Problem Set 06 · User-Defined Functions

Instructions

1. This is a team assignment. Your team can work together on all aspects of problems 2 and 3; you will use paired partners on Problem 1. You will submit one zip file of deliverables for the entire team.
2. Publishing user-defined functions (UDFs) is slightly different from publishing scripts. Read the included document *Publishing MATLAB Code – Functions* to learn how to publish functions. If you have trouble with publishing your functions, then check the document *Publishing MATLAB Code – Troubleshooting Issues* for help.
3. Read each problem carefully before starting your work. You are responsible for following all instructions within each problem. Remember that all code submissions must follow the course programming standards.
4. Below are the expected deliverables for each problem.
   - Name your files to match the format in the table below.
   - Publish your code when requested. Not all functions need to be published.

Item	Type	Deliverable to include in Submission
Problem 1: Road Salt UDF	Paired	☐ See Problem 2 list
Problem 2: Road Salt Executive Function	Team	☐ PS06_salt_cone_*loginW*_*loginX*.m ☐ PS06_salt_windrow_*loginY*_*loginZ*.m ☐ PS06_salt_exec_*sss*_*tt*.m ☐ PS06_salt_exex_ *sss*_*tt*_report.pdf
Problem 3: Cable-Stayed Bridge	Team	☐ PS06_cableUDF_*sss*_*tt*.m ☐ PS06_cables_*sss*_*tt*.m ☐ PS06_cableUDF_*sss*_*tt* _report.pdf ☐ PS06_cables_*sss*_*tt* _report.pdf

5. Save all files to your Purdue career account in a folder specific to PS06.
6. When you are ready to submit your assignment,
   - Compress all the deliverables into one zip file and name it **PS06_*sss*_*tt*.zip**. where *sss* is your section number with the leading zeros and *tt* is your team number. Be sure that you
     i. Only compress files using **.zip** format. No other compression format will be accepted.
     ii. Only include deliverables. Do **not** include the problem document, blank templates, etc.
   - Submit the zip file to the Blackboard drop box for PS06 before the due date.
7. After grades are released for this assignment, access your feedback via the assignment rubric in the My Grades section of Blackboard.

Notes Before You Begin this Assignment

Programming Standards for UDFs

A user-defined function template has more details than a script. Reread the course programming standards to learn how to use the function template and how to properly format your functions' headers.

Helpful MATLAB Commands

Learn about the following built-in MATLAB commands, which might be useful in your solutions:

`ceil`

Problem 1: Road Salt Storage UDF

Paired

Learning Objectives

Below are learning objectives that may be used to assess your work on this problem. Learning objectives from past assignments may also be used to assess your work. Use the links to find the full evidence lists for each topic.

Calculations	01.00 Perform and evaluate algebraic and trigonometric operations
Variables	02.00 Assign and manage variables
Text Display	05.00 Manage text output
User-Defined Functions	11.03 Create a user-defined function that adheres to programming standards
	11.04 Construct an appropriate function definition line
	11.05 Match the variables names used in the function definition line to those used in the function code
	11.06 Execute a user-defined function
	11.08 Convert a script to a user-defined function

Problem Setup

In PS01 Problem 1, you wrote a script that calculated the volume and weight of road salt when stored in piles. You wrote the problem using one set of values for the pile geometries, and then you changed those values after the code was functioning.

User-defined functions (UDFs) are useful when your code may need to perform the same calculation repeatedly. INDOT wants you to rewrite the script from PS01 as two separate user-defined functions. The final UDFs will

- Accept input arguments for the pile geometry
- Return as output arguments the height and weight of one pile
- Display the pile height and weight to the Command Window.

You will start with your script **PS01_salt_*yourlogin*.m** from PS01. Your team will be divided into two pairs. Pair WX will write a UDF that calculates the information for conical piles. Pair YZ will write a UDF that calculates the information for windrow piles.

Note: You will need to complete these functions before the next class.

Problem Steps

Pair WX only - convert your script into a function that calculates conical pile information

This UDF requires input and output arguments. It needs to calculate the height and weight of **one** conical pile of road salt that can have **any width (or diameter)**. The function must

- Have an operational function definition line

- Have a correct header that meets ENGR 132 programming standards (LO 11.03)
- Have one (1) input argument: cone diameter (scalar)
- Return two (2) output arguments: cone height (scalar) and cone weight (scalar)
- Print the cone height and weight to the Command Window using `fprintf` with the same display requirements as in PS01

1. Reread PS01 Problem 1. Both you and your paired partner should have your own version of this code. Examine your solutions, select one file to use for this problem, and make any necessary corrections to the calculations and print commands. You will need this script for Problem 2.
2. Open PS06_salt_UDF_template.m. Complete the full header. Note that it contains new information beyond what was expected in a script header. Save the file as **PS06_salt_cone_*loginW*_*loginX*.m**.
3. Create a function definition line for the UDF.
4. Copy the relevant sections from your script into the UDF. Modify them as necessary to make them work within the UDF. Make sure to follow good programming standards.
5. Test and debug your function.
6. Once you are satisfied with the operation of your function, clear your workspace and the Command Window.
7. Run your function from the Command Window using a cone diameter of 21.5 m. Suppress your function call using a semicolon.
8. Paste as comments the **function call** and the **displayed text** to the `COMMAND WINDOW OUTPUTS` section of your UDF code.

