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

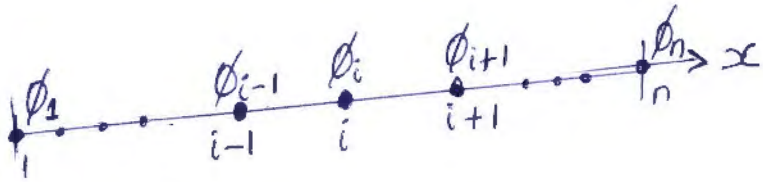
Problem Set 1

Submission Date	May 26, 2021
Programing Language Used	Fortran90

Student Name	Student Number
Ezeorah Godswill	501012886

1.a] Lets consider a finite difference discretized n-number of points as shown below, with a dependent variable ϕ,

In a 1-D case



let $\phi(x_i) = \phi_i$

in order to solve for $\left.\frac{\partial \phi}{\partial x}\right|_i$ and $\left.\frac{\partial^2 \phi}{\partial x^2}\right|_i$ we will use Taylor's series expansion to approximate their values,

T.S. for ϕ_{i+1}, centered at ϕ_i

$$\phi_{i+1} = \phi_i + \left.\frac{\partial \phi}{\partial x}\right|_i (x_{i+1}-x_i) + \left.\frac{\partial^2 \phi}{\partial x^2}\right|_i \frac{(x_{i+1}-x_i)^2}{2!} + \left.\frac{\partial^3 \phi}{\partial x^3}\right|_i \frac{(x_{i+1}-x_i)^3}{3!}$$

Lets put $\Delta x_+ = x_{i+1} - x_i$, The above eqn becomes

$$\Rightarrow \phi_{i+1} = \phi_i + \left.\frac{\partial \phi}{\partial x}\right|_i \Delta x_+ + \left.\frac{\partial^2 \phi}{\partial x^2}\right|_i \frac{\Delta x_+^2}{2} + \left.\frac{\partial^3 \phi}{\partial x^3}\right|_i \frac{\Delta x_+^3}{6} \quad \text{——— (1.1)}$$

T.S. For ϕ_{i-1}, centered at ϕ_i

$$\phi_{i-1} = \phi_i + \left.\frac{\partial \phi}{\partial x}\right|_i (x_{i-1}-x_i) + \left.\frac{\partial^2 \phi}{\partial x^2}\right|_i \frac{(x_{i-1}-x_i)^2}{2!} + \left.\frac{\partial^3 \phi}{\partial x^3}\right|_i \frac{(x_{i-1}-x_i)^3}{3!}$$

This can also be re-written and let $\Delta x_- = x_i - x_{i-1}$, we get,

$$\Rightarrow \phi_{i-1} = \phi_i - \left.\frac{\partial \phi}{\partial x}\right|_i \Delta x_- + \left.\frac{\partial^2 \phi}{\partial x^2}\right|_i \frac{\Delta x_-^2}{2} - \left.\frac{\partial^3 \phi}{\partial x^3}\right|_i \frac{\Delta x_-^3}{6} \quad \text{——— (1.2)}$$

we can also say $\frac{\text{eqn (1.1)}}{\Delta x_+}$ and $\frac{\text{eqn (1.2)}}{\Delta x_-}$, these will yield the following eqns.

continued Q1.a]

$$\gg \frac{(\phi_{i+1}-\phi_i)}{\Delta x_+} = \frac{\partial \phi}{\partial x}\bigg|_i + \frac{\partial^2 \phi}{\partial x^2}\bigg|_i \frac{\Delta x_+}{2} + \frac{\partial^3 \phi}{\partial x^3}\bigg|_i \frac{\Delta x_+^2}{6} \quad \text{---} (1.3)$$

$$\gg \frac{(\phi_{i-1}-\phi_i)}{\Delta x_-} = -\frac{\partial \phi}{\partial x}\bigg|_i + \frac{\partial^2 \phi}{\partial x^2}\bigg|_i \frac{\Delta x_-}{2} - \frac{\partial^3 \phi}{\partial x^3}\bigg|_i \frac{\Delta x_-^2}{6} \quad \text{----} (1.4)$$

We can get our $\frac{\partial \phi}{\partial x}\Big|_i$ eqn by solving eqn (1.1) − eqn (1.2)

$$\gg \frac{\partial \phi}{\partial x}\bigg|_i = \frac{(\phi_{i+1}-\phi_{i-1})}{(\Delta x_+ + \Delta x_-)} - \frac{\partial^2 \phi}{\partial x^2}\bigg|_i \frac{(\Delta x_+^2 - \Delta x_-^2)}{2(\Delta x_+ + \Delta x_-)} - \frac{\partial^3 \phi}{\partial x^3}\bigg|_i \frac{(\Delta x_+^3 - \Delta x_-^3)}{6(\Delta x_+ + \Delta x_-)} \quad \text{--} (1.5)$$

In order to have the ϕ_i term in the above eqn
Lets consider eqn (1.3) − eqn (1.4), This gives,

$$\gg \frac{\partial \phi}{\partial x}\bigg|_i = \frac{(\phi_{i+1}-\phi_i)}{2\Delta x_+} - \frac{(\phi_{i-1}-\phi_i)}{2\Delta x_-} - \frac{\partial^2 \phi}{\partial x^2}\bigg|_i \frac{(\Delta x_+ - \Delta x_-)}{4} - \frac{\partial^3 \phi}{\partial x^3}\bigg|_i \frac{(\Delta x_+^2 + \Delta x_-^2)}{12} \quad \text{----} (1.6)$$

If we equate eqn (1.5) and eqn (1.6),
and solve for $(\phi_{i+1}-\phi_{i-1})$ in the following steps,

$$\frac{(\phi_{i+1}-\phi_{i-1})}{(\Delta x_+ + \Delta x_-)} = \frac{(\phi_{i+1}-\phi_i)}{2\Delta x_+} - \frac{(\phi_{i-1}-\phi_i)}{2\Delta x_-} - \frac{\partial^2 \phi}{\partial x^2}\bigg|_i \left[\frac{(\Delta x_+ - \Delta x_-)}{4} - \frac{(\Delta x_+^2 - \Delta x_-^2)}{2(\Delta x_+ + \Delta x_-)}\right]$$
$$- \frac{\partial^3 \phi}{\partial x^3}\bigg|_i \left[\frac{(\Delta x_+^2 + \Delta x_-^2)}{12} - \frac{(\Delta x_+^3 + \Delta x_-^3)}{6(\Delta x_+ + \Delta x_-)}\right]$$

multiplying by 2 and $(\Delta x_+ + \Delta x_-)$ and simplyfying, we get,

$$(\phi_{i+1}-\phi_{i-1}) = (\phi_{i+1}-\phi_i)\left(\frac{\Delta x_-}{\Delta x_+}\right) - (\phi_{i-1}-\phi_i)\left(\frac{\Delta x_+}{\Delta x_-}\right) - \frac{\partial^2 \phi}{\partial x^2}\bigg|_i \left[\frac{(\Delta x_+ - \Delta x_-)(\Delta x_+ + \Delta x_-)}{2}\right.$$
$$\left. - \frac{(\Delta x_+^2 - \Delta x_-^2)}{1}\right] - \frac{\partial^3 \phi}{\partial x^3}\bigg|_i \left[\frac{(\Delta x_+^2 + \Delta x_-^2)(\Delta x_+ + \Delta x_-)}{6} - \frac{(\Delta x_+^3 + \Delta x_-^3)}{3}\right]$$

