Adaptive importance Sampling

MATH30011 Project (Semester One)

Author: **Wei Wang (10425633)**

Supervisor: **Dr Simon Cotter**

MANCHESTER 1824

The University of Manchester

January 2020

Abstract

Markov chain Monte Carlo methods are a class of powerful algorithms for sampling from complicated probability distributions, which are quite hard to be sampled directly. In this thesis, we first study a typical Monte Carlo method, importance sampling, and use it to sample a simple target distribution. We demonstrate that the choice of the proposal distributions will significantly affect the accuracy of the sampling.

We will study adaptive importance sampling and learn a particle ensemble algorithm, called ensemble transport adaptive importance sampling (ETAIS). This method can be flexibly equipped with different MCMC proposal kernels and resampling algorithms. We introduce two resampling algorithms to equip ETAIS, bootstrap and multinomial transformation (MT). We implement ETAIS with Gaussian kernel and two resampling algorithms in a testing example. We demonstrate that multinomial transformation performs better than bootstrap in this example.

After successfully running our algorithm, we focus on the sampling in multi-modal distributions and problems with expensive likelihoods. We demonstrate that with the same proposal distribution ETAIS performs better than standard Markov chain Monte Carlo methods in both cases.

Contents

1 Introduction

1.1 Motivation

Markov chain Monte Carlo (MCMC) methods were first invented in 1970[12] and extended by many researchers. One of the most useful applications is to sample the posterior distributions in inverse problems within a Bayesian framework.

Although MCMC methods can sample any distribution, the cost of sampling is unacceptable. To solve the high cost of the sampling, we can understand the structure of the posterior and construct specialised proposals, which can sample our posterior within fewer iterations. These methods include the Hamiltonian Monte Carlo algorithm [19] and Riemann manifold Monte Carlo methods [20]. Another issue is that the convergence is hard to reach if we sample complex structures in low dimensions where parameters' posterior are highly correlated. Most of these cases happen in the inverse problems related to epidemiology and other biological applications [13]. We aim to find an efficient algorithm for sampling from these specific inverse problems.

In this thesis, we focus on importance sampling, which is another class of methods to sample from complex distributions. And its related algorithm, adaptive importance sampling [14], has been popular since its practicability was proved in the mid-2000s [15, 16, 17]. Especially the publication of the population Monte Carlo (PMC) sampling method [21] makes AIS back to the focus. PMC is a form of iterated sampling importance, which iterates to adapt importance functions. PMC has many extensions, such as D-kernel PMC [22], mixture PMC [23] and non-linear PMC (N-PMC)[24]. What's more, many AIS methods were motivated by PMC, including adaptive multiple importance sampling (AMIS)[25] and adaptive population importance sampling (APIS)[26]. We will develop an algorithm, which is similar to PMC algorithm, but with state-of-the-art resamplers.

1.2 Thesis Outline

In section 2, we first introduce the importance sampling and implement it in an example with different proposal distributions. We will demonstrate that the goodness of the sampling highly depends on our proposal distribution.

In section 3, we will introduce the framework of a specific adaptive importance sampling algorithm, the ensemble transport adaptive importance sampling (ETAIS). It consists of two parts: importance sampling and resampling algorithm. We will introduce two resampling algorithms, bootstrap resampler and multinomial transformation. Then we will implement ETAIS with different resamplers in a sampling example.

In section 4, we study the Metropolis–Hastings algorithm and find the Metropolis–Hastings algorithm cannot perform well in multi-modal distribution. But our ETAIS does not suffer from this problem.

In section 5, we present a numerical example. We use our algorithm to compute the posterior distribution arising from a Bayesian inverse problem. And we compare with the Metropolis–Hastings algorithm.

Section 6 is a brief conclusion and discussion.

Section 7 is a list of future directions.

1.3 Contributions of the Thesis

In this thesis, all ideas about ensemble transport adaptive importance sampling (ETAIS) and multinomial transformation (MT) were invented by Dr Simon Cotter.[3] I am guided by his paper and code myself to implement them in some examples. We will highlight valuable implementations in this thesis.

In section 2.3 and 2.4, we implement the importance sampling with shifted normal proposal distribution and uniform proposal distribution to sample a standard normal distribution.

In section 3.4 and 3.5, we implement the ETAIS algorithm with Bootstrap and multinomial transformation. In section 3.6, we compare the relative error between them and demonstrate that multinomial transformation is more accurate than the bootstrap resampler.

In section 5, we implement ETAIS in a Bayesian inverse problem from Birth and Death ODE. And we successfully get the posterior and ETAIS performs better than the block-wise Metropolis–Hastings algorithm in a small sample size.

2 Importance Sampling (IS)

2.1 Introduction to IS

Importance Sampling (IS) is a collection of Monte Carlo methods. The common Monte Carlo methods contain importance sampling, rejection sampling, the Metropolis Method and Gibbs sampling.[7]

Definition 2.1. The method of importance sampling is an evaluation of

$$\mathbb{E}_f[h(X)] = \int_{\mathcal{X}} h(x)f(x)dx$$

based on samples $X_1, ..., X_n$ generated from a given distribution ϕ and approximating

$$\mathbb{E}_f[h(X)] \approx \frac{1}{m}\sum_{j=1}^{m} \frac{f(X_j)}{\phi(X_j)} h(X_j).$$

This method is based on an alternative representation, which is

$$\mathbb{E}_f[h(X)] = \int_{\mathcal{X}} h(x)\frac{f(x)}{\phi(x)}\phi(x)dx.$$

[10]

Importance Sampling is easy to implement, but it doesn't work well when the dimension of parameters is high. If we want to sample the target distribution correctly, we need to establish our proposal distribution proportional to target distribution.[8] In Bayesian statistics, we usually sample the posterior distribution. To use IS in this problem, selecting a proper proposal distribution is of vital importance. So it requires us to obtain as much information as possible from posterior distribution, which is difficult as the dimension is high. However, for the low-dimensional problem, it is easy to find a suitable proposal distribution. Then using IS algorithm to plot a weight-histogram can simulate the target distribution nicely.

2.2 Algorithm implementation for IS[8]

Algorithm 1: Importance Sampling (IS)

```
for $i = 1, ..., N$ do
1   Sample $x_i$ from $\nu$
2   Calculate
        $$\omega_i = \frac{f(x_i)}{\phi(x_i)}.$$
3   Store $(x_i, \omega_i)$.
```

Algorithm 1 is our Importance Sampling. And its advantage is that it will work even we don't know the target density function $f(\cdot)$. But it is a bit challenging to choose our proposal density function, $\phi(\cdot)$, from distribution ν. In theory, we know that the optimal proposal distribution $\phi(\cdot)$ with minimal variance is ϕ proportional to f.[8] In reality, it is challenging to obtain ϕ by this criterion. One of the problems is the tails of ϕ. The tails of the proposal distribution influence the sampling a lot. We will discuss this problem by comparing two sampling examples in one dimension. Both examples will sample a standard normal distribution but with different proposal distributions.

2.3 IS example with normal proposal distribution

Algorithm 2: IS with normal proposal distribution

```
for $i = 1, ..., N$ do
1   Sample $x_i$ from N(3,1)
2   Calculate
        $$\omega_i = \frac{N(x_i|0,1)}{N(x_i|3,1)}.$$
3   Store $(x_i, \omega_i)$.
```

In this example, our target distribution is $N(0, 1)$, which is a standard normal distribution. And our proposal distribution is $N(3, 1)$, which has the same shape as $N(0, 1)$ but with a shifted position. We will use the location of x_i and weight ω_i to create weight-histogram and see whether our weight-histogram fulfil the target distribution.

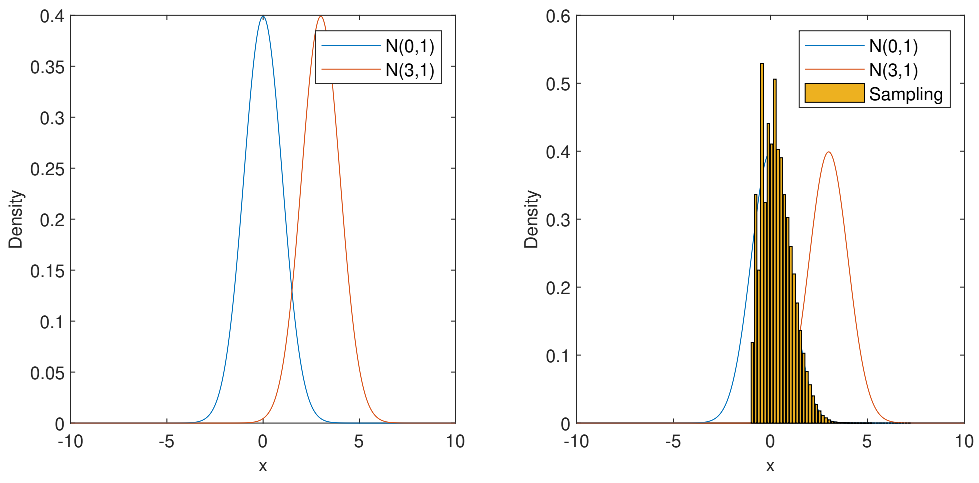


Figure 1: $N(3,1)$ proposal distribution

In Figure 1, we can clearly see that there is a tail to the left of the peak of $N(3,1)$ starting around 0, which will cause a problem to sample the target distribution. The IS result in the second graph shows that our proposal distribution doesn't work nicely, since the weighted histogram cannot cover the area of our standard normal distribution. Specifically, there isn't any sample lying in the left region of $N(0,1)$.

2.4 IS example with uniform proposal distribution

Algorithm 3: IS with uniform proposal distribution

```
for $i = 1, ..., N$ do
1   Sample $x_i$ from $Uni(-5, 5)$
2   Calculate
        $$\omega_i = \frac{N(x_i|0,1)}{0.1}.$$
3   Store $(x_i, \omega_i)$.
```

This time our proposal distribution is a uniform distribution $Unif(-5, 5)$, which is centered at 0. Thus our $\phi(x_i) = 0.1$ for every i. Again, after calculating (x_i, ω_i), we draw a weight-histogram to see whether it will sample the target distribution right.

