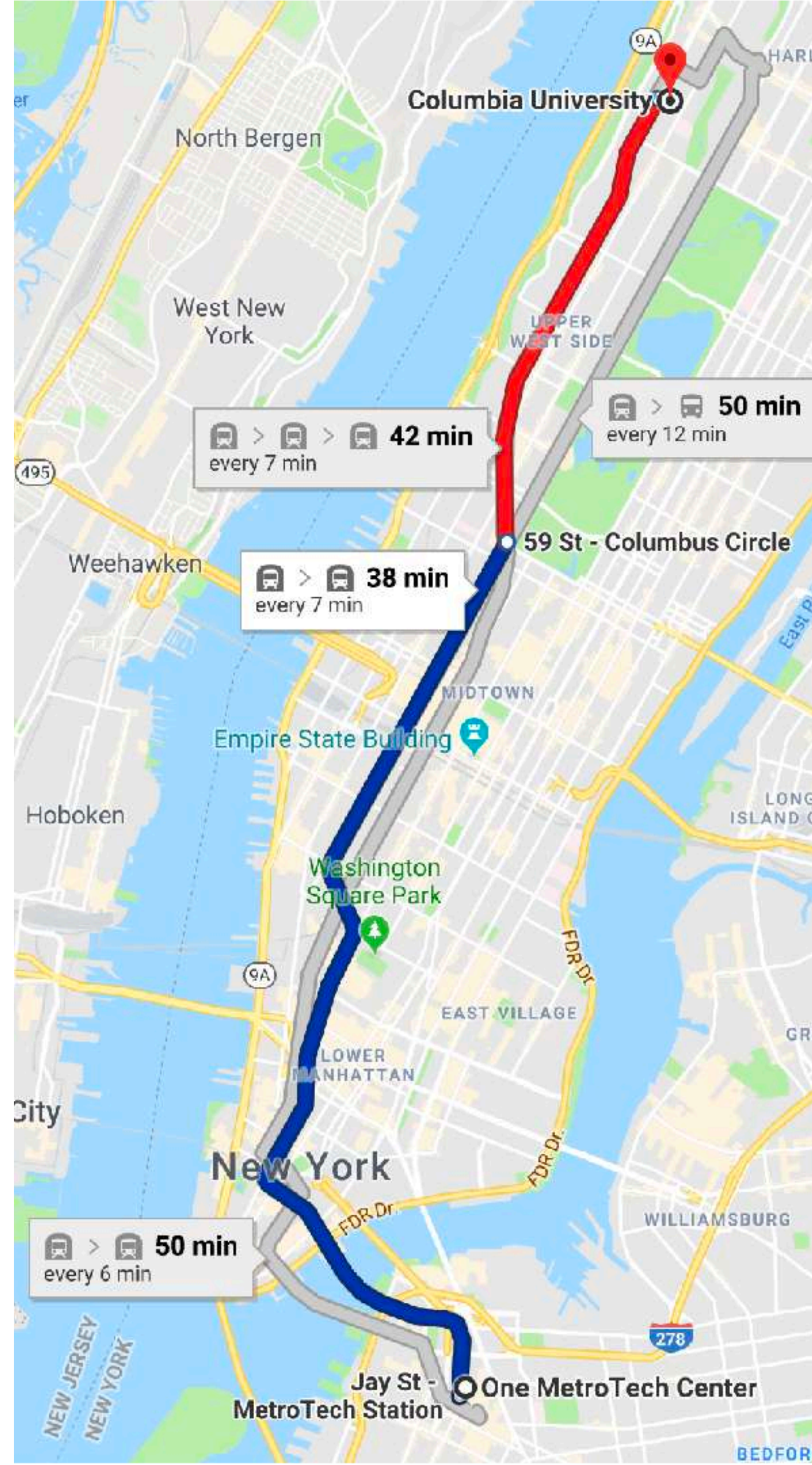
A **control policy** $\pi = \{\mu_0(x_0), \mu_1(x_1), \dots, \mu_N(x_N)\}$ is a sequence of control laws for the all the stages of the problem.

Cost to go

We call $J_n(x_n) = \min \sum_{i=n}^{N-1} g_i(x_i, u_i) + g_N(x_N)$ the **optimal cost to go**
This is the minimum cost from stage n (at state x_n) until the end

We will often write μ_k^* the optimal control law at stage k and π^* the optimal control policy. $J^*(x_0)$ will be the optimal cost.

Bellman's principle of optimality



If the fastest path from Tandon to Columbia is with the blue line with a change to the red line at Columbus Circle, then the fastest way from Columbus Circle to Columbia is with the red line as well

Subpath of optimal paths are also optimal for then own subproblem

Bellman's principle of optimality

Bellman's principle of optimality

Let $\{u_0^*, u_1^*, \dots, u_{N-1}^*\}$ be an optimal control sequence for the original optimal control problem and assume that when we are using the sequence of control, a given state x_k occurs at time k . Consider the subproblem where we start at x_k at time k and wish to minimize the *cost-to-go* from time k to time N

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

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

Dynamic programming: start from the end and compute optimal policies of sub-problems

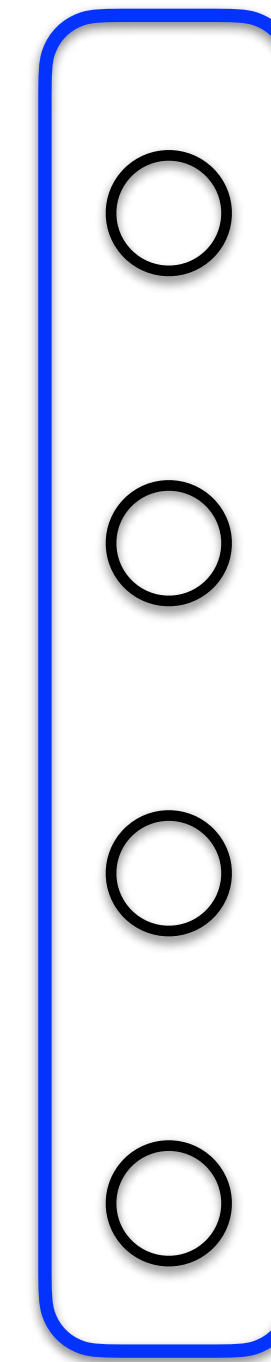
Dynamic programming

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:

$$J_N(x_N) = g_N(x_N)$$

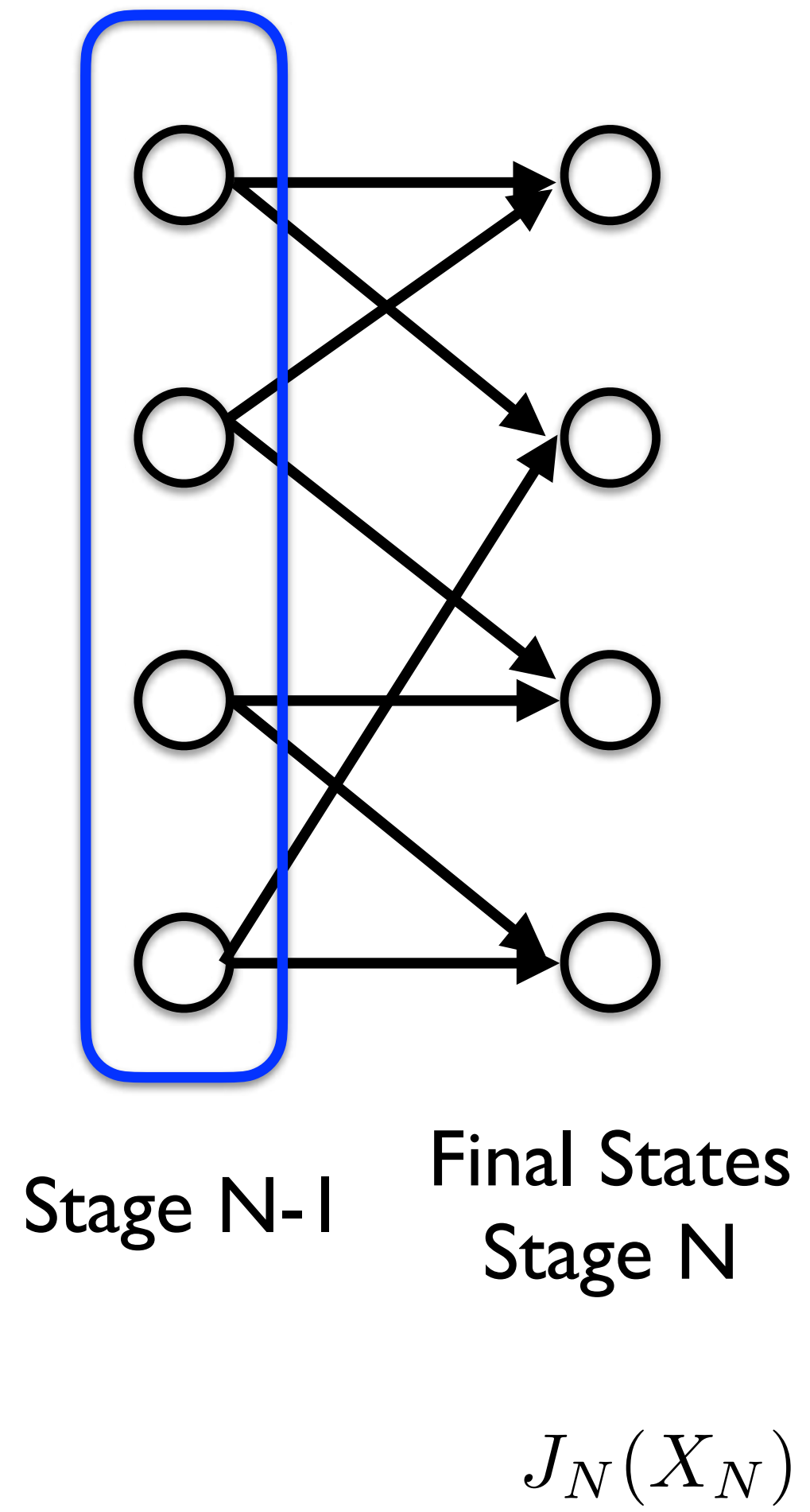
$$J_k(x_k) = \min_{u_k} g_k(x_k, u_k) + J_{k+1}(f(x_k, u_k))$$

