Slice Sampling: Theory and Methods, with Applications in R*

Submitted by:

Arkajyoti Bhattacharjee †
Max Cartwright †
Rezoanoor Rahman †

Supervised by:

Dr. Sebastian Kurtek †

THE OHIO STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

Submitted on:

26^{th} April, 2024

*This report has been prepared towards the partial fulfillment of the requirements of the course *STAT7730: Advanced Computational Statistics*

†Department of Statistics, The Ohio State University.

1 Motivation

Markov chain methods like Gibbs sampling(Gelfand (2000)) and the Metropolis algorithm(Metropolis et al. (1953), Hastings (1970)) are powerful tools for sampling from complex distributions. However, they often require tailored approaches for sampling from nonstandard distributions or selecting efficient proposal distributions. This limits their routine use and hinders automated sampler construction. Additionally, common Markov chain samplers can be inefficient due to sub-optimal step sizes and random walk behavior. Slice sampling (Neal (2003), Mira and Tierney (2002)) addresses these issues by adaptively sampling from the target distribution without the need for specialized tuning or proposal distributions, offering a more efficient exploration of the parameter space. Its adaptability to dependencies between variables makes it a promising approach, potentially surpassing traditional methods in efficiency and ease of use.

2 Slice sampling: The idea

Suppose we are interested in sampling from a distribution with density proportional to $f(x)$, where x belongs to a subset of $\mathbb{R}^n$. We introduce an auxiliary variable y and define a joint distribution over x and y that is uniform over the region $U = \{(x,y) : 0 < y < f(x)\}$. This joint density is given by

$$p(x,y) = \begin{cases} \frac{1}{Z} & \text{if } 0 < y < f(x) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $Z = \int_{\mathbb{R}^n} f(x)\,dx$. The marginal density for x is

$$p(x) = \frac{f(x)}{Z}. \tag{2}$$

To sample for x, we can sample jointly for (x,y) and then ignore y.
Gibbs sampling is one approach, where we alternate between sampling from the conditional distribution for y given the current x, which is uniform over $(0, f(x))$, and from the conditional distribution for x given the current y, which is uniform over the region $S = \{x : y < f(x)\}$, called the "slice" defined by y. If generating an independent point drawn uniformly from S is difficult, we can substitute some update for x that leaves the uniform distribution over S invariant. This approach is related to methods using multiple auxiliary variables, where the density function is proportional to a product of k functions. Each auxiliary variable y_i is bounded by $f_i(x)$. Gibbs sampling or other Markov chain procedures can be used to sample for $(x, y_1, \ldots, y_k)$. However, using many auxiliary variables can slow convergence.
Neal (2003) focuses on slice sampling with a single auxiliary variable, allowing for practicality across various problems. Updates for x need not produce a point drawn independently from S, but merely change x in a way that leaves the uniform distribution over S invariant. This enables the methods to be used for any continuous distribution, requiring only the computation of a function $f(x)$ proportional to the density.

3 Single-variable slice sampling methods

Slice sampling is typically applied when updating a single (real-valued) variable. While it's straightforward for univariate distributions, it's commonly used in multivariate cases by iteratively updating each variable. To update a variable x_i in a multivariate distribution $x = (x_1, \ldots, x_n)$, one needs to compute a function $f_i(x_i)$ proportional to $p(x_i | \{x_j\}_{j \neq i})$, where $\{x_j\}_{j \neq i}$ represents the values of the other variables.
Let x_0 denote the current value. To perform slice sampling for a single real-valued variable x, we follow a three-step procedure:

1. Draw a value y uniformly from $(0, f(x_0))$, defining a horizontal "slice" $S = \{x : y < f(x)\}$.
2. Find an interval $I = (L, R)$ around x_0 containing much of the slice.
3. Draw a new point x_1 from the part of the slice within this interval.

