Introduction to Optimal Control

ORF523 CONVEX AND CONIC OPTIMIZATION
SUMEET SINGH, GOOGLE BRAIN ROBOTICS
APRIL 22, 2021
Outline

• Optimal Control Problem
• Open- vs Closed-Loop Solutions

• Closed-Loop: Bellman’s Principle of Optimality & Dynamic Programming
  • Finite spaces
  • Continuous spaces – LQ control

• Open-Loop:
  • Gradient descent
  • Newton descent
  • DDP

• Model Predictive Control
Feedback Control

- Consider block diagram for tracking some reference signal.
Feedback Control

- Consider block diagram for tracking some reference signal.
Feedback Control

- Consider block diagram for tracking some reference signal.
Feedback Control Objectives

• *Stability*: various formulations; loosely, system output is “under control”

• *Tracking*: output should track reference “as close as possible”

• *Disturbance rejection*: output should be “as insensitive as possible” to disturbances/noise

• *Robustness*: controller should still perform well up to “some degree of model misspecification”

Adapted from: Stanford AA203, Spring 2021
What’s Missing?

- **Performance**: some mathematical quantification of all these objectives and control that realizes the tradeoffs

- **Planning**: providing an appropriate *reference trajectory* to track (can be highly non-trivial)

- **Learning**: adaptation to unknown properties of the system

Adapted from: Stanford AA203, Spring 2021
Light Statistics

Top Speed: 1.93 m/s
Max Drag: 0.55 m/s²
What’s Missing?

- **Performance**: some mathematical quantification of all these objectives and control that realizes the tradeoffs

- **Planning**: providing an appropriate reference trajectory to track (can be highly non-trivial)

- **Learning**: adaptation to unknown properties of the system

Adapted from: Stanford AA203, Spring 2021
Optimal Control Problem

3 Key Ingredients:

• Mathematical description of the system to be controlled

• Specification of a performance criterion

• Specification of constraints
State-Space Models

\[
\begin{align*}
\dot{x}_1(t) &= f_1(x_1(t), x_2(t), \ldots, x_n(t), u_1(t), u_2(t), \ldots, u_m(t), t) \\
\dot{x}_2(t) &= f_2(x_1(t), x_2(t), \ldots, x_n(t), u_1(t), u_2(t), \ldots, u_m(t), t) \\
&\vdots \quad \vdots \\
\dot{x}_n(t) &= f_n(x_1(t), x_2(t), \ldots, x_n(t), u_1(t), u_2(t), \ldots, u_m(t), t)
\end{align*}
\]

Where

- \(x_1(t), x_2(t), \ldots, x_n(t)\) are the state variables
- \(u_1(t), u_2(t), \ldots, u_m(t)\) are the control variables

Adapted from: Stanford AA203, Spring 2021
State-Space Models

In compact form:

\[ \dot{x}(t) = f(x(t), u(t), t) \]

• A history of control input values during the interval [0, T] is called a control history, denoted by \( u \)
• A history of state values during the interval [0, T] is called a state trajectory, denoted by \( x \)
Illustrative Example

- Double integrator: point mass under controlled acceleration

\[
\begin{bmatrix}
\ddot{s}(t) \\
\dot{s} \\
\dot{v}
\end{bmatrix} = \begin{bmatrix} a(t) \\
v \\
\dot{a}
\end{bmatrix}
\]
Illustrative Example

- Double integrator: point mass under controlled acceleration

\[
\begin{bmatrix}
\dot{s} \\
\dot{v}
\end{bmatrix} =
\begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
s \\
v
\end{bmatrix} +
\begin{bmatrix}
0 \\
1
\end{bmatrix} [a]
\]

\[
\dot{x}(t) = A \ x(t) + B \ u(t)
\]

\[
\begin{bmatrix}
\dot{s} \\
\dot{v}
\end{bmatrix} =
\begin{bmatrix}
0 & I \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
s \\
v
\end{bmatrix} +
\begin{bmatrix}
0 \\
I
\end{bmatrix} [a]
\]

