PENGOLAHAN SINYAL DIGITAL

Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

MATLAB

Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

WHY MATLAB?

- Matrix Labratory
- Created in late 1970’s
- Intended for used in courses in matrix theory, linear algebra and numerical analysis
- Currently has grown into an interactive system and high level programming language for general scientific and technical computation

WHY MATLAB? COMMON USES FOR MATLAB IN RESEARCH

- Data Acquisition
- Multi-platform, Multi Format data importing
- Analysis Tools (Existing, Custom)
- Statistics
- Graphing
- Modeling

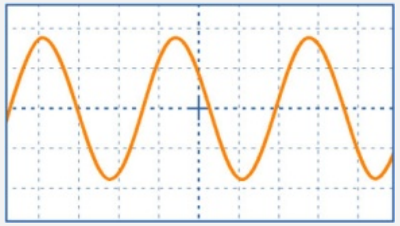
WHY MATLAB? DATA ACQUISITION

- A framework for bringing live, measured data into MATLAB using PC-compatible, plug-in data acquisition hardware

WHY MATLAB? MULTI-PLATFORM, MULTI FORMAT DATA IMPORTING

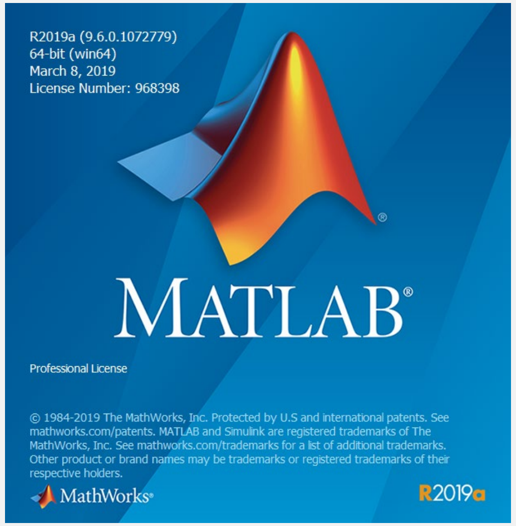
- Data can be loaded into Matlab from almost any format and platform
- Binary data files (eg. REX, PLEXON etc.)
- Ascii Text (eg. Eyelink I, II)
- Analog/Digital Data files

PC
UNIX



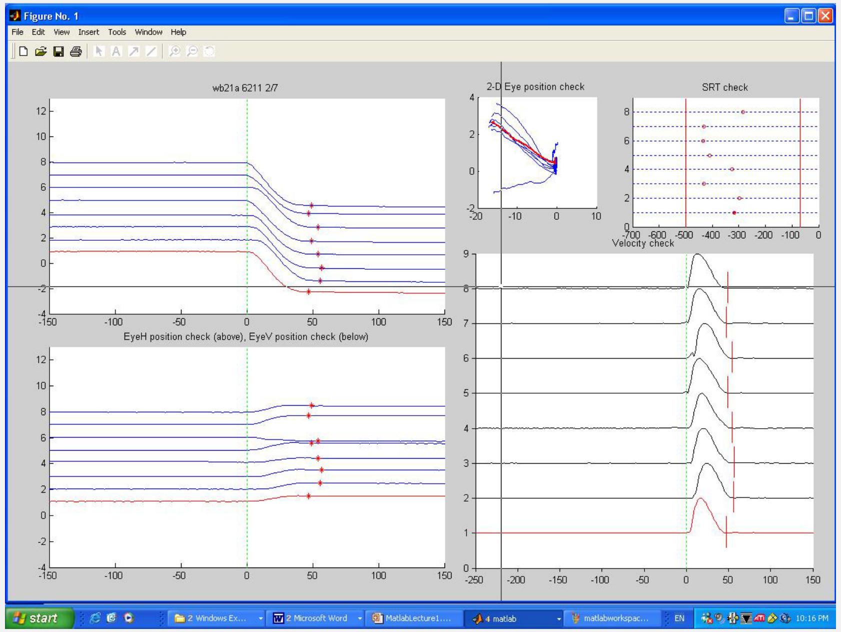
100101010

Subject 1 143
Subject 2 982
Subject 3 87 …



WHY MATLAB? ANALYSIS TOOLS

- A Considerable library of analysis tools exist for data analysis
- Provides a framework for the design, creation, and implementation of any custom analysis tool imaginable

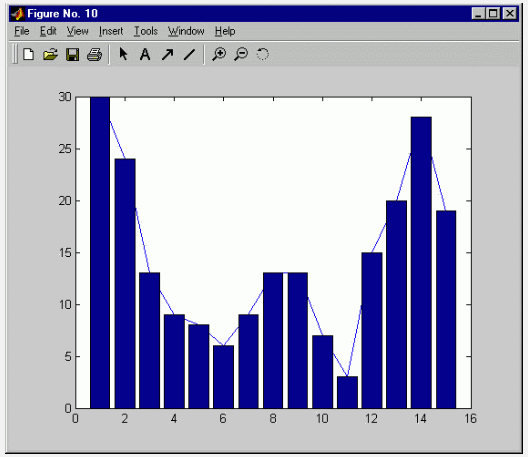


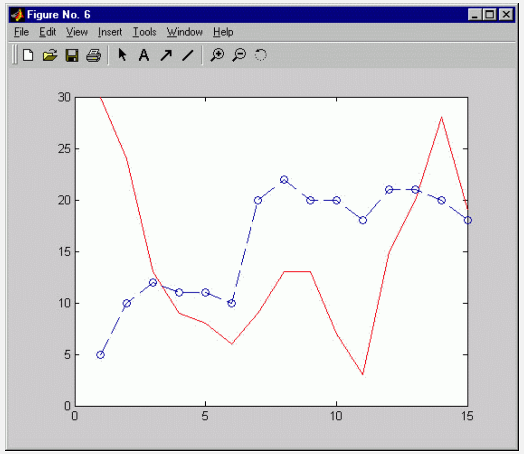
WHY MATLAB? STATISTICAL ANALYSIS

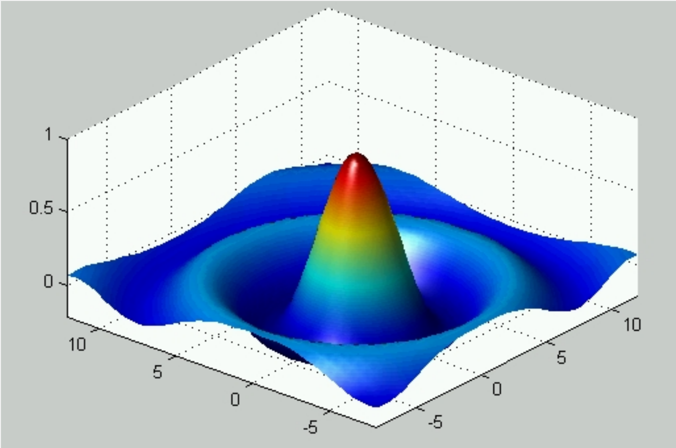
- A considerable variety of statistical tests available including:
  - T-TEST
  - Mann-Whitney Test
  - Rank Sum Test
  - ANOVAs
  - Linear Regressions
  - Curve Fitting

