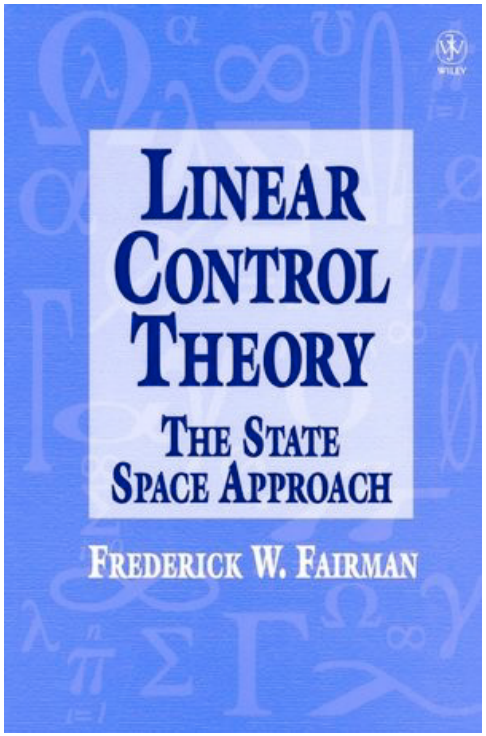
ROCO218: Control Engineering

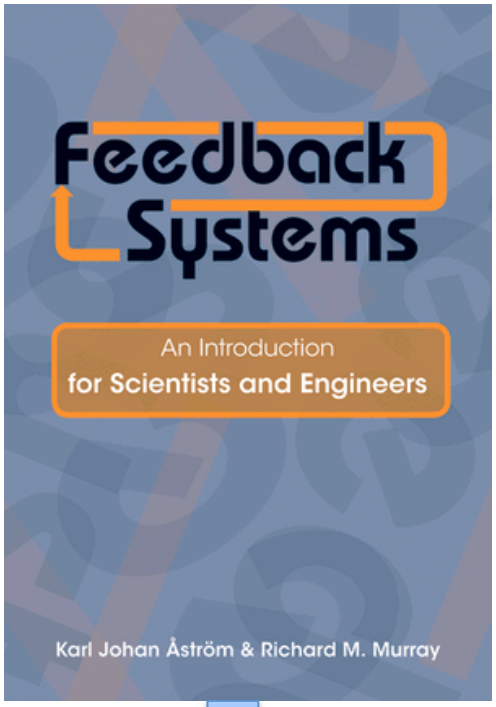
Dr Ian Howard

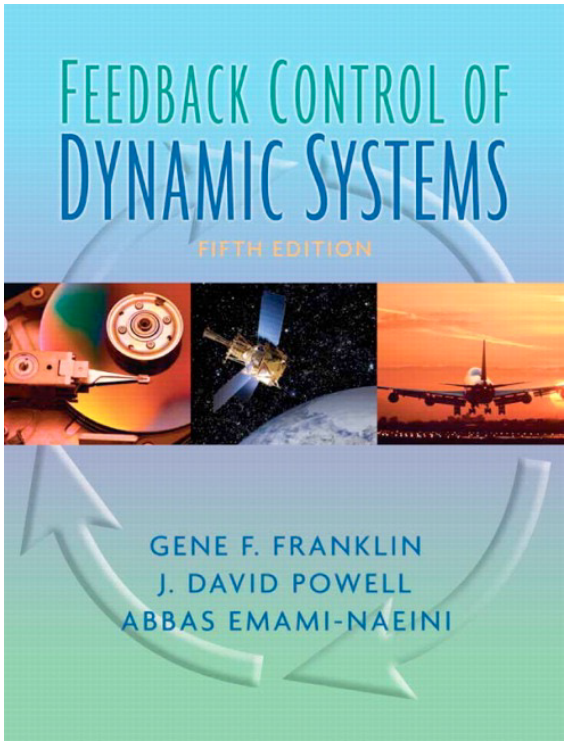
Lecture 1

Introduction to ROCO218

Resources







- http://www.cds.caltech.edu/~murray/amwiki/index.php/Main_Page

http://ctms.engin.umich.edu/CTMS

Module overview: Classical control theory

Classical control theory

- Differential equations and dynamics model of linear systems
- Laplace transforms and transfer functions
- System transient performance and steady state performance
- System frequency response, and Bode and Nyquist diagrams
- PID control

Module overview: Modern control theory

Modern control theory

- Revision of key Linear Algebra concepts essential for modern control theory
- State space design and representation ands modeling of Linear Time Invariant Systems
- Linearization and Stability
- Understanding controllability and observability
- State feedback control
- Observers and optimal control

Assessed learning outcomes

At end of the module the learner will be expected to be able to:

- LO1. Develop a transfer function for a given linear system, and characterize the linear system in terms of frequency response and transient performance
- LO2. Identify the stability of a given linear system
- LO3. Design a feedback controller to achieve stability and to improve dynamics performance for a given linear system

Assessments

- There will be a final exam assessments
  - ROCO218=60%
  - ROCO223=50%

- There will be practical assessments in the form of coursework
  - ROCO218=40%
  - ROCO223=50%

Laboratory practicals

Practical sessions

- Using MATLAB to simulate control systems
- Using SIMULINK graphic programming for simulation

- Coursework will be realistic simulation of inverted pendulum

Coursework assessment

- Please only submit your report as a PDF.
- Submission via DLE
- When you write your report, you must include you student number on every page of the report in a header and also include it in the document filename!
- Each laboratory report should contain a few pages of written explanation (although it can be more than this) as well as a set of images and diagrams to explain your solutions to the practical exercises.
- Please also incorporate links to uploaded YouTube videos to illustrate operation of your project.

Please use the correct report format!

ROCO218: Control Engineering

Dr Ian Howard

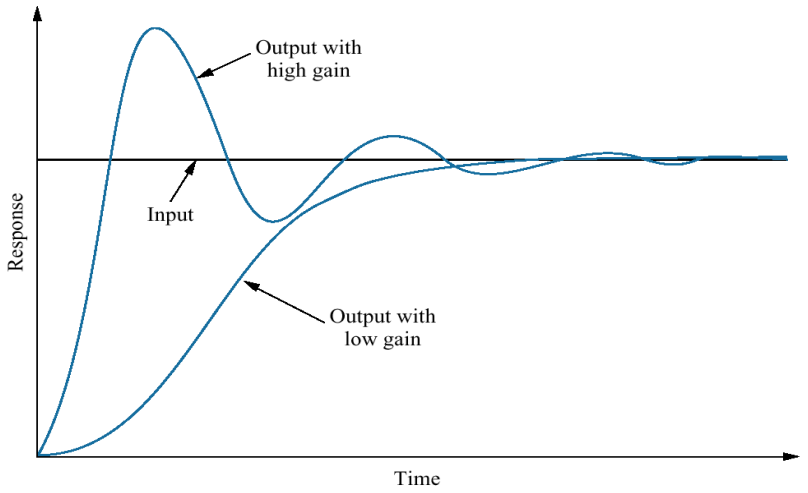
Lecture 1

Introduction to control engineering

Engineering goals of control theory

- We want to control complex dynamic systems
- Generate appropriate actions so systems do what we would like them to do

- Parameters of the controller affect performance
- Response of a position control system showing effect of high and low controller gain on the output response
- Think about steering a ship
- Not easy as it doesn't immediately respond



History of control

Ad hoc control schemes (BC - 18 century):

- Classical control (18th century - 1950s): - Contents of 1st part of this course
- Modern control (1960s) - Contents of 2nd part of this course

Post-modern control (1980s)

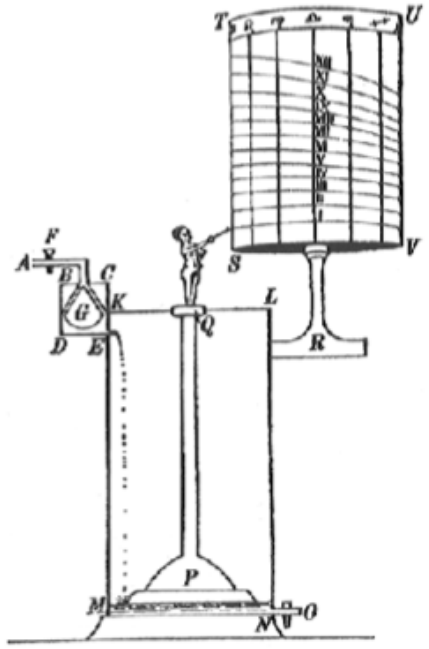
- Transfer function
- Frequency domain
- Ingenious mechanical control systems
- State-space model
- Time domain

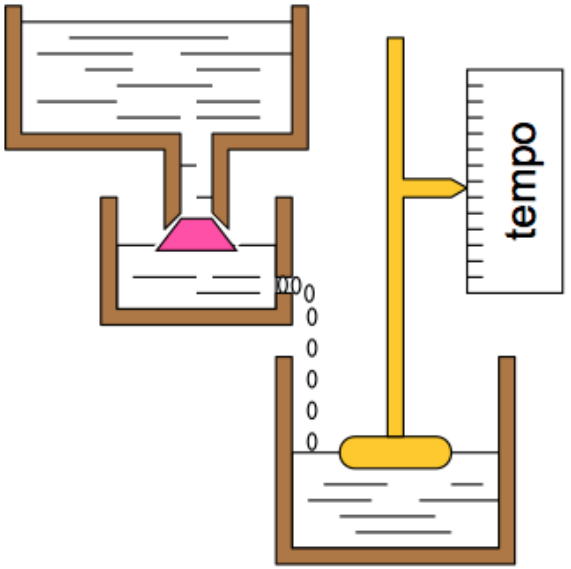
Post-modern control (present day / Research):

- Optimal control
- Hybrid control, etc.
- Robust control
- Control of high-dimensional complex
- systems, e.g.: time varying networks, etc.

History of control: Water clock

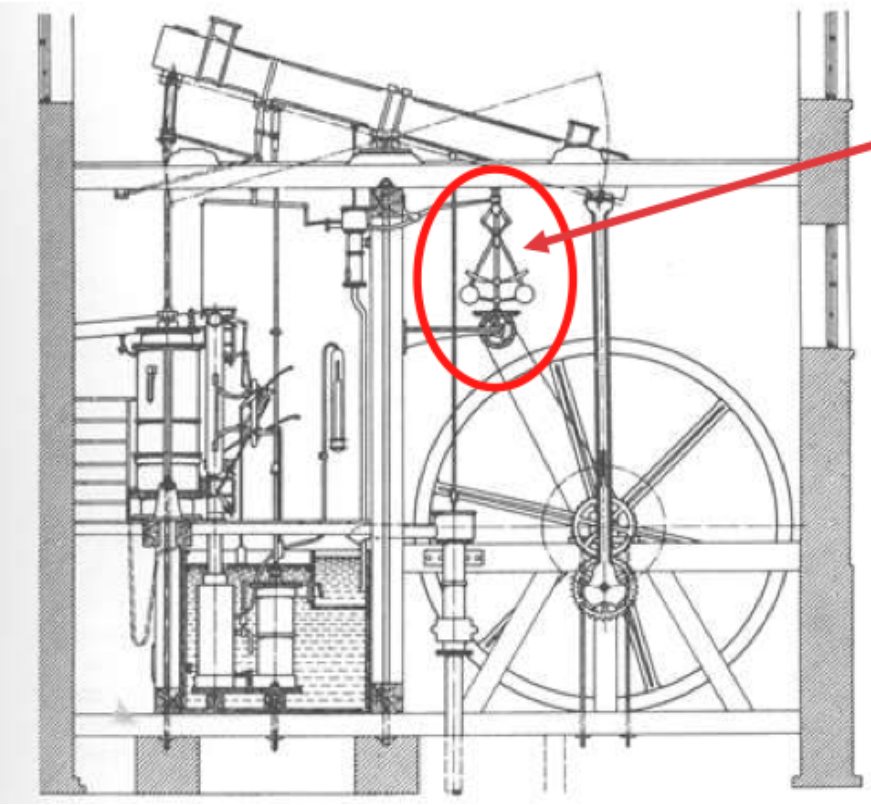
- First tank controlled to generate constant water pressure
- Second tank shows increase in volume over time – indicating passage of time
- Toilets still use the first tank filing mechanism to shut of water entering when it is full

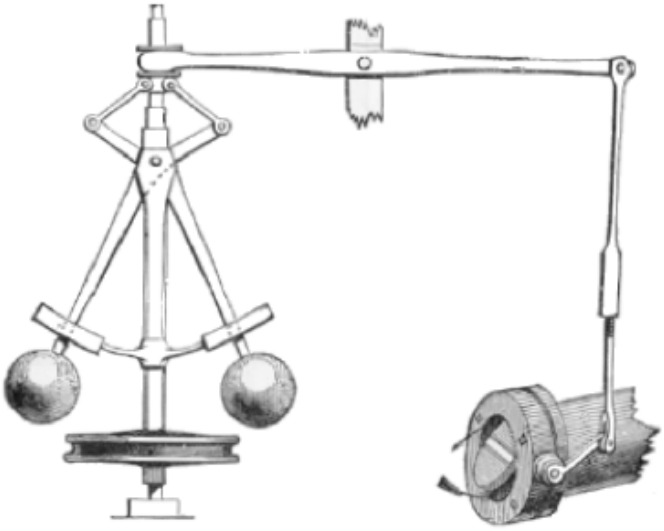




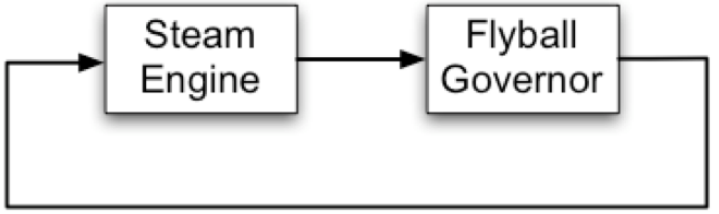
Water Clock
Ktesibios 300 B.C.