Substituting the above into eqn (1.5), we get,

$$\frac{\partial \phi}{\partial x}\bigg|_i = \frac{(\phi_{i+1}-\phi_i)\frac{\Delta x_-}{\Delta x_+} - (\phi_{i-1}-\phi_i)\frac{\Delta x_+}{\Delta x_-}}{(\Delta x_+ + \Delta x_-)} + \frac{\partial^2 \phi}{\partial x^2}\bigg|_i \left[-\frac{(\Delta x_+ - \Delta x_-)}{2} - \frac{(\Delta x_+^2 - \Delta x_-^2)}{2(\Delta x_+ + \Delta x_+}\right.$$
$$\left. + \frac{(\Delta x_+^2 - \Delta x_-^2)}{(\Delta x_+ + \Delta x_-)}\right] + \frac{\partial^3 \phi}{\partial x^3}\bigg|_i \left[\frac{(\Delta x_+^3 + \Delta x_-^3)}{3(\Delta x_+ + \Delta x_-)} - \frac{(\Delta x_+^2 + \Delta x_-^2)}{6} - \frac{(\Delta x_+^3 + \Delta x_-^3)}{6(\Delta x_+ + \Delta x_-)}\right]$$

Continued Q1.a]

Lets truncate ~~the~~ ~~fr~~om the second order term so that the residual (which contains these truncated terms) is,

$$R_i = \frac{\partial^2\phi}{\partial x^2}\bigg|_i \frac{2\Delta x_+^2 - 2\Delta x_-^2 - \Delta x_+^2 + \Delta x_-^2 - \Delta x_+^2 + \Delta x_-^2}{2(\Delta x_+ + \Delta x_-)} \to 0$$

$$+ \frac{\partial^3\phi}{\partial x^3}\bigg|_i \left[\frac{2\Delta x_+^3 + 2\Delta x_-^3 - \Delta x_+^3 - \Delta x_+^2\Delta x_- - \Delta x_-^2\Delta x_+ - \Delta x_-^3}{6(\Delta x_+ + \Delta x_-)} \right.$$
$$\left. \frac{-\Delta x_+^3 - \Delta x_-^3}{6(\Delta x_+ + \Delta x_-)}\right)$$

Simplifying, we will get,

$$R_i = \frac{\partial^3\phi}{\partial x^3}\bigg|_i \frac{\Delta x_-^2\Delta x_+ - \Delta x_+^2\Delta x_-}{6(\Delta x_+ + \Delta x_-)}$$

Therefore the eqn becomes,

$$\therefore \boxed{\frac{\partial\phi}{\partial x}\bigg|_i \approx \frac{(\phi_{i+1} - \phi_i)\frac{\Delta x_-}{\Delta x_+} - (\phi_{i-1} - \phi_i)\frac{\Delta x_+}{\Delta x_-}}{(\Delta x_- + \Delta x_+)} + R_i}$$

And the Largest ~~truncted~~ truncated term in R_i is

$$\cancel{R_i =} \boxed{\frac{\partial^3\phi}{\partial x^3}\bigg|_i \frac{\Delta x_-^3\Delta x_+ - \Delta x_+^2\Delta x_-}{6(\Delta x_+ + \Delta x_-)}}$$

1.b) if $\Delta x_+ = \Delta x_-$

we will get, $R_i = 0$ and

$$\therefore \boxed{\frac{\partial\phi}{\partial x}\bigg|_i = \frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x}}$$

where $\Delta x = \Delta x_+ = \Delta x_-$

This means that the points are equispaced along the x-axis.

1.c] Similarly to estimate for the second order term $\left.\frac{\partial^2 \phi}{\partial x^2}\right|_i$

Consider eqn (1.3) + eqn (1.4), we will get.

$$\left.\frac{\partial^2 \phi}{\partial x^2}\right|_i = \frac{2(\phi_{i+1}-\phi_i)\frac{1}{\Delta x_+} + 2(\phi_{i-1}-\phi_i)\frac{1}{\Delta x_-}}{\Delta x_+ + \Delta x_-} - \left.\frac{\partial^3 \phi}{\partial x^3}\right|_i \frac{2(\Delta x_+^2 - \Delta x_-^2)}{6(\Delta x_+ + \Delta x_-)}$$

we can have the residual term,

$$R_i = \left.\frac{\partial^3 \phi}{\partial x^3}\right|_i \frac{(\Delta x_-^2 - \Delta x_+^2)}{3(\Delta x_+ + \Delta x_-)} = \left.\frac{\partial^3 \phi}{\partial x^3}\right|_i \frac{(\Delta x_- - \Delta x_+)\cancel{(\Delta x_- + \Delta x_+)}}{3\cancel{(\Delta x_+ + \Delta x_-)}}$$

$$R_i = \left.\frac{\partial^3 \phi}{\partial x^3}\right|_i \frac{(\Delta x_- - \Delta x_+)}{3}$$

So that the eqn becomes,

$$\therefore \boxed{\left.\frac{\partial^2 \phi}{\partial x^2}\right|_i \approx \frac{(\phi_{i-1}-\phi_i)\frac{2}{\Delta x_-} + (\phi_{i+1}-\phi_i)\frac{2}{\Delta x_+}}{(\Delta x_- + \Delta x_+)} + R_i}$$

And the Largest truncated term in R_i is

$$\boxed{\left.\frac{\partial^3 \phi}{\partial x^3}\right|_i \frac{(\Delta x_- - \Delta x_+)}{3}}$$

1.d] If $\Delta x_+ = \Delta x_-$

we will have, $R_i = 0$ and

$$\therefore \boxed{\left.\frac{\partial^2 \phi}{\partial x^2}\right|_i \approx \frac{\phi_{i-1} - 2\phi_i + \phi_{i+1}}{\Delta x^2}}$$

where $\Delta x = \Delta x_+ = \Delta x_-$

This means that the discretized points are at equal distance from one another, along the x-axis.