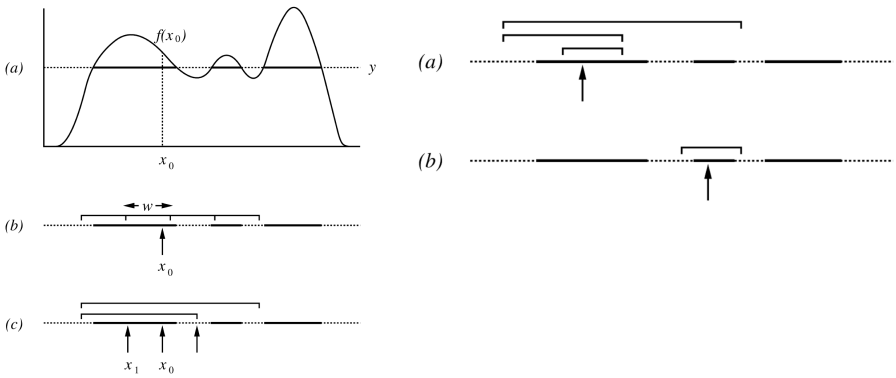

Figure 1: *Left: A single-variable slice sampling update using the stepping-out and shrinkage procedures.* A new point, x_1, is selected to follow the current point, x_0, in three steps. (a) A vertical level, y, is drawn uniformly from $(0, f(x_0))$, and used to define a horizontal "slice," indicated in bold. (b) An interval of width w is randomly positioned around x_0, and then expanded in steps of size w until both ends are outside the slice. (c) A new point, x_1, is found by picking uniformly from the interval until a point inside the slice is found. Points picked that are outside the slice are used to shrink the interval. *Right: The doubling procedure.* In (a), the initial interval is doubled twice, until both ends are outside the slice. In (b), where the start state is different, and the initial interval's ends are already outside the slice, no doubling is done[1].

Different schemes can be used to find the interval I, such as stepping out or doubling procedures (see Figure 1). The "stepping out" and "doubling" procedures are explained in more detail in the Appendix (1, 2). For other schemes and the relative merits and demerits, see Neal (2003).
After finding I, sampling from it is done by repeatedly shrinking the interval until a point within the set

$$A = \{x : X \in S \cap I \text{ and } P(Select\, I | At\, state\, x) = P(Select\, I | At\, state\, x_0)\} \tag{3}$$

of acceptable successor states is found. The shrinkage procedure is shown in detail in the Appendix (3). This ensures that the Markov chain leaves the distribution defined by $f(x)$ invariant.

4 Multivariate slice sampling and other methods

The idea of slice sampling can be extended to apply directly to multivariate distributions, rather than component-wise.

4.1 Multivariate slice sampling with hyper-rectangles

The multivariate extension of single-variable slice sampling involves replacing the interval $I = (L, R)$ with an axis-aligned hyperrectangle $H = \{x : L_i < x_i < R_i \text{ for all } i = 1, \ldots, n\}$, where L_i and R_i define the extent of the hyperrectangle along the axis for variable x_i. The process for obtaining the next state, $x_1 = (x_{1,1}, \ldots, x_{1,n})$, from the current state, $x_0 = (x_{0,1}, \ldots, x_{0,n})$, mirrors the single-variable procedure:

1. Draw a real value, y, uniformly from $(0, f(x_0))$, defining the slice $S = \{x : y < f(x)\}$.
2. Find a hyper-rectangle, $H = (L_1, R_1) \times \ldots \times (L_n, R_n)$, around x_0, ideally containing a substantial portion of the slice.
3. Draw the new point, x_1, from the part of the slice within this hyper-rectangle.

However, finding an optimal hyper-rectangle that fully encloses the slice is often challenging and may not be feasible in practice. Therefore, practical implementations may settle for a hyper-rectangle containing the current point. The process of updating involves drawing a new point from the part of the slice within this hyper-rectangle. Additionally, the hyper-rectangle is shrunk independently along each axis to ensure efficient sampling. The method is shown in Figure 2 and given in detail in the Appendix (5).

[1] Figure 1 adapted from Neal (2003).

