Finite Differences Project
(Parabolic PDE)

Minxuan Song
Guancheng Qiu
Zijun Lin

1. Introduction

Finite difference methods (FDM) have been known for its ability to solve difference equations. As is evident, different finite different methods yield different results, which largely results from the differences in input selection. This paper focuses heavily on solving practical problems using different finite difference methods and the error analysis that arises when comparing the results. Specifically, this problem will focus on solving the famous one-dimensional heat problem, and the problem that's considered an extension of it: the two-dimensional heat problem. Note that an analytical solution for both of the problem will be provided, in order to illustrate a better understanding to the problem.

2. The one-dimensional heat problem

Consider the one dimensional heat problem: find u(x,t) such that $\frac{\partial \mathrm{u}}{\partial t} = \frac{\partial^2 \mathrm{u}}{\partial x^2}$. Suppose the following condition is known: u(x,0) = sin(πx) for all x such that $0 \leq x \leq 1$, and u(0, t) = 0 for all t such that $0 \leq t \leq 1$, and u(1,t) = 0 for all t such that $0 \leq t \leq 1$.

Analytical Solution:

By firstly doing an analytical solution, we can familiarize ourselves with the essence of the problem. The solution not only informs us of the framework of this problem but also provides us with results that we can later compare with the results we get when we're trying to solve this problem using numerical methods.

First, assume the solution has the form:

$$\mathrm{u(x, t) = v(x)w(t)}$$

The original equation then becomes:

$$\frac{1}{w(t)}\frac{\mathrm{dw(t)}}{dt} = \frac{1}{\mathrm{v(x)}}\frac{d^2v(x)}{dx^2} = -\lambda$$

where λ is an arbitrary constant. This gives us two ODE's:

$$\frac{\mathrm{dw(t)}}{dt} + \lambda w(t) = 0, \quad \frac{d^2v(x)}{dx^2} + \lambda\mathrm{v(x)} = 0.$$

Given initial values $u(x, 0) = \sin(\pi x)$, boundaries $u(0{,}t) = 0,\ \mathrm{u(1,t)} = 0$, we have

$$\mathrm{v(x)w(0)} = \sin(\pi x), \quad v(0)u(t) = 0, \quad v(1)u(t) = 0.$$

For a non-trivial solution, consider boundaries to be

$$v(0) = 0, \quad v(1) = 0.$$

Let us first look at the spatial problem

$$\frac{d^2v(x)}{dx^2} + \lambda\mathrm{v(x)} = 0, \quad v(0) = 0, \quad v(1) = 0.$$

The solution is of the form:

$$\text{v(x)} = c_1\cos\left(\sqrt{\lambda}x\right) + c_2\sin\left(\sqrt{\lambda}x\right)$$

which gives:

$$v(0) = 0 = c_1,\ \ v(1) = 0 = c_2\sin\left(\sqrt{\lambda}\right).$$

For a non-trivial solution, we have $\sin\left(\sqrt{\lambda}\right) = 0$, so

$$\lambda = (n\pi)^2,\ \ v(x) = \sin(n\pi x),\ \ n = 1,2,3\ldots$$

For the time problem

$$\frac{\text{dw(t)}}{dt} + \lambda w(t) = 0,\ \ w(0) = 1,$$

the solution is

$$w(t) = \mathrm{e}^{-\lambda \mathrm{t}}$$

which gives the solution to the original equation

$$\text{u(x, t)} = \sin(\pi x)\mathrm{e}^{-\pi^2 \mathrm{t}}$$

as we let $n = 1$.

Hence, this problem is solved analytically.

Numerical Solutions:

We take on this problem with three different finite difference methods: the Explicit method, the Implicit method and the Crank-Nicolson method. Note that we're already provided with the knowledge that for $\frac{\partial \text{u}}{\partial t} = \frac{\partial^2 \text{u}}{\partial x^2}$, u(x,0) = sin(πx) for all x such that 0 ≤ x ≤1, and u(0,t) = 0 for all t such that 0 ≤ t ≤1, and u(1,t) = 0 for all t such that 0 ≤ t ≤ 1. This will provide us with interior nodes to construct the linear system that we'll be using to solve this problem.

In terms of the Explicit method, we'll use Euler's method to approach this problem.
For LHS, we have the forward difference formula:

$$\frac{\partial \mathrm{u}(\mathrm{x},\mathrm{t})}{\partial t}\Big|_{t=t_j} = \frac{\mathrm{u}\left(x, t_{j+1}\right) - \mathrm{u}\left(\mathrm{x}, t_j\right)}{\Delta t}$$

and for RHS, we have the centered three-point formula:

$$\frac{\partial^2 \mathrm{u}(\mathrm{x},\mathrm{t})}{\partial x^2}\Big|_{x=x_i} = \frac{\mathrm{u}\left(x_{i-1},\mathrm{t}\right) - 2\mathrm{u}\left(x_i,\mathrm{t}\right) + \mathrm{u}\left(x_{i+1},\mathrm{t}\right)}{\Delta x^2}$$

Now by ignoring error terms and equating the two, we have:

$$\frac{\mathrm{u}\left(x_i, t_{j+1}\right) - \mathrm{u}\left(x_i, t_j\right)}{\Delta t} = \frac{\mathrm{u}\left(x_{i-1}, t_j\right) - 2\mathrm{u}\left(x_i, t_j\right) + \mathrm{u}\left(x_{i+1}, t_j\right)}{\Delta x^2}$$

Now simplifying this equation:

$$\mathrm{u}\left(x_i, t_{j+1}\right) = \frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i-1}, t_j\right) + (1 - 2\frac{\Delta t}{\Delta x^2})\left(x_i, t_j\right) + \frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i+1}, t_j\right)$$

Since u(x,0) = sin(πx), and we already know the boundaries $(0,t) = 0$, $\mathrm{u}(1,\mathrm{t}) = 0$, it will be very straightforward to solve for $\mathrm{u}\left(x_i, t_{j+1}\right)$.

