Problem Set 08 – While Loops

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 PS08.

3. Compress all deliverables into one zip file named **PS08_yourlogin.zip**. Submit the zip file to the Blackboard drop box for PS08 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: Hall-Petch Yield Stress	Individual	Flowchart (submitted in Answer Sheet) Test Cases (submitted in Answer Sheet) PS08_HallPetch_*yourlogin*.m PS08_HallPetch_*yourlogin*_report.pdf Any data file loaded into your m-file PS06_regressionUDF_*yourlogin*.m
Problem 2: Taylor Series for cos(x)	Paired	PS08_taylor_cos_*yourlogin1*_*yourlogin2*.m PS08_taylor_cos_*yourlogin1*_*yourlogin2*_report.pdf Test Cases (submitted in Answer Sheet) Tracking Table (submitted in Answer Sheet)
Problem 3: Weather Balloon Diameter	Individual	PS08_balloon_burst_*yourlogin*.m PS08_balloon_burst_*yourlogin*_report.pdf USAtmos_1976.p Flowchart (submitted in Answer Sheet) Tracking Table (submitted in Answer Sheet)
Answer Sheet	Individual	PS08_AnswerSheet_*yourlogin*.m

Answer Sheet

You must place your flowcharts and test cases in the Answer Sheet provided in the Assignment Files. The answer sheet is named PS08_AnswerSheet_template.docx. You must resave it as **PS08_AnswerSheet_*yourlogin*.docx** before submitting it. Follow the same instructions for the answer sheet as in PS07.

Problem Set 08 – While Loops

Problem 1: Hall-Petch Yield Stress Database

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
Relational & Logical Operators	14.00 Perform and evaluate relational and logical operations
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
	11.11 Coordinate the passing of information between functions
Flowcharts	15.01 Construct a flowchart for a selection structure using standard symbols and pseudocode
	15.02 Track a flowchart with a selection structure
	15.09 Create test cases to evaluate a flowchart
	15.10 Construct a flowchart using standard symbols and pseudocode
Selection Structures	16.01 Convert between these selection structure representations: English, a flowchart, and code
	16.02 Code a selection structure

Problem Setup

Mechanical behavior is a generic term for the response of a material to applied force. This response is measured by shape changes that the material undergoes as force is applied. The simplest means for quantitatively measuring mechanical response is a tensile test. Force is applied to a test specimen of cylindrical geometry. As the applied force increases, the specimen first undergoes elastic deformation and then transition to both elastic and plastic deformation. The strength of metals are characterized by the yield stress at which the onset of plastic deformation begins (shape changes are permanent). A schematic of a tensile test and the corresponding stress-strain curve are shown in Figure 1.

Problem Set 08 – While Loops

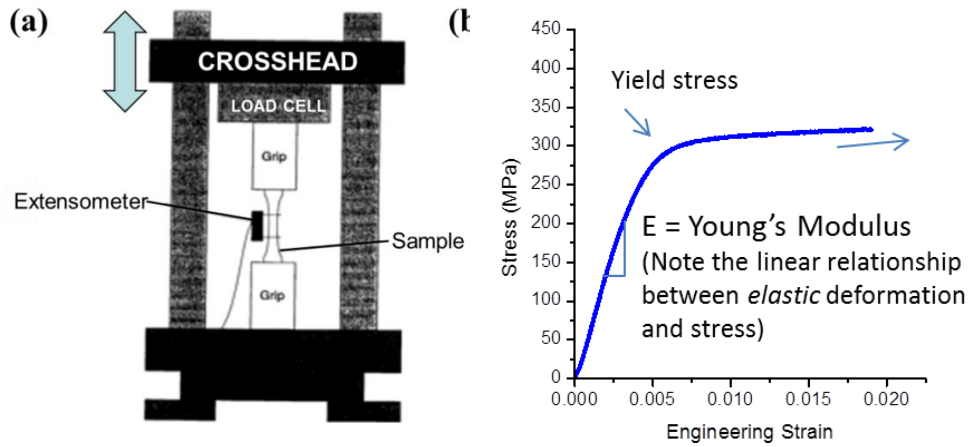


Figure 1: (a) Schematic of a mechanical testing load frame and (b) the initial portion of a stress-strain curve for a cold-worked copper alloy showing the elastic region and the onset of yielding.

The yield stress is also related to the microstructure of the material. Figure 2 shows the microstructure from a copper alloy. The copper alloy is composed of many crystal of different orientation called "grains." The yield stress of the materials is related to the size of these grains, which can be varied through processing (solidification or deformation processing and by various heat treating operations).