2.a] Given $\frac{\partial^2 T}{\partial x^2} + \cos(x) = 0$

We can solve this analytically using ODE

that, $T(x) = C.F. + P.I$ ——— (2.1)

where, C.F. is the Complementary Fn and P.I is the Particular Integral

If we re-arrange our eqn, we get

$$\frac{\partial^2 T}{\partial x^2} = -\cos(x)$$

Let $D = \frac{\partial}{\partial x}$, so that we have

$$(D^2)T = -\cos(x)$$

we can have an auxillary eqn from the above

as, $m^2 = 0$

And the roots for this aux. eqn are real and equal, i.e. $m = m_1 = m_2 = 0$

Hence, C.Omplementary fn for this type is

$$C.F. = e^{mx}(c_1 + c_2 x)$$, with $m=0$, we get

$$C.F. = C_1 + C_2 x$$

where C_1 & C_2 are some arbitrary constants.

To find the particular Integral, we have that our give eqn is of the Type II form (with a triagnometric fn in the RHS) continued Q2.a

So that $P.I = \frac{1}{f(D)} \cdot [-\cos(x)]$

for $D^2 = -a^2$ with a as the coeff. of x in ~~cos x~~ $\cos(x)$

that is $a = 1$

So that $D^2 = -1$

$$P.I = \frac{1}{D^2} \cdot [-\cos(x)]$$

$$P.I = \frac{1}{-1} \cdot [-\cos(x)]$$

$$P.I = \cos(x)$$

Substituting the values of C.F & P.I into eqn 2.1

we get, $T(x) = C_1 + C_2 x + \cos(x)$

In order to solve for the constants, lets apply the B.C.s

$$T(0) = C_1 + C_2(0) + \cos(0) = 1$$

$$C_1 = 1 - 1 = 0$$

$$C_1 = 0$$

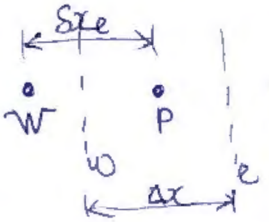
$$T(2\pi) = 0 + 2\pi C_2 = 1 - \cos(2\pi)$$

$$C_2 = 0$$

$\Rightarrow$ Therefore our analytical equation becomes

$$\boxed{T(x) = \cos(x)}$$

Q2.b] Solving $T(x)$ for an equispace grid, using Finite volume, as,



Integrating our eqn over this control volume, yields

$$\left.\frac{\partial T}{\partial x}\right|_e - \left.\frac{\partial T}{\partial x}\right|_w + \int_w^e \cos(x)\,dx = 0$$

If we consider a temperature profile for the first order derivatives, we will get

$$\left.\frac{\partial T}{\partial x}\right|_e = \frac{T_E - T_P}{\delta x_e} \quad , \quad \left.\frac{\partial T}{\partial x}\right|_w = \frac{T_P - T_W}{\delta x_w}$$

So that our eqn becomes

$$\frac{T_E - T_P}{\delta x_e} - \frac{T_P - T_W}{\delta x_w} + \left[\sin(x_e) - \sin(x_w)\right] = 0$$

This can also be written as

$$\boxed{a_W T_W - a_P T_P + a_E T_E = -b} \quad (2.2)$$

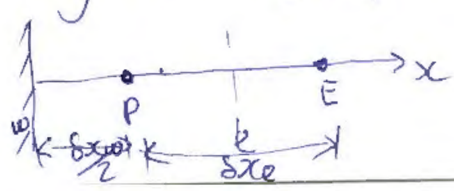
where $a = a_W = a_E = \frac{1}{\delta x}$

$b = \sin(x_e) - \sin(x_w)$

$a_P = a_W + a_E$

Considering the B.C, $T(0) = 1$

>>



continued Q2.b)

so that $\left.\frac{\partial T}{\partial x}\right|_w = \frac{T_p - T_{w.bc}}{\frac{\delta x_w}{2}} = \frac{T_p - T(0)}{\frac{\delta x_w}{2}} = \frac{2(T_p - 1)}{\delta x_w}$

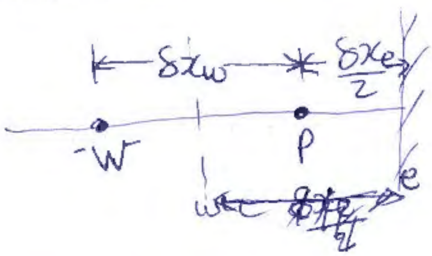
and our discretized eqn becomes

$$\frac{T_E - T_P}{\delta x_e} - \frac{2(T_P - 1)}{\delta x_w} + \sin(x_e) = 0$$

also written as,

$$\boxed{-(a_E + 2a_W)T_P + a_E T_E = -2a_W - \sin(x_e)} \quad (2.3)$$

>> Lets consider the last control volume



so that $\left.\frac{\partial T}{\partial x}\right|_e = \frac{T_{E,bc} - T_P}{\frac{\delta x_e}{2}} = \frac{T(2\pi) - T_P}{\frac{\delta x_e}{2}} = \frac{2(1 - T_P)}{\delta x_e}$

