Mathematical & Computational Finance II
Lecture Notes

PDE Methods

November 12 2015
Last update: December 4, 2017

1 FTCS

1.1 FTCS with Matrix Algebra

Last time we spoke about the FTCS method for approximating the heat equation. The FTCS algorithm can be written in matrix form as

$$\vec{U}^{n+1} = \mathbf{F}\vec{U}^n + \vec{p}^n \quad 0 \leq n \leq N-1$$

where $\mathbf{F}$ is some $(J-1)\times(J-1)$ matrix and $\vec{p}^n$ is some $(J-1)\times 1$ row vector, starting with initial condition $\vec{U}^0$ in the time coordinate $u(x,0)$. Recall that the FTCS algorithm is determined by discretizing

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \quad \text{at } j=0,...,J;\ n=0,...,N;\ x_j = x_L + j\Delta x;\ t_n = n\Delta t$$

to obtain

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} = \frac{U_{j+1}^n - 2u_j^n + U_{j-1}^n}{(\Delta x)^2}$$

or equivalently

$$U_j^{n+1} = \nu U_{j+1}^n + (1-2\nu)U_j^n + \nu U_{j-1}^n$$

where $\nu = \frac{\Delta t}{(\Delta x)^2}$, and boundary conditions

$$\begin{aligned} U_j^0 &= \lambda(x) \quad 0 \leq j < J \\ U_0^n &= \alpha(n\Delta t) \quad 0 < n \leq N \\ U_J^n &= \beta(n\Delta t) \quad 0 < n \leq N \end{aligned}$$

For $0 < n \leq N$ and $2 \leq j \leq J-2$ we have that

$$U_j^{n+1} = \begin{bmatrix} \nu & (1-2\nu) & \nu \end{bmatrix} \begin{bmatrix} U_{j-1}^n \\ U_j^n \\ U_{j+1}^n \end{bmatrix}$$

Along the boundary line x_j such that $j = 0$ we have U_0^n, for $0 < n \leq N$

$$\begin{aligned} U_j^{n+1} &= \begin{bmatrix}(1-2\nu) & \nu\end{bmatrix} \begin{bmatrix}U_1^n \\ U_2^n\end{bmatrix} + \nu U_0^n \\ &= \begin{bmatrix}(1-2\nu) & \nu\end{bmatrix} \begin{bmatrix}U_1^n \\ U_2^n\end{bmatrix} + \nu \cdot \alpha(n\Delta t) \end{aligned}$$

and along the boundary line x_j such that $j = J$ we have U_J^n, for $0 < n \leq N$

$$\begin{aligned} U_j^{n+1} &= \begin{bmatrix}(1-2\nu) & \nu\end{bmatrix} \begin{bmatrix}U_{J-1}^n \\ U_{J-2}^n\end{bmatrix} + \nu U_J^n \\ &= \begin{bmatrix}(1-2\nu) & \nu\end{bmatrix} \begin{bmatrix}U_{J-1}^n \\ U_{J-2}^n\end{bmatrix} + \nu \cdot \beta(n\Delta t) \end{aligned}$$

Putting it all together we get

$$\vec{U}^{n+1} = \mathbf{F}\vec{U}^n + \vec{p}^n$$

$$\begin{bmatrix} U_1^{n+1} \\ \vdots \\ U_j^{n+1} \\ \vdots \\ U_{J-1}^{n+1} \end{bmatrix} = \begin{bmatrix} (1-2\nu) & \nu & 0 & 0 & \cdots & 0 & 0 & 0 \\ \nu & (1-2\nu) & \nu & 0 & \cdots & 0 & 0 & 0 \\ 0 & \nu & (1-2\nu) & \nu & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & \nu & (1-2\nu) \end{bmatrix} \times \begin{bmatrix} U_1^n \\ \vdots \\ U_j^n \\ \vdots \\ U_{J-1}^n \end{bmatrix} + \begin{bmatrix} \nu \cdot \alpha(n\Delta t) \\ 0 \\ \vdots \\ 0 \\ \nu \cdot \beta(n\Delta t) \end{bmatrix}$$

or in the more succinct notation introduced earlier

$$\vec{U}^{n+1} = \mathbf{F}\vec{U}^n + \vec{p}^n$$

with our initial conditions

$$\vec{U}^0 = \begin{bmatrix} \lambda(x_1) \\ \vdots \\ \lambda(x_j) \\ \vdots \\ \lambda(x_{J-1}) \end{bmatrix}$$

where $\vec{U}^0$ is a vector of of length $J - 2$ since the initial values for U at x_0 and x_J are specified by the boundary conditions.

