Anytime Monte Carlo

Lawrence Murray

University of Oxford

Collaborators
Sumeetpal Singh (Cambridge)
Pierre Jacob (Harvard)
Anthony Lee (Warwick)

Motivation

- Typically we fix the number of samples to draw, n, and allow the time taken to draw these, $T(n)$ to be a random variable.
- Instead, we wish to fix the time, t, and allow the number of samples drawn in this time, $N(t)$, to be the random variable.
- **Why?** Real-time deadlines, cloud computing budgets, synchronisation and fault tolerance in a distributed computing environment, fair computational comparison of methods.

Existing work

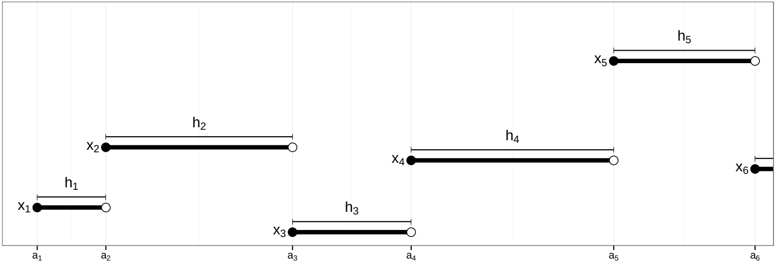
- P. W. Glynn and P. Heidelberger. Bias properties of budget constraint simulations.
  Operations Research, 38(5):801–814, 1990.
- P. W. Glynn and P. Heidelberger. Analysis of parallel replicated simulations under a completion time constraint.
  ACM Transactions on Modeling and Computer Simulations, 1 (1):3–23, 1991.
- B. Paige, F. Wood, A. Doucet, and Y. W. Teh. Asynchronous anytime sequential Monte Carlo.
  In *Advances in Neural Information Processing Systems 27*, pages 3410–3418. 2014.

Framework

- Consider a Markov chain $(X_n)_{n=0}^{\infty}$ with transition kernel $\kappa(x_{n+1} \mid x_n)$ and invariant distribution $\pi(x)$.
- A computer takes some real time H_n to complete the computations necessary to transition from X_n to X_{n+1}.
- H_n is the hold time of X_n, distributed according to $\tau(h_n \mid x_n)$.
- We can write:

$$\kappa(x_{n+1} \mid x_n) = \int \kappa(x_{n+1} \mid x_n, h_n)\tau(h_n \mid x_n)\, \mathrm{d}h_n.$$

Framework



Framework

- Intercept the running process at some time t.
- The state at that time, $X(t)$, is not—in general—distributed according to $\pi(x)$. It is length-biased with respect to compute time.
- For t sufficiently large, $X(t)$ is distributed according to $\alpha(x)$, with

$$\alpha(x) \propto \pi(x)\mathbb{E}_{\tau}[H \mid x].$$

- We refer to $\alpha(x)$ as the *anytime distribution*.

Sketch of Proofs

- Construct a real-time Markov process $(X, L)(t)$, with $L \in \mathbb{R}$, $L \geq 0$, the lag time since the last jump.

- Assume $\mathbb{E}[H \mid x]$ is finite and $H \geq \epsilon$.

- Define:

$$
\begin{aligned}
x &:= x(t) \\
l &:= l(t) \\
x_+ &:= x(t+\epsilon) \\
l_+ &:= l(t+\epsilon).
\end{aligned}
$$

Sketch of Proofs

The transition kernel is:

$$\lambda(x_+, l_+ \mid x, l) = \gamma(x)\lambda_1(x_+, l_+ \mid x, l) + (1 - \gamma(x))\,\lambda_0(x_+, l_+ \mid x, l),$$

where

$$\gamma(x) = \frac{\mathbb{P}_\tau[l < H \leq l + \epsilon \mid x]}{\mathbb{P}_\tau[H > l \mid x]}$$

is the probability of a jump occurring in the time interval $(t, t+\epsilon]$,

$$\lambda_1(x_+, l_+ \mid x, l) = \kappa(x_+ \mid x, H = l+\epsilon-l_+)\frac{\tau(H = l + \epsilon - l_+ \mid x)\mathbb{I}_{[0,\epsilon)}(l_+)}{\mathbb{P}_\tau[l < H \leq l + \epsilon \mid x]}$$

the transition kernel if one does, and

$$\lambda_0(x_+, l_+ \mid x, l) := \delta_x(x_+)\delta_{l+\epsilon}(l_+)$$

the transition kernel if one does not. As $H > \epsilon$, at most one jump can occur.

