Introduction to CFD Basics

Baidurja Ray
Rajesh Bhaskaran
Lance R Collins

February 7, 2012

This is a quick-and-dirty introduction to the basic concepts underlying CFD. The concepts are illustrated by applying them to simple 1D model problems. We will invoke these concepts while performing 'case studies' in FLUENT. Happily for us, these model-problem concepts extend to the more general situations in the case studies in most instances. Since we will keep returning to these concepts while performing the FLUENT case studies, it is worth your time to understand and digest these concepts.

We discuss the following topics briefly. These topics are the minimum necessary to perform and validate the FLUENT calculations to come later.

1. Need for CFD
2. Applications of CFD
3. Strategy of CFD
4. Discretization using the finite-difference method
5. Discretization using the finite-volume method
6. Example problem
   i. Governing equations
   ii. Discretization using finite-volume method
   iii. Assembly of discrete system and application of boundary conditions
   iv. Solution of discrete system
   v. Grid convergence
   vi. Discretization error
   vii. Dealing with nonlinearity
   viii. Direct and Iterative solvers
   ix. Iterative convergence
   x. Numerical stability
7. Explicit and Implicit schemes
8. Turbulence modeling

1 Need for CFD

Applying the fundamental laws of mechanics to a fluid gives the governing equations for a fluid. The conservation of mass equation is

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0$$

and the conservation of momentum equation is

$$\rho \frac{\partial \vec{V}}{\partial t} + \rho(\vec{V} \cdot \nabla)\vec{V} = -\nabla p + \rho \vec{g} + \nabla \cdot \tau_{ij}$$

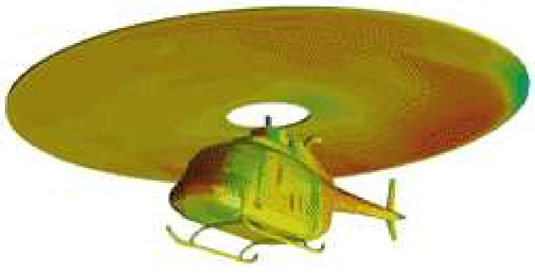
These equations along with the conservation of energy equation form a set of coupled, nonlinear partial differential equations. It is not possible to solve these equations analytically for most engineering problems.

However, it is possible to obtain approximate computer-based solutions to the governing equations for a variety of engineering problems. This is the subject matter of Computational Fluid Dynamics (CFD).

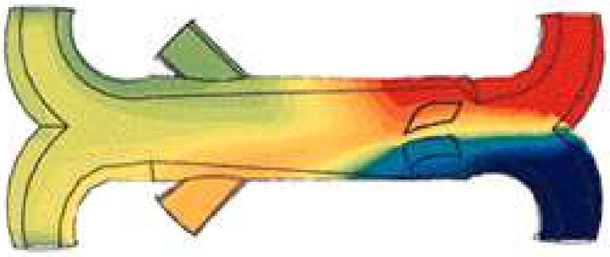
2 Applications of CFD

CFD is useful in a wide variety of applications and here we note a few to give you an idea of its use in industry. The simulations shown below have been performed using the FLUENT software.

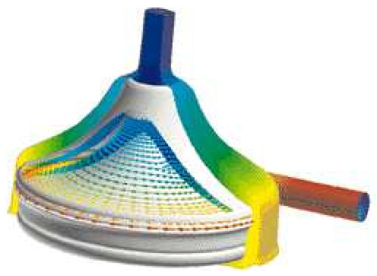
CFD can be used to simulate the flow over a vehicle. For instance, it can be used to study the interaction of propellers or rotors with the aircraft fuselage. The following figure shows the prediction of the pressure field induced by the interaction of the rotor with a helicopter fuselage in forward flight. Rotors and propellers can be represented with models of varying complexity.



The temperature distribution obtained from a CFD analysis of a mixing manifold is shown below. This mixing manifold is part of the passenger cabin ventilation system on the Boeing 767. The CFD analysis showed the effectiveness of a simpler manifold design without the need for field testing.



Bio-medical engineering is a rapidly growing field and uses CFD to study the circulatory and respiratory systems. The following figure shows pressure contours and a cutaway view that reveals velocity vectors in a blood pump that assumes the role of heart in open-heart surgery.



CFD is attractive to industry since it is more cost-effective than physical testing. However, one must note that complex flow simulations are challenging and error-prone and it takes a lot of engineering expertise to obtain validated solutions.

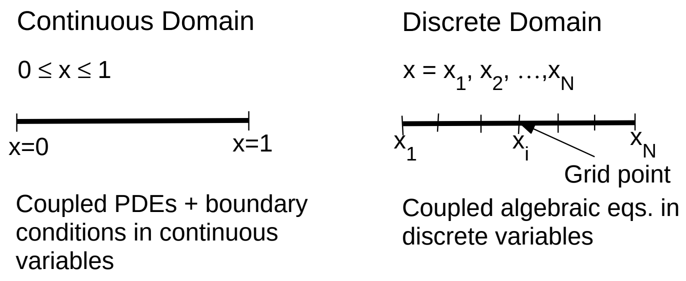
3 Strategy of CFD

Broadly, the strategy of CFD is to replace the continuous problem domain with a discrete domain using a chosen mesh or grid. In the continuous domain, each flow variable is defined at every point in the domain. For instance, the pressure p in the continuous 1D domain shown in the figure below would be given as

$$p = p(x), \ 0 < x < 1$$

In the discrete domain, each flow variable is defined only at the grid points. So, in the discrete domain shown below, the pressure would be defined only at the N grid points.

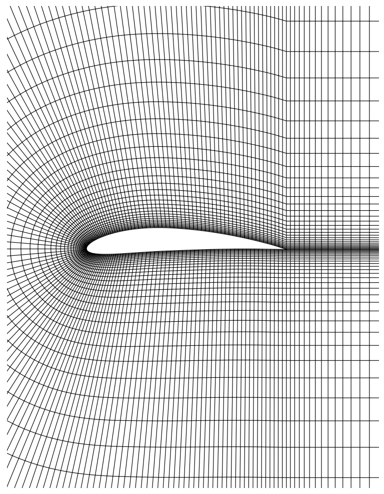
$$p_i = p(x_i), \quad i = 1, 2, \ldots, N$$


Continuous Domain
$0 \le x \le 1$
x=0
x=1
Coupled PDEs + boundary conditions in continuous variables
Discrete Domain
$x = x_1, x_2, \ldots, x_N$
x_1
x_i
x_N
Grid point
Coupled algebraic eqs. in discrete variables

In a CFD solution, one would directly solve for the relevant flow variables only at the grid points. The values at other locations are determined by interpolating the values at the grid points.

The governing partial differential equations and boundary conditions are defined in terms of the continuous variables p, $\vec{V}$ etc. One can approximate these in the discrete domain in terms of the discrete variables p_i, $\vec{V}_i$ etc. The discrete system is a large set of coupled, algebraic equations in the discrete variables. Setting up the discrete system and solving it (which is a matrix inversion problem) involves a very large number of repetitive calculations, a task we humans palm over to the digital computer.

This idea can be extended to any general problem domain. The following figure shows the grid used for solving the flow over an airfoil. We'll take a closer look at this airfoil grid soon while discussing the finite-volume method.

