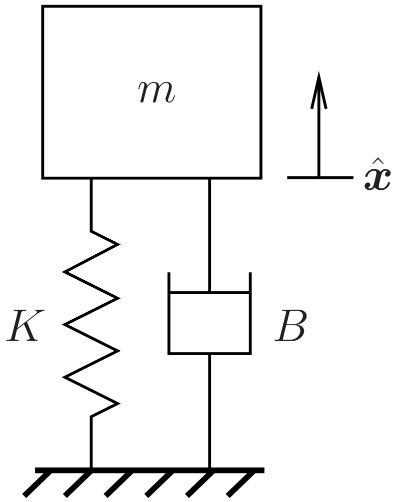
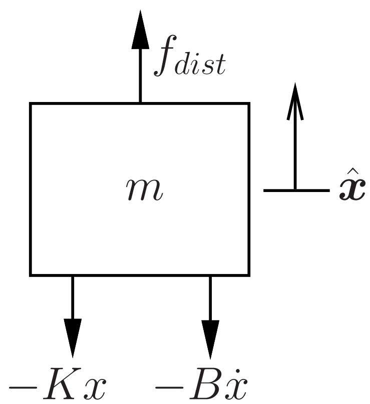
The Spring-Mass-Damper

$$F_k = -Kx$$



$$F_b = -B\dot{x}$$



$$\sum F = m\ddot{x} = -B\dot{x} - Kx$$

$$m\ddot{x} + B\dot{x} + Kx = 0, \qquad \text{or}$$

$$\ddot{x} + (B/m)\dot{x} + (K/m)x = 0$$

or,

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x = 0 \qquad \textit{harmonic oscillator}$$

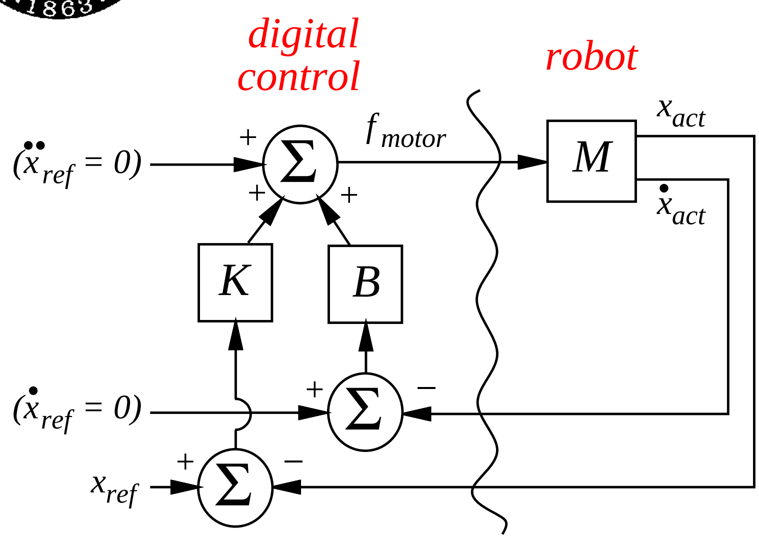
where:

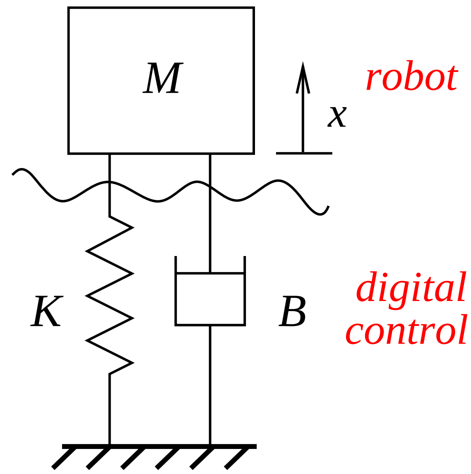
$$\omega_n = (K/m)^{1/2} \qquad [rad/sec] \text{ - } \textit{natural frequency}$$
$$\zeta = B/2(Km)^{1/2} \qquad 0 \leq \zeta \leq \infty \text{ - } \textit{damping ratio}$$

a change of variables $x'(t) = x(t) - x_{ref}$ accounts for arbitrary reference positions

