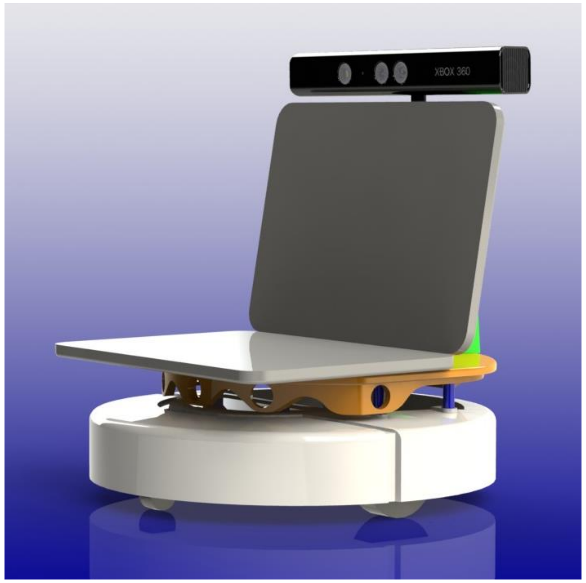
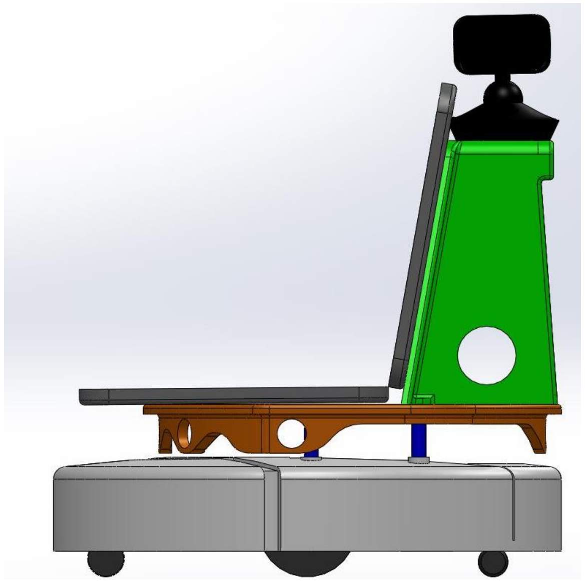
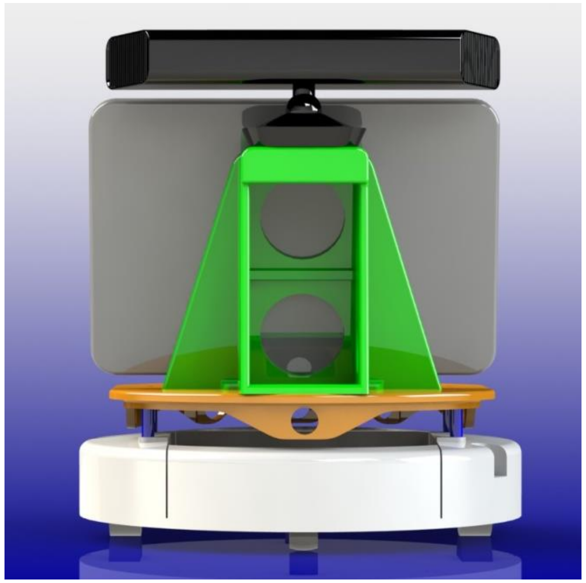
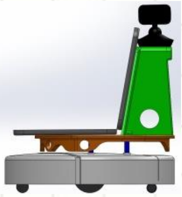
MAE 493G, CpE 493M, Mobile Robotics

