Numerical simulations using approximate random numbers

Oliver Sheridan-Methven

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

Thursday 6th February 2020

Supervisors:

Prof. Michael Giles	Oxford
Dr Christopher Goodyer	Arm

Some HPC dogmas

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

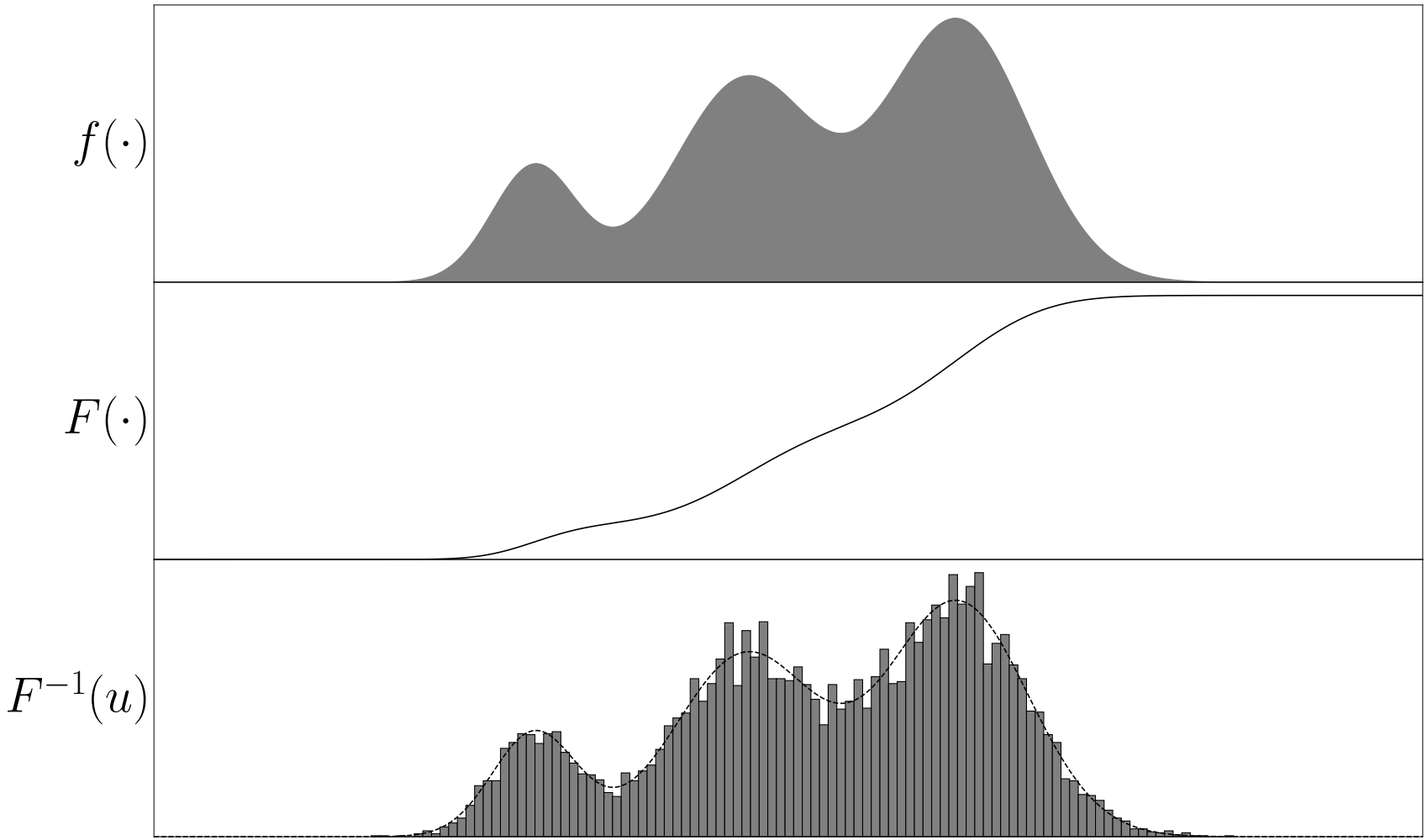
Overview

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

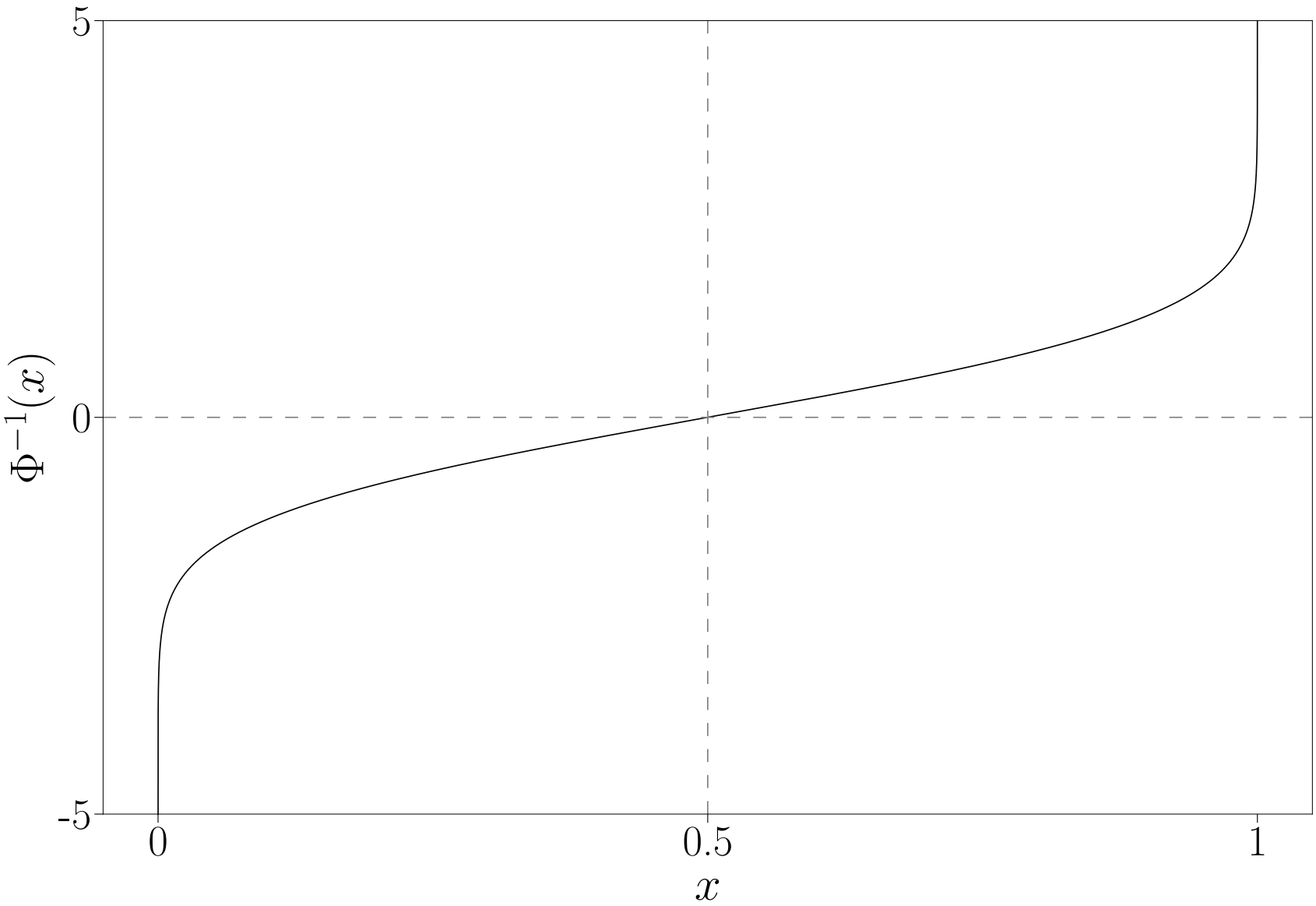
Stochastic simulations

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.

