MODEL PREDICTIVE CONTROL AND REINFORCEMENT LEARNING: A UNIFYING FRAMEWORK BASED ON DYNAMIC PROGRAMMING

Dimitri P. Bertsekas
Arizona State University

Lecture at Kyoto IFAC/NMPC, August 2024

Based on my course at ASU on DP/RL (2019-2024), and my recent books

Reinforcement Learning and Optimal Control 2019
Rollout, Policy Iteration, and Distributed Reinforcement Learning, 2020
Abstract Dynamic Programming, 3nd Edition, 2022
Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control, 2022
A Course in Reinforcement Learning, 2023

Outline

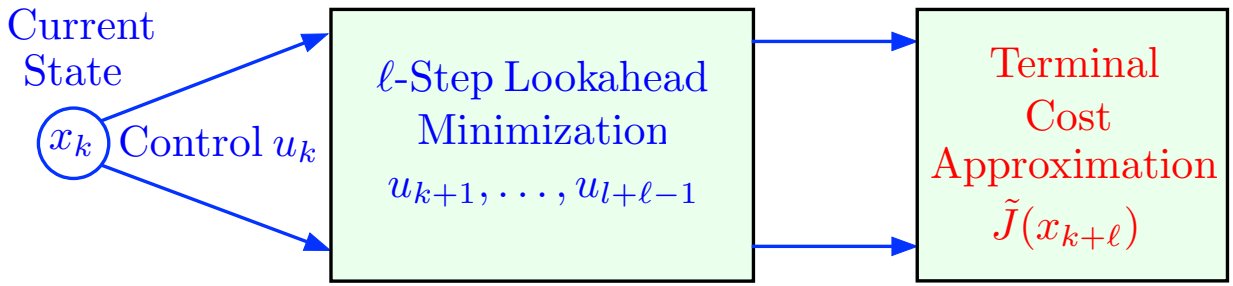
Computer Chess - AlphaZero (2017)

AlphaZero (and most chess programs) involve two algorithms:

- Off-line training of a position evaluator (among others), using deep NNs
- On-line play by multistep lookahead, and position evaluation at the end

- Most attention has been focused on the AlphaZero off-line training, which involves important innovations in NN technology, approximate policy iteration, etc
- The on-line algorithm part is more or less traditional. It is critically important for good performance
- On-line play in computer chess is strongly connected with MPC

Model Predictive Control (MPC): ℓ-Step Lookahead Optimization with Cost Approximation $\tilde{J}$ at the End



At x_k → Solve an ℓ-step lookahead problem with terminal cost $\tilde{J}$

1st ℓ Steps | Future

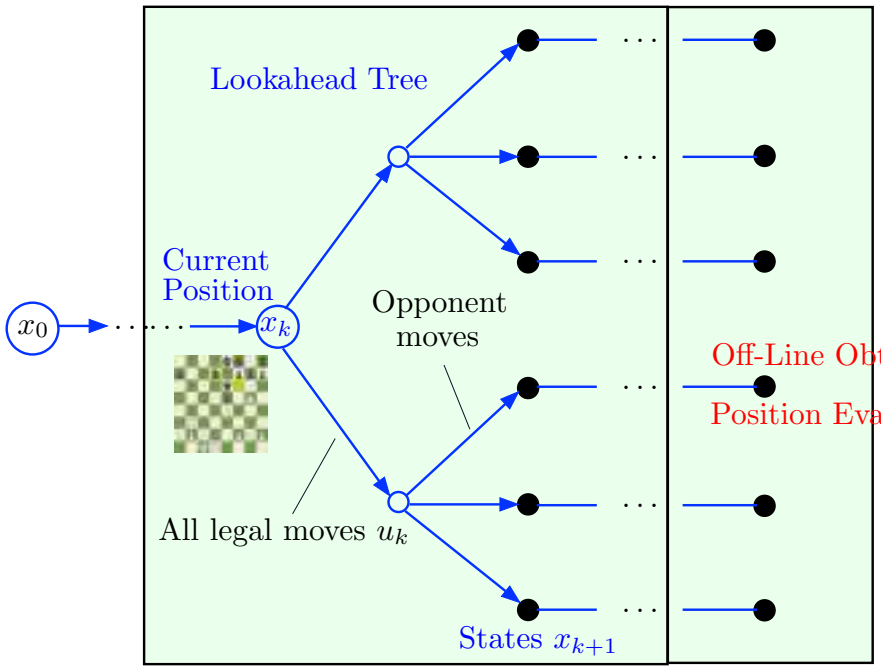
$$\min_{u_k, u_{k+1}, \ldots, u_{k+\ell-1}} \left\{ \sum_{m=0}^{\ell-1} g(x_{k+m}, u_{k+m}) + \tilde{J}(x_{k+\ell}) \right\}$$

Apply the first control $\tilde{u}_k$, discard the remaining controls

Discrete-time deterministic optimal control problem

- Dynamic system equation at time k: $x_{k+1} = f(x_k, u_k)$
- State and control at time k: x_k and u_k
- Cost at stage k: $g(x_k, u_k)$ (≥ 0; can take ∞ values to specify state and control constraints)

On-Line Play Architecture in Computer Chess



It is “isomorphic" to the MPC architecture, except:

- In chess the state and control spaces are discrete, while in MPC they are usually continuous (or mixed discrete-continuous)
- In chess the lookahead tree is usually “pruned", while in MPC the lookahead optimization is usually exact (more on this later)
- The differences are inconsequential: The underlying Newton step theory relies on abstract DP that allows arbitrary state and control spaces, and inexact lookahead
- Another difference is that chess is a two-player game. More on this later, but think of chess against a fixed opponent (this makes chess a one-player game)

Principal Viewpoints of this Talk

- On-line play w/ one-step lookahead is a single step of Newton's method for solving the problem's Bellman equation (similar interpretation applies to multistep lookahead)
- Off-line training provides the starting point for the Newton step
- On-line play is the real workhorse ... off-line training plays a secondary role
- The Newton step framework is very general, because it is couched on abstract DP ideas (arbitrary state and control spaces, stochastic, deterministic, hybrid systems, multiagent systems, minimax, finite and infinite horizon, discrete optimization)
- There is a cultural difference that we will aim to bridge:
  - Reinforcement Learning/AI research is focused largely on off-line training issues
  - Model Predictive Control research is focused largely on on-line play and stability issues
- MPC and RL have strong similarities, and stand to learn a lot from each other

MPC for Linear-Quadratic Problems

We focus on one-dimensional problems for easy visualization (x_k, u_k: scalars)

