Matlab 1: User Interface

Cheng-Hsin Hsu

National Tsing Hua University

Department of Computer Science

Slides are based on the materials from Prof. Roger Jang

What is Matlab

- Matlab stands for MATrix LABoratory
- It was first released by Mathworks in 1984
- A programming language for
  - Matrix manipulations
  - Plotting for visualization
  - Implementation of algorithms
  - User interfaces
  - Integration with other languages, including C/C++, Java, Python, and Fortran

History of Matlab

- Prof. Cleve Moler, at University of New Mexico, started developing Matlab in 1980's
- Goal was to allow people to use LINPACK and EISPACK without knowing Fortran

The authors of LINPACK:
Jack Dongarra, Cleve Moler, Pete Stewart, and Jim Bunch in 1978.

Cleve Moler

Commercialization

- John Little rewrote Matlab in C and funded Mathworks in 1984
- Switch to LAPCK in 2000
- Huge community, check Mathwork Central

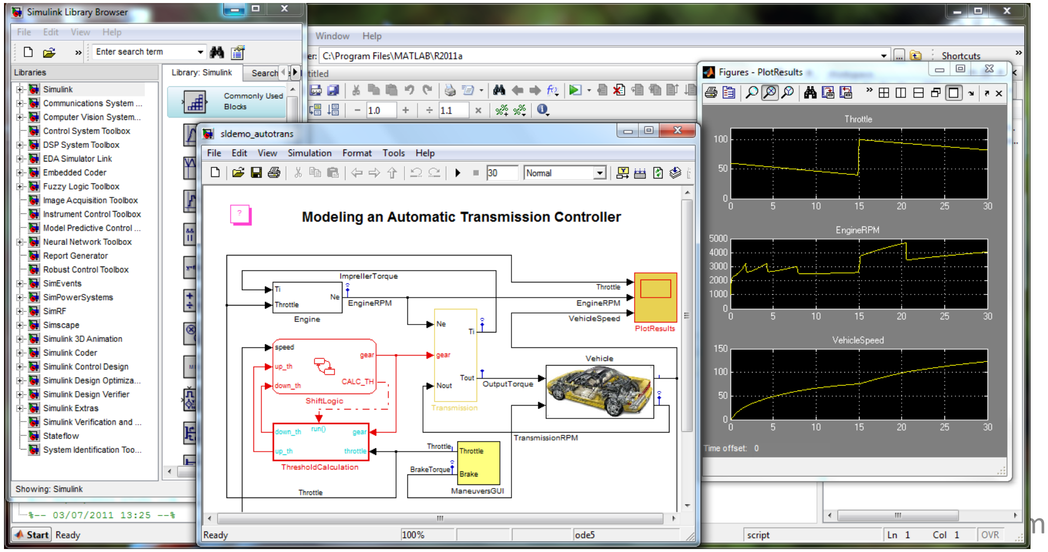
Jack Little

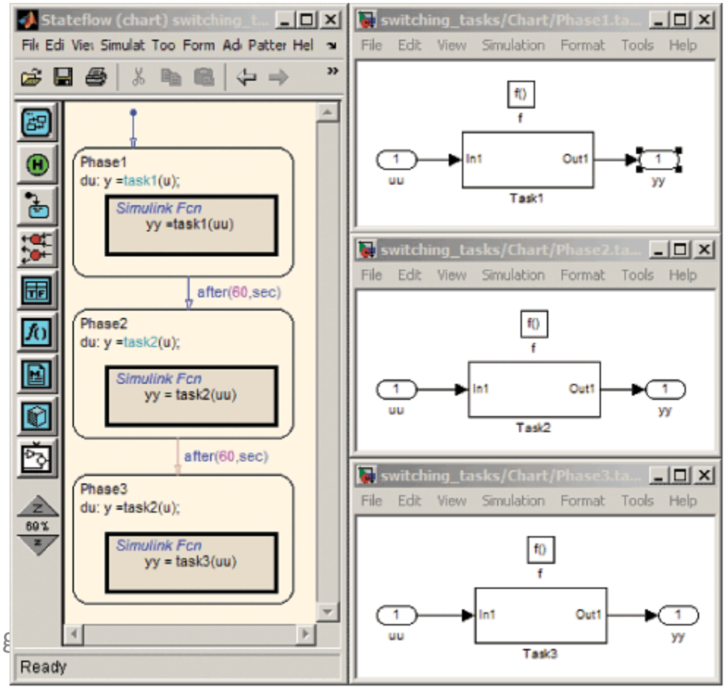
Evolution of Matlab

- Matlab is the dominating numerical computing environment, and can be extended for symbolic computing
- Initially designed for matrix computation
  - Version 4 introduces graphic handles
  - Version 5 different data types/arrays
- Core matlab can be extended by various toolboxes ← sold separately