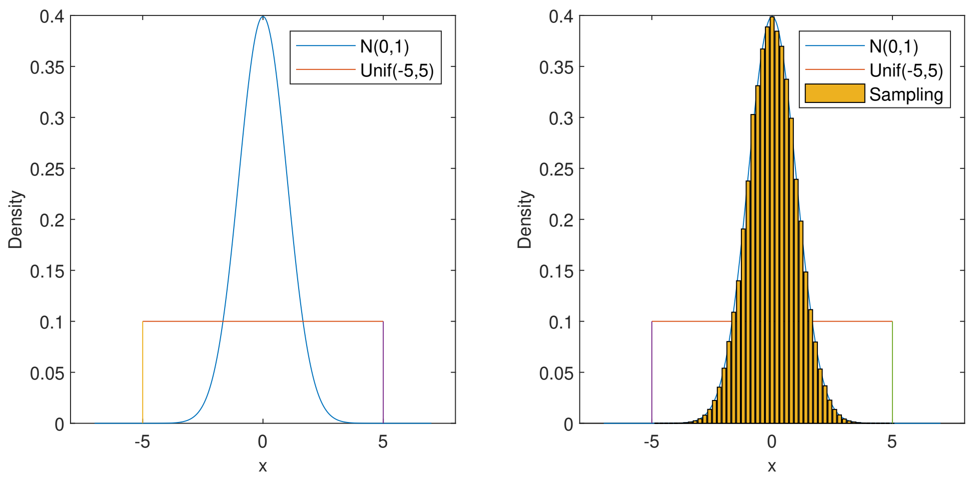


Figure 2: $Unif(-5, 5)$ proposal distribution

From Figure 2, we can find that the uniform distribution contains all the area of standard normal distribution, and the value of the pdf is 0.1 in $[-5, 5]$. In the second graph, IS works very well, and the weight-histogram gives us a real shape of standard normal distribution. This means that our sampling is successful.

2.5 Comparison & Discussion

In the previous sections, we have sampled the same target distribution but with different proposal distributions. One is shifted normal distribution, and the other is a uniform distribution. The results can be seen clearly from the figures. The shifted normal distribution failed, and the uniform distribution succeeded. What causes the problem is whether the region of sampling is appropriate to calculate the importance weights. In Figure 1, the shifted normal distribution cannot calculate true importance weights for negative values. The reason is that the value of distribution is too small and makes weights so large, which fails to sample correctly.

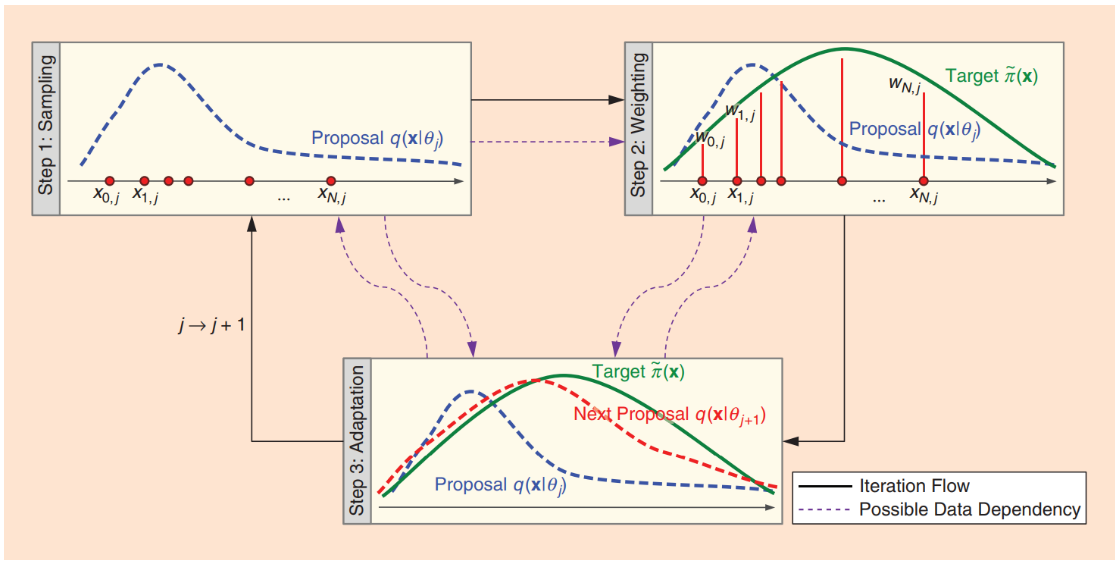
On the contrary, the uniform distribution works well at this point. Its values are all the same in the sampling area and output (Figure 2) is excellent to obtain a shape of standard normal distribution. The problem we discuss is the 'tails' issue of the proposal distribution, which sometimes causes difficulties in sampling if we didn't choose the appropriate proposal distribution.

The main challenge in IS is finding an appropriate proposal distribution. If it is not well chosen, it will produce many zeroes and have poor performance. Furthermore, if the dimension becomes higher, this challenge is much harder to solve. Some papers use adaptive algorithms to update parameters of a mixture density to optimise the performance of importance sampling.[18] Although there are many limitations in IS, we cannot deny that IS is an efficient way to sample from complex distributions.

3 Adaptive Importance Sampling (AIS)

3.1 Introduction to AIS

From the previous discussion, we found that the performance of IS algorithm largely depended on the selection of the proposal distributions. In order to get out of this, an adaptive iterative procedure can be constructed to improve the qualities of samples and accuracy of the inference. This lead to the concept of Adaptive Importance Sampling.[1] The advantage of AIS is that it has an update of proposal densities in a few iterations. And by the nicer proposal densities, it can approximate target distribution better. The generic AIS algorithm is a three-step procedure: generating samples from a proposal density (sampling), calculating the importance weights of each sample (weighting), and updating parameters of proposal densities for next iteration, which is illustrated by Figure 3.[1]



Step 1: Sampling
Proposal $q(\mathbf{x}|\theta_j)$
$x_{0,j}$ $x_{1,j}$ ... $x_{N,j}$
Step 2: Weighting
Target $\tilde{\pi}(\mathbf{x})$
$w_{N,j}$
$w_{1,j}$
$w_{0,j}$
Proposal $q(\mathbf{x}|\theta_j)$
$x_{0,j}$ $x_{1,j}$ ... $x_{N,j}$
$j \to j+1$
Step 3: Adaptation
Target $\tilde{\pi}(\mathbf{x})$
Next Proposal $q(\mathbf{x}|\theta_{j+1})$
Proposal $q(\mathbf{x}|\theta_j)$
Iteration Flow
Possible Data Dependency

Figure 3: The diagram from Bugallo's paper[1] clearly explains the three steps of the AIS algorithm.

For the generic AIS algorithm, the first thing is initializing a set of M proposal distributions $\{\nu_n(\mathbf{y}|\mathbf{x}_{n,1})\}_{n=1}^{M}$, each proposal with parameter $\mathbf{x}_{n,1}$. Then we can get samples, $\mathbf{y}_{n,1}^{(k)}$ where $n = 1, ...M$, $k = 1, ..., K$, from $\mathbf{x}_{n,1}$ and where K denotes the sample size and 1 is the first iteration time. After sampling, we can calculate the importance of them in the same way as the importance sampling, which can approximate the target distribution. Last, we update our parameter from $\mathbf{x}_{n,1}$ to $\mathbf{x}_{n,2}$. After M iterations, we output the pairs $\left\{w_{n,i}^{(k)}, \mathbf{y}_{n,i}^{(k)}\right\}$ to simulate the target distribution. The procedure can be seen in Algorithm 4.

```
Algorithm 4: The generic AIS algorithm[1]
Initialization
Choose $K, M, N, \{\mathbf{x}_{n,1}\}_{n=1}^{M}$.
for $i = 1, ..., N$ do
    1) Sampling
    Draw $K$ samples from $\{\nu_{n,i}(\mathbf{x}_{n,i})\}_{n=1}^{M}$ and get samples $\mathbf{y}_{n,i}^{(k)}$, where $k = 1, ..., K$,
     $n = 1, ..., M$.
    2) Weighting
    Compute $w_{n,i}^{(k)}$, for all $KM$ samples.
    3) Updating
    Update
        $$\{\mathbf{x}_{n,i}\}_{n=1}^{M} \to \{\mathbf{x}_{n,i+1}\}_{n=1}^{M}$$
     .
Outputs
Return the pairs $\left\{w_{n,i}^{(k)}, \mathbf{y}_{n,i}^{(k)}\right\}$ for all $k = 1, ..., K$, $n = 1, ..., M$, $i = 1, ..., N$.
```

3.2 Preliminaries about Particle Filter and Resamplers

Particle filters are a set of Monte Carlo algorithms that are used to address filtering problems. The filtering problem is concerned with estimating the true state of a system when noisy observations are given to the dynamical system. It is also called "Sequential Monte Carlo". In recent years, some efficient particle filters are emerged, such as the ensemble Kalman filter (EnKF)[6] and Sebastian Reich's ensemble transport particle filter (ETPF)[9].

The ensemble transform particle filter (ETPF) invented by Reich [9] combines the optimal coupling ideas. Here we use importance weights to replace weights in the filter problems. Then we can resample weighted samples into new equally weighted samples.

$$\sum_{i=1}^{M} w_i \delta_{y_i}(\cdot) \xrightarrow{ETPF} \sum_{i=1}^{M} \delta_{x_i}(\cdot).$$

3.3 The ensemble transport adaptive importance sampler

In this section, we discuss a specific Adaptive Importance Sampling algorithm, which is invented by Dr Simon Cotter[3]. The algorithm is called the ensemble transport adaptive importance sampler (ETAIS). IS is a very nice method when proposal distribution's centered area is similar to the target distribution. And the proposal is optimal if it is the target distribution. The ETAIS aims to utilise ensembles of states to update proposal densities and make them approach target distributions as much as possible.

In this algorithm, we need to consider two things: one is the proposal distribution, and the other is the resampler algorithm. As for the proposal distribution, we choose a mixture of distributions which is from a sum of MCMC proposal kernels. We will use the random walk Metropolis-Hastings (RWMH) proposal density. After selecting the proposal densities, we can get an ensemble from it and calculate their importance weights. At last, we need to use the resampling algorithm to get evenly weighted samples which best represent the samples.

