Finite Difference Method for an ODE Black Scholes Equation

Prateek Mishra

Department of Mathematics
University of Georgia

May 8, 2024

§1. A Financial Introduction to Options

An option is a financial instrument that gives the holder the right, but not the obligation, to buy or sell a specific quantity of an underlying asset or instrument at a specified strike price. Options are not simple financial instruments and have many different variables that weight in on what is a fair price. The two main factors that determine an options payoff are the extrinsic and intrinsic value. The intrinsic value represents the payoff of an option that would be immediately exercised. This can be represented as

$$\text{Intrinsic Value} = \max(0, S - K)$$

for a call option where S represents the current stock price and K represents the strike price. In addition to this we have the extrinsic value which can be decomposed into the following factors:

- **Expected Volatility** (σ): More variance of returns implies an increase in extrinsic value.
- **Theta Decay** ($\frac{dV}{dt}$): An option's extrinsic price will decay to 0 at the expiration date. The longer the time to expiration, the more the extrinsic value.
- **Risk Free Rate** (r): Higher rates increase extrinsic value.

An options overall price will be determined by adding the intrinsic and extrinsic values. We now take a mathematical approach to derive a formula to determine V(S, t), an options price at stock price S and time t.

§2. Derivation of the Black Scholes Equation

We now proceed with a purely mathematical approach to determine an options price. It is well known that stock prices follow Brownian Motion with drift parameter (μ) and volatility (σ). This can be represented as

$$dX_t = \mu_t\, dt + \sigma_t\, dW_t$$

where W_t is a standard Brownian Motion. This can be intuitively understood as μ representing expected change and σ_t, dW_t represents the unpredictable randomness of the process. An option depends on both dX_t and t so we can apply Ito's lemma. The differential $df(X_t, t)$ is given by:

$$df(X_t, t) = \left(\frac{\partial f}{\partial t} + \mu_t \frac{\partial f}{\partial x} + \frac{1}{2}\sigma_t^2 \frac{\partial^2 f}{\partial x^2}\right) dt + \sigma_t \frac{\partial f}{\partial x} dW_t$$

We now eliminate dW_t in a clever way. If we can eliminate the options dependency on small changes of the underlying asset, randomness on a small time interval is no longer a factor.

To accomplish this we hedge the option. Hedging refers to taking counteractive positions in related assets. In this case the two assets are the underlying stock S and the call option V. To counteract a call option, we sell the underlying asset aiming to eliminate risk. This in theory should result in the risk free interest rate r. Our hedge would have to keep being updated to maintain a risk free position relative to time. In finance this is known as dynamic hedging. The amount of shares we sell is known as delta ($\triangle$). The position in the underlying asset is adjusted by $V + \Delta S$, where we hedge our option V with delta shares of the underlying stock. ΔS also follows a geometric Brownian motion:

$$d(\Delta S) = \Delta dS = \Delta(\mu S\, dt + \sigma S\, dW_t)$$

Combining dV and $d(\Delta S)$ gives the total change in the portfolio $V + \Delta S$:

$$d(V + \Delta S) = dV + d(\Delta S)$$

Substituting the expressions:

$$d(V + \Delta S) = \left(\frac{\partial V}{\partial t} + \mu S\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}\right) dt + \sigma S\frac{\partial V}{\partial S} dW_t - \Delta(\mu S dt + \sigma S dW_t)$$

Using the substitution $\Delta = -\frac{\partial V}{\partial S}$, the term dW_t is canceled as needed:

$$d(V + \Delta S) = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}\right) dt$$

Our initial idea said a perfectly hedged option position with Δ shares would result in the risk free rate. This gives us our final equation:

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

Our equation can be used to solve for the price of an option V. This equation is a PDE which usually requires numerical methods to compute. Looking at the equation we see the RHS represents a hedged option portfolio growing at the risk free rate. As $\frac{\partial V}{\partial t}$ is not something that can be directly changed, $\frac{\partial^2 V}{\partial S^2}$ (gamma) can be considered as the cost of hedging an option. Hedge funds, market makers, and institutions all using hedging in practice. This is the basis of a huge rabbit hole considering overall market gamma exposure (GEX) where many retail traders try to understand how these firms are hedging to get a competitive edge. It is important to note that this equation makes a lot of assumptions (often unrealistic) to make a quantifiable result. These include the ability to buy and sell at the risk free rate, no transaction fees, and no arbitrage. However, these can be accounted for and the equation gives a good quick estimate for a potential hedge and pricing.

