2.12/2.120
Introduction to Robotics

February 23-24, 2023

Lab 3: DC Motor Control

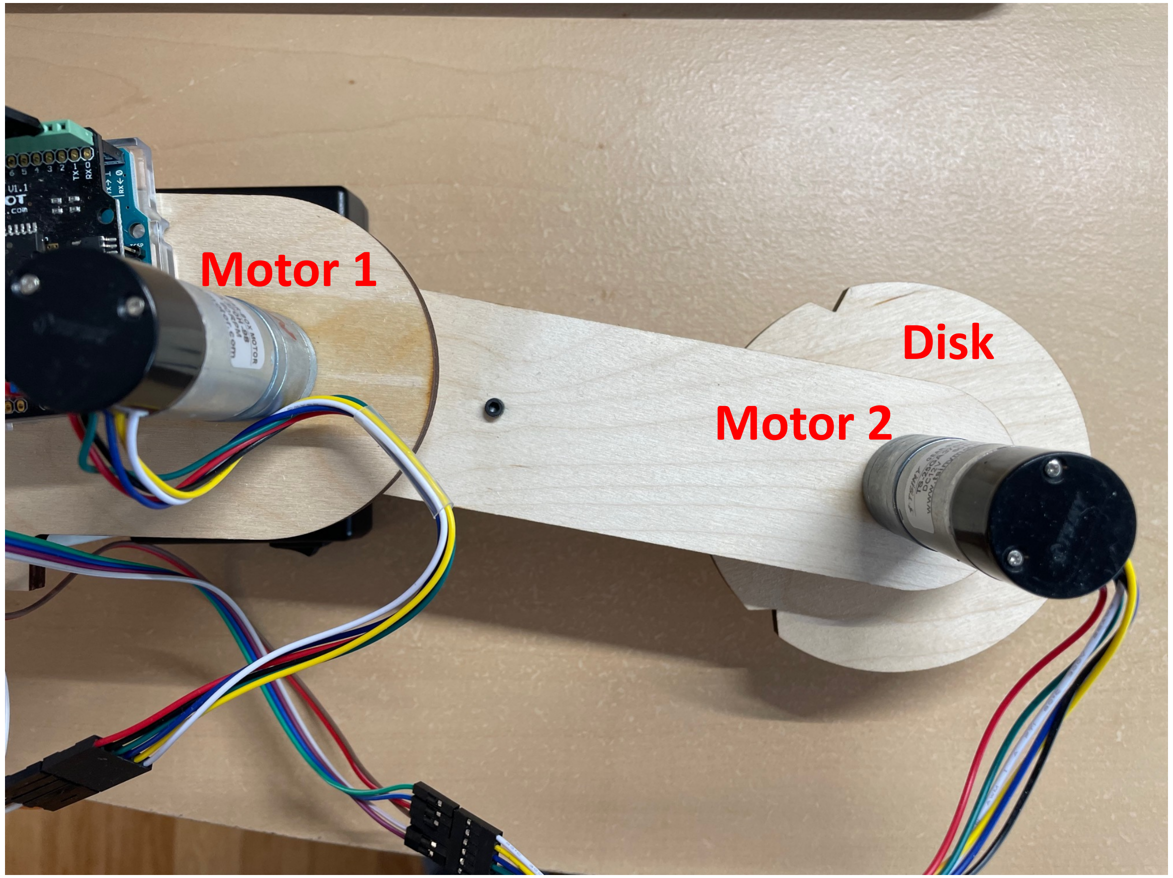
Objectives

- Understand the Proportional, Integral, and Derivative (PID) control actions.
- Derive closed-loop transfer functions.
- Design and fine-tune a PI controller for velocity control.
- Design and fine-tune a PD controller for position control.

- Term Project: Task strategy.

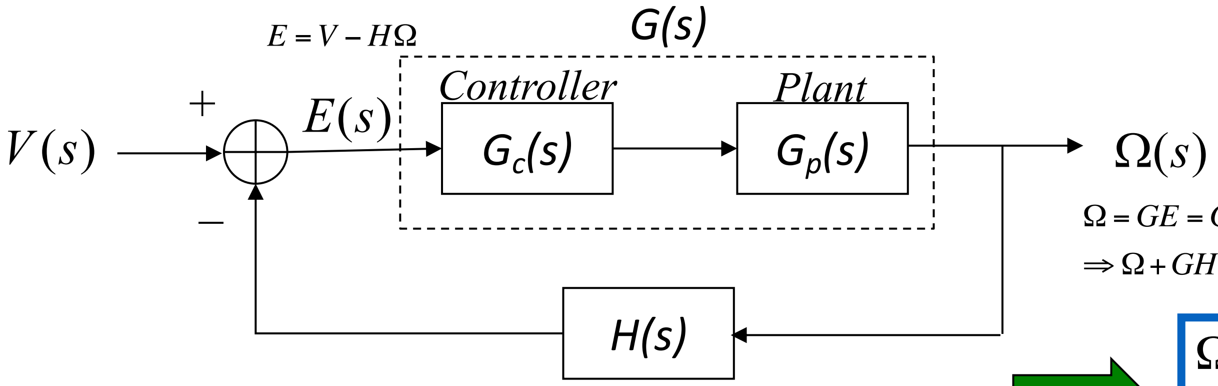
System Setup

- Detach the second link and attach the disc to Motor 2.



Standard Closed-Loop PID Control

$E = V - H\Omega$



$$\Omega = GE = G(V - H\Omega) = GV - GH\Omega$$

$$\Rightarrow \Omega + GH\Omega = (1 + GH)\Omega = GV$$

$$\boxed{\frac{\Omega(s)}{V(s)} = \frac{G(s)}{1 + G(s)H(s)} = G_{cl}(s)}$$

PID controller transfer function

$$G_c(s) = K_p + \frac{K_i}{s} + K_d s$$

$$= \frac{K_d s^2 + K_p s + K_i}{s}$$

$$= K_d \left(\frac{s^2 + \left(K_p / K_d \right) s + \left(K_i / K_d \right)}{s} \right)$$

