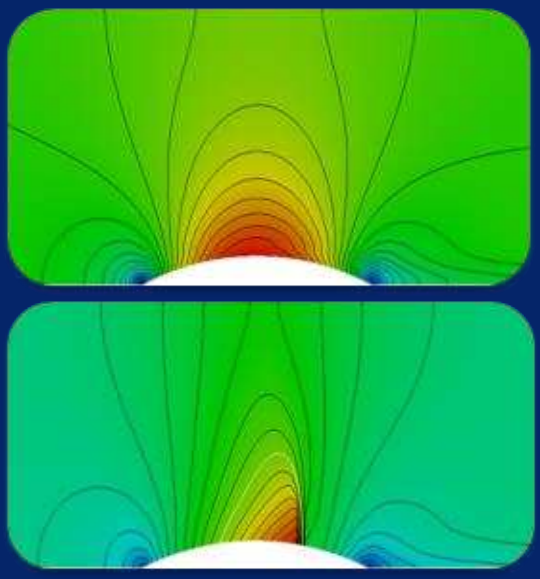
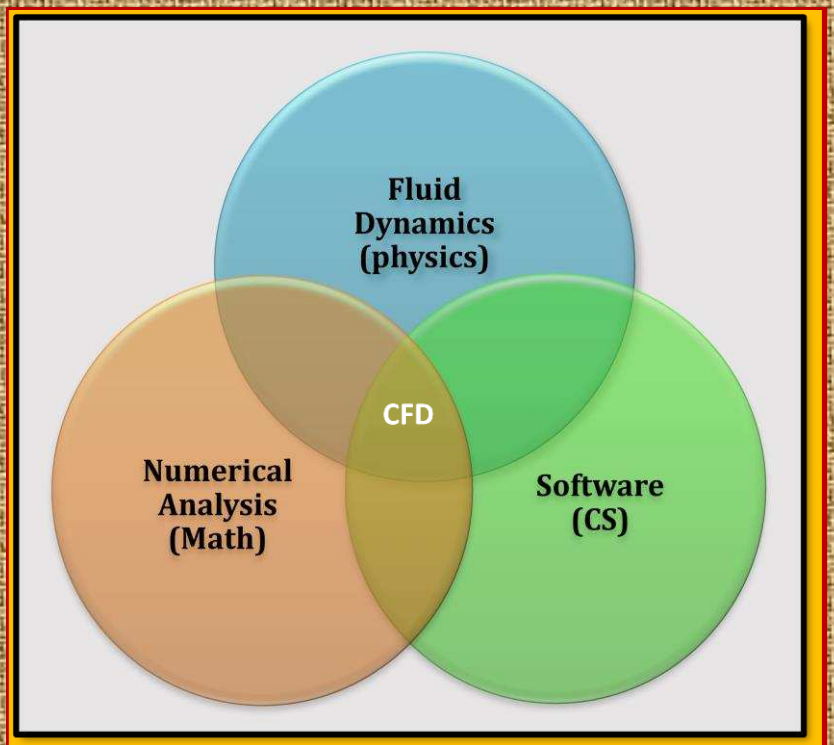
CFD Open Series/Revision 2.70

CFD Basics



Adapted & Edited by : Ideen Sadrehaghighi



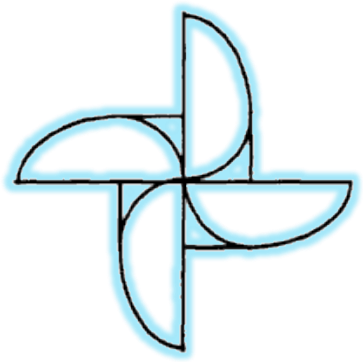
Contributions from other disciplines to CFD

ANNAPOLIS, MD

Contents

List of Tables

List of Figures



1 CFD Basics

1.1 Preliminaries

With the advent of Computational Fluid Dynamics (CFD) in recent decades, it has become an important tool in dealing with complex issues on different disciplines and engineering applications. It's relatively low cost, in relation to experimental data, and the limited availability of theoretical (analytical) data, made it an integral part of design and analysis loops. More importantly, it is relatively easy to adapt to new challenges and problems. Although in the beginning it was only used by limited number of researches and academics in aeronautical and astronautics, today it is an integral tool for wide variety of applications. But it is still considered somewhat and art and science, and in some circles, it is recommended strongly to be used in conjunction with experimental data, if available. Moreover, it is also recommended to be validated with a simplified and sub-set of theoretical results, whenever available. These colorations, known as ***Verification & Validation*** are among an intensive on-going research in CFD field. Although there are numerous definitions on CFD, all excellent in their own rights, the simplest one could be stated as ***CFD is the transformation of the governing equations from continuum domain into a discrete one.*** The PDE's are approximated by difference methods (FDE), resulting in system of algebraic equations to be solved numerically. The nature of these algebraic equations depends on the character of the problem posed by the original PDE. The equilibrium (BVP), mathematically elliptic, usually results in a system to be solved simultaneously in conjunction with satisfying the boundary conditions. While the marching problems (parabolic or hyperbolic) are usually solved one at a time, conforming to initial boundary conditions. There are three distinctive disciplines contributing to CFD analysis;

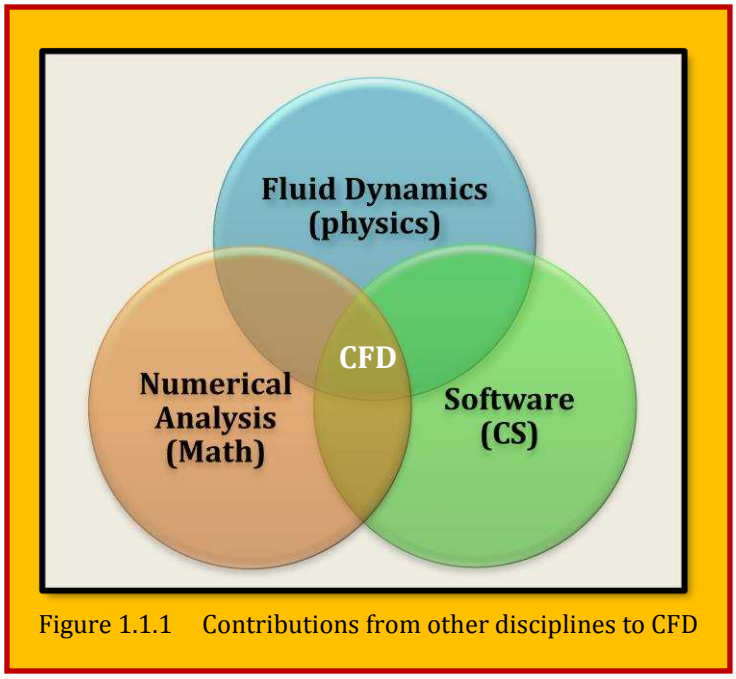


Figure 1.1.1 Contributions from other disciplines to CFD

1. The fluid dynamics which describes the physics of flow behavior as described in in any ***Fluid Dynamic*** text.
2. The mathematical side or the ***Numerical Analysis*** which transforms the fundamental governing equation from continuum into a discrete domain.
3. The ***computer science (CS)*** which does that transformation using high-level programming languages.

The question now arises that do we need expertise of three people from these disciplines in order to carry out a CFD analysis? The answer is more likely that a person that has *some proficiency* on each subsets would be sufficient. These inter-relations are depicted in **Figure 1.1.1**. Emphasis should be made here regarding the *Verification & Validation* of CFD analysis with respect to analytical or experimental data. In regard to items (1) and (2), we discussed them briefly before. We will concern our self with number (3), but restrict ourselves with general information since the aim is not to pay particular homepage to specific languages, but outline the general information available for CFD and

its particulars. The amount of information and discussion and literature available on each topic is enormous. **Figure 1.1.2** shows the three approaches to fluid dynamic problems and their strong link as these approaches do not work in isolation. All three approaches complement each other and could be used in coordination. While the experimentalist might prefer of-course the experimental approach, or the academics the analytical one, the more realist and deadline conscious analyst would probably prefer the CFD in conjunction with other two.

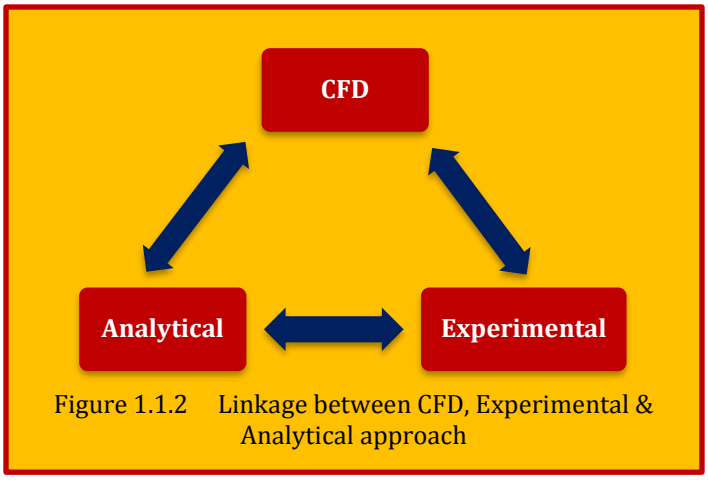


Figure 1.1.2 Linkage between CFD, Experimental & Analytical approach

1.2 Major Integration Schemes for CFD

There are several distinct methods of numerical integration of CFD as:

- Finite Volume Method (FV)
- Finite Difference Method (FD)
- Finite Element Method (FE)
- Weighted Residual (WRM)
- Spectral Methods
- Fast Fourier Transform Methods (FFT)
- Lattice Boltzmann Methods (LBM)
- Vortex Methods and other Particle Methods
- Boundary Element Method (BEM)
- Polynomial Fitting
- Integral Method

Among those, the ***Finite Difference (FD), Finite Element (FE)*** and ***Finite Volume (FV)*** methods are the most prominent ones with FD being the older one. FD solves the partial differential equations (PDEs) in the strong (or differential) form, and FE and FV solve PDEs in a weak (or integral) form [YAO et al.][1]. Each of the abovementioned methods possesses its own distinct advantages and disadvantages. Although the FV was for a long time the most popular CFD method, and that might still be the case now, the FE, with sophisticated stabilization techniques, space–time methods, and *NURBS* technology, is now in a very advanced stage.

[1] Jianyao Yao, G. R. Liu, Dong Qian, Chung-Lung Chen and George X. Xu. "*A Moving-Mesh Gradient Smoothing Method For Compressible CFD Problems*", Mathematical Models and Methods In Applied Sciences Vol. 23, No. 2 (2013) 273–305.

1.3 Anatomy of a CFD Procedure

A successful road map to a simple CFD analysis, barrowing many pitfalls which we will try to described, is modelled as following (see **Figure 1.3.1**):

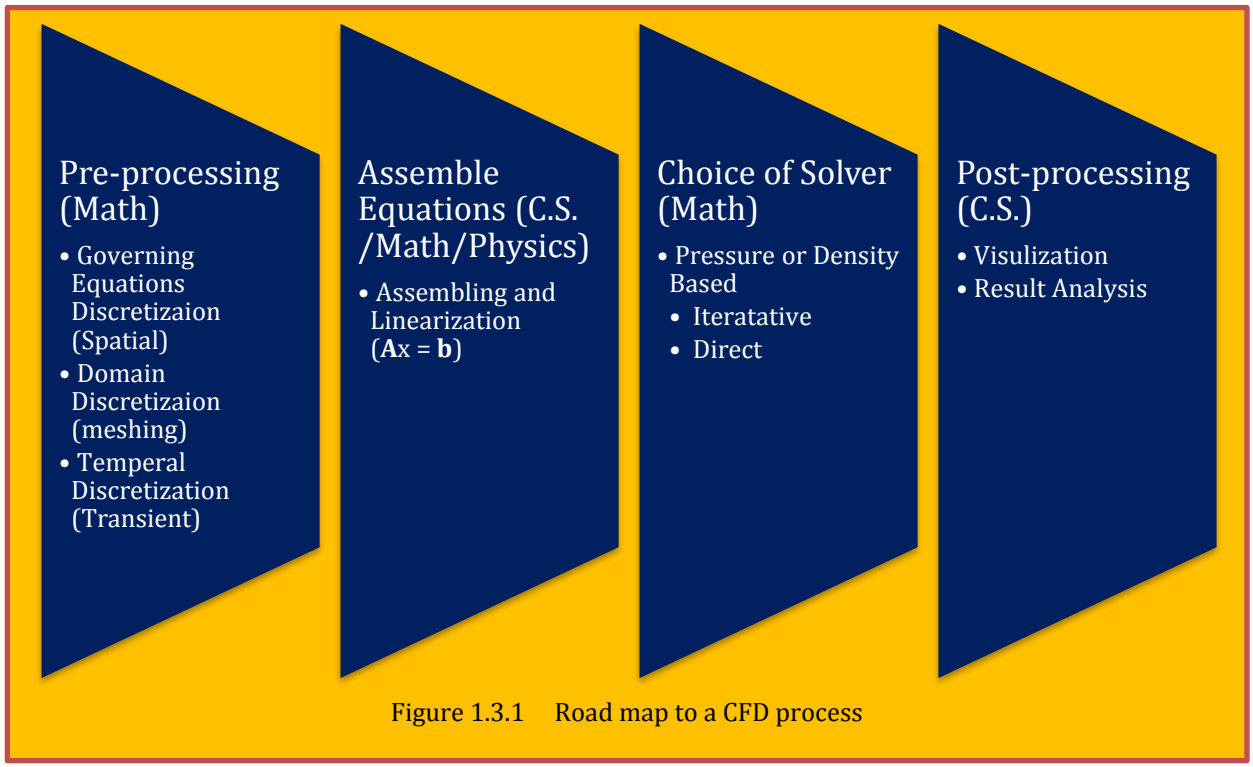


Figure 1.3.1 Road map to a CFD process

1.3.1 Domain Discretization (Gridding/Meshing)

This is probably the most the most time consuming aspect of a CFD code which could take up to 70% of engineers time. It involves two major course including

- Structured Meshing
- Unstructured Meshing

Each described in different publication[2]-[3].

1.3.2 Temporal Discretization and Time Marching

To account for transient effects, the governing equations must be discretized in time[4]. As it turns out, the temporal discretization is slightly easier to deal with than that for the spatial effects. Since the governing equation is hyperbolic/parabolic in time, the solution at time t depends upon its history and not on its future. Temporal discretization is considered as ***one sided***, while spatial is ***two sided***. Considering heat conduction where the temperature change in one point influences all spatial neighbors. Whereas time is always one way and as an example, temperature change at any point is only affected by current and past temperatures. Transient effects are usually dealt with by using a time stepping procedure, with an initial condition provided. The time dimension is divided into a set of discrete time steps, each of size Δt. The solution algorithm therefore marches forward in time, computing a solution at each time step. The spatial discretization for the time-dependent equations is identical to the steady-state case. Temporal discretization involves the integration of every term in

[2] I. Sadrehaghighi, "*Structure Meshing for CFD*", CFD Open Series, Patch 2.35.
[3] I. Sadrehaghighi, "*Unstructured Meshing for CFD*", CFD Open Series, Patch 2.60.
[4] CFD on Line discussion

the differential equations over the time step Δt. The integration of transient effects takes several forms, each yielding a different accuracy. For simplicity, we express the time dependent transport of a scalar ϕ

$$\frac{\partial(\rho\Phi)}{\partial t} = L(\Phi)$$

Eq. 1.3.1

Where the function L is an operator that incorporates all of the non-transient terms, namely: diffusion, convection, and source terms. Integrating the above equation over a control volume yields

$$\int_V \frac{\partial(\boldsymbol{\rho\phi})}{\partial t}\, dV = \int_V \mathbf{L}(\boldsymbol{\phi}) dV = \mathbf{L}(\boldsymbol{\phi})$$

Eq. 1.3.2

After the spatial discretization has been performed using the techniques described in the other sections, we obtain where L denotes the spatial discretization operator (the discretized diffusion, convection, and source terms), and V denotes the volume. At this point, we make an important observation. ***The temporal discretization of the transient term (LHS) need not be the same as that of the discretized diffusion, convection and source terms (RHS)***. Each term can be treated differently yield different accuracies. We are now ready to perform the transient discretization. In the framework of the finite volume method, there are various methods that can be used to perform this task, the most popular of which are the ***Euler Implicit, Crank-Nicolson, Fully Implicit schemes, Runge Kutta Method*** and the *2nd order Backward Difference Formula (BDF2).* The temporal discretization of the diffusion, convection, and source terms will be presented first, followed by the methods used for the transient term. We assume the values at a given control volume are known at an initial time *t* and we are interested in obtaining the values at time t+Δt. This method states that the time integral of a given variable is equal to a weighted average between existing and future values. For a given control volume ϕ, the RHS of the general discretized equation L(ϕ) contains terms involving values at ϕ and its neighboring control volumes. The temporal discretization is carried out through an integration over time of the RHS where each unknown is involved in the process. An integration over time of the RHS where each unknown is involved in the process.

$$\int_t^{t+\Delta t} \varphi\, dt \approx [f\, \varphi^{t+\Delta t} + (\mathbf{1} - \mathbf{f})\, \varphi^{t}]\, \Delta t$$

Eq. 1.3.3

Where f is a weighing factor between 0 and 1. The following holds:

- f = 0 results in the fully explicit scheme
- f = 1 results in the fully implicit scheme
- f = 0.5 results in the Crank-Nicolson scheme

The product of most frequently schemes are described below. Other notable schemes are time spilt ***McCormack*** method, ***Lax-Wendorff***, and ***ADI***, as well as many more as described in various text books.

1.3.3 Runga-Kutta Method

In numerical analysis, the ***Runge–Kutta methods*** are a family of implicit and explicit iterative methods, which includes the well-known routine called the ***Euler Methods*** ($y_{n+1} = y_n + hf(x_n,y_n)$), used

in temporal discretization for the approximate solutions of ordinary differential equations[5]. These methods were developed around 1900 by the German mathematicians [C. Runge] and [M. W. Kutta].

1.3.3.1 Explicit Treatment

The family of explicit Runge–Kutta methods is a generalization of the method mentioned above. It is given by:

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i \qquad \begin{cases} k_1 = f(t_n, y_n), \\ k_2 = f(t_n + c_2 h, y_n + h(a_{21}k_1)), \\ k_3 = f(t_n + c_3 h, y_n + h(a_{31}k_1 + a_{32}k_2)) \\ \dots. \\ k_s = f(t_n + c_s h, y_n + h(a_{s1}k_1 + a_{s2}k_2 + \dots\dots + a_{s,s-1}k_{s-1})) \end{cases}$$

Eq. 1.3.4

To specify a particular method, one needs to provide the integer s (the number of stages), and the coefficients a_{ij} (for $1 \le j < i \le s$), b_i (for i = 1, 2,..., s) and c_i (for i = 2, 3,.., s). The matrix $[a_{ij}]$ is called the ***Runge–Kutta matrix***, while the b_i and c_i are known as the weights and the nodes defined by

$$\sum_{j=1}^{i-1} a_{ij} = c_i \quad \text{for} \quad i = 2,\dots\dots,s$$

Eq. 1.3.5

By far the most often used is the classical ***4th order Runge-Kutta formula RK4,*** which has a certain sleekness of organization about it[6]. Therefore, *fourth-order Runge-Kutta* method requires four evaluations of the right hand side per each step size h as:

$$y_{n+1} = y_n + h\, f(x_n, y_n) \qquad \left\{ \begin{array}{l} k_1 = h\, f(x_n, y_n) \\ k_2 = h\, f\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \\ k_3 = h\, f\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \\ k_4 = h\, f(x_n + h, y_n + k_3) \end{array} \right\}$$

$$y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)$$

Eq. 1.3.6

1.3.3.2 Implicit Runge-Kutta Methods

All Runge-Kutta methods mentioned up to now are explicit methods. Explicit Runge–Kutta methods are generally unsuitable for the solution of stiff equations because their region of absolute stability is small; in particular, it is bounded. This issue is especially important in the solution of partial differential equations. The instability of explicit Runge–Kutta methods motivates the development of implicit methods. An implicit Runge–Kutta method has the form

---

[5] "*Runge–Kutta methods*", From Wikipedia, the free encyclopedia.

[6] "*Numerical Recipes In C: The Art of Scientific Computing*", (ISBN 0-521-43108-5), Copyright (C) 1988-1992 by Cambridge University Press.

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i \qquad \text{where} \quad k_i = f\left(t_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij} k_i\right)$$

Eq. 1.3.7

The difference with an explicit method is that in an explicit method, the sum over j only goes up to i-1. The coefficient a_{ij} of an explicit method is lower triangular. In an implicit method, the sum over j goes up to *s* and the coefficient matrix is not triangular. The consequence of this difference is that at every step, a system of algebraic equations has to be solved. This increases the computational cost considerably. If a method with *s* stages is used to solve a differential equation with *m* components, then the system of algebraic equations has *ms* components. This can be contrasted with implicit linear multistep methods (the other big family of methods for ODEs): an implicit *s*-step linear multistep method needs to solve a system of algebraic equations with only *m* components, so the size of the system does not increase as the number of steps increase.

1.3.3.3 Transient Simulation

Alternatively, the transient (un-steady) scheme could be visualized as series of quasi-Steady-State solutions to be advanced temporally until pre-defined run time reached (dual time stepping), depicts such solution propagation though time steps[7]. Unlike steady-state where the solution is advanced until residuals fall below pre-defined tolerance level, in transient analysis, there is really no pre-set convergence criteria defined expect the duration time. The transient residuals although should fall but could exhibit certain small oscillations, characteristic of unsteady flow. As long as that oscillation are small enough to limit the temporal approximation errors. In fact the solution sequence which outlined in **Figure 1.3.2 (left)** could be translated to **Figure 1.3.2 (right)** which displays residual plots for a transient flows.

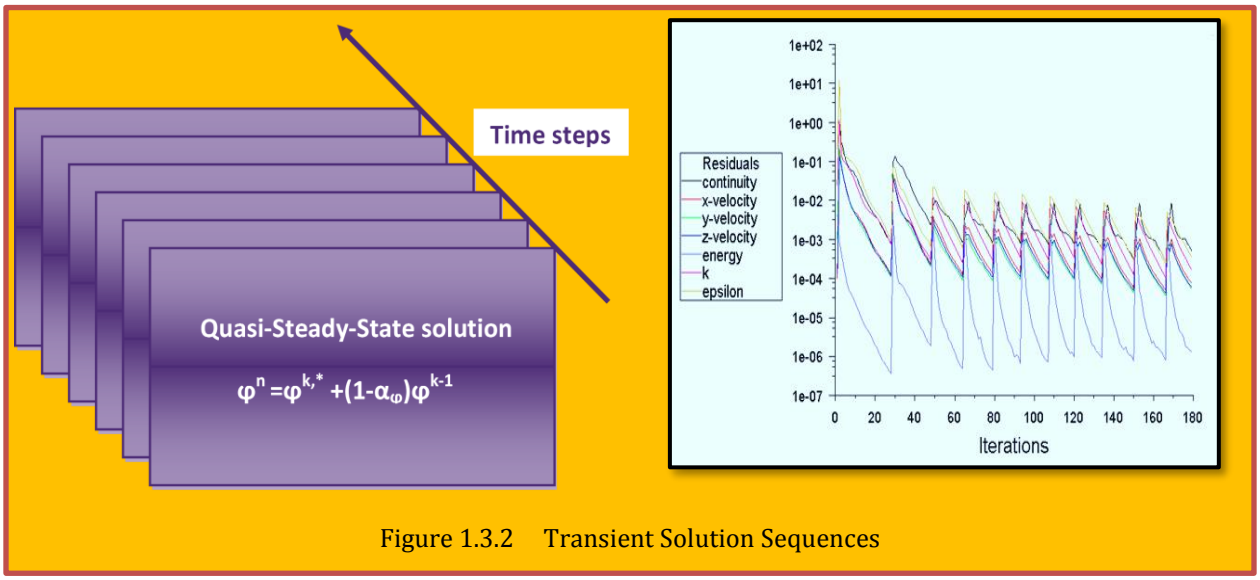


Figure 1.3.2 Transient Solution Sequences

1.3.3.4 Time Step Size

The choice of time step is of course most important and should be elected carefully. It is restricted by CFL number on explicit methods. For fully implicit methods, although in principal it could be any value, in practice other considerations impose limit as previously indicated. This limit, produces in terms of $\Delta t < h/v$ where h is the size of smallest cell, and v is typical characteristic flow velocity.

---

[7] Bakker André, Applied Computational Fluid Dynamics; Solution Methods; 2002.

Usually there are options of static (constant) step size, or more robust dynamic evaluation as dictated internally by the solver.

1.3.3.5 Steady-State vs. Transient Run

"How do I know in advance whether to perform a steady-state or an unsteady CFD simulation?" The simple answer is, "you don't know", so it is imperative to provide some help on when to use unsteady (also known as transient or time-dependent) simulations[8]. When you run a steady-state simulation it uses an iterative scheme to progress to convergence. If you see persistent oscillations in the residuals plot (solver diagnostics) or oscillations in a key monitor, such as a drag force monitor, with increasing iterations then that's a good indicator the flow may be unsteady (transient) and the simulation needs to be run as an unsteady. When you are unsure as to whether your simulation is unsteady or steady-state it is always worth running a steady-state simulation first because it typically takes an order of magnitude less CPU time to complete. If the steady-state simulation is sufficient then you'll save a lot of time over running an unsteady simulation. As an example let us examine the residual and force monitors for an unsteady simulation of ***vortex shedding behind a cylinder*** (see **Figure 1.3.3 (a)**) and compare them with those generated by the same simulation running, incorrectly, in steady-state mode. Simply put, the rule is that the flow is transient unless proven otherwise, as many flows in nature are transient.

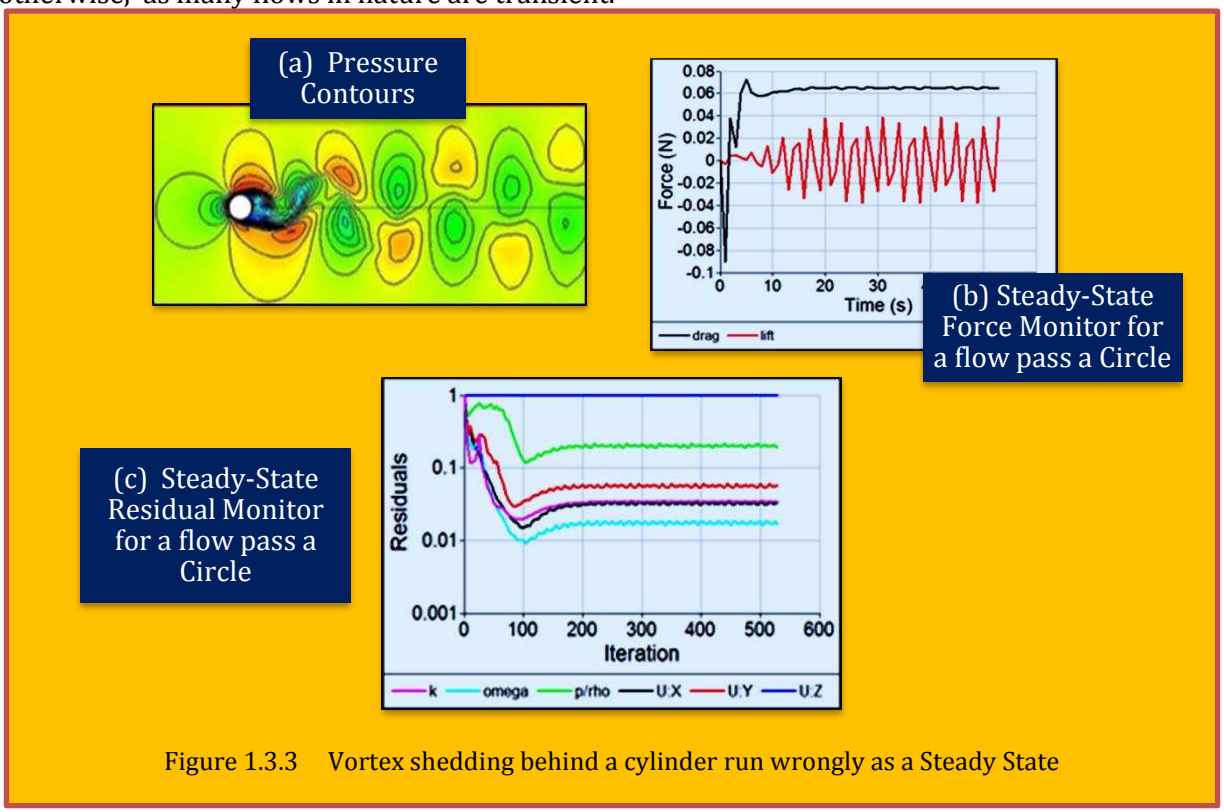


Figure 1.3.3 Vortex shedding behind a cylinder run wrongly as a Steady State

1.3.3.6 Case Study 1 - Vortex Shedding Behind a Cylinder - Ran as Steady State (Wrong!)

Remember that vortex shedding behind a cylinder is unsteady as described in the previous section, so it is interesting to see what happens when you try to run the same simulation in steady-state mode. Clearly after only 20 updates (200 iterations) there is something wrong, as expected, indicated by the widely oscillating lift force. For a steady-state simulation you'd expect some oscillations in the lift force values when you first start the simulation and then see them damp out to a fixed value with

[8] "*Steady-State or Unsteady CFD Simulation*?" Symscape, Computational Fluid Dynamics Software for All.

increasing iterations. Something is also wrong, as expected, with the residuals. For a steady-state simulation you should see all residuals reduce relatively smoothly toward or below $1e^{-3}$ with increasing iterations (see **Figure 1.3.2 (b)-(c)**).

1.3.3.7 Case study 2 - Double Sided Membrane - Ran as Steady State (Right!)

For a steady-state simulation the residuals and force profile should be something similar to those in the tutorial "Flow Over a Double-Sided Membrane". Note the converged lift and drag force values with increasing iterations[9]. The relatively smooth reduction below $1e^{-3}$ in all the residuals with increasing iterations. In conclusion, if you run a steady-state simulation for an unsteady flow then you will get poor results, but as just pointed out there is usually a good indication from the steady state results that an unsteady simulation was necessary. However, if you find that that your simulation has a steady-state then you can avoid the unnecessary expense of performing an unsteady simulation (**Figure 1.3.4**).

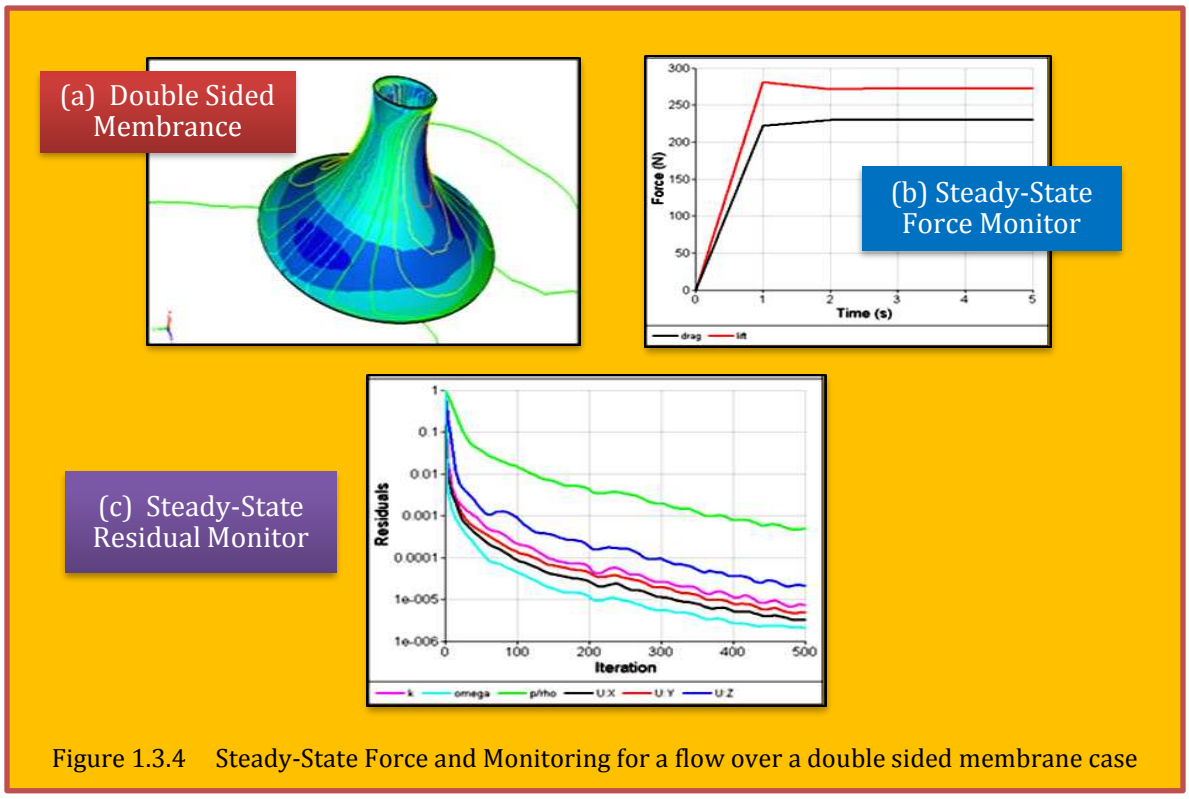


Figure 1.3.4 Steady-State Force and Monitoring for a flow over a double sided membrane case

1.3.3.8 Case Study – Evolution of the Wake of 3 Inline Square Prisms

Citation : *Zheng, Qinmin and Alam, Md. Mahbub. "Evolution of the wake of three inline square prisms", Phys. Rev. Fluids, 4, 10, 2019, American Physical Society, DOI : 10.1103/PhysRevFluids.4.104701, https://link.aps.org/doi/10.1103/PhysRevFluids.4.104701*

Expanding on that, a study of the flow around three tandem square prisms, conducted by [Zheng & Mahbub Alam][10], may provide us a better understanding of complicated flow physics related to multiple closely spaced structures. In this paper, a numerical investigation on the flow around three tandem square prisms at a Reynolds number Re =($U_\infty W/\nu$) = 150 is conducted for L/W = 1.2–10.0,

[9] "*Steady-State or Unsteady CFD Simulation?*" Symscape, CFD Software for All.
[10] Qinmin Zheng and Md. Mahbub Alam, "*Evolution of the wake of three inline square prisms*", Physical Review Fluids 4, 104701 (2019).

where U_∞ is the freestream velocity, W is the prism width, L is the prism center-to-center spacing, and ν is the kinematic viscosity of the fluid. Extensive analyses are done of flow structures, ***Strouhal numbers St,*** fluid forces, and vorticity, velocity, and pressure fields. Four distinct flow regimes and their ranges are identified, viz., single bluff-body flow

- ***Regime I*** : 1.2 ≤ L/W < 3.0), alternating reattachment flow
- ***Regime II*** : 3.0 < L/W < 4.3), synchronous co-shedding flow
- ***Regime III*** : 4.3 < L/W < 7.3) and asynchronous Co-shedding flow
  - ***Regime III A*** : 4.3 < L/W < 5.1) and dual St flow
  - ***Regime III B*** : 5.1 < L/W < 7.3).
- ***Regime IV*** : 7.3 < L/W ≤10.0).

The dependence on L/W of fluctuating and time-mean fluid forces and St of the three prisms in each regime is studied in detail and connected to the flow structures. A secondary vortex street following the primary vortex street is observed for Regimes III B and IV. The detailed physics of the evolution of the primary vortex street to the secondary is imparted. The inherent frequency associated with the secondary vortex street is smaller than that with the primary. The evolution process of the primary vortex street to the secondary leads to a tertiary frequency. Dynamic mode decomposition analysis is proposed for the first time as a useful and quantitative tool to identify and quantify the secondary vortex street and its onset position. (See **Figure 1.3.5** and [Zheng & Mahbub Alam][11]).

