Topics in Reinforcement Learning: AlphaZero, ChatGPT, Neuro-Dynamic Programming, Model Predictive Control, Discrete Optimization, Applications

Arizona State University

Course CSE 691, Spring 2025

Links to Class Notes, Videolectures, and Slides at
http://web.mit.edu/dimitrib/www/RLbook.html

Prof. Dimitri P. Bertsekas (dimitrib@mit.edu)
and
Dr Yuchao Li (yuchaoli@asu.edu)

Lecture 3
Approximation in Value Space - Visualization as Newton's Method
Problem Formulations, Reformulations, and Examples

Outline

On-Line Approximation in Value Space
System equation: $f(x, u, w)$, Cost per stage: $g(x, u, w)$, α-Discounted

One-step lookahead policy $\tilde{\mu}$

First Step "Future"

At state x → $\min_u E_w \Big\{ g(x, u, w) + \alpha \tilde{J}\big(f(x, u, w)\big) \Big\}$

CRITICAL MAPPING

Cost approximation $\tilde{J}$ → Cost function $J_{\tilde{\mu}}$

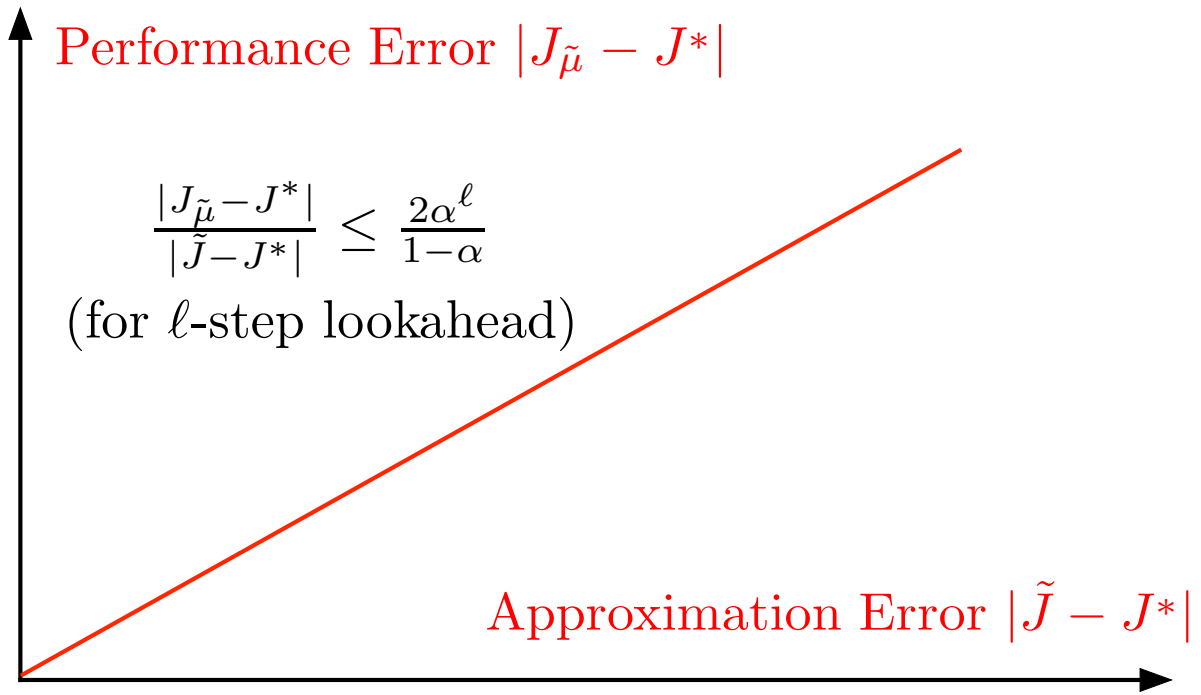
Approximation Error $|\tilde{J} - J^*|$ → Performance Error $|J_{\tilde{\mu}} - J^*|$

- Replace optimal cost J^* with an approximation $\tilde{J}$ in Bellman's equation
- Defines a lookahead policy $\tilde{\mu}$ with $\tilde{\mu}(x)$ being the minimizing u above

KEY QUESTIONS

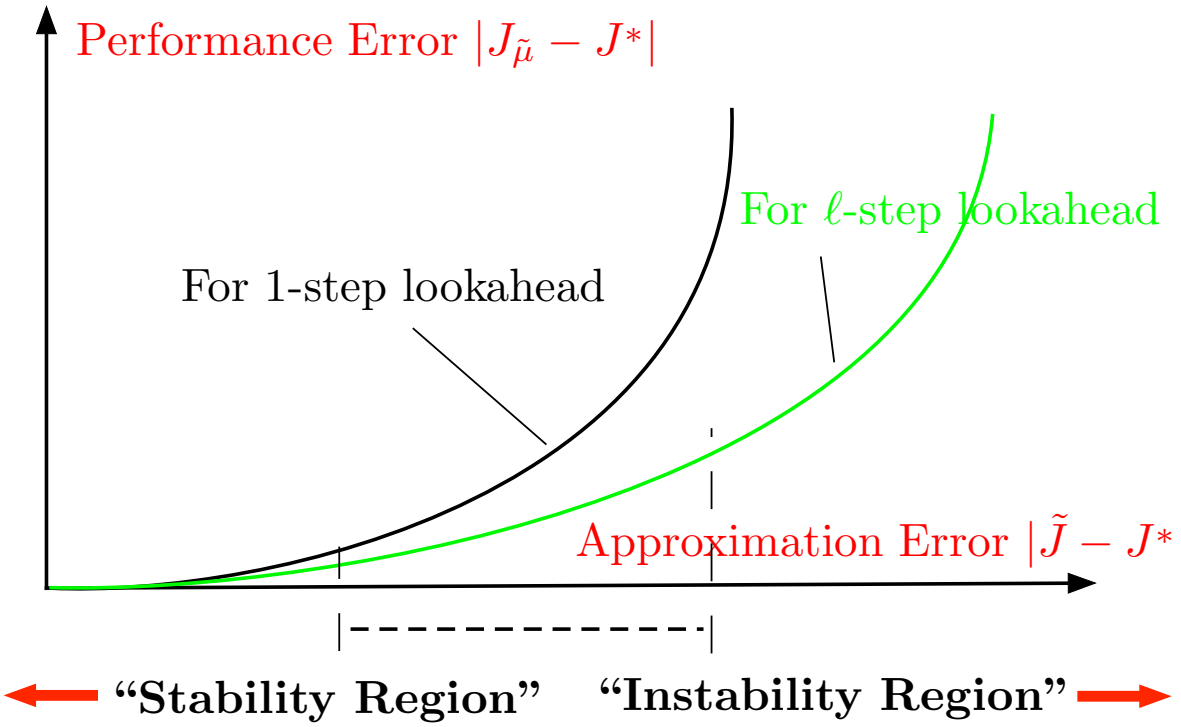
- What is the relation between $J_{\tilde{\mu}}$ and $\tilde{J}$?
- What is the role of multistep lookahead?
- How does the size of lookahead affect this relation?

The Linear Error Bound Model: An Example of Bad Theory



- These bounds are well-known to be conservative
- ... but they are broadly thought to be “qualitatively" correct
- THE REALITY IS FAR DIFFERENT
- The bounds are not only unrealistic, they are misleading
- They misdirect theoretical research and confuse the practitioners

The Real Relation is Superlinear



A key fact: The critical mapping is a Newton Step for solving the Bellman Eq.

- Convergence threshold defined by the region of convergence of Newton's method
- This has far-reaching implications for both theory and practice
- Inside the two regions, better training/more data, improving confidence intervals, etc, have marginal effect
- It is the limitations of the approximation method/neural net that are important!