Adapted from: Stanford AA203, Spring 2021
Quantifying Performance

\[
\min_{\mathbf{u}} \int_0^T \| \mathbf{x}(t) \|^2 + \| \mathbf{u}(t) \|^2 \, dt
\]

s.t. \quad \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)
\quad \mathbf{x}(0) = \mathbf{x}_0

Adapted from: Stanford AA203, Spring 2021
Quantifying Performance

\[
\min_u \int_0^T \| x(t) \|^2 + \| u(t) \|^2 \, dt \\
\text{s.t.} \quad \dot{x}(t) = A x(t) + B u(t) \\
\quad x(0) = x_0 \\
\quad x(T) = x_f
\]

Adapted from: Stanford AA203, Spring 2021
Quantifying Performance

\[
\min_u \quad \int_0^T x(t)^T Q x(t) + u(t)^T R u(t) \, dt
\]

s.t. \quad \dot{x}(t) = A x(t) + B u(t)
\quad x(0) = x_0
\quad x(T) = x_f
Quantifying Performance

• More generally:

\[ J(u, x) = \int_0^T l(t, u(t), x(t)) dt + \phi(x(T)) \]

Instantaneous or stage-wise cost

Terminal Cost

• \( l \) and \( \phi \) are scalar functions, and \( T \) may be specified or “free”
Constraints

• Initial and final conditions (boundary conditions):

\[ x(0) = x_0 \quad x(T) = x_f \]

• Constraints on state trajectory:

\[ X \leq x(t) \leq \bar{X} \]

• Control limits:

\[ U \leq u(t) \leq \bar{U} \]

• A control history and state trajectory that satisfy the control & state constraints for the entire time interval are termed *admissible*
The Optimal Control Problem

Definitions: State: \( x \in \mathbb{R}^n \), Control: \( u \in \mathbb{R}^m \)

Continuous Time:

- Performance measure (minimize): \( J(u, x) = \int_0^T l(t, u(t), x(t)) dt + \phi(x(T)) \)
- Dynamics: \( \dot{x}(t) = f(x(t), u(t), t) \)
- + other constraints, e.g., \( u(t) \in U, x(t) \in X \)
- \( T < \infty \) or \( T = \infty \)

Minimizer: \((x^*, u^*)\) is an optimal solution pair.
Existence & uniqueness not always guaranteed
The Optimal Control Problem

Discrete Time:

- Performance measure (minimize): \( J(u, x) = \sum_{n=0}^{N-1} l(n, u[n], x[n]) + \phi(x[N]) \)

- Dynamics: \( x[n + 1] = f_d(x[n], u[n], n) \)

- + other constraints

- \( N < \infty \) or \( N = \infty \)
Solution Methods

Dynamic Programming (Principle of Optimality)
- Compositionality of optimal paths
- **Closed-loop** solutions: find a solution for all states at all times

Calculus of Variations (Pontryagin Maximum/Minimum Principle)
- “Optimal curve should be such that neighboring curves don’t lead to smaller costs” → “Derivative = 0”
- **Open-loop** solutions: find a solution for a given initial state

Figures: [Kelly, 2017]
The Optimal Control Problem

Continuous Time:

- Performance measure (minimize): 
  \[ J(u, x) = \int_0^T l(t, u(t), x(t)) \, dt + \phi(x(T)) \]

- Dynamics: 
  \[ \dot{x}(t) = f(x(t), u(t), t) \]

+ other constraints, e.g., \( u(t) \in U, \ x(t) \in X \)

Closed-loop: find policy function \( \pi^*(x, t) \) s.t. \( u^*(t) = \pi^*(x(t), t) \)

Open-loop: given \( x(0) = x_0 \), find optimal signals: \( (x^*, u^*) \), i.e., functions in \( W^{1,\infty}[0, T] \) and \( L^{\infty}[0, T] \)
The Optimal Control Problem

