Friedrich-Alexander University of Erlangen

LSTM Institute of Fluid Mechanics

Dr.-Ing. M. Münsch, S. Saha

Report for Practical Sheet 1

Students: Franciskus Xaverius Erick, 22811754, franciskus.erick@fau.de
Serouj Djabraian, 22849126, serouj.d.djabraian@fau.de

Submitted: Erlangen, December 13th 2020

Table of Contents

1.1 Discretized 2D Heat Equation 1
1.2 Results and Discussion 3

Literature References 9

Appendix 10

Table of Figures

Figure 1 : Time evolution of the temperature at x=y=0.4 for Euler explicit at stable Δt=0.0001s . 4

Figure 2 : Time evolution of the temperature at x=y=0.4 for Crank Nicolson at stable Δt=0.0001s 5

Figure 3 : Vertical Temperature Profile Euler explicit scheme for x=0.4 at t=0.16s 5

Figure 4 : Vertical Temperature Profile Crank-Nicolson scheme for x=0.4 at t=0.16s 6

Figure 5 : Contour Plots of Crank-Nicolson scheme for different t values 7

1.1 Discretized 2D Heat Equation

The discretized 2D Heat equation with CDS scheme in space and Crank-Nicolson scheme in time is as follows.

$$\begin{aligned}\frac{\mathrm{w}_{i,j}^{n+1}-\mathrm{w}_{i,j}^{n}}{\Delta t} &= \frac{1}{2}\frac{(\mathrm{w}_{i+1,j}^{n+1}-2\mathrm{w}_{i,j}^{n+1}+\mathrm{w}_{i-1,j}^{n+1})}{\Delta x^2}+\frac{1}{2}\frac{(\mathrm{w}_{i,j+1}^{n+1}-2\mathrm{w}_{i,j}^{n+1}+\mathrm{w}_{i,j-1}^{n+1})}{\Delta y^2} \\ &+\frac{1}{2}\frac{(\mathrm{w}_{i+1,j}^{n}-2\mathrm{w}_{i,j}^{n}+\mathrm{w}_{i-1,j}^{n})}{\Delta x^2}+\frac{1}{2}\frac{(\mathrm{w}_{i,j+1}^{n}-2\mathrm{w}_{i,j}^{n}+\mathrm{w}_{i,j-1}^{n})}{\Delta y^2}\end{aligned} \tag{1}$$

a) Von Neumann stability analysis can be carried out to determine the stability of Crank-Nicolson scheme. In 2D, the differential equation is represented as Fourier series, with the temperature $\mathrm{w}_{i,j}^{n+1}$ represented as $W^{n+1}e^{ik\Delta x}e^{il\Delta y}$, $\mathrm{w}_{i,j}^{n}$ as $W^{n}e^{ik\Delta x}e^{il\Delta y}$, $\mathrm{w}_{i+1,j}^{n}$ as $W^{n}e^{ik(\Delta x+1)}e^{il\Delta y}$ and $\mathrm{w}_{i,j-1}^{n}$ as $W^{n}e^{ik\Delta x}e^{il(\Delta y-1)}$.

$$\begin{aligned}&\frac{W^{n+1}e^{ik\Delta x}e^{il\Delta y}-W^{n}e^{ik\Delta x}e^{il\Delta y}}{\Delta t} \\ &=\frac{1}{2}\frac{(W^{n+1}e^{ik(\Delta x+1)}e^{il\Delta y}-2W^{n+1}e^{ik\Delta x}e^{il\Delta y}+W^{n+1}e^{ik(\Delta x-1)}e^{il\Delta y})}{\Delta x^2} \\ &+\frac{1}{2}\frac{(W^{n+1}e^{ik\Delta x}e^{il(\Delta y+1)}-2W^{n+1}e^{ik\Delta x}e^{il\Delta y}+W^{n+1}e^{ik\Delta x}e^{il(\Delta y-1)})}{\Delta y^2} \\ &+\frac{1}{2}\frac{(W^{n}e^{ik(\Delta x+1)}e^{il\Delta y}-2W^{n}e^{ik\Delta x}e^{il\Delta y}+W^{n}e^{ik(\Delta x-1)}e^{il\Delta y})}{\Delta x^2} \\ &+\frac{1}{2}\frac{(W^{n}e^{ik\Delta x}e^{il(\Delta y+1)}-2W^{n}e^{ik\Delta x}e^{il\Delta y}+W^{n}e^{ik\Delta x}e^{il(\Delta y-1)})}{\Delta y^2}\end{aligned}$$