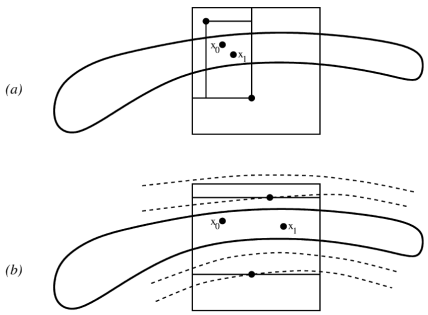

Figure 2: *Multivariate slice sampling with hyper-rectangles.* The heavy line outlines the slice, containing the current point, x_0. The large square is the initial hyper-rectangle. In (a), the hyper-rectangle is shrunk in all directions when the point drawn is outside the slice, until a new point, x_1, inside the slice is found. In (b), the hyper-rectangle is shrunk along only one axis, determined from the gradient and the current dimensions of the hyper-rectangle. The dashed lines are contours of the density function, indicating the direction of the gradient[2].

In the above method, all dimensions of the hyper-rectangle are shrunk uniformly until a point inside the slice is found. However, this approach may shrink dimensions unnecessarily in directions where the probability density does not vary rapidly. To address this issue, instead of uniformly shrinking all dimensions of the hyper-rectangle when the last point chosen was outside the slice, only the axis corresponding to variable x_i is shrunk (see Figure 2). The choice of which axis to shrink is based on maximizing the product

$$(R_i - L_i)|G_i|, \tag{4}$$

where G is the gradient of $\log f(x)$ at the last chosen point. This approach eliminates points outside the slice more effectively by focusing shrinkage on dimensions where the probability density changes the most. While these methods are relatively straightforward to implement, they may struggle to adapt well to different variable scales or dependencies present in the distribution.

4.2 Adaptive Multivariate Slice Sampling with Crumbs

In contrast to the hyperrectangle approach, adaptive multivariate slice sampling with crumbs introduces the concept of ”crumbs” to guide the selection of trial points, taking into account variable dependencies within the distribution. The process begins similarly by drawing a real value to define a slice in the multivariate space. A “crumb” is then drawn randomly from a distribution that depends on both the current point and the slice value. The first trial point is generated based on this crumb and the slice. If the trial point falls outside the slice, another crumb is drawn, and a new trial point is generated based on both the previous and current crumbs. This process continues until a trial point within the slice is found. By using different distributions for crumbs and trial points, this method has the potential to adapt more effectively to variable dependencies in the distribution. However, designing appropriate distributions and updating rules for crumbs can be complex and may require further research for optimization. For more details, see Neal (2003).

4.3 Over-relaxed slice sampling

In scenarios where variables need to be updated independently without considering their dependencies, small changes are necessary, often requiring numerous updates to transition between different parts of the distribution. To enhance sampling efficiency in such cases and mitigate the random walk behavior common in basic sampling techniques like Gibbs sampling, overrelaxed updates are employed. Unlike Gibbs sampling, where new values for variables are drawn independently from their conditional distributions, overrelaxed updates choose new values on the opposite side of the mode from the current value.

[2]Figure 2 adapted from Neal (2003).

Various attempts have been made to extend overrelaxation schemes to scenarios where conditional distributions are non-Gaussian. These schemes aim to ensure that almost every update is overrelaxed, minimizing the occurrence of rejections that leave the state unchanged, thus avoiding reintroducing random walk aspects into the sampling process. One such attempt involves integrating overrelaxed updates into slice sampling techniques. A method based on stepping out and bisection is detailed in Neal (2003), illustrated in Figure 3 and outlined in Appendix (6), aiming to minimize random walk behavior and improve sampling efficiency. Initially, a stepping-out procedure is applied to locate an interval around the current point, which is then refined using a bisection procedure to precisely determine the endpoints of the slice. The entire slice is approximated, and an overrelaxed update is performed by transitioning from the current point to a new point equidistant from the middle of the interval but on the opposite side. However, candidate points may be rejected if they lie outside predetermined intervals, ensuring detailed-balance is maintained.

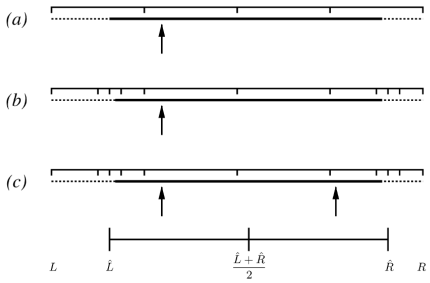

Figure 3: *Overrelaxation using the stepping out procedure and bisection.* In (a), an interval, (L,R), with both ends outside the slice is found by stepping out from the current point. In (b), the endpoints of the slice are located more accurately using bisection. In (c), a candidate point is found by flipping through the point half-way between the approximations to the endpoints. In this case, the candidate point is accepted, since it is within the slice, and within the interval prior to bisection[3].

