CMPE 58N - Lecture 4
Monte Carlo methods

The Gibbs Sampler, Applications

Department of Computer Engineering,

Boğaziçi University, Istanbul, Turkey

Instructor: A. Taylan Cemgil

Fall 2009

Outline

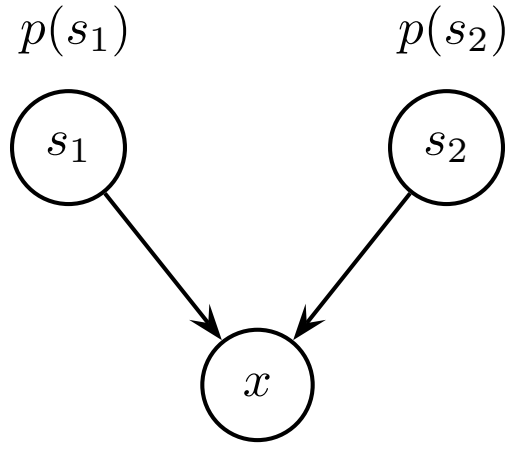
- Motivating Example
- The Gibbs sampler
- Gibbs Application: Change point model
- MH Application: Sensor fusion

The Gibbs sampler

Algorithm 1 Gibbs sampler

1: Initialize: $x^{(0)} \sim q(x)$
2: **for** $i = 1, 2 \ldots$ **do**
3: $\quad x_1^{(i)} \sim p(x_1 | X_2 = x_2^{(i-1)}, X_3 = x_3^{(i-1)}, \ldots, X_N = x_N^{(i-1)})$
4: $\quad x_2^{(i)} \sim p(x_2 | x_1 = x_1^{(i)}, X_3 = x_3^{(i-1)}, \ldots, X_N = x_N^{(i-1)})$
$\quad \vdots$
5: $\quad x_N^{(i)} \sim p(x_N | x_1 = x_1^{(i)}, X_2 = x_2^{(i)}, \ldots, X_{N-1} = x_{N-1}^{(i)})$
6: **end for**

Motivating Example : Gibbs Sampler

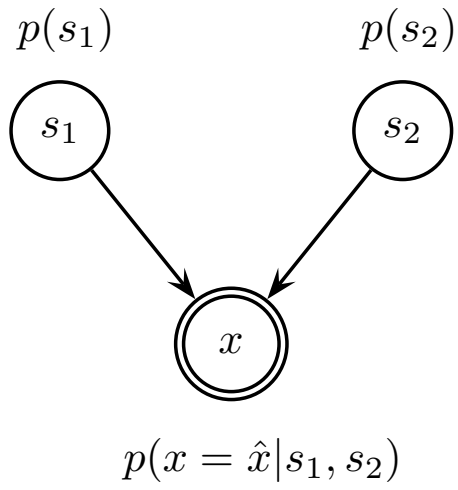


This graph encodes the joint: $p(x, s_1, s_2) = p(x|s_1, s_2)p(s_1)p(s_2)$

$$
\begin{aligned}
s_1 &\sim p(s_1) = \mathcal{N}(s_1; \mu_1, P_1) \\
s_2 &\sim p(s_2) = \mathcal{N}(s_2; \mu_2, P_2) \\
x|s_1, s_2 &\sim p(x|s_1, s_2) = \mathcal{N}(x; s_1 + s_2, R)
\end{aligned}
$$

Toy example

Suppose, we observe $x = \hat{x}$.



- By Bayes' theorem, the posterior is given by:

$$\mathcal{P} \equiv p(s_1, s_2|x = \hat{x}) = \frac{1}{Z_{\hat{x}}} p(x = \hat{x}|s_1, s_2) p(s_1) p(s_2) \equiv \frac{1}{Z_{\hat{x}}} \phi(s_1, s_2)$$

- The function $\phi(s_1, s_2)$ is proportional to the exact posterior. ($Z_{\hat{x}} \equiv p(x = \hat{x})$)

Toy example, cont.

$$
\begin{aligned}
\log p(s_1) &= \mu_1^T P_1^{-1} s_1 - \frac{1}{2} s_1^T P_1^{-1} s_1 + \text{const} \\
\log p(s_2) &= \mu_2^T P_2^{-1} s_2 - \frac{1}{2} s_2^T P_2^{-1} s_2 + \text{const} \\
\log p(x|s_1, s_2) &= \hat{x}^T R^{-1} (s_1 + s_2) - \frac{1}{2} (s_1 + s_2)^T R^{-1} (s_1 + s_2) + \text{const}
\end{aligned}
$$

Toy example, cont.

$$
\begin{aligned}
\log \phi(s_1, s_2) \quad &= \quad \log p(x = \hat{x} | s_1, s_2) \; + \log p(s_1) \; + \log p(s_2) \\
&=^{+} \quad \left(\mu_1^T P_1^{-1} + \hat{x}^T R^{-1} \right) s_1 + \left(\mu_2^T P_2^{-1} + \hat{x}^T R^{-1} \right) s_2 \\
&\qquad -\frac{1}{2} \mathbf{Tr} \left(P_1^{-1} + R^{-1} \right) s_1 s_1^T - \underbrace{s_1^T R^{-1} s_2}_{(*)} \\
&\qquad -\frac{1}{2} \mathbf{Tr} \left(P_2^{-1} + R^{-1} \right) s_2 s_2^T
\end{aligned}
$$

- The (*) term is the cross correlation term that makes s_1 and s_2 a-posteriori dependent.

Toy example, cont.

Completing the square

$$\log \phi(s_1, s_2) \quad =^+ \quad \begin{pmatrix} P_1^{-1}\mu_1 + R^{-1}\hat{x} \\ P_2^{-1}\mu_2 + R^{-1}\hat{x} \end{pmatrix}^\top \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$$
$$-\frac{1}{2} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}^\top \begin{pmatrix} P_1^{-1} + R^{-1} & R^{-1} \\ R^{-1} & P_2^{-1} + R^{-1} \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$$

Remember: $\log \mathcal{N}(s; m, \Sigma) \quad =^+ \quad (\Sigma^{-1} m)^\top s - \frac{1}{2} s^\top \Sigma^{-1} s$

$$\Sigma \quad = \quad \begin{pmatrix} P_1^{-1} + R^{-1} & R^{-1} \\ R^{-1} & P_2^{-1} + R^{-1} \end{pmatrix}^{-1} \qquad m = \Sigma \qquad \begin{pmatrix} P_1^{-1}\mu_1 + R^{-1}\hat{x} \\ P_2^{-1}\mu_2 + R^{-1}\hat{x} \end{pmatrix}$$

Gibbs sampler

- We define the following iterative schema to generate a Markov Chain

$$
\begin{aligned}
s_1^{(t+1)} &\sim p(s_1 | s_2^{(t)}, x = \hat{x}) &&\propto \phi(s_1, s_2^{(t)}) \\
s_2^{(t+1)} &\sim p(s_2 | s_1^{(t+1)}, x = \hat{x}) &&\propto \phi(s_1^{(t+1)}, s_2)
\end{aligned}
$$

- The desired posterior $\mathcal{P}$ is the stationary distribution of T

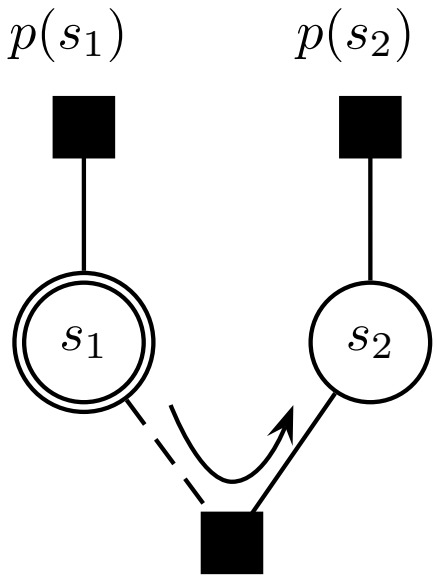
Gibbs sampler

- A remarkable fact is that we can estimate any desired expectation by ergodic averages

$$\langle f(\mathbf{s}) \rangle_{\mathcal{P}} \approx \frac{1}{t - t_0} \sum_{n=t_0}^{t} f(\mathbf{s}^{(n)})$$

- Consecutive samples $\mathbf{s}^{(t)}$ are dependent but we can "pretend" as if they are independent!

