CS 4758/6758: Robot Learning

Spring 2010: Lecture 7

Ashutosh Saxena

Robot Ingredients

- Basics (statistics, kinematics)

- Perception / Sensing
- Localization / Estimation

- Control
- Planning (Path planning)

Control

- Dynamical systems
- PID controllers
- Common controllers

Later:

- Feedback control (includes sensing)
- Learning control

Dynamical Systems

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

$$\dot{\mathbf{x}} = F(\mathbf{x}) \quad \text{where } \mathbf{x} \in \Re^n$$

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

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

$$\mathbf{u} = H_i(\mathbf{x})$$

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

Linear Dynamical Systems

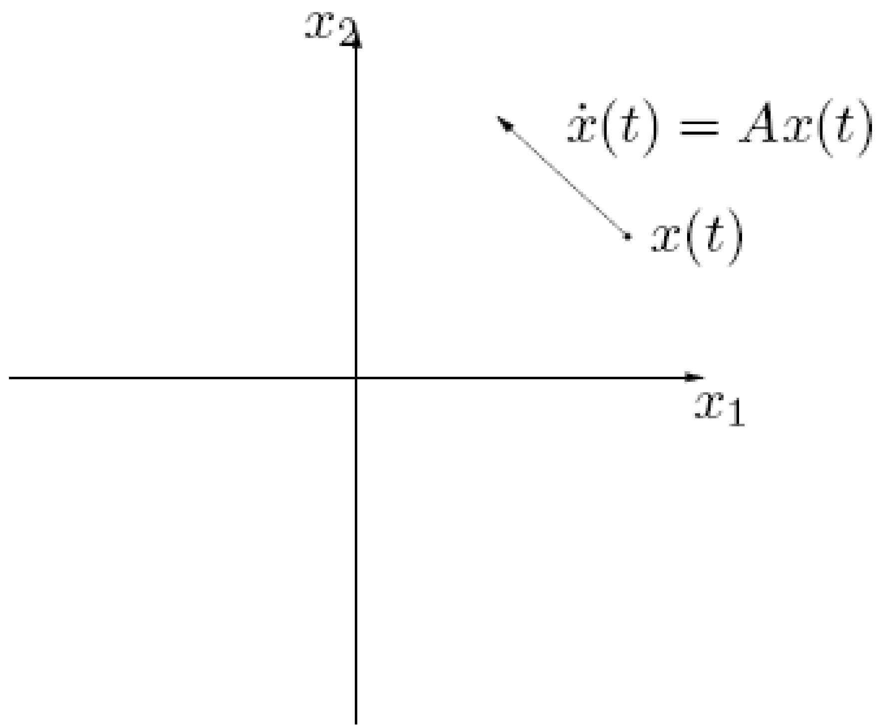
Ashutosh Saxena

Linear Dynamical System

$$\dot{x} = Ax$$

- $x(t) \in \mathbf{R}^n$ is called the state
- n is the *state dimension* or (informally) the *number of states*
- A is the *dynamics matrix*
  (system is *time-invariant* if A doesn't depend on t)

picture (*phase plane*):