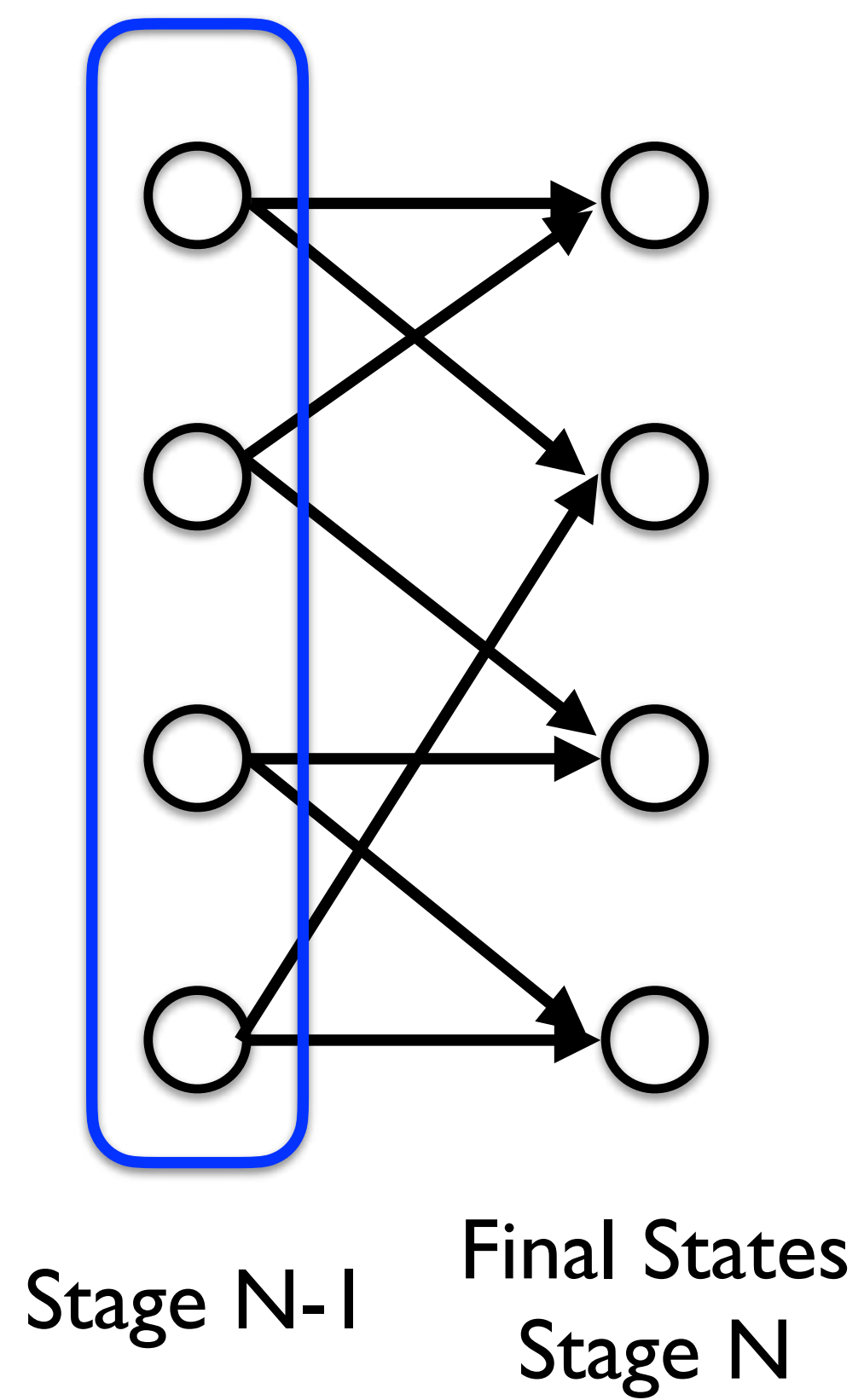
Furthermore, if the control laws $u_k^* = \mu_k^*(x_k)$ minimize the cost-to-go for each x_k and k , the policy $\pi^* = \{\mu_0^*, \dots, \mu_{N-1}^*\}$ is optimal.



Final States
Stage N

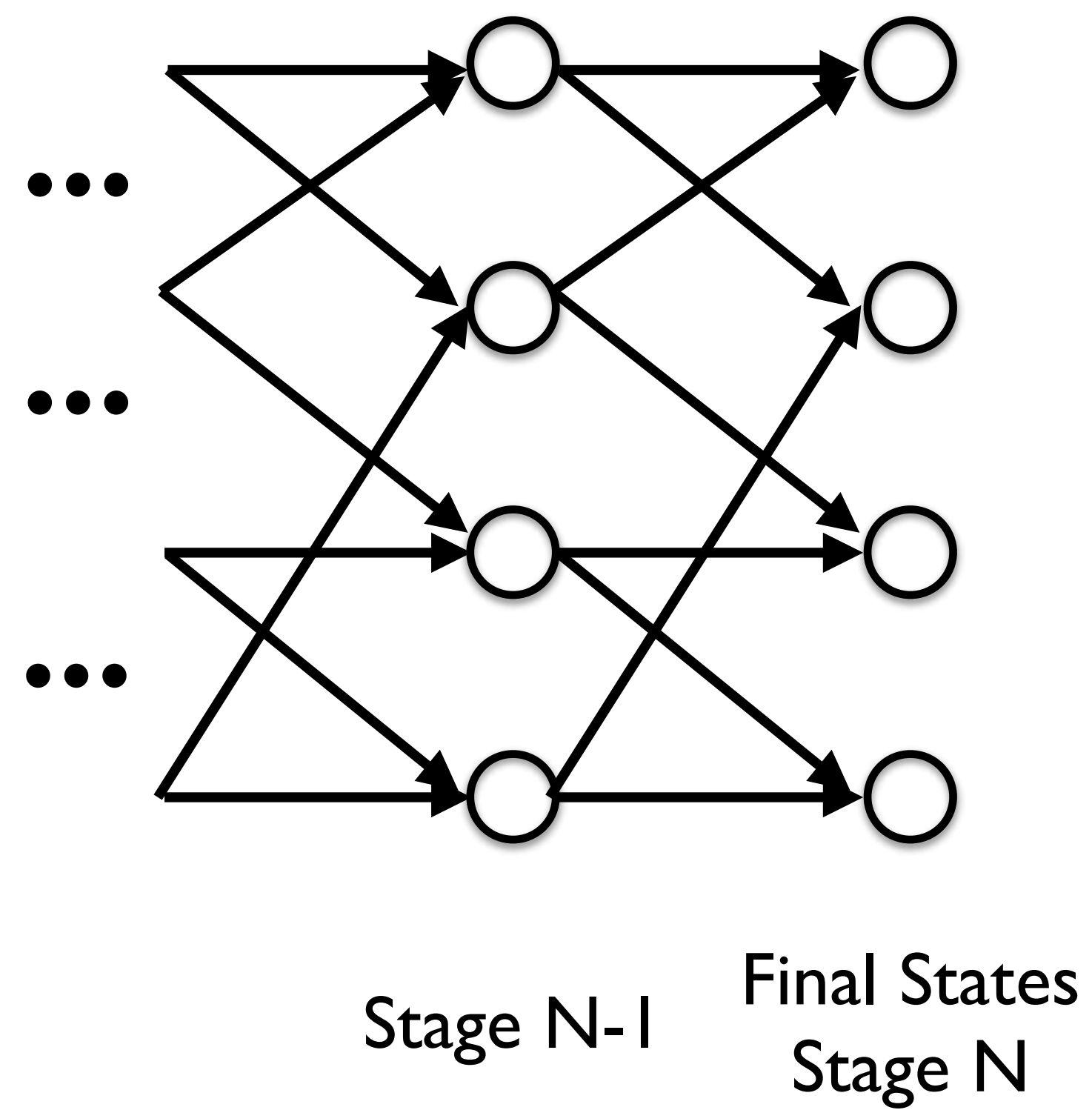
$$J_N(X_N)$$





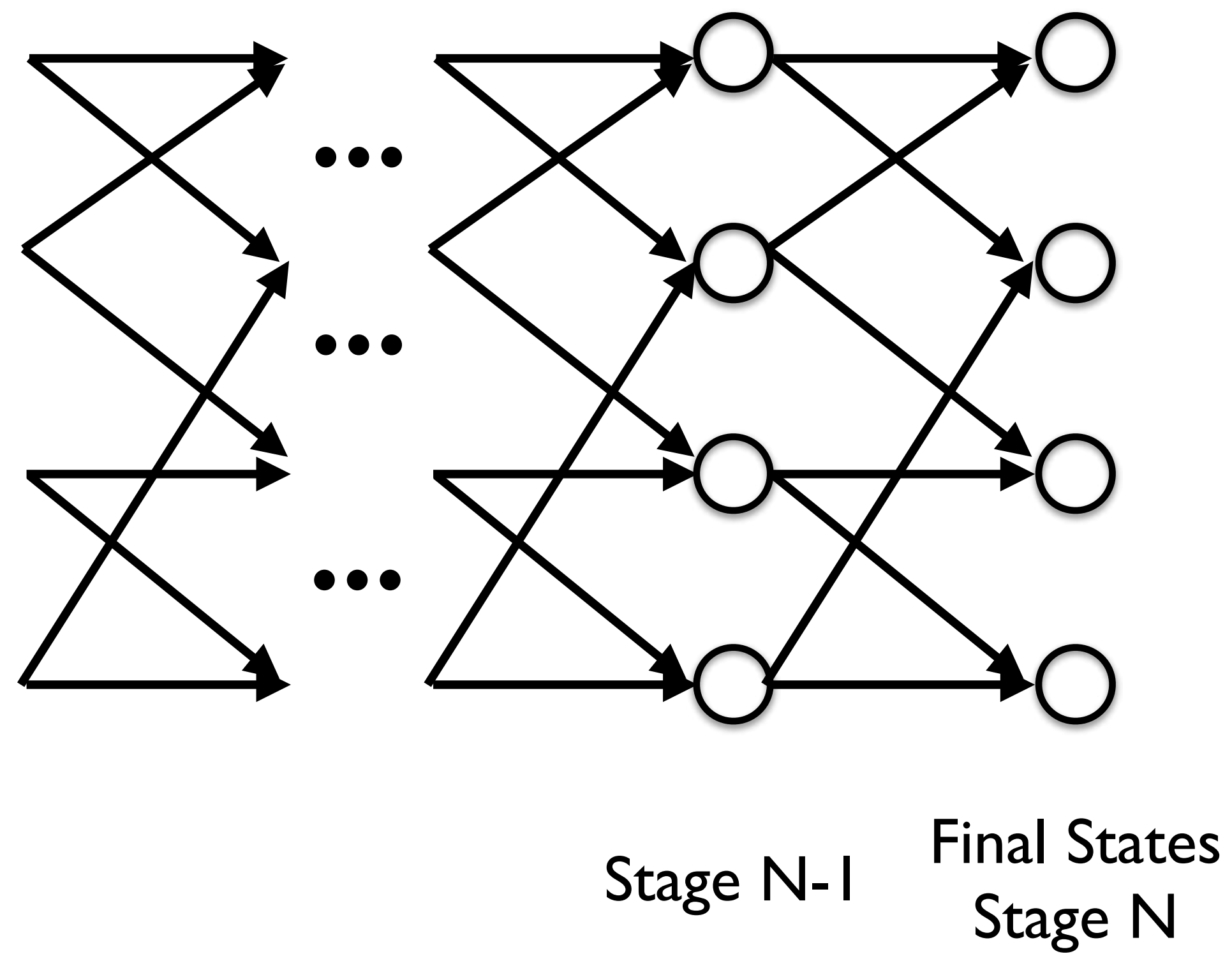
$$J_{N-1}(x_{N-1}) \longleftarrow J_N(X_N)$$

