entropy

MDPI

Article

TI-Stan: Model Comparison Using Thermodynamic Integration and HMC

R. Wesley Henderson *,† **and Paul M. Goggans** †

Department of Electrical Engineering, University of Mississippi, University, MS 38677, USA; goggans@olemiss.edu

* Correspondence: wesley.henderson11@gmail.com

† These authors contributed equally to this work.

Received: 30 September 2019; Accepted: 25 November 2019; Published: 27 November 2019

Abstract: We present a novel implementation of the adaptively annealed thermodynamic integration technique using Hamiltonian Monte Carlo (HMC). Thermodynamic integration with importance sampling and adaptive annealing is an especially useful method for estimating model evidence for problems that use physics-based mathematical models. Because it is based on importance sampling, this method requires an efficient way to refresh the ensemble of samples. Existing successful implementations use binary slice sampling on the Hilbert curve to accomplish this task. This implementation works well if the model has few parameters or if it can be broken into separate parts with identical parameter priors that can be refreshed separately. However, for models that are not separable and have many parameters, a different method for refreshing the samples is needed. HMC, in the form of the MC-Stan package, is effective for jointly refreshing the ensemble under a high-dimensional model. MC-Stan uses automatic differentiation to compute the gradients of the likelihood that HMC requires in about the same amount of time as it computes the likelihood function itself, easing the programming burden compared to implementations of HMC that require explicitly specified gradient functions. We present a description of the overall TI-Stan procedure and results for representative example problems.

Keywords: model comparison; MCMC; thermodynamic Integration; HMC

1. Introduction

TI is a numerical technique for evaluating model evidence integrals. The technique was originally developed [1] to estimate the free energy of a fluid. Various improvements and changes have been made over the decades, and the incarnation of the technique that our method is based on is the adaptively-annealed, importance sampling-based method described by Goggans and Chi [2]. Their implementation follows John Skilling's BayeSys [3], and both make use of BSS and the Hilbert curve to complete the implementation. This article proposes a modification of this method that uses PyStan [4,5] and the NUTS [6] instead of BSS and the Hilbert curve. A Python 3 implementation of this method by the authors can be found on GitHub (https://github.com/rwhender/ti-stan) [7]. This article is an adaptation of portions of the first author's doctoral dissertation [8] (Chapter 3). This article is an expanded version of an article in the proceedings of the 39th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering [7].

1.1. Motivation

The family of adaptively-annealed TI methods are important for solving model comparison problems in engineering, where we frequently need to evaluate complex physics-based mathematical models. TI methods with fixed annealing schedules (e.g., [9,10]) are useful for solving more traditional

statistics problems but tend to fail with the complex models that arise in engineering problems. TI methods that use BSS on the Hilbert curve are useful for a large set of problems; however, these methods see diminishing returns when the number of model parameters grows somewhat large ($>$ 10 or so). These performance issues can be mitigated if the model equation can be decomposed into additive components with identical form and equivalent joint priors on their parameters. However, for problems with many model parameters and with model equations that cannot be decomposed, a different class of methods is required.

Nested sampling [11,12] is another widely-used method for evaluating model evidence. One of the advantages of nested sampling is that its estimate of the evidence is not spoiled by an area of convexity in the likelihood function, whereas thermodynamic integration methods can be confounded by such likelihood functions. Skilling restates this difference in his recent conference paper [13], stating without providing examples that this difference constitutes a reason to prefer nested sampling over thermodynamic integration-based methods (also known as simulated annealing-based methods). In our experience, we have not encountered a model that yields a likelihood function with sufficient convexity to confound the thermodynamic integration estimate of the evidence. We have found instead that the biggest problem in implementing model evidence estimation methods lies in devising a method to generate new samples within an equi-likelihood contour that are sufficiently independent for nested sampling to proceed. It is worth noting that the Galilean sampling method advanced in [13] may solve these issues, shifting the balance toward nested sampling again. Further testing will be required to make that determination. For these reasons, we are motivated to further develop thermodynamic integration-based model evidence estimation methods.

1.2. Thermodynamic Integral Derivation

In this section, we derive the thermodynamic integral form of the model evidence integral. From Bayes' theorem, for model $\boldsymbol{M}$, data vector $\boldsymbol{D}$, model parameter vector $\boldsymbol{\Theta}$, and prior information I, we have

$$\mathrm{p}(\boldsymbol{D}|\boldsymbol{M}, I) = \int \mathrm{p}(\boldsymbol{D}|\boldsymbol{\Theta}, \boldsymbol{M}, I)\mathrm{p}(\boldsymbol{\Theta}|\boldsymbol{M}, I)\,\mathrm{d}\boldsymbol{\Theta}. \tag{1}$$

Now, introduce an inverse temperature parameter, β, that controls how much the likelihood influences an evidence related function $Z(\beta)$:

$$Z(\beta) = \int [\mathrm{p}(\boldsymbol{D}|\boldsymbol{\Theta}, \boldsymbol{M}, I)]^{\beta}\,\mathrm{p}(\boldsymbol{\Theta}|\boldsymbol{M}, I)\,\mathrm{d}\boldsymbol{\Theta}. \tag{2}$$

If β is 0, the integrand of Equation (2) is simply the prior. If β is 1, Equation (2) is the standard evidence integral, as in Equation (1).

Differentiate the log of the evidence with respect to β using the chain rule,

$$\frac{\mathrm{d}}{\mathrm{d}\beta}\log Z(\beta) = \frac{1}{Z(\beta)}\frac{\mathrm{d}}{\mathrm{d}\beta}Z(\beta). \tag{3}$$

Define a function we will call the energy,

$$E_L(\boldsymbol{\Theta}) = -\log \mathrm{p}(\boldsymbol{D}|\boldsymbol{\Theta}, \boldsymbol{M}, I). \tag{4}$$

Evaluate the derivative on the right-hand side of Equation (3), substituting in the definition of the energy Equation (4),

$$\frac{\mathrm{d}}{\mathrm{d}\beta}\log Z(\beta) = \frac{1}{Z(\beta)}\int \frac{\mathrm{d}}{\mathrm{d}\beta}\exp\left[-\beta E_L(\boldsymbol{\Theta})\right]\mathrm{p}(\boldsymbol{\Theta}|\boldsymbol{M},I)\,\mathrm{d}\boldsymbol{\Theta}\,, \tag{5}$$

$$= \int -E_L(\boldsymbol{\Theta})\frac{\exp\left[-\beta E_L(\boldsymbol{\Theta})\right]\mathrm{p}(\boldsymbol{\Theta}|\boldsymbol{M},I)}{Z(\beta)}\,\mathrm{d}\boldsymbol{\Theta}\,, \tag{6}$$

$$= -\int E_L(\boldsymbol{\Theta})\mathrm{p}(\boldsymbol{\Theta}|\boldsymbol{M},\boldsymbol{D},\beta,I)\,\mathrm{d}\boldsymbol{\Theta}\,. \tag{7}$$

Equation (7) is the negative expectation of the energy with respect to the parameter posterior conditioned on β,

$$\frac{\mathrm{d}}{\mathrm{d}\beta}\log Z(\beta) = -\left\langle E_L(\boldsymbol{\Theta})\right\rangle_\beta\,. \tag{8}$$

To obtain the log-evidence, integrate Equation (8) over the domain of β,

$$\log \mathrm{p}(\boldsymbol{D}|\boldsymbol{M},I) = -\int_0^1 \left\langle E_L(\boldsymbol{\Theta})\right\rangle_\beta\,\mathrm{d}\beta\,. \tag{9}$$

Equation (9) generally cannot be evaluated analytically, nor can the expected energy at each β within the integral be determined analytically. In practice, the following must be done to use Equation (9) effectively:

- use MCMC to estimate the expected energy at each value of β,
- develop a fixed schedule for β or employ an adaptive strategy to choose optimal stepped values of β,
- and compute the quadrature estimate of the integral.

1.3. Outline

The remainder of the article is organized as follows. Section 2 describes the general adaptive annealing with the importance sampling process our proposed method is built upon. Section 3 details how we represent model parameters within our proposed method. Section 4 describes the existing binary slice sampling-based thermodynamic integration methods that our proposed method is inspired by. Section 5 lays out our proposed HMC-based thermodynamic integration method. Section 6 enumerates four groups of example problems that were solved using both the existing BSS-based methods, and our proposed HMC-based method and presents results for each problem from each method. Section 7 concludes the article.

2. Adaptive Annealing and Importance Sampling

Much of the thermodynamic integration literature, e.g., [9,10], describes methods that use fixed annealing schedules. That is, all of the values of β are decided before any sampling is done. For problems where the distributions involved are not overly complex, this fixed schedule TI works well. However, for problems in which the distributions are based on physical models, are multi-modal, have correlations in the parameters, or have pronounced curving degeneracies, using a fixed schedule can lead to significant error in the log-evidence estimate. Signal detection problems tend to result in complex distributions like this, so to accommodate these distributions, an adaptive temperature annealing schedule can be used. Goggans and Chi [2] lay out a general procedure for deploying a thermodynamic integration method with adaptive annealing, which is described below:

1. Start at $\beta = 0$ where $\mathrm{p}(\boldsymbol{\Theta}|\boldsymbol{M},\boldsymbol{D},\beta,I) = \mathrm{p}(\boldsymbol{\Theta}|\boldsymbol{M},I)$, and draw C samples from this distribution (the prior).

2. Compute the Monte Carlo estimator for the expected energy at the current β,

$$\langle E_L(\boldsymbol{\Theta})\rangle_\beta \approx \frac{1}{N}\sum_{t=1}^{C} E_L(\boldsymbol{\Theta}_t)\,, \tag{10}$$

   where $\boldsymbol{\Theta}_t$ is the current position of the t-th Markov chain.
3. Increment β by $\Delta\beta_i$, where

$$\Delta\beta_i = \frac{\log \frac{\max w_j}{\min w_j}}{\max E_L(\boldsymbol{\Theta_i}) - \min E_L(\boldsymbol{\Theta_i})}, \tag{11}$$

   j is the index on the chains, w_j is the weight associated with chain j, and

$$w_j = \exp\left[-\Delta\beta_i E_L(\boldsymbol{\Theta_j})\right]. \tag{12}$$

4. Re-sample the population of samples using importance sampling.
5. Use MCMC to refresh the current population of samples. This yields a more accurate sampling of the distribution at the current temperature. This step can be easily parallelized, as each sample's position can be shifted independently of the others.
6. Return to step 2 and continue until β_i reaches 1.
7. Estimate Equation (9) using quadrature and the expected energy estimates built up using Equation (10).

