Sequential Monte Carlo Optimization and Statistical Inference

August 15, 2022

(to appear in *Wiley Interdisciplinary Reviews: Computational Statistics)*

Article Category

- Advanced Review

Authors

- **Jin-Chuan Duan**: National University of Singapore (Asian Institute of Digital Finance and Business School). Email: bizdjc@nus.edu.sg.
- **Shuping Li**: National University of Singapore (Asian Institute of Digital Finance). Email: shuping.li@u.nus.edu.
- **Yaxian Xu**: National University of Singapore (Asian Institute of Digital Finance). Email: yaxianxu@nus.edu.sg.

Conflict of Interest

The authors have declared no conflicts of interest for this article.

Abstract

Sequential Monte Carlo (SMC) is a powerful technique originally developed for particle filtering and Bayesian inference. As a generic optimizer for statistical and non-statistical objectives, its role is far less known. Density-tempered SMC is a highly efficient sampling technique ideally suited for challenging global optimization problems and is implementable with a somewhat arbitrary initialization sampler instead of relying on a prior distribution. SMC optimization is anchored at the fact that all optimization tasks (continuous, discontinuous, combinatorial, or noisy objective function) can be turned into sampling under a density or probability function short of a norming constant. The point with the highest functional value is the SMC estimate for the maximum. Through examples, we systematically present various density-tempered SMC algorithms and their superior performance vs. other techniques like MCMC. Data cloning and k-fold duplication are two easily implementable accuracy accelerators, and their complementarity is discussed. The Extreme Value Theorem on the maximum order statistic can also help assess the quality of the SMC optimum. Our coverage includes the algorithmic essence of the density-tempered SMC with various enhancements and solutions for (1) a bi-modal non-statistic

function without and with constraints, (2) a multidimensional step function, (3) offline and online optimization, (4) combinatorial variable selection, and (5) non-invertibility of the Hessian.

1 Introduction

Sequential Monte Carlo (SMC) is, as the name suggests, a family of Monte Carlo sampling techniques that sequentially adapt to the target density/probability function by moving/concentrating the simulated sample to the most desirable part of its support. SMC has its origin in the seminar work of Gordon et al. (1993) in designing a particle filter and has found its place in Bayesian inference as in Chopin (2002). Conceptually, it is fairly easy to understand its applications in the Bayesian statistics because moving from the prior to posterior distribution by SMC fits naturally to the context. However, using SMC for optimization is far less known and will need explanations. Before elaborating, we will make a perhaps provocative statement that all optimization tasks (continuous, discontinuous, combinatorial, or noisy objective function) can be turned into sampling problems where SMC can be applied.

An optimization objective can always be converted to a positive function with, say, exponentiation, without changing the optimizing point. Minimization can also be easily changed to maximization by applying a negative sign to the exponent. Therefore, the transformed objective function becomes a pseudo density/probability because it is a density, probability or mixed function just short of the norming constant. Sampling without needing to know this unknown norming constant holds the key, and for which importance sampling is such a technique. Suppose that a good-quality sample is generated under this pseudo density/probability function, the point with the highest functional value becomes the Monte Carlo solution for the maximum. Beyond this rather generic concept, the implementation challenge is to conduct importance sampling effectively and efficiently. Density-tempered sequential Monte Carlo serves this role perfectly. We will provide a systematic exposition on the theoretical concept and practical implementations in this article.

We limit the scope of this review to optimization with density-tempered SMC and contrast it, whenever possible, with another SMC method and the Markov Chain Monte Carlo (MCMC) approach, its close relative. A related focal issue is statistical inference of these two sampling techniques, reflective of their methodological origin. Optimization is an extremely large subject and has a long history. It is simply impossible to present a comprehensive review on optimization in an article of this length. Gradient-based methods historically dominate the field and their power and limitations are well understood. There are also many meta-heuristic optimization methods that have emerged over the years, including simulation-based methods such as simulated annealing and genetic algorithms. We refer readers to a book by Spall (2003) for a comprehensive coverage on many of those conventional optimization methods.

SMC algorithms should not be misunderstood as meta-heuristic because the inherent property of Monte Carlo sampling ensures convergence of the maximum order statistic to the right theoretical quantity as the sample size gets large. Typical Monte Carlo methods are used to estimate some functional, e.g, computing the mean of some functional transformation of one or many random variables, and through the Central Limit Theorem the quality of the estimate can be assessed. In the context of optimization discussed in this paper, Duan (2019) has shown that the Fisher-Tippett-Gnedenko Extreme Value Theorem can be used to provide the quality assurance of the SMC maximum.

Although particle filtering also uses SMC, that large literature is not the focus of this article. In places of discussing SMC^2 (Chopin et al., 2013, Fulop and Li, 2013, Duan and Fulop, 2015, Duan et al., 2020, Jasra et al., 2021), the outer-layer SMC is used to find the parameter optimum whereas the inner-layer SMC, i.e., a particle filter, is used to compute the fixed-parameter likelihood function for, say, a state-space model. In short, our focus remains at the out-layer SMC specifically for optimization.

The core of this article is organized into five methodological sections to cover the theoretical foundation and implementation ins-and-outs of density-tempered SMC. Our discussions will be accompanied by examples on (1) a bi-modal non-statistic function without and with constraints, (2) a multidimensional step function, (3) offline and online optimization, (4) combinatorial variable selection, and (5) non-invertibility of the Hessian.

2 Optimization by SMC sampling

SMC sampling technique was originally proposed to solve fixed-parameter filtering problems which relies on Bayes' theorem to sequentially update the system. Later, it was adopted for Bayesian inference where a set of simulated particles are used to represent a posterior distribution of parameters, $\pi(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) \propto \mathcal{L}(\boldsymbol{\theta};\boldsymbol{\mathcal{D}})\pi_0(\boldsymbol{\theta})$ with $\mathcal{L}(\boldsymbol{\theta};\boldsymbol{\mathcal{D}})$ being the likelihood function of data $\boldsymbol{\mathcal{D}}$ and $\pi_0(\boldsymbol{\theta})$ being the prior. The typical Bayesian analysis uses the posterior mean, but $\arg\max_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$ yields a point estimate that solves the optimization problem defined under the Bayesian framework. Note that if an improper (uniform) prior is chosen, the posterior coincides with the target when taking a frequentist point of view, i.e., $\arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta};\boldsymbol{\mathcal{D}})$.

In a more general and non-statistical setting, the objective function $h(\boldsymbol{x})$ need not be a probability related function. Neither is data involved in defining the function. Moreover, the functional value of the argument, $\boldsymbol{x}$, over its domain may not be positive. But a simple monotonic transformation $f(\boldsymbol{x}) \propto \exp[h(\boldsymbol{x})]$ can turn the objective into a positive function so that $f(\boldsymbol{x})$ can be viewed as a density or probability function, depending on whether it is differentiable, up to an unknown norming constant. If there is a sampling technique that does not require knowing this unknown norming constant, then Monte Carlo simulation offers a way of solving this optimization problem. SMC is indeed such a sampling technique.

The SMC solution to the maximization problem is the point that gives rise to the largest functional value in the sample, which corresponds to the maximum order statistic of the sample of functional values. Convergence to the right value by the maximum order statistic is guaranteed per the usual argument, which in turn implies the convergence of the SMC solution to the global maximizer if it is uniquely identified.

Most optimization problems in real-world applications do not have analytical solutions, and sampling-based methods come in handy for such scenarios. Sampling methods have several advantages over traditional optimization methods. This group of algorithms is generic and in principle applicable to all optimization problems. Monte Carlo simulation is also a good technique for high-dimensional problems because the convergence rate is typically invariant to the dimension. It is also derivative-free and serves as a generic technique for finding the global optimum.

In this section, we first briefly introduce the theoretical concept and the history of using SMC for optimization. Then we introduce two important algorithms - density-tempered SMC and

expanding-data SMC and explain their complementarity for offline and online optimizations. Moreover, we demonstrate the generality of density-tempered SMC sampling technique by discussing its applications to constrained, discrete, combinatorial and state-space model optimizations. In a different section, we will compare it with the Markov Chain Monte Carlo (MCMC) method.

2.1 Optimization via sampling

Two most important families of Monte Carlo based algorithms are Markov Chain Monte Carlo (MCMC) and importance sampling. SMC originates from the family of importance sampling and was due to Gordon et al. (1993), where the authors developed it for fixed-parameter particle filtering. Chopin (2002) was the first to introduce SMC into Bayesian parameter estimation.

For statistical analysis, both MCMC and SMC algorithms were originally developed for Bayesian inference. To our knowledge, Lele et al. (2007) was the first to bring MCMC into a frequentist inference setting. Specifically, the authors proposed to calculate the maximum likelihood estimate (MLE) and conduct its associated statistical inference for complex ecological hierarchical models through the use of the MCMC algorithm. Through data cloning to be elaborated later, the prior distribution $\pi_0(\boldsymbol{\theta})$ regardless of its specification will eventually be dampened to the point where the Bayesian posterior mean becomes the MLE.

There are more recent works that directly utilize SMC as an optimizer for either ML estimation or others, and demonstrate its applicability and superiority. Duan (2019), Duan et al. (2020) and Duan and Li (2021) are examples of showing that prior distribution is completely unnecessary and maximizing the target function can be achieved through using an initialization sampler and importance weight. In this article, we focus on using SMC, particularly density-tempered SMC, for general optimization problems regardless of whether optimization is for a statistical analysis or not. In short, we will systematically demonstrate the wide applicability and power of optimization using density-tempered SMC.

2.2 Density-tempered SMC for optimization

2.2.1 Importance sampling and resampling

Importance sampling is commonly used to estimate moments of a function of random variables governed by a difficult-to-sample distribution. In the current context, we deploy it to estimate instead the mode of $f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$, which as discussed earlier is a distribution short of a norming constant. The method draws a random sample from a simple distribution $g(\boldsymbol{\theta})$ whose support covers that of $f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$. The importance weights, i.e., $w_i = f(\boldsymbol{\theta}_i|\boldsymbol{\mathcal{D}})/g(\boldsymbol{\theta}_i)$ for $i = 1, 2, \cdots, N$, after self-normalization become probabilities for a weighted sample, $\{\boldsymbol{\theta}_i, w_i/\sum_j w_j\}_{i=1}^N$. The sample thus represents an empirical distribution of $f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$.

One key feature of importance sampling is that self-normalization removes the need to know the norming constant. So, $f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$ is just as good as a properly specified density or probability function. Therefore, a target function for optimization, $f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$, only needs to be non-negative so that it is proportional to a density or probability function.

When there is a sequence of target distributions, $\{f_{\delta_p}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}), p = 0, 1, 2, \cdots\}$, with an increasing δ_p and leading to $f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$, one can envision its corresponding weighted sample at stage p as $\{\boldsymbol{\theta}_i^{(p)}, w_i^{(p)}\}_{i=1}^N$. The task is to design a sequential way to gradually move the system from $f_{\delta_p}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$ to $f_{\delta_{p+1}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$ and eventually reach $f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$.

2.2.2 Density-tempering

The idea of density-tempering came from Del Moral et al. (2006), and then followed by Duan and Fulop (2013), Duan and Fulop (2015) among others. In a general setting, we consider an initial particle cloud, $\{\boldsymbol{\theta}_i, i = 1, 2, \cdots, N\}$, generated from some easy-to-sample density $I(\boldsymbol{\theta})$ which need not be the prior distribution even for Bayesian analysis (see Duan et al., 2020). Moving to the target distribution $f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$ in one step is way too ambitious when $f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$ is very different from $I(\boldsymbol{\theta})$. Instead, an inhomogeneous sequence of intermediate target distributions, $\{f_{\delta_p}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}), p = 0, 1, 2, \cdots\}$, has been devised to allow for controlled moves from $I(\boldsymbol{\theta})$ to $f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$. It goes as follows:

$$f_{\delta_p}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) \propto f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})^{\delta_p} I(\boldsymbol{\theta})^{1-\delta_p}, \tag{1}$$

where $0 = \delta_0 < \delta_1 < \delta_2 < \cdots \leq 1$. This construction defines an appropriate bridge such that $f_0(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) = I(\boldsymbol{\theta})$ and $f_1(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) = f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$. The sequence of $\delta_0 < \delta_1 < \delta_2 \cdots$ in [0,1] can be self-adaptively taken from, say, a set of 1,000 equally spaced out points on a logarithmic scale translated from $[e^{-20}, 1]$.

Denote $\{\boldsymbol{\theta}_i^{(p)}, i = 1, \cdots, N\}$ as the particle cloud that approximates $f_{\delta_p}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$, the next δ_{p+1} can be selected via a grid search such that the reweighted particles can maintain a pre-specified effective sample size (ESS) (e.g., 50% of the sample size), which is defined as

$$ESS = \frac{\left(\sum_{i=1}^N w_i\right)^2}{\sum_{i=1}^N w_i^2} \tag{2}$$

The sequential samples must be reweighted to reflect incrementally added importance weights; that is,

$$w_i^{(p+1)} = w_i^{(p)} \frac{f_{\delta_{p+1}}(\boldsymbol{\theta}_i^{(p)}|\boldsymbol{\mathcal{D}})}{f_{\delta_p}(\boldsymbol{\theta}_i^{(p)}|\boldsymbol{\mathcal{D}})}. \tag{3}$$

However, sequential importance sampling will gradually concentrate the weights on a small subset of particles, which is commonly known as the particle degeneracy problem.

To reduce serious weight imbalance across particles, Gordon et al. (1993) introduced multinomial resampling to accompany the reweighting step. Resampling basically restores equal weights to the sample and replaces any particle with a heavy weight by multiple copies of the same particle. For example, a particle carrying a 20% probability will be replaced with the same in 20% of the sample so that it is likely to remain its importance in the subsequent sample. Other resampling approaches include residual resampling (Whitley, 1994, Liu and Chen, 1998), stratified resampling (Kitagawa, 1996) and systematic resampling (Carpenter et al., 1999). The principle governing the resampling step is that the new system should be a good approximation to the original system. Douc et al. (2005) provided a comparison on several commonly used resampling schemes.

However, resampling does not fundamentally resolve the particle degeneracy problem because the number of distinct particles has been decreased in exchange for balanced weights. A rejuvenation step to restore particle diversity is a must and will be discussed next.

2.2.3 Support boosting

In the particle filtering literature, Gilks and Berzuini (2001) first proposed adding a move step following the resampling step to rejuvenate the particle set (i.e., boosting the empirical support), for which new particles are proposed via a Markov chain transition kernel conditional on the resampled particles. Running through a properly designed Markov kernel boosts the sample variety without changing the underlying target distribution because the input sample has been drawn from the same stationary distribution. Chopin (2002) extended the algorithm to Bayesian inference applications on static models for which one estimates model parameters through a posterior distribution.

