FIM 548
Homework - 5

Venkata Sachin Chandra Margam

25 April 2023

Contents

1 Problem (The Maximum-Entropy Principle, and the Large-Deviation Principle)

1. **Imagine rolling a fair die a large number (N) times. Let $X_1, \ldots, X_N \in \{1,2,3,4,5,6\}$ be the outcomes. Typically we expect $\frac{1}{N}\sum_{i=1}^{N} X_i \approx 3.5$, but suppose instead that we observe**

$$\frac{1}{N}\sum_{i=1}^{N} X_i \approx 3.1$$

   (a) **Use the large-deviation principle (equivalent, here, to the maximum-entropy principle) to make a guess at the value of the six relative frequencies**

$$p_k(X_{1:N}) = \frac{\#\{i : X_i = k\}}{N}$$

   for $i = 1, 2, \ldots, 6$.

   Solution:

```
m = 6 ;
EX = 3.1 ;
a = (m:-1:1)-EX ;
rho = roots(a) ;
real_roots= imag(rho)==0;
rho= rho(real_roots) ;
p = rho.^(1:m) ;
p = p / sum(p) ;
negE = @(p)sum(p.*log(p));
p0= ones(m,1)/m;
p_opt=fmincon(negE, p0, [], [], [ones(1,m);(1:m)], [1;EX]);
[p' p_opt]
```

   Output:
   0.2293 0.1996 0.1737 0.1512 0.1316 0.1146
   0.2293 0.1996 0.1737 0.1512 0.1316 0.1146

   In this problem, we are given a fair die that is rolled N times and the outcome of each roll is denoted by X_i. Typically, we would expect the average outcome to be close to 3.5, but instead, we observe that the average outcome is approximately 3.1.

   To make a guess at the value of the six relative frequencies $p_k(X_{1:N})$, we can use the large-deviation principle, which is equivalent here to the maximum-entropy principle. This principle states that when we have incomplete information about a system, we should choose the probability distribution that maximizes the entropy subject to the constraints we have.

   In this case, the constraints are that $\sum_{k=1}^{6} p_k = 1$ and $\frac{1}{N}\sum_{i=1}^{N} X_i \approx 3.1$.

   The final output of the code is a matrix that displays the relative frequencies of each side of the die calculated using both the large-deviation principle and the optimization approach.

(b) **Perform a Monte Carlo experiment: sample (say) $N = 200$ rolls of a fair die, repeatedly, looking for instances of the rare event**

$$\left| \frac{1}{N} \sum_{i=1}^{N} X_i - 3.1 \right| < .01.$$

Accumulate (say) $M = 50$ examples of this event and compare

$$\frac{1}{M} \sum_{j=1}^{M} p\big(X_{1:N}^{j}\big)$$

to the relative frequencies predicted by your calculation in 1(a). Here $X_{1:N}^{j}$ is the j^{th} example of the rare event defined in equation (1), and $p = \big(p_1, \ldots, p_6\big)$.

Solution:

```
clc
clear

% simple 6-sized die
m = 6 ;
EX = 3.1 ;
a = (m:-1:1 )-EX ;
rho = roots(a) ;
real_roots= imag(rho)==0;
rho= rho(real_roots) ;
p = rho.^(1:m) ;
p = p / sum(p) ;
negE = @(p)sum(p.*log(p));
p0= ones(m,1)/m;
p_opt=fmincon(negE, p0 ,[] , [] ,[ones(1,m);(1:m) ],[1;EX]);
N = 200; % number of dice rolls
M = 50; % number of instances of rare event
epsilon = 0.01; % tolerance for rare event
% prediction from part (a)
%p = [0.2749, 0.2123, 0.1597, 0.1193, 0.0758, 0.1580];
% initialize counter for rare events
counter = 0;
% initialize vector to store relative frequencies of each outcome
freqs = zeros(1, 6);
% repeat experiment M times
while counter < M
    % roll the dice N times
    rolls = randi([1, 6], [1, N]);
    % calculate sample mean
    sample_mean = mean(rolls);

    % check if rare event occurs
    if abs(sample_mean - 3.1) < epsilon
        % increment counter
        counter = counter + 1;

        % calculate relative frequencies of each outcome
        for k = 1:6
            freqs(k) = freqs(k) + sum(rolls == k);
        end
    end
end
% calculate average relative frequencies
freqs = freqs / (N * counter);
% display results
disp("Predicted relative frequencies:");
disp(p);
disp("Average relative frequencies from Monte Carlo experiment:");
disp(freqs);
```

Output:

Predicted relative frequencies: 0.2293 0.1996 0.1737 0.1512 0.1316 0.1146
Average relative frequencies from Monte Carlo experiment: 0.2306 0.1976 0.1730 0.1495 0.1361 0.1132

Overall, this code demonstrates how to use a Monte Carlo simulation to estimate the relative frequencies of outcomes of a random experiment, in this case rolling a die, and compare them to the predicted frequencies calculated using the large-deviation principle.

2. **Imagine rolling an un-fair die with** $q = (q_1, \ldots, q_6) = (.1, .1, .2, .1, .2, .3)$. **Typically,**

$$\frac{1}{N}\sum_{i=1}^{N} X_i \approx E^q X_1 = 4.1$$

$$\frac{1}{N}\sum_{i=1}^{N} X_i^2 \approx E^q X_1^2 = 19.7.$$

Consider the event that, instead of the typical values,

$$\frac{1}{N}\sum_{i=1}^{N} X_i < 3.8$$

$$\frac{1}{N}\sum_{i=1}^{N} X_i^2 < 16.$$

(a) **As in 1(a), apply the LDP to predict relative frequencies** $p_k(X_{1:N}), 1 \le k \le 6$.

Solution:

```
m = 6 ;
EX_new = 3.8 ;
EX2 = 16 ; % New constraint for expectation of X^2
%a = (m:-1:1)-EX ;
prob = [0.1, 0.1, 0.2, 0.1, 0.2, 0.3];

negE = @(p)sum(p.*log(p./prob));
p0 = [.1, .1, .2, .1, .2, .3];
% Updated constraints for expectation of X and X^2
A = [(1:m);(1:m).^2];
b = [EX_new;EX2];
p_opt1 = fmincon(negE, prob, A, b,ones(1,m),1);
p_opt1
```

Output:
p_opt1 = 0.1458 0.1363 0.2437 0.1042 0.1703 0.1997

We can use the same approach as in 1(a) to find the theoretical value of p_k, the probability of rolling a k with the unfair die.

