Numerical Solution of a Transient Two-Dimensional Heat Conduction Equation using the ADI Method

Rahul R Upadhyay*

Master of Science, Dept of Mechanical Engineering, UNC Charlotte

SUMMARY

In this project we have solved Poisson’s transient heat equation for a two dimensional plate with constant physical properties and heat source using Alternating Direction Implicit Method(ADI), its modified form(M-ADI) and Finite Difference Explicit method(FDM-E). The results obtained using these three schemes were compared. MATLAB was used to do the computations and the program was made to run until convergence criteria was reached. Here convergence implies onset of steady state. The performance of the schemes were judged on the basis of the rate of convergence and computational time of the main iteration block of each program. In this report a method to improve the conventional ADI method by successive replacement is also proposed.

KEY WORDS: Possion Equation; Finite Difference Method; Transient Conduction; MATLAB; Sparse and Banded Matrix

1. INTRODUCTION

When we solve one dimensional Poisson’s equation using the Crank Nicholson’s finite difference Implicit method we get a tri-diagonal banded and sparse matrix which is easily invertible. However when we apply the same scheme to a two dimensional case, we get a band diagonal matrix with large bandwidth, rather than a tridiagonal matrix, which is computationally costly to solve. To dodge the problems associated with solving systems of equation, FDM-E or finite difference explicit method is used, which computes the solution at $t_{n+1}$ time with the values known at the $t_n$ time. This scheme is faster but it is conditionally stable.

ADI method or Alternating Direction Implicit method is a unconditionally stable implicit algorithm which leads to tri-diagonal systems of equation. Thus this brings the advantage of tri-diagonal sparse matrix along with unconditional stability. In ADI method the given time increment is broken into two steps such that in each step the implicit scheme is applied in only one direction while the other direction is treated explicitly.

In this project we solve transient two dimensional heat conduction equation, i.e
the Poisson’s equation using ADI method, M-ADI or modified ADI method and Explicit method and compared the solutions obtained by them.

1. MATLAB programs for the above mentioned schemes were developed and the results and graphs were obtained for varying mesh sizes in each direction.
2. As it is a transient analysis, the code was made to run until a convergence criteria for the onset of steady state was reached.
3. As we are get a \( N_Y \times N_X \) matrix where \( N_X \) and \( N_Y \) are the number of nodes in y and x direction, the convergence criteria cannot be judged on the the convergence of any random point. Here we have defined a convergence criteria on the basis \( L_2 \) norm.
4. The condition for convergence is defined by the following equation:

\[
\text{abs}(||U||_2 - ||K||_2) < \frac{1}{10^6} \quad (1)
\]

where \( U \) and \( K \) are the temperature matrices of plate at \( t_{n+1} \) and \( t_n \) times.
5. The computational time, number of increments and graphical results of the various schemes were compared with each other.

2. PROBLEM STATEMENT

1. Governing Equation, Poisson’s Two Dimensional Heat Conduction Equation,

\[
K\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + Q(x, y) = \rho C_p \frac{\partial u}{\partial t} \quad (2)
\]

- \( K \) = Thermal Conductivity
- \( \rho \) = Density
- \( C_p \) = Specific Heat

2. Boundary Conditions

- \( U_{\infty} = 20 \) deg Celcius
- \( U_0 \) = Initial Temperature =20 deg Celcius

3. Heat Source Equation,

\[
r = \sqrt{(x - w)^2 + (y - h)^2} \quad \text{(3)}
\]

\[
Q(x, y) = Ae^{-b\left(\frac{r}{\tau_0}\right)^2} \quad \text{(4)}
\]

- \( b = 20 \)

Spring 2015

Department of Mechanical Engineering and Engineering Science, UNC Charlotte
• $A = 2e6$
• $r_o = 0.2$

4. We solve the above partial differential equation numerically for a rectangular plate with dimensions $2w=0.8$ and $2h=0.4$. This is a part of Homework4, Problem 1 where we used implicit method to solve for the steady state temperature distribution of the plate. In the present case we do a transient analysis and run the program until steady state is reached.

3. ALTERNATING DIRECTION IMPLICIT METHOD

In alternating direction implicit method, the given time increment is broken into two time steps and in each time step the implicit method is used in only one direction while the other direction is treated explicitly.

