David Cohen (david.cohen@chalmers.se) Chalmers & GU

COMPUTER LAB: INTRODUCTION

This computer lab is inspired by the one given by Fardin Saedpanah in 2019.

Goals. In this first computer lab, we recall vector/matrix calculation, $2D$ and $3D$ plots for functions with one and two variables, respectively. We shall also plot a function $u(x, t)$ which is given in terms of a series.

A MATLAB tutorial and a guide can be found here and here (via Chalmers library).

In case of problems, don't forget to first read the error messages (if you get some) and to use the commands `doc` *commandname* or `help` *commandname* before contacting an assistant. Check your variables with the command `whos`. You can get the value of a particular variable at any times by typing its name.

1. Vector/Matrix calculation

Vectors and matrices can be created in various ways in MATLAB.

A simple way to create a vector is using `colon` (:). You can get help from MATLAB for `colon`, or any other MATLAB functions, by typing `doc colon` or `help colon`. Try to use both `doc` and `help` in order to get a better understanding of what these two MATLAB functions are doing.

Here is an example of creating three 1×4 vectors denoted by a, b, and c:

```
>> a=1:4; b=1:0.5:2.5; c=2*b;
```

What do you get if instead of writing semi-colon (;), you use comma (,) between the commands? That is, if you run the following:

```
>> a=1:4, b=1:0.5:2.5, c=2*b
```

Now, perform the following commands to see some simple vector/vector operations. Which one of these commands is not a correct MATLAB command?

```
>> a * c
>> a .* c
>> a^2
>> a .^ 2
>> a' .^ 2
>> a' * c
>> a * c'
>> sum(a)
```

Note that `.*` and `.^` are component-wise operators.

In order to know the size of a given vector, one can use the MATLAB functions `length` and `size`:

```
>> length(a), size(a)
```

Try to understand their differences.

We shall now create some matrices in MATLAB. Try the following commands in order to see how they work (remember that you can use `doc` or `help` if you need help to understand each command):

```
>> [a c], [a,c]
>> [a;c]
>> diag(a)
>> ones(3), ones(3,2)
>> zeros(3), zeros(3,2)
>> eye(3), eye(4,3)
>> A=[1 2 3; 4 5 6; 7 8 9; 10 11 12]
>> B=[diag(a) zeros(4,1) ; ones(1,5)]
```

The next example illustrates how one can access specific row(s) or column(s) of a matrix. Try the following:

```
>> A(1,:), A(2,:), A(:,2), A(2:3, :)
>> B(end,:), B(:,end), B(:, 1:3)
```

It is now time to test some vector/matrix manipulations. Try the following:

```
>> C=repmat(a,3,1), D=repmat(a,3,2), E=repmat(a,1,2)
>> F=reshape(E,2,4), G=reshape(E,4,2)
```

Don't forget to use the help functions to know what the above is doing.

What happens with the following code?

```
>> sort(E), sum(E)
>> sum(G), sum(G,1), sum(G,2)
```

And the last very useful command, related to matrices and vectors, is the command `mldivide` or simply \. Can you guess what is does? Feel free to try this command.

To finish this section, answer the following exercise.

Exercise 1. Create the following matrix in one line command:

$$A = \begin{bmatrix} 1 & 8 & 0 & 0 & 0 & 0 \\ -1 & 2 & 8 & 0 & 0 & 0 \\ 0 & -1 & 3 & 8 & 0 & 0 \\ 0 & 0 & -1 & 4 & 8 & 0 \\ 0 & 0 & 0 & -1 & 5 & 8 \\ 0 & 0 & 0 & 0 & -1 & 6 \end{bmatrix}$$

2. Plots in 2D and 3D

First we recall how to plot the graph of a function of one variable $y = f(x)$, $x \in [a, b]$. To this end, we need a partition for the domain $[a, b]$. That is, we divide the interval $[a, b]$ into small sub-domains. This can be done, for instance, by considering a mesh step h, and then use `colon` (:), as:

```
>> x=a:h:b;
```

An other possibility is using `linspace` with some positive integer N:

```
>> x=linspace(a,b,N);
```

Note that, if one chooses $h = \frac{b-a}{N-1}$ then the vectors x in both examples would be the same.

