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 4

Submission Date	July 14, 2021
Programing Language Used	Fortran90

Student Name	Student Number
Ezeorah Godswill	501012886

Q1a| From the given problem, we can assume that the flow is incompressible (ie. $\rho = \text{const.}$), and that the flow is viscid (since $\mu = 1.84e^{-5}$ kg/ms)

So that the vector eqn for the conservation of momentum (c.o.mom) is,

$$\frac{\partial}{\partial t}(\rho \vec{u_r}) + \nabla(\rho \hat{u}_r \vec{u}_r) = -\nabla P + \nabla \cdot \tau$$

where $\tau \rightarrow$ represents the viscous stress tensor

$U_r \rightarrow$ represents the radial velocity

Since we assume $\rho = \text{const.}$, the continuity eqn will be

$$\nabla \cdot (\rho \vec{u_r}) = 0$$

This means we don't need pressure corrector to match the condition of our velocity in the above continuity eqn. So that the divergence in 1D, $\nabla \cdot \tau = \frac{1}{r}\frac{\partial}{\partial r}(r\tau)$ for cylindrical coord. and according to newton's law of viscosity, ~~(1D)~~

$$\tau = \tau_{rr}$$

where $\tau_{rr} = \mu \frac{\partial U_r}{\partial r}$ (for incompressible 1D flow)

Q1 a| cont'd.

we also have the gradient

$$\nabla P = \frac{\partial P}{\partial r}$$

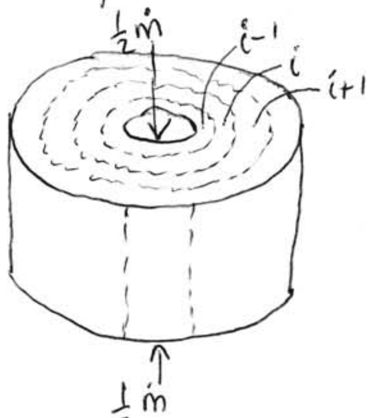
So that we have the Navier-Stokes eqn as c.o.mom.

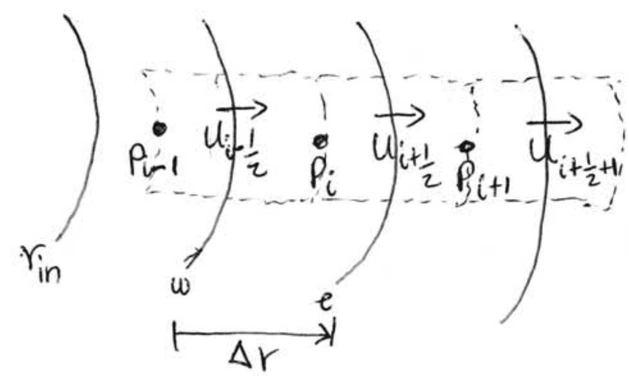
$$\rho \frac{\partial u_r}{\partial t} + \rho u_r \frac{\partial u_r}{\partial r} = -\frac{\partial P}{\partial r} + \mu \frac{1}{r}\frac{\partial}{\partial r}\left(r \frac{\partial u_r}{\partial r}\right) \qquad (1.1)$$

and the continuity eqn will be

$$\rho \frac{1}{r}\frac{\partial (r u_r)}{\partial r} = 0 \qquad (1.2)$$

if we consider the following staggered grid





So that integrating eqn (1.2) from the shifted grid $i-\frac{1}{2}$ to $i+\frac{1}{2}$

$$\int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \frac{1}{\cancel{r}} \rho \cancel{r} u_r \, dA_r = \cancel{\rho} u_{i+\frac{1}{2}} A_{i+\frac{1}{2}} - \cancel{\rho} u_{i-\frac{1}{2}} A_{i-\frac{1}{2}} = 0$$

This give $u_{i+\frac{1}{2}} A_{i+\frac{1}{2}} = u_{i-\frac{1}{2}} A_{i-\frac{1}{2}}$

Q1a) cont'd.;

Similarly let's integrate the c.o.mom. of eqn (1.1) with the grid from $i-\frac{1}{2}$ to $i+\frac{1}{2}$ on the LHS and from i to $i+1$ on the RHS. ~~as,~~ And considering simple explicit scheme for the time discretization, as,

$$\int_{\forall}\int_{n}^{n+1} \rho \frac{\partial u}{\partial t}\,\partial t\,\partial\forall + \int_{n}^{n+1}\int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \rho u_r \frac{\partial u_r}{\partial r}\,\partial\forall = -\int_{n}^{n+1}\int_{i}^{i+1} \frac{\partial P}{\partial r}\,\partial V + \int_{n}^{n+1}\int_{i}^{i+1} \mu \frac{1}{r}\frac{\partial}{\partial r}\left(\frac{r\,\partial u_r}{\partial r}\right)\partial\forall$$

This yields,

$$\left(\rho u^{n+1}_{i+\frac{1}{2}} - \rho u^{n}_{i+\frac{1}{2}}\right)\Delta V + \left(\rho u^{2}_{i+\frac{1}{2}} A_{i+\frac{1}{2}} - \rho u^{2}_{i-\frac{1}{2}} A_{i-\frac{1}{2}}\right)\Big|_{n} \Delta t =$$

$$-\left(P_{i+1} - P_i\right)\Big|_{n+1} A_{i+\frac{1}{2}}\,\Delta t + \frac{\mu \cancel{r}}{\Delta r \cancel{r}}\left(u_{i+\frac{1}{2}+1} - u_{i+\frac{1}{2}}\right)\Big|_{t} A_{i+1}\,\Delta t$$

$$-\frac{\mu \cancel{r}}{\Delta r \cancel{r}}\left(u_{i+\frac{1}{2}} - u_{i-\frac{1}{2}}\right)\Big|_{t} A_i\,\Delta t$$

Simplifying ~~gives,~~ and re-arranging gives.

