

LEVEL 7 ASSESSMENT SPECIFICATION

Student name:		Student ID number:	
Programme:	MSc-MEng / Mechanical Engineering / Engineering Product Design		
Module:	Engineering Design and Analysis		
Module code:	M7X01379	Contribution to Overall Module Assessment (%):	50%
Lecturer:	Dr A. Marotin	Internal Verifier:	Dr A. Al-Hussany
Assignment Title:	Gravitational Force between Planets	Word count (or equivalent):	3,000 words
Submission deadline:	25/01/2019	Return date of provisional marks & written feedback:	03/03/2019
Submission method:	All written assessments, where practical and possible, must be submitted via Turnitin unless otherwise instructed by the Lecturer. (Please DO NOT put this assessment specification into Turnitin or it will match many similarities with other students' submissions.) Alternative submission method (if applicable): Late submission of the assessment will result in a late penalty mark. Penalties for late submission: Up to one week late, maximum mark of 50%. Over one week late, 0%. Only the Extenuating Circumstances Panel may approve a change to submission dates.		
Academic honesty / referencing:	Academic honesty is required. In the main body of your submission you must give credit to authors on whose research and ideas your work is based. Append to your submission a reference list that indicates the books, articles, etc. that you have used, cited or quoted in order to complete this assessment.		

Module Learning Outcomes (from module syllabus)	
1.	Demonstrate an ability to analyse engineering problems, with a view to generating appropriate design solutions.
2.	Demonstrate an ability to analyse a range of engineering design problems using appropriate scientific and mathematical principles to make engineering design judgements based on the outputs of the analysis

TASK DESCRIPTION

Objective:

The Physics and Astronomy departments want you to develop a programme that helps their students with astronomical calculations. The programme will mainly compute the gravitational force between different planets in our solar system, made of the Sun, the official eight planets plus a dwarf planet (In 2006, the planet Pluto was downgrade to a dwarf planet from being the ninth planet in our solar system).

Problem Description:

In 1680s, Newton formulated what is called Newton's law of universal gravitation. This law states that every two particles attract one another with a force that is proportional to the product of their masses and inversely proportional to the square of the distance between them (see Figure 1). The proportionality constant, denoted by G , is called the universal gravitational constant. This law can be applied for masses as small as electrons in an atom or for giant objects like the planets in the universe. Its mathematical form is,

$$F_1 = F_2 = G \cdot \frac{m_1 \times m_2}{r^2} \quad (1)$$

Where,

F is the force between the two masses measured in Newton (N)

G is the gravitational constant which is approximately equal to $6.674 \times 10^{-11} \text{ N.m}^2.\text{kg}^{-2}$

m_1 and m_2 are the masses of the objects in kg

r is the distance between the **centres** of the masses in meters (m).

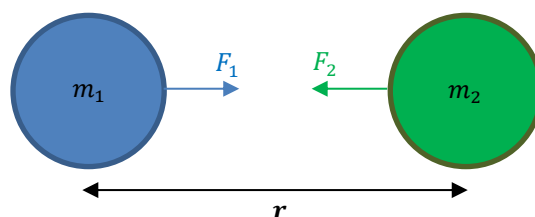


Figure 1: Gravitational force between two masses m_1 and m_2

The parameters of the Sun and each planet in our solar system are provided in an Excel file named DATA_PGCalc.xlsx. The Excel file can be found in Moodle and compiles the following table,

Data sheet 'Solar_System_a'

Planet	Mass (10 ²⁴ kg)	Diameter (km)	Distance from the Sun (AU)
Sun	1.99E+06	1.39E+06	0.000
Mercury	0.33	4879	0.390
Venus	4.87	12104	0.723
Earth	5.97	12756	1.000
Mars	0.642	6792	1.523
Jupiter	1899	142984	5.203
Saturn	568	120536	9.539
Uranus	86.8	51118	19.180
Neptune	102	49528	30.060
Pluto	0.0125	2390	39.530

Data sheet 'Solar_System_b'

From	To	Average (AU)
Mercury	Venus	0.33
Mercury	Earth	0.61
Mercury	Mars	1.13
Mercury	Jupiter	4.81
Mercury	Saturn	9.15
Mercury	Uranus	18.75
Mercury	Neptune	29.67
Mercury	Pluto	39.14
Venus	Earth	0.28
Venus	Mars	0.8
Venus	Jupiter	4.48
Venus	Saturn	8.82
Venus	Uranus	18.42
Venus	Neptune	29.34
Venus	Pluto	38.81
Earth	Mars	0.52
Earth	Jupiter	4.2
Earth	Saturn	8.54
Earth	Uranus	18.14
Earth	Neptune	29.06
Earth	Pluto	38.53
Mars	Jupiter	3.68
Mars	Saturn	8.02
Mars	Uranus	17.62
Mars	Neptune	28.54
Mars	Pluto	38.01
Jupiter	Saturn	4.34
Jupiter	Uranus	13.94
Jupiter	Neptune	24.86
Jupiter	Pluto	34.33
Saturn	Uranus	9.6
Saturn	Neptune	20.52
Saturn	Pluto	29.97
Uranus	Neptune	10.92
Uranus	Pluto	20.34
Pluto	Neptune	9.42

