Exact “combinatorial” simulation of continuous random variables

Philippe Duchon (LaBRI, U. Bordeaux)

Séminaire Flajolet, April 2, 2015

Simulating random variables

- All kinds of simulation problems require the use of random numbers

Simulating random variables

- All kinds of simulation problems require the use of random numbers
- Many classical distributions, either discrete (uniform, geometric, Poisson...) or continuous (uniform, exponential, normal...)

Simulating random variables

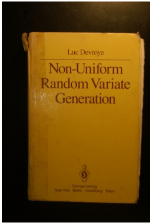
- All kinds of simulation problems require the use of random numbers
- Many classical distributions, either discrete (uniform, geometric, Poisson. . .) or continuous (uniform, exponential, normal. . .)
- (Almost ?) all programming languages provide access to pseudorandom numbers, either discrete (uniform integers over $[\![a, b]\!]$) or continuous (uniform over $[0, 1]$))

Simulating random variables

- All kinds of simulation problems require the use of random numbers
- Many classical distributions, either discrete (uniform, geometric, Poisson...) or continuous (uniform, exponential, normal...)
- (Almost ?) all programming languages provide access to pseudorandom numbers, either discrete (uniform integers over $[\![a, b]\!]$) or continuous (uniform over $[0, 1]$))
- **Exact** simulation algorithms are known for many distributions, usually assuming *exact computations over the reals*

Simulating random variables

- All kinds of simulation problems require the use of random numbers
- Many classical distributions, either discrete (uniform, geometric, Poisson. . .) or continuous (uniform, exponential, normal. . .)
- (Almost ?) all programming languages provide access to pseudorandom numbers, either discrete (uniform integers over $[[a, b]]$) or continuous (uniform over $[0, 1]$))
- **Exact** simulation algorithms are known for many distributions, usually assuming *exact computations over the reals*
- **The** reference : Devroye (1986)



Basic simulation tricks

- **Distribution function inversion :** if U is uniform and $F(x)$ is the (continuous, strictly increasing) distribution function ($F(x) = \mathbb{P}(X \leq x)$) for some distribution, $X = F^{-1}(U)$ has repartition function F

Basic simulation tricks

- **Distribution function inversion :** if U is uniform and $F(x)$ is the (continuous, strictly increasing) distribution function ($F(x) = \mathbb{P}(X \leq x)$) for some distribution, $X = F^{-1}(U)$ has repartition function F
- **Rejection :** if g is the density of some distribution (that one knows how to simulate), f is some other density with $f(x) \leq c.g(x)$ for some c and all x, the following rejection algorithm loops, on average, $1/c$ times, and simulates density f :
  - draw X (g-distributed)
  - with probability $f(X)/(c.g(X))$, output X ; otherwise, restart

Basic simulation tricks

- **Distribution function inversion :** if U is uniform and $F(x)$ is the (continuous, strictly increasing) distribution function ($F(x) = \mathbb{P}(X \leq x)$) for some distribution, $X = F^{-1}(U)$ has repartition function F
- **Rejection :** if g is the density of some distribution (that one knows how to simulate), f is some other density with $f(x) \leq c.g(x)$ for some c and all x, the following rejection algorithm loops, on average, $1/c$ times, and simulates density f :
  - draw X (g-distributed)
  - with probability $f(X)/(c.g(X))$, output X ; otherwise, restart
- Rejection can be used when densities are only *proportional* to functions f and g with, say, $f \leq g$, without identifying/computing the multiplicative constant

Typical basic examples

- If U is uniform over $[0,1]$, $-\ln(1-U)$ is exponentially distributed (distribution function inversion)

Typical basic examples

- If U is uniform over $[0,1]$, $-\ln(1-U)$ is exponentially distributed (distribution function inversion)
- (Kahn 1954 ; rejection) For the (absolute value of) a normal variable :
  - draw E and X, independent exponentials
  - if $2E \geq (X-1)^2$, return X ; otherwise, restart

Typical basic examples

- If U is uniform over $[0, 1]$, $-\ln(1 - U)$ is exponentially distributed (distribution function inversion)
- (Kahn 1954 ; rejection) For the (absolute value of) a normal variable :
  - draw E and X, independent exponentials
  - if $2E \geq (X - 1)^2$, return X ; otherwise, restart

  (conditioned on X, the acceptance probability is $\exp(-(X-1)^2/2) = \exp(-x^2/2)/\exp(-x))$

- All the previous techniques require
    1. a generator of independent, uniform variables on $[0, 1]$
    2. exact evaluation of transcendental functions and constants, integrals, etc.