$$P_i^{n+1} = \frac{\rho}{\Delta t A_{i+\frac{1}{2}}}\left(u^{n+1}_{i+\frac{1}{2}} - u^{n}_{i+\frac{1}{2}}\right)\forall_{i+\frac{1}{2}} + \frac{\rho}{A_{i+\frac{1}{2}}}\left(u^{2}_{i+\frac{1}{2}} A_{i+\frac{1}{2}} - u^{2\,n}_{i-\frac{1}{2}} A_{i-\frac{1}{2}}\right)$$

$$-\frac{\mu A_{i+1}}{\Delta r A_{i+\frac{1}{2}}}\left(u^{n}_{i+\frac{1}{2}+1} - u^{n}_{i+\frac{1}{2}}\right) + \frac{\mu A_i}{\Delta r A_{i+\frac{1}{2}}}\left(u^{n}_{i+\frac{1}{2}} - u^{n}_{i-\frac{1}{2}}\right) + P^{n+1}_{i+1}$$

Q1a| cont'd.!

From our above discretization we can put,

the shifted c.v., $\forall_{i+\frac{1}{2}} = A_{i+\frac{1}{2}} \Delta r$

with shifted cylindrical area/face ~~in the~~ $\perp$ar to the flow direction, $A_{i+\frac{1}{2}} = 2\pi r_{i+\frac{1}{2}} h$

where the heigth, $h = 0.5m$

So that $\forall_{i+\frac{1}{2}} = 2\pi r_{i+\frac{1}{2}} h \Delta r$

Substituting these into our continuity part, we have,

$$\boxed{u_{i+\frac{1}{2}} = \frac{u_{i-\frac{1}{2}} r_{i-\frac{1}{2}}}{r_{i+\frac{1}{2}}}} \quad (1.3)$$

and for the c.o.MOM. part, wehave,

$$\boxed{P_i^{n+1} = \frac{\rho \Delta r}{\Delta t}\left(u_{i+\frac{1}{2}}^{n+1} - u_{i+\frac{1}{2}}^{n}\right) + \frac{\rho}{r_{i+\frac{1}{2}}}\left(u^{2\,n}_{i+\frac{1}{2}} r_{i+\frac{1}{2}} - u^{2\,n}_{i-\frac{1}{2}} r_{i-\frac{1}{2}}\right) - \frac{\mu r_{i+1}}{r_{i+\frac{1}{2}}\Delta r}\left(u^{n}_{i+\frac{1}{2}+1} - u^{n}_{i+\frac{1}{2}}\right) + \frac{\mu r_i}{r_{i+\frac{1}{2}}\Delta r}\left(u^{n}_{i+\frac{1}{2}} - u^{n}_{i-\frac{1}{2}}\right) + P_{i+1}^{n+1}} \quad (1.4)$$

>> Now applying the initial & Boundary conditions, we should start by first solving eqn(1.3),

Q1a] cont'd.:

by assuming at ~~ghost~~ the first point $i=1$, the eqn becomes

$$\boxed{u_{1+\frac{1}{2}} = \frac{u_{in} r_{in}}{r_{1+\frac{1}{2}}}}$$

And we are given the volumetric flow rate,

$$\dot{m}_{in} = u_{in} A_{in} = u_{in} 2\pi r_{in} h = 10^{-5}(e^{2t}-1)$$

so that $\quad u_{in} r_{in} = \frac{\dot{m}}{2\pi h}$ $\qquad \boxed{u_{in} = \frac{\dot{m}}{2\pi h r_{in}}}$

This gives us,

$$\boxed{u_{1+\frac{1}{2}} = \frac{\dot{m}}{2\pi h r_{1+\frac{1}{2}}}}$$

Since this is a function of time, we can say that at an initial time, $t=0$, the radial velocity, $u_r^{n=0} = 0$ (i.e. no flow). So that we can increase the time by $t=t+\Delta t$ and calculate u_r^{n+1}, where $\Delta t = 0.01$ sec.

Hence with the calculated velocity vector, we can use eqn (1.4) to calculate the pressure, by starting from the last grid point $(i=n)$, since we are given $P_{out} = 101325\ N/m^2$.

Q1a) cont'd:

And if we assume that P_{out} is independent of time, so that at the last grid point we have,

$$\boxed{P_n^{n+1} = \frac{\rho \Delta r}{\Delta t}\left(u^{n+1}_{n+\frac{1}{2}} - u^{n}_{n+\frac{1}{2}}\right) + \frac{\rho}{r_{n+\frac{1}{2}}}\left(u^{2^n}_{n+\frac{1}{2}} r_{n+\frac{1}{2}} - u^{2^n}_{n-\frac{1}{2}} r_{n-\frac{1}{2}}\right) - \frac{\mu r_{n+1}}{\Delta r\, r_{n+\frac{1}{2}}}\cancelto{0}{\left(u^{n}_{n+\frac{1}{2}} - u^{n}_{n+\frac{1}{2}}\right)} + \frac{\mu r_n}{\Delta r\, r_{n+\frac{1}{2}}}\left(u^{n}_{n+\frac{1}{2}} - u^{n}_{n-\frac{1}{2}}\right) + P_{out}}$$

Similarly at the first grid point we will have the velocity terms as,

$$\boxed{P_1^{n+1} = \frac{\rho \Delta r}{\Delta t}\left(u^{n+1}_{1+\frac{1}{2}} - u^{n}_{1+\frac{1}{2}}\right) + \frac{\rho}{r_{1+\frac{1}{2}}}\left(u^{2^n}_{1+\frac{1}{2}} r_{1+\frac{1}{2}} - u^{2^n}_{in} r_{in}\right) - \frac{\mu r_{1+1}}{\Delta r\, r_{1+\frac{1}{2}}}\left(u^{n}_{1+\frac{1}{2}+1} - u^{n}_{1+\frac{1}{2}}\right) + \frac{\mu r_1}{\Delta r\, r_{1+\frac{1}{2}}}\left(u^{n}_{1+\frac{1}{2}} - u^{n}_{in}\right) + P_{1+1}^{n+1}}$$