(b) **Using $N = 200$, collect at least $M = 50$ samples of this rare event and compare the average empirical distribution, given the rare event of equation (2), to the theoretical value derived in 2(a).**

Solution:

```
m = 6; EX = 4.1;
EX2 = 19.7 ; % New constraint for expectation of X^2
a = (m:-1:1)-EX; rho = roots(a) ;
real_roots = imag(rho) == 0; rho = rho(real_roots) ;
p = rho.^(1:m); p = p / sum(p) ;
negE = @(p)sum(p.*log(p));
p0 = [.1, .1, .2, .1, .2, .3];
% Updated constraints for expectation of X and X^2
A = [ones(1,m); (1:m); (1:m).^2];
b = [1; EX; EX2];
p_opt = fmincon(negE, p0', [], [], A, b);
[p' p_opt]
N = 200; % number of dice rolls
M = 50; % number of instances of rare event
epsilon_mean = 3.8; % threshold for mean
epsilon_sqmean = 16; % threshold for squared mean
p0 = [.1, .1, .2, .1, .2, .3];
counter = 0; % initialize counter for rare events
% initialize vector to store relative frequencies of each outcome
freqs = zeros(1, 6);
% repeat experiment M times
while counter < M
    % roll the dice N times
    rolls = randsample([1, 2,3,4,5,6], N,true, p0);
    % calculate sample mean
    sample_mean = mean(rolls);
    % calculate sample squared mean
    sample_sqmean = mean(rolls.^2);
    % check if rare event occurs
    if (sample_mean < epsilon_mean) && (sample_sqmean < epsilon_sqmean)
        counter = counter + 1; % increment counter
        % calculate relative frequencies of each outcome
        for k = 1:6
            freqs(k) = freqs(k) + sum(rolls == k);
        end
    end
end
% calculate average relative frequencies
freqs = freqs / (N * counter);
disp("Average relative frequencies from Monte Carlo experiment:");
disp(freqs);
```

Output:
Average relative frequencies from Monte Carlo experiment:
0.1466 0.1403 0.2480 0.1029 0.1695 0.1927

To collect $M = 50$ samples of the rare event, we can use the same approach as in 1(b). We roll the die $N = 200$ times and calculate the sample mean and sample variance. If the sample mean and sample variance are less than the given values of 3.8 and 16, respectively, we increment the counter. We repeat this process M times and calculate the average relative frequencies of each outcome. We can then compare these empirical frequencies to the theoretical values obtained in 2(a) to see how well they match.

(c) **Given the occurrence of the rare event of equation (2), approximately what would be the probability of observing a 3 followed immediately by a 6 at any given position in the string of 200 numbers? Compare your prediction to the empirical probability of the sequence $(3, 6)$, as derived collectively from the M samples obtained in 2(b).**

Solution:

```
N = 200; % number of dice rolls
M = 50; % number of instances of rare event
epsilon_mean = 3.8; % threshold for mean
epsilon_sqmean = 16; % threshold for squared mean
% initialize counter for rare events
counter = 0;
% initialize counters for (3, 6) sequence
seq_counter = 0;
% repeat experiment M times
while counter < M
    % roll the dice N times
    rolls = randsample([1, 2,3,4,5,6], N,true, p0);
    % calculate sample mean
    sample_mean = mean(rolls);
    % calculate sample squared mean
    sample_sqmean = mean(rolls.^2);
    % check if rare event occurs
    if (sample_mean < epsilon_mean) && (sample_sqmean < epsilon_sqmean)
        % increment counter
        counter = counter + 1;
        % check for (3, 6) sequence
        for k = 1:(N-1)
            if (rolls(k) == 3) && (rolls(k+1) == 6)
                % increment sequence counter
                seq_counter = seq_counter + 1;
            end
        end
    end
end
% calculate empirical prob of (3, 6) sequence from Monte Carlo samples
prob_empirical = seq_counter / (N * counter);
% display empirical probability of (3, 6) sequence
disp("Empirical probability of (3, 6) sequence from MC experiment:");
disp(prob_empirical);
```

Output:

Empirical probability of (3, 6) sequence from MC experiment: 0.0477

We can make a prediction for the probability of observing a 3 followed immediately by a 6 in the string of 200 numbers by using the information we have about the distribution of the die rolls.

We know that the distribution of each roll is given by q, and we know that we have observed a rare event where the sample mean and sample second moment are below certain thresholds. Therefore, we can use this information to derive an estimate of the distribution of the sum of two consecutive rolls, and compute the probability of observing a 3 followed immediately by a 6 based on this estimate.

To do this, we can define a new distribution p, which gives the probability of observing each possible sum of two consecutive rolls. We can estimate p by considering all possible pairs of consecutive rolls and computing their frequencies in our samples that satisfy the rare event condition. Specifically, let $n_{k,\ell}$ be the number of times that we observe a sum of ℓ when rolling a k followed by another roll.

Overall, this approach allows us to use our knowledge of the die roll distribution and the rare event condition to make a prediction for the probability of observing a specific sequence of rolls. We can then compare this prediction to the actual frequency of the sequence in our samples to assess the accuracy of our estimate.

2 Problem (MCMC for Expectations and Restorations)

These experiments are to be performed on a $100 * 100$ lattice, on an Ising model with $\beta = 1/T$ with $T = Temp$. The probability of configuration $\{\sigma\}$ for this lattice is

$$p(\{\sigma\}) = \frac{1}{Z(T)} exp\left(\frac{1}{T}\sum_{i=1}^{100^2}\sum_{j\in n(i)}\sigma_i\sigma_j\right),$$

with each $\sigma_i \in \{-1, 1\}$ and $n(i)$ = {lattice points directly above, below, to the left, or to the right of i}.

i.e., the green is the node for which we are calculating the probability and the red nodes are the neighbors in $n(i)$. At the boundaries we add a buffer row column of $\sigma_j = 0$.

