ME233 Advanced Control II
Lecture 1

Dynamic Programming &
Optimal Linear Quadratic Regulators (LQR)

(ME233 Class Notes DP1-DP4)

Outline

1. Dynamic Programming

2. Simple multi-stage example

3. Solution of finite-horizon optimal Linear Quadratic Reguator (LQR)

Dynamic Programming

Invented by Richard Bellman in 1953

- From **IEEE History Center: Richard Bellman**:
  - "*His invention of dynamic programming in 1953 was a major breakthrough in the theory of multistage decision processes…"*
  - *"A breakthrough which set the stage for the application of functional equation techniques in a wide spectrum of fields…"*
  - *"…extending far beyond the problem-areas which provided the initial motivation for his ideas."*

Dynamic Programming

Invented by Richard Bellman in 1953

- From **IEEE History Center: Richard Bellman**:
  - *In 1946 he entered Princeton as a graduate student at age 26.*
  - *He completed his Ph.D. degree in a record time of three months.*
  - *His Ph.D. thesis entitled “Stability Theory of Differential Equations" (1946) was subsequently published as a book in 1953, and is regarded as a classic in its field.*

Dynamic Programming

We will use dynamic programming to derive the solution of:

- Discrete time LQR and related problems
- Discrete time Linear Quadratic Gaussian (LQG) controller.
  - Optimal estimation and regulation

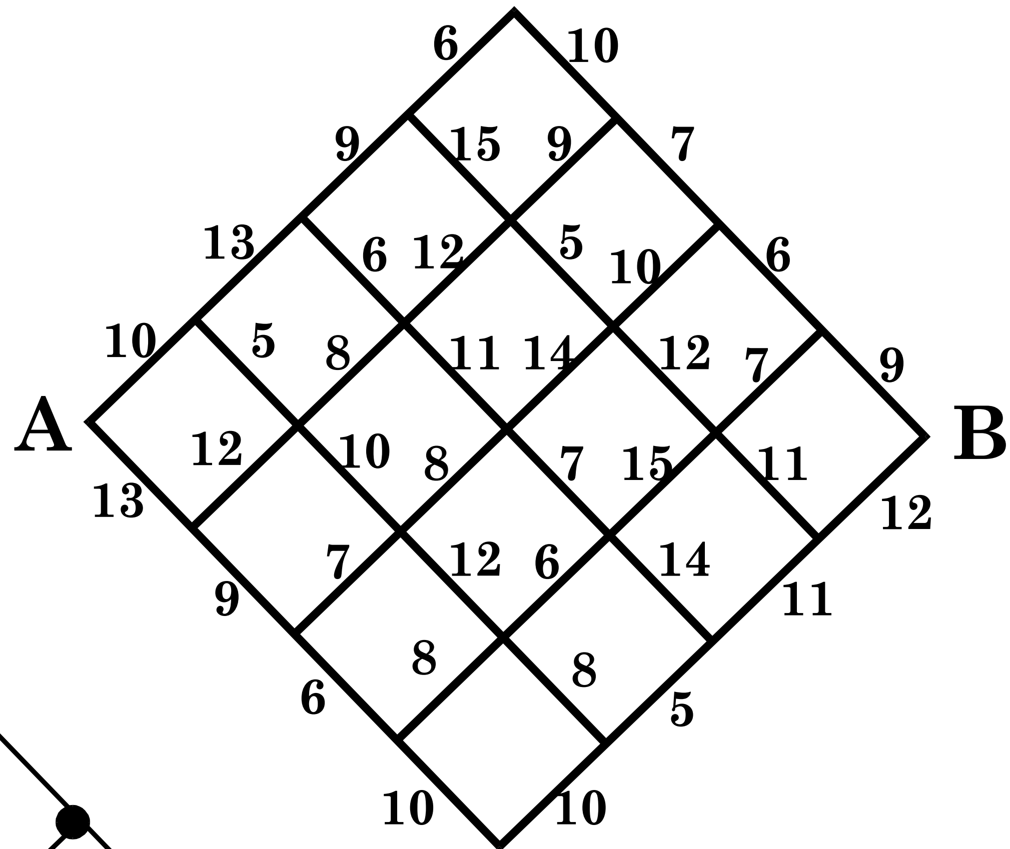
Dynamic Programming Example

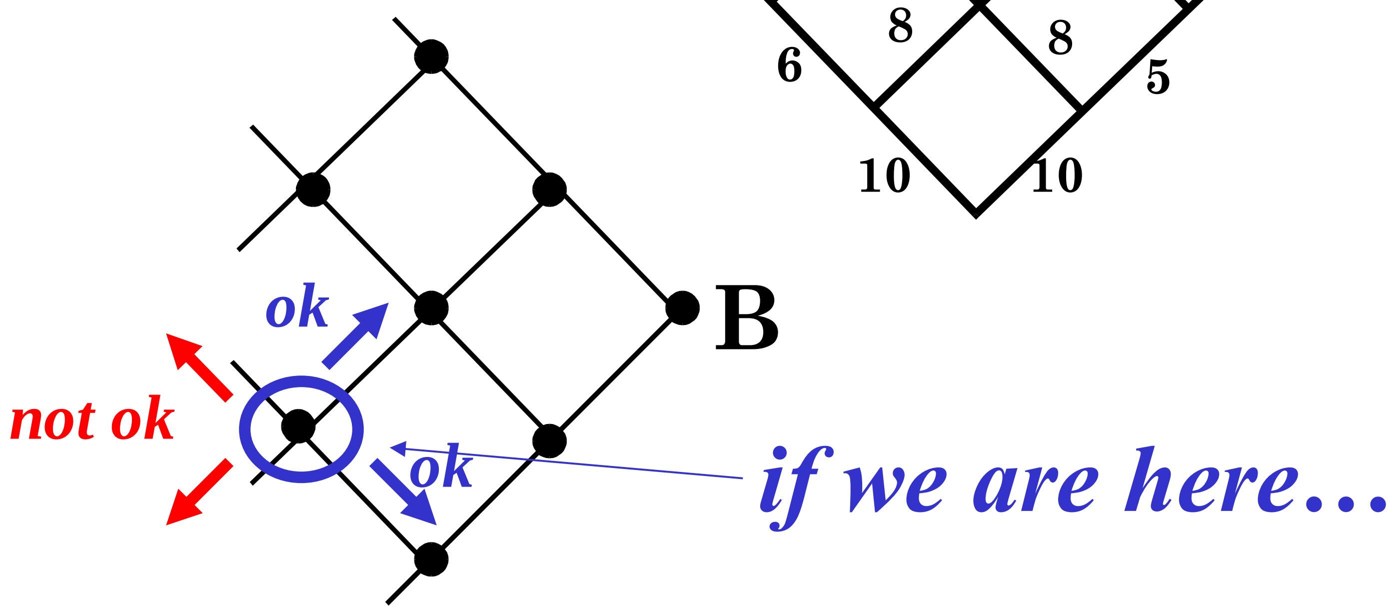
Illustrative Example:

Find “optimal” path:

From **A** to **B**

by <u>moving only to the right.</u>





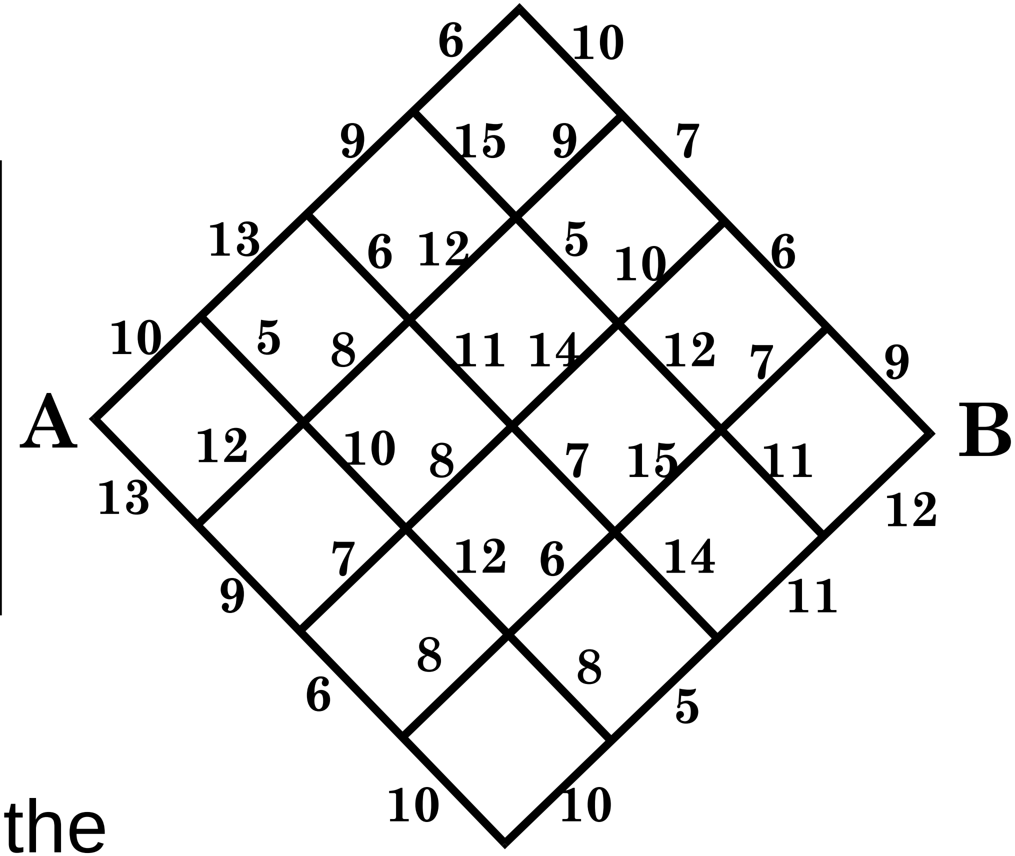
Dynamic Programming Example

Illustrative Example:

Find "optimal" path:

From **A** to **B**

by moving only to the right.