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



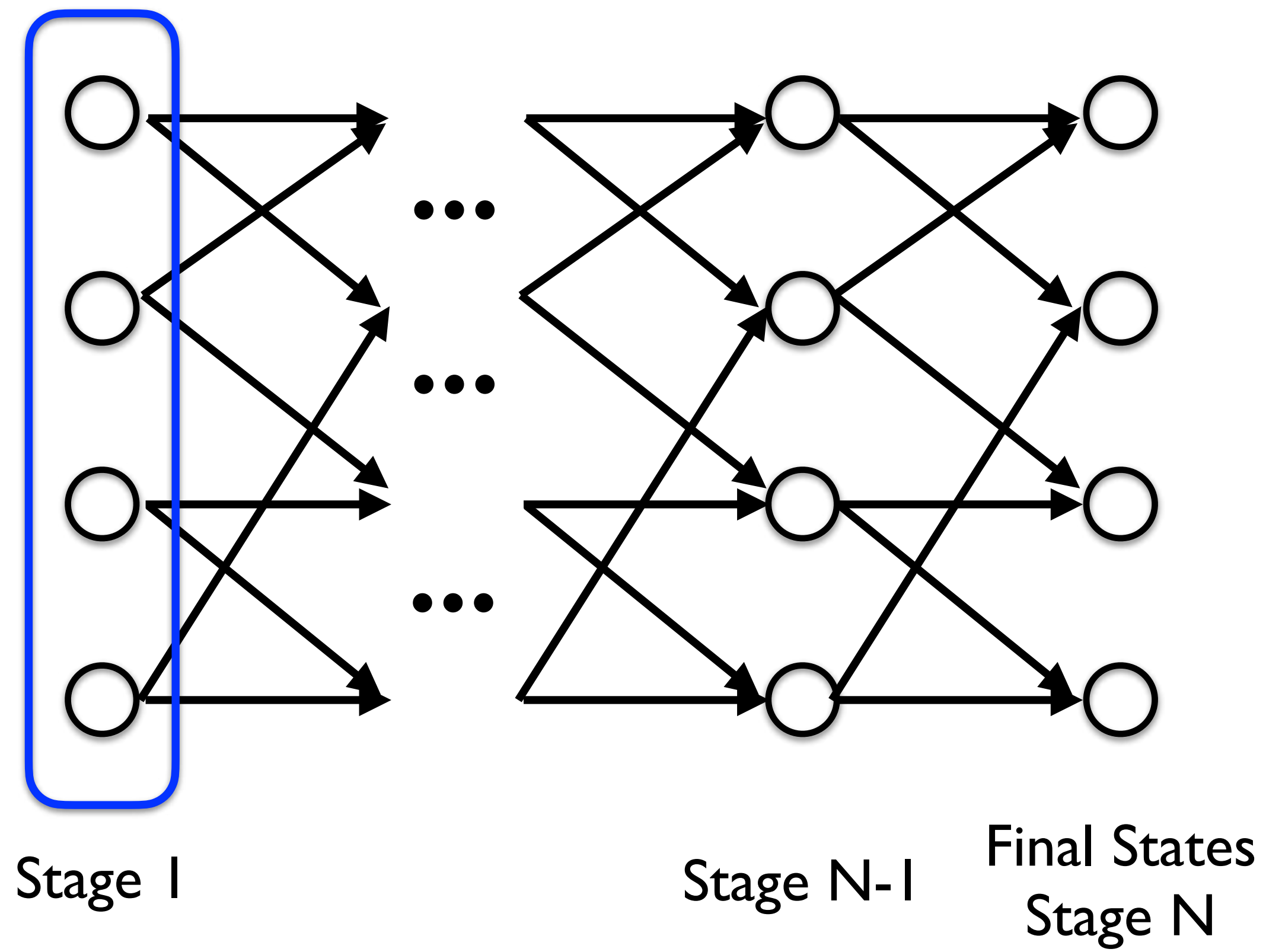
$$\dots \longleftarrow J_{N-1}(x_{N-1}) \longleftarrow J_N(X_N)$$

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



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

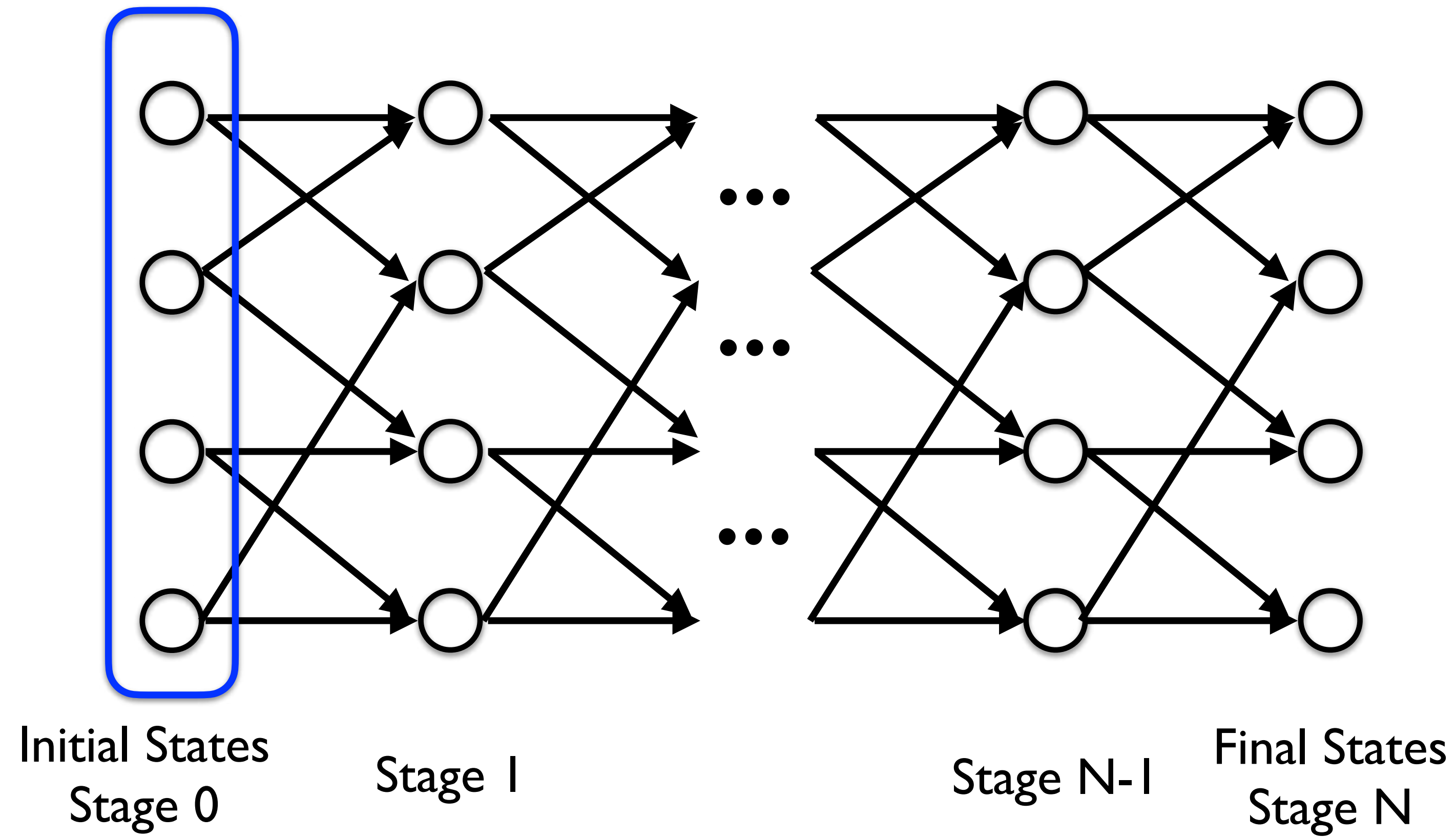
$$\mu_{N-1}(x_{N-1})$$



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

$$\mu_1(x_1)$$

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

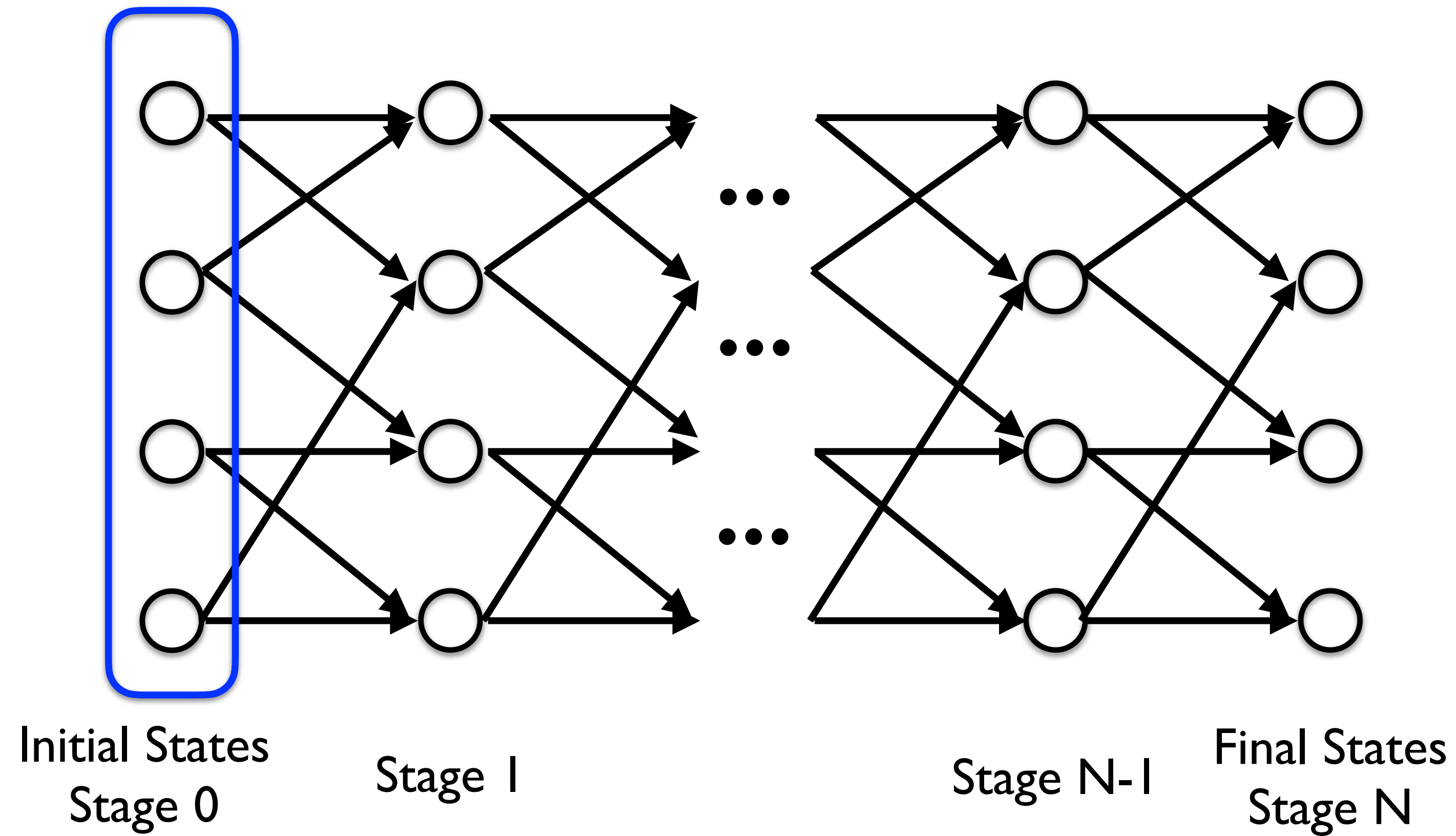


$$J_0(x_0) \longleftarrow J_1(x_1) \longleftarrow \cdots \longleftarrow J_{N-1}(x_{N-1}) \longleftarrow J_N(X_N)$$

$$\mu_0(x_0)$$

$$\mu_1(x_1)$$

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



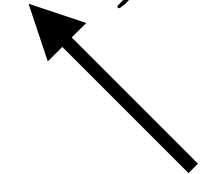
$$J_0(x_0)$$

Optimal cost for each initial state

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

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)

Markov property knowledge of state n is sufficient to predict $n+1$
 \Rightarrow there is no need to remember previous states or actions

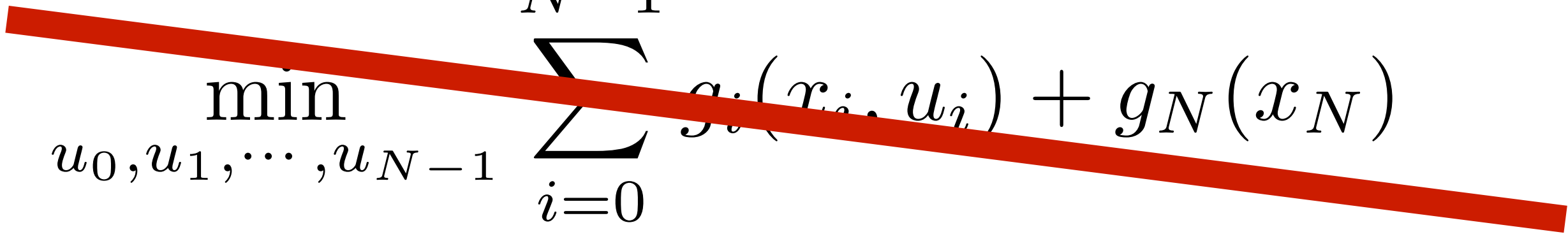
Note that \mathbf{x}_n is now a random variable.
It is a usual vector if the noise $\boldsymbol{\omega}_n$ is 0.