2.1. Importance Sampling with Re-Sampling

The importance sampling with re-sampling technique used in this method is described in [2] and follows Liu et al. [14]. Once a new value of β is chosen according to the adaptive annealing procedure, importance sampling with re-sampling is used to sample the distribution under the new value of β. The importance weights used for re-sampling are

$$w_j = \frac{\mathrm{p}(\boldsymbol{\Theta}|\boldsymbol{M},\boldsymbol{D},\beta_{i+1},I)}{\mathrm{p}(\boldsymbol{\Theta}|\boldsymbol{M},\boldsymbol{D},\beta_i,I)} = \exp\big(-\Delta\beta_i E_L(\boldsymbol{\Theta}_j)\big), \tag{13}$$

which are then normalized,

$$W_j = J\frac{w_j}{\sum_{j=1}^{J} w_j}. \tag{14}$$

The Monte Carlo approximation of the posterior distribution under the new β is then

$$\mathrm{p}(\boldsymbol{\Theta}|\boldsymbol{M},\boldsymbol{D},\beta_{i+1},I) \approx \frac{1}{J}\sum_{j=1}^{J} W_j\delta(\boldsymbol{\Theta}-\boldsymbol{\Theta}_j). \tag{15}$$

At this point, the current population of samples needs to be re-sampled according to Equation (15). In order to determine the number of each sample that should be kept, the following calculation is used:

$$N_j = \sum_{k=0}^{J-1}\left[\mathrm{U}\left(u+k-\sum_{i=1}^{j-1} W_i\right) - \mathrm{U}\left(u+k-\sum_{i=1}^{j} W_i\right)\right], \tag{16}$$

where u is a uniform random variate on $[0,1]$, and U is the unit step function. If $N_j = 0$ for a particular sample, that sample is removed. If $N_j > 1$ for a particular sample, that sample is replicated. If $N = 1$ for a particular sample, that sample is simply kept with no change. A pseudo-code implementation of importance sampling with re-sampling is shown in Algorithm 1.

Algorithm 1 Importance sampling with re-sampling

```
function ImportanceSampling(w, α, E*, C)
    Sort w, α, and E* by w
    w ← (C / ∑w)w
    u ← Rand(0, 1)
    w_old ← w
    for i ← 1, C do
        w_i ← ∑_k^i w_old,i
    end for
    j ← 0
    q ← −1
    for m ← 1, C do
        while w_m > u do
            α_j ← α_m
            E*_j ← E*_m
            q ← m
            u ← u + 1
            j ← j + 1
        end while
    end for
end function
```

The importance sampling with re-sampling processes by nature can only replicate or remove existing samples. In order to accurately sample the posterior distribution, these samples need to be refreshed periodically, with their current positions serving as starting points. In the first method, a combination of binary slice sampling and leapfrog sampling accomplish this requirement. In the second method, the No U-Turn Sampler is used instead.

2.2. Adaptive Annealing

In step 3, a new value of β is chosen. The importance weights in Equation (13) depend on the change in the value of β, and we would like the weights to be set such that, on average, one sample is discarded and replaced at each temperature. The β update Equation (11) is designed with this goal in mind.

The ratio $W = \frac{\max w_j}{\min w_j}$ in Equation (11) is a constant set by the user. This constant is the desired ratio of the maximum importance weight in the sample population to the minimum importance weight. As long as this ratio is slightly greater than 1 (e.g., 1.05), the importance sampling process will usually discard and replace no more than one sample per temperature, as is the goal. If the distribution being sampled is not particularly challenging, higher values for the ratio constant can be used for a significant decrease in run time.

The denominator in Equation (11) allows the change in temperature to be controlled by the conditions encountered by the sampler. If the maximum energy and minimum energy are close, it is likely that we are sampling the distribution effectively, and β can be changed by a larger amount without disrupting the overall sampling. However, if the values are far apart, the implication is that we are sampling a more difficult portion of the distribution, and that a more gradual change in β will better serve the overall sampling.

3. Representing the Model Parameters

Users of the techniques described in this article may wish to evaluate the probabilities for any of a wide variety of mathematical models. The parameters of these models can be assigned any (proper) prior distribution, as the user deems necessary. Ultimately, the techniques developed in this article need to be able to sample model parameter spaces according to these prior distributions. While this challenge can be met in several ways, we take an approach that involves introducing a parameter transformation step into the energy function calculation.

Within the TI-based methods proposed here, parameters are always represented as either integer values on $[0, 2^B]$ (for TI with Binary Slice Sampling) or floating point values on the interval $[0, 1]$ (for TI with Stan), each with an independent uniform prior distribution. A parameter transformation routine must be included in the energy calculation function that maps these internal parameter representations to values with the correct prior probabilities for computing the energy. If the desired true prior distribution is also uniform, the parameter transformation amounts to a simple scaling operation. Other prior distributions, such as Gaussian distributions, have similar straightforward mapping functions.

In the case that a specialized mapping function is not available, the prior CDF may be used in all cases to perform the mapping through a process known as inverse transform sampling. Let u be a variate drawn from Uniform$(0, 1)$, let $F_X(x)$ be the CDF for a prior distribution $f_X(x)$, and let $F_X^{-1}(x)$ be the functional inverse of the prior CDF. $F_X^{-1}(u)$ transforms the uniform variate into a variate drawn from $f_X(x)$.

4. Thermodynamic Integration with Binary Slice Sampling

Goggans and Chi [2] do not specify how to carry out step 5, the refreshing of the population of samples, but their reference implementation takes inspiration from BayeSys by Skilling [3]. While BayeSys uses several different MCMC "engines" to carry out its sampling, the reference implementation of Goggans and Chi [2] uses just two: binary slice sampling and leapfrog sampling. A pseudo-code listing of this procedure is shown in Algorithm 2.

Algorithm 2 Thermodynamic integration with binary slice sampling

procedure TI($P, M, S, N, C, B, W, data$)
  Inputs: P–Number of parameters, M–Number of chains steps, S–Number of slice sampling steps, N–New origin probability, C–Number of chains, B–Number of bits per parameter, W–Ratio to control adaptive annealing, $data$–Data
  for $m \leftarrow 1, C$ **do**
    $X \leftarrow \text{RandInt}(0, 2^{PB} - 1)$
    $\alpha^m \leftarrow \text{LineToAxes}(X, B, P)$
    $E_m^* \leftarrow \text{Energy}(\alpha^m, data)$
  end for
  $i \leftarrow 1$
  Compute $\langle E^* \rangle_i$
  $\beta_1 \leftarrow \min\{\log(W)/[\max(E^*) - \min(E^*)], 1\}$
  $w \leftarrow \exp(-\beta_1 E^*)$
  $\text{ImportanceSampling}(w, \alpha, E^*, C)$
  while $\beta_i > 0$ and $\beta_i < 1$ **do**
    for $i \leftarrow 1, M$ **do**
      for $m \leftarrow 1, C$ **do**
        $\text{BinarySliceSampling}(\alpha^m, E_m^*, B, C, P, S, N, \beta_i, data)$
      end for
      $\text{LeapFrog}(\alpha, E^*, B, C, P, data)$
    end for
    $i \leftarrow i + 1$
    $\Delta\beta \leftarrow \log(W)/[\max(E^*) - \min(E^*)]$
    $\beta_i \leftarrow \min(\beta_{i-1} + \Delta\beta, 1)$
    if $\beta_{i-1} + \Delta\beta > 1$ **then**
      $\Delta\beta \leftarrow 1 - \beta_{i-1}$
    end if
    $w \leftarrow \exp(-\Delta\beta E^*)$
    $\text{ImportanceSampling}(w, \alpha, E^*, C)$
  end while
  Estimate Equation (9) using trapezoid rule and $\{\beta_i\}$ and $\{\langle E^* \rangle_i\}$
end procedure

Binary slice sampling [15] is an integer-based adaptation of slice sampling by Neal [16]. Slice sampling provides a procedure for sampling distribution $f(x)$. Its procedure for moving from point $x_0 \in \mathcal{R}^n$ to another point $x_1 \in \mathcal{R}^n$ in a way that maintains detailed balance under distribution $f(x)$ is described below.

A new variable y is drawn uniformly from the interval $(0, f(x_0))$. The portions of the distribution $f(x)$ that are greater than this value y are considered part of the "slice" to be sampled. A stepping-out procedure if performed from x_0 to find points that are beyond the edges of the slice, then a stepping-in procedure is performed to find bounds that contain all or most of the slice. Once these bounds are obtained, the area defined is sampled uniformly to find the new point, x_1. This procedure is straightforward only in one dimension, though N-dimensional extensions do exist.

Skilling and MacKay [15]'s binary slice sampling follows this procedure, but it uses integer values for x and bit operations to perform the stepping maneuvers and random sampling. A pseudo-code implementation of binary slice sampling is shown in Algorithm 3.

Algorithm 3 Binary slice sampling

function BINARYSLICESAMPLING($\alpha, E^*, B_{in}, C, P, S, N, \beta, data$)
  $B \leftarrow 2^{PB_{in}}$
  $X_{orig} \leftarrow$ RANDINT$(0, B)$
  $\alpha_{orig} \leftarrow$ LINETOAXES(X_{orig}, B_{in}, P)
  $\alpha_i \leftarrow \alpha$
  for $i \leftarrow 1, S$ **do**
    if RAND$(0,1) < N$ **then**
      $X_{orig} \leftarrow$ RANDINT$(0, B)$
      $\alpha_{orig} \leftarrow$ LINETOAXES(X_{orig}, B_{in}, P)
    end if
    $\alpha_i \leftarrow (\alpha_i - \alpha_{orig}) \mod 2^{B_{in}}$
    $X \leftarrow$ AXESTOLINE(α_i, B_{in}, P)
    $U \leftarrow$ RANDINT$(0, B)$
    $y \leftarrow \beta$ENERGY$(\alpha, data) - \log($RAND$(0,1))$
    $l \leftarrow PB_{in}$
    $E^{*\prime} \leftarrow \infty$
    while $\beta E^{*\prime} > y$ and $l > 1$ **do**
      $N \leftarrow$ RANDINT$(0, 2^l)$
      $X' \leftarrow (\{[(X - U) \mod B] \oplus N\} + U) \mod B$
      $\alpha_i \leftarrow$ LINETOAXES(X', B_{in}, P)
      $\alpha_i \leftarrow (\alpha_i + \alpha_{orig}) \mod 2^{B_{in}}$
      $E^{*\prime} \leftarrow$ ENERGY$(\alpha_i, data)$
      $l \leftarrow l - P$
    end while
    $\alpha \leftarrow \alpha_i$
  end for
end function

This implementation uses a space-filling curve to map multi-dimensional coordinates to a single large integer value, allowing binary slice sampling to be used to sample multi-dimensional distributions without any modification.

It should be noted that the reference TI implementation by Goggans and Chi [2], BayeSys by Skilling [3], and our TI-based methods all omit the stepping-out portion of slice sampling. Instead of first stepping out, our TI-based methods begin by setting the slice width as the maximum range allowed by the prior and step in from there.

This TI implementation also uses leapfrog sampling to help shuffle the sample population more effectively. Leapfrog sampling uses samples' neighbors to generate new trial positions, and it works as follows. The population is sorted according to each sample's line (Hilbert or Z-order) index. For each sample in the sorted list, the left and right neighbors are determined. The midpoint between the two neighbor samples in real parameter space is found, and the sample is reflected across that midpoint. If

the new position is still between the left and right neighbors, a Metropolis acceptance test is carried out, and the new position is either accepted or rejected. A pseudo-code implementation of the leapfrog sampling method is shown in Algorithm 4.

