Problem Set 06 · User-Defined Functions

Instructions:

1. Each problem in this problem set has a set of deliverables for you to submit. You are responsible for following the appropriate guidelines and instructions below. Create appropriately-named files as instructed.

2. Save all files to your Purdue career account in a folder specific to PS06.

3. Compress all deliverables into one zip folder named **PS06_yourlogin.zip**. Submit the zip file to the Blackboard drop box for PS06 before the due date. *REMEMBER*:
   - Only include deliverables. Do not include the problem document, blank templates, etc.
   - Only compress files into a .zip folder. No other compression format will be accepted.

Deliverables List

Item	Type	Deliverable
Problem 1: Linear Regression UDF	Paired	PS06_ccpp_script_*yourlogin1_yourlogin2*.m PS06_ccpp_nino_*yourlogin1_yourlogin2*.m PS06_regressionUDF_*yourlogin1_yourlogin2*.m PS06_regressionUDF_*yourlogin1_yourlogin2*_report.pdf All data files that are loaded into your m-file(s)
Problem 2: Sit-to-Stand Device	Paired	PS06_sitstand_com_your*loginW_yourloginZ*.m OR PS06_sitstand_springs_*yourloginX_yourloginY*.m
Problem 3: Global Distances	Individual	PS06_nav_distances_*yourlogin*.m PS06_nav_distances_*yourlogin*_report.pdf

Publishing User-Defined Functions

You must publish specified MATLAB user-defined functions to a PDF file. The process is slightly different than the process for publishing scripts. Read the instructions for publishing UDFs in the "Publishing MATLAB Functions" item, included in this Assignment folder, to learn how to publish UDFs.

Note that the document is also available in the Problem Set folder in Blackboard.

Problem Set 06 · User-Defined Functions

Problem 1: Linear Regression UDF

Paired Programming

Learning Objectives

Variables	02.00 Assign and manage variables
Arrays	03.00 Manipulate arrays (vectors or matrices)
Text Display	05.00 Manage text output
Import Data	06.00 Import numeric data stored in .csv and .txt files
Linear Regression	12.00 Perform linear regression
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 Problem Set 4 Problem 3, you learned about combined cycle power plants and how different factors affected their output. You had to perform linear regression using four different parameters, which required you to repeat the same calculations each time.

Repeated calculations are good candidates for user-defined functions (UDFs). For this problem, you will write a UDF to perform the linear regression calculations. The UDF will

- Accept any independent variable vector and dependent variable vector as input arguments
- Return the best-fit line's slope and intercept and the coefficient of determination as output arguments
- Print the regression information to the Command Window.

You will start from your script from PS04 Problem 3, which you will convert into UDFs. Use the data file **Data_CCPP_measurements.csv** to test and run your function.

Problem Steps

A. Ensure your script is functional

Reread PS04 Problem 3. Both you and your paired programmer should have your own versions of this code, since PS04 Problem 3 was individual programming. Examine your solutions, select one file to use for this problem, and make any necessary corrections to the calculations and the print commands. Remove the plot commands.

1. Save the script you will use for this problem as **PS06_ccpp_script_yourlogin1_yourlogin2.m**.

Problem Set 06 · User-Defined Functions

B. Convert your script to a no-input, no-output UDF

Your script must perform linear regression on a data set specific to combined-cycle power plants. Your first UDF will

- Have a correct header that meets ENGR 132 function programming standards (LO 11.03)
- Calculate the slope and intercept of the best fit line for power output in each of the four plant conditions
- Calculate the coefficient of determination for each best fit line
- Print the resulting best-fit lines and coefficients of determination using the same formatting requirements as PS04 Problem 3, Step 2.

2. Open **PS06_regressionUDF_template.m**. Complete the full header information. Note that it contains new information above and beyond what was expected in a script header. Save the file as **PS06_ccpp_nino*_yourlogin1_yourlogin2_*.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.
5. Test and debug your function.
6. Once you are satisfied with the operation of your function, clear your workspace and the Command Window. Then, run your function from the Command Window.
7. Paste as comments the function call and the displayed text to the `COMMAND WINDOW OUTPUTS` section of your UDF code.