Hence our discretized eqn becomes

$$\frac{2(1 - T_P)}{\delta x_e} - \frac{T_P - T_W}{\delta x_w} - \sin(x_w) = 0$$

also written as

$$\boxed{a_W T_W - (2a_E + a_W)T_P = -2a_E + \sin(x_w)} \quad (2.4)$$

eqn (2.2), (2.3) & (2.4) will be used in the Fortran90 code to compute our problems, as ~~descri~~ shown below;

```fortran
!***************Begin Header***************************************************
!This program was written by Godswill Ezeorah, Student Number: 501012886 on May 20, 2021.
!This program solves a linear/non-linear equation using finite volume method
!and was written as a solution to AE8112 PS1 q2b
!***************************End Header*****************************************
program finit_vol
    !Variable declaration
    implicit none
    real, dimension(:,:), ALLOCATABLE:: Ag
    real, DIMENSION(:), ALLOCATABLE:: bg, Ti, Te
    real, PARAMETER :: Pi = 4*atan(1.0) !pi parameter definition
    real  :: L1, L2, Linf, dx
    integer :: N
    N=8
    open(1, file = 'vol_error.csv', status = 'unknown')
do while(n<=64)  !This will run for N=8,16,32 and 64
   call creat_eTDMA(Ag,bg,Ti,Te,dx)
   !Calculating the error terms
    L1=sum(abs(Ti(1:n)-Te(1:n)))
    L2=sum((Ti(1:n)-Te(1:n))**2)
    Linf=maxval(abs(Ti(1:n)-Te(1:n)))
    !outputing the results
    print 3, 'For number of grid points, n =',n
    write(*,1)  "T(x) = ",Ti
    write(*,2) "L1 = ", L1
    write(*,2) "L2 = ", L2
    write(*,2) "L∞ = ", Linf
    1 format(a6,64f8.3)
    2 format(a7,8f8.5)
    3 format(a30,i3)
    !writing one of the errors (L2) as a function of
    !grid spacing (dx) to excel file for ploting
    write(1,*) dx, L2
    N=n+n
end do
close(1)

contains
!******************************************************************************
subroutine creat_eTDMA(Ag1,bg1,Ti1,Te1,dx1)
    implicit none
    real, dimension(:,:), ALLOCATABLE, INTENT(OUT):: Ag1
    real, DIMENSION(:), ALLOCATABLE, INTENT(OUT):: bg1, Ti1, Te1
    real, INTENT(OUT) :: dx1
    real  :: aa, ap, xi
    integer :: i
    !This subroutine generates the tri-diagonal matric (TDMA) using our derived equations

    ALLOCATE(Ag1(n,n),bg1(n),Ti1(n),Te1(n)) !array allocation
    !Initializing Variables
    Ag1=0
    xi=0
```

```
        dx1=2*pi/n
        aa=1/(dx1)
        ap=aa+aa
    do i = 1, n  !This loop Matrix composition of the given problem
        xi=xi+dx1
        !for the the first gridpoint, applying B.Cs T(0)=1
        if ( i==1 ) then
            Ag1(i,i)=-aa-2*aa
            Ag1(i,i+1)=aa
            bg1(i)=-2*aa-sin(xi)
            Te1(i)=cos(dx1/2) !This is T_exact from our analytical solution
        !for the the intermediate gridpoint
        else if ( i<n ) then
            Ag1(i,i-1)=aa
            Ag1(i,i)=-ap
            Ag1(i,i+1)=aa
            bg1(i)=sin(xi-dx1)-sin(xi)
            Te1(i)=cos(xi-dx1/2)
        !for the the last gridpoint, applying B.Cs T(2pi)=1
        else
            Ag1(n,n-1)=aa
            Ag1(n,n)=-2*aa-aa
            bg1(n)=-2*aa+sin(xi-dx1)
            Te1(n)=cos(xi-dx1/2)
        end if
    end do
    call tdma(Ag1,bg1,Ti1)
end subroutine creat_eTDMA
!*********************************************************************************

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

    b=b1
   !extracting e, f and g array
    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 =  8
T(x) =   1.000   0.445  -0.341  -0.896  -0.896  -0.341   0.445   1.000
  L1 =  0.41552
  L2 =  0.02432
L∞ =  0.07612
For number of grid points, n = 16
T(x) =   1.000   0.850   0.572   0.209  -0.183  -0.546  -0.824  -0.974  -0.974
-0.824  -0.546  -0.183   0.209   0.572   0.850   1.000
  L1 =  0.20615
  L2 =  0.00299
L∞ =  0.01921
For number of grid points, n = 32
T(x) =   1.000   0.962   0.887   0.777   0.639   0.475   0.294   0.101  -0.095
-0.288  -0.469  -0.632  -0.771  -0.880  -0.955  -0.994  -0.994  -0.955  -0.880
-0.771  -0.632  -0.469  -0.288  -0.095   0.101   0.294   0.475   0.639   0.777
0.887   0.962   1.000
  L1 =  0.10288
  L2 =  0.00037
L∞ =  0.00482
For number of grid points, n = 64
T(x) =   1.000   0.990   0.971   0.943   0.905   0.859   0.804   0.742   0.673
0.597   0.515   0.429   0.338   0.244   0.148   0.050  -0.048  -0.146  -0.242
-0.336  -0.427  -0.514  -0.595  -0.671  -0.740  -0.803  -0.857  -0.904  -0.941
-0.970  -0.989  -0.998  -0.998  -0.989  -0.970  -0.941  -0.904  -0.857  -0.803
-0.740  -0.671  -0.595  -0.514  -0.427  -0.336  -0.242  -0.146  -0.048   0.050
0.148   0.244   0.338   0.429   0.515   0.597   0.673   0.742   0.804   0.859
0.905   0.943   0.971   0.990   1.000
  L1 =  0.05142
  L2 =  0.00005
L∞ =  0.00120
```

Q 2.c] To Solve $T(x)$ using Finite difference method for equispace grid, as

we have our eqn $\frac{\partial^2 T}{\partial x^2} + \cos(x) = 0$

we can the second order derivative using Tylor's series, as done in Q1.d, so that we will just substitute the dependent variable $\phi = T$, and we get

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_i = \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2}$$

