ECE 4960

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

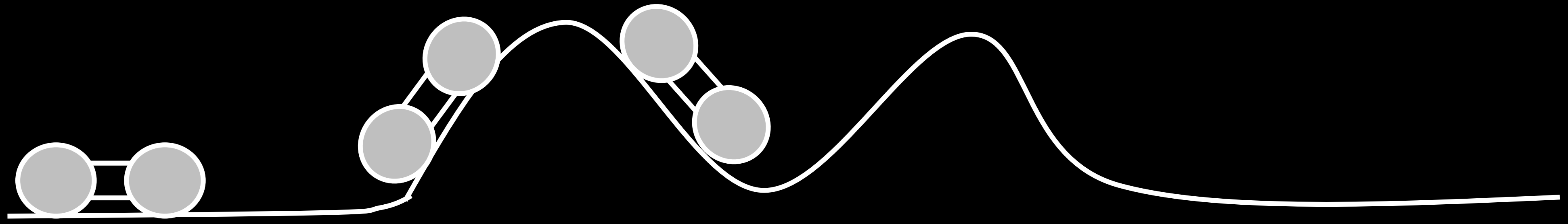
Fast Robots

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

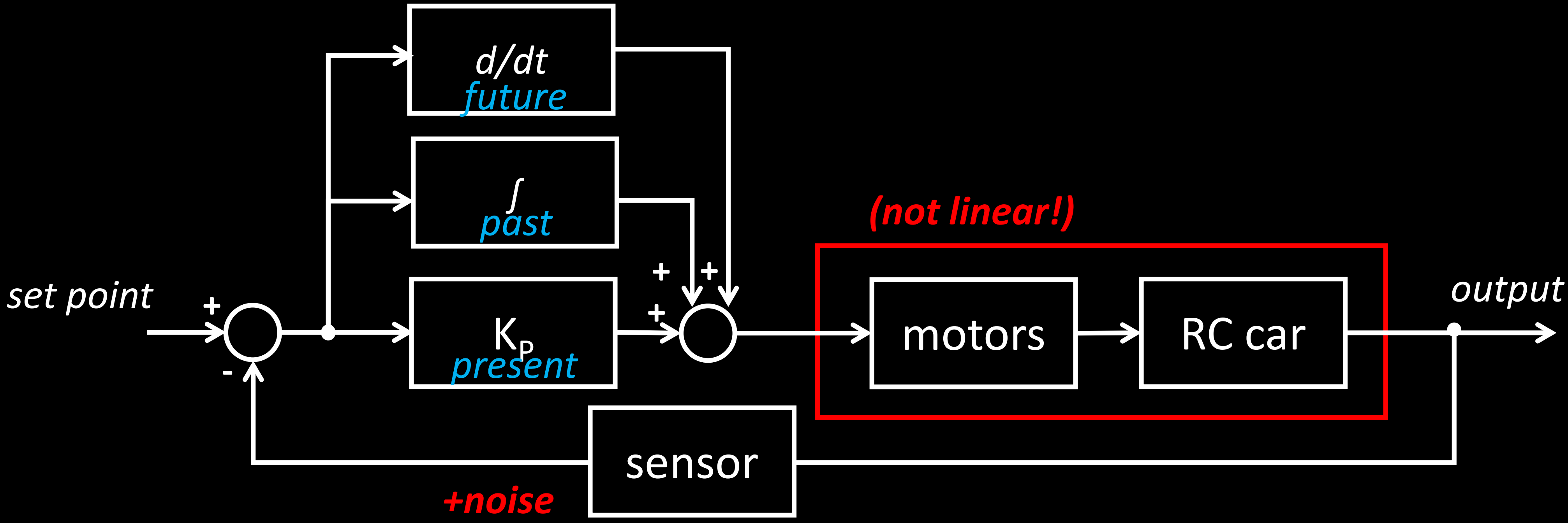
PID CONTROL (continued)

