The Bernoulli Factory, extensions and applications

making black boxes out of black boxes for practical purposes

Krzysztof Latuszynski
(University of Warwick, UK)

joint work with

Flavio B. Gonçalves Ioannis Kosmidis Jan Palczewski
Omiros Papaspiliopoulos Gareth O. Roberts Alexander Wendland

Isaac Newton Institute, Cambridge
April 23, 2014

The Benoulli Factory problem

- let $p \in (0, 1)$ be **unknown**
- given a black box that generates a sequence
$$X_1, X_2, \ldots$$
of $p-$coins
- is it possible to generate an
$$f(p) - \text{coin}$$
for a **known** f ?
- for example
$$f(p) = \min\{1, 2p\}.$$

The Benoulli Factory problem

- let $p \in (0,1)$ be **unknown**
- given a black box that generates a sequence

$$X_1, X_2, \ldots$$

  of $p-$coins
- is it possible to generate an

$$f(p) - \text{coin}$$

  for a **known** f ?
- for example

$$f(p) = \min\{1, 2p\}.$$

The Benoulli Factory problem

- let $p \in (0,1)$ be **unknown**
- given a black box that generates a sequence

$$X_1, X_2, \dots$$

  of $p-$coins
- is it possible to generate an

$$f(p) - \text{coin}$$

  for a **known** f ?
- for example

$$f(p) = \min\{1, 2p\}.$$

The Benoulli Factory problem

- let $p \in (0,1)$ be **unknown**
- given a black box that generates a sequence

$$X_1, X_2, \ldots$$

  of $p-$coins
- is it possible to generate an

$$f(p) - \text{coin}$$

  for a **known** f ?
- for example

$$f(p) = \min\{1, 2p\}.$$

Some history

- in 1951 von Neumann posed and solved

$$f(p) = 1/2$$

- Algorithm
  1. set $n = 1$;
  2. use the black box to 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.
- Asmussen posed an open poblem for:

$$f(p) = 2p$$

- but it turned out difficult

Some history

- in 1951 von Neumann posed and solved

$$f(p) = 1/2$$

- Algorithm
  1. set $n = 1$;
  2. use the black box to 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.
- Asmussen posed an open poblem for:

$$f(p) = 2p$$

- but it turned out difficult

Some history

- in 1951 von Neumann posed and solved

$$f(p) = 1/2$$

- Algorithm
  1. set $n = 1$;
  2. use the black box to 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.
- Asmussen posed an open poblem for:

$$f(p) = 2p$$

- but it turned out difficult

Some history

- in 1951 von Neumann posed and solved

$$f(p) = 1/2$$

- Algorithm
  1. set $n = 1$;
  2. use the black box to 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.
- Asmussen posed an open poblem for:

$$f(p) = 2p$$

- but it turned out difficult

Motivation I - MCMC for jump diffusions

- MCMC for jump diffusions with stochastic jump rate (ongoing work - F.B. Gonçalves, G.O. Roberts, KL)
- Consider the model $t \in [0, T]$

$$\begin{aligned} \gamma_t &\sim \text{Ornstein-Uhlenbeck}(\theta_1) \\ \lambda_t &= \exp(\gamma_t) \\ J_t &\sim \text{JumpProcess}(\lambda_t, d\Delta) \\ dV_t &= \mu(V_{t-}, \theta_2)dt + \sigma(V_{t-}, \theta_2)dB_t + dJ_t \end{aligned}$$

- Gibbs sampling **from the full posterior** will alternate between

$$((J_t, V_t) \mid \cdot) \; ; \; (\lambda_t \mid \cdot) \; ; \; (\theta_1 \mid \cdot) \; ; \; (\theta_2 \mid \cdot)$$

- let's have a look at updating

$$(\lambda_t \mid \cdot)$$

Motivation I - MCMC for jump diffusions

- MCMC for jump diffusions with stochastic jump rate (ongoing work - F.B. Gonçalves, G.O. Roberts, KL)
- Consider the model $t \in [0, T]$

$$\begin{aligned}
\gamma_t &\sim \text{Ornstein-Uhlenbeck}(\theta_1) \\
\lambda_t &= \exp(\gamma_t) \\
J_t &\sim \text{JumpProcess}(\lambda_t, d\Delta) \\
dV_t &= \mu(V_{t-}, \theta_2)dt + \sigma(V_{t-}, \theta_2)dB_t + dJ_t
\end{aligned}$$

- Gibbs sampling **from the full posterior** will alternate between

$$((J_t, V_t) \mid \cdot) \; ; \; (\lambda_t \mid \cdot) \; ; \; (\theta_1 \mid \cdot) \; ; \; (\theta_2 \mid \cdot)$$

- let's have a look at updating

$$(\lambda_t \mid \cdot)$$

Motivation I - MCMC for jump diffusions

- MCMC for jump diffusions with stochastic jump rate (ongoing work - F.B. Gonçalves, G.O. Roberts, KL)
- Consider the model $t \in [0, T]$

$$\begin{aligned}
\gamma_t &\sim \text{Ornstein-Uhlenbeck}(\theta_1) \\
\lambda_t &= \exp(\gamma_t) \\
J_t &\sim \text{JumpProcess}(\lambda_t, d\Delta) \\
dV_t &= \mu(V_{t-}, \theta_2)dt + \sigma(V_{t-}, \theta_2)dB_t + dJ_t
\end{aligned}$$

- Gibbs sampling **from the full posterior** will alternate between

$$\big((J_t, V_t) \mid \cdot\big) \; ; \; (\lambda_t \mid \cdot) \; ; \; (\theta_1 \mid \cdot) \; ; \; (\theta_2 \mid \cdot)$$

- let's have a look at updating

$$(\lambda_t \mid \cdot)$$

Motivation I - MCMC for jump diffusions

- MCMC for jump diffusions with stochastic jump rate (ongoing work - F.B. Gonçalves, G.O. Roberts, KL)
- Consider the model $t \in [0, T]$

$$
\begin{aligned}
\gamma_t &\sim \text{Ornstein-Uhlenbeck}(\theta_1) \\
\lambda_t &= \exp(\gamma_t) \\
J_t &\sim \text{JumpProcess}(\lambda_t, d\Delta) \\
dV_t &= \mu(V_{t-}, \theta_2)dt + \sigma(V_{t-}, \theta_2)dB_t + dJ_t
\end{aligned}
$$

- Gibbs sampling **from the full posterior** will alternate between

$$\big((J_t, V_t) \mid \cdot\big) \;;\; (\lambda_t \mid \cdot) \;;\; (\theta_1 \mid \cdot) \;;\; (\theta_2 \mid \cdot)$$

- let's have a look at updating

$$(\lambda_t \mid \cdot)$$

Motivation I - MCMC for jump diffusions

- for updating $(\lambda_t \mid \cdot)$ compute

$$p(\gamma_t|\cdot) = p(\gamma_t|J_t) \propto p(\gamma_t)p(J_t|\gamma_t) = p(\gamma_t)\exp\left\{-\int_0^T e^{\gamma_t}dt + \sum_{j=1}^{N_J}\gamma_{t_j}\right\}$$

$$= p(\gamma_t)K_\gamma \exp\left\{-\int_0^T e^{\gamma_t}dt\right\} = p(\gamma)K_\gamma I(\gamma)$$

- If proposal $= p(\gamma_t)$, then the **Metropolis acceptance rate** is of the form

$$\alpha(\gamma^{(i)}, \gamma^{(i+1)}) = \min\{1, K_{(\gamma^{(i)},\gamma^{(i+1)})} I(\gamma^{(i)}, \gamma^{(i+1)})\}, \qquad \text{where}$$

