

# ELEC-E8101: Digital and Optimal Control

## *Lecture 10* *Optimal Control in State-Space*

**Gökhan Alcan**

Office number: 2575, Maarintie 8 Building

Email: [gokhan.alcan@aalto.fi](mailto:gokhan.alcan@aalto.fi)

# State-Space Representation (*recap*)

- Often, physical equations that describe a system mathematically are already available in *state-space representation*, where the states are some set of physical variables (e.g., displacements, velocities, etc.), i.e.,

$$\begin{array}{ll} \text{System model: } \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t), & x(0) = x_0 & \dim(\mathbf{x}) = n \\ \text{Observation model: } \mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t) & & \dim(\mathbf{u}) = r \\ & & \dim(\mathbf{y}) = p \end{array}$$

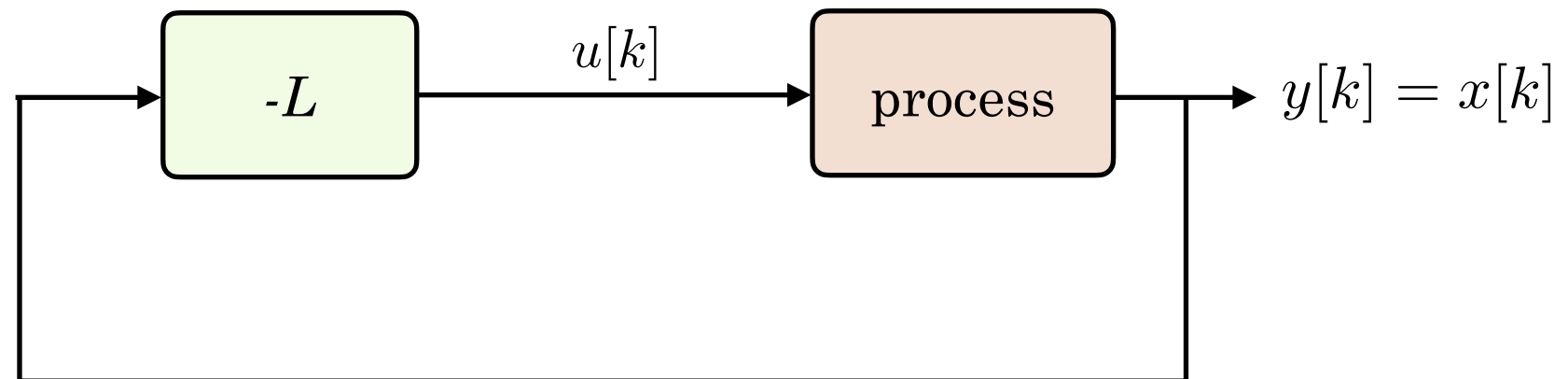
# State-Space Representation (*recap*)

- Often, physical equations that describe a system mathematically are already available in *state-space representation*, where the states are some set of physical variables (e.g., displacements, velocities, etc.), i.e.,

$$\begin{array}{ll} \text{System model: } \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t), & x(0) = x_0 & \dim(\mathbf{x}) = n \\ \text{Observation model: } \mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t) & & \dim(\mathbf{u}) = r \\ & & \dim(\mathbf{y}) = p \end{array}$$

- Advantages of the system description using the state-space representation in comparison to the conventional methods are:
  - Possibility to describe the state of the entire system each time; unlike the transfer function, which connects the input  $u(t)$  with the output  $y(t)$
  - They facilitate the solution of control problems, such as stability and optimized control
  - The simulation and scheduling in computer systems is quite easy since they are represented by a set of linear differential (later difference) equations
  - They are also able to describe some nonlinear systems, which cannot be done using the transfer function

# State-Space Representation (*recap*)

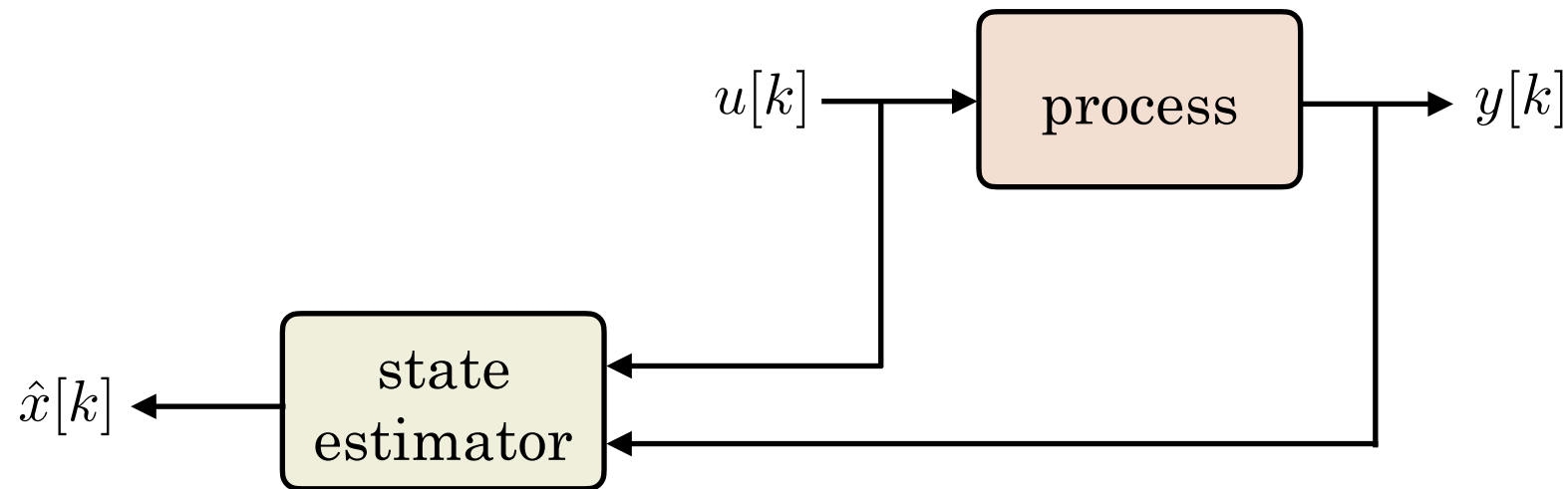


- **Designing controllers** based on state-space models to obtain specific closed-loop characteristics of the system

$$\left. \begin{array}{l} \mathbf{x}[k+1] = \Phi \mathbf{x}[k] + \Gamma \mathbf{u}[k] \\ \mathbf{u}[k] = -L \mathbf{x}[k] \end{array} \right\} \Rightarrow \mathbf{x}[k+1] = \Phi \mathbf{x}[k] - \Gamma L \mathbf{x}[k] = \underbrace{(\Phi - \Gamma L)}_{\Phi_{cl}} \mathbf{x}[k]$$



# State-Space Representation (*recap*)

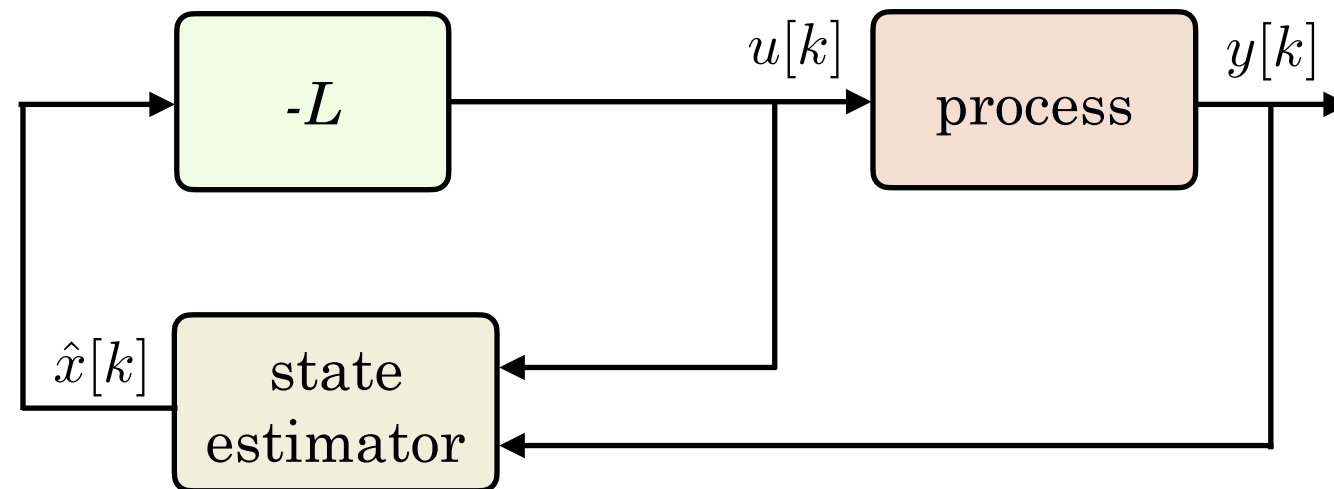


- **Designing observers** based on state-space models to obtain specific closed-loop characteristics of the estimation error

$$\begin{cases} \mathbf{x}[k+1] &= \Phi \mathbf{x}[k] + \Gamma u[k], \\ y[k] &= C \mathbf{x}[k] \end{cases}$$

$$\tilde{\mathbf{x}}[k+1] = \underbrace{(\Phi - KC)}_{\Phi_o} \tilde{\mathbf{x}}[k]$$

# State-Space Representation (*recap*)



- By combining the state-observer and state-feedback controller it is possible to **design a controller based on output measurements (feedback)**

$$\begin{cases} \begin{bmatrix} \mathbf{x}[k+1] \\ \tilde{\mathbf{x}}[k+1] \end{bmatrix} = \begin{bmatrix} \Phi - \Gamma L & \Gamma L \\ \mathbf{0} & \Phi - KC \end{bmatrix} \begin{bmatrix} \mathbf{x}[k] \\ \tilde{\mathbf{x}}[k] \end{bmatrix} \\ y[k] = \begin{bmatrix} C & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}[k] \\ \tilde{\mathbf{x}}[k] \end{bmatrix} \end{cases}$$

# Learning outcomes

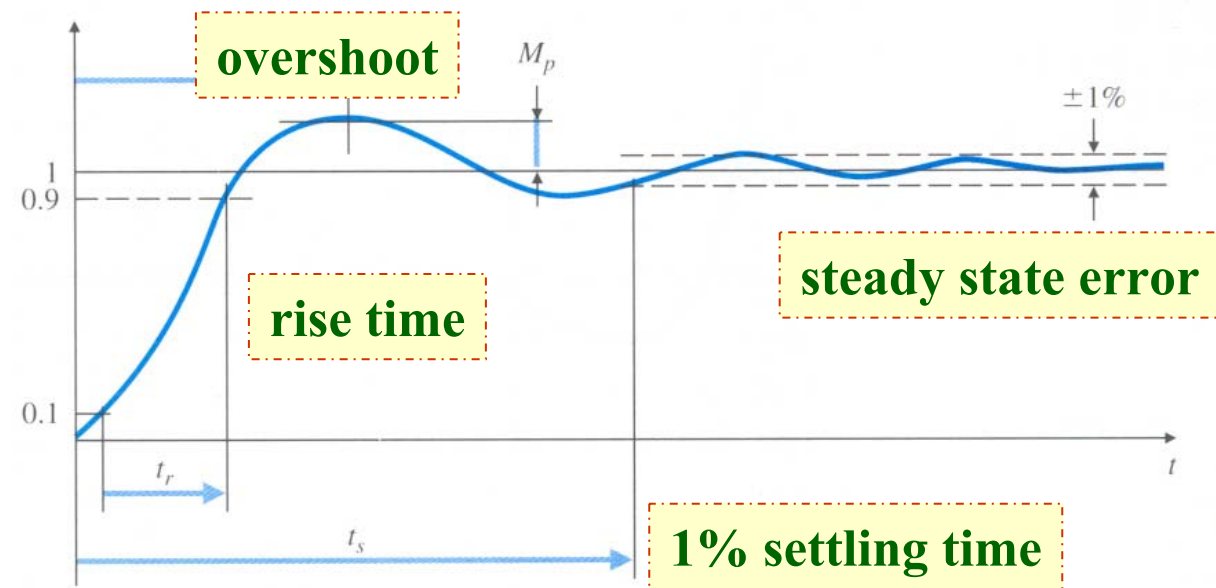
By the end of *this* lecture, you should be able to:

- **Understand** the principle of optimality
- **Understand** the Dynamic Programming
- **Design optimal controllers** based on LQ problem formulation



# Introduction

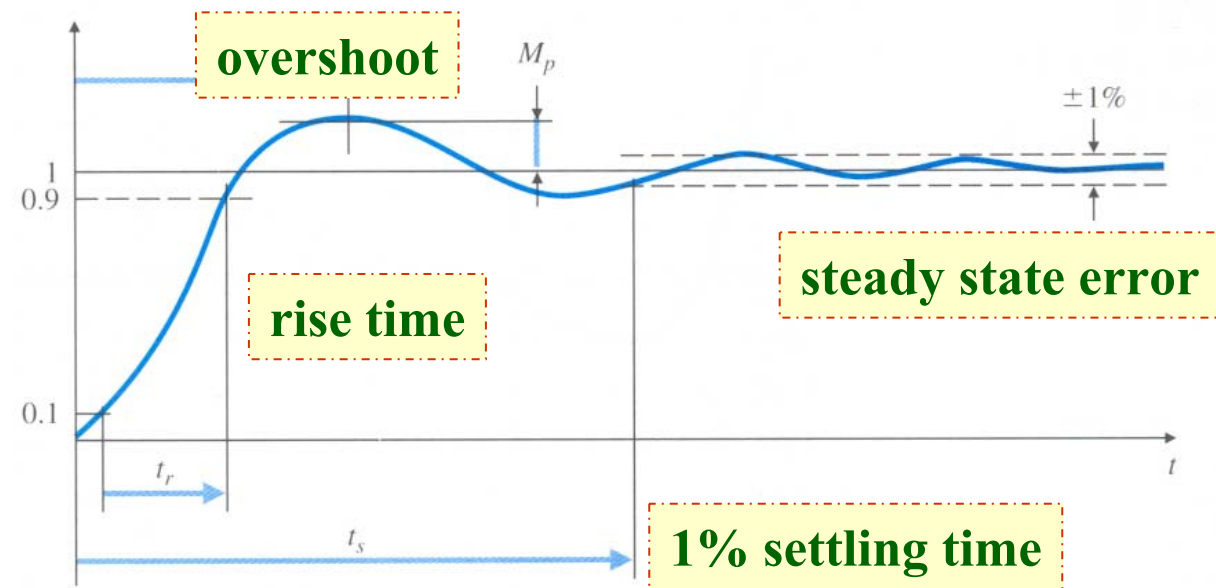
- Classical control system design - design parameters of an “acceptable” system, defined in terms of **time** and **frequency** domain criteria, e.g.,
  - rise time
  - settling time
  - peak overshoot
  - gain and phase margin
  - bandwidth



# Introduction

- Classical control system design - design parameters of an “acceptable” system, defined in terms of **time** and **frequency** domain criteria, e.g.,

- rise time
- settling time
- peak overshoot
- gain and phase margin
- bandwidth

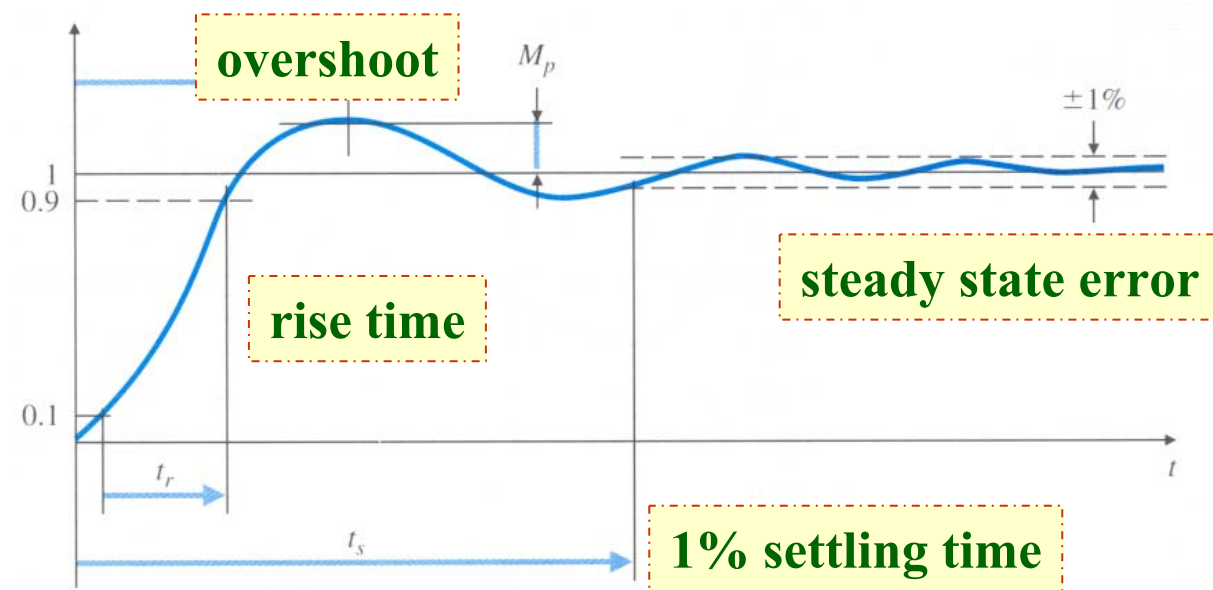


- The *main design parameters* (in this course, so far) have been the locations of the closed-loop poles

# Introduction

- Classical control system design - design parameters of an “acceptable” system, defined in terms of **time** and **frequency** domain criteria, e.g.,

