# Ve460 Control Systems Analysis and Design
# Chapter 4 Modeling of Physical Systems

Jun Zhang

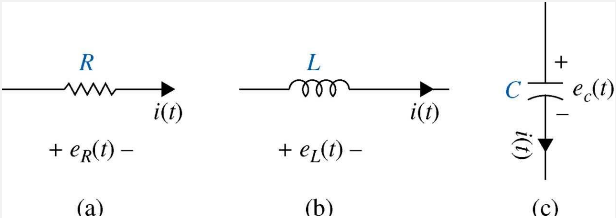
Shanghai Jiao Tong University

## Chapter 4 Modeling of Physical Systems

Two common methods:

| | | |
|---|---|---|
| TF | $\rightarrow$ | LTI |
| State Space | $\rightarrow$ | Linear and Nonlinear |

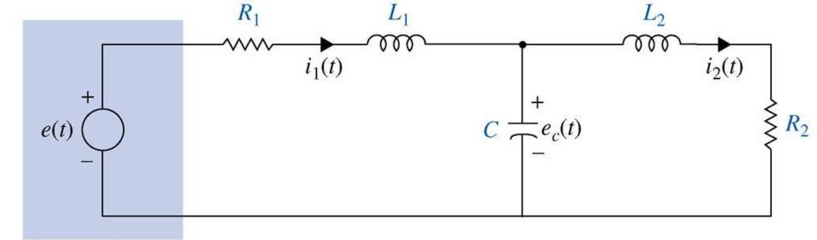
## 4-2 Electrical Networks




Resistor: $V = R \cdot I$ Inductor: $V = L\dfrac{di}{dt}$, $V(s) = sL \cdot I(s)$

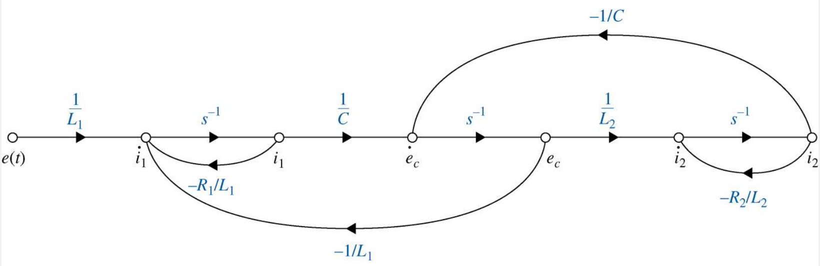
Capacitor: $i = C\dfrac{dV}{dt}$, $V(s) = \dfrac{1}{sC} \cdot I(s)$

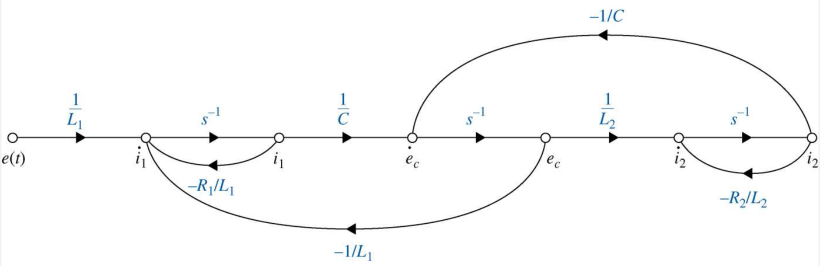
## Example




$$
\begin{cases}
L_1 \dfrac{di_1}{dt} = -R_1 i_1 - e_c + e \\
L_2 \dfrac{di_2}{dt} = -R_2 i_2 + e_c \\
C \dfrac{de_c}{dt} = i_1 - i_2
\end{cases}
$$

$$\Rightarrow \quad \frac{d}{dt}\begin{bmatrix} i_1 \\ i_2 \\ e_c \end{bmatrix} = \begin{bmatrix} -\dfrac{R_1}{L_1} & 0 & -\dfrac{1}{L_1} \\ 0 & -\dfrac{R_2}{L_2} & \dfrac{1}{L_2} \\ \dfrac{1}{C} & -\dfrac{1}{C} & 0 \end{bmatrix}\begin{bmatrix} i_1 \\ i_2 \\ e_c \end{bmatrix} + \begin{bmatrix} \dfrac{1}{L_1} \\ 0 \\ 0 \end{bmatrix} e$$