WHY MATLAB? GRAPHING

- A Comprehensive array of plotting options available from 2 to 4 dimensions
- Full control of formatting, axes, and other visual representational elements

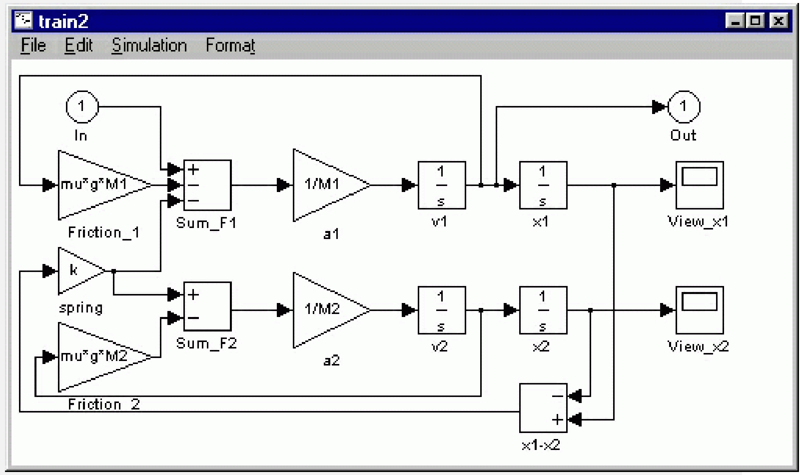


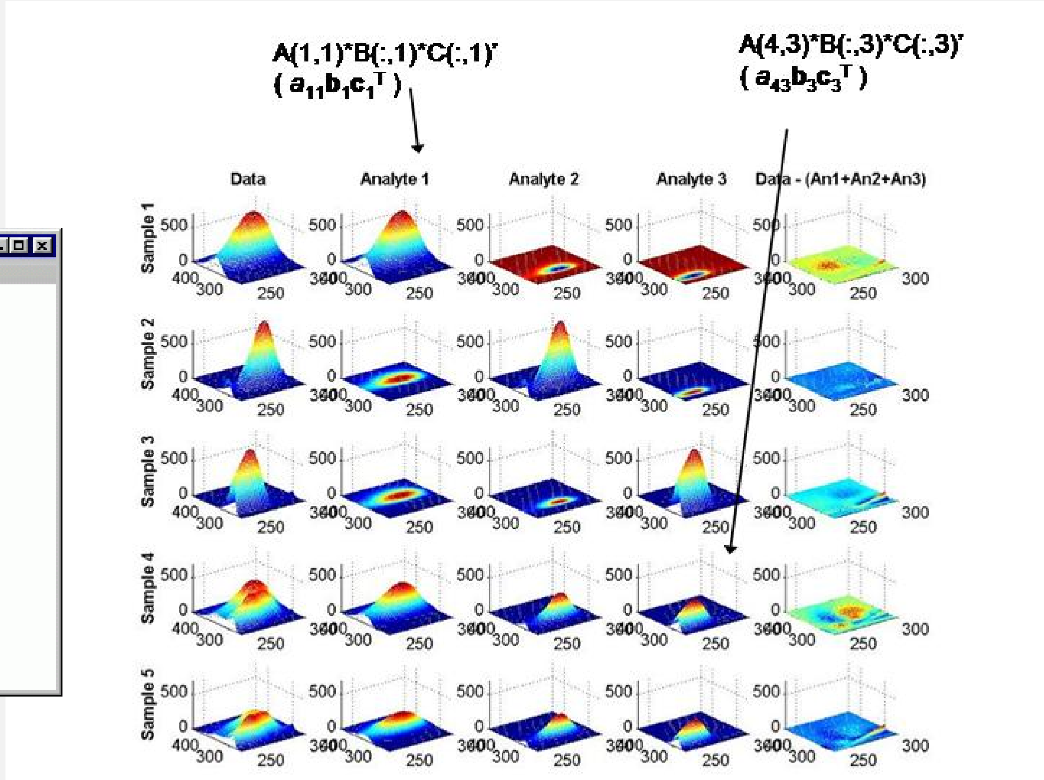




WHY MATLAB? MODELING

- Models of complex dynamic system interactions can be designed to test experimental data



