JABATAN KEJURUTERAAN MEKANIK
Department of Mechanical Engineering

LAPORAN KERTAS KERJA
Assignment Report

Sesi & Sem ***Session & Sem***	2021/2022 Semester 2
Kod & Nama Kursus ***Course Code & Name***	KIG2007 Computer Programming
Tajuk Program ***Title of Program***	Lift and Drag of a Commercial Airplane
Nama & No. Kad Matrix ***Name & Matrix Number***	1. Muhammad Hafizzudin bin Kamaluddin (U2003959/1) 2. Hassanal Hakim bin Ahmed Hakimmy Fuad (U2003881/1) 3. Muhammad Danial bin Zulkifli (U2003961/1) 4. Adli Al-Wafi bin Mohd Normarzuki (U2003982/1)
Nama Pensyarah ***Lecturer's Name***	Dr. Muhammad Khairi Faiz bin Ahmad Hairuddin
Tarikh Hantar ***Submission Date***	23th June 2022

Table of Content

1.0 INTRODUCTION

1.1 Background

Airplanes come in many different shapes and sizes depending on the use of the aircraft. For instance, commercial airplanes, jets, cargo and many more. However, all modern airplanes have some component in common. These are the fuselage, wing, tail assembly and control surfaces and landing gear. To fly an airplane, four forces are needed to be exerted to form a resultant force called Aerodynamic Resultant. Aerodynamic Resultant is a total force exerted on the airplane in combination of thrust, weight (gravity), lift and drag in its own direction. Meaning to say that if one of the forces does not exist, the airplane cannot fly.

The forces act in two different directions which are in vertical and horizontal directions. Thrust force and weight are both act horizontally on the airplane. It is just that they acted in opposite directions. Thrust force is a forward produced by the jet engine/propeller that moves the airplane forward. On the other hand, drag force is act as braking action that oppose the direction of thrust. It is caused by air flow disruption highly by the wing. Lift force and gravity(weight) act vertically on the airplane. Lift force pushes the airplane upward whereas gravity(weight) is a force that counteracts the effect of lift which pulls the airplane toward the earth.

Lift and drag coefficient are very important parameters to under the lift and drag characteristics during takeoff and landing. Lift coefficient contributes to the generation of lift which means the higher the lift coefficient, the higher the lift. On the contrary, drag coefficient is the ratio of drag pressure and dynamic pressure that contribute to drag formation. Each coefficient incorporates the same complex variables which are the shape and area of wingspan, angle of attack, velocity of flow (equivalent to velocity of flying body) and flow condition such as density and viscosity. It is worth noting that lift coefficient increases linearly with angle of attack until certain degree where the lift coefficient drops. This phenomenon called stall.

Airplanes need to generate maximum lift force so that the airplane is enable to takeoff and landing at the lowest and safest possible speeds to avoid any accident if something gone wrong. One way to do it is by pitching up the whole airplane and thus it will enlarge the angle of attack that leads to a higher lift. In addition, the use of high lift devices that are design to increase the lift produced by the wing. Some famous devices used are flaps, flaperons, slats and slots. Flaps are found on most airplane wings. They are usually inboard on the wing's trailing edges adjacent to fuselage. The flaps are used to alter the shape wings during takeoff and landing. Then, the flap will retract and return to normal shape while cruising.

Thrust power is an important parameter to analyze the jet engine/propeller performance. Thrust power is power needed for jet engine to overcome the drag force generated to push the airplane forward. It can be calculated by multiplying thrust with cruising velocity. This parameter is useful to calculate the fuel consumptions, efficiency, and R&D purposes for future improvement. However, the power determined does not include the drag that acts on the remaining parts of the aircraft such as fuselage, tail, propeller etc. Thus, theoretically, the total amount of energy required is much higher for different propulsion efficiency, η. Propulsive efficiency is defined here as the propulsive power delivered to the aircraft (which is equal to thrust times airspeed) divided by the shaft power input to the propulsor.

1.2 Objective

There are a few objectives of this case study:

1. To create a program to calculate the minimum safe velocity required before stall for each type of flap (without, single, double) during takeoff.
2. To create a program to identify the lift and drag characteristics for each type of flap (without, single, double) for different altitude.
3. To create a program to determine the power required to overcome drag for each type of flap (without, single, double) for different altitude.
4. To create a program to determine the energy input rate of fuel required by the airplane to produce the power required for each type of engine (Turboprop, Turbojet, Turboshaft, Turbofan, Ramjet).

2.0 THEORETICAL BACKGROUND

2.1 List of Input

In this program, the input that were inserted are shown below:

1. The mass of airplane (kg)
2. The length of wing cord (m)
3. The length of wingspan (m)
4. The cruising velocity (km/h)
5. The cruising altitude (m)
6. The type of flap (without flap, single flapped, double flapped)
7. The type of engine (Turboprop, Turbojet, Turboshaft, Turbofan, Ramjet)

2.2 Equations / Processes

The calculation starts by calculating the weight of the airplane using the formula below,

$$W = mg$$

Where,

W = airplane's weight (N)

m = airplane's mass (kg)

g = acceleration due to gravity (m/s^2)

Next, calculate the wing planform. The user input the length of wing cord and wingspan. Thus, the area can be calculated by using

$$A_w = L_c \times L_s$$

Where,

A_w= area of wing planform (m^2)

L_c = length of wing cord (m)

L_s = length of wingspan (m)

Before calculating the minimum velocity stall, the velocity unit need to be converted from km/h to m/s

$$V_{c,m/s} = V_{c,km/h}(\frac{1\,km}{1000\,m})(\frac{1\,h}{3600\,s})$$

Where,

V_c = the cruising velocity (input by user)

The minimum velocity needed before stall can be calculated by

$$V_{min} = \sqrt{\frac{2W}{\rho C_{L,max} A_w}}$$

Where,

V_{min} = minimum velocity before stall (m/s)

W = airplane's weight (N)

ρ = air density (different altitude) (kg/m^2)

A_w = area of wing planform

$C_{L,max}$ = maximum lift coefficient (different type of flap)

The lift characteristics of the airplane wings is assumed to be approximated by NACA 23012. Thus, for different type of flap chosen, the maximum lift coefficient can be determined using the figure below.

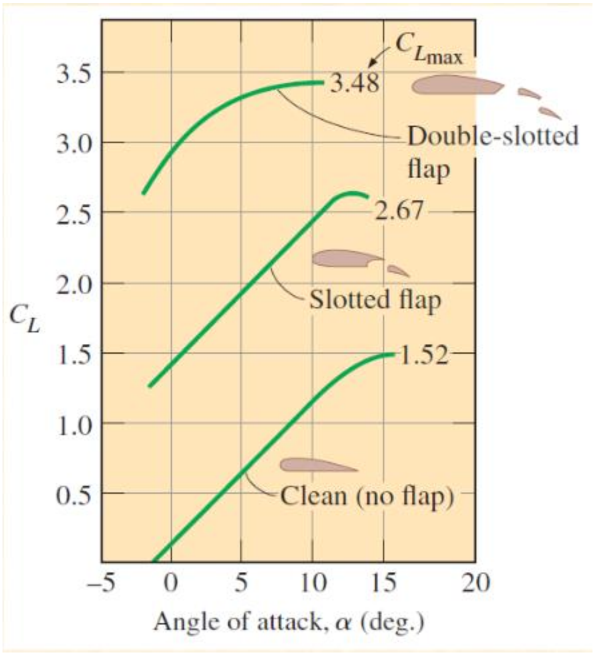


Figure 1: Lift Coefficients of Different Types of Flaps against Angle of Attack

Type of flap	Without flap	Single flapped	Double flapped
Maximum lift coefficient	1.52	2.67	3.48

