Problem Set 09 · For Loops

Instructions

1. Use the answer sheet provided in the Assignment Files to complete this problem set. Fill out the header information. Follow any additional instructions that appear in the answer sheet. Submit your finished answer sheet with the rest of your deliverables.
2. You will need your PS07_academic_integrity function for this assignment. You should make any fixes necessary to make it work properly.
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 for each problem. See PS06 for more information.
    - Do not forget to include any data files loaded into your code.

Item	Type	Deliverable to include in Submission
Problem 1: Approximation of ln(3)	Paired	☐ PS09_ln3_approx_*yourlogin1*_*yourlogin2*.m ☐ PS09_ln3_approx _*yourlogin1*_*yourlogin2*_report.pdf ☐ Test Cases (submitted in Answer Sheet) ☐ Tracking Table (submitted in Answer Sheet)
Problem 2: No-Loop Approximation of ln(3)	Individual	☐ PS09_ln3_noloop_*yourlogin*.m ☐ PS09_ln3_noloop_*yourlogin*_report.pdf
Problem 3: Airspeed	Paired	☐ Flowchart (submitted in Answer Sheet)
	Individual	☐ Tracking Table (submitted in Answer Sheet) ☐ PS09_airspeed_*yourlogin*.m ☐ PS09_airspeed_*yourlogin*_report.pdf ☐ USAtmos_1976.p
PS09 Answer Sheet	Individual	☐ PS09_AnswerSheet_*yourlogin*.docx
Academic Integrity Statement	Individual	☐ PS07_academic_integrity_*yourlogin*.m

5. Save all files in a folder specific to PS09.
6. When you are ready to submit your assignment,
    - Compress all the deliverables into one zip file and name it **PS09_yourlogin.zip**. 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 PS09 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.

Problem 1: Approximation of $\ln 3$

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
Relational & Logical Operators	14.00 Perform and evaluate relational and logical operations
User-Defined Functions	11.00 Create and execute a user-defined function
Selection Structures	15.01 Construct a flowchart for a selection structure using standard symbols and pseudocode 15.02 Track a flowchart with a selection structure 16.01 Convert between these selection structure representations: English, a flowchart, and code 16.02 Code a selection structure
Repetition Structures	15.05 Construct a flowchart for a definite looping structure using standard symbols and pseudocode 15.06 Track a flowchart with a definite looping structure 15.09 Create test cases to evaluate a flowchart 15.10 Construct a flowchart using standard symbols and pseudocode 17.05 Convert between these definite looping structure representations: English, a flowchart, and code 17.06 Code a definite looping structure 17.07 Track execution of a definite looping structure using a variable tracking table

