Diffusion Equation

Vegard Falmår and Sigurd Sørlie Rustad

University of Oslo
Norway

December 16, 2020

CONTENTS

Vegard sitt abstract

I. INTRODUCTION

The diffusion equation is used in many areas. Everything from how heat is transferred through a stick, to how an oil spill will spread. In this report we will try to solve the diffusion equation for one and two dimensions, and using different algorithms. We are going to study the diffusion equation with a constant collective diffusion coefficient, giving us a linear differential equation identical to the heat equation.

$$\frac{\partial u}{\partial t} = \nabla^2 u(\mathbf{r}, t)$$

For our one dimensional problem, we use three different algorithms, explicit forward Euler, implicit backward Euler and implicit Crank-Nicolson.

For our studies we have used c++ for heavy computation, python for visualization and automation. All the code along with instructions on how to run it, can be cloned from our GitHub repository[1].

II. THEORY

The diffusion equation

The full diffusion equation reads

$$\frac{\partial u(\mathbf{r}, t)}{\partial t} = \nabla \cdot \left[D(u, \mathbf{r}) \nabla u(\mathbf{r}, t) \right],$$

where $\mathbf{r}$ is a positional vector and $D(u, r)$ the collective diffusion coefficient. If $D(u, \mathbf{r}) = 1$ the equation simplifies to a linear differential equation

$$\frac{\partial u}{\partial t} = \nabla^2 u(\mathbf{r}, t),$$

or

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) u(x, y, z, t) = \frac{\partial u(x, y, z, t)}{\partial t} \quad (1)$$

in cartesian coordinates. In this report we are mainly going to work with the diffusion equation in one and two dimensions, i.e.

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t}$$
$$\wedge$$
$$\frac{\partial^2 u(x,y,t)}{\partial x^2} + \frac{\partial^2 u(x,y,t)}{\partial y^2} = \frac{\partial u(x,y,t)}{\partial t}.$$

[1] github.com/sigurdru/FYS3150/tree/master/Project5

Discretization

Equation (1) in one dimension reads

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t} \quad \text{or} \quad u_{xx} = u_t. \quad (2)$$

With $x \in [0, L]$ and boundary conditions

$$u(0,t) = a(t), \quad t \geq 0 \quad \wedge \quad u(L,t) = b(t), \quad t \geq 0,$$

we can approximate the solution by discretization. First introducing $\Delta x = L/(n+1)$ and Δt as small steps in x-direction and time. Then we can define the value domain of t and x,

$$t_j = j\Delta t, \quad j \in \mathbb{N}_0 \quad \wedge \quad x_i = i\Delta x, \quad \{i \in \mathbb{N}_0 | i \leq n+1\}.$$

Where $n+1$ is the number of points at which we evaluate $u(x,t)$.

Explicit and implicit schemes

It is common divide numerical algorithms for integration into explicit and implicit schemes. When perfoming a numerical integration, we iterate over a discrete set of grid points at which we evaluate the function in question. In explicit schemes, the value at the next grid point is determined entirely by the value at the current grid point. In implicit schemes, the value at a later stage is determined by solving a system of equations containing both the current state and later states.

Using an implicit method instead of an explicit method usually requires more computation in every step, and they are often harder to implement. It can, in turn, save computation by allowing larger step sizes. Implicit methods can be (ARE?) unconditionally stable, meaning they are stable for any choice of step sizes. Explicit schemes are conditionally stable, meaning there is a relation between the step sizes in time and position that must be fulfilled in order to achieve stability. Of course, in order to achieve a desired *accuracy*, the step sizes can not be arbitrarily large for either the explicit or implicit schemes. It is generally the case, though, that implicit schemes allow for larger step sizes than explicit schemes.

