|  |  |
| --- | --- |
| **Name:** | Harith Kolaganti |
| **Purdue Login:** | hkolagan |
| **Contributors:** |  |
| **Section Number:** | 005 |

|  |
| --- |
| **Instructions:**   1. Save all files to your **Purdue Career Account** in a folder specific to PS01. 2. Save this problem set document as **PS01\_AnswerSheet\_*yourlogin*.docx**, where *yourlogin* is your Purdue Career Account login.   Example: Student Kris Boiler (boiler62@purdue.edu) will save her file as PS01\_AnswerSheet\_*boiler62*.docx.   1. In **PS01\_AnswerSheet\_*yourlogin*.docx**,    * Enter your name, login, and section number above.    * If you work with others on this assignment, list their names on the Contributors line above. Leave blank only if you worked completely alone.    * Enter your answers to each question in the boxes provided as you work through the assignment. 2. Follow the instructions for each problem. Create appropriate files as instructed. 3. Submit this answer sheet and your MATLAB m-files, all zipped into one **.zip** file named PS01\_*yourlogin*.zip, to the Blackboard drop box for PS01 before the due date. |

**Problem Set**

This problem set contains a set of problems to help you become familiar with MATLAB and its basic use. Complete this Answer Sheet individually.

|  |  |  |
| --- | --- | --- |
| **Item** | **Type** | **Deliverable** |
| PS01 Answer Sheet | Individual | PS01\_AnswerSheet\_*yourlogin*.docx |
| Problem 1: Variables | Individual | *Answers on this Answer Sheet* |
| Problem 2: MATLAB as Calculator | Individual | *Answers on this Answer Sheet* |
| Problem 3: Script Files | Individual | *Answers on this Answer Sheet*  PS01\_Problem3\_*yourlogin*.m |
| Problem 4: Vectors | Individual | *Answers on this Answer Sheet*  PS01\_Problem4\_*yourlogin*.m |
| Problem 5: Matrix Manipulation | Individual | PS01\_Problem5\_*yourlogin*.m |

Note: In all filenames, replace *yourlogin* with your Purdue Career Account login.

**Submitting Deliverables**

Save all your deliverables in a folder specific to PS01. When you are ready to submit your work to Blackboard, compress the deliverables into one zipped folder and name it **PS01\_*yourlogin*.zip**. Upload the zip folder to the PS01 assignment dropbox and submit. Be sure that

* Only files you wanted graded are in your zipped folder. (Do not include any file that is not on the deliverable list unless it is necessary to run your code.)
* The zipped folder extension is **.zip** (Do not use any other compression file type)

**Programming Standards**

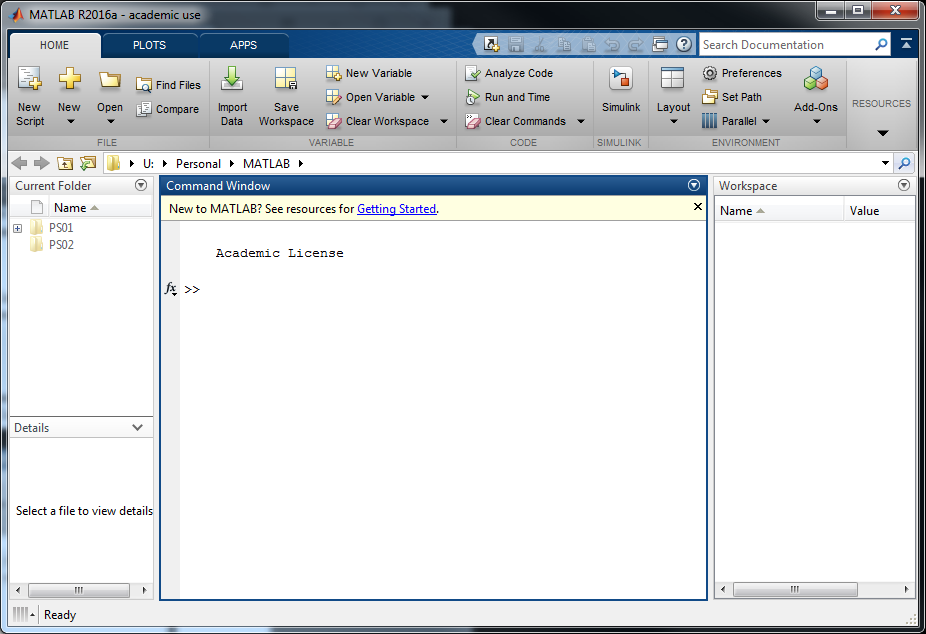
You are responsible for following all course Programming Standards guidelines, which you will need to know for this and all subsequent assignments. You can find the guidelines on Blackboard via **Course Info** (on the left-hand navigation bar) / **Course Resources** > **MATLAB Resources**.

