Advanced Monte Carlo Methods: American Options

Prof. Mike Giles

mike.giles@maths.ox.ac.uk

Oxford University Mathematical Institute

Early Exercise

Perhaps the biggest challenge for Monte Carlo methods is the accurate and efficient pricing of options with optional early exercise:

- Bermudan options: can exercise at a finite number of times t_j
- American options: can exercise at any time

The challenge is to find/approximate the optimal strategy (i.e. when to exercise) and hence determine the price and Greeks.

Early Exercise

Approximating the optimal exercise boundary introduces new approximation errors:

- An approximate exercise boundary is inevitably sub-optimal
  $\Longrightarrow$ under-estimate of "true" value, but accurate value for the sub-optimal strategy
- For the option buyer, sub-optimal price reflects value achievable with sub-optimal strategy
- For the option seller, "true" price is best a purchaser might achieve
- Can also derive an upper bound approximation

Early Exercise

Why is early exercise so difficult for Monte Carlo methods?

- leads naturally to a dynamic programming formulation working backwards in time
- fairly minor extension for finite difference methods which already march backwards in time
- doesn't fit well with Monte Carlo methods which go forwards in time

Problem Formulation

Following description in Glasserman's book, the Bermudan problem has the dynamic programming formulation:

$$\begin{aligned}\widetilde{V}_m(x) &= \widetilde{h}_m(x)\\ \widetilde{V}_{i-1}(x) &= \max\left(\widetilde{h}_{i-1}(x), \mathbb{E}[D_{i-1,i}\,\widetilde{V}_i(X_i) \mid X_{i-1}=x]\right)\end{aligned}$$

where

- X_i is the underlying at exercise time t_i
- $\widetilde{V}_i(x)$ is option value at time t_i assuming not previously exercised
- $\widetilde{h}_i(x)$ is exercise value at time t_i
- $D_{i-1,i}$ is the discount factor for interval $[t_{i-1}, t_i]$

Problem Formulation

By defining

$$\begin{aligned}h_i(x) &= D_{0,i}\,\widetilde{h}_i(x)\\ V_i(x) &= D_{0,i}\,\widetilde{V}_i(x)\end{aligned}$$

where

$$D_{0,i} = D_{0,1}\; D_{1,2}\; \ldots\; D_{i-1,i}$$

can simplify the formulation to

$$\begin{aligned}V_m(x) &= h_m(x)\\ V_{i-1}(x) &= \max\left(h_{i-1}(x), \mathbb{E}[V_i(X_i) \mid X_{i-1}=x]\right)\end{aligned}$$

Problem Formulation

An alternative point of view considers stopping rules τ, the time at which the option is exercised.

For a particular stopping rule, the initial option value is

$$V_0(X_0) = \mathbb{E}[h_\tau(X_\tau)],$$

the expected value of the option at the time of exercise.

The best that can be achieved is then

$$V_0(X_0) = \sup_\tau \mathbb{E}[h_\tau(X_\tau)]$$

giving an optimisation problem.

Problem Formulation

The continuation value is

$$C_i(x) = \mathbb{E}[V_{i+1}(X_{i+1}) \mid X_i = x]$$

and so the optimal stopping rule is

$$\tau = \min\{i : h_i(X_i) > C_i(X_i)\}$$

Approximating the continuation value leads to an approximate stopping rule.

Longstaff-Schwartz Method

The Longstaff-Schwartz method (2001) is the one most used in practice.

Start with N path simulations, each going from initial time $t=0$ to maturity $t=T=t_m$.

Problem is to assign a value to each path, working out whether and when to exercise the option.

This is done by working backwards in time, approximating the continuation value.

Longstaff-Schwartz Method

At maturity, the value of an option is

$$V_m(X_m) = h_m(X_m)$$

At the previous exercise date, the continuation value is

$$C_{m-1}(x) = \mathbb{E}[V_m(X_m) \mid X_{m-1} = x]$$

This is approximated using a set of R basis functions as

$$\widehat{C}_{m-1}(x) = \sum_{r=1}^{R} \beta_r \, \psi_r(x)$$

Longstaff-Schwartz Method

The coefficients β_r are obtained by a least-squares minimisation, minimising

$$\mathbb{E}\left\{ \left(\mathbb{E}[V_m(X_m) \mid X_{m-1}] - \widehat{C}_{m-1}(X_{m-1}) \right)^2 \right\}$$

Setting the derivative w.r.t. β_r to zero gives

$$\mathbb{E}\left\{ \left(\mathbb{E}[V_m(X_m) \mid X_{m-1}] - \widehat{C}_{m-1}(X_{m-1}) \right) \psi_r(X_{m-1}) \right\} = 0$$

and hence

$$\begin{aligned} \mathbb{E}[V_m(X_m) \, \psi_r(X_{m-1})] &= \mathbb{E}[\widehat{C}_{m-1}(X_{m-1}) \, \psi_r(X_{m-1})] \\ &= \sum_s \mathbb{E}[\psi_r(X_{m-1}) \, \psi_s(X_{m-1})] \, \beta_s \end{aligned}$$

Longstaff-Schwartz Method

This set of equations can be written collectively as

$$B_{\psi\psi} \, \beta = B_{V\psi}$$

where

$$\begin{aligned} (B_{V\psi})_r &= \mathbb{E}[V_m(X_m) \, \psi_r(X_{m-1})] \\ (B_{\psi\psi})_{rs} &= \mathbb{E}[\psi_r(X_{m-1}) \, \psi_s(X_{m-1})] \end{aligned}$$

Therefore,

$$\beta = B_{\psi\psi}^{-1} \, B_{V\psi}$$

Longstaff-Schwartz Method

In the numerical approximation, each of the expectations is replaced by an average over the values from the N paths.

For example,

$$\mathbb{E}[\psi_r(X_{m-1})\,\psi_s(X_{m-1})]$$

is approximated as

$$N^{-1}\sum_{n=1}^{N}\psi_r(X_{m-1}^{(n)})\;\psi_s(X_{m-1}^{(n)})$$

Assuming that the number of paths is much greater than the number of basis functions, the main cost is in approximating $B_{\psi\psi}$ with a cost which is $O(N\,R^2)$.

Longstaff-Schwartz Method

Once we have the approximation for the continuation value, what do we do?

- if $\widehat{C}(X_{m-1}) < h_{m-1}(X_{m-1})$, exercise the option and set

$$V_{m-1} = h_{m-1}(X_{m-1})$$

- if not, then either set

$$V_{m-1} = \widehat{C}(X_{m-1})$$

(Tsitsiklis & van Roy, 1999), or

$$V_{m-1} = V_m$$

(Longstaff & Schwartz, 2001)

Longstaff-Schwartz Method

The Longstaff-Schwarz treatment only uses the continuation estimate to decide on the exercise boundary – no loss of accuracy for paths which are not exercised.