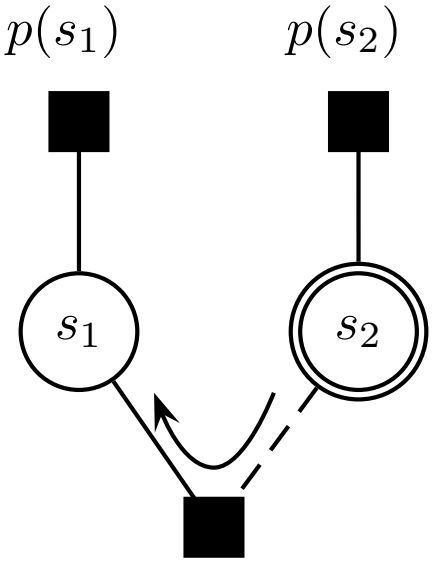
Gibbs Sampler



$$p(x = \hat{x}|s_1, s_2)$$

$$s_2^{(t+1)} \sim \mathcal{N}(s_2; m_2(s_1^{(t+1)}), S_2)$$

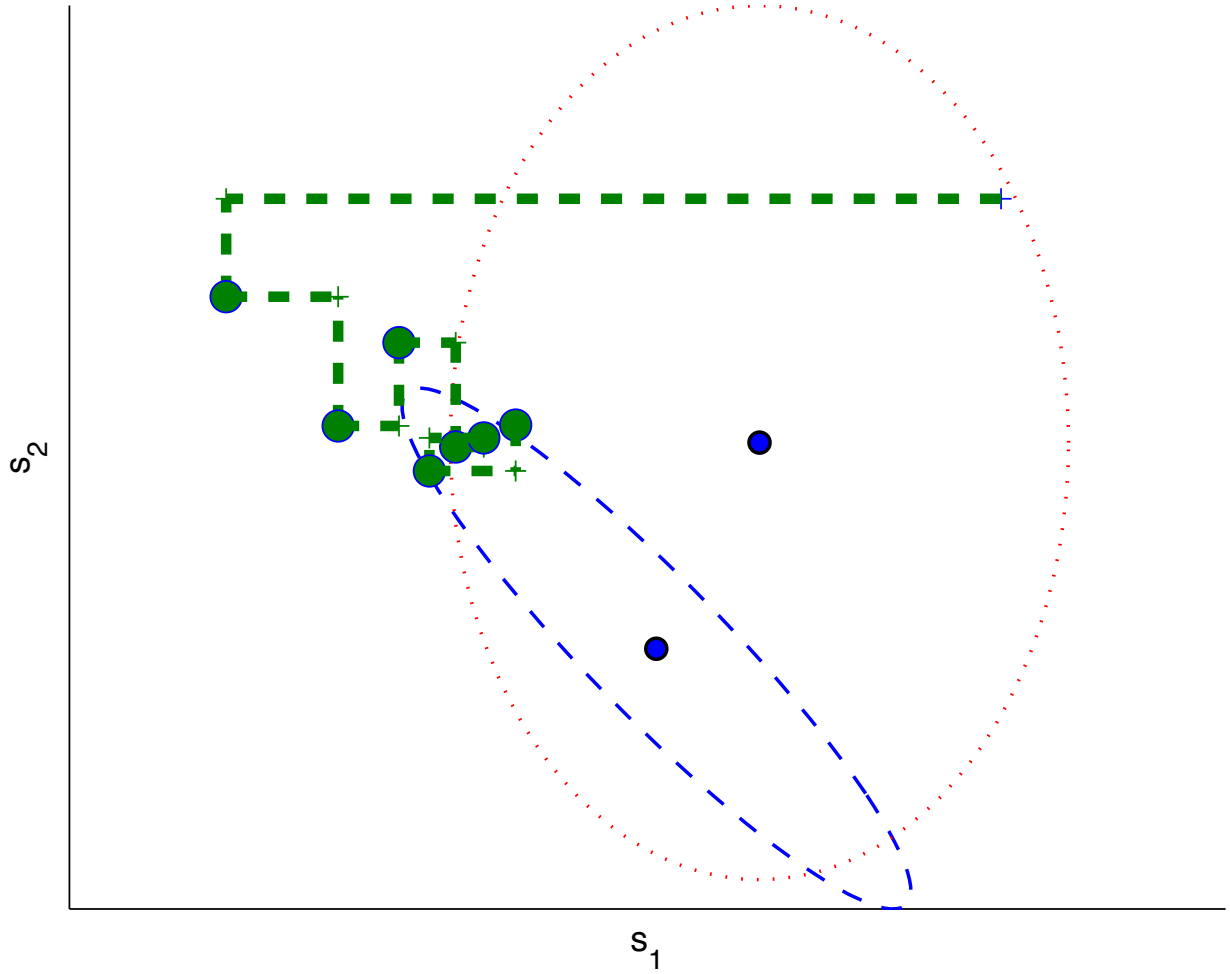
Gibbs Sampler



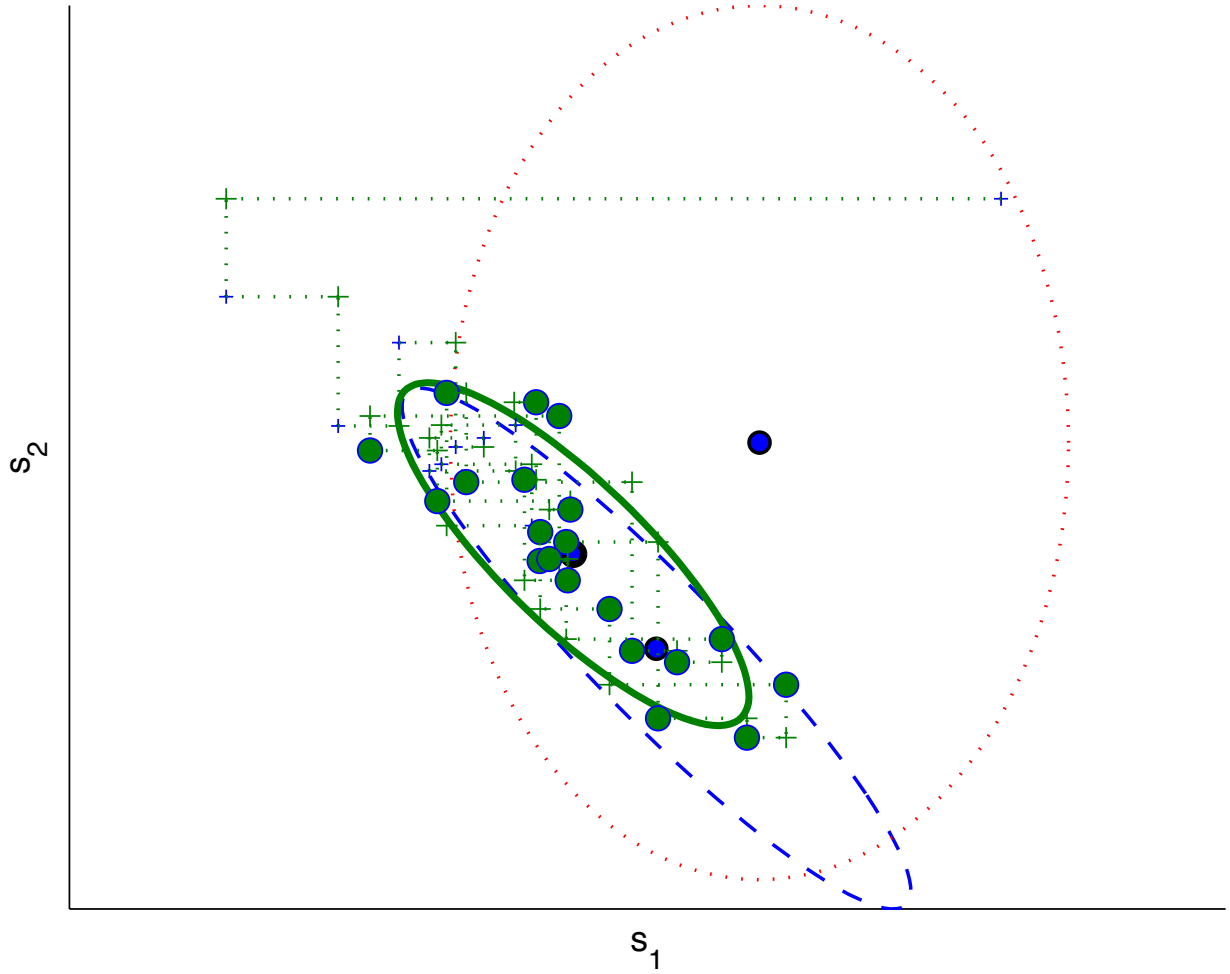
$$p(x = \hat{x}|s_1, s_2)$$

$$s_1^{(t+1)} \sim \mathcal{N}(s_1; m_1(s_2^{(t)}), S_1)$$

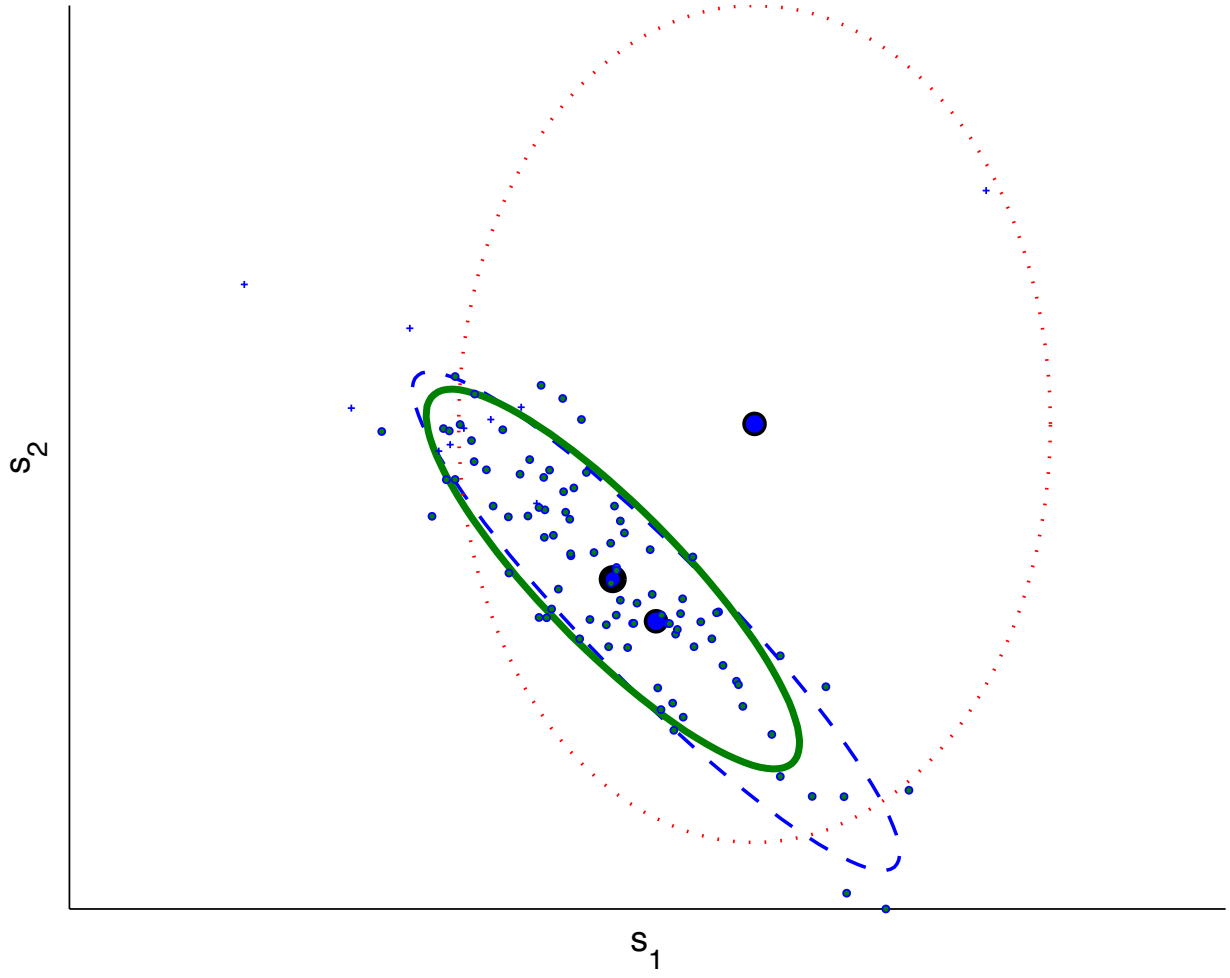
Gibbs Sampler



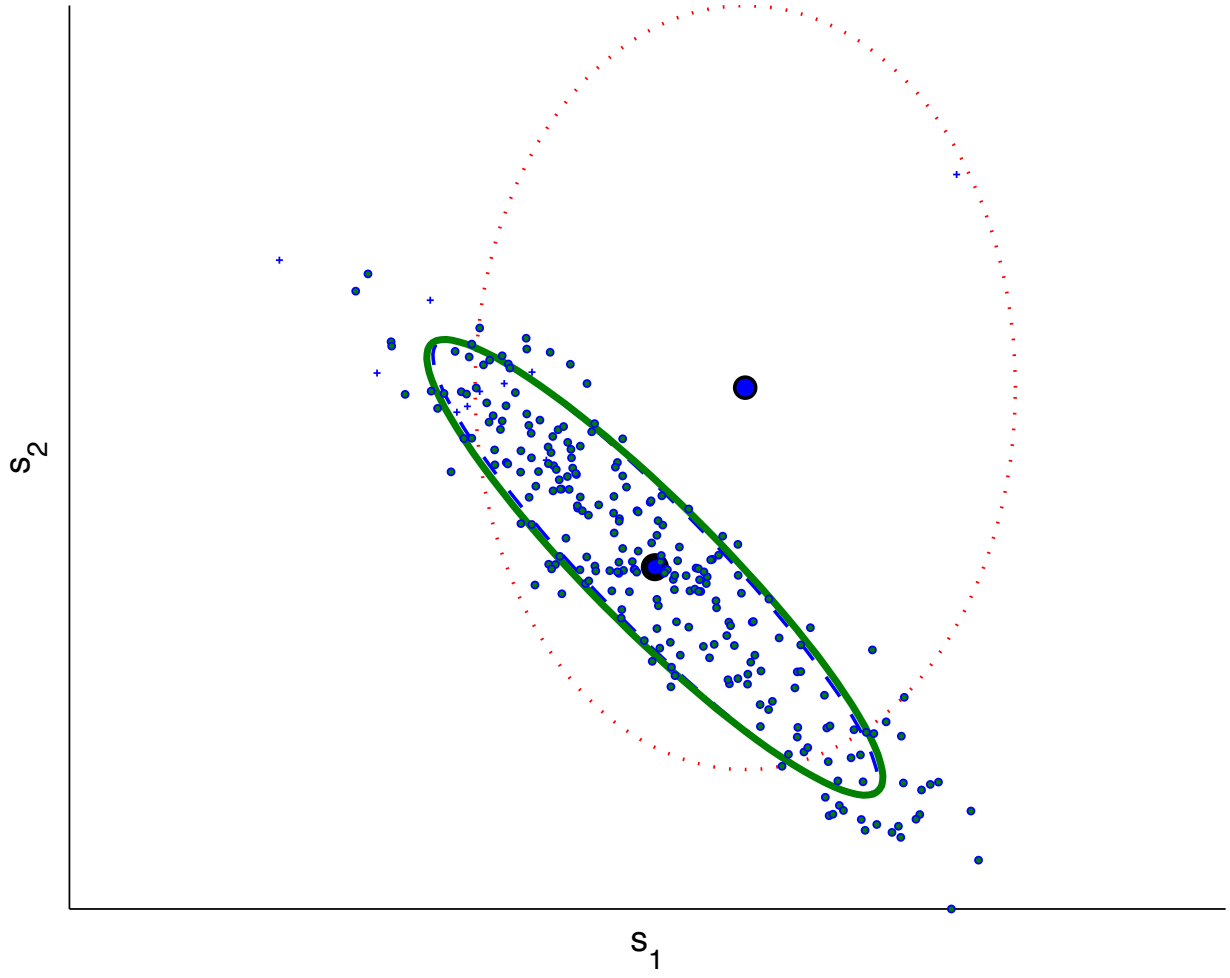
Gibbs Sampler, $t = 20$



Gibbs Sampler, $t = 100$



Gibbs Sampler, $t = 250$



Technicalities: Finding the full conditionals

$$s_1^{(t+1)} \sim p(s_1|s_2^{(t)}, x=\hat{x}) \propto \phi(s_1, s_2^{(t)}) \equiv \phi_1$$

