pqRand: better random samples for the future of Monte Carlo simulation

Keith Pedersen
(kpeders1@hawk.iit.edu)

ILLINOIS INSTITUTE OF TECHNOLOGY

Appearing in arXiv:**1704.07949**

CAPP meeting, Illinois Tech, 26 Oct 2017

Outline

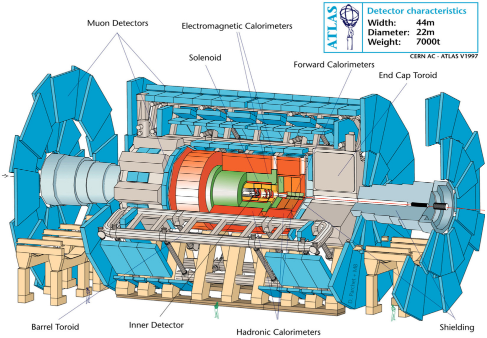
1. **The importance of Monte Carlo simulation**
   - Simulation, integration, numerical experimentation
   - The beating heart of Monte Carlo
2. **Sampling from random distributions**
   - The quantile function
   - PRNG $\to U(0,1)$ can be *too uniform*
3. **Why we should use pqRand**
   - Quasi-uniform sampling
   - Better samples; better integrals
   - The **pqRand** package

Simulation, integration, validation

Monte Carlo simulation:
Use randomness to solve difficult problems.

Big non-linear systems require **big simulations**:

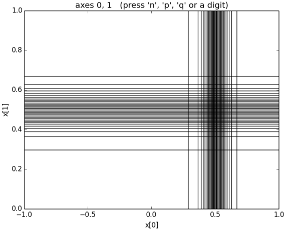
- LHC particle detectors
- Cosmic evolution
- Beam dynamics



Monte Carlo integration beats the curse of dimensionality:

- Randomly find important regions
- Easily automated for arbitrary $f(x)$

Quickly validate an analytic solution.



Monte Carlo simulations need random numbers

To sample from $f(x)$ … **IID**.

Identically: Whole sample is true to $f(x)$
Independently: No correlations!
Distributed

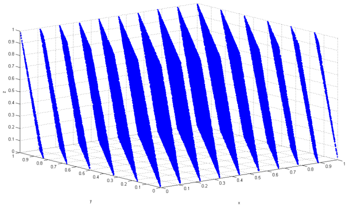
IID is hard! Need a universal tool

1. IID random bits (e.g. `uint`)
2. random bits $\rightarrow$ floating point

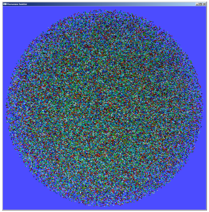
Step 1: *Pseudo*-random is better:

- Faster/cheaper on CPU (no I/O lag).
- *Repeatable* from known sed.
- When in doubt … use MT19937.

Step 2: equally important!



RANDU: 3 consecutive values live in planes

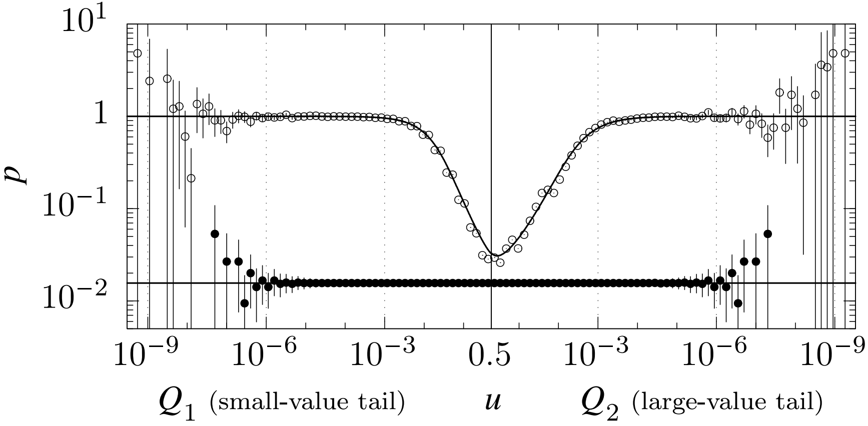


MT19937: The Mersenne twister

The problem with step 2

Sampling from $f(x) = \exp(-x)$; $N =$ unique values; $p \equiv \frac{N_{\text{measured}}}{N_{\text{expected}}}$

● `std::exponential_distribution` ○ `pqRand::exponential`



Outline

1. The importance of Monte Carlo simulation
   - Simulation, integration, numerical experimentation
   - The beating heart of Monte Carlo
2. Sampling from random distributions
   - The quantile function
   - PRNG $\rightarrow U(0,1)$ can be *too uniform*
3. Why we should use pqRand
   - Quasi-uniform sampling
   - Better samples; better integrals
   - The **pqRand** package

Drawing from the exponential distribution

Radioactive metal with decay rate λ.
How long till the next decay?

