Lecture 4-2

Stay in Control:

Feedback Control

The Robotics Primer (Ch. 10)

Course Announcements

- Bring your laptop and robot everyday
- Monday's Quiz will be on **Feedback Control (Ch. 10)**
- Lab 3 memo and code due by midnight on Friday, 4/03/09
- Upload memo and code to Angel
- Tuesday's Quiz will be on **Vision Sensors (Ch. 9)**
- Tuesday's Lecture will be on **Control Architectures (Ch. 10)**
- Spring Break, 4/11/09-4/19/09

Quote of the Week

"A common mistake people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools."

D. Adams

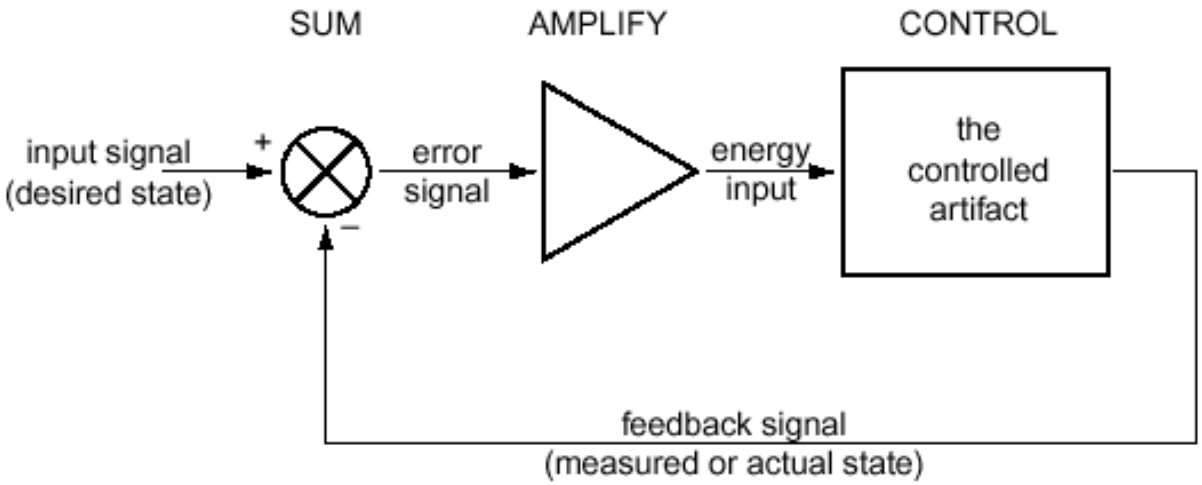
Feedback Control

Feedback Control

- Feedback Control is a means of getting a robot to achieve and maintain a desired state.
- The desired state (goal state) is the set point and it is achieved by continuously comparing its current state with its desired state.
- The difference between current and desired state is the error
- The goal of any control system is to minimize the error

Feedback Control

Feedback (closed loop) control is when a system achieves and maintains a *desired state* by continuously comparing its current and desired states, then adjusting the current state to minimize the difference



Feedback Control: Error and Overshoot

- *Error* is the difference in the current state and desired state of the system
  - The controller has to minimize the error at all times
- *Zero/non-zero error:*
  - Tells whether there is an error or not
  - The least information we could have
- *Direction of error:*
  - Which way to go to minimize the error
- *Magnitude of error:*
  - The distance to the goal state
- *Overshoot*
  - Is when the robot goes beyond its set point (position) or changes direction before stabilizing on it
- Control is much easier if we know both magnitude and direction

Feedback Control: Gains and Oscillations

- Incorrect controller gains will cause the system to undershoot or overshoot the desired state and cause oscillations
- Gain values determine if the system will:
  - keep oscillating
  - increase in oscillations
  - stabilize
- Damping - process of systematically decreasing oscillations
- A system is **properly damped** if it does not oscillate out of control or has no oscillations at all
- The robot *oscillates* around the set point. In general, the behavior of a feedback system oscillates around the desired set point (position)

Feedback Control: Wall Following

- Use feedback to design a wall following robot
- Infrared and/or sonar will provide distance to the wall (analog)
- The set point will be distance to the wall
- The error will be the difference between the set point and current distance to wall
- Control algorithm:
  - If distance-to-wall is correct, then keep moving
  - else if distance-to-wall is too large, then turn toward the wall
  - else if distance-to-wall is too small, then turn away from the wall
- With this algorithm, the robot will keep moving and wiggling back and forth as it moves. The amount of wiggling is determined by how often the error is computed and how much correction (turn) is made each time.

