Efficient importance sampling in low dimensions using affine arithmetic

Comput Stat
DOI 10.1007/s00180-017-0729-z

ORIGINAL PAPER

Efficient importance sampling in low dimensions using affine arithmetic

Richard G. Everitt[1]

Received: 20 June 2016 / Accepted: 25 April 2017

Abstract Despite the development of sophisticated techniques such as sequential Monte Carlo (Del Moral et al. in J R Stat Soc Ser B 68(3):411–436, 2006), importance sampling (IS) remains an important Monte Carlo method for low dimensional target distributions (Chopin and Ridgway in Leave Pima Indians alone: binary regression as a benchmark for Bayesian computation, 32:64–87, 2017). This paper describes a new technique for constructing proposal distributions for IS, using affine arithmetic (de Figueiredo and Stolfi in Numer Algorithms 37(1–4):147–158, 2004). This work builds on the Moore rejection sampler (Sainudiin in Machine interval experiments, Cornell University, Ithaca, 2005; Sainudiin and York in Algorithms Mol Biol 4(1):1, 2009) to which we provide a comparison.

Keywords Importance sampling · Monte Carlo methods · Affine arithmetic

1 Introduction

Monte Carlo methods are core techniques in computational statistics, in Bayesian inference in particular. The three most commonly used techniques are: rejection sampling; importance sampling and Markov chain Monte Carlo. The situations in which rejection sampling may be used effectively are limited, particularly as the dimension of the space increases. Markov chain Monte Carlo is the most generally applicable method, even in high dimensional spaces, and has been at the forefront of methods for Bayesian computation for the past 25 years. Importance sampling is limited to use for distributions of low to moderate dimension, but can be the most effective of any method

✉ Richard G. Everitt
r.g.everitt@reading.ac.uk

[1] Department of Mathematics and Statistics, Whiteknights, PO Box 220, Reading RG6 6AX, UK

Published online: 22 May 2017

under these circumstances (Chopin and Ridgway 2017). The challenge in designing an effective importance sampler is to design a "proposal" distribution from which we can simulate points, that is similar to a "target" distribution from which we cannot, but which is available to evaluate pointwise. This paper describes a new method for constructing such a proposal distribution, based on the use of *auto-validating* methods.

Sainudiin (2005), Sainudiin and York (2009) introduce a new Monte Carlo method named the *Moore rejection sampler*. This is an ordinary rejection sampler where the proposal is constructed automatically using *interval* methods (Moore 1962, 1966), a type of auto-validating method. This method for constructing the proposal is more generally applicable than previous work on "adaptive" rejection sampling (Gilks et al. 1995) and can find a proposal that is near to the target with a small computational cost. The results presented in Sainudiin (2005), Sainudiin and York (2009) are impressive, allowing points to be simulated (at a reasonable acceptance rate) from some multimodal target densities that present a significant challenge for standard Monte Carlo methods.

In Sects. 1.1 and 1.2 we summarise the strengths and weaknesses of the Moore rejection sampler and, based on this, introduce a new way of using auto-validating methods in Monte Carlo. We then give an overview of the paper in Sect. 1.3.

1.1 What are auto-validating methods?

Auto-validating methods have their roots in numerical analysis, with the original aim being to describe errors due to performing calculations using the finite representation of real numbers in a computer. The methods have the name "auto-validating" because they keep track of the errors introduced by the calculation, so it is possible to automatically decide if it is necessary to refine the calculation in order to reduce the error. The first work in this field was the introduction of *interval analysis* (Moore 1962, 1966), and these ideas have now found application in several different fields, notably ray-tracing in computer graphics (Enger 1992) and optimisation (e.g. Baker Kearfott 1996; Jaulin et al. 2001). Interval analysis has its limitations for accurately describing the error introduced by performing calculations on a computer, but this does not matter for some types of applications where only a bound on the error is required. The most notable use is that of Tucker (1999), where the Lorenz attractor is proved to exist through executing a computer program: interval analysis is used to bound the errors in the program.

Let us be a little more specific about what interval analysis lets us achieve. Suppose we have a real valued function $f : D \to \mathbb{R}$, with $D \subseteq \mathbb{R}$. The range $f(E) \subseteq \mathbb{R}$ of f over a subset $E \subseteq D$ is defined as: $f(E) = \{f(x) \mid x \in E\}$. For any interval $I_D \subseteq D$, interval analysis gives a computationally cheap way of finding an interval $I_R \subset \mathbb{R}$ such that $f(I_D) \subseteq I_R$, for a large class of functions f. This property is known as *range enclosure*. In some cases $f(I_D)$ is close enough to I_R for us to be able to use I_R as an approximation for $f(I_D)$ (in fact, in some cases $I_R = f(I_D)$). However, sometimes it is the case that I_R is significantly larger than $f(I_D)$, in which case we say we have a large *over-enclosure*.

It is this computationally cheap manner of finding a "range approximation", an interval I_R such that $I_R \approx f(I_D)$, that we make use of in this paper. The major limitation on our use of interval analysis is in the case where we have a large over-enclosure. It is this case in which the use of an alternative approach called *affine arithmetic* is often suitable. Over-enclosure tends to occur when there are many instances of the same variable in the equation of a function. To see why this is the case, and why affine arithmetic can help, requires a little knowledge of auto-validating methods, which we provide in Sect. 2.2.

1.2 When does the Moore rejection sampler perform well?

As we mention above, the key component of the Moore rejection sampler is the algorithm for finding a dominating proposal function that is a good approximation to the target. This algorithm is based on the use of the range approximation provided by interval methods. It is particularly effective (compared to standard methods) when applied to multi-modal target densities, especially when the modes are narrow and well-separated.

We will see that there are two limitations on this method:

- **Over-enclosure of the target density.** The effectiveness of the construction of the proposal in the Moore rejection sampler is driven completely by the use of interval methods to approximate the range of the target. Thus, in situations where interval analysis over-encloses the target significantly, the Moore rejection sampler is not suitable: the poor quality of such a range approximation adversely affects the effectiveness of the method in a way that is more precisely described in Sect. 3.1.4.
- **High dimensional target densities.** The method suffers from the curse of dimensionality in a similar way to numerical integration methods, since it works through subdividing the domain of the target distribution into pieces. This issue is discussed further in Sect. 3.1.5.

In this paper we aim to address the first of these limitations, through the use of affine arithmetic, which is often is less prone to over-enclosure. The second limitation remains, thus we limit our consideration to low-dimensional target distributions.

1.3 Overview

We begin the paper, in Sect. 2.1, with a review of the relevant literature on importance sampling. We then introduce auto-validating methods: in Sect. 1 we give the main results from interval analysis and discuss the problem of over-enclosure, then Sect. 1 we introduce an alternative auto-validating approach, affine arithmetic. In Sect. 3.1 we provide a review of the Moore rejection sampler in Sainudiin (2005), Sainudiin and York (2009), then in Sect. 3.2 we describe our proposed alternative method that uses affine arithmetic, and compare its properties with the interval based approach empirically in Sect. 4. In Sect. 4.4 we draw some conclusions and suggest future work.

2 Background

2.1 Importance sampling

In this section we review recent work in importance sampling, and describe where the approach proposed in this paper fits into the literature.

2.1.1 Measuring the effectiveness of an importance sampler

A quantity at the core of assessing the effectiveness of a Monte Carlo method is the Monte Carlo variance of expectation estimates. The effectiveness of an importance sampler is often measured by its relative numerical efficiency (RNE), this being the relative variance of an independent sampler to the importance sampler. We may estimate the variance from the importance sampling output (its square root being known as the numerical standard error). However, recent work Chatterjee and Diaconis (2015) shows that this estimate is unreliable for assessing the convergence of an importance sampler. This paper discusses the use of the Kullback-Leibler divergence between the target and proposal distributions for assessing when estimates from an importance sampler will be accurate.

However, estimating the this Kullback-Leibler divergence is difficult in practice. Instead, Chatterjee and Diaconis (2015) introduce an alternative estimate of the distance between the proposal and the target; one that is of a similar type to the *effective sample size* (ESS) (Kong et al. 1994), which is commonly used in the sequential Monte Carlo literature. The ESS is defined to be

$$\text{ESS} = \left(\sum_{i=1}^{N} [(\tilde{w}^{(i)})^2] \right)^{-1}.$$

We see that the ESS is readily interpretable in a similar fashion to the RNE: in the case of an independent sampler, the ESS is equal to the number of Monte Carlo points; and as the ESS reduces (when there is a larger distance between the proposal and the target) it provides an indication as to how many points would be required in order for our importance samplers to be as efficient as an independent sampler (Martino and Read 2013).

The recent review article (Agapiou et al. 2017) describes each of the above measures of effectiveness of importance sampling, and carefully describes the relationships between them. They make clear the central role of the second moment ρ of the Radon-Nikodym derivative of the target with respect to the proposal and describe how, asymptotically, the ESS may be used to estimate ρ. For this reason, we use the ESS to evaluate the new algorithms proposal in this paper.

2.1.2 Designing proposal distributions

From the discussion in the previous section it is clear that for importance sampling to be effective, we require proposal distribution that is close to the target distribution. Much of the recent research on importance sampling is on different methods for constructing

proposals; a topic that has received renewed attention since the introduction of the psuedo-marginal method (Andrieu and Roberts 2009) in which importance sampling is used within an MCMC algorithm.

Here, in order to set our work in context, we briefly outline some of the approaches that have recently been proposed to construct importance samplers.

