Chapter

Introductory Chapter: A Brief History of and Introduction to Computational Fluid Dynamics

Suvanjan Bhattacharyya, John P. Abraham, Lijing Cheng and John Gorman

1. Introduction

Werner Heisenberg: "*When I meet God, I am going to ask him two questions. Why relativity and why turbulence? I really believe he will have an answer for the first*."

Richard Feynman: "*There is a physical problem that is common to many fields, that is very old, and that has not been solved. It is not the there is a problem of finding new fundamental particles, but something left over from a long time ago … over a hundred years. Nobody in physics has been able to analyze it satisfactorily in spite of its importance to the sister sciences. It is the analysis of circulating or turbulent fluids.*"

These quotes, both from Nobel prize winning scientists, may be apocryphal. But they nevertheless help provide context for our current understanding of fluid flow. And while these statements were reportedly made many decades ago, there is still truth to them. But at a risk of being brazen, we may now have the critical tools necessary to solve complex fluid flow problems with acceptable accuracy and fidelity. Those tools of numerical simulation are the focus of this chapter and this book.

It is not an overstatement to call numerical methods in general, and numerical simulation of fluid flow, in particular, a critical development in science in the past 100 years. In this chapter, we intend to briefly discuss the historical development of this science before quickly moving into the essential aspect of its practice. In our view, perhaps the most important aspect of numerical methods is that they provide solutions to problems that would otherwise require extensive experimentation or would otherwise be intractable. Simply put, numerical simulation opens the door for solutions to many academic and real-world problems that could otherwise not be solved.

One reason for the importance of numerical simulation for thermal-fluid problems is that the governing equations of motion are highly non-linear and coupled. That is, solutions require the simultaneous consideration of momentum (in all three coordinate directions), conservation of mass, conservation of energy (particularly for problems that involve heat transfer), and the potential for additional turbulence equations and species conservation/reaction equations. Such problems are not capable of being solved analytically. Furthermore, experimentation is often prohibitively expensive, time consuming, or impossible.

As we will see, numerical simulations of flows are important at many spatial and temporal scales. From the nanoscale to astronomical scales, from microseconds to millennia. At the large scales, climate and weather simulations are nearly ubiquitously used to make predictions. Small-scale examples include flows of fluid through microchannels or around micron-scale (or smaller objects) are

representative. The breadth of scales is indicative of the wide applications this technique has been applied to.

2. A brief history of CFD

Computational Fluid Dynamics (CFD) refers to a broad set of methods that are used to solve the coupled nonlinear equations that govern fluid motion. To our best knowledge, the first attempt to calculate fluid flow was set forth by Lewis Fry Richardson, with applications for weather prediction. He envisioned a "forecast factory" that included 64,000 human "computers". Each "computer" was positioned at tiered elevations around a spherical globe, occupying computational cells that corresponded to map locations, as shown below for northern Europe. His method involved inputting weather observation data to the corresponding grid locations and then solving the forward-stepping equations.

Based on Richardson's description, the following image provides the imagined weather prediction system, commonly referred to as the "fantastic weather factory" of Lewis Richardson. Each of the red and white grid cells represents a human calculator. They are arranged across the surface of a sphere (which represents the Earth). In the center, a conductor uses spotlights to highlight calculated results at each grid cell. It was acknowledged that for such a system to work, each human calculator would be required to perform their calculations at the same speed. That is, if one human calculator was either faster or slower than its neighbors, it would send information to the neighbors at a faster or slower rate which would consequently cause numerical instability; a concept that is important even today as we will show. In the image, the blue spotlight identifies calculators that are operating too slowly, and the red spotlight identifies those that are too fast.

While the vision of Richardson is somewhat fanciful, it nevertheless provides the foundation for modern day numerical simulation. His vision was remarkably prescient.

