HITRO

Josef Leydold

leydold@statistik.wu-wien.ac.at

Department für Statistik und Mathematik, WU Wien

HITRO

A Fast Automatic MCMC Procedure

Josef Leydold

leydold@statistik.wu-wien.ac.at

Department für Statistik und Mathematik, WU Wien

Monte Carlo Method

- **Task:**
  Compute expectation of some function g with respect to a distribution with density f:

$$\mathsf{E}_f(g) = \int_{\mathbb{R}^n} g(\mathbf{x})\, f(\mathbf{x}) d\mathbf{x}$$

- **Method:** Monte Carlo Integration

$$\mathsf{E}_f(g) \approx \frac{1}{N} \sum_{i=1}^{N} g(\mathbf{X}_i) \qquad \text{where } \mathbf{X}_i \sim f$$

- **Problem:** Generation of $\mathbf{X}_i \sim f$

Generation of IID Random Vectors

In high dimension it is **very** difficult to generate RVs.

- The **conditional distribution method** requires knowledge of all full conditional distribution functions. (multivariate inversion method)
- The **rejection method** does not work well in higher (> 10) dimensions, since rejection constant and/or the memory requirements explode exponentially.

  E.g. Generate points uniformly distributed in a ball by rejection from the hypercube. For dimension 50 the acceptance probability is about 10^{-28}.

However, there is no necessity for **IID** random vectors.

Markov Chain Monte Carlo

Run a Markov chain whose stationary distribution is the required distribution.

Many such methods exist:

- Metropolis-Hastings algorithm
- Random walk sampler
- Independence sampler
- Gibbs sampler

Do not confuse with the task of simulating a Markov chain.

Markov Chain Sampler

Advantages:

- Algorithms are much simplier.
- More generally applicable.

Disadvantages:

- No IID random vectors.
- The generated points are dependent and follow the desired distribution only approximately.
- Rate of convergence of the Markov chain is a problem. Only heurist rules for convergence exist.

Markov Chain Sampler

From the WinBUGS manual:

Beware!

MCMC can be dangerous!

Metropolis-Hastings Algorithm

Such a Markov chain can be generated by means of proposal densities $q(x|\mathbf{X})$.

Algorithm:

- Choose a starting point $\mathbf{X}_0$; set $t \leftarrow 0$.

  Generate proposal $\tilde{\mathbf{X}}$ with density $q(x|\mathbf{X}_t)$

  Generate $U \sim \mathcal{U}(0,1)$.

  If $U \leq \frac{f(\tilde{\mathbf{X}})}{f(\mathbf{X}_t)} \frac{q(\mathbf{X}_t|\tilde{\mathbf{X}})}{q(\tilde{\mathbf{X}}|\mathbf{X}_t)}$ set $\mathbf{X}_{t+1} \leftarrow \tilde{\mathbf{X}}$

  Otherwise set $\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t$.

  Increment t and continue.

Metropolis-Hastings Algorithm

Such a Markov chain can be generated by means of proposal densities $q(x|\mathbf{X})$.

Algorithm:

Choose a starting point $\mathbf{X}_0$; set $t \leftarrow 0$.

- Generate proposal $\tilde{\mathbf{X}}$ with density $q(x|\mathbf{X}_t)$

Generate $U \sim \mathcal{U}(0,1)$.

If $U \leq \dfrac{f(\tilde{\mathbf{X}})}{f(\mathbf{X}_t)} \dfrac{q(\mathbf{X}_t|\tilde{\mathbf{X}})}{q(\tilde{\mathbf{X}}|\mathbf{X}_t)}$ set $\mathbf{X}_{t+1} \leftarrow \tilde{\mathbf{X}}$

Otherwise set $\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t$.

Increment t and continue.

Metropolis-Hastings Algorithm

Such a Markov chain can be generated by means of proposal densities $q(x|\mathbf{X})$.

Algorithm:

- Choose a starting point $\mathbf{X}_0$; set $t \leftarrow 0$.
- Generate proposal $\tilde{\mathbf{X}}$ with density $q(x|\mathbf{X}_t)$
- Generate $U \sim \mathcal{U}(0,1)$.
- If $U \leq \frac{f(\tilde{\mathbf{X}})}{f(\mathbf{X}_t)} \frac{q(\mathbf{X}_t|\tilde{\mathbf{X}})}{q(\tilde{\mathbf{X}}|\mathbf{X}_t)}$ set $\mathbf{X}_{t+1} \leftarrow \tilde{\mathbf{X}}$
- Otherwise set $\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t$.
- Increment t and continue.

Metropolis-Hastings Algorithm

Such a Markov chain can be generated by means of proposal densities $q(x|\mathbf{X})$.

Algorithm:

Choose a starting point $\mathbf{X}_0$; set $t \leftarrow 0$.

Generate proposal $\tilde{\mathbf{X}}$ with density $q(x|\mathbf{X}_t)$

Generate $U \sim \mathcal{U}(0,1)$.

- If $U \leq \dfrac{f(\tilde{\mathbf{X}})}{f(\mathbf{X}_t)} \dfrac{q(\mathbf{X}_t|\tilde{\mathbf{X}})}{q(\tilde{\mathbf{X}}|\mathbf{X}_t)}$ set $\mathbf{X}_{t+1} \leftarrow \tilde{\mathbf{X}}$

Otherwise set $\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t$.

Increment t and continue.

Metropolis-Hastings Algorithm

Such a Markov chain can be generated by means of proposal densities $q(x|\mathbf{X})$.

Algorithm:

- Choose a starting point $\mathbf{X}_0$; set $t \leftarrow 0$.
- Generate proposal $\tilde{\mathbf{X}}$ with density $q(x|\mathbf{X}_t)$
- Generate $U \sim \mathcal{U}(0,1)$.
- If $U \leq \dfrac{f(\tilde{\mathbf{X}})}{f(\mathbf{X}_t)} \dfrac{q(\mathbf{X}_t|\tilde{\mathbf{X}})}{q(\tilde{\mathbf{X}}|\mathbf{X}_t)}$ set $\mathbf{X}_{t+1} \leftarrow \tilde{\mathbf{X}}$
- Otherwise set $\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t$.
- Increment t and continue.

Metropolis-Hastings Algorithm

Such a Markov chain can be generated by means of proposal densities $q(x|\mathbf{X})$.

Algorithm:

Choose a starting point $\mathbf{X}_0$; set $t \leftarrow 0$.

Generate proposal $\tilde{\mathbf{X}}$ with density $q(x|\mathbf{X}_t)$

Generate $U \sim \mathcal{U}(0,1)$.

If $U \leq \frac{f(\tilde{\mathbf{X}})}{f(\mathbf{X}_t)} \frac{q(\mathbf{X}_t|\tilde{\mathbf{X}})}{q(\tilde{\mathbf{X}}|\mathbf{X}_t)}$ set $\mathbf{X}_{t+1} \leftarrow \tilde{\mathbf{X}}$

Otherwise set $\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t$.

- Increment t and continue.

Gibbs Sampler

Use **full conditional distributions**.

