PROJECT B- DIFFUSION EQUATION (B02-5)

Scientific Computing

Reetz, Kimberly D
1262537

Abstract

The diffusion equation is a parabolic partial differential equation requiring an initial condition and two boundary conditions for each dimension in space. In this report a variation of the two dimensional diffusion equation will be solved numerically using the explicit method discretization and the alternating direction implicit method discretization. The algorithms for implementing each of these methods on MATLAB is provided. Verification and validation of the algorithms as well as the results produced are also documented. The two dimensional diffusion equation to be solved is of the form described in equation 1. The initial condition for this equation is of the form presented in equation 2. The two dimensional diffusion equation solved in this report has mixed boundary conditions described as the Dirichlet boundary conditions portrayed in equations 3-4 and the Neumann boundary conditions in the form of equations 5-6.

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

$$t>0,$$

$$a_x=0<x<b_x=2\pi,$$

$$a_y=0<y<b_y=2\pi$$

Initial Condition:

$$u(x,y,t=0)= u_0(x,y)=0 \quad (2)$$

Dirichlet Boundary Conditions:

$$u(x=a_x,y)=\emptyset_{ab}(y) \;=\cos\left[\pi\left(y-a_y\right)\right]\cosh(b_y-y) \qquad (3)$$

$$u(x=b_x,y)=\varphi_{ab}(y)=(y-\;a_y)^2 sin\frac{\pi(y-a_y)}{2(b_y-a_y)} \qquad (4)$$

Neumann Boundary Conditions:

$$\frac{\partial u}{\partial y}|(y=a_y)=0 \qquad (5)$$

$$\frac{\partial u}{\partial y}|(y=a_y)=0 \qquad (6)$$

Discretization

Numerical discretization schemes are important tools for solving partial differential equations. Finite difference discretization schemes approximating derivatives using the Taylor Series are implemented in this report. In order for a numerical scheme to converge on the solution, the problem must be well posed and the approach must have consistency and stability. The two methods used for discretizing the two dimensional diffusion equation described above are the explicit approach and the alternating direction implicit approach.

Explicit Method Discretization

The explicit method approach used in this report uses the first derivative forward difference in time with an error of order Δt as follows.

$$\frac{\partial u}{\partial t} = \frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t}$$

The second derivative centered difference for the x and y dimension are used with orders or error Δx^2 and Δy^2, respectively.

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i-1,j}^{n} - 2u_{i,j}^{n} + u_{i+1,j}^{n}}{\Delta x^2}$$

$$\frac{\partial^2 u}{\partial y^2} = \frac{u_{i,j-1}^{n} - 2u_{i,j}^{n} + u_{i,j+1}^{n}}{\Delta y^2}$$

The von Neumann criterion for this approach shows that, for stability,

$$\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2} < \frac{1}{2}$$

For the two dimensional diffusion equation solved in this problem the x and y dimensions have equal intervals. It was decided to discretize them equally, thus $\Delta x = \Delta y$. Therefore, for stability,

$$\frac{\Delta t}{\Delta x^2} < \frac{1}{4}$$

The explicit discretization used reduces to, the following,

$$u_{ij}^{n+1} = \frac{\Delta t}{\Delta x^2} u_{i,j-1}^{n} + \frac{\Delta t}{\Delta x^2} u_{i-1,j}^{n} + \left(1 - 4 * \frac{\Delta t}{\Delta x^2}\right) u_{i,j}^{n} + \frac{\Delta t}{\Delta x^2} u_{i+1,j}^{n} + \frac{\Delta t}{\Delta x^2} u_{i,j+1}^{n}$$

Alternating Direction Implicit Method Discretization

The tradition implicit method would provide a nasty penta-diagonal matrix, thus the elegant alternating direction implicit scheme was implemented. This method reduces the penta-diagonal into two tri-diagonal matrices taking half a time step in each. This method is unconditionally stabile and second order in time and space. The first step is explicit in y and implicit in x, as follows,

$$\frac{u_{i,j}^{n+\frac{1}{2}} - u_{i,j}^{n}}{\Delta t/2} = \frac{u_{i-1,j}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i+1,j}^{n+\frac{1}{2}}}{\Delta x^2} + \frac{u_{i,j-1}^{n} - 2u_{i,j}^{n} + u_{i,j+1}^{n}}{\Delta y^2}$$