C. Convert your no-input, no-output UDF into an input, output UDF

This UDF requires input and output arguments. It must be able to perform linear regression for ***any*** set of data, not just CCPP data. The function must:

- Have an operational function definition line
- Have a correct header that meets ENGR 132 function programming standards (LO 11.03)
- Have two input arguments: one vector of independent-variable data and one vector of dependent-variable data
- Return **only** the best-fit line's slope, intercept, and coefficient of determination as scalar output arguments
- Print the best-fit line's slope, intercept, and coefficient of determination to the Command Window using `fprintf` statements, without any problem-specific information.

  Example print display:

```
Best-fit Line Information:
Slope = -3.5
Intercept = 203.2
Coefficient of determination = 0.852
```

Problem Set 06 · User-Defined Functions

8. Save a new copy of *PS06_ccpp_nino_ yourlogin1_yourlogin2.m* as **PS06_regressionUDF_*yourlogin1_yourlogin2*.m** and revise the header.
9. Revise the function definition line.
10. Revise the UDF code as necessary.
11. Run your function from the Command Window using the following inputs:

```
ind_vec = [8.755, 8.800, 8.813, 8.825, 8.842, 8.861, 8.871];
dep_vec = [13.507, 13.978, 14.081, 14.333, 14.175, 14.296, 14.560];
```

    where `ind_vec` is a vector of independent variable values and `dep_vec` is the corresponding vector of dependent variable values.

12. Paste as comments the function call and the displayed text to the `COMMAND WINDOW OUTPUTS` section of your code.
13. In the `ANALYSIS` section of **PS06_regressionUDF_*yourlogin1_yourlogin2*.m**, answer the following questions:

    Q1: Follow these instructions, in order, and then answer the question below.

    1. Clear your MATLAB Workspace.
    2. Run your **script** file from the Command Window.
    3. Clear your Workspace.
    4. Run **PS06_ccpp_nino_*yourlogin1_yourlogin2*.m** function from the Command Window.

    What differences do you see between the results of the script and the UDF?

    Q2: Clear the workspace. Run your **PS06_regressionUDF_*yourlogin1_yourlogin2*.m** function from the Command Window using `ind_vec` and `dep_vec` as the input vectors.

    What do you see in the Workspace? How is the result different from the result generated by PS06_ccpp_nino_*yourlogin1_yourlogin2*.m?

    Q3: Type **`help PS06_regressionUDF_yourlogin1_yourlogin2`** into the Command Window and hit enter. What do you see? Why is this helpful?

14. Publish PS06_regressionUDF_yourlogin1_yourlogin2.m to a PDF using `ind_vec` and `dep_vec` 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.

Problem Set 06 · User-Defined Functions

Problem 2: Sit-to-Stand Device

Paired programming

Learning Objectives

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
	11.10 Break a problem into a series of sub-functions

Problem Setup

A sit-to-stand assist device can serve the needs of people suffering from muscle weakness due to age or disabilities that make sit-to-stand a difficult functional task. The company you are working for is designing a passive gravity-balancing assist device for sit-to-stand motion. Fig. 2 provides an image of a prototype of such a machine.

Figure 2. Photographs of the new prototype with the subject in sit and stand positions

To make a gravity-balanced assistive device for the human body, the following procedure is used: (i) Determine the center of mass of the system, i.e., device and the human body, using auxiliary parallelograms; (ii) select springs to connect to the system center of mass such that the total potential energy of the system is invariant with configuration. This method allows one to physically determine the system center of mass and connect this point to the inertially-fixed frame through springs.

Determine the center of mass of the system using auxiliary parallelograms.

Problem Set 06 · User-Defined Functions

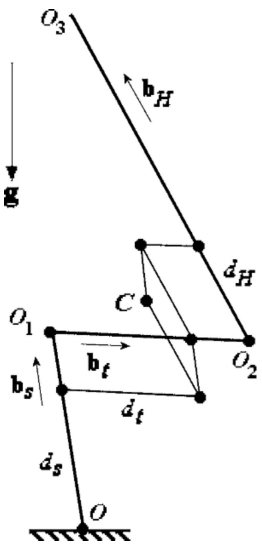

Figure 3. The 3-link human body & device with auxiliary parallelograms to determine the center of mass of the system.

