Programming Project (2)

December 2020

The second programming project for Introductory Scientific Computing provides you with a choice between three options of which you should **only complete and submit one**. This will be worth 15% of your total unit mark. The deadline for this work is **Thursday, 17th December (10pm)**.

To use Noteable to complete your assessment, you should create a new Jupyter Notebook. After completion, the Notebook should be downloaded from Noteable and uploaded to the submission point. For a short demo, see the video available on the Blackboard "Access Python" page entitled "Create new notebooks and download files". You can use an alternative tool, if you prefer, as long as your file contains Python code only and can be run using a Python interpreter (".ipynb" or ".py" file or files).

The three options for this assessment are as follows:

1. **Ozone creation and destruction** - create a toy model of the ozone cycle in the stratosphere (1) by considering ozone creation, (2) and ozone destruction.
2. **Area under a curve** - approximate the area under a curve using (1) trapezium rule, (2) random sampling.
3. **Cooling object** - create a toy model for a hot object cooling (1) in an open system, (2) in a closed system.

These are outlined in more detail below. Please name your main submission file to include your name and the option you have chosen to complete:

NAME_Project_OptionNUM.ipynb (or .py)

for example

RachelTunnicliffe_Project_Option3.ipynb (or .py)

When completed, this should be submitted to the appropriate Marks, Attendance and Feedback submission point. This will be labelled as "Introductory Scientific Computing Assessments" with options for "Programming Project 2- Option 1", "Programming Project 2- Option 2" and "Programming Project 2- Option 3".

1 Ozone creation and destruction (Option 1)

1.1 Ozone creation

Ozone (O_3) is created in the stratosphere when an oxygen molecule (O_2) absorbs a photon of the correct frequency (UV photon $< 214\,\text{nm}$) and is split into two oxygen atoms (O) in a process called photolysis, as shown in reaction equation 1.

$$O_2 + h\nu \rightarrow 2O \tag{1}$$

Each one of these oxygen atoms (O) can then quickly recombine with another oxygen molecule (O_2) to form ozone (O_3) as shown in reaction equation 2.

$$O + O_2 \rightarrow O_3 \tag{2}$$

Here, we will build a very simple "toy" ozone layer using a random number generator.

Assume that during any one timestep (1 s):

- the probability that an oxygen molecule photolyses (Equation 1) = 0.0001
- the probability that an oxygen atom reacts with an oxygen molecule (Equation 2) = 0.1. The possible number of reactions in each step will be limited by the number of oxygen atoms available to react. The number of oxygen atoms will always be less than the number of oxygen molecules for these sets of probabilities.

Assessment (option 1, part 1): Write a model to simulate ozone creation from these two reactions: first the creation of the oxygen atoms and then the creation of ozone.

- Start with 10,000 oxygen molecules (O_2) only (i.e. no oxygen atoms or ozone molecules)
- Run the model for 10,000 time steps (where each time step is equivalent to 1 s.
- Model design:
  - Consider how to use the probabilities to decide whether a reaction has occurred
  - For each time step consider how many reactions could have occurred
  - For the reaction containing two inputs, consider which input will limit the number of potential reactions for each time step
- Create output variables for the number of oxygen molecules (O_2), number of oxygen atoms (O) and number of ozone molecules (O_3) for each time step.
- After the model has completed determine: 1) How many ozone molecules are there? 2) What is the total number of oxygen atoms contained in all three gas species? Print out these results for the last time step.

1.2 Ozone cycle

Ozone (O_3) is also destroyed in the atmosphere in one of two ways. The first is when the ozone molecule absorbs a UV photon and splits into an oxygen atom (O) and an oxygen molecule (O_2), as shown in reaction equation 3 (lower energy photon than for reaction equation 1).

$$O_3 + h\nu \rightarrow O + O_2 \tag{3}$$

The second is when an ozone molecule (O_3) reacts with an oxygen atom (O) to form two oxygen molecules (O_2) as shown in reaction equation 4.

$$O + O_3 \rightarrow 2O_2 \tag{4}$$

Assume that during any one timestep (1 s):

- the probability that an ozone molecule photolyses (Equation 3) = 0.01
- the probability that ozone destruction occurs (Equation 4) = 0.01. As with reaction equation 2, the number of oxygen atoms will always be less than the number of ozone molecules for these sets of probabilities.