Deterministic Linear Quadratic Problem - Infinite Horizon, Undiscounted

Linear system $x_{k+1} = ax_k + bu_k$; quadratic cost per stage $g(x,u) = qx^2 + ru^2$

Bellman equation: $J(x) = \min_u \{qx^2 + ru^2 + J(ax+bu)\}$

Main results:

- We have $J^*(x) = K^*x^2$, with K^* being a solution of the Riccati equation

$$K^* = F(K^*) = \frac{a^2 rK^*}{r + b^2 K^*} + q$$

  i.e., K^* is a fixed point of the Riccati operator $F(K) = \frac{a^2 rK}{r+b^2 K} + q$

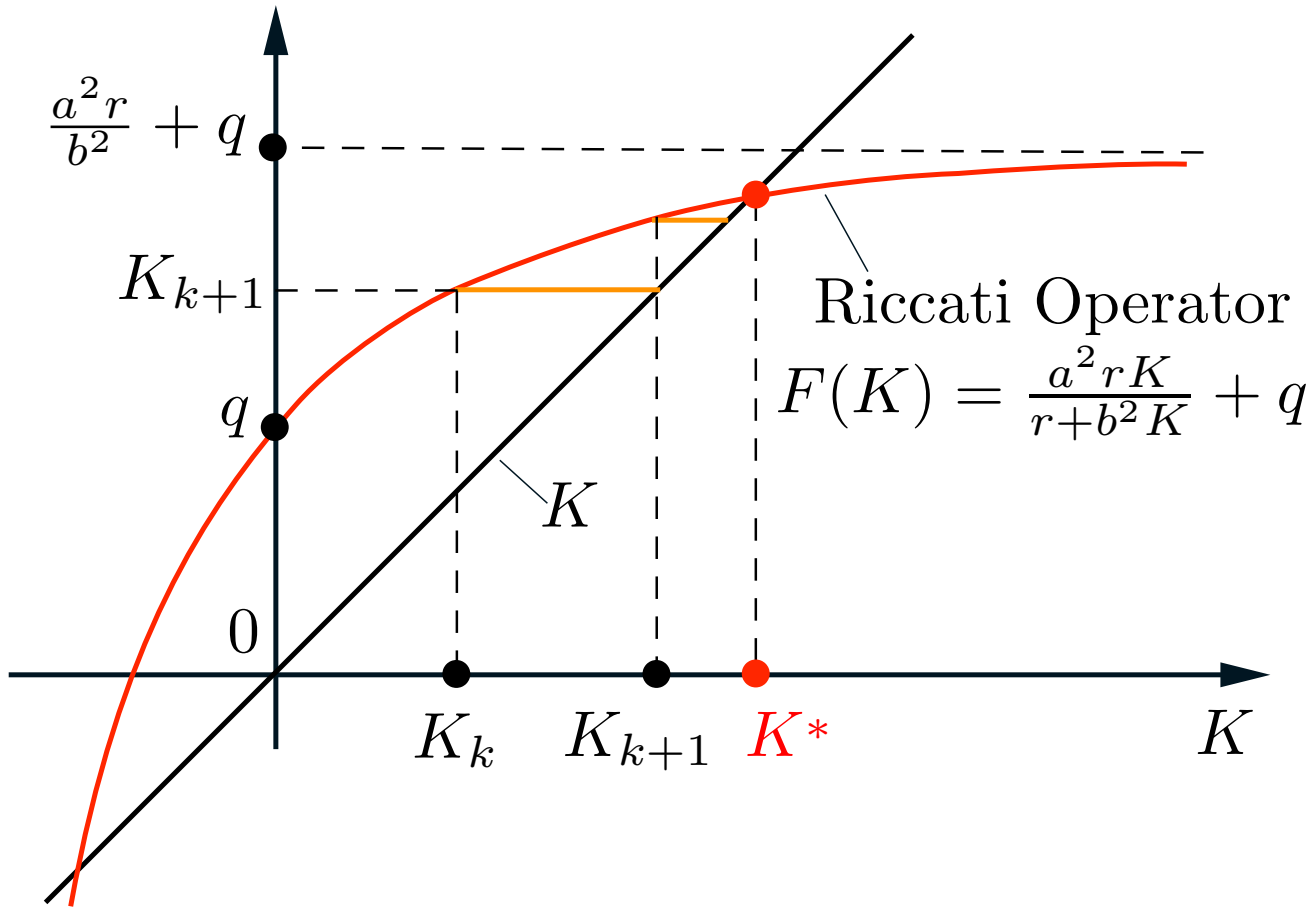
- The optimal policy is linear of the form

$$\mu^*(x) = L^*x \quad \text{with} \quad L^* = -\frac{abK^*}{r + b^2 K^*}$$

- VI generates iteratively the optimal cost functions $J_k(x_k)$ of k-stage problems
- J_k is quadratic of the form $J_k(x_k) = K_k x_k^2$, where $\{K_k\}$ is obtained by iterating with the Riccati operator F:

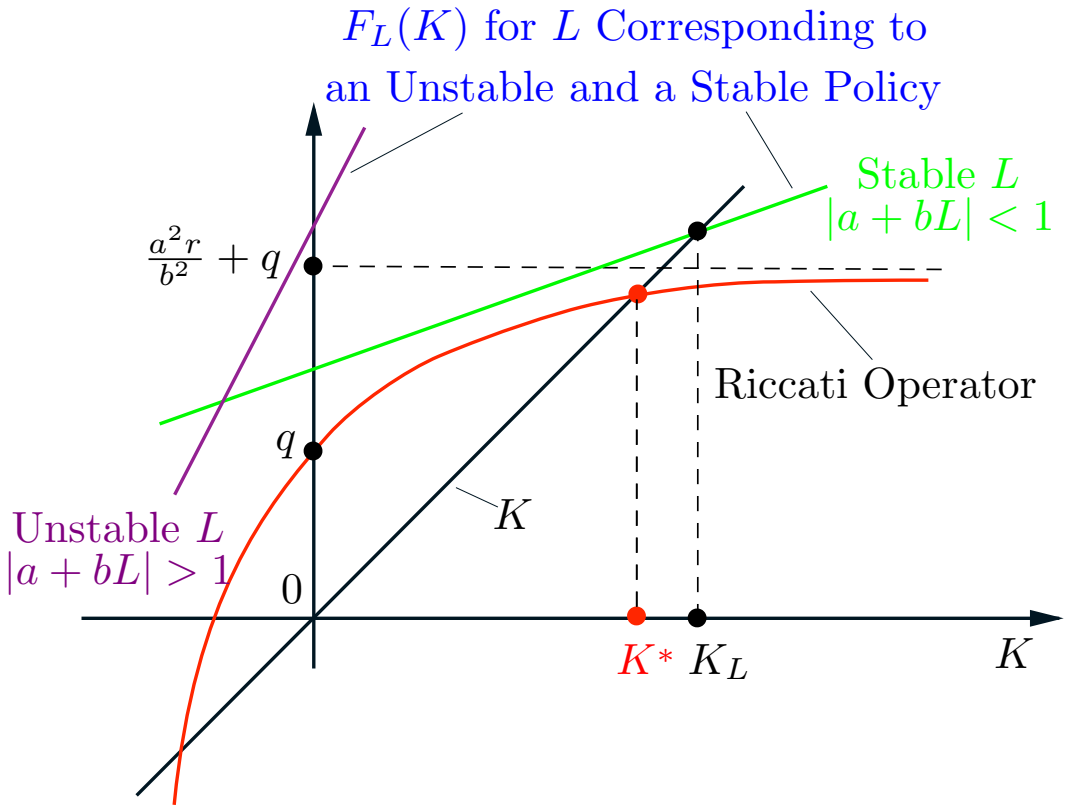
$$K_{k+1} = F(K_k), \quad k = 0, 1, \ldots, \qquad K_0 : \text{given}$$

Graphical Solution of the Riccati Equation and Value Iteration (VI)



Value iteration generates $\{K_k\}$ by iterating with the Riccati operator F: $K_{k+1} = F(K_k)$

Visualization of Riccati Equation for Linear Policies



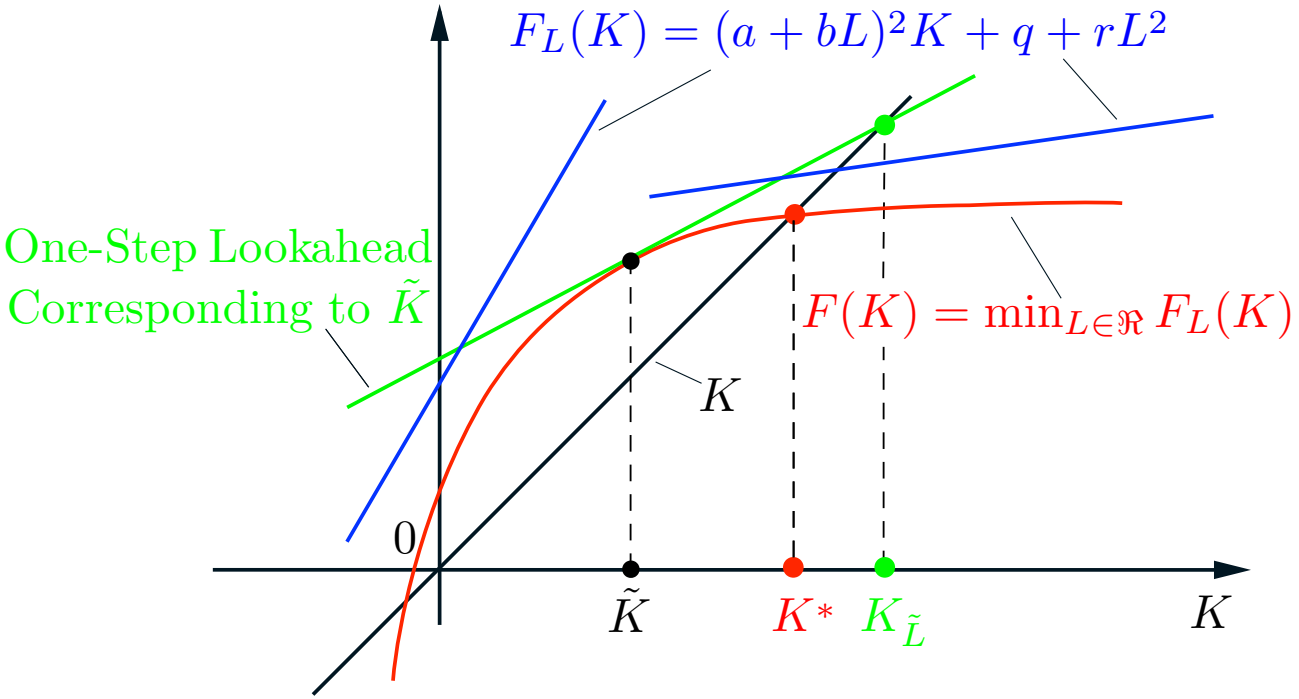
Consider a linear policy $\mu_L(x) = Lx$ and its cost function