The Choice of the Markov kernel largely determines the efficiency of an algorithm. The Metropolis-Hasting kernel (Metropolis et al., 1953 and Hastings, 1970) is most commonly used, which works as in Algorithm 1.

Algorithm 1 The Metropolis–Hastings Algorithm

Given the system's current particles $\boldsymbol{\theta}$ and at the tempering value of δ_p,

Step 1. Propose $\boldsymbol{\theta}^* \sim Q(\cdot|\boldsymbol{\theta})$, where Q is a proposal sampler's distribution. It together with $f_{\delta_p}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) \propto f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})^{\delta_p} I(\boldsymbol{\theta})^{1-\delta_p}$ satisfies the reversibility condition: $f_{\delta_p}(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{D}})Q(\boldsymbol{\theta}|\boldsymbol{\theta}^*) = f_{\delta_p}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})Q(\boldsymbol{\theta}^*|\boldsymbol{\theta})$.

Step 2. Compute the acceptance rate α,

$$\alpha = \min\left(1, \frac{f_{\delta_p}(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{D}})Q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{f_{\delta_p}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})Q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}\right) \tag{4}$$

Step 3. With probability α, accept $\boldsymbol{\theta}^*$, otherwise keep the old particle.

Step 4. Repeat step 1-3 until some criteria are met (e.g., reaching a threshold level of cumulative acceptance rate).

For the proposal sampler's density Q, a usual choice is a Gaussian distribution centered at the current location $\boldsymbol{\theta}_i$, i.e., a random walk, so that a small region near $\boldsymbol{\theta}_i$ is visited next. A random walk proposal can result in a high acceptance rate when small variance is chosen. It can easily replace identical particles but the new particles are in fact of very similar values and the sample is only artificially boosted. Chopin (2002) suggested an independent proposal sampler by adopting a Gaussian distribution with or without cross correlations where the Gaussian parameters can be estimated with the equally weighted current particles, i.e., $\{\boldsymbol{\theta}_i^{(p)}, i = 1, \cdots, N\}$. A mixture of the independent and random walk proposal samplers is actually a good choice for many applications to explore potential improvements both globally and locally. Particle replacement can also be targeted at a random sub-vector of $\boldsymbol{\theta}$ to increase the acceptance rate when the dimension of $\boldsymbol{\theta}$ is high.

The key steps for density-tempered SMC are summarized in Algorithm 2.

2.2.4 A non-statistical example

To illustrate the use of density-tempered SMC algorithm, we first consider the following non-convex optimization for an objection function that does not involve any data:

$$\max_{\boldsymbol{x}\in\mathbb{R}^2} f(\boldsymbol{x}) \equiv \phi\left(\boldsymbol{x}; \begin{bmatrix}-1\\-2\end{bmatrix}, \begin{bmatrix}4 & 0.6\\0.6 & 1\end{bmatrix}\right) + \phi\left(\boldsymbol{x}; \begin{bmatrix}2.5\\2\end{bmatrix}, \begin{bmatrix}2.25 & -0.45\\-0.45 & 2.25\end{bmatrix}\right) \tag{5}$$

Algorithm 2 The Density-Tempered SMC Algorithm

Step 0. *Initialization*: generate a particle set $\boldsymbol{\theta}^{(0)}$ from an initialization distribution $I(\boldsymbol{\theta})$. The initial weight $w_i^{(0)} = 1/N$ is associated with the initial tempering factor $\delta_0 = 0$.

Step 1. *Reweighting and determining* δ_p: set $p = 1$ and search for next δ_p over a predefined grid over $[0, 1]$ such that its corresponding ESS, computed for the reweighted particles with the incremental importance weights $w_i^{(p)} = w_i^{(p-1)} \left[\frac{f(\boldsymbol{\theta_i}|\boldsymbol{\mathcal{D}})}{I(\boldsymbol{\theta_i})}\right]^{\delta_p - \delta_{p-1}}$, is greater than $50\% \times N$. Denote the corresponding reweighted particle set as $\{\boldsymbol{\theta}_i^{(p)}, w_i^{(p)}\}_{i=1}^N$.

Step 2. *Resampling*: randomly draw N particles according to $\boldsymbol{w}^{(p)}$ to produce an equally weighted particle set. Denote the resampled particles as $\{\boldsymbol{\theta}_i^{(p),r}, 1/N\}_{i=1}^N$.

Step 3. *Support-boosting move*: propose N independent particles $\boldsymbol{\theta}^*$ and deploy the Metropolis-Hasting kernel as described in Algorithm 1 to replace $\boldsymbol{\theta}^{(p),r}$. The Markov chain transition kernel targets the density-tempered intermediate function $f_{\delta_p}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) \propto f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})^{\delta_p} I(\boldsymbol{\theta})^{1-\delta_p}$. Denote the rejuvenated particle set as $\{\boldsymbol{\theta}_i^{(p),*}, 1/N\}_{i=1}^N$.

Step 4. *Loop*: if $\delta_p < 1$, set $p = p + 1$ and return to Step 1.

Step 5. The SMC optimal solution is the particle corresponding to the maximum likelihood value.

where

$$\phi\left(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \frac{1}{|\boldsymbol{\Sigma}|} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}.$$

We can visualize the target function in Figure 1. The function has two local modes, with the global maximum at $\boldsymbol{x} = [-0.9972, -1.9990]'$. This target function can of course be viewed as a density function missing its norming constant. Typical gradient-based optimizers are known to be prone to trapping in local maxima. Therefore, a common practice is to initialize the optimizer at multiple locations. However, there is no guarantee nor practical robustness to ensure that the global solution can be found this way. For the above example of two local maxima and due to its simplicity, using multiple initialization points will almost certainly be able to locate the global maximum. We use the example mainly to illustrate the use of SMC optimization and a host of issues concerning optimization.

Figure 2 presents the results obtained by a standard density-tempered SMC algorithm using a particle set of 1,000.[1] Figure 2a is a scatter plot of the final particles from one SMC run. The sample shows concentration around the two modes, forming an empirical distribution of the target function. Figure 2b displays the distribution of the optimal solutions (the point which has highest functional value) from 500 independent SMC runs (using different random seeds). The plot shows that SMC is indeed a global optimizer because all the 500 solutions scatter around the global maximum marked by a cross.

[1] We initialize the sample for (x_1, x_2) from two independent single-variable samplers with each being $\mathcal{N}(0, 5^2)$. For move steps, we adopt a proposal sampler that is a mixture distribution combining with equal probabilities of two components: (1) an independent normal distribution using the means and the standard deviations derived from the existing SMC particles, and (2) a random walk proposal based on a scaled-down standard deviations used in the independent proposal. For each tempering stage, move steps are repeated until the cumulative acceptance rate exceeds 200%.

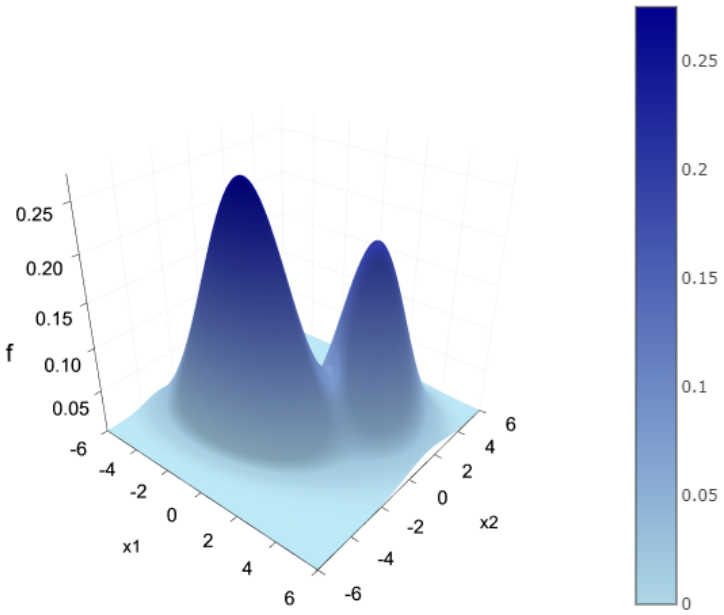
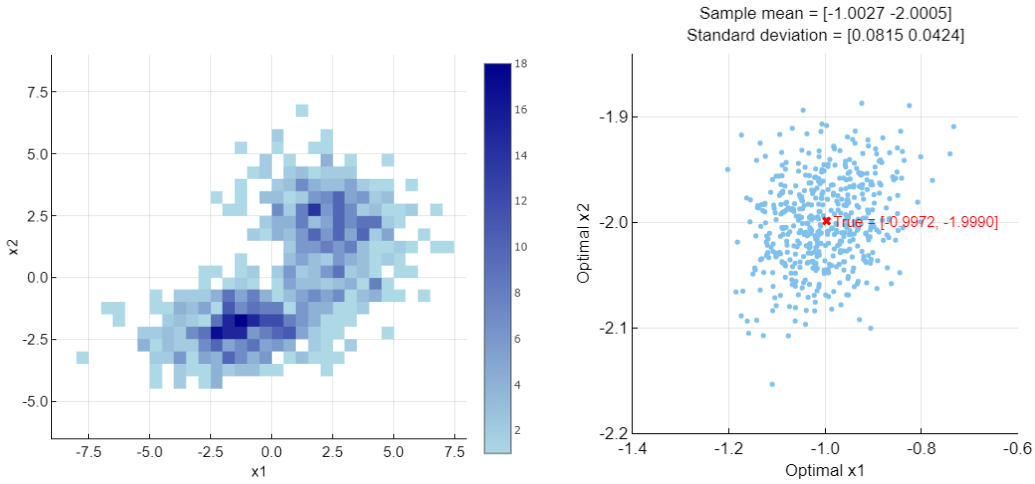


Figure 1: The bi-modal target function in equation (5)

There are easy ways to achieve a higher precision for the SMC solution, and we will take up the issue later in Section 3.



(a) Empirical distribution in one 1000-particle SMC run

(b) Optimal solutions from 500 independent 1000-particle SMC runs

Figure 2: SMC optimization for the bi-modal target function in Equation (5)

2.3 Expanding-data SMC

Let $\pi(\boldsymbol{\theta}|\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T)$ be a posterior distribution for the data $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\}$ independently drawn from some static model with parameter $\boldsymbol{\theta}$. Chopin (2002) proposed an iterated batch importance sampling algorithm for estimating the parameter. Common simulation procedures consist of numerous iterations, and within each the full sample will be evaluated. This can be very inefficient for large datasets. Therefore, Chopin (2002) proposed to first perform a "preliminary" exploration over part of the data, then sequentially add the rest of the data to the sample.

To start, one can target $\pi(\boldsymbol{\theta}|\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t, t < T)$ to obtain a preliminary estimation using the first t observations. Then, update the estimate by adding p observations, i.e., the target distribution becomes $\pi(\boldsymbol{\theta}|\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{t+p}, t+p < T)$. The observations are sequentially added until the whole sample has been exhausted to provide the Bayesian inference based on the full dataset. Fulop and Li (2013) is an example of application where they use a particle filter to evaluate a complex likelihood function in order to perform the Bayesian analysis.

Naturally, the expanding-data algorithm, a term coined in Duan and Fulop (2015), is restricted to statistical optimization where data are involved. To go beyond its Bayesian inference origin, we need to do away with the prior belief; that is, the target function is $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_1, \cdots, \boldsymbol{x}_T)$ instead of $\pi(\boldsymbol{\theta}|\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T)$. In addition, an initialization sampler as in 2.2.2 is needed to replace the prior distribution to start the recursive system for the initial batch of data. Correspondingly, the initial importance weight becomes

$$w^{(0)} = \frac{\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_1, \cdots, \boldsymbol{x}_t)}{I(\boldsymbol{\theta})} \tag{6}$$

Resampling and support-boosting moves give rise to an equally-weighted sample representing $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_1, \cdots, \boldsymbol{x}_t)$. The system is then ready for advancing to next data batch with the incremental weight:

$$w^{(1)} = \frac{\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_1, \cdots, \boldsymbol{x}_{t+p})}{\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_1, \cdots, \boldsymbol{x}_t)} = \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_{t+1}, \cdots, \boldsymbol{x}_{t+p}|\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t). \tag{7}$$

Note that $\{\boldsymbol{x}_t, t = 1, 2, \cdots, T\}$ need not be an *i.i.d.* sequence. If, for example, $\boldsymbol{x}_i$ is autoregressive of order one, $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_{t+1}, \cdots, \boldsymbol{x}_{t+p}|\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t)$ can be simplified to $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_{t+1}, \cdots, \boldsymbol{x}_{t+p}|\boldsymbol{x}_t)$. If the data are realizations of a state-space model, equation (7) is valid but cannot be further simplified. The same logic recursively applies to subsequent adding of data batches. The description for the expanding-data algorithm is in Algorithm 3.

The above incremental-weight formula indeed suggests a less costly evaluation of the incremental likelihood function because it is limited to the new data batch. It is deceptive, however, because the Metropolis-Hastings support-boosting step described earlier still needs to evaluate the likelihood function of the entire data sample.

As an optimization method for a static dataset, Duan and Fulop (2015) showed that density-tempered SMC dominates expanding-data SMC, particularly when the data contains outliers. This is intuitively understandable because expanding-data SMC may cause the parameter particle cloud to temporarily stray away from the ultimate target, i.e., the whole-sample MLE.

However, the expanding-data SMC algorithm is a natural choice for live updating systems, where new data arrive as time goes by. Interestingly, combining the two SMC algorithms forms a complementary solution for online optimization. When adding a data batch via equation (7)

Algorithm 3 The Expanding-Data SMC Algorithm

Step 0. *Initialization*: generate an particle set of size N, $\boldsymbol{\theta}^{(0)}$, from an initialization distribution $I(\boldsymbol{\theta})$. Denote b the batch number and set $b = 1$.

Step 1. *Reweighting*: if $b = 1$, update weights according to equation (6), otherwise update weights by equation (7). Denote the reweighted particle cloud as $\{\boldsymbol{\theta}_i^{(b)}, w_i^{(b)}\}_{i=1}^N$.

Step 2. *Resampling*: randomly draw N particles according to $\boldsymbol{w}^{(b)}$, resulting in an equally weighted particle set. Denote the resampled particles as $\{\boldsymbol{\theta}_i^{(b),r}, 1/N\}_{i=1}^N$.