PID controller pseudo code

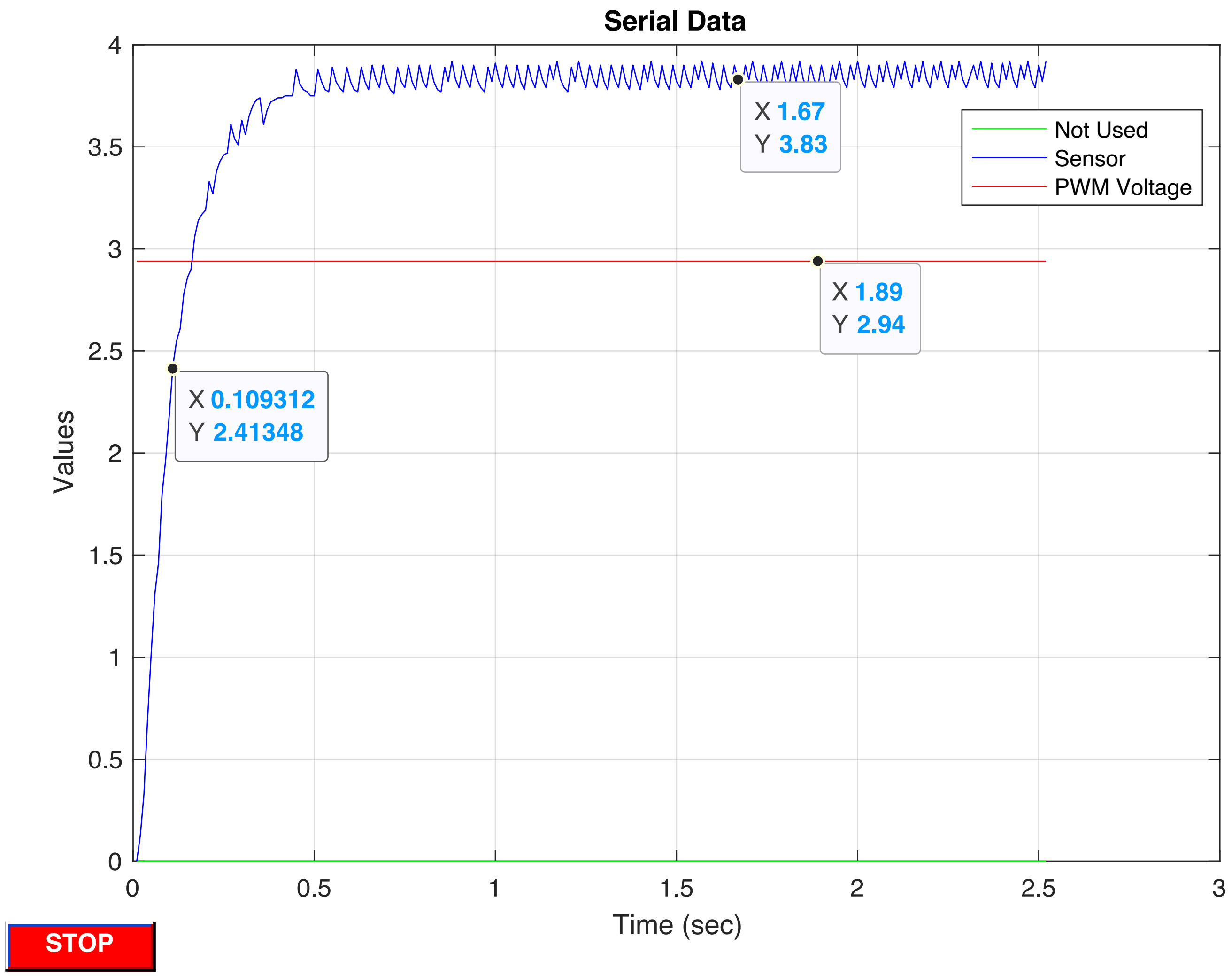
$$PID\_output = K_p e(t) + K_i \int e(t)\,dt + K_d \frac{d}{dt} e(t)$$

$$e(t) = \text{set\_point - sensor\_output}$$

$$\int e(t)\,dt \approx \sum_t e(t)\Delta t$$

$$\frac{d}{dt} e(t) \approx \frac{e(t) - e(t-1)}{\Delta t}$$

Estimate Open-Loop Transfer Function



Plant transfer function:

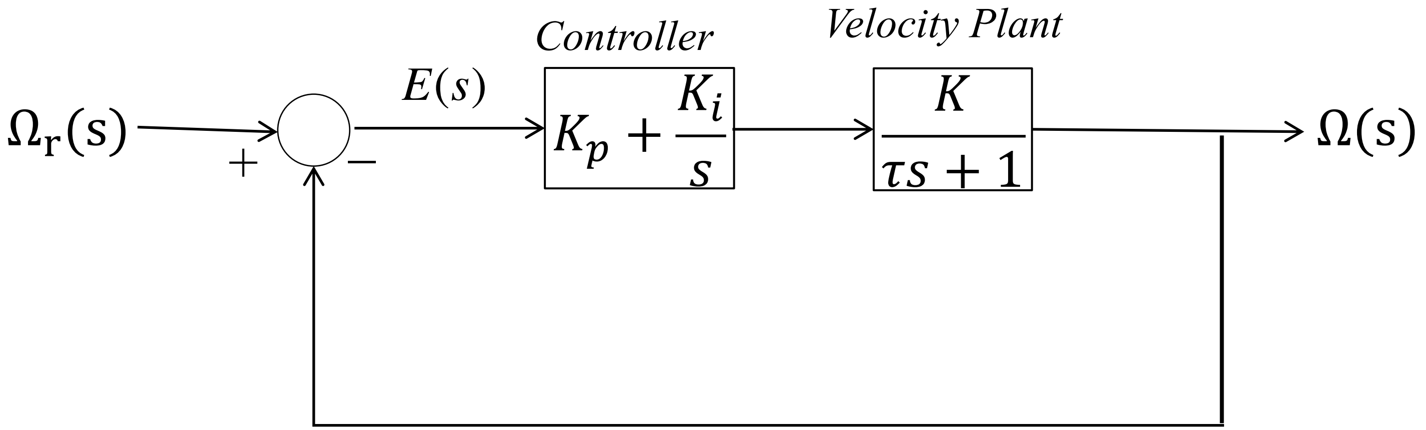
$$G_p(s) = \frac{\Omega\,(s)}{V_{pwm}(s)} = \frac{K_{dc}}{\tau s + 1} = \frac{1.3}{0.11s + 1}$$

PWM = 255 corresponds to 5V from PWM pin (1V = 51 PWM) so we can further represent the transfer function as:

$$G_p(s) = \frac{\Omega\,(s)}{PWM(s)} = \frac{\frac{K_{dc}}{51}}{\tau s + 1} = \frac{0.0255}{0.11s + 1}$$

Velocity Control with P&I Control Actions

$$K = \frac{K_{dc}}{51}$$



Closed-Loop Transfer Function: $$G_{cl}(s) = \frac{K(K_p s + K_i)}{\tau s^2 + (1 + K_p K)s + K_i K}$$

