Metropolis Algorithm

Project Report — Group 11

Sharath Kumar Yerramada (BSD-BG-2417)
Peravena Yashwanth (BSD-BG-2411)
Rathod Akash (BSD-BG-2415) Dipendu Halder (BSD-BG-2404)

May 2026
Indian Statistical Institute

Contents

1 Introduction

A central problem in statistics and physics is computing expected values of the form

$$\langle f(x) \rangle = \int f(x)\, P(x)\, dx \tag{1}$$

For simple distributions such as the Gaussian, this integral can be evaluated analytically. However, for complex distributions arising in Bayesian inference, statistical mechanics, and other fields, the integral is analytically intractable and direct sampling from $P(x)$ is often impossible.

Markov Chain Monte Carlo (MCMC) methods solve this problem by generating a sequence of samples $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ whose empirical distribution converges to the target $P(x)$. The expected value is then approximated by the sample average:

$$\langle f(x) \rangle \approx \frac{1}{K} \sum_{k=1}^{K} f\big(x^{(k)}\big) \tag{2}$$

The Metropolis algorithm [1], introduced in 1953 by Metropolis, Rosenbluth, Rosenbluth, Teller and Teller, is the most famous and widely used MCMC method. This report covers the algorithm in detail, verifies it through simulation on the Gaussian distribution using both textbook figures and our own R simulations, extends it to the bimodal distribution as shown in the textbook, and further extends it to the Exponential distribution as an original contribution of this project.

2 The Metropolis Algorithm

2.1 Target Distribution

The Metropolis algorithm applies to any probability distribution that can be written in the form

$$P(x) = \frac{e^{-S(x)}}{Z} \tag{3}$$

where $S(x)$ is called the **action** (or negative log-likelihood in statistics) and $Z = \int e^{-S(x)}\, dx$ is the normalising constant, called the **partition function** in physics.

In practice, $S(x)$ is always known, while Z is unknown because computing it requires evaluating exactly the integral we wish to avoid. Note that this form is completely general — any positive distribution $P(x)$ can be written this way by defining $S(x) = -\log P(x) + \log Z$.

2.2 Algorithm Steps

Starting from an initial value $x^{(0)}$, at each step k the algorithm proceeds as follows:

1. **Propose:** Draw $\Delta x \sim \text{Uniform}(-c, +c)$ and set $x' = x^{(k)} + \Delta x$
2. **Compute:** $\Delta S = S(x') - S\big(x^{(k)}\big)$
3. **Accept** x' with probability $\min\big(1,\, e^{-\Delta S}\big)$
4. **Update:**

$$x^{(k+1)} = \begin{cases} x' & \text{if accepted} \\ x^{(k)} & \text{if rejected} \end{cases}$$

The parameter $c > 0$ is the **step size**, a free tuning parameter chosen by the user. It controls how far each proposal jumps from the current position.

Why do we accept worse moves? If we only accepted proposals that increase the probability, the chain would get stuck in local regions and never explore the full distribution. Accepting occasional downhill moves allows the chain to fully explore $P(x)$ and guarantees convergence to the correct distribution.

2.3 The Metropolis Test

In practice, the acceptance step is implemented as the **Metropolis test**: draw $r \sim \text{Uniform}(0, 1)$ and accept x' if $r < e^{S(x^{(k)}) - S(x')}$, otherwise reject.

- Higher probability point $\rightarrow$ always accept
- Lower probability point $\rightarrow$ accept with some chance, not zero

2.4 Why Z Cancels

The acceptance probability involves the ratio of probabilities at the proposed and current points:

$$\begin{aligned} \min\Big(1,\, e^{S(x^{(k)}) - S(x')}\Big) &= \min\left(1,\, \frac{e^{-S(x')}}{e^{-S(x^{(k)})}}\right) \\ &= \min\left(1,\, \frac{P(x') \cdot Z}{P(x^{(k)}) \cdot Z}\right) \\ &= \min\left(1,\, \frac{P(x')}{P(x^{(k)})}\right) \end{aligned} \tag{4}$$

The partition function Z appears in both numerator and denominator and cancels exactly. The algorithm therefore only requires evaluating $S(x)$ at individual points — it never needs to integrate $e^{-S(x)}$ to find Z.

2.5 Historical Note

The algorithm was first published in the 1953 paper "Equation of State Calculations by Fast Computing Machines" by Nicholas Metropolis, Arianna Rosenbluth, Marshall Rosenbluth, Augusta Teller and Edward Teller [1]. In their original paper, they used $U(x)$ for the action (energy) and ξ for the uniform random number r. Figure 1 shows the original flowchart from their paper. The structure is identical to what we use today — only the notation differs.

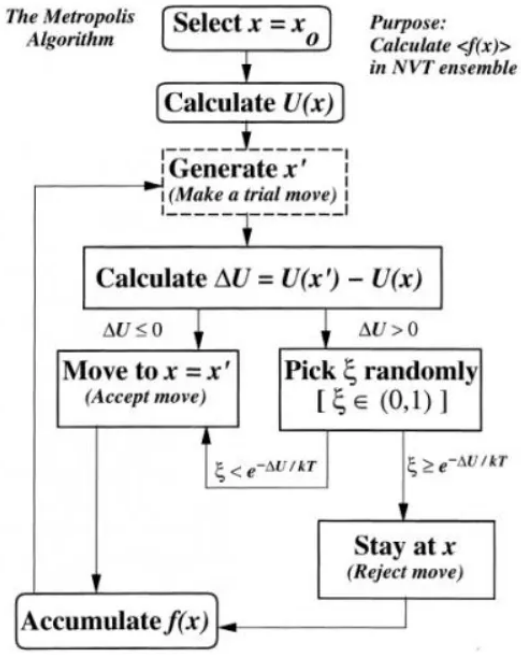

Figure 1: The original Metropolis algorithm flowchart from [1]. $U(x)$ denotes the action $S(x)$ and ξ denotes the uniform random number r.

2.6 Proof of Correctness: Detailed Balance

For the Metropolis algorithm to sample from the correct distribution $P(x)$, it must satisfy four conditions from the theory of Markov chains [2]: irreducibility, aperiodicity, positivity, and detailed balance. The first three are satisfied trivially by construction. We prove detailed balance here.

Detailed balance requires:

$$P(x)\,T(x \to x') = P(x')\,T(x' \to x) \tag{5}$$

where $T(x \to x')$ is the transition probability from x to x'.

The transition probability has two parts: the proposal probability $q(x'|x)$ and the acceptance probability $A(x \to x')$:

$$T(x \to x') = q(x'|x) \cdot A(x \to x') \tag{6}$$

For our symmetric uniform proposal $\Delta x \sim \text{Uniform}(-c, +c)$:

$$q(x'|x) = q(x|x') = \frac{1}{2c} \tag{7}$$

since the probability of proposing a jump of size $\Delta x = x' - x$ equals the probability of the reverse jump, both being $1/(2c)$.

The Metropolis acceptance probability is:

$$A(x \to x') = \min\left(1, \frac{P(x')}{P(x)}\right) \tag{8}$$

We now consider two cases.

Case 1: $P(x') \geq P(x)$.

Then $A(x \to x') = 1$ and $A(x' \to x) = P(x)/P(x')$.

Left side of (5):

$$P(x) \cdot \frac{1}{2c} \cdot 1 = \frac{P(x)}{2c} \tag{9}$$

Right side of (5):

$$P(x') \cdot \frac{1}{2c} \cdot \frac{P(x)}{P(x')} = \frac{P(x)}{2c} \tag{10}$$

Both sides are equal. ✓

Case 2: $P(x') < P(x)$.

Then $A(x \to x') = P(x')/P(x)$ and $A(x' \to x) = 1$.

Left side:

$$P(x) \cdot \frac{1}{2c} \cdot \frac{P(x')}{P(x)} = \frac{P(x')}{2c} \tag{11}$$

Right side:

$$P(x') \cdot \frac{1}{2c} \cdot 1 = \frac{P(x')}{2c} \tag{12}$$

Both sides are equal. ✓

Since detailed balance holds in both cases, the stationary distribution of the Metropolis Markov chain is $P(x)$, and the algorithm samples from the correct distribution as $K \to \infty$.

2.7 Incorrect Implementation: Asymmetric Proposal

If the proposal is asymmetric, for example $\Delta x \sim \text{Uniform}(-1/2, +1)$, then $q(x'|x) \neq q(x|x')$ in general.