Simulink and Statflow

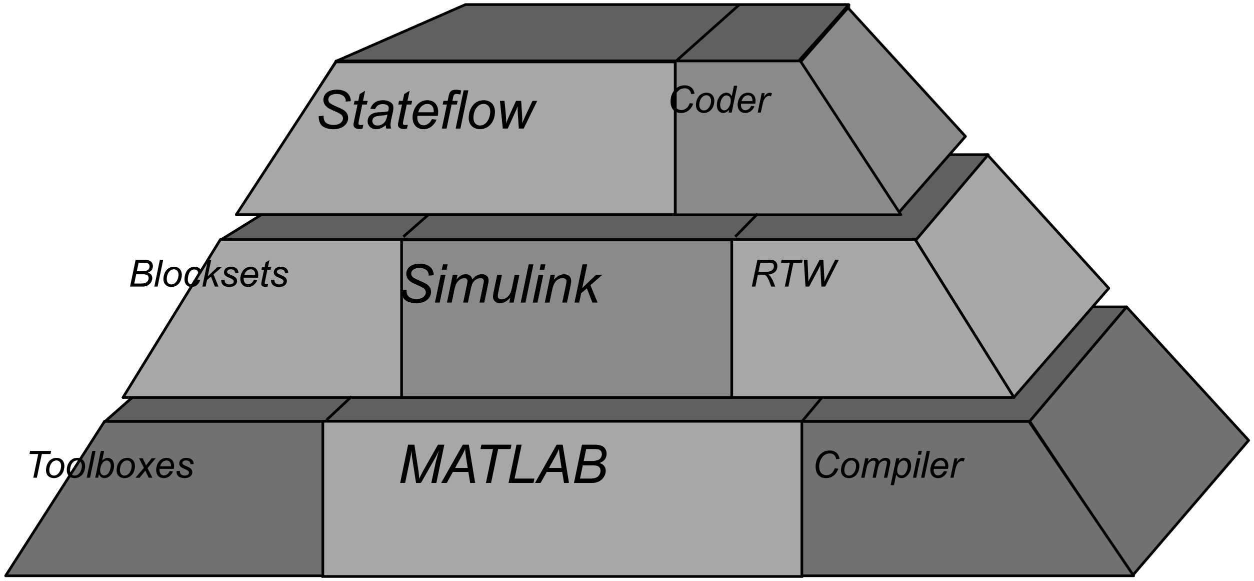
- **Simulink**: discrete- or continuous-time dynamic systems
- **Stateflow**: finite-state machines and event-driven systems