Assessment (option 1, part 2): Update your model to incorporate these additional two reactions.

- Start with the same initial setup as part 1
- Create the same output variables as part 1 for each time step.
- After the model has completed: How many ozone molecules are produced now? Did the number of oxygen atoms in the system increase or decrease? What is the ratio of oxygen molecules to ozone molecules and how does this value change throughout the model? Print out these results for the last time step.

1.3 Optional: plotting your model

Whilst this part is not assessed, it may help you construct and debug your models: by modifying some of the matplotlib code that you have seen in the workshops, plot the evolution of each of the three gas species for your two models.

2 Area under a curve (Option 2)

Consider a function of the form, shown in Figure 1:

$$y = 10 + x^2 - 0.1x^3 \tag{5}$$

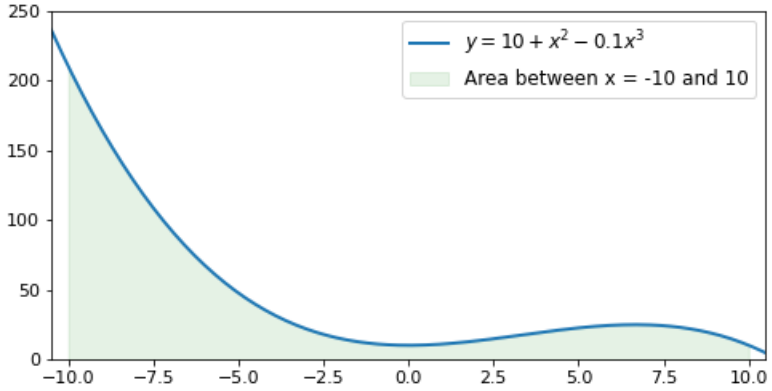

Figure 1: Plot of $y = 10 + x^2 - 0.1x^3$. The blue line shows the x-y relationship and the green shading shows the area underneath the curve for $x = -10$ to 10 (between the line and the x-axis).

2.1 Trapezium rule

One way to approximate the area under a curve, is to apply the *trapezium rule*. This is where a area can be split into regular trapezia and summed to approximate the total area as shown in Figure 2.

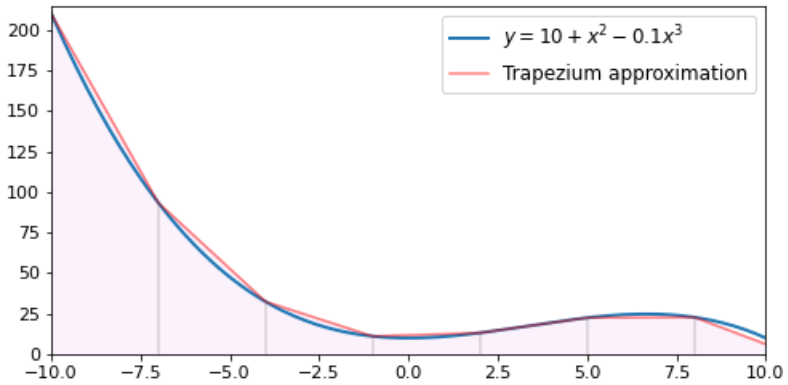

Figure 2: Using regularly spaced trapezia to approximate area the area under the curve.