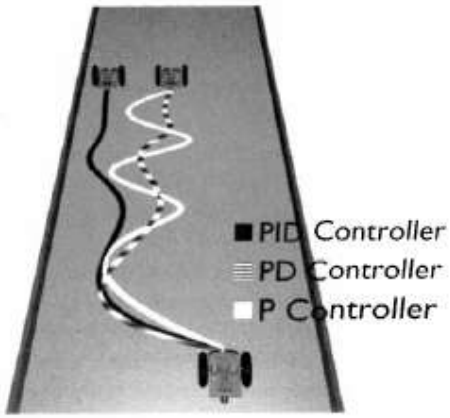
How to decrease oscillations?

- Compute the error often so the robot turns more often
- Adjust the turn angle so that the robot turns by smaller angles
- Define the right range of distances to define the robot's goal
- Determining these values depends on specific robot parameters
  - Robot's speed of movement
  - Range of the sensors
  - Sampling rate of the sensor

Types of Feedback Control

There are 3 types of

basic feedback controllers



- P: proportional control
- PD: proportional derivative control
- PID: proportional integral derivative control

Feedback Control: Proportional Control

- The response of the system is proportional to the amount of the error
- The output of the controller is proportional to the input i (speed or velocity error):

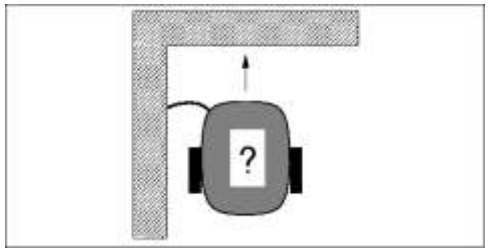
$$\text{output} = K_p * \text{error input}$$

- K_p is a proportionality constant (gain)
- The controller generates a stronger response the farther away the system is from the goal state
- The robot will turn sharply toward the wall if it is really far from it
- The robot will turn gently toward the wall if it is slightly away from it
- As the proportional gain become larger it will introduce **overshoot** and **oscillations**

Proportional Control: Wall Following

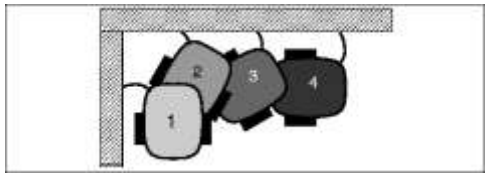
- The distance to wall parameter would be used to determine the angle and distance and/or speed that the robot moves
- The larger the error, the larger the movement
- The smaller the error, the smaller the movement
- The parameters that determine the magnitude of the system's response are called **gains**
- Determining the right gain requires trial and error, testing and calibrating the system
- If the system is well understood, the gains can be computed mathematically

Proportional Control: Wall Following



- Negotiating a corner
  - Make little turns, drive straight

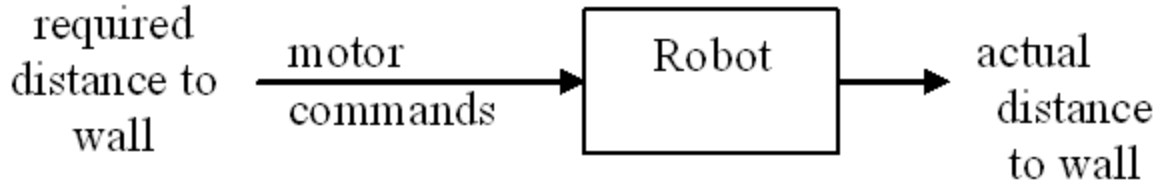
  ahead, detect the wall, back up, repeat
  - The disadvantage is that this method is time consuming and produces jerky movements