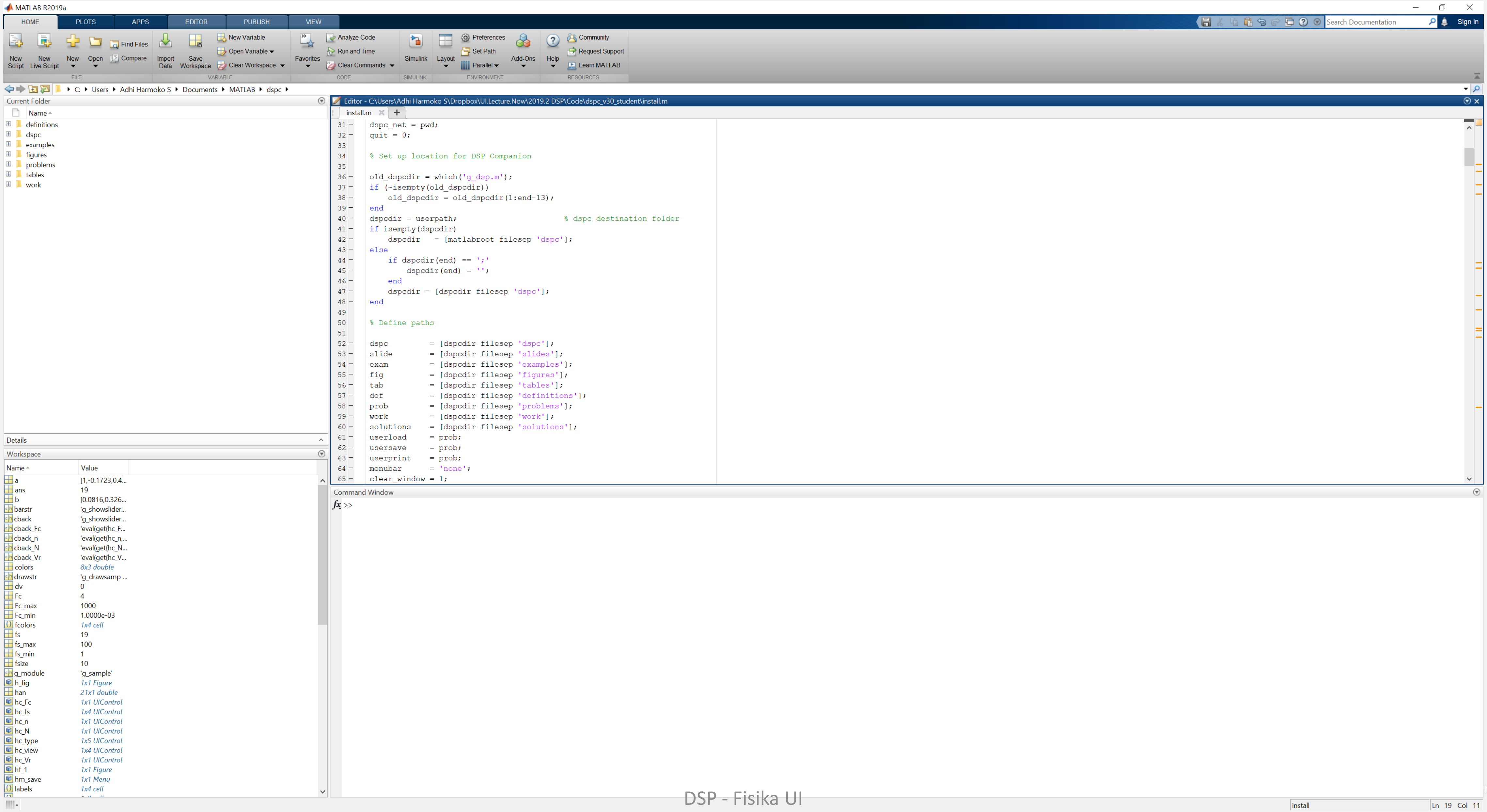


MATLAB INTERFACE

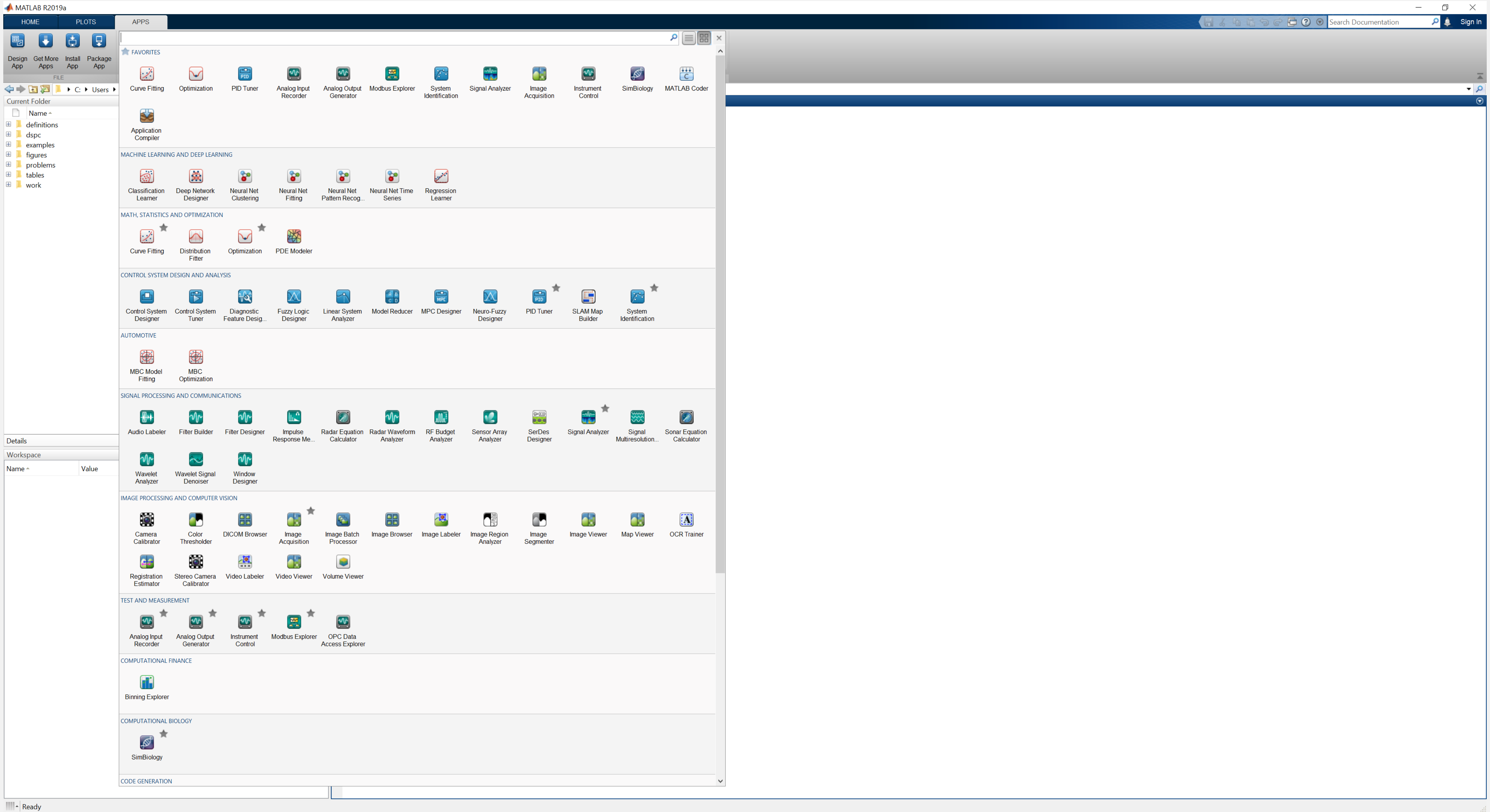
Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