Discrete Time:

• Performance measure (minimize): $$J(u, x) = \sum_{n=0}^{N-1} l(n, u[n], x[n]) + \phi(x[N])$$

• Dynamics: $$x[n + 1] = f_d(x[n], u[n], n)$$

• + other constraints

**Closed-loop**: find policy functions: \{\pi_0^*, ..., \pi_{N-1}^*\}, s.t. $$u^*[n] = \pi_n^*(x[n])$$

**Open-loop**: Given $$x[0] = x_0$$, find optimal sequences: $$(x^*[\cdot], u^*[\cdot])$$. 
Optimal Control

- Open Loop
- Closed Loop
Optimal Control

- Open Loop
- Closed Loop
  - Principle of Optimality
Principle of Optimality

Given trajectory from $a \rightarrow c$, with cost $J_{ac} = J_{ab} + J_{bc}$ minimal, then $J_{bc}$ minimal for path $b \rightarrow c$.

Proof by contradiction:

- Assume there exists an alternative path $b \rightarrow c$ with lower cost $\tilde{J}_{bc} < J_{bc}$. Then, $\tilde{J}_{ac} = J_{ab} + \tilde{J}_{bc} < J_{ab} + J_{bc} = J_{ac}$, i.e., original path was not minimal.

[Bertsekas, 2017]
Theorem (Discrete-time Principle of Optimality: Deterministic Case). Let \( \pi^* = (\pi_0^*, \ldots, \pi_{N-1}^*) \) be an optimal policy. Assume state \( x_k \) is reachable. Consider the subproblem whereby we are at \( x_k \) at time \( k \) and we wish to minimize the cost-to-go from time \( k \) to time \( N \). Then the truncated policy \( (\pi_k^*, \ldots, \pi_{N-1}^*) \) is optimal for the subproblem.

Tail policies of an optimal policy are optimal for tail sub-problems.
Applying Principle of Optimality

- Principle of Optimality: If $b - c$ is the initial segment of the optimal path from $b - f$, then $c - f$ is the terminal segment of this path.

- Thus, the optimal trajectory is found by comparing:

\[
\begin{align*}
C_{bcf} &= J_{bc} + J_{cf}^* \\
C_{bdf} &= J_{bd} + J_{df}^* \\
C_{bef} &= J_{be} + J_{ef}^*
\end{align*}
\]
Applying Principle of Optimality

• Need only to compare *concatenation of immediate decisions* with *optimal* decisions

• In practice: carry out backwards in time.
Dynamic Programming (DP)

Performance measure:
$$ J(u, x) = \sum_{n=0}^{N-1} l(n, u[n], x[n]) + \phi(x[N]) $$

Dynamics:
$$ x[n + 1] = f_d(x[n], u[n], n) $$

Dynamic Programming recursion (Bellman Recursion) proceeds backwards:
$$ J_N^*(x[N]) = \phi(x[N]) $$
$$ J_n^*(x[n]) = \min_u [l(n, u, x[n]) + J_{n+1}^*(f_d(x[n], u, n))] \quad n = N - 1, \ldots, 0 $$

Optimization over sequence ➔ sequence of one-step optimizations
**DP – Stochastic Case**

Stochastic dynamics:  \( \mathbf{x}[n + 1] = f_d(\mathbf{x}[n], u[n], n, \omega[n]) \)

Markovian assumption:  \( \omega[n] = \omega[n](\mathbf{x}[n], u[n]) \)

i.e., disturbance at time \( n \) is only a function of the state and control at time \( n \)

**Implications:**
1. Distribution of next state depends only on current state and control:  \( \mathbf{x}[n + 1] \sim P(\cdot | \mathbf{x}[n], u[n]) \)

2. Sufficient to look for optimal policy at time \( n \) as a function of \( \mathbf{x}[n] \) (and not as a function of the entire history before time \( n \))
DP – Stochastic Case