ADI method can be mathematically expressed using the following expression,

$$\frac{u_{ij}^{n+1/2} - u_{ij}^n}{\delta t/2} = \alpha\left(\frac{u_{i+1,j}^{n+1/2} - 2u_{i,j}^{n+1/2} + u_{i-1,j}^{n+1/2}}{\Delta x^2} + \frac{u_{i,j+1}^{n+1/2} - 2u_{i,j}^{n+1/2} + u_{i,j-1}^{n+1/2}}{\delta y^2}\right) + \frac{Q_{ij}^{n+1/2}}{\rho C_p}$$

(5)

$$\frac{u_{ij}^{n+1} - u_{ij}^{n+1/2}}{\delta t/2} = \alpha\left(\frac{u_{i+1,j}^{n+1/2} - 2u_{i,j}^{n+1/2} + u_{i-1,j}^{n+1/2}}{\Delta x^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{\delta y^2}\right) + \frac{Q_{ij}^{n+1}}{\rho C_p}$$

(6)
In our MATLAB code we clubbed some of the constants to make the scheme more legible, understandable and easier to program.

\[
a = -\frac{\alpha \Delta t}{2\delta y^2}
\]

\[
b = -\frac{\alpha \Delta t}{2\delta x^2}
\]

\[
c = -\frac{\Delta t}{2 \rho C_p}
\]

Thus the expression of ADI can be rewritten as,

\[
[bu_{i-1j} + (1 - 2b)u_{ij} + bu_{i+1j})]^{n+1/2} = u_{ij}^n - a[u_{ij+1} - 2u_{ij} + u_{ij-1}]^n
\]

\[
[aui_{i-1} + (1 - 2a)u_{ij} + au_{i+1j})]^{n+1} = u_{ij}^{n+1/2} - b[u_{i+1j} - 2u_{ij} + u_{i-1j}]^{n+1/2}
\]

- The method is implicit in only one direction at each step and thus yielding a tri-diagonal system of equation.
- ADI method is unconditionally stable and thus larger \( \delta t \) values can be taken.
- Since larger \( \delta t \) value can be taken, the truncation error also increases significantly.
- Using this method the solution is obtained in fewer increments even though the computational time for each increment might be larger in comparison to the explicit scheme.

4. MODIFIED ADI METHOD

Besides the conventional ADI method, in our project Modified ADI Method is also employed to solve the given problem. In conventional ADI method, in each time increment the implicit scheme is applied in x direction while y is treated explicitly followed by y being treated implicitly and x being treated explicitly in the same increment.

In M-ADI method the direction of calculation is changed alternatively in each time increment. Thus, if in the first time increment, the x direction is treated implicitly in first step and y is treated implicitly in second time step then y will be now treated implicitly in first step and x will be treated implicitly in second step in the second increment.

A comparison of ADI method and M-ADI method is done in a tabular form in the above table. It can be seen that the ADI method is biased in the sense

Department of Mechanical Engineering and Engineering Science, UNC Charlotte

Spring 2015
Table I. Alternating Direction Implicit Method.

<table>
<thead>
<tr>
<th>Time Increment</th>
<th>Method 1</th>
<th>Method 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st Time Increment</td>
<td>Implicit in x</td>
<td>Explicit in y</td>
</tr>
<tr>
<td>2nd Time Increment</td>
<td>Implicit in y</td>
<td>Explicit in x</td>
</tr>
<tr>
<td>3rd Time Increment</td>
<td>Implicit in x</td>
<td>Explicit in y</td>
</tr>
<tr>
<td>4th Time Increment</td>
<td>Implicit in y</td>
<td>Explicit in x</td>
</tr>
<tr>
<td>5th Time Increment</td>
<td>Implicit in x</td>
<td>Explicit in y</td>
</tr>
</tbody>
</table>

that implicit scheme is applied in x direction and then in the y direction in each increment. However, in M-ADI method implicit scheme is applied in x direction and then in y direction in the nth step and the direction is reversed in the n+1th step. Thus the direction is changed alternatively between the time increments.

