The Surprisingly Overlooked Efficiency of SMC

(and how to make it even more efficient)

Nicolas Chopin

ENSAE, IPP

Section 1

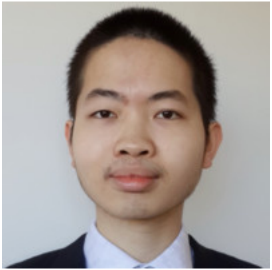
Introduction

Two talks in one

1. Overview of SMC samplers (and how/why they may be overlooked)

Two talks in one

1. Overview of SMC samplers (and how/why they may be overlooked)
2. Proposed improvement: waste-free SMC, joint work with:



Hai-Dang Dau (NUS)

Section 2

SMC: motivation

SMC: what is it?

SMC = an algorithm that combines sampling and MCMC steps in order to approximate a **sequence** of distributions, (π_t) (and their normalising constants).

SMC: what is it?

SMC = an algorithm that combines sampling and MCMC steps in order to approximate a **sequence** of distributions, (π_t) (and their normalising constants).

In some problems, you already have a sequence of distributions of interest.

SMC: what is it?

SMC = an algorithm that combines sampling and MCMC steps in order to approximate a **sequence** of distributions, (π_t) (and their normalising constants).

In some problems, you already have a sequence of distributions of interest.

In other problems, you may have a single distribution of interest, and you need to **design a sequence** that ends at the target. (I will give some recommendations.)

SMC: why?

- often only requirement is: being able to compute pointwise the (un-normalised) target density.

SMC: why?

- often only requirement is: being able to compute pointwise the (un-normalised) target density.
- parallelisable;

SMC: why?

- often only requirement is: being able to compute pointwise the (un-normalised) target density.
- parallelisable;
- estimates of the normalising constants;

SMC: why?

- often only requirement is: being able to compute pointwise the (un-normalised) target density.
- parallelisable;
- estimates of the normalising constants;
- adaptive (see 2nd part);

SMC: why?

- often only requirement is: being able to compute pointwise the (un-normalised) target density.
- parallelisable;
- estimates of the normalising constants;
- adaptive (see 2nd part);
- competitive with MCMC (e.g. C. and Ridgway, 2017; Buchholz et al, 2020).

PAC-Bayesian learning (see Alquier, 2021)

A ML method based on a pseudo-posterior:

$$\pi(\theta) \propto \mu(\theta) \exp\{-\lambda R_n(\theta)\}$$

where $R_n(\theta)$ is the empirical risk for parameter θ. For instance, for a classification task:

$$R_n(\theta) = \sum_{i=1}^{n} \mathbb{1}\{Y_i s_\theta(X_i) < 0\}$$

and s_θ could be e.g. $s_\theta(x) = \theta^T x$.

How do we choose λ?

PAC-Bayesian learning (see Alquier, 2021)

A ML method based on a pseudo-posterior:

$$\pi(\theta) \propto \mu(\theta) \exp\{-\lambda R_n(\theta)\}$$

where $R_n(\theta)$ is the empirical risk for parameter θ. For instance, for a classification task:

$$R_n(\theta) = \sum_{i=1}^{n} \mathbb{1}\{Y_i s_\theta(X_i) < 0\}$$

and s_θ could be e.g. $s_\theta(x) = \theta^T x$.

How do we choose λ?

$\Rightarrow$ Consider a sequence of values $0 = \lambda_0 < \ldots < \lambda_T$, use SMC to approximate the corresponding sequence of PAC-Bayes posteriors (and do e.g. cross validation).

PAC-Bayesian learning (see Alquier, 2021)

A ML method based on a pseudo-posterior:

$$\pi(\theta) \propto \mu(\theta) \exp\{-\lambda R_n(\theta)\}$$

where $R_n(\theta)$ is the empirical risk for parameter θ. For instance, for a classification task:

$$R_n(\theta) = \sum_{i=1}^{n} \mathbb{1}\{Y_i s_\theta(X_i) < 0\}$$

and s_θ could be e.g. $s_\theta(x) = \theta^T x$.

How do we choose λ?

$\Rightarrow$ Consider a sequence of values $0 = \lambda_0 < ... < \lambda_T$, use SMC to approximate the corresponding sequence of PAC-Bayes posteriors (and do e.g. cross validation).

