Risky Measures of Risk: Error Analysis of Numerical Differentiation

Dr. Harvey J. Stein
Head, Quantitative Finance R&D
Bloomberg L.P.
22 June 2005

Revision: 1.13

1. Overview

- Issues in numerical differentiation
  - Roundoff error
  - Convexity error
  - Cancellation error
  - Correlated errors
- Methods to improve accuracy
  - $h$ control
  - Smoothing techniques
  - Computation specific approaches

2. Derivatives as finite difference

To compute the derivative of a function $f$, one must compute

$$f'(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}.$$ 

Although one could try to take this limit numerically, this is much work. More commonly, one chooses a small value of $h = h_0$, and tries to verify that the approximation

$$f'(x) \approx f'_r(x) = \frac{f(x + h_0) - f(x)}{h_0}$$

is sufficiently close to the desired derivative.

But, once we’re not taking the limit, we run into questions, such as:

What value of $h_0$?

Why $f'_r$, and not $f'_l(x) = \frac{f(x) - f(x-h_0)}{h_0}$ or $f'_c = \frac{f(x+h_0) - f(x-h_0)}{2h_0}$?

3. Investigating behavior as a function of $h_0$

Naively, one might think that since we want the limit as $h$ goes to zero, small is better.

However, graphing the computed derivatives as functions of the step size indicates otherwise.
As can be seen in the above graph, $h_0 = 10^{-13}$ exhibits some noise (which, I’m afraid is barely visible on this slide).

As $h_0$ is decreased, the approximation gets worse and worse, with $h_0 = 10^{-17}$ giving complete nonsense - a derivative approximation that’s mostly zero:

Using $h_0 = 10^{-14}$ gives visible noise:

Why does the derivative look so bad for $h_0 = 10^{-17}$? Because then $h_0$ is disappearing into the resolution of our computer arithmetic:

Doubles by the IEEE standard are 64 bits long with a 53 bit mantissa. For any given exponent, one can only represent $2^{52}$ different positive values (one bit is
the sign bit). $2^{-52} \approx 10^{-16}$, so we only get to use at most 16 decimal digits to represent a mantissa. In particular, our computers think that $1 + 10^{-17} = 1$. This discreteness affects the input value, the output value, and all intermediate computations.

Inspecting the above graph, we see that a stepsize of $10^{-16}$ for $x$ values around 1 will give either zero or a ridiculously high value for the derivative. $10^{-15}$ will also give noisy derivatives, depending on where the actual $x + h_0$ and $x - h_0$ lie on the step function that constitutes the normal CDF at this level of magnification.

There’s another effect that’s commonly discussed. The fact that $(x+h) - (x-h) \neq 2h$ because of roundoff error. This means that the denominator of our approximation shouldn’t really be $h$. There are two ways to fix this. One is to use a power of 2 for $h$ instead of a power of 10. The other is to set $h = (x+h) - x$. The later requires a little bit of effort to prevent an optimizing compiler from reducing it to $h = h$.

But, I’ve only observed minor impact from such adjustments. For clarity, I’ll continue using powers of 10 instead of powers of 2 here.

5. Error analysis 102A - The right $h$

5. Error analysis 102A - The right $h$

But, is this the whole story? Since we’re differentiating the normal CDF, we can compare it to the analytic solution. Comparing our original calculation with $h_0 = 10^{-12}$, we see that the derivative computed wasn’t nearly as accurate as we had thought. In fact, $h_0 = 10^{-10}$ gives much smaller errors:

![Graph showing error comparison]

Clearly, $10^{-8}$ works much better than $10^{-10}$.
Continuing along, we see that $10^{-6}$ works much better than $10^{-8}$:

Finally, we see that $10^{-5}$ gives the best results, with $10^{-4}$ being smooth, but giving a high bias.

To prove I'm not cheating, let's do the same with $f(x) = x^3$:

The best $h_0$ for $f(x) = x^3$ ends up again being around $10^{-5}$. Coincidence?
Let’s look more closely at the error graph using $h_0 = 10^{-5}$:

![Error Graph](image)

It appears to consist of noise plus some periodic error. The noise is from cancellation error, and the periodic component is from convexity error.

