CE-580

COMPUTATIONAL TECHNIQUES

FOR

FLUID DYNAMICS

HOMEWORK #8

Finite Volume Solution to Turbulent Pipe Flow Using Wall Functions

Taha Yaşar Demir

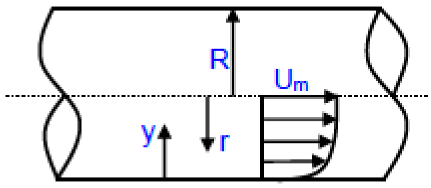
1881978

CE 580 COMPUTATIONAL TECHNIQUES FOR FLUID DYNAMICS

Homework 8

Finite Volume Solution to Turbulent Pipe Flow Using Wall Functions

Develop a computer program to obtain velocity distribution from an implicit FV solution to steady, uniform, turbulent flow in a circular pipe.



1. Use mixing length theory as the turbulence model.

2. Apply the boundary conditions

$$u = 0 \text{ at } y = 0 \qquad \text{and} \qquad u = U_m \text{ at } y = R.$$

3. Obtain initial data for velocity from the power law: $u(y) = U_m\,(y/R)^{1/7}$

4. Consider the steady momentum equation. Obtain a finite volume discretization for variable mesh size. Generate the computational grid using constant ratio method. Describe an implicit solution to the discretized FV equations.

5. Obtain the wall shear stress from the logarithmic law of the wall. Describe the percent change in shear velocity in two consecutive iterations as computational error and terminate iterations when this change is negligible.

6. Determine the Reynolds number and the friction factor (f_m) from the experimental formula. Compute the friction factor (f_c) using the computed average velocity and the wall shear stress in the Darcy's equation. Compute the percent error between the experimental and computed friction factors.

7. Run your program for the data given below:

   $U_m = 4$ m/s $R = 0.1$ m $N = 20$
   $\nu = 1\text{x}10^{-6}$ m^2/s $\rho = 1000$ kg/m^3 $\varepsilon_{max} = 1\text{x}10^{-5}$
   Grid ratio (G) = 1, 0.95, 0.90 and 0.86

8. Make logarithmic plots of the velocity profiles ($u^+ \sim y^+$) on the same page.

9. Prepare a table of results, presenting G, y_1^+, f_c, $100^*|f_c\text{-}f_m|/f_m$, and number of iterations.

10. Write a discussion on the results you obtained.

Calculations

Assumptions

- Steady, Uniform Flow
- Turbulent Flow
- Smooth Pipe
- Axisymmetric Domain

Simplified momentum equation in radial coordinates

$$\frac{1}{r}\frac{\partial}{\partial y}\left[r\nu_e\frac{\partial u}{\partial y}\right] = -\frac{1}{\rho}\frac{\partial p}{\partial x} = C_p$$