- rise time
- settling time
- peak overshoot
- gain and phase margin
- bandwidth

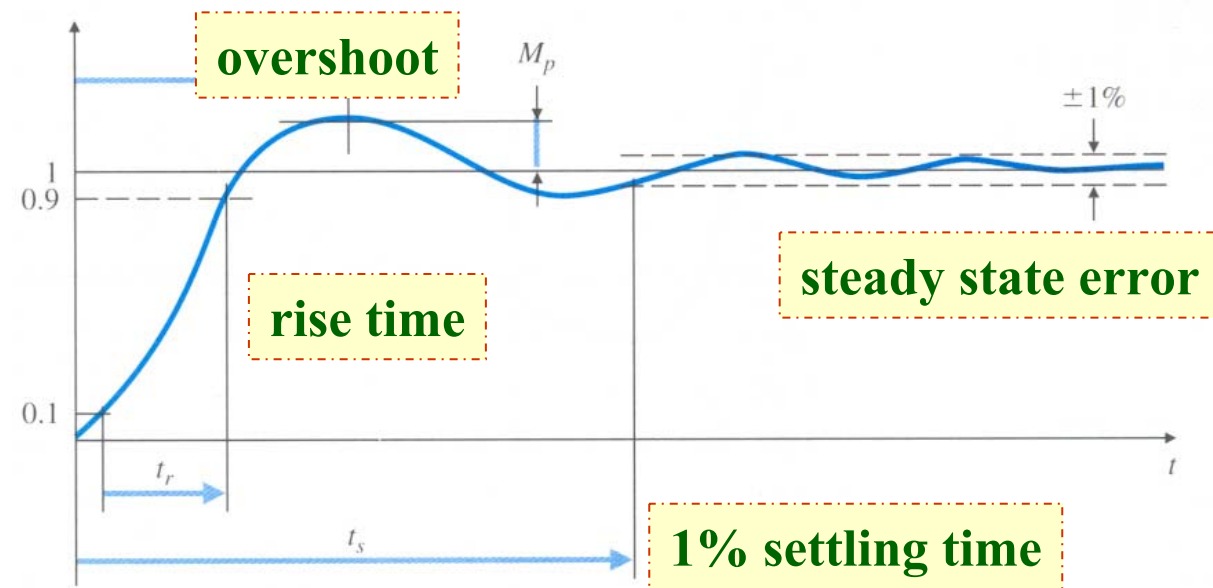


- The *main design parameters* (in this course, so far) have been the locations of the closed-loop poles
- Results **limited to single-input single-output (SISO) systems**

# Introduction

- Classical control system design - design parameters of an “acceptable” system, defined in terms of **time** and **frequency** domain criteria, e.g.,

- rise time
- settling time
- peak overshoot
- gain and phase margin
- bandwidth



- The *main design parameters* (in this course, so far) have been the locations of the closed-loop poles
- Results **limited to single-input single-output (SISO) systems**
- Different performance criteria must be satisfied by complex multi-input-multi-output (MIMO) systems, e.g., the design of a spacecraft attitude that minimizes fuel expenditure - not solvable by classical methods!

# Motivation for optimal control

- **Motivation for optimal control**
  - Get the best performance
  - Learn the limits



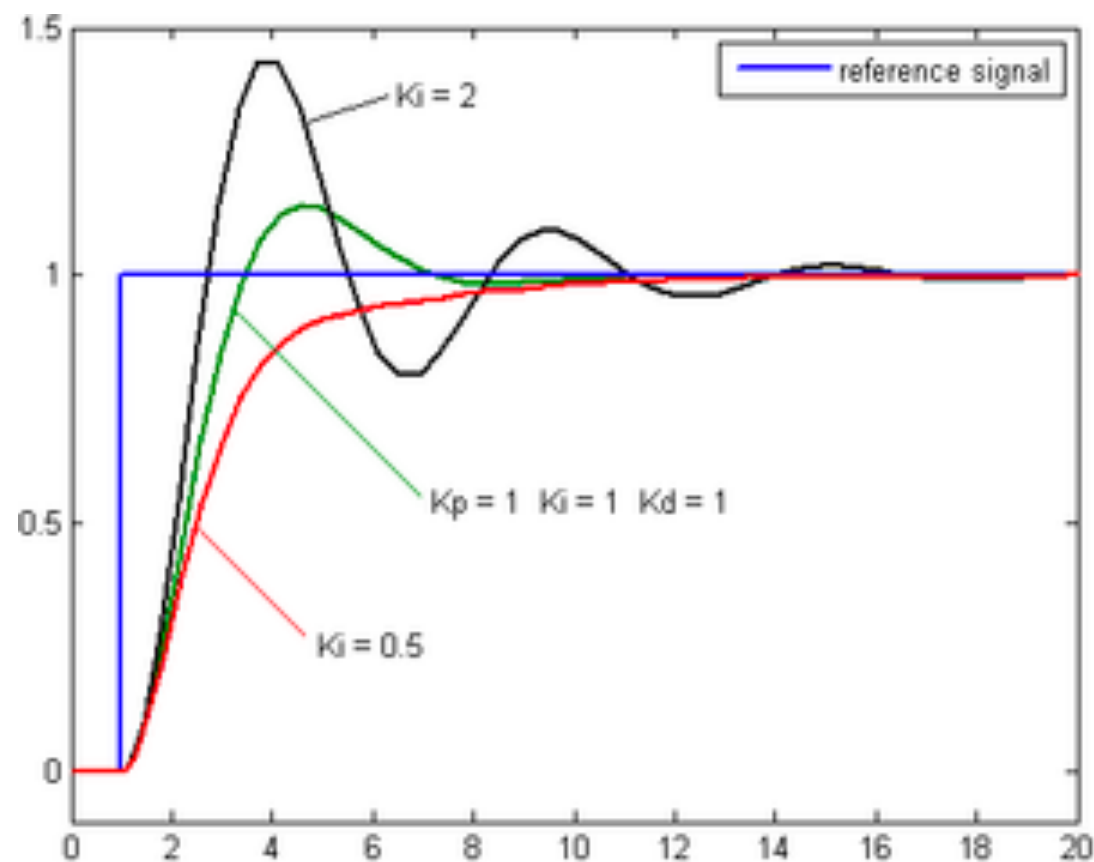
# Motivation for optimal control

- **Motivation for optimal control**

- Get the best performance
- Learn the limits

- **Control design as optimization**

1) Optimization cost (time or frequency domain). In time domain:



- Lower rise time  $\Rightarrow$  bigger overshoot
- Lower overshoot  $\Rightarrow$  large rise time

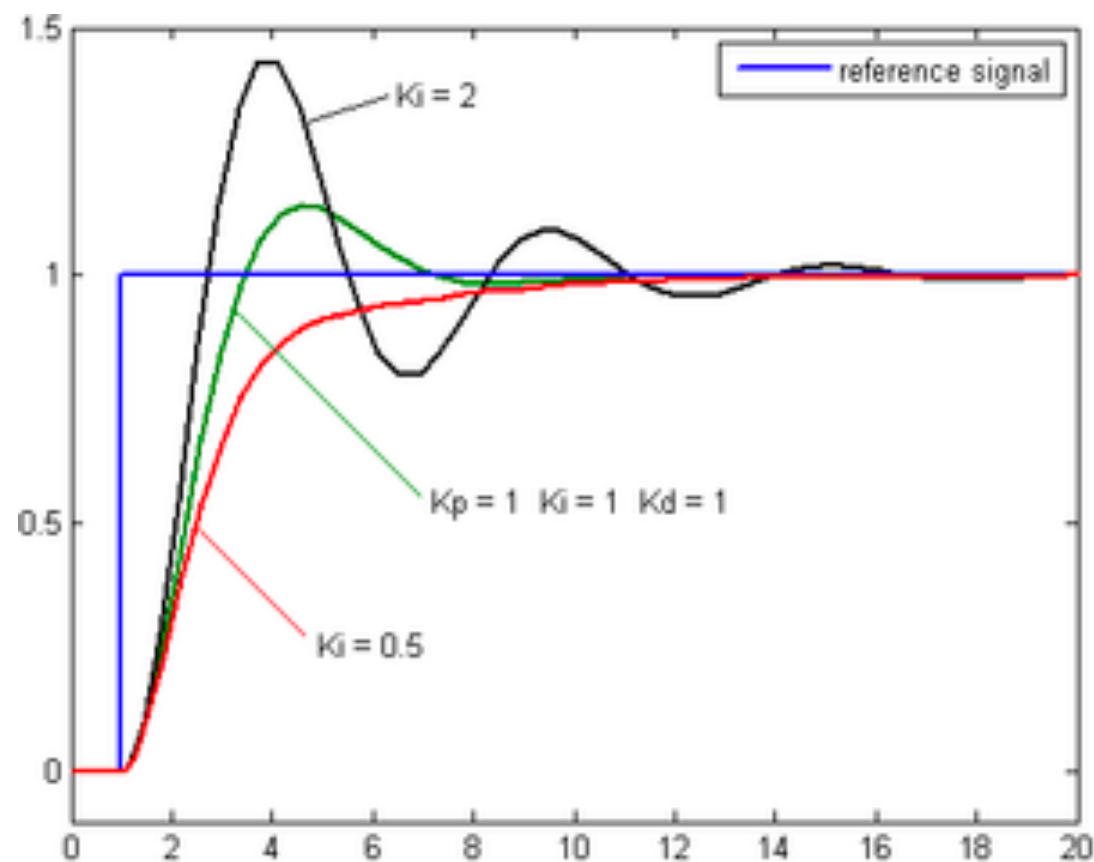
# Motivation for optimal control

- **Motivation for optimal control**

- Get the best performance
- Learn the limits

- **Control design as optimization**

1) Optimization cost (time or frequency domain). In time domain:



- Lower rise time  $\Rightarrow$  bigger overshoot
- Lower overshoot  $\Rightarrow$  large rise time

Handle  
by

- Imposing additional constraints
- **Integral criteria**

# Motivation for optimal control (cont'd)

- Integral criteria: let the error  $e = r - y$ , then one can try and minimize the error, e.g.,

$$\int_0^{\infty} |e| dt, \quad \int_0^{\infty} t|e| dt, \quad \int_0^{\infty} e^2 dt$$

# Motivation for optimal control (cont'd)

- Integral criteria: let the error  $e = r - y$ , then one can try and minimize the error, e.g.,

$$\int_0^{\infty} |e| dt, \quad \int_0^{\infty} t|e| dt, \quad \int_0^{\infty} e^2 dt$$

- However, there is a performance vs effort trade off; to balance it


$$\int_0^{\infty} (qe^2 + ru^2) dt$$

# Motivation for optimal control (cont'd)

- Integral criteria: let the error  $e = r - y$ , then one can try and minimize the error, e.g.,

$$\int_0^\infty |e|dt, \quad \int_0^\infty t|e|dt, \quad \int_0^\infty e^2dt$$

- However, there is a performance vs effort trade off; to balance it

$$\int_0^\infty (qe^2 + ru^2) dt$$



weights

# Motivation for optimal control (cont'd)

- Integral criteria: let the error  $e = r - y$ , then one can try and minimize the error, e.g.,

$$\int_0^{\infty} |e| dt, \quad \int_0^{\infty} t|e| dt, \quad \int_0^{\infty} e^2 dt$$

- However, there is a performance vs effort trade off; to balance it

$$\int_0^{\infty} (qe^2 + ru^2) dt$$


weights

## 2) Optimization variables


- **Controllers** (intractable if we do not make restrictions) → we need to fix the controllers and optimize over parameters (e.g., PID coefficients).
  - Have to express the cost  $J$  as a function of the PID parameters, which is hard and it results to numerical simulation solutions
  - Need to impose conditions for closed loop stability, generally *nonconvex* problem

# Motivation for optimal control (cont'd)

- Integral criteria: let the error  $e = r - y$ , then one can try and minimize the error, e.g.,

$$\int_0^{\infty} |e| dt, \quad \int_0^{\infty} t|e| dt, \quad \int_0^{\infty} e^2 dt$$

- However, there is a performance vs effort trade off; to balance it

$$\int_0^{\infty} (qe^2 + ru^2) dt$$


weights

## 2) Optimization variables

- **Controllers** (intractable if we do not make restrictions) → we need to fix the controllers and optimize over parameters (e.g., PID coefficients).
  - Have to express the cost  $J$  as a function of the PID parameters, which is hard and it results to numerical simulation solutions
  - Need to impose conditions for closed loop stability, generally *nonconvex* problem
- **Control signals** Not easy in continuous time, but in discrete-time it is just a sequence of numbers!

# Introduction

- **Optimal control theory:** a new and direct approach to the synthesis of these complex systems



# Introduction

- **Optimal control theory:** a new and direct approach to the synthesis of these complex systems
- **Objective:** determine the control signals that allow a process to satisfy the physical constraints and at the same time optimize some performance criterion

# Introduction

- **Optimal control theory:** a new and direct approach to the synthesis of these complex systems
- **Objective:** determine the control signals that allow a process to satisfy the physical constraints and at the same time optimize some performance criterion
- **In this lecture:**
  - the process is still assumed to be **linear**, but it may be **time-varying** LTV
  - the process may have **several** inputs and outputs MIMO

# Introduction

- **Optimal control theory:** a new and direct approach to the synthesis of these complex systems
- **Objective:** determine the control signals that allow a process to satisfy the physical constraints and at the same time optimize some performance criterion
- **In this lecture:**
  - the process is still assumed to be linear, but it may be time-varying
  - the process may have several inputs and outputs
- The problem is formulated to minimize a criterion: a quadratic function of the states and the control signals - **Linear Quadratic (LQ) control problem**

# Introduction

- **Optimal control theory:** a new and direct approach to the synthesis of these complex systems
- **Objective:** determine the control signals that allow a process to satisfy the physical constraints and at the same time optimize some performance criterion
- **In this lecture:**
  - the process is still assumed to be linear, but it may be time-varying
  - the process may have several inputs and outputs
  - *process and measurement noise* are introduced in the models
- The problem is formulated to minimize a criterion: a quadratic function of the states and the control signals - **Linear Quadratic (LQ) control problem**
- If Gaussian stochastic disturbances are allowed in the process models, then the problem is called **Linear Quadratic Gaussian (LQG) control problem**

# LQ problem (continuous-time)

- Continuous-time model:

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t), \quad \mathbf{x}(t_0) = \mathbf{x}_0$$

$$\mathbf{y}(t) = C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t)$$

# LQ problem (continuous-time)

- Continuous-time model:

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t), \quad \mathbf{x}(t_0) = \mathbf{x}_0$$

$$\mathbf{y}(t) = C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t)$$

- Criterion to be minimized:

$$\min_u J \triangleq \min_u \int_{t_0}^{t_f} \{ \mathbf{x}(t)^T Q(t) \mathbf{x}(t) + \mathbf{u}(t)^T R(t) \mathbf{u}(t) \} dt$$

i.e., find control law  $u$ , such that the criterion is minimized.

# LQ problem (continuous-time)

- Continuous-time model:

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t), \quad \mathbf{x}(t_0) = \mathbf{x}_0$$

$$\mathbf{y}(t) = C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t)$$

- Criterion to be minimized:

$$\min_u J \triangleq \min_u \int_{t_0}^{t_f} \{ \mathbf{x}(t)^T Q(t) \mathbf{x}(t) + \mathbf{u}(t)^T R(t) \mathbf{u}(t) \} dt$$

i.e., find control law  $u$ , such that the criterion is minimized.

- **Design parameters** are: the matrices  $Q$  and  $R$  in the cost function.

# LQ problem (continuous-time)

- Continuous-time model:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t), & \mathbf{x}(t_0) &= \mathbf{x}_0 \\ \mathbf{y}(t) &= C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t)\end{aligned}$$

- Criterion to be minimized:

$$\min_u J \triangleq \min_u \int_{t_0}^{t_f} \{ \mathbf{x}(t)^T Q(t) \mathbf{x}(t) + \mathbf{u}(t)^T R(t) \mathbf{u}(t) \} dt$$

i.e., find control law  $u$ , such that the criterion is minimized.

- **Design parameters** are: the matrices  $Q$  and  $R$  in the cost function.
- The final time of the optimization horizon  $t_f$  can be finite or  $\infty$ .  
Initial state  $\mathbf{x}(t_0)$  is given.  
Final state  $\mathbf{x}(t_f)$  can be fixed or free



# LQ problem (continuous-time)

- Continuous-time model:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t), & \mathbf{x}(t_0) &= \mathbf{x}_0 \\ y(t) &= C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t)\end{aligned}$$

- Criterion to be minimized:

$$\min_u J \triangleq \min_u \int_{t_0}^{t_f} \{ \mathbf{x}(t)^T Q(t) \mathbf{x}(t) + \mathbf{u}(t)^T R(t) \mathbf{u}(t) \} dt$$

i.e., find control law  $u$ , such that the criterion is minimized.

- **Design parameters** are: the matrices  $Q$  and  $R$  in the cost function.
- The final time of the optimization horizon  $t_f$  can be finite or  $\infty$ .  
Initial state  $\mathbf{x}(t_0)$  is given.  
Final state  $\mathbf{x}(t_f)$  can be fixed or free
- What is the role of  $Q$ ? What is the role of  $R$ ?

# LQ problem (discrete-time)

- Discrete-time model:

$$\begin{aligned}\mathbf{x}[k+1] &= \Phi \mathbf{x}[k] + \Gamma \mathbf{u}[k], & x[k_0] &= x_0 \\ \mathbf{y}[k] &= C \mathbf{x}[k] + D \mathbf{u}[k]\end{aligned}$$

# LQ problem (discrete-time)

- Discrete-time model:

$$\begin{aligned}\mathbf{x}[k+1] &= \Phi \mathbf{x}[k] + \Gamma \mathbf{u}[k], & x[k_0] &= x_0 \\ \mathbf{y}[k] &= C \mathbf{x}[k] + D \mathbf{u}[k]\end{aligned}$$

- Criterion to be minimized:

$$\min_u J \triangleq \min_u \left[ \sum_{k=k_0}^{N-1} \left( \mathbf{x}[k]^T Q[k] \mathbf{x}[k] + \mathbf{u}[k]^T R[k] \mathbf{u}[k] \right) + \mathbf{x}[N]^T Q[N] \mathbf{x}[N] \right]$$

i.e., find control law (sequence)  $u$ , such that the criterion is minimized.