To integrate overrelaxed updates into a comprehensive sampling scheme, such updates are sequentially applied to each variable, with a fixed number of cycles followed by a cycle of normal single-variable slice sampling updates. Alternatively, each update could have a small probability of being done normally. The frequency of normal updates serves as a tuning parameter, aiming to ensure that the Markov chain moves systematically rather than in a random walk pattern, covering a comparable distance to the largest dimension of the distribution. Adjusting the frequency helps to strike a balance between avoiding random walks and maintaining sampling efficiency.

5 Simulation Examples

Our first simulation followed the basic three-step algorithm provided in Section 4 of Neal (2003). In the univariate case with a monotone target distribution, the interval I = (L, R) drawn around each x_i can easily be defined such that it always contains the whole slice. Thus, the interval does not have to be altered to sample from the target distribution. Our simulation drew a sample of $n = 100,000$ from an $Exp(1)$ distribution. Figure 5 of the appendix provides a histogram of this simulation. The algorithm clearly sampled correctly, as the histogram mirrors the $Exp(1)$ density quite well.
Two subsequent simulations drew from a bimodal mixture of two normal distributions, defined as follows:

$$f(x) = \frac{1}{2}N(-10,36) + \frac{1}{2}N(15,4). \tag{5}$$

As the distribution is bimodal, the algorithm must be modified to ensure each x_i is drawn from within the acceptance region, A. The two methods we considered for this modification were the "stepping out" and "doubling" procedures. The "stepping out" procedure was implemented following Algorithms (1, 3) of the Appendix using parameters $w = 10$ and $m = 100$. The "doubling" procedure was implemented following Algorithms (2, 4) Appendix using parameters

[3]Figure 3 adapted from Neal (2003).

$w = 10$ and $p = 2$. Both simulations drew a sample of $n = 10,000$ from the target bimodal distribution. Side-by-side histograms are shown below in Figure 4. Again, the algorithms are sampling correctly as the density histograms clearly follow the target distribution. Both algorithms appear to have sampled equally well for the chosen parameters.

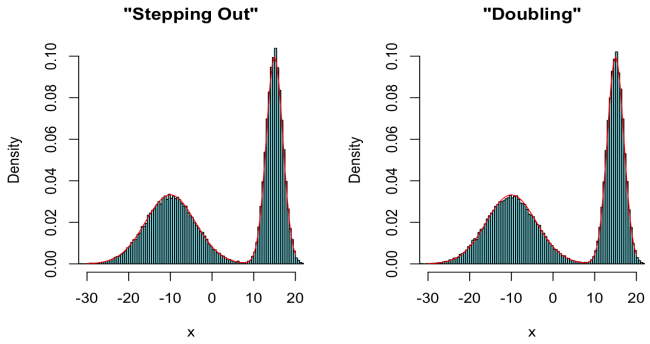

Figure 4: Side-by-side histograms comparing the "stepping out" and "doubling" procedures on the target distribution defined in (5).