One zero

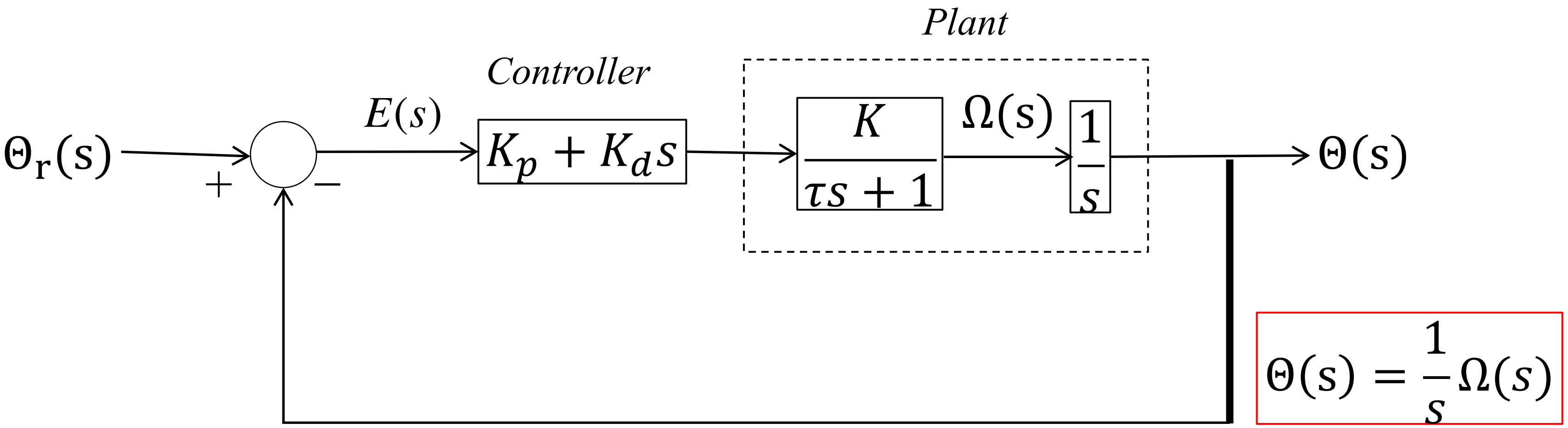
Two poles

Steady-state

$$\Omega_{ss} = \lim_{t\to\infty} \Omega(t) = \lim_{s\to 0} s\left(\frac{1}{s}\right) \mathrm{G_{cl}}(s) = \left(\frac{K_i K}{K_i K}\right) = 1$$

No steady-state error with a step input

Position Control with P&D Control Actions



Closed-Loop Transfer Function:

$$G_{cl}(s) = \frac{K(K_p + K_d s)}{\tau s^2 + (1 + K_d K)s + K_p K}$$

One zero

Two poles

A “free” integrator

No steady-state error with a step input

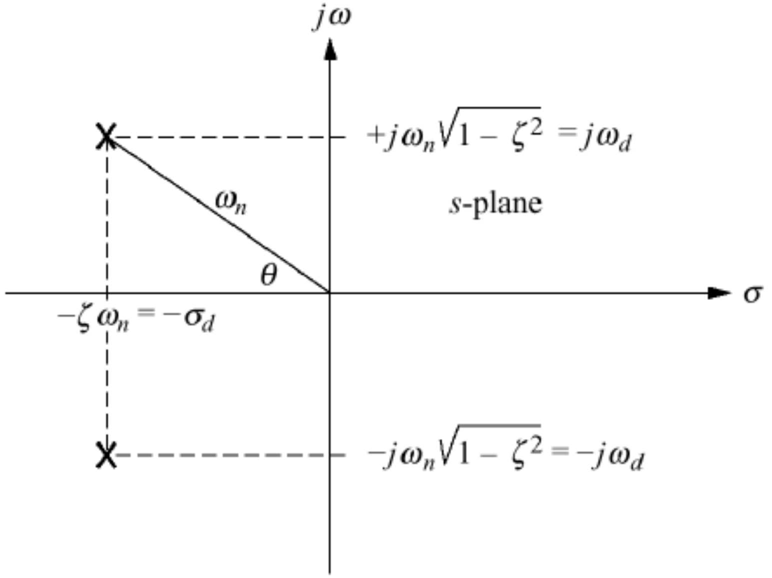
Comparison of Closed-Loop Transfer Functions

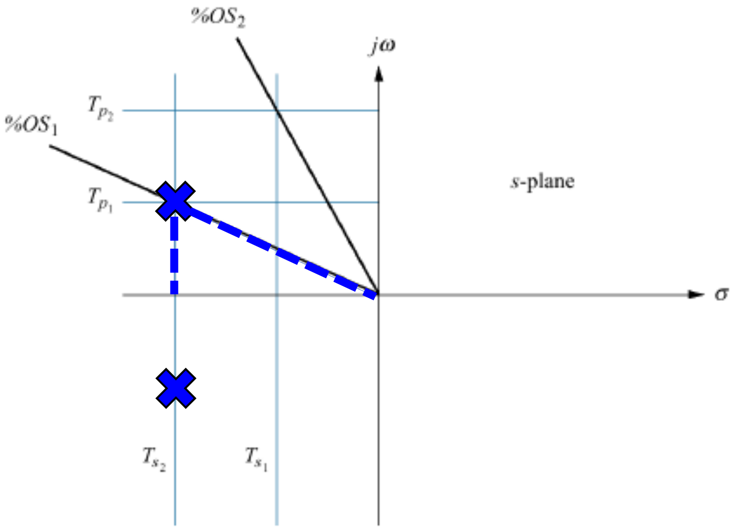
PI Control of Velocity: $$G_{cl}(s) = \frac{K(K_p s + K_i)}{\tau s^2 + (1 + K_p K)s + K_i K}$$

$$\boxed{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

PD Control of Position: $$G_{cl}(s) = \frac{K(K_p + K_d s)}{\tau s^2 + (1 + K_d K)s + K_p K}$$

2^{nd} Order System Poles





$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0$$

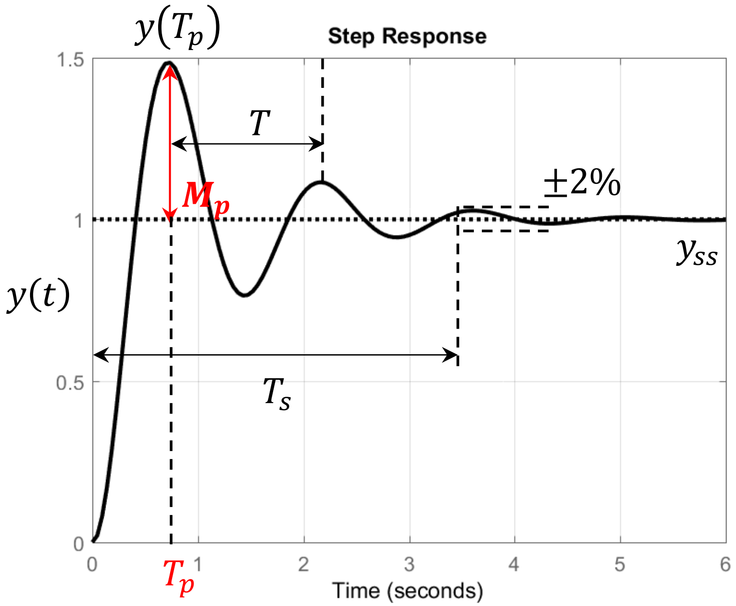
$$\boxed{s_{1,2} = -\sigma_d \pm j\omega_d = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}}$$

$$T_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}}$$

$$T_s = \frac{4}{\zeta\omega_n} \quad \textit{(within 2\%)}$$

$$\%OS = e^{-\left(\frac{\zeta\pi}{\sqrt{1-\zeta^2}}\right)} \times 100$$

Dynamic Response Characteristics



Example: Figure 1

$y_{ss} = 1 \qquad y(T_P) = 1.5$

$M_P = 0.5 \qquad \%OS = 50\%$

$$\zeta = -\frac{\ln(0.5)}{\sqrt{\pi^2 + \ln^2(0.5)}} = 0.215$$

$T_s \approx \frac{4}{\sigma} = \frac{4}{\zeta\omega_n}$ 2% Settling Time

$T_s \approx \frac{4.6}{\sigma} = \frac{4.6}{\zeta\omega_n}$ 1% Settling Time

$\omega_d = \omega_n\sqrt{1-\zeta^2} = \frac{2\pi}{T}$ Damped Natural Frequency

$T_p = \frac{\pi}{\omega_d}$ Peak Time

$M_P \triangleq y(T_p) - y_{ss}$ Overshoot (Magnitude)