# LQ problem (discrete-time)

- Discrete-time model:

$$\mathbf{x}[k+1] = \Phi \mathbf{x}[k] + \Gamma \mathbf{u}[k], \quad \mathbf{x}[k_0] = \mathbf{x}_0$$

$$\mathbf{y}[k] = C \mathbf{x}[k] + D \mathbf{u}[k]$$

- Criterion to be minimized:

$$\min_u J \triangleq \min_u \left[ \underbrace{\sum_{k=k_0}^{N-1} \left( \mathbf{x}[k]^T Q[k] \mathbf{x}[k] + \mathbf{u}[k]^T R[k] \mathbf{u}[k] \right)}_{\text{Running Stage Cost}} + \underbrace{\mathbf{x}[N]^T Q[N] \mathbf{x}[N]}_{\text{Final Terminal Cost}} \right]$$

i.e., find control law (sequence)  $u$ , such that the criterion is minimized.

# LQ problem (discrete-time)

- Discrete-time model:

$$\begin{aligned}\mathbf{x}[k+1] &= \Phi \mathbf{x}[k] + \Gamma \mathbf{u}[k], & x[k_0] &= x_0 \\ \mathbf{y}[k] &= C \mathbf{x}[k] + D \mathbf{u}[k]\end{aligned}$$

- Criterion to be minimized:

$$\min_u J \triangleq \min_u \left[ \sum_{k=k_0}^{N-1} \left( \mathbf{x}[k]^T Q[k] \mathbf{x}[k] + \mathbf{u}[k]^T R[k] \mathbf{u}[k] \right) + \mathbf{x}[N]^T Q[N] \mathbf{x}[N] \right]$$

i.e., find control law (sequence)  $u$ , such that the criterion is minimized.

- The design parameters are: the matrices  $Q$  and  $R$  in the cost function and the sampling period.

# LQ problem (discrete-time)

- Discrete-time model:

$$\begin{aligned}\mathbf{x}[k+1] &= \Phi \mathbf{x}[k] + \Gamma \mathbf{u}[k], & x[k_0] &= x_0 \\ \mathbf{y}[k] &= C \mathbf{x}[k] + D \mathbf{u}[k]\end{aligned}$$

- Criterion to be minimized:

$$\min_u J \triangleq \min_u \left[ \sum_{k=k_0}^{N-1} \left( \mathbf{x}[k]^T Q[k] \mathbf{x}[k] + \mathbf{u}[k]^T R[k] \mathbf{u}[k] \right) + \mathbf{x}[N]^T Q[N] \mathbf{x}[N] \right]$$

i.e., find control law (sequence)  $u$ , such that the criterion is minimized.

- The design parameters are: the matrices  $Q$  and  $R$  in the cost function and the sampling period. *Why?*

# LQ problem (discrete-time)

- Discrete-time model:

$$\begin{aligned}\mathbf{x}[k+1] &= \Phi \mathbf{x}[k] + \Gamma \mathbf{u}[k], & x[k_0] &= x_0 \\ \mathbf{y}[k] &= C \mathbf{x}[k] + D \mathbf{u}[k]\end{aligned}$$

- Criterion to be minimized:

$$\min_u J \triangleq \min_u \left[ \sum_{k=k_0}^{N-1} \left( \mathbf{x}[k]^T Q[k] \mathbf{x}[k] + \mathbf{u}[k]^T R[k] \mathbf{u}[k] \right) + \mathbf{x}[N]^T Q[N] \mathbf{x}[N] \right]$$

i.e., find control law (sequence)  $u$ , such that the criterion is minimized.

- The design parameters are: the **matrices  $Q$  and  $R$**  in the cost function and the **sampling period**.
- The final time of the optimization horizon  $N$  can be finite or  $\infty$ .  
Initial state  $\mathbf{x}[k_0]$  is given.  
Final state  $\mathbf{x}[N]$  can be fixed or free

# Dynamic programming

Those who cannot remember the past  
are condemned to repeat it.

-Dynamic Programming



# Dynamic programming

Those who cannot remember the past  
are condemned to repeat it.

-Dynamic Programming

- **Main idea of Dynamic Programming (DP):**  
to always remember answers to the sub-problems you've already solved

# Dynamic programming

Those who cannot remember the past  
are condemned to repeat it.

-Dynamic Programming

- **Main idea of Dynamic Programming (DP):**  
to always remember answers to the sub-problems you've already solved
- If you are given a problem, which can be broken down into smaller sub-problems, and these smaller sub-problems can still be broken into smaller ones - and if you manage to find out that there are some overlapping sub-problems, then you've encountered a DP problem.

# Dynamic programming

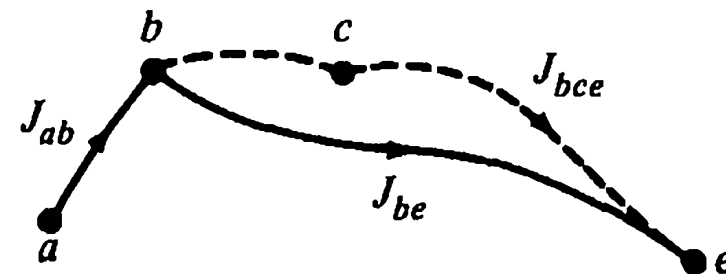
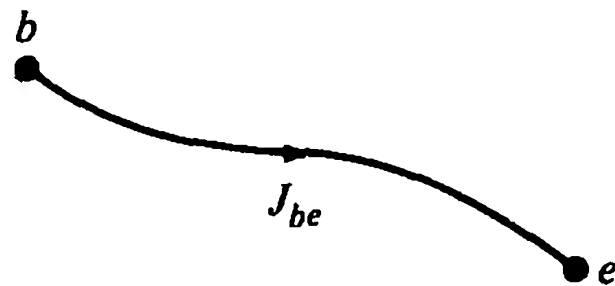
- *“An optimal policy has the property that no matter what the previous decision (i.e., controls) have been, the remaining decisions must constitute an optimal policy with regard to the state resulting from those previous decisions.”*

*- Bellman, 1957*

# Dynamic programming

- “An optimal policy has the property that no matter what the previous decision (i.e., controls) have been, the remaining decisions must constitute an optimal policy with regard to the state resulting from those previous decisions.”

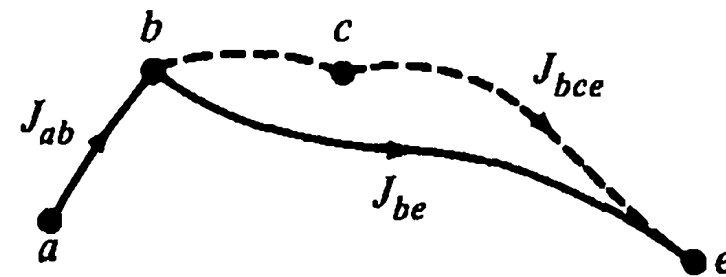
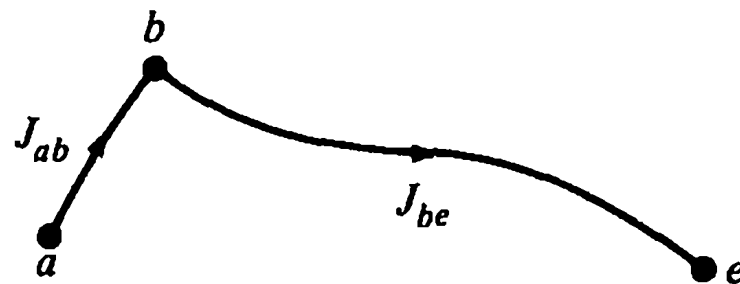
- Bellman, 1957



# Dynamic programming

- “An optimal policy has the property that no matter what the previous decision (i.e., controls) have been, the remaining decisions must constitute an optimal policy with regard to the state resulting from those previous decisions.”

- Bellman, 1957

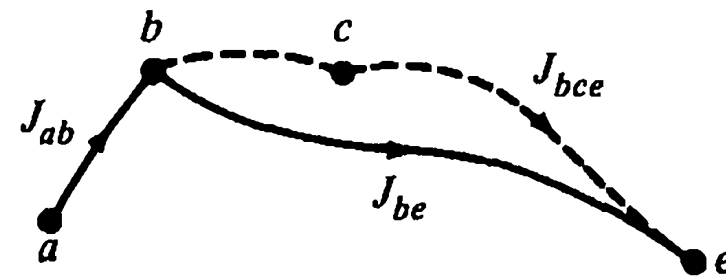
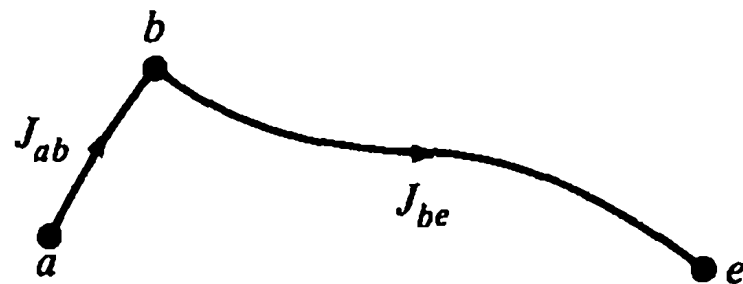


- By applying this principle the number of candidates for the optimal solution can be reduced.

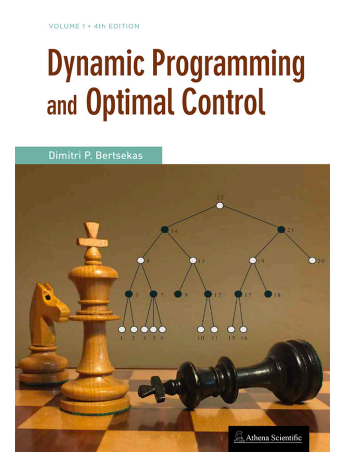
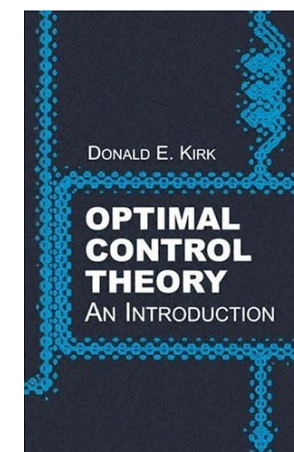
# Dynamic programming

- “An optimal policy has the property that no matter what the previous decision (i.e., controls) have been, the remaining decisions must constitute an optimal policy with regard to the state resulting from those previous decisions.”

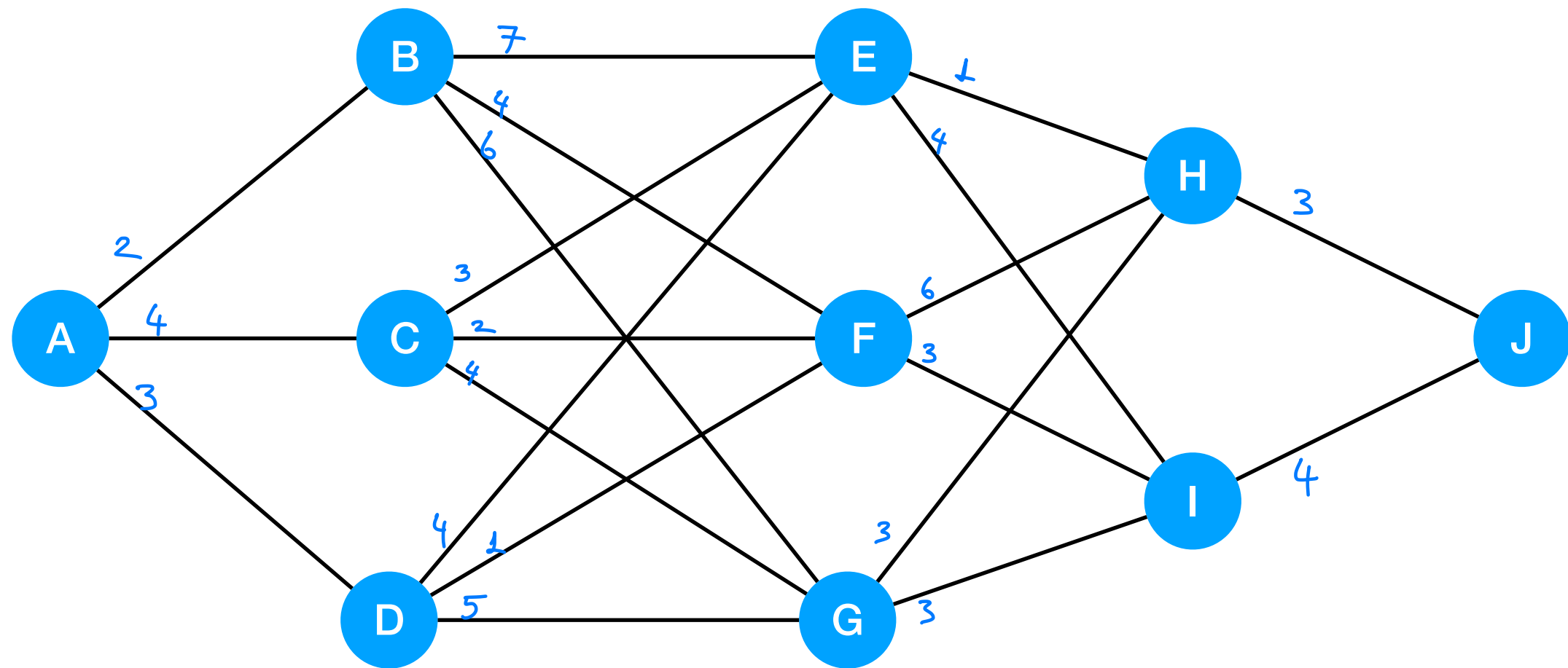
- Bellman, 1957



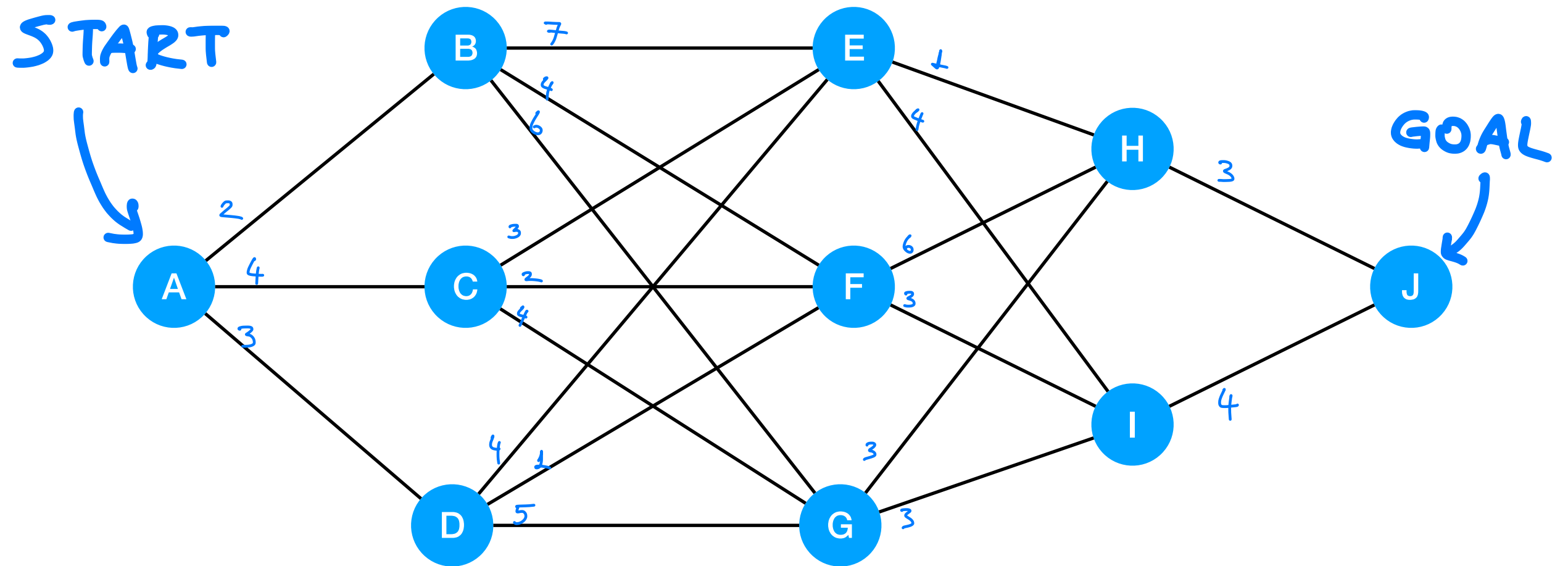
- By applying this principle the number of candidates for the optimal solution can be reduced.
- Recommended books for Dynamic Programming:
  - Dimitri Bertsekas (2017), “Dynamic Programming and Optimal Control”
  - Donald E. Kirk (1998), “Optimal Control Theory”