>> Checking the consistency of units of eqn (1.4), we have

$$\frac{kg}{ms^2} = \frac{kg\, m\, m}{m^3 s\, m\, s} + \frac{kg\, m\, m}{m^3 s\, m\, s} + \frac{kg\, m^2/m}{m^3\, m\, s^2} - \frac{kg\, m^2 m}{m^3 m s^2} - \frac{kg\, mm}{m s m m s} + \frac{kg\, mm}{m s m m s} + \frac{kg\, mm}{m s m m s} - \frac{kg\, mm}{m s m m s} + \frac{kg}{ms^2}$$

so that $\frac{kg}{ms^2} = \frac{kg}{ms^2}$ true

∴ our discretized formulation is consistent in its units.

These formulation in boxes are used in our program, using fortran90.

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

!This program was written by Godswill Ezeorah, Student Number: 501012886 on July 07, 2021.
!This program solves a Navier-Stokes equation using finite volume method
!and was written as a solution to AE8112 PS4 q1b
!****************************End Header*****************************************
program Navier_Stokes
    implicit none
    DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: ri, P, U, Uo
    DOUBLE PRECISION, PARAMETER :: h=0.5, rin=0.01, rout=0.11, Pout=0, Patm=101325
    DOUBLE PRECISION, PARAMETER :: rho=1.2, mu=1.84D-5, Pi=4*atan(1.0)
    DOUBLE PRECISION :: mdot, t, dt, dr, r, rp, rn, aa, ab, ac, ad, u1, uo1
    DOUBLE PRECISION :: rnh, rph
    INTEGER :: n, i
    open(1, file = 'PS4_Q2b_withMu.txt', status = 'unknown')
    !Variable initialization and definition
    t=0.01
    n=100 !Nuumber of control volumes
    ALLOCATE(ri(n), P(n), U(n), Uo(n))
    dr = (rout-rin)/n
    dt = 0.01  !time step
    Uo = 0

    do while (t <= 1)
        t = t+dt
        r=rin
        mdot = (10D-5)*(exp(2*t)-1)
        do i = 1, n !This loop solves velocity using our contuinity formulation
            rp = r+dr
            if (i==1) then  !for first grid point
                U(i)=mdot/(2*pi*h*rp)
            else            !for all other grid points
                U(i)=U(i-1)*r/rp
            end if
            ri(i)=r
            r=r+dr
        end do
        r=rout
        do i = n, 1, -1 !This loop solves pressure backward using our momentum formulation
            rn = r-dr
            rnh = r-dr/2
            rph = r+dr/2
            aa = (rho*dr)*(U(i)-Uo(i))/dt
            ab = rho*(r*Uo(i)**2-rn*Uo(i-1)**2)/r
            ac = mu*rph*(Uo(i+1)-Uo(i))/(dr*r)
```

```fortran
            ad = mu*rnh*(Uo(i)-Uo(i-1))/(dr*r)
            if ( i == n ) then     !for last grid point
                ac = mu*(Uo(i)-Uo(i))/dr
                P(i) = aa+ab-ac+ad+Pout
            elseif ( i >= 2 ) then !for all other grid points
                P(i) = aa+ab-ac+ad+P(i+1)
            else                   !for first grid point
                u1=(10D-5)*(exp(2*t)-1)/(2*pi*h*rin)
                uo1=(10D-5)*(exp(2*(t-dt))-1)/(2*pi*h*rin)
                aa = (rho*dr)*(U(i)-Uo(i))/dt
                ab = rho*(r*Uo(i)**2-rin*Uo1**2)/r
                ac = mu*rph*(Uo(i+1)-Uo(i))/(dr*r)
                ad = mu*r*(Uo(i)-Uo1)/(dr*r)
                P(i) = aa+ab-ac+ad+P(i+1)
            end if
            r=r-dr
        end do
        Uo = U
    end do

