Lab4

Cui Qingxuan, Nisal Amashan

2025-02-17

Contents

1 Collaborations

Cui Qingxuan: Responsible for the question 1.

Nisal Amashan: Responsible for the question 2.

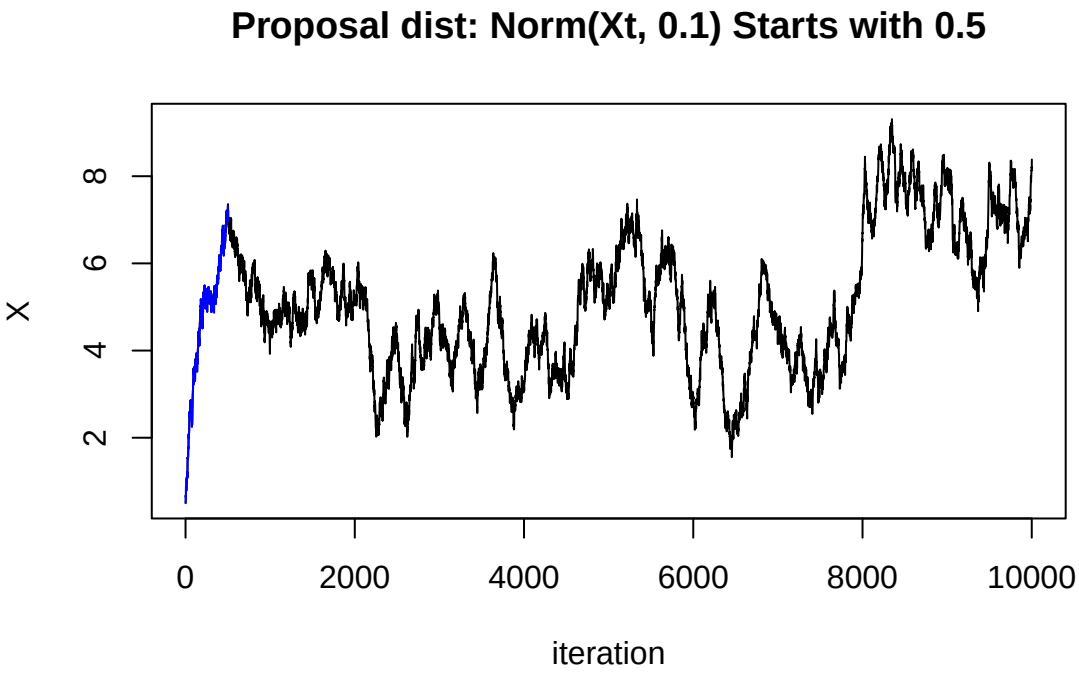
2 Question 1

2.1 Generate 1000 Samples Using Metropolis-Hastings algorithm

The Probability Density Function:

$$f(x) = 120x^5e^{-x}, \quad x > 0$$

2.1.1 Plot the Chain



2.1.2 Discussion

What can you guess about the convergence of the chain?

The Markov chain appears to have reached its stationary distribution after an initial burn-in phase. In the first few hundred iterations (shown in blue), the chain rapidly moves from its starting value (0.5) toward a region where it fluctuates more stably. After this, the chain continues to explore the state space without obvious trends or drifts, suggesting that it has mixed well.

