AERE E 1600

# Project 1: Report Title

Chinedum Junior Daniel Echedom

Department of Aerospace Engineering

10th October, 2025

Table of Contents

## Problem 1

### Problem statement

To fly safely, an aircraft must operate within certain weight and balance limits. These weight and balance limits are a measure of how much load (passengers, fuel, payload, etc.) the aircraft is carrying, as well as the specific positions that these loads are placed in the aircraft.

In this project, I created a weight and balance calculator using MATLAB to determine if an aircraft, specifically the Piper Cherokee PA-28-180, is within the safe weight and balance limits when loaded with a certain amount of fuel (in gallons), as well as Pilot, Co-pilot, and 2 passengers at most, with their weights measured in pounds.

This calculator prompts the user to enter in the above stated information and then calculates the total weight of the airplane as well as it's Center of gravity using the calculated moments of each section of the airplane and consequently the total moment of the airplane. The calculator then determines if the airplane has exceeded the maximum ramp weight (maximum total weight) of 2,400 lbs to fly safely. The calculator also determines if the aircraft is within the acceptable balance limits (moment arm between 86.8 inches and 95.8 inches) to fly safely. If the airplane is within safe ranges of weight and balance, the user is given the go-ahead message that the airplane is safe to fly. However, if the airplane is not within the safe ranges of weight and/or balance, the user receives a message telling them that the aircraft is not safe to fly. Note that the calculator will only display that the aircraft is safe to fly if and only if it fulfills both the weight and balance requirements.

**Theory**

i. The amount of fuel entered (originally captured in US gallons), is converted into pounds. This is done by simply multiplying fuel by 6. This value is assigned to the variable name ‘fuelweight’.

$$f_w = f \times 6 \qquad 1$$

$f_w$ = fuelweight
f = amount of fuel in US gallons

ii. This is a sum of all the weights derived from the user’s input (fuelWeight, pilotWeight, coPilotWeight, passenger1Weight, passenger2Weight) and the weight of the aircraft when empty (Empty_weight).
This ‘total’ aircraft weight is calculated and stored as the ramp weight (variable name, rampweight).

$$r_w = f_w + p_w + c_w + p_{1w} + p_{2w} + e_w \qquad 2$$

$r_w$= rampweight
$f_w$ = fuelweight
$p_w$ = pilotweight
$c_w$ = copilotweight
$p_{1w}$ = passenger1weight
$p_{2w}$ = passenger2weight
$e_w$ = Empty_weight

iii. The moment of each respective section is given by the formula

$$m = w \times a \qquad 3$$

m = moment
w = weight in lbs.
a = moment arm in inches

iv. To find the total moment on the now occupied aircraft/full aircraft we simply add all our calculated moments

$$t_M = f_{sM} + f_M + r_{sM} + e_M \qquad 4$$

$t_M$ = totalMoment
$f_{sM}$ = frontSeatsMoment
$f_M$ = fuelMoment
$r_{sM}$ = rearSeatsMoment
$e_M$ = empty_aircraft_Moment

v. To calculate the center of gravity for the loaded aircraft, we simply divide the total moment (when loaded) calculated by the total aircraft weight (when loaded).

$$c_G = \frac{t_M}{r_W} \qquad 5$$

$t_M$ = totalMoment
$r_w$= rampweight

**Solution**

I will break down my solution into 3 steps.

a. Receiving user-input
b. Calculations (total weight, moments, and center of gravity)
c. User interface and calculator output display.

a. Receiving user-input

Overview

The following data was made available prior to user input.

| Variable | Value |
|---|---|
| Empty Weight | 1,471 lbs |
| Center of Gravity (empty) | 85.9 in |
| Front Seats Moment Arm | 85.5 in |
| Fuel Tanks Moment Arm | 95 in |
| Rear Seats Moment Arm | 118.1 in |
| Maximum ramp weight | 2,400 lbs |
| Maximum Fuel | 50 US Gallons |

Table 1: Table with important information on this aircraft.

My calculator required the user to enter in some required data which was used in calculating the aircraft weight and balance parameters.
The calculator required the user to input the following data:
Weight of the fuel on board (in US Gallons)
Weight of the pilot (in lbs)
Weight of the co-pilot (in lbs)
Weight of passenger one (in lbs)
Weight of passenger two (in lbs)
The user was also allowed to enter a 0-lbs weight of either passenger if there were to be no passengers onboard.
To achieve this, I utilized a custom (user-defined) function which asked the user to enter in the above-mentioned information and then stored the entered data.
In the main script, the function is called and used to receive the user's input.
Note that the function does not allow the user to enter an amount of fuel higher than 50 US Gallons.

Custom (user-defined) function

As stated above, the user-defined function was created to prompt the user to enter the data required for the weight-balance calculations.
The function begins by prompting the user for fuel onboard the aircraft in gallons. Since

the maximum fuel onboard the aircraft is 50 US gallons, the user is not allowed to proceed with the calculator's services unless they enter an amount of fuel within this limit. This is accomplished by using a while loop that does not allow the proceeding code to run until the user enters an amount of fuel within the stated limit. If the user does not comply, the function displays, 'Invalid input. Fuel can not be more than 50 gallons. Please try again.'

Once the user enters the correct amount of fuel, the function moves on to prompt the user for the rest of the data (pilot and copilot weights, and the two passengers' weights) in lbs (pounds). The user is allowed to enter '0' weight for either/or both passengers if that seat is vacant.