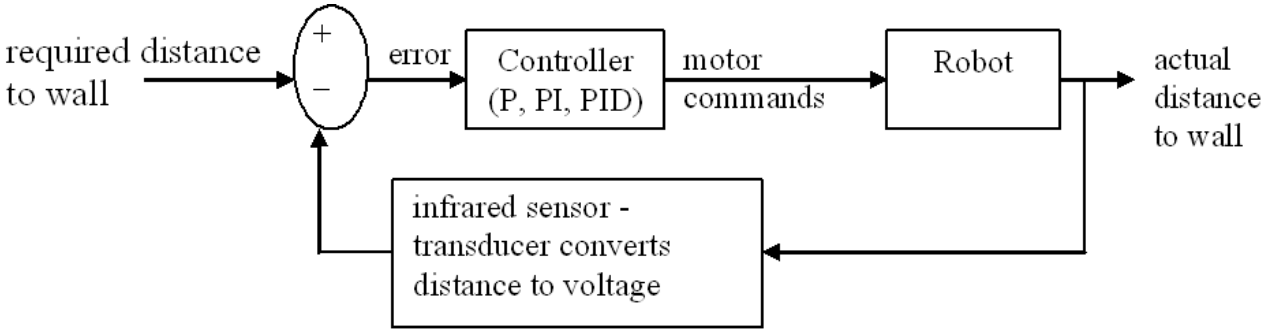


- Alternative (proportional control):
  - Execute a turn command that was timed to accomplish a ninety degree rotation
  - Works reliably only when the robot is very predictable (battery strength, traction on the surface, and friction)

http://www.cse.unr.edu/~monica/Courses/CPE470-670/cpe470-670-sp05.html

Wall Following:
Open Loop versus Closed Loop Control





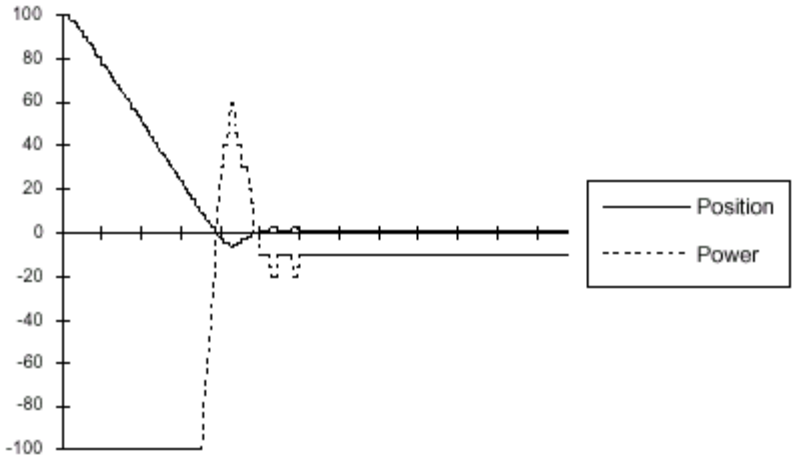
Proportional Control: Robot Position

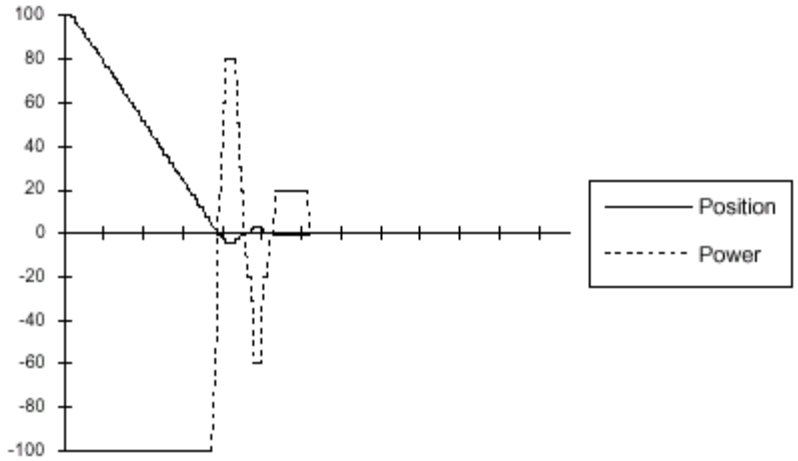
- What if the robot is 100 cm from home and your goal is to move the robot forward 100 cm
- The set point is position = 0 cm
- The error is (set point – position = 0 - position)
- A simple Proportional controller has:
  - Motor Power Command = set point – position = 0 - position
- Initially, the error is -100
  - The motor turns on full speed
  - As it starts going, the error becomes progressively smaller
- Halfway, at position -50, the error is only -50
  - at that point the motor goes at 50% of full power
- When it arrives at the intended position of 100, the error is zero
  - the motor is off

