Computational Finance: Opportunities and challenges for AD

Mike Giles

mike.giles@maths.ox.ac.uk

Oxford University Mathematical Institute

Oxford-Man Institute for Quantitative Finance

Outline

- Monte Carlo methods
- PDE methods
- challenges with lack of differentiability
- assorted comments about the finance community

Computational Finance

- biggest growth area in scientific computing, roughly 20% of Top 500 “supercomputers”
- biggest employer of Oxford mathematicians and theoretical physicists, often as “quants”
  - traders – make/lose the money
  - quants – develop the models and codes
  - IT – organise execution on large distributed systems, and connect to data and external world
- two main kinds of computations
  - Monte Carlo (60% ?)
  - PDE (30% ?)
- sensitivities are very important, so a good application area for AD

SDEs in Finance

In computational finance, stochastic differential equations are used to model the behaviour of

- stocks
- interest rates
- exchange rates
- weather
- electricity/gas demand
- crude oil prices
- …

The stochastic term accounts for the uncertainty of unpredictable day-to-day events.

SDEs in Finance

These models are then used to calculate “fair” prices for a huge range of financial options:

- an option to sell a stock portfolio at a specific price in 2 years time
- an option to buy aviation fuel at a specific price in 6 months time
- an option to sell US dollars at a specific exchange rate in 3 years time

In most cases, the buyer of the financial option is trying to reduce their risk.

SDEs in Finance

Examples:

- Geometric Brownian motion (Black-Scholes model for stock prices)

$$\mathrm{d}S = r\,S\,\mathrm{d}t + \sigma\,S\,\mathrm{d}W$$

- Cox-Ingersoll-Ross model (interest rates)

$$\mathrm{d}r = \alpha(b - r)\,\mathrm{d}t + \sigma\,\sqrt{r}\,\mathrm{d}W$$

- Heston stochastic volatility model (stock prices)

$$\begin{aligned}\mathrm{d}S &= r\,S\,\mathrm{d}t + \sqrt{V}\,S\,\mathrm{d}W_1\\ \mathrm{d}V &= \lambda\,(\sigma^2 - V)\,\mathrm{d}t + \xi\,\sqrt{V}\,\mathrm{d}W_2\end{aligned}$$

with correlation ρ between $\mathrm{d}W_1$ and $\mathrm{d}W_2$

SDEs in Finance

Stochastic differential equation with general drift and volatility terms:

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

$W(t)$ is a Wiener variable with the properties that for any $q<r<s<t$, $W(t)-W(s)$ is Normally distributed with mean 0 and variance $t-s$, independent of $W(r)-W(q)$.

In many finance applications, we want to compute the expected value of an option 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.

Payoff functions

"Call" payoff:

$$f(S) = \max(0, S-K)$$

"Put" payoff:

$$f(S) = \max(0, K-S)$$

Digital call:

$$f(S) \;=\; H(S-K) \;=\; \begin{cases} 1, & S > K \\ 0, & \text{otherwise} \end{cases}$$

More complex payoffs have similar elements, but may also depend on the underlying value at a number of intermediate dates.

Monte Carlo Estimation

Euler discretisation with timestep h:

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

where ΔW_n are Normal with mean 0, variance h.

Simplest Monte Carlo estimator for expected payoff is an average of M independent path simulations:

$$M^{-1} \sum_{i=1}^{M} f(\widehat{S}^{(i)}_{T/h})$$

Key idea is very simple, but in practice it gets much more complicated through enhancements to reduce the variance of the estimator.