Algorithm:

- Choose a starting point $\mathbf{X}_0$; set $t \leftarrow 0$.

  Foreach $i = 1, \ldots, d$ generate $X_{t+1,i}$ from density $f(x_i | X_{t+1,1}, \ldots, X_{t+1,i-1}, X_{t,i+1}, \ldots, X_{t,d})$.

  Increment t and continue.

Gibbs Sampler

Use **full conditional distributions**.

Algorithm:

Choose a starting point $\mathbf{X}_0$; set $t \leftarrow 0$.

- Foreach $i = 1, \ldots, d$ generate $X_{t+1,i}$ from density $f(x_i | X_{t+1,1}, \ldots, X_{t+1,i-1}, X_{t,i+1}, \ldots, X_{t,d})$.

Increment t and continue.

Gibbs Sampler

Use **full conditional distributions**.

Algorithm:

Choose a starting point $\mathbf{X}_0$; set $t \leftarrow 0$.

Foreach $i = 1, \dots, d$ generate $X_{t+1,i}$ from density $f(x_i | X_{t+1,1}, \dots, X_{t+1,i-1}, X_{t,i+1}, \dots, X_{t,d})$.

- Increment t and continue.

Gibbs Sampler

Use **full conditional distributions**.

Algorithm:

Choose a starting point $\mathbf{X}_0$; set $t \leftarrow 0$.

Foreach $i = 1, \ldots, d$ generate $X_{t+1,i}$ from density $f(x_i | X_{t+1,1}, \ldots, X_{t+1,i-1}, X_{t,i+1}, \ldots, X_{t,d})$.

Increment t and continue.

Problem:
How to sample from conditional distributions?

Automatic MCMC Sampler

The above methods are **“recipes”** for the design of a Markov chain that converges to the desired distribution. They have to be **adjusted** to the particular generation problem.

Automatic MCMC Sampler

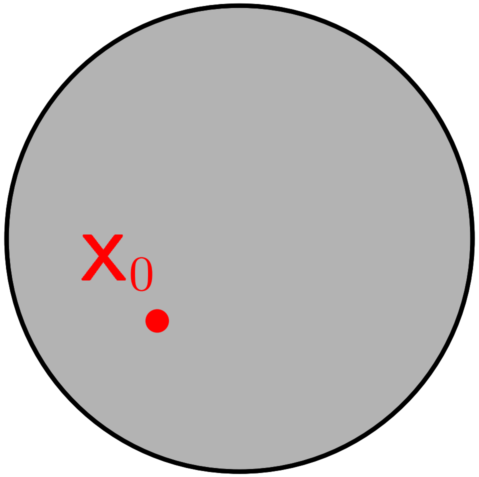
The above methods are **“recipes”** for the design of a Markov chain that converges to the desired distribution. They have to be **adjusted** to the particular generation problem.

A MCMC sampler that runs the **Hit-and-Run** sampler in combination with the **Ratio-of-Uniforms** method is much simpler and works for many distributions with given density out of the box.

Hit-and-Run Sampler

Gerate a sample of random points uniformly distributed in some fixed but arbitrary bounded open set $S \in \mathbb{R}^n$:

- Choose a starting point $\mathbf{X}_0 \in S$ and set $k = 0$.
- Generate a random direction $\mathbf{d}_k$ with distribution $\mathcal{D}$.
- Generate λ_k uniformly distributed in $\Lambda_k = S \cap \{\mathbf{x} \colon \mathbf{x} = \mathbf{x}_k + \lambda \mathbf{d}_k\}$.
- Set $\mathbf{X}_{k+1} = \mathbf{X}_k + \lambda_k \mathbf{d}_k$ and $k = k + 1$.
- Repeat from Step 2

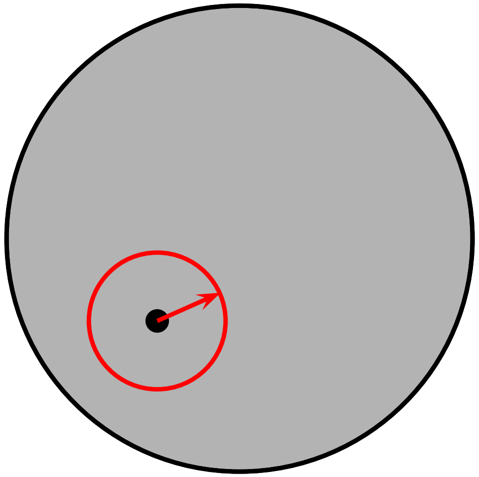


[Smith, 1984]

Hit-and-Run Sampler

Gerate a sample of random points uniformly distributed in some fixed but arbitrary bounded open set $S \in \mathbb{R}^n$:

- Choose a starting point $\mathbf{X}_0 \in S$ and set $k = 0$.
- Generate a random direction $\mathbf{d}_k$ with distribution $\mathcal{D}$.
- Generate λ_k uniformly distributed in $\Lambda_k = S \cap \{\mathbf{x} \colon \mathbf{x} = \mathbf{x}_k + \lambda \mathbf{d}_k\}$.
- Set $\mathbf{X}_{k+1} = \mathbf{X}_k + \lambda_k \mathbf{d}_k$ and $k = k + 1$.
- Repeat from Step 2

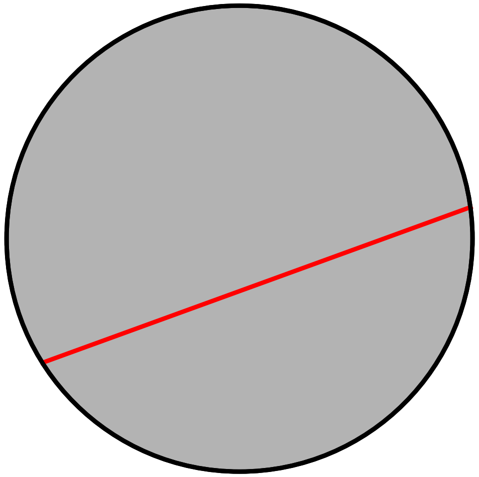


[Smith, 1984]

Hit-and-Run Sampler

Gerate a sample of random points uniformly distributed in some fixed but arbitrary bounded open set $S \in \mathbb{R}^n$:

- Choose a starting point $\mathbf{X}_0 \in S$ and set $k = 0$.
- Generate a random direction $\mathbf{d}_k$ with distribution $\mathcal{D}$.
- Generate λ_k uniformly distributed in $\Lambda_k = S \cap \{\mathbf{x} \colon \mathbf{x} = \mathbf{x}_k + \lambda \mathbf{d}_k\}$.
- Set $\mathbf{X}_{k+1} = \mathbf{X}_k + \lambda_k \mathbf{d}_k$ and $k = k + 1$.
- Repeat from Step 2