History of control: James Watt's steam governor





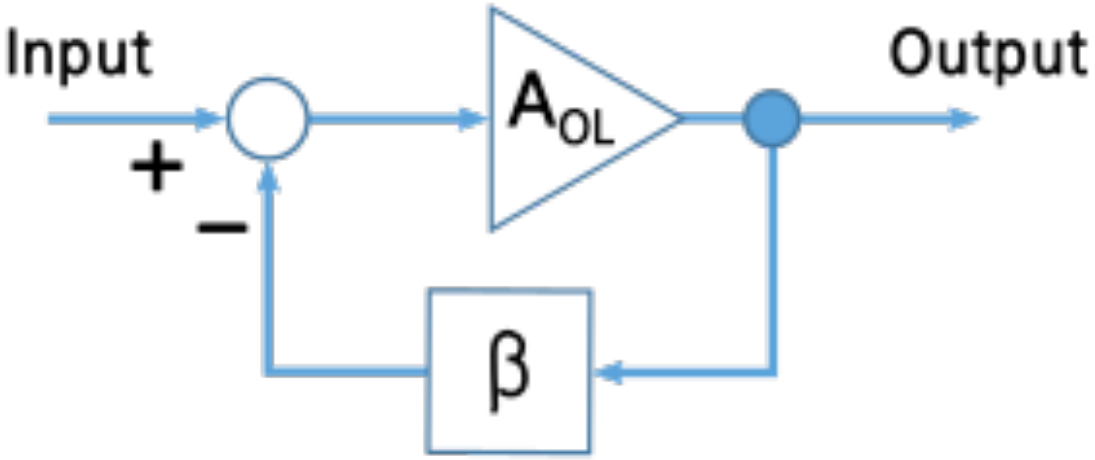
James Watt's automatic control of the speed of a steam engine (1798)
As engine speed increases, fly balls are thrown outwards and linkage to valve cut off reduces steam supply, thereby regulating speed



Feedback schematic

History of control: Negative feedback amplifier

Electronic amplifier with negative feedback Harold Black, 1927)



- Scheme has important features
- An amplifier with gain $\mathbf{A}_{OL}$
- A feedback network ß which senses the output signal and can transform it in some way (e.g., with gain of or filtering it)
- A summing circuit

Amplifier circuit subtracts a fraction of its output from its input, so that the negative feedback opposes the original signal.

The negative feedback

- improves stability, linearity, frequency response, step response
- reduces sensitivity to parameter variations (e.g. due to manufacturing or environment)

Control theory is ubiquitous

- Aerospace and space applications

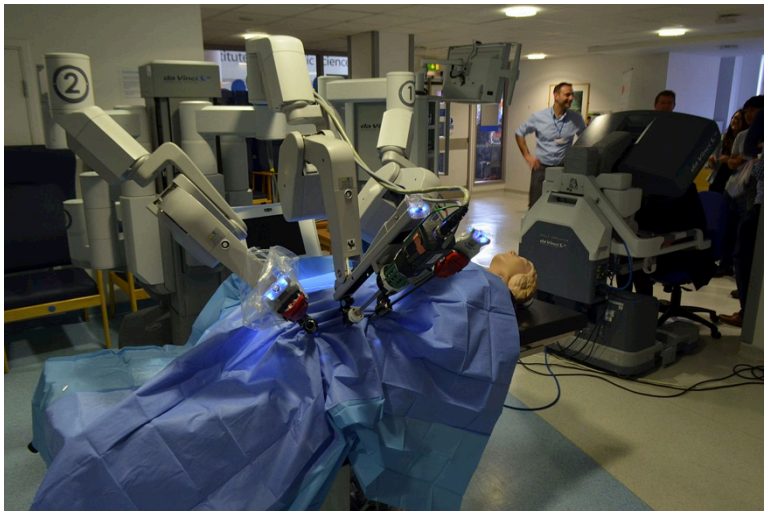
Blue Origin

https://www.youtube.com/watch?v=9pillaOxGCo

- Medical applications

Da Vinci surgery robot

https://www.youtube.com/watch?v=e_fUVBdwmwA



Control theory is ubiquitous

- Robotics

Quadcopters

https://youtu.be/w2itwFJCgFQ

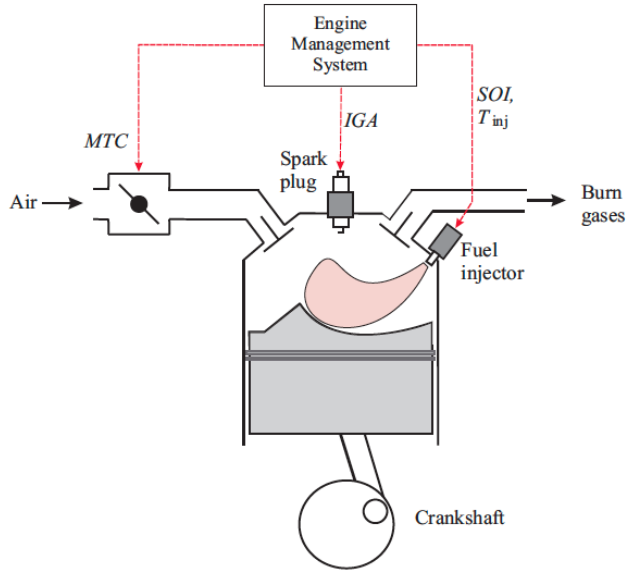
https://youtu.be/3CR5y8qZf0Y

- Manufacturing industry

- https://youtu.be/7U1-X5ogsKA

Control theory is ubiquitous

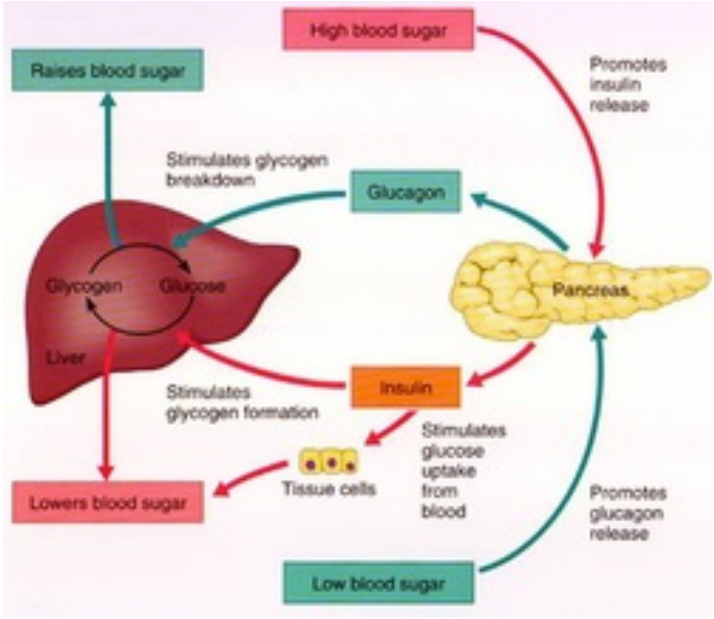
- Automotive industry
  - Engine control



- Industrial process control
  - -Mixing materials
  - -Flow control

Control theory is ubiquitous

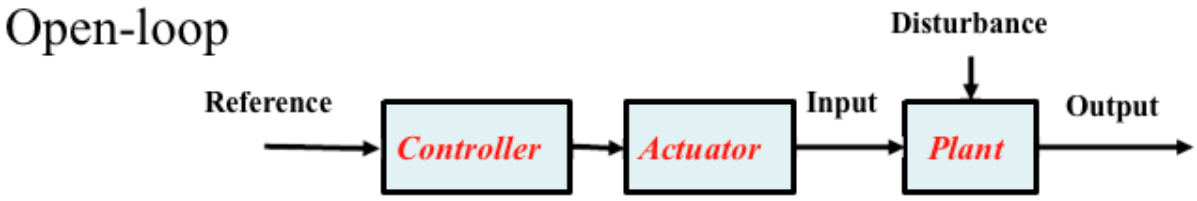
- Biological systems
  - -Regulation of hormones
  - -Blood rate
  - -Breathing
  - -etc.



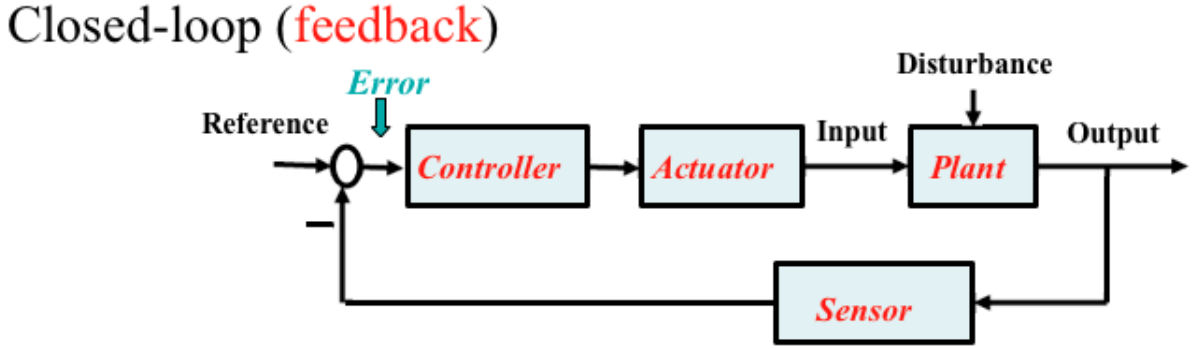
- Power networks
  - -Switching in supplies

Types of control systems

Open-Loop Control Systems utilize a feed forward controller or control actuator to obtain the desired response.

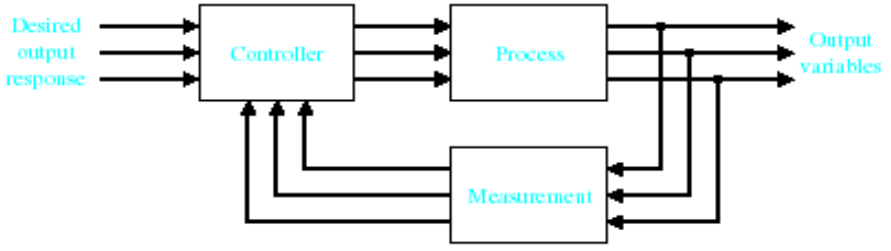


Closed-Loop Control Systems utilizes feedback to compare the actual output to the desired output response.

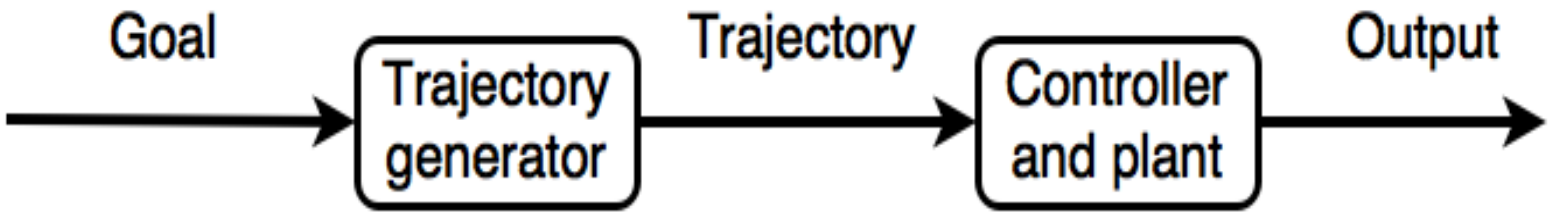


Multivariable Control System

Multiple inputs and outputs



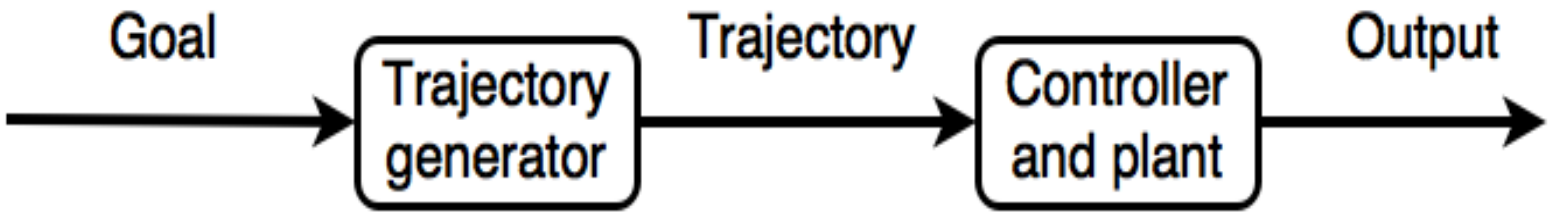
Trajectory generation



- To produce a movement trajectory, a separate module is usually used to generate the goal in a feed forward manner.

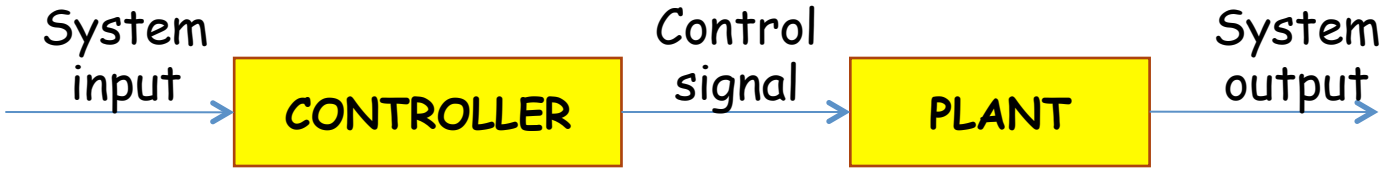
- In this scheme, although the controller can use feedback from the plant and its measured output, no feedback is used in the generation of the goal trajectory.

Feedforward control



- Basic pre-processing idea where the goal signal corresponds with a given output.
- We generally need to find a way to convert the goal into an effective signal to control input to the plant.
- An inverse model maps the input goal to the signal needed to control the plant.
- One simple feed forward way to achieve this is to provide the input signal with an inverse model of the plant
- However, such feed forward control assumes that it is known exactly how the plant will behave and these schemes cannot account for unexpected disturbances present in the task.

Open loop control



- Open-loop control systems: those systems in which the output has no effect on the control action

- E.g. electric heater where just the right number of elements switched on and then take no notice of how warm room becomes

Comments on open-loop control systems

Open loop systems are

- Simple construction and ease of maintenance.
- Less expensive than a closed-loop system.
- No stability problem.