Algorithm 4 Leapfrog sampling function

procedure LEAPFROG($\alpha, E^*, B, C, P, data$)
  for $m \leftarrow 1, C$ **do**
    $X_m \leftarrow$ AXESTOLINE(α_m, B, P)
  end for
  Sort X
  for $m \leftarrow 1, C$ **do**
    $\alpha_m \leftarrow$ LINETOAXES(X_m, B, P)
    $E^*_m \leftarrow$ ENERGY($\alpha_m, data$)
  end for
  for $m \leftarrow 1, C$ **do**
    $\alpha cur \leftarrow \alpha_m$
    if $m = 1$ **then**
      $l \leftarrow \alpha_C$
    else
      $l \leftarrow \alpha_{m-1}$
    end if
    if $m = C$ **then**
      $r \leftarrow \alpha_1$
    else
      $r \leftarrow \alpha_{m+1}$
    end if
    $Xl \leftarrow$ AXESTOLINE(l, B, P)
    $Xr \leftarrow$ AXESTOLINE(r, B, P)
    $\alpha new \leftarrow (l + r - \alpha cur) \mod 2^B$
    $Xnew \leftarrow$ AXESTOLINE($\alpha new, B, P$)
    if ($m = 1$ and ($Xnew > Xl$ or $Xnew < Xr$)) or ($m = C$ and ($Xnew > Xl$ or $Xnew < Xr$)) or ($m > 1$ and $m < C$ and $Xnew > Xl$ and $Xnew < Xr$ **then**
      $Enew \leftarrow$ ENERGY($\alpha new, data$)
      if $Enew < E^*_m$ **then**
        $E^*_m \leftarrow Enew$
        $\alpha_m \leftarrow \alpha new$
      else
        $u \leftarrow$ RAND$(0, 1)$
        if $Enew - E^*_m < -\log(u)$ **then**
          $E^*_m \leftarrow Enew$
          $\alpha_m \leftarrow \alpha new$
        end if
      end if
    end if
  end for
end procedure

Algorithm 2 allows for several BSS steps per leapfrog step to allow for good mixing.

4.1. Space-Filling Curves

A space-filling curve is a continuous and nowhere-differentiable function that maps the unit line to a unit hypercube of arbitrary dimension. When we refer to space-filling curves from here on in this article, we are referring to something related but slightly different: an approximation to a true space-filling curve that takes the form $f : \mathbb{N}_0^1 \rightarrow \mathbb{N}_0^N$ and maps the one-dimensional natural numbers to the N-dimensional natural numbers. This approximation to the space-filling curve allows multidimensional probability distributions to be sampled using one-dimensional sampling techniques, such as binary slice sampling. Specifically, it allows us to evaluate model probabilities for models

with multiple parameters by creating a 1-to-1 mapping from the multidimensional parameter space to a one-dimensional integer index. The ideal space-filling curve for this application would have the following properties:

- Locality. Points that are nearby in $\mathbb{N}_0^N$ should be nearby on the curve index in $\mathbb{N}_0^1$ as well. The converse should be true as well.
- Time-efficiency. The algorithms for performing the mapping between parameter space and curve indexes should be time-efficient.
- Bi-directionality. Algorithms should exist for mapping parameter space to curve indexes and from curve indexes to parameter space.

4.1.1. Hilbert Curve

The discrete approximation of the Hilbert curve (hereafter referred to simply as the Hilbert curve) [17] (Chapter 2), Ref. [18] is a space-filling curve that has good locality properties. If two indexes are consecutive on the Hilbert curve, the points in parameter space that correspond to them are adjacent. There are also bi-directional transform functions available for the Hilbert curve, and these transform functions can be implemented in a time-efficient way. An example 4-bit per dimension Hilbert curve for a two-dimensional parameter space is shown in Figure 1.

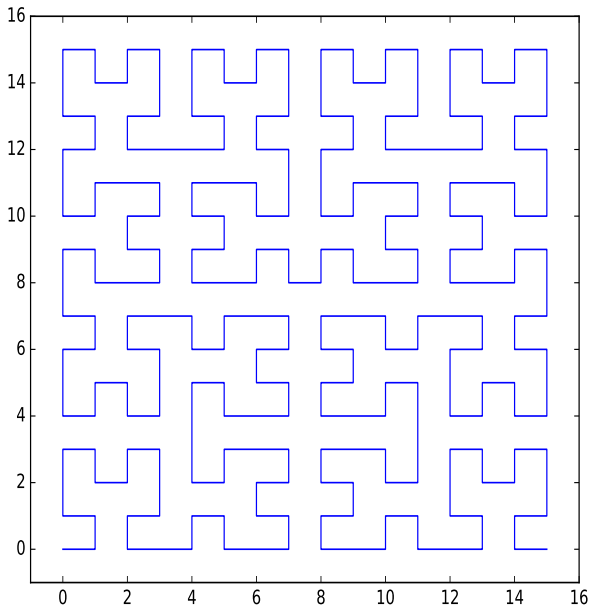

Figure 1. Hilbert curve for two dimensions with four bits per dimension.

A pseudo-code implementation of the Hilbert curve index-to-parameter transformation is shown in Algorithm 5.

Algorithm 5 Hilbert curve line-to-axes function

```
1:  function LineToAxes(Line, B, P)
2:      for i ← 1, P do
3:          linen_{P−i} ← (Line ≫ (B(i − 1))) mod 2^B
4:      end for
5:      M ← 1 ≪ B − 1
6:      for i ← 1, P do
7:          X_i ← 0
8:      end for
9:      q ← 0,    p ← M
10:     for i ← 1, P do
11:         j ← M
12:         while j > 0 do
13:             if linen_i ∧ j then
14:                 X.q ← X.q ∨ p
15:             end if
16:             q ← q + 1
17:             if q = n then
18:                 q ← 0,    p ← p ≫ 1
19:             end if
20:             j ← j ≫ 1
21:         end while
22:     end for
23:     t ← X_P ≫ 1
24:     for i ← P, 2 do
25:         X_i ← X_i ⊕ X_{i−1}
26:     end for
27:     X_1 ← X_1 ⊕ t,    M ← 2 ≪ (B − 1),    Q ← 2
28:     while Q ≠ M do
29:         P ← Q − 1
30:         for i ← P, 2 do
31:             if X_i ∧ Q then
32:                 X_1 ← X_1 ⊕ P
33:             else
34:                 t ← (X_1 ⊕ X_i) ∧ P,    X_1 ← X_1 ⊕ t,    X_i ← X_i ⊕ t
35:             end if
36:         end for
37:         if X_1 ∧ Q then
38:             X_1 ← X_1 ⊕ P
39:         end if
40:         Q ← Q ≪ 1
41:     end while
42:     return X
43: end function
```

4.1.2. Z-Order Curve

The Z-order curve [17] (Chapter 5), Ref. [19] (also known as the Lebesgue curve or Morton curve) is a space-filling curve that maintains locality somewhat less well than the Hilbert curve but otherwise fulfills our requirements for a space-filling curve. Importantly, its transformation algorithms are faster than those for the Hilbert curve. In order to transform from the N-dimensional parameter space to the one-dimensional, Z-order curve, the bits of the integer coordinates for each dimension are interleaved. If the parameter space has three dimensions and each coordinate axis is represented by a 4-bit integer, the resulting Z-order curve representation will be a 12-bit integer. An example of this bit-interleaving is shown in Figure 2, in which each letter represents a binary digit, demonstrates the bit interleaving described above.

```
Z-order index       Axes coordinates

adgj
abcdefghijkl  <-->  behk
cfil
```

Figure 2. Example of Z-order bit-interleaving for three dimensions with four bits per dimension.

An example of a Z-order curve for two dimensions with four bits per dimension is shown in Figure 3.

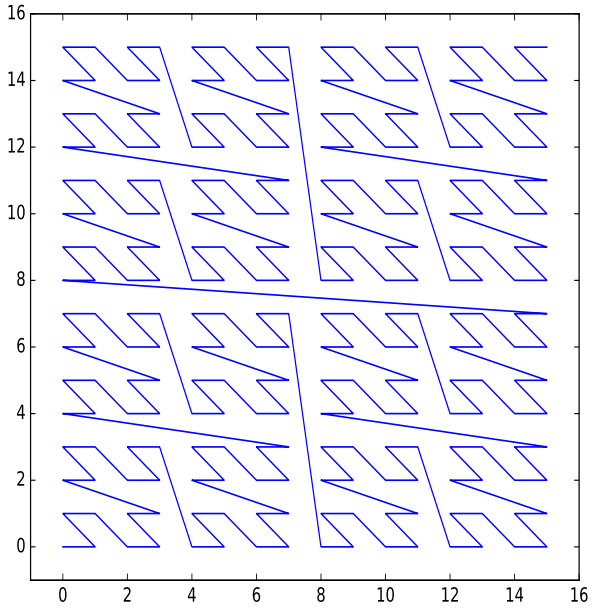

Figure 3. Z-order curve for two dimensions with four bits per dimension.

The simple way to perform the Z-order mapping is to loop over the bit and axis indexes and place each bit where it needs to be individually. However, this method does not provide a time complexity improvement over the Hilbert curve transform functions. A cleverer, bitmask-based approach exists, and it is documented in several places online. The most thorough description of an algorithm for generating the necessary bitmasks for arbitrary numbers of dimensions and bits per dimension is given in a Stackoverflow answer by user Gabriel [20]. Gabriel also describes the general method by which the mapping is performed using the bitmasks, but he does not provide algorithms for doing so. The following list outlines the basic procedure for mapping from the Z-order index to axes coordinates:

1. Generate bitmasks based on number of bits b and number of parameters n.
2. AND the first mask with the Z-order integer to select only every n bits.
3. Loop over each mask. For the ith mask, XOR the Z-order integer with itself shifted to the right by i, then mask the result.
4. Shift the original Z-order integer to the right by 1, then repeat the above from step 2 for each dimension.

A pseudo-code implementation for the bitmask computation function is shown in Algorithm 6, and a pseudo-code implementation for the bitmask-based line-to-axes transformation function is shown in Algorithm 7.

Algorithm 6 Z-order curve mask computation function

function COMPUTEBITMASK(B, P)
  $P \leftarrow P - 1$
  for $i \leftarrow 1, B$ **do**
    $bd_i \leftarrow (i+1)P$ ▷ Stored as binary strings, with the leftmost two bits discarded
  end for
  $maxLength \leftarrow$ length of longest string in bd
  $moveBits \leftarrow$ empty list
  for $i \leftarrow 1, maxLength$ **do**
    Append an empty list to $moveBits$
    for $j \leftarrow 1, P$ **do**
      if LENGTH(bd_j) $\geq i$ **then**
        if The ith bit from the end of bd_j is 1 **then**
          Append j to $moveBits_i$
        end if
      end if
    end for
  end for
  for $i \leftarrow 1, B$ **do**
    $bitPositions_i \leftarrow i$
  end for
  $maskOld \leftarrow (1 \ll B) - 1$
  $bitmasks \leftarrow$ empty list
  for $i \leftarrow$ LENGTH($moveBits$), 1 **do**
    if LENGTH($moveBits_i$) > 0 **then**
      $shifted \leftarrow 0$
      for $bitIdxToMove \in moveBits_i$ **do**
        $shifted \leftarrow shifted \vee (1 \ll bitPositions_{bitIdxToMove})$
        $bitPositions_{bitIdxToMove} \leftarrow bitPositions_{bitIdxToMove} + 2^i$
      end for
      $nonshifted \leftarrow \neg shifted \wedge maskOld$
      $shifted \leftarrow shifted \ll 2^i$
      $maskNew \leftarrow shifted \vee nonshifted$
      Append $maskNew$ to $bitmasks$
      $maskOld \leftarrow maskNew$
    end if
  end for
  return $bitmasks$
end function

Algorithm 7 Z-Order curve line-to-axes function

```
1:  function LineToAxes(z, B, P)
2:      if P = 1 then
3:          return z
4:      end if
5:      masks ← ComputeBitMask(B, P)                ▷ Call only once for each B and P
6:      Pop final entry from masks list into first
7:      Reverse the masks list
8:      Append 1 ≪ B to masks
9:      minshift ← P − 1
10:     for i ← 1, P do
11:         zz ← z ≫ i
12:         zz ← zz ∧ first
13:         shift ← minshift
14:         for mask ∈ masks do
15:             zz ← (zz ⊕ (zz ≫ shift)) ∧ mask
16:             shift ← shift ≪ 1
17:         end for
18:         α_i ← zz
19:     end for
20:     return α
21: end function
```

4.2. Parallel Implementation

In implementations of this binary slice sampling and space-filling curve-based technique, most of the computation time is taken up by the line-to-axes and axes-to-line operations. Because the vast majority of the invocations of these operations occur during the binary slice sampling routine itself, the for loop on lines 15, 16, and 17 in Algorithm 2 is a natural place to parallelize the algorithm. Parallelization at this point allows for much larger sample populations (C), which improves the accuracy of the evidence estimate. The TI-BSS example results in Section 6 are produced with a parallel implementation in this vein.

