Centre for Research in Statistical Methodology

`http://www2.warwick.ac.uk/fac/sci/statistics/crism/`

- Conferences and workshops
- Research Fellow positions (will advertise a further position in Autumn).
- PhD studentships
- Academic visitor programme.

Retrospective sampling and the Bernoulli factory

Gareth Roberts
Statistics, University of Warwick
joint work with
Krzysztof Latuszynski Ioannis Kosmidis Omiros Papaspiliopoulos

Retrospective sampling

Motivation for Bernoulli factory

The Bernoulli factory - general results and previous approaches

Reverse time martingale approach to sampling

Application to the Bernoulli Factory problem

Motivation

- Interested in simulating exactly from high and infinite dimensional problems

Motivation

- Interested in simulating exactly from high and infinite dimensional problems
- Motivation from problems in statistics (generally inference for stochastic processes and analysis of algorithms: MCMC and sequential Monte Carlo)

Motivation

- Interested in simulating exactly from high and infinite dimensional problems
- Motivation from problems in statistics (generally inference for stochastic processes and analysis of algorithms: MCMC and sequential Monte Carlo)
- Exact simulation of diffusions - possible for (pretty much) all one-dimensional diffusions - generally more efficient than discretisation methods. Similar story for other infinite dimensional models (for instance distributions linked to Dirichlet processes).

Motivation

- Interested in simulating exactly from high and infinite dimensional problems
- Motivation from problems in statistics (generally inference for stochastic processes and analysis of algorithms: MCMC and sequential Monte Carlo)
- Exact simulation of diffusions - possible for (pretty much) all one-dimensional diffusions - generally more efficient than discretisation methods. Similar story for other infinite dimensional models (for instance distributions linked to Dirichlet processes).
- Use methods based on retrospective simulation

What is retrospective sampling?

It is an attempt to take advantage of the redundancy inherent in modern simulation algorithms (particularly MCMC, rejection sampling) by subverting the traditional order of algorithm steps.

It is (in principle) very simple!

Retrospective sampling is most powerful in infinite dimensional contexts, where its natural competitors are approximate and computationally expensive. In contrast, restrospective methods are often computationally inexpensive and "exact".

Restrospective sampling has natural allies in the simulation game, for example catalytic perfect simulation and non-centering

Football Quiz

Who is the famous Polish goalkeeper who denied England in 1973 and went on to become a World Cup hero?

1. Lech Walesa
2. Jan Tomaszewski
3. Frederic Chopin

N people enter a competition to win a prize, entering their answer on a postcard. The winner is drawn uniformly from those who get the question right (ie most of them). Suppose a proportion $p > 0.5$ get it right.

Algorithm 1

1. Mark each of the N entries, placing the correct postcards into a bucket.
2. Shake the bucket and then pick out one postcard, declaring its author the winner.

Cost of this procedure, $O(N)$.

Algorithm 1

1. Mark each of the N entries, placing the correct postcards into a bucket.
2. Shake the bucket and then pick out one postcard, declaring its author the winner.

Cost of this procedure, $O(N)$.

Algorithm 2

1. Throw all the postcards into the bucket without marking them
2. Draw postcards until a winner is found

Cost of this procedure, $O(p^{-1})$.

Rejection sampling

Let f be a density of interest, and g be a density from which we can simulate. f/g bounded by K say.

1. Sample X from g.
2. Compute $p(X) = f(X)/(Kg(X))$.
3. Simulate $U \sim U(0, 1)$.
4. Accept X if $p(X) > U$. Otherwise return to 1.

Blue steps are often unnecessary!

Retrospective rejection sampling

1. Sample $V \sim U(0,1)$.
2. Identify a function $h(V,X)$ and a set $A(V)$ such that

$$\mathbf{P}_V\{h(V,X) \in A(V)\} = p(X)$$

3. Simulate $h(X,V)$.
4. If $h(X,V) \in A(V)$ the accept. Otherwise return to 1.
5. Fill in missing bits of X from distribution of $X|h(X,V)$ as required.

Simulating from unnormalised probabilities

We have $p_1, p_2, \ldots$ is a sequence of positive numbers with $p_i \leq q_i$ and $\sum_{i=j+1}^{\infty} q_i = G(j) < \infty$.
We would like to simulate from the discrete distribution with probabilities proportional to $\{p_i\}$.
Why not use the inverse CDF method?

1. Calculate $s = \sum_{i=1}^{\infty} p_i$
2. Simulate $U \sim U(0, 1)$.
3. Set $X = \inf\{j;\ \sum_{i=1}^{j} p_j/s \geq U\}$.

Retrospective inverse CDF method

$$s_j^- = \sum_{i=1}^{j} p_i$$

$$s_j^+ = \sum_{i=1}^{j} p_i + G(j)$$

Clearly

$$s_j^- \leq s_{j+1}^- \leq s \leq s_{j+1}^+ \leq s_{j+1}^+$$

$$P_i^{+j} = \sum_{k=1}^{j} \frac{p_k}{s_j^-}$$

$$P_i^{-j} = \sum_{k=1}^{j} \frac{p_k}{s_j^+}$$

$$X^{+j}(U) = \inf\{j;\ P_i^{+j} \geq U\}$$

$$X^{-j}(U) = \inf\{j;\ P_i^{-j} \geq U\}$$

1. Simulate $U \sim U(0,1)$.
2. Calculate $X^{-j}(U)$ and $X^{+j}(U)$, $j = 1, 2, \ldots$ until $X^{-j}(U) = X^{+j}(U)$. Set X to be this common value.