1.1.1 Stability in Matrix Form

Going back to the stability condition, we now consider our matrix specification of the FTCS algorithm. We can write the error of our output as[1]

$$\mathbf{E}^{n+1} = (\mathbf{I} - \nu\mathbf{A})\mathbf{E}^n + (\Delta t)\mathbf{T}^n$$

[1]We split the matrix $\mathbf{F}$ into $\mathbf{F} = \mathbf{I} - \nu\mathbf{A}$ so that we can do some eigenvalue/vector tricks.

where $\mathbf{A}$ is a tridiagonal matrix

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 2 & -1 \end{bmatrix}$$

with $\mathbf{T}^n$ the truncation error (unimportant if our method is stable). Therefore, we see that our result is entirely dependent on the matrix $(\mathbf{I} - \nu\mathbf{A})$.

Lemma 1. The eigenvalues of a $(m \times m)$ tridiagonal matrix $\mathbf{M}$ of the form

$$\mathbf{M} = \begin{bmatrix} \beta & \gamma & 0 & 0 & \cdots & 0 & 0 & 0 \\ \alpha & \beta & \gamma & 0 & \cdots & 0 & 0 & 0 \\ 0 & \alpha & \beta & \gamma & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \alpha & \beta & \gamma \\ 0 & 0 & 0 & 0 & \cdots & 0 & \beta & \gamma \end{bmatrix}$$

are[2]

$$\mu_k = \beta + 2\beta\sqrt{\alpha\gamma}\cos\left(\frac{k\pi}{m+1}\right) \qquad k = 1, ..., m$$

with the corresponding j^{th} component of the k^{th} eigenvector

$$(\psi_j)_k = \left(\sqrt{\frac{\alpha}{\gamma}}\right)^j \sin\left(\frac{jkm}{m+1}\right) \qquad j = 1, ..., m$$

where

$$\vec{\psi}_k = \begin{bmatrix} \psi_1 \\ \vdots \\ \psi_j \\ \vdots \\ \psi_m \end{bmatrix}$$

Proof. Proof left as an exercise for the reader.[3] $\square$

Using the above lemma in the FTCS algorithm with $m = (J-1), \alpha = \gamma = -1$ and $\beta = 2$ to get our eigenvalues

$$\begin{aligned} \mu_k &= (2) + 2(2)\sqrt{(-1)(-1)}\cos\left(\frac{k\pi}{J}\right) \qquad k = 1, ..., J-1 \\ &= 2 - 2\cos\left(\frac{k\pi}{J}\right) \end{aligned}$$

[2]This result was not given correctly in class (or I wrote the result incorrectly). This is the correct solution to the eigenvalues of a tridiagonal matrix.

[3]It's not really difficult just incredibly tedious and uninteresting manipulation of linear algebra. I somehow doubt we will be tested on this proof...

and using the identity $\frac{1-\cos(2x)}{2} = \sin^2 u$ we may write our eigenvalues μ_k as

$$\begin{aligned}\mu_k &= 2 - 2\cos\left(\frac{k\pi}{J}\right) \\ &= 4 \cdot \frac{1-\cos\left(2\cdot\frac{k\pi}{2J}\right)}{2} \\ &= 4\sin^2\left(\frac{k\pi}{2J}\right)\end{aligned}$$

with corresponding eigenvector/vector components

$$\begin{aligned}(\psi_j)_k &= \left(\sqrt{\frac{(-1)}{(-1)}}\right)^j \sin\left(\frac{jk\pi}{J}\right) \qquad j,k = 1,...,J-1 \\ (\psi_j)_k &= \sin\left(\frac{jk\pi}{J}\right) \\ \vec{\psi}_k &= \begin{bmatrix}\psi_1 \\ \vdots \\ \psi_j \\ \vdots \\ \psi_{J-1}\end{bmatrix}\end{aligned}$$