The Tsitsiklis-van Roy treatment introduces more error, especially for American options where it gets applied each timestep.

Also, Longstaff-Schwarz can do least squares fit only for paths which are in-the-money (i.e. $h(X)>0$) – leads to improved accuracy.

Because of the optimality condition, the option value is insensitive to small perturbations in the exercise boundary, so can assume that exercise of paths is not affected when computing first order Greeks.

Longstaff-Schwartz Method

Provided the basis functions are chosen suitably, the approximation

$$\widehat{C}_{m-1}(x) = \sum_{r=1}^{R}\beta_r\,\psi_r(x)$$

gets increasingly accurate as $R\to\infty$. Longstaff & Schwartz used 5-20 basis functions in their paper – I don't know what is standard now in practice.

Having completed the calculation for t_{m-1}, repeat the procedure for t_{m-2} then t_{m-3} and so on. Could use different basis functions for each exercise time – the coefficients β will certainly be different.

Longstaff-Schwartz Method

The estimate will tend to be biased low because of the sub-optimal exercise boundary, however might be biased high due to using the same paths for decision-making and valuation.

To be sure of being biased low, should use two sets of paths, one to estimate the continuation value and exercise boundary, and the other for the valuation.

However, in practice the difference is quite small.

This leaves the problem of computing an upper bound.

Upper Bounds

In Glasserman's Bermudan version of Roger's continuous time result (2002), he lets M_m be a martingale with $M_0\!=\!0$.

For any stopping rule τ, we have

$$\mathbb{E}[h_\tau(X_\tau)] \;=\; \mathbb{E}[h_\tau(X_\tau) - M_\tau] \;\leq\; \mathbb{E}[\max_k(h_k(X_k) - M_k)]$$

This is true for all martingales M and all stopping rules τ and hence

$$V_0(X_0) \;=\; \sup_\tau \mathbb{E}[h_\tau(X_\tau)] \;\leq\; \inf_M \mathbb{E}[\max_k(h_k(X_k) - M_k)]$$

Upper Bounds

The key duality result is that in fact there is equality

$$\sup_\tau \mathbb{E}[h_\tau(X_\tau)] \;=\; \inf_M \mathbb{E}[\max_k(h_k(X_k) - M_k)]$$

so that

- an arbitrary τ gives a lower bound
- an arbitrary M gives an upper bound
- making both of them "better" shrinks the gap between them to zero

Upper Bounds

Glasserman proves by induction that the optimal martingale M is equal to

$$M_k = \sum_1^k \Big(V_i(X_i) - \mathbb{E}[V_i(X_i) \mid X_{i-1}] \Big)$$

To get a good upper bound we approximate this martingale.

Upper Bounds

The approximate martingale for a particular path is defined as

$$\widehat{M}_k = \sum_{1}^{k} \left(V_i(X_i) - P^{-1} \sum_p V_i(X_i^{(p)}) \right)$$

where the $X_i^{(p)}$ are values for X_i from P different mini-paths starting at X_{i-1}, and

$$V_i(X_i) = \max(h_i(X_i), \widehat{C}_i(X_i))$$

with $\widehat{C}_i(X_i)$ being the approximate continuation value given by the Longstaff-Schwartz algorithm.

Glasserman suggests up to 100 mini-paths may be needed.

Final Words

- Bermudan and American options are important applications
- Longstaff-Schwartz method is popular, but still plenty of scope for improvement?
- suspect that finite difference method is used for Greeks?
- is independent second set of paths used in practice?
- are upper bounds used in practice?

Advanced Monte Carlo Methods: Quasi-Monte Carlo

Prof. Mike Giles

mike.giles@maths.ox.ac.uk

Oxford University Mathematical Institute

Quasi Monte Carlo

- low discrepancy sequences
- Koksma-Hlawka inequality
- rank-1 lattice rules and Sobol sequences
- randomised QMC
- identification of dominant dimension

Quasi Monte Carlo

Standard Monte Carlo approximates high-dimensional hypercube integral

$$\int_{[0,1]^d} f(x) \, \mathrm{d}x$$

by

$$\frac{1}{N} \sum_{i=1}^{N} f(x^{(i)})$$

with points chosen randomly, giving

- r.m.s. error proportional to $N^{-1/2}$
- confidence interval

Quasi Monte Carlo

Standard quasi Monte Carlo uses the same equal-weight estimator

$$\frac{1}{N} \sum_{i=1}^{N} f(x^{(i)})$$

but chooses the points systematically so that

- error roughly proportional to N^{-1}
- no confidence interval

(We'll get the confidence interval back later by adding in some randomisation!)

Low Discrepancy Sequences

The key is to use points which are fairly uniformly spread within the hypercube, not clustered anywhere.

The star discrepancy $D^*_N(x^{(1)}, \ldots x^{(N)})$ of a set of N points is defined as

$$D^*_N = \sup_{B \in J} \left| \frac{A(B)}{N} - \lambda(B) \right|$$

where J is the set of all hyper-rectangles of the form

$$\prod [u_i^-, u_i^+], \quad u_i^\pm \in [0,1],$$

$A(B)$ is the number of points in B, and $\lambda(B)$ is the volume (or measure) of B.

Low Discrepancy Sequences

There are sequences for which

$$D^*_N \le C \frac{(\log N)^d}{N}$$

where d is the dimension of the problem.

This is important because of the Koksma-Hlawka inequality.

Koksma-Hlawka Inequality

$$\left| \frac{1}{N} \sum_{i=1}^{N} f(x^{(i)}) - \int_{[0,1]^d} f(x)\, \mathrm{d}x \right| \le V(f)\; D^*_N(x^{(1)}, \ldots x^{(N)})$$

where $V(f)$ is the Hardy-Krause variation of f defined (for sufficiently differentiable f) as a sum of terms of the form

$$\int_{[0,1]^k} \left| \frac{\partial^k f}{\partial x_{i_1} \ldots \partial x_{i_k}} \right|_{x_j=1, j\neq i_1,\ldots,i_k} \mathrm{d}x$$

with $i_1 < i_2 < \ldots < i_k$ for $k \le d$.