Then the “safe” minimum velocity to avoid the stall region are obtained by multiplying the values by 1.2:

$$V_{min,safe} = 1.2V_{min}$$

The lift coefficient can be studied at certain lift force and cruising speed by using the formula below. Note that when an airplane is cruising steadily at a constant altitude, the lift must be equal to the weight of the airplane, $F_L = W$. Then, the lift coefficient is

$$C_L = \frac{F_L}{\frac{1}{2}\rho V_c^2 A_w}$$

Where,

C_L = lift coefficient
F_L = airplane’s lift force (N)
ρ = air density (different altitude) (kg/m^2)
V_c = the cruising velocity (input by user) (m/s)
A_w = area of wing planform (m^2)

When the airplane is cruising steadily at a constant altitude, the net force acting on the airplane is zero. Thus, the thrust provided by the jet engines must be equal to the drag force.

$$F_D = C_D A_w \frac{\rho V_c^2}{2}$$

Where,

C_D = drag coefficient
F_D = airplane’s drag force (N)
ρ = air density (different altitude) (kg/m^2)
V_c = the cruising velocity (input by user) (m/s)
A_w = area of wing planform (m^2)

The drag characteristics of the airplane wings is assumed to be approximated by NACA 23012. Thus, the drag coefficient can be determined by taking the average value based on different type of flap using the figure below.

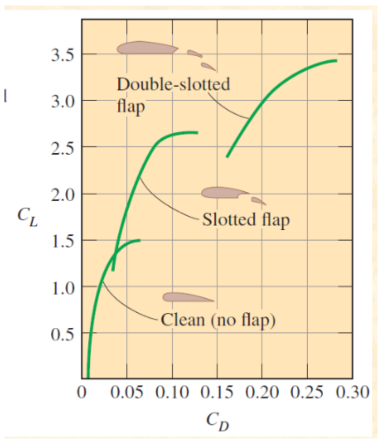


Figure 2: Lift Coefficient against Drag Coefficients for Different Types of Flaps

Type of flap	Without flap	Single flapped	Double flapped
Drag coefficient	0.04	0.085	0.22

Before calculating the thrust power, the unit of drag force need to be converted from N to kN.

$$F_{D,kN} = F_{D,N}\left(\frac{1\ kN}{1000\ N}\right)$$

The power required to overcome the wing drag is

$$P = F_D \times V_c$$

Where,

P = thrust power (kW)
F_D = airplane’s drag force (kN)
V_c = the cruising velocity (input by user) (m/s)

For a standard airplane the propulsion efficiency is between 25% to 85%, depending on the type of engine. The propulsive energy can be obtained by dividing the shaft power input to the propulsor and the propulsive efficiency.

$$P_e = \frac{P}{\eta}$$

Where,

P_e = propulsion energy (kW)
P = thrust power (kW)
η = propulsive efficiency

The propulsion efficiencies of some of the most common aircraft engines are as follows:

Turboprop Engine:	85%
Turbojet Engine:	30%
Turboshaft Engine:	25%
Turbofan Engine:	75%
Ramjet Engine:	40%

2.3 List of Output

The program will display output as below:

1. Weight of the airplane (N)
2. Planform area of the wings (m^2)
3. Density of the air at cruising altitude (kg/m^3)
4. Cruising velocity of the airplane (m/s)
5. Minimum takeoff velocity of the airplane (m/s)
6. Safe minimum takeoff velocity of the airplane (km/h)
7. Lift coefficient at cruising altitude
8. Drag coefficient at cruising altitude
9. Drag force at cruising altitude (kN)
10. Power needed to overcome drag (kW)
11. Propulsion power efficiency (%)
12. Energy input rate required to produce required power (kJ/s)

3.0 METHODOLOGY

3.1 Coding Approach

3.1.1 Headers

```
#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;
```

From line 1-3. 3 different standard library which are **<iostream>, <iomanip>** and **<cmath>** are used in this coding. **<iostream>** is for declaring functions for standard input or output which we can call it cin and cout respectively. **<iomanip>** is there also to control the format of input and output for example, if we wanted to set precision of a number, set the field width for next IO operation, and set the alignment of the input and output. **<cmath>** is used to preform mathematical functions for equations that use power formula, **pow().**

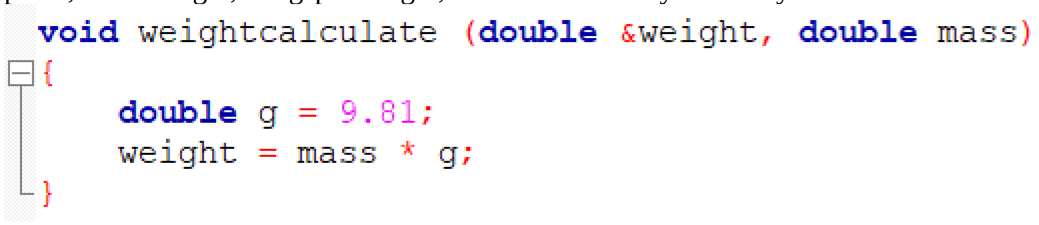
3.1.2 Declaration and Initialization

```
double safevmin, safevminkmh, FD, FD_kN, CD, energyrate, efficiency;

void weightcalculate (double &, double);
void areacalculate(double &, double, double);
void cruisingspeedconversion(double &);
void densityaltitude(double &);
void vmincalculate(double &, double, double);
void safevmincalculate(double &, double &, double);
void liftcoefficient (double &, double, double, double, double);
void dragcoefficient (double &, double);
void dragforce(double &, double, double, double, double);
void dragconversion(double &, double);
void powercalculate(double &, double, double);
void energyrequired(double &, double);
void displayoutput(double, double, double, double, double, double, double, double, double, double, double, double, double, double);
```

From line 7-21, global variables and function prototype that specifies arguments were declared. Global variables are used so that there's no need to create any objects before calling the function while function prototype that consist of 13 functions, all with no return value (void function) serve as a collection of statements or set of statements that perform the task together. In **main()** function, from line 25 to 26, all the variables are also declared.

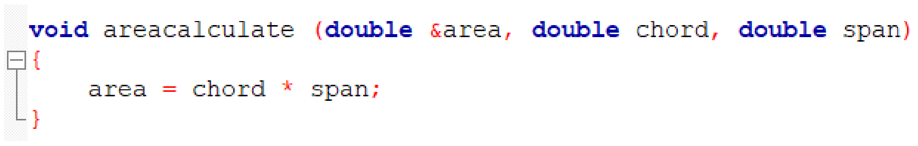
3.1.3 Input of Program

This program executes from line 23 which is from **main()** function. A do … while loop from line 28 – 65 is there to make sure the user can choose to input the values again or exit the program. At the end of the program, the option is there for user to continue the program or exit. If user entered 'Y' the program will continue, otherwise the program will end. In this function also, user needs to input value of mass of plane, chord length, wingspan length, and cruise velocity manually.



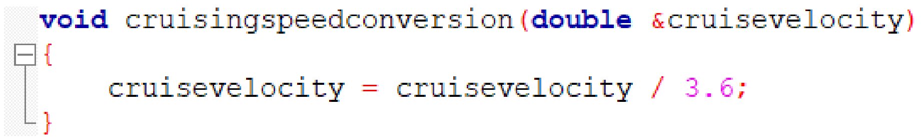
```
void weightcalculate (double &weight, double mass)
{
    double g = 9.81;
    weight = mass * g;
}
```

After user enter inputs needed, first function call, **weightcalculate()** is used to calculate the mass of the plane by declaring acceleration of gravity, g = 9.81 times with the mass of plane that user input. This function has no return value because the **weight** variable has been passed by reference from **main()**.