- System: $x_{k+1} = ax_k + bu_k$, where a, b are given scalars
- Cost: $\sum_{k=0}^{\infty}(qx_k^2 + ru_k^2)$, where q, r are positive scalars
- Basic facts: The optimal cost function is a quadratic function of the state x, and the optimal control policy is a linear function of x

Optimal solution

- Optimal cost function: $J^*(x) = K^*x^2$ where K^* is the unique positive solution of the Riccati equation

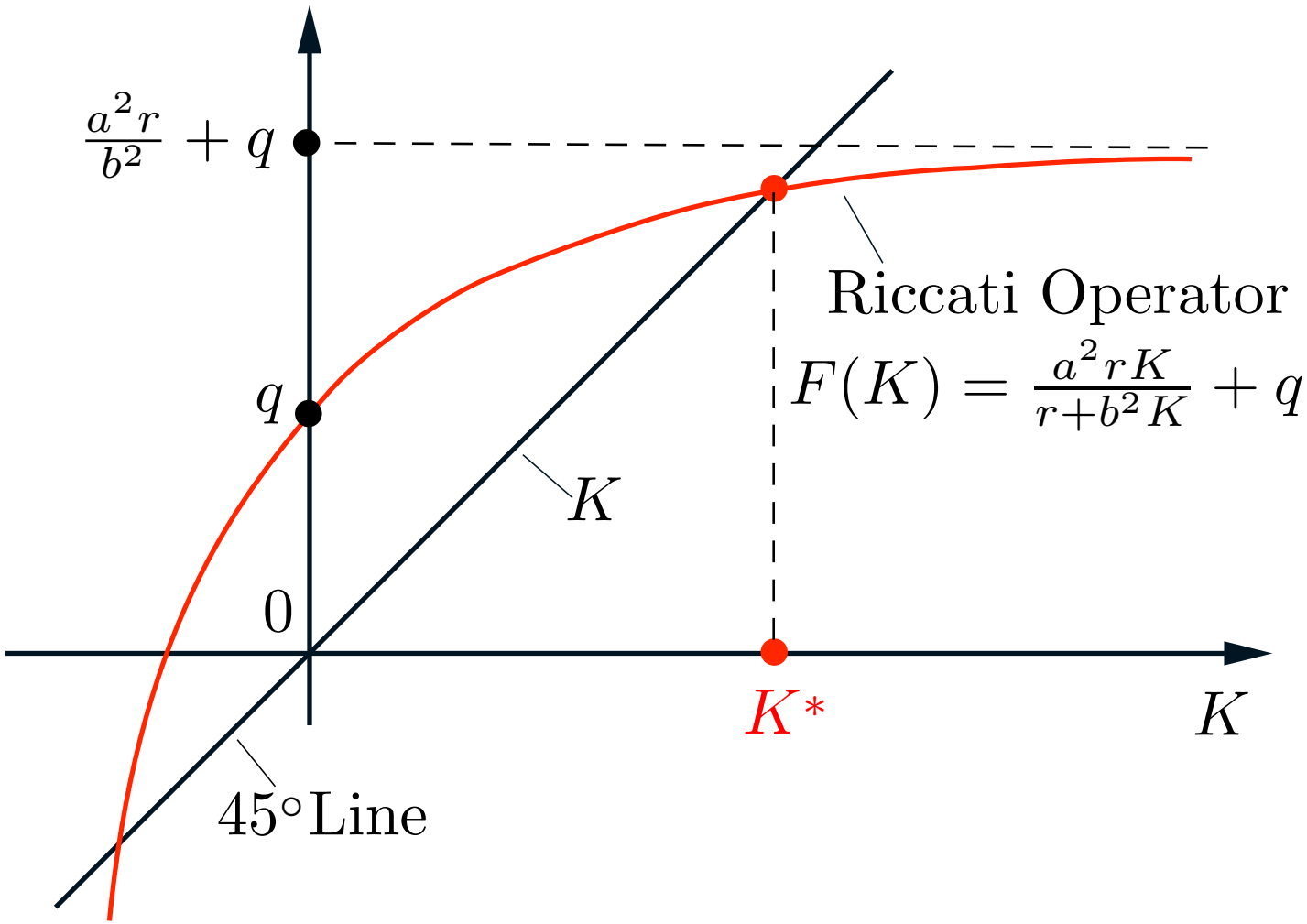
$$K = F(K), \quad \text{where} \quad F(K) = \frac{a^2rK}{r + b^2K} + q \quad \text{is the Riccati operator}$$

- Optimal policy: Linear of the form $\mu^*(x) = L^*x$, where L^* is the scalar given by

$$L^* = -\frac{abK^*}{r + b^2K^*}$$

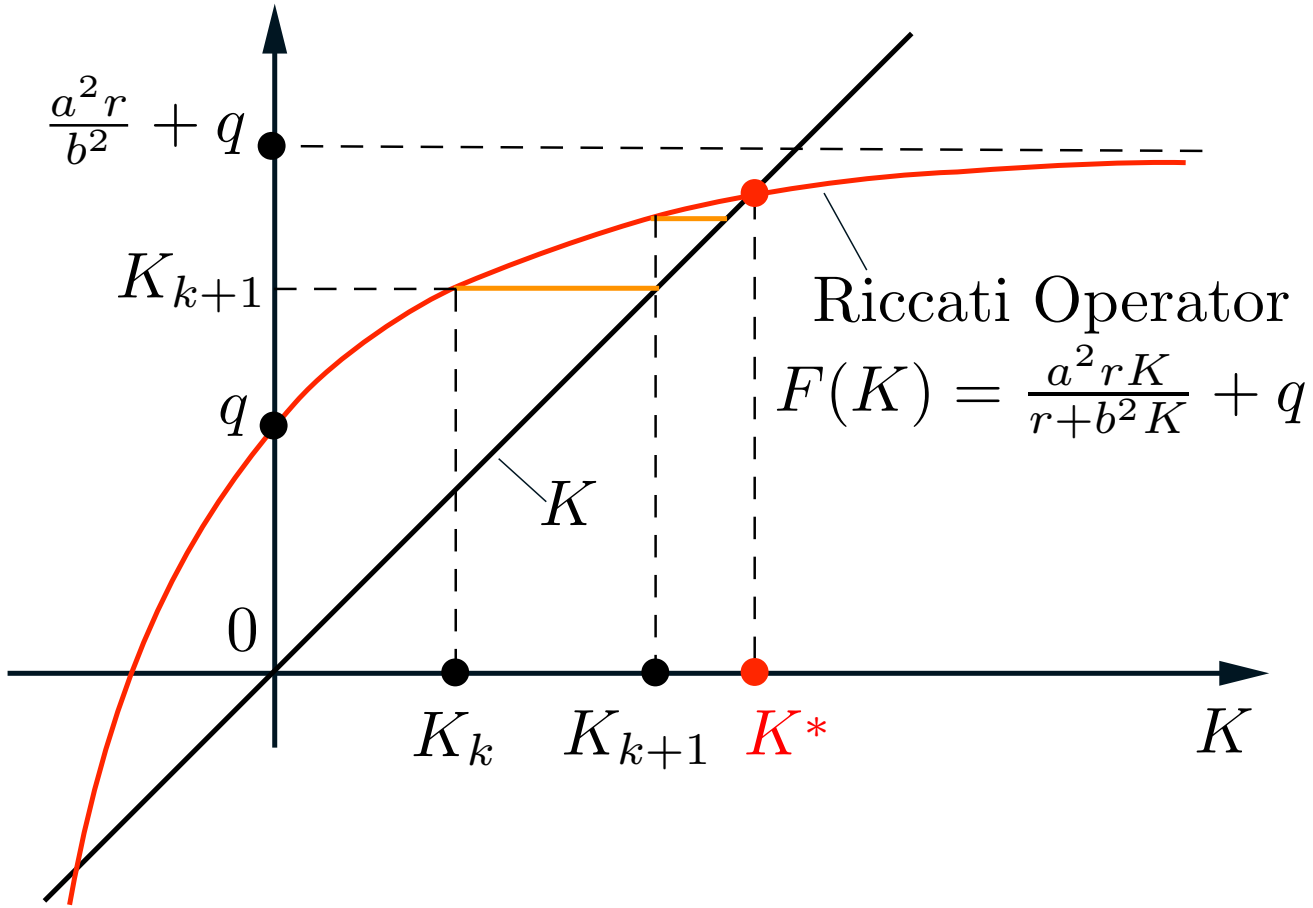
The insights from one dimensional/linear-quadratic problems generalize

