Mobile Robot Algorithms Laboratory

Lab 4

Thursday Week 4

http://www.andrew.cmu.edu/course/16-362-862

Agenda Today

- Administrative Issues
- Feedback Control
- Tuning
- Model Reference Control
- Identification
- Feedforward Control
- 2 Dof Control
- Overview of Lab 4

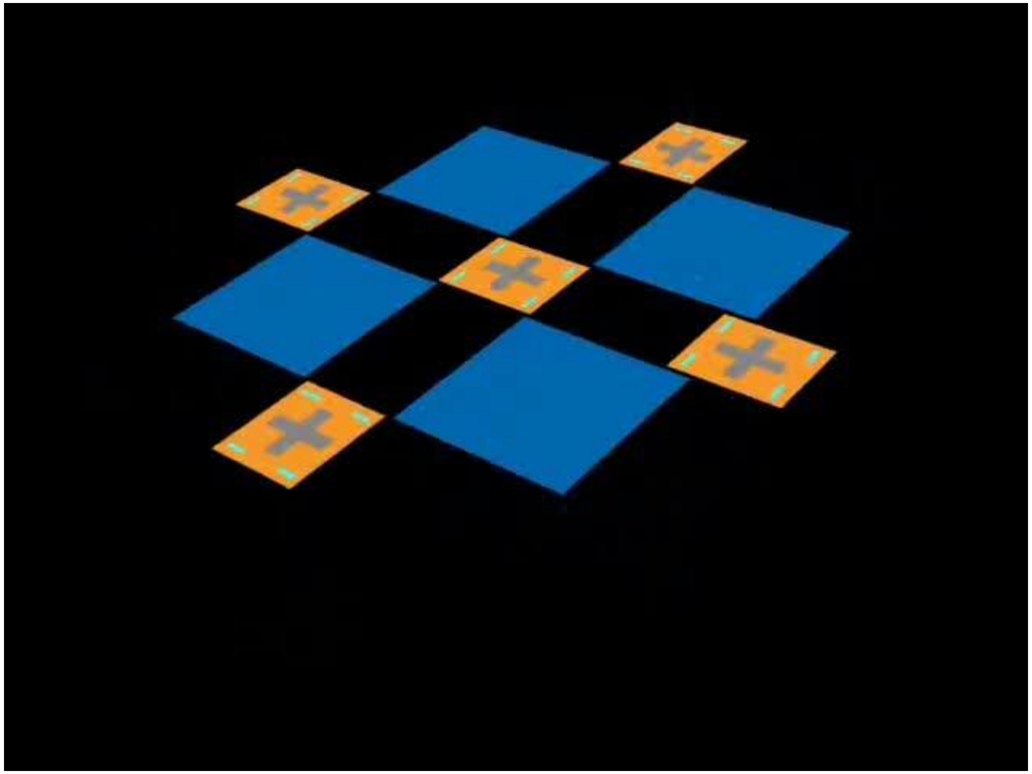
Agenda Today

- **Administrative Issues**
- Feedback Control
- Tuning
- Model Reference Control
- Identification
- Feedforward Control
- 2 Dof Control
- Overview of Lab 4

Agenda Today

- Administrative Issues
- **Feedback Control**
- Tuning
- Model Reference Control
- Identification
- Feedforward Control
- 2 Dof Control
- Overview of Lab 4

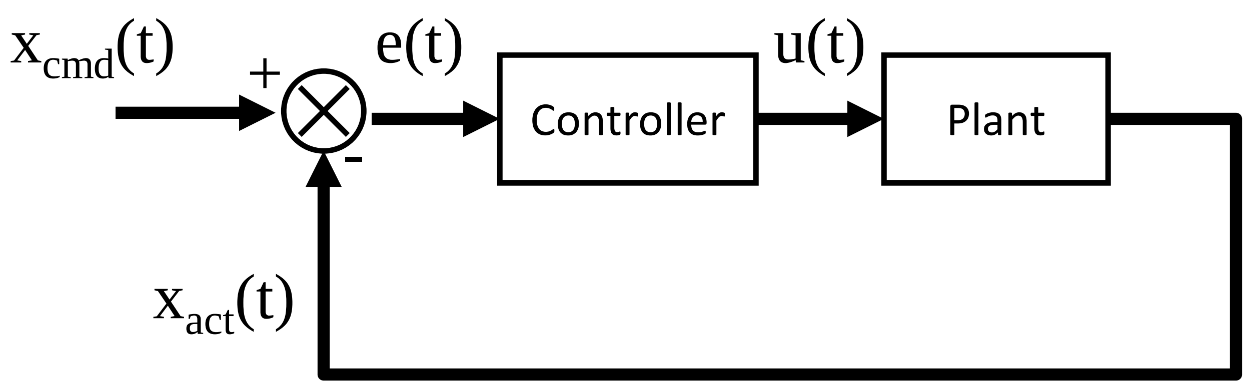
Control Matters



Control Matters

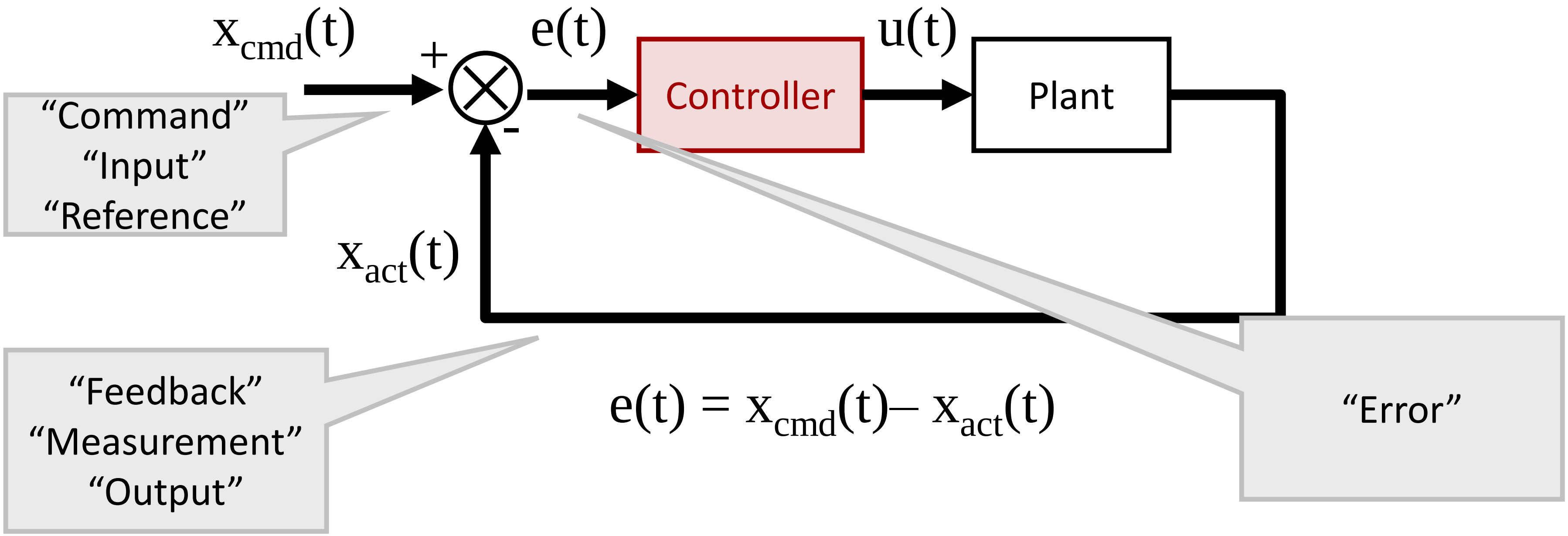
FeedBack Control Primer

- Contrast with “open loop” (send and forget) control.
- Philosophy: You can’t predict the future well in most cases (disturbances, model errors).
- Approach: Actively monitor what is going on and correct for errors that <u>were not</u> or <u>could not</u> be predicted.
- Basic issue: u(t) (control) is not x(t) (state). Often u(t) is related to derivatives of x(t). IOW, system has dynamics.



$$e(t) = x_{cmd}(t) - x_{act}(t)$$

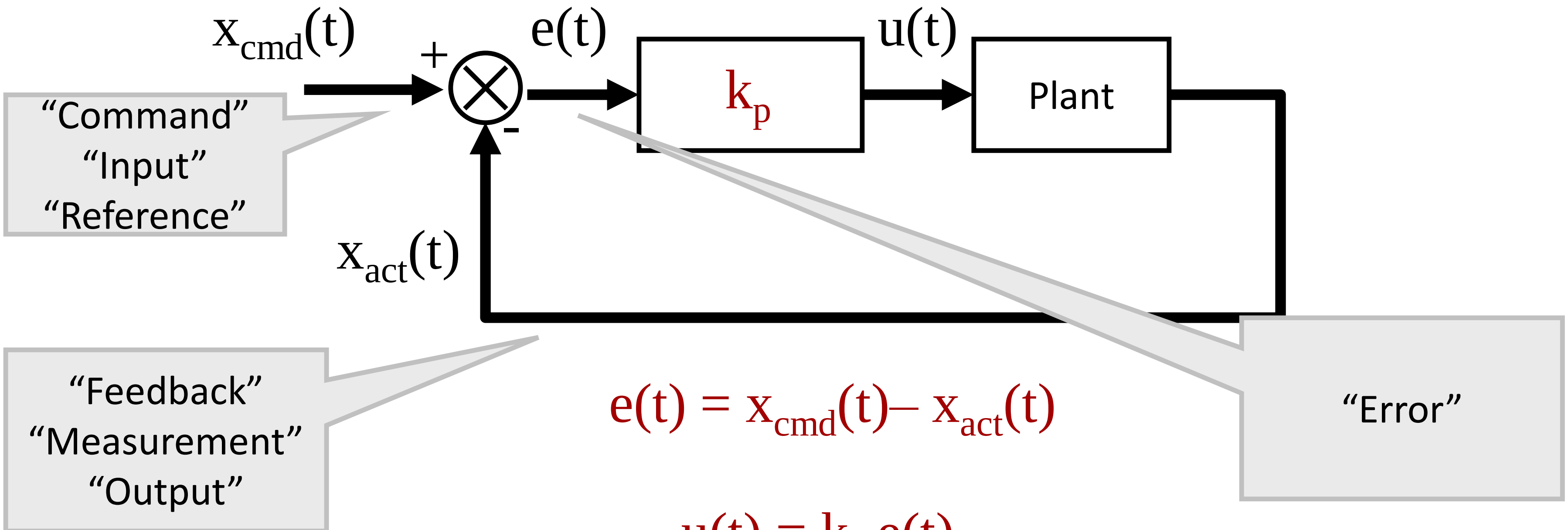
“Controllers”



$$e(t) = x_{cmd}(t) - x_{act}(t)$$

- So… What’s in the controller box?