### 7. Cancellation error

Consider $f(x + h) - f(x - h)$. Suppose each of these has 3 significant digits of accuracy. How many digits of accuracy are in the difference?

\[
f(x + h) = .335 + \text{noise}
\]

\[
-(f(x - h) = .231 + \text{noise})
\]

\[
\frac{f(x + h) - f(x - h)}{2h} = .104 + \text{noise}
\]

However,

\[
f(x + h) = .335 + \text{noise}
\]

\[
-(f(x - h) = .331 + \text{noise})
\]

\[
\frac{f(x + h) - f(x - h)}{2h} = .004 + \text{noise}
\]

### 6. Convexity error

\[
f(x + h) = f(x) + f'(x)h + \frac{f''(x)}{2!} h^2 + \frac{f'''(x)}{3!} h^3 + \ldots
\]

\[
f(x + h) - f(x - h) = 2f'(x)h + 2\frac{f''(x)}{h^3} 3! + \ldots
\]

\[
\frac{f(x + h) - f(x - h)}{2h} = f'(x) + \frac{f'''(x)}{3!} h^2 + \ldots
\]

When $h$ is large enough, the $\frac{f'''(x)}{3!} h^2$ term contributes to the error. This error tends to zero as $h$ tends to zero.

This is one of the reasons why a centered derivative is favored over a one sided derivative — the $f''h$ error term drops out, so the convexity error is smaller. Note also that the one sided derivative is the same as the two sided derivative computed at half the stepsize and shifted by half the stepsize, so in some sense the error in the one sided derivative is that it’s estimating the derivative at the wrong $x$ value.

### 7. Cancellation error

In the first case, the difference yielded 3 significant digits. In the second case, the high order digits cancelled, leaving only 1 significant digit in the difference. This is cancellation error.

Clearly, cancellation error increases as $h$ decreases. For $h$ sufficiently small, $f(x + h) = f(x - h)$ and the relative error becomes infinite.

In general, we can use relative errors to encode the number of significant digits in our computations. Let

\[
\bar{f}(x) = f(x) + \alpha(x)f(x)
\]

where $\bar{f}(x)$ is what we actually get when computing $f(x)$. Here $\alpha(x)$ is a random quantity that quantifies the relative error in calculating $f(x)$. If we’re accurate to machine precision, then $|\alpha| \leq 2^{-53}$. If our calculation only yields 5 decimal digits of accuracy, then $|\alpha| \leq 10^{-5}$. 

7. Cancellation error

\[ f(x + h) - f(x - h) = f(x + h) - f(x - h) \]
\[ + \alpha(x + h)f(x + h) - \alpha(x - h)f(x - h) \]
\[ \approx f(x + h) - f(x - h) + \alpha(x)f(x) \]

assuming \( h \) is small enough that \( f(x + h) \) and \( f(x - h) \) are about the same magnitude, and that \( \alpha(x + h) \) and \( \alpha(x - h) \) are independent noise, and ignoring the fact that summing them might cause the loss of one additional significant bit. This analysis can be done more carefully, but this will get us into the right ballpark.

This quantifies the problem of cancellation error. The absolute error in calculating \( \Delta f \) is roughly \( \alpha f \). For small \( h \), \( \Delta f \) is small, leaving the error to dominate the calculation.

The best value of \( h \) will balance the cancellation error and the convexity error.

\[ \frac{f(x + h) - f(x - h)}{2h} \approx f'(x) + \frac{f''(x)}{3!}h^2 + \frac{\alpha(x)f(x)}{2h} \]

7. Cancellation error

For \( x^2 \), we know the 3rd derivative is zero. This means that the only error is from cancellation, which means that larger \( h \) should automatically be better. Graphs confirm this theory:

To minimize the error we must minimize \( \left| \frac{f''(x)}{3!}h^2 + \frac{\alpha(x)f(x)}{2h} \right| . \)

\[ \frac{d}{dh} \left( \frac{f''(x)}{3!}h^2 + \frac{\alpha(x)f(x)}{2h} \right) = \frac{f'''(x)h}{3} - \frac{\alpha f f'(x)}{2h^2} \]
\[ = \frac{f'''(x)h^3 - 3\alpha f}{6h^2} \]

implies

\[ f'''(x)h^3 - 3\alpha f = 0 \]
or

\[ h = \sqrt[3]{\frac{3\alpha f}{f'''(x)}} \]

If our calculations are exact except for roundoff error, and \( f \) and \( f''' \) are around the same order of magnitude, then \( \alpha \approx 2^{-53} \) (52 accurate digits, base 2), which gives an optimal \( h \) of about \( 7 \times 10^{-6} \). This ties out with our above empirical work, and indicates that both functions are being computed with around full accuracy.

8. Error analysis 102B - Flying without a reference

8. Error analysis 102B - Flying without a reference

More commonly we’re faced with calculating derivatives without being able to verify against a known analytic formula. If we don’t know the derivative, and we don’t know how much error we have in our function, how does one pick \( h_0 \)?

One way is to inspect higher order derivatives. If our first derivative is jumping around, then the second derivative with the same step size will be visibly noisy.

We’ll use

\[ f''(x) \approx f(x + h) + f(x - h) - 2f(x) \]
and

\[ f'''(x) \approx \frac{f(x + 2h) - 2f(x + h) + 2f(x) - f(x - 2h)}{2h_0^2} \]

so that for the second and third derivative approximations sample \( f \) with the same spacing as the first derivative calculation.
Graphing $f''$ and $f'''$ as a function of step size shows that the second derivative is visibly poor for $h_0 = 10^{-7}$ and that the third derivative is visibly poor for $h_0 = 10^{-5}$:

This at least indicates that something between $10^{-6}$ and $10^{-4}$ is called for when computing $f'$. Can we make this more precise? Maybe, but we haven’t tried.

One might consider trying to integrate the derivative to see how close it comes back to the original function. Unfortunately this doesn’t help because the sum is effectively a telescoping series. If $\Delta x = 2h$, $X = \{x_1, x_1 + \Delta x, x_1 + 2\Delta x, \ldots, x_2\}$, then

$$\sum_X \frac{f(x + h) - f(x - h)}{2h} \Delta x = f(x_2 + h) - f(x_1 - h).$$

In other words, the error in the derivative from one point to the next cancel each other. This is clear because if a given point is too high, then the derivative to the left will be too large while on the right it will be too small.

9. Error analysis 201 - Correlated errors

This is where most error analysis ends up, but is far from the whole story. One of the key assumptions in the above analysis is that the error is random and uncorrelated from one $x$ value to another. This is rarely the case.

Consider finite difference and lattice (aka “tree”) approaches to option valuation. In these, the pricing function is a weighed average of the payoff sampled at various points. The weights change slightly as a function of the underlying, but the actual payoffs used change substantially as the strike passes a sample point. This makes the pricing function calculation roughly a piecewise linear approximation of the actual function. In the case of a European option on a stock under Black-Scholes and using a binomial lattice, it’s exactly a piecewise linear approximation. In the case of an option on a bond, it’s closer to piecewise exponential.

To prove this, let $S_{ij}$ be the stock value at node $j$ at time $t_i$. With starting value $S_0$, volatility $\sigma$, maturity time $T$, $N$ steps, $\Delta t = T/N$, and risk free rate $r$, a typical binomial lattice uses future stock values of:

$$S_{ij} = S_0 u^i d^{N-j},$$

where $u = \sigma \sqrt{\Delta t}$, and $d = 1/u$. 
The value of an option of strike $K$ is then
\[ C = e^{-rT} \sum_{j=0}^{N} \max(S_{Nj} - K, 0) \]
\[ = e^{-rT} \sum_{j=0}^{N} \max(S_0(pu)^j(qd)^{N-j} - K, 0) \]
\[ = e^{-rT} \sum_{j=j(S_0)}^{N} S_0(pu)^j(qd)^{N-j} - K \]
where $p = \frac{e^{\alpha_T} - d}{u - d}$, and $q = 1 - p$, and $j(S_0)$ is the minimum $j$ such that $S_{Nj} > K$.

