3rd Year Computational Physics : Jessica Murphy

25/01/2024

Assignment 6

This python notebook investigates a **simple climate physics model** for the Earth. In particular, key learning objectives for this assignment were investigating the varying equilibrium temperature of the Earth during orbit, investigating the sources and sinks of greenhouse gases on the Earth's atmosphere and temperature (considering *albedo* and emissivity) and learning about *radiative forcing* and therefore the relative temperature change.

Background Information: The Earth's average temperature is determined by achieving a balance between the incoming solar radiation from the Sun and the outgoing thermal radiation emitted by the Earth. Initially, this equilibrium can be approximated by a straightforward calculation, but as we refine our models, we must consider the intricate influences of the Earth's atmosphere, i.e. the aforementioned sources and sinks of greenhouse gases on the Earth's atmosphere.

Task One

The objective of this task is to investigate the *equilibrium temperature* of the most basic model Earth.

To determine the equilibrium temperature of the basic model Earth, we can set the incoming solar power equal to the outgoing thermal power and solve for the temperature. The equation for the incoming solar power (Pin) is given as:

$$P_{in} = \frac{L_{\odot}}{4\pi D^2}\pi R_E^2$$

If the Earth were a perfect blackbody, it would then emit energy according to the Stefan Boltzmann law.

$$P_{out} = 4\pi R_E^2 \sigma T^4$$

Equating these results to get an expression for T:

$$T = \left(\frac{L}{16\sigma\pi D^2}\right)^{\frac{1}{4}}$$

1. Import the python library NumPy as np for the purposes of calculation
2. Set the constants
   - Luminosity $L = 2.828x10^{26} Js^{-1}$
   - Distance of Earth from Sun $D = 1.496x10^{11} m$
   - Stefann-Blotzmann constant $\sigma = 5.67x10^{-8} W/(m^2 K^4)$
3. Calculate the equilibrium temperature in Kelvin using the formular for T derived abvove
4. Convert temperature in Kelvin to Celsius by taking away 273.15
5. Print results to compare

In [1]:
```
# on balancing the equations

# import necessary python libraries
import numpy as np

# constants
L = 3.828e26 # Luminosity of the sun (J s-1)
D = 1.496e11 # distance of earth from sun (m)
sigma = 5.67e-8  # Stefan-Boltzmann constant in W/(m^2 K^4)

# Calculate equilibrium temperature in Kelvin
T_kelvin = ((L / (16 * np.pi * sigma * D**2)))**(1/4)

# Convert temperature to Celsius
T_celsius = T_kelvin - 273.15

# print the respective calculated temperatures
# to 2 sig fig as that is the smallest value in the caluclation
print('Equilibrium temperature in Kelvin = {0:.2f} K '.format(T_kelvin))
print('Equilibrium temperature in Celsius = {0:.2f} °C '.format(T_celsius))
```

```
Equilibrium temperature in Kelvin = 278.33 K
Equilibrium temperature in Celsius = 5.18 °C
```

Task two

The objective for task two is similar to task one, but this time with the distance from the earth varying as it is in orbit. Particulary, this task determines the range of the temperature of Earth at its closest approach to the Sun - *perihelion* ($D = 147,098,074 km$) to its furthest distance

from the Sun - *aphelion* ($D = 152,097,701km$). A comment is made below on whether the eccentricity of the orbit has a significant effect on the daily average temperature.

The same equation for T is used but now inlcuding the different distances.

1. Set the perihelion and aphelion distances as constants and convert to SI units
2. Caluclate the respective temperatures for each
3. Convert to Celsius
4. Print the results

