Fun with Finite Difference

December 2022

Jesper Andreasen
Saxo Bank, Copenhagen

kwant.daddy@saxobank.com

Abstract

We summarize most of what you need to know about finite difference solution for one-dimensional partial differential equations in finance. We demonstrate the theoretical results by numerical experiments. The reader is invited and encouraged to replicate the experiments using the C++ code and spreadsheets in our GitHub repository.

Introduction

When this is in print, Brian Huge, Frederik Kryger-Baggesen and I have been teaching a finite difference course at Copenhagen University over the past semester. I have done a finite difference course before at KU, see Andreasen (2011). Back then the course was a very intense and compressed 3 day theoretical course. This time we're covering the same material, pretty much, but the emphasis is to make it more practical and hands-on. The philosophy is that you can't learn breaststroke from a book. At some stage you'll need to get wet. And the earlier, the better.

Same thing with finite differences. So our course is designed around coding yourself in C++ with an Excel add-in. Step by step. We fill in a few gaps for the students so we don't get stuck too much. The idea is to learn from experiments rather than from mathematical derivation.

This article attempts to pack everything from this KU course into a summary and the reader in encouraged to redo all our experiments using the spreadsheet and the code from our GitHub repository: *https://github.com/brnohu/compfin*.

Our course doesn't cover everything. We only do one-dimensional finite difference. Higher dimensional problems are not considered, but a lot of the results and considerations carry over. Also, we only cover the standard theta scheme and thereby the standard methods: explicit, implicit and Crank-Nicolson.

One More Time for Prince Knud

Before we get stuck in with code and numbers, we'll quickly derive the theta scheme for numerical solution of one dimensional partial differential equations.

Consider a state variable x that evolves according to

$$dx = \mu(t,x)dt + \sigma(t,x)dW \tag{1}$$

where W is a Brownian motion under the risk neutral measure. Assume that the interest rate is a function of (t,x) only, i.e. $r = r(t,x)$.

Using Ito's lemma it is straightforward to show that the expectation

$$f(t,x(t)) = E_t[e^{-\int_t^T r(u)du} f(T,x(T))] \tag{2}$$

is the solution to the backward partial differential equation

$$0 = \partial_t f + Af \quad , A = -r + \mu\partial_x + \frac{1}{2}\sigma^2\partial_{xx} \tag{3}$$

with $f(T,x)$ as the terminal boundary condition.

The partial differential equation (3) is turned into a finite difference equation in two steps.

First, we discretise the space $x_0 < x_1 < \ldots < x_{n-1}$ and introduce the first order spatial difference operators

$$\begin{aligned}
\delta_x^+ f(x_i) &= \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \\
\delta_x^- f(x_i) &= \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \\
\delta_x f(x_i) &= \frac{x_{i+1} - x_i}{x_{i+1} - x_{i-1}} \delta_x^- f(x_i) + \frac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}} \delta_x^+ f(x_i)
\end{aligned} \tag{4}$$

and second order spatial operator

$$\delta_{xx} f(x_i) = 2 \frac{\delta_x^+ f(x_i) - \delta_x^- f(x_i)}{x_{i+1} - x_{i-1}} \tag{5}$$

Taylor expansion can be used to derive the order of the accuracy in (4) and (5).

These operators can be represented as rows of a tridiagonal matrix. Specifically,

$$\begin{aligned}
(\delta_x^+)_{i\cdot} &= \begin{bmatrix} 0 & \frac{-1}{x_{i+1} - x_i} & \frac{1}{x_{i+1} - x_i} \end{bmatrix} \qquad ,0 \le i < n-1 \\
(\delta_x^-)_{i\cdot} &= \begin{bmatrix} \frac{-1}{x_i - x_{i-1}} & \frac{1}{x_i - x_{i-1}} & 0 \end{bmatrix} \qquad ,0 < i \le n-1 \\
(\delta_x^-)_{0\cdot} &= (\delta_x^+)_{n-1,\cdot} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}
\end{aligned} \tag{6}$$

and

$$\begin{aligned}
(\delta_x)_{i\cdot} &= \frac{1}{x_{i+1} - x_{i-1}} \begin{bmatrix} -\frac{x_{i+1} - x_i}{x_i - x_{i-1}} & \frac{x_{i+1} - x_i}{x_i - x_{i-1}} - \frac{x_i - x_{i-1}}{x_{i+1} - x_i} & \frac{x_i - x_{i-1}}{x_{i+1} - x_i} \end{bmatrix} \qquad ,0 < i < n-1 \\
(\delta_x)_{0\cdot} &= (\delta_x^+)_{0\cdot} \quad ,(\delta_x)_{n-1,\cdot} = (\delta_x^-)_{n-1,\cdot}
\end{aligned} \tag{7}$$

and finally

$$\begin{aligned}
(\delta_{xx})_{i\cdot} &= \frac{2}{x_{i+1} - x_{i-1}} \cdot \begin{bmatrix} \frac{1}{x_i - x_{i-1}} & (\frac{-1}{x_i - x_{i-1}} + \frac{-1}{x_{i+1} - x_i}) & \frac{1}{x_{i+1} - x_i} \end{bmatrix} \qquad ,0 < i < n-1 \\
(\delta_{xx})_{0\cdot} &= (\delta_{xx})_{n-1,\cdot} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}
\end{aligned} \tag{8}$$

In the above we use a compact notation for tridiagonal matrices where we only list the three elements around the diagonal, in line with the band diagonal matrix representation used in Press et al (1988).