Therefore, we have a linear system as this:

$$\begin{pmatrix} \frac{\Delta t}{\Delta x^2} & 1-\frac{2\Delta t}{\Delta x^2} & \frac{\Delta t}{\Delta x^2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{\Delta t}{\Delta x^2} & 1-\frac{2\Delta t}{\Delta x^2} & \frac{\Delta t}{\Delta x^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{\Delta t}{\Delta x^2} & 1-\frac{2\Delta t}{\Delta x^2} & \frac{\Delta t}{\Delta x^2} \end{pmatrix} \begin{pmatrix} u(x_0,t_j) \\ u(x_1,t_j) \\ u(x_2,t_j) \\ u(x_3,t_j) \\ \vdots \\ u(x_n,t_j) \end{pmatrix} = \begin{pmatrix} u(x_1,t_{j+1}) \\ u(x_2,t_{j+1}) \\ \vdots \\ u(x_{n-1},t_{j+1}) \end{pmatrix}$$

Then, solving this should yields an answer to the explicit method.
Matlab implementation for this method:

```
function u = Explicit1D(u, M, N, dt, dx)
    % Construct coefficient matrix
    B = zeros(N-2, N);
    tempDiag1 = 1 - 2*dt/dx^2;
```

```
        tempDiag2 = dt/dx^2;
        for i = 1 : N-2
            B(i,i) = tempDiag2;
            B(i,i+1) = tempDiag1;
            B(i,i+2) = tempDiag2;
        end

        for i = 2 : M
            a = B * (u(i-1,:))';
            u(i, 2:N-1) = a';
        end
    end
```

In terms of Implicit method, we can also take Euler's method to take on this problem.

For LHS, we have:

$$\frac{\partial \mathrm{u}(\mathrm{x},\mathrm{t})}{\partial t}\Big|_{t=t_j} = \frac{\mathrm{u}\left(x, t_j\right) - \mathrm{u}\left(\mathrm{x}, t_{j-1}\right)}{\Delta t}$$

This is the backward difference formula.

For RHS, we have:

$$\frac{\partial^2 \mathrm{u}(\mathrm{x},\mathrm{t})}{\partial x^2}\Big|_{x=x_i} = \frac{\mathrm{u}\left(x_{i-1},\mathrm{t}\right) - 2\mathrm{u}\left(x_i,\mathrm{t}\right) + \mathrm{u}\left(x_{i+1},\mathrm{t}\right)}{\Delta x^2}$$

This is the centered three-point formula.

Then, by ignoring error terms and equating the two equations, we have:

$$\frac{\mathrm{u}\left(x_i, t_j\right) - \mathrm{u}\left(x_i, t_{j-1}\right)}{\Delta t} = \frac{\mathrm{u}\left(x_{i-1}, t_j\right) - 2\mathrm{u}\left(x_i, t_j\right) + \mathrm{u}\left(x_{i+1}, t_j\right)}{\Delta x^2}$$

By simplifying this equation, we have:

$$\mathrm{u}\left(x_i, t_{j-1}\right) = \left(\frac{2\Delta t}{\Delta x^2} + 1\right)\mathrm{u}\left(x_i, t_j\right) - \frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i-1}, t_j\right) - \frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i+1}, t_j\right)$$

Now, given initial values $u(x, 0) = \sin(\pi x)$ and the boundaries $u(0,t) = 0$, u(1,t) = 0, we can construct the linear system that we need to solve for $u\left(x, t_j\right)$.

$$\begin{pmatrix} -\frac{\Delta t}{\Delta x^2} & \frac{2\Delta t}{\Delta x^2}+1 & -\frac{\Delta t}{\Delta x^2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & -\frac{\Delta t}{\Delta x^2} & \frac{2\Delta t}{\Delta x^2}+1 & -\frac{\Delta t}{\Delta x^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -\frac{\Delta t}{\Delta x^2} & \frac{2\Delta t}{\Delta x^2}+1 & -\frac{\Delta t}{\Delta x^2} \end{pmatrix}$$

$$\begin{pmatrix} u(x_0, t_j) \\ u(x_1, t_j) \\ u(x_2, t_j) \\ u(x_3, t_j) \\ \vdots \\ u(x_n, t_j) \end{pmatrix} = \begin{pmatrix} u(x_1, t_{j-1}) \\ u(x_2, t_{j-1}) \\ \vdots \\ u(x_{n-1}, t_{j-1}) \end{pmatrix}$$

Note that n is the number of partitions. Since we already know that when j – 1 = 0, u(x,0) = $\sin(\pi x)$. So by starting a loop from j = 1, and with the boundaries values $x_0 = 0$ and $x_n = 1$, we can always have $\left(x_0, t_j\right) = u\left(x_n, t_j\right) = 0$.

Matlab implementation for this method:

```
function u = Implicit1D(u, M, N, dt, dx)
    % Construct coefficient matrix
    A = zeros(N-2, N);
    tempDiag1 = 1 + 2*dt/dx^2;
    tempDiag2 = -dt/dx^2;
    for i = 1 : N-2
        A(i,i) = tempDiag2;
        A(i,i+1) = tempDiag1;
        A(i,i+2) = tempDiag2;
    end

    for i = 2 : M
        b = A \ (u(i-1, 2 : N-1))';
        u(i, :) = b';
    end
end
```

Using Crank-Nicolson:

For LHS, we have:

$$\frac{\partial \mathrm{u(x,t)}}{\partial t}|_{t=t_j} = \frac{\mathrm{u}\left(x,t_{j+1}\right)-\mathrm{u}\left(\mathrm{x},t_j\right)}{\Delta t}$$

This is the forward difference formula.

Then, for RHS, we have:

$$\frac{\partial^2 \mathrm{u(x,t)}}{\partial x^2}|_{x=x_i,\ t=t_j} = \frac{1}{2}\left(\frac{\mathrm{u}\left(x_{i-1},t_{j+1}\right)-2\mathrm{u}\left(x_i,t_{j+1}\right)+\mathrm{u}\left(x_{i+1},t_{j+1}\right)}{\Delta x^2}+\frac{\mathrm{u}\left(x_{i-1},t_j\right)-2\mathrm{u}\left(x_i,t_j\right)+\mathrm{u}\left(x_{i+1},t_j\right)}{\Delta x^2}\right)$$

Note that different from the explicit method and the implicit method, this is the centered three-point formula applied twice.