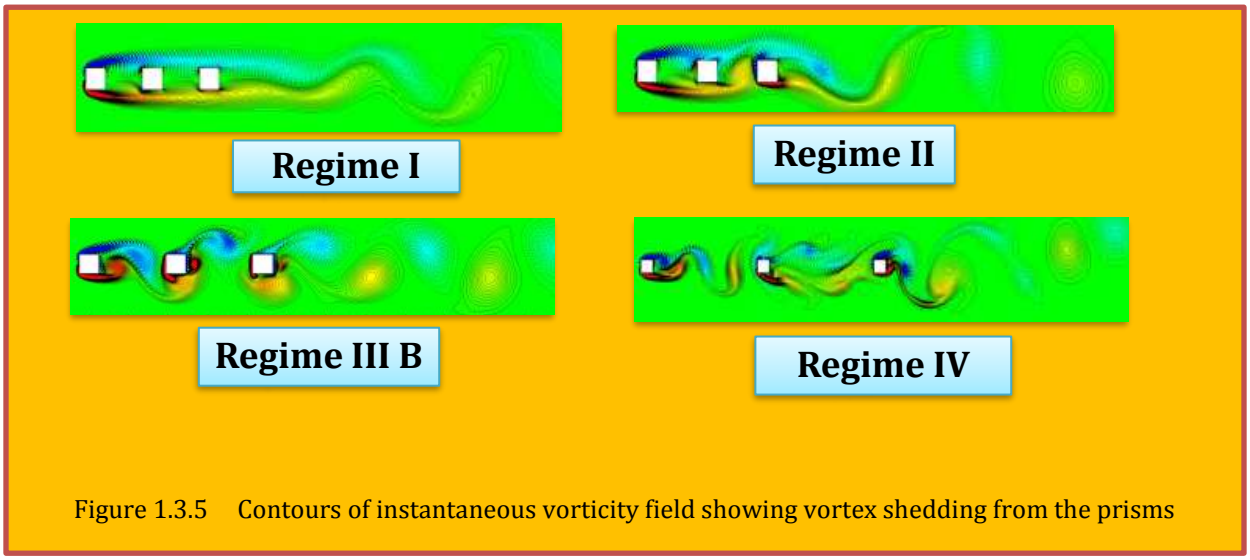


Figure 1.3.5 Contours of instantaneous vorticity field showing vortex shedding from the prisms

1.3.3.9 Concept of Dual Time Stepping

When solving an unsteady fluid flow, inadvertently, two time steps come into the picture. One a real physical time integration (outer loop), the other is pseudo timing mainly for iterative schemes on spatial coordinates (inner looping). The idea is to use an implicit scheme with a large stability region and to solver the implicit equations at each time step, while by inner iterations using an accelerated evolution scheme in ***artificial time or pseudo-time*** [Jameson][12] as :

$$\frac{d}{dt}\int_{\text{Cell}} \mathbf{W}\,dV + \oint_{\text{Cell Boundary}} \mathbf{n_i}.\mathbf{f_i}.dS = 0 \quad \text{or} \quad V\underbrace{\frac{d\mathbf{W}}{dt}}_{\text{Real Time}} + \underbrace{\mathbf{R(W)}}_{\text{Pseudo Time}} = 0$$

[11] Qinmin Zheng and Md. Mahbub Alam, "Evolution of the wake of three inline square prisms", Physical Review Fluids 4, 104701 (2019).

[12] Antony Jameson, "*Time Integration Methods In Computational Aerodynamics*", 2003.

Eq. 1.3.8

Where **w** now denotes the average value of the state in the cell. V is the cell volume, or in the two dimensional case, the cell area. **R(w)** is the residual resulting from the space discretization. Applications to external aerodynamic typically use grids in which the cell area or volume varies by many orders of magnitude between the body and the far field, and this is a principal reason for using implicit schemes for time accurate simulations.

1.3.3.10 Backward Differencing Formula (BDF)

Introducing superscripts *n* to denote the time level, the second order ***Backward Difference Formula (BDF2)*** for time integration is

$$\frac{V}{dt}\underbrace{\left(\frac{3}{2}\mathbf{W}^{n+1}-2\mathbf{W}^{n}+\frac{1}{2}\mathbf{W}^{n-1}\right)+\mathbf{R}(\mathbf{W}^{n+1})}_{\mathbf{R}^{*}(\mathbf{W})}=0$$

Eq. 1.3.9

In the dual time stepping scheme this equation is solved by marching the equation:

$$\frac{d\mathbf{W}}{d\tau}+\mathbf{R}^{*}(\mathbf{W})=0$$

Eq. 1.3.10

In solving **Eq. 1.3.3** one is free to use every available acceleration technique for fast steady state solutions without regard for time accuracy. It was shown that ***Multigrid*** techniques can be very effective for this purpose. The dual time stepping approach has been quite widely adopted, particularly in conjunction with the (BDF2) scheme. Recently there has been considerable interest in whether ***Implicit Runge-Kutta*** schemes can achieve better accuracy for a given computational cost than the ***Backwards Difference Formulas***. Most of the studies to date have focused on Diagonal Implicit Runge-Kutta (DIRK) schemes, sometimes called semi-implicit schemes, in which the stages may be solved successively.

1.3.4 Discretization (Spatial) Schemes for Governing Equations

Numerical schemes are basically methods used to interpolate flow field variables across the computational domain (***spatial discretization***). The need for such schemes arise because the system of equations resulting from applying the CFD on the computational grid poses a boundary value problem. In such problem, the values of the flow field variables are known on the boundary cells of the grid, and unknown in other cells[Khalid M. Saqr][13]. To account for transient effects, the governing equations must be discretized in time (***temporal discretization***)[14]. As it turns out, the temporal discretization is slightly easier to deal with than that for the spatial effects. Since the governing equation is hyperbolic/parabolic in time, the solution at time t depends upon its history and not on its future. Temporal discretization is considered as ***one sided***, while spatial is ***two sided***.

1.3.4.1 Desired Properties in Discretization Schemes

In general, any discretization scheme should have the following properties:

- ***Conservativeness*** - global conservation of properties must be insured, i.e., flux consistency in control volume *(Flux IN = Flux OUT).*
- ***Boundedness*** - values predicted by scheme should be within realistic bounds. For linear problems (Heat conduction), those would be the boundary values. For non-linear problems, the values within the domain could be outside of boundary values.

---

[13] Khalid M. Saqr, "*Practical Introduction to Computational Fluid Dynamics*", Course Notes, 2nd Edition, Jan. 2017.
[14] CFD on Line discussion

- ***Trans-portiveness*** - As diffusion works in all direction, but convection only in flow direction, the numerical scheme should recognize the flow direction since it effects the balance of convection vs. diffusion (Peclet Number).

The manner in which the convective and diffusive fluxes are expressed in terms of nodal φ values is one of the key factors determining accuracy and stability, for both steady-state and transient calculations. At the high Reynolds numbers often encountered in practice, the choice of convective flux approximation is particularly important.

1.3.4.2 References

[1] D. Anderson, J., Tannehill, R., Pletcher, "*Computational Fluid Mechanics and Heat Transfer*", ISBN 0-89116-471-5 – 1984.

[2] Ali Hasan Ali , Ahmed Shawki Jaber, Mustafa T. Yaseen, Mohammed Rasheed, Omer Bazighifan , and Taher A. Nofal, "*A Comparison of Finite Difference and Finite Volume Methods with Numerical Simulations: Burgers Equation Model*", Hindawi, Volume 2022, Article ID 9367638.

[3] APMA0160 (A.Yew) Spring 2011.

1.3.5 Some Issues Related to Finite Differencing Discretization

The idea of finite differencing steams from the definition of derivatives as

$$\frac{\partial u(x,y)}{\partial x} = \lim_{\Delta x \to 0} \frac{u(x+\Delta x, y) - u(x,y)}{\Delta x}$$

Eq. 1.3.11

which in turns associates to some questions such as ***truncation error, consistency, stability***, and ***convergence*** of approximations to PDE. Following sections will address those problems regarding PDE approximation.

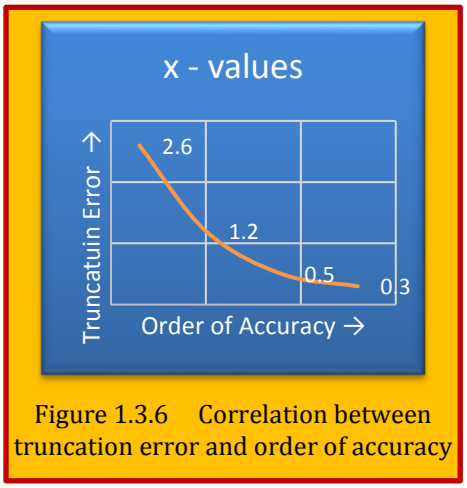


Figure 1.3.6 Correlation between truncation error and order of accuracy

1.3.5.1 Truncation Error

Assuming u to be continuous, this will be a reasonable approximation to the derivative. With the aid of Taylor's series expansion for expansion of u(x + Δx) could be expressed as

$$u(x+\Delta x, y) = u(x,y) + \frac{\partial u}{\partial x}\Delta x + \frac{\partial^2 u}{\partial x^2}\frac{(\Delta x)^2}{2!} + \ldots.$$

Eq. 1.3.12

Rearranging this and dividing by Δx,

$$\frac{\partial u}{\partial x} = \frac{u(x+\Delta x, y) - u(x,y)}{\Delta x} - \underbrace{\frac{\partial^2 u}{\partial x^2}\frac{\Delta x}{2}}_{\text{Truncation Error} = \mathcal{O}(\Delta x)} \ldots$$

Eq. 1.3.13

Where the truncation error is the difference between ***PDE*** and ***FDE*** representation and characterized by using the order notation. It is inversely related to the order of accuracy for the equations as depicted in **Figure 1.3.6** and would be an extremely important criteria in accuracy of discretized equation as it is directly related to the stability and accuracy consideration. Therefore, for **Eq. 1.3.13** it could be said that the forward finite difference representation here is 1st order accurate in space (Δx). For most practical applications, 2nd order accuracy would be sufficient as by increasing the order of accuracy, the CPU cost would also increase. Free temperature profile of a jet flow would be

an excellent example in that case (see **Figure 1.3.6**). In order to be acceptable, the difference representation needs to meet the conditions of consistency and stability, to be discussed next.

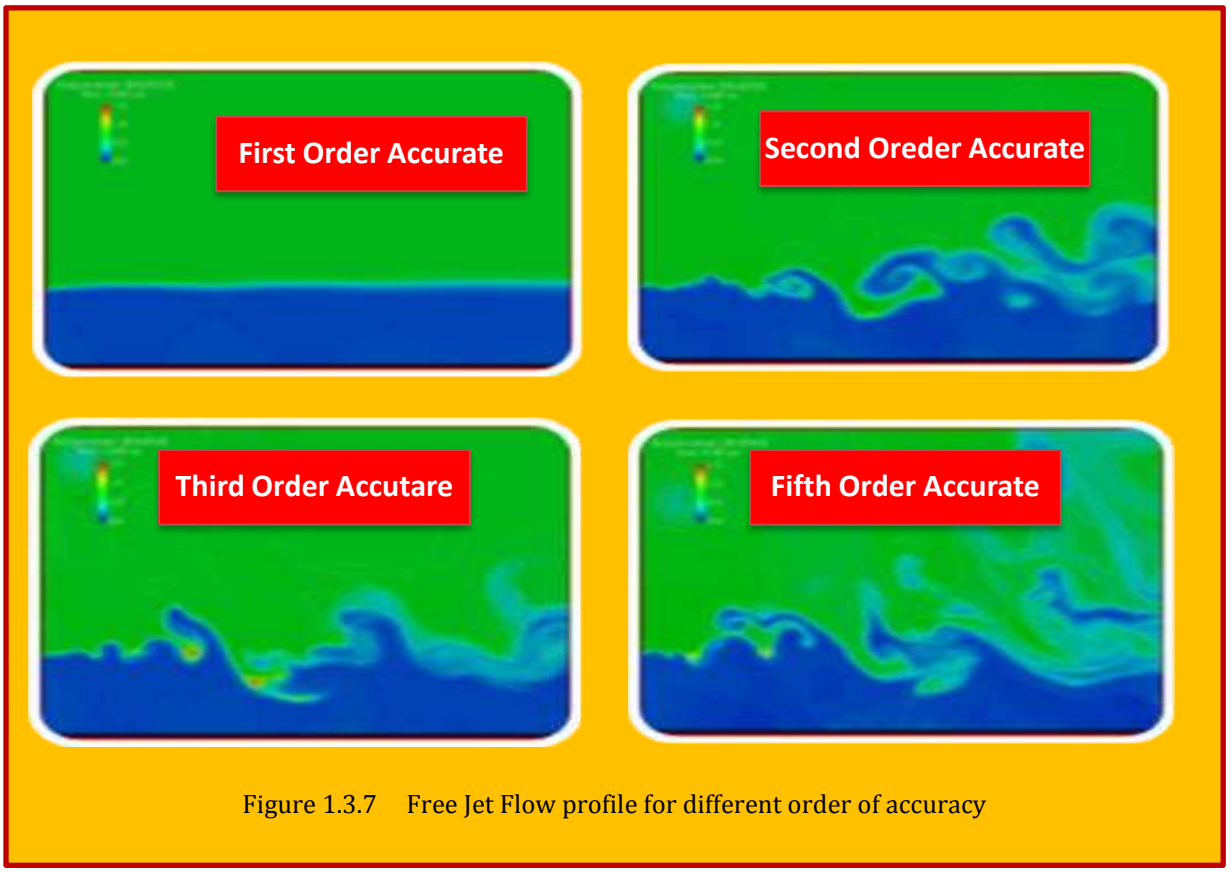


Figure 1.3.7 Free Jet Flow profile for different order of accuracy

1.3.5.2 Consistency

Consistency deals with extend on which the finite difference equations approximate the PDEs. It is directly linked to truncation error by means of showing that in the limit, as mesh been refined infinitely, the difference between the PDE and FDE vanishes, i.e. lim mesh → 0 (PDE-FDE) = 0 as lim mesh → 0 (TE) = 0. Therefore, for consistency to be satisfied, on the limit, the truncation error should vanish. An important question concerning computational solutions is what guarantee can be given that the computational solution will be close to the exact solution of the partial differential equation(s) and under what circumstances the computational solution will coincide with the exact solution. The second part of this question can be answered (superficially) by requiring that the approximate (computational) solution should converge to the exact solution as the grid spacings At, Ax shrink to zero. However, convergence is very difficult to establish directly so that an indirect route, as indicated in **Figure 1.3.7** is usually followed. The indirect route requires that the system of algebraic equations formed by the discretization process should be consistent with the governing partial differential equation(s). Consistency implies that the discretization process can be reversed, through a Taylor series expansion, to recover the governing equation(s). In addition, the algorithm used to solve the algebraic equations to give the approximate solution, T, must be stable. Then the pseudo-equation [1]:

$$\text{Consistancy} + \text{Stability} = \text{Convergence} \tag{Eq. 1.3.14}$$

is invoked to imply convergence. The conditions under which **Eq. 1.3.14** can be made precise are given by the ***Lax equivalence theorem***. It is very difficult to obtain theoretical guidance for the behavior of the solution on a grid of finite size. Most of the useful theoretical results are strictly only applicable in the limit that the grid size shrinks to zero. However the connections that are established between convergence, consistency and stability are also qualitatively useful in assessing computational solutions on a finite grid.

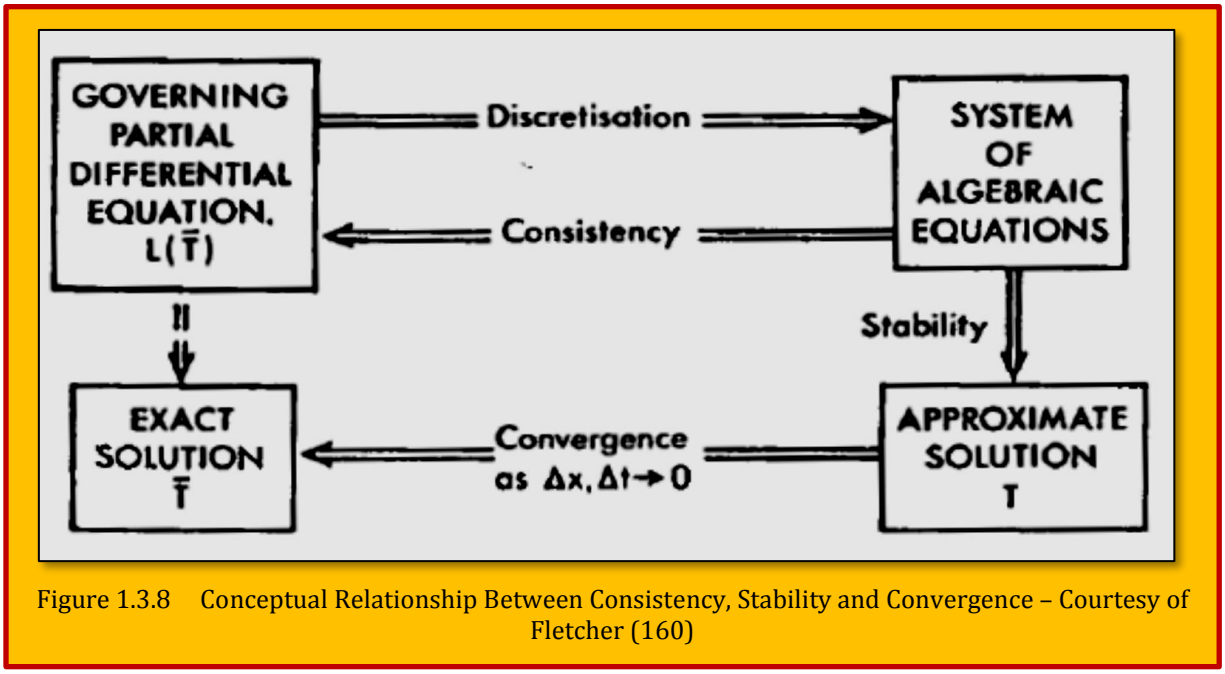


Figure 1.3.8 Conceptual Relationship Between Consistency, Stability and Convergence – Courtesy of Fletcher (160)

1.3.5.3 Stability and Effect of CFL Number

The stability concept is a bit more involved than consistency and in a restrict sense only applicable to marching problems. A stable numerical scheme is the one that errors do not propagate through the system. It devises a relation and imposes a restriction between mesh and time step sizes, defined as CFL number. There are two prominent procedures, namely, the ***Von Neumann*** and ***Fourier*** for stability analysis. To better understand, a simple 1D stability criteria could be devised where CFL = a $\Delta t/\Delta x$ with α being a positive constant, as being presented in **Figure 1.3.8**. On the left (a), where CFL > 1, Q lies ***outside*** of computational domain of influence, therefore, unstable. But on the right (b), where CFL ≤ 1, Q lies ***inside*** of computational domain of influence, therefore, stable. It is obvious that the size of mesh, Δx, is crucial in stability here and by reducing it (finer mesh). Hence, the stability constrain is satisfied, while keeping the same Δt. It is safe to assume that the numerical stability requirement for many explicit numerical methods for solving hyperbolic PDE's, is CFL condition which is

$$\mathrm{CFL} = \left| a \frac{\Delta t}{\Delta x} \right| \leq 1$$

Eq. 1.3.15

Reducing the Δt, if insisting in keeping the same Δx, would achieve that also. Which correlates to smaller time steps in transient cases. ***Thus, it needs either a finer step sizes, or smaller time steps (preferably both for transient cases), on explicit methodologies for stable conditions***. It is known that the best results for hyperbolic systems using the most common explicit methods are obtained with CFL near unity (see **Figure 1.3.8**). A more general 3D CFL number using the empirical formula [Tannehill et al., 1975] defined as:

$$\Delta t \le \frac{\alpha(\Delta t)_{CFL}}{1+2/Re_{\Delta}} \quad \text{where} \quad (\Delta t)_{CFL} < \left[\frac{|u|}{\Delta x}+\frac{|v|}{\Delta y}+\frac{|w|}{\Delta z}+a\sqrt{\frac{1}{(\Delta x)^2}+\frac{1}{(\Delta y)^2}+\frac{1}{(\Delta z)^2}}\right]^{-1}$$

$$\text{with} \quad Re_{\Delta} = \text{Min}\,(Re_{\Delta x}, Re_{\Delta y}, Re_{\Delta z}) \quad Re_{\Delta x} = \frac{\rho|u|\Delta x}{\mu} \quad Re_{\Delta y} = \frac{\rho|v|\Delta y}{\mu} \quad Re_{\Delta z} = \frac{\rho|w|\Delta z}{\mu}$$

$$\text{and} \quad a = (\gamma p/\rho)^{0.5} \quad , \quad \text{safty factor} \ \alpha \approx 0.9$$

Eq. 1.3.16

1.3.5.4 Convergence

A solution of the algebraic equations (**Figure 1.3.9**) which approximate a given partial differential equation is said to be convergent if the approximate solution approaches the exact solution of the partial differential equation for each value of the independent variable as the grid spacing tends to zero. Thus we require

$$T_j^n \to \overline{T}(x_j, t_n) \quad \text{as} \ \Delta x, \Delta t \to 0$$

Eq. 1.3.17

The difference between the exact solution of the partial differential equation and the exact solution of the system of algebraic equations is called the solution error, denoted by e^n_j; that is

$$e_j^n = \overline{T}(x_j, t_n) - T_j^n$$

Eq. 1.3.18

The exact solution of the system of algebraic equations is the approximate solution of the governing partial differential equation. The exact solution of the system of algebraic equations is obtained hen no numerical errors of any sort, such as those due to round-off, are introduced during the computation. The magnitude of the error, e^n_j, at the (j , n)-th node typically depends on the size of the grid spacings, Dx and Dt, and on the values of the higher-order derivatives at that node, omitted from the finite difference approximations to the derivatives in the given differential equation. Proof that a solution to the system of algebraic equations converges to the solution of the partial differential equation is generally very difficult, even for the simplest cases.

1.3.5.4.1 Numerical Convergence

For the equations that govern fluid How, convergence is usually impossible to demonstrate theoretically. However, for problems that possess an exact solution, like the diffusion equation, it is possible to obtain numerical solutions on a successively refined grid and compute a solution error.

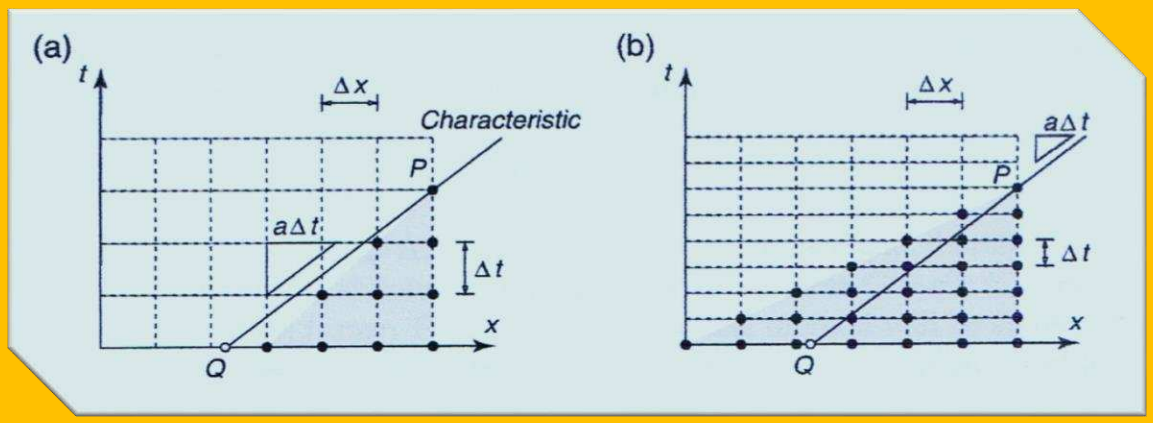


Figure 1.3.9 1D Stability Analysis Based on CFL Number; (a) CFL > 1 Unstable; (b) CFL ≤ 1 Stable

Convergence implies that the solution error should reduce to zero as the grid spacing is shrunk to zero. The solutions have been obtained on successively refined spatial grids, Ax = 0.2, 0.1, 0.05 and 0.025. The corresponding rms errors are shown in Table 1.3.1 for s = 0.50 and 0.30. It is clear that the rms error reduces like Δx^2 approximately. Based on these results it would be a reasonable inference that refining the grid would produce a further reduction in the rms error and, in the limit of Δx (for fixed s) going to zero, the solution of the algebraic equations would converge to the exact

	rms error			
$s=\alpha\Delta t/\Delta x^2$	$\Delta x=0.2$	$\Delta x=0.1$	$\Delta x=0.05$	$\Delta x=0.025$
0.50	1.658	0.492	0.121	0.030
0.30	0.590	0.187	0.048	0.012

Table 1.3.1 Solution Error (rms) Reduction with Grid Refinement – Courtesy of Fletcher

solution. The establishment of numerical convergence is rather an expensive process since usually very fine grids are necessary. As s is kept constant in the above example the timestep is being reduced by a factor of four for each halving of Dx. In **Table 1.3.1** the solution error is computed at t = 5000 s. This implies the finest grid solution at s = 0.30 requires 266 time steps before the solution error is computed. [1].

1.3.5.5 References

[1] C. A. J. Fletcher, "*Computational Techniques for Fluid Dynamics 1 - Fundamental and General Techniques*", 2nd Ed., Springer.

1.3.6 Linear System of Equations

It is a well-known fact that the governing equations of fluid flow are highly nonlinear field functions[4]. In order to make these equations suitable to be solved by computers, they have to be ***linearized***. This is because binary computers can only perform logical operations, such as addition, subtraction, multiplication, and so on. The outcome is a system of linear equations that can be solved given a solution domain space (the fluid flow volume, which must be linearized as well), proper boundary conditions, and a solution algorithm. The resulting system would be in the form of **Ax** = **b** where **A** is the matrix of coefficients, **x** is the unknowns, and **b** is the vector of resulting residuals. The choice to solve the system is either by ***Direct methods***, or by ***iterative schemes***. Most CFD codes choose the later. In the case of iterative scheme, compromising the most, a ***convergence rate*** must be applied to residuals (usually 10^{-6} or less).

1.3.7 Solving Algorithms

The main source of algorithms are based on two concepts:

- Density Based
- Pressure Based (mostly for M ≲ 3 ; low Reynolds number incompressible flow)

For most applications considered are Pressure based or a close variety of it (i.e., SIMPLE) would be best suited. It is well noted that this scheme is suited for Mach number of ≦ 3. The other consideration is the ***Implicit/Explicit*** factor which is discussed further.

1.3.8 Classification of Equation to be solved and Solution Method (Math/Physics)

The solution methods could be depending on flow characteristics such ***steady/un-steady***, ***compressible/incompressible***, ***viscous/inviscid (Euler vs. Navier-Stokes)*** as well as parameters such as ***Mach number*** which govern flow speed for ***subsonic/transonic/supersonic*** regions. Each dependency, or their combination, could alter the mathematical nature of the equations, therefore,

influencing the selection of appropriate method of solution. Moreover, the 2D & 3D versions of same equations could under similar conditions, exhibit different characteristics. For example, while 2D boundary layer equations are ***parabolic*** in nature and could be solved by marching methods, the 3D are ***hyperbolic*** and require different technique. The time dependency could also play dominant role. Prime example would be un-steady Euler equation which of course requires time integration as it is ***hyperbolic*** for all flow regions, and could benefit from a time marching procedure. On the other hand, the steady-state version, although simpler, has ***elliptic*** behavior for subsonic regions, parabolic for sonic and *hyperbolic* for supersonic flows (see **Table 1.3.2**). Therefore, requiring different strategy on each occasion as previously noted [Diskin & Thomas][15].

Flow	Subsonic M < 1	Sonic M = 1	Supersonic M > 1
Steady	Elliptic	Parabolic	Hyperbolic
Unsteady	Hyperbolic	Hyperbolic	Hyperbolic

Table 1.3.2 Classification of the Euler Equation on Different Regimes

1.1.1 Marching vs. Equilibrium Problems (Math/Physics)

Marching or propagation problems are ***transient or transient-like*** where the solution of PDE is required on an open domain, subject to a set of Initial or boundary conditions. Mathematically, these are ***parabolic*** or ***hyperbolic***, as discussed previously where the flow of information is upstream. Typical examples include unsteady-inviscid flow, steady-state inviscid supersonic, transient heat conduction, as well as boundary layer flows. Alternatively, the equilibrium problem is where the solution is subjected to BC's enclosing the domain, therefore, they are classified as ***Boundary Value Problems (BVP)***. Mathematically, they are ***elliptic*** in nature as the flow of information is scattered through the domain and propagates in all direction. Examples of equilibrium problems are steady-state temperature distribution, incompressible inviscid flows, and equilibrium stress distribution in solids.

1.3.9 Explicit vs. Implicit Methods (Math)

The solution methods could be characterized into two main category of ***Explicit*** and ***Implicit.*** While the ***Explicit*** schemes have the advantage of relative ease in implementation, as they don't require any matrix inversions, but are bounded by step size and stability considerations. On the other hand, the ***implicit*** schemes are more prone to be stable, but computationally intensive as they mostly require linearization of systems of equations and matrix solvers. The linearization of the equations are mostly achieved by ***Taylor series expansion*** of non-linear coefficient matrixes. The final resultant expression could be devised as multiple simpler steps to solve using concepts such ***Approximate Factorization (AF)***. The matrix solvers are also could be grouped as ***Iterative solvers*** (*Guass-Seidel, SOR, ADI, block iterative*) or ***direct solvers*** using schemes such as *Guassian Elimination* or *Thomas Algorithm.* The validity and feasibility of each method depends to the application. In Computational

[15] Diskin, Boris; Thomas, James: *"Comparison of Node-Centered and Cell-Centered Unstructured Finite-Volume Discretization: Inviscid Fluxes,"* AIAA Journal 2010.

Fluid Dynamics (CFD), the governing equations are nonlinear, and the number of unknown variables is typically very large. Under these conditions implicitly formulated equations are almost always solved using iterative techniques. Iterations are used to advance a solution through a sequence of steps from a starting state to a final, converged state. This is true whether the solution sought is either one step in a transient problem or a final steady-state result. In either case, the iteration steps resemble a time-like process. Of course, the iteration steps usually do not correspond to a realistic time-dependent behavior. In fact, it is this aspect of an implicit method that makes it attractive for steady-state computations, because the number of iterations required for a solution is often much smaller than the number of time steps needed for an accurate transient that asymptotically approaches steady conditions. On the other hand, it is also this "***distorted transient***" feature that leads to the question, "What are the consequences of using an implicit versus an explicit solution method for a time-dependent problem?" The answer to this question has two parts. The first part has to do with ***numerical stability***, and the second part with ***numerical accuracy***.

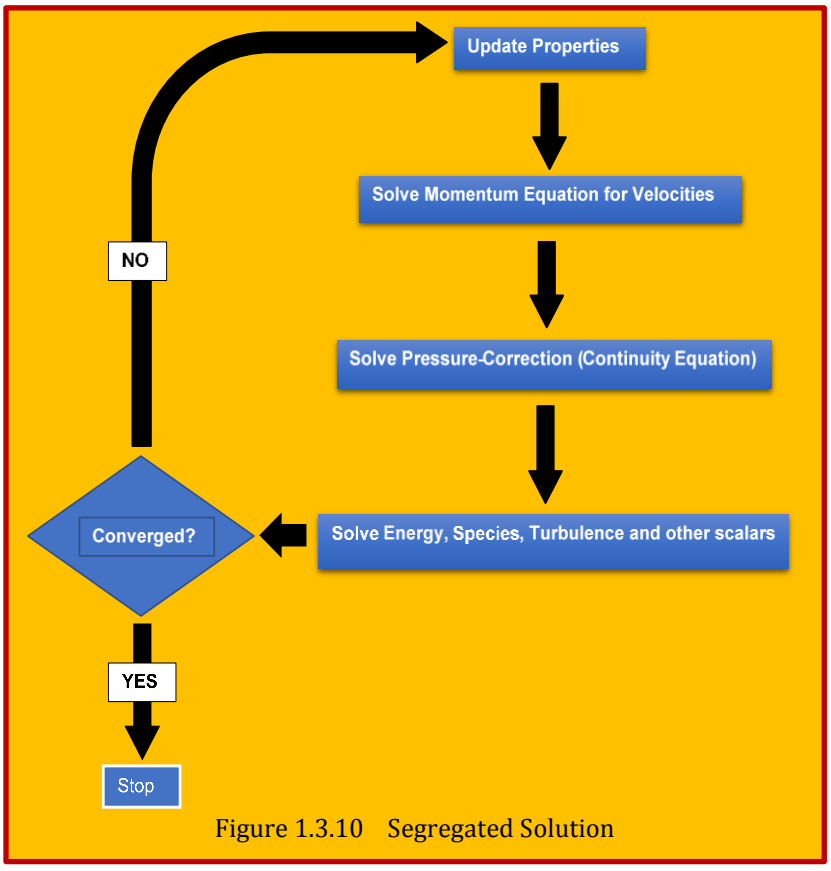


Figure 1.3.10 Segregated Solution

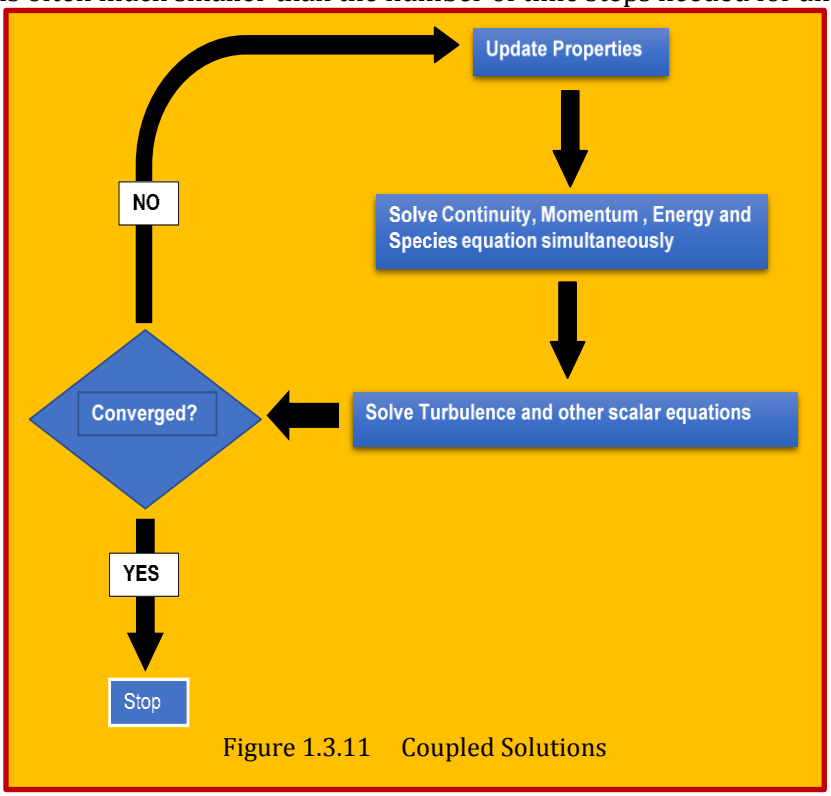


Figure 1.3.11 Coupled Solutions

1.3.10 Marching vs. Equilibrium Problems (Physics)

Marching or propagation problems are ***transient or transient-like*** where the solution of PDE is required on

an open domain, subject to a set of Initial or boundary conditions. Mathematically, these are ***parabolic*** or ***hyperbolic***, as discussed previously where the flow of information is upstream. Typical examples include unsteady-inviscid flow, steady-state inviscid supersonic, transient heat conduction, as well as boundary layer flows. Alternatively, the equilibrium problem is where the solution is subjected to BC's enclosing the domain, therefore, they are classified as Boundary Value Problems (BVP). Mathematically, they are ***elliptic*** in nature as the flow of information is scattered through the domain and propagates in all direction. Examples of equilibrium problems are steady-state temperature distribution, incompressible inviscid flows, and equilibrium stress distribution in solids.

