COMPUTER PROJECT #1

Shaun Harris
Department of Mechanical and Aerospace Engineering
Utah State University
Email: shaun.r.harris@gmail.com

ABSTRACT

A one by one unit plate is provided. The $\nabla^2\phi = 0$ equation along with specified boundary conditions is provided. The diffusion equation is solved using a finite-volume method. The discretized equations are shown, the methodology is outlined, and the results are provided in this paper.

NOMENCLATURE

Γ Diffusion coefficient
ϕ Property of fluid in 2-d steady state diffusion
ϕ_{old}, ϕ_{new} Previous and next iteration guess of ϕ solution
S Source term value
$\bar{S}$ Average value of source S
S_u, S_p Coefficients of linear approximation for source term
Δn Width of cell in n direction
δn_{ij} Width between i and j cell centers in n direction
a_i Coefficient of finite-volume method of cell i
A_i Cross-sectional area of i cell face
Ω Successive over-relaxation factor for numerical method
$\Delta\phi$ Successive over-relaxation iteration change for numerical method
ΔV Volume of finite-volume
i, j Coordinates used in stencil and locations for numerical method

INTRODUCTION

The finite-volume method was used to solve the $\nabla^2\phi = 0$ on a one by one unit plate. The boundary conditions were specified $\phi(0,y) = 0$; $\phi(1,y) = y$; $\phi(x,0) = 0$; $\phi(x,1) = x$. The analytical solution was known to be $\phi = x\,y$ and an initial guess of $\phi = 0$ was provided for the interior cells. We used a mesh consisting of 20 by 20 cells. The *thin* boundary cells of one around the perimeter also provided a slightly larger mesh of 22 by 22 cells. Though it needs to be remembered that the boundary conditions for ϕ did not change with each iteration. Thus, only the 20 by 20 cells were iterated over using the point SOR scheme discussed in class.

In order to get problem to be solved using the finite-volume method we needed to find the discretized equations for the system. The interior cells are given by equation 2 referenced from the stencil drawn in Figure 1. Each cell center has an associated ϕ, x, and y values. These values are referenced with the i, j coordinates.

It can be seen easily that this equation will work for all the interior cells. But once the boundary is reached, the *ghost* cells must be used. These cells have a Δx or Δy value that is very *thin*. With this in mind, equation 2 is still valid. The only values that change are on these boundaries, where δx and/or δy produces a value that is half the normal distance between cells. Thus, instead of a one, a two is used in the coefficient value. We used the equation shown in equation 2 to calculate on each cell iteration. In order to use the method of successive over-relaxation we can rearrange equation 2 to equation 3 as shown.

Using this iterative SOR technique, if we iterate until the Root Sum Squared values are small, than we can find a converged solution for the equation $\nabla^2\phi = 0$.

The results are show in figure 5 and the associated error is show in figure 6.

NUMERICAL METHOD

Chapter 4 of our textbook [1] indicates the steps to solving this problem using the finite-volume approach. The following steps are outlined and followed in this problem.

1. Grid generation
2. Discretization of equations
3. Solution of equations

Each step will be outlined below for a general case two-dimensional case and then it will be applied to our simplified $\nabla^2\phi = 0$ equation.

Figure 1 shows a stencil for the grid. This shows how the grid is generated for a two dimensional plate. We only show an example representation of the grid in figure 1, it is noted that we broke the grid into 20 by 20 cells. Additionally, the boundary conditions were treated as additional cells with a small cell area Δx or $\Delta y \to 0$. This allowed for our boundary conditions to be accounted for. We will refer to these cells as the boundary cells. The origin of the grid is in the lower left hand corner.

