New directions with Multilevel Monte Carlo methods

Mike Giles

Mathematical Institute, University of Oxford

Bath/RAL Numerical Analysis Day, Jan 11th 2016

Outline

- introduction to MLMC
- adaptive timesteps
- reflected diffusions
- variable precision arithmetic, and related inaccuracies

I will focus on ideas rather than lots of numerical results.

Monte Carlo simulation

The general setting is that we have

- a set of random inputs ω
- used to compute an intermediate solution $U(\omega)$
- and then an output functional $f(U(\omega))$

$$\omega \quad \rightarrow \quad U \quad \rightarrow \quad f(U)$$

and we are interested in estimating $\mathbb{E}[f]$.

ω, U, f could all be infinite-dimensional, but I'll focus on the case where the final $f(U)$ is scalar.

(An example of infinite-dimensional $f(U)$ is estimation of the probability density function for U or some functional of U.)

Control variate

Classic approach to variance reduction: approximate $\mathbb{E}[f]$ using

$$N^{-1}\sum_{n=1}^{N}\left\{f(\omega^{(n)})-\lambda\left(g(\omega^{(n)})-\mathbb{E}[g]\right)\right\}$$

where

- control variate g has known expectation $\mathbb{E}[g]$
- g is well correlated with f, and optimal value for λ can be estimated by a few samples

For the optimal value of λ, the variance is reduced by factor $(1-\rho^2)$, where ρ is the correlation between f and g.

Two-level Monte Carlo

If we want to estimate $\mathbb{E}[f_1]$ but it is much cheaper to simulate $f_0 \approx f_1$, then since

$$\mathbb{E}[f_1] = \mathbb{E}[f_0] + \mathbb{E}[f_1 - f_0]$$

we can use the estimator

$$N_0^{-1} \sum_{n=1}^{N_0} f_0^{(0,n)} \;+\; N_1^{-1} \sum_{n=1}^{N_1} \left(f_1^{(1,n)} - f_0^{(1,n)} \right)$$

Two differences from standard control variate method:

- $\mathbb{E}[f_0]$ is not known, so has to be estimated
- $\lambda = 1$

Benefit: if $f_1 - f_0$ is small, won't need many samples to accurately estimate $\mathbb{E}[f_1 - f_0]$, so cost will be reduced greatly.

Multilevel Monte Carlo

Natural generalisation: given a sequence $f_0, f_1, \ldots, f_L$

$$\mathbb{E}[f_L] = \mathbb{E}[f_0] + \sum_{\ell=1}^{L} \mathbb{E}[f_\ell - f_{\ell-1}]$$

we can use the estimator

$$N_0^{-1} \sum_{n=1}^{N_0} f_0^{(0,n)} \; + \; \sum_{\ell=1}^{L} \left\{ N_\ell^{-1} \sum_{n=1}^{N_\ell} \left(f_\ell^{(\ell,n)} - f_{\ell-1}^{(\ell,n)} \right) \right\}$$

with independent estimation for each level

Multilevel Monte Carlo

If we define

- C_0, V_0 to be cost and variance of f_0
- C_ℓ, V_ℓ to be cost and variance of $f_\ell - f_{\ell-1}$

then the total cost is $\sum_{\ell=0}^{L} N_\ell \, C_\ell$ and the variance is $\sum_{\ell=0}^{L} N_\ell^{-1} V_\ell$.

Using a Lagrange multiplier μ^2 to minimise the cost for a fixed variance

$$\frac{\partial}{\partial N_\ell} \sum_{k=0}^{L} \left(N_k \, C_k + \mu^2 N_k^{-1} V_k \right) = 0$$

gives

$$N_\ell = \mu \sqrt{V_\ell / C_\ell} \quad \Longrightarrow \quad N_\ell \, C_\ell = \mu \sqrt{V_\ell \, C_\ell}$$

Multilevel Monte Carlo

Setting the total variance equal to ε^2 gives

$$\mu = \varepsilon^{-2} \left(\sum_{\ell=0}^{L} \sqrt{V_\ell \, C_\ell} \right)$$

and hence, the total cost is

$$\sum_{\ell=0}^{L} N_\ell \; C_\ell \;=\; \varepsilon^{-2} \left(\sum_{\ell=0}^{L} \sqrt{V_\ell C_\ell} \right)^2$$

in contrast to the standard cost which is approximately $\varepsilon^{-2} \, V_0 \, C_L$.

The MLMC cost savings are therefore:

- V_L/V_0, if $\sqrt{V_\ell C_\ell}$ increases with level
- C_0/C_L, if $\sqrt{V_\ell C_\ell}$ decreases with level

Multilevel Path Simulation

Motivated by computational finance applications, in 2006 I introduced MLMC for SDEs (stochastic differential equations).

$$\mathrm{d}S_t = a(S_t, t)\,\mathrm{d}t + b(S_t, t)\,\mathrm{d}W_t$$

Level ℓ corresponds to approximation using 2^ℓ timesteps, giving approximate payoff $\widehat{P}_\ell$.

Choice of finest level L depends on weak error (bias).

Multilevel decomposition gives

$$\mathbb{E}[\widehat{P}_L] = \mathbb{E}[\widehat{P}_0] + \sum_{\ell=1}^{L} \mathbb{E}[\widehat{P}_\ell - \widehat{P}_{\ell-1}]$$

Multilevel Monte Carlo

Simplest estimator for $\mathbb{E}[\widehat{P}_\ell - \widehat{P}_{\ell-1}]$ for $\ell > 0$ is

$$\widehat{Y}_\ell = N_\ell^{-1} \sum_{n=1}^{N_\ell} \left(\widehat{P}_\ell^{(n)} - \widehat{P}_{\ell-1}^{(n)} \right)$$

using same driving Brownian path for both levels.

Standard analysis gives MSE $= \left(\mathbb{E}[\widehat{P}_L] - \mathbb{E}[P] \right)^2 + \sum_{\ell=0}^{L} N_\ell^{-1} V_\ell$

To make RMS error less than ε

- choose L so that $\left(\mathbb{E}[\widehat{P}_L] - \mathbb{E}[P] \right)^2 < \frac{1}{2}\,\varepsilon^2$
- choose $N_\ell \propto \sqrt{V_\ell / C_\ell}$ so total variance is less than $\frac{1}{2}\,\varepsilon^2$

MLMC Theorem

(Slight generalisation of original version)