1.3.11 Coupled vs Segregated Algorithms

The ***segregated*** and ***coupled*** approaches differ in the way that the continuity, momentum, and (where appropriate) energy and species equations are solved, separated. The solver solves these equations sequentially (i.e., segregated from one another – see **Figure 1.3.10**), while the coupled solver solves them simultaneously (i.e., coupled together – see **Figure 1.3.11**). Both formulations solve the equations for additional scalars (e.g., turbulence or radiation quantities) sequentially. The segregated solver traditionally has been used for incompressible and mildly compressible flows. The coupled approach, on the other hand, was originally designed for high-speed compressible flows. Both approaches are now applicable to a broad range of flows (from incompressible to highly compressible), but the origins of the coupled formulation may give it a performance advantage over the segregated solver for high-speed compressible flows [Fluent][16]. Coupled flow needs more resources like memory and computational time as it solves coupled equations. But that also means, it's more stable in cases with high density fluctuations like supersonic flow with shocks etc. It's not the best choice for a standard case as it might be unstable unless you reduce the courant number a lot. Segregated flow is a good choice for most cases since it runs fast, but can have problems with supersonic flows. Mathematically, the main difference is that the fully coupled solver operates on the full Jacobian matrix as one entity. The segregated solver splits the Jacobian matrix into smaller sub-problems, usually by degree of freedom type. Different solution strategies can then be used for each sub-problem. The optimal choice between segregated and fully coupled is problem specific. The fully coupled solver generally requires less iterations but takes up more memory and solution time per iteration [Elabbasi][17].

1.4 Character of some Model Equations

Now we examine briefly the character of some various model equations such as ***Wave equations, Heat equation, Laplace equation***, and ***Burgers equation*** as their exact solution are available and obtained before. These equations can be used to model the behavior of more complicated partial difference equations. Reader should refer to [Anderson et al.][18] for further details. Here, we only present the results in **Table 1.4.1** and **Table 1.4.2** as they unique to two different ***Implicit*** and ***Explicit*** methods. Each of model exhibits certain distinctive features that characteristic of the class of methods. Detail description on each method is available in [Anderson et al.]. Based on the information presented here, it is clear that many techniques can be used to solve the same problem. The difference is the quality of solution produced and its applicability to problem in hand. The selection can be aided by experience gained in programming the various methods to solve the equations.

---

[16] Solver Panel, Fluent® Inc. 2003.

[17] N. Elabbasi, Certified Consultant, Veryst Engineering, COMSOL Discussion Forum.

[18] D. Anderson, J., Tannehill, R., Pletcher, "Computational Fluid Mechanics and Heat Transfer", ISBN 0-89116-471-5 – 1984.

Methods	Accuracy	Stability	Test Case	Viscosity	Usage
ADI	2nd order	Yes	2D Heat	None	yes
Allen-Cheng	1st order	Conditional	1D Burgers	Viscous	no
Beam-Warming	2nd order	Conditional	1D Burgers	None	yes
Briley-MacDonald	1st order	Yes	1D Burgers	Viscous	yes
Crank-Nicolson	2nd order	Yes	1D Heat	None	yes
Euler	1st order	Yes	1D Wave	None	no
Simple Implicit	1st order	Yes	1D Heat	None	no
Splitting/Fractional-Step	1st order	Conditional	2D Heat	None	no
Trapezoidal	2nd order	Yes	1D Wave	None	yes
Upwind – (Beam Warming)	2nd order	Conditional	1D Wave	None	yes

Table 1.4.1 Implicit Discretization Methods of Model Equations

Methods	Accuracy	Stability	Test Case	Viscosity	Usage
ADE	1st order	Yes	2D Heat	None	Not Recommended
Brailovskaya	1st order	Conditional	1D Burges	Viscous	Not Recommended
DuFort-Frankel	2nd order	Yes	1D Heat	None	Recommended
Euler	1st order	No	1D Wave	None	Not Recommended
FTCS	1st order	No	1D Burges	Viscous	Not Recommended
Hopscotch	1st order	Yes	2D Heat	None	Not Recommended
Iterative Method	Explicit	Conditional	2D Laplace	None	Recommended
Lax	1st order	Conditional	1D Wave	None	Not Recommended
Lax-Wendroff	2nd order	Conditional	1D Wave	None	yes Recommended
Leap Frog	2nd order	Conditional	1D Wave	None	yes Recommended
MacCormack	2nd order	Conditional	1D Wave	None	yes Recommended
MacCormack	2nd order	yes	1D Burgers	Viscous	yes Recommended
Richardson's	2nd order	no	1D Heat	None	Not Recommended
Rusanov	4th order	Conditional	1D Burges	Viscous	Recommended
Simple Explicit	1st order	Conditional	1D Heat	None	Not Recommended
Time -Split Mac Cormack	2nd order	Conditional	2D Burgers	Viscous	Recommended
Upstream	1st order	Conditional	1D Wave	None	Not Recommended
Upwind	2nd order	Conditional	1D Wave	None	yes
Warming-Kutler-Lomax	3rd order	conditional	1D Wave	None	yes

Table 1.4.2 Explicit Discretization Methods of Model Equations

1.5 A Spatial Discretization Scheme (Beam-Warming)

Among prominent and widely used integration techniques, but not limited, are Crank-Nicolson, Beam-Warming and Mac-Cormack's predictor-corrector schemes[19]. Most could be used cross governing equations and techniques. Other notables are Euler and Trapezoidal implicit method. In fact, most of legacy methods are interpedently related. For example, the Implicit Beam-Warming

[19] Anderson, Dale A; Tannehill, John C; Plecher Richard H; 1984:"*Computational Fluid Mechanics and Heat Transfer"*, Hemisphere Publishing Corporation.

spatial differencing for vector **E** is

$$\Delta^i\mathbf{E} = \left[\frac{\theta_1 + \Delta\mathbf{x}}{1+\theta_2}\right]\frac{\partial}{\partial\mathbf{x}}(\Delta^i\mathbf{E}) + \left[\frac{\Delta\mathbf{x}}{1+\theta_2}\right]\frac{\partial}{\partial\mathbf{x}}(\mathbf{E}^i) + \left[\frac{\theta_2}{1+\theta_2}\right]\Delta^{i-1}\mathbf{E}$$
$$+ \mathcal{O}\left[\left(\theta_1 - \frac{1}{2} - \theta_2\right)(\Delta\mathbf{x})^2 + (\Delta\mathbf{x})^3\right]$$
$$\text{where } \Delta^i\mathbf{E} = \mathbf{E}^{i+1} - \mathbf{E}^i$$

Eq. 1.5.1

This general differencing formula, with appropriate choices of parameters θ_1 and θ_2 reproduces any of the standard differencing schemes as perceived in **Table 1.5.3**. As evident, many different methods could be used to solve the same problem. The difference in the quality of the solutions is frequently small and the selection of the optimal technique becomes difficult. However, the selection process could be aided by experience gained to solve model equations.

Θ_1	Θ_2	Scheme	Order of accuracy
0	0	Euler (explicit)	$(\Delta x)^2$
0	-1/2	Leap Frog (explicit)	$(\Delta x)^3$
1/2	0	Trapezoidal (implicit)	$(\Delta x)^3$
1	0	Euler (implicit)	$(\Delta x)^2$
1	1/2	Three-point backward (implicit)	$(\Delta x)^3$

Table 1.5.3 Choice of Parameters

1.6 Linearization of Equations

For any implicit method, the equation of fluid flow are non-linear in unknown due to appearance of quantities at i+1 level in coefficients. Prime example is the term u ($\partial u/\partial x$) which is from convective term in NS equation. A simple forward differencing, has been used to approximated by 1D as [Haffmann & Chiang][20]

$$u\frac{\partial u}{\partial x} \approx u_{i+1}\frac{u_{i+1} - u_i}{\Delta x}$$

Eq. 1.6.1

Which of course non-linear.

1.6.1 Frozen (Lagging) Coefficient Method

In this method simply ignore the non-linear coefficient and evaluate it at known station as

$$u\frac{\partial u}{\partial x} \approx u_i\frac{u_{i+1} - u_i}{\Delta x}$$

Eq. 1.6.2

The procedure provides a consistent representation (0th order Taylor series expansion) and ensure that the differencing scheme is formally no better than 1st order accurate in marching coordinates.

1.6.2 Simple Iterative Method

In this method, the lagged coefficient is updated through a simple iteration until a specified convergence criteria is satisfied [Haffmann & Chiang]. This is complement to previous method.

$$|u_{i+1} - u_i| \le \varepsilon \quad \text{where } \varepsilon << 1$$

[20] K. A. Haffmann, S. T. Chiang, "Computational Fluid Dynamics", 4th edition, Aug. 2000.

Eq. 1.6.3

1.6.3 Newton Raphson Linearization Method

This is by far the method of choice and used extensively. The Newton's method also called quasi-linearization proceeds as follows. Let assume (A) and (B) represent the non-linearity. Define Δ as change in variable between two consecutive iterations (n),

$$\delta_A = A^{n+1} - \hat{A}^n \quad \text{and} \quad \delta_B = B^{n+1} - \hat{B}^n$$

$$A^{n+1}B^{n+1} = (\hat{A}^n + \delta_A)(\hat{B}^n + \delta_B)$$

$$\text{Expanding, after some manipulation, and dropping the 2nd order term } (\delta_A)(\delta_B)$$

$$A^{n+1}\,B^{n+1} \approx \hat{A}^n B^{n+1} + A^{n+1}\hat{B}^n - \hat{A}^n\hat{B}^n \quad \text{for } n = 0 \rightarrow A^{n+1} = \hat{A}^n$$

Eq. 1.6.4

Which is now linear. The carrot denotes an evaluation of variables from previous iteration.

1.6.3.1 Newton Linearization with Coupling

Several investigators have observed that convergence of iteration can be accelerated by solving the momentum and continuity equations in coupled manner. The v ($\partial u/\partial y$) term is linearized by using Which of course is linear[21].

$$v_i^{n+1} = \hat{v}_i^{n+1} + \delta_v \quad , \quad u_i^{n+1} = \hat{u}_i^{n+1} + \delta_u$$

$$\text{after product terms involving } \delta \text{ dropped:}$$

$$\left(v\frac{\partial u}{\partial y}\right)^{n+1} \approx \hat{v}^{n+1}\left(\frac{\partial u}{\partial y}\right)^{n+1} + v^{n+1}\left(\frac{\partial \hat{u}}{\partial y}\right)^{n+1} - \hat{v}^{n+1}\left(\frac{\partial \hat{u}}{\partial y}\right)^{n+1}$$

Eq. 1.6.5

1.6.4 Extrapolating the Coefficients

Values of the coefficients can be obtained at n+1 level by extrapolating based on values already obtained from previous n level. Formally, the truncation error of this procedure can be made as small as we wish[22]. For example, we can use

$$u_i^{n+1} = u_i^n + \left.\frac{\partial u}{\partial x}\right|_i^n \Delta x_+ + O(\Delta x)^2 \quad \text{where} \quad \left.\frac{\partial u}{\partial x}\right|_i^n = \frac{u_i^n - u_i^{n-1}}{\Delta x_-} + O(\Delta x)^2$$

$$u_i^{n+1} = u_i^n + \frac{u_i^n - u_i^{n-1}}{\Delta x_-}\Delta x_+ + O(\Delta x)^2$$

Eq. 1.6.6

A similar procedure can be used for other coefficients needed at n+1 level.

1.6.5 Case Study – Linearization of 1D Unsteady Euler Equation

A widely used method for achieving this linearization was first suggested by ***Beam and Warming*** in 1976. For purposes of discussion, let us consider the system of unsteady 1D Euler equation:

---

[21] D. Anderson, J., Tannehill, R., Pletcher, *"Computational Fluid Mechanics and Heat Transfer"*, 1984.

[22] D. Anderson, J., Tannehill, R., Pletcher, *"Computational Fluid Mechanics and Heat Transfer"*, 1984.

$$\frac{\partial \mathbf{U}}{\partial t}+\frac{\partial \mathbf{F}}{\partial x}=0$$

Eq. 1.6.7

where **F** = **F**(**U**). Using the ***Crank-Nicolson differencing scheme***, **Eq. 1.6.7** can be written in finite-difference form as

$$\mathbf{U}_i^{n+1}=\mathbf{U}_i^n-\frac{\Delta t}{2}\left[\left(\frac{\partial \mathbf{F}}{\partial x}\right)_i^n+\left(\frac{\partial \mathbf{F}}{\partial x}\right)_i^{n+1}\right]$$

Eq. 1.6.8

(Sometimes the representation of the spatial derivatives as an ***average*** between time levels *n* and *n* + 1, is called the ***trapezoidal rule***). **Eq. 1.6.8**, as it stands, is a ***nonlinear*** difference equation. However, the Beam and Warming approach leads to a local linearization as follows. Expand **F** in a series expansion around time level n, that is,

$$\mathbf{F}_i^{n+1}=\mathbf{F}_i^n+\left(\frac{\partial \mathbf{F}}{\partial \mathbf{U}}\right)_i^n\left(\mathbf{U}_i^{n+1}-\mathbf{U}_i^n\right)+\ldots \text{ where } \left(\frac{\partial \mathbf{F}}{\partial \mathbf{U}}\right)_i^n=\mathbf{A}_i^n$$

Eq. 1.6.9

Substituting and rearranging we get

$$\mathbf{U}_i^{n+1}=\mathbf{U}_i^n-\frac{\Delta t}{2}\left[2\left(\frac{\partial \mathbf{F}}{\partial x}\right)_i^n+\frac{\partial}{\partial x}\mathbf{A}_i^n\left(\mathbf{U}_i^{n+1}-\mathbf{U}_i^n\right)\right]$$

Eq. 1.6.10

Therefore, we have achieved what we wanted. We have taken a nonlinear difference equation, namely, **Eq. 1.6.10**, and by means of a ***Taylor series expansion*** using ***lagging coefficients*** have linearized this equation, obtaining the ***linear*** difference equation. This is one way of achieving the linearization; there are others. However, the purpose of this subsection is to emphasize that ***implicit finite-difference solutions*** of the conservation form of the governing flow equations lead to ***nonlinear difference equations*** which must in some fashion be linearized before a practical numerical solution can be obtained. It should be noted that a similar idea for linearization was carried out by [Briley and McDonald]. In contrast to Beam and Warming, who treated the function, [Briley and McDonald] treated the time derivative; the results are effectively the same.

1.7 Multi-Dimensional (2D) Problem and Approximate Factorization (AF) Scheme

The difficulty of working with large matrices resulting from straightforward implementation of implicit schemes to PDEs in higher dimensions has led to the development of the so-called ***split*** or ***factored schemes***. As the name implies, such schemes split a multi-dimensional problem to a series of one-dimensional ones, which are much easier to solve. Of course, in general, this conversion cannot be done exactly and some error is incurred. However, as we will show below, the splitting error is of the same order as the error already incurred in discretizing the problem in space and time. That is, the splitting approximation does not erode the order of accuracy of the scheme.[23] The basic system uner consideration is of the form (2D) given by

[23] Moin, P; "Fundamentals Engineering Numerical Analysis", Cambridge University Press.

$$\frac{\partial \mathbf{U}}{\partial t}+\frac{\partial \mathbf{E}}{\partial x}+\frac{\partial \mathbf{F}}{\partial y}=0$$

Eq. 1.7.1

Where **U** is the vector of conservative variables and **E** and **F** are vector of function of **U**. If the Traezoidal rule ($\Theta_1=1/2$, $\Theta_2=0$) is used as the basis integration scheme,

$$\mathbf{U}^{n+1}=\mathbf{U}^{n}+\frac{\Delta t}{2}\left[\left(\frac{\partial \mathbf{U}}{\partial t}\right)^{n}+\left(\frac{\partial \mathbf{U}}{\partial t}\right)^{n+1}\right]$$

Eq.1.7.2

This expression provides a 2nd order integration algoritem for the unkown vector, $\mathbf{U}^{n+1}$, at the next time level. A local Taylors series expasion of direvtives of **E** and **F** used to obtain a Linear equation such that

$$\mathbf{E}^{n+1}=\mathbf{E}^{n}+[\mathbf{A}]\left(\mathbf{U}^{n+1}-\mathbf{U}^{n}\right) \quad \text{where} \quad [\mathbf{A}]=\frac{\partial \mathbf{E}}{\partial \mathbf{U}}$$

$$\mathbf{F}^{n+1}=\mathbf{F}^{n}+[\mathbf{B}]\left(\mathbf{U}^{n+1}-\mathbf{U}^{n}\right) \quad \text{where} \quad [\mathbf{B}]=\frac{\partial \mathbf{F}}{\partial \mathbf{U}}$$

Eq. 1.7.3

When the linearization given by
Eq. 1.7.3 is substitued into a linear system for $\mathbf{U}^{n+1}$ results as

$$\left\{[\mathbf{I}]+\frac{\Delta t}{2}\left(\frac{\partial}{\partial x}[\mathbf{A}]^{n}+\frac{\partial}{\partial y}[\mathbf{B}]^{n}\right)\right\}\mathbf{U}^{n+1}=$$

$$\left\{[\mathbf{I}]+\frac{\Delta t}{2}\left(\frac{\partial}{\partial x}[\mathbf{A}]^{n}+\frac{\partial}{\partial y}[\mathbf{B}]^{n}\right)\right\}\mathbf{U}^{n}-\Delta t\left(\frac{\partial \mathbf{E}}{\partial x}+\frac{\partial \mathbf{F}}{\partial y}\right)^{n}$$

Eq. 1.7.4

This is a linear system for theunknown $\mathbf{U}^{n+1}$. Direct solution of
Eq. 1.7.3 is usually avioded due to llarge operation count in treating muti-dimensional system. The path chosen is ususlly to reduce the multi-dimensional problem into a sequence of one-demensional inversrsions using the method of ***Approximate Factorization (AP)*** in cross flow plane. **Eq. 1.7.4** may be approximatly facotred into

$$\left([\mathbf{I}]+\frac{\Delta t}{2}\frac{\partial}{\partial x}[\mathbf{A}]^{n}\right)\left([\mathbf{I}]+\frac{\Delta t}{2}\frac{\partial}{\partial y}[\mathbf{B}]^{n}\right)\mathbf{U}^{n+1}=$$

$$\left([\mathbf{I}]+\frac{\Delta t}{2}\frac{\partial}{\partial x}[\mathbf{A}]^{n}\right)\left([\mathbf{I}]+\frac{\Delta t}{2}\frac{\partial}{\partial y}[\mathbf{B}]^{n}\right)\mathbf{U}^{n}-\Delta t\left(\frac{\partial \mathbf{E}}{\partial x}+\frac{\partial \mathbf{F}}{\partial y}\right)^{n}$$

Eq. 1.7.5

This equation is much easier and more cost effective to implement than the large system encountered in the non-factored form. Basically, the multi-dimensional problem is reduced to a series of one-dimensional problems by ignoring the cross terms while maintaine the formal order of accuracy.

$$\Delta \mathbf{U}^n = \mathbf{U}^{n+1} - \mathbf{U}^n$$

$$\left([\mathbf{I}] + \frac{\Delta t}{2}\frac{\partial}{\partial x}[\mathbf{A}]^n\right)\left([\mathbf{I}] + \frac{\Delta t}{2}\frac{\partial}{\partial y}[\mathbf{B}]^n\right)\Delta \mathbf{U}^n = -\Delta t\left(\frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y}\right)^n$$

$$\left([\mathbf{I}] + \frac{\Delta t}{2}\frac{\partial}{\partial x}[\mathbf{A}]^n\right)\Delta \mathbf{U}^{'} = -\Delta t\left(\frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y}\right)^n$$

$$\left([\mathbf{I}] + \frac{\Delta t}{2}\frac{\partial}{\partial y}[\mathbf{B}]^n\right)\Delta \mathbf{U}^n = \Delta \mathbf{U}^{'}$$

Eq. 1.7.6

The solution of this system is not trival. The x and y sweeps each require the solution of ***block tridiagonal system of equations***, or better known as ***TDMA (Thomas Algorithm).*** Each block is M x M if there are M elements in the unknown **U** vector.

1.7.1 Recommendation on Linearization

For many calculations, the linearization introduced by simply lagging the coefficients will cause no serious deterioration of accuracy. Errors associated with linearization of coefficients are simply truncation errors which can be controlled by adjustment in marching step size. Many investigators have been used this procedures satisfactory. For any problem in which this linearization causes special difficulties, extrapolation or newton's linearization with coupling is recommended[24].

1.8 Convection & Diffusion Terms Discretization

In problems where fluid flow plays a significant role we must account for the effects of convection[25]. Diffusion always occurs alongside convection in nature so here we examine methods to predict combined convection and diffusion. The principal problem in the discretization of the convective terms is the calculation of the value of transported property Q at control volume faces and its convective flux across these boundaries. (see **Eq. 1.8.1**). We introduced the ***central differencing*** method of obtaining discretized equations for the ***diffusion*** and ***source terms*** on the right hand side of equation. It would seem obvious to try out this practice, which worked so well for diffusion problems. The diffusion process affects the distribution of a transported quantity along its gradients in ***all directions***, whereas convection spreads influence only in the flow direction. This crucial difference manifests itself in a stringent upper limit to the grid size (Peclet No.), which is dependent on the relative strength of convection and diffusion, for stable convection–diffusion calculations with central differencing.

$$\underbrace{\frac{\partial}{\partial t}(\rho Q)}_{\text{Transient}} + \underbrace{\underbrace{\frac{\partial}{\partial x_j}(\rho U_j Q)}_{\text{Convection}} = \underbrace{\frac{\partial}{\partial x_j}\left(\Gamma_Q \frac{\partial Q}{\partial x_j}\right)}_{\text{Diffusion}}}_{\text{Transport}} + \underbrace{S_Q}_{\text{Source}}$$

Eq. 1.8.1

Naturally, we also present the case for a number of alternative discretization practices for the ***convective*** effects which enable stable computations under less restrictive conditions. For these

---

[24] See Previous.

[25] H K Versteeg and W Malalasekera, "*An Introduction to Computational Fluid Dynamics - The Finite Volume Method*", Second Edition, © Pearson Education Limited 1995, 2007.

terms, typically, an LUD (Linear Upwind Differencing), QUICK, TVD, NVD, second-order convective upstream split scheme (CUSP), or other upwind differencing schemes can be used. ***Upwind schemes are developed which take the direction of the flow in consideration***[26]. Nowadays, upwind schemes are the major spatial discretization technique of main research and commercial codes[27-28].

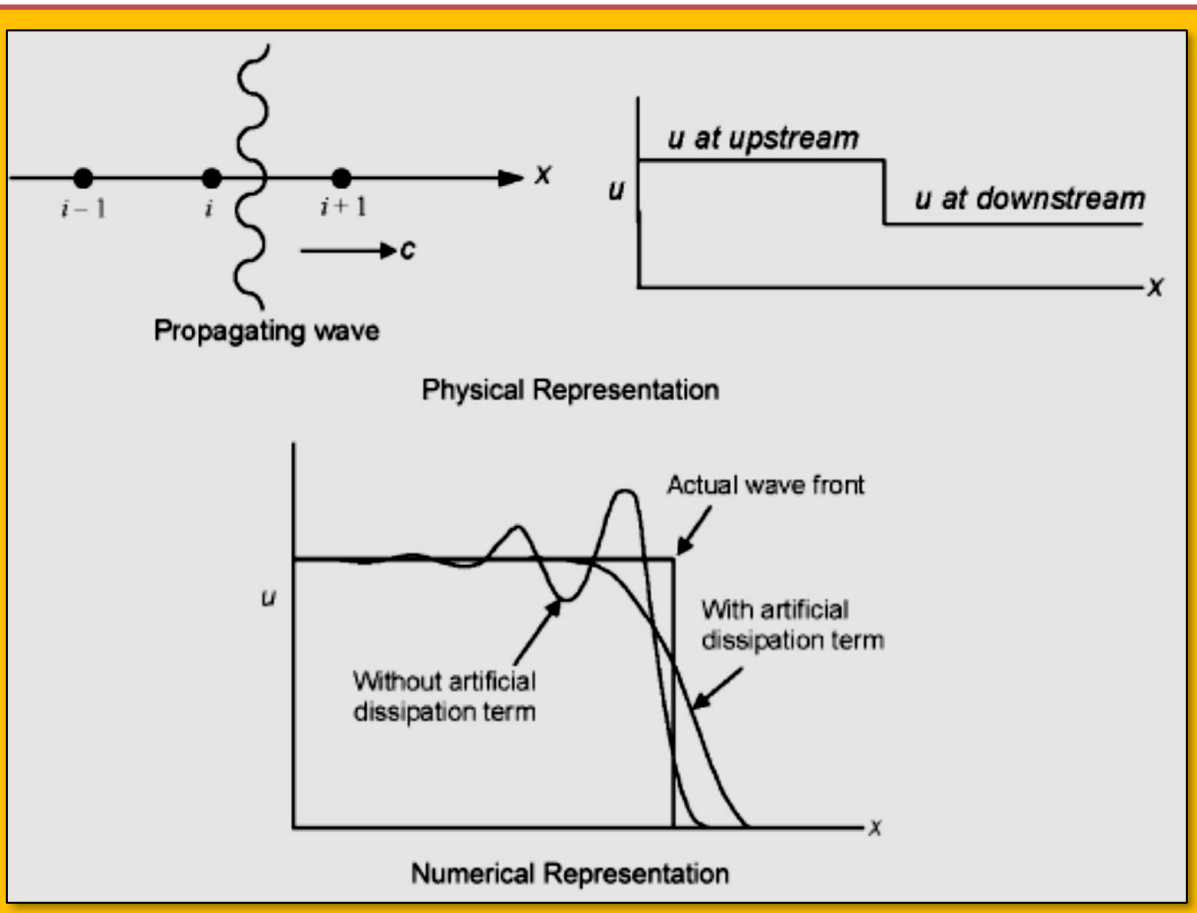


Figure 1.8.1 Schematic physical representation of a propagating wave in the positive x direction accompanied by the flow field at a given instant of time and numerical representation with and without an artificial dissipation term in the vicinity of the discontinuous wave interface

1.8.1 Upwind Differencing For Convection

It uses the propagation of information with the theory of characteristics in constructing the information traveling in opposite directions in a separate and stable manner. Second order central schemes require scalar ***artificial dissipation*** to damp oscillations generated near the high gradient

[26] Korhan Coskun, "*Three-Dimensional Laminar Compressible Navier Stokes Solver For Internal Rocket Flow Applications*", A Thesis Submitted To The Graduate School Of Natural And Applied Sciences of The Middle East Technical University, 2007.
[27] Frink, N. T., "*Upwind Scheme for Solving the Euler Equations on Unstructured Grids*", AIAA Journal Vol. 30, No. 1, pp. 70-77, January, 1992.
[28] Dadone, A., and Grossman, B., "*Characteristic-Based, Rotated Upwind Scheme for the Euler Equations*", AIAA Journal Vol. 30, No. 9, pp.2219-2226, September, 1992.

regions. As an 1D wave equation example[29], consider the simple illustration of the inviscid Euler equation in the non-conservative form:

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

Eq. 1.8.2

where c is the advective velocity describing the propagation of a wave in the direction of the x axis. There is a discontinuity in the velocity u across the wave, as described in **Figure 1.8.1**. Assuming that c is positive, properties at grid point I should depend on the upstream flow-field properties at grid point i -1. Grid point i +1, on the other hand, should not physically influence the point at i; hence the choice of numerical scheme must reflect the flow physics. If the gradient $\partial u/\partial x$ is approximated using central differencing, such as the traditional approach described above for MacCormack (1969), the velocity u profile results in an oscillatory behavior near the discontinuous wave front. In some circumstances, the numerical procedure can lead to an unstable and chaotic solution.

The common remedy, as mentioned, is to introduce an ***artificial dissipation*** term. Despite the numerical result exhibiting a monotone variation (no oscillations), the diffusive property remains an undesirable element, as illustrated for the numerical representation in **Figure 1.8.1**. Higher order upwind methods require limiter functions for second order accuracy in space. Characteristic theory is easy to understand in one-dimensional flows, but for 2D and 3D flow problems, the flow direction is not clearly identified. However, choosing the upwind direction being normal to the face of the computational cell across which the fluxes are computed is a commonly used way for 2D and 3D flows. Upwind schemes may be divided into two categories as ***Flux Vector Splitting schemes*** and ***Flux Difference Splitting (Godunov) schemes***.

1.8.1.1 Flux Vector Splitting Schemes

Upwind discretization is obtained by splitting the flux vector into two parts based on information coming from upwind and downwind of the cell face in flux vector splitting algorithms[30]. In other words, the flux terms are split according to the sign of associated with propagation speeds. The main drawback of flux vector splitting methods is evident in the vicinity of sonic conditions since the splitting of flux is done with respect to the sign of the Mach number or the velocity vector. Consider the Euler equations for unsteady, one-dimensional flow written as,

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0$$

Eq. 1.8.3

As described by **Eq. 1.8.3**, A is the Jacobian of **F**; $\mathbf{A} = \partial\mathbf{F}/\partial\mathbf{U}$. For an inviscid flow, the flux vector **F** can be expressed directly in terms of its Jacobian as

$$\mathbf{F} = \mathbf{AU}$$

Eq. 1.8.4

Let us define two matrices [λ+] and [λ-] made up of the positive and negative eigenvalues of A, respectively. For example, if we have a subsonic flow, then we have $\lambda_1 = u$ and $\lambda_2 = u + c$, both positive values, and $\lambda_3 = u - c$, a negative value. Therefore, in this case, by definition

---

[29] JiyuanTu, Guan-HengYeoh, ChaoqunLiu, "*Chapter 9 - Some Advanced Topics in CFD- Computational Fluid Dynamics (3rd Edition) A Practical Approach*", ScienceDirect, 2018.

[30] Steger, J. L., and Warming, R. F., "*Flux Vector Splitting of the Inviscid Gas Dynamics Equations with Applications to Finite Difference Methods*", Journal of Computational Physics, Vol. 40, pp. 263-293, 1981.

$$[\lambda^+] = \begin{bmatrix} u & 0 & 0 \\ 0 & u+c & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad , \quad [\lambda^-] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & u-c \end{bmatrix} \quad , c = \text{speed of sound}$$

Eq. 1.8.5

With this, we can split the flux vector **F** into two parts, $\mathbf{F}^+$ and $\mathbf{F}^-$:

$$\mathbf{F} = \mathbf{F}^+ + \mathbf{F}^- \quad \text{and} \quad \frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^+}{\partial x} + \frac{\partial \mathbf{F}^-}{\partial x} = 0$$

Eq. 1.8.6

where F^+ and F^- are defined by

$$\mathbf{F}^+ = \mathbf{A}^+\mathbf{U} \quad \rightarrow \quad \mathbf{A}^+ = \mathbf{T}\,\lambda^+\mathbf{T}^{-1}$$
$$\mathbf{F}^- = \mathbf{A}^-\mathbf{U} \quad \rightarrow \quad \mathbf{A}^- = \mathbf{T}\,\lambda^-\mathbf{T}^{-1}$$

Eq. 1.8.7

Where **T** is vector of eigenvectors of eigenvalues λ. **Eq. 1.8.6** is an example of flux-vector splitting where $\mathbf{F}^+$ corresponds to a flux in the positive x direction, with information being propagated from left to right by the positive eigenvalues $\lambda_1 = u$ and $\lambda_2 = u + c$. Hence, when $\partial\mathbf{F}^+/\partial\mathbf{x}$ is replaced by a difference expression, a backward (rearward) difference should be used since $\mathbf{F}^+$ is associated only with information coming from upstream of grid point (i , j). Similarly, $\mathbf{F}^-$ corresponds to a flux in the negative x direction, with information being propagated from right to left by the negative eigenvalue $\lambda_3 = u - c$. Hence, when $\partial\mathbf{F}/\partial\mathbf{x}$ is replaced by a difference expression, a forward difference should be used since F^- is associated only with information coming from downstream of grid point (i, j). This is why the flux-vector-splitting scheme described by **Eq. 1.8.7** is a type of upwind scheme; flux-vector splitting is a numerical algorithm which attempts to account for the physically proper transfer of information throughout the flow. There are various improvisations on flux-vector splitting in the modem CFD literature. One such example is ***Van Leer's flux splitting*** which imposes certain conditions on $\mathbf{F}^+$ and $\mathbf{F}^-$ to improve the performance of the numerical scheme for local Mach numbers near 1.

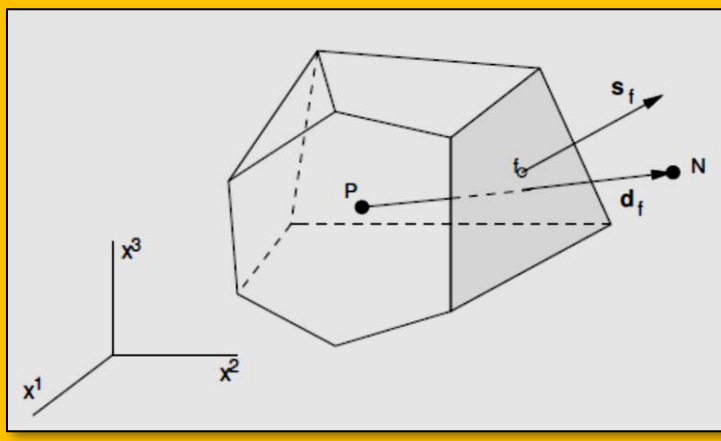


Figure 1.8.2 An Arbitrary Polyhedral Control Volume

1.8.1.2 Flux Difference Splitting (Godunov) Schemes

In the flux difference splitting schemes, local Riemann problem is solved on each face of cells. The flow variables are taken as constant over the left and right states of the cell face. Using left and right states of the face, local Riemann problem is solved to achieve convective fluxes at the face. In the original ***Godunov scheme***[31], the local Riemann problem is solved exactly. Since this approach is computationally

[31] Godunov, S. K., "*A Difference Method for the Numerical Computation of Discontinuous Solutions of Hydrodynamic Equations*", Math Sbornik, Vol. 47, pp. 271-306, 1959.

expensive, some approximate Riemann solvers have been built by [Roe][32], [Osher and Solomon][33], [Toro][34].

1.8.2 Diffusion Term Discretization

The finite-volume discretization which uses meshes made of arbitrary polyhedral control volumes, **Figure 1.8.2**, enabled one to discard with the complicated curvilinear coordinates. There, the diffusive flux of the variable Φ through an internal cell face f is approximated as

$$\mathbf{D}_f = \int_{S_f} \Gamma_\phi \, \mathrm{grad}\phi . \mathrm{ds} \approx \Gamma_\phi \; (\mathrm{grad}\phi)_f^* . s_f$$

$$(\mathrm{grad}\phi)_f^* = (\mathrm{grade}\,\phi)_f + \left[\frac{\phi_n - \phi_p}{|\mathbf{d_f}|} - \frac{(\mathrm{grad}\,\phi)_f . \mathbf{d_f}}{|\mathbf{d_f}|}\right]\frac{\mathbf{d_f}}{|\mathbf{d_f}|}$$

Eq. 1.8.8

The second term on the right hand side (term in brackets) represents the difference between the central difference approximation of the derivative in the direction of vector $\mathbf{d_f}$ and the corresponding value obtained by interpolating cell-center gradients. This correction term detects and smoothed out any unphysical oscillations that might occur in the iteration process[35]. As the results one gets:

$$\mathbf{D}_f \approx \Gamma_\phi \frac{\mathbf{d_f} . \mathbf{d_s}}{\mathbf{d_f} . \mathbf{d_f}} (\phi_N - \phi_p) + \Gamma_\phi \left[(\mathrm{grad}\phi)_f . s_f - \frac{\mathbf{d_f} . \mathbf{d_s}}{\mathbf{d_f} . \mathbf{d_f}} (\mathrm{grad}\phi)_f . \mathbf{d_f} \right]$$

Eq. 1.8.9

The first part of this term is treated implicitly, and the rest explicitly. Similarly, [Jasak][36] in his PhD thesis and in less details in the subsequent publication gives the geometric interpretation to **Eq. 1.8.8**, and **Eq. 1.8.9** naming them the minimum correction, orthogonal correction, and over-relaxed approach, respectively (**Figure 1.8.3)**. He approximates the diffusive flux as

$$\mathbf{D}_f \approx \Gamma_\phi \; (\mathrm{grad}\phi)_f^* . s_f = \Gamma_\phi \left[\frac{(\phi_N - \phi_p)}{|\mathbf{d_f}|} |\Delta \mathbf{f}| + (\mathrm{grad}\phi)_f . \mathbf{k_f} \right]$$

Eq. 1.8.10

where vectors $\mathbf{\Delta_f}$ and $\mathbf{k_f}$ satisfy the following condition

$$\mathbf{k_f} = \mathbf{s_f} - \Delta_\mathbf{f}$$

Eq. 1.8.11

[Jasak][88] compared the performance of these three approaches and found out that the over-relaxed approach is more robust and more efficient than the other two approaches, especially in case of highly non-orthogonal meshes. This is expected because, unlike the other approaches which are somewhat arbitrary, the over-relaxed approach comes as a result of a direct discretization of the

---

[32] Roe, P. L., "*Discrete Models for the Numerical Analysis of Time Dependent Multidimensional Gas Dynamics*", Journal of Computational Physics, Vol. 63, pp. 458-476, 1986.