# Principle of Optimality



# Principle of Optimality

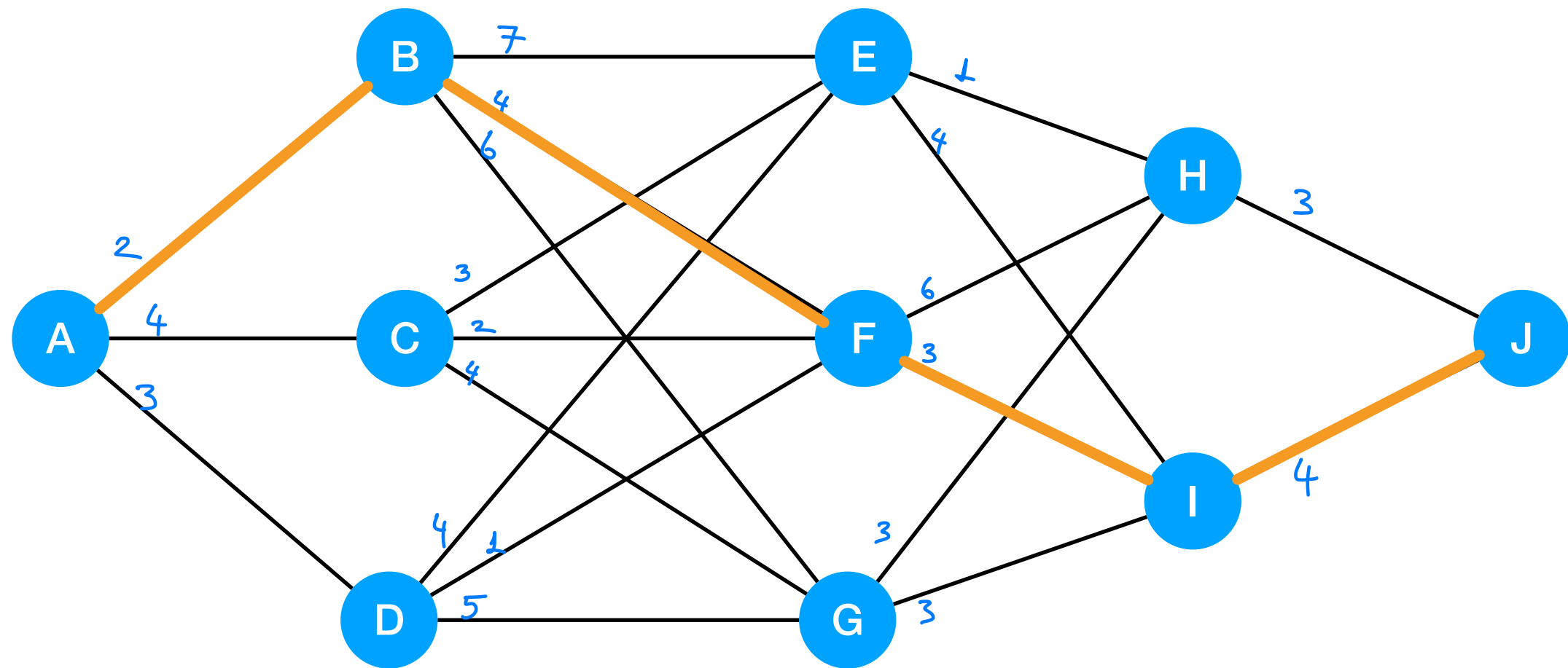




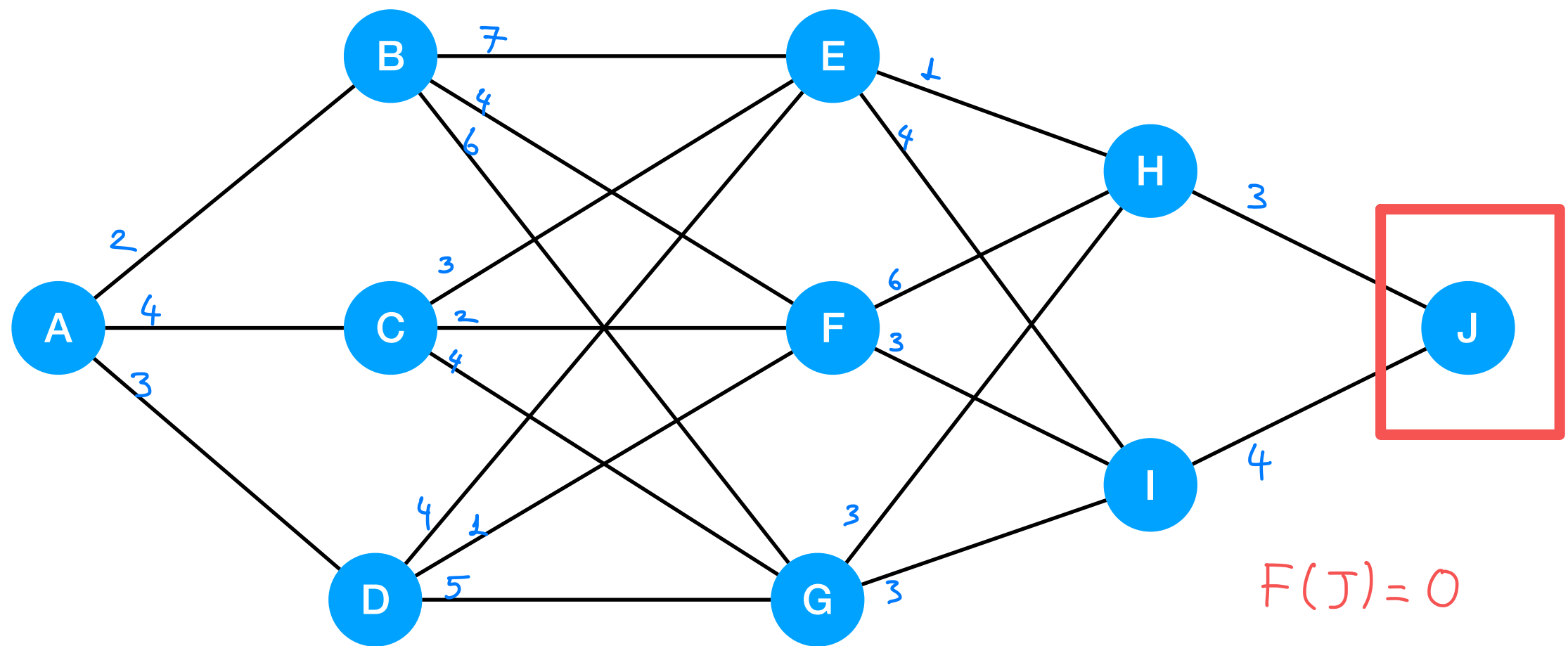
# Principle of Optimality

Naive Greedy Approach

$$2 + 4 + 3 + 4 = 13$$



# Principle of Optimality



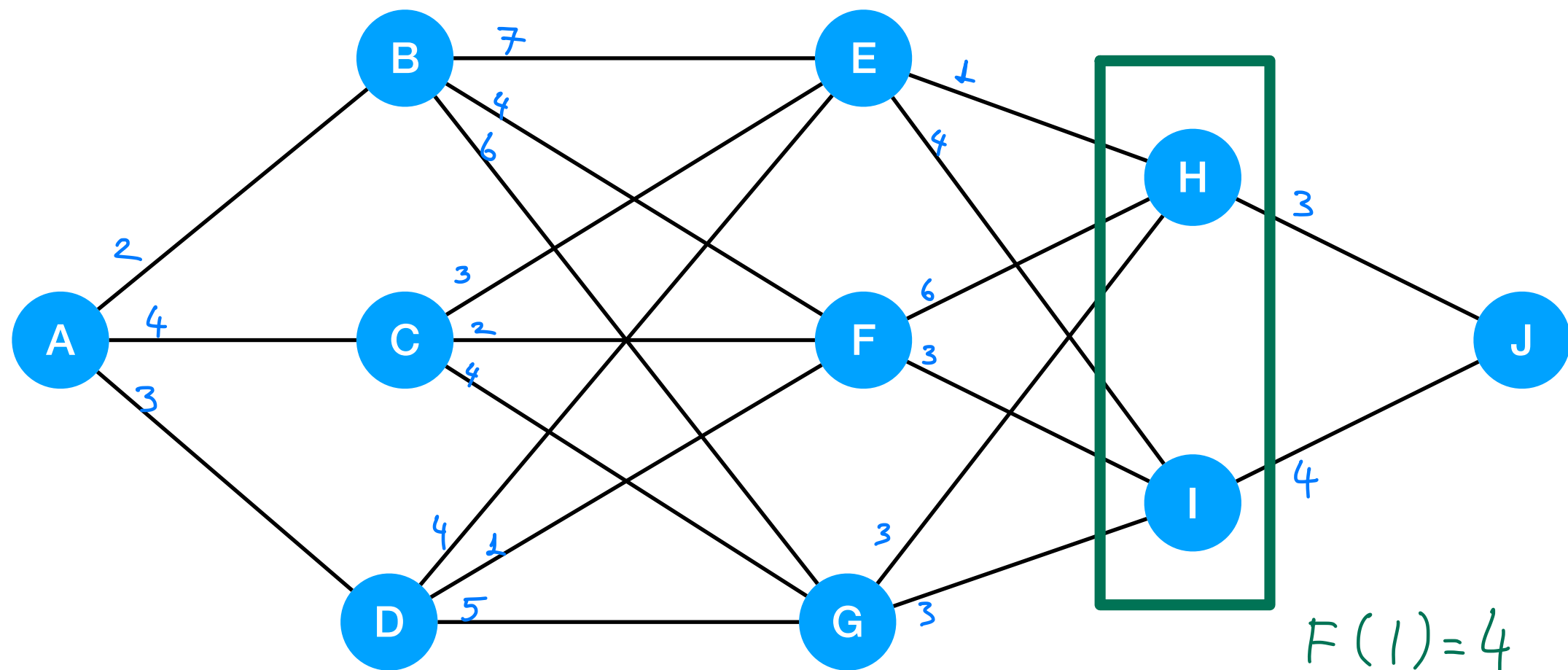
$F(\cdot)$  : cost-to-go

$$F(J) = 0$$

# Principle of Optimality

$$F(J) = 0$$

$$F(H) = 3$$



$F(.)$  : cost-to-go

# Principle of Optimality

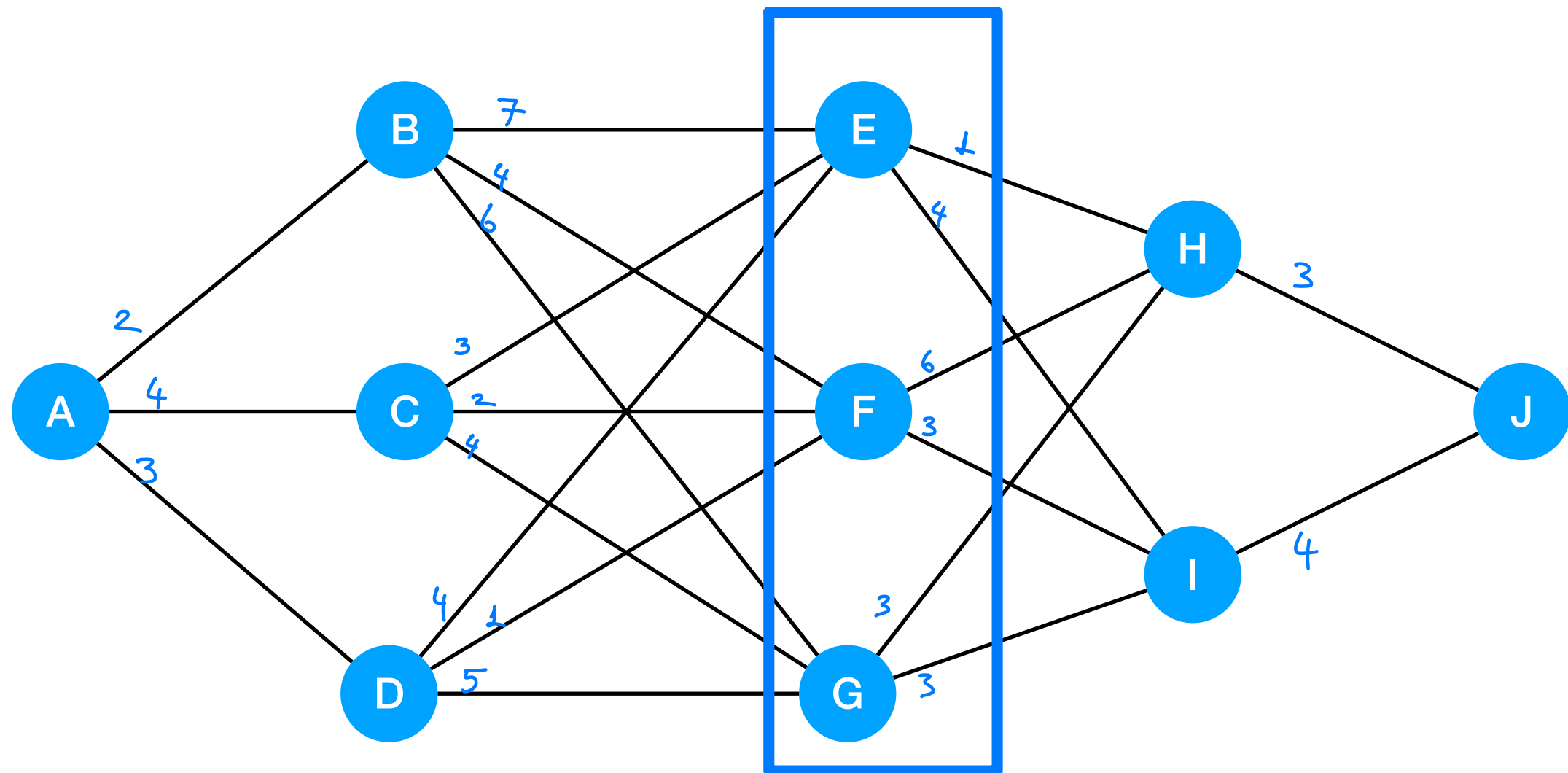
$$F(J) = 0$$

$$F(H) = 3$$

$$F(I) = 4$$

$$F(E) = \min \left\{ 1 + F(H), 4 + F(I) \right\}$$

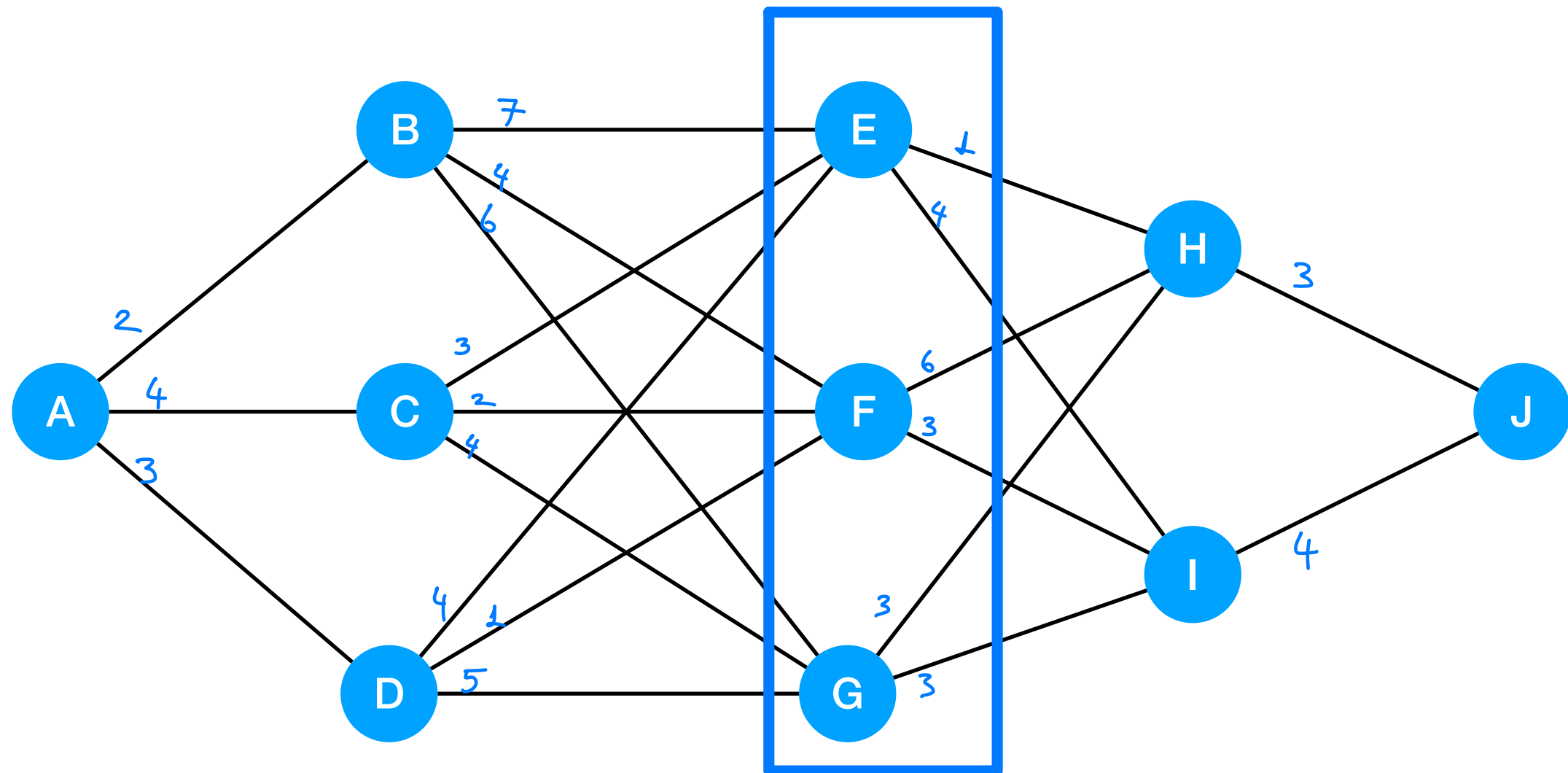
$$F(E) = 4$$



$F(.)$  : cost-to-go

# Principle of Optimality

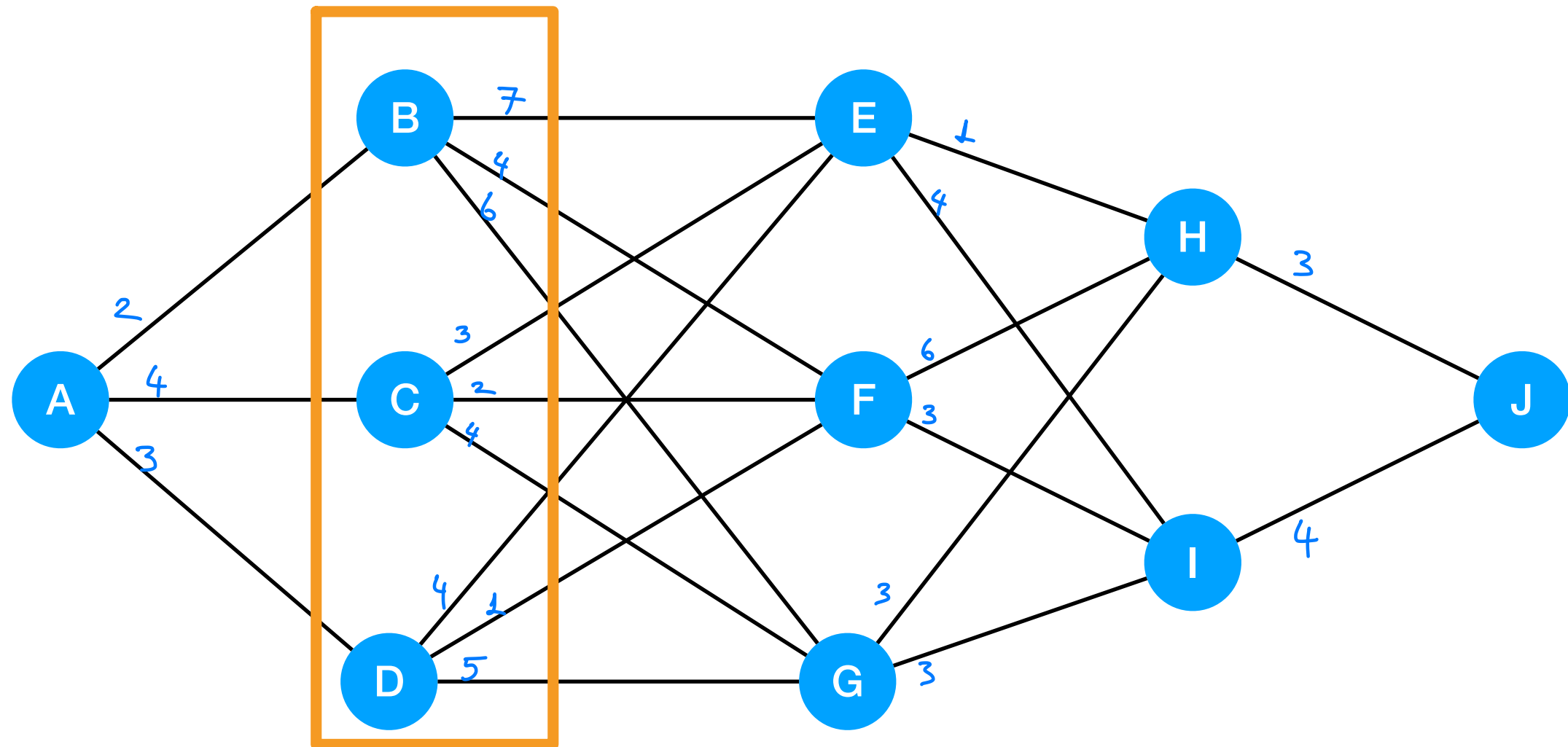
$$\begin{array}{lll} F(J) = 0 & F(H) = 3 & F(E) = 4 \\ & F(I) = 4 & F(F) = 7 \\ & & F(G) = 6 \end{array}$$



$F(.)$  : cost-to-go

# Principle of Optimality

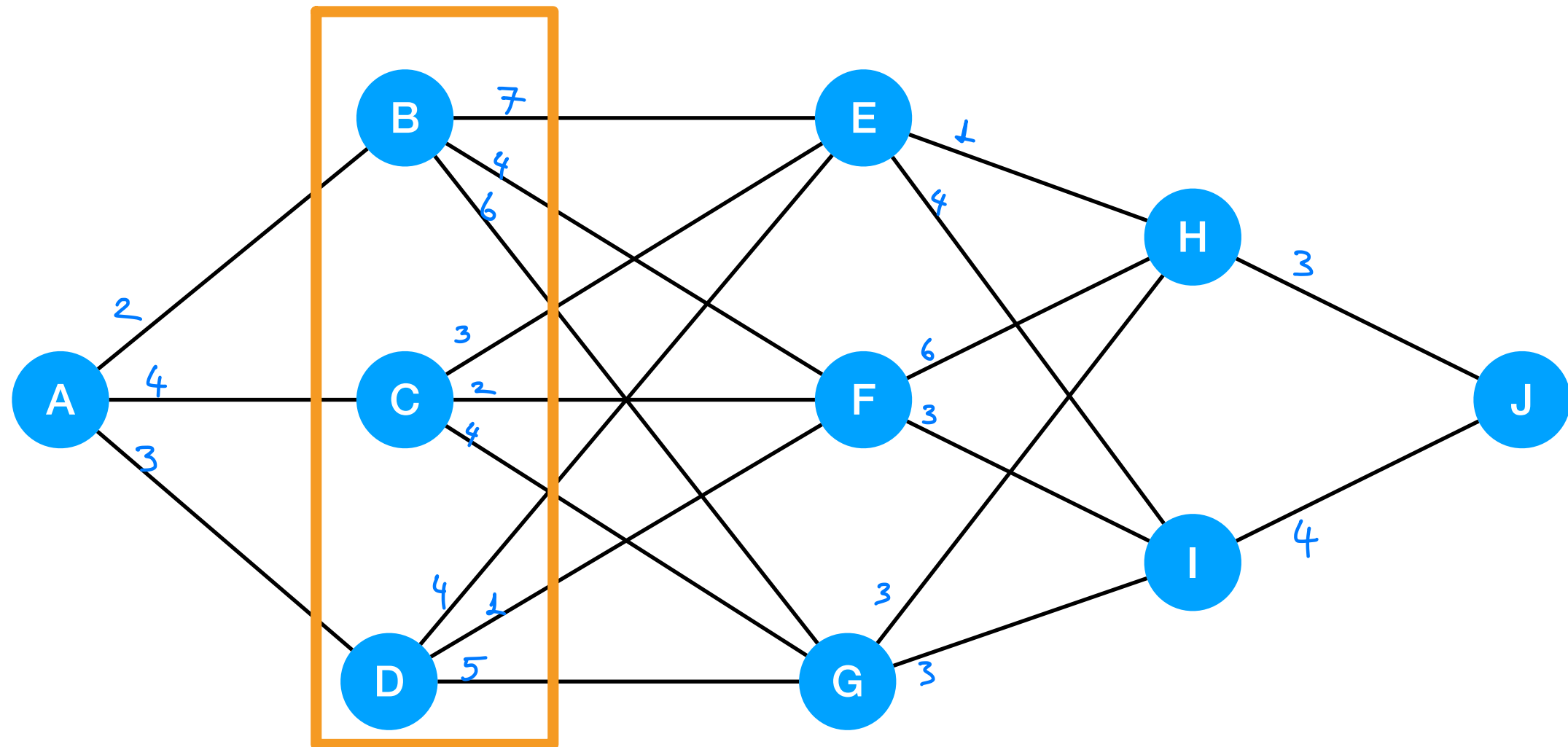
$$\begin{array}{llll}
 F(J) = 0 & F(H) = 3 & F(E) = 4 & F(B) = \min \{ 7 + F(E), 4 + F(F), 6 + F(G) \} \\
 & F(I) = 4 & F(F) = 7 & F(B) = 11 \\
 & & F(G) = 6 &
 \end{array}$$



$F(.)$  : cost-to-go