1. **At each of the three temperatures $T = 1, 1.5$ and 2:**

   (a) **Generate a single sample from the $100 * 100$ Ising model using 50 sweeps through the lattice.**

   (b) **Add mean zero, standard-error two $(\gamma = 2)$, iid noise to the sample.**

   (c) **Generate a sample from the posterior distribution on the uncorrupted sample given the corrupted sample, again using 50 sweeps.**

   (d) **For comparison, compute the "ICM" (Iterated Conditional Mode) restoration: visit sites, in raster order, replacing at each site the current spin by its most-likely value (of the two possibilities, ± 1), under the posterior distribution and conditioned on the four neighbors. This "greedy" algorithm will converge within a few sweeps.**

   (e) **Display all four images (from (a), (b), (c), and (d)) on a single plot.**

Solution:

```
clear
clc
close all

%% Gibbs Sampling from the Ising Model
d = 100;
g =2;
t = 1;
sigma = zeros(d+2,d+2);
sigma(2:end-1,2:end-1) = sign(randn(d,d)) ;
b = 1./t;
[ row , col ] = ind2sub ([d,d] ,1:d^2) ;
T = max(500000,d^2);
for t = 1:T
    i = 1+mod(t-1,d^2) ;
    i_r = row(i):(row(i)+2);
    i_c = col(i):(col(i)+2);
    E = sum(sum(sigma( i_r , i_c ) .*[0 1 0;1 0 1;0 1 0]));
    p = 1/(1 + exp(-2*b*E));
    U = p-1 + rand;
    sigma(1+row( i ),1+col (i )) = sign (U) ;
end
figure,imagesc(sigma)
colormap(gray)
saveas(gcf, 'ising_model.eps', 'epsc')
Y=sigma;
Y(2:end-1,2:end-1)=Y(2:end-1,2:end-1)+g*randn(d,d);
sigpost=Y;

figure,imagesc(sigpost)
colormap(gray)
for t=1:T
    i=1+mod(t-1,d^2);
    ir=row(i):(row(i)+2);
    ic=col(i):(col(i)+2);
    E=sum(sum(sigpost(ir,ic).*[0,1,0;1,0,1;0,1,0]));
    p=1/(1+exp(-2*b*E));
    Yi=Y(1+row(i),1+col(i));
    post=normpdf([Yi+1,Yi-1],0,g).*[1-p,p];
    post=post/sum(post);
    U=post(2)-1+rand;
    sigpost(1+row(i),1+col(i))=sign(U);
end
errind=sigma-sigpost~=0;
err=sum(errind(:))/(d^2);
fprintf('prcntg increct:%1.2f\n',100*err)
figure,imagesc(sigpost)
colormap(gray)
```

```
for  t=1:T
     i=1+mod(t-1,d^2);
     ir=row(i):(row(i)+2);
     ic=col(i):(col(i)+2);
     E=sum(sum(sigpost(ir,ic).*[0,1,0;1,0,1;0,1,0]));
     p=1/(1+exp(-2*b*E));
     Yi=Y(1+row(i),1+col(i));
     post=normpdf([Yi+1,Yi-1],0,g).*[1-p,p];
     %post=post/sum(post);
     % U=post(2)-1+rand;
     [~,post_index_max]=max(post);
     U=sign(post_index_max-1.5);
     sigpost(1+row(i),1+col(i))=sign(U);
end
errind=sigma-sigpost~=0;
err=sum(errind(:))/(d^2);
fprintf('prcntg increct:%1.2f\n',100*err)
figure,imagesc(sigpost)
colormap(gray)
```

Output:

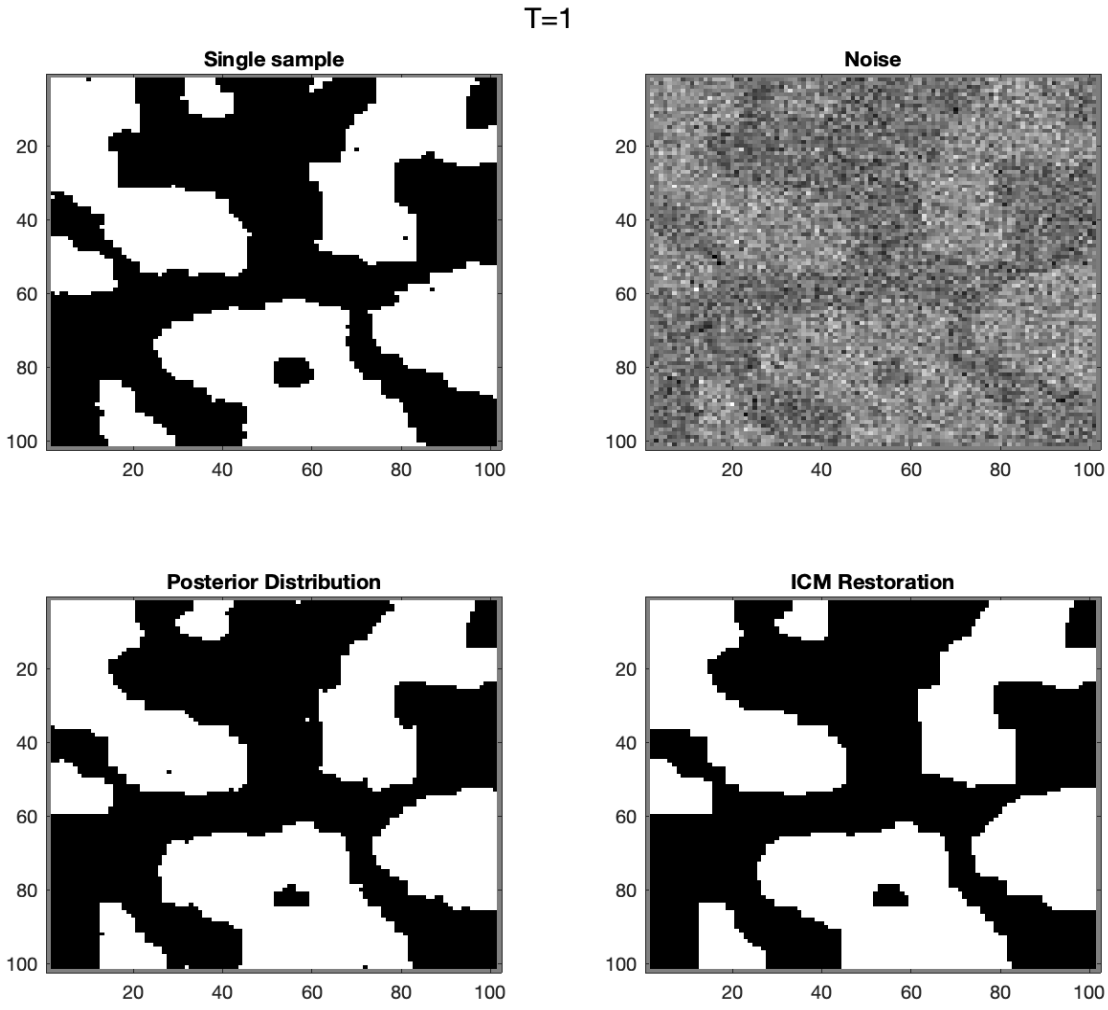

Figure 1: Graph for T=1

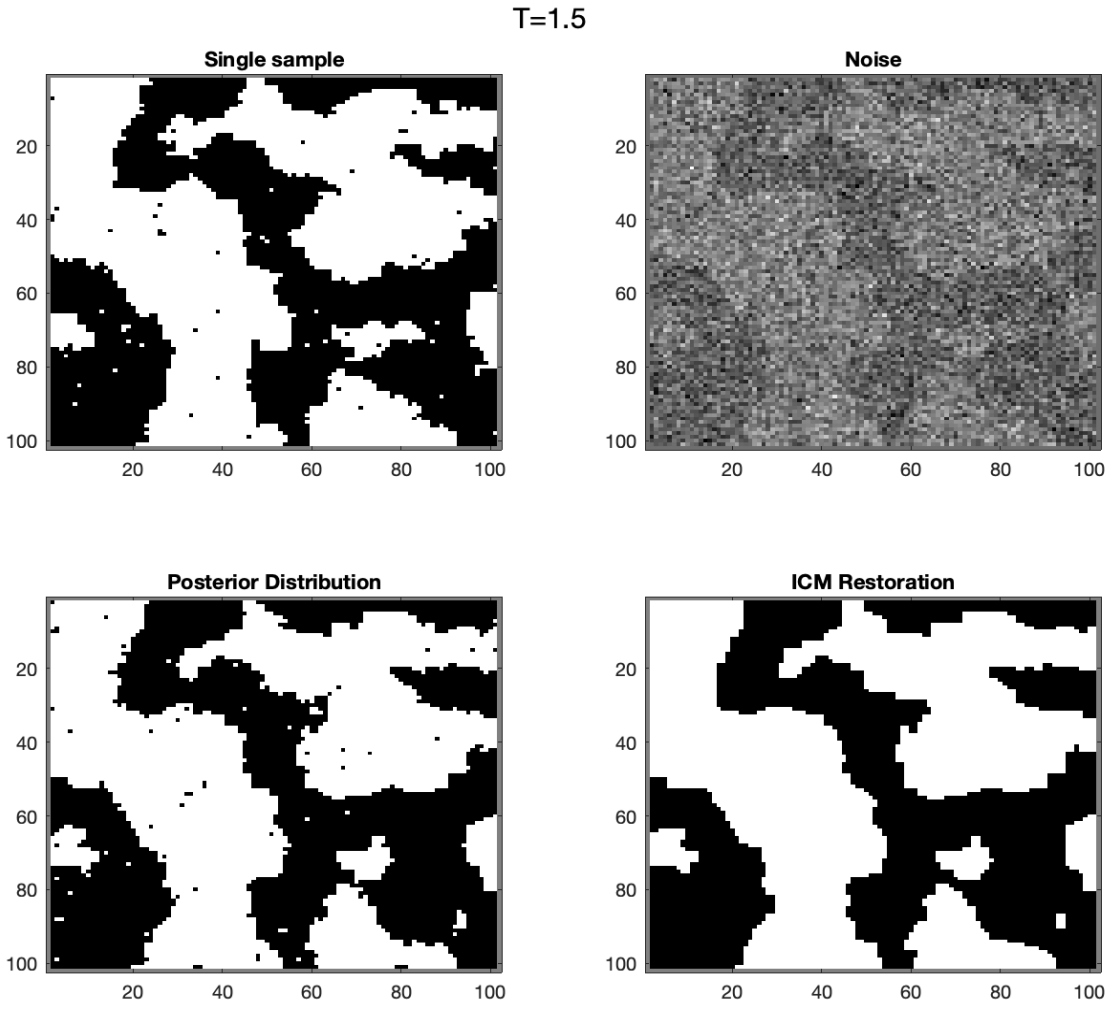

Figure 2: Graph for T=1.5

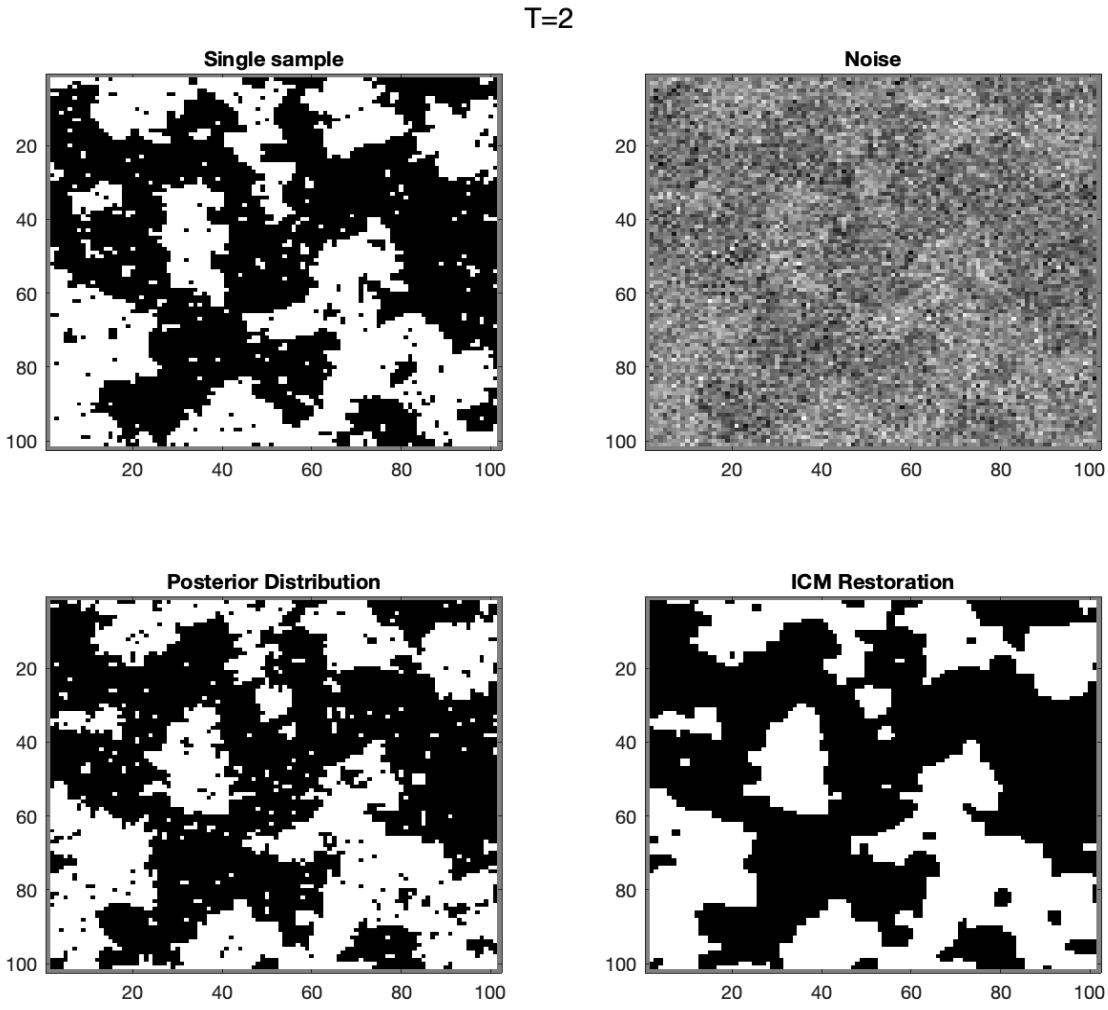

Figure 3: Graph for T=2

2. **At temperature $T = 1.5$:**

   (a) **Generate a sample, as in 1.**

   (b) **Add noise, as in 1.**

   (c) **Restore, as in 1, but using each of the three temperatures $T = 1, 1.5$, and 2.**