See also Chernozhukov and Hong (2003), Bissiri et al (2016).

Sequential Bayesian estimation (and model choice)

Parametric model, prior $\mu(\theta)$. Data arrive **sequentially**: $Y_0, Y_1, \ldots$.

Sequential Bayesian estimation (and model choice)

Parametric model, prior $\mu(\theta)$. Data arrive **sequentially**: $Y_0, Y_1, \ldots$.

Consider sequence of posterior distributions:

$$\pi_t(\theta) = \frac{1}{p(y_{0:t})}\mu(\theta)p(y_{0:t}|\theta)$$

which may be used to infer θ sequentially, and also to perform **model choice**, through the marginal likelihood:

$$p(y_{0:t}) = \int \mu(\theta)p(y_{0:t}|\theta)d\theta$$

ABC (Approximate Bayesian Computation)

Model described only through a **simulator**: $y \sim p_\theta(y)$. ABC posterior:

$$\pi_\varepsilon(\theta, y) \propto \mu(\theta) p_\theta(y) \mathbb{1}\left\{d(y, y^\star) \leq \varepsilon\right\}$$

Use a sequence $\varepsilon_0 > \varepsilon_1 > ... > \varepsilon_T$.

What if I don't have a sequence

You wish to sample from a fixed distribution:

$$\pi(\theta) \propto \mu(\theta) \exp\{-V(\theta)\}$$

and/or compute its normalising constant.

What if I don't have a sequence

You wish to sample from a fixed distribution:

$$\pi(\theta) \propto \mu(\theta) \exp\{-V(\theta)\}$$

and/or compute its normalising constant.

Introduce the **tempering** sequence:

$$\pi_t(\theta) \propto \mu(\theta) \exp\{-\lambda_t V(\theta)\}$$

where $0 = \lambda_0 < \ldots, \lambda_T = 1$.

What if I don't have a sequence

You wish to sample from a fixed distribution:

$$\pi(\theta) \propto \mu(\theta) \exp\{-V(\theta)\}$$

and/or compute its normalising constant.

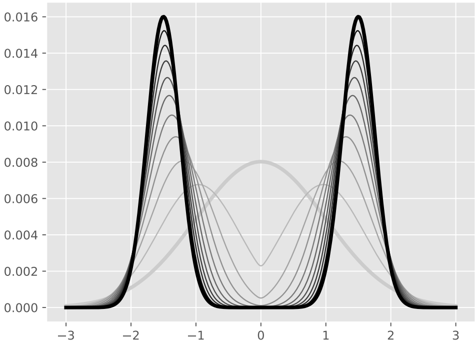
Introduce the **tempering** sequence:

$$\pi_t(\theta) \propto \mu(\theta) \exp\{-\lambda_t V(\theta)\}$$

where $0 = \lambda_0 < \dots, \lambda_T = 1$.

Linear interpolation on the log-scale.

Pictorial representation



Tempering sequence interpolating between $N(0,1)$ and a mixture of two Gaussians.

Fundamental points regarding tempering

- Tempering may be interpreted as entropic mirror descent: start at μ, then, iterate take a gradient step towards π (relative to the KL); see C., Crucinio and Korba (2024).

Fundamental points regarding tempering

- Tempering may be interpreted as entropic mirror descent: start at μ, then, iterate take a gradient step towards π (relative to the KL); see C., Crucinio and Korba (2024).
- Consider two distributions in dimension d. Then:

Fundamental points regarding tempering

- Tempering may be interpreted as entropic mirror descent: start at μ, then, iterate take a gradient step towards π (relative to the KL); see C., Crucinio and Korba (2024).
- Consider two distributions in dimension d. Then:
  - their χ^2-distance is $\mathcal{O}(e^d)$.

Fundamental points regarding tempering

- Tempering may be interpreted as entropic mirror descent: start at μ, then, iterate take a gradient step towards π (relative to the KL); see C., Crucinio and Korba (2024).
- Consider two distributions in dimension d. Then:
  - their χ^2-distance is $\mathcal{O}(e^d)$.
  - however, I can design $T = \mathcal{O}(d^{1/2})$ intermediate distributions, such that the χ^2-distance between π_t and π_{t+1} is $\mathcal{O}(1)$.