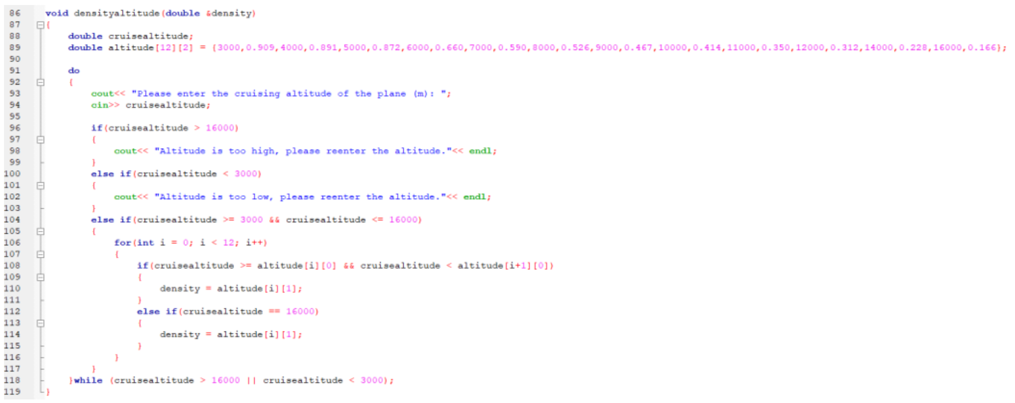
```
void areacalculate (double &area, double chord, double span)
{
    area = chord * span;
}
```

The program continue with second function call in the **main(), areacalculate().** This function calculates the area of wing planform using wingspan and chord length that user inputs. This function has no return value as area is passed by reference.



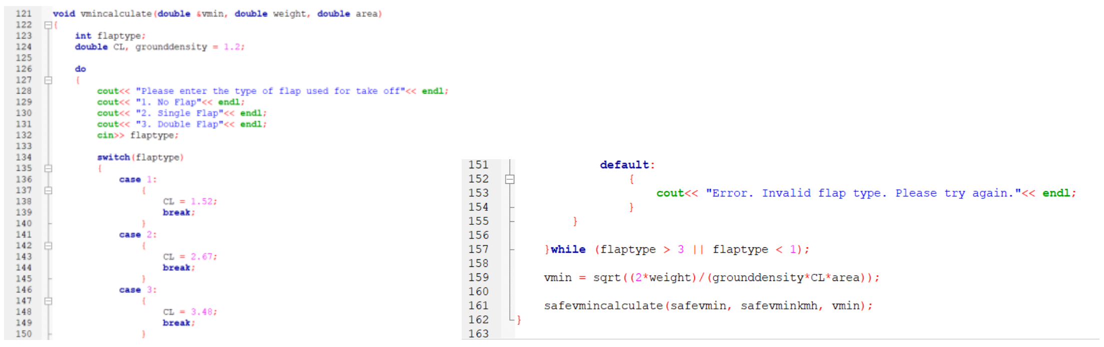
```
void cruisingspeedconversion(double &cruisevelocity)
{
    cruisevelocity = cruisevelocity / 3.6;
}
```

Next, **cruisingspeedconversion()** is called to convert cruise velocity from km/h to m/s using formula cruisevelocity = cruisevelocity/3.6.



```
86  void densityaltitude(double &density)
87  {
88      double cruisealtitude;
89      double altitude[12][2] = {3000,0.909,4000,0.891,5000,0.872,6000,0.660,7000,0.590,8000,0.526,9000,0.467,10000,0.414,11000,0.350,12000,0.312,14000,0.228,16000,0.166};
90
91      do
92      {
93          cout<< "Please enter the cruising altitude of the plane (m): ";
94          cin>> cruisealtitude;
95
96          if(cruisealtitude > 16000)
97          {
98              cout<< "Altitude is too high, please reenter the altitude."<< endl;
99          }
100         else if(cruisealtitude < 3000)
101         {
102             cout<< "Altitude is too low, please reenter the altitude."<< endl;
103         }
104         else if(cruisealtitude >= 3000 && cruisealtitude <= 16000)
105         {
106             for(int i = 0; i < 12; i++)
107             {
108                 if(cruisealtitude >= altitude[i][0] && cruisealtitude < altitude[i+1][0])
109                 {
110                     density = altitude[i][1];
111                 }
112                 else if(cruisealtitude == 16000)
113                 {
114                     density = altitude[i][1];
115                 }
116             }
117         }
118     }while (cruisealtitude > 16000 || cruisealtitude < 3000);
119 }
```

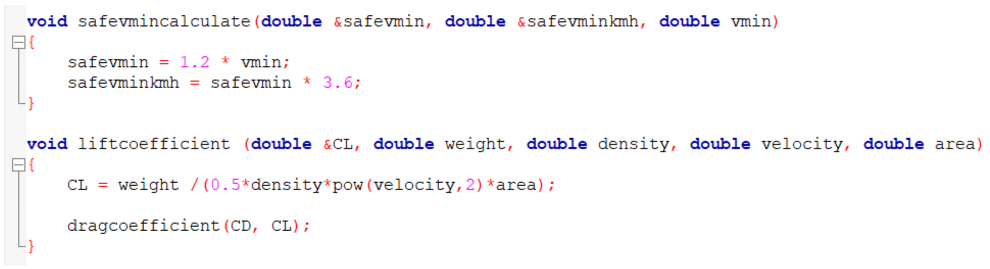
Furthermore, the program continue with **densityaltitude()** function call from line 88 to 118. In this function, we use array to declare densities of air at cruising altitude from 0 meter to 12000 meters. **Do… while statement** is used from line 91 to 118 to make sure user does not enter altitude more than 12000 or less than 0 meter. After user input an altitude, for statement is used to read density of the air at input altitude and initialize the density variable from the value of density in the array.



```
void vmincalculate(double &vmin, double weight, double area)
{
    int flaptype;
    double CL, grounddensity = 1.2;

    do
    {
        cout<< "Please enter the type of flap used for take off"<< endl;
        cout<< "1. No Flap"<< endl;
        cout<< "2. Single Flap"<< endl;
        cout<< "3. Double Flap"<< endl;
        cin>> flaptype;

        switch(flaptype)
        {
            case 1:
                {
                    CL = 1.52;
                    break;
                }
            case 2:
                {
                    CL = 2.67;
                    break;
                }
            case 3:
                {
                    CL = 3.48;
                    break;
                }
            default:
                {
                    cout<< "Error. Invalid flap type. Please try again."<< endl;
                }
        }

    }while (flaptype > 3 || flaptype < 1);

    vmin = sqrt((2*weight)/(grounddensity*CL*area));

    safevmincalculate(safevmin, safevminkmh, vmin);
}
```

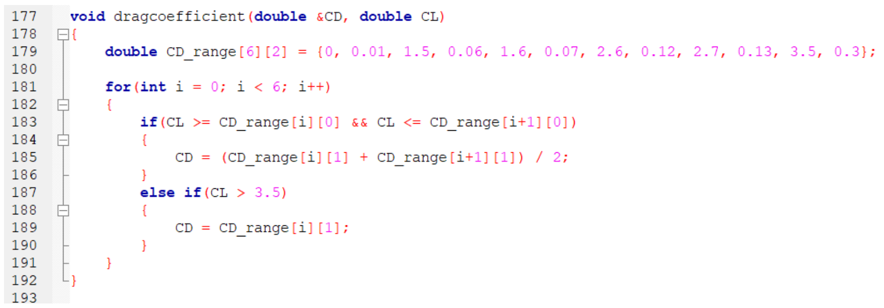
The function **vmincalculate()** is use to find lift coefficient for flap that user want to use and minimum velocity corresponding to stall conditions (type of flap used). In this function, we also use do… while statement. In this statement user need to input type of flap used for the wing (1. No Flap, 2. Single Flap, 3. Double Flap). If user input number less than 1 or bigger than 3 the program will ask user to input the value again. Switch statement is also used. If user input 1 which is No flap, the lift coefficient, CL =

1.52, for case 2 (Single Flap), CL = 2.67 and for case 3 (Double Flap), CL = 3.48. After that, the program will calculate minimum velocity, vmin. Lastly this function will call another function, **safevmincalculate()**.