- In situations in which the target distribution is unimodal, it is often appropriate to use a Gaussian proposal distribution (or a Student-t for heavy tailed targets), which can often be easily constructed using a Laplace approximation (Geweke 1989; Chopin and Ridgway 2017). In many simple models (e.g. Geweke 1989 this approach is sufficient), and can be very effective even when the target is on a space of moderate dimension (i.e. up to dimension 50).
- Methods that automatically construct rejection sampling proposals (Gilks et al. 1995; Marrelec and Benali 2004) may sometimes be suitable. We note they are usually limited to particular types of target distributions.
- In some situations a target distribution may not be available to evaluate pointwise directly. For example, consider the situation where the target distribution is the marginal posterior distribution of a parameter (integrating over a large number of latent variables) (Tran et al. 2014), or in the situation of a likelihood with an intractable partition function (Everitt et al. 2017). Along the same lines as the pseudo-marginal approach, we may use an importance sampler within an importance sampler, with the internal importance sampler serving to estimate the unavailable intractable likelihood.
- Where there is no obvious proposal distribution, but there is an obvious sequence of auxiliary target distribution that form a "path" between a simple proposal distribution and the target distribution (e.g. through use of annealing). it is possible to use a sequential importance sampling approach to reduce the effect of a poor choice of proposal. This is the approach used in annealed importance sampling (Neal 2001), along with its generalisation, sequential Monte Carlo (SMC) (Del Moral et al. 2006). These techniques are themselves now a fundamental part of the Bayesian computation toolbox (e.g. Chopin and Ridgway 2017; Lenormand et al. 2013).
- In the context of sequential Monte Carlo, Everitt et al. (2016) describes the idea of using application specific information to construct a deterministic transformation that brings a proposal distribution closer to a target.

In this paper we are concerned with target distributions that are not unimodal, thus a Laplace approximation is not appropriate. For low-dimensional problems, it is computationally expensive to use sequential Monte Carlo if it can be avoided, since the cost of this algorithm is of the same order as a single importance sampler at every intermediate target distribution. This paper describes such a way to avoid using SMC, through using affine arithmetic to construct an accurate proposal.

2.2 Auto-validating methods

Let $I \in \mathbb{I}D$ be the set of compact intervals in D. Interval analysis allows us to find a bound on the range of a function $f : D \subset \mathbb{R} \to \mathbb{R}$ over an interval $I \in \mathbb{I}D$ through

the use of a function $F : \mathbb{I}D \to \mathbb{IR}$. We obtain such a bound if F satisfies *range enclosure*: if $\forall X \in \mathbb{I}D$, $f(X) \subseteq F(X)$. Ideally, F will not over-enclose f: it will satisfy *range equivalence*, so that $f(X) = F(X)$. Such a function F is known as an "interval extension" of f.

We leave the precise details of how to construct interval extensions to the "Interval analysis" section in the Appendix. In this section, we restrict our attention to: outlining, in intuitive terms, how an interval extensions are constructed; describing the problem of over-enclosure, which limits the accuracy of range approximations of functions; and finally, how affine arithmetic may be used to reduce over-enclosure (again leaving the technical details of arithmetic to the "An introduction to affine arithmetic" section in the Appendix).

2.2.1 Interval analysis

It is straightforward to construct an interval extension F for monotone functions $f : \mathbb{R} \to \mathbb{R}$. For example, suppose f is the exponential function. In this case we obtain an interval extension F by mapping an interval $[a, b]$ to the interval $[\exp(a), \exp(b)]$, this being a function that satisfies range equivalence. It is also straightforward to see that this approach may be used to find interval extensions to compositions of monotone functions.

Interval arithmetic describes how to extend this idea to the set of elementary functions, with multivariate domains and multivariate ranges. The main consideration is how to construct an interval extension to a function with two arguments, such as addition. The aim of interval arithmetic is to rigorously describe worst-case bounds, thus we only need consider the most extreme positive and negative results that may occur. For example, the interval extension of addition of intervals X and Y is defined to be $X \times Y = [\min\{\underline{x}\,\underline{y}, \underline{x}\,\overline{y}, \overline{x}\,\underline{y}, \overline{x}\,\overline{y}\}, \max\{\underline{x}\,\underline{y}, \underline{x}\,\overline{y}, \overline{x}\,\underline{y}, \overline{x}\,\overline{y}\}]$. The details of using interval arithmetic with other functions is omitted here, but is formalised in the "Interval analysis" section in the Appendix. The Fundamental Theorem of Interval Analysis is the most important result in the appendix. It says that for any elementary function, we have a way of finding a bound on its range on any interval (or a *box* in the multivariate case).

Since the primary goal of interval arithmetic is range enclosure, it sometimes produces bounds that are overly conservative, and this is the phenomenon known as over-enclosure. Over-enclosure is encountered when the rules of interval arithmetic are applied "blindly" and do not notice when the same argument is encountered twice. For example, consider the function $f(x) = x \times x$. If an interval extension F to f is constructed by using the definition in the previous paragraph, if the input is $x = [-1, 1]$, we see that $F(x) = [-1, 1]$, even though we know that the minimum possible value is 0. In this case, the problem is easily solved by using an alternative expression for the function in which the variable x occurs only once, i.e. $f(x) = x^2$, for which the interval extension gives the exact range. However, in general, where function is a composition of many constituent parts, it may not be possible to rearrange the expression in such as way, thus the over-enclosure remains.

However, in some cases there is little over-enclosure, and the bound given by an interval extension is tight enough for us to think of it as an approximation to the true

range of the function. It is in this sense that we make use of interval analysis (it is not really of interest to us that we have a bound on the range, only that we have an approximation to it). To ensure there is no over-enclosure, so that the result is exact, we require the use of an expression where each variable only occurs once in the expression. In some cases this is possible: for example, consider a function proportional to a posterior distribution, obtaining the multiplying a prior and a likelihood. In general the likelihood is a product of a number of terms, all involving the same parameter (such as the case considered in Sect. 4.2). However, in practice it is often possible to rearrange this product in terms of a statistics of the data, so that the parameter only occurs once. Where it is not possible to find an expression where each parameter occurs only once, we should minimise the number of times each variable occurs in the expression.

It is relatively easy to see how this may bring a benefit to Monte Carlo algorithms that has not been considered previously. When considering Monte Carlo algorithms we usually treat the target function as a "black box" that converts an input into an output, rather than considering how it is composed. Thus, in order to be sure about the range of a function over an interval, we would usually have to resort in swamping a region with points until we were confident that we knew everything about the range in that region. Using interval arithmetic we can find out about the range of a function over an interval only by considering the end points of that interval (the computational cost of evaluating an interval extension is only twice that of evaluating the original function). As long as the over-enclosure is not too large, we ought to be able to use this information to our advantage in Monte Carlo algorithms.

2.2.2 Affine arithmetic

Several ways of countering the dependency problem have been devised. Any of these alternative methods can be used as a substitute for standard interval methods to approximate a range: if they yield a smaller over-enclosure with a small additional computational cost, it is likely that an improvement over the results obtained in this chapter will be obtained.

Here we consider the use of affine arithmetic (de Figueiredo and Stolfi 2004): this tackles the dependency problem in the most obvious way, but also provides a linear approximation to a function over a box. This approximation is obtained in a computationally cheap way by component-wise operations on a representation of an elementary function and we will make use of it later in the paper. In this introductory section on affine arithmetic we emphasise its use as an alternative to interval analysis, considering the linear approximation to be a side effect.

In practice, we use an affine form as an alternative representation of an interval. If we define "affine extensions" of arithmetic and standard operations that have affine forms as their inputs and outputs, then analogous to the natural interval extension we may define the natural affine extension $F_A : \mathbb{AR} \to \mathbb{AR}$ of the representation of an elementary function $f : \mathbb{R} \to \mathbb{R}$.

The idea is that the use of affine forms can allow us to keep track of dependencies between two or more quantities, helping us to avoid the dependency problem and so reduce over-enclosure. Appendix section "An introduction to affine arithmetic"

formalises this concept. The main complication occurs when affine forms are used in non-affine functions: in these cases the result of applying the function is chosen to be an affine form that approximates the true result. If this approximation is accurate, an accurate description of the dependencies is maintained, and over-enclosure is not large. Therefore, in these cases, we obtain a more accurate approximation to the range of a function.

2.2.3 Approximating functions using hyper-planes

As well as using affine arithmetic for finding a tighter bound on the range, we can also use it for another purpose. Whilst an interval extension only gives information about the range of a function over an interval, an affine extension also also contains information about the linear dependence of a function on its inputs. For any interval in the domain, it gives a linear approximation to the function over the interval (or, when the domain is multi-dimensional, for any box in the domain, it gives a hyper-plane approximation to the function over that box).

Note that such approximations differ from those that we may find using alternative methods since they are found by considering the composition of the function (as opposed to treating it as a "black box") and they take into account *all* of the points in the domain (rather than the finite number that we are forced to consider using conventional methods). We will use the approximating hyper-plane idea in Sect. 3.2. The quality of the approximation is governed by the factors we have already discussed.

2.2.4 Computational cost of interval and affine extensions

An interval extension F of a function f only requires the evaluation of the original function at the upper and lower bounds on an input interval. Therefore, if f is composed of m component functions, the cost of evaluating both the original function and its interval extension $O(m)$.

For affine arithmetic, suppose that we are evaluating the affine extension of a function made of m component functions on an input affine form with only one source of variability. In general, assuming non-affine operations, we will add a noise term at each of the m component stages. Assuming that each of the component stages is linear in the number of noise terms, the cost of evaluating the affine extension is $O(m^2)$.