Finite-horizon optimal control (non deterministic)

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

Uncertainty




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

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

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

We are selling coffee and need to order a certain quantity of coffee to meet the demand while we don't want to keep too much stock because it is costly to store

How can we plan coffee orders for the next N weeks?

x_n Quantity of coffee at week n

u_n Amount of ordered coffee at week n

ω_n Client demand \Rightarrow random variable with known distribution

x_n Quantity of coffee at week n

u_n Amount of ordered coffee at week n

ω_n Client demand \Rightarrow random variable with known distribution

non-negative inventory

$$x_{n+1} = \max(0, x_n + u_k - \omega_n)$$

maximum storage capacity is 2

$$x_n + u_n \leq 2$$

storage cost for period n

$$(x_n + u_n - \omega_n)^2$$

ordering cost is 1 per unit

$$u_n$$

total cost per stage

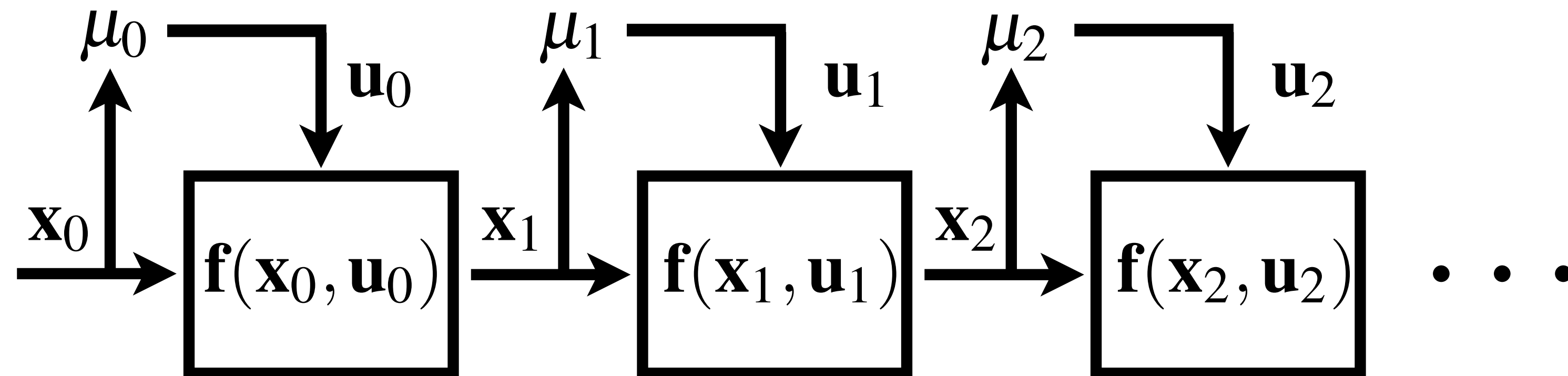
$$g_i(x_i, u_i, \omega_i) = (x_i + u_i - \omega_i)^2 + u_i$$

final cost is 0

$$g_N(x_N) = 0$$

$$\mathbf{w}_k \neq 0$$

Decisions can only be made when stage k is attained. We need to measure the state at stage k and then use it to compute the optimal control. It is a closed loop system



Let $\pi^* = \{u_0^*, u_1^*, \dots, u_{N-1}^*\}$ 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 $\{u_k^*, \dots, u_{N-1}^*\}$ is optimal for this subproblem.

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:

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

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 $\pi^* = \{\mu_0^*, \dots, \mu_{N-1}^*\}$ is optimal.

Plan for N=3 weeks to minimize cost

Initial stock is $x_0 = 0$

Demand distribution $p(\omega_k = 0) = 0.1$ $p(\omega_k = 1) = 0.7$ $p(\omega_k = 2) = 0.2$

Stock	Stage 0 Cost-to-go	Stage 0 Optimal stock to purchase	Stage 1 Cost-to-go	Stage 1 Optimal stock to purchase	Stage 2 Cost-to-go	Stage 2 Optimal stock to purchase
0	3.7	1	2.5	1	1.3	1
1	2.7	0	1.5	0	0.3	0
2	2.818	0	1.68	0	1.1	0

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

Dynamic Programming

Dynamic programming is the algorithm that finds the global optimal solution to discrete time optimal control problems.

It was “invented” by Richard Bellman in 1953.

You already know dynamic programming:

Dijkstra's shortest path algorithm, Viterbi algorithm, etc... can be viewed as special cases of DP.

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

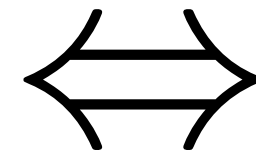
Linear systems with quadratic costs (revisited)

Linear systems with quadratic costs (revisited)

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

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

$$x_0 = x_{init}$$



$$\min_{x_n, u_n} \frac{1}{2} \begin{pmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \end{pmatrix}^T \begin{bmatrix} Q & 0 & 0 & 0 & \cdots \\ 0 & R & 0 & 0 & \cdots \\ 0 & 0 & Q & 0 & \cdots \\ 0 & 0 & 0 & R & \cdots \\ 0 & 0 & 0 & 0 & \ddots \end{bmatrix} \begin{pmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \end{pmatrix}$$

subject to

$$\begin{bmatrix} I & 0 & 0 & 0 & 0 & 0 & \cdots \\ A & B & -I & 0 & 0 & 0 & \cdots \\ 0 & 0 & A & B & -I & 0 & \cdots \\ 0 & 0 & 0 & 0 & A & B & \cdots \end{bmatrix} \begin{pmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \end{pmatrix} = \begin{pmatrix} x_{init} \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

Linear systems with quadratic costs (revisited)

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

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

$$x_0 = x_{init}$$

Efficient algorithm to solve the KKT system:

1 Compute backward (from N to 0):

$$P_N = Q$$

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

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

2 Compute forward (from 0 to N) starting with $x_0 = x_{init}$

$$u_n = -K_n x_n$$

$$x_{n+1} = A x_n + B u_n$$

K_n are called "feedback gains"

The number of multiplications needed to re-solve the KKT system without taking into account 0s will grow like N^3 while the efficient algorithm as a number of operations that grow like N . This is much better!

LQ problems using Bellman's principle

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

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

$$x_0 = x_{init}$$

Linear - Quadratic Regulator (LQR)

Problems with linear dynamics and quadratic costs can be solved explicitly!

$$\min \sum_{i=0}^{N-1} (x_i^T Q_i x_i + u_i^T R_i u_i) + x_N^T Q_N x_N \quad \text{Quadratic cost}$$

Subject to $x_{n+1} = A_n x_n + B_n u_n$ (Time-varying) linear dynamics

where $x_n \in \mathbb{R}^s$ and $u_n \in \mathbb{R}^c$ and A_n, B_n, Q_n and R_n are matrices of proper dimensions. We assume also that $Q_n \geq 0$ and $R_n > 0$

Linear - Quadratic Regulator (LQR)

$$P_N = Q_N$$

For $N-1$ to 0 do the backward recursion

$$K_n = -(B_n^T P_{n+1} B_n + R_n)^{-1} B_n^T P_{n+1} A_n$$

$$P_n = Q_n + A_n^T P_{n+1} A_n + A_n^T P_{n+1} B_n K_n$$

Discrete-time
Riccati equation

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

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

The policy is a linear feedback controller with gain K_n

Linear - Quadratic Regulator (LQR)

=> the non-deterministic case

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

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 $\mu_n^*(x_n) = K_n x_n$

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 optimal control is the same as in the deterministic case

=> The actual cost differs depending on the disturbance

Infinite horizon problems

When controlling a robot, we might want a control policy that is always valid
i.e. we want that $N = \infty$