For the resampler algorithm, the choice of the resampler algorithm does influence the goodness of our proposal distribution. For a simpler resampler, we may not get ideal proposal distribution. And it will also result in high variances of weights, which leads to slow convergence to the target distribution. In Simon's paper, he first implemented the ETPF resampler algorithm, but it was a bit costly if the number of ensembles grows. We will discuss bootstrap resampler and multinomial transformation, which are cheap resamplers.[3]

Algorithm 5: The ensemble transport adaptive importance sampling (ETAIS) [3]

```
1 Initial states $\mathbf{X}^{(0)} = \mathbf{X}_0 = [x_1^{(0)}, x_2^{(0)}, ..., x_M^{(0)}]^\top$.
2 for $i = 1, ..., N$ do
3     Sample $\mathbf{Y}^i = [y_1^{(i)}, y_2^{(i)}, ..., y_M^{(i)}]^\top$, where $y_j^{(i)} \sim \nu(\cdot; x_j^{(i-1)})$.
4     Calculate $\mathbf{W}^{(i)} = [w_1^{(i)}, w_2^{(i)}, ..., w_M^{(i)}]^\top$, $w_j^{(i)} = \frac{\pi(y_j^{(i)})}{\chi(y_j^{(i)}; \mathbf{X}^{(i-1)})}$, where
          $$\chi(\cdot; \mathbf{X}^{(i-1)}) = \frac{1}{M} \sum_{j=1}^{M} \nu(\cdot; x_j^{(i-1)}).$$
5     Use resampling algorithm: $(\mathbf{W}^{(i)}, \mathbf{Y}^{(i)}) \to (\frac{1}{M}\mathbf{1}, \mathbf{X}^{(i)})$.
6 Store $(\mathbf{W}, \mathbf{Y})$.
```

In Algorithm 5, the states $x \in X$ is from the posterior density π. Then we can show the current state of the Markov chains with $\mathbf{X} = [x_1, x_2, ..., x_M]^\top$. For M ensemble members. And for the transition kernel ν, we use the RWMH proposal density $\nu(\cdot|x) \sim N(x, \beta^2)$, where β^2 is the variance. We define the ETAIS-RW algorithm, as the ETAIS algorithm with the RWMH proposal density.

3.4 Bootstrap resampler

3.4.1 Explanation

Bootstrap techniques are a collection of computationally intensive methods that are based on resampling from the observed data [10]. They were first introduced by Efron(1979a)[5]. Bootstrap essentially replaces the input via the empirical cumulative distribution function of the sample.

Algorithm 6: The Bootstrap resampler[11]

```
1 Obtain sample $\{x_i\}_{i=1}^n$ from some distributions.
2 Construct
      $$\hat{F} = \sum_{i=1}^{n} n_i \delta_{x_i}(\cdot),\ n_i = \sum_{n}^{k=1} I(x_i = x_k).$$
3 Sample with replacement $X^* = [x_1^*, x_2^*, ..., x_n^*]^\top \sim \hat{F}$. i.e. $X^* \sim Multi(1, \frac{1}{n}[n_1, ..., n_n]^\top)$.
  ($\delta_x(\cdot)$ is the Dirac delta measure at state $x$).
```

3.4.2 Implementation with a Normal target distribution

For the resampler algorithm in AIS, we obtain a weighted sample $\{(w_i, x_i)\}_{i=1}^n$. And we can generalise the Bootstrap resampler for our algorithm ETAIS.

Algorithm 7: The Bootstrap resampler for ETAIS[11]

```
1 Obtain sample $\{(w_i, x_i)\}_{i=1}^n$ from IS step.
2 Construct
      $$\hat{F} = \sum_{i=1}^{n} \tilde{w}_i \delta_{x_i}(\cdot),\ \tilde{w}_i = \sum_{n}^{k=1} w_i I(x_i = x_k).$$
3 Sample with replacement $X^* = [x_1^*, x_2^*, ..., x_n^*]^\top \sim \hat{F}$. i.e.
    $X^* \sim Multi(1, \bar{\mathbf{w}} = \frac{1}{\sum w_i}[\tilde{w}_1, \tilde{w}_2, ..., \tilde{w}_n]^\top)$.
  ($\delta_x(\cdot)$ is the Dirac delta measure at state $x$).
```

When implementing the ETAIS, we need to store each iteration's histogram height and use the average of all the height to get better convergence of the sampling. In addition, we use the RWMH proposal density $\nu \sim N(x, \beta^2)$.

Algorithm 8: Implementation version of ETAIS with Bootstrap resampler

```
1 Initial states $\mathbf{X}^{(0)} = \mathbf{X}_0 = [x_1^{(0)}, x_2^{(0)}, ..., x_M^{(0)}]^\top$.
2 for $i = 1, ..., N$ do
3     Sample $\mathbf{Y}^i = [y_1^{(i)}, y_2^{(i)}, ..., y_M^{(i)}]^\top$, where $y_j^{(i)} \sim \nu(\cdot; x_j^{(i-1)})$.
4     Calculate $\mathbf{W}^{(i)} = [w_1^{(i)}, w_2^{(i)}, ..., w_M^{(i)}]^\top$, $w_j^{(i)} = \frac{\pi(y_j^{(i)})}{\chi(y_j^{(i)};\mathbf{X}^{(i-1)})}$, where
          $$\chi(\cdot; \mathbf{X}^{(i-1)}) = \frac{1}{M}\sum_{j=1}^{M} \nu(\cdot; x_j^{(i-1)}).$$
5     Use $(\mathbf{W}^{(i)}, \mathbf{Y}^{(i)})$ to form a histogram and store heights of bins $\mathbf{H}_i$.
6     Use Bootstrap resampler in Algorithm 7:
          $$(\mathbf{W}^{(i)}, \mathbf{Y}^{(i)}) \to (\frac{1}{M}\mathbf{1}, \mathbf{X}^{(i)})$$
      .
7 Store $(\mathbf{W}, \mathbf{Y})$ and $\{\mathbf{H}_i\}_{i=1}^n$.
```

Here we set our target distribution as $N(2, 4)$, the number of samples $M = 200$ and iteration times $N = 2000$.

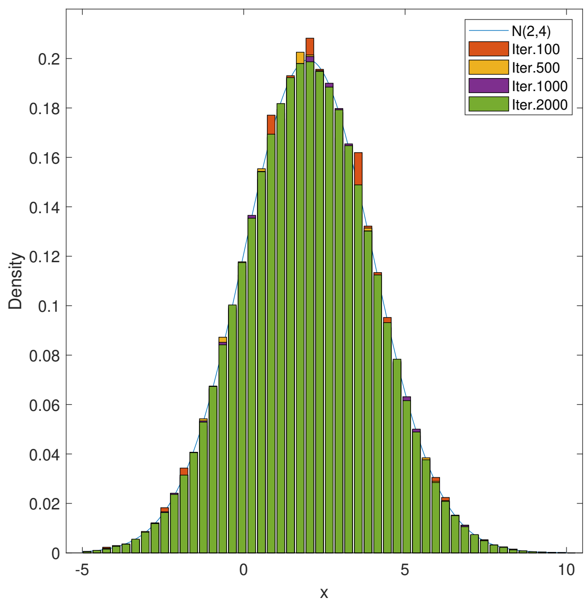


Figure 4: The weight-histogram with iterations changing by ETAIS

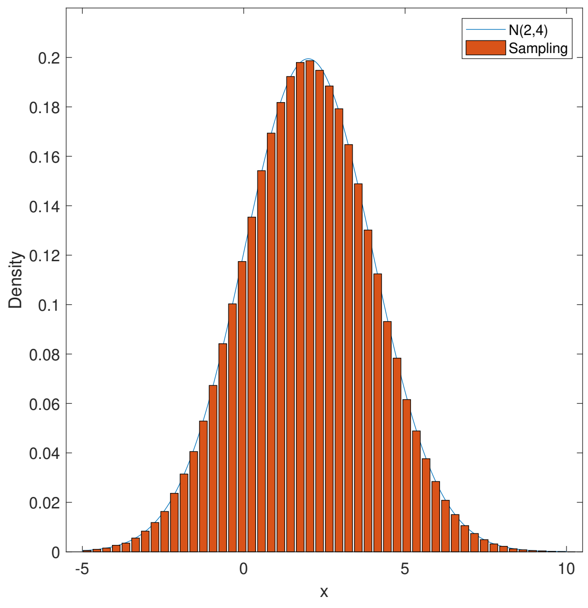


Figure 5: The final weight-histogram by ETAIS

From Figure 4 and Figure 5, it can be seen that our algorithm is implemented in the right way. Typically, in Figure 4, the weight-histogram becomes better and better as the iteration number grows. Finally, the weight-histogram successfully covers the shape of $N(2, 4)$ distribution.

3.5 Multinomial transformation

3.5.1 Explanation

It is true that the ETPF produces the optimal linear coupling, yet it becomes computationally more intensive as the ensemble size increases[3]. However, we can reduce the cost by sacrificing accuracy within an acceptable range. The Multinomial transformation (MT) is a greedy approximation of the ETPF resampler. This algorithm was invented by Dr Simon and designed to be less computational than ETPF while gaining more accuracy compared to Bootstrap resampler. In the Bootstrap resampler, we need to sample M times from an M-dimensional multinomial distribution, and each of them is different. And the vector of probabilities is

$$\frac{1}{\sum w_n}[w_1, w_2, ..., w_M] = \tilde{\mathbf{w}},$$

which corresponds to state y_n. But in the MT We will replace w_i with a vector $\mathbf{p}_i$. Then

$$\frac{1}{M}\sum \mathbf{p}_i = \tilde{\mathbf{w}},$$

where $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_M\} \subset R^M_{\geq 0}$. The samples in the MT are from each multinomial defined by vectors $\mathbf{p}_i = [p_{i,1}, p_{i,2}, ..., p_{i,M}]$ with associated states y_i. And a deterministic sample will be selected such that every sample attains the same value as the mean value of every multinomial distribution, which is similar to ETPF. The new sample $\hat{x}_i$ is

$$\hat{x}_i = \sum p_{i,j} x_j,\ i \in \{1, 2, ..., M\}\,.$$

The way we split the original multinomial is from the notion of optimal transport. We locate the largest weights and form a cluster around these points by the closest states.

The procedure of the MT is in Algorithm 9 and described with deterministic sampling, which uses the means of each of the submultinomials as the new samples.

Algorithm 9: The multinomial transformation (MT) [2]

```
1  $\mathbf{z} = M\bar{w}$.
2  for $i = 1, ..., M$ do
3      $J = \text{argmin}_j z_j$.
4      $p_i, J = \min\{1, z_J\}$.
5      $z_J = z_J - p_{i,J}$.
6      while $\sum_j p_{i,j} < 1$ do
7          $K = \text{argmin}_{k \in \{k|z_k>0\}} \|y_J - y_k\|$.
8          $p_{i,K} = \min\left\{1 - \sum_j p_{i,j},\, z_K\right\}$.
9          $z_K = z_K - p_{i,K}$.
10     $x_i = \sum_k p_{i,k} y_k$.
```

The reason why we want to find another resampler is that the Bootstrap algorithm is random. It may happen that some ensemble members will not be sampled in a specific region. What's more, the Bootstrap cannot retain the weighted sample mean. But for the MT resampler, every sample will be selected from each section and will almost avoid losing any mode in the density. And since the algorithm assigns the mean of the multinomial to the sample, it is much better than the random sample and maintains the mean. In Simon's paper[3], the MT algorithm is demonstrated less computationally expensive than the ETPF.