In [2]:
```
# task 2
# determine range of temperatures expected over the orbit of the earth in thhis model from perihelion to aphelion

# set distances
D_perihelion_km = 147098074 # distance in km
D_aphelion_km = 152097701 # distance in km

D_perihelion = D_perihelion_km * 1e3 # distance in metres
D_aphelion = D_aphelion_km * 1e3 # distance in metres

# Calculate equilibrium temperature in Kelvin
T_perihelion_Kelvin = ((L / (16 * np.pi * sigma * D_perihelion**2)))**(1/4)
T_aphelion_Kelvin = ((L / (16 * np.pi * sigma * D_aphelion**2)))**(1/4)


# Convert temperature to Celsius for easier comparison when reading
T_perihelion_celsius = T_perihelion_Kelvin - 273.15
T_aphelion_celsius = T_aphelion_Kelvin - 273.15

# print the respective calculated temperatures
# to 2 sig fig as that is the smallest value in the caluclation
print('Perihelion Equilibrium Temperature = {0:.2f}K, {1:.2f} °C '.format(T_perihelion_Kelvin, T_perihelion_celsius))
print('Aphelion Equilibrium Temperature = {0:.2f}K, {1:.2f} °C '.format(T_aphelion_Kelvin, T_aphelion_celsius))
```

```
Perihelion Equilibrium Temperature = 280.69K, 7.54 °C
Aphelion Equilibrium Temperature = 276.04K, 2.89 °C
```

From these results, it is clear to see when the Earth is closer to the Sun (perihelion), it receives more solar radiation per unit area, leading to slightly warmer temperatures. Conversely, when the Earth is farther from the Sun (aphelion), it receives less solar radiation per unit area, resulting in slightly cooler temperatures. From the figures avove, the eccentricity of the orbit only contributes as a varying factor to daily average temperature of approximately 2.5 degrees, so therefore it is fair to conclude that it is a relatively minor factor in annual season climate variations. Factors that may contribute a larger impact to varied temperature differences may be axial tilt and atmospheric conditions such as the greenhouse effect, which is explored below.

Task three

The aim for task three is to understand the different efficiencies at absorbing and emitting radiation that the Earth has. This is intertwined with the idea of the earth being a source and sink for greenhouse gases, but more on that later. Task three is about understanding the concept of albedo - the measure of reflectivity of radiation. This varies from 0-1 where 0 is very absorbing (a sink) and 1 is very reflective (a source). This task explores the involvement of albedo and emissivity as a parameter for measuring equilibrium temperature for Venus, Earth and Mars.

The updated equation for Pin is therefore:

$$P_{in} = S_o(1-\alpha)\pi R_E^2$$

where we now adopt the *solar constant* $S_o = \frac{L_\odot}{4\pi D^2} = 1370 W m^{-2}$ which uses an average value of D over the elliptical orbit of the Earth. The emissivity ϵ of the Earth is a measure of how well radiation is emitted. The updated equation for P_{out} is therefore:

$$P_{out} = 4\pi R_E^2 \epsilon \sigma T^4$$

To determine the equilibrium temperature for Venus, Earth and Mars using this improved model, an emissivity of $\epsilon = 1$ for Earth and Venus was adopted, and an emissivity of $\epsilon = 0.8$ for the rocky surface of Mars.

Below this is compared to the observed average temperetaure for each planet.

1. Set the constants
   - Solar constant $S_o = 1370 W/m^2$
   - $AU = 1.496x10^{11}$
2. For Venus, Earth and Mars determine;
   - The distance form the sun in SI units
   - the albedo (reflection constant) α
   - the emissivity value ϵ
3. Calculate the equilibrium temperature for each planet using the solar constant
4. Calculate the equilibrium temperature for each planet using the distances from the sun and luminosity factor
5. Print the respective results
6. A comparison is made of the quantities calculated
7. A trial and error is carried out to approximate a more accurate value for the emissivity to counter in the greenhouse effect on temperature