It was not until electric computers were developed that CFD could really begin its development and the human computers of Richardson could be replaced by digital computation. This generally commenced in the 1940s with the ENIAC programmable digital computer. Small scale simulations began to appear in the scientific literature in the early 1950s (for example [2]).

3. Stability of CFD simulations and the CFD process

As indicated in the discussion of **Figures 1** and **2**, many applications of CFD are inherently unsteady. Consider for example CFD for weather forecasting where local velocity, pressure and temperatures are varying continuously. In other cases, the flow situation may initially appear steady, with steady boundary conditions. However, the flow patterns emerge as naturally unsteady. **Figure 3** provides an example of such a situation. There, steady flow approaches a small square cylinder from the left. In the downstream wake region, the flow expresses unsteady alternating vortices (the so-called Karmen vortex structure). Whenever flow experiences unsteadiness, a modeler should consider the time steps required for solution stability. Even with a situation like that of **Figure 3**, integration forward in time and stability criteria are important.

The unsteadiness, in connection with the coupled nature of the nonlinear equations, make the stability of the solutions a particular vexing issue. In fact, the issue of stability was recognized in the early 1900s and criteria for stability were soon

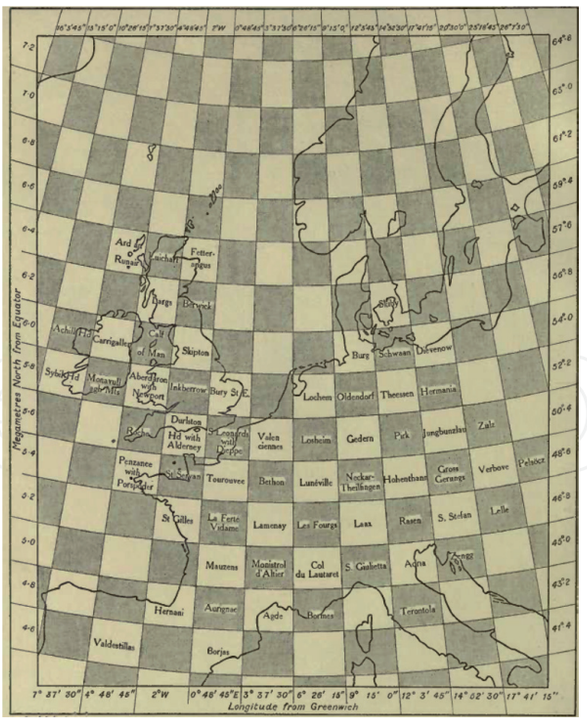

Figure 1.
Numerical grid distribution, from [1].