Generic description of the Bernoulli factory problem

- Let $p \in (0,1)$ be **unknown**.
- Given a black box that samples $p-$coins
- Can we construct a black box that samples $f(p)$ coins for known f? For example $f(p) = \min(1, 2p)$

Some history

(see for example Peres, 1992)
von Neumann posed and solved the problem: $f(p) = 1/2$ (how to make a biased coin fair)

Some history

(see for example Peres, 1992)
von Neumann posed and solved the problem: $f(p) = 1/2$ (how to make a biased coin fair)

1. set $n = 1$;
2. sample X_n, X_{n+1}
3. if $(X_n, X_{n+1}) = (0, 1)$ output 1 and STOP
4. if $(X_n, X_{n+1}) = (1, 0)$ output 0 and STOP
5. set $n := n + 2$ and GOTO 2.

The Bernoulli Factory problem

for **known** f and **unknown** p, how to generate an $f(p)-$coin?

von Neumann: $f(p) = 1/2$

Asmussen conjectured $f(p) = 2p$, but it turned out difficult

Exact simulation of diffusions as Bernoulli factory

This is the description of EA closest in spirit to Beskos and R (2005)

Simulate X_T at time $T > 0$ from:

$$\mathrm{d}X_t = \alpha(X_t)\,\mathrm{d}t + \mathrm{d}W_t, \quad X_0 = x \in \mathbf{R},\ t \in [0, T] \tag{1}$$

driven by the Brownian motion $\{W_t\,;\, 0 \leq t \leq T\}$

$W^x = \{W_t^x\,;\, 0 \leq t \leq T\}$ the Brownian motion started at $x \in \mathbf{R}$, and by $W^{x,u} = \{W_t^{x,u}\,;\, 0 \leq t \leq T\}$ the Brownian bridge started at x finishing at u at time T.

$W^x = \{W_t^x\,;\, 0 \leq t \leq T\}$ the Brownian motion started at $x \in \mathbf{R}$, and by $W^{x,u} = \{W_t^{x,u}\,;\, 0 \leq t \leq T\}$ the Brownian bridge started at x finishing at u at time T.

We need the following conditions:

1. The drift function α is differentiable.

$W^x = \{W_t^x\,;\, 0 \le t \le T\}$ the Brownian motion started at $x \in \mathbf{R}$, and by $W^{x,u} = \{W_t^{x,u}\,;\, 0 \le t \le T\}$ the Brownian bridge started at x finishing at u at time T.

We need the following conditions:

1. The drift function α is differentiable.
2. The function $h(u) = \exp\{A(u) - (u-x)^2/2T\}$, $u \in \mathbf{R}$, for $A(u) = \int_0^u \alpha(y)\mathrm{d}y$, is integrable.

$W^x = \{W_t^x\,;\, 0 \le t \le T\}$ the Brownian motion started at $x \in \mathbf{R}$, and by $W^{x,u} = \{W_t^{x,u}\,;\, 0 \le t \le T\}$ the Brownian bridge started at x finishing at u at time T.

We need the following conditions:

1. The drift function α is differentiable.
2. The function $h(u) = \exp\{A(u) - (u-x)^2/2T\}$, $u \in \mathbf{R}$, for $A(u) = \int_0^u \alpha(y)\mathrm{d}y$, is integrable.
3. The function $(\alpha^2 + \alpha')/2$ is bounded below by $\ell > -\infty$, and above by $r + \ell < \infty$.

$$\phi(u) = \frac{1}{r}[(\alpha^2 + \alpha')/2 - \ell] \in [0,1]\,, \tag{2}$$

- $\mathbb{Q}$ be the probability measure induced by the solution X of (1)
- $\mathbb{W}$ the corresponding probability measure for W^x, and
- $\mathbb{Z}$ be the probability measure defined as the following simple change of measure from $\mathbb{W}$: $\mathrm{d}\mathbb{W}/\mathrm{d}\mathbb{Z}(\omega) \propto \exp\{-A(B_T)\}$.

- $\mathbb{Q}$ be the probability measure induced by the solution X of (1)
- $\mathbb{W}$ the corresponding probability measure for W^x, and
- $\mathbb{Z}$ be the probability measure defined as the following simple change of measure from $\mathbb{W}$: $\mathrm{d}\mathbb{W}/\mathrm{d}\mathbb{Z}(\omega) \propto \exp\{-A(B_T)\}$.

$\mathbb{Z}$ has similar dynamics to the Brownian motion, except that the distribution of the marginal distribution at time T (with density, say, h) which is biased according to A.

- $\mathbb{Q}$ be the probability measure induced by the solution X of (1)
- $\mathbb{W}$ the corresponding probability measure for W^x, and
- $\mathbb{Z}$ be the probability measure defined as the following simple change of measure from $\mathbb{W}$: $\mathrm{d}\mathbb{W}/\mathrm{d}\mathbb{Z}(\omega) \propto \exp\{-A(B_T)\}$.

$\mathbb{Z}$ has similar dynamics to the Brownian motion, except that the distribution of the marginal distribution at time T (with density, say, h) which is biased according to A. Then,

$$\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{Z}}(\omega) \propto \exp\left\{-rT\int_0^T T^{-1}\phi(B_t)\mathrm{d}t\right\} \leq 1 \qquad \mathbb{Z}-\mathrm{a.s.} \qquad (3)$$

EA using Bernoulli factory

1. simulate $u \sim h$
2. generate a C_s coin where $s := e^{-rTJ}$, and $J := \int_0^T T^{-1}\phi(W_t^{x,u})\mathrm{d}t$;
3. If $C_s = 1$ output u and STOP;
4. If $C_s = 0$ GOTO 1.

Exploiting the Markov property, we can assume from now on that $rT < 1$.

The challenging part of the algorithm is Step 2, since exact computation of J is impossible due to the integration over a Brownian bridge path.

On the other hand, it is easy to generate J-coins by retrospective sampling:

$$C_J = \mathbb{I}(\psi < \phi(W_\chi^{x,u})), \quad \psi \sim U(0,1), \quad \chi \sim U(0,T)$$

independent of the Brownian bridge $W^{x,u}$ and of each other.