Graphical Solution of the Riccati Equation $K = F(K)$



K^* corresponds to the point of intersection of the graph of F with the 45-degree line

Visualization of Riccati Iteration [or Value Iteration (VI)]

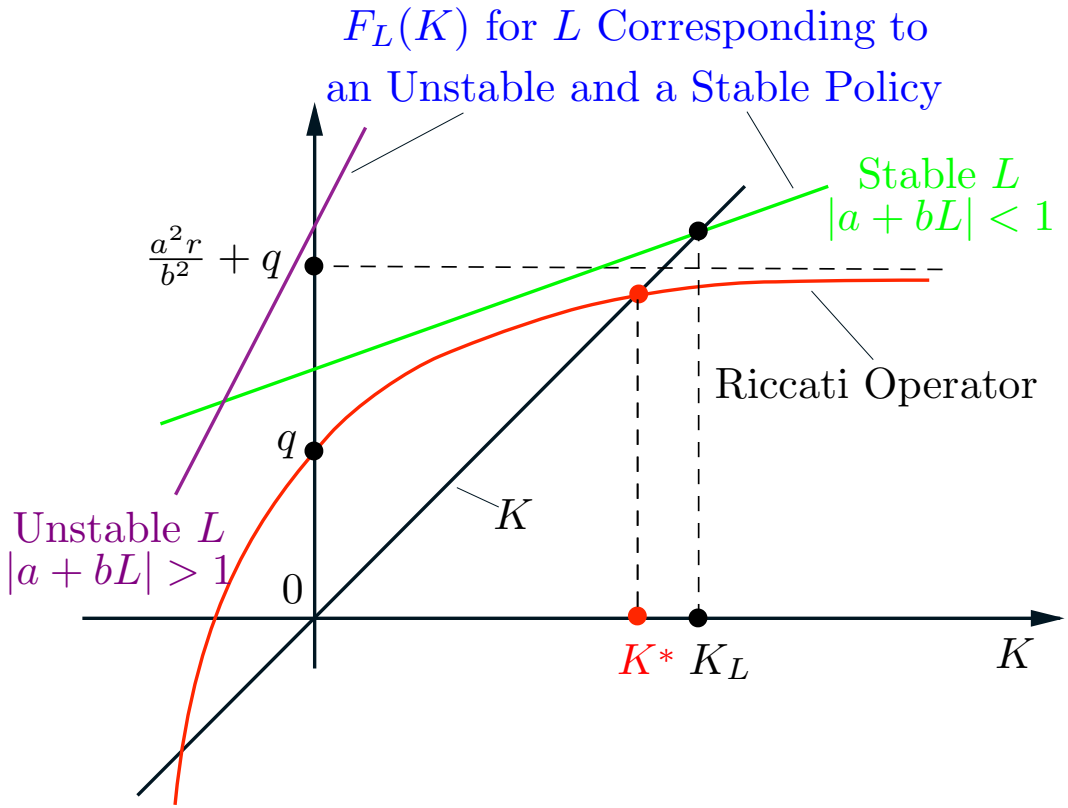


VI generates iteratively the optimal cost functions $J_k(x_k)$ of k-stage problems

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:

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

Visualization of Riccati Equation for Stable and Unstable Linear Policies



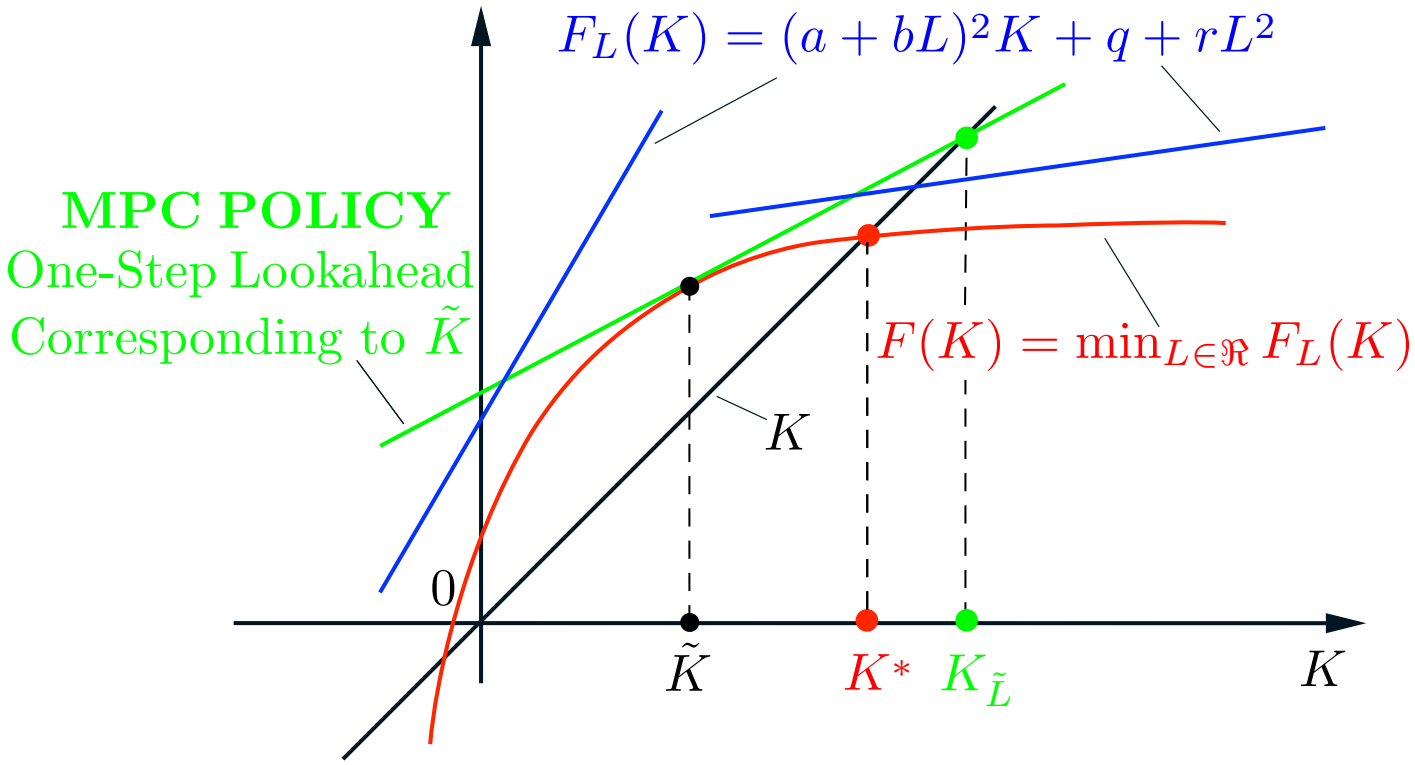
Consider a linear policy $\mu_L(x) = Lx$ and its cost function

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):

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

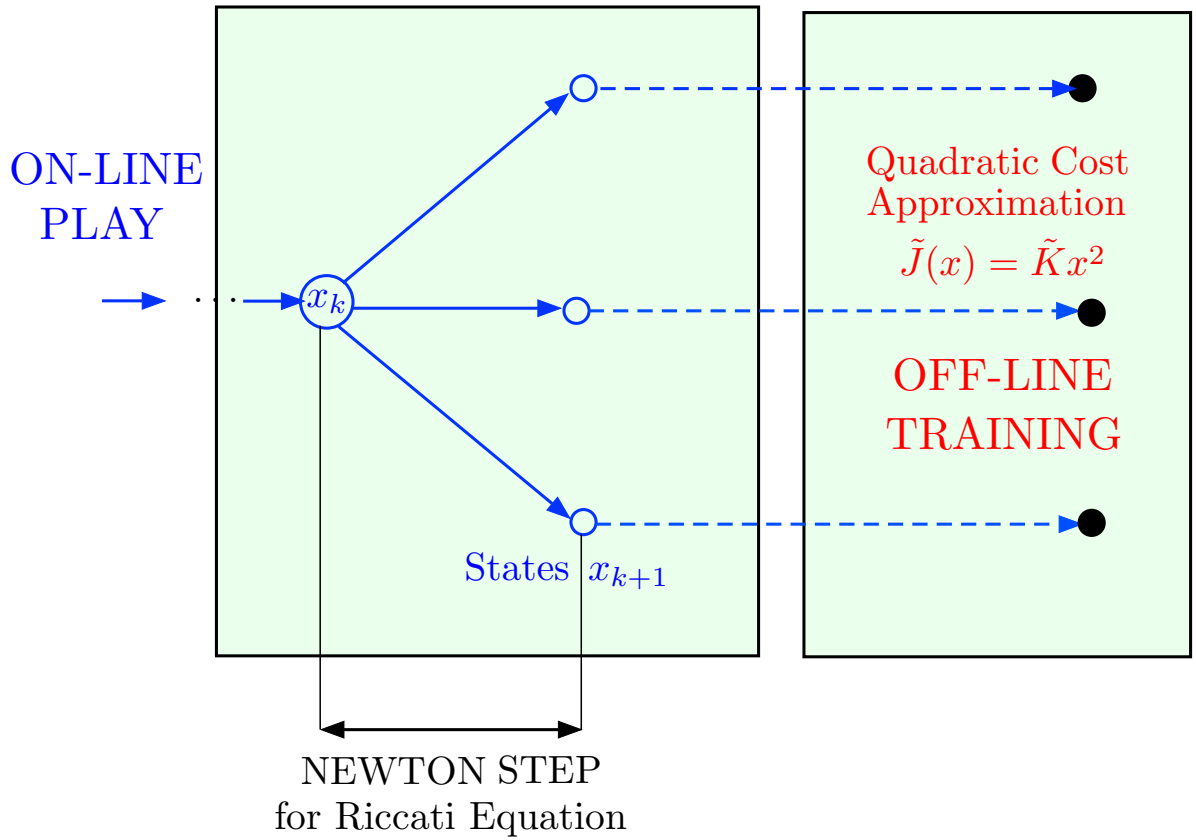
This is only for stable policies (those with $|a+bL|<1$). For unstable policies $K_L = \infty$

Visualization of MPC (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) MPC 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 of MPC (concavity of the Riccati operator is also important)

Preview of the Newton Step Interpretation (One-Step Lookahead)

