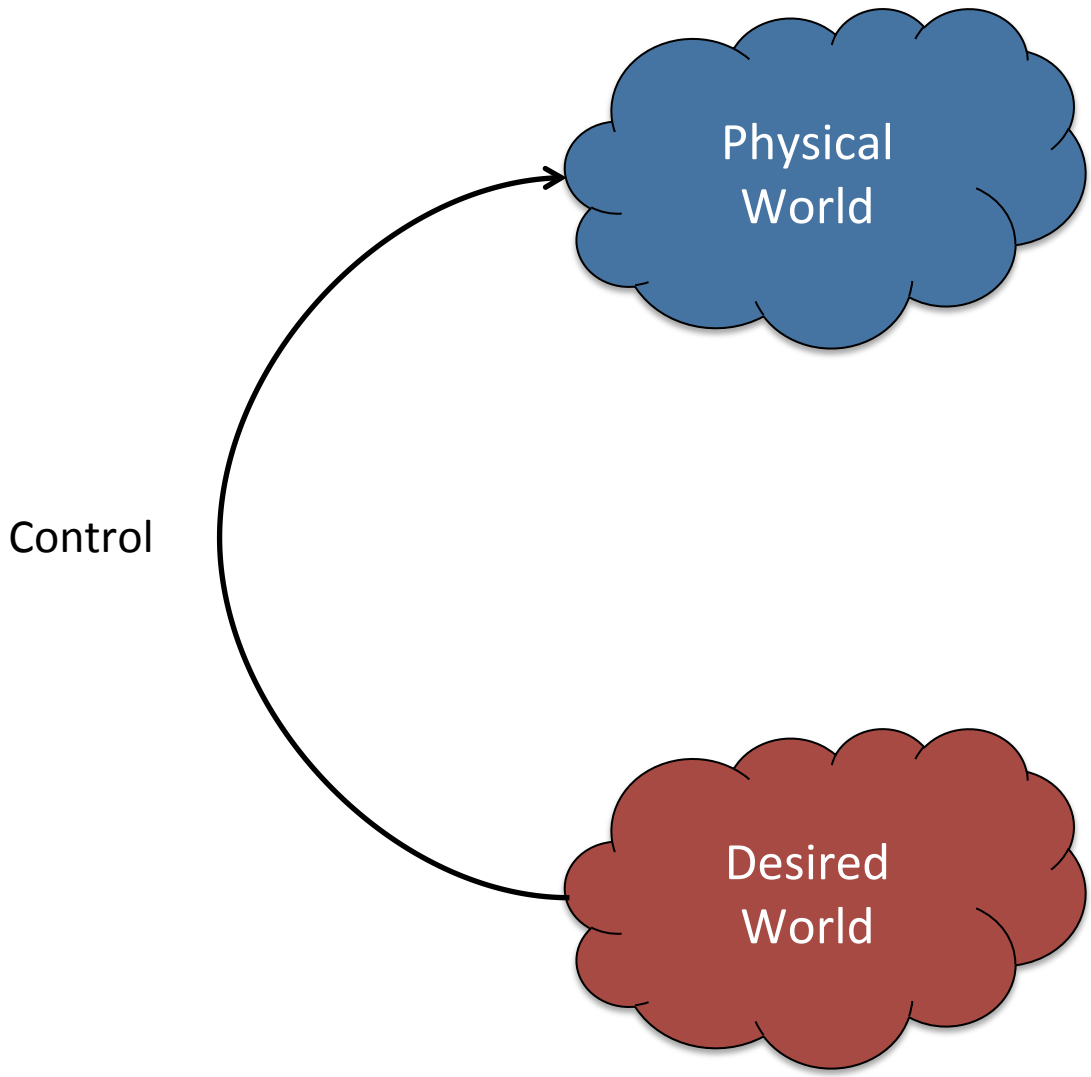
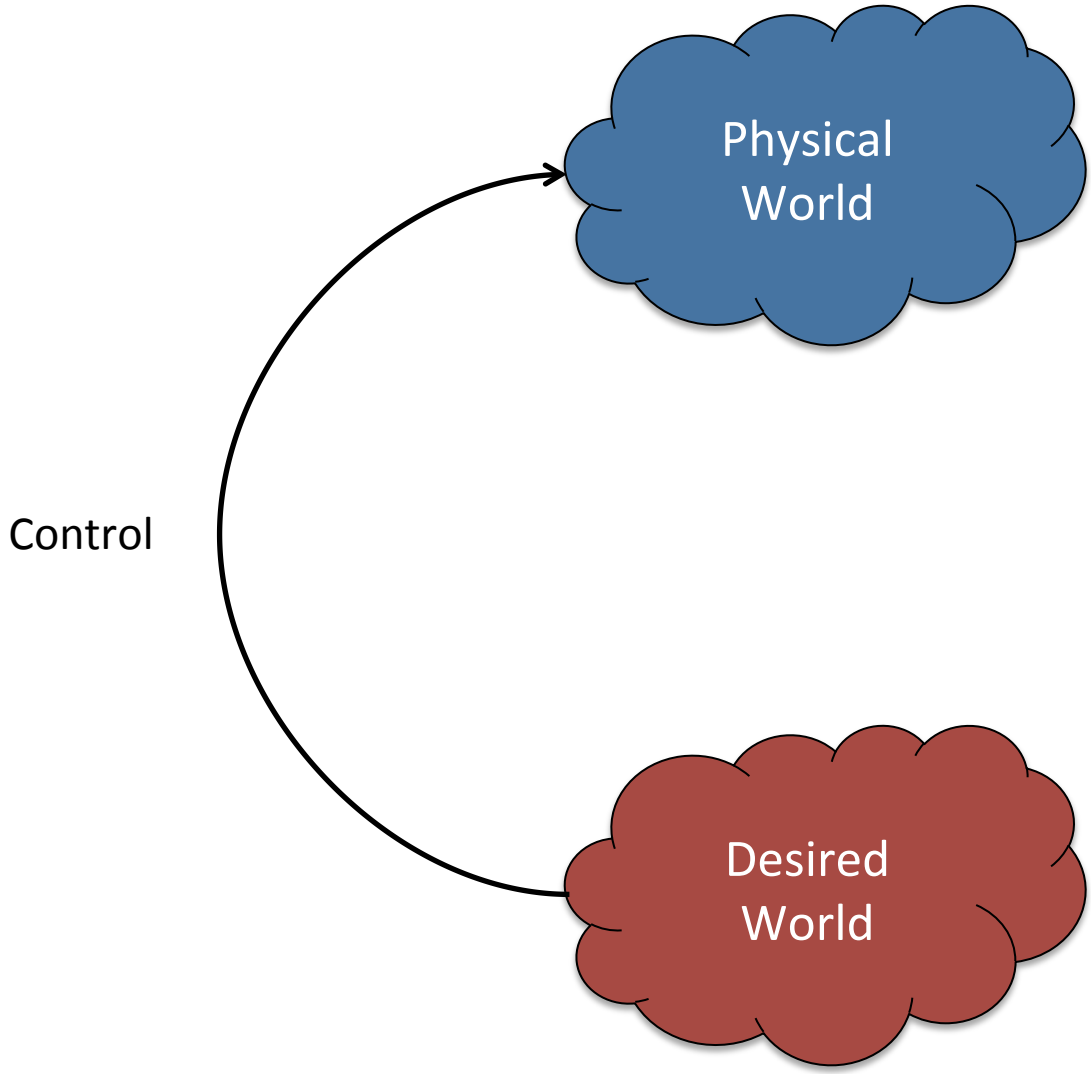
PID