We note that the eigenvalues for $\mathbf{A}$ are on the open interval $(0,4)$ and so the eigenvalues for $\mathbf{F} = \mathbf{I} - \nu\mathbf{A}$ lie on the interval $(1-4\nu, 1)$ with the smallest eigenvalue corresponding to $j = J-1$. Furthermore, $\nu \leq \frac{1}{2}$ gives us all eigenvalues in $(1 - 4\cdot\frac{1}{2}, 1) = (-1,1)$ and our eigenvector of $j = J-1$ will have entries

$$\sin\left(\frac{(J-1)k\pi}{J}\right)$$

Recall that we expressed the error as

$$\mathbf{E}^{n+1} = (\mathbf{I} - \nu\mathbf{A})\mathbf{E}^n + (\Delta t)\mathbf{T}^n$$

Then, with the eigenvalues/vectors above, this tells us that our error terms are oscillating between finite values, which is a desirable trait since this reassures us that they will not explode. We call pattern this "perfect oscillation".

2 Implicit PDE Methods

Say we cannot determine the value of U_j^{n+1} from $\vec{U}^n$. That is, we are unable (or unwilling) to express a future point as a function of the past values. A solution is to set up an implicit expression for U_j^{n+1}, with U_j^{n+1} appearing on both sides of our equation. This requires us solve an equation (or system of equations) at each step to compute U_j^{n+1}. The gain for this extra work is greater stability.

2.1 BTCS

The first implicit method we will discuss is the <u>backwards in time, central in space</u> (BTCS) algorithm. We approximate $\frac{\partial u}{\partial t}$ at point (x_j, t_{n+1}) by

$$\partial_t u_j^{n+1} \approx \frac{u_j^{n+1} - u_j^n}{\Delta t} = \delta_t^- u_j^{n+1}$$

and we approximate the second spatial derivative $\frac{\partial^2 u}{\partial x^2}$ at (x_j, t_{n+1}) by

$$\begin{aligned}
\partial_x u_j^{n+1} &\approx \frac{u_j^{n+1} - u_{j-1}^n}{\Delta x} \quad \text{(we first use the backwards } \delta_x^-\text{)}\\
\partial_x^2 u_j^{n+1} &\approx \partial_x \frac{u_j^{n+1} - u_{j-1}^n}{\Delta x}\\
&= \frac{[u_{j+1}^{n+1} - u_j^{n+1}] - [u_j^{n+1} - u_{j-1}^{n+1}]}{(\Delta x)^2} \quad \text{(we now use the forwards } \delta_x^+\text{)}\\
&= \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{(\Delta x)^2} = \delta_x^+ \delta_x^- u_j^{n+1}
\end{aligned}$$

Then, the discretized balance equations are

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} = \frac{U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}}{(\Delta x)^2}$$

Rearranging yields

$$\begin{aligned}
U_j^{n+1} - U_j^n &= \frac{\Delta t}{(\Delta x)^2}\left(U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}\right)\\
U_j^{n+1} &= U_j^n + \nu\left(U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}\right) \quad \text{for } \nu = \frac{\Delta t}{(\Delta x)^2}
\end{aligned}$$

or equivalently

$$U_j^n = -\nu U_{j+1}^{n+1} + (1+2\nu)U_j^{n+1} - \nu U_{j-1}^{n+1}$$

for $0 < j < J$. In vector notation we may write

$$\begin{aligned}
\mathbf{B}\vec{U}^{n+1} &= \vec{U}^n + \vec{q}^n \quad 0 \le n \le N-1\\
\vec{U}^n &= \mathbf{B}\vec{U}^{n+1} - \vec{q}^n
\end{aligned}$$

or, expanding the terms,

$$\vec{U}^n = -\nu U_{j+1}^{n+1} + (1+2\nu)U_j^{n+1} - \nu U_{j-1}^{n+1}$$

$$\begin{bmatrix} U_1^n \\ \vdots \\ U_j^n \\ \vdots \\ U_{J-1}^n \end{bmatrix} = \begin{bmatrix} (1-2\nu) & -\nu & 0 & \cdots & 0 & 0 & 0 \\ -\nu & (1-2\nu) & -\nu & \cdots & 0 & 0 & 0 \\ 0 & -\nu & (1-2\nu) & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -\nu & (1-2\nu) \end{bmatrix} \times \begin{bmatrix} U_1^{n+1} \\ \vdots \\ U_j^{n+1} \\ \vdots \\ U_{J-1}^{n+1} \end{bmatrix} - \begin{bmatrix} \nu \cdot \alpha((n+1)\Delta t) \\ 0 \\ \vdots \\ 0 \\ \nu \cdot \beta((n+1)\Delta t) \end{bmatrix}$$

with initial conditions, letting $u(x,0) = \lambda(x)$,

$$\vec{U}^0 = \begin{bmatrix} \lambda(x_1) \\ \vdots \\ \lambda(x_j) \\ \vdots \\ \lambda(x_{J-1}) \end{bmatrix}$$

So, in order to compute $\vec{U}^{n+1}$ we solve the linear system $\vec{U}^n = \mathbf{B}\vec{U}^{n+1} - \vec{q}^n$ at each iteration in n.