5. Thermodynamic Integration with Stan

The evolution of our approach proceeded as follows. John Skilling released the latest version of BayeSys [3] in the late 2000s. This software used an adaptively annealed thermodynamic integration framework combined with binary slice sampling on the Hilbert curve to perform jump-diffusion sampling. This approach is useful for doing model comparison in the case that the model equation can be decomposed into several identical components.

Soon afterward, Paul Goggans and Ying Chi wanted to apply this adaptively-annealed thermodynamic integration framework to a model comparison problem in which the model equation was not separable. They developed a method, TI-BSS-H [2] that used a very similar approach to BayeSys, but rather than using the thermodynamic integration framework to facilitate jump diffusion sampling, their method directly estimated the evidence for a particular model.

When our interest again returned to thermodynamic integration in 2017, we published a method, TI-BSS-Z [19] that replaced the Hilbert curve component of the TI-BSS-H approach with the Z-order curve. This change was made with the goal of increasing the speed of the method. Ultimately, this approach sacrificed too much accuracy for us to use it on a regular basis.

More recently, we have taken a different tack in further developing this approach. We have gone beyond replacing the space-filling curve used for BSS and instead replaced the BSS component all together with a more modern MCMC method. A recent survey of available MCMC tools turned up MC Stan (or simply Stan), developed by Carpenter et al. [4], as the state-of-the-art, general purpose MCMC method for performing Bayesian parameter estimation.

Stan uses the NUTS [6] as the basis for its sampling functions. NUTS is based on HMC [21], which uses the gradient of the log-likelihood function to more efficiently explore the posterior distribution. NUTS improves upon HMC by automatically choosing optimal values for HMC's tunable method parameters. NUTS has been shown to sample complex distributions effectively. We sought to build an improved thermodynamic integration implementation by using Stan instead of binary slice sampling and leapfrog sampling to refresh the sample population at each temperature within TI. The result, TI-Stan, is described in this section.

The TI-Stan algorithm is shown in Algorithm 8.

Algorithm 8 Thermodynamic integration with Stan

procedure TI($P, S, C, W, data$)
  Inputs: P–Number of parameters, S–Number of Stan iterations per temperature, C–Number of chains, W–Ratio to control adaptive annealing, $data$–Data
  for $m \leftarrow 1, C$ **do**
    $X \leftarrow \text{RandInt}(0, 2^{PB} - 1)$
    $\alpha^m \leftarrow \text{LineToAxes}(X, B, P)$
    $E^*_m \leftarrow \text{Energy}(\alpha^m, data)$
  end for
  $i \leftarrow 1$
  Compute $\langle E^* \rangle_i$
  $\beta_1 \leftarrow \min\{\log(W)/[\max(E^*) - \min(E^*)], 1\}$
  $w \leftarrow \exp(-\beta_1 E^*)$
  $\text{ImportanceSampling}(w, \alpha, E^*, C)$
  while $\beta_i > 0$ and $\beta_i < 1$ **do**
    for $m \leftarrow 1, C$ **do**
      $\text{StanSampling}(\alpha^m, E^*_m, C, P, S, \beta_i, data)$
    end for
    $i \leftarrow i + 1$
    $\Delta\beta \leftarrow \log(W)/[\max(E^*) - \min(E^*)]$
    $\beta_i \leftarrow \min(\beta_{i-1} + \Delta\beta, 1)$
    if $\beta_{i-1} + \Delta\beta > 1$ **then**
      $\Delta\beta \leftarrow 1 - \beta_{i-1}$
    end if
    $w \leftarrow \exp(-\Delta\beta E^*)$
    $\text{ImportanceSampling}(w, \alpha, E^*, C)$
  end while
  Estimate Equation (9) using trapezoid rule and $\{\beta_i\}$ and $\{\langle E^* \rangle_i\}$
end procedure

Our implementation is in Python, so we made use of the PyStan interface to Stan, developed by the Stan Development Team [5]. Stan defines its own language for defining statistical models, which allows it to efficiently compute the derivatives needed for HMC via automatic differentiation. For a particular problem, it is therefore necessary to write a Stan file that contains the Stan-formatted specification of the model, in addition to the pure-Python energy functions necessary for TI-BSS. Once one is familiar with the simple Stan language, this additional programming cost becomes trivial compared to the time savings achieved by using this method instead of TI-BSS.

6. Examples

In this section, we demonstrate the performance of the TI-Stan method using four problems as examples. These include an artificial, highly multi-modal likelihood function, a simulated spectrum analysis problem, a twin Gaussian shell problem in 20 and 30 dimensions, and the estimation of the thermodynamic partition function for an ideal gas.

6.1. Eggcrate Likelihood

The first example is the "eggcrate" toy problem from [22]. The posterior density in this problem is highly multimodal, so that a large sample population is necessary to estimate the evidence accurately and to sample the entire density.

The eggcrate function has two independent parameters, with joint prior

$$\pi(\mathbf{\Theta}) = \left(\frac{1}{10\pi}\right)^2 \mathbb{1}_{[0,10\pi]}(\Theta_1)\mathbb{1}_{[0,10\pi]}(\Theta_2). \tag{17}$$

The likelihood function is

$$\mathcal{L}(\mathbf{\Theta}) = \exp\left\{\left[2 + \cos\left(\frac{\Theta_1}{2}\right)\cos\left(\frac{\Theta_2}{2}\right)\right]^5\right\}. \tag{18}$$

In this case, the log-likelihood is more useful for visualization and the numerical dynamic range:

$$\log\mathcal{L}(\mathbf{\Theta}) = \left[2 + \cos\left(\frac{\Theta_1}{2}\right)\cos\left(\frac{\Theta_2}{2}\right)\right]^5. \tag{19}$$

Feroz et al. [22] provide an evidence value of $\log\mathcal{Z} = 235.88$ using numerical integration over a fine grid. Upon applying Bayes' theorem, the posterior distribution for the parameters $\mathbf{\Theta}$ is

$$\mathcal{P}(\mathbf{\Theta}) = \frac{\exp\left\{\left[2 + \cos\left(\frac{\Theta_1}{2}\right)\cos\left(\frac{\Theta_2}{2}\right)\right]^5\right\}\left(\frac{1}{10\pi}\right)^2 \mathbb{1}_{[0,10\pi]}(\Theta_1)\mathbb{1}_{[0,10\pi]}(\Theta_2)}{\exp(235.88)}. \tag{20}$$

6.2. Detection of Multiple Stationary Frequencies

In the second example, we want to estimate the number of stationary frequencies present in a signal as well as the value of each frequency. This problem is similar to the problem of multiple stationary frequency estimation in [23] (Chapter 6), with the additional task of determining the number of stationary frequencies present. This example demonstrates the value of the present parallel nested sampling method. Differences among log-evidence values for models containing either the most probable number of frequencies or more tend to be small, meaning that a precise estimate of these log-evidence values is essential for the task of determining the most probable model.

Each stationary frequency (j) in the model is determined by three parameters: the in-phase amplitude (A_j), the quadrature amplitude (B_j), and the frequency (f_j). Given J stationary frequencies, the model at time step t_i takes the following form:

$$g\left(t_i;\mathbf{\Theta}\right) = \sum_{j=1}^{J} A_j\cos\left(2\pi f_j t_i\right) + B_j\sin\left(2\pi f_j t_i\right), \tag{21}$$

where $\mathbf{\Theta}$ is the parameter vector

$$\mathbf{\Theta} = \left[A_1\, B_1\, f_1\, \cdots\, A_J\, B_J\, f_J\right]^T.$$

For the purposes of this example, the noise variance used to generate the simulated data is known, and we consequently use a Gaussian likelihood function,

$$\mathcal{L}(\mathbf{\Theta}) = \prod_{i=1}^{I}\exp\left\{-\frac{\left[g\left(t_i;\mathbf{\Theta}\right) - d_i\right]^2}{2\sigma^2}\right\}, \tag{22}$$

for I simulated data d_i and noise variance σ^2. The log-likelihood function is then

$$\log \mathcal{L}(\boldsymbol{\Theta}) = -\sum_{i=1}^{I} \frac{[g(t_i; \boldsymbol{\Theta}) - d_i]^2}{2\sigma^2}. \tag{23}$$

Each model parameter is assigned a uniform prior distribution with limits as shown in Table 1.

Table 1. Prior bounds for multiple stationary frequency model parameters.

	Lower Bound	Upper Bound
A_j	-2	2
B_j	-2	2
f_j	0 Hz	6.4 Hz

Our test signal is a sum of two sinusoids, and zero-mean Gaussian noise with variance $\sigma^2 = 0.01$. This signal is sampled at randomly-spaced instants of time, in order to demonstrate that this time-domain method does not require uniform sampling to perform spectrum estimation. Bretthorst [24] demonstrates that the Nyquist critical frequency in the case of nonuniform sampling is $1/2\Delta T'$, where $\Delta T'$ is the dwell time. The dwell time is not defined for arbitrary-precision time values as used in this example, so we must choose another limiting value. A more conservative limit is given by $1/10\Delta T_{\text{avg}}$, where ΔT_{avg} is the average spacing between time steps, $1/64$ s. This formulation yields a prior maximum limit of 6.4 Hz, as shown in Table 1. The parameters used to generate the simulated data are shown in Table 2.

Table 2. Parameters used to generate simulated signal.

j	A_j	B_j	f_j (Hz)
1	1.0	0.0	3.1
2	1.0	0.0	5.9

The samples from the signal with noise are shown in Figure 4.

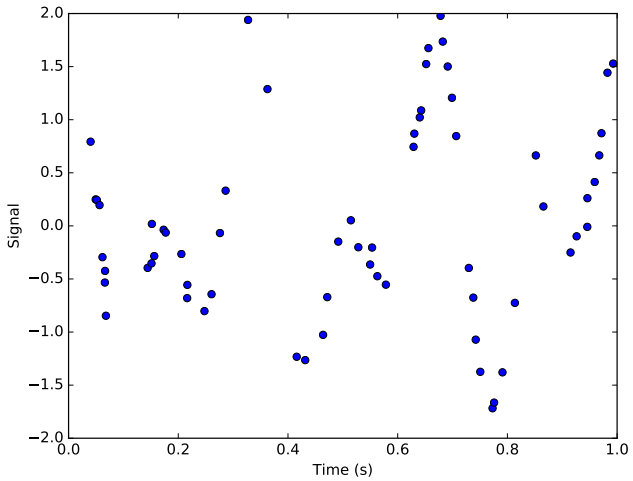

Figure 4. The simulated signal. The points represent the non-uniformly sampled points from the original signal corrupted by Gaussian noise.