Depending on the application we will use either central differencing

$$\bar{A} = -r + \mu\delta_x + \frac{1}{2}\sigma^2\delta_{xx} \tag{9}$$

or dynamic up- and down-winding

$$\bar{A} = -r + \mu^+\delta_x^+ + \mu^-\delta_x^- + \frac{1}{2}\sigma^2\delta_{xx} \tag{10}$$

If we let $f(t) = (f(t,x_0),\ldots, f(t,x_{n-1}))'$, the PDE (3) can now be replaced by a matrix equation

$$0 = \partial_t f + \bar{A}f \tag{11}$$

Over a discrete time step the solution to (11) is

$$f(t_h) = e^{\Delta t\bar{A}} f(t_{h+1}) = (\sum_{k=0}^{\infty} \frac{(\Delta t\bar{A})^k}{k!}) f(t_{h+1}) \qquad ,\Delta t = t_{h+1} - t_h \tag{12}$$

Discretising (12) now leads to the theta scheme

$$\begin{aligned} f(t_{h+1/2}) &= [I + (1-\theta)\Delta t\bar{A}] f(t_{h+1}) \\ [I - \theta\Delta t\bar{A}] f(t_h) &= f(t_{h+1/2}) \end{aligned} \tag{13}$$

where I denotes the identity. The idea is now to solve (13) repeatedly

$$t_h \leftarrow t_{h+1/2} \leftarrow t_{h+1}$$

until we obtain the solution at time t_0. We will generally let m denote the number of full time steps, and let $0 = t_0 < t_1 < \ldots < t_m$.

We generally let $\theta = 0$ denote the *explicit* scheme, $\theta = 1$ the (fully) *implicit* scheme and $\theta = 1/2$ the *Crank-Nicolson* scheme.

And now the other way around: Rearrange (13) and multiply it by the vectors $p(t_h), p(t_{h+1/2})$ and we get

$$\begin{aligned} p(t_{h+1/2})' f(t_{h+1/2}) &= p(t_{h+1/2})'[I + (1-\theta)\Delta t\bar{A}] f(t_{h+1}) \\ p(t_h)' f(t_h) &= p(t_h)'[I - \theta\Delta t\bar{A}]^{-1} f(t_{h+1/2}) \end{aligned} \tag{14}$$

If $p(t_h)$ is defined as the solution to the system

$$\begin{aligned}[I-\theta\Delta t\bar{A}]'\,p(t_{h+1/2}) &= p(t_h)\\ p(t_{h+1}) &= [I+(1-\theta)\Delta t\bar{A}]'\,p(t_{h+1/2})\end{aligned} \tag{15}$$

then

$$p(t_h)'\,f(t_h) = p(t_{h+1/2})'\,f(t_{h+1/2}) = p(t_{h+1})'\,f(t_{h+1}) \tag{16}$$

Hence, (15) defines a *forward* scheme for the discrete Green's function for the backward linear system (13). The scheme is solved by stepping forward in time

$$t_h \to t_{h+1/2} \to t_{h+1}$$

typically with the initial boundary condition $p(t_0, x_i) = 1_{x_i = x(t_0)}$.

Traditionally and conventionally, finite difference is used for going from parameters to option prices. But in finance, we often want to do the reverse, i.e. take offset in option prices and deduce the parameters consistent with the observed option prices. In our case, European call options are given by

$$c(t_h, x_i) = \sum_{j=0}^{n-1} (x_j - x_i)\, p(t_h, x_j) \tag{17}$$

For the case of $\mu = 0$, (15) is consistent with the following *forward* equation for call option prices

$$\begin{aligned}[I-\theta\Delta t\bar{A}]c(t_{h+1/2}) &= c(t_h)\\ c(t_{h+1}) &= [I+(1-\theta)\Delta t\bar{A}]c(t_{h+1/2})\end{aligned} \tag{18}$$

with the boundary condition $c(t_0, x_i) = (x(t_0) - x_i)^+$. Equation (18) can be used for directly generating initial call option prices for all expiries and strikes. But it can also be used for inferring a surface of local volatility $\{\sigma(t_h, x_i)\}$ that is consistent with the observed option prices and the finite difference scheme that we are using. As such, equation (18) is a discrete Dupire equation.

It is important to note that the equations (13), (15) and (18) are *discretely* consistent with each other. That is, if you stuff in European option prices into (18) and find the corresponding volatility surface then the scheme (13) will reproduce these option prices *exactly*.

Coding a Theta Solver

Coding a theta solver requires surprisingly little code. Actually, in terms of lines of code, it is almost as compact as code for the Black-Scholes formula and associated implied volatility calculator.

Consulting our GitHub repository you'll see that we use the following components to implement our theta solver:

1. kFiniteDifference::dx() and ::dxx(): construct the first and second order difference operators in compact matrix form (6-8).

2. kFd1d::calcAx(): constructs the matrix $I + \Delta t\bar{A}$ used in (9-10).

3. kMatrixAlgebra::banmul(): band diagonal matrix-vector multiplication (13a).

4. kMatrixAlgebra::tridag(): tridiagonal matrix-vector solution (13b).

5. kFd1d::rollBwd(): implements the backward roll by calling the above routines.

6. kMatrixAlgebra::transpose(): transposes a band diagonal matrix.