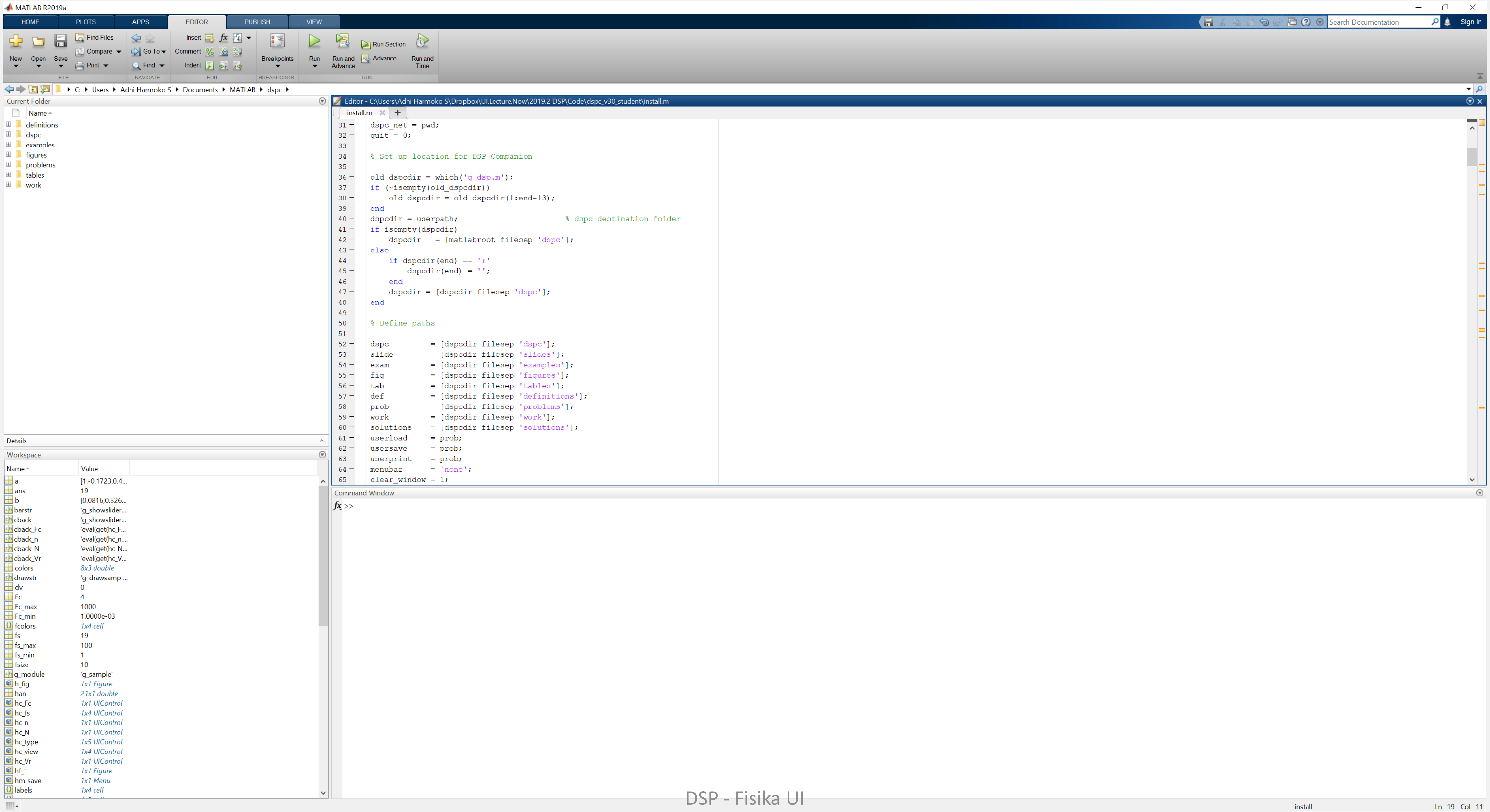
NAVIGATING THE MATLAB DESKTOP



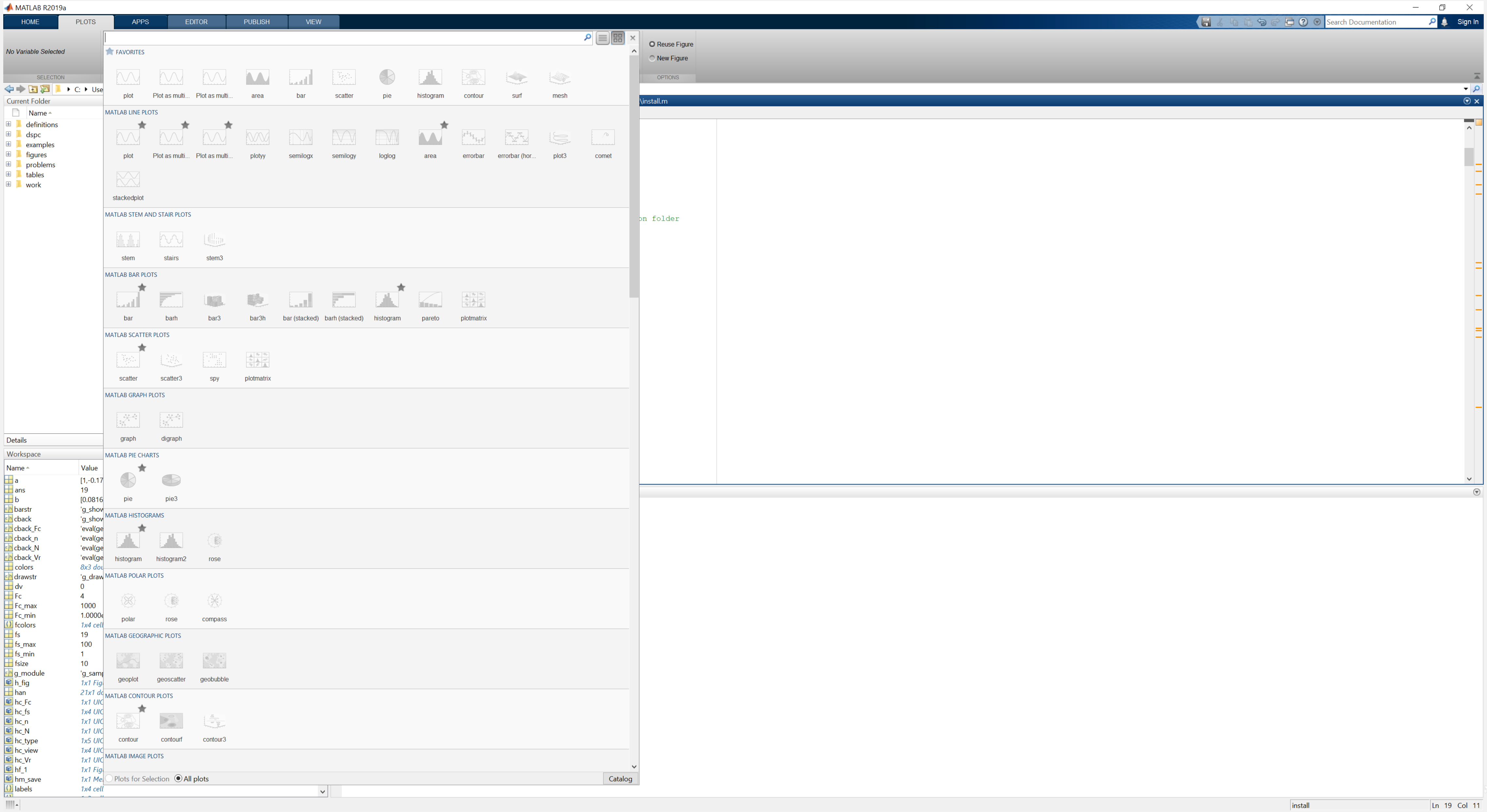
COMMONLY USED TOOLBOXES



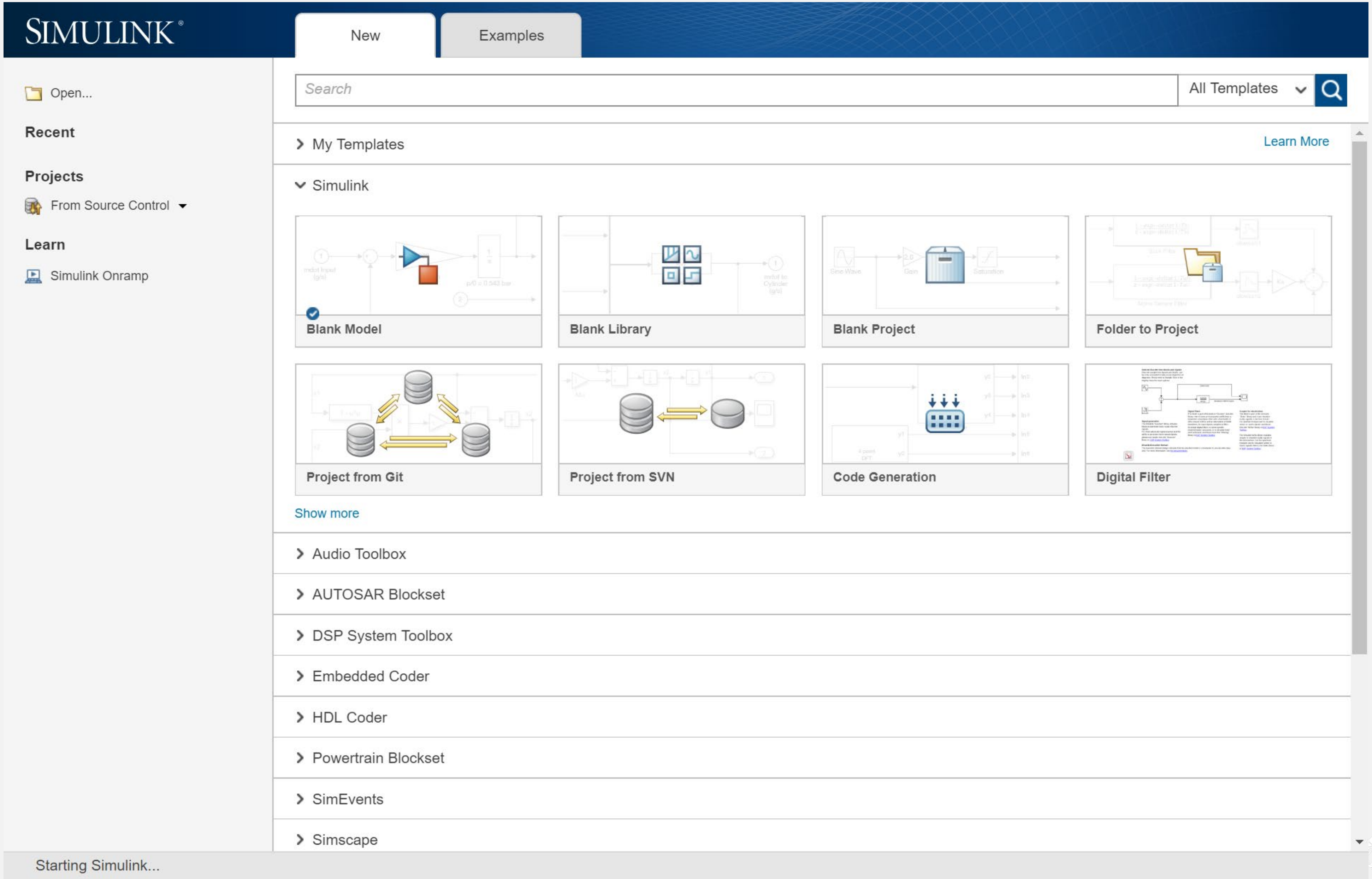
EDITOR NAVIGATION



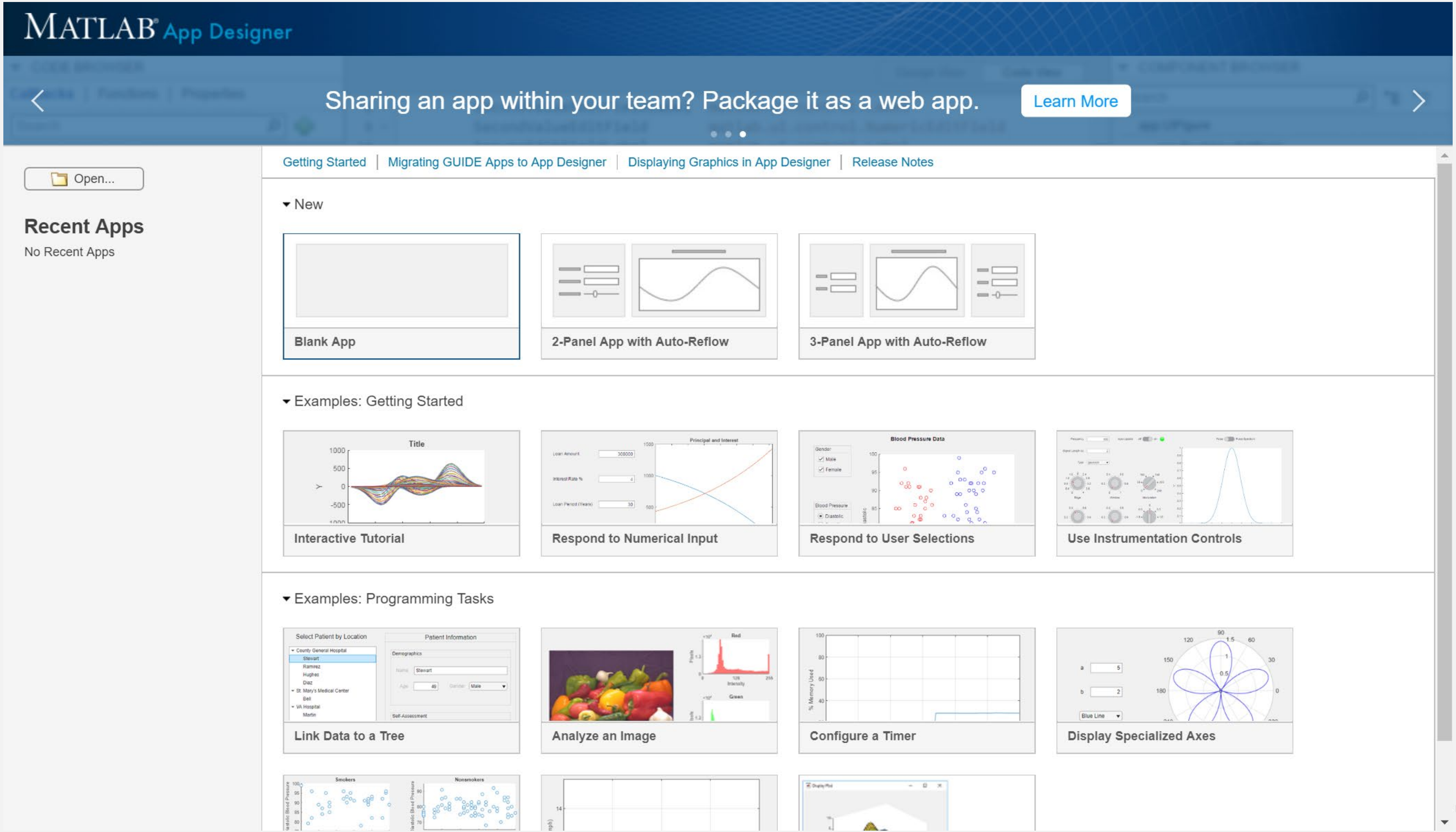
PLOTTING NAVIGATION



SIMULINK NAVIGATION



APPLICATION DESIGNER



MATLAB OPERATION

Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

NUMBERS

- MATLAB is a high-precision numerical engine and can handle all types of numbers, that is, integers, real numbers, complex numbers, among others, with relative ease.
- For example, the real number 1.23 is represented as simply 1.23 while the real number 4.56×10^7 can be written as 4.56e7.
- The imaginary number $\sqrt{-1}$ is denoted either by 1i or 1j, although in this book we will use the symbol 1j.
- Hence the complex number whose real part is 5 and whose imaginary part is 3 will be written as 5+1j*3.
- Other constants preassigned by MATLAB are pi for π, inf for ∞, and NaN for not a number (for example, 0/0).
- These preassigned constants are very important and, to avoid confusion, should not be redefined by users.

