Assignment 1 (10 Marks)

ENGG1001: Programming for Engineers – Semester 1, 2022

Due: Friday, 1st April 4pm

Introduction

You are required to determine a suitable length for an aeroplane runway that is needed for a new airport. The runway must be able to accommodate planes of varying mass (m) and engine force (F_{thrust}) and must be able to operate under varying environmental conditions.

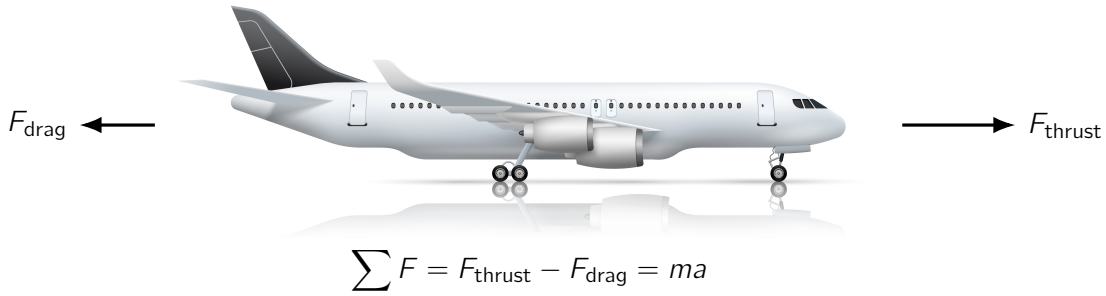

Figure 1: *Force diagram on a taxiing aircraft*

The acceleration, velocity, and position of the plane along the runway can be calculated iteratively with the following equations:

$$a_i = \frac{1}{m}\left(F_{\text{thrust}} - \overbrace{\frac{1}{2}\rho v_i^2 A_{\text{ref}} C_D}^{F_{\text{drag}}}\right) \quad (1)$$

$$v_{i+1} = v_i + a_i \Delta t \quad (2)$$

$$x_{i+1} = x_i + v_i \Delta t + \frac{1}{2} a_i (\Delta t)^2 \quad (3)$$

m	mass of the plane
F_{thrust}	engine force
A_{ref}	reference area
ρ	air density
v_0	initial velocity at start of runway
v_{lift}	lift off velocity
x_0	position of the start of the runway
Δt	time increment
C_D	drag coefficient

Where x is the distance travelled in the horizontal direction, v is the velocity in the horizontal direction, a is the acceleration in the horizontal direction, the subscript i denotes a value at t_i and Δt is the time interval at which successive new values of x_{i+1} and v_{i+1} are computed.

You need to write a program that will enable you to calculate how far a plane must travel for lift-off to occur, given a set of relevant parameters (e.g.: mass, engine force, drag coefficient, air density, lift-off velocity, etc.). We will call the velocity at which lift-off occurs v_{lift}.

To help structure your program, the functionality has been broken into a number of tasks. Follow these tasks in order, and test the code as you progress with building the program.

Task 1

Write a function to prompt the user for inputs.

```
def prompt_for_inputs() -> tuple[tuple[float, ...], float]:
    """
    Returns: values: tuple[float, ...], drag_coeff: float

    The first tuple of floats should contain the following values:
    mass, force, ref_area, density, init_velocity, lift_velocity,
    start_position, time_inc
    """
    ...
```

This function prompts the user to input various parameters at the keyboard and then returns the parameters. These parameters include the mass, engine force, reference area, air density, initial velocity, lift-off velocity, position of the start of the runway, the time increment used for calculating distances and velocities, and drag coefficient.

Example

```
>>> values, drag_coeff = prompt_for_inputs()
Input mass of the plane (in kg): 50000
Input engine force (in N): 600000
Input reference area (in m^2): 800
Input air density (in kg/m^3): 1
Input initial velocity at start of runway (in m/s): 0
Input lift-off velocity (in m/s): 70
Input position of the start of the runway (in m): 0
Input time increment (in secs): 0.1
Input drag coefficient: 0.015
>>> values
(50000.0, 600000.0, 0.015, 800.0, 1.0, 0.0, 70.0, 0.0, 0.1)
>>> drag_coeff
0.015
```

Task 2

Write a function to compute the trajectory.

```
def compute_trajectory(values: tuple[float, ...], drag_coeff: float) ->
    tuple[tuple[float, ...], tuple[float, ...]]:
    """
    Parameters:
        values (tuple[float, ...]): mass, force, ref_area, density,
            init_velocity, lift_velocity, start_position, time_inc
        drag_coeff (float): The drag coefficient.
    Returns:
        (tuple[tuple[float, ...], tuple[float, ...]]): The first tuple
            contains the positions (in meters) along the runway for each
```

```
            successive increment of time. The second tuple contains the
            velocities at successive increments in time.
    """
    ...
```

This function takes in as arguments various parameters determined in **Task 1** and returns two tuples. *After computing the position and velocities, all returned values should be rounded to three decimal places.*

Note: The last value of both tuples represents the position and velocity just before lift-off.