Eliminate terms that don't depend on s_1

$$\log \phi_1 = \log p(x=\hat{x}|s_1, s_2^{(t)}) + \log p(s_1) + \log p(s_2^{(t)})$$

$$=^{+} \underbrace{\mu_1^\top P_1^{-1} s_1 - \frac{1}{2} s_1{}^\top P_1^{-1} s_1}_{\log p(s_1)} + \underbrace{\hat{x}^\top R^{-1}(s_1 + s_2^{(t)}) - \frac{1}{2}(s_1 + s_2^{(t)})^\top R^{-1}(s_1 + s_2^{(t)})}_{p(x=\hat{x}|s_1, s_2^{(t)})}$$

Technicalities: Finding the full conditionals (cont.)

$$\log \phi_1 \quad =^{+} \quad \left(\mu_1^\top P_1^{-1} + (\hat{x} - s_2^{(t)})^\top R^{-1}\right) s_1 - \frac{1}{2} \mathbf{Tr}\left(P_1^{-1} + R^{-1}\right) s_1 s_1{}^\top$$

$$p(s_1 | s_2^{(t)}, x = \hat{x}) \quad = \quad \mathcal{N}(s_1; m_1, S_1)$$

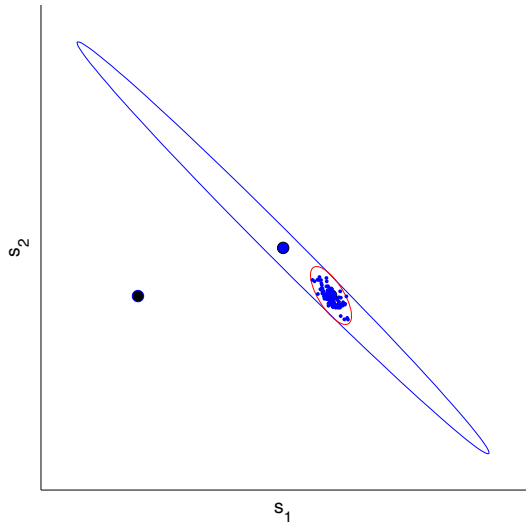
$$\begin{aligned} S_1 &= \left(P_1^{-1} + R^{-1}\right)^{-1} \\ m_1(s_2^{(t)}) &= S_1 \left(P_1^{-1}\mu_1 + R^{-1}(\hat{x} - s_2^{(t)})\right) \end{aligned}$$

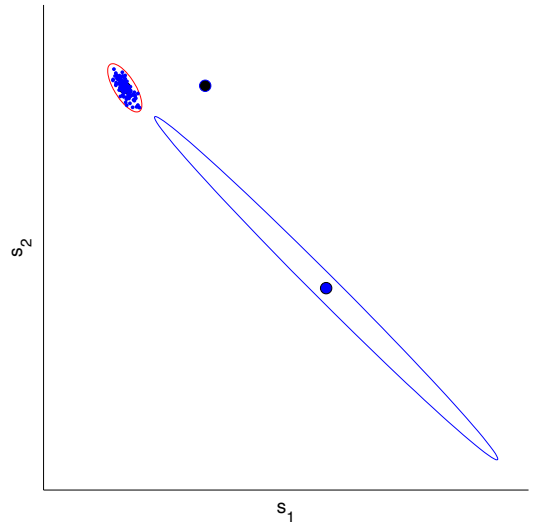
The transition kernel

$$
\begin{aligned}
T(s_{1:2}^{(t+1)}|s_{1:2}^{(t)}, s_2^{(t)}) &= T(s_2^{(t+1)}|s_1^{(t+1)}, s_1^{(t)}, s_2^{(t)}) T(s_1^{(t+1)}|s_1^{(t)}, s_2^{(t)}) \\
&= T(s_2^{(t+1)}|s_1^{(t+1)}) T(s_1^{(t+1)}|s_2^{(t)}) \\
&= \mathcal{N}(s_2^{(t+1)}; m_2(s_1^{(t+1)}), S_2) \mathcal{N}(s_1^{(t+1)}; m_1(s_2^{(t)}), S_1)
\end{aligned}
$$

Therefore, the transition kernel is also Gaussian.

The transition kernel





- But why does the chain converge to the target distribution?
- The Gibbs sampler is a special MH algorithm

Metropolis-Hastings

Algorithm 2 Metropolis-Hastings

1: Initialize: $x^{(0)} \sim q(x_0)$
2: **for** $i = 1, 2 \ldots$ **do**
3: Propose: $\xi^{(i)} \sim q(x|x^{(i-1)})$
4: Acceptance Probability:

$$\alpha(\xi^{(i)}|x^{(i-1)}) = \min\{1, \frac{q(x^{(i-1)}|\xi^{(i)})\pi(\xi^{(i)})}{q(\xi^{(i)}|x^{(i-1)})\pi(x^{(i-1)})}\}$$

5: Uniform: $u \sim \mathcal{U}(u; 0, 1)$
6: **if** $u < \alpha$ **then**
7: Accept: $x^{(i)} \leftarrow \xi^{(i)}$
8: **else**
9: Reject: $x^{(i)} \leftarrow x^{(i-1)}$
10: **end if**
11: **end for**

The acceptance probability for Gibbs

$$
\begin{aligned}
\xi &\sim p(x_n | x_{-n} = x_{-n}^{(i-1)}) \\
\alpha &\equiv \alpha\left(\xi, x_{-n}^{(i-1)} | x_n^{(i-1)}, x_{-n}^{(i-1)}\right)
\end{aligned}
$$

$$
\begin{aligned}
\alpha &= \min\left\{1, \frac{q(x_n^{(i-1)}, x_{-n}^{(i-1)} | \xi, x_{-n}^{(i-1)}) p(\xi, x_{-n}^{(i-1)})}{q(\xi, x_{-n}^{(i-1)} | x_n^{(i-1)}, x_{-n}^{(i-1)}) p(x_n^{(i-1)}, x_{-n}^{(i-1)})}\right\} \\
&= \min\left\{1, \frac{p(x_n^{(i-1)} | x_{-n}^{(i-1)}) p(\xi, x_{-n}^{(i-1)})}{p(\xi | x_{-n}^{(i-1)}) p(x_n^{(i-1)}, x_{-n}^{(i-1)})}\right\} \\
&= \min\left\{1, \frac{p(x_n^{(i-1)} | x_{-n}^{(i-1)}) p(\xi | x_{-n}^{(i-1)}) p(x_{-n}^{(i-1)})}{p(\xi | x_{-n}^{(i-1)}) p(x_n^{(i-1)} | x_{-n}^{(i-1)}) p(x_{-n}^{(i-1)})}\right\} = 1
\end{aligned}
$$

Example: A change-point model

- We have a sequence of counts x_j.
- The average of the counts is jumping to a new value after an unknown index

Generative model

- We can model the counts with Poisson random variables $x_j, j = 1 \ldots M$

$$\begin{aligned} \mathcal{PO}(x;\lambda) &= e^{-\lambda}\frac{\lambda^x}{x!} \\ &= \exp\left(+x\log\lambda - \lambda - \log(x!)\right) \end{aligned}$$

- The unknown intensity can be modelled by a Gamma random variable

$$\begin{aligned} \mathcal{G}(\lambda;a,b) &\equiv \frac{1}{\Gamma(a)}b^a\lambda^{a-1}\exp(-b\lambda) \\ &= \exp\left((a-1)\log\lambda - b\lambda - \log\Gamma(a) + a\log b\right) \end{aligned}$$

- The unknown intensity λ_1 is jumping to a new unknown value λ_2 after an unknown index m

Generative model

$$
\begin{aligned}
m &\sim \mathcal{U}\{1 \dots M\} \\
\lambda_i &\sim \mathcal{G}(\lambda_i; a, b) \\
x_j &\sim \begin{cases} \mathcal{PO}(x_j; \lambda_1) & 1 \leq j \leq m \\ \mathcal{PO}(x_j; \lambda_2) & m < j \leq M \end{cases}
\end{aligned}
$$

Goal: Compute $p(\lambda_1, \lambda_2, m | x_{1:M})$

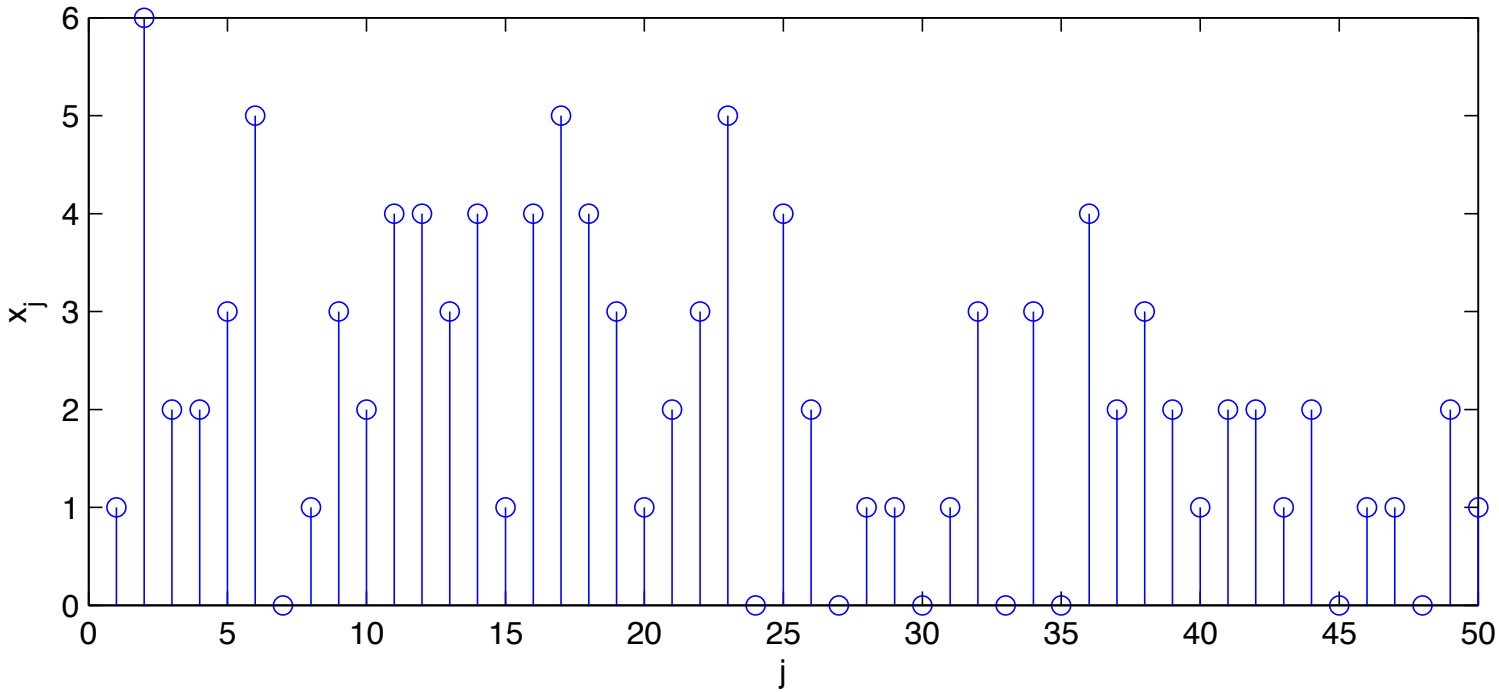
Generate data

```
% Fix the seeds for reproducibility
randn('seed', 1); rand('seed', 1);

% Hyperparameters
data.M = 50;
data.a = 2; data.b = 1;

% Changepoint
data.m = ceil(data.M*rand);

% Intensities, % !! Beware of the Gamma parametrisation !!
data.lambda = gamrnd(data.a, 1/data.b, [1 2]);

Lambda = zeros(1, M);
Lambda(1:data.m) = data.lambda(1);
Lambda((data.m+1):data.M) = data.lambda(2);

% Counts
data.x = poissrnd(Lambda);
```

