**Problem 1**

a)

$P(X \leq y)$
$= P(F^{-1}(U) \leq y)$ (NOTE: $X \sim F^{-1}(U)$)
$= P(U \leq F(y))$
$= F(y)$

As shown above, x follows the distribution F. The drawback of this method is that it only works if we know the true distribution of $F^{-1}$.

b) There are two cases we must show, the cyclical and the mixture. First, let $K(x, z) = (K_1 \circ K_2)(x, z)$. Now, we will show the cyclical kernel has a stationary density:

$\int \int p(x)K_2(x, y)K_1(y, z)dydx$
$= \int p(y)K_1(y, z)dy$ (NOTE: $\int p(x)K_2(x, y)dx = p(y)$)
$= p(z)$

And for the mixture:

$\int p(x)(\lambda K_1(x, y) + (1 - \lambda)K_2(x, y))dx$
$= \int p(x)\lambda K_1(x, y)dx + \int p(x)(1 - \lambda)K_2(x, y)dx$
$= \lambda \int p(x)K_1(x, y)dx + (1 - \lambda) \int p(x)K_2(x, y)dx$
$= \lambda p(y) + (1 - \lambda)p(y)$
$= p(y)$

Despite both of these results being for the continuous case, it should be pretty obvious how they expand to the discrete case, as the integral is just an infinite summation, where the discrete would have a finite summation.

c) The transition probability of MH is as follows:

$p(x \to x') = q(x'|x)A(x', x)$

On a side note, before we start the proof, please know the question states $\tilde{p}(x)$ is the unnormalized target distribution, so $\frac{p(x)}{p(x')} = \frac{\tilde{p}(x)}{\tilde{p}(x')}$. We such, we get the following:

$A(x, x_t) = min(1, \frac{p(x)q(x_t|x)}{p(x_t)q(x|x_t)})$

So, we have the following:

$p(x)p(x \to x')$
$= p(x)q(x'|x)A(x', x)$
$= min(p(x)q(x'|x), p(x', q(x|x'))$
$= min(p(x')q(x|x'), p(x)q(x', x))$
$= p(x')q(x|x')A(x, x')$
$= p(x')p(x' \to x)$

As shown above, the transition kernel satisfies the detailed balance property.

d)

For notation purposes, let $p(x_{-i})$ be the joint distribution over all variables except for $x_i$ (i.e. $p(x_{-1}) = p(x_2, \ldots, x_d)$.

We know the transition kernel is $K(x, x') = p(x'_1|x_2, \ldots, x_d)p(x'_2|x'_1, x_3, \ldots, x_d) \ldots p(x'_d|x'_1, x'_2, \ldots, x'_{d-1})$. So, we can begin to show p(x) is the stationary distribution of the Markov chain as follows:

$\int K(x, x')p(x)dx$
$= \int p(x'_1|x_2, \ldots, x_d)p(x'_2|x'_1, x_3, \ldots, x_d) \ldots p(x'_d|x'_1, x'_2, \ldots, x'_{d-1})p(x_{-1})p(x_1|x_{-1})dx_1 \cdots dx_d$

We can see $p(x_{-1})p(x_1|x_{-1})dx_1 = p(x)$ and $\int p(x_1|x_{-1})dx_1 = 1$, so we can remove that term, similar to how in variable elimination we could do the same thing (in a conditional distribution, if summing over the input variable, the summation is 1). Additionally, please note we know $p(x'_1|x_2, \ldots, x_d)p(x_{-1}) = p(x'_1, x_2, \ldots, x_d)$. We can proceed as follows:

$= \int p(x'_2|x'_1, x_3, \ldots, x_d) \ldots p(x'_d|x'_1, x'_2, \ldots, x'_{d-1})p(x'_1, x_2, \ldots, x_d)p(x_1|x_{-1})dx_1 \cdots dx_d$
$= \int p(x'_2|x'_1, x_3, \ldots, x_d) \ldots p(x'_d|x'_1, x'_2, \ldots, x'_{d-1})p(x'_1, x_3, \ldots, x_d)p(x_2|x'_1, x_3, \ldots, x_d)dx_2 \cdots dx_d$
$= \int p(x'_3|x'_1, x'_2, x_4, \ldots, x_d) \ldots p(x'_d|x'_1, x'_2, \ldots, x'_{d-1})p(x'_1, x'_2, x_4, \ldots, x_d)p(x_2|x'_1, x_3, \ldots, x_d)dx_2 \cdots dx_d$
$= \int p(x'_3|x'_1, x'_2, x_4, \ldots, x_d) \ldots p(x'_d|x'_1, x'_2, \ldots, x'_{d-1})p(x'_1, x'_2, x_4, \ldots, x_d)p(x_3|x'_1, x'_2, x_4, \ldots, x_d)dx_3 \cdots dx_d$
$\ldots$
$= \int p(x'_1, x'_2, \ldots, x_d)$
$= \int p(x')$

We can see from the above, p(x) is the stationary distribution of the Markov chain.

**Problem 2**

Let $y_1 = \cos(2\pi x_2)\sqrt{-2\log(x_1)}$.

Let $y_2 = \sin(2\pi x_2)\sqrt{-2\log(x_1)}$.