Monte Carlo Estimation

In the simplest case of Geometric Brownian motion

$$\mathrm{d}S(t) = r\,S\,\mathrm{d}t + \sigma\,S\,\mathrm{d}W(t),$$

the SDE can be solved analytically to give

$$S(T) = S_0 \exp\left(\left(r - \tfrac{1}{2}\sigma^2\right)T + \sigma\,W(T)\right).$$

In this case, we can avoid the Euler discretisation and directly sample $W(T)$ to get

$$V \equiv \mathbb{E}\left[f(S(T))\right] \approx M^{-1}\sum_{i=1}^{M} f(S^{(i)})$$

– will use to explain approaches to calculating sensitivities

Hedging

- Casinos make money consistently by having lots of customers on different games (statistically unlikely to all be lucky) and a large house margin (expected profit)
- Banks don’t have as many customers/markets, and the margins are smaller, so to get consistent profits they use hedging
- Basic idea: hold a portfolio of options so that the value of the portfolio isn’t affected (to leading order) by changes in stock prices, interest rates, etc.
- This risk management is very important and affects the financial reserves which the banks are forced to maintain

Hedging

Suppose a portfolio has amount c_j of option with payoff f_j:

$$P = \sum_j c_j \, f_j$$

What is the expected sensitivity to changes in interest rate?

$$\frac{\partial}{\partial r} \, \mathbb{E}[P] = \sum_j c_j \, \frac{\partial}{\partial r} \, \mathbb{E}[f_j]$$

By choosing the c_j so this is zero, the bank eliminates its risk to unexpected interest rate changes.

To eliminate its risk to other unpredictable changes the bank needs first (and sometimes second) derivatives of the expected value with respect to various parameters

MC sensitivities

The sensitivities are known collectively as “the Greeks”.

Delta: first derivative w.r.t. stock price $\Delta = \dfrac{\partial V}{\partial S_0}$

Vega: first derivative w.r.t. volatility $\dfrac{\partial V}{\partial \sigma}$

Gamma: second derivative w.r.t. stock price $\Gamma = \dfrac{\partial^2 V}{\partial S_0^2}$

– usually don’t need as much accuracy for second derivatives

MC sensitivities

Simplest approach is to use a finite difference approximation,

$$\frac{\partial V}{\partial \theta} \approx \frac{V(\theta+\Delta\theta) - V(\theta-\Delta\theta)}{2\,\Delta\theta},$$

$$\frac{\partial^2 V}{\partial \theta^2} \approx \frac{V(\theta+\Delta\theta) - 2\,V(\theta) + V(\theta-\Delta\theta)}{(\Delta\theta)^2}.$$

– very simple, but expensive and inaccurate if $\Delta\theta$ is too big or too small

However, this is what is currently used by many (most?) banks because of its simplicity!

MC sensitivities

For simple cases where we know the terminal probability distribution and so

$$V \equiv \mathbb{E}\left[f(S(\theta;T))\right] = \int f(S)\, p_S(\theta;S)\, \mathrm{d}S,$$

we can differentiate this to get

$$\frac{\partial V}{\partial \theta} = \int f\, \frac{\partial p_S}{\partial \theta}\, \mathrm{d}S = \int f\, \frac{\partial(\log p_S)}{\partial \theta}\, p_S\, \mathrm{d}S = \mathbb{E}\left[f\, \frac{\partial(\log p_S)}{\partial \theta}\right].$$

This is the Likelihood Ratio Method – can handle non-differentiable payoffs, but doesn't generalise well to cases where we have to simulate the whole path

MC sensitivities

Alternatively, for simple Geometric Brownian Motion

$$V \equiv \mathbb{E}\left[f(S(\theta;T))\right] = \int f(S(\theta;T))\, p_W(W)\, \mathrm{d}W,$$

and differentiating this gives

$$\frac{\partial V}{\partial \theta} = \int \frac{\partial f}{\partial S}\, \frac{\partial S(T)}{\partial \theta}\, p_W\, \mathrm{d}W = \mathbb{E}\left[\frac{\partial f}{\partial S}\, \frac{\partial S(T)}{\partial \theta}\right],$$

with $\partial S(T)/\partial\theta$ being evaluated at fixed W.

This is the pathwise sensitivity approach – by considering the limit of a sequence of regularised payoffs, it can be proved that this is also OK for payoffs which are continuous and piecewise differentiable

MC sensitivities

It also generalises well to cases where we have to simulate the whole path.

