AuE 835
Automotive Electronics Integration

PROJECT 2: CONTROL REVIEW

Controls

1. Introduction of Controls
2. System Modeling and Analysis
3. System Controller Designs

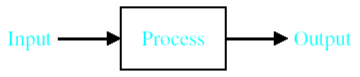
Controls

1. Introduction of Controls
2. System Modeling and Analysis
3. System Controller Designs

Introduction

System – An interconnection of elements and devices for a desired purpose.

Process – A device, plant, or system with input and output. The input and output relationship represents the cause-and-effect relationship of the process.

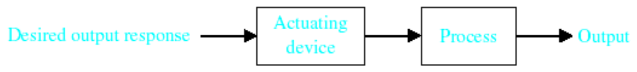


Process to be controlled.

Control System – A device, or set of devices, that manages, commands, directs or regulates the behaviors of other devices or systems.

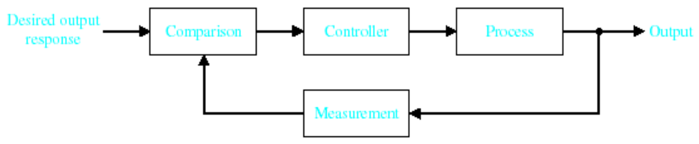
Introduction

Open-Loop Control Systems utilize a control actuator to obtain the desired response.



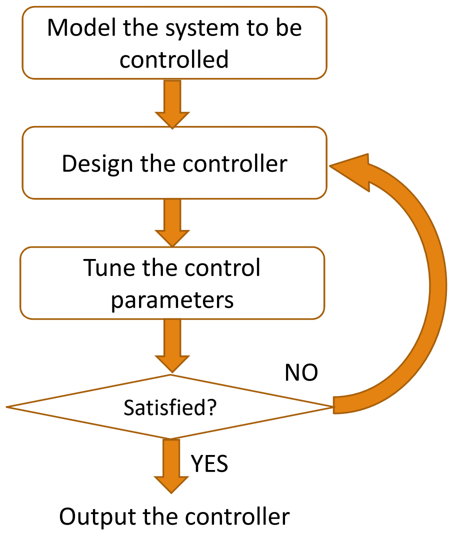
Open-loop control system (without feedback).

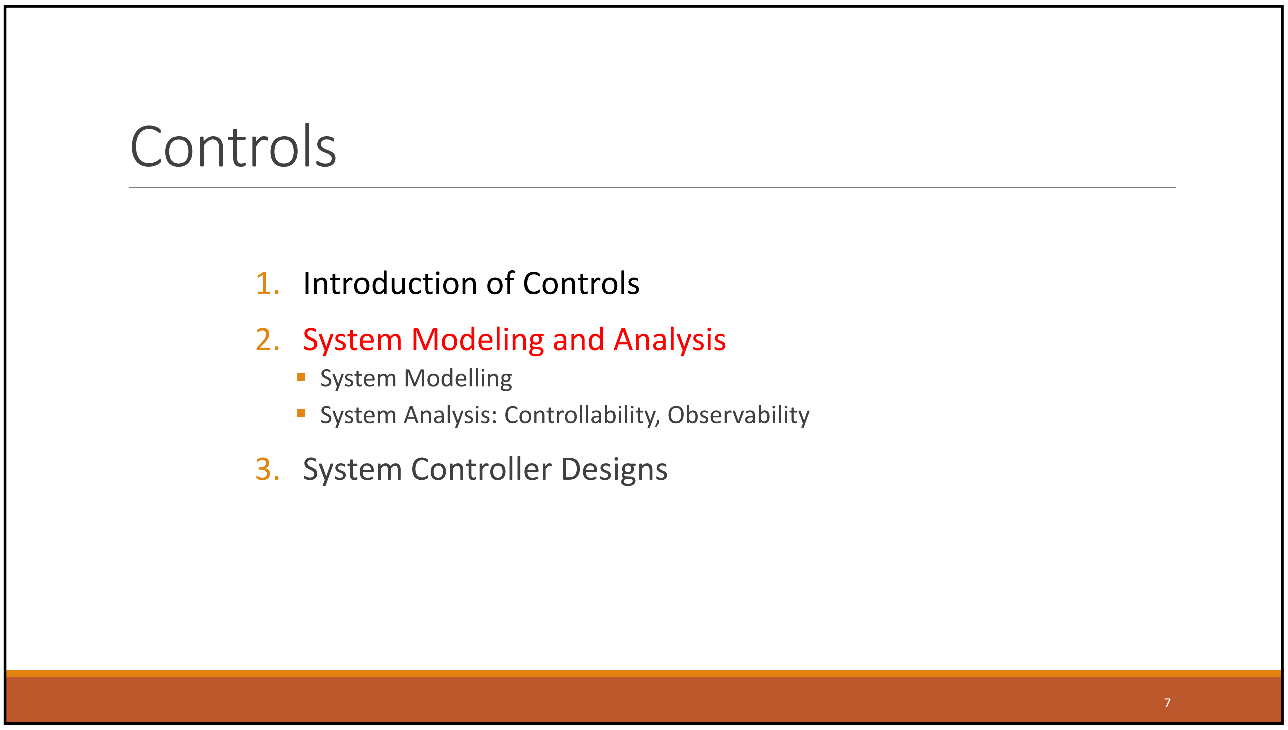
Closed-Loop Control Systems utilizes a feedback controller to compare the actual output to the desired output response.



Closed-loop feedback control system (with feedback).

Control System Design (in-short)

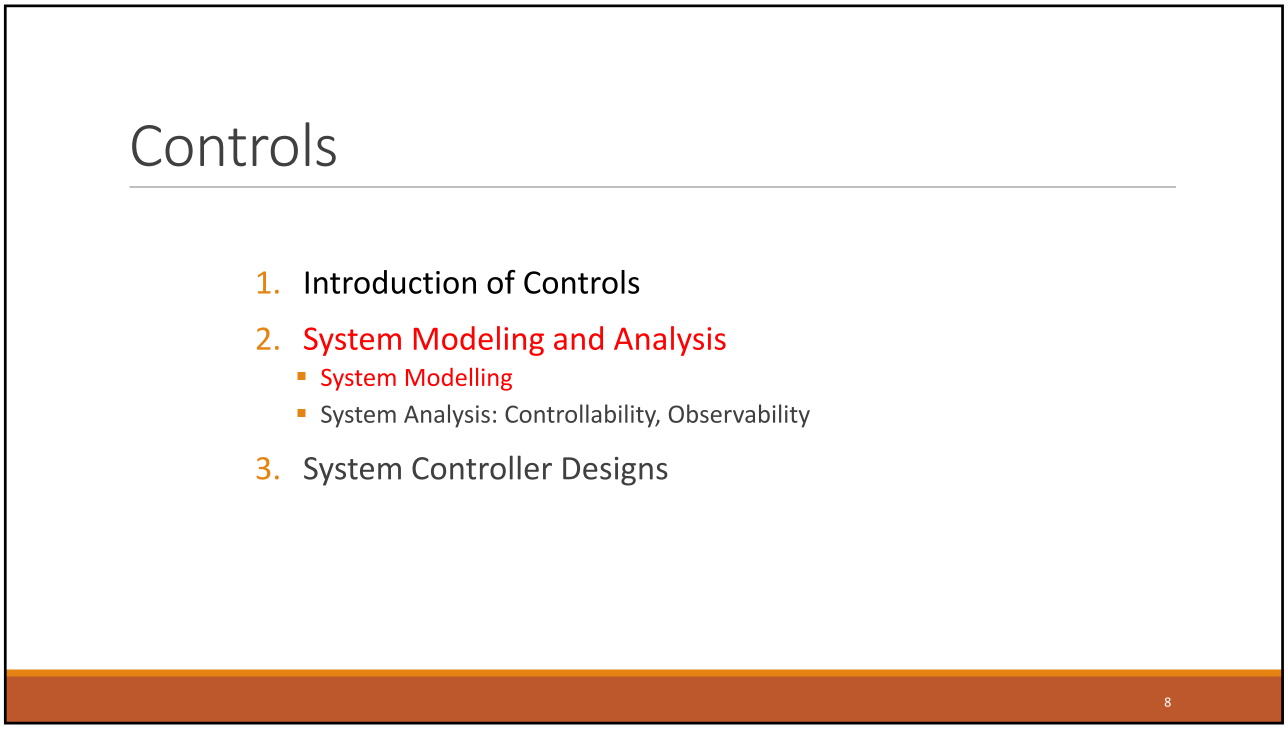

Model the system to be controlled
Design the controller
Tune the control parameters
Satisfied?
NO
YES
Output the controller



Controls

1. Introduction of Controls
2. System Modeling and Analysis
   - System Modelling
   - System Analysis: Controllability, Observability
3. System Controller Designs

7



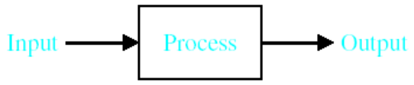
Controls

1. Introduction of Controls
2. System Modeling and Analysis
   - System Modelling
   - System Analysis: Controllability, Observability
3. System Controller Designs

8

System Modeling

- System modeling is to develop a model (typically a mathematical model) to represent the relationship between the system input and system output.



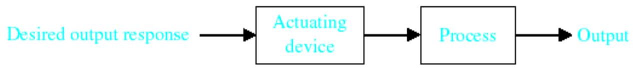
- The power of a mathematical model lies in the fact that it can be used as a basis for synthesizing controllers and also can be simulated in situations for verification.

9

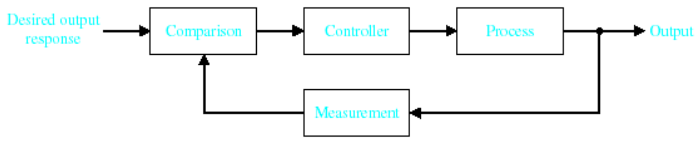
Modeling Requirements

In building a model, it is important to bear in mind that all real processes are complex and hence any attempt to build an exact description of the plant is usually an impossible goal.

Fortunately, feedback is usually very forgiving and hence, in the closed-loop control system, one can usually use rather simple models, provided they could capture the essential features of the problem.



Open-loop control: need an precise model, few disturbances/noises



Closed-loop control: need an approximate model, can handle disturbances/noises

General Modeling Approaches

- **Black Box Approach**

Use a specific model structure to build a plant model which contains some model parameters. In this approach, one tunes the model parameters either by trial and error or by a learning algorithm until the dynamic behaviors of model and plant match sufficiently well.

- **Physical Laws based Approach**

Use physical laws (such as Newton's laws, conservation of mass, energy and momentum) to construct the plant model. In this approach one uses the fact that, in any real system, there are *basic physical laws* which determine the relationships between all the signals in the system.

This course mainly studies the second approach: **State Space Models**

State Space Models

- **State Space Model of Linear Systems** (This course focuses on Linear Systems)

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned} \qquad \text{Initial state: } x(t_0) = x_0$$

where $x \in \mathrm{R}^n$ is the state vector, $u \in \mathrm{R}^m$ is the control signal, $y \in \mathrm{R}^p$ is the output, $x_0 \in \mathrm{R}^n$ is the state vector at time $t = t_0$ and **A**, **B**, **C**, and **D** are matrices of appropriate dimensions.

- **State Space Model of Nonlinear Systems**

$$\begin{aligned}\dot{x}(t) &= f(x(t), u(t), t) \\ y(t) &= g(x(t), u(t), t)\end{aligned} \qquad \text{Initial state: } x(t_0) = x_0$$

State Space Model

Modeling Procedures:

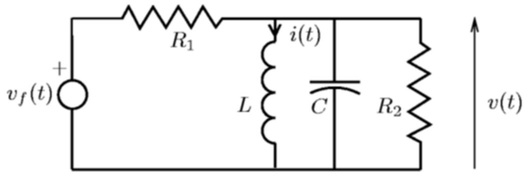
Step 1: Identify system input and output, and derive mathematical equations to represent the relationship between the input and output

Step 2: Define state space model's input u(t), output y(t), and states x(t), and write the state space model in the format below based on Step 1.

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

Example: Electrical Network

Consider the simple electrical network shown below. Assume we want to model the relationship between input voltage v_f(t) and output voltage *v*(*t*)



Step 1: Input: v_f(t), output: v(t)
Apply fundamental network laws and we obtain the following equations:

$$v(t) = L\frac{di(t)}{dt}$$
$$\frac{v_f(t) - v(t)}{R_1} = i(t) + C\frac{dv(t)}{dt} + \frac{v(t)}{R_2}$$

Example: Electrical Network

These equations can be rearranged as follows:

$$\frac{di(t)}{dt} = \frac{1}{L}v(t)$$
$$\frac{dv(t)}{dt} = -\frac{1}{C}i(t) - \left(\frac{1}{R_1C} + \frac{1}{R_2C}\right)v(t) + \frac{1}{R_1C}v_f(t)$$

Step 2: Define u(t)=v_f(t), y(t)=v(t), $x(t) = \begin{bmatrix} i(t) \\ v(t) \end{bmatrix}$, then

$$\frac{di(t)}{dt} = 0i(t) + \frac{1}{L}v(t) + 0v_f(t)$$
$$\frac{dv(t)}{dt} = -\frac{1}{C}i(t) - \left(\frac{1}{R_1C} + \frac{1}{R_2C}\right)v(t) + \frac{1}{R_1C}v_f(t)$$

$$y(t) = 0i(t) + 1v(t) + 0v_f(t)$$

Example: Electrical Network

Step 2: Define u(t)=v_f(t), y(t)=v(t), $x(t) = \begin{bmatrix} i(t) \\ v(t) \end{bmatrix}$, then

$$\frac{di(t)}{dt} = 0i(t) + \frac{1}{L}v(t) + 0v_f(t)$$
$$\frac{dv(t)}{dt} = -\frac{1}{C}i(t) - \left(\frac{1}{R_1C} + \frac{1}{R_2C}\right)v(t) + \frac{1}{R_1C}v_f(t)$$

$$\Rightarrow \quad \dot{x}(t) = \begin{bmatrix} \frac{di(t)}{dt} \\ \frac{dv(t)}{dt} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{L} \\ -\frac{1}{C} & -\left(\frac{1}{R_1C} + \frac{1}{R_2C}\right) \end{bmatrix} \begin{bmatrix} i(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{R_1C} \end{bmatrix} v_f(t)$$

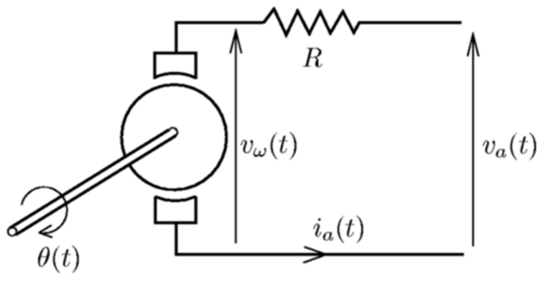
$$y(t) = 0i(t) + 1v(t) + 0v_f(t) \quad \Rightarrow \quad y(t) = [0 \quad 1] \begin{bmatrix} i(t) \\ v(t) \end{bmatrix} + 0\, v_f(t)$$

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$\mathrm{y}(t) = Cx(t) + Du(t)$$

$$\mathbf{A} = \begin{bmatrix} 0 & \frac{1}{L} \\ -\frac{1}{C} & -\left(\frac{1}{R_1C} + \frac{1}{R_2C}\right) \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{R_1C} \end{bmatrix}; \quad \mathbf{C} = [0 \quad 1]; \quad \mathbf{D} = \mathbf{0}$$

Example: DC Motor

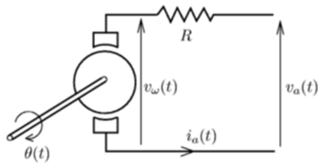
Consider a separately excited DC motor. Let $v_a(t)$ denote the armature voltage, and $\theta(t)$ is the output angle. We need to model the relationship between input $v_a(t)$ and output $\theta(t)$.