Sketch of Proofs

- We now have a Markov chain to study.
- The invariant distribution is

$$\alpha(x, l) = \frac{\mathbb{P}_\tau[H > l \mid x]}{\mathbb{E}_\tau[H]}\pi(x),$$

  with marginal

$$\alpha(x) \propto \pi(x)\mathbb{E}_\tau[H \mid x],$$

  i.e. the anytime distribution previously identified.

- The original Markov chain is recovered by recognising $\alpha(x \mid l = 0) = \pi(x)$.

Anytime Monte Carlo

- We want the anytime distribution to instead be $\pi(x)$.

Anytime Monte Carlo

- We want the anytime distribution to instead be $\pi(x)$.

- A sufficient condition to establish this is for the expected hold time to be independent of X, i.e.

$$\mathbb{E}_\tau[H \mid x] = \mathbb{E}_\tau[H]$$

so that $\alpha(x) = \pi(x)$.

Anytime Monte Carlo

- We want the anytime distribution to instead be $\pi(x)$.
- A sufficient condition to establish this is for the expected hold time to be independent of X, i.e.

$$\mathbb{E}_\tau[H \mid x] = \mathbb{E}_\tau[H]$$

  so that $\alpha(x) = \pi(x)$.
- For iid sampling, this is trivial. Have $\kappa(x_{n+1} \mid x_n) = \pi(x_{n+1})$ and $\tau(h_n \mid x_n) = \tau(h_n)$.

Anytime Monte Carlo

- For non-iid sampling, consider modifying the transition of the Markov chain to

$$X_n \sim \kappa(dx_n \mid x_{n-2}).$$

- This interleaves two independent Markov chains, where the hold times of each chain depend only on the states of the other chain.

- Generalise this to $K \geq 2$ number of chains. Using a **single processor**, repeatedly choose one at random (or systematically) and advance it forward one step.

Anytime Monte Carlo

- While for one chain we have an anytime distribution of:

  $$\alpha(x) \propto \pi(x)\mathbb{E}_{\tau}[H \mid x],$$

  for $K \geq 2$ chains, we have an anytime distribution of:

  $$\beta(x^{1:K}) = \alpha(x^k) \prod_{i=1, i \neq k}^{K} \pi(x^i),$$

  where k is the index of the currently advancing chain.

- That is, only the kth chain is length-biased, and can simply be discarded. The remaining $K - 1$ states are distributed according to $\pi(x)$.

Toy Case Study

- Consider the model

$$\begin{aligned} X &\sim \text{Gamma}(k, \theta) \\ H \mid x &\sim \text{Gamma}(x^p/\theta, \theta), \end{aligned}$$

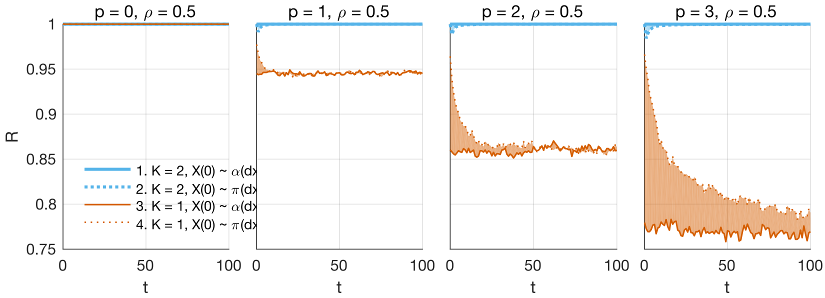
  with shape parameter k, scale parameter θ, and polynomial degree p.

