

# Numerical simulations using approximate random numbers: VLA algorithms for SVE and FP16 capable hardware

Oliver Sheridan-Methven

`oliver.sheridan-methven@maths.ox.ac.uk`

Monday 16<sup>th</sup> September 2019

Supervisors:

Prof. Michael Giles                      Oxford

Dr Christopher Goodyer                 Arm

*Of course my code will run faster if it's vectorised.*

No

*Of course my code will run faster if I worked in a lower precision.*

No

*Well my compiler should be clever enough to decide what's best.*

No

*Half-precision is useless for anyone who wants accurate answers!*

No

*You can't design code that performs well on arbitrary vector lengths.*

No



- ① What is an approximate random variable?
- ② When can we use them?
- ③ Do we still get the right answer?
- ④ What precisions can we use, and when?
- ⑤ Conclusions

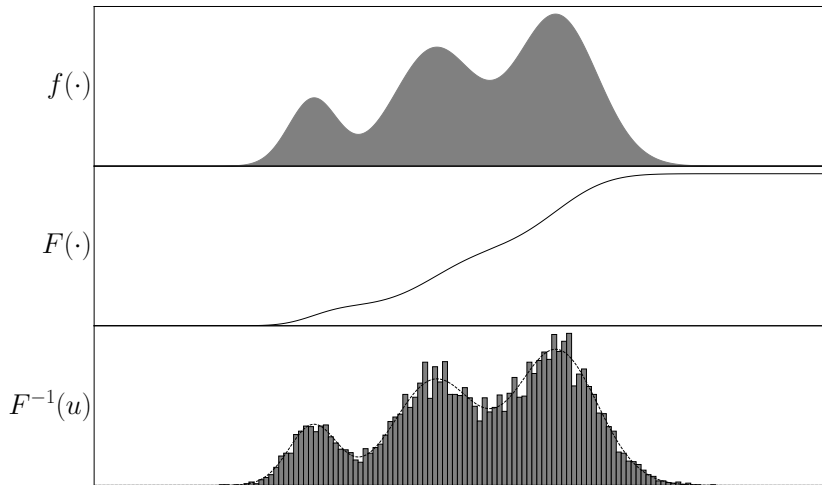
Weather Tomorrow's temperature given today's weather.

Finance Value of contract given today's prices.

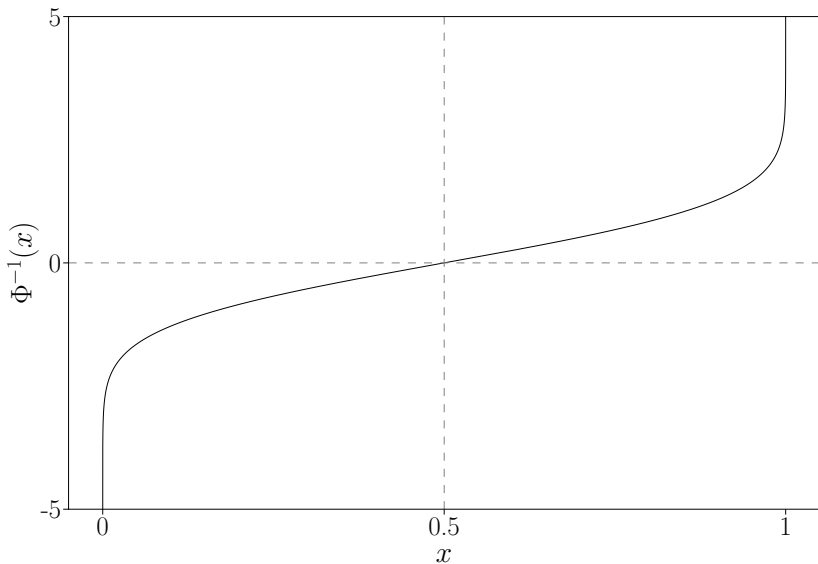
Traffic Rush hour traffic given morning congestion.