Simplified model of a DC motor

Example: DC Motor

J	- be the inertia of the shaft
$\tau_e(t)$	- the electrical torque
$i_a(t)$	- the armature current
$k_1; k_2$	- constants
R	- the armature resistance



Step 1: Input $v_a(t)$, output $\theta(t)$. Based on physical principles, we have

$$J\ddot{\theta}(t) = \tau_e(t) = k_1 i_a(t) \qquad i_a(t) = \frac{v_a(t) - k_2\dot{\theta}(t)}{R}$$

Step 2: Define u(t), y(t), x(t) and write:

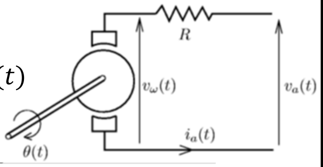
$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

Hints: $\ddot{\theta}(t) = \frac{-k_1k_2}{JR}\dot{\theta}(t) + \frac{k_1}{JR}v_a(t)$

$$x(t) = \begin{bmatrix} \theta(t) \\ \dot{\theta}(t) \end{bmatrix}$$

Example: DC Motor

$$\ddot{\theta}(t) = \frac{-k_1 k_2}{JR}\dot{\theta}(t) + \frac{k_1}{JR}v_a(t)$$



Step 2: Define u(t)=v_a(t), y(t)=θ(t), $x(t) = \begin{bmatrix} \theta(t) \\ \dot{\theta}(t) \end{bmatrix}$, then

$$\dot{\theta}(t) = 0\theta(t) + 1\dot{\theta}(t) + 0v_a(t)$$

$$\ddot{\theta}(t) = 0\theta(t) + \frac{-k_1 k_2}{JR}\dot{\theta}(t) + \frac{k_1}{JR}v_a(t)$$

$$\Rightarrow \quad \dot{x}(t) = \begin{bmatrix} \frac{d\theta(t)}{dt} \\ \frac{d\dot{\theta}(t)}{dt} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \frac{-k_1 k_2}{JR} \end{bmatrix} \begin{bmatrix} \theta(t) \\ \dot{\theta}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{k_1}{JR} \end{bmatrix} v_a(t)$$

$$y(t) = 1\theta(t) + 0\dot{\theta}(t) + 0v_a(t)$$

$$\Rightarrow \quad y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta(t) \\ \dot{\theta}(t) \end{bmatrix} + 0\, v_a(t)$$

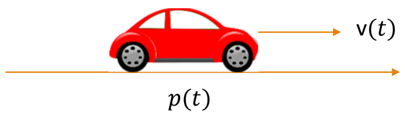
System model:

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

$$A = \begin{bmatrix} 0 & 1 \\ 0 & \frac{-k_1 k_2}{JR} \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \frac{k_1}{JR} \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad D = 0$$

Example: Vehicle Position Control

Vehicle Position Control by Speed



Step 1: Input: v(t), output: p(t), and we have:

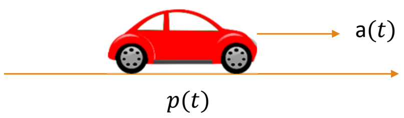
$$\dot{p}(t) = v(t)$$

Step 2: u(t)=v(t), y(t)=p(t), x(t)=p(t), and we have:

$$\dot{x}(t) = Ax(t) + Bu(t) \qquad \dot{x}(t) = 0x(t) + 1u(t)$$
$$y(t) = Cx(t) + Du(t) \qquad y(t) = 1x(t) + 0u(t)$$

Example: Vehicle Position Control

Vehicle Position Control by Acceleration



Model the relationship between vehicle position p(t) and vehicle acceleration a(t)?

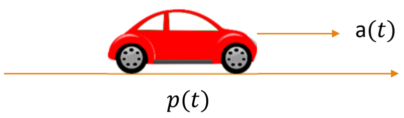
Step 1: Identify system input and output, and derive mathematical equations to represent the relationship between the input and output

Step 2: Define state space model's input u(t), output y(t), and states x(t), and write the state space model in the format below based on Step 1.

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

Example: Vehicle Position Control

Vehicle Position Control by Acceleration



Step 1: Input: a(t), output: p(t), and we have:

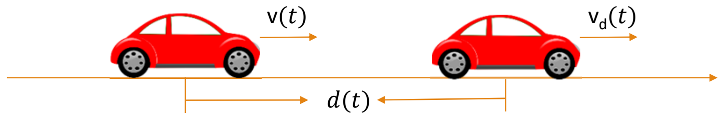
$$\begin{aligned}\dot{p}(t) &= v(t) \\ \dot{v}(t) &= a(t)\end{aligned} \qquad \begin{aligned}\dot{p}(t) &= 0p(t) + 1v(t) + 0a(t) \\ \dot{v}(t) &= 0p(t) + 0v(t) + 1a(t)\end{aligned}$$

Step 2: u(t)=a(t), y(t)=p(t), $x(t) = \begin{bmatrix} p(t) \\ v(t) \end{bmatrix}$, and we have:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned} \qquad \begin{aligned}\dot{x}(t) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) + 0u(t)\end{aligned}$$

Example: Vehicle Position Control

Adaptive Cruise Control by Speed



Model the relationship between inter-vehicle distance d(t) and vehicle speed v(t)? (We have sensors to measure the front vehicle speed $v_d(t)$)

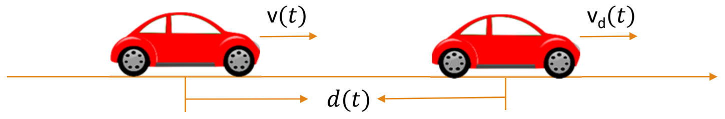
Step 1: Identify system input and output, and derive mathematical equations to represent the relationship between the input and output

Step 2: Define state space model's input u(t), output y(t), and states x(t), and write the state space model in the format below based on Step 1.

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

Example: Vehicle Position Control

Adaptive Cruise Control by Speed



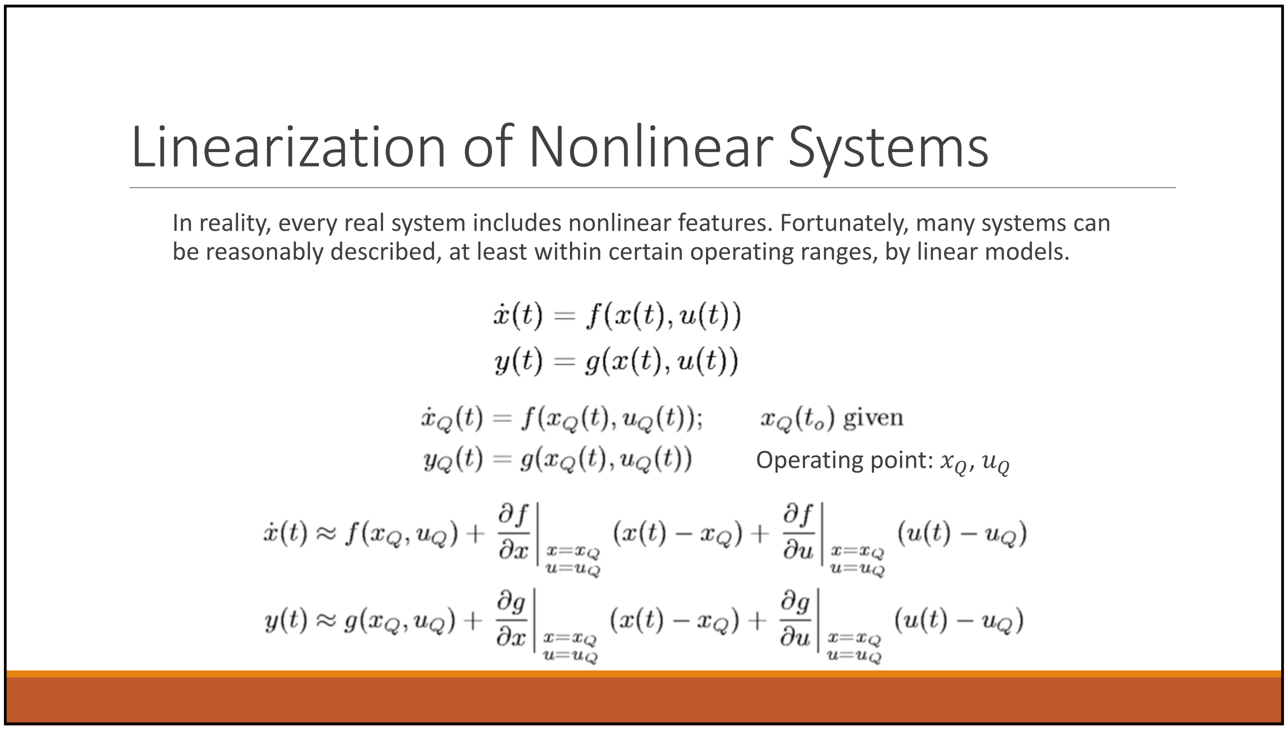
Step 1: Input: v(t), output: d(t), and we have:

$$\dot{d}(t) = v_d(t) - v(t)$$

Step 2: *u(t)=*$v_d(t) - v(t)$*, y(t)=d(t), x=d(t),* and we have:

$$\dot{x}(t) = Ax(t) + Bu(t) \qquad \dot{x}(t) = 0x(t) + 1u(t)$$
$$y(t) = Cx(t) + Du(t) \qquad y(t) = 1x(t) + 0u(t)$$

Note: $v_d(t)$ can be measured using radar and vehicle speed sensor.



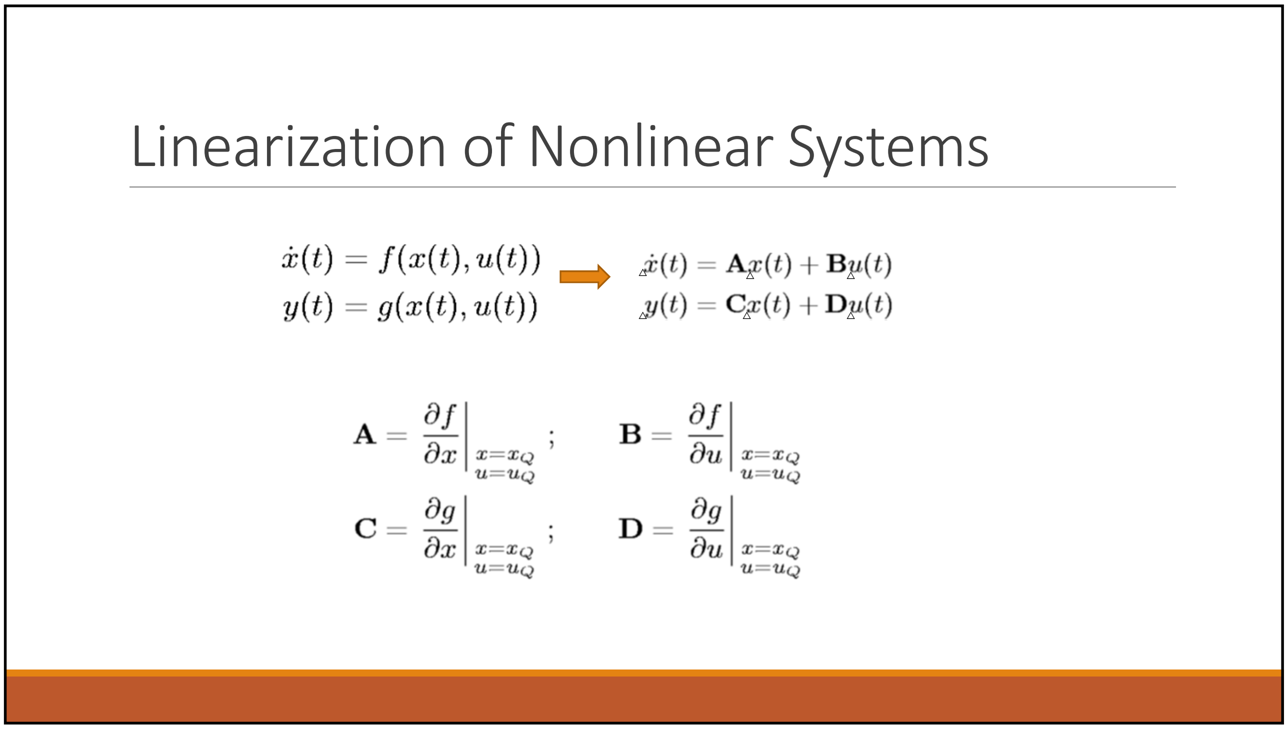
Linearization of Nonlinear Systems

In reality, every real system includes nonlinear features. Fortunately, many systems can be reasonably described, at least within certain operating ranges, by linear models.

$$\dot{x}(t) = f(x(t), u(t))$$
$$y(t) = g(x(t), u(t))$$

$$\dot{x}_Q(t) = f(x_Q(t), u_Q(t)); \qquad x_Q(t_o) \text{ given}$$
$$y_Q(t) = g(x_Q(t), u_Q(t)) \qquad \text{Operating point: } x_Q, u_Q$$

$$\dot{x}(t) \approx f(x_Q, u_Q) + \left.\frac{\partial f}{\partial x}\right|_{\substack{x=x_Q\\u=u_Q}} (x(t) - x_Q) + \left.\frac{\partial f}{\partial u}\right|_{\substack{x=x_Q\\u=u_Q}} (u(t) - u_Q)$$
$$y(t) \approx g(x_Q, u_Q) + \left.\frac{\partial g}{\partial x}\right|_{\substack{x=x_Q\\u=u_Q}} (x(t) - x_Q) + \left.\frac{\partial g}{\partial u}\right|_{\substack{x=x_Q\\u=u_Q}} (u(t) - u_Q)$$



Linearization of Nonlinear Systems

$$\begin{aligned}\dot{x}(t) &= f(x(t), u(t))\\ y(t) &= g(x(t), u(t))\end{aligned} \quad \Rightarrow \quad \begin{aligned}\Delta\dot{x}(t) &= \mathbf{A}\Delta x(t) + \mathbf{B}\Delta u(t)\\ \Delta y(t) &= \mathbf{C}\Delta x(t) + \mathbf{D}\Delta u(t)\end{aligned}$$

$$\mathbf{A} = \left.\frac{\partial f}{\partial x}\right|_{\substack{x=x_Q\\u=u_Q}}; \qquad \mathbf{B} = \left.\frac{\partial f}{\partial u}\right|_{\substack{x=x_Q\\u=u_Q}}$$
$$\mathbf{C} = \left.\frac{\partial g}{\partial x}\right|_{\substack{x=x_Q\\u=u_Q}}; \qquad \mathbf{D} = \left.\frac{\partial g}{\partial u}\right|_{\substack{x=x_Q\\u=u_Q}}$$

Example: Linearization

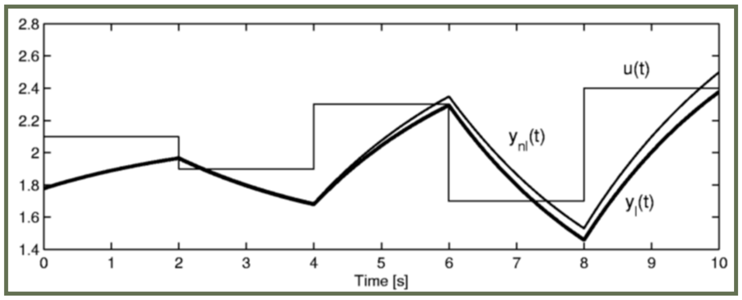
Consider a continuous time system with true model given by

$$\frac{dx(t)}{dt} = f(x(t), u(t)) = -\sqrt{x(t)} + \frac{(u(t))^2}{3}$$

Assume that the input *u*(*t*) fluctuates around *u* = 2. Find an operating point with $u_Q = 2$ and a linearized model around it.