All the collected information is stored in an array for use by the main code.

The user-defined(custom) function to achieve all this is 'userinput'

It can be called by simply entering the function name (userinput) or by assigning it to a (1 x 5) row vector to store each of the user-inputted data.

b. Calculations (total weight, moments, and center of gravity)

When the required data from the user is collected using the user-defined (custom) function, the function (userinput) is called into the main code.

In order to use the data stored by the function, it is called by assigning it's each content into it's corresponding variable name. This is done by 'assigning' the function to a 1 X 5 row.

After doing this each collected data item can be called by their proper variable names as follows

1. fuel
2. pilotweight
3. copilotweight
4. passenger1weight
5. passenger2weight

These are used throughout the code.

The main code will use these data in conjunction with the data in *Table 1* to do the following:

Calculate the weight of the fuel
Calculate the ramp weight
Calculate the moment arm for each area of the aircraft
Calculate the Center of Gravity (CG) of the aircraft
Determine if the aircraft is within its weight and balance
Print out the ramp weight, loaded moment, and if it is in W&B

To begin, the amount of fuel entered (originally captured in US gallons), is converted into pounds. This is done by simply multiplying fuel by 6. This value is assigned to the variable name ‘fuelweight’.

$$f_w = f \times 6 \qquad 1$$

$f_w$ = fuelweight
f = amount of fuel in US gallons
This value is displayed to the user as the weight of fuel they entered.
The next step is to calculate the total weight of the loaded airplane. This is a sum of all the weights derived from the user’s input (fuelWeight, pilotWeight, coPilotWeight, passenger1Weight, passenger2Weight) and the weight of the aircraft when empty (Empty_weight).
This ‘total’ aircraft weight is calculated and stored as the ramp weight (variable name, rampweight).

$$r_w = f_w + p_w + c_w + p_{1w} + p_{2w} + e_w \qquad 2$$

$r_w$= rampweight
$f_w$ = fuelweight
$p_w$ = pilotweight
$c_w$ = copilotweight
$p_{1w}$ = passenger1weight
$p_{2w}$ = passenger2weight
$e_w$ = Empty_weight

This value is displayed to the user.
Now that I have calculated the aircraft’s total weight, it is necessary to perform the first aspect of the “weight-balance” check. In this step, the rampweight is compared to an arbitrary value, “maxWeight”, of 2400lbs (pounds). If the aircraft’s total weight exceeds this value, the calculator does not proceed with the rest of the code. It displays to the user that the aircraft exceeds the maximum allowable weight and is therefore unsafe to fly. The user then has the option to restart the calculator and enter different values.
If the weight is within safe limits for safe flight, the calculator will proceed to the next part of the program, calculating the moments and center of gravity.
The moments of specific parts of an aircraft and consequently the moment of the entire aircraft is an excellent measure of balance. It is important that the aircraft’s total moment falls within a certain threshold so that it does not lean towards the tendency to tip over. Before calculating the moment of the entire aircraft, the moments of particular ‘sections’ must be calculated and then summed up. The moment of each respective section is given by the formula

$$m = w \times a \qquad 3$$

m = moment
w = weight in lbs.
a = moment arm in inches

In this calculator, I will calculate the moment in 4 main sections: the front seats, the rear seats, the fuel, and the moment with the aircraft empty.
These will be calculated using the weights now stored in MATLAB (either entered by the user, or pre-existing) and the arm lengths (distance from 'pivot') which are already stored in the script. The arm lengths and original center of gravity of the empty airplane can be found in table 1.

The weight of each section is calculated via a very intuitive process of summing up the weights of the human beings/object that occupy those sections.
The total weight of the front seats section is the sum of the Pilot and copilot's weights.
The fuel weight is the fuel weight as previously calculated in my script.
The total weight of the rear section is the sum of the passengers weights (passenger 1's weight and passenger 2's weight).
Note that we will use the weight of the empty air craft to calculate the empty aircraft's moment. This value is given already and can be found in table 1.

Once we have calculated the moments for each of these four sections, we will have each moment stored in their respective variable names:
frontSeatsMoment
fuelMoment
rearSeatsMoment
empty_aircraft_moment.

To find the total moment on the now occupied aircraft/full aircraft we simply add all our calculated moments

$$t_M = f_{sM} + f_M + r_{sM} + e_M \qquad 4$$

$t_M$ = totalMoment
$f_{sM}$ = frontSeatsMoment
$f_M$ = fuelMoment
$r_{sM}$ = rearSeatsMoment
$e_M$ = empty_aircraft_Moment

The value stored in this variable name 'totalMoment' is the total moment of the airplane when occupied. Now to get a good sense of the aircraft's overall balance, we will need to

locate the aircraft's center of gravity. For an aircraft of this size, a center of gravity between 86.8 inches and 95.8 inches will be deemed safe to fly.
To calculate the center of gravity for the loaded aircraft, we simply divide the total moment (when loaded) calculated by the total aircraft weight (when loaded).

$$c_G = \frac{t_M}{r_W} \qquad 5$$

$t_M$ = totalMoment
$r_w$= rampweight

The calculator will then display this calculated center of gravity to the user.
To determine once and for all if the aircraft is within safe weight and balance limits for flying, the program checks if the calculated center of gravity is within the aforementioned limits of 86.8 inches and 95.8 inches. Based on this logic, the calculator displays whether the aircraft is within weight and balance limits that are safe to fly with.

c. User interface and calculator output display.
This calculator receives input from the user and performs calculations based on those inputs. The outputs of those calculations are displayed to the user in specific ways. This subsection will go through how the calculator 'communicates' with the user to prompt the user for data and display the output.