Then,
\[ \frac{dC}{dS_0} = e^{-rT} \sum_{j=j(S_0)}^{N} (pu)^j(qd)^{N-j} \]

The derivative is a step function, only changing value when $j(S_0)$ changes.

But it becomes clearly evident when we inspect the difference derivative:
\[ \frac{dC}{dS}, \text{1 yr opt, 30\% vol, 3\% risk free rate, } h=0.01 \]

This is rarely noticed when graphing the function, just like the error in the derivative calculations weren’t noticed in our initial graphs:

Why only a factor of three? Because $u = \sigma \sqrt{\Delta t}$. Decreasing the step size by a factor of 10 only decreases $u$ by a factor of $\sqrt{10} \approx 3$, which only gives about 3 times the level density.

When the approximation is piecewise linear, and the stepsize is much smaller than the support of the linear segments, the first derivative is poor. In computing the second derivative, the sample endpoints almost always land on the same segment, making the estimate of the second derivative zero almost everywhere.

A 12 step lattice gives us large piecewise linear sections. A 120 step lattice, while increasing computation by a factor of 100, only decreases the sizes of the steps by about a factor of 3.
9. Error analysis 201 - Correlated errors

10. Smoothing

When the calculation is a black box, we can’t get inside to use the internals in the calculation. In this case, how can one compute a good derivative?

One trick is to use a large $h$. We suffer convexity error because it’s being swamped by error from the piecewise linearity of the function. Picking $h$ around 1 to 2x the support of the linear segments will do it.

11. $H$ adjustment

Here we can see that with a 12 step lattice, we need to compute the derivative with $h_0 \approx 17$.

The second derivative is also helped by using a larger stepsize, but still isn’t especially good:
More commonly, people would use 120 levels for a 1 year stock option, but even this requires a large value of $h_0$:

For fun, let’s take a look at what happens with 1200 levels, which is over 3 levels/day:

As you can see, we still have fairly large piecewise linear sections. We need to make $h_0$ around 2 to get reasonable derivative estimates.
11. **H adjustment**

Why did an $h_0$ of 17 for 12 levels, 5 for 120 levels and 2 for 1200 levels work reasonably well?

As mentioned before, the stepsize needed is roughly the lattice spacing. This is approximately $2S\sigma/\sqrt{T}$, which is 17 for 12 steps/year, 5.5 for 120 steps/year, and 1.7 for 1200 steps/year.

Even for a dense lattice of 1200 levels, a much larger stepsize required than is commonly recognized.

In fact, it’s common to use monthly steps in a binomial lattice for long dated bonds, and a bump size of 10bp for modified duration and key rate duration with a one sided derivative.

Let’s take a look at the behavior of this. We’ll use a trinomial lattice, which gives better results than a binomial lattice.

**11. H adjustment**

First, consider the error in computing the change in a callable bond as a function of the step size using a centered derivative.

![Centered derivatives](image)

A 25bp shift is bumpy, but looks fairly close to what it should be.

Next, consider the key rate sensitivities. The bond isn’t sensitive to the 3mo rate, so the 1st key rate sensitivity should be zero.

![Key rate duration - k1 - call sensitivity vs level](image)

The stepping in the one sided derivative for a given $h_0$ is the same as that for the centered derivative at $h_0/2$, but the convexity error is much worse.

It ends up being close to zero, but noisy, and pretty similar for all step sizes and seemingly unaffected by whether we use a centered derivative or a one sided derivative.
The second key rate sensitivities suffer from the piecewise nature of the calculation, both the centered ones as well as the one sided ones.

Comparing the sum of the one sided key rates to the 25bp centered derivatives, we see that the sum suffers both from the piecewise nature as well as the convexity, and suffers worse than the one sided full sensitivity.

The other key rates look similar.