**Help with MATLAB**

Go to **Course Resources** (on the left-hand navigation bar in Blackboard) / **Course Resources** > **MATLAB Resources** to find extra help for using MATLAB and Purdue Technology, and for a file called *MATLAB Reference 1: Basic & Complex Calculations.*

**MATLAB Interface**

MATLAB opens showing multiple interface panes. In general, you need to see the Command Window and the Workspace. Other visible panes may include the Command History and Current Folder. If you want to change the visible panes, click on the Layout button and select panes to add or hide.



# Problem 1: Variables

Individual Programming

**Step 1: Create and clear variables.**

1. In the Command Window, type each of the following lines of code (blue text only) followed by **Enter**:

>> **Height = 10**

>> **height = 22.3**

1. In the Command Window, type

>> **Height**

* 1. Why didn’t the value of **Height** change?

|  |
| --- |
| The different case makes Height a different variable than height |

1. In the Command Window, type **who**.
   1. What does this command do?

|  |
| --- |
| Displays what variables have been created |

1. Other than in the Command Window, where on the MATLAB interface do you see your variables listed?

|  |
| --- |
| Workspace tab on the right of the window |

1. What command could be used to delete the variable **height** (but not **Height**)?

|  |
| --- |
| clearvars height |

**Step 2: Apply rules for variable names.** To complete this section, refer to<http://www.mathworks.com/help/matlab/matlab_prog/variable-names.html>

1. Why is **2nd\_time not** a valid variable name?

|  |
| --- |
| The variable starts with a numeric digit, which is not allowed by standards |

What happens if that variable is used anyway? Try it; type **>> 2nd\_time = 1**

|  |
| --- |
| An error message comes up stating that the function inputted is an unexpected MATLAB expression |

1. You want to use a variable **i** to represent spectral intensity in a physics problem, but **i** is a built-in MATLAB constant - the imaginary number equal to the square root of (-1). What will happen if you type?

>> **i**

>> **i = 0.3**

>> **i**

|  |
| --- |
| The constant becomes a variable after becoming declared as 0.3 |

1. Now that you have completed Step 2, how would you use MATLAB to reset **i** to its original value?

|  |
| --- |
| clearvars i |

1. The course programming standards indicate that you must use descriptive variable names. When might it be appropriate to use **X** as a variable name?

|  |
| --- |
| When doing a computational based code, such as calculus |

1. MATLAB has multiple functions that can help you determine if certain names are available to be used as variable names.
   1. Type **help elfun** to see a list of built-in elementary math functions. You can also type **help variable\_name** or simply type **potential\_variable\_name** to see if the name is already defined in MATLAB. What are two other commands that can tell you if a variable name is already being used?

|  |
| --- |
| exist variable\_name , which variable\_name |

* 1. Is **distance** an appropriatevariable name?

|  |
| --- |
| Only if mapping toolbox isn’t licensed and installed, or it will be treated as a built-in function |

<continue to next page>

**Step 3: Identify inappropriate and invalid variable names**. *Inappropriate* variable names will work but either overwrite built-in MATLAB functions or violate programming standards. *Invalid* variable names will not work and will result in error messages.Complete the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Possible variable name** | **Select the ONE most appropriate box** | | | **Explain your selection** |
| **Okay** | **Inappropriate** | **Invalid** |
| M | X |  |  |  |
| 2cases |  |  | X | Error will appear |
| time | X |  |  |  |
| temp#2 |  |  | X | Error will appear |
| length | X |  |  |  |
| dist\_3mg | X |  |  |  |
| function |  |  | X | Error will appear |