# Principle of Optimality

$$\begin{array}{llll} F(J) = 0 & F(H) = 3 & F(E) = 4 & F(B) = 11 \\ & F(I) = 4 & F(F) = 7 & F(C) = 7 \\ & & F(G) = 6 & F(D) = 8 \end{array}$$



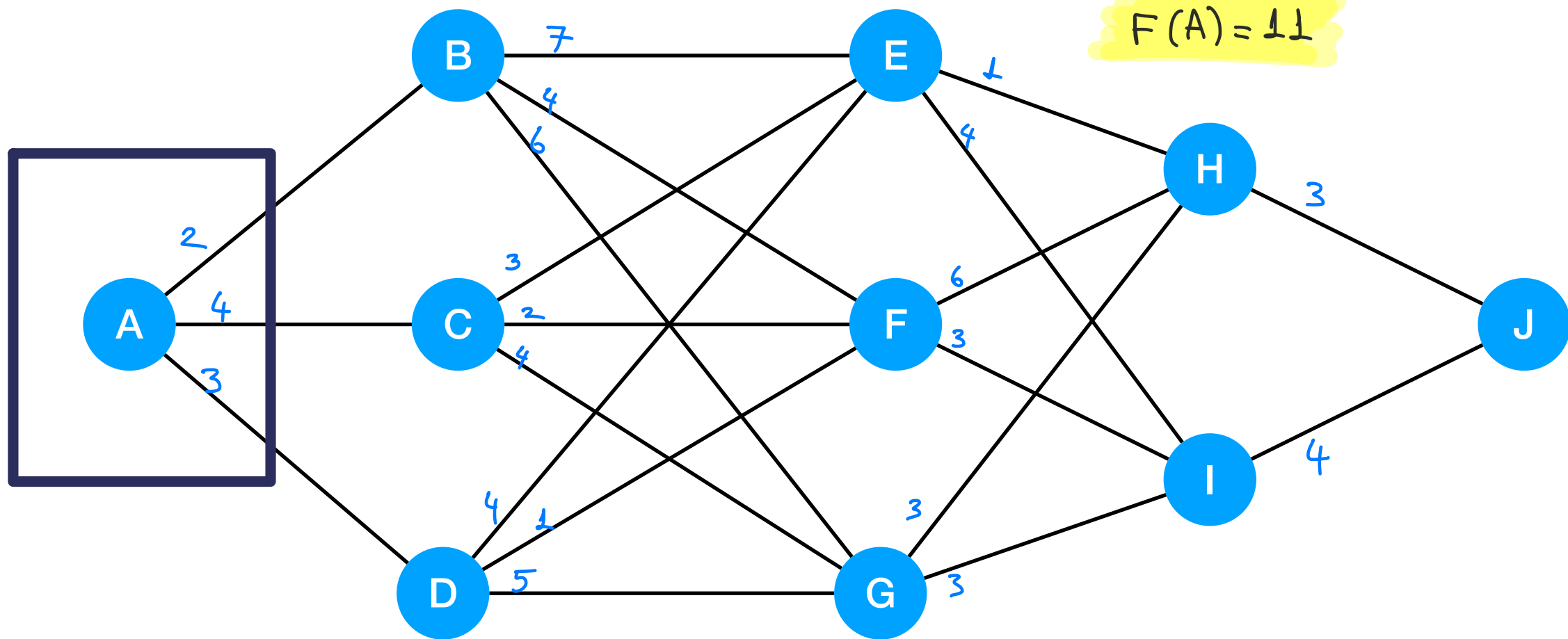
$F(.)$  : cost-to-go

# Principle of Optimality

$$\begin{array}{llll}
 F(J) = 0 & F(H) = 3 & F(E) = 4 & F(B) = 11 \\
 & F(I) = 4 & F(F) = 7 & F(C) = 7 \\
 & & F(G) = 6 & F(D) = 8
 \end{array}$$

$$F(A) = \min \{ 2 + F(B), 4 + F(C), 3 + F(D) \}$$

$$F(A) = 11$$

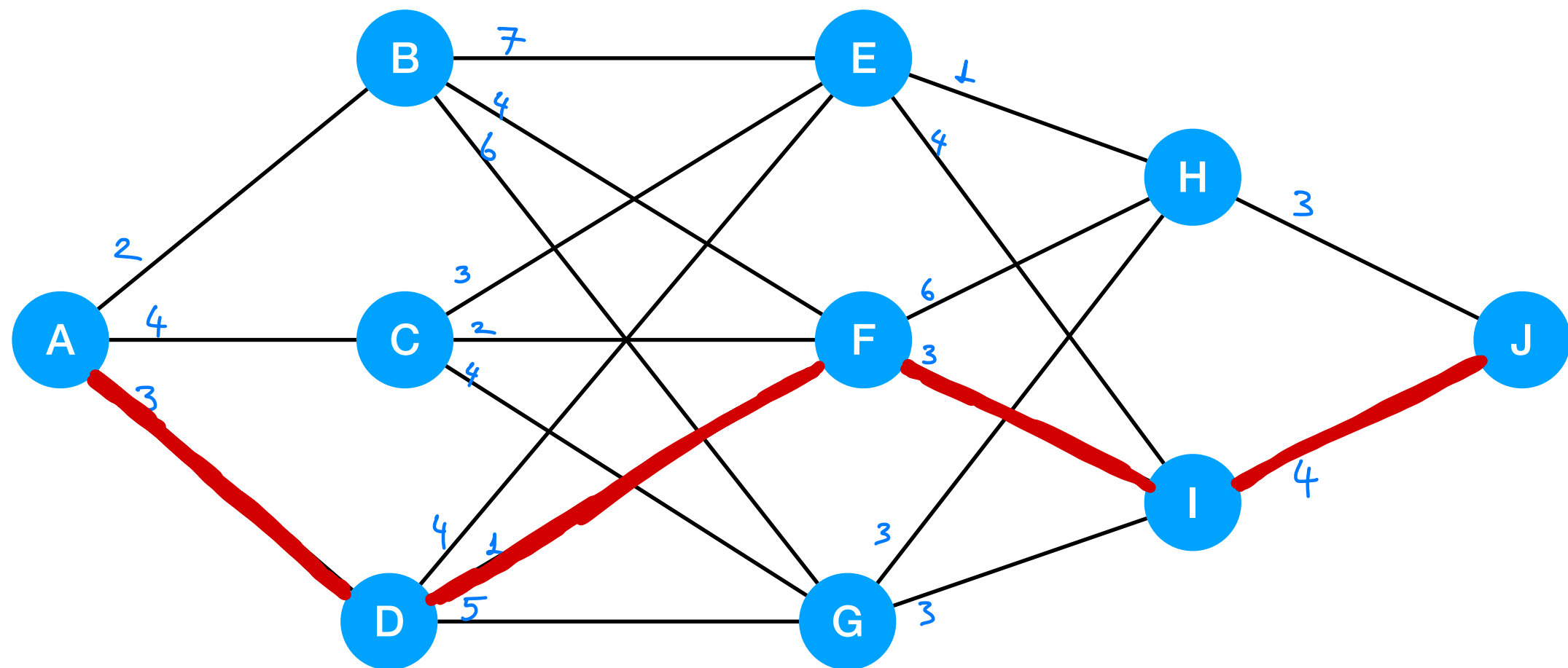


$F(.)$  : cost-to-go



# Principle of Optimality

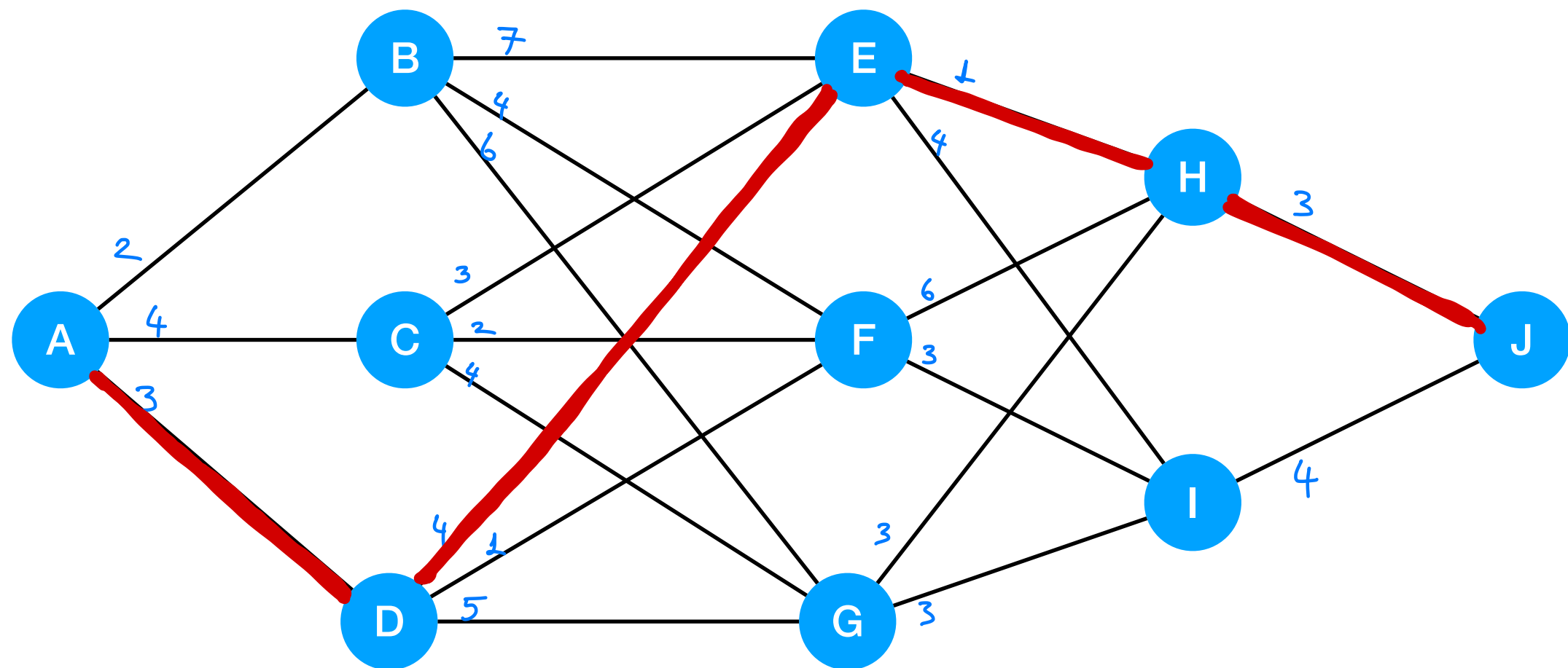
$$\begin{array}{lllll} F(J) = 0 & F(H) = 3 & F(E) = 4 & F(B) = 11 & F(A) = 11 \\ & F(I) = 4 & F(F) = 7 & F(C) = 7 & \\ & & F(G) = 6 & F(D) = 8 & \end{array}$$



$F(.)$  : cost-to-go

# Principle of Optimality

$$3 + 4 + 1 + 3 = 11$$



# Discrete-time optimization problem

- Consider the following process (plant):  $x_{k+1} = f_k(x_k, u_k)$
- Criterion/Cost to be minimized:  $J_i(x_i) = \min_{u_i, \dots, u_{N-1}} \left[ \sum_{k=i}^{N-1} g_k(x_k, u_k) + g_N(x_N) \right]$
- Note that when the final state is free, there can be an additional cost related to that state.