As a basis for future research, the equation that will be approximated is one that is time independent. This basically changes our PDE to an ODE. Making $\frac{\partial V}{\partial t} = 0$ and moving everything to the LHS, we have:

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

Our goal is to get a good numerical approximation for this equation. To do this we employ a numerical technique known as the Finite Difference Method.

§3. Finite Difference Method

The finite difference method is a numerical technique that breaks down a continuous grid into discretized points. The more points we choose for our approximation, the more it will match the continuous true solution. To do this, we use approximations for the first and second derivatives which depend on discrete points adjacent to the desired value. The approximations for each derivative can be derived using Taylor polynomials. Let us let u represent the options price V and x represent the underlying price S.

Our goal is to approximate $u(x)$ using its adjacent points. What we can do is use $x_{i+1} = x_i + \Delta x$ and $x_{i-1} = x_i - \Delta x$ expansions and then subtract them. After that we isolate $u'(x_i)$. The reason central difference approximations are used are that they are second order accurate (proportional to $(\Delta x)^2$) which is an improvement over the standard forward and backward differences.

For x_{i+1}:

$$u(x_{i+1}) = u(x_i + \Delta x) = u(x_i) + u'(x_i)\Delta x + \frac{u''(x_i)}{2}(\Delta x)^2 + O((\Delta x)^3)$$

For x_{i-1}:

$$u(x_{i-1}) = u(x_i - \Delta x) = u(x_i) - u'(x_i)\Delta x + \frac{u''(x_i)}{2}(\Delta x)^2 + O((\Delta x)^3)$$

Subtraction eliminates all even powers of Δx and isolates terms involving $u'(x_i)$:

$$u(x_{i+1}) - u(x_{i-1}) = (u(x_i) + u'(x_i)\Delta x + \frac{u''(x_i)}{2}(\Delta x)^2) - (u(x_i) - u'(x_i)\Delta x + \frac{u''(x_i)}{2}(\Delta x)^2) + O((\Delta x)^3)$$

Simplifying the above,

$$u(x_{i+1}) - u(x_{i-1}) = 2u'(x_i)\Delta x + O((\Delta x)^3)$$

We can now isolate for $u'(x_i)$ by dividing through by $2\Delta x$:

$$\frac{u(x_{i+1}) - u(x_{i-1})}{2\Delta x} = u'(x_i) + O((\Delta x)^2)$$

This gives the central difference formula for the first derivative:

$$u'_x \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(h^2) \tag{2}$$

A similar methodology is used for the second derivative. Now we add the two expansions together with one more term.

To isolate the second derivative, add the expansions of $u(x_{i+1})$ and $u(x_{i-1})$ and subtract $2u(x_i)$:

$$u(x_{i+1}) + u(x_{i-1}) - 2u(x_i) = \left(u(x_i) + u'(x_i)\Delta x + \frac{u''(x_i)}{2}(\Delta x)^2 + \frac{u'''(x_i)}{6}(\Delta x)^3\right)$$
$$+ \left(u(x_i) - u'(x_i)\Delta x + \frac{u''(x_i)}{2}(\Delta x)^2 - \frac{u'''(x_i)}{6}(\Delta x)^3\right) - 2u(x_i) + O((\Delta x)^4)$$

Simplifying, the first derivative terms $u'(x_i)\Delta x$ cancel each other out, and we are left with:

$$u(x_{i+1}) + u(x_{i-1}) - 2u(x_i) = u''(x_i)(\Delta x)^2 + O((\Delta x)^4)$$

To find an approximation for $u''(x_i)$, divide through by $(\Delta x)^2$:

$$\frac{u(x_{i+1}) + u(x_{i-1}) - 2u(x_i)}{(\Delta x)^2} = u''(x_i) + O((\Delta x)^2)$$

This gives the central difference formula for the second derivative:

$$u_{xx} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} + O(h^2) \tag{3}$$

§4. Matrix Formulation and Gauss-Seidel

Using what we have derived, plug in our derivative approximations (2) and (3) into the Black Scholes ODE (1). This results in:

$$\frac{x_i^2\sigma^2}{2}\left(\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}\right) + rx_i\left(\frac{u_{i+1} - u_{i-1}}{2\Delta x}\right) - ru_i = 0$$