Proportional Control: Robot Position Control

power= K_p * (0 - position)

- K_p=10:
  - System overshoots the zero point, and had to turn around
  - Offset Error: System did not stabilize at the goal
  - Power command too small to activate the motor
  - The response is critically damped
- K_p=20:
  - Offset error is solved
  - Oscillation: the system overshoots the goal
  - The response is slightly underdamped



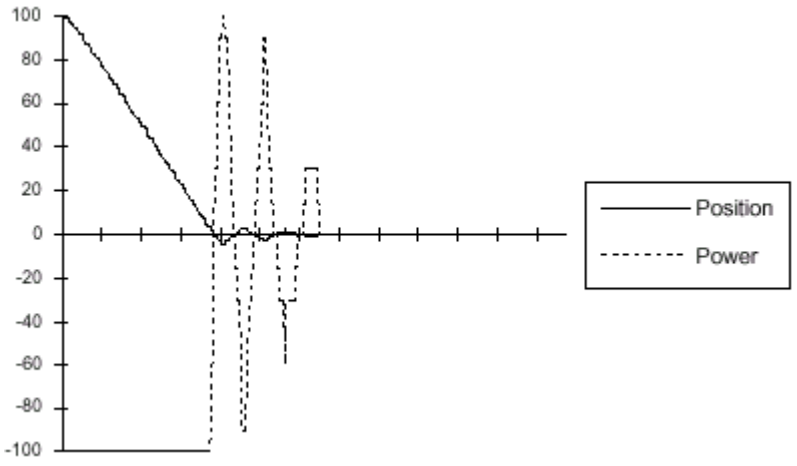


http://www.cse.unr.edu/~monica/Courses/CPE470-670/cpe470-670-sp05.html

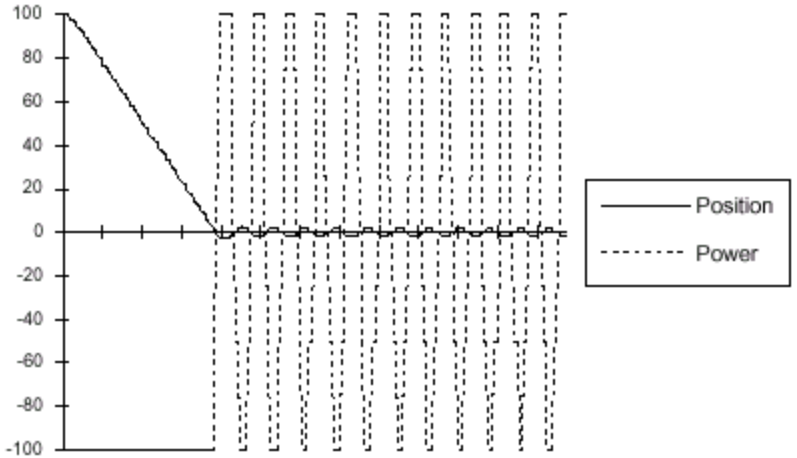
Proportional Control: Robot Position Control

power= K_p * (0 - position)

- K_p=30:
  - Oscillation problem is more pronounced;
  - The response is *underdamped*



- K_p=50:
  - Oscillation behavior has taken over
  - System cannot stabilize at the set point
  - A small error generates a power command that moves the system across the set point
  - The system is *undamped*



http://www.cse.unr.edu/~monica/Courses/CPE470-670/cpe470-670-sp05.html

Feedback Control: Controlling Velocity

- Setting gains is difficult, and simply increasing the *proportional gain* does not remove oscillations
- The system needs to be controlled differently when it is close to the desired state than when it is far from it
- In proportional control, the momentum of the correction carries the system beyond the desired state, and causes oscillations

$$\text{Momentum} = \text{mass} * \text{velocity}$$

Feedback Control: Derivative Control

- Solution: correct the momentum as the system approaches the desired state
  - Subtract an amount proportional to the velocity
- As the system nears the desired state, we subtract an amount proportional to the velocity:

$$-(\text{gain} * \text{velocity})$$

- *Derivative term* (velocity is the derivative of position)
- A controller that has a derivative term is called a *derivative (D) controller*