Feedback control

- Maintain speed prediction at different battery levels



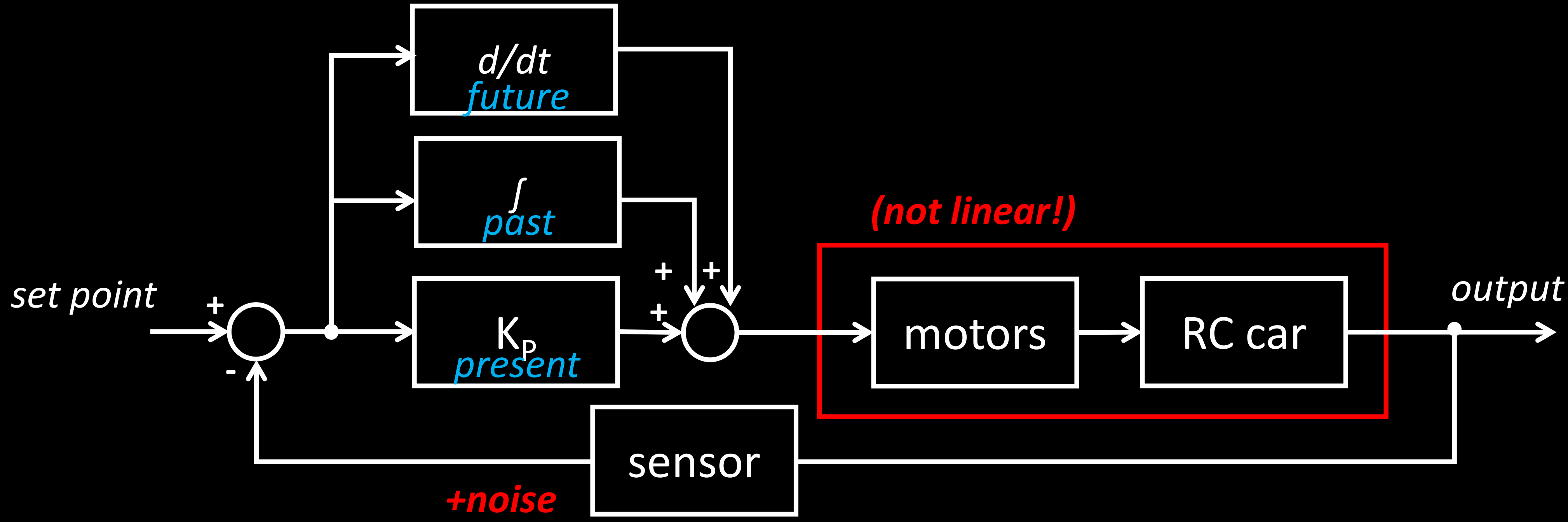
PID control



https://tinyurl.com/y67glgzk

- Integrator wind-up
- Derivative low pass filter
- Derivative kick
  - $\frac{de}{dt} = \frac{dsetpoint}{dt} - \frac{dmeasurement}{dt}$
  - When the setpoint is constant: $\frac{de}{dt} = -\frac{dmeasurement}{dt}$