Fundamental points regarding tempering

- Tempering may be interpreted as entropic mirror descent: start at μ, then, iterate take a gradient step towards π (relative to the KL); see C., Crucinio and Korba (2024).
- Consider two distributions in dimension d. Then:
  - their χ^2-distance is $\mathcal{O}(e^d)$.
  - however, I can design $T = \mathcal{O}(d^{1/2})$ intermediate distributions, such that the χ^2-distance between π_t and π_{t+1} is $\mathcal{O}(1)$.

Fundamental points regarding tempering

- Tempering may be interpreted as entropic mirror descent: start at μ, then, iterate take a gradient step towards π (relative to the KL); see C., Crucinio and Korba (2024).
- Consider two distributions in dimension d. Then:
  - their χ^2-distance is $\mathcal{O}(e^d)$.
  - however, I can design $T = \mathcal{O}(d^{1/2})$ intermediate distributions, such that the χ^2-distance between π_t and π_{t+1} is $\mathcal{O}(1)$.

Tempering lifts the curse of dimensionality.

Fundamental points regarding tempering

- Tempering may be interpreted as entropic mirror descent: start at μ, then, iterate take a gradient step towards π (relative to the KL); see C., Crucinio and Korba (2024).
- Consider two distributions in dimension d. Then:
  - their χ^2-distance is $\mathcal{O}(e^d)$.
  - however, I can design $T = \mathcal{O}(d^{1/2})$ intermediate distributions, such that the χ^2-distance between π_t and π_{t+1} is $\mathcal{O}(1)$.

Tempering lifts the curse of dimensionality.

In practice, choose the successive λ_t so that the ESS$\approx \alpha N$. **Critical** for good performance.

Global optimisation

To minimise function V, consider again sequence

$$\pi_t(\theta) \propto \mu(\theta) \exp\{-\lambda_t V(\theta)\}$$

where this time $\lambda_t \to +\infty$.

Example: variable selection, θ is a binary vector (whether predictor is included or not), and $V(\theta)$ is e.g. BIC).

Connection with genetic programming.

Section 3

SMC samplers

SMC: more than sequential importance sampling

Now that we have a certain sequence (π_t) of distribution: we could:

- sample $\theta^n \sim \pi_0(\theta) = \mu(\theta)$ at time 0.

SMC: more than sequential importance sampling

Now that we have a certain sequence (π_t) of distribution: we could:

- sample $\theta^n \sim \pi_0(\theta) = \mu(\theta)$ at time 0.
- move to target π_1 through importance sampling; assign to particle θ^n weight:

$$w_1^n \propto \frac{\pi_1(\theta^n)}{\pi_0(\theta^n)}$$

SMC: more than sequential importance sampling

Now that we have a certain sequence (π_t) of distribution: we could:

- sample $\theta^n \sim \pi_0(\theta) = \mu(\theta)$ at time 0.
- move to target π_1 through importance sampling; assign to particle θ^n weight:

$$w_1^n \propto \frac{\pi_1(\theta^n)}{\pi_0(\theta^n)}$$

- move to target π_2 through a second IS step:

$$w_2^n \propto w_1^n \frac{\pi_2(\theta^n)}{\pi_1(\theta^n)}$$

SMC: more than sequential importance sampling

Now that we have a certain sequence (π_t) of distribution: we could:

- sample $\theta^n \sim \pi_0(\theta) = \mu(\theta)$ at time 0.
- move to target π_1 through importance sampling; assign to particle θ^n weight:

$$w_1^n \propto \frac{\pi_1(\theta^n)}{\pi_0(\theta^n)}$$

- move to target π_2 through a second IS step:

$$w_2^n \propto w_1^n \frac{\pi_2(\theta^n)}{\pi_1(\theta^n)}$$

SMC: more than sequential importance sampling

Now that we have a certain sequence (π_t) of distribution: we could:

- sample $\theta^n \sim \pi_0(\theta) = \mu(\theta)$ at time 0.
- move to target π_1 through importance sampling; assign to particle θ^n weight:

$$w_1^n \propto \frac{\pi_1(\theta^n)}{\pi_0(\theta^n)}$$

- move to target π_2 through a second IS step:

$$w_2^n \propto w_1^n \frac{\pi_2(\theta^n)}{\pi_1(\theta^n)}$$

However, this boils down to IS from π_0 to π_T. Usual weight degeneracy.