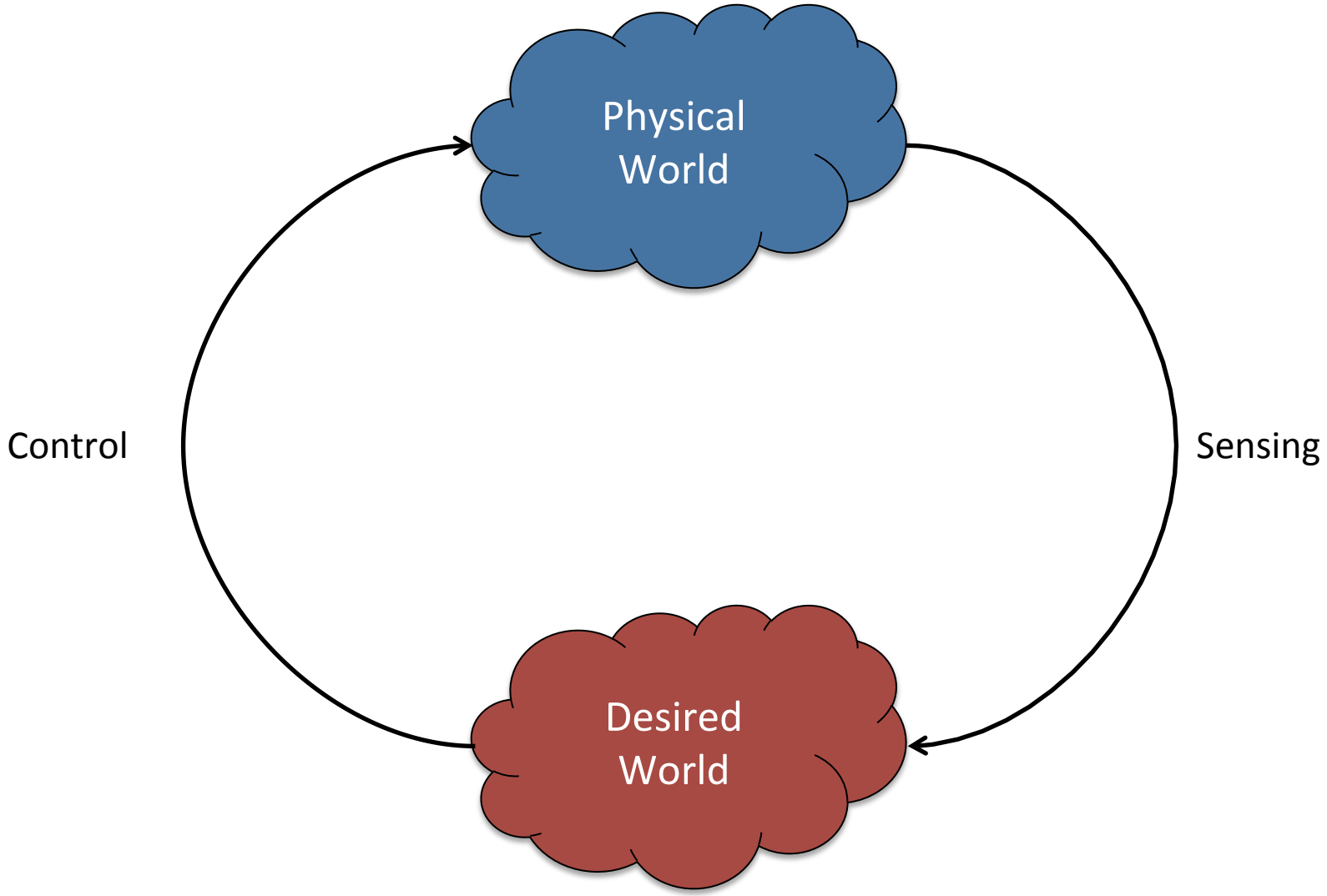
Ara Kourchians


Physical
World
Control
Desired
World

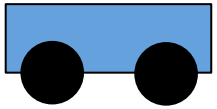
Open Loop

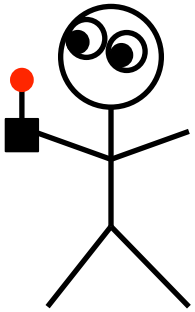


Closed Loop

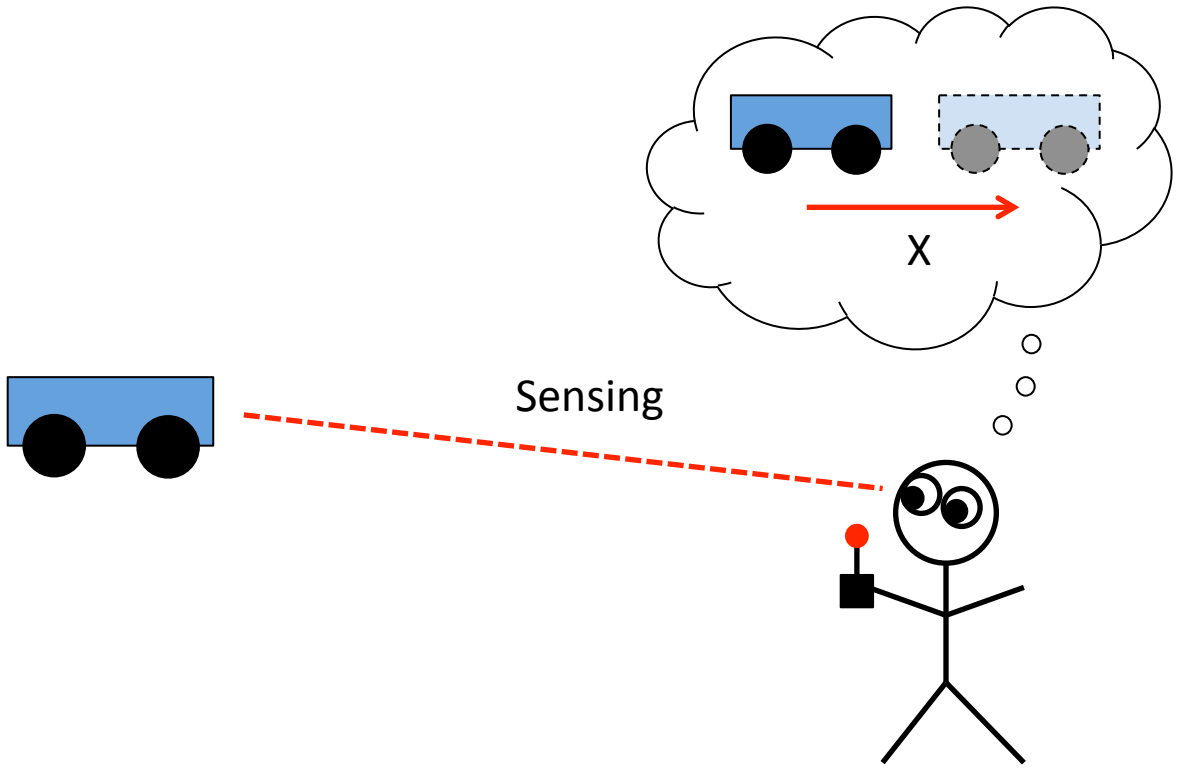


??? Loop

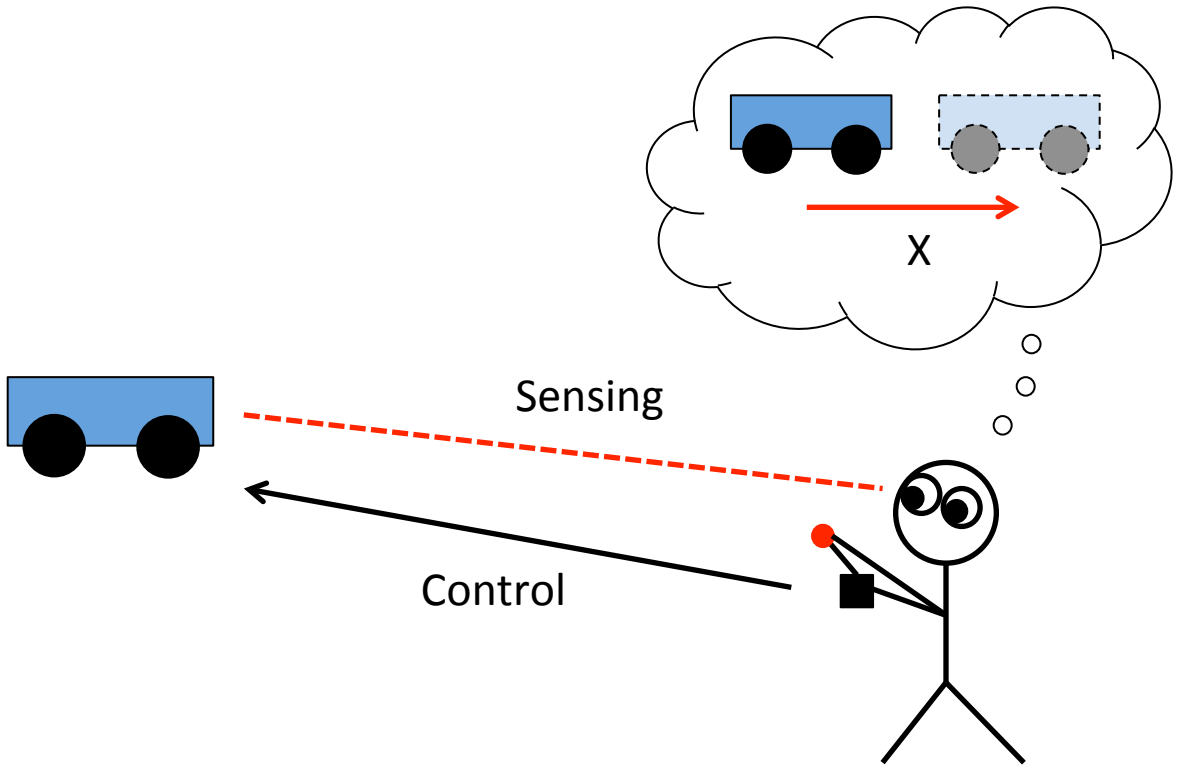




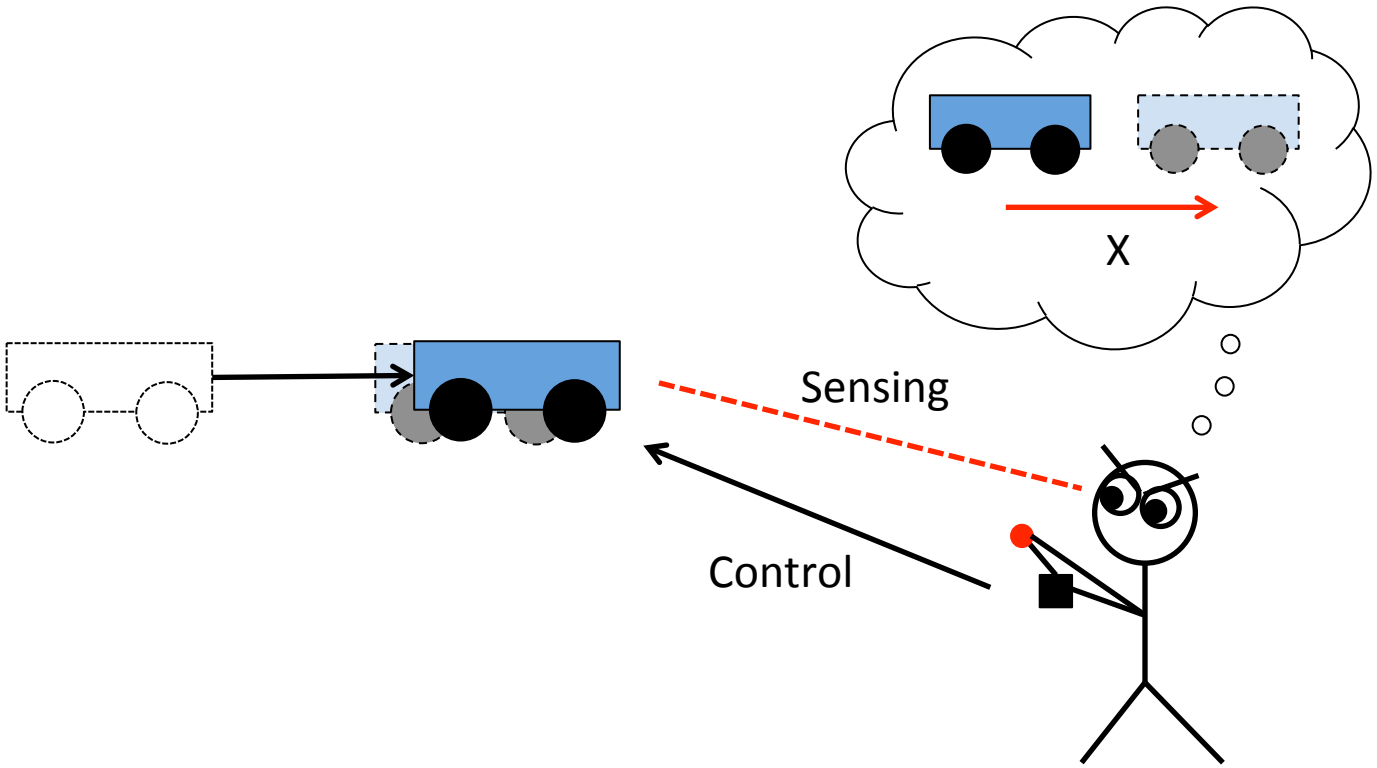
Closed Loop



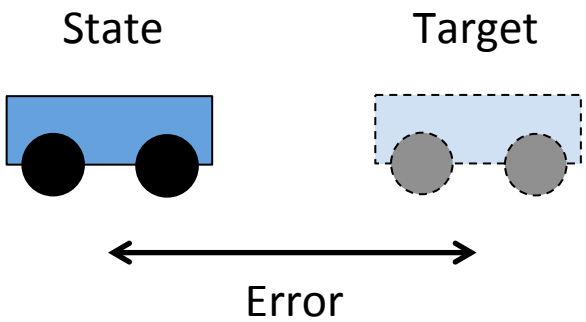
Closed Loop



Closed Loop

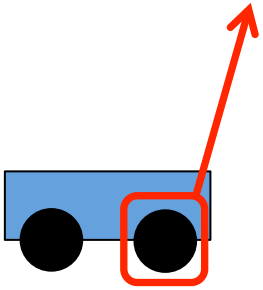


Some terminology

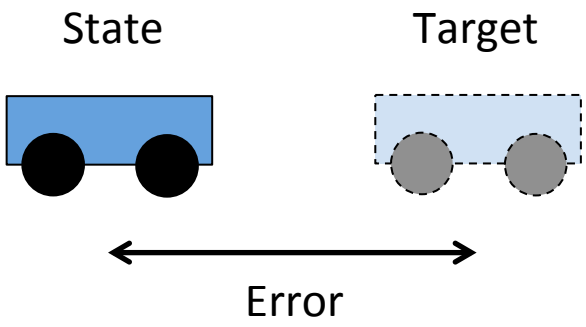


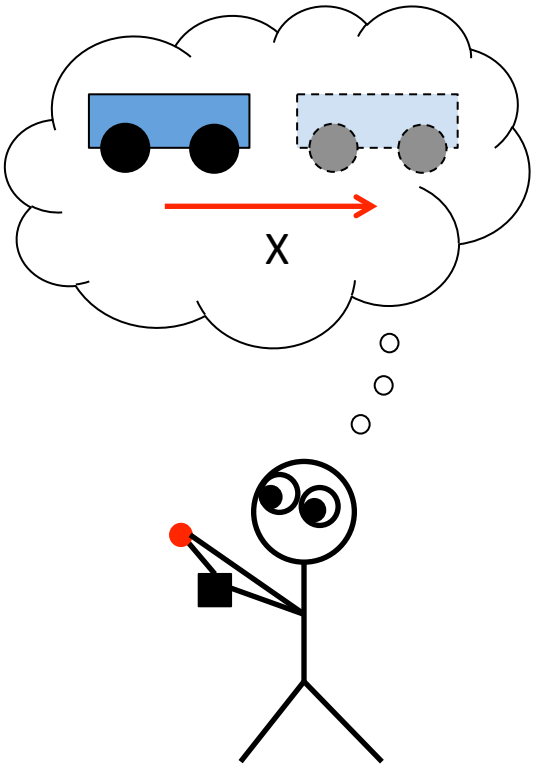
Motor

Encoder

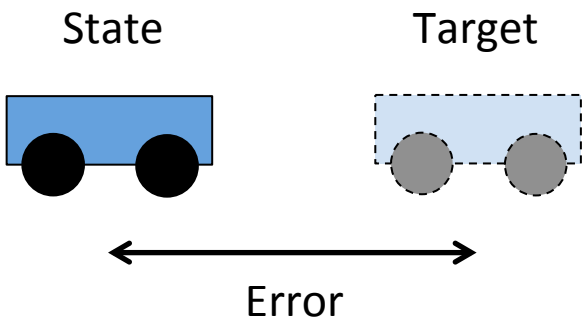


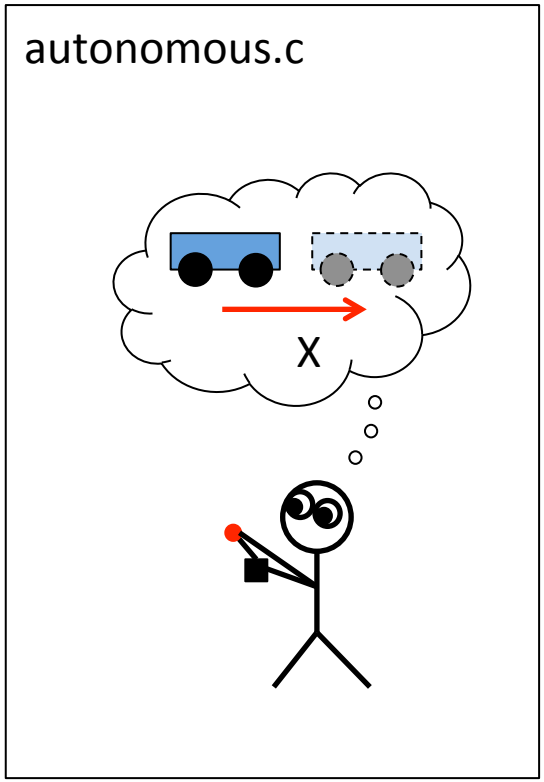
Autonomous Mode



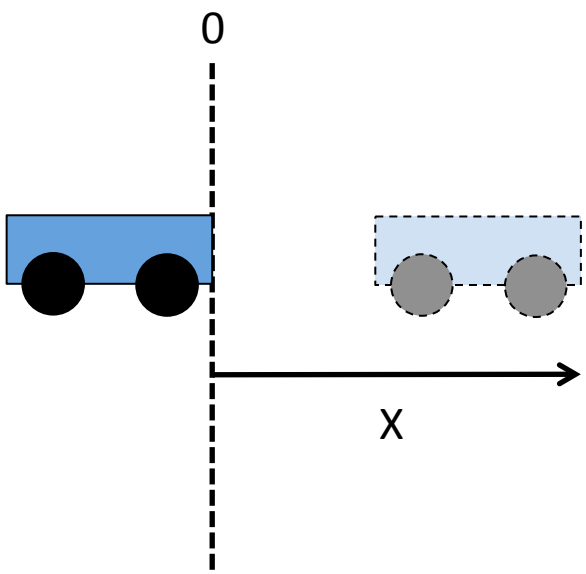


Autonomous Mode





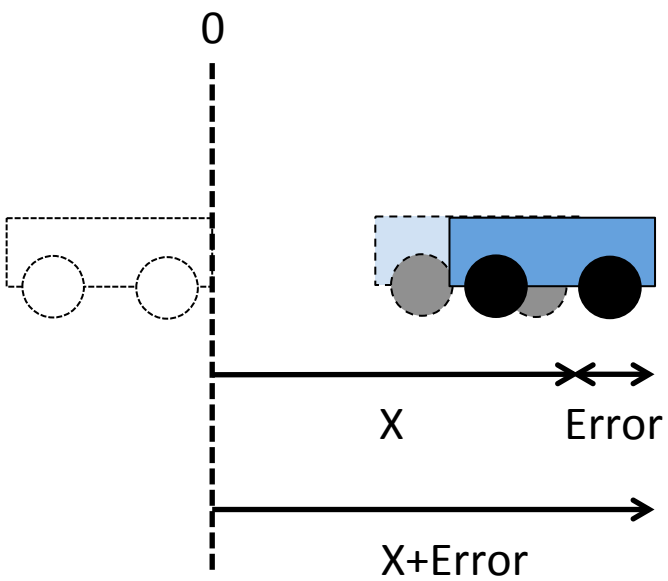
Autonomous Mode



autonomous.c

```
void go()
{
        ..
        robot.drive(X);
        ..
}
```

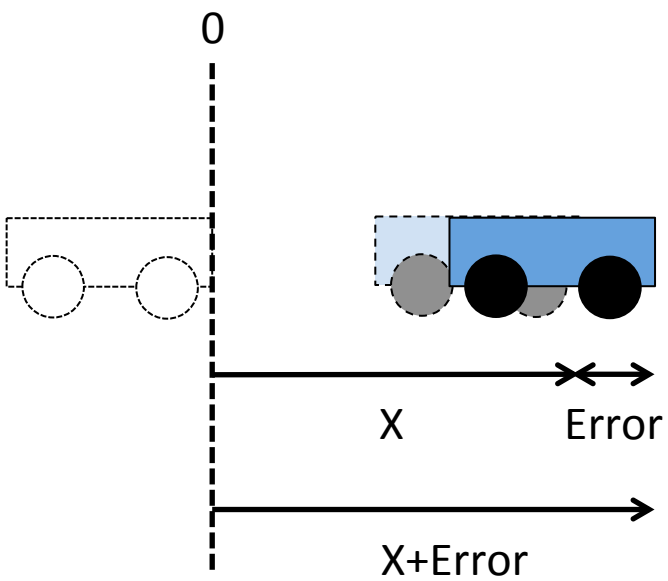
Autonomous Mode



autonomous.c

```
void go()
{
        ..
        robot.motors(100);
        wait(1_SEC);
        robot.motors(0);
         ..
}
```

Autonomous Mode



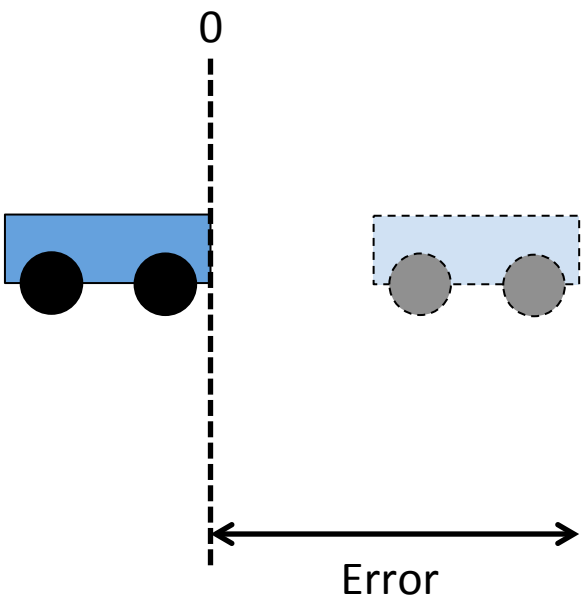
autonomous.c

```
void go()
{
        ..
        robot.motors(100);

        while(encoder < 1000)
        {
          encoder = robot.read();
        }

        robot.motors(0);
         ..
}
```

PID Control

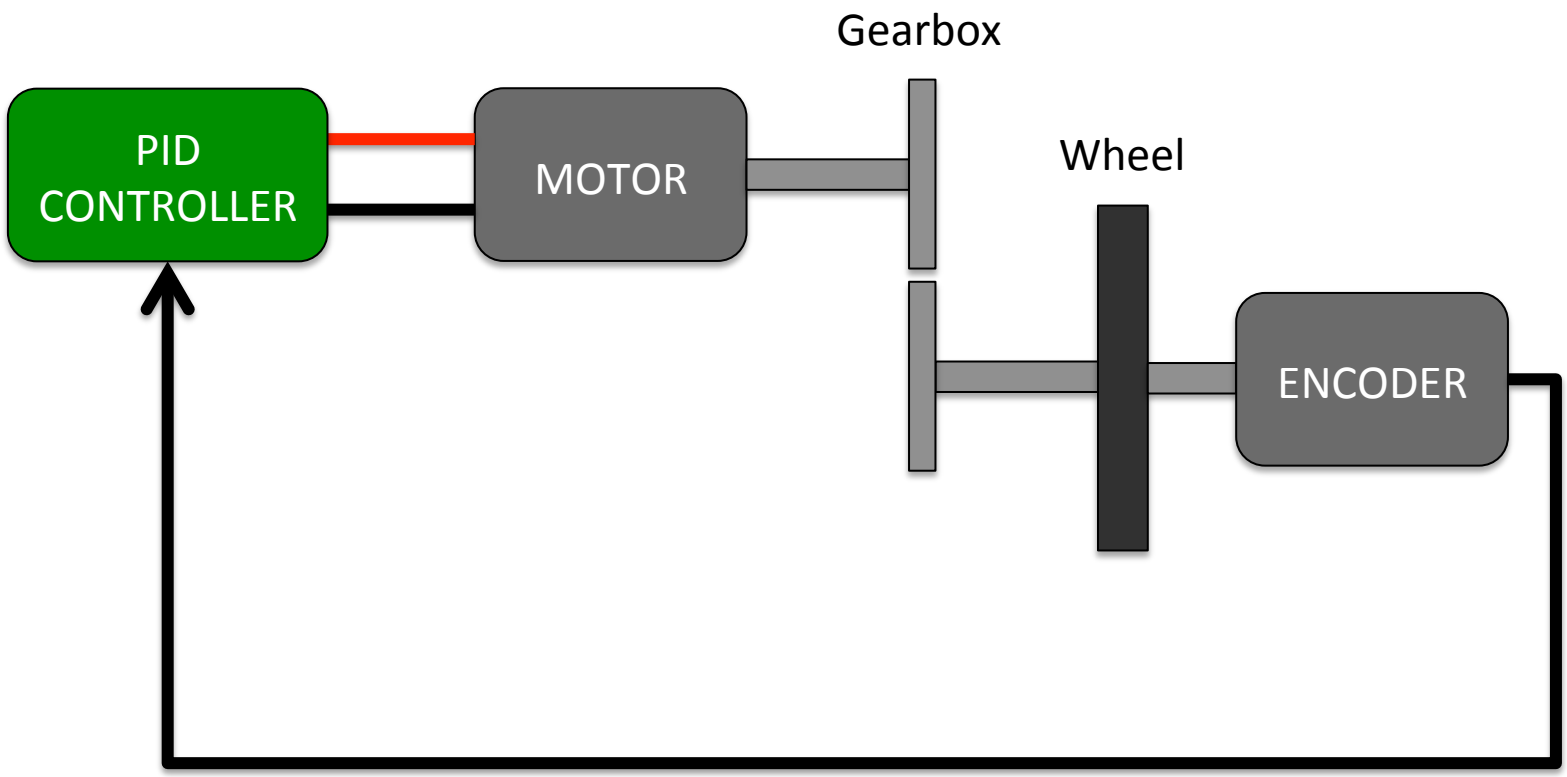


PID Control

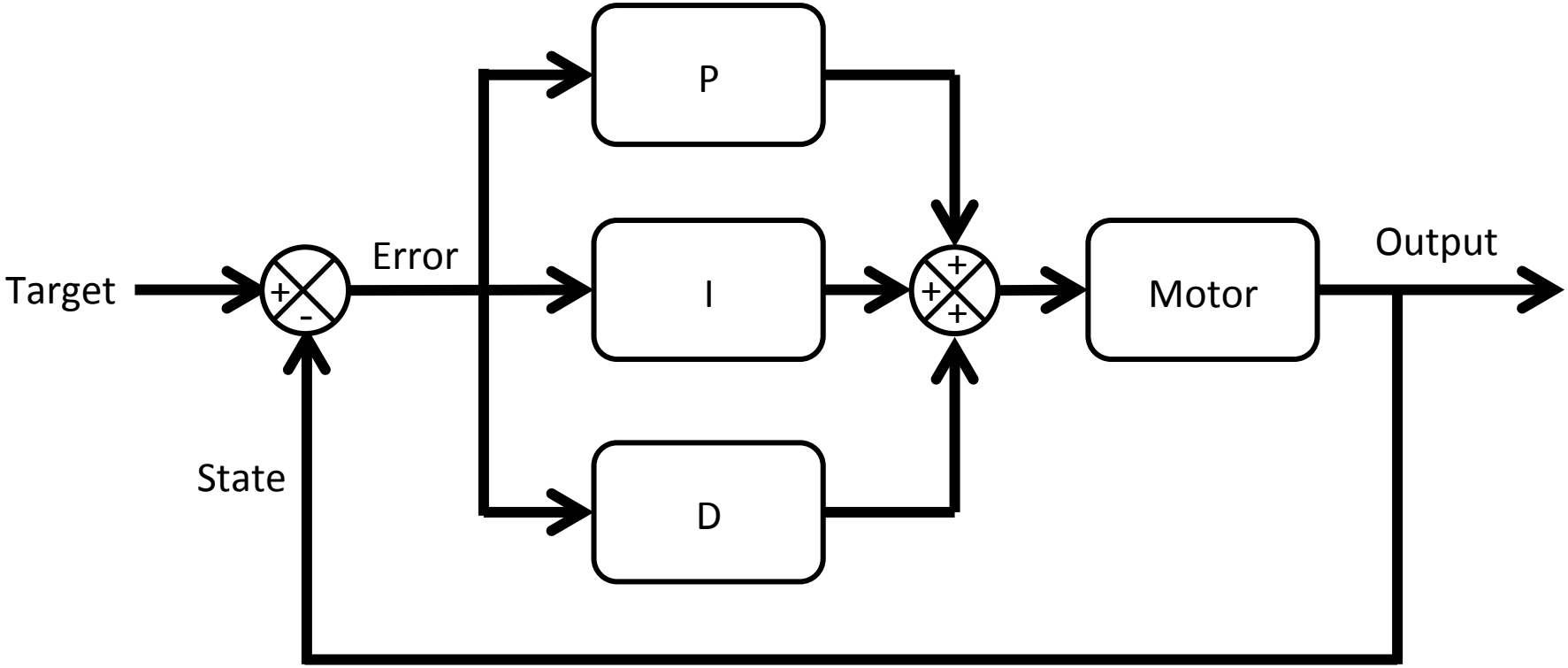
Proportional	**I**ntegral	**D**erivative
Present	Past	Future
P*Error	**+** **I***Sum_of_Errors	**+** **D***(Error- Last_Error)
5	2	1
1	5	2