VARIABLES

- The basic variable is a matrix, or an array.
- MATLAB now supports multidimensional arrays
  - **Matrix:** A matrix is a two-dimensional set of numbers arranged in rows and columns. Numbers can be real- or complex-valued.
  - **Array:** This is another name for matrix. However, operations on arrays are treated differently from those on matrices. This difference is very important in implementation.

WORKING WITH MATRICES

- Matlab works with essentially only one kind of object, a rectangular numerical matrix
- A matrix is a collection of numerical values that are organized into a specific configuration of rows and columns.
- The number of rows and columns can be any number

Example

- 3 rows and 4 columns define a 3 x 4 matrix having 12 elements

WORKING WITH MATRICES

- **Scalar:** This is a 1×1 matrix or a single number that is denoted by the *variable* symbol, that is, lowercase italic typeface like

$$a = a_{11}$$

WORKING WITH MATRICES

- **Column vector:** This is an (N × 1) matrix or a vertical arrangement of numbers.
- It is denoted by the *vector* symbol, that is, lowercase bold typeface like

$$\mathbf{x} = \left[x_{i1} \right]_{i:1,\ldots,\mathrm{N}} = \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{N1} \end{bmatrix}$$

- A typical vector in linear algebra is denoted by the column vector

WORKING WITH MATRICES

- **Row vector:** This is a (1×*M*) matrix or a horizontal arrangement of numbers.
- It is also denoted by the vector symbol, that is,

$$\mathbf{y} = \left[y_{1j} \right]_{\mathrm{j}=1,\ldots,M} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1M} \end{bmatrix}$$

- A one-dimensional discrete-time signal is typically represented by an array as a row vector.

WORKING WITH MATRICES

- **General matrix:** This is the most general case of an ($N \times M$) matrix and is denoted by the matrix symbol, that is, uppercase bold typeface like

$$\mathbf{A} = \left[a_{ij} \right]_{i=1,\ldots,\mathrm{N};\, \mathrm{j}=1,\ldots,\mathrm{M}} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix}$$

- This arrangement is typically used for two-dimensional discrete-time signals or images

WORKING WITH MATRICES - EXAMPLE

```
c = 5.66                       %c is a scalar or a 1 x 1 matrix
x = [ 3.5, 33.22, 24.5 ]       %x is a row vector or a 1 x 3 matrix
x1 = [2
       5
      3
      -1]                      %x1 is column vector or a 4 x 1 matrix
A = [ 1  2  4
      2 -2  2
      0  3  5
      5  4  9 ]                %A is a 4 x 3 matrix
```

WORKING WITH MATRICES - EXAMPLE

Command Window

```
>> c = 5.66                    %c is a scalar or a 1 x 1 matrix
x = [ 3.5, 33.22, 24.5 ]       %x is a row vector or a 1 x 3 matrix
x1 = [2
     5
      3
      -1]                  %x1 is column vector or a 4 x 1 matrix
A = [ 1  2  4
      2 -2  2
      0  3  5
      5  4  9 ]                %A is a 4 x 3 matrix

c =
    5.6600
x =
    3.5000   33.2200   24.5000
x1 =
     2
     5
     3
    -1
A =
     1     2     4
     2    -2     2
     0     3     5
     5     4     9
fx >>
```

Workspace

Name	Value
A	*4x3 double*
c	5.6600
x	[3.5000,33.22...
x1	[2;5;3;-1]

A

PLOTS | VARIABLE | VIEW

New from Selection | Open | Print | Rows 1 | Columns 1 | Insert | Delete | Transpose | Sort

VARIABLE | SELECTION | EDIT

4x3 double

	1	2	3	4	5	6
1	1	2	4			
2	2	-2	2			
3	0	3	5			
4	5	4	9			
5						
6						
7						

WORKING WITH MATRICES

- Spaces, commas, and semicolons are used to separate elements of a matrix

- Spaces or commas separate elements of a row
  [1 2 3 4] or [1,2,3,4]

- Semicolons separate columns
  [1,2,3,4;5,6,7,8;9,8,7,6] = [1 2 3 4
  5 6 7 8
  9 8 7 6]

OPERATORS

- MATLAB provides several arithmetic and logical operators, some of which follow.

= assignment	== equality
+ addition	- subtraction or minus
* multiplication	.* array multiplication
^ power	.^ array power
/ division	./ array division
<> relational operators	& logical AND
| logical OR	~ logical NOT
’ transpose	.’ array transpose

IMPORTANT OPERATIONS ON MATRICES

- **Matrix addition and subtraction:** These are straightforward operations that are also used for array addition and subtraction. Care must be taken that the two matrix operands be *exactly* the same size.

- **Matrix conjugation:** This operation is meaningful only for complex valued matrices. It produces a matrix in which all imaginary parts are negated. It is denoted by $\mathbf{A}*$ in analysis and by conj(A) in MATLAB.

IMPORTANT OPERATIONS ON MATRICES

- **Matrix transposition:** This is an operation in which every row (column) is turned into column (row). Let **X** be an (*N* × *M*) matrix. Then

$$\mathbf{X}' = \left[x_{ij} \right]; \quad j = 1, \ldots, M, \quad i = 1, \ldots, N$$

- is an (*M* × *N*) matrix

IMPORTANT OPERATIONS ON MATRICES

- **Multiplication by a scalar:** This is a simple straightforward operation in which each element of a matrix is scaled by a constant, that is

$$ab \Rightarrow a * b \quad (\text{scalar})$$

$$a\mathrm{x} \Rightarrow a * \mathrm{x} \quad (\text{vector or array})$$

$$a\mathbf{X} \Rightarrow a * \mathrm{X} \quad (\text{matrix})$$

- This operation is also valid for an array scaling by a constant

IMPORTANT OPERATIONS ON MATRICES

- **Vector-vector multiplication:** In this operation, one has to be careful about matrix dimensions to avoid invalid results.
- The operation produces either a scalar or a matrix. Let **x** be an (N × 1) and **y** be a (1 × M) vectors.
- Then

$$x * y = xy \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \begin{bmatrix} y_1 & \cdots & y_M \end{bmatrix} = \begin{bmatrix} x_1 y_1 & \cdots & x_1 y_M \\ \vdots & \ddots & \vdots \\ x_N y_1 & \cdots & x_N y_M \end{bmatrix}$$

- produces a matrix.

