MAT 275 Laboratory 1
Introduction to MATLAB

MATLAB is a computer software commonly used in both education and industry to solve a wide range of problems.

This Laboratory provides a brief introduction to MATLAB, and the tools and functions that help you to work with MATLAB variables and files.

The MATLAB Environment

★ To start MATLAB double-click on the MATLAB shortcut icon. The MATLAB desktop will open.

On the left side you will generally find the Current Folder window and on the right the Workspace and Command History windows. The Command Window is where the MATLAB commands are entered and executed. Note that windows within the MATLAB desktop can be resized by dragging the separator bar(s).

If you have never used MATLAB before, we suggest you type `demo` at the MATLAB prompt. Click on *Getting Started with MATLAB* and run the file.

Basics And Help

Commands are entered in the Command Window.
★ Basic operations are `+`, `-`, `*`, and `/`. The sequence

```
>> a=2; b=3; a+b, a*b
ans =
     5
ans =
     6
```

defines variables a and b and assigns values 2 and 3, respectively, then computes the sum $a+b$ and product ab. Each command ends with `,` (output is visible) or `;` (output is suppressed). The last command on a line does not require a `,`.
★ Standard functions can be invoked using their usual mathematical notations. For example

```
>> theta=pi/5;
>> cos(theta)^2+sin(theta)^2
ans =
     1
```

verifies the trigonometric identity $\sin^2\theta + \cos^2\theta = 1$ for $\theta = \frac{\pi}{5}$. A list of elementary math functions can be obtained by typing

```
>> help elfun
```

★ To obtain a description of the use of a particular function type help followed by the name of the function. For example

```
>> help cosh
```

gives help on the hyperbolic cosine function.
★ To get a list of other groups of MATLAB programs already available enter `help`:

```
>> help
```

★ Another way to obtain help is through the desktop Help menu, **Help** > **Product Help**.

★ MATLAB is case-sensitive. For example

```
>> theta=1e-3, Theta=2e-5, ratio=theta/Theta
theta =
  1.0000e-003
Theta =
  2.0000e-005
ratio =
    50
```

★ The quantities `Inf` (∞) and `NaN` (Not a Number) also appear frequently. Compare

```
>> c=1/0
c =
   Inf
```

with

```
>> d=0/0
d =
   NaN
```

Plotting with MATLAB

★ To plot a function you have to create two arrays (vectors): one containing the abscissae, the other the corresponding function values. Both arrays should have the same length. For example, consider plotting the function

$$y = f(x) = \frac{x^2 - \sin(\pi x) + e^x}{x-1}$$

for $0 \le x \le 2$. First choose a sample of x values in this interval:

```
    >> x=[0,.1,.2,.3,.4,.5,.6,.7,.8,.9,1, ...
    1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9,2]
 x =
  Columns 1 through 7
         0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000
  Columns 8 through 14
    0.7000    0.8000    0.9000    1.0000    1.1000    1.2000    1.3000
  Columns 15 through 21
    1.4000    1.5000    1.6000    1.7000    1.8000    1.9000    2.0000
```

Note that an ellipsis `...` was used to continue a command too long to fit in a single line.
Rather than manually entering each entry of the vector `x` we can simply use

```
>> x=0:.1:2
```

or

```
>> x=linspace(0,2,21)
```

Both commands above generate the same output vector `x`.

★ The output for x can be suppressed (by adding `;` at the end of the command) or condensed by entering

```
>> format compact
```

(This format was used for all previous outputs).

★ To evaluate the function f simultaneously at all the values contained in x, type

```
    >> y=(x.^2-sin(pi.*x)+exp(x))./(x-1)
y =
  Columns 1 through 6
   -1.0000   -0.8957   -0.8420   -0.9012   -1.1679   -1.7974
  Columns 7 through 12
   -3.0777   -5.6491  -11.3888  -29.6059       Inf   45.2318
  Columns 13 through 18
   26.7395   20.5610   17.4156   15.4634   14.1068   13.1042
  Columns 19 through 21
   12.3468   11.7832   11.3891
```

Note that the function becomes infinite at $x = 1$ (vertical asymptote). The array y inherits the dimension of x, namely 1 (row) by 21 (columns). Note also the use of parentheses.

IMPORTANT REMARK
In the above example `*`, `/` and `^` are preceded by a dot `.` in order for the expression to be evaluated for each component (entry) of x. This is necessary to prevent MATLAB from interpreting these symbols as standard linear algebra symbols operating on arrays. Because the standard `+` and `-` operations on arrays already work componentwise, a dot is not necessary for `+` and `-`.

