Smoking adjoints, part II: fast Monte Carlo Greeks

Mike Giles

mike.giles@maths.ox.ac.uk

Oxford University Mathematical Institute

Oxford-Man Institute of Quantitative Finance

Quant Congress ’08

“Smoking Adjoints”

Paper with Paul Glasserman in *Risk* in 2006 on the use of adjoints in computing pathwise sensitivities attracted a lot of interest, and questions:

- what is involved in practice in creating an adjoint code, and can it be simplified?
- what about barriers and American options?
- do we really have to differentiate the payoff?
- what about non-differentiable payoffs?

Outline

- standard and adjoint pathwise sensitivities
- use of automatic differentiation ideas/tools
- barriers and American options
- “vibrato” Monte Carlo for non-differentiable payoffs

Generic Problem

Stochastic differential equation with general drift and volatility terms:

$$\mathrm{d}S(t) = a(S,t)\,\mathrm{d}t + b(S,t)\,\mathrm{d}W(t)$$

For a simple European option we want to compute the expected discounted payoff value dependent on the terminal state:

$$V = \mathbb{E}[f(S(T))]$$

Note: the drift and volatility functions are almost always differentiable, but the payoff $f(S)$ is often not.

Numerical discretisation

Euler discretisation with timestep h:

$$\widehat{S}_{n+1} = \widehat{S}_n + a(\widehat{S}_n, t_n)\, h + b(\widehat{S}_n, t_n)\, \Delta W_n$$

gives approximate expectation

$$\widehat{V} = \mathbb{E}\left[f(\widehat{S}_N)\right]$$

Pathwise sensitivity

Differentiating with respect to an arbitrary parameter θ gives

$$\frac{\partial \widehat{S}_{n+1}}{\partial \theta} = \left(1 + \frac{\partial a}{\partial S}\, h + \frac{\partial b}{\partial S}\, \Delta W_n \right) \frac{\partial \widehat{S}_n}{\partial \theta} + \frac{\partial a}{\partial \theta}\, h + \frac{\partial b}{\partial \theta}\, \Delta W_n$$

leading to

$$\frac{\partial \widehat{V}}{\partial \theta} = \mathbb{E}\left[\frac{\partial f}{\partial S}(\widehat{S}_N)\; \frac{\partial \widehat{S}_N}{\partial \theta} \right]$$

This is valid if $a(S,\theta)$ and $b(S,\theta)$ are differentiable, and the payoff $f(S)$ is continuous and piecewise differentiable.

Adjoint sensitivity

The adjoint approach is an efficient implementation of pathwise sensitivities – it gives exactly the same value.

Consider a process in which a vector input α leads to a final state vector S which is used to compute a scalar payoff P

$$\alpha \quad \longrightarrow \quad S \quad \longrightarrow \quad P$$

Taking $\dot{\alpha}, \dot{S}, \dot{P}$ to be the derivatives w.r.t. j^{th} component of α, then

$$\dot{S} = \frac{\partial S}{\partial \alpha}\,\dot{\alpha}, \qquad \dot{P} = \frac{\partial P}{\partial S}\,\dot{S},$$

and hence

$$\dot{P} = \frac{\partial P}{\partial S}\,\frac{\partial S}{\partial \alpha}\,\dot{\alpha}.$$

Adjoint sensitivity

Alternatively, defining $\overline{\alpha}, \overline{S}, \overline{P}$ to be the derivatives of P with respect to α, S, P, then

$$\overline{\alpha} \stackrel{\text{def}}{=} \left(\frac{\partial P}{\partial \alpha}\right)^T = \left(\frac{\partial P}{\partial S}\,\frac{\partial S}{\partial \alpha}\right)^T = \left(\frac{\partial S}{\partial \alpha}\right)^T \overline{S},$$

and similarly

$$\overline{S} = \left(\frac{\partial P}{\partial S}\right)^T \overline{P},$$

giving

$$\overline{\alpha} = \left(\frac{\partial S}{\partial \alpha}\right)^T \left(\frac{\partial P}{\partial S}\right)^T \overline{P}.$$

Adjoint sensitivity

The two are mathematically equivalent, since

$$\dot{P} = \frac{\partial P}{\partial \alpha}\, \dot{\alpha} = \overline{\alpha}^T \dot{\alpha} = \overline{\alpha}_j$$

but the adjoint approach is much cheaper because a single calculation gives $\overline{\alpha}$, the sensitivity of P to each one of the elements of α.

