Review of "On Buffon Machines and Numbers" by P. Flajolet, M. Pelletier, M. Soria

Nicolas Blanchard

March 26, 2015

Extra credit toward the MPRI 2.15 class

Abstract

In their article, the authors describe a variety of simple and easily combinable procedures allowing to transform a λ Bernoulli generator ($\Gamma\mathrm{B}(\lambda)$) into a number of generators performing functions on λ or computing constants. We seek to prove some of their results and analyse the efficiency of the procedures on generators for $\frac{1}{\pi}$ and $\frac{\pi}{4}$.

1 Creating a $\Gamma\mathrm{B}(\frac{1}{\pi})$

1.1 The Procedure RAMA

In the paper the authors describe the following procedure :

Procedure RAMA(); {returns 1 with probability $1/\pi$}

S1. let $T := X_1 + X_2$, where $X_1, X_2 \in Geom(\frac{1}{4})$;

S2. with probability$\frac{5}{9}$ do $T := T + 1$;

S3. for j = 1, 2, 3 do

S4. draw a sequence of $2T$ coin flips;

if $(\Delta \equiv \#Heads - \#Tails) \neq 0$ return(0);

S5. return(1).

This is based on an identity of Ramanujan, which states that

$$\frac{1}{\pi} = \sum_{n=0}^{\infty} \binom{2n}{n}^3 \frac{6n+1}{2^{8n+4}}$$

We can first prove that Rama indeed computes the right term of the equality.

Let p_{T_n} be the probability that $T = n$ at S_3. As the probability of a sequence of $2n$ flips being balanced is $2^{-2n}\binom{2n}{n}$, the probality of Rama returning 1 is

$$\sum_{n=0}^{\infty} p_{T_n} 2^{-6n} \binom{2n}{n}^3$$

We would then expect p_{T_n} to be equal to $2^{-2n-4}(6n+1)$ for the procedure to compute $\frac{1}{\pi}$.
However, the probability of T being equal to n after S_1 is

$$\sum_{k=0}^{n} \Pr(X_1 = k) \times \Pr(X_2 = n-k)$$

$$\sum_{k=0}^{n} 2^{-2k}\frac{3}{4} \times 2^{-2n+2k}\frac{3}{4} = (n+1) \times 2^{-2n}\frac{9}{16}$$

And we get that

$$p_{T_n} = (n+1) \times 2^{-2n}\frac{9}{16} \times \frac{4}{9} + n \times 2^{-2n+2}\frac{9}{16} \times \frac{5}{9}$$

$$p_{T_n} = (n+1) \times 2^{-2n}\frac{1}{4} + n \times 2^{-2n}\frac{5}{4}$$

$$p_{T_n} = (6n+1) \times 2^{-2n}\frac{1}{4}$$

Hence we get a probability 4 times greater than expected. I could not check if the algorithm indeed outputs 1 with probability $\frac{1}{\pi}$ although it is quickly decreasing and we have that $p_{T_0} = \frac{1}{4}$ and $p_{T_1} = \frac{5}{16} + \frac{1}{64} = \frac{21}{64}$, and the probability of being accepted with $T \leq 1$ is then $\frac{1}{4} + \frac{21}{512} \approx 0,29$ with $\frac{1}{\pi} \approx 0,32$ so it isn't far. However we can quickly see that the Ramanujan identity isn't correct, as

$$\sum_{n=0}^{\infty} \binom{2n}{n}^3 \frac{6n+1}{2^{8n+4}} \leq \sum_{n=0}^{\infty} \frac{6n+1}{2^{2n+4}} \leq \frac{1}{12} + \frac{3}{8}\sum_{n=0}^{\infty} n \times 4^{-n}$$

And this gives us a total of at most $\frac{5}{36} \approx 0.14$ (as $\binom{2n}{n} \leq 2^{n-1}$ for all $n > 0$, we can actually get $\frac{13}{144} \approx 0.090$).
Which proves that the identity used was erroneous (as I couldn't find the identity in the litterature i couldn't do more), however if the identity gives $\frac{1}{4\pi} \approx 0.0796$ instead then the algorithm correctly outputs 1 with probability $\frac{1}{\pi}$.