If there exist independent estimators $\widehat{Y}_\ell$ based on N_ℓ Monte Carlo samples, each costing C_ℓ, and positive constants $\alpha, \beta, \gamma, c_1, c_2, c_3$ such that $\alpha \geq \frac{1}{2}\min(\beta, \gamma)$ and

i) $\left|\mathbb{E}[\widehat{P}_\ell - P]\right| \leq c_1\, 2^{-\alpha\,\ell}$

ii) $\mathbb{E}[\widehat{Y}_\ell] = \begin{cases} \mathbb{E}[\widehat{P}_0], & \ell = 0 \\ \mathbb{E}[\widehat{P}_\ell - \widehat{P}_{\ell-1}], & \ell > 0 \end{cases}$

iii) $\mathbb{V}[\widehat{Y}_\ell] \leq c_2\, N_\ell^{-1} 2^{-\beta\,\ell}$

iv) $\mathbb{E}[C_\ell] \leq c_3\, 2^{\gamma\,\ell}$

MLMC Theorem

then there exists a positive constant c_4 such that for any $\varepsilon < 1$ there exist L and N_ℓ for which the multilevel estimator

$$\widehat{Y} = \sum_{\ell=0}^{L} \widehat{Y}_\ell,$$

has a mean-square-error with bound $\mathbb{E}\left[\left(\widehat{Y} - \mathbb{E}[P]\right)^2\right] < \varepsilon^2$

with an expected computational cost C with bound

$$C \le \begin{cases} c_4\, \varepsilon^{-2}, & \beta > \gamma, \\ c_4\, \varepsilon^{-2} (\log \varepsilon)^2, & \beta = \gamma, \\ c_4\, \varepsilon^{-2-(\gamma-\beta)/\alpha}, & 0 < \beta < \gamma. \end{cases}$$

MLMC Challenges

- not always obvious how to couple coarse and fine levels
  i.e. what does $\widehat{P}_\ell(\omega^{(n)}) - \widehat{P}_{\ell-1}(\omega^{(n)})$ mean?

- some creativity required to handle discontinuous functionals, where a small difference between the underlying coarse and fine simulations can produce an $O(1)$ difference in the output

- numerical analysis to determine the decay rate of V_ℓ can be tough

Brownian Diffusion SDEs

Brownian increments for coarse path obtained by summing increments for fine path – very simple and natural

I like the Milstein discretisation which gives first order strong convergence

$$\left(\mathbb{E}\left[\sup_{[0,T]} \|S_t - \widehat{S}_t\|^2\right]\right)^{1/2} = O(h)$$

so for payoffs which are Lipschitz functions of the final state we get

$$\widehat{P}_\ell - \widehat{P}_{\ell-1} = O(h_\ell)$$

and hence $V_\ell = O(h_\ell^2)$.

However, not so easy for lookback, digital and barrier options. Also, in multiple dimensions sometimes requires Lévy areas, but can be avoided by an antithetic treatment, (G & Szpruch, 2013).

Financial application

- basket of 5 underlying assets, modelled by Geometric Brownian Motion

$$\mathrm{d}S_i = r\, S_i\, \mathrm{d}t + \sigma_i\, S_i\, \mathrm{d}W_i$$

  with correlation between 5 driving Brownian motions

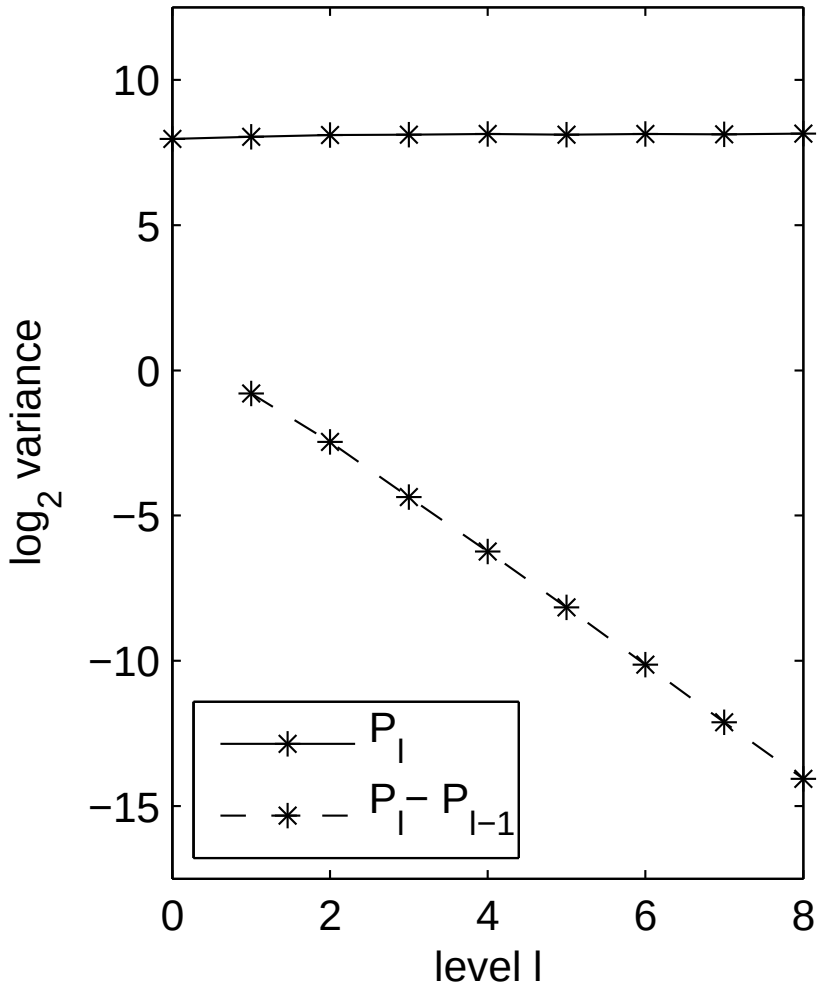
- Milstein numerical approximation

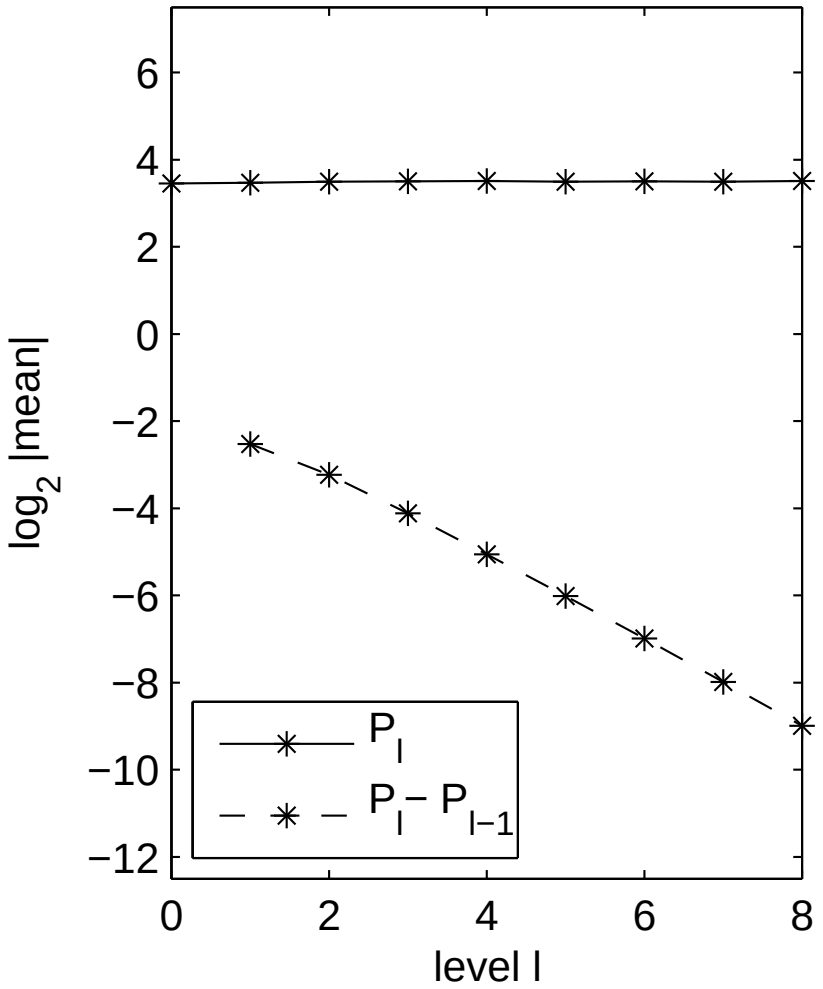
- standard call option is piecewise linear function of average at final time T