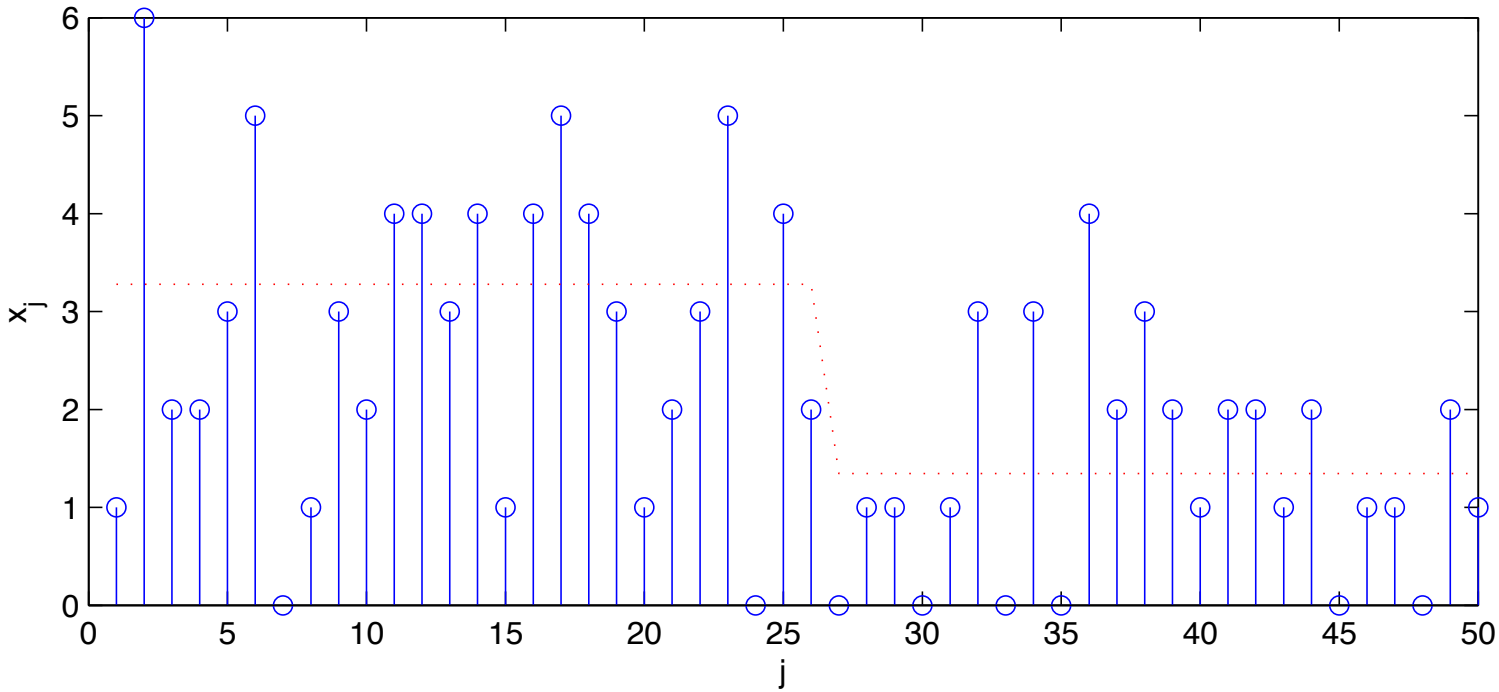
Generate data (cont.)

```
%% Visualise
stem(data.x); xlabel('j'); ylabel('x_j')
```

A Change-point model



A Change-point model



Full Joint Density

$$
\begin{aligned}
p(x_{1:M}, \lambda_1, \lambda_2, m) &= p(x_{1:M}|\lambda_1, \lambda_2, m)p(\lambda_1)p(\lambda_2)p(m) \\
&= \left(\prod_{j=1}^{m} p(x_j|\lambda_1)\right)\left(\prod_{j=m+1}^{M} p(x_j|\lambda_2)\right)p(\lambda_1)p(\lambda_2)p(m)
\end{aligned}
$$

Full Conditional Distributions

$$
\begin{aligned}
\mathcal{L} &= \log p(x_{1:M}|\lambda_1,\lambda_2,m) + \log p(\lambda_1) + \log p(\lambda_2) + \log p(m) \\
&= \sum_{j=1}^{m} (+x_j \log\lambda_1 - \lambda_1 - \log(x_j!)) \\
&\quad + \sum_{j=m+1}^{M} (+x_j \log\lambda_2 - \lambda_2 - \log(x_j!)) \\
&\quad +(a-1)\log\lambda_1 - b\lambda_1 - \log\Gamma(a) + a\log b \\
&\quad +(a-1)\log\lambda_2 - b\lambda_2 - \log\Gamma(a) + a\log b - \log M
\end{aligned}
$$

Full Conditional Distributions

$$
\begin{aligned}
\log p(\lambda_1 | m, \lambda_2, x_{1:M}) &=^{+} \sum_{j=1}^{m} (+x_j \log \lambda_1 - \lambda_1) \\
&\quad +(a-1)\log\lambda_1 - b\lambda_1 \\
&= \left(a + \sum_{j=1}^{m} x_j - 1 \right) \log\lambda_1 - (m+b)\lambda_1 \\
&=^{+} \log \mathcal{G}(a + \sum_{j=1}^{m} x_j, m+b)
\end{aligned}
$$

Full Conditional Distributions

$$
\begin{aligned}
\log p(\lambda_2 | m, \lambda_1, x_{1:M}) &=^{+} \sum_{j=m+1}^{M} (+x_j \log \lambda_2 - \lambda_2) \\
&\quad +(a-1)\log \lambda_2 - b\lambda_2 \\
&=^{+} \log \mathcal{G}(a + \sum_{j=m+1}^{M} x_j, M - m + b)
\end{aligned}
$$

Full Conditional Distributions

$$
\begin{aligned}
\log p(m|\lambda_1, \lambda_2, x_{1:M}) &= \sum_{j=1}^{m} (+x_j \log \lambda_1 - \lambda_1 - \log(x_j!)) \\
&\quad + \sum_{j=m+1}^{M} (+x_j \log \lambda_2 - \lambda_2 - \log(x_j!)) \\
&= + \left(\sum_{j=1}^{m} x_j \right) \log \lambda_1 - m\lambda_1 \\
&\quad + \left(\sum_{j=m+1}^{M} x_j \right) \log \lambda_2 - (M-m)\lambda_2
\end{aligned}
$$

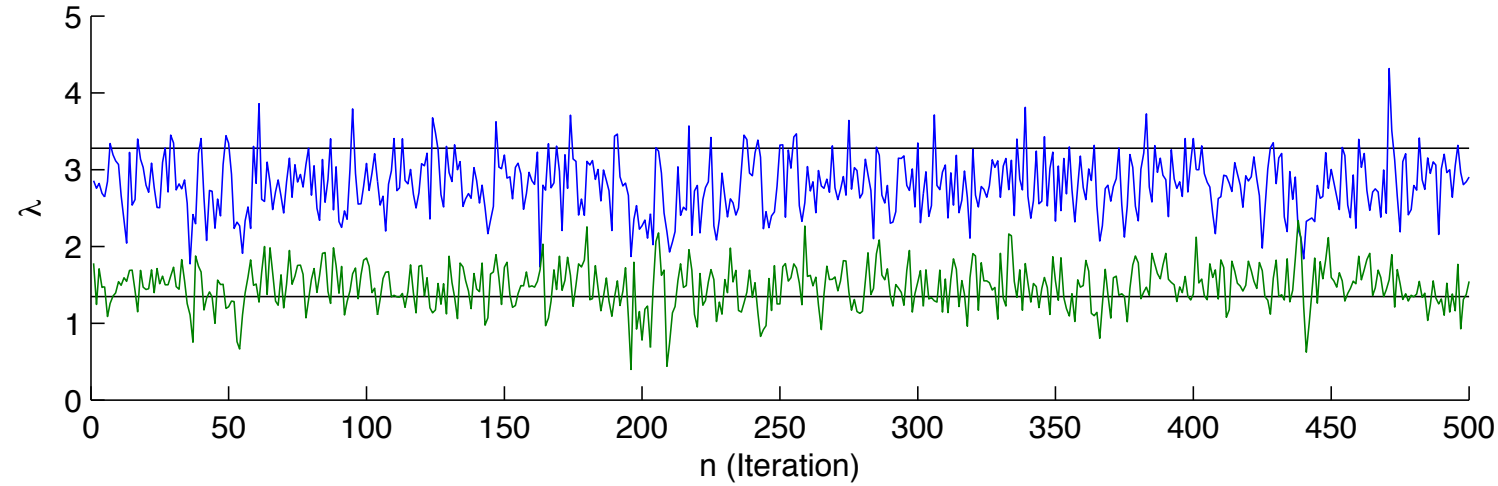
The Gibbs sampler

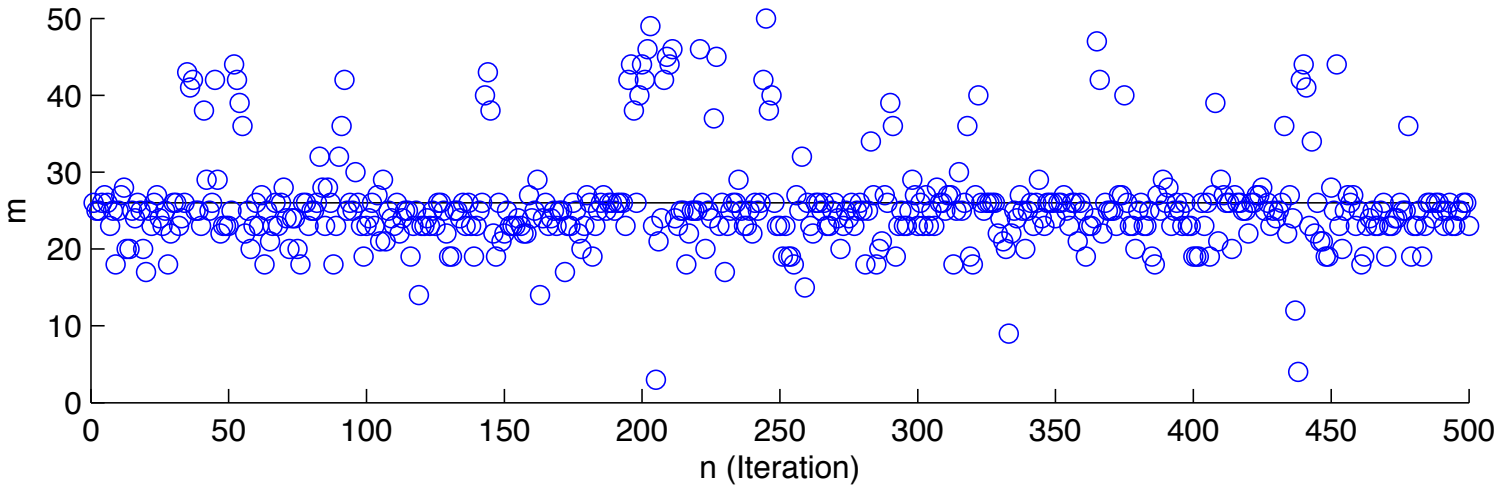
```
% Gibbs sampler
EP = 5200; % Number of epochs
BURN_IN = 200;

% Initial state of the Markov  chain
m = 10; lam = zeros(2,1);
% Storage
chain.m = zeros(1, EP-BURN_IN);
chain.lambda = zeros(2, EP-BURN_IN);
```

The Gibbs sampler (cont.)