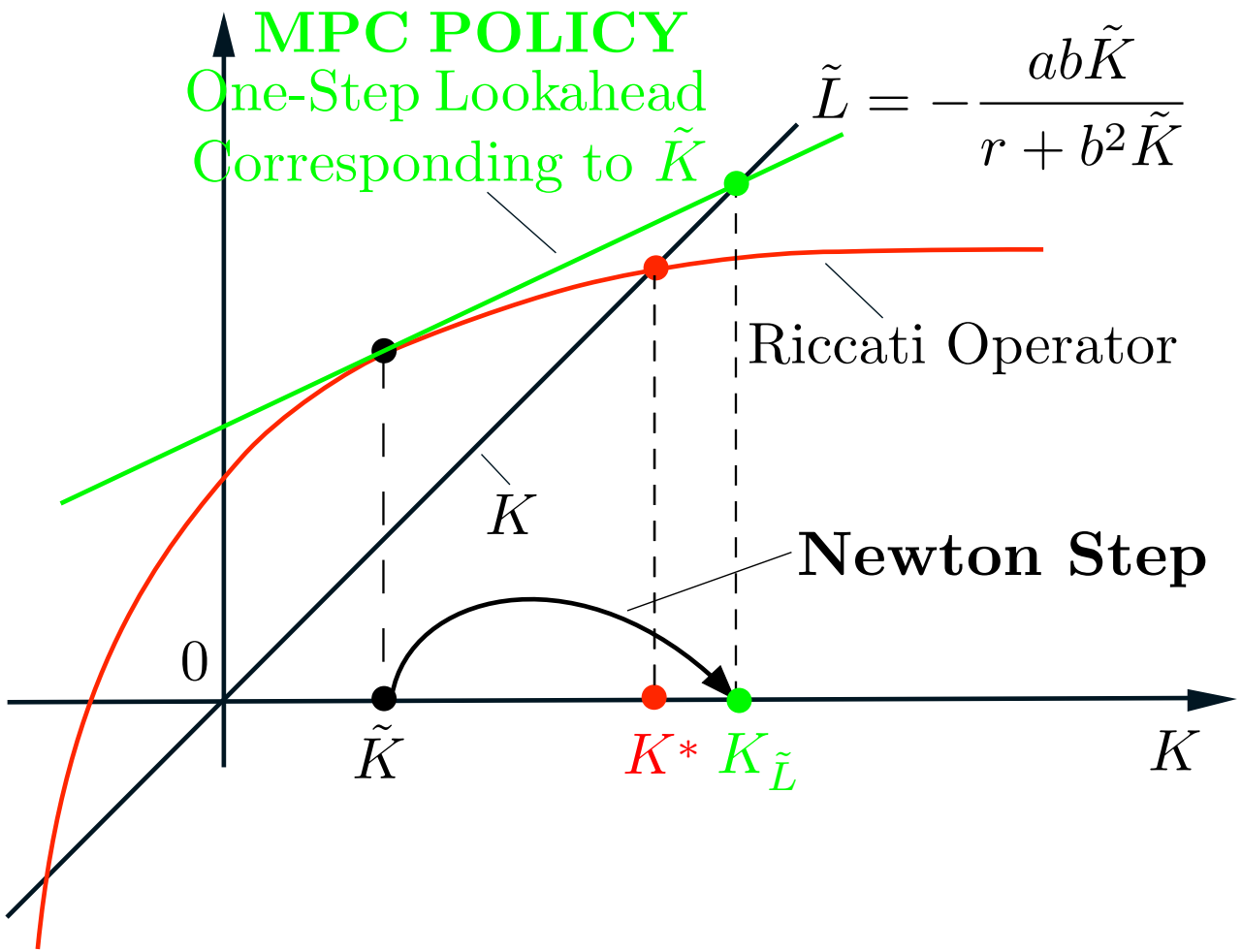


The Newton step with starting point $\tilde{K}$ is the mapping

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

i.e., it relates (superlinearly) cost function approximation with MPC performance

Newton Step Visualization

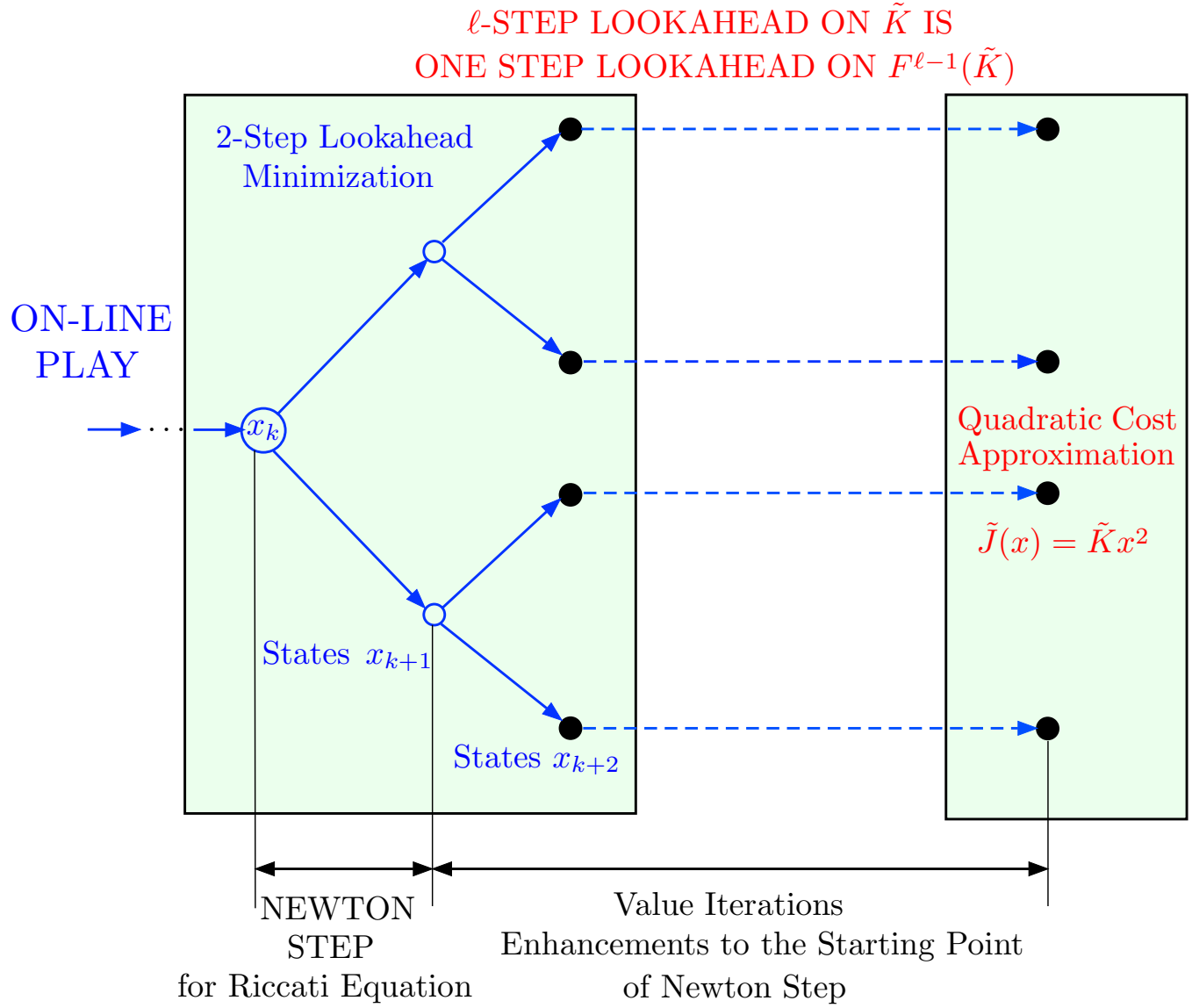


The meaning of superlinear convergence:

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

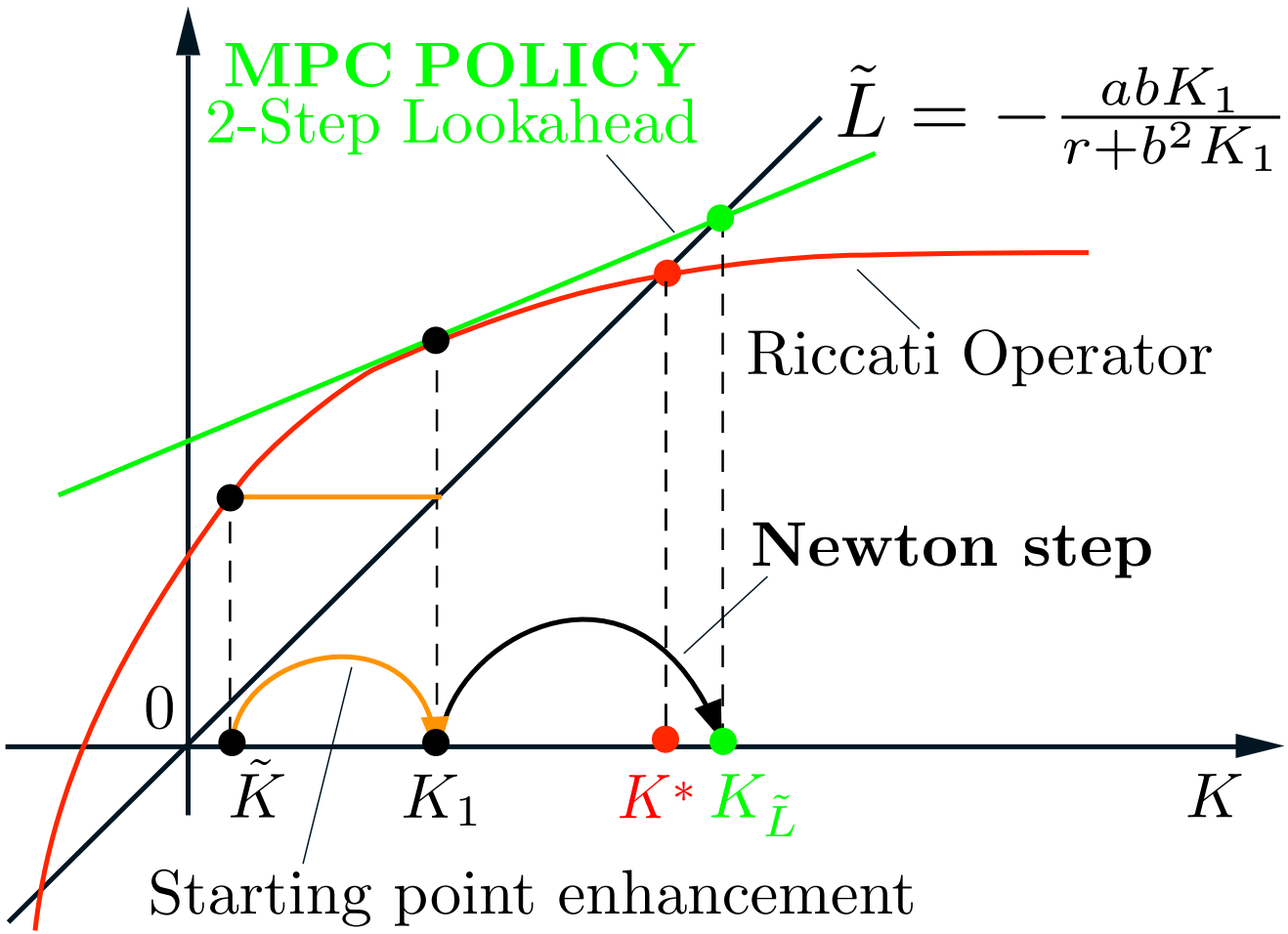
Explains the good performance of MPC in practice

Multistep Lookahead - Preview of the Newton Step



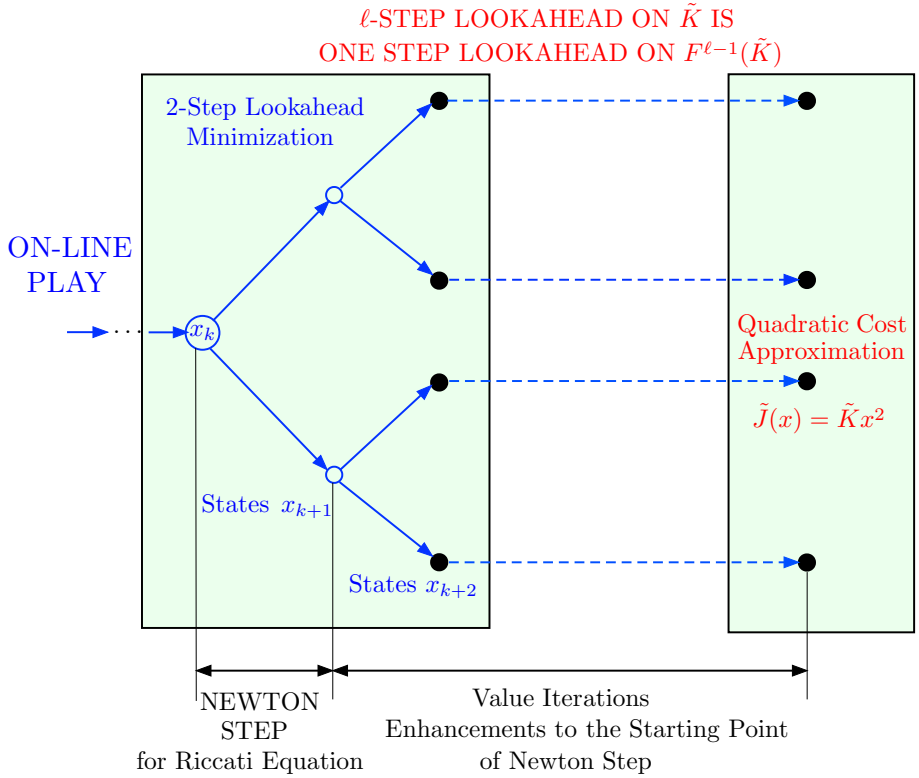
Only the first step of the lookahead is a Newton step