Step 3. *Support-boosting move*: propose N independent particles $\boldsymbol{\theta}^*$ and deploy the Metropolis-Hasting kernel as described in Algorithm 1 to replace $\boldsymbol{\theta}^{(b),r}$. The Markov chain transition kernel targets the intermediate posterior distribution $\pi(\boldsymbol{\theta}|\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{t+(b-1)p})$. Denote the rejuvenated particles by $\{\boldsymbol{\theta}_i^{(b),*}, 1/N\}_{i=1}^N$.

Step 4. The updated optimal parameter is the particle corresponding to the maximum likelihood value of data up to current batch.

Step 5. *Loop*: if $b < B$, where B is the total number of batches, set $b = b + 1$ and return to Step 1.

might be too ambitious due to its significant impact, one can easily introduce tempering steps. An application and discussion of combining the two algorithms is available in Duan and Fulop (2013), where the authors explained how the two algorithms can be applied for operations that involve real-time updates. Indeed, the combined approach has already been implemented for a live corporate default prediction system maintained under the Credit Research Initiative at the National University of Singapore.[2]

We now show an example of fitting a logistic regression to the credit card default dataset of Yeh and Lien (2009) by using the data-expanding SMC algorithm. Mathematically, the model is defined for the two-class (default and non-default) dataset with many predictive features and n data instances, denoted by $\{\boldsymbol{x}_i; i = 1, 2, \cdots, n\}$, as follow:

$$P_{\text{default}} = \frac{1}{1 + \exp\left(-\beta_0 - \boldsymbol{x}'\boldsymbol{\beta}\right)} \tag{8}$$

$$P_{\text{non-default}} = 1 - P_{\text{default}} \tag{9}$$

The sample has in total 30,000 data instances (including 6,636 defaults and 23,364 non-default cases) and 23 predictors.[3] We split the dataset into 1,000 batches with 30 data instances per batch. To illustrate, we only display the estimation results on two parameters in Figure 3. Figure 3a and 3b shows the two evolution paths of parameter values as data batches being added. The red dashed horizontal line identifies the corresponding optimal parameter value. The grey confidence bands are calculated using 1.96 times the standard deviation derived from the particle set at each data-expansion point. Note that the variation of parameter values can be large. The magnitude can be much more severe if there is a dynamic structure in data (e.g., the business/credit cycle effect on financial time series).

[2] Please refer to NUS-CRI (2021) for technical details. General information and data are available at https://nuscri.org/en/.

[3] Readers may refer to Yeh and Lien (2009) for the features used for default prediction.

Figure 3c records the computing time versus the batch number. A longer computing time is generally required to complete a single data batch as batches being gradually added, because the Metropolis-Hastings support-boosting step must be operated on the likelihood of the cumulative data sample whose size is increasing. However, the computing time is not always monotonically increasing because each update may require a different number of Metropolis-Hastings move steps to reach the targeted cumulative acceptance rate of 200%.

2.4 Constrained optimization

Constrained optimization problems are common in real-world applications. Variable constraints can vary widely from simple bounds to systems of equalities/inequalities. It may become difficult for gradient-based optimization methods to handle very complex and/or non-convex constraints. SMC sampling techniques can, however, deal with all kinds of constraints in a rather straightforward way. For simple lower and/or upper bounds, one can deploy truncated sampling distributions to generate particles. More generically, it is straightforward to introduce an indicator function, $\chi(\boldsymbol{x} \in \mathcal{C})$ which equals 1 when the constraints are met and 0 otherwise, into the objective function to become a modified target function $f(\boldsymbol{x})\chi(\boldsymbol{x} \in \mathcal{C})$ in order to handle the constraints through rejection. Incorporating complex constraints through modifying the objective function has, for example, adopted by Duan and Li (2021) and in the National University of Singapore Credit Research Initiative's monthly calibrations of its corporate default prediction system (see NUS-CRI, 2021).

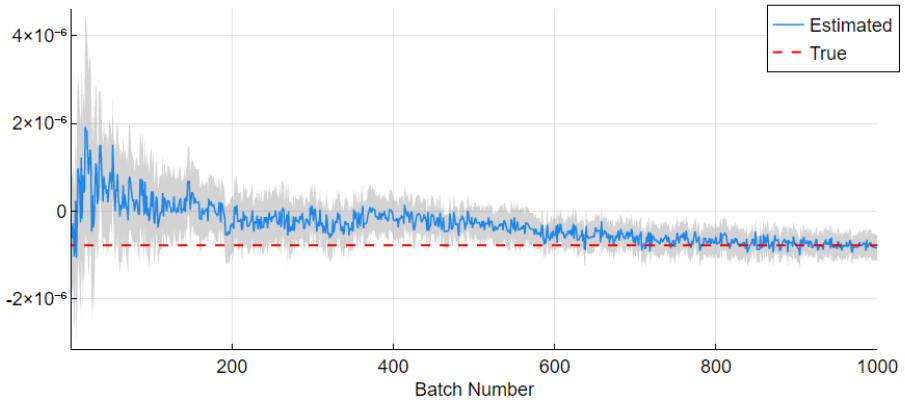
From an algorithmic point of view, the SMC algorithm only needs an additional simple condition check when new particles are proposed (in both the initialization and support-boosting move stages); that is to set the target function's value to 0 ($-\infty$ if computing a log-objective function) for particles that do not satisfy the constraints. The constraints may also call for a revision to the initialization sampler if density-tempering cannot generate an initial set of particles that meets the ESS threshold.[4] Finding a revised initialization distribution is rather straightforward. One can, for example, continue to sample until finding a suitable number of particles with strictly positive target functional values. Then, proceed to use this subset to come up with the new location and scale parameters for the revised initialization sampler.

Below we show that the SMC sampling technique can easily optimize the same function as in section 2.2.4 under a non-convex constraint that hollows out the two-dimensional Euclidean space with a square:

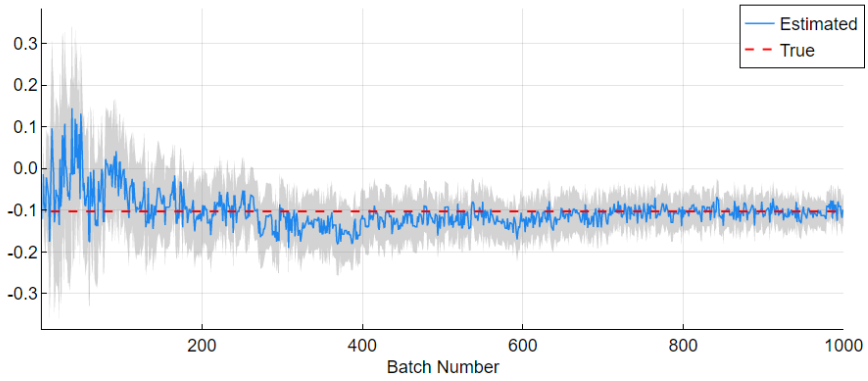
$$\mathcal{C} = \boldsymbol{R}^2 \setminus \{x_1 \in (-3, 0) \text{ and } \ x_2 \in (-3, 0)\} \tag{10}$$

The original global optimal solution violates the constraint, and the new optimal point becomes $\boldsymbol{x}_{opt} = [0.0, -1.8451]'$. We apply a standard density-tempered SMC algorithm to target $f(x)\chi(x \in \mathcal{C})$ with 1,000 particles. Figure 4a shows the empirical distribution formed by the final particle set from one SMC run, and with which one can locate the Monte Carlo estimate for the optimal solution. Figure 4b then displays the 500 optimal solutions found from 500 independent 1000-particle SMC runs. Note that the SMC solution's precision can be readily improved via data cloning or k-fold duplication to be described in Section 3.

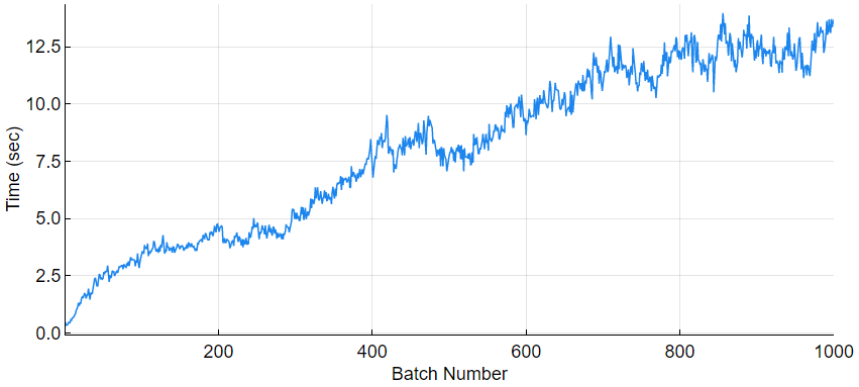
[4] This situation will occur when the set has less than, say, 500 particles with strictly positive target functional values when the ESS threshold is set to 500.



(a) Parameter for attribute "Credit Card Limit"

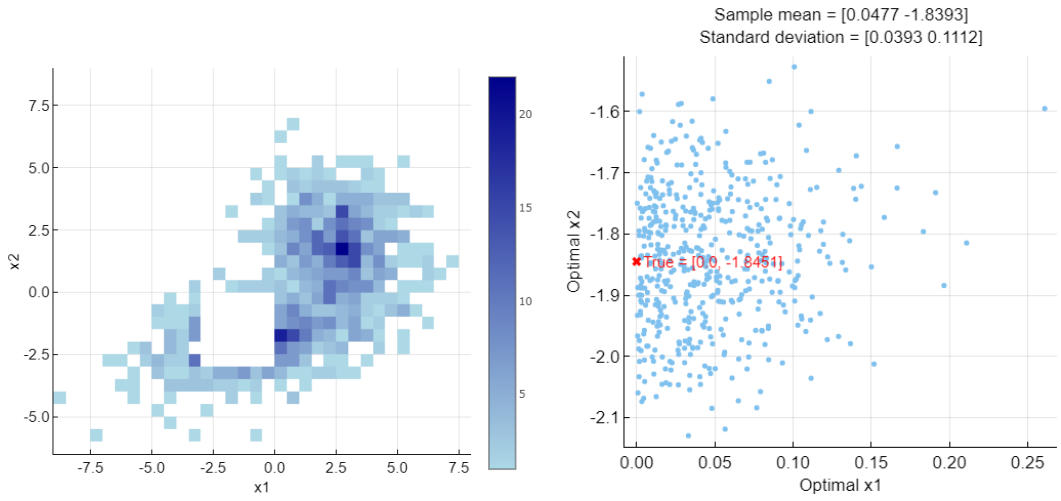


(b) Parameter for attribute "Education Level"



(c) Single-batch computing time

Figure 3: Evolution of two parameter values and single-batch computing time in an expanding-data 1000-particle SMC run



(a) Empirical distribution in one 1000-particle SMC run

(b) Optimal solutions from 500 independent 1000-particle SMC runs

Figure 4: SMC constrained optimization for the bi-modal target function in equation (5) subject to the constraint in (10)

2.5 Optimization for discontinuous functions

A multidimensional step function defined over $\boldsymbol{R}^k$ is an example of discontinuous function. If an optimum exists, the solution will be a non-singleton set of k-dimensional points sharing the same optimal functional value. Naturally, gradient-based optimization methods are ill-suited for the task, but SMC can readily handle such a function as demonstrated in Duan and Li (2021). Their optimization task is to find the cutoff boundary values and through which to map probabilities of default (PDs) into implied credit ratings by referencing the credit migration history of a credit rating agency such as S&P or Moody's. This PD-implied rating methodology improves upon an existing mapping method solely based on matching to historical average default rates of different rating categories.

Two datasets are relevant to Duan and Li (2021)'s model. First, take, say, the S&P reported annual credit migration matrices tallied over 18 years for its global rating pool in which rated firms are consolidated into nine categories without modifiers (i.e., AAA, AA, A, BBB, BB, B, CCC, CC and C). The second is the NUS-CRI database of PDs offering over 80,000 exchange-traded firms globally that we have referred to previously in Footnote 2. These PD data can be used to deduce model-generated credit migration matrices over the same time period. Naturally, the cutoff values will determine the behavior of the model-generated credit migration and a small perturbation may cause some implied credit migration matrices to jump discretely.

Duan and Li (2021)'s model defines eight cutoff values and links them to the buffer zones used to mimic the strong rating stickiness observed in any credit rating agency's data. The model also utilizes the same cutoff values to further define rating modifiers such as AA^+, AA^-, BBB^+, etc. They deployed density-tempered SMC to find the eight optimal cutoff values that best match the model-generated matrices with the corresponding observed rating migration matrices. Since the objective is a multidimensional step function, the solution is not unique. Due to a large

number of firms in the NUS-CRI database, this non-singleton solution set has a pretty small Lebesgue measure, and thus non-uniqueness makes no material difference for practical purposes.

2.6 Combinatorial optimization

Combinatorial optimization problems are mostly NP-hard, which make them difficult to solve. Generally speaking, discrete optimization algorithms can be divided into two categories: the exact and meta-heuristic methods. Exact algorithms, such as the branch-and-bound algorithm (Land and Doig, 1960) and dynamic programming methods, are guaranteed to find an optimal solution whose optimality is provable. However, the run-time often increases dramatically with dimension of the problem, therefore their capability in solving high-dimensional problems is limited. In contrast, the family of meta-heuristic algorithms trades optimality for run-time. Some popular algorithms include simulated annealing (Kirkpatrick et al., 1983), genetic algorithms (Holland, 1992), tabu search (Glover and Laguna, 1999) and evolutionary algorithms.

The SMC approach to combinatorial optimization invented by Duan (2019) is a provable approach relying on Monte Carlo convergence. It is not meta-heuristic because convergence to the right solution is ensured as the sample size gets large. It is also a practical approach because solutions can be found within a reasonable amount of computing time and the Monte Carlo error can also be assessed. The methodological essence of SMC optimization for combinatorial problems is to view the optimization target as a discrete probability function and proceed to sample from it. The point yielding the highest probability is the SMC solution. With a sufficiently large particle set, the method will in principle obtain the global maximum. The approach can be applied to many combinatorial optimization problems with minor tweaking to the sampler.