7. kFd1d::rollFwd(): implements the forward roll (15) by calling the above routines.

The routines banmul() and tridag() are pretty much straight copies of standard routines from Numerical Recipes by Press et al (1988).

Our code is compact and follows the mathematical matrix notation used in the previous section. No festival of little ants like in Figure 1. It's very well-shaped, IMUHO[1].

The first thing to check when you implement the theta solver is to reconcile the forward and backward solutions. The sheet 'duality' in the workbook 'fun with fd.xls' shows that this is indeed the case. The values in the two green fields match.

How Wide the Grid?

The rule of thumb is approximately ± 5 standard deviations. Less than this will significantly reduce accuracy and more will just result in more exercise for the electrons in the computer without any benefit. The way to think about this rule is that finite difference solution is essentially a method of numerical integration, and as such you need to cover enough of the mass of the distribution but not necessarily more than that.

One modification to this rule is that for processes with significant mean-reversion, such as for example interest rate models. For these models, it appears to be a better rule to base the grid dimensioning on quadratic variation, i.e.

$$x_{n-1} - x(t_0) \approx x(t_0) - x_0 \approx q \cdot [\int_0^{t_n} \sigma(u, x(t_0))^2 \, du]^{1/2} \qquad (19)$$

with $q \approx 5$.

[1] In-My-Usual-Humble-Opinion.

Another, modification is for very fat tailed distributions such as for example when stochastic volatility is involved, it appears that $q \approx 10$ is a better number.

In Table 2 we show the implied volatility of an option as function of the width of the grid. The sheet 'grid width' has been used to generate these numbers. Note that as the grid is narrowed, probability mass builds up on the grid boundaries.

Wot PDE?

In many cases we transform the PDE to resemble the heat equation. You can for example choose to transform the constant parameter Black-Scholes PDE

$$0 = f_t - rf + \mu S f_S + \frac{1}{2}\sigma^2 S^2 f_{SS} \tag{20}$$

to

$$0 = f_t + \frac{1}{2} f_{xx} \qquad , S = S(0) e^{\mu t - \frac{1}{2}\sigma^2 t + \sigma x} \tag{21}$$

Why would you do so? If you don't, you can end up using a disproportional number of grid points for very extreme levels and almost none around where they actually matter. The problem increases with maturity of the finite difference grid and it is illustrated in Figure 3.

Transforming the PDE as in (18-19) increases the formal accuracy but it can have drawbacks, such as for example making it more difficult to control the martingale property of the spot. We discuss this in Andreasen (2022c). The alternative to transforming the PDE is to use a non-uniform grid spacing. For the case of (14-15), we could choose to solve the PDE on the grid points

$$S_i = S(0) e^{(i-n/2)\cdot \Delta x} \quad , i = 0, \ldots, n-1 \tag{22}$$

This will reduce the formal accuracy as the spacing will now be non-uniform. However, in practice, this is often a price worth paying.

Generally, the transforms for transforming the PDE into a unit diffusion PDE is the Lamperti transform

$$y(t) = \int_{x_0}^{x(t)} \sigma(t,x)^{-1} dx \tag{23}$$

The inverse mapping $y \mapsto x$ can be used for grid spacing.

Grid Point Placement

… is somewhat more tricky than one would think. Here we describe some of the approaches that can be used.

If the strike lands on the grid point the finite difference solution will tend to undervalue the option by a quantity of order $O(\Delta x^2)$. The bias disappears if the strike is exactly between two grid points. For digital options the bias is $O(\Delta x)$. This is not just an issue for the accuracy of the solution. Specifically, if the grid isn't aligned to the strikes there will always be a noise of at least $O(\Delta x^2)$ in the system and this will particularly affect the Greeks and our ability to accurately explain the P&L.

So ideally, grid points should be placed so strike points are symmetrically surrounded by grid points.

An alternative is to smooth the payoff, i.e. to replace the payoff at level x_i by

$$(1-\lambda)\frac{2}{x_i - x_{i-1}}\int_{(x_{i-1}+x_i)/2}^{x_i} f(x)dx + \lambda\frac{2}{x_{i+1} - x_i}\int_{x_i}^{(x_i+x_{i+1})/2} f(x)dx \quad ,\lambda = \frac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}} \tag{24}$$

This is a quite effective methodology. Figure 4 is not a naughty drawing but actually an illustration of the effectiveness of smoothing the payoff. It shows the implied volatility of an option as function of the strike when the finite difference grid is kept fixed, for a non-smoothed and smoothed terminal payoff.

What about barriers? Barriers that are monitored at discrete times should be treated like digital options. I.e. either smooth the payoff or place the grid points symmetrically around the barrier. Continuously monitored barriers can be handled by placing the barrier level on the grid and enforcing absorption on the barrier by setting volatility and drift equal to zero on the barrier: $\mu = \sigma = 0$. Not doing so, will result in a dramatic accuracy loss of $O(\Delta t^{1/2})$.

What about the spot? Generally, we recommend placing the spot on the grid. If it isn't, the prices at the current time will have to be interpolated. Linear interpolation effectively corresponds to a one-step binomial tree, i.e. additional volatility will be injected into the solution. In the context of local volatility models we can correct for this but it may not always be possible. Higher order interpolation schemes like splines and polynomials will generally produce negative weights on specific points and this will in turn affect stability so we don't really like that route too much.