- standard approach: cost proportional to the number of Greeks
- adjoint approach: cost independent
- crossover point: 4 – 6 Greeks?

Adjoint sensitivity

Note that the standard approach goes forward

$$\dot{\alpha} \quad \longrightarrow \quad \dot{S} \quad \longrightarrow \quad \dot{P}$$

while the adjoint approach does the reverse

$$\overline{\alpha} \quad \longleftarrow \quad \overline{S} \quad \longleftarrow \quad \overline{P}.$$

These correspond to the forward and reverse modes of Automatic Differentiation (AD).

“Smoking Adjoints” paper extended this to multiple timesteps in the path calculation — instead, we’ll extend it to the steps in a whole computer program.

Automatic Differentiation

A computer instruction creates an additional new value:

$$\mathbf{u}^n \;=\; \mathbf{f}^n(\mathbf{u}^{n-1}) \;\equiv\; \begin{pmatrix} \mathbf{u}^{n-1} \\ f_n(\mathbf{u}^{n-1}) \end{pmatrix},$$

A computer program is the composition of N such steps:

$$\mathbf{u}^N = \mathbf{f}^N \circ \mathbf{f}^{N-1} \circ \ldots \circ \mathbf{f}^2 \circ \mathbf{f}^1(\mathbf{u}^0)$$

Differentiation w.r.t. one element of the input vector gives

$$\dot{\mathbf{u}}^N = D^N\, D^{N-1} \ldots\, D^2\, D^1\, \dot{\mathbf{u}}^0, \qquad D^n \equiv \begin{pmatrix} I^{n-1} \\ \partial f_n / \partial \mathbf{u}^{n-1} \end{pmatrix}$$

Automatic Differentiation

In reverse mode, we consider the sensitivity of one element of the output vector, to get

$$\left(\overline{\mathbf{u}}^{n-1}\right)^T \equiv \frac{\partial u_i^N}{\partial \mathbf{u}^{n-1}} = \frac{\partial u_i^N}{\partial \mathbf{u}^n}\,\frac{\partial \mathbf{u}^n}{\partial \mathbf{u}^{n-1}} = \left(\overline{\mathbf{u}}^n\right)^T D^n,$$

$$\Longrightarrow \quad \overline{\mathbf{u}}^{n-1} = \left(D^n\right)^T \overline{\mathbf{u}}^n.$$

and hence

$$\overline{\mathbf{u}}^0 = \left(D^1\right)^T \left(D^2\right)^T \ldots \left(D^{N-1}\right)^T \left(D^N\right)^T \overline{\mathbf{u}}^N.$$

Note: need to go forward through original calculation to compute/store the D^n, then go in reverse to compute $\overline{\mathbf{u}}^n$

Automatic Differentiation

This gives a prescriptive algorithm for reverse mode differentiation.

Again the reverse mode is much more efficient if we want the sensitivity of a single output to multiple inputs.

Key result is that the cost of the reverse mode is at worst a factor 4 greater than the cost of the original calculation, regardless of how many sensitivities are being computed!

The storage of the D^n is minor for SDEs – much more of a concern for PDEs.

Automatic Differentiation

Manual implementation of the forward/reverse mode algorithms is possible but a little tedious.

Fortunately, automated tools have been developed, following one of two approaches:

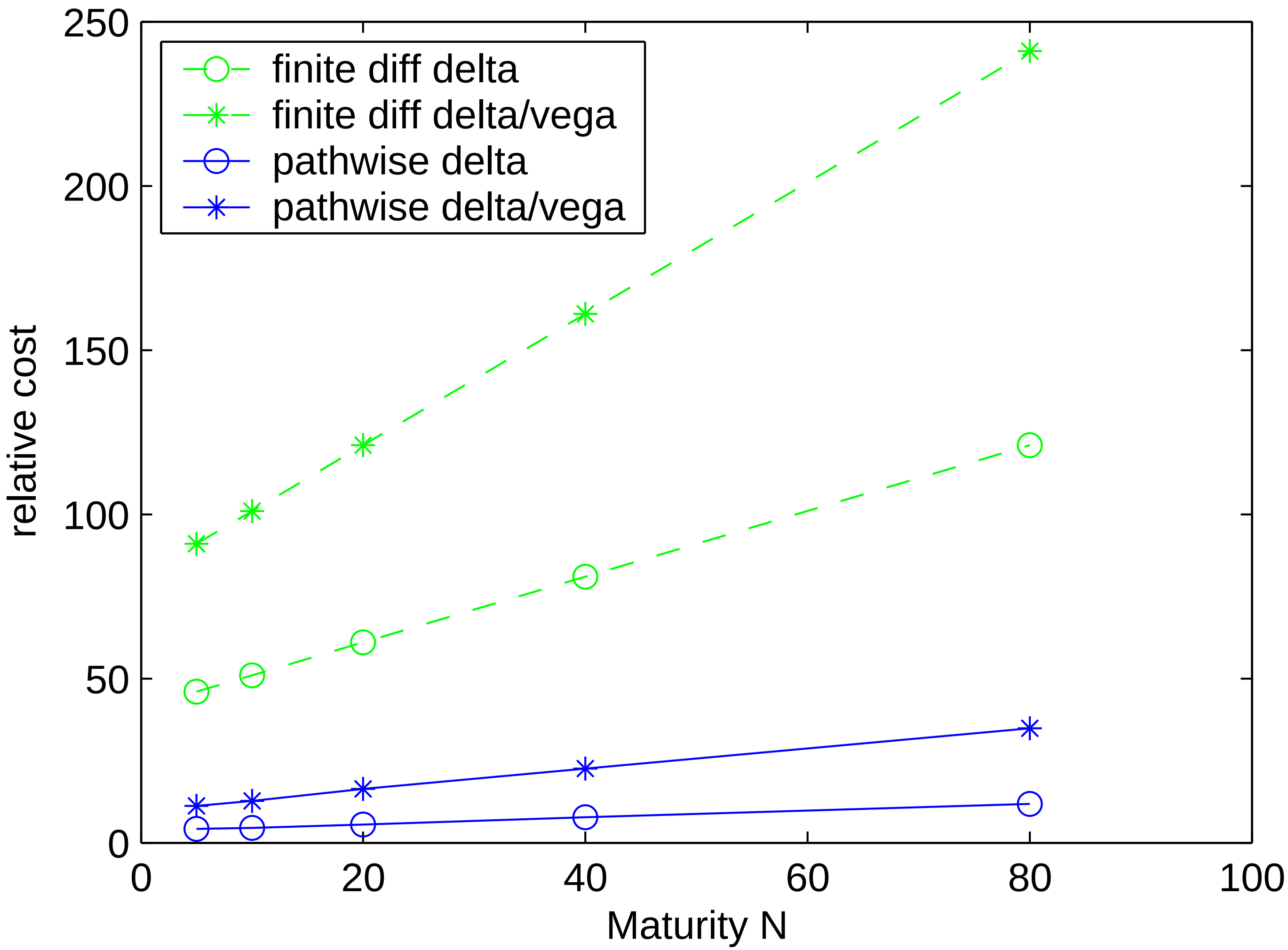
- operator overloading (ADOL-C, FADBAD++)
  - well developed, robust, not very efficient
- source code transformation (Tapenade, TAC)
  - still under development for C, major challenges posed by C++, often close to efficiency of hand-coded adjoints

LIBOR Application

- testcase from “Smoking Adjoints” paper
- test problem performs N timesteps with a vector of $N+40$ forward rates, and computes the $N+40$ deltas and vegas for a portfolio of swaptions
- hand-coded using the ideas from automatic differentiation