$\%OS \triangleq \frac{y(T_p) - y_{ss}}{y_{ss}} \times 100$ Overshoot (Percent)

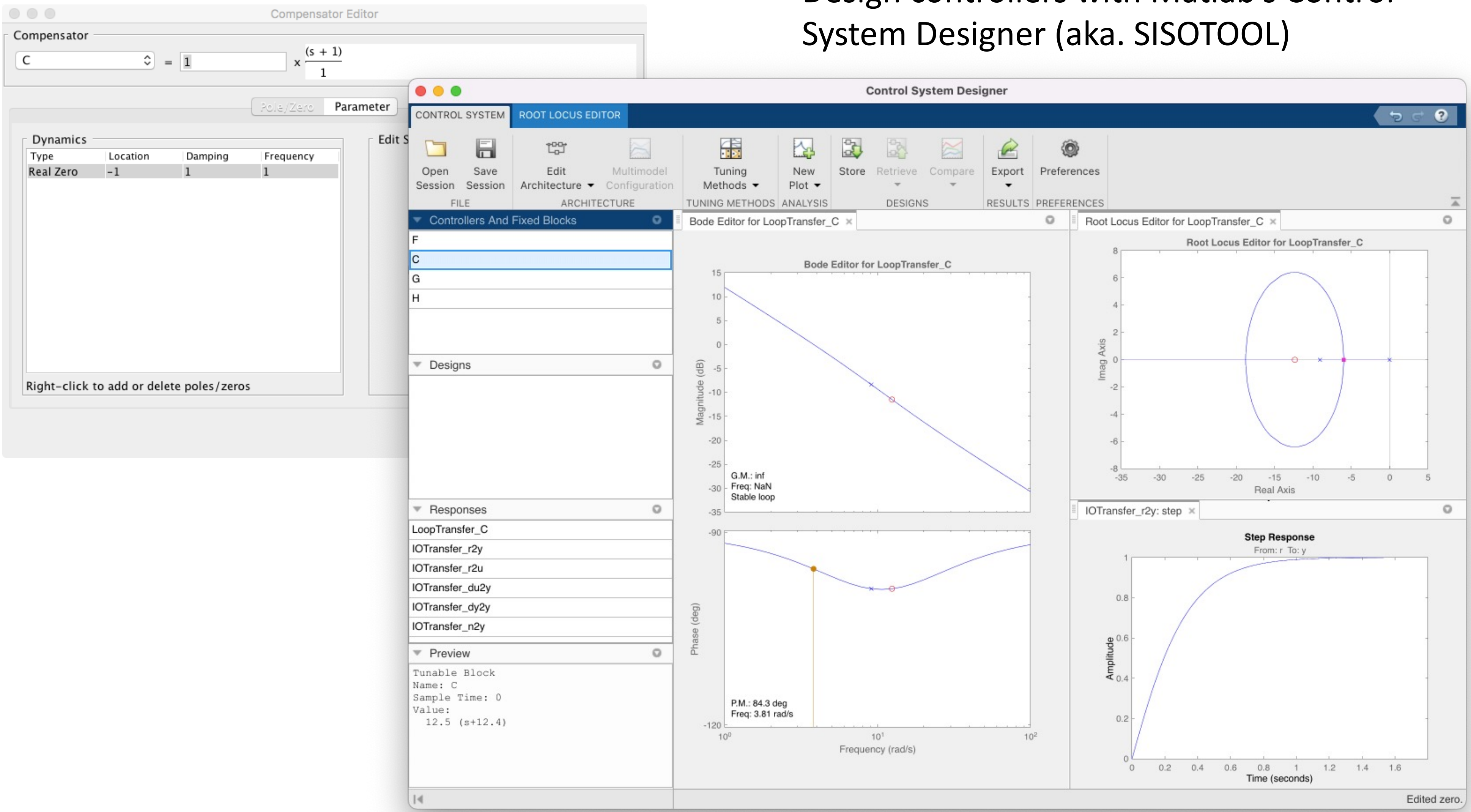
$$\frac{M_P}{y_{ss}} = \frac{\%OS}{100} = e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}} \quad , 0 \le \zeta < 1$$

Controller Design

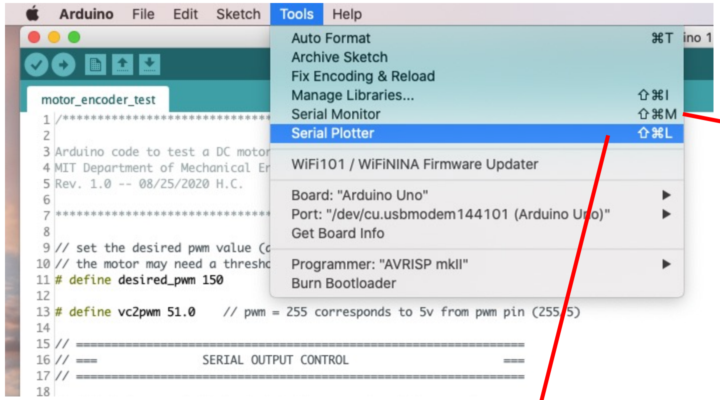
- Control action comparison:
  - *Proportional* – improves speed but with steady-state error in some cases
  - *Integral* – improves steady state error but with less stability; may create overshoot, longer transient, or integrator windup
  - *Derivative* – improves stability but sensitive to noise; may create large output when the input is not a continuous signal
- Reduce overall gain can increase stability but with slower response
- Avoid saturations
- Set integrator limits to prevent windups