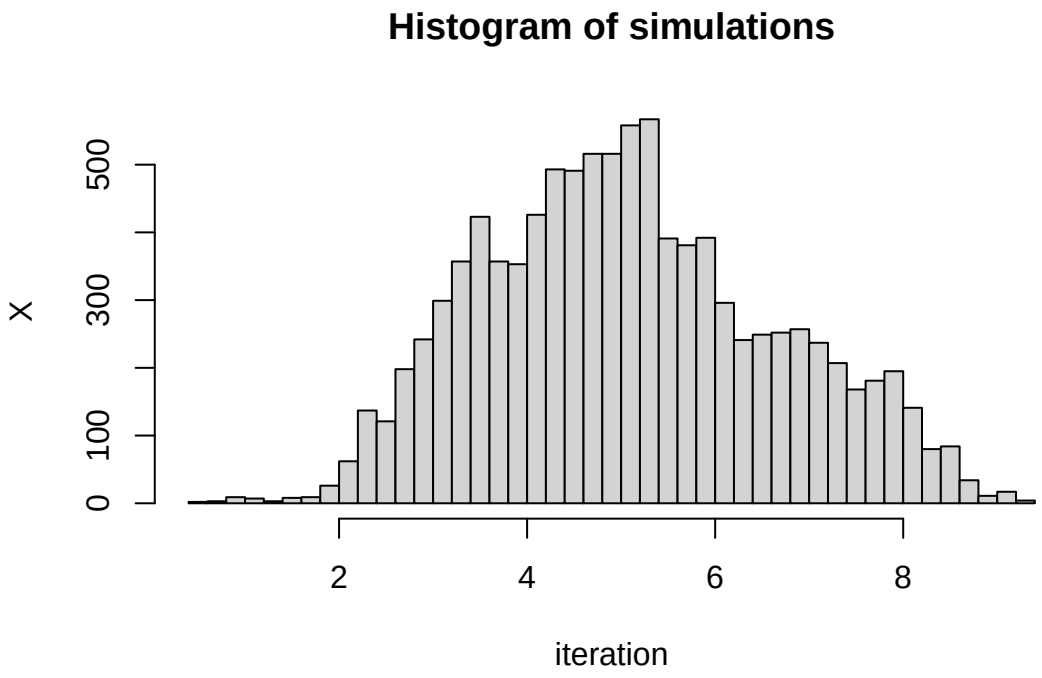
If there is a burn-in period, what can be the size of this period?

According to the plot, the simulated X-value stops increasing and starts fluctuating after 500 iterations, so we consider that simulated values are in a burn-in period before 500 iterations.

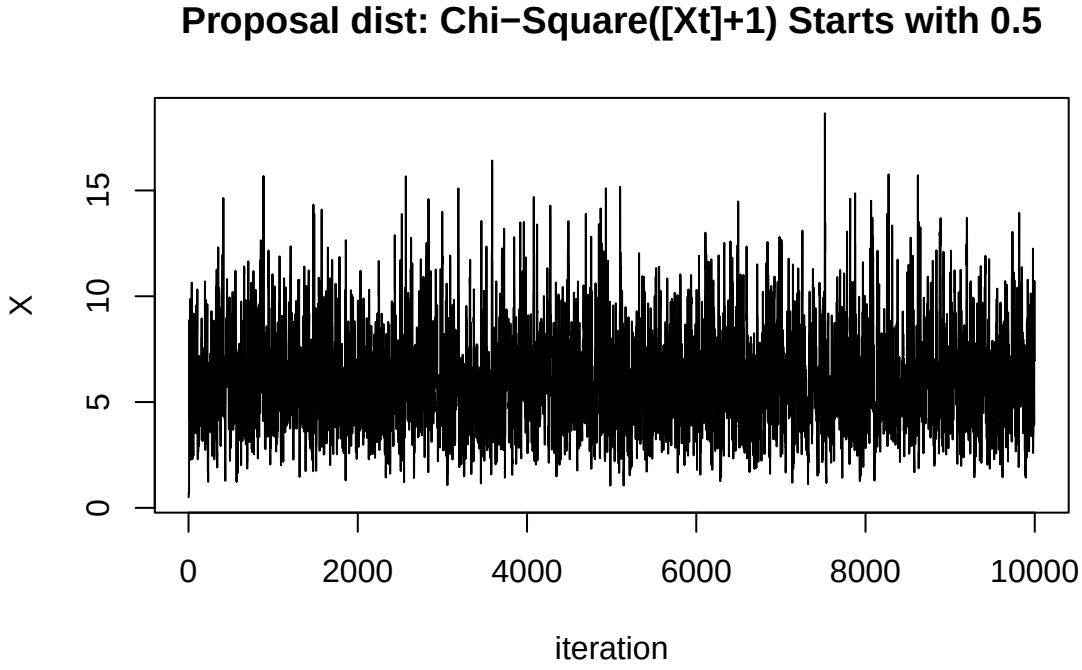
What is the accpetance rate?