PID control



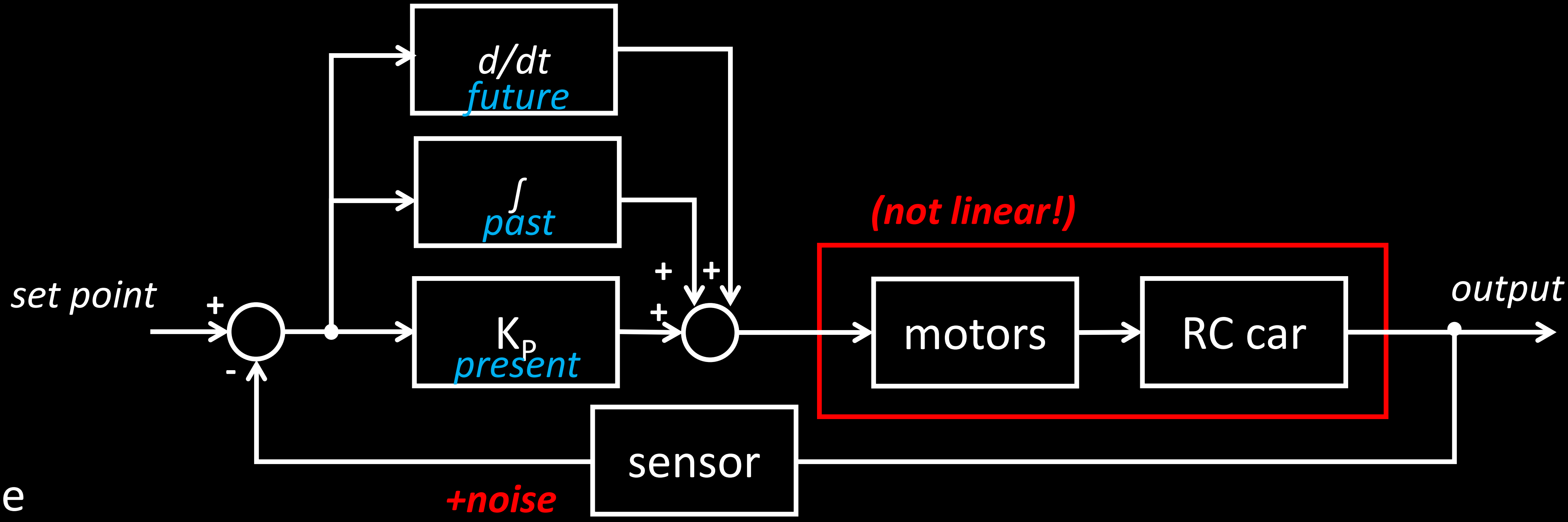
https://tinyurl.com/y67glgzk

- Integrator wind-up
- Derivative low pass filter
- Derivative kick
  - $\frac{de}{dt} = \frac{dsetpoint}{dt} - \frac{dmeasurement}{dt}$
  - When the setpoint is constant: $\frac{de}{dt} = -\frac{dmeasurement}{dt}$

1st order system: $\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \frac{-c}{I} \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{I} \end{bmatrix} u$

2nd order system: $\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ cst & \frac{-c}{I} \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{I} \end{bmatrix} u$

PID control



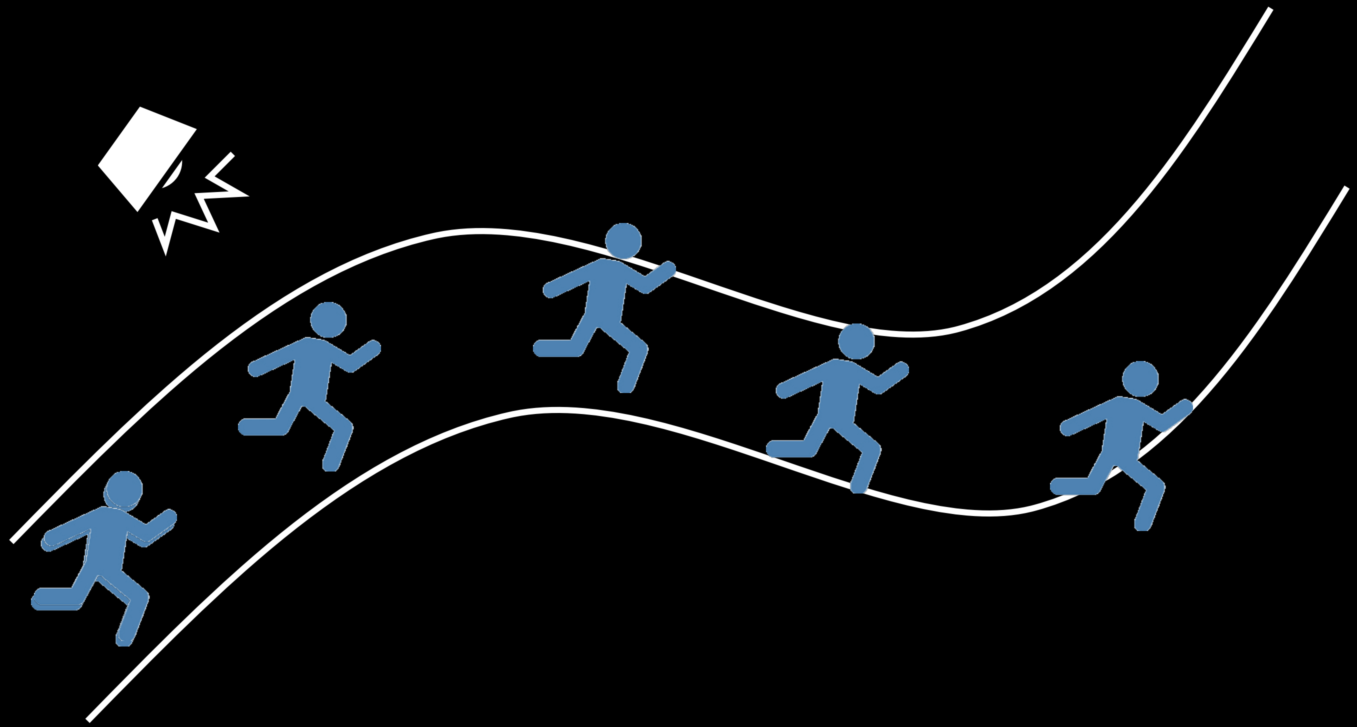
- Performance
  - Rise time/Response
    - Ex: 10% to 90% of final value
  - Peak time
    - Time to reach first peak
  - Overshoot
    - Amount in excess of final value
  - Settling time
    - Ex: Time before output settles to 1% of final value