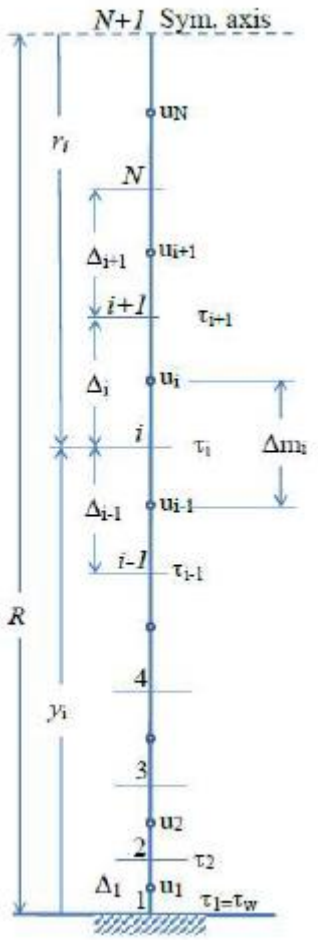
In terms of shear stress

$$\frac{d(r\tau)}{dy} = rC_p$$

Central differences used for discretization

$$\frac{(r\tau)_{i+1} - (r\tau)_i}{\Delta_i} = (r_m)_i C_p$$

Grid system used is



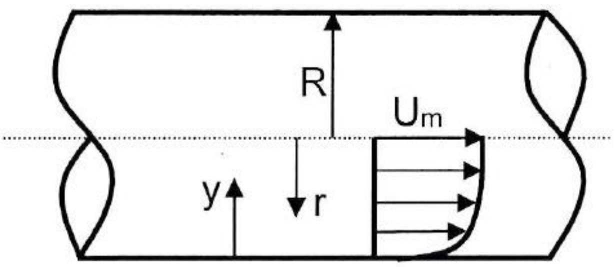


Figure 1: Grid System and Domain

Using central differences for control points

$$r_{i+1}(\nu_e)_{i+1}\frac{u_{i+1}-u_i}{(\Delta_m)_{i+1}}-r_i(\nu_e)_i\frac{u_i-u_{i-1}}{(\Delta_m)_i}=\Delta_i(r_m)_iC_p$$

Defining new parameters for constants

$$a_i=\frac{r_i}{(\Delta_m)_i}\ ,\qquad d_i=\Delta_i(r_m)_i\ ,\qquad \Delta_i=y_{i+1}-y_i\ ,\qquad (\Delta_m)_i=\frac{(\Delta_i-\Delta_{i-1})}{2}$$

The tridiagonal form is

$$-A_iu_{i+1}+B_iu_i-C_iu_{i-1}=D_i$$

And coefficients A, B, C, D are

$$A_i=a_{i+1}\nu_{e_{i+1}}\ ;\ C_i=a_i\nu_{e_i}\ ;\ B_i=A_i+C_i\ ;\ D_i=-d_iC_p$$

Effective viscosity and Turbulent viscosity are calculated using mixing-length theory.

$$\nu_e=\nu+\nu_t$$

$$\mu_t=\rho\nu_t=\rho l_m^2\frac{du}{dy}$$

Where

$$l_m=H\left[0.14-0.08\left(1-\frac{y}{H}\right)^2-0.06\left(1-\frac{y}{H}\right)^4\right]f_\mu$$

$$f_\mu=1-\exp\left(-\frac{y^+}{A^+}\right)\qquad A^+=26$$

$$y^+=\frac{yu_*}{\nu}\qquad u_*=\sqrt{\frac{\tau_w}{\rho}}$$

Instead of calculating wall shear,u_* is calculated at the first control point that is outside of the viscous sublayer using wall functions. Then wall shear is calculated using above relation.

$$\frac{u}{u_*}=\frac{1}{\kappa}\ln(y^+)+\beta$$

$$\kappa=0.41\ ,\qquad \beta=5.3$$

After calculating wall shear, Cp can be determined using the relation

$$C_p=\frac{\partial p}{\partial x}=-2\frac{\tau_w}{Radius\times\rho}$$

Grid is generated with constant ratio method Ratio between any two neighboring meshes is constant

$$\Delta Y_i=G\Delta_{i+1}$$

Distance between the last two mesh point can be calculated as

$$\Delta Y_N=\frac{H}{\sum_{i=0}^{N-2}G^i}$$

Mesh distribution can be completed marching down from N to 1

Boundary Conditions

At (i=1) wall the momentum equation is

$$a_2(\nu_e)_2(u_2 - u_1) - R\tau_w = d_1 C_p$$

After finding wall shear stress as described above, Define a relation for wall shear

$$C_w = \frac{\tau_w}{u_1}$$

Then the momentum balance for the first cell is

$$a_2(\nu_e)_2(u_2 - u_1) - RC_w u_1 = d_1 C_p$$

In tridiagonal form coefficients A, B, C, D will be

$$-A_1 u_2 + B_1 u_1 - C_1 u_0 = D_1$$

$$A_1 = a_2(\nu_e)_2 \ , \qquad C_1 = 0 \ , \qquad B_1 = A_1 + C_w R \ , \qquad D_1 = -d_1 C_p$$

Note that u_0 is not defined, For no slip boundary condition on the wall:

$$u_w = E_0 u_1 + F_0 = 0 \ , \qquad E_0 = 0 \ ; \ F_0 = 0$$

At the centerline (i=N) momentum balance is

$$a_{N+1}(\nu_e)_{N+1}(u_{N+1} - u_N) - a_N(\nu_e)_N(u_N - u_{N-1}) = d_N Cp$$

In tridiagonal form

$$-A_N u_{N+1} + B_N u_N - C_N u_{N-1} = D_N$$

Where coefficients A, B, C, D are

$$A_N = 0 \ (At \ symmetry \ line \ u_{N+!} = u_N)$$

$$C_N = a_N(\nu_e)_N \ , \qquad B_N = C_N \ , \qquad D_N = -d_N C_p$$

Initial condition

An initial velocity distribution to start the solution procedure is get from the power law.

