**Assignment 1** **Dated:12/01/2016**

## MATLAB basics for Computational Physics Course CS-201

**Practice all the following commands** and other important tasks discussed during the lectures 2 and 3.

1. LAB207: OS-Scientific Linux; Open the Command prompt of matlab.
2. Use it as calculator: 1+2; **sin(60);** pi etc. Matlab's trigonometric functions are permanently set to radians mode.
3. Type "ans" to see the result.
4. Use scientific syntax for large numbers: e.g.: 1.11e15; 1.2e-2
5. The up-arrow key will display previous commands.
6. 2 important variables: matrix and strings.
7. a=25 [creates the variable a and assigns the value 25 to it]
8. try a=a+1
9. type a; see the result.
10. String variable: **s='This is me'**
11. Numbers in MATLAB are double precision. (15 digits of accuracy)
12. Try → format long e
13. Type a and see
14. Try the following:

```
format short % the default format
format long
format long e
format short e
```

15. To erase all the variables in the workspace, type **clear**.
16. One variable → clear a
17. The "Mat" in Matlab stands for matrix, and it treats all numeric variables as matrices.
18. For instance, Matlab thinks the number 2 is a 1x1 matrix:

```
N=2
size(N)
look at ouput
```

19. 1x4 row matrix using commas and braces
20. a=[1,2,3,4]

```
size(a)
```

21. 4x1 column matrix with semicolons and braces
22. b=[1;2;3;4]
    size(b)

23. Store a 3x3 matrix;
    A=[1,2,3;4,5,6;7,8,9]
    size(A)

24. To access individual matrix elements;
    A(row,column)
    A(3,3)

**25. Use of Colon(:) command → important to remember**
    To build large, evenly spaced arrays using the colon (:) command.
    To build an array x of values starting at $x=0$, ending at $x=10$, and having a step size of
    $dx = 0.5$ type this:

**x=0:.5:10**

And if you leave the middle number out of this colon construction, like **this t=0:20** then Matlab assumes a step size of 1.

26. Selecting specific columns from a particular row.
    c=A(2,1:2)
27. b=A(1:end,3)
28. very common to select all the row of a particular column
    b=A(:,3)
29. s='This is a string'
    s(1:7)

Matrix addition, multiplication, divison very easy

30. a=[0:10]; b=[0:2:20]; a+b
31. A=[1,2,3;4,5,6;7,8,9]
    B=[3,2,1;6,4,5;8,7,9]
    A+B
    A-B

32. C=[1,2;2,3]
    B=[1,2;2,3]

    C*B

**what if C.*B**

C^2; C^n

33. Divide individual elements from a matrix (treat it as scalars)
    a(1,2)/b(2,2)

**34. Complex Arithmatic**

35. The variable "i" is the usual imaginary number. I=(-1)^1/2; so do not assign value to it
36. a=1+2i
    b=2-3i

    now try a+b and other operations
37. real(a+b); imag(a+b)
38. exp(i*pi/4) ; Eulers formula exp(ix)
39. Matlab functions → look it by yourself
40. Length, size,floor,trig functions
41. **"clc" and "clear" difference**

## Scripts or m-files

1. Containing sequences of Matlab commands to be executed over and over again.

2. When you run a script, the commands in the file are executed from top to bottom just as if you had typed them on the command screen. Before you can execute a script, you need to point Matlab's current folder to the place where your script is saved.

3. execute your script by typing the name of your file without the .m extension in the command window. For "test.m" only test

4. script name should not start with number

5. You should nearly always begin your scripts with the following two commands:
   clear;
   close all;

6. The close all command closes any figure windows that are open.

7. Any line in a script that ends with a semicolon will execute without printing to the screen. For example, add these two lines of code to your test script a=sin(5); b=cos(5). May result in slow execution.

8. To have a script request and assign a value to the variable N from the keyboard, put the command
   N=input(' Enter a value for N - ')
   in your script and run it again. Note that Matlab is asking for input in the Command Window. If you enter a single number, like 2.7, then N will be a scalar variable. If you enter an array, like this: [1,2,3,4,5], then N will be an array. If you enter a matrix, like this:
   [1,2,3;4,5,6;7,8,9], then N will be a 3x3 matrix.
And if you don't want the variable you have entered to echo on the screen, end the input command line with a semicolon.

9. **For Output**
   fprintf(' N =%g \n',500)

**Add Comments**

Document your scripts by including comment lines that begin with %, like this:

1. % This is a comment line

**Wrap Long Lines**

2. Make your code more readable by continuing long program lines onto successive
3. lines by using the ... syntax, like this
4. a=sin(x)*exp(-y) + sqrt(b^2-4*a*c)/2/a + c1*d3 +...
   log(z) + sqrt(b);

**Entering a Matrix**

When matrices become large the comma and semicolon way of entering them is awkward. A more visual way to type large matrices in a script is to use spaces in place of the commas and to press the Enter key at the end of each row in place of the semicolons, like this:

```
A = [ 1 2 3 4
      5 6 7 8
      9 10 11 12
     13 14 15 16]
```

**Breakpoints and Stepping for debugging**

It is very helpful in this debugging process to watch what the script does as it runs, and to help you do this Matlab comes with two important features: breakpoints and stepping. Try and practice this.

## Pause:

A pause command in a script causes execution to stop temporarily. To continue just hit Enter.
You can also give pause a time argument like this

```
pause(.2)
```

which will cause the script to pause for 0.2 seconds, then continue.

Write some arbitrary snippet to understand the following:

## Loops and Logic

```
for n=1:N
% Put code here
End
```

### Logic command

```
if a>0
c=1 % If a is positive set c to 1
else
c=0 %if a is 0 or negative, set c to zero
end
```

### While loop

```
while term > 1e-10 % loop until term drops below 1e-10
n=n+1; % add 1 to n so that it will count: 2,3,4,5,...
term=1/n^2; % calculate the next term to add
s=s+term; % add 1/n^2 to s until the condition is met
end % end of the loop
```

## Plotting (only basic is given here; Explore more by yourself)

Linear plots; x vs y
We need arrays and plot command

To build an array *x* of *x*-values starting at *x* Æ 0, ending at *x* Æ 10, and having a step size of
.01 type this:

```
x=0:.01:10;
y=sin(5*x);

length(x)
length(y)

plot(x,y);
```

### Controlling the Axes

Matlab chooses the axes to fit the functions that you are plotting. You can override this choice by specifying your own axes, like this:

```
axis([0 10 -1.3 1.3])
```

Or, if you want to specify just the *x*-range or the *y*-range, you can use xlim:

```
plot(x,y)
xlim([0 10])
```

### Logarithmic Plots (assignment)

### Plot appearance; line color; labeling

```
xlabel('Distance (m)')
```

### 3-D Line Plots

Matlab will draw three-dimensional curves in space with the plot3 command.
Do a spiral on the surface of a sphere; spherical to rectangular coordinates as shown in the class.

## Multiplot

You may want to put one graph in figure window 1, a second plot in figure window
2, etc. To do so, put the Matlab command figure before each plot command,
like this

```
x=0:.01:20;
f1=sin(x);
f2=cos(x)./(1+x.^2);
figure
plot(x,f1)
figure
plot(x,f2)
```

## Subplots

put multiple plots in the same figure, but on separate axes.
The command to produce plots like this is subplot, and the syntax is

subplot(rows,columns,plot number)

```
subplot(2,1,1)
plot(x,f1)
subplot(2,1,2)
plot(x,f2)
```

## Grids and Plots in Multiple Dimensions (assignment as discussed during class)

Surface plot, contour plots etc.
Display functions of the type $F(x, y)$
Vector plots

### Making 2-D Grids

display 2-dimensional data → we need to define arrays X and Y that span the region that you want to plot, then create the function $F(x, y)$ over the plane.
First, you need to understand how Matlab goes from one-dimensional arrays x and y to two-dimensional matrices X and Y using the commands meshgrid and ndgrid.

size(x)
size(y)

size(X)
size(Y)

size(F)

look at “ndgrid”

**contour(X,Y,F)**

**zoom**

**what is default viewpoints**

**Streamlines**
For fluid dynamics, streamlines show the path that a particle would follow if the arrows at each point represent the fluid velocity at that point.

quiver(x,y,u,v)

**Functions**
Matlab will let you define expressions inside a script for use as functions *within that script only*.

The second line in this listing creates a function f that is stored only in memory. The symbol @ tellsMatlab that the variable f contains a reference to a function rather than a numeric value, and the items in parentheses afterwards indicate that this function takes two arguments: x and y. The code sin(x.*y)./(x.^2+y.^2) defines how the input values are used to create output values.

**M-file functions** are subprograms stored in text files with .m extensions. A function is different than a script in that the input parameters it needs are passed to it with argument lists like Matlab commands. Practice.

**Practice all the small assignments discussed during the lectures.**