Stochastic dynamics with Markovian property: \( \mathbf{x}[n + 1] = f_d(\mathbf{x}[n], \mathbf{u}[n], n, \omega[n]) \)

Performance:

\[
\mathbb{E}_{\omega_{0:N-1}} \left[ \sum_{n=0}^{N-1} l(n, \pi_n(\mathbf{x}[n]), \mathbf{x}[n]) + \phi(\mathbf{x}[N]) \right]
\]

Applying principle of optimality and exploiting linearity of expectation:

\[
J^*_N(\mathbf{x}[N]) = \phi(\mathbf{x}[N])
\]

\[
J^*_n(\mathbf{x}[n]) = \min_{\mathbf{u}} \mathbb{E}_{\omega[n]} \left[ l(n, \mathbf{u}, \mathbf{x}[n]) + J^*_{n+1}(f_d(\mathbf{x}[n], \mathbf{u}, n, \omega[n])) \right] \\
\text{for } n = N - 1, \ldots, 0
\]
DP – Inventory Control Example

• Stochastic DP

• Stock available $x[n] \in \mathbb{N}$, order $u[n] \in \mathbb{N}$, demand $w[n] \in \mathbb{N}$

• Dynamics: $x[n+1] = \max(0, x[n] + u[n] - w[n])$

• Constraints: $x[n] + u[n] \leq 2$

• Simple stationary demand model: $p(w[n] = 0) = 0.1, p(w[n] = 1) = 0.7, p(w[n] = 2) = 0.2$

• Objective:

$$
\mathbb{E} \left[ 0 + \sum_{n=0}^{2} (u[n] + (x[n] + u[n] - w[n])^2) \right]
$$

- No terminal cost
- Cost to purchase
- Lost business/over-supply cost
DP Algorithm:

\[ J_n^*(x[n]) = \min_{0 \leq u[n] \leq 2-x[n]} \mathbb{E}_{w[n]} \left[ u[n] + (x[n] + u[n] - w[n])^2 + J_{n+1}^*(\max(0, x[n] + u[n] - w[n])) \right] \]

As an example:

\[ J_2^*(0) = \min_{u \in \{0,1,2\}} \mathbb{E}_{w[2]} \left[ u + (u[2] - w[2])^2 \right] \]

\[ = \min_{u \in \{0,1,2\}} \left[ u + 0.1u^2 + 0.7(u - 1)^2 + 0.2(u - 2)^2 \right] \]

Thus: \( J_2^*(0) = 1.3, \pi_2^*(0) = 1 \). Show: \( J_0^*(0) = 3.7, J_0^*(1) = 2.7, J_0^*(2) = 2.818 \)
DP in Discrete Spaces

Notice: 

\[ J_n^*(x[n]) = \min_u [l(n, u, x[n]) + J_{n+1}^*(f_d(x[n], u, n))] \]

Recursion needs solution for all possible next states.

- Doable for finite/discrete state-spaces (e.g., grids).
- Suffers from curse of dimensionality (e.g., consider quantizing a continuous state-space).

Value Iteration:

- Set up a recursion: 
  \[ J_n(x) \leftarrow \min_u [l(n, u, x) + J_{n+1}(f_d(x, u, n))] \]
  for all \( x \).

- Infinite horizon setting \( \rightarrow \) drop the time dependence, and iterate until convergence.

Generalized Policy Iteration:

- Interleave policy evaluation (similar recursion with \( \min \) replaced with policy), and policy improvement (argmin of Bellman formula with current value estimate).
DP in Continuous Spaces

Rarely, we have exact solution in continuous spaces. Otherwise: need function approximation: *Approximate Dynamic Programming*

Examples:
- *Fitted Value Iteration*: bootstrap off current/delayed estimate of value function to compute “targets” and regress.
- *Meshes*: perform iteration on a discrete mesh and use interpolation

**Dynamics unknown**: *Reinforcement Learning*: find optimal policy and value function using samples of experience \((x, u, x', c)\).
- Algorithms resemble stochastic approximations of recursion formulas (+tricks)
Optimal Control

