

CSE397/497-013
Introduction to Mobile Robotics

Feedback Control
18 Oct 2005



LEHIGH
UNIVERSITY

Department of
Computer Science & Engineering

P.C. Rossin
College of Engineering
and Applied Science

CSE397/497 Intro to Mobile Robotics

Today

- Homework 3 questions/comments?
- Project status updates
 - *U12*
 - *GPS*
 - *Compass*
- Feedback Control
 - *Motivation*
 - *Open Loop vs. Closed Loop*
 - *Basic Feedback Controllers*
- References
 - *Matlab Control Tutorial*
<http://www.engin.umich.edu/group/ctm/PID/PID.html>

© JR Spletzer

Motivating Example

- Yaoyao and Douglas have just finished coding up the interface to the U12 DAQ
- You are in charge regulating the linear velocity of the robot using taking inputs from the encoder and sending 0-255 outputs to the speed controller
- How would you design a controller to do this?
- Obvious Solution: Drive the car at each of the different speed controller values and store the results in a look up table
- Obviously WRONG Solution:
 - *What if the incline of the ground is not the same level?*
 - *What if the rolling friction on the ground is different?*
 - *What if the battery voltage was different from when you calibrated?*
 - *What if...*

© JR Spletzer

The Role of Control

- We have many ways of measuring the state (position, orientation, wheel velocity, etc.) of our robot
- In practice many tasks require us to alter the current state via a control output in order to achieve an objective or desired state:
 - *Cruise control: Maintain a “constant” velocity using input from our encoders and outputting a signal to the speed controller*
 - *Wall Follower: Measure our orientation to the wall using input from our IR sensors and output a change to our steering angle*
 - *Target Tracking: Pan/Tilt the camera in order to maintain the target within the camera’s image*
- Controllers provide a mechanism for achieving the robot state

© JR Spletzer

What is a Controller?

- In this context, a controller is a function that maps differences between the state x and objective state x_d values to outputs that we can regulate on our robot

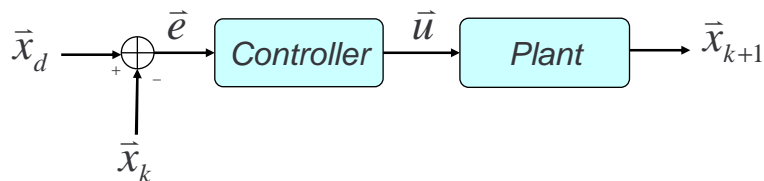
$$\bar{u} = f(\bar{x} - \bar{x}_d)$$

- Inputs are typically the state error (or the state itself)
 - We observe these from one of the 3 weeks of perception paradigms we examined
- Outputs are control signals to some actuator
 - Voltage signal to a speed controller
 - Steering signal to a servo motor

© JR Spletzer

Open Loop Control (1)

- Sometimes referred to as *ballistic movements*
- Error is measured, and a single output is calculated to bring the error to zero – no additional feedback!



- Open loop control can still be used *if* you have accurate sensing to and a well calibrated output

© JR Spletzer

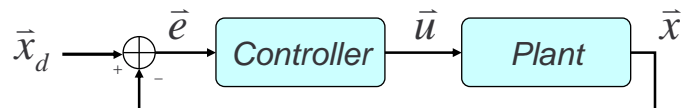
Open Loop Control (2)

- Advantages:
 - Cheap
 - Fast
 - Can be effective if well calibrated
 - Can be used for in learning a controller
 - Used by biological systems
- Disadvantages:
 - Ignores additional sensing information
 - Cannot accommodate for disturbances, changes in the environment, etc.

© JR Spletzer

Feedback Control (1)

- Aka “Closed-loop” Control
- The Idea: Why not use observations from one of those sensors to provide feedback on the current state value and adjust the input accordingly



- The Challenge: Feedback must be done in “real-time”
 - If your feedback frequency is f , you are running an open loop controller for $1/f$ seconds between updates
 - The frequency of f is a function of the application. As fast as your actuator can respond is ideal, but you can reach a point of diminishing returns

© JR Spletzer

Feedback Control (2)

- The objective of the controller is to drive the system error to zero at all times
- Feedback controllers rely both on the *magnitude* and *direction* of the error
- The higher the rate of feedback the better the controller performance
- We will examine the three primary linear controller designs:
 - *P* (*proportional*)
 - *PD* (*proportional-derivative*)
 - *PID* (*proportional-integral-derivative*)