Stability

Conventionally, stability is analysed using von Neumann analysis. The idea is to consider eigen solutions to (13) of the form

$$g(t_h, x_i) = g^{-h}e^{\imath k x_i} \quad ,\imath = \sqrt{-1}, k \in \mathbb{R}, g \in \mathbb{C} \tag{25}$$

If all eigen solutions have the property that $|g| \leq 1$, then we say that the scheme is von Neumann stable. Strictly speaking, von Neumann stability is only sufficient to conclude that the scheme is convergent for the constant parameter case. However, there are, to my knowledge, no counter examples of it not being sufficient for non-constant parameters.

Von Neumann stability doesn't always imply that all transition probabilities are positive. But it means that, as the number of time steps is increased, the transition probabilities over multiple time steps converge to positive values, hence the numerical solution will ultimately converge to the continuous time and state solution as both time and spatial steps tend to zero $(\Delta t, \Delta x) \to (0,0)$.

For the explicit scheme, $\theta = 0$, von Neumann stability is ensured if and only if the matrix $[I + \Delta t\bar{A}]$ has only non-negative elements. For the case of $r = \mu = 0$, this requires

$$\Delta t \leq \frac{\Delta x^2}{\sigma^2} \tag{26}$$

For the case when (24) holds with equality, the explicit scheme corresponds to a binomial tree. In the specific example in the sheet 'explicit' this occurs when the number of time steps is exactly 25. You can vary the number of time steps to see what happens.

The implicit scheme, $\theta = 1$, is always von Neumann stable and thereby convergent. For the drift less case the transition probabilities are always positive, and for constant volatility σ, shaped like a Laplace density, i.e. $e^{-k|x|}$ for some positive constant k. When the number of time steps is increased then the density converges to a Gaussian shape. The sheet 'implicit 1' can be used to test this.

When the problem has a drift, positive transition probabilities are guaranteed only for the case when dynamic up- and down winding is used for the first derivative. I.e. if (10) is used rather than (9). The sheet 'implicit 2' implements this. If you recalculate the sheet, the forward scheme generates transition probabilities for new random vectors r, μ, σ. The produced probabilities are always positive, though not always pretty.

The Crank-Nicolson scheme, $\theta = 0.5$, is also always von Neumann stable and thereby convergent. But even for the zero drift case, the transition probabilities will be non-negative only if the condition (26) is satisfied for the first explicit half time step. The transition probabilities oscillate but converge to a smooth positive Gaussian form as you increase the number of time steps while keeping the maturity constant. The sheet 'cn 1' shows this. Randall and Tavella (2000) and Andreasen (2011) provide theoretical arguments that if the grid is dimensioned according to the ± 5 standard deviations rule, then the number of time steps should be roughly half of the number of spatial steps: $m \approx n/2$.

Moderate drift is generally not a practical problem for the convergence of any of the finite difference methods. But for problems with very strong drift relative to the diffusion the only bullet proof cure is really dynamic up- and down-winding and/or transforms to remove or reduce the drift.

Accuracy

Using (12) its straightforward to verify that the accuracy of each step in (13) is $O(\Delta t^2)$ for $\theta \neq 1/2$ and $O(\Delta t^3)$ for $\theta = 1/2$. As the solution of (13) is repeated $O(\Delta t^{-1})$ times, the accuracy of the total scheme is $O(\Delta t)$ for the explicit and implicit schemes and $O(\Delta t^2)$ for the Crank-Nicolson scheme.

Taylor expansion can be used to show that

$$\begin{aligned} \delta_x^+ f(x_i) &= f'(x_i) + O(\Delta x) \\ \delta_x^- f(x_i) &= f'(x_i) + O(\Delta x) \\ \delta_x f(x_i) &= f'(x_i) + O(\Delta x^2) \end{aligned} \qquad (27)$$

and

$$\delta_{xx} f(x_i) = f''(x_i) + \frac{1}{3} f'''(x_i)((x_{i+1} - x_i) - (x_i - x_{i-1})) + O(\Delta x^2) \qquad (28)$$

So in summary, the implicit and explicit schemes have accuracies of $O(\Delta t + \Delta x^2)$, whereas the Crank-Nicolson scheme has an accuracy of $O(\Delta t^2 + \Delta x^2)$. The accuracies drop to respectively $O(\Delta t + \Delta x)$ and $O(\Delta t^2 + \Delta x^2)$ if up- and down-winding is used.

Table 2 shows a typical convergence pattern in number of time steps. The explicit and implicit schemes converge roughly at the same speed but the convergence of the Crank-Nicolson scheme is much quicker. The implicit scheme converges monotonously whereas the Crank-Nicolson exhibits oscillatory convergence which is only visible for a low number of time steps. The explicit scheme is not stable until you get to a certain level of time steps. Before then it will return values close to the borders of the real axis.

Figure 5 shows a log-log plot of the error where the error is computed as the difference from implied volatility at m time steps minus implied volatility for $m = 100{,}000$, i.e.

$$\ln(|\, v(m) - v(m = 100{,}000)\,|)$$

versus $\ln m$. Remarkably the explicit and implicit schemes fall right on top of each other in this measurement. The figure demonstrates $O(\Delta t)$ convergence for the explicit and implicit schemes and $O(\Delta t^2)$ convergence for the Crank-Nicolson scheme, as expected. The remaining error(s) is due to inaccuracy in the spatial domain.