This is what we call infinite horizon optimal control

The terminal cost does not make sense anymore and we look at

$$\min \sum_{i=0}^{\infty} g_i(x_i, u_i)$$

In general this sum 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

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

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

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 $n \rightarrow -\infty$

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

Example

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

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

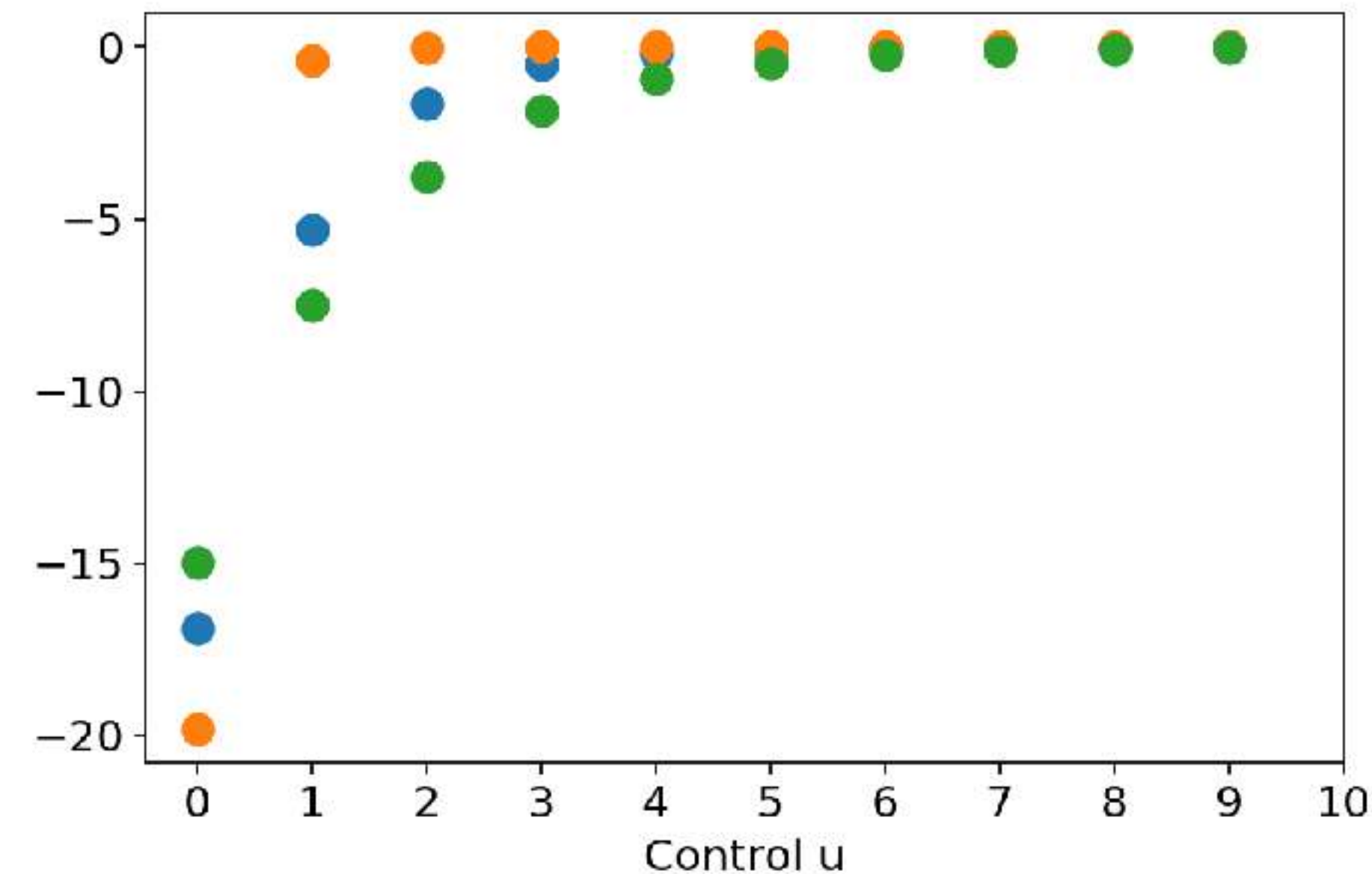
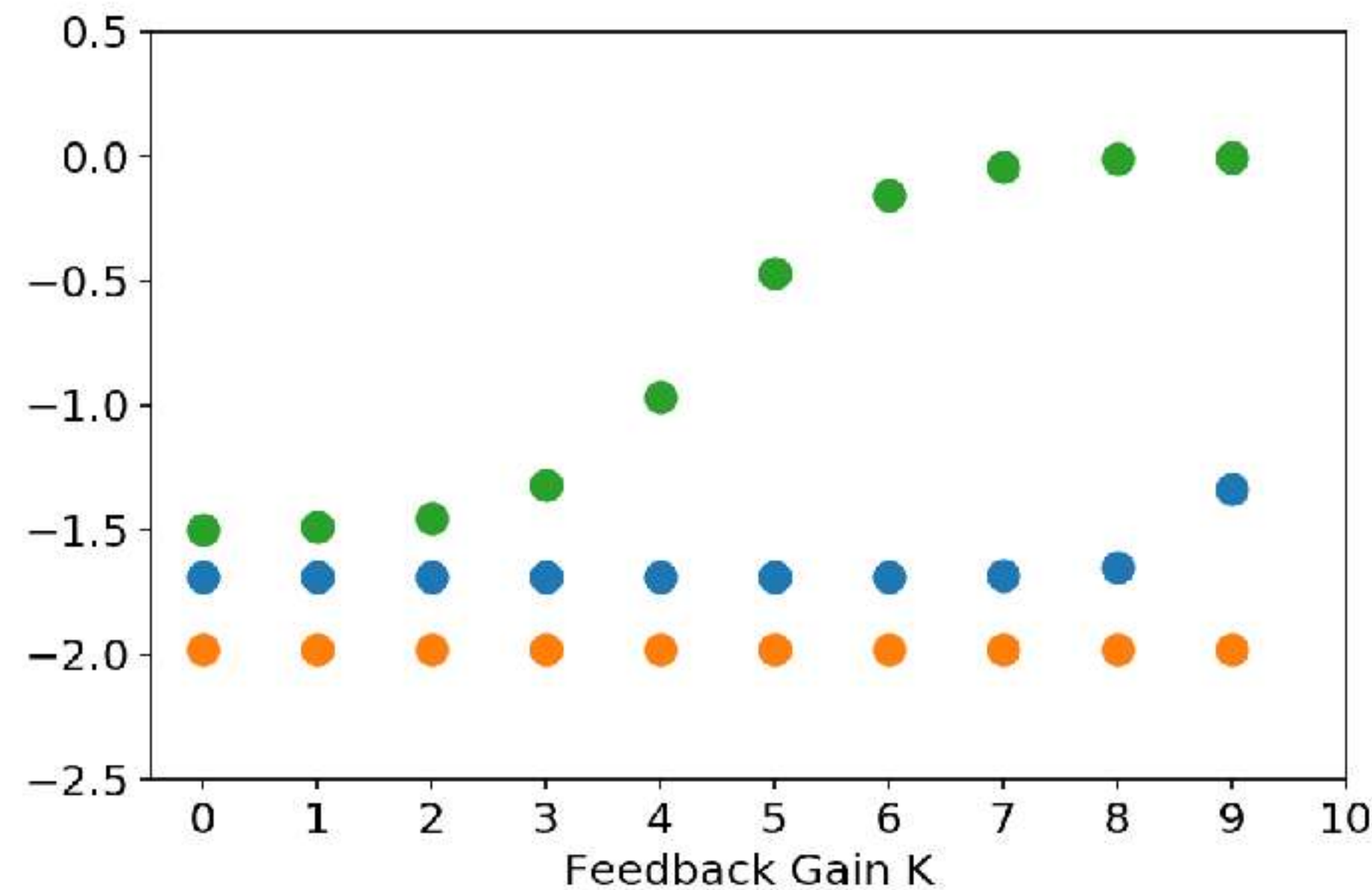
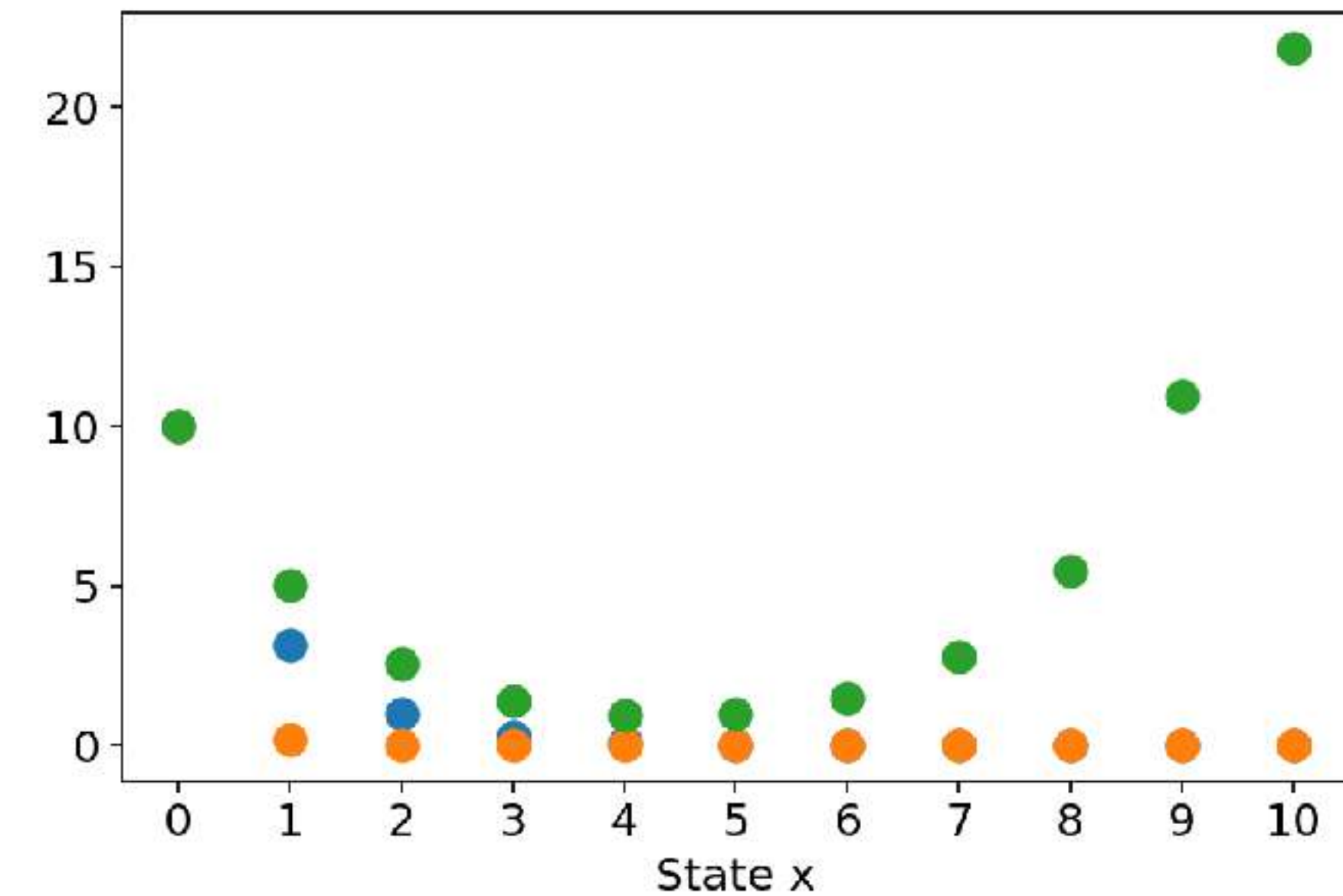
$N = 10 \quad x_0 = 100$

