Adjoint methods in computational finance

Mike Giles

Mathematical and Computational Finance Group,
Mathematical Institute, University of Oxford

Oxford-Man Institute of Quantitative Finance

12th Winter School on Mathematical Finance

Jan 21-23, 2013

Lecture outline

- PDEs and finite difference methods:
  - formulation of adjoint PDEs and finite difference methods
  - financial application
  - vanilla pricing calculation
  - sensitivities for linear explicit discretisations
  - nonlinear implicit discretisations
  - what can go wrong?
  - calibration using Fokker-Planck discretisation
  - Greeks using Black-Scholes discretisation
  - local volatility example

Forward and reverse PDEs

Suppose we are interested in the forward PDE

$$\frac{\partial p}{\partial t} = L_t \, p,$$

where L_t is a spatial operator, subject to Dirac initial data $p(x,0) = \delta(x - x_0)$, and we want the value of the output functional

$$(p(\cdot, T), \, f) \equiv \int p(x, T) \, f(x) \, \mathrm{d}x.$$

The adjoint spatial operator L_t^* is defined by the identity

$$(L_t v, w) = (v, L_t^* w), \quad \forall v, w$$

assuming certain homogeneous b.c.'s.

Forward and reverse PDEs

If $u(x,t)$ is the solution of the adjoint PDE

$$\frac{\partial u}{\partial t} = -L_t^* \, u,$$

subject to "initial" data $u(x,T) = f(x)$ then

$$\begin{aligned}
(p(\cdot,T),\, u(\cdot,T)) - (p(\cdot,0),\, u(\cdot,0)) &= \int_0^T \frac{\partial}{\partial t}(p,\, u)\ \mathrm{d}t \\
&= \int_0^T \left(\frac{\partial p}{\partial t},\, u\right) + \left(p,\, \frac{\partial u}{\partial t}\right)\ \mathrm{d}t \\
&= \int_0^T (L_t p,\, u) - (p,\, L_t^* u)\ \mathrm{d}t \\
&= 0,
\end{aligned}$$

and hence $u(x_0, 0) = (p(\cdot,T), f)$.

Forward and reverse PDEs

Hence, to compute our output of interest, we have a choice:

- forward:
  - start with Dirac initial data for $p(x, 0)$
  - solve forward PDE for $p(x, t)$
  - compute $(p(\cdot, T), f)$
- reverse:
  - start with "initial" data for $u(x, T)$
  - solve backward PDE for $u(x, t)$
  - output is $u(x_0, 0)$

We get the same answer either way, so can choose based on other considerations, such as computational efficiency

Financial relevance

Fokker-Planck (or forward Kolmogorov) equation:

$$\frac{\partial p}{\partial t} + \frac{\partial}{\partial x}\left(a\, p\right) = \frac{1}{2}\,\frac{\partial^2}{\partial x^2}\left(b^2\, p\right)$$

for probability density $p(x,t)$ for path S_t satisfying the SDE

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

Backward Kolmogorov (or Feynman-Kac) equation:

$$\frac{\partial u}{\partial t} + a\,\frac{\partial u}{\partial x} + \frac{1}{2}\,b^2\,\frac{\partial^2 u}{\partial x^2} = 0$$

where $u(x,t) = \mathbb{E}[f(S_T)|S_t = x]$

Financial relevance

The spatial operators are

$$L\,p \equiv -\frac{\partial}{\partial x}\left(a\,p\right) + \frac{1}{2}\,\frac{\partial^2}{\partial x^2}\left(b^2\,p\right)$$

and

$$L^*\,u \equiv a\,\frac{\partial u}{\partial x} + \frac{1}{2}\,b^2\,\frac{\partial^2 u}{\partial x^2}$$

The identity

$$(Lv, w) = (v, L^* w), \quad \forall v, w$$

can be verified by integration by parts, assuming

$a\,v\,w, \quad b^2 v\,\dfrac{\partial w}{\partial x}, \quad b^2 \dfrac{\partial v}{\partial x}\,w$ are zero on boundary.