Good

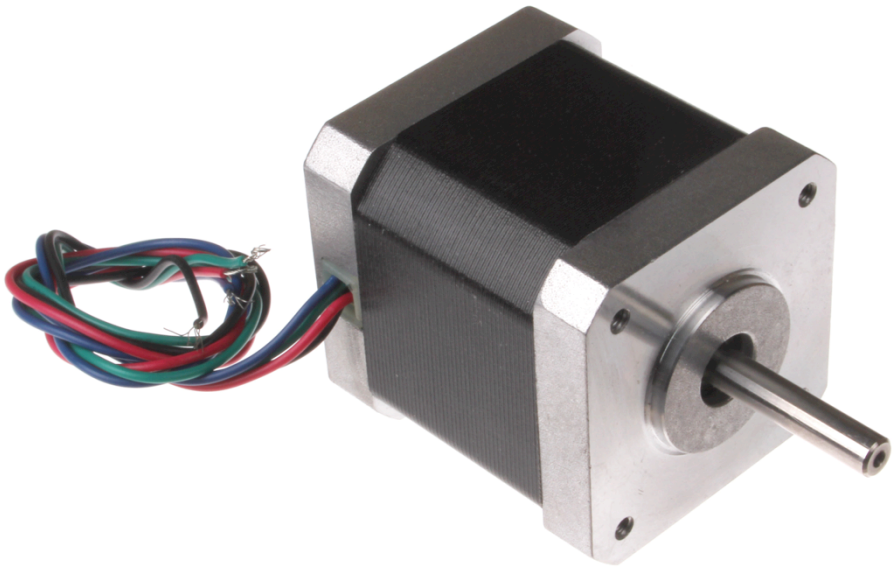
- Recalibration is necessary from time to time.
- Sensitive to disturbances, so less accurate.

Bad

When should we apply open-loop control?

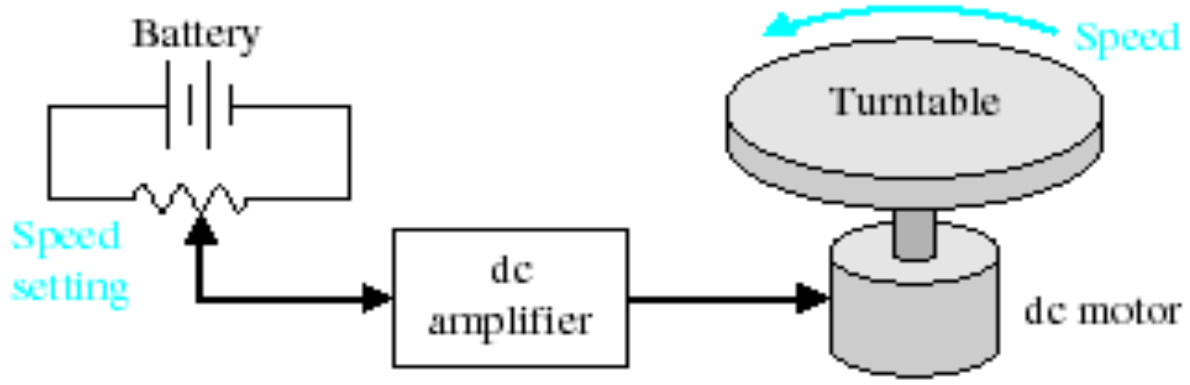
- The relationship between the input and output is exactly known
- There are neither internal nor external disturbances.
- Measuring the output precisely is very hard or economically infeasible

Stepper motor normally used open-loop

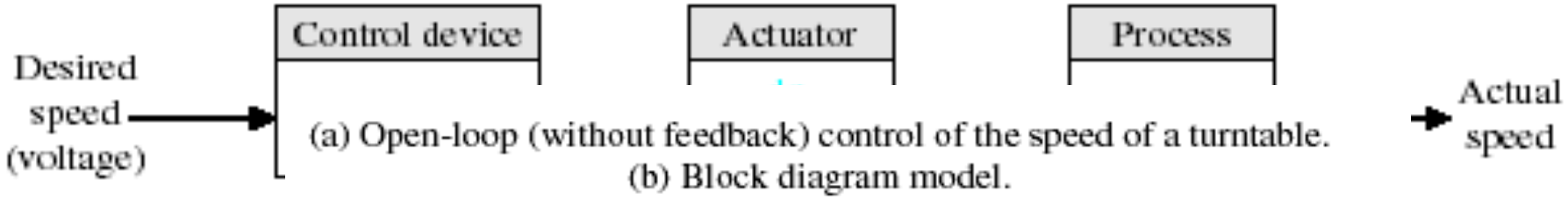


Provided calibrated and no steps are lost this solution is very effective e.g. for CNC, 3D printers

Open loop example

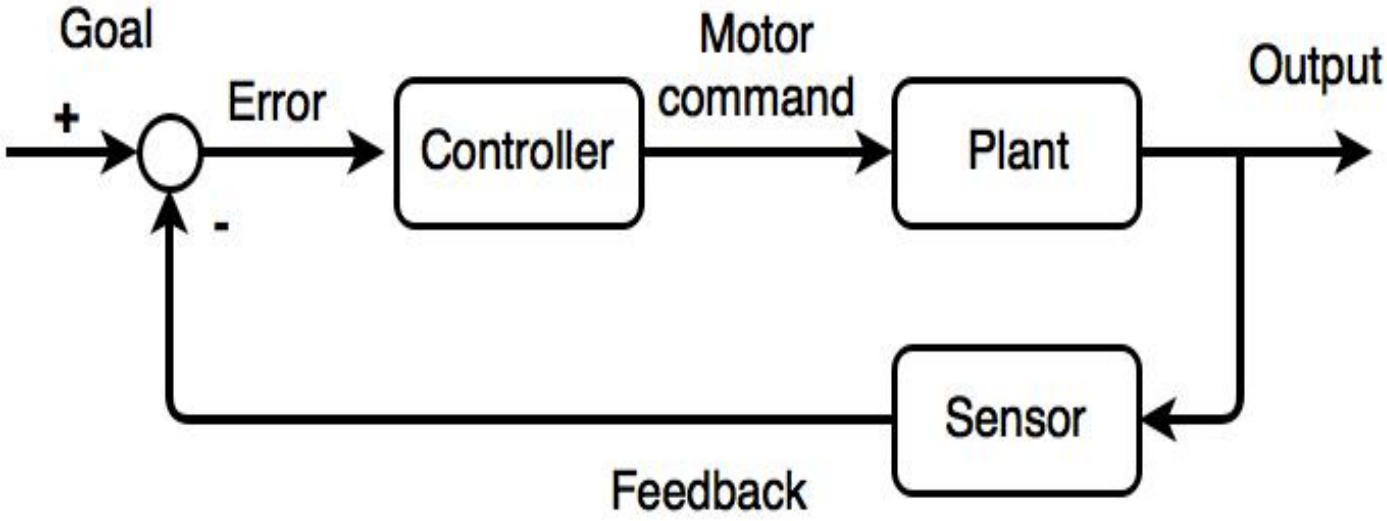


(a)



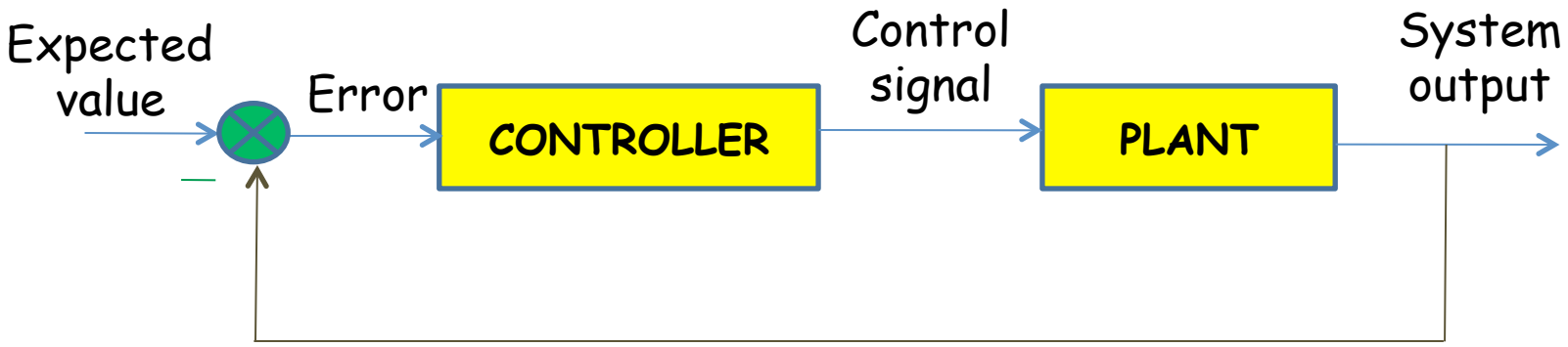
(a) Open-loop (without feedback) control of the speed of a turntable.
(b) Block diagram model.

Output feedback control



- Feedback control is often used in engineering systems to stabilize operating goals when noise is present.
- Under output feedback control, the sensory consequences are used to modify the control input by comparing it with the goal to calculate error and this is then used in an attempt to get the plant to meet the required goals.
- Error is fed via the controller to generate motor commands.
- System output then moves to the goal.
- This scheme will also have the ability to compensate for disturbances.

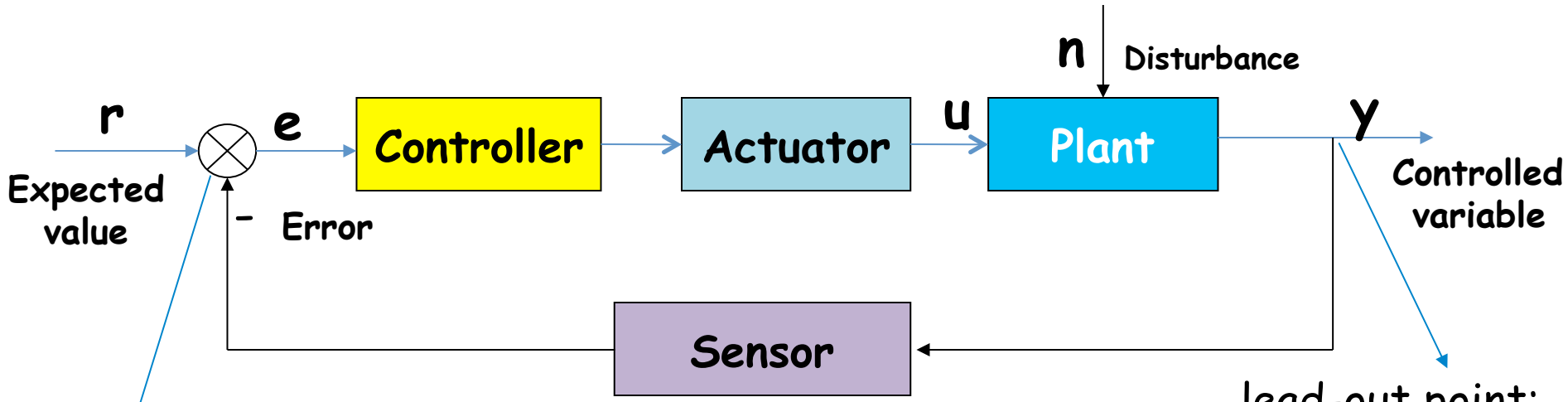
Closed loop control



- Closed-loop control systems use feedback
- They compare the actual output with the expected value and take actions based on this error

- Reduce the effect of disturbance
- Make a system insensitive to variations
- Can stabilize an unstable system

Block diagram of output feedback control system



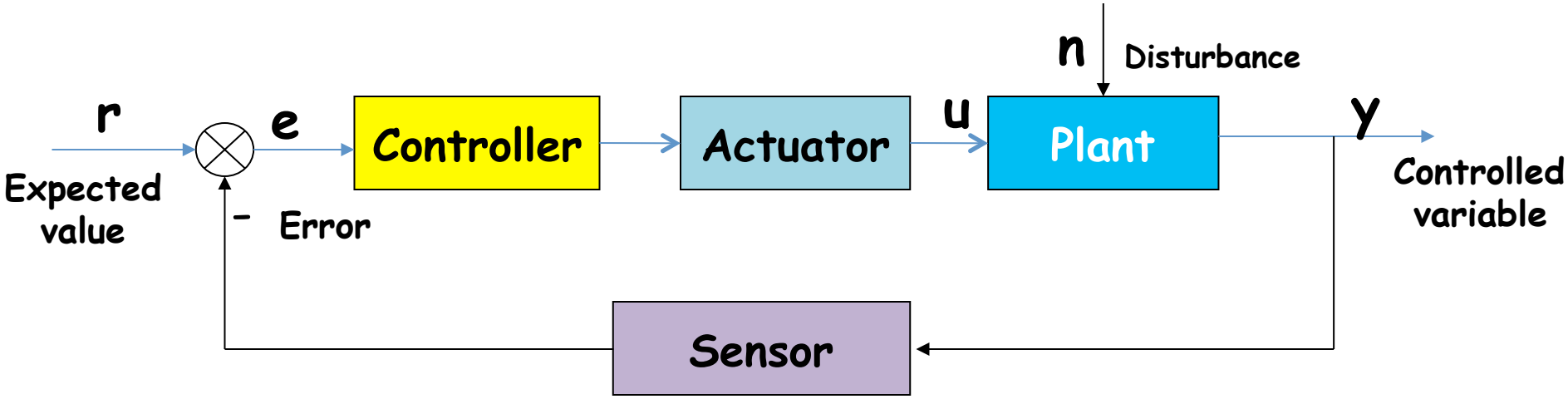
comparison component (comparison point) : its output equals the algebraic sum of all input signals.

"+": plus; "-": minus

The Block represents the function and name of its corresponding mode, we don't need to draw detailed structure, and the line guides for the transfer route.

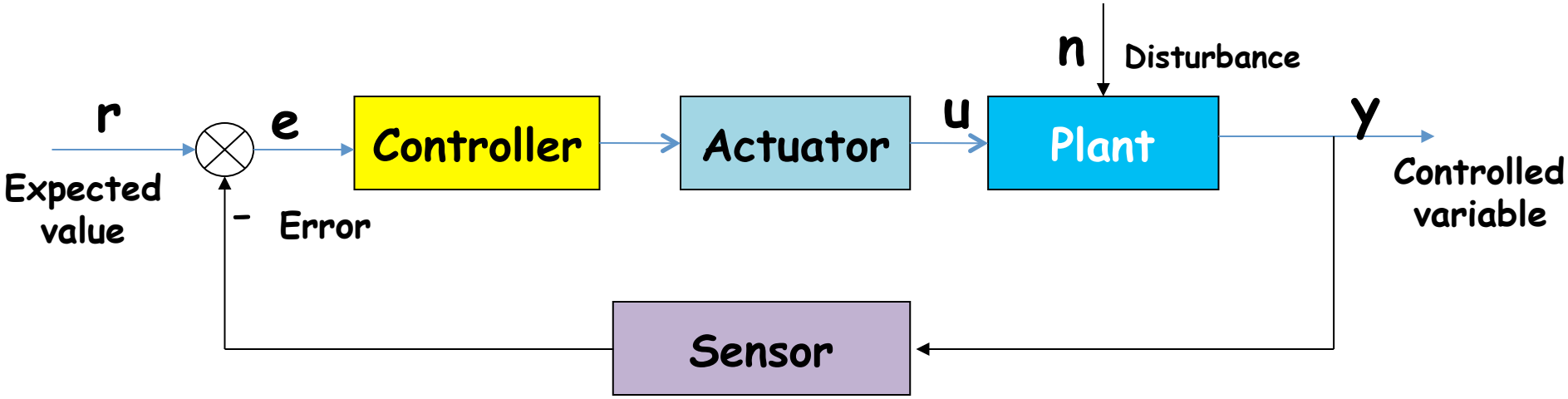
lead-out point: Here, the signal is transferred along two separate routes.