Comparing the 50bp centered key rates to the 50bp difference derivative, we see that the two are close, but are significantly different. This is because the key rates interact with the piecewise nature differently than the full curve shift.
12. Filtering

A more sophisticated approach is to smooth our pricing function. Essentially, we’d like to filter out the high frequencies that come from the corners where the slope changes, leaving only the lower frequency data arising from the changing function values.

This amounts to computing the Fourier transform of the price function, multiplying by a function that decays to zero (to dampen out the high frequency noise), and transforming back, or

\[
\text{Smooth } f = \mathcal{F}^{-1}(\mathcal{F}(f)D)
\]

where \(D\) is our damping function (or smoothing kernel), and \(\mathcal{F}\) is the Fourier transform.

Gaussian quadrature.

\[
\frac{1}{\sigma \sqrt{2\pi}} \int f(x_0 - x) \left(e^{-\frac{x^2}{2\sigma^2}}\right) \, dx = \frac{-1}{\sigma^3 \sqrt{2\pi}} \int f(x_0 - x)xe^{-\frac{x^2}{2\sigma^2}} \, dx
\]

\[
= \frac{-1}{\sigma \sqrt{2\pi}} \int f(x_0 - \sigma \sqrt{2}x)xe^{-x^2} \, dx
\]

\[
\approx \sum \frac{-w_i}{\sigma \sqrt{2\pi}} f(x_0 - \sigma \sqrt{2}x_i)x_i
\]

where \(x_i\) are the Gaussian quadrature points and \(w_i\) are the associated weights.

The theory sounds beautiful, and looks like exactly what we need, but the theory doesn’t live up to it’s promise in practice. Although I’ve used this method in the past, and it has applications in signal processing, I’ve been unable to make it perform better than a two point difference derivative. It seems that it works better in the random noise case than on piecewise linear functions.

Since \(\mathcal{F}(f \ast g) = \mathcal{F}(f)\mathcal{F}(g)\) (where \(\ast\) is the convolution operator),

\[
\text{Smooth } f = \mathcal{F}^{-1}(\mathcal{F}(f)D) = \mathcal{F}^{-1}(\mathcal{F}(f)\mathcal{F}(\mathcal{F}^{-1}(D))) = \mathcal{F}^{-1}(\mathcal{F}(f \ast \mathcal{F}^{-1}(D))) = f \ast \mathcal{F}^{-1}(D)
\]

So, smoothing a function is the same as computing its convolution with the inverse transform of the smoothing kernel. Since \((f \ast g)' = f' \ast g = f \ast g'\), smoothing the derivative can be done by convolving with the derivative of the inverse transform of the smoothing kernel. Finally, since the Fourier transform of a Gaussian PDF is a Gaussian (up to scaling), we can smooth by integrating against a Gaussian and its derivatives.

All that’s left is to integrate a function times a Gaussian, which is best done by

Using 5 points on a 120 level Black-Scholes lattice yields:

The 5 point FFT method yields similar results to the 2 point difference derivative. Both look good by inspection.
But of course, the best way to check is to compare to a good reference. In this case, we’ll compare the error relative to a difference derivative computed on the formula using a step size of $10^{-5}$.

![Image](FFT vs centered difference - 1st derivative, errors)

The FFT method is hard pressed to do better than a well chosen step size.

Error graphs confirm what we saw in the previous graph:

![Image](FFT vs centered difference - 2nd derivative, errors)

It’s hard to make either method produce both a smooth and accurate second derivative:

![Image](FFT vs centered difference - 2nd derivative)

Both look equally poor, with the FFT method requiring twice the computational effort.

13. Complex arithmetic

Another approach makes use of complex analysis. If $f$ is a real valued function of one real variable, and can be extended to a complex analytic function, then

$$f(x + ih) = f(x) + f'(x)ih - \frac{f''(x)h^2}{2} - \frac{f'''(x)ih^3}{3!} + \frac{f^{(4)}(x)h^4}{4!} + \cdots$$

so

$$\frac{\Im(f(x + ih))}{h} = f'(x) - \frac{f'''(x)h^2}{3!} + \cdots$$

This has the same convexity error as the centered derivative, but doesn’t directly suffer from cancellation error, allowing one to reduce $h$ to lower convexity error without increasing cancellation error.