It might be possible to use the following identity, also due to Ramanujan

$$\frac{1}{\pi} = \sum_{n=0}^{\infty} \binom{2n}{n}^3 \frac{42n+5}{2^{12n+4}}$$

This means that using the same generation as before we could simply use the same procedure but setting the S_1 and S_2 such that $p_{T_n} = \frac{42n+5}{2^{6n+4}}$. However finding the coefficients necessary is not that easy, and the procedure obtained would probably we quite long and require many more flips.

1.2 Analysis of RAMA

Assuming that RAMA indeed computes $\frac{1}{\pi}$ correctly, we can try to see its efficiency, in number of flips done. This is quite easily done as after S_2 the expected number of flips is $\alpha_n = 2n\left(1 + \binom{2n}{n}2^{-2n} + \binom{2n}{n}^2 2^{-4n}\right)$, so the expected number of flips is the number of flips to draw T plus $\sum_{n=0}^{\infty} p_{T_n}\alpha_n$.

We can bound α_n by using bounds from Stirling's approximation to get $\alpha_n \leq 2n(1 + \frac{e}{2\pi n} + \frac{e^2}{4\pi^2 n^2}) \leq 2n + 1 + \frac{1}{n}$.
Hence we need to estimate

$$\sum_{n=1}^{\infty} \left(12n^2 + 8n + 7 + \frac{1}{n}\right) 2^{-2n-2}$$

This is equal to

$$\sum_{n=0}^{\infty} 3 \times n(n-1)4^{-n} + \sum_{n=0}^{\infty} 5 \times n \times 4^{-n} + \frac{7}{12} + \sum_{n=1}^{\infty} \frac{1}{n} 2^{-2n-2}$$

$$\frac{8}{9} + \frac{20}{9} + \frac{7}{12} + a$$

Where $a = \sum_{n=1}^{\infty} \frac{1}{n} 2^{-2n-2} \leq \frac{1}{12}$ (we use the fact that functions of the type $\sum P(n)\alpha^n$ correspond to sums of derivatives of functions of the form $\sum \alpha^n = \frac{1}{1-\alpha}$).

We can show that this is less than $\frac{34}{9} \approx 3.77$. The number of flips needed to find X_1 and X_2 is 8/3 for each (because each $\Gamma B(\frac{1}{4})$ takes 2 flips), so a total of 16/3 flips, and the $\Gamma B(\frac{5}{9})$ can be done with an average of $\frac{16}{9} \times 4$ flips at most (or $\frac{32}{27} \times 5$ if we use a better generator).

At the very end we can see that the total number of flips is bounded on expectation by 15 (by$\approx$ 15.037). Moreover, we could also lower bound it by 14.9 by a detailed analysis of the first few cases. Finally, we can see that the variance of all but the last step is $\frac{3392}{729} \approx 4.65$ which at least gives a lower bound for the total variance.

2 Creating a $\Gamma B(\frac{\pi}{4})$

2.1 Madhava-Gregory-Leibniz generator

The last procedure in the original paper does the following :

MGL:=proc() Repeat {

if bag(U)=0 then return(1) fi;

if bag(U)=0 then return(1) fi;

if bag(U)=0 then return(0) fi;

if bag(U)=0 then return(0) fi; } end.

The goal of this procedure is to have a Bernoulli generator of ratio $\frac{\pi}{4}$. It uses two relations to obtain that, $\tan^{-1}(1) = \frac{\pi}{4}$ and $\int_0^x \frac{1}{1+x^2} = \tan^{-1}(x)$. The authors combine three tools they describe during their paper to get the generator, to get something of the form :

BAG(PARITY(SQUARE(X)))

Where SQUARE is the function that to a $\Gamma B(\lambda)$ associates a $\Gamma B(\lambda^2)$, PARITY is the one that to $\Gamma B(\lambda)$ associates $\Gamma B(\frac{1}{1+\lambda})$ and BAG is the one that to $F(\lambda)$ associates $\frac{1}{\lambda}\int_0^{\lambda} F(x)dx$.