$$
\begin{aligned}
\Delta = 1 - \Big(&-\frac{1}{s}\frac{R_1}{L_1} - \frac{1}{s}\frac{R_2}{L_2} - \frac{1}{s}\frac{1}{C}\frac{1}{s}\frac{1}{L_1} - \frac{1}{s}\frac{1}{L_2}\frac{1}{s}\frac{1}{C}\Big) \\
&+ \Big(-\frac{1}{s}\frac{R_1}{L_1}\Big)\cdot\Big(-\frac{1}{s}\frac{R_2}{L_2}\Big) + \Big(-\frac{1}{s}\frac{R_1}{L_1}\Big)\cdot\Big(-\frac{1}{s}\frac{1}{L_2}\frac{1}{s}\frac{1}{C}\Big) \\
&+ \Big(-\frac{1}{s}\frac{1}{C}\frac{1}{s}\frac{1}{L_1}\Big)\cdot\Big(-\frac{1}{s}\frac{R_2}{L_2}\Big).
\end{aligned}
$$

Therefore

$$\frac{I_1(s)}{E(s)} = \frac{L_2Cs^2 + R_2Cs + 1}{D},$$

where

$$D = L_1L_2Cs^3 + (R_1L_2 + R_2L_1)Cs^2 + (L_1 + L_2 + R_1R_2C)s + R_1 + R_2.$$

Similarly,

$$\frac{I_2(s)}{E(s)} = \frac{1}{D},$$

and

$$\frac{E_c(s)}{E(s)} = \frac{L_2s + R_2}{D}.$$

## Using Matlab Symbolic Toolbox

To this end, we first perform Laplace Transform to both sides of the dynamical equation:

$$\frac{d}{dt}\underbrace{\begin{bmatrix} i_1 \\ i_2 \\ e_c \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} -\frac{R_1}{L_1} & 0 & -\frac{1}{L_1} \\ 0 & -\frac{R_2}{L_2} & \frac{1}{L_2} \\ \frac{1}{C} & -\frac{1}{C} & 0 \end{bmatrix}}_{A}\underbrace{\begin{bmatrix} i_1 \\ i_2 \\ e_c \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} \frac{1}{L_1} \\ 0 \\ 0 \end{bmatrix}}_{b} e.$$

Then,

$$\begin{aligned} & sX(s) = AX(s) + bE(s) && \Rightarrow && (sI - A)X(s) = bE(s) \\ \Rightarrow \quad & X(s) = (sI - A)^{-1}bE(s) && \Rightarrow && I_1(s) = c(sI - A)^{-1}bE(s), \end{aligned}$$

where $c = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$.

```
>> syms R1 L1 R2 L2 C s;
>> A=[-R1/L1 0 -1/L1; 0 -R2/L2 1/L2; 1/C -1/C 0];
>> b=[1/L1; 0; 0];
>> c=[1 0 0];
>> simplify( c*inv(s*eye(3)-A)*b )
ans =
  (C*L2*s^2+C*R2*s+1)
    /(R1+R2+L1*s+L2*s+C*L1*L2*s^3+C*L1*R2*s^2+C*L2*R1*s^2+C*R1*R2*s)

>> pretty( collect( ans, 's') )
                             2
                       C L2 s  + C R2 s + 1
---------------------------------------------------------------------
         3                        2
C L1 L2 s  + (C L1 R2 + C L2 R1) s  + (L1 + L2 + C R1 R2) s + R1 + R2
```

This is the same as previously derived:

$$\frac{I_1(s)}{E(s)} = \frac{L_2 C s^2 + R_2 C s + 1}{D},$$

where

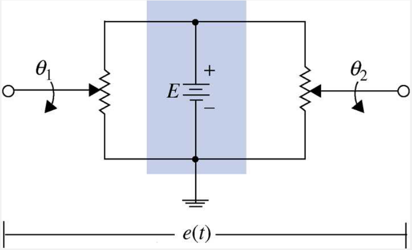
$$D = L_1 L_2 C s^3 + (R_1 L_2 + R_2 L_1) C s^2 + (L_1 + L_2 + R_1 R_2 C)s + R_1 + R_2.$$

# 4-5 Sensors & Encoders in Control System

## 4-5-1 Potentiometer

Input: Linear or rotational displacement

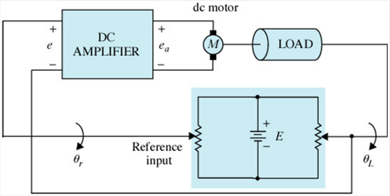
Output: Voltage proportional to input displacement




$$e(t) = K_S(\theta_1(t) - \theta_2(t))$$

$K_S$: constant

Can be used, *e.g.*, in DC motor control system for position feedback.




$\theta_r$: reference input;

$\theta_L$: input;

$e_a$: armature voltage of a DC motor.

