Ryerson University

Faculty of Engineering & Architectural Science

Department of Mechanical and Industrial Engineering

Course Number	AE8112
Course Title	Computational Fluid Dynamics and Heat Transfer
Semester/Year	Summer/Spring 2021
Instructor	Dr. Seth Dworkin

Problem Set 3

Submission Date	June 30, 2021
Programing Language Used	Fortran90

Student Name	Student Number
Ezeorah Godswill	501012886

Q1.a) Given a 1D convective/diffusion equation

$$\frac{d}{dx}(\rho u \phi) = \frac{d}{dx}\left(\Gamma \frac{d\phi}{dx}\right)$$

To solve this analytically, lets consider two points

using ODE, the above will be re-written as

$$\left(-\Gamma D^2 + \rho u D\right)\phi = 0 \qquad \text{where } D = \frac{d}{dx}$$

This will give an auxillary eqn of the form

$$-\Gamma m^2 + \rho u m = 0$$

This gives the root of m as (i.e., using almighty formular)

$$m = \frac{-\rho u \pm \sqrt{\rho^2 u^2}}{-2\Gamma}$$

$$m_1 = \frac{\rho u - \rho u}{2\Gamma} \quad \& \quad m_2 = \frac{\rho u + \rho u}{2\Gamma}$$

$$m_1 = 0 \quad \& \quad m_2 = \frac{\rho u}{\Gamma}$$

Q1.a| Cont'd.)

Since the roots are said to be real & distinct, which implies that the complementry fn (Type 1);

$$C.F. = C_1 e^{m_1 x} + C_2 e^{m_2 x}$$

so that

$$C.F. = C_1 e^{0x} + C_2 e^{\frac{\rho u}{\Gamma} x}$$

$$C.F. = C_1 + C_2 e^{\frac{\rho u x}{\Gamma}}$$

Since the RHS. of the ODE is zero, the Particular Integral

$$P.I = 0$$

So that the complete eqn becomes,

$$\phi(x) = C.F. + P.I$$

$$\phi(x) = C_1 + C_2 e^{\frac{\rho u x}{\Gamma}}$$

Solving the arbitary constants, we apply the B.Cs, where

$$\phi(0) = C_1 + C_2 = 1$$

$$C_2 = 1 - C_1$$

and

$$\phi(L) = C_1 + C_2 e^{\frac{\rho u L}{\Gamma}} = 0$$

Q1.a| cont'd.|

So that $\quad C_1 + e^{\frac{\rho u L}{\Gamma}} - C_1 e^{\frac{\rho u L}{\Gamma}} = 0$

Let $\frac{\rho u L}{\Gamma} = P_e$ (the Peclet number)

$$C_1 (e^{P_e} - 1) = e^{P_e}$$

$$C_1 = \frac{e^{P_e}}{e^{P_e} - 1}$$

Substituting this for C_2, we have

$$C_2 = 1 - \frac{e^{P_e}}{(e^{P_e} - 1)} = \frac{\cancel{e^{P_e}} - 1 - \cancel{e^{P_e}}}{(e^{P_e} - 1)}$$

$$C_2 = \frac{1}{1 - e^{P_e}}$$

Substituting these constants back for $\phi(x)$,

$$\phi(x) = \frac{e^{P_e}}{(e^{P_e} - 1)} + \frac{e^{\frac{P_e x}{L}}}{(1 - e^{P_e})}$$

$$\therefore \boxed{\phi(x) = \frac{e^{P_e} - e^{\frac{P_e x}{L}}}{e^{P_e} - 1}}$$

Q1.a| contd|

>> Now to discretize the governing eqn, with an equispaced grid, ~~for~~ this is done as follows

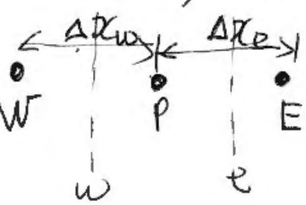
$$\frac{d}{dx}(\rho u \phi) = \frac{d}{dx}\left(\Gamma \frac{d\phi}{dx}\right)$$

Integrating for the C.V., where Γ & ρ and constants

$$\int_{\forall} \frac{d}{dx}(\rho u \phi)\, d\forall = \int_{\forall} \frac{d}{dx}\left(\Gamma \frac{d\phi}{dx}\right) d\forall$$

applying Gauss theorem, where $\int_{\forall} \nabla \cdot \vec{F} d\forall = \int_{S} \vec{F} \cdot \hat{n}\, ds$

For our 1-D case, integrating over



$$\int_{w}^{e} \left(\Gamma \frac{d\phi}{dx} - \rho u \phi\right) ds = 0$$

$$\left(\Gamma \frac{d\phi}{dx} - \rho u \phi\right)\Big|_e A_e - \left(\Gamma \frac{d\phi}{dx} - \rho u \phi\right)\Big|_w A_w = 0$$

where A is the bounding face area, and $A_e = A_w = A = \text{const.}$ (for an incompressible flow ($\nabla \cdot \vec{u} = 0$)).

Q1.a| cont'd.|

So that dividing the above by $A\Gamma$, we will get

$$\left(\frac{d\phi}{dx} - \frac{\rho u \phi}{\Gamma}\right)\Big|_e - \left(\frac{d\phi}{dx} - \frac{\rho u \phi}{\Gamma}\right)\Big|_w = 0$$

Since we know that discretizing with an assumption of a linear profile between points will lead to a problem, when $\Gamma = 0$.

Instead we will using

$$\phi_e = \left(\frac{1+\alpha_e}{2}\right)\phi_P + \left(\frac{1-\alpha_e}{2}\right)\phi_E$$

$$\phi_w = \left(\frac{1+\alpha_w}{2}\right)\phi_W + \left(\frac{1-\alpha_w}{2}\right)\phi_P$$

If we discretize the diffusion term as per usual, we get

$$\frac{\phi_E - \phi_P}{\Delta x_e} - \frac{\rho u}{\Gamma}\phi_e - \left(\frac{\phi_P - \phi_W}{\Delta x_w} - \frac{\rho u}{\Gamma}\phi_w\right) = 0$$

For equispaced grid $\Delta x_e = \Delta x_w = \Delta x$ & $\alpha_e = \alpha_w = \alpha$ and we have $\frac{\rho u}{\Gamma} = \frac{P_e}{L}$. so that putting the values of ϕ_e & ϕ_w into the above discretize eqn, we get, Q1.a | cont'd.

$$\frac{\phi_E - \phi_P}{\Delta x} - \frac{Pe}{L}\left(\frac{1+\alpha}{2}\right)\phi_P - \frac{Pe}{L}\left(\frac{1-\alpha}{2}\right)\phi_E + \frac{\phi_W - \phi_P}{\Delta x}$$

$$+ \frac{Pe}{L}\left(\frac{1+\alpha}{2}\right)\phi_W + \frac{Pe}{L}\left(\frac{1-\alpha}{2}\right)\phi_P = 0$$

re-arranging we get

$$\left[\frac{1}{\Delta x} + \frac{Pe}{2L}(1+\alpha) + \frac{1}{\Delta x} - \frac{Pe}{2L}(1-\alpha)\right]\phi_P = \left[\frac{1}{\Delta x} - \frac{Pe}{2L}(1-\alpha)\right]\phi_E$$

$$+ \left[\frac{1}{\Delta x} + \frac{Pe}{2}(1+\alpha)\right]\phi_W$$

simplifying, gives

$$\left[\frac{1}{\Delta x} - \frac{Pe}{2L}(1-\alpha)\right]\phi_E - \left[\frac{2}{\Delta x} + \frac{Pe\,\alpha}{L}\right]\phi_P + \left[\frac{1}{\Delta x} + \frac{Pe}{2L}(1+\alpha)\right]\phi_W = 0 \qquad (1.2)$$

$$\boxed{a_{Ea}\phi_E - a_{Pa}\phi_P + a_{Wa}\phi_W = 0}$$

>> For upwinded scheme; put $\alpha = 1$ into eqn (1.2),

$$\left[\frac{1}{\Delta x}\right]\phi_E - \left[\frac{2}{\Delta x} + \frac{Pe}{L}\right]\phi_P + \left[\frac{1}{\Delta x} + \frac{Pe}{L}\right]\phi_W = 0 \qquad (1.3)$$

$$\boxed{a_E\phi_E - a_P\phi_P + a_W\phi_P = 0}$$

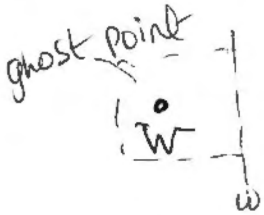
>> And For Central difference scheme, put $\alpha = 0$ into eqn (1.2), we get,

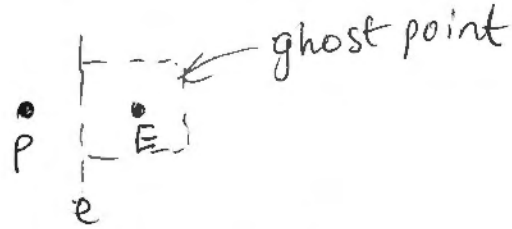
Q 1.a| Cont'd.!

$$\left[\frac{1}{\Delta x}+\frac{P_e}{2L}\right]\phi_E-\left[\frac{2}{\Delta x}\right]\phi_P+\left[\frac{1}{\Delta x}+\frac{P_e}{2L}\right]\phi_W=0 \quad (1.4)$$

$$\boxed{\frac{a_P}{2}\phi_E-2a_E\phi_P+\frac{a_P}{2}\phi_W=0}$$

>> Applying the B.C.S using ghost points





$$\phi(0)=\frac{\phi_P+\phi_W}{2}=1$$

$$\phi_W=2-\phi_P$$

$$\phi(L)=\frac{\phi_E+\phi_P}{2}=0$$

$$\phi_E=0-\phi_P$$

In order to eliminate the ghost point let ϕ_E be the Subject. as given above.