5. THOMAS ALGORITHM TO SOLVE TRI-DIAGONAL SYSTEM OF EQUATIONS

Thomas Algorithm was utilized to solve the tridiagonal system of equations. The code of which is the following:

```matlab
function k=thomas(a,b,c,f)

s = length(f);
v = zeros(s,1);
z = a(1);
k = v;
k(1) = f(1)/z;
for i=2:s
    v(i-1) = c(i-1)/z;
    z = a(i) - b(i)*v(i-1);
    k(i) = ( f(i) - b(i)*k(i-1) )/z;
end
for j=s-1:-1:1
    k(j) = k(j) - v(j)*k(j+1);
end
end
```

Department of Mechanical Engineering and Engineering Science, UNC Charlotte

Spring 2015
6. VALIDATION OF THE CODE

1. A matlab code to implement the ADI method was developed as a part of the project and there was a need to validate the proper functioning of the code and the algorithm before further detailed analysis of the method could be done.

2. For the purpose of testing the functioning of the code and ADI algorithm, a reference analytical or numerical solution was needed.

3. ssheat2d.m MATLAB code which was provided as a part of MEGR 7172 course to solve two dimensional heat conduction equation using implicit method, was utilized and the temperature matrix obtained using this was taken as the reference solution.

4. ssheat2d.m solved steady state heat conduction equation and ADI.m(our ADI code) solved transient heat conduction equation with the same boundary,initial conditions and heat source. Thus the ADI code was run until convergence and compared with the results obtained from ssheat2d.m. Here convergence implies, onset of steady state condition.

5. Following analysis was done to validate our ADI algorithm and code:

   • Analysis Parameters:

   (a) Width of Plate = 2 X 0.4 m
   (b) Height of Plate = 2X 0.2 m
   (c) Dirichlet Boundary Conditions, \( u_\infty = 20 \text{ deg C} \)
   (d) \( I = 30 \)
   (e) \( J = 20 \)
   (f) \( \delta t = 0.9 \)
   (g) Heat Source,

   \[
   r = \sqrt{(x - w/2)^2 + (y - h/2)^2} \quad (12)
   \]

   \[
   Q(x, y) = A e^{-b \left( \frac{r}{r_0} \right)^2} \quad (13)
   \]

6. The heat source function mentioned above to validate the code is used for all the analysis mentioned in this report for the sake of consistency.

7. More mesh points were chosen along the x axis as the length of plate is more compared to the width.
6.1. Results Obtained

1. Figure(a) shows the surface plot of temperature in the plate after steady state is reached. The steady state temperature is around 24.5 deg Celcius as can be seen from the plot.
2. Figure(b) shows the temperature evolution of the plate. One can see that the curve starts from point 20 on the temperature axis which corresponds
to initial temperature and it goes all the way to around 24.5 deg C which is
the steady state temperature. The plot terminates when the steady state is
reached corresponding to $10^{-7}$ convergence tolerance of the absolute value of
the difference of norms of temperature matrix in the successive increments.
In this case, the graph terminated at 1686 seconds which corresponds to
around 1860 iterations with 0.9 as the time increment.

3. Figure(c) and (d) are the contour plots obtained from ssheat2d.m and
   ADI.m.

4. On comparing the contour plots from Figure (c) and (d), we can see that
   both the solutions are almost exactly the same and thus they confirm that
   our code is correct and the ADI algorithm works.

5. The results obtained from ADI.m and M-ADI were almost the same.

7. MESH SIZE DETERMINATION

Some important points taken into consideration before plotting and analyzing
the results,

1. ADI method is unconditionally stable and thus large values of $\delta t$
can be
   chosen to make the solution reach faster. But the disadvantage of using
   large time increments is large truncation error.

2. For increasing the computational speed one can also decrease the mesh
   size thus leading to fewer computational steps. But yet again this leads to
   inaccurate results.

3. For proper analysis it is important that there is a balance between the need
   for speed of computation and the need for accuracy.

4. For meeting the need of choosing optimal mesh size for computation without
   compromising the accuracy of the end result the following was done,
   
   - For a constant $\delta t$ (1s) value the temperature evolution of the mid point
     of the plate was plotted against the time till convergence was reached.
   - It was seen that using very coarse mesh leads to highly inaccurate
     results and as the mesh size increases, the solution obtained also
     converges to similar values.
   - The following plot shows the temperature evolution of the mid point
     against various mesh sizes. In the legend "I—J", I corresponds to
     number of points+1 along x axis and J corresponds to number of
     points+1 along y axis.

5. On viewing the plot it was found that mesh size of 10—10, 15—15 are
   nowhere close to the actual result however for mesh sizes of 25—25, 30—30
   and 40—40 the solution seemed to converge. Thus I=35 and J=30 was
   chosen to do the entire batch of analysis of ADI method.