It is important to note that this calculator will continue to run until the user chooses not to. In order for the calculator to stop, the user will be giving instructions to enter number 1.
To do this I utilized a while loop that runs the entire script as long as the variable, 'answer' remains equal to zero. When the user is prompted to continue with the calculator or enter '1' if they wish to exit the calculator, the variable 'answer' is updated accordingly.
At the start of the script, 'answer' is initialized as 0 so that the code will run for the very first iteration.
After answer is initialized to be 0, the while loop runs the entire code.

Within the while loop the first point of interaction between the calculator and user is the greeting. The program greets the user at the very start of the program, saying, "Hello, today we will be using our Aircraft (Piper Cherokee PA-28-180), weight and balance calculator. Please follow instructions!"
This is done using the disp function.

Next, the script calls the function that prompts the user to input the amount of fuel and the weights needed for our weight and balance calculator. This function (userinput) is user-defined (custom-made).
First, the function prompts the user to input the amount of fuel on-board in gallons.
It achieves this using the disp function.
If the user enters a value higher than 50, the rest of the function does not run. This is achieved by using a while loop that displays to the user 'Invalid input. Fuel can not be more than 50 gallons. Please try again.'
The user is again prompted for the amount of fuel on-board in gallons.
Once that condition is satisfied, the function prompts the user for each weight to be entered (pilot's weight, copilot's weight, passenger1's weight, passenger2's weight).
Each parameter prompt throughout the MATLAB script and function code is achieved using the 'input' function.

After the above data is collected, the rest of the program runs. The aim of the calculator is to determine if the aircraft is within the correct weight and balance limits to fly safely and display that to the user. However, the calculator is also designed to display certain metrics to the user for the sake of transparency and an overall high-quality user experience.
The following calculated values are displayed to the user:
1. The total fuel weight
2. The total ramp weight (the total weight of the loaded aircraft)
Note that if the aircraft exceeds the maximum allowable ramp weight for flight (2400 lbs), the calculator will display that "Warning: The ramp weight exceeds the maximum allowable weight of %g lbs, you are not safe to fly. We will not go on with the calculation.\n", maxWeight)
3. The calculated center of gravity

After calculations, the calculator summarizes up the important calculated data and finally, the conclusion on whether the aircraft is safe to fly. It displays the following information to the user:
1. The ramp weight
2. The total moment (of the loaded aircraft)
3. Whether the aircraft is within the weight and balance limits that are safe to fly or not.

Note that each of these are displayed using the 'fprintf' function. This function is capable of displaying a sentence or strings and the actual value for the variable name mentioned.

Finally, the calculator will ask the user if they would like to perform another calculation. The user will be instructed to enter 1 if they do not want to.
If the user enters 1, the calculator displays the end greeting Thank you for using our

program, I hope you will come again! Goodbye." This is achieved using the 'input' function.
If not the entire script starts again from the beginning greeting.

Here, I will include my test values (inputs) and the corresponding calculator outputs.

To explore the full range of operation of the calculator, we will test the four (4) possible situations. These are,
a. User enters more than 50 gallons of fuel.

b. The aircraft is overweight (the total weight is greater than 2400 lbs.)

c. The aircraft is within the safe limits of weight but is outside the safe limits of balance.

d. The aircraft is within safe limits of weight and balance.

a. The user enters more than 50 gallons of fuel.
```
Hello, today we will be using our Aircraft (Piper Cherokee PA-28-180), weight and
balance calculator. Please follow instructions!
Enter the amount of fuel on-board (in gallons): 50.00005
Invalid input. Fuel can not be more than 50 gallons. Please try again.
Enter the amount of fuel on-board (in gallons):
```

b. The aircraft is overweight (total weight is greater than 2400lbs.)
```
Hello, today we will be using our Aircraft (Piper Cherokee PA-28-180), weight and
balance calculator. Please follow instructions!
Enter the amount of fuel on-board (in gallons): 40
Enter the weight of the pilot (in lbs): 200
Enter the weight of the co-pilot (in lbs): 200
Enter the weight of passenger 1 (in lbs): 2040
Enter the weight of passenger 2 (in lbs): 200
The total weight of the fuel is 240 lbs.
The ramp weight of the aircraft is 4351 lbs.
Warning: The ramp weight exceeds the maximum allowable weight of 2400 lbs, you are
not safe to fly. We will not go on with the calculation.
Would you like to perform another calculation? (1 for No):
```

c. The aircraft is within the safe limits of weight but is not within the safe limits of balance.
```
Hello, today we will be using our Aircraft (Piper Cherokee PA-28-180), weight and
```

```
balance calculator. Please follow instructions!
Enter the amount of fuel on-board (in gallons): 40
Enter the weight of the pilot (in lbs): 250
Enter the weight of the co-pilot (in lbs): 250
Enter the weight of passenger 1 (in lbs): 0
Enter the weight of passenger 2 (in lbs): 0
The total weight of the fuel is 240 lbs.
The ramp weight of the aircraft is 2211 lbs.
The Center of Gravity (CG) of the aircraft is located at 86.7973 inches.
Warning: The aircraft is outside its weight and balance limits, you are not safe to fly.
Would you like to perform another calculation? (1 for No):
```