In [3]:
```
#task 3
# calculating the equilibrium temperature for each planet and comparing it with the observed average temperature.

# constants
S = 1370 # solar constant (W/m^2)
au = 1.496e11 # 1 au

# Venus values
D_venus_au = 0.72 # distance of venus from sun (au)
D_venus = (D_venus_au) * au # converting atronomical units to the SI unit for distance meters
v_albedo = 0.72 # albedo (reflection constant)
v_epsilon = 1 #emmissivity value

# Earth values
D_earth_au = 1 # distance of earth from sun (au)
D_earth = (D_earth_au) * au # converting atronomical units to the SI unit for distance meters
e_albedo = 0.3 # albedo (reflection constant)
e_epsilon = 1 # emmissivity value

# Mars values
D_mars_au = 1.52 # distance of mars from sun (au)
D_mars = (D_mars_au) * au # converting atronomical units to the SI unit for distance meters
m_albedo = 0.25 # albedo (reflection constant)
m_epsilon = 0.8 # emmissivity value

# calculating equilibrium temperature fort each planet using the solar constant given
T_venus = ((S * (1 - v_albedo)) / (4 * v_epsilon * sigma)) ** (1/4)
T_earth = ((S * (1 - e_albedo)) / (4 * e_epsilon * sigma)) ** (1/4)
T_mars = ((S * (1 - m_albedo)) / (4 * m_epsilon * sigma)) ** (1/4)

# calculating equilibrium temperature fort each planet using the fact the solar constant = L/4piD^2
T_venus2 = (((L / (4 * np.pi * (D_venus**2))) * (1 - v_albedo)) / (4 * v_epsilon * sigma)) ** (1/4)
T_earth2 = (((L / (4 * np.pi * (D_earth**2))) * (1 - e_albedo)) / (4 * e_epsilon * sigma)) ** (1/4)
T_mars2 = (((L / (4 * np.pi * (D_mars**2))) * (1 - m_albedo)) / (4 * m_epsilon * sigma)) ** (1/4)

# # function to calculate equilibrium temperature fort each planet
# def equilibrium_temp(S, albedo, emissivity):
#     return ((S * (1 - albedo) / (4 * emissivity * sigma)) ** 1/4

# # calculating equilibrium temperature fort each planet
# T_venus = ((S * (1 - v_albedo) / (4 * v_epsilon * sigma)) ** 1/4
# T_earth = ((S * (1 - e_albedo) / (4 * e_epsilon * sigma)) ** 1/4
# T_mars = ((S * (1 - m_albedo) / (4 * m_epsilon * sigma)) ** 1/4

# print results
print('Equilibrium temperature for Venus using solar constant = {0:.2f} K '.format(T_venus))
print('Equilibrium temperature for Earth using solar constant = {0:.2f} K '.format(T_earth))
print('Equilibrium temperature for Mars using solar constant = {0:.2f} K '.format(T_mars))
print('')
print('Equilibrium temperature for Venus = {0:.2f} K '.format(T_venus2))
print('Equilibrium temperature for Earth = {0:.2f} K '.format(T_earth2))
print('Equilibrium temperature for Mars = {0:.2f} K '.format(T_mars2))
```

```
Equilibrium temperature for Venus using solar constant = 202.80 K
Equilibrium temperature for Earth using solar constant = 255.00 K
Equilibrium temperature for Mars using solar constant = 274.32 K

Equilibrium temperature for Venus = 238.61 K
Equilibrium temperature for Earth = 254.59 K
Equilibrium temperature for Mars = 222.14 K
```

Upon reflection, it is not wise to calculate the temperatures using the solar constant as it says in the manual that the distance involved in the solar constant is an average value of D over the elliptical orbit of the Earth, therefore not including Mars and Venus. This is why it is only comparibly accurate for Earth. The luminosity of the sun is constant because **luminosity is an absolute measure of radiant power; that is, its value is independent of an observer's distance from an object.**

To compare these calculated values easier, I will insert a table

	Distance from the sun in au	Albedo	Observed average temperature	Caluclated average temperature
Venus	0.72	0.72	740 K	238.61 K
Earth	1.00	0.3	288 K	254.59 K
Mars	1.52	0.25	223 K	222.14 K

On comparing, the most accurate value seems to be mars. This may be due to the fact that Mars' greenhouse effect is much weaker than Earth's, therefore the albedo and emissivity parameters may be a lot more accurate compared to Earth and Venus. The missing ingredient for the models of Venus and Earth is a significant atmosphere with a greenhouse effect due to carbon dioxide and other gases.

Now estimating a value for ϵ that will lead to the correct average temperature of approx 288K The temperature needs to be increased in both our estimates so an estimation of a lower epsilon will be carried out, assuming albedo is a constant.

In [4]:
```
# trial and error of changing value of epsilon, lowering it
e_epsilon_2 = 0.7

# calculate changed earth's temperature
T_earth3 = (((L / (4 * np.pi * (D_earth**2))) * (1 - e_albedo)) / (4 * e_epsilon_2 * sigma)) ** (1/4)
```

```
# print new temperature
print('Equilibrium temperature for Earth = {0:.2f} K '.format(T_earth3))
```

```
Equilibrium temperature for Earth = 278.33 K
```

This epsilon yields a temperature that is a bit too low so another is estimated.