In One Dimension

- Simple linear system
- Fixed point

$$\dot{x} = kx$$

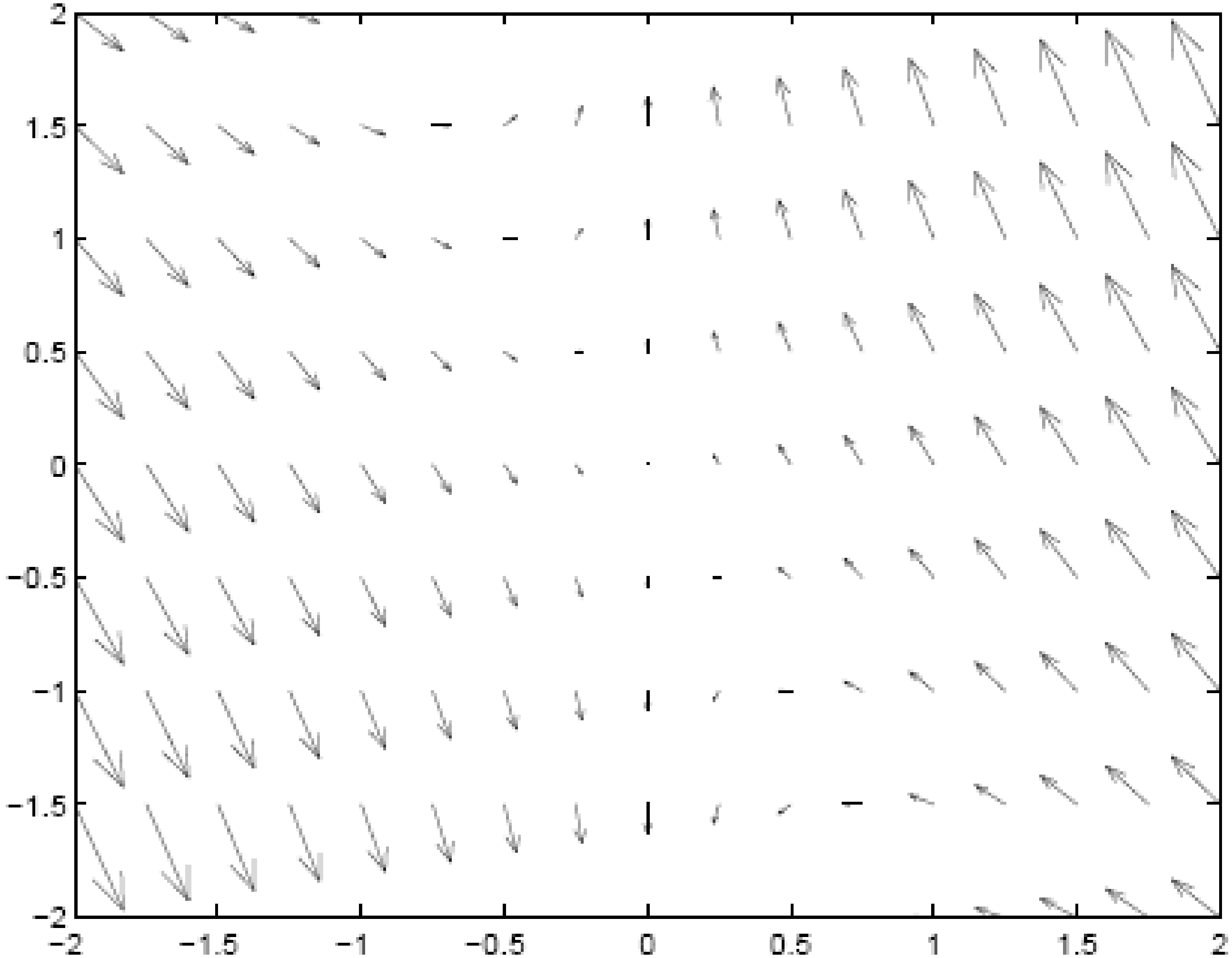
- Solution

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

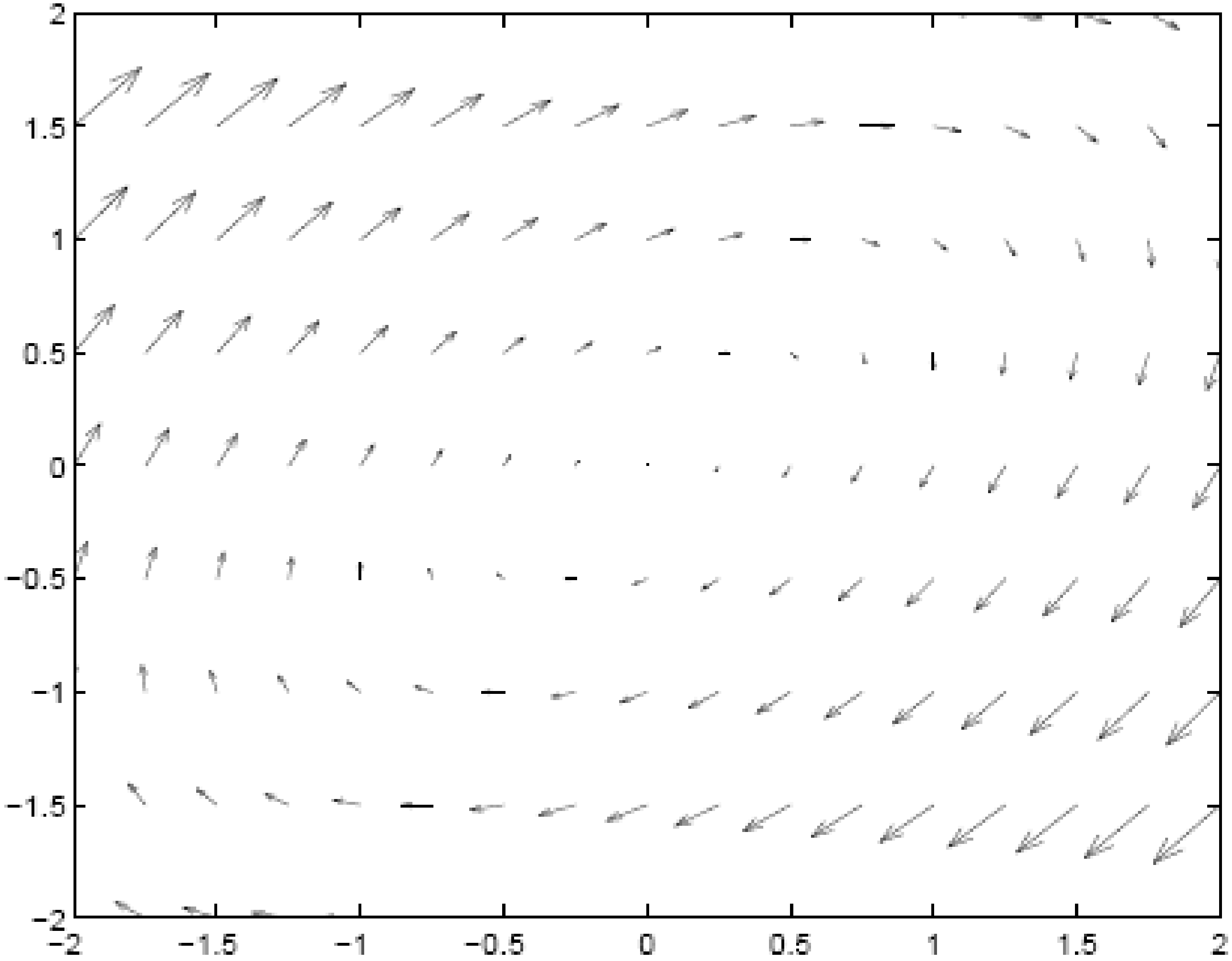
$$x(t) = x_0 e^{kt}$$

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

example 1: $\dot{x} = \begin{bmatrix} -1 & 0 \\ 2 & 1 \end{bmatrix} x$



example 2: $\dot{x} = \begin{bmatrix} -0.5 & 1 \\ -1 & 0.5 \end{bmatrix} x$



In Two Dimensions

- Often, we have position and velocity:

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

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

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

Node Behavior

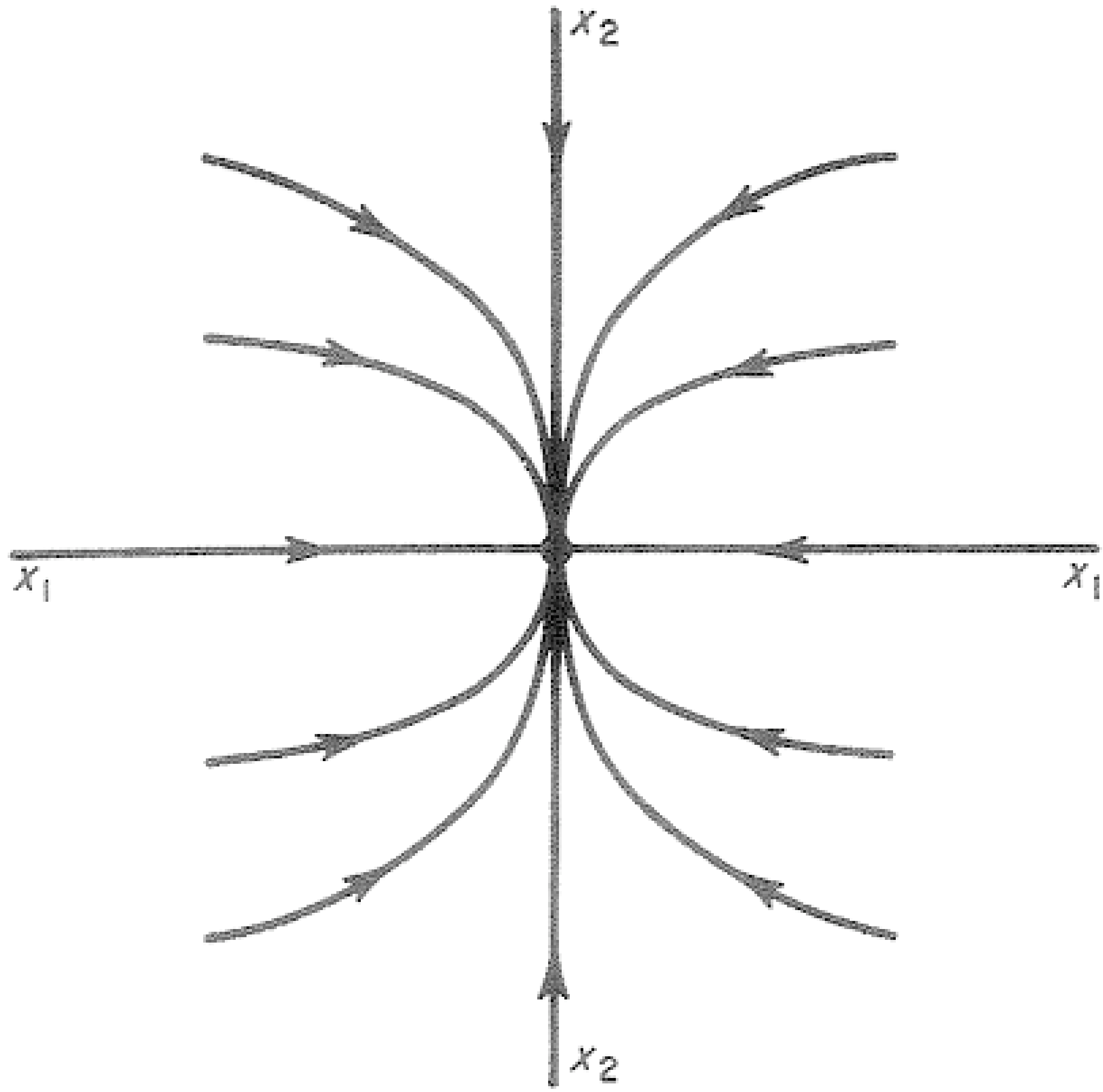


FIG. C. Node: $B = \begin{bmatrix} \lambda & 0 \\ 0 & \mu \end{bmatrix}$, $\lambda < \mu < 0$.