- All the previous techniques require
  1. a generator of independent, uniform variables on $[0, 1]$
  2. exact evaluation of transcendental functions and constants, integrals, etc.
- For many *discrete* distributions, the *Buffon machines* of **[Flajolet, Pelletier, Soria 2011]** allow to only use
  - **flip()** (Bernoulli with parameter $1/2$; "coin flips")
  - **Bern[p]()** (Bernoulli with parameter p, for unknown parameters $p \in (0, 1)$)
  - basic integer arithmetic and bookkeeping (small counters)

- All the previous techniques require
  1. a generator of independent, uniform variables on $[0, 1]$
  2. exact evaluation of transcendental functions and constants, integrals, etc.
- For many *discrete* distributions, the *Buffon machines* of **[Flajolet, Pelletier, Soria 2011]** allow to only use
  - **flip()** (Bernoulli with parameter $1/2$; "coin flips")
  - **Bern[p]()** (Bernoulli with parameter p, for unknown parameters $p \in (0, 1)$)
  - basic integer arithmetic and bookkeeping (small counters)
- Can we do the same for a variety of continuous distributions ? In a more or less systematic way ?

Precursor : von Neumann's algorithm

- **J. von Neumann, 1951** "Various techniques used in connection with random digits" (3 pages)

Precursor : von Neumann's algorithm

- **J. von Neumann, 1951** "Various techniques used in connection with random digits" (3 pages)

- describes an **exact** algorithm for the exponential distribution, using only
  - independent uniforms on $[0, 1]$
  - comparisons of reals
  - (small) integer counters

The algorithm

1. Initialize counter K to 0
2. Draw a sequence $X_1, X_2, \dots X_n$ of independent uniforms on $[0, 1]$, until the **first ascent** $(X_n > X_{n-1})$
3. If n is odd : failure ; increment failure counter K, and go to 2.
4. (Otherwise) n is even : success, return $K + X_1$

The algorithm

1. Initialize counter K to 0
2. Draw a sequence $X_1, X_2, \dots X_n$ of independent uniforms on $[0, 1]$, until the **first ascent** ($X_n > X_{n-1}$)
3. If n is odd : failure ; increment failure counter K, and go to 2.
4. (Otherwise) n is even : success, return $K + X_1$

Proposition (von Neumann) : This algorithm terminates with probability 1, and its output follows the exponential distribution (density $f(x) = \exp(-x)\mathbf{1}_{x>0}$). The expected number of uniforms used is $\frac{e+e^2}{e-1} \simeq 5.88$.

Running the algorithm : an example

- Uniform sequence : $0.78, 0.04, 0.92, 0.01, 0.83, 0.22 \ldots$

Running the algorithm : an example

- Uniform sequence : $0.78, 0.04, 0.92, 0.01, 0.83, 0.22\ldots$
- First attempt : $0.78 > 0.04$

Running the algorithm : an example

- Uniform sequence : $0.78, 0.04, 0.92, 0.01, 0.83, 0.22 \ldots$
- First attempt : $0.78 > 0.04 < 0.92$: odd length series, restart ($K = 1$)

Running the algorithm : an example

- Uniform sequence : $0.78, 0.04, 0.92, 0.01, 0.83, 0.22 \ldots$
- First attempt : $0.78 > 0.04 < 0.92$: odd length series, restart ($K = 1$)
- Second attempt : $0.01 < 0.83$: even length, stop

Running the algorithm : an example

- Uniform sequence : $0.78, 0.04, 0.92, 0.01, 0.83, 0.22 \dots$
- First attempt : $0.78 > 0.04 < 0.92$: odd length series, restart ($K = 1$)
- Second attempt : $0.01 < 0.83$: even length, stop
- The output value is $1 + 0.01 = 1.01$

Proof of the algorithm (sketch)

- The probability that, in an infinite sequence of iid uniforms, the first ascent occurs with the n-th element is $1/(n-1)! - 1/n!$

Proof of the algorithm (sketch)

- The probability that, in an infinite sequence of iid uniforms, the first ascent occurs with the n-th element is $1/(n-1)! - 1/n!$
- Summing, the probability of the first ascent being in an odd position (restarting) is

$$p = \sum_{k \geq 0} \frac{1}{(2k)!} - \frac{1}{(2k+1)!} = e^{-1}$$

Proof of the algorithm (sketch)

- The probability that, in an infinite sequence of iid uniforms, the first ascent occurs with the n-th element is $1/(n-1)! - 1/n!$
- Summing, the probability of the first ascent being in an odd position (restarting) is

$$p = \sum_{k \geq 0} \frac{1}{(2k)!} - \frac{1}{(2k+1)!} = e^{-1}$$

- For $x \in [0, 1]$ and $n \geq 2$, the probability of having $X_1 \leq x$ **and** first ascent on the n-th elements, is $x^{n-1}/(n-1)! - x^n/n!$