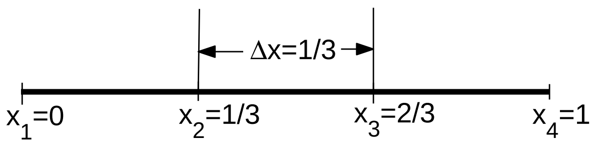


4 Discretization using the finite-difference method

To keep the details simple, consider a simple 1D equation:

$$\frac{du}{dx} + u = 0; \quad 0 \le x \le 1; \quad u(0) = 1 \tag{1}$$

We will derive a discrete representation of the above equation on the following grid:



This grid has four equally-spaced grid points with Δx being the spacing between successive points. Since the governing equation is valid at any grid point, we have

$$\left(\frac{du}{dx}\right)_i + u_i = 0 \tag{2}$$

where the subscript i represents the value at grid point x_i. In order to get an expression for $(du/dx)_i$ in terms of u at the grid points, we expand u_{i-1} in a Taylor's series:

$$u_{i-1} = u_i - \Delta x \left(\frac{du}{dx}\right)_i + O(\Delta x^2)$$

Rearranging gives

$$\left(\frac{du}{dx}\right)_i = \frac{u_i - u_{i-1}}{\Delta x} + O(\Delta x) \tag{3}$$

The error in $(du/dx)_i$ due to the neglected terms in the Taylor's series is called the truncation error. Since the truncation error above is $O(\Delta x)$, this discrete representation is termed first-order accurate.

Using (3) in (2) and excluding higher-order terms in the Taylor's series, we get the following discrete equation:

$$\frac{u_i - u_{i-1}}{\Delta x} + u_i = 0 \tag{4}$$

Note that we have gone from a differential equation to an algebraic equation!

This method of deriving the discrete equation using Taylor's series expansions is called the finite-difference method. However, most commercial CFD codes use the finite-volume or finite-element methods which are better suited for modeling flow past complex geometries. For example, the FLUENT code uses the finite-volume method whereas ANSYS uses the finite-element method. In what follows, we will illustrate the CFD solution process using the finite-volume method, which is the most popular method used in practice.

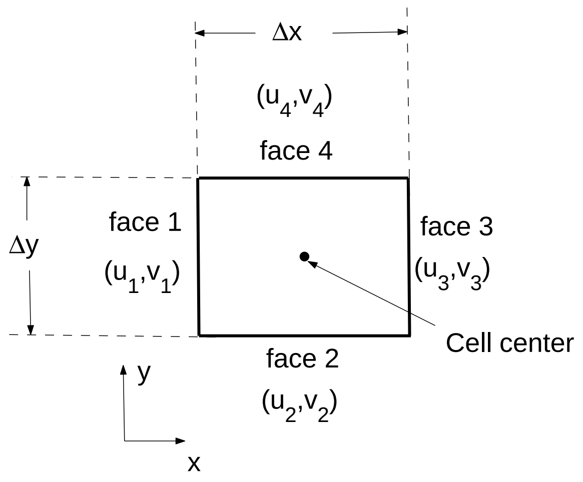
5 Discretization using the finite-volume method

If you look closely at the airfoil grid shown earlier, you'll see that it consists of quadrilaterals. In the finite-volume method, such a quadrilateral is commonly referred to as a "cell". In 2D, one could also have triangular cells. In 3D, cells are usually hexahedrals, tetrahedrals, or prisms. In the finite-volume approach, the *integral form* of the conservation equations are applied to the control volume defined by a cell to get the discrete equations for the cell. The integral form of the continuity equation for steady, incompressible flow is

$$\int_S \vec{\mathbf{V}} \cdot \hat{\mathbf{n}}\, dS = 0 \tag{5}$$

The integration is over the surface S of the control volume and $\hat{\mathbf{n}}$ is the outward normal at the surface. Physically, this equation means that the net volume flow into the control volume is zero.

Consider the rectangular cell shown below.



The velocity at face i is taken to be $\vec{V_i} = u_i\,\hat{i} + v_i\,\hat{j}$. Applying the mass conservation equation (5) to the control volume defined by the cell gives

$$-u_1\Delta y - v_2\Delta x + u_3\Delta y + v_4\Delta x = 0$$

This is the discrete form of the continuity equation for the cell. It is equivalent to summing up the net mass flow into the control volume and setting it to zero. So it ensures that the net mass flow into the cell is zero i.e. that mass is conserved for the cell. Usually, though not always, the values at the cell centers are solved for directly by inverting the discrete system. The face values u_1, v_2, etc. are obtained by suitably interpolating the cell-center values at adjacent cells.

Similarly, one can obtain discrete equations for the conservation of momentum and energy for the cell. We will describe how to implement the finite-volume method for the momentum equations in the 1D example problem below. One can readily extend these ideas to any general cell shape in 2D or 3D and any conservation equation.

Look back at the airfoil grid. When you are using FLUENT, it's useful to remind yourself that the code is finding a solution such that mass, momentum, energy and other relevant quantities are being conserved for each cell. Also, the code directly solves for values of the flow variables at the *cell centers*; values at other locations are obtained by suitable interpolation.

6 Example problem

6.1 Governing equations

Let us consider the classic problem of a viscous fluid flowing in a channel (figure 1) under a constant applied pressure gradient dp/dx. We consider a fully developed, unidirectional flow so that the velocity u is only a function of the vertical coordinate y. Under these assumptions, the Navier-Stokes equations can be simplified to the following single governing equation (x-momentum)

$$0 = -dp/dx + \mu \frac{d^2u(y)}{dy^2} \quad -1 \leq y \leq 1 \tag{6}$$

with boundary conditions given by the no-slip condition at the two channel walls as follows:

$$u(y) = 0 \ at \ y = \pm 1 \tag{7}$$

Equations (6) and (7) constitute a boundary value problem (BVP) which is to be solved numerically. Most *steady-state* problems you will encounter will be BVPs and in the next sections, we will illustrate how to discretize and solve this problem.