It is quadratic of the form $K_L x^2$, where K_L is the unique solution of the Riccati equation for linear policies (also called Lyapunov equation):

$$K = F_L(K), \quad \text{where} \quad F_L(K) = (a+bL)^2 K + q + rL^2, \quad \text{it is linear in } K$$

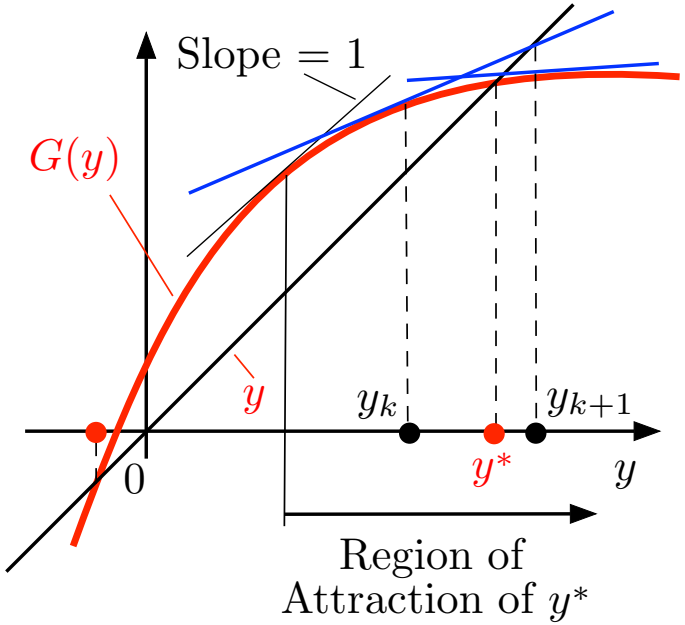
This is only for stable policies (those with $|a+bL|<1$). For unstable policies $K_L = \infty$

Visualization of Approximation in Value Space with One-Step Lookahead



- The graph of F is the lower envelope of the lines corresponding to linear policies (stable as well as unstable)
- The tangent line corresponding to $\tilde{K}$ defines the one-step lookahead policy with terminal cost $\tilde{K}x^2$
- It linearizes the Riccati operator at $\tilde{K}$
- Linearization is a critical property for the Newton step interpretation (concavity of the Riccati operator is also important)

Generic Newton's Method for Fixed Point Problem $y = G(y)$ (Theory Extends to Multi-Dimensional Problems Arising in DP)



At the typical iteration k

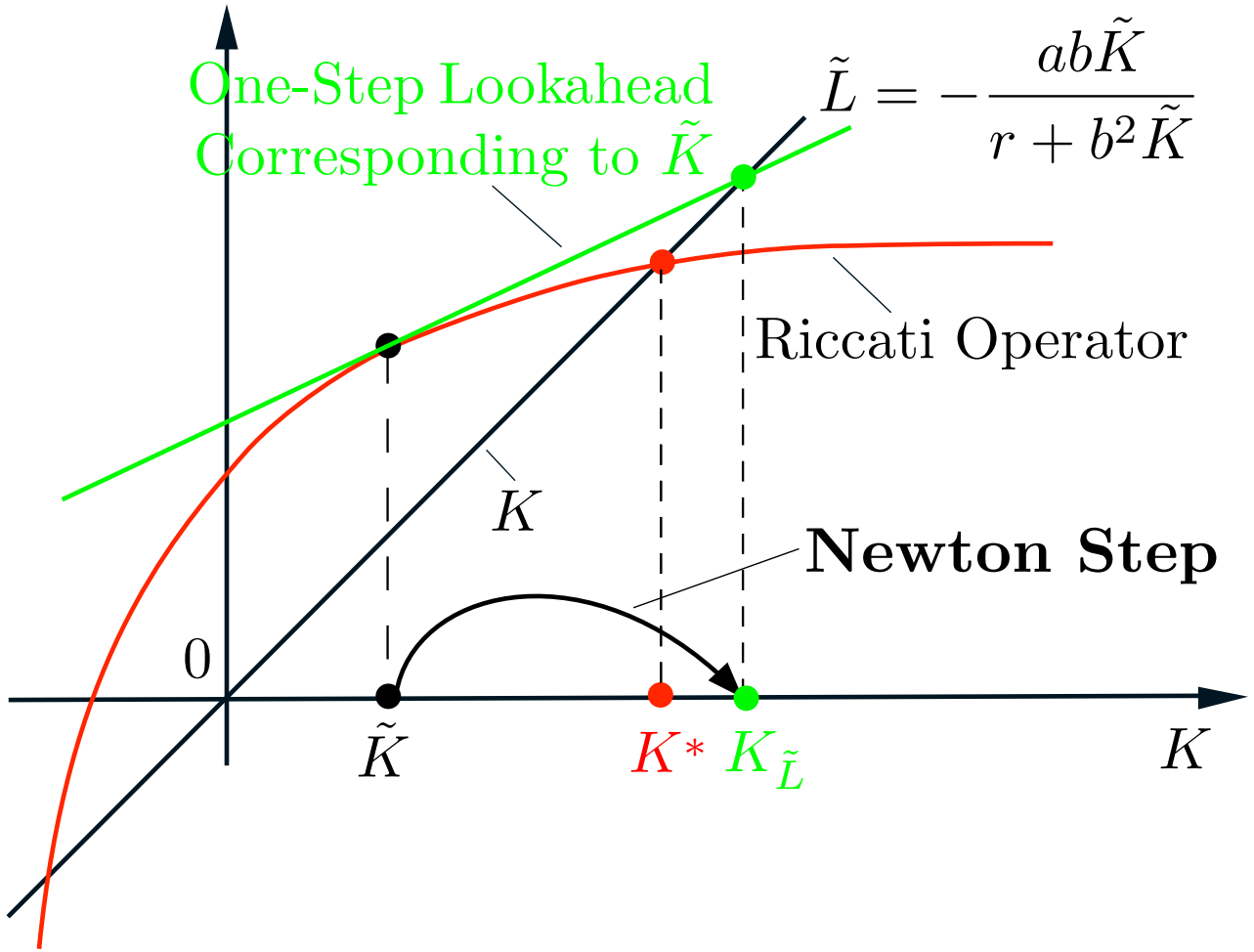
- We linearize the problem at the current iterate y_k with a first order expansion of G,

$$G(y) \approx G(y_k) + \nabla G(y_k)(y - y_k),$$

  where $\nabla G(y_k)$ is the gradient of G at y_k
- We solve the linearized problem to obtain y_{k+1}:

$$y_{k+1} = G(y_k) + \nabla G(y_k)(y_{k+1} - y_k)$$

Newton Step Visualization

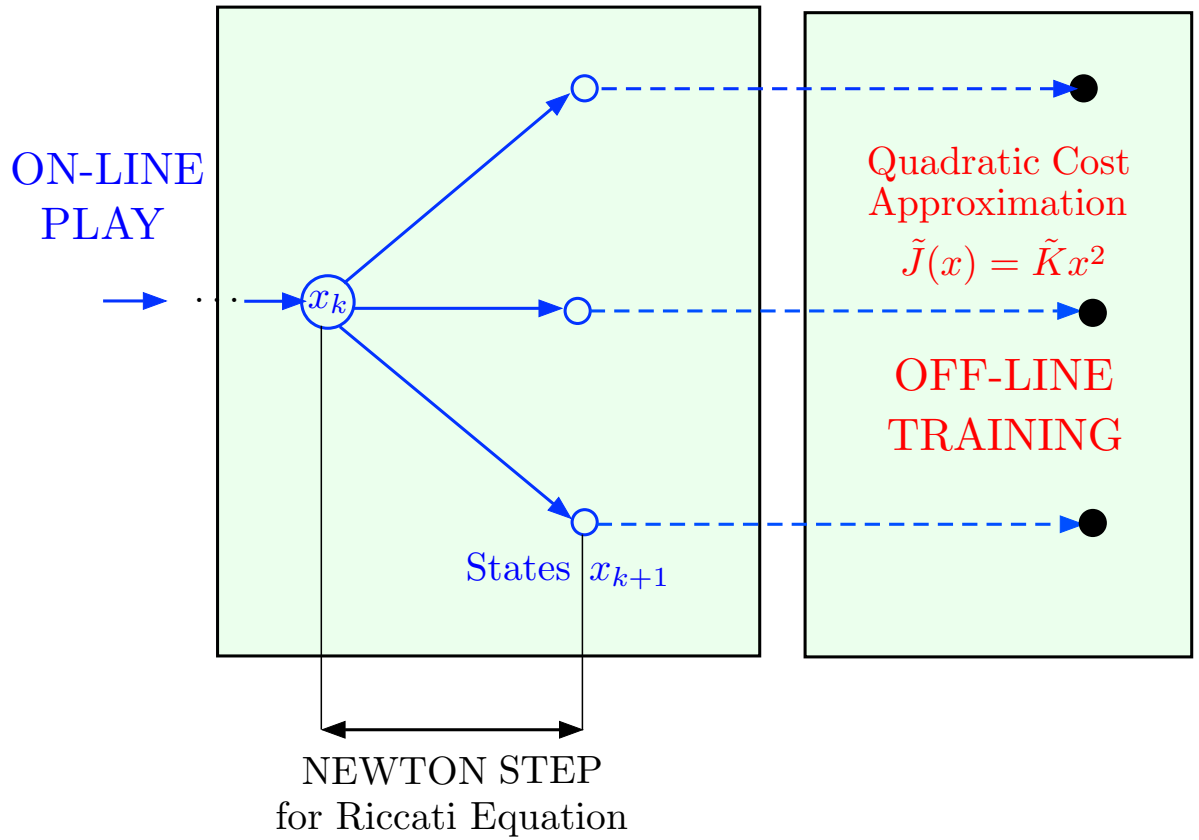


Newton step has superlinear convergence. As a result:

The error $|K_{\tilde{L}} - K^*|$ is MUCH smaller than $|\tilde{K} - K^*|$

Explains the good performance of approximation in value space in practice

Newton Step View of One-Step Lookahead

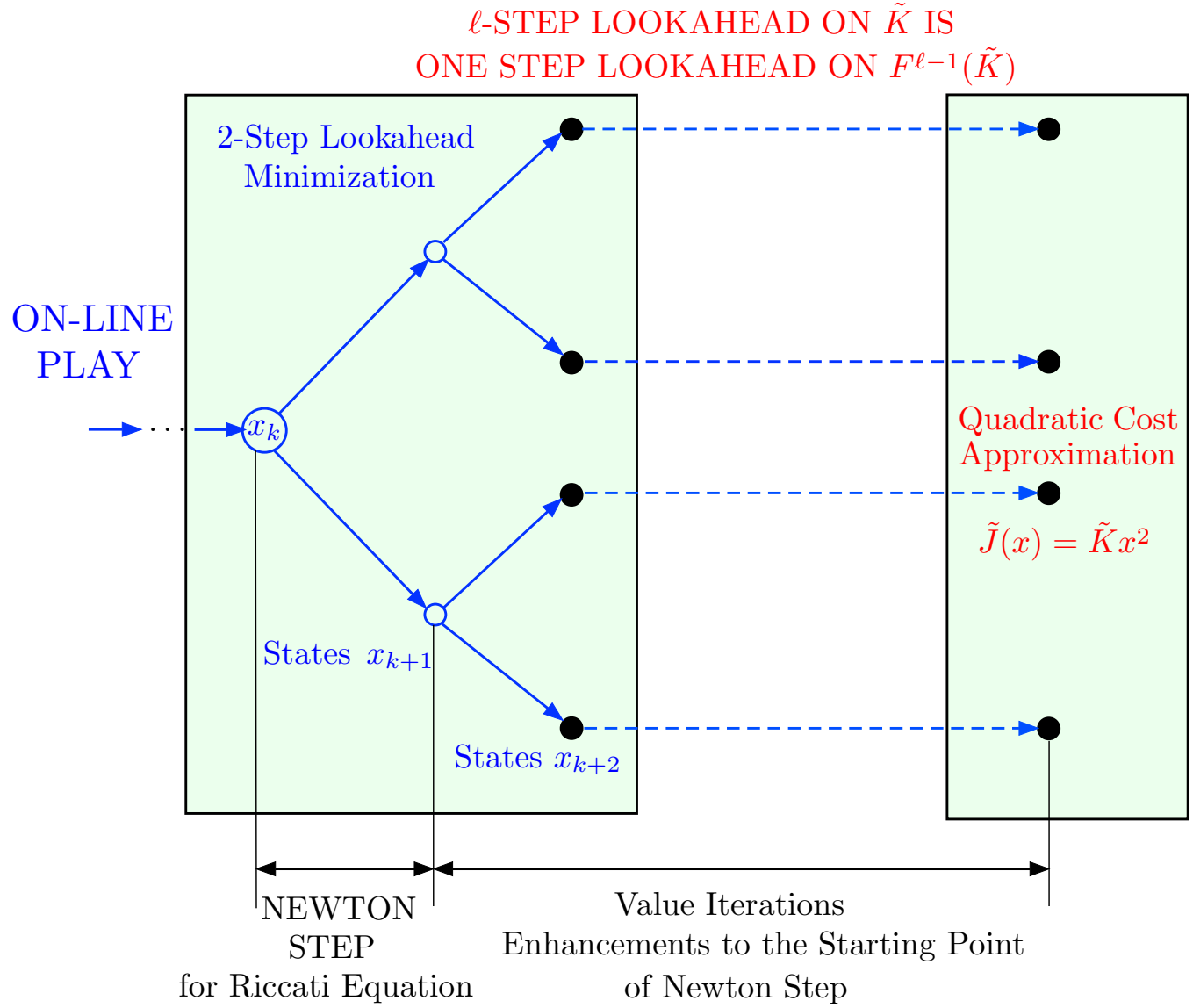


The Newton step with starting point $\tilde{K}$ is the mapping

$$\text{Cost approximation } \tilde{K} \quad \longmapsto \quad \text{Cost } K_{\tilde{L}} \text{ of the one-step lookahead policy}$$

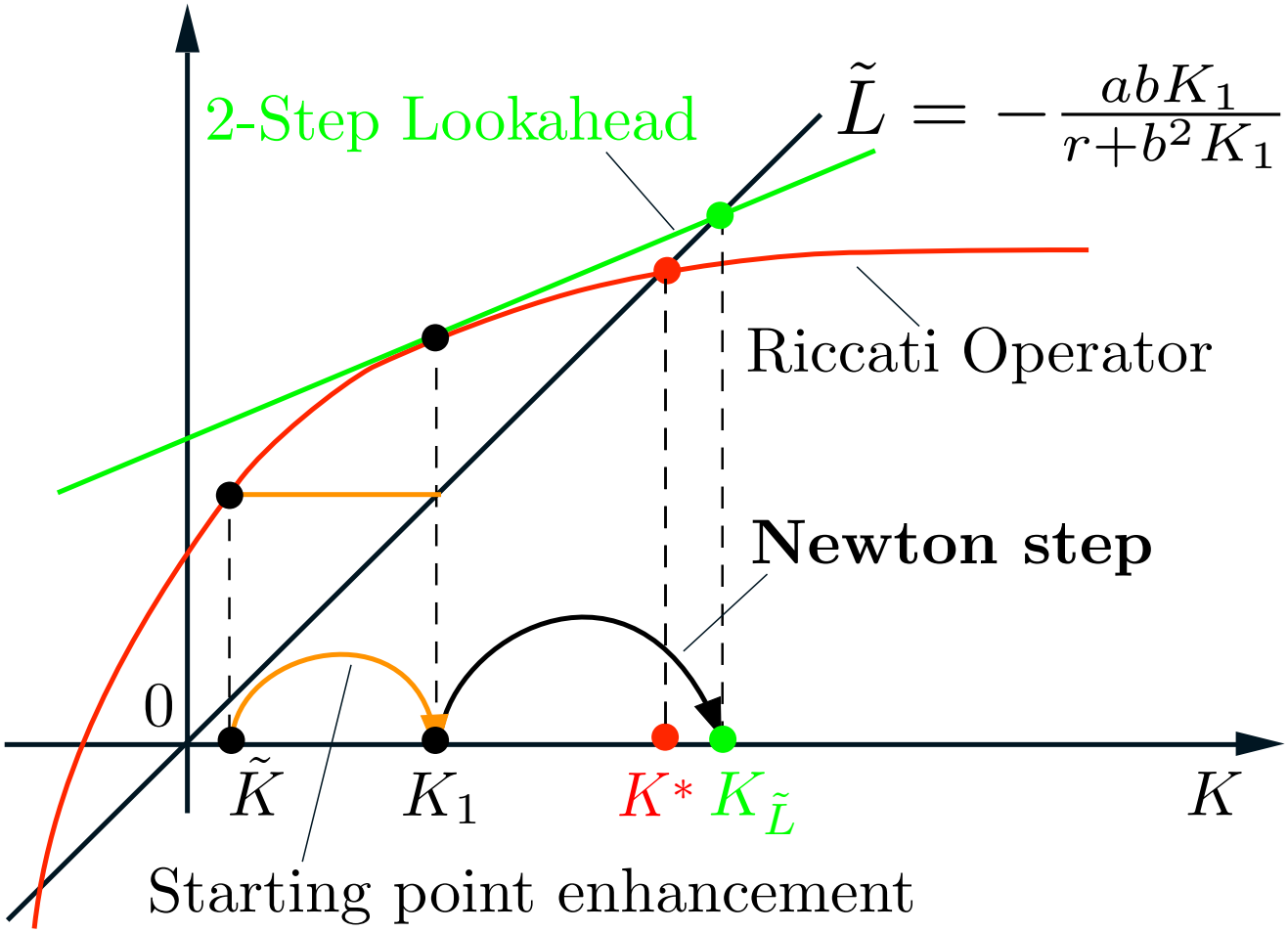
i.e., it relates (superlinearly) cost function approximation with one-step lookahead policy performance