|  |  |
| --- | --- |
| **Key Points About Variables:**   * Appropriate variable names are descriptive. * A valid MATLAB variable name starts with a letter and is followed by letters, digits, or underscores. * MATLAB will allow you to create a variable name with the same name as a built-in function, but this is inappropriate and must be avoided.   **Useful MATLAB Commands:** | |
| **Who** | List existing variables in current workspace. |
| **clear; clearvars** | Clear all variables from the workspace. |
| **clear *variable\_name***  **clearvars *variable\_name*** | Clear specified variable from the workspace. |
| **help elfun** | View MATLAB elementary math functions. |
| **help *possible\_variable\_name*** | Determine if ***possible\_variable\_name*** is a built-in function. |
| **exist *possible\_variable\_name*** | Check if variable names or functions are defined   * Returns “0” if ***possible\_variable\_name*** does not exist * Returns “1” if it is in the workspace * Returns “5” if it is a built-in function |
| **which *possible\_variable\_name*** | Identify if ***possible\_variable\_name*** exists or is a built-in function. |
| ***possible\_variable\_name*** | Returns an error if MATLAB does not recognize that name as a variable, function, etc. |

<continue to Problem 2>

# Problem 2: MATLAB as a Calculator

Individual Programming

**Step 1:** Complete the table below using MATLAB to calculate each expression. ***Be sure to verify your result by hand or another non-MATLAB method.***

|  |  |  |
| --- | --- | --- |
| **Mathematical Expression** | **MATLAB Command** | **Result** |
|  | (5+3\*(25 + 2)^(2/3))/sqrt(5+11)+6 | 14.0000 |
|  | 3^4+(4^3)/(12\*7)-10^2 | -18.2381 |
|  | abs(1 - 5.5\*sin(3\*pi)) + 2.2 | 3.2000 |

**Step 2:** Assume x = 0.1 and z = 8.2. Complete the table below. ***Be sure to verify your result by hand or another non-MATLAB method.***

|  |  |  |
| --- | --- | --- |
| **Mathematical Expression** | **MATLAB command** | **Result** |
|  | x\*z^2-((2\*z)/(3\*x))^(3/5) | -4.3075 |
|  | (0.5\*z)/(20\*x^3)+(exp(-x\*z))/(x+z) | 205.0531 |
| *c* = *ln*(*z*) | log(z) | 2.1041 |
| *d* = *log*(*z*) | log10(z) | 0.9138 |

***Hint:***

* ***ln*** *or* ***loge*** *is commonly known as the natural logarithm.*
* ***log*** *or* ***log10*** *is commonly known as log with base 10.*

|  |  |
| --- | --- |
| **MATLAB Calculations**  **Useful commands:** | |
| **log** | Y = log(X) returns the natural logarithm of the elements of X. |
| **log10** | Y = log10(X) returns the base 10 logarithm of the elements of X. |
| **exp** | Y = exp(X) returns the exponential for each element of X. |
| **sin** | Y = sin(X) returns the sin for each element of X. |

<continue to Problem 3>

# Problem 3: Script Files

Individual Programming

|  |
| --- |
| **What is a script file?**  A MATLAB script file (a type of m-file) is a file you can use to store a series of MATLAB commands, also called a program. Script files allow you to easily modify your code and run it from the Command Window without re-typing all of the commands. |

**Step 1: Prepare a script file.**

1. Open the MATLAB file **script\_header\_template.m**.
2. Save the file with the name **PS01\_Problem3\_*yourlogin*.m**, where *yourlogin* is your Purdue Career Account login.
3. Fill out the required information in the header.
   * The assignment number (i.e., PS 01)
   * Your name and official Purdue email address
   * Your section number (since you have not yet been assigned a team number)
   * The names and official Purdue email addresses of anyone you worked with on the code for this problem.
4. Read and follow the Academic Integrity Statement.
5. Follow Programming Standards when completing the file.

**Step 2: Write a small script.** This script will compute the final grade for an ENGR 132 student.

1. In the INITIALIZATION section of your code:
   * Create two variables, total\_points and total\_abs, and set them equal to the values indicated below. Add a descriptive comment to each.
   * Create a variable penalty and set it equal to the half-absence deduction value in the course syllabus (the number should be negative). Add an appropriate comment.