REFERENCES, Intro to PDEs book, Wiki

Explicit forward Euler

The algorithm for explicit forward Euler in one dimension (from [1] chapter 10.2.1) reads

$$u_{i,j+1} = \alpha u_{i-1,j} + (1 - 2\alpha) u_{i,j} + \alpha u_{i+1,j} \quad (3)$$

where

$$\alpha = \frac{\Delta t}{\Delta x^2},$$

and a local approximate error of $O(\Delta t)$ and $O(\Delta x^2)$. The discretization is explained in the appropriate section. Note that the expression on the right hand side, used to calculate the value at a time $t_j + \Delta t$, only contains the state of the system at time t_j. This can be written as a matrix equation (see [1] chapter 10.2.1)

$$\mathbf{u}_{j+1} = (\mathbb{1} - \alpha \mathbf{B})\mathbf{u}_j$$

where $\mathbb{1}$ is the identity matrix and

$$\begin{bmatrix} 2 & -1 & 0 & 0\dots \\ -1 & 2 & -1 & 0\dots \\ \dots & \dots & \dots & \dots \\ 0 & \dots & -1 & 2 \end{bmatrix} \quad \wedge \quad \mathbf{u}_j = \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \dots \\ u_{n,j} \end{bmatrix}$$

The stability requirement for this algorithm (also from [1] chapter 10.2.1) is

$$\rho(\mathbf{B}) \leq 1 \implies \frac{\Delta t}{(\Delta x)^2} \leq 1/2 \tag{4}$$

Where $\rho(\mathbf{B})$ is the spectral radius of $\mathbf{B}$:

$$\rho(\mathbf{B}) = \max\{|\lambda| : \det(\mathbf{B} - \lambda \mathbb{1}) = 0\}.$$

In two dimensions $(u(x, y))$ the method for solving is similar. Assuming the same number of integration points in x- and y-direction ($\Delta x = \Delta y = \Delta l$), the algorithm (from [1] chapter 10.2.5) reads

$$u^{l+1}_{i,j} = u^l_{i,j} + \alpha \left[u^l_{i+1,j} + u^l_{i-1,j} + u^l_{i,j+1} + u^l_{i,j-1} - 4u^l_{i,j}\right] \tag{5}$$

where $u^l_{i,j} = u(i\Delta l, j\Delta l, l\Delta t)$ and $\alpha = \Delta t/\Delta l^2$.

Implicit Backward Euler

The Backward Euler algorithm uses the same centered difference in space as Forward Euler to approximate the second derivative

$$u_{xx} \approx \frac{u(x_i + \Delta x, t_j) - 2u(x_i, t_j) + u(x_i - \Delta x, t_j))}{\Delta x^2}, \tag{6}$$

but a backward formula for the time derivative:

$$u_t \approx \frac{u(x_i, t_j) - u(x_i, t_j - \Delta t)}{\Delta t}, \tag{7}$$

which also has a local truncation error of $O(\Delta t)$ and $O(\Delta x^2)$. Again, defining

$$\alpha = \frac{\Delta t}{(\Delta x)^2}$$

we obtain by inserting (6) and (7) into our differential equation the equation describing Backward Euler:

$$u_{i,j-1} = -\alpha u_{i-1,j} + (1 + 2\alpha)u_{i,j} - \alpha u_{i+1,j} \tag{8}$$

Written out for all i, equation 8 becomes

$$\begin{aligned} -\alpha u_{0,j} + (1+2\alpha)u_{1,j} - \alpha u_{2,j} &= u_{1,j-1} \\ -\alpha u_{1,j} + (1+2\alpha)u_{2,j} - \alpha u_{3,j} &= u_{2,j-1} \\ &\dots \\ -\alpha u_{n-3,j} + (1+2\alpha)u_{n-2,j} - \alpha u_{n-1,j} &= u_{n-2,j-1} \\ -\alpha u_{n-2,j} + (1+2\alpha)u_{n-1,j} - \alpha u_{n,j} &= u_{n-1,j-1} \end{aligned}$$