Block diagram of output feedback control system



1 .**Plant**: a physical object to be controlled such as a mechanical device, a heating furnace, a chemical reactor or a spacecraft.

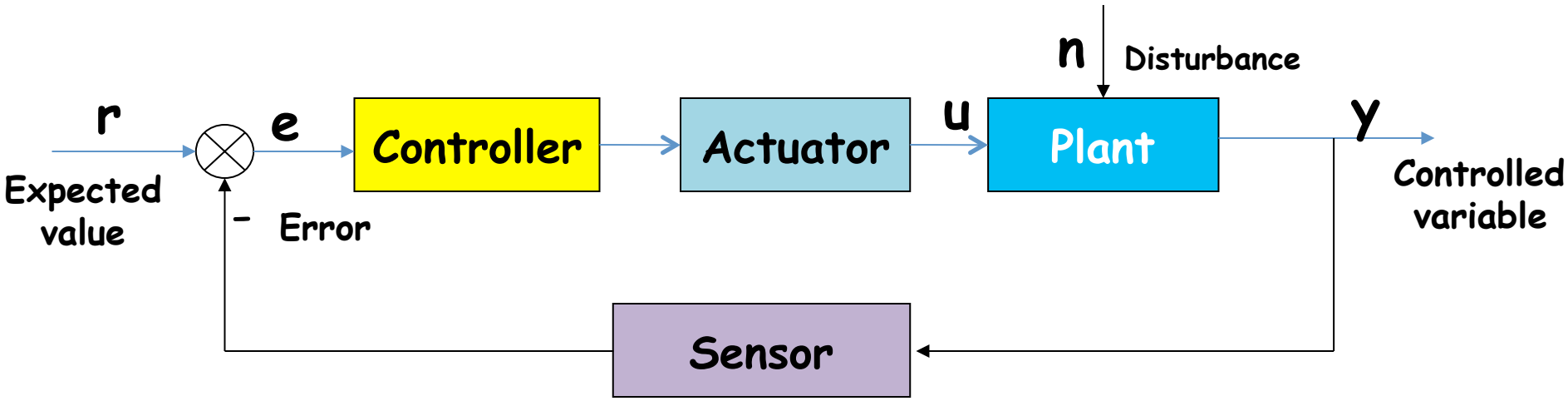
Block diagram of output feedback control system



Controlled variable

2.Controlled variable: the variable controlled by Automatic Control System , generally refers to the system output.

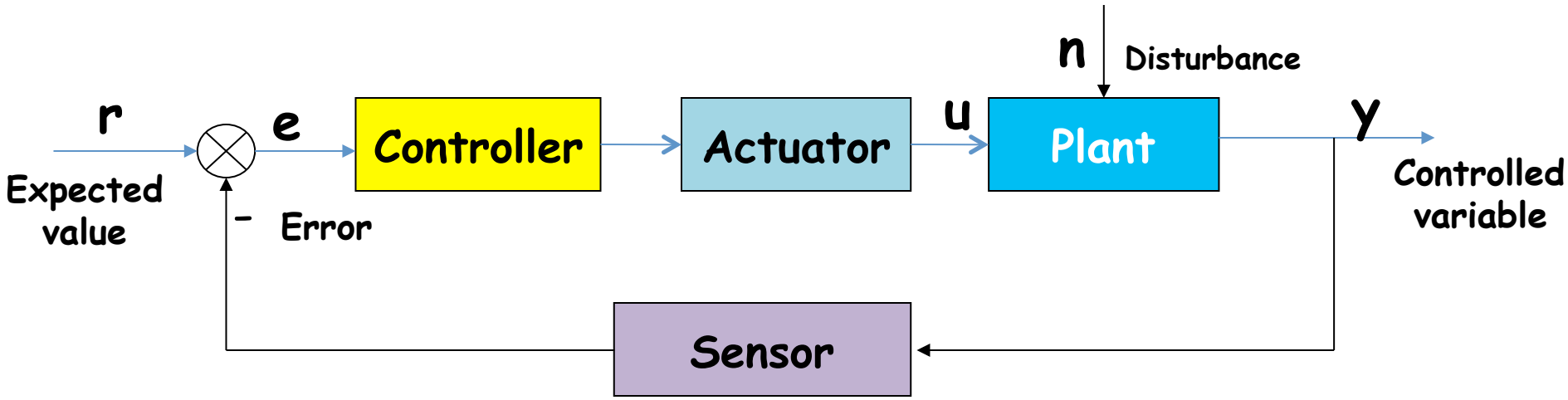
Block diagram of output feedback control system



Expected value

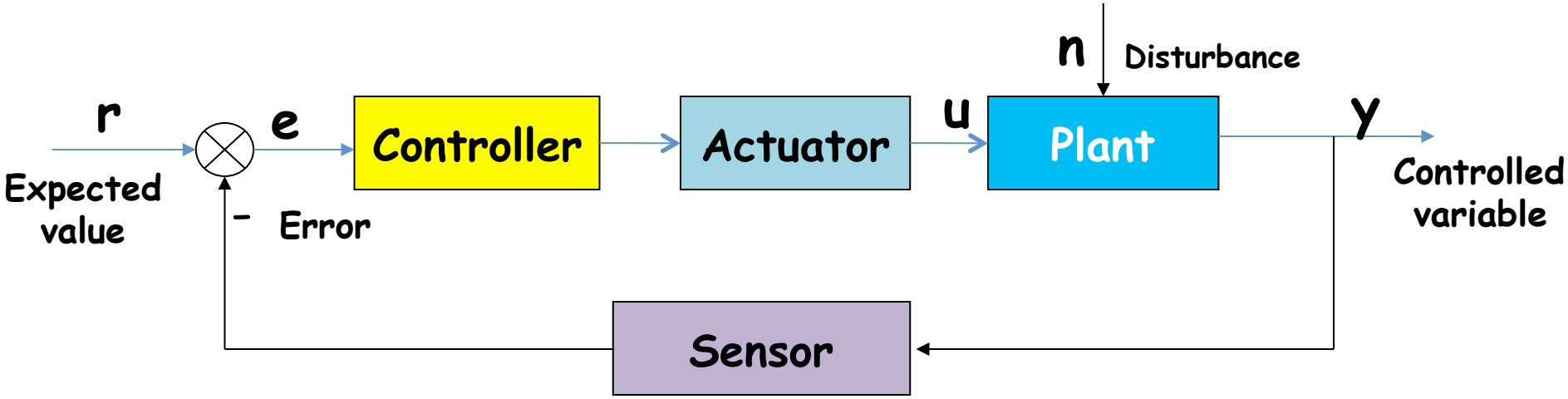
3. Expected value : the desired value of controlled variable based on requirement, often it is used as the reference input

Block diagram of output feedback control system



4. Controller: an agent that can calculate the required control signal.

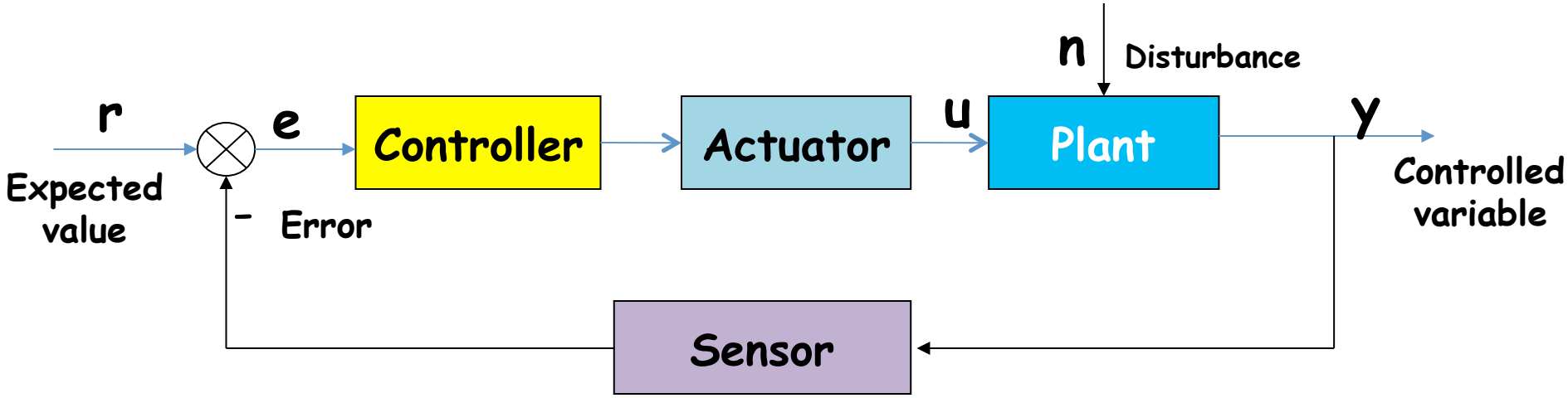
Block diagram of output feedback control system

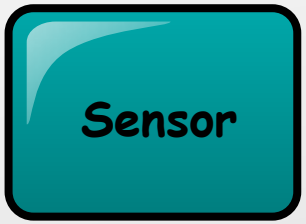


Actuator

5. Actuator: a mechanical device that takes energy, usually created by air, electricity, or liquid, and converts that into some kind of motion.

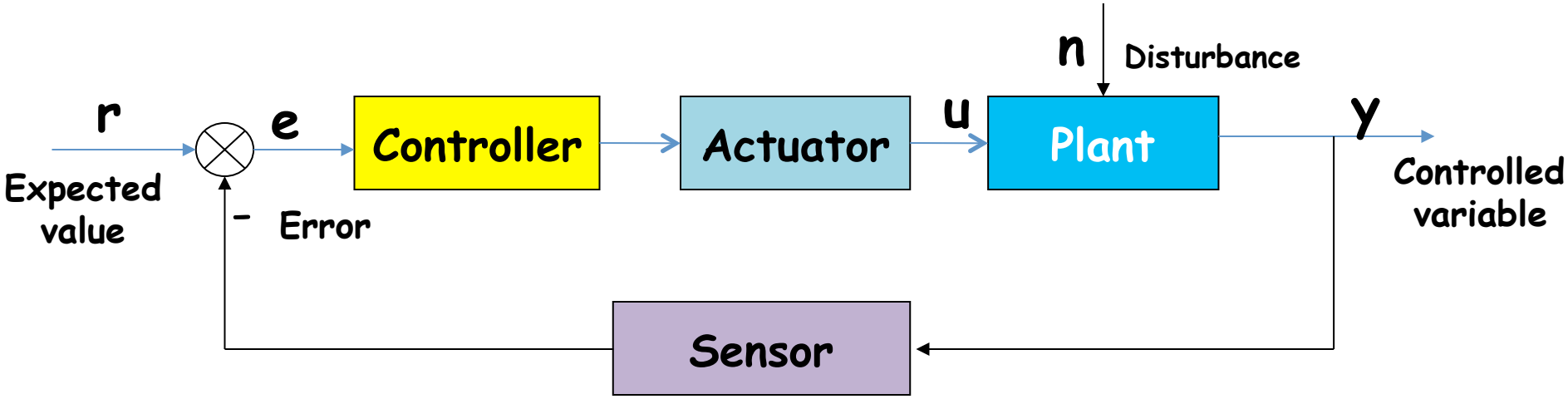
Block diagram of output feedback control system





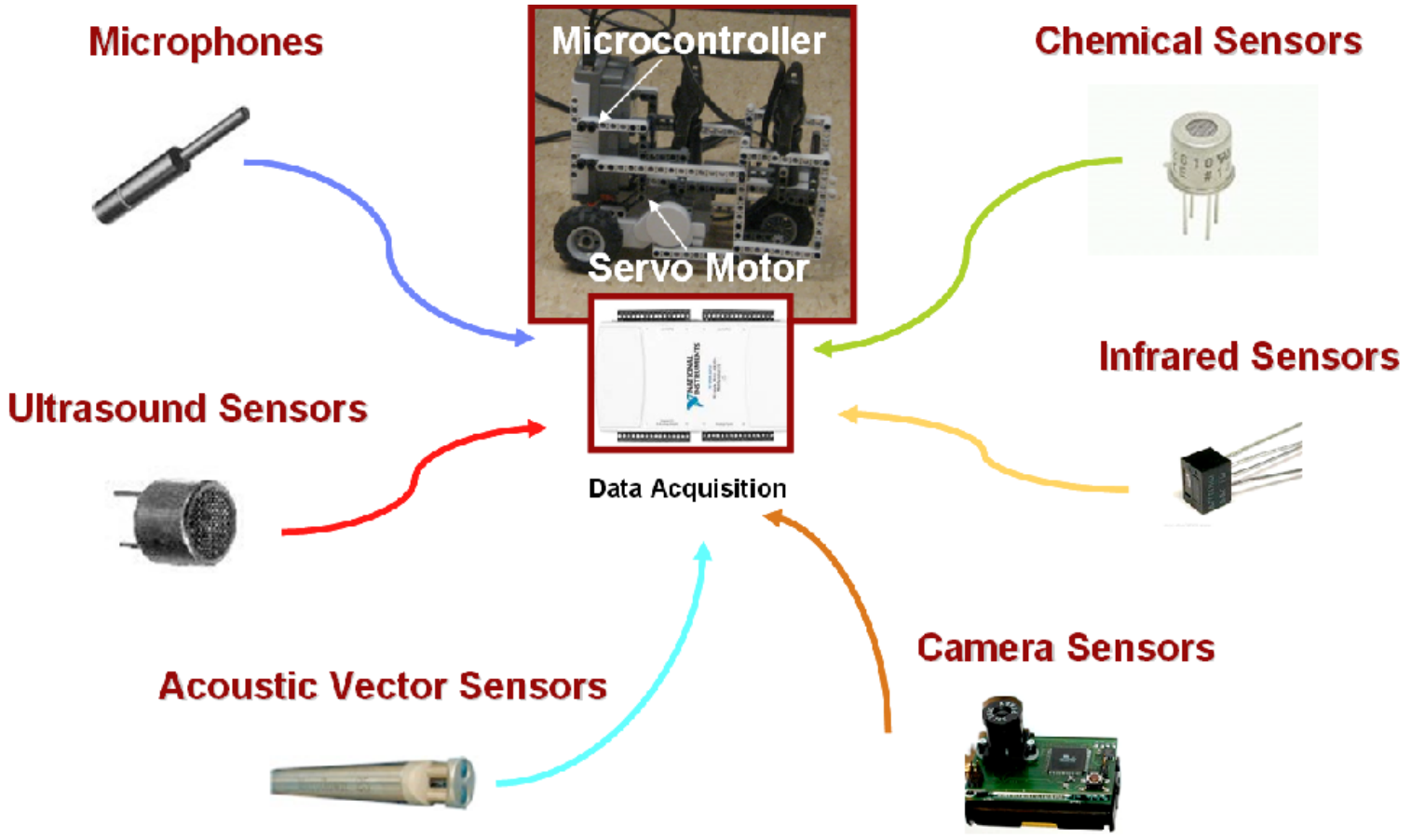
6. Sensor : a device that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument.