% --- INITIALIZATION ---

total\_points = 754 % total points earned at the end of semester

total\_abs = 4.5 % total absences for the semester

penalty = <*from syllabus>* % *<add your own comment>*

1. In the CALCULATIONS section of your program, type the text shown below to calculate the final grade for the student.

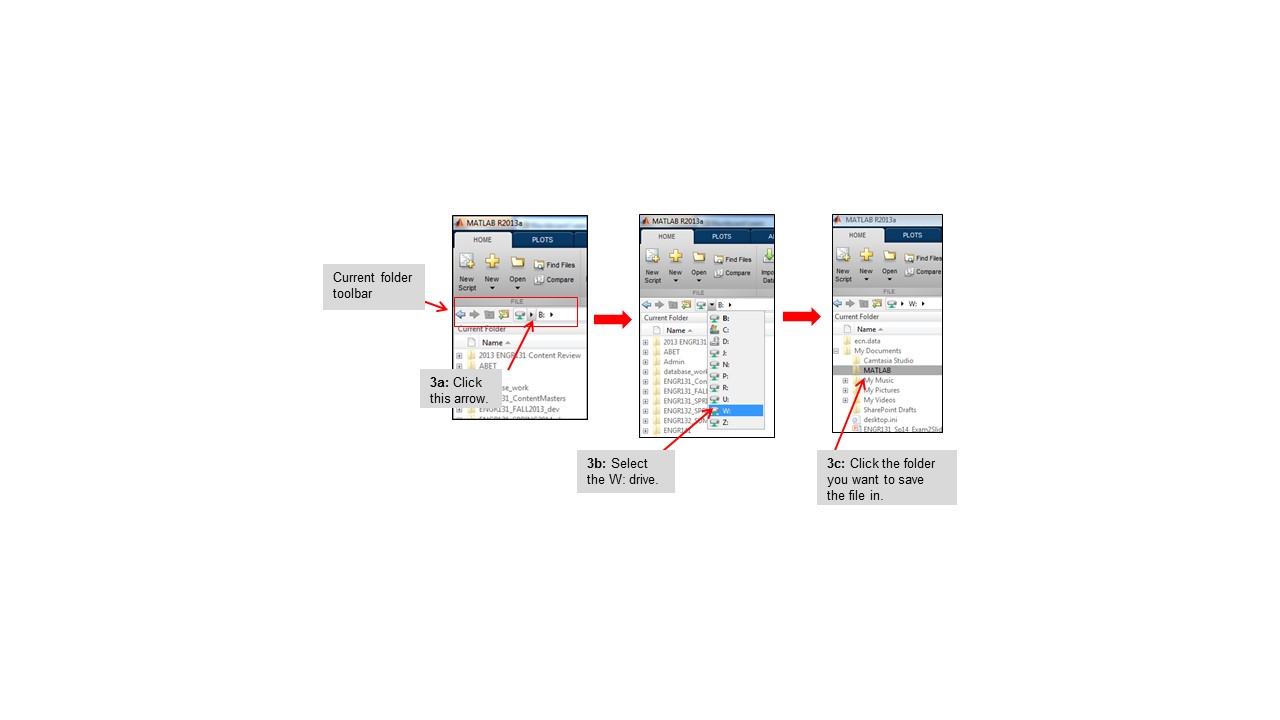
% --- CALCULATIONS ---

total\_deduction = (total\_abs – 3.0)\*2\*penalty % lost points

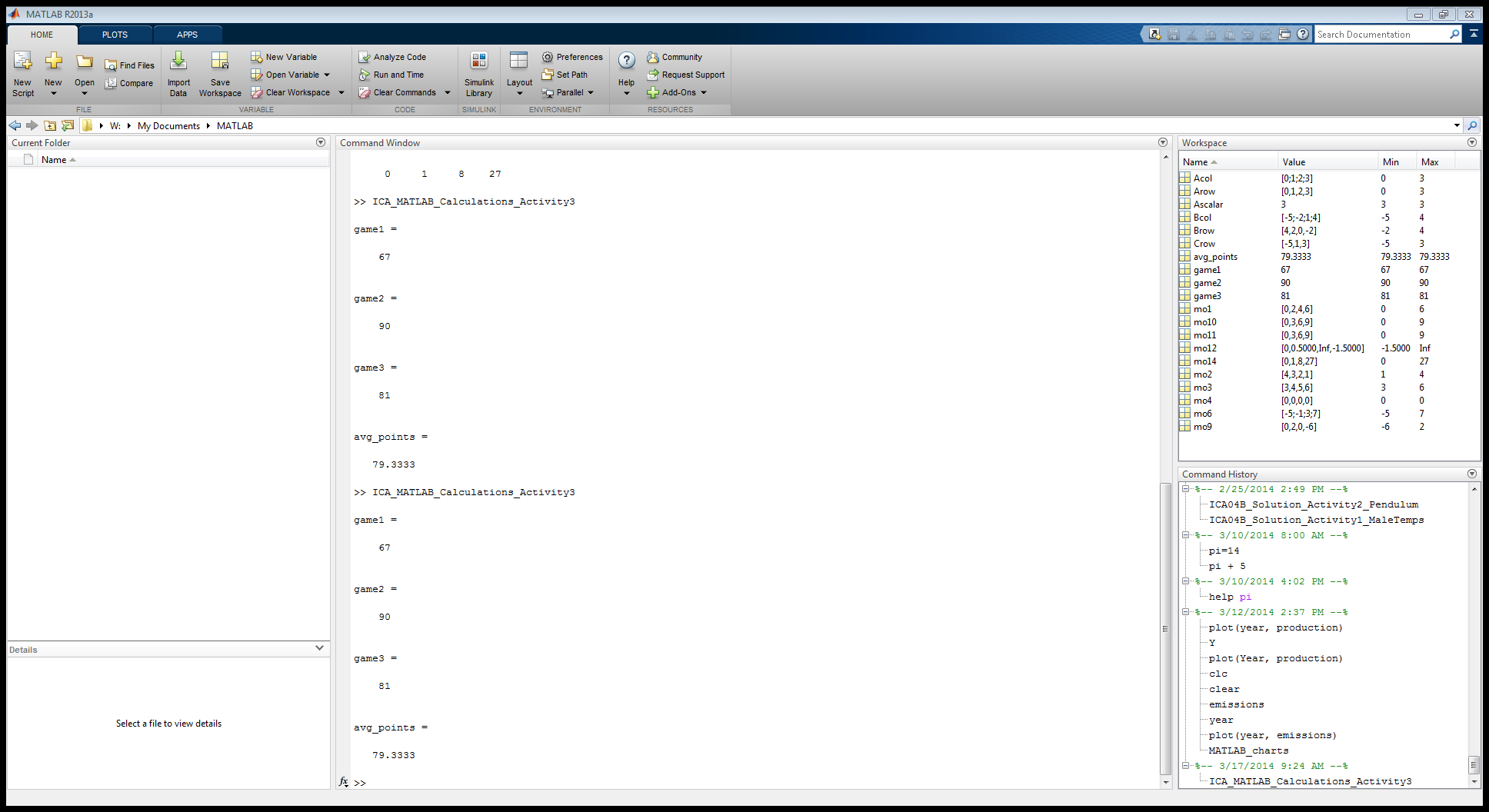
final\_points = total\_points + total\_deduction % final total

**Step 3: Run a script to generate results.**

1. Save the program file to your Purdue Career Account.
2. Return to the MATLAB Command Window.
3. You now need to set the **current folder**, wherein MATLAB will look to find files.
   1. In the current folder toolbar (see image below), click the arrow as shown.
   2. Select the **W:** drive, which is your Purdue Career Account drive.
   3. You will now see the files and folders in your W: drive in the space below the text **Current Folder.** Click the folder you want to save the file in.



The current folder toolbar will now display the path to the current folder.

******

***Note:*** *If you are using MATLAB through Software Remote,* ***you must*** *properly set the current folder or you will not be able to access your files!*

1. In the Command Window, type the name of the script file without the ‘.m’ suffix and press **Enter**.

***For example:*** *If your file name is* **PS01\_Problem3***\_****boiler62.*m***, type* ***PS01\_Problem3****\_****boiler62*** *in the Command Window. MATLAB will execute the commands in the file and display the outputs.*

In a sentence or two **describe** what you see printed in the MATLAB Command Window.

|  |
| --- |
| The variable declarations and results of the calculations have been displayed |

**Step 4: Suppress printing of variables.**

1. Clear the MATLAB Command Window of all text using the **clc** command.
2. **Suppress** the printing of the INTIALIZATION variable assignments by adding semicolons after each command as shown below.