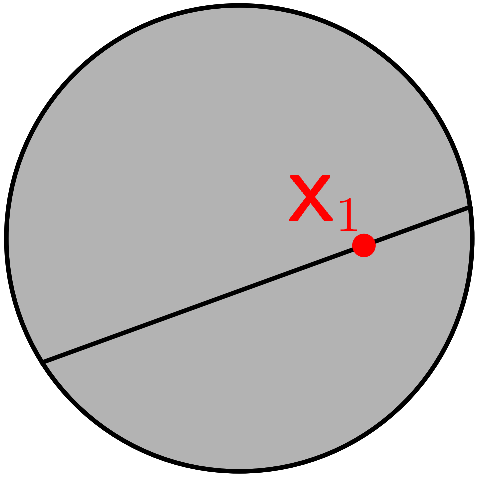


[Smith, 1984]

Hit-and-Run Sampler

Gerate a sample of random points uniformly distributed in some fixed but arbitrary bounded open set $S \in \mathbb{R}^n$:

- Choose a starting point $\mathbf{X}_0 \in S$ and set $k = 0$.
- Generate a random direction $\mathbf{d}_k$ with distribution $\mathcal{D}$.
- Generate λ_k uniformly distributed in $\Lambda_k = S \cap \{\mathbf{x}\colon \mathbf{x} = \mathbf{x}_k + \lambda \mathbf{d}_k\}$.
- Set $\mathbf{X}_{k+1} = \mathbf{X}_k + \lambda_k \mathbf{d}_k$ and $k = k + 1$.
- Repeat from Step 2

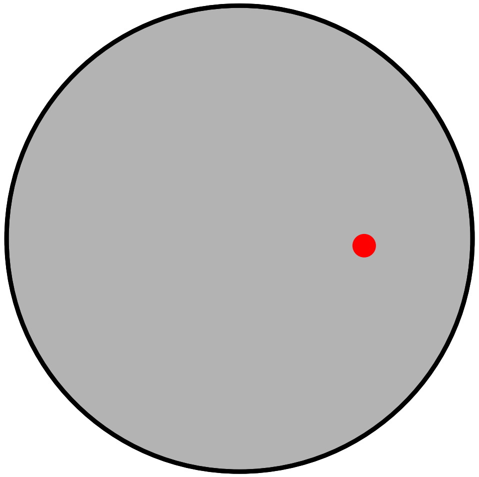


[Smith, 1984]

Hit-and-Run Sampler

Gerate a sample of random points uniformly distributed in some fixed but arbitrary bounded open set $S \in \mathbb{R}^n$:

- Choose a starting point $\mathbf{X}_0 \in S$ and set $k = 0$.
- Generate a random direction $\mathbf{d}_k$ with distribution $\mathcal{D}$.
- Generate λ_k uniformly distributed in $\Lambda_k = S \cap \{\mathbf{x}\colon \mathbf{x} = \mathbf{x}_k + \lambda \mathbf{d}_k\}$.
- Set $\mathbf{X}_{k+1} = \mathbf{X}_k + \lambda_k \mathbf{d}_k$ and $k = k + 1$.
- Repeat from Step 2



[Smith, 1984]

Hit-and-Run Sampler

The Markov chain generated by the Hit-and-Run Algorithm over converges geometrically fast to the target distribution. [Smith, 1984]

Important choices for the directional distribution $\mathcal{D}$ are

- **Hypersphere sampling:**
  $\mathcal{D}$ is the uniform distribution over the sphere.
- **Coordinate direction sampling:**
  $\mathcal{D}$ is the discrete uniform distribution over the axes.
- **Gibbs sampling:**
  Go through all axes in a fixed order.

Ratio-of-Uniforms

Theorem:

Let $f(\mathbf{x})$ be a positive integrable function on $\mathbb{R}^n$. Let $r > 0$ and suppose the point $(\mathbf{U}, V) \in \mathbb{R}^{n+1}$ with $\mathbf{U} = (U_1, \ldots, U_n)$ is uniformly distributed over the region

$$\mathcal{A}(f) = \mathcal{A}_r(f) = \left\{ (\mathbf{u}, v) \colon 0 < v < \sqrt[rn+1]{f(\mathbf{u}/v^r)} \right\},$$

then $\mathbf{X} = \mathbf{U}/V^r$ has probability density function prop. to $f(\mathbf{x})$.
[Wakefield, Gelfand, and Smith, 1991]

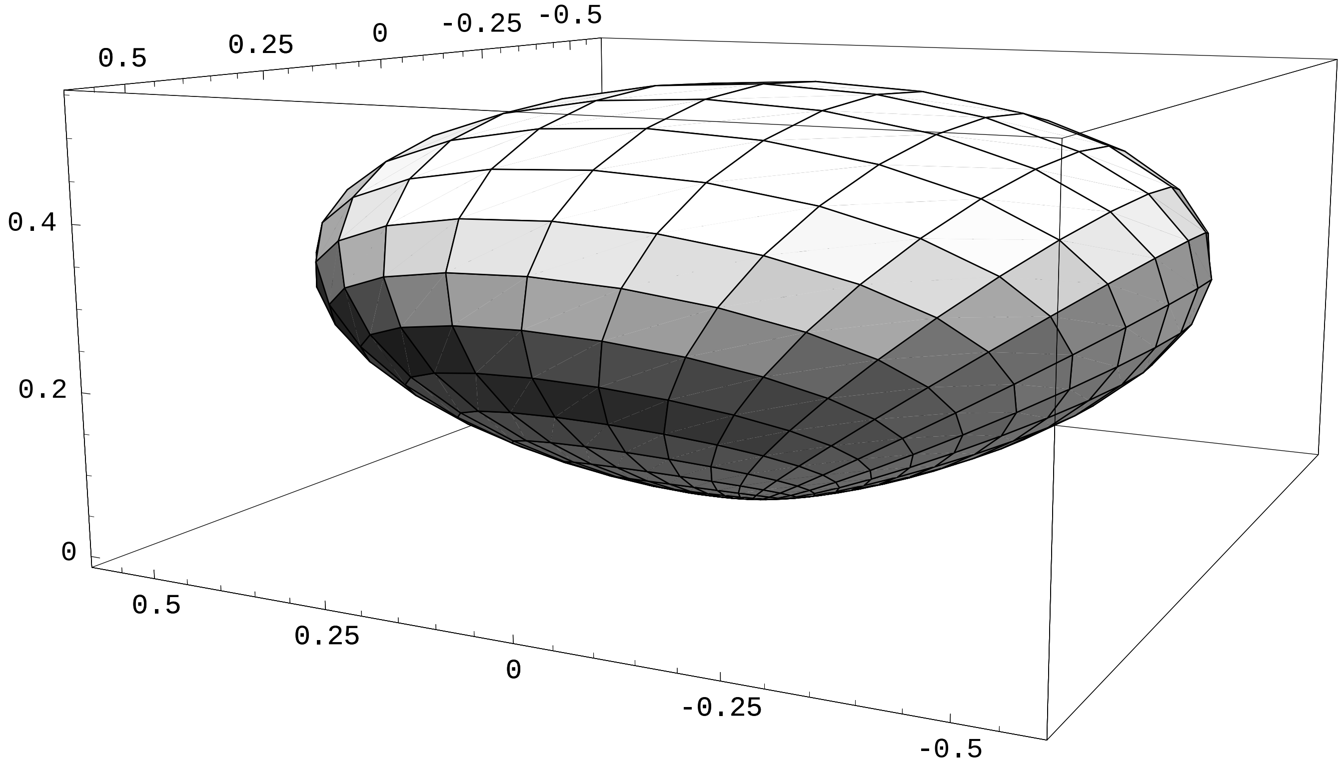
Ratio-of-Uniforms

Algorithm:

- Sample a point $(\mathbf{U}, V)$ uniformly in $\mathcal{A}(f)$.
- Return $\mathbf{X} = \mathbf{U}/V^r$.