d. The aircraft is within the safe limits of weight and balance

```
Hello, today we will be using our Aircraft (Piper Cherokee PA-28-180), weight and
balance calculator. Please follow instructions!
Enter the amount of fuel on-board (in gallons): 48
Enter the weight of the pilot (in lbs): 160
Enter the weight of the co-pilot (in lbs): 160
Enter the weight of passenger 1 (in lbs): 170
Enter the weight of passenger 2 (in lbs): 145
The total weight of the fuel is 288 lbs.
The ramp weight of the aircraft is 2394 lbs.
The Center of Gravity (CG) of the aircraft is located at 91.1781 inches.
Ramp Weight: 2394 lbs
Total Moment: 218280 lb-in
The aircraft is within its weight and balance limits.
Would you like to perform another calculation? (1 for No):
```

## Discussion

### Analysis and Interpretation of results.

In this section, I will discuss the analysis and interpretation of my results.
Weight and balance are extremely important metrics when it comes to determining if an aircraft is safe to fly. To fly, an aircraft must generate enough thrust to carry the weight of itself, its payload, and the passengers. However, even if the weight of the aircraft is below the accepted threshold, there may still be serious stability issues. The weight distribution of the aircraft plays a huge role in this. If the weights are crammed in one section of the plane, it creates a large moment. This moment in turn creates a tendency for the plane to tilt in that direction. To ensure safe flight for the Piper Cherokee PA-28-100, this calculator aims to inform the user of how safe it is to fly according to the weight and balance metrics determined by the user's input. Here is the analysis of the different possible scenarios.

a. The user enters more than 50 gallons of fuel

In such a scenario, the rest of the calculation is not proceeded with, the program stops running and displays to the user that they should enter a value that is not greater than 50. This is because the aircraft must not carry more than 50 gallons of fuel under any circumstances. In a small plane like the Piper Cherokee, I believe that carrying more than 50 gallons of fuel greatly limits the space available for passengers. This is because we use a conversion of 1 gallon to 6 lbs. for the weight conversion.
The conclusion of this scenario is that the aircraft will not fly, because it exceeds the safe weight limits for fuel onboard and has the potential to exceed the total weight limit for safe flight.

b. The aircraft is overweight (total weight is greater than 2400lbs.)

In such a scenario, the rest of the calculation is not proceeded with, the program stops running and displays to the user that the ramp weight (total weight) exceeds the maximum allowable value and hence, the aircraft is not safe to fly.

This is because exceeding the maximum weight is a huge compromise on stability and control, as well as structural integrity.
The conclusion of this scenario is that the aircraft will not fly because it exceeds the weight limits for safe flight.

c. The aircraft is within the safe limits of weight but is not within the safe limits of balance.

In such a scenario, the calculator finds and displays the total (ramp) weight of the aircraft and the Center of Gravity. However, the calculator also displays a warning that the aircraft is outside the weight and balance limits, so the aircraft is not safe to fly. In this situation, the aircraft is carrying a total weight below the maximum weight requirement. However, the distribution of the weight

between the aircraft sections makes the aircraft unstable. Although the aircraft is within weight limits, it is not within balance limits because it now has a large moment and has the tendency to become unstable. Since the sections we look at in the plane are ‘front’ and ‘rear’, improper weight distribution which causes a center of gravity too forward is likely to affect pitch. This is very unsafe.
The conclusion of this scenario is that the aircraft will not fly, because it exceeds the balance limits for safe flight.

d. The aircraft is within the safe limits of both weight and flight.

In such a scenario, the calculator finds and displays the total (ramp) weight of the aircraft and the Center of Gravity. The calculator also displays that the aircraft is within the weight and balance limits, so the aircraft is safe to fly. In this situation, the aircraft is carrying a total weight below the maximum weight requirement. and the distribution of the weight between the aircraft sections makes the aircraft stable.
The conclusion of this scenario is that the aircraft will fly, because it is within safe weight and balance limits to fly.

**Challenges faced and solutions that helped me complete this project.**
Fortunately, I did not face any serious challenges while doing this project. The only notable bottleneck was that I had to quickly revisit the topic ‘moments’ to complete the project.

**Problem 2**

**Problem statement**

Write briefly about the problem you have solved

One important aspect of flying any aircraft is the Density Altitude. The Density Altitude of an aircraft plays a vital role in many important things such as lift generation and overall aircraft performance. This altitude is different from the absolute altitude. It must be calculated using the temperature, humidity, and pressure of the atmosphere surrounding the aircraft.

In this project, I created a MATLAB program calculator that will calculate the density altitude based on real time weather information that I will retrieve with the program.

This calculator receives weather information in real time (the input data is received almost immediately) using an Application Programming Interface or API. The website Openweathermap.org is what I used in this program to provide the weather data. MATLAB can connect to this website using an API key to obtain the data, The code below is provided below was used for this.

Note that stable internet connection is needed for this to work.

The script receives the following information from the website:
a. Temperature (in Kelvin)

b. Pressure in millibars

c. Humidity (in Celsius)
The calculator then uses this information to calculate and display the Dew point temperature (in Celsius), Vapor Pressure (in millibars), Pressure (in inHG (inches mercury)), Virtual Temperature (in Kelvin), Virtual Temperature (in Rankine), and finally the calculated Density Altitude (in Feet).