SMC: resample / move steps

At time $t-1$, we have a **weighted** sample that approximates π_{t-1}:

$$\sum_{n=1}^{N} W_{t-1}^{n} \varphi(\theta_{t-1}^{n}) \approx \pi_{t-1}(\varphi)$$

where $W_{t-1}^{n} = w_{t-1}^{n} / \sum_{m=1}^{N} w_{t-1}^{m}$ (normalised weights).

In order to **rejuvenate** the sample:

- resample: draw with replacement from set of N particles, according to the weights.

SMC: resample / move steps

At time $t-1$, we have a **weighted** sample that approximates π_{t-1}:

$$\sum_{n=1}^{N} W_{t-1}^{n} \varphi(\theta_{t-1}^{n}) \approx \pi_{t-1}(\varphi)$$

where $W_{t-1}^n = w_{t-1}^n / \sum_{m=1}^N w_{t-1}^m$ (normalised weights).

In order to **rejuvenate** the sample:

- resample: draw with replacement from set of N particles, according to the weights.
- move the resampled particles according to a MCMC kernel that leaves π_{t-1} invariant.

SMC sampler algorithm

Operations involving n are performed for $n = 1, \dots, N$.

for $t \leftarrow 0$ **to** T **do**
- **if** $t = 0$ **then**
  - $\theta_0^n \sim \pi_{-1}$
- **else**
  - $A_t^{1:N} \sim \texttt{resample}(N, W_{t-1}^{1:N})$
  - $\theta_t^n \sim M_t(\theta_{t-1}^{A_t^n}, d\theta_t)$ (M_t leaves invariant π_{t-1})
- $w_t^n \leftarrow \pi_t(\theta_t^n)/\pi_{t-1}(\theta_t^n)$
- $W_t^n \leftarrow w_t^n / \sum_{m=1}^N w_t^m$

An example of a MCMC kernel: random walk Metropolis

The algorithm (with input: θ):

- Generate $\theta^p \sim N(\theta, \Sigma)$
- With probability $\alpha \wedge 1$, return θ^p, otherwise return θ, where

$$\alpha = \frac{\pi(\theta^p)}{\pi(\theta)}$$

defines a Markov kernel that leaves invariant π.

Choice of Σ critical for good performance. If $\pi \approx N(\mu, S)$, recommended to take $\Sigma = cS$, with $c = (2.38)^2/d$.

An example of a MCMC kernel: random walk Metropolis

The algorithm (with input: θ):

- Generate $\theta^p \sim N(\theta, \Sigma)$
- With probability $\alpha \wedge 1$, return θ^p, otherwise return θ, where

$$\alpha = \frac{\pi(\theta^p)}{\pi(\theta)}$$

defines a Markov kernel that leaves invariant π.

Choice of Σ critical for good performance. If $\pi \approx N(\mu, S)$, recommended to take $\Sigma = cS$, with $c = (2.38)^2/d$.

Many other types of MCMC (e.g. Gibbs, HMC, NUTS, etc.).

Practical implementation within a SMC sampler

A default strategy in SMC samplers is use (as the MCMC kernel that moves the particles at time t) k step of random walk Metropolis, with

$$\Sigma = c \times \widehat{\Sigma}$$

and $\widehat{\Sigma}$ is the empirical covariance matrix of the weighted particles.

Note how easy it is to properly tune the MCMC kernel.

But: how to choose k?

Numerical experiment: logistic regression, sonar dataset

For details, see Chap. 16 of C. and Papaspiliopoulos (2020).

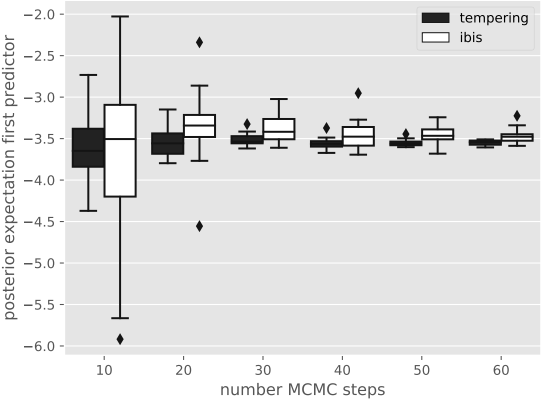


Figure 1: Box-plots over 50 runs of estimate of posterior expectation of first component, as a function of k (number of MCMC steps)