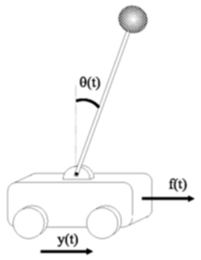
$$\frac{d\Delta x(t)}{dt} = -\frac{3}{8}\Delta x(t) + \frac{4}{3}\Delta u(t)$$

Example: Linearization



Nonlinear system output, $y_{nl}(t)$, and linearized system output, $y_l(t)$, for a square wave input of increasing amplitude, u(t).

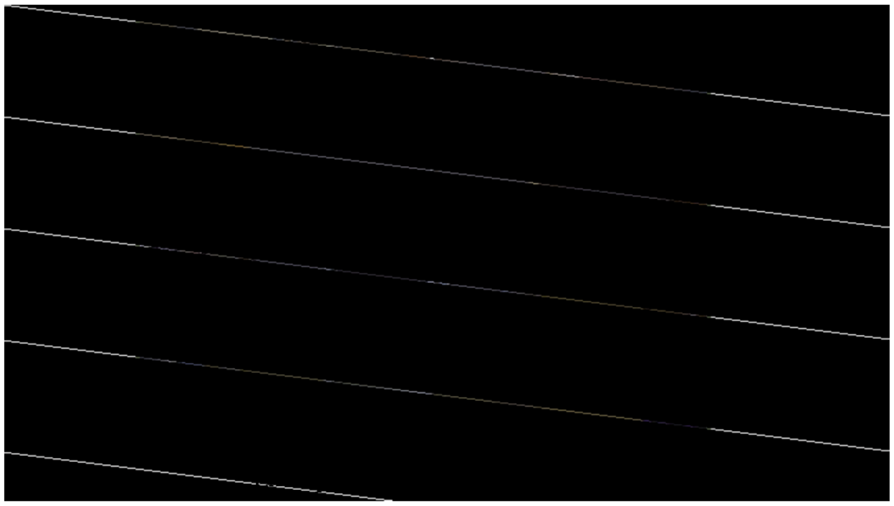
Example: Inverted Pendulum



Inverted pendulum

$y(t)$	-	distance from some reference point
$\theta(t)$	-	angle of pendulum
M	-	mass of cart
m	-	mass of pendulum (assumed concentrated at tip)
l	-	length of pendulum
$f(t)$	-	forces applied to pendulum

Example: Inverted Pendulum



Example: Inverted Pendulum

Application of Newtonian physics to this system leads to the following model:

$$\ddot{y} = \frac{1}{\lambda_m + \sin^2\theta(t)}\left[\frac{f(t)}{m} + \dot{\theta}^2(t)\ell\sin\theta(t) - g\cos\theta(t)\sin\theta(t)\right]$$

$$\ddot{\theta} = \frac{1}{\ell\lambda_m + \sin^2\theta(t)}\left[-\frac{f(t)}{m}\cos\theta(t) + \dot{\theta}^2(t)\ell\sin\theta(t)\cos\theta(t) + (1-\lambda_m)g\sin\theta(t)\right]$$

where $\lambda_m = (M/m)$

Example: Inverted Pendulum

$$\ddot{y} = \frac{1}{\lambda_m + \sin^2\theta(t)}\left[\frac{f(t)}{m} + \dot{\theta}^2(t)\ell\sin\theta(t) - g\cos\theta(t)\sin\theta(t)\right]$$

$$\ddot{\theta} = \frac{1}{\ell\lambda_m + \sin^2\theta(t)}\left[-\frac{f(t)}{m}\cos\theta(t) + \dot{\theta}^2(t)\ell\sin\theta(t)\cos\theta(t) + (1-\lambda_m)g\sin\theta(t)\right]$$

The Inverted Pendulum model can be approximated by a linear state space model described by:

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{(M+m)g}{M\ell} & 0 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ -\frac{1}{M\ell} \end{bmatrix}; \quad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{D} = \mathbf{0}$$

Controls

1. Introduction of Controls
2. System Modeling and Analysis
   - System Modelling
   - System Analysis: Controllability, Observability
3. System Controller Designs

System Model Analysis

System Model

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

- **Controllability**

Controllability tells us about the feasibility to control a system.

- **Observability**

Observability tells us about whether it is possible to know what is happening in a given system by observing its outputs.

Controllability

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

An important question that lies at the heart of control using state space models is whether we can steer the state via the control input to certain locations in the state space. Technically, this property is called controllability or reachability.

The issue of controllability concerns whether a given initial state x_0 can be steered to the origin in finite time using the input $u(t)$.

Definition: A state x_0 is said to be controllable if there exists a finite interval $[0, T]$ and an input $\{u(t), t \in [0, T]\}$ such that $x(T) = 0$. If all states are controllable, then the system is said to be completely controllable.

Controllability

Theorem: Consider the state space model

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

(*i*) The set of all controllable states is the range space of the controllability matrix $\Gamma_c[\mathbf{A}, \mathbf{B}]$, where

$$\mathbf{\Gamma}_c[\mathbf{A},\mathbf{B}] \triangleq \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A^2B} & \dots & \mathbf{A^{n-1}B} \end{bmatrix}$$

(*ii*) The model is completely controllable if and only if $\Gamma_c[\mathbf{A}, \mathbf{B}]$ has full row rank.

Controllability Example

Consider the state space model

$$\mathbf{A} = \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

The controllability matrix is given by

$$\mathbf{\Gamma}_c[\mathbf{A}, \mathbf{B}] = [\mathbf{B}, \mathbf{AB}] = \begin{bmatrix} 1 & -4 \\ -1 & -2 \end{bmatrix}$$

Clearly, rank Γ_c[**A**, **B**] = 2; thus, the system is completely controllable.

Controllability Example

Consider the state space model

$$\mathbf{A} = \begin{bmatrix} -1 & 1 \\ 2 & 0 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

The controllability matrix is given by:

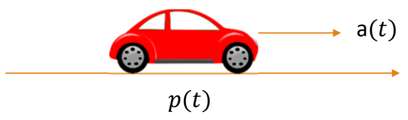
$$\mathbf{\Gamma}_c[\mathbf{A}, \mathbf{B}] = [\mathbf{B}, \mathbf{AB}] = \begin{bmatrix} 1 & -2 \\ -1 & 2 \end{bmatrix}$$

Rank Γ_c[**A**, **B**] = 1 < 2; thus, the system is not completely controllable.

$$\boldsymbol{\Gamma}_c[\mathbf{A},\mathbf{B}] \triangleq [\mathbf{B} \quad \mathbf{AB} \quad \mathbf{A^2B} \quad \dots \quad \mathbf{A^{n-1}B}]$$

Controllability Example

Vehicle Position Control by Acceleration



$$x(t) = \begin{bmatrix} p(t) \\ v(t) \end{bmatrix} \qquad \begin{aligned} \dot{x}(t) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} a(t) \\ \mathrm{y}(t) &= [1 \quad 0]x(t) + 0a(t) \end{aligned}$$

Is the system completely controllable?

$$\Gamma_c[\mathbf{A},\mathbf{B}] = [B, AB] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Rank $\Gamma_c[\mathbf{A},\mathbf{B}] = 2$; thus, the system is completely controllable.

Observability

Observability is concerned with the issue of what can be said about the state given measurements of the plant output.

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ \mathrm{y}(t) &= Cx(t) + Du(t) \end{aligned}$$

A formal definition is as follows:

Definition: The state $x_0 \neq 0$ is said to be unobservable if, given $x(0) = x_0$, and $u[k] = 0$ for $k \geq 0$, then $y[k] = 0$ for $k \geq 0$. The system is said to be completely observable if there exists no nonzero initial state that it is unobservable.

Observability

Theorem: Consider the state model

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

(*i*) The set of all unobservable states is equal to the null space of the observability matrix Γ_0[**A**, **C**], where

$$\mathbf{\Gamma}_o[\mathbf{A},\mathbf{C}] \triangleq \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}$$

(*ii*) The system is completely observable if and only if Γ_0[**A**, **C**], has full column rank.

Observability Example

Consider the following state space model:

$$\mathbf{A} = \begin{bmatrix} -3 & -2 \\ 1 & 0 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad \mathbf{C} = \begin{bmatrix} 1 & -1 \end{bmatrix}$$

Then

$$\mathbf{\Gamma}_o[\mathbf{A},\mathbf{C}] = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -4 & -2 \end{bmatrix}$$

Hence, rank Γ_0[**A**, **C**] = 2, and the system is completely observable.

Observability Example

Consider the following state space model:

$$\mathbf{A} = \begin{bmatrix} 1 & -2 \\ -1 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \qquad \mathbf{C} = \begin{bmatrix} 1 & -1 \end{bmatrix}$$

Here

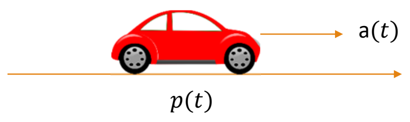
$$\Gamma_o[\mathbf{A}, \mathbf{C}] = \begin{bmatrix} 1 & -1 \\ 2 & -2 \end{bmatrix}$$

Hence, rank $\Gamma_0[\mathbf{A}, \mathbf{C}] = 1 < 2$, and the system is not completely observable.

Observability Example

$$\mathbf{\Gamma}_o[\mathbf{A}, \mathbf{C}] \triangleq \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}$$

Vehicle Position Control by Acceleration



$$x(t) = \begin{bmatrix} p(t) \\ v(t) \end{bmatrix} \qquad \begin{aligned} \dot{x}(t) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} a(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) + 0a(t) \end{aligned}$$

We can measure p(t)

Is the system completely observable?

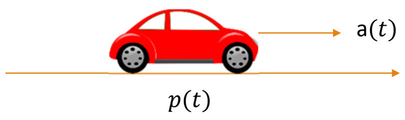
$$\Gamma_o[\mathbf{A}, \mathbf{C}] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Rank $\Gamma_c[\mathbf{A}, \mathbf{C}] = 2$; thus, the system is completely observable.

Observability Example

$$\boldsymbol{\Gamma}_o[\mathbf{A},\mathbf{C}] \triangleq \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}$$

Vehicle Position Control by Acceleration



$$x(t) = \begin{bmatrix} p(t) \\ v(t) \end{bmatrix} \qquad \dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} a(t)$$

$$\mathrm{y}(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} x(t) + 0a(t)$$

We can measure v(t)

Is the system completely observable?

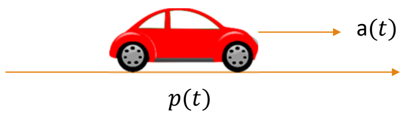
$$\Gamma_o[\mathbf{A},\mathbf{C}] = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Rank Γ_c[**A**, **C**] = 1; thus, the system is not completely observable.

Observability Example

$$\boldsymbol{\Gamma}_o[\mathbf{A},\mathbf{C}] \triangleq \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}$$

Vehicle Position Control by Acceleration



$$x(t) = \begin{bmatrix} p(t) \\ v(t) \end{bmatrix} \qquad \dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} a(t)$$

$$\mathrm{y}(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x(t) + 0a(t)$$

We can measure p(t) and v(t)

Is the system completely observable?

$$\Gamma_o[\mathbf{A},\mathbf{C}] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Rank Γ_c[**A**, **C**] = 2; thus, the system is completely observable.

Controls

1. Introduction of Controls
2. System Modeling and Analysis
3. System Controller Designs
   - Problem Statement
   - Stability
   - Model based Controller Design
   - Non-Model based Controller

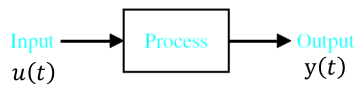
Controls

1. Introduction of Controls
2. System Modeling and Analysis
3. System Controller Designs
   - Problem Statement
   - Stability
   - Model based Controller Design
   - Non-Model based Controller

Problem Statement

Controller Design Problem

Given a system/plant:



and given a desired output: $y^d(t)$

Design a controller $u(t) = U(y^d(t), y(t))$ to make

$$y(t) \rightarrow y^d\ (t)$$

Controls

50

Stability

Control errors = desired states – actual states

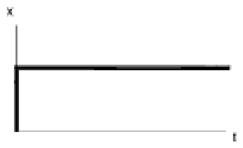
A control system is called

Stable: if the control errors are bounded

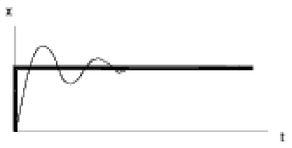
- **Asymptotically Stable:** if the control errors approach to zero
- **Exponentially Stable:** if the control errors approach to zero exponentially
- **Marginally Stable:** if the control errors oscillate within a bound

Unstable: if the control errors go unbounded

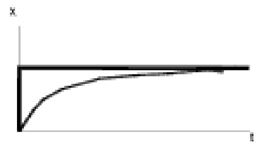
Stability



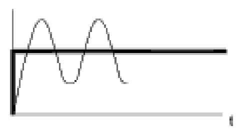
Desired States



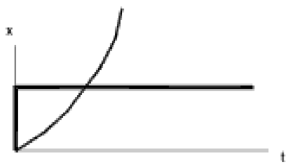
Asymptotically Stable



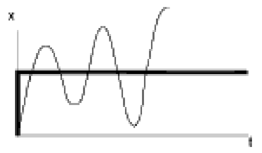
Exponentially Stable



Marginally Stable



Unstable



Unstable

Controls

System Controller Design

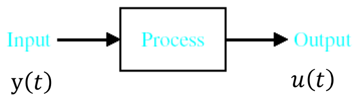
Model based Controller Design

First build system model

$$\dot{x}(t) = Ax(t) + Bu(t) \qquad x(t_0) = x_0$$
$$y(t) = Cx(t) + Du(t)$$

Design a controller $u(t) = U(y^d(t), y(t))$, and calculate the optimal control parameters based on system model to make $y(t) \rightarrow y^d\ (t)$

Non-Model based Controller Design



Use a controller with a general form

$$u(t) = U(y^d(t), y(t))$$

and manually tune the control parameters based on performance and experience.