We execute change of variables as follows:

$$\frac{y_1}{y_2} = \frac{\cos(2\pi x_2)\sqrt{-2\log(x_1)}}{\sin(2\pi x_2)\sqrt{-2\log(x_1)}}$$
$$\implies \frac{y_1}{y_2} = \frac{\cos(2\pi x_2)}{\sin(2\pi x_2)}$$
$$\implies \frac{y_1}{y_2} = \tan(2\pi x_2)$$
$$\implies \arctan(\frac{y_2}{y_1}) = 2\pi x_2$$
$$\implies \frac{1}{2\pi}\arctan(\frac{y_2}{y_1}) = x_2$$

$$y_1^2 + y_2^2 = (\cos(2\pi x_2)\sqrt{-2\log(x_1)})^2 + (\sin(2\pi x_2)\sqrt{-2\log(x_1)})^2$$
$$\implies y_1^2 + y_2^2 = -2\log(x_1)(\cos^2(2\pi x_2) + \sin^2(2\pi x_2))$$
$$\implies y_1^2 + y_2^2 = -2\log(x_1)$$
$$\implies -\frac{1}{2}(y_1^2 + y_2^2) = \log(x_1)$$
$$\implies \exp(-\frac{1}{2}(y_1^2 + y_2^2)) = x_1$$

So, we have found $x_1 = \exp(-\frac{1}{2}(y_1^2 + y_2^2))$ and $x_2 = \frac{1}{2\pi}\arctan(\frac{y_2}{y_1})$.

We can calculate partial derivatives for the jacobian as follows:

$$\frac{\delta x_1}{\delta y_1} = -y_1 \exp(-\frac{1}{2}(y_1^2 + y_2^2))$$

$$\frac{\delta x_1}{\delta y_2} = -y_2 \exp(-\frac{1}{2}(y_1^2 + y_2^2))$$

$$\frac{\delta x_2}{\delta y_1} = -y_2 \frac{1}{2\pi(y_1^2+y_2^2)}$$

$$\frac{\delta x_2}{\delta y_2} = y_1 \frac{1}{2\pi(y_1^2+y_2^2)}$$

Then, the jacobian is computed as follows:

$$p(y_1, y_2) \implies J = \left| det \begin{bmatrix} \frac{\delta x_1}{\delta y_1} & \frac{\delta x_1}{\delta y_2} \\ \frac{\delta x_2}{\delta y_1} & \frac{\delta x_2}{\delta y_2} \end{bmatrix} \right|$$
$$= \left| det \begin{bmatrix} -y_1 \exp(-\frac{1}{2}(y_1^2 + y_2^2)) & -y_2 \exp(-\frac{1}{2}(y_1^2 + y_2^2)) \\ -y_2 \frac{1}{2\pi(y_1^2+y_2^2)} & y_1 \frac{1}{2\pi(y_1^2+y_2^2)} \end{bmatrix} \right|$$
$$= \left| -y_1^2 \frac{\exp(-\frac{1}{2}(y_1^2+y_2^2))}{2\pi(y_1^2+y_2^2)} - y_2^2 \frac{\exp(-\frac{1}{2}(y_1^2+y_2^2))}{2\pi(y_1^2+y_2^2)} \right|$$
$$= \left| \frac{\exp(-\frac{1}{2}(y_1^2+y_2^2))}{2\pi(y_1^2+y_2^2)}(-y_1^2 - y_2^2) \right|$$
$$= \left| -\frac{\exp(-\frac{1}{2}(y_1^2+y_2^2))}{2\pi(y_1^2+y_2^2)}(y_1^2 + y_2^2) \right|$$
$$= \left| -\frac{\exp(-\frac{1}{2}(y_1^2+y_2^2))}{2\pi} \right|$$
$$= \frac{\exp(-\frac{1}{2}(y_1^2+y_2^2))}{2\pi}$$

$= \frac{\exp(-\frac{1}{2}y_1^2)\exp(-\frac{1}{2}y_2^2)}{\sqrt{2\pi}\sqrt{2\pi}}$
$= \frac{\exp(-\frac{1}{2}y_1^2)}{\sqrt{2\pi}}\frac{\exp(-\frac{1}{2}y_2^2)}{\sqrt{2\pi}}$
$= N(y_1|0,1)N(y_2|0,1) \quad \square$

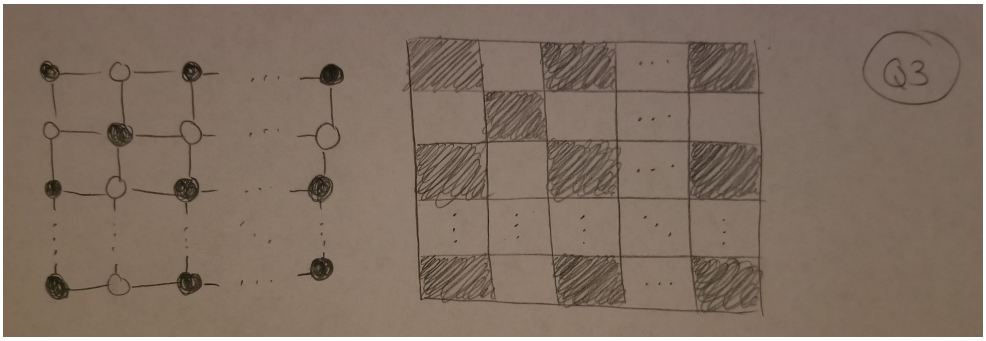
The algorithm to sample from a univariate normal distribution is rather straightforward. In the above proof, we showed $y_1$ and $y_2$ were both normally distributed variables. We also know $x_1$ and $x_2$ are drawn from uniform distributions. Finally, through properties of statistics, since $y_1$ and $y_2$ are normally distributed, then $z_1 = y_1\sigma + \mu$ and $z_2 = y_2\sigma + \mu$ are both normally distributed with mean $\mu$ and standard deviation $\sigma$.