Position Control Using PID

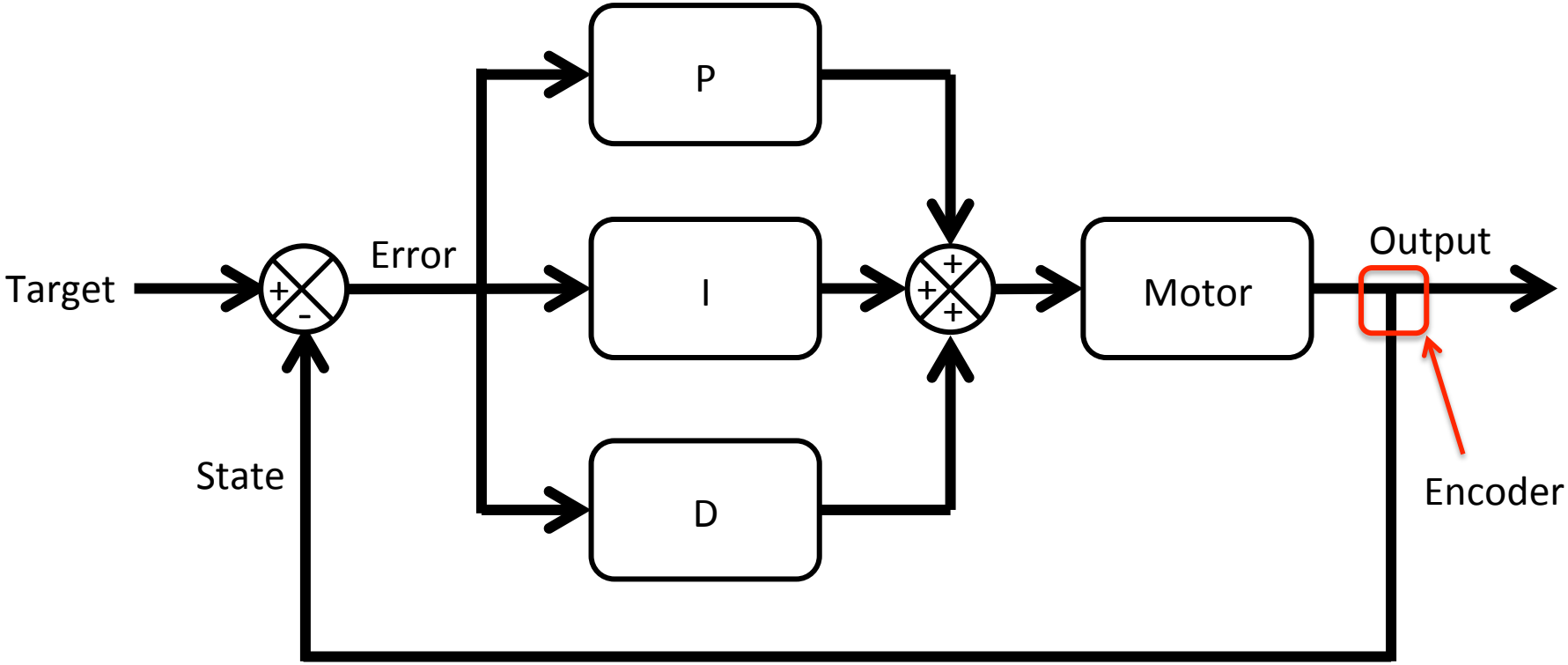
Design



PID Block Diagram

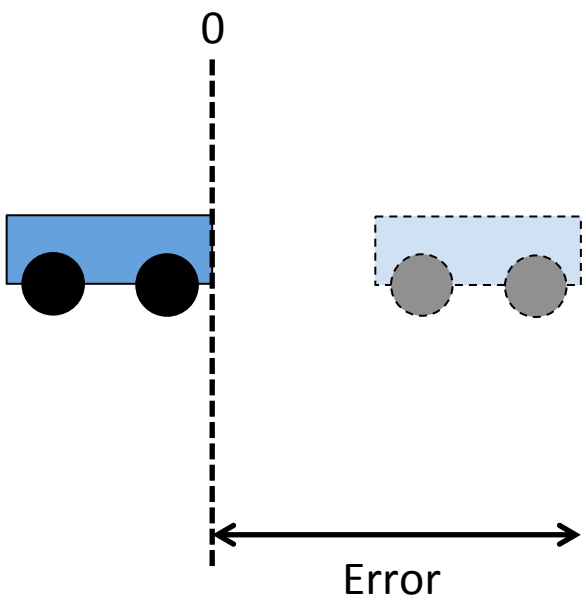


PID Block Diagram



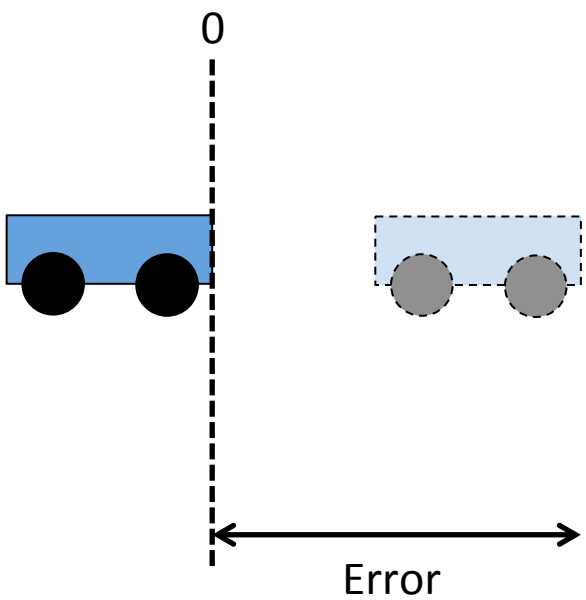
Proportional Control

Proportional Control



Motor = P * Error

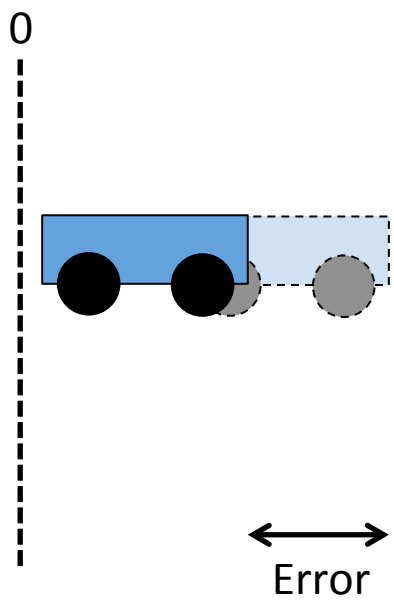
Proportional Control



Motor = P * Error

100 = 2 * 50

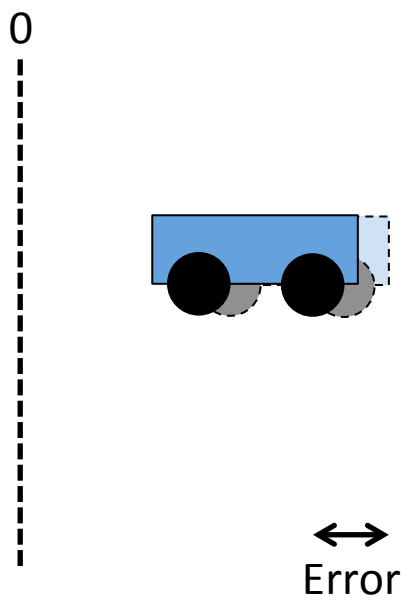
Proportional Control



Motor = P * Error

20 = 2 * 10

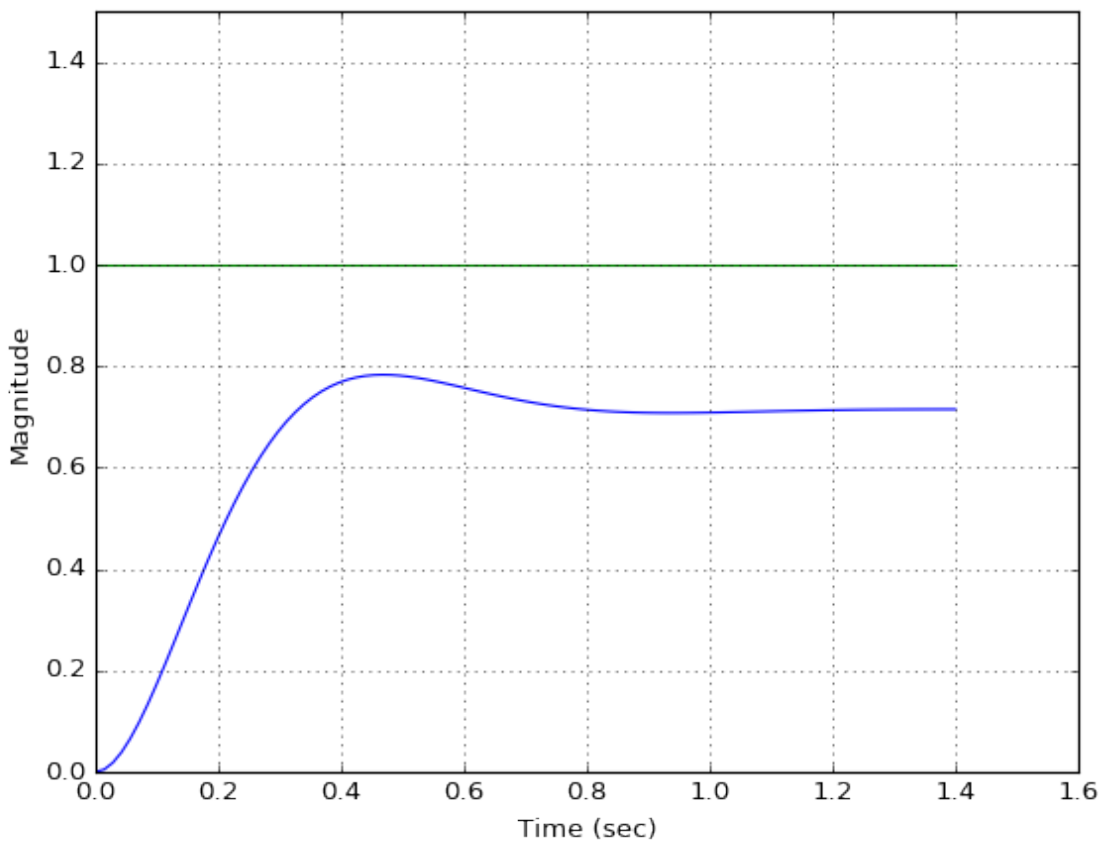
Proportional Control



Motor = P * Error

1 = 2 * 1

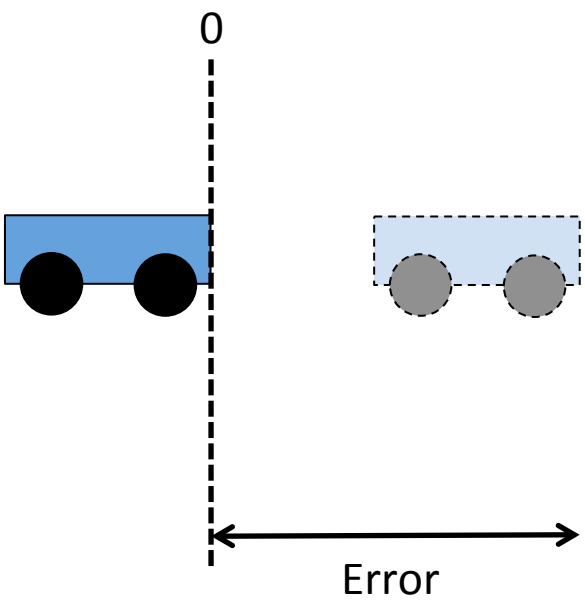
Proportional Control



Integral Control

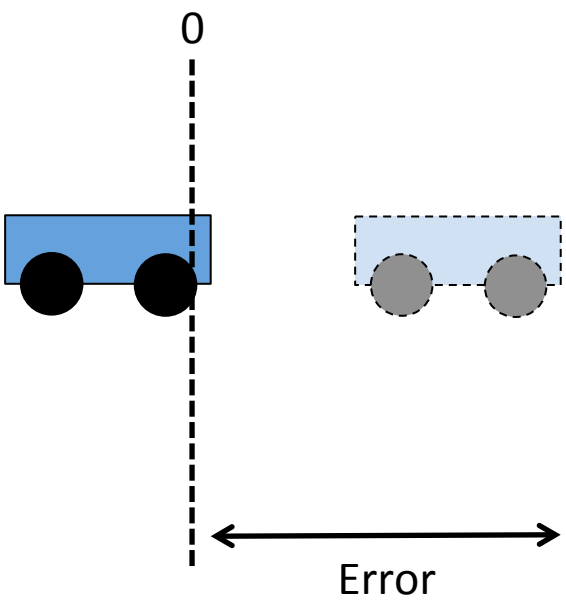
Integral Control

Error_Sum = Error_Sum + Error
50 = 0 + 50



Motor = I * Error_Sum
25 = 0.5 * 50

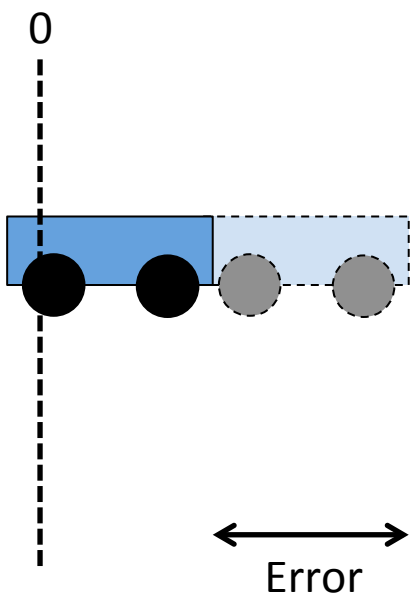
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45

Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95

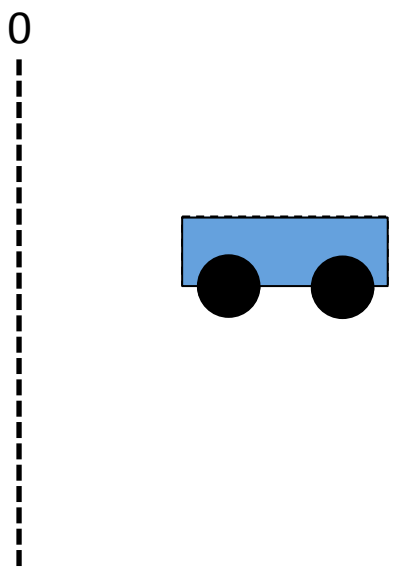
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45
115	= 95 + 20

Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95
56	= 0.5 * 115

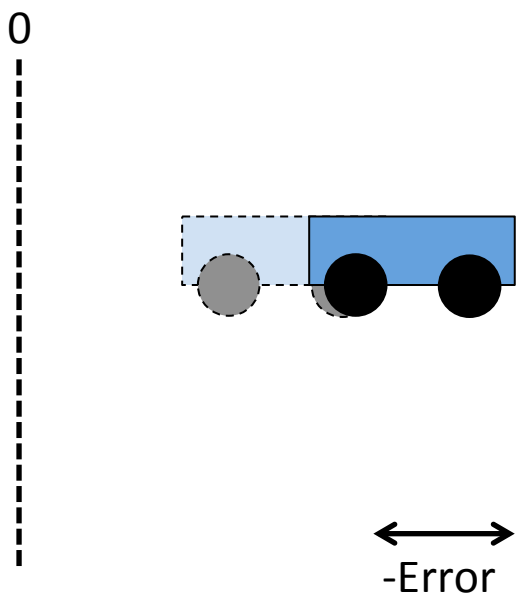
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45
115	= 95 + 20
115	= 115 + 0

Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95
56	= 0.5 * 115
56	= 0.5 * 115

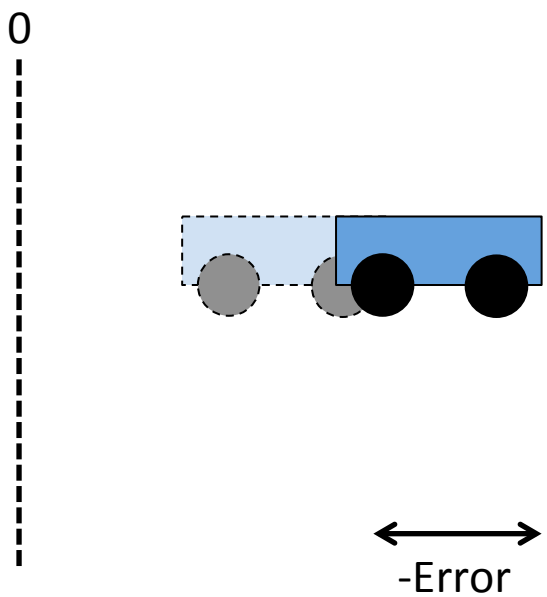
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45
115	= 95 + 20
115	= 115 + 0
75	= 115 + -40

Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95
56	= 0.5 * 115
56	= 0.5 * 115
38	= 0.5 * 75

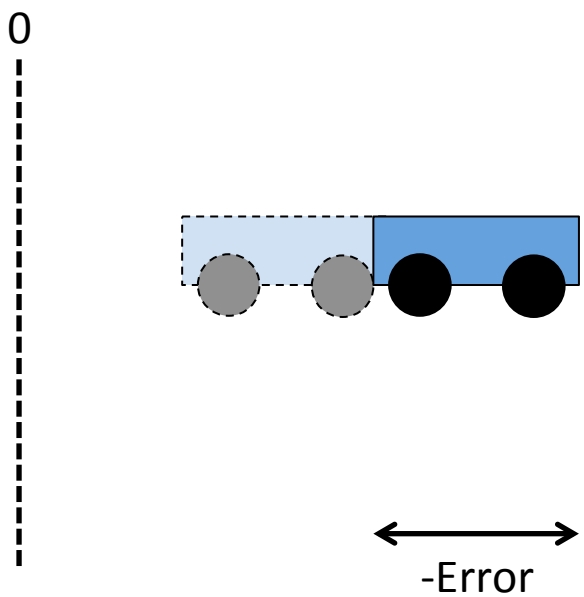
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45
115	= 95 + 20
115	= 115 + 0
75	= 115 + -40
25	= 75 + -50

Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95
56	= 0.5 * 115
56	= 0.5 * 115
38	= 0.5 * 75
13	= 0.5 * 25

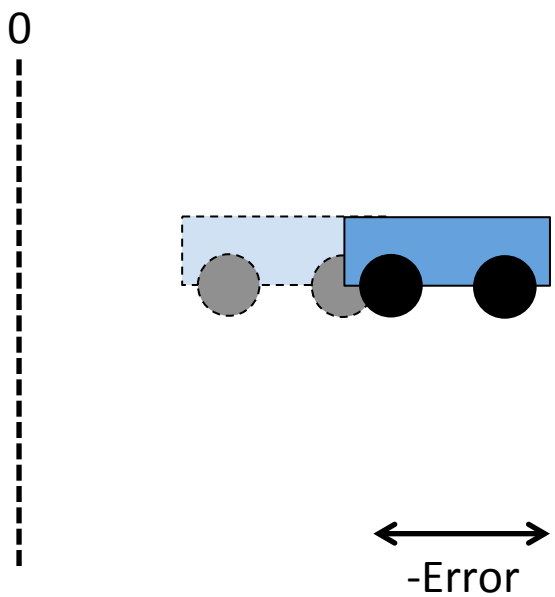
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45
115	= 95 + 20
115	= 115 + 0
75	= 115 + -40
25	= 75 + -50
-35	= 25 + -60

Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95
56	= 0.5 * 115
56	= 0.5 * 115
38	= 0.5 * 75
13	= 0.5 * 25
-17	= 0.5 * -35

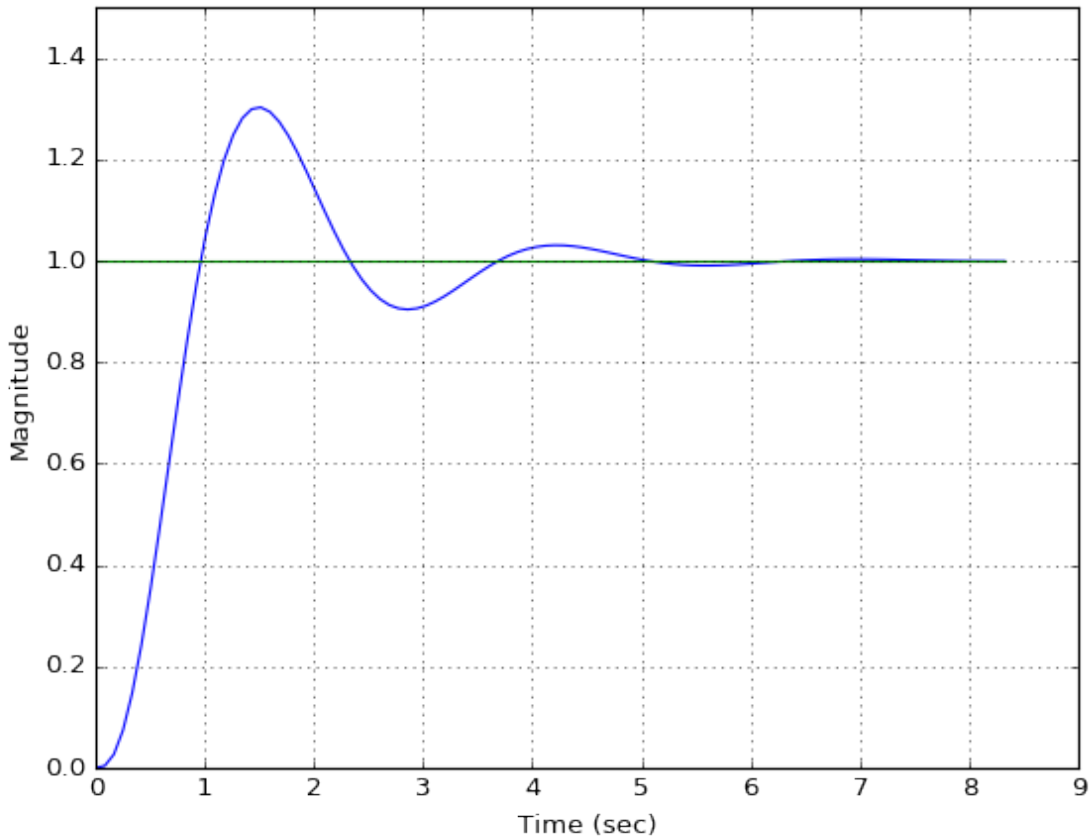
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45
115	= 95 + 20
115	= 115 + 0
75	= 115 + -40
25	= 75 + -50
-35	= 25 + -60
-75	= -35 + -40

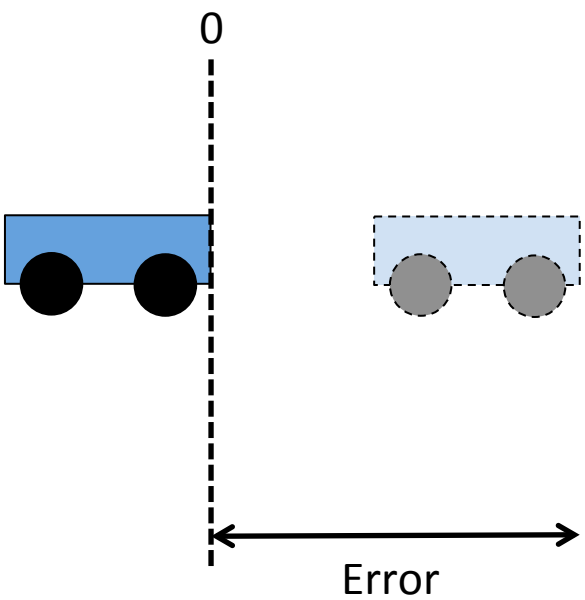
Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95
56	= 0.5 * 115
56	= 0.5 * 115
38	= 0.5 * 75
13	= 0.5 * 25
-17	= 0.5 * -35
-38	= 0.5 * -75

Integral Control



Derivative Control

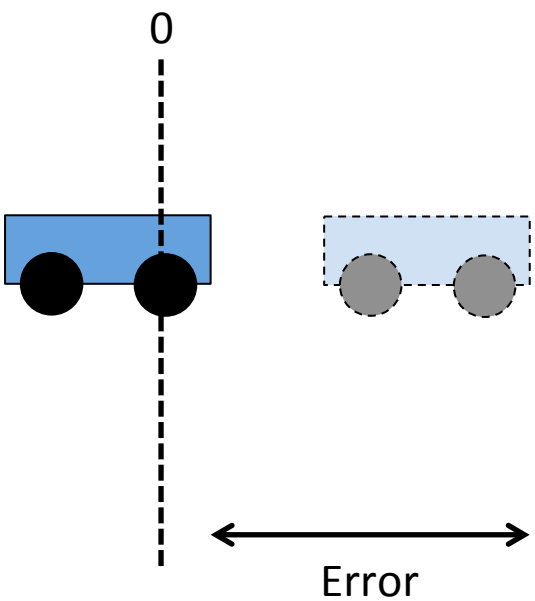
Derivative Control



Motor = D* (Error – Last_Error)

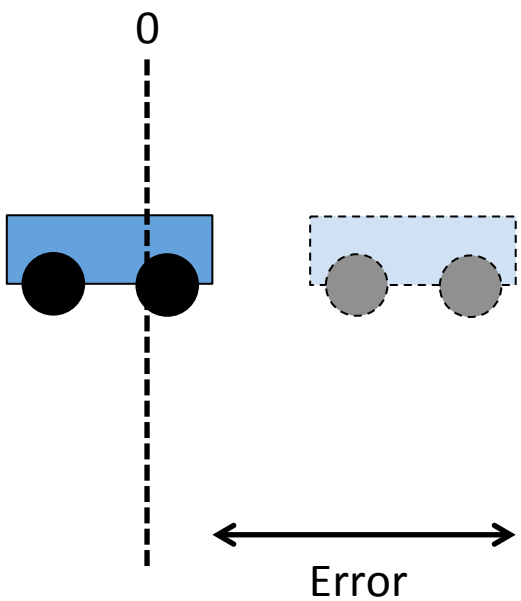
25 = 0.5 * (50 – 0)

Derivative Control



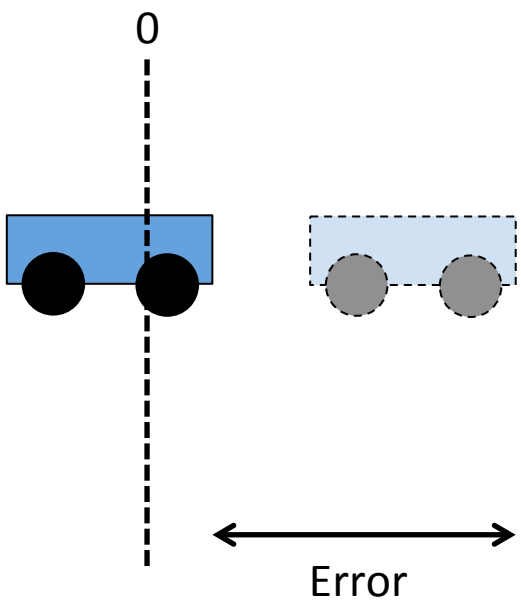
Motor	**= D* (Error – Last_Error)**
25	= 0.5 * (50 – 0)
-5	= 0.5 * (40 – 50)