The challenging part of the algorithm is Step 2, since exact computation of J is impossible due to the integration over a Brownian bridge path.

On the other hand, it is easy to generate J-coins by retrospective sampling:

$$C_J = \mathbb{I}(\psi < \phi(W_\chi^{x,u})), \quad \psi \sim U(0,1), \quad \chi \sim U(0,T)$$

independent of the Brownian bridge $W^{x,u}$ and of each other.

How can we get C_s coins from C_J ones?

Keane and O'Brien - existence result

- Keane and O'Brien (1994):

$$\text{Let} \quad p \in \mathcal{P} \subseteq (0,1) \to [0,1]$$

then it is possible to simulate an $f(p)-$coin $\iff$

- f is constant
- f is continuous and for some $n \in \mathbb{N}$ and all $p \in \mathcal{P}$ satisfies

$$\min\left\{f(p), 1-f(p)\right\} \geq \min\left\{p, 1-p\right\}^n$$

Note that the result rules out $\min\{1, 2p\}$, but not $\min\{1-2\epsilon, 2p\}$

Keane and O'Brien - existence result

- Keane and O'Brien (1994):

$$\text{Let} \quad p \in \mathcal{P} \subseteq (0,1) \to [0,1]$$

  then it is possible to simulate an $f(p)-$coin $\iff$
  - f is constant
  - f is continuous and for some $n \in \mathbb{N}$ and all $p \in \mathcal{P}$ satisfies

$$\min\left\{f(p), 1-f(p)\right\} \geq \min\left\{p, 1-p\right\}^n$$

- however their proof is **not constructive**

Note that the result rules out $\min\{1, 2p\}$, but not $\min\{1-2\epsilon, 2p\}$

Nacu-Peres (2005) Theorem - Bernstein polynomial approach

- There exists an algorithm which simulates $f \iff$ there exist polynomials

$$g_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} a(n,k) x^k y^{n-k}, \quad h_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} b(n,k) x^k y^{n-k}$$

with following properties

Nacu-Peres (2005) Theorem - Bernstein polynomial approach

- There exists an algorithm which simulates $f \iff$ there exist polynomials

$$g_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} a(n,k) x^k y^{n-k}, \quad h_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} b(n,k) x^k y^{n-k}$$

with following properties

- $0 \leq a(n,k) \leq b(n,k) \leq 1$

Nacu-Peres (2005) Theorem - Bernstein polynomial approach

- There exists an algorithm which simulates f $\iff$ there exist polynomials

$$g_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} a(n,k) x^k y^{n-k}, \quad h_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} b(n,k) x^k y^{n-k}$$

with following properties

- $0 \leq a(n,k) \leq b(n,k) \leq 1$
- $\binom{n}{k} a(n,k)$ and $\binom{n}{k} b(n,k)$ are integers

Nacu-Peres (2005) Theorem - Bernstein polynomial approach

- There exists an algorithm which simulates $f \iff$ there exist polynomials

$$g_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} a(n,k) x^k y^{n-k}, \quad h_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} b(n,k) x^k y^{n-k}$$

with following properties

- $0 \leq a(n,k) \leq b(n,k) \leq 1$
- $\binom{n}{k} a(n,k)$ and $\binom{n}{k} b(n,k)$ are integers
- $\lim_{n\to\infty} g_n(p,1-p) = f(p) = \lim_{n\to\infty} h_n(p,1-p)$

Nacu-Peres (2005) Theorem - Bernstein polynomial approach

- There exists an algorithm which simulates $f \iff$ there exist polynomials

$$g_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} a(n,k) x^k y^{n-k}, \quad h_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} b(n,k) x^k y^{n-k}$$

with following properties

- $0 \leq a(n,k) \leq b(n,k) \leq 1$
- $\binom{n}{k} a(n,k)$ and $\binom{n}{k} b(n,k)$ are integers
- $\lim_{n\to\infty} g_n(p,1-p) = f(p) = \lim_{n\to\infty} h_n(p,1-p)$
- for all $m < n$

$$(x+y)^{n-m} g_m(x,y) \preceq g_n(x,y) \quad \text{and} \quad (x+y)^{n-m} h_m(x,y) \succeq h_n(x,y)$$

Nacu-Peres (2005) Theorem - Bernstein polynomial approach

- There exists an algorithm which simulates $f \iff$ there exist polynomials

$$g_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} a(n,k) x^k y^{n-k}, \quad h_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} b(n,k) x^k y^{n-k}$$

with following properties

- $0 \leq a(n,k) \leq b(n,k) \leq 1$
- $\binom{n}{k} a(n,k)$ and $\binom{n}{k} b(n,k)$ are integers
- $\lim_{n\to\infty} g_n(p,1-p) = f(p) = \lim_{n\to\infty} h_n(p,1-p)$
- for all $m < n$

$$(x+y)^{n-m} g_m(x,y) \preceq g_n(x,y) \quad \text{and} \quad (x+y)^{n-m} h_m(x,y) \succeq h_n(x,y)$$

- Nacu & Peres provide coefficients for $f(p) = \min\{2p, 1-2\varepsilon\}$ explicitly.