To calculate the center of mass of a person using the sit-to-stand device, model the human body as three sections, each connected at a joint. The shank refers to the mass and length of the leg from the knee to ankle. The thigh refers to the mass and length of the leg from the knee to hip. The HAT (head, arms, trunk) refers to the body's mass and length above the hip. Fig. 3 shows a free-body diagram representation of the human body model. OO_1 represents the shank, O_1O_2 represents the thigh, O_2O_3 represents the HAT. The subscripts for measurements in each section are s, t, and H, respectively.

The location of the center of mass for the model of the human body shown in Fig. 3 from the point O is defined by $\mathbf{r}_{OC}$. Its expression is given by:

$$\mathbf{r}_{OC} = d_s\mathbf{b}_s + d_t\mathbf{b}_t + d_H\mathbf{b}_H \qquad \text{(eq. 1)}$$

where d_s, d_t, and d_H are scaled lengths (m) and $\mathbf{b}_s$, $\mathbf{b}_t$, and $\mathbf{b}_H$ are unit vectors in the direction of each of the body sections. The scaled-length components of r_{OC} can be found using the following equations:

$$d_s = \frac{1}{M}(m_t l_s + m_H l_s + m_s l_{cs}) \qquad \text{(eq. 2)}$$

$$d_t = \frac{1}{M}(m_H l_t + m_t l_{ct}) \qquad \text{(eq. 3)}$$

$$d_H = \frac{1}{M}(m_H l_{cH}) \qquad \text{(eq. 4)}$$

$$\text{and} \quad M = m_s + m_t + m_H \qquad \text{(eq. 5)}$$

where m_i is the mass of the body section in kg, l_i is the length of the body section in meters, l_{ci} is each section's center of mass location in meters, and M is the total mass of the system.

Select Springs to Connect to the System Center of Mass

The human body and the device can be gravity-balanced by attaching four springs to the system as shown in Fig. 4. The total potential energy of the system consists of gravitational and elastic energies due to the springs. The desired stiffness of the springs for gravity balancing of the system are as follows:

$$k = \frac{Mg}{d} \qquad \text{(eq. 6)}$$

$$k_1 = \frac{kd_s}{l_s - d_s} \qquad \text{(eq. 7)}$$

$$k_2 = \frac{kd_s}{l_s - d_s} \qquad \text{(eq. 8)}$$

$$k_3 = \frac{kd_t}{l_t - d_t} \qquad \text{(eq. 9)}$$

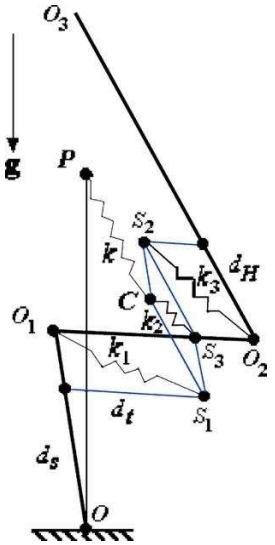

Figure 4. Spring attachments of the 3-link human body.

Problem Set 06 · User-Defined Functions

where g is the gravitational constant in m/s^2 and d is an adjustable length in meters that is chosen such that we have optimal values for the stiffness of the springs and the extension of the spring k. For the purposes of our tests, we will assume d = 0.75m.

Your team is divided into two pairs. Pair WZ will write a user-defined function that calculates the scaled-length. Pair XY will write a user-defined function that spring stiffness values. Table 1 shows a sample test patient's measurements that should be used to design and test your function.