© JR Spletzer

Proportional (P) Controller

- Output is *proportional* to the input (error)

$$u = k_p (x_d - x)$$

- Input is amplified by some proportional value k_p referred to as the controller gain to obtain the corresponding output signal

© JR Spletzer

Robot Velocity P Controller (1)

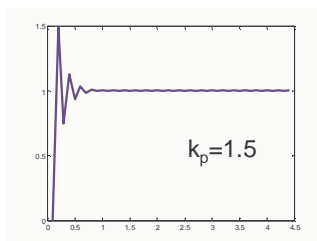
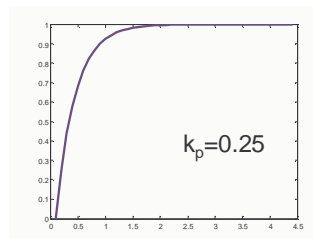
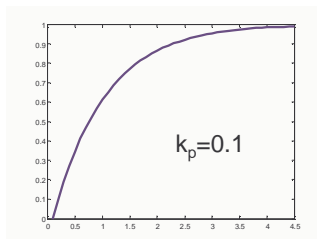
- Let's say we wish to regulate the linear velocity of our differential drive robot
- We will assume that $\omega_d = 0 \Rightarrow \dot{\phi}_L = \dot{\phi}_R = \dot{\phi}$
- Our control law then becomes

$$u = k_p (\dot{x}_d - \dot{x}) \propto \dot{\phi}_d - \dot{\phi}$$

- Let's assume that our robot is starting from a stop and is to accelerate to 1 m/s, how does our P controller perform?

© JR Spletzer

Robot Velocity P Controller (2)



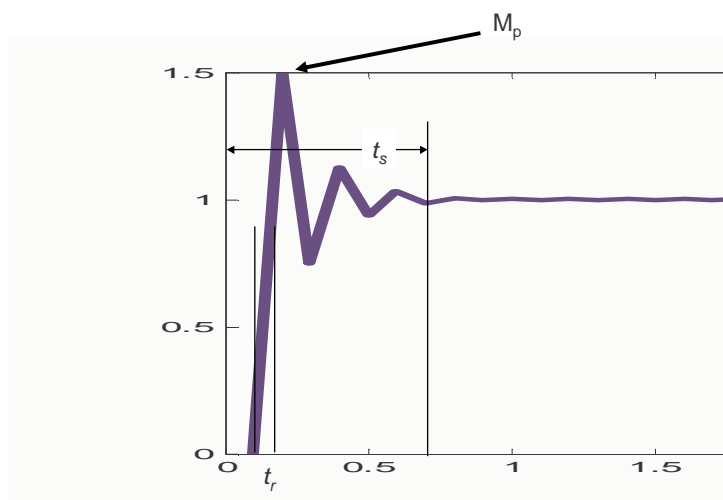
© JR Spletzer

Characterizing Controller Performance (1)

- Four main parameters often used as metrics:
 - Rise time (t_r)
 - ◆ Time to rise from 0.1 to 0.9 desired value
 - Overshoot (M_p)
 - ◆ Largest magnitude in excess of desired value
 - Settling time (t_s)
 - ◆ Time when $|e| \leq 0.01x_d$
 - Steady-state error
 - ◆ Error remaining after the controller input is no longer affecting plant output

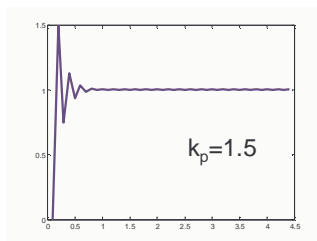
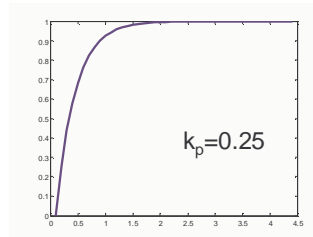
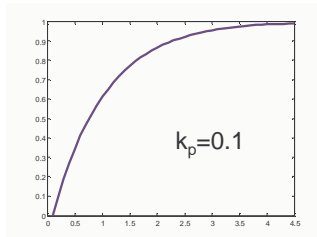
© JR Spletzer

Characterizing Controller Performance (2)



© JR Spletzer

Robot Velocity P Controller (3)



- Increases the controller gain decreases the rise time
- Excessive gain can result in overshoot