- digital call option is discontinuous function of average

Financial application

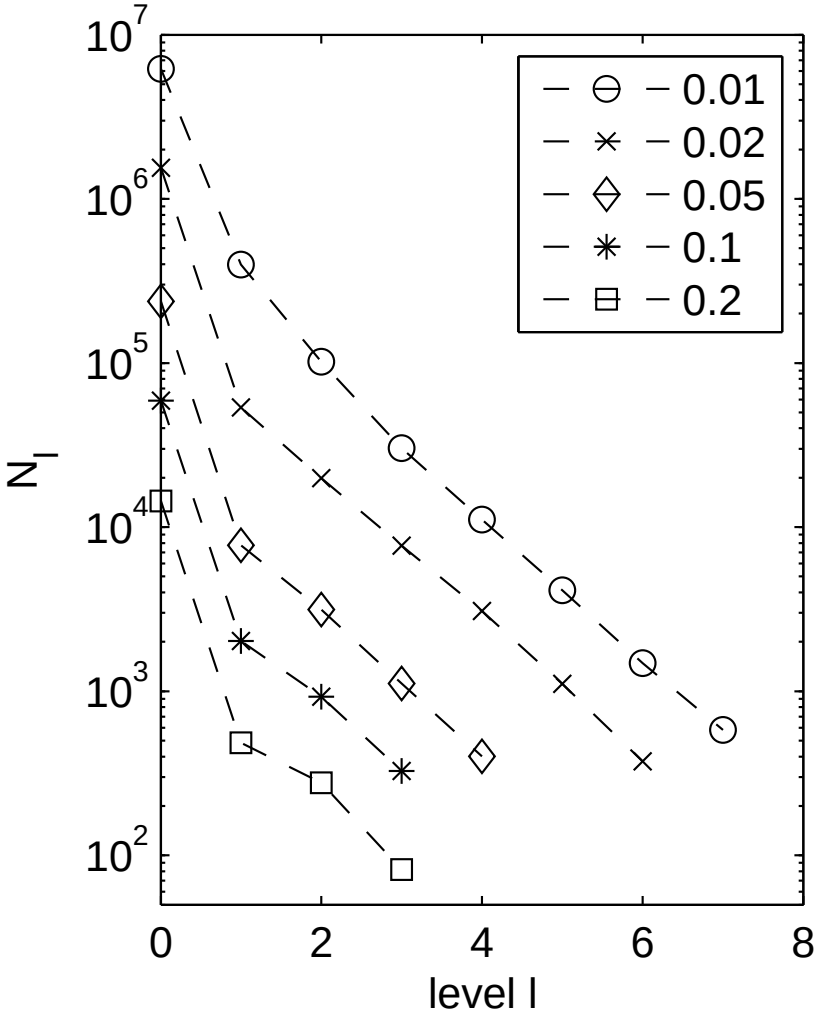
Standard call option:

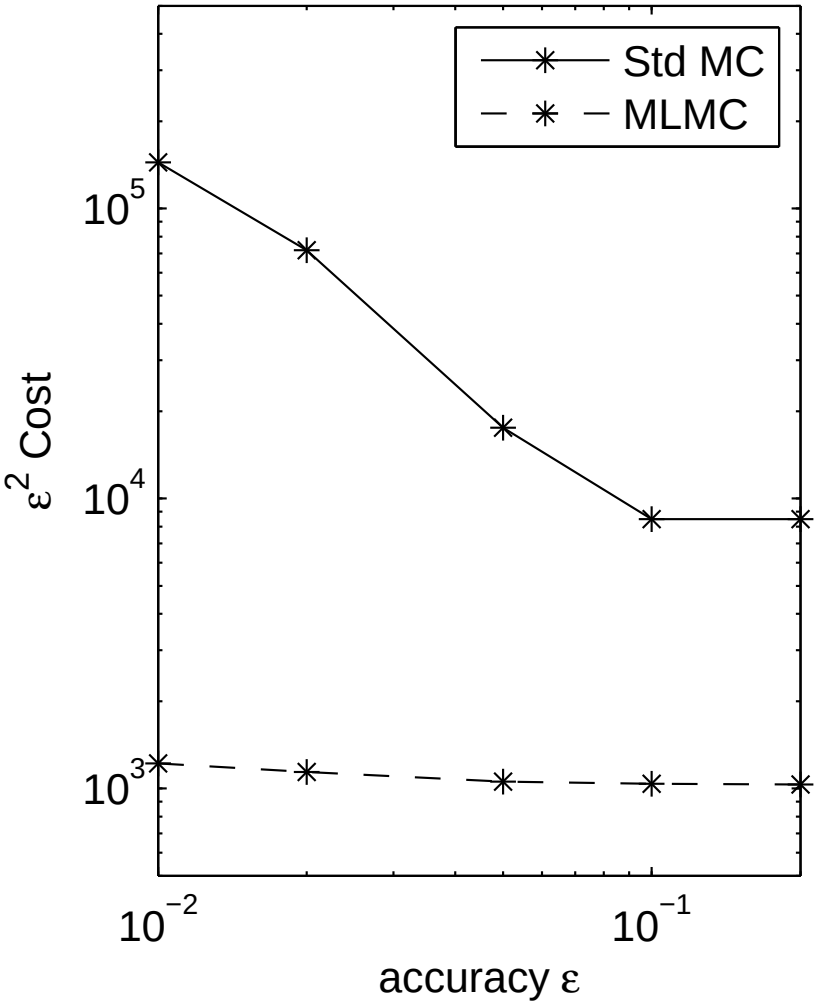




Financial application

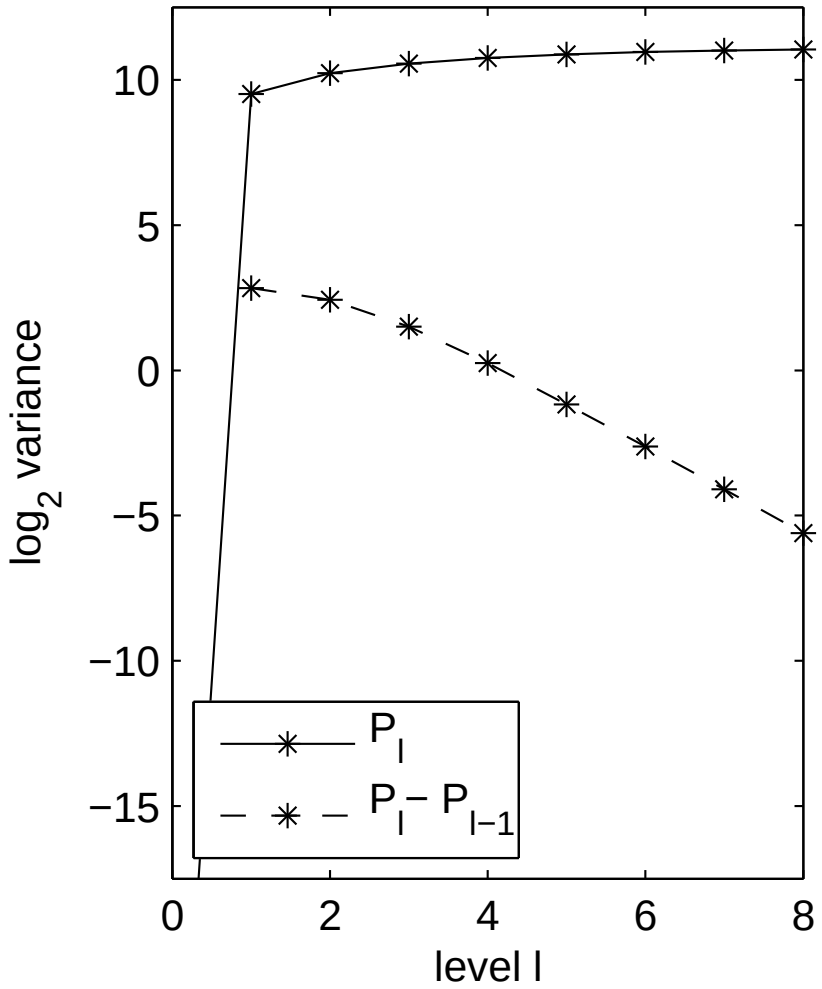
Standard call option:

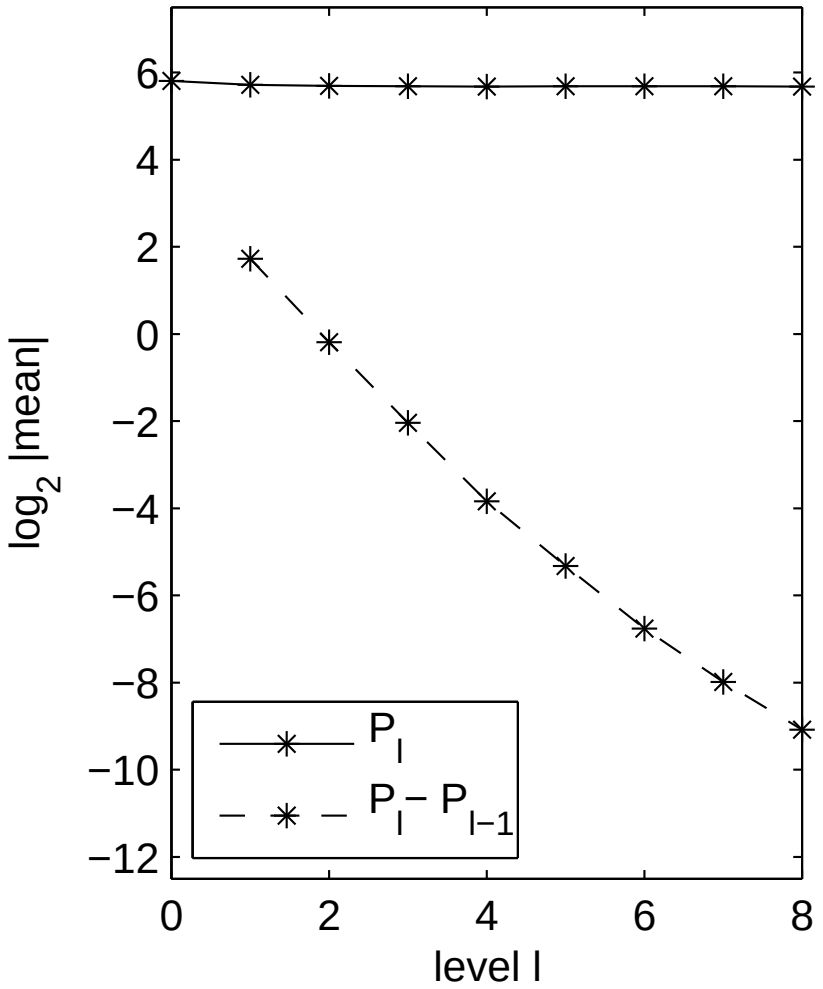




Financial application

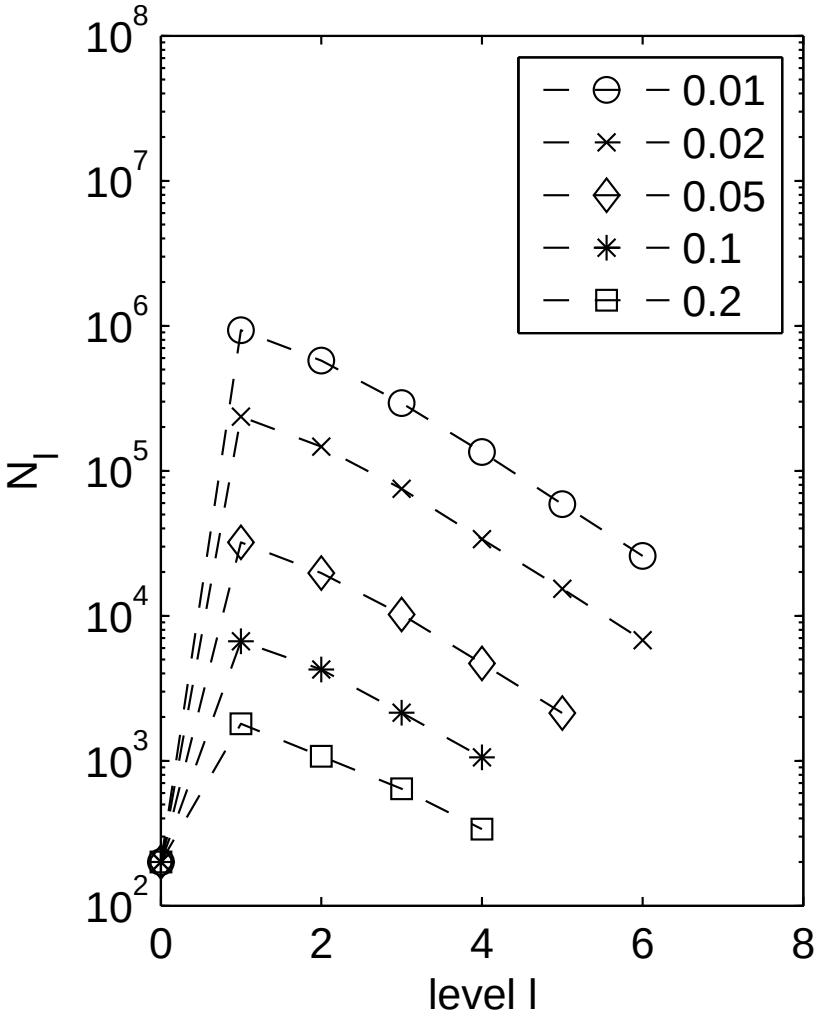
Digital call option:

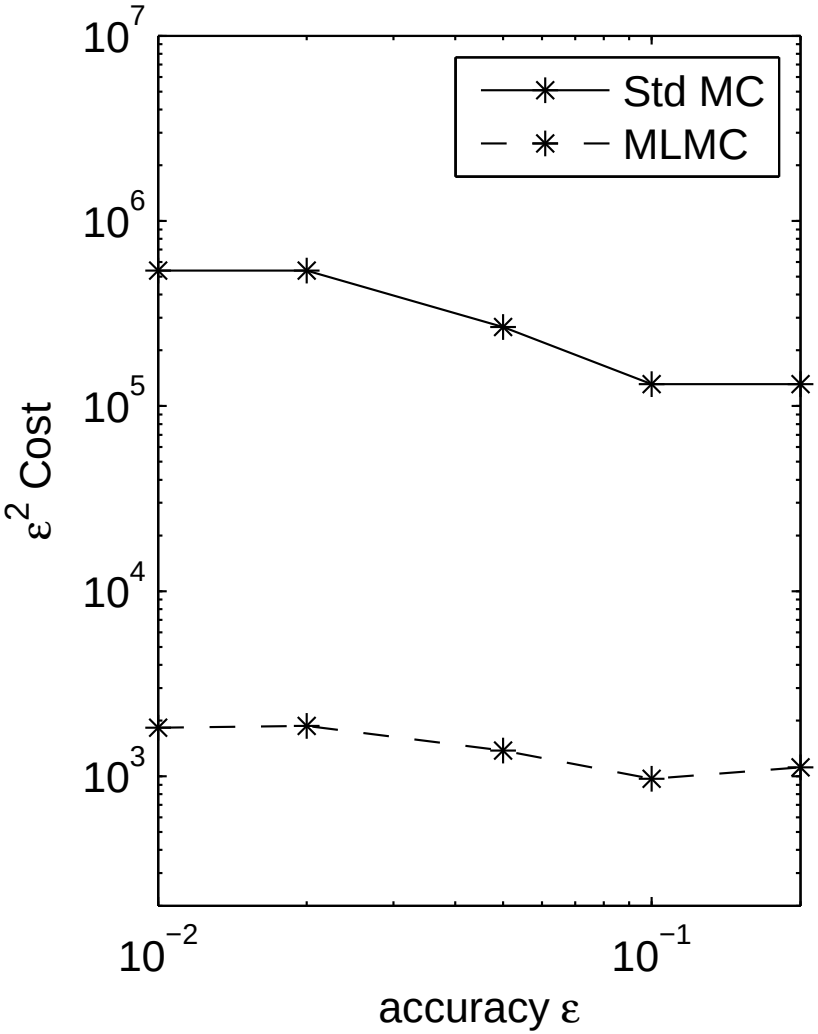




Financial application

Digital call option:





SPDEs

- quite natural application, with better cost savings than SDEs due to higher dimensionality

- range of applications
  - Graubner & Ritter (Darmstadt $\rightarrow$ Kaiserslautern) – parabolic
  - G, Reisinger (Oxford) – parabolic
  - Cliffe, G, Scheichl, Teckentrup (Bath/Nottingham/Oxford) – elliptic
  - Barth, Jenny, Lang, Meyer, Mishra, Müller, Schwab, Sukys, Zollinger (ETH Zürich) – elliptic, parabolic, hyperbolic
  - Harbrecht, Peters (Basel) – elliptic
  - Efendiev (Texas A&M) – numerical homogenization
  - Vidal-Codina, G, Peraire (MIT) – reduced basis approximation

Engineering Uncertainty Quantification

- consider 3D elliptic PDE, with uncertain boundary data
- use grid spacing proportional to $2^{-\ell}$ on level ℓ
- cost is $O(2^{+3\ell})$, if using an efficient multigrid solver
- 2nd order accuracy means that

$$\widehat{P}_\ell(\omega) - \widehat{P}(\omega) \approx c(\omega)\, 2^{-2\ell}$$

$$\Longrightarrow \quad \widehat{P}_{\ell-1}(\omega) - \widehat{P}_\ell(\omega) \approx 3\, c(\omega)\, 2^{-2\ell}$$

- hence, $\alpha = 2,\ \beta = 4,\ \gamma = 3$
- cost is $O(\varepsilon^{-2})$ to obtain ε RMS accuracy

Other MLMC research

- parametric integration, integral equations (Heinrich)
- multilevel QMC (Dick, G, Kuo, Scheichl, Schwab, Sloan)
- Lévy-driven SDEs (Dereich, Heidenreich)
- stochastic chemical reactions (Anderson & Higham, Tempone)
- mixed precision computation on FPGAs (Korn, Ritter, Wehn)
- MLMC for MCMC (Scheichl & Teckentrup, Schwab & Stuart)
- Coulomb collisions in plasma (Caflisch)
- nested simulation (Haji-Ali & Tempone, Hambly & Reisinger)
- MLMC + sparse grids = MIMC (Haji-Ali, Nobile, Tempone)

Adaptive timesteps

Interested in Langevin SDEs of the form

$$\mathrm{d}q_t = -\nabla V(q_t)\,\mathrm{d}t + \mathrm{d}W_t$$

which will be approximated by

$$q_{n+1} = q_n - \nabla V(q_n)\,h_n + \Delta W_n$$

using an adaptive timestep h_n.

Why is an adaptive timestep necessary?

Because we may have $V(q) = \frac{1}{4}q^4$ so $\nabla V = q^3$ which violates standard Lipschitz assumption for drift.

Adaptive timesteps

Why is this a problem?

Consider ODE:

$$dq = -q^3 \, \mathrm{d}t$$

with uniform timestep approximation

$$q_{n+1} = q_n - q_n^3 \, h$$

Then $|q(t)|$ should decrease in time, but we have

$$|q_{n+1}| = |q_n| \ \left|q_n^2 \, h - 1\right|$$

so $|q_n|$ increases in time if $h > 2q_n^{-2}$ leading to very dramatic blow-up.

In SDE, always a tiny probability of getting a big q_n, and this blow-up then makes all moments unbounded as $h \to 0$.

Adaptive timesteps

How does MLMC work with adaptive time-stepping?

Actually, surprisingly easy — on level ℓ use

$$h_n = 2^{-\ell}\, H(q_n)$$

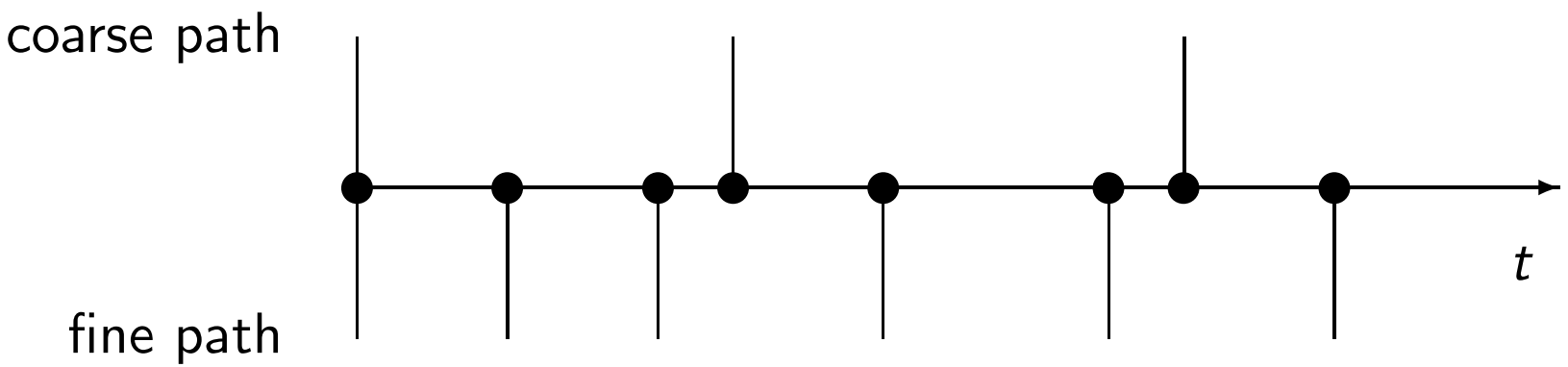
where $H(q_n)$ is adaptive choice to give largest mean-square stable timestep.