In [5]:
```
# trial and error of changing value of epsilon
e_epsilon_3 = 0.6

# calculate changed earth's temperature
T_earth4= (((L / (4 * np.pi * (D_earth**2))) * (1 - e_albedo)) / (4 * e_epsilon_3 * sigma)) ** (1/4)

# print new temperature
print('Equilibrium temperature for Earth = {0:.2f} K '.format(T_earth4))
```

```
Equilibrium temperature for Earth = 289.27 K
```

An emissivity value of $\epsilon = 0.6$ appears to be the closest the 288 K figure. Therefore this will be taken as the most accurate trial.

Additional information

Upon considering this task involves a trial and error, it was considered that a loop iterating over all the temperatures, and therefore emissivity values may be a more efficient method to get the required ϵ value.

Below are the steps as usual.

1. Set the constant for the observed earth temperature of 288 K
2. Create a list 'epsilon_values' which contains a range of values for ϵ to try with a step size of 0.01 for better accuracy than the trialled value
3. The closest epsilon value discovered and the smallest temperature difference recognised during the iteration are stored in the variables closest_epsilon and min_temp_diff, respectively, after they are initialized. Float('inf') is the initial value of min_temp_diff, which makes sure that any computed difference is as large as possible
4. The program iterates over each epsilon value in the epsilon_values list:
    - For each epsilon value, calculate the equilibrium temperature.
    - Then calculate the absolute difference (temp_diff) between the calculated equilibrium temperature (T_earth) and the observed average temperature of the Earth (observed_temp_earth).
5. During each iteration, the loop updates closest_epsilon and min_temp_diff if the current temp_diff is smaller than the previously stored minimum temperature difference
6. Print the closest ϵ found

In [6]:
```
# Constants
observed_earth_temp = 288  # in Kelvin

# Range of values for epsilon to try
epsilon_values = [i * 0.01 for i in range(1, 101)]  # from 0.01 to 1.00 with step 0.01

# Initialize variables to store the closest epsilon value and the smallest temperature difference
closest_epsilon = 0
min_temp_diff = float('inf')

# Iterate over epsilon values
for epsilon in epsilon_values:
    # Calculate equilibrium temperature for Earth
    Temp_earth = ((S * (1 - e_albedo)) / (4 * epsilon * sigma))**(1/4)

    # Calculate temperature difference
    temp_diff = abs(Temp_earth - observed_earth_temp)

    # Update closest epsilon and minimum temperature difference if needed
    if temp_diff < min_temp_diff:
        closest_epsilon = epsilon
        min_temp_diff = temp_diff

# Print the closest epsilon value
print(f"The closest epsilon value for Earth: {closest_epsilon}")
```

```
The closest epsilon value for Earth: 0.61
```

This emissivity value is quite close to the estimated value of 0.6 trialled above. Therefore this seems satisfactory.

Task four

The objective of this task is to understand the concept of *radiative forcing* and its effect on the Earth's temperature. Namely, it is providing an insight into the current climate as according to the data sheet given, CO_2 has crossed the 400 parts per million (ppm) milestone recently. It is giving an insight to the radiative forcing that is happening currently.

Radiative forcing is a measure of the change in trapped radiation due to a change in concentration from C_0 to C. This trapped radiation is caused by ϵ and feedback effects such as cloud cover. The formula for this is;

$$\Delta F = Aln(C/C_0)$$

and the temperature change related to the radiative forcing is;

$$\Delta T = B\frac{T\Delta F}{S_o(1-\alpha)}$$

Steps below:

1. Set the constants
   - $A = 5.35$
   - $C_0 = 278ppm$ the reference concentration
   - $C = 400ppm$ the new ppm concentration
   - $B = 1.7$ 7 an estimated number determined by fitting data of CO2 concentrations versus temperature
   - $T = 288K$ rewritten variable of observed average temperature of Earth
2. Calculate the radiative forcing ΔF using the formula above
3. Calculate the relative temperature change ΔT using ΔF
4. Print these results