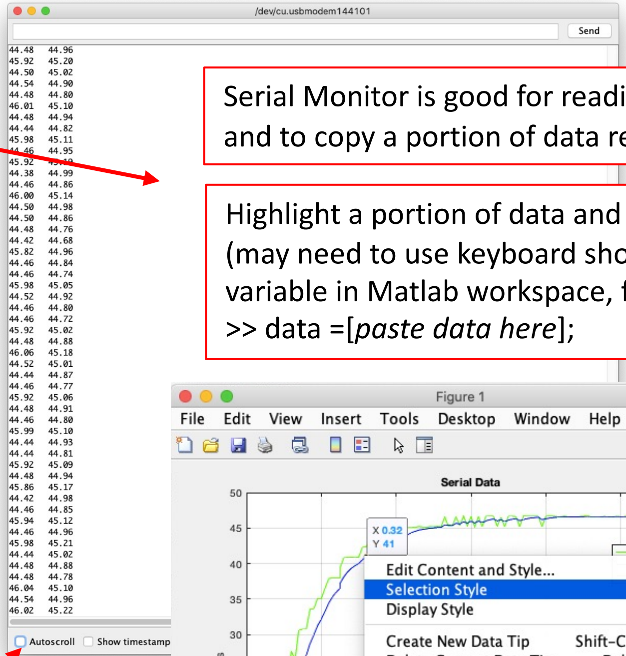
FYI: PD Controller Design with Matlab's SISOTOOL

Design controllers with Matlab's Control System Designer (aka. SISOTOOL)



Monitoring/Capturing Serial Data

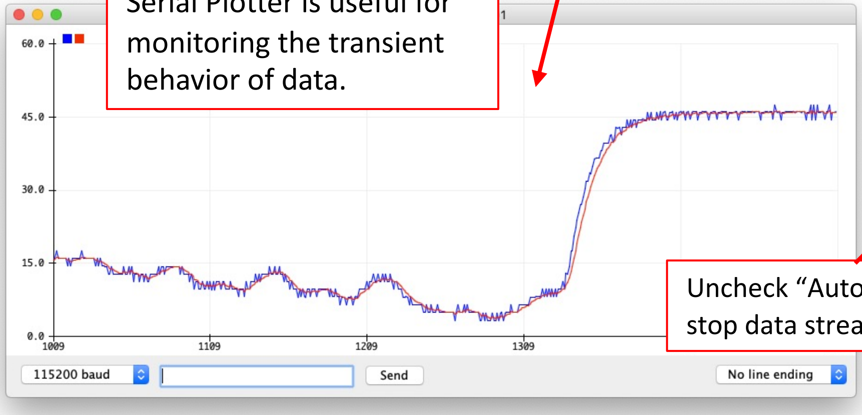




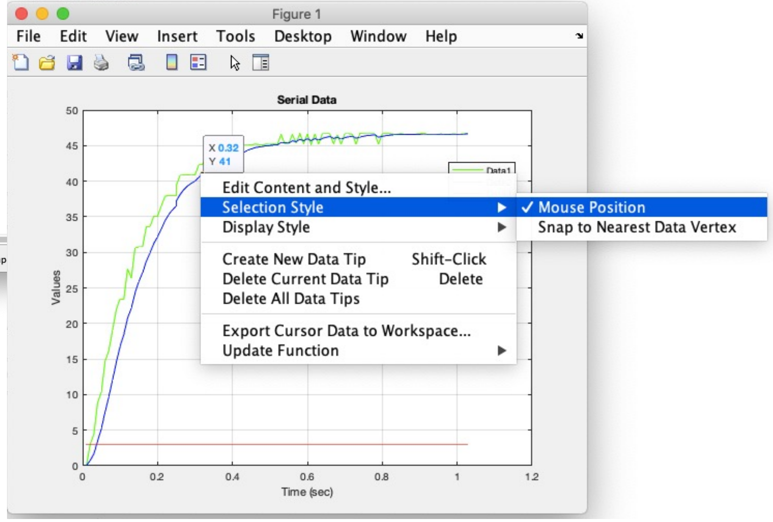
Serial Monitor is good for reading data values and to copy a portion of data record to clipboard.

Highlight a portion of data and copy and paste (may need to use keyboard shortcuts) to a variable in Matlab workspace, for example:
>> data =[*paste data here*];

Serial Plotter is useful for monitoring the transient behavior of data.



Uncheck "Autoscroll" to stop data streaming.



Neither "Serial Monitor" nor "Serial Plotter" is capable of saving data to a file.

The Matlab script "SerialRead.m" can be used to stream serial data directly into Matlab workspace for plotting and for post processing.

Arduino File

motor_control | Arduino IDE 2.0.3

Arduino Uno

motor_control.ino

```
*************************************************************************************/

//#define OPEN_LOOP
#define VELOCITY_CONTROL
//#define POSITION_CONTROL

#define period_us 10000  // define sample period in microseconds (1 sec = 1000000 us)

// ================================================================
// ===                PID FEEDBACK CONTROLLER                   ===
// ================================================================
// Provide PID controller gains here
float Kp = 0.0;
float Ki = 0.0;
float Kd = 0.0;

// ================================================================
// ===                SERIAL OUTPUT CONTROL                     ===
// ================================================================
// Switch between built-in Serial Plotter and Matlab streaming
// comment out both to disable serial output

#define arduinoSerialPrint
//#define MATLABSerialPrint

// ================================================================
// ===                SETPOINT TYPE                             ===
// ================================================================
// Define setpoint (enable one of the followings)
// disable all to use a constant setpoint

#define SQUARE_WAVE
//#define SINE_WAVE
```

Enable velocity or position control

PID gains

Enable Arduino or Matlab serial output

Enable square or sine wave setpoint

Output

Code Files and Deliverable

- Required code in GitHub: https://github.com/mit212/lab3_2023
- Demonstrate to the lab staff your closed-loop responses.

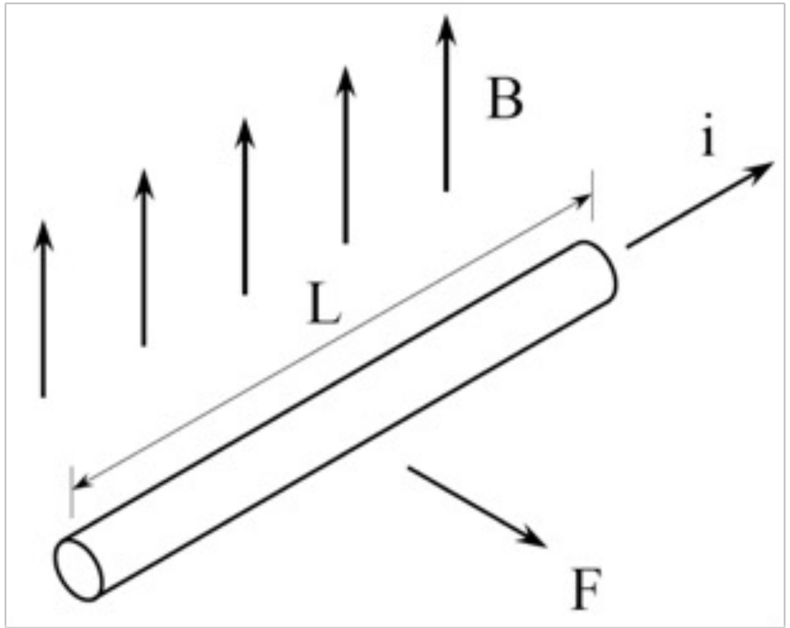
Term Project: Task Strategy

- Manipulator:
  - Design and fabricate an end-effector to perform CPR (Cardiopulmonary Resuscitation).
  - Detect and guide the end-effector to the chest of the patient.
  - May need computer vision and force feedback control.
- Mobile Robot:
  - Design and fabricate a mechanism to get an AED (Automated External Defibrillator) bag.
  - Deliver the AED bag to the destination.
  - May need obstacle avoidance, path planning, and autonomous navigation.
- Avatar:
  - Interact with the patient and the environment with VR headset and controllers.
  - Tele-operation of the manipulator.