>> First C.V. [for $\alpha=1$]

Put the value ϕ_W into eqn (1.3), we get

$$\left[\frac{1}{\Delta x}\right]\phi_E-\left[\frac{2}{\Delta x}+\frac{P_e}{L}\right]\phi_P+\left[\frac{1}{\Delta x}+\frac{P_e}{L}\right](2-\phi_P)=0$$

Simplifying

$$\left[\frac{1}{\Delta x}\right]\phi_E-\left[\frac{3}{\Delta x}+\frac{2P_e}{L}\right]\phi_P=-\left[\frac{2}{\Delta x}+\frac{2P_e}{L}\right] \quad (1.4)$$

$$\boxed{a_E\phi_E-(3a_E+2a_b)\phi_P=-2a_W}$$

Q 1.a | Cont'd:

>> Last C.V. [for $\alpha = 1$]

Putting the value of ϕ_E into eqn (1.3) we get

$$-\left[\frac{3}{\Delta x} + \frac{Pe}{L}\right]\phi_P + \left[\frac{1}{\Delta x} + \frac{Pe}{L}\right]\phi_W = 0$$

$$\boxed{-(3a_E + a_b)\phi_P + a_W \phi_W = 0}$$

>> First C.V. [for $\alpha = 0$]

Putting the value of ϕ_W into eqn (1.4), we get

$$\left[\frac{1}{\Delta x} + \frac{Pe}{2L}\right]\phi_E - \left[\frac{2}{\Delta x}\right]\phi_P + \left[\frac{1}{\Delta x} + \frac{Pe}{2L}\right](2 - \phi_P) = 0$$

Simplifying

$$\left[\frac{1}{\Delta x} + \frac{Pe}{2L}\right]\phi_E - \left[\frac{3}{\Delta x} + \frac{Pe}{2L}\right]\phi_P = -\left[\frac{2}{\Delta x} + \frac{Pe}{L}\right]$$

$$\boxed{\frac{a_P}{2}\phi_E - \left(3a_E + \frac{a_b}{2}\right)\phi_P = -a_P}$$

>> Last C.V. (for $\alpha = 0$); putting the value ϕ_E into eqn (1.4), we get,

$$-\left[\frac{3}{\Delta x} + \frac{Pe}{2L}\right]\phi_P + \left[\frac{1}{\Delta x} + \frac{Pe}{2L}\right]\phi_W = 0$$

$$\boxed{-\left(3a_E + \frac{a_b}{2}\right)\phi_P + a_W \phi_W = 0}$$

These eqns in boxes are used in fortran 90.

```fortran
!****************Begin Header****************************************************

!This program was written by Godswill Ezeorah, Student Number: 501012886 on June 20, 2021.
!This program solves a linear/non-linear equation using finite volume method
!and was written as a solution to AE8112 PS3 q1a
!****************************End Header*****************************************
program finit_vol
    !Variable declaration
    implicit none
    DOUBLE PRECISION, dimension(:,:), ALLOCATABLE:: Ag
    DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE:: bg, Phi_x, Phi_ex, x1
    DOUBLE PRECISION, PARAMETER :: L=1, Pe=1.09639004471!length of the domain and Peclet number
    DOUBLE PRECISION  :: L2, dx1, alpha
    integer :: N, ii, jj
    open(1, file = 'PS3_Q1a_error.txt', status = 'unknown')
    open(2, file = 'PS3_Q1a_soln.txt', status = 'unknown')
    open(3, file = 'PS3_Q1b.txt', status = 'unknown')
    open(4, file = 'PS3_Q1b_pe.txt', status = 'unknown')
    write(1,5) 'L2'
    write(3,7) 'L2'
    write(4,5) 'L2'
    do ii = 1, 3
        N=32
        if ( ii==1 ) then !For central difference scheme
            alpha=0
        elseif (ii==2) then !For upwinded scheme
            alpha=1
        else
            alpha=pe**2/(pe**2+5)
        end if
        do while(n<=128)  !This will run for N=32, 64 and 128
            ALLOCATE(Ag(n,n),bg(n),Phi_x(n),Phi_ex(n),x1(n))
            call creat_eTDMA(Ag,bg,Phi_x,Phi_ex,dx1,x1)
            !Calculating L2 error term
            L2=sum((Phi_x(1:n)-Phi_ex(1:n))**2)/N
            !outputing the results
            print 3, 'For number of grid points, n =',n
            print 4, 'For α =',alpha
            write(*,1)  "ϕ(x) = ",Phi_x
            write(*,1)  "ϕ_exact = ",Phi_ex
            write(*,2) "L2 = ", L2
            1 format(a6,128f8.3)
            2 format(a7,E9.3)
            3 format(a30,i7)
            4 format(a7,f7.5)
```

```
        if ( n==64 ) then
            if ( alpha==0 ) then
                write(2,*) 'solution for α =', alpha
            elseif (alpha==1) then
                write(2,*) 'solution for α =', alpha
            else
                write(2,*) 'solution for α =', alpha
            end if
            do jj = 1, n
                write(2,*) x1(jj), Phi_x(jj)
            end do
        end if
        !writing the nine L2 error terms
        write(1,6) 'For No. of CVs. = ',N,', And α = ',alpha, L2
        write(3,8) 'Δx = ',dx1,', And α = ',alpha, L2
        if ( alpha==0 .or. alpha==1 ) then
            write(4,10) 'Δx = ',log(dx1),', And α = ',alpha, log(L2)
        end if
        N=n+n
        DEALLOCATE(Ag, bg, Phi_x, Phi_ex, x1)
    end do
end do
5 format(40x,a2)
7 format(30x,a2)
6 format(a19,i3,a11,f5.3,2x,E9.3)
8 format(a6,f7.5,a11,f5.3,2x,E9.3)
10 format(a6,E12.5,a11,f5.3,2x,E10.3)
close(1);close(2);close(3);close(4)

contains
!*****************************************************************************************************

subroutine creat_eTDMA(A,b,phi,phi_e,dx,x)
    implicit none
    DOUBLE PRECISION, dimension(n,n), INTENT(OUT):: A
    DOUBLE PRECISION, DIMENSION(n), INTENT(OUT):: b, phi, phi_e, x
    DOUBLE PRECISION, INTENT(OUT) :: dx
    DOUBLE PRECISION  :: ae, ap, aw, xi, af, al, ab
    integer :: i
    !This subroutine generates the tri-diagonal matric using our derived equations

    !Initializing Variables
    A=0; b=0
    dx=L/n
    xi=dx/2
```

```
    ab=((2/dx)+(Pe/L))
    af=(3/dx)+(Pe/(2*L))*(1+alpha)
    al=(3/dx)-(Pe/(2*L))*(1-alpha)
    ae=(1/dx)-(Pe/(2*L))*(1-alpha)
    ap=(2/dx)+(Pe*alpha/L)
    aw=(1/dx)+(Pe/(2*L))*(1+alpha)
    do i = 1, n  !This loop Matrix composition of the given problem
        !for the the first gridpoint, applying B.Cs ϕ(0)=1
        if ( i==1 ) then
            A(i,i)=-(aw+ap)
            A(i,i+1)=ae
            b(i)=-(2*aw)
        !for the the intermediate gridpoint
        else if ( i<n ) then
            A(i,i-1)=aw
            A(i,i)=-ap
            A(i,i+1)=ae
            b(i)=0
        !for the the last gridpoint, applying B.Cs ϕ(L)=0
        else
            A(n,n-1)=aw
            A(n,n)=-(ae+ap)
            b(n)=0
        end if
        phi_e(i)=(exp(pe)-exp(pe*xi/L))/(exp(pe)-1) !This is ϕ_exact from our analytical solution
        x(i)=xi
        xi=xi+dx
    end do
    call tdma(A,b,phi)
end subroutine creat_eTDMA
!*****************************************************************************************************

!*****************************************************************************************************
subroutine tdma(A,b1,x)
    implicit none
    DOUBLE PRECISION, dimension(n,n), INTENT(IN):: A
    DOUBLE PRECISION, dimension(n), INTENT(IN):: b1
    DOUBLE PRECISION, DIMENSION(n), INTENT(OUT) :: x
    DOUBLE PRECISION, DIMENSION(n):: b, e, f, g
    INTEGER :: k
    !This subroutine solves a tri-diagonal linear system using the Thomas Algorithm

    b=b1
    x=0
   !extracting e, f and g array
```

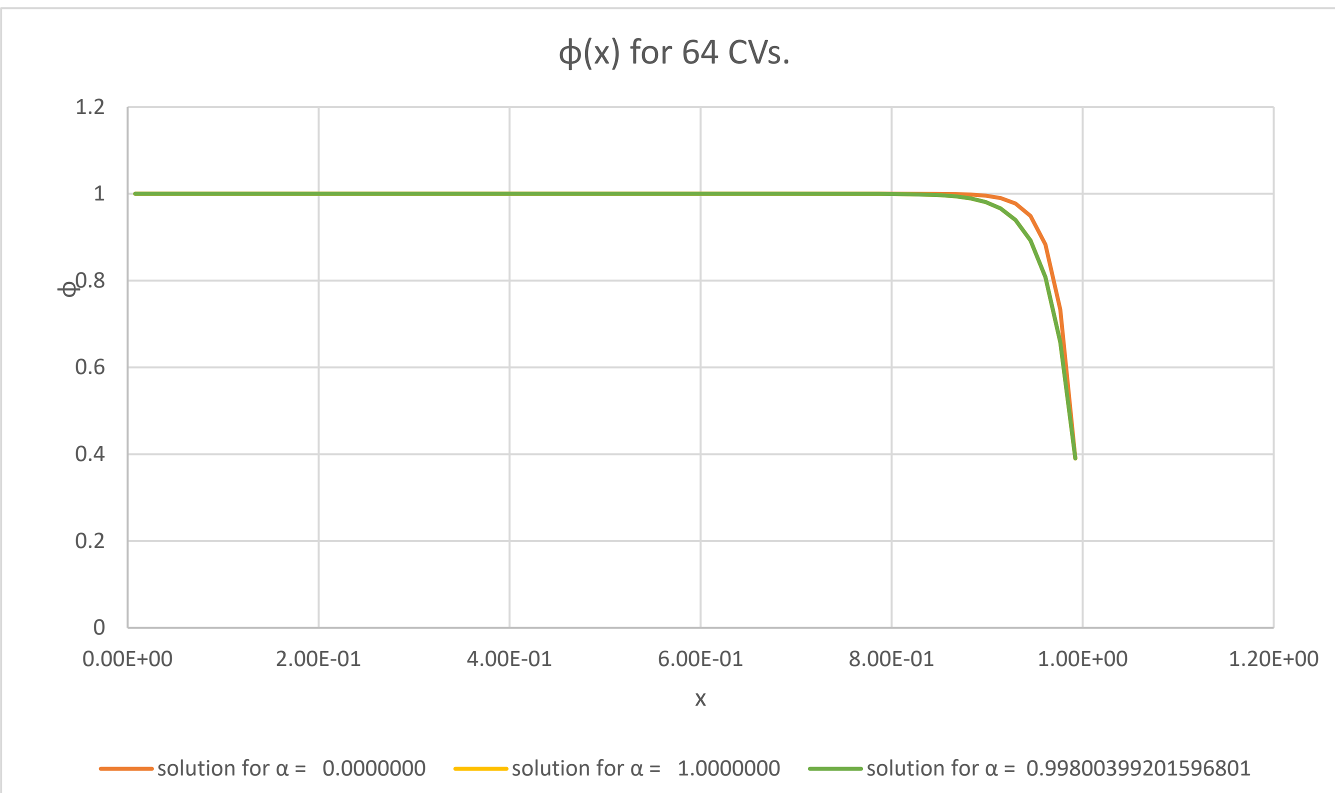
```fortran
    do k=1,n-1
        e(k+1)=A(k+1,k)
        g(k)=A(k,k+1)
    end do
    do  k = 1,n
        f(k)=A(k,k)
    end do
   !Decomposition
    do k = 2,n
        e(k) = e(k)/f(k-1)
        f(k) = f(k) - e(k)*g(k-1)
    end do
   !Forward Substitution
    do k = 2,n
        b(k) = b(k) - e(k)*b(k-1)
    end do
   !Backward Substitution
    x(n)=b(n)/f(n)
    do k = n-1,1,-1  !(step size of -1)
        x(k) = (b(k) - g(k)*x(k+1))/f(k)
    end do
end subroutine tdma
!****************************************************************************************************

end program finit_vol
```

