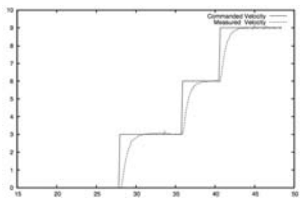
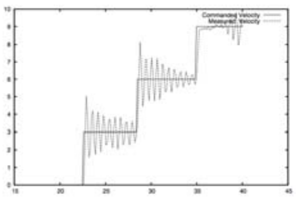


Motor Control



RSS Lecture 4

Tuesday, 16 Feb 2010

Prof. Seth Teller

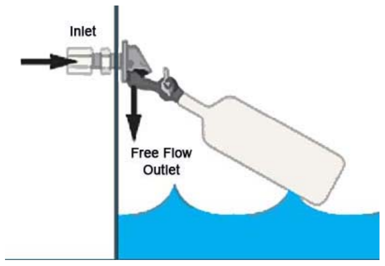
Jones, Flynn & Seiger § 7.8.2

Today: Control

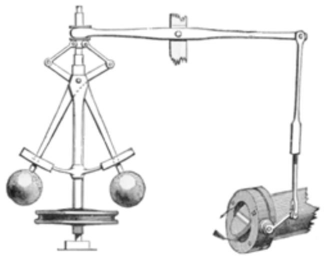
- Early mechanical examples
- Feed-forward and Feedback control
- Terminology
- Basic controllers:
  - Feed-Forward (FF) control
  - Bang-Bang control
  - Proportional (P) control
  - The D term: Proportional-Derivative (PD) control
  - The I term: Proportional-Integral (PI) control
  - Proportional-Integral-Derivative (PID) control
- Gain selection
- Applications

What is the point of control?

- Consider any mechanism with adjustable DOFs* (e.g. a valve, furnace, engine, car, robot…)
- Control is *purposeful variation* of these DOFs to achieve some specified *maintenance state*
  - Early mechanical examples:† ,



www.freshwatersystems.com



wikimedia commons

*DOFs = Degrees of Freedom

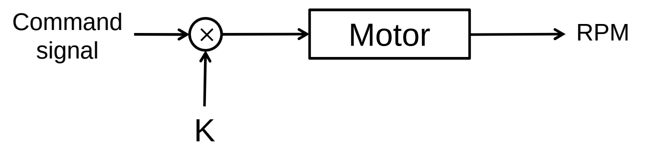
†Note blanks on your printed slides!

The Role of Control

- Many tasks in robotics are defined by (high-level) *achievement* goals requiring *planning*:
  - Go to the exit of the maze
  - Push a box around some obstacles to a goal location
  - Pass a car that's stalled on the road shoulder
- Other tasks in robotics are defined by (low-level) *maintenance* goals requiring *control*:
  - Drive at 60 mph (or in RSS, roll forward at 0.5 m/s!)
  - Keep to the center of the lane indefinitely
  - Follow some trajectory computed by the planner
  - Balance on one leg
- Today's focus is control; planning in a few weeks

Feed-Forward (FF) Control

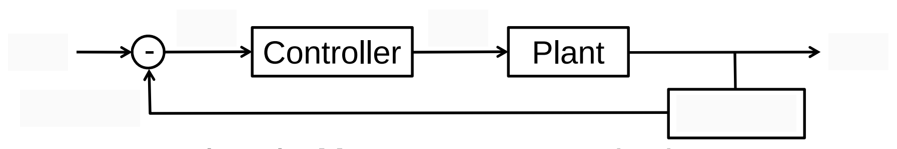
- Pass command signal from external environment directly to the *loaded element* (e.g., the motor)
- Command signal typically multiplied by a *gain* K



- … Where does the gain value K come from?
  -
- Under what conditions will FF control work well?
  -
- You will implement an FF controller in Lab 3

Feedback Control Terminology

- *Plant* **P**: process commanded by a *Controller*
- *Process Variable* **PV**: Value of some process or system quantity of interest (e.g. temperature, speed, force, …) as measured by a *Sensor*
- *Set Point** **SP**: Desired value of that quantity

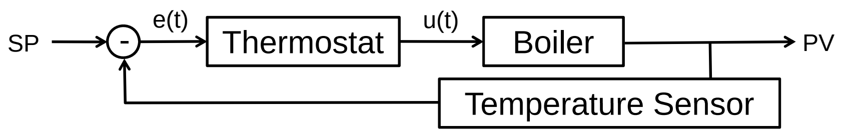


- *Error signal* **e(t)** = SP-PV: error in the process variable at time t, computed via *Feedback*
- *Control signal* **u(t)**: controller output (value of switch, voltage, PWM, throttle, steer angle, …)

*Set point is sometimes called the “Reference”

Example: Home Heating System

- *Plant* P: Boiler with on-off switch (1 = all on ; 0 = all off)
- *Process Variable* PV:
- *Controller:* *Sensor:*
- *Set Point* SP:
- *Control signal*:

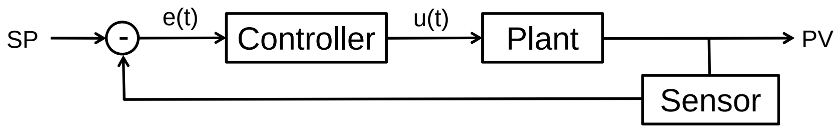


How could the function **u(t)** be implemented?