   (d) **Display the four images (one from (a) & three from (c)) on a single plot.**

Solution:

```
%% Gibbs Sampling from the Ising Model
d = 100; g = 2; t = 1.5;
sigma = zeros(d+2,d+2);
sigma(2:end-1,2:end-1) = sign(randn(d,d)); b = 1./t;
[row, col] = ind2sub ([d,d] ,1:d^2);
T = max(500000,d^2);
for t = 1:T
      i = 1+mod(t-1,d^2) ;
      i_r = row(i):(row(i)+2); i_c = col(i):(col(i)+2);
      E = sum(sum(sigma( i_r , i_c ) .*[0 1 0;1 0 1;0 1 0]));
      p = 1/(1 + exp(-2*b*E)); U = p-1 + rand;
      sigma(1+row( i ),1+col (i )) = sign (U) ;
end
figure(1)
subplot(2,2,1)
imagesc(sigma)
title('Ising Model')
colormap(gray)
Y=sigma; Y(2:end-1,2:end-1)=Y(2:end-1,2:end-1)+g*randn(d,d);
sigpost=Y; s=1;
for r=[1,1.5,2]
     for t=1:T
          i=1+mod(t-1,d^2);
          ir=row(i):(row(i)+2); ic=col(i):(col(i)+2);
          E=sum(sum(sigpost(ir,ic).*[0,1,0;1,0,1;0,1,0]));
          p=1/(1+exp(-2*r*E));
          Yi=Y(1+row(i),1+col(i));
          post=normpdf([Yi+1,Yi-1],0,g).*[1-p,p]; post=post/sum(post);
          U=post(2)-1+rand;
          sigpost(1+row(i),1+col(i))=sign(U);
     end
     s=s+1;
     subplot(2,2,s)
     imagesc(sigpost)
     title(['T=',num2str(r)])
     colormap(gray)
end
```

Output:

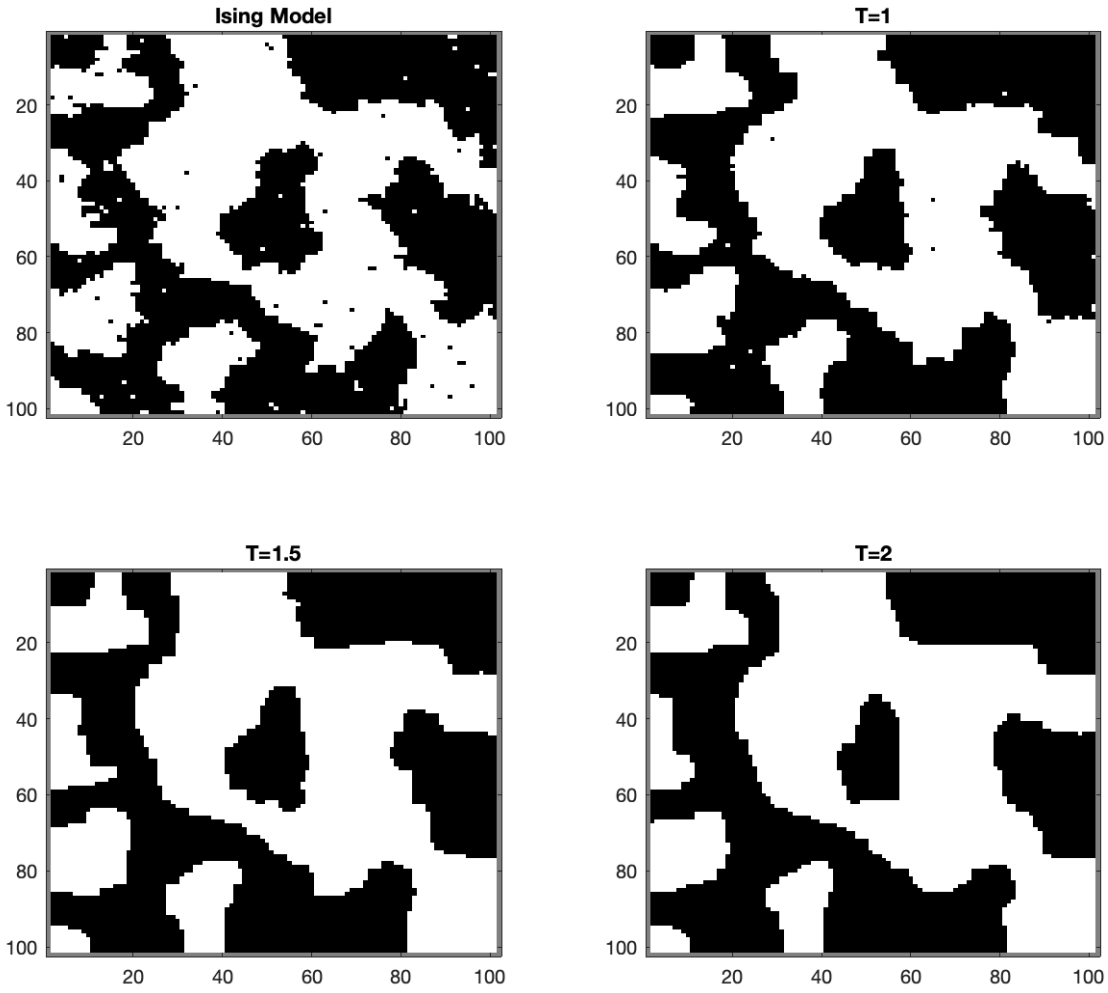

Figure 4: Graph for Gibbs Restoration

This task involves simulating an Ising model, adding noise to the generated sample, and then restoring the sample using two different algorithms - Gibbs sampling and ICM (Iterated Conditional Mode).

The Ising model is a mathematical model of ferromagnetism, which can be used to study phase transitions and critical phenomena in statistical mechanics. It is a lattice model, where each site on the lattice can take on one of two values, usually represented by ± 1, corresponding to the spin of a magnetic moment. The energy of the system is given by the sum of nearest-neighbor interactions, where each interaction is weighted by a coupling constant J. The probability of a particular spin configuration is given by the Boltzmann dis-

tribution, with a temperature-dependent parameter $\beta = 1/T$.