Proof of the algorithm (sketch)

- The probability that, in an infinite sequence of iid uniforms, the first ascent occurs with the n-th element is $1/(n-1)! - 1/n!$
- Summing, the probability of the first ascent being in an odd position (restarting) is

$$p = \sum_{k \geq 0} \frac{1}{(2k)!} - \frac{1}{(2k+1)!} = e^{-1}$$

- For $x \in [0, 1]$ and $n \geq 2$, the probability of having $X_1 \leq x$ **and** first ascent on the n-th elements, is $x^{n-1}/(n-1)! - x^n/n!$
- Summing again : the probability of "success" with $X_1 \leq x$, is $1 - e^{-x}$ (the distribution function for an exponential on $[0, 1]$)

Proof of the algorithm (sketch)

- The probability that, in an infinite sequence of iid uniforms, the first ascent occurs with the n-th element is $1/(n-1)! - 1/n!$
- Summing, the probability of the first ascent being in an odd position (restarting) is

$$p = \sum_{k \geq 0} \frac{1}{(2k)!} - \frac{1}{(2k+1)!} = e^{-1}$$

- For $x \in [0,1]$ and $n \geq 2$, the probability of having $X_1 \leq x$ **and** first ascent on the n-th elements, is $x^{n-1}/(n-1)! - x^n/n!$
- Summing again : the probability of "success" with $X_1 \leq x$, is $1 - e^{-x}$ (the distribution function for an exponential on $[0,1]$)
- For the algorithm : the final value of K follows the geometric distribution with parameter $1 - e^{-1}$, and the (independent) value of X_1 conditioned on success is distributed as an exponential, conditioned on being ≤ 1 ; the sum is exponentially distributed.

“Combinatorial simulation” of continuous distributions

- What we would like to obtain : **exact** simulation algorithms for a large enough family of continuous probability distributions, **not** requiring the use of “complex” operations over the reals

“Combinatorial simulation” of continuous distributions

- What we would like to obtain : **exact** simulation algorithms for a large enough family of continuous probability distributions, **not** requiring the use of “complex” operations over the reals
- Certainly no evaluations of transcendental functions ; if possible, only basic arithmetic operations

“Combinatorial simulation” of continuous distributions

- What we would like to obtain : **exact** simulation algorithms for a large enough family of continuous probability distributions, **not** requiring the use of “complex” operations over the reals
- Certainly no evaluations of transcendental functions ; if possible, only basic arithmetic operations
- Ideally : algorithms that could be “humanly” run by treating reals as infinite digit strings (and only using finite prefixes) - no multiplications other than by powers of 2

“Combinatorial simulation” of continuous distributions

- What we would like to obtain : **exact** simulation algorithms for a large enough family of continuous probability distributions, **not** requiring the use of “complex” operations over the reals
- Certainly no evaluations of transcendental functions ; if possible, only basic arithmetic operations
- Ideally : algorithms that could be “humanly” run by treating reals as infinite digit strings (and only using finite prefixes) - no multiplications other than by powers of 2
- If we allow arbitrary products, then Kahn’s method (and von Neumann’s algorithm for the exponential) shows that the normal distribution admits such a restricted simulation algorithm.

“Combinatorial simulation” of continuous distributions

- What we would like to obtain : **exact** simulation algorithms for a large enough family of continuous probability distributions, **not** requiring the use of “complex” operations over the reals
- Certainly no evaluations of transcendental functions ; if possible, only basic arithmetic operations
- Ideally : algorithms that could be “humanly” run by treating reals as infinite digit strings (and only using finite prefixes) - no multiplications other than by powers of 2
- If we allow arbitrary products, then Kahn’s method (and von Neumann’s algorithm for the exponential) shows that the normal distribution admits such a restricted simulation algorithm.
- **[Karney, 2013]** describes such a product-less algorithm.

