# ME233 Advance Control II
# Lecture 1

# Dynamic Programming &
# Optimal Linear Quadratic Regulators (LQR)
# Discrete Time

(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.
- Continuous time LQR (with a bit of hand waving).
- Discrete time Linear Quadratic Gaussian (LQG) controller.
  - Optimal estimation and regulation



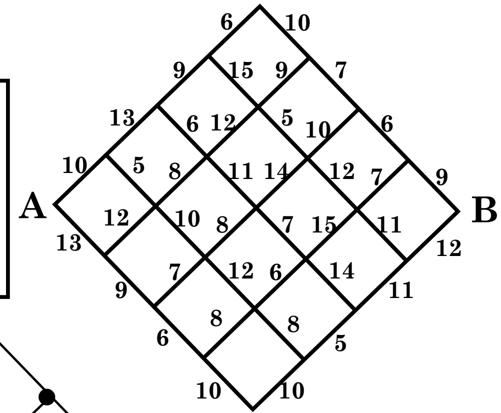
# Dynamic Programming Example

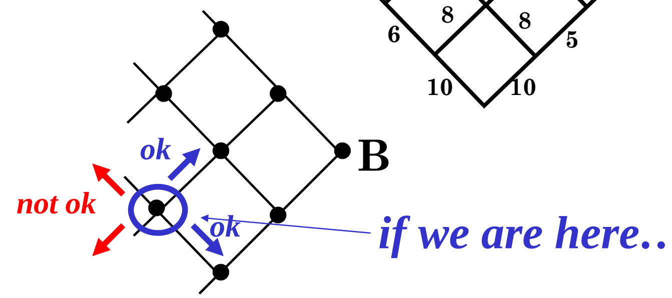
**Illustrative Example:**

Find optimal path:

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

by moving only to the right.





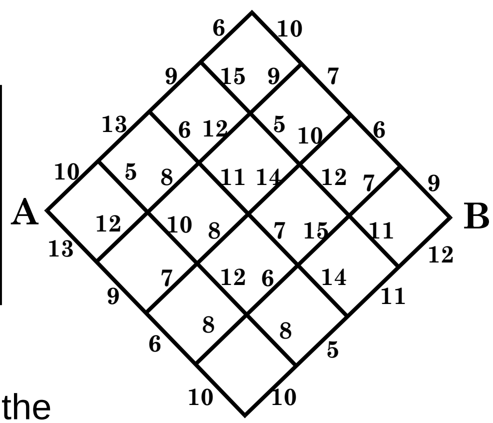

# Dynamic Programming Example

**Illustrative Example:**

Find optimal path:

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

by moving only to the right.




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



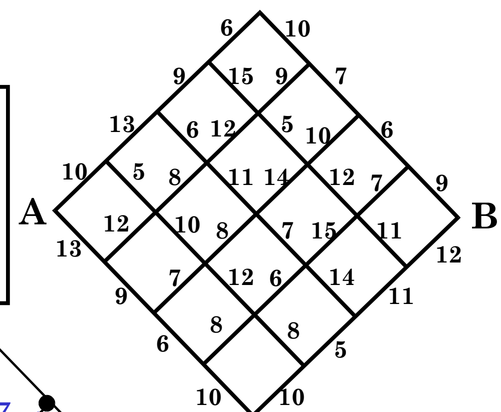
# Dynamic Programming Example

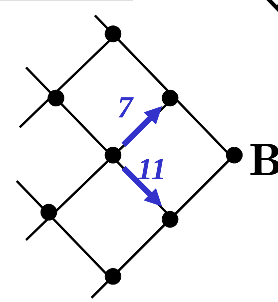
**Illustrative Example:**

Find optimal path:

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

by moving only to the right.