Feedback Control: Derivative Control

- A derivative controller has an output proportional to the derivative of its error = r input i:

$$\text{output} = K_d * d(\text{error input})/dt$$

- K_d is a proportionality constant
- The intuition behind derivative control:
  - Controller corrects for the momentum as it approaches the desired state
- Wall Following: Slow down the robot and decrease the turning angle while getting closer to the desired state
- Position Control: Decrease the motor power while getting closer to the desired state

Feedback Control: Proportional-Derivative (PD) Control

- Proportional-derivative control is a
  - Combination (sum) of proportional and derivative terms

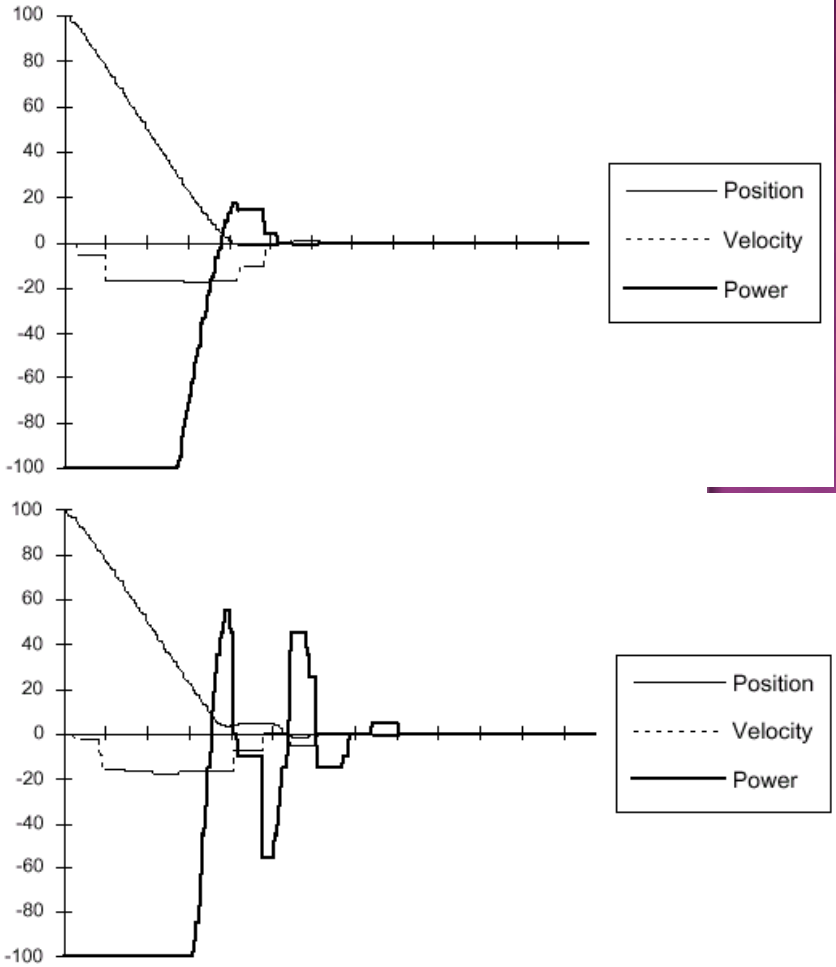
$$\text{output} = K_p * (\text{error input}) + K_d * d(\text{error input})/dt$$

- **PD Control** is used extensively in industrial process control
  - Combination of varying the power input when the system is far away from the set point, and correcting for the momentum of the system as it approaches the set point is quite effective
- **PD Control** improves the transient response of the robot to slow the controller momentum, typically with negative feedback
- **PD Control** is sometimes referred to as **rate control** because it is based upon the rate of change of the error
- In a wall following algorithm this would reduce the sharpness of the turn when the rate of change reaches some threshold

