Basic Concepts in Control

393R: Autonomous Robots

Peter Stone

Slides Courtesy of
Benjamin Kuipers

Good Morning Colleagues

- Are there any questions?

Logistics

- Assignment 2 underway
- Next week's readings - due Monday night
  - Forward/inverse kinematics
  - Aibo joint modeling
  - Frame-based control

Controlling a Simple System

- Consider a simple system: $\dot{x} = F(x, u)$
  - Scalar variables *x* and *u*, not vectors **x** and **u**.
  - Assume *x* is observable: $y = G(x) = x$
  - Assume effect of motor command *u*: $\frac{\partial F}{\partial u} > 0$
- The setpoint x_{set} is the desired value.
  - The controller responds to error: $e = x - x_{set}$
- The goal is to set *u* to reach $e = 0$.

The intuition behind control

- Use action u to push back toward error $e = 0$
  - error e depends on state x (via sensors y)
- What does pushing back do?
  - Depends on the structure of the system
  - Velocity versus acceleration control
- How much should we push back?
  - What does the magnitude of u depend on?

Car on a slope example

Velocity or acceleration control?

- If error reflects $\mathbf{x}$, does $\mathbf{u}$ affect $\mathbf{x}'$ or $\mathbf{x}''$?
- Velocity control: $\mathbf{u} \rightarrow \mathbf{x}'$ (valve fills tank)
  - let $\mathbf{x} = (x)$

$$\dot{x} = (\dot{x}) = F(x, u) = (u)$$

- Acceleration control: $\mathbf{u} \rightarrow \mathbf{x}''$ (rocket)
  - let $\mathbf{x} = (x \; v)^T$

$$\dot{x} = \begin{pmatrix} \dot{x} \\ \dot{v} \end{pmatrix} = F(x, u) = \begin{pmatrix} v \\ u \end{pmatrix}$$

$$\dot{v} = \ddot{x} = u$$

The Bang-Bang Controller

- Push back, against the *direction* of the error
  - with constant action u
- Error is $e = x - x_{set}$

$$e < 0 \quad \Rightarrow \quad u := on \quad \Rightarrow \quad \dot{x} = F(x, on) > 0$$

$$e > 0 \quad \Rightarrow \quad u := off \quad \Rightarrow \quad \dot{x} = F(x, off) < 0$$

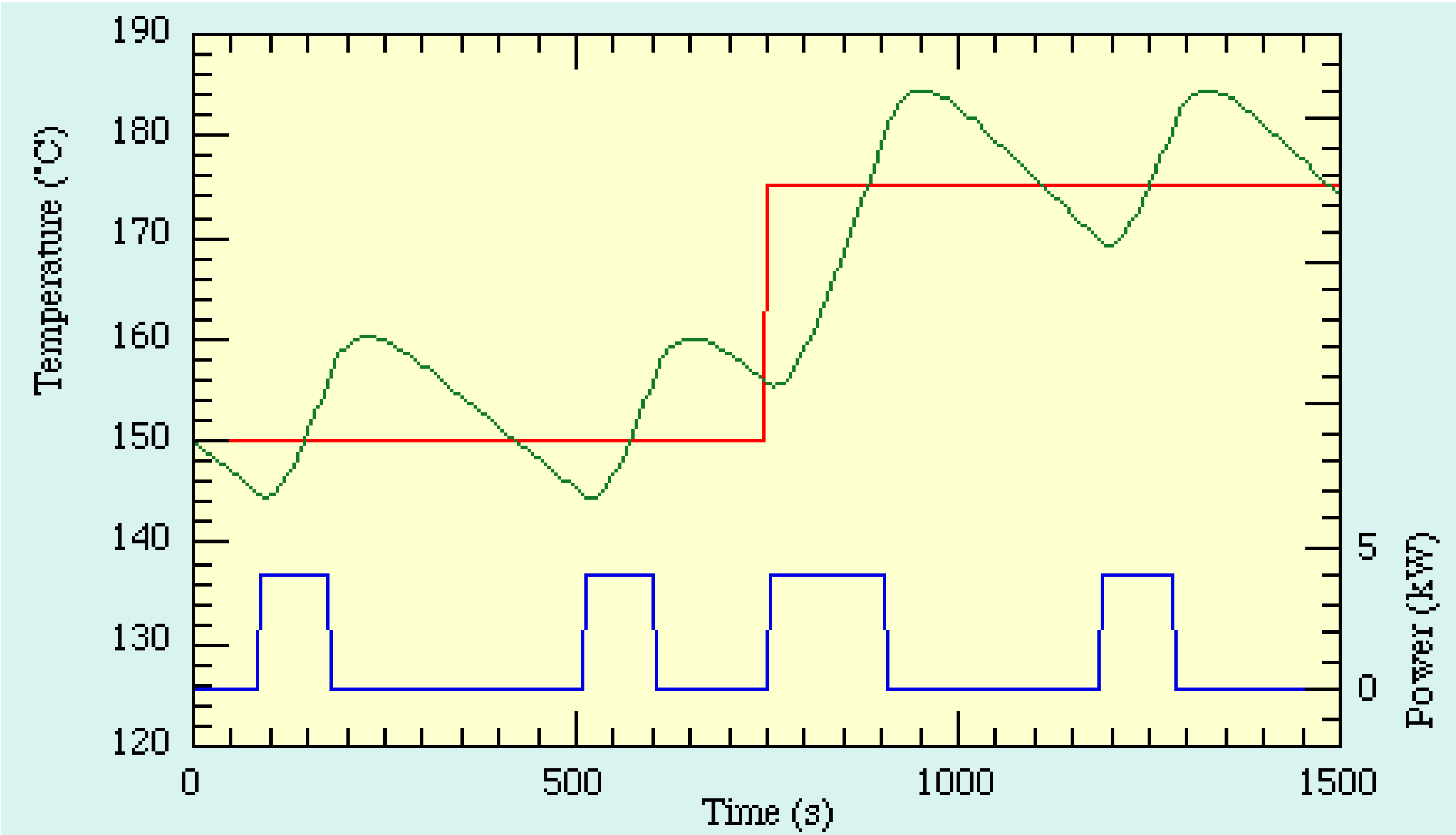
- To prevent chatter around $e = 0$,

$$e < -\varepsilon \quad \Rightarrow \quad u := on$$

$$e > +\varepsilon \quad \Rightarrow \quad u := off$$

- Household thermostat. Not very subtle.

Bang-Bang Control in Action



– Optimal for reaching the setpoint

– Not very good for staying near it

Hysteresis

- Does a thermostat work exactly that way?
  - Car demonstration
- Why not?

- How can you prevent such frequent motor action?
- Nao turning to ball example

Proportional Control

- Push back, *proportional* to the error.

$$u = -ke + u_b$$

  - set u_b so that $\dot{x} = F(x_{set}, u_b) = 0$

- For a linear system, we get exponential convergence.

$$x(t) = Ce^{-\alpha t} + x_{set}$$

- The controller gain k determines how quickly the system responds to error.

Velocity Control

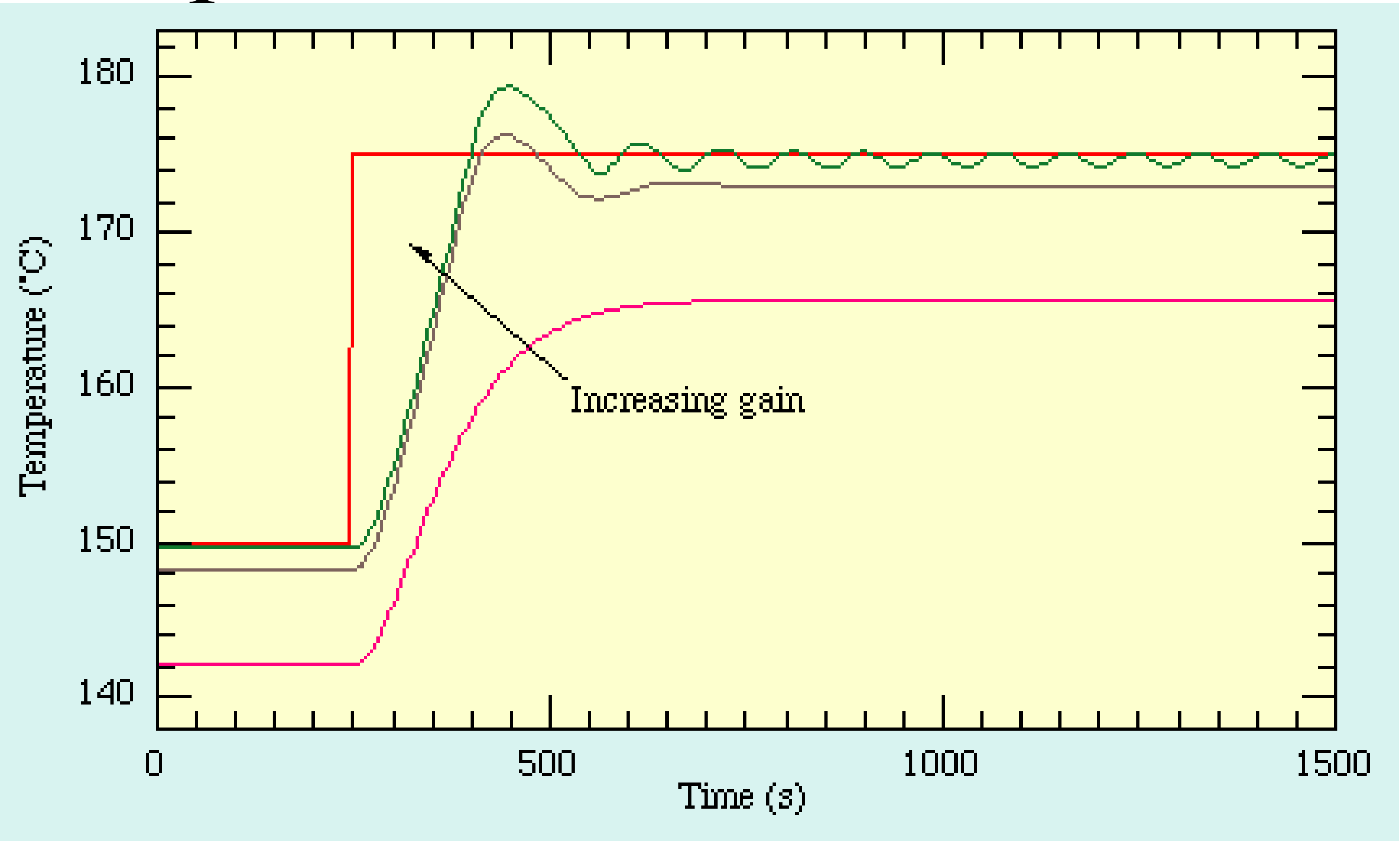
- You want to drive your car at velocity $v_{set.}$
- You issue the motor command $u = pos_{accel}$
- You observe velocity v_{obs}.

- Define a first-order controller:

$$u = -k(v_{obs} - v_{set}) + u_b$$

  – k is the controller gain.

Proportional Control in Action



- Increasing gain approaches setpoint faster
- Can lead to overshoot, and even instability
- Steady-state offset

Steady-State Offset

- Suppose we have continuing disturbances:

$$\dot{x} = F(x, u) + d$$

- The P-controller cannot stabilize at $e = 0$.
  - Why not?