Coarse and fine paths each compute their own adaptive timesteps independently – this ensures the telescoping sum works correctly

But what is involved in coarse and fine paths using same driving Brownian motion?

Adaptive timesteps

As time proceeds, Brownian increments are generated as needed at discretisation times which are a union of coarse and fine path times:



The fact that the timesteps are not nested is not a problem – strong convergence still ensures a strong coupling between the coarse and fine paths, because both approximate the true path.

Adaptive timesteps

How is maximum mean-square stable timestep defined?

Subject of new numerical analysis research with Wei Fang, considering SDEs of the form

$$\mathrm{d}x_t = f(x_t)\,\mathrm{d}t + g(x_t)\,\mathrm{d}W_t$$

These are well-posed if the drift satisfies a one-sided Lipschitz condition giving

$$f^T x \;\leq\; \alpha\,(1+\|x\|)^2$$

for some $\alpha \geq 0$, and the volatility satisfies the usual linear growth condition

$$\|g\| \leq \beta\,(1+\|x\|)$$

for some $\beta > 0$.

Adaptive timesteps

We prove that for a given time interval $[0, T]$, all moments

$$\mathbb{E}\left[\sup_t \|\hat{x}_t\|^p\right]$$

of the Euler-Maruyama approximation have a bound C_p, independent of the definition of the adaptive timestep provided it satisfies the condition

$$f^T(\hat{x}_n)\,\hat{x}_n + \tfrac{1}{2}\,h_n\|f(\hat{x}_n)\|^2 \;\le\; \hat{\alpha}\,(1+\|\hat{x}_n\|)^2$$

for some $\hat{\alpha} \ge 0$.

In the scalar case, this leads to a timestep of the form

$$h_n = \frac{1+|\hat{x}_n|}{1+|f(\hat{x}_n)|} \;=\; O\left(\frac{1}{|f'(\hat{x}_n)|}\right) \quad \text{when } |\hat{x}_n| \gg 1$$

Adaptive timesteps

We also prove that the adaptive Euler-Maruyama method gives the standard strong order of convergence, in the sense that

$$\mathbb{E}\left[\sup_t \|\hat{x}_t - x_t\|^p\right]^{1/p} = O(\mathbb{E}[\text{cost}]^{-1/2})$$

where $\mathbb{E}[\text{cost}]^{-1}$ has replaced the usual uniform timestep h.

The conclusion from this is that we get the usual MLMC complexity for these SDEs which violate the usual Lipschitz conditions, giving an alternative to other methods (tamed Euler, implicit-drift Euler)

MLMC for reflected diffusions

Joint research with Kavita Ramanan (Brown University)

Motivation comes from network queue analysis, which can be approximated by a reflected Brownian diffusion within a domain D, so the SDE is of the form

$$\mathrm{d}x_t = a(x_t)\,\mathrm{d}t + b\,\mathrm{d}W_t + \nu(x_t)\,\mathrm{d}L_t$$

where L_t is a "local time" which increases when x_t is on the boundary ∂D.

$\nu(x_t)$ can be normal to the boundary (pointing inwards), but in other cases it is not and reflection from the boundary includes a tangential motion.

A penalised version is

$$\begin{aligned} \mathrm{d}x_t &= a(x_t)\,\mathrm{d}t + b\,\mathrm{d}W_t + \nu(x_t)\,\mathrm{d}L_t \\ \mathrm{d}L_t &= -\lambda\,\min(0, d(x_t))\,\mathrm{d}t \end{aligned}$$

where $d(x_t)$ is signed distance to the boundary – negative means outside.

MLMC for reflected diffusions

3 different numerical treatments:

- projection: predictor step:

$$\widehat{X}^{(p)} = \widehat{X}_{t_n} + a(\widehat{X}_{t_n}, t_n)\, h_n + b\, \Delta W_n,$$

  followed by correction step

$$\widehat{X}_{t_{n+1}} = \widehat{X}^{(p)} + \nu(\widehat{X}^{(p)})\ \Delta\widehat{L}_n,$$

  with $\Delta\widehat{L}_n > 0$ if needed to put $\widehat{X}_{t_{n+1}}$ on boundary

- reflection: similar but with double the value for $\Delta\widehat{L}_n$ – can give improved weak convergence

- penalised: Euler-Maruyama approximation of penalised SDE

MLMC for reflected diffusions

Concern:

- because b is uniform, Euler-Maruyama method corresponds to first order Milstein scheme, suggesting an $O(h)$ strong error
- however, all treatments of boundary reflection lead to a strong error which is $O(h^{1/2})$ – this is based primarily on empirical evidence, with only limited supporting theory

Idea:

- use adaptive timesteps, with level ℓ timestep given by

$$\max\left(2^{-2\ell}h_0, \min\left(2^{-\ell}h_0, (d/((\ell+3)\,\|b\|_2)^2\right)\right).$$

  based on distance d to boundary.

MLMC for reflected diffusions

This max-min definition leads to 3 zones:

- a boundary zone where $h = 2^{-2\ell} h_0$
- an interior zone where $h = 2^{-\ell} h_0$
- an intermediate zone where $(\ell+3)\sqrt{h}\|b\|_2 = d$

As $\ell \to \infty$, there is a very high probability that no reflections take place from the interior or intermediate zones.

- boundary error is $O(\sqrt{2^{-2\ell} h_0}\,) = O(2^{-\ell})$
- interior error is $O(2^{-\ell} h_0) = O(2^{-\ell})$
- overall, strong error is $O(2^{-\ell}) \implies$ MLMC variance $= O(2^{-2\ell})$.

MLMC for reflected diffusions

Current theoretical analysis:

- if strong error is $O(\sqrt{h}\,)$ for uniform timestep then the MLMC variance is $O(2^{-2\ell})$ for Lipschitz functionals.
- the expected cost is $o(2^{-(1-\delta)\ell})$ for any $0 < \delta \ll 1$
- regarding MLMC theory, this gives $\beta = 2, \gamma \approx 1$, so the complexity is $O(\varepsilon^{-2})$ for ε r.m.s. error

Numerical analysis challenge:

- prove that the strong error is $O(\sqrt{h}\,)$ for uniform timestep with oblique reflections, preferably for generalised penalisation method for polygonal boundaries

MLMC with inaccurate computations

Joint research about to start with Klaus Ritter (TU Kaiserslautern) – no results or analysis yet!