Now, by ignoring error terms and equating two of the equations, we have:

$$\frac{\mathrm{u}\left(x_i,t_{j+1}\right)-\mathrm{u}\left(x_i,t_j\right)}{\Delta t} = \frac{1}{2}\left(\frac{\mathrm{u}\left(x_{i-1},t_{j+1}\right)-2\mathrm{u}\left(x_i,t_{j+1}\right)+\mathrm{u}\left(x_{i+1},t_{j+1}\right)}{\Delta x^2}+\frac{\mathrm{u}\left(x_{i-1},t_j\right)-2\mathrm{u}\left(x_i,t_j\right)+\mathrm{u}\left(x_{i+1},t_j\right)}{\Delta x^2}\right)$$

Further simplifying this:

$$-\frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i-1},t_{j+1}\right)+(2+\frac{2\Delta t}{\Delta x^2})\mathrm{u}\left(x_i,t_{j+1}\right)-\frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i+1},t_{j+1}\right)=\frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i-1},t_j\right)+(2-\frac{2\Delta t}{\Delta x^2})\mathrm{u}\left(x_i,t_j\right)+\frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i+1},t_j\right)$$

Therefore, we can construct a linear system accordingly, for each j:

$$\begin{pmatrix} -\frac{\Delta t}{\Delta x^2} & 2+\frac{2\Delta t}{\Delta x^2} & -\frac{\Delta t}{\Delta x^2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & -\frac{\Delta t}{\Delta x^2} & 2+\frac{2\Delta t}{\Delta x^2} & -\frac{\Delta t}{\Delta x^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -\frac{\Delta t}{\Delta x^2} & 2+\frac{2\Delta t}{\Delta x^2} & -\frac{\Delta t}{\Delta x^2} \end{pmatrix} \begin{pmatrix} u\left(x_0,t_{j+1}\right) \\ u\left(x_1,t_{j+1}\right) \\ u\left(x_2,t_{j+1}\right) \\ u\left(x_3,t_{j+1}\right) \\ \vdots \\ u\left(x_n,t_{j+1}\right) \end{pmatrix} = \begin{pmatrix} \frac{\Delta t}{\Delta x^2} & 2-\frac{2\Delta t}{\Delta x^2} & \frac{\Delta t}{\Delta x^2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{\Delta t}{\Delta x^2} & 2-\frac{2\Delta t}{\Delta x^2} & \frac{\Delta t}{\Delta x^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{\Delta t}{\Delta x^2} & 2-\frac{2\Delta t}{\Delta x^2} & \frac{\Delta t}{\Delta x^2} \end{pmatrix} \begin{pmatrix} u\left(x_0,t_j\right) \\ u\left(x_1,t_j\right) \\ u\left(x_2,t_j\right) \\ u\left(x_3,t_j\right) \\ \vdots \\ u\left(x_n,t_j\right) \end{pmatrix}$$

Matlab implementation:

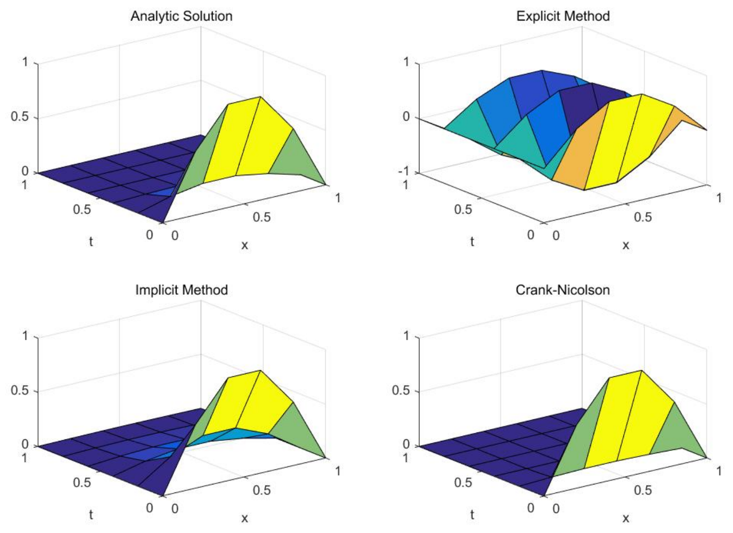
```
function u = CN1D(u, M, N, dt, dx)
    % Construct coefficient matrix
    A = zeros(N-2, N);
    B = zeros(N-2, N);

    tempDiagA1 = 2 + 2*dt/dx^2;
    tempDiagA2 = -dt/dx^2;
    for i = 1 : N-2
        A(i,i) = tempDiagA2;
        A(i,i+1) = tempDiagA1;
        A(i,i+2) = tempDiagA2;
    end

    tempDiagB1 = 2 - 2*dt/dx^2;
    tempDiagB2 = dt/dx^2;
    for i = 1 : N-2
        B(i,i) = tempDiagB2;
        B(i,i+1) = tempDiagB1;
        B(i,i+2) = tempDiagB2;
    end

    for i = 2 : M
        b = A \ (B * (u(i-1, :))');
        u(i, :) = b';
    end
end
```

Now, we have solved this problem using all three numerical methods. The differences, however, that exist in the results that each of the methods yields, will lead to a better understanding of the numerical methods that we have chosen to use, which we'll elaborate further in the next section, error analysis.

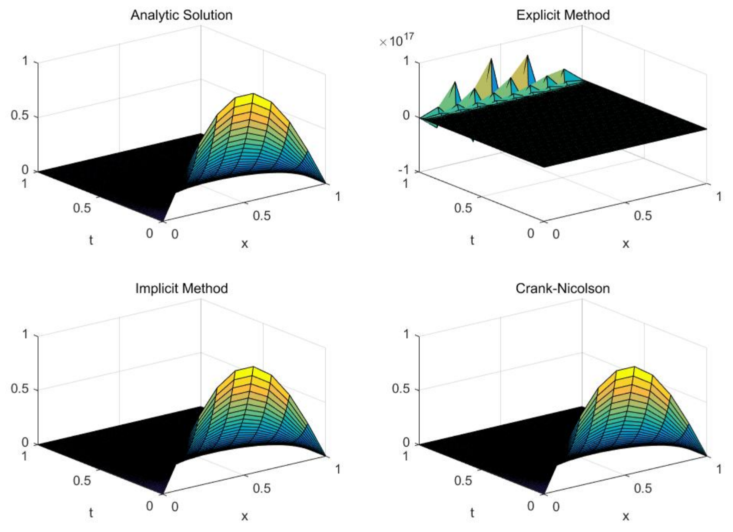
Error Analysis:

First of all we will look at the results for each methods using different FDMs.