Actually the MGL procedure seems to be wrong, because we do not have a real squaring in it.

If we just look at PARITY(SQUARE(X)), we get a generator that transforms $\Gamma B(\lambda)$ into $\Gamma B(\frac{1}{1+\lambda^2})$ with a procedure that looks like

Repeat {

if $\Gamma B(\lambda)$=0 then

if $\Gamma B(\lambda)$=0 then return(1) fi; fi;

if $\Gamma B(\lambda)$=0 then

if $\Gamma B(\lambda)$=0 then return(0) fi; fi; } end.

Assuming that BAG transforms a function correctly (this corresponds to Theorem 4.1 which is explained but lacks a formal proof), this one clearly computes $\int_0^1 \frac{1}{1+\lambda^2} d\lambda = \frac{\pi}{4}$. Either this is equivalent to the one without dependent ifs, or the first one does not compute $\Gamma\mathrm{B}(\frac{1}{1+\lambda^2})$ and does not compute $\frac{\pi}{4}$. So we need to find what the other computes, which is

$$\sum_{k=0}^{\infty} \lambda^{4k}(1-\lambda) + \lambda^{4k+1}(1-\lambda) = \sum_{k=0}^{\infty} \lambda^{4k} - \lambda^{4k+2} = \frac{1-\lambda^2}{1-\lambda^4} = \frac{1}{1+\lambda^2}$$

It indeed turns out that both procedures are equivalent so we can save some complexity and one bit of memory by using the original one.

2.2 Analysis

If we consider a call to

if $\Gamma\mathrm{B}(\lambda)$=0 then

if $\Gamma\mathrm{B}(\lambda)$=0 then return(1) fi; fi;

return(0)

As equivalent to a call to $\Gamma\mathrm{B}(\lambda^2)$ then we can see that the procedure is in fact

if $\Gamma\mathrm{B}(\lambda^2)$=0 then return(1) fi;

if $\Gamma\mathrm{B}(\lambda^2)$=0 then return(0) fi;

But the probability of $k+1$ calls to $\Gamma\mathrm{B}(\lambda^2)$ is $\lambda^{2k}(1-\lambda^2)$, and by summing over all even k, we get that the probability of success is $(1-\lambda^2)\sum \lambda^{4k} = \frac{(1-\lambda^2)}{1-\lambda^4} = \frac{1}{1+\lambda^2}$.

Moreover the expected number of calls to $\Gamma\mathrm{B}(\lambda^2)$ after the first is equal to $\sum n(\lambda^2)^n = \frac{\lambda^2}{(\lambda^2-1)^2}$.

To get the number of calls to $\Gamma\mathrm{B}(\lambda)$ we need to compute the probability of getting 2 calls knowing that we did not return 1, which is equal to $\frac{(1-\lambda)\lambda}{(1-\lambda)\lambda+\lambda}$, so the expectation is $\frac{\lambda+2(1-\lambda)\lambda}{(1-\lambda)\lambda+\lambda}$, and the total expectation in number of calls to $\Gamma\mathrm{B}(\lambda)$ is

$$\frac{\lambda^2}{(\lambda^2-1)^2} \frac{\lambda+2(1-\lambda)\lambda}{(1-\lambda)\lambda+\lambda} + 2$$

because we always have two calls to $\Gamma\mathrm{B}(\lambda)$ before returning anything.

