Problem Set 4

- *Submission format*—Submit your solution via NTULearn, with **one Python source file per problem (`.py` not `.ipynb`)**. Each file should begin with the lines

  ```
  import numpy as np
  import scipy
  import matplotlib.pyplot as plt
  ```

  You may also import other Scipy modules or standard Python modules, but do not use non-standard modules (e.g., stuff from PyPI).

- *Coding assistants*—AI coding assistants may be used if you want, though all assignments can be completed without them. If you use an AI assistant, you must carefully scrutinize, test, and fix up the code it generates. But do not perform verbatim copying of other people's work, including other students taking this course; that is treated as plagiarism.

- *Code quality*—For full marks, the submitted programs must meet these criteria:

  1. Keep all other top-level code, aside from import statements and function/class definitions, to a minimum. Most of your code should live inside functions. Your program should work if we `import` it as a module and call its functions.
  2. If the assignment asks you to write a function, follow the specification exactly. Do not modify the order or definitions of the inputs/outputs to suit yourself. You are free to define additional helper functions, classes, or data structures.
  3. Use comments to document important code blocks. In particular, if you define helper functions or class methods, document their inputs/outputs clearly.
  4. Avoid cryptic function or variable names like `abc`.
  5. Nontrivial numerical constants should be grouped appropriately (e.g., the start of a function), not "hard-coded" deep inside code blocks.
  6. Functions should run appropriate "sanity checks" (e.g., using `assert`) on inputs.
  7. Every generated plot should be properly formatted, with appropriate axis ranges, a proper figure title and axis labels, a legend if there are multiple curves per graph, etc.

0. 2D ISING MODEL (15 MARKS)

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

$$E = -J \sum_{\langle ij \rangle} S_i S_j, \tag{0}$$

where $\langle ij \rangle$ denotes pairs of nearest-neighbour i and j sites (without double-counting pairs). Apart from the energy, we're also interested in the total magnetization

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

We will assume the lattice has periodic boundary conditions: e.g., if the lattice is $N_x \times N_y$, then site $(0, 0)$ is neighbours with the 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):`	
Inputs	
`J`	Value of the J Ising model parameter.
`Nx`, `Ny`	The values of N_x and N_y, i.e. the number of sites in the x and y directions respectively.
`nsteps`	The total number of steps per Monte Carlo simulation.

The function 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 is between $1/J$ and $3.5/J$); the Boltzmann constant shall be normalized to $k_B = 1$. It should then produce the following plots:

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

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. Note: in each Monte Carlo step, it should not be necessary to re-calculate the total energy E for the whole lattice. In comments, discuss your findings from running the code.

(b) (7 marks) "Re-weighting" is a method for using a Monte Carlo trajectory, performed with temperature T_0, to get information about the system's behavior at a different temperature T_1. Suppose $A(E)$ is some function of the total energy; using properties of the partition function, one can show that the mean value of A at temperature T_1 is

$$\big\langle A(E)\big\rangle_{T_1} \approx \frac{\Big\langle A(E)\, e^{-(\beta_1-\beta_0)E}\Big\rangle_{T_0}}{\Big\langle e^{-(\beta_1-\beta_0)E}\Big\rangle_{T_0}}, \tag{2}$$

where $\beta_n \equiv 1/T_n$ and $\langle \dots \rangle_{T_0}$ denotes an average taken at temperature T_0. The two $\langle \dots \rangle_{T_0}$ quantities on the right-hand side of Eq. (2) can be computed during a Monte Carlo run at temperature T_0. Thus, we can estimate $\langle A(E)\rangle_{T_1}$ without having to do a separate Monte Carlo run at temperature T_1. In fact, a single Monte Carlo run can provide the data for many different temperatures at once.

Modify your code from part (a) into a new function, `ising_mc_reweight`, to use this technique (you can copy the initial code over; no need to worry about code duplication). In this new function, modify the plot of C_B versus T so that it includes the heat capacity estimated via re-weighting from a single Monte Carlo run. A good choice for this run is $T_0 \sim 2.4/J$, but you can experiment with other values. Label the various curves clearly.

1. MODEL FITTING BY BAYESIAN PARAMETER ESTIMATION (15 MARKS)

The Λ-CDM model describes the cosmos as a spacetime populated by a mix of dark energy and (ordinary + dark) matter. It predicts that the universe expands with time via the equation

$$H(z) = H_0 \left[\Omega_m (1+z)^3 + \Omega_\Lambda + (1-\Omega_m-\Omega_\Lambda)(1+z)^2\right]^{1/2}. \tag{3}$$

Here, the Hubble parameter $H(z)$ quantifies the rate of expansion, and the dimensionless redshift $z \geq 0$ quantifies the cosmological time ($z = 0$ is today, and larger z means earlier times). The model has three parameters: (i) the Hubble constant H_0, (ii) the matter density Ω_m, and (iii) the dark energy density Ω_Λ. Both Ω_m and Ω_Λ are dimensionless numbers representing the proportions of matter and dark energy relative to the total energy in the universe (if the universe is approximately flat, as astronomical observations strongly imply, $\Omega_m + \Omega_\Lambda \approx 1$).