Once again taking Δx to equal Δy the equation reduces to, the following,

$$-\frac{\Delta t}{2\Delta x^2}u_{i-1,j}^{n+\frac{1}{2}} + \left(1+\frac{\Delta t}{\Delta x^2}\right)u_{i,j}^{n+\frac{1}{2}} - \frac{\Delta t}{2\Delta x^2}u_{i+1,j}^{n+\frac{1}{2}} = \frac{\Delta t}{2\Delta x^2}u_{i,j-1}^{n} + \left(1-\frac{\Delta t}{\Delta x^2}\right)u_{i,j}^{n} + \frac{\Delta t}{2\Delta x^2}u_{i,j+1}^{n}$$

The second step is explicit in x and implicit in y and is reduced, as follows,

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n+\frac{1}{2}}}{\Delta t/2} = \frac{u_{i-1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i+1,j}^{n+1}}{\Delta x^2} + \frac{u_{i,j-1}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i,j+1}^{n+\frac{1}{2}}}{\Delta y^2}$$

Once again taking Δx to equal Δy the equation reduces to, the following,

$$-\frac{\Delta t}{2\Delta x^2}u_{i,j-1}^{n+1} + \left(1+\frac{\Delta t}{\Delta x^2}\right)u_{i,j}^{n+1} - \frac{\Delta t}{2\Delta x^2}u_{i,j+1}^{n+1} = \frac{\Delta t}{2\Delta x^2}u_{i-1,j}^{n+\frac{1}{2}} + \left(1-\frac{\Delta t}{\Delta x^2}\right)u_{i,j}^{n+\frac{1}{2}} + \frac{\Delta t}{2\Delta x^2}u_{i+1,j}^{n+\frac{1}{2}}$$

Algorithm

In this section of the report the algorithms for implementing the explicit discretization and alternating direction implicit discretization on MATLAB are documented. The general pseudo code with explanations is used to convey the algorithm used.

Explicit Method Algorithm

First the number of nodes N is chosen. The greater the number of nodes the finer the mesh.

N=60

The x and y dimension grid increment is defined note $\Delta x = \Delta y$, because the $interval\ of\ x =$ interval of y.

$$\Delta x = \frac{2\pi}{(N-1)}$$

The time dimension grid increment is defined. This definition is used based on rearrangement of the stability condition. A factor of 0.99 is used to ensure.

$$\Delta t = \frac{(0.99)\Delta x^2}{4}$$

Next the time vector, x vector, and y vector are defined based on their intervals and grid increments. The time interval was determined by testing for steady state. The central point value versus time was plotted until the solution no longer varied with time. Excess time was used to demonstrate the slope of the value of the central point versus time going to zero.

$$t = 0{:}\,\Delta t{:}\,30$$
$$x = y = 0{:}\,\Delta x{:}\,2\pi$$

The constants are defined to simplify the expressions before the loop.

$$k = \frac{\Delta t}{\Delta x^2}$$
$$c = \left(1 - 4 * \frac{\Delta t}{\Delta x^2}\right)$$

For use in loops the length of the of the space and time dimensions is recorded.

$$\#\ of\ elements\ in\ x = \#\ of\ elements\ in\ y = nx = length(x)$$

$$\#\ of\ elements\ in\ time = ts = length(t)$$

The Dirichlet boundary conditions were reduced and are evaluated to produce vectors.
The Dirichlet boundary condition at x=0.

$$\emptyset_{ab}(y) \ = \cos[\,\pi * y\,] .* \cosh(2\pi - y)$$

The Dirichlet boundary condition at x=2π.

$$\varphi_{ab}(y) = (y).^2\, sin\left(\frac{y}{4}\right)$$

The Dirichlet boundary conditions are imposed on the grid boundaries while the initial condition is imposed on the interior points of the grid.