In general, this can be rearranged slightly so that

$$\begin{aligned} (1+2\alpha)u_{1,j} - \alpha u_{2,j} &= u_{1,j-1} + \alpha u_{0,j} \\ -\alpha u_{1,j} + (1+2\alpha)u_{2,j} - \alpha u_{3,j} &= u_{2,j-1} \\ &\dots \\ -\alpha u_{n-3,j} + (1+2\alpha)u_{n-2,j} - \alpha u_{n-1,j} &= u_{n-2,j-1} \\ -\alpha u_{n-2,j} + (1+2\alpha)u_{n-1,j} &= u_{n-1,j-1} + \alpha u_{n,j} \end{aligned}$$

Let $\mathbf{v}_j$ be a vector containing the values of u at $n-1$ points in space at a time t_j

$$\mathbf{v}_j = \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{n-2,j} \\ u_{n-1,j} \end{bmatrix} \tag{9}$$

and the vector $\mathbf{b}_j$ be defined as follows:

$$\mathbf{b}_j = \begin{bmatrix} u_{1,j-1} + \alpha u_{0,j} \\ u_{2,j-1} \\ \vdots \\ u_{n-2,j-1} \\ u_{n-1,j-1} + \alpha u_{n,j} \end{bmatrix} \tag{10}$$

As the boundary conditions $u_{0,j}$ and $u_{n,j}$ are specified, we already know every component of $\mathbf{b}_j$. We can then rewrite equation (8) as the matrix equation

$$\mathbf{A}\mathbf{v}_j = \mathbf{b}_j \tag{11}$$

where $\mathbf{A}$ is defined as

$$A = \begin{bmatrix} (1+2\alpha) & -\alpha & 0 & 0 & \dots & 0 \\ -\alpha & (1+2\alpha) & -\alpha & 0 & \dots & 0 \\ \vdots & & \ddots & & & \vdots \\ 0 & \dots & 0 & -\alpha & (1+2\alpha) & -\alpha \\ 0 & \dots & 0 & 0 & -\alpha & (1+2\alpha) \end{bmatrix} \tag{12}$$

This is a tridiagonal matrix with $(1+2\alpha)$ on the diagonal and $-\alpha$ directly above and below the diagonal.

Implicit Crank-Nicolson

The Crank-Nicolson algorithm uses a time-centered scheme centered around $t+\Delta t/2$, with a truncation error of $O(\Delta t^2)$ and $O(\Delta x^2)$. The time derivative is given by

$$u_t \approx \frac{u(x_i, t_j+\Delta t) - u(x_i, t_j)}{\Delta t} \tag{13}$$

$$\begin{aligned} u_{xx} \approx \frac{1}{2\Delta x^2}\Big(& u(x_i+\Delta x, t_j) - 2u(x_i, t_j) + u(x_i-\Delta x, t_j) \\ & + u(x_i+\Delta x, t_j+\Delta t) - 2u(x_i, t_j+\Delta t) \\ & + u(x_i-\Delta x, t_j+\Delta t)\Big) \end{aligned}$$

Inserting these two equations into our differential equation, we obtain (see Appendix, section VII A for derivation) that the equation describing the Crank-Nicolson algorithm can be written as a matrix equation

$$\mathbf{A}\mathbf{v}_j = \mathbf{b}_j \tag{14}$$

The vector $\mathbf{v}_j$ is the same as in the case of Backward Euler, defined in (9). The matrix $\mathbf{A}$ is a tridiagonal matrix with $2(1+\alpha)$ on the diagonal and $-\alpha$ directly above and below the diagonal. The vector $\mathbf{b}_j$ is

$$\mathbf{b}_j = \begin{bmatrix} \alpha u_{0,j} + 2(1-\alpha)u_{1,j} + \alpha u_{2,j} + \alpha u_{0,j+1} \\ \alpha u_{1,j} + 2(1-\alpha)u_{2,j} + \alpha u_{3,j} \\ \vdots \\ \alpha u_{n-3,j} + 2(1-\alpha)u_{n-2,j} + \alpha u_{n-1,j} \\ \alpha u_{n-2,j} + 2(1-\alpha)u_{n-1,j} + \alpha u_{n,j} + \alpha u_{n,j+1} \end{bmatrix} \tag{15}$$