Model based Controller Design

Full-State Feedback Controller

- Problem Statement
- Controller Design
- Control Parameter Region Analysis
- Control Parameter Selection (Linear Quadratic Regulator)
- Practical Control Implementation (Model Predictive Control)

Full-State Feedback Controller

- Problem Statement
- Controller Design
- Control Parameter Region Analysis
- Control Parameter Selection (Linear Quadratic Regulator)
- Control Parameter Optimization for Practical Control Implementation (Model Predictive Control)

Full-State Feedback Controller

Problem Statement

Given a system/plant: $\dot{x}(t) = Ax(t) + Bu(t) \quad x\,(t_0) = x_0$

$$\mathrm{y}(t) = x(t)$$

and given a desired output: $x^d(t)$

Design a controller $u(t) = U(x^d(t), x(t))$ to make $x(t) \rightarrow x^d\,(t)$

Full-State Feedback Controller

- Problem Statement
- Controller Design
- Control Parameter Region Analysis
- Control Parameter Selection (Linear Quadratic Regulator)
- Practical Control Implementation (Model Predictive Control)

Full-State Feedback Controller

Control Design $\dot{x}(t) = Ax(t) + Bu(t) \qquad x\,(t_0) = x_0$

Consider a special case: $x^d(t) = 0$

Controller: $u(t) = -Kx(t)$

Controlled System:

$$\begin{aligned}\dot{x}(t) &= Ax(t) - BKx(t) \qquad x\,(t_0) = x_0 \\ &= (A - BK)x(t) \\ &= \bar{A}x(t)\end{aligned}$$

Stability: $x(t) \to 0$?

What is the range of the control parameter K to make $x(t) \to 0$?

Full-State Feedback Controller

- Problem Statement
- Controller Design
- Control Parameter Region Analysis
- Control Parameter Selection (Linear Quadratic Regulator)
- Control Parameter Optimization for Practical Control Implementation (Model Predictive Control)

Full-State Feedback Control

Controlled System:

$$\begin{aligned}\dot{x}(t) &= Ax(t) - BKx(t) \quad x(t_0) = x_0\\ &= (A - BK)x(t)\\ &= \bar{A}x(t)\end{aligned}$$

Solution: $x(t) = e^{\bar{A}t}x_0$

Matrix Decomposition:

$$\bar{A} = T\Lambda T^{-1} \qquad \Lambda = \begin{bmatrix}\lambda_1 & & \\ & \dots & \\ & & \lambda_n\end{bmatrix} \qquad \begin{aligned}\lambda_i = a_i + jb_i, \quad & j^2 = -1\\ i = 1, \dots, n\end{aligned}$$

Solution:

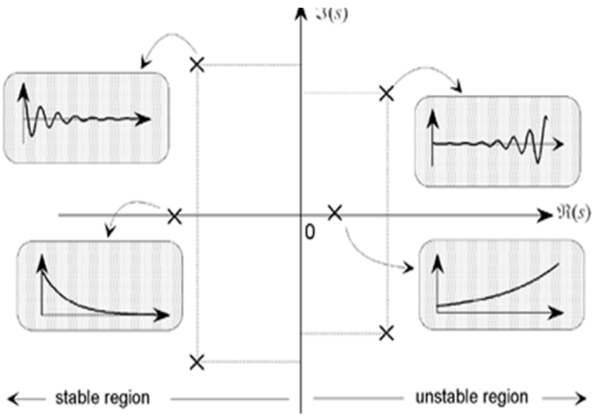
$$x(t) = e^{\bar{A}t}x_0 = T\begin{bmatrix}e^{\lambda_1 t} & & \\ & \dots & \\ & & e^{\lambda_n t}\end{bmatrix}T^{-1}x_0$$

Stability depends on the eigenvalues of $\bar{A}$

$e^{a_i t}$ $\quad e^{jb_i t} = \cos(b_i t) + j\sin(b_i t)$

Full-State Feedback Control

Solutions of x(t) when eigenvalues are at differing quadrants: $\lambda_i = a_i + jb_i$



$e^{a_i t}$ $\quad e^{jb_i t} = \cos(b_i t) + j\sin(b_i t)$

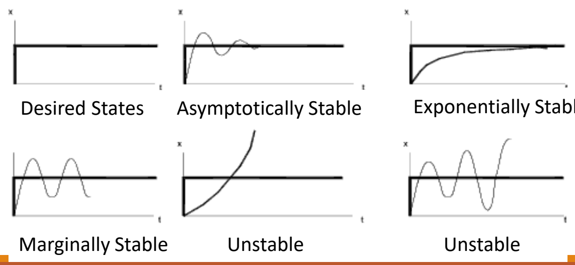
Stability

$\lambda_i = a_i + jb_i \qquad i = 1, \dots, n$

Stable (errors are bounded): if $a_i \leq 0$

- **Asymptotically Stable** (errors approach to zero): if $a_i < 0$
- **Exponentially Stable** (errors approach to zero exponentially): if $a_i < 0$, and $b_i = 0$
- **Marginally Stable** (errors oscillate within a bound): if $a_i = 0$, and b_i not overlap

Unstable (errors go unbounded): if $a_i > 0$, or b_i overlaps on imaginary axis



Stability

Controlled System:

$$\begin{aligned}\dot{x}(t) &= Ax(t) - BKx(t) \qquad x\,(t_0) = x_0 \\ &= (A - BK)x(t) \\ &= \bar{A}x(t)\end{aligned}$$

How to find eigenvalues of $\bar{A}$?

$$\text{Det}(\lambda I - \bar{A}) = 0$$

$$\Downarrow$$

$$\lambda_i = a_i + jb_i \qquad i = 1, \dots, n$$

The Routh-Hurwitz Stability Criterion

$$\text{Det}(sI - \bar{A}) = 0 \longrightarrow a_n s^n + a_{n-1}s^{n-1} + a_{n-2}s^{n-2} + \cdots + a_1 s + a_0 = 0$$

Routh array

$$\begin{array}{c|cccc}
s^n & a_n & a_{n-2} & a_{n-4} & \cdots \\
s^{n-1} & a_{n-1} & a_{n-3} & a_{n-5} & \cdots \\
s^{n-2} & b_{n-1} & b_{n-3} & b_{n-5} & \cdots \\
s^{n-3} & c_{n-1} & c_{n-3} & c_{n-5} & \cdots \\
\bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \\
s^0 & h_{n-1} & & &
\end{array}$$

The Routh-Hurwitz criterion states that the number of roots with positive real parts is equal to the number of changes in sign of the first column of the Routh array.

$$b_{n-1} = \frac{-1}{a_{n-1}}\begin{vmatrix} a_n & a_{n-2} \\ a_{n-1} & a_{n-3} \end{vmatrix} = \frac{a_{n-1}a_{n-2} - a_n a_{n-3}}{a_{n-1}}, \quad b_{n-3} = \frac{-1}{a_{n-1}}\begin{vmatrix} a_n & a_{n-4} \\ a_{n-1} & a_{n-5} \end{vmatrix}, \;\ldots\ldots$$

$$c_{n-1} = \frac{-1}{b_{n-1}}\begin{vmatrix} a_{n-1} & a_{n-3} \\ b_{n-1} & b_{n-3} \end{vmatrix}, \;\ldots\ldots$$

The Routh-Hurwitz Stability Criterion

Case One: No element in the first column is zero.

The Characteristic polynomial of a second-order system is:

$$q(s) = a_2 \cdot s^2 + a_1 \cdot s + a_0$$

The Routh array is written as:

$$\begin{array}{c|cc}
s^2 & a_2 & a_0 \\
s^1 & a_1 & 0 \\
s^0 & b_1 & 0
\end{array}$$

where:

$$b_1 = \frac{a_1 \cdot a_0 - (0) \cdot a_2}{a_1} = a_0$$

Therefore the requirement for a stable second-order system is simply that all coefficients be positive or all the coefficients be negative.

The Routh-Hurwitz Stability Criterion
Case Two: Zero in the first column, and some elements of the row containing the zero are nonzero.

The zero in the first column is replaced by a small positive number ∈

If only one element in the array is zero, it may be replaced with a small positive number ε that is allowed to approach zero after completing the array.

$$q(s) = s^5 + 2s^4 + 2s^3 + 4s^2 + 11s + 10$$

The Routh array is then:

$$\begin{array}{c|ccc} s^5 & 1 & 2 & 11 \\ s^4 & 2 & 4 & 10 \\ s^3 & b_1 & 6 & 0 \\ s^2 & c_1 & 10 & 0 \\ s^1 & d_1 & 0 & 0 \\ s^0 & 10 & 0 & 0 \end{array}$$

where:

$$b_1 = \frac{2\cdot 2 - 1\cdot 4}{2} = 0 = \varepsilon \qquad c_1 = \frac{4\varepsilon - 2\cdot 6}{\varepsilon} = \frac{-12}{\varepsilon} \qquad d_1 = \frac{6\cdot c_1 - 10\varepsilon}{c_1} = 6$$

There are two sign changes in the first column due to the large negative number calculated for c1. Thus, the system is unstable because two roots lie in the right half of the plane.

The Routh-Hurwitz Stability Criterion
Case Three: Zero in the first column, and the other elements of the row are also zero.

1. Construct an auxiliary polynomial based on the row before the all-zero row
2. Replace the all-zero row by the DERIVATIVE of the auxiliary polynomial.

$$D(s) = s^7 + 3s^6 + 3s^5 + s^4 + s^3 + 3s^2 + 3s + 1$$

$$\begin{array}{r|cccc} s^7 & 1 & 3 & 1 & 3 \\ s^6 & 3 & 1 & 3 & 1 \\ s^5 & b_1 = 8/3 & b_2 = 0 & b_3 = 8/3 & \\ \text{row before where zero} \rightarrow s^4 & c_1 = 1 & c_2 = 0 & c_3 = 1 & \\ s^3 & d_1 = 4 & d_2 = 0 & & \\ s^2 & e_1 = \varepsilon > 0 & e_2 = 1 & & \\ s^1 & f_1 = -4/\varepsilon < 0 & & & \\ s^0 & g_1 = 1 & & & \end{array}$$

whole row zero =>

1) last row $s^4 + 0s^2 + 1 = s^4 + 1$

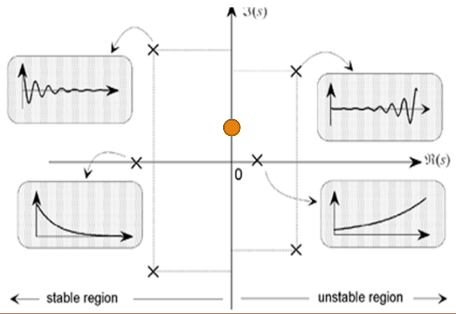
2) $\dfrac{d(s^4+1)}{ds} = 4s^3 + 0s$

Two sign changes, two roots on the right half plane. Unstable system.

The Routh-Hurwitz Stability Criterion
Case Four: Repeated/overlapped roots of the equation on the imaginary j-axis.

Unrepeated/unoverlapped roots on the imaginary j-axis make the system marginally stable.

Repeated/overlapped roots on the j-axis will cause the system to be unstable. Unfortunately, the routh-array will not be able to reveal this instability.



The Routh-Hurwitz Stability Criterion

$$a_n s^n + a_{n-1} s^{n-1} + a_{n-2} s^{n-2} + \cdots + a_1 s + a_0 = 0$$

Routh array

$$\begin{array}{c|c|cccc}
s^n & a_n & a_{n-2} & a_{n-4} & \cdots \\
s^{n-1} & a_{n-1} & a_{n-3} & a_{n-5} & \cdots \\
s^{n-2} & b_{n-1} & b_{n-3} & b_{n-5} & \cdots \\
s^{n-3} & c_{n-1} & c_{n-3} & c_{n-5} & \cdots \\
\bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \\
s^0 & h_{n-1} & & &
\end{array}$$

The number of roots with positive real parts is equal to the number of changes in sign of the first column of the Routh array.

Practice: Find how many roots with positive real parts?

$$D(s) = s^4 + 5s^3 + s^2 + 10s + 1$$

$$b_{n-1} = \frac{(a_{n-1})(a_{n-2}) - a_n(a_{n-3})}{a_{n-1}} = \frac{-1}{a_{n-1}} \begin{vmatrix} a_n & a_{n-2} \\ a_{n-1} & a_{n-3} \end{vmatrix}$$

$$b_{n-3} = \frac{-1}{a_{n-1}} \begin{vmatrix} a_n & a_{n-4} \\ a_{n-1} & a_{n-5} \end{vmatrix}, \quad c_{n-1} = \frac{-1}{b_{n-1}} \begin{vmatrix} a_{n-1} & a_{n-3} \\ b_{n-1} & b_{n-3} \end{vmatrix}$$

Full-State Feedback Controller

Control Design

For a linear system

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = x(t)$$

Consider a special case: $x^d(t) = 0$

$\boxed{\text{Controller: } u(t) = -Kx(t)}$

Use Routh-Hurwitz Stability Criterion to determine the range of K.

What is the controller *u(t)* for any $x^d(t)$ to make: $x(t) \rightarrow x^d\ (t)$?

$\boxed{\text{Controller: } u(t) = -K(x(t) - x^d(t))}$

Full State Feedback Controller Design Procedures

Controller Design Procedures:

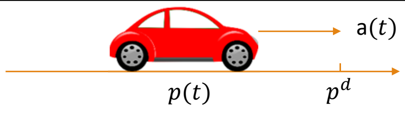
Step 1: Build the state space model of the system

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = x(t)$$

Step 2: Design the controller as $u(t) = -Kx(t)$

Write the controlled system model $\dot{x}(t) = \bar{A}x(t) = (A - BK)x(t)$ and use Routh-Hurwitz Stability Criterion to determine the range of K.

Step 3: Use the controller $u(t) = -K(x(t) - x^d(t))$ to drive $x(t) \rightarrow x^d\ (t)$



Controller Design Example

Vehicle Position Control by Acceleration

Use acceleration to control a vehicle to move to a fixed position p^d

Step 1: $\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \quad x(t) = \begin{bmatrix} p(t) \\ v(t) \end{bmatrix}$

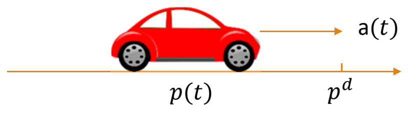
$\mathrm{y}(t) = x(t)$

Step 2: Design the controller as

$$u(t) = -Kx(t) = -[k1 \;\; k2]x(t)$$

The controlled system model is

$$\dot{x}(t) = \bar{A}x(t) = (\mathrm{A} - \mathrm{B}K)x(t) = \left(\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix}[k1 \;\; k2]\right)x(t)$$
$$= \begin{bmatrix} 0 & 1 \\ -k1 & -k2 \end{bmatrix} x(t)$$



Controller Design Example

$$\det(\lambda I - \bar{A}) = 0 \Rightarrow \left| \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -k1 & -k2 \end{bmatrix} \right| = 0$$

$$\Rightarrow \left| \begin{bmatrix} \lambda & -1 \\ k1 & \lambda + k2 \end{bmatrix} \right| = 0 \Rightarrow \lambda^2 + k2\lambda + k1 = 0$$

$$\boxed{\text{Routh array}} \quad \begin{array}{c|cc} \lambda^2 & 1 & k1 \\ \lambda^1 & k2 & 0 \\ \lambda^0 & k1 & 0 \end{array} \Rightarrow \begin{array}{l} k1 > 0 \\ k2 > 0 \end{array}$$

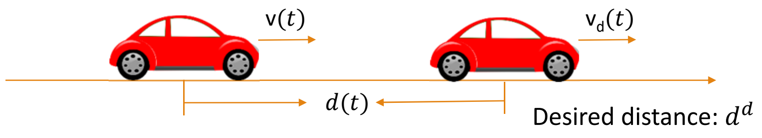
Step 3: Use the controller

$$u(t) = -K\left(x(t) - x^d(t)\right) = -[k1 \;\; k2]\left(\begin{bmatrix} p(t) \\ v(t) \end{bmatrix} - \begin{bmatrix} p^d \\ 0 \end{bmatrix}\right)$$
$$= k1\left(p^d - p(t)\right) - k2v(t)$$

Controller Design Example

Adaptive Cruise Control by Speed

Find v(t) to make $d(t)$ -> d^d ?