$$u^n = [\emptyset_{ab}(y);\ zeros(N, N+2);\ \varphi_{ab}(y)]$$

Based on the discretization for the explicit method a for loop for the time index is started at the initial condition n=1 in steps of 1 until the number of elements in time. Next, the for loop for the y space index is started at j=1, in steps of 1, until the number of elements in y. Finally, the for loop for the x index is started at i=2, in steps of I, until i= the number of elements in x minus 1. This is done because the Dirichlet boundary conditions are defined for i=1 and for i= the number of elements in x while the Nuemann boundary conditions will use the ghost nodes of $u_{i,0}^n$ and $u_{i,nx+1}^n$.

```
$for\ h = 1{:}\,ts$
$for\ j = 1{:}\,nx$
$for\ i = 2{:}\,nx - 1$
```

Next, the Neumann boundary conditions are defined using a centered difference approximation of the first derivative, to minimize error introduced.

$$\frac{\partial u}{\partial y}|(y=0) = \frac{\partial u}{\partial y}|(y=2\pi) = 0 = \frac{u_{i,j-1}^n - u_{i,j+1}^n}{2 * \Delta y}$$

Thus,

$$at\ j = 1, \qquad u_{i,0}^{n} = u_{i,2}^{n}$$
$$at\ \ j = nx, \qquad u_{i,nx-1}^{n} = u_{i,nx+1}^{n}$$

Therefore,

$$if\ j == 1 \quad u_{i,j}^{n+h} = k * u_{i-1,1}^{n} + c * u_{i,1}^{n} + k * u_{i+1,j}^{n} + 2k * u_{i,2}^{n}$$
$$elseif\ j == nx \qquad u_{i,j}^{n+h} = 2k * u_{i,nx-1}^{n} + k * u_{i-1,j}^{n} + c * u_{i,nx}^{n} + k * u_{i+1,nx}^{n}$$

$$else\ \ u_{i,j}^{n+h} = k * u_{i,j-1}^{n} + k * u_{i-1,j}^{n} + c * u_{i,j}^{n} + k * u_{i+1,j}^{n} + k * u_{i,j+1}^{n}$$

While the actual MATLAB code for performing this algorithm is more involved and optimized this is the basic algorithm for developing the solution.

Alternating Direction Implicit Method Algorithm

First the number of nodes N is chosen. The greater the number of nodes the finer the mesh.

N=60

The x and y dimension grid increment is defined note $\Delta x = \Delta y$, because the $interval\ of\ x =$ interval of y.

$$\Delta x = \frac{2\pi}{(N-1)}$$

The time dimension grid increment is defined. The stability condition for explicit is not a requirement for the alternating method implicit method, however to control error the following time increment was devised.

$$\Delta t = \Delta x^2/2$$

Next the time vector, x vector, and y vector are defined based on their intervals and grid increments. The time interval was determined by testing for steady state. The central point value versus time was plotted until the solution no longer varied with time. Excess time was used to demonstrate the slope of the value of the central point versus time going to zero.

$$t = 0{:}\ \Delta t{:}\ 30$$
$$x = y = 0{:}\ \Delta x{:}\ 2\pi$$

The constants are defined to simplify the expressions before the loop.

$$a = \frac{\Delta t}{2\Delta x^2}$$

$$b = \left(1 + \frac{\Delta t}{\Delta x^2}\right)$$

$$c = \left(1 - \frac{\Delta t}{\Delta x^2}\right)$$

For use in loops the length of the of the space and time dimensions is recorded.

$$\#\ of\ elements\ in\ x = \#\ of\ elements\ in\ y = nx = length(x)$$

$$\#\ of\ elements\ in\ time = ts = length(t)$$

The Dirichlet boundary conditions were reduced and are evaluated to produce vectors.

The Dirichlet boundary condition at x=0.

$$\emptyset_{ab}(y) \ = \cos[\,\pi * y\,] .* \cosh(2\pi - y)$$

The Dirichlet boundary condition at x=2π.

$$\varphi_{ab}(y) = (y).^{2}\, sin\left(\frac{y}{4}\right)$$

The Dirichlet boundary conditions are imposed on the grid boundaries while the initial condition is imposed on the interior points of the grid.