# Discrete-time optimization problem

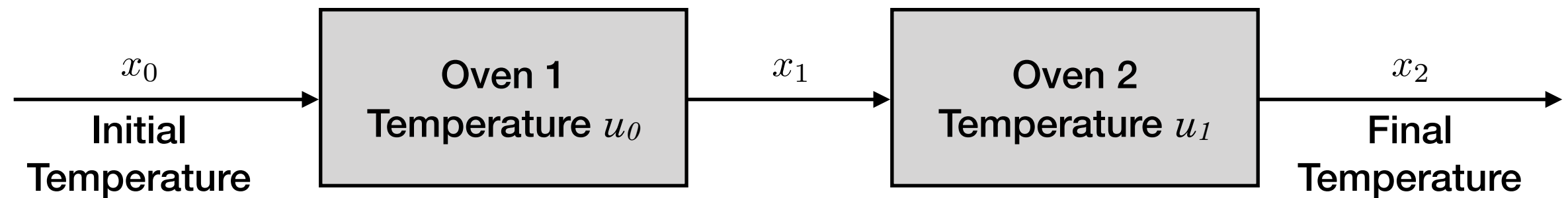
- Consider the following process (plant):  $x_{k+1} = f_k(x_k, u_k)$
- Criterion/Cost to be minimized:  $J_i(x_i) = \min_{u_i, \dots, u_{N-1}} \left[ \sum_{k=i}^{N-1} g_k(x_k, u_k) + g_N(x_N) \right]$
- Note that when the final state is free, there can be an additional cost related to that state.
- Let's use the principle of optimality:

$$J_k(x_k) = \min_{u_k} [g_k(x_k, u_k) + J_{k+1}^*(x_{k+1})]$$

We want to find  $u_k$  such that the expression is minimized and we get the optimal cost at time  $k$ .

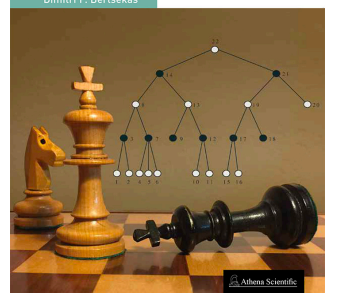
# Example

- A certain material is passed through a sequence of 2 ovens:



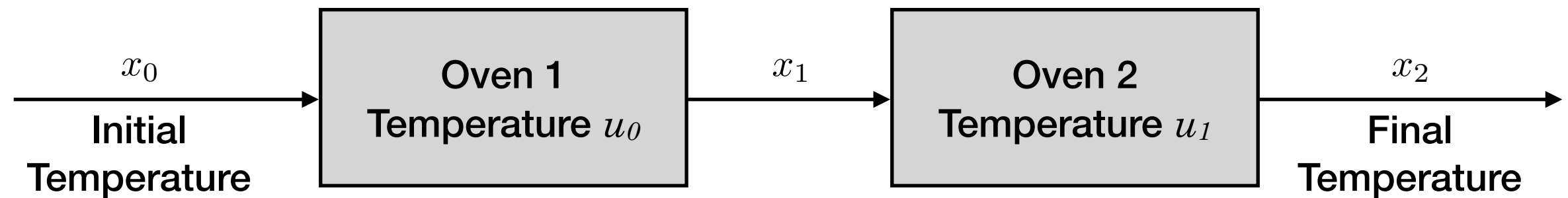
VOLUME 1 - 4TH EDITION  
**Dynamic Programming  
and Optimal Control**

Dimitri P. Bertsekas



# Example

- A certain material is passed through a sequence of 2 ovens:



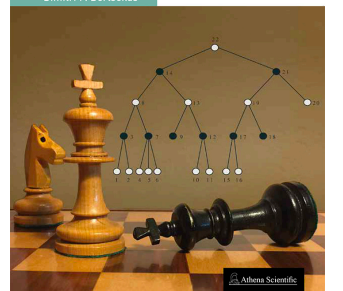
The temperature of the material evolves according to

$$x_{k+1} = (1 - a)x_k + au_k$$

where  $a$  is a known scalar from the interval  $(0,1)$ .

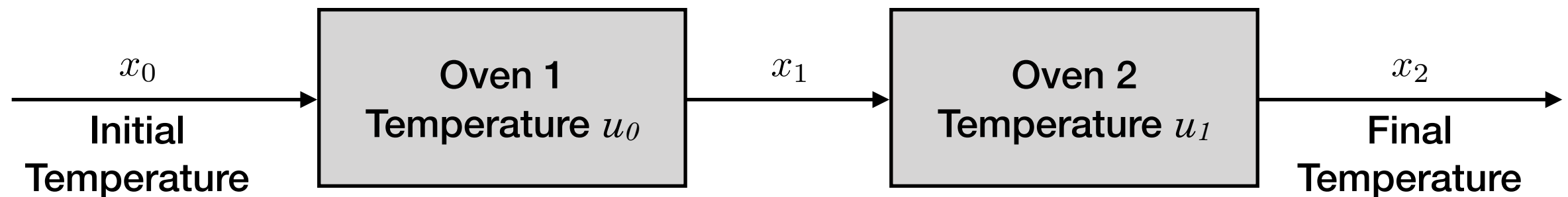
VOLUME 1 • 4TH EDITION  
**Dynamic Programming  
and Optimal Control**

Dimitri P. Bertsekas



# Example

- A certain material is passed through a sequence of 2 ovens:



The temperature of the material evolves according to

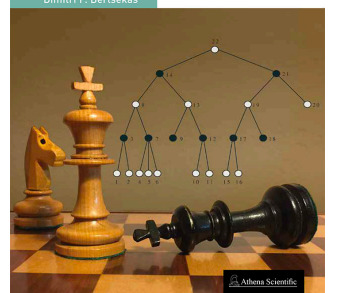
$$x_{k+1} = (1 - a)x_k + au_k$$

where  $a$  is a known scalar from the interval  $(0,1)$ .

The objective is to get the final temperature  $x_2$  close to a given target  $T$ , while expending relatively little energy.

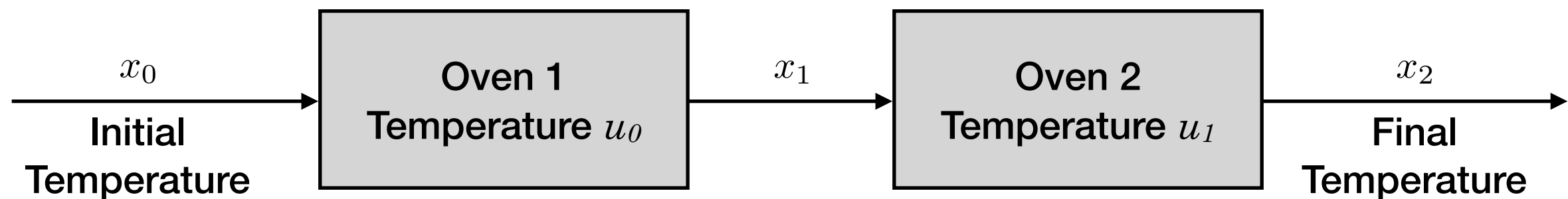
VOLUME 1 • 4TH EDITION  
**Dynamic Programming  
and Optimal Control**

Dimitri P. Bertsekas



# Example

- A certain material is passed through a sequence of 2 ovens:



The temperature of the material evolves according to

$$x_{k+1} = (1 - a)x_k + au_k$$

where  $a$  is a known scalar from the interval  $(0,1)$ .

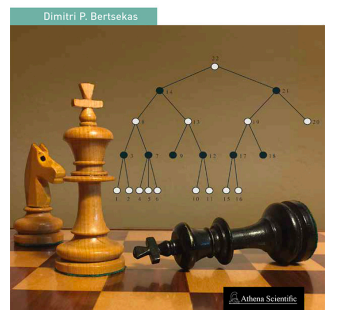
The objective is to get the final temperature  $x_2$  close to a given target  $T$ , while expending relatively little energy.

This is expressed by a cost function of the form

$$J = r(x_2 - T)^2 + u_0^2 + u_1^2$$

where  $r$  is a given scalar.

VOLUME 1 • 4TH EDITION  
**Dynamic Programming  
and Optimal Control**





## Solution:

We have  $N = 2$  and a terminal cost:  $J_2(x_2) = r(x_2 - T)^2$

## Solution:

We have  $N = 2$  and a terminal cost:  $J_2(x_2) = r(x_2 - T)^2$

For the next-to-last stage, we have:

$$\begin{aligned} J_1(x_1) &= \min_{u_1} [u_1^2 + J_2(x_2)] \\ &= \min_{u_1} [u_1^2 + J_2((1-a)x_1 + au_1)] \\ &= \min_{u_1} [u_1^2 + r((1-a)x_1 + au_1 - T)^2] \end{aligned}$$

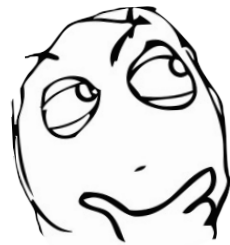
## Solution:

We have  $N = 2$  and a terminal cost:  $J_2(x_2) = r(x_2 - T)^2$

For the next-to-last stage, we have:

$$\begin{aligned} J_1(x_1) &= \min_{u_1} [u_1^2 + J_2(x_2)] \\ &= \min_{u_1} [u_1^2 + J_2((1-a)x_1 + au_1)] \\ &= \min_{u_1} [u_1^2 + r((1-a)x_1 + au_1 - T)^2] \end{aligned}$$

How to minimize this?



## Solution:

We have  $N = 2$  and a terminal cost:  $J_2(x_2) = r(x_2 - T)^2$

For the next-to-last stage, we have:

$$\begin{aligned} J_1(x_1) &= \min_{u_1} [u_1^2 + J_2(x_2)] \\ &= \min_{u_1} [u_1^2 + J_2((1-a)x_1 + au_1)] \\ &= \min_{u_1} [u_1^2 + r((1-a)x_1 + au_1 - T)^2] \end{aligned}$$

The minimization is done by setting to zero the derivative with respect to  $u_1$ :

$$\left. \frac{\partial J_1(x_1)}{\partial u_1} \right|_{u_1=u_1^*} = 2u_1^* + 2ar((1-a)x_1 + au_1^* - T) = 0$$

$$\Rightarrow u_1^* = \frac{ra(T - (1-a)x_1)}{1 + ra^2}$$

## Solution:

We have  $N = 2$  and a terminal cost:  $J_2(x_2) = r(x_2 - T)^2$

For the next-to-last stage, we have:

$$\begin{aligned} J_1(x_1) &= \min_{u_1} [u_1^2 + J_2(x_2)] \\ &= \min_{u_1} [u_1^2 + J_2((1-a)x_1 + au_1)] \\ &= \min_{u_1} [u_1^2 + r((1-a)x_1 + au_1 - T)^2] \end{aligned}$$

The minimization is done by setting to zero the derivative with respect to  $u_1$ :

$$\begin{aligned} \left. \frac{\partial J_1(x_1)}{\partial u_1} \right|_{u_1=u_1^*} &= 2u_1^* + 2ar((1-a)x_1 + au_1^* - T) = 0 \\ \Rightarrow u_1^* &= \frac{ra(T - (1-a)x_1)}{1 + ra^2} \end{aligned}$$

Not a single control, but a *control function* for every possible state  $x_1$

By substituting  $u_1^*$  in  $J_1(x_1)$ , we obtain

$$J_1^*(x_1) = \frac{r((1-a)x_1 - T)^2}{1 + ra^2}$$

By substituting  $u_1^*$  in  $J_1(x_1)$ , we obtain

$$J_1^*(x_1) = \frac{r((1-a)x_1 - T)^2}{1 + ra^2}$$

We now go back one stage:

$$\begin{aligned} J_0(x_0) &= \min_{u_0} [u_0^2 + J_1^*(x_1)] \\ &= \min_{u_0} [u_0^2 + J_1^*((1-a)x_0 + au_0)] \\ &= \min_{u_0} \left[ u_0^2 + \frac{r((1-a)^2x_0 + (1-a)au_0 - T)^2}{1 + ra^2} \right] \end{aligned}$$

By substituting  $u_1^*$  in  $J_1(x_1)$ , we obtain

$$J_1^*(x_1) = \frac{r((1-a)x_1 - T)^2}{1 + ra^2}$$

We now go back one stage:

$$\begin{aligned} J_0(x_0) &= \min_{u_0} [u_0^2 + J_1^*(x_1)] \\ &= \min_{u_0} [u_0^2 + J_1^*((1-a)x_0 + au_0)] \\ &= \min_{u_0} \left[ u_0^2 + \frac{r((1-a)^2x_0 + (1-a)au_0 - T)^2}{1 + ra^2} \right] \end{aligned}$$

The minimization is done by setting to zero the derivative with respect to  $u_0$ :

$$\begin{aligned} \left. \frac{\partial J_0(x_0)}{\partial u_0} \right|_{u_0=u_0^*} &= 2u_0^* + \frac{2a(1-a)r((1-a)^2x_0 + (1-a)au_0^* - T)}{1 + ra^2} = 0 \\ \Rightarrow u_0^* &= \frac{a(1-a)r(T - (1-a)^2x_0)}{1 + ra^2(1 + (1-a)^2)} \end{aligned}$$



By substituting  $u_0^*$  in  $J_0(x_0)$ , we obtain

$$J_0^*(x_0) = \frac{r((1-a)^2 x_0 - T)^2}{1 + ra^2(1 + (1-a)^2)}$$

This is the optimal cost and this completes the solution of the problem!

By substituting  $u_0^*$  in  $J_0(x_0)$ , we obtain

$$J_0^*(x_0) = \frac{r((1-a)^2 x_0 - T)^2}{1 + ra^2(1 + (1-a)^2)}$$

This is the optimal cost and this completes the solution of the problem!

Given  $x_0 \longrightarrow$  Find  $u_0^*$  and  $u_1^*$

$$\Rightarrow u_0^* = \frac{a(1-a)r(T - (1-a)^2 x_0)}{1 + ra^2(1 + (1-a)^2)}$$

$$x_{k+1} = (1-a)\overline{x_k} + a\overline{u_k}$$

$x_1 \qquad \qquad \qquad x_0 \qquad \qquad \qquad u_0^*$

$$\Rightarrow u_1^* = \frac{ra(T - (1-a)x_1)}{1 + ra^2}$$

By substituting  $u_0^*$  in  $J_0(x_0)$ , we obtain

$$J_0^*(x_0) = \frac{r((1-a)^2 x_0 - T)^2}{1 + ra^2(1 + (1-a)^2)}$$

This is the optimal cost and this completes the solution of the problem!

## Remarks:

- In the example, we easily obtained an analytical solution - the **quadratic nature of the cost** and the **linearity of the system equation** simplifies the solution

By substituting  $u_0^*$  in  $J_0(x_0)$ , we obtain

$$J_0^*(x_0) = \frac{r((1-a)^2 x_0 - T)^2}{1 + ra^2(1 + (1-a)^2)}$$

This is the optimal cost and this completes the solution of the problem!

## Remarks:

- In the example, we easily obtained an analytical solution - the **quadratic nature of the cost** and the **linearity of the system equation** simplifies the solution
- We will see next that, generally, when the system is linear and the cost is quadratic, the optimal policy and the cost-to-go function are given by closed form expressions, regardless of the number of stages  $N$ .

# Solution of the discrete-time LQ problem using Dynamic Programming

**Process:**  $x_{k+1} = \Phi x_k + \Gamma u_k$ ,  $x_i$  is given

**Criterion:** 
$$J = \frac{1}{2} x_N^T S_N x_N + \frac{1}{2} \sum_{k=i}^{N-1} \{ x_k^T Q x_k + u_k^T R u_k \}$$

( $S_N \geq 0$ ,  $Q \geq 0$ , and  $R > 0$  are symmetric)

# Solution of the discrete-time LQ problem using Dynamic Programming

**Process:**  $x_{k+1} = \Phi x_k + \Gamma u_k$ ,  $x_i$  is given

**Criterion:** 
$$J = \frac{1}{2} x_N^T S_N x_N + \frac{1}{2} \sum_{k=i}^{N-1} \{ x_k^T Q x_k + u_k^T R u_k \}$$

( $S_N \geq 0$ ,  $Q \geq 0$ , and  $R > 0$  are symmetric)

- $x_N$  is free (and that is why there is a terminal cost associated with it)

- Find  $u_k^*$  in the interval  $[i, N]$  minimizing the criterion

# Some matrix theory

- **Definition:** Let  $A$  be a **symmetric matrix**.