For number of grid points, n = 32
For α 0.00000
φ(x) 0.991 0.974 0.955 0.936 0.916 0.896 0.875 0.853 0.831 0.807 0.783 0.758 0.732 0.705 0.677 0.649 0.619 0.588 0.556 0.523 0.489 0.454 0.417 0.380 0.340 0.300 0.258 0.215 0.170 0.123 0.075 0.026
φ_exa 0.991 0.974 0.955 0.936 0.916 0.896 0.875 0.853 0.830 0.807 0.783 0.758 0.732 0.705 0.677 0.648 0.619 0.588 0.556 0.523 0.489 0.454 0.417 0.379 0.340 0.300 0.258 0.215 0.170 0.123 0.075 0.026
L2 = 0.221E-07
For number of grid points, n = 64
For α 0.00000
φ(x) 0.996 0.987 0.978 0.969 0.960 0.950 0.941 0.931 0.921 0.911 0.901 0.891 0.880 0.869 0.859 0.847 0.836 0.825 0.813 0.801 0.789 0.777 0.764 0.751 0.738 0.725 0.712 0.698 0.684 0.670 0.656 0.641 0.626 0.611 0.596 0.580 0.564 0.548 0.532 0.515 0.498 0.480 0.463 0.445 0.426 0.408 0.389 0.370 0.350 0.330 0.310 0.290 0.269 0.247 0.226 0.204 0.181 0.158 0.135 0.111 0.087 0.063 0.038 0.013
φ_exa 0.996 0.987 0.978 0.969 0.960 0.950 0.941 0.931 0.921 0.911 0.901 0.891 0.880 0.869 0.859 0.847 0.836 0.825 0.813 0.801 0.789 0.777 0.764 0.751 0.738 0.725 0.712 0.698 0.684 0.670 0.656 0.641 0.626 0.611 0.596 0.580 0.564 0.548 0.531 0.515 0.498 0.480 0.463 0.445 0.426 0.408 0.389 0.370 0.350 0.330 0.310 0.289 0.269 0.247 0.226 0.203 0.181 0.158 0.135 0.111 0.087 0.063 0.038 0.013
L2 = 0.138E-08
For number of grid points, n = 128
For α 0.00000
φ(x) 0.998 0.994 0.989 0.985 0.980 0.976 0.971 0.967 0.962 0.957 0.953 0.948 0.943 0.939 0.934 0.929 0.924 0.919 0.914 0.909 0.904 0.899 0.893 0.888 0.883 0.878 0.872 0.867 0.861 0.856 0.850 0.845 0.839 0.833 0.828 0.822 0.816 0.810 0.804 0.798 0.792 0.786 0.780 0.774 0.767 0.761 0.755 0.748 0.742 0.735 0.729 0.722 0.715 0.708 0.702 0.695 0.688 0.681 0.674 0.667 0.659 0.652 0.645 0.637 0.630 0.623 0.615 0.607 0.600 0.592 0.584 0.576 0.568 0.560 0.552 0.544 0.536 0.527 0.519 0.510 0.502 0.493 0.485 0.476 0.467 0.458 0.449 0.440 0.431 0.422 0.413 0.403 0.394 0.384 0.375 0.365 0.355 0.345 0.335 0.325 0.315 0.305 0.295 0.284 0.274 0.263 0.253 0.242 0.231 0.220 0.209 0.198 0.187 0.175 0.164 0.152 0.141 0.129 0.117 0.105 0.093 0.081 0.069 0.057 0.044 0.032 0.019 0.006
φ_exa 0.998 0.994 0.989 0.985 0.980 0.976 0.971 0.967 0.962 0.957 0.953 0.948 0.943 0.939 0.934 0.929 0.924 0.919 0.914 0.909 0.904 0.899 0.893 0.888 0.883 0.878 0.872 0.867 0.861 0.856 0.850 0.845 0.839 0.833 0.828 0.822 0.816 0.810 0.804 0.798 0.792 0.786 0.780 0.773 0.767 0.761 0.755 0.748 0.742 0.735 0.728 0.722 0.715 0.708 0.702 0.695 0.688 0.681 0.674 0.667 0.659 0.652 0.645 0.637 0.630 0.622 0.615 0.607 0.600 0.592 0.584 0.576 0.568 0.560 0.552 0.544 0.536 0.527 0.519 0.510 0.502 0.493 0.485 0.476 0.467 0.458 0.449 0.440 0.431 0.422 0.413 0.403 0.394 0.384 0.375 0.365 0.355 0.345 0.335 0.325 0.315 0.305 0.295 0.284 0.274 0.263 0.253 0.242 0.231 0.220 0.209 0.198 0.187 0.175 0.164 0.152 0.141 0.129 0.117 0.105 0.093 0.081 0.069 0.057 0.044 0.032 0.019 0.006
L2 = 0.862E-10

For number of grid points, n = 32
For α 1.00000
φ(x) 0.991 0.973 0.955 0.936 0.916 0.895 0.874 0.852 0.829 0.806 0.781 0.756 0.730 0.703 0.675 0.646 0.617 0.586 0.554 0.521 0.487 0.452 0.415 0.378 0.339 0.298 0.257 0.213 0.169 0.123 0.075 0.026
φ_exa 0.991 0.974 0.955 0.936 0.916 0.896 0.875 0.853 0.830 0.807 0.783 0.758 0.732 0.705 0.677 0.648 0.619 0.588 0.556 0.523 0.489 0.454 0.417 0.379 0.340 0.300 0.258 0.215 0.170 0.123 0.075 0.026
L2 = 0.216E-05
For number of grid points, n = 64
For α 1.00000
φ(x) 0.996 0.987 0.978 0.969 0.960 0.950 0.941 0.931 0.921 0.911 0.901 0.890 0.880 0.869 0.858 0.847 0.835 0.824 0.812 0.800 0.788 0.776 0.763 0.750 0.737 0.724 0.711 0.697 0.683 0.669 0.655 0.640 0.625 0.610 0.595 0.579 0.563 0.547 0.530 0.514 0.497 0.479 0.462 0.444 0.425 0.407 0.388 0.369 0.349 0.329 0.309 0.289 0.268 0.247 0.225 0.203 0.181 0.158 0.135 0.111 0.087 0.063 0.038 0.013
φ_exa 0.996 0.987 0.978 0.969 0.960 0.950 0.941 0.931 0.921 0.911 0.901 0.891 0.880 0.869 0.859 0.847 0.836 0.825 0.813 0.801 0.789 0.777 0.764 0.751 0.738 0.725 0.712 0.698 0.684 0.670 0.656 0.641 0.626 0.611 0.596 0.580 0.564 0.548 0.531 0.515 0.498 0.480 0.463 0.445 0.426 0.408 0.389 0.370 0.350 0.330 0.310 0.289 0.269 0.247 0.226 0.203 0.181 0.158 0.135 0.111 0.087 0.063 0.038 0.013
L2 = 0.598E-06
For number of grid points, n = 128

For α 1.00000
φ(x) 0.998 0.993 0.989 0.985 0.980 0.976 0.971 0.967 0.962 0.957 0.953 0.948 0.943 0.938 0.933 0.929 0.924 0.919 0.914 0.909 0.903 0.898 0.893 0.888 0.883 0.877 0.872 0.866 0.861 0.855 0.850 0.844 0.839 0.833 0.827 0.821 0.815 0.810 0.804 0.798 0.792 0.785 0.779 0.773 0.767 0.760 0.754 0.748 0.741 0.735 0.728 0.721 0.715 0.708 0.701 0.694 0.687 0.680 0.673 0.666 0.659 0.652 0.644 0.637 0.629 0.622 0.614 0.607 0.599 0.591 0.583 0.576 0.568 0.560 0.551 0.543 0.535 0.527 0.518 0.510 0.501 0.493 0.484 0.475 0.467 0.458 0.449 0.440 0.431 0.421 0.412 0.403 0.393 0.384 0.374 0.364 0.355 0.345 0.335 0.325 0.315 0.305 0.294 0.284 0.273 0.263 0.252 0.241 0.231 0.220 0.209 0.198 0.186 0.175 0.164 0.152 0.141 0.129 0.117 0.105 0.093 0.081 0.069 0.057 0.044 0.032 0.019 0.006
φ_exa 0.998 0.994 0.989 0.985 0.980 0.976 0.971 0.967 0.962 0.957 0.953 0.948 0.943 0.939 0.934 0.929 0.924 0.919 0.914 0.909 0.904 0.899 0.893 0.888 0.883 0.878 0.872 0.867 0.861 0.856 0.850 0.845 0.839 0.833 0.828 0.822 0.816 0.810 0.804 0.798 0.792 0.786 0.780 0.773 0.767 0.761 0.755 0.748 0.742 0.735 0.728 0.722 0.715 0.708 0.702 0.695 0.688 0.681 0.674 0.667 0.659 0.652 0.645 0.637 0.630 0.622 0.615 0.607 0.600 0.592 0.584 0.576 0.568 0.560 0.552 0.544 0.536 0.527 0.519 0.510 0.502 0.493 0.485 0.476 0.467 0.458 0.449 0.440 0.431 0.422 0.413 0.403 0.394 0.384 0.375 0.365 0.355 0.345 0.335 0.325 0.315 0.305 0.295 0.284 0.274 0.263 0.253 0.242 0.231 0.220 0.209 0.198 0.187 0.175 0.164 0.152 0.141 0.129 0.117 0.105 0.093 0.081 0.069 0.057 0.044 0.032 0.019 0.006
L2 = 0.157E-06
For number of grid points, n = 32
For α 0.19382
φ(x) 0.991 0.974 0.955 0.936 0.916 0.896 0.875 0.853 0.830 0.807 0.783 0.758 0.732 0.705 0.677 0.648 0.618 0.588 0.556 0.523 0.489 0.453 0.417 0.379 0.340 0.300 0.258 0.214 0.170 0.123 0.075 0.026
φ_exa 0.991 0.974 0.955 0.936 0.916 0.896 0.875 0.853 0.830 0.807 0.783 0.758 0.732 0.705 0.677 0.648 0.619 0.588 0.556 0.523 0.489 0.454 0.417 0.379 0.340 0.300 0.258 0.215 0.170 0.123 0.075 0.026
L2 = 0.370E-07
For number of grid points, n = 64
For α 0.19382
φ(x) 0.996 0.987 0.978 0.969 0.960 0.950 0.941 0.931 0.921 0.911 0.901 0.891 0.880 0.869 0.858 0.847 0.836 0.825 0.813 0.801 0.789 0.776 0.764 0.751 0.738 0.725 0.712 0.698 0.684 0.670 0.656 0.641 0.626 0.611 0.596 0.580 0.564 0.548 0.531 0.515 0.497 0.480 0.462 0.445 0.426 0.408 0.389 0.370 0.350 0.330 0.310 0.289 0.268 0.247 0.225 0.203 0.181 0.158 0.135 0.111 0.087 0.063 0.038 0.013
φ_exa 0.996 0.987 0.978 0.969 0.960 0.950 0.941 0.931 0.921 0.911 0.901 0.891 0.880 0.869 0.859 0.847 0.836 0.825 0.813 0.801 0.789 0.777 0.764 0.751 0.738 0.725 0.712 0.698 0.684 0.670 0.656 0.641 0.626 0.611 0.596 0.580 0.564 0.548 0.531 0.515 0.498 0.480 0.463 0.445 0.426 0.408 0.389 0.370 0.350 0.330 0.310 0.289 0.269 0.247 0.226 0.203 0.181 0.158 0.135 0.111 0.087 0.063 0.038 0.013
L2 = 0.156E-07
For number of grid points, n = 128
For α 0.19382
φ(x) 0.998 0.994 0.989 0.985 0.980 0.976 0.971 0.967 0.962 0.957 0.953 0.948 0.943 0.938 0.934 0.929 0.924 0.919 0.914 0.909 0.904 0.899 0.893 0.888 0.883 0.877 0.872 0.867 0.861 0.856 0.850 0.845 0.839 0.833 0.827 0.822 0.816 0.810 0.804 0.798 0.792 0.786 0.780 0.773 0.767 0.761 0.754 0.748 0.742 0.735 0.728 0.722 0.715 0.708 0.701 0.695 0.688 0.681 0.674 0.666 0.659 0.652 0.645 0.637 0.630 0.622 0.615 0.607 0.600 0.592 0.584 0.576 0.568 0.560 0.552 0.544 0.536 0.527 0.519 0.510 0.502 0.493 0.485 0.476 0.467 0.458 0.449 0.440 0.431 0.422 0.412 0.403 0.394 0.384 0.375 0.365 0.355 0.345 0.335 0.325 0.315 0.305 0.295 0.284 0.274 0.263 0.253 0.242 0.231 0.220 0.209 0.198 0.187 0.175 0.164 0.152 0.141 0.129 0.117 0.105 0.093 0.081 0.069 0.057 0.044 0.032 0.019 0.006
φ_exa 0.998 0.994 0.989 0.985 0.980 0.976 0.971 0.967 0.962 0.957 0.953 0.948 0.943 0.939 0.934 0.929 0.924 0.919 0.914 0.909 0.904 0.899 0.893 0.888 0.883 0.878 0.872 0.867 0.861 0.856 0.850 0.845 0.839 0.833 0.828 0.822 0.816 0.810 0.804 0.798 0.792 0.786 0.780 0.773 0.767 0.761 0.755 0.748 0.742 0.735 0.728 0.722 0.715 0.708 0.702 0.695 0.688 0.681 0.674 0.667 0.659 0.652 0.645 0.637 0.630 0.622 0.615 0.607 0.600 0.592 0.584 0.576 0.568 0.560 0.552 0.544 0.536 0.527 0.519 0.510 0.502 0.493 0.485 0.476 0.467 0.458 0.449 0.440 0.431 0.422 0.413 0.403 0.394 0.384 0.375 0.365 0.355 0.345 0.335 0.325 0.315 0.305 0.295 0.284 0.274 0.263 0.253 0.242 0.231 0.220 0.209 0.198 0.187 0.175 0.164 0.152 0.141 0.129 0.117 0.105 0.093 0.081 0.069 0.057 0.044 0.032 0.019 0.006
L2 = 0.497E-08

