Introduction to Mechatronics

Mechatronics is a multi-disciplinary field that combines several engineering disciplines including **Mecha**nical Engineering, Elec**tronics**, Communications, Computer Engineering, and Control Systems.

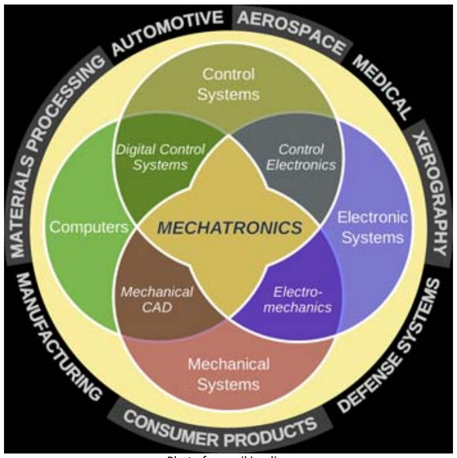


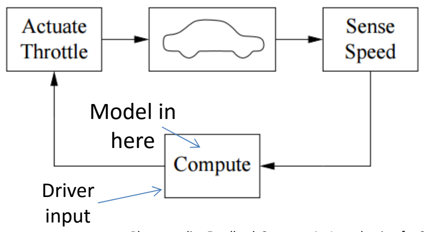
Photo from wikipedia

Mechatronic System Examples

- Some common examples of mechatronic systems include:
  - Robots (Manufacturing, medical surgery, space exploration, rescue missions, etc.)
  - Automotive systems (Cruz control, electronic stability control, anti-lock braking system, engine control unit, automated driving, etc.)
  - Manufacturing systems (e.g., CNC machines, 3D printers)
  - Heating, ventilation, air conditioning, and cooling systems
  - Chemical plants (Pressure, temperature, and fluid-level and flow-rate control)
  - Power plants (Steam and gas turbine control)
  - Elevators, cranes, segways, airplanes, hard disk drives, microscopes, and so forth.

Modeling and Control of Mechatronic Systems

- An important part of designing high-performance mechatronic systems is developing representative mathematical models for the system.
- A model-based control design will deliver required performance if (1) the model is accurate enough, and (2) the controller is designed methodically, and (3) the controller is robust with respect to uncertainties.



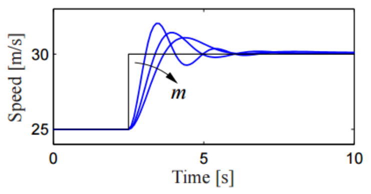
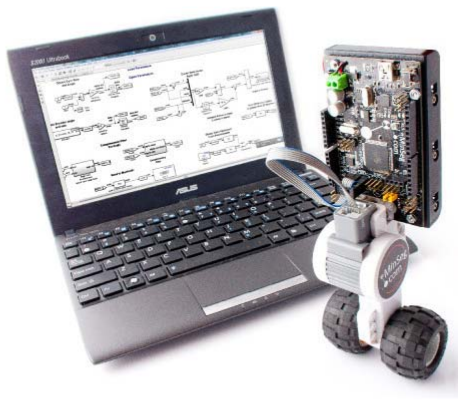


Photo credit : Feedback Systems: An Introduction for Scientists and Engineers , Åström, K. and Murray, R. (2012).

Main Platform for the Lab Experiments

- This semester, we will use a mini segway educational robot developed at Rensselaer Polytechnic Institute and sold by minseg (minseg.com). The robot includes:

  - Arduino Mega MCU
  - 3 DoF Accelerometer
  - 3 DoF Gyroscope
  - Magnetometer
  - Lego NXT DC motor
  - Motor driver
  - Encoder
  - Bluetooth module
  - Simulink support package
  - Educational labs



Learning Through MinSeg

- Through the MinSeg platform, we will learn:
  - How to interface the robot with Simulink
  - How to read data from the accelerometer, gyroscope, and the encoder
  - How to communicate data via bluetooth
  - How to model the robot mathematically
  - How to simulate the robot in Simulink
  - How to identify parameters of the robot
  - How to control the DC motor position
  - How to design a controller for the robot to stabilize it in the upright position
  - And ultimately, how to steer the robot between from one position to another.

MATLAB and Simulink

- Most of the lab assignments will be done in Simulink.
- We will simulate the dynamic system models in Simulink.
- We will be sending the data from Simulink to MATLAB for post-processing and plotting.
- We will analyze the properties of the mechatronic system models and controllers in MATLAB.
- We will develop system identification algorithms in MATLAB.
- Most tech companies use MATLAB for their research and development activities.

=> Building expertise in MATLAB and Simulink is a critical goal and a main requirement of this course.

MATLAB

- Product of Mathworks, Founded in 1984, mainly for control analysis and signal processing applications. A brief history of MATLAB is available at: http://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html
- Matlab is a high-level programming language, which is easy to learn and develop sophisticated programs.
- Contains a large number of functions and toolboxes for almost every scientific application.
- Complete documentation and a great online support community.