It is **positive definite**, if the scalar  $\mathbf{x}^T A \mathbf{x} > 0$  for all non-zero vectors  $\mathbf{x}$  ( $A > 0$ )

It is **negative definite**, if the scalar  $\mathbf{x}^T A \mathbf{x} < 0$  for all non-zero vectors  $\mathbf{x}$  ( $A < 0$ )

It is **positive semidefinite**, if the scalar  $\mathbf{x}^T A \mathbf{x} \geq 0$  for all non-zero vectors  $\mathbf{x}$  ( $A \geq 0$ )

It is **negative semidefinite**, if the scalar  $\mathbf{x}^T A \mathbf{x} \leq 0$  for all non-zero vectors  $\mathbf{x}$  ( $A \leq 0$ )

- If a matrix  $A$  is not symmetric in  $\mathbf{x}^T A \mathbf{x}$ , we can it symmetrize by (prove)

$$\mathbf{x}^T A \mathbf{x} = \mathbf{x}^T \left( \frac{A + A^T}{2} \right) \mathbf{x}$$

Hence, it can always be assumed that the weight matrices ( $Q$  and  $R$ ) are symmetric

- **From linear algebra:**

The eigenvalues of symmetric matrices are real. A symmetric matrix is positive definite, if and only if all eigenvalues are positive. The matrix is positive semidefinite, if and only if the eigenvalues are nonnegative.

# Some matrix theory

- **Definition:** Let  $A$  be a symmetric matrix.

It is **positive definite**, if the scalar  $x^T A x > 0$  for all non-zero vectors  $x$  ( $A > 0$ )

It is **negative definite**, if the scalar  $x^T A x < 0$  for all non-zero vectors  $x$  ( $A < 0$ )

It is **positive semidefinite**, if the scalar  $x^T A x \geq 0$  for all non-zero vectors  $x$  ( $A \geq 0$ )

It is **negative semidefinite**, if the scalar  $x^T A x \leq 0$  for all non-zero vectors  $x$  ( $A \leq 0$ )

Challenge Accepted!

Let's say  $F$  is not symmetric.

$$F = \frac{F + F^T}{2} + \frac{F - F^T}{2}$$

Do this trick!

→ This becomes zero!

- If a matrix  $A$  is not symmetric, it can always be assumed that the weight matrix  $A$  is symmetric by adding and subtracting the term  $x^T A x$ .

$$x^T F x = x^T \frac{F + F^T}{2} x + x^T \frac{F - F^T}{2} x$$

$$= x^T \frac{F + F^T}{2} x + \frac{x^T F x}{2} - \frac{x^T F^T x}{2}$$

Since it is scalar, I can take transpose!

Hence, it can always be assumed that the weight matrix  $A$  is symmetric and  $R$  are symmetric.

$$= x^T \frac{F + F^T}{2} x + \cancel{\frac{x^T F x}{2}} - \cancel{\frac{x^T F x}{2}}$$

- **From linear algebra:**

The eigenvalues of symmetric matrices are real. A symmetric matrix is positive definite, if and only if all eigenvalues are positive. The matrix is positive semidefinite, if and only if the eigenvalues are nonnegative.



# Some matrix theory

- Consider the vectors  $\mathbf{c}$  and  $\mathbf{x}$ , the scalar  $f(\mathbf{x}) = \mathbf{x}^T A \mathbf{x}$ , and the vector  $A\mathbf{x}$ . The following hold:

$$\frac{\partial(\mathbf{x})}{\partial \mathbf{x}} = \text{vec}(I_n)$$

$$\frac{\partial(\mathbf{x})}{\partial \mathbf{x}^T} = \frac{\partial(\mathbf{x}^T)}{\partial \mathbf{x}} = I_n$$

$$\frac{\partial(\mathbf{c}^T \mathbf{x})}{\partial \mathbf{x}} = \frac{\partial(\mathbf{x}^T \mathbf{c})}{\partial \mathbf{x}} = \mathbf{c}$$

$$\frac{\partial(\mathbf{x}^T A)}{\partial \mathbf{x}} = A$$

$$\frac{\partial(A\mathbf{x})}{\partial \mathbf{x}} = \text{vec}(A)$$

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = (A + A^T)\mathbf{x}$$

# Solution of the discrete-time LQ problem using Dynamic Programming

- Cost of the final state:
- Backwards in time to **time-instant**  $N - 1$ :
- Minimize with respect to  $u_{N-1}$ :

# Solution of the discrete-time LQ problem using Dynamic Programming

- Cost of the final state:

$$J_N^* = \frac{1}{2} x_N^T S_N x_N$$

- Backwards in time to **time-instant**  $N - 1$ :

- Minimize with respect to  $u_{N-1}$ :

# Solution of the discrete-time LQ problem using Dynamic Programming

- Cost of the final state:

$$J_N^* = \frac{1}{2} x_N^T S_N x_N$$

- Backwards in time to **time-instant**  $N - 1$ :

$$\begin{aligned} J_{N-1} &= \frac{1}{2} x_N^T S_N x_N + \frac{1}{2} x_{N-1}^T Q x_{N-1} + \frac{1}{2} u_{N-1}^T R u_{N-1} \\ &= \frac{1}{2} (\Phi x_{N-1} + \Gamma u_{N-1})^T S_N (\Phi x_{N-1} + \Gamma u_{N-1}) + \frac{1}{2} x_{N-1}^T Q x_{N-1} + \frac{1}{2} u_{N-1}^T R u_{N-1} \end{aligned}$$

- Minimize with respect to  $u_{N-1}$ :

# Solution of the discrete-time LQ problem using Dynamic Programming

- Cost of the final state:

$$J_N^* = \frac{1}{2} x_N^T S_N x_N$$

- Backwards in time to **time-instant**  $N - 1$ :

$$\begin{aligned} J_{N-1} &= \frac{1}{2} x_N^T S_N x_N + \frac{1}{2} x_{N-1}^T Q x_{N-1} + \frac{1}{2} u_{N-1}^T R u_{N-1} \\ &= \frac{1}{2} (\Phi x_{N-1} + \Gamma u_{N-1})^T S_N (\Phi x_{N-1} + \Gamma u_{N-1}) + \frac{1}{2} x_{N-1}^T Q x_{N-1} + \frac{1}{2} u_{N-1}^T R u_{N-1} \end{aligned}$$

- Minimize with respect to  $u_{N-1}$ :

$$\frac{\partial J_{N-1}}{\partial u_{N-1}} = \Gamma^T S_N (\Phi x_{N-1} + \Gamma u_{N-1}) + R u_{N-1} = 0$$

$$\Rightarrow u_{N-1}^* = - \underbrace{(\Gamma^T S_N \Gamma + R)^{-1} \Gamma^T S_N \Phi}_{L_{N-1}} x_{N-1} = -L_{N-1} x_{N-1}$$

# Solution of the discrete-time LQ problem using Dynamic Programming

- Cost of the final state:

$$J_N^* = \frac{1}{2} x_N^T S_N x_N$$

- Backwards in time to **time-instant**  $N - 1$ :

$$\begin{aligned} J_{N-1} &= \frac{1}{2} x_N^T S_N x_N + \frac{1}{2} x_{N-1}^T Q x_{N-1} + \frac{1}{2} u_{N-1}^T R u_{N-1} \\ &= \frac{1}{2} (\Phi x_{N-1} + \Gamma u_{N-1})^T S_N (\Phi x_{N-1} + \Gamma u_{N-1}) + \frac{1}{2} x_{N-1}^T Q x_{N-1} + \frac{1}{2} u_{N-1}^T R u_{N-1} \end{aligned}$$

- Minimize with respect to  $u_{N-1}$ :

$$\frac{\partial J_{N-1}}{\partial u_{N-1}} = \Gamma^T S_N (\Phi x_{N-1} + \Gamma u_{N-1}) + R u_{N-1} = 0$$



$$u = -L \cdot x$$

$$\Rightarrow u_{N-1}^* = - \underbrace{(\Gamma^T S_N \Gamma + R)^{-1} \Gamma^T S_N \Phi}_{L_{N-1}} x_{N-1} = -L_{N-1} x_{N-1}$$

# Solution of the discrete-time LQ problem using Dynamic Programming

- By substituting  $L_{N-1}$  into  $J_{N-1}$ :
- Backwards in time to **time-instant**  $N - 2$ :

---


**Recall:**

$$J_{N-1} = \frac{1}{2}x_N^T S_N x_N + \frac{1}{2}x_{N-1}^T Q x_{N-1} + \frac{1}{2}u_{N-1}^T R u_{N-1}$$

# Solution of the discrete-time LQ problem using Dynamic Programming

- By substituting  $L_{N-1}$  into  $J_{N-1}$ :

$$J_{N-1}^* = \frac{1}{2} x_{N-1}^T \underbrace{\left[ (\Phi - \Gamma L_{N-1})^T S_N (\Phi - \Gamma L_{N-1}) + Q + L_{N-1}^T R L_{N-1} \right]}_{S_{N-1}} x_{N-1}$$



$$= \frac{1}{2} x_{N-1}^T S_{N-1} x_{N-1} \quad \text{Quadratic again!}$$

- Backwards in time to **time-instant**  $N - 2$ :

---

**Recall:**

$$J_{N-1} = \frac{1}{2} x_N^T S_N x_N + \frac{1}{2} x_{N-1}^T Q x_{N-1} + \frac{1}{2} u_{N-1}^T R u_{N-1}$$



# Solution of the discrete-time LQ problem using Dynamic Programming

- By substituting  $L_{N-1}$  into  $J_{N-1}$ :

$$\begin{aligned} J_{N-1}^* &= \frac{1}{2} x_{N-1}^T \left[ \underbrace{(\Phi - \Gamma L_{N-1})^T S_N (\Phi - \Gamma L_{N-1}) + Q + L_{N-1}^T R L_{N-1}}_{S_{N-1}} \right] x_{N-1} \\ &= \frac{1}{2} x_{N-1}^T S_{N-1} x_{N-1} \end{aligned}$$

- Backwards in time to **time-instant**  $N - 2$ :

$$J_{N-2} = \frac{1}{2} x_{N-1}^T S_{N-1} x_{N-1} + \frac{1}{2} x_{N-2}^T Q x_{N-2} + \frac{1}{2} u_{N-2}^T R u_{N-2}$$

- Now, we want to determine  $u_{N-2}^*$  ... but the equations have the same form as before!

---

**Recall:**

$$J_{N-1} = \frac{1}{2} x_N^T S_N x_N + \frac{1}{2} x_{N-1}^T Q x_{N-1} + \frac{1}{2} u_{N-1}^T R u_{N-1}$$

# Solution of the discrete-time LQ problem using Dynamic Programming

- We obtain the general solution:

$$L_k = (\Gamma^T S_{k+1} \Gamma + R)^{-1} \Gamma^T S_{k+1} \Phi$$

$$u_k^* = -L_k x_k$$

$$S_k = (\Phi - \Gamma L_k)^T S_{k+1} (\Phi - \Gamma L_k) + Q + L_k^T R L_k \quad (\text{Riccati equation})$$

$$J_k^* = \frac{1}{2} x_k^T S_k x_k$$

# Solution of the discrete-time LQ problem using Dynamic Programming

- We obtain the general solution:

$$L_k = (\Gamma^T S_{k+1} \Gamma + R)^{-1} \Gamma^T S_{k+1} \Phi$$

$$u_k^* = -L_k x_k$$

$$S_k = (\Phi - \Gamma L_k)^T S_{k+1} (\Phi - \Gamma L_k) + Q + L_k^T R L_k \quad (\text{Riccati equation})$$

$$J_k^* = \frac{1}{2} x_k^T S_k x_k$$

- **Remarks:**

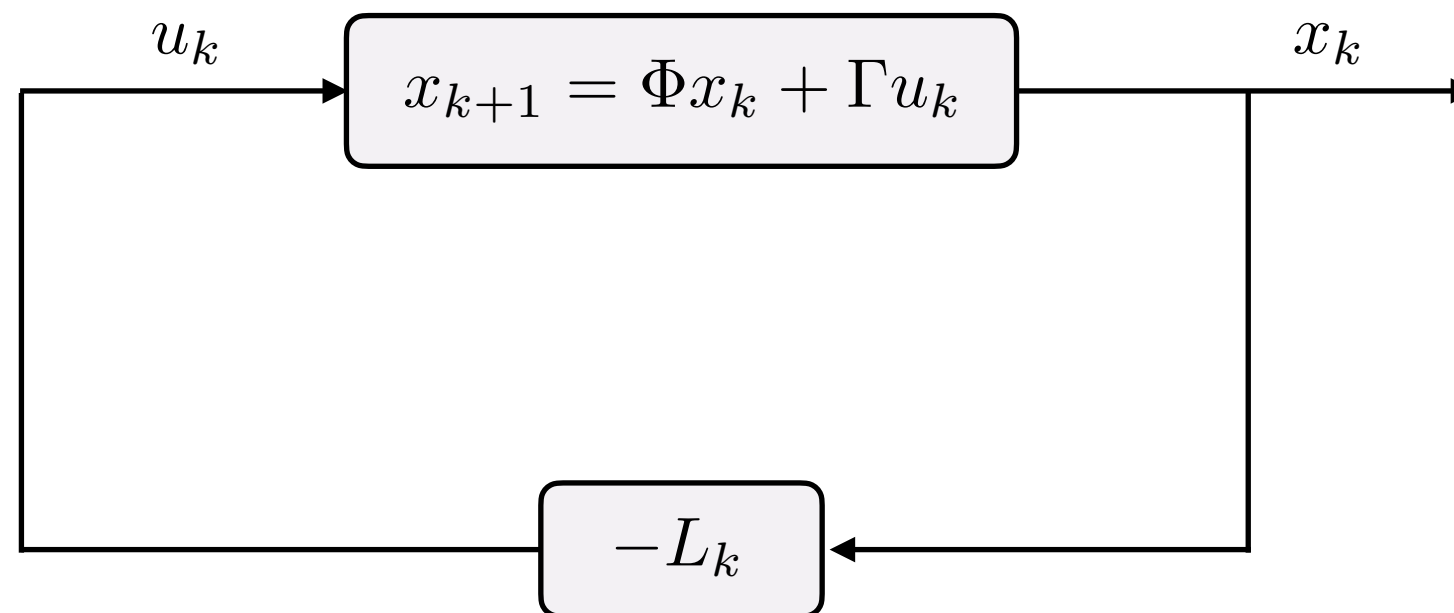
- The Riccati equation can be written in a way that is independent of  $L_k$

$$S_k = \Phi^T [S_{k+1} - S_{k+1} \Gamma (\Gamma^T S_{k+1} \Gamma + R)^{-1} \Gamma^T S_{k+1}] \Phi + Q$$

- Note that  $S_k$  and  $L_k$  are calculated “*backwards in time*”. They can be calculated in advance and saved to be used when control starts at time  $k_0$
- The procedure matches exactly the principle of optimality!

# Discussion on the solution

- The optimal cost is given by  $J_0^* = \frac{1}{2} x_0^T S_0 x_0$
- Control law - **simple and attractive in engineering applications**: state  $x_k$  is being fed back as input through the linear feedback gain matrix -  $L_k$ :



# The Riccati Equation

- The discrete algebraic Riccati equation (DARE) is

$$S_k = \Phi^T [S_{k+1} - S_{k+1} \Gamma (\Gamma^T S_{k+1} \Gamma + R)^{-1} \Gamma^T S_{k+1}] \Phi + Q$$

# The Riccati Equation

- The discrete algebraic Riccati equation (DARE) is

$$S_k = \Phi^T [S_{k+1} - S_{k+1} \Gamma (\Gamma^T S_{k+1} \Gamma + R)^{-1} \Gamma^T S_{k+1}] \Phi + Q$$

- Important role in control theory - its properties studied extensively.  
One property: if matrices  $\Phi_k$ ,  $\Gamma_k$ ,  $Q_k$  and  $R_k$  are constant and equal to  $\Phi$ ,  $\Gamma$ ,  $Q$  and  $R$ , respectively, then  $S_k$  converges as  $k$  goes to infinity to a steady state solution (**provided some conditions hold!**):

$$S = \Phi^T [S - S \Gamma (\Gamma^T S \Gamma + R)^{-1} \Gamma^T S] \Phi + Q$$

# The Riccati Equation

- The discrete algebraic Riccati equation (DARE) is

$$S_k = \Phi^T [S_{k+1} - S_{k+1} \Gamma (\Gamma^T S_{k+1} \Gamma + R)^{-1} \Gamma^T S_{k+1}] \Phi + Q$$

- Important role in control theory - its properties studied extensively.  
One property: if matrices  $\Phi_k$ ,  $\Gamma_k$ ,  $Q_k$  and  $R_k$  are constant and equal to  $\Phi$ ,  $\Gamma$ ,  $Q$  and  $R$ , respectively, then  $S_k$  converges as  $k$  goes to infinity to a steady state solution (**provided some conditions hold!**):

$$S = \Phi^T [S - S \Gamma (\Gamma^T S \Gamma + R)^{-1} \Gamma^T S] \Phi + Q$$

- Hence, for infinite horizon problems we can use the stationary solution of the Riccati equation, which also gives a constant  $L$

$$L = (\Gamma^T S \Gamma + R)^{-1} \Gamma^T S \Phi$$

# The Riccati Equation

- The discrete algebraic Riccati equation (DARE) is

$$S_k = \Phi^T [S_{k+1} - S_{k+1} \Gamma (\Gamma^T S_{k+1} \Gamma + R)^{-1} \Gamma^T S_{k+1}] \Phi + Q$$