Step 1: Build the state space model of the system

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = x(t)$$

Step 2: Design the controller as $u(t) = -Kx(t)$

Write the controlled system model $\dot{x}(t) = \bar{A}x(t) = (A - BK)x(t)$ and use Routh-Hurwitz Stability Criterion to determine the range of K.

Step 3: Use the controller $u(t) = -K(x(t) - x^d(t))$ to drive $x(t) \rightarrow x^d\ (t)$

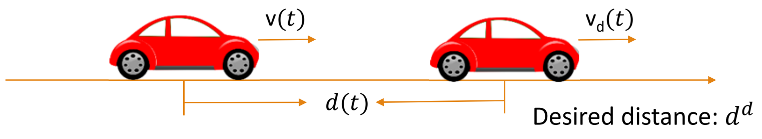
Hint for Step 1:

Define $x(t) = d(t)$, $u(t) = v_d(t) - v(t)$
Then, we have:

$$\dot{x}(t) = u(t)$$
$$y(t) = x(t)$$

Controller Design Example

Adaptive Cruise Control by Speed



Step 1: $\dot{x}(t) = u(t)$, $y(t) = x(t)$ $\qquad x(t) = d(t)$, $u(t) = v_d(t) - v(t)$

Step 2: Design the controller as

$$u(t) = -Kx(t) = -kx(t)$$

The controlled system model is

$$\dot{x}(t) = -kx(t)$$

Controller Design Example

Adaptive Cruise Control by Speed

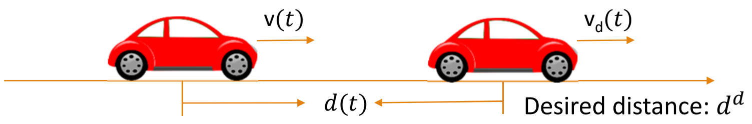
$\det(\lambda I - \bar{A}) = 0 \Rightarrow \lambda + k = 0$ $\boxed{\text{Routh array}}$ $\begin{array}{c|cc} \lambda^1 & 1 & 0 \\ \lambda^0 & k & \end{array} \Rightarrow k > 0$

Step 3: Use the controller

$$u(t) = -K\left(x(t) - x^d(t)\right) = -k(d(t) - d^d)$$

$$u(t) = v_d(t) - v(t)$$

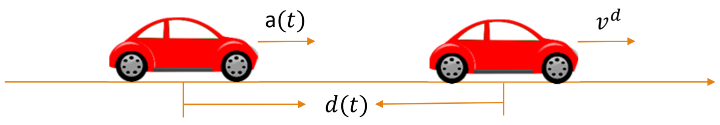
$$v(t) = v_d(t) + k(d(t) - d^d)$$



Controller Design Example

Adaptive Cruise Control by Acceleration

Desired speed: v^d



Desired distance: d^d

- Model the relationship between inter-vehicle distance d(t), vehicle speed v(t) and vehicle acceleration a(t)?
- Design a controller to control the vehicle to keep the desired distance d^d and desired speed v^d?

Full-State Feedback Controller

- Problem Statement
- Controller Design
- Control Parameter Region Analysis
- Control Parameter Selection (Linear Quadratic Regulator)
- Control Parameter Optimization for Practical Control Implementation (Model Predictive Control)

Control Parameter Selection

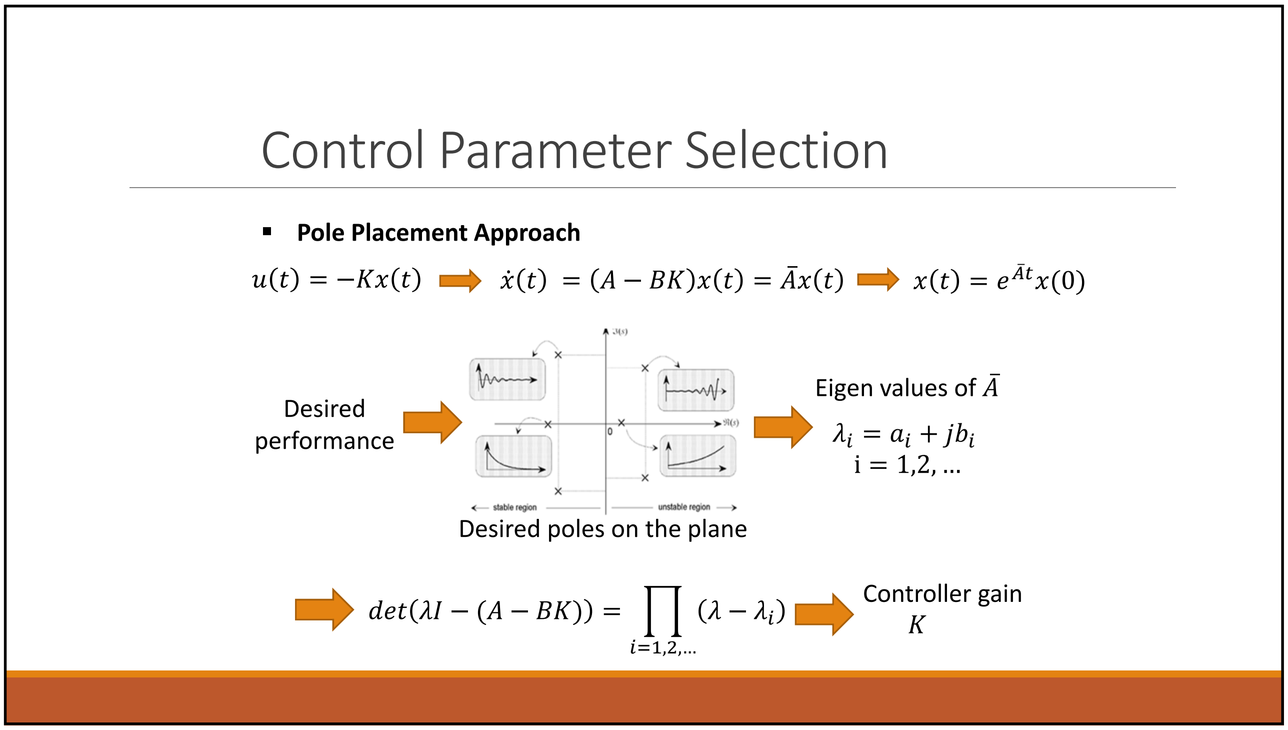
For a linear system

$$\dot{x}(t) = Ax(t) + Bu(t)$$

Controller: $u(t) = -K(x(t) - x^d(t))$

The Routh-Hurwitz Stability Criterion $\Rightarrow$ Range of K

How to select K within this range?

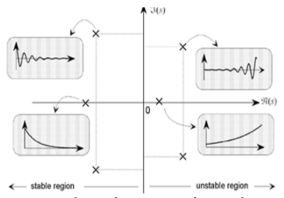


Control Parameter Selection

- **Pole Placement Approach**

$u(t) = -Kx(t) \Rightarrow \dot{x}(t) = (A - BK)x(t) = \bar{A}x(t) \Rightarrow x(t) = e^{\bar{A}t}x(0)$

Desired performance $\Rightarrow$



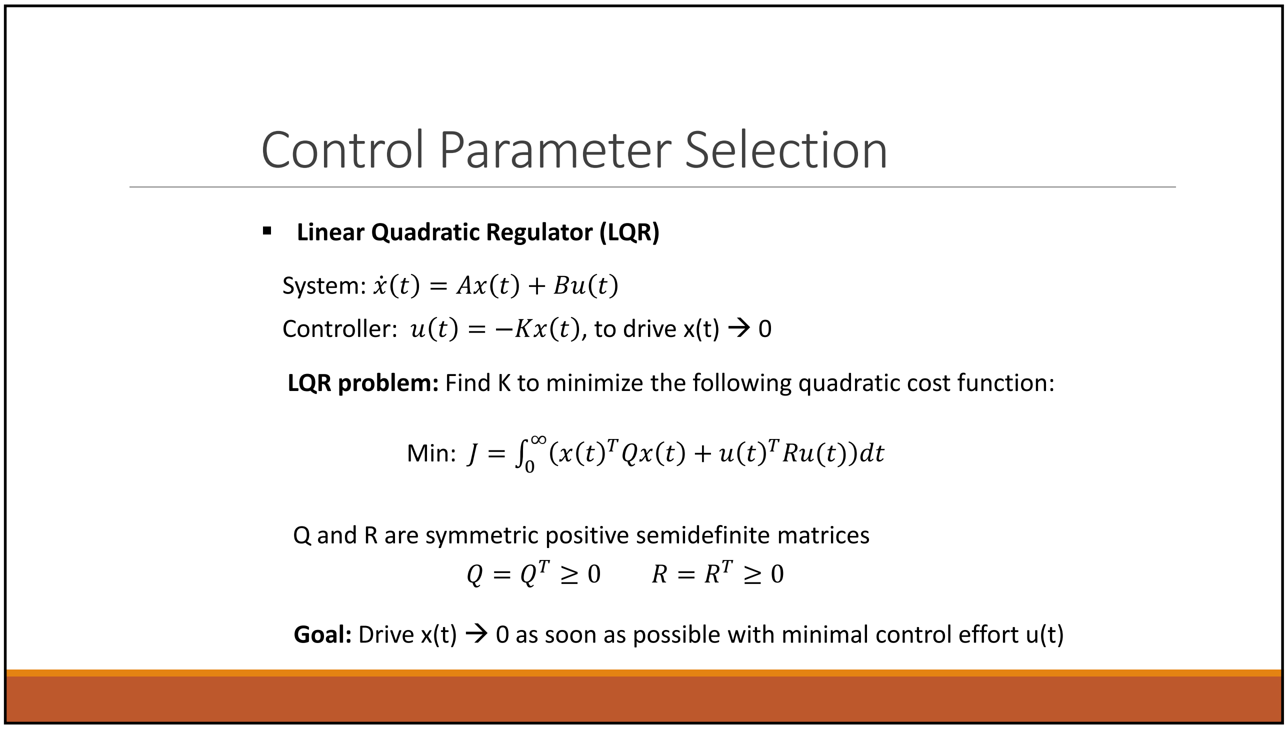
Desired poles on the plane

$\Rightarrow$ Eigen values of $\bar{A}$

$\lambda_i = a_i + jb_i$
i = 1,2, …

$$\Rightarrow det(\lambda I - (A - BK)) = \prod_{i=1,2,\ldots} (\lambda - \lambda_i) \Rightarrow$$

Controller gain K



Control Parameter Selection

- **Linear Quadratic Regulator (LQR)**

System: $\dot{x}(t) = Ax(t) + Bu(t)$

Controller: $u(t) = -Kx(t)$, to drive x(t) → 0

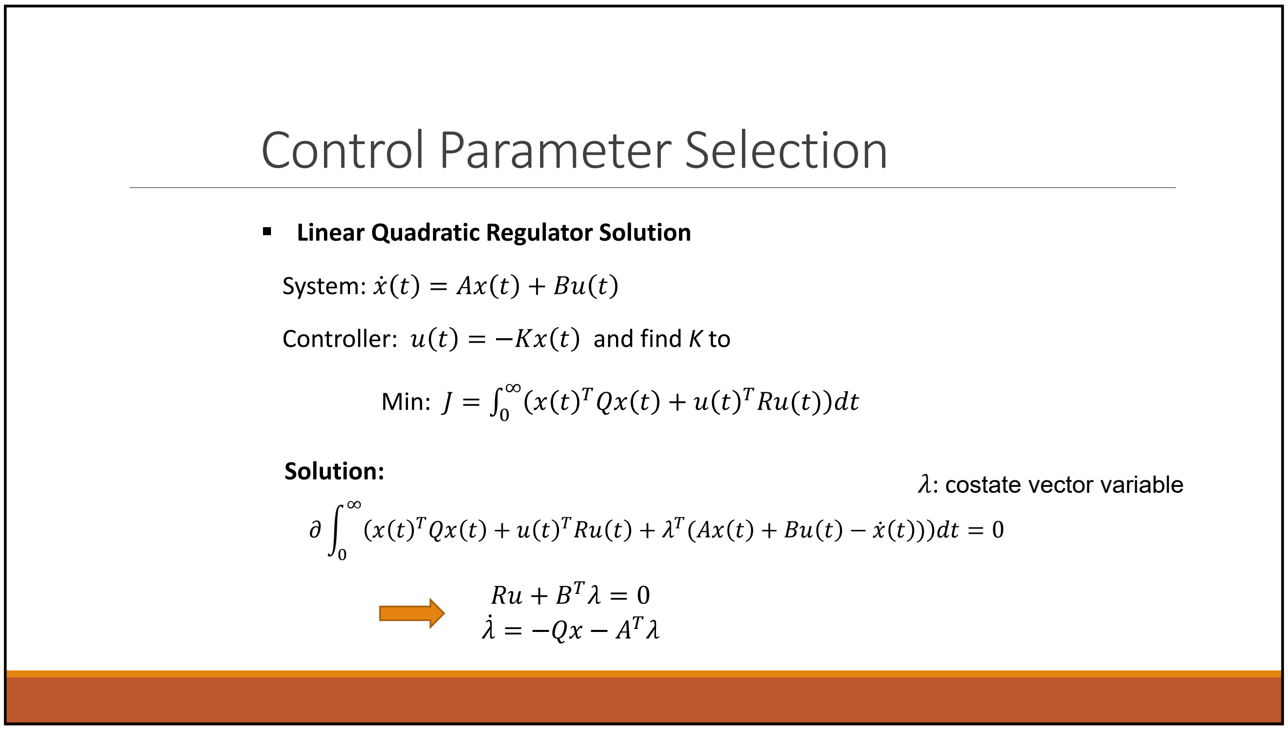
LQR problem: Find K to minimize the following quadratic cost function:

$$\text{Min: } J = \int_0^\infty (x(t)^T Q x(t) + u(t)^T R u(t))dt$$

Q and R are symmetric positive semidefinite matrices

$$Q = Q^T \geq 0 \qquad R = R^T \geq 0$$

Goal: Drive x(t) → 0 as soon as possible with minimal control effort u(t)