An interesting consideration between the "stepping" out and "doubling" procedures is their computational efficiency. Neal (2003) postulates that the "doubling" procedure may be more efficient than the "stepping out" procedure when parameter w is chosen to be too small. However, run-time and total iteration calculations for various w parameters showed the "stepping out" procedure was always considerably faster than the "doubling" procedure. This could have been due to inefficiencies in our code, yet the "stepping out" procedure also completed in fewer total iterations than the "doubling" procedure. These results can be viewed in Figure 6 and Figure 7 of the Appendix.
Alternatively, Neal (2003) describes two ways to handle the "stepping out" procedure. One "naive" method continuously resamples until x_i is in the acceptance region. A second "shrinkage" method resamples in a similar manner, only it shrinks the interval when x_i is not in the acceptance region. The "shrinkage" method is deemed more efficient for this reason; it shouldn't have to resample as often when the interval is repeatedly shrunken. Note that the bimodal simulation above incorporated the "shrinkage" method.
To explore the computational efficiency of the two "stepping out" methods and the "doubling" method, we compared their run-times for various w parameter values and a fixed m value of $m = 10$ with $n = 10,000$. Plots of the observed run-times and total iterations run can be found in Figure 6 and Figure 7 of the Appendix. Both "stepping out" methods were comparably efficient for lower values of w. As w increased, the "shrinkage" method was dramatically more efficient than the "naive" method. This makes sense - for small w, the algorithm will not need to resample x_i very often as it should fall in the acceptance region. For larger w, the algorithm will need to resample often to ensure x_i is in the acceptance region, and thus the "shrinkage" method will be faster. Again, for this specific simulation, it looks like "stepping out" algorithm with shrinkage performed better than "doubling".

6 Discussion

Slice sampling poses a tractable solution to some shortcomings of comparable sampling methods like the Gibbs and Metropolis algorithms. By not requiring efficient proposal distributions, specific approaches for irregular distributions, or extensive parameter tuning, slice sampling becomes easy to implement. The most basic monotone univariate case that was covered in our presentation is very simple, yet more robust univariate slice approaches like the "stepping out" and "doubling" methods are still quite easy to code and implement. Our simulations showed these methods worked well on two sample target distributions. Similarly, when testing our code, we found that parameters could be chosen roughly and arbitrarily, and the algorithms would still sample correctly. Though not explored in our simulations, analogous multivariate slice sampling methods involving hyper-rectangles, crumbs, and over-relaxed updates are still fairly manageable to implement. Neal (2003) also discusses about reflective slice sampling as a multivariate slice sampling method. The correctness of this MCMC algorithm is also detailed in his paper. One topic of interest for our future work would be to compare the computational efficiency of Gibbs, Metropolis, and slice sampling approaches with various target distributions, both univariate and multivariate. This could provide a more complete and informed comparison between these sampling methods.

References

Gelfand, A. E. (2000). Gibbs sampling. *Journal of the American statistical Association*, 95(452):1300–1304.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Mira, A. and Tierney, L. (2002). Efficiency and convergence properties of slice samplers. *Scandinavian Journal of Statistics*, 29(1):1–12.

Neal, R. M. (2003). Slice sampling. *The annals of statistics*, 31(3):705–767.

7 Appendix

Algorithm 1: The "stepping out" procedure for finding an interval around the current point.

Input : f = function proportional to the density
x_0 = the current point
y = the vertical level defining the slice
w = estimate of the typical size of a slice
m = integer limiting the size of a slice to mw

Output: (L,R) = the interval found

$U \sim \text{Uniform}(0,1)$;
$L \leftarrow x_0 - w \times U$;
$R \leftarrow L + w$;
$V \sim \text{Uniform}(0,1)$;
$J \leftarrow \text{Floor}(m \times V)$;
$K \leftarrow (m-1) - J$;
while $J > 0$ ***and*** $y < f(L)$ **do**
  $L \leftarrow L - w$;
  $J \leftarrow J - 1$;
while $K > 0$ ***and*** $y < f(R)$ **do**
  $R \leftarrow R + w$;
  $K \leftarrow K - 1$;

Algorithm 2: The "doubling" procedure for finding an interval around the current point. *Note* that it is possible to save some computation in the second and later iterations of the loop, since only one of $f(L)$ and $f(R)$ will have changed from the previous iteration.

Input : f = function proportional to the density
x_0 = the current point
y = the vertical level defining the slice
w = estimate of the typical size of a slice
p = integer limiting the size of a slice to $2pw$
Output: (L,R) = the interval found
$U \sim \text{Uniform}(0,1)$;
$L \leftarrow x_0 - w \times U$;
$R \leftarrow L + w$;
$K \leftarrow p$;
while $K > 0$ ***and*** $(y < f(L)$ *or* $y < f(R))$ **do**
  $V \sim \text{Uniform}(0,1)$;
  if $V < \frac{1}{2}$ **then**
    $L \leftarrow L - (R - L)$;
  else
    $R \leftarrow R + (R - L)$;
  $K \leftarrow K - 1$;

Algorithm 3: The "shrinkage" procedure for sampling from the interval. Accept (x_1) is the notation for a test of whether a point, x_1, that is, within $S \cap I$ is an acceptable next state. If "stepping out" was used for constructing the interval, all points within $S \cap I$ are acceptable. If the "doubling" procedure was used, the point must pass the test of 4.

