**DAILY ASSESSMENT FORMAT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **/5/2020** | **Name:** | **Abhishek Vasudev Mahendrakar** | |
| **Course:** | **Matlab Opramp** | **USN:** | **4AL17EC003** | |
| **Topic:** | **Chapter 8 to 14** | **Semester & Section:** | **6th-‘A’** | |
| **Github Repository:** | **ECEAbhishekVMahendrakar** | **E-mail:** | **abhi2244mahendrakar@gmail.com** | |
| **FORENOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **Report – Report can be typed or hand written for up to two pages.**  **Chapter 8: Obtaining Help**  **8.1 Obtaining help**   * The MATLAB documentation contains examples and information that can help you when working on your own problems.   **Chapter 9: Plotting Data**  **9.1 Plotting Vectors**   * Two vectors of the same length can be plotted against each other using the plot function. * The plot function accepts an additional argument that allows you to specify the color, line style, and marker style using different symbols in single quotes. * Notice that each plot command created a separate plot. To plot one line on top of another, use the hold on command to hold the previous plot while you add another line. * While the hold state is on, plots will continue to go on the same axes. To return to the default plot behavior, where each plot gets its own axes, enter hold off. * When you plot a single vector by itself, MATLAB uses the vector values as the *y-axis* data and sets the *x-axis* data to range from 1 to n (the number of elements in the vector). * The plot function accepts optional additional inputs consisting of a property name and an associated value. * You can provide additional inputs to the plot function after the line specifier   **9.2 Annotating plots**   * Labels can be added to plots using plot annotation functions, such as title. The input to these functions is a string. Strings in MATLAB are enclosed in double quotes ("). * You can add a legend to your plot using the legend function.   **Chapter 10: Review problems**  **10.1 Project- Electricity Usage**  **Code:**  load electricity  usage  usage(2,3) = 2.74  res = usage(:,1)  comm = usage(:,2)  ind = usage(:,3)  yrs = (1991:2013)'  plot(yrs,res,"b--")  hold on  plot(yrs,comm,"k:")  plot(yrs,ind,"m-.")  hold off  title("July Electricity Usage")  legend("res","comm","ind")  **10.2 Project- Audio Frequency**  Code:  load Cchord  n = numel(y)  t = 0:n-1  t = t/fs  plot(t,y)  yfft = abs(fft(y))  f = 0:n-1  f = f\*fs/n  plot(f,yfft)  xlim([0 1000])  **Chapter 11: Importing data**  **11.1 Importing data as table**   * To extract a variable from the table, you can use *dot notation*: * If you are working with a table, you might want to keep related data together. Instead of creating separate variables, you can assign the result of a calculation to a table. * If the variable *data.HeightMeters* doesn't exist, MATLAB will create a new variable in the table with the name HeightMeters. * You can interact with a table by clicking on it in the output pane of a live script. For example, you can sort a table using one of its variables. * Once you are happy with your table, you can make the changes permanent by updating the code in your script.   **Chapter 12: Logical Arrays**  **12.1 logical Indexing**   * [Relational operators](https://www.mathworks.com/help/matlab/matlab_prog/array-comparison-with-relational-operators.html), such as >, <, ==, and ~= perform comparisons between two values. The outcome of a comparison for equality or inequality is either 1 (true) or 0 (false). * You can compare a vector or matrix to a single scalar value using relational operators. The result is a logical array of the same size as the original array. * You can use a logical array as an array index, in which case MATLAB extracts the array elements where the index is true. The following example will extract all elements in v1 that are greater than six. * You can also use logical indexing with two different vectors. * You can use logical indexing to reassign values in an array. For example, if you wish to replace all values in the array x that are equal to 999 with the value 1, use the following syntax.   **Chapter 13: Programming**  **13.1 Programming Constructors**  **13.2 Decision Branching**   * The body of an if block is only executed if the condition is true. * Often in these situations, you may want to execute some other code if the condition is not met. To do this, you can use the else keyword   **13.2 For loops**   * When this code is run, the loop body will be executed three times, as the loop counter (c) progresses through the values 1:3 (1, 2, and 3).   **Chapter 14: Final Project**  **14.1 Project- Stellar Motion I**  Code:  lambdaEnd = lambdaStart + (nObs-1)\*lambdaDelta  lambda = (lambdaStart:lambdaDelta:lambdaEnd)'  s = spectra(:,2)  [sHa,idx] = min(s)  lambdaHa = lambda(idx)  hold on  loglog(lambdaHa,sHa,"rs","MarkerSize",8)  hold off  z = lambdaHa/656.28 - 1  speed = z\*299792.458  **14.2 Project- Stellar Motion II**  Code:  [sHa,idx] = min(spectra);  lambdaHa = lambda(idx);  z = lambdaHa/656.28 - 1;  speed = z\*299792.458  for v = 1:7  s = spectra(:,v);  if speed(v) <= 0  loglog(lambda,s,"--")  else  loglog(lambda,s,"LineWidth",3)  end  hold on  end  hold off  legend(starnames)  movaway = starnames(speed > 0) | | | |