

William Wilson

Iowa State University, AERE Freshman

Project 1

AERE 161

Due: 10/18/24, 6:00 PM

Table of Contents

Problem 1

Problem Statement page no. 3

Theory page no. 4

Solution page no. 5

Discussion page no. 6

Problem 2

Problem Statement page no. 7

Theory page no. 8

Solution page no. 9

Discussion page no. 10

Appendix

Function 1, Script 1..... page no. 11

Script 1..... page no. 11-13

Function 1, Script 2..... page no. 13

Script 2..... page no. 13-14

Problem 1

Problem statement

For problem 1 of Project 1 I was tasked with creating a program that is able to perform a weight and balance on a 1965 Piper Cherokee PA-28-180 aircraft. I was given a set of information pertaining to the specific craft and had to write inputs into my program to gather information like pilot weight, passenger weight, and amount of fuel. Using a mix of the constants provided and variables obtained from the user I was able to write a program that reliably calculates the CoG of the aircraft and can determine if the Piper Cherokee is safe to fly. (For educational purposes) I then needed to print out the ramp weight, loaded moment, and if it is in correct W&B.

Theory

$$FuelWLbs = FuelG \cdot 6$$

Calculates the fuel weight in pounds from fuel given in US gallons.

FuelG: Fuel in US Gallons (A user input)

FuelWLbs: Fuel weight in Lbs

$$RampW = FuelWLbs + EmptyW + PW + CPW + P1 + P2$$

Calculates the total weight of user entered variables and given weight variables by adding all the imputed masses together with our constants.

RampW: Total weight of user imputed variables and mass loaded onto the craft

EmptyW: Weight of the aircraft in Lbs. without any fuel, passengers, or pilots

PW: Weight of the pilot in Lbs.

CPW: Weight of the co-pilot in Lbs.

P1: Weight of Passenger 1 in Lbs.

P2: Weight of Passenger 2 in Lbs.

$$Mfront = (PW + CPW) \cdot FrontStMA$$

Calculates moment for the front seats by adding the pilots weight together and multiplying it by the front Moment Arm.

Mfront: Moment for the front of the craft

FrontStMA: The front seat moment arm

$$Mrear = (P1 + P2) \cdot RearStMA$$

Calculates moment for the front seats by adding the passengers weight together and multiplying it by the front rear Moment Arm.

Mrear: Moment for the rear of the craft

RearStMA: The rear seat moment arm

$$Mfuel = FuelWLbs \cdot FuelTankMA$$

Calculates moment for the fuel by multiplying the fuel weight by the Fuel Tank Moment Arm.

Mfuel: Moment for the fuel

FuelTankMA: The fuel Tank's moment arm

$$Mempty = EmptyW \cdot CoG$$

Calculates the moment for the craft when its empty and before it had variable weight

Mempty: Moment for the empty craft

CoG: A constant and the center of gravity of the craft before additional weight

$$TotalM = Mfront + Mrear + Mfuel + Mempty$$

Calculates Total Moment of the aircraft by adding all the moments together.

TotalM: The total moment of the craft

$$NewCoG = \frac{TotalM}{RampW}$$

Calculates the New Center of Gravity by dividing the total moment by the Ramp weight.
 New CoG: The newly calculated Center of Gravity for the craft

Solution

When I first approached the problem, I came in with the mindset of creating the extra loop allowing to user to exit out of the program. Because of this, I went with a double while loop style that allowed the code to repeat itself both to rerun the entire code and to rerun the weight input section of code in case the user entered a total weight that was too large.

Values I entered the program:

Fuel: 49 Gallons

Pilot Weight: 150 Lbs

Co-Pilot Weight: 120 Lbs

Passenger 1 Weight: 150 Lbs

Passenger 2 Weight: 0 Lbs

```
How much fuel is onboard? (in US Gallons) --> 49
What is the weight of the pilot (in lbs)? --> 150
What is the weight of the co-pilot (in lbs)? --> 120
What is the weight of passenger one? (if no passenger enter 0) --> 150
What is the weight of passenger two? (if no passenger enter 0) --> 0
```

Output:

Your new Center of Gravity is 89.29 inches

The aircraft is within the weight and balance. It should be able to have a stable takeoff and fly normally!