- The two distributions correspond to the target distribution $\pi(x)$ and hold-time distribution $\tau(h \mid x)$, respectively, yielding an anytime distribution $\alpha(x)$ of $\text{Gamma}(k + p, \theta)$.

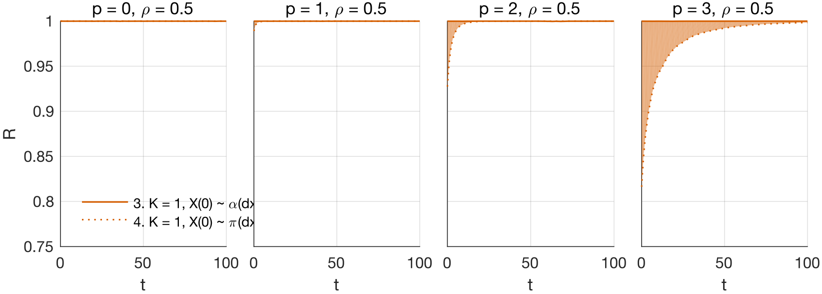
Toy Case Study

- 10000 Markov chains targeting $\pi(x)$ for 100 units of virtual time.
- At each virtual time, take the state of all chains and evaluate the probability plot (Q-Q plot) correlation coefficient comparing the empirical distribution of these samples with $\pi(x)$.
- Compare four sampling regimes:
  1. $K = 2$ chains, with $X^{1:K}(0) \sim \alpha(\mathrm{d}x^{1:K})$ and lag,
  2. $K = 2$ chains, with $X^{1:K}(0) \sim \pi(\mathrm{d}x^{1:K})$ and no lag,
  3. $K = 1$ chain, with $X(0) \sim \alpha(dx)$ and lag,
  4. $K = 1$ chain, with $X(0) \sim \pi(dx)$ and no lag.

Toy Case Study



Toy Case Study



Sequential Monte Carlo Case Study

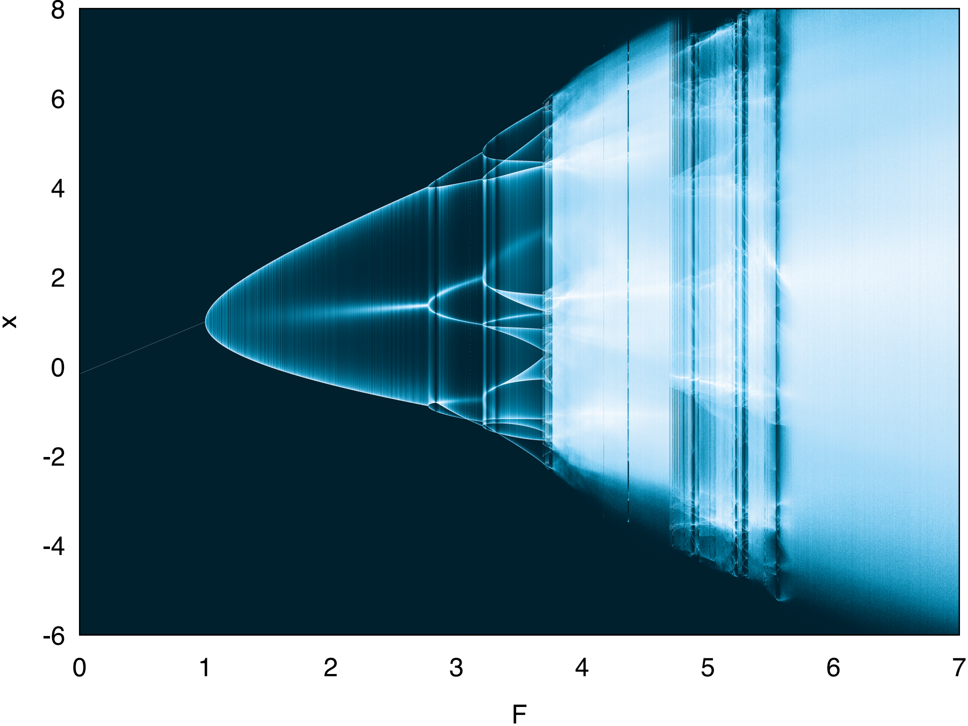
- D-dimensional Lorenz '96 model given by the equations:

$$\frac{dx_d}{dt} = x_{d-1}\left(x_{d+1} - x_{d-2}\right) - x_d + F,$$

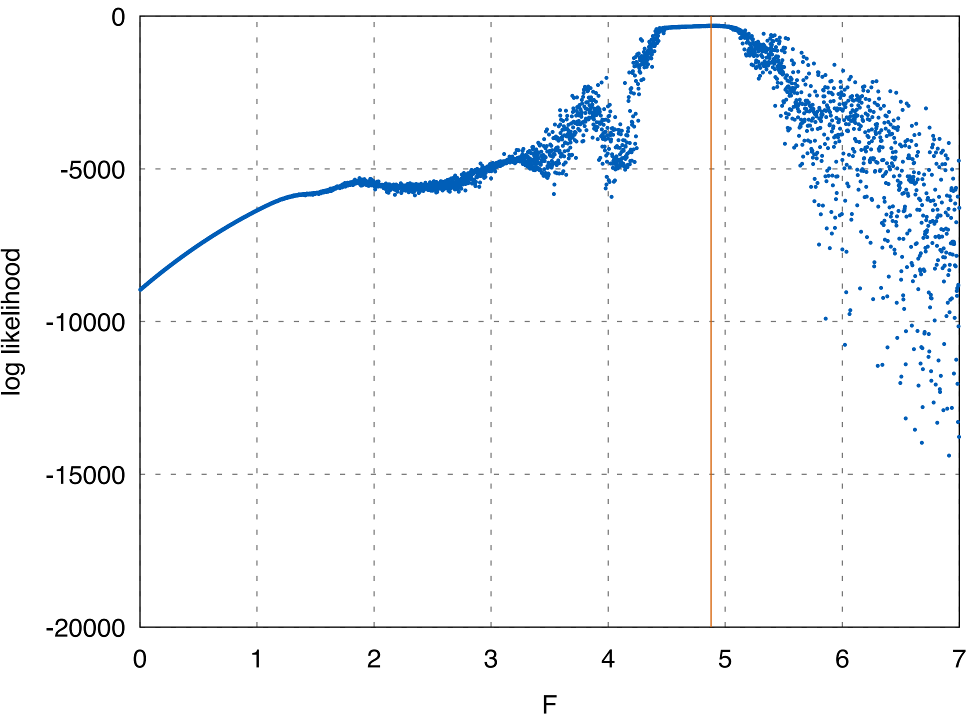
  where subscripts are interpreted cyclically, so that $x_{d-D} \equiv x_d \equiv x_{d+D}$, and F is a parameter.

- Use an 8-dimensional model here, discretised with an adaptive time-step Runge—Kutta across intervals of 0.05. Gaussian noise of variance 10^{-4}