Derivative Control



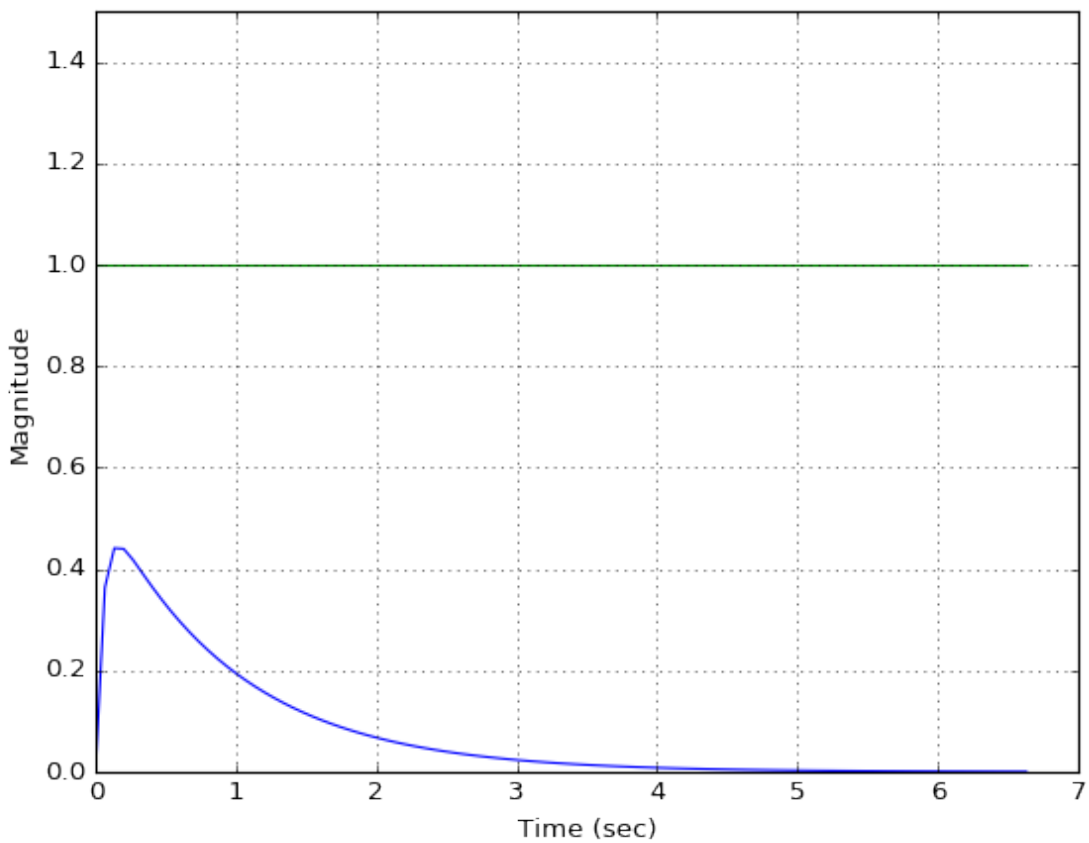
Motor	**= D* (Error – Last_Error)**
25	= 0.5 * (50 – 0)
-5	= 0.5 * (40 – 50)
-2.5	= 0.5 * (35 – 40)

Derivative Control



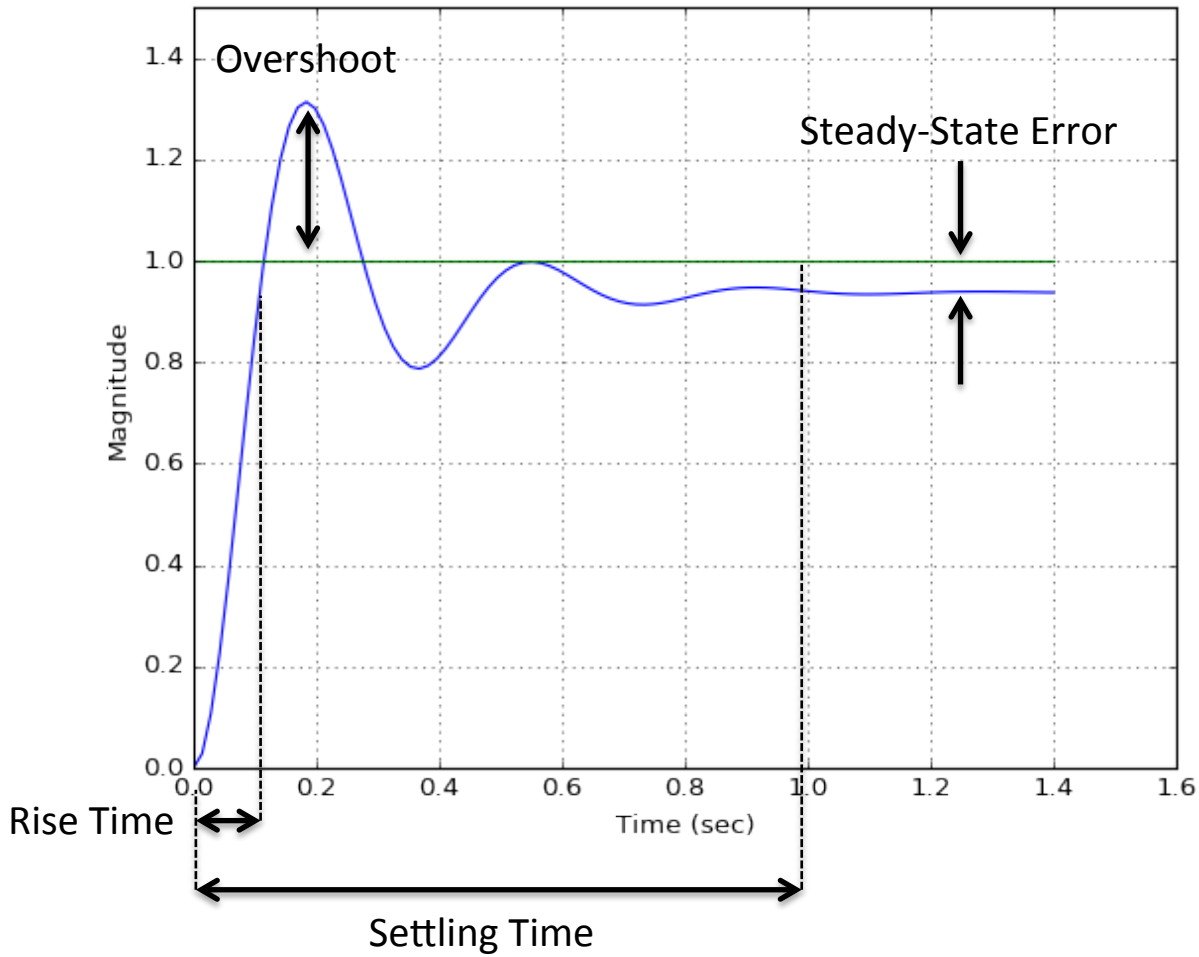
Motor	**= D* (Error – Last_Error)**
25	= 0.5 * (50 – 0)
-5	= 0.5 * (40 – 50)
-2.5	= 0.5 * (35 – 40)
0	= 0.5 * (35 – 35)

Derivative Control

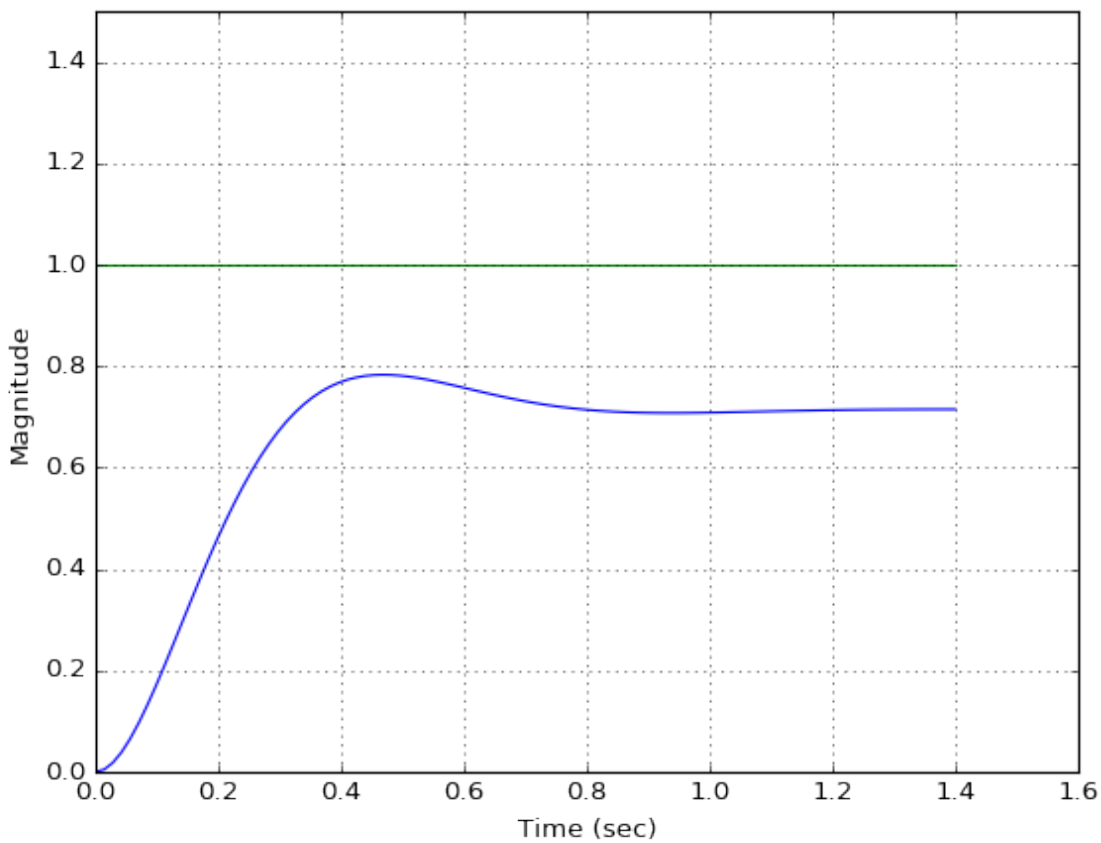


PID Plot Terminology

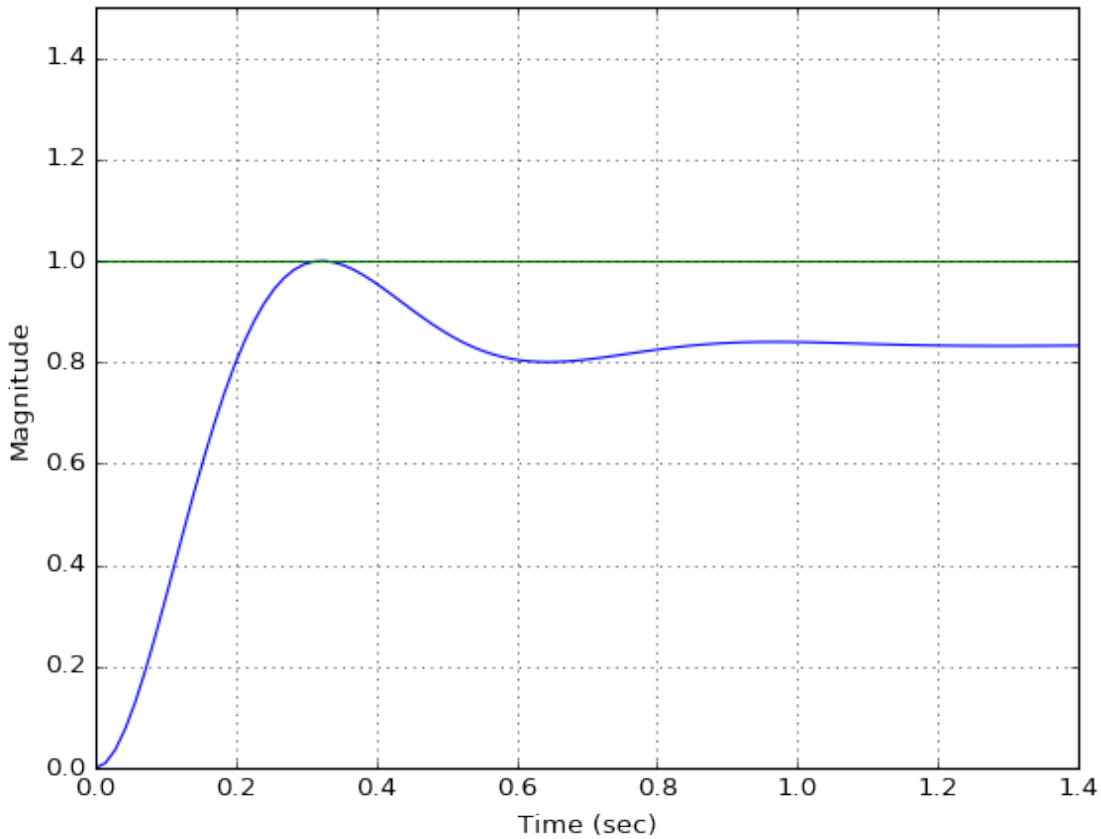
PID Plot Terminology



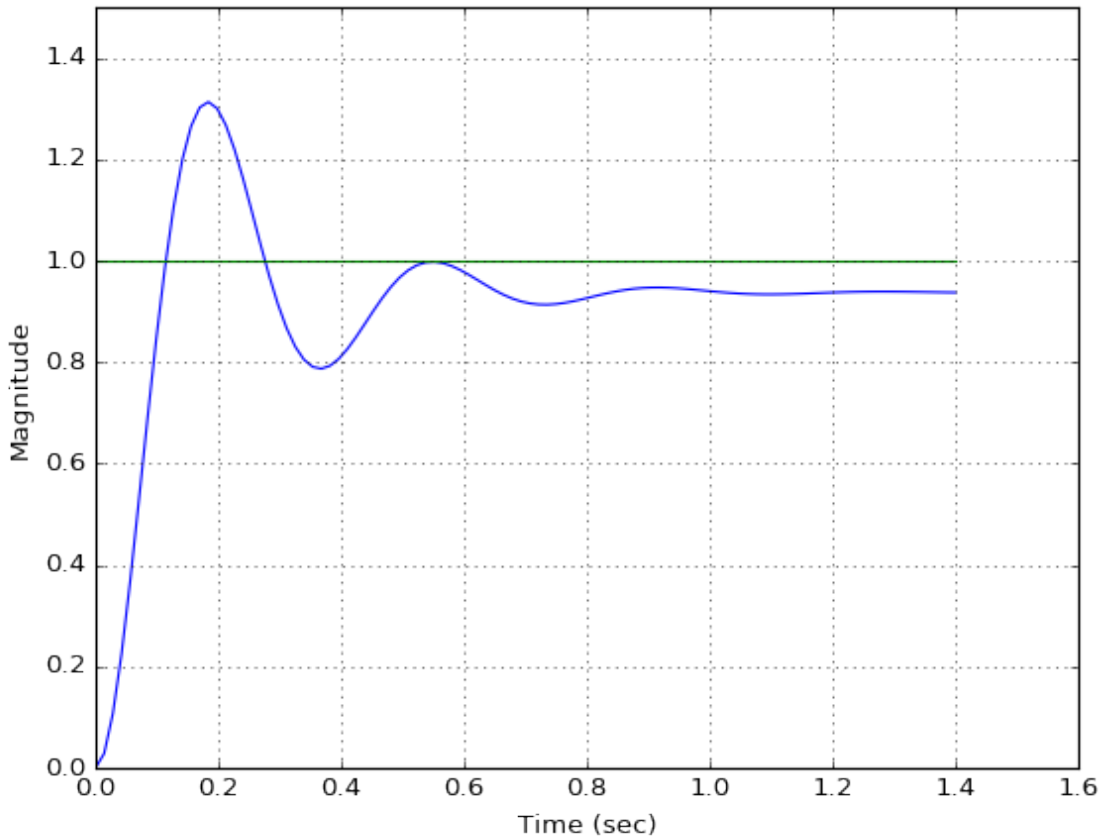
P = 50


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Magnitude
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Time (sec)

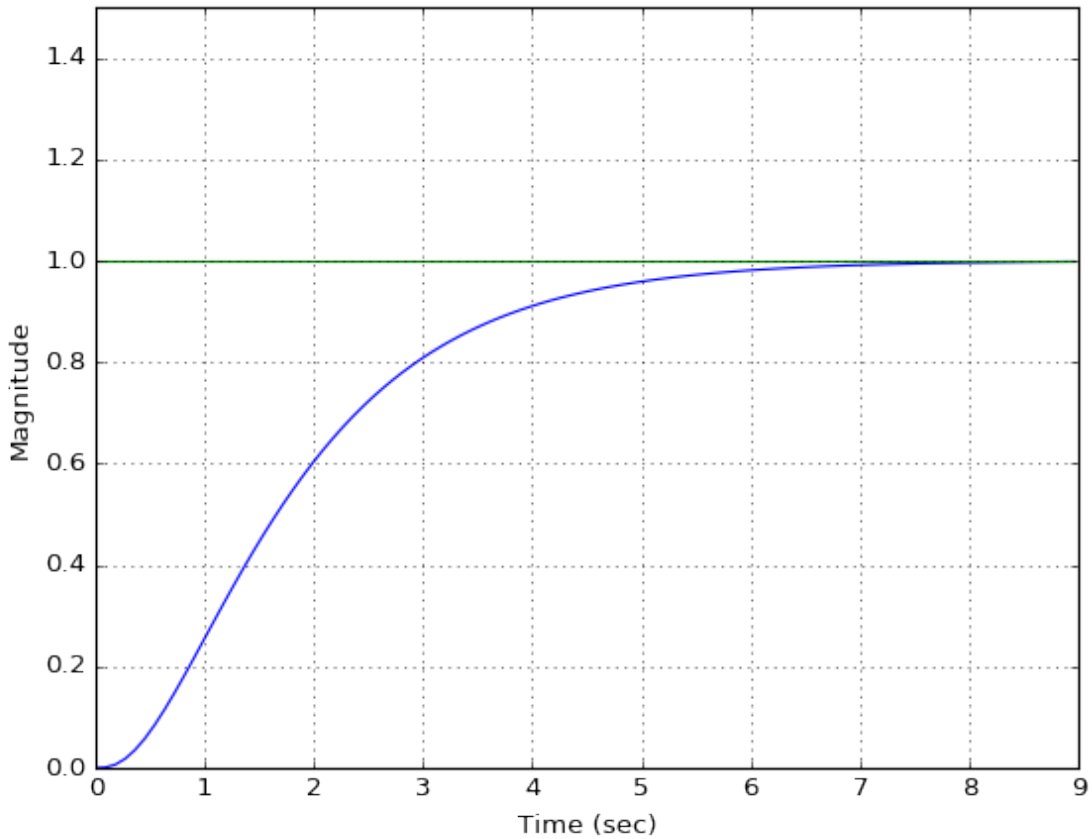
P = 100



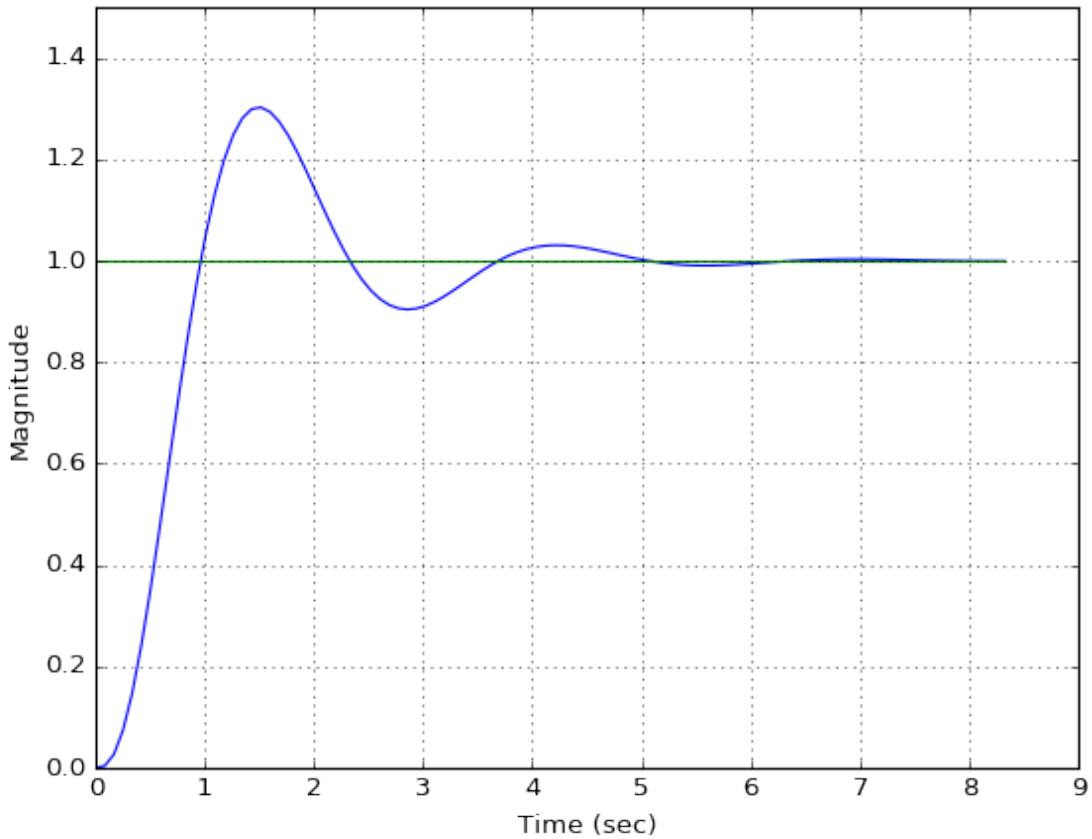
P = 300


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Magnitude
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Time (sec)