There is a way to avoid this cost (Stolfi and de Figueiredo 1997). For the applications we are interested in, most of the time, the extra noise terms that are introduced every time we perform a non-affine operation can be amalgamated into a single error term without any adverse effects. There is a chance that keeping track of the errors introduced by particular operations is beneficial, but this is rare. If these error terms are amalgamated, then the cost is much smaller—we need to apply each function to the constant term, the noise terms corresponding to those in the inputs, and the amalgamated error term. In this case, the cost of evaluating the affine extension is $O(m)$ and may actually cost only a little more than evaluating the interval extension. We note that in the applications considered in this paper, we did not in practice find the relative cost of affine arithmetic compared to interval arithmetic to be prohibitive, therefore we did not implement this approach.

2.2.5 Implementation of auto-validating methods

The only non-standard aspect of the new samplers proposed in this paper is in the use of interval and affine extensions of density functions. Fortunately, the implementation of auto-validating techniques is straightforward, through the use of free interval arithmetic libraries available in many high-level programming languages (e.g. pyinterval in python, INTLAB in Matlab). The implementations in this paper used C++: we used the interval library in boost, and the free libaffa library (available from http://savannah.nongnu.org/projects/libaffa/) for implementing the methods based on affine arithmetic.

3 Auto-validating samplers

The work in this chapter was motivated by the work in Sainudiin (2005), Sainudiin and York (2009) on the "Moore rejection sampler". In this section we present an analysis of this method, the aim being to discover its advantages and limitations. We suggest an alternative to the Moore rejection sampler that is based on affine arithmetic.

We begin in Sect. 3.1 by introducing the Moore rejection sampler and follow this by introducing a variation on the same idea using affine arithmetic in Sect. 3.2. Section 4 contains results of running these methods on several different problems, with Sect. 4.4 reviewing these methods.

3.1 The Moore rejection sampler

3.1.1 Introduction

The Moore rejection sampler is an ordinary rejection sampler where the proposal is constructed using interval methods. This construction can give a very good approximation of π. This method of constructing a proposal is more widely applicable than previous approaches to automating the construction of the proposal: "adaptive rejection sampler" of Gilks et al. (1995) can only be used when the target density is one-dimensional and log-concave. The Moore rejection sampler can be applied whenever π is an elementary function.

The interval derived approximation can approximate the target π very well, and also has the property that it dominates the target (or, at least, it is easy to ensure that is does). The latter characteristic of the interval derived approximation is less important, since (as described in Sainudiin and York 2009), we can use the same idea in IS and MCMC with an independence proposal. In these algorithms we do not need the proposal to dominate the target, but we still obtain the best performance when the proposal is as close as possible to π.[1] There is no particular reason to use the rejection sampler over

[1] In general we use the approximation that we construct as a proposal within a Monte Carlo simulation algorithm, however the approximation is sometimes so good that samples from the approximation could be used as if they are directly from the target, just as we may with standard numerical approximations such as Simpson's rule.

these other methods: Sainudiin and York (2009) indicates that an importance sampler that uses the interval-derived approximation as its proposal usually performs well, so in the remainder of this section we use this method.

The approximation used is a (normalised) step function: a mixture of disjoint uniform densities. We begin by examining the properties of this approximating density, then introduce the basic method for constructing the approximation. We then go on to describe how interval methods are used within this basic algorithm and highlight some of the advantages and disadvantages of this approach.

3.1.2 A step function approximation

Let the approximation π_a of π have probability density function:

$$\pi_a\left(x|\{l_i, u_i, w_i\}_{i=1}^k\right) = \sum_{i=1}^{k} w_i \mathcal{U}_i(x_i; l_i, u_i),$$

with $\sum_{i=1}^k w_i = 1$, $w_i \geq 0$ and with $\mathcal{U}_i(x_i; l_i, u_i)$ denoting a uniform density on x with lower bound $l_i \in \mathbb{R}^m$ and upper bound $u_i \in \mathbb{R}^m$, where the domains of the uniform densities are disjoint.

Such an approximation is useful as a proposal in Monte Carlo algorithms because of the following properties:

1. It is possible to simulate from the density in constant time, irrespective of the number of components k (although there is a setup cost). An algorithm for achieving this (for a discrete density) is presented in Walker (1977).
2. It is computationally cheap to evaluate at a point when used as a proposal in Monte Carlo algorithms. This is the case since every time the proposal is evaluated at a point in Monte Carlo algorithms, it is at a point that has been simulated from the density—in this case we know which component the point has been drawn from and the value of the density is simply the weight of that component.

3.1.3 The approximation method

Finding an approximation by a mixture of disjoint uniform densities is not difficult—we can simply divide the domain into pieces (to be the uniform components) and, for example, take the value of π at the mid-point of each component to be the weight of the component (then normalise the function). However, implemented naively, this method is clearly of little use for a large domain. The approach taken in Sainudiin (2005) is to use the interval extension of the target to decide how to divide the domain.

The basic algorithm for dividing the domain is given below in algorithm 1. There are many sensible choices for the weight of each component in the approximation found by algorithm 1: for example, choosing the value of π at the midpoint of the component. Sainudiin (2005) suggests the upper bound of the evaluation of the interval extension on the domain of the component since this guarantees a dominating proposal for use in rejection sampling (note that usually we would normalise the weights to ensure that

we have a probability density, but we would not do this when using the approximating proposal in rejection sampling). In this chapter we always choose the midpoint of the evaluation of the interval extension on the domain of the component.

Input: A target density π, a maximum number of iterations M and a domain $D \in \mathbb{IR}^m$ (an m-dimensional box).
Output: A step function approximation to π..
Initialise the partition with the original domain, $\mathcal{P} = D$.
for $i = 1 : M$
Choose a member p of $\mathcal{P}$ to partition (using a priority queue to implement this efficiently) and remove p from $\mathcal{P}$.
Split p into two pieces (for example, pick a dimension d randomly in proportion to the lengths of the dimensions of p, then split p into two equally sized pieces p_1 and p_2 along this dimension).
Add p_1 and p_2 to $\mathcal{P}$.
end
Make a mixture of disjoint uniforms from $\mathcal{P}$ and π.

Algorithm 1: A general method for finding an approximation to a target density.

3.1.4 Prioritising the partitions

The following idea represents the point at which we benefit from using interval methods in this algorithm. The guiding principle is that we ought to choose to split the member of the partition over which a constant function is in some sense the worst approximation to the true target. There are different ways that we could measure this, but we choose the "integral-based" score from Sainudiin (2005), Sainudiin and York (2009). Suppose for a moment that the interval extension Π of π gives the range of π exactly for any interval in the domain. For a member of the partition p, the integral-based score is the range of the interval $\Pi(p)$ multiplied by the length (area, or volume in multi-dimensional cases) of p as a score for describing the quality of this approximation. In Sainudiin and York (2009) this score is compared with other scores that do not take into account either the range of the function on a piece of the domain, or the size of the piece of the domain (unsurprisingly, the integral-based score outperforms both of these ideas).

In practice, the interval extension will not give the exact range. For the above criterion to work well, the enclosure of the range should be as tight as possible. Although the over-enclosure is likely to be large when we begin the algorithm, as the partition is refined the effect of the over-enclosure lessens. Therefore, if early on in the algorithm we partition because of over-enclosure rather than true variation in the range, this effect should be killed off to some degree as the partition is refined.

3.1.5 Review

Sainudiin and York (2009) presents impressive results when applying this method to some probability densities that are difficult to sample from using conventional methods (including multivariate witch's hat, Rosenbruck and bivariate Levy densities). The

main problem with these densities is that we must sample from a large number of modes: it is usually difficult to ensure that all of the modes are found without the sampler being very inefficient. Through the methods described above, the use of an interval extension helps to locate these modes quickly and leads to an efficient sampler.

When using this method, the computational cost of sampling from a target is best considered in two separate parts: constructing the approximating target; then generating samples from the target using a rejection/importance/MCMC algorithm. For some applications we may need only to find the approximation once (it can be "pre-enclosed"), but perform the sampling several times.

This method should work well when the target's domain only has a small number of dimensions, or when the target density has its mass concentrated in only a few regions. If we require a large number of partitions in order to find a good approximation to the target density (the curse of dimensionality means that high-dimensional targets almost always fall into this category) this method may not be appropriate. There are two reasons why performing a large number of partitions may be a problem:

- Performing a large number of partitions in the pre-enclosure stage of the algorithm requires a lot of computational effort;
- There is a practical limit on the method imposed by the random access memory (RAM) available, since we need to store the details of a large number of mixture components.

Despite these constraints, the results in Sainudiin (2005) indicate that the method can be useful in up to 10-11 dimensions. Indeed, for low dimensional targets, this method may outperform most standard Monte Carlo algorithms and has the advantage of requiring little or no tuning.

Sainudiin (2005) focuses on the use of this technique in Monte Carlo methods, however the key idea is in the construction of the approximating target and this has a lot in common with numerical integration, adaptive quadrature (Kuncir 1962) in particular. The approximating function may be used directly to estimate integrals, without the use of Monte Carlo. However, to provide a direct comparison with Sainudiin (2005), and to provide unbiased estimates of integrals in order that these techniques may be used within other simulation algorithms (Andrieu and Roberts 2009), we also study the approach as a means for providing proposals for Monte Carlo algorithms.

3.2 An auto-validating sampler based on affine arithmetic

We now consider the use of affine arithmetic, rather than interval arithmetic, in the method in the previous section. There are two obvious ways that affine arithmetic (or mixed interval-affine arithmetic) may be used:

- As a direct substitute for interval arithmetic, to obtain a tighter bound on the range of the function over an interval. We expect this to outperform the interval arithmetic approach for functions where the dependency problem is our main reason for over-enclosure. From hereon we refer to this approach, where mixed interval-affine arithmetic is used to find the bound as the **standard-affine** auto-validating sampler. We refer to the original auto-validating sampler of Sainudiin (2005) as the **standard-interval** approach.