Differentiating the Euler path discretisation, holding fixed the Brownian increments, we get

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

leading to

$$\frac{\partial \widehat{V}}{\partial \theta} = M^{-1} \sum_m \frac{\partial f}{\partial S}(\widehat{S}_N^{(m)})\ \frac{\partial \widehat{S}_N^{(m)}}{\partial \theta}.$$

MC sensitivities

This corresponds to standard forward mode AD, with the random number generation held fixed.

The reverse mode is obviously more efficient when one needs multiple sensitivities.

I became known in computational finance by doing this, introducing adjoints in *"Smoking Adjoints"* paper with Paul Glasserman in *Risk* magazine.

MC sensitivities

- Several (many?) major banks have adopted these ideas
- One has licensed Tapenade
  (C $\overset{\text{manual}}{\longrightarrow}$ Fortran $\rightarrow$ Fortran $\overset{\text{manual}}{\longrightarrow}$ C !!!)
- As far as I know, the others have done it by hand
- C and C++ are the dominant languages, but a restricted subset is maybe sufficient in practice
- Operator overloading is great for verifying/debugging hand-written code; and efficiency/ease-of-use tradeoff may be attractive
- MATLAB is also used increasingly for prototyping – a reverse mode source transformation AD tool would be very useful

MC sensitivities

Monte Carlo “engines” have two parts:

- SDE path simulation
- payoff evaluation

These are handled quite differently:

- SDE path models change infrequently, so written by quants in C/C++ for execution speed
  - want source transformation AD here (plus manual optimisation?) for optimal performance?
- payoffs change very frequently, so defined by traders using flexible, inefficient, proprietary scripting languages
  - forward mode operator overloading may be good here, if payoff is differentiable

MC sensitivities

AD can be used without problems for the path simulation – everything is usually twice differentiable

Payoff evaluation can be a problem because of scripting language and discontinuous payoffs.

Solutions:

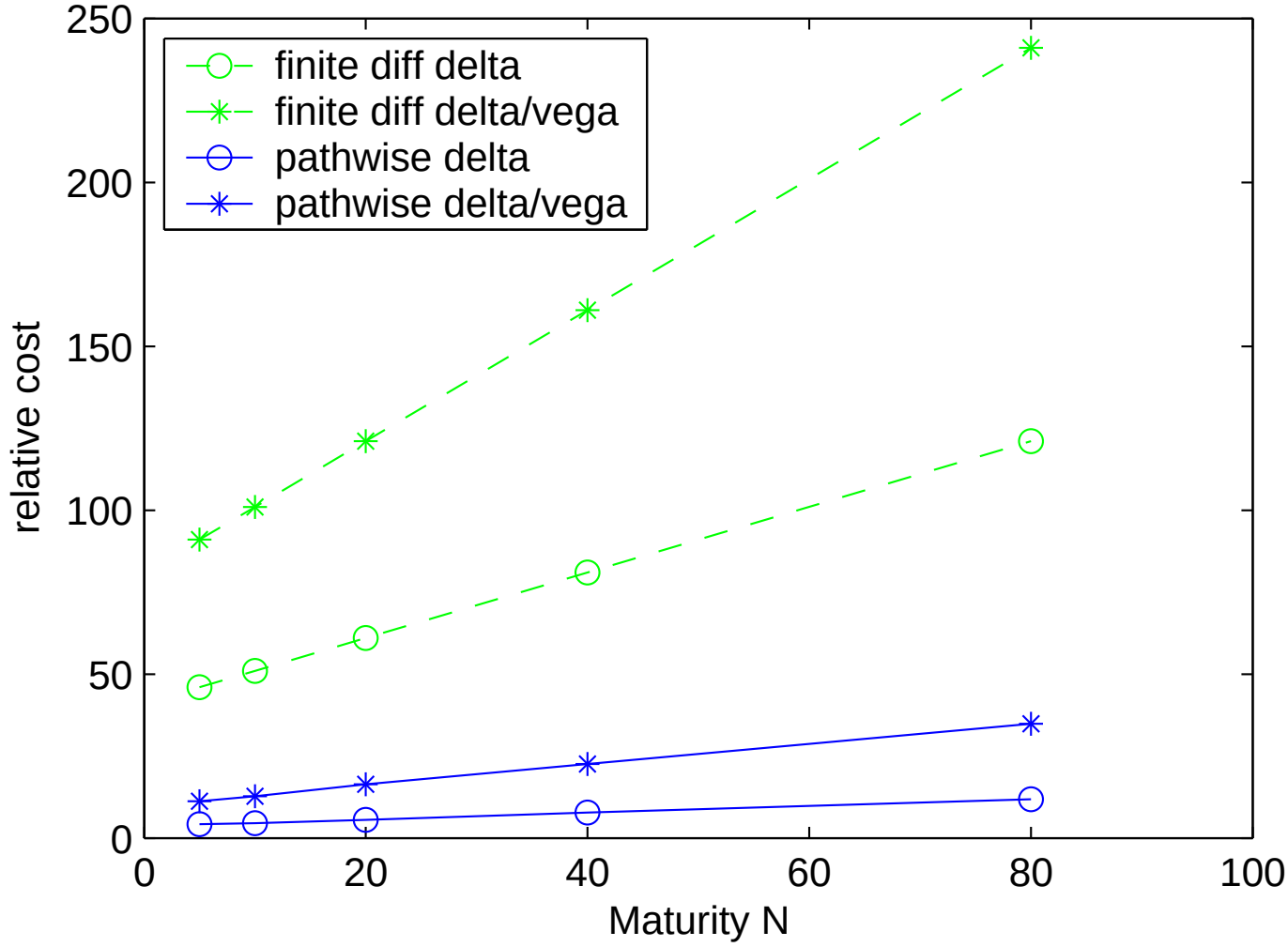
- use finite differences to estimate derivative $\partial f / \partial S$ but variance $= O(\Delta s^{-1})$ for digitals
- use a hybrid pathwise / LRM method
  - Malliavin calculus (but no efficient adjoint?)
  - my new vibrato technique: var $= O(h^{-1/2})$ for digitals
- all of these require only payoff values, not derivatives

LIBOR Test Application

- BGM model for behaviour of London InterBank Overnight interest Rates
- used to price various options which depend on current and future interest rates
- test problem performs N timesteps with a vector of $N+40$ forward rates, and computes the sensitivity of a portfolio of swaptions to changes in both initial forward rates and volatilities
- original code with hand-written sensitivities in C, extended to C++ to use FADBAD++ (Ole Stauning); results also for TAC++ (Michael Vossbeck) and ADOL-C (Andreas Griewank)
- involves quite a few exponentials, which are not repeated in hand-coded forward/reverse code