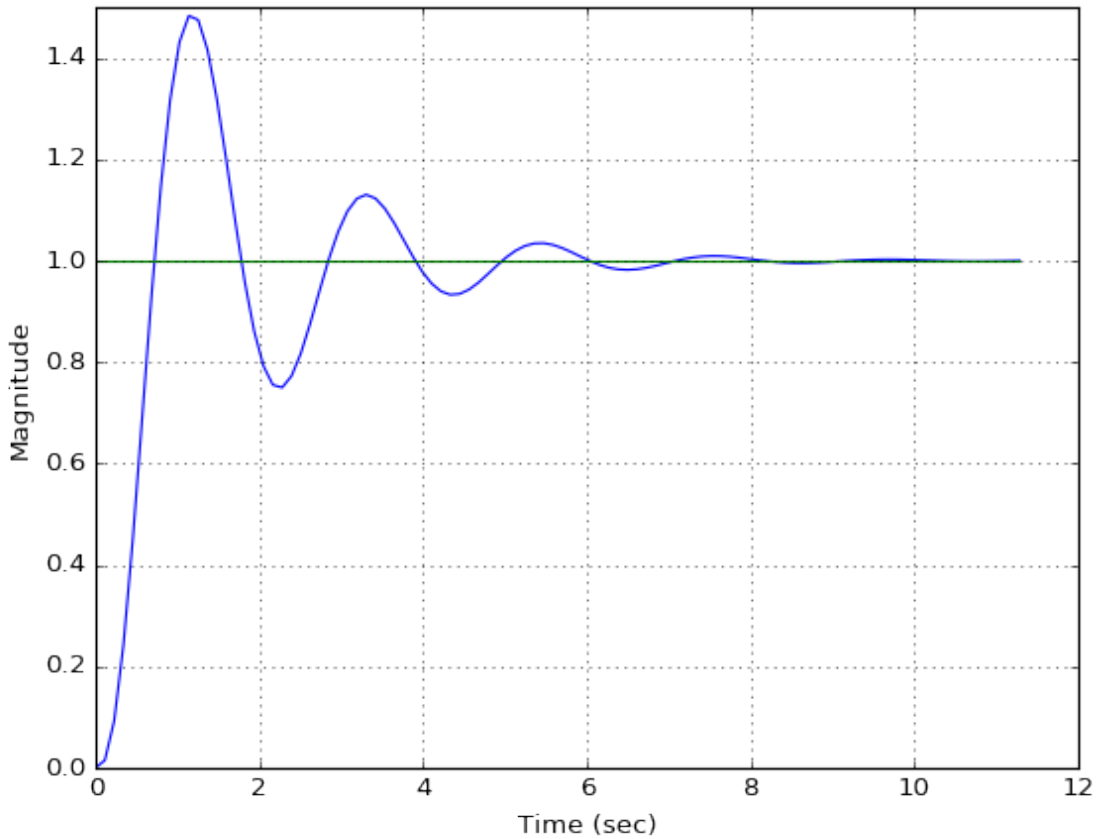
I = 10



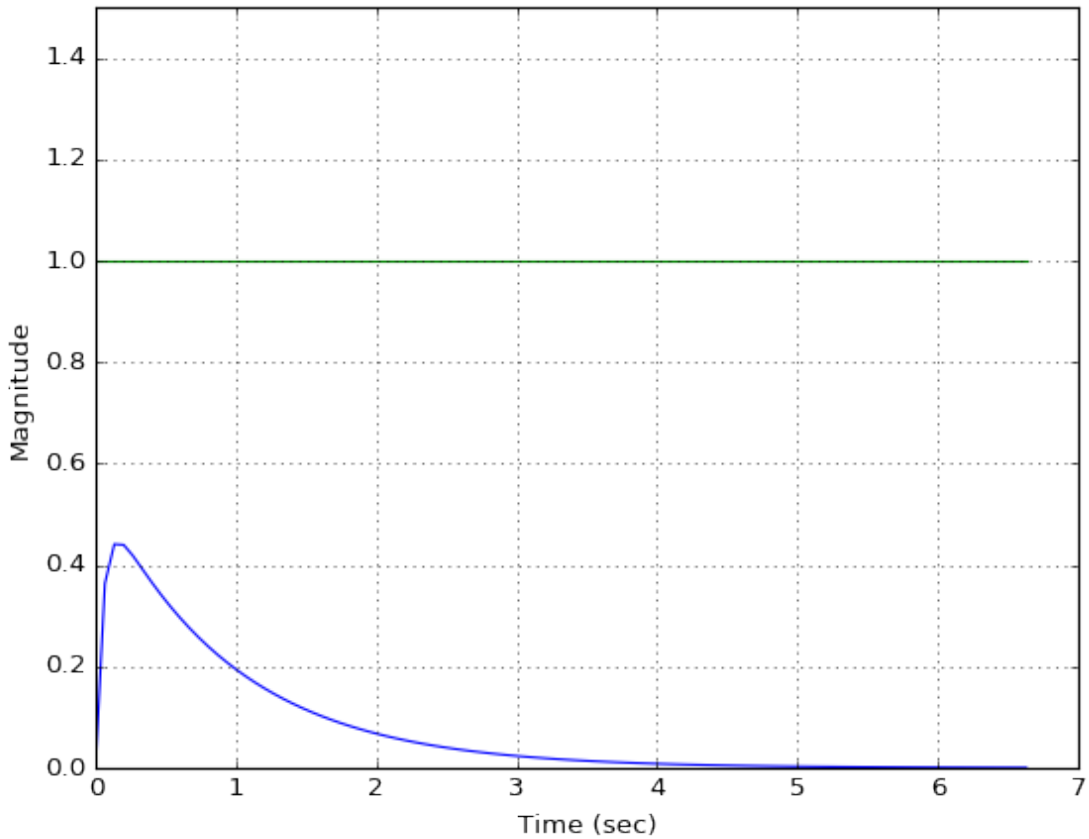
I = 50



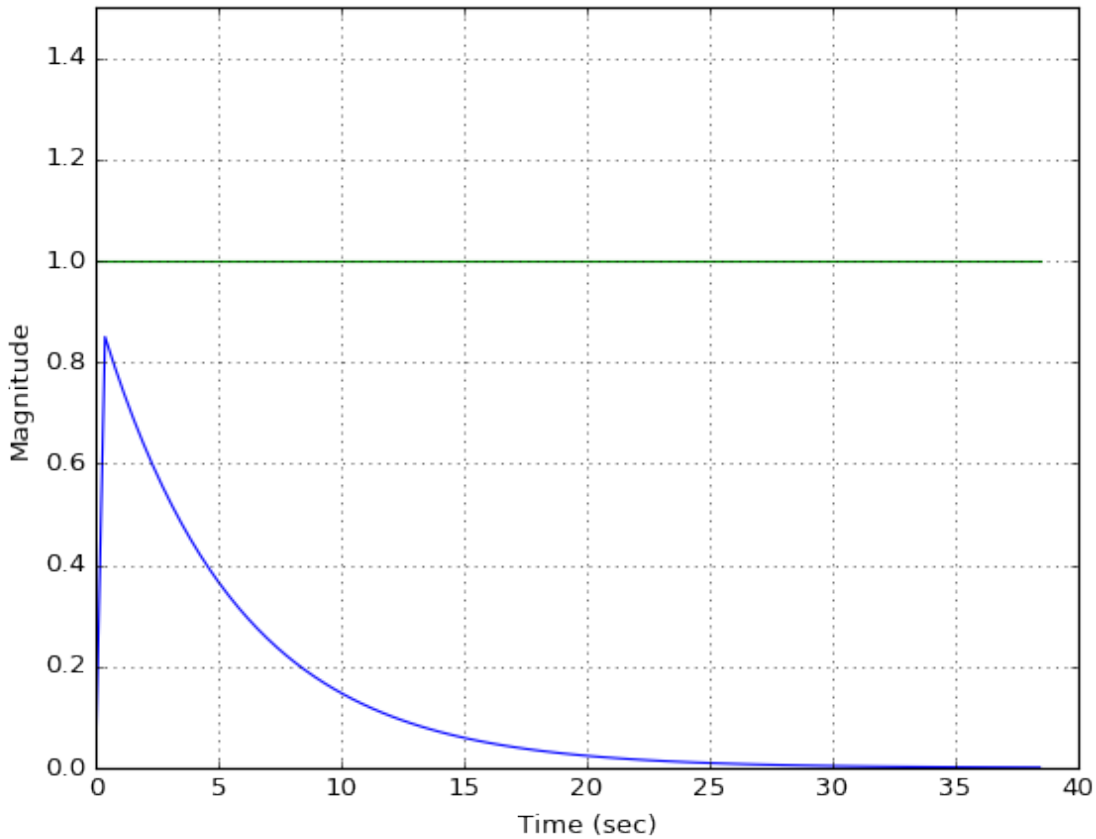
I = 80



D = 10

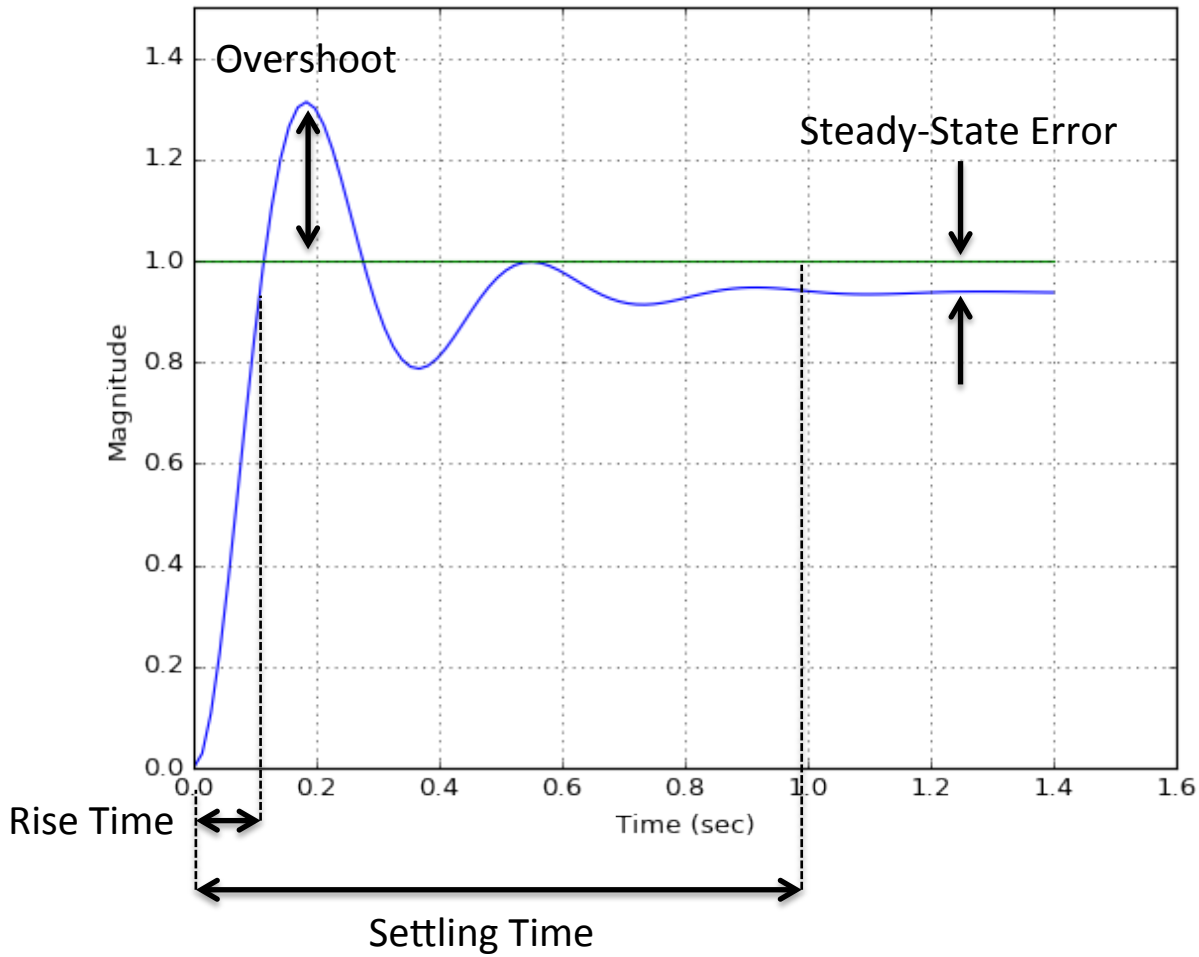


D = 100



PID Tuning Methods

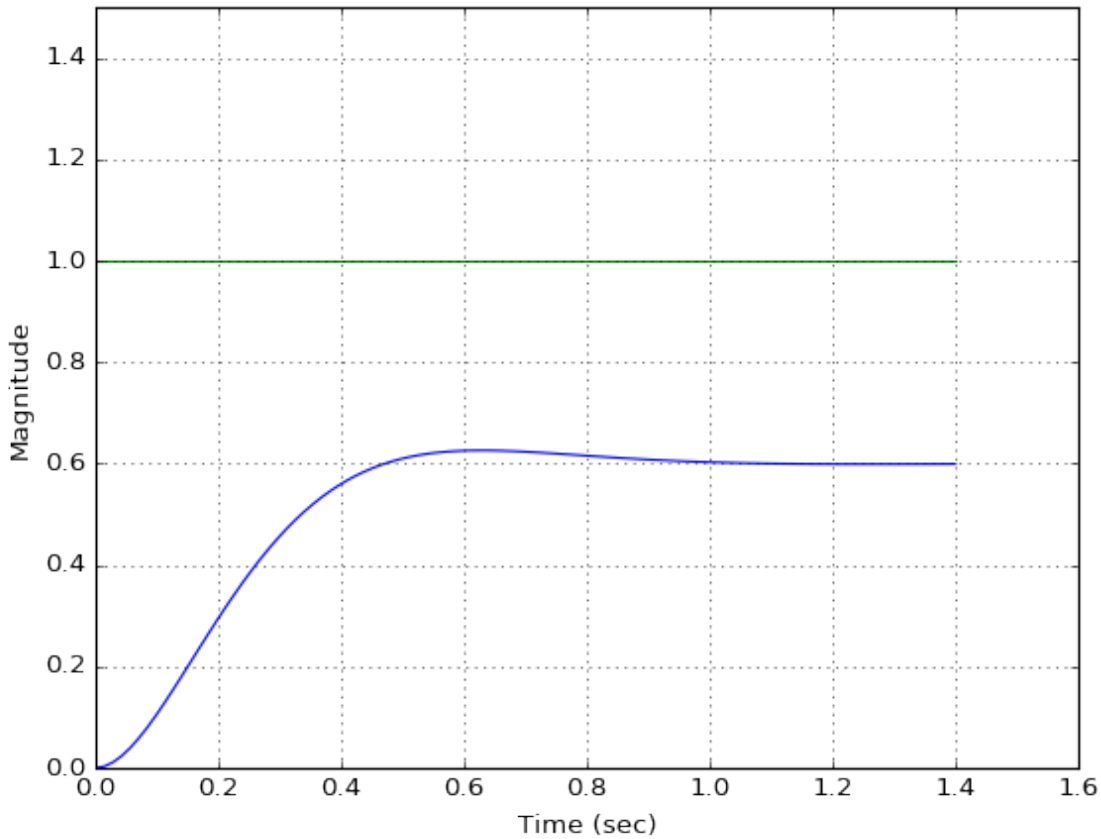
PID Plot Terminology



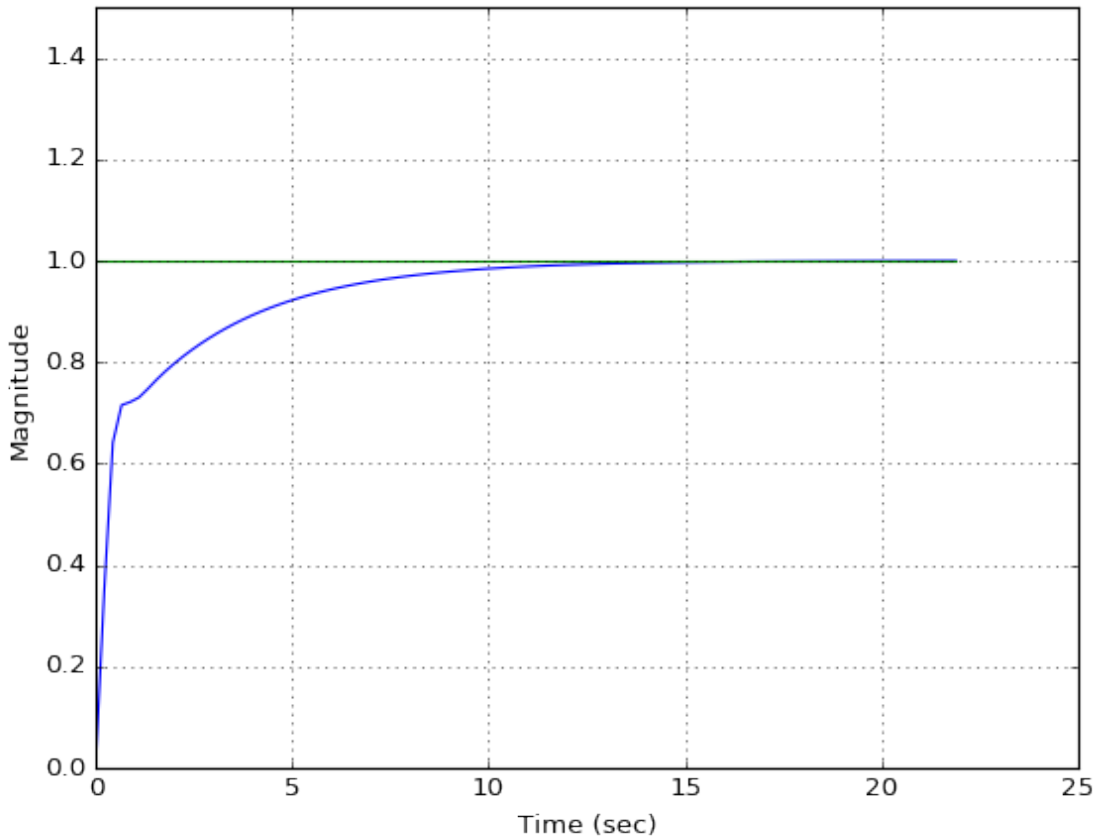
PID Tuning Methods

Parameter Increase	Rise Time	Overshoot	Settling Time	Steady-State Error
Kp	↓	↑	Small Change	↓
Ki	↓	↑	↑	↓ ↓
Kd	Small Change	↓	↓	Small Change

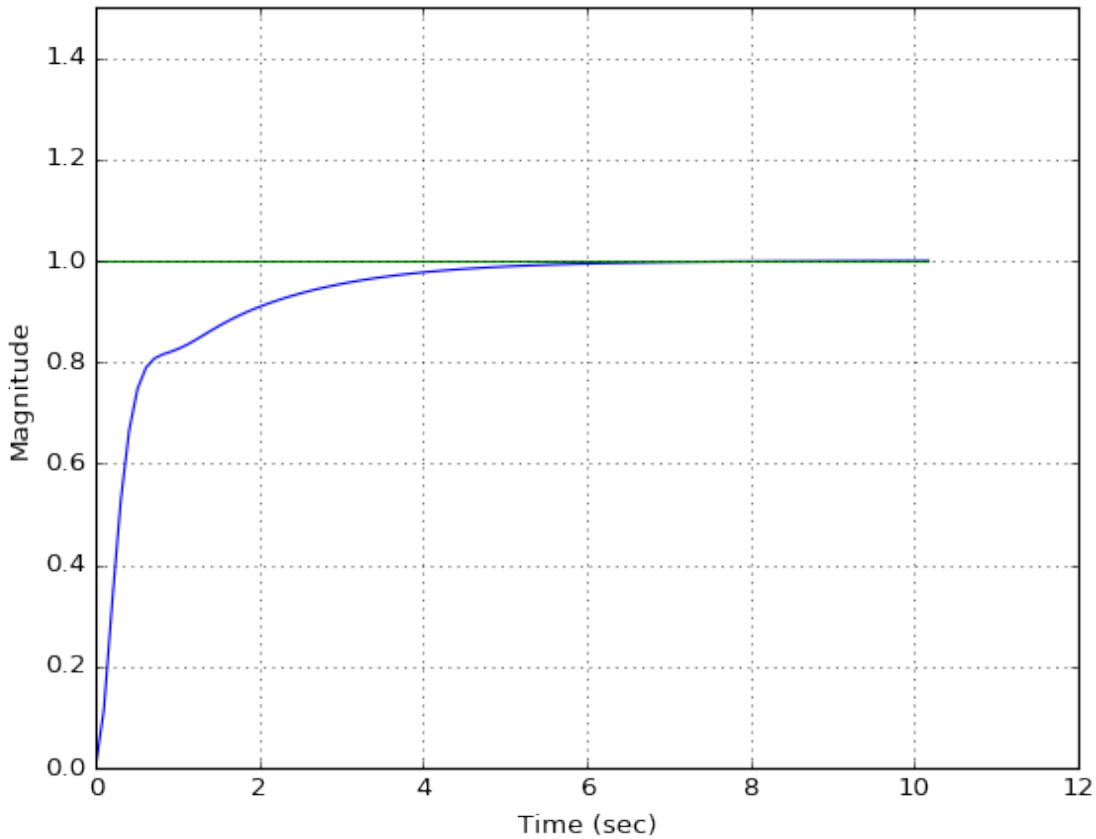
P = 30, I = 0, D = 0


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Magnitude
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Time (sec)