"Controllers"

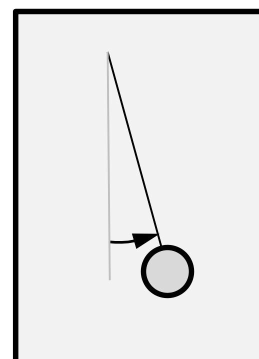


$$e(t) = x_{cmd}(t) - x_{act}(t)$$

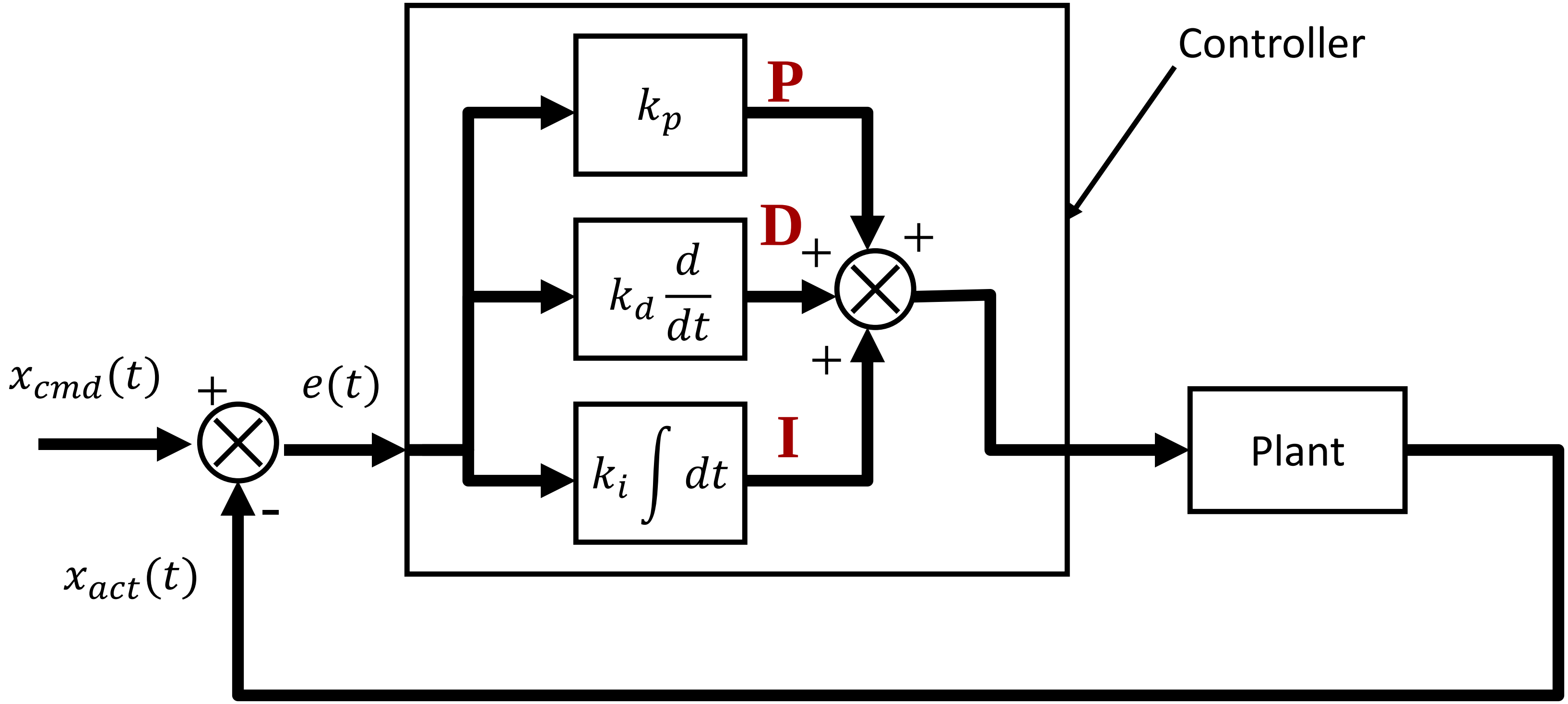
$$u(t) = k_p\, e(t)$$

Basic Proportional Control

- Generates a "restoring effort"
- "Stabilizes" the system
- like gravity stabilizes a pendulum.



Feedback Control Primer – PID Control



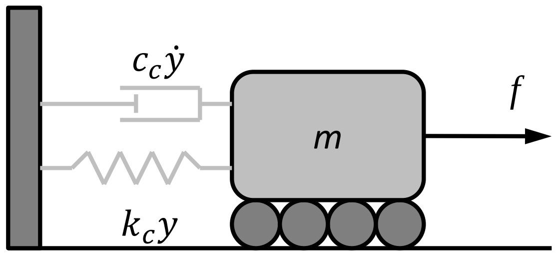
$$e(t) = x_{cmd}(t) - x_{act}(t)$$

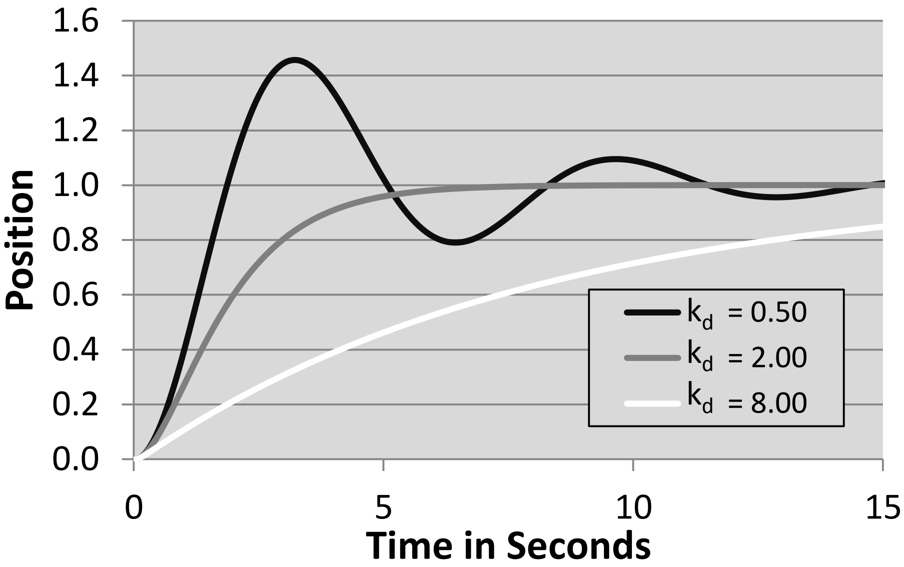
$$u(t) = k_p e(t) + k_d \frac{d}{dt} e(t) + k_i \int e(t) dt$$

PID Control

Example

- Feedback Control of Point Mass.





Feedback Control Primer – D Term

Carnegie Mellon
THE ROBOTICS INSTITUTE

- Basic idea is to drive the error to zero.
- However:
  - For the same instantaneous value of e(t) shouldn't we back off on u(t) if the error is decreasing?
  - Conversely, shouldn't we add some more to u(t) if the error is increasing?
- Reward the correct **trend** ...
  - reduce overshoot.
  - reduce settling time.

$$u(t) = k_p e(t)$$

$$u(t) \mathrel{+}= k_d \frac{d}{dt} e(t)$$

a) So decrease the control a little.
b) So increase the control a little

a) < 0 when making Progress.
b) > 0 when not making progress

Carnegie Mellon THE ROBOTICS INSTITUTE

I Term

- Basic idea is to drive the error to zero.
- However:
  - Shouldn't we increase u(t) if the error has persisted for a long time.
  - Shouldn't we decrease u(t) if the error has NOT persisted for a long time.
- Penalize persistence and <u>reduce steady state error</u>.

$$u(t) = k_p e(t)$$

$$u(t) \mathrel{+}= k_i \int e(t)dt$$

a) So increase the control a lot.

b) So increase control a little

a) large when error persists.

b) small when error is brief

Feedback Control Primer – Implementation In Java

Carnegie Mellon
THE ROBOTICS INSTITUTE

```
// Form error and its integral and derivative
lastError = error;  // DECLARE APPROPRIATELY
error     = goalState - actualState;
// COMPUTE DELTIME APPROPRIATELY
errorDerivative = (error - lastError) / delTime;
errorIntegral   += error * delTime;

// Clamp the integral to avoid windup
double sign = errorIntegral > 0.0f ? 1 : -1;
if(Math.abs(errorIntegral) > errorIntegralMax)
   errorIntegral = sign * errorIntegralMax;

// Compute desired control
double control =          error * p_gain
             + errorDerivative * d_gain
             + errorIntegral   * i_gain;
```

Feedback Control Primer – Caveats – Measuring dt

- To avoid dependence on processor load, use actual measurements of delTime to make algorithm work for any delTime.
  - Simple approximation is to use sleep().
  - Q: When does that work?
- State variable (x) being controlled may be…
  - position, velocity, angle, flow, force, etc.
- Control variable (u) may be …
  - power, torque, speed, etc.
- Each case for x and u will require different tuning.

Feedback Control Primer – Caveats - Windup

Carnegie Mellon
THE ROBOTICS INSTITUTE

- Beware integrals in real-time control systems subject to externally introduced persistent errors.
  - E.g. computing elapsed time across a breakpoint (implicit integral)
  - E.g. Error integrals in PID control (explicit integral)
- "Windup" is often VERY DANGEROUS !!!!
  - Dangerous just like positive feedback
  - Get this right the first time
- Make sure to clear all integrals at the appropriate time.
  - Don't remember integrals from one run to the next.
  - Clear the integral AFTER you press the go button
    - → not when the GUI is constructed.

Uncommanded Motion

- The error, error rate etc. can provide a good basis for detecting unsafe conditions.
- Uncommanded motion means either:
  - A) I am moving and I am not supposed to be
  - B) I am not moving and I am supposed to be.
- Generally, check if:

$$|v_{cmd} - v_{act}| > threshold$$

Loss of Control Authority

- If error has changed by a huge amount since last time, something is not right.
- Could be power:
  - someone hit the Estop button
  - Generator or engine shut off
- Could be traction / locomotion
  - someone picked up the robot
  - wheels are slipping / robot blocked by obstacle
- Could be error
  - Cable cut
  - Connector fell off
- You could check $\frac{d^2}{dt^2}e(t)$ to detect these things

Feedback Control Primer – Bad News

- Its not all good news.
- Proportional Term
  - Cannot remove steady state error
    - when there is deadband (e.g. stiction)
    - when there is motion (e.g. tracking)
- Integral Term
  - Introduce to eliminate steady state error.
  - But, causes overshoot & oscillation.
- Derivative term
  - reacts strongly to noise, causes jerky motion.
- Worse yet, the effects are coupled. Cannot tune very incrementally.

