You will find, in this file, guidelines for the assignment of the computational modeling course. The assignment is due by:

Friday May 3rd, 2019

We provide an ample deadline, already taking into account that access to the computer room will not be available for all students at all time, due to travel, illness and other limiting factor. We have hence already extended the deadline, without planning to extend it any further. If you have personal circumstances, you will need to submit a "Mitigating Circumstance Form" to the official channel at KCL (please copy the lecturers for information) with ample time before the deadline.

You will have to submit a Python Notebook, containing the code and discussion (report), on KEATS.

There is no length limit for your Python Notebook.

The lecturers and the teaching assistants will not provide any guidance to solve the problems enclosed in the report.

Students are welcome to discuss the problems between them, but **PLAGIARISM will not be tolerated**.

Python Notebooks need to be submitted online before May 3rd, 2019 23:59 , reports submitted after the deadline will not be considered and you will get a zero mark, so please allow for plenty of time to submit your report. **Remember that submitting at the last moment might put you at risk of unexpected events** (internet connection, illness, private problems…), none of those will be considered as an excuse, since you can submit at any time **BEFORE** the deadline. A section on KEATS will be opened soon to allow for your online submissions.

We first provide a set of two problems that you will need to solve by extending the Python programs that you wrote during the term. The various subsections of problem 1 and 2 will guide you to obtain the final code for each problem. It will also allow you to generate the results required for the report. The second part "guidelines to write your report" will guide you how to write your report and include/discuss the obtained data. Discussion can be included in the report via text cells, and the Python graphic tools can be used to generate and plot data.

We will aim to mark reports by the end of May, marks will be provided via KEATS.

The marking scheme is outlined below. Note that the code (Python cells) will also be marked, so please follow good coding practices outlined in the lecture (comment the code). Note that **all cells** should execute properly **without error messages** given by Python. Markers will disregard cells that don't execute properly.

Problem 1: The expanding civilization

We study here the sequence given by:

$$x_{n+1} = a * x_n * (1 - x_n)$$

with the initial condition x_0 chosen in the interval [0,1], and "**a**" is a given parameter between 0 and 4.

This sequence is a simple model to describe how a population (or civilization) evolves in a closed system with a finite amount of resources. The parameter "**a**" describes the rate of development and expansion of the civilization. The population is described by a simple variable x, which quantifies the total population (when x=0 the population is naught, when x=1 the population has reached its maximum sustainable size).

The civilization grows with the combined rate of reproduction (proportional factor **x** in the sequence) and starvation (term **1-x**). Indeed, since the resources are limited the expansion is also proportional to "**(1-x)**" and if the civilization reaches the maximum population **x**=1 it will have exhausted all its resources and then collapses (**x**=0).

1. Write a program by using a "for loop" to compute the convergence point of the sequence for a=1, and a random number between 0 and 1 for x_0. We assume in this problem that the sequence reaches the converge point with n=100 sequence elements, so the convergence point x* can be approximated by the n=100 element of the sequence (x_{100}).

2. Store in an array the value of "a" and the obtained convergence point x*

3. Add a *for loop* to repeat the same operation for 50 different random initial conditions. All obtained pairs of data points (a, x*) should be stored in a NumPy array

4. Add one more *for loop* to repeat the same procedure (1-3) for N=2'000 different rate parameters "**a**" spanning the interval [1.5,4]. This means that "a" needs to be equal to a=1.5 for the first cycle of your additional "for loop", and be equal to "*a* = 4" for the last cycle of this additional "*for loop*".

5. Plot the result (with the matploblib module) obtained for the convergence points x* (or x_{100}) as a function of the parameter ***a***. Note: if you are out of memory, or Python struggles with the amount of requested information, decrease N=2'000 to a smaller number, N=100 or N=1000.

6. You should have obtained a graphic of the fix points x* of the sequence x_n obtained for different growing rate **a**. Repeat the same procedure (4-5) with a range of **a** in [3,4] and then [3.5,3.8].