$$u^{n} = [\emptyset_{ab}(y);\ zeros(N, N+2);\ \varphi_{ab}(y)]$$

The alternating direction implicit requires evaluation of two tridiagonal matrices. A tridiagonal Gaussian elimination function is used to solve these matrices.

```
% Tridiag: Tridiagonal equation solver banded system
%   x = Tridiag(e,f,g,r): Tridiagonal system solver.
% input:
%   e = subdiagonal vector
%   f = diagonal vector
%   g = superdiagonal vector
%   r = right hand side vector
% output:
%   x = solution vector
% forward elimination
for k = 2:n-1
    factor = e(k-1)/f(k-1);
    f(k) = f(k) - factor*g(k-1);
    r(k) = r(k) - factor*r(k-1);
end

% back substitution
x(n) = r(n)/f(n);
for k = n-1:-1:2
    x(k) = (r(k)-g(k)*x(k+1))/f(k
```

The sub-diagonal, diagonal, and super-diagonal vectors for these tridiagonal matrices are based on the implicit discretization of x in the first half step in time using the Dirichlet boundary conditions and the implicit discretization of y in the second half step in time using the Neumann boundary conditions.

Implicit discretization in x for the first time step using Dirichlet boundary conditions.

```
e=[-a*ones(1,N-2) 0];
f=[ 1 b*ones(1, N-2) 1];
g=[ 0 -a*ones(1, N-2)];
```

Implicit discretization in y for the first time step using Neumann boundary conditions.

```
e2=[-a*ones(1,N-2) -2*a];
f2=b*ones(N);
g2=[-2*a -a*ones(1,N-2)];
```

The loops for time, the x dimension, and the y dimension are ready to run.

```
for h=1:ts;
    for j=1:nx;
        for i=2:nx-1;
```

r(i) is the explicit solution in y for the first half time step Neumann boundary conditions are imposed in the explict solution of y by use of the ghost nodes U(i,0) and U(i,N+1, where U(i,0)=U(i,2) and U(i,N+1)=U(i,N-1) i.e. the centered difference approximation is used at the boundaries of y. Thus, the discretization reduces to r(i)=c*U(i,1)+2*a*U(i,2) at y=0 and r(i)=2*a*U(i,nx-1)+c*U(i,nx) at y= 2pi

```
            if j==1
                r(i)=c*U(i,1)+2*a*U(i,2);
            elseif j==nx
                r(i)=2*a*U(i,nx-1)+c*U(i,nx);
```

Everywhere else the originally derived explicit side of the discretization is used

```
            else
                r(i)=a*U(i,j-1)+c*U(i,j)+a*U(i,j+1);
            end
        end
```

The Dirichlet boundary conditions are imposed on the explicit solution in y

```
        r=[U(1,j) r(2:nx-1) U(nx,j)];
```

The implicit solution in x is determined using the tridiagonal equations solver in which the sub-diagonal, diagonal, and super-diagonal where determined form discretization and the Dirichlet boundary conditions for x

```
        x=tridiag(e,f,g,r);
```

Finally u(n+1/2) is determined by the combination of the explicit solution in y and the implicit solution in x

```
        U(:,j)=x;
    end
```

For the second half step in time transverse the rows of u(n+1) and determine the right hand side vector using the explicit solution in x

```
    for i=2:nx-1;
        for j=1:nx;
```

r(i) is the explicit solution in x for the second half time step

```
             r2(j)=a*U(i-1,j)+c*U(i,j)+a*U(i+1,j);
        end
```

The implicit solution in y is determined using the tridiagonal equations solver in which the sub-diagonal, diagonal, and super-diagonal where determined form discretization and the Neumann boundary conditions for y

```
        x2=tridiag(e2,f2,g2,r2);
```

Finally u(n+1) is determined by the combination of the explicit half step in time solution in x and the implicit solution in x

```
        U(i,:)=x2;
    end
    end
```

The process is continued until the time loop reaches the end

```
end
```

Verification

Verification is essential to numerical methods. Verification is the confirmation that the mathematical laws are followed in producing the solution. Without verification the solution is useless. In order to verify the solution steady state, grid convergence, a comparison of the solutions determined. To determine steady state a centrally located point value was plotted versus time. When this value ceased changing with time the solution was determined to reach steady state. The plot of this centrally located point versus time supported the trend of diffusion likely to result from the given problems initial and boundary conditions. Grid convergence was determined by comparing values of the steady state solutions L2 norm to that of steady state solutions with double the amount of nodes until difference in norms was minimized. The solutions between the explicit and alternating direction implicit support the same trend of diffusion from the given initial and boundary conditions.