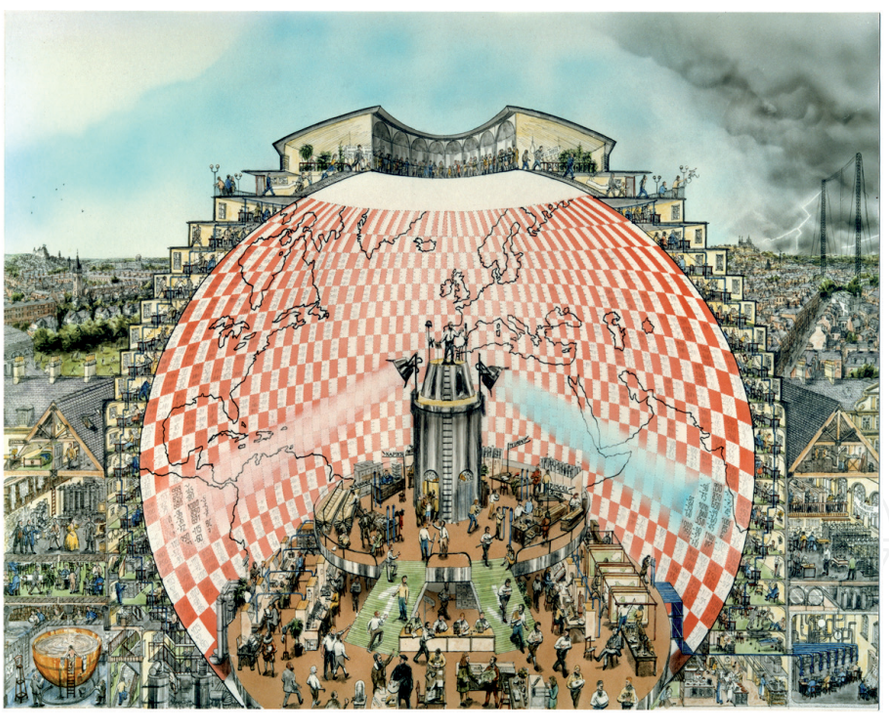

Figure 2.
Artist depiction of the Richardson "fantastic weather factory" Image: ink and watercolour © Stephen Conlin 1986. All Rights Reserved. Based on advice from Prof. John Byrne, Trinity College Dublin.

developed. In 1928 [3] a stability criterion was developed that provided a limitation to the time step that could be used in unsteady problems. The criterion, now termed the (Courant–Friedrichs–Lewy) CFL condition, is still used today. In short, the CFL conditions stipulates that information cannot flow entirely across a computational element in a single time step. Consequently, the local velocity of fluid, multiplied by the time step, must be smaller than the element size. This issue was highlighted in **Figure 2**, by the imagined red and blue spotlights.

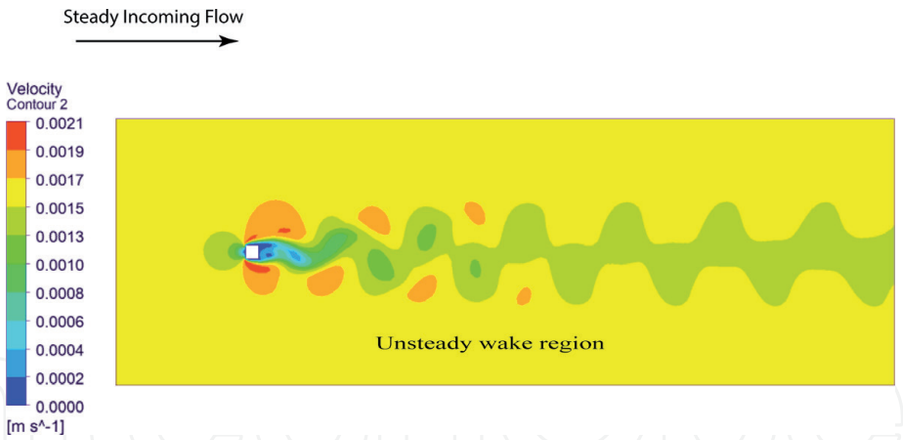

Figure 3.
An unsteady flow that results from steady boundary conditions.

The CFL criterion was developed for explicit numerical schemes but it is also used for implicit schemes as a timestep benchmark. It is worthwhile to discuss "explicit" and "implicit" numerical schemes. To aid in the discussion, **Figure 4** is provided. There, a user begins a CFD analysis by creating the flow geometry (which includes the volume occupied by the fluid). Next, the computational mesh is created (which is a collection of grid cells used to subdivide the domain.

An example of the computational mesh that was used to provide the results set forth in **Figure 3**, is shown below. These images are from the present authors' research but are typical of general CFD mesh deployments. In the figure, a series of images are provided with increasing focus on the fluid region adjacent to a square object. As seen in the series of images, the elements in the vicinity of the object are much finer than elements in further away. The use of locally refined elements is a technique to provide high accuracy in areas that are of critical importance to the analysis.

