Heat transfer 2020 – EEN-E1020
Created by: Ville Vuorinen, Aalto University
Computer class: Oct 28th 2020

Matlab Primer

Background: On the heat transfer course, Matlab is used to illustrate heat transfer phenomena. Matlab assignments consist typically of three steps. Step 1) Creating data by simulation. Step 2) Saving the data. Step 3) Post-processing and visualizing the data. Step 1) involves typically changing some small thing (like heat conductivity) in a provided code and then running the code for e.g. 1 minute to create simulation data. Step 2) involves saving the data into a file to be used later. Step 3) involves writing a post-processing script where the saved data result is loaded from a file and plotted. The purpose of this document is to support the student to understand how Matlab can be used during the course by going through Tasks 0-5 during the first computer class. The basic ideas herein can be directly applied in HW1-5.

Task 0: Open Matlab and familiarize yourself with different functions and how they are used as follows by hitting enter after each command.

```
>> help plot
>> help load
>> help save
>> help imagesc
```

Task 1: Open Matlab and write the following text line by line to the command line (>>) and hit enter after each ";" (try also with the ";").

```
>> clear *; % clears all workspace variables if any

>> a = 3; % declaration of variable a to have value 3, "Step 1"

>> T=[273 274 275 276]; % 4 temperature values in 1 by 4 vector T, "Step 1"

>> M=[1 2 3; 4 2.1 2];      % some numbers in a matrix "Step 1"

>> save('MyFirstResults.mat', 'a','T','M'); % "Step 2" - results saved to file
                                            % MyFirstResults.mat
>> clear *; % clears all workspace variables

>> whos  % tells what are the existing workspace variables - now gives nothing
        % because clear *

>> load MyFirstResults.mat % Beginning of "Step 3"

>> whos % prints on the screen the variables a, T, M
```

Standpoint 1: At this point you understand a) Matlab command line basics, b) basics of how you can declare a variable, vector and matrix, c) saving data and d) loading data i.e. you understand the typical workflow of Steps 1-3 in the computer exercise.

Task 2: Working with .m files. It is very handy that you always create a .m file where you put all the text that you want to execute in Matlab. Repeat Task 1 by the following steps.

1) In Matlab terminal, open a new script from top left panel which opens a text editor in the window.

2) Copy-paste or write the commands above to the text editor.

3) Save the file to a proper file – e.g. `myfirstfile.m` – and folder - e.g. `HeatCompClass1`. IMPORTANT: remember the `.m` extension for file names.

```
% the file myfirstfile.m could start like this
clear *;
a = 3;
```

4)Run and execute the file by pressing the green "play" button.

Standpoint 2: At this point you understand a) what is a .m file, b) that when you work in Matlab, the text script is handy to store into a .m file to avoid repetition. In general, you understand key working environment in Steps 1-3.

Task 3: Plotting a graph in Matlab. We use f(x) = sin(x) and g(x)=sin(x)cos(x) examples.

1) In Matlab terminal, open a new script and save this script e.g. in a file called `MyFirstPlots.m`

2) Type the following commands to the file.

```
% file MyFirstPlots.m starts → first declare some font settings
set(0,'DefaultAxesFontName','Times New Roman')
set(0,'DefaultAxesFontSize',15)

clear *
N = 100; % 100 grid points

L = 2*pi; % length of the domain

x = linspace(0, L, N); % divide the domain 0<= x <= L into N points
size(x) % prints the size of x on the screen

f = sin(x); % evaluate function f
size(f)

g = sin(x).*cos(x); % evaluate function g. Note: .* is very important!

figure(1); % opens a new figure with label "1"; if you have many figs give
           % another number e.g. 1,2,3, etc
clf        % clf = clear figure = delete all old plots if any
plot(x,f,'o-')
hold on    % keep the previous plot in the figure with hold on
plot(x,g,'r-','Linewidth',2) % plot in red line with thicker line

legend('f(x)','g(x)')  % add a legend to mark which graph is which

print -dpng MyFirstPicture   % save the picture of figure(1) in a file
```

Standpoint 3: At this point you can create files for plotting pictures with relevance to Step 3. You could now easily have also loaded some data from e.g. some previous simulation Step 1 & 2 and started processing the data in Step 3. **Note:** the .* is very important for pointwise multiplication!

Task 4: for loop as a basic tool to loop over timesteps. Next, let us sum together numbers 1-5 in a for loop. Open a new file for this task as well (e.g. `myforloop.m`).

```
% here we go again with a new file to sum some numbers together
clear *
mysum = 0; % initialize a counter
for(k=1:5) % k is index number which gets the numbers 1,2,3,4,5 in
            % the loop
      k     % here, when we leave out the semicolon, k is printed on
            % the screen
      mysum = mysum + k; % add k to the previous value of mysum
      mysum              % plots mysum value to the screen
end                      % for loop ends
```

Standpoint 4: you can now use a for loop needed e.g. in HW1 to solve the Newton's cooling law.

Task 5: Using ideas from previous Tasks, create a .m file to solve a simplified version of Newton's cooling law and plot the result. Note that the file below is not complete, but you need to add the timestep, and plot the figure of temperatures.

```
% file starts, solve T'(t) = dT/dt = -(T-Tref)/tau;
clear *
T = 293;     % initialize temperature in Kelvins
Tref = 278; % fridge temperature

tau = 1;     % unit is s, this can be considered as "cooling" timescale
            % in Newton's cooling law tau is a combination of m, h, cp, etc
TotalSimuTime = xxxx; % xxxx = time in seconds
HowManySteps = round(TotalSimuTime/dt); % how many timesteps, it is logical that
                                        % HowManySteps*dt >> tau

for(k=1:HowManySteps)
      dT = -(dt/tau)*(T-Tref); % use Euler method to estimate dT
       T = T + dT;
       Ts(k) = T;        % store current temperature value to a vector
end                      % for loop ends i.e. Step 1 ends

                         % then do some plot of T(t) e.g. plot(Ts)
```

Standpoint 5: you can now use a for loop to solve the Newton's cooling law.

Task 6: Plotting a correlation function in Matlab. In HW1 you are asked to plot the Hilpert correlation function. You may benefit from the code below to get started.

```
% file PlotHilpert.m starts → first declare some font settings
set(0,'DefaultAxesFontName','Times New Roman')
set(0,'DefaultAxesFontSize',15)

clear *
N = 100; % 100 grid points to discretize Reynolds number

U = 1:100; % velocities, m/s
```

```
nuw = … ; % find here the kinematic viscosity of water
alphaw = … ; % find here the thermal diffusivity of water
rhow = …; % density of water
kw = …; % heat conductivity of water, k/(rho*cp) = alpha

% give similar parameters for air

Rew = U.*D/nuw; % water Reynolds numbers
Rea = … ; % air Reynolds numbers
Nuw = … ; % Nusselt number of water as a function of Rew and Prw
```