Duan (2019)'s design specifically aims at selecting a subset from a very large set of potential variables in linear regressions where a zero-norm penalty is used instead of L_1-norm as in the popular Lasso of Tibshirani (1996). Duan (2019) formulated the target function for selecting s variables out of P potential feature variables under zero-norm as follows:

$$\underset{\boldsymbol{U}\in\boldsymbol{P}(s)}{\arg\max}\ \exp\left\{-||\boldsymbol{y}-\boldsymbol{X}_{\boldsymbol{U}}\hat{\boldsymbol{\beta}}(\boldsymbol{U})||_{l_2}^2\right\} \tag{11}$$

where $\boldsymbol{y}$ is the n-dimensional column vector for the response variable, $\boldsymbol{X}$ is the $n\times P$ matrix of features, $\boldsymbol{P}(s)$ denotes $\{\boldsymbol{U}\in\boldsymbol{P}^{\times s}$ and $U_1\neq U_2\neq\cdots\neq U_s\}$, $\boldsymbol{P}^{\times s}$ stands for the s-Cartesian product of the set of P variables, $\boldsymbol{X}_{\boldsymbol{U}}$ denotes the sub-matrix of $\boldsymbol{X}$ whose columns correspond to the features' sequence numbers in $\boldsymbol{U}$, and $\hat{\boldsymbol{\beta}}(\boldsymbol{U})=(\boldsymbol{X}_{\boldsymbol{U}}'\boldsymbol{X}_{\boldsymbol{U}})^{-1}\boldsymbol{X}_{\boldsymbol{U}}'\boldsymbol{y}$ is the optimal regression $\boldsymbol{\beta}$ when $\boldsymbol{U}$ is known.[5]

The target function in (11) is totally discrete but can be viewed as a probability function short of a norming constant. This target function is permutation-invariant if they form the same combination. Duan (2019) proposed to use the single-variable regression R^2 to set the individual variable's initialization probability. Sampling takes s-variable permutations because this probability is easily computable whereas the combination probability is harder to evaluate. The proposal probabilities for the Metropolis-Hastings move during the density-tempered SMC run are updated by counting the individual variable's occurrences in the SMC sample.

[5] It is our understanding that Duan (2019) has introduced a self-adaptive tuning parameter λ in his algorithmic implementation to in effect target $\exp\left\{-\lambda||\boldsymbol{y}-\boldsymbol{X}_{\boldsymbol{U}}\hat{\boldsymbol{\beta}}(\boldsymbol{U})||_{l_2}^2\right\}$. This tuning parameter does not affect the theoretical solution but can shape the target function to better achieve a separation of the optimal solution from others.

Satpathy and Shah (2022) has, for example, deployed this combinatorial optimization technique to solve the sparse stock index tracking problem that is widely applicable to the management of passive funds. Here, we use the Ames housing dataset compiled by De Cock (2011) to demonstrate Duan (2019)'s algorithm.[6] The dataset contains 79 features and a response variable, which is the house sale price.[7] Table 1 presents the selection results. To benchmark, we also present the selection results using the Lasso regression of Tibshirani (1996) followed by a post-selection OLS regression. We apply a 3-fold cross validation in both cases to avoid over-fitting. Using a higher number of folds can lead to a more stable model, but will take longer time to complete the task.

	1^{st}-order terms Total # of features = 80		1^{st} + 2^{nd}-order terms Total # of features = 80 + 3,101[8]	
	Zero-norm SMC	Lasso + OLS	Zero-norm SMC	Lasso + OLS
# of features selected	15	37	16	38
R-Squared	0.8430	0.8576	0.9252	0.9067
Computing time: single run	8s	0.007s	6m40s	0.22s
Computing time: 3-fold CV	4m20s	0.04s	2h34m	3s

Table 1: Comparison of feature selections via zero-norm SMC and Lasso

The left panel of Table 1 shows results for selection out of 80 potential variables (79 features plus the intercept). It is evident that Lasso has selected substantially more features (37 vs 15) with a marginal improvement in R^2 (85.76% vs 84.3%). Lasso is obviously a much more efficient algorithm using a tiny fraction of the computing time needed for the zero-norm SMC selection algorithm.

Quite common in regression analysis is to consider interaction terms created from the original feature variables. In the right panel of Table 1, the selection results are for choosing a subset of features from 3,181 potential variables that include all second-order non-redundant terms[8]. The zero-norm SMC algorithm selects 16 variables, and among them, 15 are the 2^{nd}-order terms, leading to a substantial improvement in R^2 as compared to only using the 1^{st}-order terms (92.52% vs 84.3%). For example, the interaction term, *overall quality rates* $\times$ *year built*[9], has a highly significant, positive coefficient, implying that higher prices are mainly for better-quality and newer houses. Again, Lasso has grossly over-selected variables (38 vs 16). But this time, more selected features actually lead to a counter-intuitively lower R^2, indicating a questionable selection performance.

Lasso's poor ability in handling multicollinearity is known in the literature (e.g., Zhao and Yu, 2006, Herawati et al., 2018). Duan (2019) provided a rather intuitive explanation for this shortcoming. Lasso regression minimizes an L_2 loss while slaps an L_1 penalty. Note that the L_2 loss is invariant to linear transformations but the L_1 penalty is not. In short, multicollinearity alters the penalty but leaves the loss intact, leading to an unpredictable behavior when facing multicollinearity. Duan (2019) also provided an extensive simulation study to document Lasso's

[6] This zero-norm variable selection algorithm has been implemented in DeepSelect®, a proprietary software accessible upon request at: https://variableselection.nuscri.org.

[7] A description of the 79 features is available at http://jse.amstat.org/v19n3/decock/DataDocumentation.txt.

[8] All redundant 2^{nd}-order terms are removed (e.g., the intercept multiplying by a feature). Furthermore, almost perfectly linearly dependent 2^{nd}-order terms are also removed.

[9] Quality rates run from 1 to 10 with 10 being the best. "year built" indicates the calendar year of construction so that a larger value means a newer house.

tendency to grossly over-select variables when different levels of correlation are present. Among the 3,181 features in this Ames housing dataset, there are 12,069 pairs with absolute correlations larger than 0.9, explaining why Lasso grossly over-selects features.

2.7 SMC^2 optimization

SMC is widely used to make inference for latent variable models either for the latent states (e.g. the problem of filtering and smoothing), the model parameter $\boldsymbol{\theta}$ (both from the frequentist and Bayesian point of views), or both (i.e., the joint estimation of the latent states and model parameters). There has been an extensive literature on this, e.g., Doucet, Johansen, et al. (2009), Duan and Fulop (2009), Kantas et al. (2009), Kantas et al. (2015), Xu (2018), Duan et al. (2020), to name just a few.

When considering latent variable models, the challenging problem is to conduct inference using the joint density of the parameters and the latent variables, i.e. the inference of $f(\boldsymbol{\theta}, \boldsymbol{U}|\boldsymbol{\mathcal{D}})$ where $\boldsymbol{U}$ denotes the latent variables. By targeting $f(\boldsymbol{\theta}, \boldsymbol{U}|\boldsymbol{\mathcal{D}})$, the parameter estimation problem is solved by marginalization. If Monte Carlo based methods are deployed, the inference of $\boldsymbol{\theta}$ targeting the marginal distribution $f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$ can be made by simply discarding the $\boldsymbol{U}$ component from the particle set $\{\boldsymbol{\theta}_i, \boldsymbol{U}_i\}_{i=1}^N$. To address the complicated and challenging joint density problem, the two-layer approach, commonly referred to as SMC^2, has been proposed in the literature. Various SMC^2 methods in Chopin et al. (2013), Fulop and Li (2013), Duan and Fulop (2015), Duan et al. (2020), Jasra et al. (2021), etc are available.

Common to all is that the inner layer runs a fixed-parameter SMC by marginalizing out the latent states while the outer layer runs SMC on the model parameters targeting the marginal distribution of $\boldsymbol{\theta}$, which is of a much lower dimension. The inner layer is closely related to the work pioneered by Gordon et al. (1993), where particle filtering could be used to provide an unbiased approximation of the observed data likelihood for a dynamic model. In short, it is a SMC way to marginalize the joint likelihood function by integrating out the latent variable(s). Typically, the marginal likelihood function is analytically intractable when the model is, for example, a non-linear or non-Gaussian state-space model.

We now use the density-tempered SMC^2 design of Duan and Fulop (2015) to explain. The density-tempered target in this algorithm is given below, which lies in the space augmented by some auxiliary random variables $\boldsymbol{U}$ due to the deployment of a particle filter.

$$f_\delta(\boldsymbol{\theta}, \boldsymbol{U}|\boldsymbol{\mathcal{D}}) \propto \left(\frac{\widehat{\mathcal{L}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}, \boldsymbol{U})}{I(\boldsymbol{\theta})}\right)^\delta \varphi(\boldsymbol{U}|\boldsymbol{\mathcal{D}}, \boldsymbol{\theta}) I(\boldsymbol{\theta}) \tag{12}$$

where $\widehat{\mathcal{L}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}, \boldsymbol{U})$ is the SMC estimate of $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}, \boldsymbol{U})$, the marginalized likelihood, by applying a particle filter, and $\varphi(\boldsymbol{U}|\boldsymbol{\mathcal{D}}, \boldsymbol{\theta})$ is the density function for the auxiliary random variables. Traversing through a tempering bridge indexed by $\boldsymbol{\delta}$ will advance the system from 0 to 1. Equation (12) suggests that from δ_1 to δ_2, the importance weight can be simplified to $\left(\widehat{\mathcal{L}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}, \boldsymbol{U})/I(\boldsymbol{\theta})\right)^{\delta_2 - \delta_1}$. After obtaining the final sample of SMC particles in the augmented space, one proceeds to marginalize the SMC sample by focusing on $\boldsymbol{\theta}$ only.

Two important observations are in order. First, the original design of Duan and Fulop (2015) is for Bayesian analysis where (1) the target function is the posterior distribution, i.e., $\widehat{\mathcal{L}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}, \boldsymbol{U})\pi_0(\boldsymbol{\theta})$ in the numerator, and (2) $I(\boldsymbol{\theta})$ is set to $\pi_0(\boldsymbol{\theta})$. Together, it produces a simplified but less computationally efficient density-tempered target: $f_\delta(\boldsymbol{\theta}, \boldsymbol{U}|\boldsymbol{\mathcal{D}}) \propto \left(\widehat{\mathcal{L}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}, \boldsymbol{U})\right)^\delta \varphi(\boldsymbol{U}|\boldsymbol{\mathcal{D}}, \boldsymbol{\theta})\pi_0(\boldsymbol{\theta})$.

Second, $\varphi(\boldsymbol{U}|\boldsymbol{\mathcal{D}},\boldsymbol{\theta})$ needs no evaluation because it is always cancelled in the incremental importance weight from, say, δ_1 to δ_2 and in the acceptance probability defining the Metropolis-Hastings move. We will apply the SMC^2 algorithm on a discrete-time latent stochastic volatility model later in Section 6.1.

The formulation in Equation (12) allows for computing the maximum likelihood estimate without having to worry about the non-smoothness of the particle filter with respect to the model parameters, which is the issue discussed extensively in Kantas et al. (2009) and Kantas et al. (2015). Pitt (2002) and Malik and Pitt (2011) showed that for one-dimensional latent stochastic process, the particle filter can be made continuous with respect to the parameters by using common random numbers to enable the use of a gradient-based optimizer, and Duan and Fulop (2009) is such an example. When the dimension is two or higher, practical solutions for obtaining a smooth likelihood function do not yet exist. Thus, Bayesian inference without the need for optimization seems to be an obvious choice.

Indirectly utilizing the simulated EM algorithm to conduct maximum likelihood analysis is an alternative. It recognizes a fact that the complete-data log-likelihood's gradient and Hessian are computable even though their incomplete-data log-likelihood are not smooth with respect to parameters. In short, particle filtering at the current parameter values can help with computing the expected values of those derivatives needed in the optimization step. For the EM approach, readers can find general discussions in Kantas et al. (2009), Kantas et al. (2015) and Xu and Jasra (2019), and stable variance estimation in Duan and Fulop (2011).

Now working with Equation (12), directly computing the maximum likelihood estimate when facing higher-dimensional latent stochastic processes actually becomes practical. Duan et al. (2020) further showed how to improve the precision of such maximum likelihood estimation by data cloning, which we will take up the discussion in Section 3.3.

3 Improve SMC sampling quality

Optimization problems in practice often involve complex nonlinear targets with possibly many local solutions. The classical inference problem for solving the MLE targets a likelihood function defined with the observed data, i.e., solving the $\boldsymbol{\theta}$ that maximizes $f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) = \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$. Except for simple illustrative cases, $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$ for typical real-world statistical models could be highly skewed, multi-modal or flat at the optimum. Still, the gradient ascent/descent algorithms are often adopted for the ML parameter estimation under such circumstances. In many cases, simulation-based optimization methods are obviously more appealing.

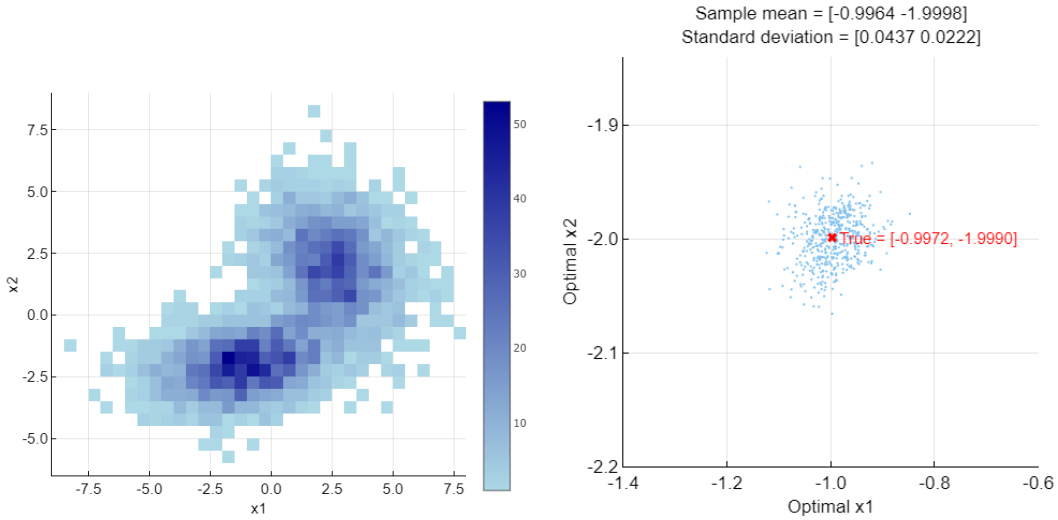
A simulation-based optimum may lack the precision needed unless the simulation sample size is large enough. It is vitally important to be able to increase sampling accuracy at low computational costs. In the next two subsections, we present two powerful and yet easily implementable approaches for increasing the quality of the SMC optimum. After that, we will turn to the discussion of applying these precision-enhancing methods in the SMC^2 setting.