To complete the discretization of the equations over a control volume, we will first look at a nodal point P. The governing equation is given by equation 1 for a two dimensional steady state diffusion. This is a general equation that is used for finite volume methods.

$$\begin{aligned}
\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right)+S_\phi = 0 \\
\int_{\Delta V}\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right)dx\,dy+\int_{\Delta V}\frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right)dx\,dy+\int_{\Delta V}S_\phi dV = 0 \\
\left[\left(\Gamma A\left(\frac{\partial\phi}{\partial x}\right)\right)_e-\left(\Gamma A\left(\frac{\partial\phi}{\partial x}\right)\right)_w\right]... \\
+\left[\left(\Gamma A\left(\frac{\partial\phi}{\partial y}\right)\right)_n-\left(\Gamma A\left(\frac{\partial\phi}{\partial y}\right)\right)_s\right]... \\
+\bar{S}\Delta V = 0
\end{aligned} \tag{1}$$

We can then substitute linear interpolations for Γ_w, Γ_e, Γ_n, and Γ_s to get these terms in terms of cell values and constants. Then we can substitute $\bar{S}\Delta V = S_u + S_p\phi_P$ to find the average value of source S in terms of a linear approximation. Note that equation 1 uses substitution of the flux through the cell faces and results in the discretized equation 2. This equation is used in the numerical method. Also, we can substitute the values of Γ and S provided by the $\nabla^2\phi = 0$ equation, which gives us equation 2

$$a_P\phi_P = a_W\phi_W + a_E\phi_E + a_S\phi_S + a_N\phi_N + S_u$$

where:

$$\begin{aligned}
a_W &= \frac{\Gamma_w A_w}{\delta x_{WP}}, a_E = \frac{\Gamma_e A_e}{\delta x_{EP}}, a_S = \frac{\Gamma_s A_s}{\delta y_{SP}}, a_N = \frac{\Gamma_n A_n}{\delta y_{NP}} \\
a_P &= a_W + a_E + a_S + a_N - S_p \\
S_u &= S_p = 0 \\
\Gamma_w &= \Gamma_e = \Gamma_s = \Gamma_n = 1 \\
A_w &= A_e = \Delta x \\
A_s &= A_n = \Delta y \\
\delta x_{WP} &= |x_{i,j} - x_{i,j-1}| \\
\delta x_{EP} &= |x_{i,j} - x_{i,j+1}| \\
\delta y_{SP} &= |y_{i,j} - y_{i-1,j}| \\
\delta y_{NP} &= |y_{i,j} - y_{i+1,j}|
\end{aligned} \tag{2}$$

Taking equation 2 we can apply this to our 20 by 20 finite volume. Each cell will apply this equation. In order to apply this equation and solve numerically for the resulting ϕ solution we will use a successive over-relaxation method. Taking equation 2 and solving for ϕ_P and then taking the right hand side and adding and subtracting ϕ_P yields equation 3.

$$\phi_{new} = \phi_{old} + \frac{\Omega}{a_P}\Delta\phi$$

where:

$$\begin{aligned}
\Omega &\equiv \text{Successive over-relaxation factor} \approx 1.8 \\
\Delta\phi &= a_E\phi_{Enew} + a_W\phi_{Wnew} + a_S\phi_{Snew} + a_N\phi_{Nnew} - a_P\phi_{Pold}
\end{aligned} \tag{3}$$

Using this we can iterate through each cell and produce a new approximated solution for ϕ. If we iterate the approximated solutions enough times until equation 4 is satisfied,then we will find the numerical diffusive value of ϕ.

$$\sum_{i=1}^{400}\sqrt{\Delta\phi_i} \le 10^{-14} \tag{4}$$

We can then compare this to the true analytical solution $\phi = x\ y$ and see the error produced. We should expect very close agreement with the analytical and the numerical solution values. They should only disagree by machine precision values and our convergence criteria.

Boundary conditions must be taken into consideration. On the cells nearest the boundaries, the cells will use the boundary

condition values from the boundary cells. They also use the x and y values of the boundary cells. The values of the δx and δy is taken into account by equation 2. This provides the necessary equations for the boundaries using the boundary cells.

RESULTS

Figure 2 shows a plot comparing various values of Ω relaxation factor and the optimal value to converge this finite-volume solution. This backs up the reason for selecting $\Omega = 1.8$ in the numerical method described above.