6.3. Twin Gaussian Shells

The third example is the twin Gaussian shell problem, also from [22]. In [22], the authors present results for this problem in up to 30 dimensions. Handley et al. [25] also use this problem in 100 dimensions to test their algorithm. This problem presents a few interesting challenges to our thermodynamic integration implementation. Because the likelihood takes the form of a thin, curved density whose mass centers on a hyper-spherical shell, exploration of the posterior at high inverse temperature values is difficult. The bimodal nature of the problem also challenges the posterior exploration process. Finally, the examples we explore are high-dimensional to the point that standard numerical integration techniques would be useless.

The likelihood function in the twin Gaussian shells problem takes the form,

$$\mathcal{L}(\boldsymbol{\Theta}) = \frac{1}{\sqrt{2\pi}w_1} \exp\left[-\frac{(|\boldsymbol{\Theta} - \mathbf{c}_1| - r_1)^2}{2w_1^2}\right] + \frac{1}{\sqrt{2\pi}w_2} \exp\left[-\frac{(|\boldsymbol{\Theta} - \mathbf{c}_2| - r_2)^2}{2w_2^2}\right]. \tag{24}$$

Following [22], we set the parameters as follows: $w_1 = w_2 = 0.1$, $r_1 = r_2 = 2$, $\mathbf{c}_1 = [-3.5, 0, \cdots, 0]^T$, and $\mathbf{c}_2 = [3.5, 0, \cdots, 0]^T$. We use a uniform prior over the hypercube that spans $[-6, 6]$ in each dimension. Figure 5 shows a pseudo-color plot of a two-dimensional twin Gaussian shell with parameters and prior range as described previously.

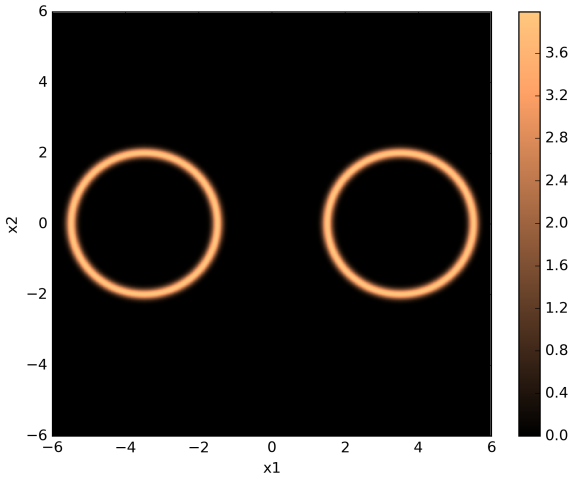

Figure 5. Pseudo-color plot of a two-dimensional twin Gaussian shell with $w_1 = w_2 = 0.1$, $r_1 = r_2 = 2$, $\mathbf{c}_1 = [-3.5, 0]^T$, and $\mathbf{c}_2 = [3.5, 0]^T$. The color values correspond to likelihood values.

6.4. Ideal Gas Partition Function

The final example is inspired by Section 31.3 of [26]. In this example problem, we wish to estimate a function proportional to the thermodynamic partition function of an ideal gas with $N_p = N/3$ particles,

$$Z = \int \exp\left[-\mathbf{p}^2/(2\sigma^2)\right] \mathrm{d}^N p, \tag{25}$$

where p is an N-dimensional vector of the particles' momenta, and $\sigma^2 = mk_BT$ is the product of the mass, Boltzmann constant, and temperature. Following [26], we assign a uniform prior distribution to the momenta $\mathbf{p}$ with support on an $(N-1)$-sphere $S(R)$ of radius R, set such that the error in the evaluation of the integral is below an acceptable threshold,

$$R = 2\sqrt{N}\sigma. \tag{26}$$

The volume of $S(R)$ is

$$V_N(R) = \frac{R^N \pi^{N/2}}{\Gamma\left(\frac{N}{2}+1\right)}. \tag{27}$$

Under this prior, we can rewrite the partition function in the form

$$Z = V_N(R) \underbrace{\int_{S(R)} \exp\left[-\mathbf{x}^2/(2\sigma^2)\right] p_0(\mathbf{x})\, \mathrm{d}^N x}_{\tilde{Z}}, \tag{28}$$

where $p_0(\mathbf{x})$ is the prior, $\exp\left[-\mathbf{x}^2/(2\sigma^2)\right]$ is the likelihood, and $\tilde{Z}$ is the function that we will actually estimate and show results for. The exact value is given by

$$\tilde{Z} = \frac{(2\pi\sigma^2)^{N/2}}{V_N(R)}. \tag{29}$$

Substituting Equation (26) into Equation (29), taking the log, and simplifying yields the exact value

$$\log \tilde{Z} = -\frac{N}{2}\log 2 - \frac{N}{2}\log N + \log\Gamma\left(\frac{N}{2}+1\right). \tag{30}$$

We use Equation (30) to judge the accuracy of our numerical results.

6.5. Results

For the first three examples, the settings used for TI-BSS are shown in Table 3, while the settings used for TI-Stan are shown in Table 4. For each example, the user-defined constant W was set to both 1.5 and 2.0. Box-plots are used extensively in this section. In these box-plots, the middle line represents the median value, the box is bounded by the upper and lower quartiles, and the whiskers extend to the range of the data that lies within 1.5 times the inter-quartile range. Any data points past this threshold are plotted as circles.

Table 3. Parameters for TI-BSS examples.

Parameter	Value	Definition
S	200	Number of binary slice sampling steps
M	2	Number of combined binary slice sampling and leapfrog steps
C	256	Number of chains
B	32	Number of bits per parameter in SFC

Table 4. Parameters for TI-Stan examples.

Parameter	Value	Definition
S	200	Number of steps allowed in Stan
C	256	Number of chains

These results for the first three examples were generated on a Google Cloud (Mountain View, CA, USA) instance with 32 virtual Intel Broadwell CPUs and 28.8 GB of RAM.

6.5.1. Eggcrate Likelihood Results

The true value of the log-evidence for this log-likelihood function is known up to five significant digits:

$$\log Z_{\text{true}} = 235.88. \tag{31}$$

A box-plot summarizing the log-evidence estimates over 20 runs each for the TI-Stan, TI-BSS-H, and TI-BSS-Z methods and for each value of W is shown in Figure 6.

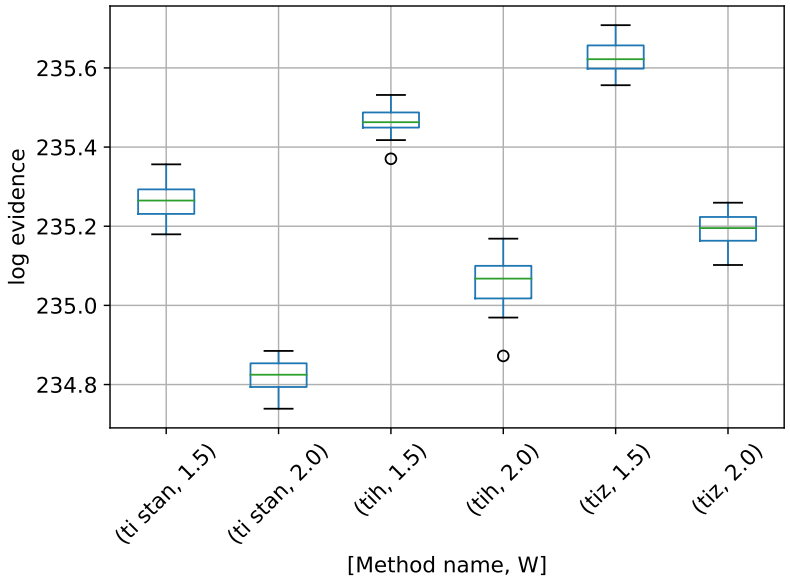

Figure 6. Box-plot of log-evidence for the egg-crate problem for each TI method. TI-Stan with W=xx is denoted by ti stan, xx; TI-BSS-H with W=xx is denoted by tih, xx; and TI-BSS-Z with W=xx is denoted by tiz, xx.

Figure 6 demonstrates that all of the TI methods at these settings underestimate the log-evidence, with TI-BSS-Z with $W = 1.5$ coming the closest. Likely a value of W closer to 1 would yield better results. It is also apparent that the precision in these results do not correlate with the accuracy, suggesting that, for problems with unknown answers, high precision over multiple runs should not be interpreted as a proxy for accuracy.

A box-plot summarizing the run times over 20 runs each for the TI methods is shown in Figure 7.

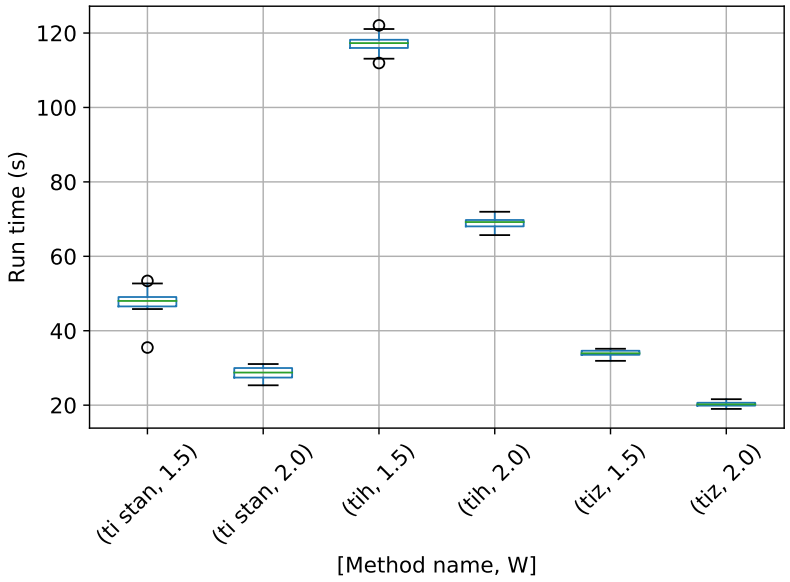

Figure 7. Box-plot of run time in seconds for the egg-crate problem for each TI method. TI-Stan with W=xx is denoted by ti stan, xx; TI-BSS-H with W=xx is denoted by tih, xx; and TI-BSS-Z with W=xx is denoted by tiz, xx.

Figure 7 shows that each method speeds up considerably with a higher value of W. TI-BSS-H took the most time here, with TI-Stan and TI-BSS-Z running much more quickly.

6.5.2. Twin Gaussian Shells' Results

This problem is run in 10 dimensions, 30 dimensions, and 100 dimensions. For the case of 10 dimensions, results are presented for all TI-BSS-H and TI-Stan, but not for TI-BSS-Z. For both values of W, TI-BSS-Z ended early in all its runs, and its estimate of the log-evidence is unusably inaccurate. This likely occurred because the Z-order curve sacrifices some locality for the sake of speed compared to the Hilbert curve. Problems due to the reduced locality likely compound in the case of higher dimensionality. For the cases of 30 and 100 dimensions, results are presented only for TI-Stan because a run of TI-BSS-H took too much time for it to be feasible to complete multiple runs. The Hilbert curve calculations become infeasibly time-consuming in high dimensions.

10-Dimensional Twin Gaussian Shells

We start with the 10-dimensional variant. A box-plot summarizing the log-evidence estimates over 20 runs each for TI-Stan and TI-BSS-H and for each value of W is shown in Figure 8.