2.1.1 Error Bounds

Earlier we stated that the advantage of implicit schemes is an increase in stability and so we should comment on the error analysis of the BTCS method. Given the Dirichlet boundary conditions we have that the error along the boundaries are zero,

$$E_0^{n+1} = E_J^{n+1} = 0$$

and we may write the error at each node (x_j, t_{n+1}) as

$$-\nu E_{j+1}^{n+1} + (1+2\nu)E_j^{n+1} - \nu E_{j-1}^{n+1} = E_j^n + \Delta t T_j^{n+1}$$

where the truncation error at node T_j^{n+1} is

$$T_j^{n+1} = \delta_t^- U_j^{n+1} - \delta_x^- \delta_x^- U_j^{n+1}$$

In matrix form we have

$$(\mathbf{I} - \nu\mathbf{A})\mathbf{E}^{n+1} = \mathbf{E}^n + \Delta t\mathbf{T}^{n+1}$$

and we can show that, similar to the FTCS case, the eigenvalues of $(\mathbf{I} - \nu\mathbf{A})$ lie in the interval $(1, 1+4\nu)$. That is, we are guaranteed to have all eigenvalues greater than 1: This is good news! We didn't have this before. With eigenvalues greater than 1, we are assured that the matrix $(\mathbf{I} - \nu\mathbf{A})$ is invertible permitting us the application of some fun matrix algebra tricks on our problem. We have

$$\mathbf{E}^{n+1} = (\mathbf{I} + \nu\mathbf{A})^{-1}\left(\mathbf{E}^n + \Delta t\mathbf{T}^{n+1}\right)$$

With the "scaled L_2 norm"

$$\|\nu\|_2^2 = \frac{1}{J+1}\sum_{k=0}^{J} |\nu_j|^2$$

we have

$$\left\|\mathbf{E}^{n+1}\right\|_2 \leq \|\mathbf{E}^n\|_2 + \Delta t \left\|\mathbf{T}^{n+1}\right\|_2$$

The key is that the error here no longer is dependent on ν. We say that the cumulative error bound is unconditional on ν (i.e. no stability restriction with respect to time or space step granularity). Taking the max error over our grid we have

$$\max_{0\leq n\leq N} \|\mathbf{E}^n\|_2 \leq c_1 T\Delta t + c_2 T(\Delta x)^2$$

for constants c_1, c_2 proportional u_{tt}, u_{xxxx}.

2.2 Crank-Nicolson Implicit Finite Difference Method

The idea behind the Crank-Nicolson method is to "average" the FTCS and BTCS algorithms.[4] That is, the Crank-Nicolson scheme adds the balance equations for the FTCS and BTCS algorithms and divides by 2. For the heat equation with get the Crank-Nicolson balance equation

$$U_j^{n+1} = \frac{1}{2}\nu U_{j+1}^{n+1} + \frac{1}{2}(1-2\nu)U_j^n + \frac{1}{2}\nu U_{j-1}^n + \frac{1}{2}U_j^n + \frac{1}{2}\nu\left(U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}\right)$$

Collecting terms, with all the $n+1$ terms on the LHS and all n terms on the RHS, we get

$$U_j^{n+1} - \frac{1}{2}\nu\left(U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}\right) = U_j^n + \frac{1}{2}\nu\left(U_{j+1}^n - 2U_j^n + U_{j-1}^n\right)$$