While this approach can be useful in analytic methods, difficulties are encountered when trying to apply it in finance. It doesn’t correct for correlated errors — when the function is piecewise linear, it just does a very good job of returning the slope of the linear sections, yielding a step function for the derivative. It’s also not as straightforward as it looks. One can’t just change all references
of double to complex because numerical code in finance makes heavy use of inequalities, as in

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

which are meaningless on the complex plane. They need to be replaced by something else. One source recommends comparing the real parts, but this prevents the function from being analytic, thus breaking the above Taylor series analysis. Finally, our analytic formulas in finance typically involve cumulative normal distributions. While there is a unique continuation to the complex plane, computing it is more involved than just calculating $\text{erf}(x/\sqrt{2})/2 + 1/2$. One would need to develop fast and accurate numerical methods for the calculation of a complex cumulative normal before this method is useful in such a context.

This method is commonly compared to a one sided derivative because both require one additional function evaluation. But, evaluating a function at a complex point can triple the computational effort. One complex addition is over double the effort of a real addition, in that it requires two real additions and works with more memory. One complex multiplication requires four real multiplications plus two real additions, and thus is over four times as expensive as a real addition.

A centered derivative is more comparable in computational effort, in which case both methods have the same convergence properties as $h_0$ tends to zero. The only difference is in the cancellation error.

It’s easy to see why this doesn’t help for Black-Scholes binomial lattices. Recalling that the lattice computation for the value of a call option is

$$C(S_0) = e^{-rT} \sum_{j=N}^{0} S_0 (pu)^j (q)N-j - K$$

we see that

$$\Im(C(S_0 + ih))/h = e^{-rT} \sum_{j=N}^{0} (pu)^j (q)N-j - K$$

Up to round off error, the complex method gives the same results as the centered difference.
15. Using internal lattice spacing

In finite difference approaches, one can often read extra information from the lattice itself. In a simple Black-Scholes lattice, one can start the lattice two levels early. This gives the option value as the middle value after the second step. The values at the other two nodes can be used for the up and down values.

One reference for this method is a 1994 article by Pelsser and Vorst, where they call it “a well known alternative” to the difference derivative.

Pelsser and Vorst compute the derivative as \( \frac{\Delta f}{\Delta x} \), which introduces convexity error by doing a difference derivative around the wrong point. Here we avoid this by using another numerical technique — fitting all three points (the up, the down and the center) to a quadratic and reading the derivatives from there.

This latter technique could also be used in general when three points are available, and should reduce convexity error, but I haven’t tested it.

Shifting the lattice often gives the best derivatives that can be gotten from a lattice.

15. Using internal lattice spacing

None the less, where this method applies, it works quite well. We’ll compare it to the best of fixed \( h_0 \) selection. Consider Black-Scholes again with a 1 year option, 30% vol, 3% risk free rate, computed using a 120 step binomial lattice. Again, the first derivatives are visually fine:

![Centered difference vs lattice shift - 1st derivative](image)

But the differences to a reference show that the shifted lattice approach is far smoother and more accurate:

![Centered difference vs lattice shift - 1st derivative, errors](image)

Unfortunately, the approach can’t always be applied. In interest rate lattices, the values at the other nodes don’t always correspond to a shift of the yield curve. In normal short rate models they do, but in log normal models they don’t. In the latter case, to apply this approach, one would have to either adjust the derivative or settle for differentiating with respect to a different sort of curve move.
The results on the second derivative are more pronounced. The fixed $h$ selection is visibly poor, while the shifted lattice still looks quite good:

Checking the differences to a reference shows how much better:

Surprisingly, this method yields reasonable results even with a monthly lattice. Here’s the first derivative:
Here’s the second derivative:

![Graph showing centered difference vs lattice shift - 2nd derivative]

The shifted lattice performs well because it samples the price function at exactly the right points. At option expiration, on the up tree there’s exactly one more node in the money and one less out of the money, and the rest get exactly the same value.