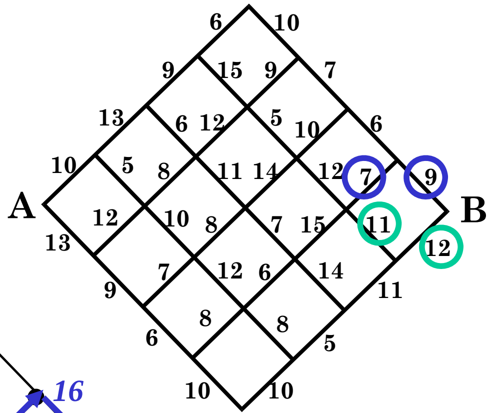



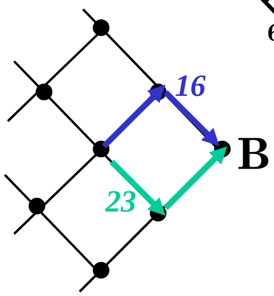
# Dynamic Programming Example

**Illustrative Example:**

Find optimal path:
From **A** to **B**

by moving only to the right.



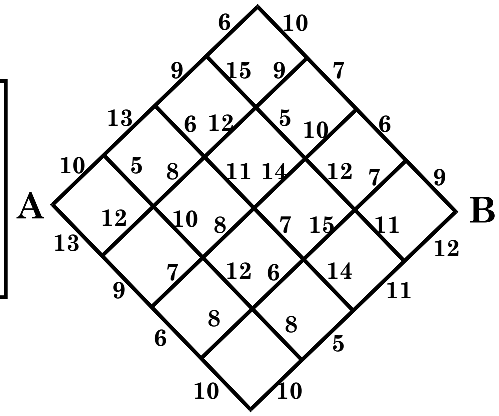



# Dynamic Programming Example

**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 20 possible routs starting from **A.**



# Dynamic Programming

Key idea:

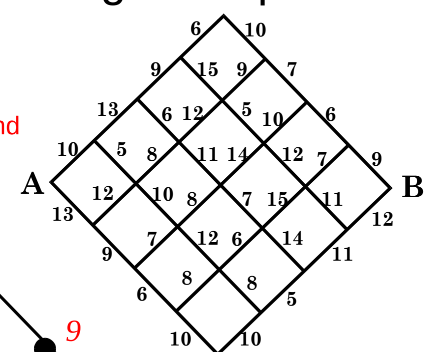
- Convert a single "large" optimization problem into a series of "small" multistage optimization problems.
  - ***Principle of optimality:*** *"From any point on an optimal trajectory, the remaining trajectory is optimal for the corresponding problem initiated at that point."*
  - ***Optimal Value Function****: Compute the optimal value of the cost from each state to the final state.*



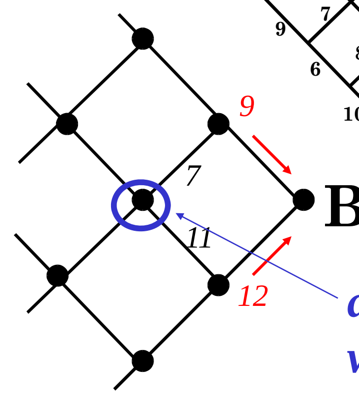
# Dynamic Programming Example

**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***

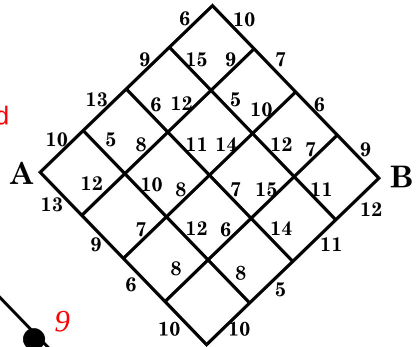



***assume that we are here…***

# Dynamic Programming Example

**Illustrative Example:**

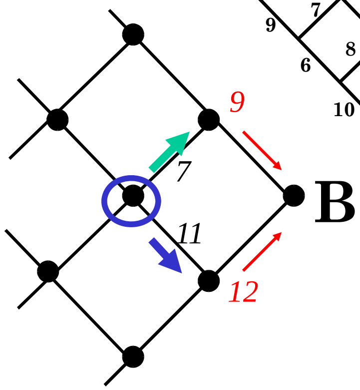
- Use principle of optimality
- Compute Optimal Value Function and optimal control at each state
- Start from the final state **B**




