UiO : University of Oslo

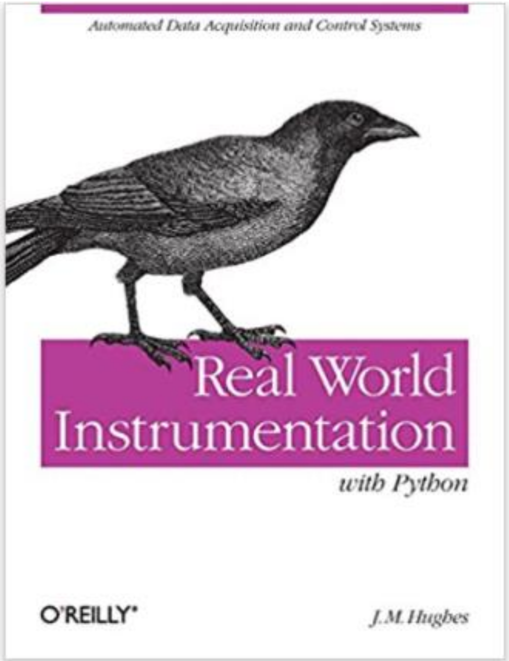
FYS3240- 4240
Data acquisition & control

Control systems

Spring 2021 – Lecture #11

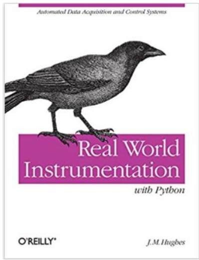
Bekkeng 10.02.2021

- Note: Some examples (Figures) from the book *Real World Instrumentation with Python: Automated Data Acquisition and Control Systems*
    - Ch. 9, page 303 – 339

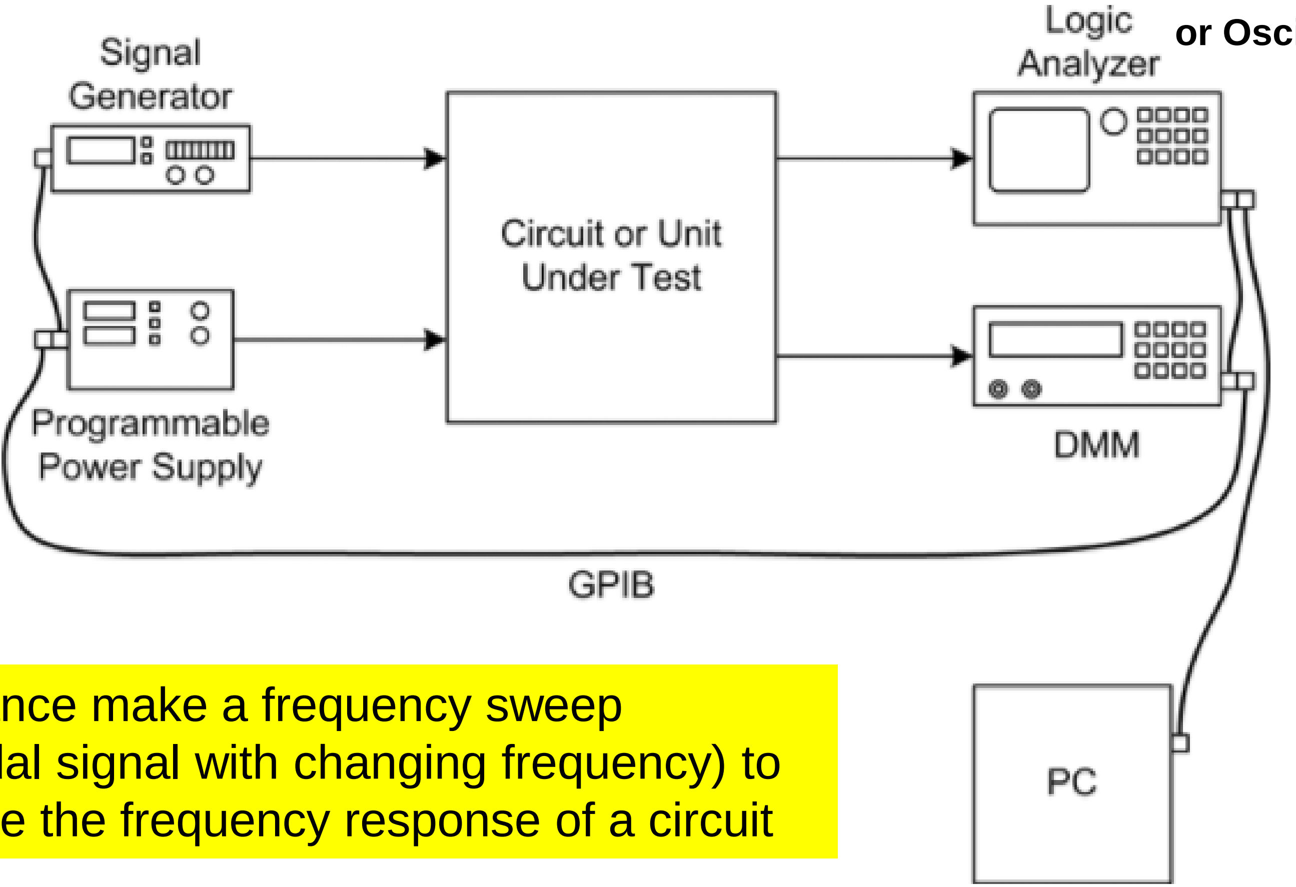

Automated Data Acquisition and Control Systems
Real World Instrumentation
with Python
O'REILLY
J.M. Hughes

Topics

- Linear vs. nonlinear control examples
- Open loop vs. closed loop control
- Discrete-time closed loop system
- PID control
- Control system examples
  - Motor control
  - Water tank
  - Satellite control
  - Missile guidance and control


Automated Data Acquisition and Control Systems
Real World Instrumentation
with Python
O'REILLY
J.M. Hughes

PC-based automated lab test setup



or Oscilloscope

For instance make a frequency sweep (sinusoidal signal with changing frequency) to determine the frequency response of a circuit

Figure from [Real World Instrumentation with Python (oreilly.com)]

Linear control systems

$$u(t) = K_p e(t) + P$$

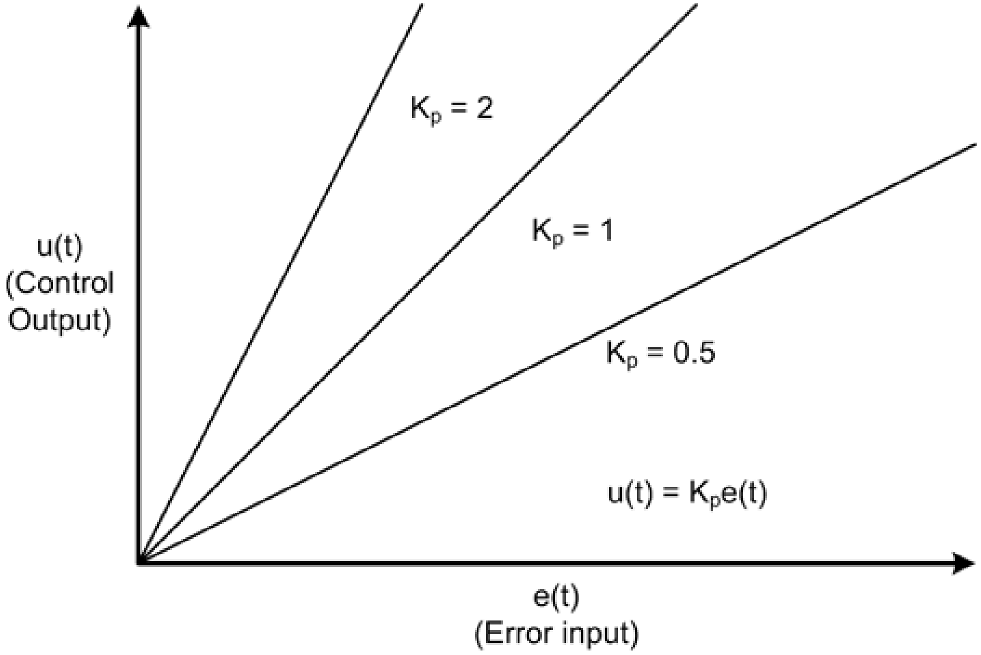
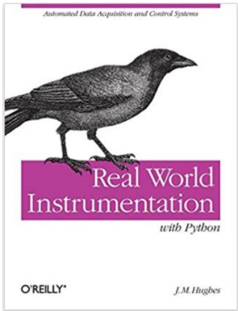


Figure 9-1. Linear control system proportional response

Figure from [Real World Instrumentation with Python (oreilly.com)]


Automated Data Acquisition and Control Systems
Real World Instrumentation
with Python
O'REILLY
J.M. Hughes

Nonlinear control systems

Example 1 (on/off controller)

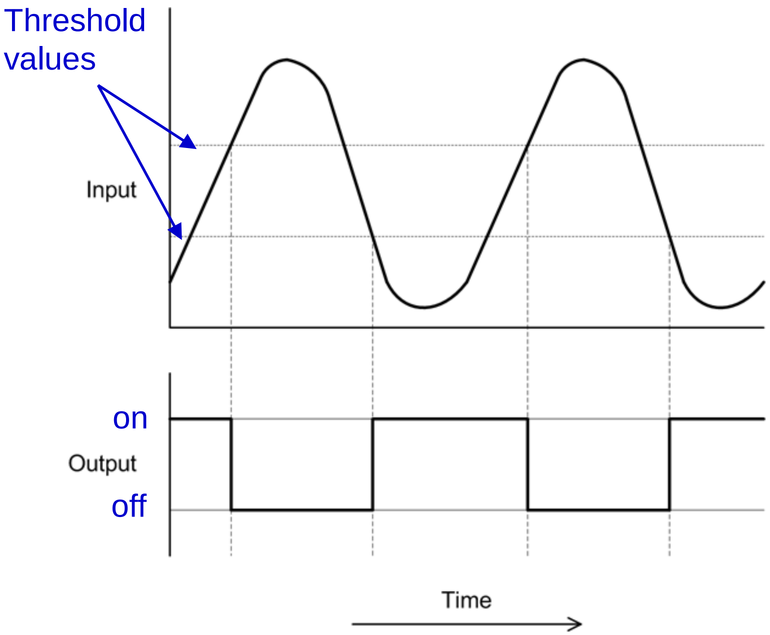


Figure 9-2. Nonlinear control system response

Example 2 (Pulse Width Modulation, PWM, controller)