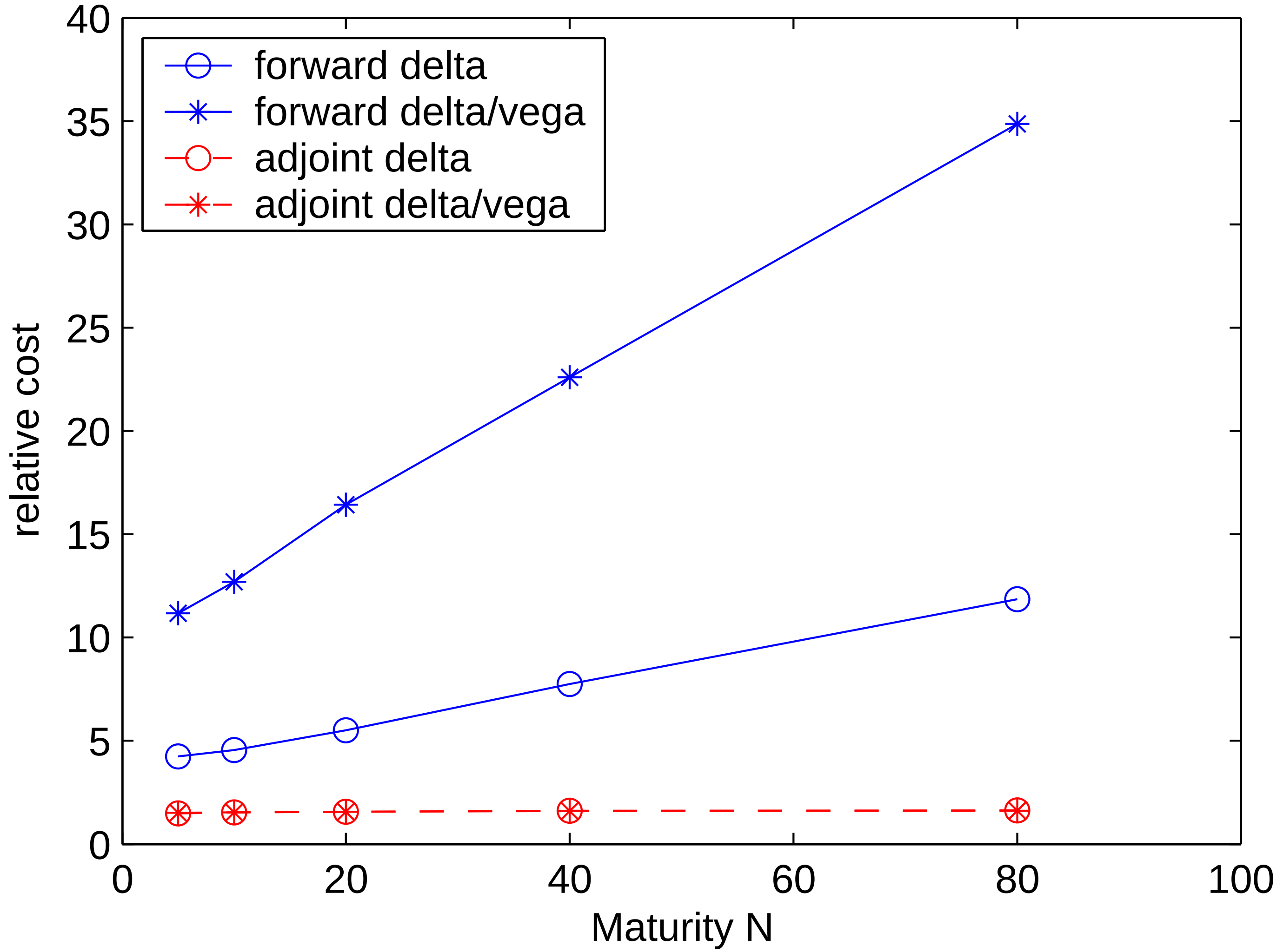
LIBOR Application

Finite differences versus forward pathwise sensitivities:



LIBOR Application

Hand-coded forward versus adjoint pathwise sensitivities:



Barrier options

The big limitation of pathwise sensitivity calculations (and their adjoint equivalent) is that the validity depends on the payoff being continuous and piecewise differentiable.

What about a barrier option?

A simple implementation uses the Euler discretisation and defines the numerical payoff of a down-and-out call as

$$\widehat{P} = e^{-rT} \, (\widehat{S}_N - K)^+ \;\; \mathbf{1}(\min_n \widehat{S}_n > B)$$

This is discontinuous when the smallest $\widehat{S}_n$ crosses B.

Barrier options

It is better to define the payoff as

$$\widehat{P} = e^{-rT}\ (\widehat{S}_N - K)^+\ \prod_n (1 - p_n)$$

where

$$p_n = \exp\left(\frac{-2\,(\widehat{S}_n - B)^+ (\widehat{S}_{n+1} - B)^+}{b_n^2\, h} \right)$$

is the approximate probability that the path crosses the barrier in time interval $[t_n, t_{n+1}]$, conditional on $\widehat{S}_n, \widehat{S}_{n+1}$.