Dividing the whole equation with $\mathrm{w}_{i,j}^{n}=W^{n}e^{ik\Delta x}e^{il\Delta y}$, we obtain:

$$\begin{aligned}\frac{G-1}{\Delta t}&=\frac{G}{2}\left(\frac{\left(e^{ik\Delta x}-2+e^{-ik\Delta x}\right)}{\Delta x^2}+\frac{\left(e^{ik\Delta y}-2+e^{-ik\Delta y}\right)}{\Delta y^2}\right) \\ &+\frac{1}{2}\left(\frac{\left(e^{ik\Delta x}-2+e^{-ik\Delta x}\right)}{\Delta x^2}+\frac{\left(e^{ik\Delta y}-2+e^{-ik\Delta y}\right)}{\Delta y^2}\right)\end{aligned}$$

With G as the amplification factor. Rearranging the equation in terms of G and substituting the exponential terms with cosine functions, we obtain:

$$G = \frac{1 - \Delta t(\frac{1 - cosk\Delta x}{\Delta x^2} + \frac{1 - cosl\Delta y}{\Delta y^2})}{1 + \Delta t(\frac{1 - cosk\Delta x}{\Delta x^2} + \frac{1 - cosl\Delta y}{\Delta y^2})}$$

The maximum of this term would be when the denominator is at its minimum and when the numerator is at its maximum, which at this case would be when $k\Delta x = l\Delta y = 2\pi$

$$G = \frac{1 - \Delta t(\frac{1 - cosk\Delta x}{\Delta x^2} + \frac{1 - cosl\Delta y}{\Delta y^2})}{1 + \Delta t(\frac{1 - cosk\Delta x}{\Delta x^2} + \frac{1 - cosl\Delta y}{\Delta y^2})} \leq \frac{1 - \Delta t(0 + 0)}{1 + \Delta t(0 + 0)} = 1$$

Therefore, as of Von Neumann stability analysis, the Crank Nicolson scheme is unconditionally stable as the amplification factor is always smaller or equal to 1.

b) To check for convergence, we need to check both the stability and consistency of the scheme. The stability is already proven in the previous part. Hence, the consistency of the scheme will now be proven. The terms in the Crank Nicolson scheme equation can be represented in terms of their Taylor Series approximations.

$$\mathrm{w}_{i,j}^{n+1} = \mathrm{w}_{i,j}^{n} + \frac{\partial w}{\partial t}\Big|_{i,j}^{n} \Delta t + \frac{\partial^2 w}{\partial t^2}\Big|_{i,j}^{n} \frac{\Delta t^2}{2} + HOT$$

$$\mathrm{w}_{i\pm1,j}^{n} = \mathrm{w}_{i,j}^{n} \pm \frac{\partial w}{\partial x}\Big|_{i,j}^{n} \Delta x + \frac{\partial^2 w}{\partial x^2}\Big|_{i,j}^{n} \frac{\Delta x^2}{2} \pm \frac{\partial^3 w}{\partial x^3}\Big|_{i,j}^{n} \frac{\Delta x^3}{6} + HOT$$

$$\mathrm{w}_{i,j\pm1}^{n} = \mathrm{w}_{i,j}^{n} \pm \frac{\partial w}{\partial y}\Big|_{i,j}^{n} \Delta y + \frac{\partial^2 w}{\partial y^2}\Big|_{i,j}^{n} \frac{\Delta y^2}{2} \pm \frac{\partial^3 w}{\partial y^3}\Big|_{i,j}^{n} \frac{\Delta y^3}{6} + HOT$$

Substituting these terms into equation (1), we will obtain the truncation error term $O(\Delta t^2) + O(\Delta x^2) + O(\Delta y^2)$ which is dependent to the on $\Delta t, \Delta x$ and Δy. As the grid is more refined, the truncation error term will also reach zero, which thus means that the scheme is consistent. As such, the scheme also converges as the grid is more refined.