A well done 50x100 Crank-Nicolson grid, is pretty darn accurate. You'll need around 2,500 time steps to achieve the same accuracy with the explicit or the implicit method.

Table 3 and figure 6 repeat the convergence exercise but this time including the spatial domain. Here we keep the number of spatial steps equal to twice the number of time steps. We only consider the implicit and the Crank-Nicolson schemes as the explicit scheme is unstable for these grid dimensions.

Speed

Table 4 reports the CPU times per time step in for a 200 spatial step finite difference grid. CPU times are for a standard lap top. Updating parameters is expensive so it makes a considerable difference whether parameters are time dependent or not. Realistic size finite difference grids run quicker than 1ms on a lap top. For the explicit method we need to run 8 times as many time steps as the implicit and the Crank-Nicolson method to keep it stable.

Models

Generally, our recommendation is to use Crank-Nicolson for parametric models where convergence is a necessity or at least a desired property. Examples of such models include Black-Scholes with time dependent parameters and term structure models that rely on closed-form bond prices.

For the term structure models with closed-form bond prices that you make use of, you really do need the second order accuracy to hit these closed-form bond prices. Examples of these include: Gaussian models, CIR, and quadratic models.

Slightly off-topic because we're going to higher dimension: We've had success with using the second order accurate Craig-Sneyd scheme for the Heston models and a modified Craig-Sneyd scheme with 5-point winding for the second state variable in the one-factor Cheyette model. We have also used Craig-Sneyd for 3 dimensional problems including Cheyette with stochastic local volatility and cross currency models with stochastic interest rates.

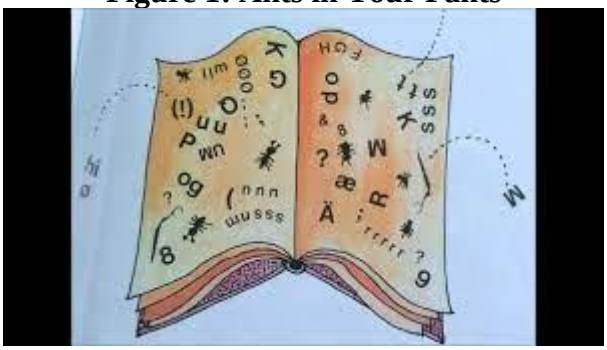
The fully implicit scheme, and particularly the 2-step variant described in Andreasen (2023), is our weapon of choice for local volatility models, and a variation of this for stochastic local volatility. For local volatility models the robustness and positive transitions probabilities matter more than convergence, because the local volatility models are constructed to exactly hit input option prices. We use local volatility not only for exotics but extensively for volatility surface generation and the margin model that we use at Saxo.

The explicit scheme, and particularly the binomial tree is only really useful for very simple models, i.e. Black-Scholes with constant volatility. Some market makers still use binomial trees for pricing the American element in exchange traded equity options. Its proponents argue that it is more flexible with respect to dividend structure than the faster and more accurate integral equation method of Andersen, Lake and Offengenden (2015).

Conclusion

We have compressed all our findings into the cheat sheet in Table 5. See if you can spot something we have missed and then please go have some fun with finite difference!

Figure 1: Ants in Your Pants



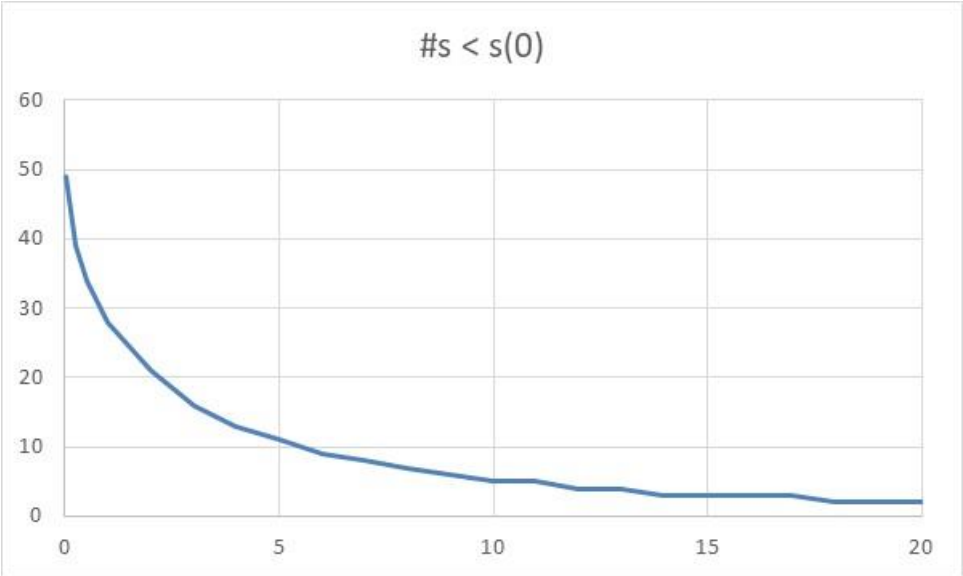
Finite difference code doesn't have to look like this.