- Number next to line is the "cost" in going along that particular path.

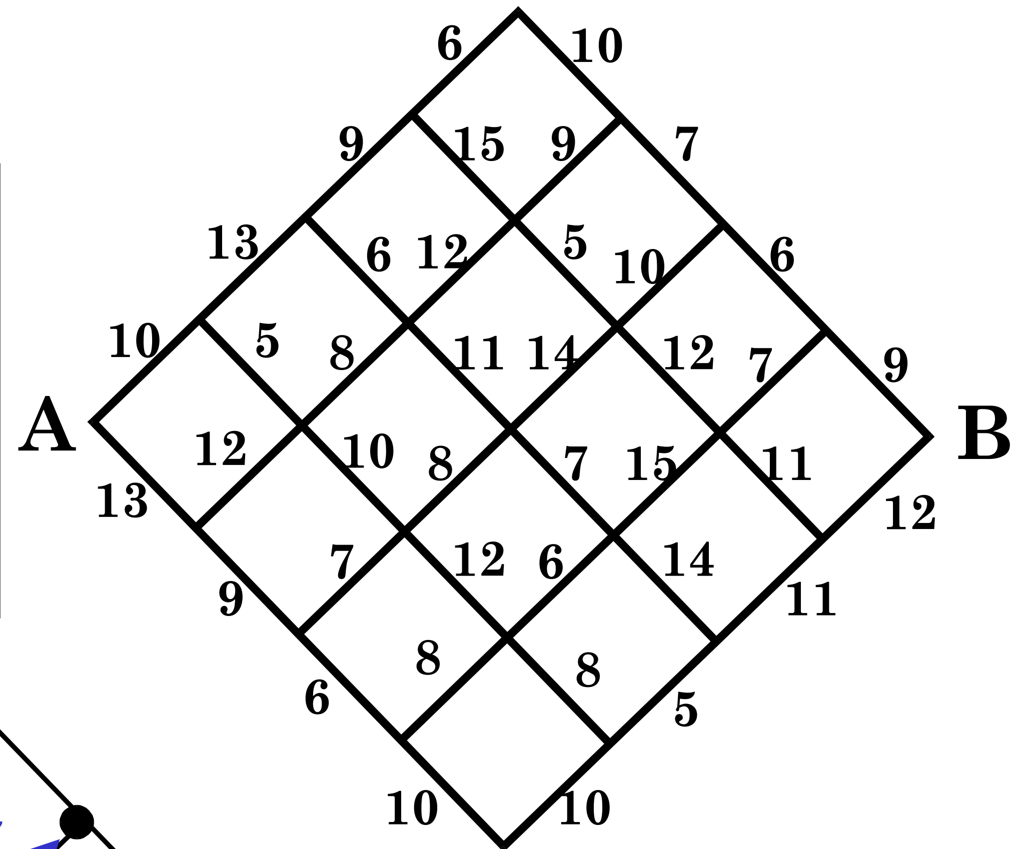
Dynamic Programming Example

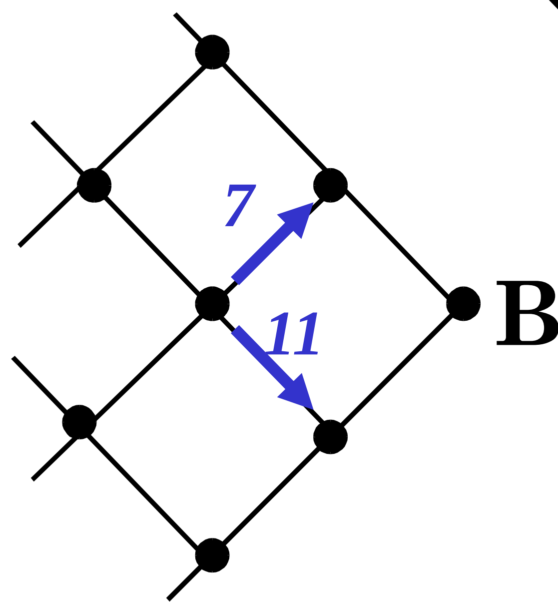
Illustrative Example:

Find “optimal” path:

From **A** to **B**

by moving only to the right.





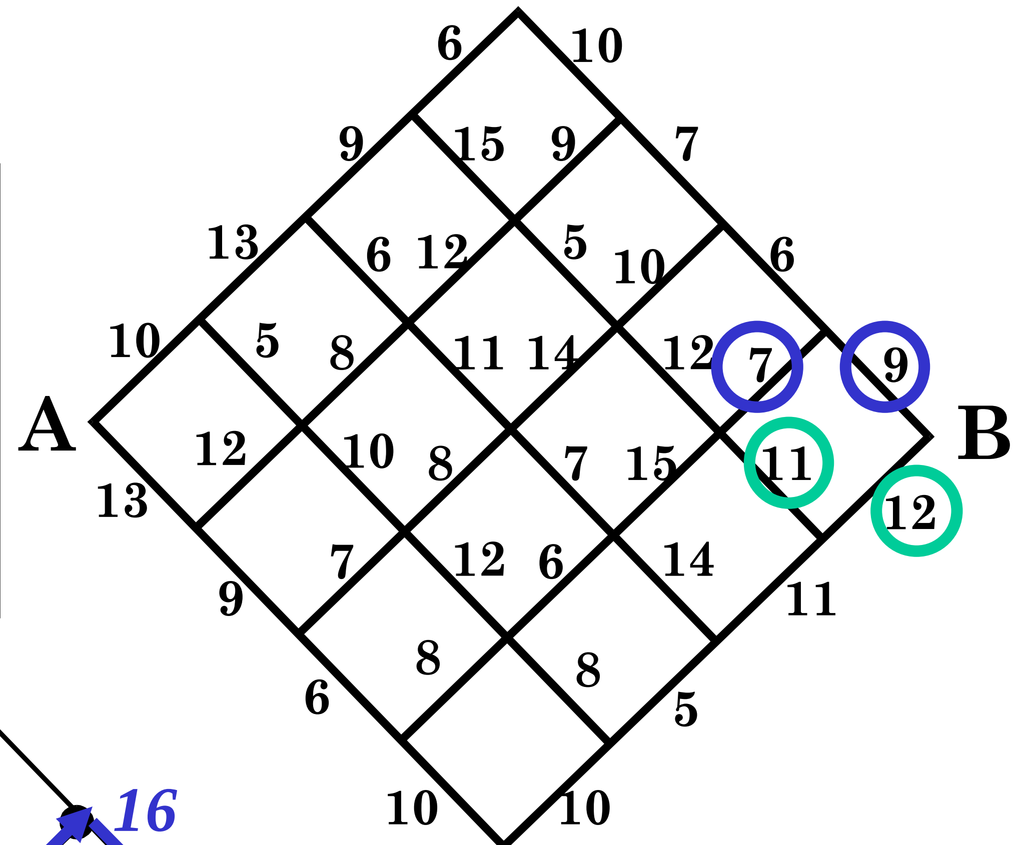
Dynamic Programming Example

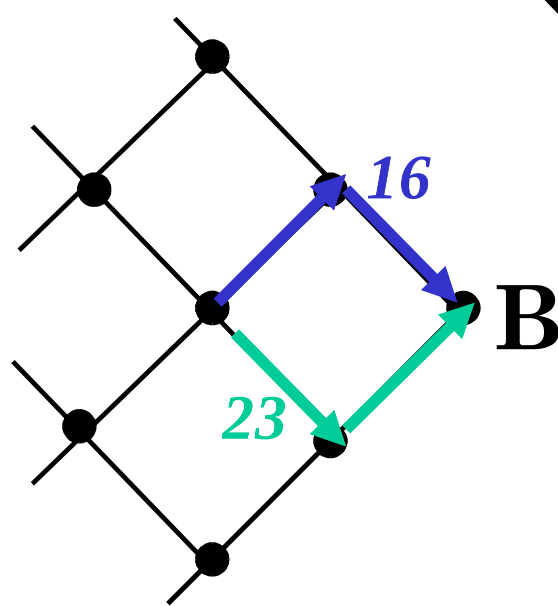
Illustrative Example:

Find “optimal” path:

From **A** to **B**

by moving only to the right.