The area of each trapezium is determined by Equation 6, with the parameters demonstrated in Figure 3.

$$A_{trapezium} = \frac{a+b}{2}h = \frac{y_0 + y_1}{2}(x_1 - x_0) \tag{6}$$

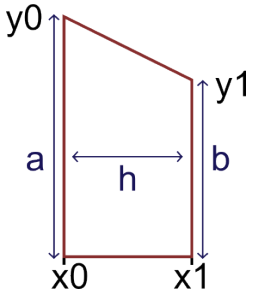

Figure 3: Parameters for determining the area of a (vertical) trapezium

Assessment (option 2, part 1): Approximate the area under the curve formed by Equation 5 using the trapezium rule.

- Calculate the area between $x = -10$ to 10 and from the curve to the x-axis (y=0) as shown in Figure 1.
- Model design:
  - Though it is possible to perform this calculation directly using the scipy integrate function, for this assessment we would like you to build the model using the tools you have encountered so far in the course.
  - Consider what steps to break this down into to find a good approximation of the true area.
  - Consider how to evaluate each step.
- Create an output variable containing the area for each trapezium.
- After the model has completed determine the total area. Print out this result.

2.2 Random sampling

Random sampling is the selection of a representative sub-sample from an overall population. By randomly sampling from a total area and determining how many of the total points fall within the area we want to calculate, this can be used to approximate the area under the curve as shown in Figure 4.

If you need a refresher on this topic, this was covered in the first Week 8 asynchronous activity where we looked at this in relation to approximating the area of a circle.

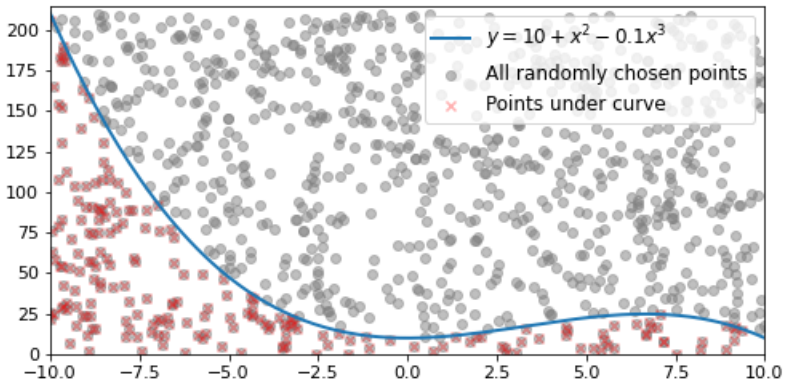

Figure 4: Using random positions to determine the ratio of points under the curve vs the total area.

Assessment (option 2, part 2): Approximate the area under the curve formed by Equation 5 using random sampling.

- Calculate the area with the same bounds as part 1 - between $x = -10$ to 10 and from the curve to the x-axis (y=0) as shown in Figure 1.
- Model design:
  - Consider how to create random positions that fall within the expected x, y range.
  - Consider how to calculate your total area and the fraction of that area under the curve.
  - Consider how many (x, y) positions you want to sample to find a good approximation to the true area.
- Create an output variable containing the number of random positions under the curve.
- After the model has completed determine the total area. Print out this result.
- If the true area under this curve is $866.\dot{6}$ - how do your two approximations compare to this value?

3 Cooling object (Option 3)

3.1 Losing heat

An object which is hotter than the ambient temperature will lose heat according to Newton's Law of Cooling given by Equation 7.

$$Q(t) = -hA\Delta T(t) \tag{7}$$

where:

- Q is the rate of heat transfer out of the body ($\mathrm{J\,s^{-1}}$)
- h is the heat transfer coefficient (constant, $\mathrm{J\,s^{-1}\,m^{-2}\,K^{-1}}$) and A is the heat transfer surface area ($\mathrm{m^2}$)
- $\Delta T(t) = T_{obj}(t) - T_{env}(t)$. This is the time-dependent temperature difference between the environment and object (K).