Problem Setup

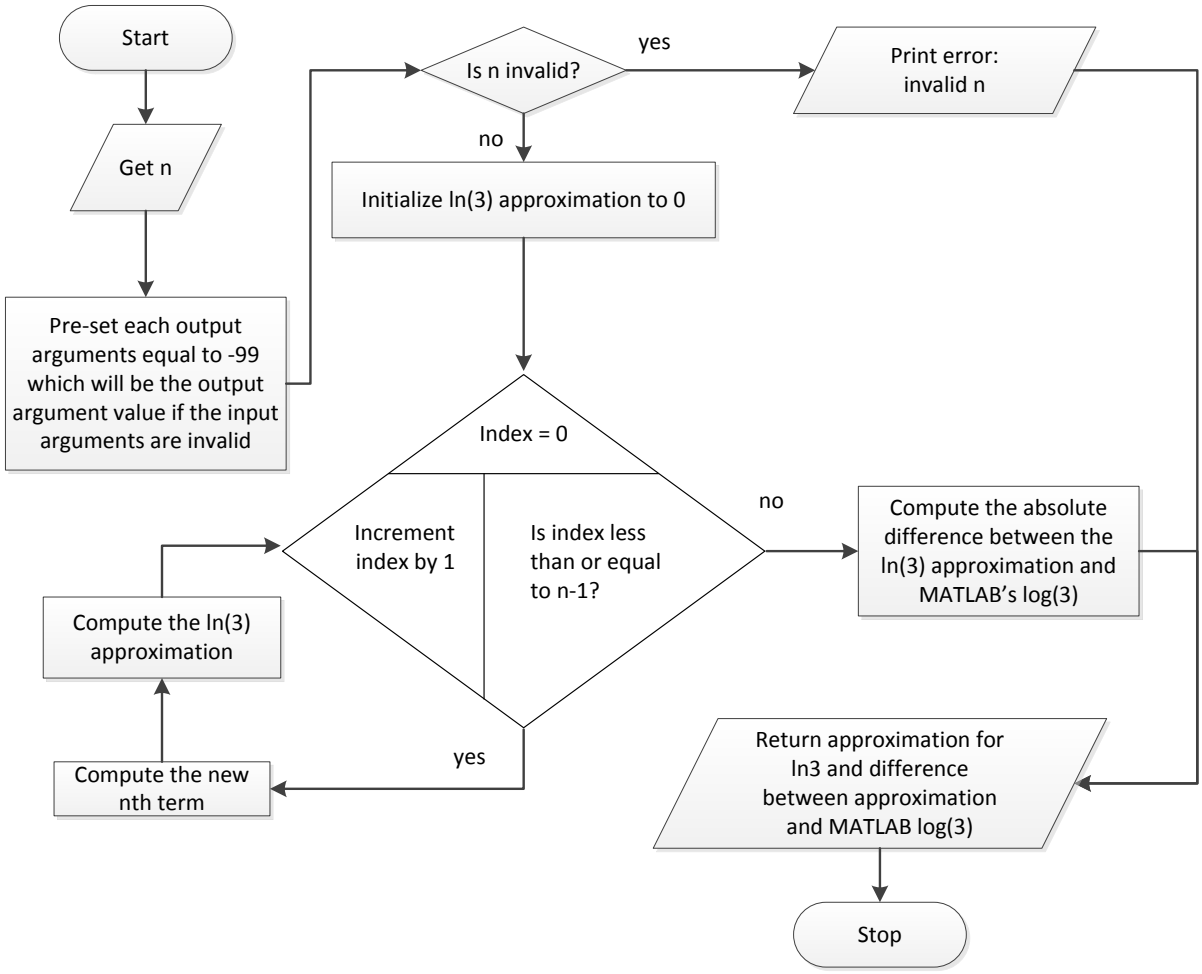
The value of $\ln(3)$ can be approximated using the following expression:

$$\ln 3 = \sum_{k=0}^{\infty} \frac{1}{4^k}\left(\frac{1}{2k+1}\right) \quad for\ k\ =\ 0, 1, 2, \ldots$$

Your task is to create a user-defined function to compute the value of $\ln(3)$ for a given number of terms. Your user-defined function must:

- Accept number of terms (n) as a scalar input argument
- test for invalid inputs
    - n must be a positive integer
    - print appropriate, helpful warning messages for invalid inputs
- return as outputs:
    - the estimate for $\ln(3)$,
    - the absolute difference between the estimate for $\ln(3)$ and the value returned by `log(3)` in MATLAB

An approved flowchart is provided for this problem and shows a method to code the approximation. Translate this specific flowchart to MATLAB code. Pay careful attention to the condition in the flowchart: the value of n is not equal to the value of k in the summation.



Problem Steps

1. **Before you start to code**: Review the flowchart to understand the process for using the expansion to compute $\ln 3$. Note that error messages are printed to the MATLAB Command Window.
2. In your Word answer sheet:
    a. Add a series of test cases to thoroughly test all the possible paths (valid and invalid paths) in the flowchart. One test case has been provided on your answer sheet.
    b. Record the corresponding flowchart outputs for each test case.
    c. Complete the variable tracking table by hand for the execution of the loop for the test case provided.
3. Translate the flowchart above to a MATLAB user-defined function named **PS09_ln3_approx*_yourlogin1_yourlogin2_*.m**. Comment your code appropriately and follow the ENGR132 Programming Standards.