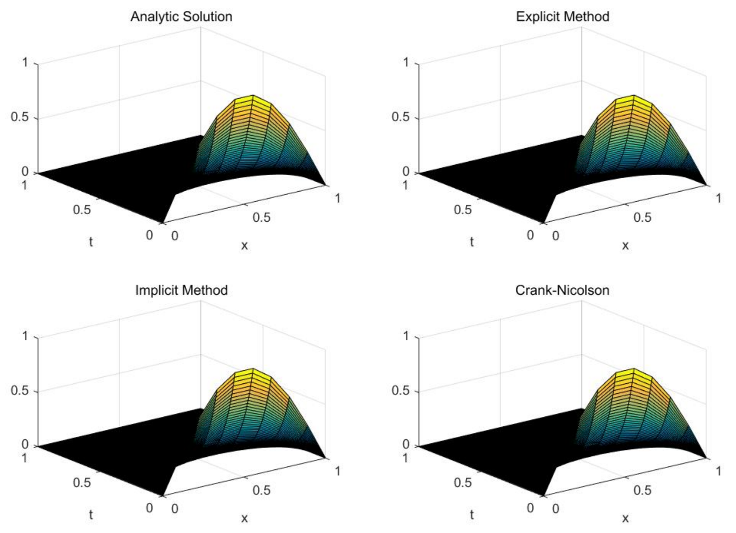
Using 6x, 6t nodes:



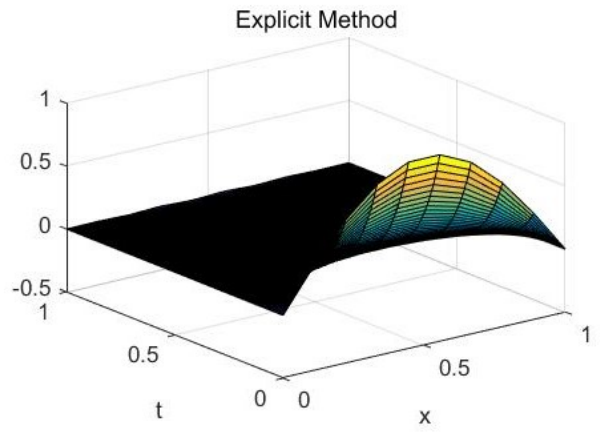
Using 10x, 100t nodes:



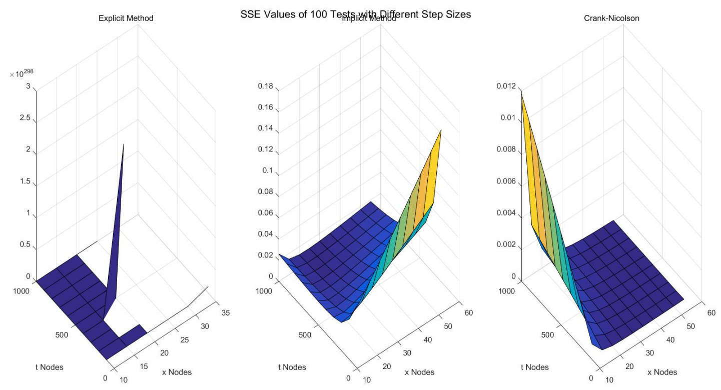
Using 10x, 200t nodes:



Note on convergence of the Explicit Method: in this problem in particular, it is guaranteed that no point in the solution should have a value that is negative. Since ranges of x and t are equal, $\dfrac{\Delta t}{\Delta x^2} \leq \dfrac{1}{2}\ iff\ 2(N-1)^2 \leq M-1$, where M denotes number of t nodes and N denotes number of x nodes. However, the Explicit Method solution starts converging as soon as $\dfrac{\Delta t}{\Delta x^2} \approx 0.6$, although producing a quite inaccurate solution, as in the case where 10 x nodes and 135 t nodes are used ($\dfrac{\Delta t}{\Delta x^2} = 0.6045$):



Now, for the other two methods, we ran 100 tests on each method, where number of x nodes can be any of 10, 15...55, and number of t nodes can be any of 100, 200...1000. We calculate SSE (sum of squared errors) using the analytic solution. The following graphs showing SSE of all tests are obtained.



In terms of SSE, we can see that it is difficult to make Explicit Method converge while maintaining observation of effects of number of x nodes, so we will focus more on performance of Implicit Method and Crank-Nicolson Method.

Implicit Method generates significantly higher errors when number of x nodes increases while there are not many t nodes. We believe this is due to accumulation of round-off errors. Crank-Nicolson Method, on the other hand, is barely affected by round-off errors in this situation.

Implicit Method also generates slightly higher errors when there are many t nodes and not many x nodes. As number of t nodes decide number of iterations, initial errors due to small number of x nodes are magnified in these significant amounts of iterations. On the other hand, Crank-Nicolson Method is significantly affected in like situation. Presumably such behavior can be attributed to same error magnification problem.

In general, Crank-Nicolson Method produces significantly smaller errors. In the tests conducted, $S\bar{S}E = 0.0331$ for Implicit Method, while $S\bar{S}E = 9.58e - 04$ for Crank-Nicolson Method, the latter about 10^2 times smaller than the former.

