REINFORCEMENT LEARNING AND MODEL PREDICTIVE CONTROL (MPC)

Dimitri P. Bertsekas
Arizona State University

Lecture at Harvard, June 2025

Based on my course at ASU on DP/RL (2019-2025), and my recent books

Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control, 2022

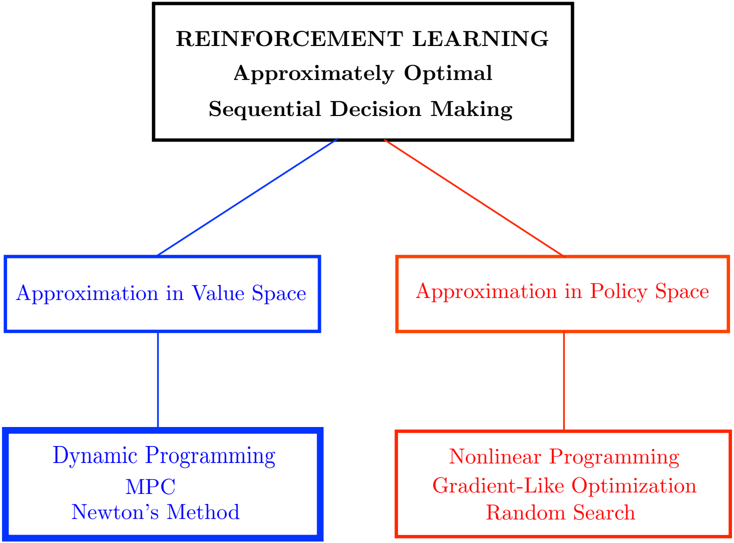
Abstract Dynamic Programming, 3rd edition, 2022

A Course in Reinforcement Learning, 2nd Edition, 2025

also my

RL/MPC Survey paper at IFAC/NMPC, 2024

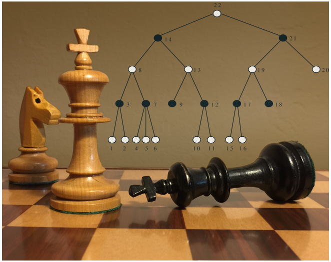
(All can be found on-line at my website)



RL deals with exactly the same mathematical problem as DP

Outline

Computer Chess - AlphaZero (2017)

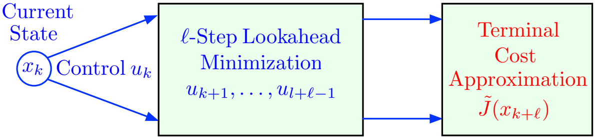


AlphaZero (and most chess programs) involve two algorithms:

- Off-line training of a position evaluator, using deep NNs and self-play
- 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, 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
- Important question: How do the two algorithms connect? (Newton's method)

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



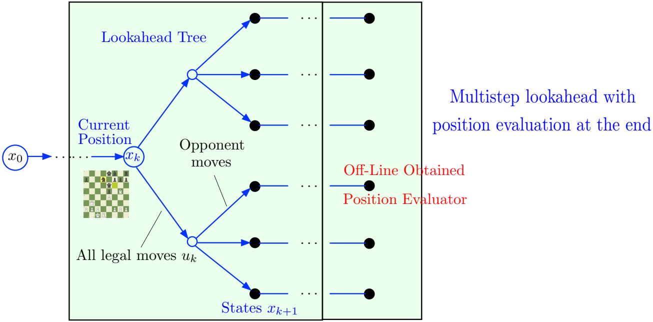
$\boxed{\text{At } x_k}$ $\Rightarrow$ Solve an ℓ-step lookahead problem with terminal cost $\tilde{J}$

$$\min_{u_k, u_{k+1}, \ldots, u_{k+\ell-1}} \Bigg\{ \underbrace{\sum_{m=0}^{\ell-1} g(x_{k+m}, u_{k+m})}_{\text{1st } \ell \text{ Steps}} + \underbrace{\tilde{J}(x_{k+\ell})}_{\text{Future}} \Bigg\}$$

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: $g(x_k, u_k)$



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

- In chess the state and control spaces are discrete, while in MPC they are usually 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: Our Newton step theory allows arbitrary state and control spaces, and inexact lookahead
- Another difference: Chess is a two-player game. More on this later, but think of chess against a fixed (“nominal") 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, finite and infinite horizon, discrete optimization)
- The Newton step framework suggests modifications of on-line play for minimax problems. We suggest a modification based on the concept of a “nominal opponent" and we illustrate it with computer chess