    !Printing results
    do i = 1, n
        write(1,*) ri(i), P(i)+Patm
    end do
    print *, 'Velo. Diff',U(1)-U(n)
    print *, 'press. Diff',P(1)-P(n)
    write(*,2) 'Radius =', ri
    write(*,1) '(Pressure – Patm) =', P
    write(*,2) 'Velocity =', U
    1 format(a20,100E10.3)
    2 format(a11,100f7.3)
    close (1)
end program Navier_Stokes
```

Radius = 0.010 0.011 0.012 0.013 0.014 0.015 0.016 0.017 0.018 0.019 0.020 0.021 0.022 0.023 0.024 0.025 0.026 0.027 0.028 0.029 0.030 0.031 0.032 0.033 0.034 0.035 0.036 0.037 0.038 0.039 0.040 0.041 0.042 0.043 0.044 0.045 0.046 0.047 0.048 0.049 0.050 0.051 0.052 0.053 0.054 0.055 0.056 0.057 0.058 0.059 0.060 0.061 0.062 0.063 0.064 0.065 0.066 0.067 0.068 0.069 0.070 0.071 0.072 0.073 0.074 0.075 0.076 0.077 0.078 0.079 0.080 0.081 0.082 0.083 0.084 0.085 0.086 0.087 0.088 0.089 0.090 0.091 0.092 0.093 0.094 0.095 0.096 0.097 0.098 0.099 0.100 0.101 0.102 0.103 0.104 0.105 0.106 0.107 0.108 0.109

(Pressure – Patm) = 0.109E-02 0.108E-02 0.106E-02 0.105E-02 0.103E-02 0.101E-02 0.984E-03 0.962E-03 0.940E-03 0.918E-03 0.897E-03 0.876E-03 0.855E-03 0.835E-03 0.815E-03 0.796E-03 0.777E-03 0.759E-03 0.741E-03 0.724E-03 0.707E-03 0.690E-03 0.674E-03 0.658E-03 0.643E-03 0.628E-03 0.613E-03 0.599E-03 0.585E-03 0.571E-03 0.558E-03 0.545E-03 0.532E-03 0.519E-03 0.507E-03 0.495E-03 0.483E-03 0.471E-03 0.460E-03 0.449E-03 0.438E-03 0.427E-03 0.416E-03 0.406E-03 0.396E-03 0.386E-03 0.376E-03 0.366E-03 0.357E-03 0.347E-03 0.338E-03 0.329E-03 0.320E-03 0.311E-03 0.302E-03 0.294E-03 0.285E-03 0.277E-03 0.269E-03 0.261E-03 0.253E-03 0.245E-03 0.237E-03 0.229E-03 0.222E-03 0.214E-03 0.207E-03 0.200E-03 0.192E-03 0.185E-03 0.178E-03 0.171E-03 0.165E-03 0.158E-03 0.151E-03 0.144E-03 0.138E-03 0.131E-03 0.125E-03 0.119E-03 0.112E-03 0.106E-03 0.100E-03 0.941E-04 0.881E-04 0.821E-04 0.762E-04 0.704E-04 0.647E-04 0.590E-04 0.533E-04 0.477E-04 0.422E-04 0.367E-04 0.313E-04 0.259E-04 0.205E-04 0.153E-04 0.100E-04 0.484E-05

Velocity = 0.019 0.017 0.016 0.015 0.014 0.013 0.012 0.012 0.011 0.010 0.010 0.009 0.009 0.009 0.008 0.008 0.008 0.007 0.007 0.007 0.007 0.007 0.006 0.006 0.006 0.006 0.006 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.004 0.004 0.004 0.004 0.004 0.004 0.004 0.004 0.004 0.004 0.004 0.004 0.004 0.003 0.002

Q2a I wish to test the grid independence of my previous Solution. I will continually double my grid, starting from 8 points, until ~~any~~ a tolerance is reached. This tolerance is set to

$$Tol = 1e^{-4}$$

And the condition for this tolerance to be met is based of L2 error of Pressure at the first grid point (P_1).

That is

$$\frac{\sqrt{\sum_{N=8}^{N+N} P_1(N)}}{J} < Tol$$

where J, is the number of consider/summed ~~gridpoints.~~

P_1 at each consider grid point set, before the tolerance is met.

>> The below code is ran ~~to~~ t = 1 sec, and at this time the table below is extracted, and used to plot the error as a function of grid points.

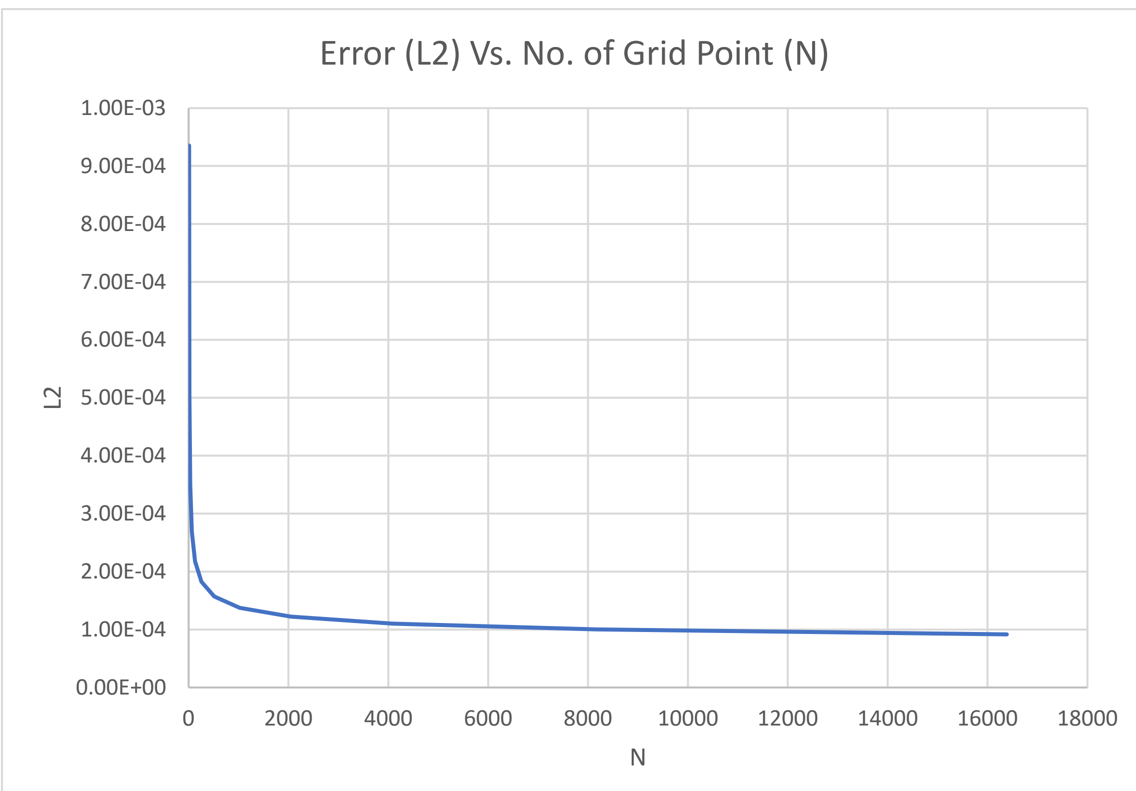
From the plot below, I observe that the solution doesn't change that much after 2048 grid points.

∴ Therefore the optimal number of grid points for this problem is 2048.