Matlab implementation for heat difference graph plotting:

```
%initialize matrix with initial and boundary
values
M = 135;  %number of nodes on time
N = 10;  %number of nodes on space
u = zeros(M, N);
dx = 1 / (N-1);
dt = 1 / (M-1);

%analytic solution
x = 0 : dx : 1;
t = 0 : dt : 1;
uExact = zeros(M, N);
for i = 1 : M
    for j = 1 : N
        uExact(i, j) = sin(pi * x(j)) * exp(-
pi^2*t(i));
    end
end
figure('Position', [300 100 900 600])
subplot(2,2,1)
surf(x, t, uExact)
title('Analytic Solution')
xlabel('x')
ylabel('t')

%initial values
for i = 0 : N-1
    u(1, i+1) = sin(pi * i * dx);
end

%calculate with all 3 methods and plot
uExplicit = Explicit1D(u, M, N, dt, dx); %Note
the problem with r <= 1/2
subplot(2,2,2)
surf(x, t, uExplicit)
title('Explicit Method')
xlabel('x')
ylabel('t')

uImplicit = Implicit1D(u, M, N, dt, dx);
subplot(2,2,3)
surf(x, t, uImplicit)
title('Implicit Method')
xlabel('x')
ylabel('t')

uCN = CN1D(u, M, N, dt, dx);
subplot(2,2,4)
surf(x, t, uCN)
title('Crank-Nicolson')
xlabel('x')
ylabel('t')
```

Matlab implementation for error display:

```
tests = 10;
Ma = linspace(100, 1000, tests);  %number of
nodes on time
```

```
Na = linspace(10, 55, tests);  %number of
nodes on space
explicitSSE = zeros(tests);
implicitSSE = zeros(tests);
CNSSE = zeros(tests);
for m = 1 : tests
    for n = 1 : tests

    %initialize matrix with initial and
boundary values
    M = Ma(m);
    N = Na(n);
    u = zeros(M, N);
    dx = 1 / (N-1);
    dt = 1 / (M-1);

    x = 0 : dx : 1;
    t = 0 : dt : 1;
    uExact = zeros(M, N);
    for i = 1 : M
        for j = 1 : N
            uExact(i, j) = sin(pi * x(j)) *
exp(-pi^2*t(i));
        end
    end

    %initial values
    for i = 0 : N-1
        u(1, i+1) = sin(pi * i * dx);
    end

    uExplicit = Explicit1D(u, M, N, dt, dx);
%Note the problem with r <= 1/2
    explicitSSE(m, n) = SSE(uExplicit,
uExact);

    uImplicit = Implicit1D(u, M, N, dt, dx);
    implicitSSE(m, n) = SSE(uImplicit,
uExact);

    uCN = CN1D(u, M, N, dt, dx);
    CNSSE(m, n) = SSE(uCN, uExact);
    end
end

figure('Position', [300 100 900 600])
suptitle('SSE Values of 100 Tests with
Different Step Sizes')
subplot(1,3,1)
surf(Na, Ma, explicitSSE)
title('Explicit Method')
xlabel('x Nodes')
ylabel('t Nodes')
subplot(1,3,2)
surf(Na, Ma, implicitSSE)
title('Implicit Method')
xlabel('x Nodes')
ylabel('t Nodes')
subplot(1,3,3)
surf(Na, Ma, CNSSE)
title('Crank-Nicolson')
xlabel('x Nodes')
ylabel('t Nodes')
```

3. Extension: The two-dimensional heat problem

With all the information reduced from the one-dimensional heat problem, we can take our research to the next level. Consider the following problem: a square plate of steel is 15 cm on a side. Initially, all points are at 0ºC. If two adjacent sides are suddenly brought to 100ºC and held at that temperature. The plate is insulated on its flat surfaces. so that heat flows only in the x and y directions. We are given that k = 0.13, ϕ = 7.8, c = 0.11 in c.g.s units. Now, solve $\frac{\partial \mathrm{u}}{\partial t} = \frac{\mathrm{k}}{c\phi}(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial \mathrm{y}^2})$

Analytical Solution:

We can solve this problem analytically in the same fashion that we did for the one-dimensional heat problem. Note that because the situation in terms of the number of parameters becomes significantly harder, the interior nodes we use increase in number. In fact, if we are to plot everything on a graph, it is straightforward to observe that this problem largely differs from the one-dimensional heat problem because the graph turns from 2D to 3D. Also, our initial condition changes. It has now become: u $\left(x, y, 0\right) = 0$ except for boundaries. For boundaries, it is: $u\left(0, y, t\right) = 100$, $\mathrm{u(x, 0,t)} = 100$.
$0 \leq \mathrm{x} \leq 15, 0 \leq \mathrm{y} \leq 15, 0 \leq \mathrm{t} \leq 20$. Therefore, using the same reduction logic, we can arrive at a satisfying expression that is similar in composition to the ones that we came up with for the one-dimensional heat problem.

Numerical Solution:

For numerical methods, we'll be using the same three methods (explicit, implicit and Crank-Nicolson) that we applied to the one-dimensional heat problem.

In terms of the Explicit method, we'll also use Euler's method to approach this problem.
For LHS, we have the forward difference formula:

$$\frac{\partial \mathrm{u}(\mathrm{x},\mathrm{y},\mathrm{t})}{\partial t}\Big|_{t=t_k} = \frac{\mathrm{u}\left(x, y, t_{k+1}\right) - \mathrm{u}\left(\mathrm{x}, y, t_k\right)}{\Delta t}$$

and for RHS, we have the centered three-point formula:

$$\frac{k}{c\varphi}\left(\frac{\partial^2 \mathrm{u}(\mathrm{x},\mathrm{y},\mathrm{t})}{\partial x^2}\Big|_{x=x_i} + \frac{\partial^2 \mathrm{u}(\mathrm{x},\mathrm{y},\mathrm{t})}{\partial y^2}\Big|_{y=y_j}\right) = \frac{k}{c\varphi}\frac{\mathrm{u}\left(x_{i-1}, y_j, \mathrm{t}\right) - 2\mathrm{u}\left(x_i, y_j, \mathrm{t}\right) + \mathrm{u}\left(x_{i+1}, y_j, \mathrm{t}\right)}{\Delta x^2} + \frac{k}{c\varphi}\frac{\mathrm{u}\left(x_i, y_{j-1}, \mathrm{t}\right) - 2\mathrm{u}\left(x_i, y_j, \mathrm{t}\right) + \mathrm{u}\left(x_i, y_{j+1}, \mathrm{t}\right)}{\Delta y^2}$$