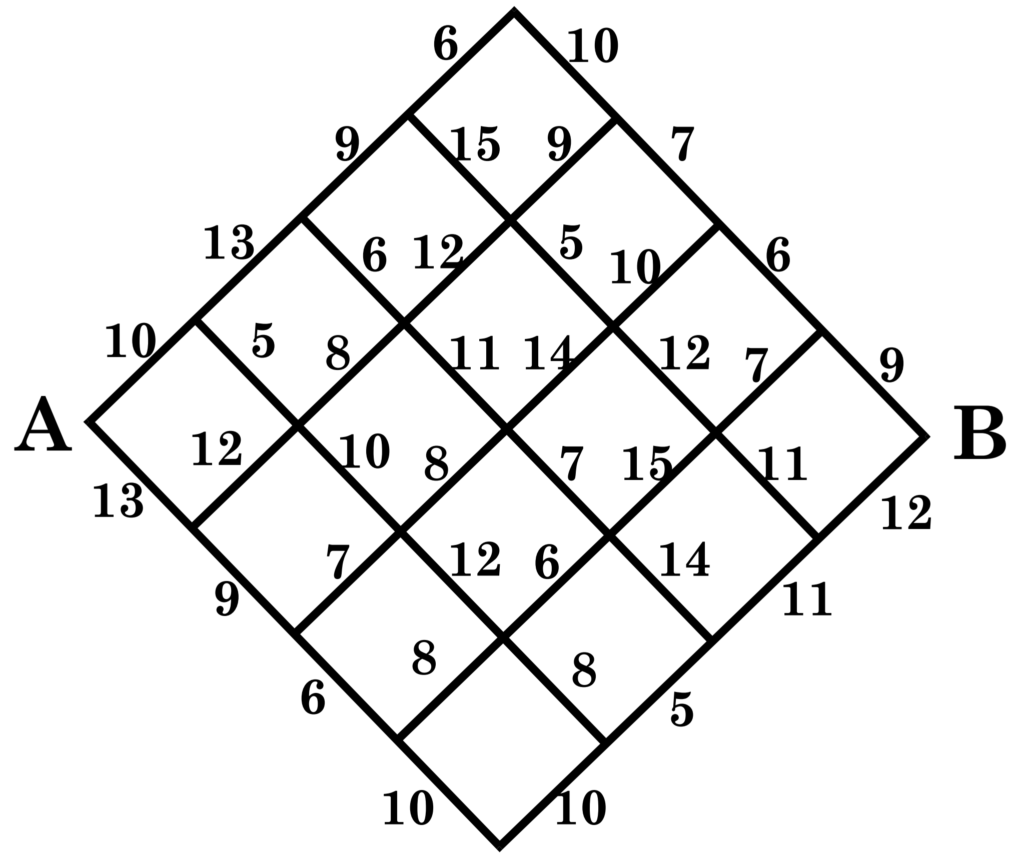
Dynamic Programming Example

Illustrative Example:

Find optimal path:

From **A** to **B**

by moving only to the right.



- **Optimal path from A to B is the one with the smallest overall cost.**

- There are 70 possible routes starting from **A.**

Dynamic Programming

Key idea:

- Convert a single "large" optimization problem into a series of "small" multistage optimization problems.

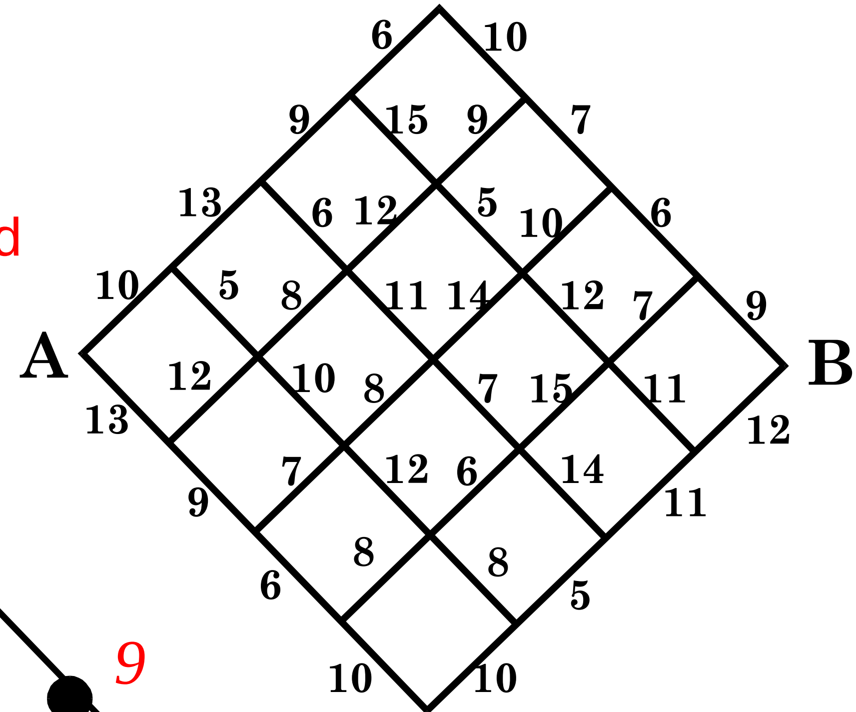
  - ***Principle of optimality:*** *"From any point on an optimal trajectory, the remaining trajectory is optimal for the corresponding problem initiated at that point."*

  - ***Optimal Value Function****: Compute the optimal value of the cost from each state to the final state.*

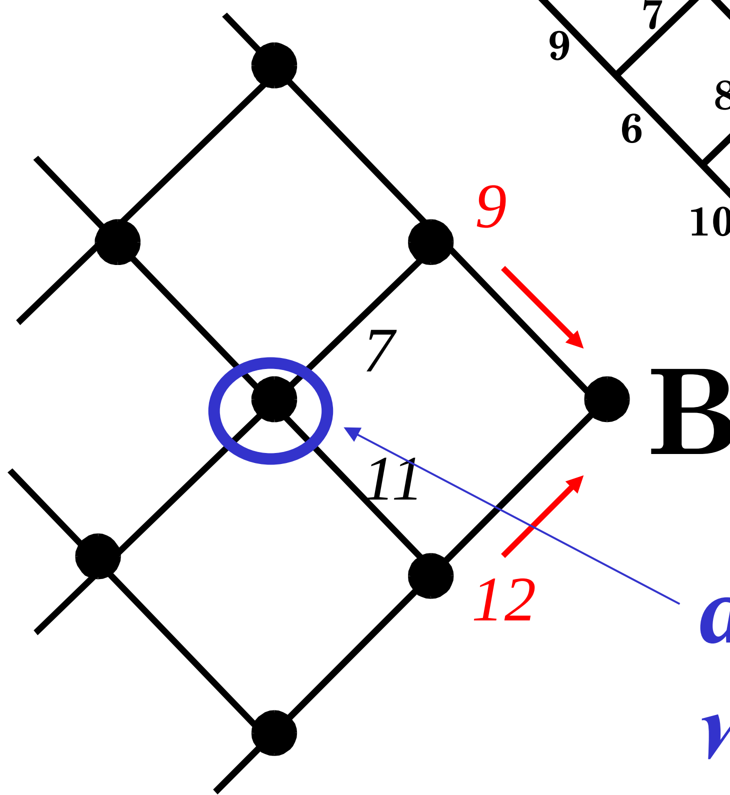
Dynamic Programming Example

Illustrative Example:

- Use principle of optimality
- Compute Optimal Value Function and optimal control at each state
- Start from the final state **B**



determine the optimal path from ⊙ to **B**

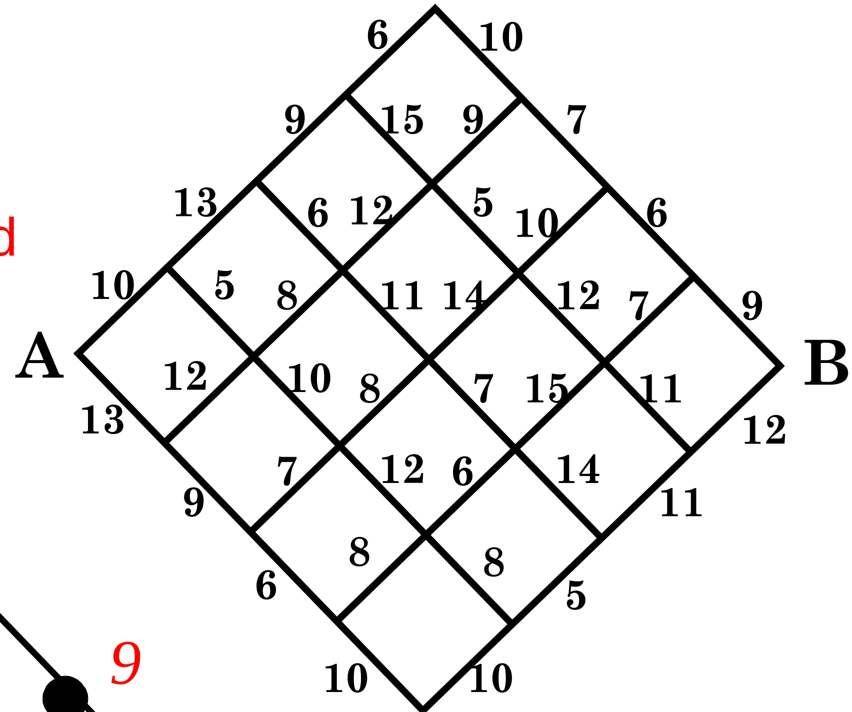


assume that we are here…

Dynamic Programming Example

Illustrative Example:

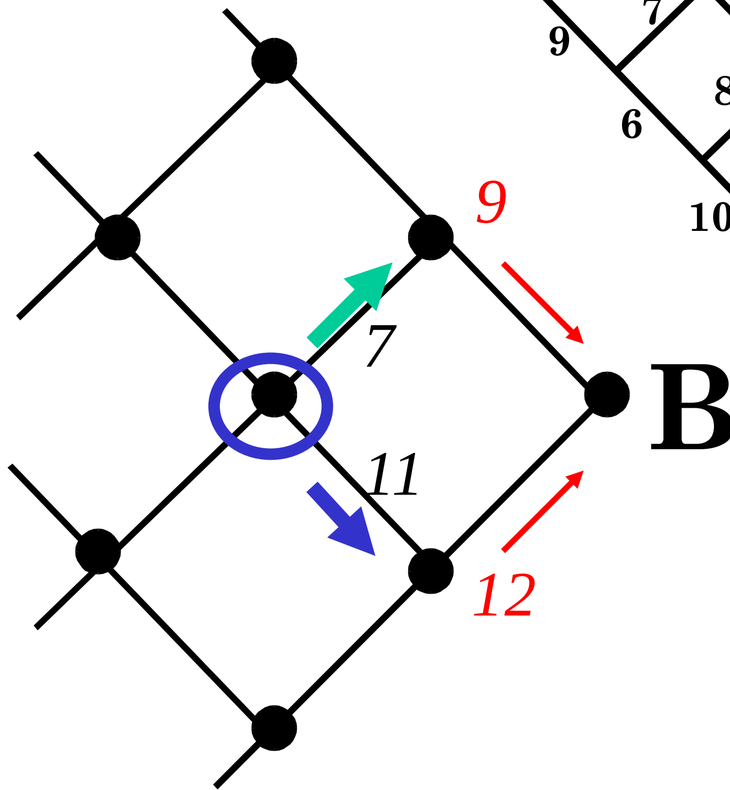
- Use principle of optimality
- Compute Optimal Value Function and optimal control at each state
- Start from the final state **B**



two options:

$7 + 9 = 16$

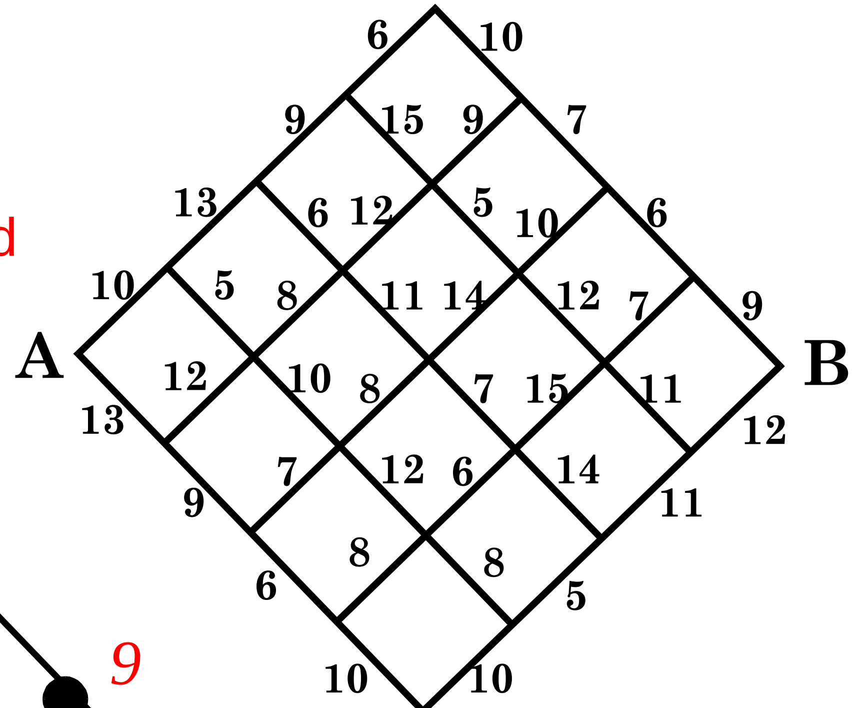
$11 + 12 = 23$



Dynamic Programming Example

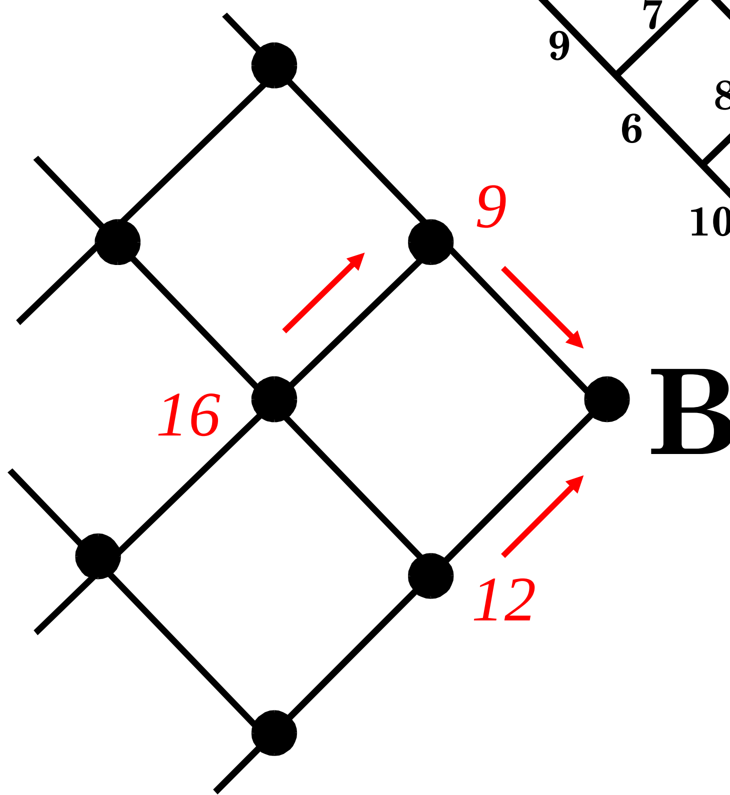
Illustrative Example:

- Use principle of optimality
- Compute Optimal Value Function and optimal control at each state
- Start from the final state **B**

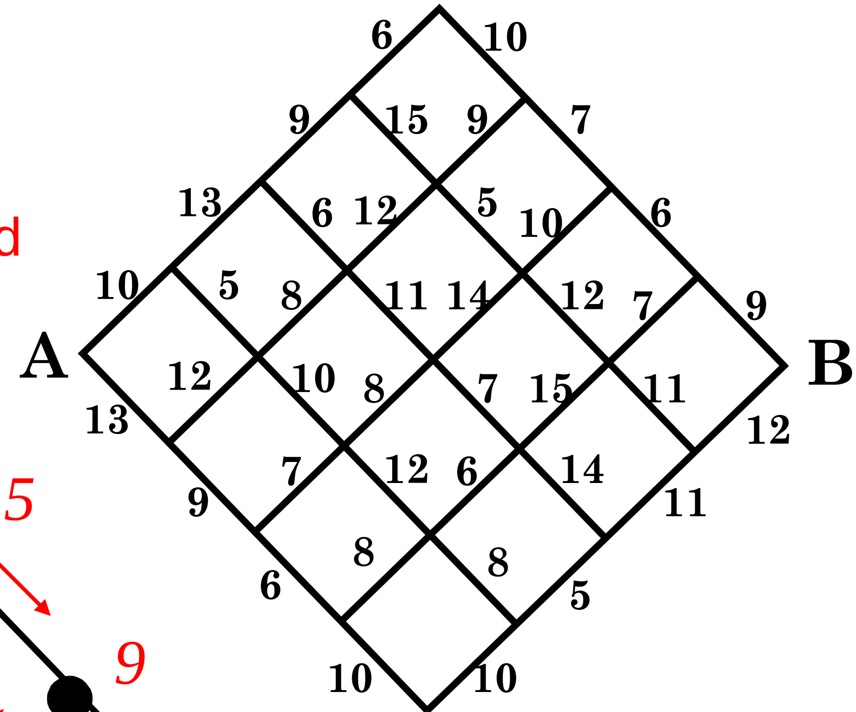


Assign:

- ***optimal path***
- ***optimal cost***



Dynamic Programming Example



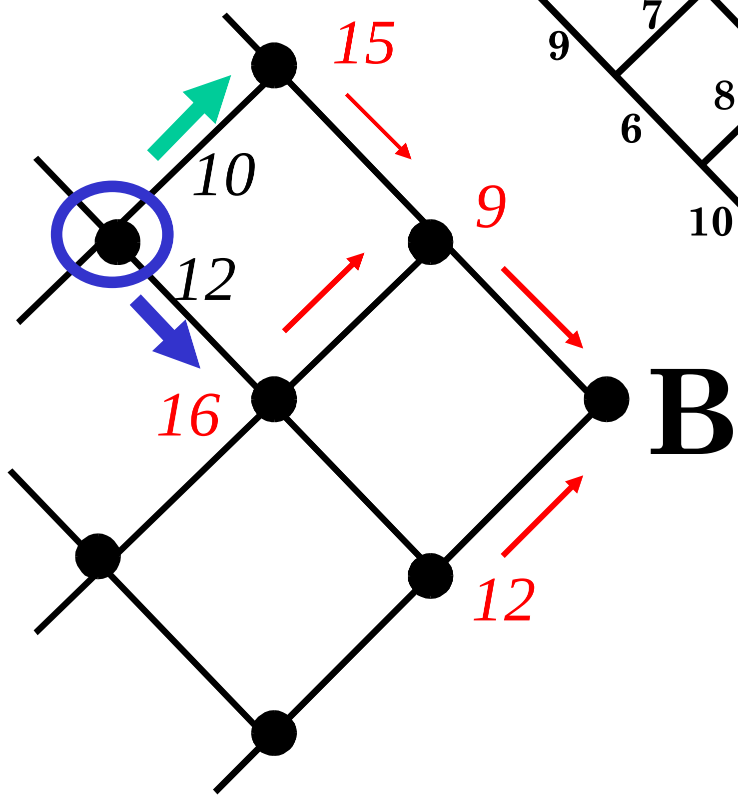
Illustrative Example:

- Use principle of optimality
- Compute Optimal Value Function and optimal control at each state
- Start from the final state **B**

Continue…

$$10 + 15 = 25$$

$$12 + 16 = 28$$



Dynamic Programming Example

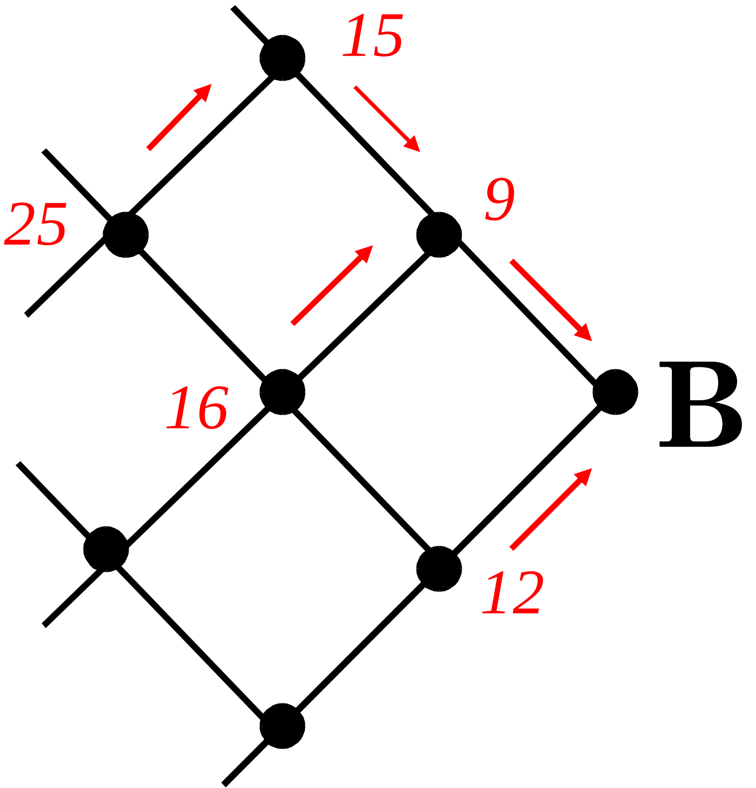
Illustrative Example:

- Use principle of optimality
- Compute Optimal Value Function and optimal control at each state
- Start from the final state **B**

Continue…

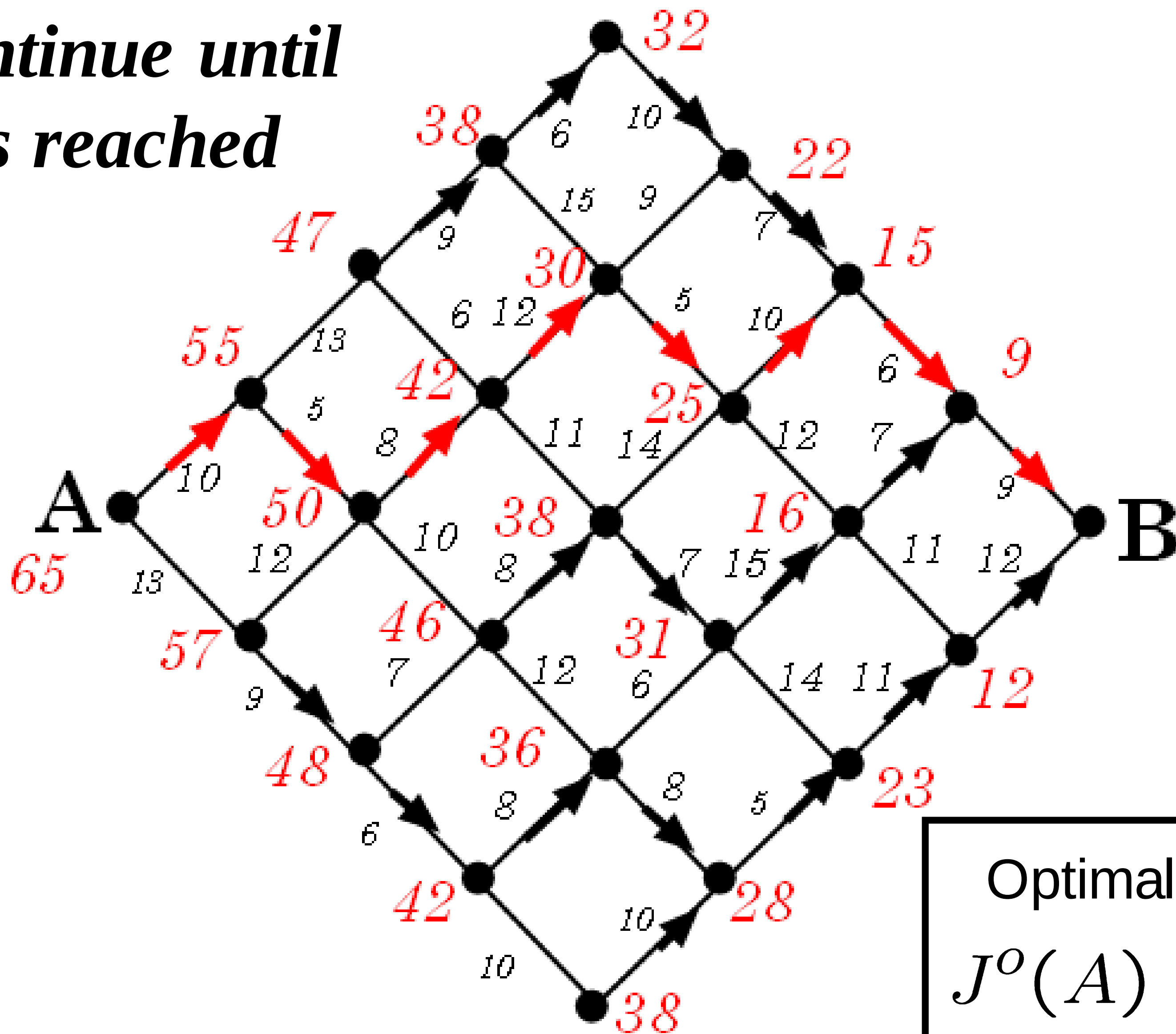
$$10 + \mathbf{15} = 25$$

$$12 + \mathbf{16} = 28$$



Dynamic Programming Example

Continue until
A is reached



Optimal cost:

$$J^o(A) = 65$$

LTI Optimal regulators

- State space description of a discrete time LTI

$$x(k+1) = A\,x(k) + B\,u(k)$$
$$x(0) = x_o$$

For now, everything is deterministic

- Find "optimal" control $u^0(k), \; k = 0, 1, 2 \cdots$

In some sense, to be defined later...

- That drives the state to the origin

$$x \to 0$$

Finite Horizon LQ optimal regulator

Consider the nth order discrete time LTI system:

$$x(k+1) \;=\; A\,x(k) + B\,u(k) \qquad\qquad x(0) = x_o$$

We want to find the optimal control sequence:

$$U_0^o \;=\; \{u^o(0),\, u^o(1),\, \cdots\,,\, u^o(N-1)\}$$

which minimizes the cost functional:

$$x^T(N)Q_f\,x(N) + \sum_{k=0}^{N-1}\left\{\begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix}\begin{bmatrix} x(k) \\ u(k) \end{bmatrix}\right\}$$

Finite Horizon LQ optimal regulator

Consider the nth order discrete time LTI system:

$$x(k+1) = A\,x(k) + B\,u(k) \qquad x(0) = x_o$$

Notice that the value of the cost depends on the initial condition $x(0) = x_o$

$$J[x(0)] = x^T(N) Q_f\, x(N) + \sum_{k=0}^{N-1} \left\{ \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \right\}$$

To emphasize the dependence on $x(0) = x_o$

LQ Cost Functional:

$$J[x(0)] = x^T(N)Q_f\,x(N) + \sum_{k=0}^{N-1} \left\{ \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \right\}$$

- N total number of steps—"horizon"
- $x^T(N)Q_f\,x(N)$ penalizes the final state deviation from the origin
- $\begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}$ penalizes the transient state deviation from the origin and the control effort

$$Q_f \succeq 0 \qquad \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \succeq 0 \qquad R \succ 0$$

symmetric

LQ Cost Functional:

Simplified nomenclature:

$$J[x(0)] = \underbrace{x^T(N) Q_f \, x(N)}_{} + \sum_{k=0}^{N-1} \underbrace{\left\{ \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \right\}}_{}$$

final state cost

transient cost at each step

$$J[x(0)] = L_f[x(N)] + \sum_{k=0}^{N-1} L[x(k), u(k)]$$

Additional notation

For $m = 0, 1, \ldots, N-1$ define:

Optimal control sequence from instance m

$$U_m^o = \Big(u^o(m), u^o(m+1), \ldots, u^o(N-1)\Big)$$

Arbitrary control sequence from instance m:

$$U_m = \Big(u(m),\, u(m+1),\, \ldots,\, u(N-1)\Big)$$

Dynamic Programming

Optimal cost functional

$$J^O[x(0)] = \min_{U_0} \underbrace{\left\{ L_f[x(N)] + \sum_{k=0}^{N-1} L[x(k), u(k)] \right\}}_{J[x(0)]}$$

Function of initial state

$$U_0 = \Big(u(0),\, u(1),\, \ldots,\, u(N-1) \Big)$$

Control sequence from instance 0

Optimal Incremental Cost Function

For $m = 0, 1, \ldots, N-1$ define:

Optimal cost function from state $x(m)$ at instant m

$$J_m^o[x(m)] = \min_{U_m} \left\{ L_f[x(N)] + \sum_{k=m}^{N-1} L[x(k), u(k)] \right\}$$

$$U_m = \Big(u(m),\, u(m+1),\, \ldots,\, u(N-1) \Big)$$

Control sequence from instance m

Optimal Cost Function

Optimal cost function at the final state $x(N)$

$$J_N^o[x(N)] = L_f[x(N)]$$

… only a function of the final state $x(N)$

Dynamic Programming

For $m = 0, 1, \ldots, N-2$:

Optimal value function: $J_m^o[x(m)]$

$$J_m^o[x(m)] = \min_{U_m} \left\{ L_f[x(N)] + \underbrace{\sum_{k=m}^{N-1} L[x(k), u(k)]} \right\}$$

$$\sum_{k=m}^{N-1} L[x(k), u(k)] = L[x(m), u(m)] + \sum_{k=m+1}^{N-1} L[x(k), u(k)]$$

Dynamic Programming

Optimal value function: $(m = 0, 1, \ldots, N-2)$

$$J_m^o[x(m)] = \min_{U_m} \left\{ L_f[x(N)] + L[x(m), u(m)] + \sum_{k=m+1}^{N-1} L[x(k), u(k)] \right\}$$

$$= \min_{u(m)} \min_{U_{m+1}} \left\{ L_f[x(N)] + L[x(m), u(m)] + \sum_{k=m+1}^{N-1} L[x(k), u(k)] \right\}$$

$$= \min_{u(m)} \left\{ L[x(m), u(m)] + \underbrace{\min_{U_{m+1}} \left\{ L_f[x(N)] + \sum_{k=m+1}^{N-1} L[x(k), u(k)] \right\}}_{J_{m+1}^o[x(m+1)]} \right\}$$

$$J_{m+1}^o[x(m+1)] = J_{m+1}^o[Ax(m) + Bu(m)]$$

Dynamic Programming

Optimal value function: $(m = 0, 1, \ldots, N-2)$

$$J_m^o[x(m)] = \min_{U_m} \left\{ L_f[x(N)] + \sum_{k=m}^{N-1} L[x(k), u(k)] \right\}$$

$$J_m^o[x(m)] = \min_{u(m)} \left\{ \underbrace{L[x(m), u(m)]} + \underbrace{J_{m+1}^o[Ax(m) + Bu(m)]} \right\}$$

given ***x(m)****, these are only functions of* ***u(m)*** *!!*

<u>only an optimization with respect to a single vector</u>

Bellman Equation

$$\boxed{J_m^o[x(m)] = \min_{u(m)} \left\{ L[x(m), u(m)] + J_{m+1}^o[x(m+1)] \right\} \qquad m = 0, 1, \ldots, \underline{N-1}}$$

1. The Bellman equation can be solved recursively (backwards), starting from N:

$$J_N^o[x(N)] = L_f[x(N)]$$

2. Each iteration involves only an optimization with respect to a single variable ($u(m)$) – ***multistage optimization***

Recursive Solution to the Bellman Equation

$$\boxed{\begin{array}{r} J_m^o[x(m)] = \min_{u(m)} \left\{ L[x(m), u(m)] + J_{m+1}^o[x(m+1)] \right\} \\ \\ m = 0, 1, \ldots, N-1 \end{array}}$$

$$J_N^o[x(N)] = L_f[x(N)]$$

boundary condition

known function of ***x(N)***

not known

Recursive Solution to the Bellman Equation

Start with ***N-1:*** *assume that* ***x(N-1)*** *is given*

find $u^0(N-1)$ by solving:

known function of $x(N)$

$$J^o_{N-1}[x(N-1)] = \min_{u(N-1)} \left\{ L[x(N-1), u(N-1)] + L_f[(x(N)] \right\}$$

$$x(N) = A\,x(N-1) + B\,u(N-1)$$

$u^0(N-1)$ *will be a function of* $x(N-1)$

Recursive Solution to the Bellman Equation

Continue with ***N-2:*** *assume that* ***x(N-2)*** *is given*

find $u^0(N-2)$ by solving:

known function of $x(N-1)$

$$J^o_{N-2}[x(N-2)] = \min_{u(N-2)} \left\{ L[x(N-2), u(N-2)] + J^o_{N-1}[x(N-1)] \right\}$$

$$x(N-1) \;=\; A\,x(N-2) + B\,u(N-2)$$

$u^0(N-2)$ *will be a function of* $x(N-2)$

Solving the Bellman Equation for a LQR

$$\boxed{J^o_m[x(m)] = \min_{u(m)} \left\{ L[x(m), u(m)] + J^o_{m+1}[x(m+1)] \right\} \qquad m = 0, 1, \ldots, N-1}$$

1) $$J^o_N[x(N)] = L_f[x(N)] = x^T(N)\, Q_f\, x(N)$$

2) $$L[x(k), u(k)] = \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}$$

Quadratic functions

Minimization of quadratic functions

For $M_{22} \succ 0$ *we have that:*

- $$\min_u \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^T & M_{22} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} = x^T \left(M_{11} - M_{12} M_{22}^{-1} M_{12}^T \right) x$$
- *Optimal* u *given by* $u^o = -M_{22}^{-1} M_{12}^T x$

Proof:

$$\begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^T & M_{22} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} = x^T M_{11} x + \underbrace{x^T M_{12} u + u^T M_{12}^T x + u^T M_{22} u}$$

Completing the square

$$\overbrace{(u + M_{22}^{-1} M_{12}^T x)^T M_{22} (u + M_{22}^{-1} M_{12}^T x) - x^T M_{12} M_{22}^{-1} M_{12}^T x}$$

Minimization of quadratic functions

For $M_{22} \succ 0$ *we have that:*

- $$\min_u \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^T & M_{22} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} = x^T \left(M_{11} - M_{12} M_{22}^{-1} M_{12}^T \right) x$$
- *Optimal* u *given by* $u^o = -M_{22}^{-1} M_{12}^T x$

Proof:

$$\begin{aligned} \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^T & M_{22} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} &= x^T (M_{11} - M_{12} M_{22}^{-1} M_{12}^T) x \\ &\quad + (u + M_{22}^{-1} M_{12}^T x)^T M_{22} (u + M_{22}^{-1} M_{12}^T x) \\ &\geq x^T (M_{11} - M_{12} M_{22}^{-1} M_{12}^T) x, \quad \forall u \end{aligned}$$

$$\begin{bmatrix} x \\ u^o \end{bmatrix}^T \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^T & M_{22} \end{bmatrix} \begin{bmatrix} x \\ u^o \end{bmatrix} = x^T (M_{11} - M_{12} M_{22}^{-1} M_{12}^T) x$$

$\blacksquare$

Finite-horizon LQR solution

$$J_k^o[x(k)] = x(k)^T P(k) x(k)$$

$$u^o(k) = -K(\underline{k+1})x(k)$$

$$K(k) = [B^T P(k) B + R]^{-1}[B^T P(k) A + S^T]$$

Where $P(k)$ is computed **backwards in time** using the *discrete Riccati difference equation* :

$$P(N) = Q_f$$

$$\begin{aligned} P(k-1) = {} & A^T P(k) A + Q \\ & - [A^T P(k) B + S][B^T P(k) B + R]^{-1}[B^T P(k) A + S^T] \end{aligned}$$

Proof of finite-horizon LQR solution

Proof (by induction on decreasing k)

Let $J^o_{k+1}[x(k+1)] = x(k+1)^T P(k+1) x(k+1)$

(Trivially holds for $k=N-1$ by definition of $J^o_N[x(N)]$)

$$J^o_{k+1}[x(k+1)] = [Ax(k) + Bu(k)]^T P(k+1) \underbrace{[Ax(k) + Bu(k)]}$$

$$x(k+1) = \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}$$

Proof of finite-horizon LQR solution

$$J^o_{k+1}[x(k+1)] = [Ax(k)+Bu(k)]^T P(k+1)\underbrace{[Ax(k)+Bu(k)]}_{\begin{bmatrix} A & B \end{bmatrix}\begin{bmatrix} x(k) \\ u(k) \end{bmatrix}}$$

$$= \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} A^T \\ B^T \end{bmatrix} P(k+1) \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}$$

$$= \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} A^T P(k+1) A & A^T P(k+1) B \\ B^T P(k+1) A & B^T P(k+1) B \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}$$

