Assignment overview

This assignment tests the following skills:

- Implement classes in Matlab
- Writing robust code, including creation of unit tests
- Create a Matlab app for a specific purpose

This assignment is worth 40% of the ES2D7 module and it contains five tasks - see summary in the table below. All tasks are to be completed and submitted to tabula by 12:00 (noon) 30th August.

Task	Type of tasks	% Assignment credit	Files to be uploaded
1	Gasket class	20%	Gasket.m class
2	MSD_sim class	20%	MSD_sim.m class
3	Test script for MSD_sim class	20%	testMSD_sim.m
4	Data_Analysis class	20%	DataAnalysis.m
5	Calculator App	20%	App + necessary files

Important notes

1) The work that you submit must be your own work. Do not copy from online sources or share your solution with other students. Plagiarism detection software will be used to identify solutions with a high similarity.

2) It is very important that you use the specified filenames, class names, variable names and method names. These are case sensitive. Some example mistakes would be:

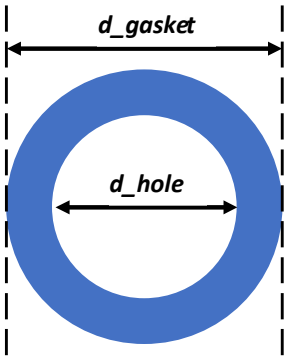
- Using ***calculatearea*** or ***Calculatearea*** or ***Calculate_Area*** as a method name instead of ***CalculateArea***
- Implementing a method called ***GetA*** or ***geta*** instead of ***getA***

3) Please carefully check that you have submitted the correct files to tabula and do NOT change the filenames from those given in the template (you will score zero for that task). 4) Several example scripts are given in this briefing sheet to show how the classes might be used. It is highly recommended to test your solutions with these test scripts as it will highlight whether the methods have been properly set-up.

`

Assignment Task 1 - Gasket Class

Using the template on moodle, write matlab code for a class called ***Gasket***, which calculates the area of a ring gasket (the blue area in the figure below). A ring gasket has an inner diameter *d_hole* and an outer diameter *d_gasket* (both in centimetres). A template class file has been provided for you with the following properties (with public access): ***d_gasket*** and ***d_hole***.



The class ***Gasket*** shall have the following methods:

- ***SetSize(d_gasket, d_hole)*** which has two input arguments: ***d_gasket*** and ***d_hole***. Upon calling this method, the two dimensions shall be stored (in cm). If **d_gasket** OR ***d_hole*** are negative, the following error message shall be given: 'ERR_NEG'.
- ***CalculateArea()*** which calculates and returns the area of the gasket in cm^2. If this method is called when **d_gasket** OR ***d_hole*** is empty, then the follow error message shall be given: 'ERR_EMPTY'.
- ***CalculateRatio()*** which calculates and returns the ratio of ***d_hole*** to ***d_gasket***. If this method is called when **d_gasket** OR ***d_hole*** is empty, then the following error message shall be given: 'ERR_EMPTY'.

The example below shows how this class might be used:

```
g = Gasket;
g.SetSize(2,1);
a = g.CalculateArea() % returns the ratio in variable a
r = g.CalculateRatio() % returns the ratio in variable r
```

In order to help you with this task, a test script has been provided with a few tests implemented (testGasket.m). Note – you will need more tests to properly test this class.

Assignment Task 2 - MSD_sim Class

Using the template on moodle, create a class ***MSD_sim*** for simulating a mass-spring-damper system, following the requirements below. You might find it useful to consult your notes/resources from the ES197 module and Chapter 9 of the Engineering Databook.

Implement requirements 4, 5, 6, 7 using the formulas given in Sections 9.3 and 9.4 of the databook. Do not use the TF or step functions for this task.

1. The class shall have a method ***SetMass*(*mass*)** which sets the value of the mass. The method has one input argument ***mass*** which is in Kg.
2. The class shall have a method for setting the damper value in N m s^{-1}: ***SetDamper*(*damper_coefficient*).**
3. The class shall have a method for setting the spring constant ***SetSpring(spring_constant).***
4. The class shall have a method ***CalcNaturalFreq()*** which calculates and returns the undamped natural frequency in rad s^{-1}.
5. The class shall have a method ***CalcDampingFactor()*** which calculates and returns the damping factor (has no units).
6. The class shall have a method ***isUnderDamped()*** which calculates whether the system is underdamped or not. If the system is underdamped then the method shall return 1, otherwise the method shall return a zero.
7. The class shall have a method ***CalcUnitStepResponse(t)*** which returns the output of the system at time ***t***, given unit step input at *t*=0. Use section 9.4 of the Engineering Databook to calculate this.