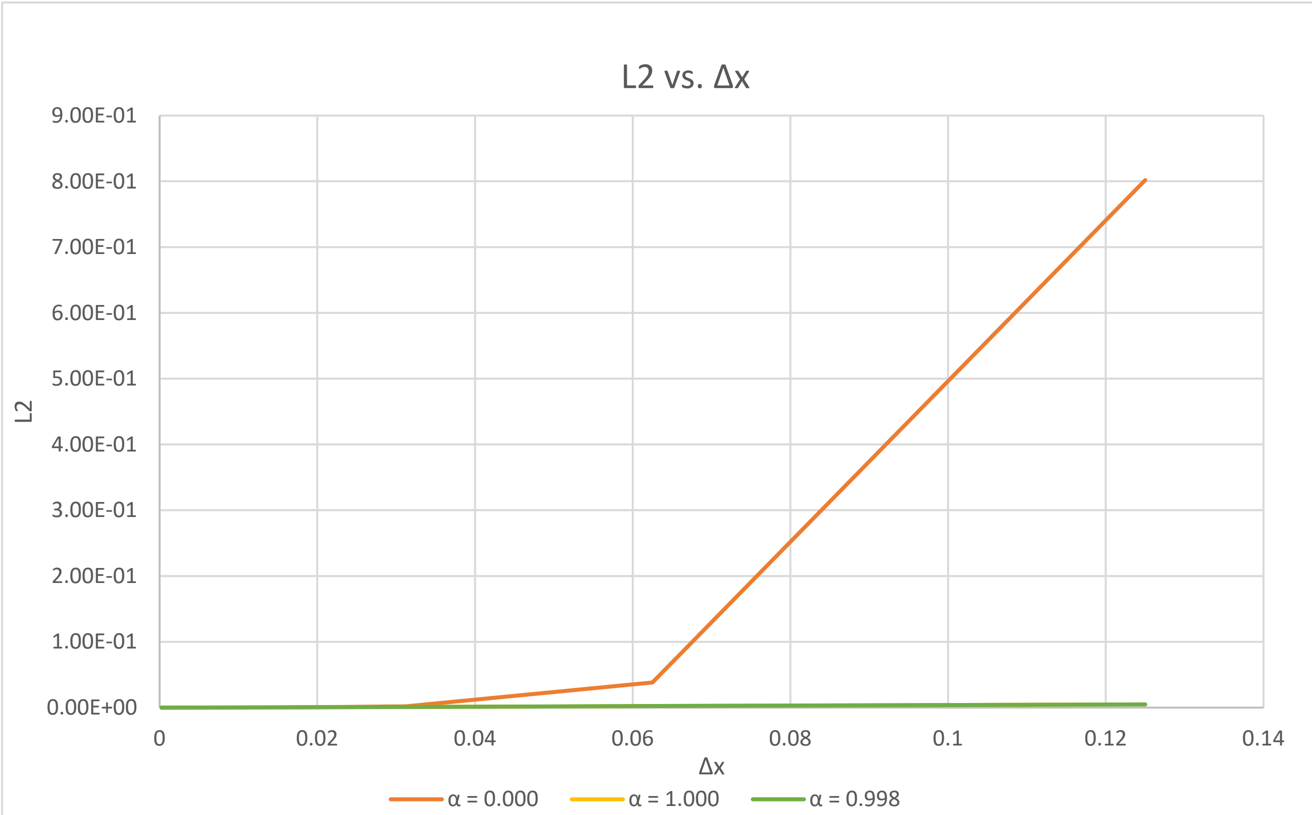
	L2
For No. of CVs. = 32, And α = 0.000	0.00194
For No. of CVs. = 32, And α = 1.000	0.000975
For No. of CVs. = 32, And α = 0.998	0.000968
For No. of CVs. = 64, And α = 0.000	0.000113
For No. of CVs. = 64, And α = 1.000	0.000384
For No. of CVs. = 64, And α = 0.998	0.000382
For No. of CVs. = 128, And α = 0.000	0.00000692
For No. of CVs. = 128, And α = 1.000	0.000131
For No. of CVs. = 128, And α = 0.998	0.00013

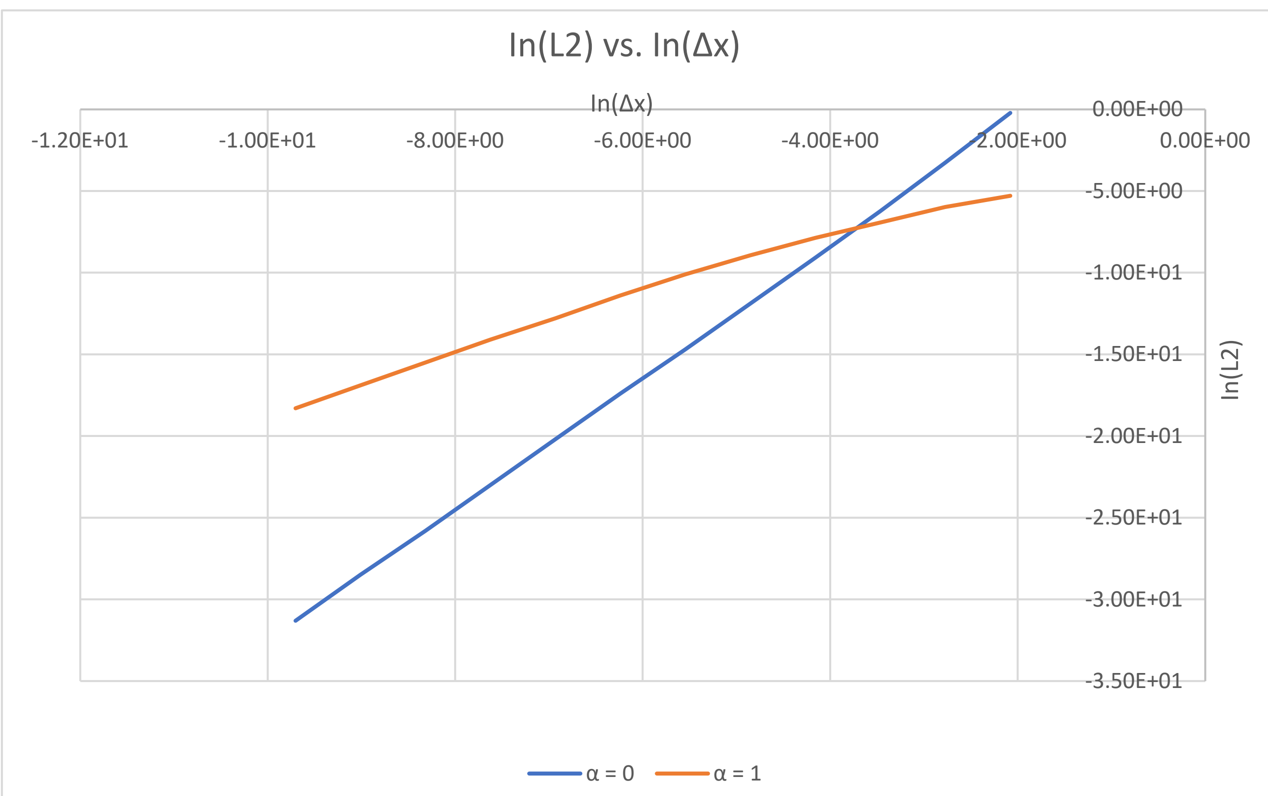


Q1.a| cont'd.|

→ From the above Plot, I observe that the Solution for $\alpha=0$ is more accurate than that of $\alpha \simeq 1$ at 64 C.Vs.. This is true, since Δx is small enough at 64 C.Vs., and thus the centered difference scheme usually yields good approximation for our solution as $\Delta x \to 0$.

Q1.b| I expected the accuracy for all three cases to increase as Δx decreases. And that the solution with $\alpha=0$ is more accurate than the other two cases, if Δx is fine enough (small) (i.e., $\Delta x \to 0$). Otherwise the other two cases yields better accuracy, which $\alpha = P_e^2(P_e^2+5)$ is slightly more accurate than $\alpha=1$.


L2 vs. Δx
9.00E-01
8.00E-01
7.00E-01
6.00E-01
5.00E-01
4.00E-01
3.00E-01
2.00E-01
1.00E-01
0.00E+00
L2
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
Δx
α = 0.000
α = 1.000
α = 0.998


ln(L2) vs. ln(Δx)
ln(Δx)
-1.20E+01
-1.00E+01
-8.00E+00
-6.00E+00
-4.00E+00
-2.00E+00
0.00E+00
0.00E+00
-5.00E+00
-1.00E+01
-1.50E+01
-2.00E+01
-2.50E+01
-3.00E+01
-3.50E+01
ln(L2)
α = 0
α = 1

Q1.b| cont'd.:

>> The Pe value where $L_2(\alpha=0) \simeq L_2(\alpha=1)$ can be extrapulated, by plotting $\ln(\Delta x)$ vs. $\ln(L_2)$ for both $\alpha=0$ and $\alpha=1$ on the same plot in Excel (This shown above).