Numerical experiment (ii)

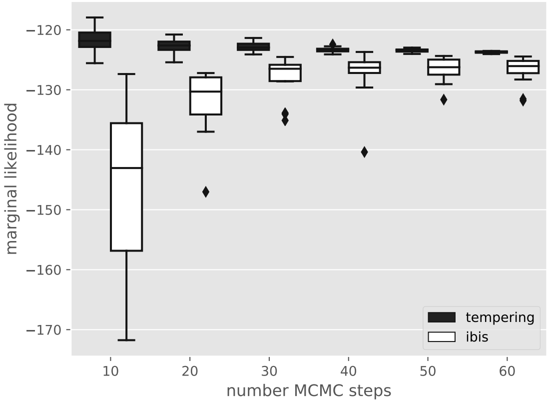
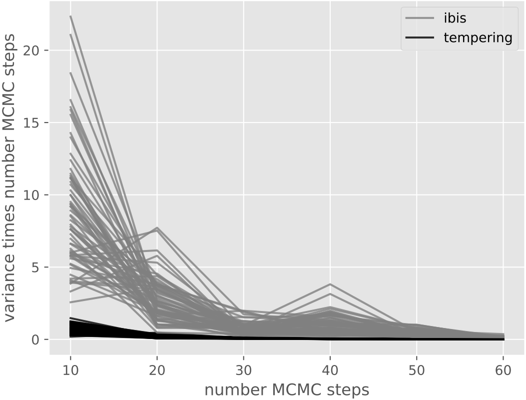


Figure 2: Same plot for log marginalising constant

Numerical experiment (iii)



Questions / remarks

- How do we choose k in practice? (without doing many pilot runs)

Questions / remarks

- How do we choose k in practice? (without doing many pilot runs)
- Should we have the same k at all iterations?

Questions / remarks

- How do we choose k in practice? (without doing many pilot runs)
- Should we have the same k at all iterations?
- If k is large, not using the intermediate states seem **wasteful**.

Section 4

Waste-free SMC

Basic idea

Let M, P two integers such that $N = M \times P$.

At iteration t:

- resample M particles;

Basic idea

Let M, P two integers such that $N = M \times P$.

At iteration t:

- resample M particles;
- apply $P - 1$ MCMC steps to each of the M resampled particles;

Basic idea

Let M, P two integers such that $N = M \times P$.

At iteration t:

- resample M particles;
- apply $P - 1$ MCMC steps to each of the M resampled particles;
- keep all the intermediate steps $\Rightarrow$ N particles. **No waste.**

Questions

- Validity?

Questions

- Validity?
- how to choose pair (M, P) (for a given N)?

P fixed, $M \to \infty$

In that regime, waste-free SMC is equivalent to a standard SMC sampler, where the target at time t corresponds to the distribution of a stationary Markov chain of length P.

Implies that:

- algorithm converges (as $N \to \infty$, while keeping P fixed)

However, this regime usually does not lead to best performance.

P fixed, $M \to \infty$

In that regime, waste-free SMC is equivalent to a standard SMC sampler, where the target at time t corresponds to the distribution of a stationary Markov chain of length P.

Implies that:

- algorithm converges (as $N \to \infty$, while keeping P fixed)
- estimate of normalising constant is unbiased.

However, this regime usually does not lead to best performance.

M fixed (or grows slowly), $P \to \infty$

$$\sqrt{N}\left(\frac{1}{N}\sum_{n=1}^{N}\varphi(X_t^n) - \pi_{t-1}(\varphi)\right) \Rightarrow \mathcal{N}\left(0, \tilde{\mathcal{V}}_t(\varphi)\right)$$

$$\sqrt{N}\left(\sum_{n=1}^{N}W_t^n\varphi(X_t^n) - \pi_t(\varphi)\right) \Rightarrow \mathcal{N}\left(0, \mathcal{V}_t(\varphi)\right)$$

where

$$\tilde{\mathcal{V}}_t(\varphi) = v_\infty(M_{t-1}, \varphi)$$

$$\mathcal{V}_t(\varphi) = \tilde{\mathcal{V}}_t\left(\bar{G}_t(\varphi - \pi_t\varphi)\right),$$

and $v_\infty(M_t, \varphi)$ is the asymptotic variance of a **stationary** Markov chain (Y_t) with kernel M_t:

$$v_\infty(M_t, \varphi) = \mathrm{Var}\left(\varphi(Y_0)\right) + 2\sum_{p=1}^{\infty}\mathrm{Cov}\left(\varphi(Y_0), \varphi(Y_p)\right).$$