An empirical relationship describing the variation in strength with grains size was proposed independently by Hall and Petch in the early 1950's. The Hall-Petch relationship can be stated as:

$$yield\_stress = \frac{k}{\sqrt{grain\_size}} + \sigma_0$$

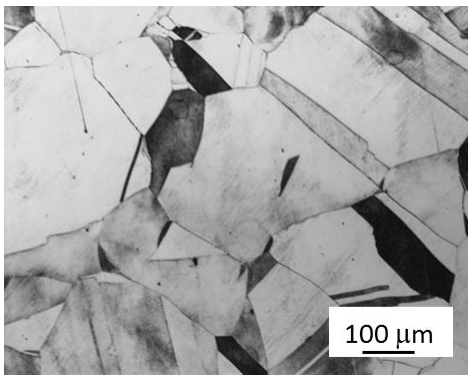


Figure 2. Microstructure of a copper alloy showing crystalline grains and annealing twins

where k = Hall-Petch slope and σ_0 corresponds the yield stress of a material with a very large grain size.

Note that plotting the yield_stress vs. grain_size data as yield_stress vs. $\frac{1}{\sqrt{grain\_size}}$ will result in a linear plot where k is the line's slope and σ_0 is the y-axis intercept.

As a materials engineer for a metallurgy company, your job is to create program that will predict the yield stress of a sample metal at a given grain size for metals in the company's database. You have been provided with the grain size and actual yield stresses for seven metals, in a file named **Data_materials_grainsz_yieldstress.csv**. You will create a user-defined function that

- Accepts the sample metal's atomic symbol (e.g., iron is Fe) as a string and the grain size in microns as input arguments
- Determines which metal in the database to use for calculations
    - If the metal symbol is not found in the database, then the function should warn the user and continue under the assumption that the metal is iron.
- Uses your PS06_regressionUDF sub-function to perform linear regression using Hall-Petch variables for the specified metal
- Calculates the predicted yield strength for the metal at the specified grain size
- Determines whether or not the specified grain size is within the range of the data

Problem Set 08 – While Loops

    - If yes, then display the predicted yield stress to the Command Window, with a reference to the metal being modelled
    - If no, then display an appropriate warning along with the predicted yield stress, with a reference to the metal being modelled
- Displays the Hall-Petch linear equation and coefficient of determination to the Command Window, with a reference to the metal being modelled.

Problem Steps

1. Open your PS06_regressionUDF file. Remove all the text displays, analysis answers, and Command Window outputs. This function should only calculate the regression information (all display commands will be in the other UDF). Save it using the name format given in this problem's Deliverables List.
2. **Before you start to code**: Create a flowchart to outline how information should move through the code.
    - Draw a flowchart to:
        - Determine which metal to use in the database
        - Calculate the Hall-Petch yield stress
        - Determine if the grain size input is within the predictive range of the model
        - Display the required information to the Command Window
    - You can draw the flowchart using any means that result in a clear image for the answer sheet. Make sure your flowchart is legible.
3. In your Answer Sheet,
    - Paste a clear image of your flowchart
    - Select a series of test cases to thoroughly test all possible paths on your flowchart
    - Record the metal and whether or not the grain size is within the predictive range of the model (do not calculate the yield stress)
4. Translate your flowchart into a two-input, no-output user-defined function. Open **PS07_HallPetch_template.m** and complete the header. Save it using the name format given in this problem's Deliverables List.
5. Test your function using the test cases that you specified in Step 3.
6. For each test case, paste the function call and results displayed in the Command Window as comments under the `COMMAND WINDOW OUTPUTS` section of your function file.
7. Publish your function to a PDF using any valid set of inputs and name the published file as required in the deliverables list.

References:

Z. C. Cordero, B. E. Knight & C. A. Schuh, "Six decades of the Hall–Petch effect – a survey of grain-size strengthening studies on pure metals," International materials Review 61 (2016) 495.

School of Materials Engineering, Purdue University, Dr. David Johnson

Problem Set 08 – While Loops

Problem 2: Taylor Series for $\cos x$

Paired Programming

Learning Objectives

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	11.03 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
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
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

Many complicated functions used in science and engineering applications are made easier to understand when represented as Taylor series. Taylor series are also used in science and engineering applications to make approximations. For instance, the cosine function can be approximated with the Taylor series as:

$$\cos x = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k}}{(2k)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$$

In this problem, your task is to create a user-defined function that calculates the approximate value of the cosine of a given number by summing an unknown number of terms in the series. The number of terms will be determined by establishing a "tolerance", which is the maximum allowable value of the final computed term in the series.

For example, the table below shows the number of terms, values of each Taylor series term, and the approximate value of cosine of 2 when the tolerance is 0.01.