Focus Behavior

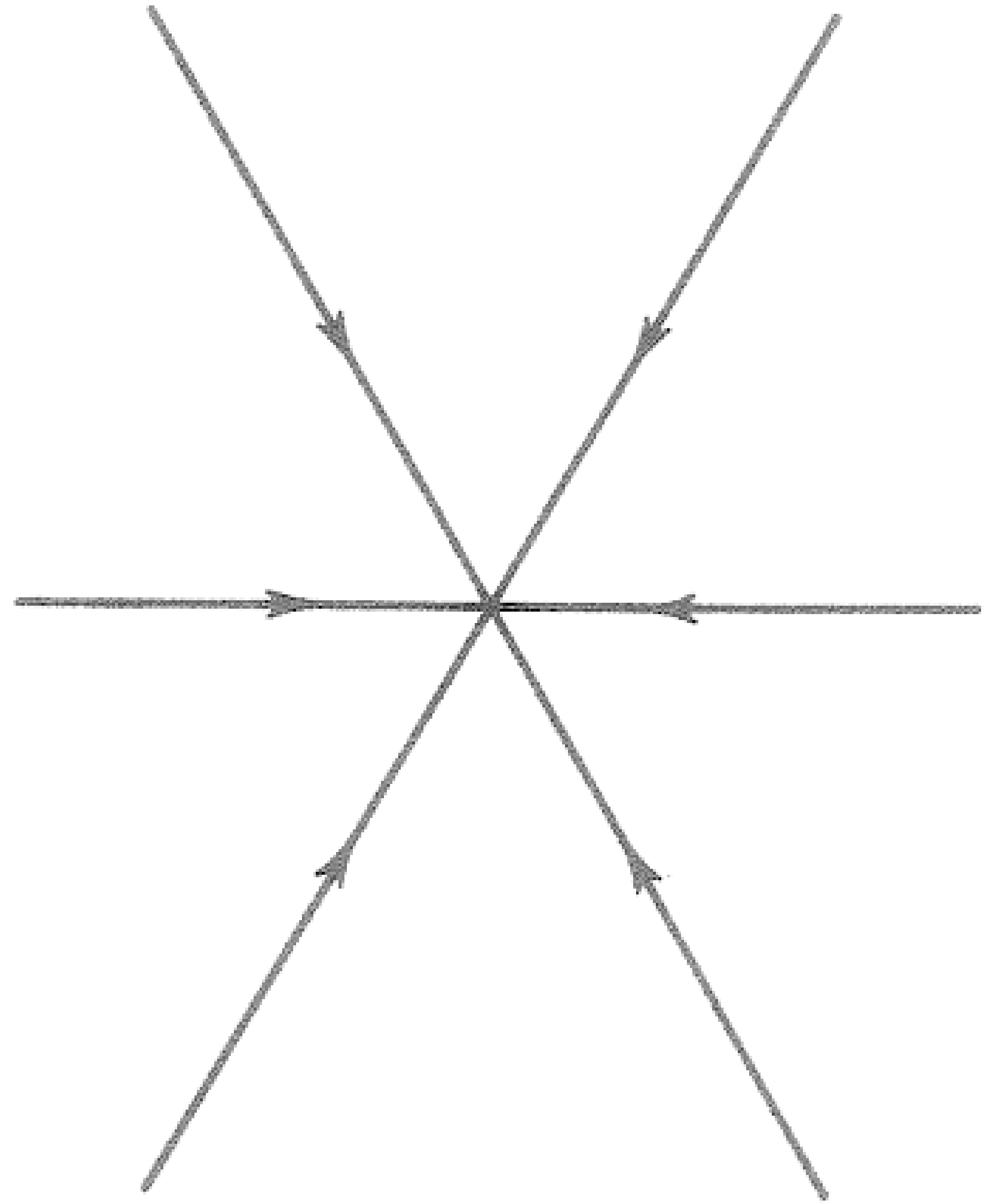


FIG. B. Focus: $B = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$, $\lambda < 0$.

ena

Saddle Behavior

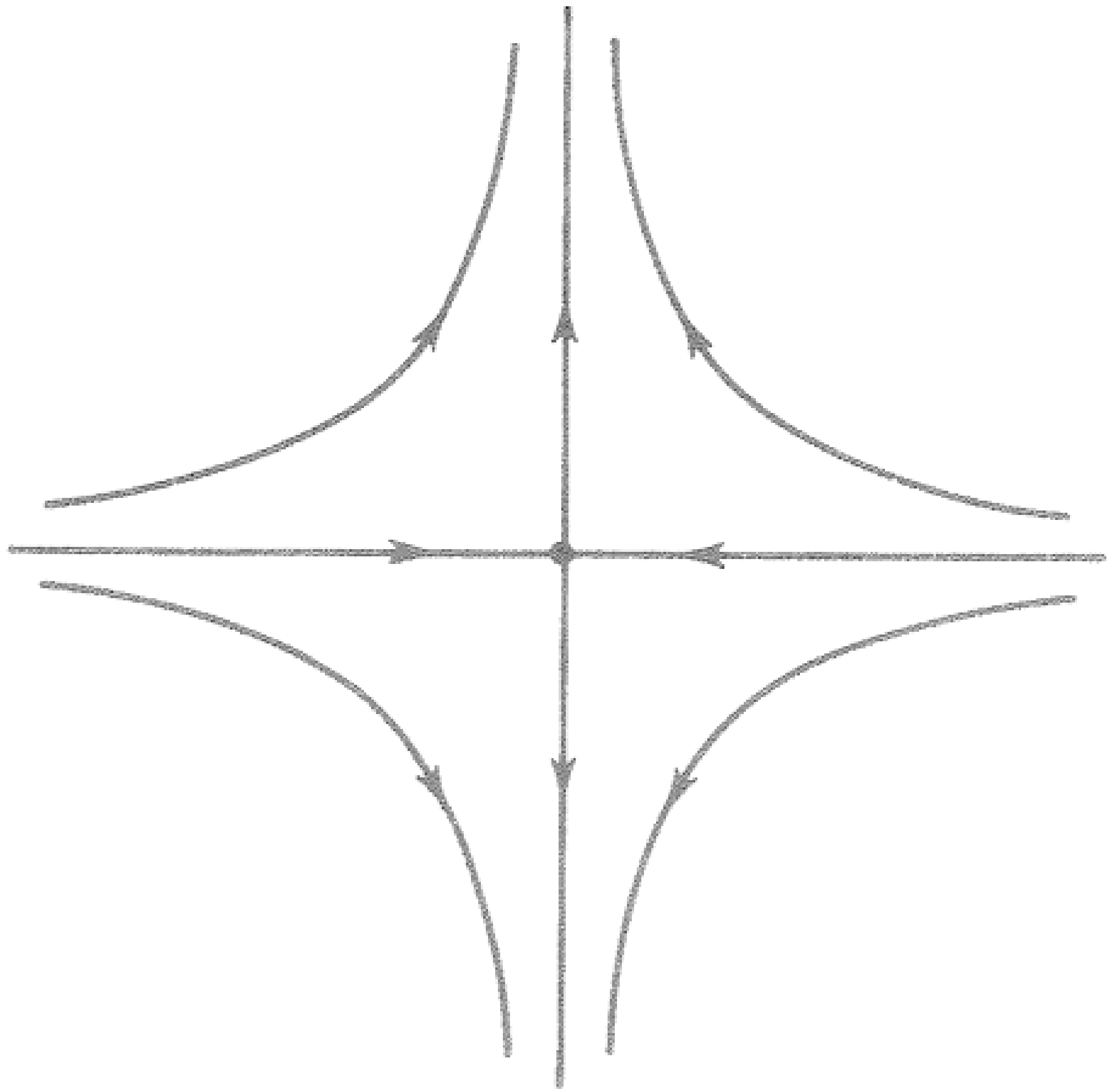


FIG. A. Saddle: $B = \begin{bmatrix} \lambda & 0 \\ 0 & \mu \end{bmatrix}$, $\lambda < 0 < \mu$.

Spiral Behavior

(stable attractor)

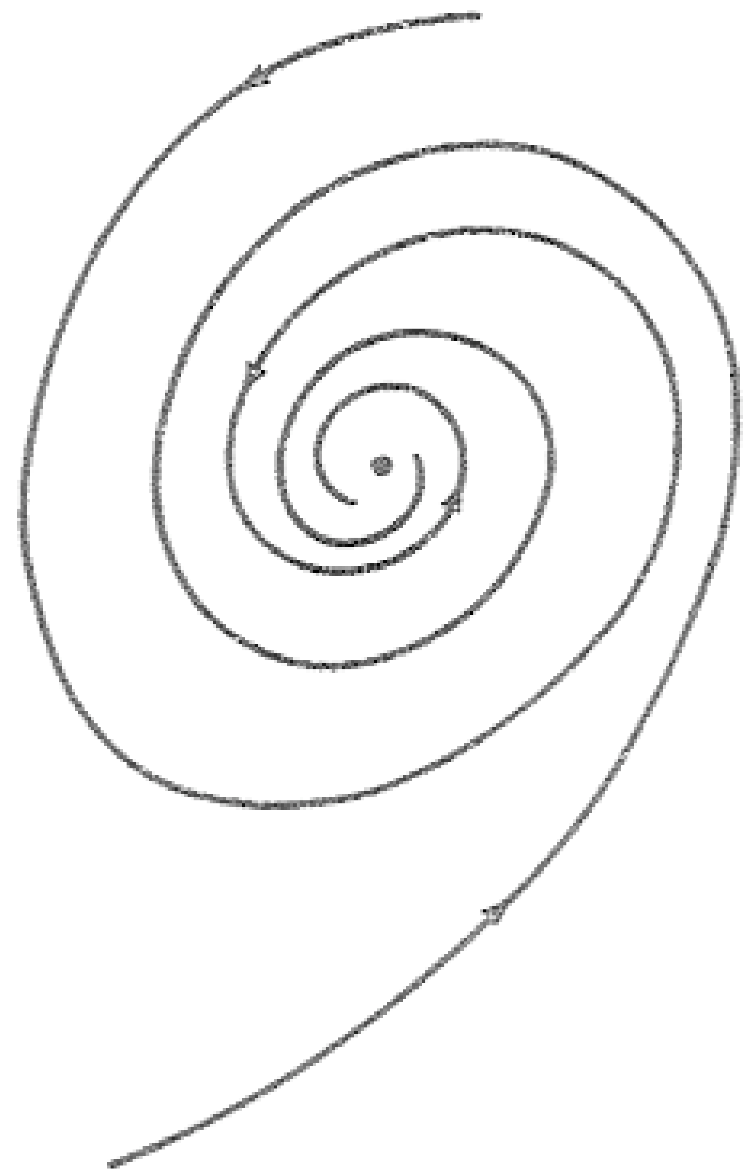


FIG. E. Spiral sink: $B = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$, $b > 0 > a$.