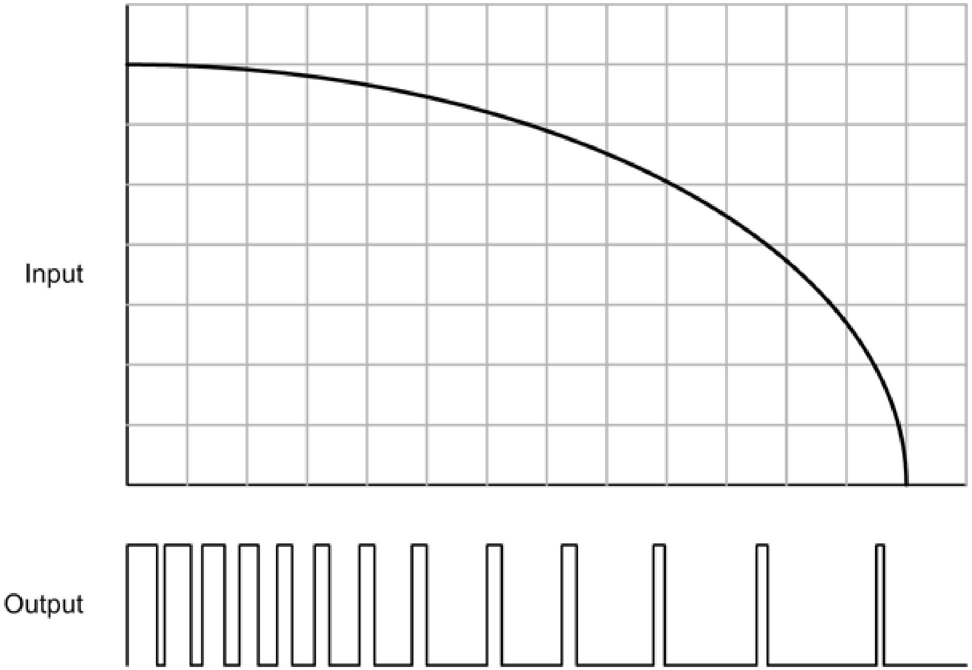
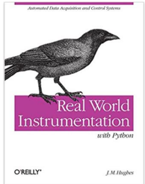


Figure 9-3. Nonlinear pulse control

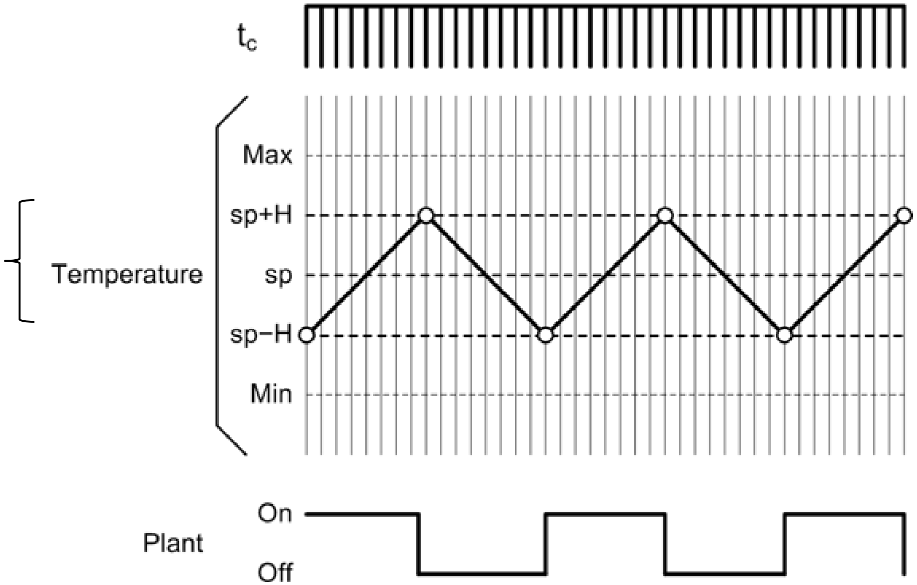
Figures from [Real World Instrumentation with Python (oreilly.com)]

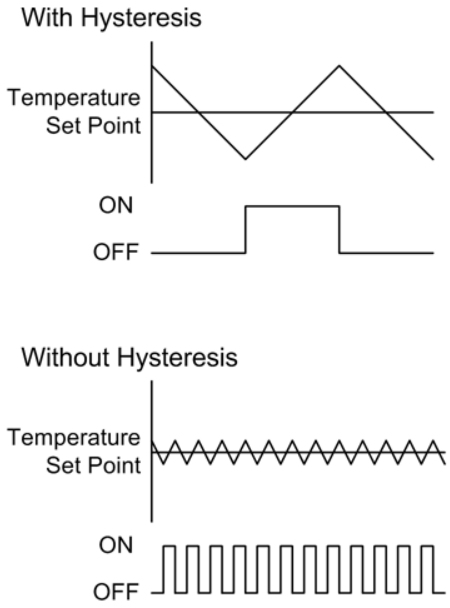


Nonlinear bang-bang (on/off) controllers

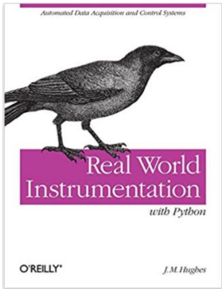
- On/off controller that switches between two states; either completely on or completely off.
    - Often used for temperature control.
    - Also used in old missiles for fin control (+/- full deflection)
- Often hysteresis is used
    - To avoid to frequent on/off switching.

Hysteresis
(low/high threshold)





Figures from Real World Instrumentation with Python (oreilly.com)


Automated Data Acquisition and Control Systems
Real World Instrumentation
with Python
O'REILLY
J.M. Hughes

Sequential control systems

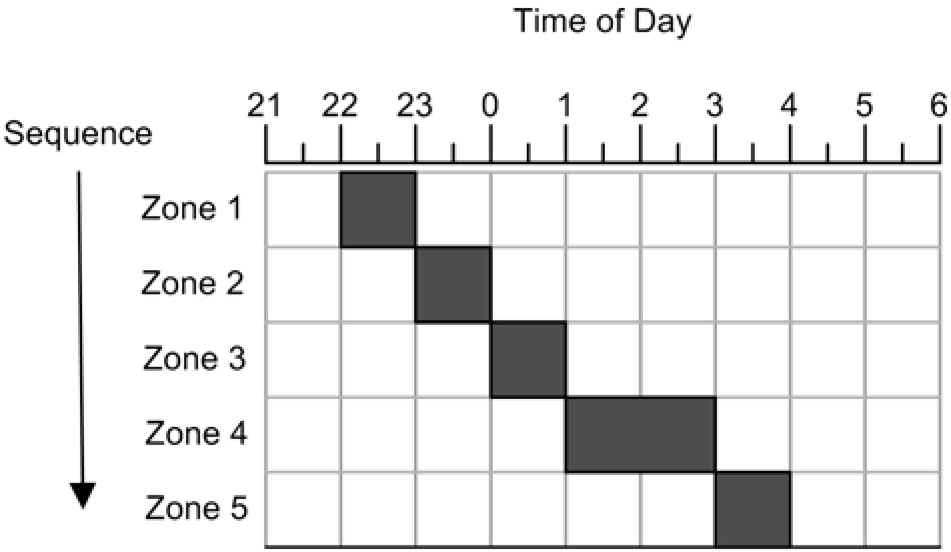


Figure 9-4. Sprinkler system sequential control

Sequential power control

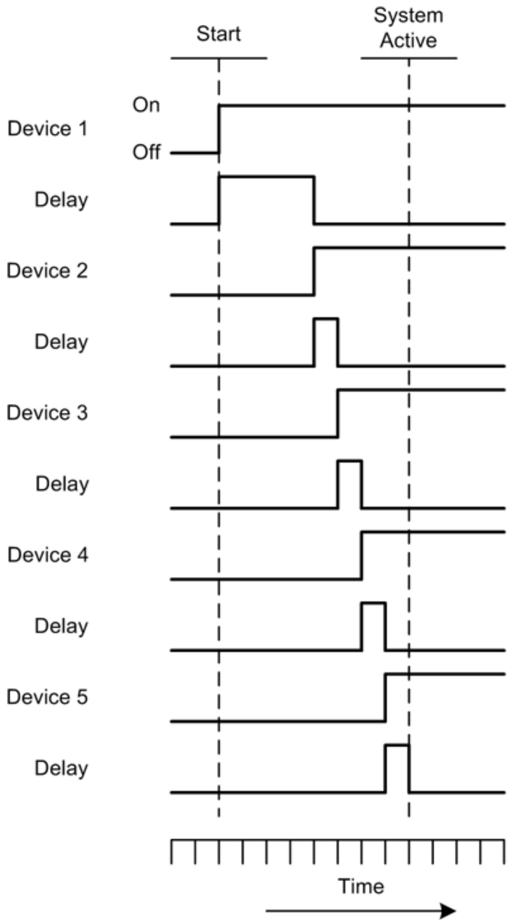
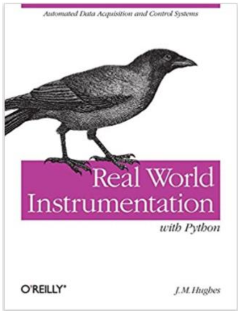


Figure 1-8. Sequential power control

Figures from Real World Instrumentation with Python (oreilly.com)

Open-loop control


Automated Data Acquisition and Control Systems
Real World Instrumentation
with Python
O'REILLY
J.M. Hughes

Open-loop control

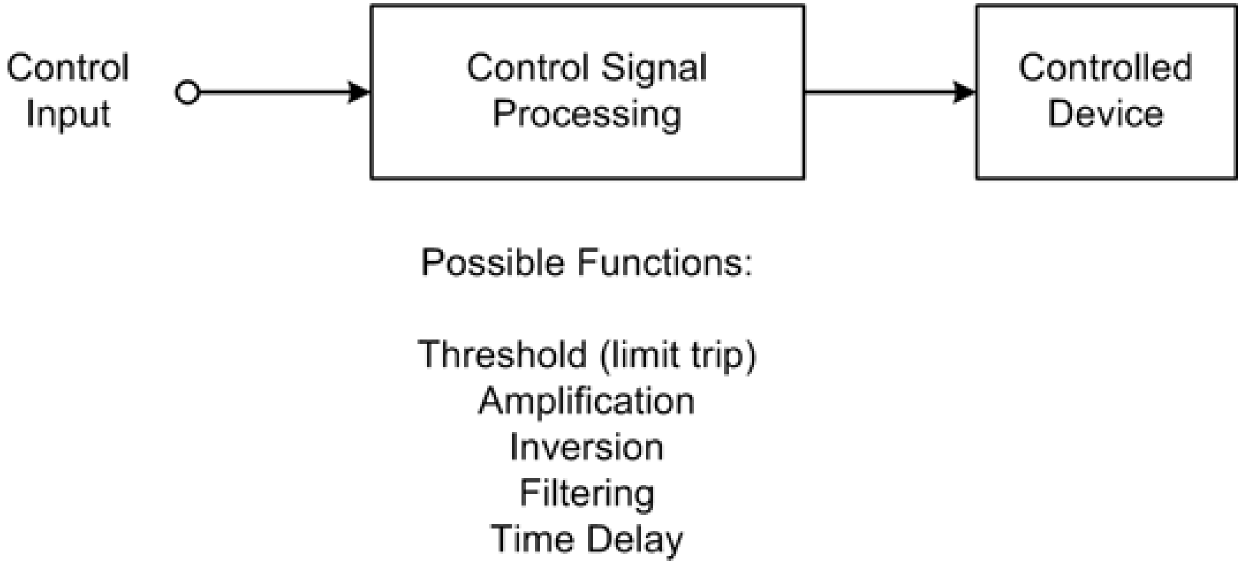
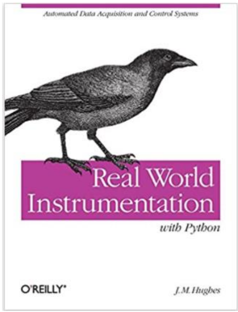
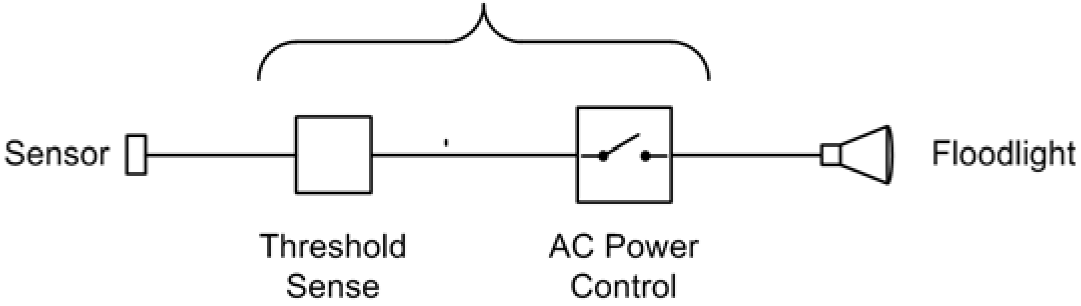