Poisson statistics $\rightarrow$ Exp. distribution

The probability distribution function

$$\text{PDF}: \quad f(t) = \lambda \exp(-\lambda\, t)$$

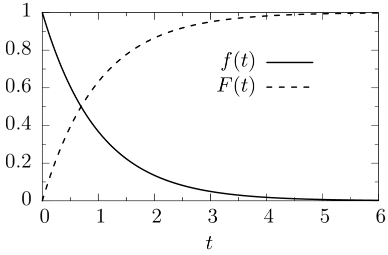
The cumulative distribution function

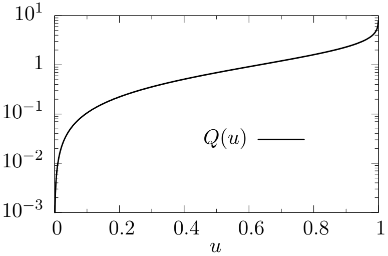
$$\text{CDF}: \quad F(t) = \int_0^t f(t')\,\mathrm{d}t' = 1 - e^{-\lambda t}$$

The quantile function (u) $\quad (0 < u < 1)$

$$Q(u) = F^{-1} = -\log(1-u)/\lambda$$

Uniform sample: $\{U(0,1)\} \rightarrow Q \rightarrow \{f\}$





How to convert PRNG $\to U(0,1)$?

Computers can't use $\not\mathbb{R}$, only $\mathbb{Q}$;
floating point numbers w/ precision P

$$\underbrace{1.010}_{\text{mantissa (P=4)}} \times \overbrace{2^1}^{\text{exponent}} = 5/2 = 2.5\,.$$

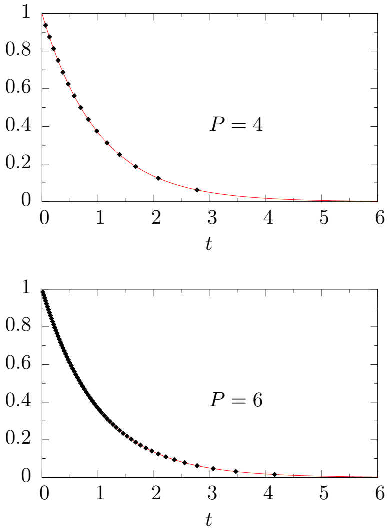
If **PRNG** is uniform, so is u:

$$u = \frac{\mathrm{float}(\mathbb{Z}(0,\, 2^P))}{2^P}$$

Sample space is evenly distributed ...

- Only $2^P - 1$ values ... repetition
- The tail is sparsely populated
- Many tail values are unattainable

$U(0,1) \to Q$

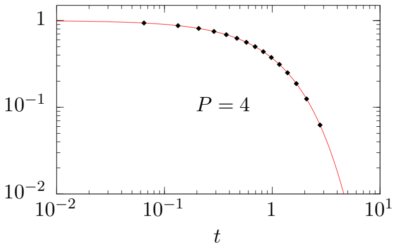


Outline

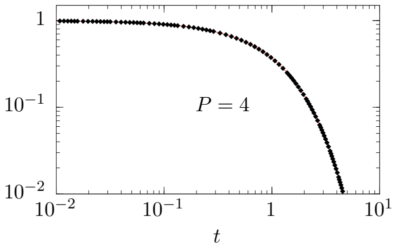
1. The importance of Monte Carlo simulation
   - Simulation, integration, numerical experimentation
   - The beating heart of Monte Carlo
2. Sampling from random distributions
   - The quantile function
   - PRNG $\rightarrow U(0,1)$ can be *too uniform*
3. Why we should use pqRand
   - Quasi-uniform sampling
   - Better samples; better integrals
   - The **pqRand** package

Getting arbitrarily close to zero

Standard $U(0,1)$

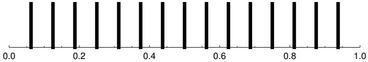


pqRand $U(0,1)$

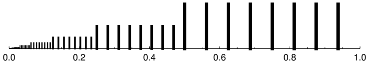


Need small u to fill tails!

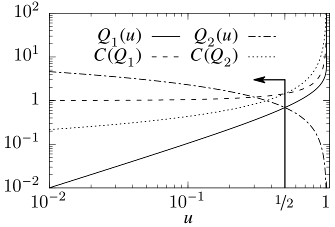
$\boldsymbol{U(0,1)}$: Draw from $\mathbb{R}$, round to $\mathbb{Q}$.