III. METHODS

A. One dimension

We will start by solving the one dimensional diffusion equation

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t}, \quad x \in [0,1]$$

with initial conditions

$$u(x,0) = 0, \quad 0 < x < 1 \tag{16}$$

and boundary conditions

$$\begin{aligned} u(0,t) &= 0, \quad t \geq 0 \quad \text{and} \\ u(L,t) &= 1, \quad t \geq 0. \end{aligned} \tag{17}$$

As mentioned we use three methods, explicit forward Euler, implicit backward Euler and Crank-Nicolson. When using Explicit forward Euler, we only need to the difference-equation described by equation (3). When using Implicit backward Euler er Crank-Nicolson we need to solve the matrix-equations (11) and (14). There are several methods for solving such tridiagonal matrix equations, however we will use the method covered in a previous project. see project[2] page 2 (section A. Solve tridiagonal matrix equation).

Analytic solution to the 1D diffusion equation

With the initial and boundary conditions described by equations (16) and (17) we can find an analytical solution. We expect the heat distribution to converge to a stable state u_E as $t \to \infty$:

$$\lim_{t\to\infty} u(x,t) = u_E(x)$$

This final state should still satisfy our differential equation such that

$$\frac{\partial^2 u_E(x)}{\partial x^2} = \frac{\partial u_E(x)}{\partial t} = 0$$

The solution to this equation is

$$u_E(x) = Ax + B,$$

and the boundary conditions give

$$\begin{aligned} u_E(0) &= B = 0 \\ u_E(L) &= AL = 1 \quad \Rightarrow \quad u_E(x) = \frac{x}{L} \end{aligned}$$

Let us now define the function

$$\begin{aligned} v(x,t) &= u(x,t) - u_E(x) \quad \text{which gives} \\ u(x,t) &= v(x,t) + u_E(x) \end{aligned}$$

We then have

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial v}{\partial t} + \frac{\partial u_E}{\partial t} = \frac{\partial v}{\partial t} \\ \frac{\partial^2 u}{\partial x^2} &= \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 u_E}{\partial x^2} = \frac{\partial^2 v}{\partial x^2} \end{aligned}$$

Thus, $u(x,t)$ and $v(x,t)$ should both satisfy the same differential equation (2). The intial and boundary conditions for $v(x,t)$ are

$$\begin{aligned} v(x,0) &= u(x,0) - u_E(x) = -u_E(x) \\ v(0,t) &= u(0,t) - u_E(0) = 0 \\ v(0,t) &= u(L,t) - u_E(L) = 0 \end{aligned}$$

In other words, v obeys the same equation as u, but with boundary conditions 0.

[2] https://github.com/sigurdru/FYS3150/tree/master/Project1

It is rather straightforward to see that

$$f_n(x,t) = e^{-C_n^2 t}\left(A\sin(C_n x) + B\cos(C_n x)\right) \tag{18}$$

is a particular solution to the one dimensional diffusion equation:

$$\frac{\partial f_n(x,t)}{\partial t} = \frac{\partial^2 f_n(x,t)}{\partial x^2} = -C_n^2 f_n(x,t)$$

The equation is linear and from the prinsiple of superposition we have that any sum of functions $f_n(x,t)$ with different values of C_n is also a solution. The boundary conditions $v(0,t) = 0$ and $v(L,t) = 0$ give

$$f_n(0,t) = B_n e^{-C_n^2 t} = 0 \quad \Rightarrow \quad B_n = 0$$
$$f_n(L,t) = A_n e^{-C_n^2 t}\sin(C_n L) = 0 \quad \Rightarrow \quad C_n = \frac{n\pi}{L}, n \in \mathbb{Z}$$