Now by ignoring error terms and equating the two, we have:

$$\frac{\mathrm{u}\left(x_i, y_j, t_{k+1}\right) - \mathrm{u}\left(x_i, y_j, t_k\right)}{\Delta t} = \frac{k}{c\varphi}\frac{\mathrm{u}\left(x_{i-1}, y_j, t_k\right) - 2\mathrm{u}\left(x_i, y_j, t_k\right) + \mathrm{u}\left(x_{i+1}, y_j, t_k\right)}{\Delta x^2} + \frac{k}{c\varphi}\frac{\mathrm{u}\left(x_i, y_{j-1}, t_k\right) - 2\mathrm{u}\left(x_i, y_j, t_k\right) + \mathrm{u}\left(x_i, y_{j+1}, t_k\right)}{\Delta y^2}$$

Now simplifying this equation:

$$\mathrm{u}\left(x_i, y_j, t_{k+1}\right) = \frac{k}{c\varphi}\left(\frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i-1}, y_j, t_k\right) + \left(\frac{c\varphi}{k} - \frac{2\Delta t}{\Delta x^2} - \frac{2\Delta t}{\Delta y^2}\right)\mathrm{u}\left(x_i, y_j, t_k\right) + \frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i+1}, y_j, t_k\right) + \frac{\Delta t}{\Delta y^2}\mathrm{u}\left(x_i, y_{j-1}, t_k\right) + \frac{\Delta t}{\Delta y^2}\mathrm{u}\left(x_i, y_{j+1}, t_k\right)\right)$$

Given $u\left(x, y, 0\right) = 0$ except for the boundaries, and we know that for the boundaries,
$u\left(0, y, t\right) = 100$, $\mathrm{u}(\mathrm{x}, 0, \mathrm{t}) = 100$, it is straightforward to solve for $\mathrm{u}\left(x_i, y_j, t_{k+1}\right)$.

Therefore, we have a linear system as this:

$$\frac{k}{c\varphi}\begin{pmatrix} \frac{\Delta t}{\Delta x^2} & \frac{c\varphi}{k} - \frac{2\Delta t}{\Delta x^2} - \frac{2\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta x^2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{\Delta t}{\Delta x^2} & \frac{c\varphi}{k} - \frac{2\Delta t}{\Delta x^2} - \frac{2\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta x^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{\Delta t}{\Delta x^2} & \frac{c\varphi}{k} - \frac{2\Delta t}{\Delta x^2} - \frac{2\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta x^2} \end{pmatrix} *$$

$$\begin{pmatrix} u(x_0, y_j, t_k) \\ u(x_1, y_j, t_k) \\ u(x_2, y_j, t_k) \\ u(x_3, y_j, t_k) \\ \vdots \\ u(x_n, y_j, t_k) \end{pmatrix} +$$

$$\frac{k}{c\varphi} \begin{pmatrix} \frac{\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta y^2} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta y^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & \frac{\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta y^2} \end{pmatrix} \begin{pmatrix} u(x_1, y_{j-1}, t_k) \\ u(x_1, y_{j+1}, t_k) \\ u(x_2, y_{j-1}, t_k) \\ u(x_2, y_{j+1}, t_k) \\ \vdots \\ u(x_{n-1}, y_{j-1}, t_k) \\ u(x_{n-1}, y_{j+1}, t_k) \end{pmatrix}$$

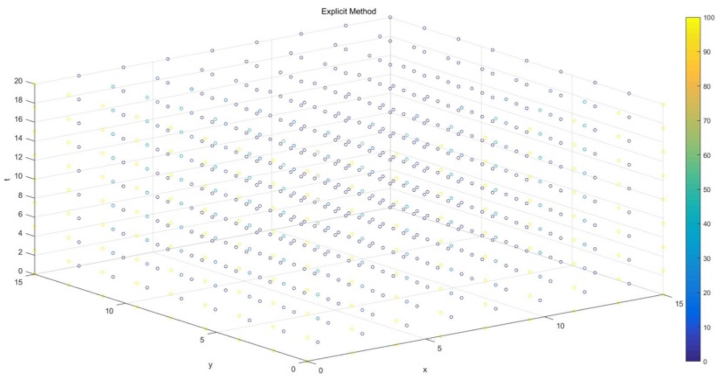
$$= \begin{pmatrix} u\left(x_1, y_j, t_{k+1}\right) \\ u\left(x_2, y_j, t_{k+1}\right) \\ \vdots \\ u\left(x_{n-1}, y_j, t_{k+1}\right) \end{pmatrix}$$

Then, solving this should yields an answer to the explicit method.

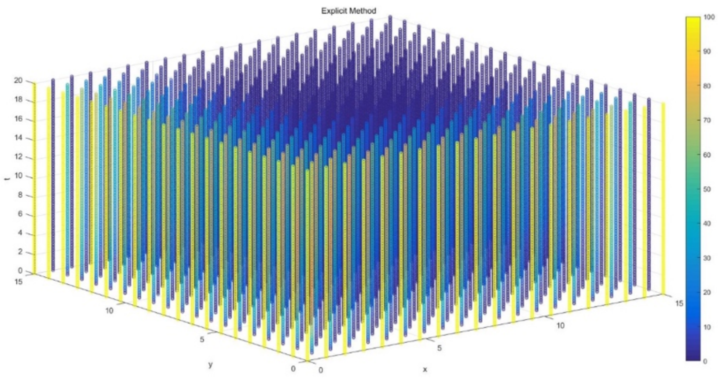
Matlab implementation:

```
function u = Explicit2D(u, c, M, N, dt, dx,
dy)
    % Construct coefficient matrix
    B1 = zeros(N-2, N);
    tempDiag11 = 1 - c * (2*dt/dx^2 + 2*dt/
dy^2);
    tempDiag12 = c * dt/dx^2;
    for i = 1 : N-2
        B1(i,i) = tempDiag12;
        B1(i,i+1) = tempDiag11;
        B1(i,i+2) = tempDiag12;
    end

    B2 = zeros(N-2, 2*N-4);
    tempDiag21 = c * dt/dy^2;
    for i = 1 : N-2
        B2(i, 2*i-1) = tempDiag21;
        B2(i, 2*i) = tempDiag21;
        end

        for i = 1 : M - 1
            for j = 1 : N - 2
                b1 = (u(:, j+1, i));
                b2 = zeros(2*N-4, 1);
                for k = 1 : N - 2
                    b2(2*k-1) = u(k+1, j, i);
                    b2(2*k) = u(k+1, j+2, i);
                end
                a = B1 * b1 + B2 * b2;
                u(2:N-1, j+1, i+1) = a;
            end
        end
    end
```

9 space nodes, 9 t nodes:



20 space nodes, 200 t nodes:



In terms of Implicit method, we can also take Euler's method to take on this problem.