Feedback Control Primer – Hints

- PID is but one option. PD is another. P is a third. There are many other control schemes. In rough order of sophistication:
  - Cascade
  - Gain scheduling
  - Reference model based
  - Feedforward
  - Two degree of freedom – Stay Tuned for this
  - Predictive
  - Optimal
  - Nonlinear
  - Adaptive
- For PID.
  - Get P working.
  - Add D to reduce settling time.
  - Add I if steady state error is an issue.
- Every term addition or gain change changes <u>everything</u>.

Feedback Control Primer – Hints

- "error" is not always error. It may be caused by unreasonable expectations.
  - Only force / acceleration can change instantaneously in nature
    - Position and velocity (and hence potential and kinetic energy and momentum) cannot.
- Don't **expect** the system to do what it cannot do, errors blow up and chaos results.
  - Think about error in terms of deviation from reasonable response motions.
- See model reference / feedforward control later.

Agenda Today

- Administrative Issues
- Feedback Control
- **Tuning**
- Model Reference Control
- Identification
- Feedforward Control
- 2 Dof Control
- Overview of Lab 4

Tuning – Best Practices

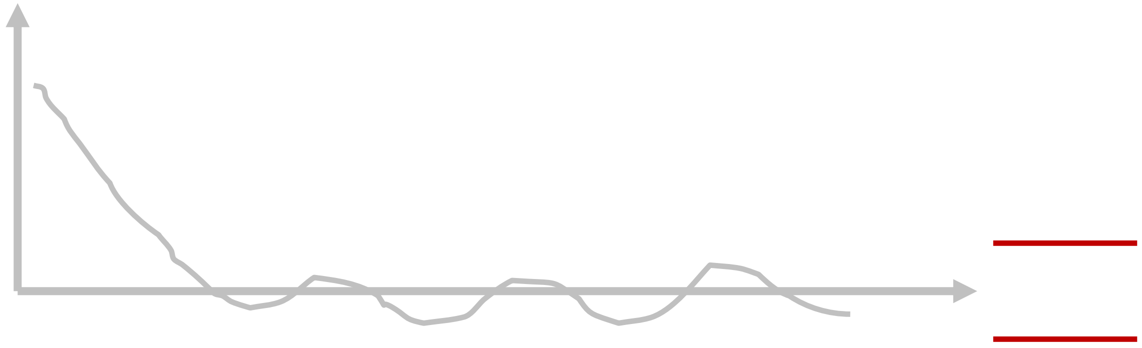
- Be structured and deliberate.
- Best gains and thresholds depend on..
  - Momentum, which depends on….
  - Speed of motion, which depends on…
  - The distance to the goal
    - So, performance depends on the task
  - The gains themselves a moment ago
    - So, performance now depends on performance earlier
    - Its circular because real systems have dynamics

Tuning - Robustness

- Know the limits of your tuning.
  - Sometimes, double the speed and it all falls apart.
- Reduce the limits of your tuning.
  - A robust “get me somewhere” is pretty useful.
- Regression testing
  - Keep your test code operational.

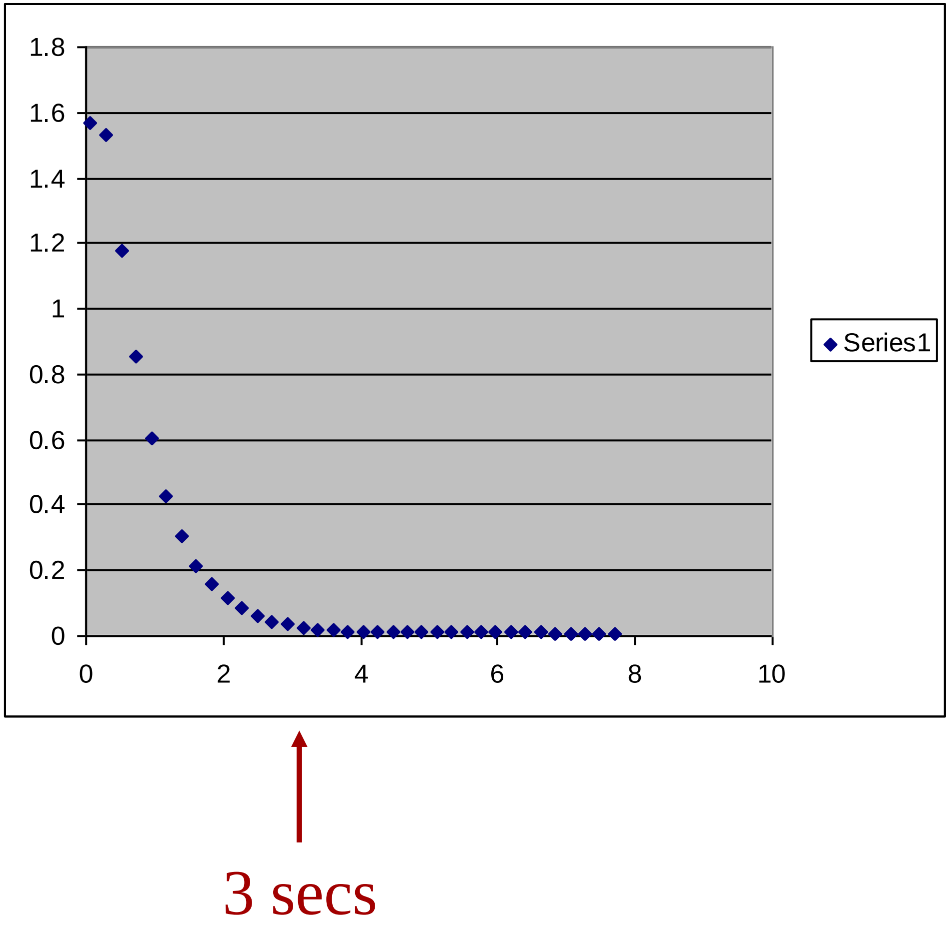
Tuning – Empiricism

- Don't try to predict what can be measured.
- Example:
  - Determine error threshold for servo shutoff ………. experimentally
    - disable the shutoff
    - plot the error



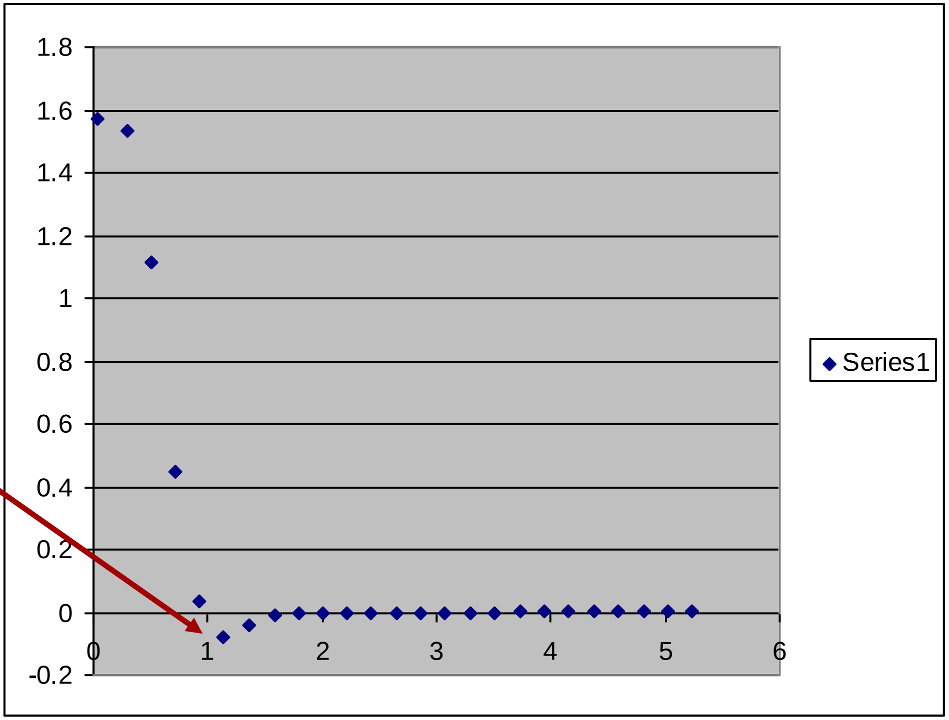
Tuning

- Turning 90 deg
- P=1, I,D=0
- Pretty good first guess.
- Takes 3-4 seconds to get there.
- Enter greed.
  - Can we make it faster?
  - Increase P gain?



Tuning

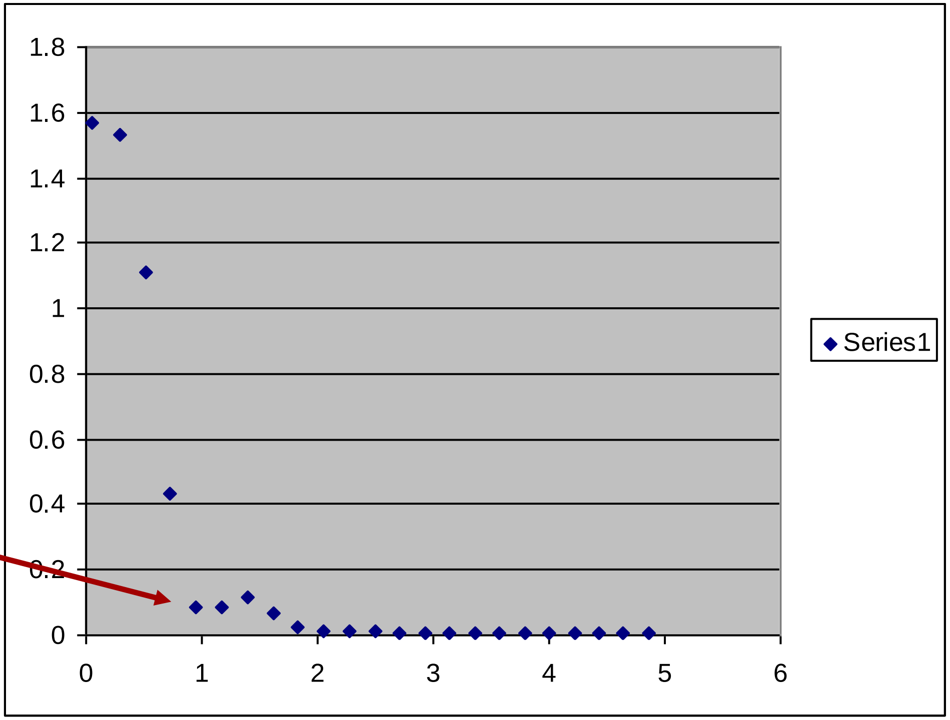
- Turning 90 deg
- P=2, D=0.1, I=0
- 3 times faster.
- But: now it overshoots.
- Try to add some D to reduce overshoot.