% --- INITIALIZATION ---

total\_points = 754**;** % total points at the end of the semester

total\_abs = 4.5**;** % total absences for the semester

penalty = <*from syllabus>****;*** % *<add your own comment>*

Run your program. In a sentence or two **describe** what you see printed in the MATLAB Command Window.

|  |
| --- |
| Only the results of the calculations have been displayed now |

***Note: Once you know that a line of code is working, it is a programming standard to suppress its printing to the Command Window****.*

**Step 5: Alter existing code.**

1. Compute the grade for another student. The new student has 790.15 points and 5.0 absences. Change the values of the variables in your code to match. Re-save your file.
2. Clear the Command Window and run your program. In a sentence or two **describe** how what you see printed in the MATLAB Command Window is different from Step 4.

|  |
| --- |
| The variable values have been changed to their new calculated values |

**Step 6: Print results to the Command Window.** The **fprintf** command is useful for formatting output to the Command Window.

*MATLAB offers many options for formatting* **fprintf** *statements. Learn more at the webpage* [*http://www.mathworks.com/help/matlab/matlab\_prog/formatting-strings.html*](http://www.mathworks.com/help/matlab/matlab_prog/formatting-strings.html).

1. In the FORMATTED TEXT section of your program, add the command:

% FORMATTED TEXT ---

fprintf(‘The final student score is %f\n’, final\_points)

1. Run your program. In a sentence or two **describe** what you see printed in the MATLAB Command Window.

|  |
| --- |
| There is now a print statement that displays a sentence followed by the value of final\_points |

**Step 7: Manage printed output to the Command Window.**

1. The output from the **fprintf** statement in Step 6, #2 has too many decimal places. Use the information from the MathWorks website to modify your **fprintf** statement to show only one decimal place. Re-save the script and run your program to confirm the results.
2. Write **two** new **fprintf** statements before the current one that state
   1. the student’s total points at the end of the semester (showing no decimal places)
   2. the student’s total absences for the semester (showing one decimal place).
3. Re-save the script and run your program to confirm the results.

|  |  |
| --- | --- |
| **Key Points About Script Files:**   * Script files (m-files) are used to store MATLAB programs. * Use the course template. * Before executing an m-file, change the current folder to the location of the m-file. * Always comment your code so it can easily be understood by others. Comments can appear anywhere in the m-file.   **Useful MATLAB Commands:** | |
| **;** | When placed at the end of a line, suppresses the output in the command window. |
| **%** | When placed before text, “comments out” the text or turns the text into a non-executable comment. |
| **fprintf** | Formats data and displays the results on the screen. |

<continue to Problem 4>

# Problem 4: Vectors

Individual Programming

|  |
| --- |
| **Resource for Problem Set 1**  Reminder: See Blackboard **Course Info** (on the left-hand navigation bar) / **Course Resources** > **MATLAB Resources** for a file called *MATLAB Reference 1: Basic & Complex Calculations.* Specifically, the section titled *Creating Vectors & Matrices* will be useful for this problem. |

**Step 1: Prepare a script file.**

1. Open the MATLAB file **script\_header\_template.m**.
2. Save the file with the name **PS01\_Problem4\_*yourlogin*.m**, where *yourlogin* is your Purdue Career Account login.
3. Fill out the required information in the header as in Problem 3.
4. Read and follow the Academic Integrity Statement.
5. Follow Programming Standards when completing the file.

**Step 2: Create row vectors.**

1. Type the following in the INITIALIZATION section of your program:

rowA1 = [0 1 2 3 4 5]

rowA2 = [0,1,2,3,4,5]

rowA3 = 0:5

rowA4 = 0:1:5

rowA5 = linspace(0, 5, 6)

1. Save and run your program.

3. Compare the results of those five rowA# assignments. Record your observations.

|  |
| --- |
| All of these results are the same; |

1. In the INITIALIZATION section of the program, write the code to create 3 different row vectors each with four elements: **4, 2, 0, and -2**. Call these vectors **rowB1**, **rowB2**, and **rowB3**.
2. In the INITIALIZATION section of the program, write the code to create 3 different row vectors each with the integer elements between −5 and 5 inclusive. Call these vectors **rowC1**, **rowC2**, and **rowC3**.