Ratio-of-Uniforms

$\mathcal{A}(f)$ for standard bivariate normal distribution and $r = 1$.



Ratio-of-Uniforms

Theorem:

For a density f and $r = 1$ the region $\mathcal{A}(f) \subset \mathbb{R}^{n+1}$ is convex if and only if the transformed density $T(f(\mathbf{x})) = -(f(\mathbf{x}))^{-1/(n+1)}$ is concave. [L, 2000]

Notice that this holds, e.g., for all log-concave densities.

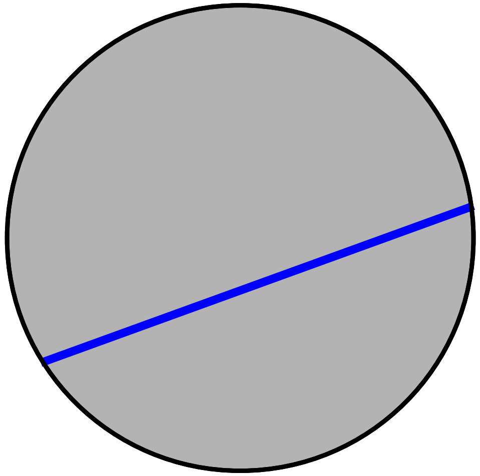
As a consequence of the Ratio-of-Uniforms methods the unbounded region below the graph of the density f is often mapped into a **bounded** and **convex** set.

The Hit-and-Run sampler is well suited to sample uniformly from $\mathcal{A}(f)$.

Adaptive Sampling from Line Segment Λ_k

Sampling from Λ_k by adaptive rejection:

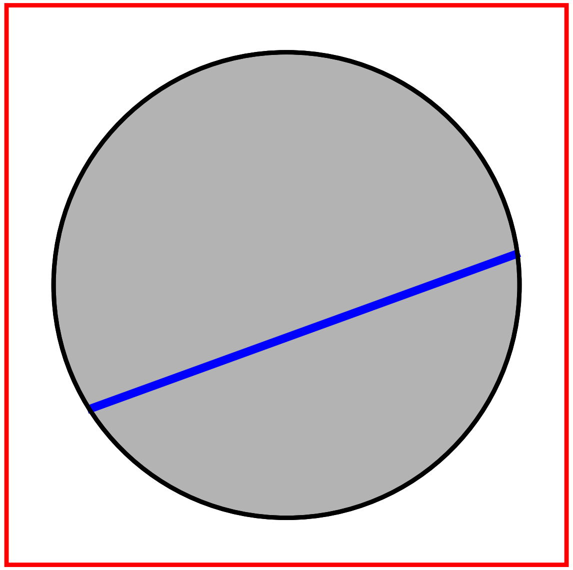
- Compute a bounding rectangle.
- Compute intersection L_k of line with rectangle.
- Sample point **X** in L_k.
- If **X** $\in \Lambda_k$ accept **X**.
- Else shrink segment L_k
  and try again.



Adaptive Sampling from Line Segment Λ_k

Sampling from Λ_k by adaptive rejection:

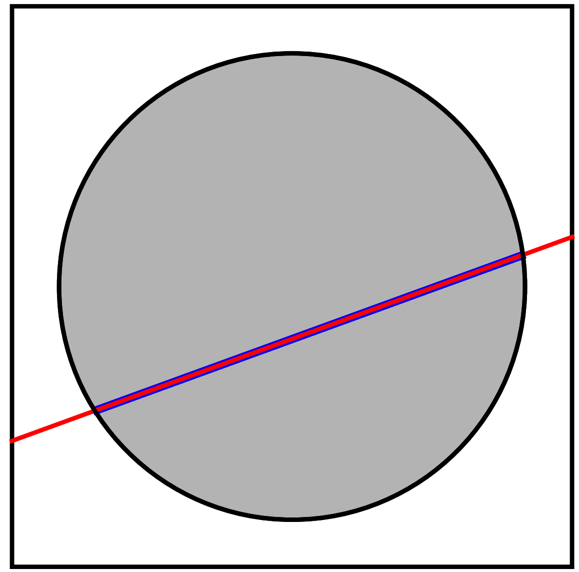
- Compute a bounding rectangle.
- Compute intersection L_k of line with rectangle.
- Sample point **X** in L_k.
- If **X** $\in \Lambda_k$ accept **X**.
- Else shrink segment L_k and try again.



Adaptive Sampling from Line Segment Λ_k

Sampling from Λ_k by adaptive rejection:

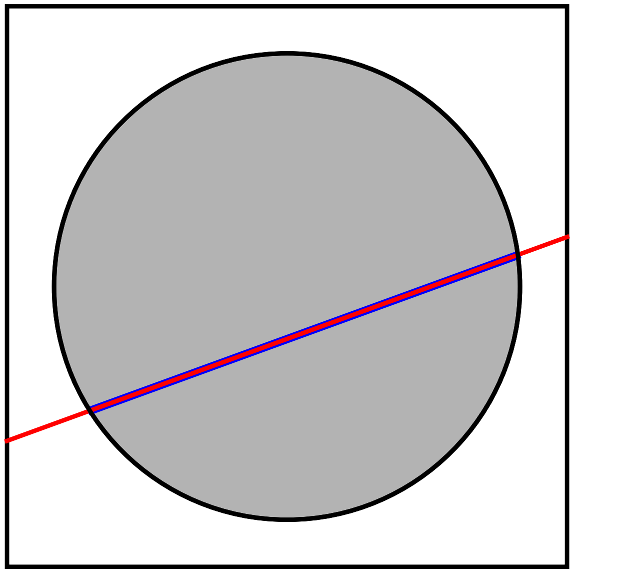
- Compute a bounding rectangle.
- Compute intersection L_k of line with rectangle.
- Sample point **X** in L_k.
- If **X** $\in \Lambda_k$ accept **X**.
- Else shrink segment L_k and try again.