Block diagram of output feedback control system



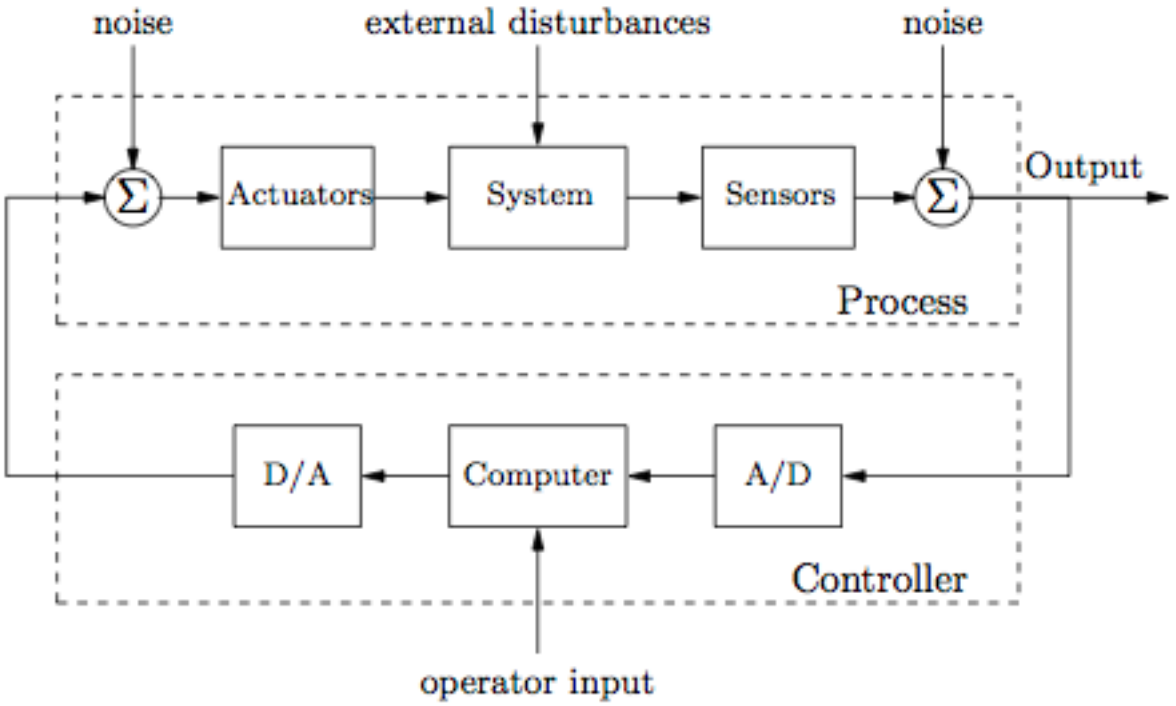
7. Disturbance: the unexpected factors disturbing the normal functional relationship between the controlling and controlled parameter variations.

Control systems need sensors



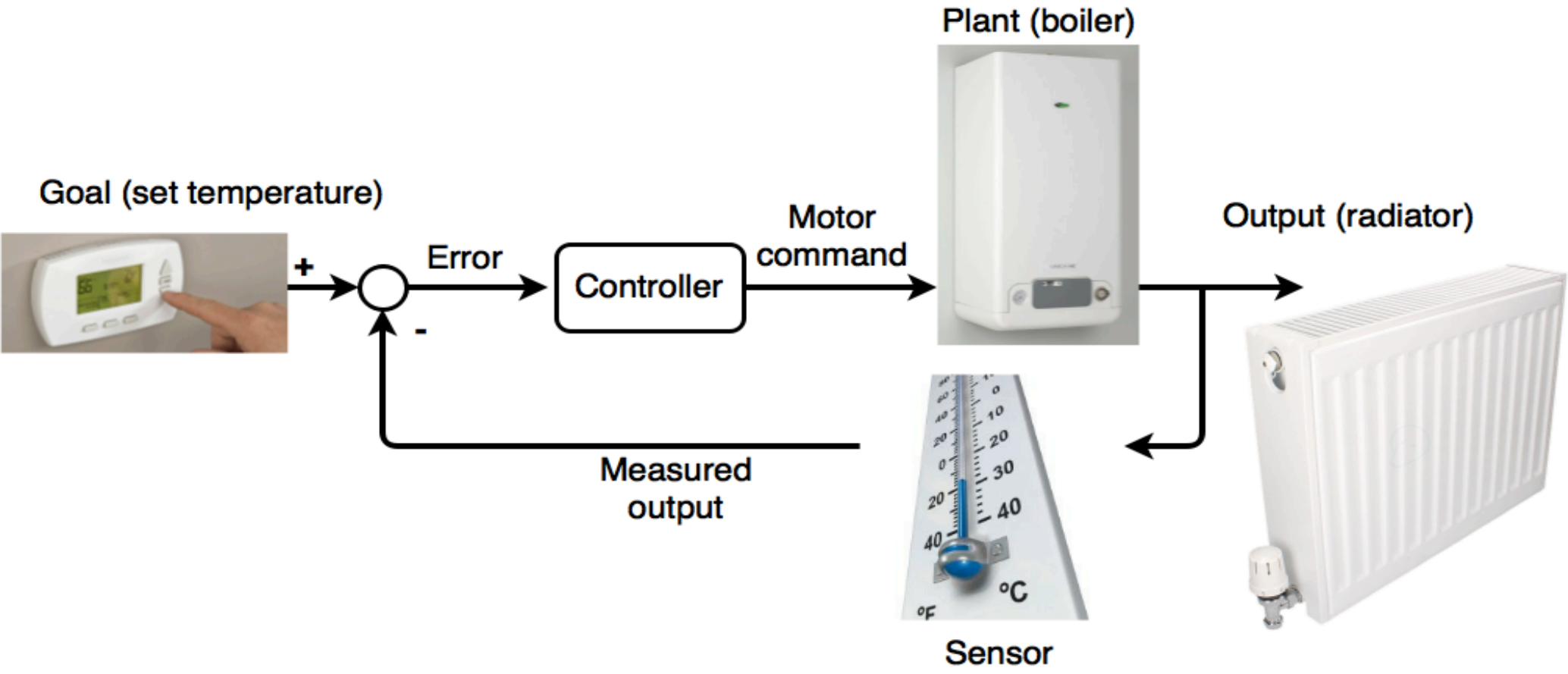
- Sensors needed to make measurements
- Can be used for feedback signal
- But sensors can also be noisy
- In general sensors need to be modeled too!

Digital computer control system

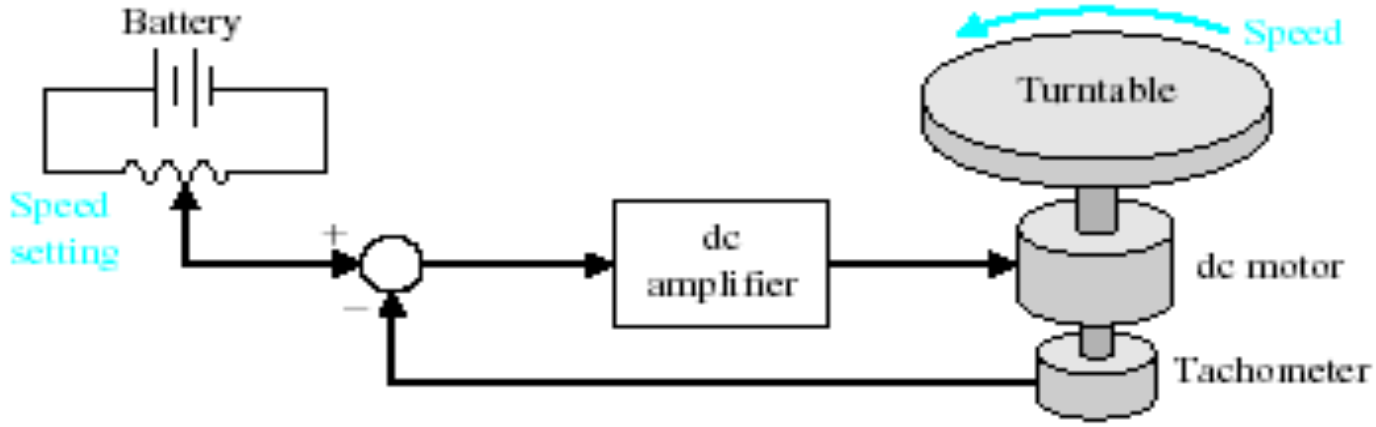


- Analog-to-digital (A/D) reads input signals
- Controller is implemented by a digital computer
- Digital-to-Analog (D/A) generates control signal
- NB: analogue and hybrid computers can also be used!
- What about the human brain?

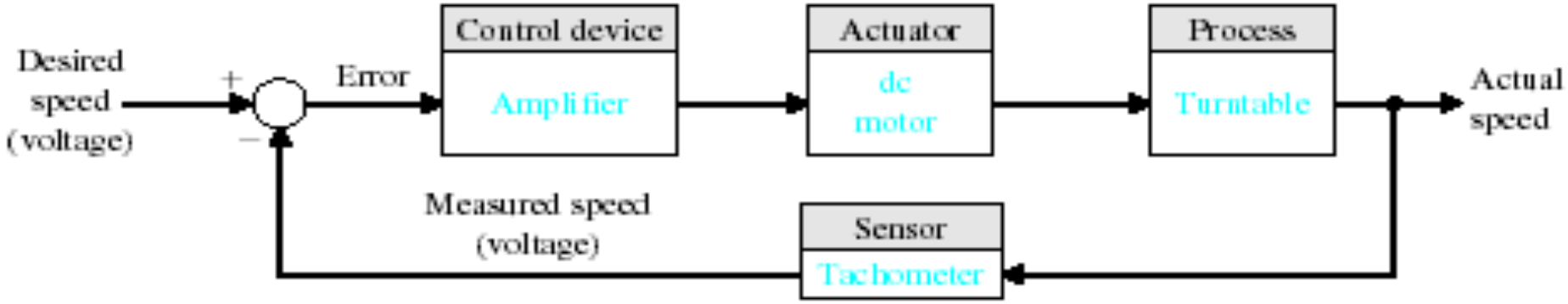
Example of simple feedback control



Closed-loop example



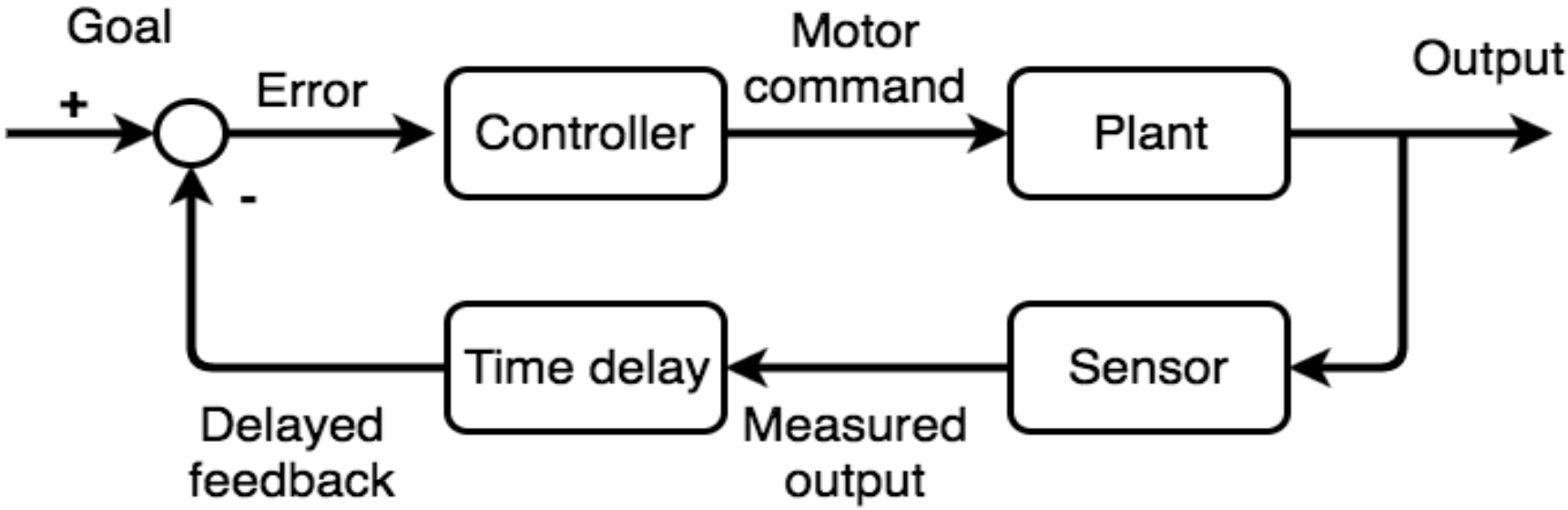
(a)



(b)

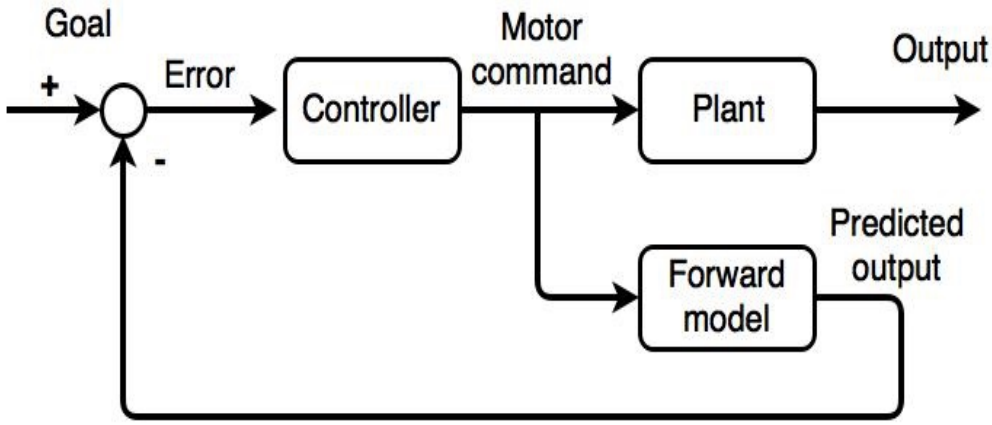
(a) Closed-loop control of the speed of a turntable.
(b) Block diagram model.