## two options:

*7 + 9 = 16*

*11 + 12 = 23*

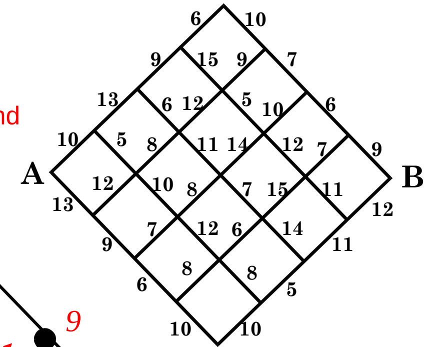





# Dynamic Programming Example

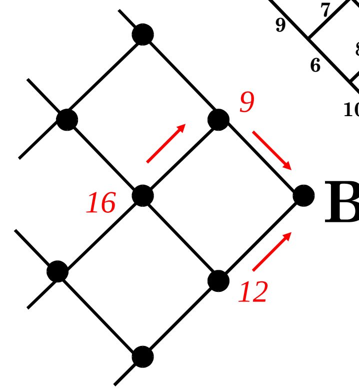
**Illustrative Example:**

- 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***

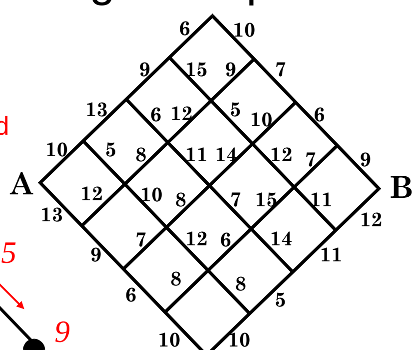





# Dynamic Programming Example

**Illustrative Example:**

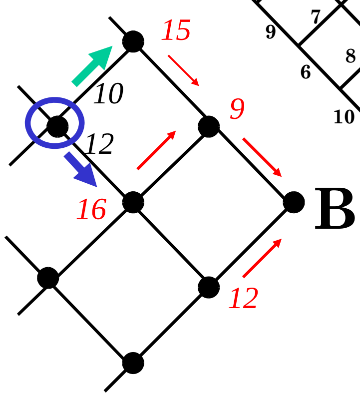
- Use principle of optimality
- Compute Optimal Value Function and optimal control at each state
- Start from the final state **B**




## Continue…

*10 + 15 = 25*

*12 + 16 = 28*






# Dynamic Programming Example

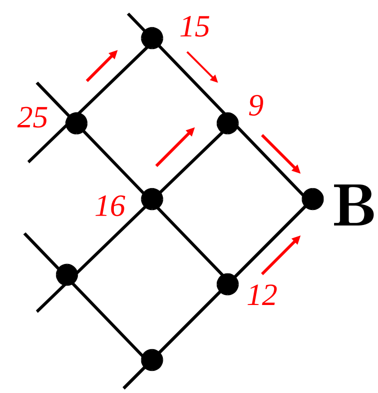
**Illustrative Example:**

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

## Continue…

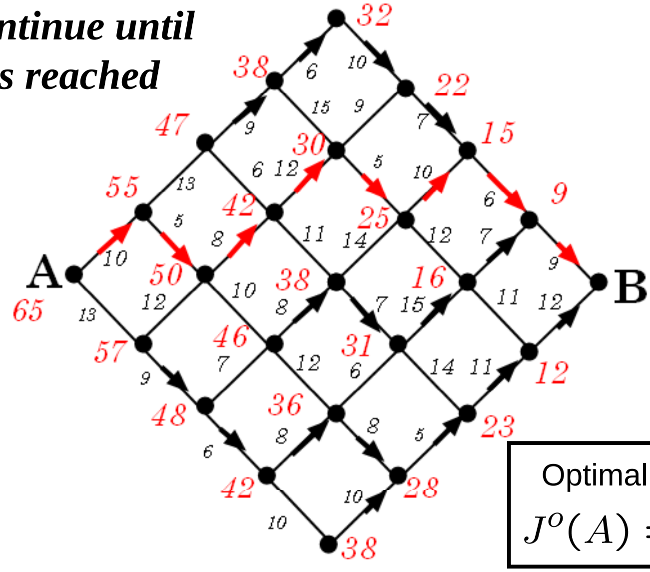
*10 + 15 = 25*

*12 + 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

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

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

- That drives the state to the origin

$$x \rightarrow 0$$



# Finite Horizon LQ optimal regulator

Consider the nth order discrete time LTI system:

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

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:

$$J = \frac{1}{2}x^T(N)\,S\,x(N) + \frac{1}{2}\sum_{k=0}^{N-1}\left\{x^T(k)\,Q\,x(k) + u^T(k)\,R\,u(k)\right\}$$



# 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 $\quad x(0) = x_o$

$$J[x_o] = \frac{1}{2}x^T(N)\,S\,x(N) + \frac{1}{2}\sum_{k=0}^{N-1}\left\{x^T(k)\,Q\,x(k) + u^T(k)\,R\,u(k)\right\}$$

*To emphasized the dependence on* $\; x(0) = x_o$

# LQ Cost Functional:

$$J[x_o] = \frac{1}{2}\,x^T(N)\,S\,x(N) + \frac{1}{2}\sum_{k=0}^{N-1}\left\{x^T(k)\,Q\,x(k) + u^T(k)\,R\,u(k)\right\}$$

- $N$ is the total number of steps
- $\frac{1}{2}\,x^T(N)\,S\,x(N)$ penalizes the final state deviation from the origin
- $\frac{1}{2}\,x^T(k)\,Q\,x(k)$ penalizes the transient state deviation from the origin
- $\frac{1}{2}\,u^T(k)\,R\,u(k)$ penalizes the control effort

$$S = S^T \succeq 0 \qquad Q = Q^T \succeq 0 \qquad R = R^T \succ 0$$



# LQ Cost Functional:

Simplified nomenclature:

$$J[x_o] = \underbrace{\frac{1}{2}\,x^T(N)\,S\,x(N)}_{\textit{final state cost}} + \frac{1}{2}\sum_{k=0}^{N-1}\underbrace{\left\{x^T(k)\,Q\,x(k) + u^T(k)\,R\,u(k)\right\}}_{\textit{transient cost at each step}}$$

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



# Additional notation

Optimal control sequence from instance $m$

$$U_m^o \;=\; \{u^o(m),\, u^o(m+1),\, \cdots,\, u^o(N-1)\}$$

$$(N-1 \geq m \geq 0)$$

Set of ***all*** possible control sequences from instance $m$:

$$U_m \;=\; \{u(m),\, u(m+1),\, \cdots,\, u(N-1)\}$$



# Dynamic Programming

Optimal cost functional

$$J^o[x_o] = \min_{U_0}\left\{S[x(N)] + \sum_{k=0}^{N-1} L[x(k), u(k)]\right\}$$

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

Set of ***all*** possible control sequences from 0

# Optimal Cost Function

Think of the optimal cost functional

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

as a function of the initial state $x(0) = x_o$

and the instant $0$

***simplified notation***

***state***

$$J^o[x(0), 0] = J^o[x(0)]$$

***instant***



# Optimal Cost Function

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

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

$$U_m = \{u(m),\, u(m+1),\, \cdots,\, u(N-1)\}$$

Set of ***all*** possible control sequences from instance $m$



# Optimal Cost Function

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

$$J^o[x(N)] = S[x(N)]$$

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



# Dynamic Programming

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

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

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

# Dynamic Programming

**Optimal value function:**

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

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

*assume that this part is*
*a* ***known*** *function of* ***x(m+1)***

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



# Dynamic Programming

**Optimal value function:**

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

$$J^o[x(m)] \;=\; \min_{u(m)} \{ \underbrace{L[x(m),u(m)]} + \underbrace{J^o[x(m+1)]} \}$$

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

$$x(m+1) \;=\; A\,x(m) + B\,u(m)$$



# Dynamic Programming

**Optimal value function:**

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

$$J^o[x(m)] \;=\; \min_{u(m)} \{ \underbrace{L[x(m),u(m)]} + \underbrace{J^o[x(m+1)]} \}$$