MATLAB Basics

- Create array variables:

```
>> x = [1 2 3] or x = [1, 2, 3]   Creates a row vector
>> x = [1; 2; 3]      Creates a column vector)
>> A = [1 2; 3 4]    Creates a 2×2 matrix
>> A = ones(2,3)    Creates a 2×3 matrix of ones
>> x = zeros(1,3)   Creates a 1×3 zero vector
>> A = eye(5)         Creates a 5×5 identity matrix
>> A = ones(3,3,5) Creates a 3×3×5 3-D matrix
>> A = rand(2,3)   Creates a 2×3 matrix of random nums.
>> y = 'abcd'          Creates a variable in char format
>> y = ['abcd' 'efgh']  Concatenates the elements:
      Creates: abcdefgh
```

MATLAB Basics

- Accessing elements of variables:

  >> x(5)   Gets the fifth element of vector x

  >> A(3,4) Element on the 3rd row and 4th col. Of matrix A

  >> A(3,4,6,10) Element on the 3rd row, 4th col., 6th third dimension, and 10th fourth dimension

  >> A(3,:)   Gets the entire 3rd row

  >> A(:,4)   Gets the entire 4th column

  >> A([3 6],:) Gets row 3 to 6 of matrix A

  >> A(end,[1:end-1]) Gets the last row, from first col. to one to the last column

  >> x(-1) Error (Array indices must be positive integers)

  >> A(0,2) Error

MATLAB Basics

- Modifying Elements of Arrays:

  >> x(10) = 0   Changes the 10th element of x to zero

  >> A(2,3) = 10 Sets the elm. on 2nd row & 3rd col. to 10

  >> A(10,:) = []   Clears the 10th row of matrix A

  >> A(3,5) = [] Error (We can only clear rows or columns)

  >> x = [x 3]   Adds a new element to x at the end
- Basic functions

  >> A(:)   Converts A to a column vector of matrix A

  >> size(A)   Returns the dimension of A

  >> size(A,1) Number of rows of A

  >> size(A,2) Number of columns of A

  >> A’   Returns the transpose of A

MATLAB Basics

- Basic functions

```
>> inv(A)   Returns inverse of matrix A
>> max(x)  Returns the largest element of vector x
>> min(x)  Returns the smallest element of vector x
>> sum(x)  Returns the cumulative sum of vector x elements
>> abs(x)   Returns absolute valve of x
>> log(x)    Returns natural logarithm of x
>> log10(x) Returns the base 10 logarithm of x
>> sin(x), cos(x), tan(x), Sine, cosine, and tan. of x in rad.
>> sind(x), cosd(x), tand(x)  sine, cosine, tan of x in deg.
>> exp(x)    Exponential of x
>> floor(x)  Rounds to the nearest integer toward -infinity
>> round(x) Rounds x to closest integer
>> length(x) Returns length of vector x
>> numel(A) Returns number of elements of A
>> size(A)     Returns dimensions of A
>> num2str(x) Converts numeric x to string (reverse: str2num)
>> disp(string)  Displays the string on the command window
```

MATLAB Basics

- Basic Operations

```
>> A + B  Elementwise summation of compatible matrices
>> A - B   Elementwise subtraction of compatible matrices
>> A*B    Matrix multiplication
>> A/B     Same as A*inv(B)
>> A\B      Same as inv(A)*B
>> x*y      Error (if x and y are vectors of same size, N)
>> x*y'     x(1)*y(1)+ x(2)*y(2)+ … +x(N)*y(N)
>> x.*y     Elementwise multiplication [x(1)*y(1) … x(N)*y(N)]
>> x./y      Elementwise division [x(1)/y(1) … x(N)/y(N)]
>> x^2      Error (if x is a vector of more than 1 elements)
>> x.^2     Same as x.*x
>> x.*exp(y)   [x(1)*exp(y(1)) … x(N)*exp(y(N))]
```

MATLAB Basics

- Logical operators and comparisons

&	AND
|	OR (Keyboard: Shift + \)
&&	Short-circuited AND
II	Short-circuited OR
~	NOT
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
~=	Not equal to

MATLAB Basics

- Find, any, isnan, isempty, etc.

```
>> find(x > 10 & x < 20) => Returns indices of x whose
      corresponding x values are between 10 and 20
>> x(find(x < 0))  => Returns the negative values in vec. x
>> find(x == 0) => Returns indices corresponding to x
                            value of zero
>> x(find(x > 10, 3)) => Returns the first three elements of
                                 x whose values are larger than 10
>> any(x)  => Checks if any elements of x are
                  nonzero
>> isempty(x) => Checks if vector x has any elements
>> isnan(x) => Checks if any elements of x are NaNs (NaN
                     stands for "Not a Number")
```