Extending the method

- “the above method can be modified to yield a distribution satisfying any first-order differential equation” (von Neumann)

Extending the method

- "the above method can be modified to yield a distribution satisfying any first-order differential equation" (von Neumann)
- Natural interpretation : assume the target *density* satisfies a *linear* first-order differential equation $y'(t) = g(t).y(t)$, for some given function g

Extending the method

- "the above method can be modified to yield a distribution satisfying any first-order differential equation" (von Neumann)
- Natural interpretation : assume the target *density* satisfies a *linear* first-order differential equation $y'(t) = g(t).y(t)$, for some given function g
- (This includes the density for the normal distribution : $y'(t) = -t.y(t)$)

Extending the method

- "the above method can be modified to yield a distribution satisfying any first-order differential equation" (von Neumann)
- Natural interpretation : assume the target *density* satisfies a *linear* first-order differential equation $y'(t) = g(t).y(t)$, for some given function g
- (This includes the density for the normal distribution : $y'(t) = -t.y(t)$)
- This is essentially the interpretation of **[Forsythe, 1972]** ; *but* the described method involves computing integrals based on solving the equation (to tabulate the probability that the target random variable takes values in a collection of disjoint intervals)

Extending the method

- "the above method can be modified to yield a distribution satisfying any first-order differential equation" (von Neumann)
- Natural interpretation : assume the target *density* satisfies a *linear* first-order differential equation $y'(t) = g(t).y(t)$, for some given function g
- (This includes the density for the normal distribution : $y'(t) = -t.y(t)$)
- This is essentially the interpretation of **[Forsythe, 1972]** ; *but* the described method involves computing integrals based on solving the equation (to tabulate the probability that the target random variable takes values in a collection of disjoint intervals)
- Today : description of an exact simulation method that is slightly more involved, but does not require the evaluation of any integrals or transcendental functions not in g.

Main result

- Suppose our target distribution (over the positive reals) has a density f, satisfying differential equation $y'(t) = -g(t).y(t)$ for some given function g (at most one solution is a probability density)

Main result

- Suppose our target distribution (over the positive reals) has a density f, satisfying differential equation $y'(t) = -g(t).y(t)$ for some given function g (at most one solution is a probability density)
- Assume g satisfies some "quadrant" condition : there should exist some $a \geq 0$ with $m = g(a) > 0$, such that
  - $g(t) \leq g(a)$ if $t \leq a$
  - $g(t) \geq g(a)$ if $t \geq a$

Main result

- Suppose our target distribution (over the positive reals) has a density f, satisfying differential equation $y'(t) = -g(t).y(t)$ for some given function g (at most one solution is a probability density)
- Assume g satisfies some "quadrant" condition : there should exist some $a \geq 0$ with $m = g(a) > 0$, such that
  - $g(t) \leq g(a)$ if $t \leq a$
  - $g(t) \geq g(a)$ if $t \geq a$
- Assume we are given g (as a "black box" function), a, and some (black box) "upper bounding" function $h(t, u)$ such that, for any $t \leq u$, $h(t, u) \geq \sup_{t \leq x \leq u} g(x)$

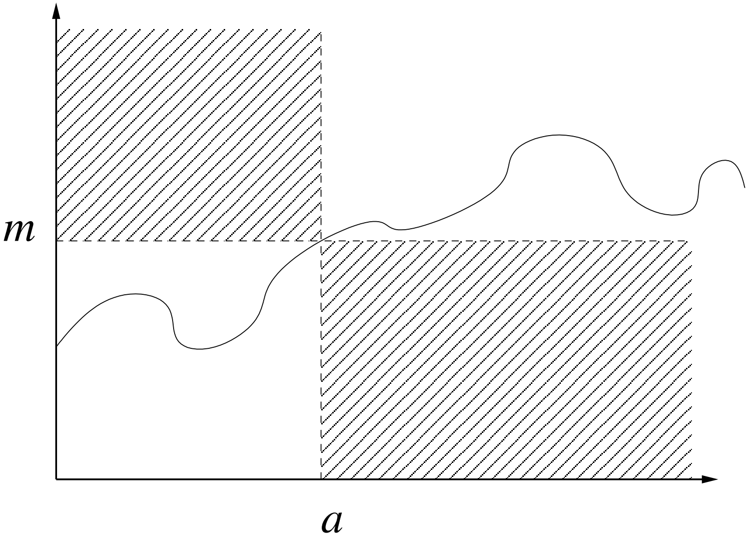
Main result

- Suppose our target distribution (over the positive reals) has a density f, satisfying differential equation $y'(t) = -g(t).y(t)$ for some given function g (at most one solution is a probability density)
- Assume g satisfies some "quadrant" condition : there should exist some $a \geq 0$ with $m = g(a) > 0$, such that
  - $g(t) \leq g(a)$ if $t \leq a$
  - $g(t) \geq g(a)$ if $t \geq a$
- Assume we are given g (as a "black box" function), a, and some (black box) "upper bounding" function $h(t, u)$ such that, for any $t \leq u$, $h(t, u) \geq \sup_{t \leq x \leq u} g(x)$
- **Then** we provide an exact simulation algorithm, using only uniform reals, additions, division by m, comparisons, and evaluations of g and h