Steady-State Offset

- Suppose we have continuing disturbances:

$$\dot{x} = F(x, u) + d$$

- The P-controller cannot stabilize at $e = 0$.
  - if u_b is defined so $F(x_{set}, u_b) = 0$
  - then $F(x_{set}, u_b) + d \neq 0$, so the system changes
- Must adapt u_b to different disturbances d.

Adaptive Control

- Sometimes one controller isn't enough.
- We need controllers at different time scales.

$$u = -k_P e + u_b$$

$$\dot{u}_b = -k_I e \text{ where } k_I << k_P$$

- This can eliminate steady-state offset.
  - Why?

Adaptive Control

- Sometimes one controller isn't enough.
- We need controllers at different time scales.

$$u = -k_P e + u_b$$

$$\dot{u}_b = -k_I e \text{ where } k_I << k_P$$

- This can eliminate steady-state offset.
  - Because the slower controller adapts u_b.

Integral Control

- The adaptive controller $\dot{u}_b = -k_I e$ means

$$u_b(t) = -k_I \int_0^t e\,dt + u_b$$

- Therefore

$$u(t) = -k_P e(t) - k_I \int_0^t e\,dt + u_b$$

- The Proportional-Integral (PI) Controller.

Nonlinear P-control

- Generalize proportional control to

  $$u = -f(e) + u_b \quad \text{where} \quad f \in M_0^+$$

- Nonlinear control laws have advantages
  - f has vertical asymptote: bounded error e
  - f has horizontal asymptote: bounded effort u
  - Possible to converge in finite time.
  - Nonlinearity allows more kinds of composition.

Stopping Controller

- Desired stopping point: $x=0$.
  - Current position: x
  - Distance to obstacle: $d=|x|+\varepsilon$
- Simple P-controller: $v=\dot{x}=-f(x)$
- Finite stopping time for $f(x)=k\sqrt{|x|}\operatorname{sgn}(x)$

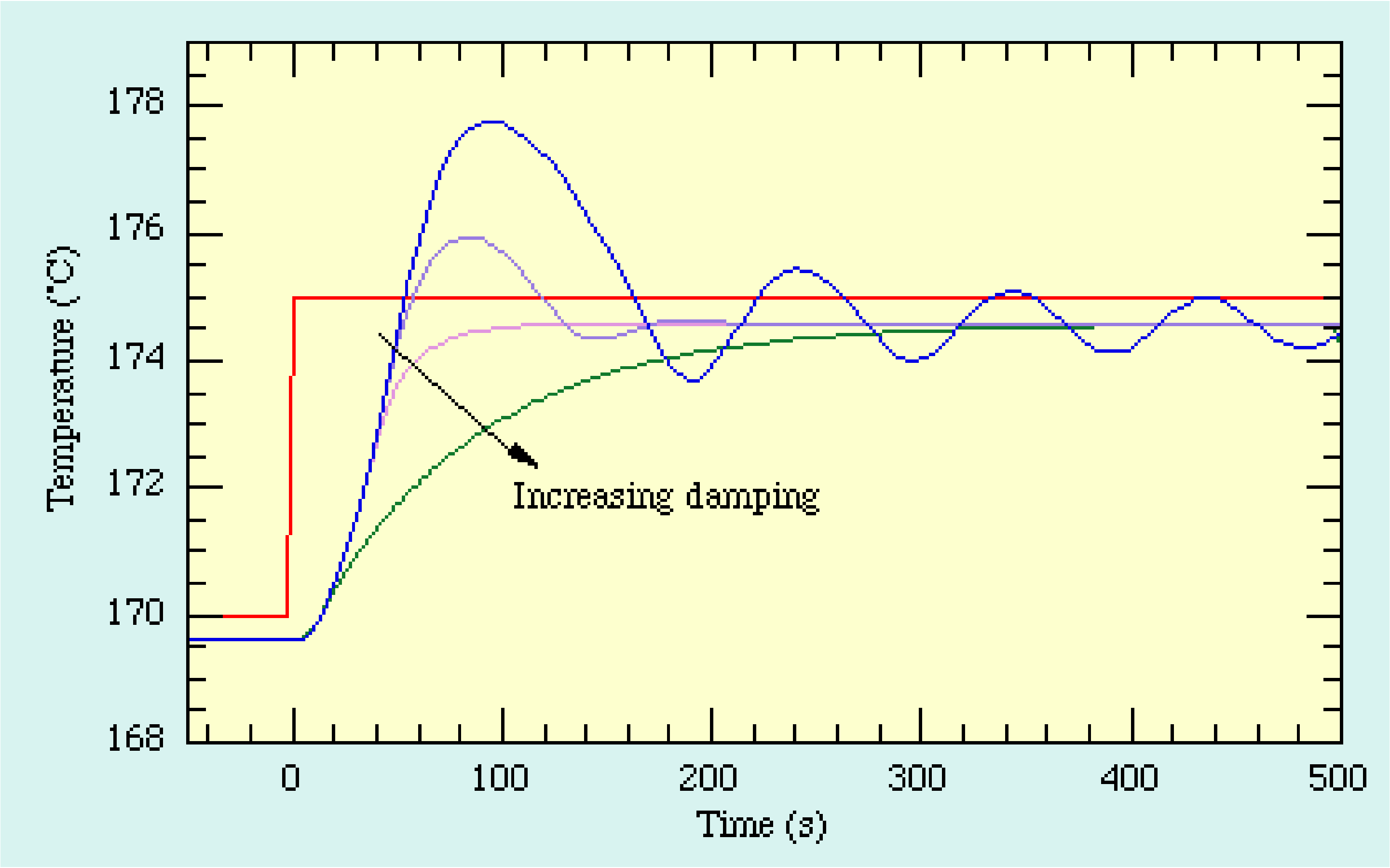
Derivative Control

- Damping friction is a force opposing motion, proportional to velocity.
- Try to prevent overshoot by damping controller response.

$$u = -k_P e - k_D \dot{e}$$

- Estimating a derivative from measurements is fragile, and amplifies noise.