Figure from Real World Instrumentation with Python (oreilly.com)


Automated Data Acquisition and Control Systems
Real World Instrumentation with Python
O'REILLY
J.M. Hughes

Open-loop light control system

- Turn on a lamp on for a given time if a motion is detected.
- On/off control



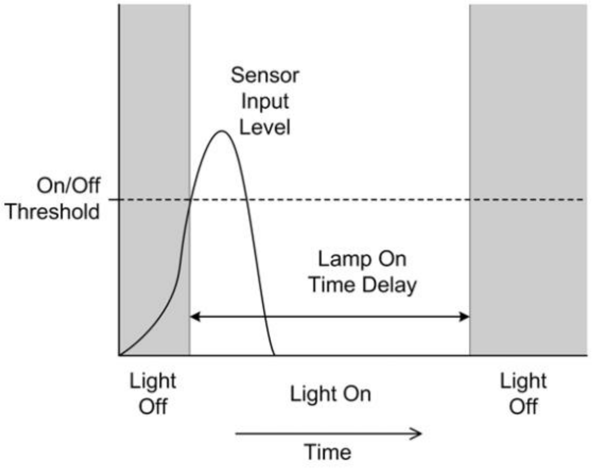
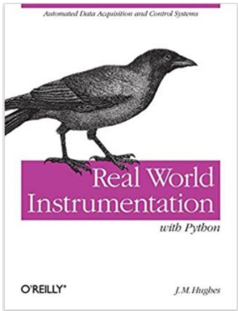


Figure from Real World Instrumentation with Python (oreilly.com)


Automated Data Acquisition and Control Systems
Real World Instrumentation
with Python
O'REILLY
J.M. Hughes

Simple open-loop motor control

- Motor rotation rate will vary with load!
  - Not a good controller!

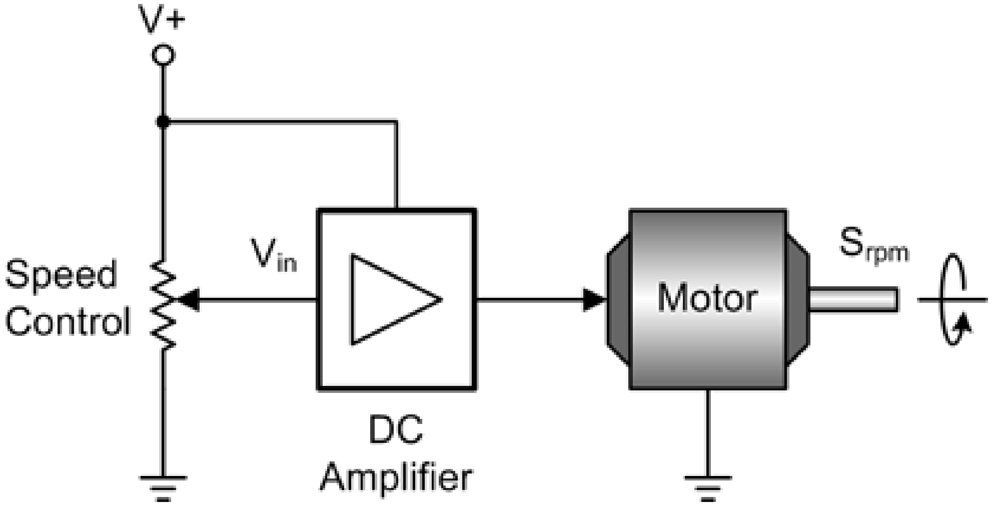
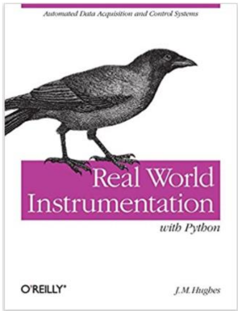


Figure 9-13. Simple open-loop DC motor control

Figure from [Real World Instrumentation with Python (oreilly.com)]

Closed-loop control


Automated Data Acquisition and Control Systems
Real World Instrumentation
with Python
O'REILLY
J.M. Hughes

Closed-loop control

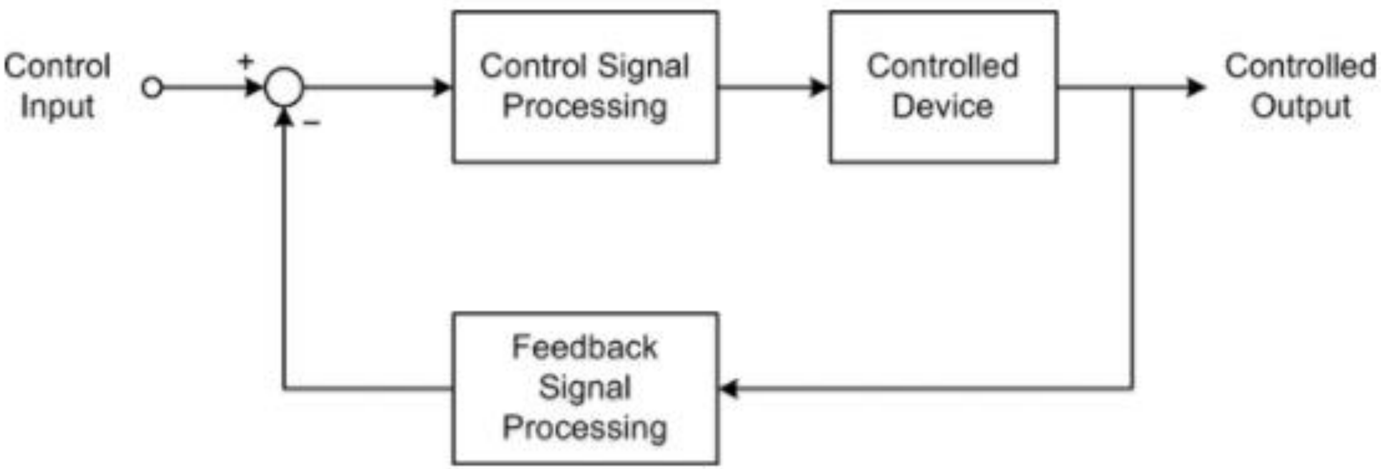
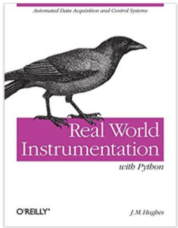


Figure 1-6. Closed-loop control

Figures from [Real World Instrumentation with Python (oreilly.com)]



PWM motor speed control with feedback

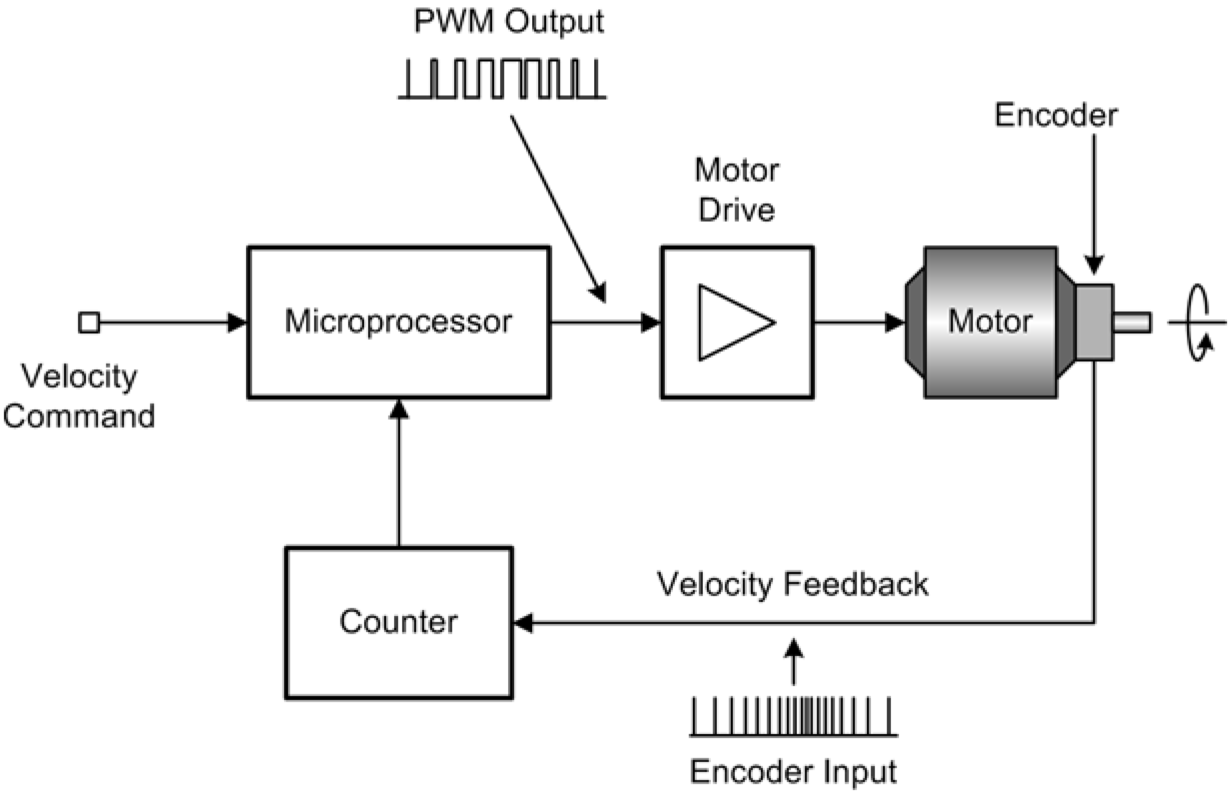
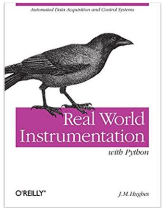
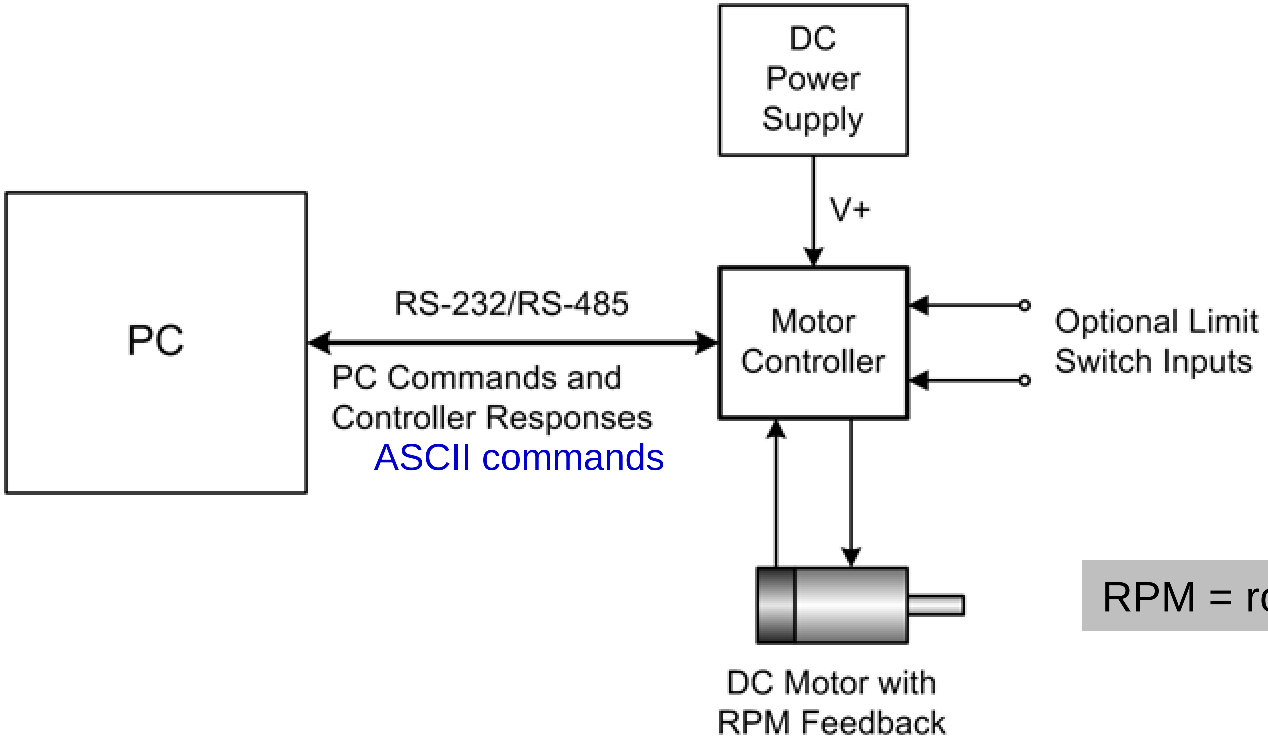