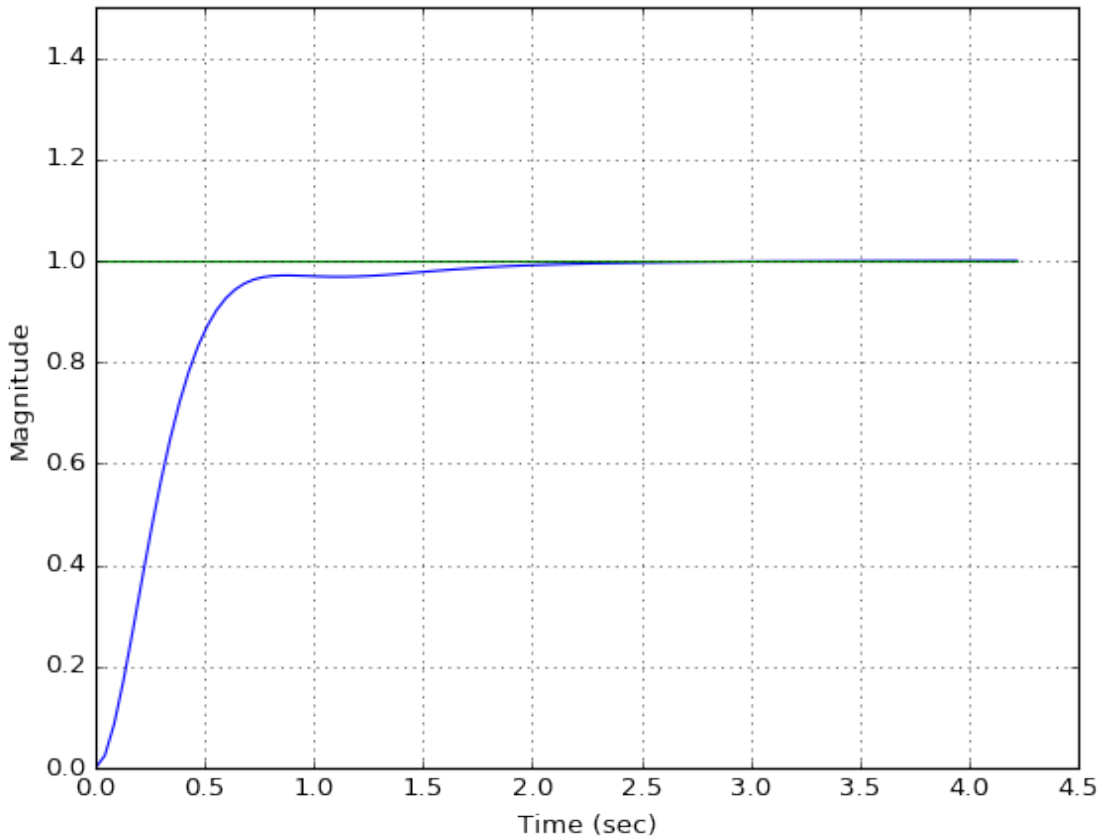
P = 30, I = 15, D = 0



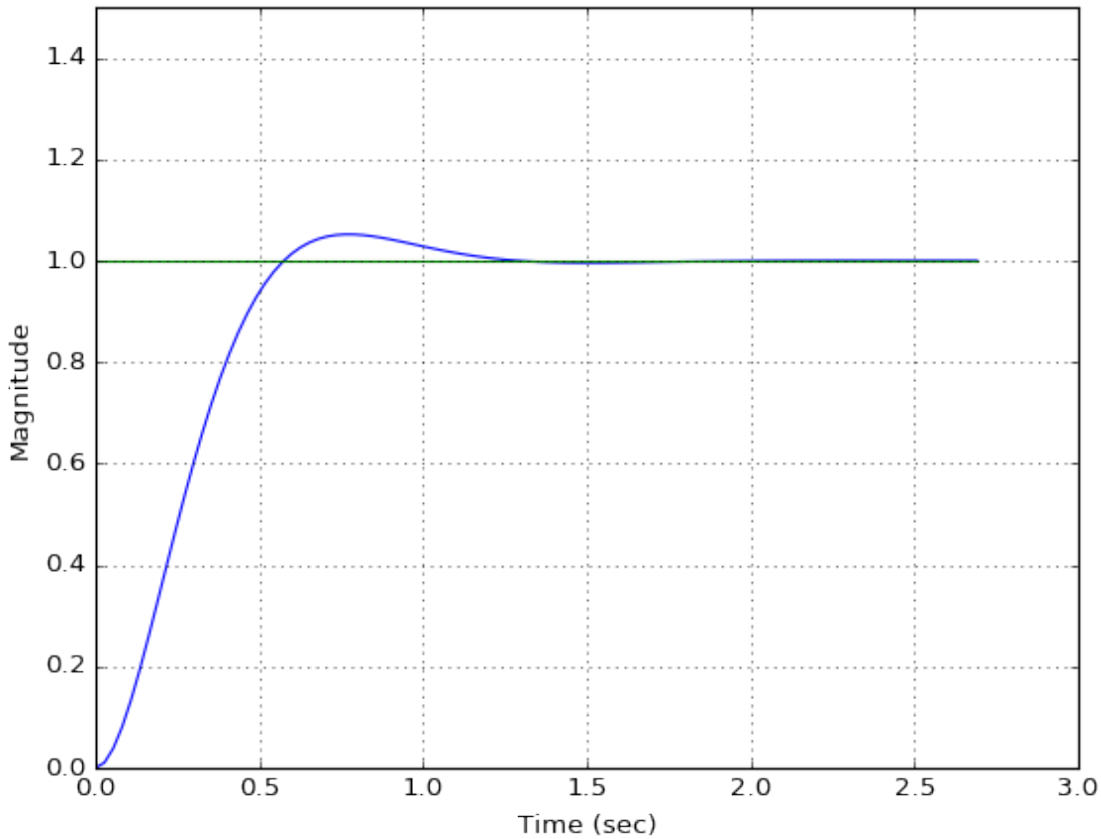
P = 30, I = 30, D = 0



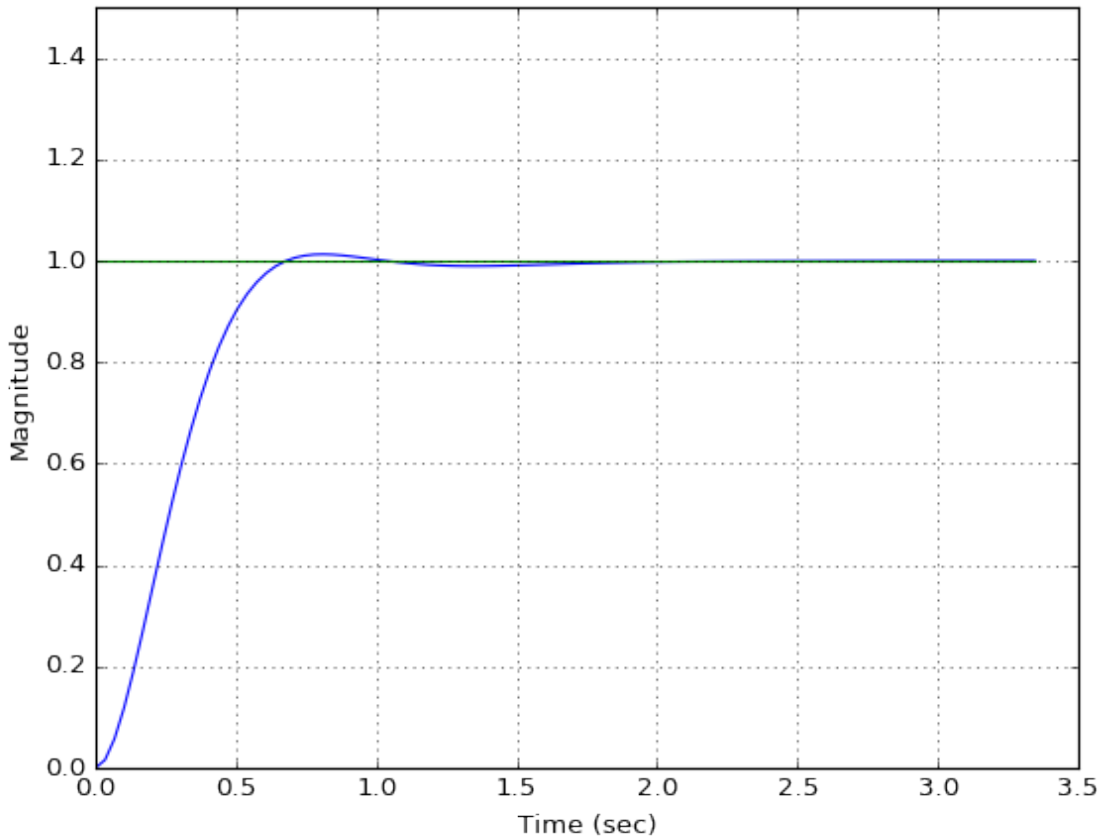
P = 30, I = 60, D = 0


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Magnitude
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
Time (sec)

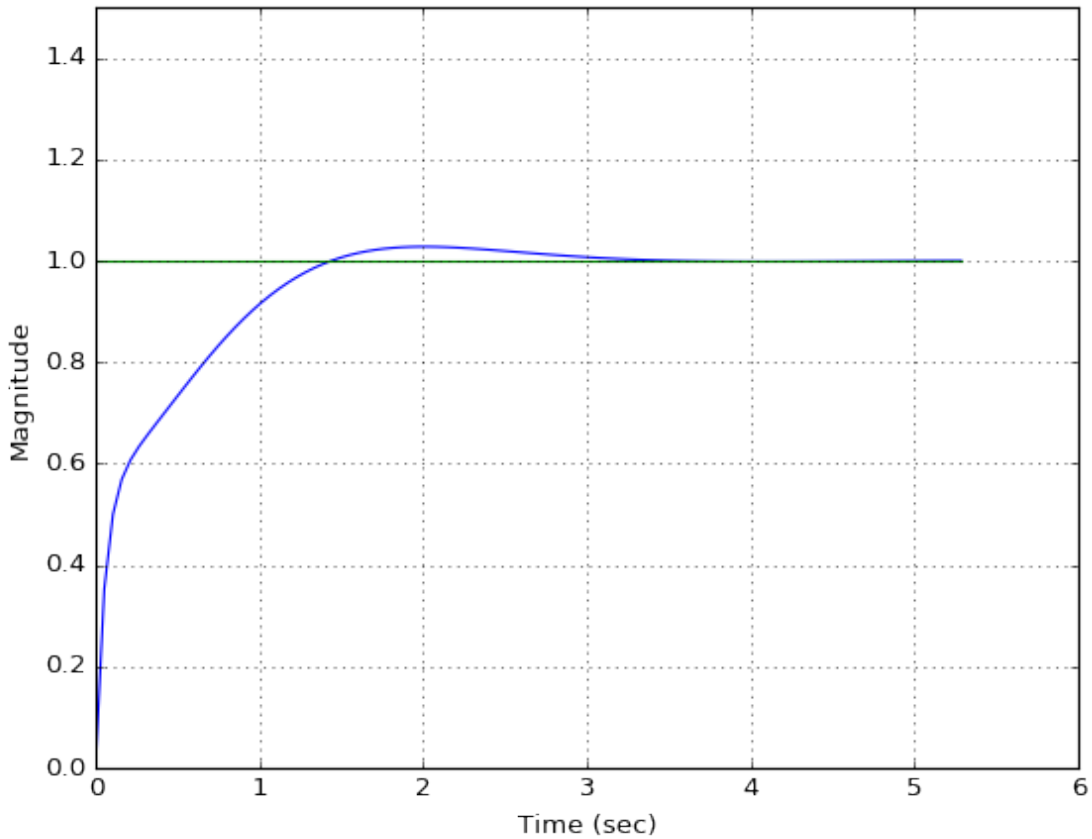
P = 30, I = 80, D = 0



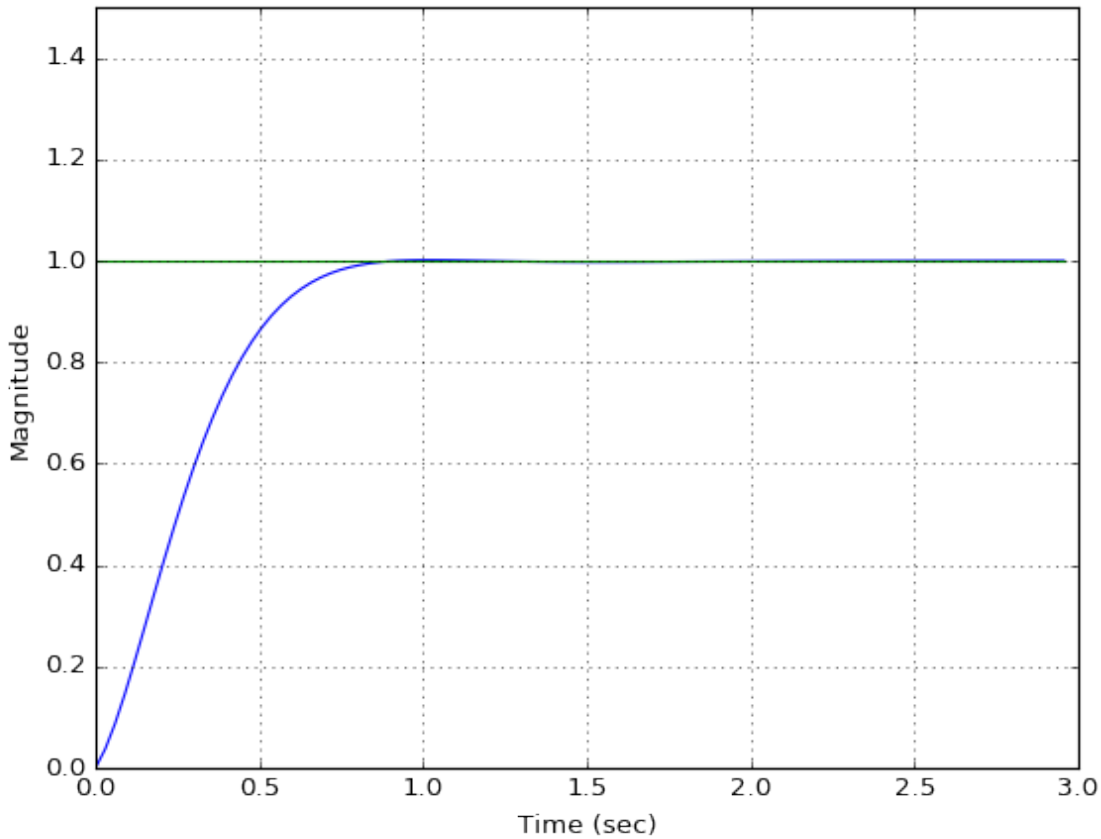
P = 30, I = 70, D = 0



P = 30, I = 70, D = 10



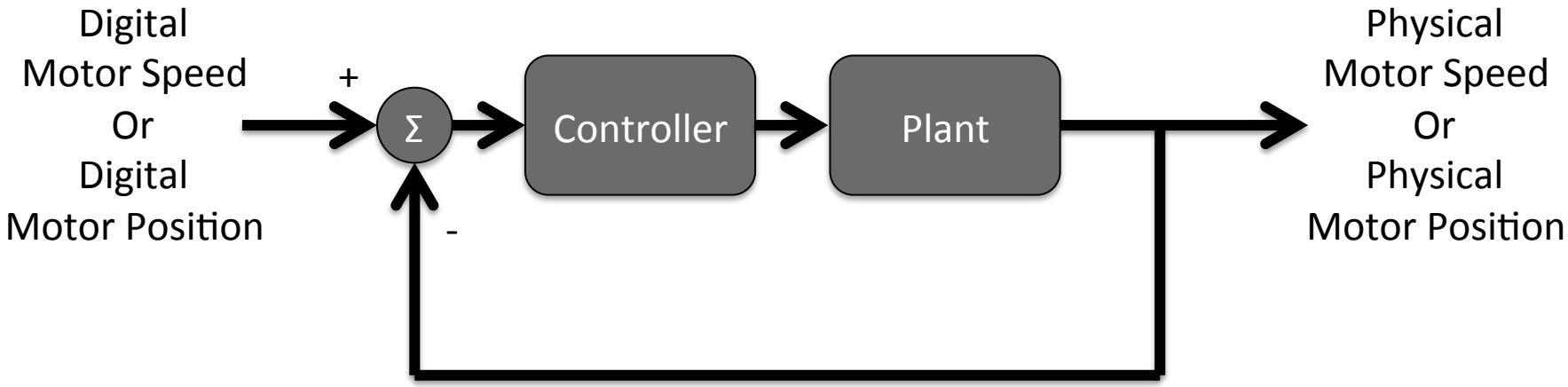
P = 30, I = 70, D = 1



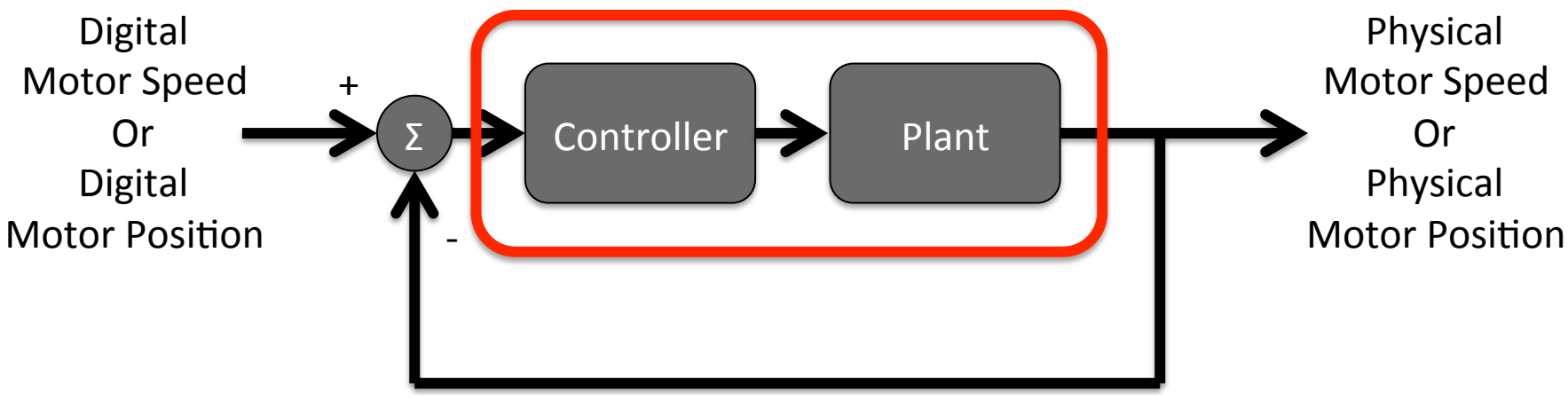
“The Right Way”

(beyond our scope)

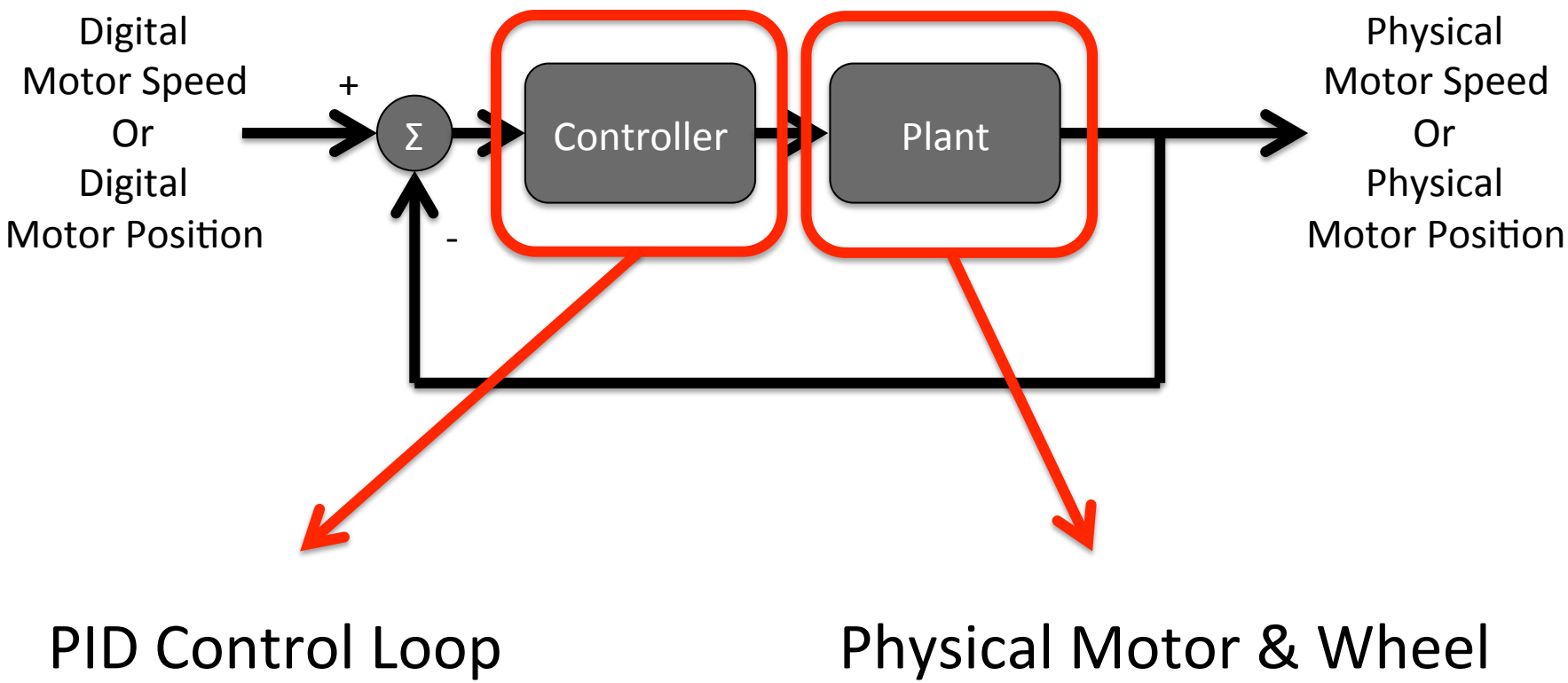
Model


Digital
Motor Speed
Or
Digital
Motor Position
+
Σ
-
Controller
Plant
Physical
Motor Speed
Or
Physical
Motor Position