Pair YZ only - convert your script into a function that calculates windrow pile information

This UDF requires input and output arguments. It needs to calculate the height and weight of **one** windrow pile of road salt that can have **any width and length**. The function must

- Have an operational function definition line
- Have a correct header that meets ENGR 132 programming standards (LO 11.03)
- Has one (2) input arguments: windrow width (scalar) and windrow length (scalar)
- Return two (2) output arguments: windrow height (scalar) and windrow weight (scalar)
- Print the windrow height and weight to the Command Window using `fprintf` and the same display requirements as in PS01

1. Reread PS01 Problem 1. Both you and your paired partner should have your own version of this code. Examine your solutions, select one file to use for this problem, and make any necessary corrections to the calculations and print commands. You will need this script for Problem 2.
2. Open PS06_salt_UDF_template.m. Complete the full header. Note that it contains new information beyond what was expected in a script header. Save the file as **PS06_salt_windrow_*loginY*_*loginZ*.m**.
3. Create a function definition line for the UDF.
4. Copy the relevant sections from your script into the UDF. Modify them as necessary to make them work within the UDF. Make sure to follow good programming standards.
5. Test and debug your function.

6. Once you are satisfied with the operation of your function, clear your workspace and the Command Window.
7. Run your function from the Command Window using a windrow width of 18.35 m and a length of 45 m. Suppress your function call using a semicolon.
8. Paste as comments the **function call** and the **displayed text** to the `COMMAND WINDOW OUTPUTS` section of your UDF code.

Problem 2: Road Salt Storage Continued

Team

Learning Objectives

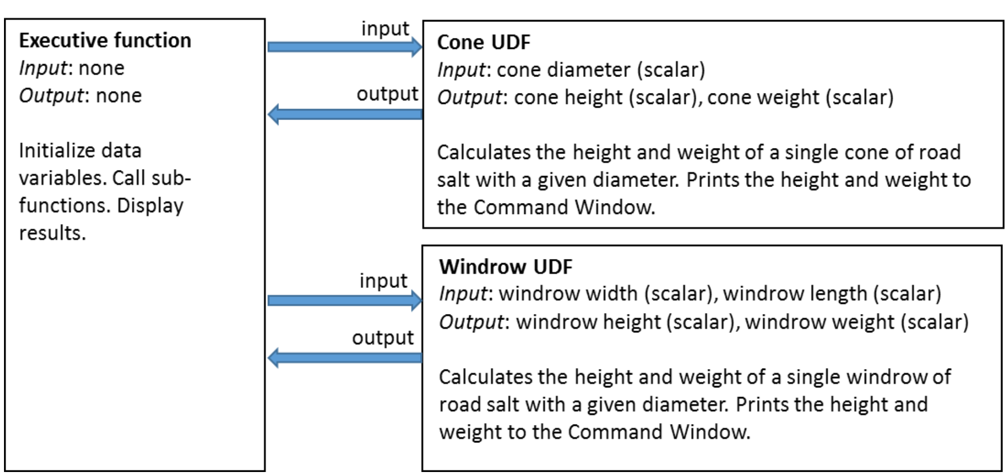
Below are learning objectives that may be used to assess your work on this problem. Learning objectives from past assignments may also be used to assess your work. Use the links to find the full evidence lists for each topic.

Variables	02.00 Assign and manage variables
Arrays	03.00 Manipulate arrays (vectors or matrices)
Text Display	05.00 Manage text output
User-Defined Functions	11.03 Create a user-defined function that adheres to programming standards
	11.04 Construct an appropriate function definition line
	11.05 Match the variables names used in the function definition line to those used in the function code
	11.06 Execute a user-defined function
	11.07 Create test cases to evaluate a user-defined function
	11.09 Track the passing of information to and from a user-defined function
	11.10 Break a problem into a series of sub-functions
	11.11 Coordinate the passing of information between functions

Problem Setup

In Problem 1, you worked in pairs WX and YZ. Each set of partners should have the function they created in Problem 1. **You will need working version of both UDFs for this problem and could be graded on either UDF**.