- Open Loop
- Closed Loop
  - Principle of Optimality
    - Discrete-time: DP
    - Continuous-time: HJB
Optimal Control

- Open Loop
- Closed Loop
  - Principle of Optimality
    - Discrete-time: DP
    - Continuous-time: HJB
  - Application to Linear Dynamics
DP For LQ Control

Linear time-varying dynamics: \( \mathbf{x}[n + 1] = A_n \mathbf{x}[n] + B_n \mathbf{u}[n] \)

Quadratic time-varying cost:

\[
J(u, x) = \frac{1}{2} \mathbf{x}[N]^T Q_N \mathbf{x}[N] + \frac{1}{2} \sum_{0}^{N-1} (\mathbf{x}[n]^T Q_n \mathbf{x}[n] + \mathbf{u}[n]^T R_n \mathbf{u}[n] + 2 \mathbf{x}[n]^T S_n \mathbf{u}[n])
\]

\( Q_n \succeq 0, \quad R_n > 0 \)

Can treat as one big (convex) QP:

\[
\begin{align*}
\min_{\mathbf{z}} & \quad \frac{1}{2} \mathbf{z}^T W \mathbf{z} \\
\text{s.t.} & \quad C \mathbf{z} + \mathbf{d} = 0
\end{align*}
\]

Instead, let’s apply DP.
DP for LQ Control

Initialize Bellman recursion:

\[ J_N^*(\mathbf{x}[N]) = \frac{1}{2} \mathbf{x}[N]^T Q_N \mathbf{x}[N] := \frac{1}{2} \mathbf{x}[N]^T V_N \mathbf{x}[N] \]

Apply recursion:

\[ J_{N-1}^*(\mathbf{x}[N - 1]) = \frac{1}{2} \min_{\mathbf{u}} \left[ \begin{bmatrix} \mathbf{x}[N - 1] \\ \mathbf{u} \end{bmatrix} \right]^T \begin{bmatrix} Q_{N-1} & S_{N-1} \\ S_{N-1}^T & R_{N-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}[N - 1] \\ \mathbf{u} \end{bmatrix} + \| \mathbf{x}[N] \|^2_{V_N} \]

Plug in for dynamics, optimize w.r.t. \( \mathbf{u} \) (set gradients to zero\(^*\)) and solve:

\[ \mathbf{u}^*[N - 1] = L_{N-1} \mathbf{x}[N - 1] \]

*Confirm:

\[ \nabla_{\mathbf{u}}^2 Q_{N-1}(\mathbf{x}, \mathbf{u}) = R_{N-1} + B_{N-1}^T V_N B_{N-1} > 0 \]

Plug optimal control law back into \( J_{N-1}^* \) to get

\[ J_{N-1}^* = \mathbf{x}[N - 1]^T V_{N-1} \mathbf{x}[N - 1] \]

Optimal cost-to-go is quadratic

Optimal policy is time-varying linear
Full backward recursion (Riccati difference recursion):

\[ V_N = Q_N \]

\[ L_n = -(R_n + B_n^T V_{n+1} B_n)^{-1} (B_n^T V_{n+1} A_n + S_n^T) \]

\[ V_n = Q_n + A_n^T V_{n+1} A_n - (A_n^T V_{n+1} B_n + S_n)(R_n + B_n^T V_{n+1} B_n)^{-1}(B_n^T V_{n+1} A_n + S_n^T) \]

\[ \pi^*_n(\mathbf{x}) = L_n \mathbf{x} \]

\[ J^*_n(\mathbf{x}[n]) = \frac{1}{2} \mathbf{x}[n]^T V_n \mathbf{x}[n] \]

For \( N = \infty \), \((A_n, B_n, Q_n, R_n, S_n) = (A, B, Q, R, S)\) with \((A, B)\) controllable, \(V_n, L_n \to \text{constant matrices} \) (thereby obtaining infinite horizon/stationary policy)
DP for LQ Control