Proof of finite-horizon LQR solution

The Bellman equation gives

$$J_k^o[x(k)] = \min_{u(k)} \left\{ L[x(k), u(k)] + J_{k+1}^o[x(k+1)] \right\}$$

$$= \min_{u(k)} \left\{ \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} A^T P(k+1) A & A^T P(k+1) B \\ B^T P(k+1) A & B^T P(k+1) B \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \right\}$$

$$= \min_{u(k)} \left\{ \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} A^T P(k+1) A + Q & A^T P(k+1) B + S \\ B^T P(k+1) A + S^T & B^T P(k+1) B + R \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \right\}$$

Proof of finite-horizon LQR solution

$$J_k^o[x(k)] = \min_{u(k)} \left\{ \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} A^T P(k+1) A + Q & A^T P(k+1) B + S \\ B^T P(k+1) A + S^T & B^T P(k+1) B + R \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \right\}$$

Using results for quadratic optimizations:

$$J_k^o[x(k)] = x(k)^T P(k) x(k)$$

$$u^o(k) = -K(k+1) x(k)$$

where

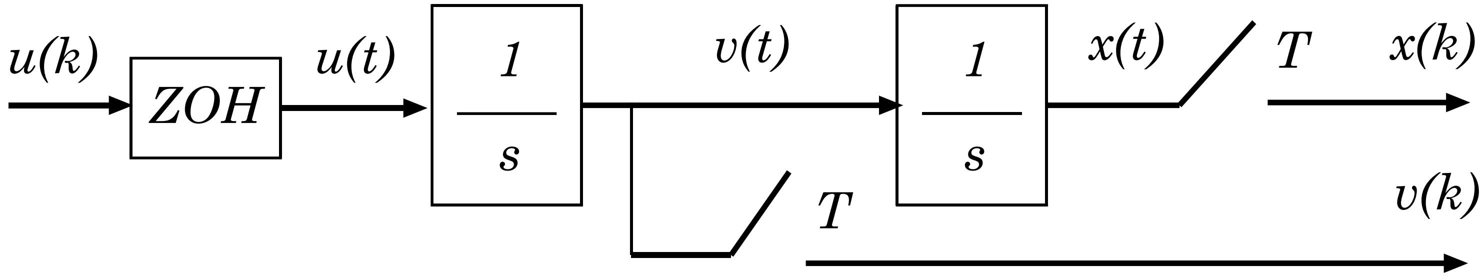
$$\begin{aligned} P(k) = A^T P(k+1) A + Q - [A^T P(k+1) B + S] \\ \times [B^T P(k+1) B + R]^{-1} [B^T P(k+1) A + S^T] \end{aligned}$$

$$K(k+1) = [B^T P(k+1) B + R]^{-1} [B^T P(k+1) A + S^T]$$

$\blacksquare$

Example – Double Integrator

Double integrator with ZOH and sampling time T =1:



$x_1(k) \longleftrightarrow x(kT)$ *position*

$x_2(k) \longleftrightarrow v(kT)$ *velocity*

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix} u(k)$$

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u(k)$$

Example – Double Integrator

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u(k)$$

LQR cost:

$$J[x_o] = x^T(N) Q_f\, x(N) + \sum_{k=0}^{N-1} \underbrace{\left\{ \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \right\}}_{x_1^2(k) + R u^2(k)}$$

Choose: $Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$

$$R > 0$$

$$S = 0$$

$$P(N) = Q_f \succeq 0$$

only penalize position x_1 and control u

Example – Double Integrator (DI)

Compute ***P(k)*** for an arbitrary $P(N) = Q_f$ and N.

Computing backwards: $P(N) = Q_f$

$$P(k-1) = A^T P(k) A + Q$$

$$- A^T P(k) B \left[B^T P(k) B + R \right]^{-1} B^T P(k) A$$

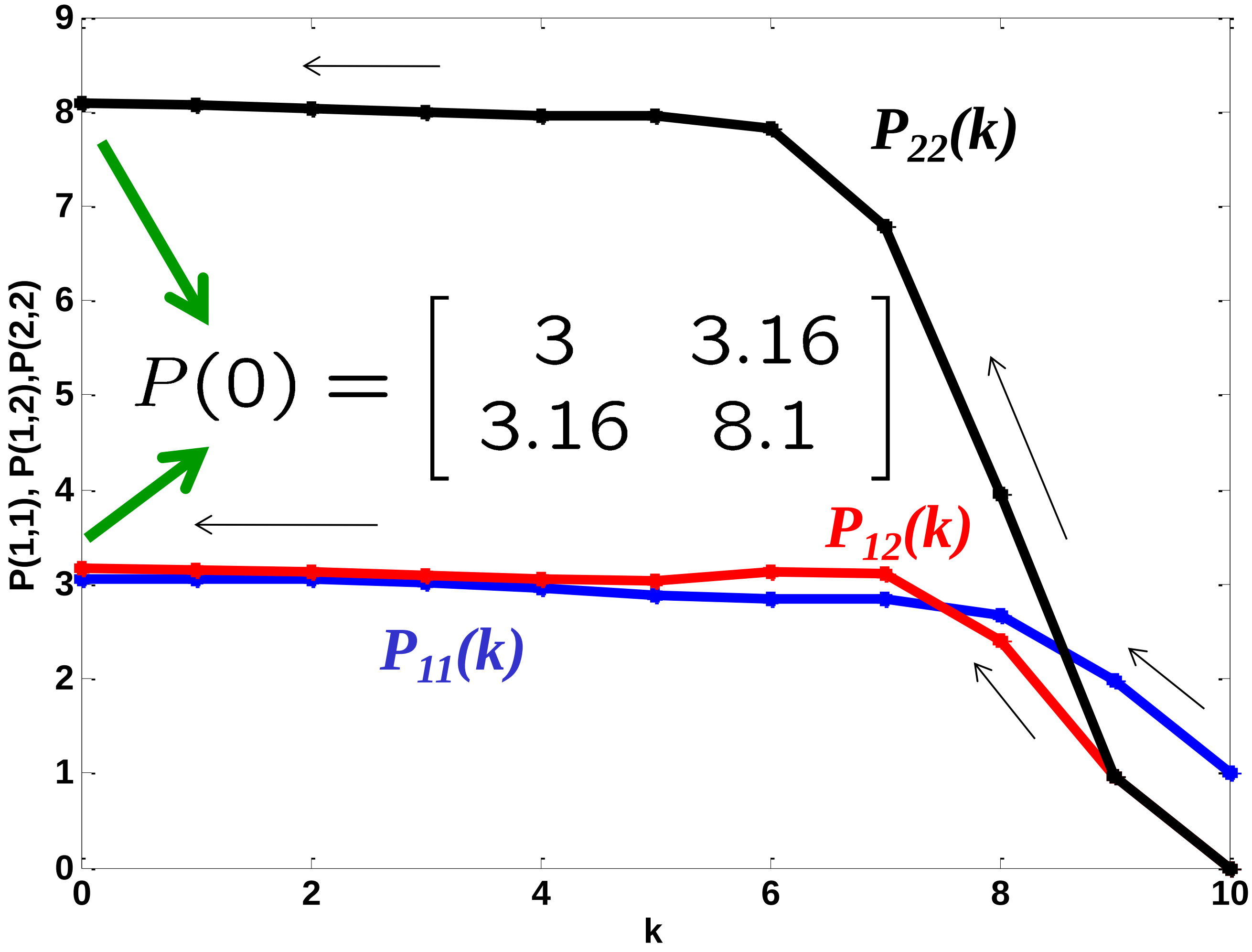
$$R > 0$$

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \quad Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