Feedback Control: Proportional-Derivative Control

- $K_p = 4$, $K_d = 1$:
  - Overshoot is minimized, no oscillatory behavior at all
- $K_p = 10$, $K_d = 5$:
  - Unstable; K_d is too large
  - Position graph: controller "puts on the brakes" too hard and the system stops moving before the destination set point
  - When the velocity hits zero, the proportional gain kicks in again and the system corrects

power= K_p * (0 - position)+ K_d * velocity



http://www.cse.unr.edu/~monica/Courses/CPE470-670/cpe470-670-sp05.html

Feedback Control: Integral Control

- Errors accumulate as the robot is running a longer time
- The system keeps track of its own repeatable, fixed errors called steady state errors
- How can we solve the problem?
  - Sum up the errors and compensate for them when they become significantly large
- The control system can be improved by introducing an *integral term*

$$output = K_i * \int (error\ input)dt$$

- K_i = proportionality constant
- Intuition:
  - System keeps track of its repeatable, steady state errors
  - These errors are integrated (summed up) over time
  - When they reach a threshold, the system compensates for them

Feedback Control: Proportional-Integral (PI) Control

- Proportional-integral control is a
  - Combination (sum) of proportional and derivative terms

$$\text{output} = K_p * (\text{error input}) + K_i * \int (\text{error input}) dt$$

- **PI Control** improves the steady state response and would correct for steady state or offset error
- **PI Control** is sometimes also referred to as **reset control** because it resets the steady state error to 0
- In a wall following algorithm, after the sum of the distance error reaches some threshold then adjust the turn angle

Feedback Control:
Proportional integral derivative (PID) Control

- PID control is a sum of all three of the basic controls

output = K_p * (error input) + $K_i * \int (error\ input)dt$ + K_d * d(error input)/dt

- This controller improves the steady-state and transient response of the robot's wall following algorithm

Feedback Control: Determining Controller Gains

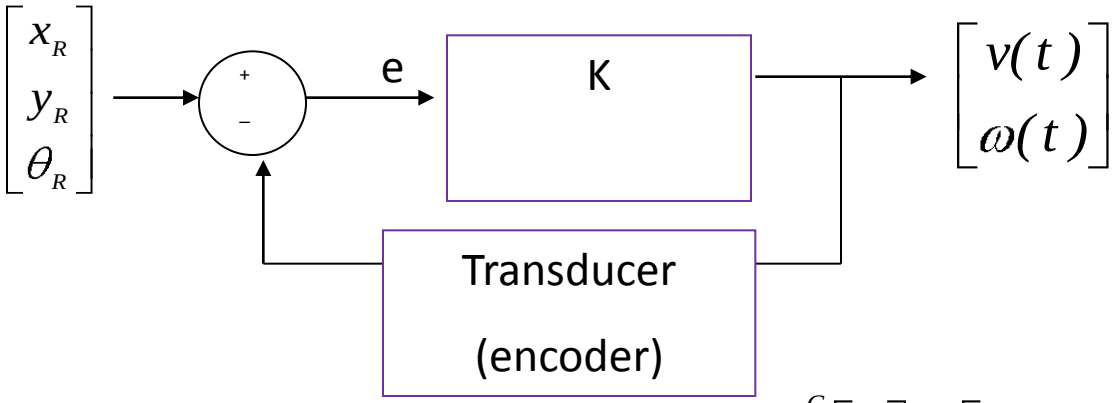
- How do we determine the controller gain?
- Empirically (trial and error):
  - requires that the system be tested extensively
- Analytically (mathematics):
  - require that the system be well understood and characterized mathematically (a model)
- Automatically
  - by trying different values at run-time