**Theory**

i. To convert the temperature from Kelvin to Celsius, the calculator calculates the dewpoint temperature in Degrees Celsius.

$$C = K - 273.15 \qquad 6$$

C = temperature in Celsius

K = temperature in Kelvin

ii. Using the calculated dewpoint temperature, the script calculates the vapor pressure, also in Degrees Celsius.

$$T_d = t_c - \left(\frac{100 - h}{5}\right) \qquad 7$$

$$e = 6.11 \times 10^{\frac{7.5 * T_d}{237.7 + T_d}} \qquad 8$$

T_d = dewpoint temperature

$t_c$ = temperature in Celsius

h = humidity

e = vapor pressure

iii. The virtual temperature in Kelvin is calculated as follows.

$$T_v = \frac{t}{1 - \left(\frac{e}{p}\right) \times (1 - 0.622)} \qquad 9$$

$T_v$ = Virtual temperature

t = temperature in Kelvin

e = vapor pressure

p = pressure in millibars

iv. The Virtual temperature in Kelvin is then converted to Rankine and stored in the variable name 'virtualtempR'.

$$T_{VR} = \left(\frac{9}{5} \times (T_v - 273.15) + 32\right) + 459.69 \qquad 10$$

$T_{VR}$ = Virtual Temperature in Rankine

$T_v$ = Virtual Temperature

v. To calculate the Density altitude, the pressure we received from the website (in millibars) must be converted into inches of mercury (inHG).

$$p_{inHG} = P_{mb} \times 0.02953 \qquad 11$$

$P_{inHG}$ = Pressure in inches Mercury

$P_{mb}$ = Pressure in millibars

vi. Finally, the Density Altitude is calculated.

$$d_A = f_{elv} + \left(145366 * \left(1 - \left(\frac{17.326 * p_{inHG}}{T_{VR}}\right)^{0.235}\right)\right) \qquad 12$$

$d_A$ = Density Altitude
$f_{elv}$ = field elevation
$P_{inHG}$ = Pressure in inches Mercury
$T_{VR}$ = Virtual Temperature in Rankine

**Solution**

I will break down my solution into 3 steps.

a. Receiving website input
b. Calculations (the Dew point temperature (in Celsius), Vapor Pressure (in millibars), Pressure (in inHG(inches mercury)), Virtual Temperature (in Kelvin), Virtual Temperature (in Rankine), and finally the calculated Density Altitude (in Feet))
c. Calculator output display.

a. **Receiving website input**

The following data was received from the Openweathermap.org website.

| Variable | Unit |
|---|---|
| Temperature | Kelvin |
| Pressure | millibars |
| Humidity | Percentage |

Table 2: Table with important weather information from the Openweathermap.org website.

The calculator received the above input from the website using the following code

```
key = '7cab1fcaf444883263bc48dd983e6018';
options = weboptions('ContentType','json');
url=['https://api.openweathermap.org/data/2.5/weather?q=','Ames','&APPID=',key];
CurrentData = webread(url, options);
temp = CurrentData.main.temp;
pressure = CurrentData.main.pressure;
humidity = CurrentData.main.humidity;
```

b. **Calculations** (the Dew point temperature (in Celsius), Vapor Pressure (in millibars), Pressure (in inHG(inches mercury)), Virtual Temperature (in Kelvin), Virtual Temperature (in Rankine), and finally the calculated Density Altitude (in Feet))

When the required data is received from the website, it is stored in the script according to the corresponding variable names.
The current temperature is stored under variable name ‘temp’, the current pressure is stored under variable name ‘pressure’, the current pressure is stored under the variable name ‘pressure’. The variable names are the means through which I will use these variables in subsequent calculations.
Before we begin with the desired calculations, we will need to first convert the temperature received from the website (in Kelvin) to Degrees Celcius. We will do this

using a user-defined (custom) function.
The custom function, 'Kelv_to_Celc' receives the temperature in Celsius, converts it to Kelvin, and gives the value in Kelvin as the output.

This function is called by the main script by assigning its output to the variable name 'tempC'. By doing this, I assign the value of temperature in Degrees Celsius to the variable name tempC for later use.
The main code will use this data in conjunction with the data in *Table 2* to do the following:

Calculate the Vapor pressure and Dewpoint
Calculate the Virtual Temperature
Convert the Virtual Temperature from Kelvin to Rankine
Calculate the Density Altitude

The script calculates these values one after the other using simple formulae. These calculations are relatively simple. However, attention to detail in the formulae is advised.

After using the custom function 'Kelv_to_Celc' to convert the temperature from Kelvin to Celsius, the calculator calculates the dewpoint temperature in Degrees Celsius.

$$C = K - 273.15 \qquad 6$$

C = temperature in Celsius
K = temperature in Kelvin
The dewpoint temperature is stored under the variable name 'T_d'
Using the calculated dewpoint temperature, the script calculates the vapor pressure, also in Degrees Celsius. This value is stored under the variable name 'e'.

$$T_d = t_c - \left(\frac{100 - h}{5}\right) \qquad 7$$

$$e = 6.11 \times 10^{\frac{7.5 * T_d}{237.7 + T_d}} \qquad 8$$

T_d = dewpoint temperature
$t_c$ = temperature in Celsius
h = humidity
e = vapor pressure

The script then reassigns the variables 'e' and 'pressure' to 'vaporpressure' and 'P_mb' respectively. This is done because the question demands so.