- $K_{(\gamma^{(i)},\gamma^{(i+1)})}$ is a **known constant**
- We **have a mechanism** to generate events of probability $I(\gamma^{(i)}, \gamma^{(i+1)})$
- so we have the Bernoulli factory problem with

$$f(p) = \min\{1, Kp\}.$$

Motivation I - MCMC for jump diffusions

- for updating $(\lambda_t \mid \cdot)$ compute

$$p(\gamma_t|\cdot) \;=\; p(\gamma_t|J_t) \;\propto\; p(\gamma_t)p(J_t|\gamma_t) \;=\; p(\gamma_t)\exp\left\{-\int_0^T e^{\gamma_t}dt + \sum_{j=1}^{N_J}\gamma_{t_j}\right\}$$

$$= \; p(\gamma_t)K_\gamma \exp\left\{-\int_0^T e^{\gamma_t}dt\right\} \;=\; p(\gamma)K_\gamma I(\gamma)$$

- If proposal $= p(\gamma_t)$, then the **Metropolis acceptance rate** is of the form

$$\alpha(\gamma^{(i)},\gamma^{(i+1)}) \;=\; \min\{1\;,\;K_{(\gamma^{(i)},\gamma^{(i+1)})}I(\gamma^{(i)},\gamma^{(i+1)})\}, \qquad \text{where}$$

- $K_{(\gamma^{(i)},\gamma^{(i+1)})}$ is a **known constant**
- We **have a mechanism** to generate events of probability $I(\gamma^{(i)},\gamma^{(i+1)})$
- so we have the Bernoulli factory problem with

$$f(p) \;=\; \min\{1, Kp\}.$$

Motivation I - MCMC for jump diffusions

- for updating $(\lambda_t \mid \cdot)$ compute

$$p(\gamma_t|\cdot) \;=\; p(\gamma_t|J_t) \;\propto\; p(\gamma_t)p(J_t|\gamma_t) \;=\; p(\gamma_t)\exp\left\{-\int_0^T e^{\gamma_t}dt + \sum_{j=1}^{N_J}\gamma_{t_j}\right\}$$

$$= \; p(\gamma_t)K_\gamma \exp\left\{-\int_0^T e^{\gamma_t}dt\right\} \;=\; p(\gamma)K_\gamma I(\gamma)$$

- If proposal $= p(\gamma_t)$, then the **Metropolis acceptance rate** is of the form

$$\alpha(\gamma^{(i)},\gamma^{(i+1)}) \;=\; \min\{1\;,\;K_{(\gamma^{(i)},\gamma^{(i+1)})}I(\gamma^{(i)},\gamma^{(i+1)})\}, \qquad \text{where}$$

- $K_{(\gamma^{(i)},\gamma^{(i+1)})}$ is a **known constant**
- We **have a mechanism** to generate events of probability $I(\gamma^{(i)},\gamma^{(i+1)})$
- so we have the Bernoulli factory problem with

$$f(p) \;=\; \min\{1, Kp\}.$$

Motivation I - MCMC for jump diffusions

- for updating $(\lambda_t \mid \cdot)$ compute

$$p(\gamma_t|\cdot) \;=\; p(\gamma_t|J_t) \;\propto\; p(\gamma_t)p(J_t|\gamma_t) \;=\; p(\gamma_t)\exp\left\{-\int_0^T e^{\gamma_t}dt + \sum_{j=1}^{N_J}\gamma_{t_j}\right\}$$

$$= \; p(\gamma_t)K_\gamma \exp\left\{-\int_0^T e^{\gamma_t}dt\right\} \;=\; p(\gamma)K_\gamma I(\gamma)$$

- If proposal $= p(\gamma_t)$, then the **Metropolis acceptance rate** is of the form

$$\alpha(\gamma^{(i)},\gamma^{(i+1)}) \;=\; \min\{1\ ,\ K_{(\gamma^{(i)},\gamma^{(i+1)})} I(\gamma^{(i)},\gamma^{(i+1)})\}, \qquad \text{where}$$

- $K_{(\gamma^{(i)},\gamma^{(i+1)})}$ is a **known constant**
- We **have a mechanism** to generate events of probability $I(\gamma^{(i)},\gamma^{(i+1)})$
- so we have the Bernoulli factory problem with

$$f(p) \;=\; \min\{1, Kp\}.$$

Motivation I - MCMC for jump diffusions

- for updating $(\lambda_t \mid \cdot)$ compute

$$p(\gamma_t|\cdot) = p(\gamma_t|J_t) \propto p(\gamma_t)p(J_t|\gamma_t) = p(\gamma_t)\exp\left\{-\int_0^T e^{\gamma_t}dt + \sum_{j=1}^{N_J}\gamma_{t_j}\right\}$$

$$= p(\gamma_t)K_\gamma \exp\left\{-\int_0^T e^{\gamma_t}dt\right\} = p(\gamma)K_\gamma I(\gamma)$$

- If proposal $= p(\gamma_t)$, then the **Metropolis acceptance rate** is of the form

$$\alpha(\gamma^{(i)}, \gamma^{(i+1)}) = \min\{1\ ,\ K_{(\gamma^{(i)},\gamma^{(i+1)})}I(\gamma^{(i)}, \gamma^{(i+1)})\}, \qquad \text{where}$$

- $K_{(\gamma^{(i)},\gamma^{(i+1)})}$ is a **known constant**
- We **have a mechanism** to generate events of probability $I(\gamma^{(i)}, \gamma^{(i+1)})$
- so we have the Bernoulli factory problem with

$$f(p) = \min\{1, Kp\}.$$

Motivation II - MCMC for Markov switching diffusions

- MCMC for Markov switching diffusions
(ongoing work - KL, J. Palczewski, G.O. Roberts)
- Consider the model $t \in [0, T]$

$$\begin{aligned} Y_t &\sim \text{Cont time Markov chain on } \{1, \ldots, m\} \text{ with intensity matrix } \Lambda \\ dV_t &= \mu(V_t, Y_t, \theta)dt + \sigma(V_t, \theta)\gamma(Y_t, \theta)dB_t \end{aligned}$$

- Gibbs sampling **from the full posterior** will alternate between

$$(V_t \mid \cdot) \; ; \; (Y_t \mid \cdot) \; ; \; (\Lambda \mid \cdot) \; ; \; (\theta \mid \cdot)$$

- let's have a look at updating

$$(Y_t \mid \cdot)$$

Motivation II - MCMC for Markov switching diffusions

- MCMC for Markov switching diffusions (ongoing work - KL, J. Palczewski, G.O. Roberts)
- Consider the model $t \in [0, T]$

$$\begin{aligned} Y_t &\sim \text{Cont time Markov chain on } \{1, \ldots, m\} \text{ with intensity matrix } \Lambda \\ dV_t &= \mu(V_t, Y_t, \theta)dt + \sigma(V_t, \theta)\gamma(Y_t, \theta)dB_t \end{aligned}$$

- Gibbs sampling **from the full posterior** will alternate between

$$(V_t \mid \cdot) \;;\; (Y_t \mid \cdot) \;;\; (\Lambda \mid \cdot) \;;\; (\theta \mid \cdot)$$

- let's have a look at updating

$$(Y_t \mid \cdot)$$

Motivation II - MCMC for Markov switching diffusions

- MCMC for Markov switching diffusions
  (ongoing work - KL, J. Palczewski, G.O. Roberts)
- Consider the model $t \in [0, T]$

$$\begin{aligned} Y_t &\sim \text{Cont time Markov chain on } \{1, \ldots, m\} \text{ with intensity matrix } \Lambda \\ dV_t &= \mu(V_t, Y_t, \theta)dt + \sigma(V_t, \theta)\gamma(Y_t, \theta)dB_t \end{aligned}$$

- Gibbs sampling **from the full posterior** will alternate between

$$(V_t \mid \cdot) \ ; \ (Y_t \mid \cdot) \ ; \ (\Lambda \mid \cdot) \ ; \ (\theta \mid \cdot)$$

- let's have a look at updating

$$(Y_t \mid \cdot)$$

Motivation II - MCMC for Markov switching diffusions

- MCMC for Markov switching diffusions
  (ongoing work - KL, J. Palczewski, G.O. Roberts)
- Consider the model $t \in [0, T]$

$$\begin{aligned} Y_t &\sim \text{Cont time Markov chain on } \{1, \ldots, m\} \text{ with intensity matrix } \Lambda \\ dV_t &= \mu(V_t, Y_t, \theta)dt + \sigma(V_t, \theta)\gamma(Y_t, \theta)dB_t \end{aligned}$$