Problem: not a useful error bound

- in finance applications f often isn't even bounded
- even when it is, it's not sufficiently differentiable and estimating $V(f)$ is computationally demanding

Koksma-Hlawka Inequality

However, still useful because of what it tells us about the asymptotic behaviour:

$$\text{Error} < C \frac{(\log N)^d}{N}$$

- for small dimension d, ($d < 10$?) this is much better than $N^{-1/2}$ r.m.s. error for standard MC
- for large dimension d, $(\log N)^d$ could be enormous, so not clear there is any benefit

Rank-1 Lattice Rule

A rank-1 lattice rule has the simple construction

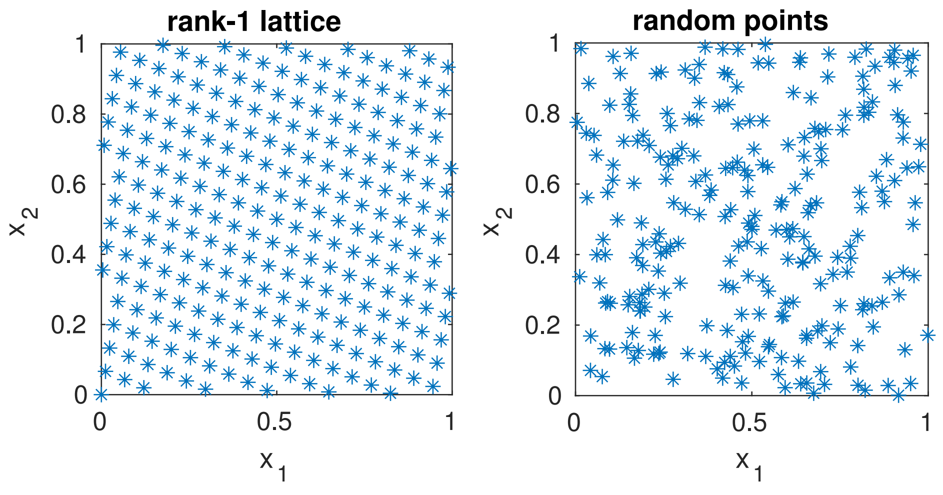
$$x^{(i)} = \frac{i}{N} z \mod 1$$

where z is a d-dimensional “generating vector”,
and $r \mod 1$ means dropping the integer part of r

In each dimension k, the values $x_k^{(i)}$ are a permutation of the equally spaced points $0, 1/N, 2/N \ldots (N-1)/N$ which is great for integrands f which vary only in one dimension.

Also very good if $f(x) = \sum_k f_k(x_k)$.

Rank-1 Lattice Rule



Sobol Sequences

The most popular QMC approach uses Sobol sequences $x^{(i)}$ which have the property that for small dimensions $d<40$ the subsequence

$$2^m \le i < 2^{m+1}$$

of length 2^m has precisely 2^{m-d} points in each of the little cubes of volume 2^{-d} formed by bisecting the unit hypercube in each dimension, and similar properties hold with other pieces.

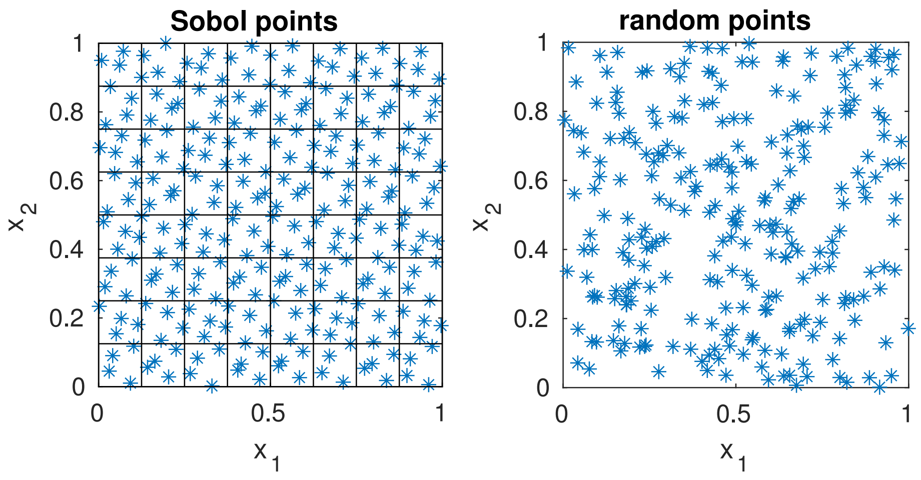
Sobol Sequences

For example:

- cutting it into halves in any dimension, each has 2^{m-1} points
- cutting it into quarters in any dimension, each has 2^{m-2} points
- cutting it into halves in one direction, then halves in another direction, each quarter has 2^{m-2} points
- etc.

The generation of these sequences is a bit complicated, but it is fast and plenty of software is available to do it.

Sobol sequences



Randomised QMC

In the best cases, QMC error is $O(N^{-1})$ instead of $O(N^{-1/2})$ but without a confidence interval.

To get a confidence interval using a rank-1 lattice rule, we use several sets of QMC points, with the N points in set m defined by

$$x^{(i,m)} = \left(\frac{i}{N}\, z \; + \; x^{(m)}\right) \quad \mod 1$$

where $x^{(m)}$ is a random offset vector.

Randomised QMC

For each m, let

$$\overline{f}_m = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i,m)})$$

This is a random variable, and since $\mathbb{E}[f(x^{(i,m)})] = \mathbb{E}[f]$ it follows that $\mathbb{E}[\overline{f}_m] = \mathbb{E}[f]$

By using multiple sets, we can estimate $\mathbb{V}[\overline{f}]$ in the usual way and so get a confidence interval

More sets $\Longrightarrow$ better variance estimate, but poorer error. Some people use as few as 10 sets, but I prefer 32.

Randomised QMC

For Sobol sequences, randomisation is achieved through digital scrambling:

$$x^{(i,m)} = x^{(i)} \veebar X^{(m)}$$

where the exclusive-or operation $\veebar$ is applied bitwise so that

$$\begin{array}{rl} & 0.1010011 \\ \veebar & 0.0110110 \\ = & 0.1100101 \end{array}$$

The benefit of the digital scrambling is that it maintains the special properties of the Sobol sequence.

Dominant Dimensions

QMC points have the property that the points are more uniformly distributed through the lowest dimensions.

Consequently, it is important to think about how the dimensions are allocated to the problem.

Ideally, we'd like to use a change of variables, so the function we're integrating depends only on the first coordinate.

Dominant Dimensions

Suppose we have an European option, based on d multiple underlying assets with

$$\log S_i(T) = \log S_i(0) + \left(r - \tfrac{1}{2}\sigma_i^2\right) T + \sigma_i\, W_i(T)$$

and the $\log S_i(T)$ have covariance matrix Σ.

If U is a d-dimensional QMC point, can produce uncorrelated quasi-random Normals using

$$X_i = \Phi^{-1} U_i$$

but how do we generate correlated quasi-Normals?

Dominant Dimensions

Previously, have generated correlated Normals through

$$Y = L\, X$$

with X i.i.d. $N(0,1)$ Normals, and L is <u>any</u> matrix such that $L\, L^T = \Sigma$.