Newton Step View of Multistep Lookahead



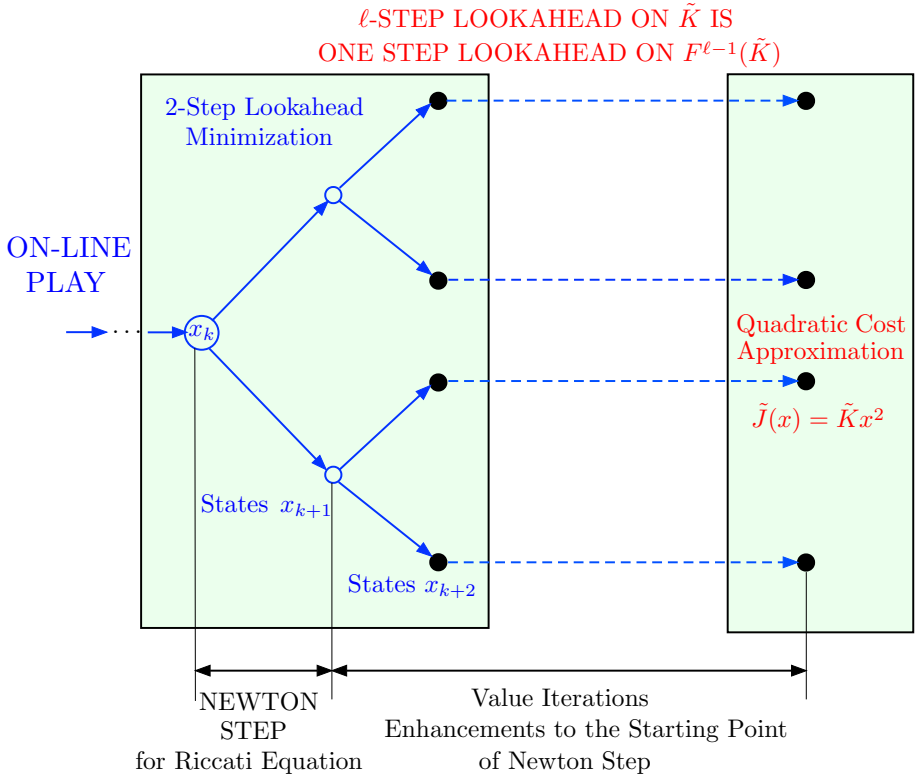
Only the first step of the lookahead is a Newton step

Multistep Lookahead Visualization



Multistep lookahead brings the starting point of the Newton step closer to K^*

The Importance of the First Step of Lookahead

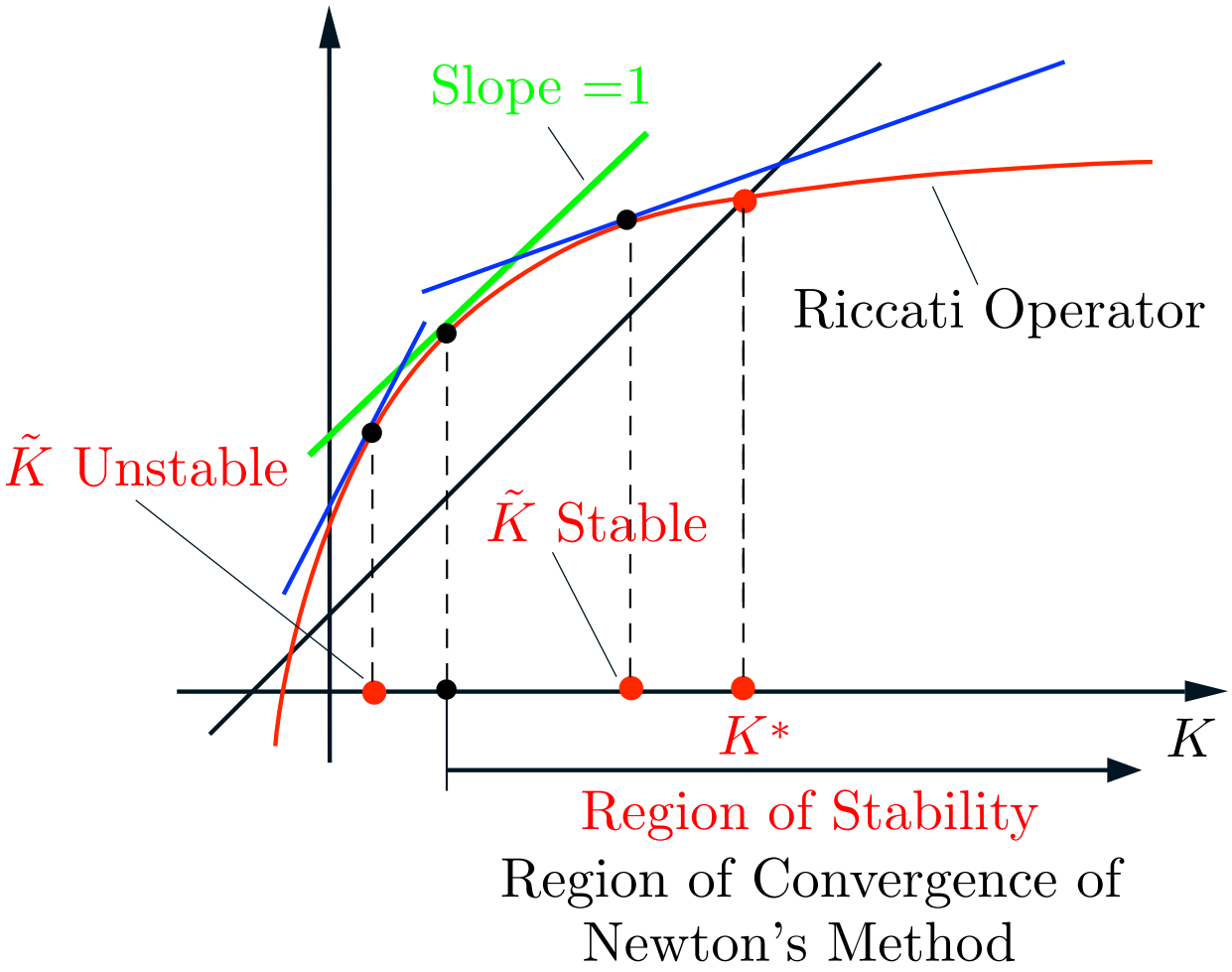


- In ℓ-step lookahead MPC, only the first step of lookahead acts as a Newton step
- The remaining $\ell - 1$ steps only serve to enhance the starting point of the first/Newton step
- Important insight: The first minimization step should be done exactly, the remaining steps can be done approximately
- Application: In stochastic problems, use certainty equivalence approximations in all lookahead steps except the first (Bertsekas and Castanon, 1999)

A Fifteen-Minute Break

Catch our breath and think about issues relating to the first half of the lecture.
Ask questions when you return.