**Step 3: Create column vectors.**

1. Type the following in the INITIALIZATION section and run your code.

columnA1 = [0;1;2;3;4;5]

columnA2 = transpose(rowA1)

columnA3 = rowA2'

columnA4 = (0:5)'

Compare the results.

|  |
| --- |
| The steps in step 2 created vector quantities that were in rows, however the steps in step 3 created quantities that trend vertically in columns |

1. In the INITIALIZATION section of the program, write the code to create 3 different column vectors each with the integer elements between −5 and 5 inclusive. Call these vectors **columnB1**, **columnB2**, and **columnB3**.

**Step 4: Code and explain vector operations.**

1. Add the operations below to the CALCULATIONS section of your script. Comment each line of code with an explanation of what the operation does (the first operation shows an example). If an error occurs, explain the error in the comment and then suppress the whole line of code with a % character so that the remaining commands can execute.

***Note:*** *When an error occurs in MATLAB, MATLAB will stop executing the code at the line where the error occurred.*

x = rowA1(3) % Assigns the 3rd element of rowA1 to x

y = columnA1(4)

vecA = rowA1 + rowA1

vecB = rowA1 + rowB1

vecC = rowA1 + 3

vecD = rowA1 - rowA1

vecE = rowA1 – rowC1

vecF = columnA1 + columnB1

**Step 5: Perform element-by-element operations.**

|  |  |
| --- | --- |
| **Matrix Math is the Default in MATLAB**  Before you perform multiplication, division, or exponentiation with arrays (e.g., vectors and matrices) in MATLAB, it is important to know the following: | |
| MATLAB performs **matrix operations** by default. | Example of matrix operations:  X = |
| MATLAB can perform **element-by-element operations**. | Example of element by element operations:  X = |

To perform *element-by-element* multiplication, division, and exponentiation of vectors, you must suppress the matrix math. This is done using **the period character** (also called the dot operator), which refers to the use of a period before the symbols shown in the table:

|  |  |  |
| --- | --- | --- |
| **Operation** | **Operator** | **Operator for**  **element-by-element operation** |
| Multiplication | **\*** | **.\*** |
| Division | **/** | **./** |
| Exponentiation | **^** | **.^** |

***Note:*** *There are no .+ or .- operators because those operations are element-by-element operations by definition.*

1. In the INITIALIZATION section of your program, define a scalar as follows:

value1 = 3

1. Add the operations below to the CALCULATIONS section of your script. Each operation is ***intended*** to produce an element-by-element result. Write a comment for each line that identifies:
   1. If an element-by-element operation occurs
   2. If the period character is used but is not necessary
   3. If an operation is using matrix math, not element-by-element operations – even if the code produces an error
   4. If mismatched array dimensions prevent an element-by-element operation from executing

The first two operations show commenting examples.

When a MATLAB error occurs, suppress the whole line of code with a % character so that the remaining commands can execute.

***Tip:*** *For help responding to MATLAB error messages, refer to the document* ***Debugging MATLAB Code****. It is available on Blackboard – click on* ***Course Info (****in the left-hand navigation bar) /* ***Course Resources*** *>* ***MATLAB Resources****.*

result1 = rowA1 \* value1 % Element-by-element multiplication

% by a scalar

result2 = rowA1 .\* value1 % Element-by-element multiplication

% by a scalar; the period character is

% NOT necessary.