`

The code below contains an example script for how the class ***MSD_sim*** can be used. Make sure that your class works with the script below and produces the correct answers.

```
msd = MSD_sim;
msd.SetMass(1); % the class stores the mass = 1Kg
msd.SetDamper(1); % the class stores the damper coefficient = 1 Nm/s
msd.SetSpring(10); % the class stores the spring constant  = 10 N/m

wn = msd.CalcNaturalFreq() % the class returns the undamped natural frequency in wn
zeta = msd.CalcDampingFactor() % the class returns the damping factor in zeta
under = msd.isUnderDamped() % the class returns whether the systems is underdamped in variable under

% these lines of code generate a step response from t=0 to t=12
tvals = 0:0.1:12;
y = [];

for t = tvals
   newy = msd.CalcUnitStepResponse(t);
   y = [y newy];
end

figure;
plot(tvals,y); % this should plot the step response
```

Assignment Task 3 - Test script for MSD_sim class

Create a test script to test your ***MSD_sim*** class for assignment task 2. The name of your test script shall be: ***testMSD_sim.m***. Ensure that your test script can be run using:

```
Runtests('testMSD_sim.m');
```

Your test script should be done in the format shown in the labs, with each test starting with a double %% symbol (see TestSqrt.m).

Marks will be awarded for:

- Does a perfectly written class pass all your tests?
- Does the test script find basic errors in a class?
- Does the test script find subtle errors in a class?

The test script should work with any implementation that meets the requirements and not just your specific solution to Task 2.

Assignment Task 4 - Data Analysis Class

Using the template on moodle, create a class ***DataAnalysis*** that stores and performs some analysis on a dataset. You may wish to consult your notes/resources from ES2C7 for fitting the linear model.

1. The class shall have a method ***LoadData(x,y)*** which stores the data given in ***x*** and ***y***. Both ***x*** and ***y*** shall contain vectors of data points and it is expected that ***x*** and ***y*** both have the same size. The method shall accept vectors which are provided in column format (e.g. 5x1) or row format (e.g. 1x5).
2. For the method ***LoadData(x,y)***, an error will be given for any of the following cases:
    - ***x*** and ***y*** have different sizes (e.g. ***x*** is a 1x6 and ***y*** is a 5x1 matrix).
    - ***x*** and/or ***y*** is empty
    - ***x*** does not contain a column vector or row vector (e.g. a 5x2 matrix has been provided)
    - ***y*** does not contain a column vector or row vector (e.g. a 5x2 matrix has been provided)
3. The class shall have a method ***Max_x***() which returns the maximum value in the vector ***x***.
4. The class shall have a method ***Max_y***() which returns the maximum value in the vector ***y***.
5. The class shall have a method ***Mean_x***() which returns the mean value of the elements in vector ***x***.
6. The class shall have a method ***Mean_y***() which returns the mean value of the elements in vector ***y***.
7. The class shall have a method ***FitLinearModel()*** which fits a linear model ***y = mx+c*** to the data stored. This method will calculate and return the coefficients ***m*** and ***c*** as shown in this example:

```
da = DataAnalysis;
x = [0; 1; 2; 3];
y = [3; 5; 7; 9];
da.LoadData(x,y);
[m,c] = da.FitLinearModel() % returns the coefficients of the
model y=mx+c in the variables m and c
```

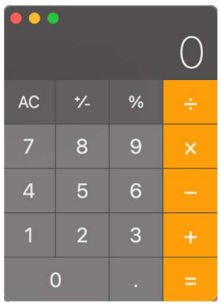
The example below shows how the class may be used:

```
da = DataAnalysis;
x = -3:3;
y = [-1.4 -0.7 1.8 4.8 6.6 8.9 10.3];
figure; plot(x,y,'r+');


da.LoadData(x',y');
[m,c] = da.FitLinearModel();


hold on;
```

```
plot(x,x*m+c,'b'); % plots a blue line of 'best fit' through the
datapoints
disp("the maximum x value is: " + num2str(da.Max_x()));
disp("the maximum y value is: " + num2str(da.Max_y()));
disp("the mean x value is: " + num2str(da.Mean_x()));
disp("the mean y value is: " + num2str(da.Mean_y()));
```

Assignment Task 5 (20%) – Calculator App

For this task, you are required to develop a Matlab app which implements a calculator. This is an open-ended problem and you can adjust this task to your level of difficulty.



As a minimum you should implement the following features:

- User can perform an add operation
- User can perform a minus operation
- User can perform a multiply operation
- User can perform a divide operation
- Clear button (AC)

–

The marks for this app will be assigned based upon:

- Number of features implemented • Accuracy of implementation
- Ease of use.

Ease of use includes:

- Layout of buttons and appearance of app
- Does the calculator behave how a user would expect?