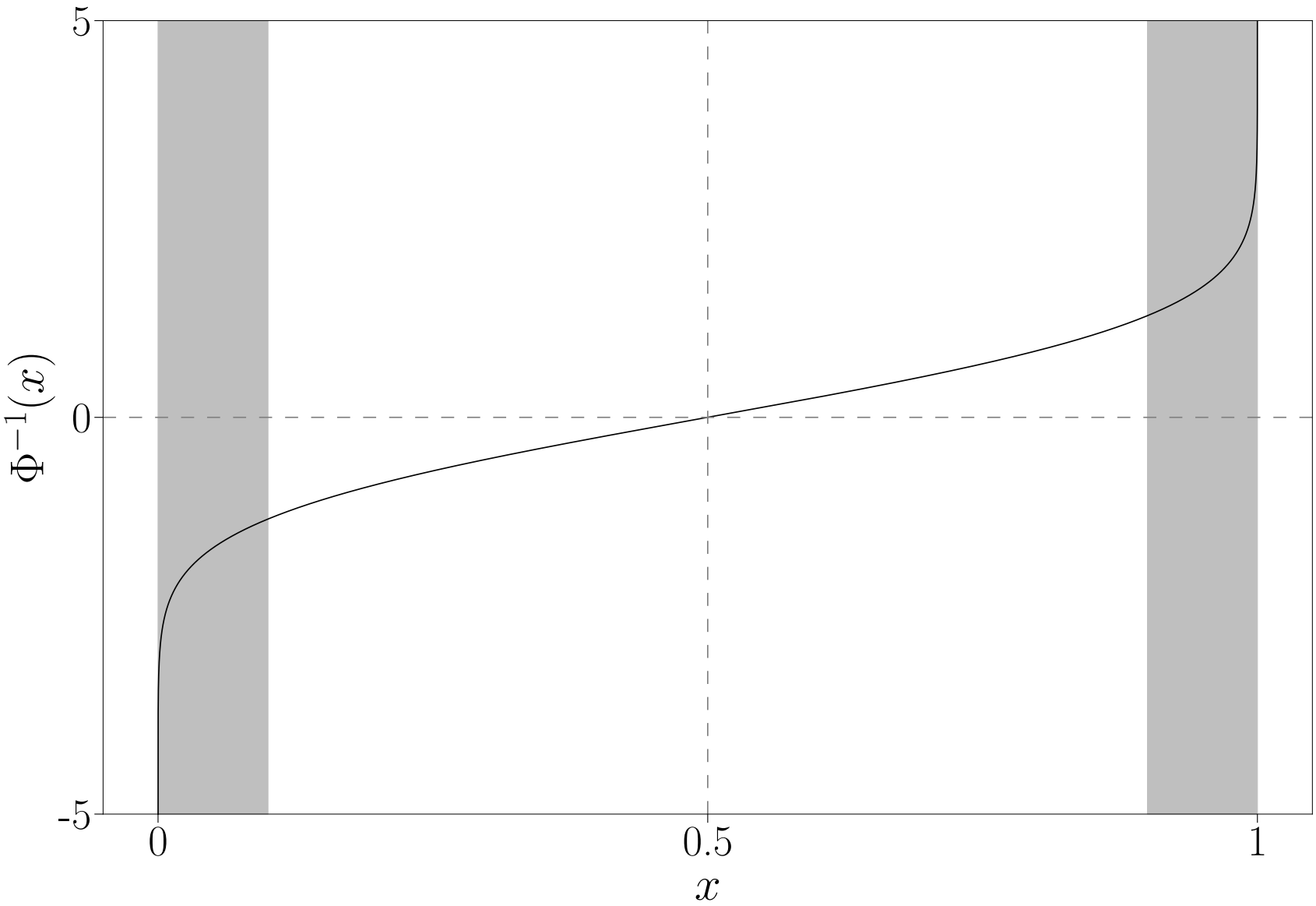
Mathematical simulations



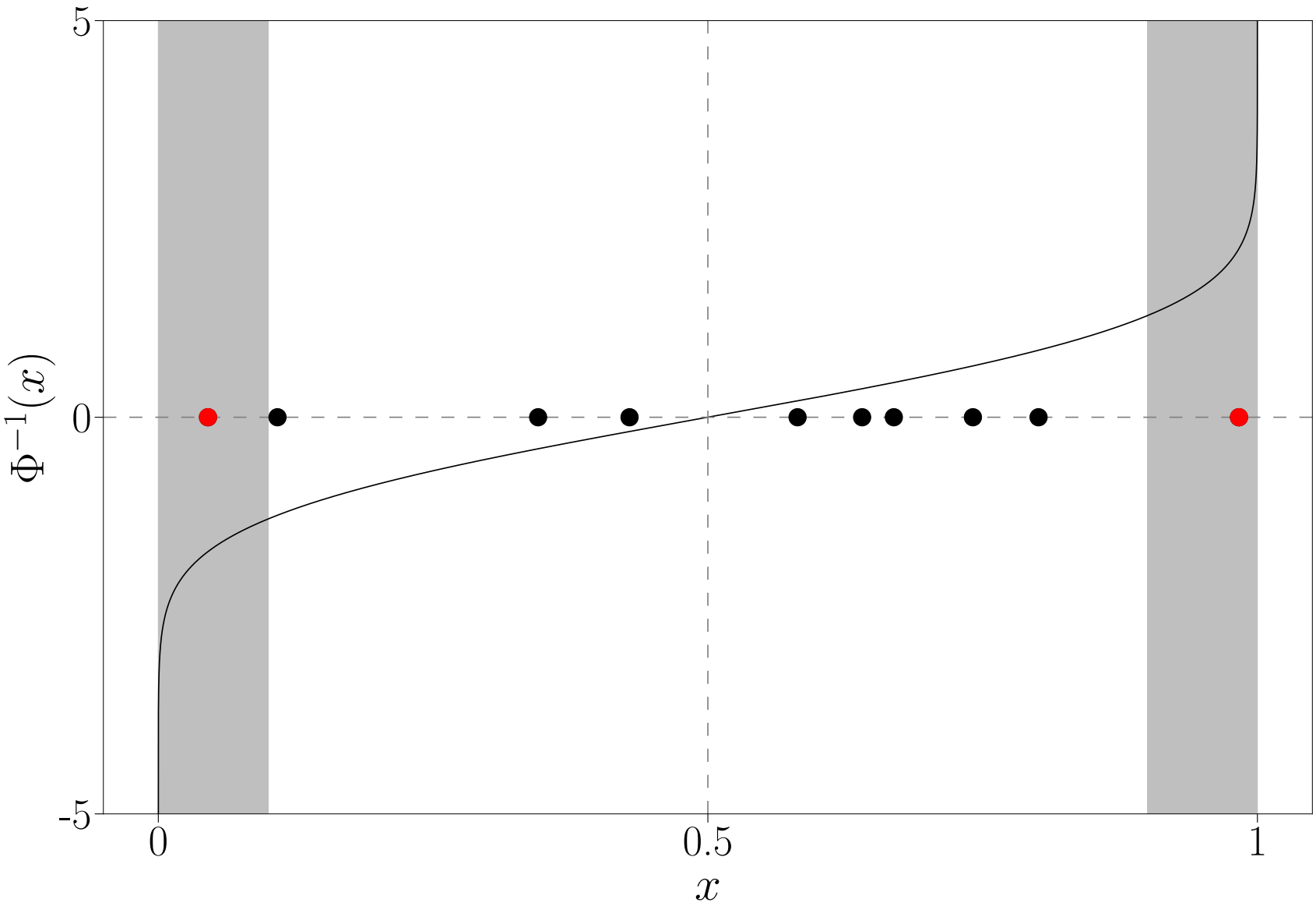
The inverse Gaussian CDF $\Phi^{-1}(\cdot)$