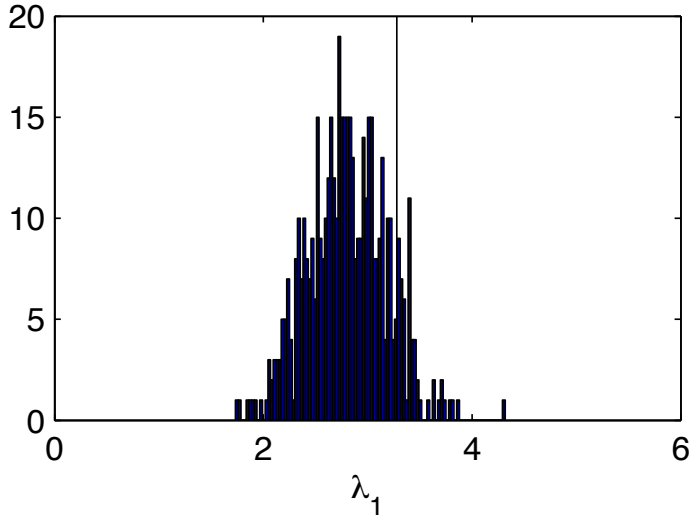
```
for e=1:EP,
  % lambda ~ p(lambda | x, m)
  lam(1) = gamrnd(data.a + sum(data.x(1:m)), 1/(m+data.b) );
  lam(2) = gamrnd(data.a + sum(data.x(m+1:end)), 1/(data.M-m+da

 % m ~ p(m | x, lambda)
  lp = zeros(1, data.M);
  for i=1:data.M,
    lp(i) = sum(data.x(1:i)).*log(lam(1)) - i*lam(1) +...
 sum(data.x((i+1):data.M)).*log(lam(2)) - (data.M-i)*lam(2) ;
  end;
  m = randgen(exp(lp-max(lp)));

  if e>BURN_IN,
    chain.lambda(:, e-BURN_IN) = lam;
    chain.m(1, e-BURN_IN) = m;
  end;
end;
```