Let us try this on a concrete example. We would like to plot the function $y = \sin(x)$, $x \in [-\pi, \pi]$. To do so, we proceed as follows

```
>> x=-pi:0.2:pi; y=sin(x); plot(x,y)
```

Another possibility would be to use anonymous functions with the `@` operator:

```
>> x=-pi:0.2:pi; yy=@sin; plot(x,yy(x))
```

Observe that the above also works for functions of several variables

```
f = @(x,y,z) x.^2 + y.^2 - z.^2
```

Finally, let us illustrate that one can use more options in the plot:

```
>> x=-pi:0.2:pi; y=sin(x); plot(x,y,'bo-')
>> title('2D-plot')
>> xlabel('x'); ylabel('y')
```

If one wants to plot more than one functions in one `figure`, one could do the following:

```
>> % plot y1(x)=sin^2(x), y2(x)=sin(x)+x^2, for x in [-5,5]
>> x=-5:0.2:5; y1=sin(x).^2; y2=sin(x)+x.^2;
>> plot(x,y1,'bo-',x,y2,'r*--');
>> title('2D-plot')
>> xlabel('x'); ylabel('y')
>> legend('y1','y2')
```

See `doc plot` for more examples and options of `plot`. You can also find more MATLAB commands which are related to `plot`, at the bottom of the help page, in the section 'See Also'.

Let us now, plot a surface defined by a function of two variables $u = f(x, y)$, $x \in [a, b]$, $y \in [c, d]$. To this end, we have to compute the values of the function at some grid points. As done in $1D$, we first divide the domain $[a, b] \times [c, d]$ into sub-domains. This is can be done, in one shot, using the function `meshgrid` for example:

```
>> [x,y]=meshgrid(a:h:b,c:k:d);
```

where h and k are some mesh steps.

We can now compute the values of the function $f(x, y)$ at the grid by using vector/vector multiplications. Another (slower) possibility could be to use `for`-loops to compute all the values of the function. Let us look at a concrete example in more details.

We want to find the values of $f(x,y) = \sin(x)\sin(y)$ for $x \in [0,5]$, $y \in [0,10]$. To do this, one could use the following

```
>> [x,y]=meshgrid(0:.2:5,0:.1:10);    % We choose h=0.2 and k=0.1
>> f=sin(x).*sin(y);
```

As written above, another possibility could be to use `for`-loops:

```
>> x=0:.2:5; y=0:.1:10;    % We choose h=0.2 and k=0.1
>> for i=1:length(x)
>>     for j=1:length(y)
>>         f(j,i)=sin(x(i))*sin(y(j));
>>     end
>> end
```

Using the above grid, one can plot the surface $u = f(x,y)$ as:

```
>> surf(x,y,f) % One has u=f(x,y)
>> xlabel('x'); ylabel('y')
```

One can also use MATLAB's function `mesh` to plot a surface. Use it and compare your result with the use of `surf`.

Let us finish this first lab session with an exercise.

Exercise 2. Let p, L, T be some positive parameters. Consider the function of one variable

$$f(x) = \begin{cases} \frac{2p}{L}x & 0 \le x \le \frac{L}{2}, \\ \frac{2p}{L}(L-x) & \frac{L}{2} \le x \le L. \end{cases} \tag{1}$$

And the function of two variables

$$u(x,t) = \frac{8p}{\pi^2}\sum_{n=1}^{\infty}\frac{1}{n^2}\sin\left(\frac{n\pi}{2}\right)\cos\left(\frac{n\pi}{L}t\right)\sin\left(\frac{n\pi}{L}x\right), \tag{2}$$

for $x \in [0,L]$ and $t \in [0,T]$.

Set $p = 1, L = 6$ and $T = 10$.

(a) Plot the function (1). What do you observe at $x = \frac{L}{2}$? Yes, the first derivative of this function does not exist at $x = \frac{L}{2}$ (the curve is sharp at that point).

(b) Plot the $3D$ graph of the function (2) for $(x,t) \in [0,L] \times [0,T]$. Note that one has to truncate the series in (2), say at $N = 100$ terms. Remember to use MATLAB command `surf` to produce $3D$ plots. Do not forget to give the proper names to the "xlabel" and "ylabel" (the variables x and t).

(c) Now, we want to see the oscillatory behaviour of $u(x,t)$ in $2D$.
Let $0 = t_1 < t_2 < \cdots < t_M = T$ be the partition of the time interval $[0,T]$ that was used in part (b). In order to see these oscillations, in a `for` loop, plot $u(x,t_j)$ for $j = 1, 2, \cdots, M$, using `hold on` and `pause(0.05)`.

3. M-files

A good practice, when programming, is to use functions or routines, These objects accept input arguments and return output arguments. One does not need to write a MATLAB function (or M-file for short) in the MATLAB base workspace, one can write it in a separate file.

The following example illustrates the basic parts of an M-file (create the file myfactorial.m and write the following in it):

```
function f = myfactorial(n)
% myfactorial(n) returns the factorial of n.
f = prod(1:n);
end
```

To compute 5!, one then input in MATLAB the following

```
>> myfactorial(5)
```