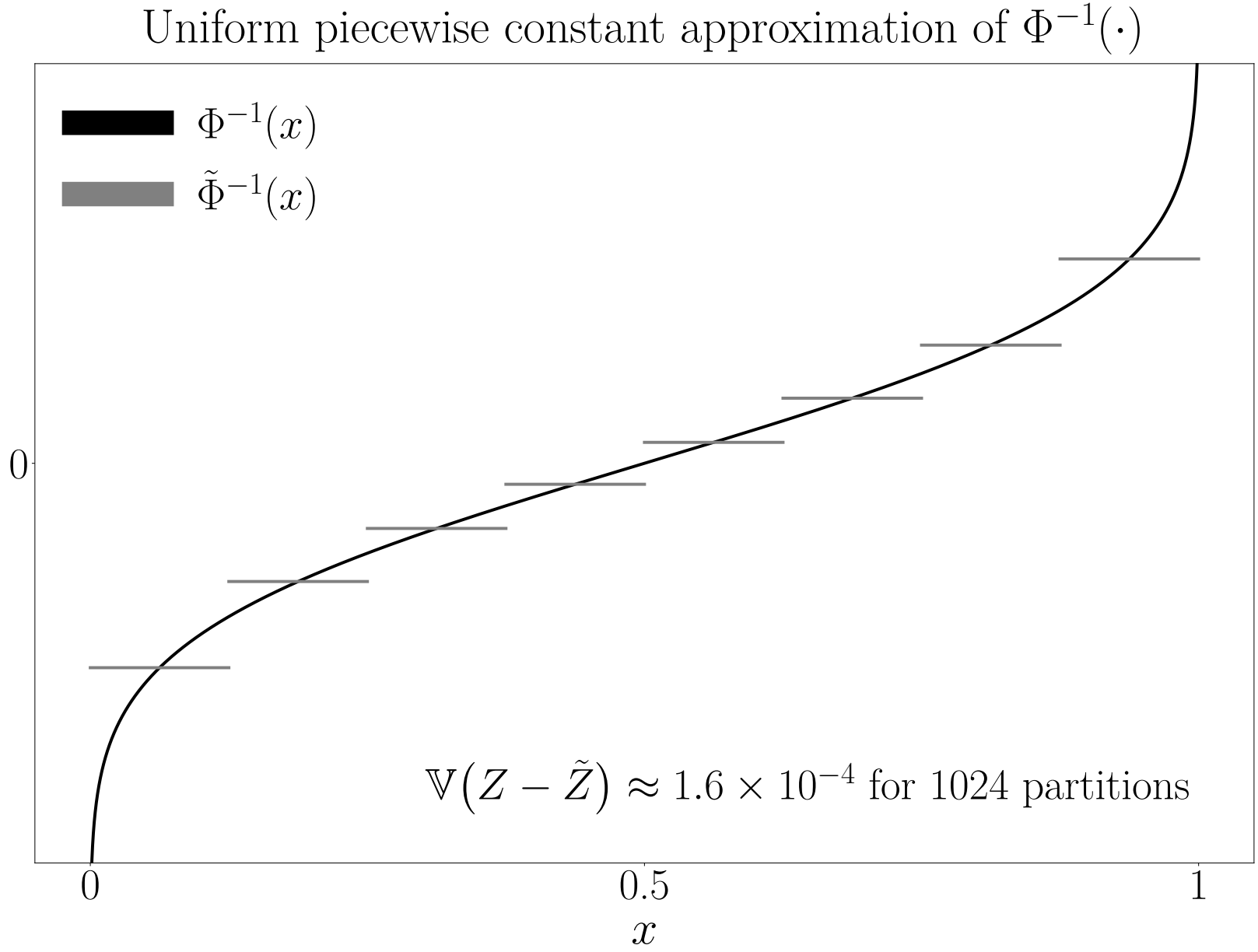
Tails kill performance in SVE and FP16



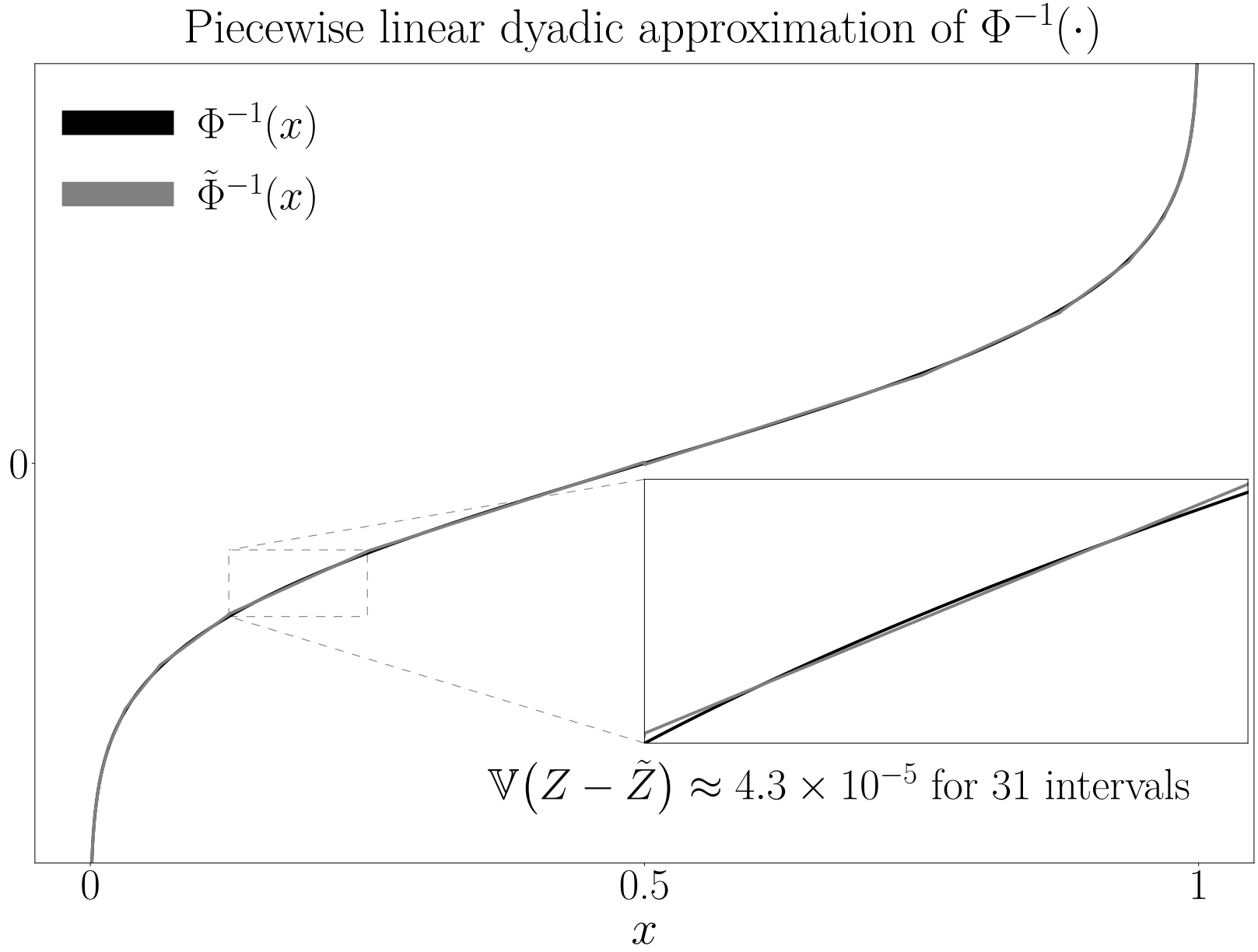
Tails kill performance in SVE and FP16