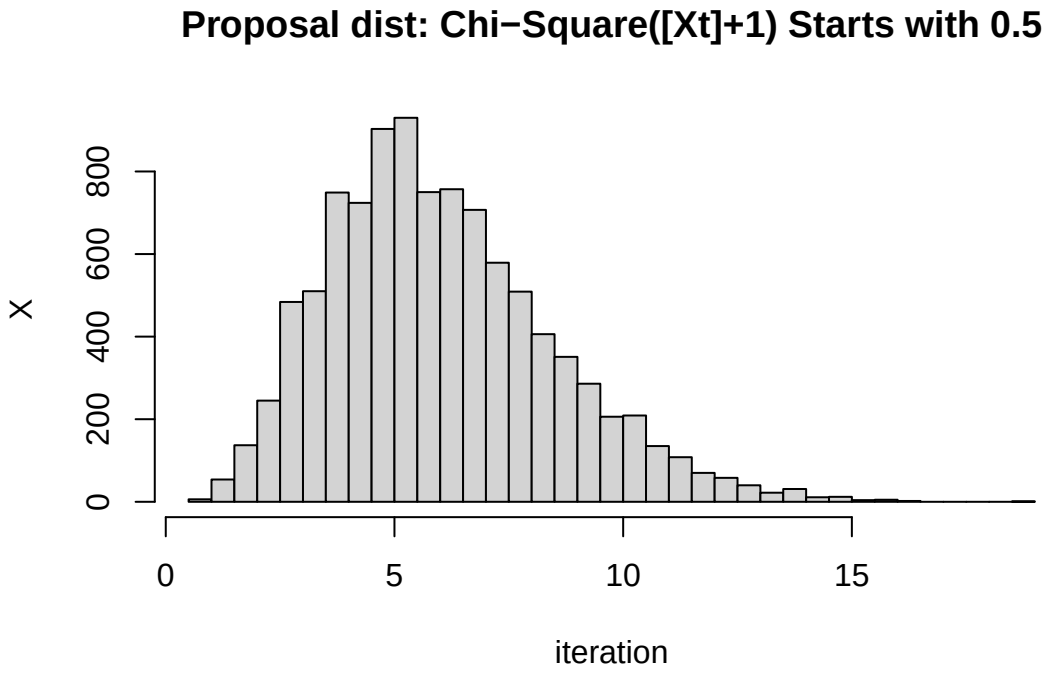
Accpetance rate is 98.59%

2.1.3 Plot the Histogram

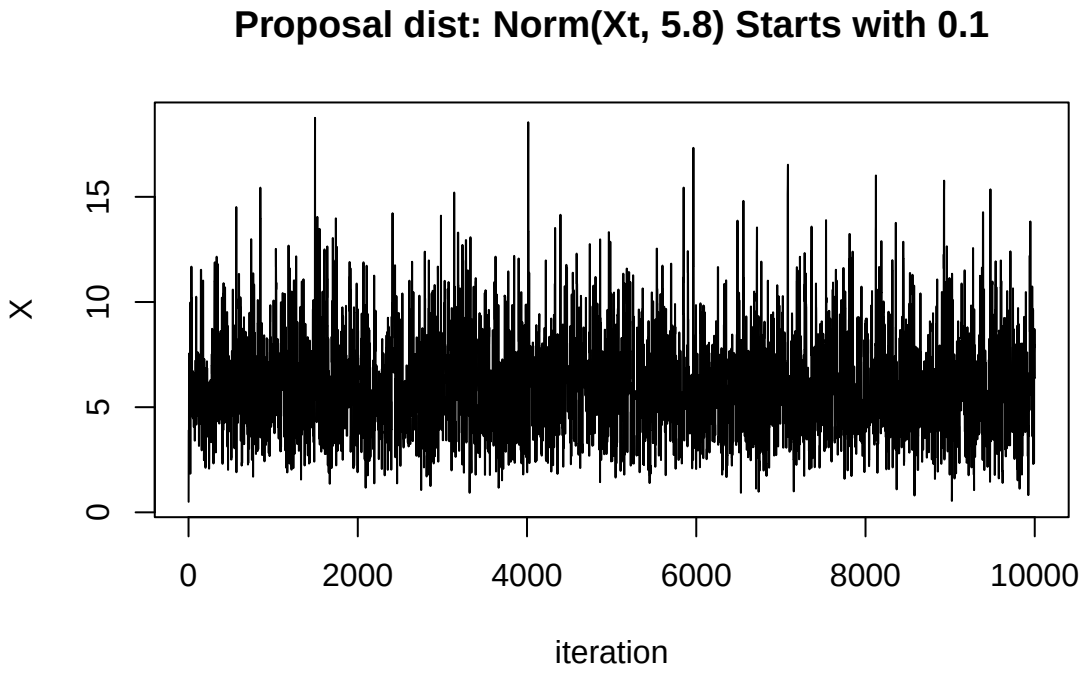


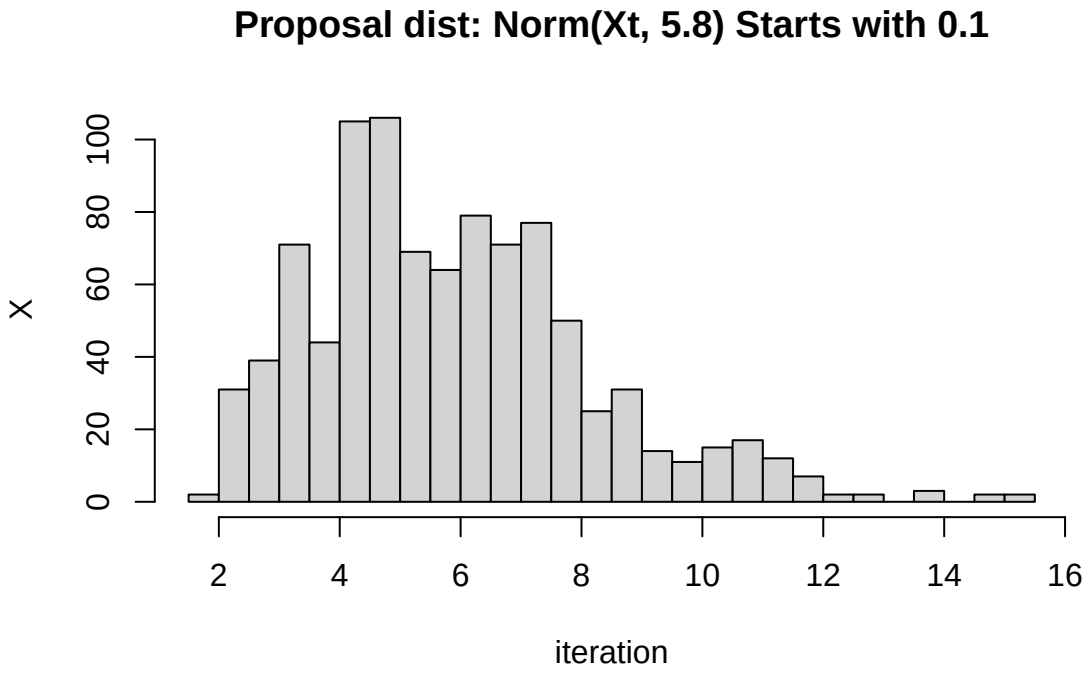
2.2 Using Chi-square Distribution $\chi^2_{\lfloor x_t+1 \rfloor}$ as Proposal Distribution





2.3 Using $\mathcal{N}(x_t, 5.8)$ as Proposal Distribution





2.4 Compare the Results and Conclude

Simulations Differ from Their Proposal Distribution:

$\mathcal{N}(x_t, 0.1)$:

The standard deviation is very small (0.1), meaning that each proposed new state differs very little from the current state. This can lead to an extremely high acceptance rate (98.59%), but the new state has a limited range of exploration and may get stuck in a localised region that does not effectively explore the entire target distribution. In this case, although the computational efficiency is high (high acceptance rate), the poor mixing (mixing) of the samples leads to slow convergence or failure to converge to the true distribution.

$\chi^2_{\lfloor x_t+1 \rfloor}$:

The chi-square distribution is asymmetric and its shape depends on the degrees of freedom. The implement would be more complicated and the acceptance rate of 60.24 percent is still high, indicating a low match between the proposal distribution and the target distribution.

$\mathcal{N}(x_t, 5.8)$:

The standard deviation is high (5.8), with an acceptance rate of 43.06 percent. This acceptance rate is close to the 20-50 percent range suggested by theory, and especially close to the theoretical optimum of 23 percent. The larger standard deviation allows for wider exploration, and although the acceptance rate is slightly lower, the sample is better mixed, allowing for more effective coverage of different regions of the target distribution and improving the rate of convergence.