MATLAB Basics

- Operator Precedence

```
>> A+B*C   is same as A+(B*C)   => Mult. precedes sum.
>> 2^5*10 is same as (2^5)*10  => Power precedes mult.
>> A < B + C   is same as A < (B+C) => Sum. precedes <, >, etc.
>> A & B + C  is same as A & (B + C) => Sum. precedes log. op.
```

Check Mathworks Website for full list of operator precedence
http://www.mathworks.com/help/matlab/matlab_prog/operator-precedence.html

MATLAB Basics

- Conditions

```
>> if A > B
>>    s = 1;
>> else
>>    s = 0;
>> end

>> if A < B
>>    s = 0;
>> elseif A==B
>>    s = 1;
>> else
>>    s = 2;
>> end

>> if A       same as if A == 1, or if A == 'true'
>>    s = 1;
>> end

>> if strcmp(strVar, 'hello')     for string comparison.  strVar == 'hello'  will give error
```

MATLAB Basics

- Loops

```
>> s = 0;
>> for i = 1:5:100
>>      s = s + 1;
>> end

>> i = 1;
>> s = 0;
>> while i < 100
>>     s = s + 1;
>>     i = i + 5;
>> end
```

MATLAB Basics

- Functions

Functions are widely used to simplify code development. To create a function, you need to open a new script, and save the function in the working directory. You can call the function in the main script or the command line.

Examples:

```
function outVar = functionName(inVar)
      outVar(1) = inVar(1) + inVar(2);
      outVar(2) = inVar(1)*inVar(2);
end
>> x = [2 3];
>> y = functionName(x)

function [outVar1 outVar2] = functionName(inVar1, inVar2)
      outVar1 = inVar1 + inVar2;
      outVar2 = inVar1*inVar2;
end
>> [y1 y2] = functionName(2,3)
```

MATLAB Basics

- Functions
  - Save the function in the working directory. Choose the filename same as the function name.
  - You cannot access the workspace variables directly inside the function unless they are global or input variables.
  - You can use "evalin" command to access workspace variables inside a function:

    x = evalin('base', 'ws_x') Assigns the value of ws_x from the base workspace to variable x inside the function.
  - Variables created inside the function are not dumped into workspace unless they are output or global variables.
  - You can use "assignin" command to send variables from a function into the workspace :

    assignin('base', 'ws_x', x) Assigns the value of x from the function to variable ws_x in the base workspace.

  Check Mathworks for more information on "evalin" and "assignin" commands.

MATLAB Basics

- Saving variables from workspace

>> save fileName Saves all variables in the workspace in a file with .mat extension.

>> save fileName varName1 varNam2 Saves selected vars.

>> save('C:\MyFiles\fileName.mat','x','y') Saves variables x and y in the specified directory

- Loading variables from a file to workspace

>> load fileName Loads variables inside the fileName to workspace

>> load('C:\MyFiles\fileName.mat') Loads variables to workspace from a specified file location

MATLAB Basics

- Plotting data

  >> plot(t,x)   Plots x vs. t with a blue line

  >> plot(t,x,‘r’)   Plots x vs. t with a red line

  >> plot(t,x,‘g’,t,y) Plots x vs. t with a green line and y vs. t with a blue line

  >> plot(t,x,‘k--’,‘linewidth’, 3) Plots a black dashed line with

  >> figure; plot(t,x)   Opens a new figure for the plot, otherwise plots on the most recent figure

  >> plot(t,x); hold on; plot(t,y) Holds on the plot and plot the second data on the first plot. >> hold off   (for holding it off)

  >> help plot   (>>Help anyfunctionName displays information on the function on the command window)

  >> doc plot   (>>doc anyfunctionName   Open a new help document explaining the function in more detail)

MATLAB Basics

- Plotting data

  >> xlabel(‘Label of x-Axis’) To include a label on the x-axis

  >> ylabel(‘Label of y-Axis’) To include a label on the y-axis

  >> title(‘Title of the Plot’)   To include a title on the plot

  >> grid   shows a grid on the plot

  >> axis([XMIN XMAX YMIN YMAX])   Limits the plot window

  >> figure; subplot(2,3,1); plot(t,x) Creates a new figure of 2-by-3 plot matrix, and plots x vs t on the 1st subplot.

  >> bar(x,y)   Plots a bar chart of y vs x

  >> legend('dataName','location','SouthEast') Creates a legent for the data on the south east corner. (Check >>help legend)

  >> close   closes the most recent figure

  >> close all   closes all the figures in running MATLAB window

MATLAB Basics

- Plotting data

  ```
  >> plotyy(x1,y1,x2,y2)    Plots x1,y1 with y1 on the left
                            axis, and x2, y2 with y2 on the right axis.
  >> plot3(x,y,z)    Plots a trajectory in 3D space
  >> surf(x,y,Z)     Plots a surface map of Z based on x and y
                     coordinates. Z is a matrix of compatible
                     size with x and y.
  ```