```fortran
!***************Begin Header****************************************************
!This program was written by Godswill Ezeorah, Student Number: 501012886 on July 07, 2021.
!This program solves a Navier-Stokes equation using finite volume method
!and was written as a solution to AE8112 PS4 q2
!****************************End Header*****************************************
program Navier_Stokes
    implicit none
    DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: ri, P, U, Uo, Po
    DOUBLE PRECISION, PARAMETER :: h=0.5, rin=0.01, rout=0.11, Pout=0
    DOUBLE PRECISION, PARAMETER :: rho=1.2, mu=1.84D-5, Pi=4*atan(1.0)
    DOUBLE PRECISION :: mdot, t, dt, dr, r, rp, rn, aa, ab, ac, ad, u1, uo1
    DOUBLE PRECISION :: rph, rnh, L2, Tol
    INTEGER :: n, i, j
    open(1, file = 'PS4_Q2a.txt', status = 'unknown')
    !Variable initialization and definition
    n=8
    L2=1
    Tol=1D-4
    j=1
    !The below loop increases the number of gridpoint until tolerance is met.
    do while (L2>Tol)
        t=0.01
        ALLOCATE(ri(n), P(n), U(n), Uo(n), Po(j))
        dr = (rout-rin)/n
        dt = 0.01  !time step
        Uo = 0
        do while (t <= 1)
            t = t+dt
            r=rin
            mdot = (10D-5)*(exp(2*t)-1)
            do i = 1, n !This loop solves velocity using our contuinity formulation
                rp = r+dr
                if (i==1) then !for first grid point
                    U(i)=mdot/(2*pi*h*rp)
                else           !for all other grid points
                    U(i)=U(i-1)*r/rp
                end if
                ri(i)=r
                r=r+dr
            end do
            r=rout
            do i = n, 1, -1 !This loop solves pressure backward using our momentum formulation
                rn = r-dr
                rnh = r-dr/2
                rph = r+dr/2
```