becomes



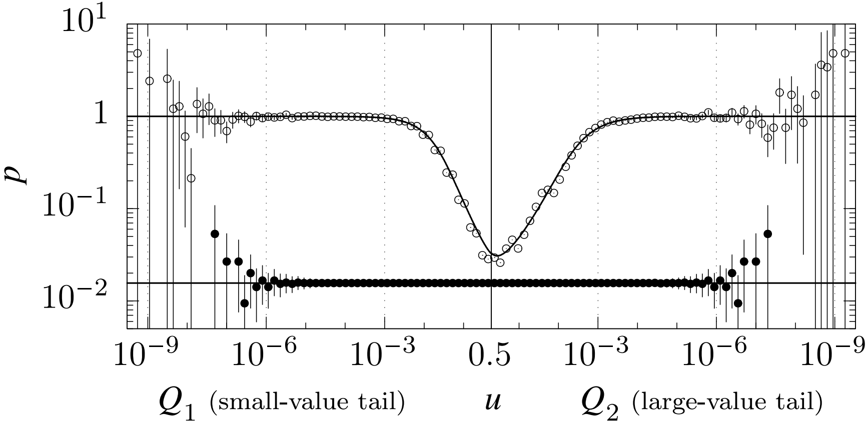
$Q_1(u) = -\log(1-u)/\lambda \qquad Q_2(u) = -\log(u)/\lambda$



Fixing the exponential distribution

Sampling from $f(x) = \exp(-x)$; $N =$ unique values; $p \equiv \frac{N_{\text{measured}}}{N_{\text{expected}}}$

● `std::exponential_distribution` ○ `pqRand::exponential`



Monte Carlo integration

Monte Carlo integration (VEGAS) is a very common HEP tool:

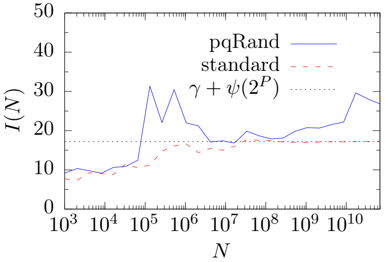
$$I(f) = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{g(x_i)} \approx \int \mathrm{d}x\, f(x)$$

where $g(x)$ is the PDF for random x_i.

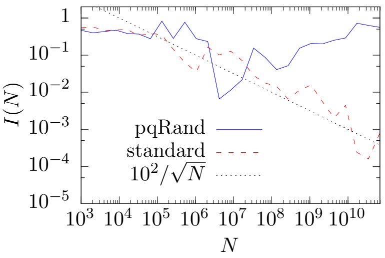
What is the mean μ of a Pareto distribution $g(x) = x^{-2}$?

$$\mu = \int_1^\infty x\,\frac{1}{x^2}\,\mathrm{d}x = \int_1^\infty \frac{1}{x}\,\mathrm{d}x = \infty$$

Why doesn't the standard method diverge? The sample space is too finite!



Relative error to $\gamma + \psi(2^P)$



pqRand for C++ and Python

pqRand is here! **https://github.com/keith-pedersen/pqRand**

```cpp
#include <cstdio>
#include "pqRand.hpp"
#include "distributions.hpp"

using namespace pqRand;
int main()
{
  engine rng;
  exponential dist(1.);

  size_t const N = size_t(1)<<20;
  double sum = 0.;

  for(size_t i = 0; i < N; ++i)
    sum += dist(rng);

  printf("%.16e\n", sum/N);
}
```

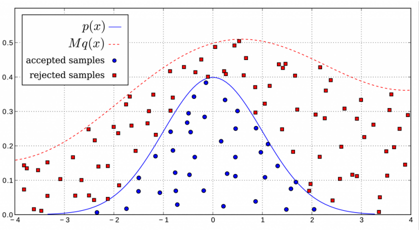
```python
import pYqRand as pqr

rng = pqr.engine()
dist = pqr.exponential(1.)

N = 1 << 20
total = 0.

for __ in range(0, N):
   total += dist(rng);

print(total/N)
```

- C++ and Python
- Exponential, normal, log-normal, pareto, weibull, and uniform distributions.

Do subtle tail effects matter?

Rejection sampling needs a high-quality **proposal distribution**

Rejection sampling

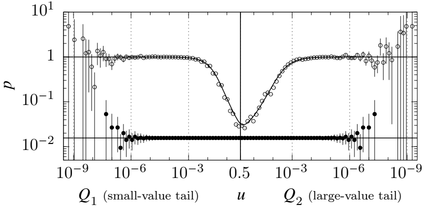


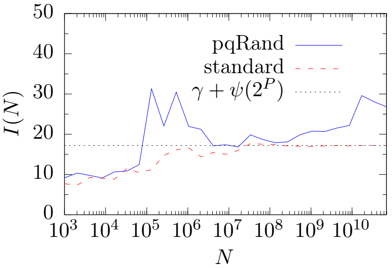
What does the future hold?

- Monte Carlo simulations are *growing*: larger N, more non-linear.

Are we sensitive to these effects?

- Who knows? Validation is hard! The best parts give the best results.





The end

Thank you for your attention!

Normal distribution

Indirect quantile function — Marsaglia polar method