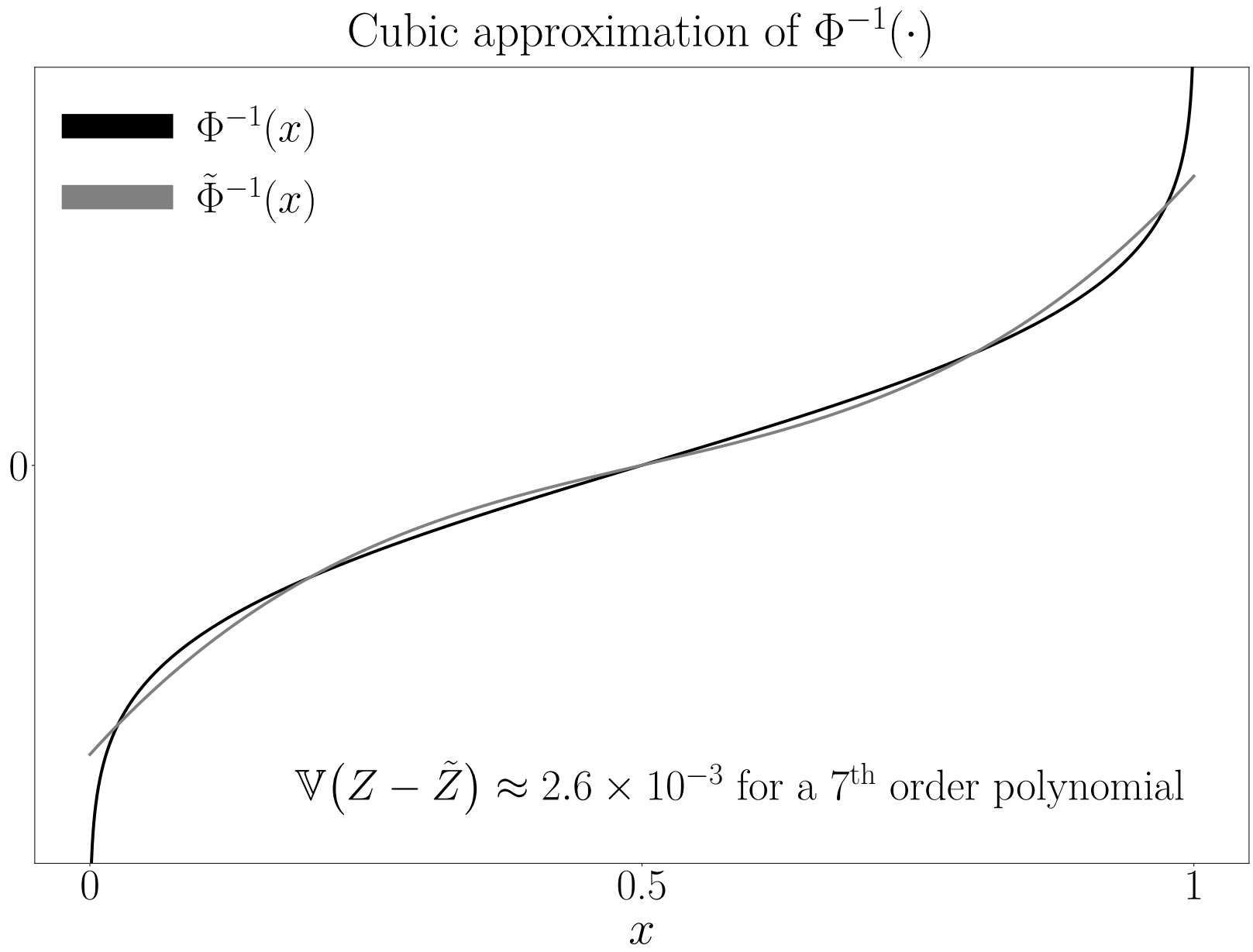
Piecewise constant approximation



Piecewise linear dyadic approximation



Polynomial approximation



Speed of execution

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