Concretely: a transition from $x = 0$ to $x = 1$ is possible since $\Delta x = 1$ is in $(-1/2, +1)$. But the reverse transition from $x = 1$ to $x = 0$ requires $\Delta x = -1$, which is outside the range. So $T(0 \to 1) \neq T(1 \to 0)$ and detailed balance is violated.

Figure 2 shows the result — the histogram is skewed and does not converge to the correct distribution.

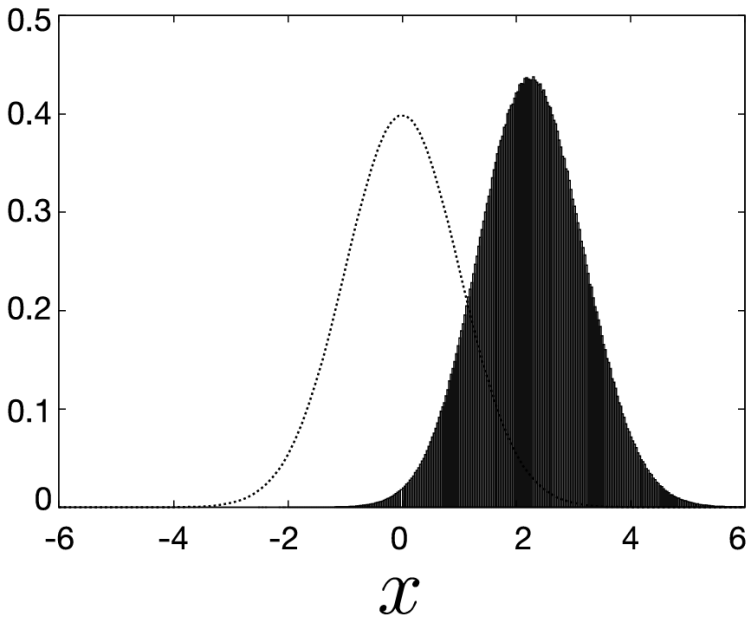

Fig. 4.3 The histogram of $K = 10^7$ configurations obtained by using the Metropolis algorithm, with a *wrong* choice $\Delta x \in \left[-\frac{1}{2}, 1\right]$. The dashed line is the target distribution $P(x) = \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$. Because the detailed balance condition is not satisfied, the target distribution is not correctly reproduced

Figure 2: Histogram resulting from the incorrect asymmetric proposal $\Delta x \sim \text{Uniform}(-1/2, +1)$ (Fig. 4.3 from [2]). The distribution is skewed and wrong because detailed balance is violated.

3 Gaussian Distribution

3.1 Setup

The standard Gaussian distribution is:

$$P(x) = \frac{e^{-x^2/2}}{\sqrt{2\pi}} \tag{13}$$

In Metropolis form, $S(x) = x^2/2$ and $Z = \sqrt{2\pi}$. The action $S(x)$ is known; Z is not needed. The true expected values are:

$$\langle x \rangle = 0, \qquad \langle x^2 \rangle = 1 \tag{14}$$

The textbook runs the simulation with initial value $x^{(0)} = 0$ and step size $c = 0.5$. The acceptance probability for a proposal $x \to x'$ is:

$$e^{S(x)-S(x')} = e^{x^2/2 - x'^2/2} \tag{15}$$

- If $|x'| < |x|$: moving closer to centre $\Rightarrow e^{(\cdot)} > 1 \Rightarrow$ always accept
- If $|x'| > |x|$: moving away from centre $\Rightarrow$ accept with probability < 1

3.2 Histogram Convergence

Figure 3 shows our R simulation histograms for $K = 10^2, 10^3, 10^5, 10^7, 10^8$. At $K = 10^2$ the histogram is completely unreliable. As K increases the histogram converges to the true Gaussian density and by $K = 10^8$ the agreement is essentially perfect.

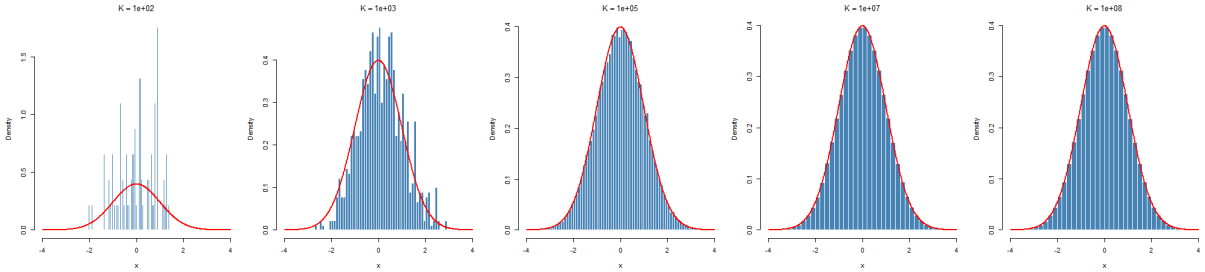

Figure 3: R simulation: histogram of Metropolis samples for the Gaussian distribution at $K = 10^2, 10^3, 10^5, 10^7, 10^8$. The red curve is the true density $P(x) = e^{-x^2/2}/\sqrt{2\pi}$. At $K = 10^2$ the histogram is unreliable; convergence is essentially perfect at $K = 10^8$.

3.3 Convergence of Expected Values

Figure 4 shows the running averages from the textbook. Both $\langle x \rangle$ and $\langle x^2 \rangle$ converge to their true values of 0 and 1 respectively.

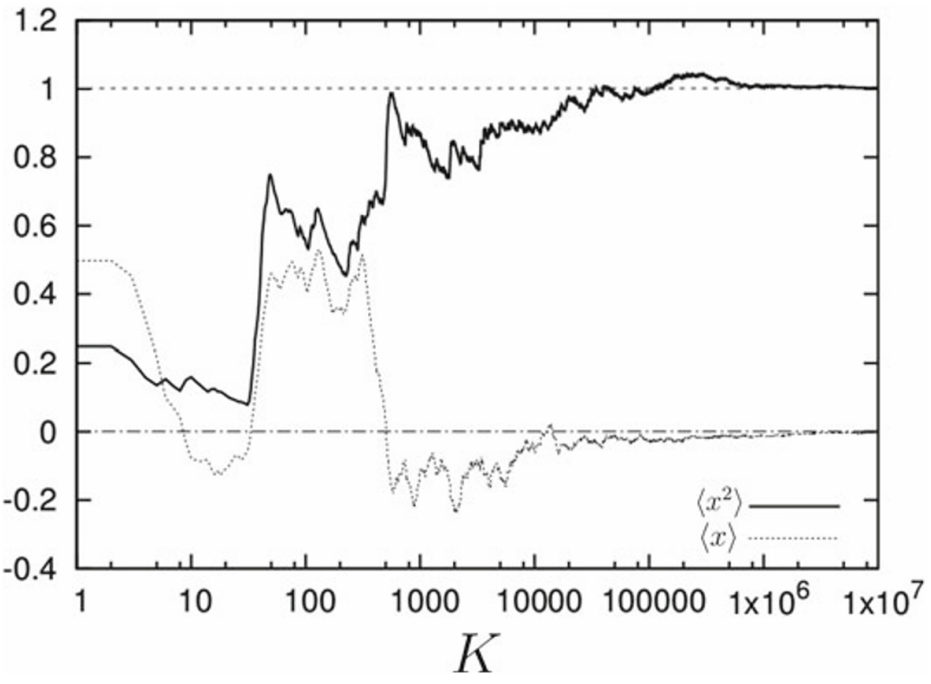

Fig. 4.2 $\langle x \rangle = \frac{1}{K}\sum_{k=1}^{K} x^{(k)}$ and $\langle x^2 \rangle = \frac{1}{K}\sum_{k=1}^{K} \left(x^{(k)}\right)^2$. They approach the correct values 0 and 1 as K increases

Figure 4: Running averages $\langle x \rangle$ and $\langle x^2 \rangle$ converging to 0 and 1 (Fig. 4.2 from [2]).

4 Beyond Gaussian — Bimodal Distribution

To demonstrate the generality of the Metropolis algorithm, the textbook applies it to a bimodal distribution: a superposition of two Gaussians centred at $x = \pm 3$:

$$P(x) = \frac{e^{-(x-3)^2/2} + e^{-(x+3)^2/2}}{2\sqrt{2\pi}} \tag{16}$$

The action for this distribution is:

$$S(x) = -\log\left(e^{-(x-3)^2/2} + e^{-(x+3)^2/2}\right) + \text{const} \tag{17}$$

Only $S(x)$ changed. The proposal step, the acceptance test, and the entire chain structure remain identical to the Gaussian case.

Figure 5 shows the histogram convergence with step sizes $c = 0.5$ and $c = 5.0$.