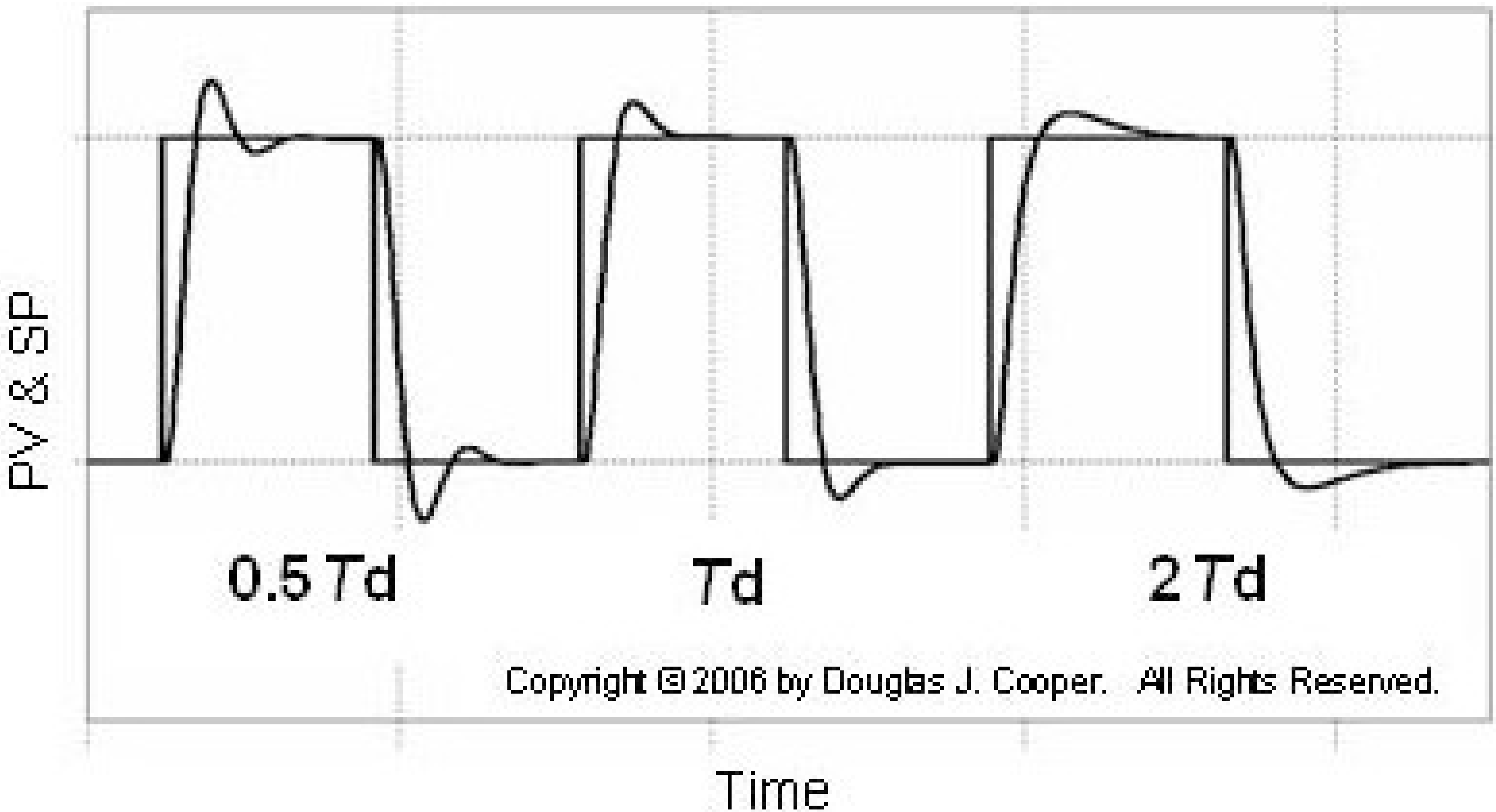
Derivative Control in Action



– Damping fights oscillation and overshoot

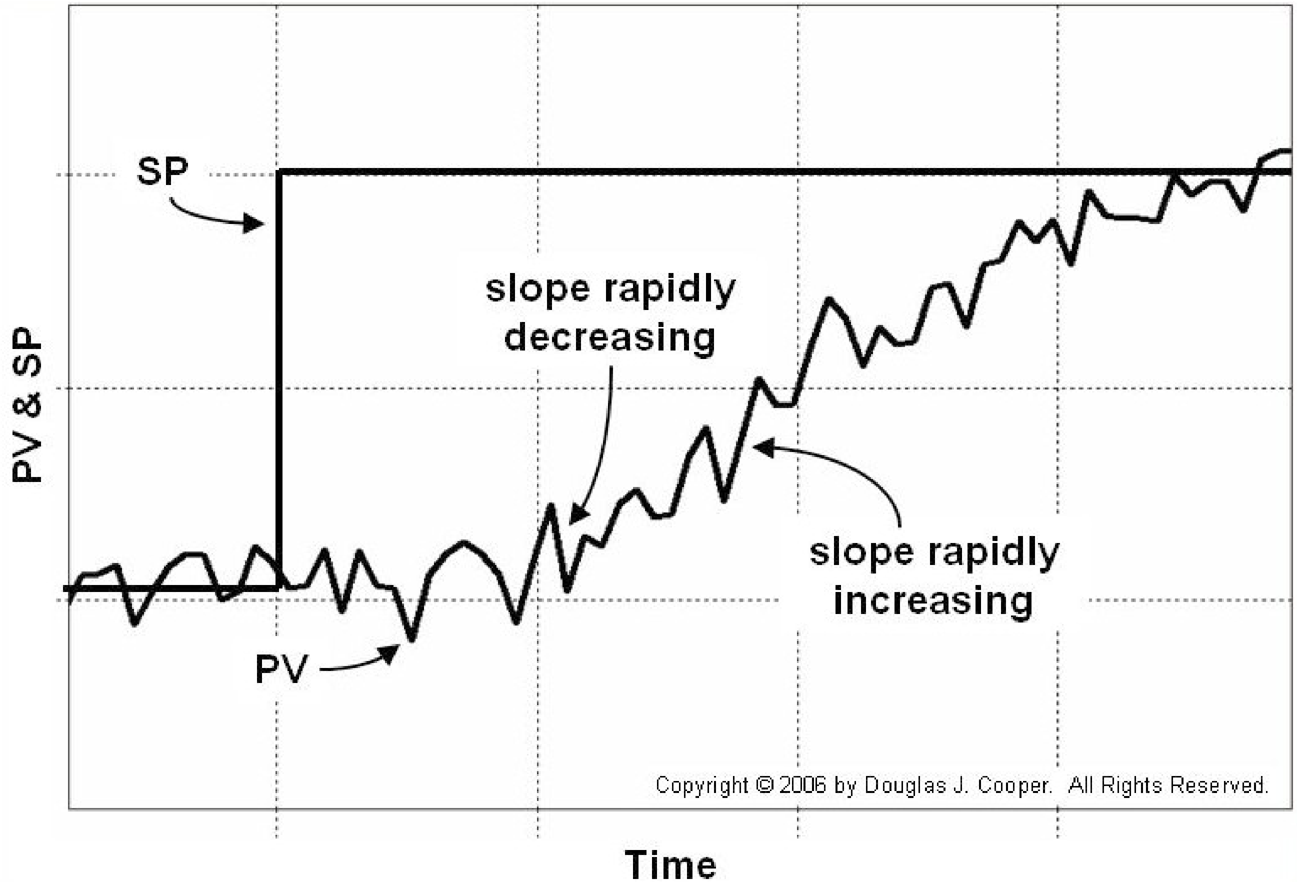
– But it's vulnerable to noise

Effect of Derivative Control



– Different amounts of damping (without noise)

Derivatives Amplify Noise



– This is a problem if control output (CO) depends on slope (with a high gain).

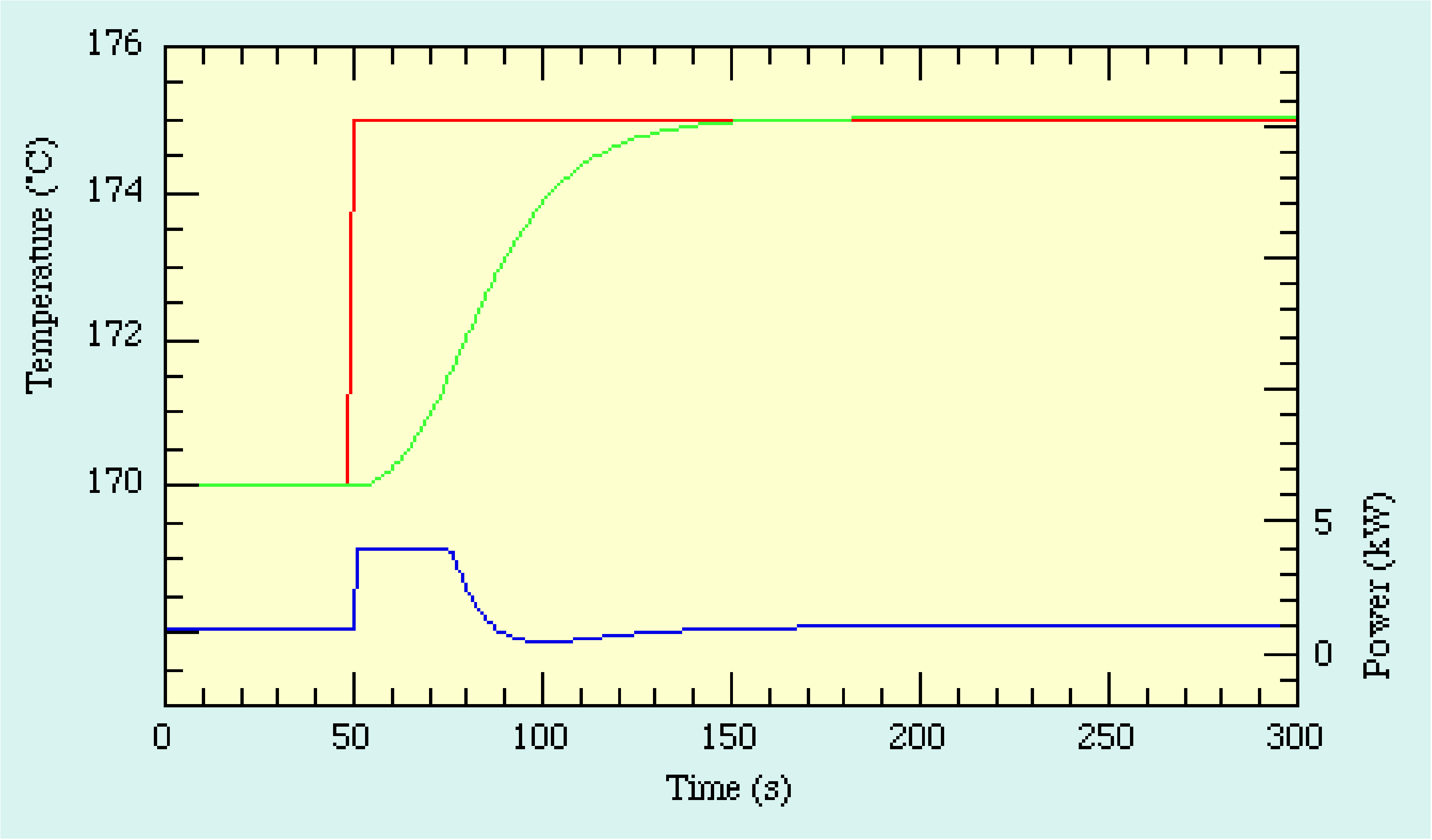
The PID Controller

- A weighted combination of Proportional, Integral, and Derivative terms.

$$u(t) = -k_P\, e(t) - k_I \int_0^t e\,dt - k_D\, \dot{e}(t)$$

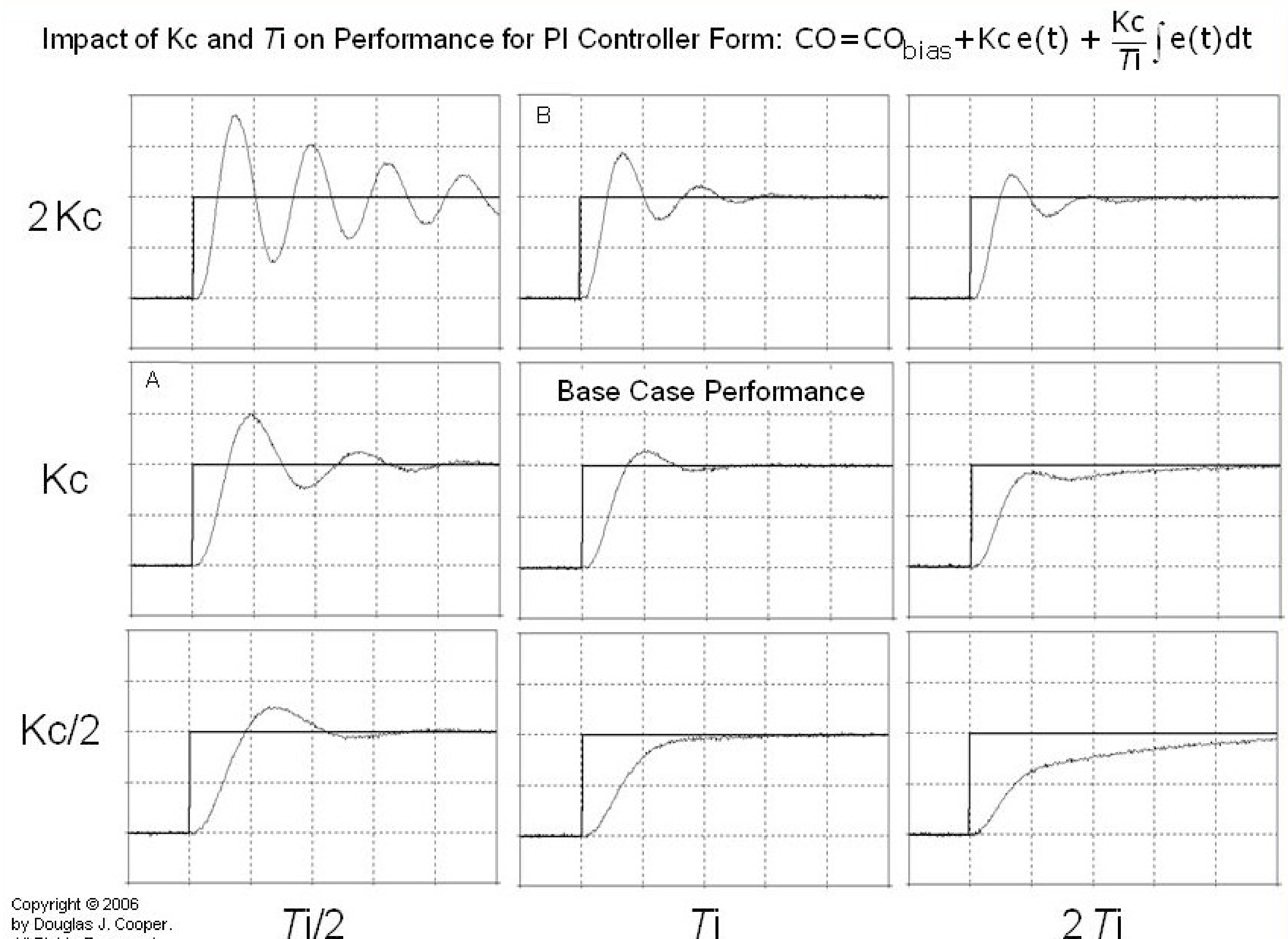
- The PID controller is the workhorse of the control industry. Tuning is non-trivial.
  - End of slides includes some tuning methods.

PID Control in Action



- But, good behavior depends on good tuning!
- Nao joints use PID control

Exploring PI Control Tuning



Habituation

- Integral control adapts the bias term u_b.
- Habituation adapts the setpoint x_{set}.
  - It prevents situations where too much control action would be dangerous.
- Both adaptations reduce steady-state error.

$$u = -k_P e + u_b$$

$$\dot{x}_{set} = +k_h e \text{ where } k_h << k_P$$

Types of Controllers

- **Open-loop control**
  - No sensing
- **Feedback control** (closed-loop)
  - Sense error, determine control response.
- **Feedforward control** (closed-loop)
  - Sense disturbance, predict resulting error, respond to predicted error before it happens.
- **Model-predictive control** (closed-loop)
  - Plan trajectory to reach goal.
  - Take first step.
  - Repeat.

Design open and closed-loop controllers for me to get out of the room.

Dynamical Systems

- A *dynamical system* changes continuously (almost always) according to

$$\dot{x} = F(x) \text{ where } x \text{ ÎÂ}^{n}$$

- A *controller* is defined to change the coupled robot and environment into a desired dynamical system.