[33] Osher, S., and Solomon, F., "*Upwind Schemes for Hyperbolic Systems of Conservation Laws*", Mathematics of Computation, Vol. 38, No. 158, pp. 339-377, 1982.

[34] Toro, E. F., *Riemann Solvers and Numerical Methods for Fluid Dynamics*, 2nd Edition, Springer, 1999.

[35] I. Demirdˇzi´c, "*On the Discretization of Diffusion Term in the Finite-Volume Continuum Mechanics*", Numerical Heat Transfer Fundamentals · July 2015.

[36] H. Jasak, Error *Analysis and Estimation for the Finite Volume Method with Applications for Fluid Flow*, PhD Thesis, University of London, 1996.

transport equations.

$$\text{Minimum Correction}: \Delta_f = \frac{\mathbf{d_f}.\mathbf{s_f}}{\mathbf{d_f}.\mathbf{d_f}}.\mathbf{d_f}$$

$$\text{Orthgonal Correction}: \Delta_f = \frac{|\mathbf{s_f}|}{|\mathbf{d_f}|}.\mathbf{d_f}$$

$$\text{Over} - \text{Relaxed Correction}: \Delta_f = \frac{\mathbf{s_f}.\mathbf{s_f}}{\mathbf{d_f}.\mathbf{s_f}}.\mathbf{d_f}$$

Eq. 1.8.12

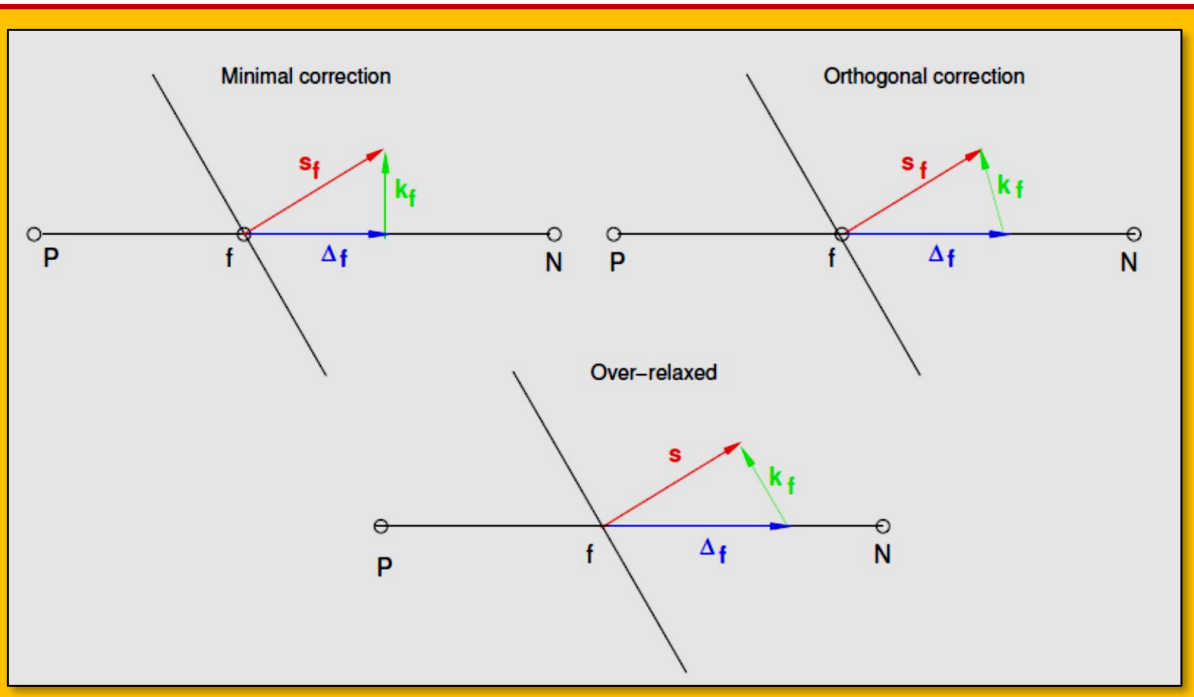
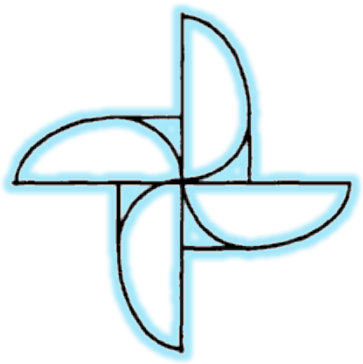


Figure 1.8.3 Geometric Interpretation of the Diffusion Term Approximation –[H. Jasak]



2 Solver Schemes I

2.1 Method of Characteristics for Inviscid Flow

For inviscid flows, ***method of characteristics*** is the oldest and most nearly exact method which still used to solve parabolic PDE's. Other notable methods are ***Shock-Capturing*** which could be used to capture large pressure gradients within a supersonic flow using a predictor-corrector scheme such as Mac Cormack's. The Euler equations for steady-state, isentropic, irrotational can be combined into a single equation as potential equation. This could be treated using a type-dependent differentiation where for supersonic regions where the flow is hyperbolic an upwind differential recommended. For subsonic region, where the flow is elliptic, a central differencing recommended. Evidently, the solution methods are dependent on several parameters confined to problem at hand. The description for some legacy discretization methods and their characteristic behavior are available in literature. Closed form solutions of non-linear hyperbolic partial differential equation do not exists for general cases. In order to obtain the solution to such an equations we are required to resort to numerical methods. The method of characteristics is the oldest and most nearly exact method in use to solve hyperbolic PDEs. Even though this technique is been replaced by newer finite difference method. A background in characteristic theory and its application is essential. The method of characteristics is a technique which utilizes the known physical behavior of the solution in each point in the flow.

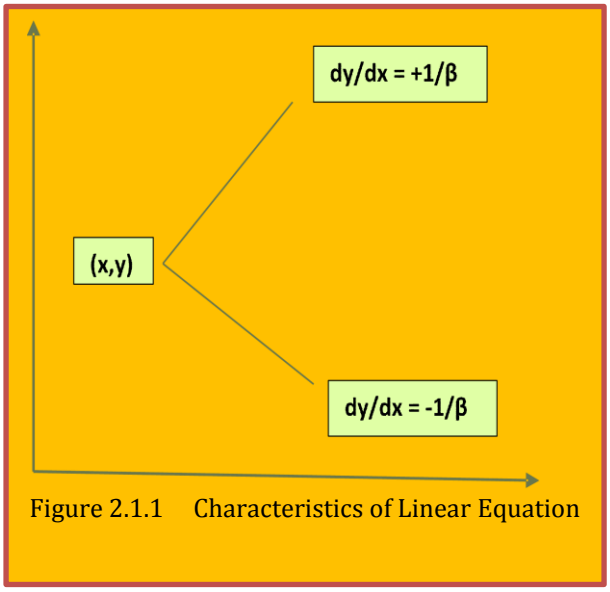


Figure 2.1.1 Characteristics of Linear Equation

2.1.1 Linear Systems

Consider ***2-D, Steady, Supersonic, Non-heat conducting*** of small ***perturbation equations*** for a perfect gas [Anderson et al.][37] as depicted in **Figure 2.2.1**.

$$(1-M_\infty)\phi_{xx}+\phi_{yy}=0 \quad \text{denoting } (1-M_\infty)=\beta^2 \text{ and } u=\frac{\partial\phi}{\partial x},\ v=\frac{\partial\phi}{\partial y}$$

$$\frac{\partial u}{\partial x}-\frac{\partial v}{\partial y}=0,\quad \frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}=0 \quad \text{write in vector form} \quad \frac{\partial \mathbf{w}}{\partial x}+[\mathbf{A}]\frac{\partial \mathbf{w}}{\partial y}=0$$

$$\text{where } \mathbf{w}=\begin{bmatrix}u\\v\end{bmatrix} \text{ and } [\mathbf{A}]=\begin{bmatrix}0 & -\frac{1}{\beta^2}\\ -1 & 0\end{bmatrix}$$

Eq. 2.1.1

The eigenvalues of this system are the eigenvalues of [**A**]. These are obtained by extracting the roots of characteristics equation of [**A**] as

[37] D. Anderson, J., Tannehill, R., Pletcher, "Computational Fluid Mechanics and Heat Transfer", ISBN 0-89116-471-5 – 1984.

$$|[\mathbf{A}] - \lambda[\mathbf{I}]| = 0 \quad \text{or} \quad \begin{vmatrix} -\lambda & -\frac{1}{\beta^2} \\ -1 & -\lambda \end{vmatrix} = 0 \ , \quad \lambda^2 - \frac{1}{\beta^2} = 0 \ , \ \lambda_1 = \frac{1}{\beta} \ , \ \lambda_2 = -\frac{1}{\beta}$$

Eq. 2.1.2

This is pair of roots from the differential equation of characteristics. Next we determine the compatibility equation. These equations are obtained by pre-multiplying the system of equations by left eigenvectors of [**A**]. This effectively provides a method for writing the equations along the characteristics. Let L^1 represents the left eigenvectors of [**A**] corresponding to λ_1 and L^2 represents the left eigenvectors corresponding to λ_2. Drive the eigenvectors of [**A**]:

$$\left[L^i\right]^T[\mathbf{A} - \lambda_i\mathbf{I}] = 0$$

$$L^1 = \begin{bmatrix} l_1 \\ l_2 \end{bmatrix} \rightarrow \underbrace{[l_1 \quad l_2]}_{L^T} \underbrace{\begin{bmatrix} -\frac{1}{\beta} & -\frac{1}{\beta^2} \\ -1 & -\frac{1}{\beta} \end{bmatrix}}_{\mathbf{A}} = 0 \qquad L^1 = \begin{bmatrix} -\beta \\ 1 \end{bmatrix}, \ L^2 = \begin{bmatrix} \beta \\ 1 \end{bmatrix}$$

Eq. 2.1.3

The compatibility equations along λ_1 is obtained from

$$\left[L^i\right]^T\left[w_x + [\mathbf{A}]w_y\right] = 0 \quad \text{or} \quad \left[L^i\right]^T\left[w_x + \lambda_i w_y\right] = 0$$

$$\text{Compatibility along } \lambda_1 \text{ is obtained} \quad [-\beta \ , 1] \begin{bmatrix} u_x + \frac{1}{\beta}u_y \\ v_x + \frac{1}{\beta}v_y \end{bmatrix} = 0$$

$$\frac{\partial}{\partial x}(\beta u - v) + \frac{1}{\beta}\frac{\partial}{\partial y}(\beta u - v) = 0 \quad , \ \text{similarly} \quad \frac{\partial}{\partial x}(\beta u + v) - \frac{1}{\beta}\frac{\partial}{\partial y}(\beta u + v)$$

Eq. 2.1.4

It is expressed the fact that quantity (βu - v) is constant along λ_1, and (βu + v) is constant along λ_2. The quantities are called ***Riemann Invariants***. Since these two quantities are constant and opposite pair of characteristics, it is easy to determine u and v at a point. If at a point we know (βu - v) and (βu + v), we can immediately compute both u and v.

2.1.2 Non-Linear Systems

The development presented so far is for a system linear equations for simplicity. In more complex nonlinear settings, the results are not as easily obtained. In the general cases, the characteristics slopes are *not constant* and vary with fluid properties[38] shown in **Figure 2.1.2**. For a general nonlinear problem, the characteristics equation must be integrated numerically to obtain a complete flow field solutions. Consider a 2D supersonics flow of a perfect gas over a flat surface. The Euler equation governing this inviscid flow as a matrix for

[38] See previous.

$$\frac{\partial \mathbf{w}}{\partial x} + [\mathbf{A}]\frac{\partial \mathbf{w}}{\partial y} = 0 \quad \text{where} \quad \mathbf{w} = \begin{bmatrix} u \\ v \\ p \\ e \end{bmatrix} \quad \text{and}$$

$$[\mathbf{A}] = \frac{1}{u^2 - a^2}\begin{bmatrix} uv & -a^2 & -\frac{v}{p} & 0 \\ 0 & \frac{v}{u}(u^2 - a^2) & \frac{u^2 - a^2}{\rho u} & 0 \\ -\rho v a^2 & \rho u a^2 & uv & 0 \\ -\rho v & \rho u & \frac{v}{u} & \frac{v}{u}(u^2 - a^2) \end{bmatrix}$$

Eq. 2.1.5

The eigenvalues of [**A**] determine the characteristics direction and are [Anderson et al.] [39]

$$\lambda_1 = \frac{v}{u}\ ,\ \lambda_2 = \frac{v}{u}\ ,\ \lambda_3 = \frac{uv + a\sqrt{u^2 + v^2 - a^2}}{u^2 - a^2}\ ,\ \lambda_4 = \frac{uv - a\sqrt{u^2 + v^2 - a^2}}{u^2 - a^2}$$

Eq. 2.1.6

The matrix of left eigenvectors associated with these values of λ may be written as

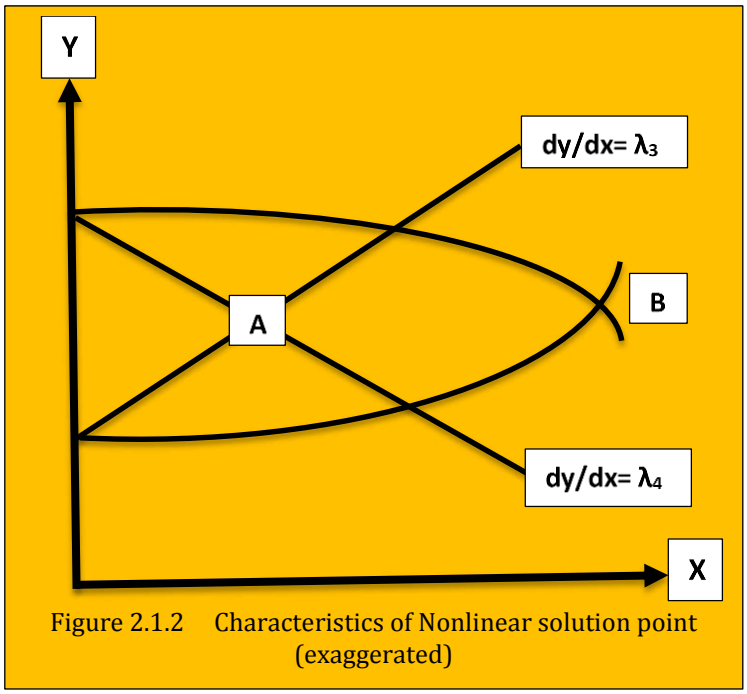


Figure 2.1.2 Characteristics of Nonlinear solution point (exaggerated)

[39] D. Anderson, J., Tannehill, R., Pletcher, "Computational Fluid Mechanics and Heat Transfer", ISBN 0-89116-471-5 – 1984.

$$[\mathbf{T}]^{-1} = \begin{bmatrix} \frac{\rho u}{a^2} & \frac{\rho v}{a^2} & 0 & 1 \\ \rho u & \rho v & 1 & 0 \\ -\frac{1}{\sqrt{u^2+v^2-a^2}} & +\frac{u}{v}\frac{1}{\sqrt{u^2+v^2-a^2}} & \frac{1}{\rho v a} & 0 \\ \frac{1}{\sqrt{u^2+v^2-a^2}} & -v\frac{u}{v}\frac{1}{\sqrt{u^2+v^2-a^2}} & \frac{1}{\rho v a} & 0 \end{bmatrix}$$

Eq. 2.1.7

We obtain the compatibility relations by pre-multiplying the original system by $[T]^{-1}$. These relations along the wave fronts are given by:

$$-v\frac{du}{ds_3} + u\frac{dv}{ds_3} + \frac{\beta}{\rho}\frac{dp}{ds_3} = 0 \quad \text{along} \quad \frac{dy}{dx} = \lambda_3 \qquad v\frac{du}{ds_4} - u\frac{dv}{ds_4} + \frac{\beta}{\rho}\frac{dp}{ds_4} = 0 \quad \text{along} \quad \frac{dy}{dx} = \lambda_4$$

Eq. 2.1.8

These are an ordinary differential equations which holds along the characteristic with slope λ_3, λ_4, while arc length along this characteristics is denoted by s_3, s_4. In contrast to linear example, the analytical solution for characteristics is not known for the general nonlinear problem. It is clear that we must numerically integrate to determine the shape of the characteristics in step by step manner. Consider the characteristic defined by λ_3. Stating at an initial data surface, the expression can be integrated to obtain the coordinates of next point at the curve. At the same time, the differentials equation defining the other wave front characteristics can be integrated. For a simple first-order integration this provide us with two equations for wave front characteristics. From this expressions, we determine the coordinate of their intersection, point A. Once the point A is known, the compatibility relations, are integrated along the characteristics to this point. This provide a system of equations at point A. This is a first-order estimate of the both the location of point A and the associated flow variables. In the next step, the new intersection point B can be calculated which now includes the nonlinear nature of the characteristic curve. In a similar manner, the dependent variables at point B are computed. Since the problem is nonlinear, the final intersection point B does not necessary appear at the same value of x for all solution points. Consequently, the solution is usually interpolated onto an x-constant surface before the next integration step. This requires additional logic and added considerably to the difficulty in turning an accurate solution [Anderson et al.][40].

2.2 Shock Capturing Method (Inviscid)

2.2.1 Explicit McCormack Method

The shock capturing schemes are most used techniques for computing inviscid flows with shocks. In this approach, the Euler's equations are cast in conservative form for and shocks wave predicted as part of solution. The shock waves predicted by these methods are smeared over several grid intervals bur the simplicity of the approach may outweigh the slight compromised in the results. First we consider 2D steady supersonic flow. We assume the x-axis forms the body surface and marching direction. The equations can be written as

[40] D. Anderson, J., Tannehill, R., Pletcher, "*Computational Fluid Mechanics and Heat Transfer*", ISBN 0-89116-471-5 – 1984.

$$\frac{\partial \mathbf{E}}{\partial x}+\frac{\partial \mathbf{F}}{\partial y}=0 \quad \text{where} \quad \mathbf{E}=\begin{bmatrix} \rho u \\ p+\rho u^2 \\ \rho uv \end{bmatrix} \quad \text{and} \quad \mathbf{F}=\begin{bmatrix} \rho v \\ \rho uv \\ p+\rho v^2 \end{bmatrix}$$

Eq. 2.2.1

This is a hyperbolic PDE and explicit Mac Cormack scheme would be a good choice as

$$\mathbf{E}_i^{\overline{n+1}}=\underbrace{\mathbf{E}_i^n-\frac{\Delta x}{\Delta y}\left(\mathbf{F}_{i+1}^n-\mathbf{F}_i^n\right)}_{\text{predictor}} \quad \rightarrow \quad \mathbf{E}_i^{n+1}=\underbrace{\frac{1}{2}\left[\mathbf{E}_i^n+\mathbf{E}_i^{\overline{n+1}}-\frac{\Delta x}{\Delta y}\left(\mathbf{F}_i^{\overline{n+1}}-\mathbf{F}_{i-1}^{\overline{n+1}}\right)\right]}_{\text{corrector}}$$

Eq. 2.2.2

At the end of predictor/corrector step, **E** must be decoded to obtain the primitive variables. This way, the flux vector can be evaluated for the next integration step. The y-components of velocity v is immediately known

$$v=\frac{E_3}{E_1} \quad , \quad \rho=\frac{E_1}{u} \quad , \quad u=\frac{\gamma}{\gamma+1}\frac{E_2}{E_1}\pm\sqrt{\left(\frac{\gamma}{\gamma+1}\frac{E_2}{E_1}\right)^2-\frac{\gamma-1}{\gamma+1}(2H-v^2)} \quad , \quad p=E_2-\rho u^2$$

Eq. 2.2.3

Having completed this process, **F** can be recalculate and the next step can be implemented. Readers also should consult the work by [Mahdi and Al-Kwarizmi][41] for shock capturing on a 2D compressible unsteady Euler equation. Using a Mac Cormack's time marching method that an explicit finite-difference technique. The test case chosen is that of a transonic and supersonic flow through a channel with a circular arc bump on the lower wall, half wedge and extended compression corner. (see **Figure 2.2.1**).

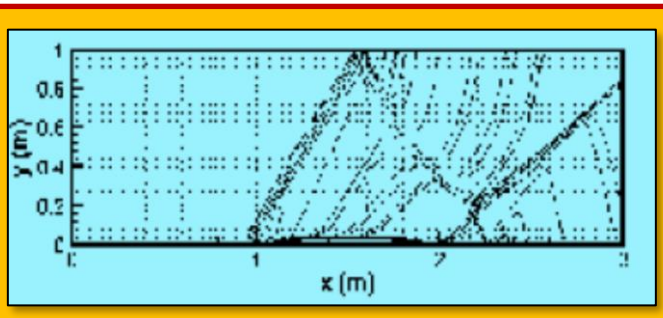


Figure 2.2.1 Supersonic Flow Over Circular Arc with Inlet M=1.4 ; Courtesy of [Mahdi and Al-Kwarizmi]

2.2.2 1D Upwind Flux-Splitting Scheme (Steger-Warming)

This is usually used as a ***Shock Capturing Method*** and belongs to a class of solutions as ***AUSM (Advection Upstream Splitting Method).*** It is developed as a numerical inviscid flux function for solving a general system of conservation equations. It is based on the upwind concept and was motivated to provide an alternative approach to other upwind methods, such as the ***Godunov*** method, flux difference splitting methods by [Roe][42], [Solomon and Osher], flux vector splitting methods by [Van Leer][43], and [Steger and Warming. The AUSM first recognizes that the inviscid flux consist of two physically distinct parts, i.e., convective and pressure fluxes. The former is associated with the flow (advection) speed, while the latter with the acoustic speed; or respectively classified as the linear and nonlinear fields. ***Currently, the convective and pressure fluxes are formulated using the eigenvalues of the flux Jacobian matrices***. To illustrates the flux splitting concepts, consider the one-dimensional system of hyperbolic PDE,

---

[41] Ahmed Fouad Mahdi, Al-Kwarizmi, "*Shock Wave Capturing Numerically in Two-Dimensional Supersonic Wind Tunnel for Different Configurations*", Eng. & Tech. Journal, Vol. 30, No.3 , 2012.

[42] Roe, *Flux Differencing Scheme* Details.

[43] Bram van Leer, "*Flux-Vector Splitting for the 1990s*", The University of Michigan, N91-21073.

$$\frac{\partial \mathbf{U}}{\partial t}+\frac{\partial \mathbf{E}}{\partial x}=0 \quad \text{define} \quad \mathbf{E}=\mathbf{E}^{+}+\mathbf{E}^{-} \quad \text{so} \quad \frac{\partial \mathbf{U}}{\partial t}+\frac{\partial \mathbf{E}^{+}}{\partial x}+\frac{\partial \mathbf{E}^{-}}{\partial x}=0$$

Eq. 2.2.4

Where the plus indicates a backward differencing, a minus forward differencing is required. The split flux can be used either for explicit or implicit algorithms. For example, a second-order upwind, predictor/corrector scheme (Beam & Warming, 1975) used as

$$\text{Predicter} \quad U_j^{\overline{n+1}} = U_j^n - \frac{\Delta t}{\Delta x}\left(\nabla E_j^{+} + \Delta E_j^{-}\right)$$

$$\text{Corrector} \quad U_j^{n+1} = \frac{1}{2}\left[U_j^n + U_j^{\overline{n+1}} - \frac{\Delta t}{\Delta x}\left(\nabla^2 E_j^{+n} + \nabla E_j^{\overline{n+1}}\right) + \frac{\Delta t}{\Delta x}\left(\Delta^2 E_j^{-n} - \Delta E_j^{-\overline{n+1}}\right)\right]$$

Eq. 2.2.5

An implicit algorithm using the trapezoidal rule is derived

$$\left\{[I] + \frac{\Delta t}{2\Delta x}\left(\nabla [A_j]^{+} + \Delta [A_j]^{-}\right)\right\}\Delta U_j^n = -\frac{\Delta t}{\Delta x}\left[\nabla E^{+} + \Delta E^{-}\right]$$

Eq. 2.2.6

This algorithm is 1st order accurate in space, 2nd order accurate in time. The left hand side can be AF and ignoring the 2nd order cross terms

$$\left([I] + \frac{\Delta t}{2\Delta x}\nabla [A_j]^{+}\right)\left([I] + \frac{\Delta t}{2\Delta x}\Delta [A_j]^{-}\right)\Delta U_j^n = -\frac{\Delta t}{\Delta x}\left[\nabla E^{+} + \Delta E^{-}\right]$$

$$\left([I] + \frac{\Delta t}{2\Delta x}\nabla [A_j]^{+}\right)\Delta U_j' = -\frac{\Delta t}{\Delta x}\left[\nabla E^{+} + \Delta E^{-}\right]$$

$$\left([I] + \frac{\Delta t}{2\Delta x}\Delta [A_j]^{-}\right)\Delta U_j^n = \Delta U_j'$$

Eq. 2.2.7

In this equations, each one dimensional sweep required the solution of two block bi-diagonal (here 1) system. Usually the advantages of AF is more pronounced in multi-dimension problem. The use of split flux techniques for shock capturing applications produces somewhat better results than standard central difference schemes.

2.2.3 Total Variation Diminishing (TVD) as other Upwind Schemes

In numerical methods, ***Total Variation Diminishing (TVD)*** is a property of certain discretization schemes used to solve ***hyperbolic*** partial differential equations. To capture the variation fine grids (Δx = very small) are needed and the computation becomes heavy and therefore un-economic. The use of coarse grids with central difference scheme, upwind scheme, hybrid difference scheme, and power law scheme gives false shock predictions. TVD scheme enables sharper shock predictions on coarse grids saving computation time and as the scheme preserves monotonicity there are no spurious oscillations in the solution. In systems described by partial differential equations, such as the following 1D hyperbolic advection equation,

$$\frac{\partial \mathbf{U}}{\partial t} + [\mathbf{A}]\frac{\partial \mathbf{U}}{\partial x} \quad \text{where } [\mathbf{A}] = \frac{\partial \mathbf{E}}{\partial \mathbf{U}}$$

$$TV = \int \left|\frac{\partial \mathbf{U}}{\partial x}\right| dx = \sum \left|U_{j+1} - U_j\right| \quad \text{subject to} \quad TV(U^{n+1}) \le TV(U^n)$$

Eq. 2.2.8

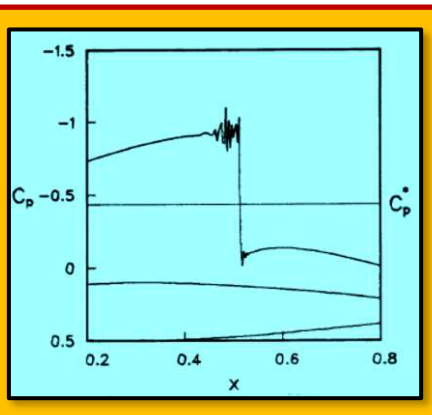
Conventional shock capturing schemes for the solution of nonlinear hyperbolic conservation laws are linear and L_2-norm stable when considered in the constant coefficient case[44]. There are three major difficulties in using such schemes to compute discontinuous solutions of a nonlinear system, such as the compressible Euler equations:

1. Schemes that are second (or higher) order accurate may produce oscillations wherever the solution is not smooth.
2. Nonlinear instabilities may develop in spite of the &-stability in the constant coefficient case.
3. The scheme may select a nonphysical solution.

It is well known that monotone conservative difference schemes always converge and that their limit is the physical weak solution satisfying an entropy inequality. Thus monotone schemes are guaranteed not to have difficulties (2) and (3). However, monotone schemes are only first-order accurate. Consequently, they produce rather crude approximations whenever the solution varies strongly in space or time. When using a second- (or higher) order accurate scheme, some of these difficulties can be overcome by adding a hefty amount of numerical dissipation to the scheme. Unfortunately, this process brings about an irretrievable loss of information that exhibits itself in degraded accuracy and smeared discontinuities. Thus, a typical complaint about conventional schemes which are developed under the guidelines of linear theory is that they are not robust and/or not accurate enough. The class of TVD schemes contains monotone schemes, but is significantly larger as it includes second-order accurate schemes.

2.2.3.1 The Upwind Connection to Explicit Artificial Dissipation

If the viscosity isn't large enough, velocity oscillations about the correct mean velocity are observed to develop behind a shock. These oscillations can be interpreted as a macroscopic version of heat energy, i.e., fluctuating kinetic energy in place of fluctuating molecular energy[45]. These upwind schemes all claim (with good justification) to be physically consistent since they follow in some sense the characteristics of the flow. They in general can be shown to produce sharp oscillation free shocks without added artificial dissipation. **Figure 2.2.2** shows the coefficient of pressure for a shock without artificial dissipation (left); with artificial dissipation (right). Also these schemes have an inherent amount of internal dissipation, due to the one sided differences,



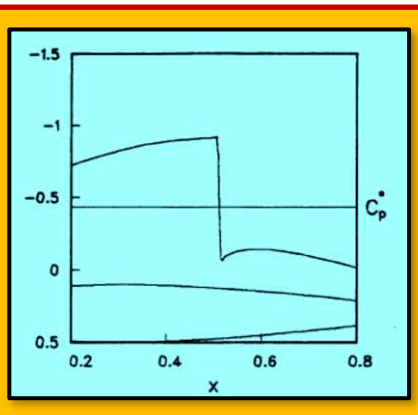


Figure 2.2.2 Coefficient of Pressure for a Shock

[44] H. C. Yee, NASA, "*Implicit Total Variation Diminishing (TVD) schemes for steady-state calculations*", Journal of Computational Physics, March 1985.

[45] Flow Science Blog, "*What are Artificial and Numerical Viscosities?*"

which cannot be modified or decreased. It may be advantageous to have the flexibility of a simple central difference scheme with a controllable amount of artificial dissipation. It can be shown that the upwind schemes have an equivalence to central difference schemes with added dissipation. The central schemes are much simpler and more flexible and are therefore desirable if the dissipation can be added in an analogous fashion to the upwind schemes[46]. This is often called the Implicit Artificial Dissipation as opposed to Explicit one which purposely added to a difference equation and, been discussed previously. In general, the flux of any scheme can be written as

$$\underbrace{f}_{\text{FTCS flux}} + \underbrace{f - f_{FTCS}}_{\text{Flux due to Artificial Viscosity}} \qquad \text{FTCS} = \text{Foward Time \& Central Space}$$

Eq. 2.2.9

2.2.4 3D Unsteady Euler Equation Solutions Using Flux Vector Splitting

To compute the flow of a store released from an aircraft, it is desirable to solve the unsteady Euler equations on a grid that moves with the store along its trajectory [Whitfield][47]. The objective here is to solve the ***3D unsteady Euler equations on a time-dependent grid***. The computations presented here are for bodies whose motion is prescribed. As the solution advances in *time,* body motion could be determined from the Euler equations by using force and moment coefficients obtained from the Euler solution in the dynamic equations of motion for the body to determine the trajectory of the body. The objective of this report is to present and verify dynamic-grid Euler equations computations. Following the development in [Whitfield], the strong conservation law of the Euler equations in curvilinear coordinates can be written as:

$$\frac{\partial \mathbf{Q}}{\partial \tau} + \frac{\partial \mathbf{F}}{\partial \xi} + \frac{\partial \mathbf{G}}{\partial \eta} + \frac{\partial \mathbf{H}}{\partial \zeta} = 0 \qquad (7.18)$$

$$\mathbf{Q} = \mathbf{J}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{bmatrix}, \quad \mathbf{F} = \mathbf{J}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ U(e+p) - \xi_t p \end{bmatrix}, \quad \mathbf{G} = \mathbf{J}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ V(e+p) - \eta_t p \end{bmatrix}, \quad \mathbf{H} = \mathbf{J}\begin{bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho v W + \zeta_y p \\ \rho w W + \zeta_z p \\ W(e+p) - \zeta_t p \end{bmatrix}$$

$$\text{and} \quad \xi = \xi(x, y, z), \ \eta = \eta(x, y, z), \ \zeta(x, y, z), \ t = \tau \quad \text{and} \quad \mathbf{J} = \frac{\partial(\xi, \eta, \zeta)}{\partial(x, y, z)}$$

Matrices of transformation are:

$$\xi_x = \mathbf{J}(y_\eta z_\zeta - y_\zeta z_\eta), \ \xi_y = \mathbf{J}(x_\eta z_\zeta - x_\zeta z_\eta), \ \xi_z = \mathbf{J}(x_\eta y_\zeta - x_\zeta y_\eta)$$

$$\eta_x = \mathbf{J}(y_\xi z_\zeta - y_\zeta z_\xi), \ \eta_y = \mathbf{J}(x_\xi z_\zeta - x_\zeta z_\xi), \ \eta_z = \mathbf{J}(x_\xi y_\zeta - x_\zeta x_\xi)$$

$$\zeta_x = \mathbf{J}(y_\xi z_\eta - y_\eta z_\xi), \ \zeta_y = \mathbf{J}(x_\xi z_\eta - x_\eta z_\xi), \ \zeta_z = \mathbf{J}(x_\xi y_\eta - x_\eta y_\xi)$$

with contvariant velocity components

$$U = \xi_x u + \xi_y v + \xi_z w \quad , \quad V = \eta_x u + \eta_y v + \eta_z w \quad , \quad W = \zeta_x u + \zeta_y v + \zeta_z w$$

Eq. 2.2.10

The 3D unsteady Euler equations are a ***hyperbolic*** system of five equations and hence have five characteristic velocities in each of the three spatial directions. These characteristic velocities are

---

[46] T. H. Pulliam, "*Solution Methods in Computational Fluid Dynamics*", NASA Ames Research Center, USA.

[47] David, L., Whitfield, "*Three-Dimensional Unsteady Euler Equation Solutions Using Flux Vector Splitting*", NASA-CR-173254 19840008789.

determined from the quasilinear form of **Eq. 2.2.10**. A finite volume discretization of **Eq. 2.2.10** balances the increase of the conserved quantity in a computational cell, or volume, with the flux of the quantity through the surface of the cell. Assuming the dependent variables are constant in the interior of cell i, j, k, and that the flux vectors **F**, **G**, and **H** are constant over the constant ξ , constant η and constant surfaces of the ζ , respectively, an explicit discretization yields:

$$\underbrace{\left(Q^{n+1}_{i,j,k} - Q^{n}_{i,j,k}\right)}_{\frac{\Delta Q}{\Delta \tau}} \Delta\xi\Delta\eta\Delta\zeta = -\left(F^{n}_{i+\frac{1}{2},j,k} - F^{n}_{i-\frac{1}{2},j,k}\right)\Delta\eta\Delta\zeta$$
$$-\left(G^{n}_{i,j+\frac{1}{2},k} - G^{n}_{i,j-\frac{1}{2},k}\right)\Delta\xi\Delta\zeta \quad -\left(H^{n}_{i,j,k+\frac{1}{2}} - H^{n}_{i,j,k-\frac{1}{2}}\right)\Delta\xi\Delta\eta$$

Eq. 2.2.11

and time step Δτ is given in [Whitfield], or using the classical ***Runge-Kutta*** scheme. The central difference operator is been used in **Eq. 2.2.11** indicates the flux vectors are evaluated at cell faces in this finite volume formulation. The numerical scheme used is a finite volume version of the second-order upwind scheme of [Beam & Warming][48], while the present scheme is an extension of that used by [Deese][49].