Figure from Real World Instrumentation with Python (oreilly.com)



Commercial DC motor controller with RPM feedback

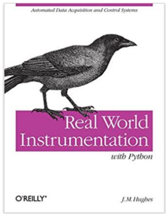


ASCII commands

RPM = rotations per minute

Figure 9-17. Commercial DC motor controller

Figure from Real World Instrumentation with Python (oreilly.com)


Automated Data Acquisition and Control Systems
Real World Instrumentation
with Python
O'REILLY
J.M. Hughes

Closed-loop water tank control system

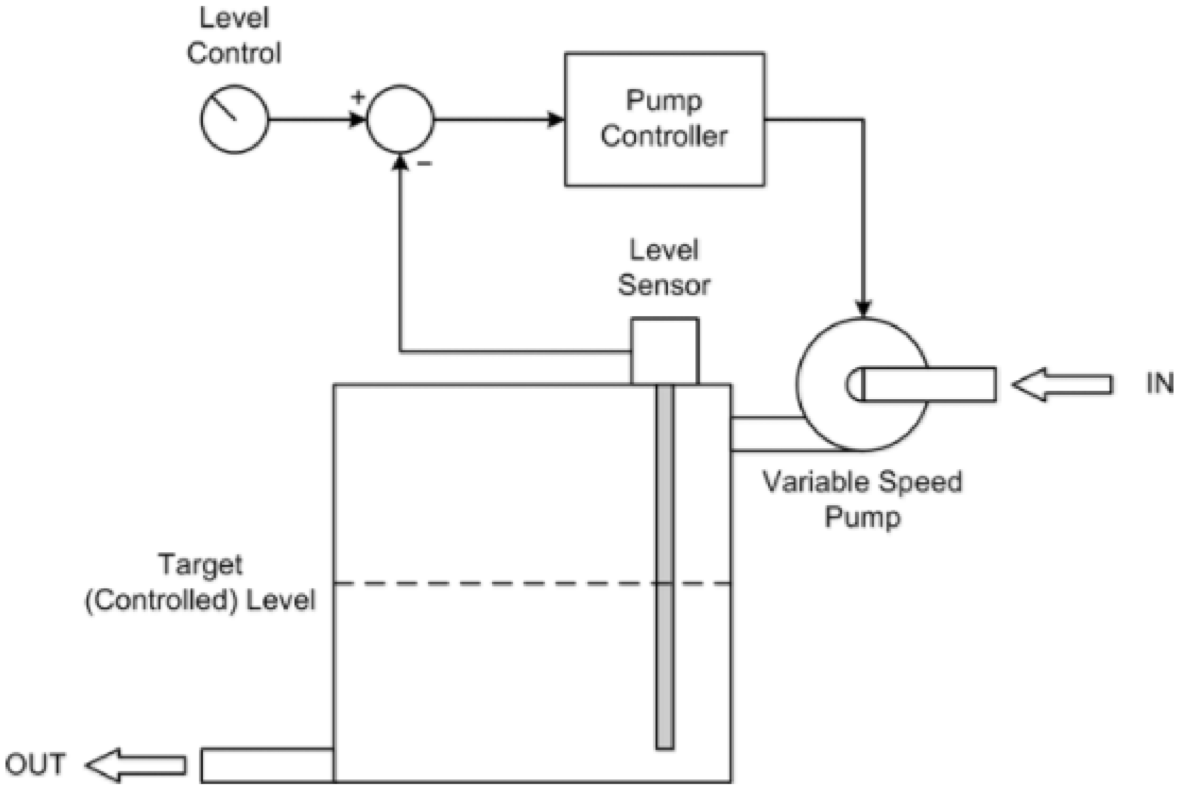
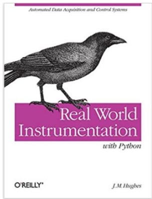
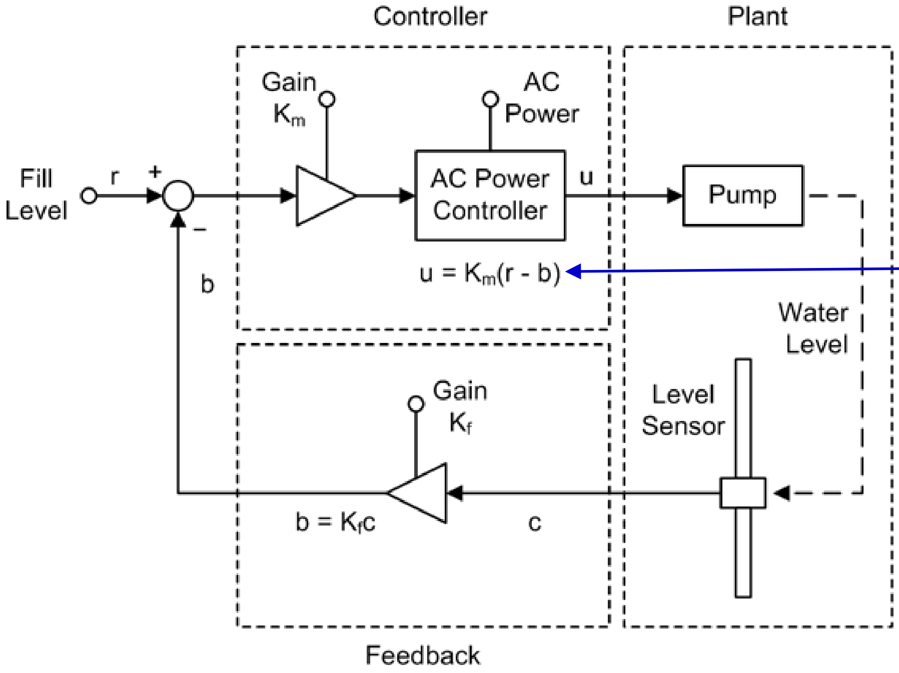


Figure from [Real World Instrumentation with Python (oreilly.com)]



Closed-loop water tank control system



Proportional control

Figure 9-11. Closed-loop water tank control system details

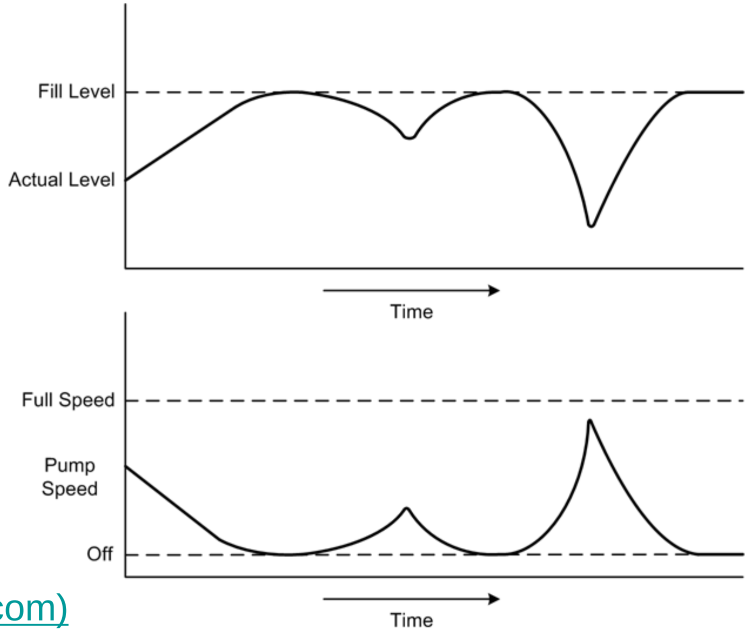
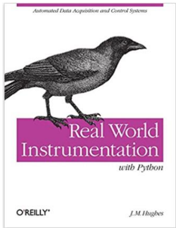


Figure 9-12. Water tank control system response graphs

Figures from [Real World Instrumentation with Python (oreilly.com)](https://www.oreilly.com)