For LHS, we have:

$$\frac{\partial \mathrm{u}(\mathrm{x},\mathrm{y},\mathrm{t})}{\partial t}\Big|_{t=t_k} = \frac{\mathrm{u}(x,y,t_k) - \mathrm{u}(\mathrm{x},y,t_{k-1})}{\Delta t}$$

This is the backward difference formula.

For RHS, we have:

$$\frac{k}{c\varphi}\left(\frac{\partial^2 \mathrm{u}(\mathrm{x},\mathrm{y},\mathrm{t})}{\partial x^2}\Big|_{x=x_i} + \frac{\partial^2 \mathrm{u}(\mathrm{x},\mathrm{y},\mathrm{t})}{\partial y^2}\Big|_{y=y_j}\right) = \frac{k}{c\varphi}\left(\frac{\mathrm{u}(x_{i-1},y_j,\mathrm{t}) - 2\mathrm{u}(x_i,y_j,\mathrm{t}) + \mathrm{u}(x_{i+1},y_j,\mathrm{t})}{\Delta x^2} + \frac{\mathrm{u}(x_i,y_{j-1},\mathrm{t}) - 2\mathrm{u}(x_i,y_j,\mathrm{t}) + \mathrm{u}(x_i,y_{j+1},\mathrm{t})}{\Delta y^2}\right)$$

This is the centered three-point formula.

Then, by ignoring error terms and equating the two equations, we have:

$$\frac{\mathrm{u}(x_i,y_j,t_k) - \mathrm{u}(x_i,y_j,t_{k-1})}{\Delta t} = \frac{k}{c\varphi}\left(\frac{\mathrm{u}(x_{i-1},y_j,t_k) - 2\mathrm{u}(x_i,y_j,t_k) + \mathrm{u}(x_{i+1},y_j,t_k)}{\Delta x^2} + \frac{\mathrm{u}(x_i,y_{j-1},t_k) - 2\mathrm{u}(x_i,y_j,t_k) + \mathrm{u}(x_i,y_{j+1},t_k)}{\Delta y^2}\right)$$

By simplifying this equation, we have:

$$\mathrm{u}(x_i,y_j,t_{k-1}) = \frac{k}{c\varphi}\left(-\frac{\Delta t}{\Delta x^2}\mathrm{u}(x_{i-1},y_j,t_k) + \left(\frac{c\varphi}{k} + \frac{2\Delta t}{\Delta x^2} + \frac{2\Delta t}{\Delta y^2}\right)\mathrm{u}(x_i,y_j,t_k) - \frac{\Delta t}{\Delta x^2}\mathrm{u}(x_{i+1},y_j,t_k) - \frac{\Delta t}{\Delta y^2}\mathrm{u}(x_i,y_{j-1},t_k) - \frac{\Delta t}{\Delta y^2}\mathrm{u}(x_i,y_{j+1},t_k)\right)$$

Now, given $u(x,y,0) = 0$ except for the boundaries, and for the boundaries, we have
$u(0,y,t) = 100$, $\mathrm{u(x,0,t)} = 100$, so we can construct a linear system, for each j and for
each k:

$$\frac{k}{c\varphi}\begin{pmatrix} -\frac{\Delta t}{\Delta x^2} & \frac{c\varphi}{k} + \frac{2\Delta t}{\Delta x^2} + \frac{2\Delta t}{\Delta y^2} & -\frac{\Delta t}{\Delta x^2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & -\frac{\Delta t}{\Delta x^2} & \frac{c\varphi}{k} + \frac{2\Delta t}{\Delta x^2} + \frac{2\Delta t}{\Delta y^2} & -\frac{\Delta t}{\Delta x^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -\frac{\Delta t}{\Delta x^2} & \frac{c\varphi}{k} + \frac{2\Delta t}{\Delta x^2} + \frac{2\Delta t}{\Delta y^2} & -\frac{\Delta t}{\Delta x^2} \end{pmatrix}$$

*

$$\begin{pmatrix} u(x_0, y_j, t_k) \\ u(x_1, y_j, t_k) \\ u(x_2, y_j, t_k) \\ u(x_3, y_j, t_k) \\ \vdots \\ u(x_n, y_j, t_k) \end{pmatrix} - \frac{k}{c\varphi} \begin{pmatrix} \frac{\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta y^2} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta y^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & \frac{\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta y^2} \end{pmatrix} \begin{pmatrix} u(x_1, y_{j-1}, t_k) \\ u(x_1, y_{j+1}, t_k) \\ u(x_2, y_{j-1}, t_k) \\ u(x_2, y_{j+1}, t_k) \\ \vdots \\ u(x_{n-1}, y_{j-1}, t_k) \\ u(x_{n-1}, y_{j+1}, t_k) \end{pmatrix}$$

Using Crank-Nicolson:

For LHS, we have:

$$\frac{\partial \mathrm{u(x, y, t)}}{\partial t}\Big|_{t=t_k} = \frac{\mathrm{u}\left(x, y, t_{k+1}\right) - \mathrm{u}\left(\mathrm{x}, y, t_k\right)}{\Delta t}$$

This is the forward difference formula.