Asymptotic variances

Note that these asymptotic variances:

- do not depend on M;

Asymptotic variances

Note that these asymptotic variances:

- do not depend on M;
- do not depend on previous iterations;

Asymptotic variances

Note that these asymptotic variances:

- do not depend on M;
- do not depend on previous iterations;
- suggest the following interpretation of the N particles: as M independent **stationary** chains of length P.

Asymptotic variances

Note that these asymptotic variances:

- do not depend on M;
- do not depend on previous iterations;
- suggest the following interpretation of the N particles: as M independent **stationary** chains of length P.

Asymptotic variances

Note that these asymptotic variances:

- do not depend on M;
- do not depend on previous iterations;
- suggest the following interpretation of the N particles: as M independent **stationary** chains of length P.

$\Rightarrow$ We can use adapt standard (initial sequence, spectral, etc.) variance estimators for MCMC chains to get a single-run estimate of the variance of our particle estimates.

Practical implications

- Take $M \ll N$.

Practical implications

- Take $M \ll N$.
- Take $M \geq$ the number of cores on a parallel machine.

Practical implications

- Take $M \ll N$.
- Take $M \geq$ the number of cores on a parallel machine.
- Single-run variance estimate based on the $M-$chain interpretation.

Section 5

Numerical experiments

Numerical experiment I

- target: posterior of a logistic regression, sonar dataset ($d = 63$, challenging, see C. and Ridgway, 2017).

Numerical experiment I

- target: posterior of a logistic regression, sonar dataset ($d = 63$, challenging, see C. and Ridgway, 2017).
- sequence: (adaptive) tempering.

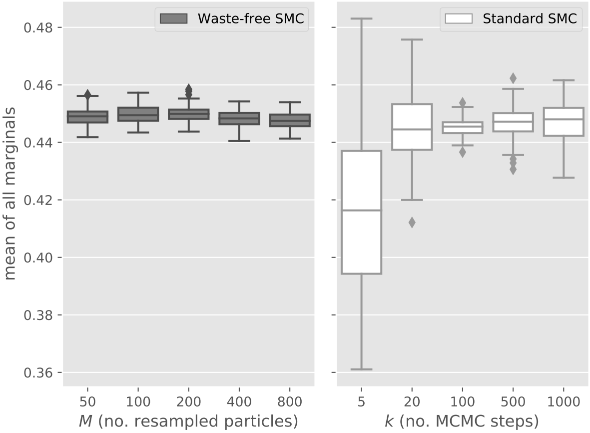
Numerical experiment I

- target: posterior of a logistic regression, sonar dataset ($d = 63$, challenging, see C. and Ridgway, 2017).
- sequence: (adaptive) tempering.
- MCMC steps: random walk Metropolis (adapted/calibrated to particle sample).

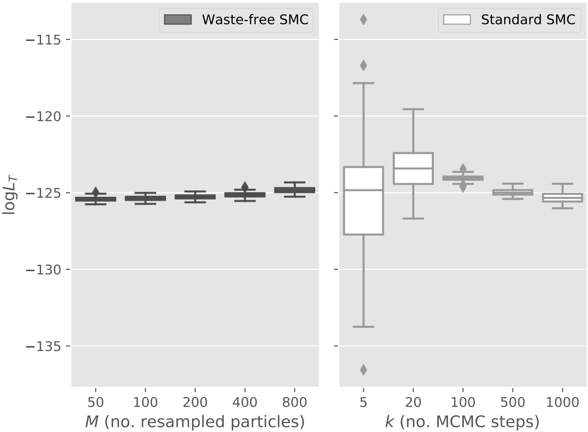
Numerical experiment I

- target: posterior of a logistic regression, sonar dataset ($d = 63$, challenging, see C. and Ridgway, 2017).
- sequence: (adaptive) tempering.
- MCMC steps: random walk Metropolis (adapted/calibrated to particle sample).
- independent runs of standard SMC (varying k, the number of MCMC steps), and waste-free SMC (varying M). Same number of likelihood evaluations: $N = 2 \times 10^5/k$ (standard SMC), and $N = 2 \times 10^5$ for waste-free.