Table 1: Option Price as Function of Grid Width

num std	imp vol
1	0.098001088456667
2	0.100039426492236
3	0.100056953428725
4	0.100056956186182
5	0.100056956190353
6	0.100056956190353
7	0.100056956190353
8	0.100056956190353
9	0.100056956190353
10	0.100056956190353

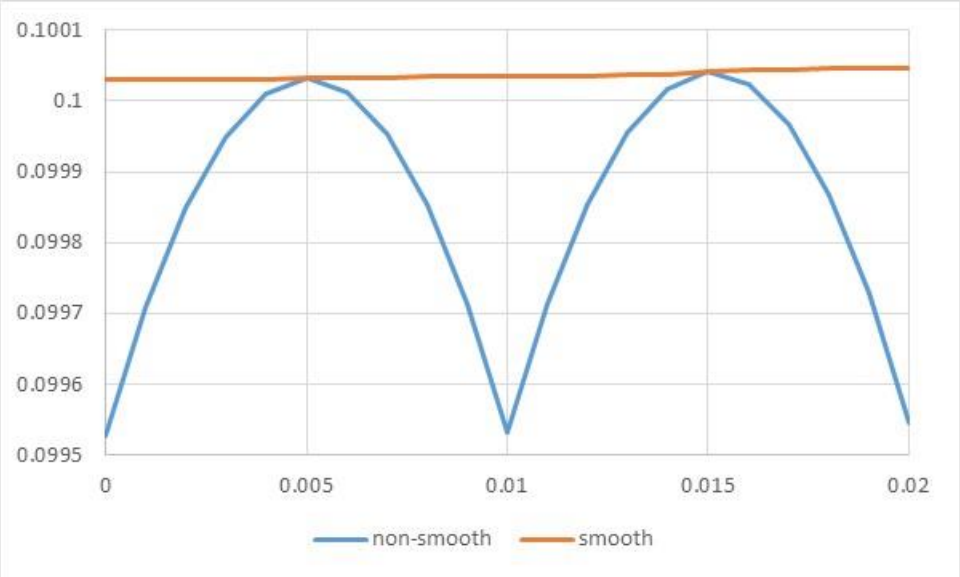
Finite difference solution for different grid width specified in number of standard deviations $q = 1,\ldots,10$ and $\Delta x = 0.02$ held constant. Other parameters are $x(0) = 0, \sigma = 0.1, \mu = r = 0, T = 0.25, k = -0.01, \#\{t_h\} = 10, \theta = 0.5$. See sheet 'grid width'.

Figure 2: Points Below $S(0)$ as Function of T



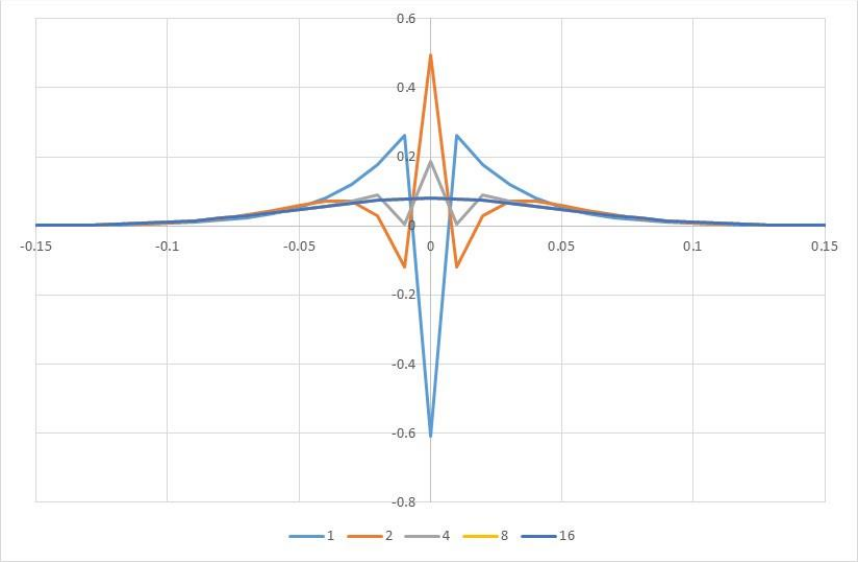
How many grid points out of total number of $n=100$ grid points is below $S(0)$ when we use constant spacing $\Delta S=(S_{n-1}-S_0)/n$ in a log-normal model with $S(0)=1,\sigma=0.2$, and a grid width of $S_0=e^{-5\sigma\sqrt{T}}, S_{n-1}=e^{5\sigma\sqrt{T}}$. See sheet 'transform'.

Figure 3: Implied Volatility As Function of Strike



Implied volatility of option price as function of strike for non-smoothed and smoothed option payoffs.
Parameters: $T = 0.25, x(0) = 0, r = 0, \mu = 0, \sigma = 0.1, n = 50, \#\{t_h\} = 10, \theta = 0.5$.
See sheets 'non-smooth' and 'smooth'.