Next, the script calculates the virtual temperature in Kelvin and stores it in the variable name ‘virtualtemp’.

$$T_v = \frac{t}{1 - \left(\frac{e}{p}\right) \times (1 - 0.622)} \qquad 9$$

T_v = Virtual temperature
t= temperature in Kelvin
e = vapor pressure
p = pressure in millibars

For calculation purposes, the variable virtual temperature is reassigned to the new variable name ‘T_v’. This is done to avoid errors when calling the variable in formulae. This variable is then converted to Rankine and stored in the variable name ‘virtualtempR’.

$$T_{VR} = \left(\frac{9}{5} \times (T_v - 273.15) + 32\right) + 459.69$$

$T_{VR}$ = Virtual Temperature in Rankine
$T_v$ = Virtual Temperature

Before calculating the density altitude, it is worth to note that with our given sets of data, the calculator would calculate the density altitude of the airplane, neglecting the altitude of Ames(the city where this data is calculated for) is 955.6 feet above sea level. This value is added to the value calculated for Density Altitude, but it is first stored in the variable name ‘fieldelv’.

The last few lines of code calculate the Density Altitude. To calculate the Density altitude, the pressure we received from the website (in millibars) must be converted into inches of mercury (inHG). It is stored in the variable name ‘pressureinHG’

$$p_{inHG} = P_{mb} \times 0.02953 \qquad 10$$

$P_{inHG}$ = Pressure in inches Mercury
$P_{mb}$ = Pressure in millibars
Finally, the Density Altitude is calculated and stored in the variable name ‘densityAltitude’.

$$d_A = f_{elv} + \left(145366 \times \left(1 - \left(\frac{17.326 * p_{inHG}}{T_{VR}}\right)^{0.235}\right)\right) \qquad 11$$

$d_A$ = Density Altitude
$f_{elv}$ = field elevation
$P_{inHG}$ = Pressure in inches Mercury

$T_{VR}$ = Virtual Temperature in Rankine

c. **Calculator output display**

This calculator does not receive any input for the user and does not communicate with the user until the output display.
At the end of the program, the calculator displays the following information

```
Dew Point Temperature (C):
Vapor Pressure (mb):
The pressure (inHG):
Virtual Temperature (K):
Virtual Temperature (R):
The calculated density altitude is:                    feet
```

This is achieved by using the ‘fprintf’ function and the corresponding variable names.

– include any output that your programs generated.
Here, I will display the information gotten from the website, and the calculators outputs.
Website information

```
temp =

  296.3400


pressure =

        1023


humidity =

    42
```

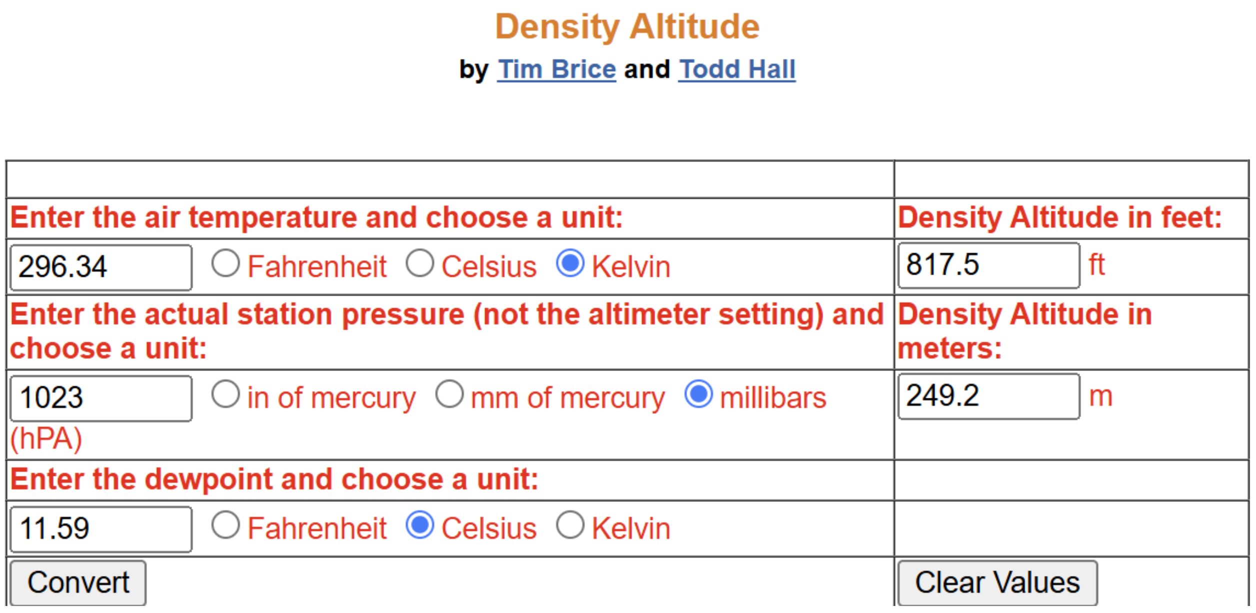
Calculator output

```
Dew Point Temperature (C): 11.59
Vapor Pressure (mb): 13.64
The pressure (inHG): 30.21
Virtual Temperature (K): 297.84
Virtual Temperature (R): 536.13
The calculated density altitude is: 1774.13 feet
```

## Discussion

### Analysis and Interpretation of results.

The density altitude calculated was 1774.13 feet. This value can be confirmed using the NOAA calculator. Note that the NOAA calculator will calculate the Density Altitude neglecting the elevation of Ames above sea level.