Enter 1 if you would like to close out of the program

and any other number to do it again. -->1

Thanks for using the W&B aviation calculator, see you next time!

```
Your new Center of Gravity is 89.29 inches
The aircraft is within the weight and balance. It should be able to have a stable takeoff and fly normally!

Enter 1 if you would like to close out of the program
and any other number to do it again. -->1

Thanks for using the W&B aviation calculator, see you next time!
```


Discussion

My code determined that the aircraft was able to safely fly with its new CoG of 89.29 inches. This means that when the craft is fully loaded with all passengers, pilots, and fuel we will not have to worry about the plane tipping over or being hard to control because of weight imbalance.

Weight and balance are very important in aviation as if your plane is unbalanced it can cause extreme turbulence in the air and could even make your play so unstable that you crash. By running the numbers ahead of time, pilots can confirm that their plane will fly in a stable state based off the weight placement. Incorrect W&B can also cause problems on takeoff and even tip planes over on the runway in extreme cases.

For this problem the biggest challenge I faced happened when my laptop broke down in the middle of the project. I lost the majority of my code for problem one and wasn't able to get it back. I was able to overcome this challenge by making use of campus resources and computer labs and since I had already written the code before it went pretty fast the second time.

I was having a problem with my calculation that was causing my final number to be off. To fix this I carefully looked over each variable and constant to ensure my numbers were correct and then after confirming that the numbers were not the problem I reviewed each equation. I found that I had forgotten to add the empty moment to the total, and this was causing my discrepancy.

Problem 2

Problem statement

For problem two I was tasked with creating a program that draws weather information from a given API and calculates the Density Altitude considering the local temperature, humidity, pressure, and field elevation.

Theory

$$Dewpt = tempC - \left(\frac{100 - humidity}{5} \right)$$

%Calculates dewpoint by subtracting 100-our humidity value over 5 by the temp in C

Dew Pt: The dew point of Ames Airport

TempC: Our temperature in Celsius

Humidity: The humidity at Ames Airport

$$Vaporpressure = 6.11 \cdot 10^{\left(\frac{7.5 \cdot Dewpt}{237.7 + Dewpt} \right)}$$

%Calculates Vapor Pressure in millibar by multiplying 6.11 with 10 to the power of (7.5 times the dewpt over 237.7+ the dewpt)

Vaporpressure: Vapor pressure calculated at the Ames airport

$$Virtualtemp = \frac{tempK}{1 - \left(\frac{Vaporpressure}{pressure} \right) \cdot (1 - 0.622)}$$

%Calculated the virtual temperature in Kelvin by taking our temperature in Kelvin and dividing it by 1 - our vappor pressure over pressure multiplied by (1 minus 0.622)

Virtualtemp: Temperature adjusted, considering our humidity, pressure, dewpt, and vapporpressure.

Pressure: the pressure we received from open weather maps API

TempK: Our calculated temperature in Kelvin

$$VirtualtempR = \left(\frac{9}{5} \right) \cdot (Virtualtemp - 273.15) + 32 + 459.69$$

%Takes the Virtual temp and converts it to Rankine through the Kelvin to Rankine conversion formula

VirtualtempR: The same thing as the virtual temp but in Rankine units

$$PressureinHG = (pressure \cdot 0.02953)$$

%Calculates the pressure from Milibars to inches of Mercury using the conversion formula

PressureinHG: The pressure of air at Ames airport in inches of Mercury

$$DensityAlt = fieldelv + 1452366 \cdot \left(1 - \left(\frac{17.326 \cdot PressureinHG}{VirtualTempR} \right)^{0.235} \right)$$

%Calculates the density altitude, considering the field elevation of Ames airport

DensityAlt: Current altitude of Ames airport relative to standard conditions

Fieldelv: The elevation that Ames's airport sits at

Solution


For this problem, I approached it in a normal linear way. I first obtained information using the code provided in the project outlined and then created a set of variables that could be used in my equations. I outlined and created all my variables before I wrote any equations. After my variables were defined, I wrote each equation to solve the air density.