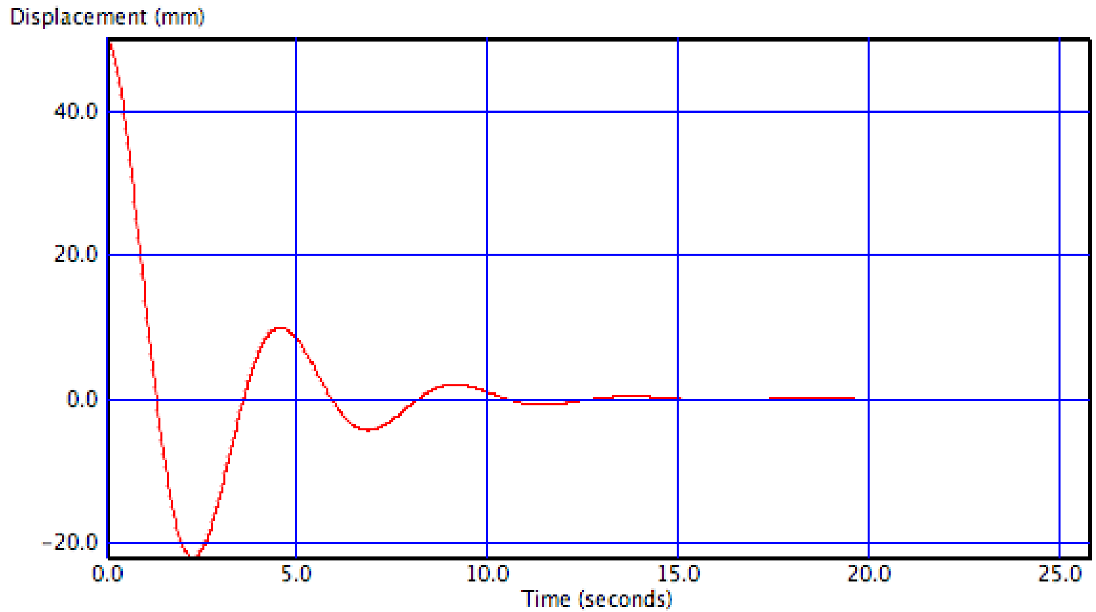
$$\dot{x} = F(x, u)$$
$$y = G(x)$$
$$u = H_i(y)$$

$$\dot{x} = F(x, H_i(G(x)))$$

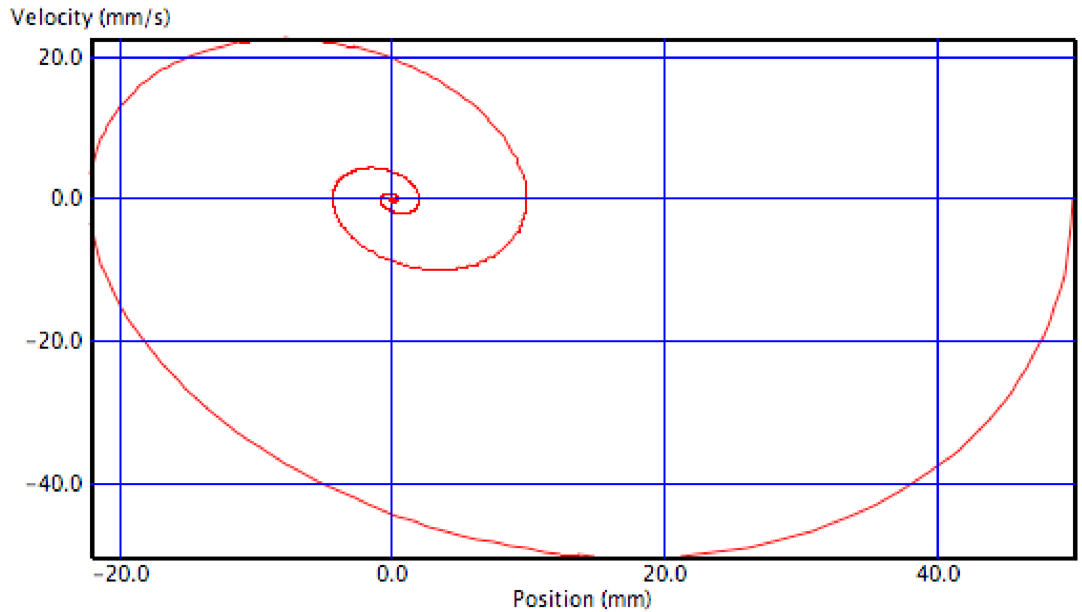
$$\dot{x} = F(x)$$

Two views of dynamic behavior

- Time plot (t,x)

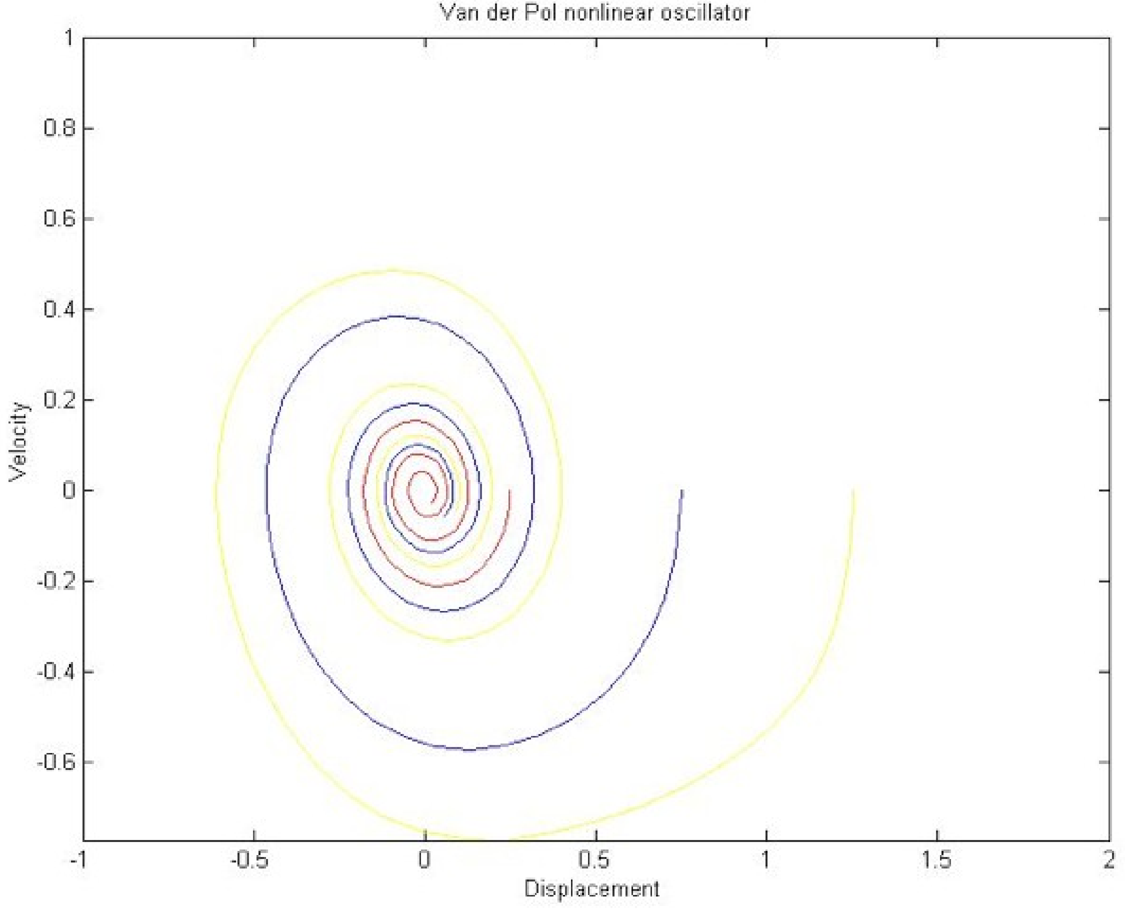


- Phase portrait (x,v)



Phase Portrait: (x,v) space

- Shows the trajectory $(x(t),v(t))$ of the system
  - Stable attractor here



In One Dimension

- Simple linear system

  $$\dot{x} = kx$$

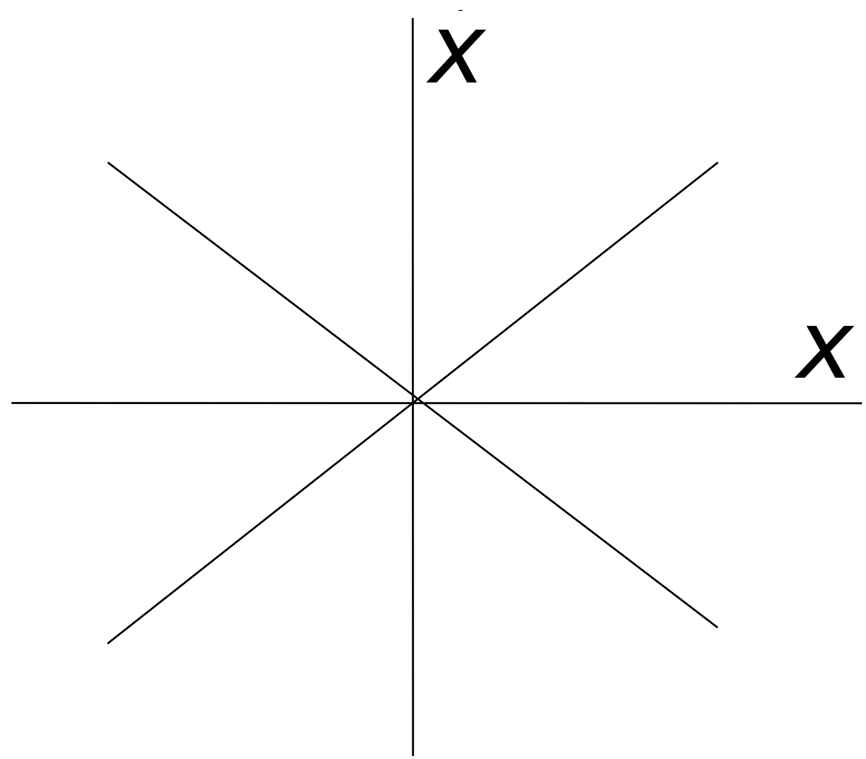
- Fixed point

  $$x = 0 \quad \Rightarrow \quad \dot{x} = 0$$

- Solution

  $$x(t) = x_0 e^{kt}$$

  - Stable if $k < 0$
  - Unstable if $k > 0$



In Two Dimensions

- Often, we have position and velocity:

$$\mathbf{x} = (x, v)^T \quad \text{where } v = \dot{x}$$

- If we model actions as forces, which cause acceleration, then we get:

$$\dot{x} = \begin{pmatrix} \dot{x} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \ddot{x} \end{pmatrix} = \begin{pmatrix} v \\ forces \end{pmatrix}$$

The Damped Spring

- The spring is defined by Hooke's Law:

$$F = ma = m\ddot{x} = \quad k_1 x$$

- Include damping friction

$$m\ddot{x} = \quad k_1 x \quad k_2 \dot{x}$$

- Rearrange and redefine constants

$$\ddot{\xi} + \beta\dot{\xi} + \chi\xi = 0$$

$$\dot{x} = \begin{pmatrix} \dot{x} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \ddot{x} \end{pmatrix} = \begin{pmatrix} v \\ -b\,\dot{x} - cx \end{pmatrix}$$