6. This mesh size gave an almost perfect balance between speed and accuracy
   of computation.
7. This mesh size was also influenced by the fact that the width of the plate is more than the height and thus it's wiser to have more mesh points along the width of the plate.

8. NUMERICAL RESULTS

In this section we put forth the results obtained from the ADI.m code for three different times, the initial phase when the temperature rises rapidly, the intermediate phase when the temperature is almost reaching steady state and lastly, the steady state phase when the temperature profile doesn't change with time.

To do this the job was initially run until convergence was reached. Then the number of iterations was noted down and the job was re-run with preset number of iterations without the convergence checking code.

With I=35, J=30 and $\delta t=1$ the analysis was run and the steady state was reached after 1649 iterations. However on seeing the graph of temperature evolution it seems that the steady state has reached quite early. This is due to the high tolerance criteria of $10^{-7}$ for convergence in the main looping body to ensure highly accurate results.

8.1. Initial Phase

- Time Span = 100 seconds
- $dt = 1$ second
- Computational Time of Looping Body = 0.7347 seconds
- Temperature of Mid point= 23.5 deg C
- Norm of Temperature matrix $= 776.2593$
8.2. Intermediate Phase

- Time Span = 250 seconds
- $dt = 1$ second
- Computational Time of Looping Body = 3.6161 seconds
- Temperature of Mid point = 23.2174 deg C
- Norm of Temperature matrix = 791.4725
8.3. Steady State Phase

- Time Span = 1649 seconds
- \( dt = 1 \) second
- Computational Time of Looping Body = 26.5629 seconds
- Temperature of Mid point= 25.1725 deg C
- Norm of Temperature matrix = 795.0918

The plots obtained for steady state were confirmed to be accurate by running the same analysis using ssheat2d.m file.
9. COMPARISON OF ADI AND M-ADI METHOD

Unlike conventional implicit method, in ADI method, implicit scheme is used in only one direction at a time while the other is treated explicitly. Thus two implicit schemes are solved in each time increment individually. Keeping this in mind, the convergence check was done after each implicit block rather than being put only at the end of the whole loop block. Doing this we realized that convergence was reached in fewer increments in comparison to when the convergence check was put in the end of the loop block.

In M-ADI method the direction of implicit scheme is altered after every time increment. This was realized using if-else condition in the main looping block. In even increments, x was treated implicitly in the first step whereas in odd increments, y was treated implicitly in the first step.
Algorithm:

ADI:

1. 
2. 
3. Loop 1-->n
   [Implicit in X]
   Convergence Check
   [Implicit in Y]
   Convergence Check
4. 
5. 
6. end Loop
-------------------

M-ADI:

Loop 1-->n
if(n is even)
   [Implicit in X]
   Convergence Check
   [Implicit in Y]
   [Convergence Check]

elseif(n is odd)
   [Implicit in Y]
   Convergence Check
   [Implicit in X]
   [Convergence Check]

end Loop
-------------------

Table II. Quantitative Comparison of ADI and M-ADI schemes.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Alternating Direction Implicit Method</td>
<td>Modified Alternating Direction Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>I</td>
<td>J</td>
<td>Iterations</td>
<td>Computing Time</td>
<td>Iterations</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>15</td>
<td>1603</td>
<td>2.6603</td>
<td>1603</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>15</td>
<td>1571</td>
<td>6.5153</td>
<td>1571</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>30</td>
<td>1610</td>
<td>13.6103</td>
<td>1610</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>10</td>
<td>1481</td>
<td>2.4381</td>
<td>1481</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>25</td>
<td>1613</td>
<td>13.484</td>
<td>1613</td>
</tr>
</tbody>
</table>

Table II. Quantitative Comparison of ADI and M-ADI schemes.

The above tabulates the data we obtained by running the ADI and M-ADI schemes with same parameters. We notice that there is no difference in the number of increments needed for convergence in both methods. Also, the computational time of M-ADI scheme is marginally lesser than ADI scheme when I<J. This difference in time increases with difference in the mesh size.
10. COMPARISON OF FDM-EXPLICIT AND ADI METHOD

In this section we will compare explicit method to solve the given governing equation with Alternating Direction Implicit Method. Both ADI.m code and the code for explicit method were put in the same work directory and there solutions (temperature evolution of Mid node) were plotted in the same figure to compare their functioning.