– If we use affine arithmetic, we automatically get information about how the target function varies linearly with its inputs. We can use this to construct an alternative approximation to the mixture of disjoint uniforms in the previous section. We will refer to this as the **fully-affine** approach. The relationship between the standard-affine approach and the fully-affine approach is analogous to the relationship between the rectangle rule and the trapezoidal rule in numerical integration.

It is the fully-affine approach that we describe in this section. We use the partitioning method in algorithm 1, but some of the details are different. Firstly we talk about the new approximating density, then how it is constructed.

3.2.1 A mixture of disjoint wedge densities

The approximating density we use in this section is non-standard, although the details are straightforward. We first define a multivariate generalisation of the triangular distribution, then define a wedge distribution, which is the sum of this multivariate triangular distribution and a multivariate uniform distribution. We then describe the mixture of disjoint wedge densities and discuss its properties.

Definition 1 Multivariate triangular distribution

The multivariate triangular distribution has probability density function $p : [l_1, u_1] \times \cdots \times [l_n, u_n] \to \mathbb{R}$ given by:

$$p(x_1, \ldots, x_n | \alpha_1, \ldots, \alpha_n) = \left(\sum_{i=1}^{n} \frac{\alpha_i}{2} (u_i^2 - l_i^2) \prod_{j \neq i} (u_j - l_j) \right) (\alpha_1 x_1 + \cdots + \alpha_n x_n).$$

Definition 2 Wedge distribution

The wedge distribution has a probability density that is a weighted sum of a multivariate triangular distribution $\mathcal{T}$ and a multivariate uniform distribution $\mathcal{U}$ that have the same domain, $[l_1, u_1] \times \cdots \times [l_n, u_n]$:

$$p(x_1, \ldots, x_n) = v_1 \mathcal{T}(x_1, \ldots, x_n) + v_2 \mathcal{U}(x_1, \ldots, x_n)$$

where $v_1 + v_2 = 1$.

The wedge density is simply the equation of a hyperplane, restricted to a box in the domain and normalised.

Then the mixture of disjoint wedge densities is:

$$p(x) = \sum_{i=1}^{k} w_i \mathcal{W}_i(x_i),$$

with $\sum_{i=1}^{k} w_i = 1$, $w_i \geq 0$ and where the $\mathcal{W}_i(x_i)$ are wedge densities with disjoint domains and $x \in \mathbb{R}^m$. The appealing properties of the mixture of disjoint uniform densities still hold for this new approximating density.

Constructing this new approximating distribution using the evaluation of an affine extension is simple. Consider the affine extension F_A of a function $f : D \to \mathbb{R}$, where $D \subset \mathbb{R}^m$. We first describe how to construct a wedge distribution on a box in the domain. Let $X_A \in \mathbb{A}D$ (a box written in affine form, in the affine extension of the set D), where the jth element of X_A has error term ϵ_j. Then:

$$F_A(X_A) = a_0 + \sum_{i=1}^{m} a_i \epsilon_i + \sum_{j=1}^{n_e} a_j \epsilon_{m+j},$$

where the error terms $m + 1$ to $m + n_e$ are the error terms introduced by the affine extension, as outlined in the "An introduction to affine arithmetic" section in the Appendix.

We construct a wedge density on the box X_A by finding the equation of a hyperplane $\text{pl}(x_1, \ldots x_m) = \alpha_0 + \sum_{j=1}^{m} \alpha_j x_j$ given by $F_A(X_A)$ as follows:

- Find each coefficient α_j for $j = 1, \ldots, m$ from the corresponding coefficient a_j in $F_A(X_A)$ and the lower bound l_j and upper bound u_j of dimension j in X_A:

$$\alpha_j = 2a_j(u_j - l_j).$$

- We use the hyperplane whose height at the centre of X_A is a_0, thus we find the constant term α_0 in the equation of the plane using:

$$\alpha_0 = a_0 - \sum_{j=1}^{m} \alpha_j c_j,$$

  where $(c_1, \ldots, c_m)$ is the centre point of the box X_A. In the following section we also make use of "upper" and "lower" approximating hyperplanes, which are the hyperplanes whose heights at the centre of X_A are $a_0 + \sum_{j=1}^{n_e} |a_j|$ and $a_0 - \sum_{j=1}^{n_e} |a_j|$ respectively.

The wedge density on X_A is simply the normalisation of this hyperplane: $(1/V_{X_A})\text{pl}(x_1, \ldots x_m)$, where V_{X_A} is the volume under the hyperplane on X_A. Our approximating distribution is given by the mixture of disjoint wedge densities, with a wedge density for each element in the partition of the domain, and the weights in the mixture given by the relative volumes under the hyperplanes from which the wedge densities are constructed.

3.2.2 When to partition

The important part of the interval based autovalidating sampler was the criterion by which we choose the element of the current partition to split. We use a similar method for the affine case. A suitable (integral-based) measure of the quality of the affine approximation is given by the volume enclosed by the upper and lower approximating hyperplanes given by $F_A(X_A)$ (as described in the previous section). For an example of this region, see the area between the green lines in Fig. 5a.

The main advantage over the original standard-interval auto-validating sampler is that we may require fewer partitions to find a function that is a good approximation of the target (since the approximation in the affine approach includes as a special case the step function approximation in the standard-interval approach). The result of this may be a computational saving in the pre-enclosure stage of the algorithm (since we do not need to refine the partition as many times) and a reduced requirement on the storage of the approximation in the RAM of the computer.

4 Examples

In this section we give some examples of applications of the auto-validating samplers presented in this paper. We begin with the very simple example of approximating then sampling from a gamma density in order to illustrate the application of both the interval and affine approaches. We then move onto more challenging examples to illustrate that these methods can be applied to problems of practical interest.

In all of our experiments we use the approximaton constructed by auto-validating methods in an importance sampler, for which we now define our notation. Let π be the target distribition and q be the importance proposal. IS constructs estimates of properties of the target distribution using points simulated from the proposal, e.g. using the property we may estimate the expectation of π using

$$\widehat{\mathbb{E}}_{\pi}[X] = \frac{1}{N}\sum_{i=1}^{N} \tilde{w}^{(i)} x^{(i)},$$

where $x^{(i)} \sim q$ and

$$\tilde{w}^{(i)} = \frac{w^{(i)}}{\sum_{i=1}^{N} w^{(i)}} \qquad \text{with} \qquad w^{(i)} = \frac{\pi\left(x^{(i)}\right)}{q\left(x^{(i)}\right)}.$$

In all cases the results are based on $N = 10{,}000$ points, and we use the ESS to measure the distance between the proposal and target distribution, as described in Sect. 2.1.

4.1 Gamma density

First we consider simulating points from a gamma density with scale 1 and shape 5:

$$p(x) = x^4 \exp(-x)/\Gamma(5), \tag{1}$$

where Γ is the gamma function.

There are standard functions in most software packages that let us sample from a gamma density. However, it is a good illustrative example on which to apply the new fully-affine approach, since the expression involves the composition of two non-affine functions—multiplication and the exponential. In fact, the performance of the method is improved considerably by rewriting the density as follows:

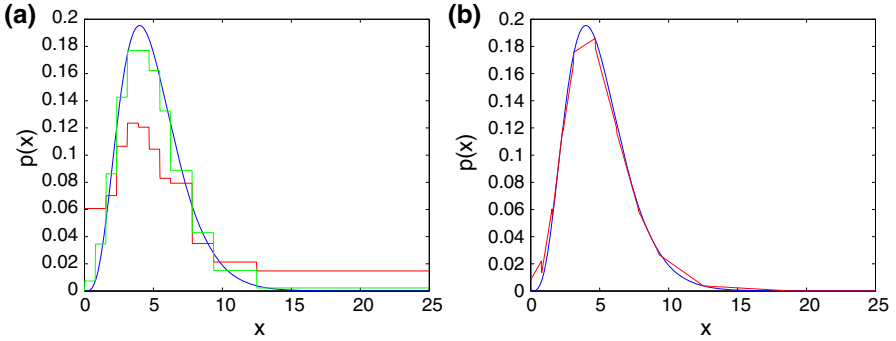

Fig. 1 Approximations of a gamma density constructed using auto-validating methods. **a** The gamma density with scale 1 and shape 5 (in *blue*), with an approximation by the standard-interval method in *red* and an approximation produced by the standard-affine method. Both approximations are based on refining the domain 10 times. **b** The gamma density with scale 1 and shape 5 (in *blue*), with an approximation (based on 10 refinements of the domain) by the fully-affine auto-validating sampler in *red* (color figure online)

$$p(x) = \exp(4\log(x) - x - \log(\Gamma(5))). \tag{2}$$

The affine extension of Eq. 2 gives improved bounds over the affine extension of Eq. 1 since the addition of the terms $4\log(x)$ and $-x$ in Eq. 2 allows some resolution of the dependency problem, whereas the multiplication of x^4 and $\exp(-x)$ in Eq. 1 does not (since it is a non-affine operation).

In this problem the affine approaches outperform the standard-interval approach. Figure 1a shows the approximations constructed by the standard-interval approach (with 10 refinements of the initial domain, $[0.001, 25]$), and the same standard-affine approach. Here we see clearly the advantage of using mixed interval-affine arithmetic. Figure 1b shows an approximation (again for 10 refinements of the domain) found by the fully-affine approximation.

Figure 2 shows the results of applying importance sampling to the gamma example, with proposals constructed via the three auto-validated approaches. Figure 2a–c shows, respectively, the ESS, the estimated mean and the estimated variance, against the number of refinements of the domain used to find the approximation (the true mean and variance are both 5). The standard-interval (red) is outperformed by the standard-affine approach (green). The results illustrate the benefit of using affine arithmetic over standard interval arithmetic in the standard approach, with both of these methods being outperformed by the fully-affine approach.