Input : f = function proportional to the density
x_0 = the current point
y = the vertical level defining the slice
(L,R) = the interval to sample from
Output: x_1 = the new point
$\overline{L} \leftarrow L$, $\overline{R} \leftarrow R$;
repeat
  $U \sim \text{Uniform}(0,1)$;
  $x_1 \leftarrow \overline{L} + U \times (\overline{R} - \overline{L})$;
  if $y < f(x_1)$ ***and*** *Accept*(x_1) **then**
    break;
  if $x_1 < x_0$ **then**
    $\overline{L} \leftarrow x_1$;
  else
    $\overline{R} \leftarrow x_1$;
until;

Algorithm 4: The test for whether a new point, x_1, that is, within $S \cap I$ is an acceptable next state, when the "doubling" procedure found the interval. The multiplication by 1.1 in the "while" condition guards against possible round-off error. The variable D tracks whether the intervals that would be generated from the new point differ from those leading to the current point. When they don't, time can be saved by omitting a check.

Input : f = function proportional to the density
x_0 = the current point
x_1 = the possible next point
y = the vertical level defining the slice
w = estimate of the typical size of a slice
(L, R) = the interval found by the doubling procedure, using w

Output: whether or not x_1 is acceptable as the next state

$\hat{L} \leftarrow L$, $\hat{R} \leftarrow R$, $D \leftarrow$ false;
repeat
  $M \leftarrow (\hat{L} + \hat{R})/2$;
  if $(x_0 < M$ *and* $x_1 \geq M)$ *or* $(x_0 \geq M$ *and* $x_1 < M)$ **then**
    $D \leftarrow$ true;
  if $x_1 < M$ **then**
    $\hat{R} \leftarrow M$;
  else
    $\hat{L} \leftarrow M$;
until $\hat{R} - \hat{L} \leq 1.1 \times w$;
if D ***and*** $y \geq f(\hat{L})$ ***and*** $y \geq f(\hat{R})$ **then**
  return *the new point is not acceptable*;
else
  return *the new point is acceptable if it is not rejected in the loop above*;

Algorithm 5: A simple multivariate slice sampling procedure, with randomly positioned hyper-rectangle and shrinkage in all directions, as in Figure 2, (b).

Input : f = function proportional to the density
x_0 = the current point, of dimension n
w_i = scale estimates for each variable, $i = 1, \ldots, n$
Output: x_1 = the new point
Step (a): Find value of y that defines the slice
$y \sim \text{Uniform}(0, f(x0))$
Step (b): Randomly position the hyperrectangle
$H = (L_1, R_1) \times \ldots \times (L_n, R_n)$
for $i = 1$ **to** n **do**
  $U_i \sim \text{Uniform}(0, 1)$
  $L_i \leftarrow x_{0,i} - w_i \cdot U_i$
  $R_i \leftarrow L_i + w_i$
Step (c): Sample from H, shrinking when points are rejected
repeat
  for $i = 1$ **to** n **do**
    $U_i \sim \text{Uniform}(0, 1)$
    $x_{1,i} \leftarrow L_i + U_i \cdot (R_i - L_i)$
    if $y < f(x_1)$ **then**
      exit loop
  for $i = 1$ **to** n **do**
    if $x_{1,i} < x_{0,i}$ **then**
      $L_i \leftarrow x_{1,i}$
    else
      $R_i \leftarrow x_{1,i}$
until;

Algorithm 6: The overrelaxation procedure using bisection.