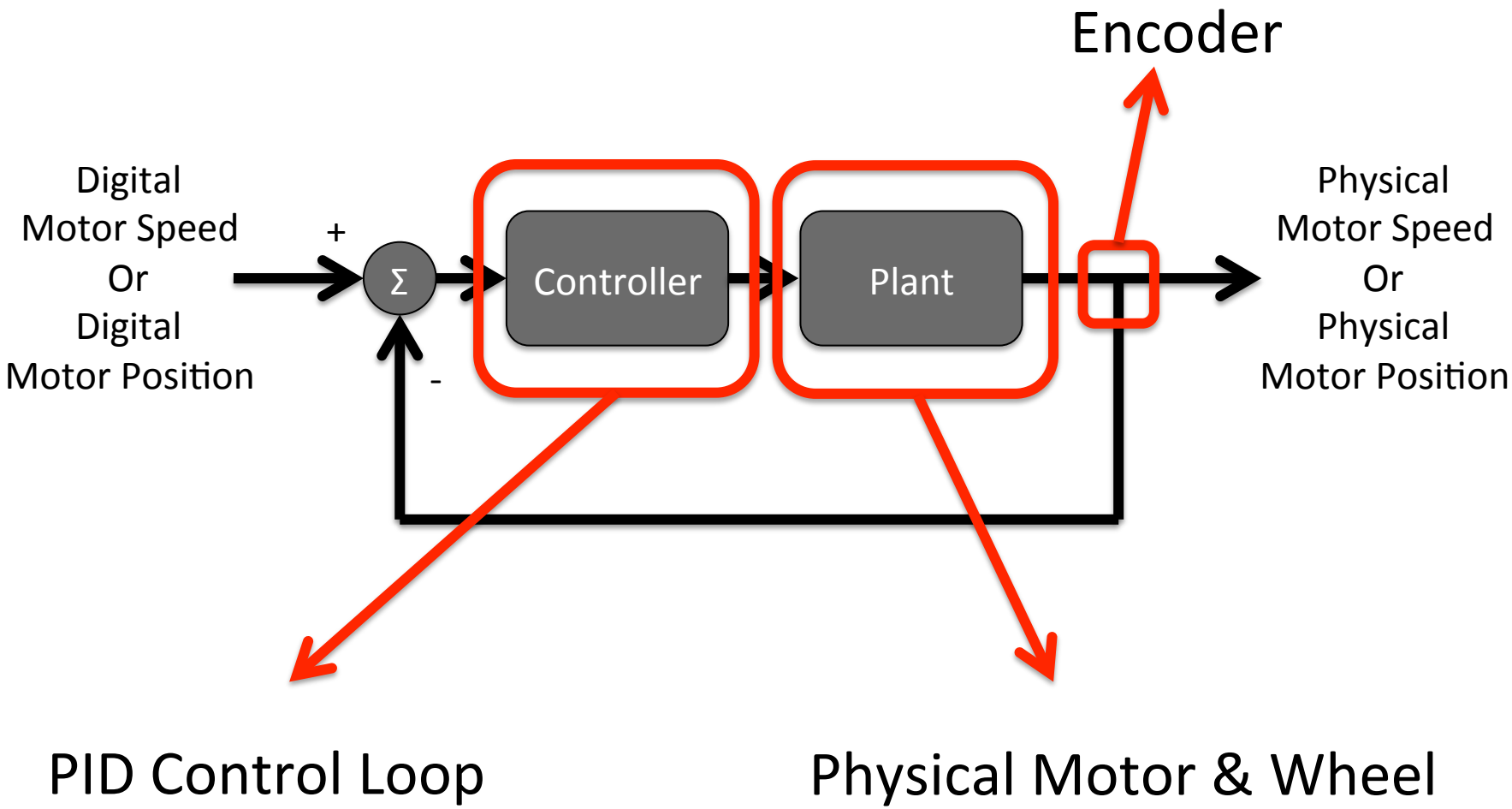
Model


Digital
Motor Speed
Or
Digital
Motor Position
+
-
Σ
Controller
Plant
Physical
Motor Speed
Or
Physical
Motor Position

Model

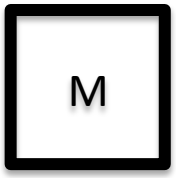

Digital
Motor Speed
Or
Digital
Motor Position
+
Σ
-
Controller
Plant
Physical
Motor Speed
Or
Physical
Motor Position
PID Control Loop
Physical Motor & Wheel

Model


Encoder
Digital
Motor Speed
Or
Digital
Motor Position
+
Σ
-
Controller
Plant
Physical
Motor Speed
Or
Physical
Motor Position
PID Control Loop
Physical Motor & Wheel

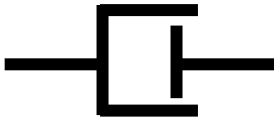
Time and S Domain

- Mass:



M*s^2

- Damper:

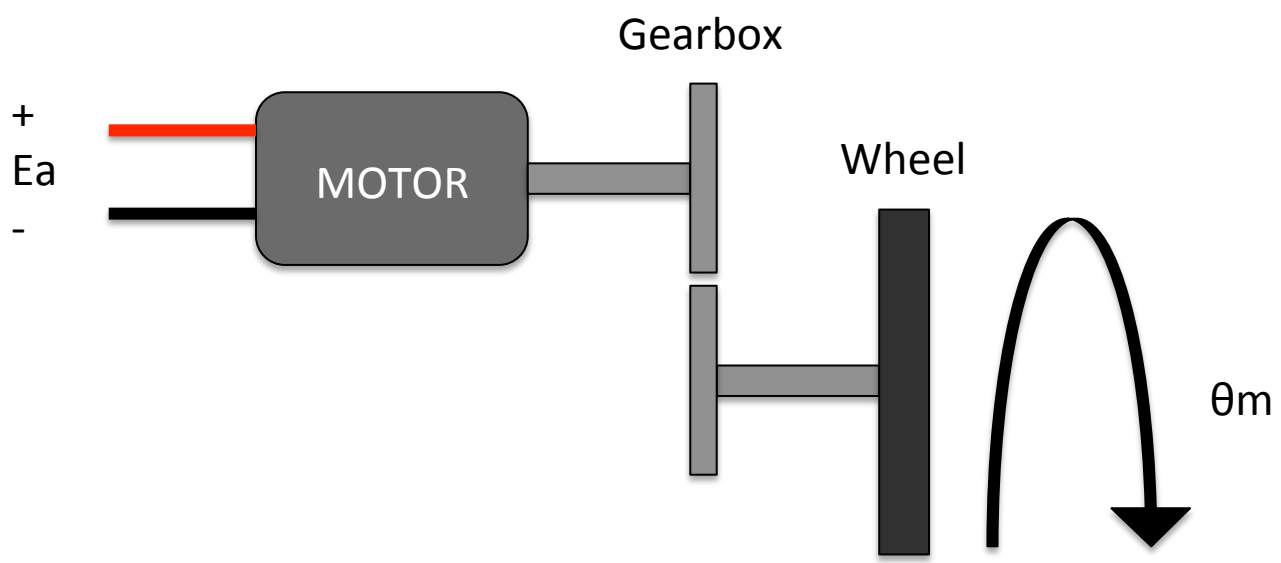


D*s

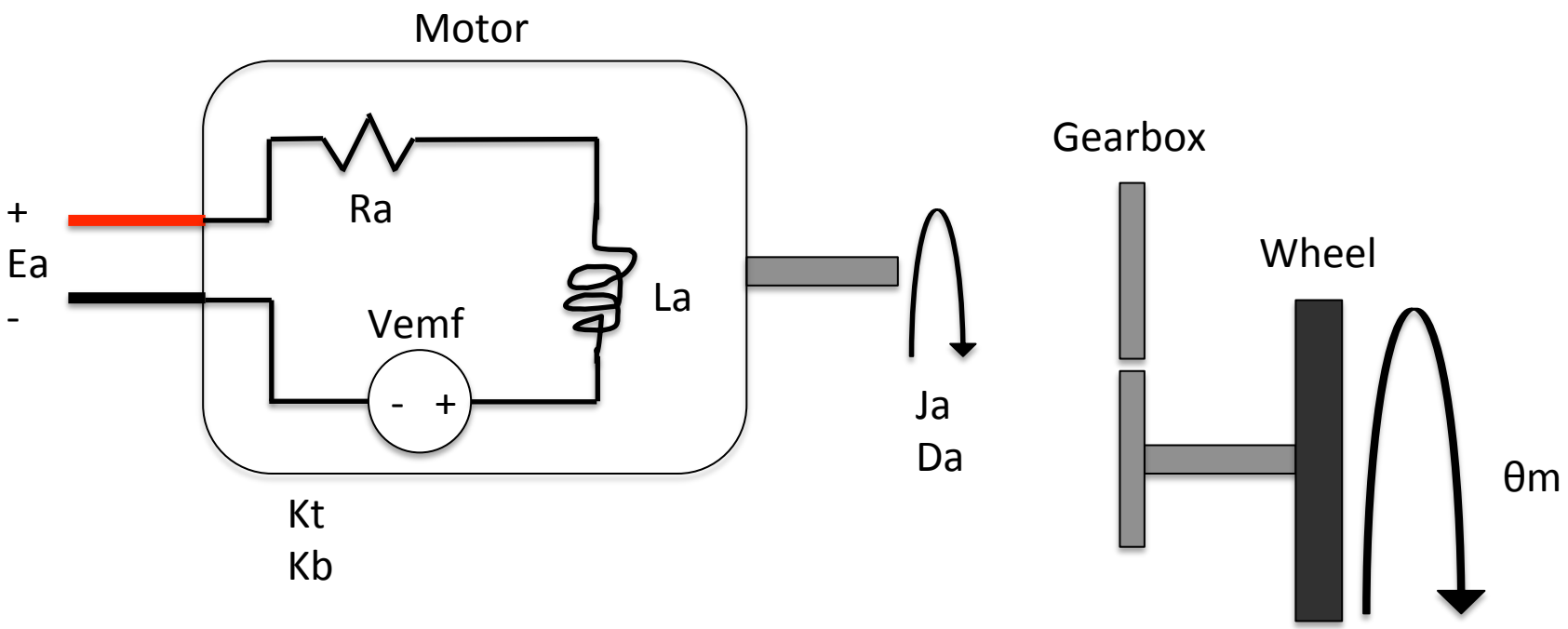
- Spring:

K

Physical Motor & Wheel



Physical Motor & Wheel



Spec'ing Motors

https://www.pololu.com/product/2214

Pololu Robotics & Electronics

DOTUPLINK@GMAIL.COM | My Account | Log Out | Wish Lists | BIG Order Form | Shopping Cart

US toll free: 1-877-7-POLOLU ~ (702) 262-6648
Same-day shipping, worldwide

Catalog | Forum | Blog | Support | Ordering | Distributors | About | Contact

Search

Feedback

Comments or questions? (opens in new window)

Services

- Custom Laser Cutting
- SMT Stencils

Products

New Products
Specials!

Robot Kits
- Zumo Robots and Accessories
- 3pi Robot and Accessories
- Tamiya Robot Kits
- Robot Kits with Soldering
- Robot Kits without Soldering
- Chassis

Electronics
- Programmable Controllers
- Motion Control Modules
- Sensors
- Regulators and Power Supplies
- Cables and Wire
- Connectors
- Electronics Prototyping
- Signal Adapters and

Pololu Metal Gearmotors » Micro Metal Gearmotors » Micro Metal Gearmotors with Extended Motor Shafts »

100:1 Micro Metal Gearmotor HP with Extended Motor Shaft

Pololu item #: 2214 **201** in stock

Price break	Unit price (US$)
1	**16.95**
10	14.41
50	12.71

Quantity: 1 Add to cart

backorders allowed Add to wish list

This gearmotor is a miniature **high-power** brushed DC motor with a **100.37:1** metal gearbox. It has a cross section of 10 × 12 mm, and the D-shaped gearbox output shaft is 9 mm long and 3 mm in diameter. This version also has a 4.5 × 1 mm **extended motor shaft**.

Key specs at 6 V: 320 RPM and 120 mA with no load, 30 oz-in (2.2 kg-cm) and 1.6 A at stall.

Select options: 100.37 :1 | 1.6A stall @ 6V (HP) | extended motor shaft? Y | Go ▶

Compare all products in Micro Metal Gearmotors with Extended Motor Shafts or High-Power (HP) Micro Metal Gearmotors.

Description | Specs (10) | Pictures (22) | Resources (3) | FAQs (1) | On the blog (1)

Overview

These tiny brushed DC gearmotors are intended for use at 6 V, though in general, these kinds of motors can run at voltages above and below this

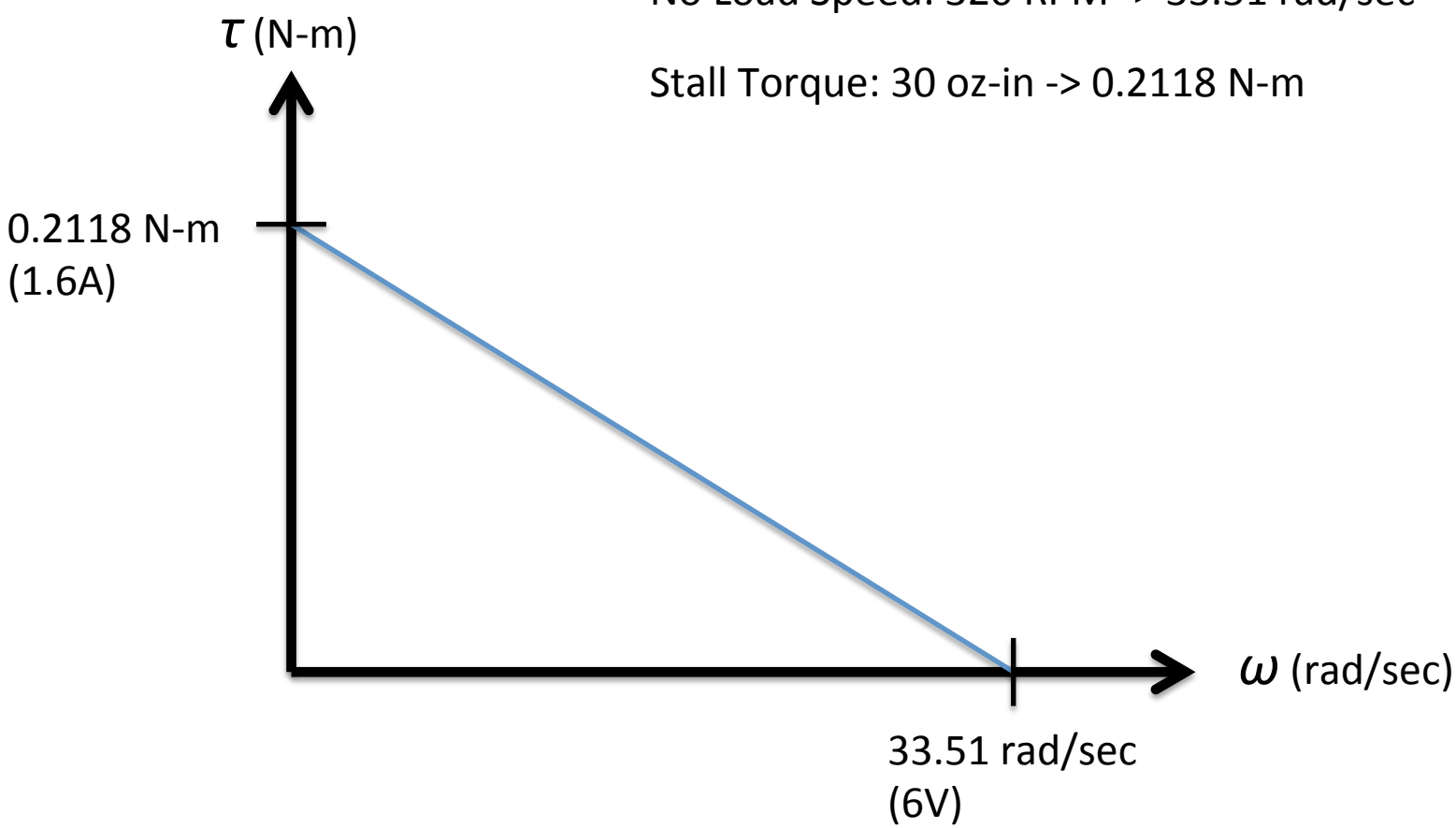
Related Products

- Magnetic Encoder Pair Kit for Micro Metal Gearmotors, 12 CPR, 2.7-18V
- Optical Encoder Pair Kit for Micro Metal Gearmotors, 5V
- Pololu 42×19mm Wheel and Encoder Set
- Pololu Micro Metal Gearmotor Bracket Extended Pair
- Pololu Micro Metal Gearmotor Bracket Pair - Black
- Pololu 15.5D mm Metal Gearmotor Bracket Pair
- Pololu Micro Metal Gearmotor Bracket Pair - White
- Machine Screw: M1.6, 3mm Length, Phillips (6-pack)
- Pololu Universal Aluminum Mounting Hub for 3mm Shaft, M3 Holes (2-Pack)
- Pololu Universal Aluminum

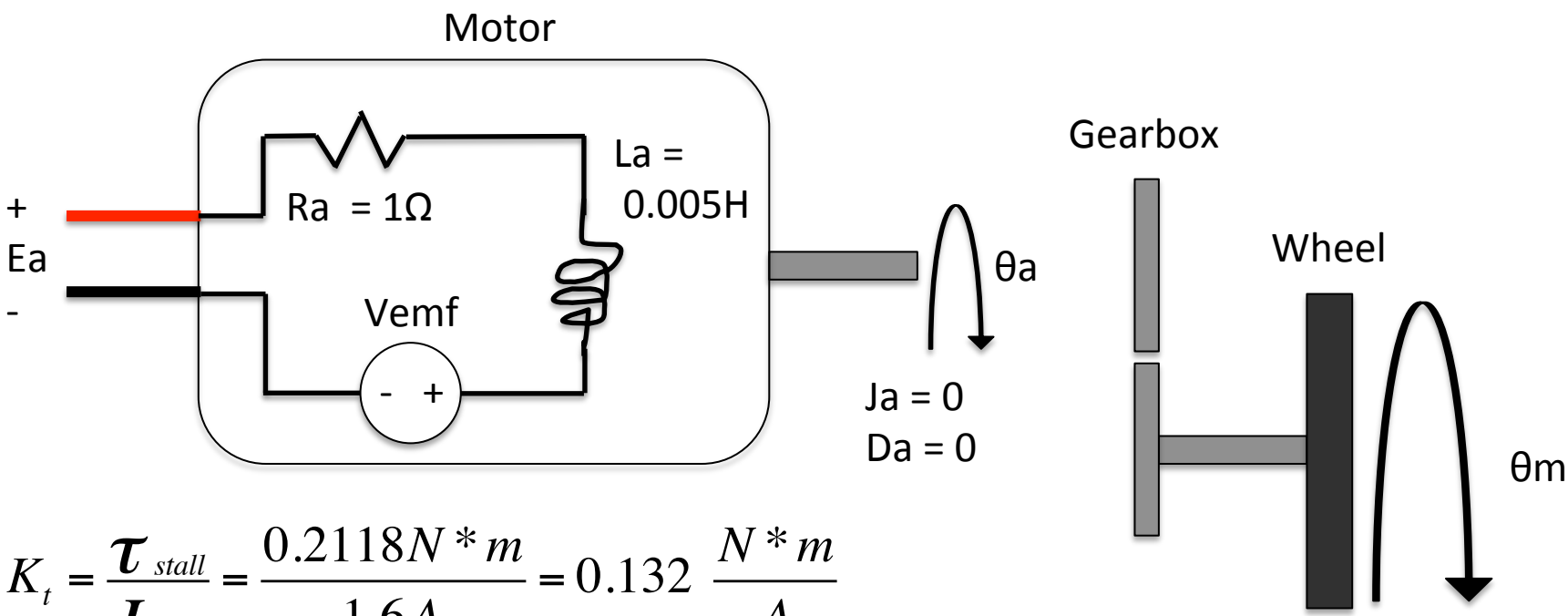
Spec'ing Motors

No Load Speed: 320 RPM -> 33.51 rad/sec

Stall Torque: 30 oz-in -> 0.2118 N-m



Physical Motor & Wheel



$$K_t = \frac{\tau_{stall}}{I_{stall}} = \frac{0.2118 N*m}{1.6A} = 0.132 \frac{N*m}{A}$$