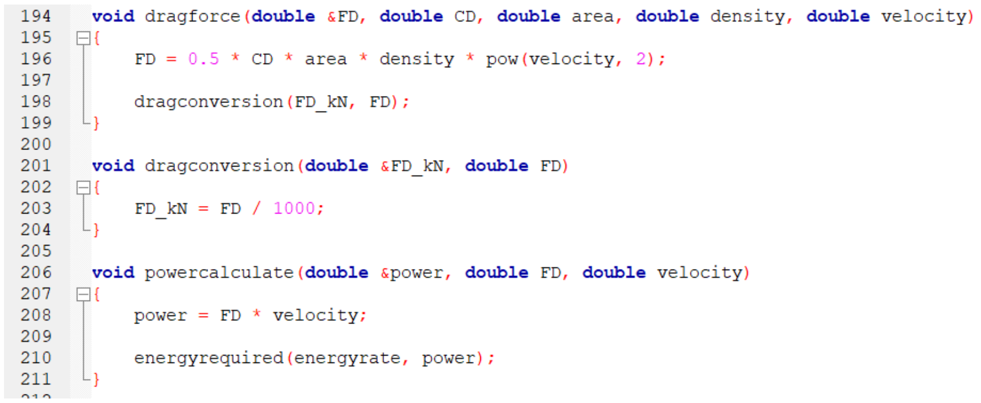
```
void safevmincalculate(double &safevmin, double &safevminkmh, double vmin)
{
    safevmin = 1.2 * vmin;
    safevminkmh = safevmin * 3.6;
}

void liftcoefficient (double &CL, double weight, double density, double velocity, double area)
{
    CL = weight /(0.5*density*pow(velocity,2)*area);

    dragcoefficient(CD, CL);
}
```

safevmincalculate() function is use for calculating safe minimum velocity to avoid stall region by multiplying with 1.2 and convert the safe velocity from m/s to km/h. While in **liftcoefficient() function,** new lift coefficient when an aircraft is crusing steadily at a constant altitude is calculated and a new function is call which is **dragcoefficient().**



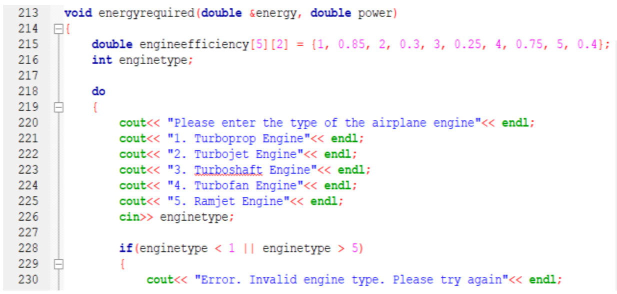
```
void dragcoefficient(double &CD, double CL)
{
    double CD_range[6][2] = {0, 0.01, 1.5, 0.06, 1.6, 0.07, 2.6, 0.12, 2.7, 0.13, 3.5, 0.3};

    for(int i = 0; i < 6; i++)
    {
        if(CL >= CD_range[i][0] && CL <= CD_range[i+1][0])
        {
            CD = (CD_range[i][1] + CD_range[i+1][1]) / 2;
        }
        else if(CL > 3.5)
        {
            CD = CD_range[i][1];
        }
    }
}
```

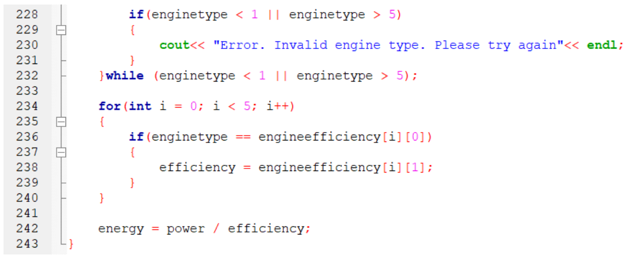
The use of **dragcoefficient()** function is for finding drag coefficient referring to new lift coefficient. Hence, we also use array to define drag coefficient at given lift coefficient.



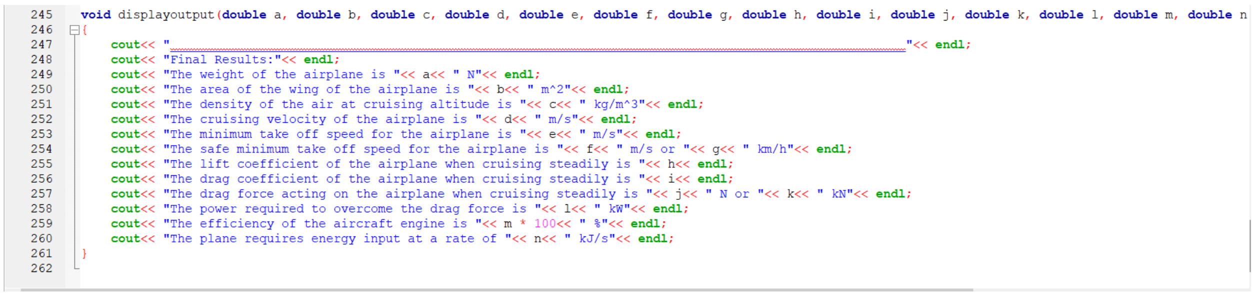
```
void dragforce(double &FD, double CD, double area, double density, double velocity)
{
    FD = 0.5 * CD * area * density * pow(velocity, 2);

    dragconversion(FD_kN, FD);
}

void dragconversion(double &FD_kN, double FD)
{
    FD_kN = FD / 1000;
}

void powercalculate(double &power, double FD, double velocity)
{
    power = FD * velocity;

    energyrequired(energyrate, power);
}
```

Next, the program will call **dragforce()** function to find the drag force corresponding to the cruising lift coefficient, and call new function **dragconversion().** In this function, the purpose is to convert drag force from newton to Kilonewton. In the other hand, the purpose of **powercalculate()** function is to find power required to overcome the drag force and call the next function, **energyrequired().**


```cpp
void energyrequired(double &energy, double power)
{
    double engineefficiency[5][2] = {1, 0.85, 2, 0.3, 3, 0.25, 4, 0.75, 5, 0.4};
    int enginetype;

    do
    {
        cout<< "Please enter the type of the airplane engine"<< endl;
        cout<< "1. Turboprop Engine"<< endl;
        cout<< "2. Turbojet Engine"<< endl;
        cout<< "3. Turboshaft Engine"<< endl;
        cout<< "4. Turbofan Engine"<< endl;
        cout<< "5. Ramjet Engine"<< endl;
        cin>> enginetype;

        if(enginetype < 1 || enginetype > 5)
        {
            cout<< "Error. Invalid engine type. Please try again"<< endl;
```


```cpp
        if(enginetype < 1 || enginetype > 5)
        {
            cout<< "Error. Invalid engine type. Please try again"<< endl;
        }
    }while (enginetype < 1 || enginetype > 5);

    for(int i = 0; i < 5; i++)
    {
        if(enginetype == engineefficiency[i][0])
        {
            efficiency = engineefficiency[i][1];
        }
    }

    energy = power / efficiency;
}
```