Visualization of MPC for Linear-Quadratic Problems

We consider 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

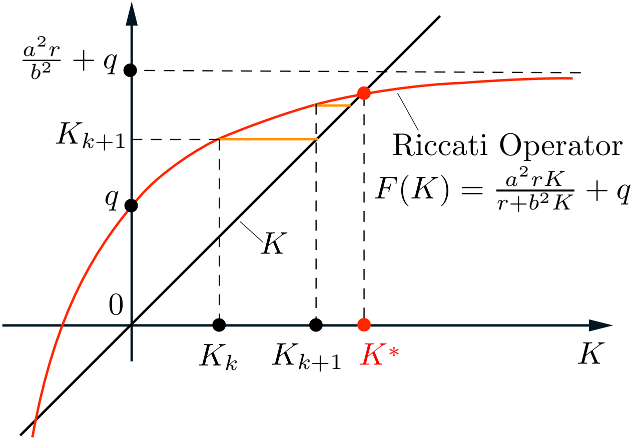
- Optimal cost function: $J^*(x) = K^* x^2$ where K^* is the unique positive solution of the Riccati equation

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

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

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

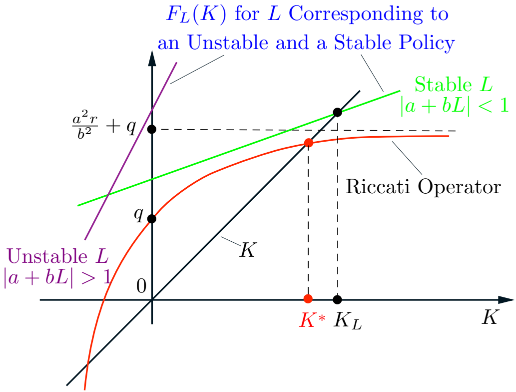
The insights from one dimensional/linear-quadratic problems generalize



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:

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



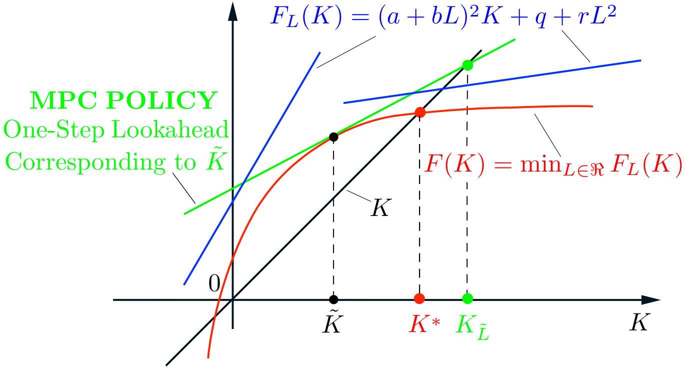
Consider a linear policy $\mu_L(x) = Lx$ and its cost function

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):

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

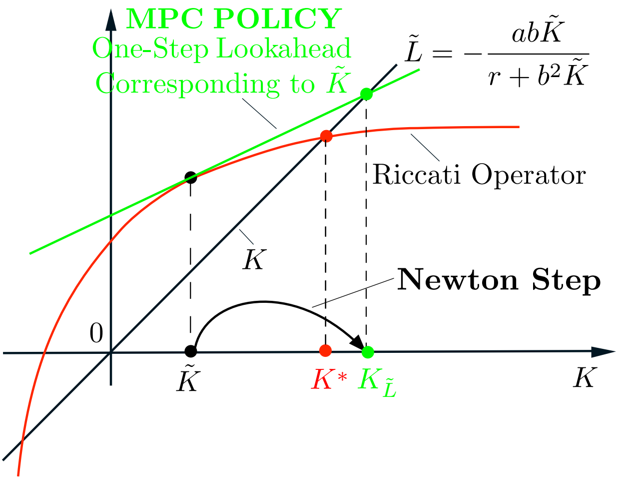
This is only for stable policies (those with $|a + bL| < 1$). For unstable policies $K_L = \infty$

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)

Newton Step Visualization

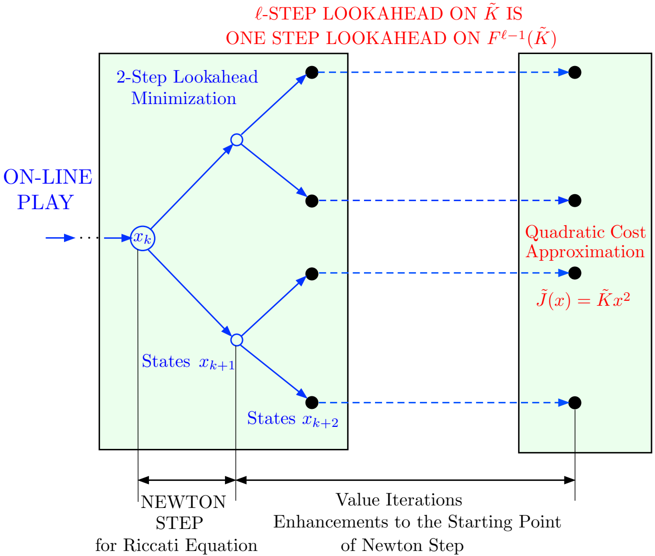


The meaning of superlinear convergence:

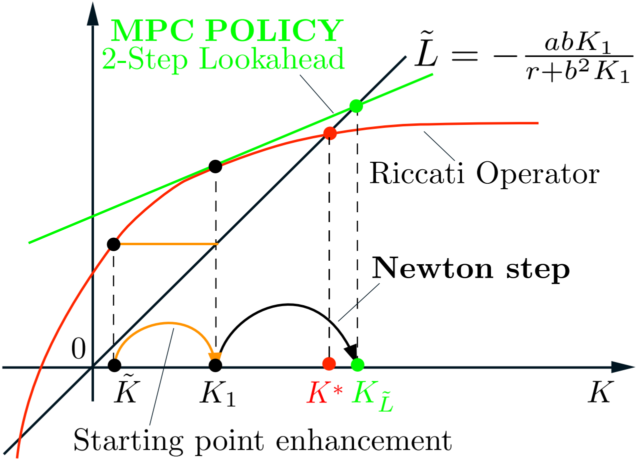
The error $|K_{\tilde{L}} - K^*|$ is MUCH smaller than $|\tilde{K} - K^*|$

Explains the good performance of MPC in practice

Multistep Lookahead - Preview of the Newton Step

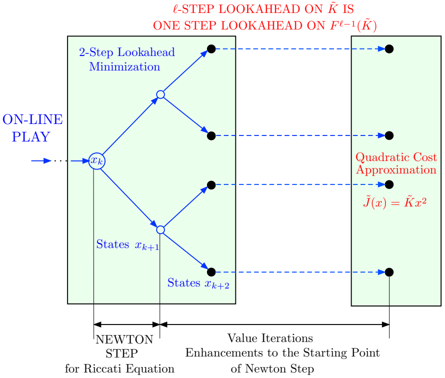


Only the first step of the lookahead is a Newton step



Multistep lookahead 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)

The Critical Mapping in Summary - One-Step Lookahead

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

First Step "Future"

At state x $\rightarrow$ $\min_{u \in U(x)} \left\{ g(x,u) + \tilde{J}\big(f(x,u)\big) \right\}$

CRITICAL MAPPING

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

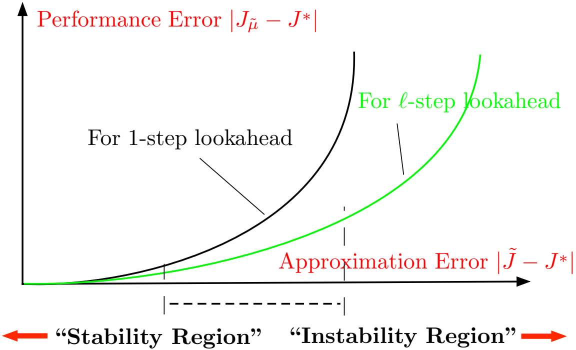
Performance Error $|J_{\tilde{\mu}} - J^*|$

Approximation Error $|\tilde{J} - J^*|$

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 Critical Mapping is a Superlinear Newton Step



The key fact: The critical mapping is superlinear; it is a Newton Step

- Convergence threshold defined by the region of convergence of Newton's method
- This has far-reaching implications for both theory and practice
- The error $|\tilde{J} - J^*|$ primarily depends on the limitations of the cost function approximation method/neural net!
- Inside the two regions, better training/more data, improving confidence intervals, etc, have marginal effect

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

We have considered applications involving discrete state and control spaces

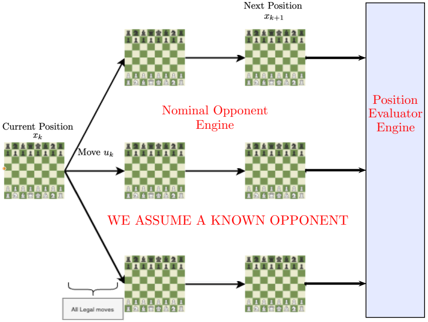
- Discrete/combinatorial optimization (e.g., traveling salesman, vehicle routing)
- Multiagent robotics: Maintenance/repair, taxi fleet management - joint work with Stephanie Gil's group at Harvard (2021-2023)
- Data association and multitarget tracking - Musunuru, Li, Weber, and DPB, 2024
- Sequential inference/decoding problems and the Wordle puzzle - Bhambri, Bhatacharjee, and DPB, 2023
- Most likely sequence generation in ChatGPT-like transformers, related HMM inference, and Viterbi-rollout algorithm - Li and DPB, 2024

A new idea for minimax problems:

A special theoretical difficulty: The Bellman operator for minimax is not concave

- This motivates approximation in value space and maximizer approximation in policy space
- Replace the maximizer by a trained “nominal opponent". This converts the two-player problem to a one-player problem that we address with MPC
- Illustration for computer chess, cf. ArXiv paper by Gundawar, Li, and DPB, 2024

MPC-MC (MetaChess): An MPC Architecture for Computer Chess



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

MPC-MC: Computational Results Using the Stockfish (SK) Base Engine Similar Results Using a Transformer Engine (DeepMind)

Tests with two variants of the algorithm:

- Standard and Fortified
- The latter plays a little better against strong opponents, and a little worse against weak 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
- Better results with other engines. (Much better for the transformer engine)
- Better results with multistep lookahead
- Parallel computation is essential to reduce the move generation time

Thank you!