The point of intersection can be calculated using,

$$\ln(\Delta x)_{int} = \frac{C_{\alpha_1} - C_{\alpha_0}}{m_{\alpha_0} - m_{\alpha_1}}$$

where c – represents the intercept for each case of α.

m – represents the slope for each case of α.

The Slope and the Intercept functions are inbuilt in Excel, and are used to compute the value as,

$$\ln(\Delta x)_{int} = -3.82$$

so that taking the exponent of both sides, we get

$$\Delta x = 0.0219278$$

So that multiplying $\frac{\rho u L}{\Gamma} = \frac{Pe_L}{L}$ by Δx, we have

$$Pe_{\Delta x} = \frac{50}{1}(0.0219278)(1)$$

$\therefore$ The Pe value for $L_2(\alpha=0) \simeq L_2(\alpha=1)$ is $\boxed{Pe_{\Delta x} = 1.09629}$

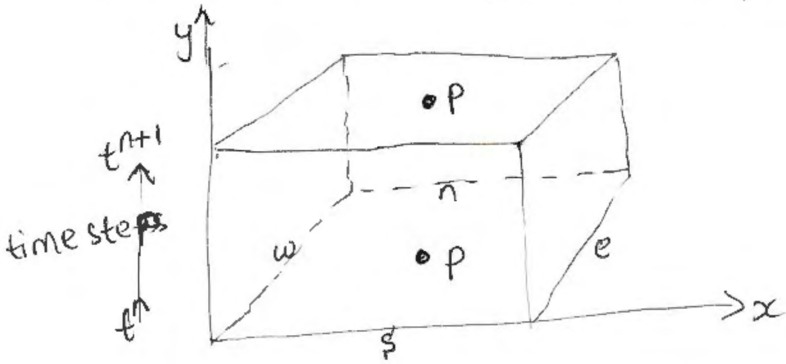
Q1c| The solution generated with $\alpha = 0$ is found not to oscilite for Pe ranging from 0. to 4

$$\boxed{0 < Pe < 4}$$

Q2| From the given governing eq$\underline{n}$, re-written in 2-D as,

$$\frac{dT}{dt} = \frac{\lambda}{\rho C_p}\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) + \frac{S}{\rho C_p}$$

If we have a C.V. in the form



Integrating for the above C.V., we get

$$\int_s^n\int_w^e\int_{t^n}^{t^{n+1}} \frac{dT}{dt}\,dt\,\partial x\,\partial y = \frac{\lambda}{\rho C_p}\int_{t^n}^{t^{n+1}}\int_s^n\int_w^e \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)\partial x\,\partial y\,dt + \int_{t^n}^{t^{n+1}}\int_s^n\int_w^e \frac{S}{\rho C_p}\,\partial x\,\partial y\,dt$$

let $\alpha = \frac{\lambda}{\rho C_p}$ and we have the source term as,

$$S = \frac{2 h_{side} A\,(T_\infty - T(x,y,t))}{V}$$

where the plate volume, $V = A\,t_K$ with t_K as the plate thickness.

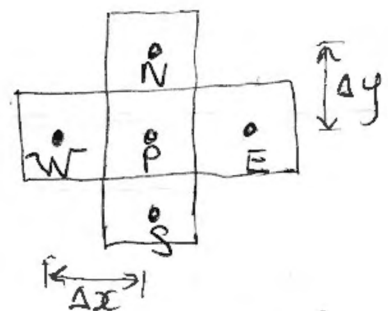
Q2 cont'd.:

so that $\bar{S} = \dfrac{2\, h_{side}\, A\, (T_\infty - T_P)}{A\, t_K} = \dfrac{2\, h_{sid}\, (T_\infty - T_P)}{t_K}$

So that the above integration for fully implicit scheme gives,

$$\left(T_P^{n+1} - T_P^{n}\right)\Delta x \Delta y = \alpha\left(\frac{\partial T}{\partial x}\Big|_e \Delta y - \frac{\partial T}{\partial x}\Big|_w \Delta y + \frac{\partial T}{\partial y}\Big|_n \Delta x - \frac{\partial T}{\partial y}\Big|_s \Delta x\right)\Bigg|^{n+1} \Delta t + \frac{2\, h_{side}(T_\infty - T_P)\,\Delta x \Delta y \Delta t}{\rho c_p t_K}\Bigg|^{n+1}$$

so that discretizing for give c.v. of equispaced grids



we get,

$$\left(T_P^{n+1} - T_P^{n}\right)\Delta x \Delta y = \alpha \Delta t\left[\frac{\left(T_E^{n+1} - T_P^{n+1}\right)}{\Delta x}\Delta y - \frac{\left(T_P^{n+1} - T_W^{n+1}\right)}{\Delta x}\Delta y + \frac{\left(T_N^{n+1} - T_P^{n+1}\right)}{\Delta y}\Delta x - \frac{\left(T_P^{n+1} - T_S^{n+1}\right)}{\Delta y}\Delta x\right] + \frac{2 h_{side}\left(T_\infty - T_P^{n+1}\right)\Delta x \Delta y \Delta t}{\rho c_p t_K}$$

collecting like temperature terms,

$$\left[\frac{\alpha \Delta t \Delta y}{\Delta x}\right]T_W^{n+1} + \left[\frac{\alpha \Delta t \Delta x}{\Delta y}\right]T_S^{n+1} - \left[\Delta x \Delta y + \frac{2\alpha \Delta t \Delta y}{\Delta x} + \frac{2\alpha \Delta t \Delta y}{\Delta y} + \frac{2 h_{side}\Delta x \Delta y \Delta t}{\rho c_p t_K}\right]T_P^{n+1} + \left[\frac{\alpha \Delta t \Delta y}{\Delta x}\right]T_E^{n+1} + \left[\frac{\alpha \Delta t \Delta x}{\Delta y}\right]T_N^{n+1}$$

$$= -\left[\Delta x \Delta y\, T_P^{n} + \frac{2 h_{side} T_\infty \Delta x \Delta y \Delta t}{\rho c_p t_K}\right]$$

Q2| cont'd.: Since we are given the dimensions of the plate as $0.05m \times 0.05m \times 0.0035m$ with the thickness, $t_K = 0.0035m$. So that for the equispaced grid. $\Delta x = \Delta y$, thus simplifying our formulation we have,

$$\alpha \Delta t T_W^{n+1} + \alpha \Delta t T_S^{n+1} - \left[\Delta x^2 + 4\alpha\Delta t + \frac{2 h_{side} \Delta x^2 \Delta t}{\rho C_p t_K}\right] T_P^{n+1}$$
$$+ \alpha \Delta t T_E^{n+1} + \alpha \Delta t T_N^{n+1} = -\left[\Delta x^2 T_P^n + \frac{2 h_{side} T_\infty \Delta x^2 \Delta t}{\rho C_p t_K}\right]$$

This will be re-written for ease of computation as,

$$\boxed{a T_W^{n+1} + a T_S^{n+1} - a_P T_P^{n+1} + a T_E^{n+1} + a T_N^{n+1} = -b} \quad (2.1)$$

This will create a pentadiagonal matrix, which we can solve using a linear solver (e.g. Bi-CGSTAB)

>> using ghost points we can discretize the given boundary conditions as follows;

$\Rightarrow$ @ $x = 0$ we have $-\lambda \left.\frac{\partial T}{\partial x}\right|_w = h(T_\infty - T_w)$

using the generic eqn $T_w = \frac{T_P + T_W}{2}$, so that with our discretization to $\left.\frac{\partial T}{\partial x}\right|_w$, we will have

$$-\lambda\left(\frac{T_P - T_W}{\Delta x}\right) = h\left(T_\infty - \frac{T_P - T_W}{2}\right)$$

Q2 | Cont'd! In order to eliminate the ghost point we make T_W the subject as,

$$T_W = \frac{hT_\infty - \left(\frac{h}{2} - \frac{\lambda}{\Delta x}\right)T_p}{\left(\frac{\lambda}{\Delta x} + \frac{h}{2}\right)}$$

lets have this re-written for ease of computation as,

$$T_W = k_1 - k_2 T_p \quad \text{——} (2.2)$$

where $k_1 = \dfrac{hT_\infty}{\frac{\lambda}{\Delta x} + \frac{h}{2}}$ and $k_2 = \dfrac{\left(\frac{h}{2} - \frac{\lambda}{\Delta x}\right)}{\left(\frac{\lambda}{\Delta x} + \frac{h}{2}\right)}$

$\Rightarrow$ @ $x = 0.05m$, we have $\lambda \left.\frac{\partial T}{\partial x}\right|_e = h(T_\infty - T_e)$

similar to the above steps we will have,

$$T_E = \frac{hT_\infty - \left(\frac{h}{2} - \frac{\lambda}{\Delta x}\right)T_p}{\left(\frac{\lambda}{\Delta x} + \frac{h}{2}\right)}$$

$$T_E = k_1 - k_2 T_p \quad \text{——} (2.3)$$

$\Rightarrow$ @ $y = 0.05m$, we have $\lambda \left.\frac{\partial T}{\partial y}\right|_n = h(T_\infty - T_n)$

also following similar steps, we will have,

$$T_N = \frac{hT_\infty - \left(\frac{h}{2} - \frac{\lambda}{\Delta x}\right)T_p}{\left(\frac{\lambda}{\Delta y} + \frac{h}{2}\right)}$$

$$T_N = k_1 - k_2 T_p \quad \text{——} (2.4)$$

"where k terms as same as before, since $\Delta x = \Delta y$

Q2 cont'd.!

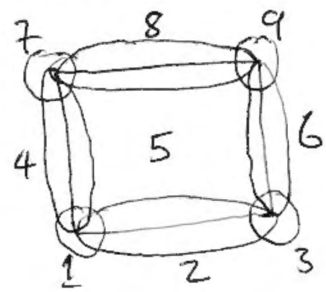
$\Rightarrow$ @ $y=0$, we have the generic eqn $T_s = \frac{T_p + T_S}{2} = 150 + 273.15$

so that $T_S = 846.3 - T_p$

lets re-write this for ease of computation as

$$T_S = k_3 - T_p \quad \text{— (2.5)}$$

Since this is a 2-D problem, we can apply this B.Cs to our formulation in 8 different boundary regions shown as,



~~Since this is a 2-D problem, we can apply this B.Cs to our formulation in~~

Since the interior region $\boxed{5}$ is governed by eqn (2.1), we can find the 8 boundary regions as follows:-

$\rightarrow$ For $\boxed{1}$ (The bottom left region); we have eqn (2.2) & (2.5) substituted into eqn (2.1), this gives

$$a(k_1 - k_2 T_p^{n+1}) + a(k_3 - T_p^{n+1}) - a_p T_p^{n+1} + a\overline{T}_E^{n+1} + aT_N^{n+1} = -b$$

re-arranging gives;

$$\boxed{-(ak_2 + a + a_p)T_p^{n+1} + aT_E^{n+1} + aT_N^{n+1} = -(b + ak_3 + ak_1)}$$

Q2| cont'd.:

→ For $\boxed{1\,2\,1}$ (The bottom region); we substitute eqn (2.5) into eqn (2.1), this gives,

$$aT_W^{n+1} + a(k_3 - T_P^{n+1}) - a_P T_P^{n+1} + aT_E^{n+1} + aT_N^{n+1} = -b$$

re-arranging gives,

$$\boxed{aT_W^{n+1} - (a + a_P)T_P^{n+1} + aT_E^{n+1} + aT_N^{n+1} = -(b + ak_3)}$$