Example

```
>>> positions, velocities = compute_trajectory(
...     (50000.0, 600000.0, 800.0, 1.0, 0.0, 70.0, 0.0, 0.1), 0.015
... )
>>> positions
(0.06, 0.24, 0.54, 0.96, 1.5, 2.16, 2.94, 3.84, 4.859, 5.999, 7.258, 8.637, 10.1
36, 11.755, 13.494, 15.352, 17.329, 19.426, 21.643, 23.979, 26.435, 29.009, 31.7
03, 34.516, 37.448, 40.499, 43.669, 46.958, 50.365, 53.891, 57.536, 61.299, 65.1
8, 69.179, 73.297, 77.532, 81.886, 86.357, 90.945, 95.651, 100.475, 105.415, 110
.473, 115.648, 120.939, 126.347, 131.872, 137.513, 143.27, 149.143, 155.132, 161
.237, 167.457, 173.792, 180.243, 186.809, 193.489, 200.285, 207.194, 214.218)
>>> velocities
(1.2, 2.4, 3.6, 4.8, 5.999, 7.199, 8.398, 9.598, 10.796, 11.995, 13.193, 14.391,
15.589, 16.786, 17.982, 19.179, 20.374, 21.569, 22.764, 23.957, 25.151, 26.343,
27.535, 28.725, 29.916, 31.105, 32.293, 33.481, 34.667, 35.853, 37.037, 38.221,
39.403, 40.585, 41.765, 42.944, 44.122, 45.299, 46.474, 47.648, 48.821, 49.992,
51.162, 52.331, 53.498, 54.664, 55.828, 56.99, 58.151, 59.311, 60.469, 61.625,
62.779, 63.932, 65.083, 66.232, 67.379, 68.525, 69.669, 70.81)
```

Task 3

```
def print_table(
    values: tuple[float, ...], drag_coeff: float, increments: int, step: float
) -> None:
    """
    Parameters:
        values (tuple[float, ...]): mass, force, ref_area, density,
            init_velocity, lift_velocity, start_position, time_inc
        drag_coeff (float): The drag coefficient.
        increments (int): The number of drag coefficients displayed.
        step (float): The difference between each drag coefficient.

    Returns:
        None
    """

```

For this function you need to compute the distance before lift-off for a range of drag coefficients and then you need to print these results in a table.

The drag coefficient of an aeroplane has a significant impact on the plane's ability to lift-off. If the drag coefficient is sufficiently high, the plane will not actually be able to generate enough speed to lift off. In this task you will write a function which will explore this phenomenon.

Example

```
>>> values = (50000.0, 600000.0, 800.0, 1.0, 0.0, 70.0, 0.0, 0.1)
>>> drag_coeff = 0.015
>>> print_table(values, drag_coeff, 10, 0.03)
**************************************
* Drag coefficient * Runway distance *
**************************************
*      0.015       *     214.218     *
*      0.045       *     224.707     *
*      0.075       *     234.859     *
*      0.105       *     251.633     *
*      0.135       *     267.901     *
*      0.165       *     297.560     *
*      0.195       *     326.334     *
*      0.225       *     367.907     *
*      0.255       *     435.575     *
*      0.285       *     583.499     *
**************************************
```

Hint: You should create the above table by using f-string formatting. Check the appendix for layout dimensions.

Task 4

For this task you need to write a function `main()`, which has no parameters and does not return anything. This function handles the user interaction. *Hint:* Make use of functions that were written in previous tasks.

Example 1

```
Please enter a command: h

    'i' - prompt for the input parameters
    'p <increments> <step>' - print out a table of distances to lift-off for
↪   different drag coefficients
    'h' - help message
    'q' - quit

Please enter a command: i
Input mass of the plane (in kg): 50000
Input engine force (in N): 600000
Input reference area (in m^2): 800
Input air density (in kg/m^3): 1
Input initial velocity at start of runway (in m/s): 0
```

```
Input lift-off velocity (in m/s): 70
Input position of the start of the runway (in m): 0
Input time increment (in secs): 0.1
Input drag coefficient: 0.015
Please enter a command: p 10 0.03
**************************************
* Drag coefficient * Runway distance *
**************************************
*      0.015       *     214.218     *
*      0.045       *     224.707     *
*      0.075       *     234.859     *
*      0.105       *     251.633     *
*      0.135       *     267.901     *
*      0.165       *     297.560     *
*      0.195       *     326.334     *
*      0.225       *     367.907     *
*      0.255       *     435.575     *
*      0.285       *     583.499     *
**************************************

Please enter a command: q
Are you sure (y/n): n
Please enter a command: q
Are you sure (y/n): y
>>>
```