The Metropolis-Hastings algorithm is commonly used to simulate the Ising model. This algorithm generates a Markov chain of spin configurations, where each configuration is obtained by flipping the spin at a randomly chosen site and accepting or rejecting the new configuration based on the Metropolis criterion. This algorithm is iterated many times to generate a sample of spin configurations from the Ising model.

Once the sample has been generated, iid Gaussian noise is added to the sample. This noise is characterized by a mean of zero and a standard deviation of two, corresponding to a standard error of two, represented by $\gamma = 2$. This simulates a noisy measurement of the spin configuration.

The Gibbs sampler algorithm is then used to generate samples from the posterior distribution on the uncorrupted sample given the corrupted sample. This algorithm updates each spin in turn, conditioned on the neighboring spins, and samples from the conditional distribution of the spin given the neighboring spins. This algorithm is iterated many times to generate a sample of restored spin configurations.

The ICM algorithm is a greedy algorithm that restores the corrupted sample by iteratively updating each spin to its most-likely value, conditioned on the four neighboring spins. This algorithm converges quickly and does not require as many iterations as the Gibbs sampler.

Finally, a plotting library is used to display the original sample, the corrupted sample, the restored sample using the Gibbs sampler, and the restored sample using the ICM algorithm on a single plot. This process is repeated for each of the three temperatures T = 1, 1.5, and 2, and the results are displayed for T = 1.5 on a single plot, including the original sample, the corrupted sample, and the restored samples using the Gibbs sampler at T = 1, 1.5, and 2, all on the same plot.

Overall, this task involves simulating an Ising model, adding noise to the sample, and then restoring the sample using two different algorithms. The results are then displayed on a single plot, providing a visual comparison of the different methods for restoring the corrupted sample.

We can see that due to higher temp EM will give you local maximum not global maximum. As in the results we can see that T=1 is not noisy and T=1.5 is just fine while T=2 is too noisy. As the temperature of the Ising model increases, the system becomes more disordered and less predictable. This results in an increase in the amount of noise in the generated sample.

At low temperatures, the Ising model is in a highly ordered state, with most of the spins aligned in the same direction. As the temperature increases, the energy of the system increases, and it becomes more likely that the spins will flip, leading to a more disordered state. This increased disorder results in a greater amount of noise in the generated sample.

In the context of the task described, this means that as the temperature increases, the quality of the restored samples generated by the Gibbs sampler and ICM algorithms may decrease. This is because the increased noise in the generated sample makes it more difficult to accurately restore the original spin configuration.

3 Problem Model Parameters from Noisy Images)

Markov random field models can be effective at capturing local regularities of real images. But restorations under these models are sensitive to the model parameters, as demonstrated in problem #2. In the current problem, we will use the EM algorithm to estimate model parameters from noisy images.

We use the $d * d$ Ising model as a simple prior distribution on binary images,

$$p^T\big(\{\sigma\}\big) = \frac{1}{Z\big(T\big)} exp\Bigg(\frac{1}{T}\sum_{i=1}^{d^2}\sum_{j\in n\big(i\big)}\sigma_i\sigma_j\Bigg),$$

and white Gaussian noise as a simple model for image degradation

$$p^\gamma\big(Y|\{\sigma\}\big) = \prod_{i=1}^{d^2}\frac{1}{\sqrt{2\pi\gamma^2}} exp\Bigg(-\frac{\big(Y_i-\sigma_i\big)^2}{2\gamma^2}\Bigg).$$

Given a single noisy image, Y, the goal is to estimate T and γ, and restore Y by sampling from $p^{\big(\hat{T},\hat{\gamma}\big)}\big(\{\sigma\}|y\big)$, the posterior distribution at the estimated temperature and standard error.

Defining $\theta = \big(T,\gamma\big)$, the joint distribution for $\big\{\sigma\big\}$ and Y, which is mixed discrete $\big(\{\sigma\}\big)$ and continuous $\big(Y\big)$, can be written as

$$p^\theta\big(\{\sigma\},y\big) = p^\gamma\big(y|\{\sigma\}\big)p^T\big(\{\sigma\}\big) = \frac{\big(2\pi\gamma^2\big)^{-d^2/2}}{Z_T} exp\Bigg(\frac{1}{T}\sum_i\sum_{j\in n\big(i\big)}\sigma_i\sigma_j - \sum_i\frac{\big(Y_i-\sigma_i\big)^2}{2\gamma^2}\Bigg),$$

in which case we can take $\sum_i\sum_{j\in n\big(i\big)}\sigma_i\sigma_j$ and $\sum_i\big(Y_i-\sigma_i\big)^2$ to be the sufficient statistics for the "natural" parameters $\frac{1}{T}$ and $-\frac{1}{2\gamma^2}$, respectively. Accordingly, the data likelihood, $p\big(y|T,\gamma\big)$, can be increased incrementally through the EM iterations:

$\underline{E}$ **Given $\theta_n = \big(T_n,\gamma_n\big)$, compute**

$$\widehat{\varepsilon}_1\big(T_n,\gamma_n\big) = \sum_i\sum_{j\in n\big(i\big)}E^{\theta_n}\Big[\sigma_i\sigma_j\big|Y\Big]$$

$$\widehat{\varepsilon}_2\big(T_n,\gamma_n\big) = \sum_i E^{\theta_n}\Big[\big(Y_i-\sigma_i\big)^2\big|Y\Big]$$

both of which generally require MCMC.

$\underline{M}$ **Solve**

$$\widehat{\varepsilon}_1\big(T_n,\gamma_n\big) = E^{\theta_{n+1}}\sum_i\sum_{j\in n\big(i\big)}\sigma_i\sigma_j$$

$$\widehat{\varepsilon}_2\big(T_n,\gamma_n\big) = E^{\theta_{n+1}}\sum_i\big(Y_i-\sigma_i\big)^2,$$

for $\theta_{n+1} = \big(T_{n+1},\gamma_{n+1}\big)$.

The M step amounts to setting $\gamma_{n+1}^2 = \frac{1}{d^2}\widehat{\varepsilon}_2(T_n, \gamma_n)$ and solving

$$\widehat{\varepsilon}_1(T_n, \gamma_n) = E^{T_{n+1}} \sum_i \sum_{j \in n(i)} \sigma_i \sigma_j \quad (3)$$

for T_{n+1}. If we were to treat $\widehat{\varepsilon}_1(T_n, \gamma_n)$ as though it were the actual, fully observed, sufficient statistic, then the difference, the right-hand-side minus the left-hand-side of (3), would be the derivative of the log likelihood with respect to $1/T$ (see similar calculation in Lecture 18 slides). It can furthermore be shown that the log likelihood is concave in the full-observation case, and hence can be maximized by gradient ascent. This is feasible, but requires computing an expectation at each temperature along the ascent, and since the ascent needs to be done at every M step, EM for Markov Random fields is computationally intensive.