Input : f = function proportional to the density
x_0 = the current point
y = the vertical level defining the slice
w = estimate of the typical size of a slice
a = integer limiting endpoint accuracy to $2^{-a}w$
(L,R) = interval found by the stepping out procedure using stepsize w

Output: x_1 = the new point

$\overline{L} \leftarrow L, \overline{R} \leftarrow R$

$\overline{w} \leftarrow w, \overline{a} \leftarrow a$

When the interval is only of size w, narrow it until the mid-point is inside the slice (or accuracy limit is reached).

if $R - L < 1.1 \times w$ **then**
    repeat
        $M \leftarrow (\overline{L} + \overline{R})/2$
        if $\overline{a} = 0$ *or* $y < f(M)$ **then**
            exit loop
        if $x_0 > M$ **then**
            $\overline{L} \leftarrow M$
        else
            $\overline{R} \leftarrow M$
        $\overline{a} \leftarrow \overline{a} - 1$
        $\overline{w} \leftarrow \overline{w}/2$
    until;

Refine endpoint locations by bisection, to the specified accuracy.

$\hat{L} \leftarrow \overline{L}, \hat{R} \leftarrow \overline{R}$

repeat
    $\overline{a} \leftarrow \overline{a} - 1$
    $\overline{w} \leftarrow \overline{w}/2$
    if $y \geq f(\hat{L} + \overline{w})$ **then**
        $\hat{L} \leftarrow \hat{L} + \overline{w}$
    if $y \geq f(\hat{R} - \overline{w})$ **then**
        $\hat{R} \leftarrow \hat{R} - \overline{w}$
until *while* $\overline{a} > 0$;

Find a candidate point by flipping from the current point to the opposite side of $(\hat{L}, \hat{R})$, then test it for acceptability.

$x_1 \leftarrow \hat{L} + \hat{R} - x_0$

if $x_1 < \overline{L}$ *or* $x_1 > \overline{R}$ *or* $y \geq f(x_1)$ **then**
    $x_1 \leftarrow x_0$

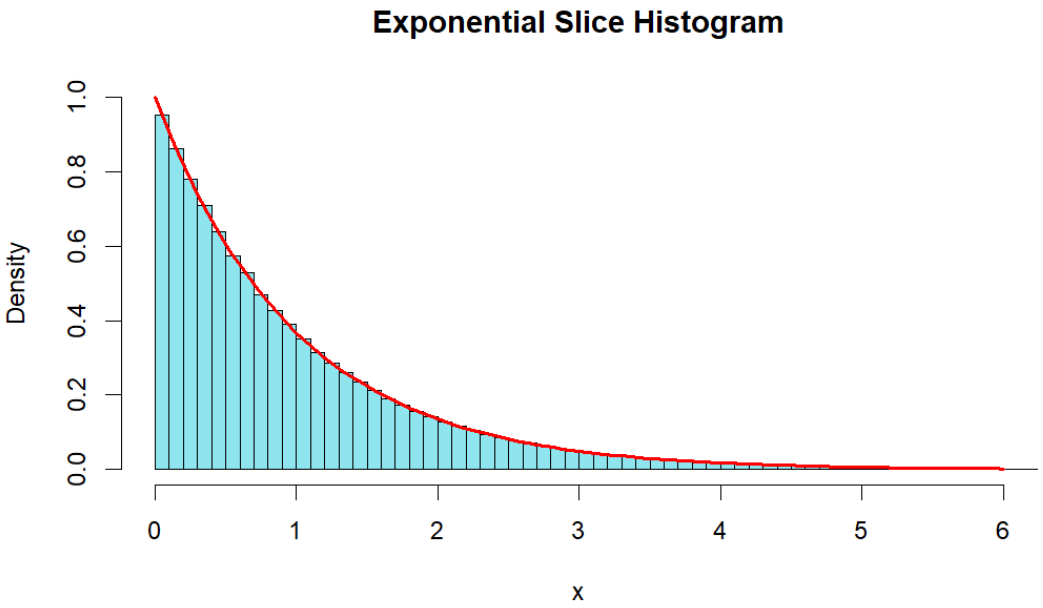

Figure 5: Histogram of simulation run with the $Exp(1)$ target distribution.