The command

```
    >> plot(x,y)
```

creates a Figure window and shows the function. The figure can be edited and manipulated using the Figure window menus and buttons. Alternately, properties of the figure can also be defined directly at the command line:

```
    >> x=0:.01:2;
    >> y=(x.^2-sin(pi.*x)+exp(x))./(x-1);
    >> plot(x,y,'r-','LineWidth',2);
    >> axis([0,2,-10,20]); grid on;
    >> title('f(x)=(x^2-sin(\pi x)+e^x)/(x-1)');
    >> xlabel('x'); ylabel('y');
```

Remarks:

- The number of x-values has been increased for a smoother curve (note that the stepsize is now .01 rather than .1).
- The option `'r-'` plots the curve in red.
- `'LineWidth',2` sets the width of the line to 2 points (the default is 0.5).
- The range of x and y values has been reset using `axis([0,2,-10,20])` (always a good idea in the presence of vertical asymptotes).
- The command `grid on` adds a grid to the plot.
- A title and labels have been added.

The resulting new plot is shown in Fig. L1a. For more options type `help plot` in the Command Window.

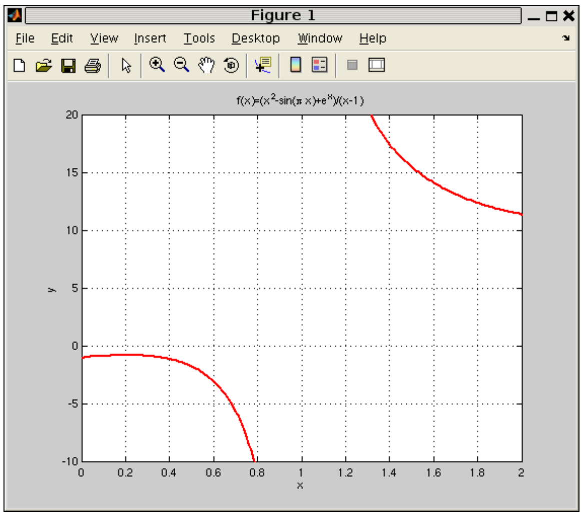
Figure 1
File Edit View Insert Tools Desktop Window Help
f(x)=(x^2-sin(π x)+e^x)/(x-1)
20
15
10
5
0
-5
-10
y
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
x

Figure L1a: A Figure window

Scripts and Functions

★ Files containing MATLAB commands are called `m`-files and have a `.m` extension. They are two types:

1. A *script* is simply a collection of MATLAB commands gathered in a single file. The value of the data created in a script is still available in the Command Window after execution. To create a new script select the MATLAB desktop File menu **File** > **New** > **Script**. In the MATLAB text editor window enter the commands as you would in the Command window. To save the file use the menu **File** > **Save** or **File** > **Save As...**, or the shortcut SAVE button .
   Variable defined in a *script* are accessible from the command window.

2. A *function* is similar to a script, but can accept and return arguments. Unless otherwise specified any variable inside a function is local to the function and not available in the command window. To create a new function select the MATLAB desktop File menu **File** > **New** > **Function**. A MATLAB text editor window will open with the following predefined commands

   ```
   function [ output_args ] = Untitled3( input_args )
   %UNTITLED3 Summary of this function goes here
   %   Detailed explanation goes here

   end
   ```

   The "output_args" are the output arguments, while the "input_args" are the input arguments. The lines beginning with % are to be replaced with comments describing what the functions does. The command(s) defining the function must be inserted after these comments and before `end`.

   To save the file proceed similarly to the Script M-file.

Use a function when a group of commands needs to be evaluated multiple times.

★ Examples of script/function:

1. **script**

   `myplot.m`

   ```
   x=0:.01:2;                                % x-values
   y=(x.^2-sin(pi.*x)+exp(x))./(x-1);        % y-values
   ```

```
plot(x,y,'r-','LineWidth',2);                 % plot in red with wider line
axis([0,2,-10,20]); grid on;                  % set range and add grid
title('f(x)=(x^2-sin(\pi x)+e^x)/(x-1)');     % add title
xlabel('x'); ylabel('y');                     % add labels
```

2. **script+function** (two separate files)

   `myplot2.m` (driver script)

```
x=0:.01:2;                                    % x-values
y=myfunction(x);                              % evaluate myfunction at x
plot(x,y,'r-','LineWidth',2);                 % plot in red
axis([0,2,-10,20]); grid on;                  % set range and add grid
title('f(x)=(x^2-sin(\pi x)+e^x)/(x-1)');     % add title
xlabel('x'); ylabel('y');                     % add labels
```

   `myfunction.m` (function)

```
function y=myfunction(x)                      % defines function
y=(x.^2-sin(pi.*x)+exp(x))./(x-1);            % y-values
```

3. **function+function** (one single file)