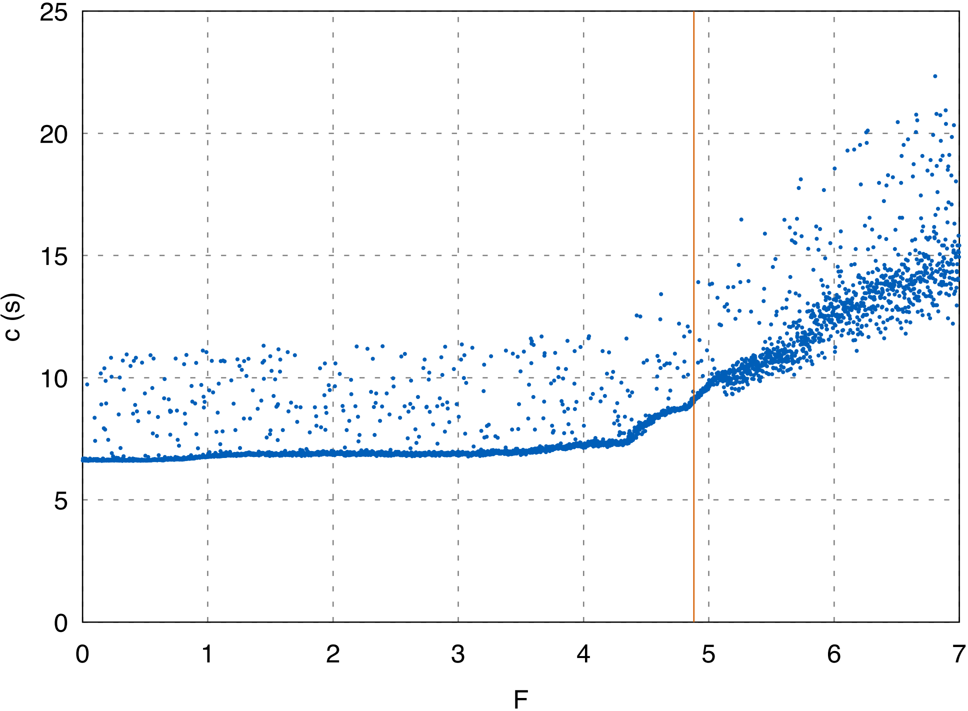
- Prior distribution $F \sim \mathcal{U}([0,7])$.



8
6
4
2
0
-2
-4
-6
x
0
1
2
3
4
5
6
7
F



0
-5000
-10000
-15000
-20000
log likelihood
0
1
2
3
4
5
6
7
F



25
20
15
10
5
0
c (s)
0
1
2
3
4
5
6
7
F

Sequential Monte Carlo (SMC)

1. For $m = 0$, draw N particles (samples) $\theta^{1:N}$ from $\pi_0(\theta)$.

2. For $m = 1, \ldots, M$

   2.1 **Weight**: assign θ^n a weight of

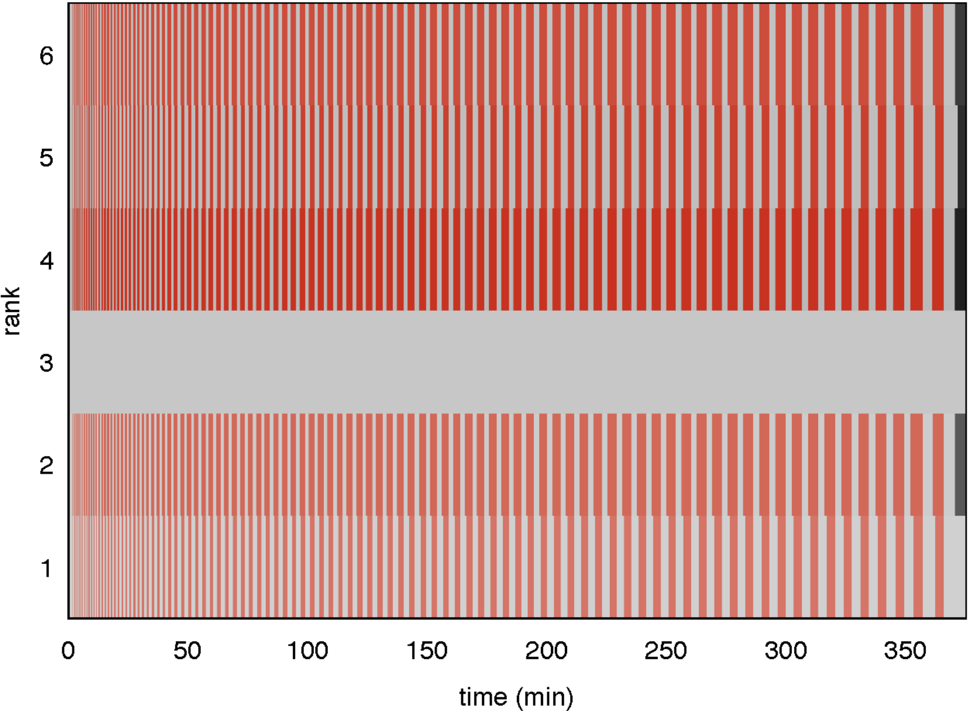
   $$\begin{aligned} w^n &= \pi_m(\theta^n)/\pi_{m-1}(\theta^n) \\ &\propto p(y_m \mid \theta^n, y_{1:m-1}) \end{aligned}$$

   2.2 **Interact**: resample all particles according to weights and adapt a new kernel $\kappa_m(\theta' \mid \theta)$ that is invariant to $\pi_m(\theta)$.