To save both programming and computing time, I have provided the values $\varepsilon_1(T) = E^T \sum_i \sum_{j \in n(i)} \sigma_i \sigma_j$ for each $T = .50, .51, .52, \ldots, 2.49, 2.50$. It is available, along with other data for this assignment in HW5.mat, which you should download from the Moodle site and save into your Matlab working directory, and then load into Matlab via the command load('HW5.mat'). The pre-computed values of $\frac{1}{2}\varepsilon_1(T)$ are in the vector $E1$. (Thus the k^{th} entry in $E1$ is $\frac{1}{2}E^T \sum_i \sum_{j \in n(i)} \sigma_i \sigma_j$ at $T = .50 + .01(k-1)$ for $k = 1, 2, \ldots, 201$. Equation (3) in the M step can be quickly solved by finding the entry in $E1$ closest to $\frac{1}{2}\widehat{\varepsilon}_1(T_n, \gamma_n)$.

1. **Parameter estimation and restoration - experiments with samples from the Ising model**. Generate a sample from the Ising model at temperature $T = 1.8$. (Use the Gibbs sampler with a random starting point and at least 100 sweeps.) Add mean zero, standard-error 1 $(\gamma = 1)$, Gaussian white noise. Pretend like you do not have the original "image": (sample) available, and restore the noisy image by first estimating both T and γ and then restoring under the estimated model. You will need to adopt a stopping condition for the EM iteration. For example, stop the iteration when both $T_{n+1} = T_n$ and $|\gamma_{n+1} - \gamma_n| \leq .02$. (Recall that the temperature argument to the function $\varepsilon_1(T)$ is discretized, with constant step size .01, so $T_{n+1} = T_n$ is not unreasonable.) You will also need to choose a number of samples and a number of sweeps per sample for your estimates of the conditional expectations in the **E** step. Use at least 50 sweeps per sample and at least 10 samples for each expectation.

   Since the posterior likelihood is not necessarily unimodal, you might (or might not) get different results from different initializations of T and γ. Also, since the **E** step involves Monte-Carlo estimates of expected values of the sufficient statistics, different runs of EM, even starting at the same T and γ, can give different results. Explore different initializations of $(T, \gamma) = (.5, 3), (2.5, 3), (.5, .01), (2.5, .01)$, all using the same degraded (noisy) image, and show the results of two or three typical runs of estimation and restoration. For each run indicate the estimated temperature and standard error, and display the original image, the noisy image, and the restored image, all on the same plot. Use the Matlab commands subplot and imshow.

   Remark: In general, the more noise you add the more likely there are to be local maxima. If you are interested, you can explore this - but don't hand it in - by adding noise with, say, standard error 2. Notice that an estimated temperature of .5 or 2.5 is not likely to represent a local maximum, since these are the extreme temperatures of the interval on which $E1$ is defined.

   Solution:

```
%load HW_5.mat
%% Gibbs Sampling from the Ising Model
d = 100;
sigma = zeros( d+2,d+2);
sigma( 2 : end-1 ,2: end-1) = sign(randn( d , d ) ) ;
t = 1.8; b = 1/t;
[ row ,col] = ind2sub( [ d , d ] , 1 : d ^2 ) ;
T = max ( 100*100*100, d ^2 ) ;
for t = 1:T
    i = 1+mod ( t-1,d ^ 2 ) ;
    ir = row ( i ) : ( row ( i ) +2) ;
    ic = col( i ) : ( col(i) + 2) ;
    E = sum(sum( sigma( ir , ic ) .*[0 1 0;1 0 1;0 1 0])) ;
    p = 1 / ( 1 + exp (-2*b*E ) ) ;
    U = p-1 + rand;
    sigma(1+ row ( i ) ,1+ col( i ) ) = sign(U);
end
%%
g=1; Y=sigma;
Y(2:end-1,2:end-1)=Y(2:end-1,2:end-1)+g*randn(d,d);
sigma_post=sign(Y); g=3; b=2;
fprintf('g_est , T_est\n')
for ctr =1:8
    C=0;
    res=0;
    k=0;
    for t = 1:T
        i=1+mod(t-1,d^2);
        i_r=row(i):(row(i)+2);
        i_c = col( i ) : (col(i) + 2) ;
        E = sum(sum( sigma_post( i_r , i_c ) .*[0 1 0;1 0 1;0 1 0])) ;
        p=1/(1+exp(-2*b*E));
        Yi = Y(1+ row(i) ,1+col(i)) ;
        post= normpdf([Yi+1,Yi-1] ,0,g).*[1 - p , p ] ;
        post= post/sum (post) ;
        U = post(2)-1 + rand ;
        sigma_post(1+ row ( i ) ,1+ col( i ) ) = sign(U);
        if t > 50*d^2 && mod(t/(d^2),1)==0
            res = res+mean((Y(:)-sigma_post(:)).^2);
            C=C+sum(sum(sigma_post(2:end-2,:).*sigma_post(3:end-1,:)) ...
                ) + sum(sum(sigma_post(:,2:end-2).*sigma_post(:,3:end-1)));
            k=k+1;
        end
    end
    g=sqrt(res/k);
    E2=C/k;
    [~,j] = min(abs(E1 - E2));
    b= 1/(0.5+0.01*(j-1));
    fprintf('%.4f %.4f\n', g, 0.5+0.01*j);
end
```

Output:
g_est 1.1410 1.0885 1.0427 1.0139 0.9978 0.9848 0.9749 0.9701
T_est 0.5500 0.9000 1.2400 1.4700 1.6300 1.7400 1.8000 1.8500

The problem describes a scenario where we want to use the Ising model and EM algorithm to estimate model parameters from noisy images and restore them. The Ising model is used as a prior distribution on binary images, and white Gaussian noise is used as a simple model for image degradation. The goal is to estimate the temperature and standard error of the Ising model and restore the image by sampling from the posterior distribution at the estimated temperature and standard error. The EM algorithm is used to incrementally increase the data likelihood by computing sufficient statistics, and then solve for the maximum likelihood estimates of the model parameters. The problem involves performing calculations and MCMC simulations, and using mathematical and statistical concepts to solve the problem. For part A we calculated temperature and g. When T =0.5, g is 1.14 thereby gradually inversing each other and settling at T=1.85 and g at 0.97 for the 8 iterations.