**Density Altitude**

**by Tim Brice and Todd Hall**

| | |
|---|---|
| | |
| **Enter the air temperature and choose a unit:** | **Density Altitude in feet:** |
| 296.34 Fahrenheit Celsius [✓ Kelvin] | 817.5 ft |
| **Enter the actual station pressure (not the altimeter setting) and choose a unit:** | **Density Altitude in meters:** |
| 1023 in of mercury mm of mercury [✓ millibars (hPA)] | 249.2 m |
| **Enter the dewpoint and choose a unit:** | |
| 11.59 Fahrenheit [✓ Celsius] Kelvin | |
| Convert | Clear Values |

The Density Altitude calculated by this website is 817.5ft. Adding the elevation of Ames (955.6ft), we get the Density Altitude to be 1773.1ft. This value is extremely close to what our calculator estimates (1774.13ft).

From my AERE E 1600 class on aircraft performance, I know that density altitude is very important in aviation because it is a strong determinant of the power required and lift for an aircraft. The higher the density altitude, the higher the power required.

### Challenges faced and solutions that helped me complete this project.

Fortunately, I did not face any serious challenges while doing this project. The only notable bottleneck was that I had to revise the order of my parentheses when entering formulae in MATLAB.

## Appendices

### Appendix A : Wbcalc (Weight Balance Calculator)

```
clear,clc
%{
Chinedum Echedom, Project, Problem 1
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.
1. Obtain missing information from the user
2. Define variables you will use with the information given
3. Calculate the weight of the fuel
4. Calculate the ramp weight
5. Calculate the moment arm for each area of the aircraft
6. Calculate the Center of Gravity (CG) of the aircraft
7. Determine if the aircraft is within it's weight and balance
8. Print out the ramp weight, loaded moment, and if it is in W&B
I will obtain the values from the user using the function I already made
"userinput"
userinput
function[fuel,pilotWeight,coPilotWeight,passenger1Weight,passenger2Weight]=
userinput(fuel,pilotWeight,coPilotWeight,passenger1Weight,passenger2Weight)
%}

answer = 0; % initialize "answer" for the while loop
while answer == 0 % While loop too run as long as the user wants to use the calculator
    disp("Hello, today we will be using our Aircraft (Piper Cherokee PA-28-180), weight and
balance calculator. Please follow instructions!"); % Greet the user to begin
```

```
    [fuel, pilotWeight, coPilotWeight, passenger1Weight, passenger2Weight] = userinput; % call
the function

    % Calculate the weight of the fuel (assuming 6 lbs per gallon)

    fuelWeight = fuel * 6; %fuel weight calculation

    fprintf("The total weight of the fuel is %g lbs. \n",fuelWeight) %display the total fuel weight
to the user

    % Calculate the ramp weight (total weight of the aircraft)

    Empty_weight = 1471; % set empty weight of the aircraft as 1471 pounds

    rampWeight = fuelWeight + pilotWeight + coPilotWeight + passenger1Weight +
passenger2Weight+Empty_weight; % calculate the ramp weight

    fprintf("The ramp weight of the aircraft is %g lbs. \n", rampWeight); % display the ramp
weight to the user.

    % Determine if we have exceeded maximum weight. If not, go on with the program

    maxWeight = 2400; % Define the maximum allowable weight for the aircraft

    if rampWeight <= maxWeight %if statement to calculate the proceeding values if the
rampweight is not greater than the maximum weight

        % Calculate the moment arm for each area of the aircraft

        Front_Seats_Moment_Arm = 85.5; % distance from reference point (fulcrum) to Front
Seats in inches

        Fuel_Tanks_Moment_Arm = 95; % distance from reference point (fulcrum) to Fuel Tanks
in inches

        Rear_Seats_Moment_Arm = 118.1; % distance from reference point (fulcrum) to Rear Seats
in inches

        Center_of_gravity = 85.9; % distance from reference point  (fulcrum) to center of gravity in
inches

        % Calculate the moments for each weight

        frontSeatsMoment = (pilotWeight+coPilotWeight) * Front_Seats_Moment_Arm; %
calculate the front seats moment

        fuelMoment = fuelWeight * Fuel_Tanks_Moment_Arm; % calculate the fuel moment
        rearSeatsMoment = (passenger1Weight + passenger2Weight) * Rear_Seats_Moment_Arm;
% calculate the rear seats moment

        empty_aircraft_moment = Empty_weight*Center_of_gravity;

        % Calculate the total moment and the Center of Gravity (CG)

        totalMoment = frontSeatsMoment + fuelMoment + rearSeatsMoment +
empty_aircraft_moment; % calculate the total moment

        CG = totalMoment / rampWeight; % Calculate the Center of Gravity

        fprintf("The Center of Gravity (CG) of the aircraft is located at %g inches.\n", CG); %
Display the CG to the user.

        % Determine if the aircraft is within its weight and balance limits

        if CG >= 86.8 && CG<= 95.8 % if statement to finish the program if the center of gravity
value is within the bounds

      % Print out the ramp weight, loaded moment, and if it is within weight and balance

      fprintf("Ramp Weight: %g lbs\n", rampWeight); % display the total airplane weight

      fprintf("Total Moment: %g lb-in\n", totalMoment); % display the total airplane moment

      fprintf("The aircraft is within its weight and balance limits.\n"); % state that the aircraft is
within weight and balance limits

        else

      fprintf("Warning: The aircraft is outside its weight and balance limits, you are not safe to
fly.\n"); % state that the aircraft is not within weight and balance limits

        end

        else

        fprintf("Warning: The ramp weight exceeds the maximum allowable weight of %g lbs, you
are not safe to fly. We will not go on with the calculation.\n", maxWeight); % state that the
aircraft is not within weight limits

    end

    answer = input('Would you like to perform another calculation? (1 for No): '); % ask the user if
they want to use the calculator again

    if answer ==1
```

```
        disp(" Thank you for using our program, I hope you will come again! Goodbye.") %
Display exit greeting to the user
    end


end
```