- Gibbs sampling **from the full posterior** will alternate between

$$(V_t \mid \cdot) \ ; \ (Y_t \mid \cdot) \ ; \ (\Lambda \mid \cdot) \ ; \ (\theta \mid \cdot)$$

- let's have a look at updating

$$(Y_t \mid \cdot)$$

Motivation II - MCMC for Markov switching diffusions

- for updating $(Y_t \mid \cdot)$ compute

$$p(Y_t|\cdot) \quad \propto \quad p(Y_t|\Lambda)G(V_t, Y_t, \theta)$$

- If proposal $= p(Y_t|\Lambda)$, then the **Metropolis acceptance rate** is of the form

$$\alpha(Y^{(i)}, Y^{(i+1)}) \;=\; \min\left\{1\,,\; K_{(Y^{(i)},Y^{(i+1)})}\tilde{G}(V_t,(Y^{(i)},Y^{(i+1)}),\theta)\right\}, \qquad \text{where}$$

- $K_{(Y^{(i)},Y^{(i+1)})}$ is a **known constant**
- We **have a mechanism** to generate event of probability $\tilde{G}(V_t,(Y^{(i)},Y^{(i+1)}),\theta)$
- so again we have the Bernoulli factory problem with

$$f(p) \;=\; \min\{1, Kp\}.$$

Motivation II - MCMC for Markov switching diffusions

- for updating $(Y_t \mid \cdot)$ compute

$$p(Y_t|\cdot) \quad \propto \quad p(Y_t|\Lambda)G(V_t, Y_t, \theta)$$

- If proposal $= p(Y_t|\Lambda)$, then the **Metropolis acceptance rate** is of the form

$$\alpha(Y^{(i)}, Y^{(i+1)}) \;=\; \min\left\{1\ ,\ K_{(Y^{(i)},Y^{(i+1)})}\tilde{G}(V_t, (Y^{(i)}, Y^{(i+1)}), \theta)\right\}, \qquad \text{where}$$

- $K_{(Y^{(i)},Y^{(i+1)})}$ is a **known constant**
- We **have a mechanism** to generate event of probability $\tilde{G}(V_t, (Y^{(i)}, Y^{(i+1)}), \theta)$
- so again we have the Bernoulli factory problem with

$$f(p) \;=\; \min\{1, Kp\}.$$

Motivation II - MCMC for Markov switching diffusions

- for updating $(Y_t \mid \cdot)$ compute

$$p(Y_t|\cdot) \quad \propto \quad p(Y_t|\Lambda)G(V_t, Y_t, \theta)$$

- If proposal $= p(Y_t|\Lambda)$, then the **Metropolis acceptance rate** is of the form

$$\alpha(Y^{(i)}, Y^{(i+1)}) \;=\; \min\left\{1\;,\; K_{(Y^{(i)},Y^{(i+1)})}\tilde{G}\big(V_t, (Y^{(i)}, Y^{(i+1)}), \theta\big)\right\}, \qquad \text{where}$$

- $K_{(Y^{(i)},Y^{(i+1)})}$ is a **known constant**
- We **have a mechanism** to generate event of probability $\tilde{G}(V_t, (Y^{(i)}, Y^{(i+1)}), \theta)$
- so again we have the Bernoulli factory problem with

$$f(p) \;=\; \min\{1, Kp\}.$$

Motivation II - MCMC for Markov switching diffusions

- for updating $(Y_t \mid \cdot)$ compute

$$p(Y_t|\cdot) \quad \propto \quad p(Y_t|\Lambda)G(V_t, Y_t, \theta)$$

- If proposal $= p(Y_t|\Lambda)$, then the **Metropolis acceptance rate** is of the form

$$\alpha(Y^{(i)}, Y^{(i+1)}) \; = \; \min\left\{1\ ,\ K_{(Y^{(i)},Y^{(i+1)})}\tilde{G}\big(V_t, (Y^{(i)}, Y^{(i+1)}), \theta\big)\right\}, \qquad \text{where}$$

- $K_{(Y^{(i)},Y^{(i+1)})}$ is a **known constant**
- We **have a mechanism** to generate event of probability $\tilde{G}\big(V_t, (Y^{(i)}, Y^{(i+1)}), \theta\big)$
- so again we have the Bernoulli factory problem with

$$f(p) \; = \; \min\{1, Kp\}.$$

Motivation II - MCMC for Markov switching diffusions

- for updating $(Y_t \mid \cdot)$ compute

$$p(Y_t|\cdot) \quad \propto \quad p(Y_t|\Lambda)G(V_t, Y_t, \theta)$$

- If proposal $= p(Y_t|\Lambda)$, then the **Metropolis acceptance rate** is of the form

$$\alpha(Y^{(i)}, Y^{(i+1)}) \;=\; \min\left\{1\,,\; K_{(Y^{(i)},Y^{(i+1)})}\tilde{G}\big(V_t, (Y^{(i)}, Y^{(i+1)}), \theta\big)\right\}, \qquad \text{where}$$

- $K_{(Y^{(i)},Y^{(i+1)})}$ is a **known constant**
- We **have a mechanism** to generate event of probability $\tilde{G}\big(V_t, (Y^{(i)}, Y^{(i+1)}), \theta\big)$
- so again we have the Bernoulli factory problem with

$$f(p) \;=\; \min\{1, Kp\}.$$

Motivation III - perfect sampling for Markov chains

- Consider $\{X_n\}_{n\geq 0}$ an ergodic Markov chain with transition kernel P and limiting distribution π.
- Under mild assumptions P can be decomposed

$$P(x,\cdot) = s(x)\nu(\cdot) + (1-s(x))Q(x,\cdot)$$

- and every time we sample from P we flip a coin with probability $s(x)$ to decide between sampling from $\nu(\cdot)$ and $Q(x,\cdot)$
- Let τ be the first time the coin points at $\nu(\cdot)$
- then $\pi(\cdot)$ admits the decomposition

$$\pi(\cdot) = \sum_{n=0}^{\infty} p_n Q^n(\nu,\cdot) \qquad \text{where} \quad p_n := \frac{\Pr(\tau \geq n)}{\mathbb{E}(\tau)}.$$

- it looks like perfect sampling from π is possible using rejection sampling. (S.Assmussen, P.W.Glynn, H.Thorisson 1992; J.P.Hobert, C.P.Robert 2004; J.Blanchet, X-L.Meng 2005; J.Blanchet, A.C.Thomas 2007)

Motivation III - perfect sampling for Markov chains

- Consider $\{X_n\}_{n\geq 0}$ an ergodic Markov chain with transition kernel P and limiting distribution π.
- Under mild assumptions P can be decomposed

$$P(x,\cdot) \;=\; s(x)\nu(\cdot) + (1 - s(x))Q(x,\cdot)$$

- and every time we sample from P we flip a coin with probability $s(x)$ to decide between sampling from $\nu(\cdot)$ and $Q(x,\cdot)$
- Let τ be the first time the coin points at $\nu(\cdot)$
- then $\pi(\cdot)$ admits the decomposition

$$\pi(\cdot) \;=\; \sum_{n=0}^{\infty} p_n Q^n(\nu,\cdot) \qquad \text{where} \quad p_n \;:=\; \frac{\Pr(\tau \geq n)}{\mathbb{E}(\tau)}.$$

- it looks like perfect sampling from π is possible using rejection sampling. (S.Assmussen, P.W.Glynn, H.Thorisson 1992; J.P.Hobert, C.P.Robert 2004; J.Blanchet, X-L.Meng 2005; J.Blanchet, A.C.Thomas 2007)