3.1 k-fold duplication

The simple idea of k-fold duplication was proposed by Duan and Zhang (2015) to increase sampling efficiency. Instead of taking a bigger sample through the SMC steps, one can directly duplicate by many folds a base sample that has already completed the density tempering sequence. First, obtain an SMC sample (of size N) representative of the target distribution $f(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$. Then, conduct several rounds of "duplicate-and-boost" to rapidly multiply the sample

size. Duplication of the base sample k folds yields k identical sets of particles. Support-boosting moves in the same way as in Section 2.2.3 are meant to restore particle diversity for the sample of kN particles. Since the density-tempering bridge has been completely skipped, k-fold duplication is much more efficient in generating a larger sample for a target distribution.

We apply one round of 4-fold duplication of 1,000 SMC particles[10] to showcase how the method works on the non-convex optimization problem introduced in Section 2.2.4. Comparing Figure 5 with Figure 2, this 4-fold duplication significantly increases the particle density and thus increases the precision of the solution.



(a) Empirical distribution of the 4,000 particles from one SMC run with one round of 4-fold duplication

(b) Optimal solutions from 500 independent 4000-particle SMC runs

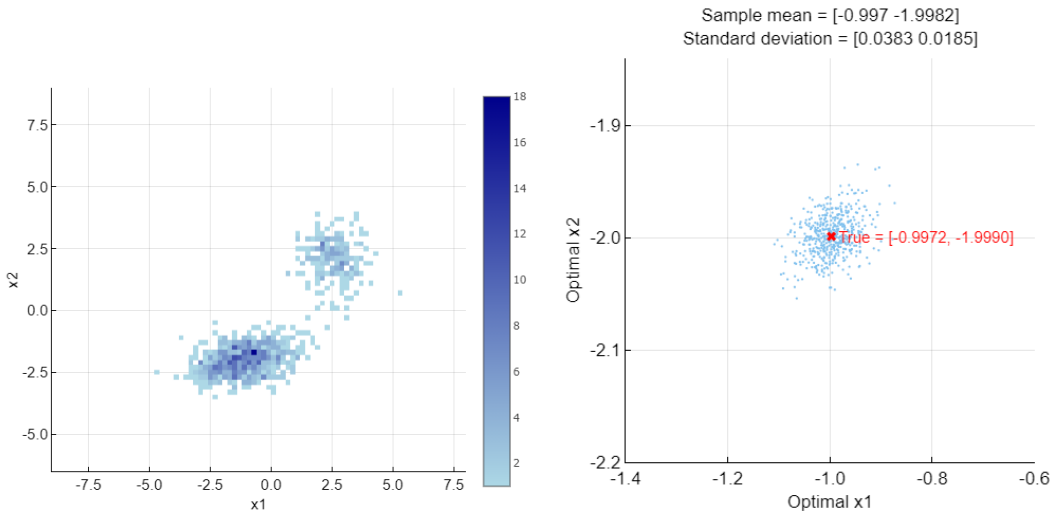
Figure 5: SMC optimization with one round of 4-fold duplication of 1,000 particles for the bimodal target function in equation (5)

3.2 Data cloning

Data cloning, which emerged from biostatistics (Lele et al., 2007, Lele et al., 2010), was initially proposed to compute the MLE and its inference for complex ecological hierarchical models. The same idea concurrently surfaced in Jacquier et al. (2007) for financial time series analysis. As the name suggests, the observed data $\mathcal{D}$ will be cloned many times, say, m, and treat each cloned copy as if it were an independent sample. The cloned target $[\mathcal{L}(\boldsymbol{\theta}|\mathcal{D})]^m$ becomes the new target of interest. It is evident that $\mathcal{L}(\boldsymbol{\theta}|\mathcal{D})$ and $[\mathcal{L}(\boldsymbol{\theta}|\mathcal{D})]^m$ share the same maximizer, and thus the MLE remains unaltered with cloning.

The optimal solution's variability due to Monte Carlo simulation can be reduced by cloning for two reasons. The first one is the usual compression of the variance because the sampled points begin to concentrate at the rate of m in response to powering. The second effect of cloning is more nuanced and caused by dampening multi-modality of the target function. Powering up the target function in effect widens the gap between the functional value at the global maximum

[10] The proposal sampler used in move steps follows what has been described in Footnote 1. Multiple move steps are made until the cumulative acceptance rate exceeds 200%.



(a) Empirical distribution of 1000 particles from one SMC run with data cloning

(b) Optimal solutions from 500 independent 1000-particle SMC runs with data cloning

Figure 6: SMC optimization with data cloning at the power of 4 for the bi-modal target function in equation (5)

and those functional values at other local maxima. As stated in Lele et al. (2007) and Lele et al. (2010), the target distribution becomes centered at the sample MLE with all other local maxima "flattened" when m is sufficiently large.

Data cloning implemented with a Bayesian computational approach, i.e., starting from the prior to the posterior distribution, can in effect remove the impact of the prior distribution as the number of cloned copies increases because the powered-up likelihood function begins to dominate.

The SMC algorithms introduced in Section 2 (i.e., density-tempered SMC and expanding-data SMC) can be used to generate particles representative of the cloned target. Moreover, the asymptotic variance of the MLE can be calculated by m times the variance estimated with the cloned SMC particles that targets $[\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})]^m$. We will take up the issue of statistical inference later in Section 6.2.

For general optimization purposes, cloning can still work for target functions that involve no data at all, i.e., $[f(\boldsymbol{x})]^m$. Its advantage is clearly shown in Figure 6 where cloning at the power of 4 is applied to the non-convex optimization problem introduced in section 2.2.4.[11] Evidently, Figure 6b reveals increased precision of the optimizer whereas Figure 6a shows the effect of cloning in reducing the number of particles scattered around a local mode.

Table 2 illustrates the performance of data cloning and k-fold duplication for the target function in Equation (5). The table presents the results from multiple rounds of cloning at the power

[11] For each cloning round, we re-initialize particles using the means and standard deviations derived from the existing SMC particles. The standard deviations are properly scaled down by the square root of the incremental cloning factor in each round, i.e., $\sqrt{4}$, to anticipate the variance reduction effect of cloning. The proposal distribution used in move steps follows what has been described in Footnote 1.

of 4 and 4-fold duplication, respectively. Specifically, the i^{th} cloning round raises the target function to the power of 4^i using N particles, whereas k-fold duplication creates $4^i N$ particles ($N = 1,000$ for this table). Table 2 clearly indicates that cloning is more efficient than k-fold duplication for this bi-modal target function. Although cloning does not change the rate of variance reduction, it lowers the convergence constant, making the algorithm more effective in reducing Monte Carlo errors.

Round	Cloning (power = 4)		4-Fold Duplication	
	x_1	x_2	x_1	x_2
1	-0.9970	-1.9982	-0.9964	-1.9998
	(0.0383)	(0.0185)	(0.0437)	(0.0222)
2	-0.9976	-1.9994	-0.9981	-1.9983
	(0.0154)	(0.0074)	(0.0220)	(0.0109)
3	-0.9976	-1.9992	-0.9966	-1.9991
	(0.0081)	(0.0040)	(0.0109)	(0.0056)
4	-0.9972	-1.9989	-0.9973	-1.9990
	(0.0041)	(0.0019)	(0.0053)	(0.0027)

Table 2: Comparison of the two precision-enhanced optimizers (mean and standard deviation) calculated with k-fold duplication and cloning using 500 independent SMC runs

The above results reflect an important fact. Cloning and k-fold duplication share the same rate of convergence as far as the SMC maximum is concerned but likely face different convergence constants. This becomes fairly intuitive if we consider a second-order Taylor expansion of the logarithm of the target function at the maximum. Cloning at the power of 4 in effect has the same expansion except for reducing the negative Hessian matrix by a factor of 4.[12] In a small neighborhood of the maximum, the target function is very close to a multivariate normal density function and cloning at the power of 4 is equivalent to halving the standard deviation in all dimensions. On the other hand, a 4-fold duplication amounts to filling in the same neighborhood with 4 times of SMC particles at the original standard deviation, which is well known by the standard Monte Carlo theory to yield the same effect on gaining precision.

As to why the convergence constants may differ, we can appreciate it with a bi-modal target function like our example. A fraction of SMC particles are generated under a local mode which contributes nothing to refining the estimate for the global maximum. Without cloning, those SMC particles essentially have been wasted. The same argument applies to skewed single-mode target functions. Another way to see this point is to consider a case for which the two approaches can actually share the same convergence constant. Had we directly targeted a normal density function, cloning and k-fold duplication would carry the same convergence constant in addition to facing the same convergence rate.

The above comparison shows the superiority of cloning over k-fold duplication. However, it is only true for continuous target functions where further solution refinements are feasible. When the target function is discontinuous as in Section 2.5 or totally discrete as in Section 2.6, only k-fold duplication is applicable because cloning will always cause the SMC sample to degenerate as m increases.

A nuanced complementarity between k-fold duplication and cloning deserves elaboration. After

[12] This property will become clear later in Section 6.2.

completing a sufficient number of cloning rounds, one should actually switch to k-fold duplication if the interest goes beyond finding the optimum. First of all, the quality of the SMC optimum under two refinement approaches at that point of cloning has become identical in terms of either the convergence rate or the convergence constant, because the cloning rounds completed have in effect removed multi-modality and/or skewness. Under cloning, the SMC sample size remains constant but k-fold duplication increases it at the k-fold speed. The standard deviations or other statistics deduced from the SMC sample, for example, will be increasingly more accurate under k-fold duplication but will not be so with cloning.

Significant cloning is more demanding on the coding practice because numerical precision can be easily lost if, for example, one sticks to multiplications and divisions of exponential functions in the importance weight instead of thinking in terms of additions and subtractions of their logarithmic equivalent quantities. Exponentiation should be reserved as the last step on the need basis. One should also refrain from multiplying the potentially very large cloning factor and save it for the last step after all additions and subtractions have been completed.

3.3 Data-cloning SMC^2

It is straightforward to apply k-fold duplication on statistical models embedded with latent variable(s) to turn it into an SMC^2 setting. But applying data cloning on a model with latency is much more nuanced and complex because the particle filter's accuracy in the inner SMC layer will begin to interfere with the rapidly decreased parameter dispersion due to cloning.

Continuing with the notations in Section 2.7, we now describe the data-cloning SMC^2 algorithm of Duan et al. (2020). When necessary, we modify the notations slightly to accommodate data cloning. This algorithm relies on constructing a sequence of targets with their marginal distributions proportional to the cloned likelihoods, $[\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})]^m$ for $m = 1, 2, \cdots$, using m independent runs of a fixed-parameter particle filter (or average likelihood for non-dynamic models) each with p particles. The common feature of this inner layer is that they are unbiased estimates of the true likelihood powered up to m.

In addition, a tempering bridge of intermediate cloned target functions is defined and governed by $0 \leq \delta \leq 1$. Let $I_m(\boldsymbol{\theta})$ denotes the (re-)initialization sampler at the m cloning stage. Specifically, the density-tempered target function is

$$f_{\delta,m}(\boldsymbol{\theta}, \boldsymbol{U}_{1:mp}|\boldsymbol{\mathcal{D}}) \propto \begin{cases} \left(\frac{\widehat{\mathcal{L}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}},\boldsymbol{U}_{1:p})}{I(\boldsymbol{\theta})}\right)^{\delta} \varphi(\boldsymbol{U}_{1:p}|\boldsymbol{\mathcal{D}},\boldsymbol{\theta}) I_1(\boldsymbol{\theta}) & \text{if } m = 1 \\ \left(\frac{f_{1,m-1}(\boldsymbol{\theta},\boldsymbol{U}_{1:(m-1)p}|\boldsymbol{\mathcal{D}})\widehat{\mathcal{L}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}},\boldsymbol{U}_{(m-1)p+1:mp})}{I_m(\boldsymbol{\theta})}\right)^{\delta} \varphi(\boldsymbol{U}_{(m-1)p+1:mp}|\boldsymbol{\mathcal{D}},\boldsymbol{\theta}) I_m(\boldsymbol{\theta}) & \text{if } m > 1 \end{cases} \tag{13}$$

At $m = 1$, it is essentially the SMC^2 algorithm of Duan and Fulop (2015) being implemented without the prior distribution per the early discussion in Section 2.7. When $m > 1$, Duan et al. (2020) set $I_m(\boldsymbol{\theta}) = f_{1,m-1}(\boldsymbol{\theta}, \boldsymbol{U}_{1:(m-1)p}|\boldsymbol{\mathcal{D}})$ as the re-initialization distribution because it is already represented by the SMC sample at the $m-1$ cloning stage. Traversing through a tempering bridge indexed by $\boldsymbol{\delta}$ then advances the system to the desired m. By Equation (13), advancing from δ_1 to δ_2 faces a simplified importance weight: $\left(\widehat{\mathcal{L}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}, \boldsymbol{U}_{1:p})/I(\boldsymbol{\theta})\right)^{\delta_2-\delta_1}$ if $m = 1$, or $\left(\widehat{\mathcal{L}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}, \boldsymbol{U}_{(m-1)p+1:mp})\right)^{\delta_2-\delta_1}$ if $m > 1$ and deploying $I_m(\boldsymbol{\theta}) = f_{1,m-1}(\boldsymbol{\theta}, \boldsymbol{U}_{1:(m-1)p}|\boldsymbol{\mathcal{D}})$ as in Duan et al. (2020).

It is evident from the above that the m-stage outcome at the end of the tempering bridge becomes

$$\begin{aligned} & f_{1,m}(\boldsymbol{\theta}, \boldsymbol{U}_{1:mp}|\boldsymbol{\mathcal{D}}) \\ \propto & \left(\prod_{i=1}^{m} \widehat{\mathcal{L}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}, \boldsymbol{U}_{(i-1)p+1:ip})\right)\left(\prod_{i=1}^{m} \varphi(\boldsymbol{U}_{(i-1)p+1:ip}|\boldsymbol{\mathcal{D}}, \boldsymbol{\theta})\right) \end{aligned} \tag{14}$$

So, this SMC^2 algorithm actually targets, ignoring the component associated with the auxiliary random variables needed for the particle filter, $\prod_{i=1}^{m} \widehat{\mathcal{L}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}, \boldsymbol{U}_{(i-1)p+1:ip})$ whose expected value equals $[\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})]^m$ because the particle filter is known to provide an unbiased estimate of the likelihood function.

Data cloning will magnify the Monte Carlo error inherent in the particle filter, i.e., the inner layer of SMC^2. Duan et al. (2020) offered a self-adapted way to determine p in response to an increased m. Basically, p will be increased for the added block from a new independent particle filter run whenever the acceptance rate in the Metropolis-Hastings move drops below a threshold value, say, 20%. The increased p requires a re-initialization of the algorithm but it is highly efficient by leveraging the already obtained SMC particles to design the re-initialization sampler.