Table 1. Sample body measurements

Section	Length of section (m)	Mass of section (kg)	Center-of-mass location (m)	Adjustable length, d (m)
Shank, *s*	$l_s = 0.421$	3.1	$l_{cs} = 0.55 l_s$	0.75
Thigh, *t*	$l_t = 0.432$	7.39	$l_{ct} = 0.59 l_t$	
HAT, *H*	$l_H = 0.8$	24.13	$l_{cH} = 0.41 l_H$	

Program Steps

A. **Pair WZ** – create a UDF to calculate the center-of-mass scale lengths

1. Using **PS06_sitstand_subUDF_template.m**, create a user-defined function named **PS06_sitstand_com_*yourloginW_yourloginZ*.m** that:
    a. Accepts 3 inputs: mass of body sections (vector), length of body sections (vector), center-of-mass locations of body sections (vector)
    b. Returns 2 output arguments: the scaled lengths (vector) and total mass of the body (scalar)
2. Use the sample patient measurements to thoroughly test and debug your function.
3. Publish your function to a PDF using appropriate values from Table 1 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.

B. **Pair XY** – create a UDF to calculate the spring stiffness constants

1. Using **PS06_sitstand_subUDF_template.m**, create a user-defined function named **PS06_sitstand_springs_*yourloginX_yourloginY*.m** that:
    a. Accepts 3 inputs: total mass of the body (scalar), length of body parts (vector), the scaled lengths (vector)
    b. Converts spring stiffness constants to kN/m
    c. Returns 1 output argument: the spring stiffness constants for the springs (vector)
2. Use the sample patient measurements to thoroughly test and debug your function.
3. Publish your function to a PDF using appropriate input values from Table 1 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.

Reference:
Design of a Passive Gravity-Balanced Assistive Device for Sit-to-Stand Task [https://doi.org/10.1115/1.2216732]
Using Table 1 (p. 1123), equations 4 & 5 (p. 1124) and 16 (p. 1125), Figure 4 & 5 (p.1124), Figure 12 (p. 1127)

Problem Set 06 · User-Defined Functions

Problem 3: Global Navigation

Individual Programming

Learning Objectives

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

Problem Setup

What is the shortest distance between two locations on Earth? If you look at a Mercator-style projection map, you will see that the shortest distance is a straight line drawn between the two locations. But if you look at a globe, which is a more realistic representation of Earth, you will see that the straight line from the map may not match the shortest line drawn on the globe.

A great circle connects two points on a sphere using the shortest distance. The center of the great circle is at the center of the sphere. The Earth's longitude lines are all great circles. The Equator is the only latitude line that is a great circle; the other latitude lines are not. Great circles are widely used in global navigation. If you project the straight line from the map onto a globe, you get what is called a rhumb line. While it appears straight on a map, it follows a curve when placed on a globe.