Adaptive Sampling from Line Segment Λ_k

Sampling from Λ_k by adaptive rejection:

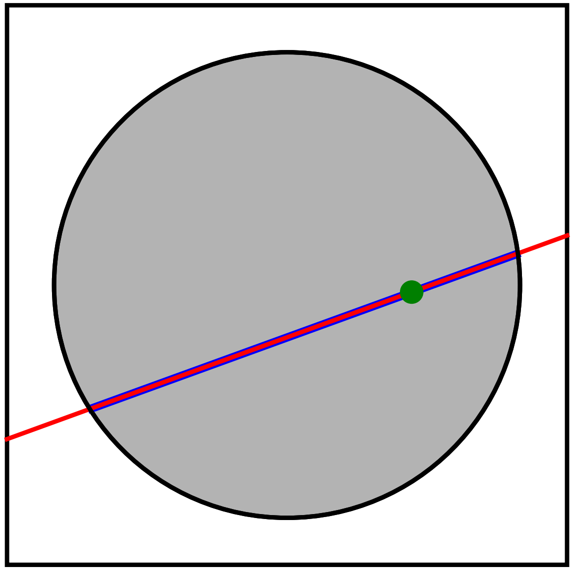
- Compute a bounding rectangle.
- Compute intersection L_k of line with rectangle.
- Sample point **X** in L_k.
- If **X** $\in \Lambda_k$ accept **X**.
- Else shrink segment L_k and try again.



Adaptive Sampling from Line Segment Λ_k

Sampling from Λ_k by adaptive rejection:

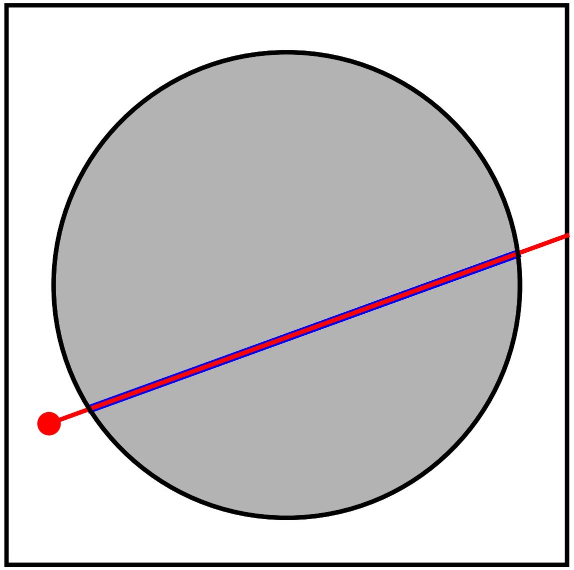
- Compute a bounding rectangle.
- Compute intersection L_k of line with rectangle.
- Sample point **X** in L_k.
- If **X** $\in \Lambda_k$ accept **X**.
- Else shrink segment L_k and try again.



Adaptive Sampling from Line Segment Λ_k

Sampling from Λ_k by adaptive rejection:

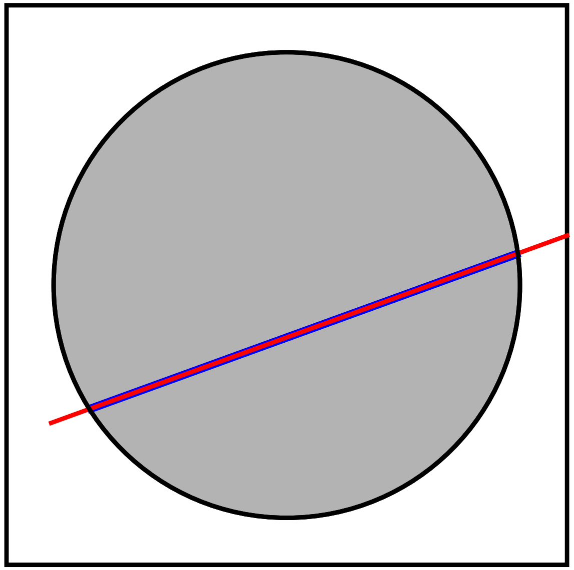
- Compute a bounding rectangle.
- Compute intersection L_k of line with rectangle.
- Sample point **X** in L_k.
- If **X** $\in \Lambda_k$ accept **X**.
- Else shrink segment L_k and try again.



Adaptive Sampling from Line Segment Λ_k

Sampling from Λ_k by adaptive rejection:

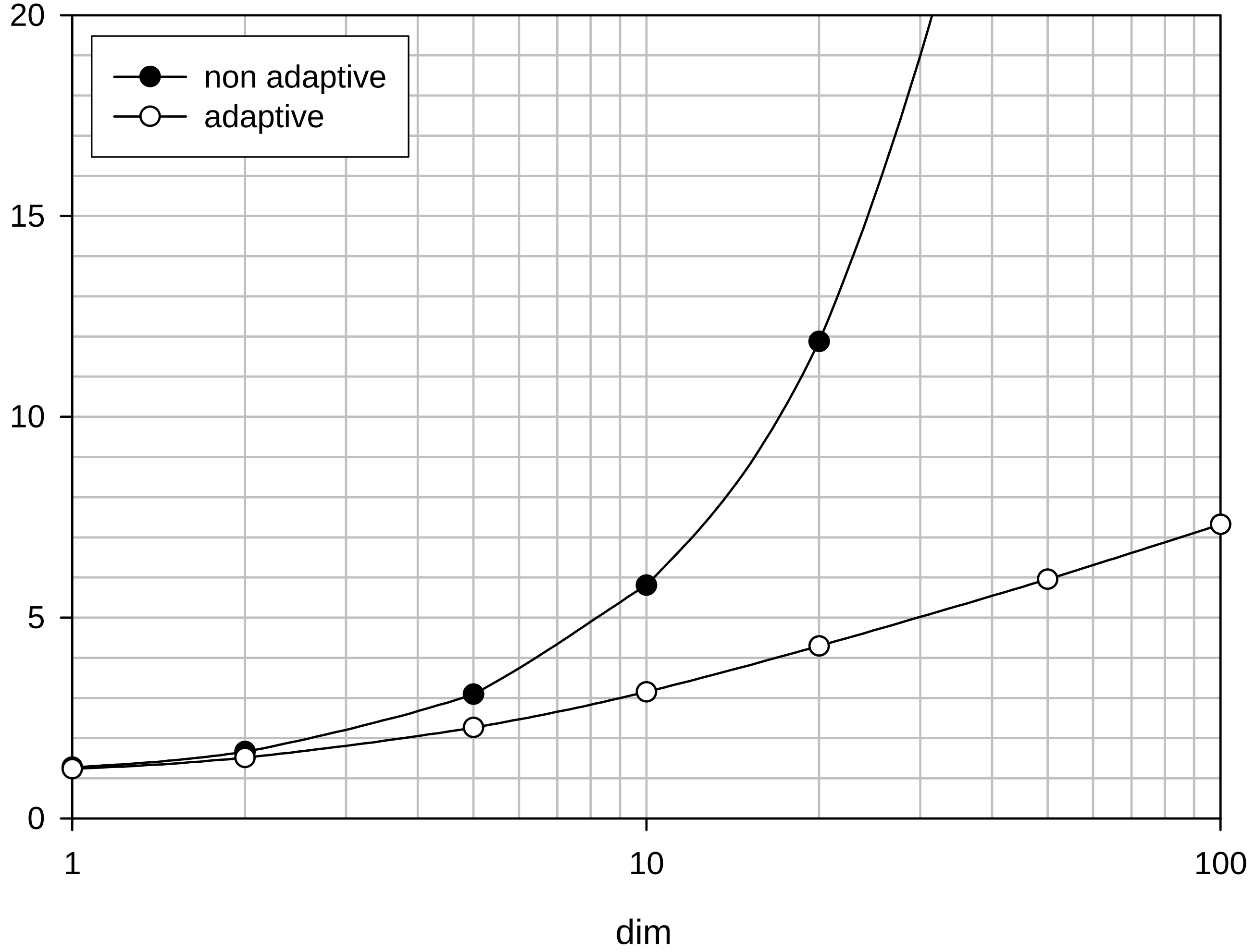
- Compute a bounding rectangle.
- Compute intersection L_k of line with rectangle.
- Sample point **X** in L_k.
- If **X** $\in \Lambda_k$ accept **X**.
- Else shrink segment L_k and try again.