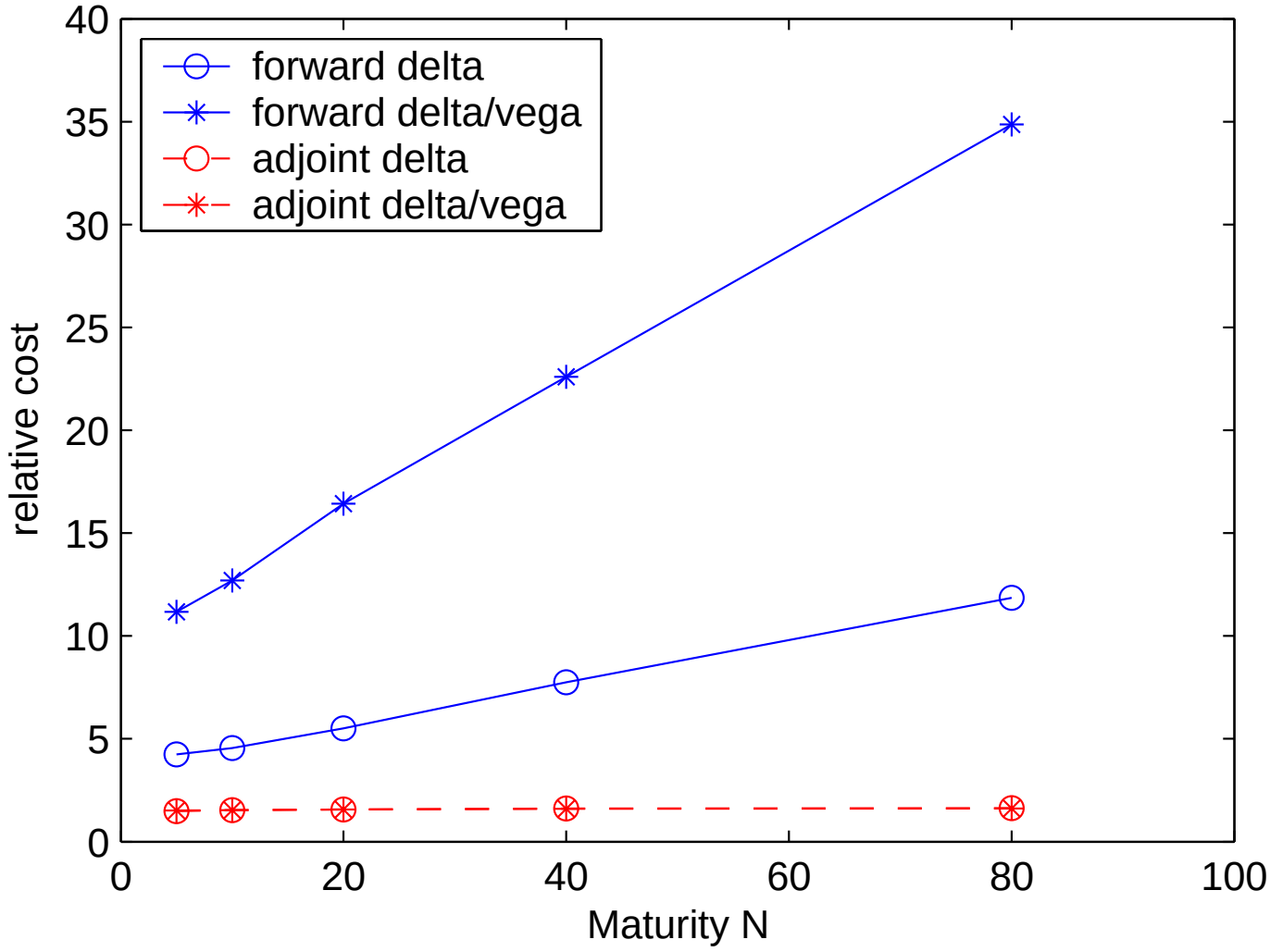
LIBOR Test Application

Finite differences versus forward pathwise sensitivities:



LIBOR Test Application

Hand-coded forward versus adjoint pathwise sensitivities:



LIBOR Test Application

milliseconds/path for 80 sensitivities	Gnu g++	Intel `icc`
original	0.37	0.10
hand-coded forward	0.97	0.52
hand-coded reverse	0.47	0.19
FADBAD++ forward	4.30	5.00
FADBAD++ reverse	6.20	4.86
hybrid (hand/FADBAD++) forward	1.02	0.63
hybrid (hand/FADBAD++) reverse	0.65	0.35
TAC++ forward	1.36	0.85
TAC++ reverse	1.28	0.45
ADOL-C reverse (scaled)	1.60	

MC final comments

- Monte Carlo is the main simulation tool in computational finance
- Application of AD to SDE path simulation is fairly straightforward
- For reverse mode, checkpoint every timestep – storage requirements are minimal
- In hand-written code, I also store the result of expensive instructions – the rest I recompute
- Challenges are in payoff evaluation:
  - use of scripting languages
  - lack of differentiability
  - solutions to these problems exist – see references

Financial PDEs

It can be shown that, given an SDE model for the underlying stock, interest rate, exchange rate, etc., the value of a financial option satisfies a backward-time convection-diffusion PDE.