No variables entered (All data was collected from information given or openweathermap API)

Output:

The density Altitude of Ames Iowa is approximately 918.4678 ft.>>

Command Window

 The density Altitude of Ames Iowa is approximately 918.4678 ft.>>

Discussion

My code resulted in a Density Altitude of Ames Iowa being 918.4678 ft. This is a value of the corrected density altitude for a plane taking into account the temperature, humidity, and pressure. This tells the plane what altitude it should think it should be flying at based on the air density.

Density altitude is important in aviation because when temperature, humidity, and pressure change the air density also changes resulting in your craft producing different amounts of aerodynamic forces with the same amount of thrust. For example, when the air density is lower you will generate less lift because fewer air molecules will be under the plane's wings pushing the craft upward.

For this problem, I ran into two challenges. One was the result of incorrect use of parenthesis and was fixed by carefully looking through my equations in code and correcting it. The second challenge came while I was trying to confirm my result and compare it with the correct value found online. I could not figure out why the values were not matching until I realized the online answers were not taking into account field elevation. Once this was realized I was able to confirm that my code was accurate.

Appendix

Function 1, Script 1:

```
function [fuelW, PW, CPW, P1, P2] = userweightvars %Creates a function with no inputs
and 5 outputs
%William Wilson, AERE 161, Project 1, Problem 1, Function 1
% Write a function that obtains and returns information on weight
% variables entered by the user. Ensure the weight meets certain criteria
% like fuel being in correct range.
x = 1;%Sets the while loop to active
while x == 1 %Creates a loop that will run until the x value is changed
    fuelW = input('How much fuel is onboard? (in US Gallons) --> '); %Asks the
    user for the fuel in gallons
    if fuelW > 50 %Checks if more than 50 gallons were entered
        disp('The entered fuel is greater then the plane can store. Please enter
        a value below 50 Gallons') %Displays that the valye entered was too much
    elseif fuelW <= 0 %Checks if enough fuel was entered to run the plane at all
        disp('The entered fuel is not enough to fly please enter a number
        greater than 0 Gallons') %Tells the user more fuel is needed
    else %Runs if the correct amount of fuel was entered
        x = 2; %Changed x value to exit loop
        PW = input('What is the weight of the pilot (in lbs)? --> '); %Asks for
        the pilot weight in lbs
        CPW = input('What is the weight of the co-pilot (in lbs)? --> '); %Asks
        for the co-pilot weight in lbs
        P1 = input('What is the weight of passenger one? (if no passenger enter
        0) --> '); %Asks for the passenger 1 weight in lbs
        P2 = input('What is the weight of passenger two? (if no passenger enter
        0) --> '); %Asks for the passenger 2 weight in lbs
    end%Ends the if statment
end%Ends the while loop
end%Ends the function
```

Script 1

```
clear, clc
%{
William Wilson, AERE 161, Project 1, Script 1
This project will involve using variables and performing simple arithmetic.
You will obtain values from the user using the "input" function and check
that the aircraft is within the weight and balance as specified from the man-
ufacture.
%}
x = 1;%Inititates the while loop
while x == 1
    clear, clc%Clears all vars so that new ones can be entered and window is clean
```

```

x = 1; %Sets the loop back to being active after vars have been cleared
%Constants
EmptyW = 1471; % Empty weight in lbs
CoG = 85.9; %Center of Gravity in inches
FrontStMA = 85.5; %Front Seat Moment Arm in inches
FuelTankMA = 95; %Fule Tanks Moment Arm in inches
RearStMA = 118.1; %Rear seat moment arm in inches
MaxRweight = 2400; %Max Ramp Weight in lbs
MaxFuel = 50; %Max fuel in Gallons
fprintf('Welcome to the W&B aviation calculator presented by Wilson
integrations!\n')%Welcome message to user
y = 1;%Initiates next while loop
    while y == 1 %While loop that runs the majority of code, repeates until
        user enters weight low enough to fly
        [fuelW, PW, CPW, P1, P2] = userweightvars; %Obatins user data for weight
        variables through a function
        FuelWlbs = fuelW * 6; %Converts the fuel that we had in Gallons into lbs
        of fuel
        RampW = FuelWlbs + EmptyW + PW + CPW + P1 + P2; %Calculates total weight
        of the aircraft once loaded
        if RampW <= MaxRweight %Checks if ramp weight is less then total weight
            allowed
                y = 2;%Changes y value ending the while loop
                disp('Nice!')%Applouds the user for setting weight right
                Mfront = (PW+CPW)*FrontStMA; %Calculates moment for the front
                seats
                Mrear = (P1+P2)*RearStMA; %Calculates moment for the rear seats
                Mfuel = FuelWlbs*FuelTankMA; %Calculates moment for the fuel
                tanks
                Mempty = EmptyW * CoG;%Calculates moment for the craft when empty
                TotalM = Mempty+Mfront+Mrear+Mfuel;%Adds up the total Moment from
                user imputs and calcuations
                NewCoG = TotalM/RampW; %Calculates new CoG
                fprintf('\nYour new Center of Gravity is %0.2f inches\n',
                NewCoG)%Prints CoG to user with a message
                if NewCoG >= 86.8 && NewCoG <= 95.8
                    fprintf("The aircraft is within the weight and balance. It
                    should be able to have a stable takeoff and fly
                    normally!\n")
                else
                    fprintf("The aircraft is not within the weight and
                    balance. It will be unstable if flown and should remain
                    grounded until its W&B is in the correct range. ")
                end
            else %Runs if the weight was too heavy and more then max allowed