Node Behavior

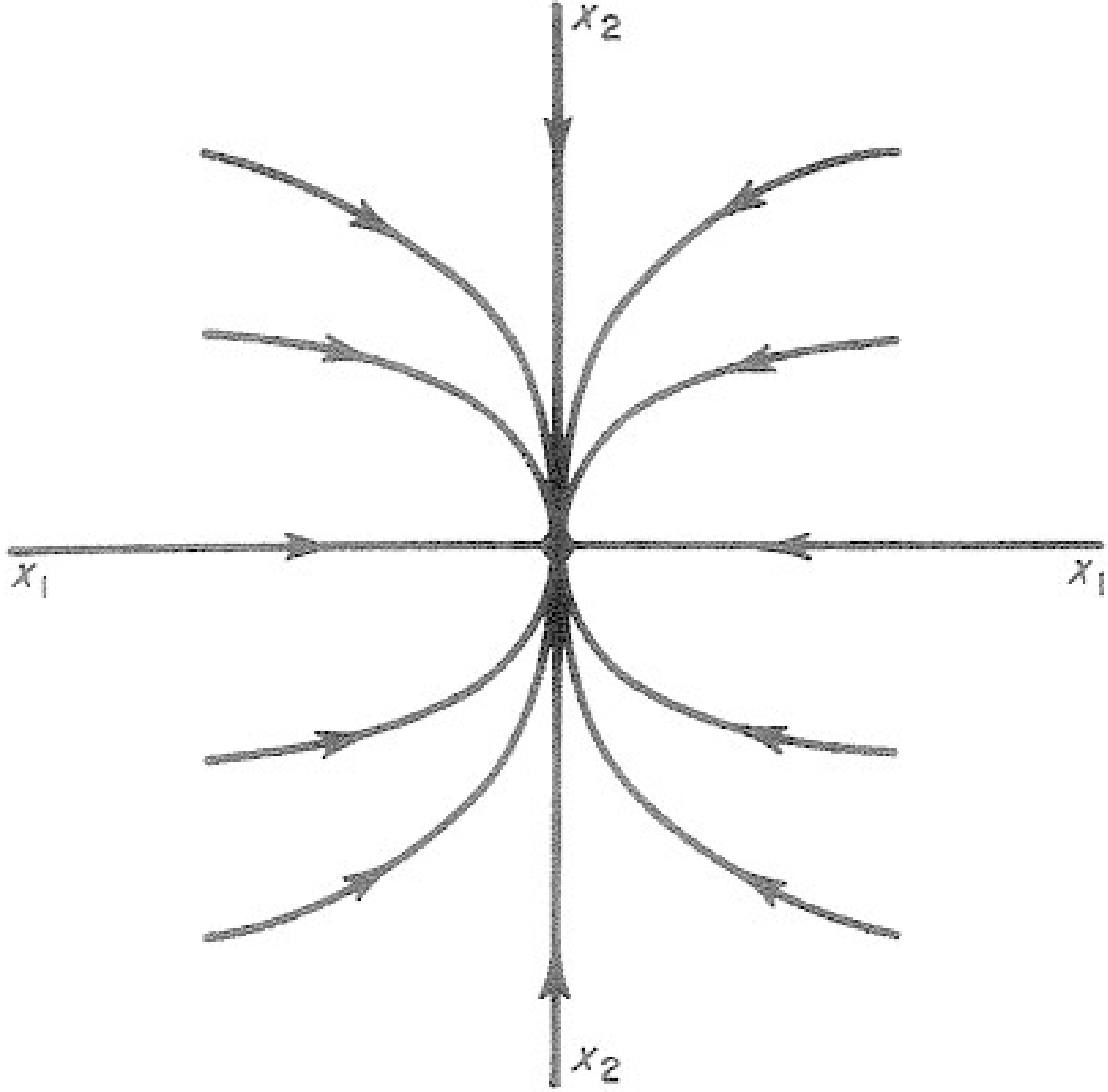


FIG. C. Node: $B = \begin{bmatrix} \lambda & 0 \\ 0 & \mu \end{bmatrix}, \lambda < \mu < 0.$

Focus Behavior

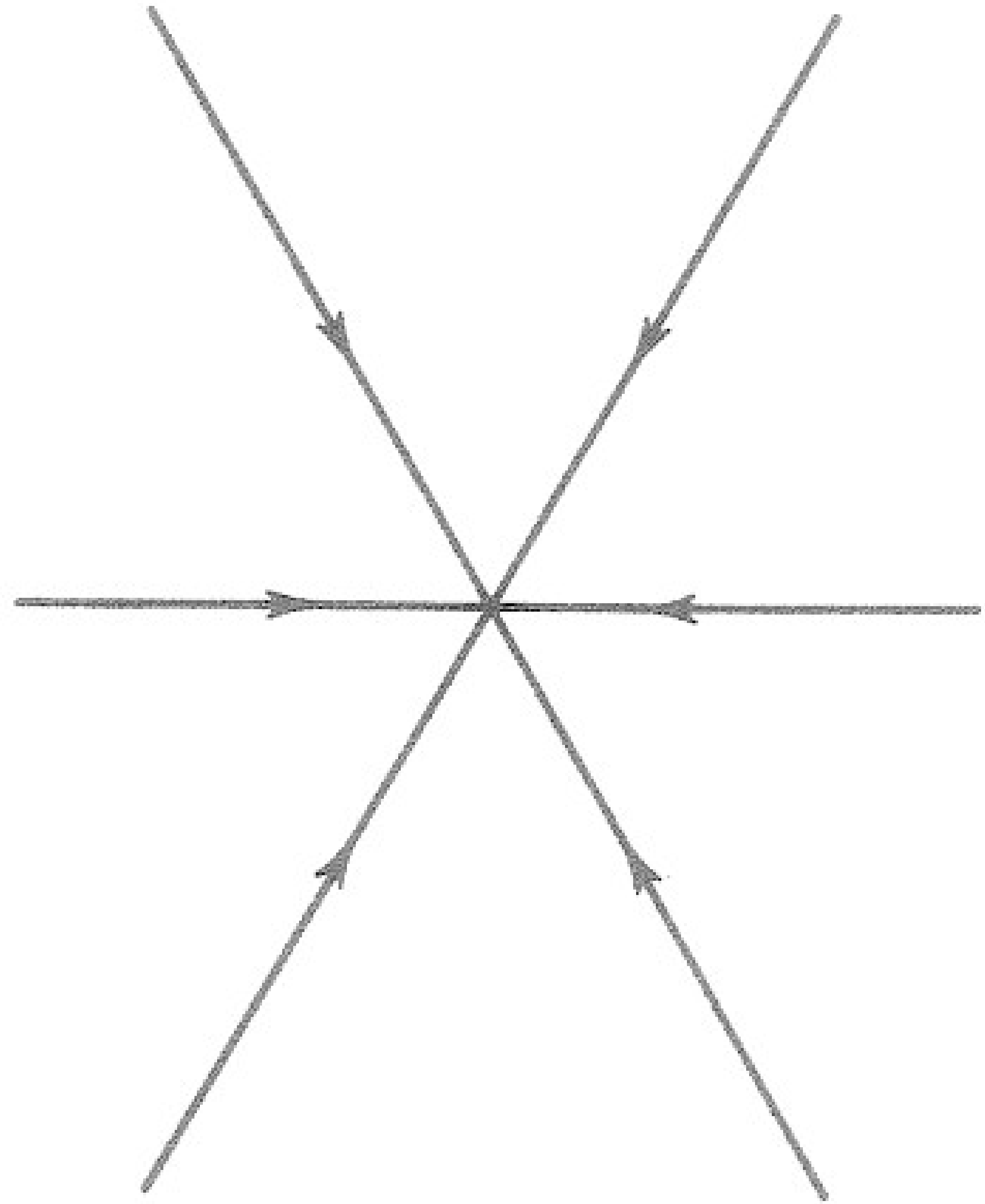


FIG. B. Focus: $B = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$, $\lambda < 0$.

Saddle Behavior

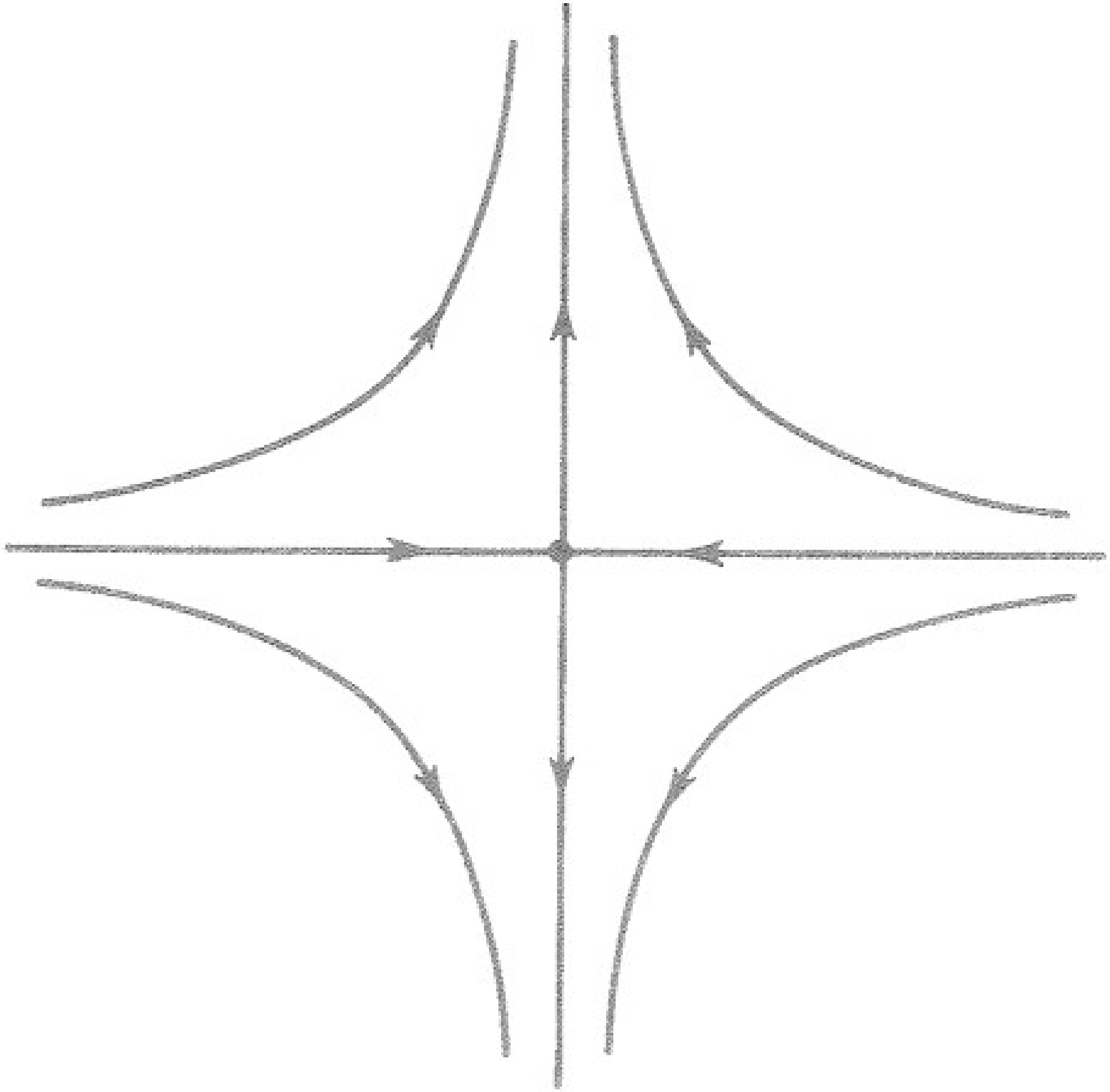


FIG. A. Saddle: $B = \begin{bmatrix} \lambda & 0 \\ 0 & \mu \end{bmatrix}$, $\lambda < 0 < \mu$.