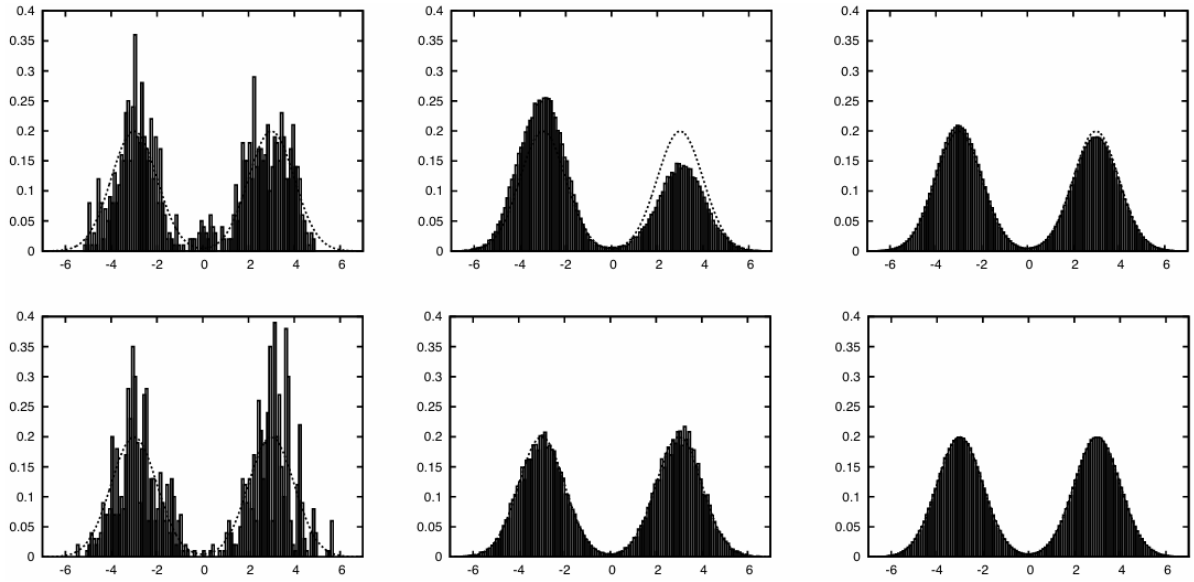

Fig. 4.9 The histograms of $x^{(1)}, x^{(2)}, \ldots, x^{(K)}$ with $K = 10^3, 10^5, 10^7$. The dashed lines are the target distribution $P(x) = \frac{e^{-\frac{(x-3)^2}{2}} + e^{-\frac{(x+3)^2}{2}}}{2\sqrt{2\pi}}$. The step size is 0.5 (top) or 5.0 (bottom). When the step size is 0.5, the deviation from the target distribution is visible even with 10^7 configurations. When the step size is 5.0, the convergence to the target distribution is much faster

Figure 5: Histogram of Metropolis samples for the bimodal distribution (Fig. 4.9 from [2]). Top row: $c = 0.5$. Bottom row: $c = 5.0$. With $c = 0.5$ the chain struggles to jump between the two modes even at $K = 10^7$. With $c = 5.0$ convergence is much faster.

This example illustrates an important practical point: for multimodal distributions, a larger step size is needed so the chain can jump between modes. With $c = 0.5$ the chain gets trapped in one mode for long periods because crossing the low-density region between the peaks is very unlikely.

5 Diagnostics

5.1 Burn-In (Thermalization)

If the chain starts far from the typical region of $P(x)$, early samples are dominated by the initial condition and should be discarded. This is called the **burn-in** period.

Figure 6 shows the chain history starting from $x^{(0)} = 100$, far from the typical region $|x| \sim 1$ of the Gaussian.

Fig. 4.4 The history of the simulation of the Gaussian distribution via the Metropolis algorithm. The step size is $c = 0.5$. Because we chose the initial value to be $x = 100$, which is very far from the typical values, the strong correlation with the initial value remains for a while, and it took a lot of steps to reach the typical values $|x| \sim 1$

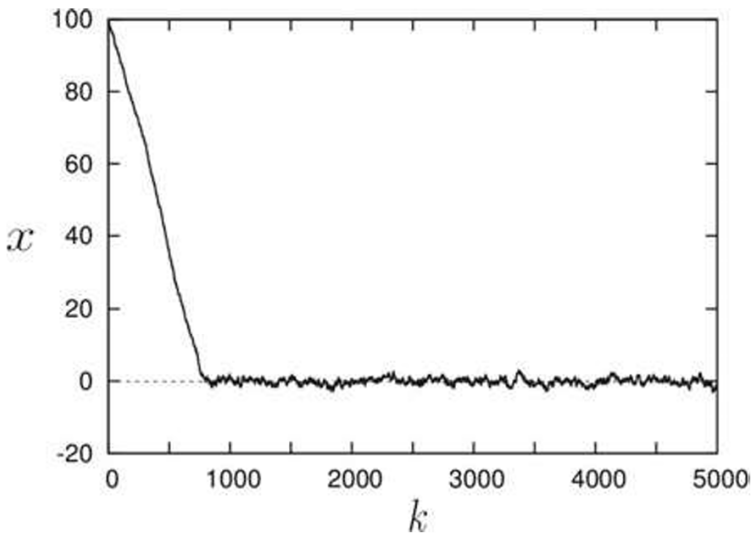

Figure 6: Chain history starting from $x^{(0)} = 100$ (Fig. 4.4 from [2]). The chain takes ~ 800 steps to reach the typical region.

How to choose K_{cut}: Plot $\langle x^2 \rangle$ as a function of K_{cut}. At small K_{cut} the value changes as we vary the cutoff because the initial condition still has influence. At sufficiently large K_{cut} the value stops changing — this is the safe discard point.

A useful physical analogy is ice in water. Until the ice melts, heat flows from water to ice and the system is non-uniform — the initial condition still matters. After the ice melts, temperature is uniform everywhere — this is the thermalized state. The Markov chain behaves the same way. In Fig. 4.4, thermalization is achieved by $k \approx 800$. Discarding $k \leq 1000$ is safe.

The word **thermalization** has two meanings:

1. **First meaning:** The chain has escaped the influence of the initial condition $x^{(0)}$.
2. **Second meaning:** Enough independent samples have been collected so that running averages stabilize.

These are different because even after burn-in, consecutive samples are still autocorrelated. With K samples and autocorrelation length τ, the effective independent count is

only K/τ. If this is small, $\langle x^2 \rangle$ still fluctuates significantly. The simulation is thermalized in the second sense when $\langle x^2 \rangle$ barely changes as more samples are added.

5.2 Autocorrelation

Consecutive samples in a Metropolis chain are not independent since each proposal is built from the current position. This **autocorrelation** causes the standard error formula to underestimate the true statistical error.

The autocorrelation function is defined as:

$$\rho(\tau) = \frac{\langle x^{(k)} x^{(k+\tau)} \rangle - \langle x \rangle^2}{\langle x^2 \rangle - \langle x \rangle^2} \tag{18}$$

At lag $\tau = 0$, $\rho(0) = 1$ always. For a well-behaved chain, $\rho(\tau) \to 0$ as $\tau \to \infty$. The **autocorrelation length** τ_c is the lag at which $\rho(\tau) \approx 0$. Two samples separated by fewer than τ_c steps cannot be treated as independent.

Figure 7 shows that strong correlations survive for at least 20–30 steps in the textbook example.

Fig. 4.5 The zoom-in of Fig. 4.4, from $k = 1000$ to $k = 2000$. Strong correlations survive at least for 20 or 30 steps

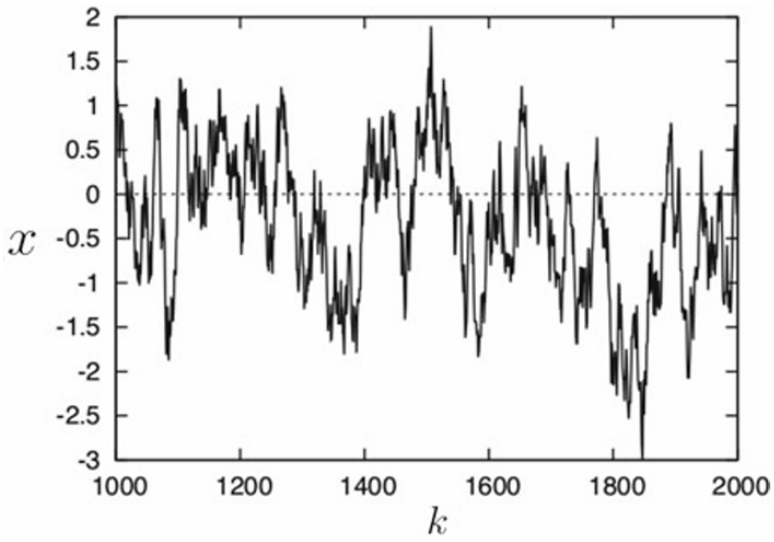

Figure 7: Zoom of chain history from $k = 1000$ to $k = 2000$ (Fig. 4.5 from [2]), showing correlations surviving for 20–30 steps. From $K = 1000$ samples with $\tau = 25$: only ≈ 40 truly independent samples.