Main result

- Suppose our target distribution (over the positive reals) has a density f, satisfying differential equation $y'(t) = -g(t).y(t)$ for some given function g (at most one solution is a probability density)
- Assume g satisfies some "quadrant" condition : there should exist some $a \geq 0$ with $m = g(a) > 0$, such that
  - $g(t) \leq g(a)$ if $t \leq a$
  - $g(t) \geq g(a)$ if $t \geq a$
- Assume we are given g (as a "black box" function), a, and some (black box) "upper bounding" function $h(t, u)$ such that, for any $t \leq u$, $h(t, u) \geq \sup_{t \leq x \leq u} g(x)$
- **Then** we provide an exact simulation algorithm, using only uniform reals, additions, division by m, comparisons, and evaluations of g and h
- (Notice that the conditions reduce to g as a black box if g is known to be nondecreasing)

The "quadrant condition"



The differential equation

- The differential equation has solutions $f(t) = f(t_0)e^{-\int_{t_0}^{t} g(u)du}$; initial condition $f(t_0)$ would be determined by condition $\int_0^\infty f(t)dt = 1$ (but we will be proceeding by rejection and thus need not compute them)

The differential equation

- The differential equation has solutions $f(t) = f(t_0)e^{-\int_{t_0}^{t} g(u)du}$; initial condition $f(t_0)$ would be determined by condition $\int_0^\infty f(t)dt = 1$ (but we will be proceeding by rejection and thus need not compute them)
- Taking $t_0 = a$ for the initial condition, the "quadrant" condition implies that the density is upper bounded by the solution to $y'(t) = -m.y(t)$ with the same initial condition : for all $t \geq 0$,
$$f(t) \leq f(a)e^{-m(t-a)}$$

The differential equation

- The differential equation has solutions $f(t) = f(t_0)e^{-\int_{t_0}^{t} g(u)du}$; initial condition $f(t_0)$ would be determined by condition $\int_0^\infty f(t)dt = 1$ (but we will be proceeding by rejection and thus need not compute them)
- Taking $t_0 = a$ for the initial condition, the "quadrant" condition implies that the density is upper bounded by the solution to $y'(t) = -m.y(t)$ with the same initial condition : for all $t \geq 0$,
$$f(t) \leq f(a)e^{-m(t-a)}$$
- We could try a rejection scheme : simulate an exponential E (using the von Neumann algorithm) and set $X = E/m$, then return X with appropriate probability, or restart.

The differential equation

- The differential equation has solutions $f(t) = f(t_0)e^{-\int_{t_0}^{t} g(u)du}$; initial condition $f(t_0)$ would be determined by condition $\int_0^\infty f(t)dt = 1$ (but we will be proceeding by rejection and thus need not compute them)
- Taking $t_0 = a$ for the initial condition, the "quadrant" condition implies that the density is upper bounded by the solution to $y'(t) = -m.y(t)$ with the same initial condition : for all $t \geq 0$,
$$f(t) \leq f(a)e^{-m(t-a)}$$
- We could try a rejection scheme : simulate an exponential E (using the von Neumann algorithm) and set $X = E/m$, then return X with appropriate probability, or restart.
- **Only**, the acceptance probability is not something we are allowed to compute :
$$\exp\left(-\int_a^X g(t)dt + m(X-a)\right) = \exp\left(-\int_a^X (g(t)-m)dt\right)$$

Digression : "Buffon generator" for $x \mapsto e^{-x}$

(**Flajolet, Pelletier, Soria 2011**)

- **Hypothesis :** we can draw uniforms, and have access to a Bernoulli generator with parameter p, for some *unknown* $0 < p < 1$, **Bern()** (*i.e.*, **Bern()** returns 1 with probability p and 0 with probability $1 - p$ on each call, with calls being independent)

Digression : “Buffon generator” for $x \mapsto e^{-x}$

(**Flajolet, Pelletier, Soria 2011**)

- **Hypothesis :** we can draw uniforms, and have access to a Bernoulli generator with parameter p, for some *unknown* $0 < p < 1$, **Bern()** (*i.e.*, **Bern()** returns 1 with probability p and 0 with probability $1 - p$ on each call, with calls being independent)
- **Then** we have a von Neumann-like algorithm for a Bernoulli with parameter e^{-p}

- Draw a sequence of independent **pairs** (X_i, B_i) with X_i uniform on $[0, 1]$, and B_i an independent Bernoulli with parameter p
- Stop at the first n such that $B_n = 0$ or $X_{n-1} < X_n$ (Bernoulli fails, or ascent in the X sequence)
- Return 1 if n is odd, 0 if n is even