Forward and reverse FDEs

Suppose that a numerical finite difference discretisation of the forward problem gives the discrete equivalent

$$p_{n+1} = A_n \, p_n$$

where p_n is an vector of approximations to $p(x_j, t_n)$ at points x_j at time t_n, and A_n is a square matrix.

For example,

$$p_{j,n+1} = p_{j,n} + \frac{\Delta t}{\Delta x^2} \left(p_{j+1,n} - 2p_{j,n} + p_{j-1,n} \right)$$

is an approximation to

$$\frac{\partial p}{\partial t} = \frac{\partial^2 p}{\partial x^2}$$

Forward and reverse FDEs

If there are N timesteps, the output $(p(x,T),\ f)$ can be approximated as

$$\sum_j p_{j,N}\ f_j\ \Delta x$$

or more generally as $f^T M\, p_N$ where M is a symmetric "mass" matrix, usually either diagonal or tri-diagonal.

The output then has the form

$$f^T M\, p_N = f^T\ M\, A_{N-1}\, A_{N-2}\, \ldots\, A_0\, p_0.$$

Forward and reverse FDEs

Taking the transpose, this can be re-expressed as

$$p_0^T \, v_0$$

where

$$v_0 = A_0^T \ \ldots \ A_{N-2}^T \, A_{N-1}^T \ M \ f$$

The adjoint solution v_n is therefore defined by

$$v_n = A_n^T \, v_{n+1}$$

subject to "initial" data $v_N = M \, f$.

Forward and reverse FDEs

It is often more appropriate to work with

$$u_n = M^{-1} v_n,$$

in which case we have

$$u_n = (M\, A_n^T\, M^{-1})\, u_{n+1}$$

subject to "initial" data

$$u_N = f,$$

and the output functional is $p_0^T\, M\, u_0$.

This is more appropriate because now u_n is an approximation to the adjoint PDE solution $u(x, t_n)$

Financial relevance

In finance, the discrete equations are usually formulated for backward equation:

$$u_n = B_n\, u_{n+1}$$

subject to payoff data $u_N = f$, and the output is $e^T u_0$ where e is a unit vector with a single non-zero entry.

The equivalent discrete adjoint problem is

$$P_{n+1} = B_n^T P_n$$

subject to initial data $P_0 = e$, and the output is $P_N^T f$.

When there is no discounting (so no $r\,u$ term in Black-Scholes PDE) then P_n corresponds to a vector of discrete probabilities – need to divide by grid spacing to get approximation to probability density.

Financial relevance

With implicit time-marching, we have an equation like

$$A_n \, u_n = C_n \, u_{n+1}$$

so

$$B_n \equiv A_n^{-1} C_n$$

In this case,

$$B_n^T \equiv C_n^T (A_n^T)^{-1}$$

so

$$P_{n+1} = C_n^T (A_n^T)^{-1} P_n$$

Note order reversal: multiplication by C_n and then by A_n^{-1} turns into multiplication by $(A_n^T)^{-1}$ and then by C_n^T

Financial relevance

Which is better – forward or reverse?

- reverse is only possibility for American options, and also gives Delta and Gamma approximations for free
- forward is best for pricing multiple European options
  - for different strikes, a single forward calculation and then a separate vector dot product for each option
  - for different maturities, do a single calculation to the final maturity, and use intermediate values at intermediate maturities
  - particularly useful when calibrating a model to vanilla options?

FDE sensitivities

Suppose we want to compute output $e^T u_0$ where $u_N = f$ and

$$u_n = B_n \, u_{n+1}.$$

Now suppose that f and B_n depend on some parameter θ, and we want to compute the sensitivity to θ.