Center Behavior

(undamped oscillator)

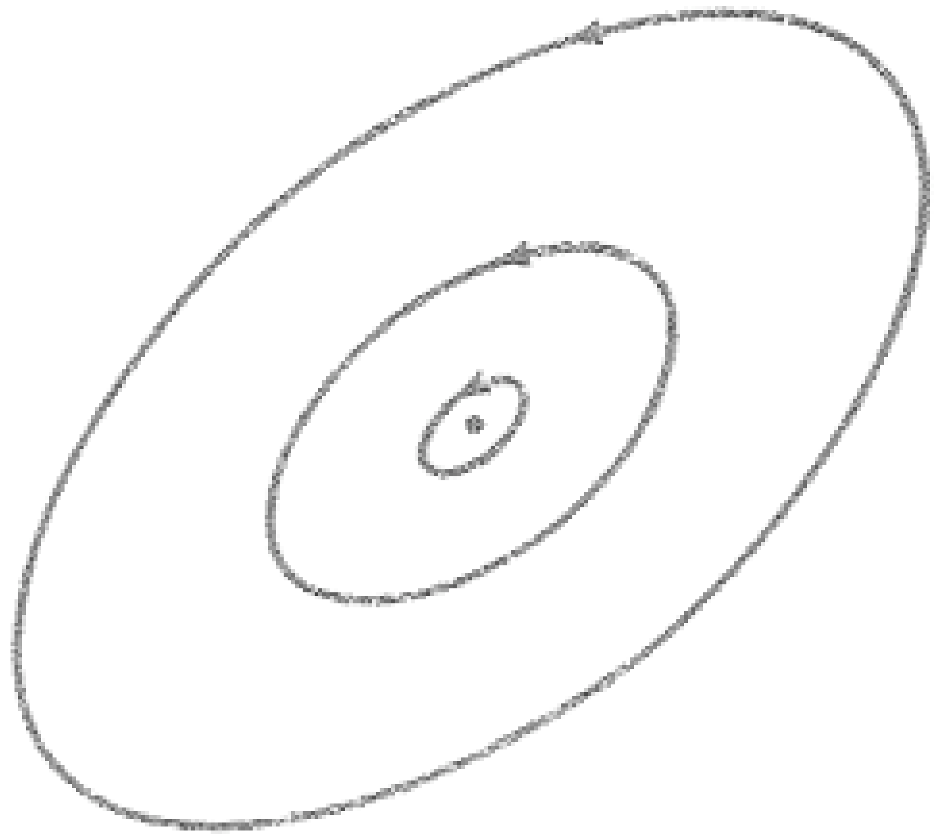
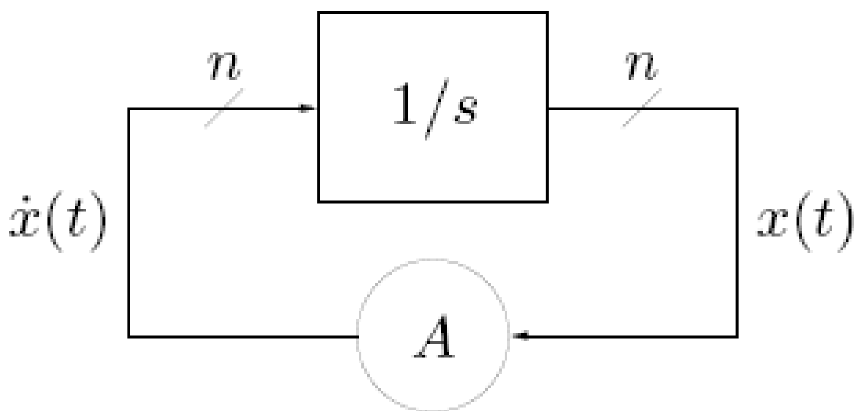


FIG. F. Center: $B = \begin{bmatrix} 0 & -b \\ b & 0 \end{bmatrix}$, $b > 0$.

Block Diagram

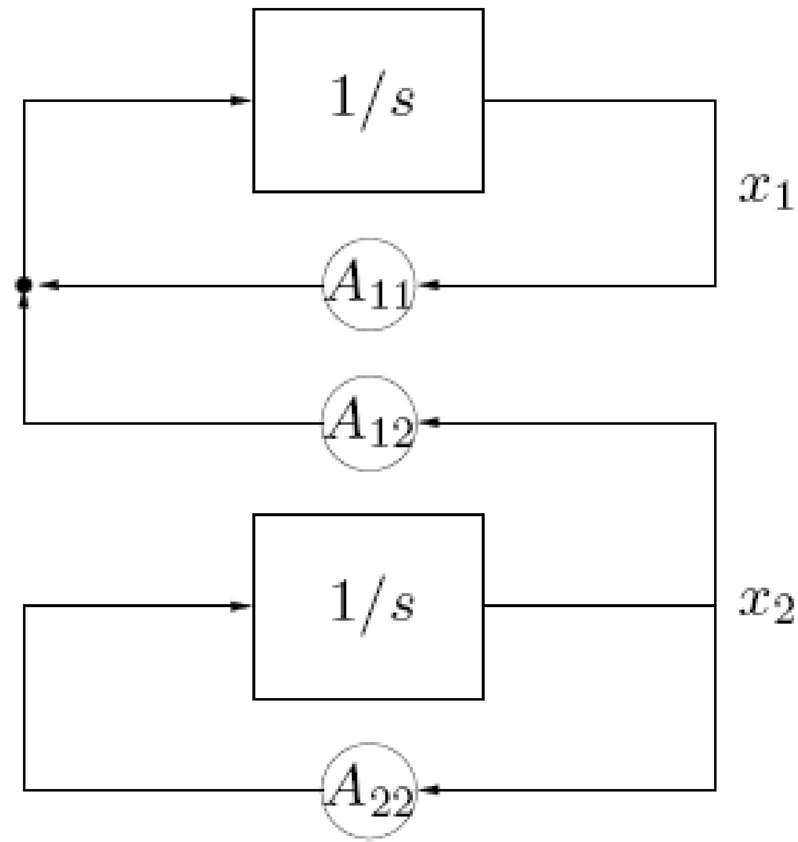
block diagram representation of $\dot{x} = Ax$:



- $1/s$ block represents n parallel scalar integrators
- coupling comes from dynamics matrix A

useful when A has structure, *e.g.*, block upper triangular:

$$\dot{x} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} x$$



here x_1 doesn't affect x_2 at all

Examples of Linear Dynamical Systems

Finite-state discrete-time Markov chain

$z(t) \in \{1, \ldots, n\}$ is a random sequence with

$$\mathbf{Prob}(\ z(t+1) = i \mid z(t) = j\ ) = P_{ij}$$

where $P \in \mathbf{R}^{n \times n}$ is the matrix of *transition probabilities*

can represent probability distribution of $z(t)$ as n-vector

$$p(t) = \begin{bmatrix} \mathbf{Prob}(z(t) = 1) \\ \vdots \\ \mathbf{Prob}(z(t) = n) \end{bmatrix}$$

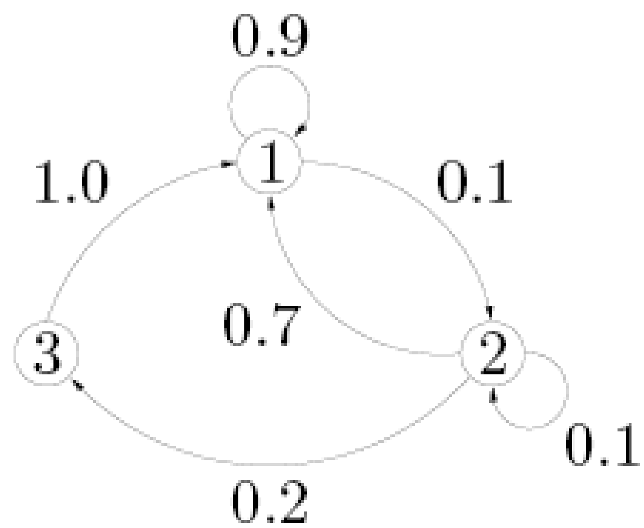
(so, *e.g.*, $\mathbf{Prob}(z(t) = 1, 2, \text{ or } 3) = [1\ 1\ 1\ 0 \cdots 0]p(t)$)

then we have $p(t+1) = Pp(t)$

P is often sparse; Markov chain is depicted graphically

- nodes are states
- edges show transition probabilities

example:



- state 1 is 'system OK'
- state 2 is 'system down'
- state 3 is 'system being repaired'

$$p(t+1) = \begin{bmatrix} 0.9 & 0.7 & 1.0 \\ 0.1 & 0.1 & 0 \\ 0 & 0.2 & 0 \end{bmatrix} p(t)$$

Higher order linear dynamical systems

$$x^{(k)} = A_{k-1}x^{(k-1)} + \cdots + A_1 x^{(1)} + A_0 x, \quad x(t) \in \mathbf{R}^n$$

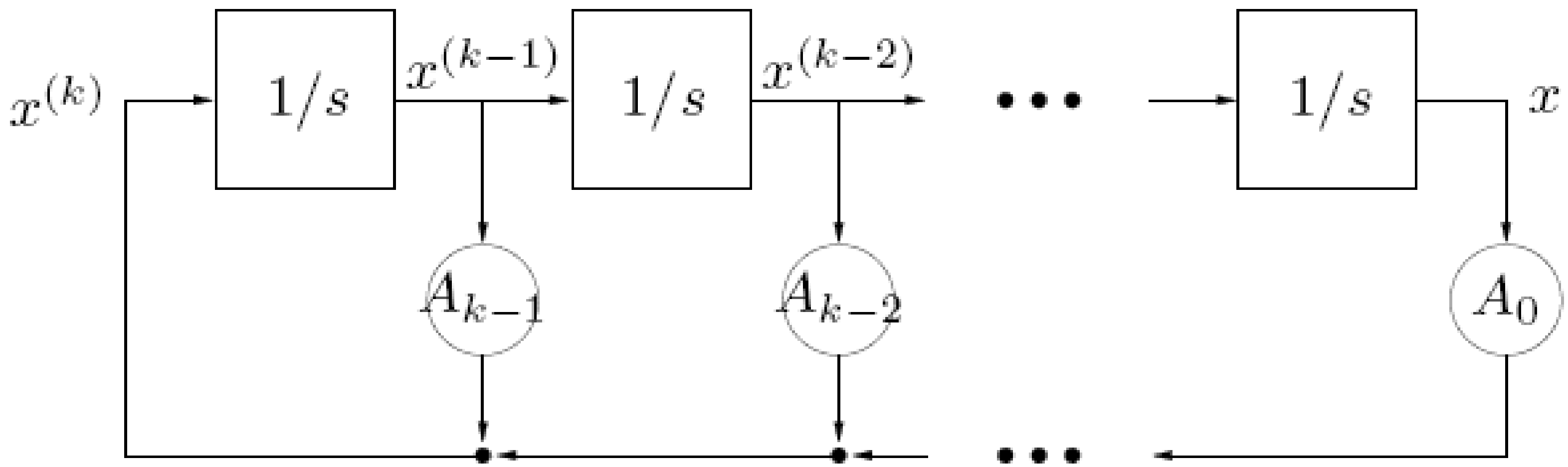
where $x^{(m)}$ denotes mth derivative

define new variable $z = \begin{bmatrix} x \\ x^{(1)} \\ \vdots \\ x^{(k-1)} \end{bmatrix} \in \mathbf{R}^{nk}$, so

$$\dot{z} = \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(k)} \end{bmatrix} = \begin{bmatrix} 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \cdots & I \\ A_0 & A_1 & A_2 & \cdots & A_{k-1} \end{bmatrix} z$$

a (first order) LDS (with bigger state)

Block Diagram



Mechanical Systems

mechanical system with k degrees of freedom undergoing small motions:

$$M\ddot{q} + D\dot{q} + Kq = 0$$

- $q(t) \in \mathbf{R}^k$ is the vector of generalized displacements
- M is the *mass matrix*
- K is the *stiffness matrix*
- D is the *damping matrix*

with state $x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$ we have

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{bmatrix} x$$

Linearization near equilibrium point

nonlinear, time-invariant differential equation (DE):

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

where $f : \mathbf{R}^n \to \mathbf{R}^n$

suppose x_e is an *equilibrium point*, *i.e.*, $f(x_e) = 0$

(so $x(t) = x_e$ satisfies DE)

now suppose $x(t)$ is near x_e, so

$$\dot{x}(t) = f(x(t)) \approx f(x_e) + Df(x_e)(x(t) - x_e)$$

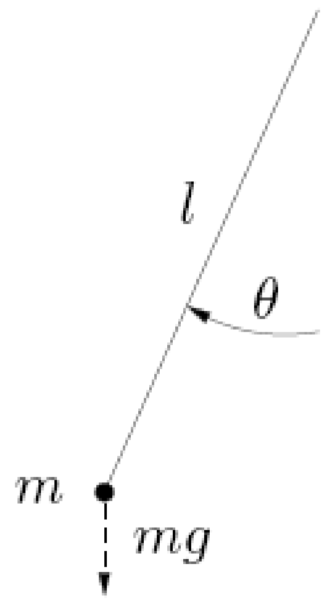
with $\delta x(t) = x(t) - x_e$, rewrite as

$$\dot{\delta x}(t) \approx Df(x_e)\delta x(t)$$

replacing $\approx$ with $=$ yields *linearized approximation* of DE near x_e

we *hope* solution of $\dot{\delta x} = Df(x_e)\delta x$ is a good approximation of $x - x_e$

example: pendulum



2nd order nonlinear DE $ml^2\ddot{\theta} = -lmg\sin\theta$

rewrite as first order DE with state $x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$:

$$\dot{x} = \begin{bmatrix} x_2 \\ -(g/l)\sin x_1 \end{bmatrix}$$

equilibrium point (pendulum down): $x = 0$

linearized system near $x_e = 0$:

$$\dot{\delta x} = \begin{bmatrix} 0 & 1 \\ -g/l & 0 \end{bmatrix} \delta x$$

Does linearization 'work'?

the linearized system usually, but not always, gives a good idea of the system behavior near x_e

example 1: $\dot{x} = -x^3$ near $x_e = 0$

for $x(0) > 0$ solutions have form $x(t) = \left(x(0)^{-2} + 2t\right)^{-1/2}$

linearized system is $\dot{\delta x} = 0$; solutions are constant

Controlling the Dynamical System

Controlling a Simple System

- Consider a simple system: $\dot{x} = F(x, u)$
  - Scalar variables x and u, not vectors $\mathbf{x}$ and $\mathbf{u}$.
  - 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)