Motivation III - perfect sampling for Markov chains

- Consider $\{X_n\}_{n\geq 0}$ an ergodic Markov chain with transition kernel P and limiting distribution π.
- Under mild assumptions P can be decomposed

$$P(x,\cdot) = s(x)\nu(\cdot) + (1-s(x))Q(x,\cdot)$$

- and every time we sample from P we flip a coin with probability $s(x)$ to decide between sampling from $\nu(\cdot)$ and $Q(x,\cdot)$
- Let τ be the first time the coin points at $\nu(\cdot)$
- then $\pi(\cdot)$ admits the decomposition

$$\pi(\cdot) = \sum_{n=0}^{\infty} p_n Q^n(\nu,\cdot) \qquad \text{where} \quad p_n := \frac{\Pr(\tau \geq n)}{\mathbb{E}(\tau)}.$$

- it looks like perfect sampling from π is possible using rejection sampling. (S.Assmussen, P.W.Glynn, H.Thorisson 1992; J.P.Hobert, C.P.Robert 2004; J.Blanchet, X-L.Meng 2005; J.Blanchet, A.C.Thomas 2007)

Motivation III - perfect sampling for Markov chains

- Consider $\{X_n\}_{n\geq 0}$ an ergodic Markov chain with transition kernel P and limiting distribution π.
- Under mild assumptions P can be decomposed

$$P(x,\cdot) = s(x)\nu(\cdot) + (1 - s(x))Q(x,\cdot)$$

- and every time we sample from P we flip a coin with probability $s(x)$ to decide between sampling from $\nu(\cdot)$ and $Q(x,\cdot)$
- Let τ be the first time the coin points at $\nu(\cdot)$
- then $\pi(\cdot)$ admits the decomposition

$$\pi(\cdot) = \sum_{n=0}^{\infty} p_n Q^n(\nu,\cdot) \qquad \text{where} \quad p_n := \frac{\Pr(\tau \geq n)}{\mathbb{E}(\tau)}.$$

- it looks like perfect sampling from π is possible using rejection sampling. (S.Assmussen, P.W.Glynn, H.Thorisson 1992; J.P.Hobert, C.P.Robert 2004; J.Blanchet, X-L.Meng 2005; J.Blanchet, A.C.Thomas 2007)

Motivation III - perfect sampling for Markov chains

- Consider $\{X_n\}_{n\geq 0}$ an ergodic Markov chain with transition kernel P and limiting distribution π.
- Under mild assumptions P can be decomposed

$$P(x,\cdot) \;=\; s(x)\nu(\cdot) + (1-s(x))Q(x,\cdot)$$

- and every time we sample from P we flip a coin with probability $s(x)$ to decide between sampling from $\nu(\cdot)$ and $Q(x,\cdot)$
- Let τ be the first time the coin points at $\nu(\cdot)$
- then $\pi(\cdot)$ admits the decomposition

$$\pi(\cdot) \;=\; \sum_{n=0}^{\infty} p_n Q^n(\nu,\cdot) \qquad \text{where} \quad p_n \;:=\; \frac{\Pr(\tau \geq n)}{\mathbb{E}(\tau)}.$$

- it looks like perfect sampling from π is possible using rejection sampling. (S.Assmussen, P.W.Glynn, H.Thorisson 1992; J.P.Hobert, C.P.Robert 2004; J.Blanchet, X-L.Meng 2005; J.Blanchet, A.C.Thomas 2007)

Motivation III - perfect sampling for Markov chains

- Consider $\{X_n\}_{n\geq 0}$ an ergodic Markov chain with transition kernel P and limiting distribution π.
- Under mild assumptions P can be decomposed

$$P(x,\cdot) = s(x)\nu(\cdot) + (1 - s(x))Q(x,\cdot)$$

- and every time we sample from P we flip a coin with probability $s(x)$ to decide between sampling from $\nu(\cdot)$ and $Q(x,\cdot)$
- Let τ be the first time the coin points at $\nu(\cdot)$
- then $\pi(\cdot)$ admits the decomposition

$$\pi(\cdot) = \sum_{n=0}^{\infty} p_n Q^n(\nu,\cdot) \qquad \text{where} \quad p_n := \frac{\Pr(\tau \geq n)}{\mathbb{E}(\tau)}.$$

- it looks like perfect sampling from π is possible using rejection sampling. (S.Assmussen, P.W.Glynn, H.Thorisson 1992; J.P.Hobert, C.P.Robert 2004; J.Blanchet, X-L.Meng 2005; J.Blanchet, A.C.Thomas 2007)

Motivation III - perfect sampling for Markov chains

- $\pi(\cdot)$ admits the decomposition

$$\pi(\cdot) = \sum_{n=0}^{\infty} p_n Q^n(\nu, \cdot) \qquad \text{where} \quad p_n := \frac{\Pr(\tau \geq n)}{\mathbb{E}(\tau)}.$$

- find a probability distribution $d(n)$ s.t. $\Pr(\tau > n) \leq Md(n)$. (e.g. using drift conditions for geometrically ergodic chains)
- Now we can write

$$\frac{\Pr(\tau > n)}{\mathbb{E}(\tau)} = \frac{\Pr(\tau > n)}{\mathbb{E}(\tau)d(n)} d(n)$$

- Goal: reject the $d(n)$ proposal with probability proportional to $\frac{\Pr(\tau > n)}{\mathbb{E}(\tau)d(n)}$
- so we can use

$$\frac{\Pr(\tau > n)}{Md(n)} =: K\Pr(\tau > n) < 1$$

- where K is known and we can sample from $\Pr(\tau > n)$.

Motivation III - perfect sampling for Markov chains

- $\pi(\cdot)$ admits the decomposition

$$\pi(\cdot) = \sum_{n=0}^{\infty} p_n Q^n(\nu, \cdot) \qquad \text{where} \quad p_n := \frac{\Pr(\tau \geq n)}{\mathbb{E}(\tau)}.$$

- find a probability distribution $d(n)$ s.t. $\Pr(\tau > n) \leq Md(n)$.
  (e.g. using drift conditions for geometrically ergodic chains)
- Now we can write

$$\frac{\Pr(\tau > n)}{\mathbb{E}(\tau)} = \frac{\Pr(\tau > n)}{\mathbb{E}(\tau)d(n)} d(n)$$

- Goal: reject the $d(n)$ proposal with probability proportional to $\frac{\Pr(\tau>n)}{\mathbb{E}(\tau)d(n)}$
- so we can use

$$\frac{\Pr(\tau > n)}{Md(n)} =: K\Pr(\tau > n) < 1$$

- where K is known and we can sample from $\Pr(\tau > n)$.

Motivation III - perfect sampling for Markov chains

- $\pi(\cdot)$ admits the decomposition

$$\pi(\cdot) = \sum_{n=0}^{\infty} p_n Q^n(\nu, \cdot) \qquad \text{where} \quad p_n := \frac{\Pr(\tau \geq n)}{\mathbb{E}(\tau)}.$$

- find a probability distribution $d(n)$ s.t. $\Pr(\tau > n) \leq Md(n)$. (e.g. using drift conditions for geometrically ergodic chains)
- Now we can write

$$\frac{\Pr(\tau > n)}{\mathbb{E}(\tau)} = \frac{\Pr(\tau > n)}{\mathbb{E}(\tau) d(n)} d(n)$$

- Goal: reject the $d(n)$ proposal with probability proportional to $\frac{\Pr(\tau > n)}{\mathbb{E}(\tau) d(n)}$
- so we can use

$$\frac{\Pr(\tau > n)}{Md(n)} =: K\Pr(\tau > n) < 1$$

- where K is known and we can sample from $\Pr(\tau > n)$.