4.2 Mixture of Gaussians

We now consider simulating points from a mixture of Gaussians:

$$\pi(x) = \mathcal{N}(x; -20, 2) + \mathcal{N}(x; 20, 0.1). \tag{3}$$

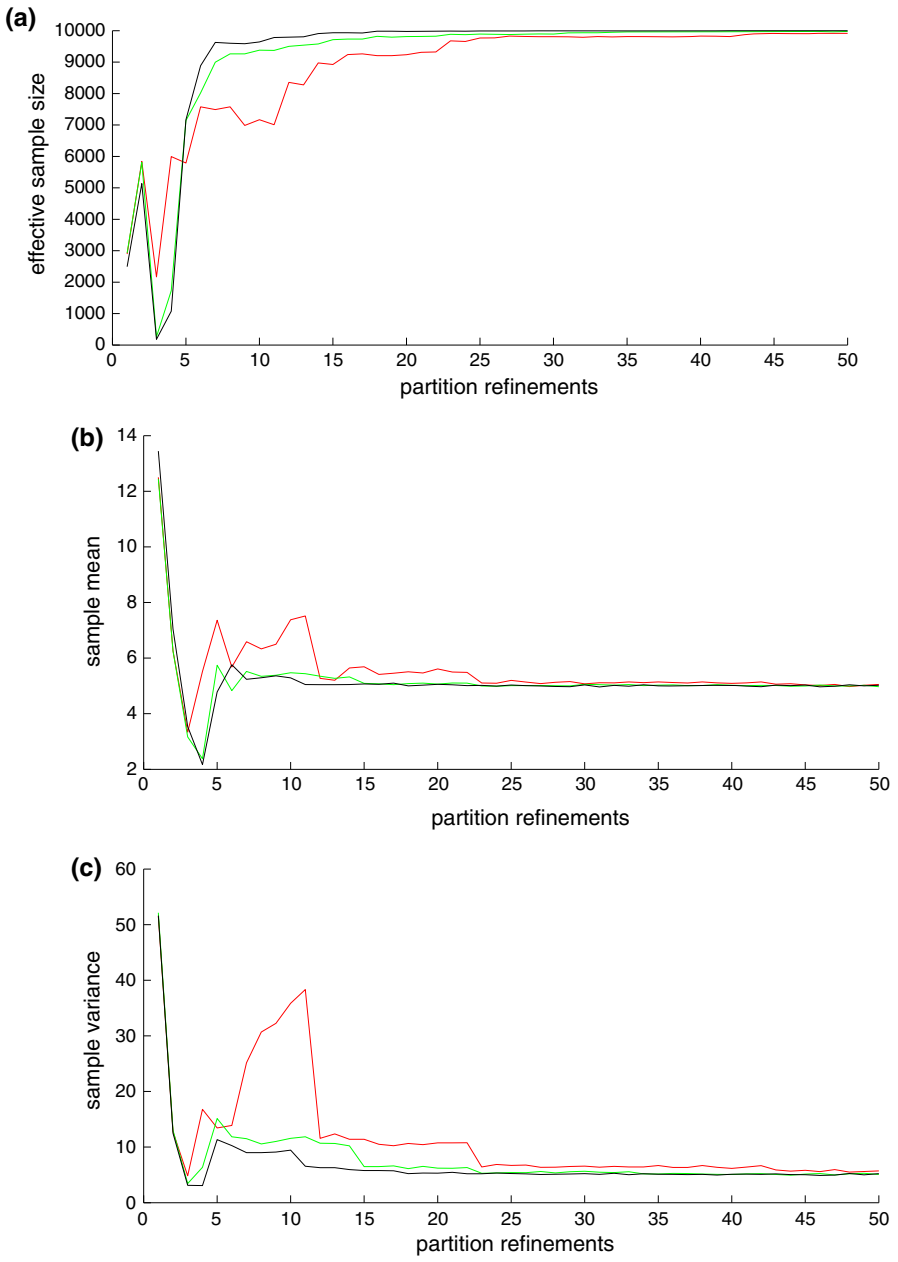

Fig. 2 Auto-validating importance sampling, using 10,000 points, applied to a gamma density. **a** Effective sample size against number of partition refinements when using the standard-interval (in *red*), standard-affine (in *green*) and fully-affine (in *black*) approximations. **b** Estimated mean against number of partition refinements when using the standard-interval (in *red*), standard-affine (in *green*) and fully-affine (in *black*) approximations. **c** Estimated variance against number of partition refinements when using the standard-interval (in *red*), standard-affine (in *green*) and fully-affine (in *black*) approximations (color figure online)

Again, we can simulate directly from this density. However, it exhibits a characteristic that leads to poor efficiency in some Monte Carlo simulation methods: it contains a mode with small support in relation to the gap between the modes. In this case it is not difficult to design a Monte Carlo algorithm to simulate points from both modes, but it is a useful example for illustrating how auto-validating samplers deal with this problem very efficiently.

Figure 3a shows an approximation constructed by the standard-interval approach (with 20 refinements of the initial domain, $[-100, 100]$), with Fig. 3b showing the approximation (again for 20 refinements of the domain) found by the fully-affine approach. Figure 3c shows the ESS (using 10,000 points) of IS against the number of refinements of the domain used to find the approximation. A better ESS is achieved for the fully-affine approach. with both approaches achieving a very high ESS with relatively few partitions.

4.3 The posterior of the means of a mixture of Gaussians

Consider the problem of estimating the parameters of a mixture of Gaussian distributions. Let $\{y_j\}_{j=1}^{M}$ be a data set with $y_j \in \mathbb{R}$. We use the model that the y_j are i.i.d. with density:

$$f\left(y|\mu_1, \mu_2, \Sigma_1, \Sigma_2, w_1, w_2\right) = w_1\mathcal{N}(y; \mu_1, \Sigma_1) + w_2\mathcal{N}(y; \mu_2, \Sigma_2),$$

where $\mathcal{N}(y; \mu_k, \Sigma_k)$ denotes a normal density on y with mean μ_k and variance Σ_k. We consider the case where the Σ_k and w_k, are fixed. The target function we consider is be the posterior:

$$\pi\left(\mu_1, \mu_2|y\right) \propto p\left(\mu_1, \mu_2\right) \prod_{j=1}^{M} f\left(y|\mu_1, \mu_2, \Sigma_1, \Sigma_2, w_1, w_2\right). \tag{4}$$

We choose $p\left(\mu_1, \mu_2\right)$ to be a multivariate Gaussian distribution centred at $(0, 0)$ with covariance $10^9 I$. This example is of interest since the large product of terms means the variables μ_1 and μ_2 occur many times in a naive representation of the function, and over-enclosure (due to the dependency problem) potentially becomes influential.

Consider a posterior based on 10 points from the two component model, where the true means of the components are 0 and 3 and the variance of both components is 0.5^2. The true posterior is shown in Fig. 4a, with Fig. 4b showing an approximation constructed by the standard-interval approach (with 900 refinements of the original domain of $[-50, 50] \times [-50, 50]$) and with Fig. 4c showing the approximation (again for 900 refinements of the domain) found by the fully-affine approach. The standard-interval approximation yields an ESS of 9076 whilst the fully affine approach yields an ESS of 9274.

Note that we require many more partition refinements to obtain a high ESS than in the previous examples. This could be due either to this being a bivariate rather than a univariate example, or to an inefficient partition method due to a large over-enclosure.