Duan et al. (2020) has applied with success their SMC^2 algorithm on several prominent latent-variable models. They also demonstrated that it is far more efficient than, for example, the method proposed by Johansen et al. (2008) or the direct implementation of Duan and Fulop (2015) with data cloning[13].

4 Assess reliability of the SMC maximum

Duan (2019) developed a method for assessing the quality of the SMC optimum by treating it as a maximum order statistic. The method hinges on applying the Fisher-Tippett-Gnedenko Extreme Value Theorem (EVT) to the sample of functional values evaluated with the SMC particles. Since the maximum exists, the EVT limit of the sample maximum functional value becomes the Weibull distribution, which can then be used to determine how far the current maximum found is from the true maximum and how large the probability is for further improvement.

The Weibull distribution for the block-maximum functional value is the EVT limit for large M:

$$F_{f_{\max}(M)}(z; f^u, \alpha, \eta) = \exp\left[-\left(\frac{f^u - z}{\eta}\right)^{\alpha}\right] \quad \text{for } z \leq f^u \tag{15}$$

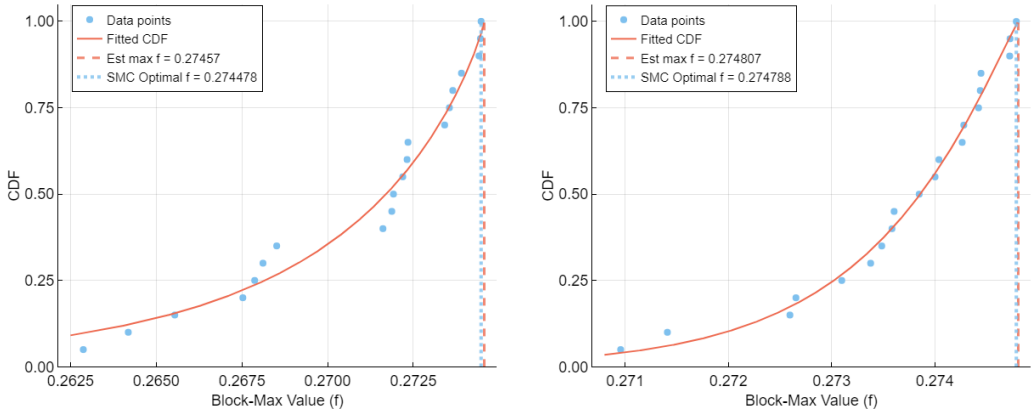
This distribution has three parameters: f^u, α and η.[14] Let $\bar{f}^u$ denote the maximum functional value recorded in the entire SMC progression, which may be strictly larger than any $f_{\max}(M) = \max\{f_i; i = 1, 2, \cdots, M\}$, where $M < N$ is the maximum functional value over a block of size M in the final SMC sample of size N.

[13] A direct implementation of Duan and Fulop (2015) with data cloning means targeting $\left[\widehat{\mathcal{L}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}, \boldsymbol{U}_{1:p(m)})\right]^m$ where $p(m)$ used in the particle filter reflects cloning and is of a comparable size as the competing algorithm.

[14] The Weibull distribution in (15) can be treated as a two- or three-parameter distribution function. Taking f^u, α and η as unknown, this distribution has three parameters. Let $G(f)$ be the distribution of target function's value and $G^{\leftarrow}(x) \equiv \inf\{f : G(f) > x\}$ be its the left continuous inverse. If we use the empirical distribution derived from the final SMC sample to approximate $G(f)$, then $\eta = f^u - G^{\leftarrow}(1 - 1/M')$ is redundant and the Weibull distribution only has two unknown parameters.

In estimation, it is natural to constrain $f^u \geq \bar{f}^u$. The parameter estimate $\hat{f}^u \geq \bar{f}^u$ predicts the true maximum functional value. The estimated exceedance probability for making further improvement by increasing N becomes $1 - F_{f_{\max}(M)}(\bar{f}^u; \hat{f}^u, \hat{\alpha}, \hat{\eta})$.

We demonstrate how the EVT works on optimizing the target function in equation (5). First, we obtain 2,000 SMC particles sampled without cloning and generate another sample of the same size with cloning. Each set of 2,000 particles is randomly partitioned into 20 blocks each with 100 particles. Fitting a Weibull distribution to the 20 block-maximum functional values yields Figure 7 for the two SMC samples of 2,000 particles each. The two estimated Weibull distributions suggest that the currently found maximum in either case is very close to the EVT-predicted optimum. In principle, the EVT prediction using the cloned SMC sample at the power of 4 should provide a more accurate prediction. However, Figures 7a and 7b suggest a rather minor improvement.



(a) Without cloning

(b) With cloning at the power of 4

Figure 7: The EVT-predicted Weibull distribution (constructed with 2,000 SMC particles partitioned into 20 random blocks each with 100 particles) for the maximal functional value of the target in equation (5)

The EVT-estimated exceedance probability is 3.52% using the 2,000 SMC particles without data cloning whereas that probability becomes 0.93% with data cloning at the power of 4. Even with a small chance of improving the SMC solution, say, 0.93%, the magnitude of potential improvement (from 0.274788 to 0.274807) as shown in Figure 7b is negligible for most practical purposes. Further improvements, if called for, are straightforward with additional rounds of cloning or 4-fold duplication to raise the refinement factor to 16, 64 or even higher.

5 MCMC vs. SMC for optimization

Markov Chain Monte Carlo (MCMC) methods are a class of algorithms that are commonly used for generating samples from a desired distribution. The most popular MCMC methods are based on the Gibbs sampler (Geman and Geman, 1984) or the Metropolis-Hastings algorithm. In principle, they can also be applied to solving optimization problems. Lele et al. (2007)'s data cloning through applying MCMC can, for example, be interpreted as optimization. In the meta-heuristic category of optimization methods, the Metropolis-Hastings algorithm has also

been adapted to work for simulated annealing, but it is not really an MCMC algorithm.

In contrast to SMC, there are several disadvantages with MCMC methods. First, the generated Markov chain needs "burn-in" before the simulated particles can be really regarded as random draws from the target distribution. The length of burn-in period can also be difficult to adequately determine. Second, the choice of the proposal distributions is a crucial factor determining MCMC's performance. Unlike SMC, where one can simply derive an independent proposal from the particle set, MCMC only has one chain before reaching its stationary distribution, which is of limited value to designing a natural proposal sampler. Finally, the MCMC particles after burn-in can be highly autocorrelated. These MCMC-induced autocorrelations in turn render the estimator much less precise.

In an intuitive way, SMC can be interpreted as concurrently running many independent MCMC chains but needs no burn-in at all. First, we note a key property of Markov transition kernel permitting stationary distribution, and that is, an input sample drawn from its stationary distribution will churn out a different sample from the same stationary distribution. Since SMC starts from a sample based on the stationary distribution via importance sampling and resampling, burn-in is not needed. The use of the Metropolis-Hastings kernel is solely to boost the empirical support, i.e., to get rid of duplicated particles due to resampling.

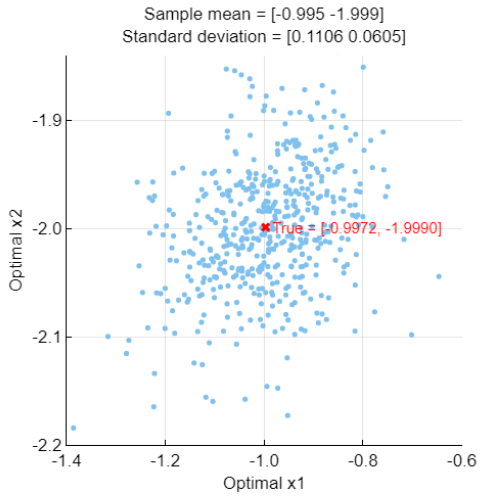
We now use MCMC to solve the same multi-modality problem defined in Section 2.2.4 and compare its performance with the density-tempered SMC method. Specifically, we run 500 independent MCMC chains using the Metropolis-Hasting algorithm with a random walk proposal.[15] To ensure reasonable convergence, we first run 10,000 iterations for each of the 500 chains for the burn-in purpose. We then obtain 1,000 post-burn-in particles by iterations in each MCMC run and use the sample to find the MCMC maximum. Figure 8a displays the distribution of the 500 optimal solutions. It is fairly clear from comparing them with those in Figure 2b that the MCMC solutions vary much more. Evidently, it is simply a manifestation of the fact that those MCMC particles are highly autocorrelated in either one of the two dimensions as shown in Figure 8b.

A simple but not yet recognized improvement step can be applied to the MCMC sample. We can run the MCMC sample through the SMC support boosting step several times to obtain an independent MCMC sample. In fact, one can first apply k-fold duplication to significantly enlarge the MCMC sample and then put the particles through support boosting. To illustrate the power of such a simple trick, we only take the first 500 MCMC particles after burn-in and apply 2-fold duplication to yield a sample of 1,000 particles. Send it through the support-boosting step five times using the Metropolis-Hastings kernel constructed with the independent proposal sampler that is normally distributed with the means and variances based on the original sample of 500 MCMC particles. Evidently from inspecting Figure 8c vis-a-vis Figure 8a, the SMC-modified MCMC sample has delivered a much better performance.

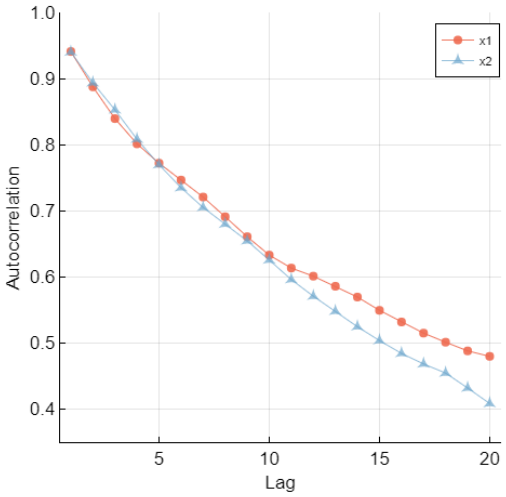
Figure 8d clearly shows that the modified sample no longer exhibits any meaningful autocorrelations. Skipping particles in the MCMC sample has been a typical way of reducing sample dependency. The autocorrelation patterns exhibited in Figure 8b suggests that even retaining one particle in every block of 20 will still burden the sample with significant autocorrelations.

The particle-MCMC approach in Andrieu et al. (2010) and Doucet et al. (2015) have been proposed in the literature to handle statistical models with latent variables. The underlying idea

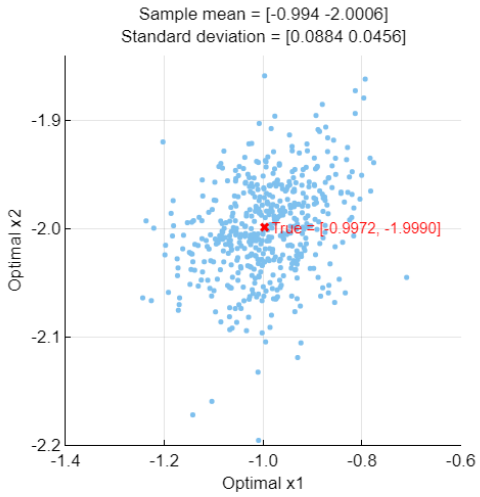
[15] We initialize the sample for (x_1, x_2) from two independent single-variable samplers with each being $\mathcal{N}(0, 5^2)$. For the random walk proposal in the Metropolis-Hasting algorithm, the standard deviation is set to 1.



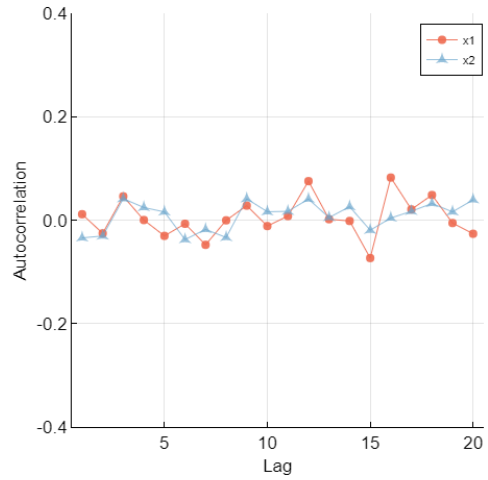
(a) Optimal solutions for the bi-modal target function in Equation (5) from 500 independent MCMC runs



(b) Autocorrelations of the sample from one MCMC run



(c) Optimal solutions for the bi-modal target function in Equation (5) from 500 independent SMC-modified MCMC runs



(d) Autocorrelations of the sample from one SMC-modified MCMC run

Figure 8: Optimization performance of MCMC and SMC-modified MCMC using a sample of 1,000 particles after burn-in

is that marginalization first with a particle filter can help reduce the complexity of the problem for MCMC to focus on parameter estimation. The SMC^2 algorithm discussed in Section 2.7 and 3.3 relies on the same marginalization idea. Now it should be fairly clear that SMC^2 can be expected to be more efficient than particle-MCMC for latent-variable models.

6 Statistical inference

6.1 Bayesian inference

SMC originates from Bayesian statistics and has been widely used in Bayesian inference. In those analyses, the particles generated by SMC are to represent the posterior distribution, summarizing the information in the data and combining it with the prior belief.

Applying SMC on state-space models is natural due to the inherent in the system. Kantas et al. (2009) and Kantas et al. (2015) provided earlier reviews of parameter estimation for general state-space models under both Bayesian and frequentist frameworks where SMC is only deployed to evaluate the fixed-parameter likelihood function. Our discussion in this section focus on using SMC as an Bayesian inference tool on parameters instead of a computational means for computing the likelihood value at the fixed parameters. Therefore, it is in line with Chopin (2002), Del Moral et al. (2006), Fulop and Li (2013), Duan and Fulop (2015), among others that rely on SMC to arrive at the posterior distribution.

Differing from the typical Bayesian literature, it will be more efficient to leverage the initialization sampler $I(\boldsymbol{\theta})$ as discussed in Section 2.2.2 and to reformulate the posterior distribution as a special case of a density-tempered target function; that is $\pi(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) = f_1(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$ where

$$f_\delta(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) \propto \left(\frac{\mathcal{L}(\boldsymbol{\theta};\boldsymbol{\mathcal{D}})\pi_0(\boldsymbol{\theta})}{I(\boldsymbol{\theta})}\right)^\delta I(\boldsymbol{\theta}) \tag{16}$$

Were we to initialize with the prior distribution, i.e., $I(\boldsymbol{\theta}) = \pi_0(\boldsymbol{\theta})$, that would bring us back to the standard approach commonly seen in the Bayesian literature.