Copyright ©2021 Roderic Grupen

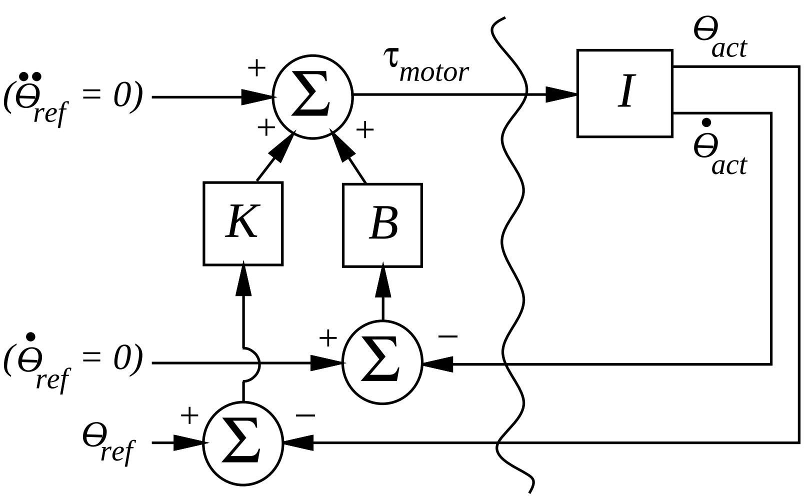
Closed-Loop Control



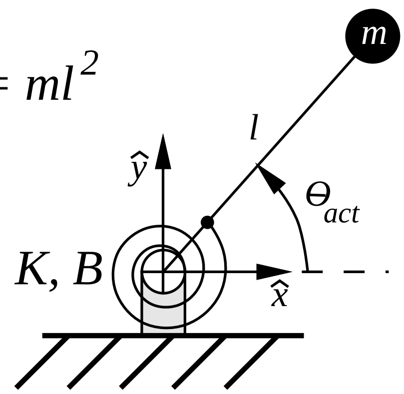


sample and hold $\Delta t = \tau$
where $\frac{1}{\tau}$ [Hz] is the servo rate

analog $\Delta t \rightarrow 0$



$I = ml^2$



Copyright ©2021 Roderic Grupen

The Laplace Transform

$$f(t) \sim e^{st}$$

$$\frac{d}{dt}[f(t)] = \dot{f}(t) \sim se^{st}$$

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2\theta = 0 \quad \underset{\mathcal{L}^{-1}(\cdot)}{\overset{\mathcal{L}(\cdot)}{\rightleftarrows}} \quad \left[s^2 + 2\zeta\omega_n s + \omega_n^2\right] X(s) = 0$$

the **Laplace transform** turns a second-order differential equation with constant coefficients into a **quadratic** polynomial in s whose roots determine the type of response