2.2.5 Flux Splitting

Hyperbolic partial differential equations, such as the Euler equations, are characterized by the existence of a limited domain of dependence. The solution at a point does not depend on every other point in the field; this means that information travels only in certain characteristic directions. Numerical schemes intended to solve hyperbolic equations are usually enhanced by insuring that the numerical method propagates information in the direction specified by the partial differential equation. This can be done by using an upwind method, or one in which the difference operator is taken in the direction from which the information should come. Stability properties are often improved by unwinding, and it is usually unnecessary to add smoothing terms or artificial viscosity to an upwind method. The 3D Euler equations, **Eq. 2.2.10**, are a hyperbolic system of five equations and hence have five characteristic velocities in each of the three spatial directions. These characteristic velocities are determined from the quasilinear form of **Eq. 2.2.10**. Details of splitting and test cases can be obtained from [Whitfield]. It is suffice to show that five eigenvalues are:

$$\lambda^1_k = \lambda^2_k = \lambda^2_k = \underbrace{ku + kv + kz}_{\theta_k} \quad , \quad \lambda^4_k = \theta_k + c|\nabla k| \quad , \quad \lambda^5_k = \theta_k - c|\nabla k|$$
$$\text{where} \quad k = \{\xi, \eta, \zeta\} \quad \text{and} \quad |\nabla k| = \left(k_x^2 + k_y^2 + k_z^2\right)^{1/2}$$

Eq. 2.2.12

And c is speed of sound. As evident, two of the five right eigenvectors form a 2D subspace, within the general 5-dimensional space spanned by all right eigenvectors, and that every member of this subspace is itself an eigenvector. This phenomenon was attributed to the fact that their corresponding eigenvalues are repeated, which creates a "symmetry" within the eigenvector space. Although it may seem difficult to visualize any symmetry within a 5-dimensional vector space, part of this symmetry reveals itself when we geometrically interpret the 2-dimensional subspace as a

---

[48] Warming, R. F. , and Beam, K. M., "*Upwind Second·-Order Difference Schemes .md Applications in Aerodynamic Flows,*" AIAA ,Journal, Vol. 14. No.9, September 1976, pp. 1241-1249.

[49] Deese, J. E, "*Numerical Experi1D:ents with the Split-Flux··Vector Form of the Euler Equations,*" AIAA Paper No. 83-0122, January 1983.

plane. The Euler equations contain three types, or families of waves, one for every distinct eigenvalue. Each family of waves carries a different *signal*. The waves traveling at the speed of the flow ($\lambda_k^{1,2,3}$) are called entropy waves, their signal being entropy, whereas waves traveling at the speed of sound relative to the flow ($\lambda_k^{4,5}$) are called acoustic waves. Unfortunately, the signal carried by acoustic waves is not quantifiable in simple thermodynamic terms, but let us just say that they carry acoustic information [Rohde] [50].

2.3 The Riemann Problem

The Riemann problem consists of a conservation law together with piecewise constant data having a single discontinuity (i.e., shocks)[Kong][51]. Here we will discuss the problem for a linear system, and then discuss how the Riemann problem for the Euler equations, addressing specific problems that will be focused on during comparisons of schemes. The initial state of the system is defined as

$$u(x, t = 0) = \begin{cases} u_L & \text{for } x \leq 0 \\ u_R & \text{for } x \geq 0 \end{cases}$$

Eq. 2.3.1

To put into perspective, the initial state is constant for all negative x, and constant for all positive x, but differs between left and right. In the one-dimensional case we can consider this problem as a gas with one temperature and density located to the left of a removable wall and another gas with another temperature and density to the right of the wall. At time t = 0 the wall is instantly removed and the results are observed. In numerical analysis Riemann problems appear in a natural way in finite volume methods for the solution of conservation law equations due to the discreteness of the grid. For this we use approximate Riemann solvers, since iterative schemes are too costly some assumptions must be made, which will be discussed further in the following section.

2.3.1 Roe Approximate Riemann Solver

The Roe solver, devised by [Roe][52], is an approximate Riemann solver based around the Godunov scheme and works by looking for an estimate for the inter cell numerical flux or Godunov flux at the interface between two computational cells. ***It essentially determine the approximate solution by solving a constant coefficient linear system instead of the original nonlinear system***. We study the break-up of a single discontinuity. Those cases where $\mathbf{F}(\mathbf{u})$ is linear are well-known and essentially trivial. Those cases where u is scalar and **F** is non-linear can be surprisingly complicated, but have been thoroughly investigated. If **u** is a vector and **F** is non-linear, then the problem involves non-linear algebraic equations together with, usually, logical conditions which express the fact that a given member of the wave system may be present either as a shockwave or as an expansion fan. In general, the most efficient way to solve these equations will depend on the system of conservation laws from which they derive; ingenuity is required to exploit special features of each individual system. The usual way of incorporating the Riemann problem into the numerical solution is to take ($\mathbf{u}^n$, $\mathbf{u}^{n+i}$), for each i in turn, as pairs of states defining a sequence of Riemann problems, which are then thought of as providing information about the solution within each interval (i , i+1). Various individual methods are then distinguished by the way in which this information is put to use. Here, we consider approximate solutions which are exact solutions to an approximate problem:

---

[50] Axel Rohde, "*Eigenvalues And Eigenvectors Of The Euler Equations In General Geometries*", AIAA 2001-2609.

[51] Charlotte Kong, "*Comparison of Approximate Riemann Solvers*", A dissertation submitted in partial fulfilment of the requirement for the degree of Master of Science in Mathematical and Numerical Modelling of the Atmosphere and Oceans, 2011.

[52] P. L. Roe, "*Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes*", Journal of Computational Physics 43, 357-372 (1981).

$$u_t + \mathbf{A}u_x = 0 \quad \equiv \quad u_t + \hat{\mathbf{A}}u_x = 0 \quad , \quad \mathbf{A} = \frac{\partial \mathbf{F}}{\partial \mathbf{u}}$$

$$\text{where} \quad \hat{\mathbf{A}} = \frac{1}{2}(\mathbf{A}_L + \mathbf{A}_R) = \mathbf{A}\frac{1}{2}(\mathbf{u}_L + \mathbf{u}_R)$$

$$\mathbf{f}_{Roe} = \frac{1}{2}\left[\mathbf{F}(\mathbf{u}_L) + \mathbf{F}(\mathbf{u}_R) - \sum_p \left|\tilde{\lambda}^p\right| \tilde{\alpha}^p \tilde{\mathbf{R}}^p\right] \quad \text{where} \quad \tilde{\alpha}^p = \tilde{\mathbf{L}}^p(\mathbf{u}_R - \mathbf{u}_L)$$

and $\tilde{\lambda}^p, \tilde{\mathbf{R}}^p, \tilde{\mathbf{L}}^p$ being the eigenvalues and the right and left eigenvectors of $\hat{\mathbf{A}}$

and p = number of equations

Eq. 2.3.2

For further information, reader should consult the [Roe][329].

2.4 Solution Methods for Compressible N-S Equation

The unsteady compressible N-S equations are a mixed set of hyperbolic-parabolic equations, while the unsteady incompressible is a mixed set of elliptic-parabolic. As a consequence, different numerical techniques should be used to solve N-S equations in compressible vs incompressible flow regimes. A general formulation in vector form is given by **Eq. 2.4.1**). We start with compressible methods first and divide the categories to ***Explicit*** and ***Implicit*** schemes.

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} = 0 \quad \text{where } \mathbf{U}, \mathbf{E}, \mathbf{F}, \text{ and } \mathbf{G} \text{ are given by}$$

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E_t \end{bmatrix} \quad , \quad \mathbf{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p - \tau_{xx} \\ \rho uv - \tau_{xy} \\ \rho uw + p - \tau_{xz} \\ (E_t + p)u - u\tau_{xx} - v\tau_{xy} - w\tau_{xz} + q_x \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} \rho v \\ \rho uv - \tau_{xy} \\ \rho v^2 + p - \tau_{yy} \\ \rho vw - \tau_{yz} \\ (E_t + p)v - u\tau_{xy} - v\tau_{yy} - w\tau_{yz} + q_y \end{bmatrix} \quad , \quad \mathbf{G} = \begin{bmatrix} \rho w \\ \rho uw - \tau_{xz} \\ \rho vw - \tau_{yz} \\ \rho w^2 + p - \tau_{zz} \\ (E_t + p)w - u\tau_{xz} - v\tau_{yz} - w\tau_{zz} + q_z \end{bmatrix}$$

where

$$\tau_{xx} = \frac{2}{3}\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} - \frac{\partial w}{\partial z}\right) \quad , \quad \tau_{yy} = \frac{2}{3}\mu\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} - \frac{\partial w}{\partial z}\right) \quad , \quad \tau_{zz} = \frac{2}{3}\mu\left(2\frac{\partial w}{\partial z} - \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right)$$

$$\tau_{xy} = \tau_{yx} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \quad , \quad \tau_{xz} = \tau_{zx} = \mu\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right) \quad , \quad \tau_{yz} = \tau_{zy} = \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)$$

$$q_x = -k\frac{\partial T}{\partial x} \quad , \quad q_y = -k\frac{\partial T}{\partial y} \quad , \quad q_z = -k\frac{\partial T}{\partial z} \quad \text{and} \quad E_t = \rho\left(e + \frac{u^2 + v^2 + w^2}{2}\right)$$

Eq. 2.4.1

By setting density to constant, we obtain the ***incompressible N-S equation for Newtonian flow*** as:

$$\nabla.\mathbf{V} = 0$$

$$\frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V}.\nabla)\mathbf{V} + \nabla p = \nu\nabla^2\mathbf{V} + \mathbf{g} \quad \text{where} \quad \mathbf{V} = \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \quad \text{and} \quad \mathbf{g} = \begin{Bmatrix} g_x \\ g_y \\ g_z \end{Bmatrix}$$

Eq. 2.4.2

2.4.1 General Transformation

These equations can be expressed in terms of a generalized non-orthogonal curvilinear coordinated system

$$\xi = \xi(x, y, z) \;, \eta = \eta(x, y, z) \;, \zeta(x, y, z) \;, \quad t = t$$

$$\mathbf{J} = \frac{\partial(\xi, \eta, \zeta)}{\partial(x, y, z)} = \frac{1}{\mathbf{J}^{-1}} = \frac{1}{\dfrac{\partial(x, y, z)}{\partial(\xi, \eta, \zeta)}} = 1/\begin{vmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{vmatrix}$$

where matrices are :

$$\xi_x = \mathbf{J}(y_\eta z_\zeta - y_\zeta z_\eta) \;, \quad \xi_y = \mathbf{J}(x_\eta z_\zeta - x_\zeta z_\eta) \;, \quad \xi_z = \mathbf{J}(x_\eta y_\zeta - x_\zeta y_\eta)$$

$$\eta_x = \mathbf{J}(y_\xi z_\zeta - y_\zeta z_\xi) \;, \quad \eta_y = \mathbf{J}(x_\xi z_\zeta - x_\zeta z_\xi) \;, \quad \eta_z = \mathbf{J}(x_\xi y_\zeta - x_\zeta x_\xi)$$

$$\zeta_x = \mathbf{J}(y_\xi z_\eta - y_\eta z_\xi) \;, \quad \zeta_y = \mathbf{J}(x_\xi z_\eta - x_\eta z_\xi) \;, \quad \zeta_z = \mathbf{J}(x_\xi y_\eta - x_\eta y_\xi)$$

Eq. 2.4.3

The generalized transformation to the compressible N-S equations (**Eq. 2.4.1**) written in curvilinear vector form [Anderson et al.][53] as:

$$\mathbf{U_t} + \xi_x\mathbf{E_\xi} + \eta_x\mathbf{E_\eta} + \zeta_x\mathbf{E_\zeta} + \xi_y\mathbf{F_\xi} + \eta_y\mathbf{F_\eta} + \zeta_y\mathbf{F_\zeta} + \xi_z\mathbf{G_\xi} + \eta_z\mathbf{G_\eta} + \zeta_z\mathbf{G_\zeta} = \mathbf{0}$$

$$\left(\underbrace{\frac{\mathbf{U}}{\mathbf{J}}}_{\mathbf{U_1}}\right)_t + \left(\underbrace{\frac{\mathbf{E}\xi_x + \mathbf{F}\xi_y + \mathbf{G}\xi_z}{\mathbf{J}}}_{\mathbf{E_1}}\right) + \left(\underbrace{\frac{\mathbf{E}\eta_x + \mathbf{F}\eta_y + \mathbf{G}\eta_z}{\mathbf{J}}}_{\mathbf{F_1}}\right) + \left(\underbrace{\frac{\mathbf{E}\zeta_x + \mathbf{F}\zeta_y + \mathbf{G}\zeta_z}{\mathbf{J}}}_{\mathbf{G_1}}\right) = 0$$

$$\frac{\partial \mathbf{U_1}}{\partial t} + \frac{\partial \mathbf{E_1}}{\partial \xi} + \frac{\partial \mathbf{F_1}}{\partial \eta} + \frac{\partial \mathbf{G_1}}{\partial \zeta} = 0$$

Eq. 2.4.4

By no means is this form of representation is conclusive. Many texts and researchers are attain their own representation as they see to be relevant. But the concepts should be the same. This is the strong conversation form of governing equation which is best suited for finite differencing schemes.

2.4.2 Explicit Scheme (Mac Cormack)

The ***Mac Cormack Explicit***, the ***predictor/corrector*** which gained a lots of popularity among

[53] Anderson, Dale A; Tannehill, John C; Plecher Richard H; 1984:*"Computational Fluid Mechanics and Heat Transfer"*, Hemisphere Publishing Corporation.

research cycles in 70s and early 80s. For an excellent discussion on these and many more, reader is advised review[54]. The compressible N-S equations results:

$$\frac{\partial \mathbf{U}}{\partial t}+\frac{\partial \mathbf{E}}{\partial x}+\frac{\partial \mathbf{F}}{\partial y}+\frac{\partial \mathbf{G}}{\partial z}=0$$

Predictor: $$\mathbf{U}_{i,j,k}^{\overline{n+1}}=\mathbf{U}_{i,j,k}^{n}-\frac{\Delta t}{\Delta x}\left(\mathbf{E}_{i+1,j,k}^{n}-\mathbf{E}_{i,j,k}^{n}\right)-\frac{\Delta t}{\Delta y}\left(\mathbf{F}_{i,j+1,k}^{n}-\mathbf{F}_{i,j,k}^{n}\right)-\frac{\Delta t}{\Delta z}\left(\mathbf{G}_{i,j,k+1}^{n}-\mathbf{G}_{i,j,k}^{n}\right)$$

Corrector: $$\mathbf{U}_{i,j,k}^{n+1}=\frac{1}{2}\left[\mathbf{U}_{i,j,k}^{n}+\mathbf{U}_{i,j,k}^{\overline{n+1}}-\frac{\Delta t}{\Delta x}\left(\mathbf{E}_{i,j,k}^{\overline{n+1}}-\mathbf{E}_{i-1,j,k}^{\overline{n+1}}\right)-\frac{\Delta t}{\Delta y}\left(\mathbf{F}_{i,j,k}^{\overline{n+1}}-\mathbf{F}_{i,j-1,k}^{\overline{n+1}}\right)-\frac{\Delta t}{\Delta z}\left(\mathbf{G}_{i,j,k}^{\overline{n+1}}-\mathbf{G}_{i,j,k-1}^{\overline{n+1}}\right)\right]$$

Eq. 2.4.5

Where x= i Δx, y= j Δy and z = k Δz. This is 2nd order accurate in both space and time. The choice of Δt for stability consideration is obtained from[55] as:

$$\Delta t\le\frac{\alpha(\Delta t)_{CFL}}{1+2/\mathrm{Re}_{\Delta}}\quad\text{where}\quad(\Delta t)_{CFL}<\left[\frac{|u|}{\Delta x}+\frac{|v|}{\Delta y}+\frac{|w|}{\Delta z}+a\sqrt{\frac{1}{(\Delta x)^2}+\frac{1}{(\Delta y)^2}+\frac{1}{(\Delta z)^2}}\right]^{-1}$$

$$\text{with}\quad\mathrm{Re}_{\Delta}=\mathrm{Min}(\mathrm{Re}_{\Delta x},\mathrm{Re}_{\Delta y},\mathrm{Re}_{\Delta z})\quad\mathrm{Re}_{\Delta x}=\frac{\rho|u|\Delta x}{\mu}\quad\mathrm{Re}_{\Delta y}=\frac{\rho|v|\Delta y}{\mu}\quad\mathrm{Re}_{\Delta z}=\frac{\rho|w|\Delta z}{\mu}$$

$$\text{and}\quad a=(\gamma p/\rho)^{0.5}\quad,\quad\text{safty factor }\alpha\approx 0.9$$

Eq. 2.4.6

The explicit scheme is 2nd order accurate in both space and time. In the present form, forward differencing are used for all special derivatives in the predictor step while backward differences are used in the corrector step. The forward and backward differencing can be alternated between predictor and corrector steps as well as between the three spatial derivatives in sequential fashion. This eliminates any bias due to the on sided differencing. Moreover, the derivatives appearing in viscous terms of **E**, **F** and **G** must be differenced correctly in order to maintain 2nd order accuracy. The x derivative terms appearing in **E** are differenced in the opposite direction to that for d**E**/dx while the y-derivative and z are approximated with central differencing. Likewise, the y derivative terms appearing in **F** and z derivative terms appearing in **G** are differenced in opposite direction to that used for dF/dy and d**G**/dz respectively. Cross-derivative terms in **F** and **G** are approximated with central differencing. After each predictor or corrector step, the primitive variables (ρ, u, v, w, e, p, T) can be found be decoding the **U** vector. McCormack modified the original method by splitting the original McCormack scheme into a sequence of one-dimensional operations[56]. Thus, it become possible to advance the solution in each direction (Δt_x, Δt_y, and Δt_z) with large differences in mesh spacing (Δx, Δy, Δz). The explicit Mac Cormack algorithm is a suitable method for solving both steady and unsteady flows at moderate to low Reynolds numbers. However, it is not satisfactory method for

---

[54] Anderson, Dale A; Tannehill, John C; Plecher Richard H; 1984:"*Computational Fluid Mechanics and Heat Transfer"*, Hemisphere Publishing Corporation.

[55] Tannehill, J, C., Holst, T, L, and Rakich, J, V, "Numerical Computation of Two-Dimensional Viscous Blunt Body Flows and Impinging Shock", AIAA Paper 75-154, Pasadena, Ca, USA, 1975.

[56] MacCormack, R, W, "*Numerica solution of the Interaction of a shock wave with Laminar boundary layer*", Proceding of 2nd International Conference in Numerical Methods in Fluid Dynamics, pp. 151-163.

solving high Reynolds number flows where the viscous regions become very thin. For these flows, the mesh must be highly refined to resolve the viscous regions. This leads to small time steps and longer computer time.

2.4.3 Case Study – Dual Block Applied to Navier-Stokes Equations in 2D

A 2D Navier-Stokes equation using ***explicit Mac Cormack*** method on multi-block structured mesh are investigated by [Almeida et al.][57], for steady state and unsteady state compressible fluid flows. The multi-block technique and generalized coordinate system are used to develop a numerical solver which can be applied for a large range of compressible flow problems on complex geometries without modifying the governing equations and numerical method. Besides that the numerical method is based on a finite difference approach and the generalized coordinates introduced allow the application of the boundary conditions easily. The subsonic flow over a backward facing step and supersonic flow over a curved ramp are presented, and the results are compared with the experimental and numerical data.

2.4.3.1 Governing Equations

The two-dimensional compressible Navier-Stokes equations in generalized coordinates system (x ,h) without body forces, mass diffusion, finite-rate chemical reactions, or external heat addition can be written in non-dimensional conservative law form as [Anderson et al.][58].

$$\frac{\partial \hat{\mathbf{Q}}}{\partial t} + \frac{\partial\left(\hat{\mathbf{E}}_{\mathbf{i}} - \hat{\mathbf{E}}_{\mathbf{v}}\right)}{\partial \xi} + \frac{\partial\left(\hat{\mathbf{F}}_{\mathbf{i}} - \hat{\mathbf{F}}_{\mathbf{v}}\right)}{\partial \eta} = 0$$

Eq. 2.4.7

where **Q** is the state vector of conservative variables defined as below and **Ê**i and ˆ**F**i are the inviscid flux vectors,

$$\hat{\mathbf{Q}} = \begin{bmatrix} \rho \\ \rho U \\ \rho V \\ E_t \end{bmatrix} \quad , \quad \hat{\mathbf{E}}_{\boldsymbol{i}} = \mathbf{J}^{-1} \begin{bmatrix} \rho U \\ \rho uU + \xi_x P \\ \rho vU + \xi_y P \\ (E_t + p)U \end{bmatrix} \quad , \quad \hat{\mathbf{E}}_{\boldsymbol{i}} = \mathbf{J}^{-1} \begin{bmatrix} \rho V \\ \rho uV + \eta_x P \\ \rho vV + \eta_y P \\ (E_t + p)V \end{bmatrix}$$

Eq. 2.4.8

The **E**v and **F**v are the viscous flux vectors in the x and h directions, which are given below

$$\hat{\mathbf{E}}_{\mathbf{v}} = \mathbf{J}^{-1} \begin{bmatrix} 0 \\ Re^{-1}\left(\xi_x \tau_{xx} + \xi_y \tau_{xy}\right) \\ Re^{-1}\left(\xi_x \tau_{xy} + \xi_y \tau_{yy}\right) \\ Re^{-1}\left(\xi_x \beta_x + \xi_y \beta_y\right) \end{bmatrix} \quad , \quad \hat{\mathbf{E}}_{\mathbf{i}} = \mathbf{J}^{-1} \begin{bmatrix} 0 \\ Re^{-1}\left(\eta_x \tau_{xx} + \eta_y \tau_{xy}\right) \\ Re^{-1}\left(\eta_x \tau_{xy} + \eta_y \tau_{yy}\right) \\ Re^{-1}\left(\eta_x \beta_x + \eta_y \beta_y\right) \end{bmatrix}$$

Eq. 2.4.9

where J is the Jacobian of the transformation, r is the density, u and v are the velocity components in the x and h coordinate directions, U and V are the contravariant velocities, E_t is total energy per unit of volume, ξx, ξy, η_x and η_y are the metrics of transformation, $\beta_x = \tau_{xx}u + \tau_{xy}v$, and $\beta_y = \tau_{xy}u + \tau_{yy}v$, p is

---

[57] Jeferson Osmar de Almeida, Diomar Cesar Lobão, Cleyton Senior Stampa, Gustavo Benitez Alvarez, "*Multi-block Technique Applied to Navier-Stokes Equations in Two Dimensions*", Original Article, DOI: 10.5433/1679-0375.2018v39n2p115.

[58] Anderson, D.; Tannehill, J.; Pletcher, R. *Computational fluid mechanics and heat transfer*, Hemisphere Publishing Corporation - McGraw Hill, 1984.

the static pressure and t describes the stress components for viscous flow. The Navier-Stokes equations are based on the universal law of conservation of mass, conservation of momentum and conservation of energy. However, to complete this system of equations is necessary to add an equation of state that can be written as:

$$p = (\gamma - 1)\left[E_t - \frac{1}{2}\rho\,(u^2 + v^2)\right]$$

Eq. 2.4.10

in which the fluid is considered a perfect gas. In the above equation, γ denotes the ratio of the specific heats. The dimensionless of the variables is performed to eliminate the scale problems. Please see [Anderson et al.][59].

2.4.3.2 Numerical Method

The Mac Cormack method[60] is used for solving the governing equations as given by **Eq. 2.4.7**. The numerical method is an explicit predictor-corrector scheme based on a finite difference formulation that has been used for compressible flow, and it has second-order of accuracy in both space and time. When the method is applied to the two-dimensional compressible Navier-Stokes equations given by **Eq. 2.4.7**, it can be written as

$$\text{Predictor:}\quad \widehat{Q}_{i,j}^{\overline{n+1}} = \widehat{\mathbf{Q}}_{i,j}^{n} - \frac{\Delta t}{\Delta \xi}\left(\hat{\mathbf{E}}_{i+1,j}^{n} - \hat{\mathbf{E}}_{i,j}^{n}\right) - \frac{\Delta t}{\Delta \eta}\left(\hat{\mathbf{F}}_{i,j+1}^{n} - \hat{\mathbf{F}}_{i,j}^{n}\right)$$

$$\text{Corrector:}\quad \widehat{Q}_{i,j}^{\overline{n+1}} = \frac{1}{2}\left[\widehat{\mathbf{Q}}_{i,j}^{n} + \widehat{\mathbf{Q}}_{i,j}^{\overline{n+1}} - \frac{\Delta t}{\Delta \xi}\left(\hat{\mathbf{E}}_{i,j}^{\overline{n+1}} - \hat{\mathbf{E}}_{i-1,j}^{\overline{n+1}}\right) - \frac{\Delta t}{\Delta \eta}\left(\hat{\mathbf{F}}_{i,j}^{\overline{n+1}} - \hat{\mathbf{F}}_{i,j-1}^{\overline{n+1}}\right)\right]$$

Eq. 2.4.11

where $\hat{\mathbf{E}} = \hat{\mathbf{E}}_i - \hat{\mathbf{E}}_v$ and $\hat{}\mathbf{F} = \hat{}\mathbf{F}_i - \hat{}\mathbf{F}_v$. The explicit Mac Cormack method requires the predictor be calculated first to n = 0 (initial conditions). As stated by [Roe][61] any flow which goes through Mach 1.0, the classical numerical methods (Finite Difference/Finite Volume) present physical discontinuities, so needs special numerical treatment. In the present work the flow is much greater than Mach 1.0, well developed supersonic flow. In this case the Mac Cormack method does not require such special treatment. Note that Mac Cormack method does not carry any special treatment for the convective terms (inviscid) or any other terms in the formulation, it is a very naive method. Here, the boundary conditions are defined as follows: inflow, outflow and lower and upper solid wall. The inflow condition, all variables are prescribed as Dirichlet boundary conditions in supersonic regime. For outflow condition, with subsonic flow, the value of the static pressure p is prescribed as Dirichlet boundary conditions and all other variables are extrapolated, but if the flow is supersonic, all other variables are extrapolated. The solid wall condition are represented by non-slip boundary condition u = 0 for viscous flow and free-slip boundary condition U. **n** = 0 for inviscid flow, where in the last case the velocity components are calculated through of the contravariant velocity components. However, in both cases all other variables are extrapolated.

2.4.3.3 Block Interface

A multi-block structured mesh with two blocks is used to clarify the block interface boundary condition treatment. The first block, which is in red, is where the fluid enters. The second block, which

---

[59] Anderson, D.; Tannehill, J.; Pletcher, R. *Computational fluid mechanics and heat transfer*, Hemisphere Publishing Corporation - McGraw Hill, 1984.

[60] Mac Cormack, R. *The effect of viscosity in hypervelocity impact cratering*. AIAA., 1969. p. 69–354, 1969.

[61] ROE, P. *Approximate Riemann solvers, parameter vectors, and difference schemes*. Journal of Computational Physics, 1981. v. 43, p. 357–372, 1981.

is in blue, is where the fluid from block1 becomes its inlet. The mesh of each block has 60 x 50 nodes. For the treatment of the boundary between blocks1 and 2, it is observed that they are physically coincident. For the exchange of information between the blocks, the multi-block structured mesh is coincident at the boundary. At the boundary between the blocks, the boundary of block1 is considered as an outflow boundary condition, since the fluid leaves this block. In block2, the boundary is the inlet of the fluid exiting block1, being considered an inlet boundary condition. In order to avoid a numerical discontinuity of the solution at the boundary between the blocks, after each iteration a numerical boundary condition is applied at this border, where the values of the variables are determined through the variables neighboring of such boundary. For this numerical condition, please consult the work by [Almeida et al.][62]. With the input and initial conditions and the boundary conditions mentioned above, it is possible to obtain satisfactory numerical results by implementing the Mac Cormack original method and the use of multi-block structured mesh in the cases discussed as follow.

2.4.3.4 Stability Consideration

The monitoring of the stability of the numerical scheme is performed by observing the residue at all internal points of the discretized domain, according to the equation below:

$$(\Delta t) = \frac{CFL}{\frac{2v}{Re} + CFx + CFy} \quad \text{where} \quad CFx = \left|\xi_x u + \xi_y v\right| + \sqrt{\frac{\gamma P}{\rho}}\sqrt{\xi_x^2 + \xi_y^2}$$

$$\text{and} \qquad CFy = \left|\eta_x u + \eta_y v\right| + \sqrt{\frac{\gamma P}{\rho}}\sqrt{\eta_x^2 + \eta_y^2}$$

Eq. 2.4.12

Among the conserved variables of ˆQ, the one used in the above equation for determining the residue is the density ρ. To keep the numerical scheme stable, the value of the time integration step can be defined by the following empirical formula in generalized coordinates [Anderson et al.][63], [Mac Cormack][64], where CFL is the Courant-Friedrichs-Lewy number. For the explicit Mac Cormack scheme the CFL must be less than or equal to 0.5. The values of CFx and CFy for the above equation can be determined through **Error! Reference source not found.**.

2.4.3.4.1 Test 1 - Flow Over a Backward-Facing Step

The first test is the well-known subsonic flow over a backward facing step that is often used as benchmark problem in computational fluid dynamics. The main feature of this flow is that it has a simple geometry that generates an interesting and complex flow field, such as flow separation, reattachment zone and recirculation bubbles on the upper and lower wall of the channel. These characteristics are dependent on the Reynolds number and the geometrical parameters[65]; [Biswas et

---

[62] Jeferson Osmar de Almeida, Diomar Cesar Lobão , Cleyton Senior Stampa, Gustavo Benitez Alvarez, "*Multi-block Technique Applied to Navier-Stokes Equations in Two Dimensions*", Original Article, DOI: 10.5433/1679-0375.2018v39n2p115.

[63] Anderson, D.; Tannehill, J.; Pletcher, R. *Computational fluid mechanics and heat transfer*, Hemisphere Publishing Corporation - McGraw Hill, 1984.

[64] Mac Cormack, R. *The effect of viscosity in hypervelocity impact cratering*. AIAA., 1969. p. 69–354, 1969.

[65] Armaly, B.; Durst, F.; Pereira, J.; Schonung, B. *Experimental and theoretical investigation of backward facing step flow*. J. Fluid Mech, 1983. v. 127, p. 473–496, 1983.

al.][66], [Breuer and Durst][67], [Saleel et al.][68]. For the numerical simulations, the following initial conditions are assumed: density r_0 = 1.21kg/m^3, pressure p_0 = 1.01x10^{5}N/m^2, coefficient of dynamic viscosity μ = 1.81x10^5 kg/(m.s), and ratio of the specific heats γ = 1.4. The block1 (upstream block) has a mesh size of 25x25 and the block2 (downstream block) has size of 160 x 50. The **Figure 2.4.1 (a-b)** shows the velocity contours of the steady state flow field for two different Reynolds number (Re = 50 and 100) for expansion ratio H/h=1.9423 (≃2). In all cases shown in figures, a vortex is found in the concave corner behind the step and the maximum velocity is located on the upstream side of the channel, as found in the experiments performed by, [Biswas et al.][69], [Saleel et al.[70]]. It can also be seen that the size of the recirculation region increases with increasing Reynolds number. **Table 2.4.1** describes the variation of reattachment length (x_r) for two different Reynolds numbers (Re = 50 and 100), which clearly shows that the reattachment length increase with increasing Reynolds number, as found in the references. In the second column are presented experimental values

Experimental x_r	Numerical x_r			
Re	REF-1	Authors	REF-2	REF-3
50	1.70	1.61	1.55	1.55
100	3.06	2.82	2.80	2.81

Table 2.4.1 Comparation of the Results For Flow Over a Backward Facing Step

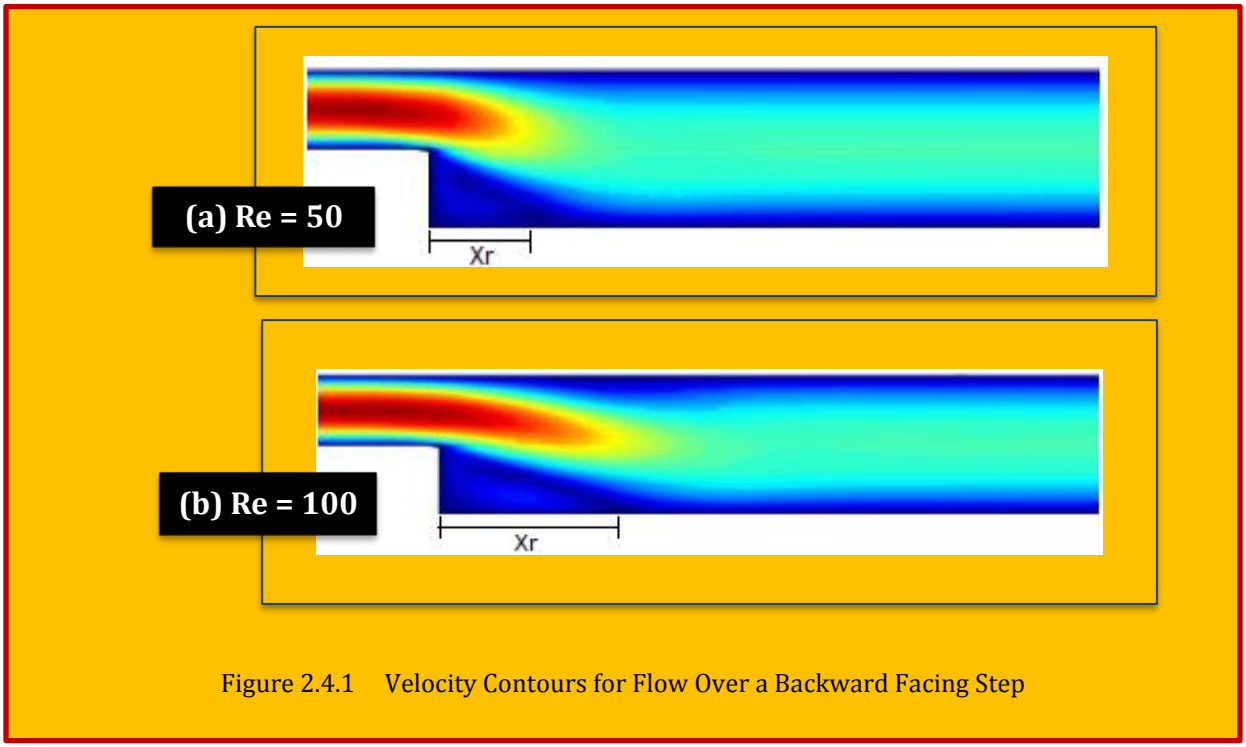


Figure 2.4.1 Velocity Contours for Flow Over a Backward Facing Step

[66] Biswas, G.; Breuer, M.; Durst, F. *Backward-facing step flows for various expansion ratios at low and moderate Reynolds numbers.* Journal of Fluids Engineering, 2004. v. 126, p. 362–374, 2004. DOI: 10.1115/1.1760532.
[67] Baird, S.; Mcguirk, J. J. *Multi-block parallel simulation of fluid flow in a fuel cell. In: High-Performance Computing and Networking. HPCN-Europe 1999*. Springer Lecture Notes in Computer Science, 1999. v. 1583.
[68] Saleel, C.; Shaija, S.; Jayaraj, S. *On simulation of backward facing step flow using immersed boundary method.* American Journal of Fluid Dynamics, 2013.
[69] See 283.
[70] Saleel, C.; Shaija, S.; Jayaraj, S. On simulation of backward facing step flow using immersed boundary method. American Journal of Fluid Dynamics, 2013.

obtained in [Armaly et al.][71], and denoted by REF-1. The third column presents the results obtained in this work, and denoted by Authors. The fourth and fifth columns correspond to values obtained by numerical simulations in [Biswas et al.][72] and [Saleel et al.][73], and denoted by REF-2 and REF-3 respectively. The results obtained by the present work are quite compatible with results found in the references. As can be clearly seen in **Figure 2.4.1 (a-b)** and **Table 2.4.1** the results are in agreement with the numerical data especially with respect to the reattachment length.