PID control

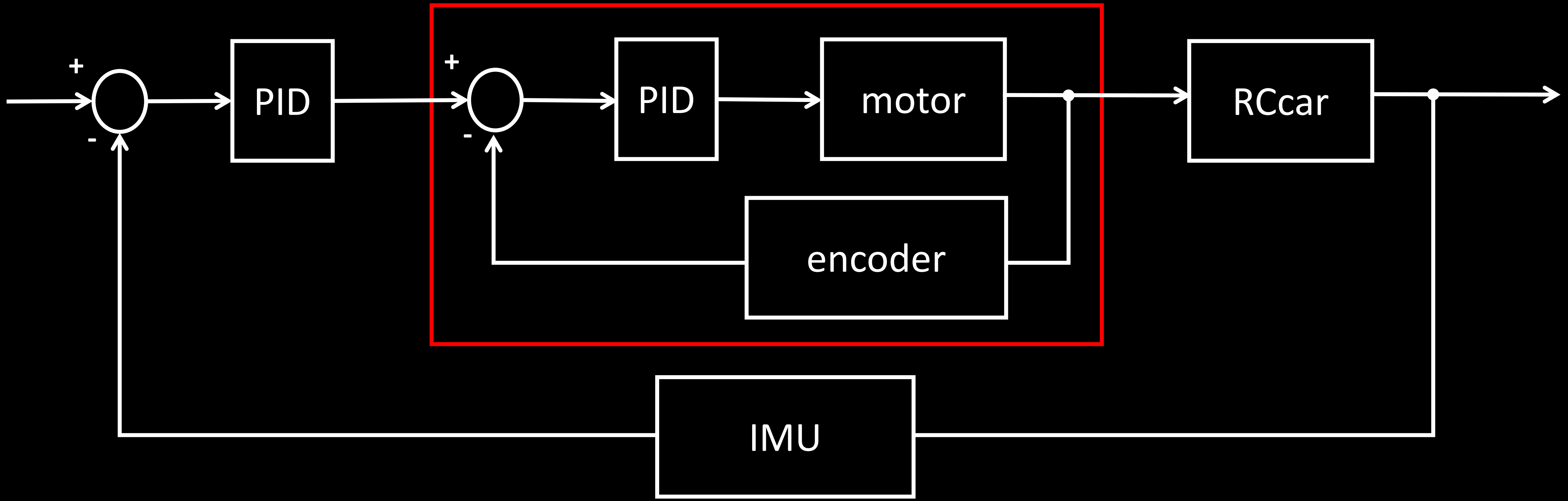
- **Heuristic procedure #1:**
  - Set Kp to small value, KD and KI to 0
  - Increase KD until oscillation, then decrease by factor of 2-4
  - Increase KP until oscillation or overshoot, decrease by factor of 2-4
  - Increase KI until oscillation or overshoot
  - Iterate
- **Heuristic procedure #2:**
  - Set KD and KI to 0
  - Increase KP until oscillation, then decrease by factor of 2-4
  - Increase KI until loss of stability, then back off
  - Increase KD to increase performance in response to disturbance
  - Iterate

Discrete PID Control

- Sampling time
- Control ~10 times faster than the system



Cascaded Control Loops



Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

LAB 3 PRESENTATIONS

Characterizing the RC Car

- Robert
- Charles/Emma
- Kathleen
- Tim
- Jungshien
- Katie/Jade/Emily
- Caitlin
- Christopher
- Daniel
- Andrew
- Jason

- Dimensions
- Wheel diameter
- Weight
- Battery life time
- Battery charging time
- Max speed
- Max acceleration
- On-axis turns
- Max inclination
- Surfaces
- Tricks