   2.3 **Move**: apply the kernel $\kappa_m(\theta' \mid \theta)$ to each particle some number of times.

SMC^2

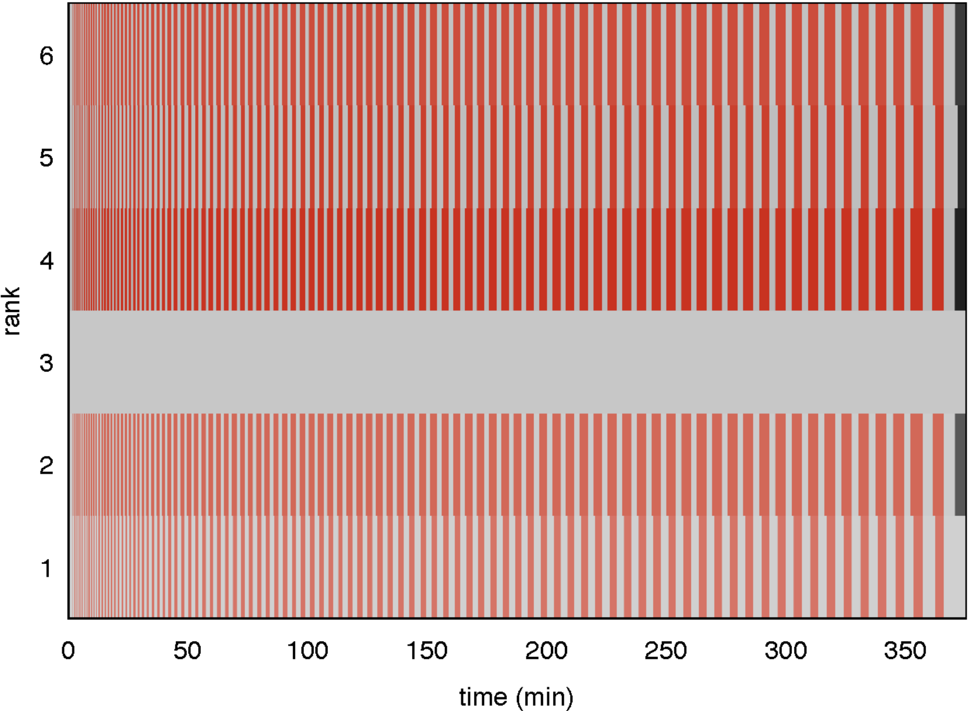
- Run SMC^2 using LibBi (`www.libbi.org`) on a local compute server.
- 6 GPUs each with 1536 cores.
  About 10,000 way parallelism.
- 2^8 θ-particles each with 2^{20} x-particles.
  About 250,000,000 particles.