The expected number of iterations is

$$1 - \log(\mu(L_k)/\mu(\Lambda_k)) \cdot \frac{e}{1-\log 2}$$

Performance: Calls to Density f



hypersphere sampling, multinormal distribution

Alternative Methods

Computing a bound rectangle is very expensive in higher dimensions. This can be avoided:

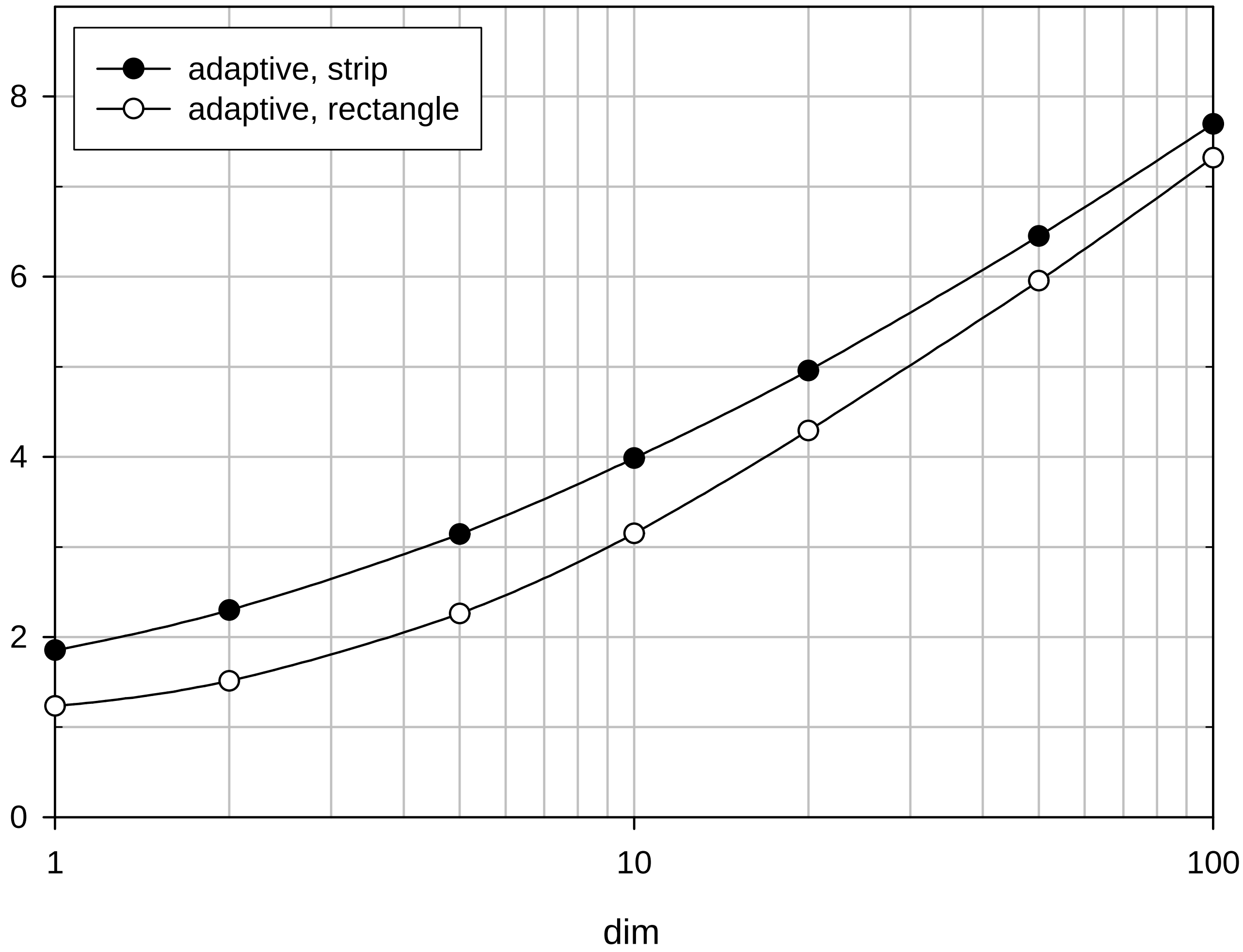
- Only use an upper bound for f.
  (Thus the bounding rectangle is a plate.)

  The expected number of iterations is only slightly increased in higher dimension.

  Or:

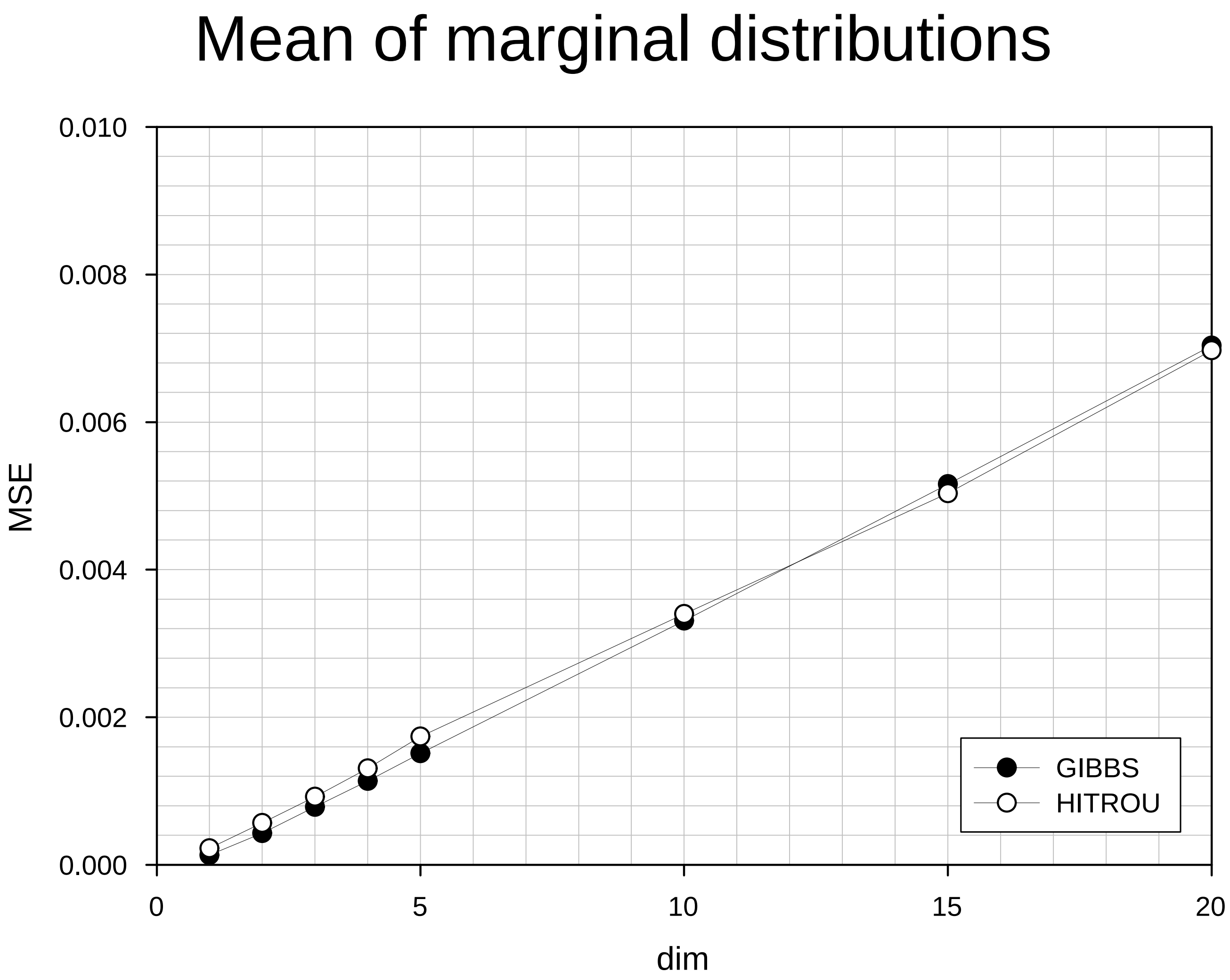
- Compute L_k by a simple search algorithm.
  (This works when $\mathcal{A}(f)$ is convex.)

Performance: Calls to Density f



hypersphere sampling, multinormal distribution

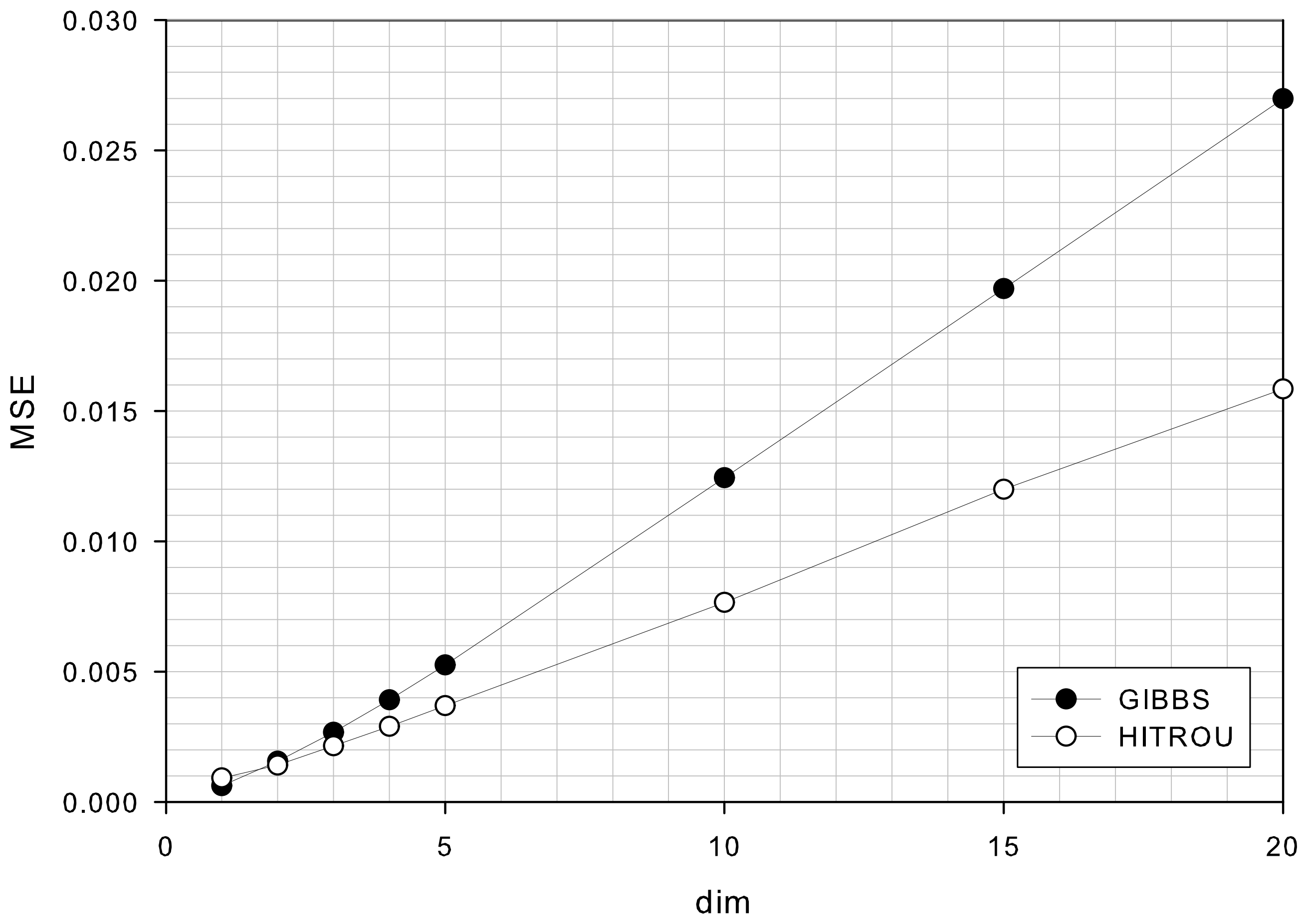
Performance: MSE compared to Gibbs Sampler



CD sampling, $N = 10^4$, multi-student distribution ($\nu = 8$)