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

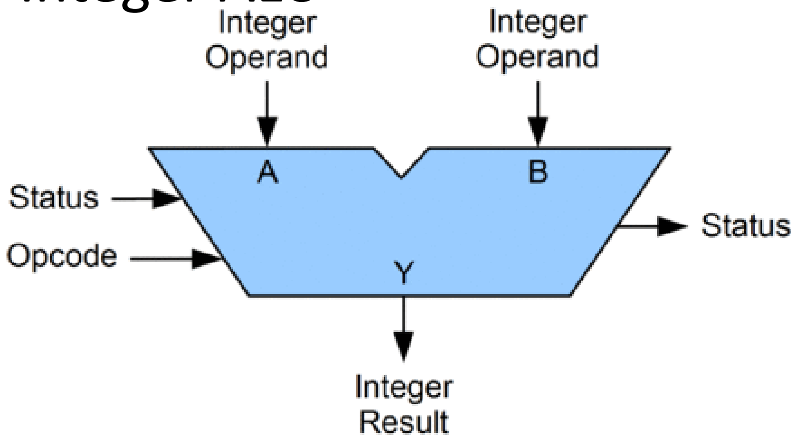
DATA TYPES

Data types

- Variable memory allocation depends on your processor *and* the compiler
  - Char
    - $Char_{8bit}$: 8 bits
    - $Char_{32bit}$: 8 bits
  - Int
    - Int_{8bit} : 16 bits
    - Int_{32bit} : 32 bits
  - Long
    - $Long_{32bit}$: 32bits
    - $Long_{64bit}$: 64 bits
- Two's complement
    - 0 b 0 0 0 0 0 1 0 1
    - 0 b 0 0 0 0 0 1 0 1 > invert > 0 b 1 1 1 1 1 0 1 0 > add 1 > 0 b 1 1 1 1 1 0 1 1
    - 0 b 1 1 1 1 1 1 1 1 ?
    - Range of a signed $char_{32bit}$: $[-2^7 ; 2^7 -1]$ = [-128 ; 127]
    - Range of a signed int_{32bit}: $[-2^{31} ; 2^{31}-1]$

You can specify the length:
- int16_t
- uint32_t

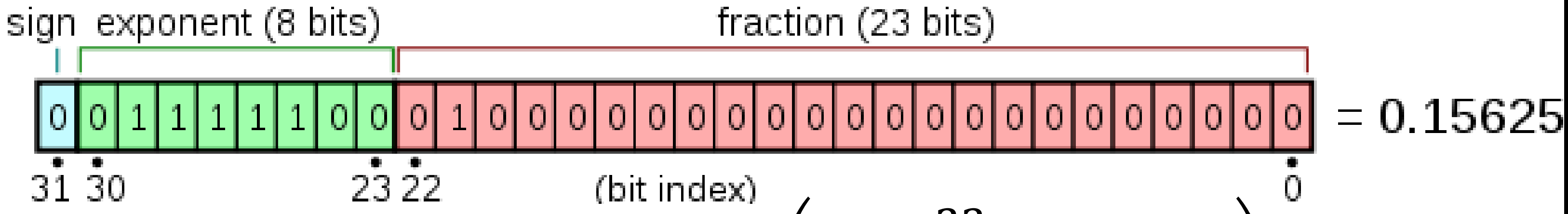
Integer ALU



Data types

- Variable memory allocation depends on your processor *and* the compiler
  - Float
    - Float_{8bit} : 32 bits
    - Float_{32bit} : 32 bits
  - Double
    - Double_{8bit} : 64 bits
    - Double_{32bit} : 64 bits
  - Long Double
    - 8, 12, 16 bytes

Float:



$$value = -1^{sign} \cdot 2^{e-127} \cdot \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i}\right)$$

- Single-precision floating point number (assuming IEEE 754 representation):
  - Max value : $(2 - 2^{-23}) \times 2^{127} \approx 3.4028235 \times 10^{38}$
  - Decimal string “3.141593” => float => decimal string “3.141593”
  - float pi = PI => decimal string “3.14159265” => float

Data types

https://cdn.sparkfun.com/assets/d/e/8/b/4/Apollo3_Blue_MCU_Data_Sheet_v0_12_1_rZ9Akgo.pdf

ambiq micro

Apollo3 Blue MCU Datasheet

Ultra-Low Power Apollo MCU Family

Features

Ultra-low supply current:
- 6 µA/MHz executing from FLASH or RAM at 3.3 V
- 1 µA deep sleep mode (BLE Off) with RTC at 3.3 V (BLE in SD)

High-performance ARM Cortex-M4 Processor
- 48 MHz nominal clock frequency, with 96 MHz high performance TurboSPOT™ Mode
- Floating point unit
- Memory protection unit
- Wake-up interrupt controller with 32 interrupts