Now rearrange the equation to find coefficients at each mesh point:

$$\left(\frac{x_i^2\sigma^2}{2\Delta x^2} - \frac{rx_i}{2\Delta x}\right)u_{i-1} - \left(\frac{x_i^2\sigma^2}{\Delta x^2} + r\right)u_i + \left(\frac{x_i^2\sigma^2}{2\Delta x^2} + \frac{rx_i}{2\Delta x}\right)u_{i+1} = 0 \tag{4}$$

Now it is possible to create an equation for every point and then adjusting for the boundaries. The structure of the matrix will be as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & \cdots & 0 \\ 0 & a_{32} & a_{33} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & a_{N-1,N} \\ 0 & 0 & \cdots & a_{N,N-1} & a_{NN} \end{bmatrix}, \quad x = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix}, \quad b = \begin{bmatrix} c_1 \\ 0 \\ \vdots \\ c_N \end{bmatrix}$$

$$a_{ii} = -\frac{x_i^2\sigma^2}{\Delta x^2} - r \text{ for } u_i,$$
$$a_{i,i-1} = \left(\frac{x_i^2\sigma^2}{2\Delta x^2} - \frac{rx_i}{2\Delta x}\right) \text{ for } u_{i-1},$$
$$a_{i,i+1} = \left(\frac{x_i^2\sigma^2}{2\Delta x^2} + \frac{rx_i}{2\Delta x}\right) \text{ for } u_{i+1}$$

Notice the matrix forms a tridiagnol due to the points having a dependence to only their adjacent points from the central difference formulas. The first and last terms of vector b are constants which depend on the values on the boundary values at u_0 and u_{N+1}. Let us look at direct example for the first term. Say $u_0 = 3$ and refer to (4). We plug in $i = 1$ and plug in 3 for u_0. Bringing the constant term to the RHS, the following is found:

$$-\left(\frac{x_1^2\sigma^2}{\Delta x^2} + r\right) u_1 + \left(\frac{x_1^2\sigma^2}{2\Delta x^2} + \frac{rx_1}{2\Delta x}\right) u_2 = -\left(\frac{x_1^2\sigma^2}{2\Delta x^2} - \frac{rx_1}{2\Delta x}\right) \times 3$$

The same process is used for the final term, taking the $i+1$ term to the RHS. This is why the first and last terms are nonzero for vector b.

For larger size matrices, usually we need a significant amount of computational power. However, there do exist other numerical techniques which can simplify these types of matrices. One such method is the Gauss-Seidel method which uses an iterative technique to approximate all values of the solution matrix at a faster rate.

$$u_i^{(\text{new})} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} u_j^{(\text{new})} - \sum_{j=i+1}^{n} a_{ij} u_j^{(\text{old})}}{a_{ii}}$$

The equation looks rather unpleasant but each component is simple to understand. b_i represents the solution vector at each point u_i, the first summation gives all indices a_{ij} which are left of the main diagnol on row i, and the second summation gives the same but the right of the main diagnol. Noticing the pattern in our matrix, the equation simplifies rather nicely as we only have to worry about the sub and super diagnols for the summation. This can be simplified down into:

$$u_i^{(k+1)} = \frac{b_i - a_{i,i-1} u_{i-1}^{(k+1)} - a_{i,i+1} u_{i+1}^{(k)}}{a_{ii}}$$

where k represents the iteration. The initial vector will be a zero vector and after that the iteration process will start. It is important to note that this method only converges for diagonally dominant, positive-definite, or symmetric matrices. To demonstrate diagonal dominance, we need to verify the following condition for each i:

$$|a_{ii}| > |a_{i,i-1}| + |a_{i,i+1}|$$

Substituting the expressions, we get:

$$\left|-\frac{x_i^2\sigma^2}{\Delta x^2} - r\right| > \left|\frac{x_i^2\sigma^2}{2\Delta x^2} - \frac{rx_i}{2\Delta x}\right| + \left|\frac{x_i^2\sigma^2}{2\Delta x^2} + \frac{rx_i}{2\Delta x}\right|$$

This would of course depend on the constants chosen and if this condition does not hold, the method could diverge.