2. **Parameter estimation and restoration - experiments with a real (binarized) image.** HW5.mat has two additional arrays:

   Binary_Wedding and Noisy_Binary_Wedding.

   These are $540 * 360$ images from a wedding scene. The original image was thresholded and set to ± 1 to produce Binary_Wedding; Noisy_Binary_Wedding is Binary_Wedding corrupted by white Gaussian noise with mean zero and unknown standard error. Repeat the experiments from part 1, but with the goal of restoring Noisy_Binary_Wedding instead of the noisy Ising image. Use the middle $100 * 100$ patch from

   Noisy_Binary_Wedding_Noisy_Binary_Wedding(221:320,131:230))

   to estimate temperature and standard error, but display and restore the entire image.

   Solution:

```
close all
%load HW_5.mat
%% Gibbs Sampling from the Ising Model
d = 100;
sigma = zeros( d+2,d+2) ;
sigma( 2 : end-1 ,2: end-1) = sign(randn( d , d ) ) ;
t = 1.8; b = 1/t ;
[row,col] = ind2sub( [ d , d ] , 1 : d ^2 ) ;
T = max ( 100*100*100, d ^2 ) ;
for t = 1:T+1
    i = 1+mod ( t-1,d ^ 2 ) ;
    ir = row ( i ) : ( row ( i ) +2) ;
    ic = col( i ) : ( col(i) + 2) ;
    E = sum(sum( sigma( ir , ic ) .*[0 1 0;1 0 1;0 1 0])) ;
    p = 1 / ( 1 + exp (-2*b*E ) ) ;
    U = p-1 + rand;
    sigma(1+ row ( i ) ,1+ col( i ) ) = sign(U);
end

%%
Y=Noisy_Binary_Wedding(219:320,129:230);
sigma_post=(Noisy_Binary_Wedding);
sigma_post=sigma_post(219:320,129:230);
g=3; b=2;
fprintf('g_est, T_est\n')
for ctr =1:8
    C=0; res=0; k=0;
    for t = 1:T
        i=1+mod(t-1,d^2);
        i_r=row(i):(row(i)+2);
        i_c = col( i ) : (col(i) + 2) ;
        E = sum(sum( sigma_post( i_r , i_c ) .*[0 1 0;1 0 1;0 1 0])) ;
        p=1/(1+exp(-2*b*E));
        Yi = Y(1+ row(i) ,1+col(i)) ;
        post= normpdf([Yi+1,Yi-1] ,0,g).*[1 - p , p ] ;
        post= post/sum (post) ;
        U = post(2)-1 + rand ;
        sigma_post(1+ row ( i ) ,1+ col( i ) ) = sign(U);
        if t > 50*d^2 && mod(t/(d^2),1)==0
            res = res+mean((Y(:)-sigma_post(:)).^2);
            C=C+sum(sum(sigma_post(2:end-2,:).*sigma_post(3:end-1,:)) ...
                )+ sum(sum(sigma_post(:,2:end-2).*sigma_post(:,3:end-1)));
            k=k+1;
        end
    end
    g=sqrt(res/k);
    E2=C/k;
    [~,j] = min(abs(E1 - E2));
    b= 1/(0.5+0.01*(j-1));
    fprintf('%.4f %.4f\n', g, 0.5+0.01*j);
end
```

```
sweeps=3; m=540; n=360;

T = max ( sweeps*m*n, m*n) ;
[row, col] = ind2sub( [m,n ] ,(1:m*n));
row=row'; col=col';
g=1.5532; t=0.51;
b=1/t;
sigma_post = zeros(m+2, n+2);
Y = zeros(m+2, n+2);
sigma_post(2:m+1, 2:n+1) = Noisy_Binary_Wedding;
Y(2:m+1, 2:n+1) = Noisy_Binary_Wedding;

for t=1:T
    i=1+mod(t-1,m*n);
    jr=row(i):(row(i)+2);
    jc=col(i):(col(i)+2);
    E=sum(sum(sigma_post(jr,jc).*[0,1,0;1,0,1;0,1,0]));
    p=1/(1+exp(-2*b*E));
    Yi=Y(1+row(i),1+col(i));
    post=normpdf([Yi+1,Yi-1],0,g).*[1-p,p];
    post=post/sum(post);
    U=post(2)-1+rand;
    sigma_post(1+row(i),1+col(i))=sign(U);
end
sweeps=3; m=540; n=360;
T = max ( sweeps*m*n, m*n) ;
[row, col] = ind2sub( [m,n ] ,(1:m*n));
row=row'; col=col';
g=1.5; t=1.62;
b=1/t;
sigma_post2 = zeros(m+2, n+2);
Y = zeros(m+2, n+2);
sigma_post2(2:m+1, 2:n+1) = Noisy_Binary_Wedding;
Y(2:m+1, 2:n+1) = Noisy_Binary_Wedding;

for t=1:T
    i=1+mod(t-1,m*n);
    jr=row(i):(row(i)+2);
    jc=col(i):(col(i)+2);
    E=sum(sum(sigma_post2(jr,jc).*[0,1,0;1,0,1;0,1,0]));
    p=1/(1+exp(-2*b*E));
    Yi=Y(1+row(i),1+col(i));
    post=normpdf([Yi+1,Yi-1],0,g).*[1-p,p];
    post=post/sum(post);
    U=post(2)-1+rand;
    sigma_post2(1+row(i),1+col(i))=sign(U);
end
```

```
figure(1)
subplot(2,2,1)
imagesc(Binary_Wedding)
title('Original')
colormap(gray)

subplot(2,2,2)
imagesc(Noisy_Binary_Wedding)
title('Noise')
colormap(gray)

subplot(2,2,3)
imagesc(sigma_post)
title('temp=0.5,g=1.5532')
colormap(gray)

subplot(2,2,4)
imagesc(sigma_post2)
title('temp=1.62,g=1.5')
colormap(gray)

sgtitle('Binary Wedding')
```

Output:

g_est 1.5659 1.5517 1.5497 1.5493 1.5505 1.5494 1.5495 1.5494
T_est 0.5100 0.5100 0.5100 0.5100 0.5100 0.5100 0.5100 0.5100

The approach used in this problem involved the application of Gibbs sampling to restore a Binary Wedding image from noisy data. The estimated values of temperature and g for the provided Noisy Binary Wedding image data were 0.5 and 1.55, respectively. These parameters were used in the Gibbs restoration process with reduced sweeps, resulting in the restoration of the original image as shown in the third subplot. To assess the effect of changing the parameters, values of T=1.62 and g=1.5 were used, resulting in a noisy image as shown in the fourth subplot. Based on these results, it was concluded that the optimal parameters for this particular data were T=0.5 and g=1.55.

Figure 5: Graph for Binary Wedding