- Important role in control theory - its properties studied extensively.  
One property: if matrices  $\Phi_k$ ,  $\Gamma_k$ ,  $Q_k$  and  $R_k$  are constant and equal to  $\Phi$ ,  $\Gamma$ ,  $Q$  and  $R$ , respectively, then  $S_k$  converges as  $k$  goes to infinity to a steady state solution (**provided some conditions hold!**):

$$S = \Phi^T [S - S \Gamma (\Gamma^T S \Gamma + R)^{-1} \Gamma^T S] \Phi + Q$$

- Hence, for infinite horizon problems we can use the stationary solution of the Riccati equation, which also gives a constant  $L$

$$L = (\Gamma^T S \Gamma + R)^{-1} \Gamma^T S \Phi$$

This control law is *stationary*, i.e., it does not change over time



# Example

- The linear discrete-time system

$$\mathbf{x}[k+1] = \begin{bmatrix} 0.9974 & 0.0539 \\ -0.1078 & 1.1591 \end{bmatrix} \mathbf{x}[k] + \begin{bmatrix} 0.0013 \\ 0.0539 \end{bmatrix} u[k]$$

is to be controlled to minimize the cost

$$J = \frac{1}{2} \sum_{k=0}^{N-1} \{ 0.25x_1^2[k] + 0.05x_2^2[k] + 0.05u^2[k] \}$$

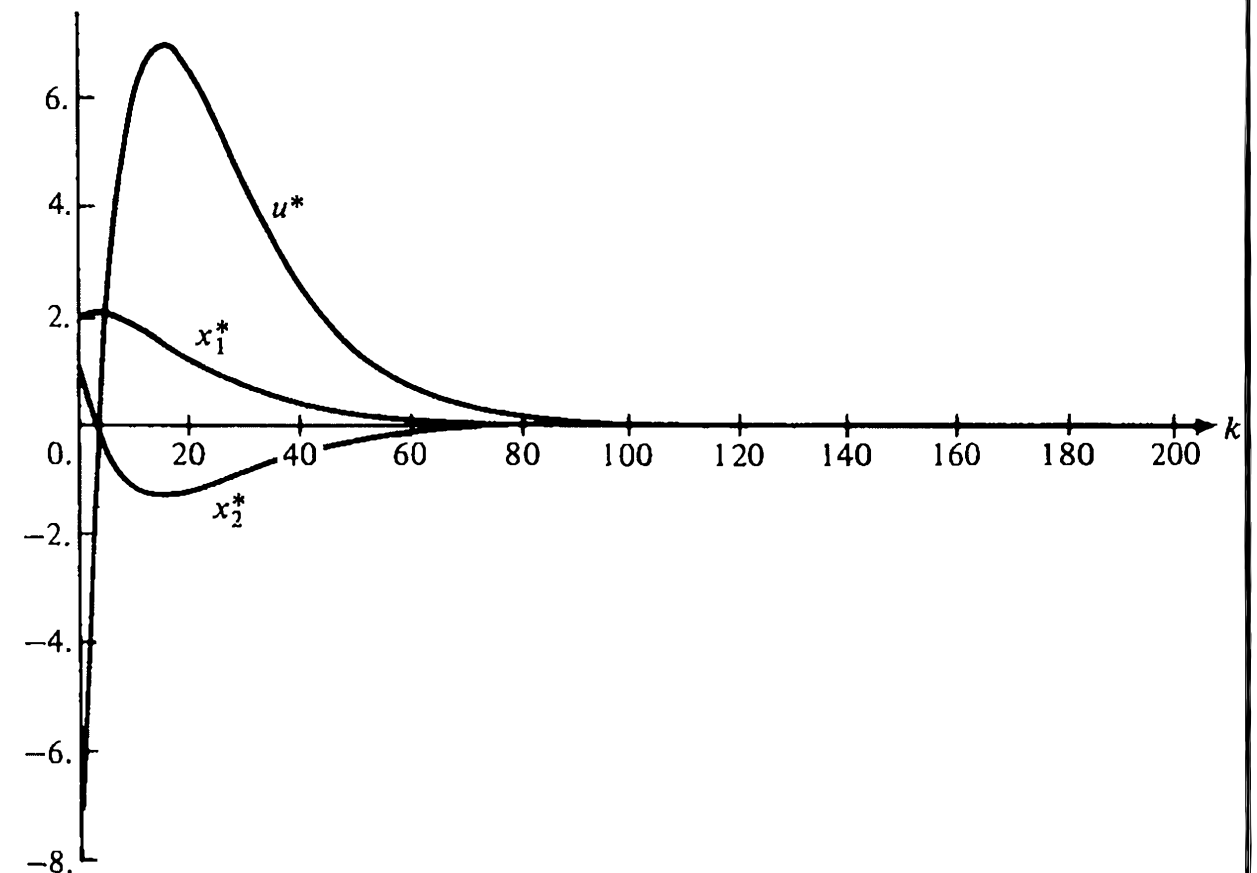
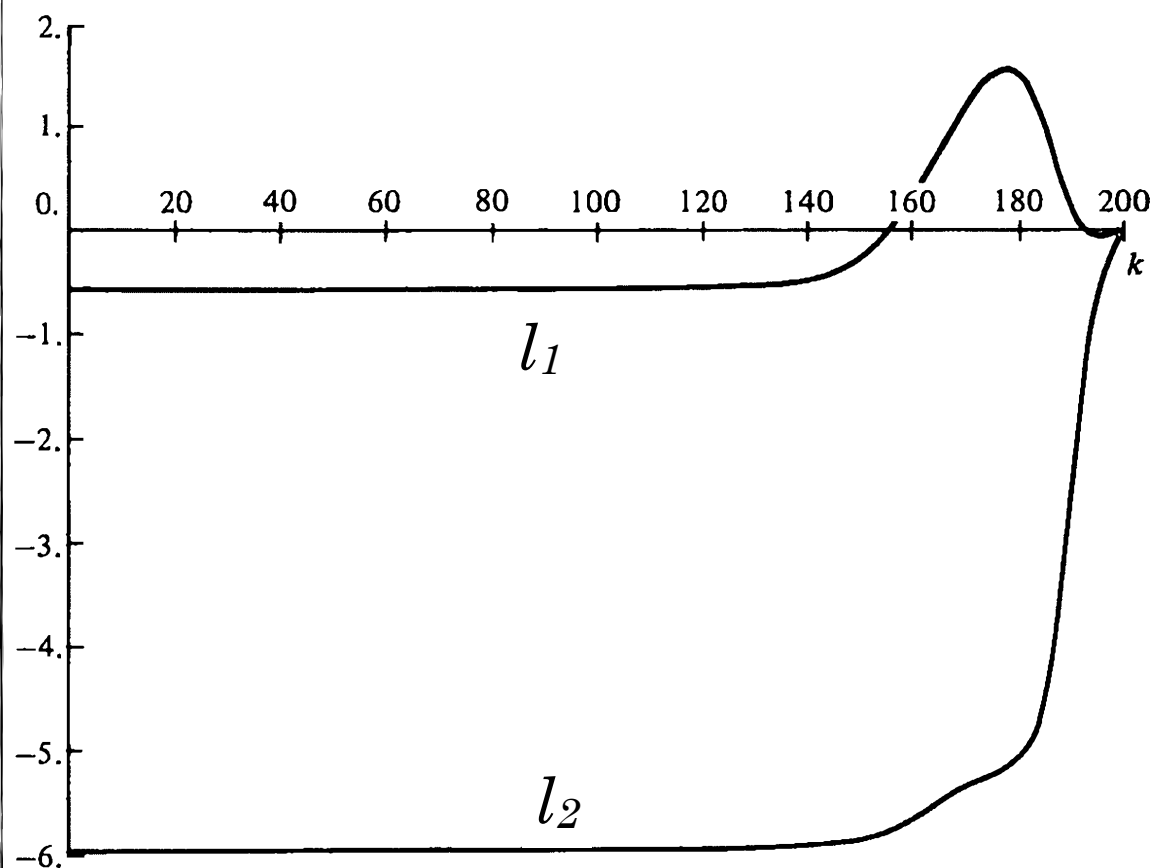
Determine the optimal control law.

## Solution:

- We can write it in the form we know:

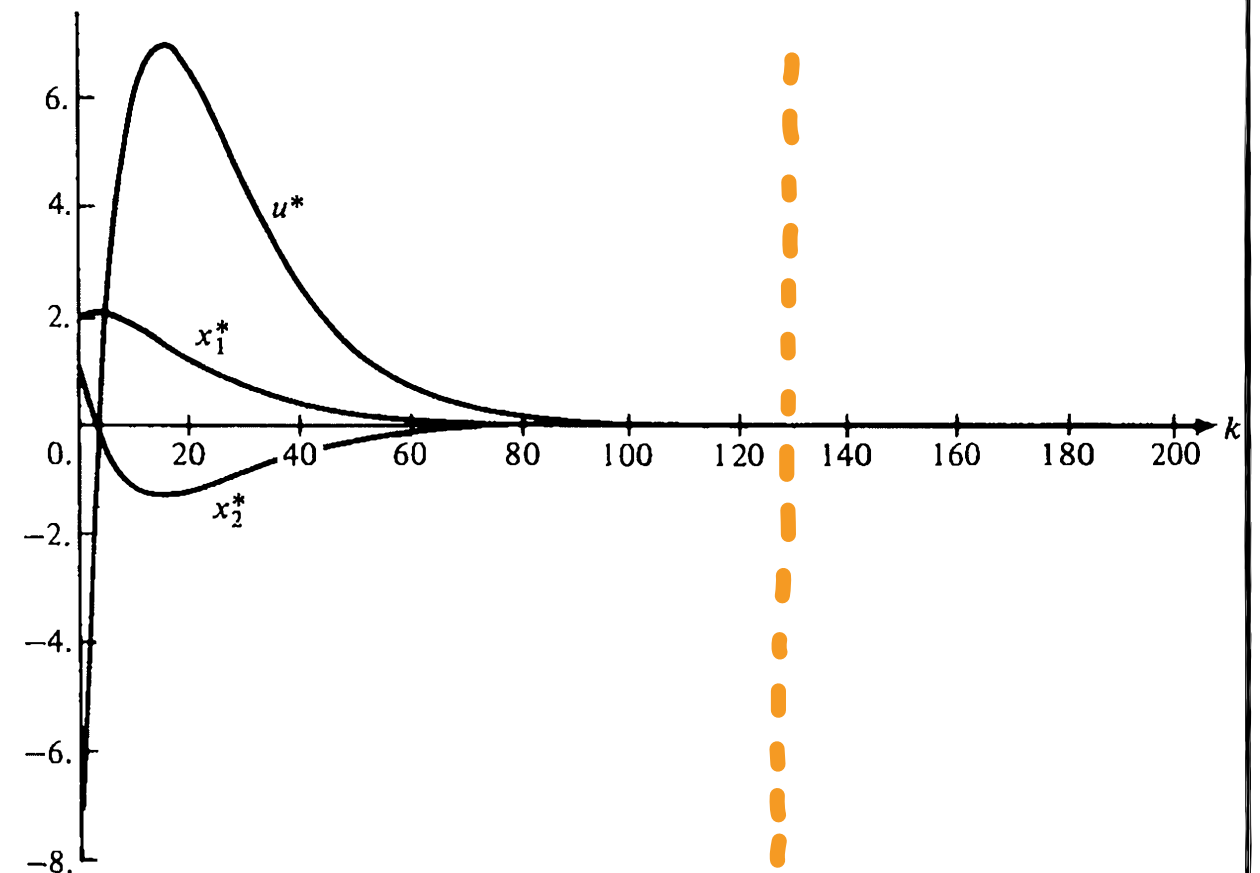
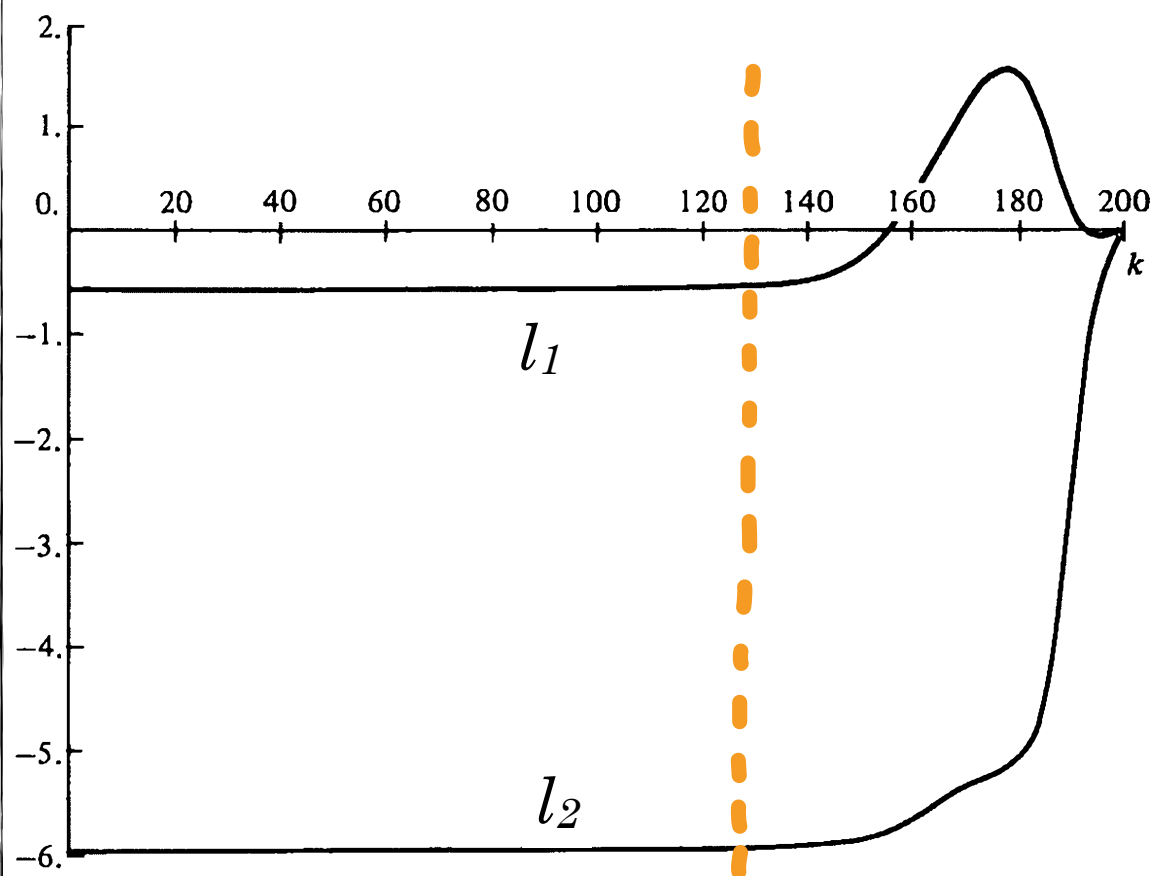
$$S_N = 0 \qquad Q = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.05 \end{bmatrix} \qquad R = 0.05$$

- The optimal feedback gain matrix  $L_k$  is shown below for  $N=200$ . Looking backwards, we observe that at  $k$  around 130  $L_k$  has reached its steady-state value



- The optimal trajectory has essentially reached 0 at  $k=100$ . Thus, we would expect that the performance would not be reduced significantly by simply using the steady-state value of  $L$ .

- The optimal feedback gain matrix  $L_k$  is shown below for  $N=200$ . Looking backwards, we observe that at  $k$  around 130  $L_k$  has reached its steady-state value



- The optimal trajectory has essentially reached 0 at  $k=100$ . Thus, we would expect that the performance would not be reduced significantly by simply using the steady-state value of  $L$ .

# Properties of the LQ-controller

- The pole-placement controller and the stationary LQ-controller have the same structure. However, they are obtained differently, so there are some differences in their properties.

# Properties of the LQ-controller

- The pole-placement controller and the stationary LQ-controller have the same structure. However, they are obtained differently, so there are some differences in their properties.
- The pole-placement procedure is well suited for single-input single-output systems. It is, however, difficult to compromise between the speed of the system and the magnitude of the control signal.

# Properties of the LQ-controller

- The pole-placement controller and the stationary LQ-controller have the same structure. However, they are obtained differently, so there are some differences in their properties.
- The pole-placement procedure is well suited for single-input single-output systems. It is, however, difficult to compromise between the speed of the system and the magnitude of the control signal.
- **The LQ-controller has several good properties:**
  - It is applicable to **multi-variable** and **time-varying** systems
  - changing the relative magnitude between the elements in the **weighting matrices** means a compromise between the speed of the recovery and the magnitudes of the control signals

# How to find the weighting matrices?

- The weighting matrices should ideally come from physical arguments - not usually the case
- LQ control theory has found considerable use even when this cannot be done.
- The feedback law is obtained directly by solving the *Riccati equation*.
- The closed-loop system obtained is then analyzed with respect to transient response, frequency response, robustness, and so on.
- The elements of the cost function are modified until the desired result is obtained.
- It has been found empirically that LQ-theory is quite easy to use in this way. The search will automatically guarantee stable closed-loop systems with reasonable margins.

# Extension to Nonlinear Systems

Nonlinear system:  $x_{t+1} = f(x_t, u_t)$

We can keep the system at the state  $x^*$  iff

$$\exists u^* \text{ s.t. } x^* = f(x^*, u^*)$$

Linearizing the dynamics around  $x^*$  gives:

$$x_{t+1} \approx f(x^*, u^*) + \underbrace{\frac{\partial f}{\partial x}(x^*, u^*)}_{\text{A}}(x_t - x^*) + \underbrace{\frac{\partial f}{\partial u}(x^*, u^*)}_{\text{B}}(u_t - u^*)$$

Equivalently:

$$x_{t+1} - x^* \approx A(x_t - x^*) + B(u_t - u^*)$$

Let  $z_t = x_t - x^*$ , let  $v_t = u_t - u^*$ , then:

$$z_{t+1} = Az_t + Bv_t, \quad \text{cost} = z_t^\top Q z_t + v_t^\top R v_t \quad [= \text{standard LQR}]$$

$$v_t = K z_t \Rightarrow u_t - u^* = K(x_t - x^*) \Rightarrow u_t = u^* + K(x_t - x^*)$$