But time delays are a problem



- Time delays in processing place limitations on feedback control schemes
- When signal fed back in-phase system can go unstable!

Output feedback control with forward model



- Time delays during processing place limitations on feedback control schemes.
- One solution to this problem is to employ a control scheme that employs a forward model of the plant dynamics to provide an un-delayed estimate of the system output.
- The forward model estimates the output, which is then used in the feedback loop, rather than having to wait until actual direct feedback is received.
- Using a forward model to predict sensory output, which is then used in the feedback path stabiles plant operation

Closed-loop control systems

- Feedback is a key idea in the discipline of control.
- In practice, feedback control system and closed-loop control system are used interchangeably
- Closed-loop control always implies the use of feedback control action in order to reduce system error

- What is the Control System Engineer trying to achieve?
- First, understand the broader picture of the application to best apply a suitable control system.
- A good control system is a system that will
  - generate a response quickly and without oscillation (*good transient response*),
  - have low error once settled (*good steady-state response*),
  - and will not oscillate wildly or damage that system (*stability*).

Key points regarding feedback control

- Main advantages of feedback:
  - reduce disturbance effects
  - make system insensitive to variations
  - stabilize an unstable system
- create well-defined relationship between output and reference
- Potential drawbacks of feedback:
  - cause instability if not used properly
  - couple noise from sensors into the dynamics of a system
  - increase the overall complexity of a system

Open-loop versus closed-loop

- Open-loop control

Simple structure, low cost

Easy to regulate

Low accuracy and resistance to disturbance

- Closed-loop control

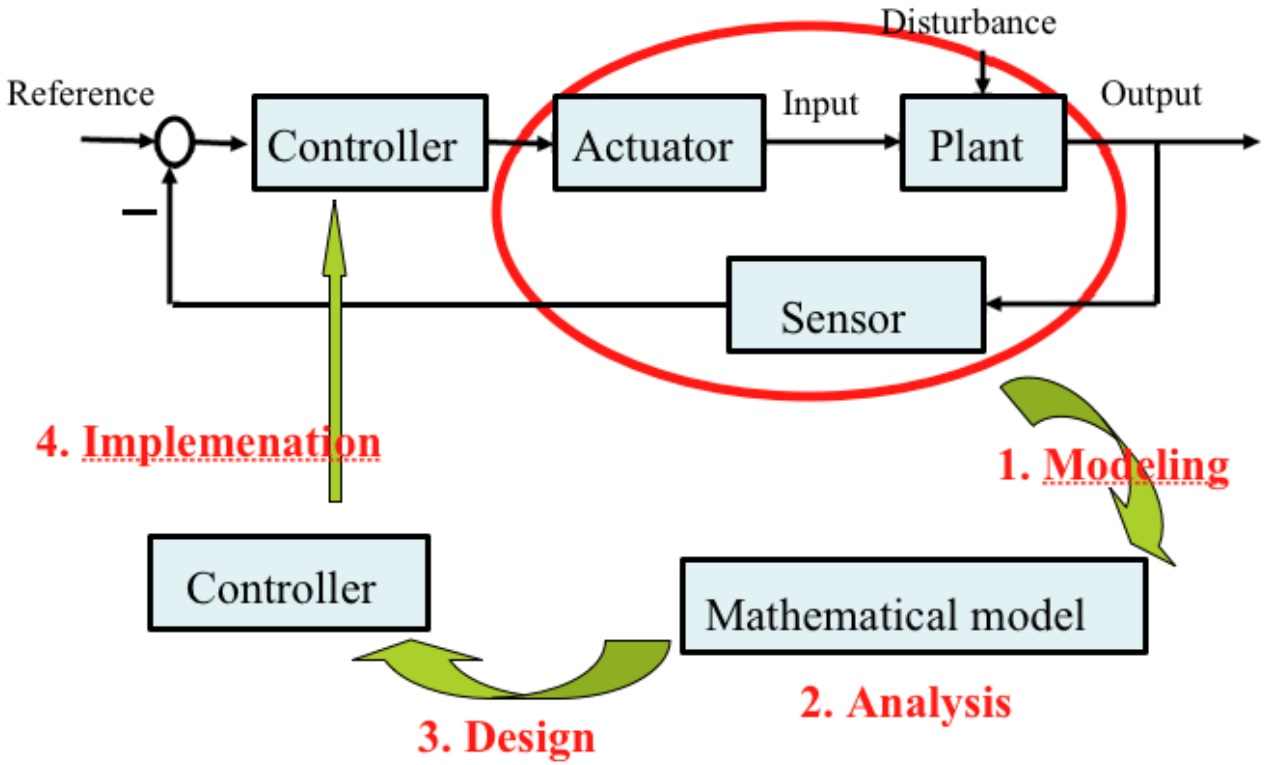
Ability to correct error

High accuracy and resistance of disturbance

Complex structure, high cost

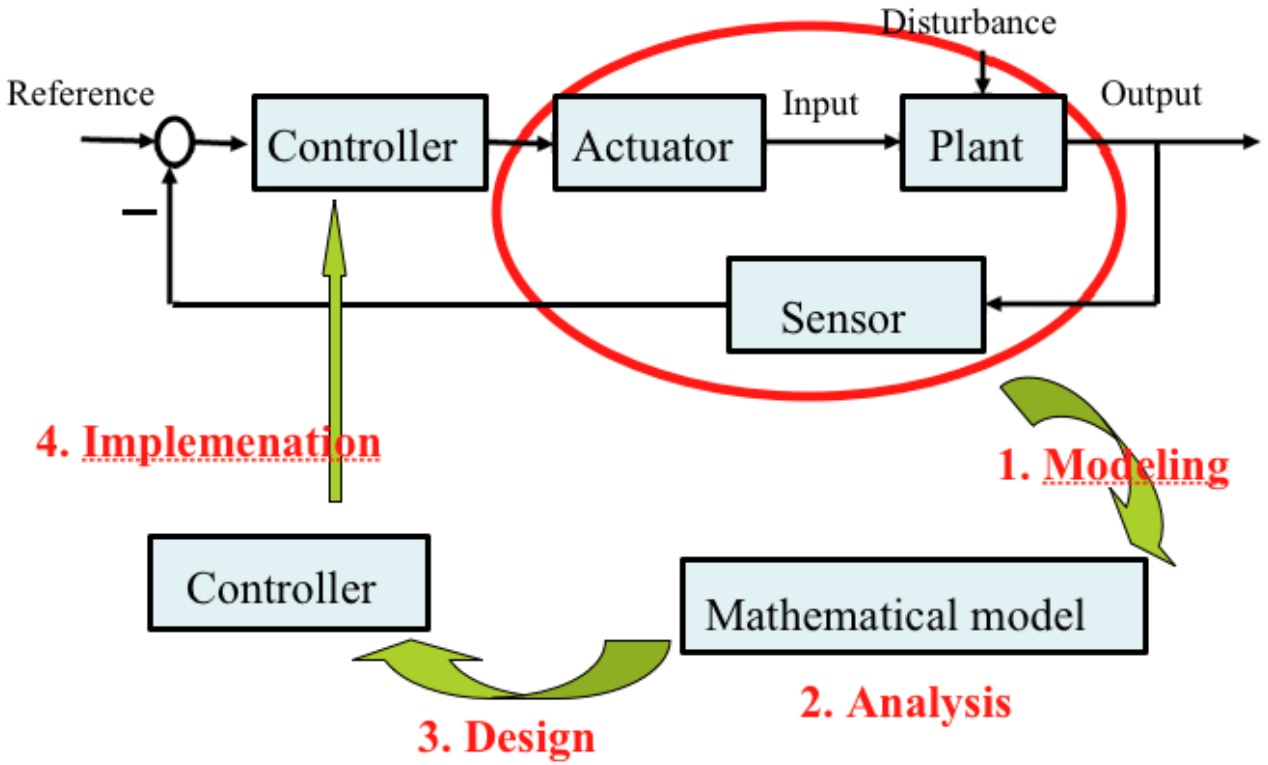
Selecting parameter is critical (may cause stability problem)

Controller design process



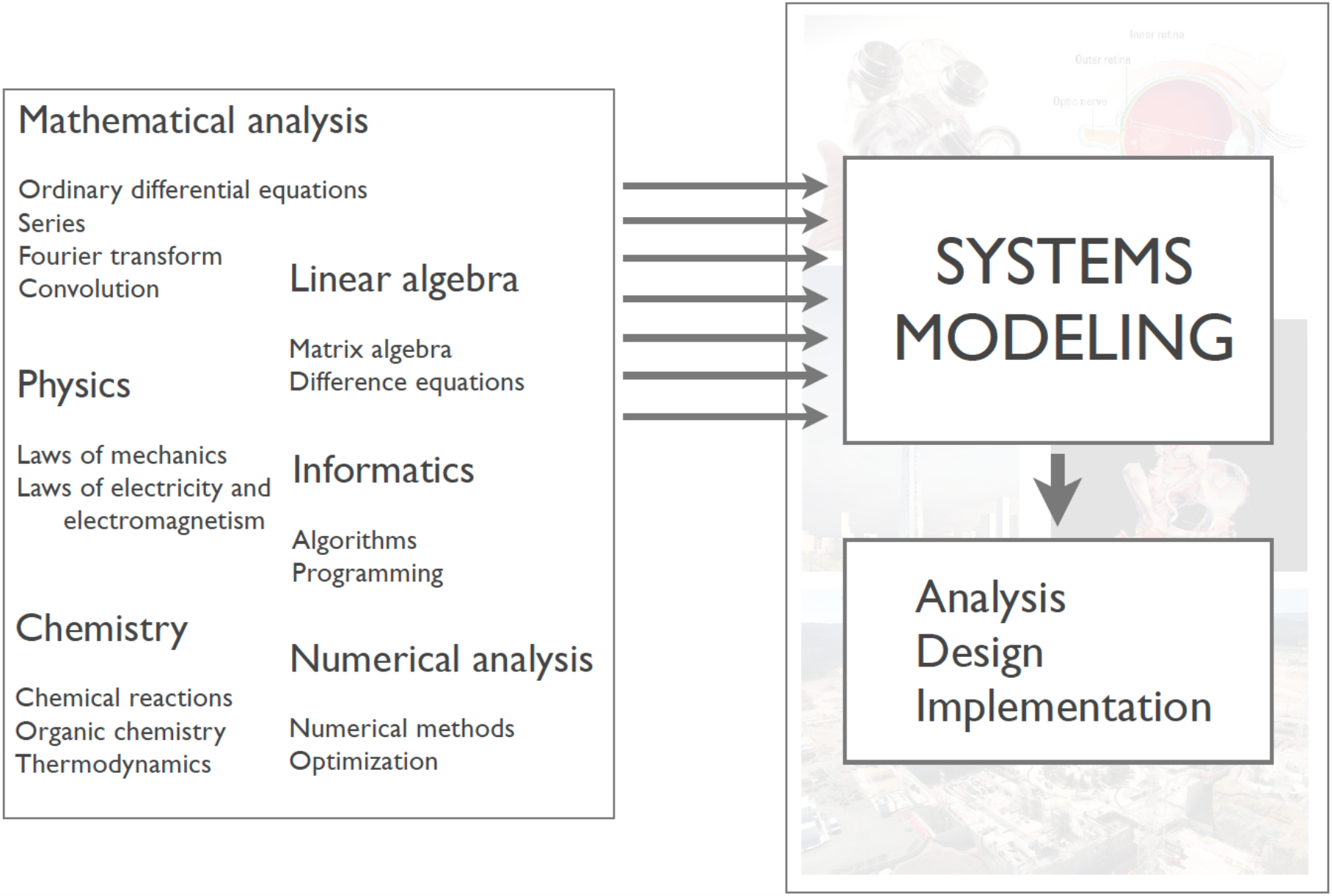
- Design control logic to ensure the dynamics of the closed loop system are stable
  - i.e. bounded disturbances give bounded errors
- Want to achieve desired behavior
  - good disturbance rejection, fast responsiveness to changes in goal, etc.