Figure 4: Transition Probabilities in Crank-Nicolson



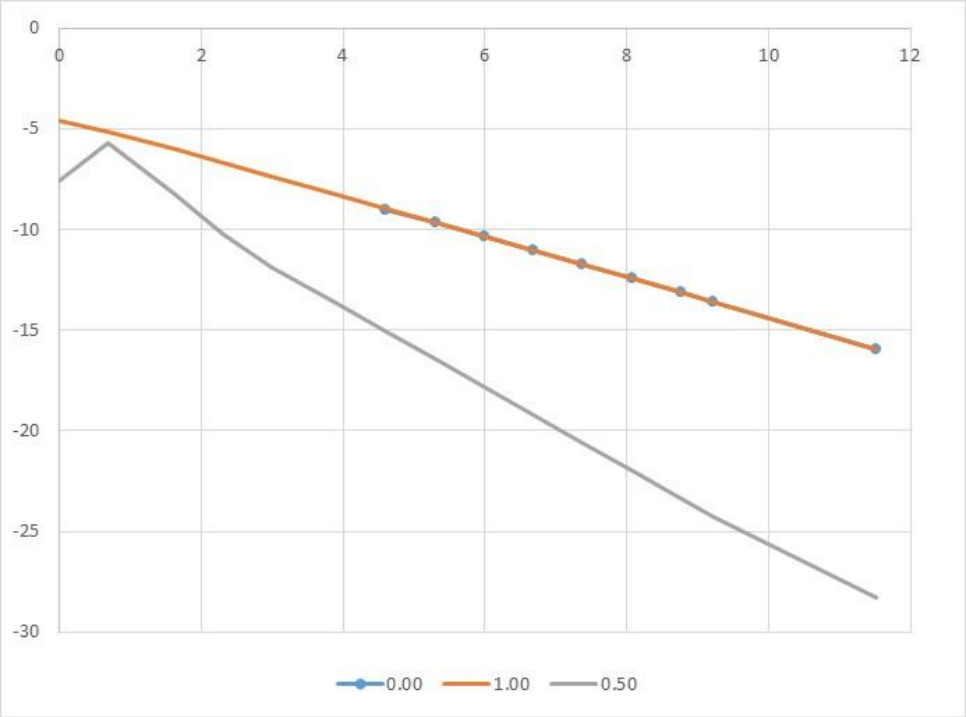
Transition probabilities in Crank-Nicolson for different number of time steps $m = 1,2,4,8,16$. Parameters: $T = 0.25, x(0) = 0, r = 0, \mu = 0, \sigma = 0.1, n = 50, \theta = 0.5$.
See sheet 'cn'.

Table 2: Implied Volatility as Function of Number of Time Steps

num t	explicit	implicit	Crank-Nicolson
1		0.0899255101685190	0.0994629647975475
2		0.0942391831261295	0.1033406443958640
5		0.0974974495350823	0.1002725557019060
10		0.0987121552908050	0.1000226016286290
20		0.0993421002329765	0.0999931289656106
50		0.0997267980327348	0.0999870876627954
100	0.1001063434741420	0.0998561040203962	0.0999862213345753
200	0.1000510401081950	0.0999209535149835	0.0999860047627463
400	0.1000184703491380	0.0999534269163274	0.0999859506204330
800	0.1000021974546890	0.0999696757212618	0.0999859370848974
1,600	0.0999940640112053	0.0999778031423667	0.0999859337010046
3,200	0.0999899980413299	0.0999818676066240	0.0999859328550338
6,400	0.0999879652444594	0.0999839000270940	0.0999859326435392
10,000	0.0999872334683153	0.0999846317292372	0.0999859326019388
100,000	0.0999860626604642	0.0999858024863355	0.0999859325738124
1,000,000	0.0999859455804971	0.0999859195642117	0.0999859325732724

Implied volatility of option solved with different solvers for different number of time steps. Parameters: $T = 1, k = 0.045, x(0) = 0, r = 0.03, \mu = -0.03, \sigma = 0.1, n = 100$.
See sheet ‘accuracy’.

Figure 5: Log-Log Plot of Convergence in Time



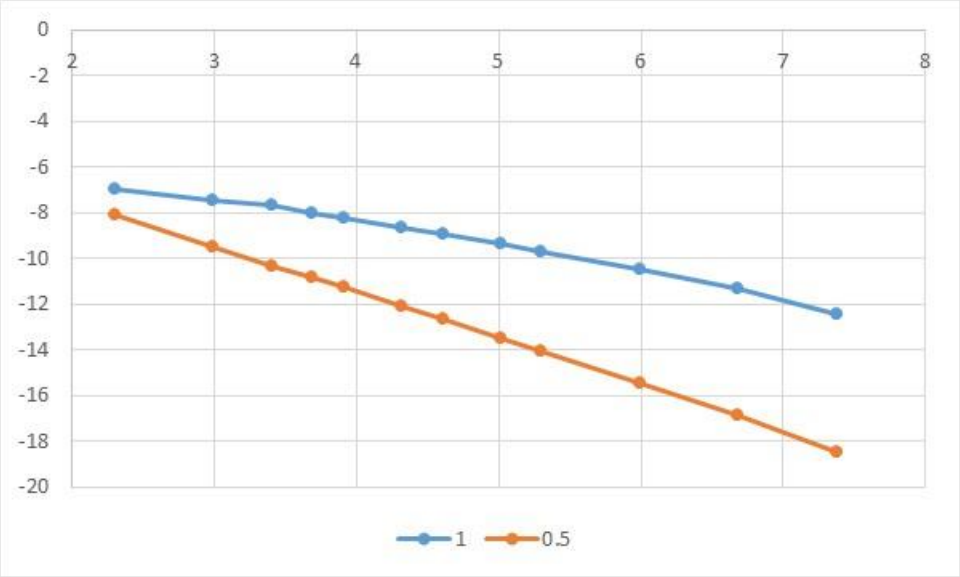
Log-log plot of error of implied volatility of an option solved with different finite difference solvers as function of number of time steps.
Parameters: $T = 1, k = 0.045, x(0) = 0, r = 0.03, \mu = -0.03, \sigma = 0.1, n = 100$.
See sheet 'accuracy'.