This is more accurate (bias is $O(h)$ instead of $O(h^{1/2})$), and continuous as $\widehat{S}_n$ crosses the barrier.

American options

Longstaff-Schwartz approach:

- calculate a set of paths to maturity
- step backwards in time from maturity
  - calculate least-squares approximation of continuation value
  - if best to exercise, set value equal to exercise value
  - otherwise, discount existing path value

American options

- small discontinuity because of difference between continuation value and discounted path value
- suspect the error due to this is small (negligible?) but haven't yet investigated
- pathwise sensitivity analysis is equivalent to fixing the exercise boundary, which has zero effect in PDE formulation
- could be viewed as naive barrier option treatment; use probabilistic treatment to regain continuity
- alternatively, use Tsitsiklis & Van Roy treatment; set path value to approximate continuation value

Vibrato Monte Carlo

One remaining problem – what if payoff is not differentiable?

- Likelihood Ratio Method (LRM)
  - estimator variance proportional to h^{-1}
- Malliavin calculus
  - recent paper by Glasserman and Chen shows it can be viewed as a pathwise/LRM hybrid
  - might be good choice when few Greeks needed
- new “vibrato” Monte Carlo idea
  - also a pathwise/LRM hybrid
  - variance proportional to $h^{-1/2}$
  - efficient adjoint implementation

Vibrato Monte Carlo

- new idea, based on conditional expectation for a simple digital option (Glasserman's MC book)
- output of each SDE path calculation becomes a narrow (multivariate) Normal distribution
- combine pathwise sensitivity for the differentiable SDE, with LRM for the non-differentiable payoff
- avoiding the differentiation of the payoff also simplifies the implementation in real-world setting

Vibrato Monte Carlo

Final timestep of Euler path discretisation is

$$\widehat{S}_N = \widehat{S}_{N-1} + a(\widehat{S}_{N-1}, t_{N-1})\, h + b(\widehat{S}_{N-1}, t_{N-1})\, \Delta W_{N-1}$$

Instead of using random number generator to get a value for ΔW_{N-1}, consider the whole distribution of possible values, so $\widehat{S}_N$ has a Normal distribution with mean

$$\mu(W) = \widehat{S}_{N-1} + a(\widehat{S}_{N-1}, t_{N-1})\, h$$

and standard deviation

$$\sigma(W) = b(\widehat{S}_{N-1}, t_{N-1})\, \sqrt{h}$$

where $W \equiv (\Delta W_0, \Delta W_1, \ldots \Delta W_{N-2})$.

Vibrato Monte Carlo

For a particular path given by a particular vector W, the expected payoff is

$$\mathbb{E}_Z[f(\mu+\sigma Z)]$$

where Z is a Normal random variable with zero mean and unit variance.

Averaging over all W then gives the same overall expectation as before.

Note also that, for given W, $\widehat{S}_N$ has a Normal distribution with

$$p_S(\widehat{S}_N) = \frac{1}{\sqrt{2\pi}\,\sigma}\,\exp\left(-\frac{(\widehat{S}_N-\mu)^2}{2\,\sigma^2}\right)$$

Vibrato Monte Carlo

In the case of a simple digital call with strike K, can use the analytic solution

$$\mathbb{E}_Z[f(\mu+\sigma\, Z)] = \exp(-rT)\ \Phi\left(\frac{\mu-K}{\sigma}\right).$$

- for each W, the payoff is now smooth, differentiable
- derivative is $O(h^{-1/2})$ near strike, near zero elsewhere
  $\Longrightarrow$ variance is $O(h^{-1/2})$

Analytic evaluation of conditional expectation not possible in general for multivariate cases, so use Monte Carlo estimation!

Vibrato Monte Carlo

Main novelty comes in calculating the sensitivity.