Nacu-Peres (2005) Theorem - Bernstein polynomial approach

- There exists an algorithm which simulates f $\iff$ there exist polynomials

$$g_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} a(n,k) x^k y^{n-k}, \quad h_n(x,y) = \sum_{k=0}^{n} \binom{n}{k} b(n,k) x^k y^{n-k}$$

with following properties

- $0 \leq a(n,k) \leq b(n,k) \leq 1$
- $\binom{n}{k} a(n,k)$ and $\binom{n}{k} b(n,k)$ are integers
- $\lim_{n\to\infty} g_n(p,1-p) = f(p) = \lim_{n\to\infty} h_n(p,1-p)$
- for all $m < n$

$$(x+y)^{n-m} g_m(x,y) \preceq g_n(x,y) \quad \text{and} \quad (x+y)^{n-m} h_m(x,y) \succeq h_n(x,y)$$

- Nacu & Peres provide coefficients for $f(p) = \min\{2p, 1-2\varepsilon\}$ explicitly.
- Given an algorithm for $f(p) = \min\{2p, 1-2\varepsilon\}$ Nacu & Peres develop a calculus that collapses every real analytic g to nesting the algorithm for f and simulating g.

too nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n - subsets of all 01 strings of length n

too nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n - subsets of all 01 strings of length n
- the cardinalities of A_n and B_n are precisely $\binom{n}{k} a(n,k)$ and $\binom{n}{k} b(n,k)$

too nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n - subsets of all 01 strings of length n
- the cardinalities of A_n and B_n are precisely $\binom{n}{k} a(n,k)$ and $\binom{n}{k} b(n,k)$
- the upper polynomial approximation is converging slowly to f

too nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n - subsets of all 01 strings of length n
- the cardinalities of A_n and B_n are precisely $\binom{n}{k} a(n,k)$ and $\binom{n}{k} b(n,k)$
- the upper polynomial approximation is converging slowly to f
- length of 01 strings is $2^{15} = 32768$ and above, e.g. $2^{25} = 16777216$

too nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n - subsets of all 01 strings of length n
- the cardinalities of A_n and B_n are precisely $\binom{n}{k} a(n,k)$ and $\binom{n}{k} b(n,k)$
- the upper polynomial approximation is converging slowly to f
- length of 01 strings is $2^{15} = 32768$ and above, e.g. $2^{25} = 16777216$
- one has to deal **efficiently** with the set of $2^{2^{25}}$ strings, of length 2^{25} each.

too nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n - subsets of all 01 strings of length n
- the cardinalities of A_n and B_n are precisely $\binom{n}{k} a(n,k)$ and $\binom{n}{k} b(n,k)$
- the upper polynomial approximation is converging slowly to f
- length of 01 strings is $2^{15} = 32768$ and above, e.g. $2^{25} = 16777216$
- one has to deal **efficiently** with the set of $2^{2^{25}}$ strings, of length 2^{25} each.
- we shall develop a reverse time martingale approach to the problem

too nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n - subsets of all 01 strings of length n
- the cardinalities of A_n and B_n are precisely $\binom{n}{k} a(n,k)$ and $\binom{n}{k} b(n,k)$
- the upper polynomial approximation is converging slowly to f
- length of 01 strings is $2^{15} = 32768$ and above, e.g. $2^{25} = 16777216$
- one has to deal **efficiently** with the set of $2^{2^{25}}$ strings, of length 2^{25} each.
- we shall develop a reverse time martingale approach to the problem
- we will construct reverse time super- and submartingales that perform a **random walk** on the Nacu-Peres polynomial coefficients $a(n,k)$, $b(n,k)$ and result in a black box that has **algorithmic cost linear** in the number of original $p-$coins

Before giving the most general algorithm, let us think gradually how to simulate events of unknown probability **constructively**

Before giving the most general algorithm, let us think gradually how to simulate events of unknown probability **constructively**

We begin with some more **retrospective simulation** ideas.

Algorithm 1 - randomisation

- **Lemma:** Sampling events of probability $s \in [0,1]$ is equivalent to constructing an unbiased estimator of s taking values in $[0,1]$ with probability 1.

Algorithm 1 - randomisation

- **Lemma:** Sampling events of probability $s \in [0,1]$ is equivalent to constructing an unbiased estimator of s taking values in $[0,1]$ with probability 1.
- **Proof:** Let $\hat{S}$, s.t. $\mathbb{E}\hat{S} = s$ and $\mathbb{P}(\hat{S} \in [0,1]) = 1$ be the estimator. Then draw $G_0 \sim U(0,1)$, obtain $\hat{S}$ and define a coin $C_s := \mathbb{I}\{G_0 \leq \hat{S}\}$.

$$\mathbb{P}(C_s = 1) = \mathbb{E}\,\mathbb{I}(G_0 \leq \hat{S}) = \mathbb{E}\left(\mathbb{E}\left(\mathbb{I}(G_0 \leq \hat{s}) \mid \hat{S} = \hat{s}\right)\right) = \mathbb{E}\hat{S} = s.$$

The converse is straightforward since an $s-$coin is an unbiased estimator of s with values in $[0,1]$.

Algorithm 1 - randomisation

- **Lemma:** Sampling events of probability $s \in [0,1]$ is equivalent to constructing an unbiased estimator of s taking values in $[0,1]$ with probability 1.
- **Proof:** Let $\hat{S}$, s.t. $\mathbb{E}\hat{S} = s$ and $\mathbb{P}(\hat{S} \in [0,1]) = 1$ be the estimator. Then draw $G_0 \sim U(0,1)$, obtain $\hat{S}$ and define a coin $C_s := \mathbb{I}\{G_0 \leq \hat{S}\}$.

$$\mathbb{P}(C_s = 1) = \mathbb{E}\,\mathbb{I}(G_0 \leq \hat{S}) = \mathbb{E}\left(\mathbb{E}\left(\mathbb{I}(G_0 \leq \hat{s}) \mid \hat{S} = \hat{s}\right)\right) = \mathbb{E}\hat{S} = s.$$

  The converse is straightforward since an $s-$coin is an unbiased estimator of s with values in $[0,1]$.
- **Algorithm 1**
  1. simulate $G_0 \sim U(0,1)$;
  2. obtain $\hat{S}$;
  3. if $G_0 \leq \hat{S}$ set $C_s := 1$, otherwise set $C_s := 0$;
  4. output C_s.