$$u(y) = U_{max}\left(\frac{y}{R}\right)^{\frac{1}{7}}$$

Error Calculation

Error is defined as relative change in shear velocity, such as

$$Error = \frac{|u_{*old} - u_*|}{u_{*old}}$$

Outputs

Required outputs can be calculated once the solution is converged

Discharge

$$Q = \int u \times dA$$

Above integral can be taken numerically as such

$$Q = 2\pi \sum_{i=2}^{N} \frac{r_i + r_{i-1}}{2} \times \frac{(u_i + u_{i-1})}{2} \times \Delta_i$$

Average velocity

$$V_{ave} = \frac{Q}{A}$$

Reynolds Number

$$Re = \frac{V_{ave} \times D}{\nu}$$

Friction factors

From experimental data, using Swamee-Jain formula

$$f_m = \frac{0.25}{\left[\log\left(\frac{k_s}{3.7D} + \frac{5.74}{Re^{0.9}}\right)\right]^2}$$

Since we assumed smooth pipe we can drop k_s term and equation becomes

$$f_m = \frac{0.25}{\left[\log\left(\frac{5.74}{Re^{0.9}}\right)\right]^2}$$

From the converged solution, using Darcy's friction factor

$$f_d = \frac{8\tau_w}{\rho V_{ave}^2}$$

Input Data

$$U_{max} = 4\frac{m}{s} \ , \qquad R = 0.1\ m \ , \qquad N = 20$$

$$\nu = 1 \times 10^{-6}\frac{m^2}{s} \ , \qquad \rho = 1000\frac{kg}{m^3} \ , \qquad \epsilon_{max} = 1 \times 10^{-5}$$

$$Grid\ Raito = G = 1; 0.95; 0.90; 0.86$$

Results and Discussion

Using wall functions saves us from grid refinement on solid boundaries. However, there is still a restriction on y_+ value at the first grid point. It should be between 30 and 300, in other words, first grid point should be at fully turbulent region of the boundary layer. The reason is that wall function that is used (log law) is valid at turbulent region. If y_+ value is in the valid range, wall shear can be estimated and using wall shear pressure drop can be calculated.