Controller design process



- Controller design achieved by modeling and analysis techniques that capture the essential physics of the system
- This permit the consideration of possible behaviors in the presence of uncertainty, noise and component failure

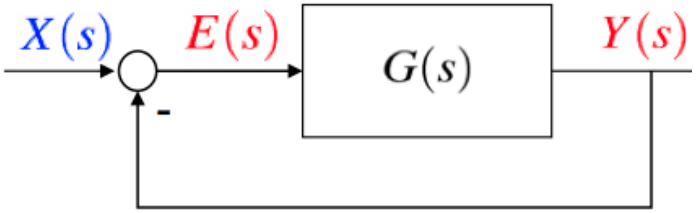
Controller design process



Classical control theory

Modeling systems using transfer function and a block diagrams

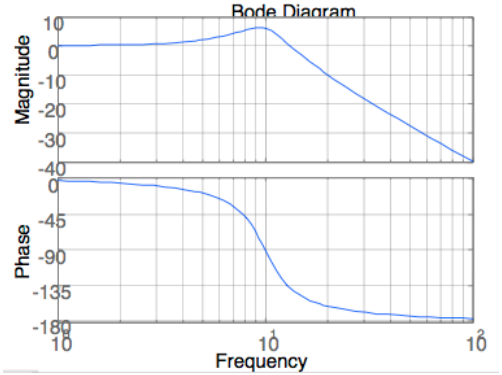
- Frequency techniques based on Laplace transform, Bode Diagrams etc.
- Cam be applied to mechanical, electrical, electromechanical systems, etc.



$$Y(s) = \frac{G(s)}{1 + G(s)} X(s)$$

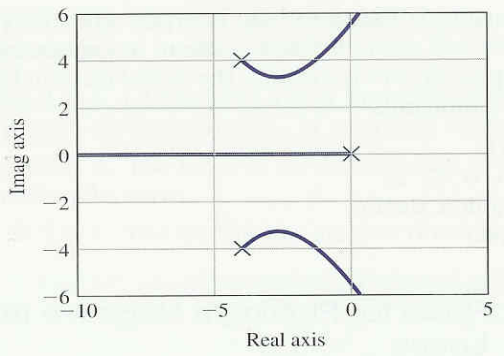
Analysis using:

- Time and Frequency response
- Stability: Routh-Hurwitz criterion, Nyquist criterion, etc

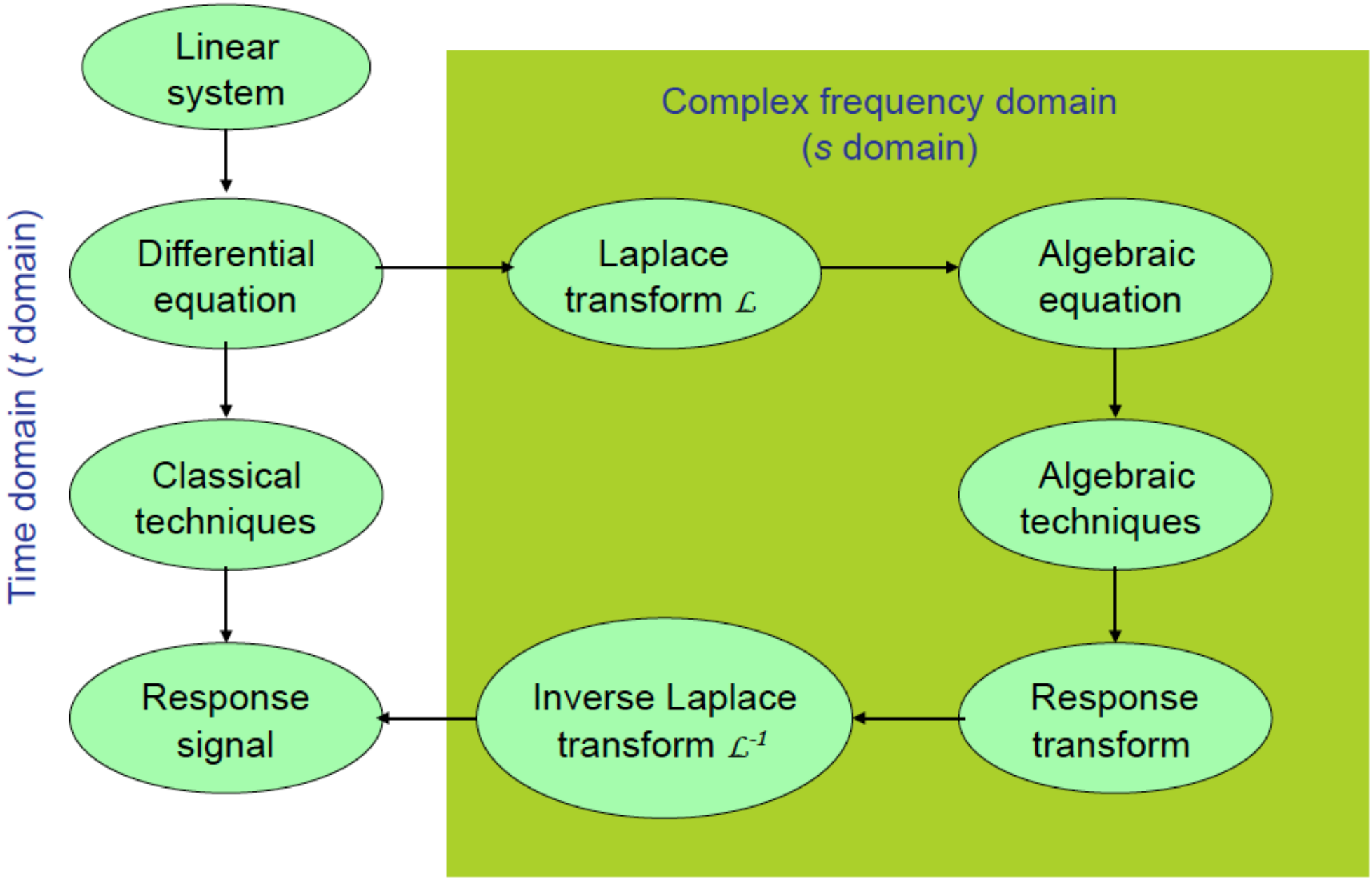


Design using:

- Root locus technique, frequency response techniques
- PID control, lead/lag compensator, etc



Overview of classical control theory



Limitations of classical control theory

Classical control theory deals only with:

- Single-input single-output (SISO) systems
- Linear time-invariant (LTI) systems

But

- No optimality concept in controller design
- No robustness consideration, except gain margin and phase margin
- It can't handle nonlinear and time varying systems!

Therefore

- We need an advanced control theory

System with undesirable cross coupling

Consider system with an undesirable cross coupling:
Two handle mixer taps

- Both input signals affects a given output signal
  - Both output signals affected by/depend on a given input signal
- Hot tap changes temperature and flow rate
  - Cold tap changes both temperature and flow rate

System with independent coupling

- Much easier to control uncoupled systems
- Consider one-handle mixer system with a “nice” cross coupling

- Each input signal now affects (almost) only one output signal
- Each output is now affected (almost) only by one input signal

Interlude

10 minute break

ROCO218: Control Engineering
Dr Ian Howard

Lecture 1

Simple introduction to Matlab

What is Matlab?

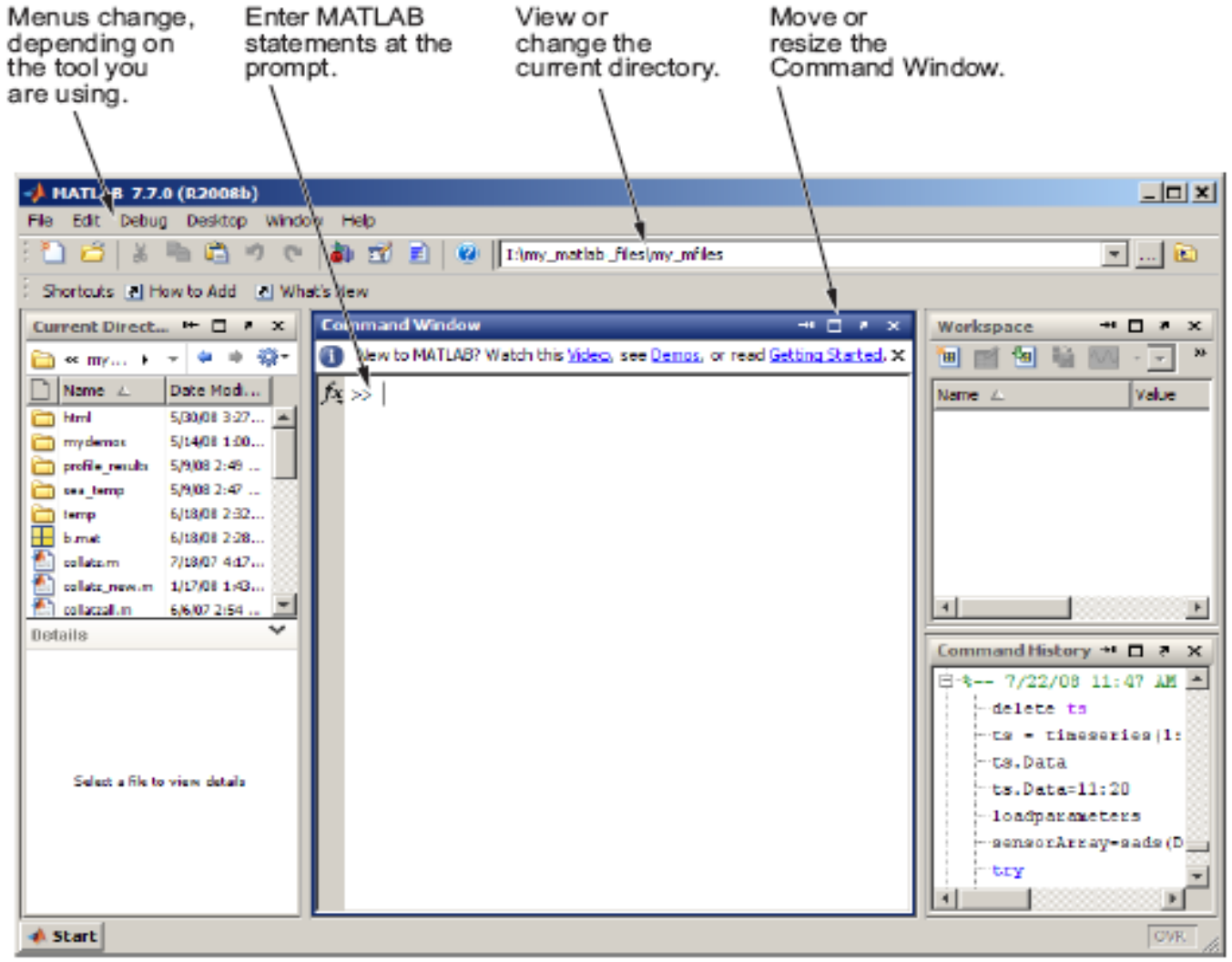
- MATLAB (MATrix LABoratory)
- High-performance language for technical computing, computation, visualization, and programming in an easy-to-use environment
- Typical uses include:
- Math and computation

- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building

Matrix details

- Rows and columns are always numbered starting at 1
- Matlab matrices are of various types to hold different kinds of data (usually floats or integers)
- A single number is a 1 x 1 matrix in Matlab!
- Matlab variables are not given a type, and do not need to be declared
- Any matrix can be assigned to any variable

Main Matlab Window



Building matrices with [ ]

A = [1 3 5]

1	3	5

Rows x columns = 1x3

A = [1; 3; 5;]

1
3
5

Rows x columns = 3x1

A = [2 7 4; 3 8 9]

2	7	4
3	8	9

Rows x columns = 2x3

B = [A A]

2	7	4	2	7	4
3	8	9	3	8	9

Rows x columns = 2x6

Matrices in Matlab

```
% Row vectors
% enclosing  elements in square brackets with space in between
% can setup 1D matrix at once
% setup 1 dimensional  1 x 4 vector
A_ROW = [10 20 30 40];
size(A_ROW)
A_ROW

% can also access elements in matrix individually
% setup 1 dimensional  1 x 4 vector element by element
AA_ROW(1)  = 10;
AA_ROW(2)  = 20;
AA_ROW(3)  = 30;
AA_ROW(4)  = 40;
size(AA_ROW)
AA_ROW
```

```
A_ROW

ans =

     1     4


A_ROW =

    10    20    30    40
```

Matrices in Matlab

```
% Column vectors
% enclosing  elements in square brackets with semicolon in be
B_Column = [10; 20; 30; 40;];
size(B_Column)
B_Column

% can uee transpose operator ' to exhange rows and columns
BB_Column = [10 20 30 40]';
size(BB_Column)
BB_Column
```

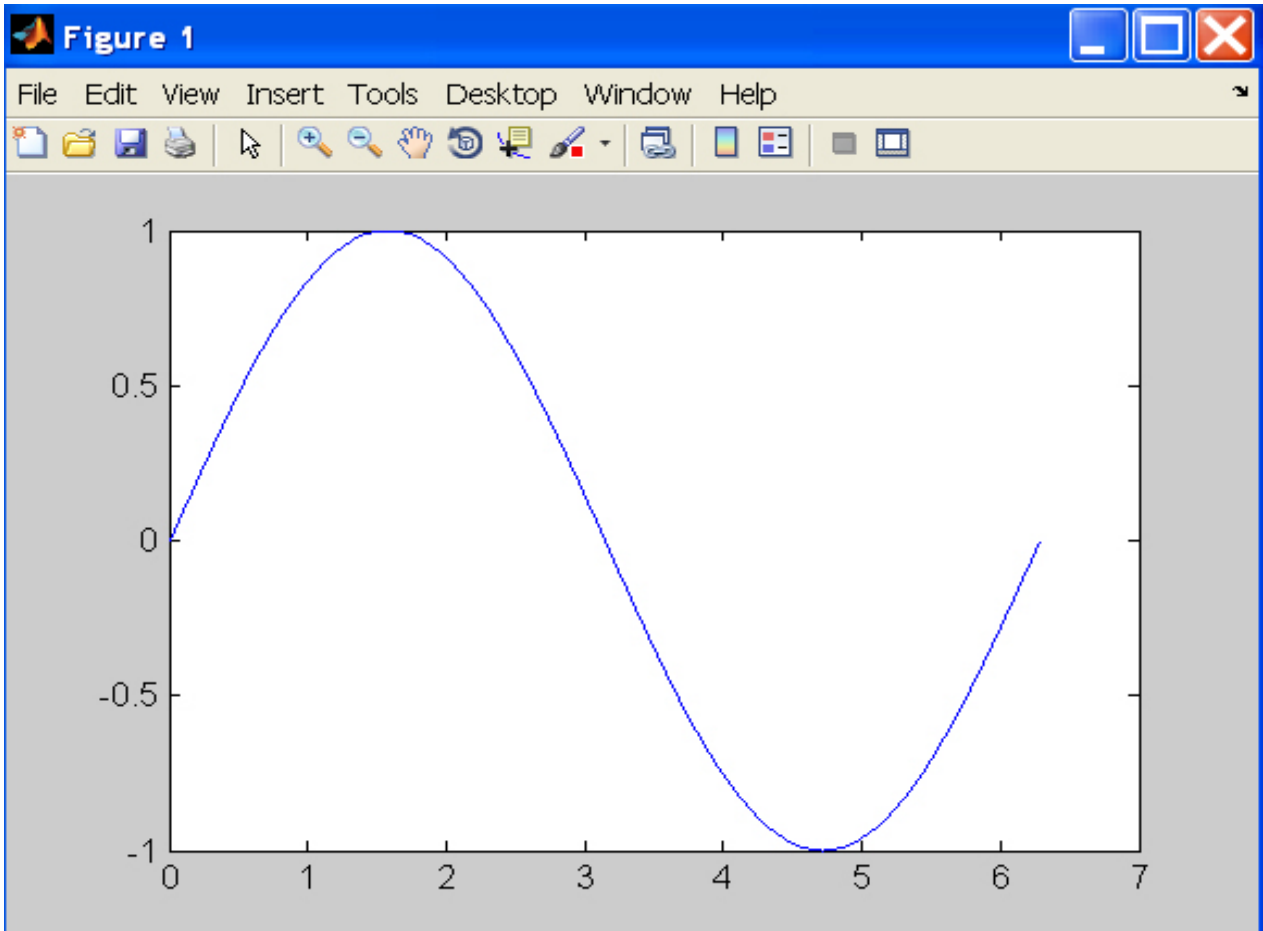
```
B_Column

ans =

     4     1


B_Column =

    10
    20
    30
    40
```