Algorithm 2 - lower and upper monotone deterministic bounds

- let $l_1, l_2, \ldots$ and $u_1, u_2, \ldots$ be sequences of lower and upper monotone bounds for s converging to s, i.e.

$$l_i \nearrow s \qquad \text{and} \qquad u_i \searrow s.$$

Algorithm 2 - lower and upper monotone deterministic bounds

- let $l_1, l_2, \ldots$ and $u_1, u_2, \ldots$ be sequences of lower and upper monotone bounds for s converging to s, i.e.

$$l_i \nearrow s \qquad \text{and} \qquad u_i \searrow s.$$

- **Algorithm 2**
  1. simulate $G_0 \sim U(0,1)$; set $n = 1$;
  2. compute l_n and u_n;
  3. if $G_0 \leq l_n$ set $C_s := 1$;
  4. if $G_0 > u_n$ set $C_s := 0$;
  5. if $l_n < G_0 \leq u_n$ set $n := n + 1$ and GOTO 2;
  6. output C_s.

Algorithm 2 - lower and upper monotone deterministic bounds

- let $l_1, l_2, \ldots$ and $u_1, u_2, \ldots$ be sequences of lower and upper monotone bounds for s converging to s, i.e.

$$l_i \nearrow s \qquad \text{and} \qquad u_i \searrow s.$$

- **Algorithm 2**
  1. simulate $G_0 \sim U(0,1)$; set $n = 1$;
  2. compute l_n and u_n;
  3. if $G_0 \leq l_n$ set $C_s := 1$;
  4. if $G_0 > u_n$ set $C_s := 0$;
  5. if $l_n < G_0 \leq u_n$ set $n := n + 1$ and GOTO 2;
  6. output C_s.
- Remark: $P(N > n) = u_n - l_n$.

This is a practically useful technique, suggested for example in Devroye (1986), and implemented for simulation from random measures in Papaspiliopoulos and R (2008).

Algorithm 3 - monotone stochastic bounds

$$L_n \leq U_n \tag{4}$$

$$L_n \in [0,1] \quad \text{and} \quad U_n \in [0,1] \tag{5}$$

$$L_{n-1} \leq L_n \quad \text{and} \quad U_{n-1} \geq U_n \tag{6}$$

$$\mathbb{E}\, L_n = l_n \nearrow s \quad \text{and} \quad \mathbb{E}\, U_n = u_n \searrow s. \tag{7}$$

$$\mathcal{F}_0 = \{\emptyset, \Omega\}, \quad \mathcal{F}_n = \sigma\{L_n, U_n\}, \quad \mathcal{F}_{k,n} = \sigma\{\mathcal{F}_k, \mathcal{F}_{k+1}, ...\mathcal{F}_n\} \quad \text{for} \quad k \leq n.$$

Algorithm 3 - monotone stochastic bounds

$$L_n \leq U_n \tag{4}$$

$$L_n \in [0,1] \quad \text{and} \quad U_n \in [0,1] \tag{5}$$

$$L_{n-1} \leq L_n \quad \text{and} \quad U_{n-1} \geq U_n \tag{6}$$

$$\mathbb{E}\, L_n = l_n \nearrow s \quad \text{and} \quad \mathbb{E}\, U_n = u_n \searrow s. \tag{7}$$

$$\mathcal{F}_0 = \{\emptyset, \Omega\}, \quad \mathcal{F}_n = \sigma\{L_n, U_n\}, \quad \mathcal{F}_{k,n} = \sigma\{\mathcal{F}_k, \mathcal{F}_{k+1}, ...\mathcal{F}_n\} \quad \text{for} \quad k \leq n.$$

- **Algorithm 3**
  1. simulate $G_0 \sim U(0,1)$; set $n = 1$;
  2. obtain L_n and U_n; conditionally on $\mathcal{F}_{1,n-1}$
  3. if $G_0 \leq L_n$ set $C_s := 1$;
  4. if $G_0 > U_n$ set $C_s := 0$;
  5. if $L_n < G_0 \leq U_n$ set $n := n + 1$ and GOTO 2;
  6. output C_s.

Algorithm 3 - monotone stochastic bounds

$$L_n \leq U_n \tag{4}$$

$$L_n \in [0,1] \quad \text{and} \quad U_n \in [0,1] \tag{5}$$

$$L_{n-1} \leq L_n \quad \text{and} \quad U_{n-1} \geq U_n \tag{6}$$

$$\mathbb{E}\, L_n = l_n \nearrow s \quad \text{and} \quad \mathbb{E}\, U_n = u_n \searrow s. \tag{7}$$

$$\mathcal{F}_0 = \{\emptyset, \Omega\}, \quad \mathcal{F}_n = \sigma\{L_n, U_n\}, \quad \mathcal{F}_{k,n} = \sigma\{\mathcal{F}_k, \mathcal{F}_{k+1}, ...\mathcal{F}_n\} \quad \text{for} \quad k \leq n.$$

- **Algorithm 3**
  1. simulate $G_0 \sim U(0,1)$; set $n = 1$;
  2. obtain L_n and U_n; conditionally on $\mathcal{F}_{1,n-1}$
  3. if $G_0 \leq L_n$ set $C_s := 1$;
  4. if $G_0 > U_n$ set $C_s := 0$;
  5. if $L_n < G_0 \leq U_n$ set $n := n + 1$ and GOTO 2;
  6. output C_s.

Algorithm 3 - monotone stochastic bounds

Lemma

Assume (4), (5), (6) and (7). Then Algorithm 3 outputs a valid $s-$coin. Moreover the probability that it needs $N > n$ iterations equals $u_n - l_n$.

Proof.