1.2 Results and Discussion

a) For the case of $\Delta x = \Delta y = h$, the discretized equation can be simplified as follows:

$$\frac{\mathrm{w}_{i,j}^{n+1} - \mathrm{w}_{i,j}^{n}}{\Delta t} = \frac{1}{2}\frac{(\mathrm{w}_{i+1,j}^{n+1} + \mathrm{w}_{i,j+1}^{n+1} - 4\mathrm{w}_{i,j}^{n+1} + \mathrm{w}_{i-1,j}^{n+1} + \mathrm{w}_{i,j-1}^{n+1})}{h^2} + \frac{1}{2}\frac{(\mathrm{w}_{i+1,j}^{n} + \mathrm{w}_{i,j+1}^{n} - 4\mathrm{w}_{i,j}^{n} + \mathrm{w}_{i-1,j}^{n} + \mathrm{w}_{i,j-1}^{n})}{h^2} \quad (2)$$

The equation is then rearranged and the term $\frac{\Delta t}{h^2}$ is expressed as r to obtain an equation with the implicit terms exclusively on the left hand side and the explicit terms exclusively on the right hand side.

$$\mathrm{w}_{i,j}^{n+1}(1+2r) - \frac{r}{2}\left(\mathrm{w}_{i+1,j}^{n+1} + \mathrm{w}_{i,j+1}^{n+1} - 4\mathrm{w}_{i,j}^{n+1} + \mathrm{w}_{i-1,j}^{n+1} + \mathrm{w}_{i,j-1}^{n+1}\right) = \mathrm{w}_{i,j}^{n}(1+2r) + \frac{r}{2}(\mathrm{w}_{i+1,j}^{n} + \mathrm{w}_{i,j+1}^{n} + \mathrm{w}_{i-1,j}^{n} + \mathrm{w}_{i,j-1}^{n}) \quad (3)$$

The equation and temperature can then be represented as follows.

$$Aw^{n+1} = B + Cw^n \quad (4)$$

Whereby A is the coefficient matrix containing the coefficients for the temperature nodes in the implicit part of the equation, w^{n+1} is the array containing all the temperature node values at time n+1, B is the array containing all the known values in the equation due to the boundary condition, C is the coefficient matrix containing coefficients for the temperature nodes in the explicit part of the equation and w^n is the array containing all temperature node values at time n. A and C are pentadiagonal matrices and the coefficients are set up to reflect equation (3) properly. The equation is solved by performing the symbolic left matrix division of the equation system in MATLAB.

Similarly, explicit Euler scheme are represented as follows:

$$\mathrm{w}_{i,j}^{n+1} = \mathrm{w}_{i,j}^{n}(1+4r) + r(\mathrm{w}_{i+1,j}^{n} + \mathrm{w}_{i,j+1}^{n} + \mathrm{w}_{i-1,j}^{n} + \mathrm{w}_{i,j-1}^{n}) \quad (5)$$

$$w^{n+1} = B + Cw^n \quad (6)$$

For explicit Euler scheme, no matrix division is required as the implicit term is just the result of the multiplications and additions of all known explicit terms.

b) Since the Crank Nicolson scheme is unconditionally stable, we only need to obtain the value of stable Δt for the explicit Euler scheme. From Von Neumann stability analysis of the explicit Euler scheme, we obtain the stability condition:

$$\Delta t \leq \frac{h^2}{4}$$

Thus, a stable Δt for the explicit Euler scheme would be $\Delta t \leq \frac{h^2}{4} = \frac{\left(\frac{1}{40}\right)^2}{4} = 0.000156.$ Therefore we can take a stable Δt value of 0.0001 for both schemes.

i)

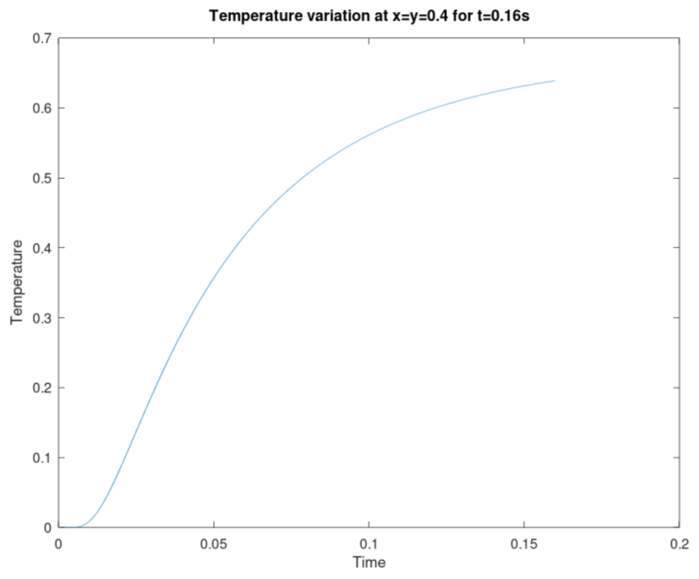

Figure 1 : Time evolution of the temperature at x=y=0.4 for Euler explicit at stable Δt=0.0001s

Figure 1 shows the evolution of the temperature at the node location x=y=0.4 for the Euler explicit scheme computed at a stable Δt of 0.0001s. It can be observed that the temperature at this node exhibits an increase from T=0 at t=0s to a maximum value of T=0.63879 at t=0.16s. It is visible that the temperature at this node stays at a value of T=0 for the first few seconds of the computation that it experiences at sudden increase. This is due to the fact that for the very first iterations this node has no connecting nodes with temperature values different than zero, since the initial condition of the temperature are all set to zero except for the boundary conditions.