Example 2

```
Please enter a command: p 10 0.03
Please enter the parameters first.
Please enter a command: q
Are you sure (y/n): n
Please enter a command: i
Input mass of the plane (in kg): 50000
Input engine force (in N): 600000
Input reference area (in m^2): 800
Input air density (in kg/m^3): 1
Input initial velocity at start of runway (in m/s): 0
Input lift-off velocity (in m/s): 70
Input position of the start of the runway (in m): 0
Input time increment (in secs): 0.1
Input drag coefficient: 0.015
Please enter a command: p 10 0.03
**************************************
* Drag coefficient * Runway distance *
**************************************
*      0.015       *     214.218     *
*      0.045       *     224.707     *
*      0.075       *     234.859     *
*      0.105       *     251.633     *
*      0.135       *     267.901     *
*      0.165       *     297.560     *
```

```
*      0.195       *     326.334     *
*      0.225       *     367.907     *
*      0.255       *     435.575     *
*      0.285       *     583.499     *
************************************

Please enter a command: q
Are you sure (y/n): y
```

Task 5 (Optional task for 1 bonus mark)

You can get up to 1 bonus mark by doing the task specified below. Note that although the bonus mark would allow you to improve your mark, you cannot get more that 10 out of 10 for your final score. i.e. your final mark for the assignment will be capped at 10 out of 10.

Add an extra function, `bonus` in your program to calculate the maximum drag coefficient that can be accommodated for a given length of runway. Assume that the mass of the plane is 50 000 kg, the thrust is 600 000 N, the reference area is 800 m^2, the air density is 1 kg/m^3, the initial velocity at the start of the runway is 0 m/s, the lift-off velocity is 70 m/s, the plane starts at 0 m along the runway and the time increment for iteratively calculating the distance and velocity is 0.01 s. Assume also that the runway can be between 500 m and 2000 m in length. The function must print the drag coefficient which is rounded to 5 decimal places. A sample printout is given below.

```
1  def bonus(values: tuple[float, ...], runway_distance: float) -> None:
2      """
3      Parameters:
4          values (tuple[float, ...]): mass, force, ref_area, density,
5              init_velocity, lift_velocity, start_position, time_inc
6          runway_distance (float): The runway distance.
7
8      Returns:
9          None
10     """
11
```

A suggested approach is to write some code which loops through the possible values of drag coefficient and find the one which corresponds most closely to the required runway length. Do not search for drag coefficient values greater than 0.30611[1]. You may find it useful to do the looping in at least two stages so that the processing does not take too long. In the first stage you can find a coarse estimate for the drag coefficient and in the second stage you can find the estimate accurate to 5 decimal places.

Example 1

```
>>> bonus(((50000, 600000, 800, 1, 0, 70, 0, 0.1)), 1500)
0.30594
```

[1] At this value, the thrust produced isn't enough to overcome the drag at 70 m/s, and the plane can't take off!

Note that this function will be marked separately and no automated tests will be made available to you.

Writing your code

You must download the file, **a1.py**, from Blackboard, and write your code in that file. When you submit your assignment to Gradescope **you must submit only one file and it must be called a1.py**. Do not submit any other files or it can disrupt the automatic code testing program which will grade your assignments.

Design

You are expected to use good programming practice in your code, and you must incorporate at least the functions described in the previous part of the assignment sheet.

Assessment and Marking Criteria

The maximum achievable mark for this assignment is 10 marks. This is *including* the bonus marks if you are granted those marks.

Functionality Assessment — 7 Marks

The functionality will be marked out of 7. Your assignment will be put through a series of tests and your functionality mark will be proportional to the number of tests you pass. If, say, there are 25 functionality tests and you pass 20 of them, then your functionality mark will be $20/25 \times 7$. You will be given the functionality tests before the due date for the assignment so that you can gain a good idea of the correctness of your assignment yourself before submitting. You should, however, make sure that your program meets all the specifications given in the assignment. That will ensure that your code passes all the tests. Note: Functionality tests are automated and so string outputs need to exactly match what is expected. Do not leave it until the last minute because it generally takes time to get your code to pass the tests.

Code Style Assessment — 3 Marks

The style of your assignment will be assessed by one of the tutors, and you will be marked according to the style rubric provided with the assignment. The style mark will be out of 3.

Assignment Submission

You must submit your completed assignment to Gradescope. The only file you submit should be a single Python file called **a1.py** (use this name — all lower case). This should be uploaded to Gradescope. You may submit your assignment multiple times before the deadline — only the last submission will be marked.

Late submission of the assignment will **not** be accepted. In the event of exceptional personal or medical circumstances that prevent you from handing in the assignment on time, you may submit a request for an extension. See the course profile for details on how to apply for an extension.

Requests for extensions **must** be made according to the guidelines in the ECP.

Appendix A: Layout Dimensions

Table dimensions:

```
**************************************
* Drag coefficient * Runway distance *
**************************************
*      0.015       *     214.218     *
*      0.045       *     224.707     *
*      0.075       *     234.859     *
*      0.105       *     251.633     *
*      0.135       *     267.901     *
*      0.165       *     297.56      *
*      0.195       *     326.334     *
*      0.225       *     367.907     *
*      0.255       *     435.575     *
*      0.285       *     583.499     *
**************************************
 |<-- 18 chars -->|  |<-- 17 chars -->|
```