This is called " **control**." Would it work well?

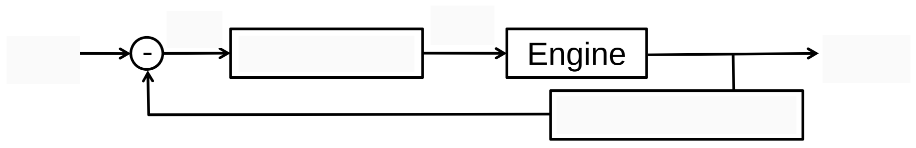
Proportional Control

- Suppose plant can be commanded by a *continuous*, rather than discrete, signal
  - Valve position to a pipeline or carburetor
  - Throttle to an internal combustion engine
  - PWM value to a DC motor
- What's a natural thing to try?
  - *Proportional* (P) C*ontrol*: make the command signal



Example: Cruise Control (CC) System

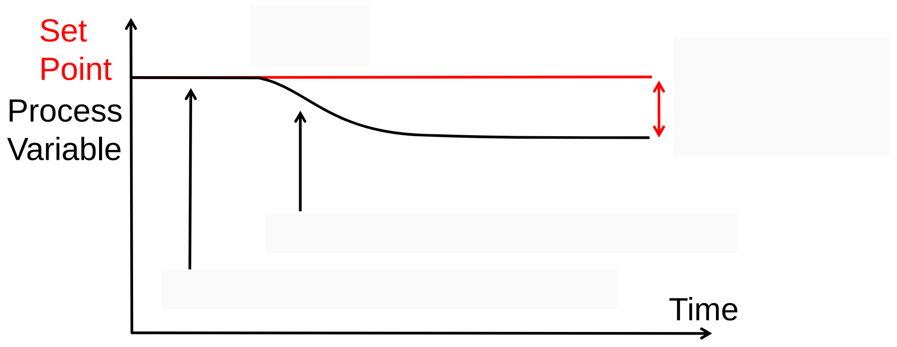
- *Plant* P: Engine with throttle setting $u \in [0..1]$
- *Process Variable* PV:
- *Controller:* *Sensor:*
- *Set Point* SP:
- *Control signal*:



Define e(t) = , u(t) = ,
i.e. Throttle =
Does this controller “settle” at the desired speed?

Proportional Control: Why SSE?

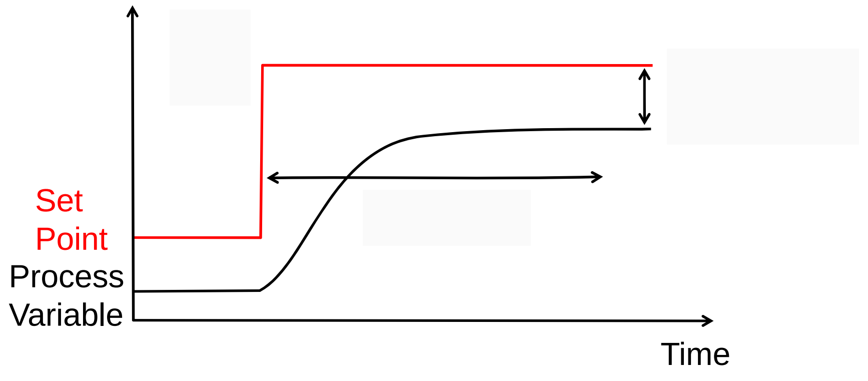
- Suppose e(t) = 0. Then u(t) =
- Process Variable
- But any real physical system has a
- Deviation,



Why not just introduce constant term, $u(t) = A + K_P * e(t)$?

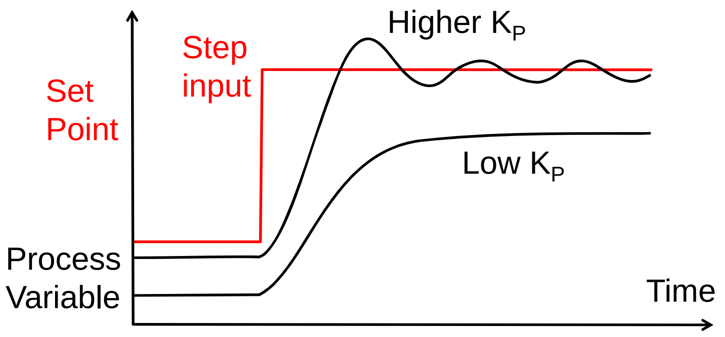
Proportional Control Step Response

Notional plot and terminology:



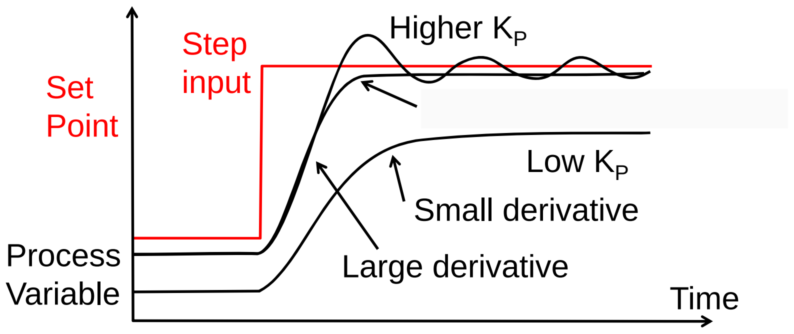
Proportional Control and SSE

- Can combat SSE by ("the P gain")
- This gives a and
- But the gain too much leads to



Combatting Overshoot: The D Term

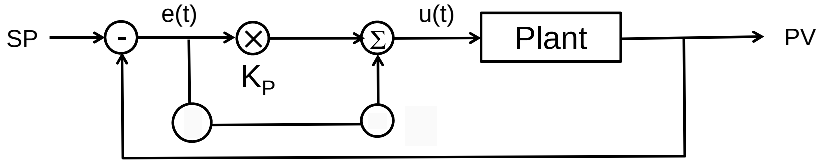
- Note the *derivative* of error in responses below
- it from output to counteract overshoot
- Then u(t) = $K_P \times e(t)$ +
  - K_D the "derivative" or "damping" term in PD controller



- … But still haven't eliminated steady-state error!

Combatting Steady-State Error: I Term

- Idea: apply correction based on *integrated* error
  - If error persists, integrated term will grow in magnitude
  - Sum proportional and integral term into control output

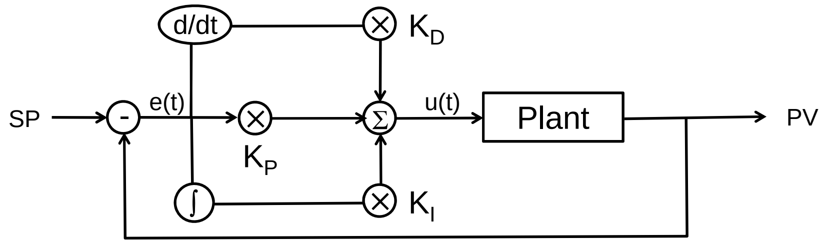


Then u(t) = $K_P \times e(t)$ + (where the integral of the error term is taken over some specified time interval)
This produces a *proportional-integral* (PI) controller

Incorporating the I term eliminates SSE by modulating the plant input so that the .
You'll hear robotics people speak of controller "wind-up"

Putting it All Together: PID Control

- Incorporate P, I and D terms in controller output
  - Combine as a weighted sum, using gains as weights



Then u(t) = + +
This is a "proportional-integral-derivative" or *PID controller*

Are gains *unitless*? …. interpretable as *physical quantities*?

How to determine effective gain values?

- P controller: search 1-D space of gains K_P
  - Identify various behavior regimes; you'll do this in Lab 3
- Choose analytical or empirical approach (how?)
- Hybrid: Ziegler-Nichols Tuning Method (Heuristic)
  - Useful in absence of a system model (if ,)
  - Start with pure P control (how?); Increase K_P until system *oscillates indefinitely*; note critical gain K_C and period T_C
  - Then for P, PI, or PID control, set gains as follows:

	K_P	K_I	K_D
P	$0.5\ K_C$		
PI	$0.45\ K_C$	$1.2\ K_P\ /\ T_C$	
PID	$0.6\ K_C$	$2\ K_P\ /\ T_C$	$K_P T_C\ /\ 8$

  - Yields acceptable but not optimal controller behavior

Other Applications of Feedback Control

- Mobility:
  - Lane-keeping
  - Trajectory-following
  - Standoff maintenance
- Manipulation:
  - Maintaining a steady contact force for grasping
  - Holding a mass at a certain location or attitude
  - Pushing a sliding object at constant velocity
- Sensing:
  - Automatic gain control, white balance, etc.
  - Target-tracking for active vision (body, head, eyes…)
- Many, many more

To Think About

- Lab 4 involves following a hand-held ball
- Lab 5 involves moving alongside a solid wall
- Lab 7 involves picking up a block from the ground
- How might you use feedback control to implement any of these behaviors?
- What sensor(s) would you use, and what sort of error signal(s) would you infer from them?
- What would your robot's behavior look like?

What’s Next?

- For more on control, consider taking any of:
  - 2.003, 2.004, 2.086, 2.12, 2.14x, 2.151, 2.152, 2.830, …
  - 6.01, 6.003, 6.011, 6.142, 6.231, 6.241, 6.243, 6.832, …
  - 16.06, 16.30, 16.31, 16.301, 16.32x, 16.72 (ATC!), …
  - 9.05, 9.272, 10.450, 10.976, HST.545, …

- Today in lab: Team briefings for Lab 2
- Today & W in Lab 3: implementing controllers
- Tomorrow in lecture: Cameras, low-level vision
- Reminder: Individual PAR due Friday by 5pm
- Lab 3 wiki materials and briefings due M 22 Feb