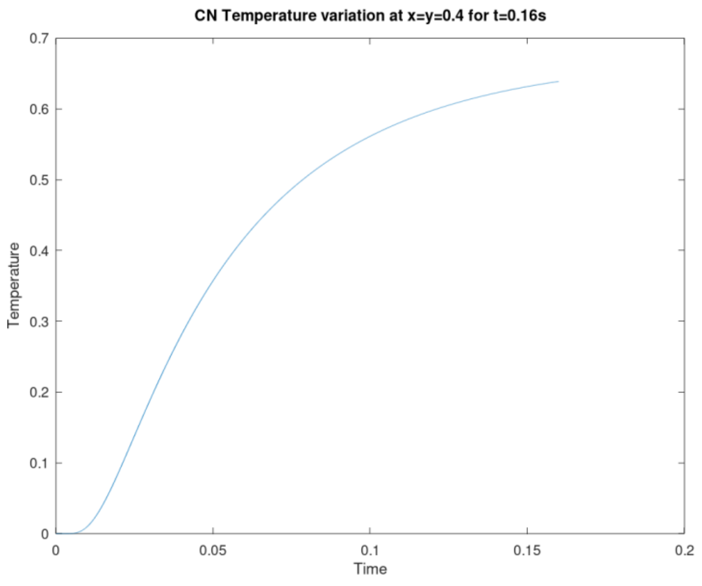

Figure 2 : Time evolution of the temperature at x=y=0.4 for Crank Nicolson at stable Δt=0.0001s

Figure 2 shows the evolution of the temperature at the node location x=y=0.4 for the Crank Nicolson scheme computed at a stable Δt of 0.0001s. The evolution is the same as the one for the Euler explicit scheme where the temperature value at the node x=y=0.4 exhibits an increase from T=0 at t=0s to a maximum value of T=0.63882 at t=0.16s. The final temperature values for both schemes are the same.

ii)

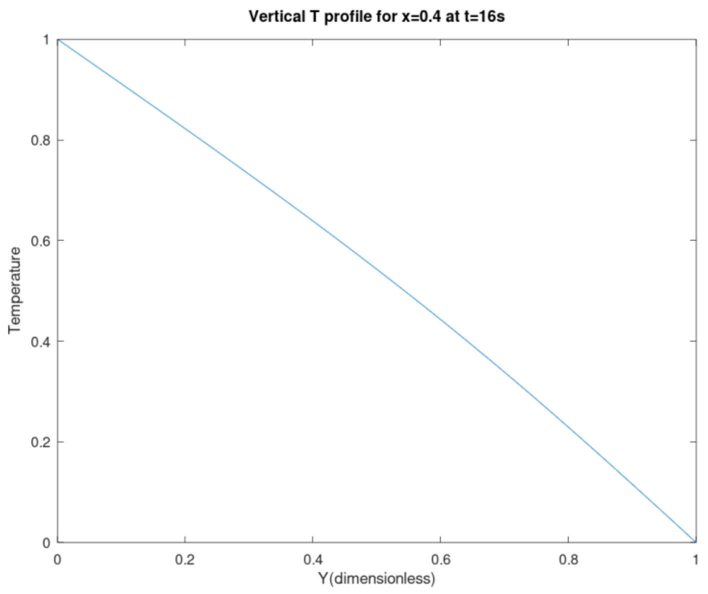

Figure 3 : Vertical Temperature Profile Euler explicit scheme for x=0.4 at t=0.16s

Figure 3 represents the vertical temperature profile for the Euler explicit scheme at x=0.4 for the final time step where t=0.16s. It is noted that the temperature is highest at the node (0.4,0) in x-y coordinates and it is equal to T=1 which is at the boundary condition. The temperature exhibits a decrease when moving away from this boundary with increasing y until it reaches a value of T=0 which is the node of the second boundary where T is set to 0.

