Stochastic Simulation

Autumn Semester 2024

Prof. Fabio Nobile *Assistant: Matteo Raviola*

Lab 10 – 21 November 2024

Markov Chains Monte Carlo

Exercise 0

Recall the Metropolis-Hastings algorithm 1.

Algorithm 1: Metropolis-Hastings Algorithm

Require: λ (initial distribution), Q (proposal distribution), π (target distribution)

Generate $X_0 \sim \lambda$
for $n = 0, 1, \dots$ **do**
  Generate candidate new state $\tilde{X}_{n+1} \sim Q(X_n, \cdot)$
  Generate $U \sim \mathcal{U}([0,1])$
  if $U \leq \alpha(X_n, \tilde{X}_{n+1})$ **then**
    Set $X_{n+1} = \tilde{X}_{n+1}$ {Candidate accepted with probability α}
  else
    Set $X_{n+1} = X_n$ {Candidate rejected with probability $1 - \alpha$}
  end if
end for

1. Compute the transition matrix of the Markov chain $\{X_n\}_{n\geq 0}$ generated by Algorithm 1.
2. Show that the transition matrix is in detailed balance with π.

Solution

See section 8.1.1 of the lecture notes.

Exercise 1

Let us consider a 2D uniform square-lattice with atoms placed at each vertex, as is sketched in Figure 1. The atoms can have an upward (red arrow) or a downward (blue arrow) pointing magnetic moment (so-called *spin*). Specifically, let the lattice be made out of $m \times m$ atoms. Therefore the system's possible states are the 2^{m^2} possible spin choices for the m^2 atoms. That is, the spin of the atom at position (i, j) in the lattice is denoted with s_{ij}, $1 \leq i, j \leq m$, and can take a value in $\{-1, +1\}$. A specific system configuration is described by the matrix $\boldsymbol{S} = (s_{ij}) \in \{-1, +1\}^{m\times m}$, containing the spin of each of the m^2 atoms.

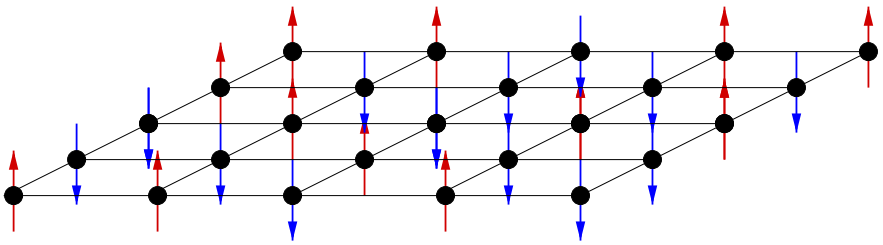

Figure 1: Sketch of 2D square-lattice Ising model.

The energy of a given system state of this Ising model is given by

$$H(\boldsymbol{S}) = -\sum_{i,j=1}^{m} \left(\frac{1}{2} J s_{ij} (s_{i-1,j} + s_{i+1,j} + s_{i,j-1} + s_{i,j+1}) + B s_{ij} \right) , \tag{1}$$

where J is a magnetic coupling constant and B is a constant describing the external magnetic field. To account for boundary effects, we set $s_{-1,j} = s_{j,-1} = s_{m,j} = s_{j,m} = 0$ in (1). The probability of obtaining a specific system state is then given by the *Boltzmann* distribution with Probability Mass Function (PMF)

$$f(\boldsymbol{S}) \equiv f_\beta(\boldsymbol{S}) = \frac{1}{Z_\beta} e^{-H(\boldsymbol{S})\beta} , \tag{2}$$

where $\beta = 1/(k_B T)$ denotes the so-called inverse-temperature (or thermodynamic beta) with k_B being the Boltzmann constant and T the absolute temperature. Here, Z_β denotes the normalization constant that makes the target distribution $f_\beta \colon \{-1, +1\}^{m \times m} \to \mathbb{R}_+$ a proper PMF.

Let's denote by $M(\boldsymbol{S}) = \sum_{i,j=1}^{m} s_{ij}/m^2$ the system's average magnetic moment corresponding to the configuration $\boldsymbol{S}$. Notice that the random realizations of the configuration matrix $\boldsymbol{S}$ depend on the inverse temperature β. The expected value of the average magnetic moment $\overline{M}(\beta)$ as a function of the inverse temperature β thus reads

$$\overline{M}(\beta) = \sum_{\boldsymbol{S} \in \mathcal{K}} M(\boldsymbol{S}) f_\beta(\boldsymbol{S}) = \frac{1}{Z_\beta} \sum_{\boldsymbol{S} \in \mathcal{K}} M(\boldsymbol{S}) e^{-H(\boldsymbol{S})\beta} , \tag{3}$$

where $\mathcal{K} = \{-1, 1\}^{m \times m}$ is the set of all possible system configurations. Since the explicit computation of the normalization constant Z_β is computationally expensive (Explain why!), we rely on the Metropolis–Hastings algorithm here. That is, at each step a candidate configuration is proposed by randomly choosing an atom, with uniform probability, and "flipping" its spin.

1. Write a `Python` function that implements the Metropolis–Hastings algorithm for the Ising model. The input parameters for your function are: the number of steps n of the chain that should be simulated, the number of atoms m^2, the inverse temperature β, the constants J and B, and the initial state of the system. The function should return a list of energies and mean magnetic moments computed for each step of the chain, as well as the final configuration of the system.

2. Use your `Python` function with $\beta = 1/3$ and for n, such that both the energy and the average magnetic moment appear to have reached stationarity. Plot also the final system configuration. Furthermore, compute the mean magnetic moment $\overline{M}(\beta)$ for different values of $\beta \in [\frac{1}{3}, 1]$ and $n = 5 \cdot 10^6$. Choose a lattice of 50×50 atoms, $J = 1$, and $B > 0$ for all simulations.

Solution

Notice that the energy

$$H(\boldsymbol{S}) = -\sum_{i,j=1}^{m} \left(\frac{1}{2} J s_{ij}(s_{i-1,j} + s_{i+1,j} + s_{i,j-1} + s_{i,j+1}) + B s_{ij} \right) ,$$

contains the products $s_{ij}s_{i-1,j}, \dots$ twice, so that we can rewrite the energy as

$$H(\boldsymbol{S}) = -\sum_{i,j=1}^{m-1} J s_{ij}(s_{i+1,j} + s_{i,j+1}) - \sum_{i,j=1}^{m} B s_{ij} ,$$

which is more amenable for an implementation (and, in fact, used below). It is noteworthy however, that also the evaluation of the energy in this rewritten form requires $\mathcal{O}(m^2)$ operations. That is, if m is large, the evaluation is computationally expensive!

However, in the Metropolis-Hastings algorithm, we only need to evaluate the energy difference between two different states. In particular, let us denote by $\boldsymbol{S}$ the current system configuration at step n of the algorithm and by $\boldsymbol{S}^c$ the proposed candidate configuration. Due to the particular proposal structure, it follows that the only difference between $\boldsymbol{S}$ and the candidate $\boldsymbol{S}^c$ is one spin. Suppose that this difference is at the atom in position (l, k), so that

$$\boldsymbol{S} = (s_{ij}) , \qquad \boldsymbol{S} = \left(s^c_{ij}\right) , \qquad \text{with} \qquad s^c_{ij} = \begin{cases} s_{ij} , & \text{if} \quad i \neq l,\ j \neq k , \\ -s_{ij} , & \text{if} \quad i = l,\ j = k . \end{cases}$$

Consequently, we can write the energy difference conveniently as

$$\begin{aligned} \Delta H(\mathbf{S}, \mathbf{S}^c) = H(\boldsymbol{S}^c) - H(\boldsymbol{S}) &= -J\left(s^c_{lk} - s_{lk}\right)\left(s_{l-1,k} + s_{l+1,k} + s_{l,k-1} + s_{l,k+1}\right) - B\left(s^c_{lk} - s_{lk}\right) \\ &= 2J s_{lk}(s_{l-1,k} + s_{l+1,k} + s_{l,k-1} + s_{l,k+1}) + 2B s_{lk} , \end{aligned}$$

which simplifies the implementation. Notice that the proposal transition matrix $Q(\mathbf{S}, \mathbf{S}^*)$ is symmetric. Indeed $Q(\mathbf{S}, \mathbf{S}^*) = Q(\mathbf{S}^*, \mathbf{S}) = \frac{1}{m^2}$ if $\mathbf{S}$ and $\mathbf{S}^*$ differ by only one spin and $Q(\mathbf{S}, \mathbf{S}^*) = 0$ if they differ by two or more spins. Hence the Metropolis-Hastings acceptance rate becomes

$$\alpha(\mathbf{S}, \mathbf{S}^c) = \min\left\{1, \frac{f(\mathbf{S}^c)Q(\mathbf{S}^c, \mathbf{S})}{f(\mathbf{S})Q(\mathbf{S}, \mathbf{S}^c)}\right\} = \min\left\{1, \frac{f(\mathbf{S}^c)}{f(\mathbf{S})}\right\} = \min\left\{1, \exp\left[-\beta \Delta H(\mathbf{S}, \mathbf{S}^c)\right]\right\} . \quad (4)$$

A possible `Python` code that uses these formulas is shown below. Figs. 2, 3, 4 and 5 show the evolution of energy and magnetic moment as well as the state configuration for $\beta = 1/3$ and $\beta = 1$ respectively. For $\beta = 1/3$, the configuration plot shows the formation of spin-up or spin-down clumps. The expected total magnetic moment is expected to be zero, and we get an estimated value of 4.17. For $\beta = 1$, the configuration plot shows a phase transition, namely that the spins all align eventually in the same direction. The absolute value of the

expected total magnetic moment of the invariant distribution is 2500 in theory, and we get an estimated value of approximately 1968 since we have not excluded any burn-in time from the calculation of the mean. We use the ergodic estimator

$$\overline{M}(\beta) \approx \frac{1}{n} \sum_{k=0}^{n} M(\mathbf{S}_k), \tag{5}$$

where $\mathbf{S}_k$ denotes the state at the k^{th} step of the MH algorithm.

Python code

```
import numpy as np

def ising(n, m, beta, J, B, S0):
    # Allocate vectors derived from a system's state
    E = np.zeros(n+1)
    M = np.zeros(n+1)

    # Initialize system
    S = S0
    # Magnetic moment associated with the initial condition
    M[0] = S.sum()
    # Energy associated with the initial condition
    E[0] = - J * ( (S[:,:m-1] * S[:,1:]).sum() + (S[:m-1,:]*S[1:,:]).sum() ) - B * M[0]

    for k in range(n):
        # PROPOSAL : generate candidate state
        Sc = S.copy()
        # select randomly an atom on the lattice
        i = int( np.floor(m * np.random.random()) )
        j = int( np.floor(m * np.random.random()) )
        Sc[i,j] = - S[i,j] # flip the spin

        # Change in magnetic moment due to this flip
        dM = - 2 * S[i,j]

        # Change in the energy due to this flip
        dE = 0
        if i>0:
                dE = dE + S[i-1,j]
        if i < m-1:
                dE = dE + S[i+1,j]
        if j>0:
                dE = dE + S[i,j-1]
        if j<m-1:
                dE = dE + S[i,j+1]
```

```
        dE = 2*S[i,j]*(J*dE + B)

        # ACCEPT-REJECT STEP
        alpha = np.min([np.exp(-dE*beta),1])
        U = np.random.random()
        if U < alpha:
                S = Sc # Accept proposed candidate state
        else:
                dM = 0
                dE = 0

        # Update energy and magnetic moment
        E[k+1] = E[k] + dE
        M[k+1] = M[k] + dM

return E, M, S
```

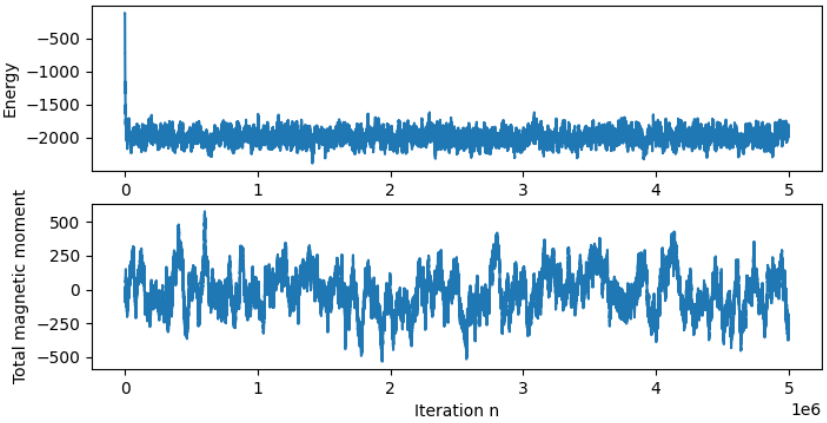

Figure 2: Evolution of energy and magnetic moment for $\beta = 1/3$.

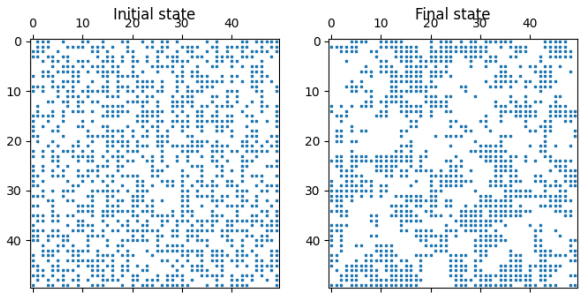

Figure 3: Snapshots of the configuration for $\beta = 1/3$.

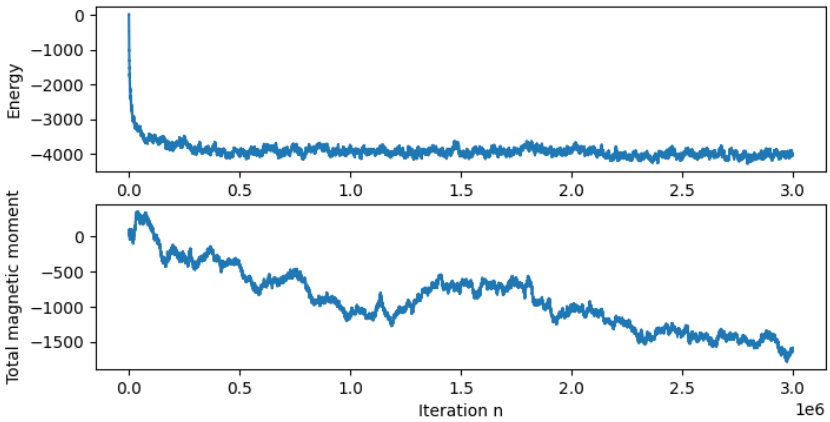

Figure 4: Evolution of energy and magnetic moment for $\beta = 1$.

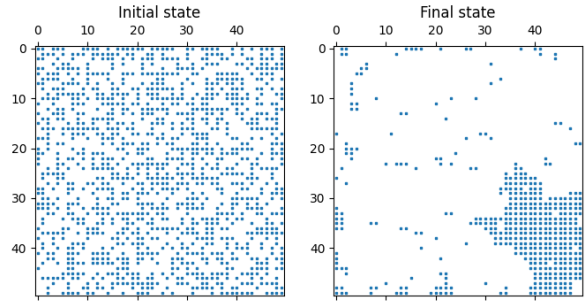

Figure 5: Snapshots of the configuration for $\beta = 1$.

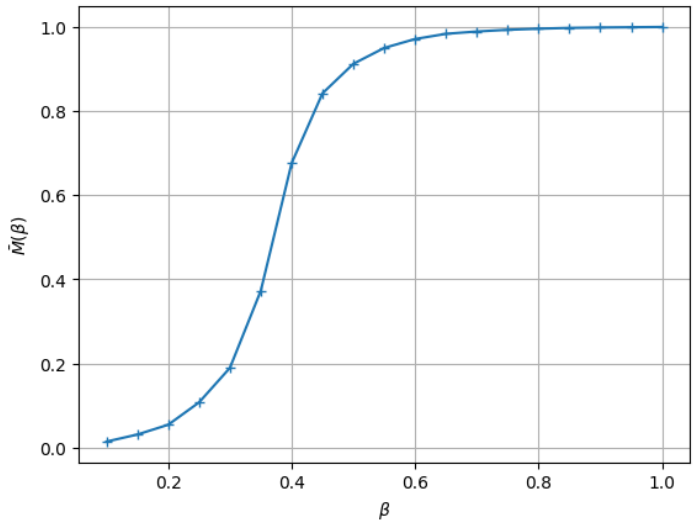

Figure 6: Mean magnetization versus β for $B = 0.1$

Exercise 2

Recall that the standard Metropolis-Hastings algorithm accepts a new candidate state j drawn from the transition matrix Q, given the current state i, with probability $\alpha(i,j) = \min\left(\frac{\pi_j Q_{ji}}{\pi_i Q_{ij}}, 1\right)$, where π is the target probability measure. Consider now a Metropolis-Hastings algorithm that uses the follwing alternative acceptance probabilities

$$\alpha_1(i,j) = \frac{\pi_j Q_{ji}}{\pi_j Q_{ji} + \pi_i Q_{ij}},$$

and

$$\alpha_2(i,j) = \frac{\delta_{ij}}{\pi_i Q_{ij}},$$

with δ such that $\delta_{ij} \leq \pi_i Q_{ij} \forall i,j$. Show that, in both cases, the produced Markov chain satisfies the detailed balance condition.

Solution

Note that the detailed balance condition reads:

$$\alpha(i,j)\pi_i Q_{ij} = \alpha(j,i)\pi_j Q_{ji}.$$

Consider the first case $\alpha_1(i,j) = \frac{\pi_j Q_{ji}}{\pi_j Q_{ji} + \pi_i Q_{ij}}$. We then have that

$$\alpha_1(i,j)\pi_i Q_{ij} = \frac{\pi_j Q_{ji}\pi_i Q_{ij}}{\pi_j Q_{ji} + \pi_i Q_{ij}} = \alpha_1(j,i)\pi_j Q_{ji}.$$

Consider now the second case $\alpha_2(i,j) = \frac{\delta_{ij}}{\pi_i Q_{ij}}$ where $\delta_{ij} = \delta_{ji}$ is chosen such that $\alpha_2(i,j) \leq 1$, $\forall i,j$. Then

$$\alpha_2(i,j)\pi_i Q_{ij} = \delta_{ij} = \delta_{ji} = \frac{\delta_{ji}}{\pi_j Q_{ji}}\pi_j Q_{ji} = \alpha_2(j,i)\pi_j Q_{ji}$$

Exercise 3

Consider the following AR(k) model defined by

$$\boldsymbol{y}_n = A\boldsymbol{y}_{n-1} + \boldsymbol{\xi}_n, \quad \boldsymbol{\xi}_n \overset{\text{iid}}{\sim} \mathcal{N}(0, \Gamma), \quad \boldsymbol{\xi}_n \in \mathbb{R}^k,$$

with $A \in \mathbb{R}^{k\times k}$, invertible and $\Gamma \in \mathbb{R}^{k\times k}$ full rank.

(a) Show that the previous process is a Markov chain.

(b) Show that if $\boldsymbol{y}_0 \sim \mathcal{N}(0, \Gamma_0)$, then $\boldsymbol{y}_n$ follows a multivariate Gaussian distribution for all n.

(c) Find the invariant distribution of an AR(1) process (i.e, a special case of the previous model).

(d) Simulate the AR(1) process and assess its convergence to the invariant distribution. In addition, verify the ergodic theorem on the quantity

$$\hat{\mu}^N = \frac{1}{N}\sum_{n=1}^{N} y_n.$$

(e) Establish theoretically the convergence of $\hat{\mu}^N$ by using the strong law of large numbers, and a weighted version of the central limit theorem (e.g. Lindberg-Feller)

Solution

(a) Trivially, using the definition of the AR(k) model and denoting $y_{[0,n]} = (y_0, y_1, \ldots, y_n)$, we have that

$$\begin{aligned}\mathbb{P}(\boldsymbol{y}_n \leq \boldsymbol{y}|\boldsymbol{y}_{[0,n-1]}) &= \mathbb{P}(\boldsymbol{\xi}_n \leq \boldsymbol{y} - A\boldsymbol{y}_{n-1}|\boldsymbol{y}_{[0,n-1]}) = \mathbb{P}(\boldsymbol{\xi}_n \leq \boldsymbol{y} - A\boldsymbol{y}_{n-1}|\boldsymbol{y}_{n-1}) \\ &= \mathbb{P}(\boldsymbol{y}_n \leq \boldsymbol{y}|\boldsymbol{y}_{n-1}),\end{aligned}$$

since $\mathbb{P}(\boldsymbol{\xi}_n \leq \boldsymbol{y} - A\boldsymbol{y}_{n-1}|\boldsymbol{y}_{n-1})$ is the Gaussian $\mathcal{N}(0, \Gamma)$ cummulative distribution function evaluated at $\boldsymbol{y} - A\boldsymbol{y}_{n-1}$.

(b) We can show that the characteristic function for $\boldsymbol{y}_1$ is

$$\mathbb{E}[e^{i\boldsymbol{t}'\boldsymbol{y}_1}] = \mathbb{E}[e^{i\boldsymbol{t}'A\boldsymbol{y}_0}]\mathbb{E}[e^{i\boldsymbol{t}'\boldsymbol{\xi}_1}] = e^{-\frac{1}{2}\boldsymbol{t}'A\Gamma_0A'\boldsymbol{t}}e^{-\frac{1}{2}\boldsymbol{t}'\Gamma\boldsymbol{t}} = e^{-\frac{1}{2}\boldsymbol{t}'(A\Gamma_0A'+\Gamma)\boldsymbol{t}} \tag{6}$$

that is the c.f. function of a Gaussian with zero mean and covariance $A\Gamma_0A' + \Gamma$. Using induction and repeating the same argument one can easily show that for arbitrary n the covariance is of $\boldsymbol{y}_n$ is $A^n\Gamma_0\left(A^n\right)' + \sum_{i=0}^{n-1} A^i\Gamma\left(A^i\right)'$.

(c) For $k = 1$ we have $y_n = ay_{n-1} + \xi_n$, $\xi_n \sim N(0, \gamma)$. Assume that $y_{n-1} \sim N(0, \sigma^2)$, then $y_n \sim N(0, a^2\sigma^2 + \gamma)$. Hence $N(0, \sigma^2)$ is invariant iff $\sigma^2 = a^2\sigma^2 + \gamma$ which implies $\sigma^2 = \frac{\gamma}{1-a^2}$ and $|a| < 1$.

(d) The empirical invariant distribution of AR(1) follows after simulating a large number of "paths" (here 10^4) and plotting the histogram and a large $n = 10^4$. Resulting distribution is shown in Fig. 7 for $a = 0.8$, $\gamma_0 = 2$, $\gamma = 0.2$. To verify the convergence, we observe the distribution of $\hat{mu}^N$ for different values of N in Fig. 8 and observe that the CLT is verified.

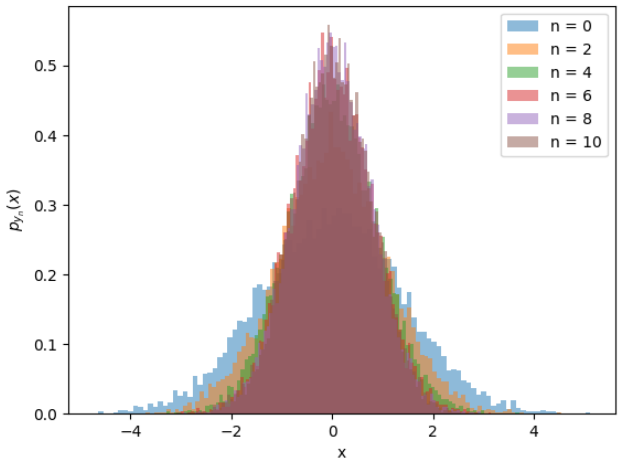

Figure 7: Histogram of the invariant distribution for different iteration numbers.

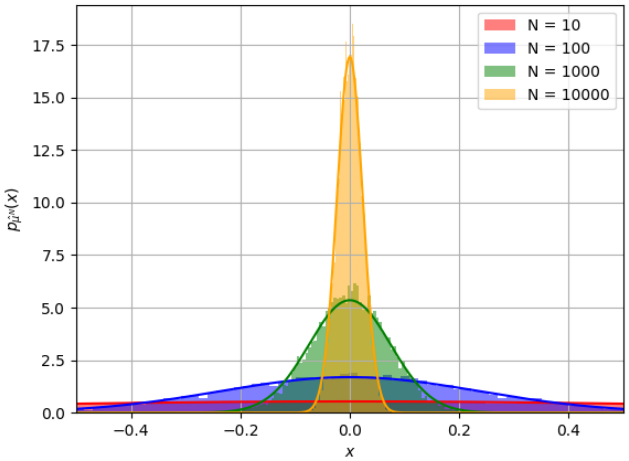

Figure 8: $\hat{\mu}^N$ histograms superposed with normal distributions with variance scaling as $1/N$.

(e) For the almost surely convergence we write

$$\begin{aligned}
\hat{\mu}^N &= \frac{1}{N}\sum_{n=1}^{N} y_n = \frac{1}{N}\sum_{n=1}^{N}\left(a^n y_0 + \sum_{i=1}^{n} a^{n-i}\xi_i\right) && (7)\\
&= y_0\frac{1}{N}\sum_{n=1}^{N} a^n + \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{n} a^{n-i}\xi_i && (8)\\
&= y_0\frac{1}{N}\sum_{n=1}^{N} a^n + \frac{1}{N}\sum_{i=1}^{N}\sum_{n=i}^{N} a^{n-i}\xi_i && (9)\\
&= y_0\frac{1}{N}\sum_{n=1}^{N} a^n + \frac{1}{N}\sum_{i=1}^{N}\frac{1-a^{N-i}}{1-a}\xi_i && (10)\\
&= y_0\frac{1}{N}\sum_{n=1}^{N} a^n + \frac{1}{1-a}\frac{1}{N}\sum_{n=1}^{N}\xi_i - \frac{a}{(1-a)N}\sum_{i=1}^{N} a^{N-i}\xi_i, && (11)
\end{aligned}$$

where by the SLLN, the first term on the right hand side $A_N := y_0\frac{1}{N}\sum_{n=1}^{N} a^n \to 0$ a.s., the second term $B_N := \frac{1}{1-a}\frac{1}{N}\sum_{i=1}^{N}\xi_i \to 0$ a.s.. For the third term $C_N := \frac{a}{(1-a)N}\sum_{i=1}^{N} a^{N-i}\xi_i$ we have

$$\mathbb{V}(C_N) = \frac{a^2}{(1-a)^2N^2}\sum_{i=0}^{N-1} a^{2i}\Gamma \le \frac{a^2}{(1-a)^2(1-a^2)}\frac{\Gamma}{N^2} \tag{12}$$

and $\mathbb{E}[C_N] = 0$, hence

$$\mathbb{P}(|C_N| < \epsilon) \le \frac{\mathbb{V}(C_N)}{\epsilon^2} \le \frac{k(\epsilon)}{N^2} \tag{13}$$

with $k(\epsilon) = \frac{a^2\Gamma}{\epsilon^2(1-a)^2(1-a^2)}$ and $\sum_{N=1}^{\infty}\mathbb{P}(|C_N| > \epsilon) \le \sum_{N=1}^{\infty}\frac{k(\epsilon)}{N^2} < +\infty$. So by the Borel-Cantelli lemma $C_N \to 0$ a.s.

For the CLT, we can use the Lindeberg-Feller theorem for triangular arrays:
For each N let $X_{N,m}$, $1 \le m \le N$ be independent random variables such that

- $\mathbb{E}[X_{N,m}] = 0$
- $\sum_{m=1}^{N}\mathbb{E}[X_{N,m}^2] \to \sigma^2 < \infty$
- For all $\epsilon > 0$, $\lim_{N\to\infty}\sum_{m=1}^{N}\mathbb{E}[|X_{N,m}|^2; |X_{N,m}| > \epsilon] = 0$.

Then $\sum_{m=1}^{N} X_{N,m} \to N(0,\sigma^2)$ in distribution. We apply the theorem with $X_{N,m} = \frac{1}{\sqrt{N}}\frac{1-a^{N-m+1}}{1-a}\xi_m$. Then
(i) $\sum_{m=1}^{N}\mathbb{E}[X_{N,m}^2] = \sum_{m=1}^{N}\frac{1}{N}\frac{1-a^{N-m+1}}{1-a}\gamma \to \frac{\gamma}{1-a}$
(ii)

$$\begin{aligned}
\sum_{m=1}^{N}\mathbb{E}\left[X_{N,m}^2\middle||X_{N,m}| > \epsilon\right] &= \sum_{m=1}^{N}\mathbb{E}\left[\frac{(1-a^{N-m+1})}{(1-a)^2N}\epsilon_m^2\middle||\xi_m| > \epsilon\sqrt{N}(1-a)/(1-a^{N-m+1})\right]\\
&\le \sum_{m=1}^{N}\frac{1}{(1-a)^2N}\mathbb{E}\left[\xi_m^2\middle||\xi_m| > \epsilon\sqrt{N}(1-a)\right]\\
&= \frac{1}{(1-a)^2}\mathbb{E}\left[\xi_1^2\middle||\xi_1| > \epsilon\sqrt{N}(1-a)\right] \to 0.
\end{aligned}$$

Hence $\sum_{m=1}^{N} X_{N,m} = \sqrt{N}\left(\frac{1}{N}\sum_{i=1}^{N}\frac{1-a^{N-i+1}}{1-a}\xi_i\right) \to N(0, \frac{\gamma}{1-a})$ and $\sqrt{N}\hat{\mu}^N \to N(0, \frac{\gamma}{1-a})$. Notice that $\frac{\gamma}{1-a} > \mathbb{V}[y_\infty]$, i.e. the estimator $\hat{\mu}^N$ is less efficient (has a larger variance) than a Monte Carlo estimator $\tilde{\mu}^N = \frac{1}{N}\sum_{n=1}^{N} y_n$ where y_n are iid $N(0, \frac{\gamma}{1-a^2})$.

Python code

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

a = 0.8
gamma0 = 2.0
gamma = 0.2

M = 10000 # Number of samples
y0 = st.norm.rvs(size = (M,)) * np.sqrt(gamma0)

plt.figure(2)
plt.hist(y0, bins = 100, density = True, label='n = 0',alpha=0.5)
N = 10000 # Number of steps
y = y0.copy()
mu = y0.copy()
Ntest = [10,100,1000,10000]
j = 0
colour = ['red', 'blue', 'green', 'orange']
Ntest2= [0,2,4,6,8,10]

x_axis = np.linspace(-1,1,1001)

mus = []
mu_vars = []
for i in range(N):
    y_new = a * y + st.norm.rvs(size = (M,)) * np.sqrt(gamma)
    y = y_new.copy()
    mu = ((i+1.)/(i+2.)) * mu + y / (i+2.) # incremental mean computation
    if (i+1) in Ntest:
        mus.append(np.mean(np.abs(mu)))
        mu_vars.append(np.var(mu))
        plt.figure(1)
        plt.hist(mu, bins=100,density=True,color=colour[j],alpha=0.5,label=r'N = '+str(i+1))
        plt.plot(x_axis, st.norm.pdf(x_axis, loc=0, scale=np.sqrt(gamma/(1-a**2)*10/(i+1))),c
        j = j + 1
    if (i+1) in Ntest2:
        plt.figure(2)
        plt.hist(y, bins = 100, density = True, label='n = '+str(i+1),alpha=0.5)
```

```
plt.figure(1)
#plt.loglog(Ntest, np.sqrt(mu_vars),'-+',label=r'$Var[\hat{\mu}^N]$')
#plt.loglog(Ntest, [100/np.sqrt(N) for N in Ntest],'--',color='red',label=r'$1/\sqrt{N}$')
plt.xlim((-0.5,0.5))
plt.grid(which='both')
plt.ylabel(r'$p_{\hat{\mu}^N}(x)$')
plt.xlabel(r'$x$')
plt.legend()
plt.savefig('../figures/asymptotic.png')

plt.figure(2)
plt.legend()
plt.xlabel(r'x')
plt.ylabel(r'$p_{y_n}(x)$')
plt.savefig('../figures/hist_invariant.png')
plt.grid(which='both')
plt.show()
```