If however we use the original generator, the analysis is much simpler, as the expectation on the number of calls becomes $(1-\lambda)\sum_{k=0}^{\infty} \lambda^k(k+1) = \frac{1-\lambda}{(1-\lambda)^2} = \frac{1}{1-\lambda}$. As $\frac{\lambda^2}{(\lambda^2-1)^2} \frac{\lambda+2(1-\lambda)\lambda}{(1-\lambda)\lambda+\lambda} + 2 > \frac{1}{1-\lambda}$, the expectation of the original generator is also lower (generally by about 1, unless λ is close to 1).

The authors assert that the expectation of this generator is infinite, let's prove it. First we can see that the expected number of flips done is exactly the expected number of flips done before encountering a 0 with the BAG. However the BAG generates a real x between 0 and 1 randomly and returns 0 if the bit number X is 0, where $X \in Geo(\frac{1}{2})$. If $x > 1-2^k$ we can see that the expected number of flips to get a 0 is at least 2^k. Hence we can sum the expected number of calls to BAG (as the events considered are disjoint)

$$\mathbb{E}(\sharp BAG) \geq \sum_{n=0}^{\infty} \Pr(1-2^{-n} < x < 1-2^{-n-1}) \times 2^k$$

And as the right part diverges, so does the expectation.

The interesting part is that generating any other $\tan^{-1}(x)$ for $x < 1$ can be done in finite expected number of flips, as we can multiply the right hand side in the previous inequality by two to make it bigger than the expectation.

Moreover we also need to see the number of calls to find which bit of x we're looking at. The expectation on the number of calls to bag is then

$$\mathbb{E}(\sharp BAG) \leq \sum_{n=0}^{\alpha} \Pr(1-2^{-n} < x < 1-2^{-n-1}) \times 2^{k+1}$$

For example we can compute $\tan^{-1}(\frac{1}{2})$ or $\tan^{-1}(\frac{1}{3})$ by setting $\alpha = 1$ and get

$$\mathbb{E}(\sharp BAG) \leq 4$$

And as the number of flips made in each call is at most 2 in expectation (plus 1 if we encounter a new bit of x), we get an upper bound of 12 flips (which can be lowered once again by a refined analysis). This gives a total of 24 operations at most (in practice it is about 4 times lower in the authors' simulations).

2.3 Another estimator for $\frac{\pi}{4}$

To compare with the previous procedure we can look at the classical operation which draws two numbers at random in $[0;1]$ and see if $x^2 + y^2 < 1$. To analyse this we could imagine cutting $[0;1]^2$ into 2^{2k} squares and accepting if and only if the partial bits chosen for the two numbers correspond to a square that is strictly under the curve, and refusing if it is strictly over the curve. It is evident that this computes $\frac{\pi}{4}$, and as there are at most 2^{k+1} squares crossed by the curve out of 2^{2k} we can see that the probability of drawing the $k-th$ bit is at most 2^{k-2}. Hence the expected number of flips is at most

$$2 \times \sum_{n=0}^{\infty} n 2^{2-n}$$

The initial 2 is there to account for the fact that we are flipping for both numbers. This is not a very accurate estimate but it already gives an upper bound of 16. If we refine we can see that the curve actually crosses at most $2^k + 1$ squares, which gives $2 \times \sum_{n=0}^{\infty} n(2^{1-n} + 2^{-2n}) \leq 9 + \frac{1}{3}$. Actually, the final value is also at least $2 \times \sum_{n=0}^{\infty} n 2^{-n} \geq 4$. Tests made using a small python script showed that the actual average is ≈ 5.330 on a million tries, which is slightly better than the both of the generators given.

2.3.1 Python estimator script

```
import random

i = 0; S = 0;

while i<10000:

    k=0; x=[]; y=[];

    while (sum(x)**2+sum(y)**2<1 and (sum(x)+(1/2**k))**2+(sum(y)+(1/2**k))**2>=1) :

        k = k+1; x.append(random.randrange(0,2)/(2**k)); y.append(random.randrange(0,2)/(2**k));

    S=S+2*k; i=i+1;

print (S/10000)
```