Probability that Algorithm 3 needs more then n iterations equals $\mathbb{E}(U_n - L_n) = u_n - l_n \to 0$ as $n \to \infty$. And since $0 \leq U_n - L_n$ is a decreasing sequence a.s., we also have $U_n - L_n \to 0$ a.s. So there exists a random variable $\hat{S}$, such that for almost every realization of sequences $\{L_n(\omega)\}_{n\geq 1}$ and $\{U_n(\omega)\}_{n\geq 1}$ we have $L_n(\omega) \nearrow \hat{S}(\omega)$ and $U_n(\omega) \searrow \hat{S}(\omega)$. By (5) we have $\hat{S} \in [0,1]$ a.s. Thus for a fixed ω the algorithm outputs an $\hat{S}(\omega)-$coin a.s (by Algorithm 2). Clearly $\mathbb{E}\, L_n \leq \mathbb{E}\, \hat{S} \leq \mathbb{E}\, U_n$ and hence $\mathbb{E}\, \hat{S} = s$. □

Algorithm 4 - reverse time martingales

$$
\begin{aligned}
L_n &\leq U_n \\
L_n \in [0,1] \quad &\text{and} \quad U_n \in [0,1] \\
L_{n-1} \leq L_n \quad &\text{and} \quad U_{n-1} \geq U_n \\
\mathbb{E}\, L_n = l_n \nearrow s \quad &\text{and} \quad \mathbb{E}\, U_n = u_n \searrow s.
\end{aligned}
$$

$$
\mathcal{F}_0 = \{\emptyset, \Omega\}, \quad \mathcal{F}_n = \sigma\{L_n, U_n\}, \quad \mathcal{F}_{k,n} = \sigma\{\mathcal{F}_k, \mathcal{F}_{k+1}, ...\mathcal{F}_n\} \quad \text{for} \quad k \leq n.
$$

Algorithm 4 - reverse time martingales

$$
\begin{aligned}
L_n &\leq U_n \\
L_n \in [0,1] \quad &\text{and} \quad U_n \in [0,1] \\
L_{n-1} \leq L_n \quad &\text{and} \quad U_{n-1} \geq U_n \\
\mathbb{E}\, L_n = l_n \nearrow s \quad &\text{and} \quad \mathbb{E}\, U_n = u_n \searrow s.
\end{aligned}
$$

$$
\mathcal{F}_0 = \{\emptyset, \Omega\}, \quad \mathcal{F}_n = \sigma\{L_n, U_n\}, \quad \mathcal{F}_{k,n} = \sigma\{\mathcal{F}_k, \mathcal{F}_{k+1}, ...\mathcal{F}_n\} \quad \text{for} \quad k \leq n.
$$

The final step is to weaken 3rd condition and let L_n be a reverse time supermartingale and U_n a reverse time submartingale with respect to $\mathcal{F}_{n,\infty}$. Precisely, assume that for every $n = 1, 2, ...$ we have

$$
\mathbb{E}\,(L_{n-1} \mid \mathcal{F}_{n,\infty}) = \mathbb{E}\,(L_{n-1} \mid \mathcal{F}_n) \leq L_n \ \text{a.s.} \quad \text{and} \tag{8}
$$

$$
\mathbb{E}\,(U_{n-1} \mid \mathcal{F}_{n,\infty}) = \mathbb{E}\,(U_{n-1} \mid \mathcal{F}_n) \geq U_n \ \text{a.s.} \tag{9}
$$

Algorithm 4 - reverse time martingales

- **Algorithm 4**
  1. simulate $G_0 \sim U(0,1)$; set $n = 1$; set $L_0 \equiv \tilde{L}_0 \equiv 0$ and $U_0 \equiv \tilde{U}_0 \equiv 1$

Algorithm 4 - reverse time martingales

- **Algorithm 4**
  1. simulate $G_0 \sim U(0,1)$; set $n = 1$; set $L_0 \equiv \tilde{L}_0 \equiv 0$ and $U_0 \equiv \tilde{U}_0 \equiv 1$
  2. obtain L_n and U_n given $\mathcal{F}_{0,n-1}$,

Algorithm 4 - reverse time martingales

- **Algorithm 4**
  1. simulate $G_0 \sim U(0,1)$; set $n = 1$; set $L_0 \equiv \tilde{L}_0 \equiv 0$ and $U_0 \equiv \tilde{U}_0 \equiv 1$
  2. obtain L_n and U_n given $\mathcal{F}_{0,n-1}$,
  3. compute $L_n^* = \mathbb{E}\left(L_{n-1} \mid \mathcal{F}_n\right)$ and $U_n^* = \mathbb{E}\left(U_{n-1} \mid \mathcal{F}_n\right)$.

Algorithm 4 - reverse time martingales

- **Algorithm 4**
  1. simulate $G_0 \sim U(0,1)$; set $n = 1$; set $L_0 \equiv \tilde{L}_0 \equiv 0$ and $U_0 \equiv \tilde{U}_0 \equiv 1$
  2. obtain L_n and U_n given $\mathcal{F}_{0,n-1}$,
  3. compute $L_n^* = \mathbb{E}\left(L_{n-1} \mid \mathcal{F}_n\right)$ and $U_n^* = \mathbb{E}\left(U_{n-1} \mid \mathcal{F}_n\right)$.
  4. compute

$$\tilde{L}_n = \tilde{L}_{n-1} + \frac{L_n - L_n^*}{U_n^* - L_n^*}\left(\tilde{U}_{n-1} - \tilde{L}_{n-1}\right) \tag{10}$$

$$\tilde{U}_n = \tilde{U}_{n-1} - \frac{U_n^* - U_n}{U_n^* - L_n^*}\left(\tilde{U}_{n-1} - \tilde{L}_{n-1}\right) \tag{11}$$

  5. if $G_0 \leq \tilde{L}_n$ set $C_s := 1$;