Standard "forward mode" sensitivity analysis gives sensitivity $e^T \dot{u}_0$ where $\dot{u}_N = \dot{f}$ and

$$\dot{u}_n = B_n \, \dot{u}_{n+1} + \dot{b}_n$$

with

$$\dot{b}_n \equiv \dot{B}_n \, u_{n+1}$$

FDE sensitivities

What is reverse mode adjoint?

Work "backwards" applying the linear algebra rules.

$$\overline{u}_0 = e$$

$$\overline{u}_{n+1} = B_n^T \, \overline{u}_n, \quad \overline{b}_n = \overline{u}_n$$

$$\overline{f} = \overline{u}_N$$

Note: the original code goes from $n\!=\!N$ to $n\!=0$, so the reverse mode goes from $n\!=\!0$ to $n\!=\!N$, using stored values for u_{n+1}.

FDE sensitivities

This gives $\overline{f}$ and $\overline{b}_n$ and then payoff sensitivity is given by

$$\overline{\theta} = \overline{f}^{\,T} \dot{f} + \sum_n \overline{b}_n^{\,T} \dot{b}_n$$

This can be evaluated using AD software, or hand-coded following the AD algorithm.

$\theta, u_{n+1} \longrightarrow B_n\, u_{n+1}$	original code
$\theta, u_{n+1} \longrightarrow \dot{B}_n\, u_{n+1}$	forward mode, keeping u_{n+1} fixed
$\theta, u_{n+1}, \overline{b}_n \longrightarrow \overline{\theta}$ incr	reverse mode, keeping u_{n+1} fixed

FDE sensitivities

We now consider nonlinear discretisations (e.g. for American options)

In 1D, these are usually one of the following types:

- explicit:

$$u_{j,n} = g(u_{j-1,n+1}, u_{j,n+1}, u_{j+1,n+1})$$

  – function of the nearest "old" values from the previous timestep

- one-step implicit:

$$a_j\, u_{j-1,n} + b_j\, u_{j,n} + c_j\, u_{j+1,n} = g(u_{j-1,n+1}, u_{j,n+1}, u_{j+1,n+1})$$

  – needs solution of tridiagonal system of equations at each timestep

- iterative implicit:

$$g(u_{j-1,n}, u_{j,n}, u_{j+1,n},\;\; u_{j-1,n+1}, u_{j,n+1}, u_{j+1,n+1}) = 0$$

  – a nonlinear system of simultaneous equations to be solved iteratively

FDE sensitivities

Considering perturbations to these, "forward mode" sensitivity analysis gives

$$A\,\dot{u}_n = C_n\,\dot{u}_{n+1} + \dot{b}_n$$

with tridiagonal A, C and vector $\dot{b}_n$.

For example, in the third case we have $\dot{b}_{j,n} \equiv \dfrac{\partial g}{\partial \theta}$ and

$$A_{j,j-1} \equiv -\,\frac{\partial g}{\partial u_{j-1,n}}, \quad A_{j,j} \equiv -\,\frac{\partial g}{\partial u_{j,n}}, \quad A_{j,j+1} \equiv -\,\frac{\partial g}{\partial u_{j+1,n}},$$

$$C_{j,j-1} \equiv \frac{\partial g}{\partial u_{j-1,n}}, \quad C_{j,j} \equiv \frac{\partial g}{\partial u_{j,n}}, \quad C_{j,j+1} \equiv \frac{\partial g}{\partial u_{j+1,n}},$$

with $A, C, \dot{b}_n$ dependent on $u_{j-1,n}, u_{j,n}, u_{j+1,n},\; u_{j-1,n+1}, u_{j,n+1}, u_{j+1,n+1}$.

FDE sensitivities

"Reverse mode" gives

$$\overline{u}_{n+1} = C_n^T \, (A_n^T)^{-1} \overline{u}_n, \quad \overline{b}_n = (A_n^T)^{-1} \, \overline{u}_n$$

This again gives $\overline{b}_n$ and AD ideas can then be used to compute the increments to $\overline{\theta}$.

So far, I have talked of θ being a single input parameter, but it can be a vector of input parameters.