→ For $\boxed{3}$ (The bottom right region); we substitute eqn (2.5) and (2.3) into eqn (2.1), this gives,

$$aT_W^{n+1} + a(k_3 - T_P^{n+1}) - a_P T_P^{n+1} + a(k_1 - k_2 T_P^{n+1}) + aT_N^{n+1} = -b$$

re-arranging gives,

$$\boxed{aT_W^{n+1} - (a + a_P + ak_2)T_P^{n+1} + aT_N^{n+1} = -(b + ak_1 + ak_3)}$$

→ For $\boxed{4}$ (The left region); we substitute eqn (2.2) into eqn (2.1), this gives,

$$a(k_1 - k_2 T_P^{n+1}) + aT_S^{n+1} - a_P T_P^{n+1} + aT_E^{n+1} + aT_N^{n+1} = -b$$

re-arranging gives,

$$\boxed{aT_S^{n+1} - (ak_2 + a_P)T_P^{n+1} + aT_E^{n+1} + aT_N^{n+1} = -(b + ak_1)}$$

→ for $\boxed{6}$ (The right region); we substitute eqn (2.3) into eqn (2.1), this gives,

$$aT_W^{n+1} + aT_S^{n+1} - a_P T_P^{n+1} + a(k_1 - k_2 T_P^{n+1}) + aT_N^{n+1} = -b$$

Q 2| cont'd.:

re-arranging gives,

$$\boxed{aT_W^{n+1} + aT_S^{n+1} - (a_P + ak_2)T_P^{n+1} + aT_N^{n+1} = -(b + ak_1)}$$

→ for $\boxed{7}$ (The top left region); we substitute eqn (2.2) & (2.4) into eqn (2.1), we will have,

$$a(k_1 - k_2T_P^{n+1}) + aT_S^{n+1} - a_PT_P^{n+1} + aT_E^{n+1} + a(k_1 - k_2T_P^{n+1}) = -b$$

re-arranging gives,

$$\boxed{aT_S^{n+1} - (2ak_2 + a_P)T_P^{n+1} + aT_E^{n+1} = -(b + 2ak_1)}$$

→ for $\boxed{8}$ (The top region); we substitute eqn (2.4) into eqn (2.1), this gives,

$$aT_W^{n+1} + aT_S^{n+1} - a_PT_P^{n+1} + aT_E^{n+1} + a(k_1 - k_2T_P^{n+1}) = -b$$

re-arranging gives,

$$\boxed{aT_W^{n+1} + aT_S^{n+1} - (a_P + ak_2)T_P^{n+1} + aT_E^{n+1} = -(b + ak_1)}$$

→ For $\boxed{9}$ (The top right region); we substitute eqn (2.4) & (2.3) into eqn (2.1), we will have,

$$aT_W^{n+1} + aT_S^{n+1} - a_PT_P^{n+1} + a(k_1 - k_2T_P^{n+1}) + a(k_1 - k_2T_P^{n+1}) = -b$$

re-arranging gives,

$$\boxed{aT_W^{n+1} + aT_S^{n+1} - (a_P + 2ak_2)T_P^{n+1} = -(b + 2ak_1)}$$

using the above eqns in boxes, and the given parameters we will simulate the problem in fortran90 as show below;

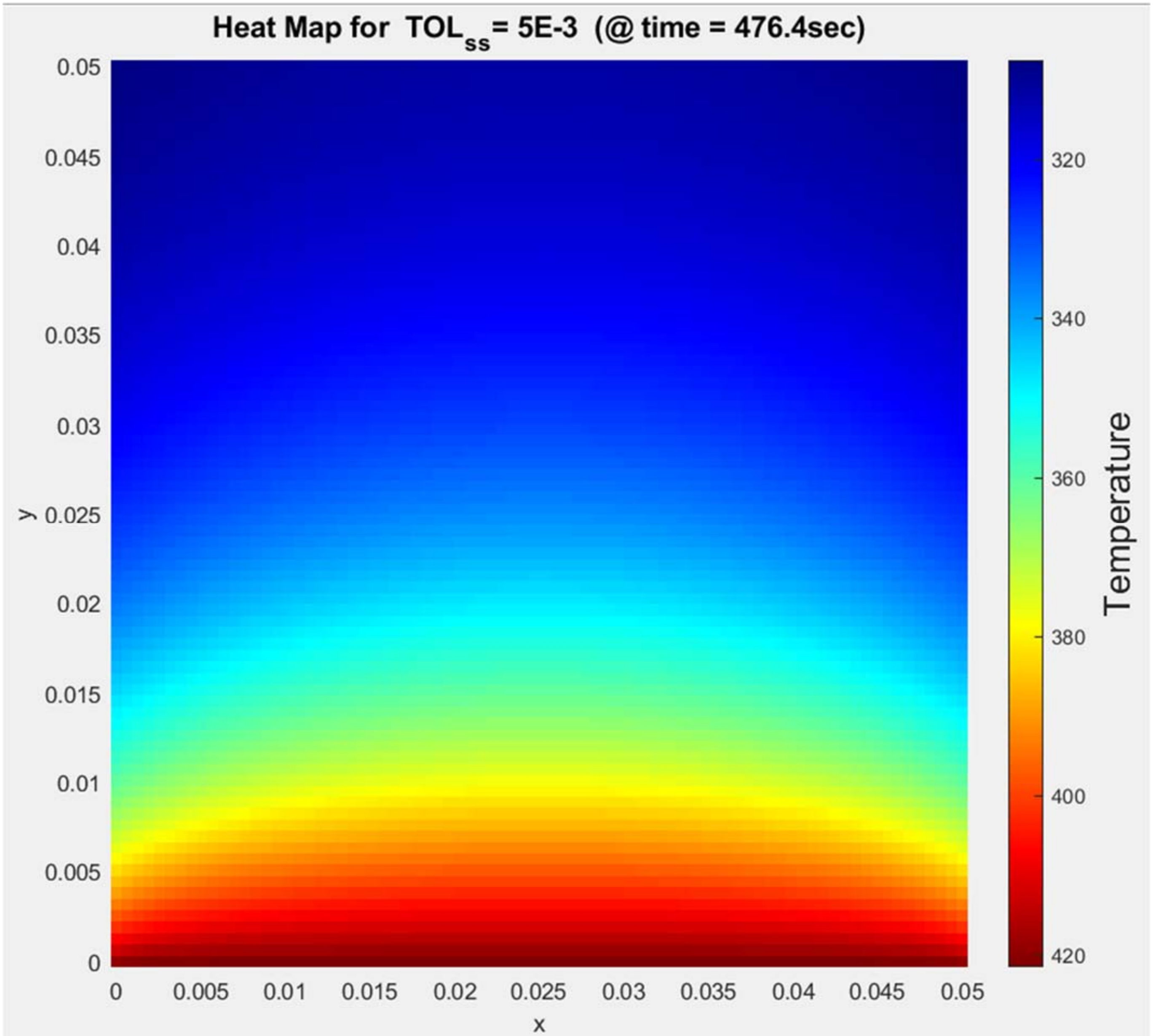
```fortran
!****************Begin Header*****************************************************
!This program was written by Godswill Ezeorah, Student Number: 501012886 on June 29, 2021.
!This program solves an unsteady linear/non-linear equation using finite volume method
!and was written as a solution to AE8112 PS3 q2
!****************************End Header*******************************************
program unsteady_Vfinite
    implicit none
    !Variable declaration
    DOUBLE PRECISION, dimension(:), ALLOCATABLE :: Ti, xii
    DOUBLE PRECISION, PARAMETER :: rh=1716, Cp=4817, lamda=14.6, ha=472, hs=36.4
    DOUBLE PRECISION, PARAMETER :: L=0.05, tk=0.0035 !plate dimension
    DOUBLE PRECISION :: dx, dt, tt, Tinf, tols
    INTEGER :: N, ii, jj, i1, j1, l1
    open(1, file = 'PS3_Q2.txt', status = 'unknown')
    N=80 !Number of Control Volume along x&y direction
    dx=L/n
    dt=0.01     !given timestep
    Tinf=298.15 !Ambient temperature
    ALLOCATE(Ti(n*n),xii(n*n))
    do ii = 1,4
        Ti=298.15 !initial temperature of the plate
        if (ii==1) then
            tols=5E-3
        elseif (ii==2) then
            tols=5E-5
        elseif (ii==3) then
            tols=5E-8
        else
            tols=5E-10
        end if
        call unsteady(Ti, tt, xii) !solves the unsteady system
        !Printing results
        write(1,2) "Tolerance = ",tols, "t_total = ",tt-dt,"sec"
        j1=n*n-(n-1)
        l1=n*n
        jj=n
        do i1 = j1, 1, -n
            write(*,1) Ti(i1:l1)
            write(1,1) xii(jj), Ti(i1:l1)
            l1=l1-n
            jj=jj-1
        end do
        write(1,4) xii(1:n)
        Print *, "t_total = ", tt-dt,"sec"
    end do
```

```fortran
    1 format(f7.3,80f8.3)
    4 format(7x,80f8.3)
    2 format(a15,E12.1,a15,E17.3,a4)
    close(1)

    contains
!*************************************************************************************
    subroutine unsteady(Tp, t, xi)
        DOUBLE PRECISION, dimension(n*n) :: d, e, f, g, h, b, Tp, xi, T_old
        DOUBLE PRECISION :: aa, ap, bb, k1, k2, k3, alpha, t, b_simp, Tnorm
        integer  :: i, j, k
    !!This subroutine solves unsteady thermal problem using fully implicit method


        !Variable initialization
        d=0; e=0; f=0; g=0; h=0
        t=0
        Tnorm=1
        alpha=lamda/(rh*Cp) !Thermal diffusivity
        aa=alpha*dt
        ap=(dx**2)+4*alpha*dt+(2*hs*dt*dx**2)/(rh*cp*tk)
        k1=ha*Tinf/((lamda/dx)+(ha/2))
        k2=((ha/2)-(lamda/dx))/((lamda/dx)+(ha/2))
        k3=846.3
        b_simp=(2*hs*dt*Tinf*dx**2)/(rh*cp*tk)
        do while (Tnorm>tols) !Loop for time step
            j=n+1
            k=n+n
            T_old=Tp
            do i = 1,n*n !Loop for control volume
                bb=Tp(i)*(dx**2)+b_simp
                !The 9 if-statements below are for the 9 regions of our CV formulation
                if ( i==1 ) then !bottom left region
                    f(i)=-(aa*k2+aa+ap)
                    g(i+1)=aa
                    h(i+n)=aa
                    b(i)=-(bb+aa*k3+aa*k1)
                    xi(i)=dx/2
                else if ( i < n ) then        !bottom region
                    e(i-1)=aa
                    f(i)=-(aa+ap)
                    g(i+1)=aa
                    h(i+n)=aa
                    b(i)=-(bb+aa*k3)
                    xi(i)=xi(i-1)+dx
```