Algorithm 4 - reverse time martingales

- **Algorithm 4**
  1. simulate $G_0 \sim U(0,1)$; set $n = 1$; set $L_0 \equiv \tilde{L}_0 \equiv 0$ and $U_0 \equiv \tilde{U}_0 \equiv 1$
  2. obtain L_n and U_n given $\mathcal{F}_{0,n-1}$,
  3. compute $L_n^* = \mathbb{E}\left(L_{n-1} \mid \mathcal{F}_n\right)$ and $U_n^* = \mathbb{E}\left(U_{n-1} \mid \mathcal{F}_n\right)$.
  4. compute

$$\tilde{L}_n = \tilde{L}_{n-1} + \frac{L_n - L_n^*}{U_n^* - L_n^*}\left(\tilde{U}_{n-1} - \tilde{L}_{n-1}\right) \tag{10}$$

$$\tilde{U}_n = \tilde{U}_{n-1} - \frac{U_n^* - U_n}{U_n^* - L_n^*}\left(\tilde{U}_{n-1} - \tilde{L}_{n-1}\right) \tag{11}$$

  5. if $G_0 \leq \tilde{L}_n$ set $C_s := 1$;
  6. if $G_0 > \tilde{U}_n$ set $C_s := 0$;

Algorithm 4 - reverse time martingales

- **Algorithm 4**
  1. simulate $G_0 \sim U(0,1)$; set $n = 1$; set $L_0 \equiv \tilde{L}_0 \equiv 0$ and $U_0 \equiv \tilde{U}_0 \equiv 1$
  2. obtain L_n and U_n given $\mathcal{F}_{0,n-1}$,
  3. compute $L_n^* = \mathbb{E}\left(L_{n-1} \mid \mathcal{F}_n\right)$ and $U_n^* = \mathbb{E}\left(U_{n-1} \mid \mathcal{F}_n\right)$.
  4. compute

$$\tilde{L}_n = \tilde{L}_{n-1} + \frac{L_n - L_n^*}{U_n^* - L_n^*}\left(\tilde{U}_{n-1} - \tilde{L}_{n-1}\right) \tag{10}$$

$$\tilde{U}_n = \tilde{U}_{n-1} - \frac{U_n^* - U_n}{U_n^* - L_n^*}\left(\tilde{U}_{n-1} - \tilde{L}_{n-1}\right) \tag{11}$$

  5. if $G_0 \leq \tilde{L}_n$ set $C_s := 1$;
  6. if $G_0 > \tilde{U}_n$ set $C_s := 0$;
  7. if $\tilde{L}_n < G_0 \leq \tilde{U}_n$ set $n := n + 1$ and GOTO 2;
  8. output C_s.

Algorithm 4 - reverse time martingales

Theorem

Assume (4), (5), (7), (8) and (9). Then Algorithm 4 outputs a valid $s-$coin. Moreover the probability that it needs $N > n$ iterations equals $u_n - l_n$.

We show that $\tilde{L}$ and $\tilde{U}$ satisfy (4), (5), (7) and (6) and hence Algorithm 4 is valid since Algorithm 3 was valid.
In fact, we have constructed a **mean preserving** transformation, in the sense $\mathbb{E}[\tilde{L}_n] = \mathbb{E}[L_n]$, and $\mathbb{E}[\tilde{U}_n] = \mathbb{E}[U_n]$. Therefore, the proof is based on establishing this property and appealing to Algorithm 3.

- By construction $L_0 = 0, U_0 = 1$ a.s thus $L_0^* = 0, U_0^* = 1$

- By construction $L_0 = 0, U_0 = 1$ a.s thus $L_0^* = 0, U_0^* = 1$
- Therefore, $\tilde{L}_1 = L_1$, and $\tilde{U}_1 = U_1$ (intuitive)

- By construction $L_0 = 0, U_0 = 1$ a.s thus $L_0^* = 0, U_0^* = 1$
- Therefore, $\tilde{L}_1 = L_1$, and $\tilde{U}_1 = U_1$ (intuitive)
- Thus,

$$\tilde{L}_2 = L_1 + \frac{L_2 - L_2^*}{U_2^* - L_2^*}(U_1 - L_1)$$

- By construction $L_0 = 0, U_0 = 1$ a.s thus $L_0^* = 0, U_0^* = 1$
- Therefore, $\tilde{L}_1 = L_1$, and $\tilde{U}_1 = U_1$ (intuitive)
- Thus,

$$\tilde{L}_2 = L_1 + \frac{L_2 - L_2^*}{U_2^* - L_2^*}(U_1 - L_1)$$

- Take conditional expectation given $\mathcal{F}_2$
- Therefore result holds for $n = 1, 2$, then use induction (following the same approach)

A version of the Nacu-Peres Theorem

An algorithm that simulates a function f on $\mathcal{P} \subseteq (0,1)$ exists if and only if for all $n \geq 1$ there exist polynomials $g_n(p)$ and $h_n(p)$ of the form

$$g_n(p) = \sum_{k=0}^{n} \binom{n}{k} a(n,k) p^k (1-p)^{n-k} \text{ and } h_n(p) = \sum_{k=0}^{n} \binom{n}{k} b(n,k) p^k (1-p)^{n-k},$$

s.t.

(i) $0 \leq a(n,k) \leq b(n,k) \leq 1$,

(ii) $\lim_{n\to\infty} g_n(p) = f(p) = \lim_{n\to\infty} h_n(p)$,

(iii) For all $m < n$, their coefficients satisfy

$$a(n,k) \geq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} a(m,i), \qquad b(n,k) \leq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} b(m,i). \quad (12)$$

Algorithm 4 - reverse time martingales

Proof: *polynomials* $\Rightarrow$ *algorithm*.

- Let $X_1, X_2, \ldots$ iid tosses of a $p-$coin.

Algorithm 4 - reverse time martingales

Proof: *polynomials* $\Rightarrow$ *algorithm*.

- Let $X_1, X_2, \ldots$ iid tosses of a $p-$coin.
- Define $\{L_n, U_n\}_{n \geq 1}$ as follows:

$\square$

Algorithm 4 - reverse time martingales

Proof: *polynomials* $\Rightarrow$ *algorithm*.

- Let $X_1, X_2, \ldots$ iid tosses of a $p-$coin.
- Define $\{L_n, U_n\}_{n\geq 1}$ as follows:
- if $\sum_{i=1}^{n} X_i = k$, let $L_n = a(n,k)$ and $U_n = b(n,k)$.