- Draw a sequence of independent **pairs** (X_i, B_i) with X_i uniform on $[0, 1]$, and B_i an independent Bernoulli with parameter p
- Stop at the first n such that $B_n = 0$ or $X_{n-1} < X_n$ (Bernoulli fails, or ascent in the X sequence)
- Return 1 if n is odd, 0 if n is even

(proof along the same line as for von Neumann's algorithm, with powers of p addded, hence the e^{-p} instead of e^{-1})

Back to the simulation algorithm

- We need to "accept with probability $\exp(-I)$", *i.e.* draw a Bernoulli whose parameter is the exponential of some integral.

Back to the simulation algorithm

- We need to "accept with probability $\exp(-I)$", *i.e.* draw a Bernoulli whose parameter is the exponential of some integral.
- Under suitable conditions, an integral can be interpreted as a probability for an easy-to-simulate event (that a random point falls into some domain)

Back to the simulation algorithm

- We need to "accept with probability $\exp(-I)$", *i.e.* draw a Bernoulli whose parameter is the exponential of some integral.
- Under suitable conditions, an integral can be interpreted as a probability for an easy-to-simulate event (that a random point falls into some domain)
- If needed, the integral can be written as a sum of integrals on smaller intervals (and the exponential becomes a product of exponentials ; the Bernoulli variable becomes a product of Bernoulli variables).

Picking intervals

Assume $X > a$; the case $X < a$ is treated analogously)

- We need to split the interval $[a, X]$ into a number of smaller intervals $A_1, \ldots, A_K$; $A_i = [a_{i-1}, a_i]$.

Picking intervals

Assume $X > a$; the case $X < a$ is treated analogously)

- We need to split the interval $[a, X]$ into a number of smaller intervals $A_1, \ldots, A_K$; $A_i = [a_{i-1}, a_i]$.
- Set $a_0 = 0$.

Picking intervals

Assume $X > a$; the case $X < a$ is treated analogously)

- We need to split the interval $[a, X]$ into a number of smaller intervals $A_1, \ldots, A_K$; $A_i = [a_{i-1}, a_i]$.
- Set $a_0 = 0$.
- Assume a_i is known : compute $M = h(a_i, 1 + a_i)$. If $M \leq 1$, then set $a_{i+1} = 1 + a_i$, and repeat.

Picking intervals

Assume $X > a$; the case $X < a$ is treated analogously)

- We need to split the interval $[a, X]$ into a number of smaller intervals $A_1, \ldots, A_K$; $A_i = [a_{i-1}, a_i]$.
- Set $a_0 = 0$.
- Assume a_i is known : compute $M = h(a_i, 1 + a_i)$. If $M \leq 1$, then set $a_{i+1} = 1 + a_i$, and repeat.
- If $M > 1$, then let M' denote the smallest power of 2 larger than M, and, for each $1 \leq k \leq M'$, set $a_{i+k} = a_i + k/M'$ (M' intervals of length $1/M'$), and repeat

Picking intervals

Assume $X > a$; the case $X < a$ is treated analogously)

- We need to split the interval $[a, X]$ into a number of smaller intervals $A_1, \ldots, A_K$; $A_i = [a_{i-1}, a_i]$.
- Set $a_0 = 0$.
- Assume a_i is known : compute $M = h(a_i, 1 + a_i)$. If $M \leq 1$, then set $a_{i+1} = 1 + a_i$, and repeat.
- If $M > 1$, then let M' denote the smallest power of 2 larger than M, and, for each $1 \leq k \leq M'$, set $a_{i+k} = a_i + k/M'$ (M' intervals of length $1/M'$), and repeat
- Stop at the first K such that $a_k \geq X$; instead set $a_K = X$

Rejection probability

- Now the wanted integral is

$$\int_a^X (g(t) - m)dt = \sum_{i=0}^{K} \int_{A_i} (g(t) - m)dt = \sum_i P_i$$

Rejection probability

- Now the wanted integral is

$$\int_a^X (g(t) - m)dt = \sum_{i=0}^{K} \int_{A_i} (g(t) - m)dt = \sum_i P_i$$

- Each smaller integral can be interpreted as a probability, *i.e.* the probability that a uniform random point (X, Y) in the rectangle $A_i \times [0, 1/(a_i - a_{i-1})]$ (with area 1) satisfies $m \leq Y \leq g(X)$