3.5.2 Implementation with a Normal target distribution

This time we still use the implementation version of ETAIS but with the MT resampler algorithm. Our target distribution is $N(2, 4)$ and we set parameters number of samples $M = 200$ and iteration times $N = 2000$.

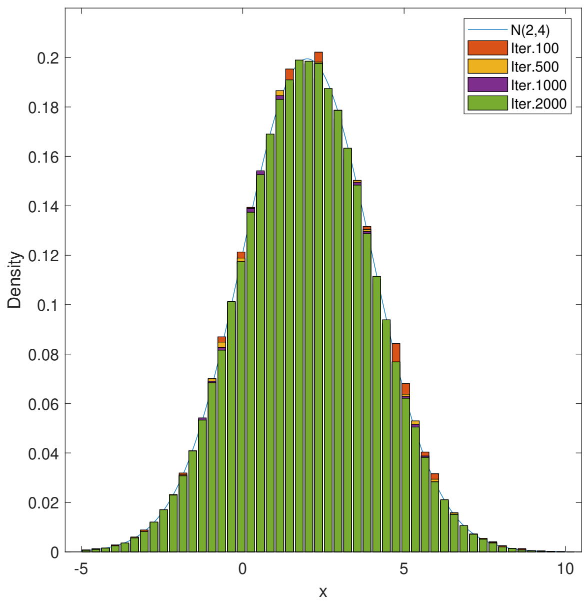


Figure 6: The weight-histogram with iterations changing by ETAIS

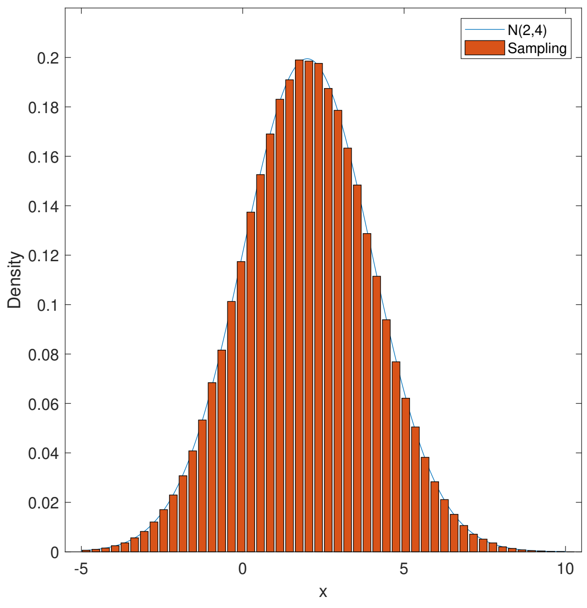


Figure 7: The final weight-histogram by ETAIS

From Figure 6 and 7, we know that the MT algorithm is successfully implemented. And the shape of $N(2,4)$ can be seen clearly from the final weight-histogram.

3.6 Comparison

We have seen from the previous section that both Bootstrap and MT resampler works correctly in the sampling example. Now we want to compare the relative error between them. We choose the range of ensemble size M from 50 to 200. The relative error will be computed as follows.

$$\epsilon_i = \frac{|\upsilon_i - \upsilon_{ith\ bin}|}{|\upsilon_i|},\ i = 1, ..., M,$$

where υ_i is the value of the target density at ith bin's location, and $\upsilon_{ith\ bin}$ is the height of ith bin.

$$E = \sum_{i=1}^{M} \epsilon_i$$

We will compare $E_{Bootstrap}$ and E_{MT}.

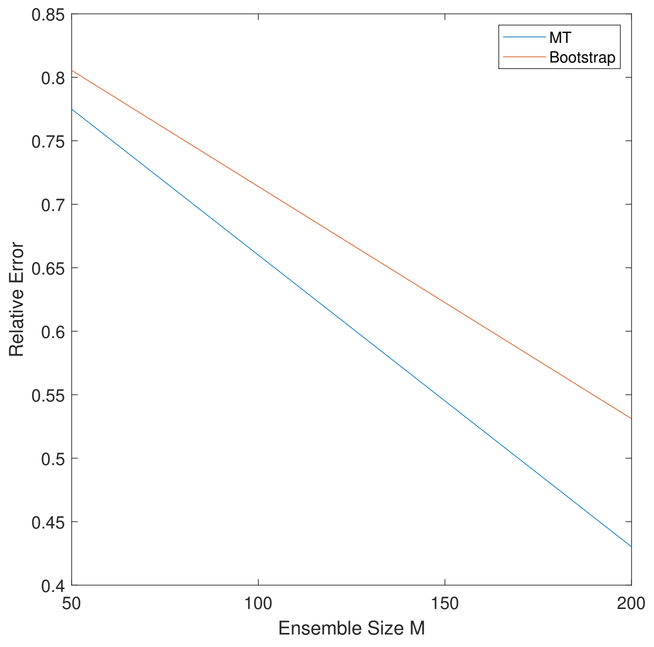


Figure 8: Relative error comparison

From Figure 8, we conclude that the MT resampler behaves more accurate than Bootstrap resampler. And in Simon's paper[3], he proves that it is less accurate but much faster than ETPF.

4 Comparing ETAIS-RW with the Metropolis–Hastings (MH) algorithm in a Gaussian mixture model example

4.1 Introduction to the MH algorithm

Importance Sampling gives us independent samples from the posterior distribution. The Metropolis–Hastings (MH) algorithm provides a dependent sample. It is much quicker to implement and can extend to high-dimensional spaces easily even we don't know where the probability of distribution is high. But its disadvantage is obvious, it struggles badly with multi-modal distributions.

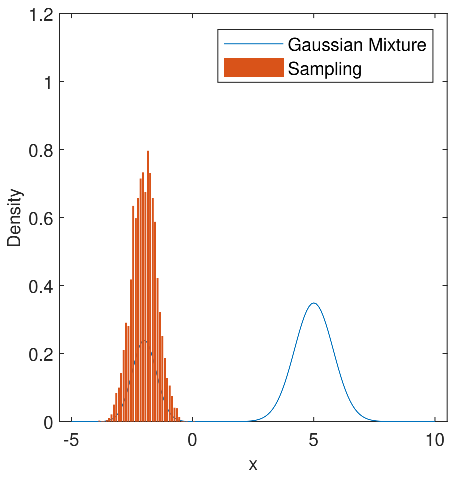
Algorithm 10: The Metropolis-Hastings Algorithm [11]

```
1 Initialise $x_0$.
2 for $i = 1, ..., N$ do
3     Sample $x^* \sim q(\cdot|x_{i-1})$ and $u \sim U([0,1])$.
4     Calculate
          $\alpha(x_{i-1}, x^*) = \max(1, \frac{p(x^*)q(x_{i-1}; x^*)}{q(x^*; x_{i-1})p(x_{i-1})})$.
5     if $u < \alpha(x_{i-1}, x^*)$ then
          $x_i = x^*$
      else
          $x_i = x_{i-1}$
6     Store $x_i$, $i = 0, 1, ..., N$
```

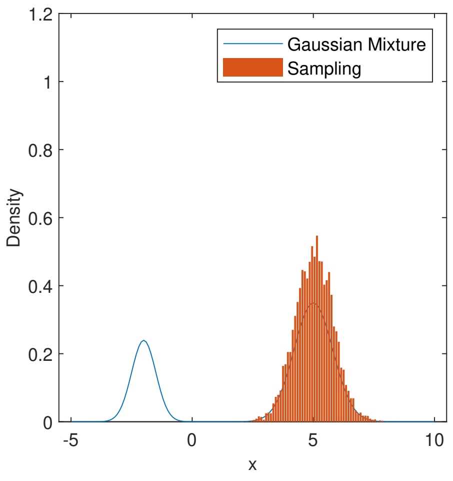
The Algorithm 10 outlines the procedure of the MH algorithm. And we should define the proposal density $q(\cdot)$ and sample from it. After storing $x_0, x_1, ..., x_N$, the empirical distribution of saved states will give a shape of sampling distribution $p(\cdot)$.

4.2 implementation with Gaussian mixture model

In this section, we will implement the MH algorithm with Gaussian proposal density to sample a Gaussian mixture distribution $\pi \sim 0.3N(-2, 0.5) + 0.7N(5, 0.8)$.



(a) With initial point $x_0 = 0$



(b) With initial point $x_0 = 5$

Figure 9: The MH Algorithm sampling outputs with $N = 1e4$

It can be seen clearly from the Figure 9 that the MH algorithm cannot sample multi-modal efficiently even it is one-dimensional distribution.

4.3 Comparison with ETAIS

We know that the MH algorithm will perform poorly in a multi-modal. But for our ETAIS algorithm, we will not suffer from this problem.

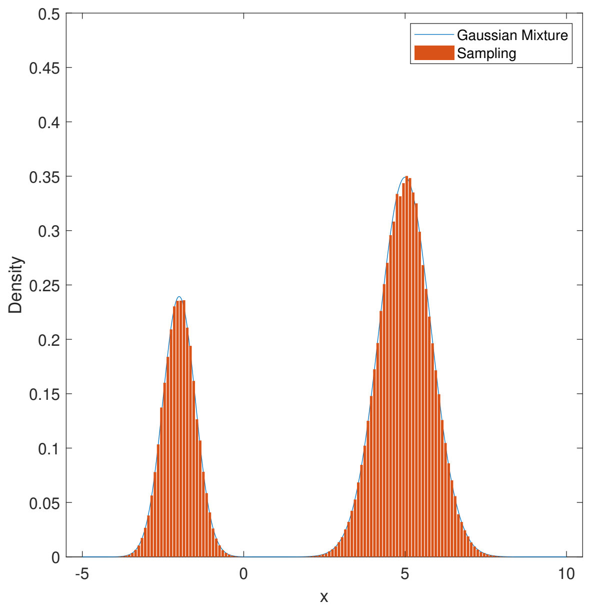


Figure 10: Sampling using ETAIS-RW with $M = 200$ ensembles

5 Data assimilation with Birth & Death ODE

5.1 Preliminary: Bayesian Inverse Problem [3]

We will focus on characterizing posterior distributions emerging from Bayesian Inverse problems. First, noisy observations will be made on an unknown quantity $x \in X$, where X is a Hilbert space. The observations $D \in \mathbb{R}^d$ are assumed to be subject to Gaussian noise. And $\mathcal{G} : X \to \mathbb{R}^d$ is the observation operator, which will map x into observation space as follows.

$$D = \mathcal{G}(x) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \Sigma) \tag{1}$$

Then we form the likelihood of the observed data D given $x = x^*$. Rearranging (1), we obtain

$$\mathbb{P}(D|x = x^*) \propto \exp\left(-\frac{1}{2}\left\|\mathcal{G}(x^*) - D\right\|_\Sigma^2\right) = \exp(\Phi(x^*)) \tag{2}$$

, where $\|x_1 - x_2\|_\Sigma^2 = (x_1 - x_2)^\top \Sigma^{-1}(x_1 - x_2)$ for $x_1, x_2 \in \mathbb{R}^d$.