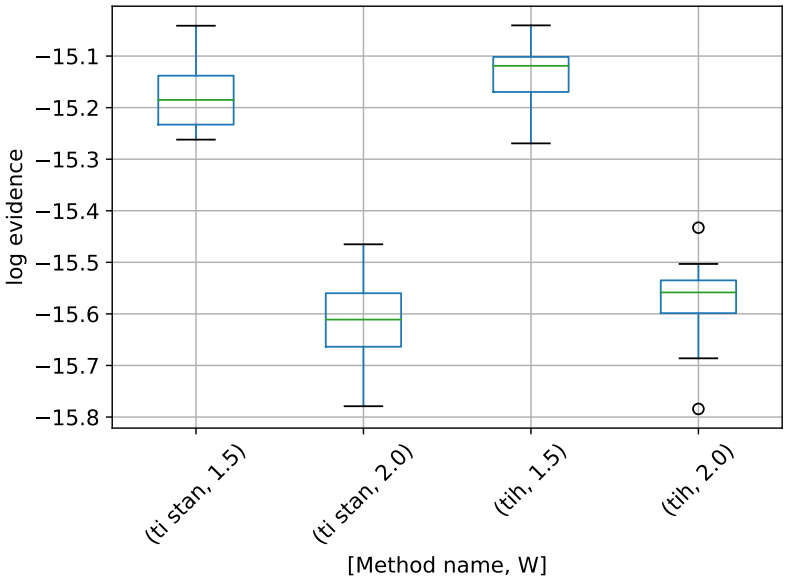

Figure 8. Box-plot of log-evidence for the 10-D twin Gaussian shell problem for TI-Stan and TI-BSS-H. TI-Stan with W=xx is denoted by ti stan, xx and TI-BSS-H with W=xx is denoted by tih, xx.

From [22], the analytical log-evidence for this distribution is -14.59. None of the configurations tested actually reached that value, but the runs using $W = 1.5$ got closest, suggesting that a value of W closer to 1 would perhaps approach the correct value.

A box-plot summarizing the run times over 20 runs each for the TI methods is shown in Figure 9.

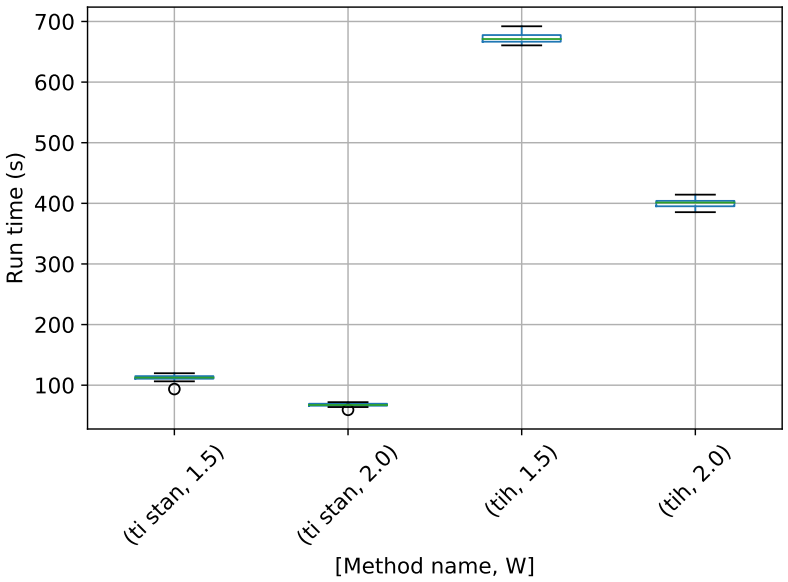

Figure 9. Box-plot of run time in seconds for the 10-D twin Gaussian shell problem for TI-Stan and TI-BSS-H. TI-Stan with W=xx is denoted by ti stan, xx and TI-BSS-H with W=xx is denoted by tih, xx.

Figure 9 shows the same trend with W as in the egg-crate problem, i.e., that the run time drastically increases as W approaches 1. It also shows that TI-BSS-H takes about six times longer, on average, than TI-Stan to compute its estimate of the log-evidence. According to Figure 8, the two methods have comparable accuracy and precision, so this difference in run time illustrates the difficulty the Hilbert curve-based method has with distributions of high dimension.

30-Dimensional Twin Gaussian Shells

Regarding the 30-dimensional variant, the limitations of TI-BSS become even more apparent, as one run could not finish in a reasonable amount of time. TI-Stan, however, has no problem with the 30-D variant. A box-plot summarizing the log-evidence estimates over 20 runs each for TI-Stan at two values of W is shown in Figure 10.

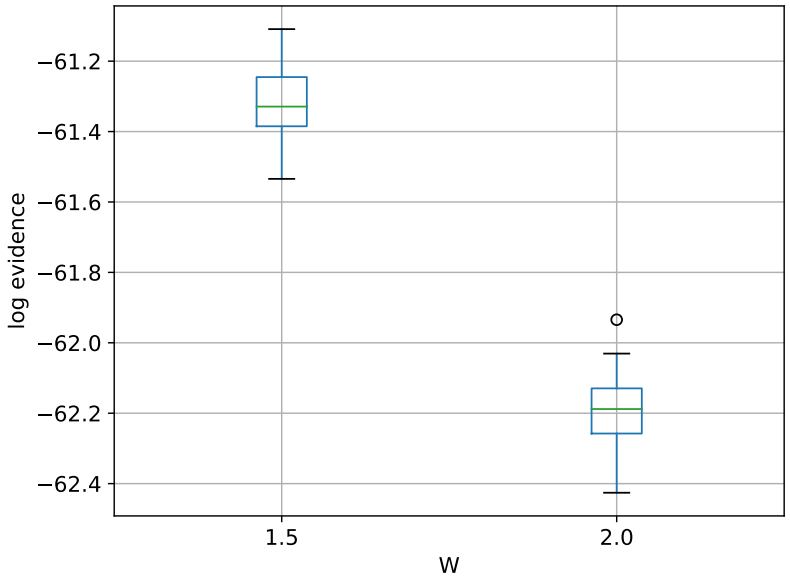

Figure 10. Box-plot of log-evidence for the 30-D twin Gaussian shell problem for TI-Stan.

Again from [22], the analytical log-evidence value is -60.13. Neither value of W allows TI-Stan to come within 1 unit of the correct answer here, and the gap is actually slightly larger here than in the 10-D variant. Again, a smaller value of W would probably be helpful here.

A box-plot summarizing the run times over 20 runs each for TI-Stan is shown in Figure 11.

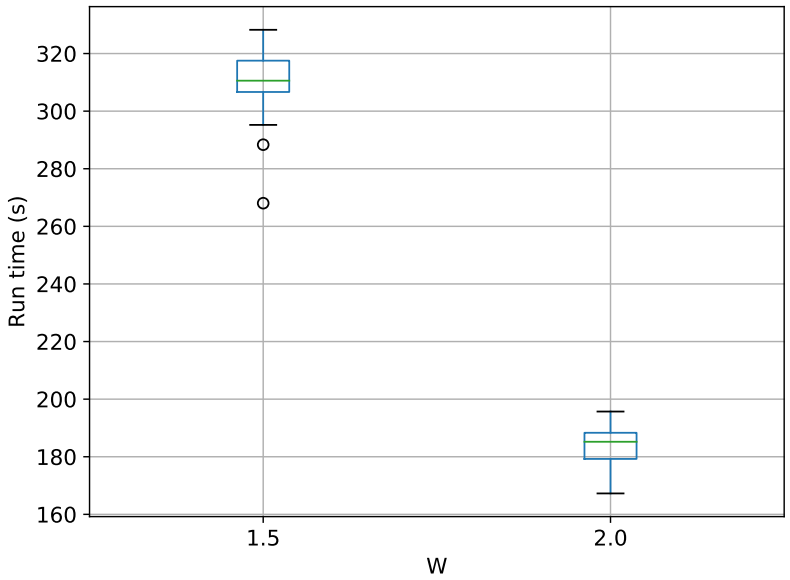

Figure 11. Box-plot of run time in seconds for the 30-D twin Gaussian shell problem for TI-Stan.

The common trend in run time vs W is observed here as well. TI-Stan takes about three times as long on average to compute log-evidence values in the 30-D case compared to the 10-D case, suggesting a possible linear relationship between run time and the number of dimensions for this problem.

100-Dimensional Twin Gaussian Shells

A box-plot summarizing the log-evidence estimates over 20 runs each for TI-Stan at two values of W is shown in Figure 12.

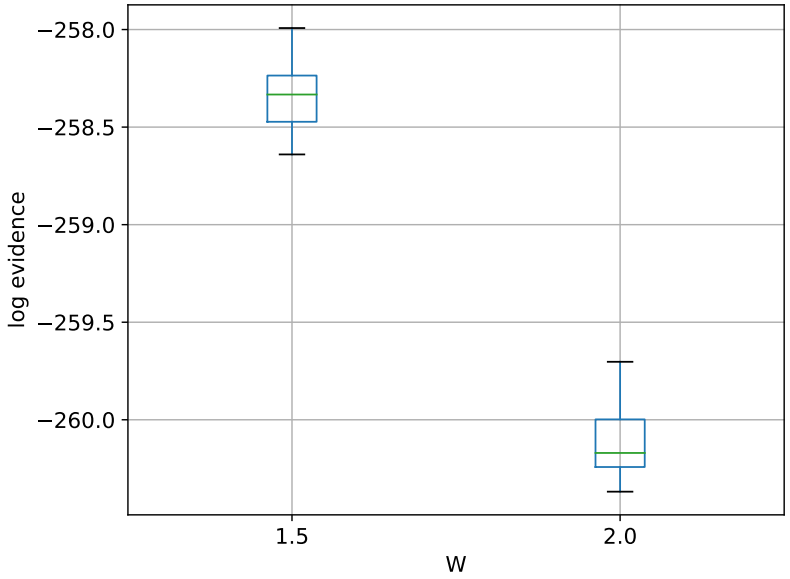

Figure 12. Box-plot of log-evidence for the 100-D twin Gaussian shell problem for TI-Stan.

There is no analytical value available in the literature for this variant of the problem, but the result here seems believable. If it follows the trend of the previous examples, the log-evidence has been underestimated by some margin.

A box-plot summarizing the run times over 20 runs each for TI-Stan is shown in Figure 13.

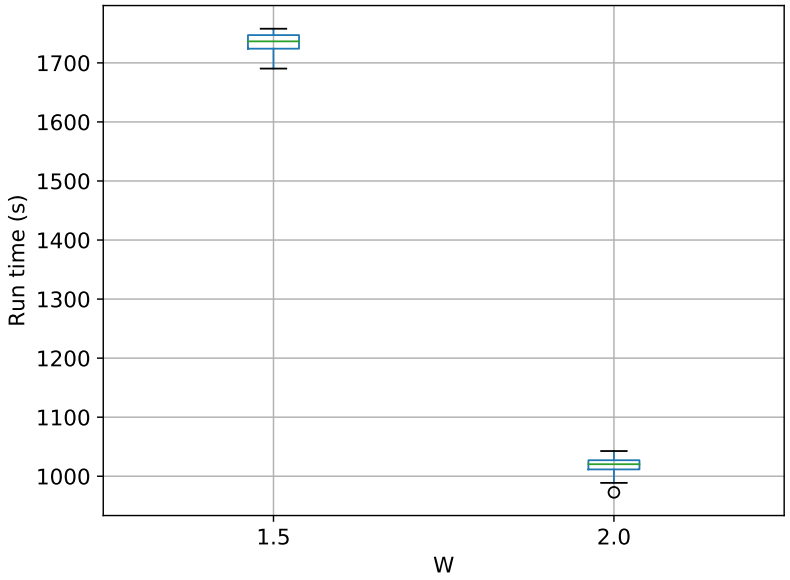

Figure 13. Box-plot of run time in seconds for the 100-D twin Gaussian shell problem for TI-Stan.