However, for QMC different L's are equivalent to a change of coordinates and it can make a big difference. Usually best to use a PCA construction

$$L = U\, \Lambda^{1/2}$$

with eigenvalues in diagonal matrix Λ (and associated eigenvectors U) arranged in descending order, from largest ($\Longrightarrow$ most important?) to smallest.

Path-dependent options

Same ingredients as simple European options:

- Sobol or lattice rule quasi-uniform generators
- PCA to best use QMC inputs for multi-dimensional applications
- randomised QMC to regain confidence interval

New ingredient:

- how best to use QMC inputs to generate Brownian increments

Quasi-Monte Carlo

When using standard Normal random inputs for MC simulation, can express expectation as a multi-dimensional integral with respect to inputs

$$V = \mathbb{E}[\widehat{f}(\widehat{S})] = \int \widehat{f}(\widehat{S})\ \phi(Z)\ \mathrm{d}Z$$

where $\phi(Z)$ is multi-dimensional standard Normal p.d.f.

Putting $Z_n = \Phi^{-1} U_n$ turns this into an integral over a M-dimensional hypercube

$$V = \mathbb{E}[\widehat{f}(\widehat{S})] = \int \widehat{f}(\widehat{S})\ \mathrm{d}U$$

Quasi-Monte Carlo

This is then approximated as

$$N^{-1} \sum_n \widehat{f}(\widehat{S}^{(n)})$$

and each path calculation involves the computations

$$U \ \rightarrow\ Z\ \rightarrow\ \Delta W\ \rightarrow\ \widehat{S}\ \rightarrow\ \widehat{f}$$

The key step here is the second, how best to convert the vector Z into the vector ΔW. With standard Monte Carlo, as long as ΔW has the correct distribution, how it is generated is irrelevant, but with QMC it does matter.

Quasi-Monte Carlo

For a scalar Brownian motion $W(t)$ with $W(0)\!=\!0$, defining $W_n\!=\!W(nh)$, each W_n is Normally distributed and for $j \geq k$

$$\mathbb{E}[W_j\, W_k] = \mathbb{E}[W_k^2] + \mathbb{E}[(W_j\!-\!W_k)\, W_k] = t_k$$

since $W_j\!-\!W_k$ is independent of W_k.

Hence, the covariance matrix for W is Ω with elements

$$\Omega_{j,k} = \min(t_j, t_k)$$

Quasi-Monte Carlo

The task now is to find a matrix L such that

$$L\ L^T = \Omega = h \begin{pmatrix} 1 & 1 & \ldots & 1 & 1 \\ 1 & 2 & \ldots & 2 & 2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & 2 & \ldots & M\!-\!1 & M\!-\!1 \\ 1 & 2 & \ldots & M\!-\!1 & M \end{pmatrix}$$

We will consider 3 possibilities:

- Cholesky factorisation
- PCA
- Brownian Bridge treatment

Cholesky factorisation

The Cholesky factorisation gives

$$L = \sqrt{h} \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 1 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & 1 & 0 \\ 1 & 1 & \dots & 1 & 1 \end{pmatrix}$$

and hence

$$W_n = \sum_{m=1}^{n} \sqrt{h}\, Z_m \quad \Longrightarrow \quad \Delta W_n = W_n - W_{n-1} = \sqrt{h}\, Z_n$$

i.e. standard MC approach

PCA construction

The PCA construction uses

$$L = U\,\Lambda^{1/2} = \left(U_1 \,\middle|\, U_2 \,\middle|\, \dots \right) \begin{pmatrix} \lambda_1^{1/2} & & \\ & \lambda_2^{1/2} & \\ & & \dots \end{pmatrix}$$

with the eigenvalues λ_n and eigenvectors U_n arranged in descending order, from largest to smallest.

Numerical computation of the eigenvalues and eigenvectors is costly for large numbers of timesteps, so instead use theory due to Åkesson and Lehoczky (1998)

PCA construction

It is easily verified that

$$\Omega^{-1} = h^{-1} \begin{pmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \dots & \dots & \dots & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 1 \end{pmatrix}.$$

This looks like the finite difference operator approximating a second derivative, and so the eigenvectors are Fourier modes.

PCA construction

The eigenvectors of both Ω^{-1} and Ω are

$$(U_m)_n = \frac{2}{\sqrt{2M+1}} \sin\left(\frac{(2m-1)\, n\pi}{2M+1} \right)$$

and the eigenvalues of Ω are the reciprocal of those of Ω^{-1},

$$\lambda_m = \frac{h}{4} \left(\sin\left(\frac{(2m-1)\, \pi}{2\,(2M+1)} \right) \right)^{-2}$$

Because the eigenvectors are Fourier modes, an efficient FFT transform can be used (Scheicher, 2006) to compute

$$L\,Z \;=\; U\left(\Lambda^{1/2}\,Z\right) \;= \sum_m (\sqrt{\lambda_m}\, Z_m)\, U_m$$

Brownian Bridge construction

The Brownian Bridge construction uses the theory from a previous lecture.

The final Brownian value is constructed using Z_1:

$$W_M = \sqrt{T}\, Z_1$$

Conditional on this, the midpoint value $W_{M/2}$ is Normally distributed with mean $\frac{1}{2}W_M$ and variance $T/4$, and so can be constructed as

$$W_{M/2} = \tfrac{1}{2}W_M + \sqrt{T/4}\, Z_2$$

Brownian Bridge construction

The quarter and three-quarters points can then be constructed as

$$\begin{aligned} W_{M/4} &= \tfrac{1}{2}W_{M/2} + \sqrt{T/8}\, Z_3 \\ W_{3M/4} &= \tfrac{1}{2}(W_{M/2} + W_M) + \sqrt{T/8}\, Z_4 \end{aligned}$$

and the procedure continued recursively until all Brownian values are defined.

(This assumes M is a power of 2 – if not, the implementation is slightly more complex)

I have a slight preference for this method because it is particularly effective for European option for which $S(T)$ is very strongly dependent on $W(T)$.

Multi-dimensional Brownian motion

The preceding discussion concerns the construction of a single, scalar Brownian motion.

Suppose now that we have to generate a P-dimensional Brownian motion with correlation matrix Σ between the different components.

What do we do?

Multi-dimensional Brownian motion

First, using either PCA or BB to construct P uncorrelated Brownian paths using

- $Z_1, Z_{1+P}, Z_{1+2P}, Z_{1+3P}, \ldots$ for first path
- $Z_2, Z_{2+P}, Z_{2+2P}, Z_{2+3P}, \ldots$ for second path
- $Z_3, Z_{3+P}, Z_{3+2P}, Z_{3+3P}, \ldots$ for third path
- etc.

This uses the “best” dimensions of Z for the overall behaviour of all of the paths.