The solution $v(x,t)$ can then be written as sum

$$v(x,t) = \sum_n A_n e^{-\left(\frac{n\pi}{L}\right)^2 t}\sin\left(\frac{n\pi}{L}x\right)$$

The coefficients A_n are given by the initial condition $v(x,0) = -u_E(x)$ and can be determined from (see Lecture Notes CITE)

$$\begin{aligned} A_n &= \frac{2}{L}\int_0^L -u_E(x)\ \sin\left(\frac{n\pi}{L}x\right)\,\mathrm{d}x \\ &= -\frac{2}{L^2}\int_0^L x\ \sin\left(\frac{n\pi}{L}x\right)\,\mathrm{d}x \end{aligned}$$

The solution to this is

$$A_n = \frac{2\left(\pi n\cos(\pi n) - \sin(\pi n)\right)}{\pi^2 n^2} \tag{19}$$

The complete solution to our differential equation is thus

$$u(x,t) = \frac{x}{L} + \sum_{n=1}^{\infty} A_n e^{-\left(\frac{n\pi}{L}\right)^2 t}\sin\left(\frac{n\pi}{L}x\right), \tag{20}$$

where the coefficients A_n are given by equation (19).

B. Two dimensions

Moving on to two dimensions we use explicit forward Euler. This means we need to solve the difference equation (5), for $x, y \in [0,1]$. We will use homogeneous Dirichlet boundary conditions

$$\begin{aligned} &u(0,y,t) = u(1,y,t) = 0, \quad 0 \le y \le 1, \quad t \ge 0 \text{ and} \\ &u(x,0,t) = u(x,1,t) = 0, \quad 0 \le x \le 1, \quad t \ge . \end{aligned} \tag{21}$$

and initial conditions given by

$$u(x,y,0) = \begin{cases} 1 & \text{if } 0 < y \le \frac{1}{2} \text{ and } 0 < x < 1 \\ 0 & \text{if } \frac{1}{2} < y < 1 \text{ and } 0 < x < 1 \end{cases} \tag{22}$$

parallelization

When solving the difference equation (5) we parallelize the program using OpenMp. While calculating $u^l_{i,j}$ we only need to worry about the surrounding points for the previous step, namely $u^{l-1}_{i,j}$ and $u^{l-1}_{i\pm1,j\pm1}$. In a grid pattern this would look something like:

$$\begin{array}{ccc} & u^{l-1}_{i,j+1} & \\ u^{l-1}_{i-1,j} & u^{l-1}_{i,j} & u^{l-1}_{i+1,j} \\ & u^{l-1}_{i,j-1} & \end{array}$$

This means that whenever we are done calculating $u^l_{i,j}$ and $u^l_{i\pm1,j\pm1}$ we can start calculating the next time step $u^{l+1}_{i,j}$. We found that if we run through $u^l_{i,j}$ diagonally, we can start calculating the next time step ($u^{l+1}_{i,j}$) one diagonal behind. A visual representation of this can be seen in figure 1.

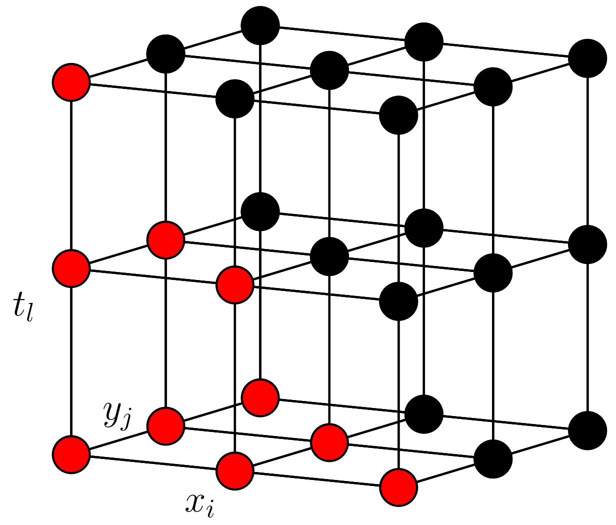

Figure 1. Visualization of how the parallelization works.