Table 3: Implied Volatility as Function of Grid Size

grid	implicit	crank nicolson
10x20	0.1009600862853950	0.0996994550115734
20x40	0.1005724138037610	0.0999266585632249
30x60	0.0995338608120574	0.0999668138357912
40x80	0.0996555249307086	0.0999806939278167
50x100	0.0997267980327348	0.0999870876627954
75x150	0.0998208336012766	0.0999944334583716
100x200	0.0998665531614816	0.0999967712562736
150x300	0.0999117487301578	0.0999985660745013
200x400	0.0999340798010779	0.0999991933073142
400x800	0.0999672421462753	0.0999997983576170
800x1600	0.0999836718244728	0.0999999495913732
1600x3200	0.0999918486245818	0.0999999873979747
3200x6400	0.0999959274940633	0.0999999968498630

Implied volatility of option solved with implicit and Crank-Nicolson solvers for different grid sizes. Payoff smoothing is used.
Parameters: $T = 1, k = 0.045, x(0) = 0, r = 0.03, \mu = -0.03, \sigma = 0.1, n = 100$.
See sheet 'accuracy 2'.

Figure 6: Log-Log Plot of Convergence in Time and State



Log-log plot of error of implied volatility of an option solved with implicit and Crank-Nicolson as function of number grid sizes. Payoff smoothing is used. Data in table 3 is used. Parameters: $T = 1, k = 0.045, x(0) = 0, r = 0.03, \mu = -0.03, \sigma = 0.1, n = 100$.
See sheet 'accuracy 2'.

Table 4: Speed per Time Step

	explicit	implicit	crank-nicolson
200 w/o update	9.10E-07	2.09E-06	3.00E-06
200 w/ update	3.11E-06	4.00E-06	6.80E-06
400, 100, 100 timesteps	1.25E-03	4.00E-04	6.80E-04

Finite difference grid calculation speed in seconds per time step for a grid of 200 spatial steps. The first row is for the case when parameters are constant in time, the second row for time dependent parameters, and the third row is total CPU time for 400, 100, 100 time steps in the three different methods, respectively.

Table 5: The Finite Difference Cheat Sheet

Schemes	Explicit $\theta = 0$	Implicit $\theta = 1$	Crank-Nicolson $\theta = 1/2$
PDE	$0 = f_t + Af \quad , A = -r + \mu\partial_x + \frac{1}{2}\sigma^2\partial_{xx}$		
FD central	$\bar{A} = -r + \mu\delta_x + \frac{1}{2}\sigma^2\delta_{xx}$		
FD winding	$\bar{A} = -r + \mu^+\delta_x^+ + \mu^-\delta_x^- + \frac{1}{2}\sigma^2\delta_{xx}$		
Boundaries	Absorption: $\mu = \sigma = 0$ or reflection: $\mu_0 > 0, \mu_{n-1} < 0, \sigma = 0$		
Backward	$f(t_h) = [I - \theta\Delta t\bar{A}]^{-1}[I + (1-\theta)\Delta t\bar{A}]\, f(t_{h+1})$		
Forward	$p(t_{h+1}) = [I + (1-\theta)\Delta t\bar{A}'][I - \theta\Delta t\bar{A}']^{-1}\, p(t_h)$		
Dupire	$c(t_{h+1}) = [I + (1-\theta)\Delta t\bar{A}][I - \theta\Delta t\bar{A}]^{-1} c(t_h) \quad , \mu = 0$		
Grid width	$\pm 5 \cdot (\int_0^T \sigma(u, x(0))^2\, du)^{1/2}$		
Transform	PDE or grid spacing: $y = \int_{x_0}^{x} \sigma(a)^{-1} da$		
Vanilla strikes	Mid between grid points or $O(\Delta x^2)$ error		
Digitals	Mid between grid points or $O(\Delta x)$ error		
Cont barriers	On grid and absorption or $O(\Delta t^{1/2})$ error		
Von Neumann	$\Delta t \le O(\Delta x^2)$	Always	Always
$p \ge 0, \mu = 0$	$\Delta t \le O(\Delta x^2)$	Always	$\Delta t \le O(\Delta x^2)$
$p \ge 0, \mu \ne 0$	$\Delta t \le O(\Delta x^2)$	With winding	$\Delta t \le O(\Delta x^2)$
Accuracy central	$O(\Delta t + \Delta x^2)$	$O(\Delta t + \Delta x^2)$	$O(\Delta t^2 + \Delta x^2)$
Accuracy winding	$O(\Delta t + \Delta x)$	$O(\Delta t + \Delta x)$	$O(\Delta t^2 + \Delta x)$
Time/spatial steps	4	0.5	0.5
Models	Brownian motion	Non-parametric	Parametric
CPU/time step	3e-6s	4e-6s	7e-6s

References

Andersen, L, M Lake, D Offengenden (2015): "High Performance American Option Pricing." *SSRN*.

Andreasen, J (2011): "Finite Difference Methods for Financial Problems." *Copenhagen University*.

Andreasen, J (2023): "Catch Up." *Wilmott* January.

Andreasen, J, B Huge and F Kryger-Baggesen (2022): *https://github.com/brnohu/compfin*.

Press, W, S Teukolsky, S Vetterling, and B Flannery (1988): *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press.

Randall, D and C Randall (2000): *Pricing Financial Instruments: The Finite Difference Method.* Wiley.