10. Robot Control



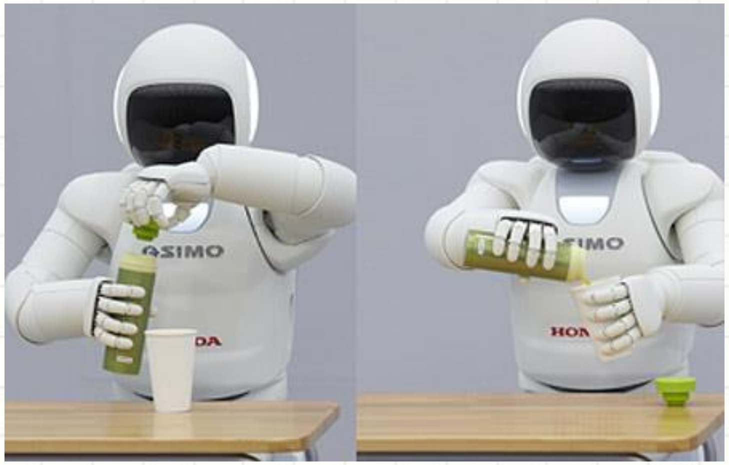



Instructor: Yu Gu, Fall 2013



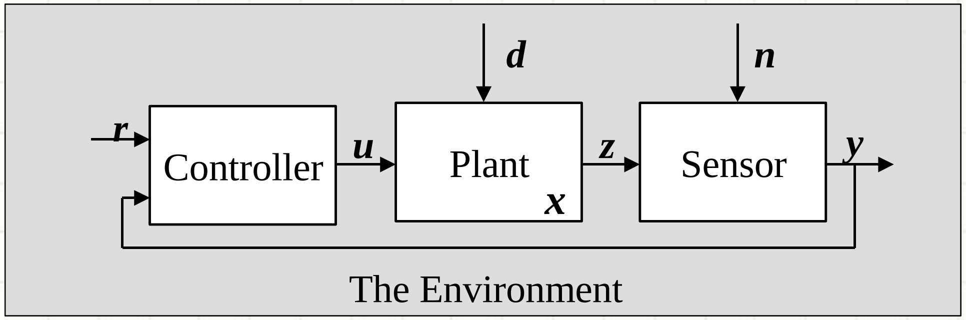
Mobile Robot Control

- Automatic control is an important aspect of mobile robotics;
- After all, a mobile robot needs to move, and this movement needs to be properly controlled;
- As robots get more complex and with higher requirements, the need for advanced control theory increases;
- Control is also tightly related to sensing, estimation, reasoning, and planning;
- For example, we can use vision data to guide our controller while controlling the camera position to achieve a better view...



Control System

- A control system is a ‘system’ that includes the controller, the plant (process or device to be controlled), actuators, and sensors.



- There are many reasons for using automatic control:
  - To do tedious jobs;
  - Handle problems of different (physical, power, and time) scale;
  - To achieve higher precision;
  - Compensate disturbance;
  - Operate remotely or autonomously.

r – reference inputs or setpoints;
u – plant inputs/control commands;
x – system states;
z – plant outputs/controlled variables;
y – output measurements;
d – disturbances (exogenous input);
n – measurement noises.
“*Plant*” here include the process and actuators

- Controllers today are still not nearly as intelligent or flexible as humans;
- That’s why we still have pilots (but no more elevator operators...).

Open and Closed-Loop Control

- An *open-loop controller* predicts what may happen (based on models) and issue a series of (*feed-forward*) command without checking the current state;
- A *closed-loop controller* use sensor outputs for *feedback* and adjust the controller output accordingly;
- Open loop controllers are often (but not always) simpler and faster;
- Closed-loop controllers have a different set of advantages:
  - Handle uncertainty and disturbance;
  - Stabilize an unstable system;
  - Linearize a nonlinear system (sometimes).

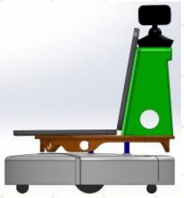
Open-Loop Control – Feed-Forward Control – Deliberative Control

Closed-Loop Control – Feedback Control – Reactive Control

Two Degrees of Freedom Control – Hybrid Approach

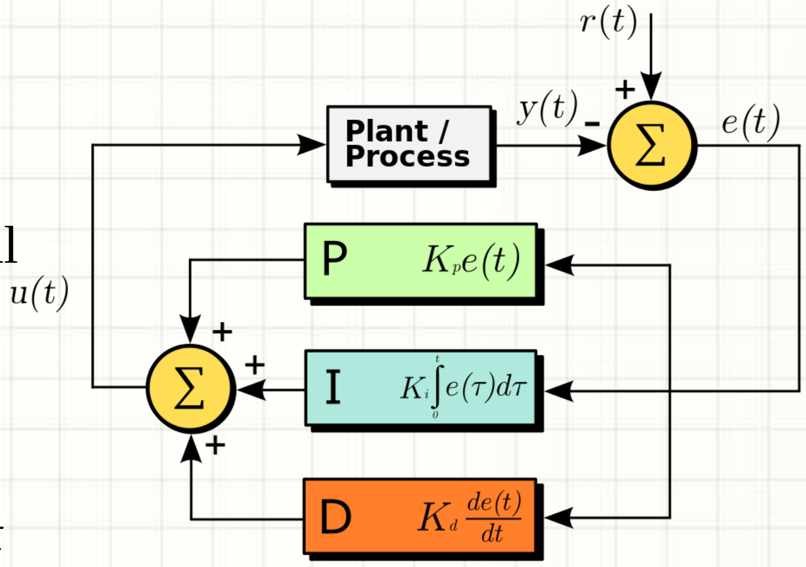
Controller Classifications

- Analog control and digital control;
- Logic control (e.g. if a>b, turn left) and continuous control;
- System classifications:
  - Linear Time Invariant (LTI); linear systems satisfy the properties of superposition and scaling; (physical systems are not really LTI)
  - Linear time varying;
  - Nonlinear;
  - Single-Input-Single-Output (SISO);
  - Multiple-Input-Multiple-Output (MIMO);
- Controller Types: PID control, optimal control, adaptive control, model predictive control, robust control, nonlinear control...
- Regulator (maintaining a designated characteristic) and tracker (track a changing reference input).