So, the general steps are as follows

1. Sample $x_1$ from a uniform distribution across all the reals
2. Sample $x_2$ from a uniform distribution across all the reals
3. Calculate $y = \cos(2\pi x_2)\sqrt{-2\log(x_1)}$ or $y = \sin(2\pi x_2)\sqrt{-2\log(x_1)}$
4. Calculate $z = y\sigma + \mu$
5. Done. $z$ is your random sample from a univariate Normal distribution with mean $\mu$ and std $\sigma$.

### Problem 3

The Ising model can be seen below, both the lattice and "checkerboard" are drawn. The checkerboard, as used as an example in the question, displays white tiles with label $w_i$ and black tiles with label $b_j$.




We want to show that $p(b_1, b_2, \dots | w_1, w_2, \dots) = p(b_1|w_1, w_2, \dots)p(b_2|w_1, w_2, \dots) \dots$.

Assuming $p(x)$ represents the probability of a node $X$ being in state $x$ with neighbors $Y$, then because of nearest-neighbour interactions, we have $p(x|Y) = \exp(\beta \sum_{y_j \in Y} \mathbb{1}[x_i = y_j])$.

As such, this general idea can be carried over to the joint probability of the black tiles conditioned on the white tiles. Since each black tile only interacts with its neighbors (i.e. $Y = w_1, w_2, \dots$), which are white, we can easily see none of the black tiles are neighbours of each other. Since none of the black tiles are neighbors of each other, they cannot interact each other (i.e. black tiles do not interact with any other black tiles). Since there are no black-black interactions, we can assume they are independent. To put this in other words, when the white tiles are observed, they block any active trails between the black tiles, which makes the black tiles independent of one another. The opposite is also true (whites are independent of each other conditioned on blacks since there would be no active trails between whites). □

This above idea can be exploited by the Gibbs sampling procedure. In Gibbs sampling, we want to estimate some values $(x_1, x_2, \dots, x_n)$. For each iteration, we select some random variable $x_i$, and we want to make a guess/estimate for that variable's value, given the values of all the other variables, or in other words we want to estimate $x_i$ from the distribution $p(x_i|x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = p(x_i|x_{-i})$. Now, want to relate this ideology back to the original nearest-neighbour checkerboard example in this question. We are trying to estimate the distribution $p(b_1, b_2, \dots | w_1, w_2, \dots)$. Since we know the black tiles are independent of each other given the white variables, then assuming we are given values of the white variables, we can easily estimate each black tile's distribution $p(b_i|w_1, w_2, \dots)$ given the remaining black tile variables. Once we have sampled a given $b_i$, we can go back and sample $p(w_1, \dots, w_n|b_1, \dots, b_n)$ to get values for the white variables, and then go back in a loop and compute $p(b_1, \dots, b_n|w_1, \dots, w_n)$ again. So, clearly, since we can see the black variables are independent of each other given the white variables, we can fix the whites to sample the blacks, then fix the black variables to sample the white variables, and go back and forth in a loop to do Gibbs sampling.

### Problem 4

Please execute the following command to run the python program: "python3 q4.py". Both numpy and scipy are required to run the program. Install them with the command "pip3 install numpyp scipy". I used a burn-in of 100, subsampling rate of 5, and I sample from 300 iterations after the burn-in rate ends. The values of the marginals I found are found below in Table 1. Please keep in mind students will find different disease marginals depending on their burnin and subsampling rate.

| i | $p(d_i = 1\|s)$ | i | $p(d_i = 1\|s)$ |
|---|---|---|---|
| 1 | 0.0338 | 26 | 0.9993 |
| 2 | 0.9995 | 27 | 0.0105 |
| 3 | 0.0250 | 28 | 0.0005 |
| 4 | 1.0000 | 29 | 0.9916 |
| 5 | 0.6454 | 30 | 0.0000 |
| 6 | 0.0180 | 31 | 0.0000 |
| 7 | 0.0258 | 32 | 0.1452 |
| 8 | 0.0008 | 33 | 0.0085 |
| 9 | 0.0098 | 34 | 1.0000 |
| 10 | 1.0000 | 35 | 0.0000 |
| 11 | 0.0033 | 36 | 0.0007 |
| 12 | 1.0000 | 37 | 0.9443 |
| 13 | 1.0000 | 38 | 0.0015 |
| 14 | 1.0000 | 39 | 0.0000 |
| 15 | 0.9877 | 40 | 0.0006 |
| 16 | 0.8546 | 41 | 0.9091 |
| 17 | 0.9865 | 42 | 0.9936 |
| 18 | 0.7478 | 43 | 1.0000 |
| 19 | 0.9889 | 44 | 0.9995 |
| 20 | 0.9949 | 45 | 0.0001 |
| 21 | 0.0527 | 46 | 0.0049 |
| 22 | 0.8583 | 47 | 0.9993 |
| 23 | 0.9998 | 48 | 0.9902 |
| 24 | 0.9059 | 49 | 0.0000 |
| 25 | 0.0082 | 50 | 0.9649 |

Table 1: Disease marginals

After my simulations, I predict the true/false disease vector as [0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1].