By Bayes' theorem, the posterior density will be given by,

$$\pi \propto \exp(-\Phi(x^*))\pi_0(x^*),$$

where π_0 is the prior density of x^*.

5.2 Birth & Death ODE

In this section, we are going to investigate a specific type of Birth & Death ODE. The population model is

$$\frac{dX}{dt} = K_b - K_d X \qquad (3),$$

where K_b is "birth rate" of the population and K_d is the "death rate". Both K_b and K_d are constant here.

The real solution of this ODE is

$$X = c\exp(-K_d t) + \frac{K_b}{K_d} \tag{4}$$

where c is a constant.

5.3 Target distribution

For this example, we get observations of the position of a particle with noise in an equally spaced time interval $t \in (0, 1]$. And the time step is $h = 1 \times 10^{-2}$.

Now the observation operator is

$$\mathcal{G}(K_b, K_d) = [X(t_1), X(t_2), ..., X(t_n)]^\top$$

where $t_1 = 0.01$ and $t_n = 1$, x is in the solution of ODE. We choose the parameters $K_b^* = K_d^* = 3$ and assume they are independent.

Our observation Y is

$$Y = G(K_b^*, K_d^*) + \eta, \;\; \eta \sim \mathcal{N}(0, I_n).$$

Priors for the parameters were given by

$$K_b, K_d \sim \text{Lognormal}(3, 1^2).$$

The posterior density function takes the form

$$\pi(K_b, K_d|Y) \propto \exp\left\{-\frac{1}{2}\left\|\mathcal{G}(K_b, K_d) - Y\right\|^2\right\}\pi_0(K_b)\pi_0(K_d).$$

5.4 Implementation

For this example, what we want to demonstrate is the correlation between the parameter K_b and K_d. From the analytic solution of ODE in (4), the term $\frac{K_b}{K_d}$ implies that K_b and K_d should have correlation 1.

For the implementation, we use ETAIS-RW with an ensemble size of $M = 5000$ and the MT resampling algorithm. And the iteration time is $N = 50$.

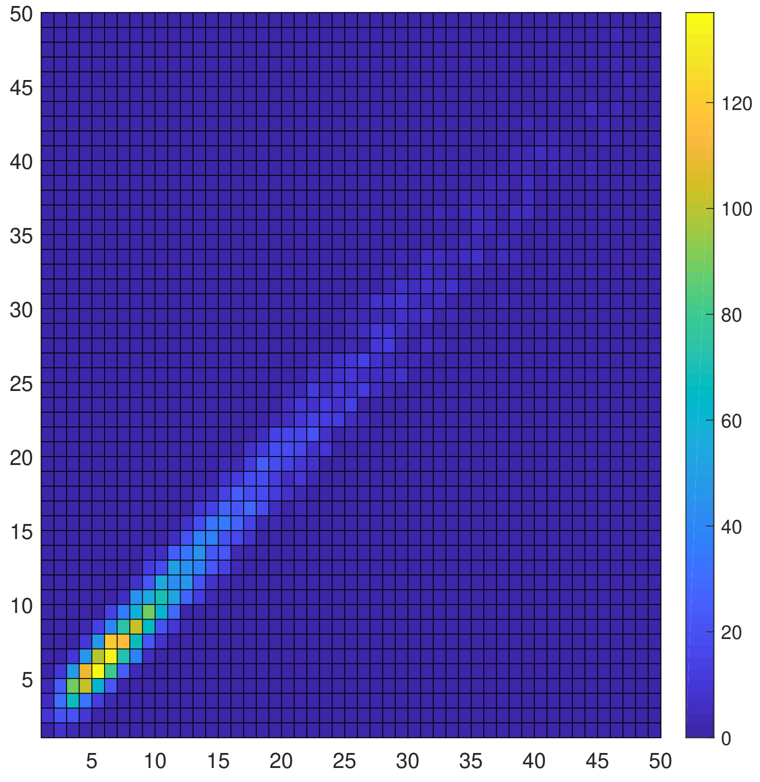


Figure 11: Posterior for K_b and K_d by ETAIS-RW

The Figure 11 shows that relation between the posterior of K_b and K_d is a shape of a narrow ellipsoid. It means that K_b and K_d are strongly correlated as expected.

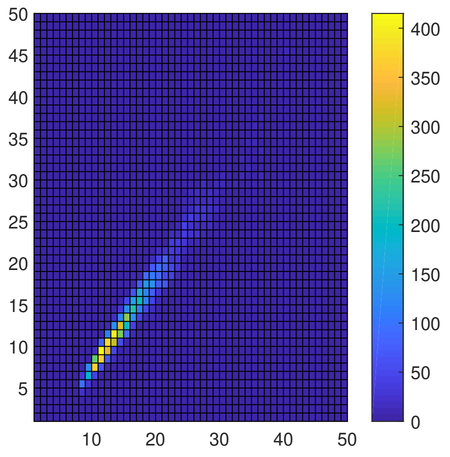
5.5 Comparison with the MH algorithm

Since we are going to perform multidimensional sampling by MH algorithm. The first method comes to my mind is block-wise updates. It means that the dimension of the proposal density $q(\mathbf{x})$ is the same as the target density $p(\mathbf{x})$.

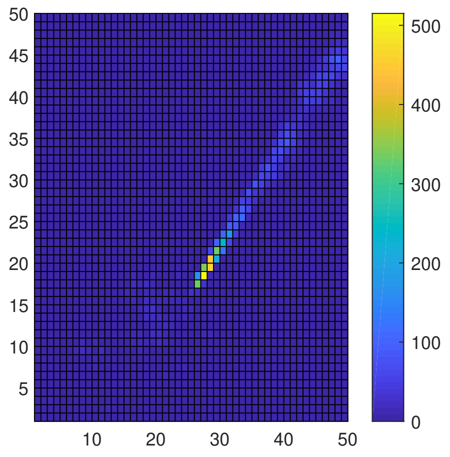
Algorithm 11: The block-wise Metropolis-Hastings Algorithm[4]

```
1 Initialise $\mathbf{x}_0$.
2 for $i = 1, ..., N$ do
3     Sample $\mathbf{x}^* \sim q(\cdot|\mathbf{x}_{i-1})$ and $u \sim U([0,1])$.
4     Calculate
          $$\alpha(\mathbf{x}_{i-1}, \mathbf{x}^*) = \max(1, \frac{p(\mathbf{x}^*)q(\mathbf{x}_{i-1}; \mathbf{x}^*)}{q(\mathbf{x}^*; \mathbf{x}_{i-1})p(\mathbf{x}_{i-1})}.$$
5     if $u < \alpha(\mathbf{x}_{i-1}, \mathbf{x}^*)$ then
          $\mathbf{x}_i = \mathbf{x}^*$
      else
          $\mathbf{x}_i = \mathbf{x}_{i-1}$
6     Store $\mathbf{x}_i$, $i = 0, 1, ..., N$
```

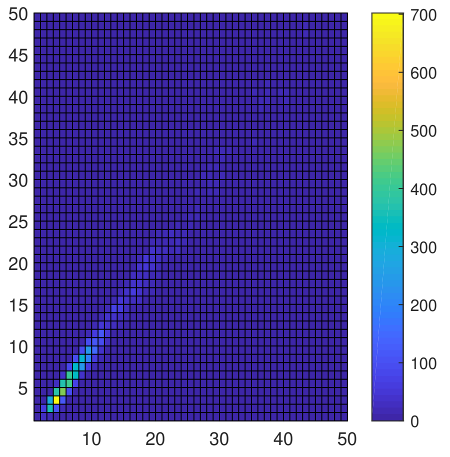
Here we choose the multivariate normal distribution as the proposal density distribution $q(\mathbf{x})$. First, we implement with the sample size $N = 1 \times 10^4$.



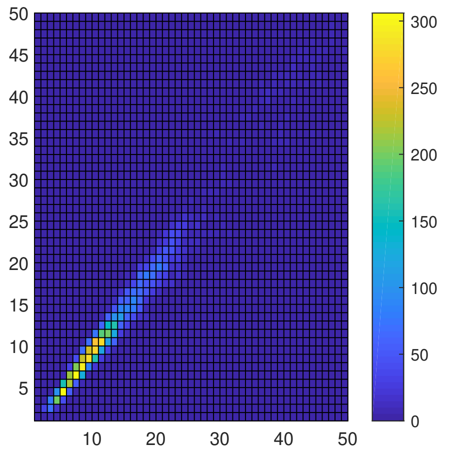
(a)



(b)



(c)



(d)

Figure 12: Posterior for K_b and K_d by block-wise MH with 1×10^4 samples

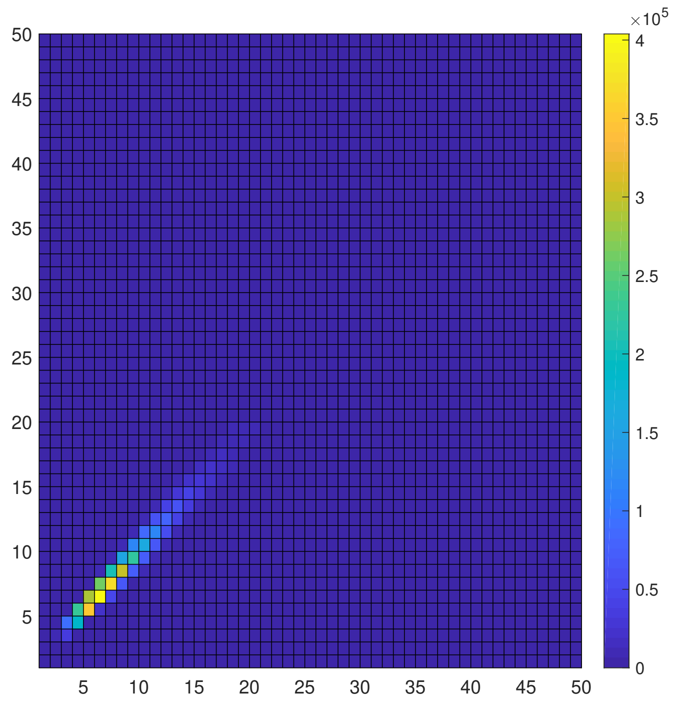


Figure 13: Posterior for K_b and K_d by block-wise MH with 5×10^6 samples

From Figure 12, it can be seen that some samplings are not converged well, and some are shifted. Only the last one looks nice. And we cannot tell the correlation between K_b and K_d is 1 from some outputs. So we increase the number of samples to $N = 5 \times 10^6$. From Figure 13, it achieves good convergence that can be reproduced and shows the correlation as expected.