$Q = 2 \quad R = 1$

$Q = 100 \quad R = 1$

$Q = 1 \quad R = 1000$

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



Effect of the horizon N

$$Q = 2 \quad R = 1$$

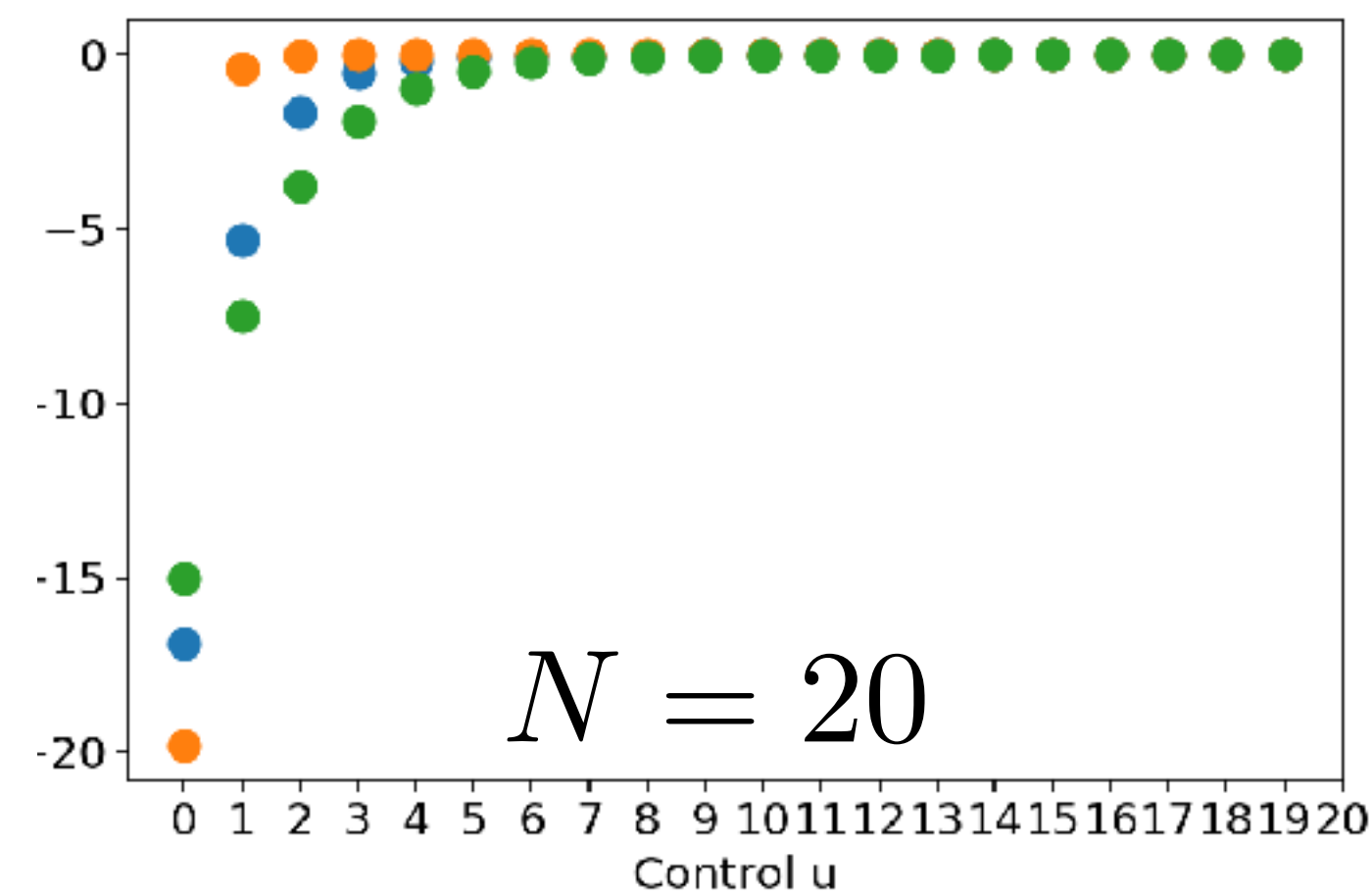