5.3 Jackknife Method

The Jackknife method estimates both the autocorrelation length τ_c and the true statistical error simultaneously.

Divide the post-burn-in samples into n blocks of size w. The average of $f(x)$ in block

l:

$$\tilde{f}^{(l,w)} = \frac{1}{w} \sum_{j=(l-1)w+1}^{lw} f\left(x^{(j)}\right) \tag{19}$$

The Jackknife error is:

$$\Delta_w = \sqrt{\frac{1}{n(n-1)} \sum_{l=1}^{n} \left(\tilde{f}^{(l,w)} - \bar{f}\right)^2} \tag{20}$$

where $\bar{f}$ is the overall mean.

As w increases:

- $w < \tau_c$: blocks still correlated $\Rightarrow$ Δ_w increases
- $w \approx \tau_c$: blocks become independent $\Rightarrow$ Δ_w plateaus
- $w > \tau_c$: Δ_w stays constant

Using Δ_1 (no grouping) as the error underestimates the true error by a factor of $\sqrt{w_c}$.

Figure 8 shows the Jackknife plot from the textbook. The plateau occurs at $w_c \approx 40$.

Fig. 4.6 The expectation value of x^2 and the Jackknife error

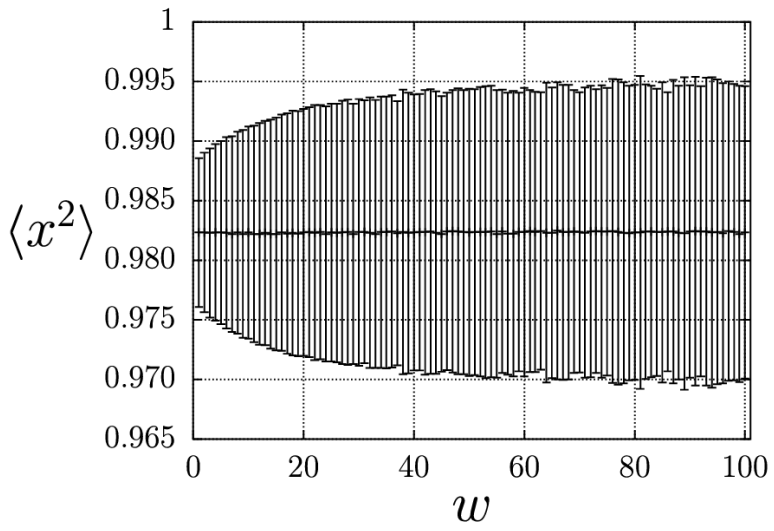

Figure 8: Jackknife error Δ_w vs block size w (Fig. 4.6 from [2]). The plateau gives $w_c \approx 40$.

Figure 9 shows the final result with $w = 50$ safely above $w_c \approx 40$. The final estimate is:

$$\langle x^2 \rangle = 0.982 \pm 0.012 \qquad \text{(true value} = 1 \checkmark\text{)} \tag{21}$$

This is how MCMC results should be correctly reported with proper error bars.

Fig. 4.7 The average of each group $\tilde{f}^{(l,w)}$, with the width $w = 50$

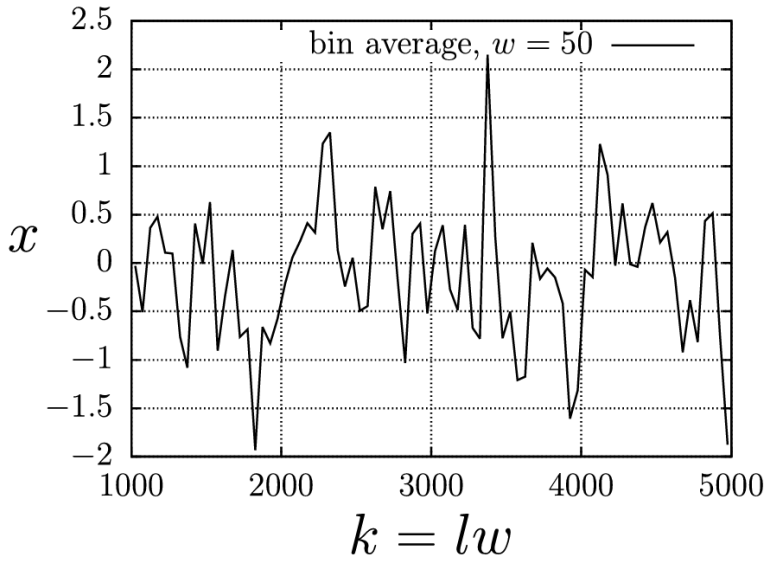

Figure 9: Group averages $\tilde{f}^{(l,w)}$ with $w = 50$ (Fig. 4.7 from [2]). Final estimate: $\langle x^2 \rangle = 0.982 \pm 0.012$.

6 Step Size Tuning

6.1 Effect of Step Size

The step size c controls how far each proposal jumps. Both extremes give poor performance:

- c **too small:** Almost every proposal is accepted but each move is tiny. The chain barely explores the distribution and autocorrelation is very large.
- c **too large:** Most proposals land in low-probability regions and are rejected. The chain is almost stuck and autocorrelation is also very large.

6.2 Random Walk Argument

The Metropolis proposal is a random walk with step size c. After n steps the displacement is approximately $c\sqrt{n}$. To explore the typical region of the standard Gaussian (size ~ 1):

$$c\sqrt{n} \sim 1 \implies n \sim \frac{1}{c^2} \tag{22}$$

Therefore:

$$\tau_c \propto \frac{1}{c^2} \tag{23}$$

Larger c reduces τ_c — but only until the acceptance rate becomes too low. This tension produces an optimal c.

6.3 Acceptance Rate Table

Table 1 shows acceptance rates at different step sizes for the Gaussian distribution from the textbook.

Table 1: Acceptance rates for the Gaussian distribution (Table 4.1 from [2])

Step size c	Acceptance rate	$c\times$ A.R.
0.5	0.9077	0.454
1.0	0.8098	0.810
2.0	0.6281	1.256
3.0	0.4864	1.459
4.0	0.3911	1.564
6.0	0.2643	1.586
8.0	0.1993	1.594

At $c = 0.5$ and $c = 1.0$ the acceptance rate is high, indicating c is too small. For $c > 4$ the product $c \times$ A.R. is nearly constant, indicating c is too large. The sweet spot is $c = 2.0$ to 4.0, giving acceptance rate 30%–80%.

The practical rule is: run a short test simulation and measure the acceptance rate. If A.R. $> 80\%$ increase c. If A.R. $< 30\%$ decrease c.

6.4 Convergence for Different Step Sizes

Figure 10 shows $\langle x^2 \rangle$ converging for different values of c from the textbook.

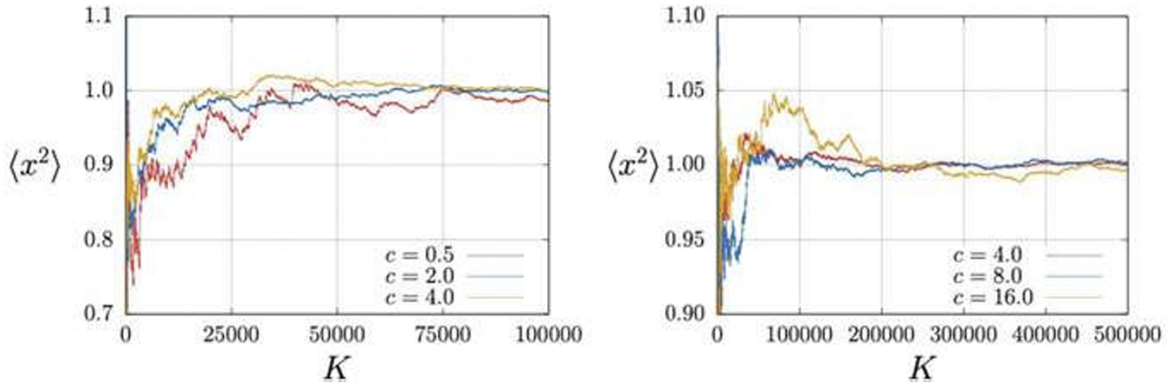

Fig. 4.8 $\langle x^2 \rangle = \frac{1}{K} \sum_{k=1}^{K} \left(x^{(k)}\right)^2$ with several different choices of the step size c. The simulation is not efficient when the step size is too big or too small

Figure 10: Convergence of $\langle x^2 \rangle$ for different step sizes (Fig. 4.8 from [2]). The values $c = 2.0$ and $c = 4.0$ converge fastest. Both $c = 0.5$ (too small) and $c = 8.0$ (too large) converge much more slowly.