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

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

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

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

The Bang-Bang Controller

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

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

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

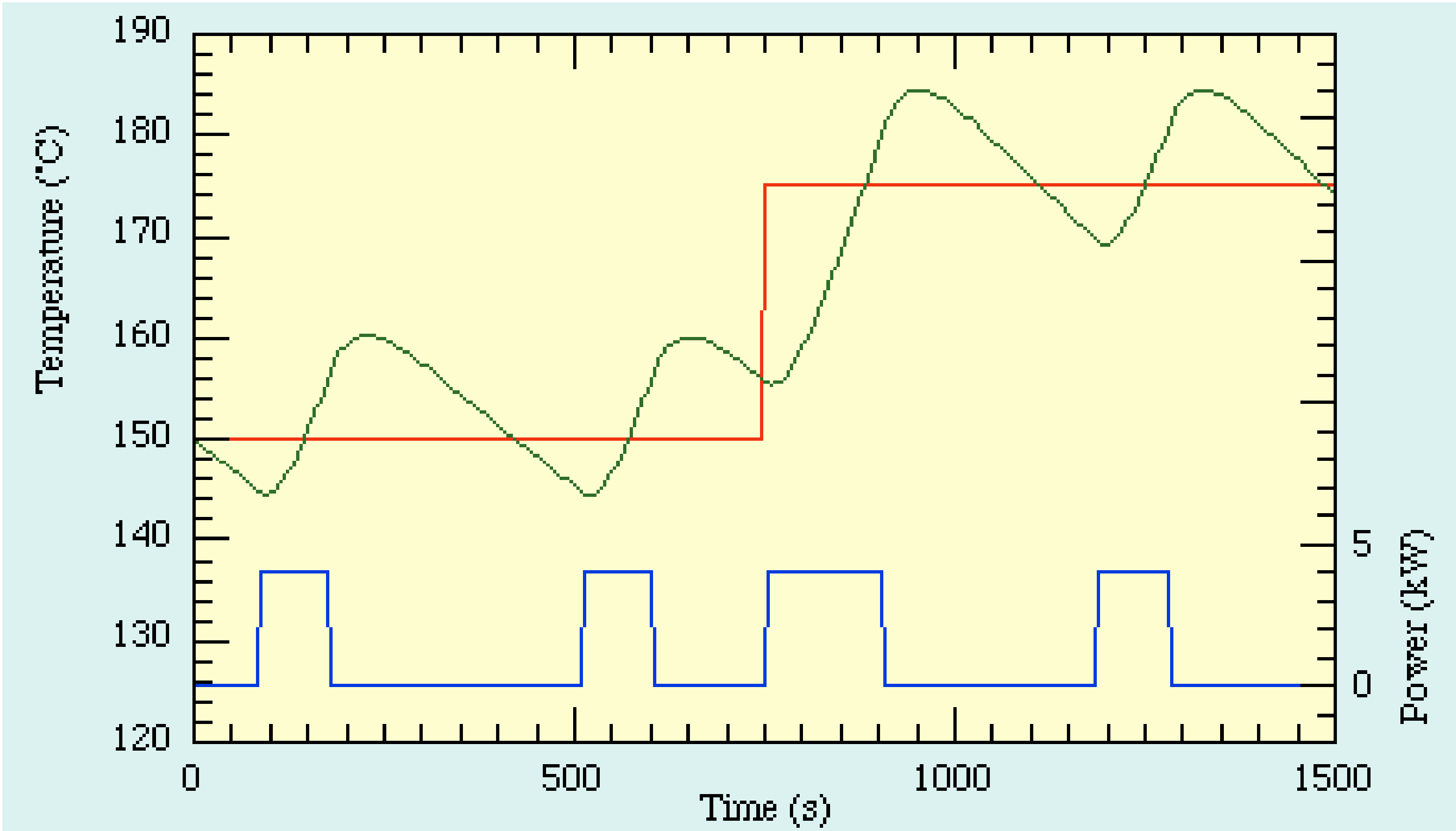
- To prevent chatter around e = 0,

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

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

- Household thermostat. Not very subtle.

Bang-Bang Control in Action



- Optimal for reaching the setpoint
- Not very good for staying near it

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

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

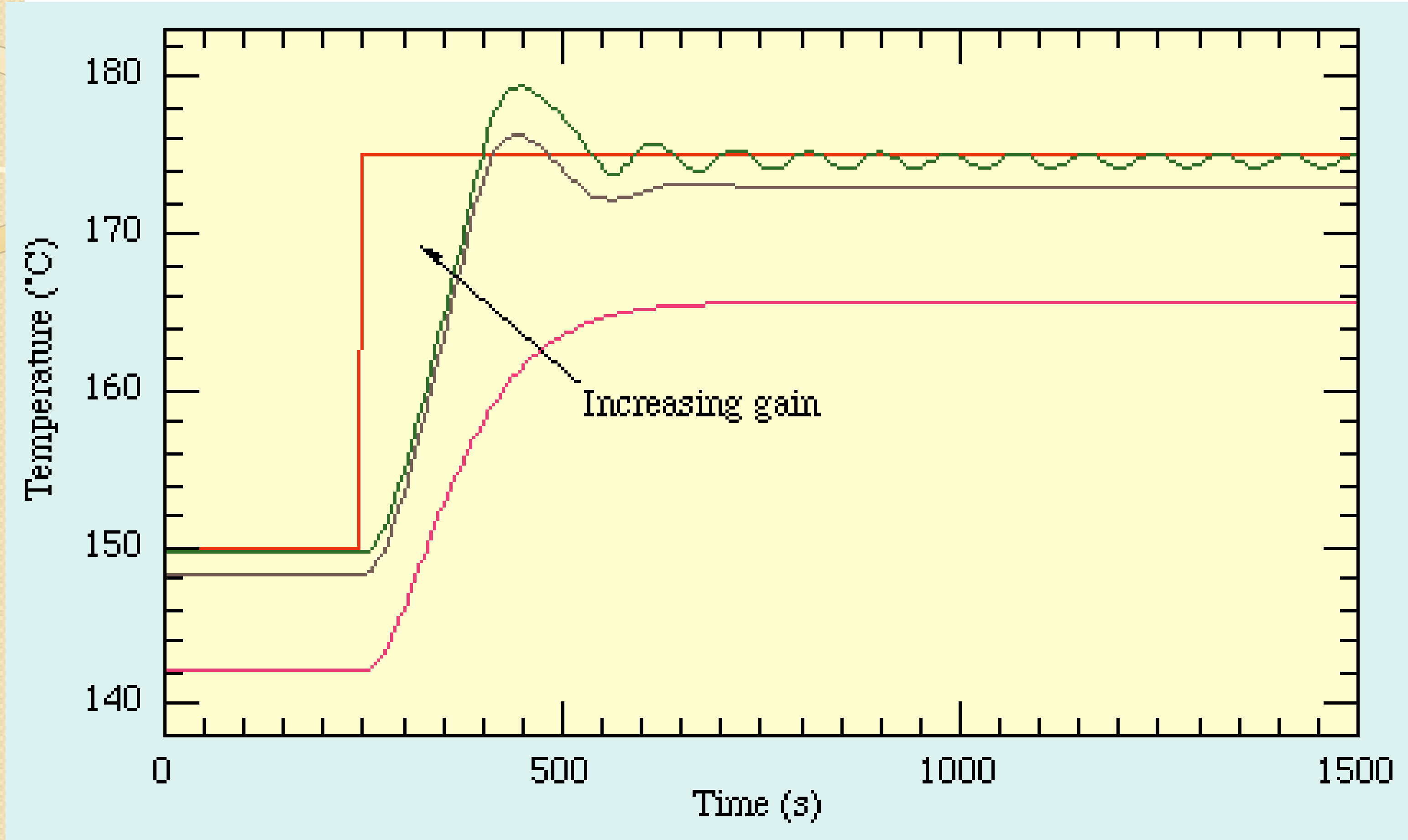
- You want to drive your car at velocity $v_{set.}$
- You issue the motor command $u = pos_{accel}$
- You observe velocity v_{obs}.

- 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 leads to overshoot, and even instability
- Steady-state offset

Find the control problem

Video

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.