Multi-dimensional Brownian motion

Second, define

$$W_n^{corr} = L_\Sigma \, W_n^{uncorr} \quad \Longrightarrow \quad \Delta W_n^{corr} = L_\Sigma \, \Delta W_n^{uncorr}$$

where W_n^{uncorr} is the uncorrelated sequence, W_n^{corr} is the correlated sequence, and

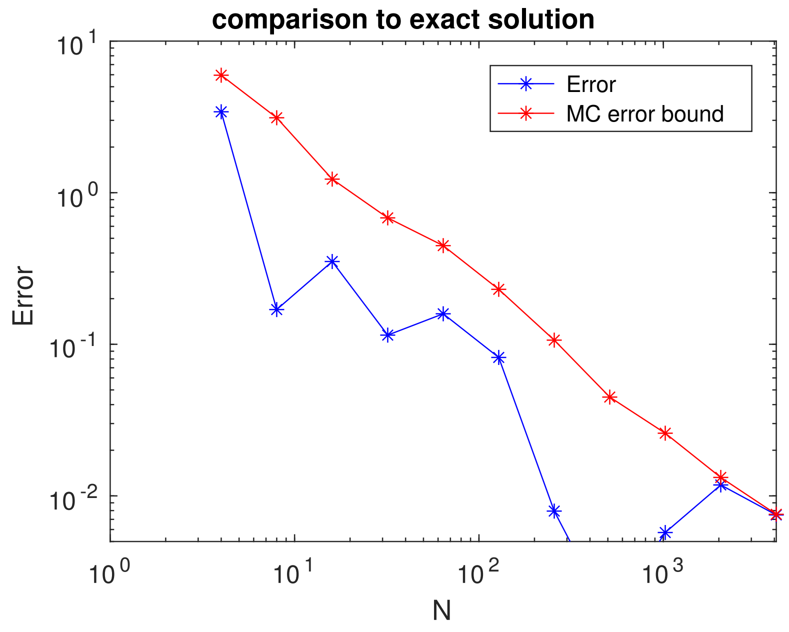
$$L_\Sigma \, L_\Sigma^T = \Sigma$$

Numerical results

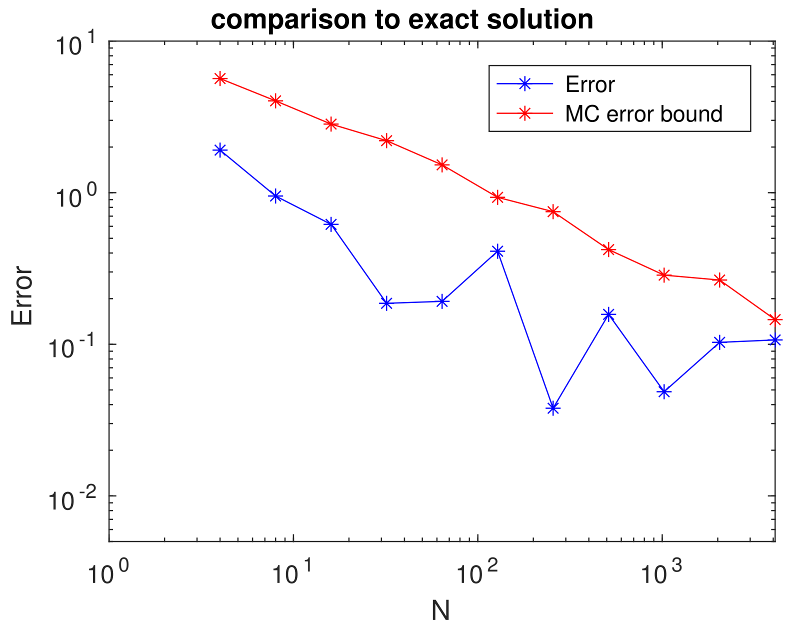
Usual European call test case based on geometric Brownian motion:

- 128 timesteps so weak error is negligible
- comparison between
  - QMC using Brownian Bridge
  - QMC without Brownian Bridge
  - standard MC
- QMC calculations use Sobol generator
- all calculations use 64 “sets” of points – for QMC calcs, each has a different random offset
- plots show error and 3 s.d. error bound

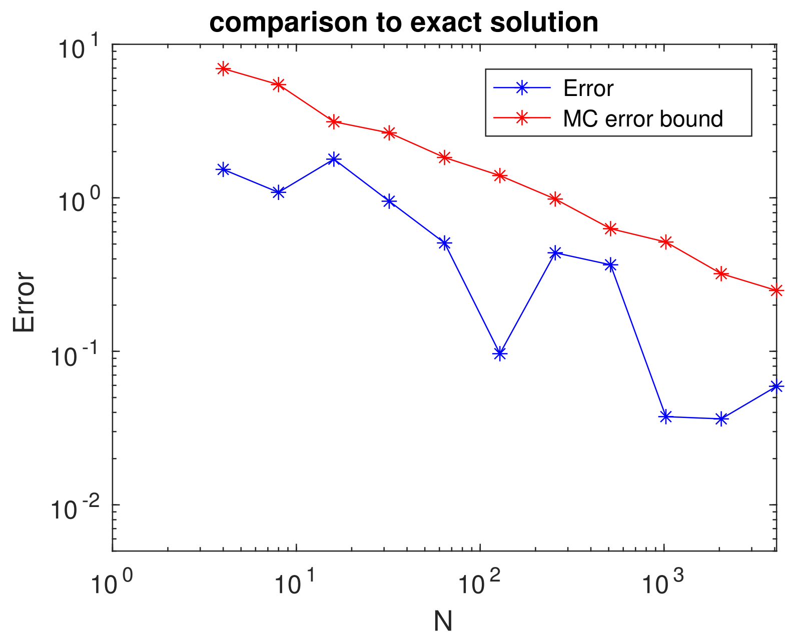
QMC with Brownian Bridge



QMC without Brownian Bridge



Standard Monte Carlo



Final words

- QMC offers large computational savings over the standard Monte Carlo approach
- best to use randomised QMC to regain confidence intervals, at the cost of slightly poorer accuracy
- very important to use PCA or Brownian Bridge construction to create discrete Brownian increments – much better than “standard” approach which is equivalent to Cholesky factorisation of covariance matrix

Advanced Monte Carlo Methods: Computing Greeks

Prof. Mike Giles

mike.giles@maths.ox.ac.uk

Oxford University Mathematical Institute

Outline

Computing Greeks

- finite differences
- likelihood ratio method
- pathwise sensitivities
- “Smoking adjoints” implementation

SDE path simulation

For the generic stochastic differential equation

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

an Euler approximation with timestep h is

$$\widehat{S}_{n+1} = \widehat{S}_n + a(\widehat{S}_n)\,h + b(\widehat{S}_n)\,Z_n\,\sqrt{h},$$

where Z is a $N(0,1)$ random variable. To estimate the value of a European option

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

we take the average of N paths with M timesteps:

$$\widehat{V} = N^{-1} \sum_i f(\widehat{S}_M^{(i)}).$$

Greeks

As in Module 2, in addition to estimating the expected value

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

we also want to know a whole range of “Greeks” corresponding to first and second derivatives of V with respect to various parameters:

$$\Delta = \frac{\partial V}{\partial S_0}, \qquad \Gamma = \frac{\partial^2 V}{\partial S_0^2},$$

$$\rho = \frac{\partial V}{\partial r}, \qquad \text{Vega} = \frac{\partial V}{\partial \sigma}.$$

These are needed for hedging and for risk analysis.