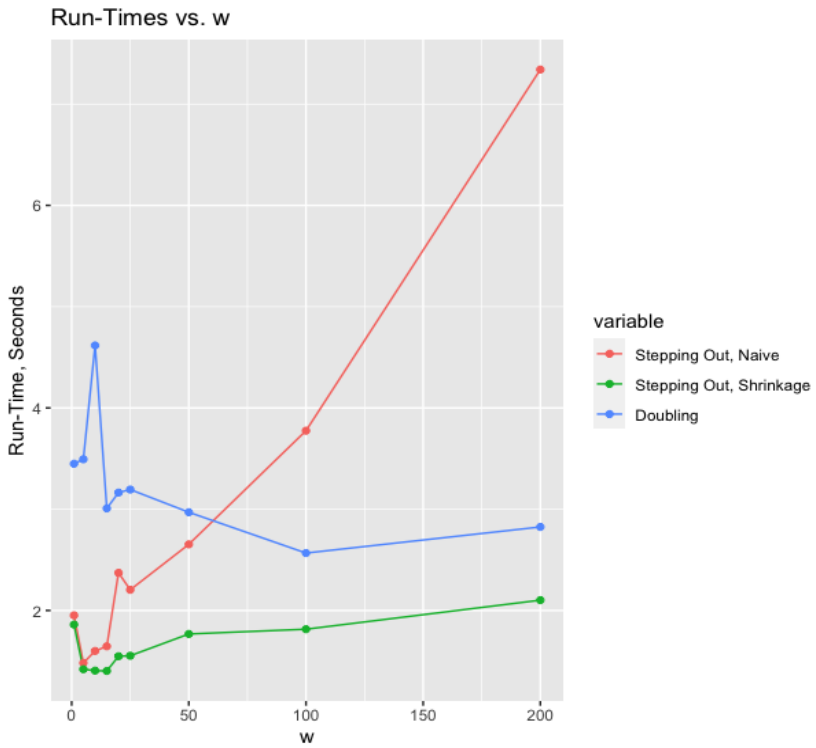

Figure 6: Run-time comparison between the "shrinkage" and "naive" submethods of the "stepping out" procedure and the "doubling" procedure using the bimodal target distribution in (5).

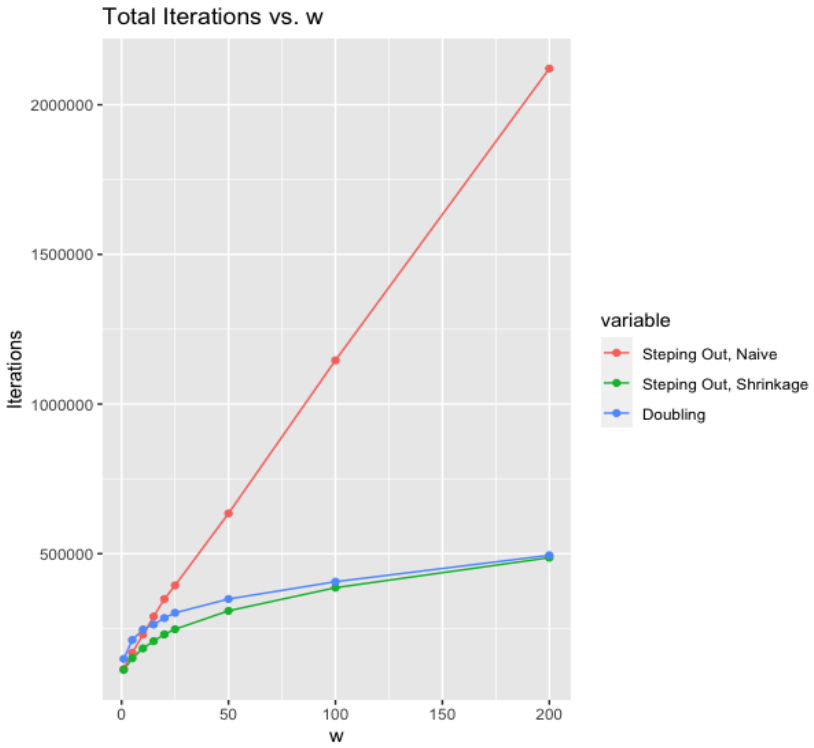

Figure 7: Total iterations run between the "shrinkage" and "naive" submethods of the "stepping out" procedure and the "doubling" procedure using the bimodal target distribution in (5).