Motivation III - perfect sampling for Markov chains

- $\pi(\cdot)$ admits the decomposition

$$\pi(\cdot) = \sum_{n=0}^{\infty} p_n Q^n(\nu, \cdot) \qquad \text{where} \quad p_n := \frac{\Pr(\tau \geq n)}{\mathbb{E}(\tau)}.$$

- find a probability distribution $d(n)$ s.t. $\Pr(\tau > n) \leq Md(n)$.
  (e.g. using drift conditions for geometrically ergodic chains)
- Now we can write

$$\frac{\Pr(\tau > n)}{\mathbb{E}(\tau)} = \frac{\Pr(\tau > n)}{\mathbb{E}(\tau) d(n)} d(n)$$

- Goal: reject the $d(n)$ proposal with probability proportional to $\frac{\Pr(\tau > n)}{\mathbb{E}(\tau) d(n)}$
- so we can use

$$\frac{\Pr(\tau > n)}{Md(n)} =: K\Pr(\tau > n) < 1$$

- where K is known and we can sample from $\Pr(\tau > n)$.

Motivation III - perfect sampling for Markov chains

- $\pi(\cdot)$ admits the decomposition

$$\pi(\cdot) = \sum_{n=0}^{\infty} p_n Q^n(\nu, \cdot) \qquad \text{where} \quad p_n := \frac{\Pr(\tau \geq n)}{\mathbb{E}(\tau)}.$$

- find a probability distribution $d(n)$ s.t. $\Pr(\tau > n) \leq Md(n)$. (e.g. using drift conditions for geometrically ergodic chains)
- Now we can write

$$\frac{\Pr(\tau > n)}{\mathbb{E}(\tau)} = \frac{\Pr(\tau > n)}{\mathbb{E}(\tau)d(n)} d(n)$$

- Goal: reject the $d(n)$ proposal with probability proportional to $\frac{\Pr(\tau>n)}{\mathbb{E}(\tau)d(n)}$
- so we can use

$$\frac{\Pr(\tau > n)}{Md(n)} =: K\Pr(\tau > n) < 1$$

- where K is known and we can sample from $\Pr(\tau > n)$.

Motivation III - perfect sampling for Markov chains

- $\pi(\cdot)$ admits the decomposition

$$\pi(\cdot) = \sum_{n=0}^{\infty} p_n Q^n(\nu, \cdot) \qquad \text{where} \quad p_n := \frac{\Pr(\tau \geq n)}{\mathbb{E}(\tau)}.$$

- find a probability distribution $d(n)$ s.t. $\Pr(\tau > n) \leq Md(n)$. (e.g. using drift conditions for geometrically ergodic chains)
- Now we can write

$$\frac{\Pr(\tau > n)}{\mathbb{E}(\tau)} = \frac{\Pr(\tau > n)}{\mathbb{E}(\tau) d(n)} d(n)$$

- Goal: reject the $d(n)$ proposal with probability proportional to $\frac{\Pr(\tau > n)}{\mathbb{E}(\tau) d(n)}$
- so we can use

$$\frac{\Pr(\tau > n)}{Md(n)} =: K\Pr(\tau > n) < 1$$

- where K is known and we can sample from $\Pr(\tau > n)$.

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 $\Longleftrightarrow$

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

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

- however their proof is **not constructive**
- note that the result rules out $\min\{1, 2p\}$, but not $\min\{1-\varepsilon, 2p\}$.

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, or
  - 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 $\min\{1-\varepsilon, 2p\}$.

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 $\Longleftrightarrow$

- f is constant, or
- 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 $\min\{1-\varepsilon, 2p\}$.

Nacu-Peres Theorem - Bernstein polynomial approach

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

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

- $0 \leq a(n,k) \leq b(n,k) \leq 1$
- $\binom{n}{k} a(n,k)$ 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 $m < n$

$$a(n,k) \geq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} a(m,i), \qquad b(n,k) \leq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} b(m,i). \tag{1}$$

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

Nacu-Peres Theorem - Bernstein polynomial approach

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

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

- $0 \leq a(n,k) \leq b(n,k) \leq 1$
- $\binom{n}{k} a(n,k)$ 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 $m < n$

$$a(n,k) \geq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} a(m,i), \qquad b(n,k) \leq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} b(m,i). \tag{1}$$

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

Nacu-Peres Theorem - Bernstein polynomial approach

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

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

- $0 \leq a(n,k) \leq b(n,k) \leq 1$
- $\binom{n}{k} a(n,k)$ 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 $m < n$

$$a(n,k) \geq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} a(m,i), \qquad b(n,k) \leq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} b(m,i). \tag{1}$$

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

Nacu-Peres Theorem - Bernstein polynomial approach

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

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

- $0 \leq a(n,k) \leq b(n,k) \leq 1$
- $\binom{n}{k} a(n,k)$ 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 $m < n$

$$a(n,k) \geq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} a(m,i), \qquad b(n,k) \leq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} b(m,i). \qquad (1)$$

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

Nacu-Peres Theorem - Bernstein polynomial approach

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

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

- $0 \leq a(n,k) \leq b(n,k) \leq 1$
- $\binom{n}{k} a(n,k)$ 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 $m < n$

$$a(n,k) \geq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} a(m,i), \qquad b(n,k) \leq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} b(m,i). \qquad (1)$$

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

Nacu-Peres Theorem - Bernstein polynomial approach

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

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

- $0 \leq a(n,k) \leq b(n,k) \leq 1$
- $\binom{n}{k} a(n,k)$ 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 $m < n$

$$a(n,k) \geq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} a(m,i), \qquad b(n,k) \leq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} b(m,i). \tag{1}$$

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

Nacu-Peres Theorem - Bernstein polynomial approach

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

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

- $0 \leq a(n,k) \leq b(n,k) \leq 1$
- $\binom{n}{k} a(n,k)$ 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 $m < n$

$$a(n,k) \geq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} a(m,i), \qquad b(n,k) \leq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} b(m,i). \tag{1}$$

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

Summary of theoretical results

- Nacu and Peres show that the random running time of their algorithm has exponentially decaying tails for every real analytic function f.
- There are further interesting theoretical results relating the smoothness of f to existence of Bernoulli Factory algorithms with certain running time. (see O Holtz, F Nazarov, Y Peres, 2011)
- Other results (E Mossel, Y Peres, C Hillar - 2005) relate to constructing a Bernoulli Factory for f rational over $\mathbb{Q}$ be a finite automaton.

Summary of theoretical results

- Nacu and Peres show that the random running time of their algorithm has exponentially decaying tails for every real analytic function f.
- There are further interesting theoretical results relating the smoothness of f to existence of Bernoulli Factory algorithms with certain running time. (see O Holtz, F Nazarov, Y Peres, 2011)
- Other results (E Mossel, Y Peres, C Hillar - 2005) relate to constructing a Bernoulli Factory for f rational over $\mathbb{Q}$ be a finite automaton.

Summary of theoretical results

- Nacu and Peres show that the random running time of their algorithm has exponentially decaying tails for every real analytic function f.
- There are further interesting theoretical results relating the smoothness of f to existence of Bernoulli Factory algorithms with certain running time. (see O Holtz, F Nazarov, Y Peres, 2011)
- Other results (E Mossel, Y Peres, C Hillar - 2005) relate to constructing a Bernoulli Factory for f rational over $\mathbb{Q}$ be a finite automaton.

Bernstein polynomial approach - to nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n
  A_n and B_n are 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

Bernstein polynomial approach - to nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n
  A_n and B_n are 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

Bernstein polynomial approach - to nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n
  A_n and B_n are 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

Bernstein polynomial approach - to nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n
  A_n and B_n are 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

Bernstein polynomial approach - to nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n
  A_n and B_n are 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

Bernstein polynomial approach - to nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n
  A_n and B_n are 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

Bernstein polynomial approach - to nice to be true?

- at time n the N-P algorithm computes sets A_n and B_n
  A_n and B_n are 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

Reverse time martingale approach to sampling

- Reverse time martingale approach to sampling events of unknown probability (KL, I. Kosmidis, O. Papaspiliopoulos, G.O. Roberts, RSA 2011)
- We shall progress gradually from a simple to a general algorithm for sampling events of unknown probabilities **constructively**
- s is the unknown "target" probability ("$s = f(p)$")
- It is determined uniquely but can not be computed and increasing knowledge/precision about s is expensive algorithmically.