$$u = -k_P e + u_b$$

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

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

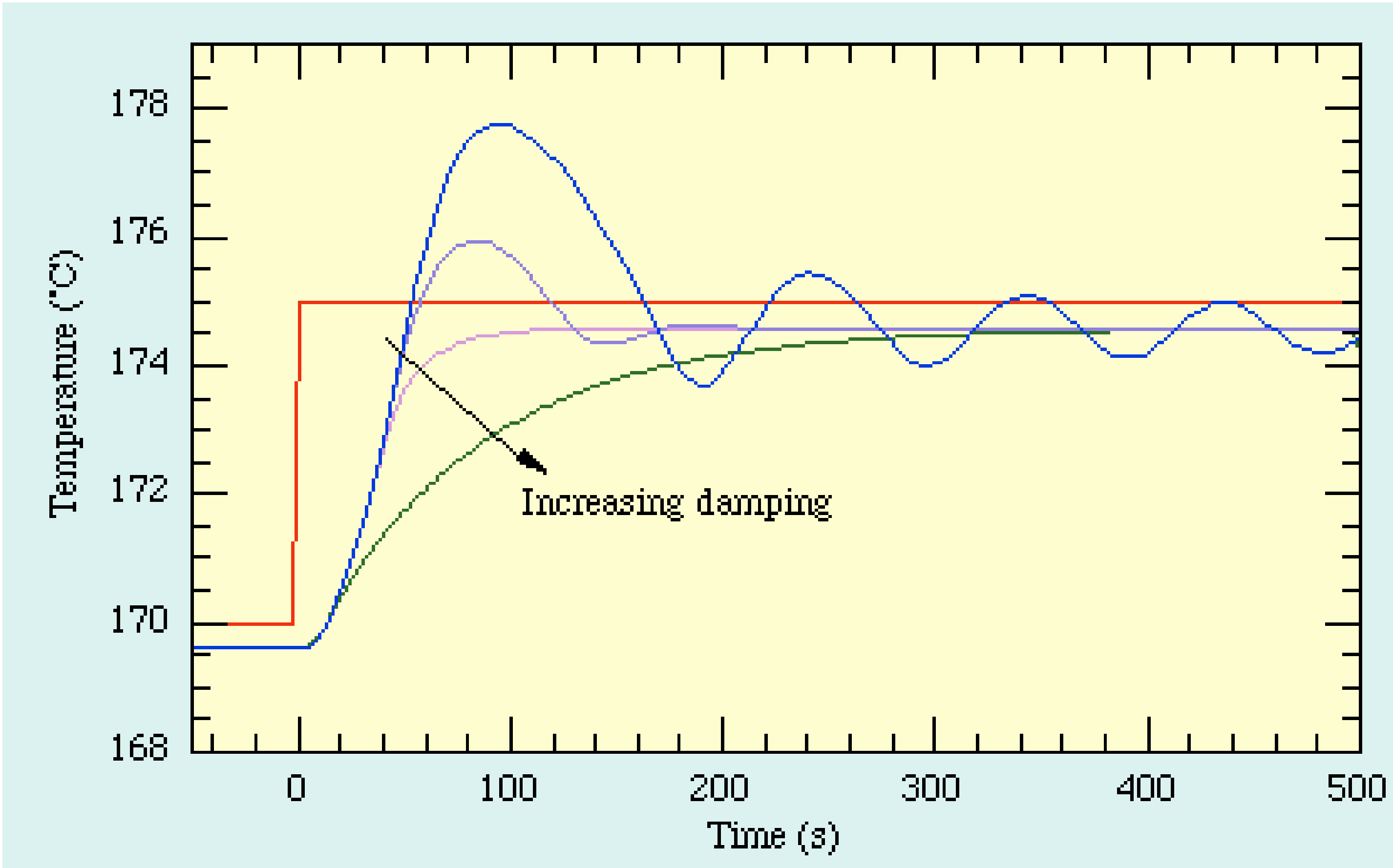
Derivative Control

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

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

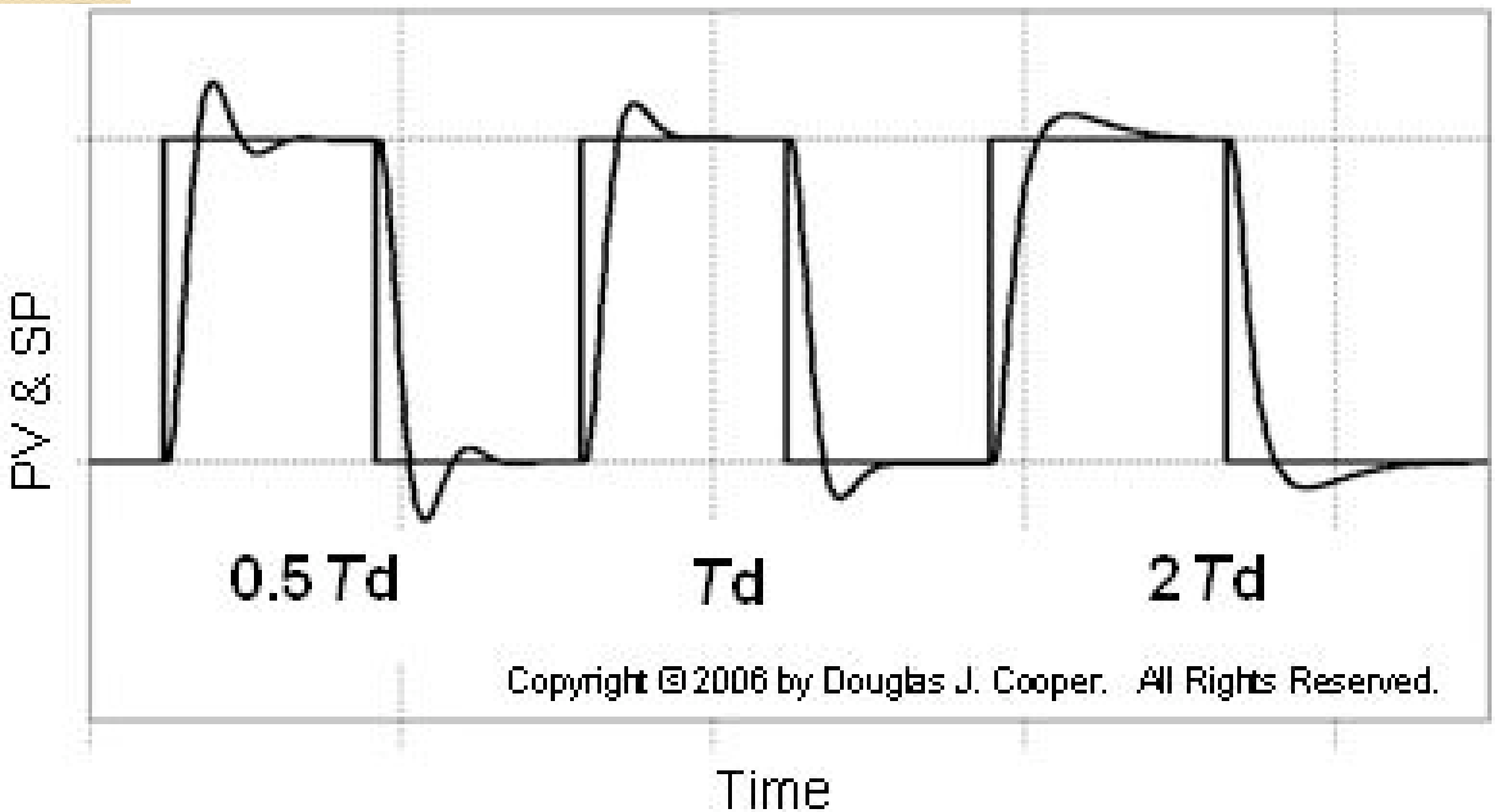
- Estimating a derivative from measurements is fragile, and amplifies noise.

Derivative Control in Action



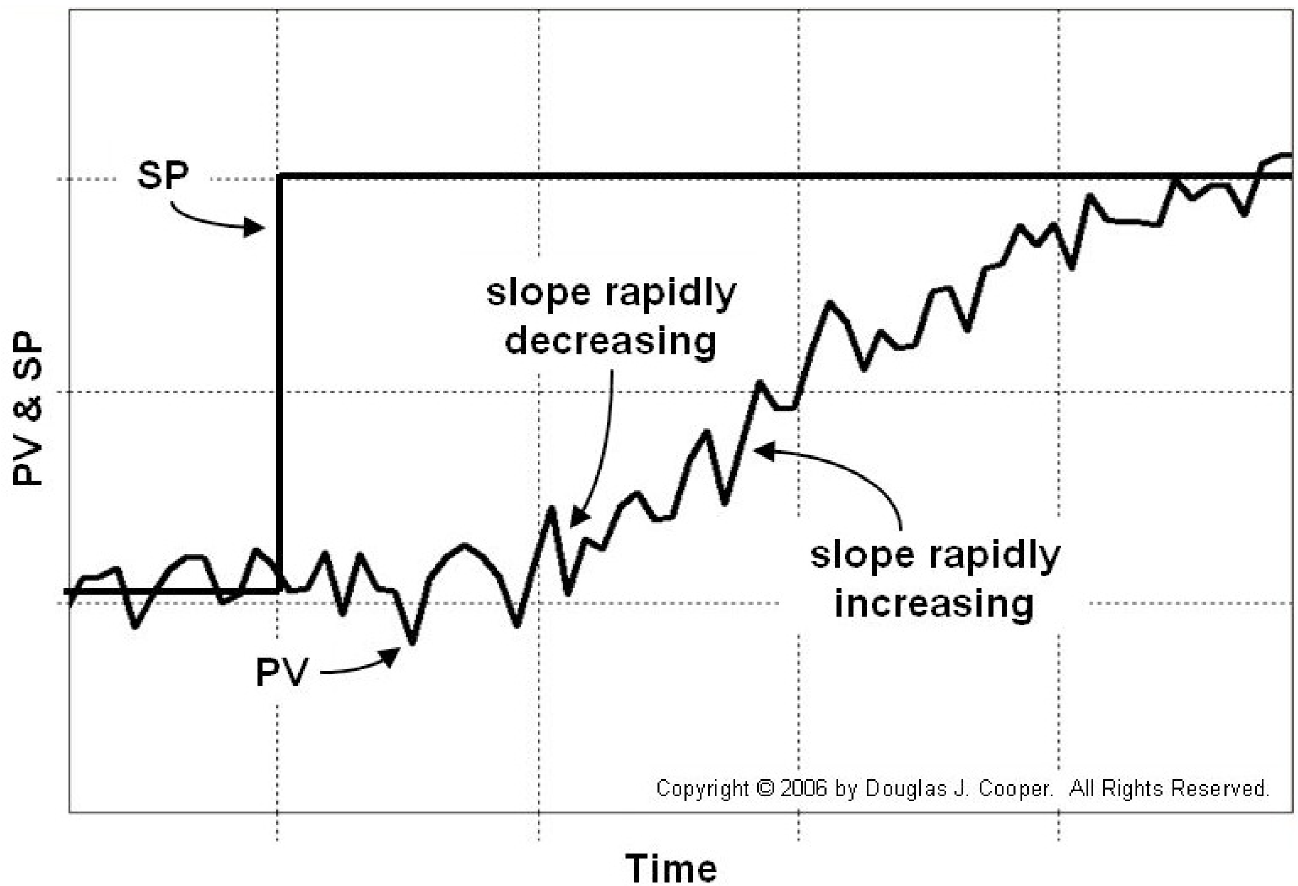
- Damping fights oscillation and overshoot
- But it's vulnerable to noise

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).