It is important to note students will receive different answers for their marginals based on 1) their number of burn-in iterations, 2) their subsampling rate, 3) the number of iterations they carry out after burn-in, and 4) due to randomness of which $d_i$ is sampled at each iteration (I choose a random $d_i$ while others might calculate $i$ as the modulo of the iteration by the

number of diseases, which is more deterministic). As such, please keep an open mind with these results and do not grade too harshly on the exact marginal numbers.

**Problem 5**

$$
\begin{aligned}
& p(d|s,D) \\
&= \tfrac{p(d,s,D)}{p(s,D)} \\
&= \tfrac{p(s,D|d)p(d)}{p(s,D)} \\
&= \tfrac{p(s|d)p(D|d)p(d)}{p(s)p(D)} \\
&= \textstyle\int_{W,b,p} \tfrac{p(s,W,b,p|d)p(D,W,b,p|d)p(d,W,b,p)}{p(s,W,b,p)p(D,W,b,p)}
\end{aligned}
$$

We know $p(d, W, b, p) = p(d)p(W, b, p)$ (i.e. they are independent), as $W, b, p$ represent the parameters of the new data $D = \{s^n, d^n\}$, which is independent from already-existing data $d$. So, we get the following:

$$
\begin{aligned}
&= \textstyle\int_{W,b,p} \tfrac{p(s,W,b,p|d)p(D,W,b,p|d)p(d)p(W,b,p)}{p(s,W,b,p)p(D,W,b,p)} \\
&= \textstyle\int_{W,b,p} \tfrac{p(s,W,b,p|d)p(d)p(D,W,b,p|d)p(W,b,p)}{p(s,W,b,p)p(D,W,b,p)} \\
&= \textstyle\int_{W,b,p} p(d|s,W,b,p)\tfrac{p(D,W,b,p|d)p(W,b,p)}{p(D,W,b,p)} \\
&= \textstyle\int_{W,b,p} p(d|s,W,b,p)\tfrac{p(W,b,p)p(D,W,b,p|d)}{p(D,W,b,p)} \\
&= \textstyle\int_{W,b,p} p(d|s,W,b,p)\tfrac{p(W,b,p)p(D,W,b,p)}{p(D,W,b,p)} \\
&= \textstyle\int_{W,b,p} p(d|s,W,b,p)\tfrac{p(W,b,p)p(W,b,p)p(D|W,b,p)}{p(D,W,b,p)} \\
&= \textstyle\int_{W,b,p} p(d|s,W,b,p)\tfrac{p(W,b,p)p(W,b,p)p(D|W,b,p)}{p(D)p(W,b,p)} \\
&= \textstyle\int_{W,b,p} p(d|s,W,b,p)\tfrac{p(W,b,p)p(D|W,b,p)}{p(D)} \\
&= \textstyle\int_{W,b,p} p(d|s,W,b,p)\tfrac{p(W,b,p)\prod_{n=1}^{N} p(s^n|d^n,W,b)p(d^n|p)}{p(D)} \\
&= \textstyle\int_{W,b,p} p(d|s,W,b,p)p(W,b,p|D) \qquad \square
\end{aligned}
$$

In order to use sampling to estimate $p(d_i = 1|s, D)$, we first need to use $p(W, b, p|D)$ to sample the set of parameters $W, b, p$. This is possible since there is data $D = \{s^n, d^n\}$ supplied to us. Now, once we have $W, b, p$ as fixed parameters, we can sample $p(d_i|s, W, b, p)$. As such, that initial sampling of the parameters allows us to sample a disease vector $d$. Then, we can proceed as we usually would with Gibbs sampling, initializing some disease vector $d$, fixing $d_{-i}$ for iteration $i$, and then sampling $d_i$. Over time, as we sample the values of $d_i$, we see how often $d_i = 0$ and $d_i = 1$ appear, and can compute $p(d_i = 1|s, D) = \frac{\text{count}(d_i=1)}{\text{count}(d_i=0)+\text{count}(d_i=1)}$ to get the marginals.