Reverse time martingale approach to sampling

- Reverse time martingale approach to sampling events of unknown probability (KL, I. Kosmidis, O. Papaspiliopoulos, G.O. Roberts, RSA 2011)
- We shall progress gradually from a simple to a general algorithm for sampling events of unknown probabilities **constructively**
- s is the unknown "target" probability ("$s = f(p)$")
- It is determined uniquely but can not be computed and increasing knowledge/precision about s is expensive algorithmically.

Reverse time martingale approach to sampling

- Reverse time martingale approach to sampling events of unknown probability (KL, I. Kosmidis, O. Papaspiliopoulos, G.O. Roberts, RSA 2011)
- We shall progress gradually from a simple to a general algorithm for sampling events of unknown probabilities **constructively**
- s is the unknown "target" probability ("$s = f(p)$")
- It is determined uniquely but can not be computed and increasing knowledge/precision about s is expensive algorithmically.

Reverse time martingale approach to sampling

- Reverse time martingale approach to sampling events of unknown probability (KL, I. Kosmidis, O. Papaspiliopoulos, G.O. Roberts, RSA 2011)

- We shall progress gradually from a simple to a general algorithm for sampling events of unknown probabilities **constructively**

- s is the unknown "target" probability ("$s = f(p)$")
- It is determined uniquely but can not be computed and increasing knowledge/precision about s is expensive algorithmically.

Algorithm 0 - randomization

- **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 0**
  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 0 - randomization

- **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 0**
  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 0 - randomization

- **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 0**
  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 1 - 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 1**
  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: $P(N > n) = u_n - l_n$.

Algorithm 1 - 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 1**
  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: $P(N > n) = u_n - l_n$.

Algorithm 1 - 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 1**
  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: $P(N > n) = u_n - l_n$.

Algorithm 2 - monotone stochastic bounds

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

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

- **Algorithm 2**
  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 $G_0 \le L_n$ set $C_s := 1$;
  4. if $G_0 > U_n$ set $C_s := 0$;
  5. if $L_n < G_0 \le U_n$ set $n := n+1$ and GOTO 2;
  6. output C_s.
- **Thm** In the above algorithm $EC_s = s$

Algorithm 2 - monotone stochastic bounds

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

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

- **Algorithm 2**
  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.
- **Thm** In the above algorithm $EC_s = s$

Algorithm 2 - monotone stochastic bounds

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

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

- **Algorithm 2**
  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.
- **Thm** In the above algorithm $EC_s = s$

Algorithm 3 - reverse time martingales

$$L_n \leq U_n \tag{2}$$
$$L_n \in [0,1] \quad \text{and} \quad U_n \in [0,1] \tag{3}$$
$$L_{n-1} \leq L_n \quad \text{and} \quad U_{n-1} \geq U_n \tag{4}$$
$$\mathbb{E}\, L_n = l_n \nearrow s \quad \text{and} \quad \mathbb{E}\, U_n = u_n \searrow s. \tag{5}$$
$$\mathcal{F}_0 = \{\emptyset, \Omega\}, \qquad \mathcal{F}_n = \sigma\{L_n, U_n\}, \qquad \mathcal{F}_{k,n} = \sigma\{\mathcal{F}_k, \mathcal{F}_{k+1}, \ldots \mathcal{F}_n\} \quad \text{for } k \leq n.$$

The final step is to weaken condition (4) 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 $n = 1, 2, \ldots$ we have

$$\mathbb{E}\,(L_{n-1} \mid \mathcal{F}_{n,\infty}) = \mathbb{E}\,(L_{n-1} \mid \mathcal{F}_n) \leq L_n \quad \text{a.s.} \quad \text{and} \tag{6}$$
$$\mathbb{E}\,(U_{n-1} \mid \mathcal{F}_{n,\infty}) = \mathbb{E}\,(U_{n-1} \mid \mathcal{F}_n) \geq U_n \quad \text{a.s.} \tag{7}$$

Algorithm 3 - reverse time martingales

- **Algorithm 3**
  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

$$\begin{aligned}\tilde{L}_n &= \tilde{L}_{n-1} + \frac{L_n - L_n^*}{U_n^* - L_n^*}\left(\tilde{U}_{n-1} - \tilde{L}_{n-1}\right)\\ \tilde{U}_n &= \tilde{U}_{n-1} - \frac{U_n^* - U_n}{U_n^* - L_n^*}\left(\tilde{U}_{n-1} - \tilde{L}_{n-1}\right)\end{aligned}$$

  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.
- $\tilde{L}_n$ and $\tilde{U}_n$ satisfy assumptions of Algorithm 2.

Algorithm 3 - reverse time martingales

- **Algorithm 3**
  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

$$\begin{aligned} \tilde{L}_n &= \tilde{L}_{n-1} + \frac{L_n - L_n^*}{U_n^* - L_n^*}\left(\tilde{U}_{n-1} - \tilde{L}_{n-1}\right) \\ \tilde{U}_n &= \tilde{U}_{n-1} - \frac{U_n^* - U_n}{U_n^* - L_n^*}\left(\tilde{U}_{n-1} - \tilde{L}_{n-1}\right) \end{aligned}$$

  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.
- $\tilde{L}_n$ and $\tilde{U}_n$ satisfy assumptions of Algorithm 2.

Application to the Bernoulli Factory problem

- 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

$$L_n = a(n,k) \qquad \text{and} \qquad U_n = b(n,k).$$

- Verify assumptions of Algorithm 3.
- Here $\{L_n, U_n\}_{n\geq 1}$ are random walks on the coefficients of Nacu-Peres polynomials with **dynamics driven by the original** $p-$coins.

Application to the Bernoulli Factory problem

- 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

$$L_n = a(n,k) \qquad \text{and} \qquad U_n = b(n,k).$$

- Verify assumptions of Algorithm 3.
- Here $\{L_n, U_n\}_{n\geq 1}$ are random walks on the coefficients of Nacu-Peres polynomials with **dynamics driven by the original** $p-$coins.

Application to the Bernoulli Factory problem

- 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
$$L_n = a(n,k) \qquad \text{and} \qquad U_n = b(n,k).$$
- Verify assumptions of Algorithm 3.
- Here $\{L_n, U_n\}_{n\geq 1}$ are random walks on the coefficients of Nacu-Peres polynomials with **dynamics driven by the original** $p-$coins.

Application to the Bernoulli Factory problem

- 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
$$L_n = a(n,k) \qquad \text{and} \qquad U_n = b(n,k).$$
- Verify assumptions of Algorithm 3.
- Here $\{L_n, U_n\}_{n\geq 1}$ are random walks on the coefficients of Nacu-Peres polynomials with **dynamics driven by the original** $p-$coins.

Application to the Bernoulli Factory problem

- 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

$$L_n = a(n,k) \qquad \text{and} \qquad U_n = b(n,k).$$

- Verify assumptions of Algorithm 3.
- Here $\{L_n, U_n\}_{n\geq 1}$ are random walks on the coefficients of Nacu-Peres polynomials with **dynamics driven by the original** $p-$coins.

Application to the Bernoulli Factory problem

- The reverse time martingale approach is the first constructive and practical implementation of a general Bernoulli Factory
- In particular the Nacu-Peres polynomials can be utilised for $f(p) = \min\{1-\varepsilon, Kp\}$ yielding a practical algorithm for the Metropolis accept-reject step in the discussed scenarios (and many others, see e.g. work by R. Herbei and M. Berliner)
- J. Flegal and R. Herbei use the reverse martingale approach to implement the Markov Chain perfect sampling algorithm discussed above.

Application to the Bernoulli Factory problem

- The reverse time martingale approach is the first constructive and practical implementation of a general Bernoulli Factory
- In particular the Nacu-Peres polynomials can be utilised for $f(p) = \min\{1 - \varepsilon, Kp\}$ yielding a practical algorithm for the Metropolis accept-reject step in the discussed scenarios (and many others, see e.g. work by R. Herbei and M. Berliner)
- J. Flegal and R. Herbei use the reverse martingale approach to implement the Markov Chain perfect sampling algorithm discussed above.