Distance in the solar system are very large. Therefore using meters to express the distances is not practical. As a result, in astronomy, astronomical unit (AU) is usually used to represent the distances. AU represents the distances normalised with respect to the Earth's distance from the Sun. Hence, the distance of the Earth from the Sun is 1 AU. In other words, we know that the average distance from the Earth to the Sun is approximately 149,600,000 km. Therefore,

$$1 \text{ AU} = 149,600,000 \text{ km} \quad (2)$$

The physics and astronomy department ask you to write a programme that helps the students to easily find the information and gravitational forces between different planets and the Sun in our solar system. Your tasks are as follows,

- i. Load the file DATA_PGCalc.xlsx which is a cell array that contains the table shown above.
Note: the input excel file can be found on Moodle.
- ii. Create and arrange the data given in the cell array into a vector of ten structures called *Solar*. Each object in this structure has the following fields: (a) Name, (b) Mass, (c) Diameter, and (d) Distance from the Sun. For example: `Solar(1).Name = 'Sun'`, `Solar(1).Mass = 1.989×1030`, `Solar(1).Diameter = 1390000×103`, and `Solar(1).Distance = 0`.
- iii. Create a function (Distance_Planets.m) that returns the distance between any two objects reading the information provided in Excel file DATA_PGCalc.xlsx. The input arguments of this function should be the two planet names. The output argument should be the distance (in AU).
- iv. Create a function (Gravitational_Force_Fields.m) that computes the gravitational force between any two objects using Equation 1. The input arguments of this function should be the two masses (in kilograms), and the distance between the two objects (in meters). The output argument should be the gravitational force (in Newton).
- v. You will have to also create a function (AU_m_Conversion.m) that will convert the distance from AU to meters (using Equation 2), because the distance used in the function you created in step (iii) should be in meters.
Note: You cannot make this conversion as part of the function in step (iii), because this is a conversion that is specifically used in astronomy, on the other hand, the function you created in step (iii) can be used with different applications.
- vi. In your main programme (that will make use of the above functions), the user should be prompt for a pair of objects in the solar system which she/he wishes to compute the gravitational force in between.
- vii. Now, it might happen that the user enters a wrong name of the object or does not know the full correct name of it. You can create a smarter programme that can precisely determined what the user want to input.
One intricate way to handle this, is that the programme guesses the name based on the first few letters in the input (i.e., comparing the first few letters to the list of objects there is), and then it can ask the user if he/she meant to input the guessed name.
Or more radically and easily, if the entered name is not recognised, the programme could offer a list of names to choose from.
- viii. Your programme should do the following,
 - ✓ Display the information of each object, distance between these two objects in km, and the gravitational force between these two objects.
 - ✓ Save these displayed data to an Excel output file that carries the name of the two objects.

Samples run of the required programme are shown below,

Sample Output 1 (PGCalc)

Input:

Welcome to PGCalc.

Please enter the name of the first object: **Earht**

Did you mean "Earth"? [1- Yes 2- No]: **1**

Please enter the name of the second object: **Man**

Did you mean "Mars"? [1- Yes 2- No]: **2**

Please enter the name of the second object: **Sun**

Output:

The Mass of the Earth is: $5.97\text{e}+24$ kg

The Diameter of the Earth is: 12756 km

The Mass of the Sun is: $1.989\text{e}+30$ kg

The Diameter of the Sun is: $1.39\text{e}+06$ km

The distance from the centre of the Earth to the centre of the Sun is: $1.496\text{e}+08$ km

The Gravitational Force between the Earth and the Sun is: $3.54105\text{e}+22$ N

The output data is saved in the excel file: EarthSun.xlsx

Thank you for using PGCalc.

Sample Output 2 (PGCalc)

Input:

Welcome to PGCalc.

Please enter the name of the first object: **Earht**

Name of entered planet is not recognised. Please choose from the following list,

1:Sun–2:Mercury–3:Venus–4:Earth–5:Mars–6:Jupiter–7:Saturn–8:Uranus–9:Neptune–10:Pluto

Selection: **4**

Please enter the name of the second object: **Man**

Name of entered planet is not recognised. Please choose from the following list,

1:Sun–2:Mercury–3:Venus–4:Earth–5:Mars–6:Jupiter–7:Saturn–8:Uranus–9:Neptune–10:Pluto

Selection: **a**

Selection should be a number between 1 and 10. Choose again,

Selection: **1**

Output:

The Mass of the Earth is: $5.97\text{e}+24$ kg

The Diameter of the Earth is: 12756 km

The Mass of the Sun is: $1.989\text{e}+30$ kg

The Diameter of the Sun is: $1.39\text{e}+06$ km

The distance from the centre of the Earth to the centre of the Sun is: $1.496\text{e}+08$ km

The Gravitational Force between the Earth and the Sun is: $3.54105\text{e}+22$ N

The output data is saved in the excel file: EarthSun.xlsx

Thank you for using PGCalc.