If cost has linear terms $q_n^T x[n] + r_n^T u[n]$ and/or the dynamics has a drift term:

$$x[n + 1] = A_n x[n] + B_n u[n] + c_n$$

Then, re-write using the composite state $y = (x, 1)^T$:

$$y[n + 1] = \begin{bmatrix} x[n + 1] \\ 1 \end{bmatrix} = \begin{bmatrix} A_n & c_n \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x[n] \\ 1 \end{bmatrix} + \begin{bmatrix} B_n \\ 0 \end{bmatrix} u[n] := \tilde{A}_n y[n] + \tilde{B}_n u[n]$$

Implications:
- Optimal cost-to-go is a general quadratic:
  $$J_n^*(x[n]) = \frac{1}{2} x[n]^T V_n x[n] + v_n^T x[n] + p_n$$
- Optimal policy is time-varying affine:
  $$\pi_n^*(x) = L_n x + k_n$$
Optimal Control

Open Loop

Indirect Methods
1. Derive conditions of optimality
2. Solve these equations e.g., Pontryagin Minimum Principle

Direct Methods
1. Just solve as one big optimization problem.

Closed Loop

Principle of Optimality

Discrete-time: DP

Continuous-time: HJB

Application to Linear Dynamics
Optimal Control

Open Loop
- Indirect Methods
- Direct Methods

Closed Loop
- Principle of Optimality
  - Discrete-time: DP
  - Continuous-time: HJB

Application to Linear Dynamics
Open-Loop Optimal Control

Discrete Time:

• Performance measure (minimize): 
  \[ J(u, x) = \sum_{n=0}^{N-1} l(n, u[n], x[n]) + \phi(x[N]) \]

• Dynamics: 
  \[ x[n + 1] = f_d(x[n], u[n], n) \]

Open-loop: Given \( x[0] = x_0 \), find optimal sequences: \( (x^*[\ ], u^*[\ ]) \).

• If objective convex and dynamics linear \( \rightarrow \) convex problem.
Gradient Descent

• More generally, define the stage-wise Hamiltonian:

\[ H_n(x, u, \lambda) = l(n, x, u) + \lambda^T f_d(x, u, n) \]

• Then, with \( J_R(u) := J(u, x[u]) \), we have:

\[
\langle \nabla_u J_R, \delta u \rangle = \sum_{n=0}^{N-1} \nabla_{u[n]} H_n(x[n], u[n], \lambda[n + 1])^T \delta u[n]
\]

• Where \( \lambda \) (co-state/adjoint) satisfies a backward recursion:

\[
\lambda[n] = \frac{\partial l(n, x[n], u[n])}{\partial x} + \left( \frac{\partial f_d(x[n], u[n], n)}{\partial x} \right)^T \lambda[n + 1] \quad \lambda[N] = \frac{\partial \phi(x[N])}{\partial x}
\]
Newton Descent

Moreover, we also have:

\[
\langle \nabla_u J_R, \delta u \rangle + \frac{1}{2} \langle \delta u, \nabla^2_u J_R \delta u \rangle = \frac{1}{2} \delta x[N]^T \nabla^2_{xx} \phi(x[N]) \delta x[N] + \\
+ \sum_{n=0}^{N-1} \left( q_n^T \delta x[n] + r_n^T \delta u[n] + \frac{1}{2} \begin{bmatrix} \delta x[n] \\ \delta u[n] \end{bmatrix}^T \begin{bmatrix} \nabla^2_{xx} H_n & \nabla^2_{xu} H_n \\ \nabla^2_{ux} H_n & \nabla^2_{uu} H_n \end{bmatrix} \begin{bmatrix} \delta x[n] \\ \delta u[n] \end{bmatrix} \right)
\]

Where

\[
\delta x[n + 1] = A_n \delta x[n] + B_n \delta u[n], \quad \delta x[0] = 0
\]

\[
(A_n, B_n) = (\nabla_x f_d(n, x[n], u[n]), \nabla_u f_d(n, x[n], u[n]))
\]
Newton Descent

Moreover, we also have:

Newton Direction $\leftrightarrow$ Solve an LQ problem

Can we do better?
Optimal Control

Open Loop
- Indirect Methods
- Direct Methods: GD, Newton, SQP/SCP, IP, etc.