So, we have set up our equation such that the values U_j^{n+1}, U_{j-1}^{n+1} and U_{j+1}^{n+1} determined by U_j^n, U_{j-1}^n, and U_{j+1}^n. As a linear system we write

$$\mathbf{B}\vec{U}^{n+1} = \mathbf{F}\vec{U}^n + \vec{r}^n \quad 0 \le n \le N-1$$

with

$$\mathbf{B} = \text{tridiag}\left(-\frac{1}{2}\nu, (1+\nu), -\frac{1}{2}\nu\right) \in (J-1)\times(J-1)$$
$$\mathbf{F} = \text{tridiag}\left(\frac{1}{2}\nu, (1-\nu), \frac{1}{2}\nu\right) \in (J-1)\times(J-1)$$

and $\vec{r}$ is the "average" of the boundary conditions from the FTCS & BTCS algorithms

$$\vec{r}^n = \begin{bmatrix} \frac{1}{2}\nu\cdot[\alpha(n\Delta t) + \alpha((n+1)\Delta t)] \\ 0 \\ \vdots \\ 0 \\ \frac{1}{2}\nu\cdot[\beta(n\Delta t) + \beta((n+1)\Delta t)] \end{bmatrix}$$

The stability of the Crank-Nicolson algorithm turns out to be unconditional on ν as in the BTCS method (good!) and the truncation error is $\mathcal{O}\left((\Delta t)^2 + (\Delta x)^4\right)$. In general, we see faster convergence using the Crank-Nicolson approach than the BTCS approach, but keeping the same stability (very good!).

3 The Binomial Model as a Finite Difference Problem

Lets go back to how the binomial model was specified and see if we can recover how it is just a special case of a finite difference method. We will apply the FTCS algorithm on the Black-Scholes PDE:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

[4]There are also PDE schemes that apply weighted averages to the FTCS and BTCS algorithms which have advantages when dealing with certain PDEs.

We perform the substitution $x = \log S$, so

$$\begin{aligned}
\frac{\partial V}{\partial S} &= \frac{\partial V}{\partial x}\frac{\partial x}{\partial S} = \frac{1}{S}\frac{\partial V}{\partial x} \\
\frac{\partial^2 V}{\partial S^2} &= \frac{\partial}{\partial S}\left(\frac{1}{S}\frac{\partial V}{\partial x}\right) \\
&= -\frac{1}{S^2}\frac{\partial V}{\partial x} + \frac{1}{S^2}\frac{\partial V}{\partial x}
\end{aligned}$$

leaving us with

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2\frac{\partial^2 V}{\partial x^2} + \left(r - \frac{1}{2}\sigma^2\right)\frac{\partial V}{\partial x} - rV = 0$$

Then, we let $V = e^{rt}W$ (i.e. we undiscount the value of the contingent claim), so

$$\begin{aligned}
\frac{\partial V}{\partial t} &= re^{rt}W + e^{rt}\frac{\partial W}{\partial t} \\
\frac{\partial V}{\partial x} &= e^{rt}\frac{\partial W}{\partial x} \\
\frac{\partial^2 V}{\partial x^2} &= e^{rt}\frac{\partial^2 W}{\partial x^2}
\end{aligned}$$

leaving us with

$$\begin{aligned}
\left[re^{rt}W + e^{rt}\frac{\partial W}{\partial t}\right] + \frac{1}{2}\sigma^2\left[e^{rt}\frac{\partial^2 W}{\partial x^2}\right] + \left(r - \frac{1}{2}\sigma^2\right)\left[e^{rt}\frac{\partial W}{\partial x}\right] - r\left[e^{rt}W\right] &= 0 \\
re^{rt}W + e^{rt}\frac{\partial W}{\partial t} + \frac{1}{2}\sigma^2 e^{rt}\frac{\partial^2 W}{\partial x^2} + \left(r - \frac{1}{2}\sigma^2\right)e^{rt}\frac{\partial W}{\partial x} - re^{rt}W &= 0 \\
\frac{\partial W}{\partial t} + \frac{1}{2}\sigma^2\frac{\partial^2 W}{\partial x^2} + \left(r - \frac{1}{2}\sigma^2\right)\frac{\partial W}{\partial x} &= 0
\end{aligned}$$