For this specific example, the difference between ETAIS and MH algorithm is that ETAIS only needs a small number of samples to converge. But the cost per sample is a bit higher than MH algorithm. From Simon's paper[3], he implemented ETAIS with Lorenz's 1963 atmospheric convection equations and compared with MH algorithm. This time ETAIS got a significantly better convergence with ETAIS and then demonstrated the benefits of ETAIS for problems with costly likelihoods and sophisticated posteriors.

6 Discussions & Conclusions

We first explore importance sampling and see how different proposal distributions influence the accuracy of importance sampling. We demonstrate that importance sampling is a nice sampling technique but it must be equipped with appropriate proposal distributions.

Then we study adaptive importance sampling and implement a novel one, ETAIS. For the resampling algorithm, we compare bootstrap and multinomial transformation. We demonstrate that the multinomial transformation outperforms bootstrap.

Our ETAIS has two outstanding advantages [3]:
(1)It can redistribute the ensemble members to areas where further explorations are needed, which makes the algorithm sample successfully with a multi-modal distribution.
(2)It can be equipped with any MCMC proposals. Nowadays, more and more complicated MCMC algorithms have been developed. All of them could be incorporated into ETAIS's framework.

So in the thesis, we compare ETAIS with MH algorithm in a Gaussian mixture target. And we demonstrate that ETAIS can nicely sample from our multi-modal distribution. However, MH algorithm cannot perform well in multi-modal distribution. For the proposal we used in ETAIS, we only try the random walk Metropolis-Hastings (RWMH) proposal density. If we have time, we can investigate more useful MCMC proposal to make our algorithm more efficient.

Finally, we demonstrate that ETAIS with the multinomial transformation, which is a greedy approximation of ETPF, performs well in our numerical example. Comparing to the standard MH algorithm, ETAIS can achieve the convergence in a small number of samples.

7 Future directions

1. We have proved that the multinomial transformation (MT) is a nice resampling algorithm. It is a greedy approximation of the ETPF resampler and is less accurate than ETPF but much faster. So we will continue exploring detailed features of MT. We will first learn about the theory of the ETPF resampler and then compare ETPF with our MT resampler. The comparison will include the accuracy and the cost. Ideally, we would be able to find MT resampler costs much less than ETPF but is less accurate.

2. In this thesis, we only use the random walk (RW) proposal. We will investigate more complicated proposal distributions, like the Metropolis-adjusted Langevin algorithm (MALA), preconditioned Crank–Nicolson (pCN) proposal, preconditioned Crank-Nicolson Langevin (pCNL) and Hamiltonian Monte Carlo (HMC) proposals.

3. We have compared ETAIS with MH algorithm in a one-dimensional Gaussian mixture model. We can further implement a two-dimensional Gaussian mixture model and compare relative error between ETAIS and MH algorithm.

4. We implemented a simple Bayesian inverse problem in this thesis. If we have time, we will apply ETAIS algorithm to more realistic Bayesian inverse problems and compare relative error with other MCMC methods.

References

[1] M. F. Bugallo and V. Elvira, "Adaptive Importance Sampling: The past, the present, and the future," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 60–79, Jul. 2017.

[2] C. Cotter, S. Cotter, and P. Russell, "Parallel Adaptive Importance Sampling " ISSN 1749-9097. 2015.

[3] C. Cotter, S. Cotter, and P. Russell, "Ensemble Transport Adaptive Importance Sampling," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 7, no. 2, pp. 444–471, Jan. 2019.

[4] Dustinstansbury, "MCMC: Multivariate Distributions, Block-wise, Component-wise Updates," *The Clever Machine*, 05-Nov-2012. [Online]. Available: https://theclevermachine.wordpress.com/2012/11/04/mcmc-multivariate-distributions-block-wise-component-wise-updates/. [Accessed: 02-Jan-2020].

[5] B. Efron, "Bootstrap Methods: Another Look at the Jackknife," *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, Jan. 1979.

[6] G. Evensen, "Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics," *Journal of Geophysical Research*, vol. 99, no. C5, p. 10143, 1994.

[7] D. MacKay, *Introduction to Monte Carlo Methods.* Springer, 1998, pp. 175–204.

[8] A. Owen and Y. Zhou, "Safe and Effective Importance Sampling," *Journal of the American Statistical Association*, vol. 95, no. 449, pp. 135–143, Mar. 2000.

[9] S. Reich, "A Nonparametric Ensemble Transform Method for Bayesian Inference," *SIAM Journal on Scientific Computing*, vol. 35, no. 4, pp. A2013–A2024, Jan. 2013.

[10] C. P. Robert and G. Casella, *Monte Carlo statistical methods.* New York: Springer, 2010, p. 82.

[11] P. Russell, "Parallel MCMC Methods and their applications in Inverse Problems," 2017.

[12] W. K. Hastings, "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika*, vol. 57, no. 1, p. 97, Apr. 1970.

[13] T. House, A. Ford, S. Lan, S. Bilson, E. Buckingham-Jeffery, and M. Girolami, "Bayesian uncertainty quantification for transmissibility of influenza, norovirus and Ebola using information geometry," *Journal of The Royal Society Interface*, vol. 13, no. 121, p. 20160279, Aug. 2016.

[14] J. S. Liu, *Monte Carlo strategies in scientific computin*g. New York: Springer, 2008.

[15] M. C. A. M. Bink, M. P. Boer, C. J. F. ter Braak, J. Jansen, R. E. Voorrips, and W. E. van de Weg, "Bayesian analysis of complex traits in pedigreed plant populations," *Euphytica*, vol. 161, no. 1–2, pp. 85–96, Aug. 2007.

[16] G. Celeux, J.-M. Marin, and C. P. Robert, "Iterated importance sampling in missing data problems," *Computational Statistics Data Analysis*, vol. 50, no. 12, pp. 3386–3404, Aug. 2006.

[17] A. Blake, "CONDENSATION-Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[18] O. Cappé, R. Douc, A. Guillin, J.-M. Marin, and C. P. Robert, "Adaptive importance sampling in general mixture classes," *Statistics and Computing*, vol. 18, no. 4, pp. 447–459, Apr. 2008.

[19] J. C. Sexton and D. H. Weingarten, "Hamiltonian evolution for the hybrid Monte Carlo algorithm," *Nuclear Physics B*, vol. 380, no. 3, pp. 665–677, Aug. 1992.

[20] M. Girolami and B. Calderhead, "Riemann manifold Langevin and Hamiltonian Monte Carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, no. 2, pp. 123–214, Mar. 2011.

[21] O. Cappé, A. Guillin, J.-M. Marin, and C. P. Robert, "Population Monte Carlo," *J. Computational Graphical Statist.*, vol. 13, no.4, pp.907–929, 2004.

[22] G. R. Douc, J.-M. Marin, and C. Robert, "Convergence of adaptive mixtures of importance sampling schemes," *Ann. Statist.*, vol. 35, pp. 420–448, 2007.

[23] O. Cappé, R. Douc, A. Guillin, J. M. Marin, and C. P. Robert, "Adaptive importance sampling in general mixture classes," *Statist. Computing*, vol. 18, pp. 447–459, 2008.

[24] E. Koblents and J. Míguez, "A population Monte Carlo scheme with transformed weights and its application to stochastic kinetic models," *Statist. Computing*, vol. 25, no. 2, pp. 407–425, 2015.

[25] J. M. Cornuet, J.-M. Marin, A. Mira, and C. P. Robert, "Adaptive multiple importance sampling," *Scandinavian J. Statist.*, vol. 39, no. 4, pp. 798–812, Dec. 2012.

[26] L. Martino, V. Elvira, D. Luengo, and J. Corander, "An adaptive population importance sampler: Learning from the uncertainty," *IEEE Trans. Signal Process.*, vol. 63, no. 16, pp. 4422–4437, 2015.