So that for grid point W, P & E we have

$$\frac{T_W - 2T_P + T_E}{\Delta x^2} + \cos(x_P) = 0$$

This can also be written as

$$\boxed{a_W T_W - a_P T_P + a_E T_E = -b} \qquad (2.5)$$

where $a = a_W = a_E = \frac{1}{\Delta x^2}$ and $b = \cos(x_P)$

$$a_P = a_W + a_E$$

>> And at the Boundary we have

$$\boxed{\begin{aligned} T(0) &= T_1 = 1 \\ T(2\pi) &= T_N = 1 \end{aligned}}$$

We will also use eqn (2.5) in the Fortran 90 program, as shown below;

```
!***************Begin Header***************************************************
!This program was written by Godswill Ezeorah, Student Number: 501012886 on May 20, 2021.
!This program solves a linear/non-linear equation using finite difference method
!and was written as a solution to AE8112 PS1 q2c
!***************************End Header*****************************************
program finit_diff
    !Variable declaration
    implicit none
    real, dimension(:,:), ALLOCATABLE:: Ag
    real, DIMENSION(:), ALLOCATABLE:: bg, Ti, Te
    real, PARAMETER :: Pi = 4*atan(1.0) !pi parameter definition
    real  :: L1, L2, Linf, dx
    integer :: N
    N=9

    open(1, file = 'diff_error.csv', status = 'unknown')
do while(n<=65)  !This will run for N=9,17,33 and 65
    call creat_eTDMA(Ag,bg,Ti,Te,dx)

   !Calculating the error terms
    L1=sum(abs(Ti(1:n)-Te(1:n)))
    L2=sum((Ti(1:n)-Te(1:n))**2)
    Linf=maxval(abs(Ti(1:n)-Te(1:n)))
    !outputing the results
    print 3, 'For number of grid points, n =',n
    write(*,1)  "T(x) = ",Ti
    write(*,2) "L1 = ",L1
    write(*,2) "L2 = ",L2
    write(*,2) "L∞ = ",Linf
    1 format(a6,65f9.3)
    2 format(a7,9f9.5)
    3 format(a30,i3)
    !writing one of the errors (L2) as a function of
    !grid spacing (dx) to excel file for ploting
    write(1,*) dx, L2
    N=n+n-1

end do
close(1)

contains
!******************************************************************************
subroutine creat_eTDMA(Ag1,bg1,Ti1,Te1,dx1)
    implicit none
    real, dimension(:,:), ALLOCATABLE, INTENT(OUT):: Ag1
    real, DIMENSION(:), ALLOCATABLE, INTENT(OUT):: bg1, Ti1, Te1
    real, INTENT(OUT) :: dx1
    real  :: aa, ap, xi
    integer :: i
    !This subroutine generates the tri-diagonal matric (TDMA) using our derived equations

    ALLOCATE(Ag1(n,n),bg1(n),Ti1(n),Te1(n)) !array allocation
    !Initializing Variables
```

```fortran
    Ag1=0
    xi=0
        dx1=2*pi/(n-1)
        aa=1/(dx1)**2
        ap=aa+aa
    do i = 1, n  !This loop Matrix composition of the given problem
        !for the the first gridpoint, applying B.Cs T(0)=1
        if ( i==1 ) then
            Ag1(i,i)=1
            bg1(i)=1
            Te1(i)=cos(xi) !This is T_exact from our analytical solution
        !for the the intermediate gridpoint
        else if ( i<n ) then
            Ag1(i,i-1)=aa
            Ag1(i,i)=-ap
            Ag1(i,i+1)=aa
            bg1(i)=-cos(xi)
            Te1(i)=cos(xi)
        !for the the last gridpoint, applying B.Cs T(2pi)=1
        else
            Ag1(n,n)=1
            bg1(n)=1
            Te1(n)=cos(xi)
        end if
        xi=xi+dx1
    end do
    call tdma(Ag1,bg1,Ti1)
end subroutine creat_eTDMA
!********************************************************************************

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

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

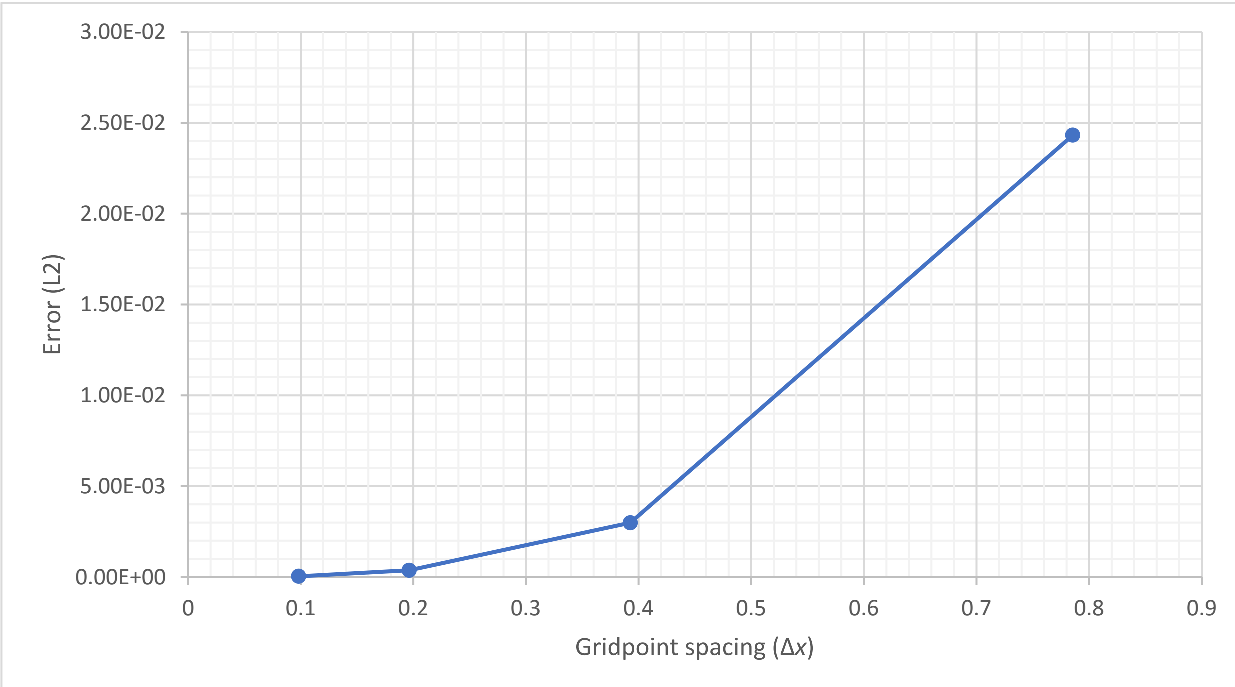
```
        f(k) = f(k) - e(k)*g(k-1)
    end do
   !Forward Substitution
    do k = 2,n
        b(k) = b(k) - e(k)*b(k-1)
    end do
   !Backward Substitution
    x(n)=b(n)/f(n)
    do k = n-1,1,-1  !(step size of -1)
        x(k) = (b(k) - g(k)*x(k+1))/f(k)
    end do
end subroutine tdma
!************************************************************************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