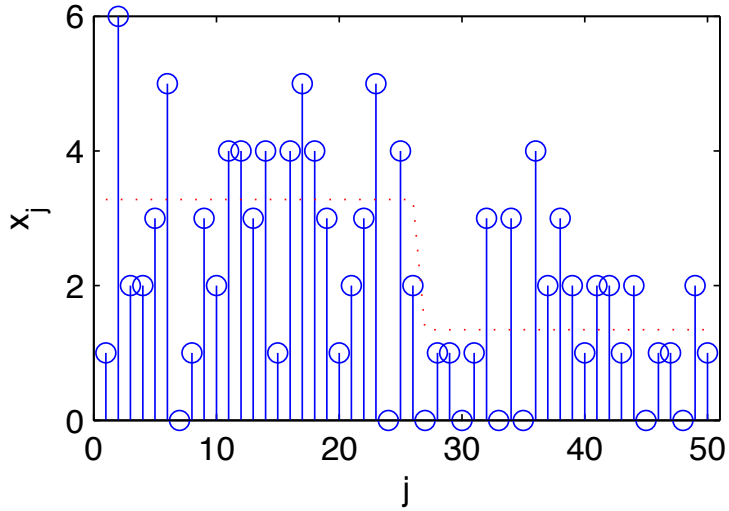
Results

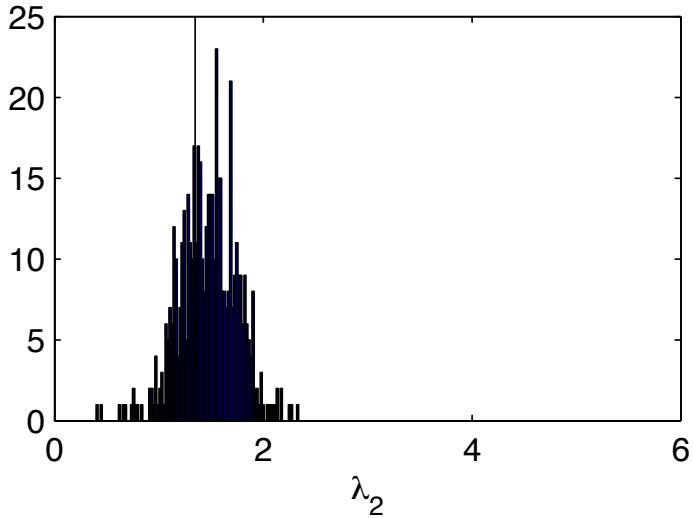


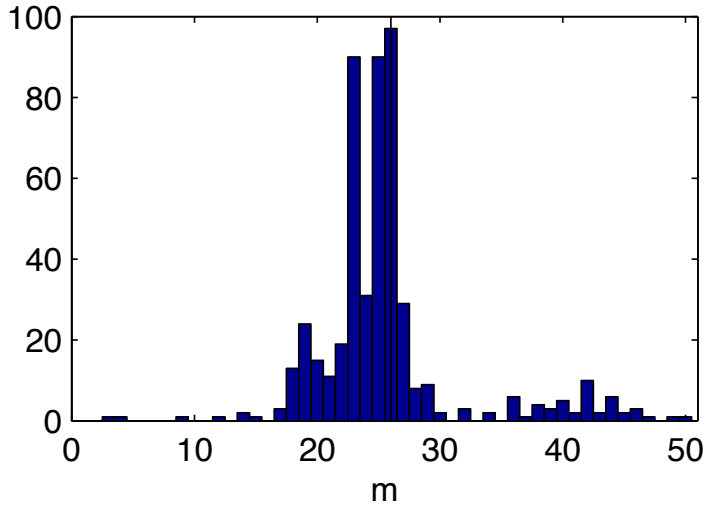


Results

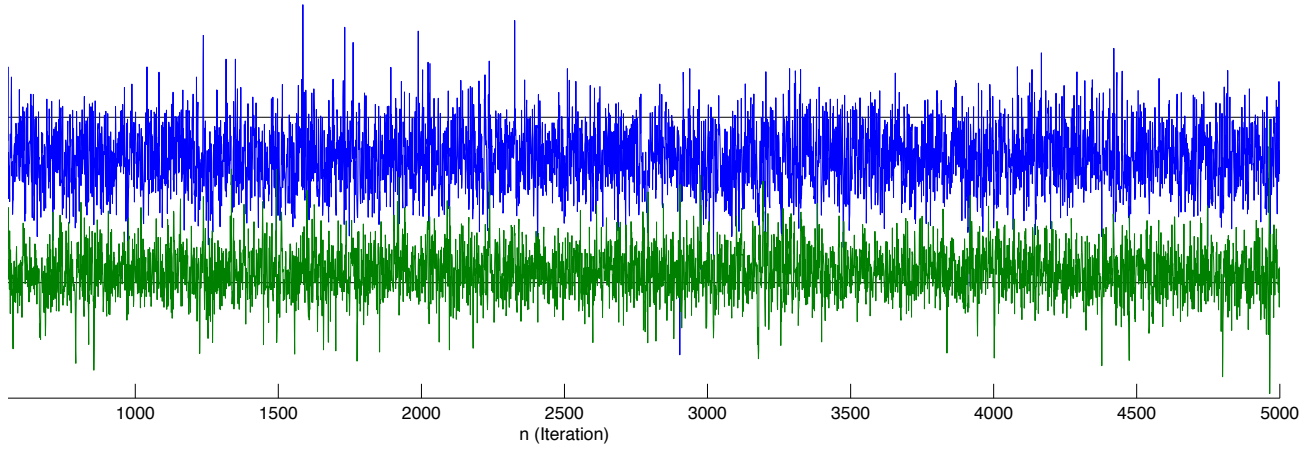

20
15
10
5
0
0
2
4
6
λ1

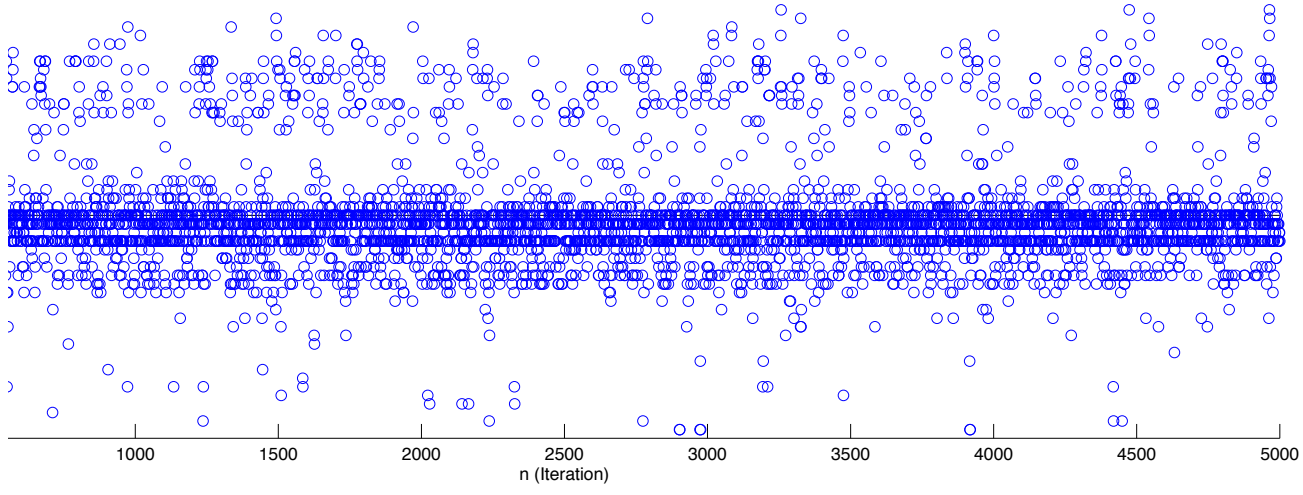

6
4
2
0
xj
0
10
20
30
40
50
j


25
20
15
10
5
0
0
2
4
6
λ2

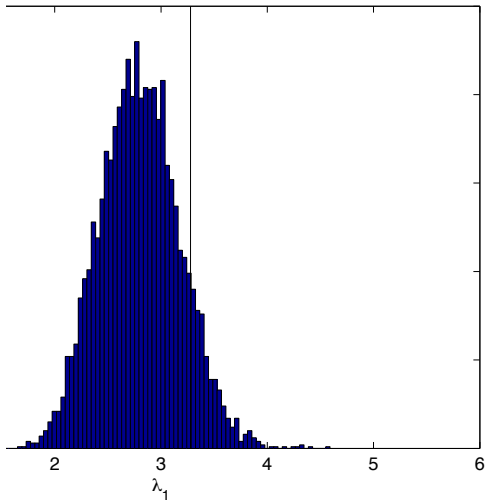

100
80
60
40
20
0
0
10
20
30
40
50
m

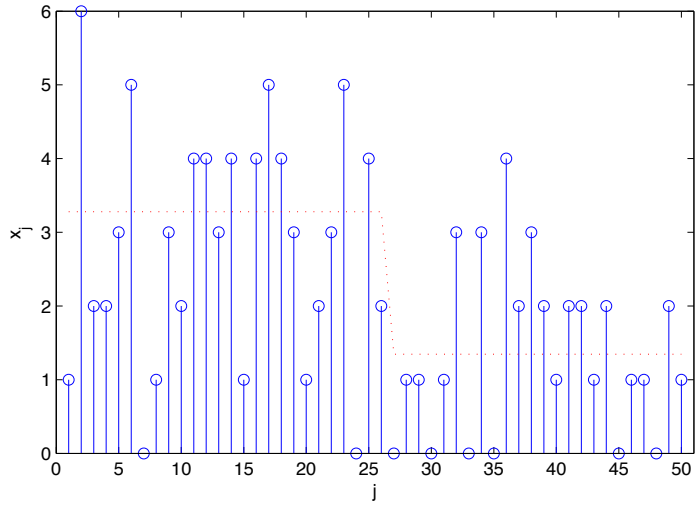
Results, Longer chain

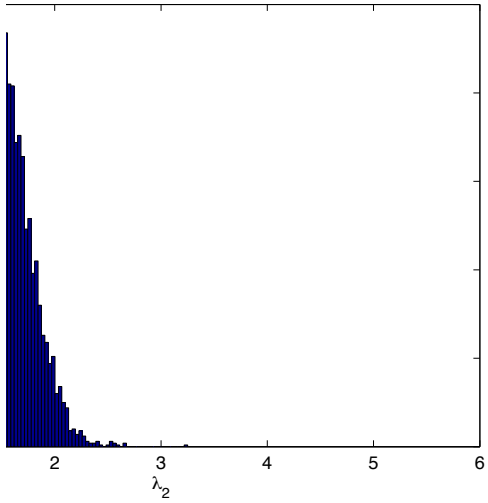


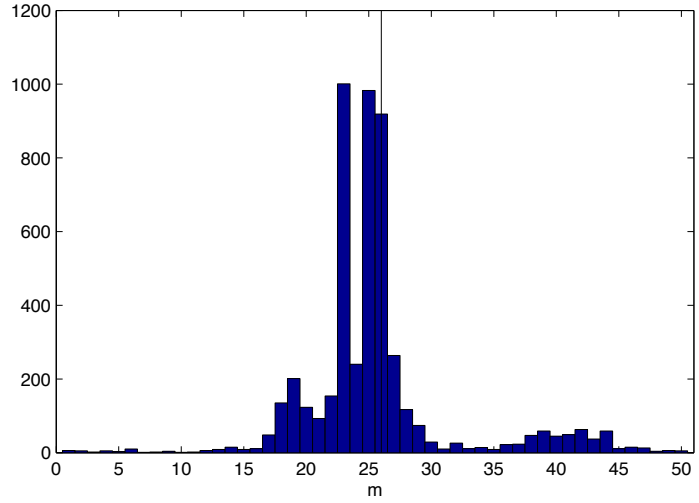


Results, Longer chain









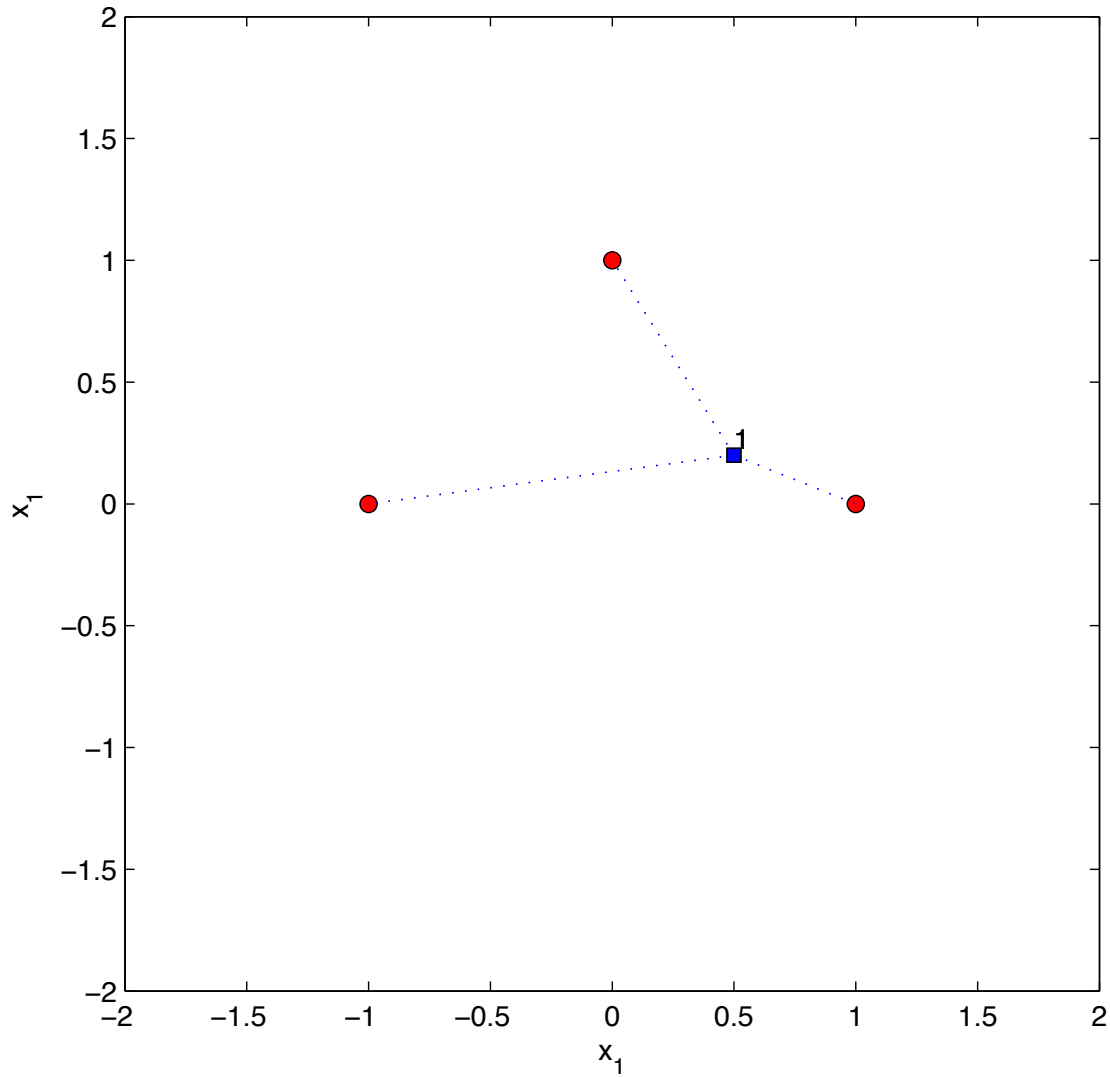
Metropolis Hastings Example

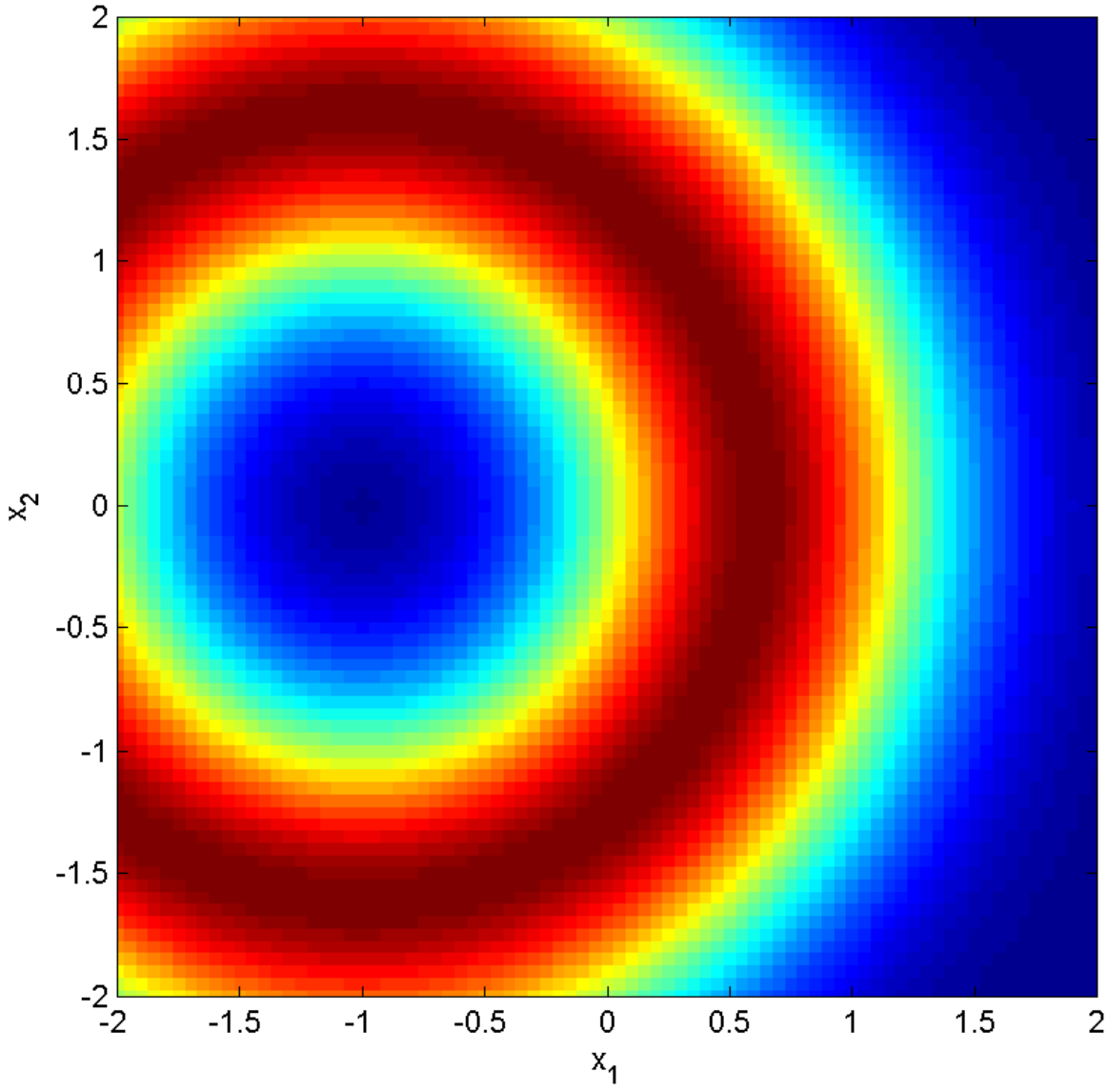
- Problem : Object position estimation from range measurements
- We have L sensors at known positions s_j, $j = 1 \ldots L$, each measuring the distance of a point object with noise