Discrete-time closed loop system

- ADC
- DAC
- Clock
  - For synchronization

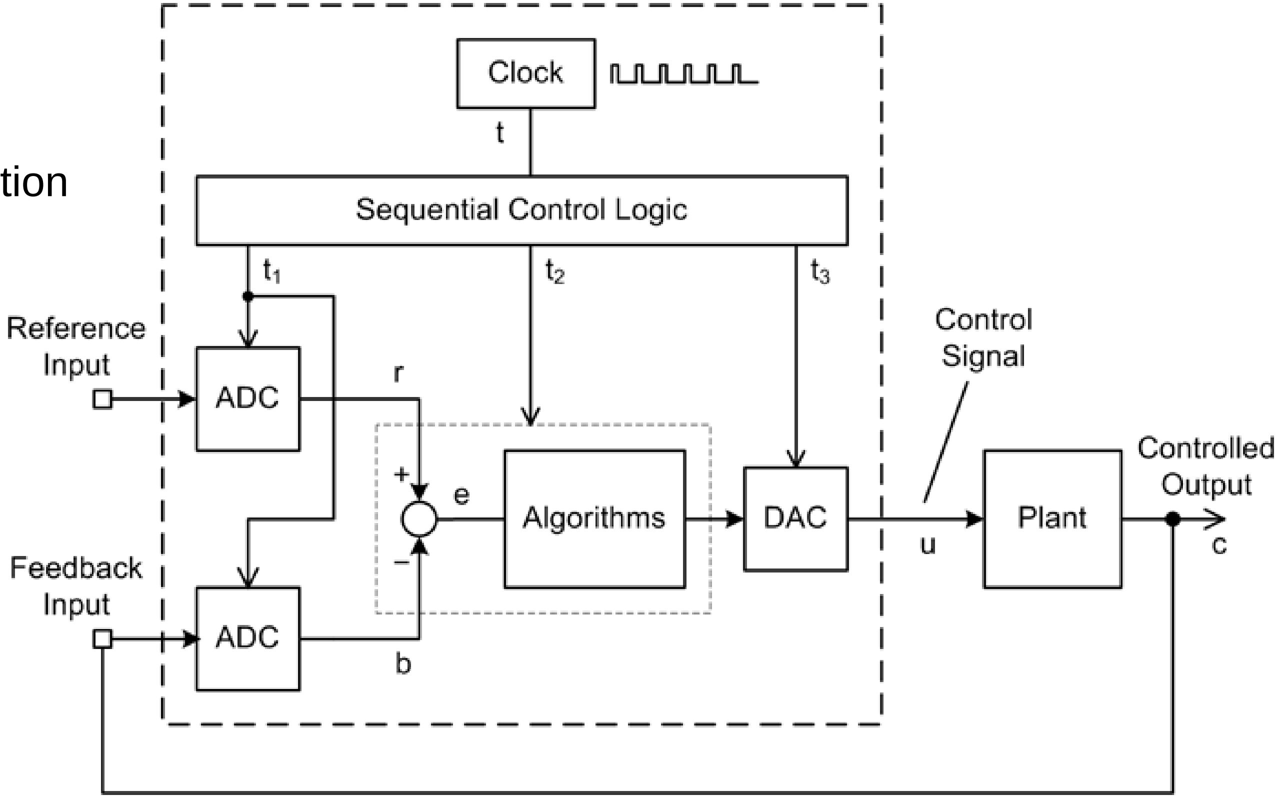
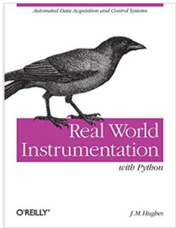
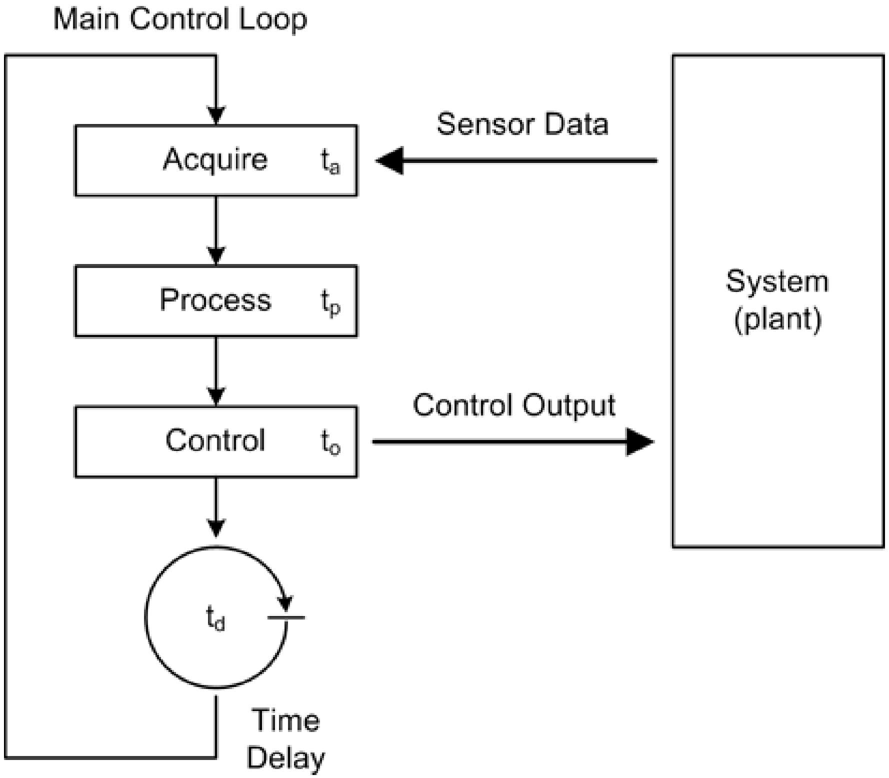


Figure from Real World Instrumentation with Python (oreilly.com)


Automated Data Acquisition and Control Systems
Real World Instrumentation with Python
O'REILLY
J.M. Hughes

Control software flow

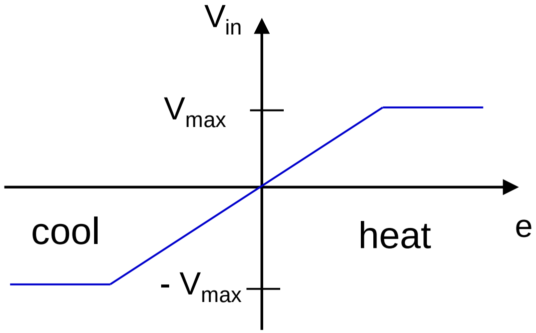


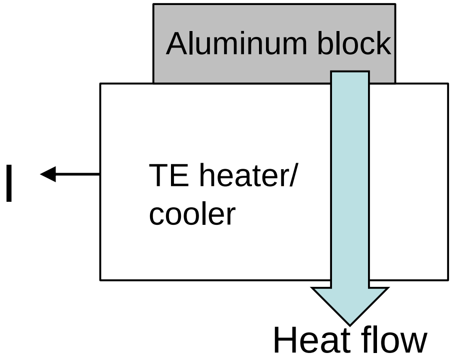
Figures from [Real World Instrumentation with Python (oreilly.com)](https://www.oreilly.com)

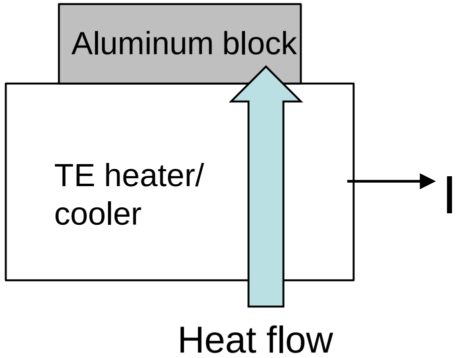
PID control

Temperature controller example

- We need to build a temperature controller that can control the temperature of an aluminum block to be close to a given set point $T_{set\ point}$ which is higher then the surrounding temperature.
- We will read the temperature of the block, T_{block}, using a temperature sensor.
- Assume a Thermoelectric (TE) heater/cooler attached to the block, such that we can heat or cool the block by changing the current (I) direction.
- A voltage V_{in} is used to control the current flow
  - Using a voltage to current converter (driver) circuit.







Proportional control of temperature

- The error *e* is defined to be: $e = T_{set\ point} - T_{block}$

- The control voltage is set to $V_{in} = K_p e$, where K_p is a constant called the proportional gain.

- Problem:
  - When e = 0 the power to the heater is turned of.
  - → The block will stabilize at a temperature below the set point, since the sample will lose heat to the surroundings.

- Solution:
  - Include a constant V_0 that can counteract the heat loss to the surroundings

$$V_{in} = K_p e + V_0$$

PI control of temperature

- Instead of trying to find the correct constant value for V_0 we want to build some intelligence into the control system.

- We want to find the proper value of V_0, given the chosen set point, automatically.
  - We use an integral to construct the correct constant during runtime.

$$V_{in} = K_p e + K_i \int_0^t e \, dt$$

- The integral keeps a running sum of all the error values (up to time t).
- K_i is a constant that need to be selected.

PID control of temperature

- A derivative term can be added to damp out oscillations

$$V_{in} = K_p e + K_i \int_0^t e \, dt + \boxed{K_d \frac{de}{dt}}$$

constants

Discrete-sampling PID control of temperature

- Sample interval of *Δ*t
- After n samples the continuous time equation can be approximated with:

$$V_{in} = K_p e_n + K_i \Delta t \underbrace{\sum_{m=0}^{n} e_m}_{\text{Sum over all error values since the control algorithm was turned on}} + \frac{K_d}{\Delta t}(e_n - e_{n-1})$$

PID controller summary

- Proportional-Integral-Derivative (PID) algorithm is the most common control algorithm
  - Used for heating and cooling systems, fluid level control, pressure control, …
- Calculates a term **proportional to the error** - the P term.
- Calculates a term **proportional to the integral of the error** - the I term.
- Calculates a term **proportional to the derivative of the error** - the D term.
- The three terms - the P, I and D terms, are added together to produce a control signal that is applied to the system being controlled.
- Sometimes a PI-controller is used.

PID controller – general terms

- A PID controller continuously calculates **an *error*** *value* as the difference between **a measured process variable** and a desired **set point**.
- The controller attempts to minimize the error *e* over time, by adjustment of a *control variable u(t)* , such as the position of a control valve.

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

- *P* accounts for present values of the error.
- I accounts for past values of the error, accumulates over time.
- *D* accounts for possible future values of the error, based on its current rate of change.
- Must tune the coefficients Kp, Ki and Kd.

In general PID does not provide *optimal* control, since no modelling of the Plant/process is used.

