# PH4419/PAP723: Computational Physics

Assignment 3

**INSTRUCTIONS**

- *Submission:* Assignment is due: **12th April 2019**.

- *Submission format:* All submission are to be done through NTULearn. There are 3 (three) questions in this assignment. Answer each question in a separate **python .py** file. Label your files using the following convention:

  **Matric Number_Assignment 3_Question X.py**

  For example: U1234567A_Assignment 3_Question 0.py

  Each file should begin with the lines

```
from scipy import *
import matplotlib.pyplot as plt
```

  Any other Python libraries used must be listed directly below these two lines.

- *Grading:* For full credit, code must follow good programming style. Key code blocks must be clearly commented. Function names must be properly named for ease of understanding your codes. The program structure should be modular; avoid unnecessary code duplication, and group numerical constant definitions neatly together. The output of your programs should be clear. Each generated plot should be well labeled, and if multiple curves are shown, they should be well-labeled.

- *PAP723:* Parts of questions marked with a $(*)$ are for students taking the course PAP723, they must be attempted to receive credit. Undergraduate students who can produce a working code with correct output for these parts will receive 1 (one) bonus mark per question.

- *Documentation:* Whenever the assignment mentions a Scipy function you might need, consult the Scipy/Numpy online documentation to learn what the function does and how to use it.

- *Techniques:* The core techniques you will need to apply in this problem set are: (i) Random numbers, (ii) Monte Carlo Simulation and (iii) Metropolis Algorithm

### 00. Fractals and Pseudo Random Numbers [5 marks]

Chaos theory represents a large class of dynamical systems in our physical world where seemingly probabilistic actions lead of unpredictable behavior. The beauty of dynamical system is that even in the midst of using probabilistic models, there exist some interesting results contrary to intuition. One such example is the formation of fractals. Consider the following transformation:

$$f_i(x, y) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ g \end{bmatrix} \tag{1}$$

The constants are determined as such

- for $i = 1$: $a = 0.0$, $b = 0.0$, $c = 0.0$, $d = 0.16$, $e = 0.0$ and $g = 0.0$
- for $i = 2$: $a = 0.85$, $b = 0.04$, $c = -0.04$, $d = 0.85$, $e = 0.0$ and $g = 1.6$
- for $i = 3$: $a = 0.2$, $b = -0.26$, $c = 0.23$, $d = 0.22$, $e = 0.0$ and $g = 1.6$
- for $i = 4$: $a = -0.15$, $b = 0.28$, $c = 0.26$, $d = 0.24$, $e = 0.0$ and $g = 0.44$

A random number $p$ is drawn from a uniform distribution $[0, 1]$. For 1% of the time, $i = 1$ is chosen, that is, the point $(x_n, y_n)$ is mapped to $(x_{n+1}, y_{n+1})$ under the transformation $f_1$ as such

$$\begin{aligned} x_{n+1} &= 0 \\ y_{n+1} &= 0.16 y_n \end{aligned}$$

85% of the time, $i = 2$ is chosen, and 7% of the time, $i = 3$ and $i = 4$ are each chosen. Starting from $(x, y) = (0, 0)$, write a function

| `def fractal(xmin, xmax, ymin, ymax, N):` | |
|---|---|
| Input | |
| `xmin, xmax, ymin, ymax` | The domain of the plot. Take $-2.2 \leq x \leq 2.7$ and $0 \leq y \leq 10$. |
| `N` | An integer, it is the number of iterations to take. Set `N` $= 100000$ |

The function has no output value, but it returns a plot that marks a point for each of the $(x_n, y_n)$, showing the result by following the above rules (you are allowed to use pseudo-random numbers).

### 01. Radioactive Decay Chain [7 marks]

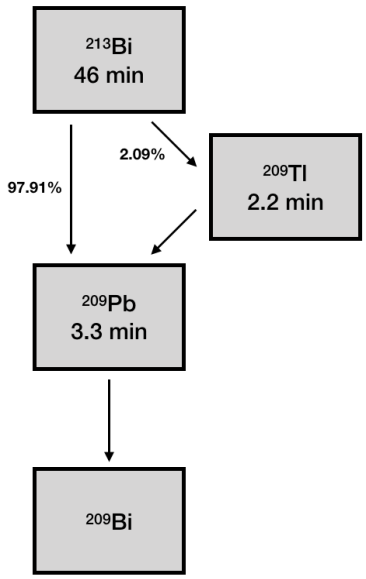
The isotope $^{213}$Bi decays to stable $^{209}$Bi via one of two different routes, with probabilities and half-lives as shown in the figure below. (Technically, $^{209}$Bi is not really stable, but it has a half-life of $10^9$ years, a billion times the age of the universe, so it might as well be considered as stable.) The probability $p(t)$ that any particular isotope, with half-life $\tau$ has decayed is

$$p(t) = 1 - 2^{-t/\tau} \tag{2}$$

Write a program that starts with a sample consisting of 10 000 isotopes of $^{213}$Bi, simulate the decay of the isotopes by dividing time into slices of duration `dt` = 1.0 s each and on each step, your code should perform the following:

- For each isotope of $^{209}$Pb in turn, decide at random, with the appropriate probability, whether it decays or not. Count the total number that decay, subtract it from the number of $^{209}$Pb isotopes, and add it to the number of $^{209}$Bi isotopes.
- Do the same for $^{209}$Tl, except that the decaying isotopes are subtracted from the total for $^{209}$Tl and added to the total for $^{209}$Pb.
- For $^{213}$Bi, the situation is slightly more complicated: when a $^{213}$Bi isotope decays you have to decide at random with the appropriate probability the route by which it decays. Count the numbers that decay by each route and add and subtract accordingly.

Note that you have to work up the chain from the bottom as described, not down from the top. Keep track of the number of isotopes at each time step. Your code should output a graph showing the composition of each isotope after 10000 seconds.




In your code comments, briefly describe the result you obtained with reference to the half-lives, and explain why you have to work from the bottom of the chain up, instead of top to down.

## 02. 2D Ising Model [8+5 marks]

A 2D Ising model consists of $N$ spins arranged in a square lattice. Let $S_i$ denote the spin at site $i$, which can have value $+1$ or $-1$. The system is in a state specified by the values of all the spins, $\{S_i\}$. Assuming no external magnetic field, the total energy is

$$E = -J \sum_{\langle ij \rangle} S_i S_j \tag{3}$$

where $\langle ij \rangle$ denotes pairs of nearest-neighbour $i$ and $j$ sites (note that $\langle 12 \rangle$ and $\langle 21 \rangle$ are the same pairs, so their energies should not be counted twice). Apart from the energy, we are also interested in the total magnetization

$$M = \frac{1}{N} \sum_i S_i \tag{4}$$

To simplify the problem, we will assume that the lattice has periodic boundary conditions: e.g., if the lattice is $N_x \times N_y$, then site $(0, 0)$ is neighbours with sites $(1, 0)$, $(N_x - 1, 0)$, $(0, 1)$ and $(0, N_y - 1)$

(a) **[8 marks]** Write a function

| `def ising_mc(J=1., Nx=16, Ny=16, nsteps=50000):` | |
|---|---|
| Input | |
| `J` | Value of the parameter $J$ in the Ising model |
| `Nx, Ny` | The values of $N_x$ and $N_y$, the number of sites in the $x$ and $y$ directions respectively. |
| `nsteps` | The total number of steps per Monte Carlo simulation. |

The function has no output value, but it should perform Monte Carlo simulations of the 2D Ising model at various temperatures $T$. You may hard-code an appropriate range for $T$ (a good choice of $T$ is between $1/J$ and $10/J$), here we have set the Boltzmann constant to be $k_B = 1$. The function should then produce the following plots:

- $\langle E \rangle$ versus $T$
- $\langle M \rangle$ versus $T$.
- The heat capacity $C_B$ versus $T$. The heat capacity is defined as $C_B = \left[\langle E^2 \rangle - \langle E \rangle^2\right]/T$.
- The magnetic susceptibility $\chi_m$ versus $T$. The magnetic susceptibility is defined as $\chi_m = \langle M^2 \rangle - \langle M \rangle^2$

Simulations should be performed using the Metropolis algorithm. When calculating averages, you may wish to discard some early steps to let the Markov chain settle towards the stationary distribution. Here is the outline to the Metropolis Algorithm:

1. Prepare an initial configuration of $N_x \times N_y$ spins
2. Flip the spin of a randomly chosen lattice site.
3. Calculate the change in energy $dE$.

4. If $dE < 0$, accept the move. Otherwise accept the move with probability $e^{-dE/T}$. This satisfies the detailed balance condition, ensuring a final equilibrium state.
5. Repeat 2-4.

During each Monte Carlo step, you are required to calculate total energy $E$ for the whole lattice. There are 2 ways that you might want to consider calculating the total energy $E$ (you may decide which is more efficient)

1. Recalculate the total energy $E$ by calculating the energy for the whole $N_x \times N_y$ lattice.
2. The energy difference resulting from flipping spin $i$, which can be determined using the values of the neighbouring spins. Write out all the possible neighbour configuration and observe a pattern in the change in energy. Knowing the change of energy is sufficient information to determine the total energy of the system.

(b) **[5 mark]** (*) Set $T = 2.0/J$. Explain the difference if you started your simulation with an ordered initial state (i.e. all $S_i = +1$) versus if you started your simulation with a random initial state. If there is no difference, explain why.

**[You will be graded on the efficiency on your code.]**