Application to the Bernoulli Factory problem

- The reverse time martingale approach is the first constructive and practical implementation of a general Bernoulli Factory
- In particular the Nacu-Peres polynomials can be utilised for $f(p) = \min\{1 - \varepsilon, Kp\}$ yielding a practical algorithm for the Metropolis accept-reject step in the discussed scenarios (and many others, see e.g. work by R. Herbei and M. Berliner)
- J. Flegal and R. Herbei use the reverse martingale approach to implement the Markov Chain perfect sampling algorithm discussed above.

Barkers Algorithm

- Recall the Metropolis algorithm:
- when constructing the Metropolis transition kernel P, in order to sample from π we propose from $q(x,y)$ and accept with probability

$$\alpha_M(x,y) = \min\{1, \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}\}.$$

- The design is such that for every x, y we obtain detailed balance

$$\pi(x)p(x,y) = \pi(y)p(y,x).$$

- Other choices of the acceptance function can yield detailed balance too.
- The Barkers acceptance rate is

$$\alpha_B(x,y) = \frac{\pi(y)q(y,x)}{\pi(y)q(y,x) + \pi(x)q(x,y)}.$$

- And it yields detailed balance too!

Barkers Algorithm

- Recall the Metropolis algorithm:
- when constructing the Metropolis transition kernel P, in order to sample from π we propose from $q(x,y)$ and accept with probability

$$\alpha_M(x,y) = \min\{1, \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}\}.$$

- The design is such that for every x, y we obtain detailed balance

$$\pi(x)p(x,y) = \pi(y)p(y,x).$$

- Other choices of the acceptance function can yield detailed balance too.
- The Barkers acceptance rate is

$$\alpha_B(x,y) = \frac{\pi(y)q(y,x)}{\pi(y)q(y,x) + \pi(x)q(x,y)}.$$

- And it yields detailed balance too!

Barkers Algorithm

- Recall the Metropolis algorithm:
- when constructing the Metropolis transition kernel P, in order to sample from π we propose from $q(x,y)$ and accept with probability

$$\alpha_M(x,y) = \min\{1, \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}\}.$$

- The design is such that for every x, y we obtain detailed balance

$$\pi(x)p(x,y) = \pi(y)p(y,x).$$

- Other choices of the acceptance function can yield detailed balance too.
- The Barkers acceptance rate is

$$\alpha_B(x,y) = \frac{\pi(y)q(y,x)}{\pi(y)q(y,x) + \pi(x)q(x,y)}.$$

- And it yields detailed balance too!

Barkers Algorithm

- Recall the Metropolis algorithm:
- when constructing the Metropolis transition kernel P, in order to sample from π we propose from $q(x,y)$ and accept with probability

$$\alpha_M(x,y) = \min\{1, \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}\}.$$

- The design is such that for every x, y we obtain detailed balance

$$\pi(x)p(x,y) = \pi(y)p(y,x).$$

- Other choices of the acceptance function can yield detailed balance too.
- The Barkers acceptance rate is

$$\alpha_B(x,y) = \frac{\pi(y)q(y,x)}{\pi(y)q(y,x) + \pi(x)q(x,y)}.$$

- And it yields detailed balance too!

Barkers Algorithm

- Recall the Metropolis algorithm:
- when constructing the Metropolis transition kernel P, in order to sample from π we propose from $q(x, y)$ and accept with probability

$$\alpha_M(x, y) = \min\{1, \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\}.$$

- The design is such that for every x, y we obtain detailed balance

$$\pi(x)p(x, y) = \pi(y)p(y, x).$$

- Other choices of the acceptance function can yield detailed balance too.
- The Barkers acceptance rate is

$$\alpha_B(x, y) = \frac{\pi(y)q(y, x)}{\pi(y)q(y, x) + \pi(x)q(x, y)}.$$

- And it yields detailed balance too!

Barkers Algorithm

- Recall the Metropolis algorithm:
- when constructing the Metropolis transition kernel P, in order to sample from π we propose from $q(x,y)$ and accept with probability

$$\alpha_M(x,y) = \min\{1, \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}\}.$$

- The design is such that for every x, y we obtain detailed balance

$$\pi(x)p(x,y) = \pi(y)p(y,x).$$

- Other choices of the acceptance function can yield detailed balance too.
- The Barkers acceptance rate is

$$\alpha_B(x,y) = \frac{\pi(y)q(y,x)}{\pi(y)q(y,x) + \pi(x)q(x,y)}.$$

- And it yields detailed balance too!

Barkers Algorithm - efficiency

- The Metropolis acceptance function is optimal with respect to Peskun ordering (explain)
- Suppose we estimate $\pi f := \int f(x)\pi(dx)$ by $\hat{\pi f} := \frac{1}{n}\sum_{i=1}^{n} f(X_i)$
- The under mild assumptions the Markov chain CLT holds:

$$\sqrt{n}(\pi f - \hat{\pi f}) \quad \to \quad N(0, \sigma_{as}(f, P)).$$

- By Peskun ordering

$$\sigma_{as}(f, P_{Barker}) \geq \sigma_{as}(f, P_{Metropolis})$$

- However:

$$\sigma^2_{as}(f, P_{Barker}) \leq 2\sigma^2_{as}(f, P_{Metropolis}) + \sigma^2_{as}(f, \pi)$$

(KL, GO Roberts, 2013)
- So Barkers is not that much worse than Metropolis!

Barkers Algorithm - efficiency

- The Metropolis acceptance function is optimal with respect to Peskun ordering (explain)
- Suppose we estimate $\pi f := \int f(x)\pi(dx)$ by $\hat{\pi f} := \frac{1}{n}\sum_{i=1}^{n} f(X_i)$
- The under mild assumptions the Markov chain CLT holds:

$$\sqrt{n}(\pi f - \hat{\pi f}) \quad \to \quad N(0, \sigma_{as}(f, P)).$$

- By Peskun ordering

$$\sigma_{as}(f, P_{Barker}) \geq \sigma_{as}(f, P_{Metropolis})$$

- However:

$$\sigma_{as}^2(f, P_{Barker}) \leq 2\sigma_{as}^2(f, P_{Metropolis}) + \sigma_{as}^2(f, \pi)$$

(KL, GO Roberts, 2013)

- So Barkers is not that much worse than Metropolis!

Barkers Algorithm - efficiency

- The Metropolis acceptance function is optimal with respect to Peskun ordering (explain)
- Suppose we estimate $\pi f := \int f(x)\pi(dx)$ by $\hat{\pi f} := \frac{1}{n}\sum_{i=1}^{n} f(X_i)$
- The under mild assumptions the Markov chain CLT holds:

$$\sqrt{n}(\pi f - \hat{\pi f}) \quad \to \quad N(0, \sigma_{as}(f, P)).$$

- By Peskun ordering

$$\sigma_{as}(f, P_{Barker}) \geq \sigma_{as}(f, P_{Metropolis})$$

- However:

$$\sigma_{as}^2(f, P_{Barker}) \leq 2\sigma_{as}^2(f, P_{Metropolis}) + \sigma_{as}^2(f, \pi)$$

(KL, GO Roberts, 2013)
- So Barkers is not that much worse than Metropolis!

Barkers Algorithm - efficiency

- The Metropolis acceptance function is optimal with respect to Peskun ordering (explain)
- Suppose we estimate $\pi f := \int f(x)\pi(dx)$ by $\hat{\pi f} := \frac{1}{n}\sum_{i=1}^{n} f(X_i)$
- The under mild assumptions the Markov chain CLT holds:

$$\sqrt{n}(\pi f - \hat{\pi f}) \quad \to \quad N(0, \sigma_{as}(f, P)).$$

- By Peskun ordering

$$\sigma_{as}(f, P_{Barker}) \geq \sigma_{as}(f, P_{Metropolis})$$

- However:

$$\sigma_{as}^2(f, P_{Barker}) \leq 2\sigma_{as}^2(f, P_{Metropolis}) + \sigma_{as}^2(f, \pi)$$

(KL, GO Roberts, 2013)
- So Barkers is not that much worse than Metropolis!