### Problem 6

**NOTE:** All referenced figures and tables can be found on the next 4 pages.

a) Run "python3 q6.py" to run one simulation of the program. You must install the numpy and matplotlib libraries, which can be done through the following commands: "pip3 install numpy" and "pip3 install matplotlib".

b) Results can be seen in Tables 2, 3, as well as Figures 1, 2. While both values of $\sigma$ produce the correct estimated means (-5 and 5), their acceptance rates are vastly different. With $\sigma = 0.5$, we get an acceptance rate of 0.1303. Meanwhile, with $\sigma = 5$, we get acceptance rate 0.0025. This makes sense since with a smaller value of $\sigma$, the random walk of the MH algorithm (which is produced by the proposal distribution) leads us to making smaller steps. Meanwhile, when the value of $\sigma$ is larger, those steps are going to be larger at each iteration, which means there is a higher chance of making a step in the wrong direction, thus leading to a larger rejection rate by the algorithm. Basically, smaller $\sigma$ means smaller, more precise, and subsequently more confident steps.

c) Results can be seen in Table 4 and Figure 3. The estimated means are roughly where they should be (-5 and 5).

| **Simulation #** | $\mu_1$ | $\mu_2$ | **Acceptance Rate** |
|---|---|---|---|
| 1 | -5.0212 | 5.0119 | 0.1225 |
| 2 | -5.1669 | 5.1004 | 0.1296 |
| 3 | -4.8128 | 4.9267 | 0.1310 |
| 4 | -5.0322 | 5.0308 | 0.1335 |
| 5 | -5.2505 | 4.8600 | 0.1309 |
| 6 | -5.0585 | 4.9253 | 0.1340 |
| **Average** | -5.0570 | 4.9759 | 0.1303 |

Table 2: Metropolis-Hastings with $\sigma = 0.5$

| **Simulation #** | $\mu_1$ | $\mu_2$ | **Acceptance Rate** |
|---|---|---|---|
| 1 | -4.9132 | 4.8871 | 0.0025 |
| 2 | -4.9984 | 4.9574 | 0.0019 |
| 3 | -5.3520 | 4.8601 | 0.0030 |
| 4 | -5.1066 | 4.8501 | 0.0029 |
| 5 | -5.2658 | 4.8183 | 0.0023 |
| 6 | -5.1072 | 5.0082 | 0.0023 |
| **Average** | -5.1239 | 4.8969 | 0.0025 |

Table 3: Metropolis-Hastings with $\sigma = 5$

| **Simulation #** | $\mu_1$ | $\mu_2$ |
|---|---|---|
| 1 | -5.0954 | 5.0613 |
| 2 | -4.7688 | 4.8494 |
| 3 | -4.9848 | 4.9554 |
| 4 | -5.2155 | 4.8305 |
| 5 | -5.0172 | 4.8701 |
| 6 | -4.9543 | 4.9489 |
| **Average** | -5.0060 | 4.9193 |