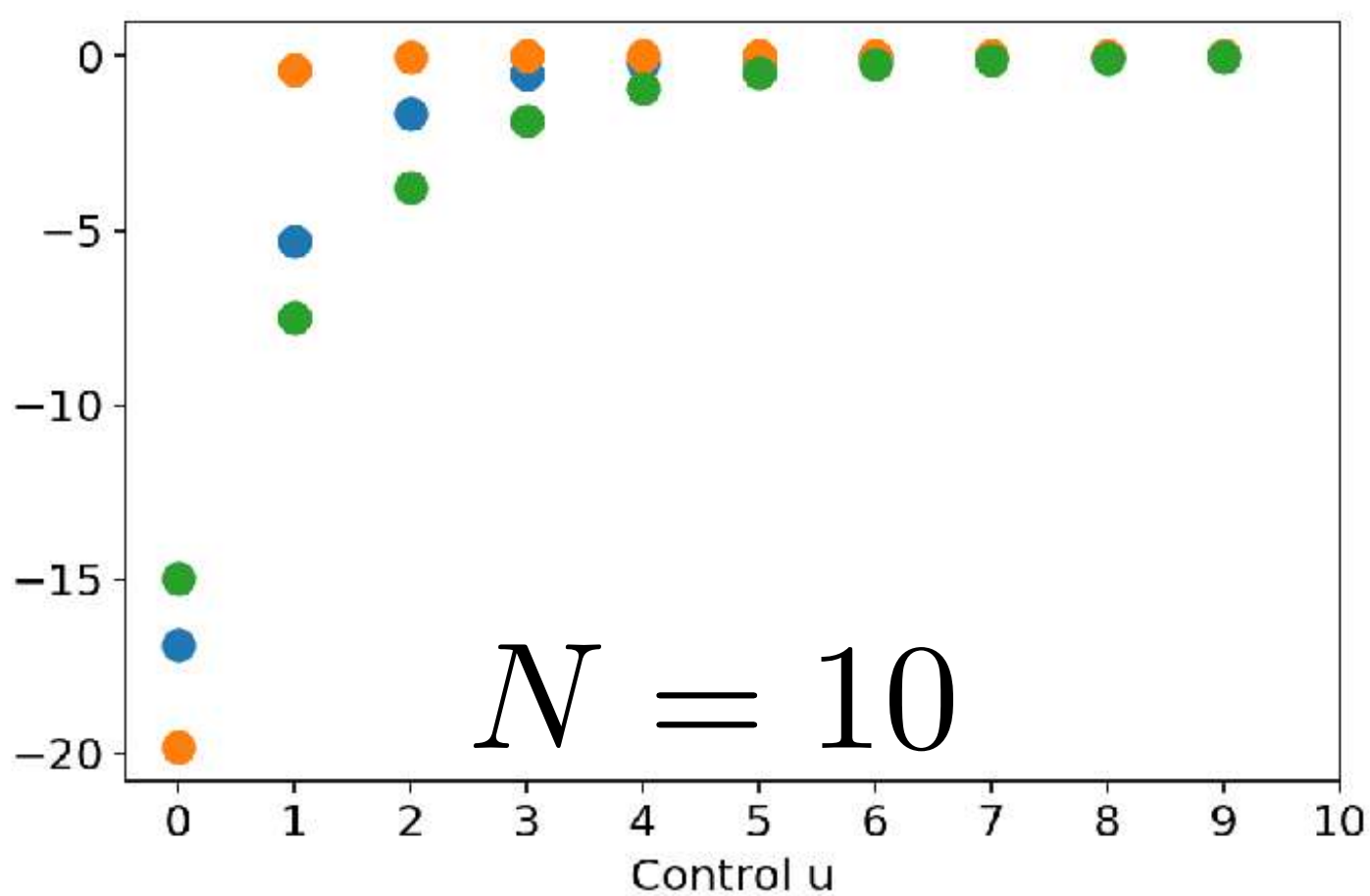
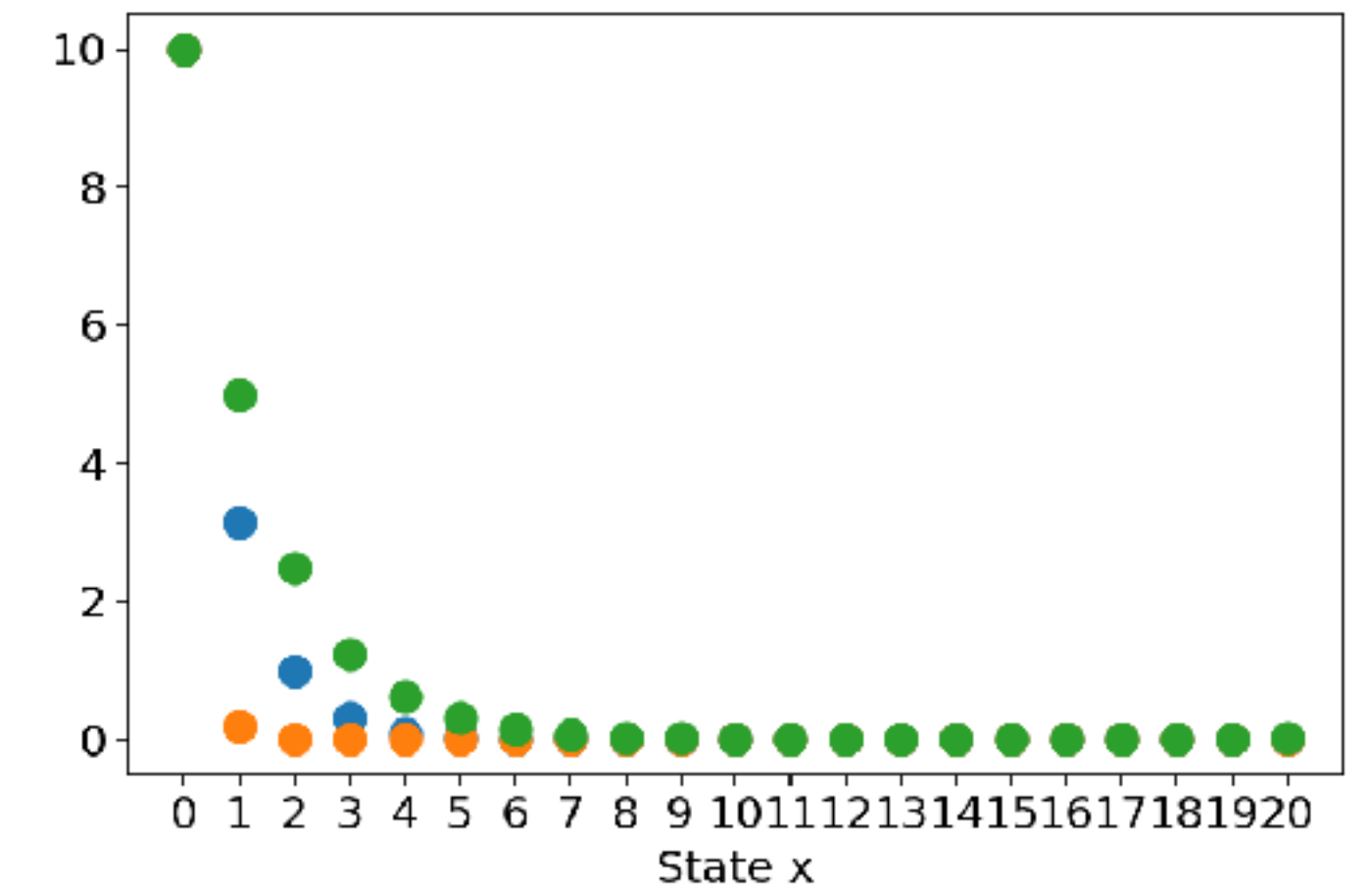
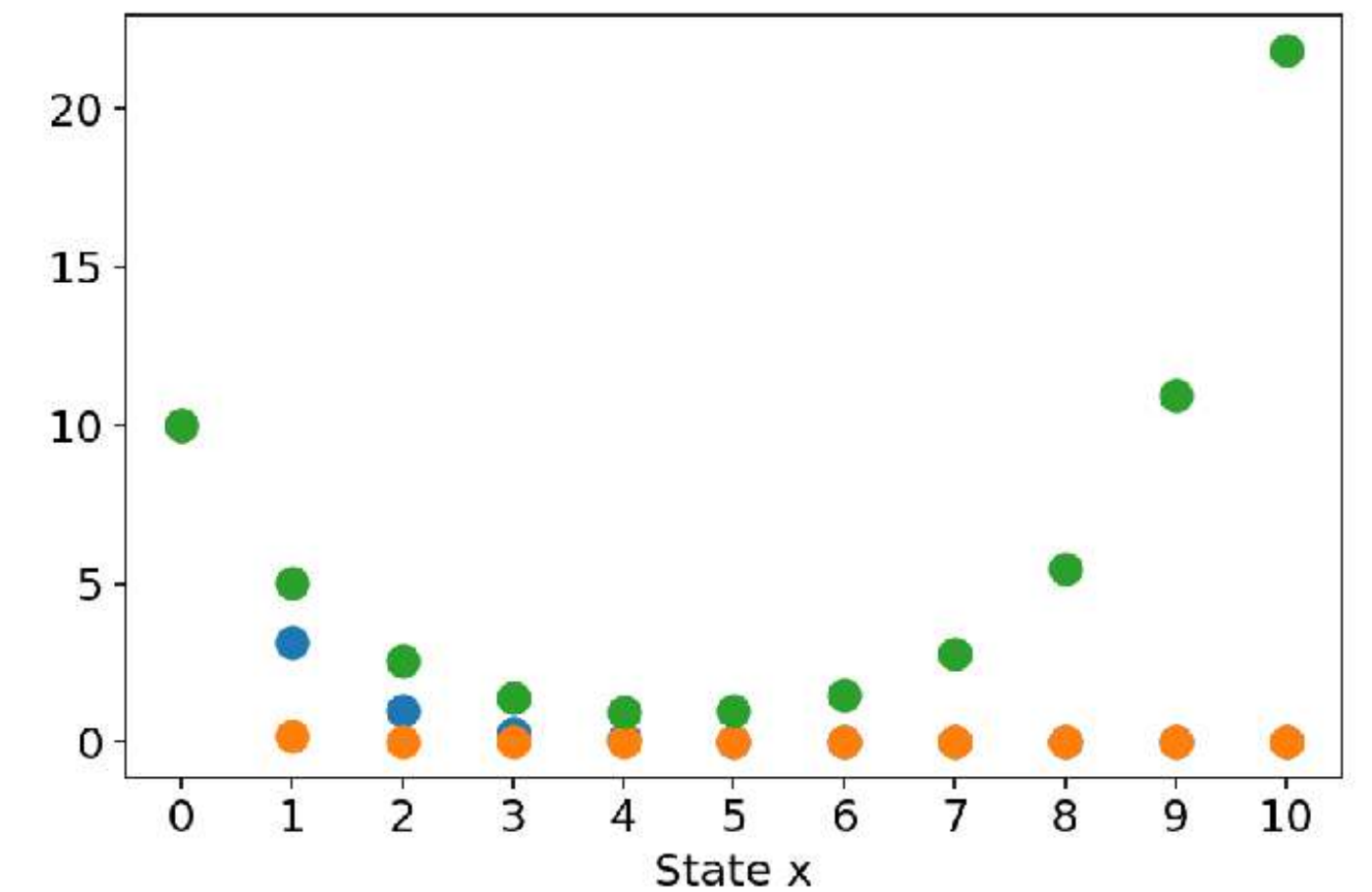
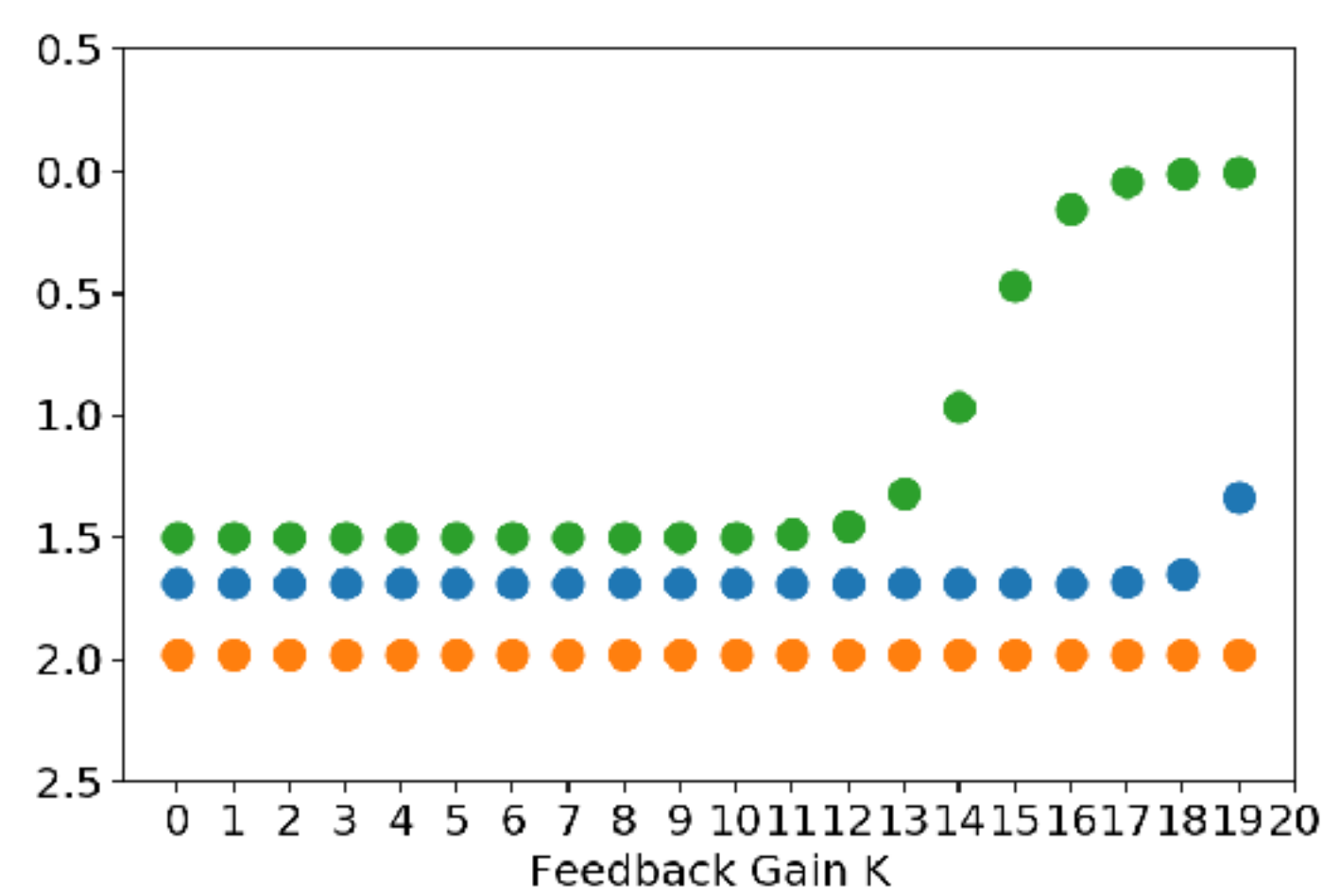
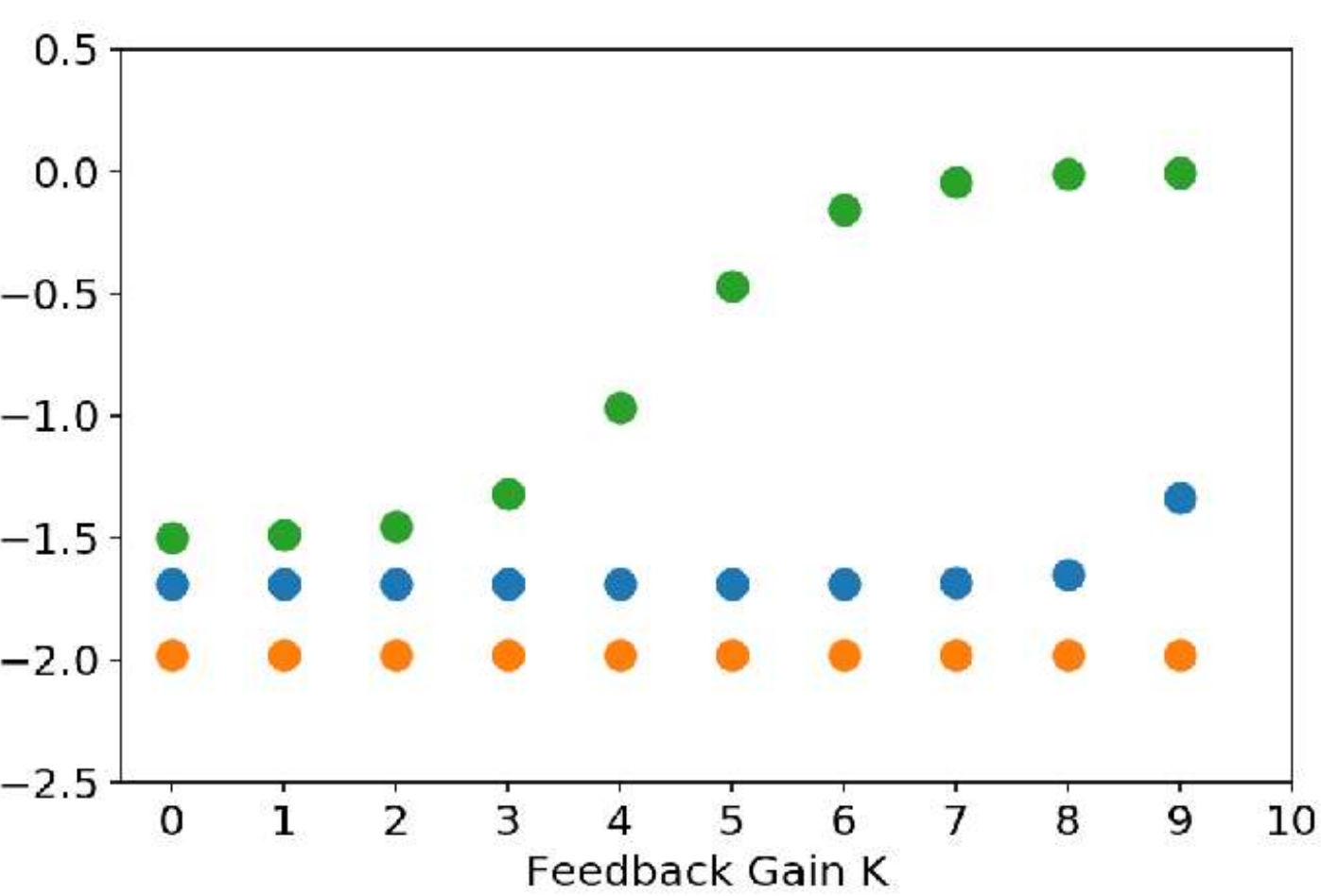
$$Q = 100 \quad R = 1$$