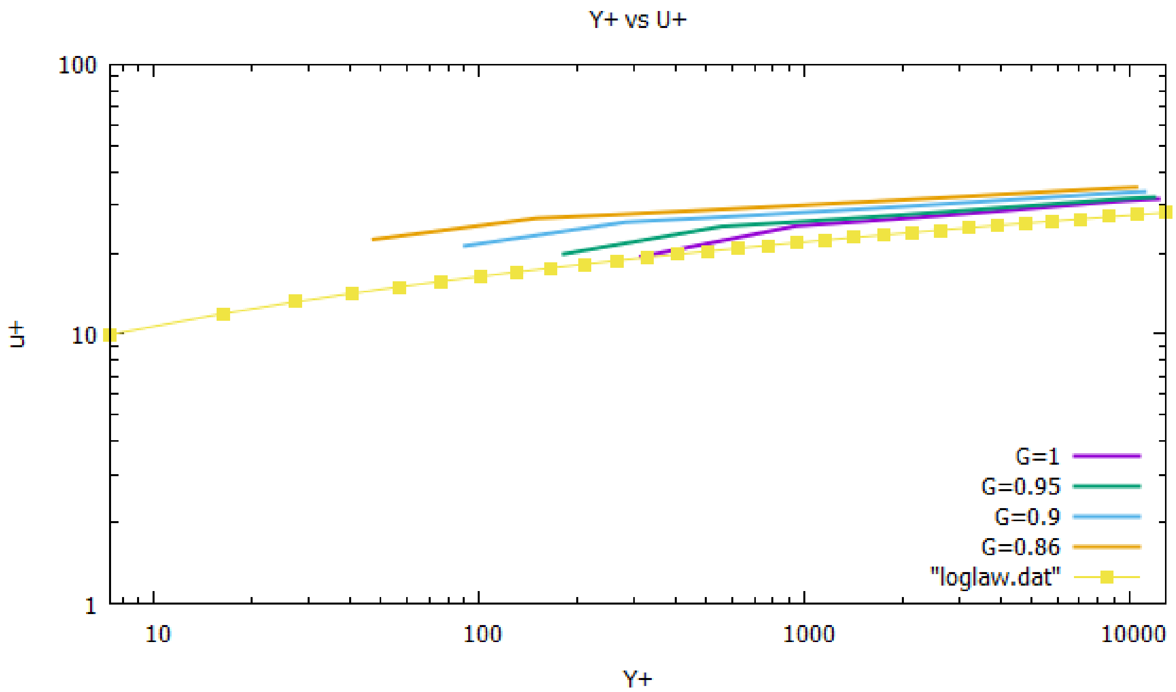


Figure 2: y_+ vs u_+ variation along grid points

Figure 2 shows the variation of u_+ with respect to y_+. As grid refinement applied towards to the wall the solution stray away from log law behavior. This may be caused by interaction with buffer layer and turbulent layer at the first grid point, since first grid is very close to the wall. On the other hand, without grid refinement the solution matches with log law, meaning that first grid point is well placed to approximate wall shear.

Grid Ratio	Y+(1)	f_c	100*|fc-fm|/fm	Iterations
0.86	47.28	8.8 10^-3	28.96	50
0.9	90.43	9.5710^-3	22.79	51
0.95	181.81	1.0510^-2	14.68	51
1	314.23	1.0910^-2	11.81	50

Table 1: Output Values

Table 1 shows the output values, after change in shear velocity in consequent iterations is small enough. Since the solution is implicit convergence achieved rapidly, and due to the algorithm to find shear velocity all three cases converged at 50 iterations.

Although all y_+ values are in range of 30 and 300. The best result is achieved with the grid ratio of 1. As first grid moves closer to the wall, percent difference between Darcy's friction factor and experimental formula becomes larger. Again the reason for that may be the interaction with buffer layer and turbulent layer.