The results in Figure 13 disprove the hypothesis that the run time is related linearly to the number of parameters in this problem. The established relationship between run time and W continues here. This example is only a toy problem, but these results suggest that TI-Stan could be useful in problems with actual data and very high-dimensional models.

6.5.3. Detection of Multiple Stationary Frequencies' Results

The final example problem is the multiple stationary frequencies detection problem described in Section 6.2. Box-plots of log-evidence values for a model assuming one, two, and three frequencies present are shown in Figures 14–16. For the models with one and three frequencies present, results are shown for TI-Stan, TI-BSS-H, and TI-BSS-Z. For the model with two frequencies present (the model also used to generate the test signal), results for TI-BSS-Z are not shown. As in the Gaussian shell problems, TI-BSS-Z ended early here and did not arrive at a reasonable result. Here, as well, the problem likely lies in the decreased locality of the Z-order curve when compared with the Hilbert curve. In addition, when evaluating the model used to generate the original data (in other words, the "true" model), the likelihood function will be sharper than when evaluating the other candidate models. This, combined with the decreased locality of the Z-order curve, likely led to its failure in this case.

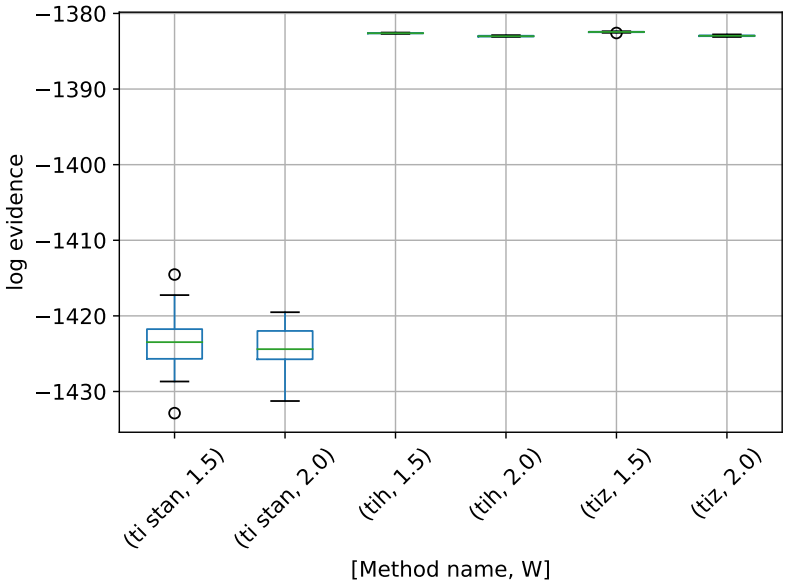

Figure 14. Box-plot of log-evidence for the one stationary frequency model for TI-Stan, TI-BSS-H, and TI-BSS-Z, for two values of W. TI-Stan with W=xx is denoted by ti stan, xx; TI-BSS-H with W=xx is denoted by tih, xx; and TI-BSS-Z with W=xx is denoted by tiz, xx.

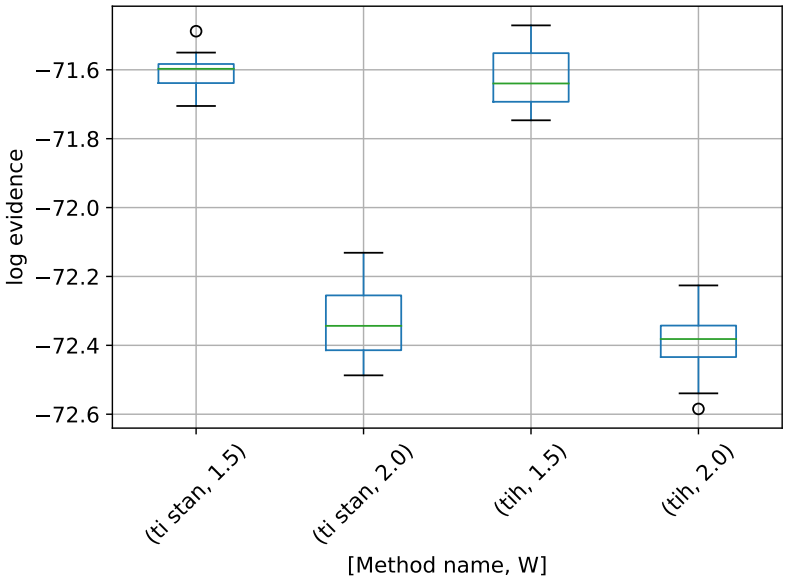

Figure 15. Box-plot of log-evidence for the two stationary frequency model for TI-Stan and TI-BSS-H, for two values of W. TI-Stan with W=xx is denoted by ti stan, xx and TI-BSS-H with W=xx is denoted by tih, xx.

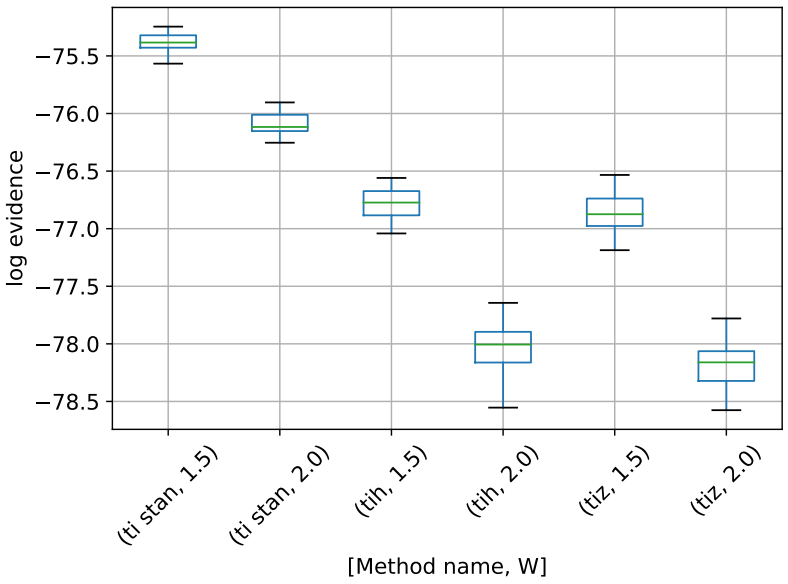

Figure 16. Box-plot of log-evidence for the three stationary frequency model for TI-Stan, TI-BSS-H, and TI-BSS-Z, for two values of W. TI-Stan with W=xx is denoted by ti stan, xx; TI-BSS-H with W=xx is denoted by tih, xx; and TI-BSS-Z with W=xx is denoted by tiz, xx.

There are no analytical log-evidence values available for this example. We argue that a method is successful if the model used to generate the data clearly has the highest log-evidence, with a good margin between it and the log-evidence for the other models. There is some significant disagreement among the various methods for the "wrong" models (those with one and three frequencies), but the methods are in much closer agreement for the two-frequency model. For TI-Stan and TI-BSS-H and for both values of W, the two-frequency model is clearly the maximum-log-evidence choice. Even with the variations in the runs, the results do not overlap at any point from model to model, and the closest model-to-model margins are all greater than 2.3, which corresponds to odds of 10.

Box-plots of the run time for models assuming one, two, and three frequencies present are shown in Figures 17–19.

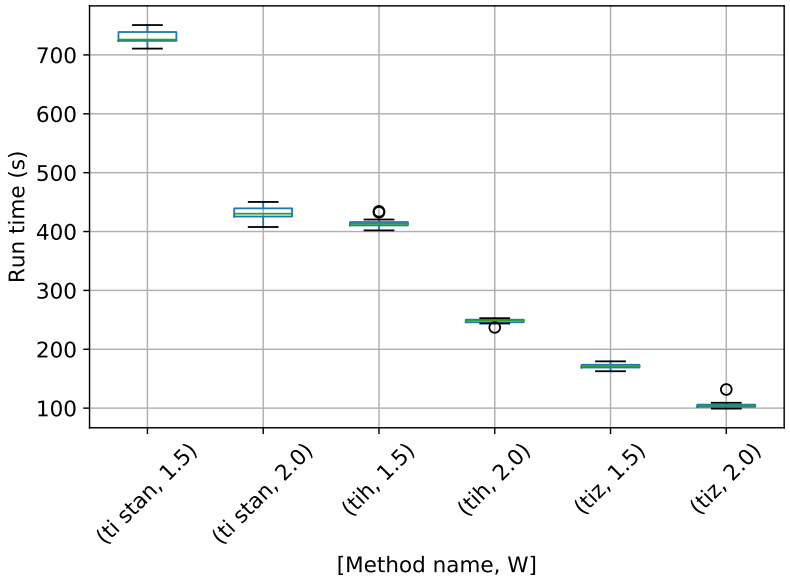

Figure 17. Box-plot of run time for the $J = 1$ stationary frequency model for TI-Stan, TI-BSS-H, and TI-BSS-Z, for two values of W. TI-Stan with W=xx is denoted by ti stan, xx; TI-BSS-H with W=xx is denoted by tih, xx; and TI-BSS-Z with W=xx is denoted by tiz, xx.

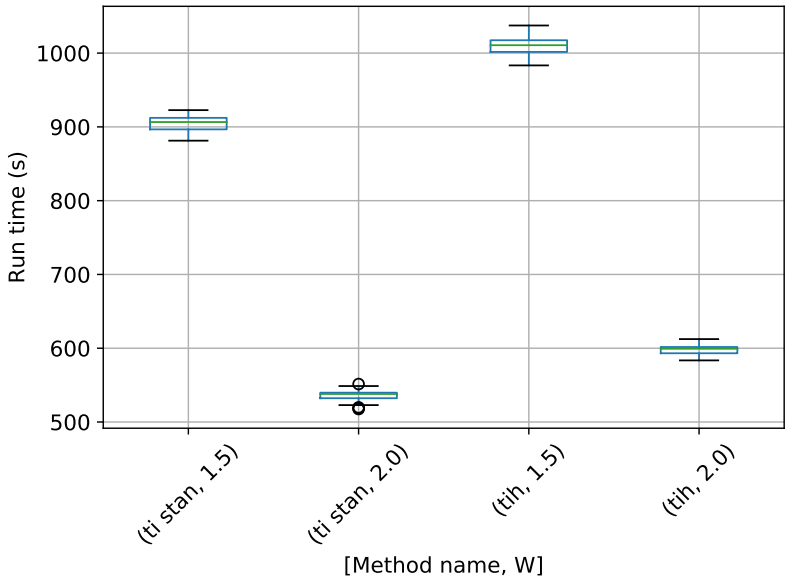

Figure 18. Box-plot of run time for the $J = 2$ stationary frequency model for TI-Stan and TI-BSS-H, for two values of W. TI-Stan with W=xx is denoted by ti stan, xx; and TI-BSS-H with W=xx is denoted by tih, xx.