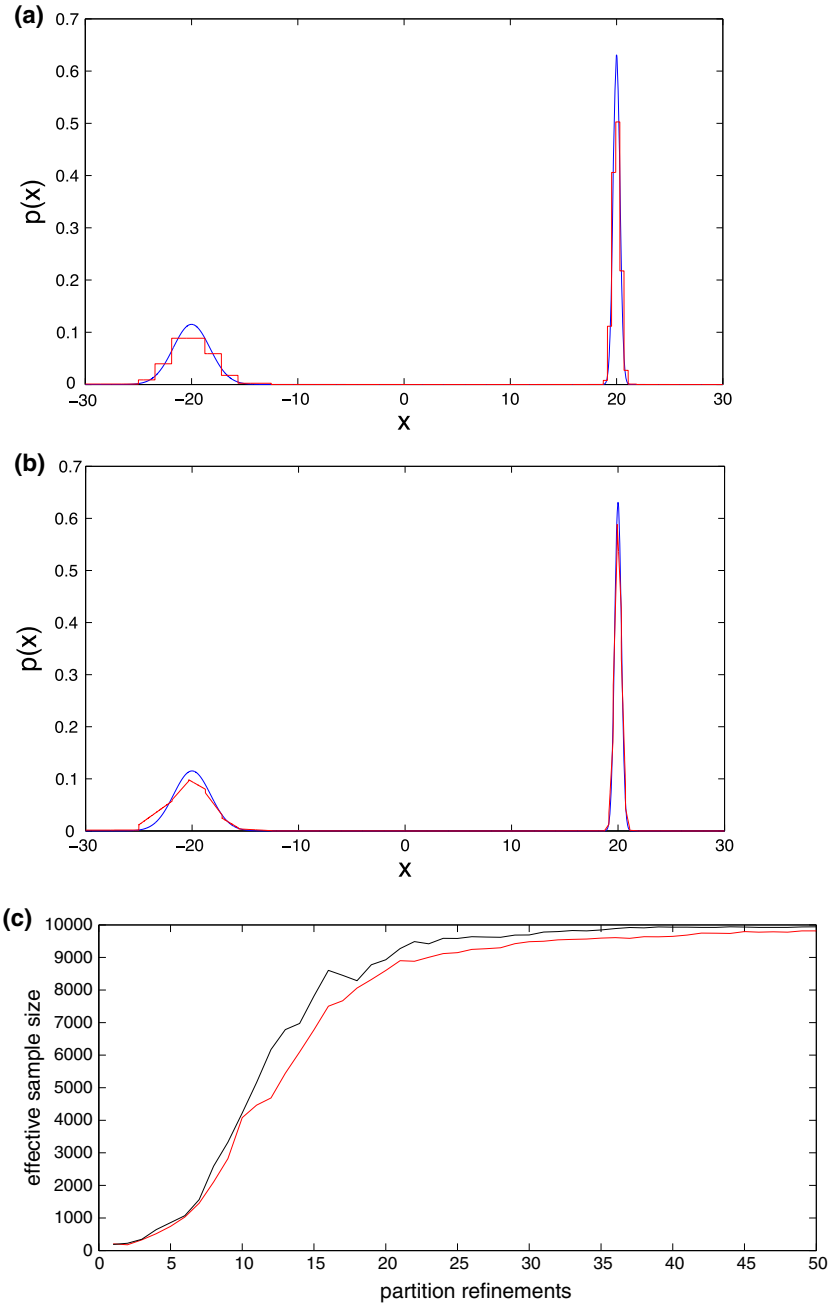

Fig. 3 Approximations of a mixture of Gaussians constructed using auto-validating methods. **a** The mixture density in Eq. 3, with an approximation by the standard-interval method based on refining the domain 20 times. **b** The mixture density in Eq. 3, with an approximation by the fully-affine method based on refining the domain 20 times. **c** Effective sample size against number of partition refinements for the standard-interval (in *red*) and fully-affine (in *black*) approximations, for the mixture example (color figure online)

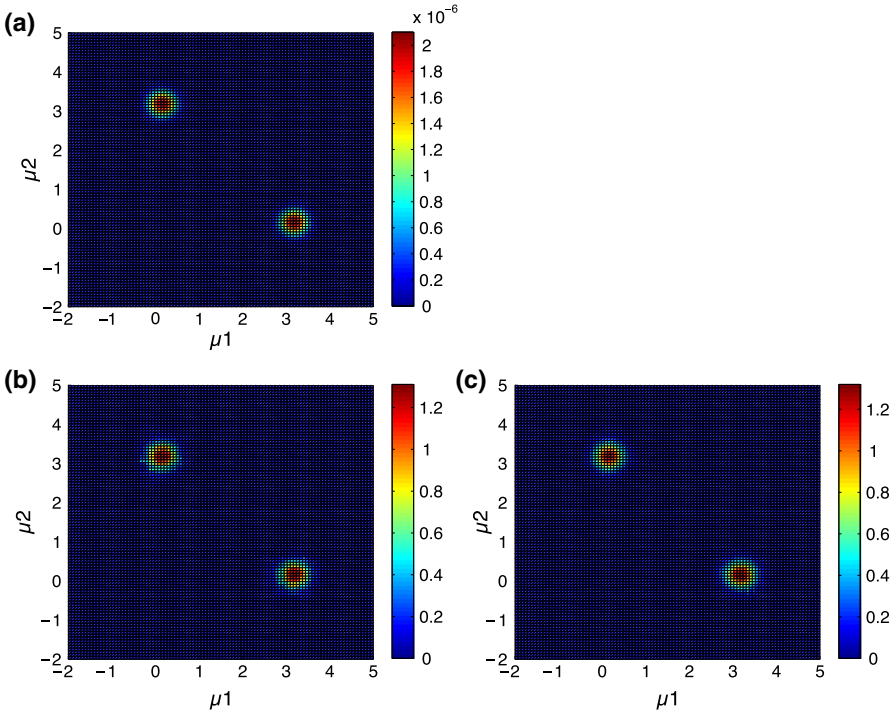

Fig. 4 Approximations of a posterior distribution of the means of mixture of Gaussians constructed using auto-validating methods. **a** The (unnormalised) posterior of the means of a two component mixture of Gaussians. **b** An approximation by the standard-interval method based on refining the domain 900 times, of the posterior of the means of a two component mixture of Gaussians. **c** An approximation by the fully-affine method based on refining the domain 900 times, of the posterior of the means of a two component mixture of Gaussians

We first examine how the size of the representation of the posterior (which is linear in the number of data points) affects the number of partitions we require. We do this simply by changing the number of terms in the likelihood. More than a few terms in the likelihood results in a very large over-enclosure by the interval extension.

For a likelihood based on 50 data points in the same two component example, the standard-interval method finds an approximation that gives an ESS of 5292 after 600 partition refinements. For 100 data points 600 refinements provides an approximation that gives an ESS of 5080. A larger number of terms makes the peaks in the likelihood narrower, so we expect a lower ESS for a fixed partition size for a larger representation. It is likely that this effect largely accounts for the effect that we see, and that the over-enclosure because of the larger representation has little effect. This conclusion is supported by other experiments conducted using this likelihood with different numbers of data points. The lack of dependence on the size of the representation in this case is probably because the over-enclosure is at its worst where the mass of the posterior is found (although we note this is not likely be true for every target function).

This lack of dependence on the size of the the representation is in contrast to the results presented in Sainudiin and York (2009) where the performance of the Moore rejection sampler on a likelihood is very poor. The cause of this is probably the use of rejection sampling rather than importance sampling. When we use importance sampling, we can normalise the approximating proposal, so it is only the relative heights of the uniform components in the approximation that matter. However, when used as a dominating function in rejection sampling the over-enclosures are very significant and lead to a high rejection rate.

To investigate the effect of the dimension of the target, we used the standard-interval approach on the posterior of the means of a 3 component mixture model (based on 15 data points, 5 points from each of three components, with the same variances as before, and with true means of 0, 3 and 6). On this example it takes 20,000 partition refinements to obtain an ESS of 6608 (compared with 480 refinements to obtain an ESS of 6502 in the 2-d case). This result gives some evidence for requiring an exponential increase in refinements with the increase in dimension.

4.4 Conclusions

Based on the empirical investigation in this paper, and taking into account the previous work of Sainudiin (2005), Sainudiin and York (2009), we make the following conclusions about the auto-validating samplers presented in this paper.

- For a low-dimensional elementary target function (sometimes up to 10 dimensions), the methods in this paper give a way to simulate points from the target with a computational cost that is small compared to many other approaches. The use of interval arithmetic ensures that multi-model target functions present few problems. The method also requires little or no tuning.
- The approximating proposals are found prior to sampling, which may be useful in some situations. The approximations may also be used directly to give numerical approximations to integrals.
- Over-enclosure of a target function (such as that observed with likelihood functions) is not necessarily a barrier to using the approach. The approximating proposal can sometimes be used profitably in importance samplers and independent MCMC algorithms. However, the original approach of finding a dominating proposal for use in a rejection sampler is severely affected.
- The new fully-affine method often yields a more accurate approximating density than the original standard-interval approach, for the same number of partitions of the domain.

Auto-validating methods are not widely used in statistics, although there has been much recent interest in a related field of automatic differentiation; another field in which the representation of a function is exploited.

Appendix: An introduction to auto-validating methods

Interval analysis

In this section we introduce the basic concepts of interval analysis, the original auto-validating method. Most of our definitions are taken from Tucker (2005) and Sainudiin (2005). The most important result in interval analysis from our perspective concerns a computationally cheap way of finding an interval $I_R \subset \mathbb{R}$ such that $f(I_D) \subseteq I_R$, for a large class of functions f. This section is devoted to explaining how to achieve this result.

Definitions

The interval extension of the real numbers, $\mathbb{IR}$, is the set of compact intervals in $\mathbb{R}$: $\mathbb{IR} := \{[a, b] \mid a, b \in \mathbb{R}, a < b\}$. We define the interval extension of a subset $D \subset \mathbb{R}$ to be $\mathbb{I}D := \{[a, b] \mid a, b \in D, a < b\}$. Interval analysis allows us to find a bound on the range of a function $f : D \subset \mathbb{R} \to \mathbb{R}$ over an interval $I \in \mathbb{I}D$ through the use of a function $F : \mathbb{I}D \to \mathbb{IR}$. We obtain such a bound if F satisfies *range enclosure*: if $\forall X \in \mathbb{I}D$, $f(X) \subseteq F(X)$. Ideally, F will not over-enclose f: it will satisfy *range equivalence*, so that $f(X) = F(X)$. In this section, for any f that is a composition of commonly used functions, we give the definition of a function F that satisfies range enclosure and also *inclusion isotony*: $\forall X, X' \in \mathbb{I}D$ with $X \subseteq X'$, $F(X) \subseteq F(X')$. We say that a function $F : \mathbb{I}D \to \mathbb{IR}$ that satisfies these two properties is called an interval extension of f.

The interval extension that we define is a "natural" interval extension. To do this we need to consider a "representation" of f: that is, we need to express it as a composition of its constituent operations. We then define the natural interval extension to be the composition of natural interval extensions of each of these simple functions. This idea of operating on a function "component-wise" is also used in other subjects, notably *automatic differentiation*, where a computer automatically finds the expression for the derivative of a function by applying the chain rule of differentiation. We first introduce an arithmetic on $\mathbb{IR}$, then define a set of "standard" functions, and go on to define a class of functions that are compositions of these operations. We denote the minimum value of an interval $X \in \mathbb{IR}$ by $\underline{x}$ and its maximum by $\overline{x}$.