Your job is to write an executive function that will call both the cone and windrow functions and use their results to determine how many piles INDOT will need to store a given amount salt. The figure below shows how the functions must interact with each other.



INDOT's District 10 has 24,361 metric tons of road salt that need storage. They want to know how many storage facilities they need if they store the salt in conical piles with 21.5-meter diameters or in windrows with 18.35-meter widths and 45-meter lengths. They have requested that your function display all the cone information before the windrow information.

Problem Steps

1. Open both PS06_salt_cone and PS06_salt_windrow functions. Examine the code and the function definition lines. Ensure that both work as expected.
2. Using PS06_salt_exec_template.m, create an executive function and name it **PS06_salt_exec_*sss*_*tt*.m**. Using the proper code sections within the template, the function must
   a. Initialize the total salt weight and the pile parameters for both the cone and the windrow
   b. Call the cone and the windrow UDFs
   c. Calculate the number of conical piles necessary to store all the salt and the number of windrow piles necessary to store all the salt.
      - Round the number up to the nearest integer towards infinity
   d. Print the number of conical piles needed and the number of windrow piles needed.
   e. Remember that INDOT has requested that all information regarding the height, weight, and number of conical piles be displayed before the windrow information.
3. Once your code is debugged and working, run your executive function. In the `COMMAND WINDOW OUTPUT` section of your **executive function**, paste as comments the function call and the displayed results.
4. In the `ANALYSIS` section of your executive function, answer the following questions
   Q1. Follow these instructions in order, and then answer the question below.
      1. Clear your MATLAB workspace.
      2. Run your **PS01 script** form the Command Window.
      3. Clear your workspace.
      4. Run your executive function from the Command Window.

      What differences to you see between the results of running the script and the executive function?

   Q2. Clear the workspace. Run your **PS06_salt_cone** function from the Command Window using an input diameter of 21.5 m. What do you see in the workspace? How is the result different from what the result generated when you ran your executive function?

   Q3. Type **`help PS06_salt_cone_loginW_loginZ`** into the Command Window and hit enter. What do you see? Why is this helpful?
5. In your executive function, update the cone diameter to be 25 meters, the windrow width to be 20 meters, and the windrow length to be 48 meters. Publish only your **executive function** as a PDF file using the file name indicated in the Deliverables list. See the included document for help on how to publish functions.

Problem 3: Cable-Stayed Bridge

Team

Learning Objectives

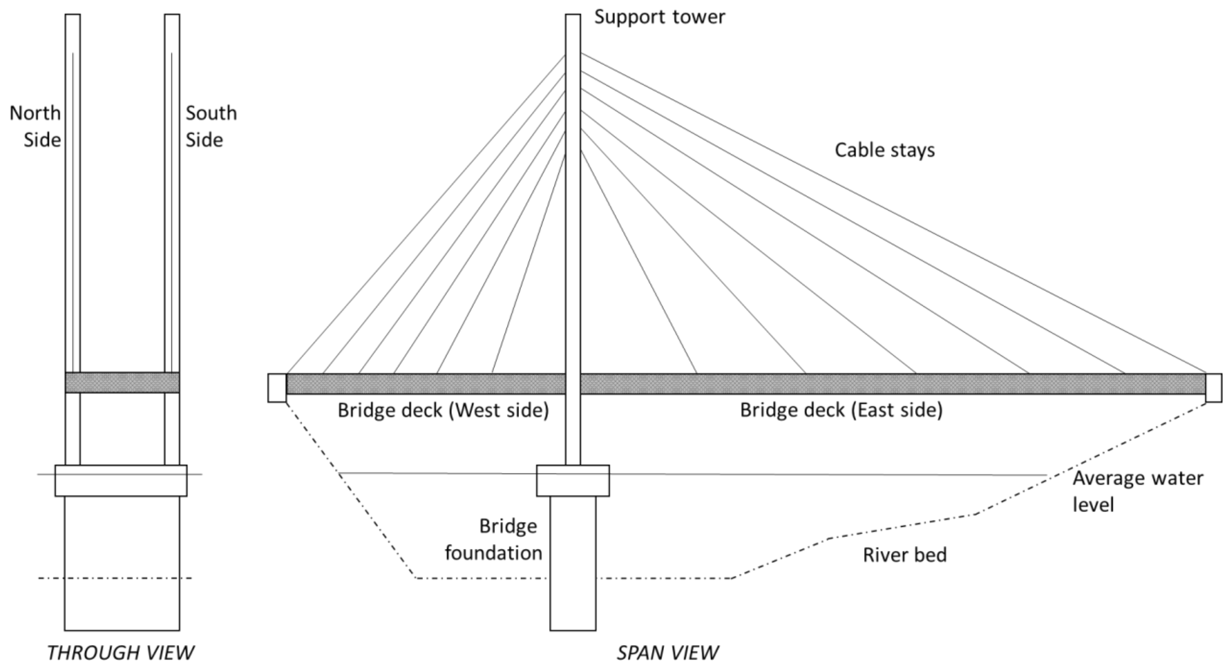
Below are learning objectives that may be used to assess your work on this problem. Learning objectives from past assignments may also be used to assess your work. Use the links to find the full evidence lists for each topic.