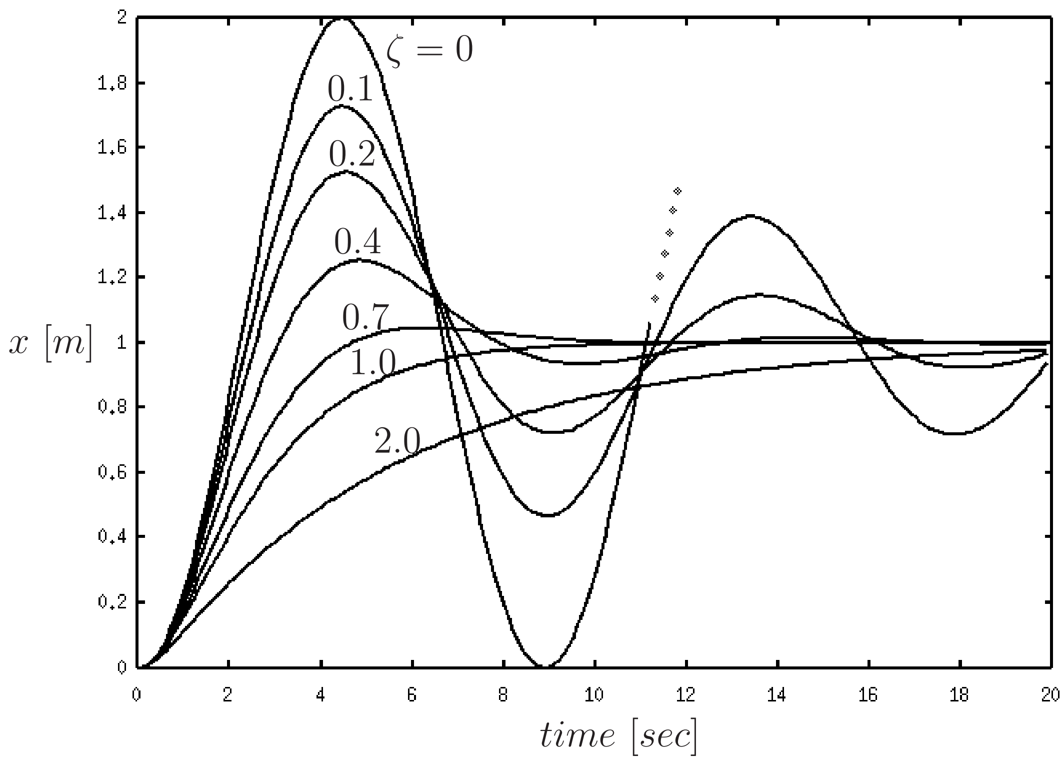
three cases:

- repeated real roots ($\zeta = 1$) — critically damped
- distinct real roots ($\zeta > 1$) — overdamped
- complex conjugates roots ($\zeta < 1$) — underdamped

Copyright ©2021 Roderic Grupen

The Spring-Mass-Damper

the *characteristic second-order behavior*



$$(K = 1.0\ [N/m],\ M = 2.0\ [kg])$$

Copyright ©2021 Roderic Grupen

Experimental Method

shoot for critical damping:

1. start with a small value for K & B (you don't know m)
2. increase K until Roger is responsive to small ($\sim \pi/4$) errors
3. when you like K, start adjusting B
   (a) if Roger is oscillating - increase B
   (b) if Roger is non-oscillating - decrease B until Roger *just* begins to oscillate (error changes sign)

Copyright ©2021 Roderic Grupen

Roger MotorUnits.c
Master Control Procedure

```
/* == the simulator executes control_roger() once ==*/
/* == every simulated 0.001 second (1000 Hz)      ==*/
control_roger(roger, time)
Robot * roger;
double time;
{
  update_setpoints(roger);

  // turn setpoint references into torques
  PDController_base(roger, time);
  PDController_arms(roger, time);
  PDController_eyes(roger, time);
}
```


Copyright ©2021 Roderic Grupen

Roger MotorUnits.c PDController_eyes()

```
double Kp_eye, Kd_eye;
// gain values set in enter_params()

/* Eyes PD controller:
/*    -pi/2 < eyes_setpoint < pi/2 for each eye */
PDController_eyes(roger, time)
Robot * roger;
double time;
{
  int i;
  double theta_error;

  for (i = 0; i < NEYES; i++) {
    theta_error = roger->eyes_setpoint[i]
                      - roger->eye_theta[i];
    // roger->eye_torque[i] = ...
  }
}
```