For example, simple Geometric Brownian Motion leads to the Black-Scholes equation for $V(S,t)$

$$V_t + r\,S\,V_S + \tfrac{1}{2}\sigma^2 S^2 V_{SS} = r\,V$$

to be solved backwards in time from the known payoff value.

Each separate stock price, interest rate, etc, introduces an extra dimension in the PDE. An option based on 20 stock prices gives a 20-dimensional PDE!

Financial PDEs

In practice, PDEs are only used for dimensions 1-3; beyond that Monte Carlo methods are used.

Numerical approximations usually use:

- simple structured grids (no FE methods)
- implicit time-marching (Crank-Nicolson plus Rannacher startup)
- either ADI or preconditioned iterative solution of implicit equations at each timestep

Also, simple options lead to linear PDEs, but there are other more complex options with optional early exercise features (American, Bermudan options) which lead to nonlinear PDEs

PDE sensitivities

Sensitivities are again needed for hedging.

Application of AD is straightforward, in principle, but can require care to avoid loss of accuracy

$O(\Delta S^m)$ accuracy for option value does not imply $O(\Delta S^m)$ accuracy for sensitivities

Might only get $O(\Delta S^{m-1})$ accuracy for first derivative, and $O(\Delta S^{m-2})$ accuracy for second derivative

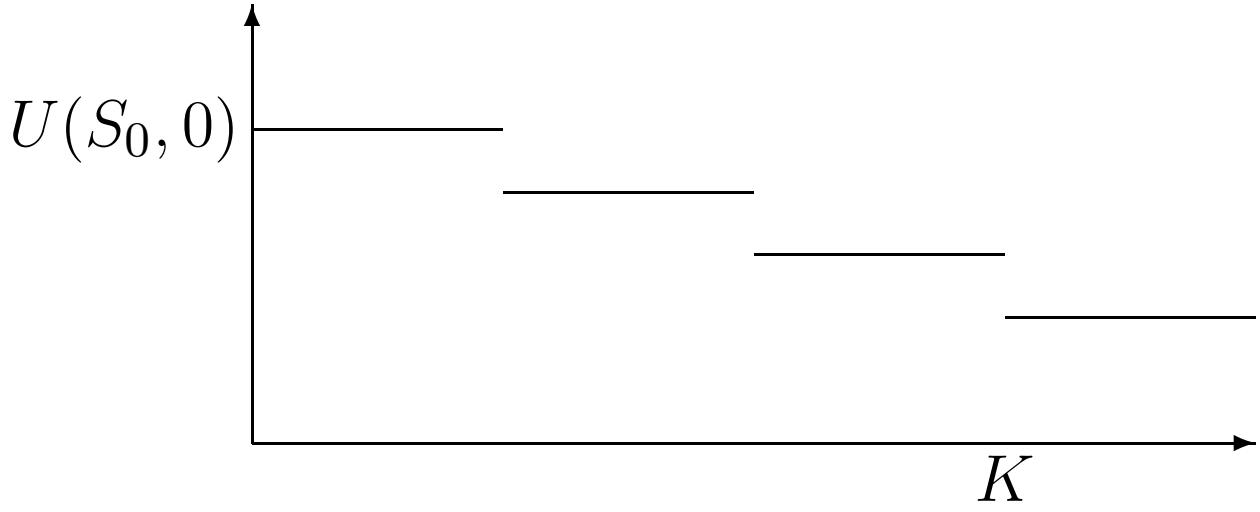
This could be a problem since usually $m \leq 2$.

PDE sensitivities

An extreme example: Black-Scholes PDE with discontinuous digital call payoff:

$$V(S,T) = H(S-K)$$

where K is the strike. If we use a fixed S grid and take the numerical payoff pointwise, then the accuracy in $V(S,0)$ is $O(\Delta S)$, but AD will give $\partial V/\partial K = 0$.