Evidently, different initialization samplers do not alter the posterior distribution but may affect the computational efficiency of the density-tempered SMC algorithm. If the data is highly informative, the posterior distribution is by definition far different from the prior distribution. Thus, using the prior distribution to start the SMC process as typically seen in the literature is bound to be highly inefficient. Decoupling the initialization and prior distributions is an important insight because it allows analysts to experiment with different initialization samplers without changing their prior beliefs.

We now demonstrate the use of SMC^2 for Bayesian inference on the parameters of a latent variable model. Specifically, we deploy the SMC^2 algorithm of Duan and Fulop (2015) and illustrate with a standard discrete-time stochastic volatility model as in Duan et al. (2020) but using data on a different stock market index over a more recent time period.

The dynamics of the observed and latent processes under the standard discrete-time stochastic volatility model are specified as follows:

$$y_t = \sigma_{t-1}\epsilon_t \tag{17}$$

$$\log\sigma_t^2 = \alpha + \phi\log\sigma_{t-1}^2 + \sigma_v\,\eta_t \tag{18}$$

where

$$\begin{bmatrix}\epsilon_t\\ \eta_t\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}1 & \rho\\ \rho & 1\end{bmatrix}\right) \text{ and } \log\sigma_0^2 \sim \mathcal{N}\left(\frac{\alpha}{1-\phi}, \frac{\sigma_v^2}{1-\phi^2}\right)$$

Since there is no closed-form solution for the likelihood function of this model, SMC^2 is used where the inner layer runs a fixed-parameter particle filter to estimate the observed data likelihood and the outer layer conducts SMC on the model parameter to target the posterior distribution.

The parameter vector of interest is $\boldsymbol{\theta} = (\alpha, \phi, \sigma_v, \rho)$ with $-1 \leq \phi \leq 1$, $-1 \leq \rho \leq 1$ and $\sigma_v > 0$. The posterior inference on $\boldsymbol{\theta}$ is conducted with a 6-year daily log-return series ($T = 1507$) of the Straits Times Index between January 1, 2016 and December 31, 2021. The prior is set to be relatively uninformative, with a four-component truncated Gaussian distribution having the mean vector of $(0, 0.9, 0.5, 0)$ and a diagonal covariance matrix where the diagonal entries are $(1, 0.1, 0.5, 0.2)$. The initialization density $I(\boldsymbol{\theta})$ is set to be the same as the prior. The proposal sampler in the support-boosting step is constructed by fitting a four-component mixture normal model to the SMC sample of the parameter particles at that stage.

Parameter	Mean	Standard Deviation
α	-0.5608	0.0345
ϕ	0.9431	0.0035
σ_v	0.2658	0.0087
ρ	-0.3597	0.0077
run time	277.3293	33.3890

Table 3: Summary statistics on the posterior means of $\boldsymbol{\theta} = (\alpha, \phi, \sigma_v, \rho)$ from 50 independent SMC^2 runs on the discrete-time stochastic volatility model

We implement 50 independent runs of the SMC^2 algorithm and report in Table 3 some summary statistics on the posterior mean parameter values for $\boldsymbol{\theta}$. The standard deviations as compared to the corresponding means as reported in the table suggest that the Monte Carlo errors introduced by SMC are immaterial as far as the typical statistical inference is concerned.

In Figure 9, we plot the prior and posterior densities for each of the four parameters in $(\alpha, \phi, \sigma_v, \rho)$. The difference between the two densities is evident in all cases, suggesting that the data has helped in pinning down the parameter values. Worth pointing out is the fact that with the same prior distributions, the SMC^2 algorithm can run more efficiently if we simply adjust the initialization sampler away from the prior distributions. This is often possible because typical empirical studies go through several preliminary test runs on data. The results from each test run can help shape the initialization sampler but still leave the original prior belief intact.

Naturally, data cloning described earlier plays no role in Bayesian inference because any powering up in effect amounts to using the same data multiple times and thus renders the posterior distribution invalid. That explains why we did not apply the data-cloning SMC^2 algorithm of Duan et al. (2020) in the above empirical analysis. k-fold duplication discussed in Section 3.1 can however be utilized to efficiently increase the SMC particle size so as to improve the quality of Bayesian inference.

Bayesian and frequentist inferences become similar for large sample sizes on the basis of Walker's Consistency Theorem (Walker, 1969), which states that the posterior distribution converges to a multivariate normal distribution with mean equal to the MLE as the sample size increases. Since Bayesian and frequentist inferences for a large data sample are practically the same due to the Walker Theorem, the Bayesian computational approach has actually driven the initial development of data cloning for frequentist analysis (see Lele et al., 2007, Jacquier et al., 2007 and Lele et al., 2010).

6.2 Frequentist approach to inference

When the likelihood function is an optimization target, the MLE or some extreme value estimate is often the focus of frequentist inference. For example, the likelihood ratio test can be conducted

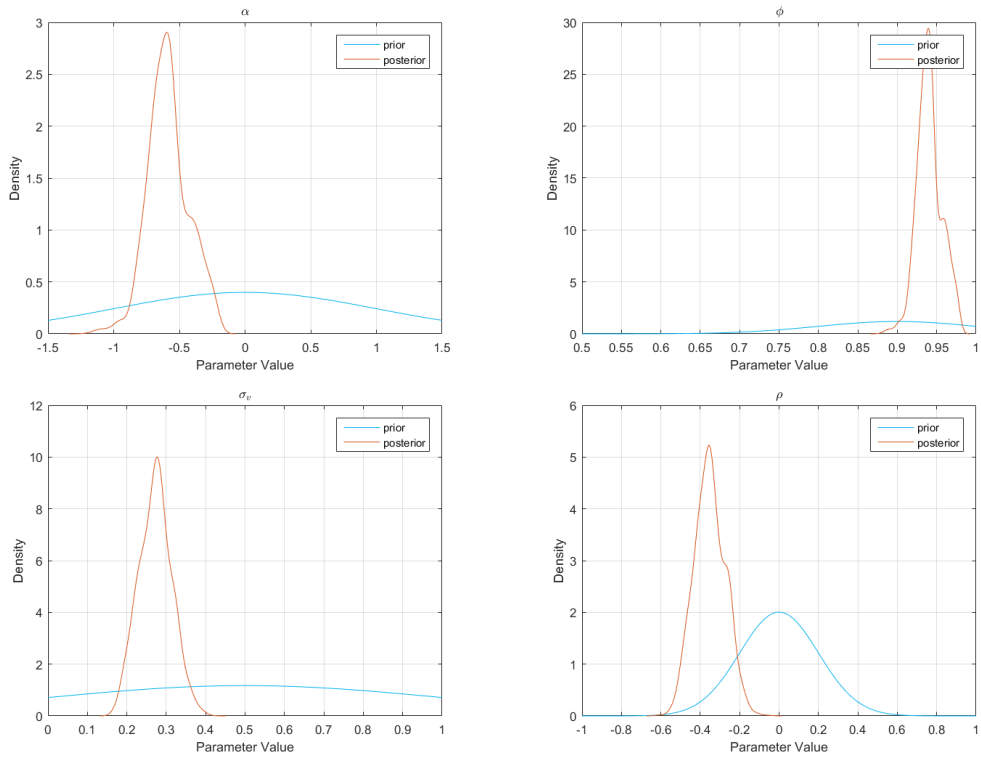


Figure 9: Plots of the prior and posterior densities for the parameters $(\alpha, \phi, \sigma_v, \rho)$ from one SMC^2 run on the stochastic volatility model

on the functional values of the MLE with and without the constraint. Beyond the likelihood ratio test, one typically needs to compute standard errors and/or correlations of the parameters for other inferences such as the Wald test and Z-test.

The inverse of the Fisher information matrix by the classical statistical theory provides the asymptotic variance matrix of the MLE, which can in turn be approximated by the negative of the Hessian of the log-likelihood function evaluated at the MLE (see Stuart et al., 1991). Typical tests revolve around the Hessian matrix evaluated at the MLE. Since a strong appeal of SMC optimization is to bypass derivatives, going back to computing derivatives (analytical or numerical) after optimization seems to be at odds with the very spirit of SMC optimization for statistical functions.

Moreover, analytical Hessian is often unavailable for statistical models in practice and requires approximation by, say, calculating a numerical Hessian. For general state-space models where the likelihood function may need a particle filter to compute, producing numerical derivatives may not even be possible due to inherent discontinuity of particle filter (see Kantas et al., 2009 and Kantas et al., 2015). Moreover, inverting a large-dimensional Hessian may also be numerically unstable, prone to yielding a poor-quality asymptotic variance.

Interestingly, the seminal work of Lele et al. (2007) and Lele et al. (2010) on data cloning has offered a direct solution to the Hessian computable from the cloned SMC sample. Let $\hat{\boldsymbol{\theta}}$

denote the MLE and $\boldsymbol{H}(\hat{\boldsymbol{\theta}}|\boldsymbol{\mathcal{D}})$ stand for the Hessian of the log-likelihood function. By their result, the posterior distribution with the cloned data likelihood converges to a multivariate normal distribution whose mean equals $\hat{\boldsymbol{\theta}}$ and variance is $-\frac{1}{m}\boldsymbol{H}(\hat{\boldsymbol{\theta}}|\boldsymbol{\mathcal{D}})^{-1}$ as the cloning power m approaches infinity.[16] Thus, the covariance matrix computed from the sufficiently cloned SMC particles becomes $-\frac{1}{m}\boldsymbol{H}(\hat{\boldsymbol{\theta}}|\boldsymbol{\mathcal{D}})^{-1}$. Multiplying by m gives rise to $-\boldsymbol{H}(\hat{\boldsymbol{\theta}}|\boldsymbol{\mathcal{D}})^{-1}$, the asymptotic covariance matrix for $\hat{\boldsymbol{\theta}}$.

To appreciate this intriguing result of Lele et al. (2007) and Lele et al. (2010), we provide an intuitive derivation.[17] The optimality conditions for the MLE ensure that the log-likelihood function's gradient equals zero and its Hessian is strictly negative definite. When these and some additional regularity conditions are met, applying a second-order Taylor expansion gives rise to

$$\begin{aligned}\log\left[\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})^m\pi_0(\boldsymbol{\theta})\right] &= \log\pi_0(\boldsymbol{\theta}) + m\log\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) \\ &\simeq \log\pi_0(\hat{\boldsymbol{\theta}}) + m\log\mathcal{L}(\hat{\boldsymbol{\theta}}|\boldsymbol{\mathcal{D}}) + \frac{m}{2}(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})'\boldsymbol{H}(\hat{\boldsymbol{\theta}}|\boldsymbol{\mathcal{D}})(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}) \quad (19)\end{aligned}$$

for large m because the powered-up target function causes $\boldsymbol{\theta}$ to be heavily concentrated in a small neighborhood of $\hat{\boldsymbol{\theta}}$. It follows that

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})^m\pi_0(\boldsymbol{\theta}) &\simeq \pi_0(\hat{\boldsymbol{\theta}})\mathcal{L}(\hat{\boldsymbol{\theta}}|\boldsymbol{\mathcal{D}})^m\exp\left[-\frac{(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})'\left(-m\boldsymbol{H}(\hat{\boldsymbol{\theta}}|\boldsymbol{\mathcal{D}})\right)(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})}{2}\right] \\ &\propto \Phi\left(\boldsymbol{\theta};\hat{\boldsymbol{\theta}},-\frac{1}{m}\boldsymbol{H}(\hat{\boldsymbol{\theta}}|\boldsymbol{\mathcal{D}})^{-1}\right) \quad (20)\end{aligned}$$

In the above, $\Phi(\cdot;\boldsymbol{\mu},\boldsymbol{\Sigma})$ is the multivariate normal density function with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Evidently, $\pi_0(\hat{\boldsymbol{\theta}})\mathcal{L}(\hat{\boldsymbol{\theta}}|\boldsymbol{\mathcal{D}})^m$ on the right-hand side has been absorbed into the norming constant for normality. In short, data cloning yields the same limiting distribution for either $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})^m\pi_0(\boldsymbol{\theta})$ or $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})^m$ after factoring in the norming constant because both are proportional to the same multivariate normal distribution. In short, the prior distribution in the context of data cloning becomes a moot point.

Regardless of the target function being $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})^m\pi_0(\boldsymbol{\theta})$ or $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})^m$, the covariance matrix can be approximated by m times of the sample covariance computed from the cloned SMC particles. Also interesting to note is the fact that the SMC approach allows for bypassing the inversion of the Hessian matrix, which near-singularity may become a source of numerical imprecision when there are many parameters.

When the likelihood function has multiple local maxima all sharing the same maximum functional value, i.e., the global maximum is not unique, data cloning is not expected to dampen those local maxima for an obvious reason. In short, data cloning cannot help resolve the fundamental lack of identifiability. That being said, data cloning may be a powerful tool for resolving estimability of poorly identified models where (1) the global maximum is unique but is closely followed by many local maxima of smaller but comparable functional values or (2) the likelihood function is relatively flat along some dimensions at around the global maximum. In essence, data cloning in the former case widens the gap between the functional value of the global maximum and those of

[16] In both Lele et al. (2007) and Lele et al. (2010), the authors stated the convergence to the Fisher information matrix. More precisely speaking, it is the negative of the Hessian of the log-likelihood function evaluated at the MLE, which is also known as the observed Fisher information matrix.

[17] For a rigorous proof, readers are referred to Lele et al. (2010).

other local maxima. Thus, it truly resolves both estimability and inference for poorly identifiable models of this type. For the latter, data cloning turns the target function into a much sharper form at around the global maximum. It helps resolves estimability through increasing precision, but the sampling errors, as having been discussed above, will still remain large after the variances being duly scaled back.

Importantly, convergence to normality by data cloning works for any data sample size. This should not be confused with the MLE's asymptotic convergence for which the data size must be large. When the sample size is small, cloning ensures a good approximation of the negative Hessian at the MLE by the sample covariance matrix derived from the SMC particles. However, the MLE's asymptotic convergence to normality will only occur for large sample sizes. One should be mindful of the fact that the negative Hessian computed with data-cloning SMC or other methods on a small sample will likely be a poor approximation of the Fisher information matrix.

Nevertheless, the MLE is often used in practice even for small samples. Technical issues concerning inverting the Hessian matrix occasionally surfaces. It is not uncommon to see a non-invertible Hessian in applied research possibly due to multicollinearity or small sample size (see Gill and King, 2004a, Gill and King, 2004b). We now use an example in Gill and King (2004b) to demonstrate how data-cloning SMC can lend a helping hand.