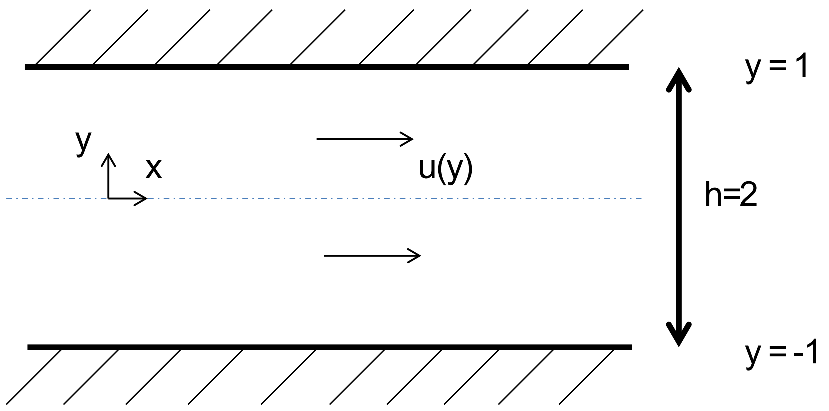


Figure 1: Example problem

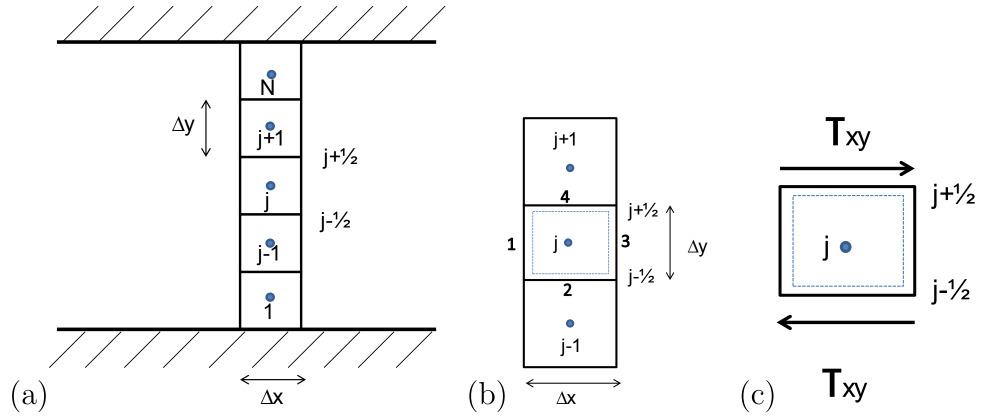


Figure 2: Example problem: discretization

6.2 Discretization using finite-volume method

In this section, we will describe the discretization of equation (6) using the finite-volume (FV) scheme, similar to what FLUENT would do. Let us divide the vertical extent of the domain into N cells each having a height Δy and arbitrary width Δx as shown in figure 2(a). The FV method involves writing out the integral form of the governing equations over a discrete control volume j (CV_j) and then using the divergence theorem to convert it into a surface integral around the control surface (CS_j), which we then evaluate using discrete values of the variable at the center of each cell (note that this is the cell-centered approach employed by FLUENT. A node-centered approach also exists but we will not talk about it here). Referring to figure 2(a), we can write equation (6) as

$$0 = \int_{CV_j} (-dp/dx) d\mathcal{V}_j + \int_{CV_j} \mu \frac{d^2 u(y)}{dy^2} d\mathcal{V}_j \tag{8}$$

Here, we have used the symbol $\mathcal{V}$ for volume, to distinguish it from the velocity vector $\vec{V}$. Since dp/dx is a constant, the first-term on the RHS is really simple to deal with and we will come back to it later. Let us for the moment concentrate on the second term

$$\mu \int_{CV_j} \frac{d^2 u(y)}{dy^2} d\mathcal{V}_j$$

Using the divergence theorem,

$$\int_{CV_j} \frac{d^2 u(y)}{dy^2} d\mathcal{V}_j = \int_{CV_j} \nabla^2 u d\mathcal{V}_j = \int_{CS_j} \nabla u \cdot \hat{n} dS_j$$

where $\hat{n}$ is the outward normal to the control surface. Let us now traverse the the control surfaces of CV_j in CCW sense. Referring to figure 2(b), we have

$$\int_{CS_j} \nabla u \cdot \hat{n} dS_j = \int_{CS_1} \nabla u \cdot \hat{n} dS_1 + \int_{CS_2} \nabla u \cdot \hat{n} dS_2 + \int_{CS_3} \nabla u \cdot \hat{n} dS_3 + \int_{CS_4} \nabla u \cdot \hat{n} dS_4$$

With respect to the axes provided in figure 1, we see that the outward normal $\hat{n}$ to the surface is opposite to the direction of gradient for surfaces 1 and 2 and aligned with it for surfaces 3 and

4. Also, since the flow is fully developed, there are no net fluxes in the horizontal direction, i.e, through surface 1 and 3. Therefore, we can now write for any interior cell j,

$$\begin{aligned}
\int_{CS_j} \nabla u \cdot \hat{n} dS_j &= 0 + \int_{CS_2} \nabla u \cdot \hat{n} dS_2 + 0 + \int_{CS_4} \nabla u \cdot \hat{n} dS_4 \\
&= -\left(\frac{du}{dy}\right)_{j-\frac{1}{2}} \Delta x + \left(\frac{du}{dy}\right)_{j+\frac{1}{2}} \Delta x \\
&= -\frac{u_j - u_{j-1}}{\Delta y}\Delta x + \frac{u_{j+1} - u_j}{\Delta y}\Delta x \\
&= (u_{j-1} - 2u_j + u_{j+1})\frac{\Delta x}{\Delta y}
\end{aligned}$$

This term (multiplied by μ) is essentially a sum of the shear forces acting on the control volume CV_j as shown in figure 2(c), which is balanced by the pressure force. Notice that we have used a central differencing scheme to evaluate the fluxes at the interface of adjacent cells. You can use other schemes as well but it is worth noting that this choice may influence your spatial discretization error. More on that in a bit. But first let us complete our discretization of equation (8) by considering the first term on the LHS. Since dp/dx is a constant, we can write

$$\begin{aligned}
\int_{CV_j} (-dp/dx) d\mathcal{V}_j &= (-dp/dx)\mathcal{V}_j \\
&= (-dp/dx)\Delta x \Delta y
\end{aligned}$$

Therefore, putting everything together we get the discretized form of equation (8) as

$$0 = (-dp/dx)\Delta x \Delta y + \mu(u_{j-1} - 2u_j + u_{j+1})\frac{\Delta x}{\Delta y}$$

Dividing by $\Delta x \Delta y$, we have the final discretized form as

$$0 = (-dp/dx) + \mu \frac{u_{j-1} - 2u_j + u_{j+1}}{(\Delta y)^2} \tag{9}$$

6.3 Assembly of discrete system and application of boundary conditions

In order to make our calculations simple, let us assume $dp/dx = -1$ corresponding to a flow in the postive x direction and $\mu = 1$ and consider a discretization involving $N = 3$ cells as shown in figure 3(a). Then, our final discretized form for an interior cell becomes

$$u_{j-1} - 2u_j + u_{j+1} = -(\Delta y)^2 \tag{10}$$

As discussed earlier, this equation is true as long as our cell (or control volume) is not adjacent to the boundary. For the boundary cells ($j = 1$ and $j = N$ in figure 2(a)), we need to slightly modify the way we calculate the fluxes in order to accomodate the boundary points $N + \frac{1}{2}$ and

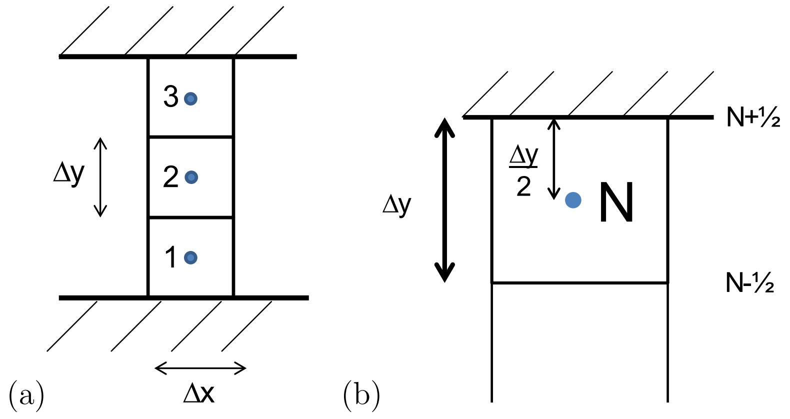


Figure 3: Example problem: discretization

$1-\frac{1}{2}$. Referring to figure 3(b), we can see that for $j=N$,

$$\begin{aligned}\int_{CS_N} \nabla u \cdot \hat{n} dS_N &= -\frac{u_N - u_{N-1}}{\Delta y}\Delta x + \frac{u_{N+\frac{1}{2}} - u_N}{\Delta y/2}\Delta x \\ &= (u_{N-1} - 3u_N + 2u_{N+\frac{1}{2}})\frac{\Delta x}{\Delta y}\end{aligned}$$