We use the approximations

$$\begin{aligned}
\frac{\partial W}{\partial t} &\approx \delta_t^- W_j^{n+1} = \frac{W_j^{n+1} - W_j^n}{\Delta t} \\
\frac{\partial W}{\partial x} &\approx \delta_x^- W_j^{n+1} = \frac{W_j^{n+1} - W_{j-1}^{n+1}}{\Delta x} \\
\frac{\partial^2 W}{\partial x^2} &\approx \delta_x^+\delta_x^- W_j^{n+1} = \frac{W_{j+1}^{n+1} - 2W_j^{n+1} - W_{j-1}^{n+1}}{(\Delta x)^2}
\end{aligned}$$

and so our PDE is approximated by

$$\left(\frac{W_j^{n+1} - W_j^n}{\Delta t}\right) + \frac{1}{2}\sigma^2\left(\frac{W_{j+1}^{n+1} - 2W_j^{n+1} - W_{j-1}^{n+1}}{(\Delta x)^2}\right) + \left(r - \frac{1}{2}\sigma^2\right)\left(\frac{W_{j+1}^{n+1} - W_{j-1}^{n+1}}{\Delta x}\right) = 0$$

Letting $(\Delta x)^2 = \sigma^2 \Delta t$ we find

$$
\begin{aligned}
\left(\frac{W_j^{n+1} - W_j^n}{\Delta t}\right) + \frac{1}{2}\sigma^2 \left(\frac{W_{j+1}^{n+1} - 2W_j^{n+1} - W_{j-1}^{n+1}}{\sigma^2 \Delta t}\right) + \left(r - \frac{1}{2}\sigma^2\right)\left(\frac{W_{j+1}^{n+1} - W_{j-1}^{n+1}}{\sigma\sqrt{\Delta t}}\right) &= 0 \\
W_j^{n+1} - W_j^n + \frac{1}{2}\left(W_{j+1}^{n+1} - 2W_j^{n+1} - W_{j-1}^{n+1}\right) + \frac{\sqrt{\Delta t}}{\sigma}\left(r - \frac{1}{2}\sigma^2\right)\left(W_{j+1}^{n+1} - W_{j-1}^{n+1}\right) &= 0 \\
-W_j^n + \frac{1}{2}\left(W_{j+1}^{n+1} - W_{j-1}^{n+1}\right) + \frac{\sqrt{\Delta t}}{\sigma}\left(r - \frac{1}{2}\sigma^2\right)\left(W_{j+1}^{n+1} - W_{j-1}^{n+1}\right) &= 0 \\
\frac{1}{2}\left(W_{j+1}^{n+1} - W_{j-1}^{n+1}\right) + \frac{\sqrt{\Delta t}}{\sigma}\left(r - \frac{1}{2}\sigma^2\right)\left(W_{j+1}^{n+1} - W_{j-1}^{n+1}\right) &= W_j^n \\
\left(W_{j+1}^{n+1} - W_{j-1}^{n+1}\right)\left(\frac{1}{2} + \frac{\sqrt{\Delta t}}{\sigma}\left(r - \frac{1}{2}\sigma^2\right)\right) &= W_j^n \\
\left(W_{j+1}^{n+1} - W_{j-1}^{n+1}\right)\left(\frac{1}{2} + \sqrt{\Delta t}\left(\frac{r}{\sigma} - \frac{\sigma}{2}\right)\right) &= W_j^n
\end{aligned}
$$

and letting

$$\tilde{p} = \left(\frac{1}{2} + \sqrt{\Delta t}\left(\frac{r}{\sigma} - \frac{\sigma}{2}\right)\right)$$

we note that[5]

$$
\begin{aligned}
-\tilde{p} &= -\left(\frac{1}{2} + \sqrt{\Delta t}\left(\frac{r}{\sigma} - \frac{\sigma}{2}\right)\right) \\
&= -\frac{1}{2} - \sqrt{\Delta t}\left(\frac{r}{\sigma} - \frac{\sigma}{2}\right) \\
&= 1 - \frac{1}{2} - \sqrt{\Delta t}\left(\frac{r}{\sigma} - \frac{\sigma}{2}\right) \\
&= 1 - \tilde{p}
\end{aligned}
$$

so we are left with

$$\tilde{p} W_{j+1}^{n+1} + (1 - \tilde{p}) W_{j-1}^{n+1} = W_j^n$$

and transforming back to V we get

$$V_j^n = e^{-r\Delta t}\left(\tilde{p} V_{j+1}^{n+1} + (1 - \tilde{p}) V_{j-1}^{n+1}\right)$$

which is identical to the binomial model. This $(\Delta x)^2 = \sigma^2 \Delta t$ puts the method "on the edge of stability/instability". This explains the oscillations in error we see between the binomial and Black-Scholes "true" solution when pricing options if we increase the number of time steps in the binomial tree.

[5] This is not correct. Line 3 is a mistake.