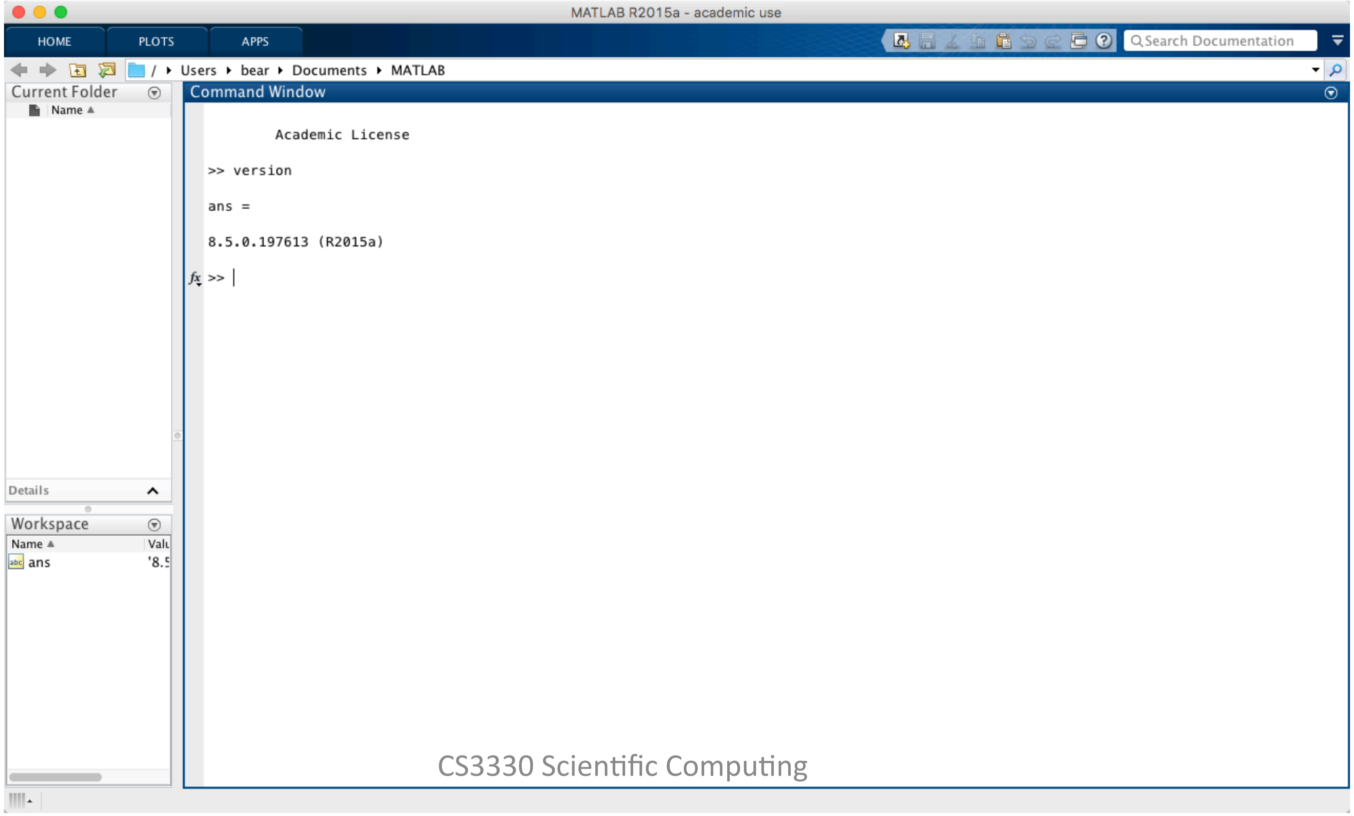
Matlab, Simulink, and Stateflow

- Combining them allow us to carry out diverse tasks, ranging from complex system simulations to integrated-circuit design