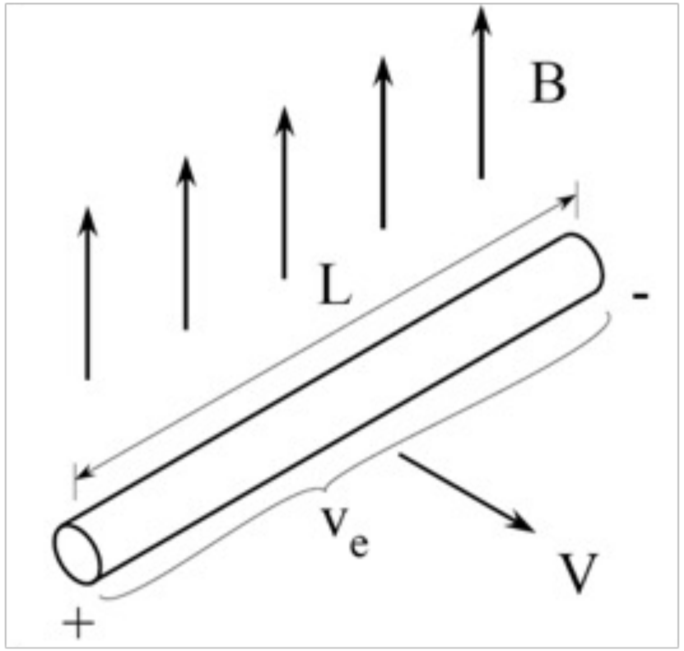
DC Motor

DC Motor Principle

Lorentz law:
magnetic field applies force to a current (Lorentz force)



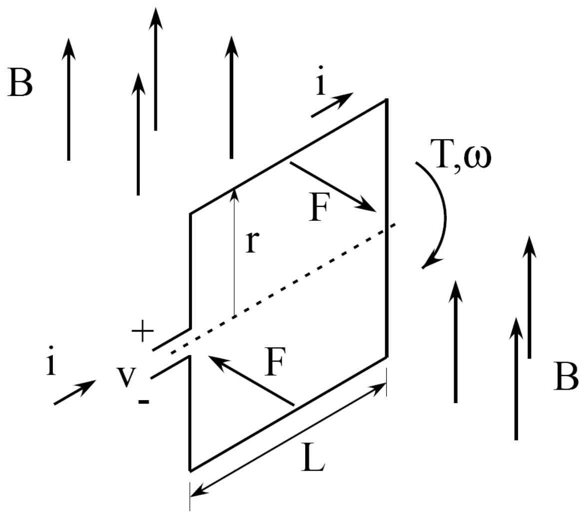
Faraday law:
moving in a magnetic field results in potential (back EMF)

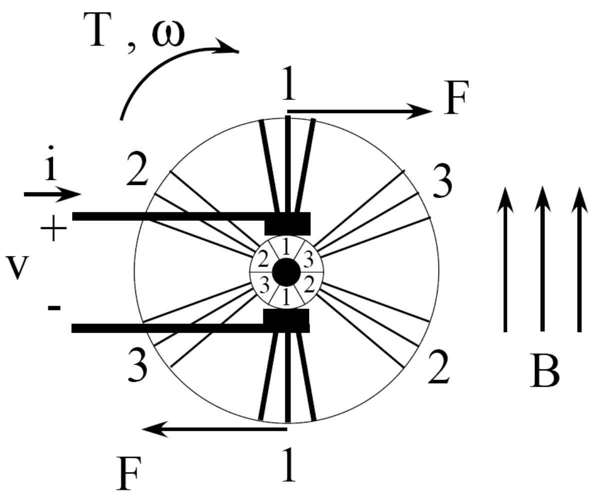


$$F = (\mathbf{i} \times \mathbf{B}) \cdot l = iBl \qquad (\mathbf{i} \perp \mathbf{B})$$

$$v_e = \mathbf{V} \times \mathbf{B} \cdot l = VBl \qquad (\mathbf{V} \perp \mathbf{B})$$

DC Motor Principle (Cont.)





multiple windings N:
continuity of torque

$$T = 2Fr = 2(iBNl)r \qquad \text{(Lorentz law)}$$

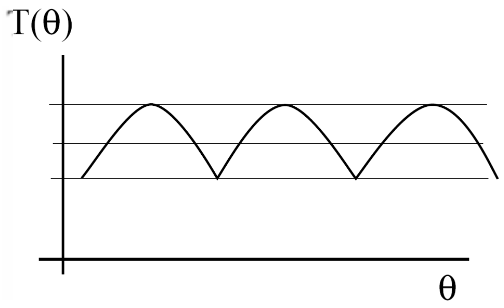
$$v_e = 2VBNl = 2(\omega r)BNl \qquad \text{(Faraday law)}$$

or where

$$T = K_m i$$

$$v_e = K_v \omega$$

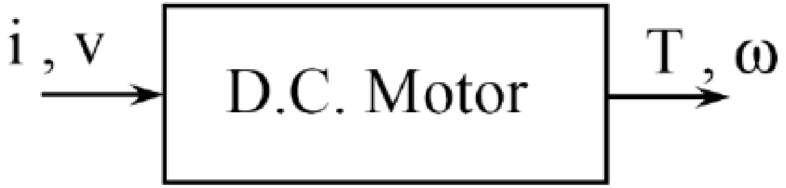
- $K_m \equiv 2BNlr$ torque constant
- $K_v \equiv 2BNlr$ back-emf constant

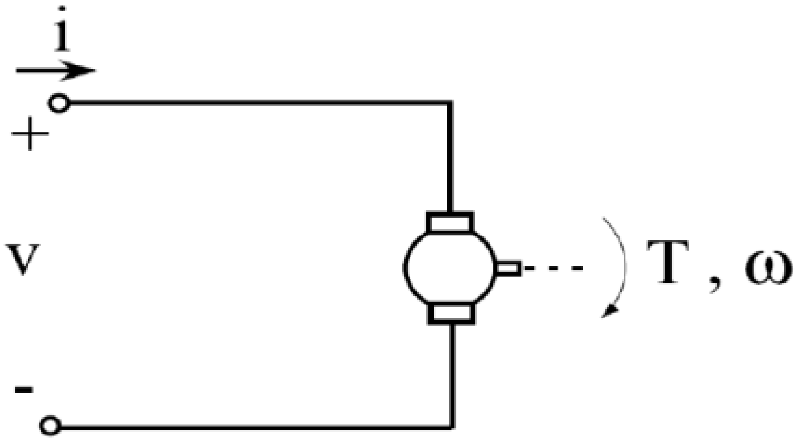


DC Motor as A Transducer

DC motor is an electromechanical system that may have a time constant close to the plant.