$\square$

Algorithm 4 - reverse time martingales

Proof: *polynomials* $\Rightarrow$ *algorithm*.

- Let $X_1, X_2, \ldots$ iid tosses of a $p-$coin.
- Define $\{L_n, U_n\}_{n\geq 1}$ as follows:
- if $\sum_{i=1}^{n} X_i = k$, let $L_n = a(n,k)$ and $U_n = b(n,k)$.
- In the rest of the proof we check that (4), (5), (7), (8) and (9) hold for $\{L_n, U_n\}_{n\geq 1}$ with $s = f(p)$. Thus executing Algorithm 4 with $\{L_n, U_n\}_{n\geq 1}$ yields a valid $f(p)-$coin.
  Clearly (4) and (5) hold due to (i). For (7) note that
  $\mathbb{E}\, L_n = g_n(p) \nearrow f(p)$ and $\mathbb{E}\, U_n = h_n(p) \searrow f(p)$.

$\square$

Algorithm 4 - reverse time martingales

Proof - continued.

- To obtain (8) and (9) define the sequence of random variables H_n to be

Algorithm 4 - reverse time martingales

Proof - continued.

- To obtain (8) and (9) define the sequence of random variables H_n to be
- the number of heads in $\{X_1, \ldots, X_n\}$, i.e. $H_n = \sum_{i=1}^{n} X_i$

Algorithm 4 - reverse time martingales

Proof - continued.

- To obtain (8) and (9) define the sequence of random variables H_n to be
- the number of heads in $\{X_1, \ldots, X_n\}$, i.e. $H_n = \sum_{i=1}^{n} X_i$
- and let $\mathcal{G}_n = \sigma(H_n)$. Thus $L_n = a(n, H_n)$ and $U_n = b(n, H_n)$, hence $\mathcal{F}_n \subseteq \mathcal{G}_n$ and it is enough to check that $\mathbb{E}(L_m|\mathcal{G}_n) \leq L_n$ and $\mathbb{E}(U_m|\mathcal{G}_n) \geq U_n$ for $m < n$.

Algorithm 4 - reverse time martingales

Proof - continued.

- To obtain (8) and (9) define the sequence of random variables H_n to be
- the number of heads in $\{X_1, \ldots, X_n\}$, i.e. $H_n = \sum_{i=1}^{n} X_i$
- and let $\mathcal{G}_n = \sigma(H_n)$. Thus $L_n = a(n, H_n)$ and $U_n = b(n, H_n)$, hence $\mathcal{F}_n \subseteq \mathcal{G}_n$ and it is enough to check that $\mathbb{E}(L_m|\mathcal{G}_n) \leq L_n$ and $\mathbb{E}(U_m|\mathcal{G}_n) \geq U_n$ for $m < n$.
- The distribution of H_m given H_n is hypergeometric and

$$\mathbb{E}(L_m|\mathcal{G}_n) = \mathbb{E}(a(m, H_m)|H_n) = \sum_{i=0}^{H_n} \frac{\binom{n-m}{H_n-i}\binom{m}{i}}{\binom{n}{H_n}} a(m, i) \leq a(n, H_n) = L_n.$$

Clearly the distribution of H_m given H_n is the same as the distribution of H_m given $\{H_n, H_{n+1}, \ldots\}$. The argument for U_n is identical.

□

Conclusions

- Provides a practical implementable solution to the Bernoulli factory problem.

Conclusions

- Provides a practical implementable solution to the Bernoulli factory problem.
- However methods are slow when $f(p)$ is close to 1.

Conclusions

- Provides a practical implementable solution to the Bernoulli factory problem.
- However methods are slow when $f(p)$ is close to 1.
- Currently using the method in work on "exact" inference for diffusions.

Conclusions

- Provides a practical implementable solution to the Bernoulli factory problem.
- However methods are slow when $f(p)$ is close to 1.
- Currently using the method in work on "exact" inference for diffusions.
- The *retrospective simulation* ideas central to this talk have many applications in simulation problems, eg exact simulation of diffusions, Bayesian inference using Dirichlet mixture (and related) models, coupling from the past, etc.

Some References

- A. Beskos, G.O. Roberts. Exact simulation of diffusions. *Ann. Appl. Probab.* 15(4): 2422–2444, 2005.
- A. Beskos, O. Papaspiliopoulos, G.O. Roberts. Retrospective exact simulation of diffusion sample paths with applications. *Bernoulli* 12: 1077–1098, 2006.
- A. Beskos, O. Papaspiliopoulos, G.O. Roberts, and P. Fearnhead. Exact and computationally efficient likelihood-based estimation for discreetly observed diffusion processes (with discussion). *Journal of the Royal Statistical Society B,* 68(3):333–382, 2006.
- L. Devroye. *Non-uniform random variable generation.* Springer-Verlag, New York, 1986.
- M.S. Keane and G.L. O'Brien. A Bernoulli factory. *ACM Transactions on Modelling and Computer Simulation (TOMACS),* 4(2):213–219, 1994.

- P.E. Kloeden and E.Platen. *Numerical solution of stochastic differential equations*, Springer-Verlag, 1995.
- K. Latuszynski, I. Kosmidis, O. Papaspiliopoulos, G.O. Roberts. Simulating events of unknown probabilities via reverse time martingales. *Random Structures and Algorithms, to appear.*
- S. Nacu and Y. Peres. Fast simulation of new coins from old. *Annals of Applied Probability,* 15(1):93–115, 2005.
- O. Papaspiliopoulos, G.O. Roberts. Retrospective Markov chain Monte Carlo for Dirichlet process hierarchical models. *Biometrika,* 95:169–186, 2008.
- Y. Peres. Iterating von Neumann's procedure for extracting random bits. *Annals of Statistics,* 20(1): 590–597, 1992.