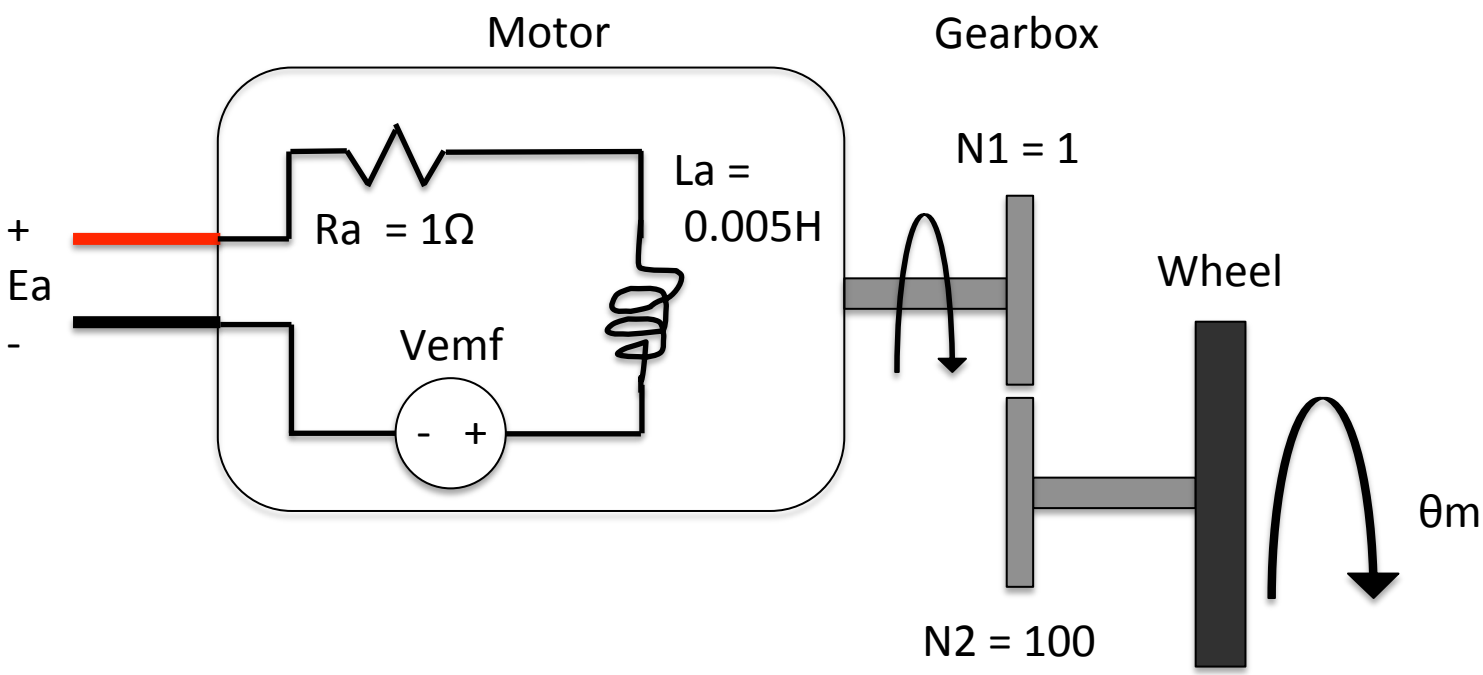
$$K_b = \frac{V_{noload}}{\omega_{noload}} = \frac{6.0V}{33.51 \frac{rad}{\sec}} = 0.179 \frac{V}{\frac{rad}{\sec}}$$

$$J_l = M_{wheel} * (r_{wheel})^2$$

$$J_l = 7.98 * 10^{-7} kg * m^2$$

$$D_l = 3.50 * 10^{-6} N * m * s / rad$$

Physical Motor & Wheel



$$K_t = \frac{\tau_{stall}}{I_{stall}} = \frac{0.2118 N*m}{1.6A} = 0.132\ \frac{N*m}{A}$$

$$K_b = \frac{V_{noload}}{\omega_{noload}} = \frac{6.0V}{33.51\frac{rad}{\sec}} = 0.179\frac{V}{\frac{rad}{\sec}}$$

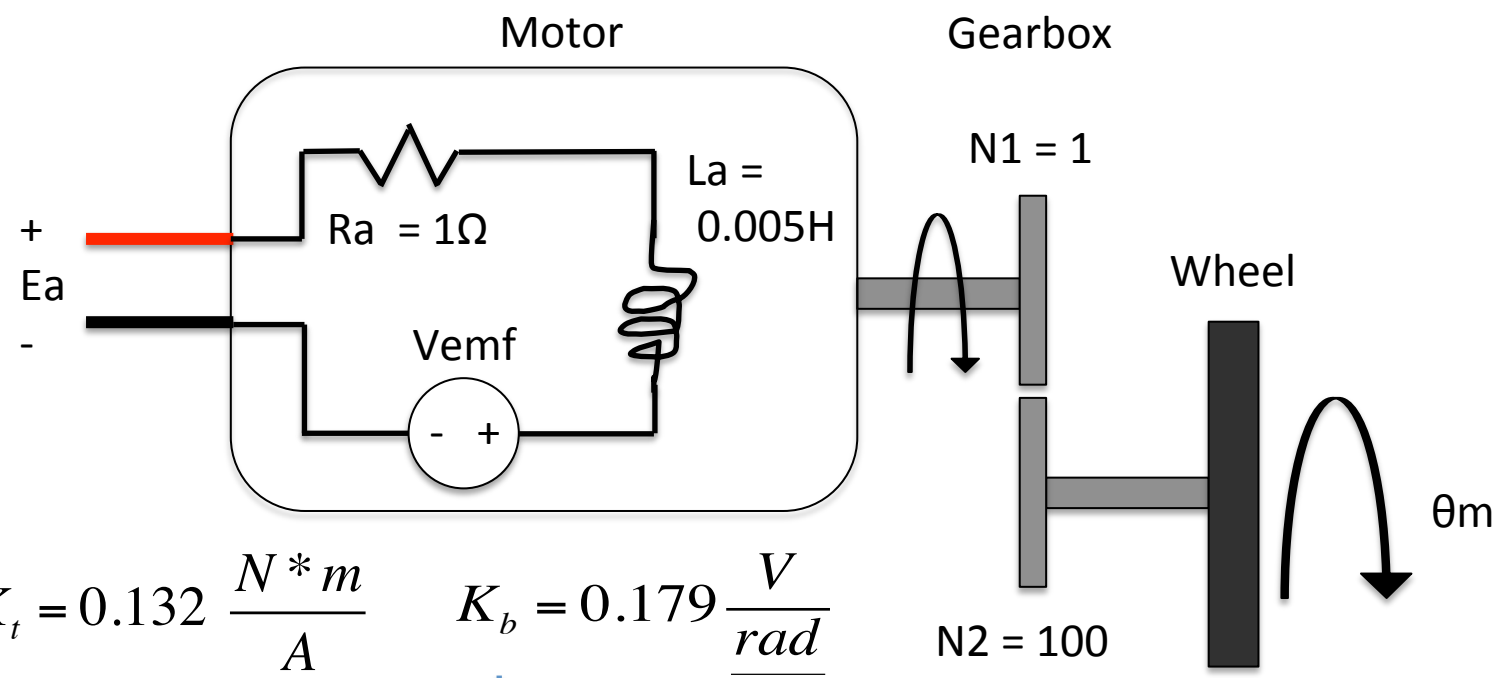
$$J_m = \frac{N_1}{N_2} J_l = \frac{1}{100} 7.98 * 10^{-7} kg * m^2$$

$$J_m = 7.98 * 10^{-9} kg * m^2$$

$$D_m = \frac{N_1}{N_2} D_l = \frac{1}{100} 3.50 * 10^{-6}\ N * m * s/rad$$

$$D_m = 3.50 * 10^{-8}\ N * m * s/rad$$

Physical Motor & Wheel



$$K_t = 0.132\ \frac{N*m}{A}$$

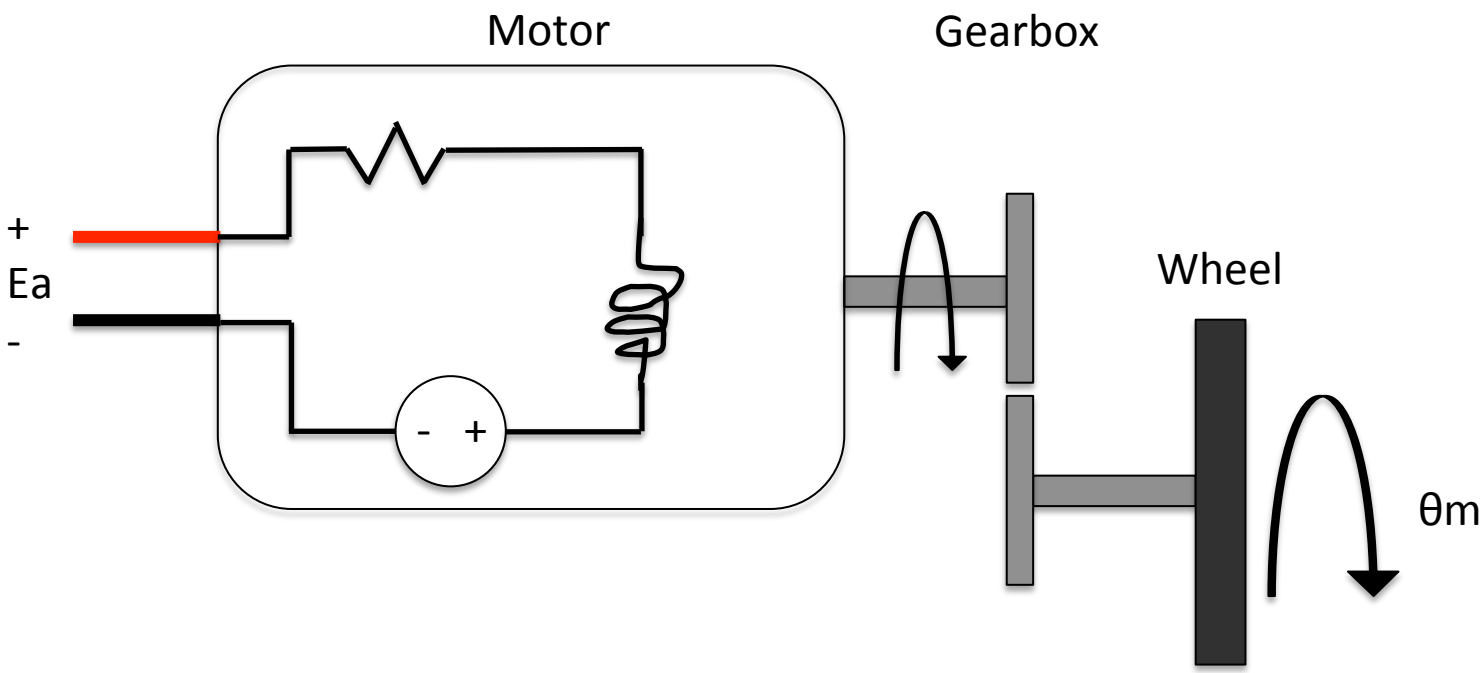
$$K_b = 0.179\ \frac{V}{\frac{rad}{\sec}}$$

$$J_m = 7.98*10^{-9}\,kg*m^2$$

$$D_m = 3.50*10^{-8}\,N*m*s/rad$$

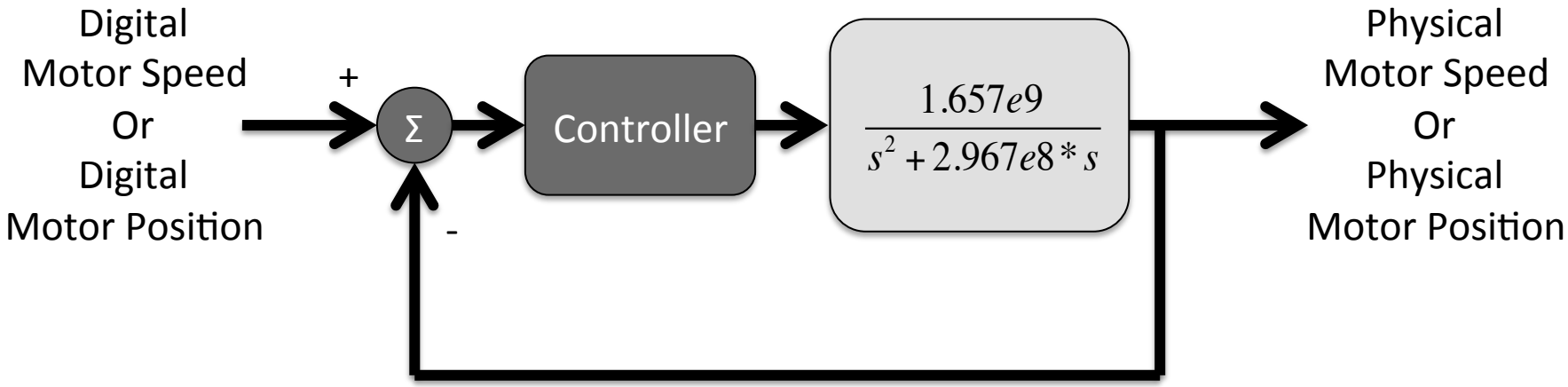
$$P(s) = \frac{\theta_m(s)}{E_a(s)} = \frac{\frac{K_t}{R_a J_m}}{s*\left[s + \frac{1}{J_m}*\left(D_m + \frac{K_t K_b}{R_a}\right)\right]}$$

Physical Motor & Wheel



$$P(s) = \frac{\theta_m(s)}{E_a(s)} = \frac{1.657 * 10^9}{s^2 + 2.967 * 10^8 * s}$$

Design


Digital
Motor Speed
Or
Digital
Motor Position
+
Σ
-
Controller
$\frac{1.657e9}{s^2 + 2.967e8 * s}$
Physical
Motor Speed
Or
Physical
Motor Position

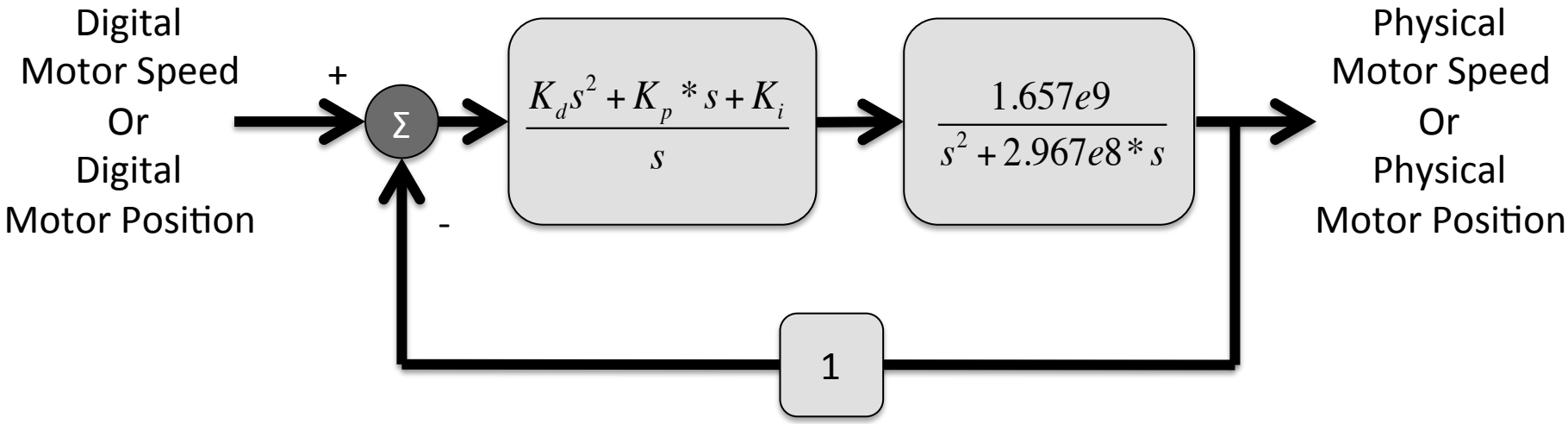
Controller

ACTION = NOW + PAST + FUTURE

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

$$C(s) = K_p + \frac{K_i}{s} + K_d * s = \frac{K_d * s^2 + K_p * s + K_i}{s}$$

Model


Digital
Motor Speed
Or
Digital
Motor Position
+
-
Σ
$\frac{K_d s^2 + K_p * s + K_i}{s}$
$\frac{1.657e9}{s^2 + 2.967e8 * s}$
Physical
Motor Speed
Or
Physical
Motor Position
1

Controller

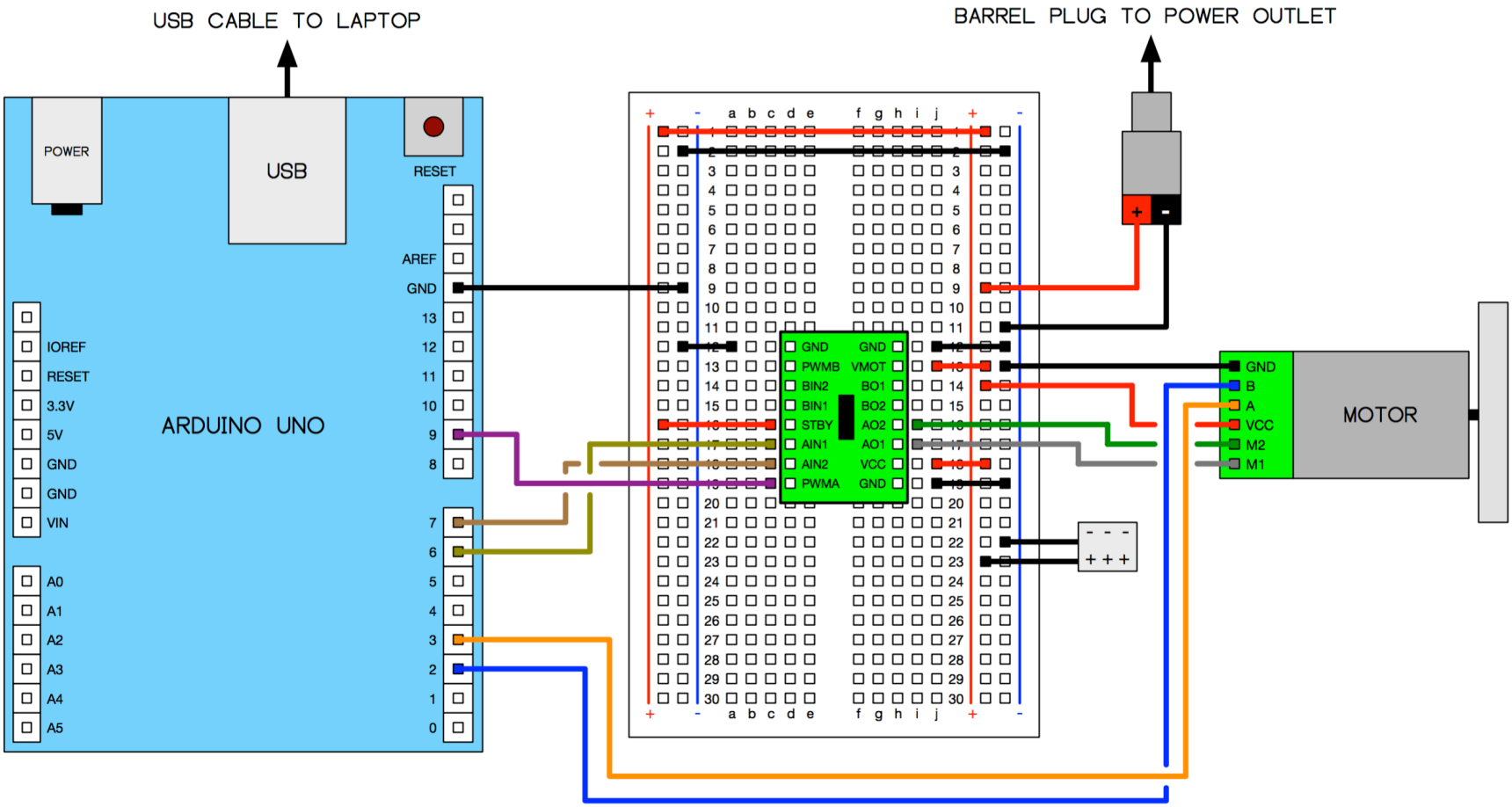
Kp

Ki

Kd

DEMO!

Hardware


USB CABLE TO LAPTOP
BARREL PLUG TO POWER OUTLET
POWER
USB
RESET
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
1
0
IOREF
RESET
3.3V
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5
ARDUINO UNO
GND
PWMB
BIN2
BIN1
STBY
AIN1
AIN2
PWMA
GND
VMOT
BO1
BO2
AO2
AO1
VCC
GND
GND
B
A
VCC
M2
M1
MOTOR

Common Mistakes

- PID loop in non timer based interrupt

```
void loop()
{                                   // Update the angle using the gyro as often as possible.
    updateAngleGyro();  // Every 20 ms (50 Hz), correct the angle using the
                                    // accelerometer, print it, and set the motor speeds.
    static byte lastCorrectionTime = 0;
    byte m = millis();

    if ((byte)(m - lastCorrectionTime) >= 20)
    {
        lastCorrectionTime = m;
        correctAngleAccel();
        printAngles();
        setMotors();    // PID Loop
    }
 }
```

Common Mistakes

- PID loop in non timer based interrupt
  - Why is this such a bad thing to do?
    - Holding up the system kills the PID controller
    - Variation in execution time means changing dt in integration and derivative terms
  - What does this mean?
    - No longer meeting timing/control authority constraints
    - Possibly unstable system

Common Mistakes

- Not recording & plotting e(t) and c(t) during the tuning process
  - You're left to visually observe systems response
  - There maybe more going on than what's visible
  - Time wasted trying different gain values without quantitatively knowing how it effects the response time
- Not recording & plotting the P, I, D terms individually
  - Not knowing how each term is contributing to c(t)
  - Hard to work back where noise/problems are coming from

Common Mistakes

- Not methodically tuning
  - Doing random P, I, D term tuning can be a HUGE time sink
  - Solution?
    - Follow a method
    - Understand the response
    - <u>Make educated changes to your gain values</u>
- Not knowing which terms you need
  - PID? PI? PD?
  - I.E. – using D term on a noisy accelerometer

Common Mistakes

- Not normalizing/mapping
  - P = 2432, I = 1693, D = 303… wat
  - Normalize/Unity Gain: 0.0 – 1.0 (assuming FPU)
- Using floating point variables
  - Dangerous to do floating point math on embedded systems without FPU