Control Parameter Selection

- **Linear Quadratic Regulator Solution**

System: $\dot{x}(t) = Ax(t) + Bu(t)$

Controller: $u(t) = -Kx(t)$ and find K to

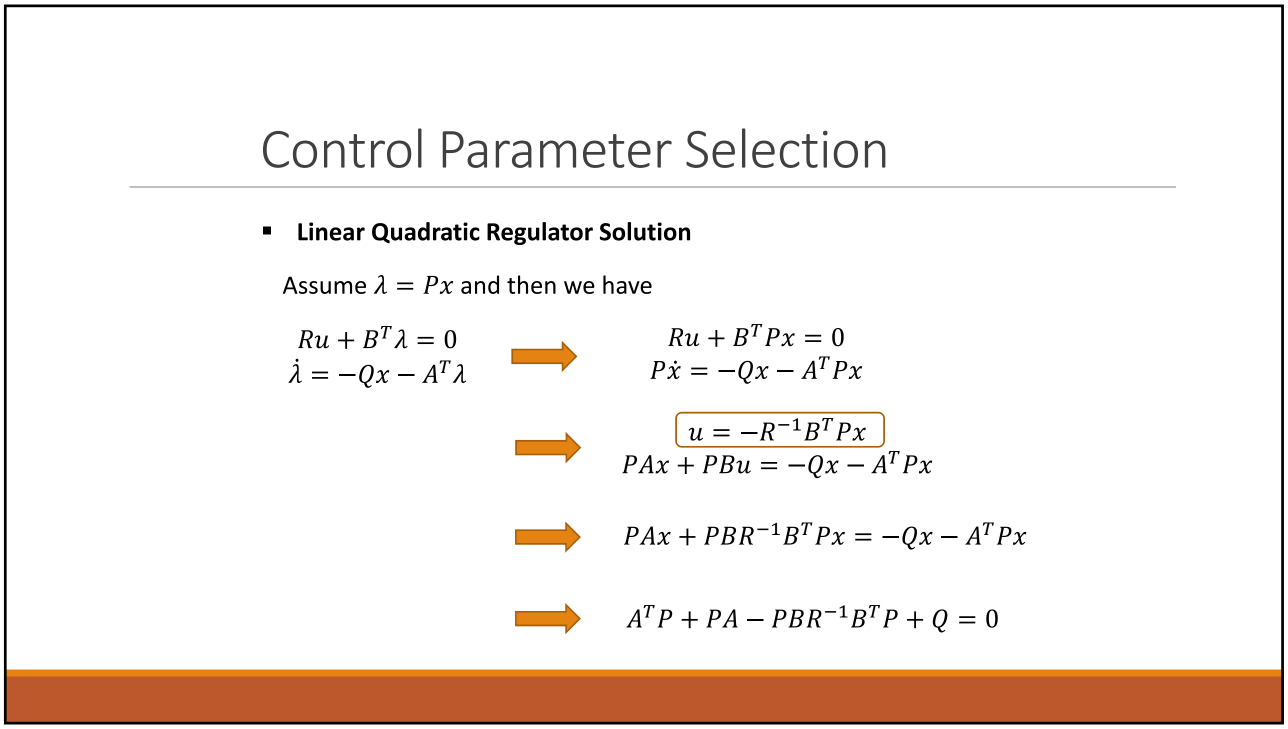
Min: $J = \int_0^\infty (x(t)^T Qx(t) + u(t)^T Ru(t))dt$

Solution:

λ: costate vector variable

$$\partial \int_0^\infty (x(t)^T Qx(t) + u(t)^T Ru(t) + \lambda^T (Ax(t) + Bu(t) - \dot{x}(t)))dt = 0$$

$$\Rightarrow \quad \begin{gathered} Ru + B^T\lambda = 0 \\ \dot{\lambda} = -Qx - A^T\lambda \end{gathered}$$



Control Parameter Selection

- **Linear Quadratic Regulator Solution**

Assume $\lambda = Px$ and then we have

$$\begin{gathered} Ru + B^T\lambda = 0 \\ \dot{\lambda} = -Qx - A^T\lambda \end{gathered} \quad \Rightarrow \quad \begin{gathered} Ru + B^TPx = 0 \\ P\dot{x} = -Qx - A^TPx \end{gathered}$$

$$\Rightarrow \quad \begin{gathered} \boxed{u = -R^{-1}B^TPx} \\ PAx + PBu = -Qx - A^TPx \end{gathered}$$

$$\Rightarrow \quad PAx + PBR^{-1}B^TPx = -Qx - A^TPx$$

$$\Rightarrow \quad A^TP + PA - PBR^{-1}B^TP + Q = 0$$

Control Parameter Selection

- **Linear Quadratic Regulator Solution**

System: $\dot{x}(t) = Ax(t) + Bu(t)$

Controller: $u(t) = -Kx(t)$ and find K to

$$\text{Min: } J = \int_0^\infty (x(t)^T Q x(t) + u(t)^T R u(t)) dt$$

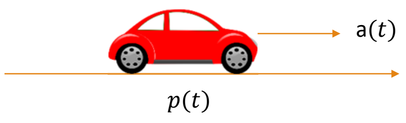
Solution:

$$K = R^{-1} B^T P$$

where $P = P^T \geq 0$ is the symmetric positive semidefinite solution of the following Algebraic Riccati Equation (ARE)

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

LQR Example



Vehicle Position Control by Acceleration

System model: $\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \quad u(t) = a(t),\ x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} p(t) \\ v(t) \end{bmatrix}$

Problem: find controller $u(t) = -Kx(t)$ to minimize

$$\text{Min: } J = \int_0^\infty (x_1(t)^T 1 x_1(t) + u(t)^T 4 u(t)) dt$$

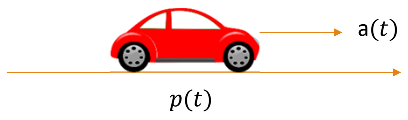
$$\text{Min: } J = \int_0^\infty ({x_1}^T 1 x_1 + {x_1}^T 0 x_2 + {x_2}^T 0 x_1 + {x_2}^T 0 x_2 + u(t)^T 4 u(t)) dt$$

$$\text{Min: } J = \int_0^\infty \left(x(t)^T \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}}_{Q} x(t) + u(t)^T \underbrace{4}_{R} u(t) \right) dt$$

Solution: $u(t) = -R^{-1} B^T P x(t)$, where $P = \begin{bmatrix} p_1 & p_2 \\ p_2 & p_3 \end{bmatrix}$ is from

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

LQR Example



$$\begin{bmatrix}0 & 0\\1 & 0\end{bmatrix}\begin{bmatrix}p_1 & p_2\\p_2 & p_3\end{bmatrix}+\begin{bmatrix}p_1 & p_2\\p_2 & p_3\end{bmatrix}\begin{bmatrix}0 & 1\\0 & 0\end{bmatrix}-\begin{bmatrix}p_1 & p_2\\p_2 & p_3\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\frac{1}{4}\begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}p_1 & p_2\\p_2 & p_3\end{bmatrix}+\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}=0$$

$$\begin{bmatrix}\frac{1}{4}p_2^2-1 & p_1-\frac{1}{4}p_2p_3\\ p_1-\frac{1}{4}p_2p_3 & 2p_2-\frac{1}{4}p_3^2\end{bmatrix}=0=\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$$

$$\frac{1}{4}p_2^2-1=0 \qquad p_1-\frac{1}{4}p_2p_3=0 \qquad 2p_2-\frac{1}{4}p_3^2=0$$

$$\Rightarrow \quad p_1=2,\ p_2=2, p_3=4 \quad \Rightarrow \quad P=\begin{bmatrix}p_1 & p_2\\p_2 & p_3\end{bmatrix}=\begin{bmatrix}2 & 2\\2 & 4\end{bmatrix}$$

$$\Rightarrow \quad u(t)=-R^{-1}B^TPx(t)=-\frac{1}{4}\begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}2 & 2\\2 & 4\end{bmatrix}\begin{bmatrix}p(t)\\v(t)\end{bmatrix}$$
$$=-0.5\,p(t)-v(t)$$

Linear Quadratic Regulator

- Problem Statement
- Controller Design
- Control Parameter Region Analysis
- Control Parameter Selection (Linear Quadratic Regulator)
- Practical Control Implementation (Model Predictive Control)

Model Predictive Control (MPC)

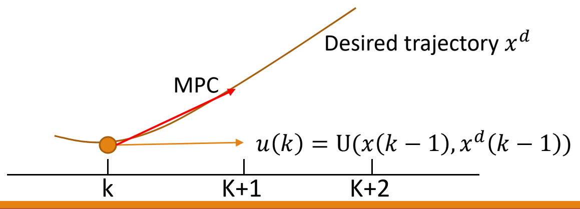
When implementing a controller on digital computers, discrete system model and control are applied.

System: $x(k) = Ax(k-1) + Bu(k)$

To make $x(k)$ track the desired trajectory $x^d(k)$

Controller: $u(k) = \mathrm{U}(x(k-1), x^d(k-1))$

The controller may be perfect at time k, will it be good during (k, k+1)?



Model Predictive Control (MPC)

MPC General Form

For system $x(k) = Ax(k-1) + Bu(k)$, to make $x(k)$ track the desired trajectory $x^d(k)$ with minimal control effort, for each step k, *u(k)* is found by

$$\underset{u(k),\dots,u(k+N)}{Min} \sum_{j=k}^{k+N} \left(\left(x^d(j) - x(j)\right)^T Q \left(x^d(j) - x(j)\right) + u(j)^T R u(j) \right)$$

Subject to: $u(k) \in b_u, x(k) \in b_x,$

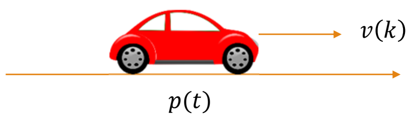
A Special One-Step Form

For system $x(k) = Ax(k-1) + Bu(k)$, to make $x(k)$ track the desired trajectory $x^d(k)$ with minimal control effort, for each step k, *u(k)* is found by

$$\underset{u(k)}{Min} \left(\left(x^d(k) - x(k)\right)^T Q \left(x^d(k) - x(k)\right) + u(k)^T R u(k) \right)$$

Subject to: $u(k) \leq b_u, x(k) \leq b_x$

MPC Example



Vehicle Position Control by Speed

System model $x(k) = x(k-1) + u(k)$ $\quad u(k) = v(t), x(t) = p(t)$

To make $x(k)$ track the desired trajectory $x^d(k)$ with minimal control effort, for each step k, *u(k)* is found by

$$\underset{u(k),\dots,u(k+N)}{Min} \quad \sum_{j=k}^{k+N} \left(\left(x^d(k) - x(k-1)\right)^T \left(x^d(k) - x(k)\right) + u(k)^2 \right)$$

Subject to: $u(k) \leq v_{max}$

For each time step, use searching algorithm to search the best $u(k), \dots, u(k+N)$ in the range of [0, v_{max}] to make the cost function above minimal, and then only apply u(k) at this time step.

LQR Vs. MPC

Linear Quadratic Regulator:

- Find the controller u(t) by offline computation before implementation. Once the controller is found, it won't change during implementation.
- Simple calculation and implementation; May have relatively more errors.

Model Predictive Control:

- Find the controller u(t) by online searching in every control step during the implementation. The controller is changing in every step during the implementation.
- Complicated calculation and implementation; Relatively smaller errors.

Controls

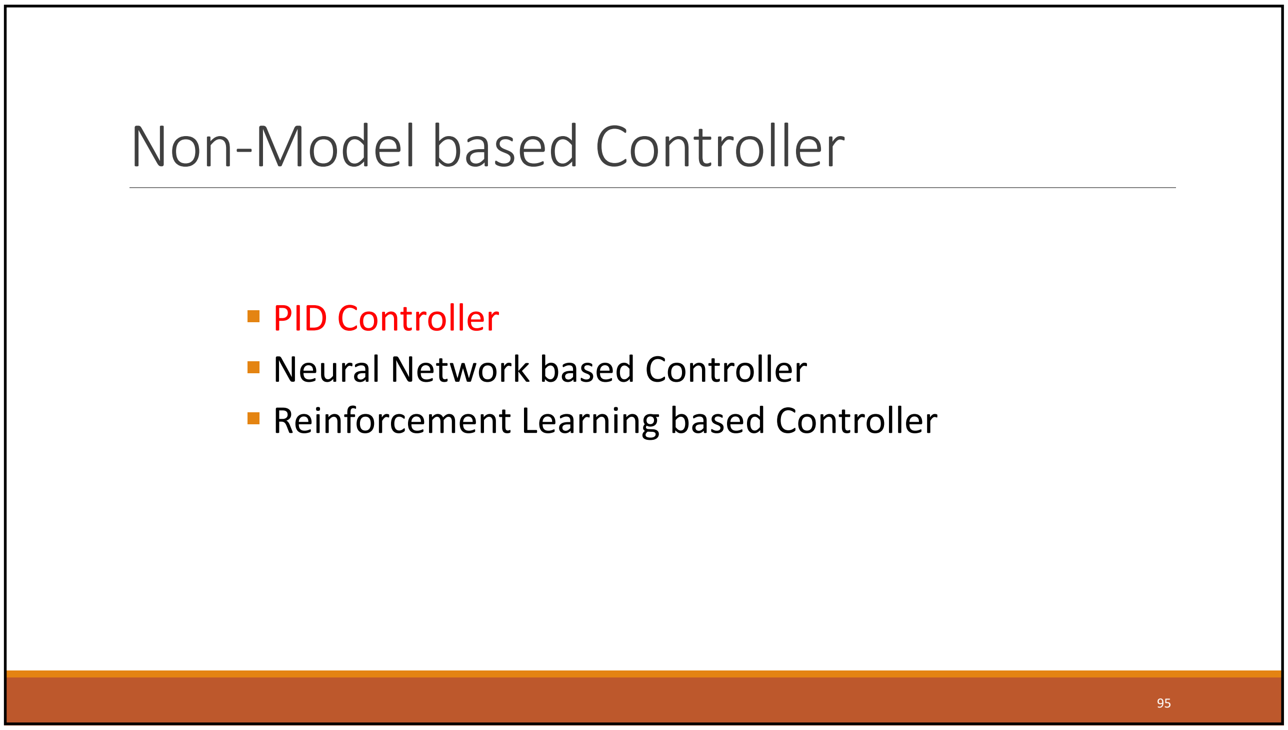
1. Introduction of Controls
2. System Modeling and Analysis
3. System Controller Designs
   - Problem Statement
   - Stability
   - Model based Controller Design: Full-State Feedback Controller
   - Non-Model based Controller

93

Non-Model based Controller

- PID Controller
- Neural Network based Controller
- Reinforcement Learning based Controller