2.4.3.4.2 Test 2 - Flow Over a Curved Ramp

The second test case is a transient supersonic flow over a curved ramp at Mach number 1.5 that can be used as benchmark problem for the supersonic Euler equations. This flow is characterized by the formation of a detached bow shock in front of the curved ramp and of an expansion wave. Moreover, other feature is the wave shock reflection from the solid wall [Lobao][74-75], [Allen][76]. For the numerical simulation, the initial conditions are as follows: density $r_0 = 1.21 kg/m^3$, pressure $p_0 = 1.01 \times 10^5 N/m^2$, and ratio of the specific heats $\gamma = 1.4$. The mesh for the multi-block solutions of this problem. The mesh sizes in the first and second blocks are 110 x 80 and 60 x 80, respectively. The pressure and Mach number contour over a curved ramp are shown in **Figure 2.4.2 (a-b)** respectively, for Mach number 1.5. The detached bow shock in front of the curved ramp can be seen in the figures as well as the shock reflection from the solid wall. The **Figure 2.4.2 (b)** shows the expansion wave, where is possible to observe the subsonic, transonic and supersonic zones before of the curved ramp. As can be observed in **Figure 2.4.2 (a-b)** there is no spurious oscillation at the interface boundary between the two blocks even in presence of a very strong shock wave structure. The numerical interface

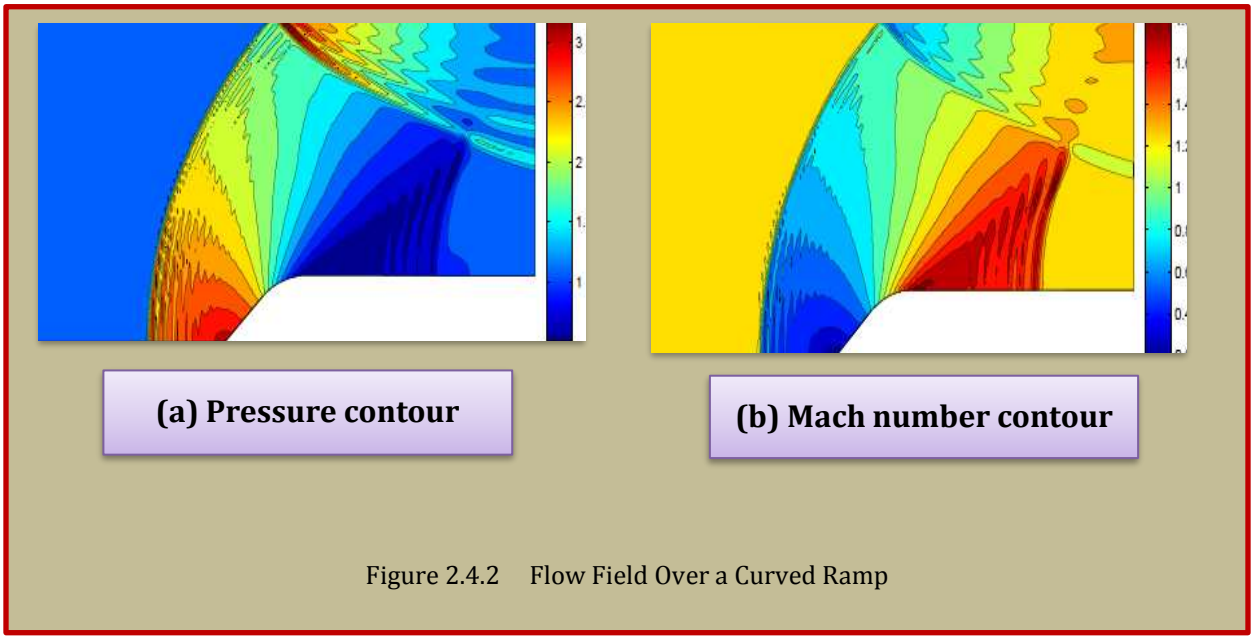


Figure 2.4.2 Flow Field Over a Curved Ramp

[71] Armaly, B.; Durst, F.; Pereira, J.; Schonung, B. *Experimental and theoretical investigation of backward facing step flow*. J. Fluid Mech, 1983. v. 127, p. 473–496, 1983.
[72] Biswas, G.; Breuer, M.; Durst, F. *Backward-facing step flows for various expansion ratios at low and moderate Reynolds numbers.* Journal of Fluids Engineering, 2004. v. 126, p. 362–374, 2004
[73] Saleel, C.; Shaija, S.; Jayaraj, S. *On simulation of backward facing step flow using immersed boundary method.* American Journal of Fluid Dynamics, 2013.
[74] Lobao, D. *High Resolution Schemes Applied to the Euler Equations*. (PhD thesis), University of Bristol, 1992.
[75] Lobao, D. *Numerical simulations of Navier-stokes for transient flows in 2d. In: Mechanical Computational.* Buenos Aires, Argentina: [s.n.], 2010.
[76] Allen, C. *An Efficient Euler Solver for Predominantly Supersonic Flows with Embedded Subsonic Pockets*. (PhD thesis, University of Bristol, 1992.

boundary condition is well suited for this simulation which demonstrates its applicability. The present numerical results are compared with the numerical data presented in [Lobao][77].

2.4.3.5 Conclusions

In this work, the discretization of the Navier-Stokes and Euler equations on multi-block structured mesh is realized using explicit Mac Cormack method. The numerical results for steady state and unsteady state compressible fluid flow in two-dimensional geometries are presented for two different test cases in subsonic and supersonic regimes, the flow over a backward facing step and flow over a curved ramp, respectively, which indicated good agreement with the references data. Therefore, it is shown the capability of the methodology to obtain numerical results for these types of subsonic and supersonic flows in two-dimensional complex geometries.

2.4.4 Other Explicit Schemes

In addition to Mac Cormack scheme, other explicit methods can be used to solve the Compressible N-S equation, including:

- Hopscotch Method
- Leapfrog/DuFort-Frankel Method
- Brailovskaya Method
- Allen-Cheng Method
- Lax-Wendroff Method

These as discussed earlier might be used for Heat equation and viscous Burger's equation, but difficult to apply to more complicated equation like compressible N-S. For one thing, all the above, except the Lax-Wendroff, are 1st order accurate in time. So they cannot be used for accurately compute the time evolution of a flow field. In addition, all have stability restriction which limits the maximum time steps.

2.4.5 Implicit Schemes

2.4.5.1 Beam-Warming

In the Beam-Warming scheme, the solution is marched in time using:

$$\Delta^n\mathbf{U} = \left(\frac{\theta_1\Delta t}{1+\theta_2}\right)\frac{\partial}{\partial t}(\Delta^n\mathbf{U}) + \left(\frac{\Delta t}{1+\theta_2}\right)\frac{\partial}{\partial t}(\mathbf{U}^n) + \left(\frac{\theta_2}{1+\theta_2}\right)\Delta^{n-1}\mathbf{U} + \mathcal{O}\left[\left(\theta_1 - \frac{1}{2} - \theta_2\right)(\Delta t)^2 + (\Delta t)^3\right]$$

$$\Delta^n\mathbf{U} = \mathbf{U}^{n+1} - \mathbf{U}^n$$

Eq. 2.4.13

This is a general difference formula (revisited) with appropriate choice of parameters Θ_1 and Θ_2, represents many of the standard difference scheme. For compressible N-S equations, either the Euler implicit scheme ($\Theta_1 = 1$, $\Theta_2 = 0$), which is first order accurate in time, or a three point backward implicit scheme ($\Theta_1 = 1$, $\Theta_2 = 1/2$), which is 2nd order accurate in time, is normally used. The difference formula so called delta form is linearized using truncated Taylor series expansion. For example

[77] Lobao, D. *High Resolution Schemes Applied to the Euler Equations. (PhD thesis)*", University of Bristol, 1992.

$$\mathbf{E}^{n+1} = \mathbf{E}^{n} + \left(\frac{\partial \mathbf{E}}{\partial \mathbf{U}}\right)^{n} (\mathbf{U}^{n+1} - \mathbf{U}^{n}) + O[(\Delta t)^2]$$

$$\Delta^{n}\mathbf{E} = [\mathbf{A}]\Delta^{n}\mathbf{U} + O[(\Delta t)^2] \qquad \text{where } \mathbf{E} = \mathbf{E}_i - \mathbf{E}_v$$

Similary

$$\Delta^{n}\mathbf{F} = [\mathbf{B}]\Delta^{n}\mathbf{U} + O[(\Delta t)^2]$$

$$\Delta^{n}\mathbf{G} = [\mathbf{C}]\Delta^{n}\mathbf{U} + O[(\Delta t)^2]$$

Eq. 2.4.14

Where [**A**], [**B**], and [**C**] are the Jacobian matrix which their definition can be obtained from different sources, such as [Chung][78]. The details for a 2D compressible N-S equation is provided in[79].

2.4.5.2 Mac Cormack

Mac Cormack (1981) also developed an implicit algorithm analog to his explicit method. This new method consists of two stages. In the first stage, uses the original Mac Cormack scheme while the second stage employs an implicit algorithm which eliminates any stability restriction. The resulting matrix equations are either lower or upper block equations with details in[80]. The implicit Mac Cormack method is defined by:

$$\text{Predictor}: \quad \left(1 + \frac{\lambda \Delta t}{\Delta x}\right)\Delta \overline{u_i^{n+1}} = (\Delta u_i^{n})_{explicit} + \frac{\lambda \Delta t}{\Delta x}\Delta \overline{u_{i+1}^{n+1}}$$

$$\Delta \overline{u_i^{n+1}} = \overline{u_i^{n+1}} - u_i^{n} \qquad (\Delta u_i^{n})_{explicit} = \left(\overline{u_i^{n+1}}\right)_{explicit} - u_i^{n}$$

$$\text{Corrector}: \quad \left(1 + \frac{\lambda \Delta t}{\Delta x}\right)\Delta u_i^{n+1} = (\Delta \overline{u_i^{n+1}})_{explicit} + \frac{\lambda \Delta t}{\Delta x}\Delta u_{i-1}^{n+1}$$

$$u_i^{n+1} = \frac{1}{2}\left(u_i^{n} + \overline{u_i^{n+1}} + \Delta u_i^{n+1}\right) \qquad \left(u_i^{n+1}\right)_{explicit} = \frac{1}{2}\left[u_i^{n} + \left(\overline{u_i^{n+1}}\right)_{explicit} + \left(\Delta \overline{u_i^{n+1}}\right)_{explicit}\right]$$

$$\lambda \geq \left[\left(c + \frac{2\mu}{\Delta x} - \frac{\Delta x}{\Delta t}\right) \;,\; 0.0\right] \qquad c\,, \mu > 0$$

Eq. 2.4.15

This method is unconditionally stable and 2nd order accurate in both space and time, provided that $\mu\Delta t/(\Delta x)^2$ is bounded as Δt and Δx approach zero. For detail explanation of method and accuracy, as well as a 2nd order N-S example, reader could refer to[81].

---

[78] T. J. Chung, "*Computational Fluid Dynamics*", University of Alabama in Huntsville, Cambridge University Press 2002.

[79] Anderson, Dale A; Tannehill, John C; Plecher Richard H; 1984:"*Computational Fluid Mechanics and Heat Transfer"*, Hemisphere Publishing Corporation.

[80] Anderson, Dale A; Tannehill, John C; Plecher Richard H; 1984:"*Computational Fluid Mechanics and Heat Transfer"*, Hemisphere Publishing Corporation.

[81] Anderson, Dale A; Tannehill, John C; Plecher Richard H; 1984:"*Computational Fluid Mechanics and Heat Transfer"*, Hemisphere Publishing Corporation.

3 Solver Schemes II

3.1 Pressure Based (Incompressible)

Finding a suitable solution for incompressible NS equations are the central issue in the subject of CFD, dating back to at least the 60's, when the ***MAC scheme*** [Harlow and Welch][82] and the ***Projection method*** [Chorin][83] were invented. MAC scheme and Staggered Grid enjoyed immediate success and similar ideas were also applied to electromagnetics. On the other hand, it wasn't until the late 80's and early 90's, did we begin to understand the mysteries surrounding the projection method and realize its potential. By now projection method is by far the most popular method in this field. The main difficulty in numerically solving incompressible NS equations is the lack of proper evolutionary equation for pressure. After all for incompressible flows, pressure does not carry its usual thermodynamic meaning. It is there mainly as a Lagrange multiplier for the constraint of incompressibility. This translates to a lack of proper boundary condition, when equations for pressure are derived. The prime Finite Volume solution methods, are ***pressure*** or ***density*** based methods[84]. ***The pressure based schemes are mainly developed for incompressible, low Reynolds number application when pressure value is guessed to begin and updated using Poisson's equation after solving the momentum equations***. The main advantage is the decoupling of the momentum and energy equation when the flow variables could be determined in segregated fashion. Among notable methods are ***SIMPLE*** (Semi-Implicit Methods for Pressure-Linked Equations) and ***PISO***. The density based methods are more rigorous due to fact that governing equations are solved in ***coupled*** environment where pressure is obtained through equation of state. In both cases, additional scalar equations are solved in a ***segregated*** style. Most commercial CFD vendors try to provide both methods[85].

With segregated methods an equation for a certain variable is solved for all cells, and then the equation for the next variable is solved for all cells, etc. With coupled methods, for a given cell equations for all variables are solved, and that process is then repeated for all cells[86]. The segregated solution method is the default method in most commercial finite volume codes. It is best suited for incompressible flows or compressible flows at low Mach number. Compressible flows at high Mach number, especially

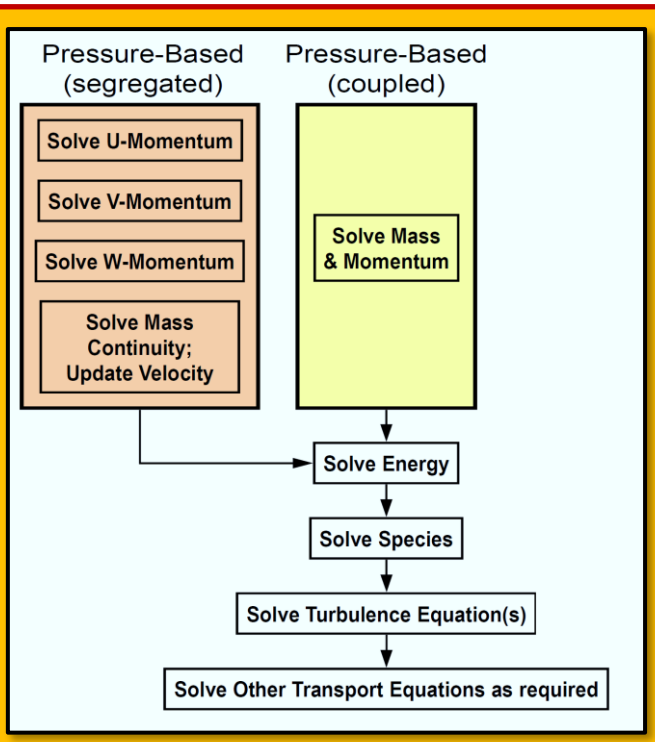


Figure 3.1.1 Accessible Pressure Solvers in Fluent

[82] F.H. Harlow and J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluids, 1965,

[83] A. J. Chorin, "*On the convergence of discrete approximations of the Navier-Stokes equations",* Math. Comp. 1969.

[84] Weinan E, "*Numerical Methods for Viscous Incompressible Flows: Some Recent Advances*", Department of Mathematics and Program in Applied and Computational Mathematics, Princeton University, Princeton, NJ.

[85] Introduction to ANSYS Fluent, 2010.

[86] Georgia Tech Computational Fluid Dynamics, "*Solution Methods for Navier Stokes Equations*", Spring 2007.

when they involve shock waves, are best solved with the coupled solver. Here we are focused here in incompressible (M < 0.3). Equation system to be solved is the continuity equation together with the momentum equations, while the continuity equation is considered as a side condition, demanding a divergence-free flow field. Pressure does not appear in the continuity equation of continuity and equations of momentum and continuity need to be coupled (**Figure 3.1.1**).

3.1.1 Methods Based on the Vorticity Equation

By taking the curl of viscous incompressible flow, the pressure term vanishes. As for drawbacks it is limited to 2D applications, but revised 3D approaches are available.

3.1.2 Methods Based on Artificial (Pseudo) Compressibility

The pseudo-compressibility method is a preconditioning technique of overcoming the numerical issues related to the uncoupling of pressure and velocity field in the incompressible equations [Chorin][87]-[88]. The governing equations are made artificially hyperbolic by adding a density time-derivative to the continuity equation[89]. Based on physical analogies, the time derivative of density is then linked to the pressure by introducing an artificial compressibility factor. A number of possible approaches are listed below:

$$\nabla . u^{\varepsilon} + \varepsilon p^{\varepsilon} = 0$$
$$\nabla . u^{\varepsilon} + \varepsilon p_t^{\varepsilon} = 0$$
$$\nabla . u^{\varepsilon} - \varepsilon \Delta p^{\varepsilon} = 0$$
$$\nabla . u^{\varepsilon} - \varepsilon \Delta p_t^{\varepsilon} = 0$$

Eq. 3.1.1

The first three versions are well known while the last one was recently introduced in [90]. We will refer them as a class of pseudo-compressibility methods for the unsteady incompressible Navier-Stokes equations. The aim of this paper is to present some recent analyses on this class of pseudo-compressibility methods as the perturbation parameter $\varepsilon \to 0$. An important aspect of this work is the error analysis of time discretization schemes (they can be employed, in principle, with any consistent space discretization) associated with the pseudo-compressibility methods[91]. As an example investigated by [Nguyen & Park][92], the topic that is still receiving increasing attention, particularly for applications in many hydraulic and hydrodynamic problems.

3.1.3 Methods Based on Pressure Iterations or Pressure Correction

These are refer to number of methods as:

1. ***Projection Methods*** [Chorin & Temam, 1968]
   The projection method is an effective means of numerically solving time-dependent incompressible fluid-flow problems. It was originally introduced by Alexandre Chorin in 1967 as an efficient means of solving the incompressible Navier-Stokes equations. The key

---

[87] Alexandre Joel Chorin, "*Numerical Solution of the Navier-Stokes Equations*", AEC Computing and Applied Mathematics Center, Courant Institute of Mathematical Sciences.

[88] Alexandre Joel Chorin, "*The Numerical Solution of the Navier-Stokes Equations for an Incompressible Fluid*", supported by the U. S. Atomic Energy Commission, Contract No. AT(30-1)-1480.

[89] Michele Ferlauto, "*A Pseudo-Compressibility Method For Solving Inverse Problems Based On The 3D Incompressible Euler Equations*", Taylor & Francis in *Inverse Problems in Science and Engineering* ,2014.

[90] "*A new pseudo-compressibility method for the Navier-Stokes equations*", SIAM J. Math. Anal, 1994.

[91] Jie Shen, "*Pseudo-Compressibility Methods for the Unsteady Incompressible Navier-Stokes Equations*", Department of Mathematics, Penn State University, USA.

[92] Van-Tu Nguyen and Warn-Gyu Park, "*A free surface flow solver for complex three-dimensional water impact problems based on the VOF method*", International Journal For Numerical Methods In Fluids Int. J. Numerical Meth. Fluids -2015.

advantage of the projection method is that the computations of the velocity and the pressure fields are decoupled[93].

- ***Fractional Step Method*** [Kim & Moin, 1975]
  The method is based on a fractional step, or time-splitting, scheme in conjunction with the approximate factorization technique.

2. ***Marker and Cell Method – MAC*** [Harlow & Welch, 1965]
   technique is described for the numerical investigation of the time-dependent flow of an incompressible fluid, with limited capability. The primary dependent variables are the pressure and the velocity components. Also used is a set of marker particles which move with the fluid. These called the Marker And Cell (MAC) method.

3. ***SIMPLE, SIMPLER, SIMPLEST, PISO***, [Patankar and Spalding, 1972, 1981]. See below.

4. ***Fast Fluid Dynamic*** [Stam, 1999]. See below.

While 1 & 2 are scarcely used in CFD, the third item (3), known as pressure corrections methods, are used extensively. The basic idea is to make use of a Poisson equation for the pressure; as pressure appears only in the momentum equations in form of a partial derivative of first order. This can be achieved by computing the derivative $\partial P/\partial x^i$ such that the result yields a divergence free flow field. An adequate iterative method is formulated that iterates the pressure until the conservation of mass and momentum is obtained. These methods are also known as projection methods. This is celebrated Poisson Equation. The RHS of the equation is a function of the partial derivatives of the velocity components and is obtained from the convective part of the momentum equations. All other terms i.e. the time derivative and the local change of the molecular transport (i.e. the diffusive type terms) are removed because of the continuity side condition[94]. The basic procedure can be visualized as Computation of a velocity field from the solution of the momentum equations with an initial (guessed) or no pressure field.

1. Computation of the pressure from the Poisson equation with the previously computed velocity field.
2. Correction of the velocity field with the "new" pressure.

$$\frac{\partial u_i}{\partial x_i} = 0 \qquad , \qquad \frac{\partial u_j}{\partial t} + \frac{\partial (u_i u_j)}{\partial x_i} = -\frac{1}{\rho}\frac{\partial p}{\partial x_j} + \frac{\mu}{\rho}\left(\frac{\partial u_j}{\partial x_i}\right)^2$$

$$\text{Take Div. of Momentum} \rightarrow \frac{\partial}{\partial x_j}\left[\frac{\partial u_j}{\partial t} + \frac{\partial (u_i u_j)}{\partial x_i}\right] = \frac{\partial}{\partial x_j}\left[-\frac{1}{\rho}\frac{\partial p}{\partial x_j} + \frac{\mu}{\rho}\left(\frac{\partial u_j}{\partial x_i}\right)^2\right]$$

$$\underbrace{\frac{\partial}{\partial t}\left(\frac{\partial u_j}{\partial x_j}\right)}_{0} + \frac{\partial}{\partial x_j}\left[\frac{\partial (u_i u_j)}{\partial x_i}\right] = -\frac{1}{\rho}\frac{\partial^2 p}{\partial {x_j}^2} + \frac{\mu}{\rho}\frac{\partial}{\partial x_i}\left[\frac{\partial u_j}{\partial x_i}\right]\underbrace{\left[\frac{\partial u_j}{\partial x_j}\right]}_{0}$$

$$\underbrace{\frac{\partial^2 p}{\partial {x_j}^2}}_{\nabla^2 p} = -\underbrace{\rho \frac{\partial}{\partial x_j}\left[\frac{\partial (u_i u_j)}{\partial x_i}\right]}_{f(u_i)}$$

Eq. 3.1.2

---

[93] Wikipedia.
[94] Georgia Tech Computational Fluid Dynamics, "Solution Methods for Navier Stokes Equations", 2007.

Additional detailed information regarding this can be obtained from original classic paper by [Patankar & Spalding][95], as well as excellent discussion regarding pressure-velocity coupling (SIMPLE, SIMPLER, SIMPLEC, PISO) algorithms by [Versteeg and Malalasekera][96]. There is also an investigation by [Rhie, & Chow][97] to demonstrate the pressure scheme of the 2D incompressible, steady N-S equations for flow past an airfoil. This method is applied to the turbulent flows over airfoils with and without trailing edge separation, with the aid of k-ε modeling of the turbulent flow. Instead of the staggered grid, an ordinary grid system is employed and a specific scheme is developed to suppress the pressure oscillations.

3.1.3.1 Case Study - Numerical Study of Compressible Lid Driven Cavity Flow

Lid-driven cavity flow of Newtonian fluids is one of the most well-known problems in CFD literature due to its peculiar challenges in the form of singularities in spite of its simple geometry. In addition, the availability of both analytical solutions and experimental results for the lid-driven cavity flow field has enabled researchers to test and improve their computational methods through this benchmark geometry. Recently, [Hussain][98] is investigated a two-dimensional (2D), mathematical model is adopted to investigate the development of circulation patterns for ***compressible***, laminar, and shear driven flow inside a rectangular cavity. And the results of a commercial code (*Fluent*) is compared with in-house code, as well as the vorticity-stream function formulation of the 2D incompressible N-S equations [Ghia et al.][99], using a ***pressure based*** coupled algorithm. The simulations are carried out for the unsteady, lid driven cavity flow problem with moving boundary (bottom) for different Reynolds number (Re = 100, 400, 1000), Mach numbers (M = 0.5), bottom velocities and high initial pressure and temperature. Similar studies has been conducted by [Hosseinzadeh, et al.][100]. A snap shot of the results provided in **Figure 3.1.2** and **Figure 3.1.3.** Another study done by [Giannetti][101] present the results of a linear stability analysis applied to an incompressible flow in a 3D lid-driven cavity.

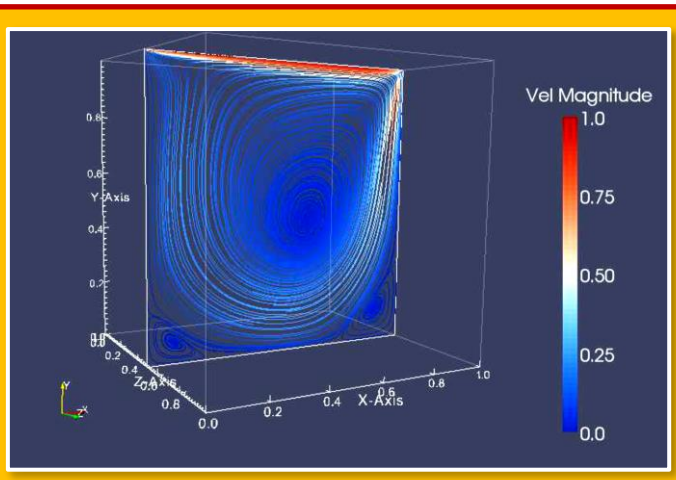


Figure 3.1.2 Streamlines of the Base Flow in the Symmetry Plane z = 0.5 and Re = 1000

---

[95] S. V. Patankar and D. B. Spalding,"*Calculation Procedure For Heat, Mass And Momentum Transfer In Three-Dimensional Parabolic Flows*", International Journal Of Hear Mass Transfer, Vol. Is, Pp. 1787-1806, 1972.

[96] H K Versteeg and W Malalasekera, "An Introduction To Computational Fluid Dynamics - The Finite Volume Method", Second Edition, Pearson Education Limited 1995, 2007

[97] C. M. Rhie, W. L. Chow, "*Numerical Study of the Turbulent Flow Past An Airfoil With Trailing Edge Separation*", AIAA Journal, Vol. 21, No. 11, November 1983.

[98] Amer Hussain, "*A Numerical Study of Compressible Lid Driven Cavity Flow with a Moving Boundary*", Thesis Submitted to the Graduate Faculty of the University of New Orleans in partial fulfillment of the requirements for the degree of Master of Science in Engineering Mechanical, 2016.

[99] Ghia, U., Ghia, K.N., and Shin, C.T., 1982, "*High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method,*" Journal of Computational Physics, 48, pp. 387-411.

[100] S. Hosseinzadeh, R. Ostadhossein, H.R. Mirshahvalad and J. Seraj, "Using Simpler Algorithm for Cavity Flow Problem", Mechatronics and Applications: An International Journal (MECHATROJ), Vol. 1, No.1, January 2017.

[101] Flavio Giannetti, Paolo Luchini, Luca Marino, "*Linear stability analysis of three-dimensional lid-driven cavity flow*", Department of Mechanical Engineering, University of Salerno, Italy.

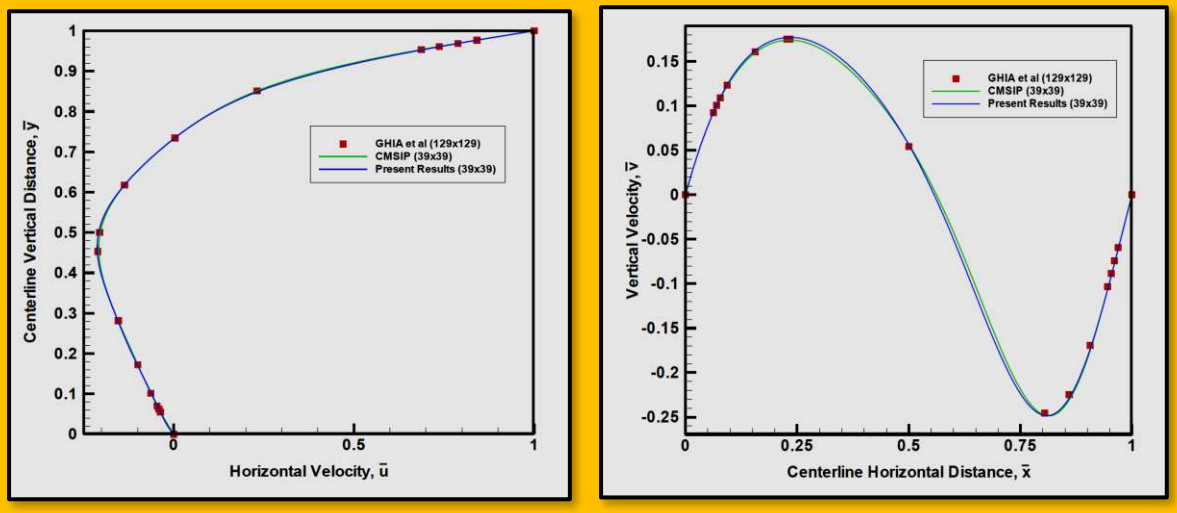


Figure 3.1.3 Distribution of Velocities (u, v) along Centerline Horizontal Distance for Re = 100

The principal goal is the evaluation of the critical value of the Reynolds number and the main characteristics of the flow instability.

3.1.4 PISO Algorithm

PISO algorithm (Pressure-Implicit with Splitting of Operators) was proposed by [Issa] in 1986 without iterations and with large time steps and a lesser computing effort[102]. It is an extension of the ***SIMPLE*** algorithm. ***PISO*** is a pressure-velocity calculation procedure for the Navier-Stokes equations developed originally for non-iterative computation of unsteady compressible flow, but it has been adapted successfully to steady-state problems. PISO involves one predictor step and two corrector steps and is designed to satisfy mass conservation using predictor-corrector steps. The PISO algorithm with neighbor correction is highly recommended for all transient flow calculations, especially when you want to use a large time step. (For problems that use the LES turbulence model, which usually requires small time steps, using PISO may result in increased computational expense, so SIMPLE or SIMPLEC should be considered instead). PISO can maintain a stable calculation with a larger time step and an under-relaxation factor of 1.0 for both momentum and pressure. For steady-state problems, PISO with neighbor correction does not provide any noticeable advantage over SIMPLE or SIMPLEC with optimal under-relaxation factors.

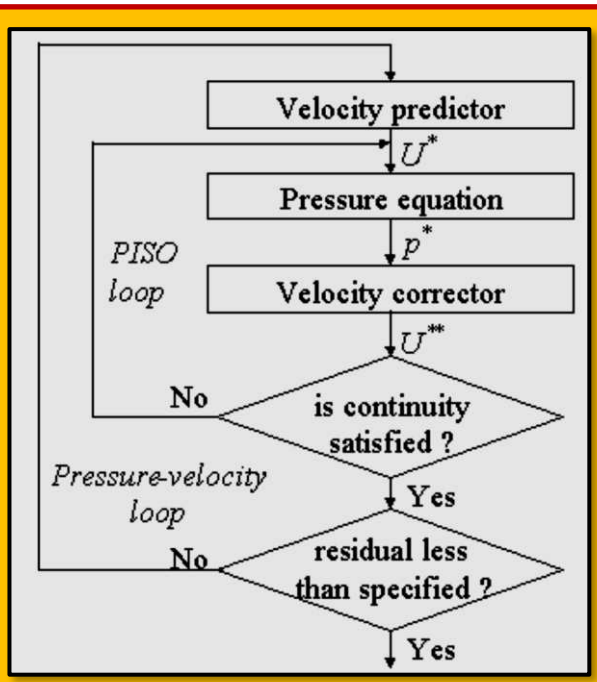


Figure 3.1.4 PISO algorithm flow chart (Courtesy of Giannopapa, and G. Papadakis)

PISO with skewness correction is recommended for both steady-state and transient calculations on meshes with a high degree of distortion. When you use PISO neighbor correction, under-relaxation

[102] Wikipedia.

factors of 1.0 or near 1.0 are recommended for all equations. If you use just the PISO skewness correction for highly-distorted meshes (without neighbor correction), set the under relaxation factors for momentum and pressure so that they sum to 1 (e.g., 0.3 for pressure and 0.7 for momentum)[103]. A PISO algorithm flow chart can be appreciated in **Figure 3.1.4**. The method is not restricted to using the PISO algorithm, e.g. the Simple algorithm can be used as well[104].

3.1.4.1 Case Study - Second Order Non-Iterative PISO-Algorithm vs Iterative PISO

A new variant of the non-iterative PISO-algorithm for the solution of implicitly discretized fluid flow equations is proposed and investigated by [Tukovic et al.][105]. The governing equations are discretized in space using a second-order accurate finite volume method. A second-order accurate three-level implicit temporal discretization scheme is used to discretize the momentum equation, where the non-linear convection term is linearized using a second- order explicit approximation of mass flux at the new time level, based on the results from previous two time steps. In order to ensure temporal consistency on collocated meshes, a consistent Rhie-Chow interpolation for the computation of mass fluxes is used, which is here extended to moving meshes. The proposed non-iterative PISO-algorithm is tested on three fixed and moving mesh test cases, demonstrating its second order accuracy in time.

3.1.4.1.1 Introduction

The non-iterative nature of the algorithm implies that momentum equation is discretized and solved only once in the predictor step at the beginning of time step calculation. Predictor step is followed by two or more corrector steps, where the conservation of mass and momentum is enforced by the solution of pressure equation and explicit momentum correction. In the original PISO-algorithm, temporal discretization is per- formed using first-order accurate (two-level) Euler implicit scheme and the overall temporal accuracy is first order. It is emphasized in that the application of second-order accurate three-level implicit temporal discretization scheme does not yield second-order temporal accuracy, since in the linearized convection term in the momentum equation mass flux from the previous time step is used. The second-order accuracy in time can be achieved by using outer iteration loop enclosing the non-iterative PISO-algorithm. The mass flux in the linearized convection terms stems then from the previous iteration (in the first iteration, it stems from the previous time step). Such an iterative PISO-algorithm does not have substantial advantages when compared to other iterative algorithms. [Park][106] proposed a non-iterative PISO-algorithm of second-order temporal accuracy, where the non-linear convection term in the momentum equation is advanced in time using an explicit temporal discretization scheme of higher order. He proposed using the second-order Adams–Bashforth or the third-order Runge–Kutta scheme. The disadvantage of computing convection flux fully explicitly is the limit on time-step size due to stability constraints of the explicit scheme. In this study we propose an alternative approach to achieve second-order temporal accuracy of the non-iterative PISO-algorithm. Only the explicit part of the linearized convection flux is approximated at the new time level using a second-order extrapolation from two previous time steps. This is equivalent to using the second-order Adams–Bashforth scheme to advance the mass flux in time within the predictor stage of the PISO-algorithm. Owing to the fact that remaining part of the convection flux is treated implicitly, the stability limit is increased.

3.1.4.1.2 Preliminary Results

---

[103] Fluent User Guide.

[104] C. G. Giannopapa, and G. Papadakis, "*Indicative Results and Progress on the Development of The Unified Single Solution Method for Fluid-Structure Interaction Problems*", DOI: 10.1115/PVP2007-26420, 2007.

[105] Željko Tukovi ´c a , Milovan Peri´c , Hrvoje Jasak, "*Consistent second-order time-accurate non-iterative PISO-algorithm*", Computers and Fluids, 2018.

[106] Park TS . "*Effects of time-integration method in a large-eddy simulation using the PISO algorithm: Part I – Flow field"*. Numerical Heat Transfer Part A 1999;50:229–45 .

The proposed second-order non-iterative PISO-algorithm (here labelled as ePISO) is validated on flow around fixed and oscillating cylinder in a channel. The same test case are also computed using the second-order iterative version of the PISO-algorithm (here labelled iPISO) and the non-iterative version equivalent to ePISO, except that explicit terms are computed using values from the previous time step (here labelled nPISO). **Figure 3.1.5** displays inlet part of the spatial domain, which is discretized with 7126 quadrilateral finite volumes. The Reynolds number based on the average inlet velocity and cylinder diameter is Re = 100. At the channel inlet, parabolic velocity profile is specified with the average velocity **u** = 1m/s. Calculation is first performed using the iterative PISO-algorithm and the time step size Δt = 0 . 02s (corresponding to the maximal Courant number around 1.2) until a periodic response of the lift and drag force is obtained (t = 50s). Afterwards, calculation is continued from the time instance *t* = 50 s using the three PISO-algorithms named in the previous test case: iPISO, nPISO and ePISO. In such a way all algorithms start with the same initial state; the time step is also kept the same.