For a particular W, we have a Normal probability distribution for $\widehat{S}_N$ and can apply the Likelihood Ratio method to get

$$\frac{\partial}{\partial\theta}\,\mathbb{E}_Z\left[f(\widehat{S}_N)\right] \;=\; \mathbb{E}_Z\left[f(\widehat{S}_N)\;\frac{\partial(\log p_S)}{\partial\theta}\right],$$

where

$$\begin{aligned}\frac{\partial(\log p_S)}{\partial\theta} &= \frac{\partial(\log p_S)}{\partial\mu}\,\frac{\partial\mu}{\partial\theta}+\frac{\partial(\log p_S)}{\partial\sigma}\,\frac{\partial\sigma}{\partial\theta}\\ &= \frac{Z}{\sigma}\,\frac{\partial\mu}{\partial\theta}+\frac{Z^2-1}{\sigma}\,\frac{\partial\sigma}{\partial\theta}.\end{aligned}$$

Averaging over all W then gives the expected sensitivity.

Vibrato Monte Carlo

To improve the variance, we note that

$$\mathbb{E}[1] = 1 \quad \Longrightarrow \quad \mathbb{E}_Z\left[\frac{\partial(\log p_S)}{\partial\theta}\right] = 0$$

and hence

$$\frac{\partial}{\partial\theta}\,\mathbb{E}_Z\left[f(\widehat{S}_N)\right] \;=\; \mathbb{E}_Z\left[\Big(f(\mu{+}\sigma Z) - f(\mu)\Big)\,\frac{\partial(\log p_S)}{\partial\theta}\right].$$

The quantity

$$\widehat{P} = \Big(f(\mu{+}\sigma Z) - f(\mu)\Big)\,\frac{\partial(\log p_S)}{\partial\theta}$$

has $O(1)$ variance when $f(S)$ is Lipschitz.

Vibrato Monte Carlo

In the multivariate extension,

$$\widehat{S}(W,Z) = \mu + C\,Z$$

where μ is the mean, $\Sigma = C\,C^T$ is the variance, and Z is a vector of uncorrelated Normals. The joint p.d.f. is

$$\log p_S = -\tfrac{1}{2}\log|\Sigma| - \tfrac{1}{2}(\widehat{S}-\mu)^T\Sigma^{-1}(\widehat{S}-\mu) - \tfrac{1}{2}d\,\log(2\pi).$$

$$\Longrightarrow \quad \frac{\partial \log p_S}{\partial \mu} = C^{-T} Z, \quad \frac{\partial \log p_S}{\partial \Sigma} = \tfrac{1}{2}\,C^{-T}\left(Z Z^T - I\right) C^{-1}$$

Can also handle options dependent on values at intermediate times by using Brownian interpolation between simulation times on either side.

Vibrato Monte Carlo

For each W, in forward mode we have

$$\alpha,\ \dot{\alpha} \quad \longrightarrow \quad \mu,\ \dot{\mu},\ \Sigma,\ \dot{\Sigma} \quad \longrightarrow \quad \text{payoff + sensitivity}$$

- first bit – pathwise sensitivity calculation
- second bit – Likelihood Ratio Method

For maximum efficiency can use adjoint/reverse mode

$$\begin{array}{ccccc} \alpha & \longrightarrow & \mu,\ \Sigma & \longrightarrow & \text{payoff} \\ \overline{\alpha} & \longleftarrow & \overline{\mu},\ \overline{\Sigma} & \longleftarrow & \text{sensitivity} \end{array}$$

$\overline{\mu},\ \overline{\Sigma}$ are coefficients multiplying $\dot{\mu},\ \dot{\Sigma}$ in forward mode