```
            aa = (rho*dr)*(U(i)-Uo(i))/dt
            ab = rho*(r*Uo(i)**2-rn*Uo(i-1)**2)/r
            ac = mu*rph*(Uo(i+1)-Uo(i))/(dr*r)
            ad = mu*rnh*(Uo(i)-Uo(i-1))/(dr*r)
            if ( i == n ) then !for last grid point
                ac = mu*(Uo(i)-Uo(i))/dr
                P(i) = aa+ab-ac+ad+Pout
            elseif ( i >= 2 ) then !for all other grid points
                P(i) = aa+ab-ac+ad+P(i+1)
            else                   !for first grid point
                u1=(10D-5)*(exp(2*t)-1)/(2*pi*h*rin)
                uo1=(10D-5)*(exp(2*(t-dt))-1)/(2*pi*h*rin)
                aa = (rho*dr)*(U(i)-Uo(i))/dt
                ab = rho*(r*Uo(i)**2-rin*Uo1**2)/r
                ac = mu*rph*(Uo(i+1)-Uo(i))/(dr*r)
                ad = mu*r*(Uo(i)-Uo1)/(dr*r)
                P(i) = aa+ab-ac+ad+P(i+1)
            end if
            r=r-dr
        end do
        Uo = U
    end do
    Po(j)=P(1)
    L2=norm2(Po)/j
    print *, 'No. of gridpoints.',n
    print *, 'L2 Error',L2
    write(1,*) n, L2
    if ( L2>Tol ) then
        DEALLOCATE(ri, P, U, Uo, Po)
        n=n+n
        j=j+1
    end if
  end do
  close (1)
end program Navier_Stokes
```

No. of Grid Point (N)	Error (L2)
8	9.35E-04
16	5.05E-04
32	3.51E-04
64	2.69E-04
128	2.18E-04
256	1.83E-04
512	1.57E-04
1024	1.38E-04
2048	1.22E-04
4096	1.10E-04
8192	1.00E-04
16384	9.18E-05



Q 2 b| ~~cont'd.~~!

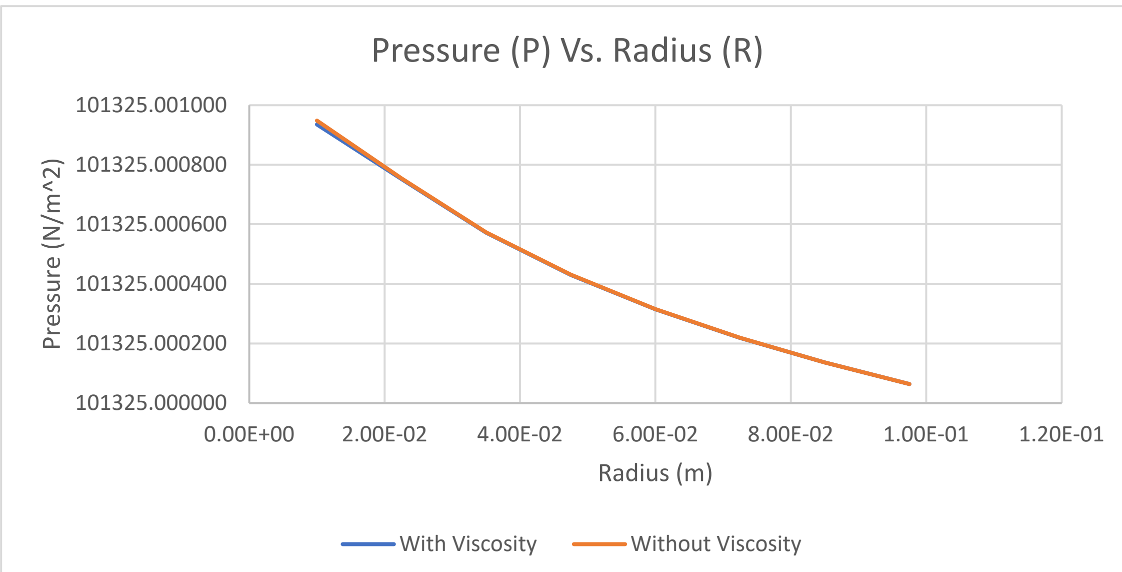
In addition, I also considered the effect of neglecting the viscous term, to our solution.

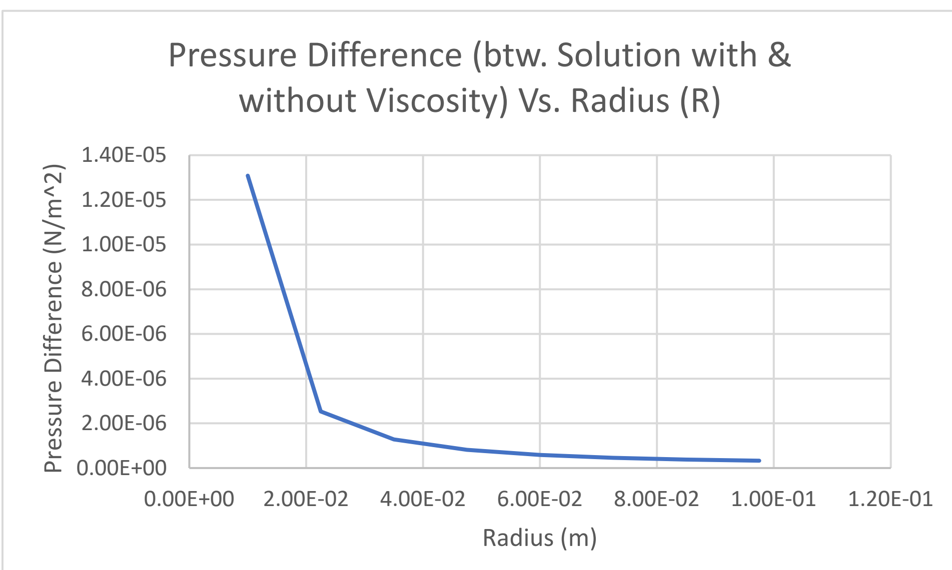
From table below, we can see that the pressure solution withour viscosity ~~is very~~ has a very very small increase from the solution with viscosity. ~~these~~

This is true as can be observed from the Navier-Stokes eqn (c.o.mom.). But the increase is very negligible, especially towards the outer radius, as observed from the pressure difference (between with & without viscosity) plot below, as a function of radius for 8 gridpoint.

∴ Therefore from the observation of the solution plots between→with & without viscosity, we can say that viscosity has negligible effect on our solution.

With Viscosity		Without Viscosity	
Radius(m)	Pressure (N/m^2)	Radius (m)	Pressure (N/m^2)
1.00E-02	101325.000935	1.00E-02	101325.000948
2.25E-02	101325.000752	2.25E-02	101325.000754
3.50E-02	101325.000572	3.50E-02	101325.000573
4.75E-02	101325.000430	4.75E-02	101325.000431
6.00E-02	101325.000315	6.00E-02	101325.000316
7.25E-02	101325.000219	7.25E-02	101325.000219
8.50E-02	101325.000136	8.50E-02	101325.000137
9.75E-02	101325.000064	9.75E-02	101325.000064





Q2c) Finally I also consider solving problem 1, using finite difference method, described below; we have our c.o.mom eqn as (from eqn (1.1)),

$$\rho \frac{\partial u_r}{\partial t} + \rho u_r \frac{\partial u_r}{\partial r} = -\frac{\partial P}{\partial r} + \mu \frac{1}{r}\frac{\partial}{\partial r}\left(r \frac{\partial u}{\partial r}\right)$$

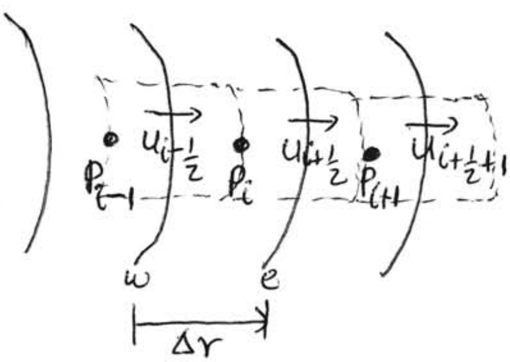
and the continuity eqn as, (from eqn (1.2)),

$$\rho \frac{1}{r}\left(\frac{\partial (r u_r)}{\partial r}\right) = 0$$

We know that from Tylor series expansion, we have the first and second order derivative for equispaced staggered grid (assuming from $u_{i+\frac{1}{2}}$ to $u_{i+\frac{1}{2}}$), we have

$$\frac{\partial u_r}{\partial r} = \frac{u_{i+\frac{1}{2}} - u_{i-\frac{1}{2}}}{2\Delta r}$$

and $$\frac{\partial^2 u_r}{\partial r} = \frac{u_{i-\frac{1}{2}} - 2u_i + u_{i+\frac{1}{2}}}{\Delta r^2}$$



Applying the derivative product rule to the continuity, we have,

$$\rho \frac{1}{r}\frac{\cancel{\partial r}}{\cancel{\partial r}} u_r + \rho \frac{1}{\cancel{r}} \cancel{r} \frac{\partial u_r}{\partial r}$$

and applying the Tylor series derived eqn, we have, Q2c) cont'd.)