We are interested in three sources of inaccuracy, other than the usual timestep discretisation errors:

- reduced number of bits in uniform random number generation
- reduced accuracy in converting uniform r.v.'s to Normal
- reduced precision in performing path calculations

Objectives:

- design good MLMC estimators
- analyse variance behaviour and determine complexity (based on some appropriate cost model)

Uniform random number generation

Standard theory is based on uniform random variables U in $(0,1)$.

Computational implementations are close to this ideal with most uniform generators giving values of the form

$$U = (k + \tfrac{1}{2})/m$$

where m is fixed and large (e.g. $m = 2^{32}$) and k is uniformly distributed in $\{0, 1, 2, \ldots, m-1\}$.

We are concerned with what happens when m is small (e.g. $m = 8$) which will clearly lead to errors.

Conversion to Normals

If $U \in (0,1)$, it can be converted to a standard Normal random variable by

$$Z = \Phi^{-1}(U).$$

Alternatively, if $U \in (-1,+1)$, then it can be converted by

$$Z \;=\; \Phi^{-1}(\tfrac{1}{2}(U+1)) \;\equiv\; \mathrm{erfinv}(U),$$

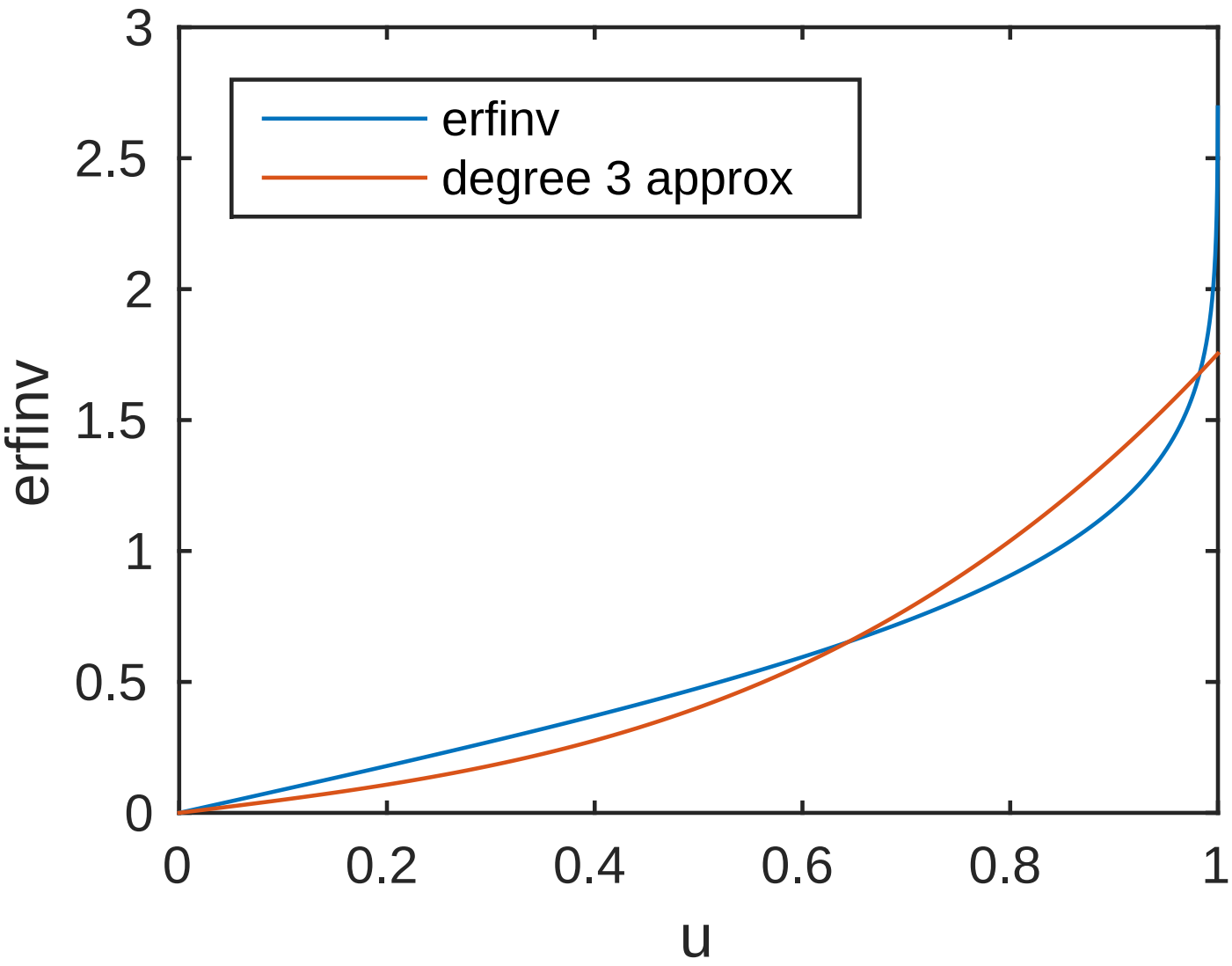
where $\mathrm{erfinv}()$ is the inverse error function

In the latter case, we can approximate the conversion by

$$\widetilde{Z} = \widetilde{\mathrm{erfinv}}(U)$$

where $\widetilde{\mathrm{erfinv}}()$ is an (inaccurate) approximation to $\mathrm{erfinv}()$.

Conversion to Normals



Low precision computations

Standard C/C++ computations have a choice of two levels of floating point precision:

- single (`float`) – 32-bit with 24-bit mantissa
- double (`double`) – 64-bit with 53-bit mantissa

In each case, only finitely many numbers can be represented – other numbers are rounded to nearest representable value

1 ulp (unit of least precision) is the difference between one representable value and its nearest neighbour

$$\text{relative error} \equiv 1 \text{ ulp/value} \approx 2^{-\text{mantissa}} \approx \begin{cases} 10^{-7}, & \text{single} \\ 10^{-16}, & \text{double} \end{cases}$$

Low precision computations

Most people do calculations in double precision, so why should we care?

On CPUs, single precision is twice as fast as double, if code is vectorised.

On GPUs, the difference can be a factor 5–10, and there is an extra factor 2 on the latest NVIDIA GPUs if you use half-precision with a 11-bit mantissa.

Even more extreme, FPGAs (field-programmable gate arrays) allow arbitrary fixed-point and floating point arithmetic, with huge potential savings.

MLMC reminder

For the telescoping sum to work correctly, we need same value for $\mathbb{E}[\widehat{P}_\ell]$ in both $\mathbb{E}[\widehat{P}_{\ell+1} - \widehat{P}_\ell]$ and $\mathbb{E}[\widehat{P}_\ell - \widehat{P}_{\ell-1}]$.