Integrated Bluetooth[1] 5 low-energy module
- RF sensitivity: -93 dBm (typical)
- TX: 3 mA @ 0 dBm, RX: 3 mA
- Tx peak output power: 4.0 dBm (max)

Ultra-low power memory:
- Up to 1 MB of flash memory for code/data
- Up to 384 KB of low leakage RAM for code/data
- 16 kB 2-way Associative/Direct-Mapped Cache

Ultra-low power interface for on- and off-chip sensors:
- 14 bit ADC at up to 1.2 MS/s, 15 selectable input channels available

- 3.37 x 3.25 mm(<0.35mm thk pkg) 66-pin CSP with 37 GPIO
- 5 x 5 mm (<0.5mm thk pkg) 81-pin BGA with 50 GPIO

Applications

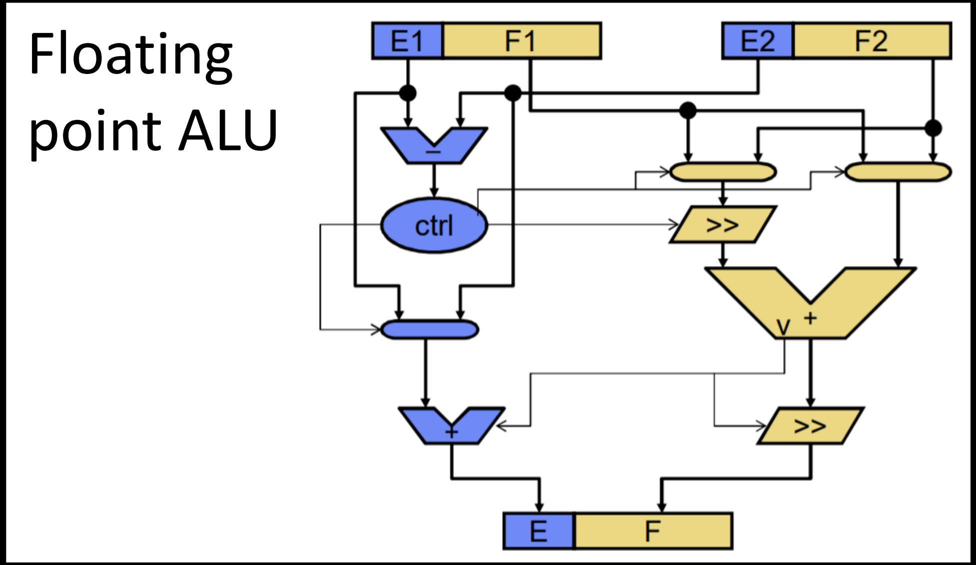
- Voice-on-SPOT™ compatible for always-listening keyword detect, audio command recognition and voice assistant integration in battery-powered devices including:
  - Bluetooth headsets, earbuds, and truly wireless earbuds
  - Remote and Gaming Controls
  - Smart home
- Wearables including smart watches and fitness/activity trackers
- Hearing aids, Digital Health Monitoring and Sensing Devices
- Smart Home Automation, Security and Lighting control applications

Description

The Apollo MCU Family is an ultra-low power, highly integrated microcontroller platform based on Ambiq Micro's patented Subthreshold Power Optimized Technology (SPOT™) and designed for battery-powered and portable, mobile devices. The Apollo3 Blue MCU sets a new standard in energy efficiency for battery-powered devices with an integrated ARM Cortex-M4 processor with Floating Point Unit and TurboSPOT™ increasing the compu-

Data types

- Variable storage depends on your processor *and* the compiler
  - Float
    - Float_{8bit} : 32 bits
    - Float_{32bit} : 32 bits
  - Double
    - Double_{8bit} : 64 bits
    - Double_{32bit} : 64 bits
  - Long Double
    - 8, 12, 16 bytes



- Single-precision floating point number (assuming IEEE 754 representation):
  - Max value : $(2 - 2^{-23}) \times 2^{127} \approx 3.4028235 \times 10^{38}$
  - Decimal string “3.14159” => float => decimal string “3.14159”
  - float pi = PI => decimal string “3.14159265” => float

Data types

- What do you have in your system?
  - Bluetooth: char
  - Time of flight: unsigned int
  - Serial.print: strings
  - IMU: float
  - PID: double
  - millis(): unsigned long
  - if-statements: bool
- *Pay attention!*
- https://www3.ntu.edu.sg/home/ehchua/programming/java/datarepresentation.html