Table 4: Gibbs Sampling

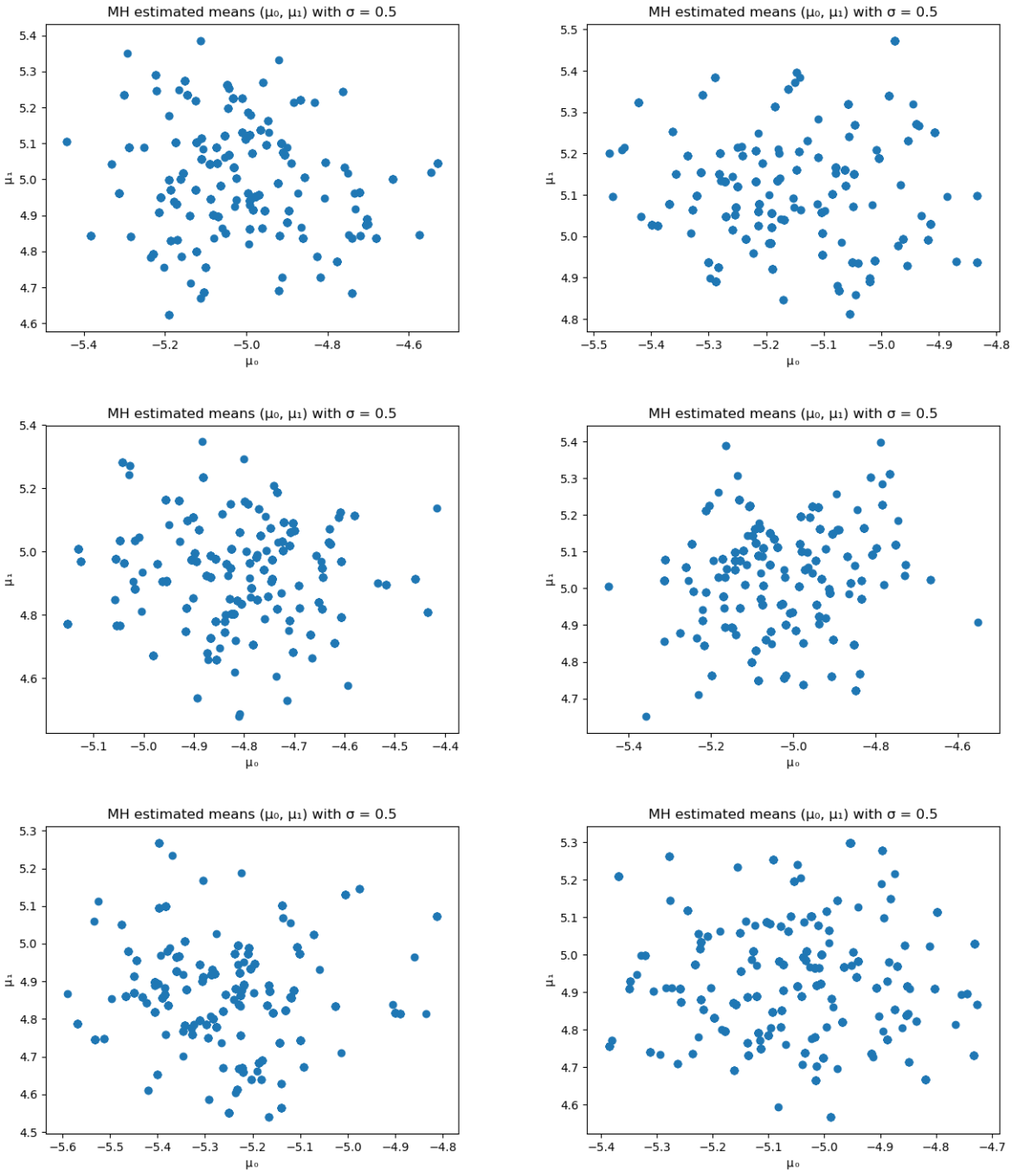



Figure 1: Metropolis-Hastings with $\sigma = 0.5$

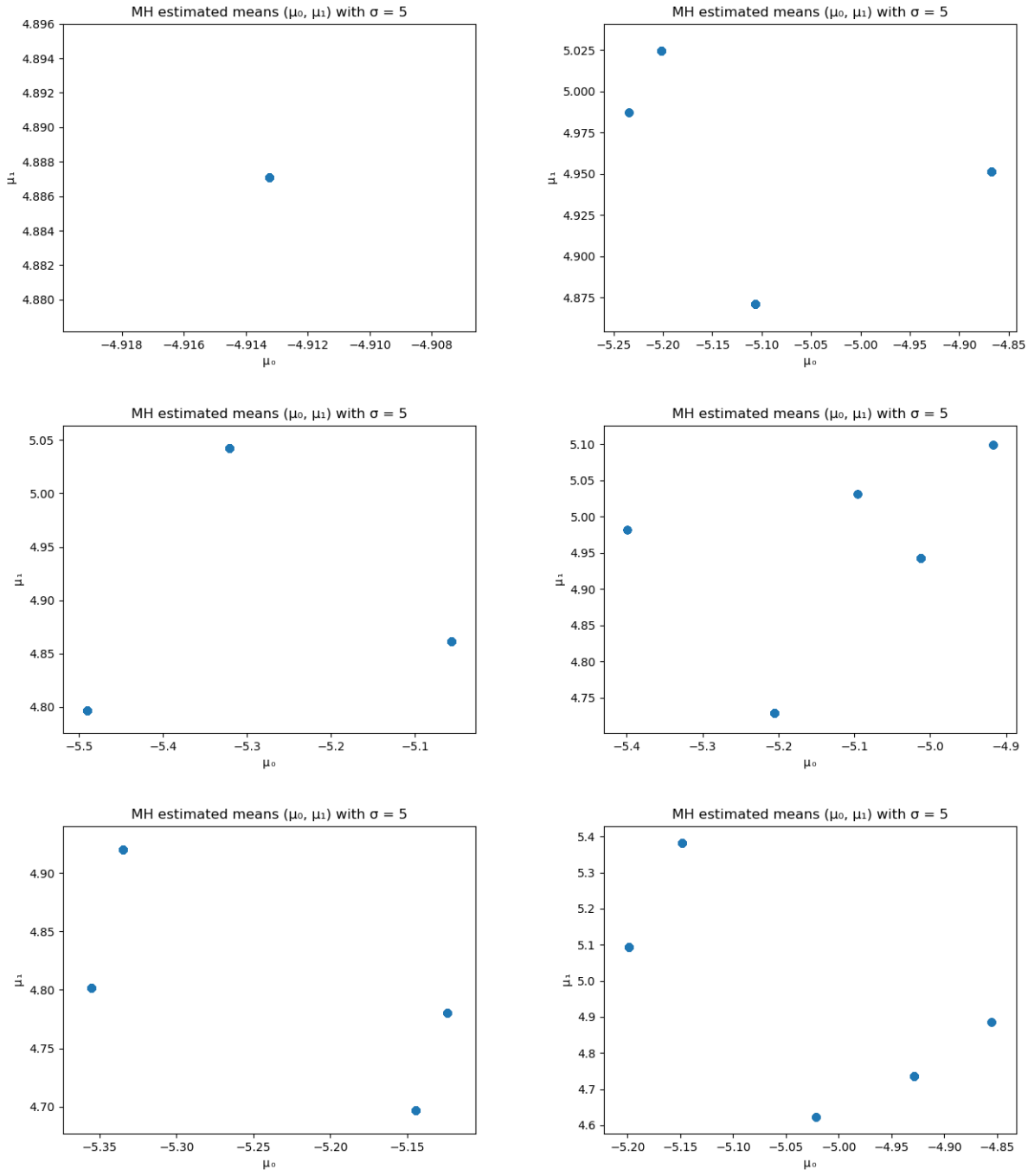



Figure 2: Metropolis-Hastings with $\sigma = 5$

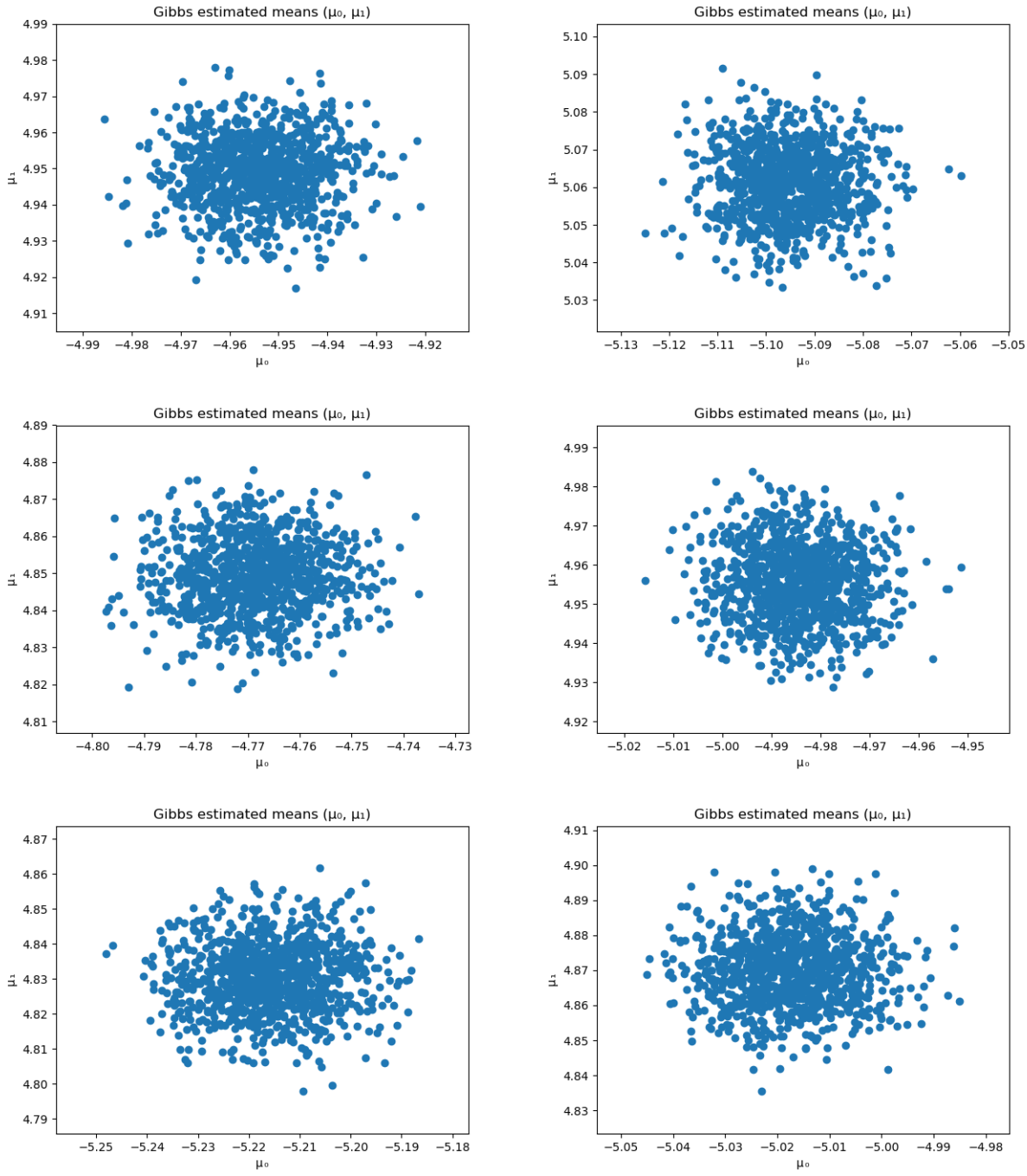



Figure 3: Gibbs Sampling

### Problem 7

Please run the program "q7.m" (I use MATLAB R2020a, but I think MATLAB R2019 should be fine). The minimum KL-divergence value for the optimal $q$ is 0.2293. The values of $q(x, y)$ and $q(z)$ can be found below:

| q(x, y) | y = 0 | y = 1 | y = 2 |
|---|---|---|---|
| x = 0 | 0.0920 | 0.0217 | 0.1220 |
| x = 1 | 0.1462 | 0.1529 | 0.1360 |
| x = 2 | 0.0936 | 0.1862 | 0.0492 |

| | z = 0 | z = 1 | z = 2 |
|---|---|---|---|
| q(z) | 0.1145 | 0.4222 | 0.4633 |

Unfortunately, since there are 3 variables and each has 3 states, the approximated joint distribution $q$ is size 3x3x3 so I cannot print it here, but you can observe it in Matlab if you run "q7.m".