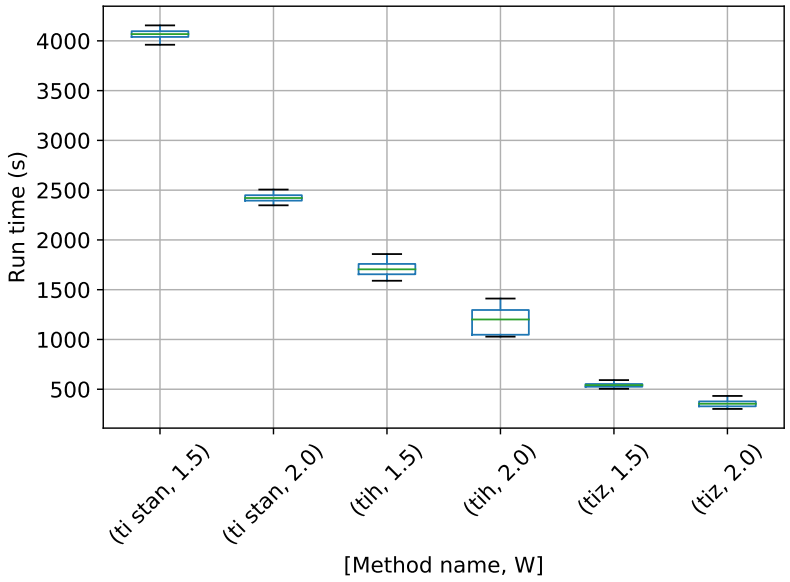

Figure 19. Box-plot of run time for the $J = 3$ stationary frequency model for TI-Stan, TI-BSS-H, and TI-BSS-Z, for two values of W. TI-Stan with W=xx is denoted by ti stan, xx; TI-BSS-H with W=xx is denoted by tih, xx; and TI-BSS-Z with W=xx is denoted by tiz, xx.

In Figure 17, TI-Stan has the greatest run time for both values of W, suggesting that its adaptive sampling process had trouble efficiently sampling distributions based on this high-error model. TI-BSS-H was much faster, and TI-BSS-Z was faster still. In Figure 18, the run times of TI-Stan and TI-BSS-H are comparable. This suggests that TI-Stan was able to more effectively sample the distribution based on the lower-error model. Figure 19 shows a similar pattern in the run times to Figure 17. The fact that this model is able to fit the noise in the data (yielding especially sharp distributions) and the fact that the distribution is increasingly multi-modal as the number of frequencies increases may explain why TI-Stan took a long time to compute a result here.

6.5.4. Ideal Gas Partition Function Results

Results for the ideal gas partition function problem are presented for only the TI-Stan method. This problem requires parameter counts exceeding what TI-BSS can usefully deal with. The method parameters for this problem are different from those used in the previous three examples. For this problem, $S = 20$ number of steps per iteration of Stan were allowed, while $C = 24$ chains were used. A similar Google Cloud instance using 32 Intel Broadwell vCPUS and 28.8 GB of RAM was used here.

Table 5 shows the results for an ideal gas partition function problem. The first column indicates the value of W used by TI-Stan, the second column shows the number of dimensions, the third and fourth columns show statistics for the relative error of the numerical result with respect to the analytic value, the fifth and sixth columns show statistics for the $\log \tilde{Z}$ results, and the seventh column shows the analytic value of $\log \tilde{Z}$ for each value of N as given in Equation (30). Table 6 shows the run time statistics for each value of W and N.

Table 5. Ideal Gas Partition Function $\log \tilde{Z}$ results for TI-Stan for two values of W. Twenty TI-Stan runs were completed for each value of W and N.

W	N	Mean Relative Error	StDev Relative Error	Mean $\log \tilde{Z}$	StDev $\log \tilde{Z}$	Analytic $\log \tilde{Z}$ (30)
1.05	12	0.52%	0.37%	−12.43	0.0565	−12.49
"	102	0.51%	0.20%	−118.20	0.235	−118.81
"	1002	0.62%	0.26%	−1184.16	3.04	−1191.51
1.5	12	2.94%	1.72%	−12.12	0.215	−12.49
"	102	3.39%	1.44%	−114.78	1.71	−118.81
"	1002	4.50%	1.40%	−1137.83	16.73	−1191.51

Table 6. Ideal Gas Partition Function $\log \tilde{Z}$ run times for TI-Stan for two values of W. Twenty TI-Stan runs were completed for each value of W and N.

W	N	Mean Run Time (s)	StDev Run Time (s)
1.05	12	69.80	3.01
"	102	230.56	5.74
"	1002	2076.28	44.82
1.5	12	8.42	0.43
"	102	27.87	1.44
"	1002	247.53	6.80

These results demonstrate that TI-Stan can estimate thermodynamic quantities of interest with a small amount of error. They also show that the method can produce useful results with distributions with up to 1000 dimensions.

7. Conclusions

This article has presented three distinct model comparison algorithms based on thermodynamic integration: TI-BSS-H, TI-BSS-Z, and TI-Stan. Among these, TI-Stan is the most versatile and robust, providing log-evidence estimates in a reasonable amount of time for a wide range of example problems. TI-BSS-H is also a robust technique, yielding accurate estimates for three of the example problems; however, TI-BSS-H has trouble yielding results in a timely manner when the number of dimensions increases beyond a certain point. TI-BSS-Z is less robust, not yielding results at all in many of the examples presented. For problems with less troublesome distributions, it may be worth considering as an option, especially because its run time tends to be significantly less than TI-BSS-H.

Author Contributions: Conceptualization, R.W.H. and P.M.G.; Investigation, R.W.H. and P.M.G.; Software, R.W.H.; Supervision, P.M.G.; Validation, R.W.H.; Visualization, R.W.H.; Writing-original draft, R.W.H.; Writing-review & editing, P.M.G.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

NS	Nested sampling
NS-CC	Combined-chain nested sampling
NS-MR	Multiple-replacement nested sampling
TI	Thermodynamic integration
BSS	Binary slice sampling
TI-Stan	Thermodynamic integration with Stan
TI-BSS	Thermodynamic integration with binary slice sampling
TI-BSS-H	Thermodynamic integration with binary slice sampling and the Hilbert curve
TI-BSS-Z	Thermodynamic integration with binary slice sampling and the Z-order curve
HMC	Hamiltonian Monte Carlo
NUTS	No U Turn Sampler
MCMC	Markov chain Monte Carlo
CDF	cumulative distribution function

References

1. Kirkwood, J.G. Statistical Mechanics of Fluid Mixtures. *J. Chem. Phys.* **1935**, *3*, 300–313. [CrossRef]
2. Goggans, P.M.; Chi, Y. Using thermodynamic integration to calculate the posterior probability in Bayesian model selection problems. *AIP Conf. Proc.* **2004**, *707*, 59–66. [CrossRef]
3. Skilling, J. *BayeSys and MassInf*; Maximum Entropy Data Consultants Ltd., 2004. Available online: http://www.inference.phy.cam.ac.uk/bayesys (accessed on 24 November 2019).
4. Carpenter, B.; Gelman, A.; Hoffman, M.D.; Lee, D.; Goodrich, B.; Betancourt, M.; Brubaker, M.; Guo, J.; Li, P.; Riddell, A. Stan: A Probabilistic Programming Language. *J. Stat. Softw.* **2017**, *76*. [CrossRef]
5. Stan Development Team. PyStan: The Python interface to Stan, 2018. Available online: https://pystan.readthedocs.io/en/latest/ (accessed on 24 November 2019).
6. Hoffman, M.D.; Gelman, A. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.* **2014**, *15*, 1593–1623.
7. Henderson, R.W. TI-Stan: Adaptively Annealed Thermodynamic Integration with HMC. In Proceedings of the Multidisciplinary Digital Publishing Institute Proceedings, Garching, Germany, 30 June–5 July 2019; Volume 33, p. 9.
8. Henderson, R.W. Design and Analysis of Efficient Parallel Bayesian Model Comparison Algorithms. Ph.D. Dissertation, University of Mississippi, Oxford, MS, USA, 2019.
9. Gelman, A.; Meng, X.L. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Stat. Sci.* **1998**, *13*, 163–185. [CrossRef]
10. Oates, C.J.; Papamarkou, T.; Girolami, M. The Controlled Thermodynamic Integral for Bayesian Model Evidence Evaluation. *J. Am. Stat. Assoc.* **2016**, *111*, 634–645. [CrossRef]
11. Skilling, J. Nested sampling. Bayesian inference and maximum entropy methods in science and engineering. In *American Institute of Physics Conference Series*; AIP Publishing: Melville, NY, USA, 2004; Volume 735, pp. 395–405. [CrossRef]
12. Skilling, J. Nested sampling for general Bayesian computation. *Bayesian Anal.* **2006**, *1*, 833–859. [CrossRef]
13. Skilling, J. Galilean and Hamiltonian Monte Carlo. In Proceedings of the 39th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering, Garching, Germany, 30 June–5 July 2019.
14. Liu, J.S.; Chen, R.; Logvinenko, T. A Theoretical Framework for Sequential Importance Sampling with Resampling. In *Sequential Monte Carlo Methods in Practice*; Doucet, A., de Freitas, N., Gordon, N., Eds.; Springer: New York, NY, USA, 2001; pp. 225–246. [CrossRef]
15. Skilling, J.; MacKay, D.J.C. [Slice Sampling]: Discussion. *Ann. Stat.* **2003**, *31*, 753–755.
16. Neal, R.M. Slice Sampling. *Ann. Stat.* **2003**, *31*, 705–767. [CrossRef]
17. Sagan, H. *Space-Filling Curves*; Springer: New York, NY, USA, 1994.

18. Skilling, J. Programming the Hilbert curve. In *AIP Conference Proceedings*; Erickson, G., Zhai, Y., Eds.; AIP Publishing: Melville, NY, USA, 2004; Volume 1, pp. 381–387.
19. Henderson, R.W.; Goggans, P.M. Using the Z-order curve for Bayesian model comparison. In *International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering*; Polpo, A., Stern, J., Louzada, F., Izbicki, R., Takada, H., Eds.; Springer: Berlin/Heidelberger, Germany, 2018; Volume 239, pp. 295–304.
20. Gabriel. How to Compute a 3D Morton Number (Interleave the Bits of 3 Ints). 2013. Available online: https://stackoverflow.com/questions/1024754/how-to-compute-a-3d-morton-number-interleave-the-bits-of-3-ints (accessed on 24 November 2019).
21. Neal, R.M. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*; Brooks, S., Gelman, A., Jones, G., Meng, X.L., Eds.; Chapman and Hall: London, UK, 2011.
22. Feroz, F.; Hobson, M.P.; Bridges, M. MultiNest: An efficient and robust Bayesian inference tool for cosmology and particle physics. *Mon. Not. Roy. Astron. Soc.* **2009**, *398*, 1601–1614. [CrossRef]
23. Bretthorst, G.L. *Bayesian Spectrum Analysis and Parameter Estimation*; Springer: Berlin/Heidelberger, Germany, 1988.
24. Bretthorst, G.L. Nonuniform sampling: Bandwidth and aliasing. In *AIP Conference Proceedings*; AIP Publishing: Melville, NY, USA, 2001; Volume 567, pp. 1–28. [CrossRef]
25. Handley, W.; Hobson, M.; Lasenby, A. PolyChord: Next-generation nested sampling. *Mon. Not. Roy. Astron. Soc.* **2015**, *453*, 4384–4398. [CrossRef]
26. Von der Linden, W.; Dose, V.; Von Toussaint, U. *Bayesian Probability Theory: Applications in the Physical Sciences*; Cambridge University Press: Cambridge, UK, 2014.