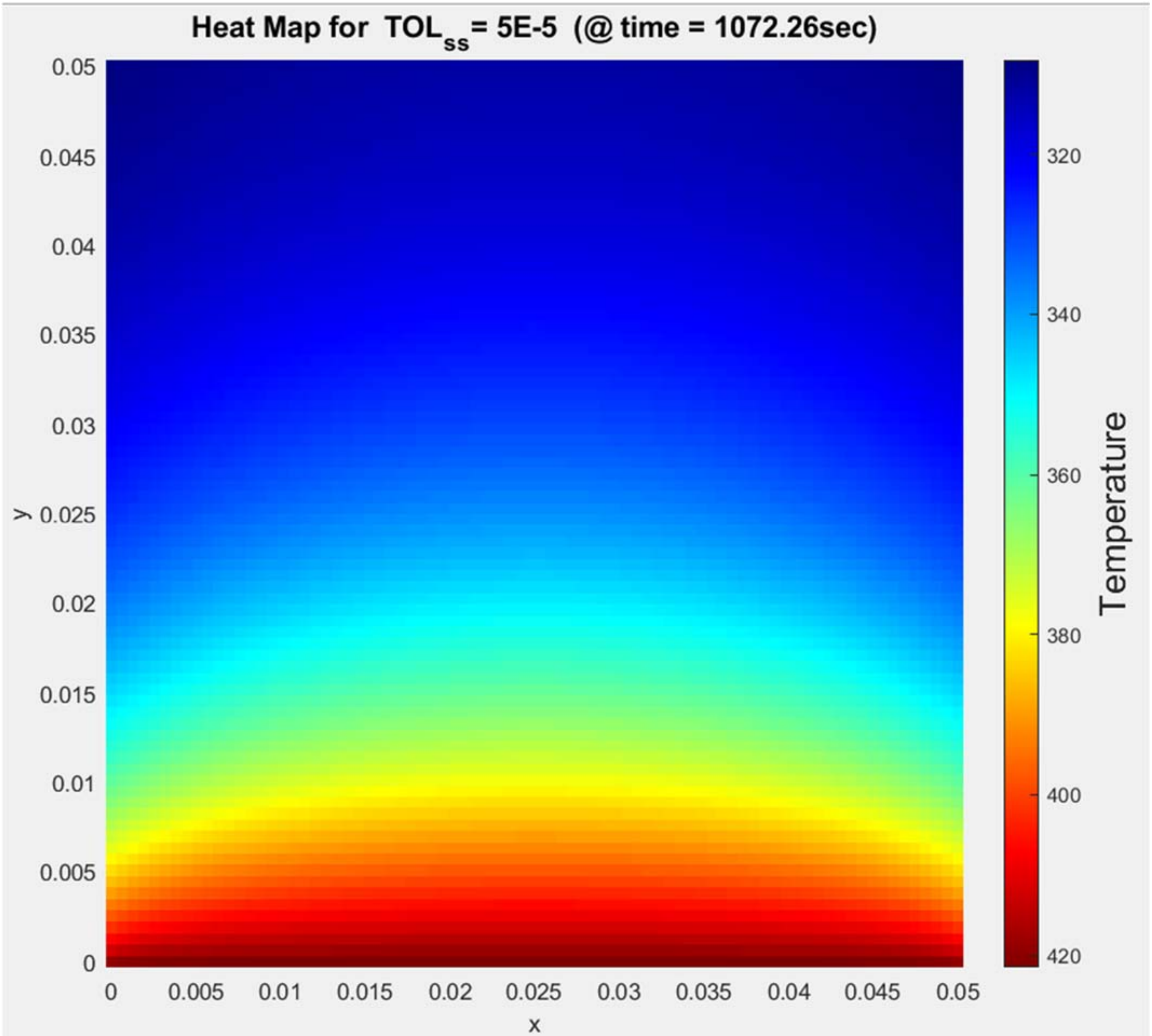
```fortran
    else if ( i==n ) then         !bottom right region
        e(i-1)=aa
        f(i)=-(aa*k2+aa+ap)
        h(i+n)=aa
        b(i)=-(bb+aa*k3+aa*k1)
        xi(i)=xi(i-1)+dx
    else if ( i==n*n-(n-1) ) then !top left region
        d(i-n)=aa
        f(i)=-(2*aa*k2+ap)
        g(i+1)=aa
        b(i)=-(bb+2*aa*k1)
    else if ( i==n*n ) then       !top right region
        d(i-n)=aa
        e(i-1)=aa
        f(i)=-(2*aa*k2+ap)
        b(i)=-(bb+2*aa*k1)
    else if ( i==j ) then         !left region
        d(i-n)=aa
        f(i)=-(aa*k2+ap)
        g(i+1)=aa
        h(i+n)=aa
        b(i)=-(bb+aa*k1)
        j=j+n
    else if ( i==k ) then         !right region
        d(i-n)=aa
        e(i-1)=aa
        f(i)=-(aa*k2+ap)
        h(i+n)=aa
        b(i)=-(bb+aa*k1)
        k=k+n
    else if ( i < n*n-n ) then    !Interior region
        d(i-n)=aa
        e(i-1)=aa
        f(i)=-ap
        g(i+1)=aa
        h(i+n)=aa
        b(i)=-bb
    else                          !top region
        d(i-n)=aa
        e(i-1)=aa
        f(i)=-(aa*k2+ap)
        g(i+1)=aa
        b(i)=-(bb+aa*k1)
    end if
end do
```

```fortran
            !For computational efficiency, the, A penta-diagonal matrix is split to 5 vectors
            call Bi_CGSTAB_P(d,e,f,g,h,b,Tp) !Solves the linear system at each time-step
            Tnorm=(sqrt(sum((T_old(1:n*n)-Tp(1:n*n))**2)))/n*n
            t=t+dt
        end do

    end subroutine unsteady
!*************************************************************************************
!*************************************************************************************
    subroutine Bi_CGSTAB_P(d,e,f,g,h,b,x)
        implicit none
        DOUBLE PRECISION, DIMENSION(n*n), INTENT(OUT) :: x
        DOUBLE PRECISION, dimension(n*n), INTENT(IN):: d, e, f, g, h, b
        DOUBLE PRECISION, DIMENSION(n*n) ::d1,e1,g1,h1,d2,e2,g2,h2
        DOUBLE PRECISION, DIMENSION(n*n) ::   p, r, r0, y, z, v, s, t, k, ks, kt
        DOUBLE PRECISION :: rho0, rho, w, alpha, beta, r_check, tol
        INTEGER :: i
    !!This subroutine solves a penta-
diagonal linear system using The Preconditioned Bi_CGSTAB Algorithm
    !!by Van Der Vorst

        !Variable initialization
        d2=0; e2=0; g2=0; h2=0
        !since we are dealing with vectors, I have done the multiplication of two vectors,
        !using this pattern
        !this multiples each vectors with x, for A*x
        d1=d(1:n*n)*x(1:n*n);e1=e(1:n*n)*x(1:n*n);g1=g(1:n*n)*x(1:n*n);h1=h(1:n*n)*x(1:n*n);
        !This circularlly shifts the vectors to the appropiate position, before addition
        d2(n+1:n*n)=d1(1:n*n-n);e2(2:n*n)=e1(1:n*n-1);g2(1:n*n-1)=g1(2:n*n);h2(1:n*n-n)=h1(n+1:n*n)
        !Hence A*x = d2+e2+f(1:n*n)*x(1:n*n)+g2+h2
        r0=b-(d2+e2+f(1:n*n)*x(1:n*n)+g2+h2)
        r=r0
        w=1; alpha=1; rho0=1; r_check=1
        v=0; p=0;  k=0
        tol=10E-9
        !For the inverse of K
        do i = 1, n*n
            k(i)=1/f(i)
        end do
        i=0
        !main algorithm loop
        do while (r_check>tol)
            i=i+1
            rho=dot_product(r0,r)
            beta=(rho/rho0)*(alpha/w)
```

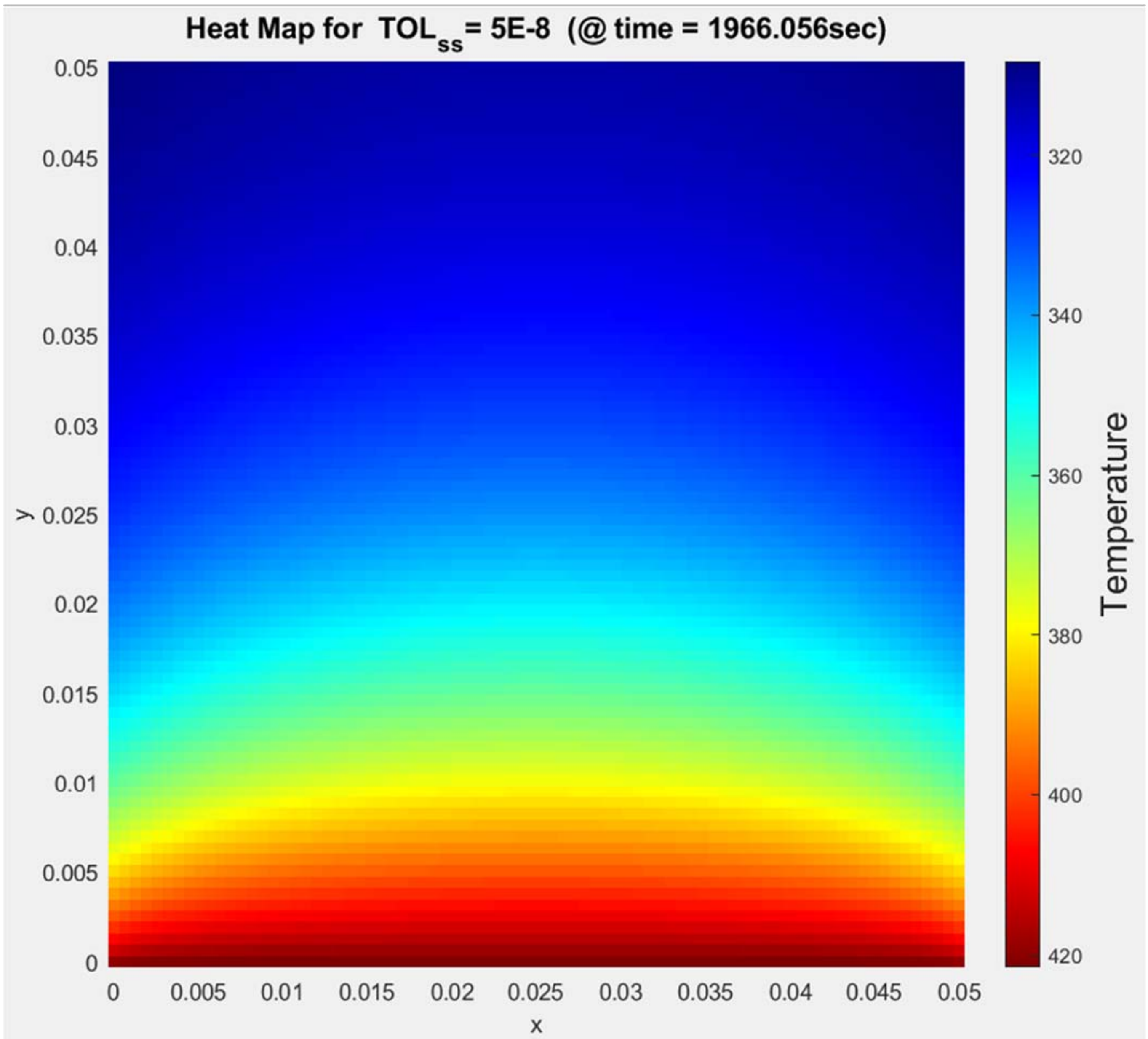
```
            rho0=dot_product(r0,r)
            p=r+beta*(p-w*v)
            y=k(1:n*n)*p(1:n*n)
            d1=d(1:n*n)*y(1:n*n);e1=e(1:n*n)*y(1:n*n);g1=g(1:n*n)*y(1:n*n);h1=h(1:n*n)*y(1:n*n);
            d2(n+1:n*n)=d1(1:n*n-n);e2(2:n*n)=e1(1:n*n-1);g2(1:n*n-1)=g1(2:n*n);h2(1:n*n-n)=h1(n+1:n*n)
            v=(d2+e2+f(1:n*n)*y(1:n*n)+g2+h2)
            alpha=rho/dot_product(r0,v)
            s=r-alpha*v
            z=k(1:n*n)*s(1:n*n)
            d1=d(1:n*n)*z(1:n*n);e1=e(1:n*n)*z(1:n*n);g1=g(1:n*n)*z(1:n*n);h1=h(1:n*n)*z(1:n*n);
            d2(n+1:n*n)=d1(1:n*n-n);e2(2:n*n)=e1(1:n*n-1);g2(1:n*n-1)=g1(2:n*n);h2(1:n*n-n)=h1(n+1:n*n)
            t=(d2+e2+f(1:n*n)*z(1:n*n)+g2+h2)
            d1=d(1:n*n)*s(1:n*n);e1=e(1:n*n)*s(1:n*n);g1=g(1:n*n)*s(1:n*n);h1=h(1:n*n)*s(1:n*n);
            d2(n+1:n*n)=d1(1:n*n-n);e2(2:n*n)=e1(1:n*n-1);g2(1:n*n-1)=g1(2:n*n);h2(1:n*n-n)=h1(n+1:n*n)
            ks=(d2+e2+f(1:n*n)*s(1:n*n)+g2+h2)
            d1=d(1:n*n)*t(1:n*n);e1=e(1:n*n)*t(1:n*n);g1=g(1:n*n)*t(1:n*n);h1=h(1:n*n)*t(1:n*n);
            d2(n+1:n*n)=d1(1:n*n-n);e2(2:n*n)=e1(1:n*n-1);g2(1:n*n-1)=g1(2:n*n);h2(1:n*n-n)=h1(n+1:n*n)
            kt=(d2+e2+f(1:n*n)*t(1:n*n)+g2+h2)
            w=dot_product(kt,ks)/dot_product(kt,kt)
            x=x+alpha*y+w*z
            r=s-w*t
            r_check=norm2(r)
        end do
    end subroutine Bi_CGSTAB_P
!*************************************************************************************
end program unsteady_Vfinite
```