We can see how quickly these values converged by looking at figure 3. This figure shows on a log-log scale how quickly our solution of $\Omega = 1.8$ converged to machine zero. We can see that after a 150 iterations or so, the solution does not converge much more after this point.

To reduce the amount of iterations, we can use the stopping criteria specified in equation 4. Figure 4 shows the convergence plot. The solution does not converge more after the 152 iterations required to reach this stopping criteria. The other solutions in figures 5 and 6 are results from 152 iterations using $\Omega = 1.8$.

Figure 5 shows the values of phi using the SOR method described. We can see that the values are very close to the analytical solution of $x\,y$.

The error between these two solutions is shown in figure 6. We can see that every cell has an error that is less than or equal to 10^{-15}.

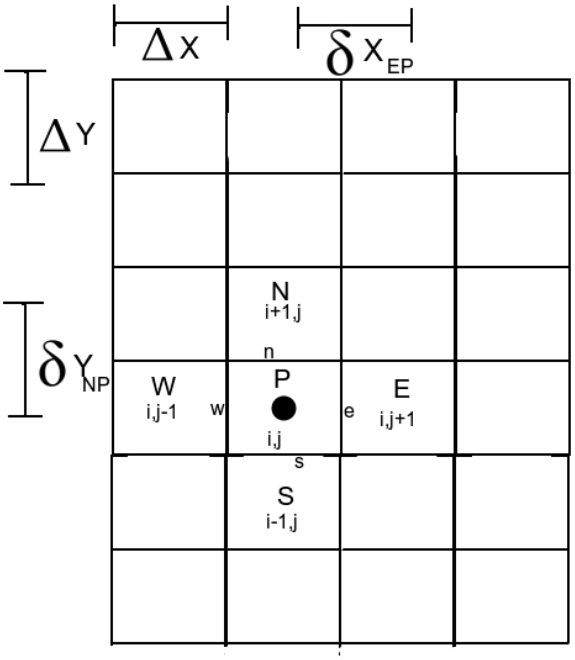

FIGURE 1. REPRESENTATION OF THE STENCIL FOR THE FINITE VOLUME APPROACH

CONCLUSION

We have shown the use of a finite-volume approach to solving the flat plate of one square unit with 400 cells using a successive over-relaxation approach to solve the diffusive $\nabla^2\phi = 0$ equation. We used the discretized equation shown in equation 2. Iterating over this equation with the successive over-relaxation provides us with a numerical approximation for the solution of ϕ at each location on the plate. This numerical approximation method taught me the steps to using the finite-volume method and how to implement it into a simple diffusive equation. These same steps can be used in other applications to solve other equations. This analysis can be used in many various situations to solve many harder applications.

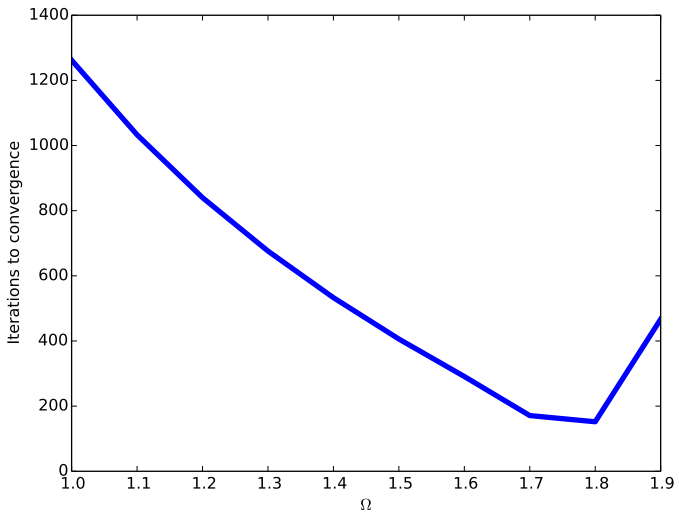

FIGURE 2. PLOT SHOWING ITERATIONS TO NECESSARY CONVERGENCE FOR EACH VALUE OF Ω RELAXATION FACTOR

REFERENCES

[1] Versteeg, H. K., and Malalasekera, W., 2007. *An Introduction to COMPUTATIONAL FLUID DYNAMICS The Finite Volume Method*. PEARSON Prentice Hall.