Problem Set 08 – While Loops

Number of Terms , n	Value of (k)	Value of nth Term	Taylor Series Value of $\cos(2)$
1	0	1	1
2	1	-2	-1
3	2	0.6667	-0.3333
4	3	-0.0889	-0.4222
5	4	0.0063 (|0.0063| ≤ 0.01, so final computed term)	-0.4159
	MATLAB's built-in natural log function: `cos(2) = -0.4161`		

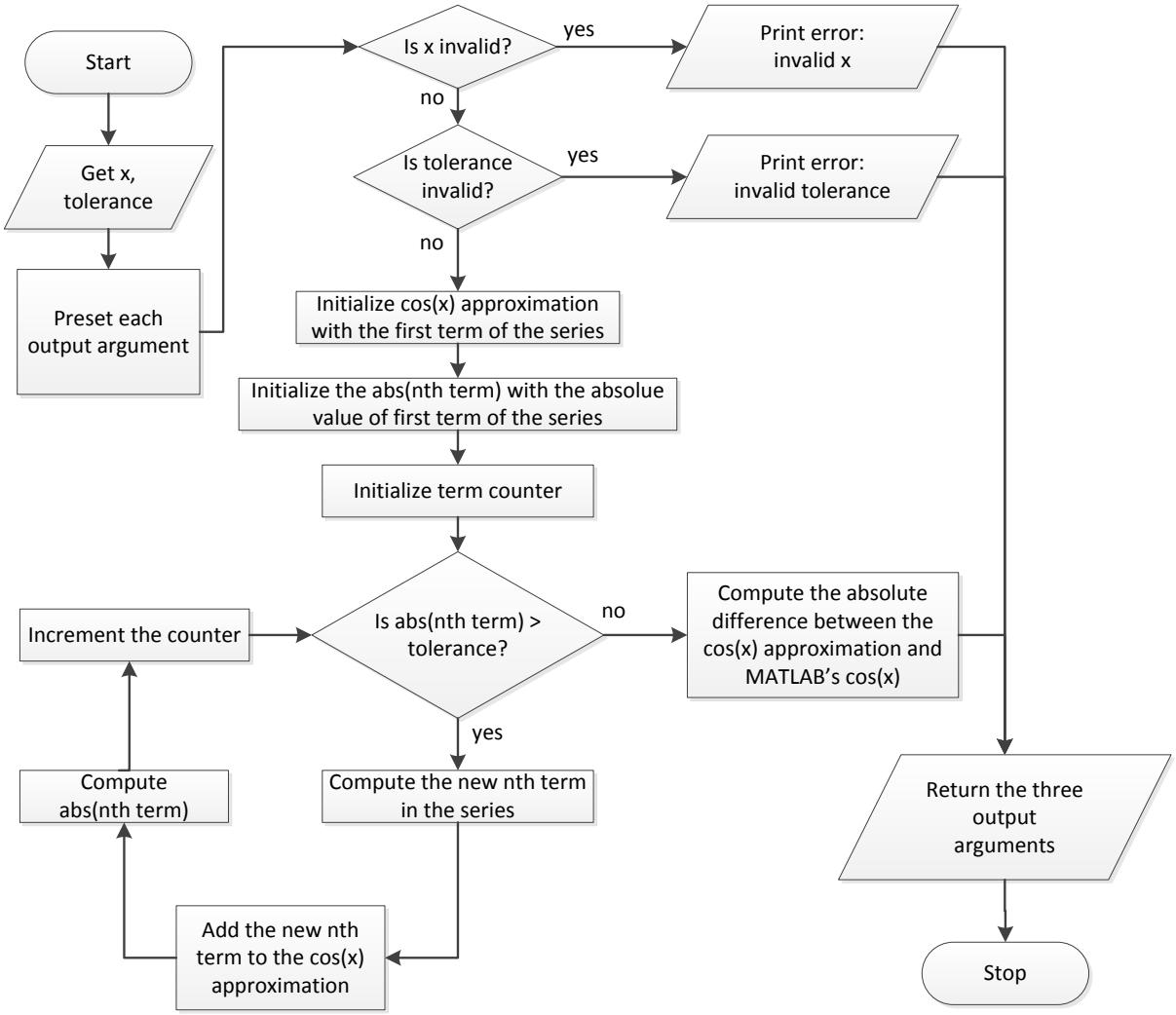
After determining the approximate value of the cosine function, your UDF needs to calculate the absolute difference between that value and MATLAB's value for the cosine. In the table above, the absolute difference is |(-0.4159) – (-0.4161)|.

Your user-defined function must:

- accept as input arguments x and the tolerance value for the last term of the Taylor series
- test for invalid inputs
  - x must be a scalar;
  - the maximum value of the last term must be between 0 and 1, not inclusive
  - print appropriate, helpful error messages for invalid inputs
- return as outputs:
  - the number of terms used in the Taylor series computation,
  - the computed value of $\cos x$ using the Taylor series, and
  - the absolute difference between the Taylor series computed value of $\cos x$ and the value returned by MATLAB's built-in cosine function
  - if an input argument is invalid, then return each output argument above as -99

A flowchart is provided for this problem and shows a method to code the Taylor series of the cosine function. Translate this specific flowchart to MATLAB code.