Since for $j=N$, there are no points corresponding to $N+1$, we have used the boundary point $N+\frac{1}{2}$ to compute the flux. We can employ the same tactic for the bottom boundary cell $j=1$. Therefore, for $N=3$ cells, our final discretized set of equations corresponding to the governing equation (6) is

$$2u_{1-\frac{1}{2}} - 3u_1 + u_2 = -(\Delta y)^2 \quad (j=1) \tag{11}$$

$$u_1 - 2u_2 + u_3 = -(\Delta y)^2 \quad (j=2) \tag{12}$$

$$u_2 - 3u_3 + 2u_{3+\frac{1}{2}} = -(\Delta y)^2 \quad (j=3) \tag{13}$$

Equations (11)-(13) form a system of three simultaneous algebraic equations in the three unknowns u_1, u_2 and u_3 with specified boundary values $u_{1-\frac{1}{2}}$ and $u_{3+\frac{1}{2}}$, for which we can immediately apply the no-slip boundary conditions $u_{1-\frac{1}{2}} = u_{3+\frac{1}{2}} = 0$. In this case, you can solve these equations by inspection, but for practical systems we need to use a large number of cells. Therefore, it is generally convenient to write this system in matrix form:

$$\begin{pmatrix} -3 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -3 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = -(\Delta y)^2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{14}$$

Now, you can easily see how the system would look like as we add more and more cells. For example, if we have $N=5$, we would have the following system:

$$\begin{pmatrix} -3 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -3 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} = -(\Delta y)^2 \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{15}$$

Make sure that you see how we get this. In a general situation therefore, one would apply the discrete equations to the cells in the interior of the domain. For the cells adjacent to the boundary (or in some cases, near the boundary), one would apply a combination of the discrete equations and boundary conditions. In the end, one would obtain a system of simultaneous algebraic equations with the number of equations being equal to the number of independent discrete variables.

FLUENT, like other commercial CFD codes, offers a variety of boundary condition options such as velocity inlet, pressure inlet, pressure outlet, etc. It is very important that you specify the proper boundary conditions in order to have a well-defined problem to solve numerically. Also, read through the documentation for a boundary condition option to understand what it does before you use it (it might not be doing what you expect). A single wrong boundary condition can give you a totally wrong result.

6.4 Solution of discrete system

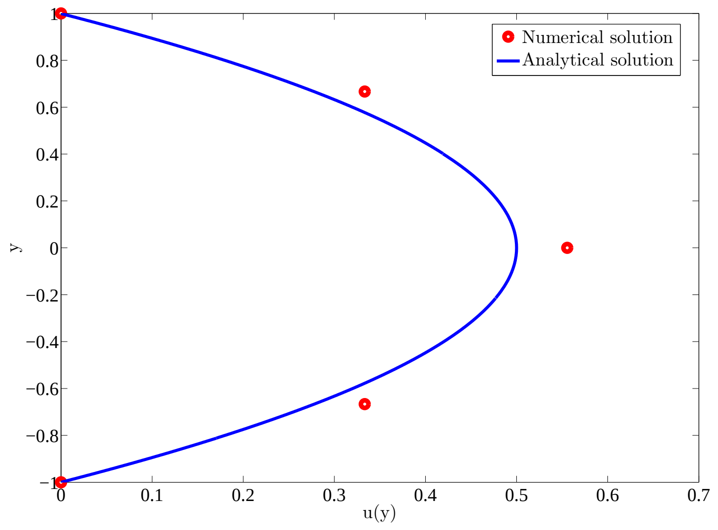
The discrete system (14) for our 1D example can be easily inverted to obtain the unknowns at the grid points. Using $\Delta y = 2/3$, we can solve for u_1, u_2 and u_3 in turn and obtain

$$u_1 = 1/3 \quad u_2 = 5/9 \quad u_3 = 1/3$$

The exact or analytical solution for this problem is easily calculated to be

$$u_{exact}(y) = -y^2/2 + 1/2$$

The figure below shows the comparison of the discrete solution obtained on the three-cell grid along with the exact solution. Verify that the relative error is largest at the cells 1 and 3 each being equal to 20%.

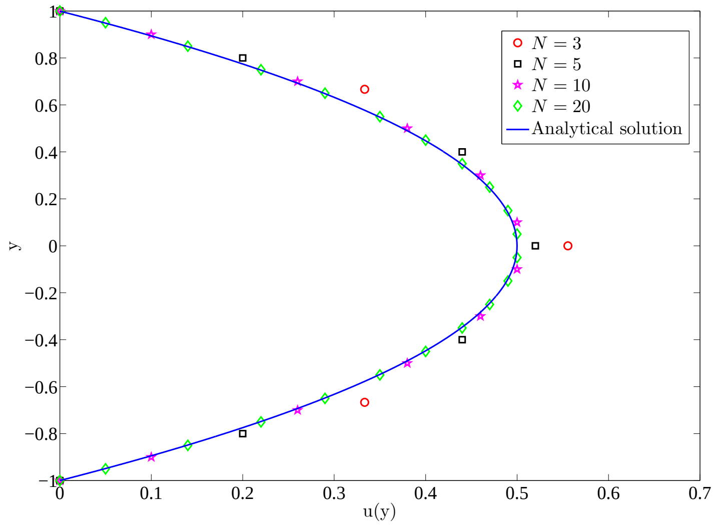


In a practical CFD application, one would have thousands to millions of unknowns in the discrete system and if one uses, say, a Gaussian elimination procedure naively to invert the matrix, you'd probably graduate before the computer finishes the calculation! So a lot of work goes into optimizing the matrix inversion in order to minimize the CPU time and memory required. The matrix to be inverted is usually sparse i.e., most of the entries in it are zeros since the discrete equation at a grid point or cell will contain only quantities at the neighboring points or cells; verify that this will indeed be the case for our matrix system (15) as we increase the number of cells N. A CFD code would store only the non-zero values to minimize memory usage. It would also generally use an iterative procedure to invert the matrix; the longer one iterates, the closer one gets to the true solution for the matrix inversion.

6.5 Grid convergence

We will show later that the spatial discretization error in our finite-volume approximation of equation (6) is $\mathcal{O}(\Delta y^2)$. This means that as we increase the number of cells, we decrease Δy which should lead to reduction in the numerical discretization error.

Let us consider the effect of increasing the number of cells N on the numerical solution. We will consider $N = 5$, $N = 10$ and $N = 20$ in addition to the $N = 3$ case solved previously. We repeat the above assembly and solution steps on each of these additional grids. The resulting discrete system was solved using MATLAB. The following figure compares the results obtained on the four grids with the exact solution. As you can see by inspection, the numerical solution becomes better and better as the number of cells is increased.



When the numerical solutions obtained on different grids agree to within a level of tolerance specified by the user, they are referred to as 'grid converged' solutions. It is very important that you investigate the effect of grid resolution on the solution in every CFD problem you solve. Never trust a CFD solution unless you have convinced yourself that the solution is grid converged to an acceptable level of tolerance (which would be problem dependent).