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

***only an optimization with respect to a single variable***



# Dynamic Programming

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

$$\begin{aligned}
J^o[x(m)] &= \min_{U_m} \left\{ S[x(N)] + \sum_{k=m}^{N-1} L[x(k),u(k)] \right\} \\
&= \min_{U_m} \left\{ L[x(m),u(m)] + \left[ S[x(N)] + \sum_{k=m+1}^{N-1} L[x(k),u(k)] \right] \right\} \\
&= \min_{u(m)} \left\{ L[x(m),u(m)] + \underbrace{\min_{U_{m+1}} \left[ S[x(N)] + \sum_{k=m+1}^{N-1} L[x(k),u(k)] \right]}_{J^o[x(m+1)]} \right\}
\end{aligned}$$

$$\boxed{J^o[x(m)] = \min_{u(m)} \{ L[x(m),u(m)] + J^o[x(m+1)] \}}$$

# ME233 Advance Control II
# Lecture 1

# Dynamic Programming &
# Optimal Linear Quadratic Regulators (LQR)
# Discrete Time
# Part II

(ME233 Class Notes DP1-DP4)



## Finite Horizon LQ optimal regulator

Consider the nth order discrete time LTI system:

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

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:

$$J[x_o] = \frac{1}{2}x^T(N)\,S\,x(N) + \frac{1}{2}\sum_{k=0}^{N-1}\left\{x^T(k)\,Q\,x(k) + u^T(k)\,R\,u(k)\right\}$$



## LQ Cost Functional:

Simplified nomenclature:

$$J[x_o] = \underbrace{\frac{1}{2}x^T(N)\,S\,x(N)}_{\textit{final state cost}} + \frac{1}{2}\sum_{k=0}^{N-1}\underbrace{\left\{x^T(k)\,Q\,x(k) + u^T(k)\,R\,u(k)\right\}}_{\textit{transient cost at each step}}$$

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



## Optimal Cost Function

<u>Optimal cost function</u> from state $x(m)$ at instant $m$

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

$$U_m = \{u(m),\, u(m+1),\, \cdots,\, u(N-1)\}$$

Set of ***all*** possible control sequences from instance $m$

# Bellman Equation

$$\boxed{J^o[x(m)] = \min_{u(m)} \{L[x(m), u(m)] + J^o[x(m+1)]\}}$$

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

$$J^o[x(N)] = S[x(N)]$$

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



# Recursive Solution to the Bellman Equation

$$\boxed{J^o[x(m)] = \min_{u(m)} \{L[x(m), u(m)] + J^o[x(m+1)]\}}$$

$$J^o[x(N)] = S[x(N)]$$ *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 optimal $u^0(N-1)$ by solving:

*known function of* $x(N)$

$$J^o[x(N-1)] = \min_{u(N-1)} \{L[x(N-1), u(N-1)] + S[(x(N))]\}$$

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

*optimal* $u^0(N-1)$ *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 optimal $u^0(N-2)$ by solving:

*known function of* $x(N-1)$

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

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

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

## Solving the Bellman Equation for a LQR

$$\boxed{J^o[x(m)] = \min_{u(m)} \{L[x(m), u(m)] + J^o[x(m+1)]\}}$$

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

2) $$L[x(k), u(k)] = \frac{1}{2}\left\{x^T(k)\, Q\, x(k) + u^T(k)\, R\, u(k)\right\}$$

***Quadratic functions***



## Minimization of quadratic functions

- Let $V[u(m)]$

  be an unconstrained quadratic function

- Then $$u^o(m) = ARG\left[\min_{u(m)} \{V[u(m)]\}\right]$$

  satisfies $$\left.\frac{\partial V[u(m)]}{\partial u(m)}\right|_{u^o(m)} = 0$$



## Multivariable function $V(u) \in \mathcal{R}$

Let $$u = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} \in \mathcal{R}^n$$

We will use the following convention:

$$\frac{\partial V(u)}{\partial u} = \begin{bmatrix} \frac{\partial V(u_1, \cdots, u_n)}{\partial u_1} \\ \vdots \\ \frac{\partial V(u_1, \cdots, u_n)}{\partial u_n} \end{bmatrix} \in \mathcal{R}^n$$