Code Listing

```
          program PipeFlow
c Taha Yaşar Demir / 1881978
c CE-580 HomeWrok #8
          parameter(mx=100)
          common/grid/ Rad,r(mx),rc(mx),y(mx),yc(mx),dy(mx),dm(mx),N,G
          common/var/  tau_w,u(mx),vis,vist(mx),vise(mx),rho,Um,rtau(mx)
          common/const/Cp,rkp,beta,Cw,ca(mx),cd(mx),a(mx),b(mx),c(mx),d(mx)
          common/turb/ fml(mx),fu(mx),yp(mx),Ap,us,up(mx)
          common/out/  fm,fd,V_ave,Disc,Re,Pi,err,Tol,count

          call Init
          call Prior
          open(11,file="yplus.dat")
          open(12,file="output.dat")
          err =  1.
          count= 0.
          do while(err.gt.Tol)
c         do k=1,100
                    call Solution
                    count = count +1
                    call Update
                    print*, 'iteration number', count , 'Error' , err
                    call Output
          enddo
          close(11)
          close(12)

          stop
          end
c----------------------------------------------------------------------
          subroutine Init
          parameter(mx=100)
          common/grid/ Rad,r(mx),rc(mx),y(mx),yc(mx),dy(mx),dm(mx),N,G
          common/var/  tau_w,u(mx),vis,vist(mx),vise(mx),rho,Um,rtau(mx)
          common/const/Cp,rkp,beta,Cw,ca(mx),cd(mx),a(mx),b(mx),c(mx),d(mx)
          common/turb/ fml(mx),fu(mx),yp(mx),Ap,us,up(mx)
          common/out/  fm,fd,V_ave,Disc,Re,Pi,err,Tol,count

          Um  = 4.   ! m/s
          vis = 1e-6 ! m^2/s
          G   = 1.   ! Grid Ratio 1 - 0.95 - 0.9 - 0.86
          Rad = 0.1  ! m
          rho = 1000.! kg/m^3
          N   = 20   ! Grid points
          Tol = 1e-5 ! Error criteria
          Ap  = 26.
          Pi  = 22./7.
          rkp = 0.41
          beta= 5.3
```

```
      call MakeGrid
      call Analytic

      return
      end

c----------------------------------------------------------------------
      subroutine MakeGrid
      parameter(mx=100)
      common/grid/ Rad,r(mx),rc(mx),y(mx),yc(mx),dy(mx),dm(mx),N,G
   real sum

      sum = 0.0
      do i=0,N-1
            sum = sum + G**i
      enddo
      dy(N+1) = Rad/sum
      y(N+1)  = Rad

      do i=N+1,2,-1
            y(i-1)  = y(i)-dy(i)
            dy(i-1) = G*dy(i)
            yc(i-1) = (y(i)+y(i-1))/2.
            r(i)    = Rad - y(i)
            rc(i-1) = Rad - yc(i-1)
c           print*, 'grid', y(i-1),dy(i-1),yc(i-1),r(i),rc(i-1)
      enddo
      do i=2,N
            dm(i) = (dy(i)+dy(i-1))/2
      enddo
      r(1) = Rad
      y(1) = 0.

      return
      end

c----------------------------------------------------------------------
      subroutine Analytic
      parameter(mx=100)
      common/grid/ Rad,r(mx),rc(mx),y(mx),yc(mx),dy(mx),dm(mx),N,G
      common/var/  tau_w,u(mx),vis,vist(mx),vise(mx),rho,Um,rtau(mx)

      do i=1,N
        u(i) = Um*(yc(i)/Rad)**(1./7.)
      enddo


      return
      end
c----------------------------------------------------------------------
      subroutine Prior
      parameter(mx=100)
```

```
      common/grid/ Rad,r(mx),rc(mx),y(mx),yc(mx),dy(mx),dm(mx),N,G
      common/var/  tau_w,u(mx),vis,vist(mx),vise(mx),rho,Um,rtau(mx)
      common/const/Cp,rkp,beta,Cw,ca(mx),cd(mx),a(mx),b(mx),c(mx),d(mx)
      common/turb/ fml(mx),fu(mx),yp(mx),Ap,us,up(mx)
      common/out/  fm,fd,V_ave,Disc,Re,Pi,err,Tol,count

      Disc = 0.
      do i=1,N
        Disc = Disc + 2.*pi*rc(i)*u(i)*dy(i)
      enddo
      V_ave = Disc/(pi*Rad**2)
c     V_ave = 0.85*Um
      Re    = 2.*V_ave*Rad/vis
      fm    = 0.25/((log10(5.74/(Re**0.9)))**2)
      fd    = fm
      tau_w = (rho*fd*V_ave**2)/8.
      us    = sqrt(tau_w/rho)
      do i=1,N
            yp(i)   = yc(i)*us/vis
            fu(i)   = 1-exp(-yp(i)/Ap)
            fml(i)  = Rad*(0.14-0.08*(1-(yc(i)/Rad))**2
   &              -0.06*(1-(yc(i)/Rad))**4)*fu(i)
      enddo
      do i=2,N
            vist(i) = (fml(i)**2)*abs((u(i)-u(i-1))/dm(i))
            vise(i) = (vis + vist(i))
      enddo

      return
      end
c---------------------------------------------------------------------
      subroutine Solution
      parameter(mx=100)
      common/grid/ Rad,r(mx),rc(mx),y(mx),yc(mx),dy(mx),dm(mx),N,G