![Temperature Evolution of Mid Node](image)

Figure 7. Temperature Evolution of Mid Point With Different Mesh Sizes for ADI and Explicit Method code.

In the above figure temperature evolution of the mid point of a plate are plotted using ADI method(with red color and 2 width line) and Explicit method(with green color and 4 width line).

1. Top most curve corresponds to I=14, J=15
2. Second curve corresponds to I=35, J=30
3. Third Curve corresponds to I=10, J=10

Thus we can see that the results from both the schemes are the same however for a given tolerance criteria, ADI method reaches convergence before the explicit method in terms of number of increments. All the calculations were performed for $\delta t=0.5$ seconds and convergence tolerance of $1e^{-7}$. This case is prominent mainly when the difference between the mesh sizes are less.

1. In explicit method, the solution algorithm is simple to setup but the criteria for stability restricts one to choose large values of $\delta t$ leading to too many increments to carry out calculations over a given time interval.
2. ADI method does not restrict one from choosing large values of $\delta t$ as it unconditionally stable.
3. ADI method is chosen over other implicit methods mainly because here we solve a tridiagonal system of equation which is computationally cheap and convergence is reached much faster.

4. M-ADI can be chosen over conventional ADI method when the mesh size is biased in one direction by a good extent.

5. ADI method is not unconditionally stable in three dimensions and thus is not as practically useful as it is in case of two dimensions.

11. IMPROVED ADI METHOD

As already stated, In ADI method the time increment is broken into two steps such that in each step one direction is treated implicitly while the other explicitly. In conventional ADI method, the values in the solution matrix are not updated in the steps and the new values obtained after solution of each tridiagonal system is put in a new matrix which is used as the initial solution for the next time step.

On the lines of Gauss Siedel method, ADI.m code was altered and the solution matrix was made to update such that the rows or columns were replaced with new rows or columns as soon as the latest solution was available. After this, both the ADI.m and the new altered ADI code were run for different values of I and J and the number of iterations needed for convergence was recorded.

<table>
<thead>
<tr>
<th>I</th>
<th>J</th>
<th>Conventional ADI Method</th>
<th>Altered ADI Method</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>11</td>
<td>1886</td>
<td>1847</td>
<td>39</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>1913</td>
<td>1866</td>
<td>47</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1948</td>
<td>1853</td>
<td>95</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>1969</td>
<td>1844</td>
<td>125</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>1999</td>
<td>1780</td>
<td>219</td>
</tr>
<tr>
<td>25</td>
<td>30</td>
<td>2015</td>
<td>1771</td>
<td>244</td>
</tr>
</tbody>
</table>

Table III. Iterations in conventional and altered ADI method.

One can clearly see from the table that the number of iterations needed for convergence in case of new altered ADI method is much lesser than conventional ADI method and the difference of required iterations increases with size of mesh. One can state that altered ADI method becomes better with increase in mesh size.

*Department of Mechanical Engineering and Engineering Science, UNC Charlotte*
REFERENCES


12. APPENDIX

% ADI method
clc
clf

%Properties of Aluminium
k=247; %Thermal Conductivity
p=2.8e3; %Density
Cp=795; %Specific Heat
alpha=k/(p*Cp);

%--------------------------------%
%--------Meshing-----------------%
w=.4; %Width of Plate
h=.4; %Height of Plate
I=4; %number of parts along x axis= number of nodes -1
J=4; %number of parts along y axis= number of nodes -1

nx = I+1; %total number of points in x direction
ny = J+1; %total number of points in y direction

[x,y,dx,dy] = mesh(w,h,nx,ny);

dt=0.5; %Time Increment.
u = 20*ones(J+1,I+1); %Initial temperature profile.
unew = 20*ones(J+1,I+1);
uneww = 20*ones(J+1,I+1);
unewww = 20*ones(J+1,I+1);
B= zeros(I+1,I+1); % Coeff matrix corresponding to each row
A= zeros(J+1,J+1); % Coeff matrix corresponding to each column
fb= zeros(I+1,1); % Force vector corresponding to each row
a1= zeros(I+1,1);
b1=zeros(I+1,1); c1=zeros(I+1,1);
fa=zeros(J+1,1); % force vector corresponding to each column
a2=zeros(J+1,1);
b2=zeros(J+1,1);
c2=zeros(J+1,1);
%----Clubbing Constants----------%
a= -alpha*dt/(2*(dy^2));
b= -alpha*dt/(2*(dx^2));
c= -(dt/2)*(1/(p*Cp));
%--------------------------------%
hold all