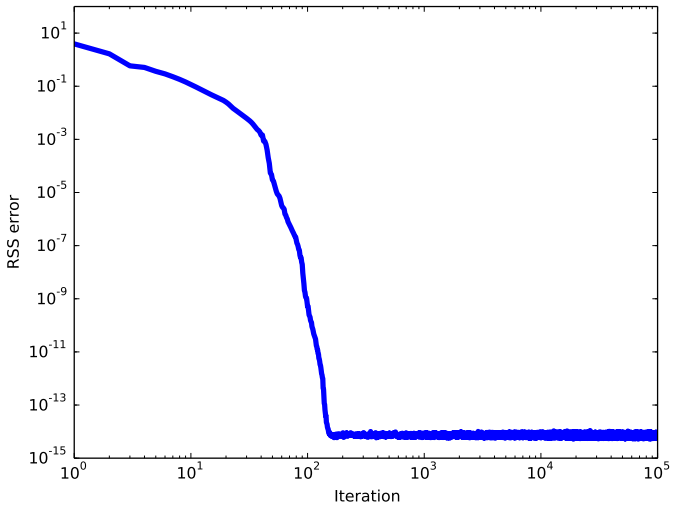

FIGURE 3. CONVERGENCE LOG-LOG PLOT OF THE ITERATION SOLUTION FOR MANY ITERATIONS

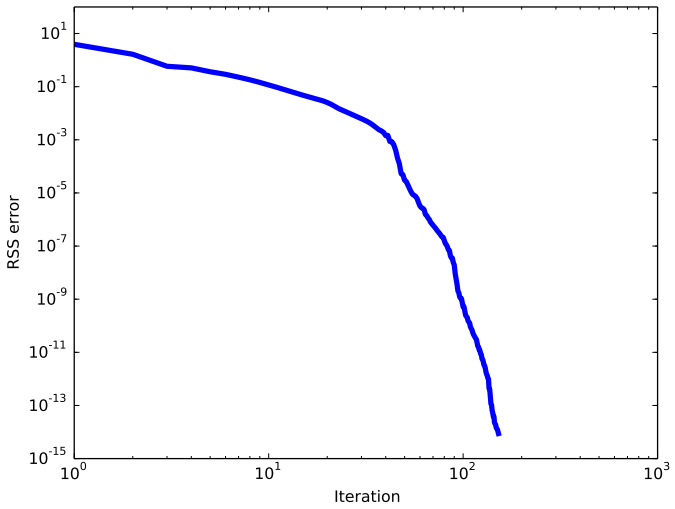

FIGURE 4. CONVERGENCE LOG-LOG PLOT OF THE ITERATION SOLUTION FOR 152 ITERATIONS

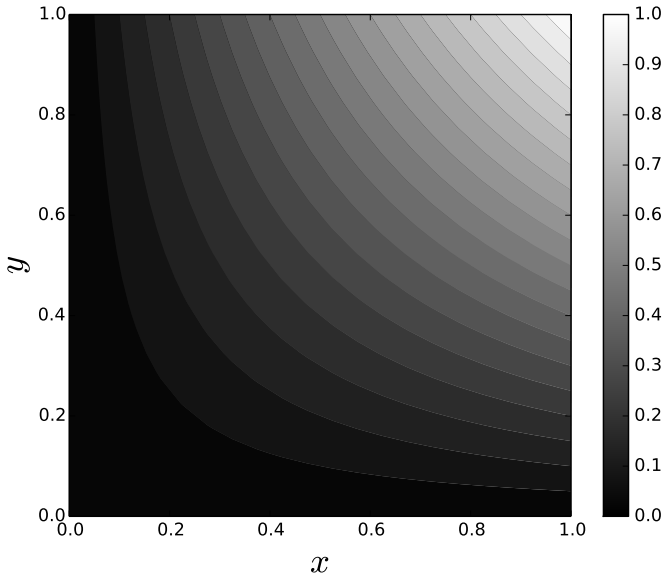

FIGURE 5. CONVERGED SOLUTION ϕ CONTOUR PLOT

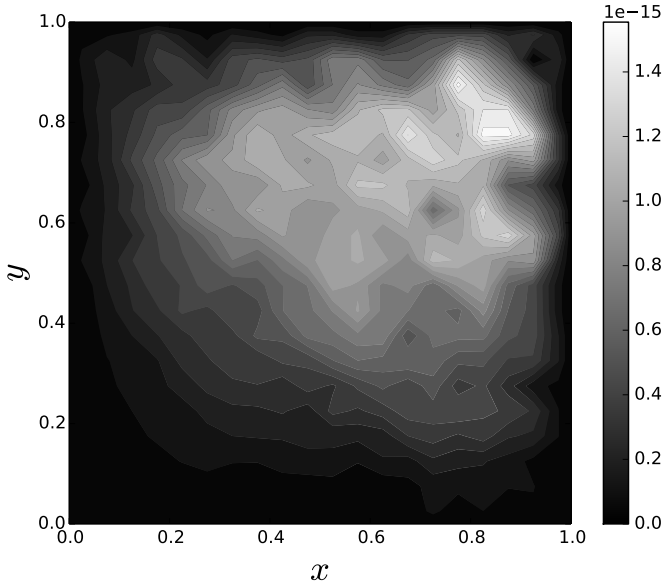

FIGURE 6. CONTOUR PLOT SHOWING THE ERROR BETWEEN THE NUMERICAL AND ANALYTICAL SOLUTION

Appendix A: Code

```
! user defined variables to define finite volume
! x and y direction # of cells
#define max_x 20
#define max_y 20
! x and y number of cells plus 1
#define max_xp 21
#define max_yp 21
! x and y number of cells plus 2 (to account for boundary nodes)
#define max_x2p 22
#define max_y2p 22
MODULE types
    !purpose: define data type struct
    IMPLICIT NONE
    TYPE::dat
        REAL::x,y,phi,phi_new
        INTEGER::n
    END TYPE dat
CONTAINS
    SUBROUTINE set_xy (strct,dx,dy,nx,ny)
        REAL,INTENT(IN)      ::  dx,dy
        INTEGER,INTENT(IN)   ::  nx,ny   ! size of strct in x and y directions
        TYPE(dat),DIMENSION(0:nx-1,0:nx-1),INTENT(INOUT)::strct ! data contained from 0:nx-1 where cells 0 and nx-1 are boundary nodes (cell volume approaches 0 on boundary nodes)
        INTEGER ::  i,j,n   ! for do loops and n is counter for cell number
        REAL    ::  xi,yi   ! x and y values for each cell
        n=1                 ! cell number 1
        DO i=1,ny-2         ! 1 to ny-2 for boundary nodes (we only are iterating through the middle values)
            yi = i*dy - dy/2.   ! y coordinate
            DO j=1,nx-2
                xi = j*dx - dx/2.       ! x coordinate
                strct(i,j)%n = n        ! input n node
                strct(i,j)%x = xi       ! x coordinate to strct
                strct(i,j)%y = yi       ! y coordinate to strct