7 Box-Muller as a Special Case of MCMC

The Box-Muller method generates standard Gaussian samples directly using:

$$Z_1 = \sqrt{-2\ln U_1}\cos(2\pi U_2), \qquad Z_2 = \sqrt{-2\ln U_1}\sin(2\pi U_2) \tag{24}$$

where $U_1, U_2 \sim \text{Uniform}(0,1)$.

This can be regarded as a special case of MCMC where the transition probability is $T(x \to x') = P(x')$. Each new sample is drawn independently from $P(x)$ regardless of the current position.

Detailed balance holds trivially:

$$P(x) \cdot T(x \to x') = P(x) \cdot P(x') = P(x') \cdot T(x' \to x) \tag{25}$$

All four MCMC conditions are satisfied and the autocorrelation length is $\tau_c = 1$ — every sample is independent of the previous one.

Table 2: Comparison of Box-Muller and Metropolis

	Box-Muller	**Metropolis**
Autocorrelation	Zero ($\tau_c = 1$)	$\tau_c > 0$
Works for	Gaussian only	Any $P(x)$
Needs Z	Yes	No

Box-Muller represents the ideal extreme of MCMC — zero autocorrelation — but is limited to the Gaussian. Metropolis trades some autocorrelation for complete generality.

8 Exponential Distribution — R Simulation

8.1 Motivation

As an original contribution of this project, we apply the Metropolis algorithm to the Exponential distribution — an example not present in the reference textbook. This demonstrates that only the action $S(x)$ needs to change; the entire algorithm structure remains identical.

8.2 Setup

The Exponential distribution is:

$$P(x) = e^{-x}, \quad x \geq 0 \tag{26}$$

In Metropolis form:

$$S(x) = \begin{cases} x & \text{if } x \geq 0 \\ \infty & \text{if } x < 0 \end{cases} \tag{27}$$

Proposals that fall below zero are automatically rejected since $S(x) = \infty$ there. The normalising constant is $Z = 1$ but is never needed. The true expected values are:

$$\langle x \rangle = 1, \qquad \langle x^2 \rangle = 2 \tag{28}$$

We run with initial value $x^{(0)} = 1$ and step size $c = 1.0$, giving approximately 63% acceptance rate — within the optimal range.

8.3 Histogram Convergence

Figure 11 shows our simulated histograms for $K = 10^2, 10^3, 10^5, 10^7$. The shape is completely different from the Gaussian — one-sided and decaying monotonically — yet the algorithm handles it without any structural change.

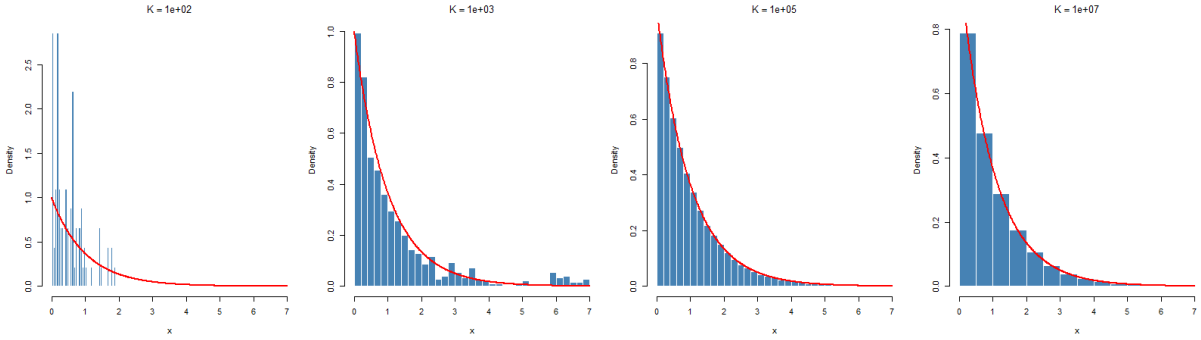

Figure 11: R simulation: histogram of Metropolis samples for the Exponential distribution at $K = 10^2, 10^3, 10^5, 10^7$. The red curve is the true density $P(x) = e^{-x}$. The algorithm converges correctly despite the completely different shape from the Gaussian.

8.4 Convergence of Expected Values

Figure 12 shows the running averages for the Exponential distribution. The convergence is slower than the Gaussian because the autocorrelation length is larger.

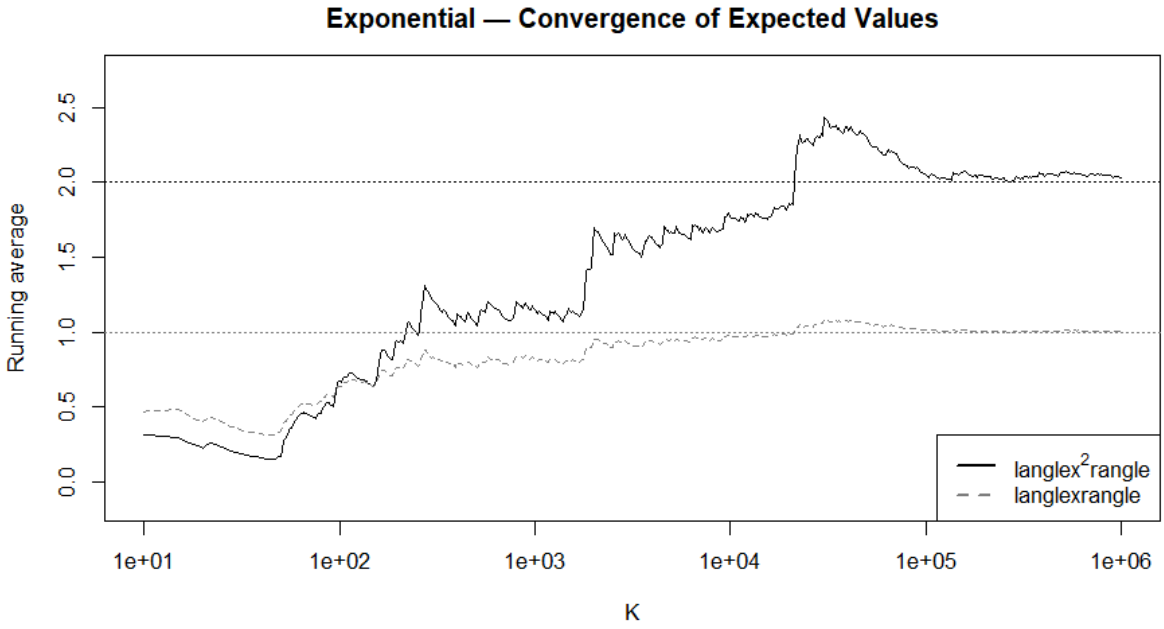

Figure 12: R simulation: running averages $\langle x \rangle$ and $\langle x^2 \rangle$ for the Exponential distribution converging to their true values 1 and 2.

8.5 Burn-In

Figure 13 shows our simulated chain history starting from $x^{(0)} = 15$. The chain reaches the typical region within a few hundred steps.