tic
for G=1:300000 %maximum Iterations
K=u;
t(G)=G*dt; %stores time
v(G)=u(ceil(J/2),ceil(I/2)) ; %stores temperature of a point
for j=2:J
  B(1,1)=1;
  B(I+1,I+1)=1;
  fb(1)=20;
  fb(I+1)=20;
  for m = 2: I
    B(m,m-1)=b;
    B(m,m)=(1-2*b);
    B(m,m+1)=b;
  end
  for m=2:I
    x1=x(m);
    y1=y(j);
    Q=heatsource(x1,y1,w,h);
    fb(m)=u(j,m)-a*(u(j-1,m)+u(j+1,m)-2*u(j,m))-c*Q;
    b1(m)=b;
    a1(m)=(1-2*b);
    c1(m)=b;
  end
Department of Mechanical Engineering and Engineering Science, UNC Charlotte

Spring 2015
a1(1)=1;
a1(m+1)=1;
b1(m+1)=0;
c1(1)=0;
soln=thomas(a1,b1,c1,fb);
%soln=B\fb;

%u(j,:)=soln;
unew(j,:)=soln;
end

u=unew;
if(abs(norm(u)-norm(K))<1e-7) %Convergence is checked at two places. One after each implicit direction
    break
end
for i=2:I
    A(1,1)=1;
    A(J+1,J+1)=1;
    fa(1)=20;
    fa(J+1)=20;
    fa(J+1)=20;
    for n = 2:J
        A(n,n-1)=a;
        A(n,n)=(1-2*a);
        A(n,n+1)=a;
    end

    for n=2:J
        x1=x(i);
        y1=y(n);
        Q=heatsource(x1,y1,w,h);

        fa(n)=u(n,i)-b*(u(n,i-1)+u(n,i+1)-2*u(n,i))-c*Q;
        b2(n)=a;
        a2(n)=(1-2*a);
        c2(n)=b;
    end
    a2(1)=1;
    a2(n+1)=1;
    b2(n+1)=0;
    c2(1)=0;
soln = thomas(a2, b2, c2, fa);
% soln = A \ fa;
uneww(:, i) = soln;

end

u = uneww;

% convergence criteria.
if (abs(norm(u) - norm(K)) < 1e-7)
    break
end
q = toc
G
% G * dt
% plot(t, v, '-r', 'LineWidth', 2)
% plot(t, v)
% xlabel('Time Span');
% ylabel('Temperature at Mid Point');
% title('Temperature Evolution of Mid Node');

% contourf(x, y, u)
% xlabel('xaxis');
% ylabel('yaxis');
% title('Contour Plot by ADI.m');

% surf(x, y, u)
% xlabel('xaxis');
% ylabel('yaxis');
% title('Surface Temperature Plot by ADI.m');
u = u';
hold off
% surf(x, y, u)

-----------------------------
function q = heatsource(w, h, x1, y1)
A = 2e6;
b = 20.0;
r0 = 0.1;
r = sqrt((x1 - w)^2 + (y1 - h)^2);
\[ q = A \exp(-b r^{-2}/r_0^{-2}); \]

---

function \([x, y, dx, dy] = mesh(w, h, nx, ny)\)

% mesh generates a rectangular grid with spacing dx and dy and nodal coordinates x and y.
% x = x-coordinate of a mesh point/node
% y = y-coordinate of a mesh point/node
% dx, dy = mesh spacing in x and y directions respectively
% w, h: 2w and 2h are the width and height of the rectangular domain

dx = 2*w/(nx-1);
dy = 2*h/(ny-1);

for i=1:nx
    x(i) = (i-1)*dx;
end;
for i=1:ny
    y(i) = (i-1)*dy;
end;

---

function \(k = \text{thomas}(a, b, c, f)\)

s = length(f);
v = zeros(s,1);
z = a(1);

k = v;
k(1) = f(1)/z;
for i=2:s
    v(i-1) = c(i-1)/z;
    z = a(i) - b(i)*v(i-1);
k(i) = (f(i) - b(i)*k(i-1))/z;
end
for j=s-1:-1:1
    k(j) = k(j) - v(j)*k(j+1);
end
end

---

Department of Mechanical Engineering and Engineering Science, UNC Charlotte

Spring 2015