PHYS4070 Assignment 2

Ryan White

30th of April, 2024

Part 1 – Numerical Solution of a Simple N-Body Problem

Part 1.1 – Two-Body Approximation

To begin, we're interested in integrating ODEs in a dynamics context to model the trajectory of an orbiting particle. The ultimate goal is to simulate an Planet-Projectile-Moon system such that we can manoeuvre the projectile into a stable orbit of the Moon, all done within our simple C++ script.

Our initial goal is to calculate the necessary vertical speed of a projectile such that it has zero velocity at the distance of the Moon surface. Since energy is conserved, we can 'drop' the particle from the position of the Moon's surface: the resulting velocity at the planet surface will be the required velocity to shoot it up to just graze the Moon surface. Our problem is then to find the velocity given

$$\ddot{r} = \frac{m_j}{|r_{ij}|^2}\hat{r}_{ij} \tag{1}$$

Since the problem can be defined in a 1-dimensional coordinate system, we can take the motion just in the x direction, giving

$$\ddot{x} = -\frac{m_{\text{pl}}}{x^2} = a(x) = \frac{dv(x)}{dt} \tag{2}$$

$$= \frac{dv(x)}{dx}\frac{dx}{dt} \tag{3}$$

$$= \frac{dv(x)}{dx}v(x) \tag{4}$$

and so

$$\int_{v_0}^{v} v(x)\ dv(x) = \int_{x_0}^{x} a(x)\ dx \tag{5}$$

Defining $x_0 = 18.75R_{\text{pl}} = 18.75$ as the location of the Moon surface relative to the planet CoM (since the Moon is orbiting at a distance of 19 units and has a radius of 0.25), and an initial velocity $v_0 = 0$, we want to integrate to $x = 1R_{\text{pl}} = 1$ and find v:

$$\left[\frac{1}{2}v(x)^2\right]_{v_0}^{v} = \int_{18.75}^{1} -\frac{1}{x^2}\ dx \tag{6}$$

$$\frac{1}{2}v^2 = \left[\frac{1}{x}\right]_{18.75}^{1} \tag{7}$$

$$= 1 - \frac{1}{18.75} \tag{8}$$

$$\Rightarrow v \simeq 1.376 \tag{9}$$

and so the required upwards velocity at the surface to launch the projectile to the surface of the Moon is about 1.376. Given that, for this non-dimensionalised system, the escape velocity is $\sqrt{2} \simeq 1.414$, this velocity is significant. We then implemented the 4th order Runge-Kutta integrator to numerically calculate the dynamics of the system (integrating over position and velocity). The relevant code for this is in `integrator.hpp` in the submission, where we perform each N-body step with the RK4 method in the function `Nbody_step()` which in turn calculates the gravitational potential from the function `ODEs_vector()`.

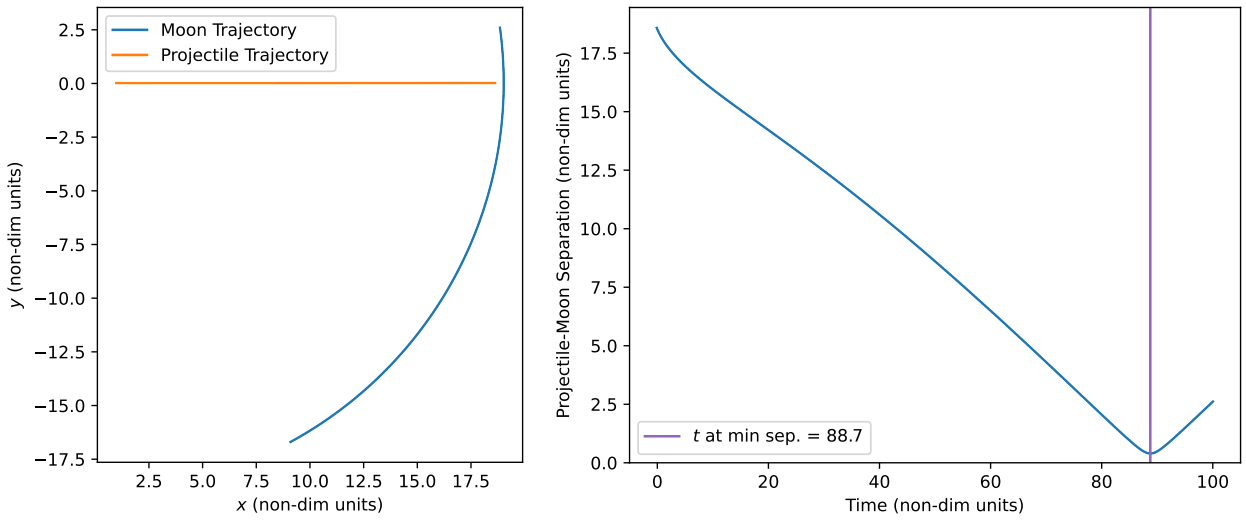

Figure. 1 We show here the dynamics of the (massless) Moon and projectile given an integration time of 100 non-dimensionalised units with a step size of 5×10^{-4} with our RK4 implementation. *Left:* The trajectories of the Moon and projectile in the planet-centered coordinates. We see the moon begin far away from the planet at (0, 0) and launched projectile, but get closer to the slowing projectile as it moves in its circular orbit. *Right:* The Moon-projectile separation shows a minimum at 88.7 time units as expected from our calculations.

The RK4 method works essentially the same as an Euler integrator, where a parameter on the next time step is equal to its current value, plus the timestep times its derivative:

$$x_{i+1} = x_i + \frac{dx_i}{dt}\Delta t \tag{10}$$

The strength in the RK4 method arises from using this method iteratively for values between 0 and Δt,

$$\mathbf{k}_1 = \mathbf{f}\left(\mathbf{X}_i,\ t_i\right) \tag{11}$$

$$\mathbf{k}_2 = \mathbf{f}\left(\mathbf{X}_i + \frac{\mathbf{k}_1 \Delta t}{2},\ t_i + \frac{\Delta t}{2}\right) \tag{12}$$

$$\mathbf{k}_3 = \mathbf{f}\left(\mathbf{X}_i + \frac{\mathbf{k}_2 \Delta t}{2},\ t_i + \frac{\Delta t}{2}\right) \tag{13}$$

$$\mathbf{k}_4 = \mathbf{f}\left(\mathbf{X}_i + \mathbf{k}_3 \Delta t,\ t_i + \Delta t\right) \tag{14}$$

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right)\Delta t \tag{15}$$

where equation (15) is analogous to equation (10). The function $\mathbf{f}(\mathbf{X}_i,\ t_i)$ represents the derivative of the parameter (so for position, the function returns the velocity, and for velocity the function returns acceleration). The notation is boldface above to show that we integrate over both position and velocity, where our acceleration at each step is analytically determined from the gravitational equations of motion.

When the projectile was launched along the x-direction from the planet surface at an instantaneous velocity of 1.376, we find that the projectile reaches a maximum distance from the planet surface of 18.6196 at a time of approximately 88.7 non-dimensionalised units (see right side of Fig 1). This maximum distance doesn't exactly coincide with the position of the moon surface, and I attribute this to numerical error due to the step size of 5×10^{-4} (which could be made arbitrarily small without a limit on the output size). The energy (in non-dimensionalised units) of the system at initialisation was calculated as -5.3312×10^{-26} and was -5.3707×10^{-26} after integration – a change of about 0.74%; the error in the maximum distance is correspondingly $\sim 0.69\%$ and of the same magnitude as the energy error.

Now we turn our focus to enforcing a projectile-moon encounter in the system (while keeping the Moon mass at 0 so as to not affect the dynamics). To determine the initial conditions of the Moon that results in a projectile encounter at its apogee, we first need to calculate some system parameters of the Moon. The circular velocity and

orbital period are calculated as

$$v_c = \sqrt{\frac{GM}{r}} = \sqrt{\frac{m_{\rm pl}}{R_{\rm orb}}} = \sqrt{\frac{1}{19}} \simeq 0.2294 \tag{16}$$

$$T = 2\pi\sqrt{\frac{a^3}{GM}} = 2\pi\sqrt{\frac{R_{\rm orb}^3}{m_{\rm pl}}} = 2\pi\sqrt{19^3} \simeq 520.368 \tag{17}$$

Given that we found the projectile maximum distance occurs at 88.7 time units, we want the Moon to be $88.7/520.368 \simeq 17.04\%$ behind in its orbit. That is, we want the moon to be at $2\pi(1-0.1704) \simeq 1.6592\pi$ radians with respect to the x axis at its initial position if take our coordinate system such that the Moon is orbiting counter clockwise.

The position of the Moon given an angular coordinate will be

$$\mathbf{r}_{\rm moon} = \{R_{\rm orb}\cos\theta,\ R_{\rm orb}\sin\theta\} \tag{18}$$

and so our initial position is (found by substituting $\theta = 1.6556\pi$) $\mathbf{r}_{\rm init} = \{9.111, -16.672\}$. Similarly, the velocity vector of the Moon given an angular coordinate will be

$$\dot{\mathbf{r}}_{\rm moon} = \{-v_c\sin\theta,\ v_c\cos\theta\} \tag{19}$$

Substituting in our initial angular coordinate then yields the initial velocity of $\dot{\mathbf{r}}_{\rm init} = \{0.201, 0.110\}$.

With these initial Moon position and velocity vectors, we re-ran our integration. We see that the time at minimum separation is approximately 88.7 (Figure 1, right) – exactly the time that the projectile reaches its apogee. Hence the Moon CoM passes by the projectile at exactly the time the projectile reaches zero velocity. As expected, Figure 1 shows that the Moon has a circular orbit and passes by extremely close to the projectile at one point in time. The small gap between the projectile and Moon CoM is because of the numerical error mentioned earlier and the fact that we did not plot the radius of the Moon in the same plot.

Part 1.2 – Three-Body Problem

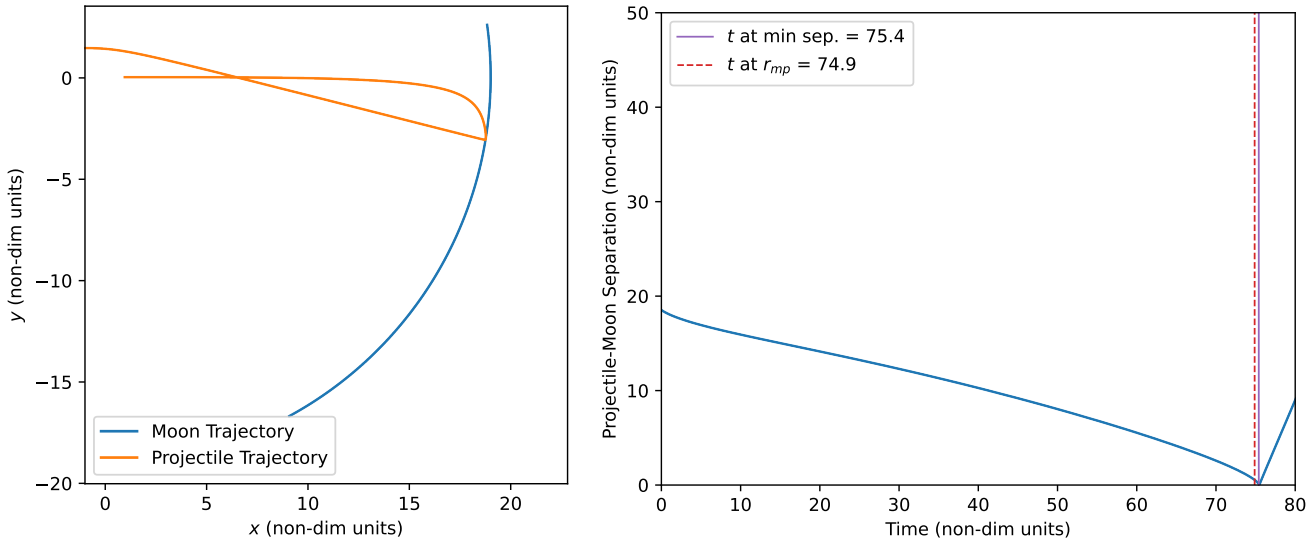

Figure. 2 Largely analogous to Figure 1, we show the dynamics of the planet-moon-projectile system but this time having initialised the Moon with a mass of 0.1 non-dimensionalised units. *Left:* We now see that the projectile's motion is no longer restricted to the x-axis, with the Moon's gravity pulling it off course and giving a gravity-kick back in the direction of the planet. *Right:* The time at minimum separation is now significantly less than before due to the extra acceleration of the moon 'pulling' the projectile outwards. We note the times at minimum separation and when the separation reaches $r_{\rm mp} = 0.5$.

We now look at our system including the mass of the Moon. The corresponding equation of motion for each

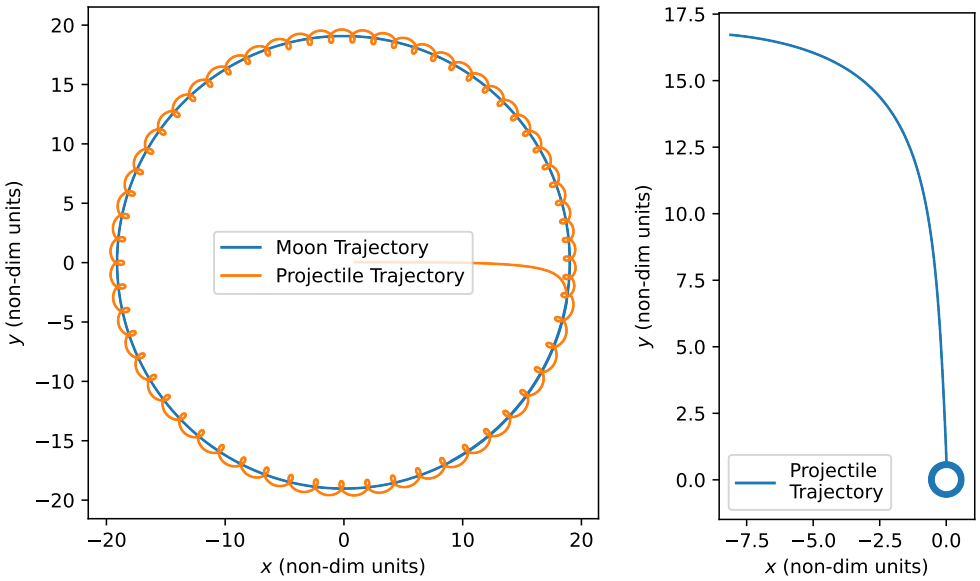

Figure. 3 After the kick described by equation (24), the projectile follows a close orbit to the Moon which orbits around the planet. We integrate for an entire Moon orbit, plus the time taken up to the kick, i.e. a total time of $520.368 + 74.9 = 595.268$. *Left:* The moon-particle trajectories centered on the planet CoM. *Right:* The projectile trajectory relative to the Moon CoM. We see the projectile approach the Moon from a large distance, then instantaneously bound in a roughly circular orbit. The line width is constant here, so the seemingly thicker region around (0, 0) corresponds to a deviation from a perfectly circular orbit with orbital precession.

component in the system is

$$\ddot{\mathbf{r}}_{\text{pl}} = 0 \tag{20}$$

$$\ddot{\mathbf{r}}_{\text{moon}} = \frac{m_{\text{pl}}}{|\mathbf{r}_{\text{moon, pl}}|^2}\hat{\mathbf{r}}_{\text{moon, pl}} + \frac{m_{\text{proj}}}{|\mathbf{r}_{\text{moon, proj}}|^2}\hat{\mathbf{r}}_{\text{moon, proj}} \tag{21}$$

$$\ddot{\mathbf{r}}_{\text{proj}} = \frac{m_{\text{pl}}}{|\mathbf{r}_{\text{proj, pl}}|^2}\hat{\mathbf{r}}_{\text{proj, pl}} + \frac{m_{\text{moon}}}{|\mathbf{r}_{\text{proj, moon}}|^2}\hat{\mathbf{r}}_{\text{proj, moon}} \tag{22}$$

where the planet is not moving due to our simplistic assumption that its mass is too high to be significantly altered by the gravity of the other components. In reality the Moon's mass would impose a slight gravitational wobble of the planet, but this is too small to affect the dynamics of the projectile once in a close orbit of the Moon (our end goal in this simulation).

Using the same code and formulation as before, we re-ran the simulation including the Moon mass. Prior to integration our system had an energy of -0.0026316 and after an energy of -0.0026309 – an error of only 0.027%. Figure 2 (left) shows the trajector of the particle is now strongly influenced by the gravity of the Moon once it is sufficiently close. For most of the initial ascent of the projectile, the linearised approximation in Part 1.1 is accurate enough (at least by eye), but the dynamics significantly change at late times. Notably too, Figure 2 (right) shows that the encounter between the projectile and Moon occurs at a much earlier time than in Part 1.1 as expected. The goal of the next part is to introduce an instantaneous kick to the projectile so that it enters into a circular orbit around the Moon with a semi-major axis of $r_{\text{mp}} = 0.5R_{\text{pl}} = 0.5$. We see in Figure 2 that the projectile reaches this distance from the Moon at approximately 74.9 non-dimensionalised time units, and so this is when we need the kick to take place in our simulation.

To determine the magnitude and direction of the kick needed to place the projectile in a circular orbit around the Moon, we neglect the gravitational affect of the planet due to its distance (which would mean it has a mostly negligible affect due to the inverse square law of gravity) and the proximity of the projectile to the Moon. To begin, we can cancel out any velocity that the projectile has, and so we will need a $-\mathbf{v}_{\text{proj}}$ term in $\mathbf{\Delta v}$. Since we want our projectile to be moving relative to the Moon, we should also add the Moon's instantaneous velocity at the time of the kick so that the projectile is then stationary with respect to the Moon CoM, and so we will also need a $+\mathbf{v}_{\text{moon}}$ term in $\mathbf{\Delta v}$. Finally, we want to impose a circular orbit for the projectile. This means we'll need a final orbital velocity of $v_{\text{circ}} = \sqrt{GM/r} = \sqrt{m_{\text{moon}}/r_{\text{mp}}} = \sqrt{0.1/r_{\text{mp}}}$. If we want our projectile to orbit anti-clockwise

around the Moon (with our choice of $x-y$ coordinate system), we need to impose a circular velocity direction of $\{-\sin\theta_{\rm mp},\ \cos\theta_{\rm mp}\}$, where

$$\theta_{\rm mp} = \text{atan2}\left(y_{\rm moon} - y_{\rm proj},\ x_{\rm moon} - x_{\rm proj}\right) - \pi \tag{23}$$

We chose to utilise the atan2 function over a simple arctan implementation so that our kick calculation was robust to the relative positions of the projectile around the coordinate axis of the Moon, where the $-\pi$ term corrects the angular calculation to center the coordinate axis at the Moon CoM. The final kick calculation that we employ in the code is then

$$\mathbf{\Delta v} = \sqrt{\frac{0.1}{r_{\rm mp}}}\{-\sin\theta_{\rm mp},\ \cos\theta_{\rm mp}\} - \mathbf{v}_{\rm proj} + \mathbf{v}_{\rm moon} \tag{24}$$

which is done at $t = 74.9$.

We show the result of this kick in Figure 3. The orbit of the projectile is apparently ever so slightly elliptical (as per the figure caption), or is circular but it is being perturbed by the influence of the planet regardless. Nonetheless, for our purposes the implementation of the kick achieved our goal with great success.

In summary, we've used the RK4 method to numerically integrate gravitational equations of motion so that we could simulate a particle and Moon trajectory and gravitational encounter. Using an analytically determined kick at a pre-determined time, we were able to inject this projectile into a quasi-stable orbit around the Moon which remained for at least one Moon orbital period of 520.368 non-dimensionalised time units.

Part 2 – MCMC for the 2D Ising Model

For the next part of the assignment, we developed a C++ implementation of the Ising model – a toy model that replicates the behaviour of a magnetic crystalline structure with periodic boundary conditions. The model was implemented entirely within the class `IsingLattice` within `isingmodel.hpp`, where the class methods call on parameters unique to an initialised lattice.

In all simulations below (except where otherwise stated), Ising lattices were initialised with $L \times L$ randomly oriented dipoles at a temperature well above the critical temperature of $T_c = 2/\ln(1+\sqrt{2}) \simeq 2.27$ in non-dimensionalised units. Also except where otherwise stated, we ran $1000 \times \eta(T)$ sweeps of the lattice for each data point, where a sweep corresponds to $10 \times L \times L$ monte carlo steps on randomly chosen dipoles, and the arbitrarily chosen function

$$\eta(T) = 1 + 19\exp\left(-\left(\frac{T - T_c)}{0.3}\right)^2\right) \tag{25}$$

allowed us to sample more sweeps around the critical temperature where the statistical properties of the lattice quickly changes. The state of the system was output after each sweep, and the following figures and analysis uses the $1000 \times \eta(T)$ data samples for each temperature.

A Monte Carlo step is where we check the change in energy (using the spins of the nearest neighbours) if we were to flip a specific dipole. If the change in energy is negative, we immediately flip that dipole and update the lattice total energy and magnetisation accordingly. If that change in energy is positive, we generate a standard uniform random number and flip the dipole if the change in energy is less than or equal to that random number.

The code was configured to (optionally) output the state of the lattice after sweeping, which would produce in the terminal some text reminiscent of Figure 4.

Part 2.1 – Thermodynamic Properties of the 2D Ising model

To begin, we consider a 20×20 Ising lattice at a temperature of $T = 5$ non-dimensionalised units. The goal of this section is to consider how the properties of the lattice change as we change the temperature and hopefully settle into a stable lattice configuration of entirely up or down spin dipoles for temperatures far below the critical temperature.

In the absence of a magnetic field, the total energy of the Ising lattice is calculated as

$$\mathcal{E} = -J\sum_{\langle i,j\rangle}^{L}\sigma_i\sigma_j \tag{26}$$

where J is the spin-spin interaction strength (here taken as $J = 1$) and $\sigma_i = \pm 1$ is the spin of the ith dipole. The summation is taken over pairs of the nearest 4 neighbouring dipoles (i.e. only along each axis and not diagonally).

```
0 - - - 0 0 - 0 0 - 0 0
- - 0 0 0 0 0 0 - - 0 -
0 0 0 - - - 0 0 0 0 0 0
0 - - - 0 - 0 0 0 0 0 -
0 - - 0 - 0 0 0 0 0 - 0
0 0 0 - - - - 0 0 - 0 -
- - - - - - - 0 0 0 - -
0 - 0 - 0 - - 0 0 0 - -
0 - - 0 0 0 0 - - 0 - -
- - - - - - - - - 0 0 -
0 0 0 - - - 0 0 0 0 0 -
- - - 0 0 - - 0 - 0 - 0
```

Figure. 4 A 12×12 random configuration lattice, which forms the initial state of a lattice at high temperature. Here "0" corresponds to a spin 2up dipole and "−" corresponds to a spin down dipole. Since the boundaries are periodic, the right-most dipoles are effectively neighbouring with the left-most dipoles and the top-most with the bottom-most.

During the simulation, we only need to take into account $\Delta\mathcal{E}$ when deciding to flip a spin, and so we add on that change in energy to the total energy whenever we decide to flip a spin.

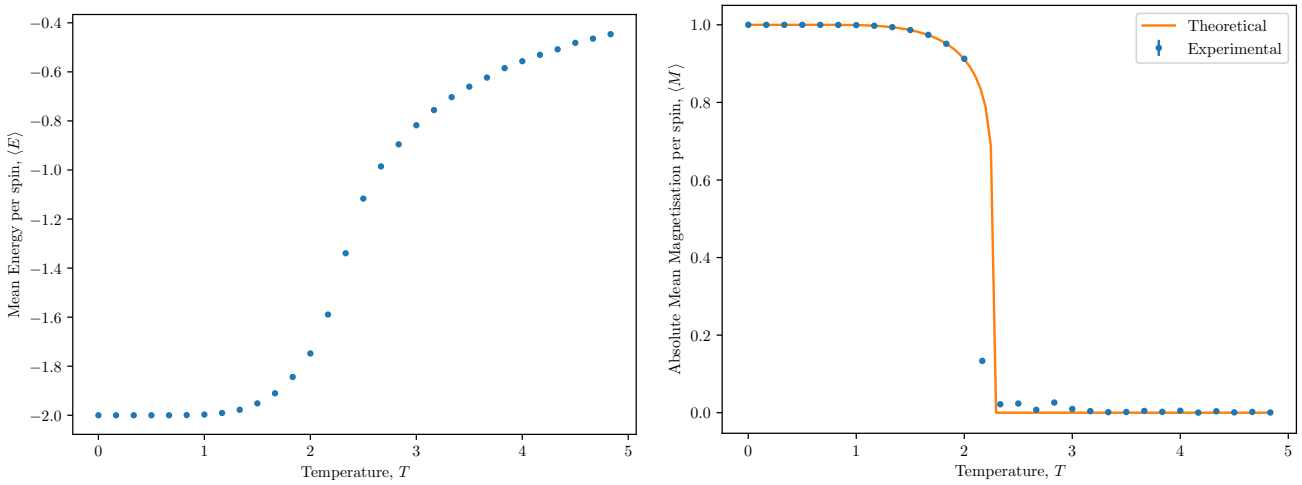

Figure. 5 The system parameters are sensitive to the temperature of the thermal bath around the lattice. The error bars in each plot are $\sqrt{\text{var}(E)/\mathcal{N}}$ and $\sqrt{\text{var}(M)/\mathcal{N}}$ respectively but are largely too small to see on account of the large number of sweeps. *Left:* The mean energy per dipole tends to a single value when the lattice is approximately all either spin up or spin down. The rate of change of this is highest at the critical temperature where it then starts decreasing as neighbouring spins tend to no longer be correlated with their neighbouring spins. *Right:* We plot the (absolute) mean magnetisation per dipole to avoid any inconsistencies with the lattices being entirely spin up or down (which would result in a flipped magnetisation value. Overlaid is the expected theoretical value of the absolute magnetisation [equation (27)] which is 0 after the critical temperature (i.e. there is an expected equal number of up and down spins).

In principle we expect that the energy of the lattice as a whole will be very negative for low temperatures and more positive for high temperatures. A very negative energy corresponds to neighbouring spins having the same sign resulting in the largest possible sum in equation (26) and hence the most negative overall energy. If all neighbouring spins were anti-aligned, we would then expect that the sum in that equation would tend to 0, and hence a more positive (tending to 0) energy would emerge. A useful measure independent of the lattice size we can use is the energy per dipole, $E = \mathcal{E}/(L \times L)$. On viewing Figure 5 we see exactly this relationship between energy per spin and temperature, where the critical temperature corresponds to the largest gradient in the energy per spin.

Another fundamental state parameter of a lattice is the total magnetisation $\mathcal{M}$, or equivalently the magnetisation per spin $M = \mathcal{M}/(L \times L)$. This is another measure of the overall magnetisation of the lattice (as with energy)

except doesn't rely on neighbour interactions. Since the 2D Ising model is a solved problem, we know that the (absolute) magnetisation per dipole follows

$$|M| = \begin{cases} \left[1 - \sinh^{-4}\left(\frac{2J}{k_B T}\right)\right]^{1/8} & T < T_c \\ 0 & T \geq T_c \end{cases} \tag{27}$$

We plot this overlaid onto our (absolute) data in Figure 5. In general this theoretical value may disagree with the data since the lattice can settle into an entirely spin up or spin down state at low temperature – this is why we plot the absolute values as the theoretical curve only agrees in magnitude of magnetisation. We see in the plot that below the critical temperature, the net magnetisation of the lattice is strongly positive and falls to about 0 for $T > T_c$.

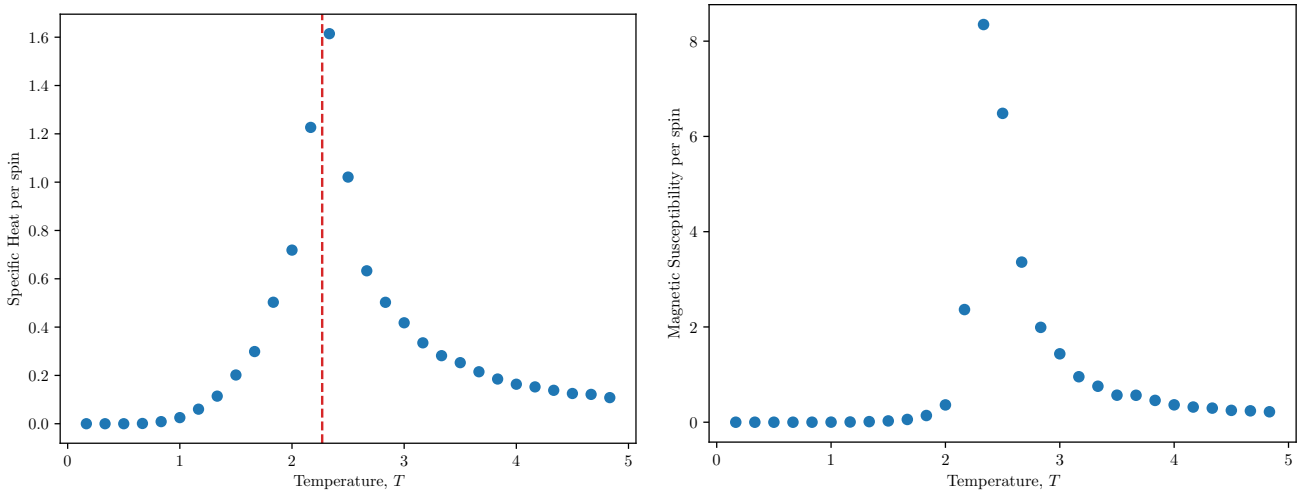

Figure. 6 *Left:* The specific heat per spin shows an apparent divergence at the critical temperature T_c. *Right:* The magnetic susceptibility per spin shows a clear bump around the critical temperature.

Some other quantities of interest are the specific heat per spin, c, and the magnetic susceptibility per spin, χ, of the lattice,

$$c = \frac{\text{var}(E)N_s}{T^2} \qquad \chi = \frac{\text{var}(|M|)N_s}{T} \tag{28}$$

where these equations are non-dimensionalised, $N_s = L \times L$, and the variance is taken over the sweeps done at that temperature. The former quantity describes the energy needed to change the temperature of the lattice. If the variance in the energy over the sweeps is very low, then the lattice very readily flips dipoles to an equilibrium state. Conversely if the variance in the energy over the sweeps is high, the lattice will have regions of quasi-stability will consequently need more sweeps to approach an equilibrium. The latter quantity describes how sensitive the lattice would be to an external magnetic field. When the magnetic susceptibility is high, an external magnetic field can more easily/quickly change the spin orientation of the lattice. Intuitively we would expect this parameter to be large near the critical temperature from above when the lattice begins to approach an equilibrium all up or down spin state.

Figure 6 shows how these parameters change in our simulated 20×20 lattice. We see broadly what we expect where each parameter spikes near the critical temperature $T_c \simeq 2.27$. Of particular note is the long tail (leading towards higher temperature) for the magnetic susceptibility. This is due to the regions (or 'islands') of quasi-stability that are very sensitive to an external magnetic field – below the critical temperature these islands all merge into one of two macrostates (all spin up or down) given enough sweeps. We also see a (slight) tail for the specific heat per spin plot too largely for the same reason. We expect that very low temperatures will have effectively no islands of quasi-stability and so will quickly change any rogue dipole to align with the lattice as a whole.

Part 2.2 – Critical Scaling Behaviour of the 2D Ising model

We now turn our attention to how the observables in the Ising model change when varying lattice size. Since physical systems are composed of a (very) large number of 'dipoles', we are usually interested in how the system behaves

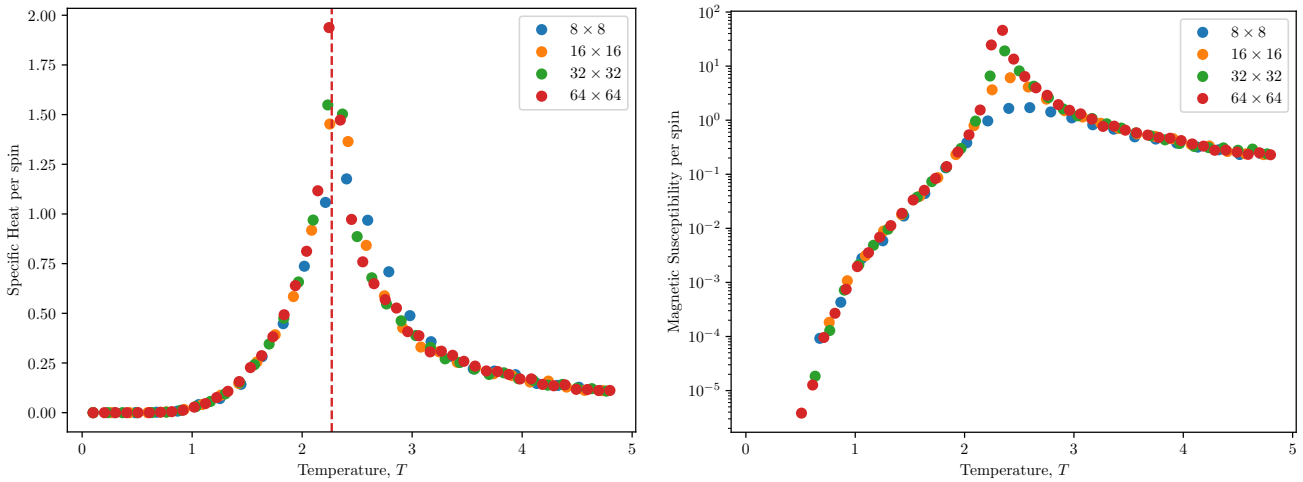

Figure. 7 We plot the specific heat and magnetic susceptibility per spin parameters for different lattice sizes and temperatures. In each case, the behaviour of the systems are largely the same for temperatures far from the critical temperature. Near the critical temperature, we see the parameter peaks get steeper and shifting to the left for larger lattice sizes. There are correspondingly more samples for larger lattices sizes to better sample the divergent behaviour near the critical temperature peak.

for a very large lattice. This is, in practice, impractical to always compute, and so knowing how the parameters change allows us to infer the behaviour of very large lattices from our simulations with smaller lattices.

We expect that the energy and magnetisation per spin stay consistent no matter the lattice size as they are mostly indicative of the system macrostate rather than how the system is changing with temperature. Figure 7 shows how the specific heat per spin and the magnetic susceptibility per spin change varying temperature and lattice size. Since these quantities encode some variational information about the system, we do expect these to be a function of lattice size.

We see broadly that the peaks in each plot are steeper and slightly shifted towards lower temperatures (or more accurately the critical temperature) for larger lattice sizes. In the case of the magnetic susceptibility, we would expect that a quasi-stable island in smaller lattice would be more stable against an opposing external magnetic field due to fewer nearest-neighbour interactions which results in a resistance to large-scale change across the lattice. Hence a larger lattice might have more, but relatively smaller, quasi-stable regions which will more easily respond to a magnetic field. This is the same intuition needed to explain the varying-lattice-size behaviour in the specific heat plot also. If there are more quasi-stable regions, the thermal bath would need to exert correspondingly more energy to flip the spins toward a stable equilibrium which results in a larger peak. The peak shifting towards the critical temperature shows how larger lattices are less responsive to immediate change (i.e. they require more sweeps to see a significant change in macrostate) and so a lower temperature is needed to enforce large scale spin flipping.

With the behaviour of different lattices investigated for a few representative sizes, the logical next step is to see if a single function can encode the system behaviour independent of lattice size. These finite-size scaling laws,

$$c(t) = \log(L) f(L^{1/\nu} t) \qquad \Rightarrow \qquad f(t) = \frac{c(L^{1/\nu} t)}{\log(L)} \tag{29}$$

$$\chi(t) = L^{\gamma/\nu} g(L^{1/\nu} t) \qquad \Rightarrow \qquad g(t) = \frac{\chi(L^{1/\nu} t)}{L^{\gamma/\nu}} \tag{30}$$

represent a unified form for specific heat per spin, $f(t)$, and magnetic susceptibility per spin, $g(t)$, respectively for a rescaled temperature $t = (T - T_c)/T_c$. The right side of equations (29, 30) shows the needed transformations to represent these unified forms in terms of our lattice-size dependent observables. In the above equations, the critical exponents are $\nu = 1$ and $\gamma = 7/4$, and we plot these rescaled observables in Figure 8. The rescaled specific heat per spin is only unified for rescaled temperatures extremely close to the critical temperature ($t = 0$), whereas the magnetic susceptibility is uniform with lattice size over a much larger domain about the critical temperature. Since we've shown that these observables can collapse to a single (respective) curve independent of lattice size, we can at least approximate the behaviour of very large systems (the so called thermodynamic limit with an infinite lattice) using the results of computationally practical small systems. As we increase the lattice size we're effectively requiring

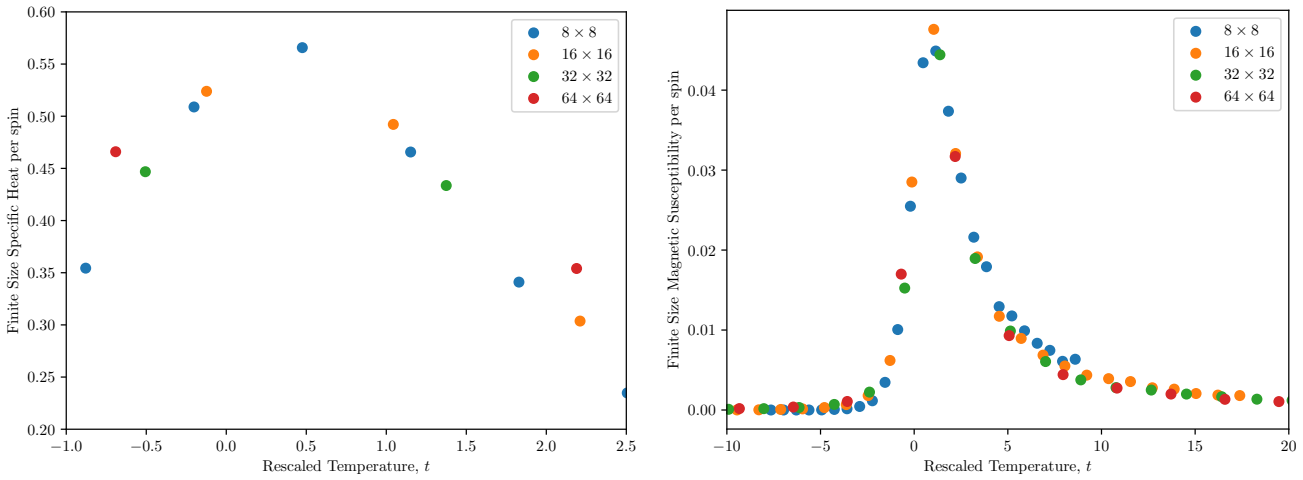

Figure. 8 The data transformed to fit the finite-size scaling laws shows good agreement very close to the critical temperature ($t = 0$). *Left:* An approximate form of $f(t)$ showing the finite-size specific heat per spin. Only the region very close to the critical temperature seems to obey the finite-size scaling law. *Right:* An approximate form of $g(t)$ showing the finite-size magnetic susceptibility per spin. The domain of agreement between lattice sizes is much larger here, going from at least $-5 \leq t \leq 5$.

the correlation length to increase proportionally when finding these unified curves. That is, we're determining these observables over an increasingly better approximation to the thermodynamic limit which helps us be more confident in our finite-scaling curves as an approximation to this limit.

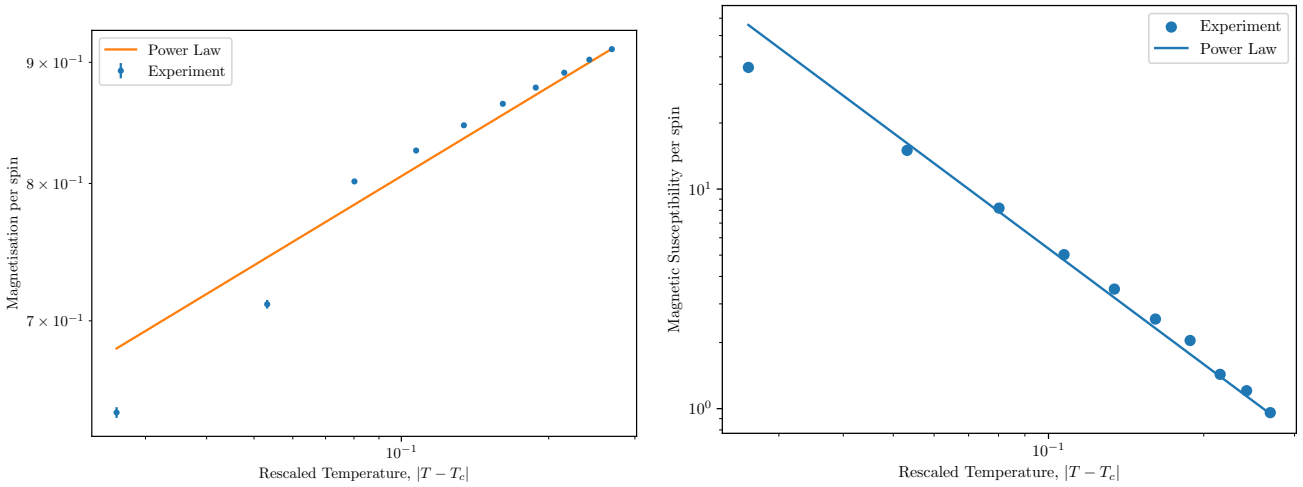

Figure. 9 Here we show the behaviour of a 40×40 lattice very close to the critical temperature, sampled in the range $2 < T < T_c$ for 10 equally spaced steps. Overlaid are power model fits according to equation (31), i.e. of the form $y = cx^b$, where c was found numerically via a least-squares optimisation on the data.

In the thermodynamic limit, we expect that the magnetisation per spin and magnetic susceptibility per spin are unified in a power law against $|T - T_c|$,

$$|M| \sim |T - T_c|^{\beta} \qquad \chi \sim |T - T_c|^{-\gamma} \tag{31}$$

where the critical exponents are $\beta = 1/8$ and $\gamma = 7/4$ as before. To investigate this, we ran a 40×40 lattice in the temperature range $2 < T < T_c$ with 10^4 sweeps per temperature. The results are plotted in Figure 9 with an overlaid power law of the form $y = c\,|T - T_c|^b$ as in equation (31). These power laws show the magnetic properties of the lattice as a function of distance from the critical temperature $|T - T_c| = 0$. In the case of the magnetisation per spin, we see that the lattices get very quickly more magnetised after the critical temperature, and correspondingly

the lattice is less susceptible to the affects of an external magnetic field (or equivalently that the lattice is quickly self-approaching a magnetic equilibrium or all spin up or down).

Part 2.3 – OpenMP Implementation

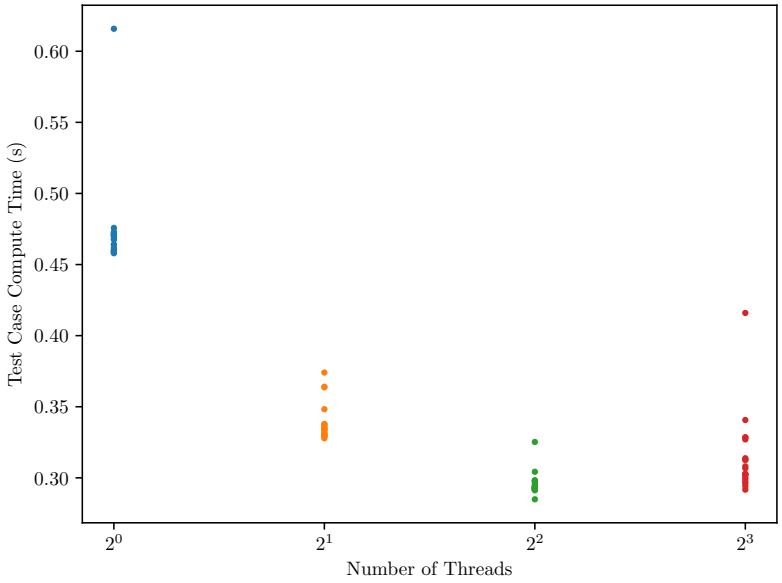

Figure. 10 We ran 1 sweep of a 64×64 lattice 20 times at varying number of CPU threads to get an idea of the distribution of speeds vs threadcount; each point in this scatter plot corresponds to 1 of those 20 samples. Each sample was taken from the same lattice at $T = 4$ – well above the critical temperature $T_c \simeq 2.27$ and so the lattice should have a quasi-random orientation in each sample. This data was taken through WSL2 on a Windows 11 AMD 8-core CPU machine with unavoidable background processes still operating during testing.

Given that the spin flips in the Ising lattice are only dependent on the spins of nearest-neighbours, the simulation code is well suited to a computational parallelisation approach where flips far away from each other can be flipped at the same time. To implement this, we wrote new parallelised code that iterates over the even rows in the lattice (delegating the rows evenly to the number of threads), sequentially moving through the columns to assess whether to flip each dipole, then moves through the odd rows in the same fashion. Since each even (odd) row is independent of each other, this avoids the problem of trying to flip neighbouring spins at the same time. Since OpenMP's parallel for loops include a barrier at their conclusion, we didn't need to manually add this to our code. We also intentionally share the lattice state within the parallelisation region so that each thread can read and write to it independently, where our spin flip function updates the lattice state parameters.

To benchmark the behaviour of our code, we ran this on an 8-core machine, where a test case of 1 sweep on a 64×64 lattice was timed multiple times for varying threadcount; the results of this are shown in Figure 10. Immediately we see a general trend of speed up in the code for increasing threadcount. As the number of running threads approaches the system core-count, though, the code starts to slow down due to other system processes running which 'hog' the system performance. Similarly, the relative overhead for creating and destroying child processes increases which adds a slowdown constant to the benchmark timing. Another key feature of the plot is that there is always ~ 1 data point far above the rest – this is where the program is first initialising the threads which carries with it some computational overhead.

The parallelised code, in principle, works better for larger lattice sizes with more sweeps. For a larger lattice size, the effect of sharing the computational load between threads makes for more efficient computing. However, the computational overhead increases correspondingly in which case a greater relative speed up happens when iterating over many more sweeps. As we ran this benchmark on a consumer machine and not a HPC, we chose to stick to a modest lattice size with few sweeps so as to not encounter significant computation time.

In summary, using our relatively simple C++ implementation of the Ising model we investigated the thermodynamic properties of lattices in two dimensions when varying the temperature and lattice size. We found that analytic forms of properties fit our data very well, including an investigation into finite-size scaling methods which

unify the lattice-size-varying properties into a single curve which can be used to infer the behaviour in the thermodynamic limit. Finally, we offer a parallelised version of the code where we demonstrated a significant speed up for increasing threadcount in a simple benchmark.