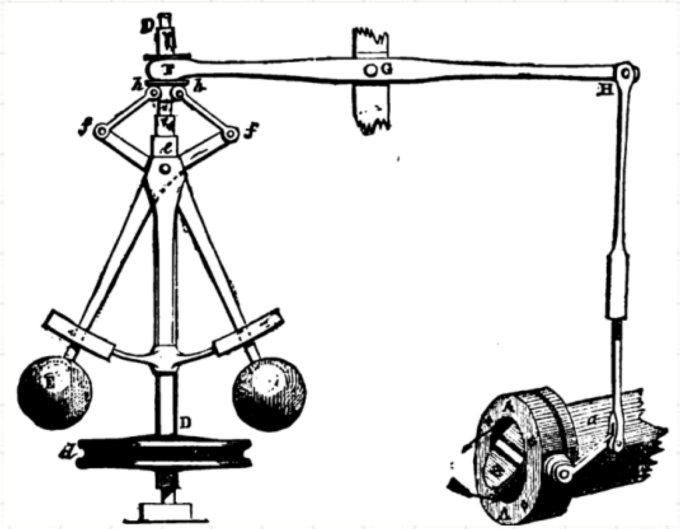


PID Controller

- A *proportional-integral-derivative (PID) controller* is a *negative feedback* control method widely used in industrial systems;
- The proportional, integral, and derivative values can be interpreted in terms of time: *P* depends on the *present* error, *I* on the accumulation of *past* errors, and *D* is a prediction of *future* errors, based on current rate of change;
- The *weighted sum* of these three actions is used as the control input;
- PID control is a simple but highly useful (and popular) control technique.



Proportional Controller



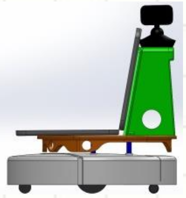
- If we ignore the integral and derivative components of the PID controller for a moment, the *proportional controller* has a very simple form:

$$u(t) = K_p e(t) = K_p \left(r(t) - y(t) \right)$$

- In other words, the control input (controller output) is proportional (with a gain K_p) to the *error*, which is the difference between the reference input and the sensor measurement;

- In many cases, a proportional controller is perfectly adequate. In other cases, a proportional controller may not meet the desired *transient response* and *steady state error* requirements;

- You can try in Simulink some examples of proportional control.

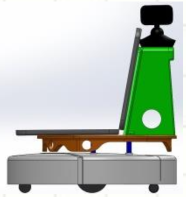


Proportional Derivative Controller

- A *Proportional and Derivative (PD) controller* has two sets of gains:

$$u(t) = K_p e(t) + K_d \frac{d}{dt} e(t)$$

- The derivative of the error is calculated by determining the slope of the error over time. The magnitude of the contribution of the derivative term to the overall control action is termed the *derivative gain*, K_d.
- Derivative action predicts system behavior and thus improves *settling time* (time for the error to settle down) and *stability* of the system;
- However, the derivatives are inherent sensitive to the measurement noise;
- Large, sudden changes in the error (which typically occur when the reference input is changed) cause a sudden, large control action stemming from the derivative term (*derivative kick*). Therefore PD controller is often used with a *low pass filter*:



Proportional Integral Controller

- A *Proportional and Integral (PI) controller* also has two sets of gains:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau$$

where, K_d is the integral gain;

- The contribution from the integral term is proportional to both the *magnitude* and *duration* of the error. The integral in a PI controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously;
- The integral term accelerates the movement of the process towards the reference input and *eliminates the residual steady-state error* that occurs with a pure proportional controller;
- However, since the integral term responds to accumulated errors from the past, it can cause the present value to *overshoot* the reference value.

Integral Wind-Up

- When a large change in reference input occurs (say a positive change) and the integral terms accumulates a significant error during the rise (*windup*), thus overshooting and continuing to increase as this accumulated error is unwound (offset by errors in the other direction);
- Integral windup particularly occurs as a limitation of physical systems, compared with ideal systems, due to the ideal output being physically impossible (e.g. *actuator saturation*);
- There are several solutions to this problem:
  - Increasing the reference input in a suitable ramp to prevent actuator saturation;
  - Disabling the integral function until the to-be-controlled process variable has entered the controllable region;
  - Preventing the integral term from accumulating above or below pre-determined bounds.