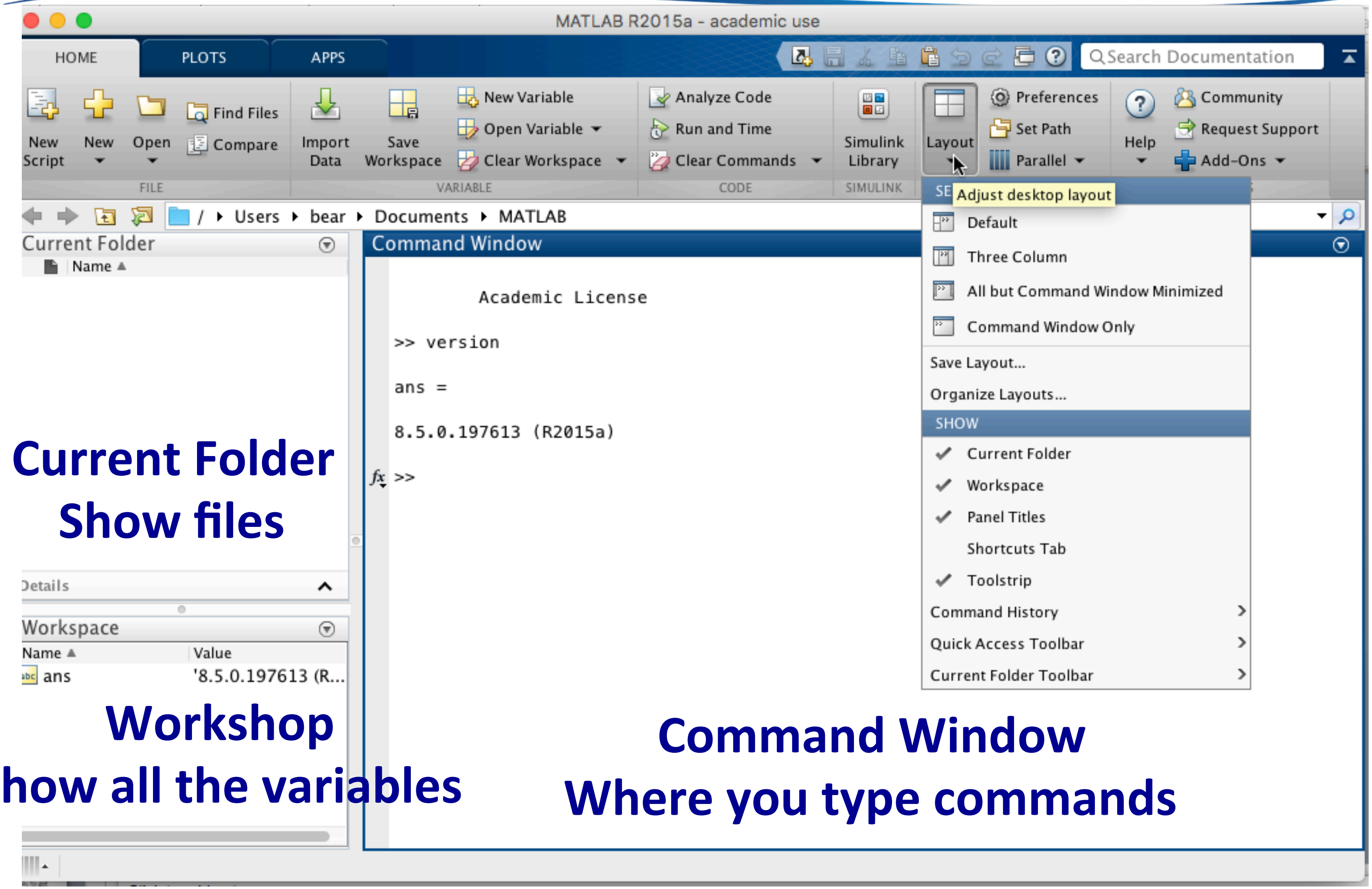
Appearance of Matlab

- Matlab 8.5 (2015a) was released in Mar 2015
- Use spolight to launch it, or find it in Finder → Applications



```
        Academic License

>> version

ans =

8.5.0.197613 (R2015a)

>> |
```

Windows and Layout



```
      Academic License

>> version

ans =

8.5.0.197613 (R2015a)

>>
```

Current Folder
Show files

Workshop
Show all the variables

Command Window
Where you type commands

Matlab Commands for Fun

- Similar to SageMath, you type commands in the command window, and will get immediate responses
- Try
  - version
  - ver ← What is the difference from version?
  - bench

Arithmetic Operations and Variables

- After the prompt (>>), type math formula and press enter

```
>> (5 * 3.5) / pi
ans = 5.5704        ← a builtin variable, see workspace
```

- Use equal (=) to create or update a variable

```
>> x=3/5
x = 0.6000
>>
```

- Add a semicolon (;) at the end of each line to supress the answer

```
>> y=4/6;
>>
```

Naming Policy of Variable

- The first character must be an English letter, followed by letters, numbers, or underscore
- The variable names must be < 64 characters ← truncated if otherwise
- Variables are used without declaration, and by default they are 8-byte double

```
>> whos x
  Name      Size                Bytes  Class     Attributes
  x         1x1                     8  double
```

Comments

```
function y = mean(x,dim,flag,flag2)
%MEAN   Average or mean value.
%   S = MEAN(X) is the mean value of the
elements in X if X is a vector.
%   For matrices, S is a row vector containing
the mean value of each
%   column.
%   For N-D arrays, S is the mean value of the
elements along the first
%   array dimension whose size does not equal 1.
%
.....
```

Vectors and Matrices

- Variables can also be vectors and matrices

```
>> s = [1, 2, 3, 5];
>> s * 2.5 / 12

ans = 0.2083    0.4167    0.6250    1.0417
```

Matrix Operations

- Update a matrix element
- Append one more element
- Delete an element

```
>> s(2)=999
s = 1   999     3     5
>> s(5)=123
s = 1   999     3     5   123
>> s(2)=[]
s = 1     3     5   123
```

2-Dimensional Arrays

- To create a 2-D array, add a semicolon (;) after each row

```
>> a = [1, 2, 3; 4, 5, 6]
a =
     1     2     3
     4     5     6
```

2-Dimensional Array Operations

- Update a specific array element

```
>> a(2, 1)
ans = 4
>> a(2,1)=999
a =         1     2     3
          999     5     6
```

- Store a row of an existing array and store it in a different variable

```
>> b = a(2,1:3)
b = 999     5     6
```

2-Dimensional Array Operations (cont.)

- Combine two arrays, notice the ;

```
>> c=[a;b*2]
c =        1           2           3
         999           5           6
        1998          10          12
```

- Remove the second column, : means whole column (or row)

```
>> c(:,2)=[]
c =        1           3
         999           6
        1998          12
```

2-Dimensional Array Operations (cont.)