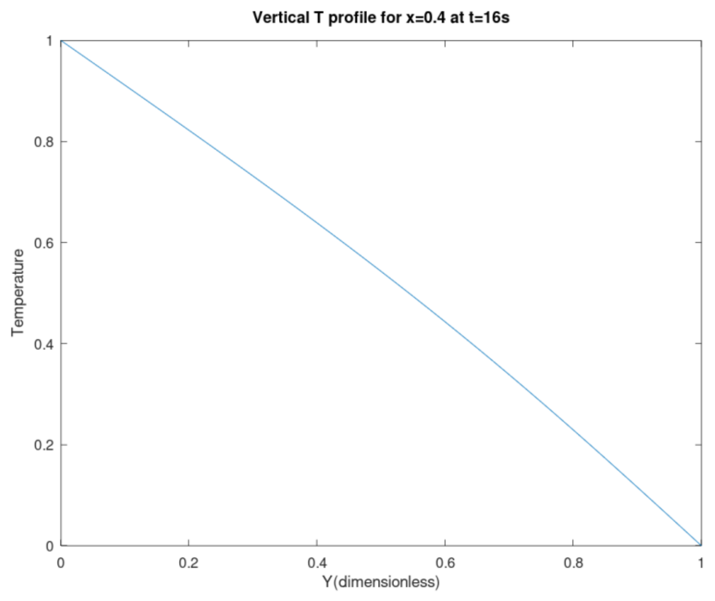

Figure 4 : Vertical Temperature Profile Crank-Nicolson scheme for x=0.4 at t=0.16s

Figure 4 represents the vertical temperature profile for the Crank Nicolson scheme at x=0.4 for the final time step where t=0.16s. It can be noted that the results of this plot is similar to that of the explicit Euler scheme where again the temperature decreases with increase in the y position of the nodes.

iii) From the figures above, it is noted that both Euler explicit and Crank Nicolson output the same results but the Euler explicit scheme is only stable for Δt=0.0001. Thus, it is critical to use very small time steps for Euler explicit compared to the Crank Nicolson scheme where a larger value of Δt can be used and still obtain the same result. This is due to the fact that Crank Nicolson uses temperature values at n+1 instead of only temperature values at n which dramatically increases the speed computation and then less time steps are needed to be calculated. The downside of Crank Nicolson is that the solution is more complex to solve where we have several variables which need to be solved in our matrix.

iv)