## Bi-linear function $W(x,y) \in \mathcal{R}$

$$W(x,y) = x^T\, M\, y = y^T\, M^T\, x$$

$x \in \mathcal{R}^n$ ($M$ is not necessarily square) $y \in \mathcal{R}^m$

$$\frac{\partial [W(x,y)]}{\partial x} = M\, y \;\in \mathcal{R}^n$$

$$\frac{\partial [W(x,y)]}{\partial y} = M^T\, x \;\in \mathcal{R}^m$$

## Quadratic function $V(u) \in \mathcal{R}$

$$V(u) = u^T\, M\, u = u^T\, M^T\, u$$

($M$ is not necessarily symmetric) $\qquad u \in \mathcal{R}^n$

***V(u)*** can always be re-written as

$$V(u) = u^T\, N\, u$$

$$N = \frac{1}{2}\left[M + M^T\right] \qquad \text{(symmetric } N\text{)}$$



## Solving the Bellman Equation for a LQR

$$\boxed{J^o[x(m)] = \min_{u(m)} \{L[x(m), u(m)] + J^o[x(m+1)]\}}$$

simplified notation

$$J^O[x(m)] = \min_{u(m)} \{J(x(m), u(m))\}$$

where

$$\begin{aligned} J[x(m), u(m)] &= L[x(m), u(m)] \\ &\quad + J^o[Ax(m) + B(u(m))] \end{aligned}$$



## $u^0(N-1)$ for a quadratic $J[x(N-1), u(N-1)]$

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

$$J^o[x(N-1)] = \min_{u(N-1)} \{J[x(N-1), u(N-1)]\}$$

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

$$\left.\frac{\partial J[x(N-1), u(N-1)]}{\partial u(N-1)}\right|_{u^o(N-1)} = 0$$



## Computing $J[x(N-1), u(N-1)]$

Assume ***x(N-1)*** is given

$$J[x(N-1), u(N-1)] = \underbrace{L[x(N-1), u(N-1)]} + S[x(N)]$$

$$\begin{aligned} L[x(N-1), u(N-1)] &= \frac{1}{2}\Big\{x^T(N-1)\, Q\, x(N-1) \\ &\quad + u^T(N-1)\, R\, u(N-1)\Big\} \end{aligned}$$

# Computing $J[x(N-1), u(N-1)]$

Assume ***x(N-1)*** is given

$$J[x(N-1), u(N-1)] = L[x(N-1), u(N-1)] + S[x(N)]$$

$$S[x(N)] = \frac{1}{2} x^T(N)\, S\, x(N)$$

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

$$S[x(N)] = \frac{1}{2}[A\, x(N-1) + B\, u(N-1)]^T S [A\, x(N-1) + B\, u(N-1)]$$



# Computing $J[x(N-1), u(N-1)]$

Assume ***x(N-1)*** is given

$$S[x(N)] = \frac{1}{2}[A\, x(N-1) + B\, u(N-1)]^T S [A\, x(N-1) + B\, u(N-1)]$$

$$\begin{aligned} S[x(N)] = \; & \frac{1}{2} x^T(N-1) \left[A^T S A\right] x(N-1) \\ + \; & \frac{1}{2} x^T(N-1) \left[A^T S B\right] u(N-1) \\ + \; & \frac{1}{2} u^T(N-1) \left[B^T S A\right] x(N-1) \\ + \; & \frac{1}{2} u^T(N-1) \left[B^T S B\right] u(N-1) \end{aligned}$$

*can be combined*



# Computing $J[x(N-1), u(N-1)]$

Assume ***x(N-1)*** is given

$$S[x(N)] = \frac{1}{2}[A\, x(N-1) + B\, u(N-1)]^T S [A\, x(N-1) + B\, u(N-1)]$$

$$\begin{aligned} S[x(N)] = \; & \frac{1}{2} x^T(N-1) \left[A^T S A\right] x(N-1) \\ + \; & x^T(N-1) \left[A^T S B\right] u(N-1) \\ + \; & \frac{1}{2} u^T(N-1) \left[B^T S B\right] u(N-1) \end{aligned}$$



# Computing $J[x(N-1), u(N-1)]$

$$J[x(N-1), u(N-1)] = L[x(N-1), u(N-1)] + S[x(N)]$$

$$\begin{aligned} = \; & \frac{1}{2} x^T(N-1)\, Q\, x(N-1) \\ + \; & \frac{1}{2} u^T(N-1)\, R\, u(N-1) \\ + \; & \frac{1}{2} x^T(N-1) \left[A^T S A\right] x(N-1) \\ + \; & x^T(N-1) \left[A^T S B\right] u(N-1) \\ + \; & \frac{1}{2} u^T(N-1) \left[B^T S B\right] u(N-1) \end{aligned}$$

