Lecture 02: Matlab Basics: Data Types and Arrays, Input / Output, Built-in Functions, M-Files, and Plotting

Learning Objectives:
- Use variables, operators, and control structures to implement simple sequential algorithms
- Use Matlab m-files to create user-defined programs and functions

Topics/Outline:
1. Variables and the assignment operator
2. Arrays and matrices
3. Element-wise operation
4. Built-in functions
5. Plotting
6. Save/load commands
7. M-Files
Example: Working with Matlab

Plot \( f(t) = k_0 \exp(-\beta t^2) \)

\( f(t) \) function value.

\( k_0 \) independent variable.

\( \beta \) constants

\( t \) independent variable.

How do you get Matlab to do this?

Variables & Assignment Operator:

Matlab stores information in variables

\( k_0 = 4.3 \)

Variable name.

floating point number (data type)

assignment: Store RHS result in LHS variable

Matlab: "Matrix Laboratory"

Default variable type is an array.

\( k_0(1) \) Array index inside ()

counter starts at 1 indices are integers.

\( \text{ans} = 4.3 \)
Initializing Arrays: Many ways.

$$ t = [0, 1, 2, 3] $$

$$ t = [0, 0.5, 1, 1.5, 3, 7] $$

$$ t = 0 : 0.01 : 10 \quad \text{colon operator:} \begin{array}{c}
\uparrow & \uparrow & \uparrow \\
\text{start} & \text{step} & \text{stop}
\end{array} \quad \text{performs iteration.}

$$ t = \text{linspace}(0, 10, 100) \quad \begin{array}{c}
\uparrow & \uparrow & \Rightarrow \\
\text{start} & \text{stop} & \text{# of values}
\end{array}

$$ t = \text{zeros}(20, 1) \quad \begin{array}{c}
\Rightarrow & \Rightarrow \\
\# of & \# of
\end{array} \begin{array}{c}
\text{rows} & \text{columns}
\end{array}

Matlab Arrays are Matrices:

$$ A = \begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix} $$

\uparrow \text{new line}

\text{inv}(A) \text{ or } A^\text{-1} : \text{matrix inverse}

A' : \text{matrix transpose}

A \times A : \text{matrix multiplication.}

$$ b = \begin{bmatrix} 7 \\ 10 \end{bmatrix} : \text{column vector (2 \times 1)} $$

$$ c = \begin{bmatrix} 7 & 10 \end{bmatrix} : \text{row vector (1 \times 2)} $$

Legal matrix operations:

$$ c \times A ; \quad A \times b ; \quad A' \times b ; \quad A \times c' ; \quad A^\text{-1} \times b $$
Element-wise Operation:

Sometimes we don't want the array to be a matrix, but want to do something to each element of the array.

\[
\gg t = 0: 0.01: 10
\]
\[
\gg t.^2 \quad \text{matrix multiplication (1x1) (1x1)}
\]

Use the operator

\[
\gg t.^2 \quad \text{square each value in } t.
\]

Built-in Functions: There are MANY.

For searching for unknown functions, use online documentation.

In command window:

\[
\gg \text{help exp}
\]

you have to know the right name.

\[
\gg \text{lookfor exponential}
\]

can only be one word.

\[
\gg \text{help elfun}
\]
gives a nice list of basic calculator functions.
Plot Example:

```matlab
>> k0 = 4.3;  % Stop Matlab from echoing the result to the screen.

>> beta = 0.07;
>> t = 0:0.01:10;
    % Starts a comment
>> f = k0 * exp (- beta * t.^2);

% Create plot
>> plot (t, f)  % read help for:
    % plot, xlabel, ylabel, axis, grid
    % legend
```

Save: save matrix to an ASCII file

```matlab
>> data = [t, f']  % forms 
    % transpose to (n x 1)

>> save my_data.dat -ascii data
    % filename  data
    % type      variable
```
Load: Read data from disk.

```matlab
data = load( 'my_data.dat' );
```

↑ must contain only numbers in full rows and columns.
Can also have % comments.

Can access data with "::" slice operator:

```matlab
t = data(:, 1)
```

↑ & column 1.

means
"all rows"

Try:

```matlab
data (5:20, 1)
data (50: end, 2)
data (1: 50)
data (1: 50, :)
data (1: 50, 1:2)
```

M-Files: see my-basic-plot.m

Matlab executes everything in a text .m-file as if the commands were entered at the command prompt.

We will call these files programs.