IMPORTANT OPERATIONS ON MATRICES

- **Matrix-vector multiplication:** If the matrix and the vector are compatible (i.e., the number of matrix-columns is equal to the vector-rows), then this operation produces a column vector:

$$y = A * x \Rightarrow \mathbf{y} = \mathbf{A}\mathbf{x} = \begin{bmatrix} a_{11} & \cdots & a_{1M} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NM} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_M \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

IMPORTANT OPERATIONS ON MATRICES

- **Matrix-matrix multiplication:** Finally, if two matrices are compatible, then their product is well-defined.
- The result is also a matrix with the number of rows equal to that of the first matrix and the number of columns equal to that of the second matrix.
- Note that the order in matrix multiplication is very important.

ARRAY OPERATIONS

- These operations treat matrices as arrays.
- They are also known as *dot operations* because the arithmetic operators are prefixed by a dot (.), that is, .*, ./, or .^.

ARRAY OPERATIONS

- **Array multiplication:** This is an element by element multiplication operation.
- For it to be a valid operation, both arrays must be the same size. Thus we have

$$x.*\,y \rightarrow \text{1D array}$$
$$\text{X.*Y} \rightarrow \text{2D array}$$

ARRAY OPERATIONS

- **Array exponentiation:** In this operation, a scalar (real- or complexvalued) is raised to the power equal to every element in an array, that is,

$$a.\hat{}\, x \equiv \begin{bmatrix} a^{x_1} \\ a^{x_2} \\ \vdots \\ a^{x_N} \end{bmatrix}$$

ARRAY OPERATIONS

- **Array transposition:** As explained, the operation A. produces transposition of real- or complex-valued array A.

INDEXING MATRICES

- A $m \times n$ matrix is defined by the number of m rows and number of n columns
- An individual element of a matrix can be specified with the notation A(i,j) or Ai,j for the generalized element, or by A(4,1)=5 for a specific element.

```
>> A = [1 2 4 5;6 3 8 2]
```

A is a 2 x 4 matrix

```
>> A(1,2)
Ans =  2
```

- The colon operator can be used to index a range of elements

```
>> A(2,1:3)
Ans = 6 3 8
```

INDEXING MATRICES

- Specific elements of any matrix can be overwritten using the matrix index

Example:

```
A = [1 2 4 5
     6 3 8 2]

A(2,1) = 9
Ans
 A = [1 2 4 5
      9 3 8 2]
```

MATRIX SHORTCUTS

- The ones and zeros functions can be used to create any m x n matrices composed entirely of ones or zeros

Example

```
a = ones(3,2)
a = [1 1
     1 1
     1 1]
b = zeros(1,5)
b = [0 0 0 0 0]
```

CONTROL-FLOW

if-elseif-else structure

```
if condition1
      command1
elseif condition2
      command2
else
      command3
end
```

CONTROL-FLOW

for..end loop

```
for index = values
     program statements
     :
end
```

EXAMPLE

- Consider the following sum of sinusoidal functions

$$x(t) = \sin(2\pi t) + \frac{1}{3}\sin(6\pi t) + \frac{1}{5}\sin(10\pi t) = \sum_{k=1}^{3} \frac{1}{k}\sin(2\pi k t), \quad 0 \le t \le 1$$

- Using MATLAB, we want to generate samples of $x(t)$ at time instances 0:0.01:1.

EXAMPLE

- **Approach 1** Here we will consider a typical C or Fortran approach, that is, we will use two for..end loops, one each on t and k.
- This is the most inefficient approach in MATLAB, but possible.

```
t = 0:0.01:1; N = length(t); xt = zeros(1,N);
for n = 1:N
        temp = 0;
        for k = 1:3
                temp = temp + (1/k)*sin(2*pi*k*t(n));
        end
        xt(n) = temp;
end
```

EXAMPLE

- **Approach 2** In this approach, we will compute each sinusoidal component in one step as a vector, using the time vector t = 0:0.01:1 and then add all components using one for..end loop.

```
t = 0:0.01:1; xt = zeros(1,length(t));
for k = 1:3
    xt = xt + (1/k)*sin(2*pi*k*t);
end
```

SCRIPTS AND FUNCTIONS

- ***Scripts***
  - implemented using a *script* file called an m-file (with an extension .m), which is only a text file that contains each line of the file as though you typed them at the command prompt.
  - built-in editor, which also provides for context-sensitive colors and indents for making fewer mistakes and for easy reading.
  - executed by typing the name of the script at the command prompt.
  - script file must be in the current directory on in the directory of the path environment.

SCRIPTS AND FUNCTIONS

- Example:
- General form of sinusoidal function is

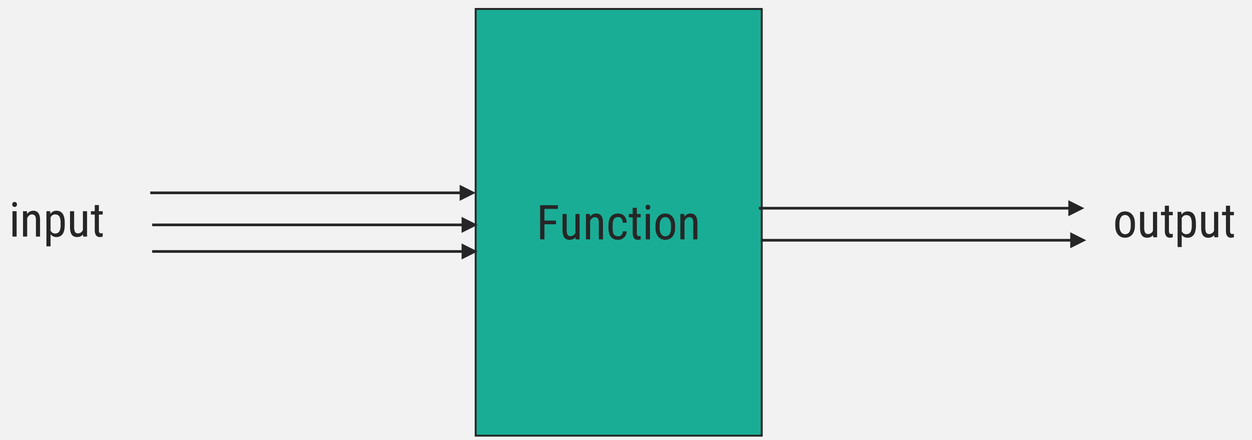
$$x(t) = \sum_{k=1}^{K} c_k \sin(2\pi kt)$$