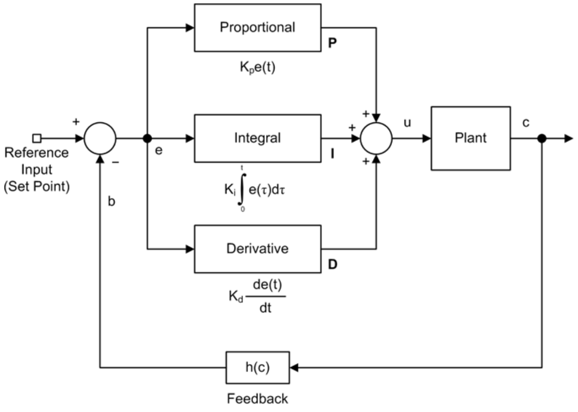
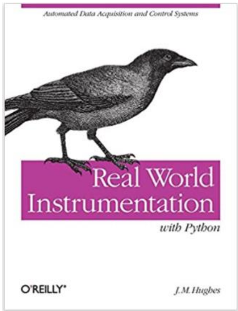


Figure 9-24. PID control block diagram


Automated Data Acquisition and Control Systems
Real World Instrumentation
with Python
O'REILLY
J.M. Hughes

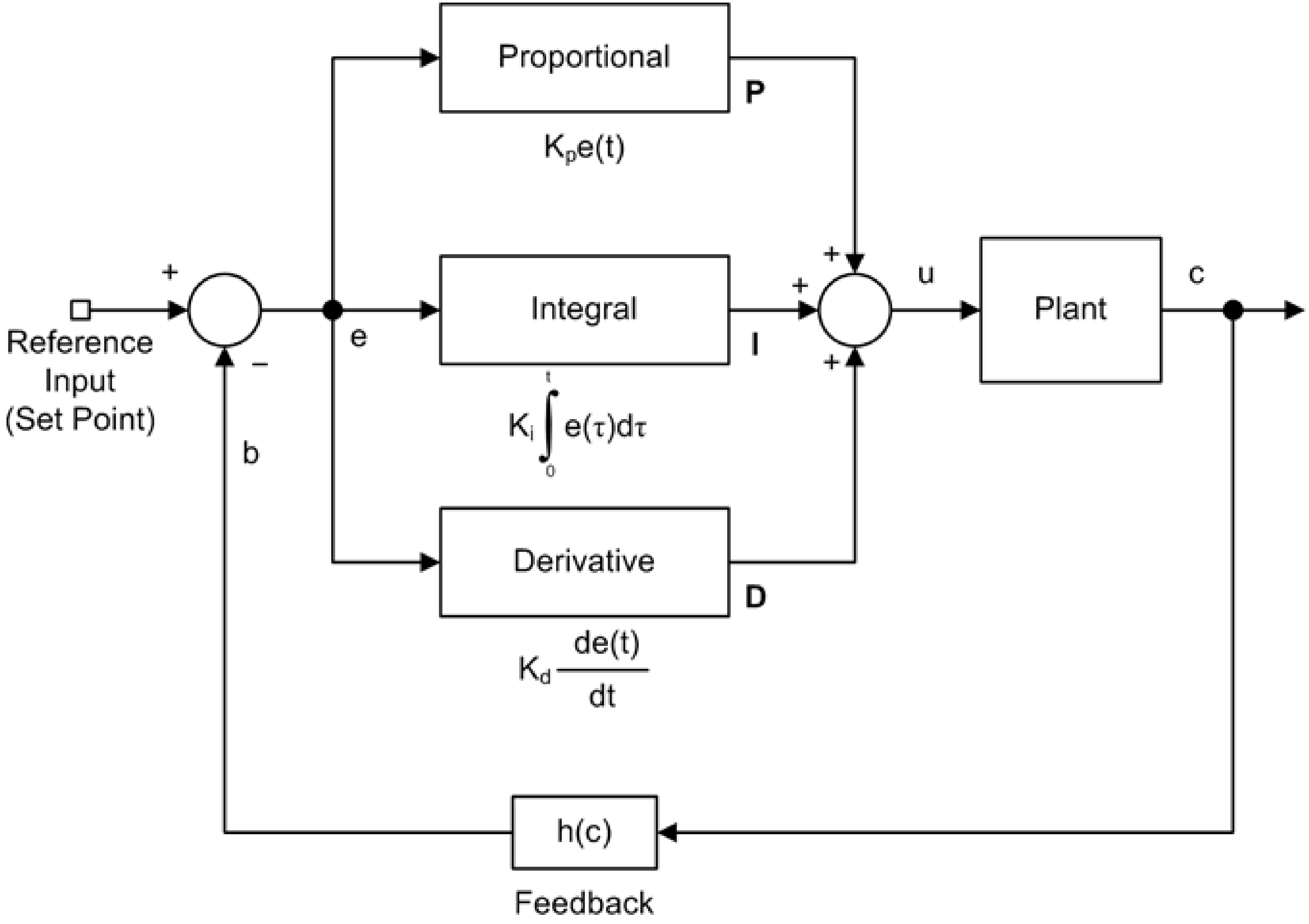
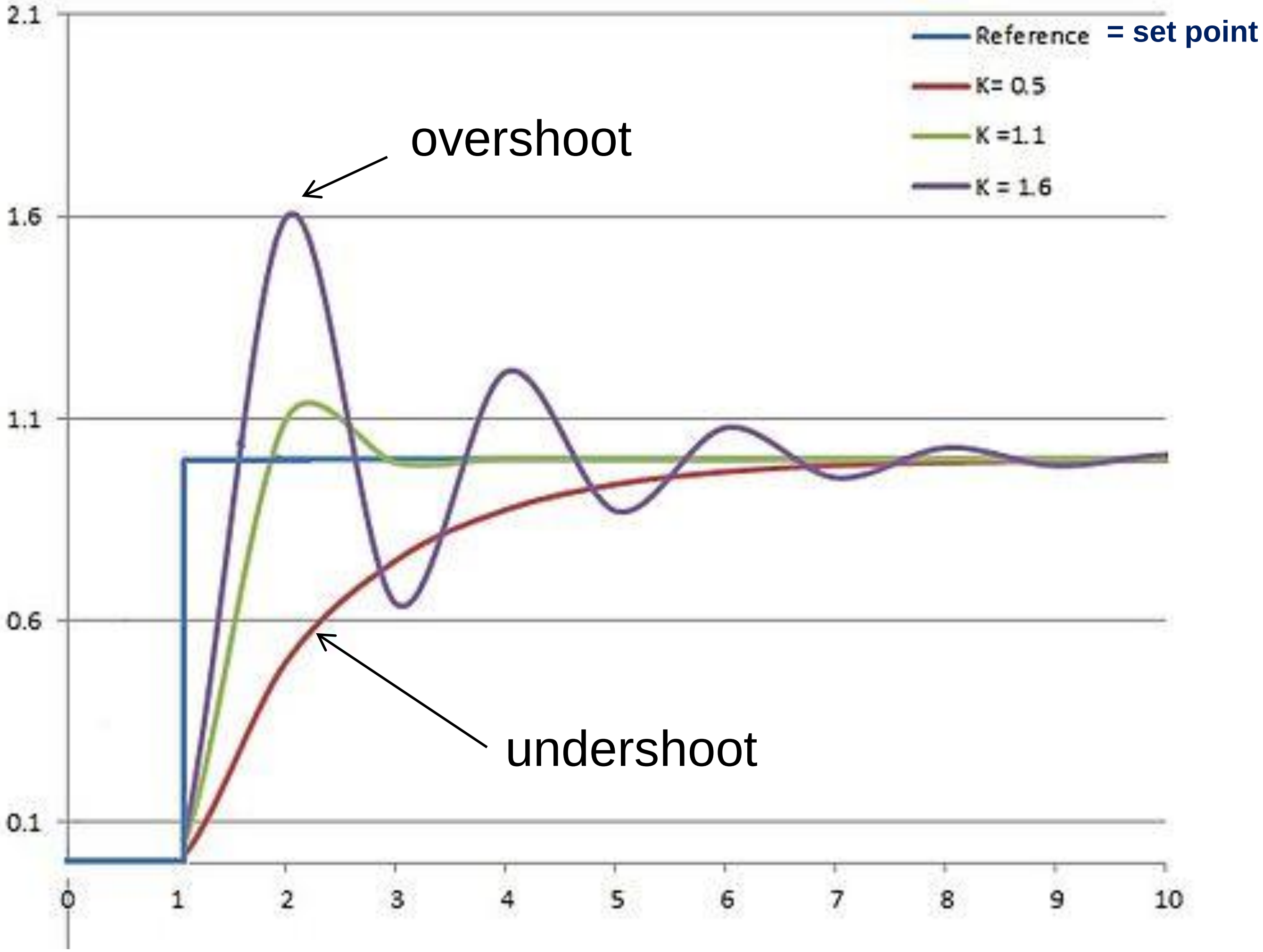


Figure 9-24. PID control block diagram

Figure from Real World Instrumentation with Python (oreilly.com)

PID controller tuning examples



= set point

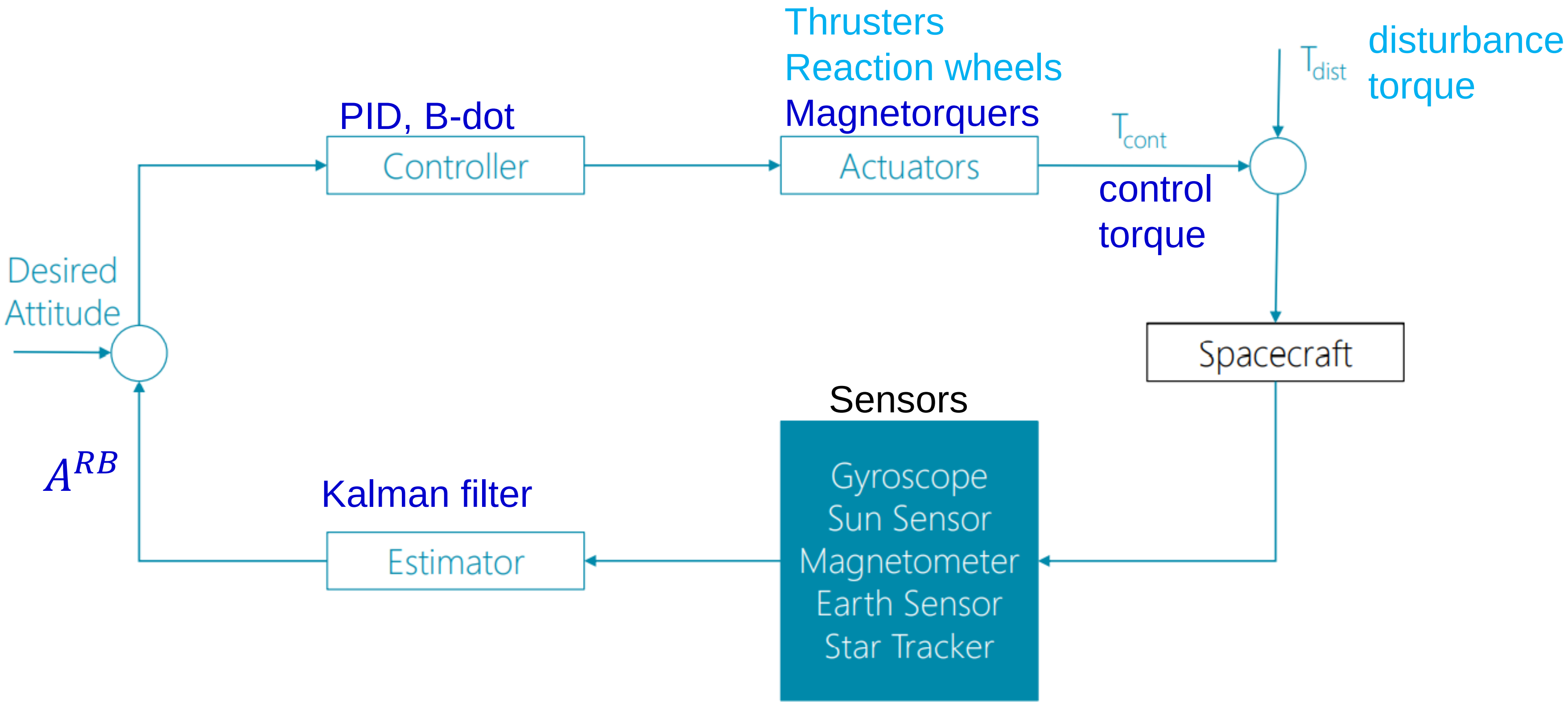
overshoot

undershoot

Satellite control

Active attitude control

- With active attitude control, we estimate the spacecraft attitude and control actuators to actively change it to a desired attitude.