Rejection probability

- Now the wanted integral is

$$\int_a^X (g(t) - m)dt = \sum_{i=0}^{K} \int_{A_i} (g(t) - m)dt = \sum_i P_i$$

- Each smaller integral can be interpreted as a probability, *i.e.* the probability that a uniform random point (X, Y) in the rectangle $A_i \times [0, 1/(a_i - a_{i-1})]$ (with area 1) satisfies $m \leq Y \leq g(X)$
- **Thus** we can apply the "exponential Buffon" construction to obtain a Bernoulli with parameter $\exp(-P_i)$

Rejection probability

- Now the wanted integral is

$$\int_a^X (g(t) - m)dt = \sum_{i=0}^{K} \int_{A_i} (g(t) - m)dt = \sum_i P_i$$

- Each smaller integral can be interpreted as a probability, *i.e.* the probability that a uniform random point (X, Y) in the rectangle $A_i \times [0, 1/(a_i - a_{i-1})]$ (with area 1) satisfies $m \leq Y \leq g(X)$
- **Thus** we can apply the "exponential Buffon" construction to obtain a Bernoulli with parameter $\exp(-P_i)$
- **and in turn**, obtain the wanted Bernoulli with parameter $\exp(-\sum_i P_i)$, by taking the product (conjunction) of each individual Bernoulli for each smaller interval : this completes the algorithm.

The case of the normal distribution

- Take the normal density, $f(t) = \sqrt{2\pi}^{-1} \exp(-x^2/2)$, as the target density.

The case of the normal distribution

- Take the normal density, $f(t) = \sqrt{2\pi}^{-1} \exp(-x^2/2)$, as the target density.
- (We simulate the absolute value, then add a random sign)

The case of the normal distribution

- Take the normal density, $f(t) = \sqrt{2\pi}^{-1} \exp(-x^2/2)$, as the target density.
- (We simulate the absolute value, then add a random sign)
- Differential equation : $y'(t) = -t.y(t)$, $g(t) = t$.

The case of the normal distribution

- Take the normal density, $f(t) = \sqrt{2\pi}^{-1} \exp(-x^2/2)$, as the target density.
- (We simulate the absolute value, then add a random sign)
- Differential equation : $y'(t) = -t.y(t)$, $g(t) = t$.
- g is increasing, any value of $a > 0$ will do ; $a = 1$ is Kahn's method (and minimizes the rejection probability)

The case of the normal distribution

- Take the normal density, $f(t) = \sqrt{2\pi}^{-1} \exp(-x^2/2)$, as the target density.
- (We simulate the absolute value, then add a random sign)
- Differential equation : $y'(t) = -t.y(t)$, $g(t) = t$.
- g is increasing, any value of $a > 0$ will do ; $a = 1$ is Kahn's method (and minimizes the rejection probability)
- The upper bounding function is naturally $h(t, u) = \max(t, u)$

The case of the normal distribution

- Take the normal density, $f(t) = \sqrt{2\pi}^{-1} \exp(-x^2/2)$, as the target density.
- (We simulate the absolute value, then add a random sign)
- Differential equation : $y'(t) = -t.y(t)$, $g(t) = t$.
- g is increasing, any value of $a > 0$ will do ; $a = 1$ is Kahn's method (and minimizes the rejection probability)
- The upper bounding function is naturally $h(t, u) = \max(t, u)$
- Strictly applying the previous interval description yields 2 intervals $[1, 3/2]$ and $[3/2, 2]$, then 4 intervals for each of $[2, 3]$ and $[3, 4]$, then 8 intervals for each of $[k, k+1]$ for $k = 4, 5, 6, 7$, and so on.

The case of the normal distribution

- Take the normal density, $f(t) = \sqrt{2\pi}^{-1} \exp(-x^2/2)$, as the target density.
- (We simulate the absolute value, then add a random sign)
- Differential equation : $y'(t) = -t.y(t)$, $g(t) = t$.
- g is increasing, any value of $a > 0$ will do ; $a = 1$ is Kahn's method (and minimizes the rejection probability)
- The upper bounding function is naturally $h(t, u) = \max(t, u)$
- Strictly applying the previous interval description yields 2 intervals $[1, 3/2]$ and $[3/2, 2]$, then 4 intervals for each of $[2, 3]$ and $[3, 4]$, then 8 intervals for each of $[k, k+1]$ for $k = 4, 5, 6, 7$, and so on.
- (In practice, large values of X are very likely to be rejected ; the rejection part should be run after each increment of the K counter for the exponential after $K = 1$, so as to allow early rejection)

Running the algorithm digit-by-digit

- The previous algorithms are very suitable to an adaptation to bit-by-bit computations

