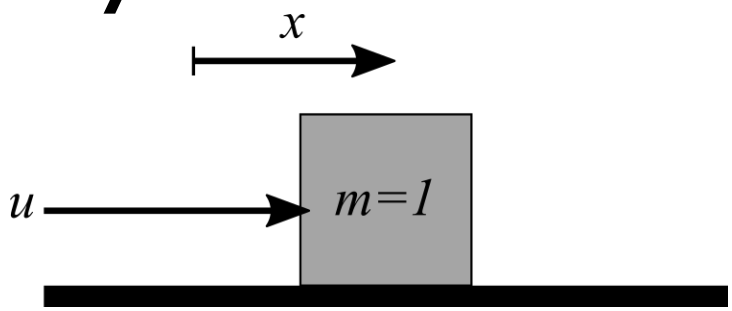
PD control for second order systems

MIP track, week 2

Double integrator: a Simple Dynamic System

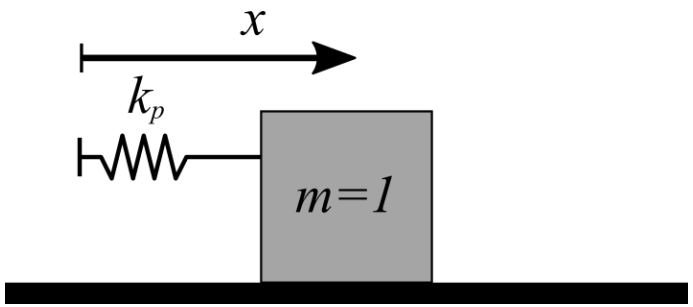


- Newton's law: $\ddot{x} = u$
- Think about *u* as "input", *x* as "output"

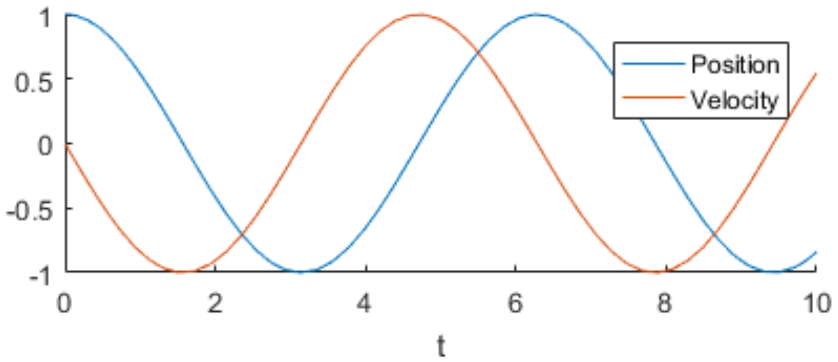
- Assume we want to send $x \rightarrow 0$

Virtual Spring for Control

- Idea: attach a spring between the block and the desired position
- Hooke's law $u_p(x) := k_p(x_{\text{des}} - x)$

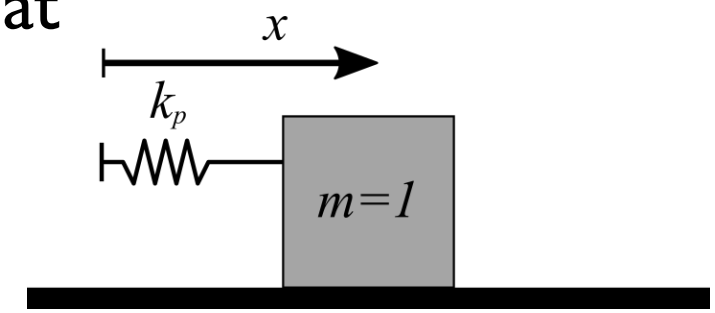


- Try using MATLAB to integrate $\ddot{x} = u_p(x)$
- Change stiffness to approach goal faster
- Overshoot



Dissipation

- Need to remove energy to stop at goal



- Viscous friction $u_d(x, \dot{x}) := -k_d \dot{x}$

- Add to previous input $\ddot{x} = u_p(x) + u_d(x, \dot{x})$

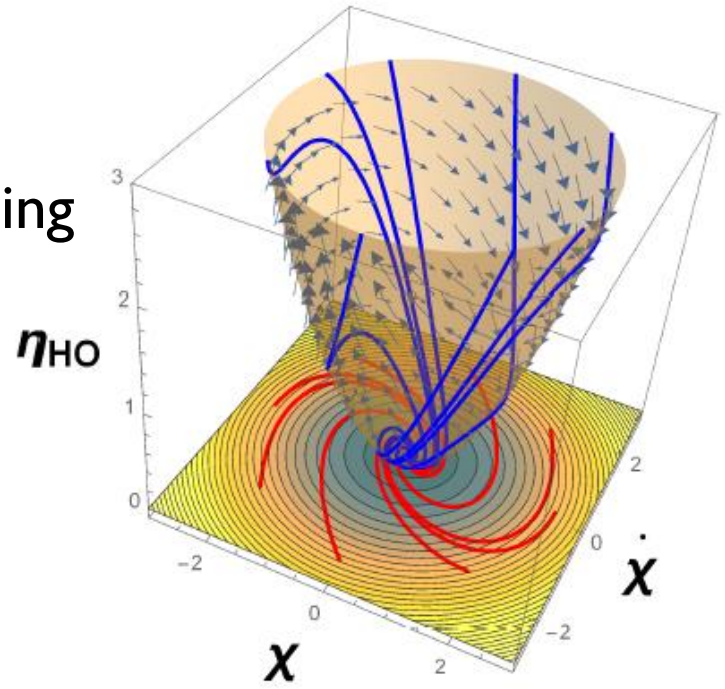
- Try to tune gains to approach without overshoot

Artificial energy landscape (total energy is a Lyapunov function)

- Can *prove* that the PD controller is stabilizing
- Consider "total energy" $\eta(x, \dot{x}) = \frac{1}{2} k_p (x - x_{\text{des}})^2 + \frac{1}{2} \dot{x}^2$
- Take the time derivative, substituting $\ddot{x}$
- Total energy *monotonically decreases* (rolling downhill)
- Have your MATLAB simulation plot

$$\eta(x(t), \dot{x}(t))$$

- PD controller creates this artificial hill



Where is PD control applicable?

- Nonlinear plant: $\ddot{x} = f(x, \dot{x}) + g(x, \dot{x})u$
- g invertible: set $u := g(x, \dot{x})^{-1}(-f(x, \dot{x}) + v)$
- Back to $\ddot{x} = v$
- "Feedback linearization" / "inverse dynamics"
- Implementation difficulties

- Recall hyperbolic approximation (Mobility 1.2)
- PD on nonlinear plants: e.g.

$$\ddot{x} = f(x, \dot{x}) + g(x, \dot{x})u := x^3 + (1 + x^2)u$$