Typically, a researcher will not know *a priori* whether a particular mesh is suitable for a calculation. The requirements of the mesh depend in part on the necessary accuracy of the simulations. However, it is common for researchers to perform mesh refinement studies to ensure a level of accuracy. In a mesh refinement

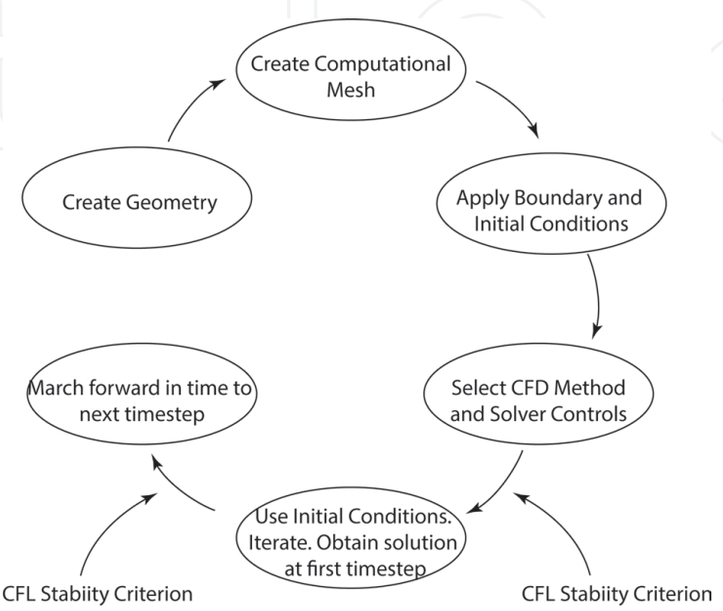

Figure 4.
The CFD process.

study, the user will refine a computational mesh and perform replicate simulations, until further refinement of the mesh does not lead to significant changes in the results. Such a situation is termed "mesh independent". It is standard practice that final solutions are mesh independent and at least within the scientific literature, mesh-independent solutions are standard. For unsteady simulations, the size of the time step is also important and similar time-step independent studies can also be performed.

With the computational mesh now created, the user moves to the next step in the process which is the application of boundary and initial conditions. Traditionally, initial conditions refer to the starting conditions of an unsteady problem. However with CFD, initial conditions are necessary even if the problem is truly steady state. For steady problems, initial conditions are the solution that commences the iterations.

Next, the numerical method and solver controls are defined. This step includes decisions such as:

- Is the flow laminar or turbulent?
- If the flow is turbulent, what turbulent model will be employed?
- Will an explicit or implicit solver be used?
- How many iterations should be performed to converge the system of equations?
- How small should the iteration-to-iteration changes be before convergence?
- Is relaxation necessary?

Next, the actual calculations can commence. The calculations involve iterating to convergence the coupled nonlinear equations. At each computational element (grid cell) the equations of mass, momentum, and energy conservation are applied. So to are the equations of turbulence. For a two-equation turbulent model, each grid cell will require seven equations (mass, three momentum, one energy, two turbulence). Consequently, a 1,000,000 element simulation will result in 7,000,000 coupled, nonlinear equations. Obviously an iteration solution strategy is required.

This iteration procedure results in a solution at the first time step. Since a time step has been taken, the CFL stability criterion is employed. Traditionally, the CFL stability criterion is enforced for explicit time-stepping schemes, but not for implicit methods. With an implicit time-stepping method, the results at the next time step are solely based on the solution at the prior time step. A result of an explicit scheme is that the equations are simpler to formulate and solve, compared to implicit methods.

As an alternative to explicit time stepping, a user may wish to use an implicit approach. With implicit algorithms, the pressure, velocity, temperature and turbulent results at a future time step depend both on the results of the prior time step as well as on the results of the future time step. Obviously, such a definition requires a more comprehensive sub-iteration procedure in order to converge to a solution, but the results are categorically stable (and therefore not subject to the CFL criterion). There are variations in implicit schemes, for example a fully implicit scheme relies only on information at the current time step. On the other hand, a Crank-Nicholson scheme relies equally upon results at a prior and future time step. Regardless of the details, implicit schemes are stable.