In **energyrequired()** function, array is used to determine efficiency of the engine at given type of engine. **Do… while** statement is used so that user can input a valid engine type (From 1 to 5). After that, energy is calculated using power that calculated from **powercalculate()** divide by efficiency.


```cpp
void displayoutput(double a, double b, double c, double d, double e, double f, double g, double h, double i, double j, double k, double l, double m, double n
{
    cout<< "____________________________________________________________________________________________________"<< endl;
    cout<< "Final Results:"<< endl;
    cout<< "The weight of the airplane is "<< a<< " N"<< endl;
    cout<< "The area of the wing of the airplane is "<< b<< " m^2"<< endl;
    cout<< "The density of the air at cruising altitude is "<< c<< " kg/m^3"<< endl;
    cout<< "The cruising velocity of the airplane is "<< d<< " m/s"<< endl;
    cout<< "The minimum take off speed for the airplane is "<< e<< " m/s"<< endl;
    cout<< "The safe minimum take off speed for the airplane is "<< f<< " m/s or "<< g<< " km/h"<< endl;
    cout<< "The lift coefficient of the airplane when cruising steadily is "<< h<< endl;
    cout<< "The drag coefficient of the airplane when cruising steadily is "<< i<< endl;
    cout<< "The drag force acting on the airplane when cruising steadily is "<< j<< " N or "<< k<< " kN"<< endl;
    cout<< "The power required to overcome the drag force is "<< l<< " kW"<< endl;
    cout<< "The efficiency of the aircraft engine is "<< m * 100<< " %"<< endl;
    cout<< "The plane requires energy input at a rate of "<< n<< " kJ/s"<< endl;
}
```

Lastly, last function, **displayoutput()** is used to output or display the results that are calculated from the functions. As mentioned before, the program will continue to run again if user input "Y", otherwise the program will be closed.

3.2 Flow chart

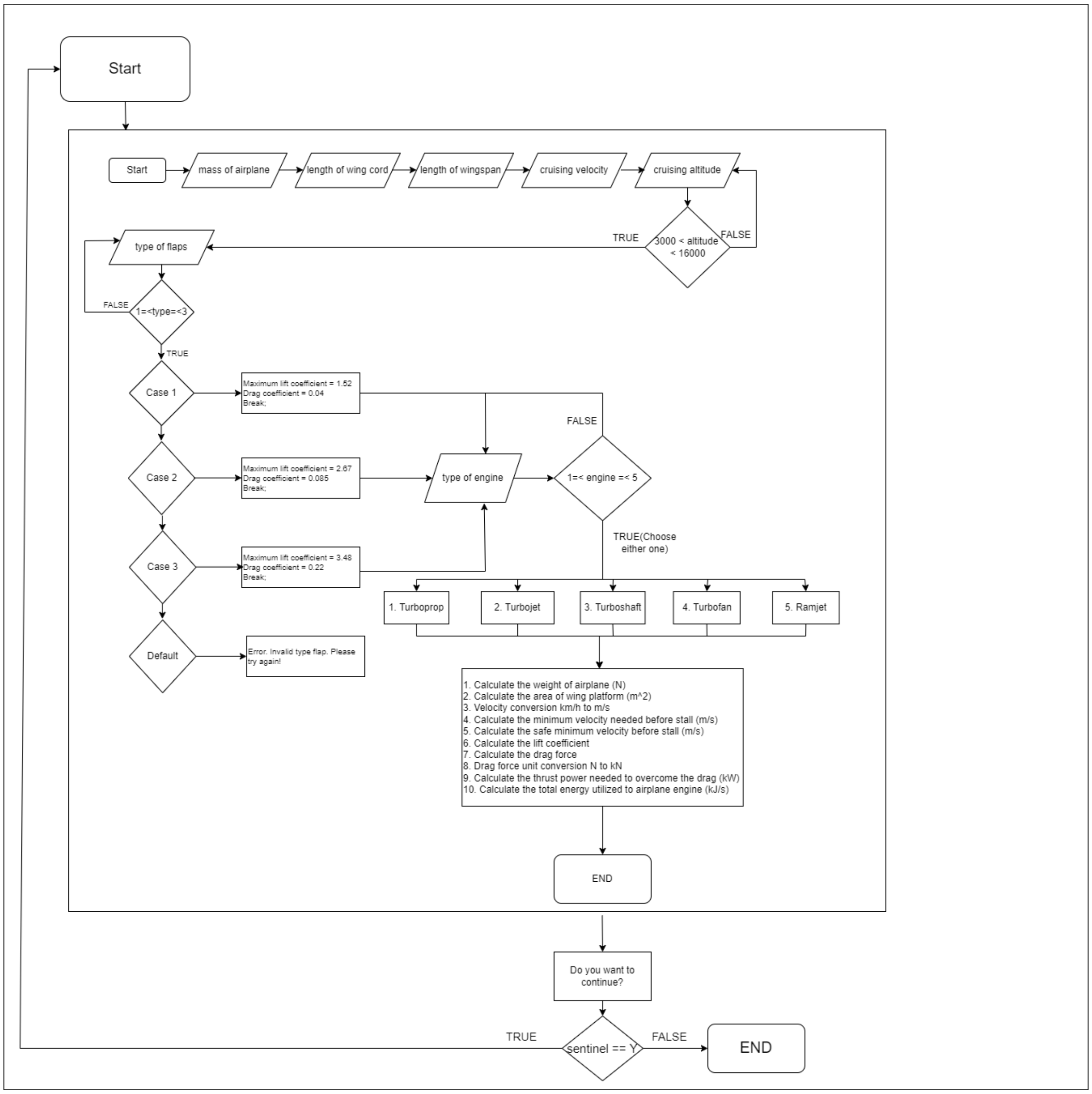

Start
Start
mass of airplane
length of wing cord
length of wingspan
cruising velocity
cruising altitude
3000 < altitude < 16000
TRUE
FALSE
type of flaps
1=<type=<3
FALSE
TRUE
Case 1
Maximum lift coefficient = 1.52
Drag coefficient = 0.04
Break;
Case 2
Maximum lift coefficient = 2.67
Drag coefficient = 0.085
Break;
Case 3
Maximum lift coefficient = 3.48
Drag coefficient = 0.22
Break;
Default
Error. Invalid type flap. Please try again!
type of engine
1=< engine =< 5
FALSE
TRUE(Choose either one)
1. Turboprop
2. Turbojet
3. Turboshaft
4. Turbofan
5. Ramjet
1. Calculate the weight of airplane (N)
2. Calculate the area of wing platform (m^2)
3. Velocity conversion km/h to m/s
4. Calculate the minimum velocity needed before stall (m/s)
5. Calculate the safe minimum velocity before stall (m/s)
6. Calculate the lift coefficient
7. Calculate the drag force
8. Drag force unit conversion N to kN
9. Calculate the thrust power needed to overcome the drag (kW)
10. Calculate the total energy utilized to airplane engine (kJ/s)
END
Do you want to continue?
sentinel == Y
TRUE
FALSE
END