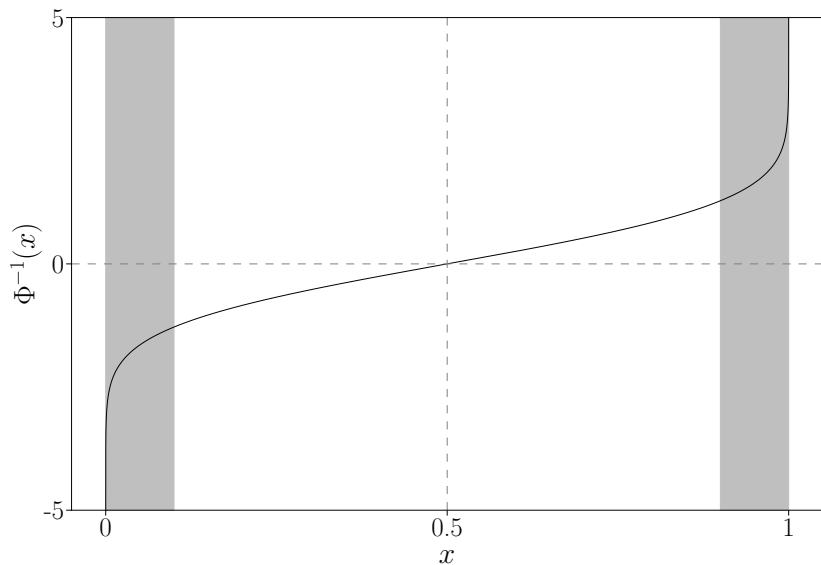
Health Risk of later secondary condition given current health.

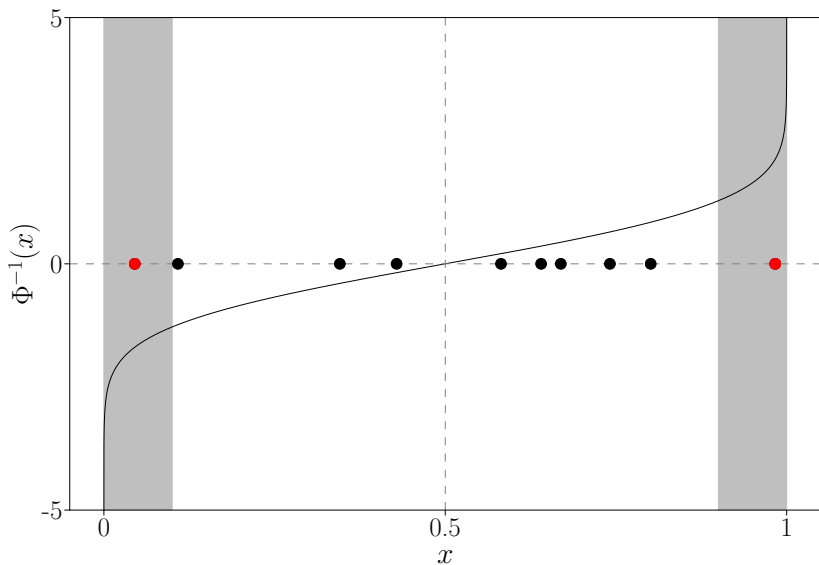
Technology Expected number of online visitors given current search trends.