6.6 Discretization error

We mentioned earlier that the discretization error for our scheme is $\mathcal{O}(\Delta y^2)$. We can see that this is so using a Taylor series expansion. Recall that the flux at interface $j + \frac{1}{2}$ of cell j was given by

$$\left(\frac{du}{dx}\right)_{j+\frac{1}{2}} = \frac{u_{j+1} - u_j}{\Delta y}$$

Let us now consider the Taylor series expansions of u_{j+1} and u_j about some value $u_{j+\frac{1}{2}}$.

$$u_{j+1} = u_{j+\frac{1}{2}} + (\Delta y/2)\left(\frac{du}{dx}\right)_{j+\frac{1}{2}} + \frac{(\Delta y/2)^2}{2}\left(\frac{d^2u}{dx^2}\right)_{j+\frac{1}{2}} + \mathcal{O}(\Delta y^3)$$

$$u_j = u_{j+\frac{1}{2}} + (-\Delta y/2)\left(\frac{du}{dx}\right)_{j+\frac{1}{2}} + \frac{(-\Delta y/2)^2}{2}\left(\frac{d^2u}{dx^2}\right)_{j+\frac{1}{2}} + \mathcal{O}(\Delta y^3)$$

Here, $\mathcal{O}(\Delta y^3)$ indicates that in the series of terms that follow, the term containing the cubic power of Δy will dominate. This is because $\Delta y << 1$, which is a necessary condition to write a Taylor series expansion. Subtracting the two, terms with even powers of Δy cancel and odd powers of Δy add up to give,

$$u_{j+1} - u_j = 2(\Delta y/2)\left(\frac{du}{dx}\right)_{j+\frac{1}{2}} + \mathcal{O}(\Delta y^3)$$

$$\Rightarrow \left(\frac{du}{dx}\right)_{j+\frac{1}{2}} = \frac{u_{j+1} - u_j}{\Delta y} + \mathcal{O}(\Delta y^2)$$

Hence, we see that our discretization scheme is second-order accurate in space i.e., it has an error that is of $\mathcal{O}(\Delta y^2)$. Notice that we have essentially computed the error in a finite-difference approximation (in this case, a central difference scheme) used as part of the finite volume discretization.

6.7 Dealing with nonlinearity

Just for a moment, consider the above flow but now in the developing region of the channel i.e., the flow is not yet fully developed but still steady under a constant pressure gradient. All channel flows will contain a section near the entrance where the flow is not fully developed. Boundary layers start growing on the upper and lower channel walls and the flow can become fully developed only after those two layers have merged. The Navier-Stokes equations along the x-direction for the velocity field (now 2D, $\vec{V} = [u, v]$) for such a developing flow can be written as

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -(1/\rho)dp/dx + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

Such an equation is non-linear due to the convection terms $u\frac{\partial u}{\partial x}$ and $v\frac{\partial u}{\partial y}$, where we have a product of two functions of the dependent variable. Therefore, we have to almost always deal with non-linearity when dealing with the momentum conservation equation for a fluid due to the convection term, which in vector form looks like $(\vec{V}\cdot\nabla)\vec{V}$. Phenomena such as turbulence and chemical reaction introduce additional nonlinearities. The highly nonlinear nature of the governing equations for a fluid makes it challenging to obtain accurate numerical solutions for complex flows of practical interest.

In our example, we can demonstrate the effect of non-linearity and how to handle it, by considering an additional body force per unit volume $-\alpha u^2$ acting on the fluid, where α is a constant. Such forces are not common for normal fluids, but can be observed for exotic fluids like ferrofluids which respond to magnetic fields. Our governing equation, therefore, is modified to

$$0 = -dp/dx + \mu\frac{d^2u(y)}{dy^2} - \alpha u(y)^2 \quad -1 \leq y \leq 1 \tag{16}$$

We can determine the finite-volume approximation to this equation following the method described previously. The additional non-linear term can be discretized assuming a constant value of u in each cell i.e.,

$$\begin{aligned}\int_{CV_j} -\alpha u^2 d\mathcal{V}_j &= -\alpha u_j^2 \mathcal{V}_j \\ &= -\alpha u_j^2 \Delta x \Delta y\end{aligned}$$

Therefore, the discretized equation for each interior cell is given by

$$0 = (-dp/dx) + \mu \frac{u_{j-1} - 2u_j + u_{j+1}}{(\Delta y)^2} - \alpha u_j^2 \tag{17}$$

This is a nonlinear algebraic equation with the u_j^2 term being the source of the nonlinearity.

The strategy that is adopted to deal with nonlinearity is to linearize the equations about a *guess value* of the solution and to iterate until the guess agrees with the solution to a specified tolerance level. We will illustrate this on the above example. Let u_{g_j} be the guess for u_j. Define

$$\Delta u_j = u_j - u_{g_j}$$

Rearranging and squaring this equation gives

$$u_j^2 = u_{g_j}^2 + 2u_{g_j}\Delta u_j + (\Delta u_j)^2$$

Assuming that $\Delta u_j \ll u_{g_j}$ (we want to iterate until this condition is satisfied), we can neglect the Δu_j^2 term to get

$$u_j^2 \simeq u_{g_j}^2 + 2u_{g_j}\Delta u_j = u_{g_j}^2 + 2u_{g_j}(u_j - u_{g_j})$$

Thus,

$$u_j^2 \simeq 2u_{g_j}u_j - u_{g_j}^2$$

After linearization, the discretized equations become

$$0 = (-dp/dx) + \mu \frac{u_{j-1} - 2u_j + u_{j+1}}{(\Delta y)^2} - \alpha(2u_{g_j}u_j - u_{g_j}^2) \tag{18}$$

Since the error due to linearization is $\mathcal{O}(\Delta u^2)$ (remember that is the term we neglected), it will tend to zero as $u_g \to u$.

In order to calculate the finite-volume approximation (18), we need guess values u_g at the grid points. We start with an initial guess value in the first iteration. For each subsequent iteration, the u value obtained in the previous iteration is used as the guess value.

Iteration 1: $u_g^{(1)} =$ Initial guess
Iteration 2: $u_g^{(2)} = u^{(1)}$
$\vdots$
Iteration m: $u_g^{(m)} = u^{(m-1)}$

The superscript indicates the iteration level. We continue the iterations until they converge. We will defer the discussion on how to evaluate convergence until a little later.

This is essentially the process used in CFD codes to linearize the nonlinear terms in the conservations equations, with the details varying depending on the code. The important points to remember are that the linearization is performed about a guess and that it is necessary to iterate through successive approximations until the iterations converge.

6.8 Direct and Iterative solvers

We saw that we need to perform iterations to deal with the nonlinear terms in the governing equations. We next discuss another factor that makes it necessary to carry out iterations in practical CFD problems.

Verify that the discrete equation system resulting from the finite-volume approximation (18) on our three-cell grid is

$$\begin{pmatrix} -3-2\alpha u_{g_j}\Delta y^2 & 1 & 0 \\ 1 & -2-2\alpha u_{g_j}\Delta y^2 & 1 \\ 0 & 1 & -3-2\alpha u_{g_j}\Delta y^2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = -(\Delta y^2) \begin{pmatrix} 1+\alpha u_{g_j}^2 \\ 1+\alpha u_{g_j}^2 \\ 1+\alpha u_{g_j}^2 \end{pmatrix}$$

$$Au = b \tag{19}$$

Note that we can write the discretized system of equations as the standard linear system with the coefficient matrix A, vector of unknowns u and the RHS vector b. In a practical problem, one would usually have thousands to millions of grid points or cells so that each dimension of the above matrix would be of the order of a million (with most of the elements being zeros). Inverting such a matrix directly would take a prohibitively large amount of memory. So instead, the matrix is inverted using an iterative scheme as discussed below.