Analytical solution to the 2D diffusion equation

with boundary and initial conditions described by equations (21) and (22) we can find analytical solutions to the diffusion equation. Lets say $x \in [0,a]$ and $y \in [0,b]$, with homogeneous Dirichlet boundary and initial conditions defined by some function $f(x,y)$. Then, from [2], the general solution is given by equation (23).

$$u(x,y,t) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} A_{mn}\sin(\mu_m x)\sin(\nu_n y)e^{-\lambda^2_{mn}t}, \tag{23}$$

where

$$\mu_m = \frac{m\pi}{a} \quad \wedge \quad \nu_n = \frac{n\pi}{b} \quad \wedge \quad \lambda_{mn} = \sqrt{\mu_m^2 + \nu_n^2},$$

for $m, n, \in \mathbb{N}$ and

$$A_{mn} = \frac{4}{ab}\int_0^a \int_0^b dydx f(x,y)\sin\Big(\frac{m\pi}{a}x\Big)\sin\Big(\frac{n\pi}{b}\Big).$$

Inserting our initial conditions (22) and $a = b = 1$ into the equation above, we get the coefficients

$$\begin{aligned}
A_{mn} &= 4\int_0^1 \int_0^1 dydxu(x,y,0)\sin(m\pi x)\sin(n\pi y) \\
&= 4\int_0^1 \sin(m\pi x)dx \int_0^{1/2} \sin(n\pi y)dy \\
&= 4\left[-\frac{\cos(m\pi x)}{m\pi}\right]_0^1 \left[-\frac{\cos(n\pi y)}{n\pi}\right]_0^{1/2} \\
&= 4\left(-\frac{(-1)^m - 1}{m\pi}\right)\left(-\frac{\cos\left(\frac{n\pi}{2}\right) - 1}{n\pi}\right) \\
&= \frac{4}{\pi^2}\left(\frac{\left((-1)^m - 1\right)\left(\cos\left(\frac{n\pi}{2}\right) - 1\right)}{mn}\right)
\end{aligned}$$

Inserting this into equation (23)

$$\begin{aligned}
u(x,y,t) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \frac{4}{\pi^2}\left(\frac{\left((-1)^m - 1\right)\left(\cos\left(\frac{n\pi}{2}\right) - 1\right)}{mn}\right) \cdot \\
\sin(m\pi x)\sin(n\pi y)\exp\big(-((m\pi)^2 + (n\pi)^2)t\big)
\end{aligned}$$

C. Unit testing

IV. RESULTS

Results

V. DISCUSSION

LU-decomposition

VI. CONCLUSION

Conclusion

[1] Morten Hjorth-Jensen, Computational Physics, Lecture Notes Fall 2015, August 2015, https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Lectures/lectures2015.pdf.

[2] Ryan C. Daileda, Trinity University, Partial Differential Equations, March 6, 2012, http://ramanujan.math.trinity.edu/rdaileda/teach/s12/m3357/lectures/lecture_3_6_short.pdf

VII. APPENDIX

A. Derivation of equation for the Crank-Nicolson scheme

Again we define

$$\alpha = \frac{\Delta t}{(\Delta x)^2}$$

Inserting the approximations of u_t and u_{xx} stated in the Theory section into our differential equation, we get

$$\begin{aligned}
u_t &= u_{xx} \\
\frac{u_{i,j+1} - u_{i,j}}{\Delta t} &= \frac{1}{2\Delta x^2}\Big(u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}\Big) \\
u_{i,j+1} - u_{i,j} &= \frac{\alpha}{2}\Big(u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}\Big) \\
u_{i,j+1} - \frac{\alpha}{2}\Big(u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}\Big) &= u_{i,j} + \frac{\alpha}{2}\Big(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}\Big) \\
2u_{i,j+1} - \alpha u_{i+1,j+1} + 2\alpha u_{i,j+1} - \alpha u_{i-1,j+1} &= 2u_{i,j} + \alpha u_{i+1,j} - 2\alpha u_{i,j} + \alpha u_{i-1,j} \\
-\alpha u_{i-1,j+1} + 2(1+\alpha)u_{i,j+1} - \alpha u_{i+1,j+1} &= \alpha u_{i-1,j} + 2(1-\alpha)u_{i,j} + \alpha u_{i+1,j}
\end{aligned}$$