Multistep Lookahead - Illustration of the Newton Step



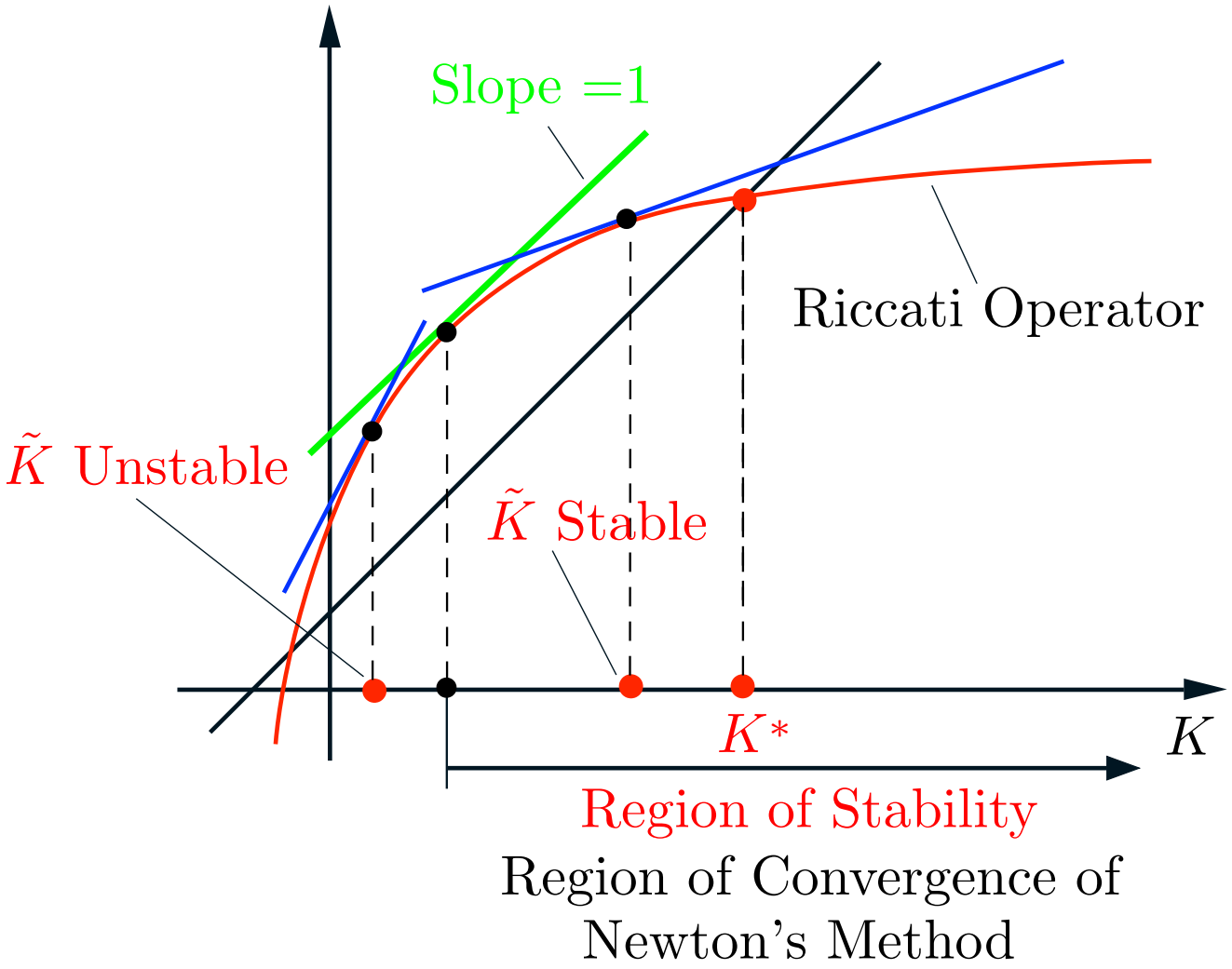
Multistep lookahead brings the starting point of the Newton step closer to K^*

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)

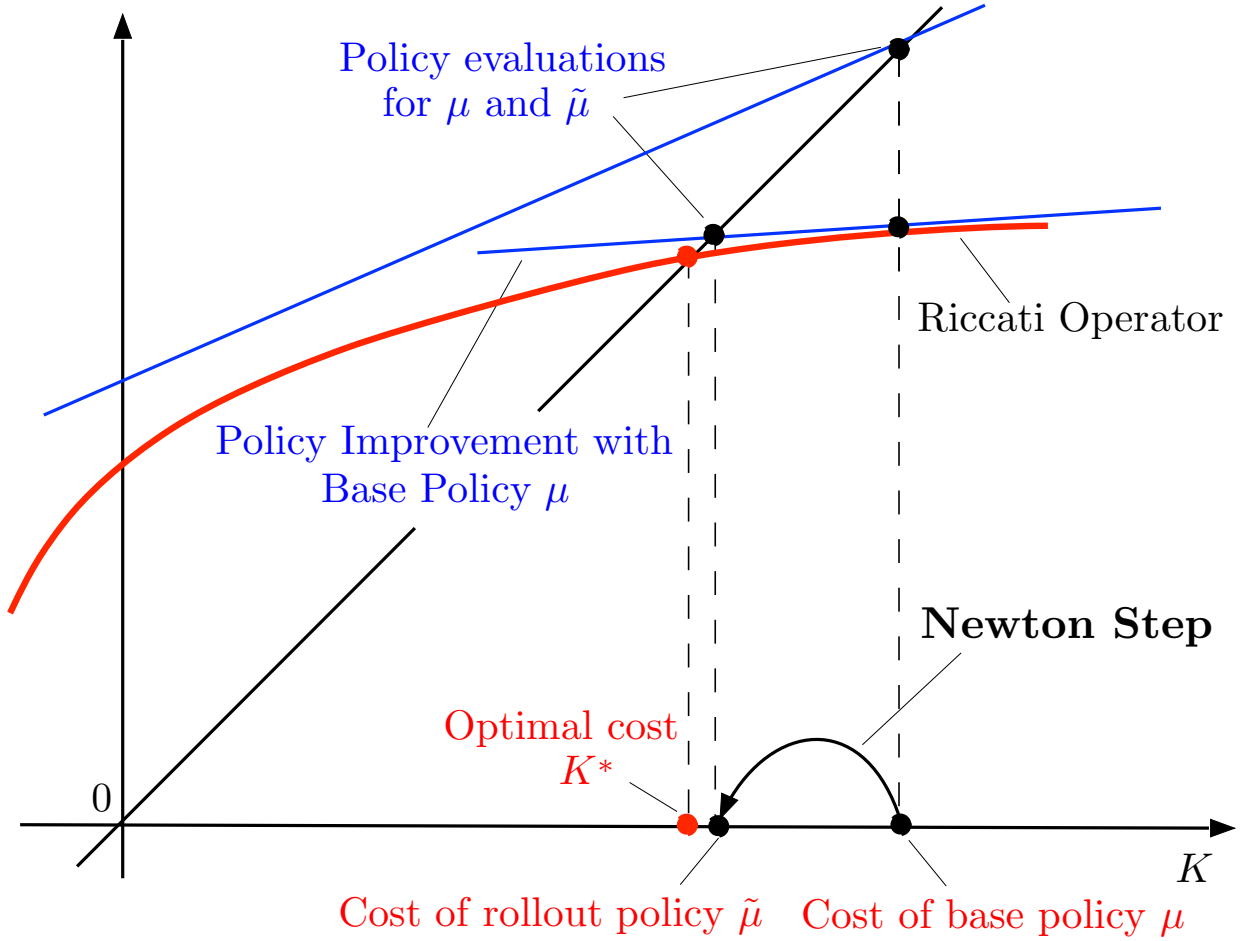
Region of Stability - When Does MPC Produce a Stable Policy