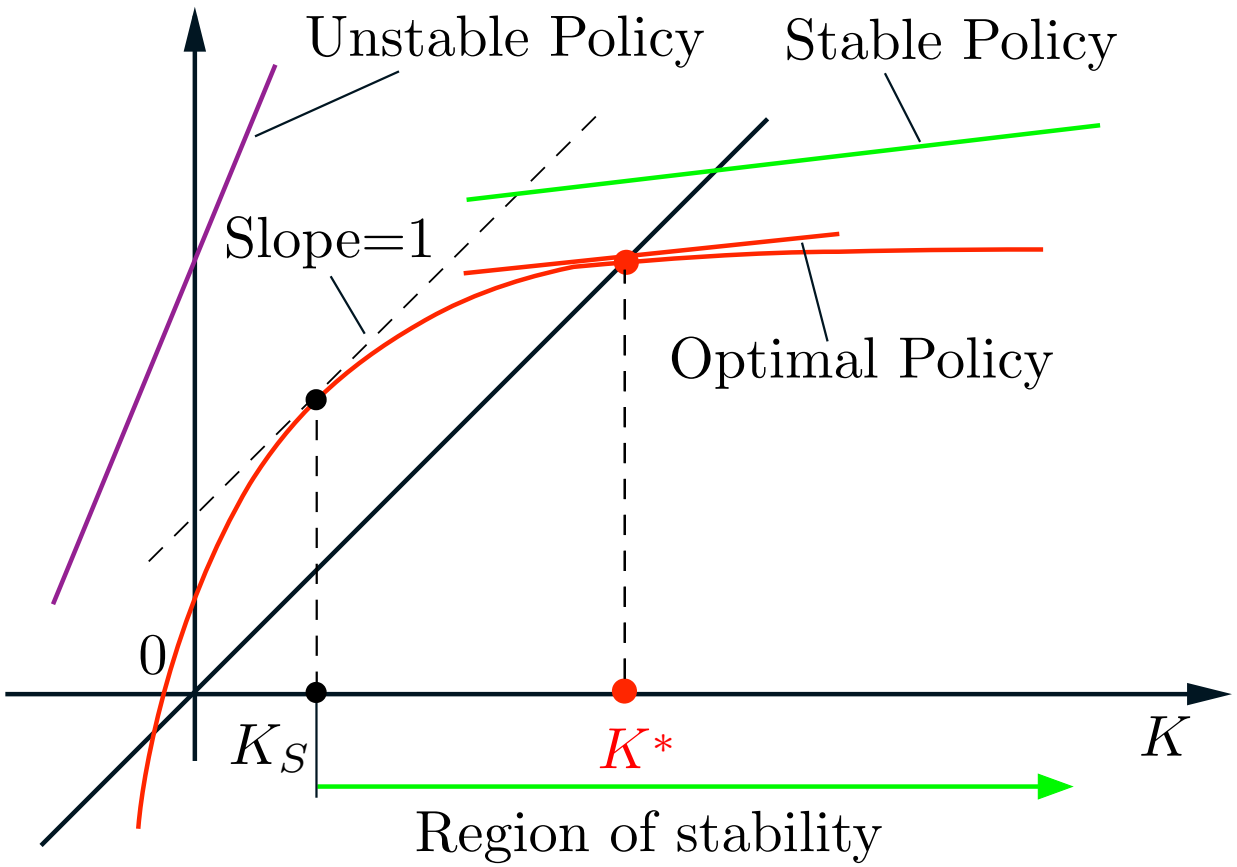
Region of Stability - When Does Approximation in Value Space Produce a Stable Policy?



The region of stability expands as the lookahead becomes longer

(Value iterations bring $\tilde{K}$ closer to K^*)

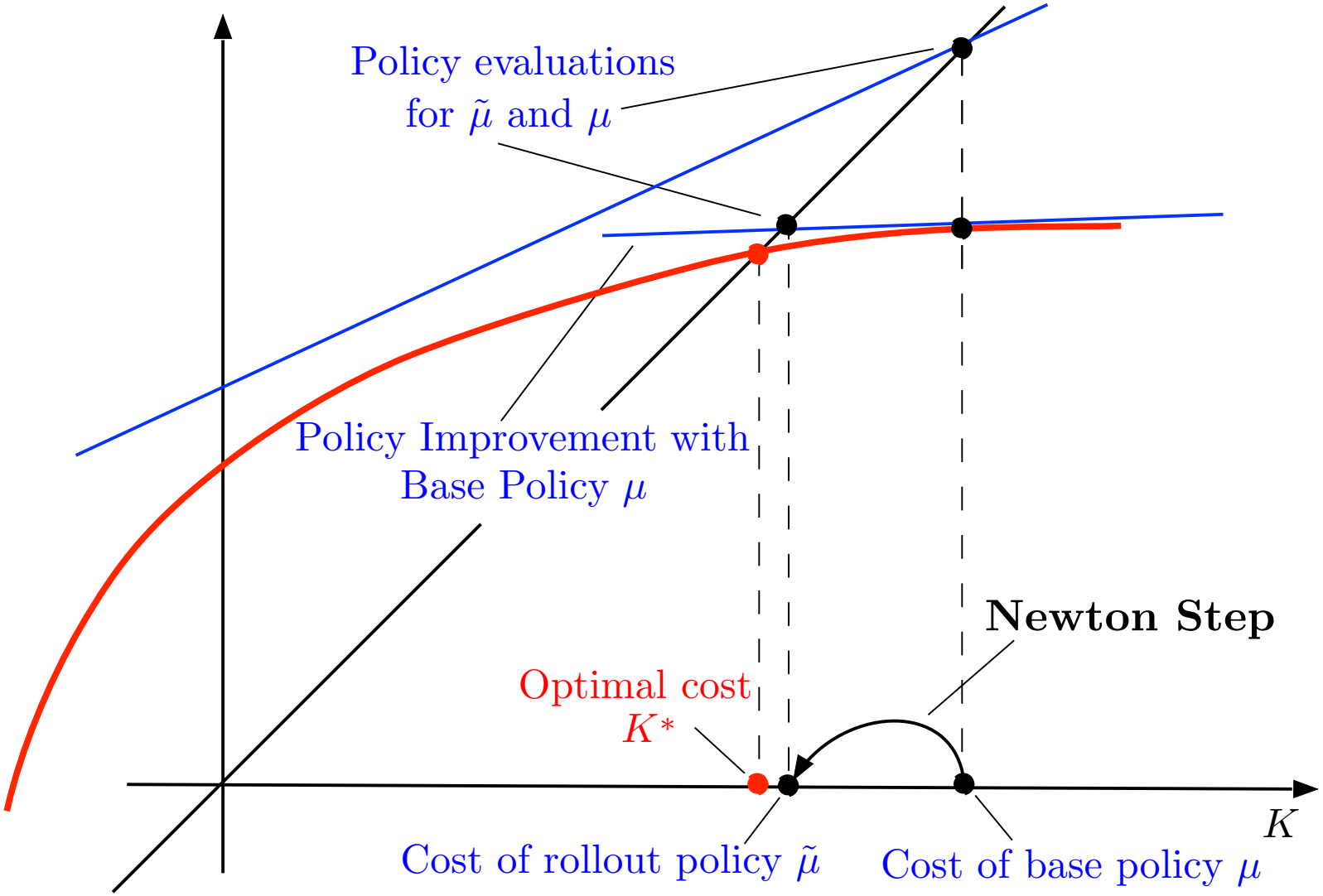
Visualization of Region of Stability of the One-Step Lookahead Policy $\tilde{\mu}$
The Set of $\tilde{K}$ for which $\tilde{\mu}$ is Stable



also
Region of Attraction of
Newton's Method

The start of the Newton step must be within the region of stability

Visualization of Rollout with Stable Linear Base Policy μ: $\tilde{J} = J_\mu$



Preservation of stability: If μ is stable, $\tilde{\mu}$ is also stable

Rollout is very reliable (practice is consistent with the superlinear convergence theory)

Policy Iteration for the Linear Quadratic Problem (Repeated Rollout)

Starts with linear policy $\mu^0(x) = L_0 x$, generates sequence of linear policies $\mu^k(x) = L_k x$ with a two-step process

- Policy evaluation:

$$J_{\mu^k}(x) = K_k x^2$$

where

$$K_k = \frac{q + rL_k^2}{1 - (a + bL_k)^2}$$

- Policy improvement:

$$\mu^{k+1}(x) = L_{k+1} x$$

where

$$L_{k+1} = -\frac{abK_k}{r + b^2 K_k}$$

- Rollout is a single Newton iteration
- PI is a full-fledged Newton method for solving the Riccati equation $K = F(K)$
- An important variant, Optimistic PI, consists of repeated truncated rollout iterations
- Can be viewed as a Newton-SOR method (repeated application of a Newton step, preceded by first order VIs)

Generalization

The Newton step interpretation of approximation in value space generalizes very broadly
See the "Lessons from AlphaZero ..." textbook

- Riccati operators —> Bellman operators
- Newton's method for solving the min-Riccati equation —> Newton's method for solving the min-Bellman equation
- A mathematical point: Nondifferentiabilty of the Bellman operator is not an issue (a form of Newton's method that can deal with nondifferentiability is used; see the "Lessons from AlphaZero ..." textbook)
- Approximation in value space is a single Newton iteration, enhanced by multistep lookahead (if any), and by truncated rollout (if any)
- Rollout is a single Newton iteration starting from the cost function of the (stable) base policy
- Exact PI is a full-fledged Newton's method
- Multistep lookahead and truncated rollout enhance the stability properties of the policy produced by approximation in value space

How do we Formulate DP Problems in Practice?