end program finit_diff
```

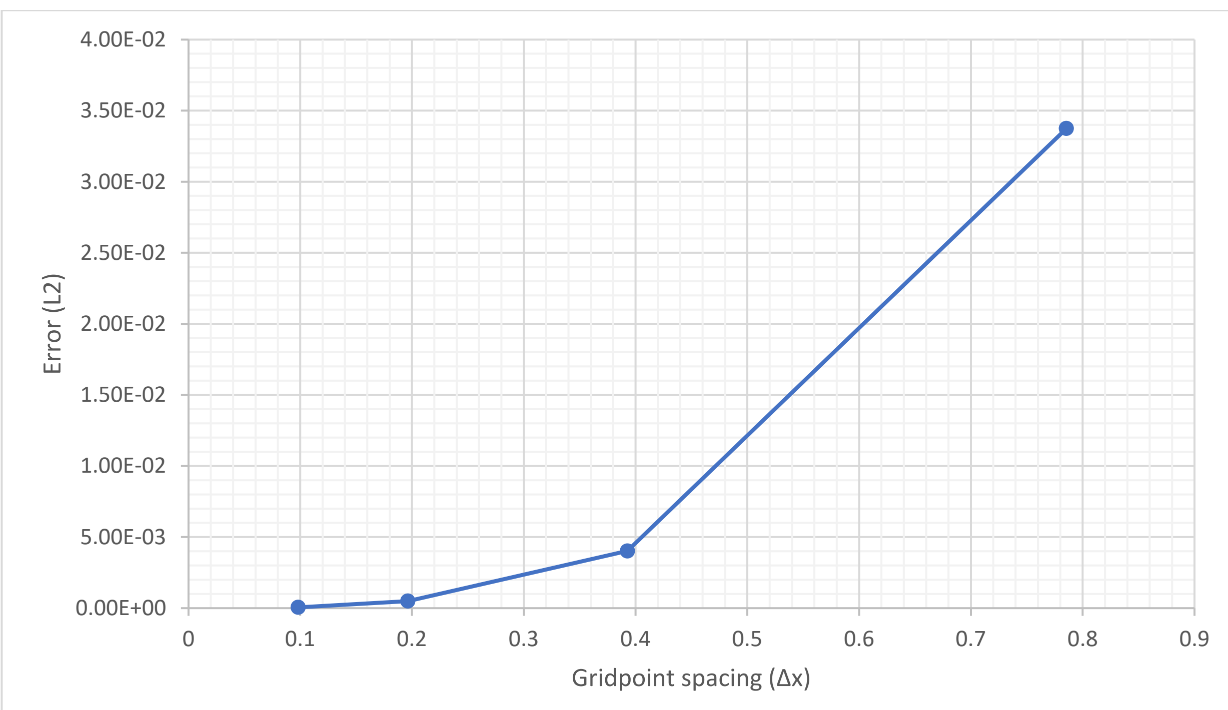
```
For number of grid points, n =  9
T(x) =    1.000    0.692   -0.053   -0.798   -1.106   -0.798   -0.053    0.692
1.000
  L1 =   0.42423
  L2 =   0.03375
L∞ =   0.10606
For number of grid points, n = 17
T(x) =    1.000    0.923    0.703    0.375   -0.013   -0.401   -0.729   -0.949
-1.026   -0.949   -0.729   -0.401   -0.013    0.375    0.703    0.923    1.000
  L1 =   0.20721
  L2 =   0.00403
L∞ =   0.02590
For number of grid points, n = 33
T(x) =    1.000    0.981    0.924    0.831    0.706    0.554    0.381    0.192
-0.003   -0.199   -0.387   -0.561   -0.713   -0.837   -0.930   -0.987   -1.006
-0.987   -0.930   -0.837   -0.713   -0.561   -0.387   -0.199   -0.003    0.193
0.381    0.554    0.706    0.831    0.924    0.981    1.000
  L1 =   0.10299
  L2 =   0.00050
L∞ =   0.00644
For number of grid points, n = 65
T(x) =    1.000    0.995    0.981    0.957    0.924    0.882    0.831    0.773
0.707    0.634    0.555    0.471    0.382    0.290    0.194    0.097   -0.001
-0.099   -0.196   -0.291   -0.384   -0.473   -0.557   -0.636   -0.708   -0.774
-0.833   -0.883   -0.925   -0.959   -0.982   -0.997   -1.002   -0.997   -0.982
-0.959   -0.925   -0.883   -0.833   -0.774   -0.708   -0.636   -0.557   -0.473
-0.384   -0.291   -0.196   -0.099   -0.001    0.097    0.194    0.290    0.382
0.471    0.555    0.634    0.707    0.773    0.831    0.882    0.924    0.957
0.981    0.995    1.000
  L1 =   0.05136
  L2 =   0.00006
L∞ =   0.00161
```

$Q2.d]$

Error Plot for Finite Volume Method:



Error Plot for Finite Difference Method:



Q 2. d] » In this contesct n is the number of discretized point for each run.

» From the above plot, I expected that as the number of discretization n, increases, the accuracy should also increase. This implies that the error reduces as the number of grid point increases.

» I would recomment Finite Volume method for this problem. Because the Obtained plots, shows that at similar~~ty~~ number of discretization (n), the error is lower for the Finite volume method. This is roughly about 74% more accurate than that of results from the finite difference method.

3.a] Given $\frac{\partial^2 T}{\partial x^2} + 100x^2 = 0$

We can solve this analytically by using ODE

$T(x) = C.F. + P.I.$, Same as we did in Q2.a

Re-arranging our eqn and puting $D = \frac{\partial}{\partial x}$

we get $(D^2)T = -100x^2$

so that we have an aux. eqn $m^2 = 0$

with roots that are equal and real

$m = m_1 = m_2 = 0$

The complemetary Fn, $C.F. = e^{mx}(c_1 + c_2 x)$ become

$C.F. = C_1 + C_2 x$

where C_1 & C_2 are some arbitary constants.

We have the R.H.S of our eqn is of the Type III form

So that the particular Integral

$$P.I = \frac{1}{f(D)} \cdot x^n$$

$$P.I = \frac{1}{D^2} \cdot (-100x^2)$$

$$P.I = \iint (-100x^2)\,\partial x^2 = \int \left(-\frac{100x^3}{3}\right)\partial x = -\frac{100x^4}{12}$$

$$P.I. = -\frac{25}{3}x^4$$

Substituting back C.F. & P.I into eqn $T(x)$, we get

$$T(x) = C_1 + C_2 x - \frac{25}{3}x^4 \quad \text{———} \quad (3.2)$$

now we can solve for the constants using B.Cs continued Q 3.a

For the first B.C. $\frac{\partial T}{\partial x}(0)=0$, let consider the derivative of eqn (3.2) w.r.t x, we get

$$\frac{\partial T}{\partial x}=C_2-\frac{100x^3}{3}$$

$$\frac{\partial T}{\partial x}(0)=C_2-\frac{100(0)^3}{3}=0$$

$$C_2=0$$

at end point B.C $T(1)=0$

$$T(1)=C_1+0(1)-\frac{25(1)^4}{3}=0$$

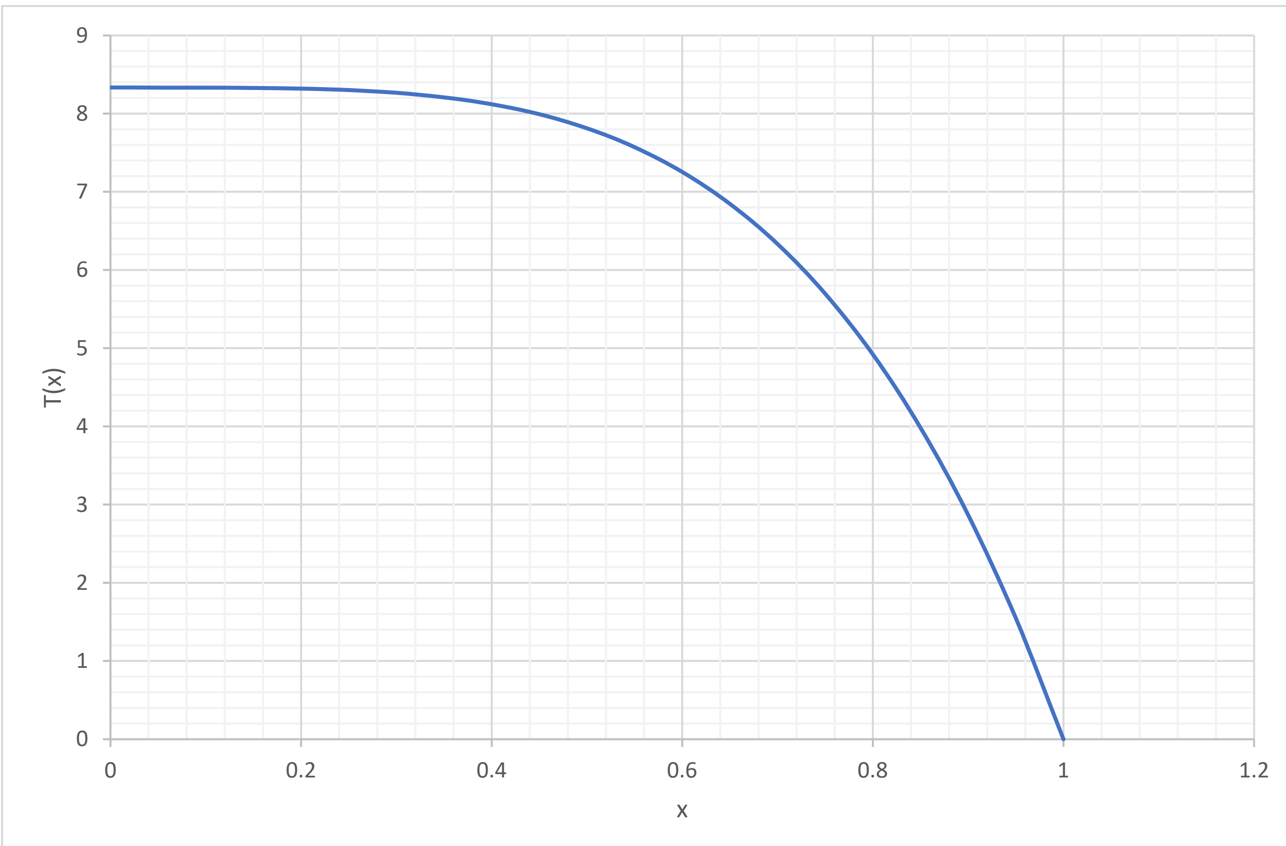
$$C_1=25/3$$

Therefore eqn (3.2) becomes $T(x)=\frac{25}{3}-\frac{25}{3}x^4$

$$\boxed{T(x)=\frac{25}{3}(1-x^4)}$$

3.b Using this solution we plot $T(x)$ for $0\leq x\leq 1$, as shown below, and we also discover T varies slowly near $x=0$, and rapidly near $x=1$.

$Q3. a]$



3.b] To solve for $T(x)$ for a non-uniform grid using finite difference, lets consider the derivative from Q1.C, but we remove the residual term (which contains the third order term)

we get, where $\phi = T$

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_i = \frac{(T_{i-1}-T_i)\frac{2}{\Delta x_-} + (T_{i+1}-T_i)\frac{2}{\Delta x_+}}{(\Delta x_- + \Delta x_+)}$$

So that for our system we will have

$$\frac{(T_W - T_P)\frac{2}{\Delta x_w} + (T_E - T_P)\frac{2}{\Delta x_e}}{(\Delta x_w + \Delta x_e)} + 100 x_P^2 = 0$$

where $\Delta x_w = x_P - x_W$ & $\Delta x_e = x_E - x_P$

For ease of computation, we can write this as,

$$\boxed{T_W a_W - T_P a_P + T_E a_E = -b}$$

where: $a_W = \frac{2}{\Delta x_w}$, $a_E = \frac{2}{\Delta x_e}$, $a_P = a_W + a_E$

$b = 100 x_P^2 (\Delta x_w + \Delta x_e)$

And at the first Boundary, we us Tyalor's series, and get

$$\frac{T_E - T_P}{\Delta x_e} = 0$$

$$\boxed{-T_P a + T_E a = 0}$$ where $a = 1/\Delta x_e$

also we can have $a = (1 - \Delta x^3)$ if we don't truncate the second order term of T.S.

And B.C. $T(1) = T_n = 0$, The Fortran 90 code is shown below;