The **transpose** of a matrix A is a matrix where the rows of A become the columns of the new matrix and the columns of A become its rows. For example

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \text{becomes} \quad \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

Plot data

```
t = 0:pi/100:2*pi;
y = sin(t);
plot(t,y);
```

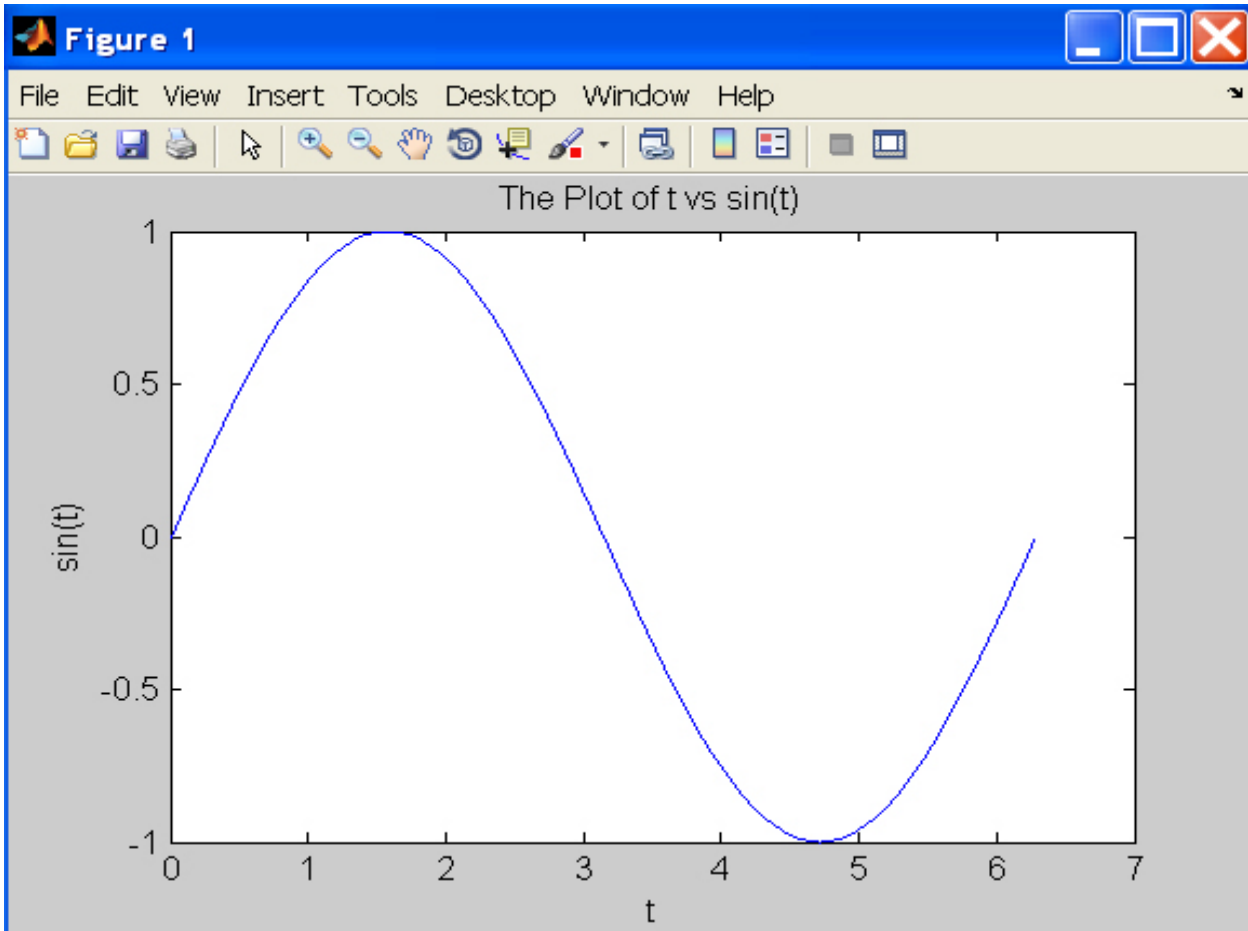


Add title and labels

xlabel('t');

ylabel('sin(t)');

title('The plot of t vs sin(t)');



Adding matrices

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \text{and} \qquad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Add the elements

$$A + B = \begin{bmatrix} 1+\mathbf{5} & 2+\mathbf{6} \\ 3+\mathbf{7} & 4+\mathbf{8} \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

Matlab: A + B

Subtract the elements

$$D = A - B = \begin{bmatrix} 1-\mathbf{5} & 2-\mathbf{6} \\ 3-\mathbf{7} & 4-\mathbf{8} \end{bmatrix} = \begin{bmatrix} -4 & -4 \\ -4 & -4 \end{bmatrix}$$

Matlab: A - B

Multiplication of a matrix by a skalar

In general if A is an $m \times n$ matrix with typical element a_{ij} then the product of a number k with A is written kA and has the corresponding elements ka_{ij}.

$$7A = 7\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 7 & 14 \\ 21 & 28 \end{bmatrix}$$

Matlab: 7 * [1 2; 3 4;]

Use matrices to represent equations

$$\begin{aligned} 3x + 2y - z &= 3 \\ x - y + z &= 4 \\ 2x + 3y + 4z &= 5 \end{aligned}$$

Can write equations as

$$AX = B$$

$$\begin{bmatrix} 3 & 2 & -1 \\ 1 & -1 & 1 \\ 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$$

Where

$$A = \begin{bmatrix} 3 & 2 & -1 \\ 1 & -1 & 1 \\ 2 & 3 & 4 \end{bmatrix} \qquad X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad B = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$$

We will use this a lot when we look at state space representations!

Multiplying row and column matrices

Multiply rows by columns

Matlab: A * B

1×2 row matrix and B be a 2×1 column matrix:

$$A = \begin{bmatrix} a & b \end{bmatrix} \qquad B = \begin{bmatrix} c \\ d \end{bmatrix}$$

$$AB = \begin{bmatrix} a & b \end{bmatrix} \times \begin{bmatrix} c \\ d \end{bmatrix} = [ac + bd]$$

$$\begin{bmatrix} 2 & -3 \end{bmatrix} \times \begin{bmatrix} 6 \\ 5 \end{bmatrix} = [12 - 15] = [-3]$$

Multiplying 2x2 matrices

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} w \\ y \end{bmatrix} & \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} \\ \begin{bmatrix} c & d \end{bmatrix} \begin{bmatrix} w \\ y \end{bmatrix} & \begin{bmatrix} c & d \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} \end{bmatrix} = \begin{bmatrix} aw + by & ax + bz \\ cw + dy & cx + dz \end{bmatrix}$$

$$\begin{bmatrix} 2 & -1 \\ 3 & -2 \end{bmatrix} \times \begin{bmatrix} 2 & 4 \\ 6 & 1 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 2 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ 6 \end{bmatrix} & \begin{bmatrix} 2 & -1 \end{bmatrix} \begin{bmatrix} 4 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 3 & -2 \end{bmatrix} \begin{bmatrix} 2 \\ 6 \end{bmatrix} & \begin{bmatrix} 3 & -2 \end{bmatrix} \begin{bmatrix} 4 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} -2 & 7 \\ -6 & 10 \end{bmatrix}$$

Multiplying 3x3 matrices

$$C = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} r & s & t \\ u & v & w \\ x & y & z \end{bmatrix} = \begin{bmatrix} ar + bu + cx & as + bv + cy & at + bw + cz \\ dr + eu + fx & ds + ev + fy & dt + ew + fz \\ gr + hu + ix & gs + hv + iy & gt + hw + iz \end{bmatrix}$$

Multiplying non-square matrices

The general rule is that an $n \times p$ matrix A can be multiplied by a $p \times m$ matrix B to form an $n \times m$ matrix $AB = C$.

$$AB = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 5 \\ 6 & 1 \\ 4 & 3 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 6 \\ 4 \end{bmatrix} & \begin{bmatrix} 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} 5 \\ 1 \\ 3 \end{bmatrix} \\ \begin{bmatrix} 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ 6 \\ 4 \end{bmatrix} & \begin{bmatrix} 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 5 \\ 1 \\ 3 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 22 & 13 \\ 38 & 25 \end{bmatrix}$$

Matlab: A * B

Matlab matrix multiplication

Matrix multiplication

A = [1 2 ; 4 5]

B = A * A calculates

$$\begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 9 & 12 \\ 24 & 32 \end{bmatrix}$$

Matrix element wise multiplication

B = A .* A calculates

$$\begin{bmatrix} 1\times 1 & 2\times 2 \\ 4\times 4 & 5\times 5 \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 16 & 25 \end{bmatrix}$$

Inverse of a 2x2 matrix

For a 2x2 matrix $$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Its determinant: $$\det(A) = \left| \begin{pmatrix} a & b \\ c & d \end{pmatrix} \right| = (ad - bc)$$

Matlab: det(A)

Inverse: $$A^{-1} = \frac{1}{\det(A)} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} = \frac{1}{(ad - bc)} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Matlab: inv(A)

For loop and if statement

```
% simple for loop for i = 1, 2, 3, 4, 5
for i = 1:5
    disp(i);
end
```

Output:

```
1
2
3
4
5
```

```
% fancier output printing for loop for i = 1, 2, 3, 4, 5
for i = 1:5
    disp(sprintf('i = %d', i));
end
```

Output:

```
i = 1
i = 2
i = 3
i = 4
i = 5
```

```
% simple if in for loop for i = 1, 2, 3, 4, 5
% print value when i = 3
for i = 1:5
    if(i==3)
    disp(sprintf('found value: i = %d', i));
    end
end
```

Output:

```
found value: i = 3
```

Script m-file

When you have a group of commands that are expected to be executed repeatedly it is convenient to save them in a file.

Here we save previous example as forif.m.

A script shares the same scope with that which it operates.

Editor - /Users/ihoward/Documents/private/MyWork/matlab/ROCO218 Control engineering 2016/2017/forIf.m

```
1
2     % simple for loop for i = 1, 2, 3, 4, 5
3 -   for i = 1:5
4 -       disp(i);
5 -   end
6
7     % fancier output printing for loop for i = 1, 2, 3, 4, 5
8 -   for i = 1:5
9 -       disp(sprintf('i = %d', i));
10 -  end
11
12
13    % simple if in for loop for i = 1, 2, 3, 4, 5
14    % print value when i = 3
15 -  for i = 1:5
16 -      if(i==3)
17 -      disp(sprintf('found value: i = %d', i));
18 -      end
19 -  end
20
```

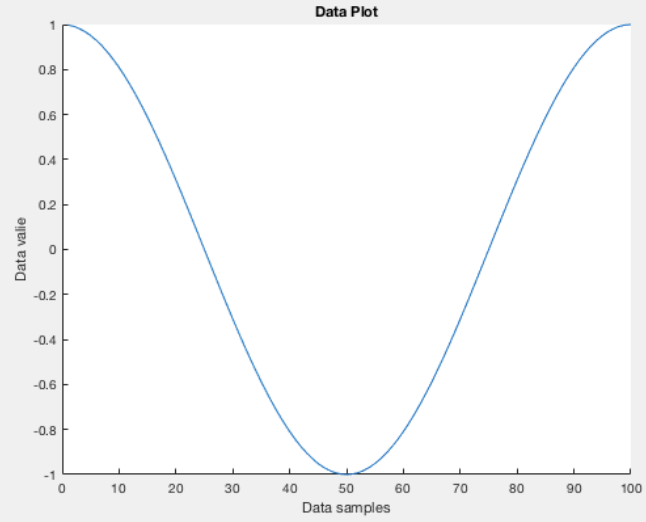
Function m-files

- Can build function in Matlab
- Recommendation you save function in file with filename same as function name
- It can then be called from a script, another function, or on command line:

Define function

```
function [ dataSig ] = GetData( length )
% return dataSig containing 'length' samples
% of cosine waveform

% compute set of angles
angles = 2 * pi * (1:length)/length;

% get cos of angles
dataSig = cos(angles);

end
```

Output figure



Call function from script

```
% data length
length = 100;

% generate some data using function
data = GetData(length);


figure
hold on
plot(data);
title('Data Plot');
xlabel('Data samples');
ylabel('Data valie');
```

Some Frequently Used Functions

magic(n) % creates a special *n x n* matrix; handy for testing
zeros(n,m) % creates *n x m* matrix of *zeros* (0)
ones(n,m) % creates *n x m* matrix of *ones* (1)
rand(n,m) % creates *n x m* matrix of random numbers
repmat(a,n,m*)* *%* replicates *a* by *n* rows and *m* columns
diag(M) % extracts the diagonals of a matrix M
help elmat % list all elementary matrix operations (or *elfun*)
abs(x); % absolute value of *x*
exp(x); % *e* to the x-th power
fix(x); % rounds x to integer towards 0
log10(x); % common logarithm of x to the base 10
rem(x,y); % remainder of x/y
mod(x, y); % modulus after division – unsigned rem
sqrt(x); % square root of x
sin(x); % sine of x; x in radians
acoth(x) % inversion hyperbolic cotangent of x

Some Frequently Used Functions

Generally, MATLAB's default graphical settings are adequate which make plotting fairly effortless. For more customized effects, use the *get* and *set* commands to change the behavior of specific rendering properties.

```
h= plot(1:5)                % returns the handle of this line plot
get(h)                      % to view line plot's properties and their values
set(h, 'lineWidth')         % show possible values for lineWidth
set(h, 'lineWidth', 2)      % change line width of plot to 2
gcf                 % returns current figure handle
gca                 % returns current axes handle
get(gcf)            % gets current figure's property settings
set(gcf, 'Name', 'My First Plot')       % Figure 1 => Figure 1: My First Plot
get(gca)            % gets the current axes'  property settings
figure(1)           % create/switch to Figure 1 or pop Figure 1 to the front
clf                 % clears current figure
close               % close current figure; "close 3" closes Figure 3
close all           % close all figures
```

Further information

- No point of me reinventing the wheel
- Loads of documentation available on using Matlab
- Have a look yourselves

- Matlab get started guide
  -http://www.mathworks.co.uk/help/pdf_doc/matlab/getstart.pdf

- Book
  -J.M. Maciejowski, Getting started with MATLAB.