We can rearrange the finite-volume approximation (19) on an interior cell j so that u_j is expressed in terms of the values at the neighboring cells and the guess values:

$$u_j = \frac{\Delta y^2(1+\alpha u_{g_j}^2) + u_{j-1} + u_{j+1}}{2+2\alpha u_{g_j}\Delta y^2}$$

We can write a similar equation for the boundary cells as well. Here, we consider the *Gauss-Siedel* iterative technique for solving the linear system of equations (19) one at a time starting from a guessed solution and using results from the most recent iteration as soon as they become available. For example, if a neighboring value at the current iteration level is not available, we use the value at the previous iteration but if it has already been computed for the present iteration, we use the current value. Let's say that we sweep from the bottom to the top of our grid i.e., we update u_1 first, based on the value of u_1 and u_2 at the previous iteration. In the next step, when we update u_2, we use the previous iteration for u_3 and u_2, but for u_1 we would use the value already computed in the current iteration, and so on. Thus, in the m^{th} iteration, u_{j+1}^m and u_j^m is not available while u_{j-1}^m is. Therefore, we use the previous iteration values as our guess values $u_{g_j}^m = u_j^{m-1}$ and u_{j+1}^{m-1} to get for the interior cells

$$u_j^m = \frac{\Delta y^2(1+\alpha(u_j^{m-1})^2) + u_{j-1}^m + u_{j+1}^{m-1}}{2+2\alpha u_j^{m-1}\Delta y^2} \tag{20}$$

It is easy to see that for a boundary cell (for example $j=1$), the update looks like

$$u_j^m = \frac{\Delta y^2(1+\alpha(u_j^{m-1})^2) + u_{j+1}^{m-1}}{3+2\alpha u_j^{m-1}\Delta y^2} \tag{21}$$

Note that we use $u_{g_j}^m = u_j^{m-1}$ to iterate for the non-linear terms whereas we use u_{j+1}^{m-1} and u_{j-1}^m to iterate for the matrix inversion. Since we are using some chosen guess values to start our iteration (and previous iteration values as our guesses thereafter) at neighboring points, we are

effectively obtaining only an approximate solution for the inversion of the coefficient matrix A in (19) during each iteration. But in the process, we have greatly reduced the memory required for the inversion. This tradeoff is good strategy since it does not make sense to expend a great deal of resources to do an exact matrix inversion when the matrix elements depend on guess values which are continuously being refined. Notice that 'in an act of cleverness', we have combined the iteration to handle nonlinear terms with the iteration for matrix inversion into a single iteration process. Most importantly, as the iterations converge and $u_g \to u$, the approximate solution for the matrix inversion tends towards the exact solution for the inversion since the error introduced by using u_g instead of u in (20) tends to zero.
Once again, remember that in this case the iteration serves two purposes:

1. It allows for efficient matrix inversion with greatly reduced memory requirements.

2. It is necessary to solve nonlinear equations.

In steady problems, a common and effective strategy used in CFD codes is to solve the unsteady form of the governing equations and 'march' the solution in time until the solution converges to a steady value. In this case, each time step is effectively an iteration, with the the guess value at any time step being given by the solution at the previous time step.

6.9 Iterative convergence

Recall that as $u_g \to u$, the linearization and matrix inversion errors tends to zero. So we continue the iteration process until some selected measure of the difference between u_g and u, refered to as the residual, is 'small enough'. We could, for instance, define the residual R as the absolute value of the difference between u at time-step (or iteration number) 'm' with that at the previous time-step '$m-1$' summed over all the cells:

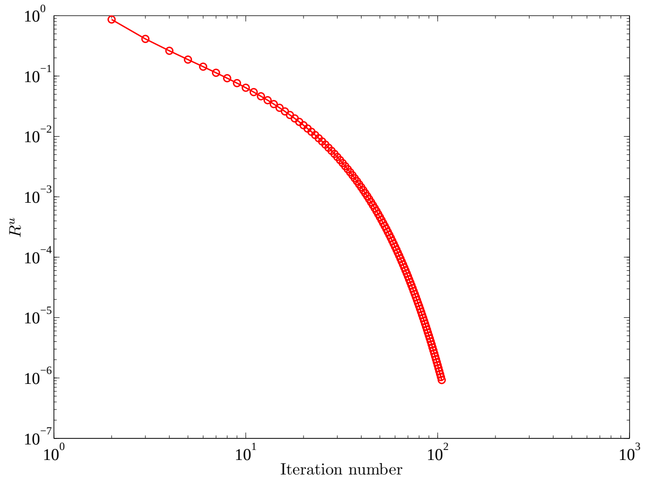
$$R \equiv \sum_{j=1}^{N} |u_j^m - u_j^{m-1}|$$

It is useful to scale this residual in some way because an unscaled residual of, say, 0.01 would be relatively small if the average value of u in the domain is 5000 but would be relatively large if the average value is 0.1. Scaling ensures that the residual is a relative rather than an absolute measure. Scaling the above residual by dividing by the absolute sum of u over all the cells gives

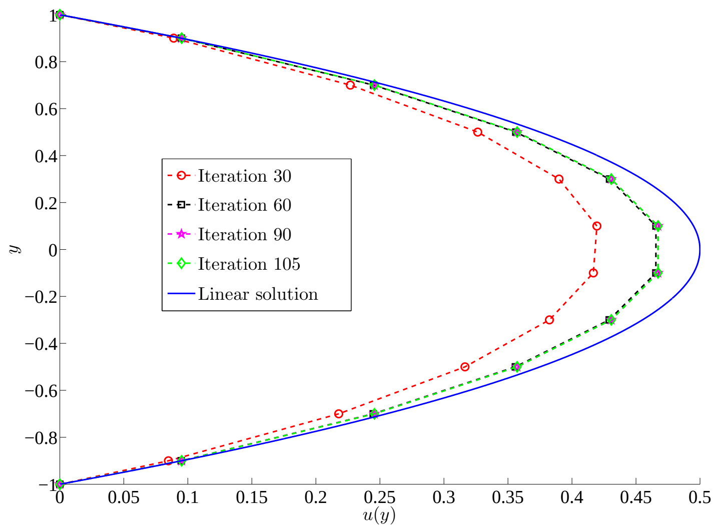
$$R^u \equiv \frac{\displaystyle\sum_{j=1}^{N} |u_j^m - u_j^{m-1}|}{\displaystyle\sum_{j=1}^{N} |u_j^{m-1}|} \tag{22}$$

In fact, this is the form of the residual that FLUENT uses for its pressure-based solver for momentum and scalar-transport equation. For full details on how FLUENT computes residuals in other cases, please refer to the user manual.

For the nonlinear example problem we have, we will take the initial guess at each cell to be equal to zero (arbitrarily) i.e., $u_{g_j}^{(1)} = 0$ for all j and consider $N = 10$ cells. In each iteration from here on, we update u, going from cell $j = 1$ to $j = N$ using equation (20) for the interior cells and equation (21) for the boundary cells. We can calculate the residual using (22) at each time-step and monitor it to determine convergence. We usually terminate the iterations when the residual falls below some threshold value (for example, 10^{-6}), which is referred to as the convergence criterion. Take a few minutes to implement this procedure in MATLAB which will help you gain some familiarity with the mechanics of the implementation. The variation of the residual with iterations obtained from MATLAB is shown below. Note that a logarithmic scale is used for the ordinate. Verify that the iterative process converges to residual level smaller than 10^{-6} in about 105 iterations. In more complex problems and/or with a more stringent convergence criterion, a lot more iterations would be necessary for achieving convergence.