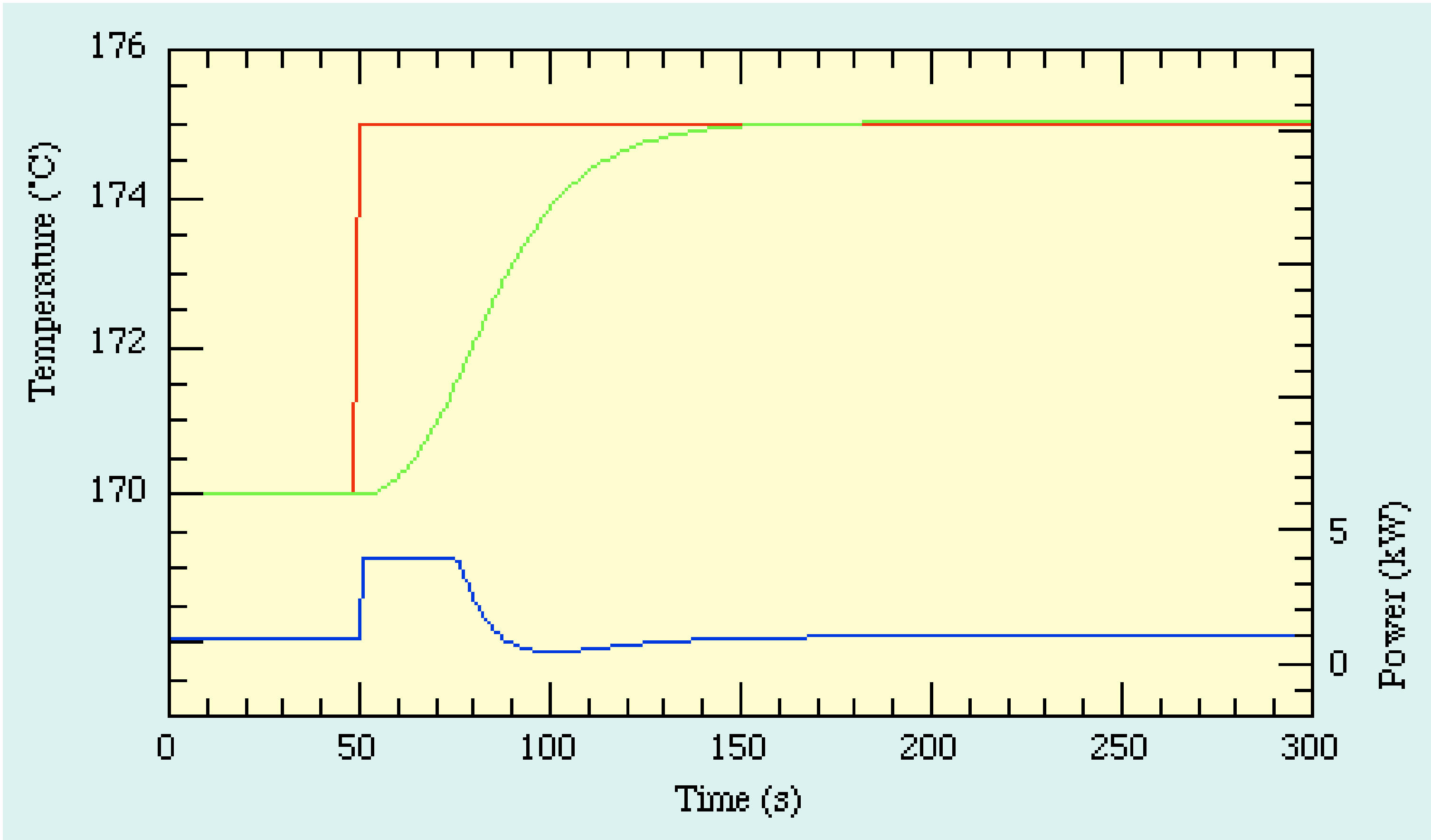
The PID Controller

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

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

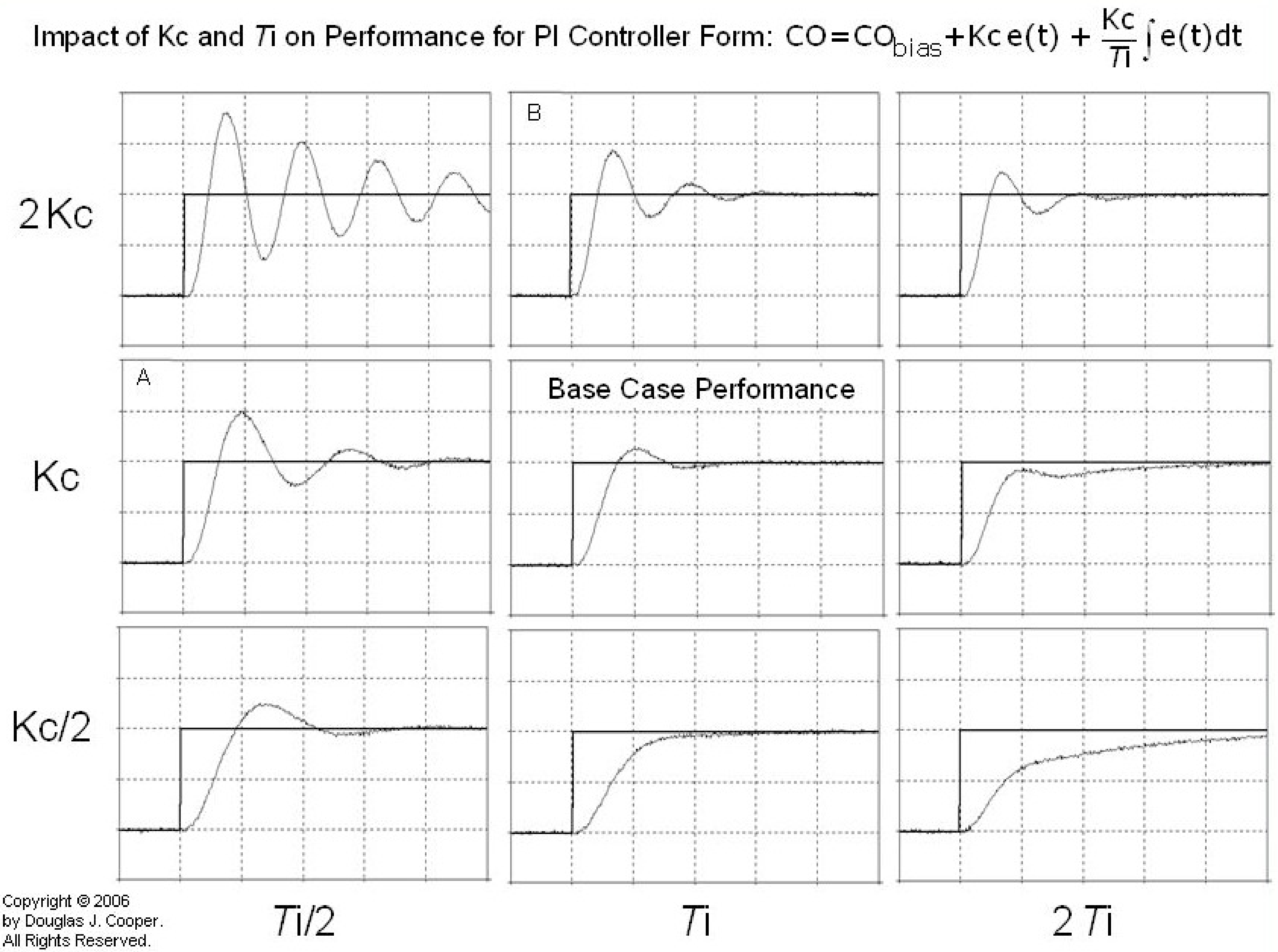
- The PID controller is the workhorse of the control industry. Tuning is non-trivial.

PID Control in Action



- But, good behavior depends on good tuning!

Exploring PI Control Tuning



Parameters are hard to tune manually. Once tuned, the system could change (parameter drift) and retune parameters?

- Use Machine Learning?