A Matlab codes

A.1 Codes for weight-histogram function

```

% Peter Kovesi
% Centre for Exploration Targeting
% The University of Western Australia
% peter.kovesi at uwa edu au
%
% November 2010

function [h,edges] = weightedhistc(vals, weights, numedges)
    [N,edges] = hist(vals,numedges);
    if ¬isvector(vals) || ¬isvector(weights) || length(vals)≠length(weights)
        error('vals and weights must be vectors of the same size');
    end

    Nedge = length(edges);
    h = zeros(size(edges));

    for n = 1:Nedge-1
        ind = find(vals ≥ edges(n) & vals < edges(n+1));
        if ¬isempty(ind)
            h(n) = sum(weights(ind));
        end
    end

    ind = find(vals == edges(end));
    if ¬isempty(ind)
        h(Nedge) = sum(weights(ind));
    end
    h = h/(edges(2)-edges(1));
    bar(edges+(edges(2)-edges(1))/2,h);
end
```

```
function [h,edges] = weightedhistca(vals, weights, numedges,k)
    [N,edges] = hist(vals,numedges);
    if ¬isvector(vals) || ¬isvector(weights) || length(vals)≠length(weights)
        error('vals and weights must be vectors of the same size');
    end

    Nedge = length(k);
    h = zeros(size(k));

    for n = 1:Nedge-1
        ind = find(vals ≥ k(n) & vals < k(n+1));
        if ¬isempty(ind)
            h(n) = sum(weights(ind));
        end
    end

    ind = find(vals == k(end));
    if ¬isempty(ind)
        h(Nedge) = sum(weights(ind));
    end
    h = h/(k(2)-k(1));

end
```

A.2 Codes for Section 2.3

```

clear all; close all;

f = @(x) x;

mu = 0; sigma = 1;
p = @(x) 1/sqrt(2*pi*1) * exp(-(x-mu).^2/(2*sigma^2));

mu1 = 3; sigma = 1;
q = @(x) 1/sqrt(2*pi*1) * exp(-(x-mu1).^2/(2*sigma^2));

num_iter = [1e5]; num_trials = 50;
f_est = {{}};
for trial = 1:num_trials
    for iter = 1:length(num_iter)
```

```
        x = normrnd(mu1,sigma,num_iter(iter),1);

        w = p(x)./q(x);

        fw = (w/sum(w)).*f(x);

        f_est{iter}{trial} = sum(fw);
        w_var{iter}{trial} = var(w/sum(w));
    end
end

f_est_mean = zeros(size(f_est,2),1);
for i=1:size(f_est,2)
    f_est_mean(i) = mean([f_est{i}{:}]);
end
f_est_mean
xx=[-7:0.01:7]
plot(xx,p(xx))
hold on
plot(xx,q(xx))
hold on
weightedhistc(x,w/(sum(w)),100)
```

A.3 Codes for Section 2.4

```

clear all; close all;

f = @(x) x;

mu = 0; sigma = 1;
p = @(x) 1/sqrt(2*pi*1) * exp(-(x-mu).^2/(2*sigma^2));

mu1 = 3; sigma = 1;
q = @(x) 0.1;
%% importance sampling
num_iter = [1e5]; num_trials = 50;
f_est = {{}};
for trial = 1:num_trials
```

```
    for iter = 1:length(num_iter)


        x = -5 + 10*rand(num_iter(iter),1);

        w = p(x)./q(x);

        fw = (w/sum(w)).*f(x);

        f_est{iter}{trial} = sum(fw);
        w_var{iter}{trial} = var(w/sum(w));
    end
end
f_est_mean = zeros(size(f_est,2),1);
for i=1:size(f_est,2)
    f_est_mean(i) = mean([f_est{i}{:}]);
end
f_est_mean
xx=[-7:0.01:7]
plot(xx,p(xx))
hold on
plot([-5,5],[0.1,0.1])
hold on
plot([-5,-5],[0,0.1])
hold on
plot([5,5],[0,0.1])
hold on
weightedhistc(x,w/(sum(w)),100)
```

A.4 Codes for Section 3.4

```
clear all
sigma1 = 2;

beta = 1;
muu = 2;
pi_fun = @(x) 1/sqrt(2*pi*sigma1^2)*exp(-(x-muu).^2/(2*sigma1^2));%target ...
    distribution
```

```
q_fun = @(x,mu) 1/sqrt(2*pi*beta^2)*exp(-(x-mu).^2/(2*beta^2));%sampling ...
    distribution

n=200;
it=2000;
look=0;
u=[1/n:1/n:1];
kk=[-5:0.3:10];
x_ini = -5+15*muu*rand(n,1);
hp=0;
x=x_ini;

for iter=1:it


    for i = 1:n
        sum1=0;
        y(i)= normrnd(x(i),beta,1);

        for k = 1:n
            sum1=sum1+q_fun(y(i),x(k));

        end
        sum1=sum1/n;
        w(i)=pi_fun(y(i))/sum1;
    end

    w_bar=w/sum(w);
    [h1 edges]=weightedhistca(y,w/(sum(w)),51,kk);
    hp=hp+h1;

    if (iter==100)||(iter==500)||(iter==1000)
        if iter==100
            xx=[-5:0.01:10];
       plot(xx,pi_fun(xx));
        end

       hold on;

       %weightedhistc(x,w/(sum(w)),50);
k=[-5:0.3:10];
```

```

bar(kk+(kk(2)-kk(1))/2,hp/iter);
hold on;
xlabel('x');
ylabel('Density');legend('N(2,4)','Sampling');
axis([-5.5 10.5 0 0.22]);
        %pause;
    end

    %bootstrap
    p=w_bar;


    if iter≠it
        for h = 1:n
            p_sum=0;
            for p_sear=1:n
                p_sum=p_sum+p(p_sear);
                if u(h)<p_sum
                    x(h)=y(p_sear);
                    break;
                end

            end
        end
        look=look+1;
    end
    look;
end
xx=[-5:0.01:10];
plot(xx,pi_fun(xx));
hold on;
%weightedhistc(x,w/(sum(w)),50);
k=[-5:0.3:10];
bar(kk+(kk(2)-kk(1))/2,hp/it);
xlabel('x');
ylabel('Density');

%legend('N(2,4)','Iter.100','Iter.500','Iter.1000','Iter.2000');
legend('N(2,4)','Sampling');
axis([-5.5 10.5 0 0.22]);
```

A.5 Codes for Section 3.5

```
clear all
sigma1 = 2;
beta = 1;
muu = 2;
pi_fun = @(x) 1/sqrt(2*pi*sigma1^2)*exp(-(x-muu).^2/(2*sigma1^2));
q_fun = @(x,mu) 1/sqrt(2*pi*beta^2)*exp(-(x-mu).^2/(2*beta^2));
pi_fun1 = @(x) x*(1/sqrt(2*pi*sigma1^2)*exp(-(x-muu).^2/(2*sigma1^2)));

n=50;
it=2000;
kk=[-5:0.3:10];

look=0;


x_ini = -5+15*muu*rand(n,1);
x=x_ini;
hp=0;

for iter=1:it


    for i = 1:n
        sum1=0;
        y(i)= normrnd(x(i),beta,1);

        for k = 1:n
            sum1=sum1+q_fun(y(i),x(k));

        end
        sum1=sum1/n;
        w(i)=pi_fun(y(i))/sum1;

end

w_bar=w/sum(w);
[h1 edges]=weightedhistca(y,w/(sum(w)),51,kk);
    hp=hp+h1;

    if (iter==100)||(iter==500)||(iter==1000)
         if iter==100
            xx=[-5:0.01:10];
        plot(xx,pi_fun(xx));
         end

        hold on;

        %weightedhistc(x,w/(sum(w)),50);
k=[-5:0.3:10];

%bar(kk+(kk(2)-kk(1))/2,hp/iter);
plot(kk+(kk(2)-kk(1))/2,hp/iter)
hold on;
xlabel('x');
ylabel('Density');legend('N(2,4)','Sampling');
axis([-5.5 10.5 0 0.22]);
        %pause;
    end
%multinomial transformation

z=n*w_bar;


if iter≠it

for i=1:n
    p_mt=zeros(1,n);

    [a b]=max(z);
    J=b;
    p_mt(J)=min(1,z(J));
    z(J)=z(J)-p_mt(J);
    while (1-sum(p_mt)>1e-7)


        p_check=p_mt;
        k1=find(z>0);

        y_choose=[];
        for j=k1
            y_choose(j) = (y(J)-y(j))^2;
        end
        del1=[];
        del1=find(y_choose≠0); y_pick=y_choose(del1);

        [c,d]=min(y_pick);
        K=find(y_choose==c);

        p_mt(K)=min(1-sum(p_mt),z(K));
        z(K)=z(K)-p_mt(K);
        if sum(z)==0
            break
        end

    end
    x(i)=sum(p_mt.*y);
end
end
look=look+1;
end

xx=[-5:0.01:10];
%plot(xx,pi_fun(xx));
hold on;
%weightedhistc(x,w/(sum(w)),50);
%bar(kk+(kk(2)-kk(1))/2,hp/it);
plot(kk+(kk(2)-kk(1))/2,hp/it);
xlabel('x');
ylabel('Density');
legend('N(2,4)','Sampling');
%legend('N(2,4)','Iter.100','Iter.500','Iter.1000','Iter.2000');
axis([-5.5 10.5 0 0.22]);
```

A.6 Codes for Section 3.6

```

function [error]=mt_iss(n,it)

```

```
sigma1 = 2;
beta = 1;
muu = 2;
pi_fun = @(x) 1/sqrt(2*pi*sigma1^2)*exp(-(x-muu).^2/(2*sigma1^2));
q_fun = @(x,mu) 1/sqrt(2*pi*beta^2)*exp(-(x-mu).^2/(2*beta^2));
pi_fun1 = @(x) x*(1/sqrt(2*pi*sigma1^2)*exp(-(x-muu).^2/(2*sigma1^2)));

look=0;
x_ini = -5+15*muu*rand(n,1);
x=x_ini;
hp=0;

for iter=1:it


    for i = 1:n
        sum1=0;
        y(i)= normrnd(x(i),beta,1);

        for k = 1:n
            sum1=sum1+q_fun(y(i),x(k));

        end
        sum1=sum1/n;
        w(i)=pi_fun(y(i))/sum1;

end

w_bar=w/sum(w);

%multinomial transformation

z=n*w_bar;


if iter≠it

for i=1:n
    p_mt=zeros(1,n);

    [a b]=max(z);
    J=b;
    p_mt(J)=min(1,z(J));
    z(J)=z(J)-p_mt(J);
    while (1-sum(p_mt)>1e-7)


        p_check=p_mt;
        k1=find(z>0);

        y_choose=[];
        for j=k1
            y_choose(j) = (y(J)-y(j))^2;
        end
        del1=[];
        del1=find(y_choose≠0); y_pick=y_choose(del1);

        [c,d]=min(y_pick);
        K=find(y_choose==c);

        p_mt(K)=min(1-sum(p_mt),z(K));
        z(K)=z(K)-p_mt(K);
        if sum(z)==0
            break
        end

    end
    x(i)=sum(p_mt.*y);
end
end
look=look+1;
end



xx=[-5:0.01:10];
[hp kk]=weightedhistc(x,w/(sum(w)),50);
error=0;

for i=1:50
    error=error+(abs(pi_fun(kk(i)+0.15)-hp(i)))/abs(pi_fun(kk(i)+0.15));