$$Q = 1 \quad R = 1000$$

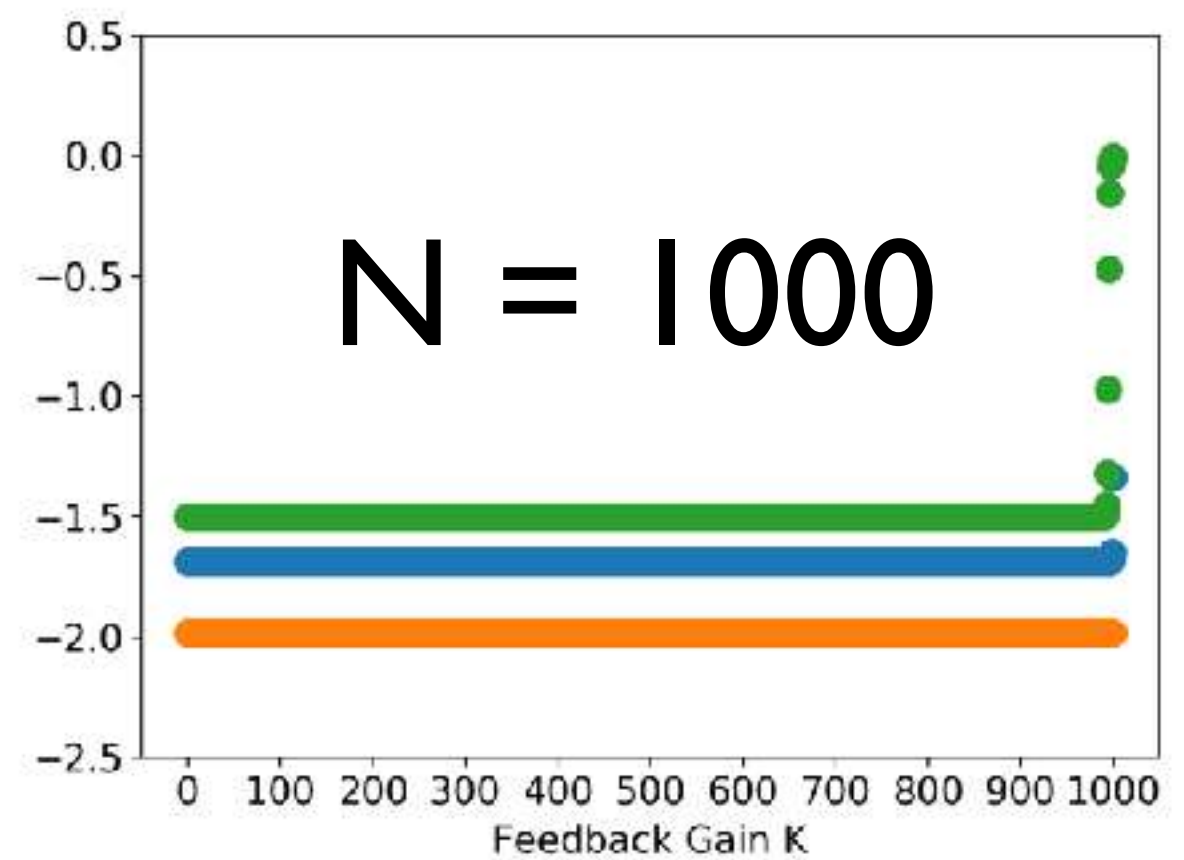
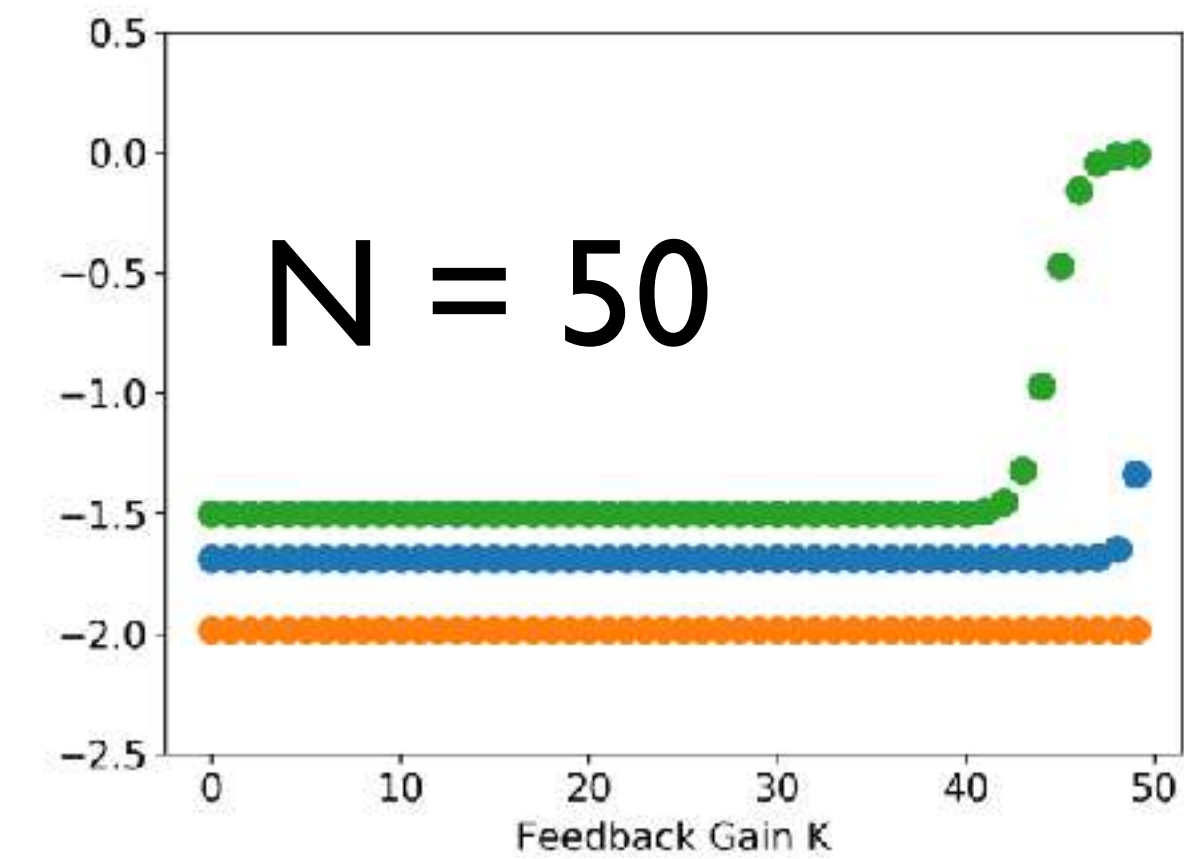
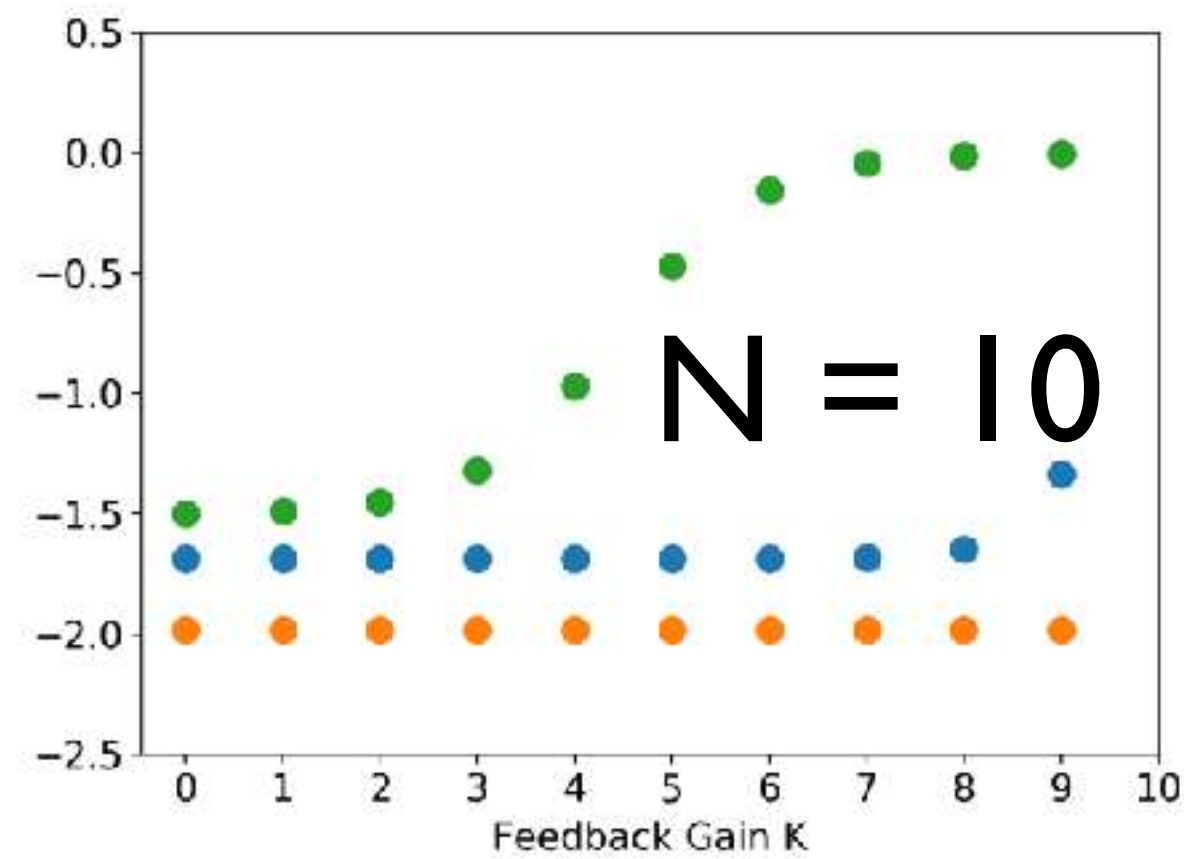
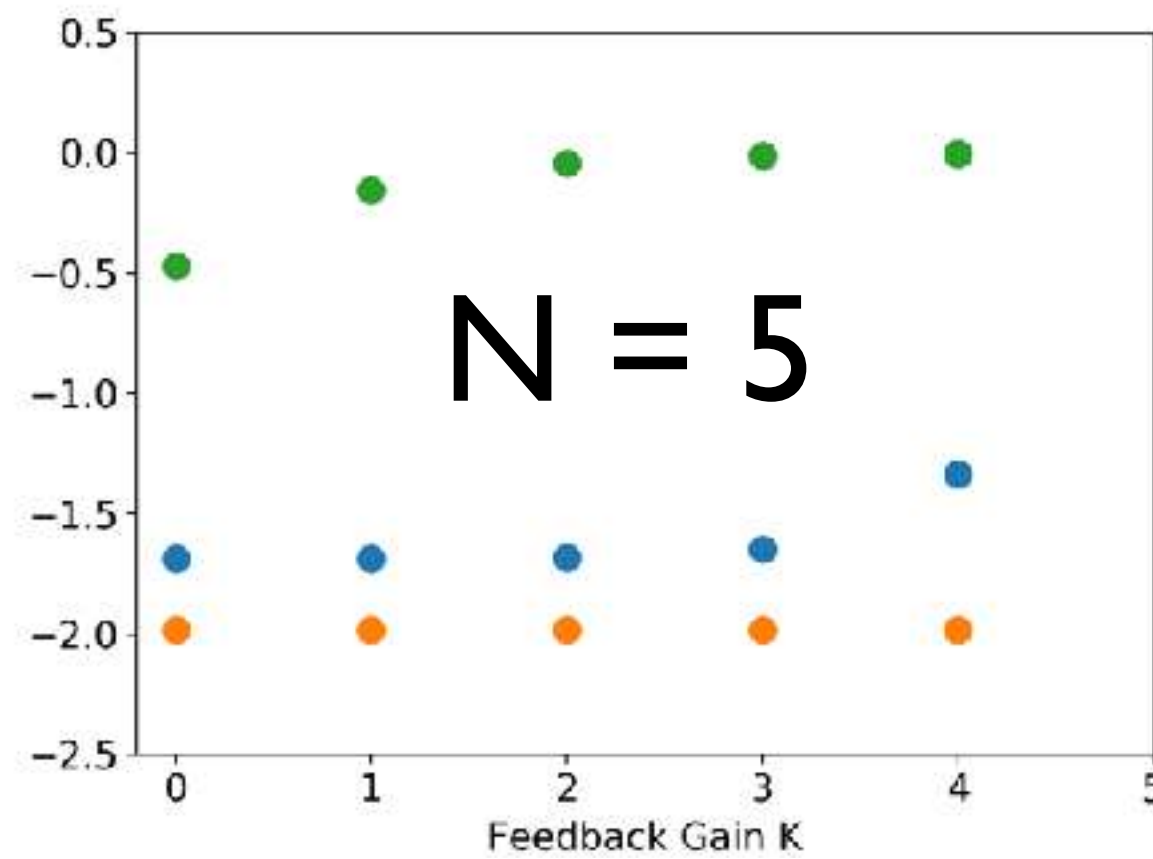
gains seem constant for
early stages

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 $\lim_{n \rightarrow \infty} K_n = K$ $\lim_{n \rightarrow \infty} P_n = P$

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)

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