Example – DI Finite Horizon Case 1

- $N = 10$, $R = 10$,

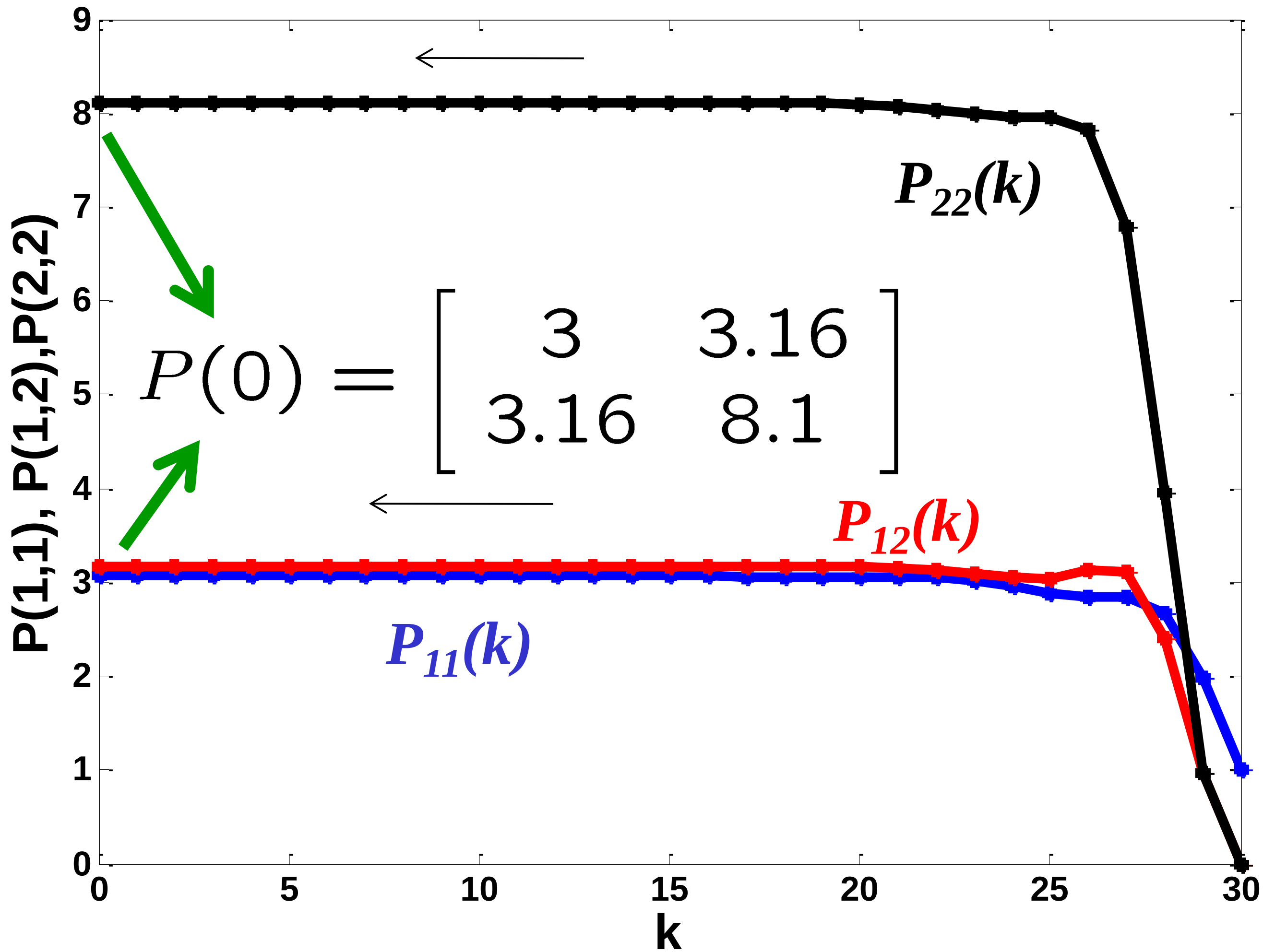
$$P(10) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$



$$P(0) = \begin{bmatrix} 3 & 3.16 \\ 3.16 & 8.1 \end{bmatrix}$$

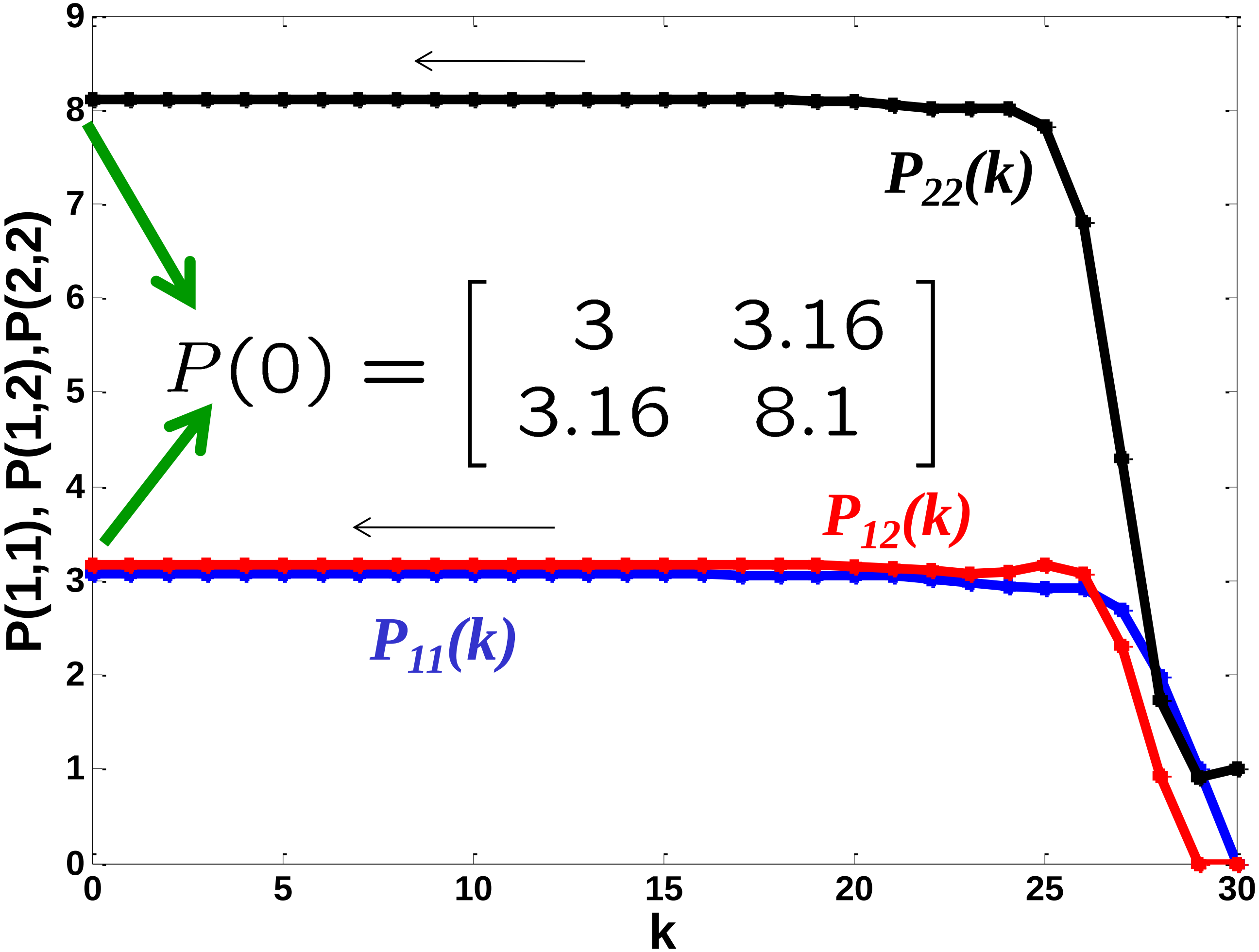
Example – DI Finite Horizon Case 2

- $N = 30$, $R = 10$, $$P(30) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$



Example – DI Finite Horizon Case 3

- $N = 30$, $R = 10$, $\quad P(30) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$



$$P(0) = \begin{bmatrix} 3 & 3.16 \\ 3.16 & 8.1 \end{bmatrix}$$

Example – DI Finite Horizon

Observation:

In all cases, regardless of the choice of $P(N) = Q_f$

when the horizon, N, is sufficiently large

the backwards computation of the Riccati Eq. always converges to the same solution:

$$P(0) = \begin{bmatrix} 3 & 3.16 \\ 3.16 & 8.1 \end{bmatrix}$$

We will return to this important idea in a few lectures

Properties of Matrix $\boldsymbol{P(k)}$

$P(k)$ satisfies:

1) $P(k) = P^T(k)$ (symmetric)

2) $P(k) \succeq 0$ (positive semi-definite)

Properties of Matrix ***P(k)***

$$P(k) = P^T(k) \qquad \text{(symmetric)}$$

Proof: (by induction on decreasing k)

Base case, $k=N$:

$$P(N)^T = Q_f^T = Q_f = P(N)$$

For $k \in \{0, 1, \ldots, N-1\}$:

$$P(k) = A^T P(k+1) A + Q - [A^T P(k+1) B + S] \times [B^T P(k+1) B + R]^{-1} [B^T P(k+1) A + S^T]$$

Transpose both sides of the equation ■

Properties of Matrix ***P(k)***

$P(k) \succeq 0$ (positive semi-definite)

Proof: (by induction on decreasing k)

<u>*Base case, $k=N$:*</u>

$P(N) = Q_f \succeq 0$

<u>*For $k \in \{0, 1, \ldots, N-1\}$:*</u>

$$P(k) = A^T P(k+1) A + Q - [A^T P(k+1) B + S] \times [B^T P(k+1) B + R]^{-1} [B^T P(k+1) A + S^T]$$

$\Downarrow$ *Algebra…*

$$= [A - BK(k+1)]^T P(k+1) [A - BK(k+1)] + \begin{bmatrix} I \\ -K(k+1) \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} I \\ -K(k+1) \end{bmatrix} \succeq 0$$

$\blacksquare$

Summary

- Bellman's dynamic programming invention was a major breakthrough in the theory of multistage decision processes and optimization

- Key ideas

  - Principle of optimality
  - Computation of optimal cost function

- Illustrated with a simple multi-stage example

Summary

- Bellman's equation:

$$J_m^o[x(m)] = \min_{u(m)} \left\{ L[x(m), u(m)] + J_{m+1}^o[x(m+1)] \right\}$$

  - has to be solved backwards in time
  - may be difficult to solve
  - the solution yields a feedback law

$$J^o[x(m)] = \min_{U_m} \left\{ L_f[x(N)] + \sum_{k=m}^{N-1} L[x(k), u(k)] \right\}$$

Summary

Linear Quadratic Regulator (LQR)

- Bellman's equation is easily solved
- Optimal cost is a quadratic function

$$J^o[x(k)] = \frac{1}{2} x^T(k)\, P(k)\, x(k)$$

- matrix $\boldsymbol{P}$ is solved using a Riccati equation
- Optimal control is a linear time varying feedback law

$$u^o(k) = -K(k+1)\, x(k)$$