1.5 secs

Tuning

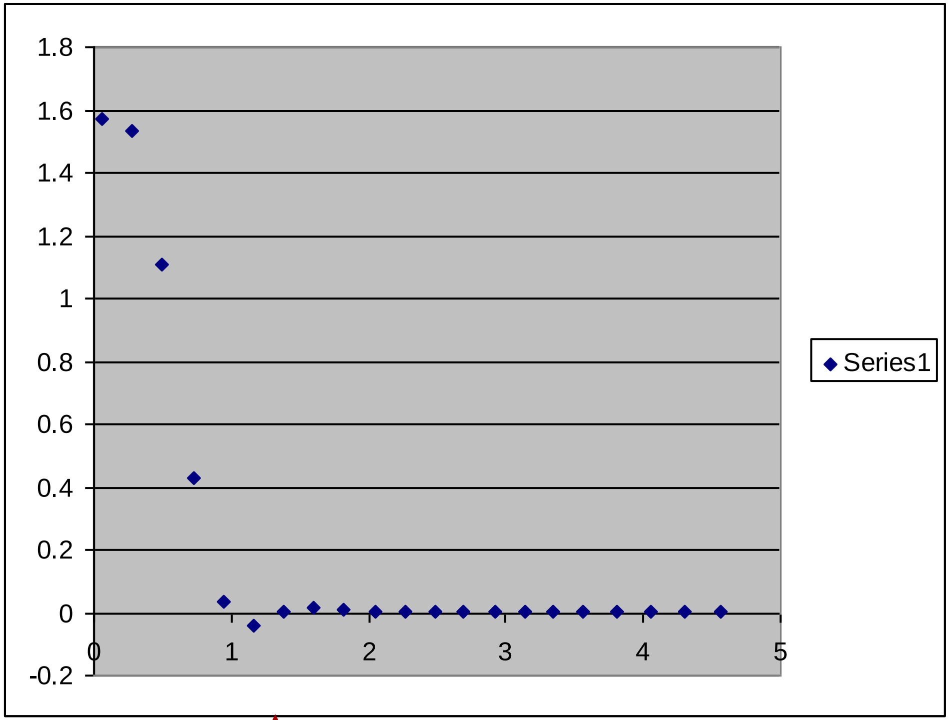
- Turning 90 deg
- P=2, D=0.3, I=0
- Too much D leads to slamming on the brakes early.
  - Robot seems to bounce off the goal.



2.0 secs

Tuning

- Turning 90 deg
- P=2, D=0.15, I=0
- Best compromise.



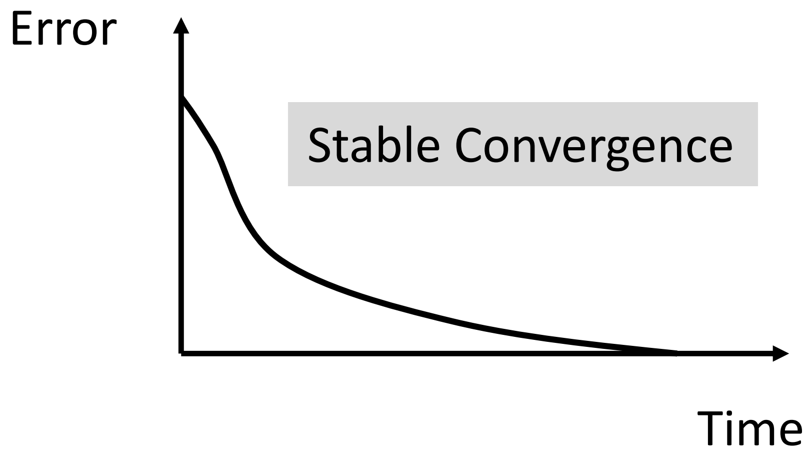
1.25 secs

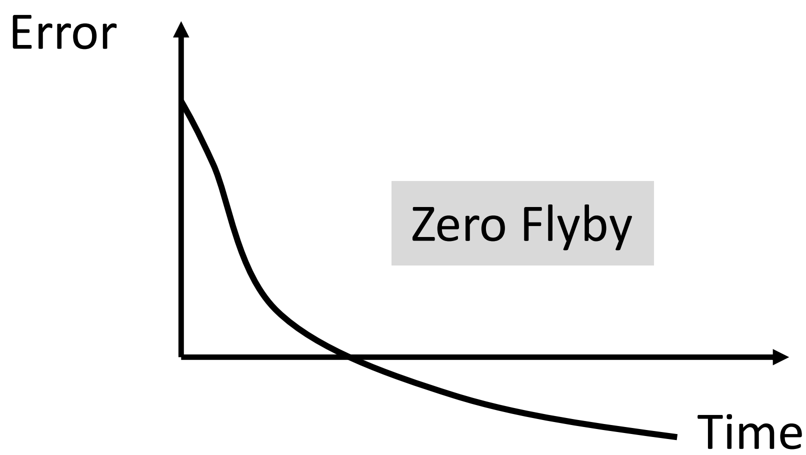
Tuning

- Its not about beautiful code!
  - Beautiful code helps you maintain it but it won't prop up a fundamentally bad approach.
  - There is no point maintaining it if it does not work.
- Behavior is hard to understand and hard to quantify.
  - Get QUANTITATIVE when possible. It beats subjective assessments in most cases.
- Basic approach is ……. wait for it …….
  - R.E.C.O.R.D……. S.O.M.E……. D.A.T.A.
  - And study it
- MATLAB makes this easy

Termination & Convergence

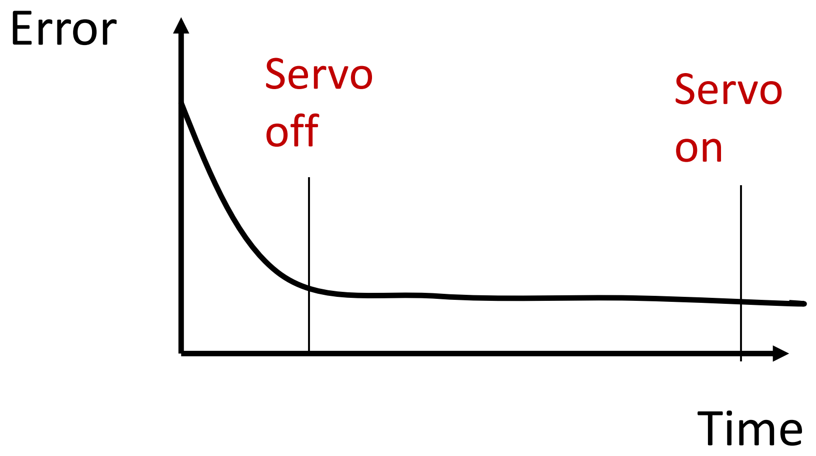
- In general, you should check error <u>velocity</u> as well as error itself. Why?
  - Could have a lucky sample while flying past the goal
- If you do that, might as well implement the derivative term of PID.
  - Can always set the gain to zero.

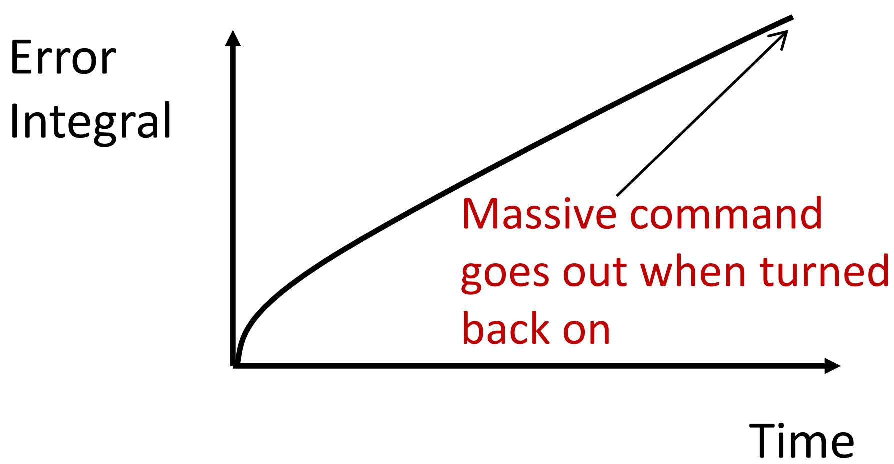




Beware Last Cycle Memory

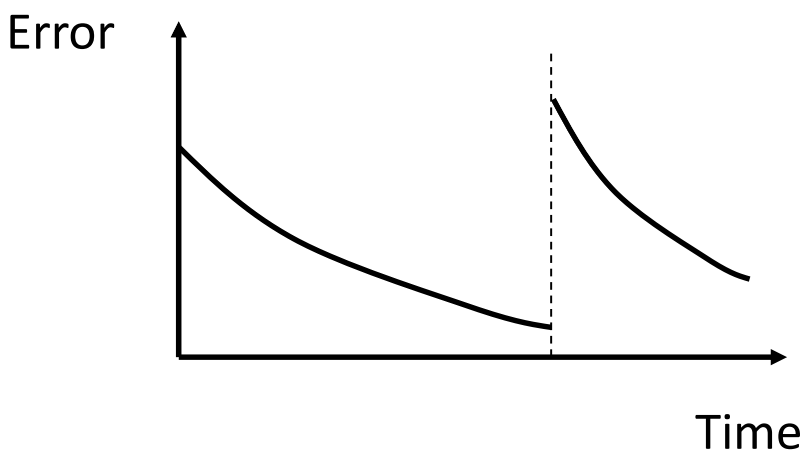
- Last cycle memory (for derivatives and integrals) can be tricky when starting and stopping servos.
- For integrators, the key thing is that the integrator must be either or both of:
  - shut off
  - nulled