result3 = rowA1 \* rowA1

result4 = rowA1 .\* rowA1

result5 = rowA1 .\* rowB1

result6 = rowA1 .\* columnA1

result7 = rowA1 / value1

result8 = rowA1 ./ value1

result9 = rowA1 / rowA1

result10 = rowA1 ./ rowA1

result11 = rowA1 ./ rowB1

result12 = rowA1 ./ columnA1

result13 = rowA1 ^ value1

result14 = rowA1 .^ value1

result15 = rowA1 ^ rowA1

result16 = rowA1 .^ rowA1

result17 = rowA1 .^ rowB1

result18 = rowA1 .^ columnA1

**Step 6: Perform more complex element-by-element operations.**

1. Perform element-by-element calculations in the CALCULATIONS section of your script. Assign **s** and **t** as follows:

s = [0.1 0.6 0.9]

t = [8.2 8.4 9.1]

|  |  |  |
| --- | --- | --- |
| **Mathematical Expression** | **MATLAB Command** | **Results** |
|  | a = s .\* t.^2 -((2\*t)./ (3\*s))^(3/5) | -4.3075  38.5164  71.3869 |
|  | (0.5\*t)./(20\*s.^3)+(exp(-s.\*t)./(s + t)) | 205.0531  0.9729  0.3121 |

1. Complete the table above.

|  |  |
| --- | --- |
| **Key Points About Vectors & Matrices:**   * In MATLAB, a vector is an array with either one row or one column. * The period character suppresses matrix operations and enables element-by-element operations.   **Useful MATLAB Commands:** | |
| rowA1 = [0 1 2 3 4 5]  rowA2 = [0,1,2,3,4,5]  rowA3 = 0:5 (starting point : ending point)  rowA4 = 0:1:5 (starting point : increment : ending point) | Create row vector from 0-5 in increments of 1 |
| linspace(a, b, n)  linspace (0, 5, 6) generates the vector [0 1 2 3 4 5] | Create a vector of n points linearly spaced between a and b. |
| columnB1 = transpose(vectorA)  columnB2 = vectorA'  columnB3 = (0:5)' | Transpose a matrix from row to column or column to row |
| array1 .\* array2  array1 ./ array2  array1 .^ array2 | Perform element-by-element operations between arrays with the same dimensions |

<continue to Problem 5>

# Problem 5: Matrix Manipulations I

Individual Programming

**Step 1: Prepare a script file.**

1. Open the MATLAB file **script\_header\_template.m**.
2. Save the file with the name **PS01\_Problem5\_*yourlogin*.m**, where *yourlogin* is your Purdue Career Account login.
3. Fill out the required information in the header as in Problem 3.
4. Read and follow the Academic Integrity Statement.
5. Follow Programming Standards when completing the file.

**Step 2: Create a matrix and perform operations on the matrix.**

1. In the INITIALIZATION section of the program, type the following command to create the following matrix:

|  |  |
| --- | --- |
| **MATLAB Command:**  A =[2 5 8 5; 10 9 1 4; 6 3 2 10] | **Resulting matrix:** |

1. In the CALCULATIONS section, type the lines of MATLAB code shown below. Run your program, and then add a comment on each line of your program explaining what that line does. If an error is created, explain the error and suppress that line of code.

m = A(2,3)

s = A(5,4)

B = A(1,:)

C = A(2,:)

D = A(:,3)

E = sort(D)

A(2,4) = 40

F = A(1:2)

G = A(2:3)

H = A(1:2,2:3)

J = [C B]

K = [C;B]

L = [D C]

**Step 3: Perform operations with matrices.**

1. In the CALCULATIONS section, write a line of MATLAB code to complete each task listed in the table below. **Do not hardcode the values; do use the existing matrices to create new variables or modify existing variables.** Run your program to check your work. Comment your code.

|  |
| --- |
| **Tasks** |
| Use array indexing to assign the third row of matrix **A** to a variable named **M**. |
| Use array indexing to assign the value that is in Row 2, Column 3 of matrix **A** to a variable named **N**. |
| Use array indexing to replace the value that is in Row 1 and Column 2 of matrix **A** with the value **55**. |

|  |  |
| --- | --- |
| **Useful MATLAB Commands:** | |
| Z = [1 3; 9 2] or  Z = [1, 3; 9, 2] | Creates the matrix Z .  Semicolons separate each row of the matrix. |
| Z(m, n)  % where Z is a matrix | Refers to the element in row m and column n of matrix Z [array indexing uses a row,column address] |
| B = Z(:, n) | Copies the nth column of matrix Z to vector B  (Read the colon as “*all the*”; copy *all the* rows of the nth column of Z and assign to B) |
| B = Z(m, :) | Copies the mth row of matrix Z to vector B  (Read the colon as “*all the*”; copy *all the* columns of the mth row of Z and assign to B) |