Spiral Behavior

(stable attractor)

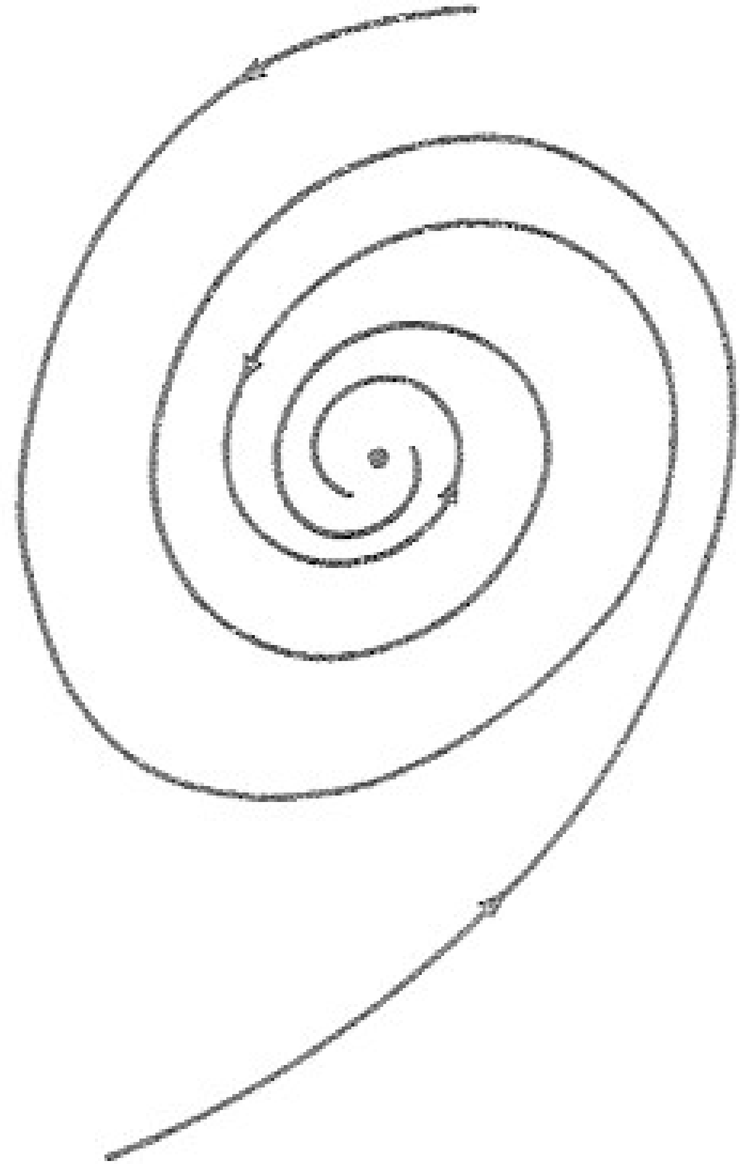


FIG. E. Spiral sink: $B = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$, $b > 0 > a$.

Center Behavior

(undamped oscillator)

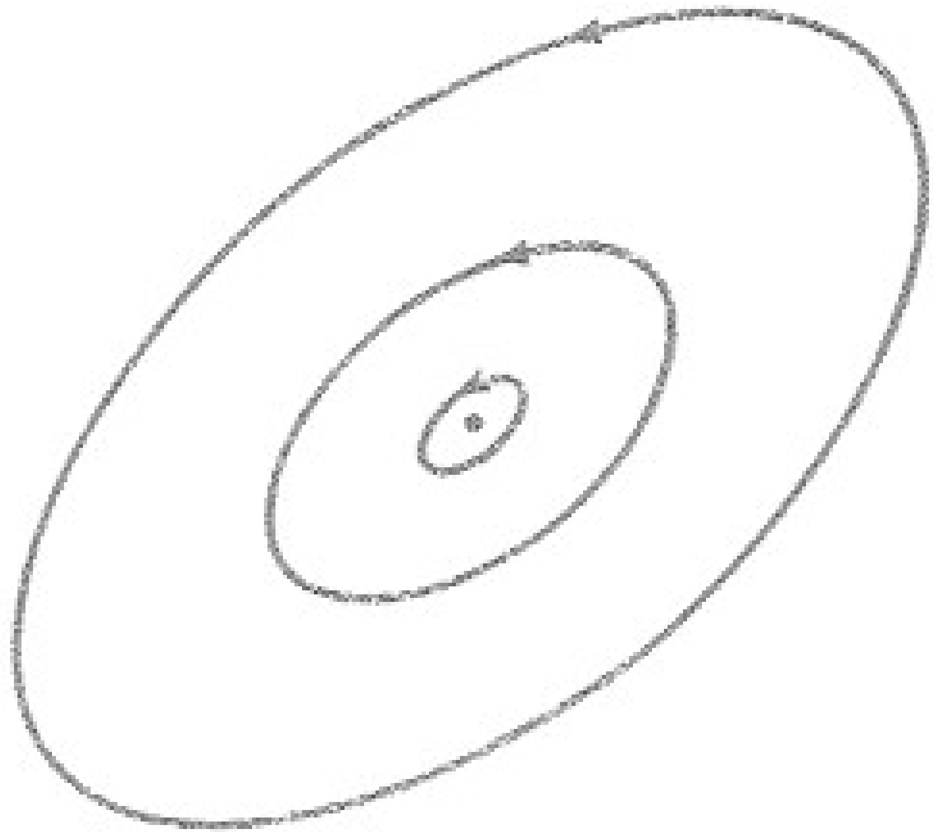
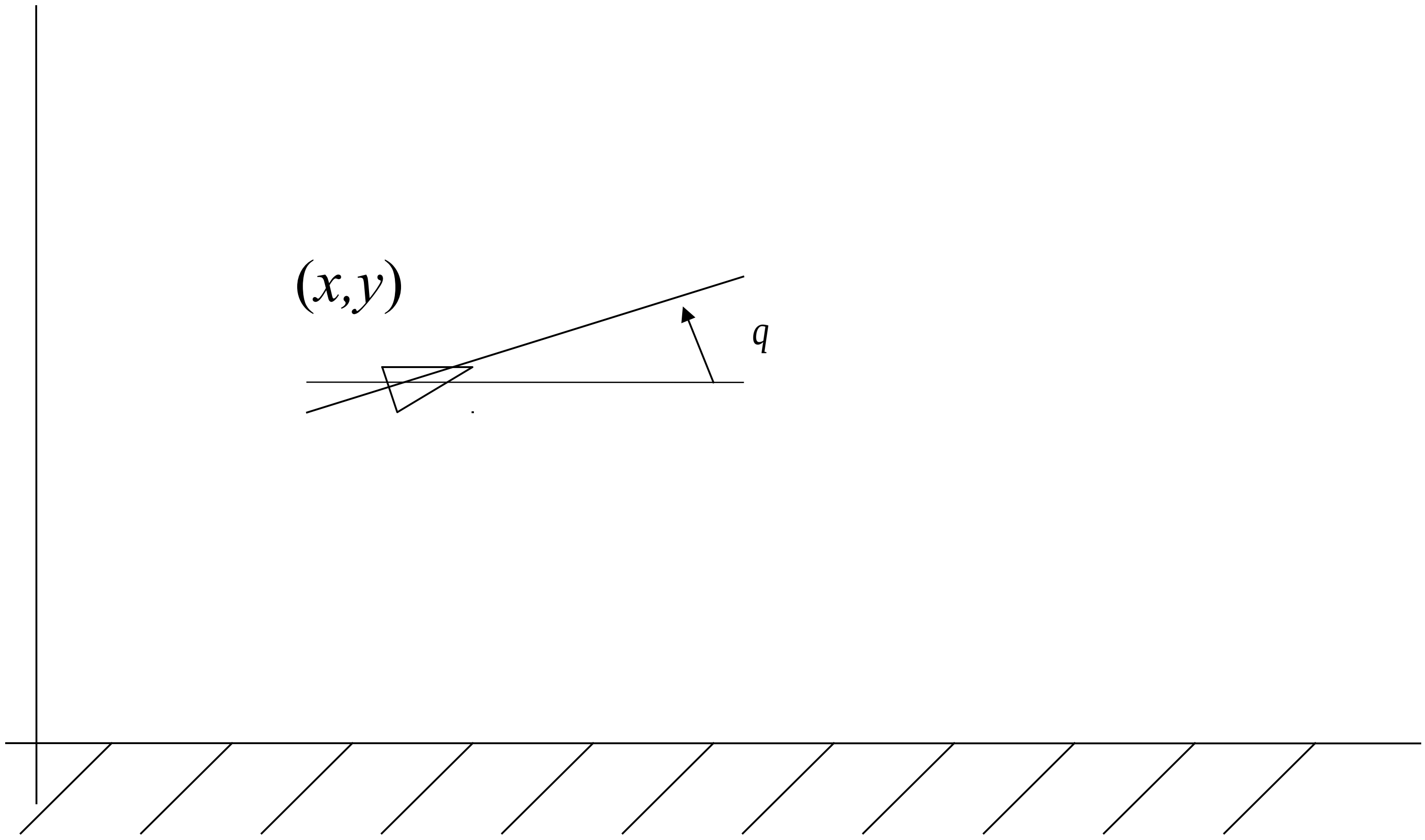


FIG. F. Center: $B = \begin{bmatrix} 0 & -b \\ b & 0 \end{bmatrix}, b > 0.$

The Wall Follower



The Wall Follower

- Our robot model:

$$\dot{x} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{q} \end{pmatrix} = F(x, u) = \begin{pmatrix} v \cos q \\ v \sin q \\ w \end{pmatrix}$$

$$\mathbf{u} = (v \;\; \omega)^{\mathrm{T}} \qquad \mathbf{y} = (y \;\; \theta)^{\mathrm{T}} \qquad \theta \approx 0.$$

- We set the control law $\mathbf{u} = (v \;\; \omega)^{\mathrm{T}} = H_i(\mathbf{y})$

The Wall Follower

- Assume constant forward velocity $v = v_0$
  - approximately parallel to the wall: $\theta \approx 0$.
- Desired distance from wall defines error:

  $e = y - y_{set}$ so { $\dot{e} = \dot{y}$ and { $\ddot{e}$ ¿ $= \ddot{y}$ ¿

- We set the control law $\mathbf{u} = (v \;\; \omega)^T = H_i(\mathbf{y})$
  - We want e to act like a "damped spring"

$$\ddot{e} + k_1 \dot{e} + k_2 e = 0$$

The Wall Follower

- We want a damped spring: $\ddot{e} + k_1 \dot{e} + k_2 e = 0$
- For small values of θ

$$\dot{e} \quad ¿ \quad \dot{y} \quad ¿ \quad v \sin q \quad » \quad vq$$
$$\ddot{e} \quad ¿ \quad \ddot{y} \quad ¿ \quad v \cos q \, \dot{q} \quad » \quad vw$$

- Substitute, and assume $v = v_0$ is constant.

$$v_0 w + k_1 v_0 q + k_2 e = 0$$

- Solve for ω

The Wall Follower

- To get the damped spring $\ddot{e}+k_1\dot{e}+k_2 e=0$
- We get the constraint

$$v_0 w+k_1 v_0 q+k_2 e=0$$

- Solve for ω. Plug into **u**.