Extra Challenge (10 marks)

Astronomers might be interested in a single planet, and the relation between this planet and all other objects in the solar system. The way the data is presented affects the speed and clarity of interpretation of the programme's output. For example, sorting the data in ascending or descending order can help the user to process which object in the solar system has the highest effect and vice versa. In this part you are asked to create a programme that performs the following tasks:

- i. Prompt the user to enter name of the object of interest.
- ii. Using the function `Gravitational_Force_Fields.m`, calculate the gravitational force between the object of interest and all other objects in the solar system. Store these values in a vector that carries the name of the planet entered in step i.
- iii. Sort the gravitational forces stored in the vector created in step ii. In descending order. You can perform sorting using any sorting algorithm.

Important Note: You are not allowed to use any built-in sorting function in Matlab. You will need to implement your own sorting algorithm. Also, you need to clearly explain it in your report.

A sample run of the required programme is shown below,

Sample Output 1 (Sorting_PGCalc)	
<u>Input:</u>	
Welcome to Sorting_PGCalc. Please enter the name of the first object: Venus	
<u>Output:</u>	
The Gravitational Force between Venus and all other objects in the solar system (in descending order) are:	
Object's Name	Gravitational Force
'Sun'	[5.52599e+22]
'Jupiter'	[1.37411e+18]
'Earth'	[1.10589e+18]
'Saturn'	[1.06038e+17]
'Mercury'	[4.40086e+16]
'Mars'	[1.45682e+16]
'Uranus'	[3.71529e+15]
'Neptune'	[1.72081e+15]
'Pluto'	[1.20524e+11]
<i>Thank you for using Sorting_PGCalc.</i>	

You must complete the following:

1. Sources files:
 - PGCalc.m
 - Distance_Planets.m
 - Gravitational_Force_Fields.m
 - AU_m_Conversion.m
 - Sorting_PGCalc.m
2. A report providing clear and thorough explanations of your scripts.

Your report should contain the Matlab scripts of all source files in appendices. The report must be handed in through Turnitin. Any sources used must be referenced correctly using Harvard referencing. The Matlab scripts will also need to be emailed to the lecturer at: arnaud.marotin@uwtsd.ac.uk

GUIDANCE FOR STUDENTS IN THE COMPLETION OF TASKS

Program Correctness

This refers to:

- 1- whether and how well the program meets the requirements specified in the assignment,
- 2- how well the program addresses the problem specified in the assignment, and
- 3- whether and how logically correct the solution is. It is possible that your code may yield incorrect results with logically correct solution to the problem specified in the assignment. It is also possible that your code may yield correct results with logically incorrect solution to the problem specified in the assignment.

Software Design

This refers whether:

- 1- the overall software organization is clear and sensible,
- 2- the program file naming convention and data file naming convention.

A solution that gives correct results may be poorly designed. Software clarity or simplicity, consistency, modularity, extensibility, and maintainability are a few important principles of good software design.

Programming Style

This refers to:

- 1- good and clear indentations (e.g., a tab or at least three spaces for an indentation),
- 2- good use of variable names (e.g., `numberOfStudents` is better than `x`),
- 3- proper declaration of global constants,
- 4- efficient use of functions to prevent redundant coding,
- 5- adequate length of each method functions, and so on.

The consistency of your programming style throughout your code is extremely important. It allows readers of your code to understand your code much more easily.

Testing

This refers to the grader testing of your program, using his own input data file and matching your output to the correct output. Boundary conditions will be tested to see whether your program can handle those conditions. For example, suppose that your program reads an input integer from the screen and processes it. Will your program still function if instead a real number is entered?

Documentation

This refers to all the good practices regarding the report. The report should include:

- 1- an indicative contents underlining the report structure,
- 2- introduction with description of what the code does,
- 3- description of how the code achieves what it wants to do,
- 4- at times, why the code uses a particular solution strategy,
- 5- thorough testing of the code,
- 6- conclusion
- 7- the complete Matlab scripts as appendices.

STUDENT FEEDBACK FORM

This section details the *extent* to which the assessment criteria are demonstrated by you, which in turn determines your mark. The marks available for each category of skill are shown. Lecturers will use the space provided to comment on the achievement of the task(s), including those areas in which you have performed well and areas that would benefit from development/improvement.

Generic Assessment Criteria	Marks available	Marks awarded
1. Program Correctness.		
	45	
2. Programming Style		
	15	
3. Software Design		
	10	
4. Testing		
	10	
5. Documentation		
	10	
6. Extra Challenge		
	10	

Assessment Mark (Assessment marks are subject to ratification at the Exam Board. These comments and marks are to give feedback on module work and are for guidance only until they are confirmed.)	Late Submission Penalties (tick if appropriate)		%
	Up to 1 week late (50% Max)		
	Over 1 week late (0%)		