Each of the arithmetic operators $\{+, -, \times, /\}$ on $\mathbb{IR}$ is defined such that the output interval covers the entire range of possible outcomes yielded by applying the equivalent operator on $\mathbb{R}$ to any members of the input intervals. For example, for $X, Y \in \mathbb{IR}$, addition is defined as, $X + Y = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$ and multiplication $X \times Y = [\min\{\underline{x}\,\underline{y}, \underline{x}\,\overline{y}, \overline{x}\,\underline{y}, \overline{x}\,\overline{y}\}, \max\{\underline{x}\,\underline{y}, \underline{x}\,\overline{y}, \overline{x}\,\underline{y}, \overline{x}\,\overline{y}\}]$. We define the operation of a

set of *standard* functions on $\mathbb{IR}$ in an analogous way. Let the set standard functions be

$$\mathcal{G} = \{a^x, \log_b(x), x^a, ax, x + a, |x|, \sin(x), \cos(x), \tan(x), \sinh(x), \cosh(x), \tanh(x), \arcsin(x), \arccos(x), \arctan(x)\},$$

where $a \in \mathbb{R}$ and $b \in \mathbb{R}_{\geq 0, \neq 1}$ are constants. For each of the functions in $\mathcal{G}$ that is monotone, the interval extension can be defined in terms of the boundary points of an input interval (just as for the arithmetic operations). For example, for $X \in \mathbb{IR}$:

$$\exp(X) = [\exp(\underline{x}), \exp(\overline{x})].$$

It is also simple to define interval extensions of the members of $\mathcal{G}$ that are not monotone but piecewise monotone (e.g. $\cos(x)$). It is easy to see that functions on $\mathbb{IR}$ defined in this way satisfy both inclusion isotony and range equivalence.

The set of *elementary functions*, $\mathcal{E}$, is defined on $f : D \subseteq \mathbb{R} \to \mathbb{R}$ is a function that may be expressed as a finite combination of constants, variables, arithmetic operations, standard functions and compositions. We refer to a particular expression of $f \in \mathcal{E}$ as a finite combination of constants, variables, arithmetic operations, standard functions and compositions as a *representation* for f. Every member of $\mathcal{E}$ has an infinite number of representations. For example $f(x) = 1 - \frac{1}{x+1}$ can be rewritten as $f(x) = \frac{x}{x+1} + 5 - 5$. A representation may be drawn as a graph, in which each operation is a node. We refer to the number of nodes in this graph as the *size* of the representation.

The *natural interval extension* of an elementary function $f \in \mathcal{E}$ depends on the representation of the function: it is defined to be the function $F : \mathbb{I}D \to \mathbb{IR}$ where every constant, variable, operation and standard function in the representation of f is replaced by its interval counterpart.

The Fundamental Theorem of Interval Analysis, whose proof is in Moore (1966), gives the important result that natural interval extensions of elementary functions satisfy inclusion isotony and range enclosure. In general, range equivalence is not satisfied and the size of the over-enclosure depends on the representation of f that is used. We discuss over-enclosure in Sect. 1, but first we extend the definition to multi-dimensional elementary functions.

Define $\mathbb{IR}^n$ to be the set of compact *boxes* (Cartesian products of intervals) in $\mathbb{R}^n$:

$$\mathbb{IR}^n := \{(X_1, \ldots, X_n) \mid \text{for } i = 1, \ldots, N, X_i = [a_i, b_i] \text{ with } a_i, b_i \in \mathbb{R}, a_i < b_i\}.$$

It is trivial to extend our definition of an elementary function to include functions with a domain $D \subseteq \mathbb{R}^m$ for any $m \in \mathbb{N}$. The multivariate version of the Fundamental Theorem of Interval Analysis also holds in this setting.

Over-enclosure

In general, a natural interval extension of $f \in \mathcal{E}$ will over-enclose f. We first give some general results about the extent of the over-enclosure of elementary functions,

then discuss the *dependency problem*, which is the main cause of over-enclosure in practice.

Definition 3 Order of an interval extension.

We say that an interval extension $F : \mathbb{IR}^m \to \mathbb{IR}^n$ of a representation of $f : \mathbb{R}^m \to \mathbb{R}^n$ is of order α if there exists a constant K, independent of the box X where:

$$w(F(X)) - w(f(X)) \leq K w(X)^{\alpha}$$

with $w(X)$ denoting the maximum of each the widths of each of the dimensions of the box X.

Natural interval extensions of elementary functions, as defined above, are of order 1 (Baker Kearfott 1996), meaning that the size of an over-enclosure increases linearly with the volume of the box in the domain. In the literature on interval methods it is usually advantageous to use an interval extension that has the largest order possible, since the behaviour of the interval extension as the size of the domain decreases is usually of interest. The result that natural interval enclosures are first-order is significant for the applications we consider: it tells us that the over-enclosure is independent of the dimension of the domain.

The extent of the over-enclosure of a function $f \in \mathcal{E}$ by a natural interval extension F depends on the representation used for f. Over-enclosure does not occur if each variable only occurs once in a representation for a function. When the same variable occurs more than once, an over-enclosure due to the dependency problem is often the result. A simple illustrative example is evaluating the interval extension of the function $f(x) = x \times x$ on the interval $[-1, 1]$. We obtain a bound on the range of $[-1, 1]$. We do not obtain the true range of $[0, 1]$, since we do not acknowledge that the arguments of the $\times$ operation are the same (if we used the function x^2 we would not have this problem). To avoid the dependency problem and thus minimise over-enclosure, a good "rule of thumb" is to use a representation that minimises the number of occurrences of each variable.

This lack of acknowledgement of dependency is a major limitation of basic interval arithmetic. It is one of the primary motivations for the development of affine arithmetic defined in the following section.

An introduction to affine arithmetic

Introduction

In Sect. 1 we introduce the *affine form* and run through the basics of using affine arithmetic as an alternative to interval analysis, then consider the main problem with affine arithmetic: the overshoot problem. We then briefly outline how to use affine arithmetic to construct approximating hyperplanes in Sect. 2.2.3, and compare its computational cost with interval arithmetic in Sect. 2.2.4.

The basic principles of affine arithmetic

Affine arithmetic is an extension of interval arithmetic, where the use of an interval is replaced by a different object; an *affine form*. In this section we define affine forms and the affine extension of the real numbers.

An affine form $\hat{x}$ is defined to be a first degree formal polynomial in $\{\epsilon_i\}_{i=1}^n$:

$$\hat{x} = x_0 + x_1\epsilon_1 + \cdots + \cdots x_n\epsilon_n,$$

where the coefficients x_i are in $\mathbb{R}$ and the ϵ_i are all different instances of the interval $[-1, 1] \in \mathbb{R}$, each with a different index i.

Each of the ϵ_i can be thought of as different "sources of variability" in the affine form. The ϵ_i are often referred to as "noise terms" in the literature. The affine extension, $\mathbb{AR}$, of $\mathbb{R}$ is defined to be:

$$\mathbb{AR} := \{x_0 + x_1\epsilon_1 + \cdots + \cdots x_n\epsilon_n \mid n \in \mathbb{N} \cup \{0\}, x_i \in \mathbb{R}, \epsilon_i = [-1, 1]\}.$$

A multivariate version of affine forms can be defined in an analogous way to the multivariate generalisation of intervals described in the previous section.

In practice, we use an affine form as an alternative representation of an interval. If we define "affine extensions" of arithmetic and standard operations that have affine forms as their inputs and outputs, then analogous to the natural interval extension we may define the natural affine extension $F_A : \mathbb{AR} \to \mathbb{AR}$ of the representation of an elementary function $f : \mathbb{R} \to \mathbb{R}$. The idea is that the use of affine forms can allow us to keep track of dependencies between two or more quantities, helping us to avoid the dependency problem and so reduce over-enclosure.

It can be shown that an analogous form of the Fundamental Theorem of Interval Arithmetic holds for affine extensions. The remainder of this section describes how this is achieved, and describes how to implement an affine extension, by addressing the following points:

- converting an interval into an affine form (to find the inputs to the affine extension);
- converting an affine form into an interval (to express the output of the affine extension as an interval);
- giving the affine extensions of arithmetic and standard operations.

Consider a representation of an elementary function $f : \mathbb{R}^d \to \mathbb{R}$ with d inputs with natural affine extension F_A. Suppose we wish to evaluate the affine extension on the input $(I_1, \ldots, I_d)$ where each $I_j = [a_j, b_j] \in \mathbb{IR}$. In this case we may convert each interval I_j to an affine form A_j as follows:

$$A_j = 0.5(a_j + b_j) + 0.5(b_j - a_j)\epsilon_j.$$

Note that we use a different noise term ϵ_j for each input interval. In some cases, we may know that the inputs are dependent. For example, we may know that $I_2 = f(I_1)$, where f is some monotone function. In this, case we would express this dependence

through the noise terms and convert I_1 and I_2 to affine forms as follows:

$$A_1 = 0.5(a_1 + b_1) + 0.5(b_1 - a_1)\epsilon_1,$$
$$A_2 = 0.5(f(a_1) + f(b_1)) + 0.5(f(b_1) - f(a_1))\epsilon_1.$$

Expressing this dependence explicitly allows us to avoid the dependency problem between I_1 and I_2.