Running the algorithm digit-by-digit

- The previous algorithms are very suitable to an adaptation to bit-by-bit computations
- In any base B, uniforms over $[0, 1]$ have a B-ary development made of independent uniform $\{0, \dots, B-1\}$ digits

Running the algorithm digit-by-digit

- The previous algorithms are very suitable to an adaptation to bit-by-bit computations
- In any base B, uniforms over $[0, 1]$ have a B-ary development made of independent uniform $\{0, \ldots, B-1\}$ digits
- Comparisons of reals reduce to lexicographic order of strings

Running the algorithm digit-by-digit

- The previous algorithms are very suitable to an adaptation to bit-by-bit computations
- In any base B, uniforms over $[0, 1]$ have a B-ary development made of independent uniform $\{0, \dots, B-1\}$ digits
- Comparisons of reals reduce to lexicographic order of strings
- The algorithms can be adapted to bit-by-bit simulation : each uniform is only simulated up to the precision required by comparisons, and later completed as needed

Running the algorithm digit-by-digit

- The previous algorithms are very suitable to an adaptation to bit-by-bit computations
- In any base B, uniforms over $[0, 1]$ have a B-ary development made of independent uniform $\{0, \ldots, B-1\}$ digits
- Comparisons of reals reduce to lexicographic order of strings
- The algorithms can be adapted to bit-by-bit simulation : each uniform is only simulated up to the precision required by comparisons, and later completed as needed
- Such algorithms output a finite B-ary string, with the meaning "if more precision is needed, add random digits"

Running the algorithm digit-by-digit

- The previous algorithms are very suitable to an adaptation to bit-by-bit computations
- In any base B, uniforms over $[0, 1]$ have a B-ary development made of independent uniform $\{0, \ldots, B-1\}$ digits
- Comparisons of reals reduce to lexicographic order of strings
- The algorithms can be adapted to bit-by-bit simulation : each uniform is only simulated up to the precision required by comparisons, and later completed as needed
- Such algorithms output a finite B-ary string, with the meaning "if more precision is needed, add random digits"
- von Neumann's algorithm was analysed by **[Flajolet, Saheb, 1986]** ; uses on average $k + 5.72..$ bits to output k bits of the exponential random variable

Running the algorithm digit-by-digit

- The previous algorithms are very suitable to an adaptation to bit-by-bit computations
- In any base B, uniforms over $[0, 1]$ have a B-ary development made of independent uniform $\{0, \dots, B-1\}$ digits
- Comparisons of reals reduce to lexicographic order of strings
- The algorithms can be adapted to bit-by-bit simulation : each uniform is only simulated up to the precision required by comparisons, and later completed as needed
- Such algorithms output a finite B-ary string, with the meaning "if more precision is needed, add random digits"
- von Neumann's algorithm was analysed by **[Flajolet, Saheb, 1986]** ; uses on average $k + 5.72..$ bits to output k bits of the exponential random variable
- In our general differential equation algorithm, we need a bit more than just a black box function g (unless g is known to be increasing)

Conclusion

- We obtain exact, “von-Neumann-Buffon-like” algorithms for the simulation of a (not too well-defined) class of distributions that includes the normal distribution

Conclusion

- We obtain exact, "von-Neumann-Buffon-like" algorithms for the simulation of a (not too well-defined) class of distributions that includes the normal distribution
- For the normal distribution, this is very similar to Karney's algorithm, described at the digit level

Conclusion

- We obtain exact, "von-Neumann-Buffon-like" algorithms for the simulation of a (not too well-defined) class of distributions that includes the normal distribution
- For the normal distribution, this is very similar to Karney's algorithm, described at the digit level
- In the general case, this is very close to what Devroye described as "the von Neumann-Forsythe method"

Conclusion

- We obtain exact, "von-Neumann-Buffon-like" algorithms for the simulation of a (not too well-defined) class of distributions that includes the normal distribution
- For the normal distribution, this is very similar to Karney's algorithm, described at the digit level
- In the general case, this is very close to what Devroye described as "the von Neumann-Forsythe method"
- No analysis of the expected (bit) complexity yet (will depend on the quality of upper bound h in the general method)

Conclusion

- We obtain exact, "von-Neumann-Buffon-like" algorithms for the simulation of a (not too well-defined) class of distributions that includes the normal distribution
- For the normal distribution, this is very similar to Karney's algorithm, described at the digit level
- In the general case, this is very close to what Devroye described as "the von Neumann-Forsythe method"
- No analysis of the expected (bit) complexity yet (will depend on the quality of upper bound h in the general method)
- The method is unlikely to be competitive with numerical methods (possibly paired with certified floating point calculations), **unless** one needs very high precision on their random variables

Thank you for your attention