A transducer converts energy from one domain (e.g., electrical) to another (e.g., mechanical), or vice versa.

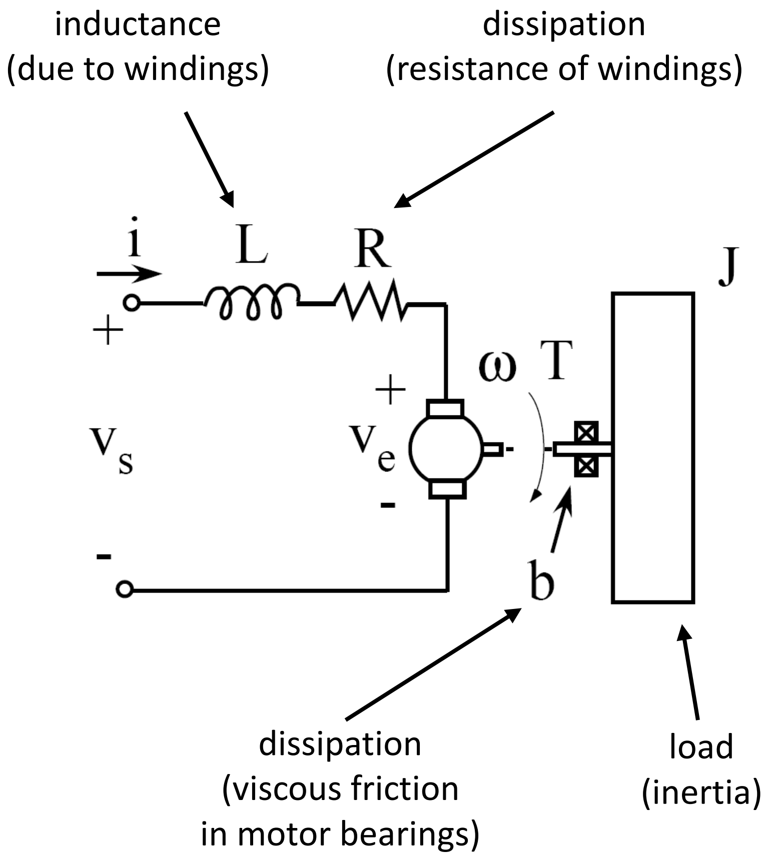




$$P_{in} = P_{out}$$

$$i(t) \cdot v(t) = T(t) \cdot \omega(t)$$

DC Motor with Mechanical Load



K_m :	motor torque constant
K_v :	motor speed (back EMF) constant

Equation of motion – Electrical

KVL: $v_s - v_L - v_R - v_e = 0$

$$\Rightarrow v_s - L\frac{di}{dt} - Ri - K_v\omega = 0$$

Equation of motion – Mechanical

Torque Balance: $T = T_b + T_J$

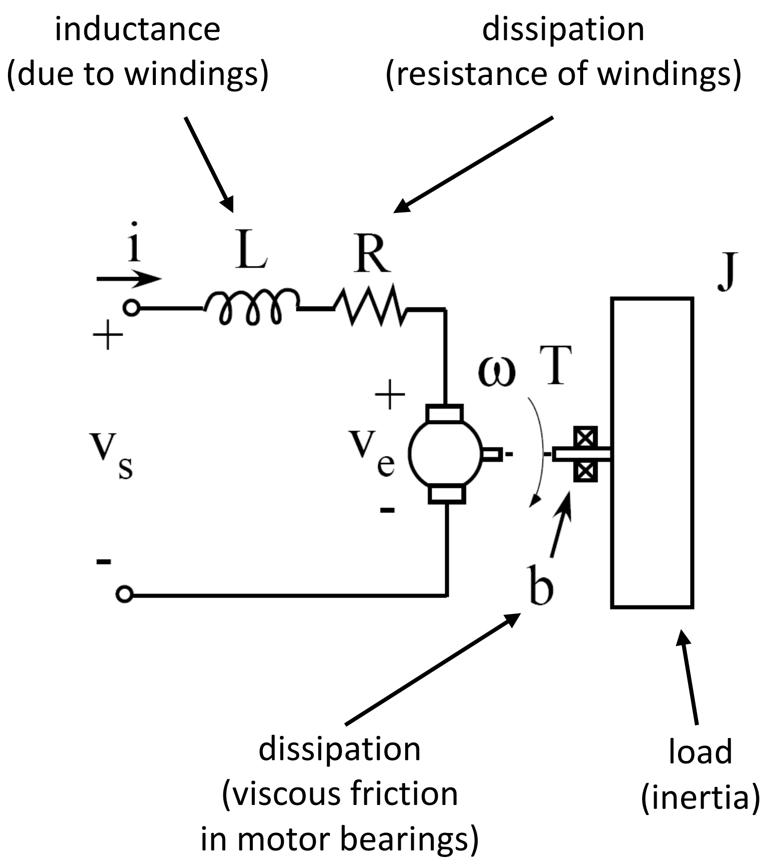
$$\Rightarrow K_m i - b\omega = J\frac{d\omega}{dt}$$