This energy change is related to a change in temperature which is governed by Equation 8.

$$Q_{thermal}(t) = mC_p\delta T(t) \tag{8}$$

where:

- $Q_{thermal}(t)$ is the rate of thermal energy change ($\mathrm{J\,s^{-1}}$)
- m is the mass (kg)
- C_p is the specific heat capacity ($\mathrm{J\,kg^{-1}K^{-1}}$)
- $\delta T(t)$ is change in temperature of the object per second ($\mathrm{K\,s^{-1}}$)

Note that $\Delta T(t)$ and $\delta T(t)$ represent different quantities in Equations 7 and 8. $\Delta T(t)$ is the difference in the temperature of the hot object compared to its surroundings whereas $\delta T(t)$ is how the temperature changes in response to an increase (or decrease) in thermal energy.

Assessment (option 3, part 1): Model the cooling of this hot object until it reaches the temperature of the environment (within 0.01° C or K).

- We will consider appropriate parameters (constants) for a copper sphere:
  - h (heat transfer coefficient) for this object is $8.7\,\mathrm{J\,s^{-1}\,m^{-2}\,K^{-1}}$
  - A (area) is $0.13\,\mathrm{m^2}$
  - m (mass) of the object is $0.1\,\mathrm{kg}$
  - C_p (specific heat capacity) is $385\,\mathrm{J\,kg^{-1}\,K^{-1}}$.
- The initial temperature of the object is $40\,^{\circ}\mathrm{C}$ (equivalent to $313.15\,\mathrm{K}$)
- The temperature of the environment is $20\,^{\circ}\mathrm{C}$ (equivalent to $293.15\,\mathrm{K}$) and we can approximate that this will not change throughout the simulation.
- Run the model for however many model steps are necessary to reach the stated condition, using time steps of $1\,\mathrm{s}$.
- Model design:
  - Consider how to break this down into steps and how to calculate the energy lost in each time step
  - Consider how to convert this energy loss into a temperature loss
- Create output variables containing (1) the temperature of the hot object for each time step and (2) the *difference* in temperature between the object and the environment for each time step.
- After the model has completed determine: how many seconds did it take for the hot object to cool by half the original temperature difference (i.e. cool by 10 degrees)? Print out this result.

3.2 Closed system

Now consider that this object is within a closed box where the box itself does not lose or gain any energy (closed system) as shown in Figure 5. As the hot object cools down (loses heat), the thermal energy will transfer to the surrounding air and heat this by a small amount in each time step.

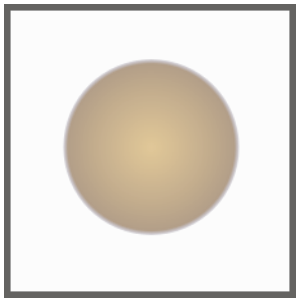

Figure 5: Hot object within a closed system

Assessment (option 3, part 2): Update the model to include the energy lost from the cooling body resulting in a temperature increase in the surrounding air. Run this model until the body and the surrounding air (environment) are the same temperature (have reached equilibrium, within 0.01° C or K).

- Use the same input parameters as part 1
- For the surrounding air use the following parameters:
  - m (mass) of the surrounding air is 0.6 kg
  - C_p (specific heat capacity) is 1000 $\mathrm{J\,kg^{-1}\,K^{-1}}$.
- As before, run the model for however many model steps are necessary to reach the stated condition, using time steps of 1 s.
- Create output variables containing (1) the temperature of the hot object, (2) the temperature of the environment and (3) the difference in temperature between the body and environment and for each time step.
- After the model has completed determine: how many seconds did it take for the hot object to cool by half the original temperature difference (i.e. cool by 10 degrees) this time? Print out this result.

3.3 Optional: plotting your model

Whilst this part is not assessed, it may help you construct and debug your models: by modifying some of the matplotlib code that you've seen in the workshops, plot the evolution of temperature and temperature difference in both parts. For the second part, it may help to plot the temperature of the environment as well.