Magnetic Torque Attitude control

- The attitude control is performed using actuator coils.
- Three coils (magnetorquers) are used to control the attitude, one for each axis.
- The coils generates a magnetic field that interacts with the Earths magnetic field and creates a moment.

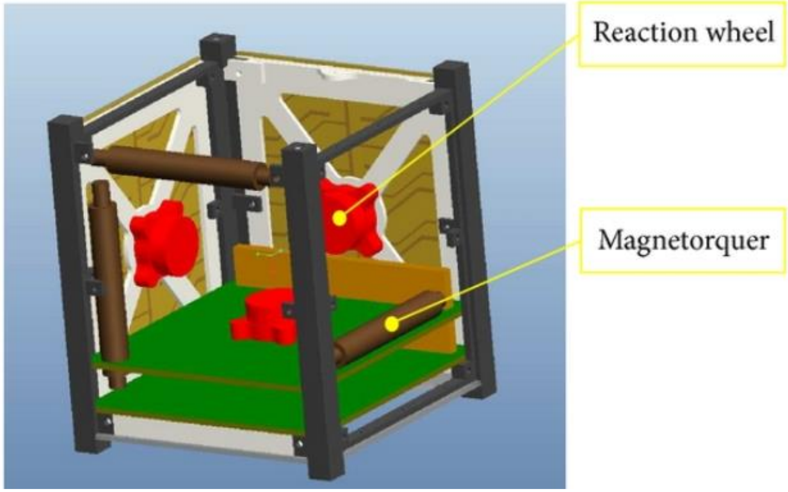


Figure from Hindawi

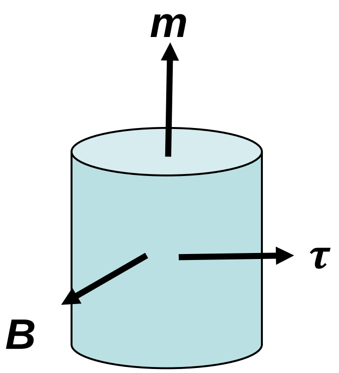
One possible control method – usually combined with another method

Magnetic Torque Attitude control

- The moment $\boldsymbol{\tau}$ is given by $\boldsymbol{\tau} = \boldsymbol{m}\ x\ \boldsymbol{B}$

commanded magnetic dipole moment

Earth's magnetic field vector (proportional to 1/r3 with r the distance of the center of Earth to the spacecraft

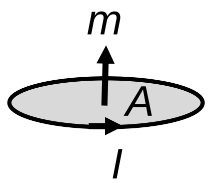


- Where *m* = *nIA*

Number of turns (constant)

Current through the loop **(direction and magnitude)**

Coil/loop area (constant)

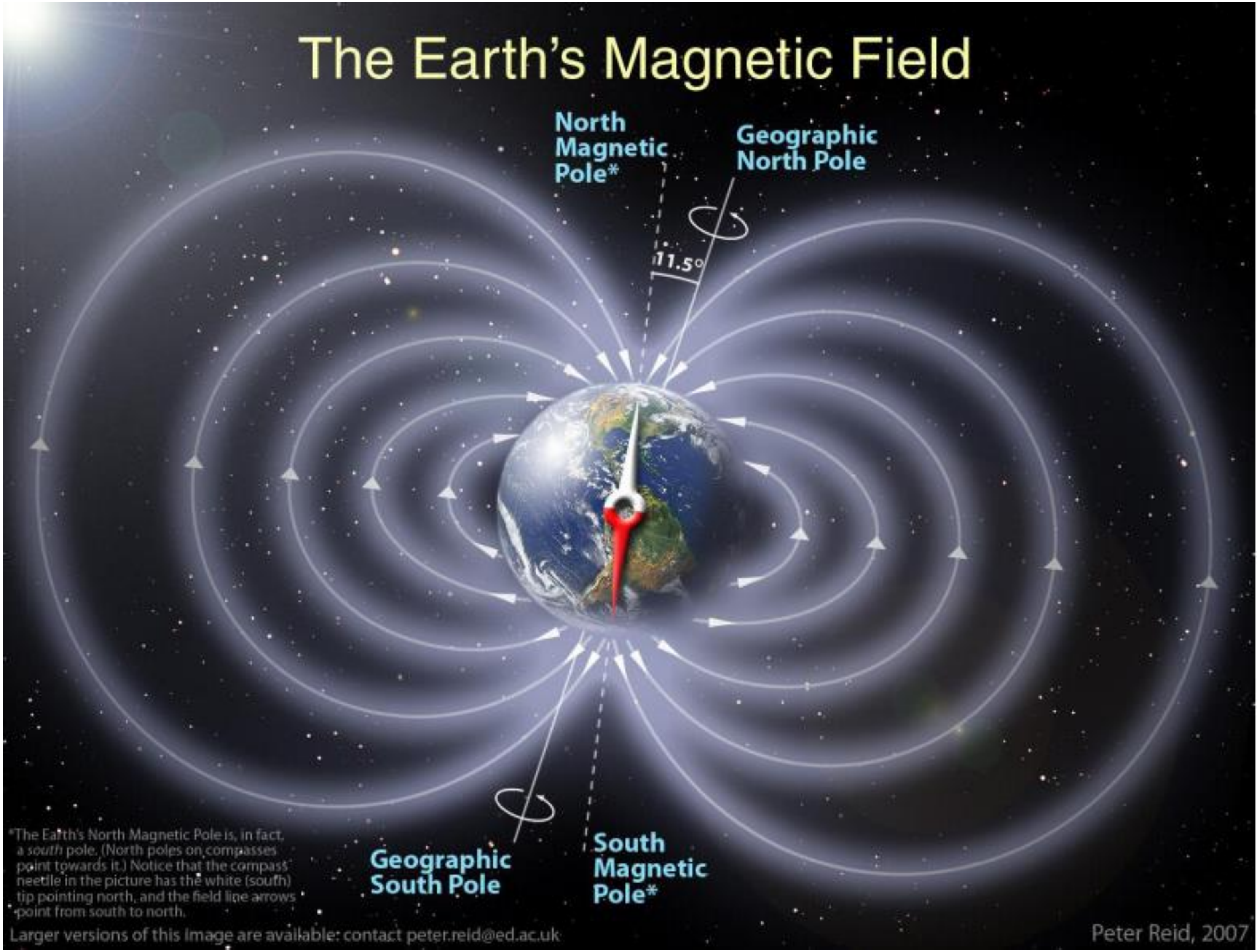


- Decomposed in the spacecraft body frame {b} we get:

$$\boldsymbol{\tau}^b = \boldsymbol{m}^b \times R^{be}\boldsymbol{B}^e$$

Required dipole moment must be calculated!

magnetic field vector in frame {e} given from a model



*The Earth's North Magnetic Pole is, in fact, a *south* pole. (North poles on compasses point towards it.) Notice that the compass needle in the picture has the white (south) tip pointing north, and the field line arrows point from south to north.

Larger versions of this image are available: contact peter.reid@ed.ac.uk

Peter Reid, 2007

Satellite control – detumbling

- The first task a spacecraft attitude control system must perform after separation from the launcher is to detumble the spacecraft, i.e., to bring it to a final condition with a sufficiently small angular velocity in all three axis.

- Will present a common control algorithm that is used to **detumble** (null the angular velocity) of a spacecraft.
  - Magnetic control has been used for decades to fulfill this task.

- Detumbling is necessary for satellites after orbital insertion
  - Also used on CubeSats.

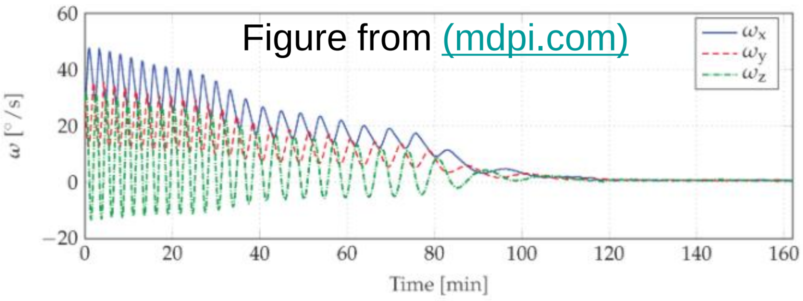


Figure from (mdpi.com)

CubeSTAR - Department of Physics (uio.no)

Detumbling using B-dot algorithm

- The principle of the B-dot algorithm relies on the usage of magnetorquers to generate a torque which is opposed to the “natural” rotation of the satellite, in order to reduce the angular rate.
- The control law creates a magnetic dipole in the opposite direction to the change in the magnetic field.
- The B-dot controller **uses a magnetometer to derive the angular rates** → No IMU / rate gyroscope is required!
- B-dot control law (represented in body frame):

$$\mathbf{m}^{b} = -k\,\dot{\mathbf{B}}^{b}$$

required magnetic dipole moment

a constant (to be tuned)

The time derivative of the B-field vector measured by the magnetometer

NB: see inertial navigation lecture

Why the B-dot controller works

- The relation between the B-field vector in body frame {b} and an inertial frame {i}: $\mathbf{B}^i = R^{ib}\mathbf{B}^b$
- Time derivation of the B vector gives:

$$\dot{\mathbf{B}}^i = R^{ib}\left(\dot{\mathbf{B}}^b + \Omega^{bi}\mathbf{B}^b\right)$$

~ 0 during the short sampling interval

$$\dot{\mathbf{B}}^i = R^{ib}\left(\dot{\mathbf{B}}^b + \boldsymbol{\omega}^{bi} \times \mathbf{B}^b\right)$$

$$\Downarrow$$

$$\dot{\mathbf{B}}^b = -\boldsymbol{\omega}^{bi} \times \mathbf{B}^b$$

Measured by an IMU/rate gyroscope. So, it is possible to avoid derivation of the B-field …

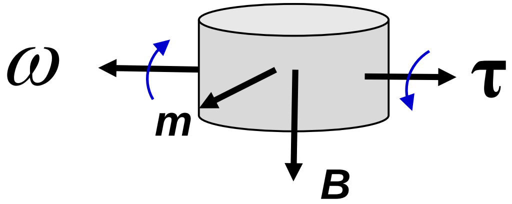
$$\Downarrow$$

Remember:

$$\dot{R}^{AB} = R^{AB}\Omega^{BA};$$

$$\Omega^{bi} = \boldsymbol{\omega}^{bi} \times$$

B-dot method generate a torque which is opposed to the rotation of the satellite



Derivation of signals

- B-dot can be calculate from

$$\dot{B}_n = \frac{B_n - B_{n-1}}{\Delta t}$$

- Note that a derivation (finite difference) amplify the noise in the measurements
  - A filter should be used to lower the noise level.