$$\frac{\rho u_i}{r_i} + \frac{\rho}{2\Delta r}\left(u_{i+\frac{1}{2}} - u_{i-\frac{1}{2}}\right) = 0$$

let $u_i = \dfrac{u_{i+\frac{1}{2}} + u_{i-\frac{1}{2}}}{2}$ assuming linear profile for our velocity. we get

$$\frac{\rho}{2r_i} u_{i+\frac{1}{2}} + \frac{\rho}{2r_i} u_{i-\frac{1}{2}} + \frac{\rho}{2\Delta r} u_{i+\frac{1}{2}} - \frac{\rho u_{i-\frac{1}{2}}}{2\Delta r} = 0$$

multiplying by $\frac{2}{\rho}$, we get (and re-arranging)

$$\boxed{u_{i+\frac{1}{2}} = \frac{u_{i-\frac{1}{2}}\left(\frac{1}{\Delta r} - \frac{1}{r_i}\right)}{\left(\frac{1}{r_i} + \frac{1}{\Delta r}\right)}}$$

Now apply the derivative product rule to our momentum eq$\underline{n}$, we have

$$\rho \frac{\partial u_r}{\partial t} + \rho u_r \frac{\partial u_r}{\partial r} = -\frac{\partial P}{\partial r} + \frac{\mu}{r}\frac{\cancel{\partial r}}{\cancel{\partial r}}\frac{\partial u}{\partial r} + \frac{\mu}{\cancel{r}}\cancel{r}\frac{\partial^2 u}{\partial r^2}$$

we have

$$\rho \frac{\partial u_r}{\partial t} + \rho u_r \frac{\partial u_r}{\partial r} = -\frac{\partial P}{\partial r} + \frac{\mu}{r}\frac{\partial u_r}{\partial r} + \mu \frac{\partial^2 u_r}{\partial r^2}$$

Now lets apply the standard forward diff,

$$\frac{\partial u_r}{\partial t} = \frac{u_i^{n+1} - u_i^n}{\Delta t}$$

and the Tylor series derived eq$\underline{n}$, we have, Q2C| Cont'd.!

$$\frac{\rho}{\Delta t}\left(u_i^{n+1}-u_i^{n}\right)+\frac{\rho u_i}{2\Delta r}\left(u_{i+\frac{1}{2}}-u_{i-\frac{1}{2}}\right)=-\frac{\left(P_{i+1}-P_i\right)}{2\Delta r}+\frac{\mu}{2r_i\Delta r}\left(u_{i+\frac{1}{2}}-u_{i-\frac{1}{2}}\right)$$

$$+\frac{\mu}{\Delta r^2}\left(u_{i-\frac{1}{2}}-2u_i+u_{i+\frac{1}{2}}\right)$$

multiplying by $2\Delta r$ and re-arranging, we get, (assuming explicit time discretization). ~~we~~

$$\boxed{P_i^{n+1}=\frac{2\Delta r\rho}{\Delta t}\left(u_i^{n+1}-u_i^{n}\right)+\rho u_i^n\left(u^n_{i+\frac{1}{2}}-u^n_{i-\frac{1}{2}}\right)-\frac{\mu}{r_i}\left(u^n_{i+\frac{1}{2}}-u^n_{i-\frac{1}{2}}\right)}$$

$$\boxed{-\frac{2\mu}{\Delta r}\left(u^n_{i-\frac{1}{2}}-2u_i^n+u^n_{i+\frac{1}{2}}\right)+P_{i+1}^{n+1}}$$

>> Applying B.Cs for the first grid point $(i=1)$ to the continuity formulation, we have

$$u_{1+\frac{1}{2}}=\frac{u_{in}\left(\frac{1}{\Delta r}-\frac{1}{r_1}\right)}{\left(\frac{1}{r_1}+\frac{1}{\Delta r}\right)}$$

and we know $u_{in}=\dfrac{\dot{m}}{2\pi h r_0}$ at some ghost point

So that we have

$$\boxed{u_{1+\frac{1}{2}}=\frac{\dot{m}\left(\frac{1}{\Delta r}-\frac{1}{r_1}\right)}{2\pi h r_0\left(\frac{1}{r_1}+\frac{1}{\Delta r}\right)}}$$

>> And for the momentum formulation we have that at the last grid point $(i=n)$

Q2C| cont'd.:

$$\boxed{P_n^{n+1} = \frac{2\Delta r \rho}{\Delta t}\left(u_n^{n+1} - u_n^n\right) + \rho u_n^n\left(u_{n+\frac{1}{2}}^n - u_{n-\frac{1}{2}}^n\right) - \frac{\mu}{r_n}\left(u_{n+\frac{1}{2}}^n - u_{n-\frac{1}{2}}^n\right) - \frac{2\mu}{\Delta r}\left(u_{n-\frac{1}{2}}^n - 2u_n^n + u_{n+\frac{1}{2}}^n\right) + P_{out}}$$

also at the first grid point $(i=1)$ we will have the velocity terms as

$$\boxed{P_1^{n+1} = \frac{2\Delta r \rho}{\Delta t}\left(u_1^{n+1} - u_1^n\right) + \rho u_1^n\left(u_{1+\frac{1}{2}}^n - u_{in}^n\right) - \frac{\mu}{r_1}\left(u_{1+\frac{1}{2}}^n - u_{in}^n\right) - \frac{2\mu}{\Delta r}\left(u_{in}^n - 2u_1^n + u_{1+\frac{1}{2}}^n\right) + P_{1+1}^{n+1}}$$

These eqns in boxes are used in our fortran90 code and the result is shown below;

>> From the below plot comparison between the solution gotten using finite volume and finite difference method, one can hypothetically guess that finite ~~volume~~ difference method is more suited for this problem. But this needs ~~further~~ verification using an exact analytical solution, and comparing the error norms between the two methods.

