**DAILY ASSESSMENT FORMAT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **13/07/2020** | **Name:** | **Abhishek Vasudev Mahendrakar** | |
| **Course:** | **Matlab Opramp** | **USN:** | **4AL17EC003** | |
| **Topic:** | **Chapter: 1 to 7** | **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 1: Course Overwiew**    **Chapter 2: Commands**  **2.1 Entering Commands**   * You can execute commands by entering them in the command window after the MATLAB prompt (>>) and pressing the **Enter** key. * Unless otherwise specified, MATLAB stores calculations in a variable named ans. * The equals sign (=) in MATLAB is the *assignment* operator, meaning that the expression on the right of the equals sign is assigned to the variable on the left. * **Workspace** window (on the right) shows all the variables currently in the workspace. * Adding a semicolon to the end of a command will suppress the output, though the command will still be executed, as you can see in the workspace. When you enter a command without a semicolon at the end, MATLAB displays the result in the command prompt. * You can recall previous commands by pressing the Up arrow key on your keyboard. Note that the **Command Window** must be the active window for this to work. * When you enter just a variable name at the command prompt, MATLAB returns the current value of that variable.   **2.2 Naming Variables**   * You can name your MATLAB variables anything you'd like as long as they **start** with a letter and contain only letters, numbers, and underscores (\_). * MATLAB variables are also case sensitive. * You can name all your variables a or x, but it is more useful to name your variables something meaningful.   **2.3 Saving and Loading Variables**   * You can save variables in your workspace to a MATLAB specific file format called a MAT-file using the save command. * When you switch to a new problem in MATLAB, you might want to tidy up your workspace. You can remove all variables from your workspace with the clear function. * You can load variables from a MAT-file using the load command. * Notice that the variable data is listed in the workspace. You can see contents of any variable by entering the name of the variable. * The clear function cleans up the workspace. You can use the clc command to clean up the **Command Window**.   **2.4 Using Built-in functions and Constants**   * Matlab contains built-in constants, such as pi. * MATLAB contains a wide variety of built-in functions, such as abs (absolute value) and eig   **Chapter 3: Matlab desktop and text editor**  **3.1 Matlab desktop and text editor**  **3.2 The Matlab editor**   * You can enter commands in a script by clicking on the gray code box. * When you're ready, you can submit your code by clicking the blue **Submit** button.   **3.3 Running scripts**   * This live script contains formatted text, code, and section breaks. In this course, scripts will include **Task** headers to show where you should enter your code. * You can test your code before submitting by running the script. To execute the entire script, click the **Run** button   **Chapter 4: Vector and Matrices**  **4.1 Manually entering arrays**   * A single number, called a *scalar*, is actually a 1-by-1 array, meaning it contains 1 row and 1 column. * You can create arrays with multiple elements using square brackets. * When you separate numbers by spaces (or commas) as shown in the previous task, MATLAB combines the numbers into a *row vector*, which is an array with one row and multiple columns (1-by-*n*). When you separate numbers by semicolons, MATLAB creates a *column vector* (*n*-by-1). * You can combine spaces and semicolons to create a *matrix*, which is an array with multiple rows and columns. When entering a matrix, you must enter them row by row. * In MATLAB, you can perform calculations within the square brackets.   **4.2 Creating Evenly-Spaced Vectors**   * It is common to create vectors containing evenly-spaced numbers * For long vectors, entering individual numbers is not practical. An alternative, shorthand method for creating evenly-spaced vectors is to use the : operator and specify only the start and end points. * The : operator uses a default spacing of 1, however you can specify your own spacing * If you know the number of elements you want in a vector (instead of the spacing between each element), you could instead use the linspace function: * Both linspace and the : operator create row vectors. However, you can convert a row vector into a column vector using the transpose operator ('). * You can create column vectors in a single command by creating the row vector and transposing it all on one line. Note the use of parentheses here to specify the order of operations.   **4.3 Array Creation Functions**   * MATLAB contains many functions that help you to create commonly used matrices, such as matrices of random numbers. * Many matrix creation functions allow you to input one number to create a square matrix (*n*-by-*n*) or input two numbers to create non square matrices.   **Chapter 5: Indexing into modifying array**  **5.1 Indexing into arrays**   * You can extract values from an array using *row, column indexing*. * You can use the MATLAB keyword end as either a row or column index to reference the last element. * Note that you can use arithmetic with the keyword end.   **5.2 Extracting Multiple elements**   * When used as an index, the colon operator (:) specifies all the elements in that dimension. * The colon operator can refer to a range of values. The following syntax creates a matrix containing the first, second, and third rows of the matrix A. * A single index value can be used to reference vector elements * A single range of index values can be used to reference a subset of vector elements.   **5.3 Changing values in arrays**   * Remember you can use the : character to extract entire columns of data. * Elements of a variable can be altered by combining indexing with assignment.   **Chapter 6: Arrays Calculations**  **6.1 Preforming array operations on vectors**   * MATLAB is designed to work naturally with arrays. For example, you can add a scalar value to all the elements of an array. * You can add together any two arrays of the same size. * You can multiply or divide all of the elements of an array by a scalar. * Basic statistical functions in MATLAB can be applied to a vector to produce a single output. The maximum value of a vector can be determined using the max function. * MATLAB has functions that perform mathematical operations on an entire vector or array of values in a single command. * The \* operator performs [matrix multiplication](http://www.mathworks.com/help/matlab/ref/mtimes.html). So, if you use \* to multiply two equally sized vectors, since the inner dimensions do not agree, you will get an error message. * In contrast, the .\* operator performs elementwise multiplication and allows you to multiply the corresponding elements of two equally sized arrays.   **Chapter 7: Calling Functions**  **7.1 Obtaining Multiple Outputs from Function Calls**   * The size function can be applied to an array to produce a single output variable containing the array size. * The size function can be applied to a matrix to produce either a single output variable or two output variables. Use square brackets ([ ]) to obtain more than one output. * The maximum value of a vector and its corresponding index value can be determined using the max function. The first output from the max function is the maximum value of the input vector. When called with two outputs, the second output is the index value. | | | |