§5. Numerical Experiments and Code

To start let us check if our matrix coefficients are indeed correct before proceeding to the boundary value problem. Let $u(x) = x(x-1)$ with boundary conditions $u_0 = 0$ and $u_1 = 0$. The reason to do this is we now have a solution that we can check with our matrix and get a corresponding error term. Plugging the corresponding derivatives of $u(x)$ into our original equation, an exact solution can be computed and compared. Let us use $\sigma = 0.8$ and risk free rate of $r = 0.2$ then compare results with various matrix sizes.

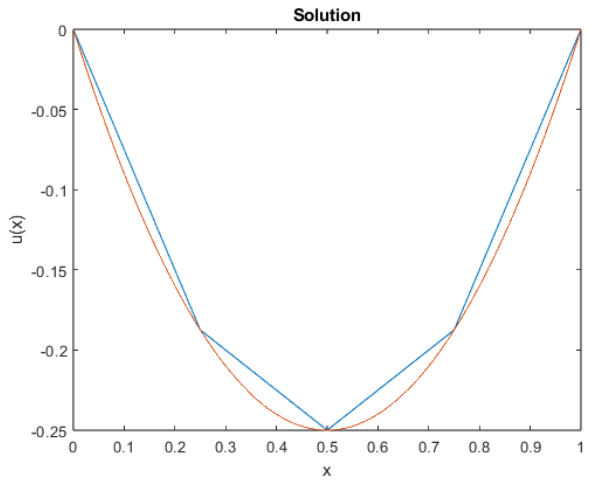

Figure 1: Solution with 3 discrete points

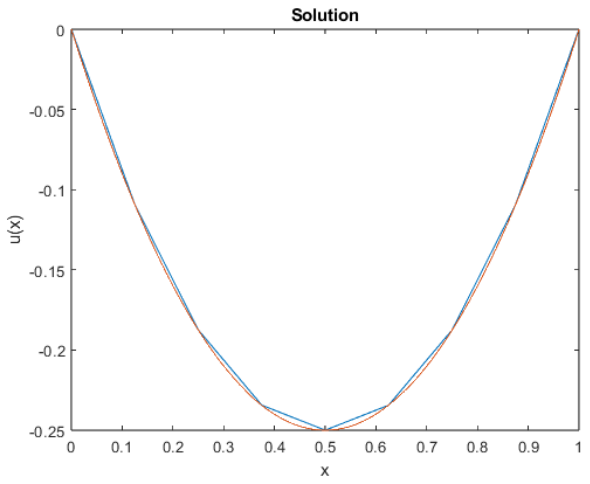

Figure 2: Solution with 7 discrete points

$$A = \begin{bmatrix} -0.8400 & 0.4200 & 0 \\ 1.0800 & -2.7600 & 1.4800 \\ 0 & 2.5800 & -5.9600 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} -0.1875 \\ -0.2500 \\ -0.1875 \end{bmatrix}$$

The following correspond to Figure 1. The size of the matrix will depend on the amount of discretized points on the domain with a NxN matrix for N interior points. The error of our approximation is calculated through vecnorm. As we are approximating a continuous function with discrete points, by taking an L^2 norm of all points, a singular error term is achieved. It is important to note that this does not account for the error between the continuous and discrete functions at points rather than those approximated. So even though with less discrete points there is a smaller error, it does not mean it is a better approximation.