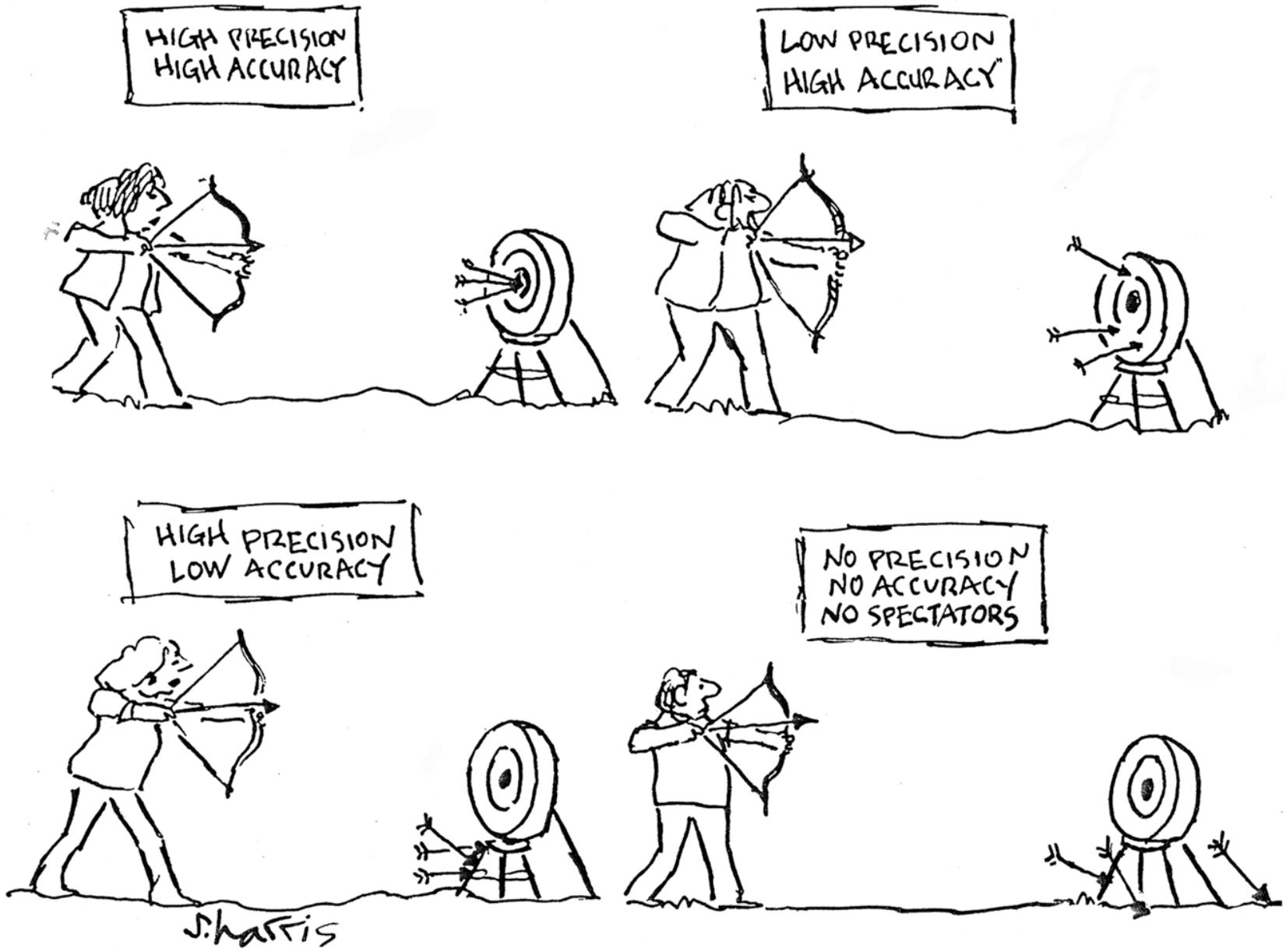
Speed of execution

	Average speed (clock cycles)
Intel MKL	8
Lookup table	2
GNU GSL	128
Cephes	83
NAG	117