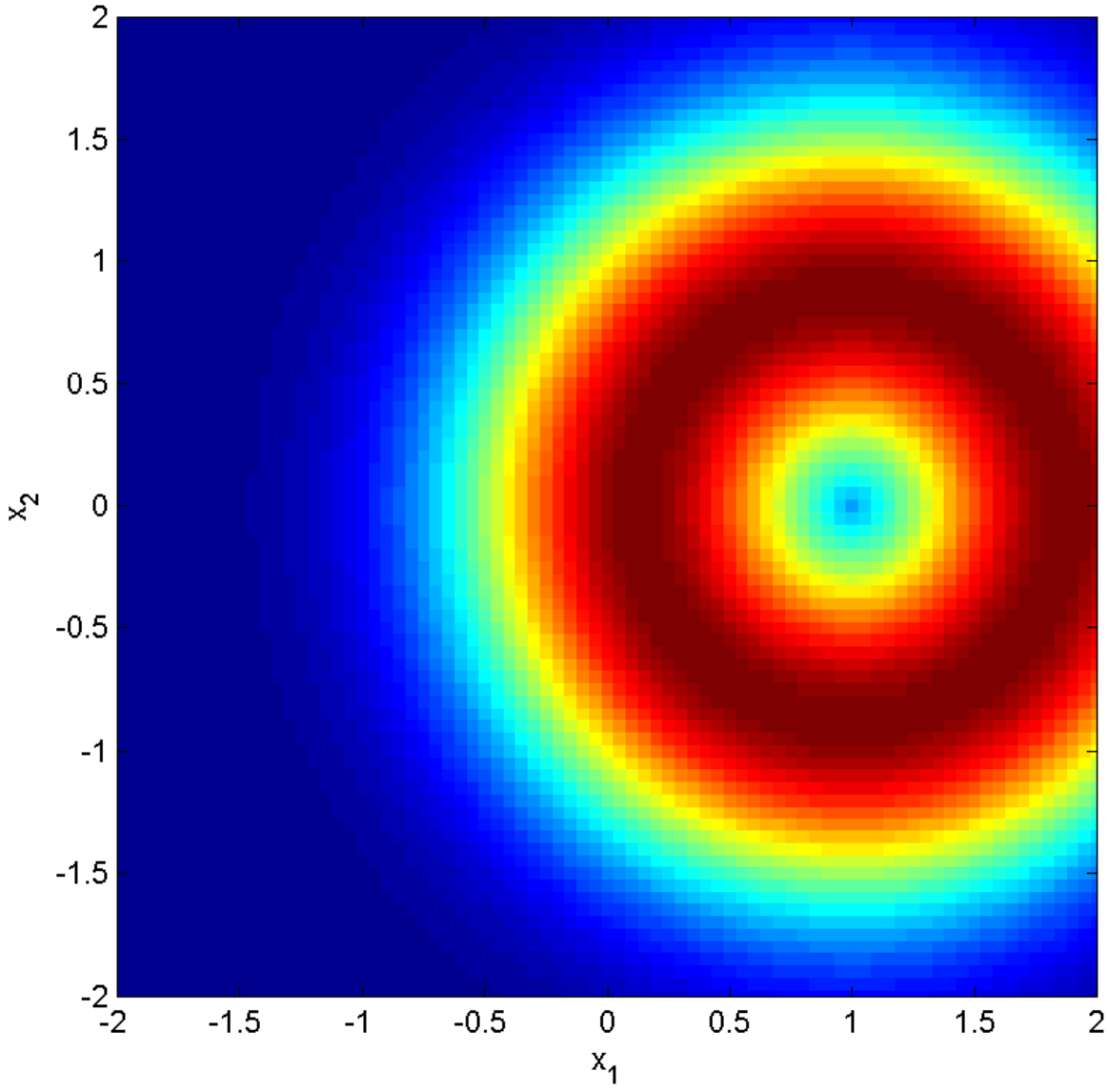
Range measurements

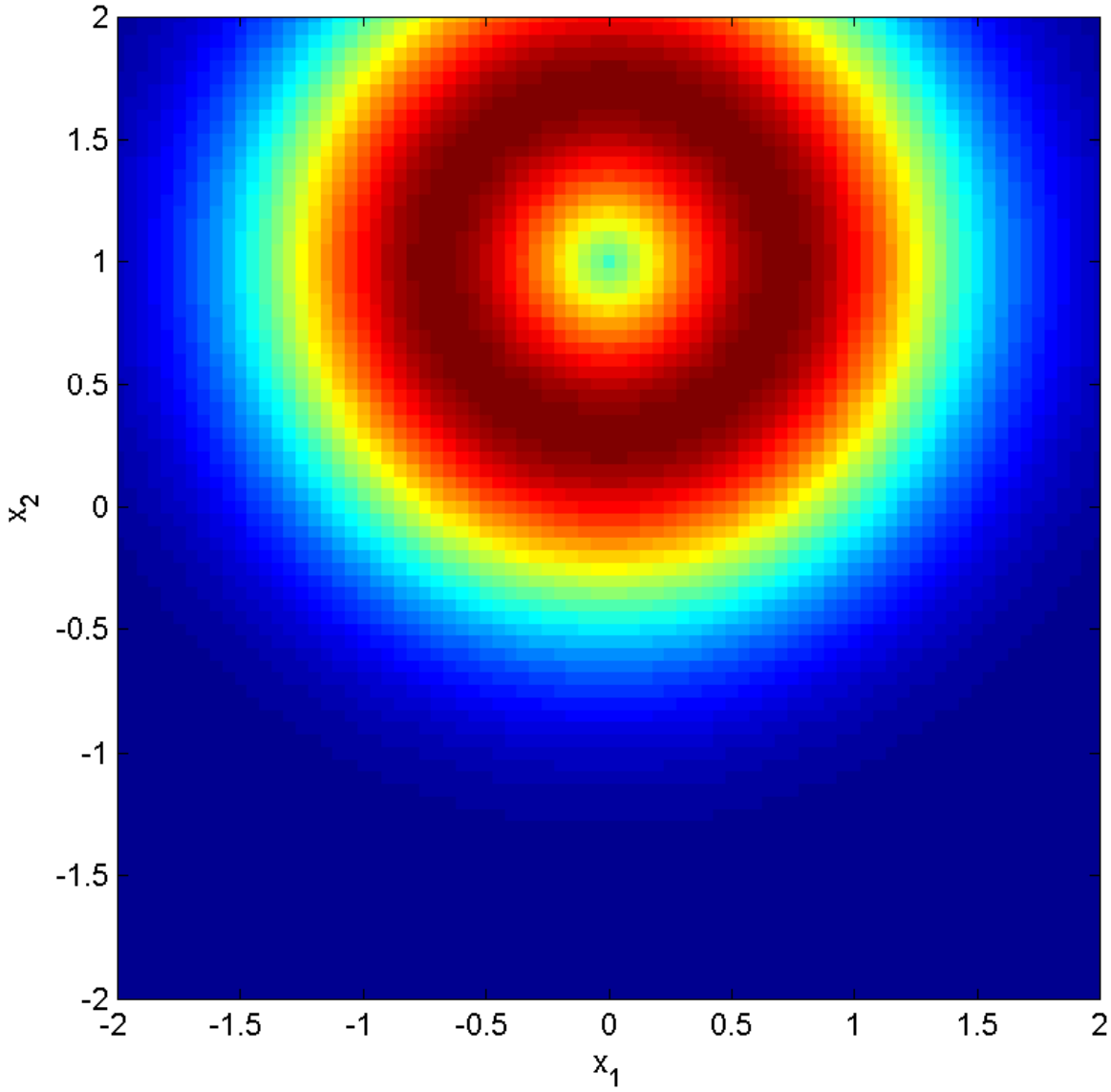
$$
\begin{aligned}
x &\sim p(x) \propto 1 \\
y_j | x, s_j &\sim \mathcal{N}(y_j; \|x - s_j\|, R)
\end{aligned}
$$

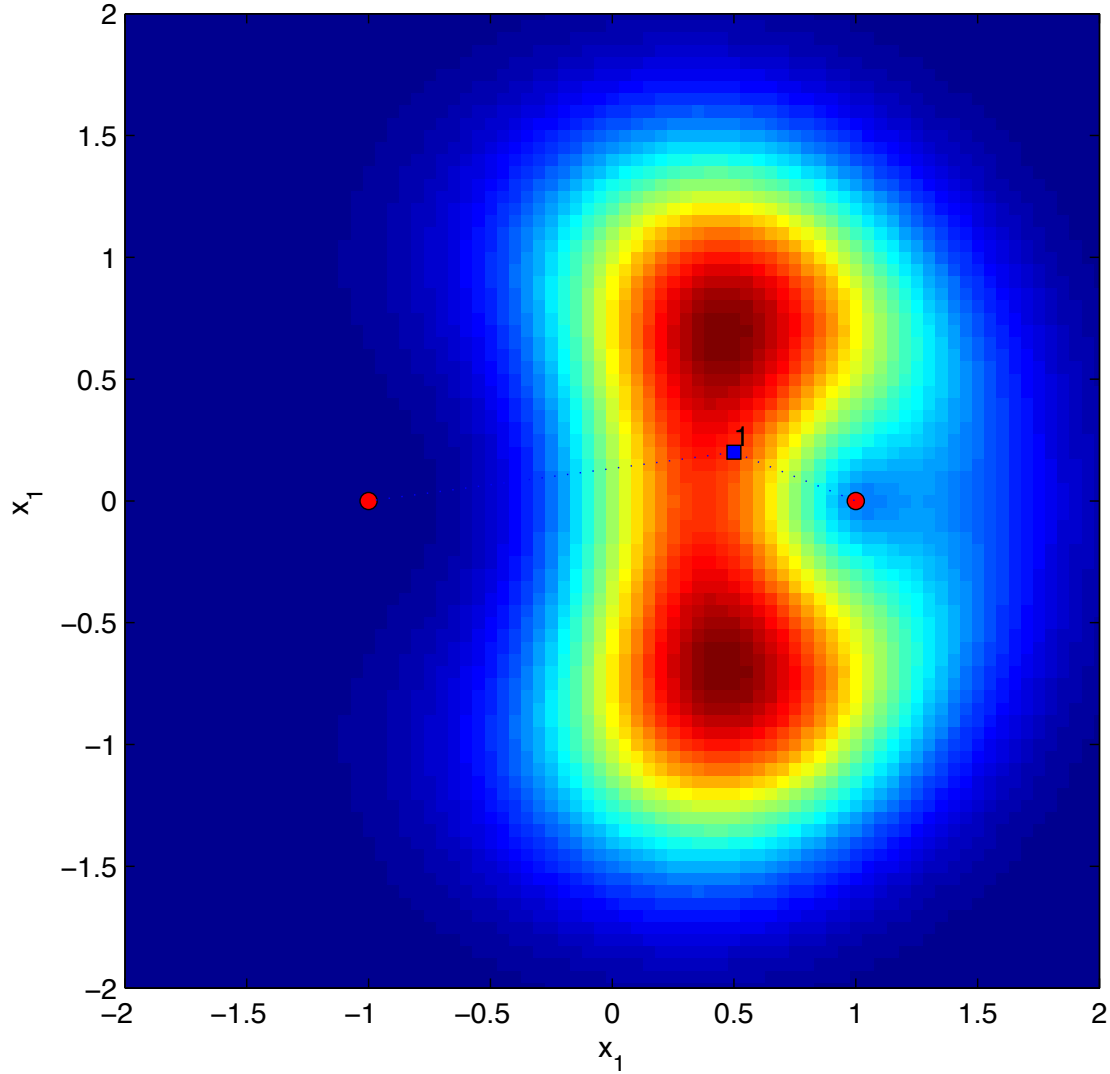
$$
\|x\| \equiv \left(\sum_k x_k^2 \right)^{1/2}
$$

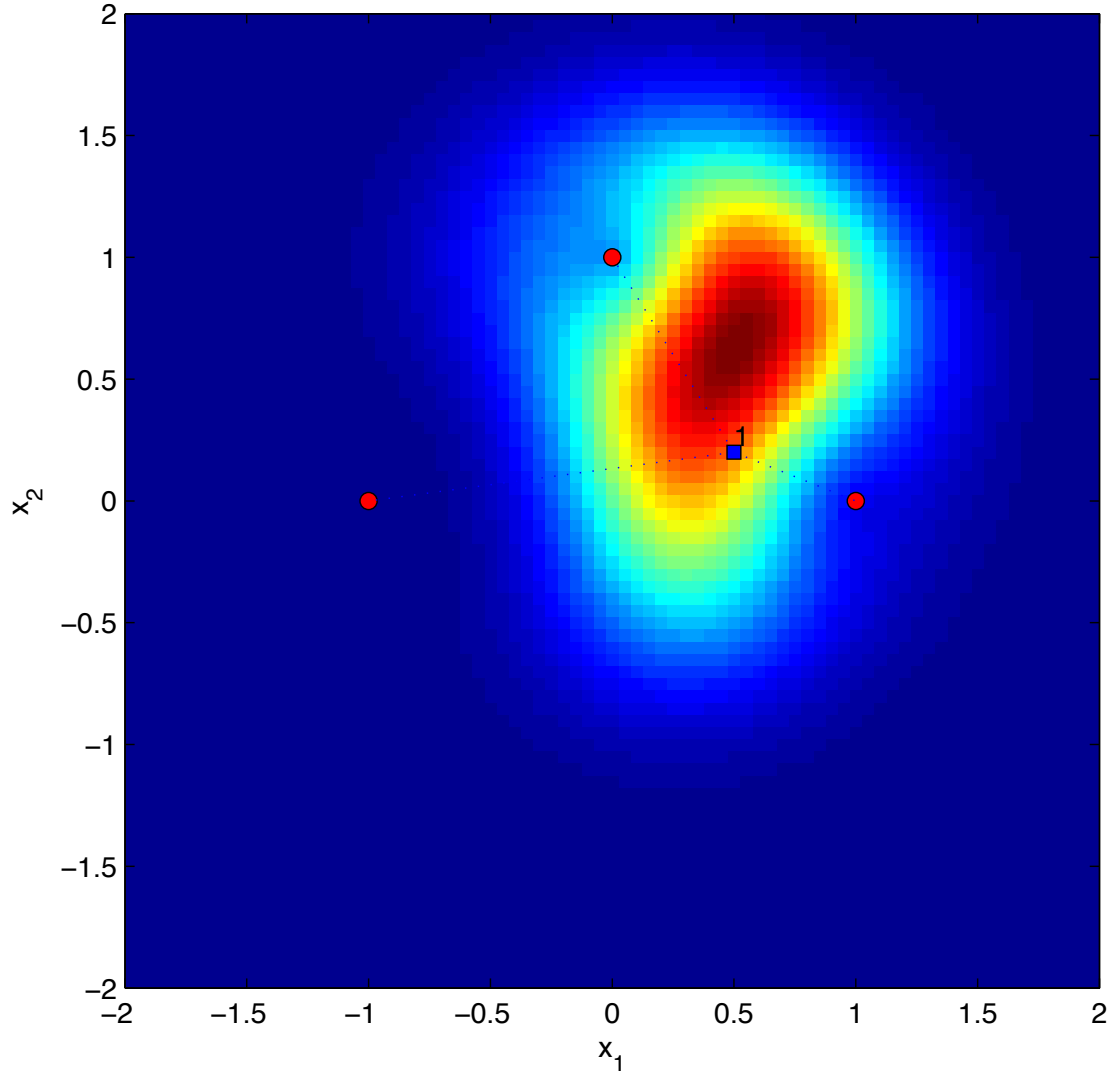

2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
x1
x1
1


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
x2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
x1


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
x_2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
x_1


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
x2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
x1


2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
x1
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
x1
1


2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
x2
x1
1

Generative Model

```
% Fix the seeds for reproducibility
randn('seed', 11); rand('seed', 1);

% Sensor positions
s = [-1 0; 1 0; 0 1]';
L = size(s, 2); % Number of sensors

M = 1; % Number of targets
x = [0.5;0.2]; % True Target position
R_r = 0.3; %% Range sensor noise variance

m_r = zeros(M, L);
for i=1:M, % For each target
  % Find the true distance from the sensor
  d = repmat(x(:,i), [1 L] ) - s;
  m_r(i, :) = sqrt(d(1,:).^2 + d(2,:).^2);
end;

y_r = sqrt(R_r)*randn(size(m_r)) + m_r;
```

Finding numerically the true posterior

```
% Prepare a uniform grid
xx = -2:0.05:2;
[X1 X2] = meshgrid(xx);

llk = zeros([size(X1) L]);

i = 1; %% Taget idx, assume M = 1;
for j=1:L,
  d = sqrt((X1-s(1,j)).^2 + (X2 - s(2, j)).^2);
  llk(:,:,j) = -0.5*(d-y_r(i, j)).^2/R_r -0.5*log(2*pi*R_r);
end;
lk = sum(llk, 3);

imagesc(xx, xx, exp(lk-max(lk(:))))
```