## Computing $J[x(N-1), u(N-1)]$

combine

$$J[x(N-1), u(N-1)] = \frac{1}{2} x^T(N-1)\, Q\, x(N-1)$$

combine

$$+ \ \frac{1}{2} u^T(N-1)\, R\, u(N-1)$$

$$+ \ \frac{1}{2} x^T(N-1) \left[A^T S A\right] x(N-1)$$

$$+ \ x^T(N-1) \left[A^T S B\right] u(N-1)$$

$$+ \ \frac{1}{2} u^T(N-1) \left[B^T S B\right] u(N-1)$$



## Computing $J[x(N-1), u(N-1)]$

$$J[x(N-1), u(N-1)] = \frac{1}{2} x^T(N-1) \left[Q + A^T S A\right] x(N-1)$$

$$+ \ x^T(N-1) \left[A^T S B\right] u(N-1)$$

$$+ \ \frac{1}{2} u^T(N-1) \left[R + B^T S B\right] u(N-1)$$

Quadratic and bilinear functions of ***x(N-1)*** and ***u(N-1)***



## Computing $\dfrac{\partial J[x(N-1), u(N-1)]}{\partial u(N-1)}$

$$\underbrace{J[x(N-1), u(N-1)]}_{J[*]} = \frac{1}{2} x^T(N-1) \left[Q + A^T S A\right] x(N-1)$$

$$+ \ x^T(N-1) \left[A^T S B\right] u(N-1)$$

$$+ \ \frac{1}{2} u^T(N-1) \left[R + B^T S B\right] u(N-1)$$



## Computing $\dfrac{\partial J[x(N-1), u(N-1)]}{\partial u(N-1)}$

$$\frac{\partial J[*]}{\partial u(N-1)} = \cancelto{0}{\frac{\partial}{\partial u(N-1)} \left\{ \frac{1}{2} x^T(N-1) \left[Q + A^T S A\right] x(N-1) \right\}}$$

$$+ \ \frac{\partial}{\partial u(N-1)} \left\{ x^T(N-1) \left[A^T S B\right] u(N-1) \right\}$$

$$+ \ \frac{\partial}{\partial u(N-1)} \left\{ \frac{1}{2} u^T(N-1) \left[R + B^T S B\right] u(N-1) \right\}$$

## Computing $\frac{\partial J[x(N-1),u(N-1)]}{\partial u(N-1)}$

$$\frac{\partial J[*]}{\partial u(N-1)} = \cancelto{0}{\frac{\partial}{\partial u(N-1)}\left\{\frac{1}{2}x^T(N-1)\left[Q+A^TSA\right]x(N-1)\right\}}$$

$$+ \cancelto{B^TSA\,x(N-1)}{\frac{\partial}{\partial u(N-1)}\left\{x^T(N-1)\left[A^TSB\right]u(N-1)\right\}}$$

$$+ \cancelto{[R+B^TSB]\,u(N-1)}{\frac{\partial}{\partial u(N-1)}\left\{\frac{1}{2}u^T(N-1)\left[R+B^TSB\right]u(N-1)\right\}}$$

58

## Computing $\frac{\partial J[x(N-1),u(N-1)]}{\partial u(N-1)}$

$$\frac{\partial J[*]}{\partial u(N-1)} = \cancelto{0}{\frac{\partial}{\partial u(N-1)}\left\{\frac{1}{2}x^T(N-1)\left[Q+A^TSA\right]x(N-1)\right\}}$$

$$+ \cancelto{B^TSA\,x(N-1)}{\frac{\partial}{\partial u(N-1)}\left\{x^T(N-1)\left[A^TSB\right]u(N-1)\right\}}$$

$$+ \cancelto{[R+B^TSB]\,u(N-1)}{\frac{\partial}{\partial u(N-1)}\left\{\frac{1}{2}u^T(N-1)\left[R+B^TSB\right]u(N-1)\right\}}$$