The region of stability expands as the lookahead becomes longer

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

Rollout, Policy Iteration, and Newton's Method



Rollout uses $\tilde{K}$ = the cost coefficient of some linear stable base policy μ

A stable base policy produces a stable rollout policy through a Newton step

Policy iteration is repeated rollout, i.e., Newton's method (Kleinman 1968)

There are several interesting variants of rollout (truncated, fortified, simplified)

Range of Applications of MPC/Rollout - Our Work at ASU

Our Newton step view of MPC applies very broadly, thanks to the generality of the underlying abstract DP theory

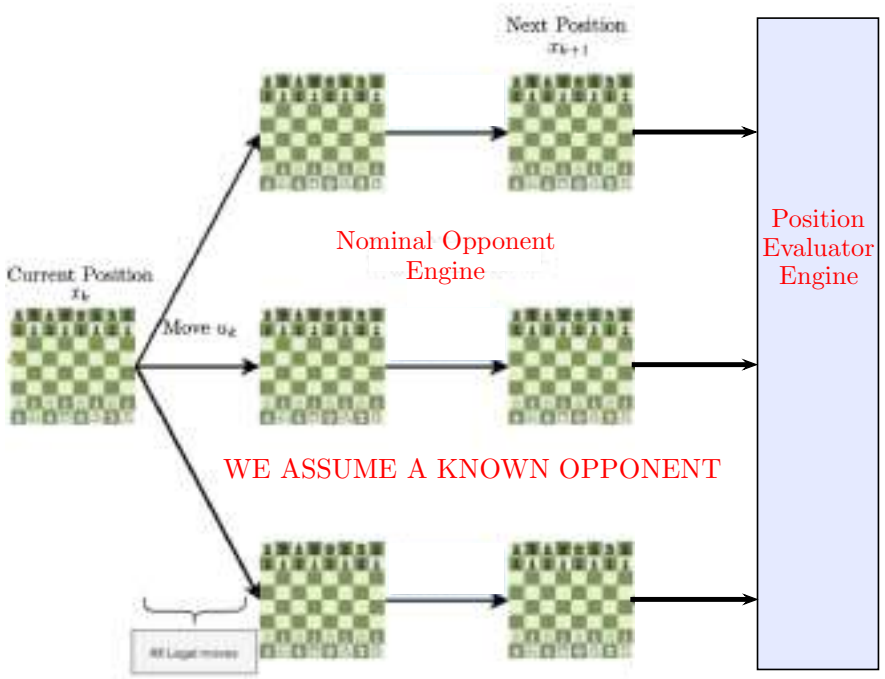
Traditional MPC applications have dealt with continuous state and control problems

Chemical process control, robotics, aerospace, self-driving cars, etc

Many successful applications involve discrete state and control spaces

- Multiagent robotics: Maintenance/repair, taxi fleet management - joint work with Stephanie Gil's group at Harvard (2021-2023)
- Data association and multitarget tracking - Musunuru, Li, Weber, and DPB, 2024
- Sequential inference/decoding problems and the Wordle puzzle - Bhambri, Bhatacharjee, and DPB, 2023)
- Most likely sequence generation in ChatGPT-like transformers, related HHM inference, and Viterbi-rollout algorithm - Li and DPB, 2024
- Computer chess - Gundawar, Li, and DPB, 2024

MPC-MC (MetaChess): An Algorithm for Better Computer Chess (paper posted on arXiv)



We introduce a one-player MPC architecture (the true opponent is approximated by a “nominal" opponent)

We use two available chess engines as components (a meta algorithm)

- The position evaluator engine: Evaluates any given position
- The nominal opponent engine: Predicts the move of the true opponent of MPC-MC (exactly or approximately)
- Each move involves a Newton step starting at the position evaluation function

MPC-MC: Computational Results Using the Stockfish (SK) Engine

We tested two variants of the algorithm

- Standard version
- Fortified version (plays a little better against very strong opponents, a little worse against weaker opponents)

Table: MPC-MC vs SK

SK Strength	Exact. Known Opponent		Approx. Known Opponent	
	Standard	Fortified	Standard	Fortified
0.5 secs	7.5-2.5	8-2	8-2	7-3
2 secs	5-5	5.5-4.5	5.5-4.5	6.5-3.5
5 secs	5-5	5.5-4.5	10-10	10.5-9.5

- We use MPC-MC (one-step lookahead), with SK as both the position evaluator and the nominal opponent, to play against SK
- Similar results obtained with other engines
- We can obtain better results with multistep lookahead
- Parallel computation is essential to reduce the move generation time

Concluding Remarks

- The MPC architecture is the right framework for sequential decision making and approximate DP, provided it makes use of the synergy between on-line play and off-line training
- Using just off-line policy training without on-line play may not work well
  - On-line play uses an exact Newton step (not subject to training errors), and can deal with changing system parameters (a connection with adaptive control)
- Using just on-line play without off-line training misses out on performance
  - Off-line training can produce good starting points for the Newton step, and enhance the stability of the MPC policy
- The role of Newton's method is central - this is a new insight that can guide both analysis and algorithmic design
- Generality: Arbitrary state and control spaces, discrete optimization applications, multiagent problems, etc (see the book references given in the cover slide)
- MPC can be very computationally intensive ... but massive parallel computational power is possible and can allow more sophisticated on-line play strategies
- The cultural divide between RL/AI and control can be bridged: MPC uses a very similar architecture to AlphaZero, and can benefit from RL/AI ideas
- The Decision/Control/MPC community can provide intellectual leadership in the journey ahead, thanks to its mathematical orientation

Thank you!