    Hint: To see variable values to more decimal places type `format long` in MATLAB Command Window. https://www.mathworks.com/help/matlab/matlab_env/format-output.html?s_tid=gn_loc_drop
4. Test your function by calling it with the test cases you created in step 2a. Add the following test cases:
    a. n = 5

b. n = 10

c. n = 20

Do not suppress the output when you call your function. Paste the function call and results displayed in the Command Window as comments under the `COMMAND WINDOW OUTPUTS` section of your function file.

5. Call your academic integrity function in the `ACADEMIC INTEGRITY` section.

6. Publish your function as a PDF file using any valid test case and name the file as directed in the deliverables list.

Problem 2: No-Loop Approximation of $\ln 3$

Individual

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
Relational & Logical Operators	14.00 Perform and evaluate relational and logical operations
User-Defined Functions	11.00 Create and execute a user-defined function
Selection Structures	15.01 Construct a flowchart for a selection structure using standard symbols and pseudocode 15.02 Track a flowchart with a selection structure 16.01 Convert between these selection structure representations: English, a flowchart, and code 16.02 Code a selection structure
Repetition Structures	17.08 Eliminate unnecessary definite looping structures

Problem Setup

In Problem 1, you had to follow a flowchart that outlined a method for approximating $\ln 3$ using a summation. Review that problem and your code for it.

Your task for this problem is to perform the same summation without using a loop. Your new code will be a UDF that

- Has the same inputs and outputs as defined in Problem 1
- Continues to check for invalid inputs and print error messages as defined in Problem 1
- Finds the sum of the terms for the approximate value of $\ln 3$ without using a loop.

Problem Steps

1. Using PS09_ln3_noloop_template.m file, create a function has the same functionality as Problem 1 but without a loop. Name this new function according to the naming format in the Deliverables List.
2. Test your function by calling it from the Command Window with the following test cases:
    - n = 8
    - n = 12
    - n = 24
    - n = -0.25

Do not suppress your function call in the Command Window. This will allow the output arguments display to the Command Window. Paste the function call and results displayed in the Command Window as comments under the `COMMAND WINDOW OUTPUTS` section of your function file.

3. Call your academic integrity function in the `ACADEMIC INTEGRITY` section.
4. Publish your function as a PDF file using any valid test case and name the file as directed in the deliverables list.

Problem 3: Airspeed

Individual

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
Relational & Logical Operators	14.00 Perform and evaluate relational and logical operations
User-Defined Functions	11.00 Create and execute a user-defined function
Flowcharts	15.03 Construct a flowchart for an indefinite looping structure using standard symbols and pseudocode
	15.04 Track a flowchart with an indefinite looping structure
	15.09 Create test cases to evaluate a flowchart
	15.10 Construct a flowchart using standard symbols and pseudocode
Selection Structures	16.00 Create and troubleshoot a selection structure
Repetition Structures	17.02 Convert between these indefinite looping structure representations: English, a flowchart, and code
	17.03 Code an indefinite looping structure

Problem Setup

Indicated airspeed (IAS) is a vital aircraft measurement. IAS is measured directly from pressure measurements taken from the aircraft exterior. One method to obtain these measurements is to use a pitot-static system.

A pitot tube is a hollow tube, bent at 90 degrees, that extends off the fuselage or wing of an aircraft. The center opening of the pitot tube points into the direction of the oncoming airflow and measures the total pressure. Another opening that is perpendicular to the airflow, either on the side of the pitot tube or on the aircraft fuselage, measures the static pressure. The difference between these two pressures is called the dynamic pressure. The dynamic pressure is directly related to the aircraft indicated airspeed through a version of Bernoulli's equation for compressible flow:

$$q = P\left[\left(1 + \frac{\gamma - 1}{2\gamma}\frac{\rho}{P}(Ma)^2\right)^{\frac{\gamma}{\gamma-1}} - 1\right]$$

Where

q is the dynamic pressure (kPa)

P is the static pressure at flight altitude (kPa) - in this problem, it is equal to the atmospheric pressure

γ is a ratio between specific heats of air and is 1.4 for this application (dimensionless)