	Standard Logistic Regression			Data-cloning SMC			
	Full Set	Minus FED		Bounds [-30, 30]		Bounds [-45, 45]	
Parameter	Estimate	Estimate	Standard Error	Estimate	Standard Error	Estimate	Standard Error
BLK	5.8607	5.5808	5.3426	5.8607	5.8162	5.8607	5.7536
LAT	4.0793	3.2140	8.1122	4.0794	8.3173	4.0793	8.4498
GVT	-1.5347	-1.5874	1.2481	-1.5347	1.2806	-1.5347	1.2768
SVC	-2.9296	-2.5625	1.6965	-2.9296	1.7400	-2.9296	1.8117
FED	-26.0561	-	-	-28.9943	$1.5846 * 10^5$	-44.0251	$4.4206 * 10^7$
XFR	2.9755	2.3326	1.2965	2.9755	1.4453	2.9755	1.4168
POP	-1.4270	-0.8176	0.7232	-1.4270	0.7576	-1.4270	0.7599
Intercept	12.2686	6.4492	6.7366	12.2686	6.8957	12.2686	6.9219
Log-likelihood	-13.618618183359	-16.043125180069		-13.618618183309		-13.618618183296	

Table 4: Comparison of data-cloning SMC with standard logistic regression on the State of Florida sample

This data sample is for the State of Florida taken from the 1989 county-level economic and demographic survey in the US, which has 33 data instances and comprises 7 explanatory variables and a dichotomous response indicating whether 20% or more of the county's residents live in poverty. Gill and King (2004b) showed that logistic regression yields the MLE but encounters a non-invertible Hessian, making standard errors uncomputable. After omitting FED[18], the variable causing the difficulty, logistic regression can produce a sensible outcome. Gill and King (2004b) proceeded to offer two solutions while keeping in mind an important applied research consideration that dictates the use of same model specification for an across-state comparison study.

Data-cloning SMC enables us to conduct the MLE inference on this data sample and to reveal an underlying problem in connection to the MLE estimate on FED. Put it simply, the standard logistic regression software does not handle the situation well. First, we reproduce Gill and King

[18]FED is a dummy variable indicating whether federally owned lands make up 30 percent or more of a county's land area

(2004b)'s results with the Python package 'statsmodels' with its default optimization solver.[19] We then place loose bounds on the parameter for FED and run density-tempered SMC with cloning to produce the results in Table 4.[20] As indicated in the table, placing bounds, [-30, 30] or [-45, 45], helps solve the maximization problem and reveals the nature of the optimum. In short, the objective function slowly approaches its limiting value and the parameter value corresponding to FED decreases toward negative infinity as it should be for this data sample before losing numerical precision with the typical coding in float64.[21]

Importantly, the results indicate that other parameters and their standard errors are not materially impacted. To re-emphasize a key point, data-cloning SMC can directly generate a quality estimate for covariance matrix through sampling and avoid inverting the Hessian altogether. The standard error for the parameter on FED reported in Table 4 is suggestive, pointing to the fact that the SMC optimizer wants to go further negative in the FED dimension but is stopped by the bound.[22]

7 Conclusion

Despite its Bayesian origin, SMC can be a powerful tool for optimization, particularly for problems that are difficult for conventional methods. We have reviewed the literature by first focusing on optimization through casting all optimization problems as sampling tasks. SMC is well known in Bayesian statistics reflective of its origin but has received scant attention as a powerful global optimizer. Density-tempered SMC in particular can work for different types of objective functions, be they continuous or discrete, and can handle complex and/or non-convex constraints. We then proceed to discuss the use of SMC for statistical inference, either Bayesian or frequentist.

This article also discusses how the density-tempered and expanding-data SMC techniques can complement each other to effectively handle both offline and online optimization/inference tasks. Since the SMC maximum is equivalent to the maximum order statistic of a sample, the Extreme Value Theorem is readily applicable to assessing the quality of an SMC optimum. When sensing a need to improve the Monte Carlo precision, k-fold duplication and data cloning are two effective and simple techniques that can quickly improve the SMC solution's quality. Moreover, we have discussed efficient ways to conduct Bayesian inference and to avoid inverting the Hessian matrix for frequentist inference, which can sometimes be challenging.

Although we have explained and demonstrated the power of SMC for complex and challenging optimization tasks, it should not be viewed as a replacement for conventional gradient-based optimizers because they are highly efficient tools for convex optimization problems. If we were to use SMC to solve, for example, Lasso, it would be doable but would take several orders of magnitude more computing time to complete the work. This is because the highly efficient

[19] The reported values here are in essence same as those in Gill and King (2004b) with differences caused by the optimization package and default setting.

[20] Cloning is executed with the power of 4^i (i being the cloning round number) and terminated when an additional cloning round produces the log-likelihood improvement less than 10^{-8} and all parameter changes are smaller than 10^{-2} times their respective absolute parameter values. The SMC standard errors are computed after applying 64-fold duplication to the sample of 1,000 after cloning.

[21] FED is a dummy variable with the value of 1 occurs only in three cases and all their responses are 0. Naturally, the logistic regression's optimal coefficient goes to negative infinity to gain a better fit.

[22] Strictly speaking, Equation (19) is not directly applicable to the parameter for FED because its first derivative is not zero at the SMC solution. However, it has long reached the flat part in the vicinity of the lower bound to have a near-zero derivative.

proximal gradient ascent/descent method will typically obtain the Lasso solution in a fraction of a second.

SMC does not belong to the category of meta-heuristic methods but can be applied in a heuristic way much like the stochastic gradient ascent/descent algorithm has become the standard optimization tool for finding neural network solutions. Neural networks are obviously non-convex with respect to parameters (i.e., synaptic weights and biases). They are in fact full of local optima and saddle points. However, that has not prevented the usage of stochastic gradient ascent/descent method to heuristically solve neural-network models. For users who are not fixated on getting the true optimum, we see the potential of deploying SMC to prune those typically over-parameterized neural-network models while preserving their prediction accuracy.

Acknowledgments

This research is funded under the grant "Asian Institute of Digital Finance" supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of National Research Foundation, Singapore.

References

Andrieu, C., Doucet, A., & Holenstein, R. (2010). Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *72*(3), 269–342.

Carpenter, J., Cliffordy, P., & Fearnhead, P. (1999). An improved particle filter for non-linear problems. *IEE Proceedings - Radar, Sonar and Navigation*, *146*, 2–7(5). https://digital-library.theiet.org/content/journals/10.1049/ip-rsn_19990255

Chopin, N. (2002). A sequential particle filter method for static models. *Biometrika*, *89*(3), 539–551. http://www.jstor.org/stable/4140600

Chopin, N., Jacob, P. E., & Papaspiliopoulos, O. (2013). Smc2: An efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *75*(3), 397–426.

De Cock, D. (2011). Ames, iowa: Alternative to the boston housing data as an end of semester regression project. *Journal of Statistics Education*, *19*(3), null. https://doi.org/10.1080/10691898.2011.11889627

Del Moral, P., Doucet, A., & Jasra, A. (2006). Sequential monte carlo samplers. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, *68*(3), 411–436. http://www.jstor.org/stable/3879283

Douc, R., Cappé, O., & Moulines, E. (2005). Comparison of resampling schemes for particle filtering. *Proc of the 4th International Symposium on Image and Signal Processing and Analysis, ISPA'05*.

Doucet, A., Johansen, A. M. et al. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, *12*(656-704), 3.

Doucet, A., Pitt, M. K., Deligiannidis, G., & Kohn, R. (2015). Efficient implementation of markov chain monte carlo when using an unbiased likelihood estimator. *Biometrika*, *102*(2), 295–313.

Duan, J.-C. (2019). Variable selection with big data based on zero norm and via sequential monte carlo. *National University of Singapore working paper*. http://dx.doi.org/10.2139/ssrn.3377038

Duan, J.-C., & Fulop, A. (2009). Estimating the structural credit risk model when equity prices are contaminated by trading noises. *Journal of Econometrics*, *150*(2), 288–296. https://doi.org/https://doi.org/10.1016/j.jeconom.2008.12.003

Duan, J.-C., & Fulop, A. (2011). A stable estimator of the information matrix under em for dependent data. *Statistics and Computing*, *21*, 83–91. https://doi.org/10.1080/07350015.2014.940081

Duan, J.-C., & Fulop, A. (2013). Multiperiod corporate default prediction with the partially conditioned forward intensity. *National University of Singapore working paper*.

Duan, J.-C., & Fulop, A. (2015). Density-tempered marginalized sequential monte carlo samplers. *Journal of Business & Economic Statistics*, *33*(2), 192–202. https://doi.org/10.1080/07350015.2014.940081

Duan, J.-C., Fulop, A., & Hsieh, Y.-W. (2020). Data-cloning smc^2: A global optimizer for maximum likelihood estimation of latent variable models. *Computational Statistics & Data Analysis*, *143*(March). https://doi.org/https://doi.org/10.1016/j.csda.2019.106841

Duan, J.-C., & Li, S. (2021). Enhanced pd-implied ratings by targeting the credit rating migration matrix. *Journal of Finance and Data Science*, *7*(November), 115–125. https://doi.org/https://doi.org/10.1016/j.jfds.2021.05.001

Duan, J.-C., & Zhang, C. (2015). Non-gaussian bridge sampling with an application. *National University of Singapore working paper, Available at SSRN 2675877*.

Fulop, A., & Li, J. (2013). Efficient learning via simulation: A marginalized resample-move approach. *Journal of Econometrics*, *176*(2), 146–161. https://doi.org/https://doi.org/10.1016/j.jeconom.2013.05.002

Geman, S., & Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-6*(6), 721–741. https://doi.org/10.1109/TPAMI.1984.4767596

Gilks, W. R., & Berzuini, C. (2001). Following a moving target—monte carlo inference for dynamic bayesian models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *63*(1), 127–146. https://doi.org/https://doi.org/10.1111/1467-9868.00280

Gill, J., & King, G. (2004a). Numerical issues involved in inverting hessian matrices. *Numerical issues in statistical computing for the social scientist. Hoboken: Wiley*, 143–176.

Gill, J., & King, G. (2004b). What to do when your hessian is not invertible: Alternatives to model respecification in nonlinear estimation. *Sociological methods & research*, *33*(1), 54–87.

Glover, F., & Laguna, M. (1999). *Tabu search i* (Vol. 1). https://doi.org/10.1287/ijoc.1.3.190

Gordon, N., Salmond, D., & Smith, A. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, *140*, 107–113. https://doi.org/10.1049/ip-f-2.1993.0015

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, *57*(1), 97–109. https://doi.org/10.1093/biomet/57.1.97

Herawati, N., Nisa, K., Setiawan, E., Nusyirwan, N., & Tiryono, T. (2018). Regularized multiple regression methods to deal with severe multicollinearity. *International Journal of Statistics and Applications*, *8*(4), 167–172.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Jacquier, E., Johannes, M., & Polson, N. (2007). Mcmc maximum likelihood for latent state models. *Journal of Econometrics*, *137*(2), 615–640.

Jasra, A., Law, K. J., & Xu, Y. (2021). Multi-index sequential monte carlo methods for partially observed stochastic partial differential equations. *International Journal for Uncertainty Quantification*, *11*(3).

Johansen, A. M., Doucet, A., & Davy, M. (2008). Particle methods for maximum likelihood estimation in latent variable models. *Statistics and Computing*, *18*(1), 47–57.

Kantas, N., Doucet, A., Singh, S., & Maciejowski, J. (2009). An overview of sequential monte carlo methods for parameter estimation in general state-space models [15th IFAC Symposium on System Identification]. *IFAC Proceedings Volumes*, *42*(10), 774–785. https://doi.org/https://doi.org/10.3182/20090706-3-FR-2004.00129

Kantas, N., Doucet, A., Singh, S. S., Maciejowski, J., & Chopin, N. (2015). On Particle Methods for Parameter Estimation in State-Space Models. *Statistical Science*, *30*(3), 328–351. https://doi.org/10.1214/14-STS511

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Kitagawa, G. (1996). Monte carlo filter and smoother for non-gaussian nonlinear state space models. *J. Comput. Graph. Statist.*, *1*, 1–25. https://doi.org/10.2307/1390750

Land, A. H., & Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, *28*(3), 497–520. http://www.jstor.org/stable/1910129

Lele, S. R., Dennis, B., & Lutscher, F. (2007). Data cloning: Easy maximum likelihood estimation for complex ecological models using bayesian markov chain monte carlo methods. *Ecology letters*, *10*(7), 551–563.

Lele, S. R., Nadeem, K., & Schmuland, B. (2010). Estimability and likelihood inference for generalized linear mixed models using data cloning. *Journal of the American Statistical Association*, *105*(492), 1617–1625.

Liu, J., & Chen, R. (1998). Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, *93*. https://doi.org/10.1080/01621459.1998.10473765

Malik, S., & Pitt, M. K. (2011). Particle filters for continuous likelihood evaluation and maximisation. *Journal of Econometrics*, *165*(2).

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. https://doi.org/10.1063/1.1699114

NUS-CRI. (2021). *Credit research initiative technical report version: 2021 update 1*. https://d.nuscri.org/static/pdf/Technical%20report_2021.pdf

Pitt, M. (2002). Smooth particle filters likelihood evaluation and maximisation. *University of Warwick economics research paper*. https://warwick.ac.uk/fac/soc/economics/research/workingpapers/2008/twerp651.pdf

Satpathy, T., & Shah, R. (2022). Sparse index tracking using sequential monte carlo. *Quantitative Finance*. https://doi.org/10.1080/14697688.2022.2057353

Spall, J. C. (2003). *Introduction to stochastic search and optimization: Estimation, simulation, and control*. John Wiley & Sons, Inc.

Stuart, A., Ord, J. K., & Kendall, M. G. (1991). *Kendall's advanced theory of statistics: Classical inference and relationship*. Edward Arnold.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*(1), 267–288. http://www.jstor.org/stable/2346178

Walker, A. M. (1969). On the asymptotic behaviour of posterior distributions. *Journal of the Royal Statistical Society: Series B (Methodological)*, *31*(1), 80–88.

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing*, *4*, 65–85. https://doi.org/10.1007/BF00175354

Xu, Y. (2018). Sequential monte carlo algorithms for high-dimensional filtering and smoothing. *PhD's thesis, National University of Singapore*. http://scholarbank.nus.edu.sg/handle/10635/146933

Xu, Y., & Jasra, A. (2019). A method for high-dimensional smoothing. *Journal of the Korean Statistical Society*, *48*(1), 50–67.

Yeh, I.-C., & Lien, C.-h. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, *36*(2, Part 1), 2473–2480. https://doi.org/https://doi.org/10.1016/j.eswa.2007.12.020

Zhao, P., & Yu, B. (2006). On model selection consistency of lasso. *The Journal of Machine Learning Research*, *7*, 2541–2563.