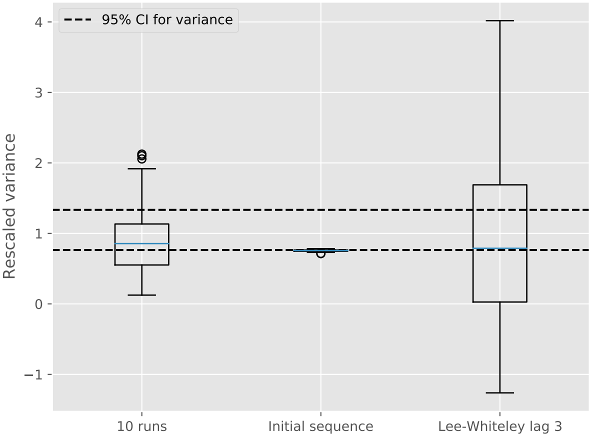
Posterior expectation of the average of the components



Log marginal likelihood



Single-run variance estimators



Numerical experiment II

$$\begin{matrix} 1 & 3 & 2 \\ 2 & 1 & 3 \\ 3 & 2 & 1 \end{matrix}$$

A Latin square of size 3.

- Objective: counting the number $l(d)$ of latin squares of size d.

Numerical experiment II

1	3	2
2	1	3
3	2	1

A Latin square of size 3.

- Objective: counting the number $l(d)$ of latin squares of size d.
- Sequence:

$$\pi_t(\theta) = \frac{1}{L_t}\mu(\theta)\exp\{-\lambda_t V(\theta)\}$$

where $V(\theta) \geq 0$, $= 0$ iff θ is a Latin square. As $\lambda_t \to \infty$, L_t goes to $l(d)$.

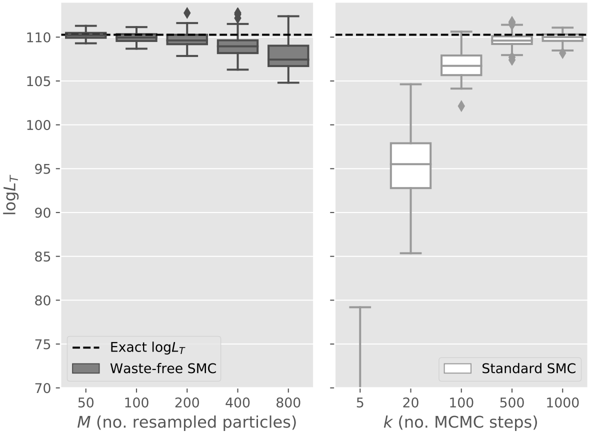
Numerical experiment II

1	3	2
2	1	3
3	2	1

A Latin square of size 3.

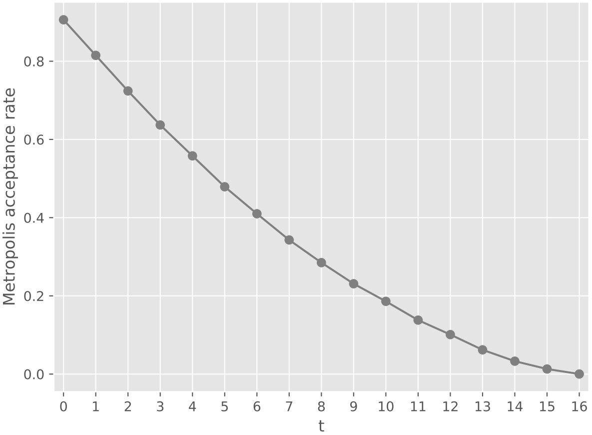
- Objective: counting the number $l(d)$ of latin squares of size d.
- Sequence:
$$\pi_t(\theta) = \frac{1}{L_t}\mu(\theta)\exp\{-\lambda_t V(\theta)\}$$
where $V(\theta) \geq 0$, $= 0$ iff θ is a Latin square. As $\lambda_t \to \infty$, L_t goes to $l(d)$.
- MCMC kernels: Metropolis, swap two entries.