$$u=\begin{pmatrix} v \\ w \end{pmatrix}=\begin{pmatrix} v_0 \\ -k_1 q-\frac{k_2}{v_0}e \end{pmatrix}=H_i(e,q)$$

  – This makes the wall-follower a **PD** controller.
  – Because:

Tuning the Wall Follower

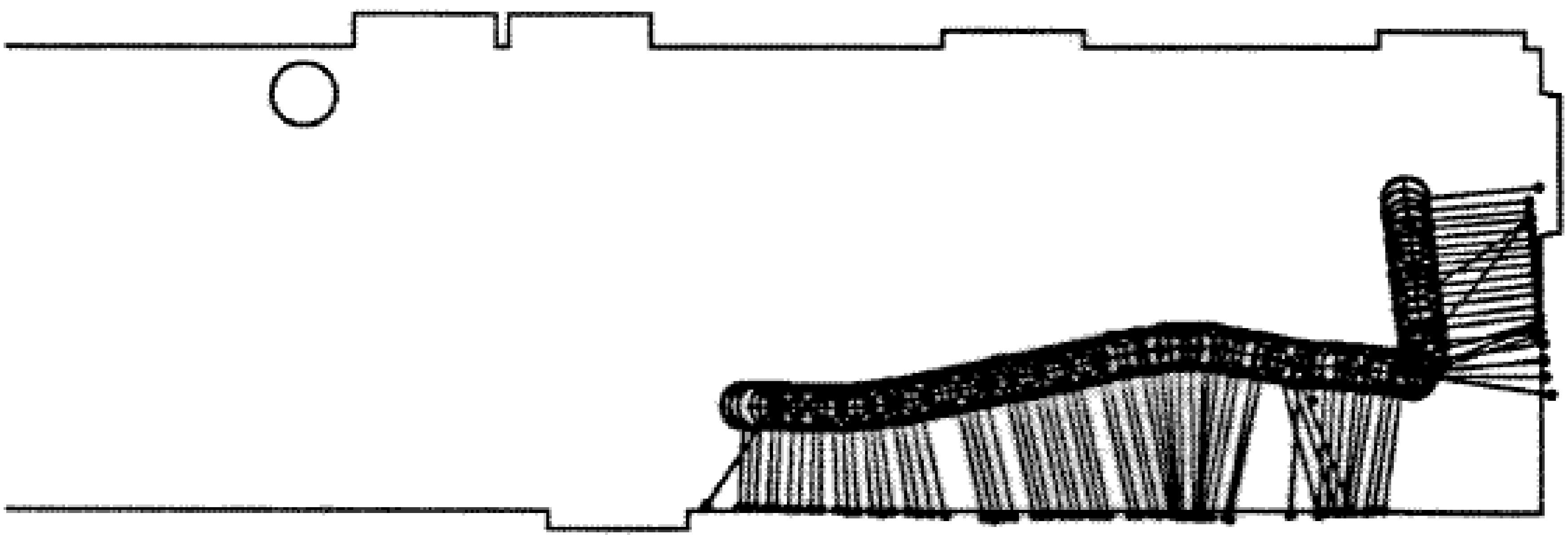
- The system is $\ddot{e} + k_1 \dot{e} + k_2 e = 0$
- Critical damping requires $k_1^2 - 4k_2 = 0$

$$k_1 = \sqrt{4k_2}$$

- Slightly underdamped performs better.
  - Set k_2 by experience.
  - Set k_1 a bit less than $\sqrt{4k_2}$

An Observer for Distance to Wall

- Short sonar returns are reliable.
  - They are likely to be perpendicular reflections.



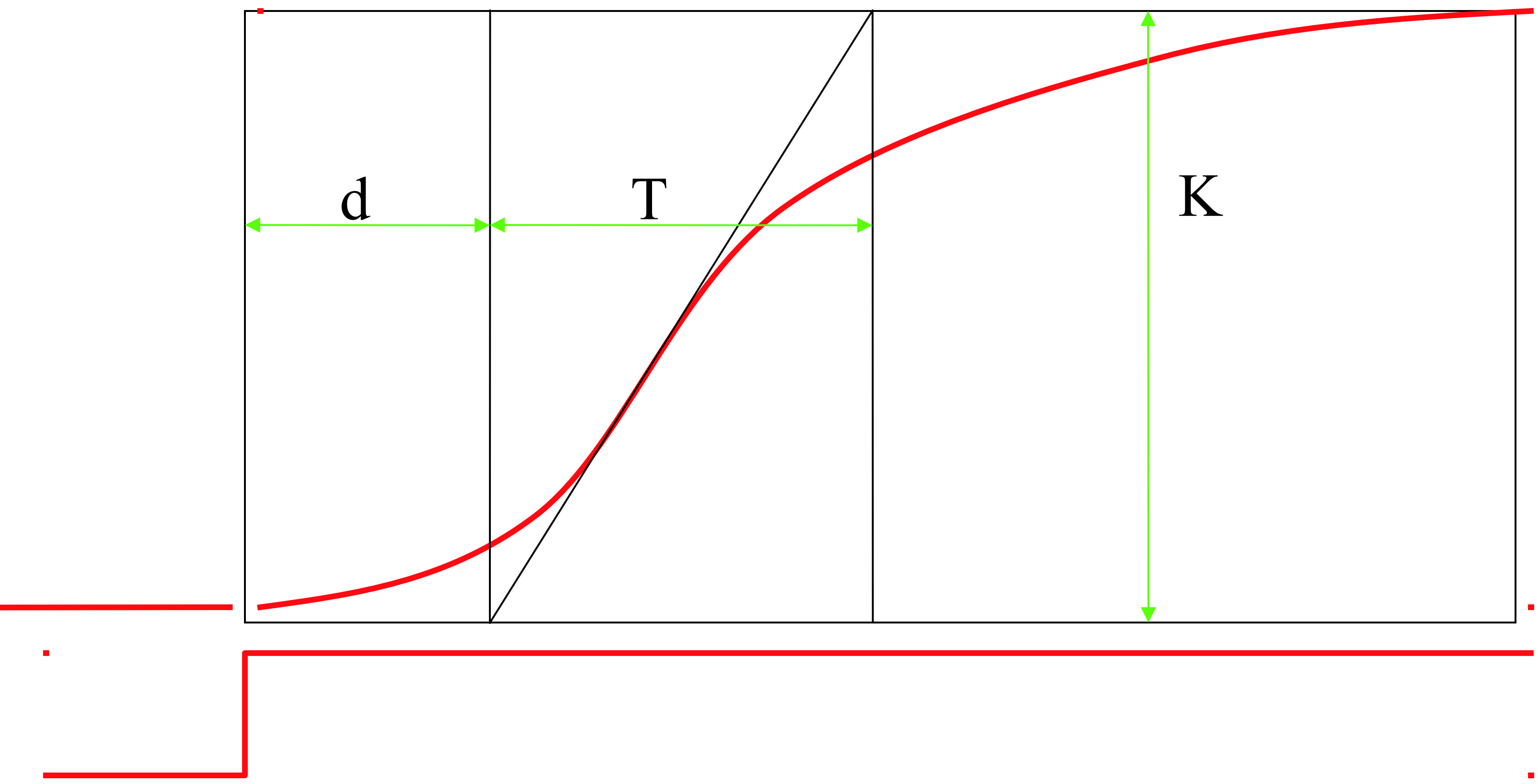
Alternatives

- The wall follower is a PD control law.
- A target seeker should probably be a PI control law, to adapt to motion.

- Can try different tuning values for parameters.
  - This is a simple model.
  - Unmodeled effects might be significant.

Ziegler-Nichols Tuning

- Open-loop response to a unit step increase.
  - d is deadtime. T is the process time constant.
  - K is the process gain.



Tuning the PID Controller

- We have described it as:

$$u(t) = -k_P e(t) - k_I \int_0^t e\,dt - k_D \dot{e}(t)$$

- Another standard form is:

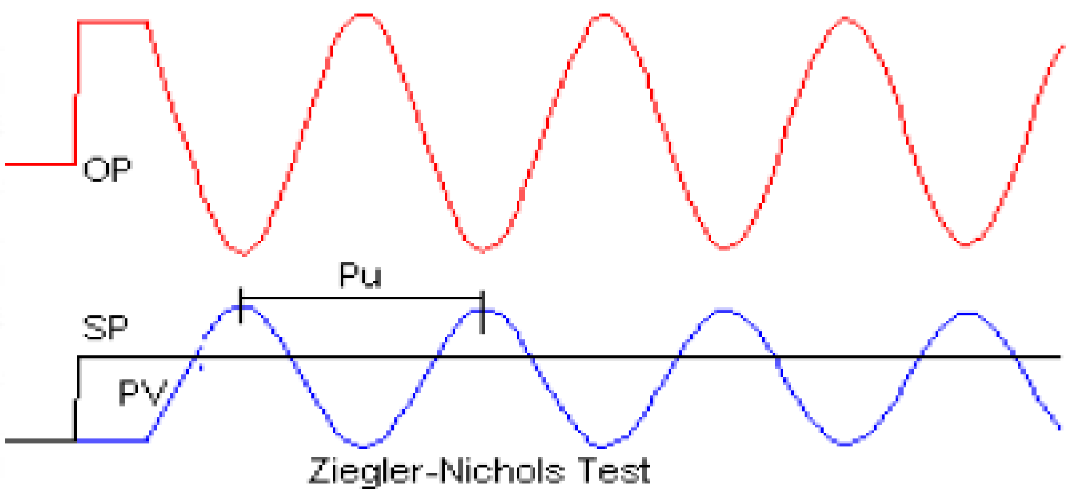
$$u(t) = -P\left[e(t) + T_I \int_0^t e\,dt + T_D \dot{e}(t)\right]$$

- Ziegler-Nichols says:

$$P = \frac{1.5 \cdot T}{K \cdot d} \quad T_I = 2.5 \cdot d \quad T_D = 0.4 \cdot d$$

Ziegler-Nichols Closed Loop

1. Disable D and I action (pure P control).
2. Make a step change to the setpoint.
3. Repeat, adjusting controller gain until achieving a stable oscillation.
   - This gain is the “ultimate gain” K_u.
   - The period is the “ultimate period” P_u.



Closed-Loop Z-N PID Tuning

- A standard form of PID is:

$$u(t) = -P\left[e(t) + T_I \int_0^t e\,dt + T_D \dot{e}(t)\right]$$

- For a PI controller:

$$P = 0.45 \cdot K_u \quad T_I = \frac{P_u}{1.2}$$

- For a PID controller:

$$P = 0.6 \cdot K_u \quad T_I = \frac{P_u}{2} \quad T_D = \frac{P_u}{8}$$

Summary of Concepts

- Dynamical systems and phase portraits
- Qualitative types of behavior
  - Stable vs unstable; nodal vs saddle vs spiral
  - Boundary values of parameters
- Designing the wall-following control law
- Tuning the PI, PD, or PID controller
  - Ziegler-Nichols tuning rules
  - For more, Google: controller tuning

Followers

- A follower is a control law where the robot moves forward while keeping some error term small.
  - Open-space follower
  - Wall follower
  - Coastal navigator
  - Color follower

Control Laws Have Conditions

- Each control law includes:
  - A *trigger*: Is this law applicable?
  - The law itself: $\mathbf{u} = H_i(\mathbf{y})$
  - A *termination condition*: Should the law stop?

Open-Space Follower

- Move in the direction of large amounts of open space.
- Wiggle as needed to avoid specular reflections.
- Turn away from obstacles.
- Turn or back out of blind alleys.