Converting an affine form to an interval is simple: we simply use the interval addition and multiplication operations on the affine form to amalgamate each of the noise terms ϵ_i. The result of this operation in general is (using the representation of an affine form with n noise terms from above):

$$X = \left[x_0 - \sum_{i=1}^{n} |x_i|, x_0 + \sum_{i=1}^{n} |x_i| \right].$$

Affine extensions of affine transformations (addition, subtraction, multiplication by a constant and addition of a constant) are easy to define. The important characteristic of these transformations is that the result can also be expressed as an affine form. As an example, the affine extension of the subtraction operator is defined as follows for inputs $X = x_0 + x_1\epsilon_1 + \cdots + \cdots x_n\epsilon_m$ and $Y = y_0 + y_1\epsilon_1 + \cdots + \cdots y_n\epsilon_n$ (where $m > n$):

$$\begin{aligned} X - Y &= x_0 + x_1\epsilon_1 + \cdots + \cdots x_m\epsilon_m - (y_0 + y_1\epsilon_1 + \cdots + \cdots y_n\epsilon_n) \\ &= (x_0 - y_0) + (x_1 - y_1)\epsilon_1 + \cdots + (x_n - y_n)\epsilon_n + x_{n+1}\epsilon_{n+1} + \cdots + x_m\epsilon_m. \end{aligned}$$

The important point about this operation is that if two affine forms contain the same source of variability (the coefficients of ϵ_i are non-zero in both inputs for some i) this is recognised when performing the operation, thus we avoid the dependency problem.

The result of applying a non-affine transformation f to an affine form $\hat{x}$ with n noise terms cannot be expressed as an affine form containing only these n terms. The approach taken in affine arithmetic is to find an affine form in these n terms, $f^a(\hat{x})$, that approximates the true answer plus an error term $x_k\epsilon_k$. The error term is introduced to ensure that the answer bounds the range of f. The coefficient x_k is the maximum distance between the approximation f^a and the true f and the new noise term ϵ_k is not used in any other affine form. ϵ_k is actually a function of $\epsilon_1, \ldots, \epsilon_n$, however no operations from this point on are "aware" of this, and ϵ_k is treated as an independent source of variability. The consequence of this is that subsequent operations are affected by the dependency problem to some degree, since there is an unseen dependence of ϵ_k on every other noise term.

If a composition of many non-affine operations is performed on an affine form, or if the approximating affine form f^a is poor and gives a large error, significant dependency on the sources of variability in the input is lost and affine arithmetic offers little advantage over interval arithmetic.

The choice of approximation $f^a(\hat{x})$ is significant: any approximation is valid, as long as we can find the error between the approximation and the true function. For the

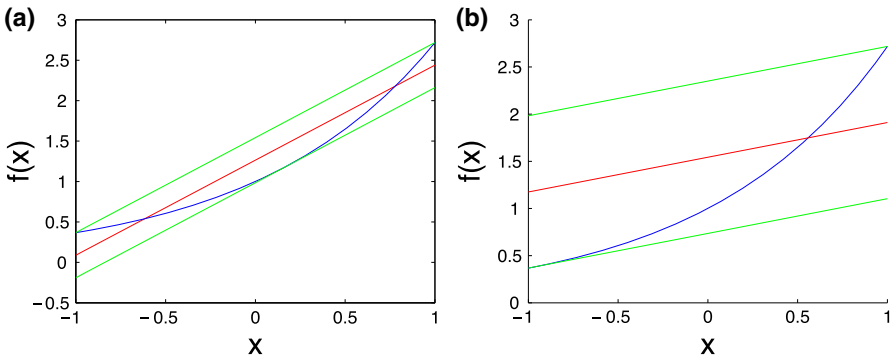

Fig. 5 Approximations of the exponential function (*blue*) on $[-1, 1]$. **a** The Chebyshev approximation (in *red*) and its error bounds (*green*). **b** The min-range approximation (in *red*) and its error bounds (*green*) (color figure online)

approximation to be of any use in practice we need to be able to find the approximation and its error in a computationally efficient manner (preferably with a linear cost in terms of the number of noise terms of the input) and for the error of the approximation to be as simple as possible. we discuss the two most important choices of approximation (Stolfi and de Figueiredo 1997), the *Chebyshev* (or *minimax*) and *min-range* approximations, then discuss the error resulting from the approximation of a non-affine operation.

The Chebyshev approximation is defined to be the affine function that minimises the maximum absolute error of the approximation over an interval I in the domain. Finding the approximation and the size of the error is trivial using results from Chebyshev approximation theory (Stolfi and de Figueiredo 1997)—the only restrictions are that the function to be approximated is bounded, twice differentiable and that the sign of the second derivative does not change on I. A plot of the Chebyshev approximation for $\exp(x)$ on $[-1, 1]$ is shown in Fig. 5a.

The Chebyshev approximation is the optimal choice in terms of minimising the error, which means that we have the best possible information about the way the output depends on the ϵ in the input. This means that in subsequent operations we have the best chance of combating the dependency problem. However, Fig. 5a illustrates clearly that the Chebyshev approximation actually gives a worse bound on the range of $\exp(x)$ than interval arithmetic (in some cases it may even give a negative lower bound)! This is known as the *overshoot problem*.

To counter the overshoot problem, rather than the Chebyshev approximation, we can use the affine approximation that yields the minimum range of the output: the min-range approximation. This approximation and its error are also easy to compute (Stolfi and de Figueiredo 1997) and we avoid the overshoot problem. Figure 5b shows the min-range approximation for $\exp(x)$ on $[-1, 1]$.

Any terms that are not linear in the noise of the inputs are amalgamated into an error term in affine arithmetic. As the size of the input intervals decreases the quadratic terms in the error will dominate. Thus the error of an affine approximation is quadratic in the size of the input intervals (Stolfi and de Figueiredo 1997). This gives it an advantage over interval arithmetic (where the error is linear) for small intervals. However, as the

size of the input intervals increases there will be a point where the benefits of using affine arithmetic are lost because of the increased computation time required by affine arithmetic.

Finally, Stolfi and de Figueiredo (1997) suggests a way to use the Chebyshev approximation (so that we do our best to avoid the dependency problem) and still avoid the overshoot problem. Their suggestion is to combine interval arithmetic and affine arithmetic (this does not add a large computational cost to using affine arithmetic alone). This suggestion means more than performing the interval and affine operations in parallel for each component operation in some function f, then picking the intersection of the results at the end. The best interval at the end of each operation is the intersection of the results obtained by the interval and affine methods, and the construction of an affine approximation for the next operation can be based on this intersection.

References

Agapiou S, Papaspiliopoulos O, Sanz-Alonso D, Stuart AM (2017) Importance sampling: computational complexity and intrinsic dimension. Stat Sci (**to appear**)

Andrieu C, Roberts GO (2009) The pseudo-marginal approach for efficient Monte Carlo computations. Ann Stat 37(2):697–725

Baker Kearfott R (1996) Rigorous global search: continuous problems. Springer, Berlin

Chatterjee S, Diaconis P (2017) The sample size required in importance sampling. arXiv:1511.01437

Chopin N, Ridgway J (2017) Leave Pima Indians alone: binary regression as a benchmark for Bayesian computation. Stat Sci 32:64–87

de Figueiredo LH, Stolfi J (2004) Affine arithmetic: concepts and applications. Numer Algorithms 37(1–4):147–158

Del Moral P, Doucet A, Jasra A (2006) Sequential Monte Carlo samplers. J R Stat Soc Ser B 68(3):411–436

Enger W (1992) Interval ray tracing—a divide and conquer strategy for realistic computer graphics. Vis Comput 9(2):91–104

Everitt RG, Culliford R, Medina-Aguayo F, Wilson D (2016) Sequential Bayesian inference for mixture models and the coalescent using sequential Monte Carlo samplers with transformations. arXiv:1612.06468

Everitt RG, Johansen AM, Rowing E, Evdemon-Hogan M (2017) Bayesian model comparison with unnormalised likelihoods. Stat Comput 27(2):403–422

Geweke J (1989) Bayesian inference in econometric models using monte carlo integration. Econometrica 57(6):1317–1339

Gilks WR, Best NG, Tan KKC (1995) Adaptive rejection metropolis sampling within Gibbs sampling. J R Stat Soc Ser C 44(4):455–472

Jaulin L, Kieffer M, Didrit O, Walter E (2001) Applied interval analysis: with examples in parameter and state estimation, robust control and robotics. Springer, Berlin

Kong A, Liu JS, Wong WH (1994) Sequential imputations and bayesian missing data problems. J Am Stat Assoc 89(425):278–288

Kuncir GF (1962) Algorithm 103: Simpson's rule integrator. Commun ACM 5(6):347

Lenormand M, Jabot F, Deffuant G (2013) Adaptive approximate Bayesian computation for complex models. Comput Stat 28(6):2777–2796

Marrelec G, Benali H (2004) Automated rejection sampling from product of distributions. Comput Stat 19(2):301–315

Martino L, Read J (2013) On the flexibility of the design of multiple try metropolis schemes. Comput Stat 28(6):2797–2823

Moore RE (1962) Interval Arithmetic and automatic error analysis in digital computing. Ph.D. thesis, Stanford University

Moore RE (1966) Interval analysis. Prentice-Hall, New York

Neal RM (2001) Annealed importance sampling. Stat Comput 11(2):125–139

Sainudiin R (2005) Machine interval experiments. Ph.D. thesis, Cornell University, Ithaca
Sainudiin R, York T (2009) Auto-validating von Neumann rejection sampling from small phylogenetic tree spaces. Algorithms Mol Biol 4(1):1
Stolfi J, de Figueiredo LH (1997) Self-validated numerical methods and applications. In: Brazilian mathematics colloquium
Tran MN, Scharth M, Pitt MK, Kohn R (2014) Importance sampling squared for Bayesian inference in latent variable models. arXiv:1309.3339
Tucker W (1999) The Lorenz attractor exists. Comptes Rendus de l'Académie des Sciences - Series I - Mathematics 328(12):1197–1202
Tucker W (2005) Validated numerics for pedestrians. In: European congress of mathematics 4
Walker AJ (1977) An efficient method for generating discrete random variables with general distributions. ACM Trans Math Softw 3(3):253–256