Results ($d = 11$)



$\log L_T$
110
105
100
95
90
85
80
75
70
Exact $\log L_T$
Waste-free SMC
50
100
200
400
800
M (no. resampled particles)
Standard SMC
5
20
100
500
1000
k (no. MCMC steps)

Acceptance rate vs iteration



Numerical experiment III

- Objective: evaluate orthant probability $\mathbb{P}(X \geq 0)$, $X \sim \mathcal{N}(\mu, \Sigma)$ and/or sample from the corresponding truncated Gaussian dist.

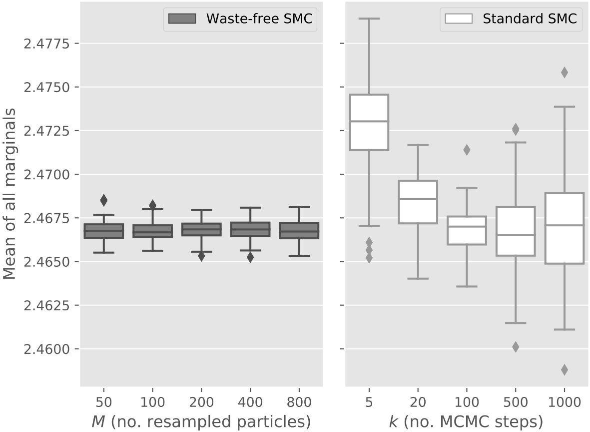
Numerical experiment III

- Objective: evaluate orthant probability $\mathbb{P}(X \geq 0)$, $X \sim \mathcal{N}(\mu, \Sigma)$ and/or sample from the corresponding truncated Gaussian dist.
- Sequence: increasing dimension.

Numerical experiment III

- Objective: evaluate orthant probability $\mathbb{P}(X \geq 0)$, $X \sim \mathcal{N}(\mu, \Sigma)$ and/or sample from the corresponding truncated Gaussian dist.
- Sequence: increasing dimension.
- MCMC steps: Gibbs

Results



Waste-free SMC
Standard SMC
Mean of all marginals
2.4775
2.4750
2.4725
2.4700
2.4675
2.4650
2.4625
2.4600
50
100
200
400
800
M (no. resampled particles)
5
20
100
500
1000
k (no. MCMC steps)

Conclusion: waste-free

- Same advantages as standard SMC:

Conclusion: waste-free

- Same advantages as standard SMC:
  - automatable;

Conclusion: waste-free

- Same advantages as standard SMC:
  - automatable;
  - unbiased estimate of the normalising constant;

Conclusion: waste-free

- Same advantages as standard SMC:
  - automatable;
  - unbiased estimate of the normalising constant;
  - parallelisable (up to M cores)

Conclusion: waste-free

- Same advantages as standard SMC:
  - automatable;
  - unbiased estimate of the normalising constant;
  - parallelisable (up to M cores)
- In addition:

Conclusion: waste-free

- Same advantages as standard SMC:
  - automatable;
  - unbiased estimate of the normalising constant;
  - parallelisable (up to M cores)
- In addition:
  - even more automatable (no need to choose k)

Conclusion: waste-free

- Same advantages as standard SMC:
  - automatable;
  - unbiased estimate of the normalising constant;
  - parallelisable (up to M cores)
- In addition:
  - even more automatable (no need to choose k)
  - single-run variance estimators (**not** based on genealogy)

Conclusion: waste-free

- Same advantages as standard SMC:
  - automatable;
  - unbiased estimate of the normalising constant;
  - parallelisable (up to M cores)
- In addition:
  - even more automatable (no need to choose k)
  - single-run variance estimators (**not** based on genealogy)

Conclusion: waste-free

- Same advantages as standard SMC:
  - automatable;
  - unbiased estimate of the normalising constant;
  - parallelisable (up to M cores)
- In addition:
  - even more automatable (no need to choose k)
  - single-run variance estimators (**not** based on genealogy)

In particular, **tempering SMC** is a very good default strategy if you have a single target distribution.

Implementations

- Python: https://github.com/nchopin/particles

Implementations

- Python: https://github.com/nchopin/particles
- Blackjax

References

- Waste-free SMC (Dau & C., JRSSB, 2021): https://doi.org/10.1111/rssb.12475

References

- Waste-free SMC (Dau & C., JRSSB, 2021): https://doi.org/10.1111/rssb.12475
- Chapter 16 of C. and Papaspiliopoulos (2020):