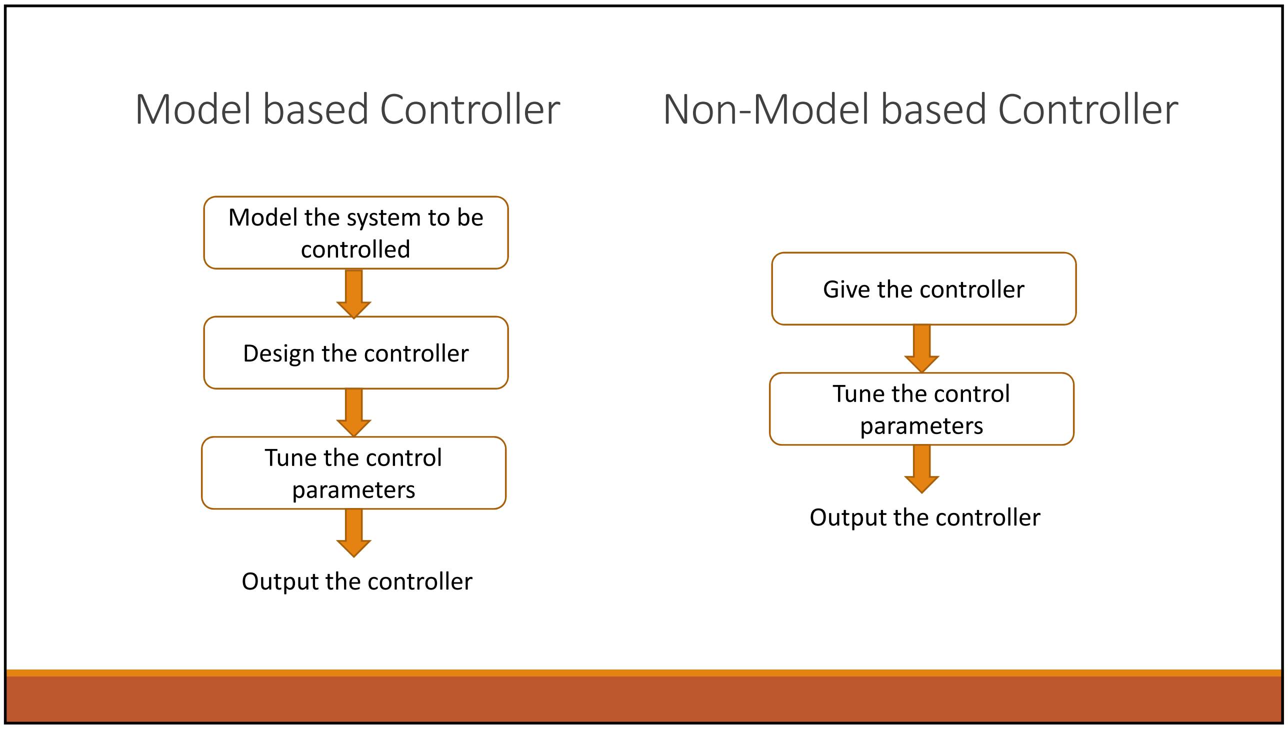
94



Non-Model based Controller

- PID Controller
- Neural Network based Controller
- Reinforcement Learning based Controller

95

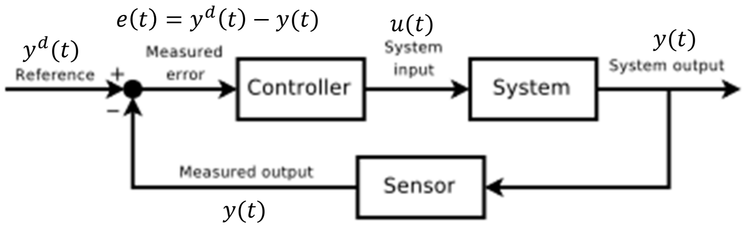


Model based Controller

Non-Model based Controller

Model the system to be controlled
Design the controller
Tune the control parameters
Output the controller
Give the controller
Tune the control parameters
Output the controller

PID Controller



Proportional–Integral–Derivative controller (PID controller)

- Do not need to model the system relationship between u(t) and y(t)
- Compute the system input u(t) based on error e(t) directly

PID Controller

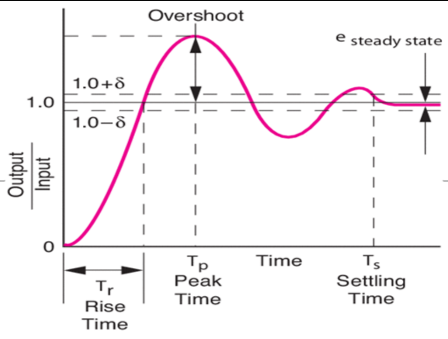
Proportional–Integral–Derivative controller (PID controller)

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de(t)}{dt}$$

where

$e(t) = y^d(t) - y(t)$: control error
k_p: Proportional gain, reduce current error e(t)
k_i: Integral gain, reduce total errors (steady-state error)
k_d: Derivative gain, reduce overshoot and oscillations

PID Controller



CL RESPONSE	**RISE TIME**	**OVERSHOOT**	**SETTLING TIME**	**S-S ERROR**
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Change	Decrease	Decrease	No Change

- The use of the PID control does not guarantee optimal control of the system or even its stability.
- The key challenge is how to tune the three PID controller gains

PID Controller

Tune PID Controller

There are many approaches available. All are heuristic approaches.
(e.g., https://en.wikipedia.org/wiki/PID_controller#Manual_tuning)

Example 1: Ziegler–Nichols method

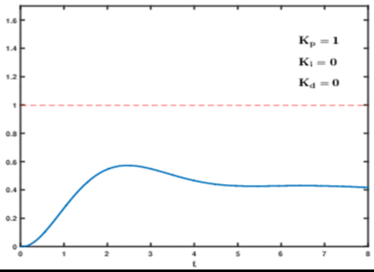
1) Set k_p, k_i, k_d to 0
2) Increase k_p until k_u at which the output oscillates consistently with a period T_u
3) Set $k_p = 0.6k_u$, $k_i = T_u/2$, $k_d = T_u/8$

PID Controller

Tune PID Controller

Example 2: Manual turning

1) Set k_p, k_i, k_d to 0
2) Increase k_p until k_u at which the output oscillates consistently. Set $k_p = 0.6k_u$
3) Increase k_i until any steady-state error is small enough
4) Increase k_d until the oscillations disappear



Implementation of PID Controller

Proportional–Integral–Derivative controller (PID controller)

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de(t)}{dt}$$

How to implement this using a digital computer?

Digital/discrete PID controller

$$u(k) = k_p e(k) + k_i \sum_{j=1}^{k} e(j) + k_d \frac{e(k)-e(k-1)}{T}$$

$$u(k) = k_p e(k) + k_i \sum_{j=1}^{k} e(j) + k_d (e(k) - e(k-1))$$

new $k_d = k_d$/T

Implementation of PID Controller

Digital/discrete PID controller

$$\mathrm{u}(k) = k_p e(k) + k_i \textstyle\sum_{j=1}^{k} e(j) + k_d(e(k) - e(k-1))$$

$$\mathrm{u}(k-1) = k_p e(k-1) + k_i \textstyle\sum_{j=1}^{k-1} e(j) + k_d(e(k-1) - e(k-2))$$

$$\Downarrow$$

$$\mathrm{u}(k) - u(k-1) = k_p e(k) - k_p e(k-1) + k_i e(k)$$
$$+k_d e(k) - 2k_d e(k-1) + k_d e(k-2)$$

$$\Downarrow$$

$$\mathrm{u}(k) = u(k-1) + \underbrace{\boxed{(k_p + k_i + k_d)}}_{k_1} e(k) + \underbrace{\boxed{(-k_p - 2k_d)}}_{k_2} e(k-1)$$
$$+ \underbrace{\boxed{k_d}}_{k_3} e(k-2)$$

Implementation of PID Controller

Digital/discrete PID controller

$$\mathrm{u}(k) = u(k-1) + k_1 e(k) + k_2 e(k-1) + k_3 e(k-2)$$

where

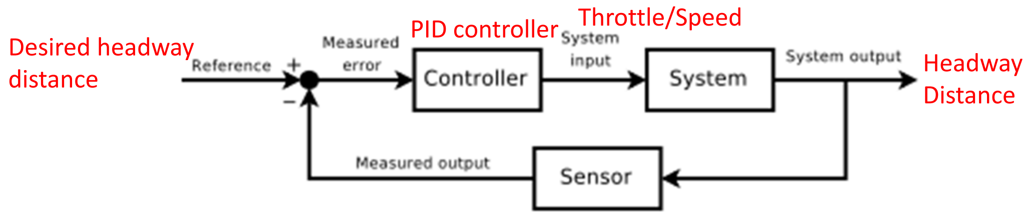
$$k_1 = k_p + k_i + k_d$$
$$k_2 = -k_p - 2k_d$$
$$k_3 = k_p$$

Implementation of PID Controller

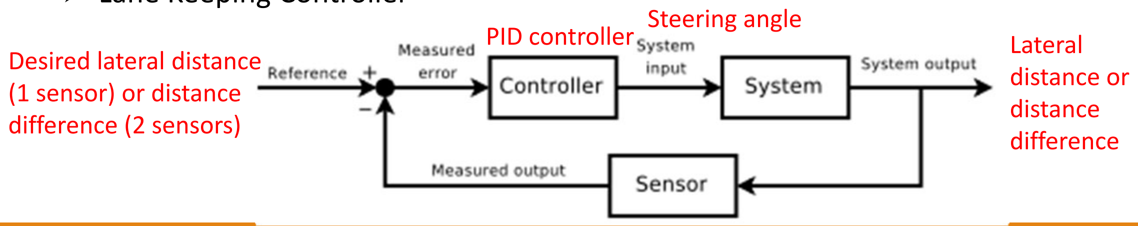
```
float e = 0, e1 = 0, e2 = 0;
float u = 0;
float kp = 0, ki =0, kd = 0;  // Define PID parameters
float k1 = kp + ki + kd;
float k2 = -kp - 2*kd;
float k3 = kd;

void loop() {
  e2 = e1;
  e1 = e;
  e = f(......); // Compute current control error
  u = u + k1*e + k2*e1 + k3*e2;
  //Implement u as control input
  ............
}
```

PID Controller Design

- Adaptive Cruise Controller



- Lane Keeping Controller



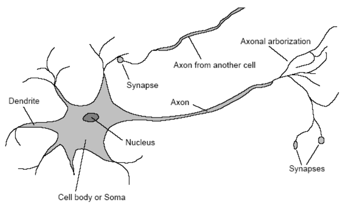
Non-Model based Controller

- PID Controller
- Neural Network based Controller
- Reinforcement Learning based Controller

(Artificial) Neural Networks

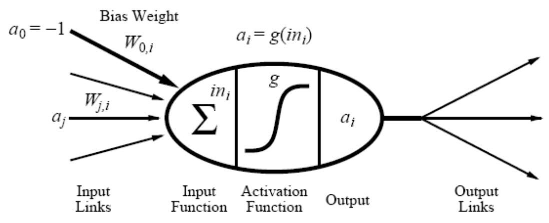
Motivation: human brain

- massively parallel (*10^{11} neurons, ~20 types*)
- small computational units with simple low-bandwidth communication (*10^{14} synapses, 1-10ms cycle time*)



Realization: neural network

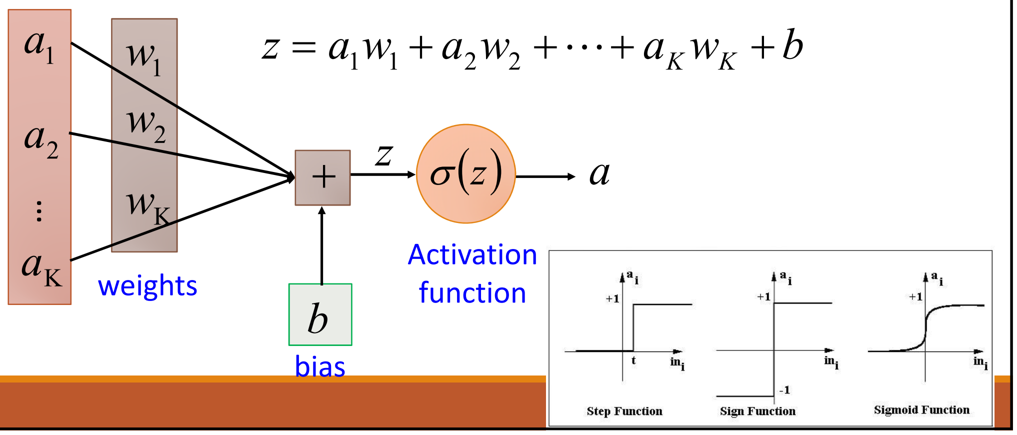
- *units* (≈ neurons) connected by *directed weighted links*
- *activation function* from inputs to outpu



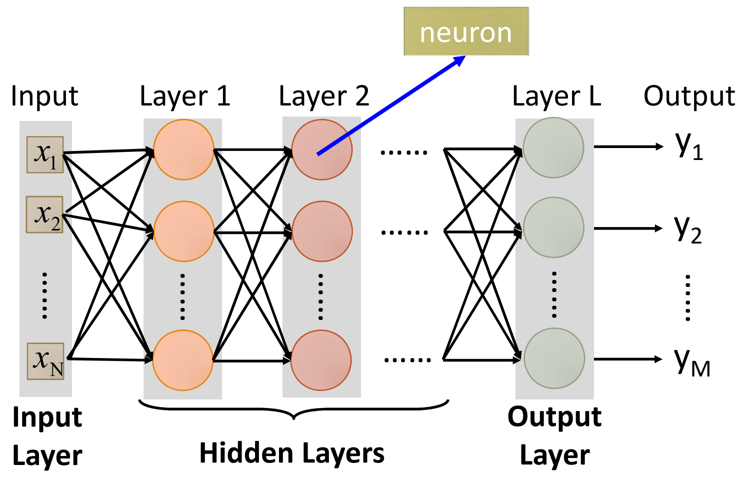
Element of Neural Network

Neuron $f: R^K \rightarrow R$

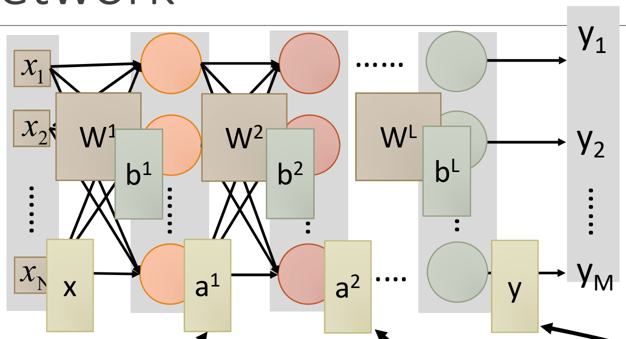
$$z = a_1 w_1 + a_2 w_2 + \cdots + a_K w_K + b$$



Neural Network



Neural Network

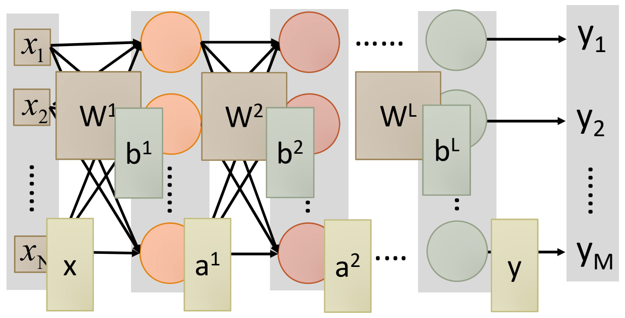


$$\sigma(W^1 x + b^1) \rightarrow a^1$$

$$\sigma(W^2 a^1 + b^2) \rightarrow a^2$$

$$\sigma(W^L a^{L-1} + b^L) \rightarrow y$$

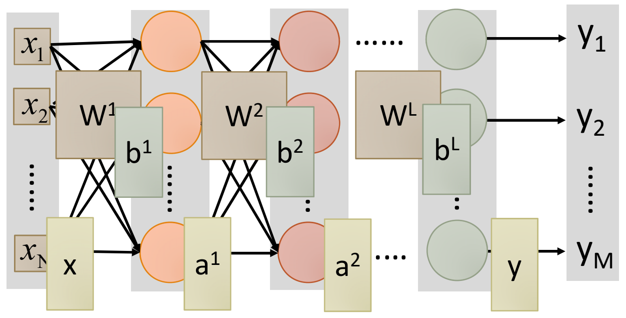
Neural Network



$$y = f(x)$$

$$= \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

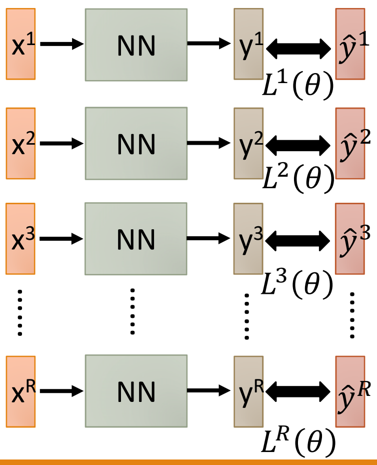
How to set neural network parameters



NN Parameters: $\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$

How to set the network parameters θ?