Wall Follower

- Detect and follow right or left wall.
- PD control law.
- Tune to avoid large oscillations.
- Terminate on obstacle or wall vanishing.

Coastal Navigator

- Join wall-followers to follow a complex "coastline"
- When a wall-follower terminates, make the appropriate turn, detect a new wall, and continue.
- Inside and outside corners, 90 and 180 deg.
- Orbit a box, a simple room, or the desks.

Color Follower

- Move to keep a desired color centered in the camera image.
- Train a color region from a given image.
- Follow an orange ball on a string, or a brightly-colored T-shirt.

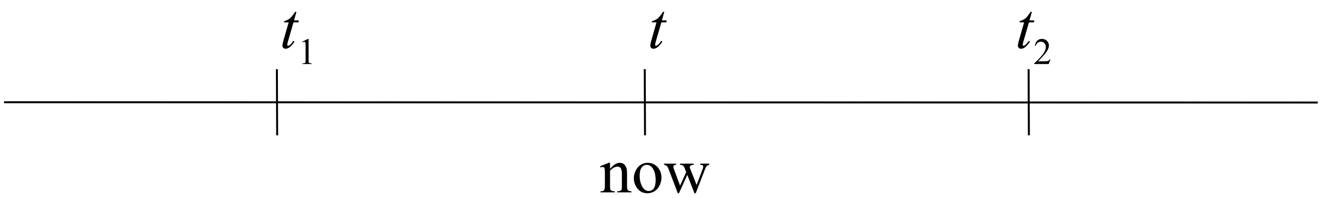
Problems and Solutions

- Time delay
- Static friction
- Pulse-width modulation
- Integrator wind-up
- Chattering
- Saturation, dead-zones, backlash
- Parameter drift

Unmodeled Effects

- Every controller depends on its simplified model of the world.
  - Every model omits almost everything.
- If unmodeled effects become significant, the controller's model is wrong,
  - so its actions could be seriously wrong.
- Most controllers need special case checks.
  - Sometimes it needs a more sophisticated model.

Time Delay



- At time t,
  - Sensor data tells us about the world at $t_1 < t$.
  - Motor commands take effect at time $t_2 > t$.
  - The lag is $dt = t_2 - t_1$.
- To compensate for lag time,
  - Predict future sensor value at t_2.
  - Specify motor command for time t_2.

Predicting Future Sensor Values

- Later, *observers* will help us make better predictions.
- Now, use a simple prediction method:
  - If sensor s is changing at rate ds/dt,
  - At time t, we get $s(t_1)$, where $t_1 < t$,
  - Estimate $s(t_2) = s(t_1) + ds/dt * (t_2 - t_1)$.
- Use $s(t_2)$ to determine motor signal $u(t)$ that will take effect at t_2.

Static Friction ("Stiction")

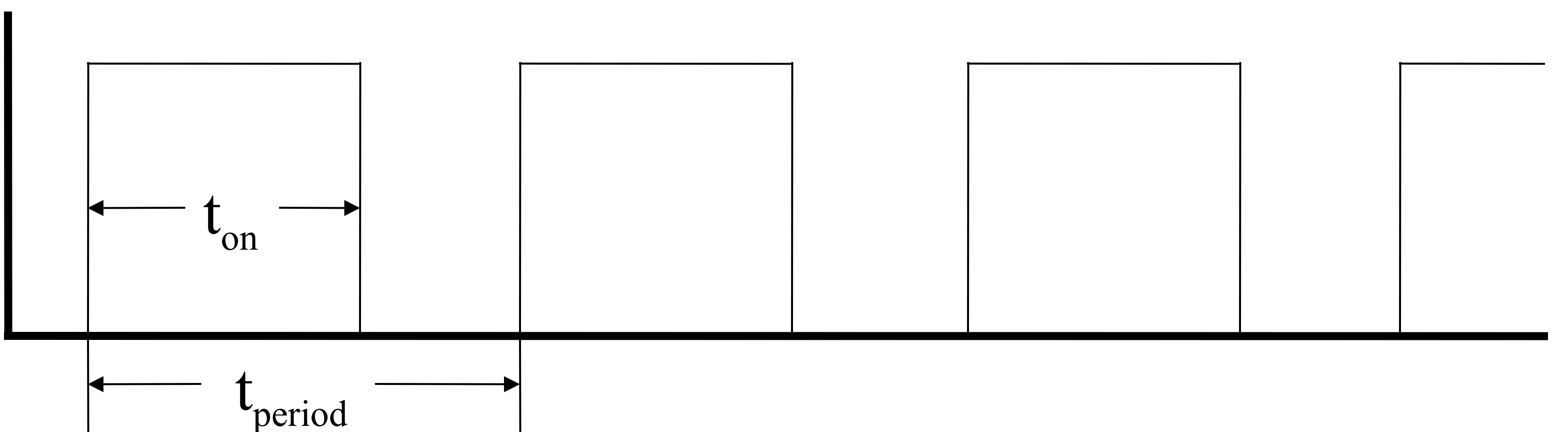
- Friction forces oppose the direction of motion.
- We've seen damping friction: $F_d = -f(v)$
- Coulomb ("sliding") friction is a constant F_c depending on force against the surface.
  - When there is motion, $F_c = \eta$
  - When there is no motion, $F_c = \eta + \varepsilon$
- Extra force is needed to unstick an object and get motion started.

Why is Stiction Bad?

- Non-zero steady-state error.
- Stalled motors draw high current.
  - Running motor converts current to motion.
  - Stalled motor converts *more* current to heat.
- Whining from pulse-width modulation.
  - Mechanical parts bending at pulse frequency.

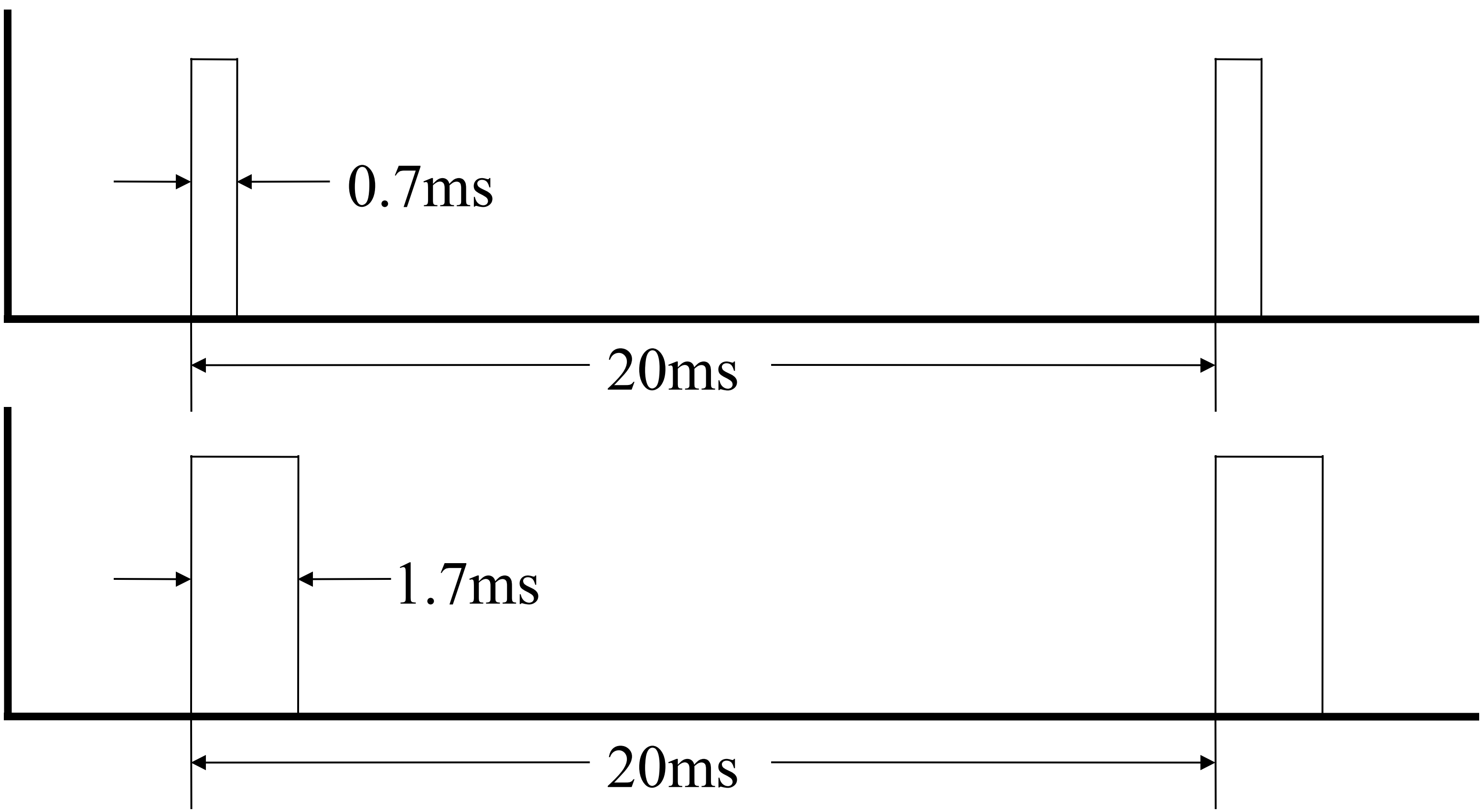
Pulse-Width Modulation

- A digital system works at 0 and 5 volts.
  - Analog systems want to output control signals over a continuous range.
  - How can we do it?
- Switch very fast between 0 and 5 volts.
  - Control the average voltage over time.
- Pulse-width ratio = t_{on}/t_{period}. (30-50 µsec)



Pulse-Code Modulated Signal

- Some devices are controlled by the length of a pulse-code signal.
  - Position servo-motors, for example.



Integrator Wind-Up

- Suppose we have a PI controller

$$u(t) = -k_P e(t) - k_I \int_0^t e\, dt + u_b$$

- Motion might be blocked, but the integral is winding up more and more control action.

$$u(t) = -k_P e(t) + u_b$$
$$\dot{u}_b(t) = -k_I e(t)$$

- Reset the integrator on significant events.

Chattering

- Changing modes rapidly and continually.
  - Bang-Bang controller with thresholds set too close to each other.
  - Integrator wind-up due to stiction near the setpoint, causing jerk, overshoot, and repeat.

Dead Zone

- A region where controller output does not affect the state of the system.
  - A system caught by static friction.
  - Cart-pole system when the pendulum is horizontal.
  - Cruise control when the car is stopped.

- Integral control and dead zones can combine to cause integrator wind-up problems.

Saturation

- Control actions cannot grow indefinitely.
  - There is a maximum possible output.
  - Physical systems are necessarily nonlinear.

- It might be nice to have bounded error by having infinite response.
  - But it doesn’t happen in the real world.

Backlash

- Real gears are not perfect connections.
  - There is space between the teeth.

- On reversing direction, there is a short time when the input gear is turning, but the output gear is not.

Parameter Drift

- Hidden parameters can change the behavior of the robot, for no obvious reason.
  - Performance depends on battery voltage.
  - Repeated discharge/charge cycles age the battery.
- A controller may compensate for small parameter drift until it passes a threshold.
  - Then a problem suddenly appears.
  - Controlled systems make problems harder to find

Unmodeled Effects

- Every controller depends on its simplified model of the world.
  - Every model omits almost everything.
- If unmodeled effects become significant, the controller's model is wrong,
  - so its actions could be seriously wrong.
- Most controllers need special case checks.
  - Sometimes it needs a more sophisticated model.