Calculations	01.00 Perform and evaluate algebraic and trigonometric operations
Variables	02.00 Assign and manage variables
Arrays	03.00 Manipulate arrays (vectors or matrices)
Text Display	05.00 Manage text output
User-Defined Functions	11.03 Create a user-defined function that adheres to programming standards
	11.04 Construct an appropriate function definition line
	11.05 Match the variables names used in the function definition line to those used in the function code
	11.06 Execute a user-defined function
	11.07 Create test cases to evaluate a user-defined function
	11.09 Track the passing of information to and from a user-defined function
	11.10 Break a problem into a series of sub-functions
	11.11 Coordinate the passing of information between functions

Problem Setup

A cable-stayed bridge is one where cables attach the bridge deck directly to the support towers. The cables transfer the deck loads through the towers to the bridge foundation. Cable-stayed bridges are growing in popularity in the United States due to interesting architectural designs and improved technology that has lowered their cost for mid-length bridges compared to other bridge types.

Your team is part of a civil engineering firm that is working on a pedestrian bridge proposal. The bridge design has a double tower structure that holds the cables, which then comes down to attach to the edges of the bridge deck. The north and south sides of the bridge are identical.



Your group is designing the cable structure and has planned the cable locations for the tower and bridge deck for two different designs. Your task is to estimate the total length of cable necessary for the bridge, its total weight, and its estimated cost for all the required cable. You know the following information about the bridge cables on the northern side of the bridge (note that the southern side is identical):

Table 1. Eastern side of bridge

Cable ID	Height from bridge deck to tower anchorage, in meters	Distance from tower base to deck anchorage, in meters
C1E	50	30
C2E	54	58
C3E	58	84
C4E	62	108
C5E	66	130
C6E	70	150

Table 2. Western side of bridge

Cable ID	Height from bridge deck to tower anchorage, in meters	Distance from tower base to deck anchorage, in meters
C1W	50	18
C2W	54	34
C3W	58	48
C4W	62	60
C5W	66	70
C6W	70	78

The bridge cables will be constructed using Grade 270 steel strands commonly used in bridge cables. Cables 1-5 will be made of 45 strands. Cable 6 will use 36 strands. One strand weighs 1.1 kilogram per meter. The estimated cost for one kilogram of a single strand is $25.

Problem Steps

1. Using **PS06_cableUDF_template.m**, create a user-defined **function** that
    a. Is renamed to match the name format indicated in the Deliverables list
    b. Has an appropriate function definition line
    c. Meets the ENGR 132 function programming standards (LO 11.03)
    d. Accepts three (3) input arguments, which are
        i. A vector of heights from the bridge deck to the cable tower anchorage
        ii. A vector of corresponding distances between the tower base and the cable deck anchorage
        iii. A vector of number of strands in the cable
    e. Returns three (3) output arguments, which are
        i. A scalar value of the total cable length for the inputted cables
        ii. A scalar value of the total cable weight for the inputted cables
        iii. A scalar value of the estimated total cost for the inputted cables
2. Debug your function until it works as expected.
3. Test your function from the Command Window using a vector of heights and distances from Table 1. Do **not** suppress your function call while testing.
4. Paste as comments the **function call** and the **displayed results** to the `COMMAND WINDOW OUPUTS` section of your function.
5. Publish PS06_cableUDF_*sss_tt*.m to a PDF using the information from Table 2 as the input arguments. See the included document for help on how to publish functions with input arguments. Name the published file as required in the deliverables list.
6. Using **PS06_cable_script_template.m**, create a script that will call the function as many times as necessary to get results that can be used to display the following information to the Command Window, using professional formatting:
    a. Total weight of all 12 cables on the **east** side of the bridge
    b. Total weight of all 12 cables on the **west** side of the bridge
    c. Combined length of all 24 cables and the total estimated cost for all 24 cables
7. Suppress your function call when calling it from inside the script. The only information in the Command Window after running the script should be the print statements from Step 6.
8. Publish only your script using the same technique you used in PS02-PS04. Name it as required in the Deliverables List.

References:

https://compass.astm.org/download/A416A416M.23084.pdf

https://transportation.ky.gov/Highway-Design/VE%20Study/VE200804.pdf