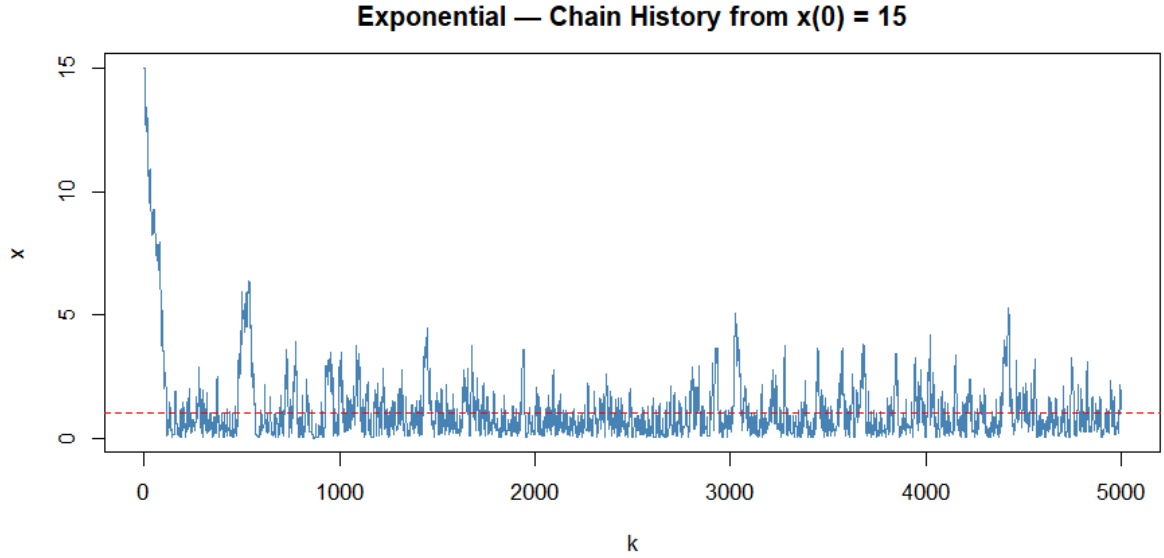

Figure 13: R simulation: Exponential chain history from $x^{(0)} = 15$. The red dashed line marks the true mean $\langle x \rangle = 1$.

8.6 Autocorrelation

Figure 14 shows the ACF plot for the Exponential distribution. The ACF drops to zero by lag ≈ 80, compared to lag ≈ 15 for the Gaussian. This means the Exponential requires approximately 5 times more steps between independent samples.

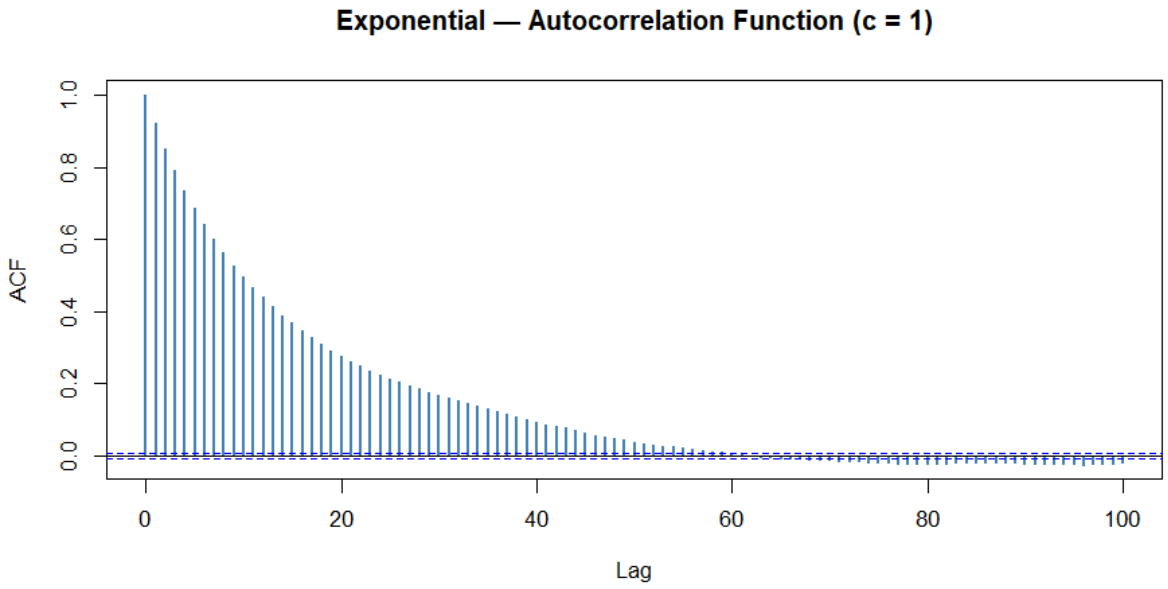

Figure 14: R simulation: ACF for the Exponential ($c = 1.0$). Drops to zero by lag ≈ 80. Much larger autocorrelation than the Gaussian.

8.7 Jackknife

Figure 15 shows the Jackknife plot for the Exponential. The Δ_w plateaus at $w_c \approx 100$, consistent with the longer ACF decay.

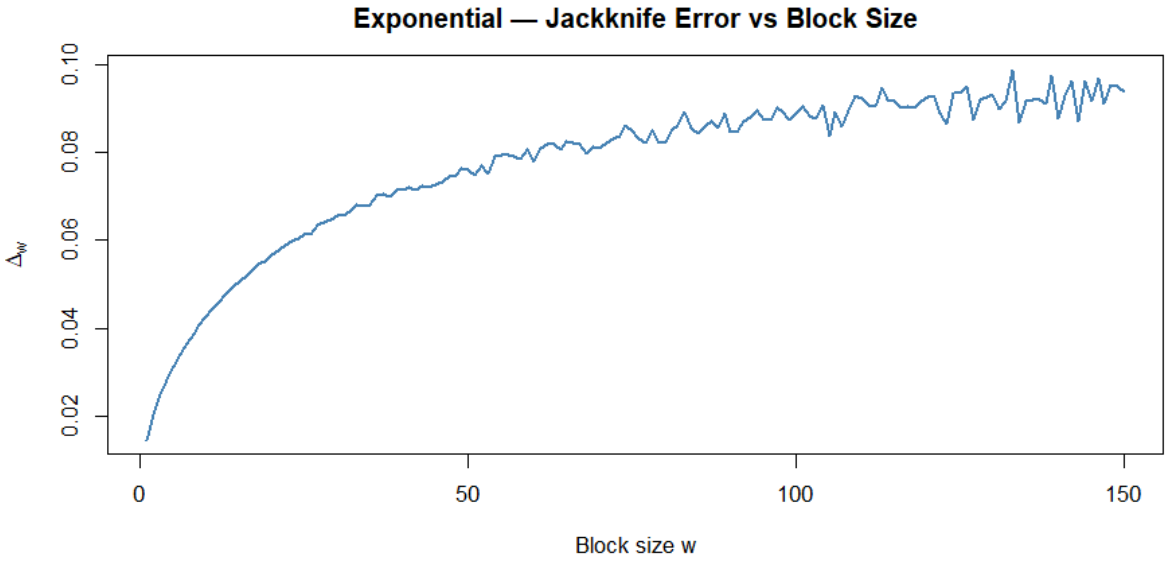

Figure 15: R simulation: Jackknife error Δ_w for the Exponential. Plateau at $w_c \approx 100$, confirming the autocorrelation length.

8.8 Acceptance Rate Table

Table 3 shows acceptance rates at different step sizes for the Exponential distribution from our simulation.

Table 3: Acceptance rates for the Exponential distribution from our R simulation (10,000 samples, $x^{(0)} = 1$)

Step size c	Acceptance rate	$c\times$ A.R.
0.2	0.9087	0.1817
0.5	0.7901	0.3951
1.0	0.6322	0.6322
2.0	0.4357	0.8714
3.0	0.3272	0.9816
4.0	0.2542	1.0168

For $c > 2$ the product $c \times$ A.R. is nearly constant, indicating c is too large. The optimal range is $c = 1.0$ to $c = 2.0$.

8.9 Step Size Comparison

Figure 16 shows $\langle x^2 \rangle$ converging for three values of c. The intermediate value $c = 1.0$ converges fastest, confirming the random walk argument.