```fortran
!****************Begin Header*****************************************************
!This program was written by Godswill Ezeorah, Student Number: 501012886 on May 20, 2021.
!This program solves a linear/non-linear equation using finite difference method
!and was written as a solution to AE8112 PS1 q3b
!**************************End Header*******************************************
program finit_diff_ugrid
    !Variable declaration
    implicit none
    real, dimension(:,:), ALLOCATABLE:: Ag
    real, DIMENSION(:), ALLOCATABLE:: bg, Ti, Te
    real, PARAMETER :: L=1.0 !Domain length parameter definition
    real  :: Alp, Linf
    integer :: N, Nd
    N=81   !The number of discretized points
    Alp=0.7
    Nd=19  !The number of times we are dividing Alpha (it must be an odd value)
    open(1, file = 'diffu_error.csv', status = 'unknown')
do while(Alp<=1.3 .and. Alp/=1)  !This will run for 0.7<α<1.3.
   call creat_uTDMA(Ag,bg,Ti,Te) !call the subroutine to solve for unequispaced grid
    Linf=maxval(abs(Ti(1:n)-Te(1:n)))
    !outputing the results
    print 3, 'For α =',Alp
    write(*,1)  "T(x) = ",Ti
    write(*,2) "L∞_u = ",Linf
    !writing one of the errors (Linf) as a function of Alpha (α) to excel file for ploting
    write(1,*) Alp, Linf
    Alp=Alp+(1.3-0.7)/Nd
end do
call creat_eTDMA(Ti,Te)    !call the subroutine to solve for equispaced grid
Linf=maxval(abs(Ti(1:n)-Te(1:n)))
write(*,2) "L∞_e= ", Linf
1 format(a6,81f9.3)
2 format(a7,9f9.5)
3 format(a10,f5.2)
close(1)

contains
!*******************************************************************************
subroutine creat_uTDMA(Ag1,bg1,Ti1,Te1)
    implicit none
    real, dimension(:,:), ALLOCATABLE, INTENT(OUT):: Ag1
    real, DIMENSION(:), ALLOCATABLE, INTENT(OUT):: bg1, Ti1, Te1
    real :: dx1
    real  :: aw, ae, ap, aa, xi, b
    integer :: i, Ns
    !This subroutine generates the tri-diagonal matric (TDMA) for unequal grid point

    ALLOCATE(Ag1(n,n),bg1(n),Ti1(n),Te1(n)) !array allocation
    !Initializing Variables
    Ag1=0
```