© JR Spletzer

Proportional-Derivative (PD) Controller

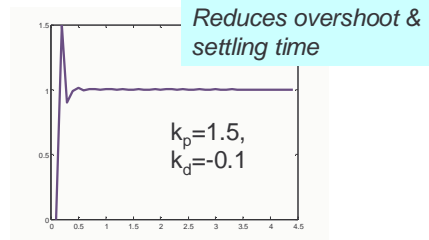
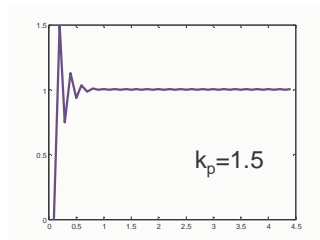
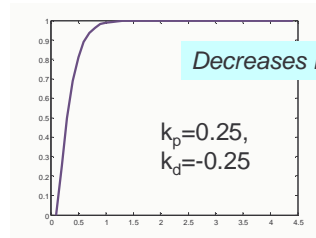
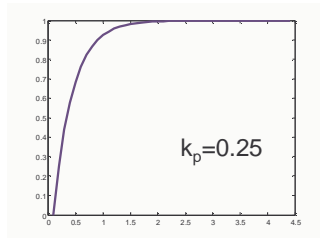
- Real-world systems often have momentum
- Do not want to have a high velocity when near the desired state value, as this can lead to overshoot
- PD controller adds a derivative term to the P controller that is proportional to the rate of change in the error ($x_d - \dot{x}$)

$$u = k_p e + k_d \dot{e}$$

- By regulating the input based on not just the error, but the rate of change of the error can help mitigate oscillations in the controller

© JR Spletzer

Robot Velocity PD Controller



© JR Spletzer

Proportional-Integral-Derivative (PID) Controller

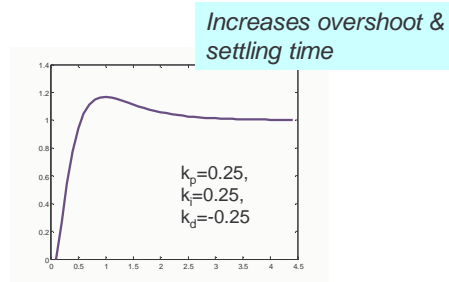
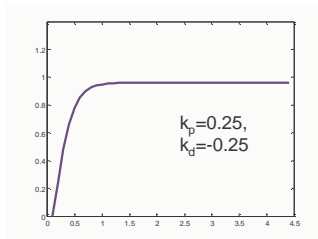
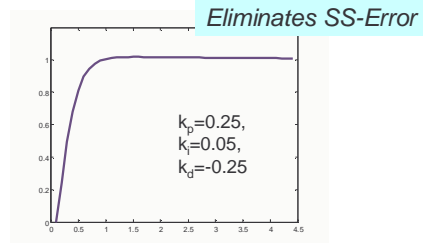
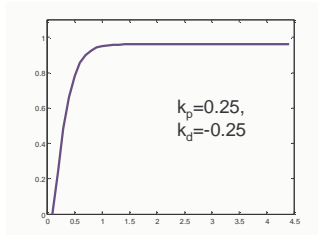
- The last stage added is an integral component
- The PID controller adds an integral term that can offset steady-state errors observed by the system

$$u = k_p e + k_i \int_0^t e dt + k_d \dot{e}$$

- The idea is that if there is a steady-state error, over time the integral term will become sufficiently large and begin to influence the output to compensate for these errors

© JR Spletzer

Robot Velocity PID Controller



© JR Spletzer

Controller Characteristics Summary

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
K_p	Decrease	Increase	Small Change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small Change	Decrease	Decrease	Small Change

- Note that when using multiple terms, correlations between the components can alter effects
- Treat this table as a guideline in choosing your gain values as a result

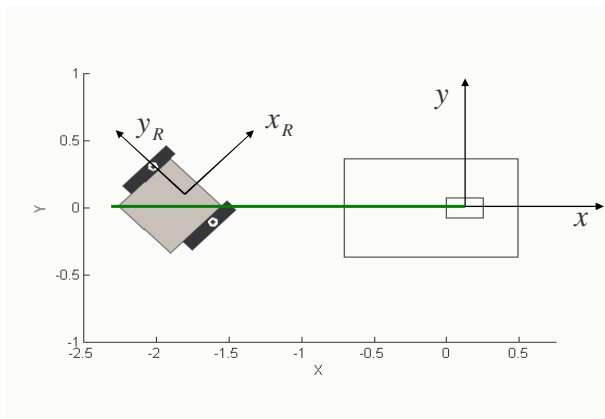
Tuning Gains

- Choosing appropriate gains is critical to optimizing controller performance
 - Sometimes this can be accomplished analytically
 - Other times require empirical modeling
- Even if you can obtain the optimal gain from analysis, you have only done this on a model of your system
 - You will still need to tune the gains empirically on the actual hardware to optimize performance
 - If your model is of sufficient fidelity, you will have a feasible region of gain values where your controller will perform well in practice