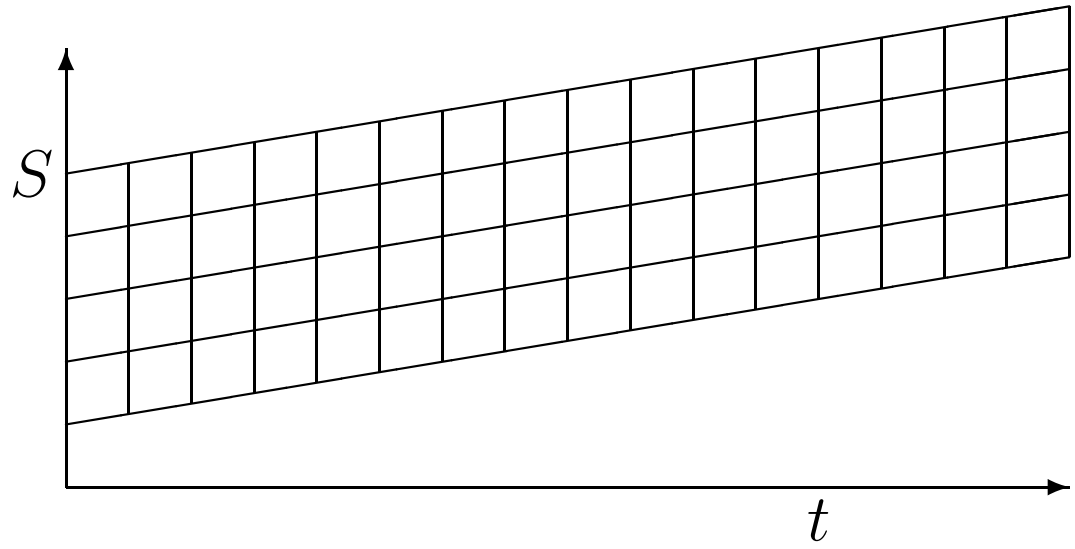


PDE sensitivities

How do we fix this?

Use a grid which moves with K: $S_j^n = S_j^0 + \frac{t^n}{T}(K - K_0)$

and then differentiate w.r.t. K



This can significantly increase the programming complexity, but is not needed for all derivatives.

PDE sensitivities

Adjoints (reverse mode AD) are again very useful when large numbers of sensitivities are needed

- multiple future interest rates
- "knot" values in (bi-)cubic spline function defining local volatility function $\sigma(S,t)$

Adjoints for financial PDE applications were introduced, I think, by Achdou & Pironneau (SIAM, 2005), carrying over ideas from CFD

PDE sensitivities

Alex Prideaux's PhD research:

- investigation of differentiability problems
- high-level NLA adjoints for ADI and other implicit time-marching schemes
- adjoints for integro-PDE problems (jump-diffusion models) with implementations involving FFTs
- modular AD for applications involving several stages:

  σ knot values $\rightarrow$ spline and evaluate at grid points $\rightarrow$ PDE solution
- second derivatives and use of AD tools

Conclusions

- computational finance is a rapidly growing field, just waking up to the benefits of adjoints and AD
- language needs are C/C++ and MATLAB, but a restricted subset of each is sufficient
- there are important challenges due to lack of differentiability, but solutions exist
- for Monte Carlo applications, apply AD to path simulation, and use other techniques for payoff evaluation
- for PDE simulations, take care to avoid loss of one order of accuracy with each derivative

Further Reading

P. Glasserman, *Monte Carlo Methods in Financial Engineering*, Springer, 2004.

M.B. Giles & P. Glasserman, “Smoking Adjoints: fast Monte Carlo Greeks”, *Risk*, January 2006.

M.B. Giles, “Monte Carlo evaluation of sensitivities in computational finance”, HERCMA conference, 2007.

Y. Achdou & O. Pironneau, *Computational Methods for Option Pricing*, SIAM, 2005

A. Prideaux, PhD thesis on adjoints with financial PDEs (due end of 2008)

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