Problem 2: Monte Carlo integration

1) We first want to compute the integral of $\sin^4(x)$ between $x = 0$ and $x = \pi / 2$. Write a Monte Carlo code to obtain the integral of $\sin^4(x)$ by using N=10,000 random points. Find a strategy to define the adequate framing "box". For the example of the area of the circle we were using a framing square, you need to adapt the same strategy to estimate the area contained under the function $\sin^4(x)$.

2) We now want to obtain the volume contained between a sphere of radius 0.75 and another larger sphere of radius 1. Modify the Monte Carlo written for the volume of a sphere. Use N=10,000 sampling points (random points in the cube).

Guidelines to write your Python Notebook

The structure of the Python Notebook should be as follows:

• title: find a title for the report, which describes what you are presenting in this report [2 marks]

• Introduction: a short summary of what you are going to address in this work. For example, that you are using a computational approach based to tackle a variety of common physics problems, such as sequences and integrals with Monte Carlo. Give a short motivation why programming (and why Python) is useful to tackle the type of problems solved here, and how this fits with the rest of your studies. Typically a few paragraphs (3-4) are expected here. [6 marks]

• Problem 1 : Sequence. See below for what to include in this section. [17 marks]

• Problem 2 : Monte Carlo integration. See below for what to include in this section. [18 marks]

• Conclusion : write in a couple of paragraphs what you learnt from this course, and highlight/summarize the main results obtained above. Provide ideas for future work (outlooks), explain why the results above are significant and why was the research carried out above worthwile [8 marks]

• Coding : Code presentation, comments, no mistakes, relevant Python module management (correctly loaded), the cell execute without any crash [6 marks]

Total marks: 57

The detailed content of the sections related to Problems 1 & 2 is detailed thereafter.

Section Problem 1:

1. Explain briefly what is your approach to generate a sequence, how it is realized within the general structure of a computer code. You can include this discussion after, or before the relevant programming cell.
2. Discuss now which are the different "for-loops" involved in the code, and what they achieve, what is their purpose?
3. Using the code you have written, obtain a Python graph which shows on the horizontal axis the growing rate "**a**" and on the vertical axis the obtained fix points "**x***" for the set of initial random x_0 values. Show one graphic with **"a"** ranging from 1.5 to 4, and then also with "**a**" ranging from 3 to 4, and 3.5 to 3.8. Show the three plots in your notebook.
4. For **a**=1.5 there is only one fix point. For which values of a do you obtain more than one fix point?
5. Discuss the observed trend as "a" is increasing, and the consequence of the accumulation of fix points for the stability of the growing civilization.
6. The plots that you obtain in (3) have similar structures, this is called self-similarity. Do a quick literature search to discuss this concept and its connection with the sequence discussed in this problem.

Section Problem 2:

1. Summarize the main ideas of the Monte Carlo algorithm used to compute areas or volumes.
2. Discuss in this context how you modified the algorithm used in the lecture for the area of the circle, to achieve the integral of the $\sin^4(x)$ function.
3. Show the obtained estimate of the integral of the $\sin^4(x)$ function with N=10,000 trial points, and compare the estimate with the exact answer (compute the integral with pen and paper). Find in the literature what is the error of Monte Carlo, in terms of scaling with the number of points N. Does your obtained difference, between the Monte Carlo and the exact result, match with the typical error expected for a Monte Carlo algorithm?
4. Explain now your strategy to compute the volume contained between two spheres. Compare the Monte Carlo obtained result with the analytical result (obtained with pen and paper).
5. Compare the error obtained in the points (4) and (3) above, discuss if they should be different or similar, and explain your reasoning.
6. For the integral of the volume contained between two spheres, would a method using a regular mesh of points be easier to implement? Discuss in your view the advantages and drawbacks of the Monte Carlo algorithm in terms of its efficiency, and its implementation. For which type of integrals is the Monte Carlo evaluation particularly well suited?