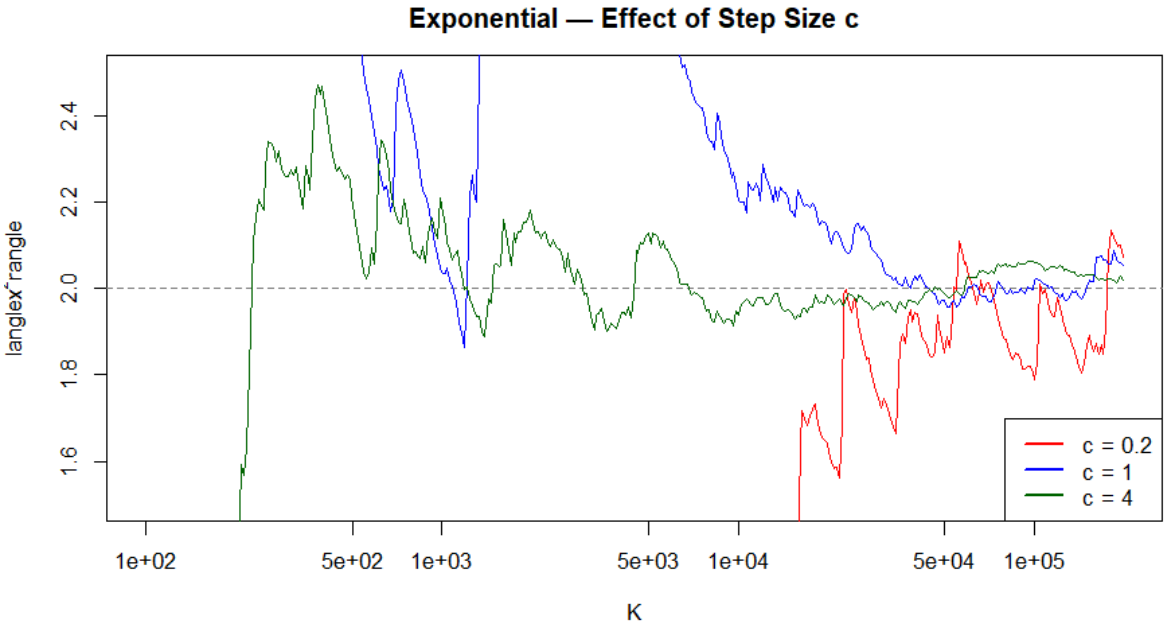

Figure 16: R simulation: $\langle x^2 \rangle$ for three step sizes in the Exponential. $c = 1.0$ (blue) converges fastest. $c = 0.2$ (red) is too slow. $c = 4.0$ (green) is mostly rejected.

9 Gaussian Verification via R Simulation

In this section we verify the textbook Gaussian results using our own R simulations. All plots here were generated independently using our R code and should be compared with the textbook figures shown in Sections 3, 5, and 6.

9.1 Running Mean Convergence

Figure 17 shows our R simulation of the running averages $\langle x \rangle$ and $\langle x^2 \rangle$ for the Gaussian. Both converge to 0 and 1 respectively, consistent with Fig. 4.2 from the textbook shown earlier in Section 3.

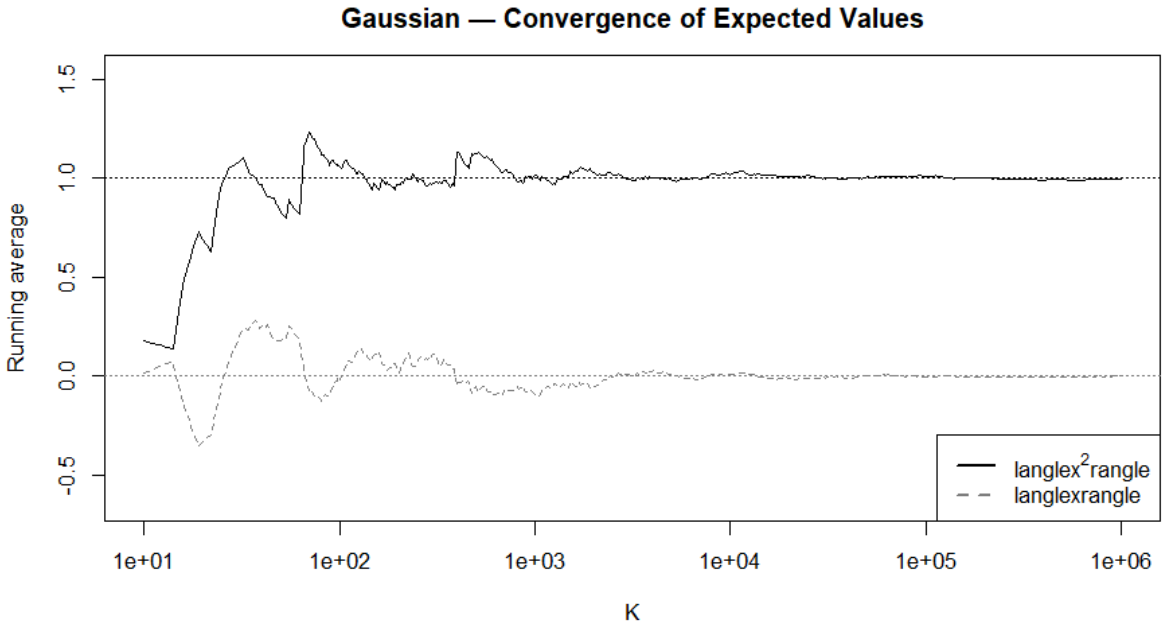

Figure 17: R simulation: running averages $\langle x \rangle$ and $\langle x^2 \rangle$ for the Gaussian distribution converging to their true values 0 and 1. Compare with Fig. 4.2 from [2] in Section 5.

9.2 Burn-In

Figure 18 shows our R simulation chain history starting from $x^{(0)} = 20$. The chain descends to the typical region within approximately 100 steps, consistent with the textbook result in Fig. 4.4 shown in Section 5.

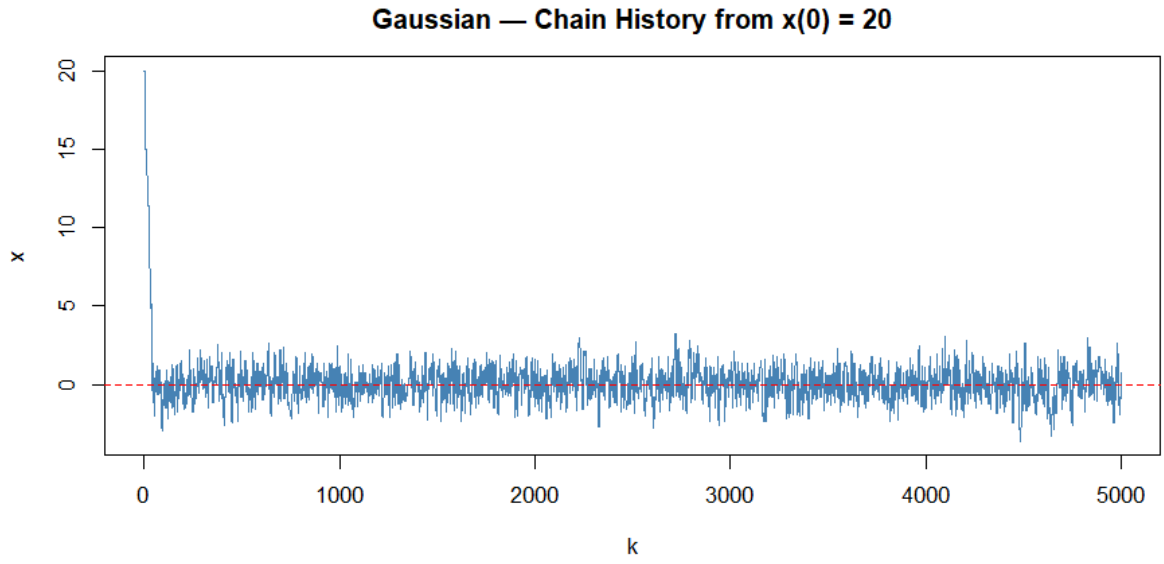

Figure 18: R simulation: Gaussian chain history from $x^{(0)} = 20$. The red dashed line marks the true mean $\langle x \rangle = 0$. Compare with Fig. 4.4 from [2] in Section 5.

9.3 Autocorrelation

Figure 19 shows our R simulation ACF plot for the Gaussian with $c = 2.0$. The ACF drops to zero by lag ≈ 15, consistent with the textbook result in Fig. 4.5 which showed correlations surviving for 20–30 steps.