# The inverse Gaussian CDF $\Phi^{-1}(\cdot)$

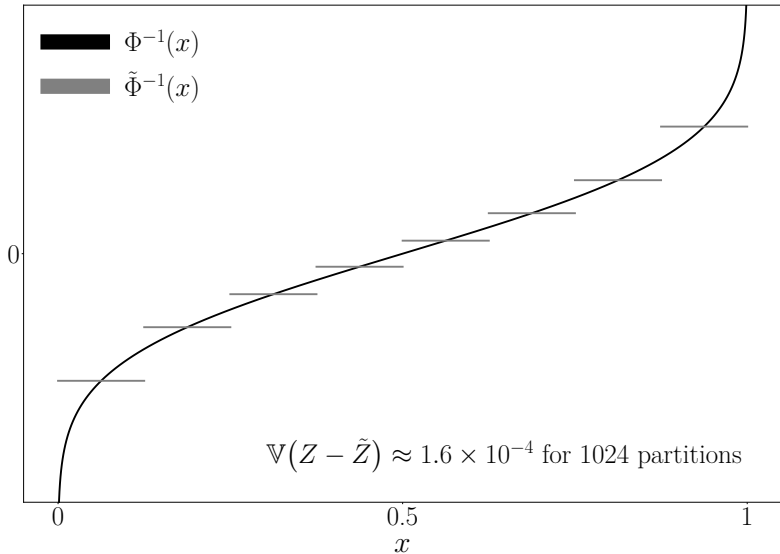




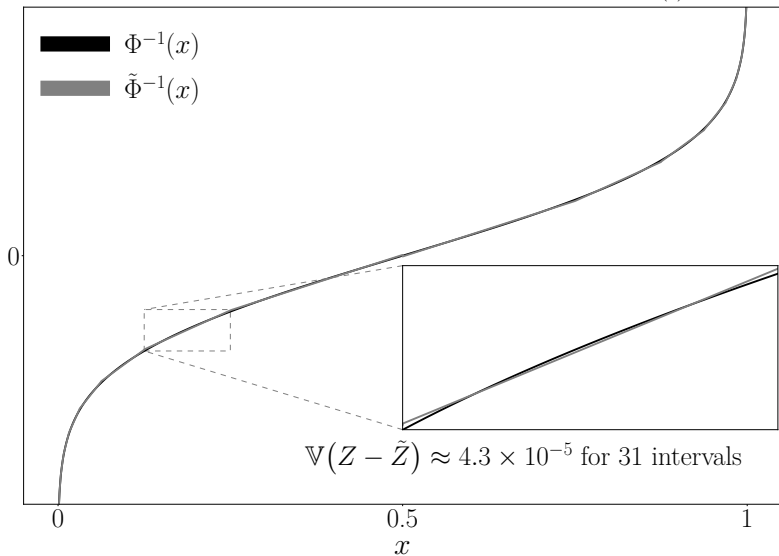


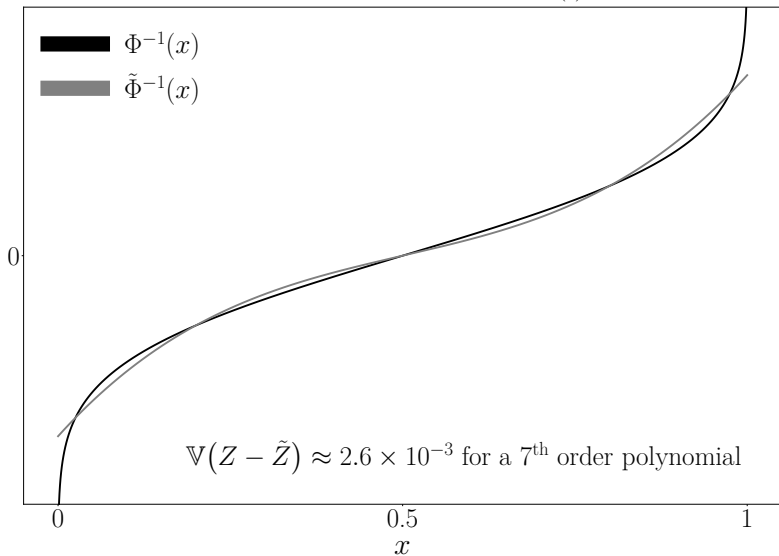


Uniform piecewise constant approximation of  $\Phi^{-1}(\cdot)$



Piecewise linear dyadic approximation of  $\Phi^{-1}(\cdot)$



Cubic approximation of  $\Phi^{-1}(\cdot)$ 

	Average speed (clock cycles)
Intel MKL	8
Lookup table	2

Average speed  
(clock cycles)