Combined equations of motion

$$L\frac{di}{dt} + Ri + K_v\omega = v_s$$

$$J\frac{d\omega}{dt} + b\omega = K_m i$$

DC Motor with Mechanical Load



Equation of motion – Electrical

KVL: $V_s(s) - V_L(s) - V_R(s) - V_e(s) = 0$

$$V_s(s) - LsI(s) - RI(s) - K_v\Omega(s) = 0$$

Equation of motion – Mechanical

Torque Balance: $T(s) = T_b(s) + T_J(s)$

$$K_m I(s) - b\Omega(s) = Js\Omega(s)$$

Combined equations of motion

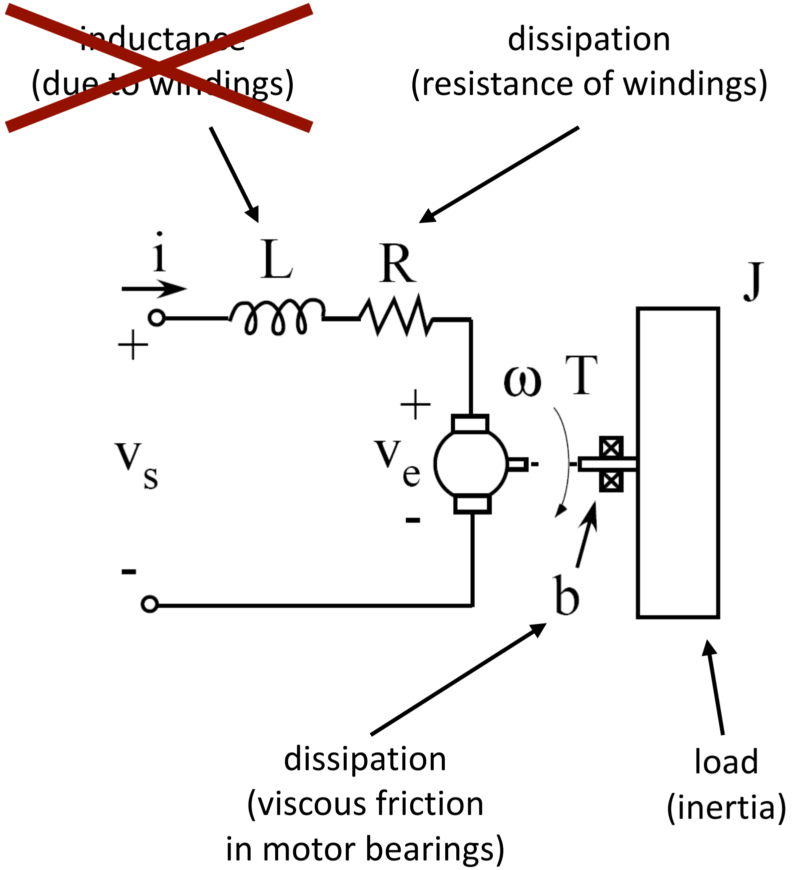
$$LsI(s) + RI(s) + K_v\Omega(s) = V_s(s)$$

$$Js\Omega(s) + b\Omega(s) = K_m I(s)$$

$$\Rightarrow \left[(Ls+R)\left(\frac{Js+b}{K_m}\right) + K_v\right]\Omega(s) = V_s(s)$$

$$\boxed{\Rightarrow \left[\frac{LJ}{R}s^2 + \left(\frac{Lb}{R}+J\right)s + \left(b + \frac{K_m K_v}{R}\right)\right]\Omega(s) = \frac{K_m}{R}V_s(s)}$$

DC Motor with Mechanical Load



Neglecting the inductance (why?)

$$L \approx 0$$

$$\Rightarrow \left[Js + \left(b + \frac{K_m K_v}{R} \right) \right] \Omega(s) = \frac{K_m}{R} V_s(s)$$

This is our familiar 1st–order system!

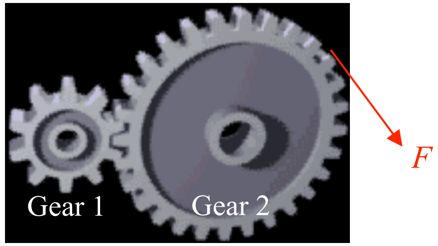
If we are given step input $v_s(t) = V_0 u(t)$
$\Rightarrow$ we already know the step response

$$\omega(t) = \frac{K_m}{R} V_0 \left(1 - e^{-t/\tau}\right) u(t),$$

where now the time constant is

$$\boxed{\tau = \frac{J}{\left(b + \dfrac{K_m K_v}{R} \right)}.}$$

Gear Ratio and Power Conservation



$$\text{Gear Ratio } N: \quad \frac{n_1}{n_2} = \frac{r_1}{r_2} = \frac{\Omega_2}{\Omega_1} = \frac{T_1}{T_2} = \frac{F_1^T r_1}{F_2^T r_2}$$

$$\frac{n_1}{n_2} = \frac{44}{180}$$

Unit Conversion:

$$rpm = \frac{2\pi}{60}(rad/s)$$

$$N = \frac{kg \times m}{s^2}$$

$$V(voltage) = \frac{kg \times m^2}{s^3 \times A}$$

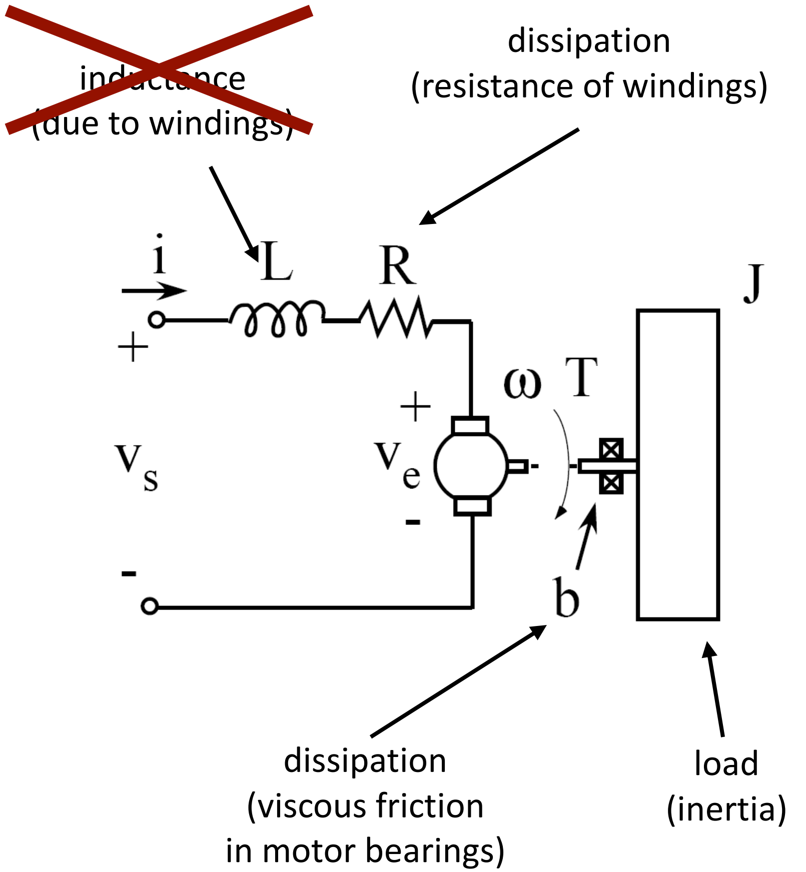
Power Conservation:

$$P_{mechanical} = P_{electrical}$$

$$\begin{aligned} P_{mechanical}(t) &= T(t) \times \Omega(t) \\ &= K_m \times i(t) \times \Omega(t) \end{aligned}$$

$$\begin{aligned} P_{electrical}(t) &= v_b(t) \cdot i(t) \\ &= K_v \cdot \Omega(t) \cdot i(t) \end{aligned}$$

DC Motor with Mechanical Load



K_m : motor torque constant

K_v : motor speed (back EMF) constant

Neglecting the inductance…

$$L \approx 0$$

$$\Rightarrow \left[Js + \left(b + \frac{K_m K_v}{R}\right)\right]\Omega(s) = \frac{K_m}{R}V_s(s)$$

This is our familiar 1st–order system!

If we are given step input $v_s(t) = V_0 u(t)$
$\Rightarrow$ we already know the step response

$$\omega(t) = \frac{K_m}{R}V_0\left(1 - \mathrm{e}^{-t/\tau}\right)u(t),$$

where now the time constant is

$$\tau = \frac{J}{\left(b + \dfrac{K_m K_v}{R}\right)}.$$