In our view, the stability of implicit schemes is not a strength, rather it is a weakness. The basis for this opinion is that it is possible to use a time step that is too large to achieve accurate solutions with an implicit solver, but the solution will nevertheless be stable. An unexperienced CFD research may presume that a stable solution is also an accurate solution – but this presumption is often in error. Therefore, even when using an implicit time stepping scheme, the user should pay close attention to the influence of time step size on accuracy and we recommend that the CFL criterion be applied as a guide for determining the required time step size.

It is also important to recognize that the time step size varies inversely with the element size. Consequently, when elements are made small to improve accuracy, the time steps also must become smaller to ensure convergence. Because of this, for unsteady calculations, a mesh refinement study will require more effort to solve each iteration, and more time steps are required because the time steps must be accordingly smaller.

The above discussion relates to what are often termed "numerical error". But there is another, more nuanced source of error we refer to as "modeling error". Modeling error is not related to element size or time step size, it is instead focused on the inexact input of material properties, boundary conditions, and other features of the simulation. Colloquially, we refer to "garbage in gives garbage out" and this adage is true. Insofar as inputs to the computational model deviate from a real-life situation, a user can expect differences between the simulated and actual results. In our experience, modeling errors are more significant than numerical errors. They are often much harder to diagnose and remove. Our recommendation is that CFD users pay particular attention to ensuring the inputs to their computational model match the expected inputs in real life.

4. Summary of other important CFD advances

Here we will discuss some technical innovations that have allowed CFD to become a widely available tool for researchers.

4.1 Coupling of velocity–pressure equations

Regardless of whether a simulation is steady or unsteady, an iterative process must be undertaken wherein solutions are fed back into the coupled equations and then the solutions are updated to improve accuracy. The pressure and velocity fields are coupled via the governing equations and it is challenging to calculate them simultaneously. A series of approaches was developed that began with the so-called SIMPLE algorithm (Semi-Implicit Method for Pressure Linked Equations) [4, 5]. These references provide extensive detail on the algorithm, which has been joined by a modified SIMPLER version, the SIMPLEC method, and the PISO method. All of these methods continue to be used in today's algorithms. Use of these approaches greatly improves the stability of the pressure–velocity coupling.

4.2 Multigrid calculations

Another major innovation has been the utilization of multigrid simulations to increase the speed of solution. With a multi-grid method, the governing equations are first solved on a coarse mesh. This solution is then sent to a new mesh that contains smaller elements and iterations with the new mesh are performed. The solution is again passed to a further refined mesh and the process continues until the mesh is identical with the cells that the user has defined (for example in **Figure 5**). Following a converged