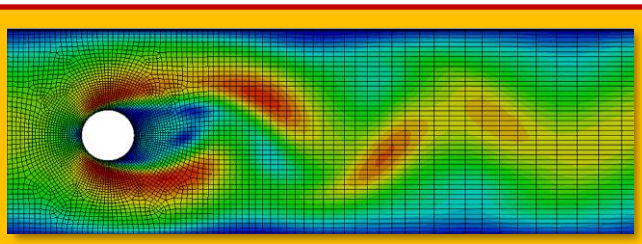


Figure 3.1.5 Flow Around Circular Cylinder in a

Figure 3.1.6 shows lift force coefficient as a function of time, calculated using different variants of the PISO-algorithm. The solution obtained using iterative PISO-algorithm (iPISO) can be considered as the most accurate for a given time step size. One can notice significant departure of the solution obtained using nPISO, while the present non-iterative PISO-algorithm (ePISO) gives solution with a negligible difference compared to the solution obtained using the iterative PISO-algorithm. It should be also noted that numerical results obtained in this study (frequency and amplitude of lift coefficient) agree well.

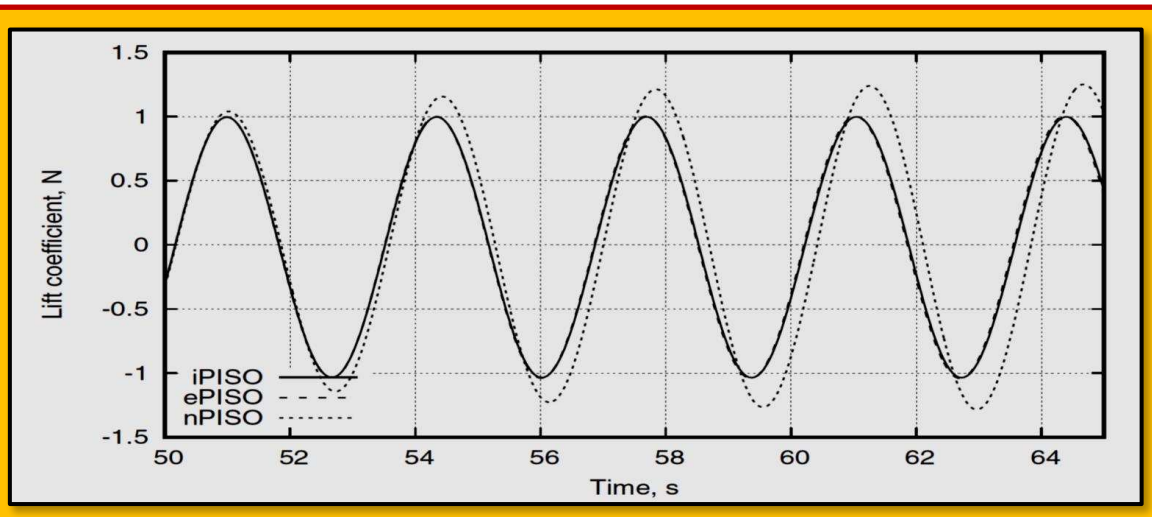


Figure 3.1.6 Lift Force as a Function of Time, using Different PISO-Algorithms for the Flow Around Fixed Cylinder in a Channel (Courtesy of Tukovic et al.)

3.1.5 Fast Fluid Dynamic (FFD)

As the naming specifies, Fast fluid Dynamic is a fast (50 times by some estimate) method to solve the incompressible NS equation. It was purposed originally by (Stam)[107] for computer graphics and animation. In recent years it has been also used for indoor airflow simulation. The model would not

[107] Jos Stam, "Stable Fluids".

be accurate enough for most engineering applications. Indeed, it suffers from too much "numerical dissipation", i.e., the flow tends to dampen too rapidly as compared to actual experiments. In a computer graphical application, on the other hand, this is not so bad, especially in an interactive system where the flow is "kept alive" by an animator applying external forces. It applies a time-splitting technique to separate the Navier-Stokes equations into several simple equations. Then those spitted equations are solved one by one. Thus, FFD can solve the Navier-Stoke equations without a

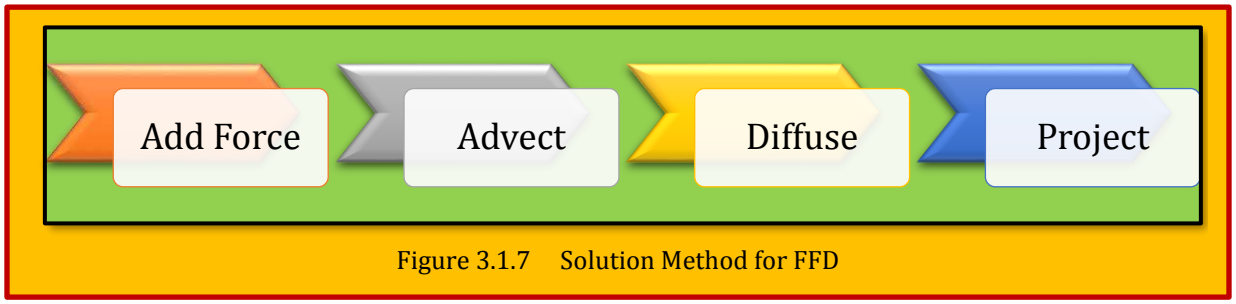


Figure 3.1.7 Solution Method for FFD

trial correction iterations used in CFD. This is the reason why FFD is much faster than CFD. However, current FFD model applied many low-order schemes, so its accuracy is poorer than CFD. It splits the momentum equation over time in 4 steps, (see **Figure 3.1.7**).

Because of a linear interpolation used in a semi-Lagrangian approach for advection equation, the FFD has a significant numerical diffusion. To reduce the numerical diffusion in the FFD model, improvements have been purposed in two fronts. One is applying a high-order interpolation scheme in the semi-Lagrangian solver [Wangda and Qingyan][108]. The other improvement was to solve the advection equation using the convectional method used in CFD, such as the Lax-Wendroff scheme and the QUICK. The first approach is the use of Fast Fluid Dynamics (FFD) that is an intermediate model between the nodal and CFD models. The FFD, developed for computer flow visualization, can efficiently solve the incompressible Navier-Stokes equations (top), energy equation (middle) and species transport equations (bottom):

$$\left.\begin{aligned} \frac{\partial u}{\partial t} &= -u\frac{\partial u}{\partial x} - \frac{1}{\rho}\frac{\partial P}{\partial x} + \nu\frac{\partial^2 u}{\partial x^2} + \frac{1}{\rho}S_F \\ \frac{\partial T}{\partial t} &= -u\frac{\partial T}{\partial x} + \alpha\frac{\partial^2 T}{\partial x^2} + S_T \\ \frac{\partial c}{\partial t} &= -u\frac{\partial c}{\partial x} + k_c\frac{\partial^2 c}{\partial x^2} + S_c \end{aligned}\right\} \quad \frac{\partial \phi}{\partial t} = -u\frac{\partial \phi}{\partial x} + k\frac{\partial^2 \phi}{\partial x^2} + S + G$$

Eq. 3.1.3

where S is the source term and G is the pressure term. The FFD method applies a time-splitting method (Ferziger and Peric 2002) to solve the governing equations. The purpose of the splitting method is to divide a complex problem (equation) into several simple ones [Ferziger and Peric 2002; John 1982; Levi and Peyroutet 2001] since solving these simple equations is mathematically easy and numerically fast[109]. Then solutions of these simple equations can be integrated into an approximated solution for the complex equation. The splitted equations in the FFD are as follows:

[108] Wangda Zuo, Qingyan Chen, "*Improvements On The Fast Fluid Dynamic Model For Indoor Airflow Simulation*", Fourth National Conference of IBPSA-USA, New York City, New York, August 11 – 13, 2010.

[109] Wangda Zuo, Qingyan Chen, "*Simulations of Air Distributions In Buildings By FFD On GPU",* HVAC&R Research.

$$\frac{\partial \varphi}{\partial t} = -\mathbf{u}\frac{\partial \varphi}{\partial x} + k\frac{\partial^2 \varphi}{\partial x^2} + S + G$$

$$\frac{\varphi^{(1)} - \varphi^{(n)}}{\Delta t} = \underbrace{S}_{\text{Source}} \quad , \quad \frac{\varphi^{(2)} - \varphi^{(1)}}{\Delta t} = \underbrace{k\frac{\partial^2 \varphi^{(2)}}{\partial x^{(2)}}}_{\text{Diffusion}} \quad ,$$

$$\frac{\varphi^{(3)} - \varphi^{(2)}}{\Delta t} = \underbrace{\mathbf{u}\frac{\partial^2 \varphi^{(2)}}{\partial x^{(2)}}}_{\text{Advection}} \quad \text{and Finally} \quad \frac{\varphi^{(n+1)} - \varphi^{(3)}}{\Delta t} = \underbrace{G}_{\text{Projection (Preesure)}}$$

Eq. 3.1.4

where superscripts (1), (2) and (3) represent temporary variables. The FFD computes sequentially the above four equations. First source is added through equation. Then the FFD calculates diffusion equation by using a first order implicit scheme. After that, advection equation is solved with a semi-Lagrangian solver. For the momentum equation, the FFD solves pressure equation together with continuity equation by using a pressure-correction projection method [Chorin][110]. It is worth to notice that there is an extra projection step before the advection step in the implemented FFD code, which is to provide a divergence-free velocity field for the semi-Lagrangian solver in the advection equation.

[110] Chorin, A.J. (1967) "*A numerical method for solving incompressible viscous flow problems*," J. of Comp. Physics.

3.2 Density Based (Compressible) Schemes

The system of governing equations for a single-component fluid, written to describe the mean how properties[111], is cast in integral Cartesian form for an arbitrary control volume V with differential surface area dA as follow

$$\frac{\partial}{\partial t}\int_V \mathbf{W}dV + \oint [\mathbf{F}-\mathbf{G}]\,.dA = \int_V \mathbf{H}dV \qquad \mathbf{W}, \mathbf{F} \text{ and } \mathbf{G} \text{ are defined as:}$$

$$\mathbf{W} = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{Bmatrix} \quad , \quad \mathbf{F} = \begin{Bmatrix} \rho v_i \\ \rho v_i u + p\hat{i} \\ \rho v_i v + p\hat{j} \\ \rho v_i w + p\hat{k} \\ \rho v_i E + pv \end{Bmatrix} \quad , \quad \mathbf{G} = \begin{Bmatrix} 0 \\ \tau_{xi} \\ \tau_{yi} \\ \tau_{zi} \\ \tau_{ij} v_j + q \end{Bmatrix} \qquad i,j = 1,3$$

Eq. 3.2.1

and the vector **H** contains source terms such as body forces and energy sources. Here ρ, v, E, and p are the density, velocity, total energy per unit mass, and pressure of the uid, respectively. **τ** is the viscous stress tensor, and q is the heat flux. Total energy **E** is related to the total enthalpy **H** by

$$\mathbf{E} = \mathbf{H} - P/\rho \quad \text{and} \quad \mathbf{H} = h + \frac{|\mathbf{V}|^2}{2}$$

Eq. 3.2.2

The implicit-time stepping method (also known as dual-time formulation) is available in the density-based explicit and implicit formulation. When performing unsteady simulations with implicit-time stepping (dual-time stepping), we use a low Mach number time-derivative unsteady preconditioner to provide accurate results both for pure convective processes (e.g., simulating unsteady turbulence) and for acoustic processes (e.g., simulating wave propagation). Here we introduce a preconditioned pseudo-time-derivative term into **Eq. 3.2.1** as

$$\frac{\partial}{\partial t}\int_V \mathbf{W}dV + \underbrace{\frac{\partial \mathbf{W}}{\partial \mathbf{Q}}}_{\Gamma}\frac{\partial}{\partial \tau}\int_V \mathbf{Q}\,dV + \oint [\mathbf{F}-\mathbf{G}].dA = \int_V \mathbf{H}\,dV$$

$$\text{where } \mathbf{Q} = [p, \mathbf{u}, T]^T \;, \quad \frac{\partial \mathbf{W}}{\partial \mathbf{Q}} = \mathbf{\Gamma} = \begin{bmatrix} \rho_P & 0 & 0 & 0 & \rho_T \\ \rho_P u & \rho & 0 & 0 & \rho_T u \\ \rho_P v & 0 & \rho & 0 & \rho_T v \\ \rho_P w & 0 & 0 & \rho & \rho_T w \\ \rho_P H - \delta & \rho u & \rho v & \rho w & \rho_T H + \rho C_P \end{bmatrix}$$

$$\text{and } \rho_P = \left.\frac{\partial \rho}{\partial P}\right|_T, \quad \rho_T = \left.\frac{\partial \rho}{\partial T}\right|_P \quad \text{with } \delta = \begin{cases} 1 & \rightarrow \text{(ideal Gas)} \\ 0 & \rightarrow \text{(incompressible)} \end{cases}$$

Eq. 3.2.3

Where t denotes physical-time and τ is a pseudo-time used in the time-marching procedure. Note that as τ→1, the second term on the left side of **Eq. 3.2.3** vanishes and Error! Reference source not

[111] "*Fluent Guide*, Using the Solver", 2006.

found. is recovered. The time-dependent term in **Eq. 3.2.3** is discretized in an implicit fashion by means of either a first or second-order accurate, backward difference in time. Here, we used a three point (2nd order) Backward Differencing scheme of [Jameson][112]. The dual-time formulation is written in semi-discrete form as follows:

$$\left[\frac{\boldsymbol{\Gamma}}{\Delta\tau}+\frac{\varepsilon_0}{\Delta t}\frac{\partial \mathbf{W}}{\partial}\right]\Delta\mathbf{Q}^{k+1}+\frac{1}{V}\oint[\mathbf{F}-\mathbf{G}]\,.dA=H-\frac{1}{\Delta t}\left(\varepsilon_0\mathbf{W}^{k}-\varepsilon_1\mathbf{W}^{n}+\varepsilon_2\mathbf{W}^{n-1}\right)$$

Eq. 3.2.4

Where ($\varepsilon_0 = \varepsilon_1 = 1/2$; $\varepsilon_2 = 0$) gives first-order time accuracy, and ($\varepsilon_0 = 3/2$; $\varepsilon_1 = 2$; $\varepsilon_2 = 1/2$) gives second-order. The k is the inner iteration counter and n represents any given physical-time level. The pseudo-time-derivative is driven to zero at each physical time level by a series of inner iterations using either the implicit or explicit time-marching algorithm. Many details of density schemes will be discussed later.

3.2.1 Case Study 1 - Density Based Solver for Turbulent Compressible Flows

The development of coupled implicit density based solver for compressible flows are considered by JF¨urst][113]. The main flow field variables are updated using lower-upper symmetric Gauss-Seidel method (LU-SGS) whereas the turbulence model variables are updated using implicit *Euler method*. The equations can be expressed in the ***integral form*** suitable for finite volume approximation as follows (assuming fixed control volume Ω)

$$\frac{d}{dt}\int_{\Omega}\mathbf{W}\,d\Omega+\oint_{\partial\Omega}(\mathbf{F}^{c}-\mathbf{F}^{v})\,dS=0$$

Eq. 3.2.5

where $\mathbf{W} = [\rho, \rho\mathbf{U}, \rho E]$ is the vector of conservative variables and $\mathbf{F}^c$ and $\mathbf{F}^v$ represent inviscid and viscous fluxes. In the case of turbulent flow the above mentioned system of equations is coupled to an additional turbulence model (e.g. a two-equation SST model[114]) which provides components of the Reynolds stress tensor and the turbulent heat flux. For detail description special and temporal discretization, please consult the [F¨urst] [115].

3.2.1.1 Result for Subsonic and Transonic Flow over a Bump

The two-dimensional subsonic flow over a 10% circular bump has been chosen as a first benchmark. The inviscid flow passes through a channel of height H = 1m and length L = 3 m. The total pressure at the inlet was $p_{T,in}$ = 100 kPa, the total temperature $T_{T,in}$ =293.15K and the inlet flow direction was parallel to x - axis. The average outlet pressure corresponds to isentropic Mach number M = 0.1. The solution was obtained using a structured mesh with 150×50 quadrilateral cells. The second benchmark is a transonic flow over the bump using the same geometry as in the previous case. The same boundary conditions were used at the inlet and the average outlet pressure was set to a value corresponding to isentropic M = 0.675. In this case the flow accelerates over the bump and reaches supersonic speed. Then it passes through a shock wave and continues towards the outlet. One can see that all methods including the transonic variant of SIMPLE solver give similar results. Nevertheless the resolution of the shock wave is much better with the AUSM or HLLC based LUSGS

---

[112] Jameson, A., "*Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings*," AIAA Paper 91-1596, June 1991.

[113] Ji˘r´ı F¨ursta, "*On the implicit density based OpenFOAM solver for turbulent compressible flows*", EPJ Web of Conferences 143, 02027- (2017).

[114] Menter, F. R. "*Two-equation eddy-viscosity turbulence models for engineering applications*". *AIAA Journal* 32.8 (1994), pp. 1598–1605.

[115] same as 109.

Figure 3.2.1 Mach number distribution for flow over a Bump

scheme than with the SIMPLE solver which spreads the shock over 4 cell due to higher amount of numerical viscosity. (see **Figure 3.2.1**).

3.2.2 Case Study 2 - Transonic Flow Through a Turbine Cascade

The next benchmark is two-dimensional compressible turbulent flow through test turbine cascade SE-1050. The geometry and boundary conditions correspond to the test from database. The inlet total pressure and total temperature were $p_{tot,in}$ = 100 kPa and $T_{tot,in}$ = 293.15K and the inlet angle was α in = 19.3◦. The average outlet pressure corresponds to isentropic outlet Mach number M_{out} = 1.2, the Reynolds number related to outlet flow conditions and blade pitch was Re = 1.2 ×10^6 and the inlet turbulence intensity was T_{uin} = 2%. The hybrid mesh was composed of structured quadrilateral layer around the blade and unstructured triangular part in the rest of the domain. The mesh contained approximately 66×103 cells and the near-wall refinement corresponded to $y+ \approx 0.2$.

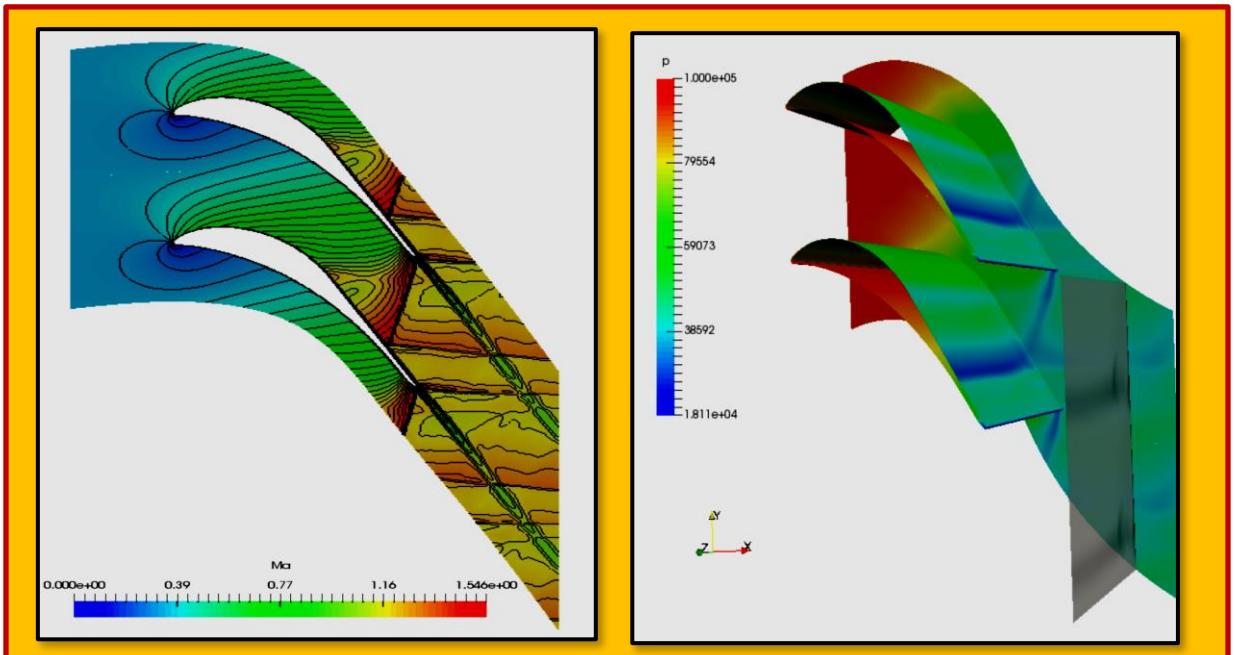


Figure 3.2.2 2D Mach Number Distribution in the Test Turbine Cascade (left) - 3D Pressure Distribution at the Blades (right)

Figure 3.2.2 (left) shows contours of the Mach number in two periods of the mesh obtained using LU-SGS scheme with HLLC flux. One can see here both branches of the shock emanating from the

trailing edge as well as the interaction of the inner branch of the shock wave with the blade. One can see that both numerical methods shows very good agreement experimental data.
The next case is the 3D flow through a test turbine cascade with prismatic blades. The geometry of blades correspond to the previous case and the blade has finite span. The flow regime corresponds to outlet isentropic Mach number M_{out} = 1.2 and Reynolds number related to chord length Re = 1.34×10^6. Inlet conditions include a simple model of boundary layer corresponding to conditions in the experiment. The numerical solution was obtained on an unstructured mesh with 3.3×10^6 cells with near wall refinement to $y+ \approx 0.2$. The **Figure 3.2.2 (right)** shows the distribution of the pressure at the blades and the side wall and the distribution of entropy in the test plane. The regions with higher level of entropy show re-distribution of the energy losses in the vicinity of side walls.

3.2.3 Case Study 3 - 3D Structured/Unstructured Hybrid Navier-Stokes Method for Turbine Blade Rows

Authors : *F.-L. Tsung and J. Loellbach[1], O. Kwon[2], C. Haht[3]*
Affiliation : *[1]Institute for Computational Mechanics in Propulsion[116], NASA Lewis Research Center Cleveland, Ohio*
[2]NYMA, Inc. NASA Lewis Research Center Group, Brook Park, Ohio
[3]NASA Lewis Research Center, Cleveland, Ohio
Sponsorship : *NASA TM-106813, AIAA-94-3369*
Variations of Reproduction *: Minor due to format*

A 3D viscous structured/unstructured hybrid scheme has been developed for numerical computation of high Reynolds number turbomachinery flows [Tsung et al.][117]. The procedure allows an efficient structured solver to be employed in the densely clustered, high aspect-ratio grid around the viscous regions near solid surfaces, while employing an unstructured solver elsewhere in the flow domain to add flexibility in mesh generation. Test results for an inviscid flow over an external transonic wing and a Navier-Stokes flow for an internal annular cascade are presented.

3.2.3.1 Introduction

In modern turbomachinery designs, the rotor and stator blade rows often possess extreme turning angles such that the flow direction deviates greatly from the axial direction. These geometries frequently cause difficulties in generating structured meshes[118]. In order to impose point-to-point periodicity on periodic boundaries in the domain, the computational grids are typically forced into highly skewed shapes that decrease the accuracy of the solvers. One remedy for reducing the grid skewness is to forego the point to point correspondence on the periodic boundaries and to interpolate solution values where necessary. The disadvantage of the non-periodic grid approach is that the local density of grid lines on either side of a periodic boundary may be vastly different. Hence, flow features resolved on one side of the domain boundary may be diffused or lost when interpolated to a coarser grid on the other side. Another solution to grid skewness is to use unstructured meshes.
However, the cost and technology of present day unstructured solvers still leave much room for improvement. The computation of both inviscid and viscous flows on unstructured triangular meshes in two dimensions, and tetrahedral meshes in three dimensions, has matured significantly in recent

[116] located at the Ohio Aerospace Institute adjacent to the NASA Lewis Research Center in Cleveland, Ohio.
[117] F.-L. Tsung, J. Loellbach, O. Kwon, and C. Hah, "*A Three-Dimensional Structured/Unstructured Hybrid Navier-Stokes Method for Turbine Blade Rows*", Prepared for the 30th Joint Propulsion Conference and Exhibit cosponsored by AIAA, ASME, SAE, and ASEE Indianapolis, Indiana, June 27-29, 1994.
[118] Aronone, A., Liou, M-S., and Povinelli, A., "*Transonic Cascade Flow Calculations Using Non-Periodic C-type Grids,*" CFD Symposium on Aero-propulsion, NASA Conference Publication 3078, 1990.

years[119-120]. The ability of unstructured solvers to handle complex geometries has proven to be valuable for many computations. However, the geometric flexibility of unstructured grid solvers is also the source of their disadvantages when compared to structured solvers. Due to the naturally-ordered data connectivity of structured grids, structured solvers require much less memory (especially for implicit schemes), implicit structured solvers are straight forward to code, and turbulence modeling is easier to implement.

The shortcomings of both structured and unstructured methodologies are continuously being addressed and improved. In the meantime, however, it is possible to take advantage of the best properties of both methodologies. In the present paper, a solution procedure which couples an efficient structured solver with an unstructured solver is presented. The hybrid procedure takes advantage of the computational efficiency of structured codes and, at the same time, is able to benefit from the geometric flexibility of unstructured solvers. With the hybrid approach, densely packed structured grids can be placed in the highly viscous regions near solid surfaces, and unstructured grids can be used away from solid surfaces [121]. This approach can avoid the severe grid skewness commonly experienced by fully structured grids around turbine blades with high turning angles. The procedure has various other applications as well, such as providing a means of connecting multi-body geometries for Chimera-type grids to insure node-to-node correspondence[122-123]. The strategy of coupling structured and unstructured methods has been implemented for two dimensional turbomachinery computations by many researchers in the past[124].

The present work extends this strategy to three-dimensional turbomachinery flows. The hybrid solution technique is first tested for an external fixed-wing transonic flow case and the result is compared with a structured solution for validation. The code is then applied to an annular cascade and the result is compared with experimental data.

3.2.3.2 Governing Equation Unstructured Solver

The time dependent, Reynolds averaged, compressible Navier-Stokes equations, which express the conservation of mass, momentum, and energy, are solved. The turbulence viscosity is calculated using the high Reynolds number turbulence model of [Launder and Spalding][125]. The equations of motion, written in an integral form for a bounded domain Ω with a boundary $\partial\Omega$, are

$$\frac{\partial}{\partial t}\iiint_{\Omega}\mathbf{Q}\,dV+\iint_{\partial\Omega}\mathbf{F}(\mathbf{Q}).\hat{\mathbf{n}}\,ds-\iint_{\partial\Omega}\mathbf{G}(\mathbf{Q}).\hat{\mathbf{n}}\,ds=\iiint_{\Omega}\mathbf{S}(\mathbf{Q})dV$$

Eq. 3.2.6

where **Q** is the unknown vector containing the conserved properties

$$Q=[\rho,\rho u,\rho v,\rho w,e_0,\rho k,\rho\varepsilon]^T$$

Eq. 3.2.7

---

[119] Barth, T.J., "*A 3-D Upwind Euler Solver for Unstructured Meshes*," AIAA Paper 91-1548, June 1991.

[120] Batina, J.T., "*A Fast Implicit Upwind Solution Algorithm for Three-Dimensional Unstructured Dynamic Meshes*," AIAA Paper 92-0447, Jan. 1992.

[121] Weatherill, N.P., "*Mixed Structured-Unstructured Meshes for Aerodynamic Flow Simulation*," *Aeronautical Journal,* pp. 111-123, Apr. 1990.

[122] Nakahashi, K. and Obayashi, S., "*FDM-FEM Zonal Approach for Viscous Flow Computation Over Multiple-Bodies,*" AIAA Paper 87-0604, Jan. 1987.

[123] Kao, K.H. and Liou, M.S., "*Direct Replacement of Arbitrary Grid-Overlapping by Non-Structured Grid,*" NASA TM-106601, May 1994.

[124] *Nakahashi, K., Nozaki, O., Kikuchi, K-, and Tamura, A., "Navier-Stokes Computations of Two- and Three-Dimensional Cascade Flow fields,*" Journal of Propulsion and Power, Vol. 5, No. 3, pp.320-326, 1989.

[125] Launder, B.E. and Spalding, D.B., *'The Numerical Computation of Turbulent Flows*," Computer Methods in Applied Mechanics and Engineering, Vol. 3, 1974.

In the above equation, ρ is the fluid density. *u, v,* and *w* are the Cartesian velocity components in x, y, and *z* directions, respectively. e_o is the total energy per unit volume. The turbulent kinetic energy and turbulent kinetic energy dissipation rate are represented by k and ε. The vector **F** is the convective flux term, **G** is the viscous term, and **S** is the source term containing the production and destruction of turbulent kinetic energy. A complete description of the governing equations is presented in [Reference][126] and will not be detailed here for brevity.

3.2.3.3 Inviscid Flux Spatial Discretization

The inviscid flux across each cell face x is computed using Roe's flux-difference splitting formula[127] Here, $\mathbf{Q}_L$ and $\mathbf{Q}_R$ are the state variables to the left and right of the interface K. The matrix **A** is computed by evaluating

$$\mathbf{F_k} = \frac{1}{2}\{\mathbf{F}(\mathbf{Q}_L) - \mathbf{F}(\mathbf{Q}_R) - |\bar{\mathbf{A}}|(\mathbf{Q}_R - \mathbf{Q}_L)\}_k \quad , \quad \bar{\mathbf{A}} = \frac{\partial \mathbf{F}}{\partial \mathbf{Q}}$$

Eq. 3.2.8

with Roe-averaged quantities so that

$$\mathbf{F}(\mathbf{Q}_R) - \mathbf{F}(\mathbf{Q}_L) = \bar{\mathbf{A}}\,[\mathbf{Q}_R - \mathbf{Q}_L]$$

Eq. 3.2.9

is satisfied. For a first-order scheme, the primitive variables at each cell face are set equal to the cell-centered averages on either side of the face. For a higher-order scheme, estimation of the state at each face is achieved by interpolating the solution at each time step with a Taylor series expansion in the neighborhood of each cell center. The cell-averaged solution gradients required at the cell center for the expansions are computed using Gauss' theorem by evaluating the surface integral for the closed surface of the tetrahedra. This process can be simplified using geometric invariant features of tetrahedra. The resulting second-order formula for the flow state at each cell face can be written as

$$\mathbf{q_{f_{1,2,3}}} = q_i + \frac{1}{4}\left[\frac{1}{3}\,(q_{n1} + q_{n2} + q_{n3}) - q_{n4}\right]$$

Eq. 3.2.10

where the subscripts nl , n2 , n3 denote the nodes comprising face f 1,2,3 of cell i, and n4 corresponds to the opposite node. The expansion also requires the nodal value of the solution, which can be computed from the surrounding cell center data using a second order accurate pseudo-Laplacian averaging procedure as suggested by [Homes and Connell][128]. The three dimensional extension by [Frink][129] is adopted in the present calculations. The convective terms of the turbulence equations are calculated using a first-order accurate scheme in the present paper to reduce the computational cost and to ensure the numerical stability of the time integration[130].

3.2.3.4 Viscous Flux Spatial Discretization

---

[126] Kwon, O.J. and Hah, C., "*Solution of the 3-D Navier-Stokes Equations with a Two-Equation Turbulence Model on Unstructured Meshes Applied to Turbomachinery*," AIAA Paper 94-1833, June 1994.
[127] Roe, P.L., "*Characteristic-Based Schemes for the Euler Equations*," Annual Review of Fluid Mechanics, 1986.
[128] Holmes, D.G. and Connell, S.D., "*Solution of the 2D Navier-Stokes Equations on Unstructured Adaptive Grids*," AIAA Paper 89-1932, June 1989.
[129] Frink, N.T., "*Recent Progress Toward a Three-Dimensional Unstructured Navier-Stokes Flow Solver*," AIAA Paper 94-0061, Jan. 1994.
[130] Mavriplis, D.J., "*Multigrid Solution of Compressible Turbulent Flow on Unstructured Meshes Using a Two-Equation Model*," AIAA Paper 91-0237, Jan. 1991.

The evaluation of the viscous term G requires first derivatives of the velocity, temperature, and k-E values at the cell faces. They are achieved by first evaluating the gradient of each required flow quantity at the cell center from the known primitive variables at each time step. The gradient of the desired quantity is obtained by applying the gradient theorem,

$$\nabla \mathbf{Q}_n = \frac{1}{V_\Omega} \oint_{\partial\Omega} \mathbf{Q}\hat{\mathbf{n}} dS$$

Eq. 3.2.11

where Ω represents the volume of the domain over which the theorem is applied. The scalar quantity ¢ can be the three components of velocity, the temperature, or turbulence quantities. In the present calculations, the integral domain is defined as the individual tetrahedral cell, and the surrounding surface area as2 consists of the four triangular surfaces covering the cell. This formulation is consistent with the numerical procedure of evaluating the convective fluxes of the present cell-centered scheme. Once the gradients of the desired quantities are known at the cell center, nodal values are calculated using the Pseudo-Laplacian averaging mentioned earlier for the convective terms. The flux through each of the triangular faces in **Eq. 3.2.11** is obtained by averaging the three nodal values for the triangle. Once the gradients of the primitive variables are obtained, the shear stresses and can be calculated, from which **G** is evaluated at the cell center. The nodal values of these quantities are calculated once again by applying the Pseudo-Laplacian averaging of the surrounding cell center values. The surface flux of these quantities in **Eq. 3.2.6** is obtained by taking the average of the three nodal values for each triangular face of each cell.

3.2.3.5 Time Integration

The solution vector is integrated in time using the implicit Euler method, which is first-order accurate in time. The nonlinear inviscid flux vectors are linearized at every time level about their values at the previous time level using Taylor expansions, e.g.

$$\mathbf{E}^{n+1} = \mathbf{E}^n + \frac{\partial \mathbf{E}^n}{\partial \mathbf{Q}} \Delta \mathbf{Q}^{n+1} + O(\Delta\boldsymbol{\tau})^2 = \mathbf{E}^n + \mathbf{A}^n \Delta \mathbf{Q}^{n+1}$$

Eq. 3.2.12

The viscous terms are evaluated explicitly. Explicit treatment of the viscous terms still permits the use of large time steps since the Reynolds numbers of interest here are fairly large. To further reduce computational time and memory, the radial/spanwise flux derivatives are treated explicitly at the old time level, but the new values are incorporated as soon as they become available. This explicit treatment of the spanwise flux terms enables the scheme to solve the three-dimensional equations by solving one spanwise station implicitly at a time. To eliminate any dependency the solution may have on the spanwise marching direction, the solver reverses the marching direction with every spanwise sweep. The resulting left-hand-side of the matrix equation is approximately factored into alternating directions

$$\left(\mathbf{I} + \Delta\tau\delta_\xi \mathbf{A}^n\right)\left(\mathbf{I} + \Delta\tau\delta_\zeta \mathbf{C}^n\right)\Delta\mathbf{Q}^{n+1} = -\Delta\tau\left[\left(\mathbf{E}^n - \mathbf{E}_v^{n,n+1}\right)_\xi + \left(\mathbf{F} - \mathbf{F}_v^{n,n+1}\right)_\eta + \left(\mathbf{G}^n - \mathbf{G}_v^{n,n+1}\right)_\zeta\right]$$

Eq. 3.2.13

The implicit operating matrices are then diagonalized using a similarity transformation [27].

3.2.3.6 Spatial Discretization

Standard second-order central differencing is used for the spatial derivatives. Spectral-radius scaled fourth/second-difference artificial dissipation[131] is added for stability and to eliminate oscillations near shocks. The viscous terms are evaluated using half-point central differencing so that the computational stencil for the stress terms uses only three nodes in each of the three directions.

3.2.3.7 Hybrid Coupling

Since the structured code has multi-block capability, the coupling procedure treats the unstructured portion of the hybrid meshes as a separate block. The blocks are solved independently with respect to each other. Explicit boundary conditions are updated at the end of each time iteration. The two codes are loosely coupled, essentially running independently of each other. The only interaction between the structured solver and the unstructured solver is through boundary conditions. The advantage of this formulation is that almost any two given codes can be coupled together. In this paper, a central-differencing, finite-difference procedure and an upwind cell-centered finite volume procedure are coupled. The structured solver is formulated in an inertial frame of reference whereas the unstructured solver is in a blade-fixed coordinate system. For non-moving-boundary cases, such as pressure-driven turbine stator flows, there is no difference between the two formulations. However, for moving-boundary cases, care must be taken when transferring data between the two solvers depending on how boundary conditions are prescribed. For example, consider a wing traveling at a constant velocity.