## Computing $u^0(N-1)$

$$\frac{\partial J[x(N-1),u(N-1)]}{\partial u(N-1)} = B^TSA\,x(N-1)$$

$$+ \; [R+B^TSB]\,u(N-1)$$

$$\left.\frac{\partial J[x(N-1),u(N-1)]}{\partial u(N-1)}\right|_{u^o(N-1)} = 0 \Rightarrow$$

$$\left[B^TSA\right]x(N-1) + \left[R+B^TSB\right]u^o(N-1) = 0$$



## Computing $u^0(N-1)$

$$\left[B^TSA\right]x(N-1) + \left[R+B^TSB\right]u^o(N-1) = 0$$

$$\boxed{u^o(N-1) = -\left[R+B^TSB\right]^{-1}\left[B^TSA\right]x(N-1)}$$

$$R = R^T \succ 0 \qquad B^TSB \succeq 0 \qquad \Rightarrow R + B^TSB \succ 0$$

***A linear feedback function !!***

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

## Find the optimal cost function $J^o[x(N-1)]$

$$J^o[x(N-1)] = \min_{u(N-1)} \{J[x(N-1), u(N-1)]\}$$

$$= J[x(N-1), u^o(N-1)]$$

where,

$$u^o(N-1) = -\left[R + B^T S B\right]^{-1} \left[B^T S A\right] x(N-1)$$



## Computing $J^o[x(N-1)]$

$$J^o[x(N-1)] = \frac{1}{2} x^T(N-1) \left[Q + A^T S A\right] x(N-1)$$
$$+ \; x^T(N-1) \left[A^T S B\right] \underline{u^o(N-1)}$$
$$+ \; \frac{1}{2} \underline{u^{oT}(N-1)} \left[R + B^T S B\right] \underline{u^o(N-1)}$$

we need to substitute the following expression,

$$u^o(N-1) = -\left[R + B^T S B\right]^{-1} \left[B^T S A\right] x(N-1)$$



## Computing $J^o[x(N-1)]$

Doing the algebra:

$$J^o[x(\underline{N-1})] =$$

$$\frac{1}{2} x^T\underline{(N-1)} \underbrace{\left\{Q + A^T S A - A^T S B \left[R + B^T S B\right]^{-1} B^T S A\right\}} x\underline{(N-1)}$$

*Just a square and symmetric matrix*

$$P(\underline{N-1})$$



## Computing $J^o[x(N-1)]$

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

A quadratic function of ***x(N-1)*** !!

where,

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

## Computing $J^o[x(N-1)]$

For $m = N, N\text{-}1$ we have:

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

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

Where,

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

Set: $P(N) = S$



## Computing $J^o[x(N-1)]$

For $m = N, N\text{-}1$ we have:

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

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

Where, $P(N) = S$

$$\begin{aligned} P(N-1) \;&=\; Q + A^T P(N) A \\ &\quad - A^T P(N) B \left[R + B^T P(N) B\right]^{-1} B^T P(N) A \end{aligned}$$



## Solving the Bellman Equation for a LQR

Thus, for $m = N, N\text{-}1$ we have:

$$P(N) = S$$

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

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

$$u^o(N-1) = -\left[R + B^T P(N) B\right]^{-1} \left[B^T P(N) A\right] x(N-1)$$

**These equations are entirely recursive!!**



## The optimal cost function $J^o[x(k)]$

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

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

$P(N) = S$ *boundary condition*

***Computation of $P$ entirely recursive !!***
***(starting from N and going backwards)***

# The optimal control $u^o(k)$

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

**Time varying linear feedback law !!**

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

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



# Finite Horizon LQR Solution:

Thus, for $k = 0, \ldots N-1$ we have:

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

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

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

Riccati difference equation (computed backwards):

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

$$P(N) = S$$



# Summary

- Bellman's dynamic programming invention was a major breakthrough in the theory of multistage decision processes and optimization
- Key idea's
  - Principle of optimality
  - Computation of optimal cost function

illustrated with a simple multi-stage example



# Summary

- Bellman's equation:

$$J^o[x(m)] = \min_{u(m)} \{L[x(m), u(m)] + J^o[x(m+1)]\}$$

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

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

# 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

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