```

```

        fprintf('\nTotal weight was too heavy, please enter again with
        new    weights. Total weight must be under 2400\n')%Tells the
        user to try  again
    end%Ends the if statment
end%Ends the while loop
loopbreak = input('\nEnter 1 if you would like to close out of the program
\nand any other number to do it again. -->');%checks if user wants to go again
if loopbreak == 1 %Checks if user said no
    x = 2;%Changes x value to break out of the loop
    fprintf('\n')%Creates a space so it is easier to read
    disp('Thanks for using the W&B aviation calculator, see you next
    time!')%Prints and exit message
end %Ends the for loop
end%Ends the while loop and program

```

Function 1, Script 2:

```

function [tempC] = tempKtoC(tempK)%Creates a function with one input and 1 output
%William Wilson, AERE 161, Project 1, Problem 2, Function 1
%    Write a function that converts given temperature in Celcius to
%    Temperature in Kelvin and return using variable tempC
tempC = tempK - 273.15; %Converts the temperature in Kelvin to the tempature in
Celius
end%Ends the function

```

Script 2

```

clear, clc
%{

William Wilson, AERE 161 Project 1, Question 2
Calculate the air density at the local Ames airport given whether specifics
from openweathermap API and field elevation given to us.

%}

key = '7cab1fc4f444883263bc48dd983e6018';
options = weboptions('ContentType','json');
url=['https://api.openweathermap.org/data/2.5/weather?q=', 'Ames', '&APPID=',key];
CurrentData = webread(url, options);
%All the above code retrieved the whether data from the openweathermap API

tempK = CurrentData.main.temp; %Current temp in Kelvin stored in tempK obtained from
openwhether maps API
pressure = CurrentData.main.pressure; %Current pressure in Milibar stored in pressure
obtained from openwhether maps API
humidity = CurrentData.main.humidity; %Current humidity stored in humidity obtained
from openwhether maps API

```

```

fieldelv = 955.6; %Field elevation of Ames Airport (KAMW) in ft

[tempC] = tempKtoC(tempK); %Calls a function that calculates the temp in C from the
temp given in K

Dewpt = tempC-((100-humidity)/5); %Calculates dewpoint temp in C
Vaporpressure = (6.11*(10^((7.5*Dewpt)/(237.7+Dewpt)))); %Calculates Vapor Pressure
in millibar

Virtualtemp = tempK/(1-((Vaporpressure/pressure)*(1-0.622))); %Calculated the virtual
temperature in Kelvin
VirtualtempR = (((9/5)*(Virtualtemp-273.15))+32)+459.69; %Takes the Virtual temp
and converts it to Rankine

PressureinHG = (pressure*0.02953); %Calculates the pressure from Milibars to inches
of Mercury

DensityAlt = fieldelv + (145366*(1-(((17.326*PressureinHG)/VirtualtempR)^0.235)));
%Calculates the density altitude

fprintf('The density Altitude of Ames Iowa is approximatley %0.4f ft.', DensityAlt)
%Prints to the user what teh density altitude is in Ames Iowa from the airport

```