   `myplot1.m` (driver script converted to function + function)

```
function myplot1
x=0:.01:2;                                    % x-values
y=myfunction(x);                        % evaluate myfunction at x
plot(x,y,'r-','LineWidth',2);                 % plot in red
axis([0,2,-10,20]); grid on;                  % set range and add grid
title('f(x)=(x^2-sin(\pi x)+e^x)/(x-1)');     % add title
xlabel('x'); ylabel('y');                     % add labels
%-----------------------------------------
function y=myfunction(x)                      % defines function
y=(x.^2-sin(pi.*x)+exp(x))./(x-1);            % y-values
```

In case 2 `myfunction.m` can be used in any other `m`-file (just as other predefined MATLAB functions). In case 3 `myfunction.m` can be used by any other function in the same `m`-file (`myplot1.m`) only. Use 3 when dealing with a single project and 2 when a function is used by several projects.
★ Note that the function `myplot1` does not have explicit input or output arguments, however we cannot use a script since the construct *script+function* in one single file is not allowed.
★ It is convenient to add descriptive comments into the script file. Anything appearing after `%` on any given line is understood as a comment (in green in the MATLAB text editor).
★ To execute a script simply enter its name (without the .m extension) in the Command Window (or click on the SAVE & RUN button ▶).

The function `myfunction` can also be used independently if implemented in a separate file `myfunction.m`:

```
>> x=2; y=myfunction(x)
y =
  11.3891
```

A script can be called from another script or function (in which case it is local to that function).

If any modification is made, the script or function can be re-executed by simply retyping the script or function name in the Command Window (or use the up-arrow on the keyboard to browse through past commands).

IMPORTANT REMARK
By default MATLAB saves files in the Current Folder. To change directory use the Current Directory box on top of the MATLAB desktop.

★ A function file can contain a lot more than a simple evaluation of a function $f(x)$ or $f(t,y)$. But in simple cases $f(x)$ or $f(t,y)$ can simply be defined using the `inline` syntax.

For instance, if we want to define the function $f(t,y) = t^2 - y$, we can write the function file `f.m` containing

```
function dydt = f(t,y)
dydt = t^2-y;
```

and, in the command window, we can evaluate the function at different values:

```
>> f(2,1)    % evaluate the function f at t = 2 and y = 1
ans =
 3
```

or we can define the function directly on the command line with the `inline` command:

```
>> f = inline('t^2-y','t','y')
f =
   Inline function:
   f(t,y) = t^2-y
>> f(2,1)    % evaluate the function f at t = 2 and y = 1
ans =
 3
```

However, an inline function is only available where it is used and not to other functions. It is not recommended when the function implemented is too complicated or involves too many statements.

Alternatively, the function can be entered as an *anonymous* function

```
>> f = @(t,y)(t^2-y)
```

★ CAUTION!

- The names of script or function M-files must begin with a letter. The rest of the characters may include digits and the underscore character. You may not use periods in the name other than the last one in '.m' and the name cannot contain blank spaces.
- Avoid name clashes with built-in functions. It is a good idea to first check if a function or a script file of the proposed name already exists. You can do this with the command `exist('name')`, which returns zero if nothing with name `name` exists.
- *NEVER name a script file or function file the same as the name of the variable it computes.* When MATLAB looks for a name, it first searches the list of variables in the workspace. If a variable of the same name as the script file exists, MATLAB will never be able to access the script file.

Exercises

Instructions:

You will need to record the results of your MATLAB session to generate your lab report. Create a directory (folder) on your computer to save your MATLAB work in. Then use the Current Directory field in the desktop toolbar to change the directory to this folder. Now type

```
diary lab1_yourname.txt
```

followed by the Enter key. Now each computation you make in MATLAB will be save in your directory in a text file named lab1_yourname.txt. When you have finished your MATLAB session you can turn off the recording by typing `diary off` at the MATLAB prompt. You can then edit this file using your favorite text editor (e.g. MS Word).

Lab Write-up: Now that your diary file is open, enter the command `format compact` (so that when you print out your diary file it will not have unnecessary blank lines), and the comment line

```
% MAT 275 MATLAB Assignment # 1
```

Include labels to mark the beginning of your work on each part of each question, so that your edited lab write-up has the format

```
% Exercise 1
        .
        .
% Exercise 2
```

Final Editing of Lab Write-up: After you have worked through all the parts of the lab assignment you will need to edit your diary file.