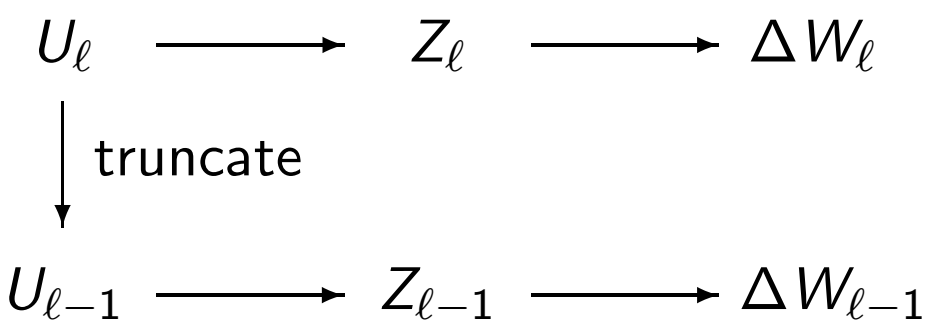
i.e. the level ℓ approximation has the same expected value regardless of whether it is the finer of the two levels or the coarser.

This is usually not a problem, but here we will need to be a little careful.

MLMC algorithm 1

For the first algorithm, we use the same number of timesteps on all levels, and just vary the accuracy:

- uniform r.v. generator with possibly different bit length $\log_2 m_\ell$ on each level – if m_ℓ is a power of 2, then chopping extra bits can truncate U_ℓ to $U_{\ell-1}$ and respect telescoping sum
- different erfinv_ℓ approximation on each level
- different precision on each level



MLMC algorithm 2

For the second algorithm, we also vary the number of timesteps on each level.

Problem: how do we couple the simulations on different levels?

In the standard MLMC algorithm for SDEs using 2^ℓ timesteps on level ℓ, we generate Brownian increments ΔW_n on the finer level, then sum them in pairs to get the Brownian increments for the coarser level.

Since

$$\sqrt{h}\; N(0,1) + \sqrt{h}\; N(0,1) \;\sim\; \sqrt{2h}\; N(0,1)$$

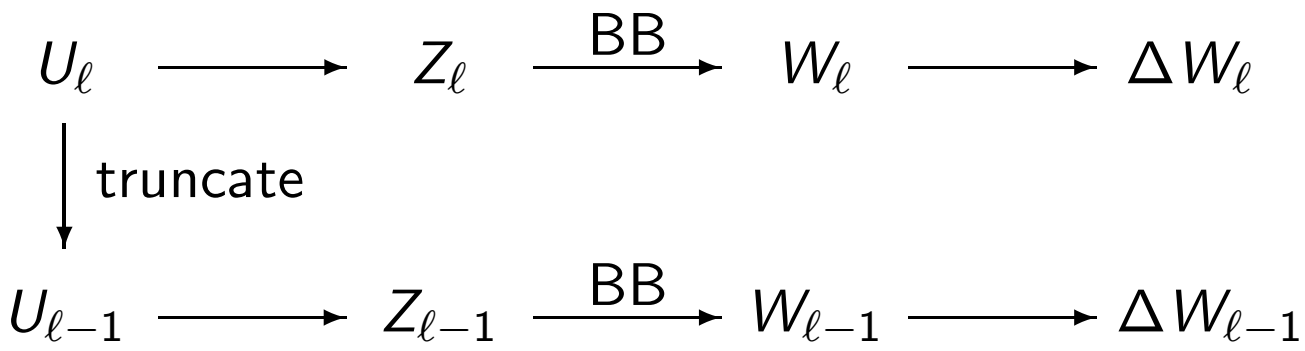
this is equivalent in distribution to directly generating the Brownian increments on the coarser level – hence the telescoping sum is OK.

MLMC algorithm 2

However, summing the Brownian increments in pairs does not result in an equivalent distribution when we have the new inaccuracies.

Thus, we would violate the telescoping sum if we did it this way.

Solution: use a Brownian Bridge construction!



This guarantees the same distribution for Brownian increments on level ℓ, regardless of whether it is the finer or coarser level, because U_ℓ has same uniform discrete distribution.

Conclusions

- multilevel idea is very simple
- challenge can be how to apply it in new situations
- discontinuous output functions can cause problems, but there is a lot of experience now in coping with this
- there are also "tricks" which can be used in situations with poor strong convergence
- being used for an increasingly wide range of applications; biggest computational savings when coarsest (helpful) approximation is much cheaper than finest
- currently, getting at least $100\times$ savings for SPDEs and stochastic chemical reaction simulations

References

Webpage for my research/papers:

`people.maths.ox.ac.uk/gilesm/mlmc.html`

Webpage for new 70-page *Acta Numerica* review and MATLAB test codes:

`people.maths.ox.ac.uk/gilesm/acta/`

– contains references to almost all MLMC research, including some very early related work by Achi Brandt

Webpage for new MLMC code in MATLAB, C++ and Python:

`people.maths.ox.ac.uk/gilesm/mlmc/`

MLMC Community

Webpage: people.maths.ox.ac.uk/gilesm/mlmc_community.html

Abo Academi (Avikainen) – numerical analysis
Basel (Harbrecht) – elliptic SPDEs, sparse grids
Bath (Kyprianou, Müller, Scheichl, Shardlow, Yates) – elliptic SPDEs, MCMC, Lévy-driven SDEs, stochastic chemical modelling
Chalmers (Lang) – SPDEs
Duisburg (Belomestny) – Bermudan and American options
Edinburgh (Davie, Szpruch) – SDEs, numerical analysis
EPFL (Abdulle) – stiff SDEs and SPDEs
ETH Zürich (Jenny, Jentzen, Schwab) – SPDEs, multilevel QMC
Frankfurt (Gerstner, Kloeden) – numerical analysis, fractional Brownian motion
Fraunhofer ITWM (Iliev) – SPDEs in engineering
Hong Kong (Chen) – Brownian meanders, nested simulation in finance
IIT Chicago (Hickernell) – SDEs, infinite-dimensional integration, complexity analysis
Kaiserslautern (Heinrich, Korn, Ritter) – finance, SDEs, parametric integration, complexity analysis
KAUST (Tempone, von Schwerin) – adaptive time-stepping, stochastic chemical modelling
Kiel (Gnewuch) – randomized multilevel QMC
LPMA (Frikha, Lemaire, Pagès) – numerical analysis, multilevel extrapolation, finance applications
Mannheim (Neuenkirch) – numerical analysis, fractional Brownian motion
MIT (Peraire) – uncertainty quantification, SPDEs
Munich (Hutzenthaler) – numerical analysis
Oxford (Baker, Giles, Hambly, Reisinger) – SDEs, SPDEs, numerical analysis, finance applications, stochastic chemical modelling
Passau (Müller-Gronbach) – infinite-dimensional integration, complexity analysis
Stanford (Glynn) – numerical analysis, randomized multilevel
Strathclyde (Higham, Mao) – numerical analysis, exit times, stochastic chemical modelling
Stuttgart (Barth) – SPDEs
Texas A&M (Efendiev) – SPDEs in engineering
UCLA (Caflisch) – Coulomb collisions in physics
UNSW (Dick, Kuo, Sloan) – multilevel QMC
UTS (Baldeaux) – multilevel QMC
Warwick (Stuart, Teckentrup) – MCMC for SPDEs
WIAS (Friz, Schoenmakers) – rough paths, fractional Brownian motion, Bermudan options
Wisconsin (Anderson) – numerical analysis, stochastic chemical modelling