M is the Mach number (dimensionless)

a is the speed of sound at the flight level (m/s), which is calculated using

$$a = \sqrt{\gamma RT}$$

Where

R is the specific gas constant for air, 287.04 (N·m/(kg·K))

T = absolute temperature at flight altitude (K)

To predict the atmospheric pressure and temperature at various altitudes, you will need to use an atmospheric model. The US Standard Atmosphere 1976 is a well-known model and you have access to a MATLAB version of the model, in a file named **USAtmos_1976.p**. It will calculate idealized atmospheric conditions at any altitude from sea level (0 km) and up to, but not including, 86 km using the US Standard Atmosphere 1976 model. The help lines for the code are supplied below.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Program Description
%   This function calculates the temperature and pressure of the
%   Earth's atmosphere at a given altitude, from 0 to below 86 km,
%   using the US Standard Atmosphere 1976 as the model.
%
% Function Call
%      [atm_pressure, atm_temperature] = USAtmos_1976(altitude)
%
% Input Arguments
% 1. altitude = altitude above sea level, 0 <= h < 86 (km) [scalar]
%
% Output Arguments
% 1. atm_pressure = atmospheric pressure (kPa) [scalar]
% 2. atm_temperature = atmospheric temperature (K) [scalar]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

You are an aerospace engineer designing an airspeed indicator that uses a pitot-static system. You plan to test your system, and you want to use Bernoulli's equation to predict the dynamic pressure at various atmospheric conditions before running any experiments. Your plan is to mimic the altitudes and velocity of crusing jet liners. Your first experiement will examine the pitot-static performance from altitudes of 20000 – 45000 ft at a constant Mach number of 0.85.

Your task is to write a user-defined function that will display information about the standard atmosphere and about the dynamic pressure and speed of sound at the experiment altitudes. Your function will accept no inputs and will return two outputs, one vector of of dynamic pressures and one vector of speeds of sound.

Your first step is to better understand the US Standard Atmosphere by visualizing the USAtmos_1976.p outputs over its full range of altitudes. You need to create one figure with two subplots where one subplot shows the atmospheric temperature every 0.5 kilometers from 0 to 85.5 km and the other shows the atmospheric pressure over the same altitude range. Note that it is convention to plot altitude on the y-axis when displaying atmospheric model information.

After you complete that, you will then use the standard atmosphere model and Bernoulli's equation to predict the dynamic pressure and speed of sound using an altitude vector of 20,000 ft, 28,000 ft, 32,000 ft, 36,000 ft, and 45,000 ft. You will store the dynamic pressure and speed of sound as vectors, which will be returned by the function as output arguments. You will also create two figures, one plotting dynamic pressure for each altitude and one plotting the speed of sound.

Problem Steps

1. **Before you start to code**: Create a flowchart to outline how information should move through the code.
    a. Open the answer sheet and examine the high-level flowchart.
    b. Create the two flowchart sections, one for the atmospheric pressure loop and one for the dynamic pressure and speed of sound loop.
    c. You can draw the flowchart using any means that result in a clear image for the answer sheet. Make sure your flowchart is legible.
    d. Add your flowchart sections to the answer sheet.
2. In your Answer Sheet, complete the tracking table for the dynamic pressure and speed of sound loop.
3. Use the high-level flowchart along with your flowchart sections to translate everything into a user-defined function. Use programming standards to place code in the appropriate sections within the template.
4. Test your function using the vector of altitudes provided for the tracking table. Run the function without suppressing the function call so that you can see the dynamic pressure and speed of sound results from the code.
5. Paste the function call and results displayed in the Command Window as comments under the `COMMAND WINDOW OUTPUTS` section of your function file.
6. Call your academic integrity function in the `ACADEMIC INTEGRITY` section.
7. Change the experimental altitude vector to be every 1000 ft from 20,000 to 45,000 ft.
8. Publish your function to a PDF and name the published file as required in the deliverables list. Suppress your function call when you publish the function.

References

https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19800015804.pdf

https://www.grc.nasa.gov/www/k-12/airplane/pitot.html