The solution after 30, 60 and 90 iterations and the converged solution (105 iterations) are shown below along with the linear solution obtained previously. The solutions for iterations 90 and 105 are indistinguishable on the graph. This is another indication that the solution has converged. We can see that the negative force (in the x-momentum equation) on the sytem leads to a drag that reduces the velocities across the channel. Unfortunately, we do not have an analytical solution for this problem. Therefore, we should be careful not to trust our numerical solution blindly. The topic of verification and validation of the numerical solution is critically important to a CFD engineer. We will discuss it in class later on. For now, we can at least see that our solution is converged. Also, you can easily evaluate whether your solution is grid converged or not by varying the number of cells N. Checking for solution convergence and grid convergence are the two most simple checks you can do to determine whether your numerical solution is correct or not. Further methods of validating your solution will be discussed in class. Another consideration is the comparison of the iterative convergence error, which is of order 10^{-6}, and the truncation error which is of order $\Delta y^2 \sim 10^{-2}$. So, although we drive the residual down to 10^{-6}, the accuracy of our solution is limited by the truncation error (of order 10^{-2}), which is a waste of computing resources. In a good calculation, both errors would be of comparable level and less than a tolerance level chosen by the user. Therefore, our numerical solution would get better on refining the grid so that truncation error does not dominate.



Some points to note:

1. Different codes use slightly different definitions for the residual. Read the documentation to understand how the residual is calculated.

2. In the FLUENT code, residuals are reported for each conservation equation. The calculation of the residual is similar for the momentum and energy equations (which is similar to what we have described) but different for the continuity equation. It also depends on whether we are solving compressible or incompressible flows (i.e., the solver used). Please refer to the FLUENT user manual for detailed descriptions.

3. The convergence criterion you choose for each conservation equation is problem and code-dependent. It is a good idea to start with the default values in the code. One may then have to tweak these values based on the problem.

6.10 Numerical stability

In our example problem, the iterations converged very rapidly with the residual falling below the convergence criterion of 10^{-6} in just 105 iterations. In more complex problems, the iterations converge more slowly and in some instances, may even diverge. One would like to know *a* priori the conditions under which a given numerical scheme converges. This is determined by performing a stability analysis of the numerical scheme. A numerical method is referred to as being stable when the iterative process converges and as being unstable when it diverges. It is not possible to carry out an exact stability analysis for the Euler or Navier-Stokes equations. But a stability analysis of simpler, model equations provides useful insight and approximate conditions for stability. A common strategy used in CFD codes for steady problems is to solve the unsteady equations and march the solution in time until it converges to a steady state. A stability analysis is usually performed in the context of time-marching.

While using time-marching to a steady state, we are only interested in accurately obtaining the asymptotic behavior at large times. So we would like to take as large a time-step Δt as possible to reach the steady state in the least number of time-steps. There is usually a maximum allowable time-step Δt_{max} beyond which the numerical scheme is unstable. If $\Delta t > \Delta t_{max}$, the numerical errors will grow exponentially in time causing the solution to diverge from the steady-state result. The value of Δt_{max} depends on the numerical discretization scheme used. Two major classes of numerical shemes are explicit and implicit, with very different stability characteristics as we will briefly discuss next.

7 Explicit and Implicit schemes

The difference between explicit and implicit schemes can be most easily illustrated by applying them to the wave equation

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0$$

where c is the wavespeed. One possible way to discretize this equation at grid point i and time-level n is

$$\frac{u_i^n - u_i^{n-1}}{\Delta t} + c\,\frac{u_i^{n-1} - u_{i-1}^{n-1}}{\Delta x} = O(\Delta t, \Delta x) \tag{23}$$

The crucial thing to note here is that the spatial derivative is evaluated at the $n-1$ time-level. Solving for u_i^n gives

$$u_i^n = \left[1 - \left(\frac{c\Delta t}{\Delta x}\right)\right] u_i^{n-1} + \left(\frac{c\Delta t}{\Delta x}\right) u_{i-1}^{n-1} \tag{24}$$

This is an explicit expression i.e. the value of u_i^n at any grid point can be calculated directly from this expression without the need for any matrix inversion. The scheme in (23) is known as an explicit scheme. Since u_i^n at each grid point can be updated independently, these schemes are easy to implement on the computer. On the downside, it turns out that this scheme is stable only when

$$\mathrm{C} \equiv \frac{c\Delta t}{\Delta x} \leq 1$$

where C is called the Courant number. This condition is refered to as the Courant-Friedrichs-Lewy or CFL condition. While a detailed derivation of the CFL condition through stability analysis is outside the scope of the current discussion, it can seen that the coefficient of u_i^{n-1} in (24) changes sign depending on whether $C > 1$ or $C < 1$ leading to very different behavior in the two cases. The CFL condition places a rather severe limitation on Δt_{max}.

In an implicit scheme, the spatial derivative term is evaluated at the n time-level:

$$\frac{u_i^n - u_i^{n-1}}{\Delta t} + c\,\frac{u_i^n - u_{i-1}^n}{\Delta x} = O(\Delta t, \Delta x)$$

In this case, we can't update u_i^n at each grid point independently. We instead need to solve a system of algebraic equations in order to calculate the values at all grid points simultaneously. It can be shown that this scheme is unconditionally stable for the wave equation so that the numerical errors will be damped out irrespective of how large the time-step is.

The stability limits discussed above apply specifically to the wave equation. In general, explicit schemes applied to the Euler or Navier-Stokes equations have the same restriction that the Courant number needs to be less than or equal to one. Implicit schemes are *not* unconditonally stable for the Euler or Navier-Stokes equations since the nonlinearities in the governing equations often limit stability. However, they allow a much larger Courant number than explicit schemes. The specific value of the maximum allowable Courant number is problem dependent.

Some points to note:

1. CFD codes will allow you to set the Courant number (which is also referred to as the CFL number) when using time-stepping. Taking larger time-steps leads to faster convergence to the steady state, so it is advantageous to set the Courant number as large as possible, within the limits of stability, for steady problems.

2. You may find that a lower Courant number is required during startup when changes in the solution are highly nonlinear but it can be increased as the solution progresses.

8 Turbulence modeling

There are two radically different states of flows that are easily identified and distinguished: laminar flow and turbulent flow. Laminar flows are characterized by smoothly varying velocity fields in space and time in which individual "laminae" (sheets) move past one another without generating cross currents. These flows arise when the fluid viscosity is sufficiently large to damp out any perturbations to the flow that may occur due to boundary imperfections or other irregularities. These flows occur at low-to-moderate values of the Reynolds number. In contrast, turbulent flows are characterized by large, nearly random fluctuations in velocity and pressure in both space and time. These fluctuations arise from instabilities that grow until nonlinear interactions cause them to break down into finer and finer whirls that eventually are dissipated (into heat) by the action of viscosity. Turbulent flows occur in the opposite limit of high Reynolds numbers.

A typical time history of the flow variable u at a fixed point in space is shown in Fig. 4(a). The dashed line through the curve indicates the "average" velocity. We can define three types of averages:

1. Time average

2. Volume average

3. Ensemble average

The most mathematically general average is the ensemble average, in which you repeat a given experiment a large number of times and average the quantity of interest (say velocity) at the same position and time in each experiment. For practical reasons, this is rarely done. Instead, a time or volume average (or combination of the two) is made with the assumption that they are equivalent to the ensemble average. For the sake of this discussion, let us define the time average for a stationary flow[1] as

$$\overline{u}(y) \equiv \lim_{\tau \to \infty} \frac{1}{2\tau} \int_{-\tau}^{\tau} u(y,t)dt \tag{25}$$

The deviation of the velocity from the mean value is called the fluctuation and is usually defined as

$$u' \equiv u - \overline{u} \tag{26}$$

Note that by definition $\overline{u'} = 0$ (the average of the fluctuation is zero). Consequently, a better measure of the *strength* of the fluctuation is the average of the *square* of a fluctuating variable. Figures 4(b) and 4(c) show the time evolution of the velocity fluctuation, u', and the square of that quantity, u'^2. Notice that the latter quantity is always greater than zero as is its average.