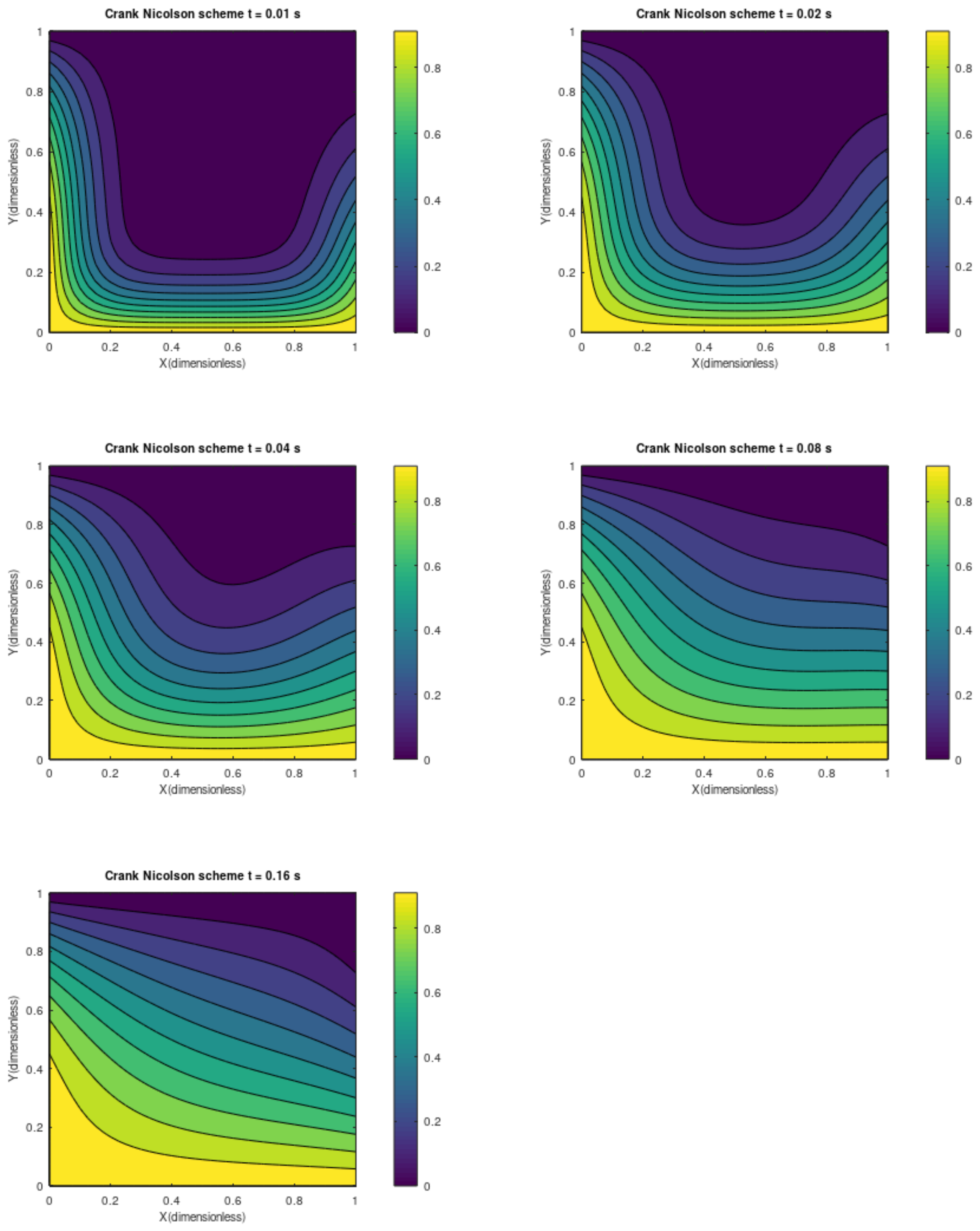

Figure 5 : Contour Plots of Crank-Nicolson scheme for different t values

The above figures represent the temperature profile of the whole domain at time(t)= 0.1s, 0.2s, 0.4s, 0.8s and 0.16s respectively. The scheme chosen is Crank-Nicolson scheme at time step dt = 0.0001 s. The temperature plot shows the physical phenomena of diffusion without convection and heat source/sink. Heat is being transferred from high to low temperature value nodes. Initially the highest value is at the lower boundary condition where the temperature is constant at T=1 presumably due to constant heating at the bottom wall and the lowest value is at the upper boundary condition where the temperature is constant at T=0 presumably due to constant cooling at the top wall. The right and left side boundaries are varying relative to the position of the node since they are expressed in terms of 2 different functions. In the beginning , the distribution of temperature is not uniform , with high temperature gradients especially near the bottom of the wall. As time passes, the temperature gradient gets smoother and more uniform throughout the space and the inner temperature value increases gradually from the initial value of 0.

Literature References

[1] M. Münsch, *Numerical Methods in Thermo-Fluid Dynamics I ; Simplification of Conservation Eqs. up to Boundary Layer Equations*, Numerical Methods in Thermo-Fluid Dynamics I, **2020**, Friedrich-Alexander Universität Erlangen, Microsoft Power Point Presentation

[2] M. Münsch, S. Saha, N. Chen, *Numerical Methods in Thermo-Fluid Dynamics I: Deliverable Task Sheet 1*, **2020**, Friedrich-Alexander Universität Erlangen,Practical Task Sheet.

[3] P. Lermusiaux, *Finite Difference*, 2.29 Numerical Fluid Mechanics, **2015**, Massachusetts Institute of Technology, Massachusetts, Lecture Notes, Retrieved from: https://ocw.mit.edu/courses/mechanical-engineering/2-29-numerical-fluid-mechanics-spring-2015/lecture-notes-and-references/MIT2_29S15_Lecture14.pdf