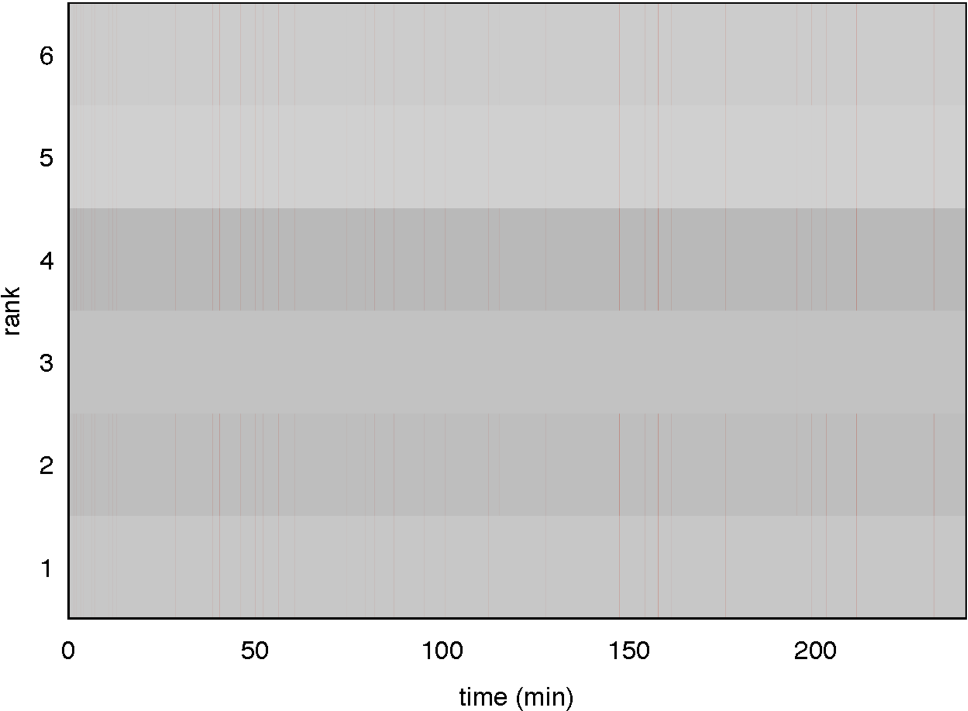


rank
6
5
4
3
2
1
0
50
100
150
200
250
300
350
time (min)

Anytime SMC^2

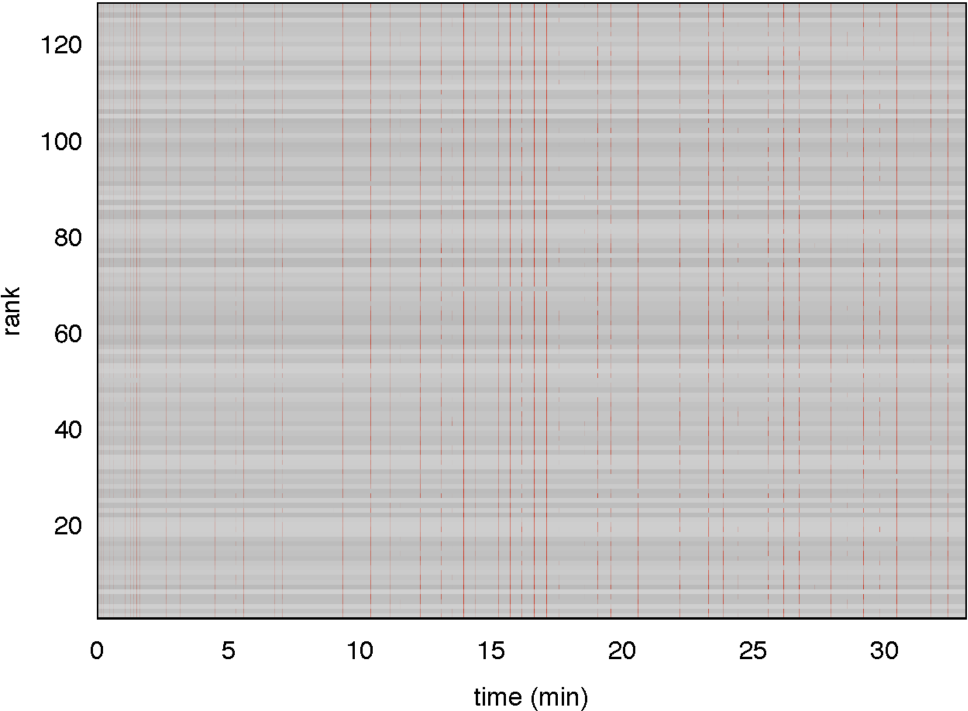
- Set a deadline to finish the *m*th move step.
- During the **move** step, repeatedly choose a θ-particle at random and apply the kernel.
- When time is up, discard the θ-particle currently selected, and proceed to the next **weight** and **interact** steps.


rank
6
5
4
3
2
1
0
50
100
150
200
250
300
350
time (min)