Finite difference sensitivities

If $V(\theta) = \mathbb{E}[f(S(T))]$ for a particular value of an input parameter θ, then the sensitivity $\dfrac{\partial V}{\partial \theta}$ can be approximated by one-sided finite difference

$$\frac{\partial V}{\partial \theta} = \frac{V(\theta{+}\Delta\theta) - V(\theta)}{\Delta\theta} \; + \; O(\Delta\theta)$$

or by central finite difference

$$\frac{\partial V}{\partial \theta} = \frac{V(\theta{+}\Delta\theta) - V(\theta{-}\Delta\theta)}{2\Delta\theta} \; + \; O((\Delta\theta)^2)$$

Nothing changes here from Module 2 because of the path simulation.

Finite difference sensitivities

As before, the clear advantage of this approach is that it is very simple to implement (hence the most popular in practice?)

However, the disadvantages are:

- expensive (2 extra sets of calculations for central differences)
- significant bias error if $\Delta\theta$ too large
- large variance if $f(S(T))$ discontinuous and $\Delta\theta$ small

Also, very important to use the same random numbers for the “bumped” path simulations to minimise the variance.

Likelihood ratio method

As a recap from Module 2, if we define $p(S)$ to the probability density function for the final state $S(T)$, then

$$V = \mathbb{E}[f(S(T))] = \int f(S)\, p(S)\, \mathrm{d}S,$$

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

The quantity $\dfrac{\partial (\log p)}{\partial \theta}$ is sometimes called the “score function”.

Likelihood ratio method

Extending LRM to a SDE path simulation with M timesteps, with the payoff a function purely of the discrete states $\widehat{S}_n$, we have the M-dimensional integral

$$V = \mathbb{E}[f(\widehat{S})] = \int f(\widehat{S})\, p(\widehat{S})\, \mathrm{d}\widehat{S},$$

where $\quad \mathrm{d}\widehat{S} \equiv \mathrm{d}\widehat{S}_1\, \mathrm{d}\widehat{S}_2\, \mathrm{d}\widehat{S}_3\, \ldots\, \mathrm{d}\widehat{S}_M$

and $p(\widehat{S})$ is the product of the p.d.f.s for each timestep

$$\begin{aligned} p(\widehat{S}) &= \prod_n p_n(\widehat{S}_{n+1} | \widehat{S}_n) \\ \log p(\widehat{S}) &= \sum_n \log p_n(\widehat{S}_{n+1} | \widehat{S}_n) \end{aligned}$$

Likelihood ratio method

For the Euler approximation of GBM,

$$\log p_n = -\log \widehat{S}_n - \log \sigma - \tfrac{1}{2}\log(2\pi h) - \tfrac{1}{2}\,\frac{\left(\widehat{S}_{n+1}-\widehat{S}_n(1+r\,h)\right)^2}{\sigma^2\,\widehat{S}_n^2\,h}$$

$$\begin{aligned}\Longrightarrow\quad \frac{\partial(\log p_n)}{\partial\sigma} &= -\frac{1}{\sigma} + \frac{\left(\widehat{S}_{n+1}-\widehat{S}_n(1+r\,h)\right)^2}{\sigma^3\,\widehat{S}_n^2\,h}\\ &= \frac{Z_n^2-1}{\sigma}\end{aligned}$$

where Z_n is the unit Normal defined by

$$\widehat{S}_{n+1}-\widehat{S}_n(1+r\,h) = \sigma\,\widehat{S}_n\,\sqrt{h}\,Z_n$$

Likelihood ratio method

Hence, the approximation of Vega is

$$\frac{\partial}{\partial\sigma}\,\mathbb{E}[f(\widehat{S}_M)] = \mathbb{E}\left[\left(\sum_n \frac{Z_n^2-1}{\sigma}\right) f(\widehat{S}_M)\right]$$

Note that again this gives zero for $f(S)\equiv 1$.

Note also that $\mathbb{V}[Z_n^2-1]=2$ and therefore

$$\mathbb{V}\left[\left(\sum_n \frac{Z_n^2-1}{\sigma}\right) f(\widehat{S}_M)\right] \;=\; O(M) \;=\; O(T/h)$$

This $O(h^{-1})$ blow-up is the great drawback of the LRM.

Pathwise sensitivities

Under certain conditions (e.g.$f(S), a(S,t), b(S,t)$ all continuous and piecewise differentiable)

$$\frac{\partial}{\partial\theta}\,\mathbb{E}[f(S(T))] = \mathbb{E}\left[\frac{\partial f(S(T))}{\partial\theta}\right] = \mathbb{E}\left[\frac{\partial f}{\partial S}\,\frac{\partial S(T)}{\partial\theta}\right].$$

with $\dfrac{\partial S(T)}{\partial\theta}$ computed by differentiating the path evolution.

Pros:

- less expensive (1 cheap calculation for each sensitivity)
- no bias

Cons:

- can't handle discontinuous payoffs

Pathwise sensitivities

In Module 2, when we could directly sample $S(T)$ this led to the estimator

$$\frac{1}{N}\sum_{i=1}^{N}\frac{\partial f}{\partial S}(S^{(i)})\,\frac{\partial S^{(i)}}{\partial\theta}$$

which is the derivative of the usual price estimator

$$\frac{1}{N}\sum_{i=1}^{N} f(S^{(i)})$$

Gives incorrect estimates when $f(S)$ is discontinuous.
e.g. for digital put $\dfrac{\partial f}{\partial S}=0$ so estimated value of Greek is zero – clearly wrong.