Vibrato Monte Carlo

Test case: Geometric Brownian motion

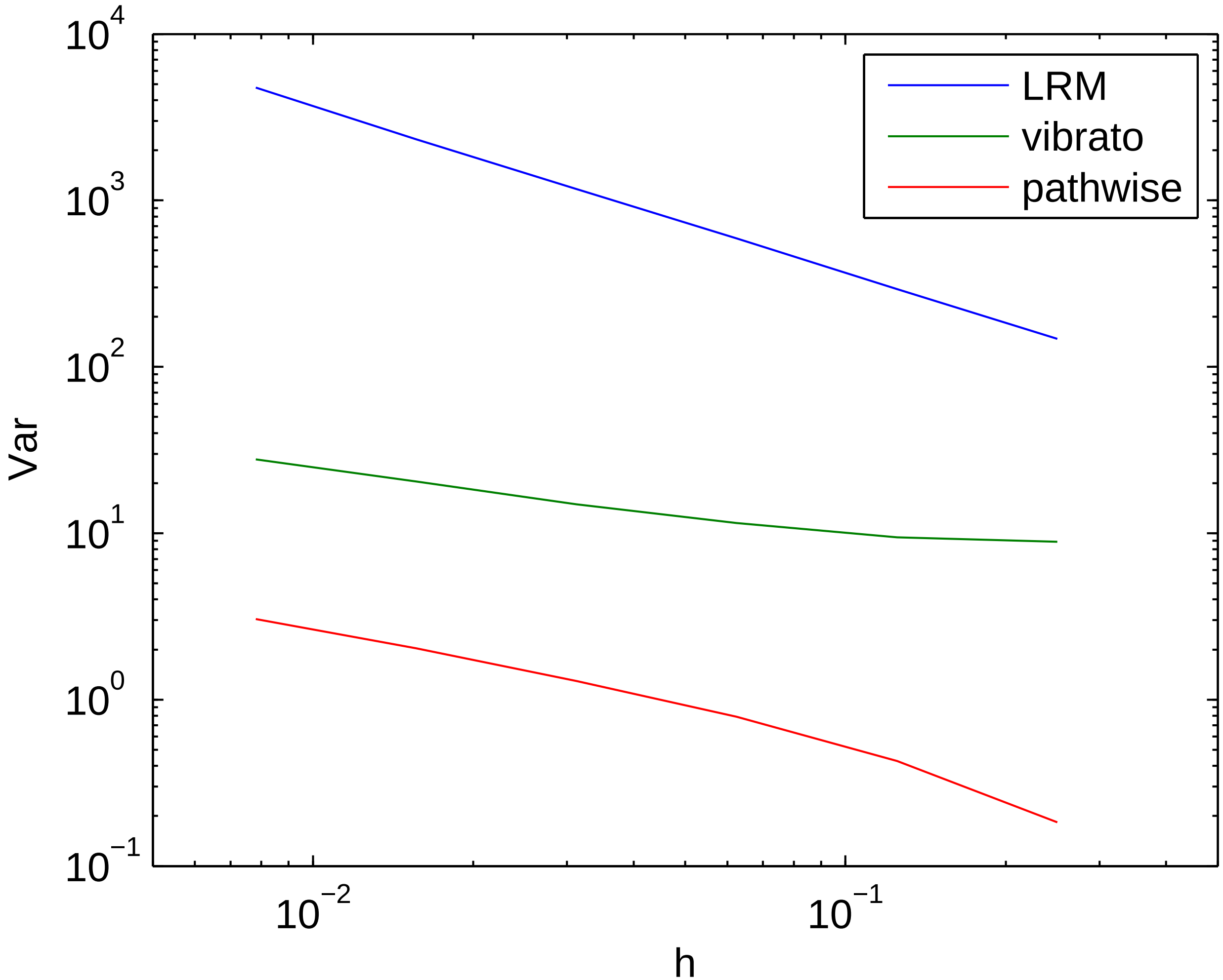
$$\begin{aligned} \mathrm{d}S_t^{(1)} &= r\, S_t^{(1)}\,\mathrm{d}t + \sigma^{(1)}\, S_t^{(1)}\,\mathrm{d}W_t^{(1)} \\ \mathrm{d}S_t^{(1)} &= r\, S_t^{(2)}\,\mathrm{d}t + \sigma^{(2)}\, S_t^{(2)}\,\mathrm{d}W_t^{(2)} \end{aligned}$$

with a simple digital call option based solely on $S_T^{(1)}$.

Parameters: $r=0.05,\ \sigma^{(1)}=0.2,\ \sigma^{(2)}=0.3,\ T=1$,
$S_0^{(1)}=S_0^{(2)}=100,\ K=100,\ \rho=0.5$

Numerical results compare LRM, vibrato with one Z per W, and pathwise using conditional expectation.

Vibrato Monte Carlo



More Z samples per path would bring the vibrato results closer to the pathwise results based on analytic expectation

Conclusions

- Adjoint implementation of pathwise sensitivities is very efficient when multiple Greeks are needed
- Automatic differentiation tools can aid software development
- Barrier and American options can be handled with care
- New "vibrato" MC idea can handle discontinuous payoffs and avoid the need to differentiate payoffs

Further information:

- `people.maths.ox.ac.uk/~gilesm/`
- Email: `mike.giles@maths.ox.ac.uk`