When the call option price is:

$$C(S_0) = e^{-rT} \sum_{j=j(S_0)}^{N} S_0(pu)^j(qd)^{N-j} - K$$

and $1 \leq j(S_0) \leq N - 1$, the up price is:

$$C(S_0u) = e^{-rT} \sum_{j=j(S_0)-1}^{N} S_0u(pu)^j(qd)^{N-j} - K$$

and the down price is:

$$C(S_0d) = e^{-rT} \sum_{j=j(S_0)+1}^{N} S_0d(pu)^j(qd)^{N-j} - K$$
16. Differentiation under the integral sign

In Monte Carlo calculations, one computes an integral via random sampling of
the payoff. Pricing errors in Monte Carlo based calculations are typically much
larger than in other methods, making shifting methods particularly poor.

One approach (advocated by Vladimir Piterbarg) is to exploit the fact that
the integral and derivative commute to integrate the derivative of the payoff
function instead of differentiating the integral of the payoff. This approach may
lead to staircasing, but even so, it’s still better than the random noise observed
in attempting a direct finite difference calculation.

Another approach (advocated by Fournié, Lasry, Lebuchoux, Lions and Touzi,
as well as by Benhamou) applies Malliavin calculus in an effort to reduce the
error in computing the expectation of the derivative. Here, instead of computing
\( \frac{d}{dS_0} E[X(S_0)] \), we find a random variable \( \pi \) such that \( \frac{d}{dS_0} E[X(S_0)] = E[\pi X(S_0)] \),
which again allows computing the derivative directly by Monte Carlo instead of
taking the difference of two Monte Carlo price calculations.

Differentiation under the integral can also be used when valuing options via FFT.

17. Analytic techniques

17. Analytic techniques

There’s a large literature on working out various greeks analytically which I
haven’t reviewed. Because of the pricing PDE, there are relations that can be
exploited to avoid the need to compute all the greeks — some can be gotten from
others. Symmetries and in general, behavior under specific transformations can
be exploited as well. Papers by Peter Carr as well as by Oliver Reiss and Uwe
Wystup are good places to get started.

18. Summary

18. Summary

- Approximating the derivative by a difference magnifies the error of the original
  function.
- Small step sizes give huge errors due to cancellation error.
- Large step sizes give huge errors due to convexity error.
- Balancing convexity error and cancellation error requires unexpectedly large
  step sizes — as large as \( 10^{-5} \) when calculations are accurate to machine
  precision.
- It’s hard to judge accuracy without an an accurate reference, but one can
  try to make due by graphing higher order derivatives with small stepsize.
- Finite difference methods produce piecewise linear (or exponential) functions,
  which require extra care. Large step sizes are needed to produce reasonable results. We observed the need for step sizes of 17 for a 12 level
  binomial lattice, and 25 – 50 bp for a 12 level trinomial lattice. Hedges in
  practice could be way off.
- Fixing this by increasing lattice density is computationally infeasible because
  level spacing is proportional to \( \sqrt{\Delta t} \).
19. Summary

- Beware of key rate durations. They’re especially inaccurate.
- Beware of one sided derivatives. They’re more sensitive to piecewise linear functions and more sensitive to convexity – the worst of both worlds.
- Other methods appear in the literature, but don’t always help.
- One simple method that does help is using the points in the lattice for the up and down values, extending the lattice back in time if necessary to get those points.

19. References


Risk Sensitivities of Bermuda Swaptions, Vladimir Piterbarg, Bank of America Working Paper, November 1, 2002


19. References

What Every Computer Scientist Should Know About Floating-Point Arithmetic, David Goldberg, Computing Surveys, March 1991
http://docs.sun.com/source/806-3568/ncg_goldberg.html

Numerical Recipes in C/C++/Fortran, William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery


The Complex-Step Derivative Approximation (Sensitivity Analysis Workshop, Livermore, August 2001)