```
!****************Begin Header*****************************************************
!This program was written by Godswill Ezeorah, Student Number: 501012886 on July 07, 2021.
!This program solves a Navier-Stokes equation using finite difference method
!and was written as a solution to AE8112 PS4 q2
!****************************End Header*******************************************
program Navier_Stokes
    implicit none
    DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: ri, P, U, Uo
    DOUBLE PRECISION, PARAMETER :: h=0.5, rin=0.01, rout=0.11, Pout=0, Patm=101325
    DOUBLE PRECISION, PARAMETER :: rho=1.2, mu=1.84D-5, Pi=4*atan(1.0)
    DOUBLE PRECISION :: mdot, t, dt, dr, r, rp, rn, aa, ab, ac, ad, u1, uo1
    DOUBLE PRECISION :: rh
    INTEGER :: n, i
    open(1, file = 'PS4_Q2c_withMu.txt', status = 'unknown')
    !Variable initialization and definition
    t=0.01
    n=100 !Nuumber of control volumes
    ALLOCATE(ri(n), P(n), U(n), Uo(n))
    dr = (rout-rin)/n
    dt = 0.01  !time step
    Uo = 0

    do while (t <= 1)
        t = t+dt
        r=rin
        mdot = (10D-5)*(exp(2*t)-1)
        do i = 1, n !This loop solves velocity using our Contuinity formulation
            rp = r+dr
            rh = r+dr/2
            if (i==1) then   !for first grid point
                U(i)=mdot*(1/dr-1/rh)/((2*pi*h*(rin-dr/2))*(1/rh+1/dr))
            else             !for all other grid points
                U(i)=U(i-1)*(1/dr-1/rh)/(1/rh+1/dr)
            end if
            ri(i)=r
            r=r+dr
        end do
        r=rout
        do i = n, 1, -1 !This loop solves pressure backward using our momentum formulation
            rn = r-dr
            rh = r-dr/2
            aa = (2*rho*dr)*((U(i)+U(i-1))/2-(Uo(i)+Uo(i-1))/2)/dt
            ab = rho*((Uo(i)+Uo(i-1))/2)*(Uo(i)-Uo(i-1))
            ac = mu*(Uo(i)/r-Uo(i-1)/rn)
```

```
            ad = 2*mu*(Uo(i-1)-(Uo(i)+Uo(i-1))+Uo(i))/dr
            if ( i == n ) then      !for last grid point
                P(i) = aa+ab-ac-ad+Pout
            elseif ( i >= 2) then  !for all other grid points
                P(i) = aa+ab-ac-ad+P(i+1)
            else                   !for first grid point
                u1=(10D-5)*(exp(2*t)-1)*(2*pi*h*(rin-dr/2))
                uo1=(10D-5)*(exp(2*(t-dt))-1)*(2*pi*h*(rin-dr/2))
                aa = (2*rho*dr)*((U(i)+U1)/2-(Uo(i)+Uo1)/2)/dt
                ab = rho*((Uo(i)+Uo1)/2)*(Uo(i)-Uo1)
                ac = mu*(Uo(i)-Uo1)/rh
                ad = 2*mu*(Uo1-(Uo(i)+Uo1)+Uo(i))/dr
                P(i) = aa+ab-ac-ad+P(i+1)
            end if
            r=r-dr
        end do
        Uo = U
    end do

    !Printing results
    do i = 1, n
        write(1,*) ri(i), P(i)+Patm
    end do
    print *, 'Velo. Diff',U(1)-U(n)
    print *, 'press. Diff',P(1)-P(n)
    write(*,2) 'Radius =', ri
    write(*,1) '(Pressure – Patm) =', P
    write(*,2) 'Velocity =', U
    1 format(a20,100E10.3)
    2 format(a11,100f7.3)
    close (1)
end program Navier_Stokes
```

Radius = 0.010 0.011 0.012 0.013 0.014 0.015 0.016 0.017 0.018 0.019 0.020 0.021 0.022 0.023 0.024 0.025 0.026 0.027 0.028 0.029 0.030 0.031 0.032 0.033 0.034 0.035 0.036 0.037 0.038 0.039 0.040 0.041 0.042 0.043 0.044 0.045 0.046 0.047 0.048 0.049 0.050 0.051 0.052 0.053 0.054 0.055 0.056 0.057 0.058 0.059 0.060 0.061 0.062 0.063 0.064 0.065 0.066 0.067 0.068 0.069 0.070 0.071 0.072 0.073 0.074 0.075 0.076 0.077 0.078 0.079 0.080 0.081 0.082 0.083 0.084 0.085 0.086 0.087 0.088 0.089 0.090 0.091 0.092 0.093 0.094 0.095 0.096 0.097 0.098 0.099 0.100 0.101 0.102 0.103 0.104 0.105 0.106 0.107 0.108 0.109

(Pressure - Patm) = 0.103E-02 0.824E-03 0.781E-03 0.735E-03 0.690E-03 0.648E-03 0.608E-03 0.571E-03 0.537E-03 0.506E-03 0.478E-03 0.452E-03 0.428E-03 0.405E-03 0.385E-03 0.366E-03 0.348E-03 0.332E-03 0.316E-03 0.302E-03 0.288E-03 0.276E-03 0.264E-03 0.253E-03 0.242E-03 0.232E-03 0.223E-03 0.214E-03 0.206E-03 0.198E-03 0.190E-03 0.183E-03 0.176E-03 0.169E-03 0.163E-03 0.157E-03 0.151E-03 0.146E-03 0.140E-03 0.135E-03 0.131E-03 0.126E-03 0.121E-03 0.117E-03 0.113E-03 0.109E-03 0.105E-03 0.101E-03 0.976E-04 0.941E-04 0.907E-04 0.874E-04 0.842E-04 0.812E-04 0.782E-04 0.753E-04 0.726E-04 0.699E-04 0.672E-04 0.647E-04 0.622E-04 0.598E-04 0.574E-04 0.552E-04 0.530E-04 0.508E-04 0.487E-04 0.467E-04 0.447E-04 0.427E-04 0.408E-04 0.390E-04 0.372E-04 0.354E-04 0.337E-04 0.320E-04 0.304E-04 0.288E-04 0.272E-04 0.257E-04 0.242E-04 0.227E-04 0.213E-04 0.199E-04 0.185E-04 0.172E-04 0.159E-04 0.146E-04 0.133E-04 0.121E-04 0.109E-04 0.970E-05 0.854E-05 0.740E-05 0.628E-05 0.518E-05 0.411E-05 0.305E-05 0.202E-05 0.999E-06

Velocity = 0.018 0.015 0.013 0.011 0.010 0.009 0.008 0.007 0.006 0.005 0.005 0.005 0.004 0.004 0.003 0.003 0.003 0.003 0.003 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.001 0.000