Written out explicitly for all i, the equation reads

$$\begin{aligned}
-\alpha u_{0,j+1} + 2(1+\alpha)u_{1,j+1} - \alpha u_{2,j+1} &= \alpha u_{0,j} + 2(1-\alpha)u_{1,j} + \alpha u_{2,j} \\
-\alpha u_{1,j+1} + 2(1+\alpha)u_{2,j+1} - \alpha u_{3,j+1} &= \alpha u_{1,j} + 2(1-\alpha)u_{2,j} + \alpha u_{3,j} \\
&\vdots \\
-\alpha u_{n-3,j+1} + 2(1+\alpha)u_{n-2,j+1} - \alpha u_{n-1,j+1} &= \alpha u_{n-3,j} + 2(1-\alpha)u_{n-2,j} + \alpha u_{n-1,j} \\
-\alpha u_{n-2,j+1} + 2(1+\alpha)u_{n-1,j+1} - \alpha u_{n,j+1} &= \alpha u_{n-2,j} + 2(1-\alpha)u_{n-1,j} + \alpha u_{n,j}
\end{aligned}$$

Similar to the case of Backward Euler, this set of equations can be rearranged as follows

$$\begin{aligned}
2(1+\alpha)u_{1,j+1} - \alpha u_{2,j+1} &= \alpha u_{0,j} + 2(1-\alpha)u_{1,j} + \alpha u_{2,j} + \alpha u_{0,j+1} \\
-\alpha u_{1,j+1} + 2(1+\alpha)u_{2,j+1} - \alpha u_{3,j+1} &= \alpha u_{1,j} + 2(1-\alpha)u_{2,j} + \alpha u_{3,j} \\
&\vdots \\
-\alpha u_{n-3,j+1} + 2(1+\alpha)u_{n-2,j+1} - \alpha u_{n-1,j+1} &= \alpha u_{n-3,j} + 2(1-\alpha)u_{n-2,j} + \alpha u_{n-1,j} \\
-\alpha u_{n-2,j+1} + 2(1+\alpha)u_{n-1,j+1} &= \alpha u_{n-2,j} + 2(1-\alpha)u_{n-1,j} + \alpha u_{n,j} + \alpha u_{n,j+1}
\end{aligned}$$

We can define the vector $\mathbf{b}_j$ to hold the values on the right side of each of these equations:

$$\mathbf{b}_j = \begin{bmatrix} \alpha u_{0,j} + 2(1-\alpha)u_{1,j} + \alpha u_{2,j} + \alpha u_{0,j+1} \\ \alpha u_{1,j} + 2(1-\alpha)u_{2,j} + \alpha u_{3,j} \\ \vdots \\ \alpha u_{n-3,j} + 2(1-\alpha)u_{n-2,j} + \alpha u_{n-1,j} \\ \alpha u_{n-2,j} + 2(1-\alpha)u_{n-1,j} + \alpha u_{n,j} + \alpha u_{n,j+1} \end{bmatrix} \tag{24}$$

Using this and the definition of the vector $\mathbf{v}_j$ from (9), we can write the set of equations as

$$\mathbf{A}\mathbf{v}_j = \mathbf{b}_j$$

where the matrix $\mathbf{A}$ is a tridiagonal matrix with $2(1+\alpha)$ on the diagonal and $-\alpha$ directly above and below the diagonal.