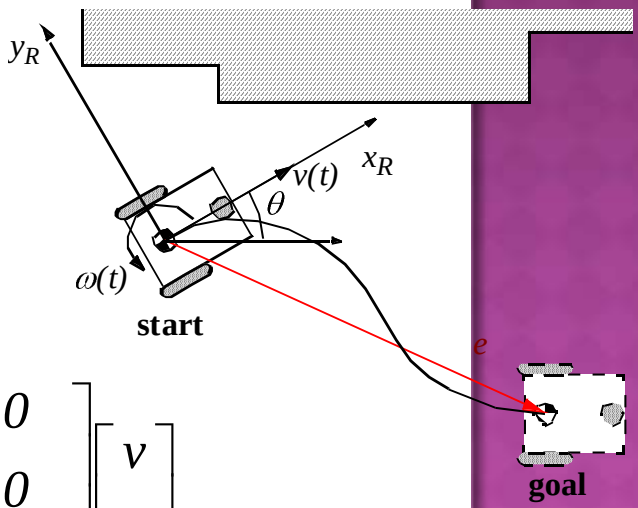
Feedback Control and Control Theory

- Feedback control is the lower level of control theory for controlling wheels or continuously moving actuators
- However, for achieving higher level goals
  - navigation
  - coordination
  - interaction
  - collaboration
  - human-robot interaction
- It is necessary to use techniques from the field of artificial intelligence

Feedback Control Example

Given a robot with an arbitrary position and orientation and a predefined goal position and orientation. Design a control matrix for a real-state feedback controller to drive the pose error to zero.





$$G\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix}=\begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix}$$

Motion Control: Feedback Control, Problem Statement

- The task of the controller is to find a control matrix ***K***, if it exists

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \end{bmatrix}$$

- Where the gains are a function of time and error,

$$k_{ij} = k(t,e)$$

- The controller output influences control signals, $v(t)$ and $\omega(t)$

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = K \cdot e$$

- drive the position error ***e(t)*** to zero: $\lim_{t \to \infty} e(t) = 0$

Kinematic Model (polar coordinates)

Robot starts facing the goal point

$$\rho = \sqrt{\Delta x^2 + \Delta y^2}$$

$$\alpha = -\theta + \operatorname{atan2}(\Delta y, \Delta x)$$

$$\beta = -\theta - \alpha$$

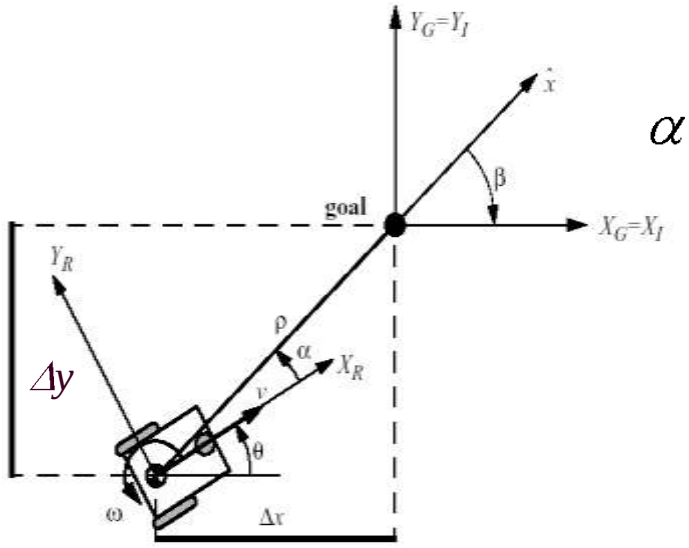
$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} \cos\alpha & 0 \\ -\frac{\sin\alpha}{\rho} & 1 \\ \frac{\sin\alpha}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$\alpha \in (-\pi/2, \pi/2]$$

Robot starts with back to the goal point

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos\alpha & 0 \\ \frac{\sin\alpha}{\rho} & -1 \\ -\frac{\sin\alpha}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$\alpha \in [\pi/2, \pi) \cup (-\pi, -\pi/2]$$



The Control Law

- The controls signals v and ω must be designed to drive the robot from (ρ_o, α_o, β_o) to the goal position
- Consider the control law, $v = k_\rho \rho$ and $\omega = k_\alpha \alpha + k_\beta \beta$
- The closed loop system description becomes,

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -k_\rho \rho \cos\alpha \\ k_\rho \sin\alpha - k_\alpha \alpha - k_\beta \beta \\ -k_\rho \sin\alpha \end{bmatrix}$$

Kinematic Position Control: Resulting Path