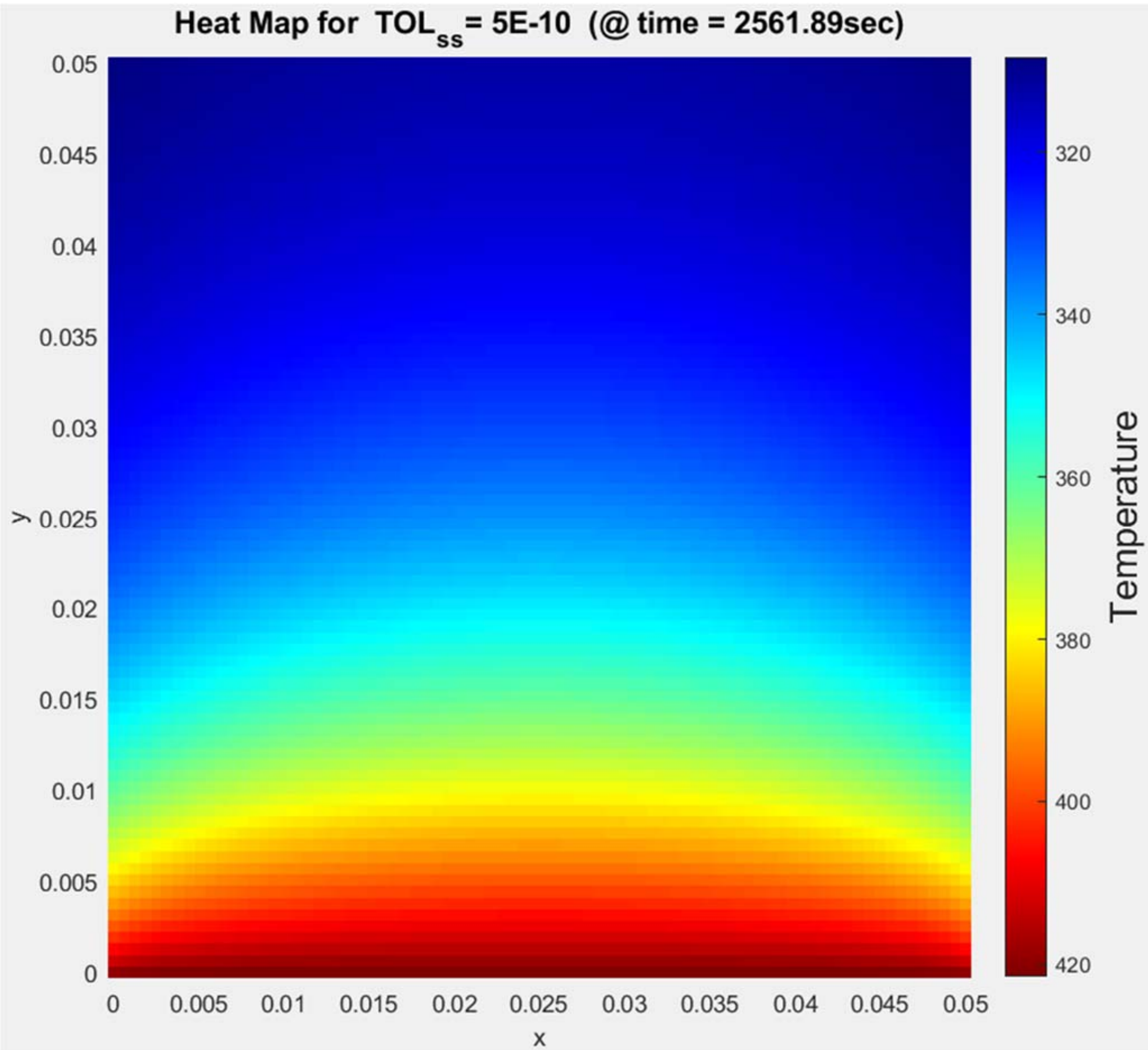
Heat Map for TOL$_{ss}$= 5E-3 (@ time = 476.4sec)
y
x
Temperature
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
320
340
360
380
400
420



Heat Map for TOL_{ss}= 5E-5 (@ time = 1072.26sec)
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
y
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
x
320
340
360
380
400
420
Temperature



Heat Map for TOL$_{ss}$= 5E-8 (@ time = 1966.056sec)
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
y
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
x
320
340
360
380
400
420
Temperature



Heat Map for TOL$_{ss}$= 5E-10 (@ time = 2561.89sec)
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
y
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
x
320
340
360
380
400
420
Temperature

Q2| Cont'd.|

>> From the above plots we can see that as the TOL_{ss} reduces, the time taken to reach steady state increases.

>> As observed from the temperature data with the plate, the evolution of the temperature will be; ~~mean m~~ that as the dwell time increasing, there will be more heat diffusion upward from the source $(y=0)$. N/B: This is not very visible in the above plots but it can be seen from the temperature data.

For example, if N=10x10, here is a sample of what the temperature data will look like:

```
Tolerance = 0.5E-02
304.111 304.839 305.390 305.758 305.942 305.942 305.758 305.390 304.839 304.111
305.926 306.872 307.583 308.056 308.291 308.291 308.056 307.583 306.872 305.926
308.859 310.151 311.113 311.746 312.059 312.059 311.746 311.113 310.151 308.859
313.172 314.961 316.274 317.126 317.542 317.542 317.126 316.274 314.961 313.172
319.209 321.669 323.434 324.557 325.098 325.098 324.557 323.434 321.669 319.209
327.427 330.745 333.044 334.464 335.135 335.135 334.464 333.044 330.745 327.427
338.459 342.797 345.644 347.324 348.096 348.096 347.324 345.644 342.797 338.459
353.245 358.637 361.869 363.651 364.437 364.437 363.651 361.869 358.637 353.245
373.335 379.347 382.417 383.941 384.575 384.575 383.941 382.417 379.347 373.335
401.651 406.245 407.901 408.582 408.843 408.843 408.582 407.901 406.245 401.651
t_total = 208.69999533519149 sec

Tolerance = 0.5E-04
309.691 311.114 312.202 312.934 313.303 313.303 312.934 312.202 311.114 309.691
312.139 313.857 315.166 316.045 316.486 316.486 316.045 315.166 313.857 312.139
315.465 317.578 319.175 320.240 320.772 320.772 320.240 319.175 317.578 315.465
319.891 322.516 324.475 325.766 326.405 326.405 325.766 324.475 322.516 319.891
325.725 328.996 331.388 332.937 333.694 333.694 332.937 331.388 328.996 325.725
333.402 337.463 340.337 342.147 343.016 343.016 342.147 340.337 337.463 333.402
343.552 348.524 351.860 353.873 354.814 354.814 353.873 351.860 348.524 343.552
357.142 363.019 366.626 368.663 369.577 369.577 368.663 366.626 363.019 357.142
375.784 382.100 385.407 387.090 387.805 387.805 387.090 385.407 382.100 375.784
402.487 407.185 408.920 409.656 409.945 409.945 409.656 408.920 407.185 402.487
t_total = 805.36998199857771 sec

Tolerance = 0.5E-07
309.752 311.182 312.275 313.012 313.382 313.382 313.012 312.275 311.182 309.752
312.205 313.932 315.247 316.130 316.573 316.573 316.130 315.247 313.932 312.205
315.534 317.656 319.260 320.329 320.863 320.863 320.329 319.260 317.656 315.534
319.960 322.593 324.559 325.855 326.496 326.496 325.855 324.559 322.593 319.960
325.790 329.070 331.467 333.020 333.780 333.780 333.020 331.467 329.070 325.790
333.460 337.528 340.408 342.222 343.093 343.093 342.222 340.408 337.528 333.460
343.600 348.578 351.919 353.935 354.878 354.878 353.935 351.919 348.578 343.600
357.179 363.060 366.671 368.709 369.625 369.625 368.709 366.671 363.060 357.179
375.806 382.126 385.434 387.119 387.835 387.835 387.119 385.434 382.126 375.806
402.494 407.193 408.930 409.666 409.955 409.955 409.666 408.930 407.193 402.494
t_total = 1702.0199619568884 sec
```

Tolerance = 0.5E-09

309.752	311.182	312.275	313.012	313.382	313.382	313.012	312.275	311.182	309.752
312.205	313.932	315.247	316.130	316.573	316.573	316.130	315.247	313.932	312.205
315.534	317.656	319.260	320.330	320.863	320.863	320.330	319.260	317.656	315.534
319.960	322.593	324.559	325.855	326.496	326.496	325.855	324.559	322.593	319.960
325.790	329.070	331.468	333.021	333.780	333.780	333.021	331.468	329.070	325.790
333.460	337.528	340.408	342.222	343.093	343.093	342.222	340.408	337.528	333.460
343.600	348.578	351.919	353.935	354.878	354.878	353.935	351.919	348.578	343.600
357.179	363.060	366.671	368.709	369.625	369.625	368.709	366.671	363.060	357.179
375.806	382.126	385.434	387.119	387.835	387.835	387.119	385.434	382.126	375.806
402.494	407.193	408.930	409.666	409.955	409.955	409.666	408.930	407.193	402.494

```
t_total = 2299.7999485954642 sec
```