```
    xi=0
    Ns=n-1   !this is the number of segment
    dx1=L*(1-Alp)/(1-Alp**Ns)
!This is also noticed to yield same result as, aa=1/dx1, (they are enterchangable)
aa=1-dx1**3
    do i = 1, n  !This loop Matrix composition of the given problem
        aw=2/dx1
        ae=2/(Alp*dx1)
        ap=aw+ae
        b=100*(xi**2)*(dx1+Alp*dx1)
        !for the the first gridpoint, applying B.Cs T(0)=1
        if ( i==1 ) then
            Ag1(i,i)=-aa
            Ag1(i,i+1)=aa
            bg1(i)=0
        !for the the intermediate gridpoint
        else if ( i<n ) then
            Ag1(i,i-1)=aw
            Ag1(i,i)=-ap
            Ag1(i,i+1)=ae
            bg1(i)=-b
            dx1=Alp*dx1
        !for the the last gridpoint, applying B.Cs T(2pi)=1
        else
            Ag1(n,n)=1
            bg1(n)=0
            dx1=Alp*dx1
        end if
        Te1(i)=(25.0/3)*(1-xi**4)
        xi=xi+dx1
    end do
    call tdma(Ag1,bg1,Ti1)
end subroutine creat_uTDMA
!*******************************************************************************

!*******************************************************************************
subroutine creat_eTDMA(Ti1,Te1)
    implicit none
    real, dimension(:,:), ALLOCATABLE :: Ag1
    real, DIMENSION(:), ALLOCATABLE, INTENT(OUT):: Ti1, Te1
    real, DIMENSION(:), ALLOCATABLE:: bg1
    real:: dx1
    real  :: a, ap, aa, xi, b
    integer :: i, Ns
    !This subroutine generates the tri-diagonal matric (TDMA) for equal grid spacing

    ALLOCATE(Ag1(n,n),bg1(n),Ti1(n),Te1(n)) !array allocation
    !Initializing Variables
    Ag1=0
    xi=0
    Ns=n-1   !this is the number of segment
    dx1=L/(n-1)
```

```
    aa=1/dx1  !This is also noticed to yield same result as, aa=1/dx1, (they are enterchanga
ble)
    do i = 1, n  !This loop Matrix composition of the given problem
        a=1/dx1**2
        ap=a+a
        b=100*(xi**2)
        !for the the first gridpoint, applying B.Cs T(0)=1
        if ( i==1 ) then
            Ag1(i,i)=-aa
            Ag1(i,i+1)=aa
            bg1(i)=0
        !for the the intermediate gridpoint
        else if ( i<n ) then
            Ag1(i,i-1)=a
            Ag1(i,i)=-ap
            Ag1(i,i+1)=a
            bg1(i)=-b
        !for the the last gridpoint, applying B.Cs T(2pi)=1
        else
            Ag1(n,n)=1
            bg1(n)=0
        end if
        Te1(i)=(25.0/3)*(1-xi**4)
        xi=xi+dx1
    end do
    call tdma(Ag1,bg1,Ti1)
end subroutine creat_eTDMA
!*******************************************************************************

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

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

A sample result (N/B: this is not asked for, in the assignment):

```
  For α = 0.92
T(x) =    8.343    8.343    8.339    8.325    8.293    8.237    8.153    8.038
7.893    7.719    7.517    7.292    7.047    6.786    6.511    6.228    5.940
5.649    5.358    5.070    4.787    4.510    4.242    3.982    3.732    3.492
3.263    3.046    2.839    2.643    2.458    2.284    2.120    1.966    1.822
1.687    1.561    1.443    1.334    1.231    1.136    1.048    0.965    0.889
0.818    0.752    0.691    0.635    0.583    0.534    0.490    0.448    0.410
0.375    0.342    0.312    0.284    0.258    0.235    0.213    0.192    0.174
0.156    0.141    0.126    0.112    0.100    0.088    0.078    0.068    0.059
0.051    0.043    0.036    0.030    0.024    0.018    0.013    0.008    0.004
0.000
L∞_u   0.01247
  For α = 0.95
T(x) =    8.337    8.337    8.336    8.334    8.328    8.318    8.301    8.275
8.241    8.196    8.140    8.073    7.994    7.904    7.802    7.690    7.568
7.436    7.295    7.146    6.989    6.826    6.658    6.485    6.308    6.128
5.945    5.761    5.576    5.390    5.205    5.021    4.837    4.656    4.476
4.300    4.125    3.954    3.787    3.623    3.462    3.305    3.153    3.004
2.859    2.719    2.582    2.450    2.322    2.198    2.078    1.963    1.851
1.743    1.639    1.539    1.443    1.350    1.260    1.175    1.092    1.013
0.937    0.864    0.794    0.726    0.662    0.600    0.541    0.484    0.430
0.378    0.328    0.280    0.235    0.191    0.149    0.109    0.071    0.035
0.000
L∞_u   0.00447
  For α = 0.98
T(x) =    8.333    8.333    8.333    8.333    8.333    8.332    8.331    8.329
8.327    8.323    8.319    8.313    8.305    8.296    8.284    8.270    8.254
8.236    8.214    8.190    8.162    8.132    8.098    8.060    8.019    7.974
7.925    7.872    7.815    7.755    7.690    7.621    7.547    7.470    7.388
7.302    7.212    7.118    7.019    6.916    6.809    6.698    6.583    6.464
6.341    6.214    6.083    5.948    5.810    5.668    5.523    5.374    5.222
5.066    4.908    4.746    4.581    4.414    4.243    4.070    3.895    3.717
3.537    3.354    3.169    2.982    2.793    2.603    2.410    2.216    2.021
1.824    1.625    1.425    1.225    1.023    0.820    0.616    0.411    0.206
0.000
L∞_u   0.00041
```

3.c] >> I expect α to be confined to ~~$0.9 \geq \alpha \leq 1$~~

$0.9 < \alpha < 1$, because within this range, we should get more accurate result from the finite difference method (i.e L_∞ is small within this range)

>> $\boxed{\alpha = 0.99575883}$, we will get a minimum value of $L_\infty = 3.973e^{-5}$

>> From the simulation for equispaced grid, we have $L_\infty = 0.00131$. This Larger error value ~~for same~~ in comparison to the non-uniform grid solution, at $\alpha = 0.99575883$.

~~3.b]~~

3.d] The Plot of L_∞ as a fn of α is Shown below, and I observe that the erro is low at 0.7 and gently reduces as the curve approaches 1. And after about 1.14, the error curve increases exponentialy towards 1.3.

This shows that the α should be very close to 1 but not equal to 1 for us to have a more accurate solution.

$Q3.d]$