Figure 3 - Representation of the program by flow chart

4.0 DISCUSSION

4.1 Input

Table 1: The Input Parameters

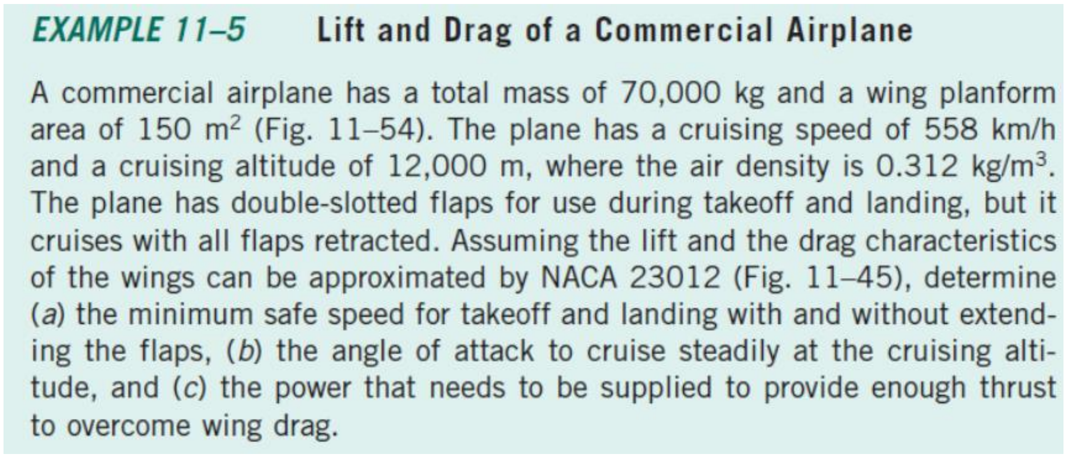
Mass of Airplane (kg)	Chord Length (m)	Wingspan (m)	Cruising Velocity (km/h)	Cruising Altitude (m)	Type of Flap	Type of Engine
70000	5	30	558	12000	2. Single Flap	2. Turbojet Engine
45000	10	60	550	10000	3. Double Flap	4. Turbofan Engine
10000	2	10	260	3000	1. No Flap	1. Turboprop Engine
85000	9	50	600	16000	2. Single Flap	2. Turbojet Engine
31200	3.6	12.247	256	8500	3. Double Flap	3. Turboshaft Engine
25678	3.12	25.69	320	6300	2. Single Flap	2. Turbojet Engine
56500	6	60	450	11650	3. Double Flap	4. Turbofan Engine
90000	15	65	600	16000	3. Double Flap	5. Ramjet Engine
66000	8.5	45.6	560	12500	2. Single Flap	3. Turboshaft Engine
30000	5	55	420	9030	3. Double Flap	1. Turboprop Engine

4.2 Output

Table 2: Output Values from Program

Weight (N)	Planform Area (m^2)	Air Density (kg/m^3)	Cruising Velocity (m/s)	Minimum Take off Velocity (m/s)	Safe Minimum Take off Velocity (km/h)	Lift Coefficient	Drag Coefficient	Drag Force (kN)	Power Required (kW)	Engine Efficiency (%)	Energy Input Rate (kJ/s)
686700	150	0.312	155	46.8244	202.282	1.22148	0.035	19.6765	3049.85	30	10166.2
441450	600	0.414	152.778	18.7715	81.0931	0.152279	0.035	101.464	15501.4	75	20668.5
98100	20	0.909	41.6667	73.3368	316.815	6.21624	0.3	4.73438	197.266	85	232.077
833850	450	0.166	166.667	34.01	146.923	0.803711	0.035	36.3125	6052.08	30	20173.6
306072	44.0892	0.526	71.1111	57.6608	249.095	5.21988	0.3	17.5907	1250.9	25	5003.59
251901	80.1528	0.66	88.8889	44.2919	191.341	1.20532	0.035	7.31469	650.194	30	2167.31
554265	360	0.35	125	27.1545	117.308	0.563063	0.035	34.4531	4306.64	75	5742.19
882900	975	0.166	166.667	20.8252	89.9647	0.392764	0.035	78.6771	13112.8	40	32782.1
647460	387.6	0.312	155.556	32.2911	139.498	0.442521	0.035	51.2091	7965.87	25	31863.5
294300	275	0.467	166.667	22.6393	97.8019	0.336726	0.035	30.5901	3568.85	85	4198.64

4.3 Manual Calculations



EXAMPLE 11–5 Lift and Drag of a Commercial Airplane

A commercial airplane has a total mass of 70,000 kg and a wing planform area of 150 m^2 (Fig. 11–54). The plane has a cruising speed of 558 km/h and a cruising altitude of 12,000 m, where the air density is 0.312 kg/m^3. The plane has double-slotted flaps for use during takeoff and landing, but it cruises with all flaps retracted. Assuming the lift and the drag characteristics of the wings can be approximated by NACA 23012 (Fig. 11–45), determine (*a*) the minimum safe speed for takeoff and landing with and without extending the flaps, (*b*) the angle of attack to cruise steadily at the cruising altitude, and (*c*) the power that needs to be supplied to provide enough thrust to overcome wing drag.

Figure 3: Example Problem

Given Parameters:
Mass = 70000 kg
Wing Planform Area = 150 m^2
Cruising Speed = 558 km/h
Cruising Altitude = 12000 m
Air Density at Cruising Altitude = 0.312 kg/m^3
Flap Type during Takeoff = Double Slotted Flaps
Engine Propulsion Efficiency = 30%

Calculations:

$$W = mg = (70000)(9.81) = 686700N$$

$$V = (558\ km/h)\left(\frac{1\, m/s}{3.6\, km/h}\right) = 155\, m/s$$

$$V_{min} = \sqrt{\frac{2W}{\rho C_{L,max} A}} = \sqrt{\frac{2(686700)}{(1.2)(3.48)(150)}} = 46.8\ m/s$$

$$V_{min,safe} = 1.2V_{min} = 1.2(46.8) = 56.2\, m/s = 202\, km/h$$

$$C_L = \frac{F_L}{\frac{1}{2}\rho V^2 A} = \frac{686700}{\frac{1}{2}(0.312)(155^2)(150)} = 1.22$$

Corresponding C_D = 0.03

$$F_D = C_D A \frac{\rho V^2}{2} = (0.03)(150)\frac{(0.312)(155^2)}{2} = \frac{16900N}{1000} = 16.9\ kN$$

$$P = F_D V = (16.9)(155) = 2620\ kW$$

$$E = \frac{P}{0.3} = \frac{2620}{0.3} = 8730\, kJ/s$$

Where,
W = Weight of Airplane
V = Cruising Speed of Airplane in m/s
V_{min} = Minimum Velocity for Takeoff
$V_{min,\ safe}$ = Safe Minimum Velocity for Takeoff
C_L = Lift coefficient of Airplane at Cruising Altitude
C_D = Drag coefficient of Airplane at Cruising Altitude
F_D = Drag force acting on the wings of the Airplane
P = Power Required to Overcome Drag Force
E = Energy Input Rate to Produce Required Power

4.4 Comparison between Manual Calculations and Output of Program

Table 3: Comparison of Output from Manual Calculations and Output of Program

Weight (N)	Planform Area (m^2)	Air Density (kg/m^3)	Cruising Velocity (m/s)	Minimum Take off Velocity (m/s)	Safe Minimum Take off Velocity (km/h)	Lift Coefficient	Drag Coefficient	Drag Force (kN)	Power Required (kW)	Engine Efficiency (%)	Energy Input Rate (kJ/s)
686700	150	0.312	155	46.8244	202.282	1.22148	0.035	19.6765	3049.85	30	10166.2
686700	150	0.312	155	46.8	202	1.22	0.03	16.9	2620	30	8730

The problem above was simulated with the coding and the output was shown in Table 3 with the values obtained from the coding shown in the first row and the results of manual calculations shown in the second row. It is seen that the values obtained for the weight, planform area, air density and cruising velocities, and engine efficiency are identical between the manual calculation and coding output, which suggests that the program works as intended and gives accurate results. However, for the minimum takeoff velocity, safe minimum takeoff velocity, and lift coefficients, there are some differences of 0.05%, 0.14% and 0.12% respectively. These small differences occur due to number of significant figures which are displayed by program and does not mean that the output of the program produced inaccurate outputs. For the drag coefficients, drag force, power required and energy input rate, the error is higher at 16.67%, 16.43%, 16.41%, and 16.45% respectively. This error is due to a feature of the program itself in which it calculates the drag coefficient as the average of the corresponding drag coefficients from the minimum and maximum lift coefficients of each of the flap types. Thus, the drag coefficient given by the program is not as accurate as only the average is used whereas it should cover a wider range of values. As a result, the drag force, power required, and energy input rate are also affected as these values are closely linked together. Even though the errors produced by these values are quite large, it can be said that the program gives precise calculations given that the errors are similar. Thus, it can be said that the program performs as intended and is able to output the proper answers.