The key is that they all use the same $\overline{f}$ and $\overline{b}_n$, and it is just this final AD step which depends on θ, and the cost is independent of the number of parameters.

What can go wrong?

Differentiation like this gives the sensitivity of the numerical approximation to changes in the input parameters.

This is not necessarily a good approximation to the true sensitivity

Simplest example: a digital put option with strike K when wanting to compute $\dfrac{\partial V}{\partial K}$, the sensitivity of the option price to the strike

What can go wrong?

Using the simplest numerical approximation,

$$f_j = H(K - S_j)$$

and so $\dot{f} = 0$ which leads to a zero sensitivity!

Using a better approximation

$$f_j = \frac{1}{\Delta S} \int_{S_j - \frac{1}{2}\Delta S}^{S_j + \frac{1}{2}\Delta S} H(K - S)\, \mathrm{d}S$$

gives an $O(\Delta S^2)$ approximation to the price, and an $O(\Delta S)$ approximation to the sensitivity to K.

What can go wrong?

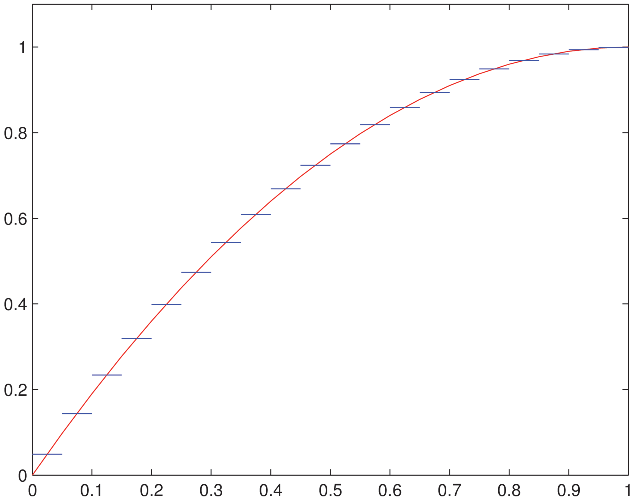


Figure: A stepped approximation to the function $2x - x^2$

What can go wrong?

More generally, discontinuities are not the only problem.

Suppose our analytic problem with input x has solution

$$u = x^2$$

and our discrete approximation with step size $h \ll 1$ is

$$u_h = x^2 + h^2 \sin(x/h)$$

then $u_h - u = O(h^2)$ but $u_h' - u' = O(h)$

This seems to be typical, that in bad cases you lose one order of convergence each time you differentiate.

What can go wrong?

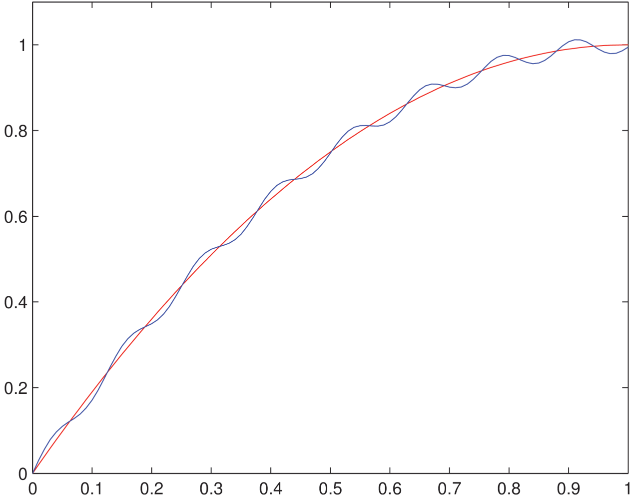


Figure: A wavy approximation to the function $2x - x^2$

What can go wrong?

Careful construction of the approximation can usually avoid these problems.

In the digital put case, the problem was the strike moving across the grid.