Results

Grid Independence

Grid convergence was determined by comparing values of the steady state solutions L2 norm to that of steady state solutions with double the amount of nodes until difference in norms was minimized. Below in Figure 1 the graph for convergence using the explicit method is portrayed. The implicit method developed suffered from truncation error at a much smaller number of nodes. This is most likely due to the use of Gaussian Elimination in the solving of the tridiagonal matrices produced by the implicit part of the solution. Using the finest converging grid as reference the relative error plot was produced in Figure 2.

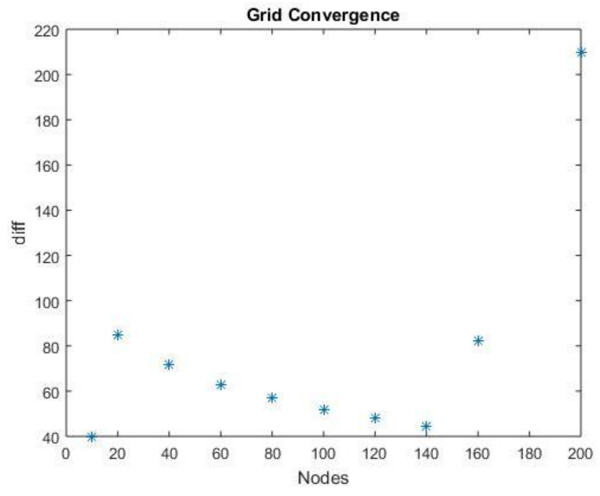

Figure 1 Explicit Grid Convergence

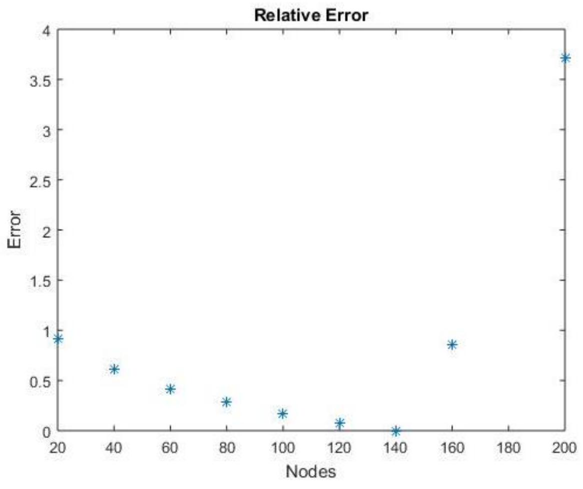

Figure 2 Explicit Relative Error

Graphs

For the explicit method the results produced using node sizes between 40 and 140 produced similar results. Surface plots of some of these solutions are provided in Figures 3 and 4. Tracking of the values given by x= pi y= pi produced the value of 7 as the solution reached steady state. For the alternating direction implicit method the value at this location also resulted in a value of 7 as the solution reached steady state. Graphs of these results are presented in Figures 5 and 6. The surface plots produced by various grid sizes using the alternating direction implicit are provided in Figures 7 and 8.