Neural Network Training



x^1 NN y^1 $\hat{y}^1$ $L^1(\theta)$
x^2 NN y^2 $\hat{y}^2$ $L^2(\theta)$
x^3 NN y^3 $\hat{y}^3$ $L^3(\theta)$
x^R NN y^R $\hat{y}^R$ $L^R(\theta)$

Total Training Cost:

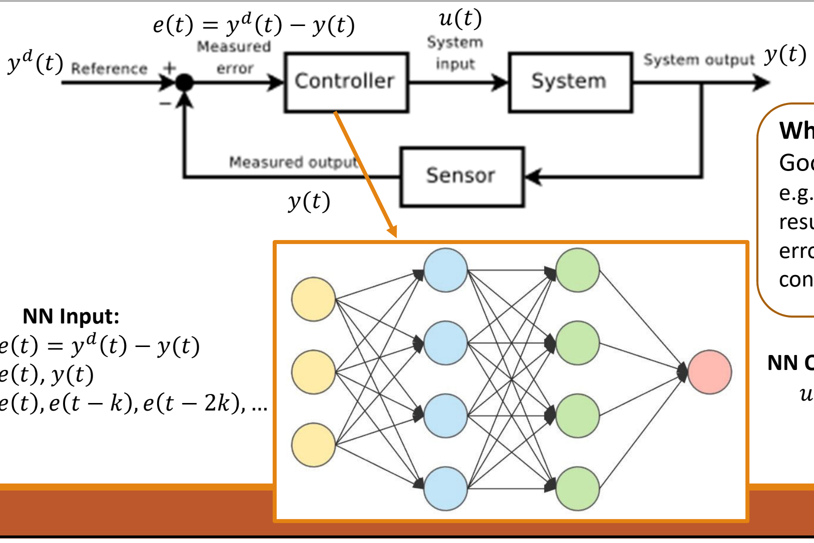
$$C(\theta) = \sum_{r=1}^{R} L^r(\theta)$$

Evaluate how good/bad the network parameters (weights) θ is on this task.

Optimization:
Find the neural network parameters θ^* that can minimize the cost $C(\theta)$

Training approach:
- *Perceptron training*
- *Linear programming*
- *Delta rule*
- *Backpropagation*
- ……

Neural Network based Control



What data for training?
Good data only!
e.g., model-based control results with small/decreasing errors or human-annotated control data

NN Input:
- Option 1: $e(t) = y^d(t) - y(t)$
- Option 2: $e(t), y(t)$
- Option 3: $e(t), e(t-k), e(t-2k), \dots$
- …….

NN Output:
$u(t)$

Non-Model based Controller

- PID Controller
- Neural Network based Controller
- Reinforcement Learning based Controller

116

Reinforcement Learning

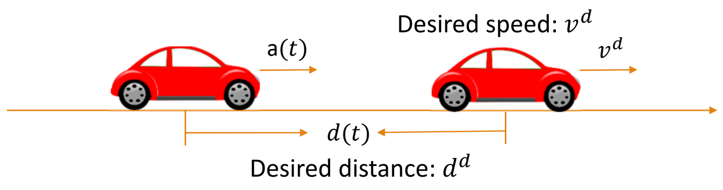
- What is Reinforcement Learning?

Definition: Reinforcement learning (RL) is an area of machine learning concerned with how an agent takes actions in an environment in order to maximize the cumulative reward.

Intuitive Description: “Learn how an agent senses and optimally acts in its environment to achieve its goal”

Reinforcement Learning

- Reinforcement Learning Example



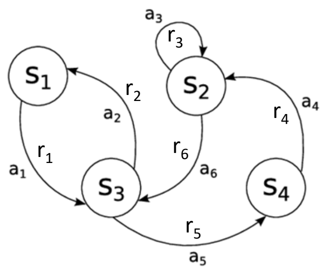
The vehicle has sensors to observe the environment and actuators to perform actions. The RL task is to learn a control policy that can drive the vehicle to achieve its goal.

Reinforcement Learning

- Markov Decision Process (MDP)

$$\mathrm{M} = (\mathrm{S}, \mathrm{A}, \mathrm{T}, \mathrm{R})$$

- **S:** is a set of states called the state space
  One state is represented by $s \in S$
- **A:** is a set of actions called the action space
  One action is represented by $a \in A$
- **T:** is the transition of a state s to s' after applying an action a
  One transition is represented by $\delta(s, a) = s' \in T$
- **R:** is the reward of a transition
  One reward is represented by $r(s, a) \in R$

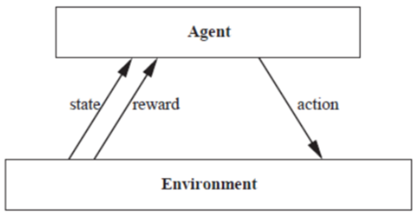


Reinforcement Learning

- Markov Decision Process (MDP)

MDP in Reinforcement Learning:

- The agent can perceive a set of states S and has a set of actions A to perform
- At each time step t, the agent senses the current state s_t, chooses a current action a_t and performs it
- The environment responds by returning a reward $r_t = r(s_t, a_t)$ and changing the state to $s_{t+1} = \delta(s_t, a_t)$



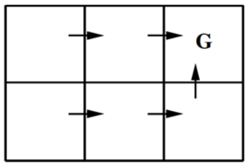
Reinforcement Learning

- Reinforcement Learning Task

RL Goal: Learn a policy π to make an agent sense and choose an optimal action to achieve its goal.

Policy π: $S \rightarrow A$

At state s, what action a to take?



How to know if a policy π is good or not? (How to evaluate it?)

Discounted cumulative reward $V^{\pi}(s_t), s_t \in S$

Reinforcement Learning

- Discounted cumulative reward

Given a policy π, the transition starting from state s_t will be

$$\begin{tikzcd}
s_t \arrow[r, "a_t"] \arrow[d] & s_{t+1} \arrow[r, "a_{t+1}"] \arrow[d] & s_{t+2} \arrow[r, "a_{t+2}"] \arrow[d] & s_{t+3} \arrow[r] & \dots\dots \\
r_t & r_{t+1} & r_{t+2} & &
\end{tikzcd}$$

The discounted cumulative reward for state s_t after applying policy π is calculated by

$$V^{\pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

$$= \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

$0 \leq \gamma \leq 1$ is a discounted factor

(Intuitively, this indicates how far the state s_t is from the goal under policy π)

Reinforcement Learning

- Reinforcement Learning Formulation

RL Goal: Learn a policy π to make an agent sense and choose an optimal action to achieve its goal.

Policy π: $S \rightarrow A$

Since the policy π is evaluated by the discounted cumulative reward, finding the best policy π^* is equivalent to finding a policy which can maximize the discounted cumulative reward for any state

$$\pi^*(s) = \underset{\pi}{argmax}\ V^\pi(s)\,, \qquad s \in S$$

Note: The discounted cumulative reward for state s after applying best policy π^* will be represented by

$$V^*(s) = \underset{\pi}{max}\ V^\pi(s)$$

Reinforcement Learning

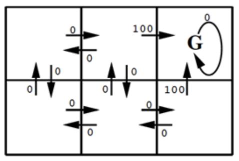
- Reinforcement Learning Formulation

$$\begin{aligned}\pi^*(s_t) &= \underset{\pi}{argmax}\ V^\pi(s_t)\\ &= \underset{\pi}{argmax}\{r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots\}\\ &= \underset{\pi}{argmax}\{r_t + \gamma(r_{t+1} + \gamma r_{t+2} + \cdots)\}\\ &= \underset{a}{argmax}\{r_t + \gamma \underset{\pi}{max}\{r_{t+1} + \gamma r_{t+2} + \cdots\}\}\\ &= \underset{a}{argmax}\{r_t + \gamma V^*(s_{t+1})\}\end{aligned}$$

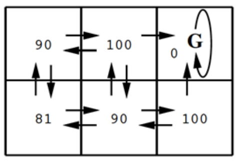
$$\Rightarrow \boxed{\pi^*(s) = \underset{\pi}{argmax}\ V^\pi(s) = \underset{a}{argmax}\ \{\boxed{r(s,a)} + \gamma \boxed{V^*(\delta(s,a))}\}}$$

Reinforcement Learning

- Reinforcement Learning Example

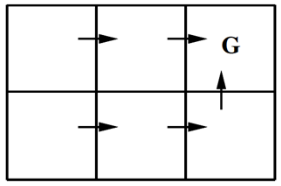


$r(s,a)$ (immediate reward)



$V^*(s)$ values

$$\pi^*(s) = \underset{\pi}{argmax}\ V^\pi(s) = \underset{a}{argmax}\ \{r(s,a) + \gamma V^*(\delta(s,a))\}$$



One optimal policy

Reinforcement Learning

- Reinforcement Learning Approaches
  - ❑ Model-based Approach: Learn the policy with learning the entire model

$$\pi^*(s) = \underset{\pi}{argmax}\ V^\pi(s) = \underset{a}{argmax}\ \{\boxed{r(s,a)} + \gamma \boxed{V^*(\delta(s,a))}\}$$

  - ❑ Non-Model based Approach: Learn the policy without learning the model
    - Q-learning (most widely used reinforcement learning approach)

Reinforcement Learning

- Q-Learning

$$\pi^*(s) = \underset{\pi}{argmax}\ V^{\pi}(s) = \underset{a}{argmax}\ \{\underbrace{r(s,a) + \gamma V^*(\delta(s,a))}_{}\}^{Q(s,a)}$$

$$\Rightarrow \quad \pi^*(s) = \underset{a}{argmax}\ Q(s,a)$$

where $Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a))$

Instead of learning $r(s,a)$ and $V^*(\delta(s,a))$, directely learn $Q(s,a)$!

Reinforcement Learning

- Q-Learning

Key Idea: Iterative learning of $Q(s,a)$

- When we have an estimate of $Q(s,a)$, then we have

$$V^*(s) = \underset{\pi}{max}\ V^{\pi}(s) = \underset{a}{max}\ \{r(s,a) + \gamma V^*(\delta(s,a))\} = \underset{a}{max}\ Q(s,a)$$

- When the agent applies an action a at state s, the state changes to s' and it returns an reward $r(s,a)$, and we can re-estimate $Q(s,a)$ by

$$Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a)) = r(s,a) + \gamma \underset{a}{max}\ Q(s',a)$$

Reinforcement Learning

- Q-Learning Algorithm

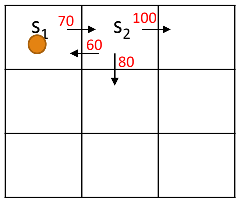
Initialization: For each (s, a) pair, give an initial estimate of Q-table: $Q(s, a)$, e.g., 0.

- At current state s, apply an action a
- Observe the transited state s' after applying action a
- Receive or calculate an award r
- Update the Q table by
$$Q(s, a) = r + \gamma \max_a Q(s', a)$$
- $s \leftarrow s'$

After Q-learning: $a^* = \pi^*(s) = \underset{a}{\operatorname{argmax}} Q(s, a)$

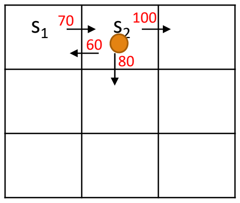
Reinforcement Learning

- Q-Learning Example



The agent takes an action a_{right}

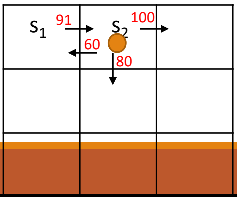
Return an award r =1



Update Q table

$$Q(s, a) = r + \gamma \max_a Q(s', a)$$

$$Q(s_1, a_{right}) = r(s_1, a_{right}) + \gamma \max_a Q(s_2, a)$$
$$= 1 + 0.9 * \max\{60, 80, 100\}$$
$$= 91$$



- The agent is at state s_1
- Arrows mean moving to different directions
- Numbers mean current Q values

One Implementation Example of Reinforcement Learning based Adaptive Cruise Control

Reinforcement Learning

- Q-Learning Algorithm

Initialization: For each (s, a) pair, give an initial estimate of Q-table: $Q(s, a)$, e.g., 0.

- At current state s, apply an action a
- Observe the transited state s' after applying action a
- Receive or calculate a reward r
- Update the Q table by
$$Q(s, a) = r + \gamma \max_{a} Q(s', a)$$
- $s \leftarrow s'$

After Q-learning: $a^* = \pi^*(s) = \underset{a}{argmax}\ Q(s, a)$

Reinforcement Learning

- **Initialization:**

For each (s, a) pair, give an initial estimate of Q-table: $Q(s, a)$, e.g., 0.

What is s?
[speed error, distance error]
→ s = [ES, ED]

What is a?
Acceleration
→ a = A

What Q-table $Q(s, a)$?
A table: [speed error, distance error, acceleration] = Q value
→ Q [ES, ED, A] = Q value

Implementation

Discretization: Convert continuous states to discrete states (as indices of Q):

ES = $f_{discrete}$(speed error), ES ∈ $\{1,2, \dots \dots\}$
ED = $f_{discrete}$(distance error), ED ∈ $\{1,2, \dots \dots\}$
A = $f_{discrete}$(acceleration), A ∈ $\{1,2, \dots \dots\}$

Reinforcement Learning

- Q-Learning Algorithm
  - **Training:**

- At current state s, apply an action a
- Observe the transited state s' after applying action a
- Receive or calculate a reward r
- Update the Q table by

$$Q(s, a) = r + \gamma \max_a Q(s', a)$$

- $s \leftarrow s'$

What is r?
f_r(speed error change, distance error change)

What data to use?
All data from model-based control

Reinforcement Learning

- Q-Learning Algorithm
  - **Control:**
    - Load Q table;
    - Find current s
    - Find a in the Q table by $a^* = \underset{a}{argmax}\ Q(s,a)$

```
Implementation
Load Q;
Calculate current speed error and distance error;
ES  = f_discrete(speed error);
ED  = f_discrete(distance error);
A = find(Q(ES, ED, :)==max(Q(ES, ED, :))); % Find A to maximize Q(ES, ED, :)
acceleration  =  f^-1_discrete(A)  % Inverse of discretization function
```

Questions?