Speed of execution

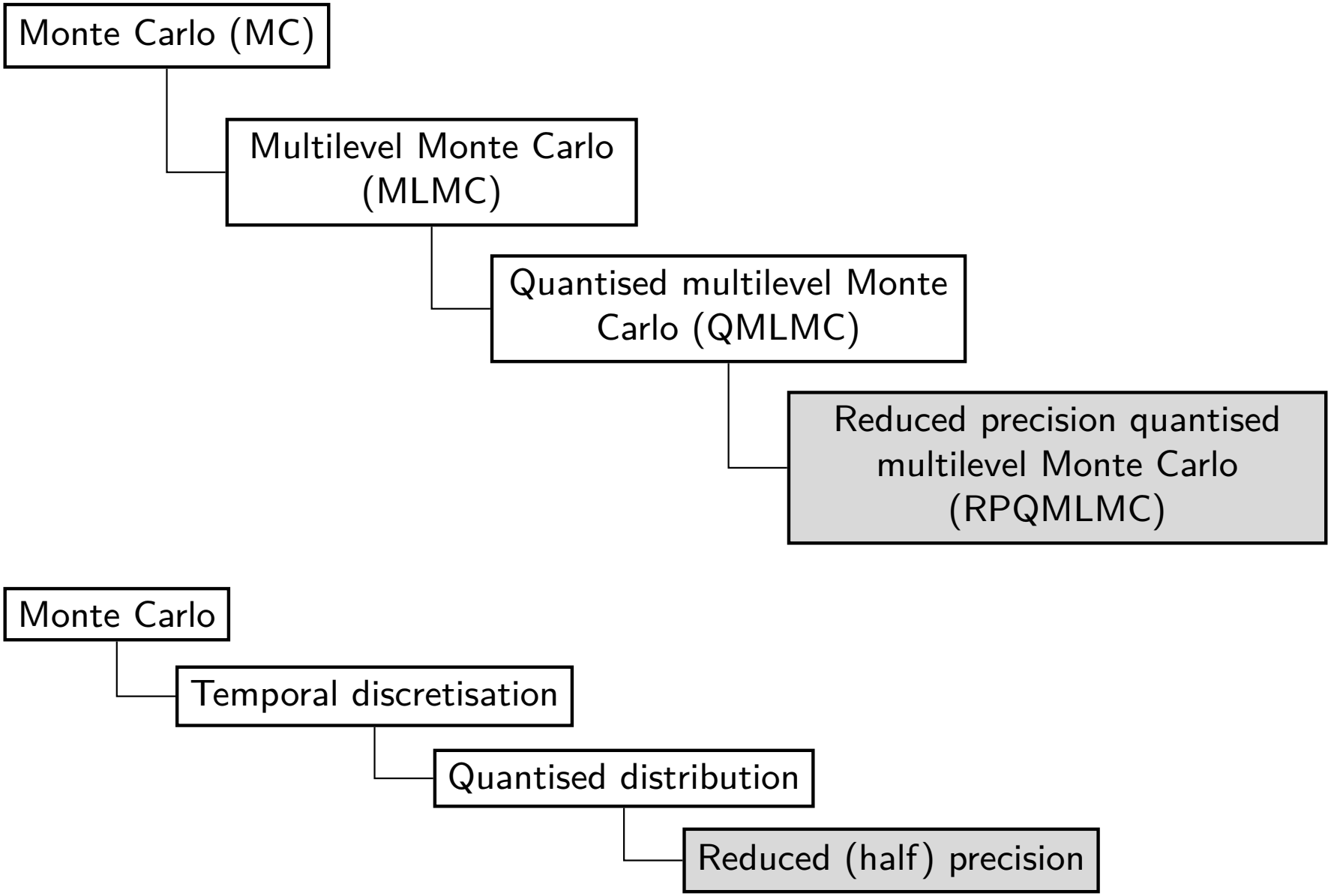
	Average speed (clock cycles)
Intel MKL	8
Lookup table	2
GNU GSL	128
Cephes	83
NAG	117
GSL (optimised)	17
Giles [1] (NVIDIA?)	12