- Remove all typing errors.
- Unless otherwise specified, your write-up should contain the MATLAB input commands, the corresponding output, and the answers to the questions that you have written.
- If the exercise asks you to write an M-file, copy and paste the file into your diary file in the appropriate position (after the problem number and before the output generated by the file).
- If the exercise asks for a graph, copy the figure and paste it into your diary file in the appropriate position. Crop and resize the figure so that it does not take too much space. Use ";" to suppress the output from the vectors used to generate the graph. Make sure you use enough points for your graphs so that the resulting curves are nice and smooth.
- Clearly separate all exercises. The exercises' numbers should be in a larger format and in **boldface**. Preview the document before printing and remove unnecessary page breaks and blank spaces.
- Put your name and class time on each page.

Important: An unedited diary file without comments submitted as a lab write-up is not acceptable.

1. All points with coordinates $x = r\cos(\theta)$ and $y = r\sin(\theta)$, where r is a constant, lie on a circle with radius r, i.e. satisfy the equation $x^2 + y^2 = r^2$. Create a row vector for θ with the values $0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi$, and $\frac{5\pi}{4}$.
   Take $r = 2$ and compute the row vectors x and y. Now check that x and y indeed satisfy the equation of a circle, by computing the radius $r = \sqrt{x^2 + y^2}$.
   Hint: To calculate r you will need the array operator `.^` for squaring x and y. Of course, you could also compute x^2 by `x.*x`.

2. Use the `linspace` command or the colon operator `:` to create a vector `t` with 91 elements: $1, 1.1, 1.2, \ldots, 10$ and define the function $y = \dfrac{e^{t/10}\sin(t)}{t^2+1}$ (make sure you use ";" to suppress the output for both `t` and y).

   (a) Plot the function y in black and include a title with the expression for y.

   (b) Make the same plot as in part (a), but rather than displaying the graph as a curve, show the unconnected data points. To display the data points with small circles, use `plot(t,y,'o')`. Now combine the two plots with the command `plot(t,y,'o-')` to show the line through the data points as well as the distinct data points.

3. Use the command `plot3(x,y,z)` to plot the circular helix $x(t) = \sin t,\ y(t) = \cos t,\ z(t) = t$ $0 \le t \le 20$.
   NOTE: Use semicolon to suppress the output when you define the vectors t, x, y and z. Make sure you use enough points for your graph so that the resulting curve is nice and smooth.

4. Plot $y = \cos x$ in red with a solid line and $z = 1 - \frac{x^2}{2} + \frac{x^4}{24}$ in blue with a dashed line for $0 \le x \le \pi$ on the same plot.
   Hint: Use `plot(x,y,'r',x,z,'--')`.
   Add a grid to the plot using the command `grid on`.
   NOTE: Use semicolon to suppress the output when you define the vectors x, y and z. Make sure you use enough points for your graph so that the resulting curves are nice and smooth.

5. The general solution to the differential equation $\dfrac{dy}{dx} = x + 2$ is

$$y(x) = \frac{x^2}{2} + 2x + C \quad \text{with } y(0) = C.$$

   The goal of this exercise is to write a `function` file to plot the solutions to the differential equation in the interval $0 \le x \le 4$, with initial conditions $y(0) = -1, 0, 1$.
   The `function` file should have the structure `function+function` (similarly to the M-file `myplot1.m` Example 3, page 5). The function that defines $y(x)$ must be included in the same file (note that the function defining $y(x)$ will have two input arguments: x and C).
   Your M-file should have the following structure (fill in all the `??` with the appropriate commands):

```
function ex5
x =  ?? ;       % define the vector x in the interval [0,4]
y1 = f(??);  % compute the solution with C = -1
y2 = f(??);  % compute the solution with C = 0
y3 = f(??);  % compute the solution with C = 1
plot(??)     % plot the three solutions with different line-styles
title(??)    % add a title
legend(??)   % add a legend
end

function y = f(x,C)
y = ??  % fill-in with the expression for the general solution
end
```

   Plot the graphs in the same window and use different line-styles for each graph. To plot the graphs in the same window you can use the command `hold on` or use the plot command similarly to Exercise 4.
   Add the title 'Solutions to dy/dx = x + 2'.
   Add a legend on the top left corner of the plot with the list of C values used for each graph.

(Type `help plot` for a list of the different line-styles, and `help legend` for help on how to add a legend.) Include both the M-file and the plot in your report.
NOTE: the only output of the function file should be the graph of the three curves. Make sure you use enough points so that the curves are nice and smooth.

6. (a) Enter the function $f(x,y) = x^3 + \dfrac{ye^x}{x+1}$ as an `inline` or *anonymous* function (see page 6). Evaluate the function at $x = 2$ and $y = -1$.

   (b) Type `clear f` to clear the value of the function from part (a). Now write a *function* M-file for the function $f(x,y) = x^3 + \dfrac{ye^x}{x+1}$. Save the file as `f.m` (include the M-file in your report). Evaluate the function at $x = 2$ and $y = -1$ by entering `f(2,-1)` in the command window.