A practical implementation of the B-dot algorithm

- B-dot control is often implemented as a **bang-bang control law** (to avoid the difficulty of tuning the constant k).

- Assume that each magnetorquers can produce a maximum dipole moment of $\pm m_i^{max}$ in each axis, where i = 1,2,3.

- Then the bang-bang B-dot detumbling control commands in each axis is given by

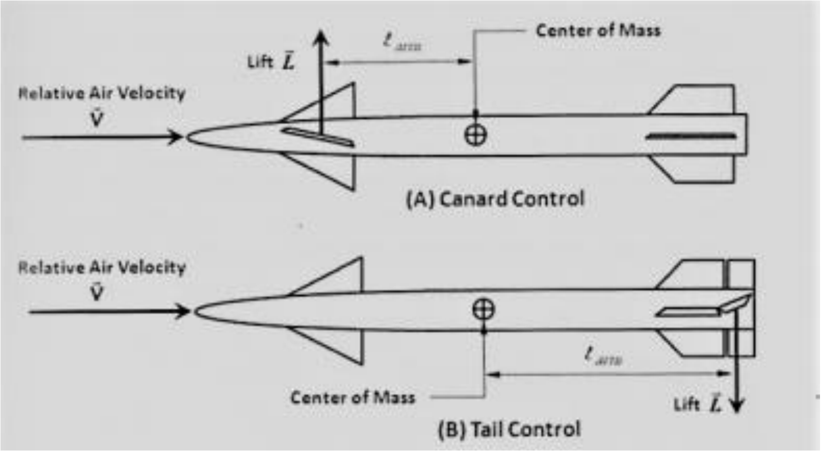
$$m_i = -m_i^{max} \underbrace{sign(\dot{B}_i)}_{\substack{1 \text{ if } \dot{B}_i > 0 \\ -1 \text{ if } \dot{B}_i < 0}}$$

Missile Guidance & Control

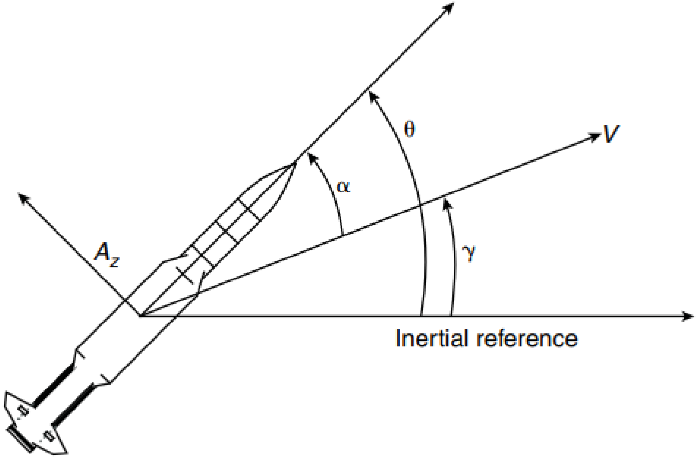
Missile guidance

- A missiles is guided towards a target by generating an acceleration A_z normal to the missile longitudinal axis
  - This force gives a change in the velocity vector V.

- (The required force is created by a lift force, by controlling aerodynamic surfaces / fins)



$$A_z = \frac{F_z}{m}$$



Missile Guidance, navigation & control (GNC)

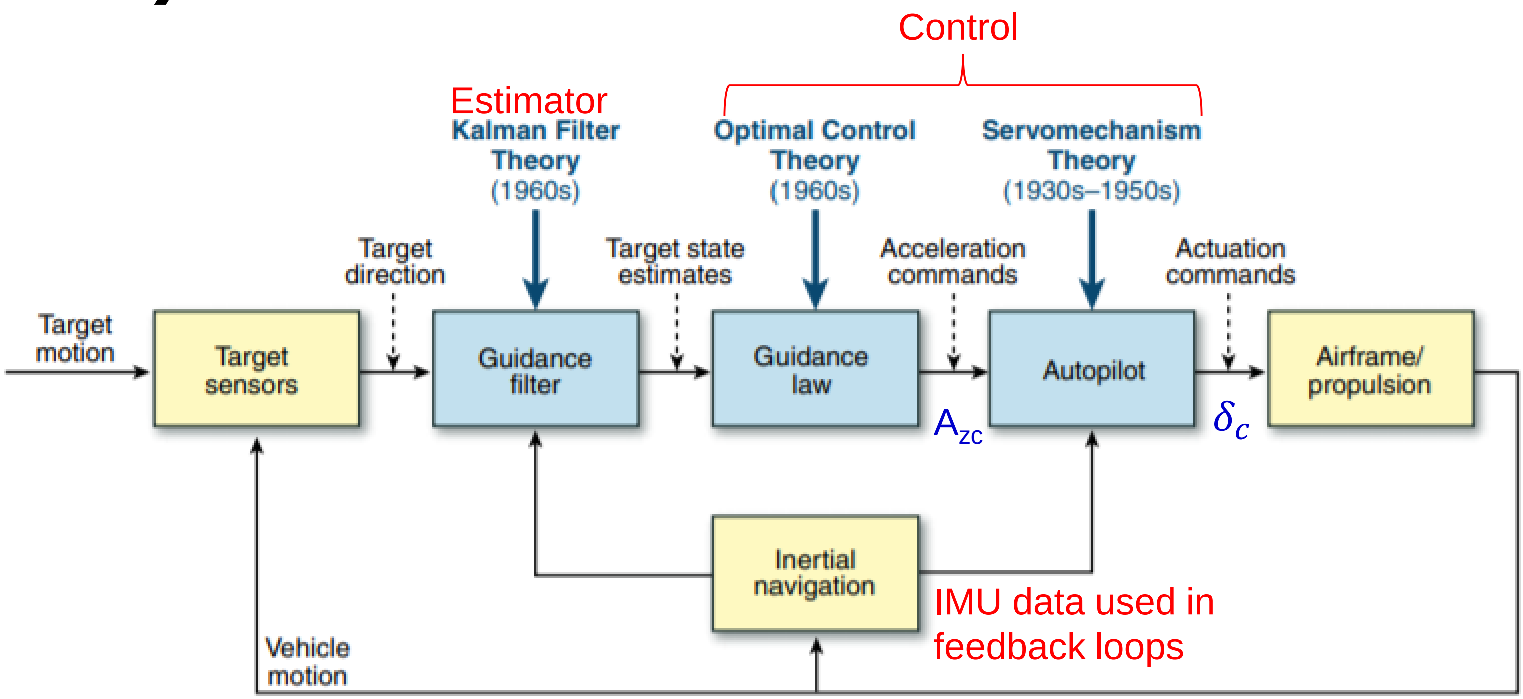
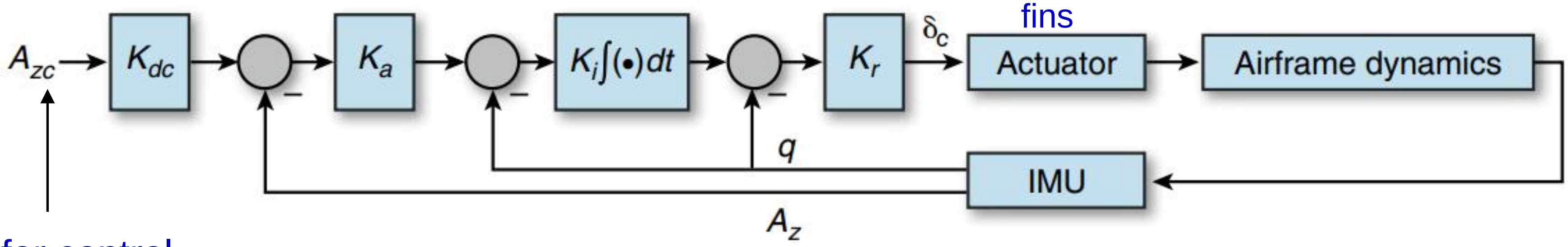


Figure 3. Traditional missile GNC topology. The traditional GNC topology for a guided missile comprises guidance filter, guidance law, autopilot, and inertial navigation components. Each component may be synthesized by using a variety of techniques, the most popular of which are indicated here in blue text.

Figure from Johns Hopkins - HOMING MISSILE GUIDANCE AND CONTROL

Missile acceleration control autopilot

- Used in all missiles.
- Classical approach to the design of an acceleration control autopilot:
  - The difference between the scaled input acceleration command A_{zc} and the measured acceleration A_z is multiplied by a gain K_a to effectively form a pitch rate command. The difference between the effective pitch rate command and the measured pitch rate q is multiplied by a gain K_i and integrated with respect to time. The resulting integral is differenced with the measured pitch rate q and multiplied by a third gain K_r to form the control command δ_c such as desired fin-deflection angle



fins

c for control

Figure from Johns Hopkins - HOMING MISSILE GUIDANCE AND CONTROL

But how do we calculate the required acceleration A_z ?

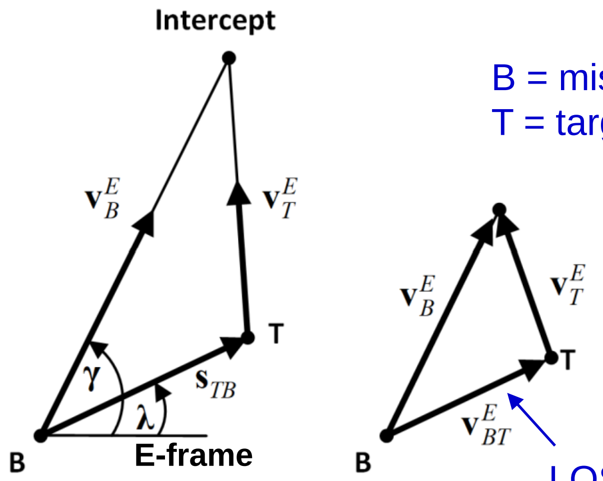
- Lets have a look in 2D (for simplification).
- Homing missiles (with a missile seeker in the nose) use a guidance method called **proportional navigation (PN)**, given by

$$A_z = NV\dot{\lambda}$$

Navigation constant (3 – 5)

Missile velocity magnitude $|v^E|$ calculated from an **IMU**

Line-of-sight (LOS) rate towards the target, calculated by the seeker



B = missile
T = target

LOS-vector

- The missile is guided towards an intercept point.