5.0 CONCLUSION

5.1 Limitation

The limitation that occurs for this simulator is that the drag and lift force produce only at the wing of the plane. In reality, the drag and lift force occur on every part of the plane as all plane parts have surface area. Next, the calculation of the area of wings is assumed to be two-dimensional but the wing of airplanes is three-dimensional where the volume of the wing also plays an important role in determining the drag and lift force of the plane. Moreover, the simulator only is applicable for NACA23012 which means that it is not applicable for other types of wings. Other than that, the value for drag coefficient is taken as an average between two corresponding values as the drag coefficient cannot be quantified mathematically.

5.2 Improvement

The improvement that can be made for this simulator is to include the other parts of the airplane such as the upper and lower body of the airplane and the tail of the airplane to get more accurate drag and lift force. Next, when calculating the area of the wing, the curve shape of the wing needs to be taken as considered as the surface area is not just the length and the width. Moreover, more types of wings can be added to increase the usefulness of this simulator as it can help calculate the drag and lift force for other types of wings. Other than that, taking the exact value of the angle of attack which can help to find the exact value for the lift coefficient and thus, the drag coefficient.

6.0 REFERENCE

Yunus A. Cengel & John M. Cimbala (2020). Fluid Mechanics (SI unit): Fundamental and Application, USA, Mc Graw Hill.

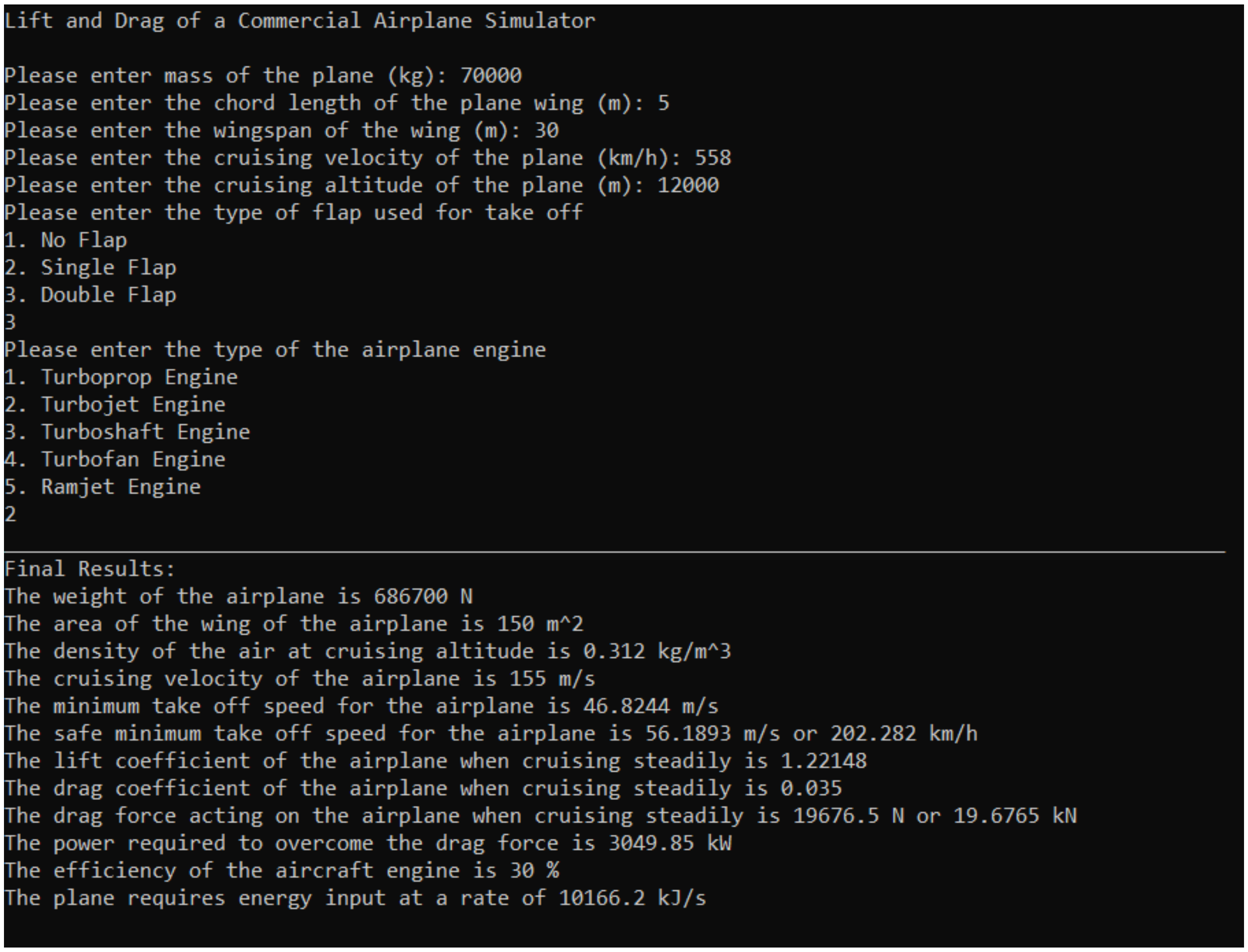
Thomas Forenz (2016). Basic Aerodynamics: Aviation Maintenance Technician Certification Series, USA, Aircraft Technical Book Company.

Roger Peterson & Omar Khan (2020). Aircraft Aerodynamic Structures and System: Aviation Maintenance Technician Certification Series, USA, Aircraft Technical Book Company.

Talay, Theodore A. *Introduction to the Aerodynamics of Flight.* SP-367, Scientific and Technical Information Office, National Aeronautics and Space Administration, Washington, D.C. 1975. Available at http://history.nasa.gov/SP-367/cover367.htm

National Academies of Sciences, Engineering, and Medicine. 2016. Commercial Aircraft Propulsion and Energy Systems Research: Reducing Global Carbon Emissions. Washington, DC: The National Academies Press. https://doi.org/10.17226/23490.

7.0 APPENDIX



```
Lift and Drag of a Commercial Airplane Simulator

Please enter mass of the plane (kg): 70000
Please enter the chord length of the plane wing (m): 5
Please enter the wingspan of the wing (m): 30
Please enter the cruising velocity of the plane (km/h): 558
Please enter the cruising altitude of the plane (m): 12000
Please enter the type of flap used for take off
1. No Flap
2. Single Flap
3. Double Flap
3
Please enter the type of the airplane engine
1. Turboprop Engine
2. Turbojet Engine
3. Turboshaft Engine
4. Turbofan Engine
5. Ramjet Engine
2

Final Results:
The weight of the airplane is 686700 N
The area of the wing of the airplane is 150 m^2
The density of the air at cruising altitude is 0.312 kg/m^3
The cruising velocity of the airplane is 155 m/s
The minimum take off speed for the airplane is 46.8244 m/s
The safe minimum take off speed for the airplane is 56.1893 m/s or 202.282 km/h
The lift coefficient of the airplane when cruising steadily is 1.22148
The drag coefficient of the airplane when cruising steadily is 0.035
The drag force acting on the airplane when cruising steadily is 19676.5 N or 19.6765 kN
The power required to overcome the drag force is 3049.85 kW
The efficiency of the aircraft engine is 30 %
The plane requires energy input at a rate of 10166.2 kJ/s
```