PID Controller and Tuning

- A PID controller has all three sets of gains:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$

- With a PID controller, you may achieve good stability, fast response, and low steady-state error if you can *tune* the gains properly;
- Tuning the PID controller is a difficult process although there are only three tuning knobs;
- Many requirements, such as short transient and high stability, are *conflicting* and hard to achieve at the same time;
- There are several methods for PID tuning, which includes manual and computerized methods; we will talk about a manual tuning approach in the next couple slides;
- A PID controller is a linear controller, but its application is not limited to linear systems.

Effect of Tuning

- Tuning of K_p , K_i , and K_d will each have different effects on the control system performance;
- For example, if you increase K_p, the steady state error and rise time will be reduces, but the system stability is also degraded;
- The effect of increasing each of the three parameters independently is summarized in the table below:

Parameter	**Rise Time**	**Overshoot**	**Settling Time**	**Steady State Error**	**Stability**
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor Change	Decrease	Decrease	No effect	Improve if K_d is small

Manual Tuning

- One tuning method is to first set K_i and K_d values to zero. Increase the K_p until the output of the loop oscillates, then the K_p should be set to approximately half of that value for a "quarter amplitude decay" type response;
- Then increase K_i until any steady state error is corrected in sufficient time for the process. However, too much K_i will cause instability;
- Finally, increase K_d, if required, until the loop is acceptably quick to reach its reference after a load disturbance. However, too much K_d will cause excessive response and overshoot;
- A fast PID loop tuning usually overshoots slightly to reach the reference input more quickly; however, some systems cannot accept overshoot, in which case an over-damped closed-loop system is required, which will require a K_p setting significantly less than half that of the K_p setting that was causing oscillation.

Robot Control Example #1

Moving to a Point with a Steering Type Robot:

- Consider the problem of moving toward a goal point (x^*, y^*) in the plane. We will control the robot's velocity to be proportional to its distance from the goal: $v^* = K_v \sqrt{(x^* - x)^2 + (y^* - y)^2}$

  , and to steer toward the goal which is at the vehicle-relative angle in the world frame of $\theta^* = \arctan \frac{y^* - y}{x^* - x}$

  using a proportional controller: $\gamma = K_h(\theta^* - \theta)$
- Keep in mind that we use the angular difference here. i.e. 15° - 330°= 45 ° for the feedback;
- This example and the following ones are in our textbook and the MATLAB Robotics Toolbox by Peter Corke.
- Check: sl_drivepoint.mdl

Robot Control Example #2

Following a Line Defined by $ax + by + c = 0$**:**

- This requires two controllers to adjust steering. One controller steers the robot to minimize the robot's normal distance from the line with a proportional controller to turn the robot toward the line:

$$d = \frac{ax + by + c}{\sqrt{a^2 + b^2}} \qquad \alpha_d = -K_d d$$

- The second controller adjusts the heading angle, or orientation, of the vehicle to be parallel to the line with a proportional controller:

$$\theta^* = \arctan \frac{-a}{b}, \quad \alpha_h = K_h(\theta^* - \theta)$$

- The combined control law turns the steering wheel so as to drive the robot toward the line and move along it: $\gamma = -K_d d + K_h(\theta^* - \theta)$

- Check: sl_driveline.mdl

Robot Control Example #3

Following a Pre-Specified Path:

- The path might come from a sequence of coordinates generated by a motion planner, or in real-time based on the robot's sensors;
- A simple and effective algorithm for path following is pure *pursuit* in which the goal $(x^*(t), y^*(t))$ moves along the path at constant speed, and the vehicle always heads toward the goal (carrot and donkey!);
- This problem is very similar to Example #1 (moving to a point), except the point is now moving. The robot maintains a distance d^* behind the pursuit point and the error is: $e = \sqrt{(x^* - x)^2 + (y^* - y)^2} - d^*$
- This error can be regulated to zero by controlling the robot's velocity using a PI controller: $v^* = K_v e + K_i \int e dt$
- A second proportional controller steers the robot toward the target:

$$\theta^* = \arctan \frac{y^* - y}{x^* - x} \qquad \gamma = K_h(\theta^* - \theta)$$

- Check: sl_pursuit.mdl

Summary

- It’s important for a mobile robot to control itself properly...
- A control system is a system, not just the controller itself;
- Feed-forward and feedback links each play different roles in a control system;
- A PID controller uses information from the past, present, and future to decide its control action;
- Turning a PID controller can be tricky due to multiple conflicting requirements;
- Many types of robot actions can be achieved with simple controller designs.

Further Reading

- Our Textbook!
- PID Control with MATLAB and Simulink: http://www.mathworks.com/discovery/pid-control.html
- Anti-Windup Control Using a PID Controller http://www.mathworks.com/help/simulink/examples/anti-windup-control-using-a-pid-controller.html
- Search keywords ‘PID controller’ and ‘integral wind-up’ in Wikipedia.