Then, for RHS, we have:

$$\frac{\partial \mathrm{u(x, y, t)}}{\partial t}\Big|_{t=t_k} = \frac{\mathrm{u}\left(x, y, t_{k+1}\right) - \mathrm{u}\left(\mathrm{x}, y, t_k\right)}{\Delta t}$$

Now, by ignoring error terms and equating two of the equations, we have:

$$\frac{\mathrm{u}\left(x_i, y_j, t_{k+1}\right) - \mathrm{u}\left(x_i, y_j, t_k\right)}{\Delta t} = \frac{k}{2c\varphi}\left(\frac{\mathrm{u}\left(x_{i-1}, y_j, t_k\right) - 2\mathrm{u}\left(x_i, y_j, t_k\right) + \mathrm{u}\left(x_{i+1}, y_j, t_k\right)}{\Delta x^2} + \frac{\mathrm{u}\left(x_i, y_{j-1}, t_k\right) - 2\mathrm{u}\left(x_i, y_j, t_k\right) + \mathrm{u}\left(x_i, y_{j+1}, t_k\right)}{\Delta y^2} + \frac{\mathrm{u}\left(x_{i-1}, y_j, t_k\right) - 2\mathrm{u}\left(x_i, y_j, t_k\right) + \mathrm{u}\left(x_{i+1}, y_j, t_k\right)}{\Delta x^2} + \frac{\mathrm{u}\left(x_i, y_{j-1}, t_k\right) - 2\mathrm{u}\left(x_i, y_j, t_k\right) + \mathrm{u}\left(x_i, y_{j+1}, t_k\right)}{\Delta y^2}\right)$$

Further simplifying this:

$$-\frac{k}{2c\varphi}\left(\frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i-1}, y_j, t_{k+1}\right) - \left(\frac{2c\varphi}{k} + \frac{2\Delta t}{\Delta x^2} + \frac{2\Delta t}{\Delta y^2}\right)\mathrm{u}\left(x_i, y_j, t_{k+1}\right) + \frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i+1}, y_j, t_{k+1}\right) + \frac{\Delta t}{\Delta y^2}\mathrm{u}\left(x_i, y_{j-1}, t_{k+1}\right) + \frac{\Delta t}{\Delta y^2}\mathrm{u}\left(x_i, y_{j+1}, t_{k+1}\right)\right) = \frac{k}{2c\varphi}\left(\frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i-1}, y_j, t_k\right) + \left(\frac{2c\varphi}{k} - \frac{2\Delta t}{\Delta x^2} - \frac{2\Delta t}{\Delta y^2}\right)\mathrm{u}\left(x_i, y_j, t_k\right) + \frac{\Delta t}{\Delta x^2}\mathrm{u}\left(x_{i+1}, y_j, t_k\right) + \frac{\Delta t}{\Delta y^2}\mathrm{u}\left(x_i, y_{j-1}, t_k\right) + \frac{\Delta t}{\Delta y^2}\mathrm{u}\left(x_i, y_{j+1}, t_k\right)\right)$$

Therefore, we can construct a linear system accordingly, for each j and each k:

$$-\frac{k}{2c\varphi}\begin{pmatrix} \frac{\Delta t}{\Delta x^2} & -\left(\frac{2c\varphi}{k} + \frac{2\Delta t}{\Delta x^2} + \frac{2\Delta t}{\Delta y^2}\right) & \frac{\Delta t}{\Delta x^2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{\Delta t}{\Delta x^2} & -\left(\frac{2c\varphi}{k} + \frac{2\Delta t}{\Delta x^2} + \frac{2\Delta t}{\Delta y^2}\right) & \frac{\Delta t}{\Delta x^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{\Delta t}{\Delta x^2} & -\left(\frac{2c\varphi}{k} + \frac{2\Delta t}{\Delta x^2} + \frac{2\Delta t}{\Delta y^2}\right) & \frac{\Delta t}{\Delta x^2} \end{pmatrix}$$

*

$$\begin{pmatrix} u\left(x_0, y_j, t_{k+1}\right) \\ u\left(x_1, y_j, t_{k+1}\right) \\ u\left(x_2, y_j, t_{k+1}\right) \\ u\left(x_3, y_j, t_{k+1}\right) \\ \vdots \\ u\left(x_n, y_j, t_{k+1}\right) \end{pmatrix} - \frac{k}{2c\varphi}\begin{pmatrix} \frac{\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta y^2} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta y^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & \frac{\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta y^2} \end{pmatrix}\begin{pmatrix} u(x_1, y_{j-1}, t_{k+1}) \\ u(x_1, y_{j+1}, t_{k+1}) \\ u(x_2, y_{j-1}, t_{k+1}) \\ u(x_2, y_{j+1}, t_{k+1}) \\ \vdots \\ u(x_{n-1}, y_{j-1}, t_{k+1}) \\ u(x_{n-1}, y_{j+1}, t_{k+1}) \end{pmatrix} = \frac{k}{2c\varphi}\begin{pmatrix} \frac{\Delta t}{\Delta x^2} & \frac{2c\varphi}{k} - \frac{2\Delta t}{\Delta x^2} - \frac{2\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta x^2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{\Delta t}{\Delta x^2} & \frac{2c\varphi}{k} - \frac{2\Delta t}{\Delta x^2} - \frac{2\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta x^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{\Delta t}{\Delta x^2} & \frac{2c\varphi}{k} - \frac{2\Delta t}{\Delta x^2} - \frac{2\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta x^2} \end{pmatrix}\begin{pmatrix} u(x_0, y_j, t_k) \\ u(x_1, y_j, t_k) \\ u(x_2, y_j, t_k) \\ u(x_3, y_j, t_k) \\ \vdots \\ u(x_n, y_j, t_k) \end{pmatrix} + \frac{k}{2c\varphi}\begin{pmatrix} \frac{\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta y^2} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta y^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & \frac{\Delta t}{\Delta y^2} & \frac{\Delta t}{\Delta y^2} \end{pmatrix}\begin{pmatrix} u(x_1, y_{j-1}, t_k) \\ u(x_1, y_{j+1}, t_k) \\ u(x_2, y_{j-1}, t_k) \\ u(x_2, y_{j+1}, t_k) \\ \vdots \\ u(x_{n-1}, y_{j-1}, t_k) \\ u(x_{n-1}, y_{j+1}, t_k) \end{pmatrix}$$

Solving this linear system should give us a satisfying result that is based on the Crank-Nicolson method.

Error Analysis:

Using the same logic that we applied in the one-dimensional heat problem, we can obtain the error analysis for this problem in the same fashion.

4. Conclusion

By solving this heat equation problem, we have gained some insights about the three finite difference methods. We consider Explicit Method and Implicit Method to be both efficient, but the former is less favored due to its stability constraint. Crank-Nicolson Method is considered to be better than Implicit Method in this problem: while it is not as efficient, it is less affected by round-off errors and produce significantly smaller errors in general.