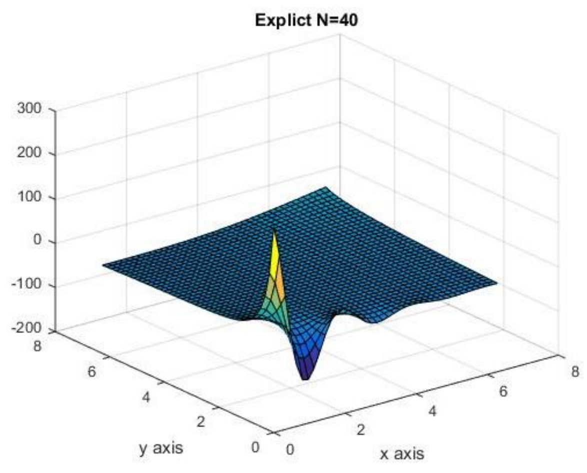

Figure 3

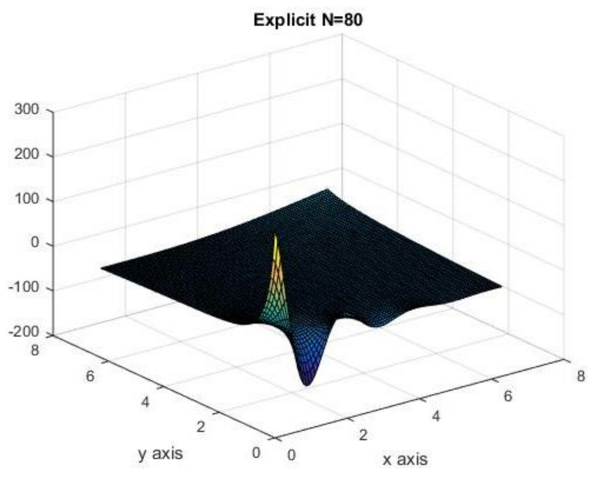

Figure 4

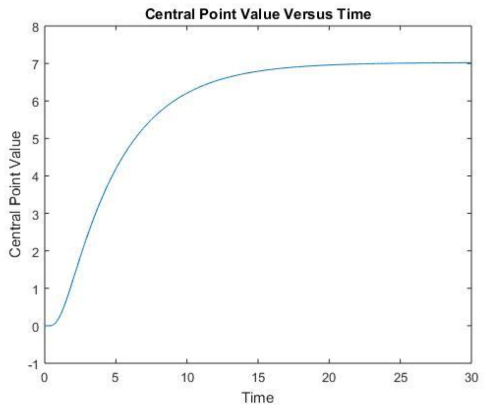

Figure 5 Explicit

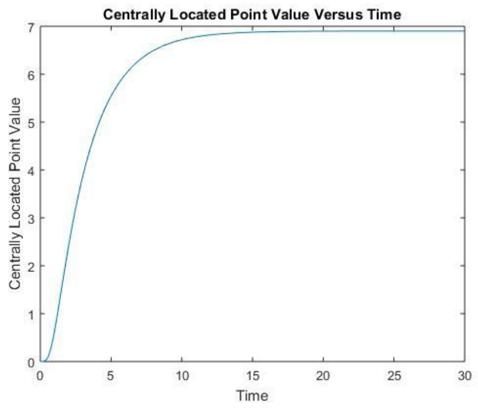

Figure 6 Implicit

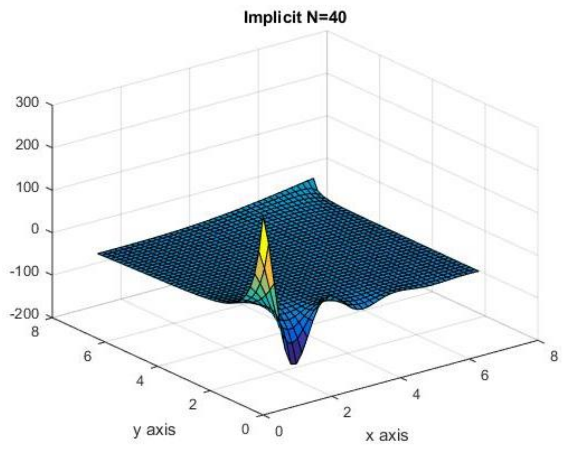

Figure 7

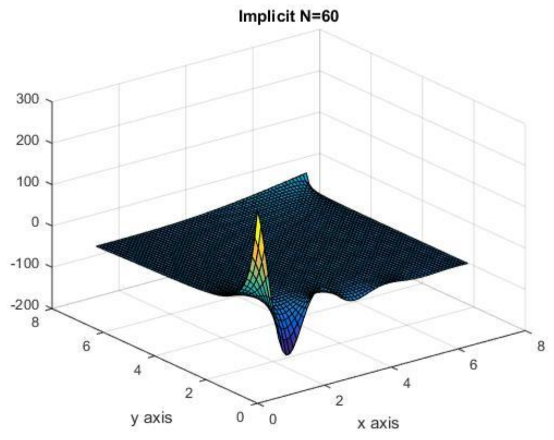

Figure 8