Closed Loop
- Principle of Optimality
  - Discrete-time: DP
  - Continuous-time: HJB
- Application to Linear Dynamics

Indirect Methods

Direct Methods
Differential Dynamic Programming (DDP)

Consider the DP recursion:

\[ J_n(x[n]) = \min_u [l(x[n], u) + J_{n+1}(f_d(x[n], u))] \]

Fix sequence of controls \( u \), with corresponding state sequence \( x \), and for \( n' = N - 1 \), consider:

\[ J_{n'}(x[n'] + \delta x) = \min_{\delta u} [l(x[n'] + \delta x, u[n'] + \delta u) + \phi(f_d(x[n'] + \delta x, u[n'] + \delta u))] \]

Taylor expand \( Q_{n'} \) about \((x[n'], u[n'])\) to 2nd order and minimize w.r.t. \( \delta u \)
- Yields affine control law: \( \delta u^*[n'] = L_n \delta x[n'] + \delta u_{n'} \)

\[ \text{“feedforward correction”} \]
- Substitute back into \( Q_{n'} \), yielding quadratic approximation for \( J_{n'} \) about \( x[n'] \)
- Continue recursion going backwards, with
  \[ Q_n(\delta x, \delta u) = l(x[n] + \delta x, u[n] + \delta u) + J_{n+1}(f_d(x[n] + \delta x, u[n] + \delta u)) \]
- *Almost* the Riccati backward recursion for the Newton LQ problem

Forward pass \( u + \delta u \) through \( f_d \) using affine control law: **DDP Algorithm**: “Quasi-Newton” Descent
- Newton would forward pass through linearized dynamics

Near optimal, behaves like Newton. Far from optimal, much more efficient.
Optimal Control

- Open Loop
  - Indirect Methods
  - Direct Methods
    - GD, Newton, SQP/SCP, IP, etc.
  - DDP

- Closed Loop
  - Principle of Optimality
    - Discrete-time: DP
    - Continuous-time: HJB
  - Application to Linear Dynamics

MPC
Model Predictive Control (MPC)
Tradeoffs on Open vs Closed

Open-Loop Control
- Rarely used blindly online due to model errors
- Expensive to compute online (need to use MPC and/or simplifications)
- Can compute offline for a set of initial conditions (e.g., a trajectory library) and adapt online via fine-tuning (e.g., Boston Dynamics Atlas Robot)

Closed-Loop Control
- Needed if there is any source of unmodelled effects (dynamics, disturbances, other agents, sensor noise), i.e., always
- Difficult to compute true optimal in discrete-time (functional recursion) or continuous-time (PDE)
  - Difficult to certify optimality
  - Instead, we look for certifying correctness and safety (e.g., via Lyapunov)
- MPC: bridge of open/closed-loop via online re-planning
- Feedback tracking: bridge of open/closed-loop: track open-loop plans with feedback controllers
Other Topics

Hybrid systems (e.g., locomotion)
Adaptive Control
Reinforcement Learning
Stochastic dynamics
Approximate Dynamic Programming
Partial observability
High-dimensional observations (vision)
Feedback controller design
Task & Motion Planning
Some References

The (updated) classic: Optimal Control & Dynamic Programming:
  ◦ Bertsekas Volumes 1 & 2

Introductory text – a must have:
  ◦ Kirk

Applied Optimal control – more advanced, generally assumes knowledge of the basics:
  ◦ Bryson and Ho

Model Predictive control – from a more modern perspective:
  ◦ Kouvaritakis & Cannon

Applied Nonlinear control – a comprehensive intro to nonlinear control:
  ◦ Slotine & Li