Accuracy and precision



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

Nested multilevel Monte Carlo I



Discretisation, quantisation, and roundoff

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

Discretisation, quantisation, and roundoff

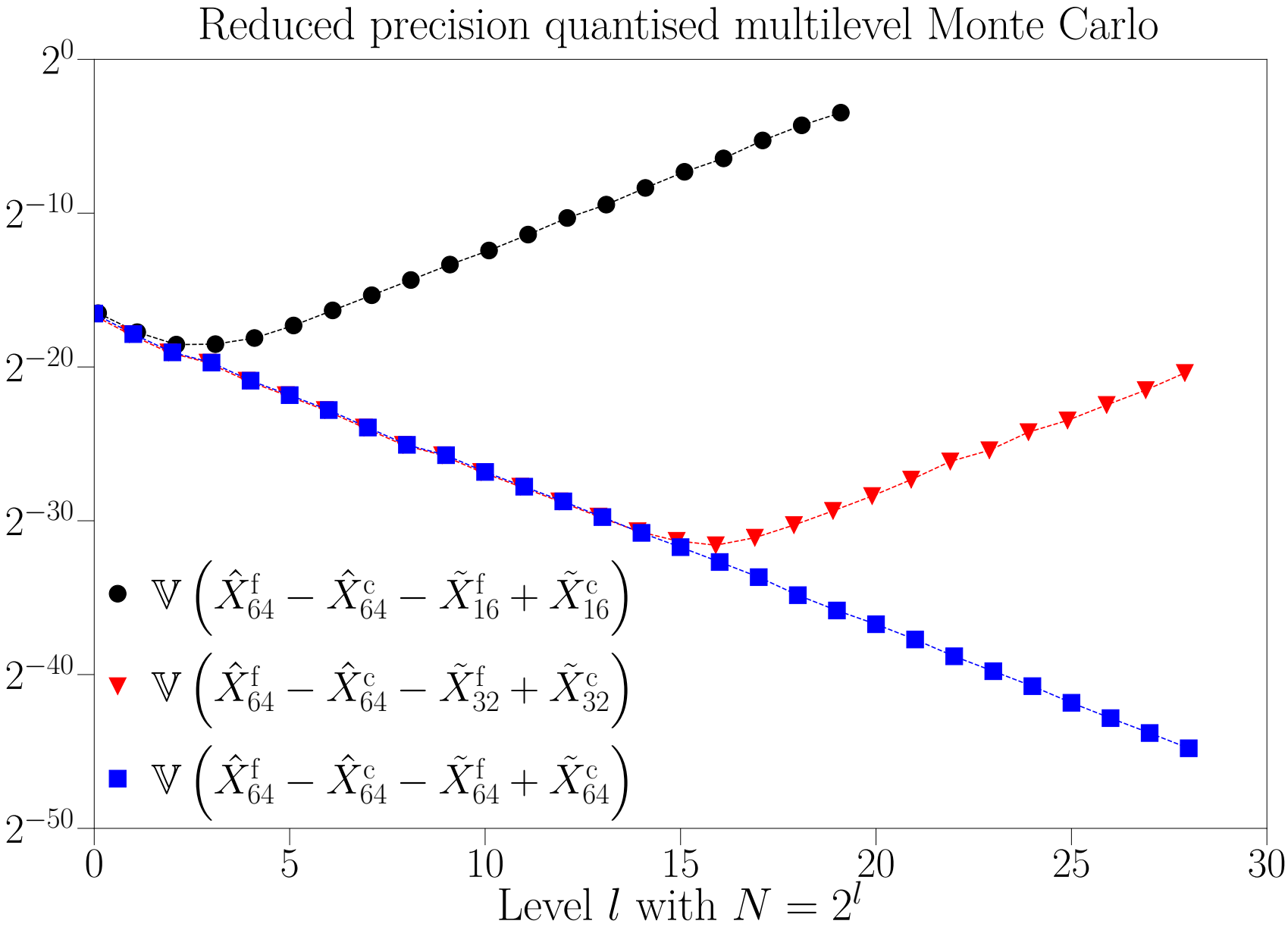
$$\text{Correction} = \text{Discretisation} \times \text{Quantisation}$$

$$+ \ \text{Roundoff}$$

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

$$+ \underbrace{\text{Roundoff}}_{\frac{1}{\text{Discretisation}}}$$

When can't we use half-precision?



Numerical results

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

Numerical results

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
	$\downarrow \times 2$	
Quantised MLMC	13.2	3.99

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

Numerical results

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

Numerical results

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
		$\downarrow \times 4$
Quantised MLMC	13.2	3.99

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

Conclusions

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 `erfinv` function. In *GPU Computing Gems, Jade Edition*, volume 2, pages 109–116. Elsevier, 2011.