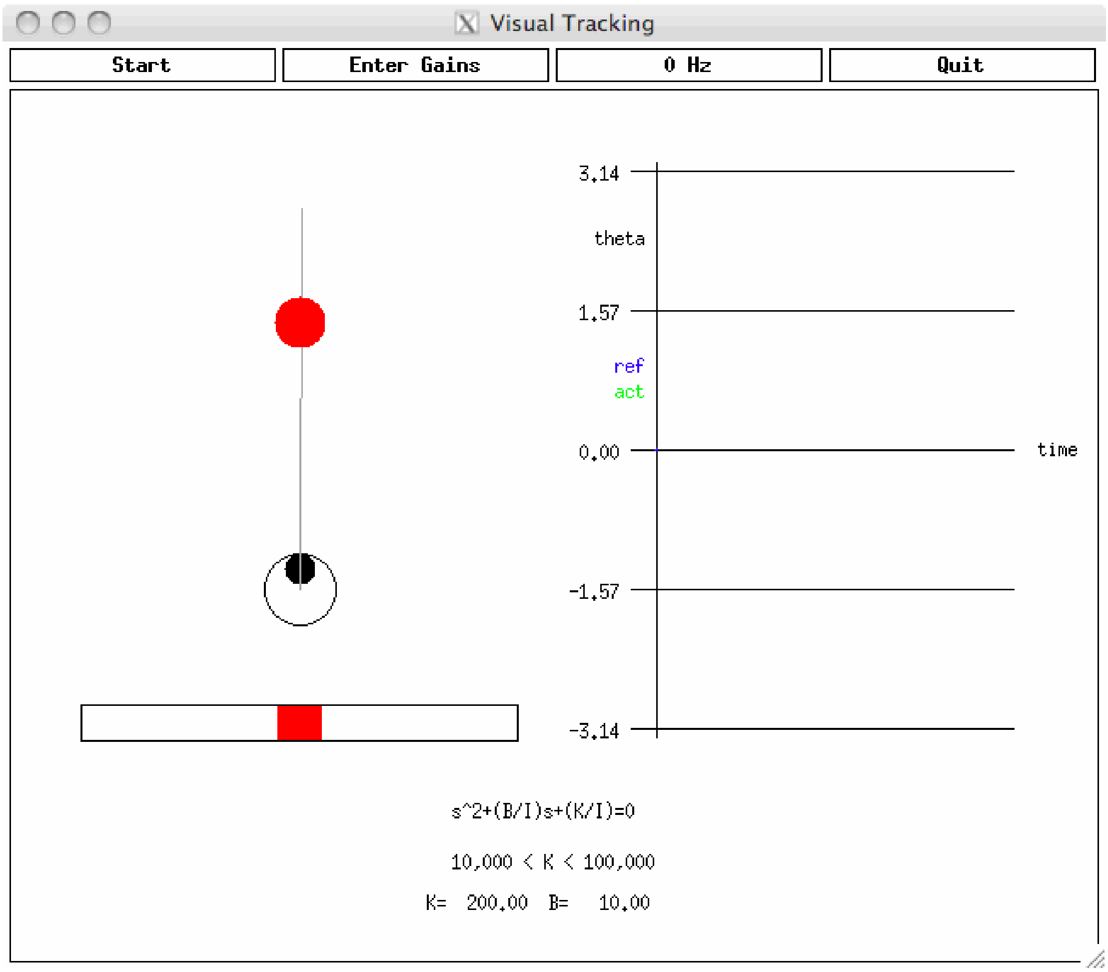

Copyright ©2021 Roderic Grupen

Roger MotorUnits.c PDcontroller_arms()

```
double Kp_arm, Kd_arm;
// gain values set in enter_params()

/* Arms PD controller: -pi < arm_setpoint < pi */
/* for the shoulder and elbow of each arm      */
PDController_arms(roger, time)
Robot * roger;
double time;
{
  int i;
  double theta_error;

  for (i = LEFT; i <= RIGHT; ++i) {
    theta_error = roger->arm_setpoint[i][0]
                     - roger->arm_theta[i][0];

    // -M_PI < theta_error < +M_PI

    // roger->arm_torque[i][0] = ...
    // roger->arm_torque[i][1] = ...
  }
}
```

 Copyright ©2021 Roderic Grupen

Class Exercise



Copyright ©2021 Roderic Grupen