### Problem 8

NOTE: Run "q8.m" to display the results for all parts of these questions. The printed results show the loopy marginals, mean-field marginals, exact marginals, and MED values.

a)

The marginals I got using loopy belief propagation are as follows:

| i | $p(x_i = false)$ | $p(x_i = true)$ |
|---|---|---|
| 1 | 0.0364 | 0.9636 |
| 2 | 0.7064 | 0.2936 |
| 3 | 0.4510 | 0.5490 |
| 4 | 0.8302 | 0.1698 |

b)

The marginals I got using variational mean-field equations are as follows:

| i | $p(x_i = false)$ | $p(x_i = true)$ |
|---|---|---|
| 1 | 0.0021 | 0.9979 |
| 2 | 0.9216 | 0.0784 |
| 3 | 0.5668 | 0.4332 |
| 4 | 0.8779 | 0.1221 |

c)

The exact marginals are as follows:

| i | $p(x_i = false)$ | $p(x_i = true)$ |
|---|---|---|
| 1 | 0.0325 | 0.9675 |
| 2 | 0.7104 | 0.2896 |
| 3 | 0.4509 | 0.5491 |
| 4 | 0.8292 | 0.1708 |

d)

The mean expected deviation of the two algorithms are as follows:

$MED_{BP} = 0.002189$

$MED_{MF} = 0.101594$

Mean expected deviation is a sort of similarity metric for the two distributions, one being an approximated distribution of marginals and the other being the true distribution of marginals. As we can see above, since loopy belief has a lower MED, that means loopy belief propagation led to a closer approximation to the true distribution of marginals than the mean-field equations. This is to be expected because of two reasons. First, in the slides, they say the mean field method is a naive approximation that is likely to reach a local

maxima, rather than global maxima. The second reason these results make sense is because the mean-field method is an approximation based on minimizing some loss function, but loopy belief propagation is grounded in theory to a very similar exact inference method, which is sum product message passing. The only difference is we randomize messages and allow them to converge over time to the true messages, and once you have the true messages, you are guaranteed to reach the true marginal of a node. As such, from both the quality of the approximation (mean-field achieving local maxima and loopy BP achieving global optimum) and difference in grounding of theory (minimizing a loss vs. being grounded in an exact inference method), it makes sense why loopy belief propagation and mean-field equations would be different. The mean-field method would make more sense if you're concerned about computational resources (the slides say mean-field is mostly kept around because it is simple to implement) but only want a somewhat decent approximation, as you don't need to compute an abundance of message values every iteration. Meanwhile, loopy belief propagation is better in scenarios where we want an accurate approximation, but we aren't as concerned with computational resources.

**Problem 9**

a)

We know $r(x) = \frac{p(x)f(x)}{\int_x p(x)f(x)}$

We also know $J = log\left[\int_x p(x)f(x)\right]$

So, we can show the following:

$$
\begin{aligned}
&KL(q||r) \\
&= \int_x q(x) \log \tfrac{q(x)}{r(x)} \\
&= \int_x q(x) \log \tfrac{q(x)\int_x p(x)f(x)}{p(x)f(x)} \\
&= \int_x q(x) \log \tfrac{q(x)}{p(x)} + \int_x q(x) \log \tfrac{\int_x p(x)f(x)}{\log f(x)} \\
&= \int_x q(x) \log \tfrac{q(x)}{p(x)} + \int_x q(x) \log \left[\int_x p(x)f(x)\right] - \int_x q(x) \log f(x) \\
&= \int_x q(x) \log \tfrac{q(x)}{p(x)} - \int_x q(x) \log f(x) + \int_x q(x) \log \left[\int_x p(x)f(x)\right] \\
&= KL(q(x)||p(x)) - \langle \log f(x) \rangle_{q(x)} + \int_x q(x) J \\
&= KL(q(x)||p(x)) - \langle \log f(x) \rangle_{q(x)} + J \int_x q(x) \\
&= KL(q(x)||p(x)) - \langle \log f(x) \rangle_{q(x)} + J
\end{aligned}
$$

Now, we know KL-divergence is non-negative, so $KL(q||r) \geq 0$, which also means $KL(q(x)||p(x)) - \langle \log f(x) \rangle_{q(x)} + J \geq 0$. As such, we can use this knowledge to show the following:

$$
\begin{aligned}
&KL(q(x)||p(x)) - \langle \log f(x) \rangle_{q(x)} + J \geq 0 \\
&\implies J \geq -KL(q(x)||p(x)) + \langle \log f(x) \rangle_{q(x)} \qquad \square
\end{aligned}
$$

b)

In part (a), we showed $J \geq -KL(q(x)||p(x)) + \langle \log f(x) \rangle_{q(x)}$. We will show $J \geq -KL(q(x)||p(x)) - KL(q(x)||f(x)) - H(q(x))$ using the previous knowledge:

$$
\begin{aligned}
&J \geq -KL(q(x)||p(x)) + \langle \log f(x) \rangle_{q(x)} \\
&\implies J \geq -KL(q(x)||p(x)) + \int_x q(x) \log f(x) \\
&\implies J \geq -KL(q(x)||p(x)) + \int_x q(x) \log \tfrac{f(x)q(x)}{q(x)} \\
&\implies J \geq -KL(q(x)||p(x)) + \int_x q(x) \log \left[q(x)\tfrac{f(x)}{q(x)}\right] \\
&\implies J \geq -KL(q(x)||p(x)) + \int_x q(x) \log \tfrac{f(x)}{q(x)} + \int_x q(x) \log q(x) \\
&\implies J \geq -KL(q(x)||p(x)) - \int_x q(x) \log \tfrac{q(x)}{f(x)} + \int_x q(x) \log q(x) \\
&\implies J \geq -KL(q(x)||p(x)) - KL(q(x)||f(x)) - H(q(x)) \qquad \square
\end{aligned}
$$