© JR Spletzer

Sample Application: ATRS Wheelchair Docking

- Consider a differential drive robot



Kinematic model:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v \cos \theta \\ v \sin \theta \\ \omega \end{bmatrix}$$

© JR Spletzer

Wheelchair Docking

- Assumptions:
 - Linear velocity is constant
 - Our output is angular velocity of the chair
- For error, we will use the distance from the y-axis

$$e = y_d - y$$

- From which we design a PD controller as:

$$u = k_d (\dot{y}_d - \dot{y}) + k_p (y_d - y)$$

- Since $y_d=0$ and is constant, this reduces to

$$u = -k_d \dot{y} - k_p y$$

Wheelchair Docking

- From our kinematic model we have

$$\dot{y} = v \sin \theta$$

$$\ddot{y} = -v \cos \theta \dot{\theta} + \dot{v} \sin \theta = -\omega v \cos \theta$$

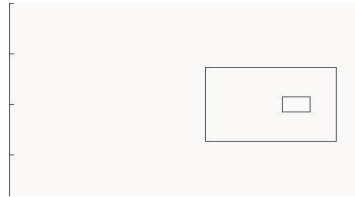
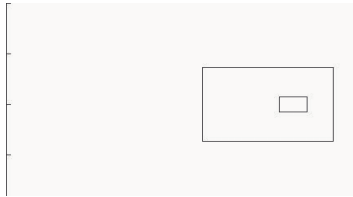
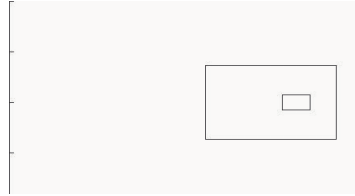
- By defining $u \equiv \ddot{y}$ we obtain

$$\omega = -k_d \tan \theta - \frac{k_p y}{v \cos \theta}$$

Experimental Results

- This is a PD controller designed using input-output feedback linearization.
- A critically damped behavior is obtained by setting

$$k_v = 2\sqrt{k_p}$$



© JR Spletzer

ATRS Experimental Results

ATRS Wheelchair Docking Experiments



Humberto Sermeno-Villalta & John Spletzer
Lehigh University (June 2005)



LEHIGH
UNIVERSITY

Department of
Computer Science & Engineering

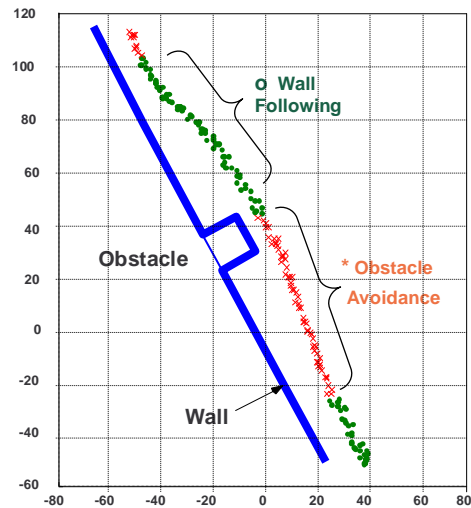
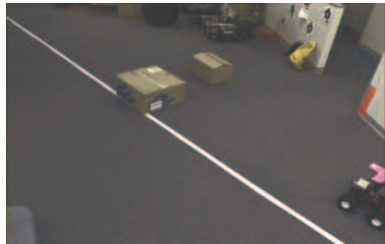
P.C. Rossin
College of Engineering
and Applied Science

© JR Spletzer

Experimental Results

- This is a PD controller designed using input-output feedback linearization.
- A critically damped behavior is obtained by setting

$$k_v = 2\sqrt{k_p}$$



© JR Spletzer

Summary

- General Rules for Design:
 - Use the simplest controller that does the job
 - Start with a proportional controller and add additional terms as necessary
 - Tune the gains based on theory and empirical results (this can be as much of an art as a science)
 - Less terms = less gains
 - ◆ Tuning a P controller is a 1D optimization problem
 - ◆ Tuning a PID controller is a 3D optimization problem
- PD/PID controllers are very common in industrial applications

© JR Spletzer