Metropolis

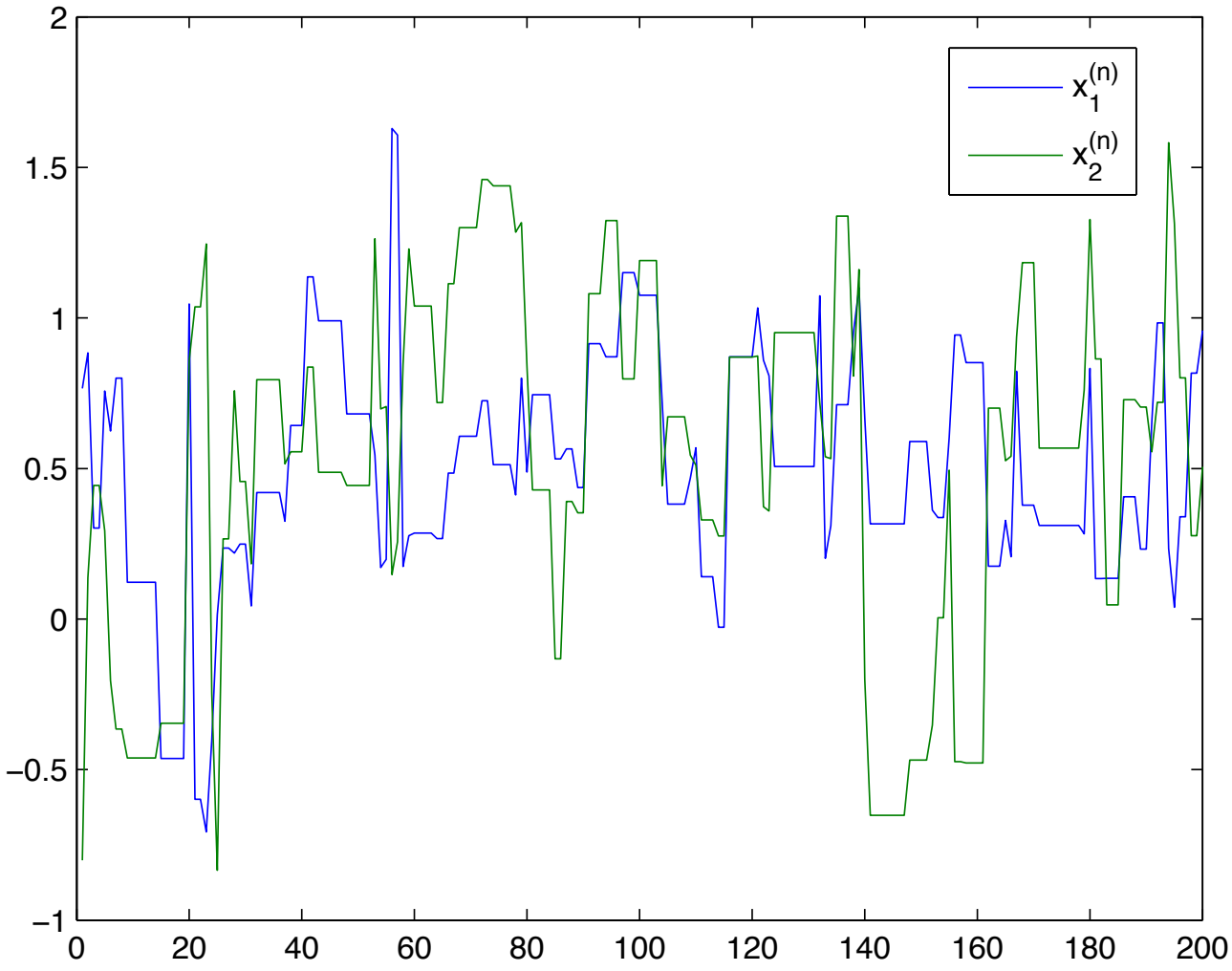
```
% Metropolis
EP = 50200; % Number of epochs
BURN_IN = 200;
R_q = 0.4; % proposal variance
i = 1; % Which target ?
% Storage
chain.x = zeros(2, EP-BURN_IN);
chain.lk = zeros(1, EP-BURN_IN);
```

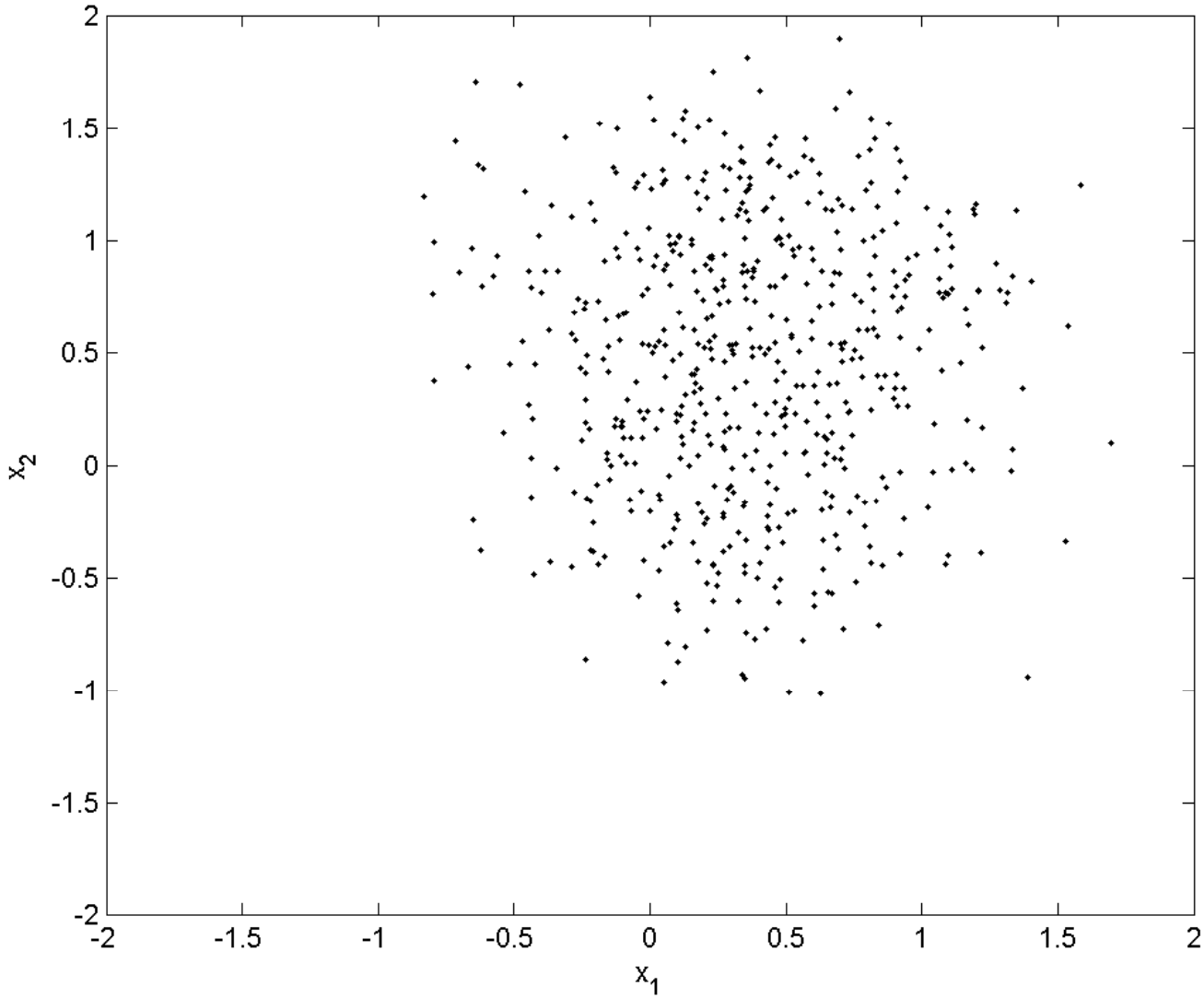
Metropolis (cont.)

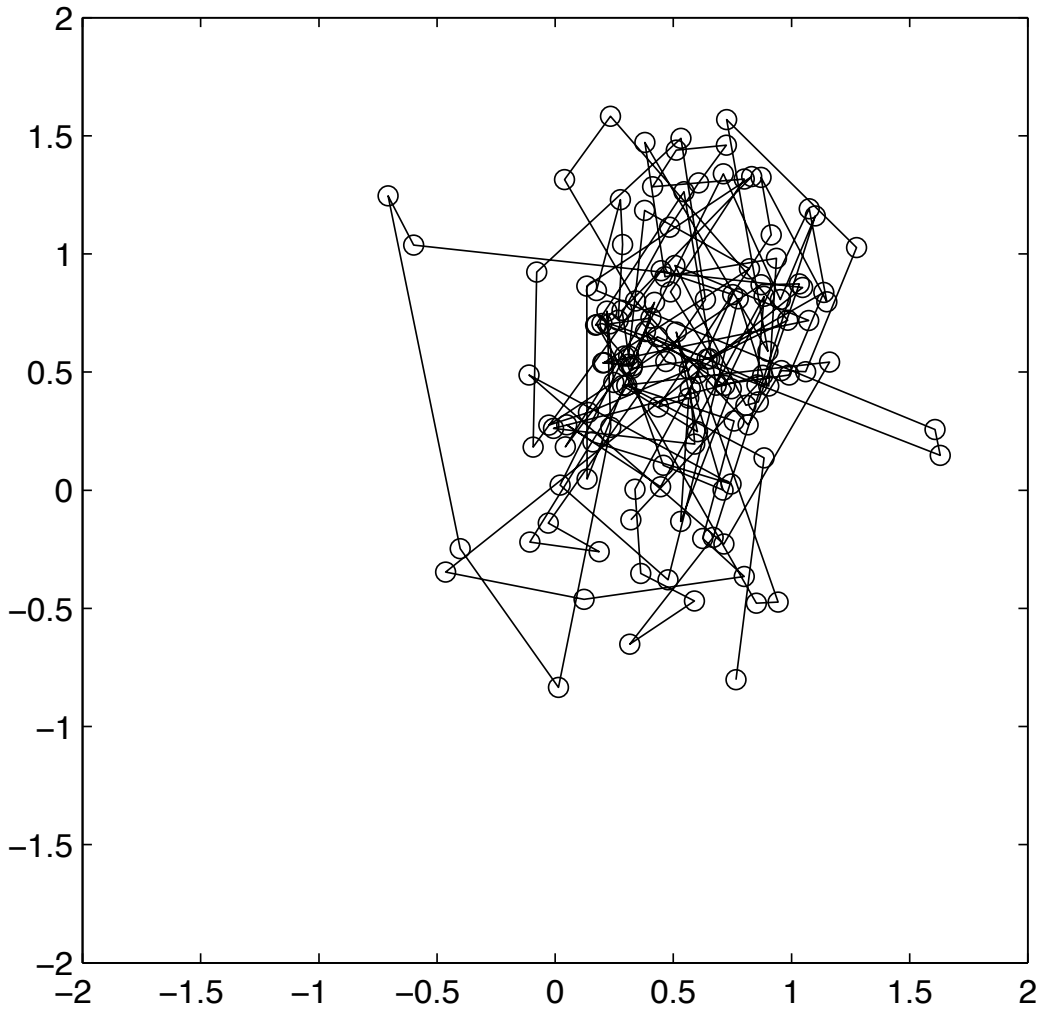
```
x_c = [0; 0]; lk = -Inf;
for e=1:EP,
 % propose a new position
  x_new = x_c + sqrt(R_q)*randn(size(x_c));
 % Evaluate the unnormalised joint posterior
  d = sqrt((x_new(1)-s(1,:)).^2 + (x_new(2) - s(2, :)).^2);
  lk_new = sum(-0.5*(d-y_r(i, :)).^2/R_r -0.5*log(2*pi*R_r));
 % Compute the log-acceptance probability
  log_a = min(0, lk_new - lk);

  if log(rand)<log_a, % Accept,
    x_c = x_new; lk = lk_new;
  end;   % else reject

  if e>BURN_IN,
    chain.x(:, e-BURN_IN) = x_c;
    chain.lk(1, e-BURN_IN) = lk;
  end;
end;
```

Metropolis (cont.)

```
[z] = hist3(chain.x', 'ctrs', {xx',xx' });
imagesc(xx, xx, z'); set(gca, 'ydir', 'n')
xlabel('x_1');
ylabel('x_2');
```


2
1.5
1
0.5
0
−0.5
−1
0
20
40
60
80
100
120
140
160
180
200
$x_1^{(n)}$
$x_2^{(n)}$


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
x2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
x1


2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2

A 2-D histogram from a longer chain