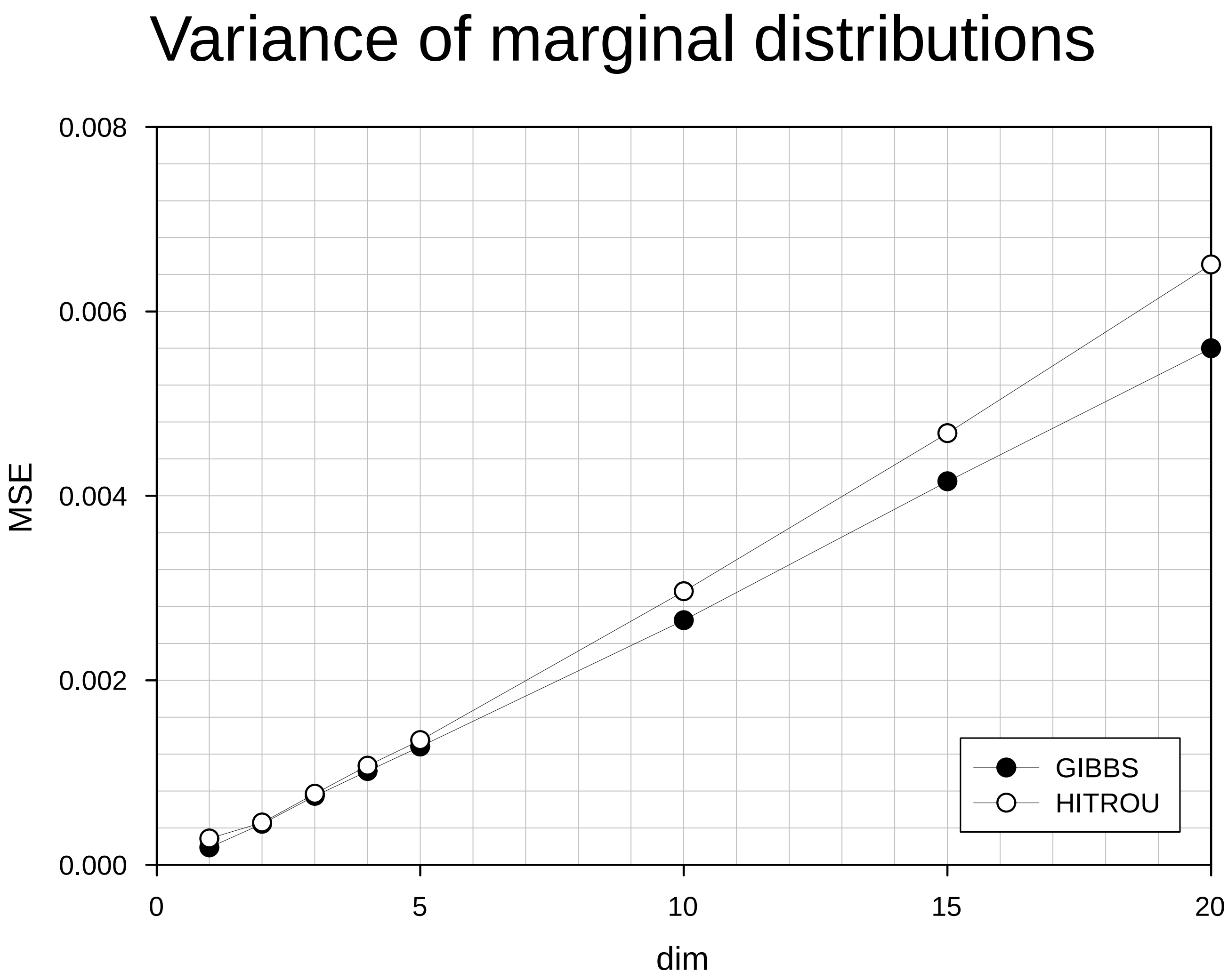
Performance: MSE compared to Gibbs Sampler

Variance of marginal distributions



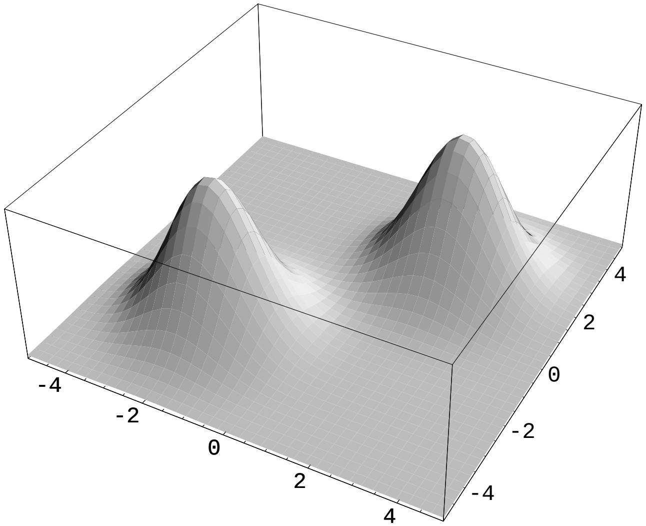
CD sampling, $N = 10^4$, multi-student distribution ($\nu = 8$)

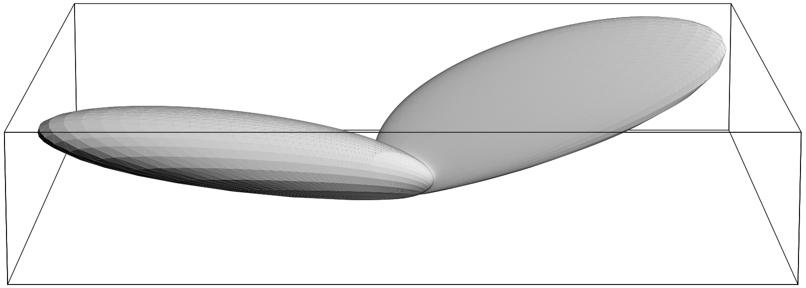
Performance: MSE compared to Gibbs Sampler

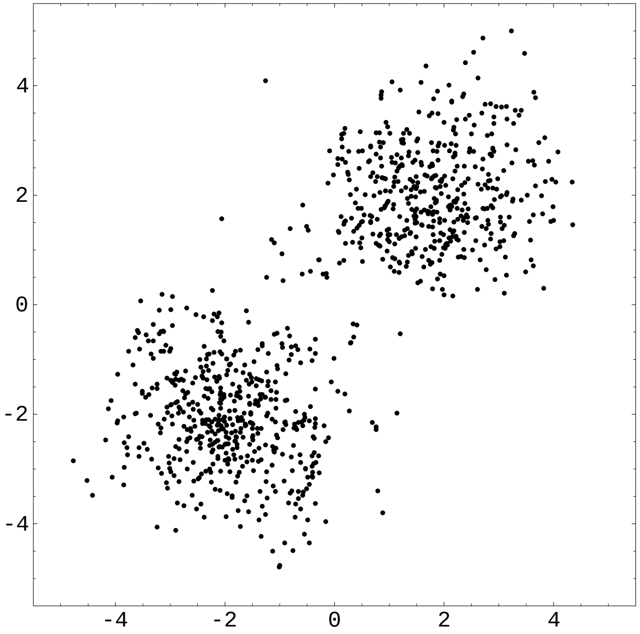


CD sampling, $N = 10^4$, multinormal distribution

Non-unimodal Density







Conclusion

The **HITRO** (Hit-and-run-Ratio-Of-uniforms) sampler is

- simple;
- easy to implement;
- relatively fast;
- works for many (not necessarily unimodal) distributions out of the box;
- performs similar to the Gibbs sampler (in terms of MSE) but does not need a special and/or expensive generator for conditional distributions.

Thank you!

References

R. L. Smith. Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32:1296–1308, 1984.

J. C. Wakefield, A. E. Gelfand, and A. F. M. Smith. Efficient generation of random variates via the ratio-of-uniforms method. *Statist. Comput.*, 1(2):129–133, 1991.