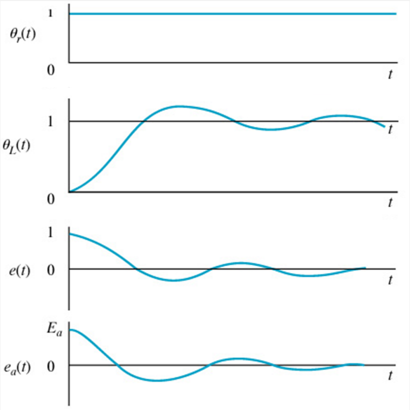


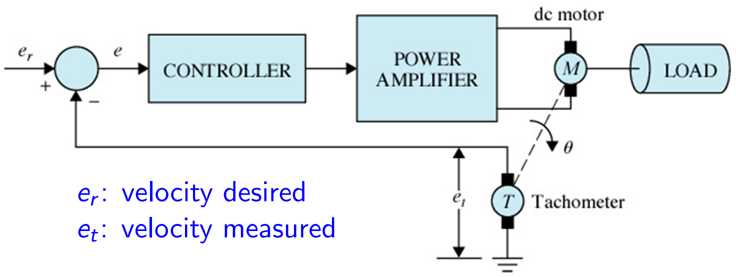

Figure 4.1: Typical waveforms of signals

## 4-5-2 Tachometers

Convert mechanical energy into electrical energy

Output: Voltage proportional to angular velocity input;

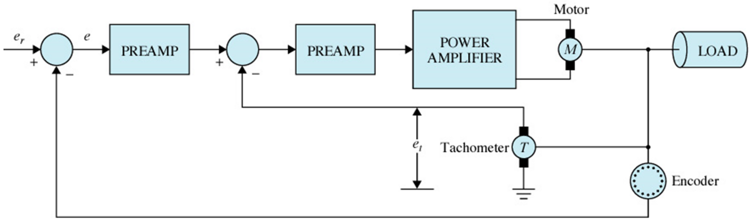
Used as: Velocity indicator




$e_r$: velocity desired

$e_t$: velocity measured

- Velocity Control: accuracy of tachometer is highly critical




- Position Control
  Tachometer: velocity feedback
  → to improve stability or damping, accuracy of tachometer is not that critical.

- Another usage: visual speed readout of a rotating shaft

### Mathematical model

$$e_t(t) = K_t \frac{d\theta(t)}{dt} = K_t \omega(t),$$

where

$e_t(t)$: output voltage,

$\theta(t)$: rotor displacement (in radians),

$\omega(t)$: rotor velocity (in rad/sec),

$K_t$: tachometer constant (in V/(rad/sec)).

$$\Rightarrow \quad \frac{E_t(s)}{\Theta(s)} = K_t s.$$

### 4-5-3 Incremental Encoder

Convert linear / rotary displacement into digitally coded / pulse signals

- Absolute encoder: output a distinct digital code indicating each particular position within the range (does not need knowledge of previous positioning);
- Incremental encoder: cyclical, provides a pulse for each increment.

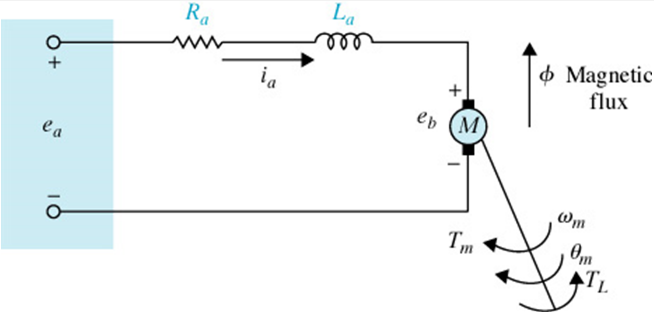
## 4-6 DC Motors – widely used as mover in industry

### Mathematical modeling

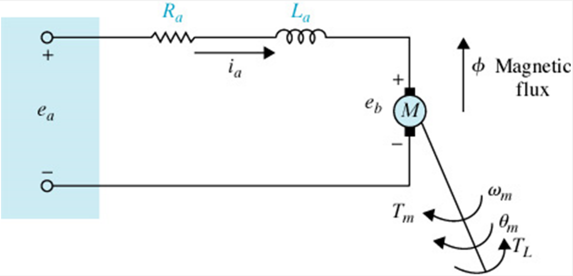
- Convert electric energy into mechanical energy: the torque $T_m$ on the motor shaft $\propto$ field flux $\phi$ and armature current $i_a$:

$$\underbrace{T_m}_{\text{motor torque}} = \underbrace{K_m}_{\text{proportional constant}} \cdot \underbrace{\phi}_{\text{magnetic flux}} \cdot \underbrace{i_a}_{\text{armature current}}$$

- Back emf: when conductor moves, a voltage is generated that opposes the current flow, proportional to shaft velocity.