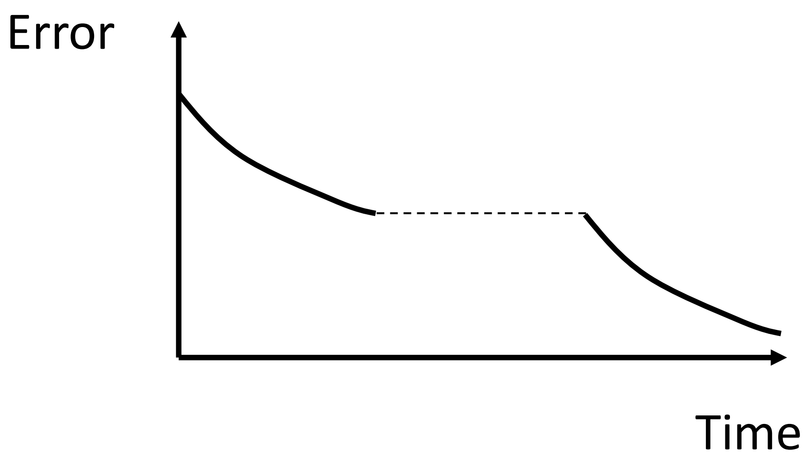




Beware Derivatives Too…

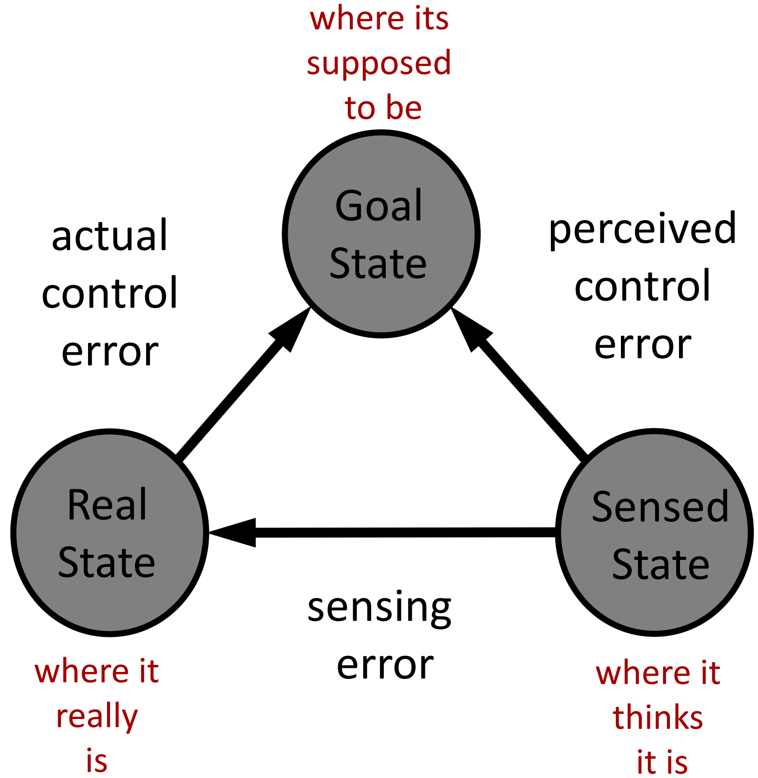
- Derivatives can blow up:
  - When the system is moved while not under control.
- Derivatives can also attenuate:
  - When system is stopped for long period of time.





Control Versus Sensor Error

- The robot cannot tell if its pose is in error.
- Sensing Error =
  real pose – sensed pose
- Perceived Control Error =
  goal pose – sensed pose.
- Actual Control Error =
  goal pose – real pose.
- Your job is usually to make sure the <u>perceived control error</u> is small.
- How can you make sensing error smaller?



Agenda Today

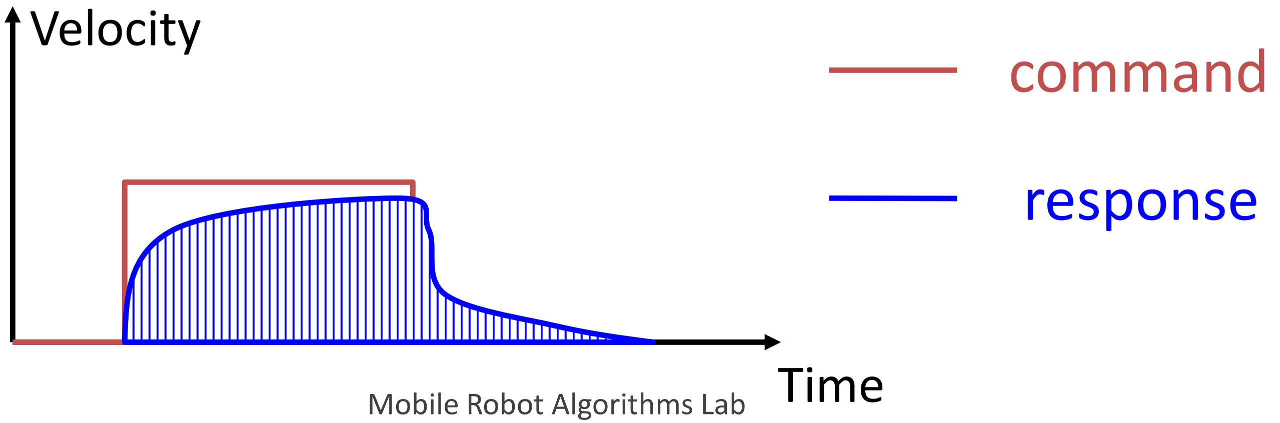
- Administrative Issues
- Feedback Control
- Tuning
- **Model Reference Control**
- Identification
- Feedforward Control
- 2 Dof Control
- Overview of Lab 4

Weal World Wobots
(are waskewy)

- Effect 1: Dynamics
  - It often takes much longer to do anything because real systems have dynamics (are low pass filters).
- Effect 2: Comms Delays:
  - It takes two cycles for response to commands generated in this cycle to be measurable.
  - What is a simple way to compensate for delays?
- Effect 3: Hierarchy / Cascading
  - There are lower control levels below yours. See next.

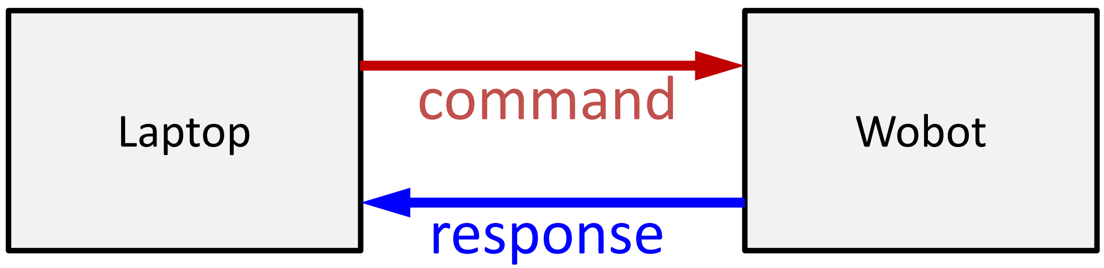
Effect 1: Dynamics (Momentum)

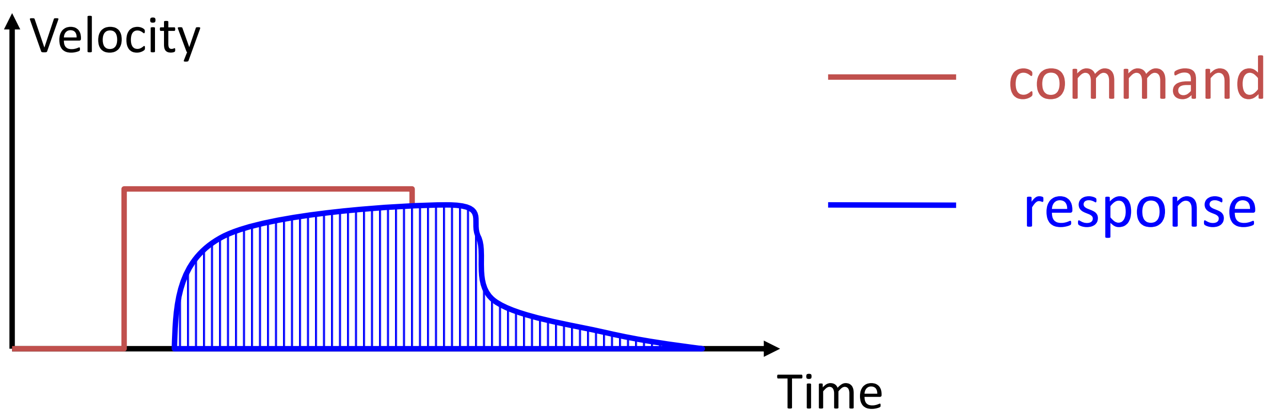
- Robots have mass and hence momentum.
  - A form of filter you have little control over.
- They do not follow their (velocity, position) “commands” precisely.
  - Yet, Newton’s laws will certainly hold.
  - If you knew the actual forces and inertias and frictions …
- Its both good and bad
  - Attenuates disturbances
  - Attenuates your commands



Effect 2: Delays

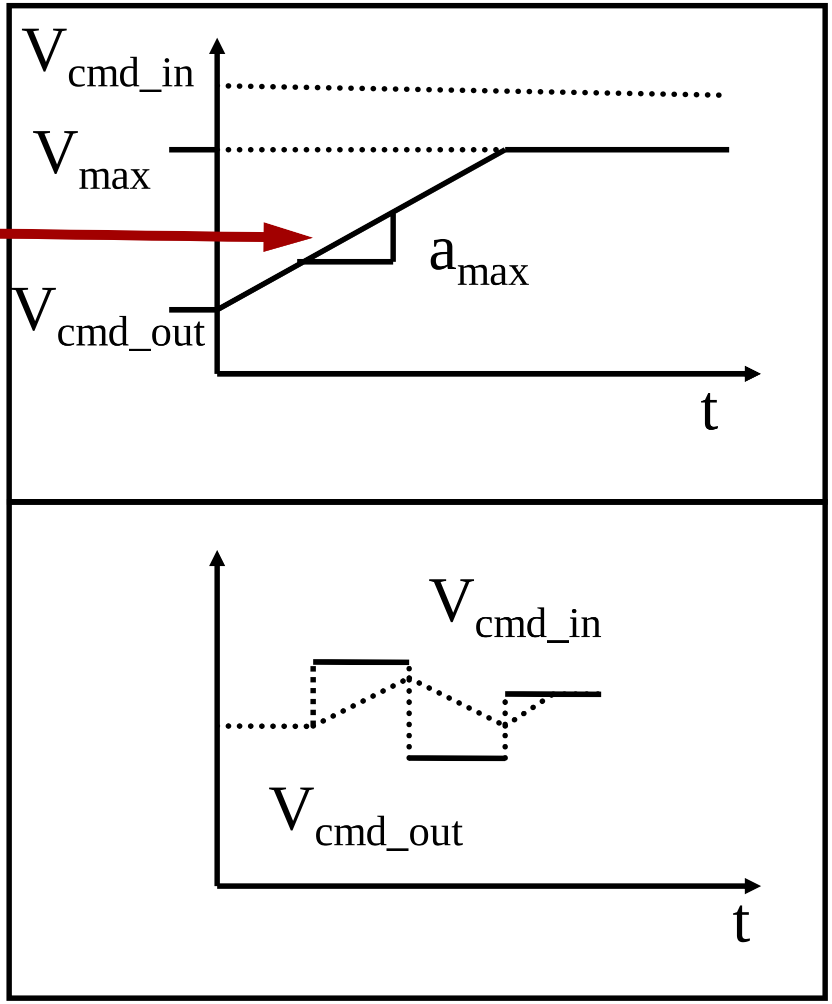
- Do not expect the feedback to change for a few cycles after a new command is issued.