Solution: move the grid with the strike at maturity $t = T$, keeping the end at time $t = 0$ fixed.

$$\log S_j(t) = \log S_j^{(0)} + (\log K - \log K^{(0)}) \frac{t}{T}$$

This uses a baseline grid $S_j^{(0)}$ corresponding to the true strike $K^{(0)}$ then considers perturbations to this which move with the strike.

Use of adjoint sensitivities

Fokker-Planck discretisation:

- standard calculation goes forward in time, then performs a separate vector dot product for each vanilla European option
- adjoint sensitivity calculation goes backward in time, gives sensitivity of vanilla prices to initial prices, model constants
- if the Greeks are needed for each option, then a separate adjoint calculation is needed for each – might be better to use "forward mode" AD instead, depending on number of parameters and options
- one adjoint calculation can give a weighted average of Greeks – useful for calibrating a model to market data

Use of adjoint sensitivities

A calibration procedure might find the optimum vector of parameters θ which minimises the mean square difference between vanilla option model prices and market prices:

$$\frac{1}{2} \sum_k \left(C^{(k)}_{model}(\theta) - C^{(k)}_{market} \right)^2$$

Gradient-based optimisation would need to compute

$$\sum_k \left(C^{(k)}_{model} - C^{(k)}_{market} \right) \frac{\partial C^{(k)}_{model}}{\partial \theta}$$

which is just a weighted average (with both positive and negative weights) of the Greeks.

Use of adjoint sensitivities

Since the vanilla option price is of the form

$$C^{(k)}_{model} = f_k^T P_N$$

then, provided f_k does not depend on θ, the adjoint calculation works backwards in time from the "initial" condition:

$$\overline{P}_N = \sum_k \left(C^{(k)}_{model} - C^{(k)}_{market)} \right) f_k$$

Use of adjoint sensitivities

Black-Scholes / backward Kolmogorov discretisation:

- standard calculation goes backward in time for pricing an exotic option, with possible path-dependency and optional exercise
- adjoint sensitivity calculation goes forward in time, giving sensitivity of price to initial prices, model constants, etc.

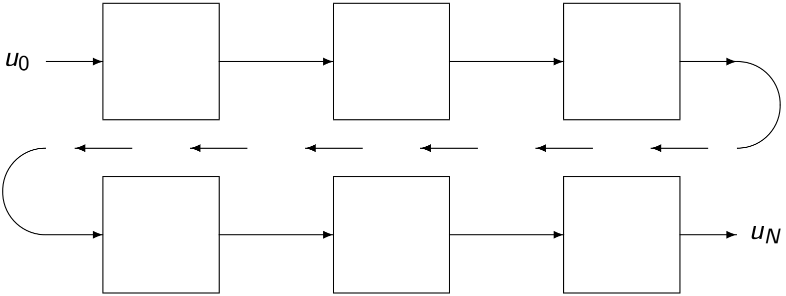
Use of adjoint sensitivities

Many applications may involve a process which goes through several stages:

- market implied vol $\sigma_I \implies$ local vol σ_L at a few points using Dupire's formula
- local vol σ_L at a few points $\implies \sigma_L, \sigma'_L$ through cubic spline construction
- $\sigma_L, \sigma'_L \implies \sigma$ at FD grid points using cubic spline interpolation
- σ at FD grid points $\implies$ option value V using FD calculation

Generic black-box problem

Remember generic black-box viewpoint



Key assumption: each step is (locally) differentiable

Generic black-box problem

Forward mode:

$$\dot{u}_{n+1} = D_n \; \dot{u}_n, \qquad D_n \equiv \frac{\partial u_{n+1}}{\partial u_n}$$

Reverse mode:

$$\overline{u}_n = D_n^T \; \overline{u}_{n+1}$$

starting from given $\overline{u}_N$, and with all of the D_n or u_n stored from the original black-box computation.