Intel MKL	8
Lookup table	2
GNU GSL	128
Cephes	83
NAG	117

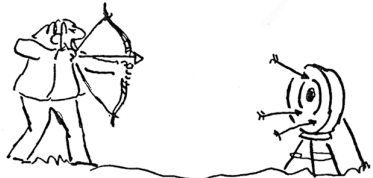
	Average speed (clock cycles)
Intel MKL	8
Lookup table	2
GNU GSL	128
Cephes	83
NAG	117
GSL (optimised)	17
Giles [1] (NVIDIA?)	12

# Accuracy and precision

HIGH PRECISION  
HIGH ACCURACY



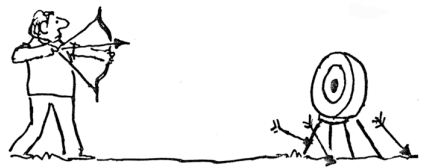
LOW PRECISION  
HIGH ACCURACY



HIGH PRECISION  
LOW ACCURACY



NO PRECISION  
NO ACCURACY  
NO SPECTATORS

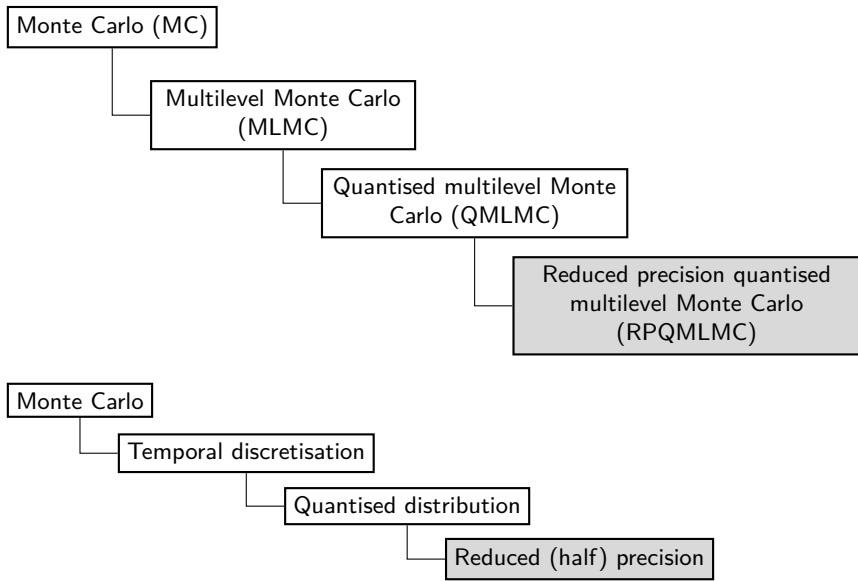


J. HARRIS

$$\mathbb{E}(P) \approx \mathbb{E}(\hat{P}_{\text{Accurate}}) = \mathbb{E}(\hat{P}_{\text{Crude}}) + \mathbb{E}(\hat{P}_{\text{Accurate}} - \hat{P}_{\text{Crude}})$$