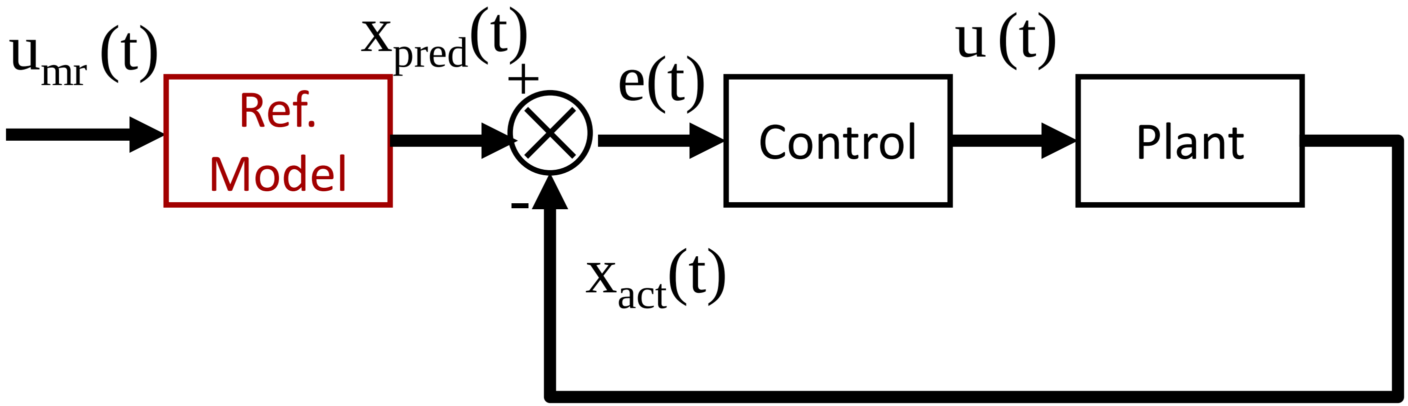


Effect 3: Lower Control Levels

- Commanded acceleration is limited.
  - Response for a step speed input is often a ramp.
- Commanded speed is limited.
- That gives an odd sort of filter for a continuously varying input.
- Actual robot response to the black line is a separate matter.



Model Reference Control



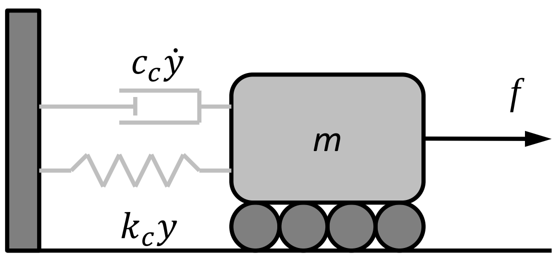
$$e(t) = x_{pred}(t) - x_{act}(t)$$

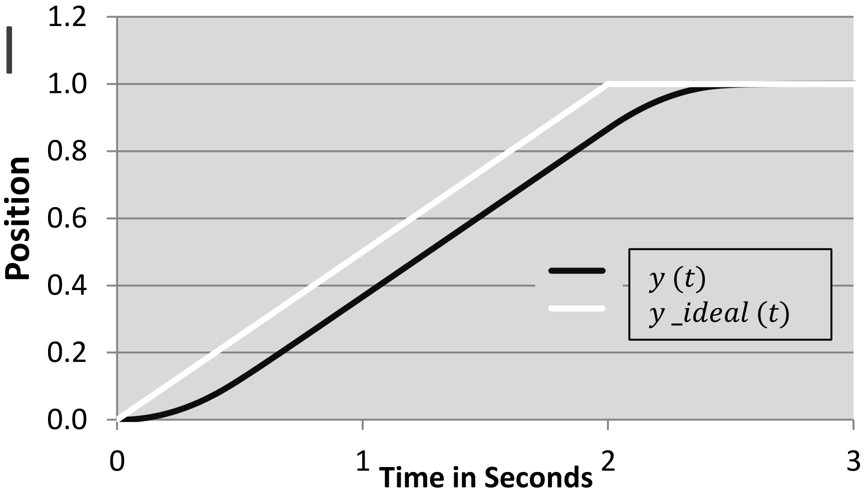
$$u(t) = k_p e(t) + k_d \frac{d}{dt} e(t) + k_i \int e(t) dt$$

- Basic idea is to form errors w.r.t. predicted performance of “reference model”.

Example

- Model Reference Control of Point Mass.
  - For a little better model





- 2.5 secs. versus 5 secs. (for critically damped feedback) and no overshoot!
  - Maybe this is a good idea!
  - Where does the reference model come from?
    - 1: Make it up
    - 2: Model the real system – system identification.

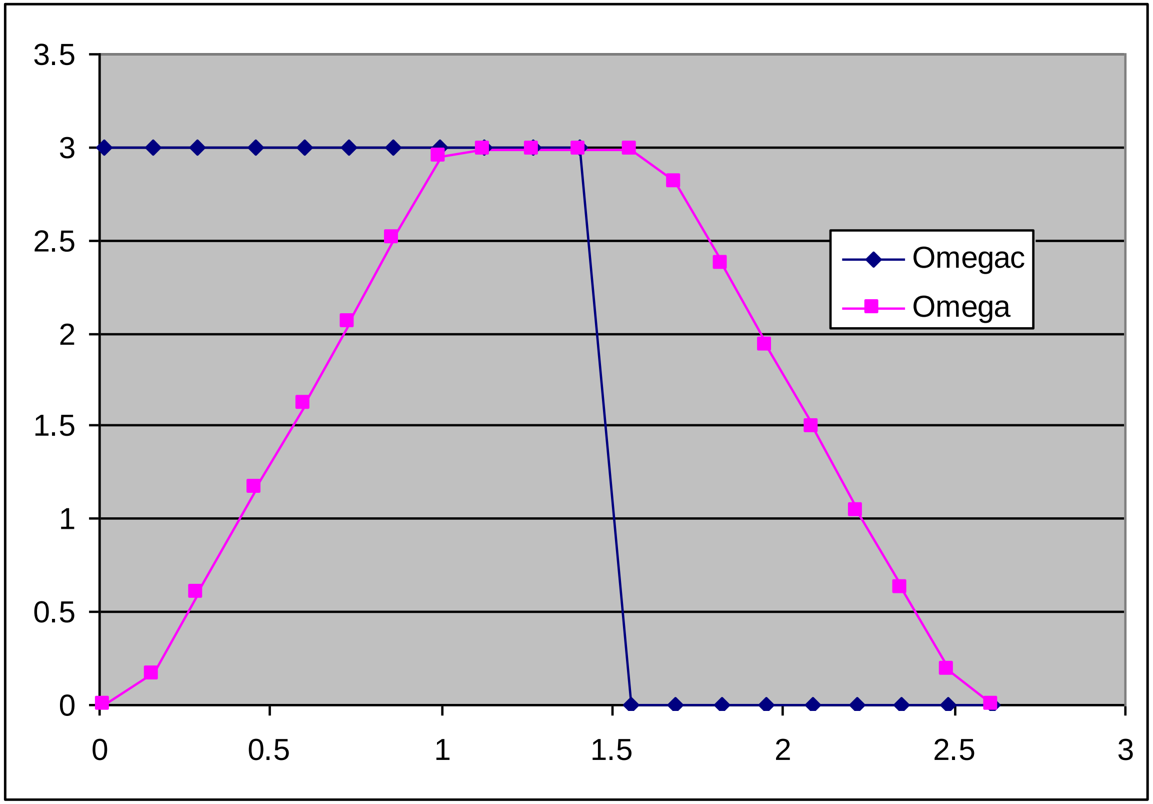
Agenda Today

- Administrative Issues
- Feedback Control
- Tuning
- Model Reference Control
- **Identification**
- Feedforward Control
- 2 Dof Control
- Overview of Lab 4

Rotary Identification

(Example From Nomad Robot)

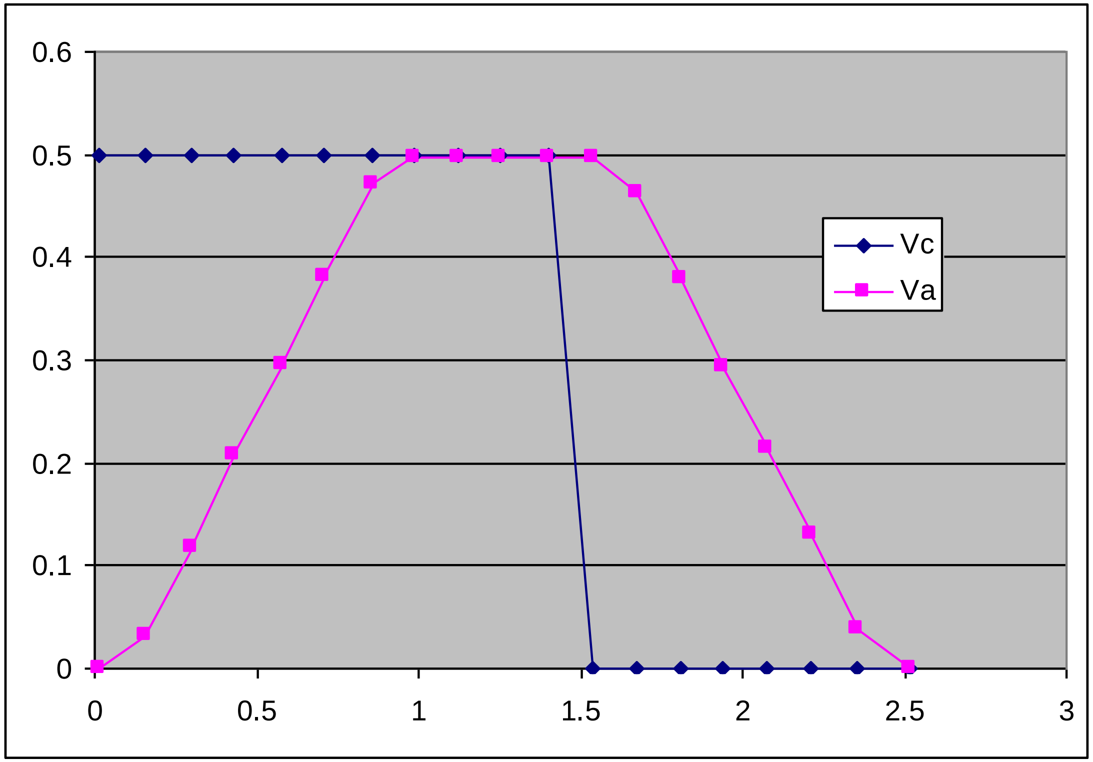
- Notice:
  - Delay (~ 0.15 secs)
  - Acceleration limit (= ~3 rads per sec per sec)