```

```
end

error=error/50;

end
```

```

function [error]=mt_iss(n,it)

sigma1 = 2;
beta = 1;
muu = 2;
pi_fun = @(x) 1/sqrt(2*pi*sigma1^2)*exp(-(x-muu).^2/(2*sigma1^2));
q_fun = @(x,mu) 1/sqrt(2*pi*beta^2)*exp(-(x-mu).^2/(2*beta^2));
pi_fun1 = @(x) x*(1/sqrt(2*pi*sigma1^2)*exp(-(x-muu).^2/(2*sigma1^2)));

look=0;
x_ini = -5+15*muu*rand(n,1);
x=x_ini;
hp=0;

for iter=1:it


    for i = 1:n
        sum1=0;
        y(i)= normrnd(x(i),beta,1);

        for k = 1:n
            sum1=sum1+q_fun(y(i),x(k));

        end
        sum1=sum1/n;
        w(i)=pi_fun(y(i))/sum1;

end

w_bar=w/sum(w);

%multinomial transformation
```

```

z=n*w_bar;


if iter≠it

for i=1:n
    p_mt=zeros(1,n);

    [a b]=max(z);
    J=b;
    p_mt(J)=min(1,z(J));
    z(J)=z(J)-p_mt(J);
    while (1-sum(p_mt)>1e-7)


        p_check=p_mt;
        k1=find(z>0);

        y_choose=[];
        for j=k1
            y_choose(j) = (y(J)-y(j))^2;
        end
        del1=[];
        del1=find(y_choose≠0); y_pick=y_choose(del1);

        [c,d]=min(y_pick);
        K=find(y_choose==c);

        p_mt(K)=min(1-sum(p_mt),z(K));
        z(K)=z(K)-p_mt(K);
        if sum(z)==0
            break
        end

    end
    x(i)=sum(p_mt.*y);
end
end
look=look+1;
end
```

```



xx=[-5:0.01:10];
[hp kk]=weightedhistc(x,w/(sum(w)),50);
error=0;

for i=1:50
    error=error+(abs(pi_fun(kk(i)+0.15)-hp(i)))/abs(pi_fun(kk(i)+0.15));

end

error=error/50;

end
```

```
function [error]=boot_is_1(n,it)

sigma1 = 2;
beta = 1;
muu = 2;
pi_fun = @(x) 1/sqrt(2*pi*sigma1^2)*exp(-(x-muu).^2/(2*sigma1^2));
q_fun = @(x,mu) 1/sqrt(2*pi*beta^2)*exp(-(x-mu).^2/(2*beta^2));
x_ini = -5+15*muu*rand(n,1);
x=x_ini;
hpp=0;
u=[1/n:1/n:1];

for iter=1:it


    for i = 1:n
        sum1=0;
        y(i)= normrnd(x(i),beta,1);

        for k = 1:n
            sum1=sum1+q_fun(y(i),x(k));

        end
        sum1=sum1/n;
        w(i)=pi_fun(y(i))/sum1;

end

w_bar=w/sum(w);



p=w_bar;


    if iter≠it
        for h = 1:n
            p_sum=0;
            for p_sear=1:n
                p_sum=p_sum+p(p_sear);
                if u(h)<p_sum
                    x(h)=y(p_sear);
                    break;
                end

            end
        end
        look=look+1;
    end
end
[hp kk]=weightedhistc(x,w/(sum(w)),50);
error=0;

for i=1:50
    error=error+(abs(pi_fun(kk(i)+0.15)-hp(i)))/abs(pi_fun(kk(i)+0.15));

end

error=error/50;
end
```

A.7 Codes for Section 4.2

```
clear all;
X(1)=5;
%X(1)=0;%initial point
N=1e4;
p_fun = @(x) 0.3*normpdf(x,-2,0.5)+0.7*normpdf(x,5,0.8);
xx=[-5:0.1:10];
plot(xx,p_fun(xx));
hold on;

for i=1:N
    u=rand;x=X(i);
    x_prop=normrnd(x,1);
    px_prop=p_fun(x_prop); px=p_fun(x);
    qx_prop=normpdf(x_prop,x,1);qx=normpdf(x,x_prop,1);

    if u<min(1,px_prop*qx/(px*qx_prop))
        X(1+i)=x_prop;
    else
        X(1+i)=x;
    end
end
N0=1;
nb=histc(X(N0+1:N),xx); bar(xx+0.1/2,nb/(N-N0)/0.1);
xlabel('x');
ylabel('Density');
legend('Gaussian Mixture','Sampling');
%legend('N(2,4)','Iter.100','Iter.500','Iter.1000','Iter.2000');
axis([-5.5 10.5 0 1.2]);
```

A.8 Codes for Section 4.3

```
clear all
sigma1 = 4;
beta = 1;
muu = 2;
pi_fun = @(x) 0.3*normpdf(x,-2,0.5)+0.7*normpdf(x,5,0.8);
q_fun = @(x,mu) 1/sqrt(2*pi*beta^2)*exp(-(x-mu).^2/(2*beta^2));

n=200;
```

```
it=1000;
kk=[-5:0.1:10];
x_ini = 5*muu*rand(n,1);
x=x_ini;
hp=0;

for iter=1:it


    for i = 1:n
        sum1=0;
        y(i)= normrnd(x(i),beta,1);

        for k = 1:n
            sum1=sum1+q_fun(y(i),x(k));

        end
        sum1=sum1/n;
        w(i)=pi_fun(y(i))/sum1;

end

w_bar=w/sum(w);
[h1 edges]=weightedhistca(y,w/(sum(w)),151,kk);
    hp=hp+h1;
%multinomial transformation

z=n*w_bar;


if iter≠it

for i=1:n
    p_mt=zeros(1,n);

    [a b]=max(z);
    J=b;
    p_mt(J)=min(1,z(J));
    z(J)=z(J)-p_mt(J);
    while (1-sum(p_mt)>1e-7)

```

```

        p_check=p_mt;
        k1=find(z>0);

        y_choose=[];
        for j=k1
            y_choose(j) = (y(J)-y(j))^2;
        end
        del1=[];
        del1=find(y_choose≠0); y_pick=y_choose(del1);

        [c,d]=min(y_pick);
        K=find(y_choose==c);

        p_mt(K)=min(1-sum(p_mt),z(K));
        z(K)=z(K)-p_mt(K);
        if sum(z)==0
            break
        end

    end
    x(i)=sum(p_mt.*y);
end
end
look=look+1;
end



xx=[-5:0.01:10];
plot(xx,pi_fun(xx));
hold on;
%weightedhistc(x,w/(sum(w)),50);

hp=hp/it;
bar(kk+(kk(2)-kk(1))/2,hp);
xlabel('x');
ylabel('Density');
legend('Gaussian Mixture','Sampling');
%legend('N(2,4)','Iter.100','Iter.500','Iter.1000','Iter.2000');
axis([-5.5 10.5 0 0.5]);
```

A.9 Codes for Section 5.4

```
clear all;

muu=2;
beta = 1;
mu_post=3;

sigma_post=1;

Kb_true=3;
Kd_true=3;
sigma1=1;
t=[0.01:0.01:1];
G_fun = @(kb,kd) exp(-kb*t)+kb/kd;
epsilo=normrnd(0,sigma1,1,100);
Y=G_fun(Kb_true,Kd_true)+epsilo;

pi_fun = @(kb,kd) ...
    log(lognpdf(kb,mu_post,sigma_post)*lognpdf(kd,mu_post,sigma_post)...
*exp(-1/(2*sigma1^2)*sum((Y-G_fun(kb,kd)).^2)));
q_fun = @(x,mu) 1/sqrt(2*pi*beta^2)*exp(-(x-mu).^2/(2*beta^2));%sampling ...
    distribution


n=5000;
it=50;
u=[1/n:1/n:1];
kk=[0:0.1:10];
x_ini = rand(2,n);
hp1=0;hp2=0;

x=x_ini;
for iter=1:it


    for i = 1:n
        sum1=0;
        sum2=0;
        y(1,i)= normrnd(x(1,i),beta,1);
        y(2,i)= normrnd(x(2,i),beta,1);

        for k = 1:n
            sum1=sum1+q_fun(y(1,i),x(1,k))*q_fun(y(2,i),x(2,k));


        end
        sum1=sum1/n;

        log_w(i)=pi_fun(y(1,i),y(2,i))-log(sum1);
        w(i)=exp(log_w(i));
    end

    w_bar=w/sum(w);
    [h1 edges1]=weightedhistca(y(1,:),w_bar,101,kk);
    hp1=hp1+h1;
    [h2 edges2]=weightedhistca(y(2,:),w_bar,101,kk);
    hp2=hp2+h2;

    %bootstrap
    z=n*w_bar;


if iter≠it

for i=1:n
    p_mt=zeros(1,n);

    [a b]=max(z);
    J=b;
    p_mt(J)=min(1,z(J));
    z(J)=z(J)-p_mt(J);
    while (1-sum(p_mt)>1e-7)


        p_check=p_mt;
        k1=find(z>0);

        y_choose=[];
        for j=k1
            y_choose(j) = (y(J)-y(j)).^2;
        end
        del1=[];
        del1=find(y_choose≠0); y_pick=y_choose(del1);

        [c,d]=min(y_pick);
        K=find(y_choose==c);

        p_mt(K)=min(1-sum(p_mt),z(K));
        z(K)=z(K)-p_mt(K);
        if sum(z)==0
            break
        end

    end
    x(1,i)=sum(p_mt.*y(1,:));
    x(2,i)=sum(p_mt.*y(2,:));
end
end

end
[N C]=hist3(y',[50 50]);
pcolor(N)
```

A.10 Codes for Section 5.5

```
clear all;
X(:,1)=[0 0]; %initial point
N=5e6;
muu=2;
beta = 1;
mu_post=3;
sigma_post=1;

Kb_true=3;
Kd_true=3;
sigma1=1;
t=[0.01:0.01:1];
G_fun = @(kb,kd) exp(-kb*t)+kb/kd;
epsilo=normrnd(0,sigma1,1,100);
```

```
Y=G_fun(Kb_true,Kd_true)+epsilo;
p_fun = @(kb,kd) lognpdf(kb,mu_post,sigma_post)*lognpdf(kd,mu_post,sigma_post)...
        *exp(-1/(2*sigma1^2)*sum((Y-G_fun(kb,kd)).^2));

sigmaa=[1 0;0 1];


for i=1:N

    u=rand;x=X(:,i);
    x_prop=mvnrnd(x,sigmaa);
    px_prop=p_fun(x_prop(1),x_prop(2)); px=p_fun(x(1),x(2));
    qx_prop=mvnpdf(x_prop,x',sigmaa);
    qx=mvnpdf(x,x_prop',sigmaa);

    if u<min(1,(px_prop*qx)/(px*qx_prop))
        X(:,1+i)=x_prop;
    else
        X(:,1+i)=x;
    end


    end




[N1 C]=hist3(X',[50 50]);
pcolor(N1)
```