$i_a(t) =$ armature current
$R_a =$ armature resistance
$e_b(t) =$ back emf
$T_L =$ load torque
$T_m =$ motor torque
$\theta_m(t) =$ rotor displacement
$K_i =$ torque constant
$L_a =$ armature inductance
$e_a =$ applied voltage
$K_b =$ back-emf constant
$\phi =$ magnetic flux in the air gap
$\omega_m(t) =$ rotor angular velocity
$J_m =$ rotor inertia
$B_m =$ viscous-friction coefficient

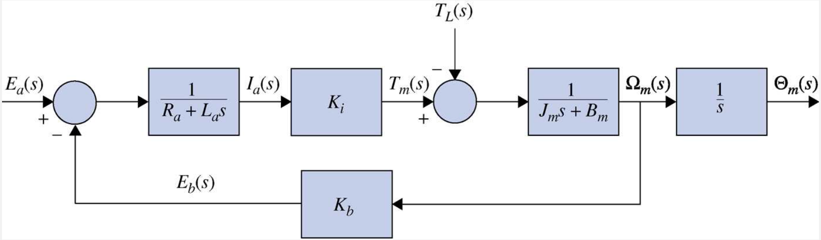



When $\phi$ is constant, $T_m = K_m\phi \cdot i_a = K_i i_a$,

$$\begin{cases} L_a \dfrac{di_a}{dt} = e_a - R_a i_a - e_b, \\ e_b(t) = K_b \dfrac{d\theta_m(t)}{dt} = K_b\omega_m(t), \quad \text{Back emf} \\ T_m = K_i i_a, \\ J_m \dfrac{d^2\theta_m(t)}{dt^2} = T_m(t) - T_L(t) - B_m \dfrac{d\theta_m(t)}{dt} \end{cases}$$

$$\begin{cases} L_a \dfrac{di_a}{dt} = e_a - R_a i_a - K_b \omega_m, \\ J_m \dfrac{d^2\theta_m(t)}{dt^2} = K_i i_a - T_L - B_m \omega_m \end{cases}$$

$$\therefore \frac{d}{dt}\begin{bmatrix} i_a \\ \omega_m \\ \theta_m \end{bmatrix} = \begin{bmatrix} -\dfrac{R_a}{L_a} & -\dfrac{K_b}{L_a} & 0 \\ \dfrac{K_i}{J_m} & -\dfrac{B_m}{J_m} & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_a \\ \omega_m \\ \theta_m \end{bmatrix} + \begin{bmatrix} \dfrac{1}{L_a} & 0 \\ 0 & -\dfrac{1}{J_m} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} e_a \\ T_L \end{bmatrix}$$




$$
\begin{aligned}
\frac{\Theta_m(s)}{E_a(s)} &= \frac{K_i}{L_a J_m s^3 + (R_a J_m + B_m L_a)s^2 + (K_b K_i + R_a B_m)s} \\
&= \frac{K_i}{s\left[L_a J_m s^2 + (R_a J_m + B_m L_a)s + K_b K_i + R_a B_m\right]}
\end{aligned}
$$

- Essentially an integrator
- A built-in feedback loop

## 4-7 Linearization of Nonlinear Systems

### Motivation

Consider a nonlinear system represented by vector-matrix state equation:

$$\frac{dx(t)}{dt} = f(x(t), u(t)).$$

State vector: $x(t) \in \mathbb{R}^n \rightarrow n \times 1$ vector;

Input vector: $u(t) \in \mathbb{R}^p \rightarrow p \times 1$ vector;

Vector field: $f(x(t), u(t)) \in \mathbb{R}^n$.

**Linearization:** expanding $f$ into a Taylor series about a nominal operating point or trajectory, discarding higher order terms.

## Linearization

Consider a nominal trajectory (equilibrium):

$$\dot{x}_0(t) = f(x_0(t), u_0(t)).$$

Then,

$$\begin{aligned}\dot{x}_i(t) &= f_i(x(t), u(t)) \\ &= f_i(x_0(t), u_0(t)) + \sum_{j=1}^{n} \left.\frac{\partial f_i(x,u)}{\partial x_j}\right|_{(x_0,u_0)} (x_j - x_{0j}) \\ &\quad + \sum_{j=1}^{p} \left.\frac{\partial f_i(x,u)}{\partial u_j}\right|_{(x_0,u_0)} (u_j - u_{0j}),\end{aligned}$$

where $i = 1, \cdots, n$. Let $\Delta x_i = x_i - x_{0i}$, $\Delta u_i = u_i - u_{0i}$:

$$\Delta \dot{x}_i = \dot{x}_i - \dot{x}_{0i} = \dot{x}_i(t) - f_i(x_0(t), u_0(t)).$$