Linear Identification

(Example From Nomad Robot)

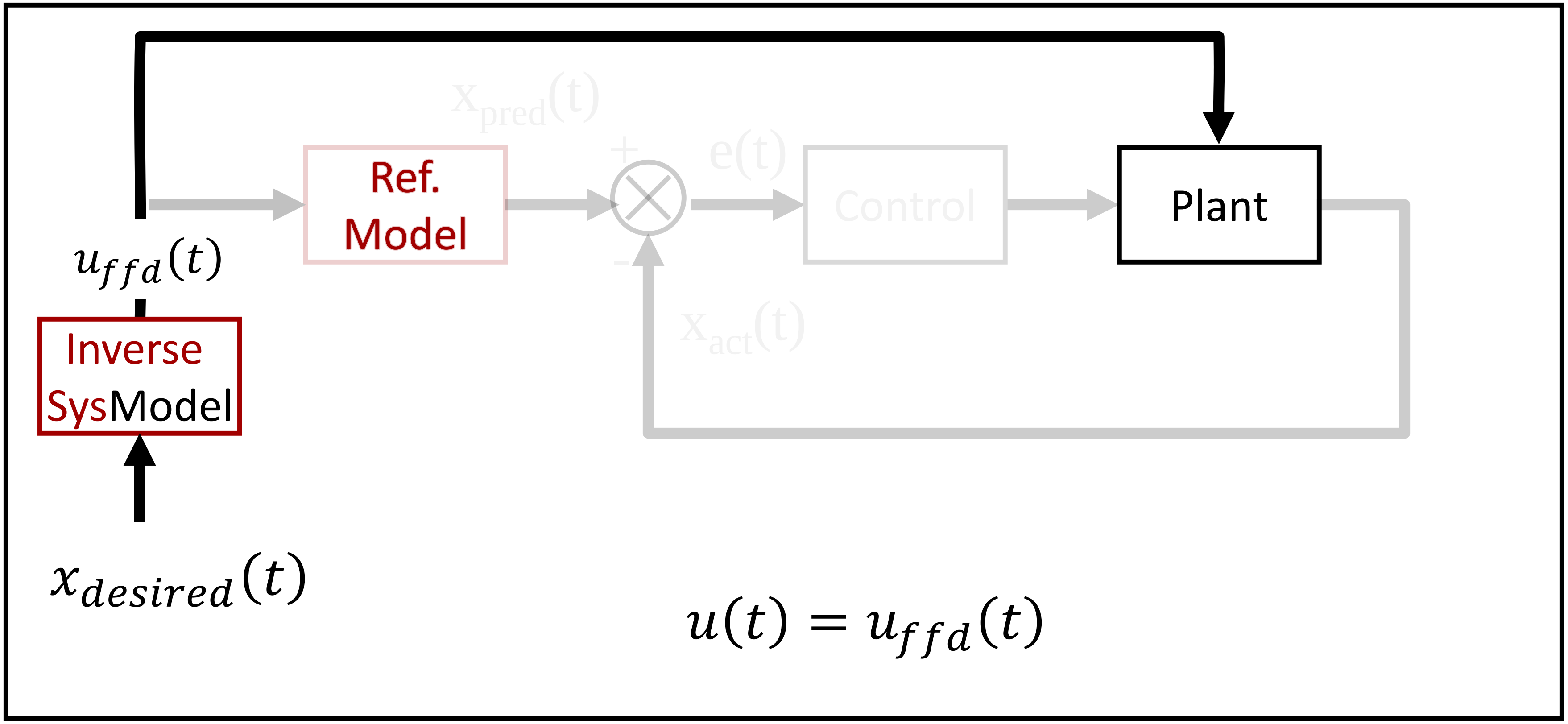
- Notice:
  - Delay (~ 0.15 secs)
  - Acceleration limit (= 0.5 m.sec)



Agenda Today

- Administrative Issues
- Feedback Control
- Tuning
- Model Reference Control
- Identification
- **Feedforward Control**
- 2 Dof Control
- Overview of Lab 4

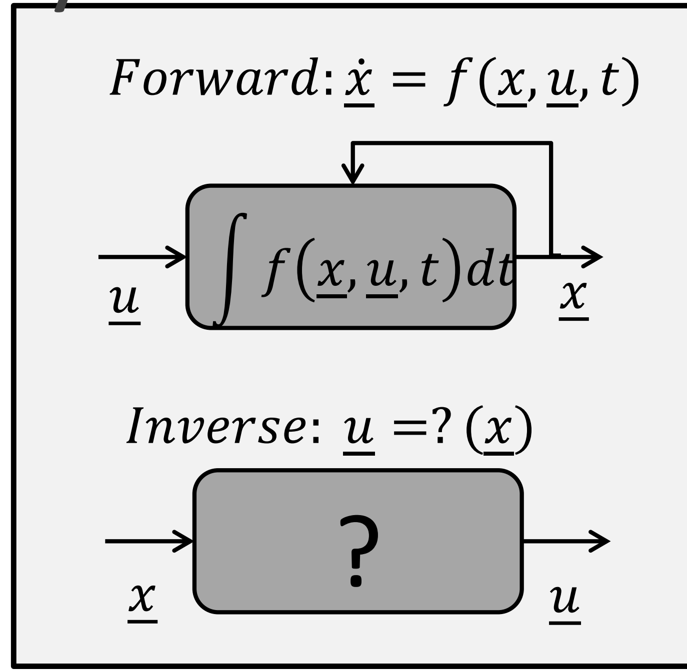
Feedforward Control

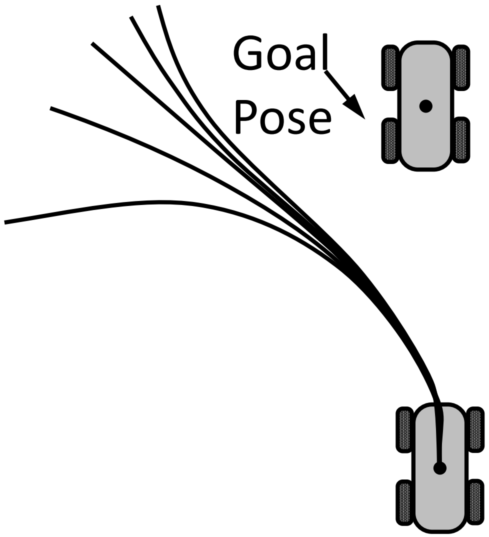


- Basic idea is know what to ask for in order to get what you want.
  - Not only do you have a reference model, but you invert it too.
- What is $x_{desired}(t)$ you ask? – any curve that starts where you are and ends where you want.

Inverting is not so easy….

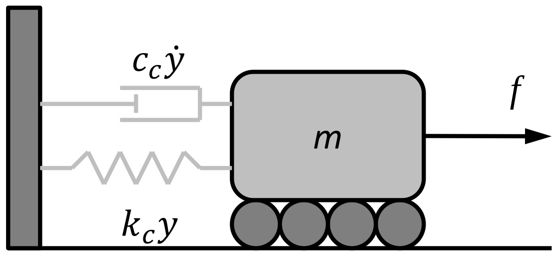
- How do you invert a differential equation?
- I.e. how do you choose the input that produces you output you want?
- There may be no solution….

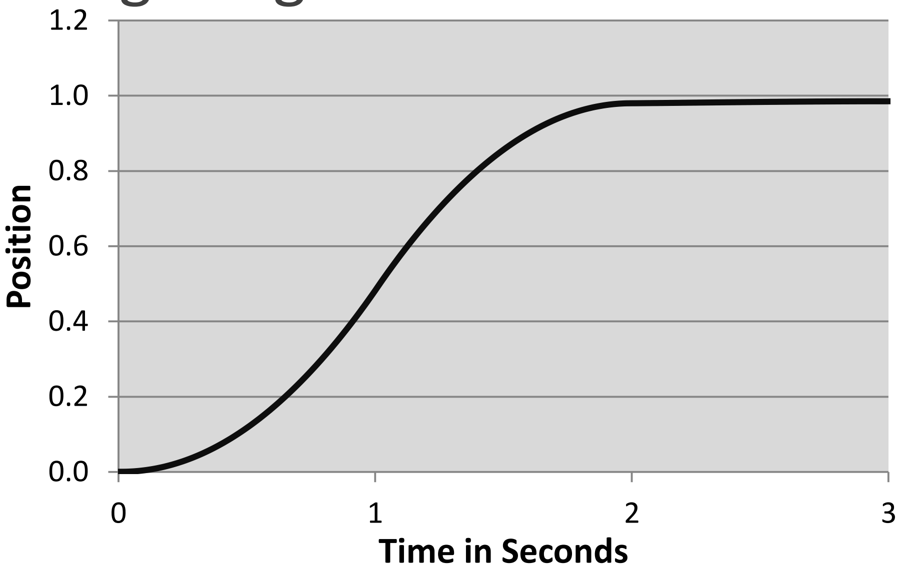




Example – that really works

- Feedforward Control of Point Mass
  - In this case it is also a "bang bang" control