Pathwise sensitivities

Returning to the generic stochastic differential equation

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

an Euler approximation with timestep h gives

$$\widehat{S}_{n+1} = F_n(\widehat{S}_n) \equiv \widehat{S}_n + a(\widehat{S}_n)\,h + b(\widehat{S}_n)\,Z_n\,\sqrt{h}.$$

Defining $\Delta_n = \dfrac{\partial \widehat{S}_n}{\partial S_0}$, then $\Delta_{n+1} = D_n\,\Delta_n$, where

$$D_n \;\equiv\; \frac{\partial F_n}{\partial \widehat{S}_n} \;=\; I + \frac{\partial a}{\partial S}\,h + \frac{\partial b}{\partial S}\,Z_n\,\sqrt{h}.$$

Pathwise sensitivities

The payoff sensitivity to the initial state (Deltas) is then

$$\frac{\partial f(\widehat{S}_N)}{\partial S_0} = \frac{\partial f(\widehat{S}_N)}{\partial \widehat{S}_N}\,\Delta_N$$

If $S(0)$ is a vector of dimension m, then each timestep

$$\Delta_{n+1} = D_n\,\Delta_n,$$

involves a $m\times m$ matrix multiplication, with $O(m^3)$ CPU cost – costly, but still cheaper than finite differences which are also $O(m^3)$ but with a larger coefficient.

Cost may be less in practice because D_n may have a lot of zero entries.

Pathwise sensitivities

To calculate the sensitivity to other parameters (such as volatility $\Longrightarrow$ vegas) consider a generic parameter θ.

Defining $\Theta_n = \partial\widehat{S}_n/\partial\theta$, then

$$\Theta_{n+1} = \frac{\partial F_n}{\partial \widehat{S}_n}\,\Theta_n + \frac{\partial F_n}{\partial \theta} \equiv D_n\,\Theta_n + B_n,$$

and hence

$$\frac{\partial f}{\partial \theta} = \frac{\partial f(\widehat{S}_N)}{\partial \widehat{S}_N}\,\Theta_N$$

Vega example

Suppose we have a down-and-out barrier option based on a single GBM asset, and we want to compute vega.

Euler approximation with timestep h:

$$\widehat{S}_{n+1} = F_n(\widehat{S}_n) \equiv \widehat{S}_n + r\,\widehat{S}_n\,h + \sigma\,\widehat{S}_n\,Z_n\,\sqrt{h}$$

Differentiating this gives:

$$\frac{\partial \widehat{S}_{n+1}}{\partial \sigma} = \frac{\partial \widehat{S}_n}{\partial \sigma}\left(1 + r\; + \sigma Z_n\,\sqrt{h}\right) + \widehat{S}_n\,Z_n\,\sqrt{h}$$

with initial condition $\dfrac{\partial \widehat{S}_0}{\partial \sigma} = 0.$

Vega example

Using the treatment discussed in Module 4, where $p_n = p_n(\widehat{S}_n, \widehat{S}_{n+1}, \sigma)$ is conditional probability of being across the barrier in n^{th} timestep, the discounted payoff is

$$\exp(-rT)\ (\widehat{S}_N - K)^+\ P_N$$

where

$$P_n = \prod_{m=0}^{n-1} (1 - p_m),$$

is probability of not crossing the barrier in first n timesteps, and $P_0 = 0$.

Vega example

Since

$$P_{n+1} = P_n\,(1 - p_n)$$

then

$$\frac{\partial P_{n+1}}{\partial \sigma} = \frac{\partial P_n}{\partial \sigma}\,(1 - p_n) - P_n \left(\frac{\partial p_n}{\partial \widehat{S}_n} \frac{\partial \widehat{S}_n}{\partial \sigma} + \frac{\partial p_n}{\partial \widehat{S}_{n+1}} \frac{\partial \widehat{S}_{n+1}}{\partial \sigma} + \frac{\partial p_n}{\partial \sigma} \right)$$

with initial condition $\dfrac{\partial P_0}{\partial \sigma} = 0$.

The payoff sensitivity is then

$$\exp(-rT) \left(\mathbf{1}_{\widehat{S}_N > K} \frac{\partial \widehat{S}_N}{\partial \sigma}\ P_N + (\widehat{S}_N - K)^+ \frac{\partial P_N}{\partial \sigma} \right)$$

Automatic Differentiation

Generating the pathwise sensitivity code is tedious, but straightforward, and can be automated:

- source-source code generation: takes an old code for payoff evaluation and produces a new code which also computes sensitivities
- operator overloading: defines new object (value + sensitivity), and re-defines operations appropriately e.g.

$$\begin{pmatrix} a \\ \dot{a} \end{pmatrix} * \begin{pmatrix} b \\ \dot{b} \end{pmatrix} \equiv \begin{pmatrix} a\,b \\ \dot{a}\,b + a\,\dot{b} \end{pmatrix}$$

For more information, see
`www.autodiff.org/`
`people.maths.ox.ac.uk/gilesm/libor/`

Discontinuous payoffs

Pathwise sensitivity needs the payoff to be continuous.

What can you do when it is not?

- for digital options, can use a crude piecewise linear approximation
- alternatively, use conditional expectations which effectively smooth the payoff
  - the barrier option is a good example of this, using the probability of crossing conditional on the path values at discrete times
  - Glasserman discusses a similar approach for digital options, stopping the path simulation one timestep early then taking a conditional expectation

Discontinuous payoffs

Glasserman's approach has problems in multiple dimensions (hard to evaluate expected value analytically) so I developed an approach I call "vibrato Monte Carlo".

It is a hybrid method. Conditional on the path value $\widehat{S}_{N-1}$ one timestep before the end, the value value $\widehat{S}_N$ has a Normal distribution, if using an Euler discretisation.

Hence, can use LRM for the final timetsep to get the sensitivity to changes in $\widehat{S}_{N-1}$, and combine this with pathwise to get sensitivity of $\widehat{S}_{N-1}$ to the input parameters.

M.B. Giles, 'Vibrato Monte Carlo sensitivities', pp. 369-392 in Monte Carlo and Quasi Monte Carlo Methods 2008, Springer, 2009.

Adjoint approach

The adjoint (or reverse mode AD) approach computes the same values as the standard (forward) pathwise approach, but much more efficiently for the sensitivity of a single output to multiple inputs.

The approach has a long history in applied math and engineering:

- optimal control theory (find control which achieves target and minimizes cost)
- design optimization (find shape which maximizes performance)

Adjoint approach

Returning to the generic stochastic o.d.e.

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

with Euler approximation

$$\widehat{S}_{n+1} = F_n(\widehat{S}_n) \equiv \widehat{S}_n + a(\widehat{S}_n)\,h + b(\widehat{S}_n)\,Z_n\,\sqrt{h}$$

if $\Delta_n = \dfrac{\partial \widehat{S}_n}{\partial S_0}$, then $\Delta_{n+1} = D_n\,\Delta_n, \quad D_n \equiv \dfrac{\partial F_n(\widehat{S}_n)}{\partial \widehat{S}_n}$,
and hence

$$\frac{\partial f(\widehat{S}_N)}{\partial S_0} = \frac{\partial f(\widehat{S}_N)}{\partial \widehat{S}_N}\Delta_N = \frac{\partial f}{\partial S}\,D_{N-1}\,D_{N-2}\ldots\,D_0\,\Delta_0$$

Adjoint approach

If S is m-dimensional, then D_n is an $m\times m$ matrix, and the computational cost per timestep is $O(m^3)$.