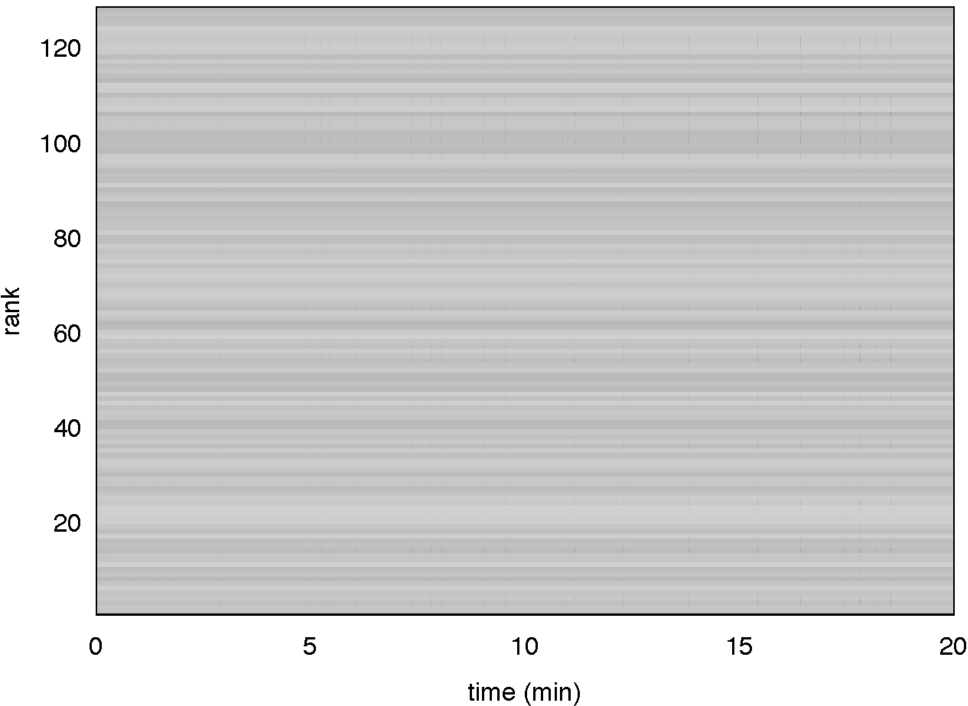

6
5
4
3
2
1
rank
0
50
100
150
200
time (min)

Cloud Computing

- Run SMC^2 using LibBi (`www.libbi.org`) on Amazon EC2.
- 128 GPU instances each with 1536 cores.
  About 200,000 way parallelism.
- 2^{12} θ-particles each with 2^{20} x-particles.
  About 4,000,000,000 particles.



120
100
80
60
40
20
rank
0
5
10
15
20
25
30
time (min)



120
100
80
60
40
20
rank
0
5
10
15
20
time (min)

Summary

- The anytime framework allows Monte Carlo algorithms to be configured in terms of real time rather than number of samples.
- Can be used to satisfy real-time deadlines and budget constraints, perhaps provide fault tolerance.
- Because, for non-iid sampling, it requires multiple states, it is particularly useful within SMC, which already has multiple states (particles).
- In a distributed computing setting, mitigates problems associated with synchronisation that can otherwise limit scalability.

References

P. W. Glynn and P. Heidelberger. Bias properties of budget constraint simulations. *Operations Research*, 38(5):801–814, 1990.

P. W. Glynn and P. Heidelberger. Analysis of parallel replicated simulations under a completion time constraint. *ACM Transactions on Modeling and Computer Simulations*, 1(1):3–23, 1991.

L. M. Murray. Bayesian state-space modelling on high-performance hardware using LibBi. *Journal of Statistical Software*, 67(10):1–36, 2015. ISSN 1548-7660. doi: 10.18637/jss.v067.i10. URL `http://www.jstatsoft.org/index.php/jss/article/view/v067i10`.

B. Paige, F. Wood, A. Doucet, and Y. W. Teh. Asynchronous anytime sequential Monte Carlo. In *Advances in Neural Information Processing Systems 27*, pages 3410–3418. 2014.

LibBi software, `www.libbi.org`.