                strct(i,j)%phi = 0.     ! phi value initialized guess
                strct(i,j)%phi_new = 0. ! phi value initialized guess for next iteration
                n=n+1                   ! count cell numbers up one
            END DO
        END DO
    END SUBROUTINE set_xy
END MODULE types

PROGRAM project1
    USE types !use module defined by types
    IMPLICIT NONE
    ! declare variables
    INTEGER ::  i,j,iter!,max_x=20,max_y=20
    REAL    ::  dx,dy,gamma,deltaPhi,ae,aw,an,as,ap,error
    TYPE(dat),DIMENSION(0:max_xp,0:max_yp)::phi ! 22 if you count edges (thin cell)
    ! set dx and dy and gamma and coefficients (without dividing by delta x between node centers)
    dx=1./REAL(max_x)
    dy=1./REAL(max_y)
    gamma = 1.
    ! initialize phi data and x,y for middle values
    CALL set_xy(phi,dx,dy,max_x2p,max_x2p)
    !! initialize BC's
    ! left Boundary
    phi(:,0)%x = 0.
    phi(0,0)%y = 0.
    phi(1:max_x,0)%y = RESHAPE((/ (i*dy - dy/2. ,i=1,max_x) /),(/ max_x /))
    phi(max_xp,0)%y = 1.
    phi(:,0)%phi = 0.
    phi(:,0)%phi_new = 0.
    ! top boundary
    phi(0,0)%x = 0.
    phi(0,1:max_y)%x = RESHAPE((/ (i*dx - dx/2. ,i=1,max_y) /),(/ max_y /))
    phi(0,max_yp)%x = 1.
    phi(0,:)%y = 0.
    phi(0,:)%phi = 0.
    phi(0,:)%phi_new = 0.
    ! right boundary
    phi(:,max_yp)%x = 1.
    phi(:,max_yp)%y = phi(:,0)%y
    phi(:,max_yp)%phi = phi(:,max_yp)%y
    phi(:,max_yp)%phi_new = phi(:,max_yp)%y
    ! bottom boundary
    phi(max_xp,:)%x = phi(0,:)%x
    phi(max_xp,:)%y = 1.
    phi(max_xp,:)%phi = phi(max_xp,:)%x
    phi(max_xp,:)%phi_new = phi(max_xp,:)%x
    ! point SOR method to solve for the exact values of phi using the BC (only loop through inner 20X20 values)
    open(unit=5,file="convergence.txt")
    DO iter=0,100000
        error = 0.
        DO i=max_y,1,-1
            DO j=max_x,1,-1
                ae=dy*gamma/(abs(phi(i,j)%x-phi(i,j+1)%x))