**Appendix B: userinput (User-defined function to receive input)**

```
function[fuel,pilotWeight,coPilotWeight,passenger1Weight,passenger2Weight]=
userinput(fuel,pilotWeight,coPilotWeight,passenger1Weight,passenger2Weight)

% Echedom,AERE1610,Project Problem 1

% write a function that will ask the user for the input. Your function will prompt the user

% for an input  Your function will need to ask the user to input how much fuel is on-board, the
weight of the pilot,

% co-pilot, passenger 1 and passenger 2.

% if the fuel is less than 50 gallons, tell the user that it's invalid and

% needs to be inputed again

fuel = input('Enter the amount of fuel on-board (in gallons): ');

while fuel > 50

   disp('Invalid input. Fuel can not be more than 50 gallons. Please try again.');

   fuel = input('Enter the amount of fuel on-board (in gallons): ');

end

% Prompt the user for the weight of the pilot, co-pilot, and passengers

pilotWeight = input('Enter the weight of the pilot (in lbs): ');

coPilotWeight = input('Enter the weight of the co-pilot (in lbs): ');

passenger1Weight = input('Enter the weight of passenger 1 (in lbs): ');

passenger2Weight = input('Enter the weight of passenger 2 (in lbs): ');

% Store the weights in an array (including fuel) for further processing if needed

end
```

**Appendix C - DensityAltCalc (Density Altitude Calculator)**

```
clear,clc
%{
Chinedum Echedom, Project, Problem 2
Matlab program that will calculate the density
altitude based on real time weather information that you will retrieve with
your program.
First, let's obtain our
weather data. Calculating the air density altitude requires several other
items to be calculated but we will base our calculation on obtaining real
weather information. We will then proceed to use this data to make other
calculations until we finally calculate our density altitude.
The Kelvin to Celcius function is
[Celcius] = Kelv_to_Celc(Kelvin)
%}

% receive real-time weather data from the website
key = '7cab1fcaf444883263bc48dd983e6018';
options = weboptions('ContentType','json');
url=['https://api.openweathermap.org/data/2.5/weather?q=','Ames','&APPID=',key];
CurrentData = webread(url, options);
temp = CurrentData.main.temp;
pressure = CurrentData.main.pressure;
humidity = CurrentData.main.humidity;

tempC = Kelv_to_Celc(temp); % Convert temperature from Kelvin to Celsius
```

```
T_d = tempC - ((100 - humidity) / 5); % Calculate the dew point temperature (T_d) (in celcius) using the humidity and temperature

e = 6.11*10^((7.5*T_d)/(237.7+T_d));% Calculate the vapor pressure (e) in millibars


vaporpressure = e; %set vapor pressure (millibars) equal to 'e'

P_mb = pressure; % reassign pressure to P_mb (millibars)

T_v =  temp/(1-(e/pressure)*(1-0.622));% Calculate Virtual Temperature (kelvin)

virtualtemp = T_v; % reassign T_v to virtual temp in kelvin

virtualtempR = (9/5 * (virtualtemp - 273.15)+32) + 459.69;% Convert virtual temperature from Kelvin to Rankine

fieldelv = 955.6; % Factor into consideration that Ames has a field elevation of 955.6 feet


% Calculate Density Altitude

% We need to conver pressure from millibar to inches of HG

pressureinHG = P_mb * 0.02953; % Convert pressure from millibar to inches of Hg

densityAltitude = fieldelv + (145366 * (1 - ((17.326*pressureinHG)/virtualtempR)^0.235)); % Calculate the density altitude (feet) using the formula

%Display all calculated values

fprintf('Dew Point Temperature (C): %.2f\n', T_d); % Display the Dew point temperature (Celcius)

fprintf('Vapor Pressure (mb): %.2f\n', vaporpressure);% Display the calculated Vapor Pressure (millibars)

fprintf('The pressure (inHG): %.2f\n', pressureinHG)

fprintf('Virtual Temperature (K): %.2f\n', virtualtemp); % Display the calculated Virtual temperature (Kelvin)

fprintf('Virtual Temperature (R): %.2f\n', virtualtempR);% Display the calculated Virtual temperature (Rankine)

fprintf('The calculated density altitude is: %.2f feet\n', densityAltitude); % Display the calculated density altitude (feet)
```

## Appendix D: Kelv_to_Celc (User-defined function to convert temperature from Kelvin to Celsius)

```
function [Celcius] = Kelv_to_Celc(Kelvin)

% Function to convert inputed temperature from Kelvin to dgerees Celcius.

%[Celcius] = Kelv_to_Celc(Kelvin)

Celcius = Kelvin - 273.15;

end
```