Appendix

Appendix A

```
%%%Euler Explicit in time with CDS scheme in space %%%

clear all
clc

%%%Setting up variables & conditions %%%

T=0.16;            %Total Time
dt=0.0001;         %Stepsize
t=T/dt;            %Total Number of steps
dx=1/40;           %Node size in x-direction
dy=1/40;           %Node size in y-direction
o=0:dx:1;          %used for plotting(Plate dimension)
p=0:dy:1;          %used for plotting(Plate dimension)
Lx=1;              %Plate size in x-direction
Ly=1;              %Plate size in y-direction
Nx=(Lx/dx)+1;      %Nb of nodes in x-direction
Ny=(Ly/dy)+1;      %Nb of nodes in y-direction
w=zeros(Nx,Ny,t); %initializing a matrix of zeroes
y=0:dx:1;
for a=1:1:Ny
  w(a,1,:)=1-y(a)^3;               %BC Left Side
endfor
for b=1:1:Nx
  w(b,Nx,:)=1-sin((pi/2)*y(b));  %BC Right Side
endfor
w(1,:,:)=1;                        %BC Top Side
w(Nx,:,:)=0;                       %BC Bottom Side
%%% Solution for a 3D matrix %%%
for k=1:1:t
  for i=2:1:Nx-1
    for j=2:1:Ny-1
      w(i,j,k+1)=w(i,j,k)+dt*(((w(i-1,j,k)-2*w(i,j,k)+w(i+1,j,k))/dx^2)+((w(i,j-
1,k)-2*w(i,j,k)+w(i,j+1,k))/dy^2));
    endfor
  endfor
  k
endfor

%%%Storing final temperature values%%%
vecsize=(Nx-2)*(Ny-2);                %Total number of elements
Tr=transpose(w(2:1:Nx-1,2:1:Ny-1,t));
M=reshape(Tr,vecsize,1);              %From matrix to a vector
%These 3 above are use to export Temperature values
%into an excel sheet for example by using a writing function
```

```
%%% Plotting the data from the solution matrix %%%
figure 1
contourf(o,p,w(:,:,t),10);
colorbar;
ylabel('Y(dimensionless)')
xlabel('X(dimensionless)')
title('Temperature profile at t=0.16s')
saveas(1,'ContourPlot.png')

%%% Plot for stable dt at x=y=0.4 for T=0.16s %%%
for i=1:1:t
  v(i)=w(0.4/dx+1,0.4/dx+1,i);  %Temperature value
endfor
timevector=[0:dt:T-dt];
figure 2
plot (timevector,v)
xlabel('Time(s)')
ylabel('Temperature(dimensionless)')
title('Temperature variation at x=y=0.4 for t=0.16s')
saveas(2,'stable x=y=0.4.png')

%%% Plot for stable dt at x=0.4 at t=0.16 %%%
for i=1:1:Ny
  B(i)=w(i,0.4/dx+1,t);        %Temperature value
endfor

figure 3
plot (o,B)
xlabel('Y(dimensionless)')
ylabel('Temperature(dimensionless)')
title('Vertical T profile for x=0.4 at t=16s')
saveas(3,'stable x=4.png')
```