## Nested multilevel Monte Carlo I





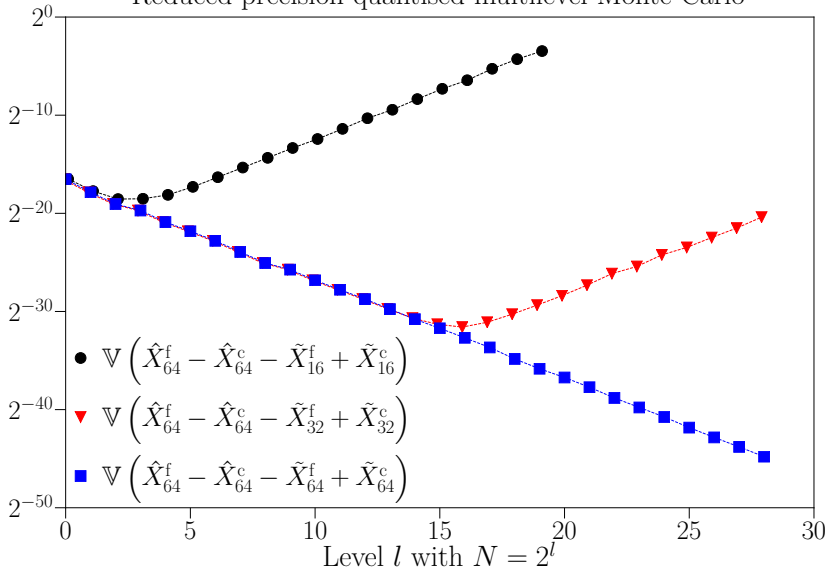
Correction = Discretisation  $\times$  Quantisation

$$\text{Correction} = \text{Discretisation} \times \text{Quantisation} \\ + \text{Roundoff}$$

Correction = Discretisation  $\times$  Quantisation

$$+ \frac{\underbrace{\text{Roundoff}}_1}{\text{Discretisation}}$$

Reduced precision quantised multilevel Monte Carlo



Running our QMLMC estimator with a single quantisation level (1024 bins) for finely discretised paths gives the following average time per path:

Relative accuracy $\varepsilon = 10^{-3}$ Times per path $10^{-4}$ s	Memory intensive	Work intensive
Original MLMC	24.8	17.0
Quantised MLMC	13.2	3.99

Level	paths
Original	1 920 000
Quantised	1 980 000
Correction	14 000

Running our QMLMC estimator with a single quantisation level (1024 bins) for finely discretised paths gives the following average time per path:

Relative accuracy $\varepsilon = 10^{-3}$ Times per path $10^{-4}$ s	Memory intensive	Work intensive
Original MLMC	24.8	17.0
Quantised MLMC	13.2	3.99

↓ ×2

Level	paths
Original	1 920 000
Quantised	1 980 000
Correction	14 000

Running our QMLMC estimator with a single quantisation level (1024 bins) for finely discretised paths gives the following average time per path:

Relative accuracy $\varepsilon = 10^{-3}$ Times per path $10^{-4}$ s	Memory intensive	Work intensive
Original MLMC	24.8	→17.0
Quantised MLMC	13.2	→3.99

Level	paths
Original	1 920 000
Quantised	1 980 000
Correction	14 000



Running our QMLMC estimator with a single quantisation level (1024 bins) for finely discretised paths gives the following average time per path:

Relative accuracy $\varepsilon = 10^{-3}$ Times per path $10^{-4}$ s	Memory intensive	Work intensive
Original MLMC	24.8	17.0
Quantised MLMC	13.2	3.99

$\downarrow \times 4$

Level	paths
Original	1 920 000
Quantised	1 980 000
Correction	14 000

Errors from using a cheap proxy distribution can be **quantified** and controlled by the introduction of a nested multilevel Monte Carlo framework.

There is a degree of freedom in the construction of this proxy. Put the results in the low level **cache**, or use a very cheap (piece-wise) **polynomial**.

The resultant approximations **converge**.

The **approximate schemes scale** as we move to wider vectors (SVE) and lower precisions (FP16), benefiting from greater SIMD parallelisation and faster FP16 calculations.

**Half-precision can be used** in the coarsest calculations (which consume most of the computer time). (BFloat16 may require Kahan summation...).

- [1] Mike Giles. Approximating the  $\text{erfinv}$  function. In *GPU Computing Gems, Jade Edition*, volume 2, pages 109–116. Elsevier, 2011.