[1]A stationary flow is defined as one whose statistics are not changing in time. An example of a stationary flow is steady flow in a channel or pipe.

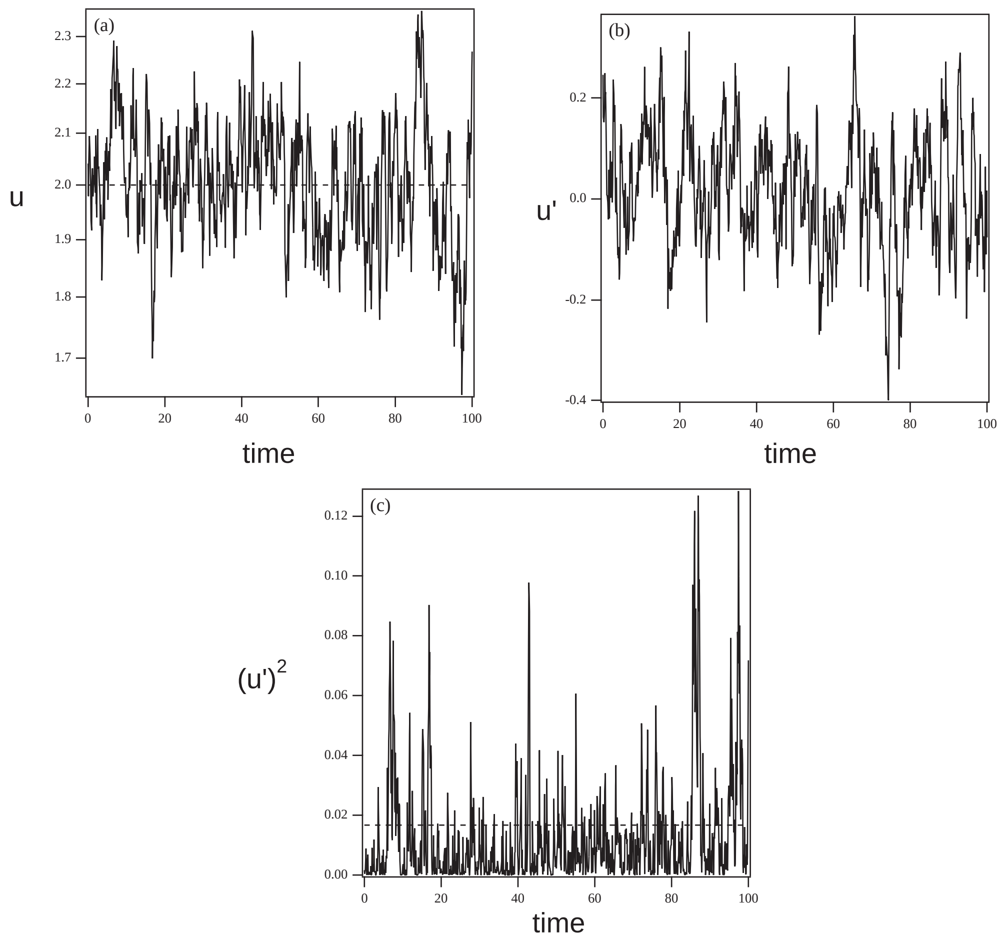


Figure 4: Example of a time history of a component of a fluctuating velocity at a point in a turbulent flow. (a) Shows the velocity, (b) shows the fluctuating component of velocity $u' \equiv u - \overline{u}$ and (c) shows the square of the fluctuating velocity. Dashed lines in (a) and (c) indicate the time averages.

The equations governing a turbulent flow are precisely the same as for a laminar flow; however, the solution is clearly much more complicated in this regime. The approaches to solving the flow equations for a turbulent flow field can be roughly divided into two classes. Direct numerical simulations (DNS) use the speed of modern computers to numerically integrate the Navier Stokes equations, resolving all of the spatial and temporal fluctuations, without resorting to modeling. In essence, the solution procedure is the same as for laminar flow, except the numerics must contend with resolving all of the fluctuations in the velocity and pressure. DNS remains limited to very simple geometries (e.g., channel flows, jets and boundary layers) and is extremely expensive to run.[2] The alternative to DNS found in most CFD packages (including FLUENT) is to solve the Reynolds Averaged Navier Stokes (RANS) equations. RANS equations govern the *mean* velocity and pressure. Because these quantities vary smoothly in space and time, they are much easier to solve; however, as will be shown below, they require *modeling* to "close" the equations and *these models introduce significant error into the calculation.*

To demonstrate the closure problem, we consider fully developed turbulent flow in a channel

[2]The largest DNS to date was recently published by Kaneda et al., *Phys. Fluids* **15**(2):L21–L24 (2003); they used 4096^3 grid point, which corresponds roughly to 0.5 terabytes of memory per variable!

of height $2H$. Recall that with RANS we are interested in solving for the *mean* velocity $\overline{u}(y)$ only. If we formally average the Navier Stokes equations and simplify for this geometry we arrive at the following

$$\frac{d\overline{u'v'}}{dy} + \frac{1}{\rho}\frac{d\overline{p}}{dx} = \nu \frac{d^2\overline{u}(y)}{dy^2} \tag{27}$$

subject to the boundary conditions

$$\begin{aligned} y &= 0 \quad \frac{d\overline{u}}{dy} = 0\ , && (28)\\ y &= H \quad \overline{u} = 0\ , && (29) \end{aligned}$$

The kinematic viscosity $\nu{=}\mu/\rho$. The quantity $\overline{u'v'}$, known as the Reynolds stress,[3] is a higher-order moment that must be modeled in terms of the knowns (i.e., $\overline{u}(y)$ and its derivatives). This is referred to as the "closure" approximation. The quality of the modeling of this term will determine the reliability of the computations.[4]

Turbulence modeling is a rather broad discipline and an in-depth discussion is beyond the scope of this introduction. Here we simply note that the Reynolds stress is modeled in terms of two turbulence parameters, the turbulent kinetic energy k and the turbulent energy dissipation rate ϵ defined below

$$\begin{aligned} k &\equiv \frac{1}{2}\left(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}\right) && (30)\\ \epsilon &\equiv \nu\left[\left(\frac{\partial u'}{\partial x}\right)^2 + \left(\frac{\partial u'}{\partial y}\right)^2 + \left(\frac{\partial u'}{\partial z}\right)^2 + \left(\frac{\partial v'}{\partial x}\right)^2 + \left(\frac{\partial v'}{\partial y}\right)^2 + \left(\frac{\partial v'}{\partial z}\right)^2\right. \\ &\qquad \left. + \left(\frac{\partial w'}{\partial x}\right)^2 + \left(\frac{\partial w'}{\partial y}\right)^2 + \left(\frac{\partial w'}{\partial z}\right)^2\right] && (31) \end{aligned}$$

where (u', v', w') is the fluctuating velocity vector. The kinetic energy is *zero* for laminar flow and can be as large as 5% of the kinetic energy of the mean flow in a highly turbulent case. The family of models is generally known as k–ϵ and they form the basis of most CFD packages (including FLUENT). We will revisit turbulence modeling towards the end of the semester.

Acknowledgements

The authors acknowledge Sam Smith for helping with the schematic figures.

[3]Name after the same Osborne Reynolds from which we get the Reynolds number.

[4]Notice that if we neglect the Reynolds stress the equations reduce to the equations for laminar flow; thus, the Reynolds stress is solely responsible for the difference in the mean profile for laminar (parabolic) and turbulent (blunted) flows.