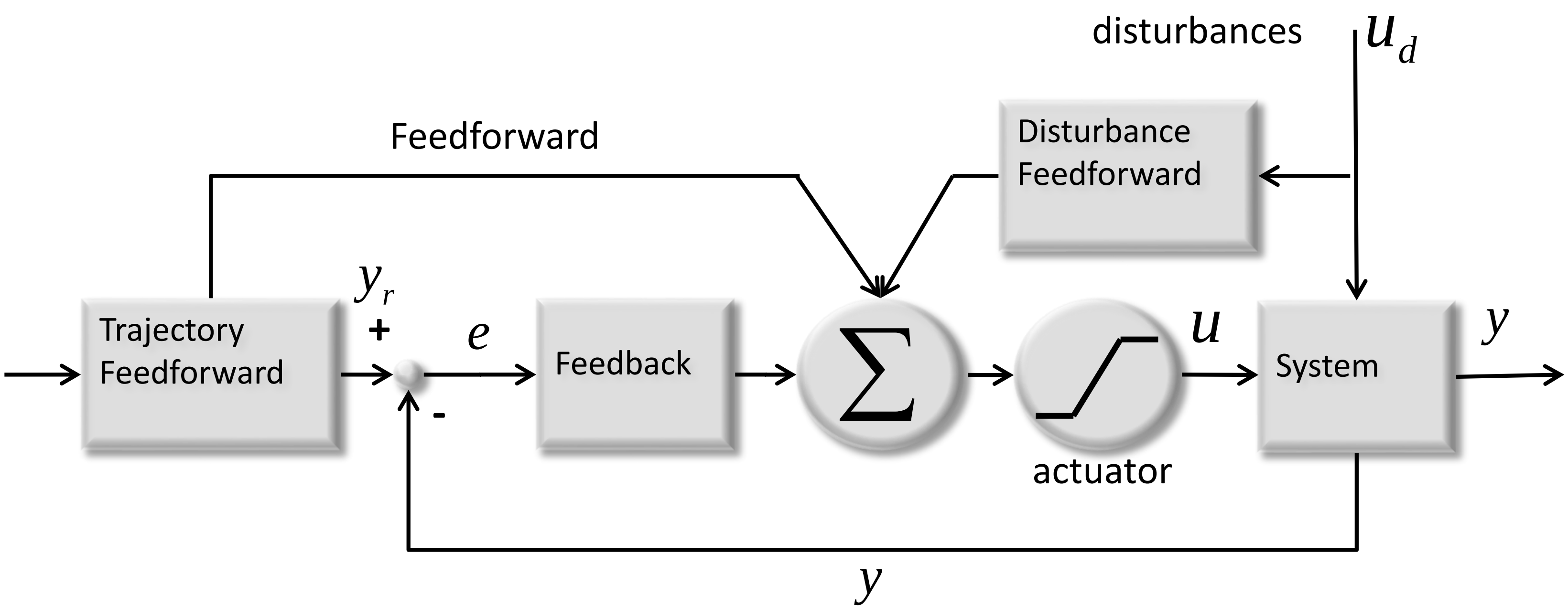
- This is <u>the</u> time optimal control (2 secs) for this case.
- Issue: What if there is (unknown) friction?

Agenda Today

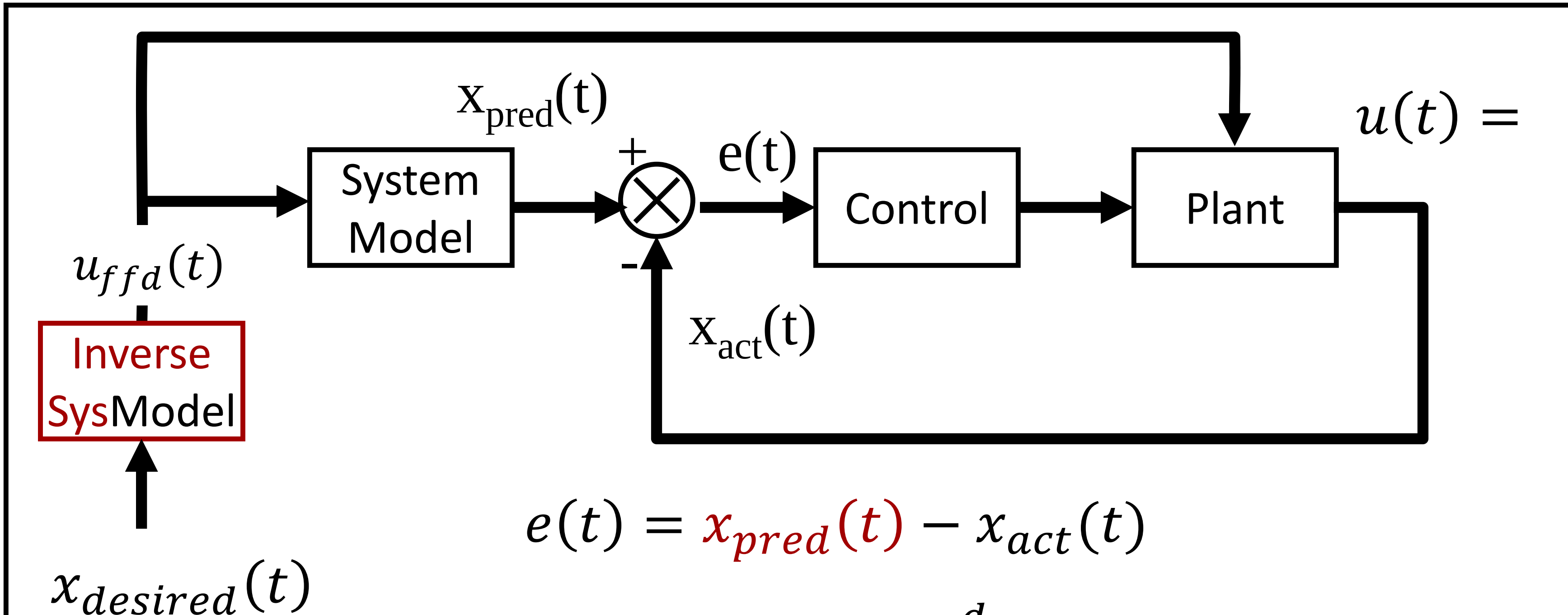
- Administrative Issues
- Feedback Control
- Tuning
- Model Reference Control
- Identification
- Feedforward Control
- **2 Dof Control**
- Overview of Lab 4

Two Degree of Freedom
(Feedforward with Feedback Trim)

- Do your best to determine the input which causes the system to go where you want open loop.
  - Predict even the disturbances if that is possible.
- Form errors on the predicted response.



2 DOF Control



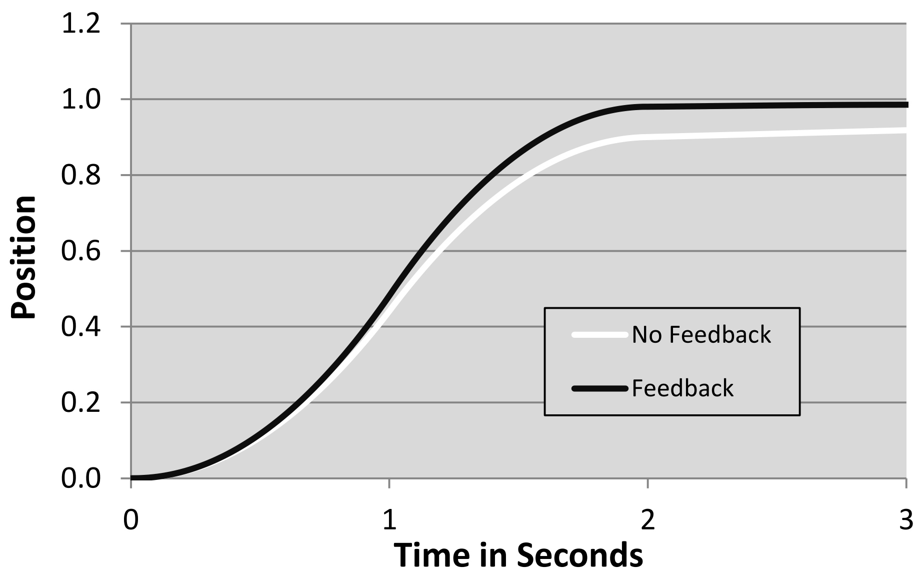
$$e(t) = x_{pred}(t) - x_{act}(t)$$

$$u(t) = u_{ffd}(t) + k_p e(t) + k_d \frac{d}{dt} e(t) + k_i \int e(t) dt$$

- Controller has a feedforward term and a feedback term.
  – Feedforward works perfectly if there are no errors.
  – Feedback removes the (small) errors if any occur.

Final Result

- An optimal trajectory that also rejects disturbances.

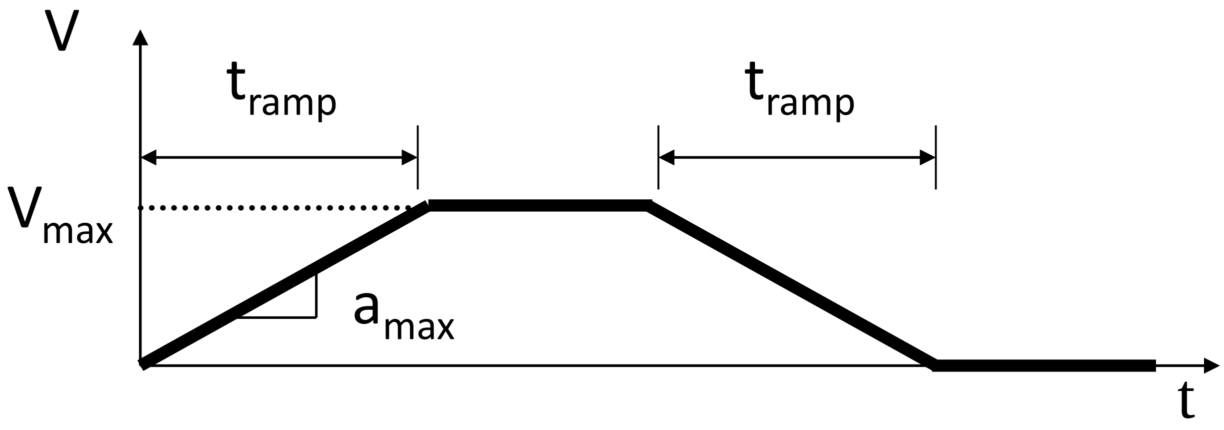


Bottom Line

	Feedback	Feedforward
Removes Unpredictable Errors and Disturbances	(+) YES	(-) NO
Removes Predictable Errors and Disturbances	(-) NO	(+) YES
Removes Errors and Disturbances Before They Happen	(-) NO	(+) YES
Requires Model of System	(+) NO	(-) YES
Affects Stability of System	(-) YES	(+) NO

This Lab

- Command a ffwd trapezoidal velocity profile:
  - with a terminal V=0 period 1 second long.
- Assume the system will follow a trapezoidal velocity profile with delay.
- Compare the real response to the "expected" and remove the errors with feedback.



Preparation

- Did everyone read lab 4 before today?

Lab 4 Preparation

- [Click for Lab4 Writeup]