Appendix B

```
%initializing dimensions for discretization in space
n = 41;                  %number of nodes
L = 1;                   %length
m =(n-2)*(n-2);          %dimension of the array containing inner node
temperature values
sm = sqrt(m);            %dimension of matrix containing inner node
temperature values
h = L/(n-1);             %height, which is in this case 1/40

%initializing temperatuer array with all the inner node temperature arrays
% and the corresponding temperature solution matrix
T_array   = zeros(m,1);
T_solution = zeros(n,n);
T_contour = zeros(5,m);

%%% Setting up the boundary conditions into arrays %%%
T_solution(1,1:n) = 0;         %Top Wall
T_solution(n,1:n) = 1;         %Bottom Wall
for i_Tl = 1 : 1 : n           %Left wall, applying the boundary condition
  y = ( n- i_Tl )/( n - 1 );
  T_solution(i_Tl, 1 ) = 1 - y^3;
end
```

```
for i_Tr = 1 : 1 : n           %Right wall, applying the boundary condition
  y = ( n - i_Tr )/( n - 1 );
  T_solution(i_Tr, n ) = 1 - sin(y*pi/2);
end
T_contour_matrix = permute(repmat(T_solution, 1,1,5), [3 1 2]);

%initializing coefficient matrices ( A and C) and boundary conditions array C
A  = zeros(m,m);
C = A;
B  = zeros(sm,sm);

%initiliazing parameters for time discretization
dt = 0.0001;
t = 0 : dt : 0.16;
r = dt /(h^2);

% arrays for figures and assignment questions
T_time_evolution = zeros(length(t),1);
T_vertical_profile = zeros(sm+2, 1);

%%% Setup coefficient matrix A, coefficient matrix for temperature at n+1 %%%
for ix = 1 : 1 : m
    for jx =1 : 1 : m
        if (ix == jx)
                A(ix,jx) = (1 + 2*r);
        elseif ( (ix == jx + 1) && ( (ix - 1) ~= sm * round( (ix-1)/sm) ) ) %RHS
                A(ix,jx) = -r/2;
        elseif ( (ix == jx - 1) && ( ix ~= sm*round(ix/sm) ) )
                A(ix,jx) = -r/2;
        elseif (ix == jx + sm)
                A(ix,jx) = -r/2;
        elseif (jx == ix + sm)
                A(ix,jx) = -r/2;
        else
                A(ix,jx) = 0;
        end
    end
end

%Setup boundary condition array
for iy = 1 : 1 : sm
    for jy = 1 : 1 : sm
        if (iy == 1) && (jy == 1)
            B(iy,jy) = r *(T_solution(1,2)+ T_solution(2,1)); %Tl + Tt
        elseif (iy == 1) && (jy == sm)
            B(iy,jy) = r *( T_solution(1,sm +1) + T_solution(2,sm+2));    %Tt + Tr
%RHS
        elseif (iy == sm) && (jy == sm)
            B(iy,jy) = r * (T_solution(sm+2,sm +1) + T_solution(sm+1,sm+2)); % Tb +
Tr
        elseif (iy == sm) && (jy == 1)
            B(iy,jy) = r * (T_solution(sm+1,1) + T_solution(sm+2,2)); % Tb + Tl
        elseif (iy == 1)&&(jy > 1 || jy < sm)
            B(iy,jy) = 0;
        elseif (jy == sm) && (iy > 1 || iy < sm)
            B(iy,jy) = r * T_solution(iy + 1, sm + 2);
        elseif (iy == sm) && ( jy > 1 || jy < sm)
            B(iy,jy) = r * 1;
        elseif (jy == 1) && ( iy > 1 || jy < sm)
```

```
            B(iy,jy) = r * T_solution(iy+1, 1);
        else
            B(iy,jy) = 0;
        end
    end
end
Bx = reshape(B,[],1);  %Convert matrix to array

% Setup coefficient matrix C for temperature array at n
for iz = 1 : 1 : m
    for jz =1 : 1 : m
        if (iz == jz)
                C(iz,jz) = (1 - 2*r);
        elseif ( (iz == jz + 1) && ( (iz - 1) ~= sm * round( (iz-1)/sm) ) ) %RHS
                C(iz,jz) = r/2;
        elseif ( (iz == jz - 1) && ( iz ~= sm*round(iz/sm) ) )
                C(iz,jz) = r/2;
        elseif (iz == jz + sm)
                C(iz,jz) = r/2;
        elseif (jz == iz + sm)
                C(iz,jz) = r/2;
        else
                C(iz,jz) = 0;
        end
    end
end

%%% Solution %%%
selected_time = [ 0.01, 0.02, 0.04, 0.08, 0.16 ];
counter = 1;
for l = 1 : length(t)   %time steps
     Xx = ( C*T_array + Bx ); %RHS of the equation where all the values are known
     T_array = A \ Xx; % LHS of the equation to get value of T at n+1
     T_time_evolution(l) = T_array(15*39+24);

     %storing values for contour plots at different t
     if any( selected_time == t(l))
       T_contour(counter,:) = T_array;
       T_contour_matrix(counter, 2:n-1, 2:n-1 ) = reshape( T_array , sm , sm);
       counter = counter + 1;
     endif



end
%Convert inner node temperature solution to matrix
T_matrix = reshape( T_array , sm , sm); %convert array to matrix


%Appending the inner node temperature solution matrix component to the whole
temperature solution matrix

T_solution(2:n-1, 2:n-1 ) = T_matrix;
T_vertical_profile = flip(T_solution( :, 17 ));



%initializing meshgrid with 41 evenly spaced points in length and width
x = linspace(0,L,n);
```

```
y = x;
[X,Y]=meshgrid(x,y);

%%% Plot %%%
f1 = figure( 'Position', [0, 0, 1000, 1200]);
for i = 1 : 1 :5
  subplot(3,2,i)
  contourf(X,flipud(Y),squeeze(T_contour_matrix(i,:,:)),10);
  ylabel('Y(dimensionless)','FontSize',10);
  xlabel('X(dimensionless)','FontSize',10);
  title(sprintf('Crank  Nicolson  scheme  t  =  %.2f  s',    selected_time(i)),
'FontSize',10);
  colorbar
end
print -dpng -color "-S1000,1200" ContourSubplots.png


f2 = figure
plot(t,T_time_evolution)
xlabel('Time(s)')
ylabel('Temperature(dimensionless)')
title('Temperature variation at x=y=0.4 against time')


f3 = figure
plot(y,T_vertical_profile)
xlabel('Y(dimensionless)')
ylabel('Temperature(dimensionless)')
title('Vertical T profile for x=0.4 at t=1
```