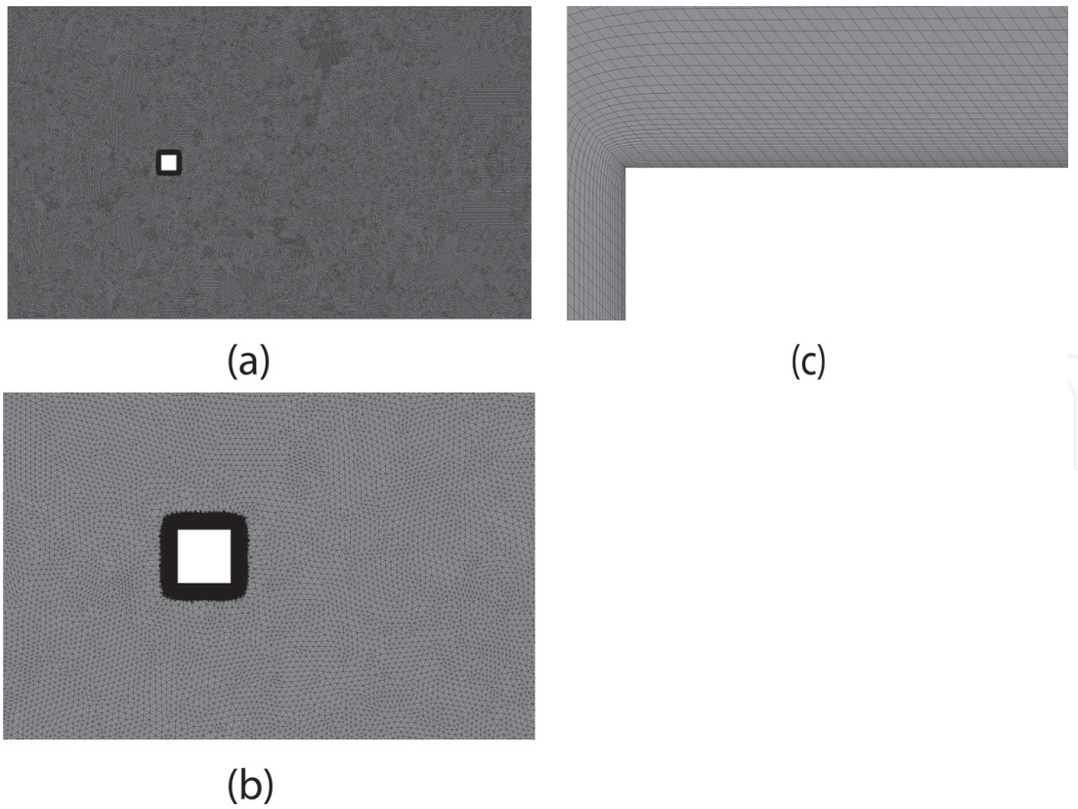

Figure 5.
A computational mesh with increased focus on the near-wall region.

solution on the smallest cells, the process is reversed, information is passed to increasingly coarse meshes until the algorithm arrives back at the initial, coarse mesh.

Through this process, solution information is transferred from the boundary conditions to the interior of the solution domain much faster than if the solution were obtained only on the finest computational mesh. It should be recognized that information is able to travel the distance of a cell size in a single iteration. If the solution domain is composed entirely of small elements, it means many iterations must be processed simply to transfer boundary information to the interior of the domain. The multi-grid approach largely solves this problem.

4.3 Parallel computing

Parallel computing refers to the simultaneous use of multiple processors during the iterative procedure. The software separates the computational mesh and parses different parts of the mesh to individual CPUs. Each CPU performs iterations on its batch of cells. The results are then collected, information is sent between the batches of cells, and the process is repeated.

To provide an example, a simulation that requires 10,000,000 elements that is solved with 10 processors would involve an allocation of approximately 1,000,000 elements to each processor. Clearly solving for 1,000,000 elements is far faster than the entire solution, and thus a time savings is achieved. On the other hand, there is a loss of efficiency in the partitioning (separating the elements into individual batches for each processor and passing information between the processors).

Parallel processing speed is measured as follows:

$$Speed\ Up\ Time = \frac{Time\ Required\ for\ Single\ Processor}{Time\ Required\ for\ Multiple\ Processors} \tag{1}$$