Problem Set 08 – While Loops



Problem Steps

1. **Before you start to code**: Review the flowchart to understand the process for using the Taylor series to compute $\cos x$. 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 **PS08_taylor_cos_*yourlogin1_yourlogin2*.m**. Comment your code appropriately and follow the ENGR132 Programming Standards.

    Hint: learn about MATLAB's `factorial` command

Problem Set 08 – While Loops

4. Test your function by calling it from the Command Window with the test cases you created in step 2a. 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. Publish your function as a PDF file using any valid test case and name the file as directed in the deliverables list.

Problem 3: Weather Balloon Diameter

Individual Programming

Learning Objectives

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	11.03 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
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
Repetition Structures	17.02 Convert between these indefinite looping structure representations: English, a flowchart, and code
	17.03 Code an indefinite looping structure

Problem Set 08 – While Loops

Problem Setup

Weather balloons are used every day to measure meteorological data. These balloons are filled with helium and carry a small payload of atmospheric measuring devices.

The volume and diameter of a balloon will increase as the balloon's altitude increases. This is due to the decreased atmospheric pressure. Eventually, the balloon will reach its maximum diameter and will burst. Balloons tend to burst at altitudes of 25 – 40 km.

The balloon's diameter increases according to the ideal gas law. If you assume that the balloon is a perfect sphere, you can use its diameter to calculate its volume and vice versa. If you know the atmospheric conditions along the balloon's flight, then you can predict the diameter of the balloon as it rises through the atmosphere using the following equation:

$$d_h = d_0 \cdot \sqrt[3]{\frac{P_0 T_h}{P_h T_0}}$$

Where:

d_0	diameter of the balloon at launch
P_0	atmospheric pressure at launch
T_0	atmospheric temperature at launch
d_h	diameter of the balloon at a given altitude
P_h	atmospheric pressure at a given altitude
T_h	atmospheric temperature at a given altitude

You are an engineer working with atmospheric scientists who perform weather balloon experiments. Your job is to find balloons that will meet the needs of the science team, which will vary depending on the experiment. Your task is to write a user-defined function that will calculate an expected burst altitude for a weather balloon with an expected burst diameter and a known initial launch diameter.

Your team already has a p-code, named **`USAtmos_1976.p`**, that 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)
```

Problem Set 08 – While Loops

```
%
% 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]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The science team has identified three potential balloons to use in the experiments, which are shown in Table 1 below.

Table 1. Balloon Specifications

Balloon Weight	Maximum Fill Diameter prior to launch	Burst Diameter
600 g	1.5 m	6.0 m
1500 g	1.9 m	9.5 m
3000 g	2.1 m	13.1 m

Your user-defined function must meet the following requirements:

- Accept as input arguments the fill diameter of the balloon at launch and its expected burst diameter
- Assume any balloon launches from an altitude of 0 km
- Calculate the burst altitude of the balloon to the nearest 0.5 kilometer using a looping structure. Use USAtmos_1976.p to determine the necessary atmospheric pressure and temperature
- Print the altitude, atmospheric pressure, atmospheric temperature, and balloon altitude for **each** iteration
- Be a no-output function

Problem Steps

1. **Before you start to code**: Create a flowchart to outline how information should move through the code.
    a. You can draw the flowchart using any means that result in a clear image for the answer sheet. Make sure your flowchart is legible. Options include:
        i. Drawing it by hand and taking a clear photo
        ii. Drawing it directly in the Word answer sheet using Word's drawing tools
        iii. Drawing it in Microsoft's Powerpoint, Publisher, or Visio
        iv. Using another flowchart tool, such as Lucidchart
2. In your Word answer sheet:
    a. Complete the variable tracking table by hand for the first 4 iterations of the loop for the test case provided.

Problem Set 08 – While Loops

3. Translate your flowchart to a MATLAB user-defined function. Comment your code appropriately and follow the ENGR132 Programming Standards.
4. Test your function by calling it with the balloon specifications listed in Table 1. After testing and debugging,
    a. Indicate, on the answer sheet, how many iterations are needed to determine the expected burst altitude of the balloon in the test case.
5. For the test case, paste the function call and results displayed in the Command Window as comments under the `COMMAND WINDOW OUTPUTS` section of your function file.
6. Publish your function as a PDF file using the **1500g balloon specifications as the inputs** and name the file as directed in the deliverables list.

References

[image]http://radiosondemuseum.org/what-is-a-radiosonde/

http://kaymontballoons.com/Weather_Forecasting.html