An informal recipe: First define the controls, then the stages (and info available at each stage), and then the states

- Define as state x_k something that “summarizes" the past for purposes of future optimization, i.e., as long as we know x_k, all past information is irrelevant.
- Rationale: The controller applies action that depends on the state. So the state must subsume all info that is useful for decision/control.

Some examples

- In the traveling salesman problem, we need to include all the relevant info in the state (e.g., the past cities visited, and the current city). Other info, such as the costs incurred so far, need not be included in the state.
- In partial or imperfect information problems, we use “noisy" measurements for control of some quantity of interest y_k that evolves over time (e.g., the position/velocity vector of a moving object). It is correct to use I_k (the collection of all measurements up to time k) as state.
- It may also be correct to use alternative states; e.g., the conditional probability distribution $P_k(y_k \mid I_k)$. This is called belief state, and subsumes all the information that is useful for the purposes of control choice.

State Augmentation: Delays

$$x_{k+1} = f_k(x_k, x_{k-1}, u_k, u_{k-1}, w_k)$$

- Introduce additional state variables y_k and s_k, where $y_k = x_{k-1}$, $s_k = u_{k-1}$. Then

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \\ s_{k+1} \end{pmatrix} = \begin{pmatrix} f_k(x_k, y_k, u_k, s_k, w_k) \\ x_k \\ u_k \end{pmatrix}$$

- Define $\tilde{x}_k = (x_k, y_k, s_k)$ as the new state, we have

$$\tilde{x}_{k+1} = \tilde{f}_k(\tilde{x}_k, u_k, w_k)$$

- Reformulated DP algorithm: Start with $J_N^*(x_N) = g_N(x_N)$

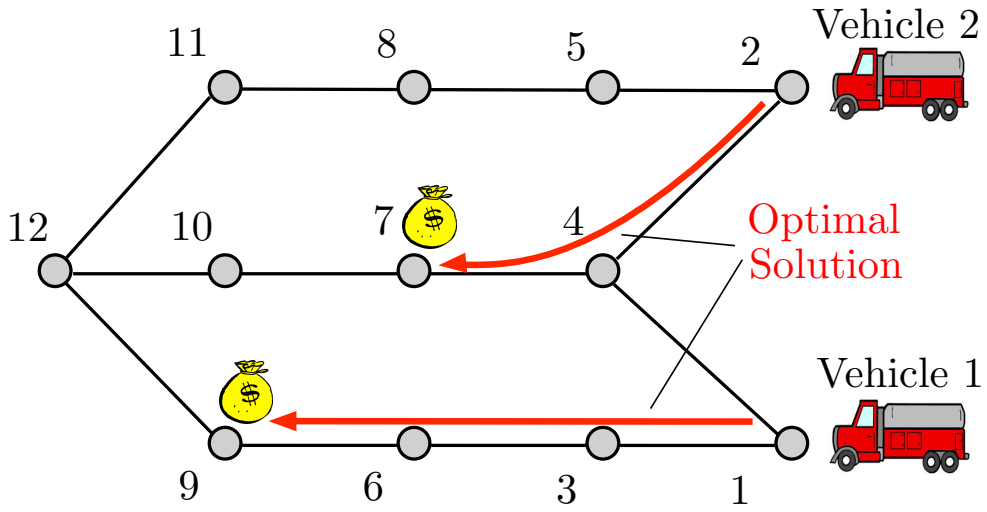
$$J_k^*(x_k, x_{k-1}, u_{k-1}) = \min_{u_k \in U_k(x_k)} E_{w_k}\left\{ g_k(x_k, u_k, w_k) + J_{k+1}^*\left(f_k(x_k, x_{k-1}, u_k, u_{k-1}, w_k), x_k, u_k\right)\right\}$$

$$J_0^*(x_0) = \min_{u_0 \in U_0(x_0)} E_{w_0}\left\{ g_0(x_0, u_0, w_0) + J_1^*\left(f_0(x_0, u_0, w_0), x_0, u_0\right)\right\}$$

See the textbook for other types of state augmentation (e.g., forecasts of future uncertainty)

Problems with a Cost-Free and Absorbing Terminal (Goal) State

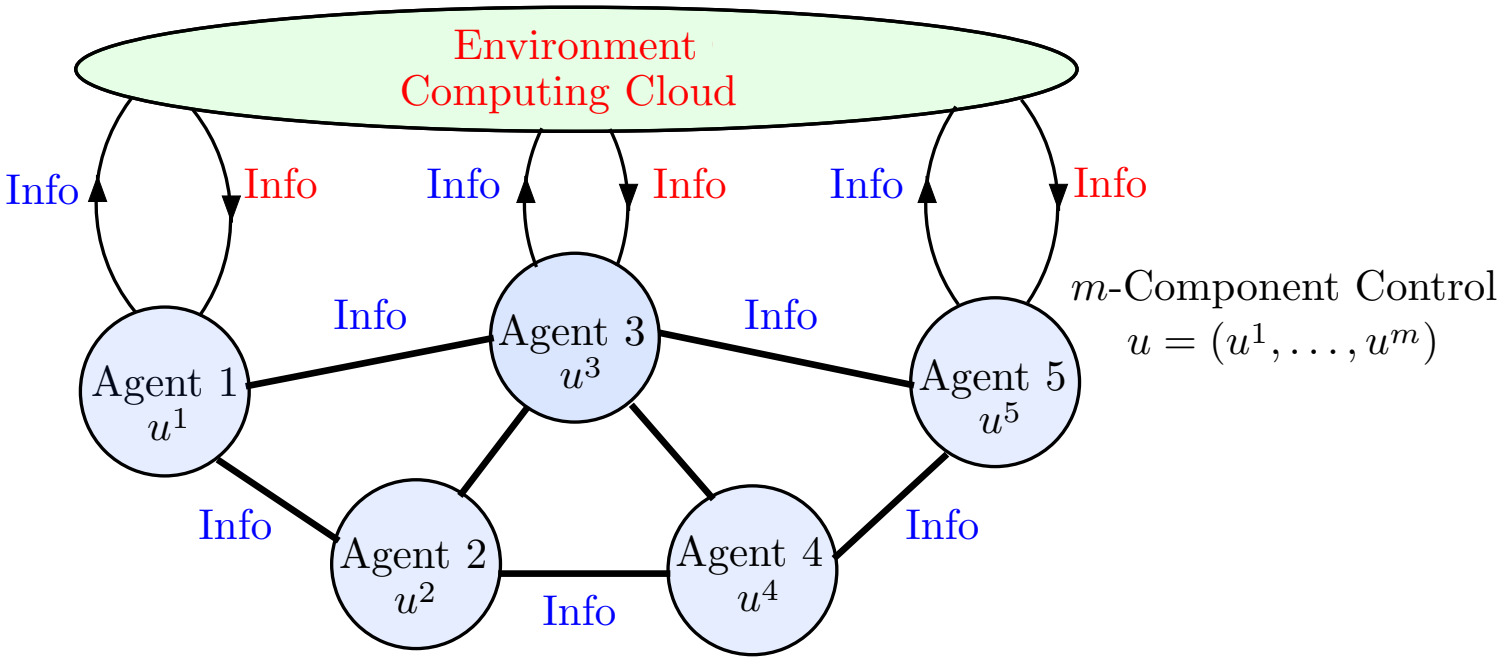
- Generally, we can view them as infinite horizon problems
- Another possibility is to convert to a finite horizon problem: Introduce as horizon an upper bound to the optimal number of stages (assuming such a bound is known)
- Add BIG penalty for not terminating before the end of the horizon



Example: Multiple vehicles move simultaneously one step at a time

- Minimize the number of moves to perform all tasks (i.e., reach the terminal state)
- How to formulate as DP? States? Controls? Terminal state? Horizon?
- Problem “size"? Astronomical, even for modest number of tasks and vehicles
- A good candidate for the multiagent framework to be introduced next

Multiagent Problems (1960s $\rightarrow$)