The code for the approximation is provided below.

```
% An approximation to the differential equation:
% (x^2 * sig^2 * uxx)/2 + rx * ux - ru = f for u(x)=x(x-1)
prompt = "Please input the reciprocal of the step size: ";
prompt3 = "Please input the constant sigma: ";
prompt4 = "Please input the constant r: ";
N = input(prompt);
sig = input(prompt3);
r = input(prompt4);

h = 1/N; % Step size
points = (h:h:1-h)'; % Domain not including boundary points

% Main diagonal
main_diag = (-1 * N.^2 * ((points).^2 * sig.^2) - r);
% Sub-diagonal
under_diag = (N.^2 * (points(2:end)).^2 * sig.^2)/2 - (r * (points(2:end) * N)/2);
% Super-diagonal
right_diag = (N.^2 * (points(1:end-1)).^2 * sig.^2)/2 + (r * (points(1:end-1) * N)/2);
A = diag(main_diag, 0) + diag(under_diag, -1) + diag(right_diag, 1);
b = zeros(N-1, 1) + (points).^2 * sig.^2 + r * points * (2 * (points) - 1) - r * (points.^2 - po
disp(A);
x = A \ b;
disp(x);

figure(1);
plot([0; points; 1], [0; x; 0]);
xlabel('x')
ylabel('u(x)')
title('Solution')

% Error calculation
u_exact = @(x) x * (x - 1); % Exact solution function
x_values = linspace(0, 1, N+1); % Linearly spaced vector from 0 to 1
u_values_exact = u_exact(x_values); % Calculate exact values
x_refined = linspace(0, 1, 1000); % More refined space for plotting
hold on
plot(x_refined, u_exact(x_refined)); % Plot refined exact solution
hold off
figure(2);
plot(x_values, u_values_exact); % Plot exact solution over x_values
u_values_approx = x.'; % Transpose of solution vector
err = abs(u_values_exact(2:end-1) - u_values_approx); % Absolute error
disp(vecnorm(err)) % Display norm of the error vector
```

As shown the code creates a tridiagnol matrix and then compares it with the true solution. The boundary terms are not included in the error calculation because they are already known. Now let us move on to a homogeneous case where the RHS=0 and matches our original equation.
The only thing that changes from the previous code is the vector b.

```
b = zeros(length(points), 1);
```

```
b(1) = -a * ((N^2 * points(1)^2 * sig^2) / 2 - (r * points(1) * N) / 2);
b(end) = -c * ((N^2 * points(end)^2 * sig^2) / 2 + (r * points(end) * N) / 2);
```

This will define vector b properly for boundary conditions a and c. Below is an example of a potential solution.