- create a script file !

```
% Script file to implement
t = 0:0.01:1; k = 1:2:5; ck = 1./k;
xt = ck * sin(2*pi*k’*t);
```

SCRIPTS AND FUNCTIONS

- ***Functions***
  - The second construct of creating a block of code is through subroutines.
  - A major difference between script and function files is that the first executable line in a function file begins with the keyword function followed by an output-input variable declaration

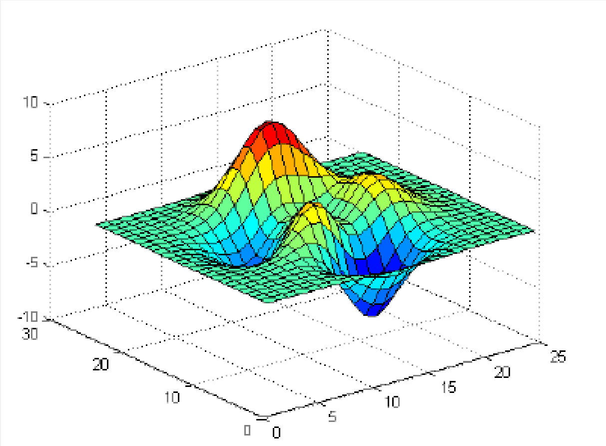


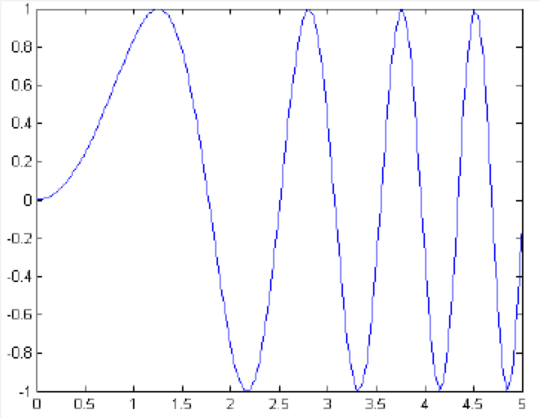
SCRIPTS AND FUNCTIONS - EXAMPLE

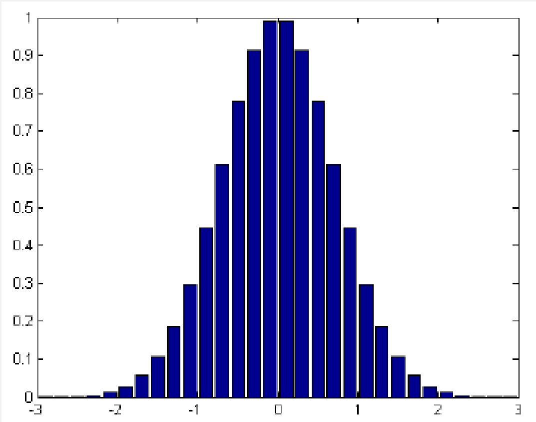
```
function xt = sinsum(t,ck)
% Computes sum of sinusoidal terms of the form in (1.1)
% x = sinsum(t,ck)
%
K = length(ck); k = 1:K;
ck = ck(:)’; t = t(:)’;
xt = ck * sin(2*pi*k’*t);
```

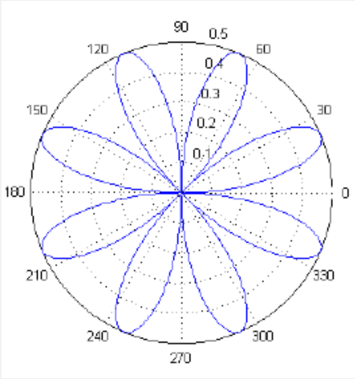
PLOTTING

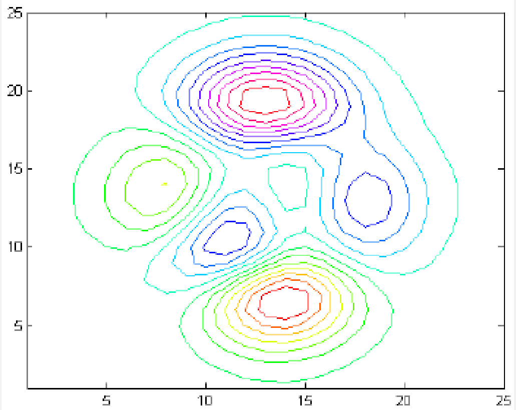
- Matlab has a powerful plotting engine that can generate a wide variety of plots.

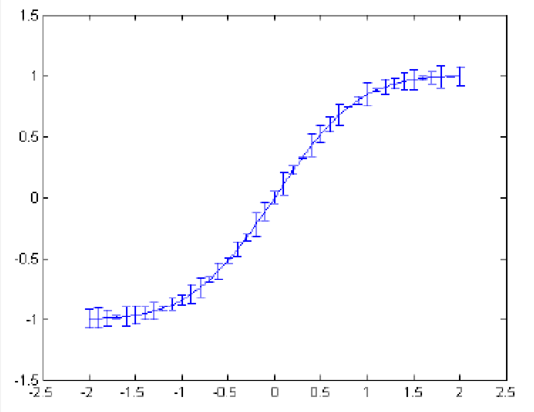












PLOTTING

- The basic plotting command is the plot(t,x) command, which generates a plot of x values versus t values in a separate figure window.
- The arrays t and x should be the same length and orientation.
- Optionally, some additional formatting keywords can also be provided in the plot function.
- The commands xlabel and ylabel are used to add text to the axis, and the command title is used to provide a title on the top of the graph.
- All aspects of a plot (style, size, color, etc.) can be changed by appropriate commands embedded in the program or directly through the GU

PLOTTING EXAMPLE

```
% Plot of a simple sinusoidal wave
% putting axis labels and title on the plot

t = 0:0.01:2; % sample points from 0 to 2 in steps of 0.01
x = sin(2*pi*t); % Evaluate sin(2 pi t)
plot(t,x,’b’); % Create plot with blue line
xlabel(’t in sec’); ylabel(‘x(t)’); % Label axis
title(’Plot of sin(2\pi t)’); % Title plot
```

PLOTTING EXAMPLE

- MATLAB provides an ability to display more than one graph in the same figure window.
- By means of the hold on command, several graphs can be plotted on the same set of axes.
- The hold off command stops the simultaneous plotting

```
plot(t,xt,’b’); hold on; % Create plot with blue line
Hs = stem(n*0.05,xn,’b’,’filled’); % Stem-plot with handle Hs
set(Hs,’markersize’,4); hold off; % Change circle size
```

PLOTTING EXAMPLE

- The subplot command, which displays several graphs in each individual set of axes arranged in a grid, using the parameters in the subplot command.

```
...
subplot(2,1,1); % Two rows, one column, first plot
plot(t,x,'b'); % Create plot with blue line
...
subplot(2,1,2); % Two rows, one column, second plot
Hs = stem(n,x,'b','filled'); % Stem-plot with handle Hs
...
```

TERIMA KASIH

Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia