i) Setting up - basic operations

Create a matlab directory where you can store the temporary files we will be using for this tutorial and enter this directory.

```
        mkdir Matlab <enter>
        cd Matlab <enter>
```

a) starting matlab

To start matlab you need to enter the following

```
        add matlab <enter>
        matlab & <enter>
```

You will probably need to use matlab a lot, so you may want to put the following statement in your `.environment` file which resides in your home directory:

```
        add matlab
```

You can do this using emacs. Then you only have to type

```
        matlab & <enter>
```

to start matlab.

b) getting help

If you type `help` at the matlab prompt, you will get a listing of all the topics matlab supports with the help command. You can then type `help` *topic-name* .

You can also use `lookfor` *topic-name* to get information on a topic.

c) easy math

Enter the following to get a feel of how matlab works:

```
        10+8+6          % Addition
        25-4            % Subtraction
        11*0.9          % Multiplication
        17/2            % Division
        2\17
```

d) using variables

Variables are case sensitive and can only contain letters, numbers and the underscore character. Try assigning the following variables:

```
        E = 30000
        area_1 = 3
        length_1 = 120

        k_1 = E*area_1 / length_1
```

clearing variables:

You can clear variables by using:

```
 clear name_of_variable#1 name_of_variable#2
```

Entering `clear` without any arguements will delete all of the variables that have been assigned in the matlab session.

e) using functions provided by matlab

Matlab has an enormous number of functions you can use. These include everything from mathematical functions to graphing functions. You can find our how to use a specific function by using help.

```
        help elfun  &ltenter>
        help exp    &ltenter>
        exp(1)    &ltenter>
```

f) vectors, arrays and matrices

Vectors and matrices are entered in brackets. Semi-colons are used to separate rows. Example

```
        x = [ 1 2 3 4 ]
        x = [ 1; 2; 3; 4 ]
        x = (1:5)
        x = (3:-1:1)
        A = [ 1 2 3; 4 5 6 ; 7 8 9]

        y = [ 5 6 7 ]

        z = [ x y ]   This vector is made of two vectors
        z = [ x y ; y x ]

        z'            Transpose of a Matrice
```

When using m-files (we'll go over this later) the following also works:

```
        A = [ 1 2 3
              4 5 6
              7 8 9 ]
```

Try the following operations:

```
        5*z
        z*z'
```

^top

ii) m- files

Rather than entering your operations in to matlab line by line, you can create an m-file which does these operations for you. In essence, what you are creating is a program. When you feel you need to, you can also write functions just as in a C program. You can create an m-file using emacs or any other text editor. The m-file must have the .m extension. Make sure that it is in the Matlab path, or that your m-file is in the same directory you started Matlab from. You can enter `cd pwd ls` inside of matlab to check this. Create a file called example1.m in emacs:

```
        emacs example1.m &
```

Enter the following information in the M-file:

```
clear
K = zeros(4,4);
        k = 1;

        k_el=[ k+k, -k;         % Forms the element matrix.
                -k, k ];


        k_el(1,1);              % get the value of the first element.


        K( 1:2,1:2 ) = k_el;    % Form the K ( Stiffness ) matrix
        K( 2:3,2:3 ) = k_el;
        K( 3:4,3:4 ) = k_el;
        K - K';                  % Check if K is symmetric.
        det(K);                  % Check that K is non-singular

        P = 0.25: 0.25 : 1;
        P = P';


        % Now, Let us solve for U

        U = K\P;                 % This is one way of doing it.

        U = inv(K)*P;            % This is another way of doing it.
                                 % The first way is the better one of
                                 % the two.  Requires: less memory and
                                 % less time, and also the error is less !


        % Check if K is a Positive Definite Matrix:

        eigenvalues_of_K = eig( K );

        % Strain Energy

        strain_energy =1/2* U'*K*U;
```
Now save this file and type `example1` in Matlab. Make sure you omit the .m at the end of this file name. Matlab will process all the commands entered. Use `whos` to get a list of variables. The % symbol is used for comments. The semi-colon at the end of the line surpresses the operations output on the matlab terminal. Omit the semi-colon and the results of the operation will be printed on the matlab teminal.

^top

iii) Decision making in matlab (for loops, if-else-end structures while loops)

As you develop programs with Matlab, you will probably need to use some of these structures. FOR LOOP
```
        y = zeros(20,1);

        for x=1:1:20        % Note that the step is 1 by default.
          y(x) = exp(-x)  ;
        end
        y
```
WHILE LOOP
```
        z = zeros(20,1);
        t = 1;
        while t <= 20
                z(t) = exp(t);
                t = t + 1;
        end
```
IF STATEMENT
```
        a = input('Enter an integer between 1 and 10, 0 to stop: ');
        Test = 1;
        while(a ~= 0)
                if(a == 7)
                        ('Correct')
                elseif( a <0 | a> 10)
```

```
                        ('Not an integer between 1 and 10')
                else
                        ('Wrong Guess')
                end
                a = input('Enter an integer between 1 and 10, 0 to stop: ');
        end
```

^top

iv) Plotting functions

You can now plot the functions created with the above while loop and for loop.

We have 20 values for y and z. Let's plot them as a function of x. We need to create values for the x axis.

```
x = (1:20)'

plot(x,y)
xlabel('x')
ylabel('y')

figure
plot(x,z)
xlabel('x')
ylabel('z')
```

^top