Table 1: Compare the accpetance rate

	norm(xt, 0.1)	chi-square([xt]+1)	norm(xt, 5.8)
accptance_rate	98.59%	60.49%	43.06%

2.5 Estimate the Expectation

Table 2: Compare the estimation of expectation

	norm(xt, 0.1)	chi-square([xt]+1)	norm(xt, 5.8)
expectation	5.34	6	5.87

2.6 Define the Expectation of the Gamma Distribution and Comparison

Based on the given information, we can know that $f(x) = 120x^5e^{-x}, \quad x > 0$ is the probability density function of Gamma Distribution, with $\alpha = 6, \beta = 1$, so $E[X] = \frac{\alpha}{\beta} = \frac{6}{1} = 6$.

From Table2, we can see that, except for the first simulation where the estimated expectation, i.e., the proposal distribution is normally distributed with 0.1 standard deviation, which deviates from the true expectation, the performance of the other two simulations is very close to the true value, although the acceptance rate of the simulation with chi-square as the proposal distribution is higher than that of the optimal interval [23.4%, 44%].

3 Question 2

4 Appendix

```
# target function
target = function(x){
  return(120*exp(-x)*x**5)
}

proposal = function(x, mu){
  dev = 0.1
  return(exp(-(x-mu)**2/(2*dev**2))/(sqrt(2*pi)*dev))
}

# MH method, its proposal distribution is norm distribution
# return 3 values: iteration, X, reject rate
MH_norm = function(x0, n, dev=0.1){
  if (!exists(".Random.seed", envir = .GlobalEnv)) {
    set.seed(123)
  }
  reject = 0
  x = rep(0, n+1)
  x[1] = x0
  set.seed(123)
  for(t in 1:n){
    # sample from proposal
    # proposal distribution is sysmmetric
    x_star = rnorm(1,x[t], dev)
    R = target(x_star)/target(x[t])
    u = runif(1)
    if(u < R) x[t+1] = x_star
    else{
      x[t+1] = x[t]
```

```
      reject = reject + 1
    }
  }

  return (list(c(1:(n+1)), x, reject*100/n))
}

MH_chi = function(x0, n){
  set.seed(12345)
  reject = 0
  x = rep(0, n+1)
  x[1] = x0
  for(t in 1:n){
    # sample from proposal
    # proposal distribution is not sysmmetric
    # Ratio Target required
    # Ratio Proposal required
    x_star =rchisq(1, floor(x[t])+1)
    Rt = target(x_star)/target(x[t])
    Rp = dchisq(x[t], df = floor(x_star)+1) / dchisq(x_star, df = floor(x[t])+1)

    alpha = min(1, Rt * Rp)
    u = runif(1)
    if(u < alpha) x[t+1] = x_star
    else{
      x[t+1] = x[t]
      reject = reject + 1
    }
  }
  return (list(c(1:(n+1)), x, reject*100/n))
}

n=10000
res = MH_norm(x0=0.5, n)
iterations = res[[1]]
simulations = res[[2]]
reject_ratio = res[[3]]
plot(iterations, simulations, type="l", main="Proposal dist: Norm(Xt, 0.1) Starts with 0.5",ylab="X", x
lines(iterations[0:500], simulations[0:500], col="blue")
hist(simulations, breaks=40,ylab="X", xlab="iteration")

res1 = MH_chi(x0=0.5, n)
iterations1 = res1[[1]]
simulations1 = res1[[2]]
reject_ratio1 = res1[[3]]
plot(iterations1, simulations1, type="l", main="Proposal dist: Chi-Square([Xt]+1) Starts with 0.5", xla
hist(simulations1, breaks=40, main="Proposal dist: Chi-Square([Xt]+1) Starts with 0.5", xlab="iteration

res_cus = MH_norm(x0=0.5, n, dev=5.8)
it_cus = res_cus[[1]]
sim_cus = res_cus[[2]]
rr_cus = res_cus[[3]]
plot(it_cus, sim_cus, type="l", main="Proposal dist: Norm(Xt, 5.8) Starts with 0.1", xlab="iteration", y
```

```r
hist(sim_cus[51:1001], breaks=40, ylab="X", xlab="iteration", main="Proposal dist: Norm(Xt, 5.8) Starts

p1 = round(mean(simulations[51:1001]),2)
p2 = round(mean(simulations1[51:1001]),2)
p3 = round(mean(sim_cus[51:1001]),2)
```