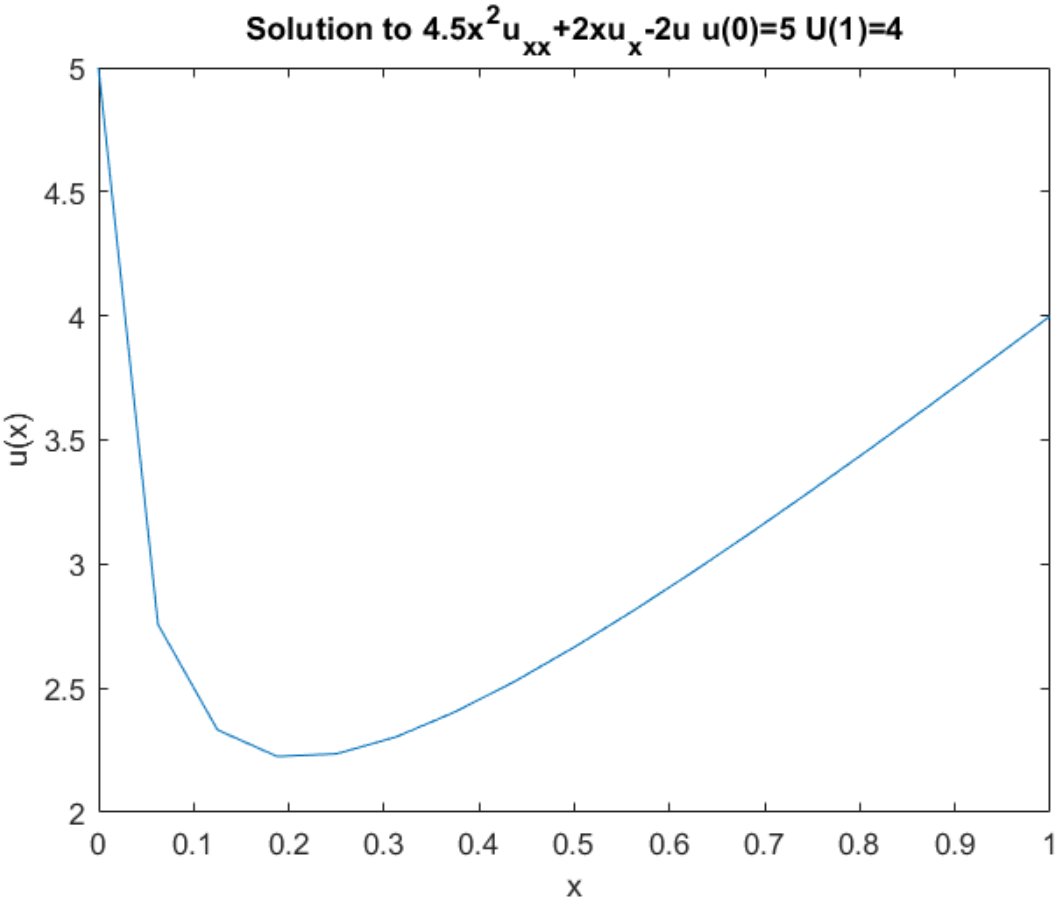

Figure 3: A solution to the homogeneous BSE with chosen boundary conditions

Boundary value problems are quite difficult to solve by hand and are sometimes impossible. This is because the ODE must be of a very specific structure with two constants with different solutions. Some functions can have serious errors if the bounds are chosen poorly.

Now I will present some code regarding the Gauss-Seidel method. This can be implemented into the previous code for larger matrices however as the values of the sub and super diagnols are not constant, it is important to check that the matrix is indeed diagonally dominant.

```
% Gauss-Seidel parameters
tol = 1e-5;
maxIter = 10000000;
err = inf;
iter = 0;
```

```
% Gauss-Seidel iteration
while err > tol && iter < maxIter
x_old = x; % Save the old value of x for convergence check
for i = 1:N
% Calculate the sum of A*x for j != i
sum1 = A(i, 1:i-1) * x(1:i-1); % Contributions from before the diagonal
sum2 = A(i, i+1:end) * x_old(i+1:end); % Contributions from after the diagonal

x(i) = (b(i) - sum1 - sum2) / A(i, i);
end

% Update error and iteration count
err = norm(x - x_old); % Update the error (Euclidean norm)
iter = iter + 1;
end
```

This is quite an easy computation as it only depends on the values before and after the main diagonal. Of course the first and last row do not have dependency on the sub and superdiagnols respectively so it is indexed as such. This can also be used in for solving the PDE version of the Black Scholes equation due to the same pattern arising.

§6. Crank-Nicholson and Conclusion

This paper gives only an approximation to the ODE version of the Black Scholes Equation but how can we approximate the PDE? The easiest method from here is to implement the Crank-Nicholson method which uses a forward difference scheme of two centered differences to get the best result. I shall provide the first couple steps to use this method.

Consider the full PDE:

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

From section 3 we derived the formulas needed for the Finite Difference Approximation. Now take those formulas and average them out at time step n and n+1. This creates a forward approximation for the time term based on the centered approximation's of the x term. The first term dV/dt can be approximated by holding the i term constant and using a forward difference scheme.

$$\frac{u_i^{n+1} - u_i^n}{\Delta t}$$

The terms depending on S will be using two centered difference approximations but averaged between time points n and n+1. This can be represented as follows:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + \frac{1}{2}(L_n + L_{n+1}) = 0$$

where L_n and L_{n+1} are defined as:

$$L_n = \frac{1}{2}\sigma^2 x_i^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + r x_i \frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} - r u_i^n,$$
$$L_{n+1} = \frac{1}{2}\sigma^2 x_i^2 \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} + r x_i \frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{2\Delta x} - r u_i^{n+1}.$$

These are identical to what we approximated before, just with additional time indices. The next step would be to isolate for n+1 on the LHS and n on the RHS and create a system which solves the matrix iteratively.

In this paper I have shown how to approximate the Black Scholes ODE with the Finite Difference Method. We went over the derivation through Ito's lemma, a proof of centered difference formulas, and then showed numerical experiments and examples employing these methods. Further work will be done using the Crank Nicholson method and also applying Gauss Seidel to the PDE approximation. Then exotic options such as lookback, asian, and barriers can be solved. I hope this gave some good insight on essential concepts for financial math and numerical methods.

§6. References

- Burden, Richard L., and J. Douglas Faires. *Numerical Analysis*. 9th ed., Cengage Learning, 2010.
- Louis, Robert. "What is Delta Hedging – The Ultimate Guide." *Trade Options*, 2020.
- Black-Scholes Equation. *Wikipedia*. Wikimedia Foundation, 2024. *URL*.
- Shreve, Steven E. *Stochastic Calculus for Finance II: Continuous-Time Models*. Springer, 2004.
- Peterson, Jim. "Crank-Nicolson Method." *Florida State University*, 2024.