Barkers Algorithm - efficiency

- The Metropolis acceptance function is optimal with respect to Peskun ordering (explain)
- Suppose we estimate $\pi f := \int f(x)\pi(dx)$ by $\hat{\pi f} := \frac{1}{n}\sum_{i=1}^{n} f(X_i)$
- The under mild assumptions the Markov chain CLT holds:

$$\sqrt{n}(\pi f - \hat{\pi f}) \quad \to \quad N(0, \sigma_{as}(f, P)).$$

- By Peskun ordering

$$\sigma_{as}(f, P_{Barker}) \geq \sigma_{as}(f, P_{Metropolis})$$

- However:

$$\sigma_{as}^2(f, P_{Barker}) \leq 2\sigma_{as}^2(f, P_{Metropolis}) + \sigma_{as}^2(f, \pi)$$

(KL, GO Roberts, 2013)

- So Barkers is not that much worse than Metropolis!

Barkers Algorithm - efficiency

- The Metropolis acceptance function is optimal with respect to Peskun ordering (explain)
- Suppose we estimate $\pi f := \int f(x)\pi(dx)$ by $\hat{\pi f} := \frac{1}{n}\sum_{i=1}^{n} f(X_i)$
- The under mild assumptions the Markov chain CLT holds:

$$\sqrt{n}(\pi f - \hat{\pi f}) \quad \to \quad N(0, \sigma_{as}(f, P)).$$

- By Peskun ordering

$$\sigma_{as}(f, P_{Barker}) \geq \sigma_{as}(f, P_{Metropolis})$$

- However:

$$\sigma_{as}^2(f, P_{Barker}) \leq 2\sigma_{as}^2(f, P_{Metropolis}) + \sigma_{as}^2(f, \pi)$$

(KL, GO Roberts, 2013)

- So Barkers is not that much worse than Metropolis!

Barkers and the Bernoulli Factory

- In the scenarios where w need Bernoulli Factory to execute the Metropolis acceptance rate, we typically can also write the Barkers acceptance rate in the form of

$$\alpha_B(x,y) = \frac{Kq}{Mp + Kq},$$

- where K and M are known constants and p and q re probabilities that we can sample.
- Obtaining an event of probability $\alpha_B(x,y)$ may be more efficient by the following algorithm:

Barkers and the Bernoulli Factory

- In the scenarios where w need Bernoulli Factory to execute the Metropolis acceptance rate, we typically can also write the Barkers acceptance rate in the form of
$$\alpha_B(x,y) = \frac{Kq}{Mp + Kq},$$
- where K and M are known constants and p and q re probabilities that we can sample.
- Obtaining an event of probability $\alpha_B(x,y)$ may be more efficient by the following algorithm:

Barkers and the Bernoulli Factory

- In the scenarios where w need Bernoulli Factory to execute the Metropolis acceptance rate, we typically can also write the Barkers acceptance rate in the form of
$$\alpha_B(x,y) = \frac{Kq}{Mp + Kq},$$
- where K and M are known constants and p and q re probabilities that we can sample.
- Obtaining an event of probability $\alpha_B(x,y)$ may be more efficient by the following algorithm:

The two coin algorithm

- Assume there is a black box generating $p-$coin and another black box generating $q-$coins.
- Assume p and q are unknown and, for known $K, M.$, we are interested to obtain an event of probability

$$\frac{Kq}{Mp + Kq} = \frac{\frac{K}{K+M}q}{\frac{M}{K+M}p + \frac{K}{K+M}q}$$

- **Two coin algorithm**
  (1) draw $C \sim \frac{K}{K+M}-$coin,
  (2) if $C = 1$ draw $X \sim q-$coin,
  if $X = 1$, output 1 and STOP
  if $X = 0$, GOTO (1).
  (3) if $C = 0$ draw $X \sim p-$coin,
  if $X = 1$, output 0 and STOP
  if $X = 0$, GOTO (1).
- The number of iterations N needed by the algorithm for a single output has geometric distribution with parameter $\frac{M}{K+M}p + \frac{K}{K+M}q$.

The two coin algorithm

- Assume there is a black box generating $p-$coin and another black box generating $q-$coins.
- Assume p and q are unknown and, for known $K, M.$, we are interested to obtain an event of probability

$$\frac{Kq}{Mp + Kq} = \frac{\frac{K}{K+M}q}{\frac{M}{K+M}p + \frac{K}{K+M}q}$$

- **Two coin algorithm**
  (1) draw $C \sim \frac{K}{K+M}-$coin,
  (2) if $C = 1$ draw $X \sim q-$coin,
  if $X = 1$, output 1 and STOP
  if $X = 0$, GOTO (1).
  (3) if $C = 0$ draw $X \sim p-$coin,
  if $X = 1$, output 0 and STOP
  if $X = 0$, GOTO (1).
- The number of iterations N needed by the algorithm for a single output has geometric distribution with parameter $\frac{M}{K+M}p + \frac{K}{K+M}q$.

The two coin algorithm

- Assume there is a black box generating $p-$coin and another black box generating $q-$coins.
- Assume p and q are unknown and, for known $K, M.$, we are interested to obtain an event of probability

$$\frac{Kq}{Mp+Kq} = \frac{\frac{K}{K+M}q}{\frac{M}{K+M}p + \frac{K}{K+M}q}$$

- **Two coin algorithm**
  (1) draw $C \sim \frac{K}{K+M}-$coin,
  (2) if $C = 1$ draw $X \sim q-$coin,
  if $X = 1$, output 1 and STOP
  if $X = 0$, GOTO (1).
  (3) if $C = 0$ draw $X \sim p-$coin,
  if $X = 1$, output 0 and STOP
  if $X = 0$, GOTO (1).
- The number of iterations N needed by the algorithm for a single output has geometric distribution with parameter $\frac{M}{K+M}p + \frac{K}{K+M}q.$

The two coin algorithm

- Assume there is a black box generating $p-$coin and another black box generating $q-$coins.
- Assume p and q are unknown and, for known $K, M.$, we are interested to obtain an event of probability

$$\frac{Kq}{Mp + Kq} = \frac{\frac{K}{K+M}q}{\frac{M}{K+M}p + \frac{K}{K+M}q}$$

- **Two coin algorithm**
  (1) draw $C \sim \frac{K}{K+M}-$coin,
  (2) if $C = 1$ draw $X \sim q-$coin,
  if $X = 1$, output 1 and STOP
  if $X = 0$, GOTO (1).
  (3) if $C = 0$ draw $X \sim p-$coin,
  if $X = 1$, output 0 and STOP
  if $X = 0$, GOTO (1).
- The number of iterations N needed by the algorithm for a single output has geometric distribution with parameter $\frac{M}{K+M}p + \frac{K}{K+M}q$.

Some References

- 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.
- F. Gonçalves, G.O. Roberts, K. Latuszynski. Exact MCMC inference for jump diffusion models with stochastic jump rate, *in preparation.*
- M.S. Keane and G.L. O'Brien. A Bernoulli factory. *ACM Transactions on Modelling and Computer Simulation (TOMACS),* 4(2):213–219, 1994.
- K. Latuszynski, I. Kosmidis, O. Papaspiliopoulos, G.O. Roberts. Simulating events of unknown probabilities via reverse time martingales. *Random Structures and Algorithms*, 38(4): 441–452, 2011.
- K. Latuszynski, J. Palczewski, G.O. Roberts. Exact Inference for a Markov switching diffusion model with discretely observed data, *in preparation.*
- S. Nacu and Y. Peres. Fast simulation of new coins from old. *Annals of Applied Probability,* 15(1):93–115, 2005.
- Y. Peres. Iterating von Neumann's procedure for extracting random bits. *Annals of Statistics,* 20(1): 590–597, 1992.