      common/var/  tau_w,u(mx),vis,vist(mx),vise(mx),rho,Um,rtau(mx)
      common/const/Cp,rkp,beta,Cw,ca(mx),cd(mx),a(mx),b(mx),c(mx),d(mx)
      common/turb/ fml(mx),fu(mx),yp(mx),Ap,us,up(mx)
      common/out/  fm,fd,V_ave,Disc,Re,Pi,err,Tol,count

      Cp = -2.*(tau_w/Rad)/rho
      Cw = tau_w/u(1)
      do i=1,N
            cd(i) = dy(i)*rc(i)
      enddo
      do i=2,N
            ca(i) = r(i)/dm(i)
      enddo

      rtau(N+1) = 0.
      do i=N,2,-1
            rtau(i) = rtau(i+1) - dy(i)*rc(i)*Cp
      enddo
```

```
c          u(N) = 4.
           do i=N,3,-1
                     u(i-1)= u(i) - (rtau(i)*dm(i))/(r(i)*vise(i))
           enddo
           u(1) = u(2) - (rtau(2)*dm(2))/(r(2)*vise(2))/r(2) - tau_w/rho
           print*,'u1', u(1) ,'tau_w',tau_w,'Cp',Cp, 'uN', u(N)

           return
           end

c-------------------------------------------------------------------
           subroutine Update
           parameter(mx=100)
           common/grid/ Rad,r(mx),rc(mx),y(mx),yc(mx),dy(mx),dm(mx),N,G
           common/var/  tau_w,u(mx),vis,vist(mx),vise(mx),rho,Um,rtau(mx)
           common/const/Cp,rkp,beta,Cw,ca(mx),cd(mx),a(mx),b(mx),c(mx),d(mx)
           common/turb/ fml(mx),fu(mx),yp(mx),Ap,us,up(mx)
           common/out/  fm,fd,V_ave,Disc,Re,Pi,err,Tol,count
           real law, us_old

           us_old = us
           call ustar
           tau_w = (us**2)*rho
           err = abs(us_old-us)/us_old
           print*, 'error',err,us
           do i=1,N
                     yp(i)   = yc(i)*us/vis
                     fu(i)   = 1-exp(-yp(i)/Ap)
                     fml(i)  = Rad*(0.14-0.08*(1-(yc(i)/Rad))**2
   &                    -0.06*(1-(yc(i)/Rad))**4)*fu(i)
           enddo
           do i=2,N
                     ! Take average of two consequent viscous stress to eliminate oscilation
                     vist(i) =(vist(i)+((fml(i)**2)*abs((u(i)-u(i-1))/dm(i))))/2.
c                    vist(i) =(((fml(i)**2)*abs((u(i)-u(i-1))/dm(i))))
                     vise(i) = vis + vist(i)
           enddo

           return
           end
c-------------------------------------------------------------------
           subroutine ustar
           parameter(mx=100)
           common/grid/ Rad,r(mx),rc(mx),y(mx),yc(mx),dy(mx),dm(mx),N,G
           common/var/  tau_w,u(mx),vis,vist(mx),vise(mx),rho,Um,rtau(mx)
           common/const/Cp,rkp,beta,Cw,ca(mx),cd(mx),a(mx),b(mx),c(mx),d(mx)
           common/turb/ fml(mx),fu(mx),yp(mx),Ap,us,up(mx)
           common/out/  fm,fd,V_ave,Disc,Re,Pi,err,Tol,count
           real test, yplus
           yplus = us*yc(1)/vis
           test  = us*(1/rkp)*log(yplus) + us*beta
           if(test.lt.u(1)) then
```

```
                         us   = us+ 0.005*us/(10*count)
          else

                         us = (0.995*us+us)/2
          endif

          return
          end
c-------------------------------------------------------------------
          subroutine Output
          parameter(mx=100)
          common/grid/ Rad,r(mx),rc(mx),y(mx),yc(mx),dy(mx),dm(mx),N,G
          common/var/  tau_w,u(mx),vis,vist(mx),vise(mx),rho,Um,rtau(mx)
          common/const/Cp,rkp,beta,Cw,ca(mx),cd(mx),a(mx),b(mx),c(mx),d(mx)
          common/turb/ fml(mx),fu(mx),yp(mx),Ap,us,up(mx)
          common/out/  fm,fd,V_ave,Disc,Re,Pi,err,Tol,count
          real loglaw,relative
          do i=1,N
            Disc = Disc + 2.*pi*rc(i)*u(i)*dy(i)
          enddo
          V_ave = Disc/(pi*Rad**2)
c         V_ave = 0.85*Um
          Re    = 2.*V_ave*Rad/vis
          fm    = 0.25/((log10(5.74/(Re**0.9)))**2)
          fd    = (8*tau_w)/(rho*(V_ave**2))
          if(err.lt.Tol) then
          do i=1,N
                   yp(i) = us*yc(i)/vis
                   up    = u(i)/us
                   loglaw= (1./rkp)*log(yp(i))+beta
                   write(11,*) yp(i),up(i),loglaw
          enddo
          else
          relative = 100*abs(fd-fm)/fm
          write(12,*) G,yp(1),fd,relative,count
          print*,'Ratio',' yp1',' fc',' difference',' Iteration'
          print*, G,yp(1),fd,relative,count
          endif
          return
```

Fortran Code Used for Calculations