We then have

$$\Delta \dot{x}_i = \sum_{j=1}^{n} \left.\frac{\partial f_i(x,u)}{\partial x_j}\right|_{(x_0,u_0)} \Delta x_j + \sum_{j=1}^{p} \left.\frac{\partial f_i(x,u)}{\partial u_j}\right|_{(x_0,u_0)} \Delta u_j$$

$$= \begin{bmatrix} \frac{\partial f_i}{\partial x_1} & \cdots & \frac{\partial f_i}{\partial x_n} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \vdots \\ \Delta x_n \end{bmatrix} + \begin{bmatrix} \frac{\partial f_i}{\partial u_1} & \cdots & \frac{\partial f_i}{\partial u_p} \end{bmatrix} \begin{bmatrix} \Delta u_1 \\ \vdots \\ \Delta u_p \end{bmatrix}$$

In vector-matrix form

$$
\begin{bmatrix} \Delta\dot{x}_1 \\ \Delta\dot{x}_2 \\ \vdots \\ \Delta\dot{x}_n \end{bmatrix}
=
\underbrace{\begin{bmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\
\frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n}
\end{bmatrix}}_{A \text{ (Jacobian Matrix)}}
\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{bmatrix}
$$

$$
+
\underbrace{\begin{bmatrix}
\frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \cdots & \frac{\partial f_1}{\partial u_p} \\
\frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \cdots & \frac{\partial f_2}{\partial u_p} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_n}{\partial u_1} & \frac{\partial f_n}{\partial u_2} & \cdots & \frac{\partial f_n}{\partial u_p}
\end{bmatrix}}_{B}
\begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ \vdots \\ \Delta u_p, \end{bmatrix}
$$

or

$$\Delta\dot{x} = A \cdot \Delta x + B \cdot \Delta u.$$

## Example

$$\begin{cases} \dot{x}_1(t) = \dfrac{-1}{x_2^2(t)}, \\ \dot{x}_2(t) = u(t) \cdot x_1(t). \end{cases}$$

Consider a nominal trajectory $(x_{01}(t), x_{02}(t))$ starting from $x_1(0) = x_2(0) = 1$ and $u(t) = 0$. First, we solve the nominal trajectory.

$$u(t) = 0 \quad \Rightarrow \quad \dot{x}_2(t) = 0 \quad \Rightarrow \quad x_2(t) = \text{const} = x_2(0) = 1.$$

Therefore,

$$\dot{x}_1(t) = -1 \quad \Rightarrow \quad x_1(t) = -t + x_1(0) = -t + 1.$$

The nominal trajectory is then

$$\begin{cases} x_{01}(t) = -t + 1, \\ x_{02}(t) = 1. \end{cases}$$

The Jacobian matrix can be obtained as:

$$\frac{\partial f_1}{\partial x_1} = 0, \qquad \frac{\partial f_1}{\partial x_2} = \frac{2}{x_2^3(t)}, \quad \frac{\partial f_1}{\partial u} = 0,$$

$$\frac{\partial f_2}{\partial x_1} = u(t), \quad \frac{\partial f_2}{\partial x_2} = 0, \qquad \frac{\partial f_2}{\partial u} = x_1(t).$$

We then get

$$\begin{bmatrix} \Delta \dot{x}_1 \\ \Delta \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & \frac{2}{x_2^3(t)} \\ u_0(t) & 0 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ x_{01}(t) \end{bmatrix} \Delta u$$

$$= \begin{bmatrix} 0 & 2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1-t \end{bmatrix} \Delta u.$$

## 4-8 Time Delay

Often seen in hydraulic system and computer control


Let $b(t) = y(t - T_d)$, $B(s) = e^{-T_d s} Y(s)$. Then

$$\frac{B(s)}{Y(s)} = e^{-T_d s}.$$

This is difficult to handle.

We can then approximate it by rational functions:

$$
\begin{aligned}
e^{-T_d s} &\approx 1 - T_d s + \frac{T_d^2 s^2}{2} \\
&\approx \frac{1}{1 + T_d s + \frac{T_d^2 s^2}{2}}.
\end{aligned}
$$

However, this is not valid when $T_d$ is large. A better one is Padé approximation:

$$
e^{-T_d s} \approx \frac{1 - T_d s/2}{1 + T_d s/2}.
$$

A zero in RHP may result in a small negative undershoot in step response.