****

**CE-580**

**COMPUTATIONAL TECHNIQUES**

**FOR**

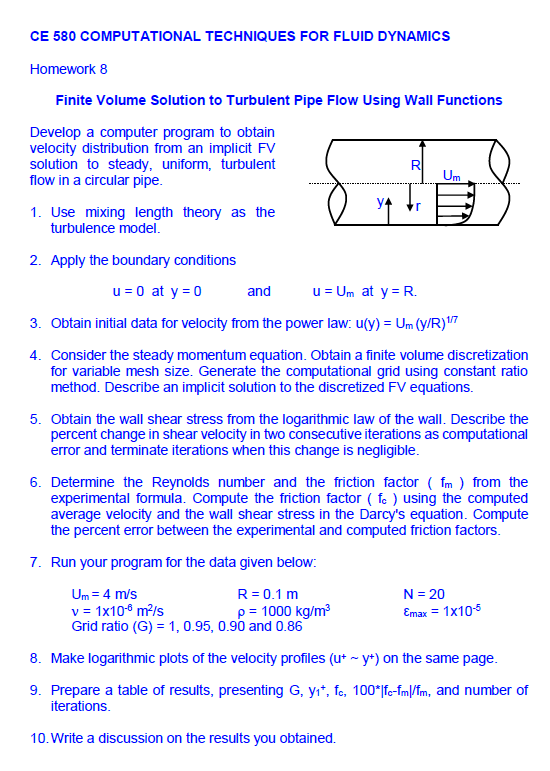
**FLUID DYNAMICS**

**HOMEWORK #8**

**Finite Volume Solution to Turbulent Pipe Flow Using Wall Functions**

**Taha Yaşar Demir**

**1881978**

****

# Calculations

***Assumptions***

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

Simplified momentum equation in radial coordinates

In terms of shear stress

Central differences used for discretization

Grid system used is

|  |  |
| --- | --- |
|  |  |
| Figure 1: Grid System and Domain | |

Using central differences for control points

Defining new parameters for constants

The tridiagonal form is

And coefficients A, B, C, D are

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

Where

Instead of calculating wall shear, 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.

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

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

Distance between the last two mesh point can be calculated as

Mesh distribution can be completed marching down from N to 1

***Boundary Conditions***

At (i=1) wall the momentum equation is

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

Then the momentum balance for the first cell is

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

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

At the centerline (i=N) momentum balance is

In tridiagonal form

Where coefficients A, B, C, D are

***Initial condition***

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

***Error Calculation***

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

***Outputs***

Required outputs can be calculated once the solution is converged

Discharge

Above integral can be taken numerically as such

Average velocity

Reynolds Number

Friction factors

From experimental data, using Swamee-Jain formula

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

From the converged solution, using Darcy’s friction factor

***Input Data***

# Results and Discussion

Using wall functions saves us from grid refinement on solid boundaries. However, there is still a restriction on 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 value is in the valid range, wall shear can be estimated and using wall shear pressure drop can be calculated.

|  |
| --- |
|  |
| Figure 2: variation along grid points |

Figure 2 shows the variation of with respect to . 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 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 |