In [7]:
```
# task 4

# constants
A = 5.35
C_0 = 278 # reference concentration (parts per million ppm)
C = 400 # ppm of new concentration
B = 1.7 # constant
T = 288 # Observed average temperature of earth (Kelvin)

# calculating the radiative forcing from the formula (W/m^2)
delta_F = A * np.log(C / C_0)

# now caluclating the temperature change using this value (K)
delta_T = B * ((T * delta_F) / (S * (1 - e_albedo)))

# print the results
print('Radiative forcing caused by C02 crossing the 400 ppm milestone = {0:.2f} W/m^2 '.format(delta_F))
print('Temperature change caused by C02 crossing the 400 ppm milestone = {0:.2f} K '.format(delta_T))
```

```
Radiative forcing caused by C02 crossing the 400 ppm milestone = 1.95 W/m^2
Temperature change caused by C02 crossing the 400 ppm milestone = 0.99 K
```

Task five

The purpose of task five is to use real world data from the University of Galway's Mace Head research station to model the concentration of carbon dioxide over time.

To do this, the file containing the information about carbon dioxide, namely 'co_macehead.txt' was imported. You then split the file into the parts wanted, create a dictionary to store a key value pair of the year and all the carbon dioxide values. A more detailed step guide below.

1. Import matplotlib.pyplot as plt for plotting purposes
2. Open the file in read mode, and use readlines to get a list of all the lines in data
3. Iterate over these lines and display this list
4. Use another for loop to split the line into components using a comma as a delimiter. Alternativy a space would have just worked as well as a delimiter.

In [8]:
```
# task 5
import matplotlib.pyplot as plt

# open the relevent file in read mode
with open('co_macehead.txt', 'r') as file:
    # read all lines from the file
    lines = file.readlines()

    # print the lines to verify they are being read correctly
    for line in lines:
        print(line)

    # iterate over each line
    for line in lines:
        # split the line into components based on a comma delimiter
        components = line.split(',')  # Using comma as the delimiter

#         # print a sample of the components (e.g., first 3 components)
#         print(components[:0 : 1])  # Print the first 3 components

# # extracting time and CO2 concentration
# time = file[:, 0]
# co2_concentration = file[:, 1]
```

```
# number_of_header_lines: 68
# comment:
# comment:  ************ USE OF ESRL GMD DATA ***************
# comment:
# comment: These data are made freely available to the public and the
# comment: scientific community in the belief that their wide dissemination
# comment: will lead to greater understanding and new scientific insights.
# comment: The availability of these data does not constitute publication
# comment: of the data.  NOAA relies on the ethics and integrity of the user to
# comment: ensure that GML receives fair credit for their work.  If the data
# comment: are obtained for potential use in a publication or presentation,
# comment: GML should be informed at the outset of the nature of this work.
# comment: If the GML data are essential to the work, or if an important
# comment: result or conclusion depends on the GML data, co-authorship
# comment: may be appropriate.  This should be discussed at an early stage in
# comment: the work.  Manuscripts using the GML data should be sent to GML
# comment: for review before they are submitted for publication so we can
# comment: ensure that the quality and limitations of the data are accurately
# comment: represented.
# comment:
# comment: Every effort is made to produce the most accurate and precise
# comment: measurements possible.  However, we reserve the right to make
# comment: corrections to the data based on recalibration of standard gases
# comment: or for other reasons deemed scientifically justified.
# comment:
# comment: We are not responsible for results and conclusions based on use
# comment: of these data without regard to this warning.
# comment:
# contact_parameter: CO2 CCGG (Individual Flasks)
# contact_name: Ed Dlugokencky
# contact_telephone: (303) 497-6228
# contact_email: Ed.Dlugokencky@noaa.gov
# comment:
# comment:  ************ COLLABORATORS ***************
# comment:
# comment: NOAA thanks the following collaborators without
# comment: whom these measurements would not be possible.
# comment:
# collaborator_name: National University of Ireland, Galway [NUI]
# comment:
# comment:  ************ RECIPROCITY AGREEMENT ***************
# comment:
# comment: Use of these data implies an agreement to reciprocate.
# comment: Laboratories making similar measurements agree to make their
# comment: own data available to the general public and to the scientific
# comment: community in an equally complete and easily accessible form.
# comment: Modelers are encouraged to make available to the community,
```