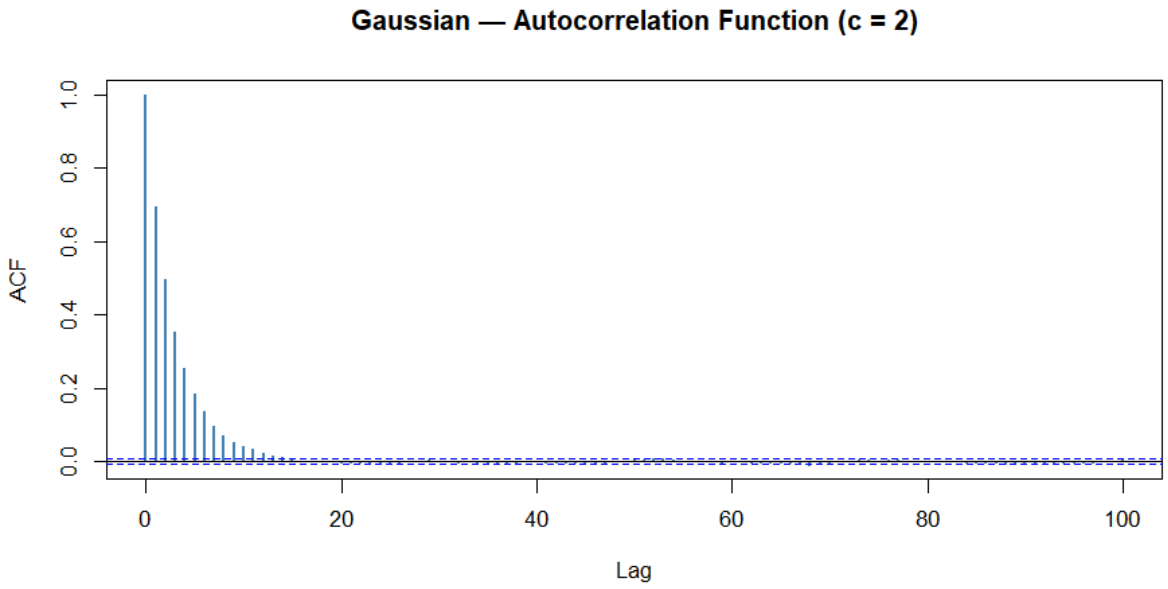

Figure 19: R simulation: ACF for the Gaussian ($c = 2.0$). Drops to zero by lag ≈ 15. Compare with Fig. 4.5 from [2] in Section 5.

9.4 Jackknife

Figure 20 shows our R simulation Jackknife plot for the Gaussian. The Δ_w plateaus at $w_c \approx 20$, which is consistent with the textbook result in Fig. 4.6 where the plateau was at $w_c \approx 40$.

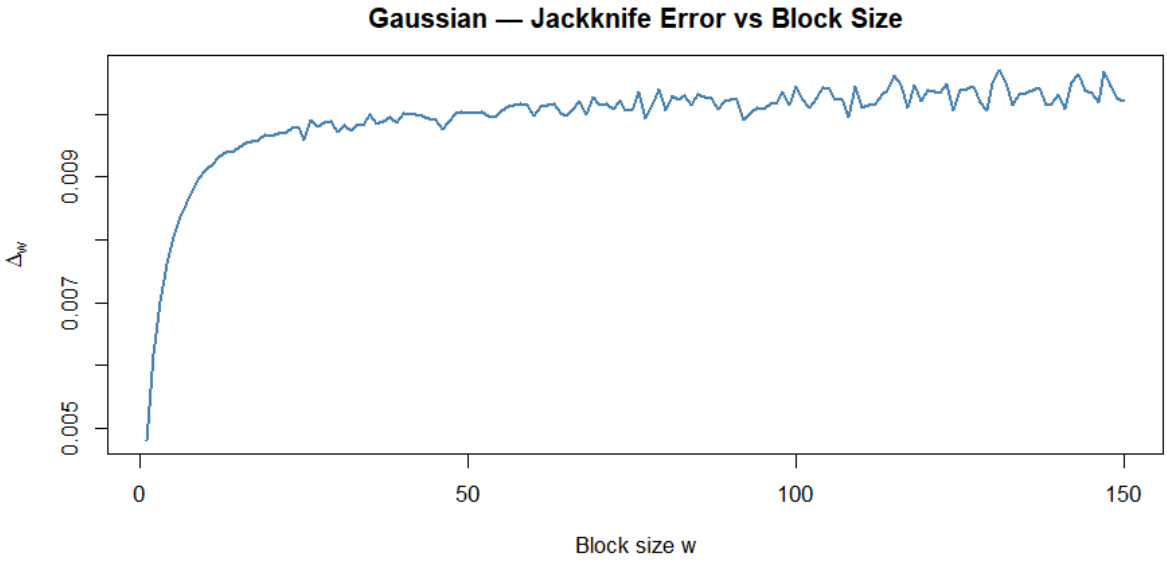

Figure 20: R simulation: Jackknife error Δ_w for the Gaussian. Plateau at $w_c \approx 20$. Compare with Fig. 4.6 from [2] in Section 5.

9.5 Step Size Comparison

Figure 21 shows our R simulation of $\langle x^2 \rangle$ converging for three values of c. The intermediate value $c = 2.0$ converges fastest, consistent with the textbook result in Fig. 4.8 shown in Section 6.

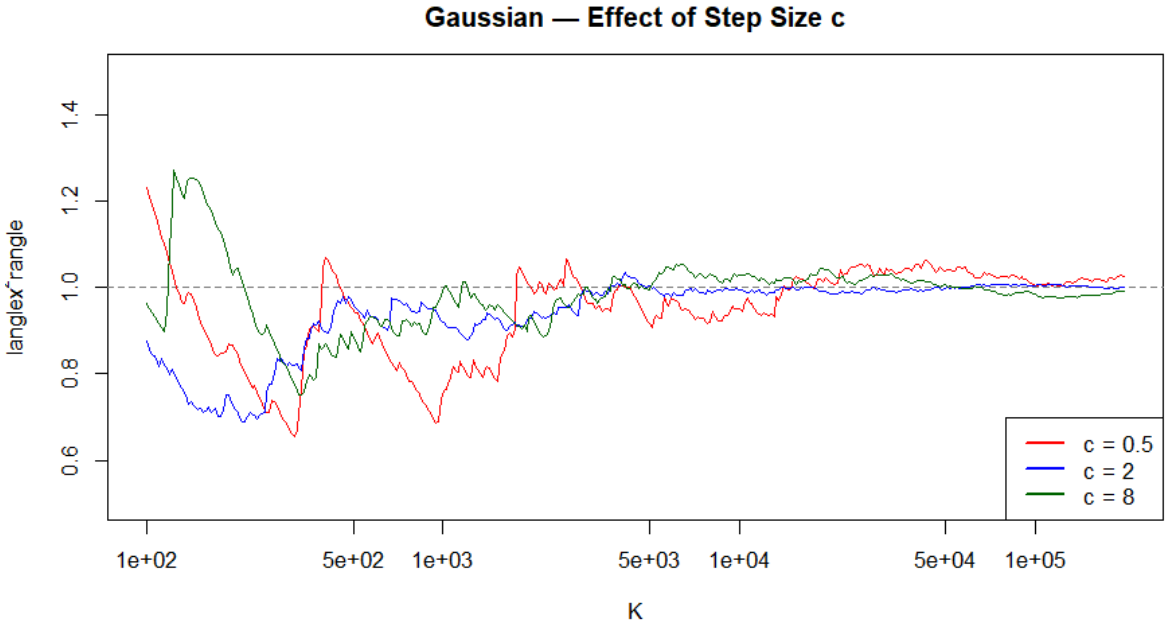

Figure 21: R simulation: $\langle x^2 \rangle$ for three step sizes in the Gaussian. $c = 2.0$ (blue) converges fastest. $c = 0.5$ (red) is too slow. $c = 8.0$ (green) is mostly rejected. Compare with Fig. 4.8 from [2] in Section 6.

10 Conclusion

We have studied the Metropolis algorithm in detail and verified its correctness through simulation. The key findings are:

1. The algorithm samples from $P(x) = e^{-S(x)}/Z$ without ever computing Z, because Z cancels in the acceptance ratio.

2. Detailed balance is satisfied by construction when the proposal is symmetric, guaranteeing convergence to the correct target distribution. An asymmetric proposal violates detailed balance and produces incorrect results.

3. Histogram and expectation values converge to their true targets as $K \to \infty$, verified for the Gaussian, bimodal, and Exponential distributions. At $K = 10^2$ the histogram is clearly unreliable, demonstrating the need for sufficiently many samples.

4. Our R simulations reproduce all textbook Gaussian results, confirming the correctness of our implementation.

5. The autocorrelation length is larger for the Exponential ($\tau_c \approx 100$) than the Gaussian ($\tau_c \approx 20$) at their respective optimal step sizes.

6. The Jackknife method correctly estimates the autocorrelation length and gives the true statistical error. Using Δ_1 without grouping underestimates the error by $\sqrt{w_c}$.

7. The optimal step size gives 30%–80% acceptance rate, minimising $\tau_c \propto 1/c^2$.

8. The algorithm generalises directly to any distribution by changing only $S(x)$, demonstrated on the bimodal and Exponential distributions.

References

[1] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, *Equation of State Calculations by Fast Computing Machines*, Journal of Chemical Physics, 21(6):1087–1092, 1953.

[2] M. Hanada and S. Matsuura, *MCMC from Scratch*, Springer Nature, 2022.