- Add one more column in an array

```
>> c=[c(1,:), 10; c(2,:), 20; c(3,:), 30]
c =         1           3          10
          999           6          20
         1998          12          30
```

- Remove two columns

```
>> c(:,[1, 3])=[]
c =
      3
      6
     12
```

2-Dimensional Array Operations (cont.)

- Transpose a matrix

```
>> c‘
ans =
     3     6    12
```

- Exercise, explain what does the following command do ← help is your friend…

```
>> a=magic(12); b=a([2 5 3], [1 4])
b = 13    16
    96    93
    25    28
```

Popular Functions

- Figure out what do the functions do
  - abs(x)
  - sin(x)
  - exp(x)
  - log(x)
  - min(x)
  - max(x)
  - sort(x)
  - sum(x)
  - mean(x)
- Pass a matrix, say magic(5) into each of the function and figure out what happens

For Loops

```
for i = [vector]
    commands
end
```

- Each iteration, i is assigned with a new value, and commands are executed

```
>> for i = [100, 150, 200]
disp(i)
end
   100
   150
   200
```

While Loops

```
While expression
    commands
end
```

```
>> i=0; while i < 3; disp(i); i=i+1;end
     0
     1
     2
```

Conditional Executions

```
If expression
    commands
else
    commands
end
```

```
>> if 100 > 2; disp('true'); else; disp('false'); end
true
>> if 100 < 2; disp('true'); else; disp('false'); end
false
```

M Files

- M files are for Matlab
- There are two kinds of M files: scripts and functions
- Scripts: all variables are stored in workspace
- Functions: only input and output variables are connected to the workspace; other variables are thrown away after executions

Script File Example

```
% segment a bookshelf picture into multiple racks.....
% note that we didn't implement the landscape/portrait modes..
% We save a region for the second phase: book segmentation

url = 'file:///Users/cheng-hsinhsu/work/dt/asset/src/image/30724732f03_o.jpg';
pic = imread(url);
picg = rgb2gray(pic);

d_theta = 10; % degree deviation threshold is acceptable..
d_xy = 50;    % filter out closeby lines

picedge = edge(picg,'canny');
[pichough, theta, rho] = hough(picedge);
peaks = houghpeaks(pichough, 100, 'Threshold', 0.5 * max(pichough(:)));
lines = houghlines(picg, theta, rho, peaks, 'FillGap', 20, 'MinLength', 100);
......
```

Function File Example

```
% LOWPASSFILTER - Constructs a low-pass butterworth filter.
%
% usage: f = lowpassfilter(sze, cutoff, n)
%
% The frequency origin of the returned filter is at the corners.
%
% See also: HIGHPASSFILTER, HIGHBOOSTFILTER, BANDPASSFILTER
%

function f = lowpassfilter(sze, cutoff, n)

   if cutoff < 0 | cutoff > 0.5
   error('cutoff frequency must be between 0 and 0.5');
   end
......
```

Scripts versus Functions

- Scripts store all the variables in workspace → easier to check and manipulate their values
- Functions offer better encapsulation → don’t need to worry about overwriting variables in workspace
- Recursive function:

```
function out=fact02(n)

if n==1
     out=1;
     return
end
out=n*fact02(n-1);
```

Search Path

- path: display the current path setting
- which: figure out where is a specific function
- addpath: add a new path into the search paths
- rmpath: remove a path from the search paths

Variables in Workspace

- who: list all the variables in workspace
- whos: list details about the variable in workspace
- clear: clean up the workspace variables
  - Default is clear all variables, or you may specify a specific variable
- save: save variables into a file
  - save ← save all variables to matlab.mat binary file
  - save filename x, y, z ← save variables x, y, z to filename.mat

Quit Matlab

- exit
- quit
- or just close the window

Matlab #1 Homework (M1)

1. (2%) Write a one-line MATLAB statement for the following short questions:
   - Change element 3 of vector x by multiplying it by 5
   - Delete columns 2 and 4 from matrix A
   - Swap rows 1 and 3 of matrix A
   - Extract columns 4, 2, and 5 of matrix A and assign them to matrix B

Matlab #1 Homework (M1) (cont.)

2. (1%) Fibonacci numbers are defined recursively as follows. $F_1=F_2=1$, and $F_n=F_{n-1}+F_{n-2}$ for all integers $n>=3$. Write a Matlab recursive function to calculate x-th Fibonacci number, where x is an input argument. Note that you get zero point if you don’t use recursion in your code.