Validation:

$$\frac{\partial u_N}{\partial u_n} \, \frac{\partial u_n}{\partial \theta} = \frac{\partial u_N}{\partial u_{n+1}} \, \frac{\partial u_{n+1}}{\partial \theta} \quad \Longrightarrow \quad \overline{u}_n^T \; \dot{u}_n = \overline{u}_{n+1}^T \; \dot{u}_{n+1}$$

This must hold for any $\dot{u}_n, \overline{u}_{n+1}$ – very helpful for checking the forward and reverse mode versions of each black-box component.

Use of adjoint sensitivities

To obtain the sensitivity of the option value to changes in the market implied vol, go through all of the stages in the reverse order:

- $\overline{V} \implies \overline{\sigma}$
- $\overline{\sigma} \implies \overline{\sigma_L}, \overline{\sigma'_L}$
- $\overline{\sigma_L}, \overline{\sigma'_L} \implies \overline{\sigma_L}$
- $\overline{\sigma_L} \implies \overline{\sigma_I}$

Each stage needs to be developed and validated separately, then they all fit together in a modular way.

Use of adjoint sensitivities

It is not necessary to use adjoint techniques at each stage.

For example, the final stage in the last example computes

$$\overline{\sigma_I} = \left(\frac{\partial \sigma_L}{\partial \sigma_I}\right)^T \overline{\sigma_L}$$

The matrix

$$\frac{\partial \sigma_L}{\partial \sigma_I}$$

can be obtained by forward mode sensitivity analysis (more expensive), or approximated by bumping (more expensive and less accurate)

Cubic spline step

For a point $S_j < S < S_{j+1}$, cubic spline interpolation is defined by an equation of the form

$$\sigma(S) = a_j(S)\ \sigma_j + b_j(S)\ \sigma_{j+1} + c_j(S)\ \sigma'_j + d_j(S)\ \sigma'_{j+1},$$

where $a_j(S), b_j(S), c_j(S), d_j(S)$ are cubic polynomials.

The σ' values are obtained from the σ values by solving a tri-diagonal system of equations:

$$A\ \sigma' = B\ \sigma$$

Cubic spline step

In the forward mode we get

$$A\,\dot{\sigma}' = B\,\dot{\sigma},$$

and then

$$\dot{\sigma}(S) = a_j(S)\,\dot{\sigma}_j + b_j(S)\,\dot{\sigma}_{j+1} + c_j(S)\,\dot{\sigma}'_j + d_j(S)\,\dot{\sigma}'_{j+1}$$

assuming that the point at which the spline is evaluated does not change.

As usual, this is relatively intuitive.

Cubic spline step

In the reverse mode we have

$$\begin{aligned}
\overline{\sigma}_j &\mathrel{+}= a_j(S)\,\overline{\sigma}(S) \\
\overline{\sigma}_{j+1} &\mathrel{+}= b_j(S)\,\overline{\sigma}(S) \\
\overline{\sigma'}_j &\mathrel{+}= c_j(S)\,\overline{\sigma}(S) \\
\overline{\sigma'}_{j+1} &\mathrel{+}= d_j(S)\,\overline{\sigma}(S)
\end{aligned}$$

which gives the increments to $\overline{\sigma}_j, \overline{\sigma}_{j+1}, \overline{\sigma'}_j, \overline{\sigma'}_{j+1}$ due to the spline evaluation.

Reversing the calculation of the spline derivatives then gives

$$\overline{\sigma} \mathrel{+}= B^T (A^T)^{-1}\, \overline{\sigma'},$$

which adds to $\overline{\sigma}$ the extra dependence due to the way in which σ' is calculated from σ.

Final comments

- for pricing multiple European options, cheaper to solve one Forward Kolmogorov equation for evolution of density, rather than multiple Backward Kolmogorv (Black-Scholes) equations for option value
- doesn't work for American or Bermudan options because they're nonlinear
- for sensitivity calculations, the big benefit from adjoint methods comes (as usual) when there are lots of sensitivities to be computed – local volatility case is a good example
- must remember there's a potential loss of accuracy when differentiating – a good approximation to the option value does not necessarily imply a good approximation to the Greeks