Alternatively,

$$\frac{\partial f(\widehat{S}_N)}{\partial S_0} = \frac{\partial f}{\partial S}\,D_{N-1}\,D_{N-2}\cdots D_0\,\Delta_0 = V_0^T\,\Delta_0,$$

where adjoint $V_n = \left(\dfrac{\partial f(\widehat{S}_N)}{\partial \widehat{S}_n}\right)^T$ is calculated from

$$V_n = D_n^T V_{n+1}, \quad V_N = \left(\frac{\partial f}{\partial \widehat{S}_N}\right)^T,$$

at a computational cost which is $O(m^2)$ per timestep.

Adjoint approach

Note the flow of data within the path calculation:

$$\begin{tikzcd}
\boxed{S_0} \arrow[r] \arrow[dd, dashed, "D_0"] & \boxed{S_1} \arrow[r] \arrow[dd, dashed, "D_1"] & \cdots \arrow[r] & \boxed{S_{N-1}} \arrow[r] \arrow[dd, dashed, "D_{N-1}"] & \boxed{S_N} \arrow[dr, bend left] & \\
 & & & & & \boxed{\partial f/\partial S} \arrow[dl, bend left] \\
\boxed{V_0} & \boxed{V_1} \arrow[l] & \cdots \arrow[l] & \boxed{V_{N-1}} \arrow[l] & \boxed{V_N} \arrow[l] &
\end{tikzcd}$$

– memory requirements are not significant because data only needs to be stored for the current path.

Adjoint approach

To calculate the sensitivity to other parameters, consider a generic parameter θ. Defining $\Theta_n = \partial \widehat{S}_n / \partial\theta$, then

$$\Theta_{n+1} = \frac{\partial F_n}{\partial S}\,\Theta_n + \frac{\partial F_n}{\partial \theta} \equiv D_n\,\Theta_n + B_n,$$

and hence

$$\begin{aligned}
\frac{\partial f}{\partial \theta} &= \frac{\partial f}{\partial \widehat{S}_N}\,\Theta_N \\
&= \frac{\partial f}{\partial \widehat{S}_N}\Big\{ B_{N-1} + D_{N-1} B_{N-2} + \ldots \\
&\qquad\qquad\qquad + D_{N-1} D_{N-2} \ldots D_1 B_0 \Big\} \\
&= \sum_{n=0}^{N-1} V_{n+1}^T B_n .
\end{aligned}$$

Adjoint approach

Different θ's have different B's, but same V's

$\Longrightarrow$ Computational cost $\simeq m^2 \; + \; m \times$ # parameters,

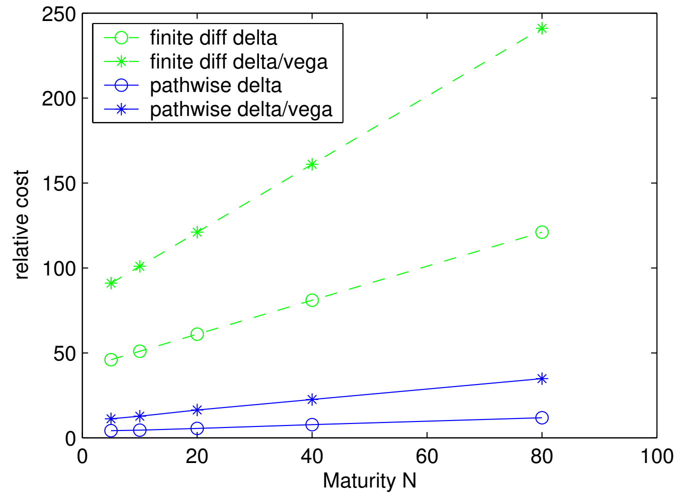
compared to the standard forward approach for which

Computational cost $\simeq m^2 \times$ # parameters.

However, the adjoint approach only gives the sensitivity of one output, whereas the forward approach can give the sensitivities of multiple outputs for little additional cost.

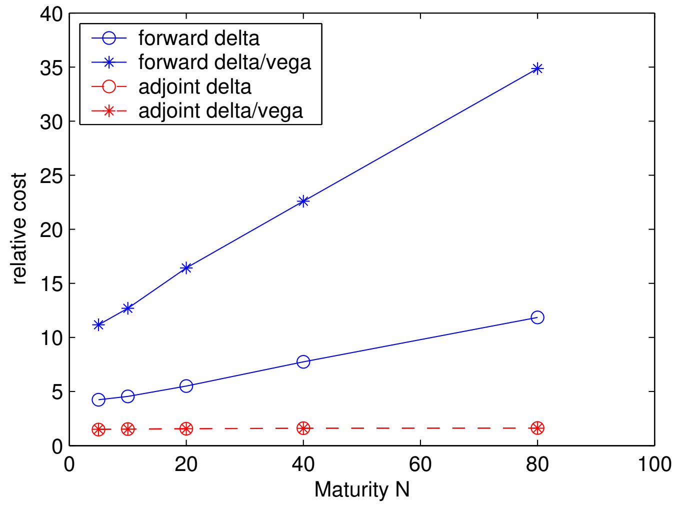
LIBOR Market Model

Finite differences versus forward pathwise sensitivities:



LIBOR Market Model

Forward versus adjoint pathwise sensitivities:



Conclusions

- Greeks are vital for hedging and risk analysis
- Finite difference approximation is simplest to implement, but far from ideal
- Likelihood ratio method for discontinuous payoffs
- In all other cases, pathwise sensitivities are best
- Payoff smoothing may handle the problem of discontinuous payoffs
- Adjoint pathwise approach gives an unlimited number of sensitivities for a cost comparable to the initial valuation

References

- M.B. Giles, P. Glasserman. ’Smoking adjoints: fast Monte Carlo Greeks’, RISK, 19(1):88-92, January 2006.
- M. Leclerc, Q. Liang, I. Schneider, ’Fast Monte Carlo Bermudan Greeks’, RISK, 22(7):84-88, 2009.
- L. Capriotti, M.B. Giles. ’Fast correlation Greeks by adjoint algorithmic differentiation’, RISK, 23(4):77-83, 2010.
- L. Capriotti, J. Lee, M. Peacock, ’Real Time Counterparty Credit Risk Management in Monte Carlo’, RISK 24(6):86-90, 2011.
- L. Capriotti, ’Fast Greeks by algorithmic differentiation’, Journal of Computational Finance 14(3):3-35, 2011.
- L. Capriotti, M.B. Giles. ’Algorithmic differentiation: adjoint Greeks made easy’, RISK, 25(10), 2012.