$H(z)$ can be measured at different z's by many different kinds of astronomical observations. The attached file `hubble-data.csv` contains 38 measurements collected by a recent paper [1]. The file's first line is a comment, and each subsequent line n contains three comma-separated numbers

$$z_n, H_n, \sigma_n$$

where z_n is the redshift, H_n is the measured Hubble parameter, and σ_n is the uncertainty. Both H_n and σ_n are reported in units of km/second/megaparsec. From the data, we aim to estimate the distributions for H_0, Ω_m, and Ω_Λ (and hence their "best fit" values, uncertainty, etc.).

In Bayesian statistics, the probability for parameters $X = (H_0, \Omega_m, \Omega_\Lambda)$, conditioned on the data $D = \{z_n, H_n, \sigma_n\}$, is given by

$$P(X|D) \propto P(D|X)\, P(X). \tag{4}$$

The "likelihood function" $P(D|X)$ is the probability of finding D given X; we shall assume the errors are gaussian-distributed, so

$$P\Big(D = \{z_n, H_n, \sigma_n\} \,\Big|\, X = (H_0, \Omega_m, \Omega_\Lambda)\Big) = \prod_n \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left[-\frac{\big(H(z_n) - H_n\big)^2}{2\sigma_n^2}\right]. \tag{5}$$

The prior distribution, $P(X)$, is up to you to decide; normally, one chooses uniform distributions with reasonable ranges.

The distribution given by Eq. (4) can be sampled using the Markov Chain Monte Carlo (MCMC) method. This involves simulating a random walk in the parameter space of $X = (H_0, \Omega_m, \Omega_\Lambda)$. At each step, you will do the Metropolis decision step:

1. Calculate $P(X_{\rm new}|D) = e^{-E_{\rm new}}$ and compare it to $P(X_{\rm old}|D) = e^{-E_{\rm old}}$.
2. If $E_{new} \leq E_{\rm old}$, accept the move.
3. If $E_{new} > E_{\rm old}$, accept the move with probability $e^{-(E_{\rm new} - E_{\rm old})}$. Otherwise, stay at $X_{\rm old}$.

The Monte Carlo trajectory through the X space can then be used to estimate $P(X|D)$.

(a) (2 marks) First, write a function to load the `hubble-data.csv` file:

`def load_hubble_data(fn="hubble-data.csv"):`	
Inputs	
`fn`	The CSV file to load (defaults to `hubble-data.csv`). The first line is a header line which will be ignored. Subsequent lines have the form z_n, H_n, σ_n.
Return values	
`z`, `H`, `sigma`	1D arrays storing z_n, H_n, and σ_n respectively.

(b) (8 marks) Write a function to perform a Markov Chain Monte Carlo run:

`def LCDM_mc(z, H, sigma, nsteps):`	
Inputs	
`z`, `H`, `sigma`	1D arrays storing the observational data z_n, H_n, and σ_n.
`Nx`, `Ny`	The values of N_x and N_y, i.e. the number of sites in the x and y directions respectively.
`nsteps`	The number of steps to simulate.
Return values	
`H0`, `Om`, `OL`	1D arrays storing the values of the Λ-CDM parameters H_0, Ω_m, and Ω_Λ over the Monte Carlo trajectory.

You should include code comments documenting all relevant details, including your choice of the prior distributions, how the random walk is executed, and how X is initialized.

Hint—When doing the Monte Carlo steps, you may find it more convenient to work using the log-likelihood rather than the likelihood.

(c) (5 marks) Write a function, `LCDM_estimates()`, which takes no inputs and produces these plots:

- A histogram for the distribution of the Hubble constant H_0, with a text label stating the mean estimate and uncertainty.
- A histogram for the distribution of the matter parameter Ω_m, with a text label stating the mean estimate and uncertainty
- A histogram for the distribution of the dark energy parameter Ω_Λ, with a text label stating the mean estimate and uncertainty.
- A heat map showing the joint probability estimates for Ω_m and Ω_Λ.
- A heat map showing the joint probability estimates for H_0 and Ω_m.

Hint—You may wish to discard the first few steps in the Monte Carlo trajectory as unrepresentative. If you do so, document your choice clearly.

---

[1] O. Farooq1, F. R. Madiyar, S. Crandall, and B. Ratra, Hubble parameter measurement constraints on the redshift of the deceleration–acceleration transition, dynamical dark energy, and space curvature, Astr. J. **835**, 1 (2017).