Sample Output

The whole coding program

```
#include <iostream>

#include <iomanip>

#include <cmath>


using namespace std;


double safevmin, safevminkmh, FD, FD_kN, CD, energyrate, efficiency;


void weightcalculate (double &, double);

void areacalculate(double &, double, double);
void cruisingspeedconversion(double &);
void densityaltitude(double &);
void vmincalculate(double &, double, double);
void safevmincalculate(double &, double &, double);
void liftcoefficient (double &, double, double, double, double);
void dragcoefficient (double &, double);
void dragforce(double &, double, double, double, double);
void dragconversion(double &, double);
void powercalculate(double &, double, double);
void energyrequired(double &, double);
void displayoutput(double, double, double, double, double, double, double, double, double, double,
double, double, double, double);


int main()
{
    double massplane, chordlength, wingspan, cruisevelocity, weight, area, density, vmin, CL, power;
    char sentinel = 'Y';

    do
    {
        cout<< "Lift and Drag of a Commercial Airplane Simulator\n\n";

        cout<< "Please enter mass of the plane (kg): ";
        cin>> massplane;

        cout<< "Please enter the chord length of the plane wing (m): ";
        cin>> chordlength;

        cout<< "Please enter the wingspan of the wing (m): ";
        cin>> wingspan;
        cout<< "Please enter the cruising velocity of the plane (km/h): ";
        cin>> cruisevelocity;



        weightcalculate(weight, massplane);
        areacalculate(area, chordlength, wingspan);
        cruisingspeedconversion(cruisevelocity);
        densityaltitude(density);
        vmincalculate(vmin, weight, area);
        liftcoefficient(CL, weight, density, cruisevelocity, area);
        dragforce(FD, CD, area, density, cruisevelocity);
        powercalculate(power, FD_kN, cruisevelocity);
        displayoutput(weight, area, density, cruisevelocity, vmin, safevmin, safevminkmh, CL, CD, FD,
FD_kN, power, efficiency, energyrate);



        cout<< "\n\nDo you wish to continue?\nPress Y for yes and N for no"<< endl;
        cin>> sentinel;

        if(sentinel == 'Y')
        {
            cout<< "\n\n";
            cout<<
"____________________________________________________________________________________
_____________________"<< endl;
        }

    } while (sentinel == 'Y');

    return 0;
```

```
}

void weightcalculate (double &weight, double mass)
{
    double g = 9.81;
    weight = mass * g;
}

void areacalculate (double &area, double chord, double span)
{
    area = chord * span;
}

void cruisingspeedconversion(double &cruisevelocity)
{
    cruisevelocity = cruisevelocity / 3.6;
}

void densityaltitude(double &density)
{
    double cruisealtitude;
    double altitude[12][2] =
{3000,0.909,4000,0.891,5000,0.872,6000,0.660,7000,0.590,8000,0.526,9000,0.467,10000,0.414,11000,0.
350,12000,0.312,14000,0.228,16000,0.166};

    do
    {
        cout<< "Please enter the cruising altitude of the plane (m): ";
        cin>> cruisealtitude;

        if(cruisealtitude > 16000)
        {
            cout<< "Altitude is too high, please reenter the altitude."<< endl;
        }
        else if(cruisealtitude < 3000)
        {
            cout<< "Altitude is too low, please reenter the altitude."<< endl;
        }
        else if(cruisealtitude >= 3000 && cruisealtitude <= 16000)
        {
            for(int i = 0; i < 12; i++)
            {
                if(cruisealtitude >= altitude[i][0] && cruisealtitude < altitude[i+1][0])
                {
                    density = altitude[i][1];
                }
                else if(cruisealtitude == 16000)
                {
                    density = altitude[i][1];
                }
            }
        }
    }while (cruisealtitude > 16000 || cruisealtitude < 3000);
}

void vmincalculate(double &vmin, double weight, double area)
{
    int flaptype;
    double CL, grounddensity = 1.2;

    do
```

```
{
    cout<< "Please enter the type of flap used for take off"<< endl;
    cout<< "1. No Flap"<< endl;
    cout<< "2. Single Flap"<< endl;
    cout<< "3. Double Flap"<< endl;
    cin>> flaptype;

    switch(flaptype)
    {
        case 1:
            {
                CL = 1.52;
                break;
            }
        case 2:
            {
                CL = 2.67;
                break;
            }
        case 3:
            {
                CL = 3.48;
                break;
            }
        default:
            {
                cout<< "Error. Invalid flap type. Please try again."<< endl;
            }
    }
    }while (flaptype > 3 || flaptype < 1);

    vmin = sqrt((2*weight)/(grounddensity*CL*area));

    safevmincalculate(safevmin, safevminkmh, vmin);
}

void safevmincalculate(double &safevmin, double &safevminkmh, double vmin)
{
    safevmin = 1.2 * vmin;
    safevminkmh = safevmin * 3.6;
}

void liftcoefficient (double &CL, double weight, double density, double velocity, double area)
{
    CL = weight /(0.5*density*pow(velocity,2)*area);

    dragcoefficient(CD, CL);
}

void dragcoefficient(double &CD, double CL)
{
    double CD_range[6][2] = {0, 0.01, 1.5, 0.06, 1.6, 0.07, 2.6, 0.12, 2.7, 0.13, 3.5, 0.3};

    for(int i = 0; i < 6; i++)
    {
        if(CL >= CD_range[i][0] && CL <= CD_range[i+1][0])
        {
            CD = (CD_range[i][1] + CD_range[i+1][1]) / 2;
        }
        else if(CL > 3.5)
        {
            CD = CD_range[i][1];
        }
    }
}

void dragforce(double &FD, double CD, double area, double density, double velocity)
{
    FD = 0.5 * CD * area * density * pow(velocity, 2);

    dragconversion(FD_kN, FD);
}

void dragconversion(double &FD_kN, double FD)
{
    FD_kN = FD / 1000;
}

void powercalculate(double &power, double FD, double velocity)
{
    power = FD * velocity;

    energyrequired(energyrate, power);
}

void energyrequired(double &energy, double power)
{
    double engineefficiency[5][2] = {1, 0.85, 2, 0.3, 3, 0.25, 4, 0.75, 5, 0.4};
    int enginetype;
    do
    {
        cout<< "Please enter the type of the airplane engine"<< endl;
        cout<< "1. Turboprop Engine"<< endl;
        cout<< "2. Turbojet Engine"<< endl;
        cout<< "3. Turboshaft Engine"<< endl;
        cout<< "4. Turbofan Engine"<< endl;
        cout<< "5. Ramjet Engine"<< endl;
        cin>> enginetype;

        if(enginetype < 1 || enginetype > 5)
        {
            cout<< "Error. Invalid engine type. Please try again"<< endl;
        }
    }while (enginetype < 1 || enginetype > 5);

    for(int i = 0; i < 5; i++)
    {
        if(enginetype == engineefficiency[i][0])
        {
            efficiency = engineefficiency[i][1];
        }
    }

    energy = power / efficiency;
}

void displayoutput(double a, double b, double c, double d, double e, double f, double g, double h, double
i, double j, double k, double l, double m, double n)
```

```
{
    cout<<
"_____________________________________________________________________________________________________"<< endl;
    cout<< "Final Results:"<< endl;
    cout<< "The weight of the airplane is "<< a<< " N"<< endl;
    cout<< "The area of the wing of the airplane is "<< b<< " m^2"<< endl;
    cout<< "The density of the air at cruising altitude is "<< c<< " kg/m^3"<< endl;
    cout<< "The cruising velocity of the airplane is "<< d<< " m/s"<< endl;
    cout<< "The minimum take off speed for the airplane is "<< e<< " m/s"<< endl;
    cout<< "The safe minimum take off speed for the airplane is "<< f<< " m/s or "<< g<< " km/h"<< endl;
    cout<< "The lift coefficient of the airplane when cruising steadily is "<< h<< endl;
    cout<< "The drag coefficient of the airplane when cruising steadily is "<< i<< endl;
    cout<< "The drag force acting on the airplane when cruising steadily is "<< j<< " N or "<< k<< "
kN"<< endl;
    cout<< "The power required to overcome the drag force is "<< l<< " kW"<< endl;
    cout<< "The efficiency of the aircraft engine is "<< m * 100<< " %"<< endl;
    cout<< "The plane requires energy input at a rate of "<< n<< " kJ/s"<< endl;
}
```