                aw=dy*gamma/(abs(phi(i,j)%x-phi(i,j-1)%x))
                an=dx*gamma/(abs(phi(i,j)%y-phi(i+1,j)%y))
                as=dx*gamma/(abs(phi(i,j)%y-phi(i-1,j)%y))
                ap=(ae+aw+an+as)
                deltaPhi = ae*phi(i,j+1)%phi_new + &     ! use new iteration values
                    aw*phi(i,j-1)%phi_new + &            ! use new iteration values
                    an*phi(i+1,j)%phi_new + &            ! use new iteration values
                    as*phi(i-1,j)%phi_new - &            ! use new iteration values
                    ap*phi(i,j)%phi                      ! use old iteration values
                phi(i,j)%phi_new = phi(i,j)%phi + 1.8/ap * deltaPhi     ! each iterations approximate new phi value
                error = error + deltaPhi**2              ! root sum square the error for each iteration
            END DO
        END DO
        error = SQRT(error)                              ! sqrt to have RSS of error
        write(5,*) iter,error
        phi%phi = phi%phi_new                            ! set old phi to the new iteration guess
        IF (error .lt. 1.E-14) THEN
            WRITE(*,*) 'Did in ',iter,' iterations'
            WRITE(*,*) 'With RSS error = ',error
            EXIT                                         ! exit loop if error is small enough
        END IF
    END DO
    !output
    ! user will need to specify size of
    open(unit=9,file="x.txt");open(unit=10,file="y.txt");open(unit=11,file="phi.txt");open(unit=12,file="error.txt")
    100 FORMAT (max_x2p F10.6)
    200 FORMAT (max_x2p ES16.6)
    WRITE(9,100) ( phi(i,:)%x ,i=0,21 )
    WRITE(10,100) ( phi(i,:)%y ,i=0,21 )
    WRITE(11,100) ( phi(i,:)%phi ,i=0,21 )
    WRITE(12,200) ( abs(phi(i,:)%phi - (phi(i,:)%x*phi(i,:)%y)),i=0,21 )
    close(9);close(10);close(11);close(12);close(5);
END PROGRAM project1
```