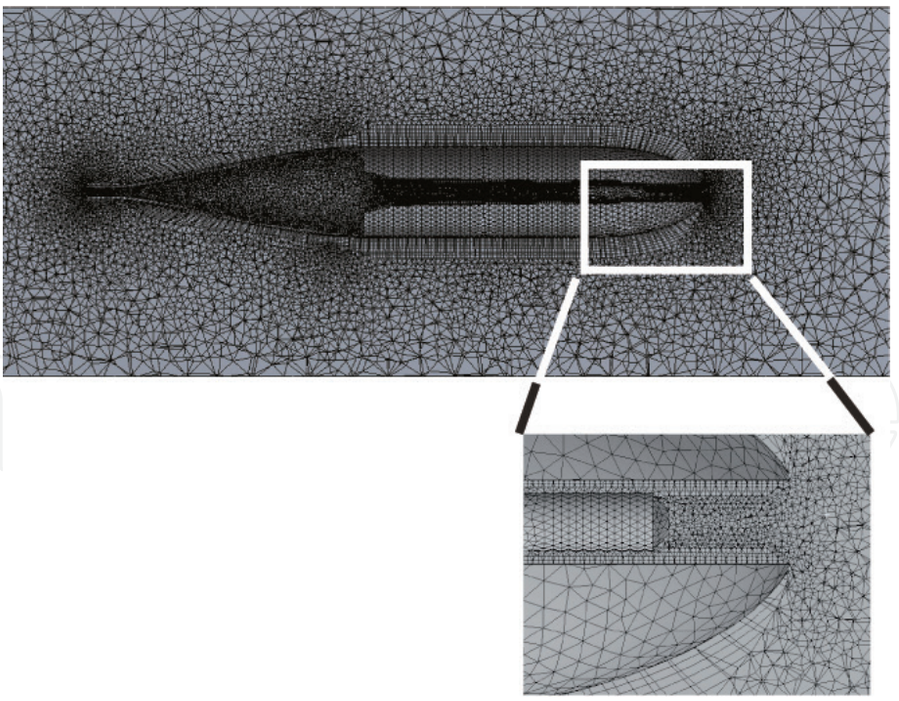

Figure 6.
Example of elements that follow curved boundaries.

Even with standard personal computers which typically have 4–8 CPU cores, a significant speed up time can be achieved. However, parallel computing is not advised for problems with a small number of elements. As a rule of thumb, each processor core should have approximately 100,000 computational elements assigned to it in order for there to be a time savings. For problems with 100,000 elements, a single processor is typically more efficient. For problems with more than approximately 200,000 elements, two processors are recommended, and so forth.

4.4 Complex element shapes

The last technical innovation that will be discussed is the advancement in element shapes. Initially, computational elements were made with simple shapes (squares and rectangles). While rectangular elements are suitable for rectangular solution domains, they are not necessarily appropriate for more complex shapes. For example, at curved boundaries, the elements would follow the boundary using a stair-step deployment. Modern day simulation algorithms generally no longer require rectangular elements. The governing equations can be solved with arbitrarily shaped elements. As a representative example, we provide a mesh that was recently used to simulate water flow past an oceanography instrument called the eXpendable BathyThermograph (XBT) As evident from the figure, a mesh has been deployed that is able to exactly follow the curved surface of the device and accurately depict the flow region (**Figure 6**).

5. Concluding remarks

We hope this brief discuss will provide valuable context for CFD users, as they read the following chapters in this book.

Author details

Suvanjan Bhattacharyya[1*], John P. Abraham[2], Lijing Cheng[3] and John Gorman[2]

1 Department of Mechanical Engineering, Birla Institute of Technology and Science Pilani, Pilani Campus, Vidya Vihar, Pilani, Rajasthan, India

2 University of St. Thomas, St. Paul, Minnesota, USA

3 Institute of Atmospheric Physics, Chinese Academy of Sciences, Beijing, China

*Address all correspondence to: suvanjan.bhattacharyya@pilani.bits-pilani.ac.in

References

[1] Richardson, L. Weather Prediction by Numerical Process, Cambridge University Press, 1922.

[2] Kawaguti W. Numerical simulation of the NS equations for flow around a circular cylinder at Reynolds number 40. Journal of the Physical Society of Japan. 1953;8:747-753.

[3] Courant R, Friedrichs K, Lewy H. Uber die partiellen differenzen gleichungen der mathematischen physic. Mathematische Annalen. 1928:32-74

[4] Patankar S, Spalding D. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. International Journal of Heat and Mass Transfer. 1972;15:1787-1806.

[5] Patankar S. Numerical heat transfer and fluid flow, Taylor and Francis, 1980.