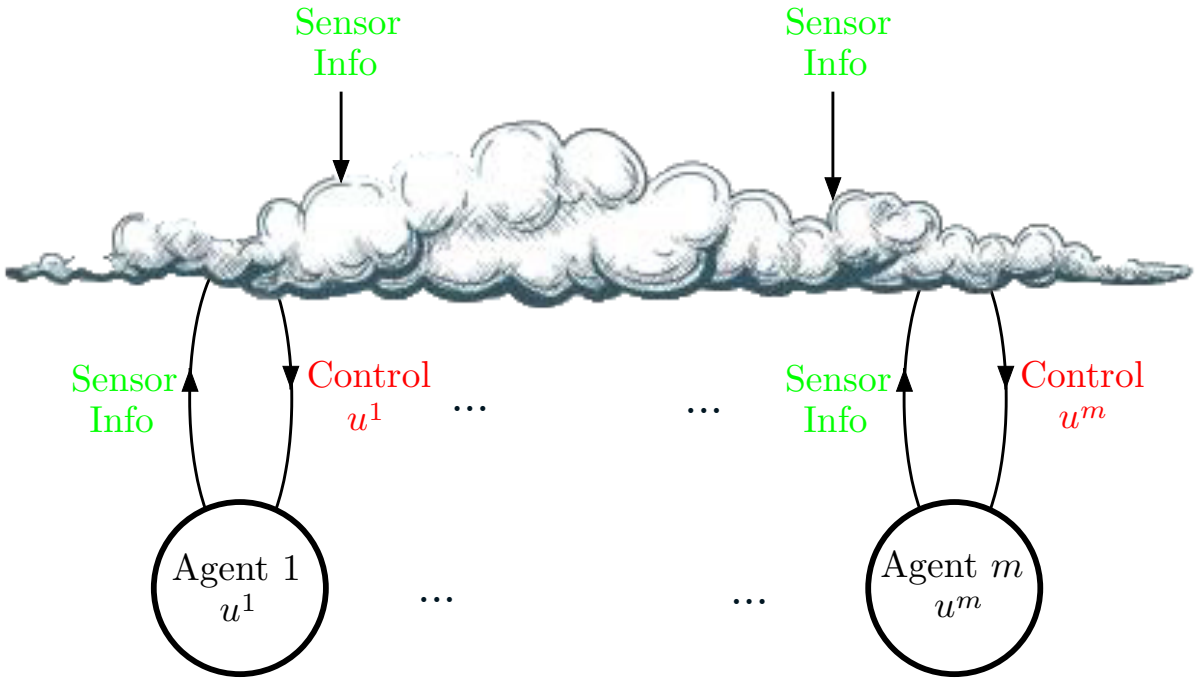


- Multiple agents collecting and sharing information selectively with each other and with an environment/computing cloud
- Agent i applies decision u^i sequentially in discrete time based on info received

The major mathematical distinction between problem structures

- The classical information pattern: Agents are fully cooperative, fully sharing and never forgetting information. Can be treated by DP
- The nonclassical information pattern: Agents are partially sharing information, and may be antagonistic. HARD because it is hard to treat by DP

Starting Point: A Classical Information Pattern (We Generalize Later)

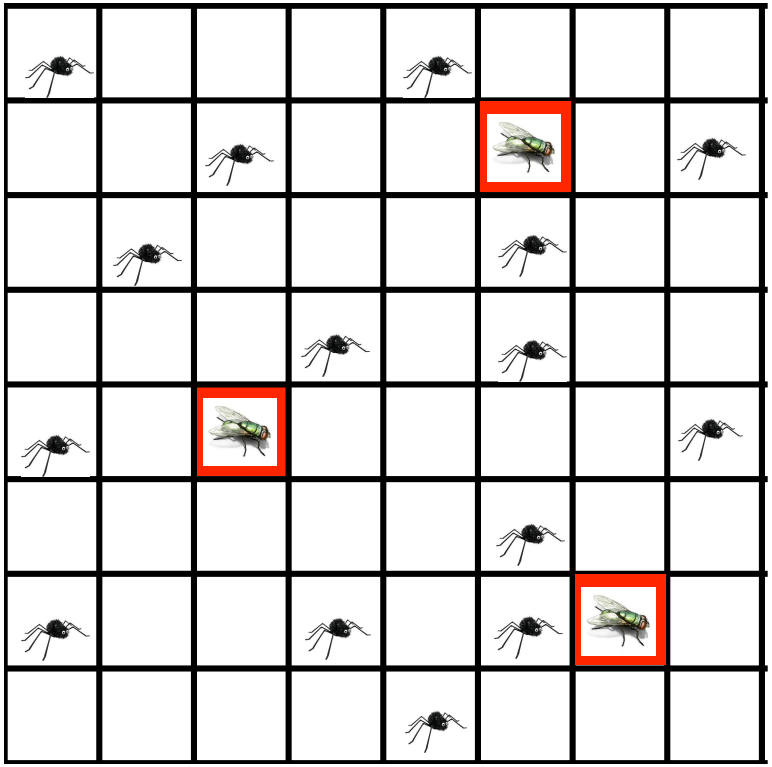


At each time: Agents have exact state info; choose their controls as function of state

Model: A discrete-time (possibly stochastic) system with state x and control u

- Decision/control has m components $u = (u^1, \ldots, u^m)$ corresponding to m "agents"
- "Agents" is just a metaphor - the important math structure is $u = (u^1, \ldots, u^m)$
- The theoretical framework is DP. We will reformulate for faster computation
  - We first aim to deal with the exponential size of the search/control space
  - Later we will discuss how to compute the agent controls in distributed fashion (in the process we will deal in part with nonclassical info pattern issues)

Spiders-and-Flies Example
(e.g., Vehicle Routing, Maintenance, Search-and-Rescue, Firefighting)



15 spiders move in 4 directions with perfect vision

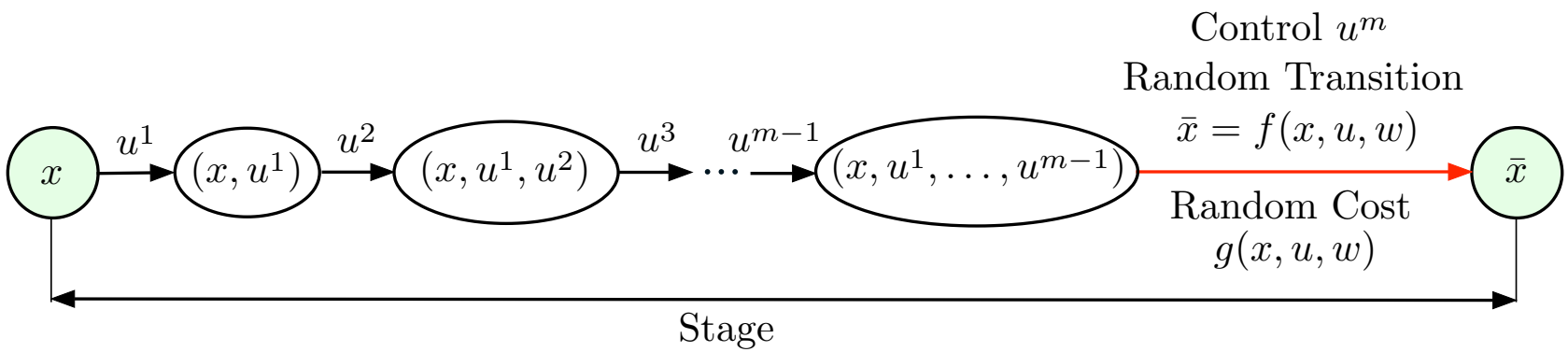
3 blind flies move randomly

Objective is to

Catch the flies in minimum time

- At each time we must select one out of $\approx 5^{15}$ joint move choices
- We will reduce to $5 \cdot 15 = 75$ (while maintaining good properties)
- Idea: Break down the control into a sequence of one-spider-at-a-time moves
- For more discussion, including illustrative videos of spiders-and-flies problems, see https://www.youtube.com/watch?v=eqbb6vVlN38&t=1654s

Reformulation Idea: Trading off Control and State Complexity (Bertsekas and Tsitsiklis NDP Book, 1996)



An equivalent reformulation - "Unfolding" the control action

- The control space is simplified at the expense of $m-1$ additional layers of states, and corresponding $m-1$ cost functions

$$J^1(x, u^1), J^2(x, u^1, u^2), \ldots, J^{m-1}(x, u^1, \ldots, u^{m-1})$$

- Allows far more efficient rollout (one-agent-at-a-time). This is just standard rollout for the reformulated problem (so it involves a Newton step)
- The increase in size of the state space does not adversely affect rollout (only one state and its successors are looked at each stage during on-line play)
- Complexity reduction: The one-step lookahead branching factor is reduced from n^m to $n \cdot m$, where n is the number of possible choices for each component u^i

About the Next Lecture

We will discuss special types of problem domains and reformulations, including POMDP, adaptive, and model predictive control

HOMEWORK 3 (DUE IN ONE WEEK):
EXERCISE 1.2 OF THE LATEST VERSION OF THE CLASS TEXTBOOK

READ AHEAD SECTION 1.6 OF THE LATEST VERSION OF THE CLASS TEXTBOOK

This is a good time to watch the summary videolecture at
https://www.youtube.com/watch?v=A7OGgpuRnuo (1-hour version)
of the book
Lessons for AlphaZero for Optimal, Model Predictive, and Adaptive Control
Also the multiagent videolecture at
https://www.youtube.com/watch?v=eqbb6vVlN38