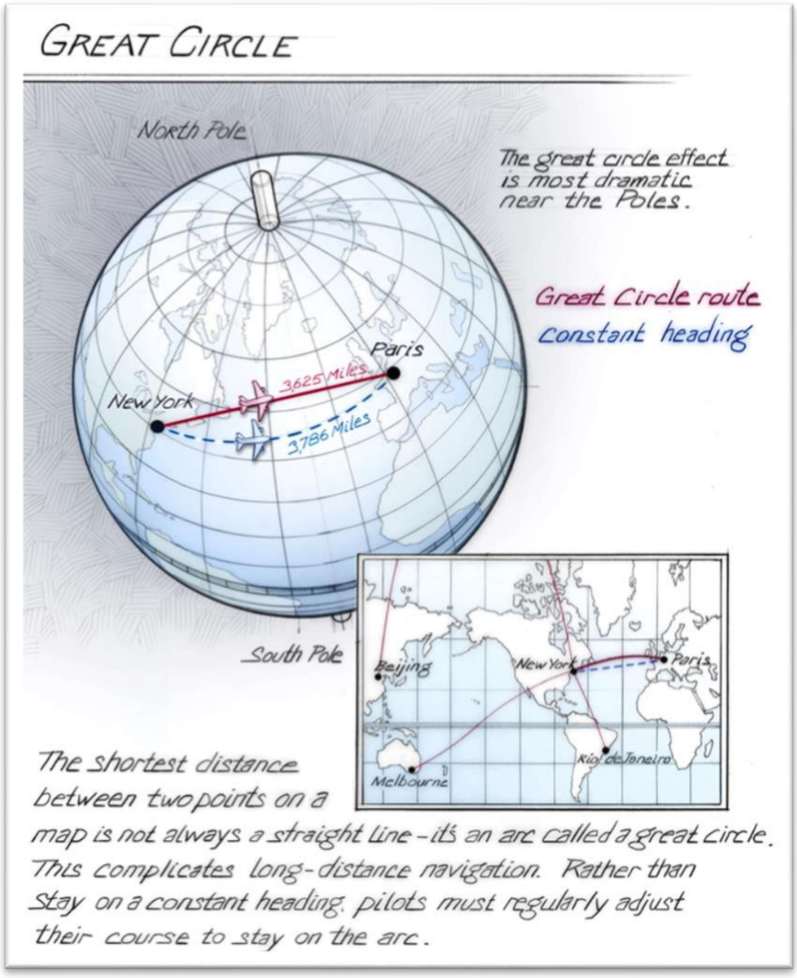

Fig 1: Great Circle Effect
Taken from the Smithsonian; see link below
https://timeandnavigation.si.edu/multimedia-asset/great-circle-route

Problem Set 06 · User-Defined Functions

Using the following definitions, special MATLAB commands, and equations, you can calculate the distances for a great circle path and a rhumb line path.

φ_i	Latitude of point *i* (radians)
$\Delta\varphi$	Change in latitude between two points (radians)
λ_i	Longitude of point *i* (radians)
$\Delta\lambda$	Change in longitude between two points (radians)
R	Mean radius of Earth = 6,371,000 meters
`atan2`	MATLAB's four-quadrant inverse tangent function that requires two inputs – use the two inputs as shown in the equation c below.
d_c	Distance along the great circle path (meters)
d_r	Distance along the rhumb line path (meters)

Use the following equations to determine the distances on either a great circle or on a rhumb line.

The distance on the great circle path:

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 * \cos\varphi_2 * \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 * \text{atan2}(\sqrt{a}, \sqrt{1-a})$$

$$d_c = R * c$$

The distance on the rhumb line path:

$$d_r = R * \sqrt{(\Delta\varphi)^2 + (\cos\left(\frac{\Delta\varphi}{2}\right) * \Delta\lambda)^2}$$

You work for Boiler Aeronautical Consulting Systems (BACS) as a flight systems engineer. Your team is working on an aircraft navigational software package. Your task is to design a user-defined functions (UDF) to calculate the distance between two locations on a great circle path and a rhumb line path. Table 1 shows a list of airports with their corresponding latitudes and longitudes. Use these values to test your function.

Table 1. Airport Information

Airport	Code	Latitude (decimal degrees)	Longitude (decimal degrees)
Chicago – O'Hare International	ORD	41.978603	-87.904842
Frankfurt International	FRA	50.026403	8.543131
Tokyo International	HND	35.5523	139.78
Los Angeles International	LAX	33.942536	-118.408075
Dubai International	DXB	25.2528	55.3644

Problem Set 06 · User-Defined Functions

Problem Steps

1. Using **PS06_nav_distances_template.m**, create a user-defined function that:
    a. Has an appropriate function definition line
    b. Has a correct header that meets ENGR 132 function programming standards (LO 11.03)
    c. Accept six input arguments, which are
        i. the three-letter airport code, as a string, for the starting location
        ii. the latitude and longitude of the starting destination, in decimal degrees
        iii. the three-letter airport code, as a string, for the final destination
        iv. the latitude and longitude of the final destination, in of decimal degrees
    d. Return as output arguments the great circle distance and the rhumb line distance between the two locations, in units of kilometers
    e. Prints the great circle and rhumb line distances, with reference to the starting and ending airports
    f. Is named using the format given in the Deliverables list.
2. Debug the function to ensure that it works as expected.
3. Test your function to find the distances between FRA and HND. Paste as comments the function call and the displayed text to the `COMMAND WINDOW OUTPUTS` section of your code. Repeat for LAX to DXB.
4. Publish the function to a PDF file using inputs for ORD as the starting location and LAX as the final destination. Save the PDF with the file name indicated in the Deliverables list. See the included document for help on how to publish functions.

References
http://www.edwilliams.org/avform.htm
http://www.opennav.com/
https://www.fcc.gov/media/radio/dms-decimal