Since the structured solver assumes the wing is moving and the unstructured solver assumes the domain is fixed in space, the total energy must be converted between the two frames of reference. At the end of each iteration, mass, momentum, and energy values are communicated between the structured and unstructured solvers. Since the unstructured solver is a cell-centered finite-volume scheme and the structured solver is a vertex-based finite-difference scheme, the unstructured and the structured grids are overlap by one layer at the boundary in order to minimize interpolation and extrapolation errors at the boundary. By overlapping the grids instead of abutting them against each other, the boundary nodes for the structured solver are interpolated from the interior cell center values of the unstructured solver Similarly, the cell boundary face values of the unstructured solver are interpolated from interior structured nodes. An illustration of the overlapped grid in 2D is shown in **Figure 3.2.3.** Once the grid construction is completed, a table is generated to index unstructured nodes to their corresponding structured nodes, and vise-versa. The index is similar to the table generated between structured block boundaries for multi-block calculations.

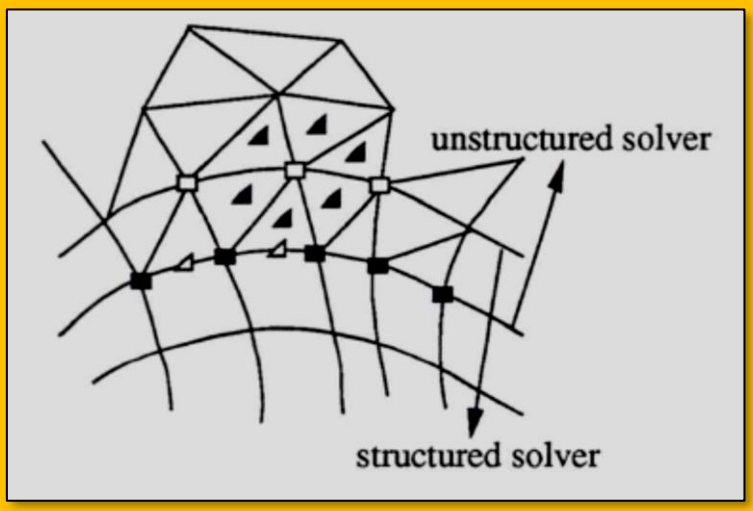


Figure 3.2.3 Structured-Unstructured Mesh Overlapping by One Layer – Courtesy of [Tsung et al.]

3.2.3.8 Results and Discussion

[131] Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solution of the Euler Equations by Finite-Volume Methods Using Runge-Kutta Time-Stepping Schemes," ALAA Paper 81-1259, June 1981.

Validation for both the structured and the unstructured solvers have been documented previously and will not be presented here. To test the structured-unstructured hybrid analysis, calculations for a transonic inviscid flow over a fixed wing and a viscous flow for a turbine stator blade are presented. For all calculations in this paper, a structured grid is first generated for the entire geometry. The unstructured grids are generated by subdividing a portion of the structured cells each into six unstructured tetrahedra.

3.2.3.8.1 Case 1 : Transonic Wing

The first calculation selected for the hybrid scheme is a transonic wing at a freestream Mach number of 0.8 and at 1.25 0 angle-of-attack. For this test case, flow periodicity is imposed on the two end wall boundaries to simulate a two-dimensional flow. A purely structured solution for this case is first obtained for comparison. An O-H grid topology is used; that is, an O-grid is generated at each spanwise station. For the structured calculation, each spanwise station consists of 121 (streamwise) x 41 (normal) points. Six spanwise stations are required due to the memory allocation algorithms for the multi-block structured solver. The number of spanwise stations used does not alter the results since the code simulates a two-dimensional problem.

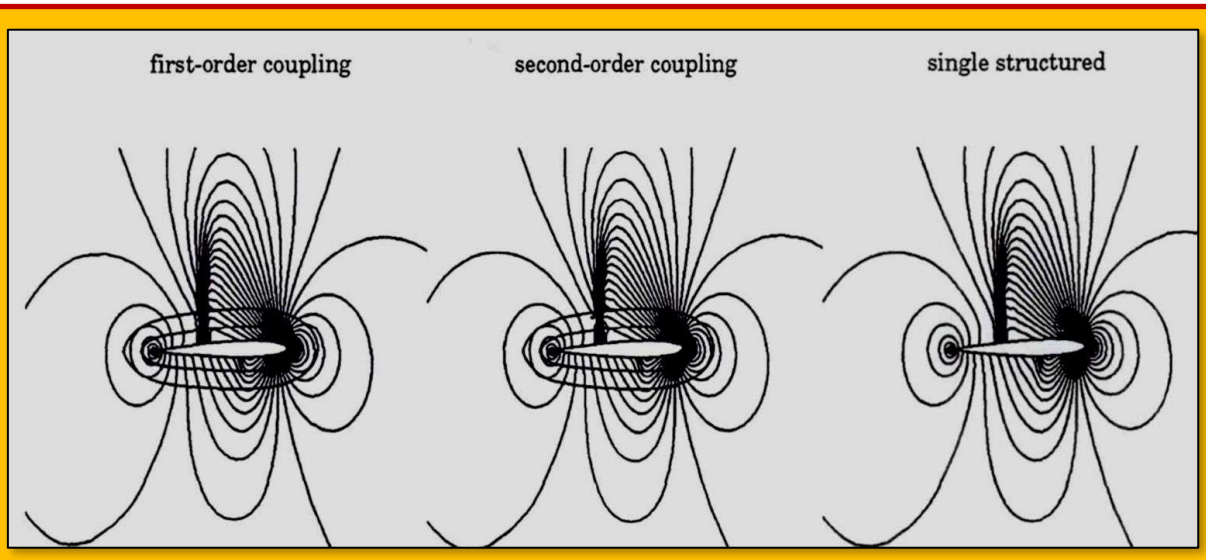


Figure 3.2.4 Pressure contour lines across the structured-unstructured zonal interface – Courtesy of [Tsung et al.]

For the structured-unstructured coupled procedure, the grid is divided into three zones. The inner zone, which wraps around the solid surface (121x12), is calculated using the structured solver. The middle zone (121x6), which wraps around the inner structured zone, is subdivided into tetrahedra for the unstructured solver. The outer zone, which extends from the middle zone to the free stream, is once again calculated using the structured solver. The purpose of dividing the calculation into three zones is to simulate a Chimera type of grid where a simple background grid is generated for the entire domain and a structured grid is used around a solid object. Instead of interpolating between the two grids, an unstructured solver can be placed to connect the two set of grids and obtain node-to-node correspondence which remove the needs for interpolation. For a finite volume approach, this also insures flow conservation. This approach is also recommended for multi-body calculations. To examine the coupling between the structured/unstructured solver across the three-zones, pressure contour lines are shown in **Figure 3.2.4** for the first-order and the second-order coupling, along with the single block structured solver for comparison. For the first order coupling, a slight discontinuity exists across the zonal boundary. However, the discontinuity is small and barely perceptible on a global scale. For the second-order coupling, the contours show the flow is smooth with little

indication of the presence of the zonal boundaries. The mass flows in and out of the sandwiched unstructured zone are within 0.03% of each other, indicating that good flow conservation is achieved. The unstructured shock is somewhat more compact then the structured solver. One reason for this is that every structured cell becomes six tetrahedra so that in the streamwise direction, the unstructured cells are twice as dense as the structured cells. The present hybrid procedure converges considerably slower than the single structured grid due to the explicit formulation of the incorporated unstructured solver, which requires a low CFL number.

3.2.3.8.2 Case 2 : 3D Viscous Flow Through Turbine Blades-Hybrid Procedure

The second test case selected is a 3D annular cascade [29]. The geometry consists of an annular ring of 36 turbine stator vanes. The blades are 38.10 mm in span, untwisted, and of constant profile with an axial chord of 38.23 mm. The stator has a tip diameter of 508 mm and a hub-to-tip radius ratio of 0.85. The inlet flow angle is parallel to the axis of the cascade. The Reynolds number based on the inlet total quantities and axial chord length is 898,650. The inlet total pressure and total temperature at one axial-chord length upstream of the blade leading edge are known from experiment. The exit hub static pressure to inlet total pressure ratio is 0.65 at 2.6 axial-chord lengths downstream of the blade trailing edge.

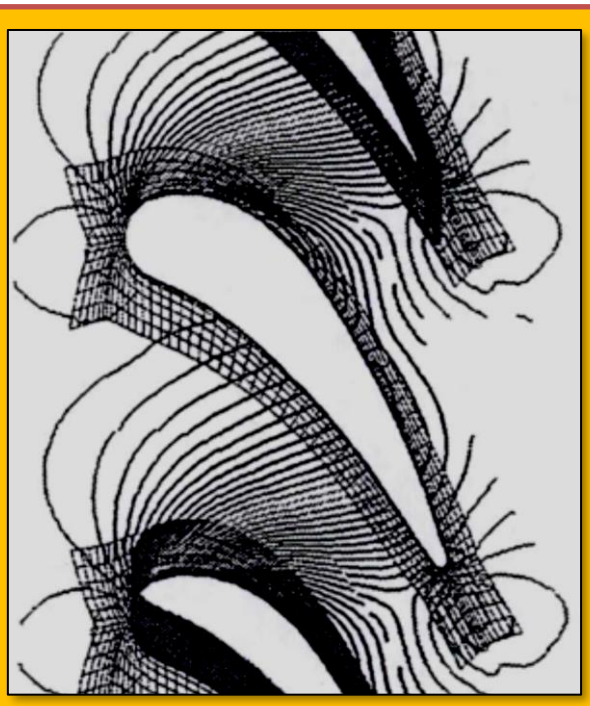


Figure 3.2.5 Pressure Contour Across the Structured-Unstructured Interface – Courtesy of [Tsung et al.]

For the hybrid calculation, the computational domain extends one axial-chord length upstream of the blade leading edge and 2.6 axial-chord lengths downstream of the blade trailing edge. The exit hub static pressure to inlet total pressure ratio is known from experiment. The grid has 105 (inlet to exit) x29 (blade to blade) x15 (hub to tip) nodes and 23,000 cells are subdivided into unstructured cells. The structured grid wraps around the solid surfaces of the hub and the turbine blade while the rest of the domain is filled with unstructured cells. For the present grid, only 15 hub to shroud stations are used and the grid density near the trailing edge is very coarse (only 3 points defining the rounded trailing edge). While such a coarse grid is not acceptable for accurate prediction of viscous effects, it is sufficient for monitoring the communication between the structured and unstructured zones and to predict the overall flow field. A no-slip boundary condition is applied at the hub and on the blade surfaces, and an inviscid slip condition is applied on the shroud. In the coupled calculation, the unstructured solver is run without turbulence modeling because the effects of turbulence are much reduced outside the boundary layer. The pressure contour lines across the structured and unstructured zones are plotted in **Figure 3.2.5.**

3.2.3.9 Concluding Remarks

A 3D unstructured, Navier-Stokes flow solver has been coupled with a three dimensional structured code to allow structured unstructured hybrid calculations. The two codes are loosely coupled and interact only through boundary conditions. The structured solver is a center differenced finite-difference scheme and the unstructured solver is an upwind-differenced finite volume scheme. The

hybrid procedure has been tested for a transonic wing and an annular cascade, and good results have been obtained. The flow properties across the boundary between the structured and unstructured portions of the grid are smooth and well behaved. The results show that two distinct discretization techniques can be coupled together with little effect on the solution accuracy, as long as both are of same order.

3.2.3.10 References

1. Aronone, A., Liou, M-S., and Povinelli, A., "Transonic Cascade Flow Calculations Using Non-Periodic C-type Grids," CFD Symposium on Aero propulsion, NASA Conference Publication 3078, 1990.
2. Barth, T.J., "A 3-D Upwind Euler Solver for Unstructured Meshes," AIAA Paper 91-1548, June 1991.
3. Batina, J.T., "A Fast Implicit Upwind Solution Algorithm for Three-Dimensional Unstructured Dynamic Meshes," AIAA Paper 92-0447, Jan. 1992.
4. Frink, N.T., "Upwind Scheme for Solving the Euler Equations on Unstructured Tetrahedral Meshes," *AIAA Journal, Vol.* 30, No. 1, pp. 70-77, Jan. 1992.
5. Mavriplis, D.J., "Three-Dimensional Unstructured Multigrid for the Euler Equations, "ALAA Journal, Vol. 30, No. 7, pp. 1753-1761, July 1992.
6. Dawes, W.N., "The Simulation of Three-Dimensional Viscous Flow in Turbomachinery Geometries Using a Solution-Adaptive Unstructured Mesh Methodology," ASME Paper 91-GT-124, 1991.
7. Spragle, G.S., Smith, WA, and Yadlin, Y., "Application of an Unstructured Flow Solver to Planes, Trains, and Automobiles," AIAA Paper 93-0889, Jan. 1993.
8. Weatherill, N.P., "Mixed Structured-Unstructured Meshes for Aerodynamic Flow Simulation," *Aeronautical Journal,* pp. 111-123, Apr. 1990.
9. Mathur, S.R., Madavan, N.K., and Rajagopalan, R.G., "A Solution-Adaptive Hybrid-Grid Method for the Unsteady Analysis of Turbomachinery, "AIAA Paper 93-3015, July 1993,
10. Soetrisno, M., Imlay, S.T., and Roberts, D.W., "A Zonal Implicit Procedure for Hybrid Structured-Unstructured Grids," AIAA Paper 94-0645, Jan, 1994.
11. Nakahashi, K. and Obayashi, S., "FDM-FEM Zonal Approach for Viscous Flow Computation Over Multiple-Bodies," AIAA Paper 87-0604, Jan. 1987.
12. Kao, K.H. and Liou, M.S., "Direct Replacement of Arbitrary Grid-Overlapping by Non-Structured Grid," NASA TM-106601, May 1994.
13. Nakahashi, K., Nozaki, O., Kikuchi, K-, and Tamura, A., "Navier-Stokes Computations of Two- and Three-Dimensional Cascade Flowfields," *Journal of Propulsion and Power,* Vol. 5, No. 3, 1989.
14. Hwang, C.J. and Liu, J.L., "Inviscid and Viscous Solutions for Airfoil/Cascade Flows Using a Locally Implicit Algorithm on Adaptive Meshes, " *Journal of Turbomachinery,* Vol. 113, pp.553-560, Oct. 1991.
15. Mathur, A., Sanjay, R., Madavan, B., Nateri, K., and R.ajagopalan, R.G., "A Hybrid Structured-Unstructured Grid Method for Unsteady Turbomachinery Flow Computations," ALAA paper 93-0387, Jan. 1993.
16. Launder, B.E. and Spalding, D.B., 'The Numerical Computation of Turbulent Flows," *Computer Methods in Applied Mechanics and Engineering, Vol.* 3, 1974, pp. 269-289.
17. Kwon, O.J. and Hah, C., "Solution of the 3-D Navier-Stokes Equations with a Two-Equation Turbulence Model on Unstructured Meshes Applied to Turbomachinery," AIAA Paper 94-1833, June 1994.
18. Roe, P.L., "Characteristic-Based Schemes for the Euler Equations," *Annual Review of Fluid Mechanics, Vol.* 18, pp. 337-365, 1986.
19. Holmes, D.G. and Connell, S.D., "Solution of the 2D Navier-Stokes Equations on Unstructured Adaptive Grids," AIAA Paper 89-1932, June 1989.
20. Frink, N.T., "Recent Progress Toward a Three-Dimensional Unstructured Navier-Stokes Flow Solver," AIAA Paper 94-0061, Jan. 1994.
21. Mavriplis, D.J., "Multigrid Solution of Compressible Turbulent Flow on Unstructured Meshes Using a Two-Equation Model," AIAA Paper 91-0237, Jan. 1991.

22. Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solution of the Euler Equations by Finite-Volume Methods Using Runge-Kutta Time-Stepping Schemes," ALAA Paper 81-1259, June 1981.
23. Kunz, R.F. and Lakshminarayana, B., "Stability of Explicit Navier-Stokes Procedures Using k-e and k-e/Algebraic Reynolds Stress Turbulence Models," *J. of Computational Physics,* Vol. 103, 1992.
24. Tsung, F.-L. and Sankar, L.N., "Numerical Simulation of Flow Separation for Rotors and Fixed Wings," AIAA paper 92-0635, Jan. 1992.
25. Baldwin, B.S., Lomax, H., "Thin-Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78-257, June 1978.
26. Rizk, Y.M. and Chausee, D.S., "Three-Dimensional Viscous-Flow Computations Using a Directionally Hybrid Implicit-Explicit Procedure," AIAA Paper 83-1910, 1983.
27. Pulliam, T.H. and Chaussee, D.S., "A Diagonal Form of an Implicit Approximate-Factorization Algorithm," *J. of Computational Physics, Vol.* 39, pp. 347-363, 1981.
28. Kwon, O.J. and Hah, C., "Three-Dimensional Unstructured Grid Euler Method Applied to Turbine Blades," AIAA Paper 93-0196, Jan. 1993.
29. Goldman, L.J. and McLallin, K.L., "Cold-Air Annular-Cascade Investigation of Aerodynamic Performance of Core-Engine-Cooled Turbine Vanes. I-Solid-Vane Performance and Facility Description," NASA TM X-3224, April 1975.

3.3 Some Observations on Usage of Pressure Based vs. Density Based Schemes

Several researchers commented on CFD blogs as where to use the Pressure or Density based solver, and it was mentioned here as confidence builder measure. Some interesting ones are:

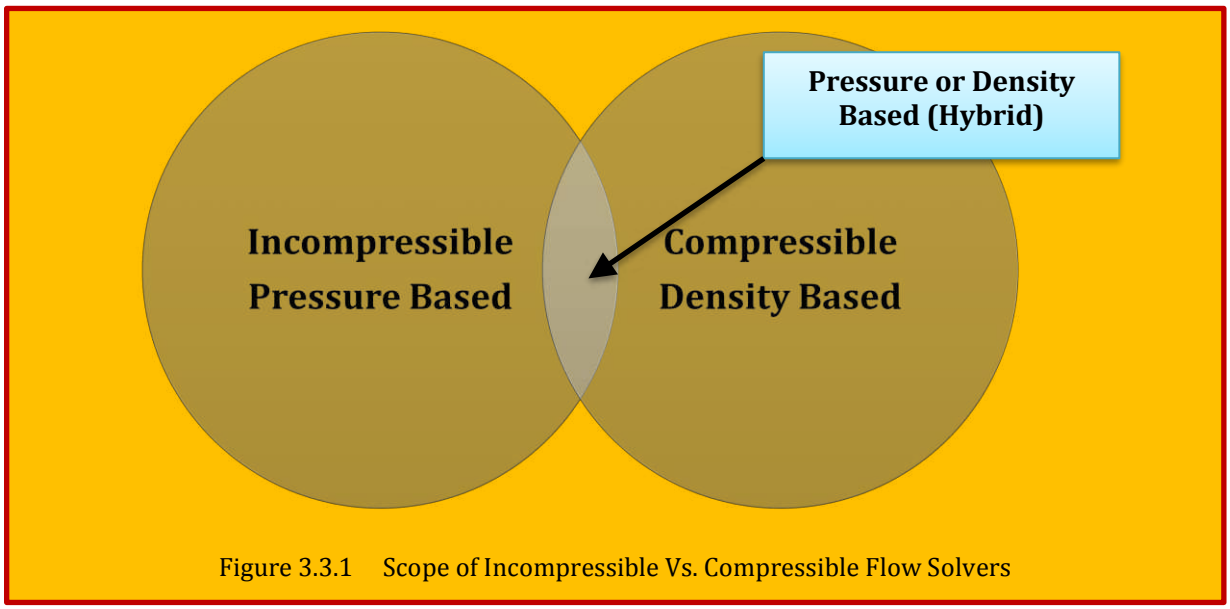


Figure 3.3.1 Scope of Incompressible Vs. Compressible Flow Solvers

- Strictly speaking, the ***pressure-based*** approach was developed for low-speed incompressible flows, while the ***density-based*** approach was mainly used for high-speed compressible flows. However, recently both methods have been extended and reformulated to solve and operate for a wide range of flow conditions beyond their traditional or original intent (see **Figure 3.3.1**). In both methods the velocity field is obtained from the momentum equations. In the density-based approach, the continuity equation is used to obtain the density field while the pressure field is determined from the equation of state. On the other hand, in the pressure-based approach, the pressure field is extracted by solving a pressure or pressure correction equation which is obtained by manipulating continuity and momentum equations[132].
- The pressure-based solver traditionally has been used for incompressible and mildly compressible flows. The density-based approach, on the other hand, was originally designed for high-speed compressible flows. Both approaches are now applicable to a broad range of flows (from incompressible to highly compressible), but the origins of the density-based formulation may give it an accuracy (i.e. shock resolution) advantage over the pressure-based solver for high-speed compressible flows.
- The reason many people extend the simple algorithm for compressible flows is for using a single unified solver which is valid over the entire flow regime. Many algorithms which are developed for compressible flows are extended to incompressible flows through preconditioning for the same reason. Also in certain flow scenarios, there may be wide variations of the Mach number in the flow region and hence an unified solver is a better way of handling it. [Harish, CFD online].
- Conferring to [Mateu], the main difference is the approach for density variation. The SIMPLE type algorithm seek for pressure correction with some assumption or simplification. In other words, we first have an approximate velocity field, for this we wish to correct it such that continuity satisfies. The velocity correction are expressed in terms of pressure correction and thus we get pressure correction equation, that is solved to get new

[132] Johnson Emmanuel , CFD on line.

pressure field. When the flow is compressible, the SIMPLE type method goes little further and relate density change to pressure correction. Since flux is related to density change, this could also be included into pressure correction equation. This way you get pressure correction and you update velocity and density. Density based solvers work on flux balances directly I suppose.

According to [Gaurav Dhir] everything is boils down to ***time scale***. The ***density based*** solvers calculate their ***timestep*** based on the ***acoustic timescale***. In case of low Mach number flows (M < 0.3), speed of sound is very high (almost approaching infinity). Therefore, in order to ensure stability of the solver, a very low timestep will be needed leading to extremely high computational cost.
A ***pressure based solver*** solves this problem in an indirect manner. For strictly incompressible flow calculations (M = 0), a Poisson's equation for pressure can be obtained by manipulating the continuity and momentum equations. The pressure based solver solves this equation in an implicit manner. It turns out that while performing this manipulation, we have completely neglected the acoustic timescales (due to M = 0 approximation). Therefore, the pressure based solver now calculates the timestep based on flow velocity which is much smaller than the acoustic velocity. This reduces the computational cost for incompressible flow simulations. To get a more intuitive insight into the physics, the above statement can also be interpreted as saying that the pressure based solver resolves all features based on the ***flow timescale*** and neglects features emerging from ***acoustic timescale***.

3.4 Residual Implication

In short, the non-linear NS equations are linearized and the accuracy of the linearized equations to the non-linear ones is the residual where the linear solver solves the linear equations. The discretization of the spatial terms in the governing equations results in a large coupled set of ordinary differential equations of the form

$$\frac{d(VM\mathbf{W})}{dt} + \mathbf{R}(\mathbf{W}) = 0$$

Eq. 3.4.1

Where V represents the local control volume, M the mass matrix, and **R(W)** the spatially discretized terms. For a vertex-based scheme, the mass matrix relates the average value of a control volume to the values at the vertices of the mesh. Both V and M are constants for static meshes, and can therefore be taken outside of the time derivative, and in absence of M it becomes

$$V\frac{d\mathbf{W}}{dt} + \mathbf{R}(\mathbf{W}) = 0$$

Eq. 3.4.2

For steady-state cases, these equations must be integrated in time towards $t \to \infty$, where the time derivatives become vanishingly small. Since time accuracy is not a concern in such cases, and each equation may be advanced with the maximum permissible time step, as determined from local stability considerations. There are two general methods of solving, as Explicit or Implicit scheme, defined following.

3.4.1 Explicit Schemes

A simple time integration scheme is obtained by replacing the time derivative by a simple forward difference and evaluating the residual at the current time level:

$$V\frac{\mathbf{W}^{n+1} - \mathbf{W}^{n}}{\Delta t} + \mathbf{R}(\mathbf{W}^{n}) = 0$$

Eq. 3.4.3

This corresponds to a single-stage explicit scheme, since updates are obtained from one evaluation of currently available quantities. Like ***Runge-Kutta*** schemes for ordinary differential equations, multistage time-stepping schemes (which are required with higher-order discretization for stability reasons) involve the combination of updates obtained at multiple explicitly evaluated intermediate states [Mavriplis][133]. The stage coefficients of these schemes can be optimized either to provide large time steps at the expense of time accuracy or to enhance high-frequency damping properties of the scheme, which is required in the context of a multigrid algorithm [Jameson et al][134] and [van Leer et al.][135]. These schemes are extremely simple to implement and require little additional computer storage. However, stability considerations restrict the maximum permissible time step to values proportional to the local cell size. Thus, finer meshes lead to smaller time steps, which in turn lead to larger solution times. For very fine meshes, the convergence of explicit schemes becomes unacceptably slow, and more sophisticated solution strategies must be adopted.

3.4.2 Implicit Schemes

An implicit scheme is obtained by evaluating the spatial residual terms at the new time level $n+1$. Since these quantities are not known explicitly, a linearization must be performed about the current time level:

$$\left(\frac{\mathrm{V}}{\Delta \mathrm{t}}+\frac{\partial \mathbf{R}}{\partial \mathbf{W}}\right) \Delta \mathbf{W}=-\mathbf{R}\left(\mathbf{W}^{\mathrm{n}}\right)$$

Eq. 3.4.4

Where $\partial \mathbf{R}/\partial \mathbf{w}$ represents the Jacobian and constitutes a large sparse matrix[136]. At each time step, the above linear system must be solved for the corrections $\Delta \mathbf{w} = \mathbf{w}^{n+1} - \mathbf{w}^{n}$ from which the flow variables can be updated. Implicit schemes may be classified by the degree to which the true Jacobian matrix is approximated. If an exact linearization is employed, the method is unconditionally stable and reduces to Newton's method for $\Delta t \rightarrow \infty$. Although the quadratic convergence properties of Newton's method generally produce solutions in a very small number of time steps (often of the order of 10) [Venkatakrishnan & Barth][137], [Barth & Linton][138], [Nielsen][139], each time step requires the inversion of a large sparse matrix, which becomes prohibitively expensive in terms of storage and CPU-time for fine meshes. A common simplification is to replace the exact linearization with one based on a first-order discretization [Mavriplis and Anderson][140]. While this considerably reduces the storage requirements of the linear system and improves its condition number, it precludes attaining quadratic convergence rates, owing to the use of an inexact linearization. This in turn favors the use of iterative methods to solve the linear system, which can be used to converge the system only partially, because exact inversion of the Jacobian is no longer beneficial, owing to the mismatch between Jacobian and residual. However, rigorous criteria for determining the optimal level of convergence of the linear system at each time step have yet to be determined. Simple iterative

---

[133] D. J. Mavriplis, "*Unstructured Grid Techniques*", Annu. Rev. Fluid. Mech. 1997. 29:473–514.

[134] Jameson A, Schmidt W, Turkel E.," *Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes*", *AIAA Pap. 81-125- 1981.*

[135] van Leer B, Tai CH, Powell KG.," *Design of optimally-smoothing multi-stage schemes for the Euler equations*", *AIAA Pap. 89-1933,1989.*

[136] See 148.

[137] Venkatkrishnan V, Barth TJ., "*Application of direct solvers to unstructured meshes for the Euler and Navier-Stokes equations using upwind schemes*", *AIAA Pap. 89-0364, 1989.*

[138] Barth TJ, Linton SW.,"*An unstructured mesh Newton solver for compressible fluid flow and its parallel implementation*", *AIAAPap. 95-022, 1995.*

[139] Nielsen EJ, Anderson WK, Walters RW, Keyes DE.,"*Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code*", *Proc. AIAA CFD Conf., 12th, San Diego. AIAA Pap. 95-1733-CP- 1995.*

[140] Anderson WK, Rausch R, Bonhaus D.,"*Implicit Multigrid Algorithms for Incompressible Turbulent Flows On Unstructured Grids*", *Proc. AIAA CFD Conf., 12th, San Diego. AIAA Pap. 95-1740-CP, 1995.*

schemes such as Jacobi and Gauss-Seidel schemes have been utilized successfully in the context of implicit schemes with first-order linearization. However, since these are still local techniques, their convergence degrades with grid size. More sophisticated techniques such as preconditioned GMRES methods provide enhanced convergence rates, particularly when applied with powerful preconditioners such as Incomplete Lower Upper (ILU) factorization methods, which provide more global information for the solution of the linear system.

Although 1st order linearization methods require substantially less memory overheads than exact linearization methods, their memory requirements are still nontrivial. For example, on a 3D tetrahedral grid, the first-order Jacobian of a vertex-based scheme requires on the order of 350 storage locations per vertex, over three times the number required for the implementation of an explicit scheme. An ILU preconditioning strategy can require an equivalent additional amount of memory. Since GMRES methods only require the evaluation of matrix vector products of the form $(\partial R/\partial w).\Delta w$, it is possible to forgo the storage of the Jacobian, and evaluate the Jacobian vector product directly by finite-difference techniques:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{W}} \cdot \Delta \mathbf{W} = \frac{\mathbf{R}(\mathbf{W} + \varepsilon\, \Delta \mathbf{W}) - \mathbf{R}(\mathbf{W})}{\varepsilon}$$

Eq. 3.4.5

Where ε represents a small parameter. This requires multiple residual evaluations (one for each matrix vector product) and represents a classic tradeoff between storage and computation, particularly for expensive nonlinear residual constructions. However, it also permits the use of the more accurate second order linearization in the implicit scheme, by using the same second-order residual in the finite difference evaluation as in the right-hand side of equation, as is done in the so-called ***Newton-Krylov*** methods.

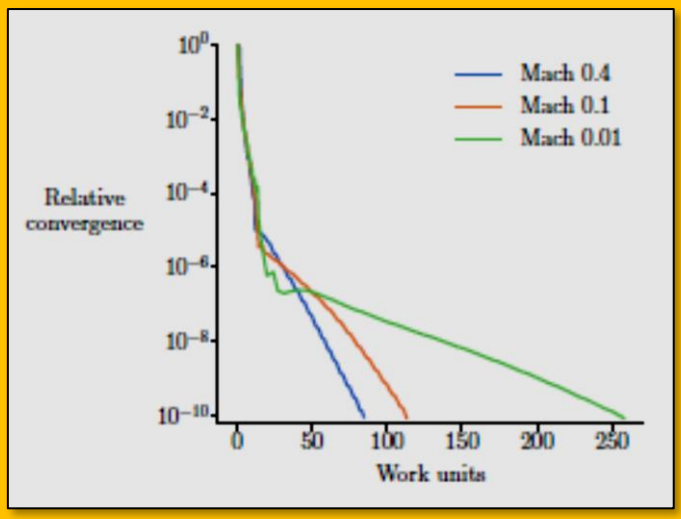


Figure 3.4.1 Cost increases as Mach number decreases (NACA 0012)

3.4.2.1 Convergence Rate and Storage Requirement

Compressible flow solvers face slower convergence at low Mach numbers [Seraj et al.][141]. Using NACA 0012 as a test case, the solution for Mach 0.01 is three times slower than Mach 0.4 (**Figure 3.4.1**). The slower convergence is the result of the difference in magnitude between the advective and acoustic wave speeds. For Newton-based solvers, this is reflected in the stiffness of the linear system that is solved at each nonlinear iteration[142].

Figure 3.4.2 compares the convergence rates of a ***Newton-Krylov*** method that employs ***ILU preconditioning*** with convergence of a ***Gauss-Seidel iterative method*** applied to a first-order linearization, as well as with the convergence obtained by a multigrid scheme using the same Gauss-

[141] Sabet Seraj, Anil Yildirim, Joshua L. Anibal, Joaquim R. R. A. Martins, "*Improving the Performance of a Compressible RANS Solver for Low and High Mach Number Flows*", Eleventh International Conference on Computational Fluid Dynamics (ICCFD11), Maui, HI, USA, July 11-15, 2022.

[142] See Previous

Seidel solver as a multilevel smoother ***(Solid line: Gauss Siedel; Dotted line: ILU Preconditioned Newton-Krylov; Dashed line: Multigrid method -using Gauss Siedel as smoother)***. The depicted case involves the solution of two-dimensional subsonic inviscid flow over a NACA 0012 airfoil on a mesh of 8578 points, using a vertex-based second-order upwind scheme[143].

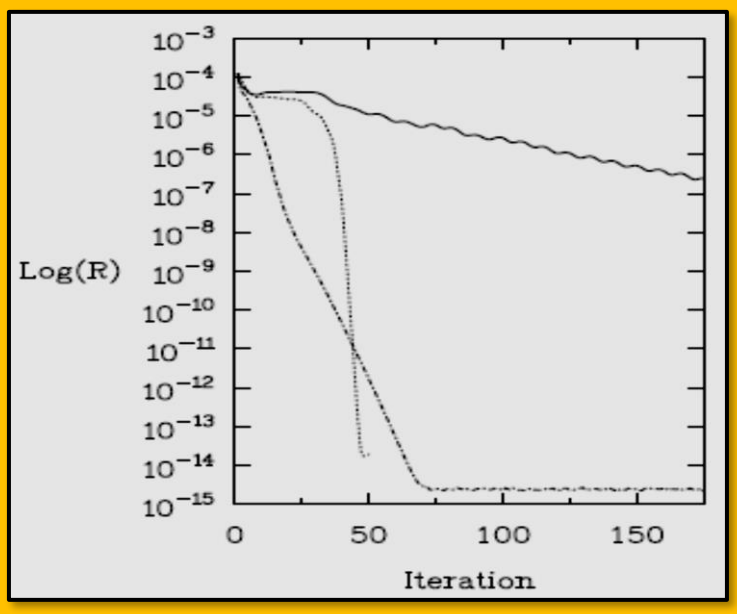


Figure 3.4.2 Comparing the Convergence Rates of the ILU-Preconditioned Newton-Krylov Method

The quadratic convergence property of the Newton-Krylov approach is evident, which achieves a very rapid asymptotic convergence rate in terms of number of iterations. When compared in terms of overall CPU-time, the ***Newton-Krylov*** scheme, although still superior to the Gauss-Seidel implicit method, is overtaken by the multigrid scheme, due to the expense of each ***Newton-Krylov*** iteration. Additional improvements to *Newton-Krylov* methods can be achieved through continuation techniques, which provide better initial guesses (for example by coarse to fine mesh sequencing), in order to reduce the initial phase of poor convergence of these schemes. However, one of the drawbacks of these methods is that they still require powerful preconditioners in order to be effective, and the best known preconditioners are generally matrix based (such as ILU), which entail storage requirements similar to those of the first order linearization. A compromise, with obvious limitations, is to use a weaker but low-storage preconditioner, such as block-diagonal with a ***Newton-Krylov Method***[144]. For upwind discretization based on reconstruction techniques, higher-order Jacobian vector products can be formed by multiplying the first-order Jacobian with a higher-order reconstructed vector. Although the method requires the storage of the first-order Jacobian, it is exact and, unlike finite-difference techniques, can also be applied directly to the solution of the adjoin equations, which is required in particular formulations of the design optimization problem. Additional approximations to the exact Jacobian can be employed to further reduce the memory requirements of implicit schemes. For example, one may choose to group sub-regions of the mesh together and only retain the terms of the Jacobian that pertain to coupling within each region[145].

These regions may be determined by a variety of methods and may exhibit varying degrees of overlap. These approximations may be applied to the Jacobian itself but are most often applied to the matrix-based preconditioners operating on the true Jacobian. Alternatively, the regions may be reduced to lines through the mesh joining neighboring vertices, as in the method of line lets, which attempts to approximate the ***Alternate Direction Implicit (ADI)*** scheme of structured grids[146]. On sequential machines, the regions or lines can be processed individually, thus reducing the maximum

---

[143] Nielsen EJ, Anderson WK, Walters RW, Keyes DE. ,"*Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code*", Proc. AIAA CFD Conf., 12th, San Diego. AIAA Pap. 95-1733-CP, 1995.

[144] Shakib F, Hughes TJR, Johan Z., "*A multi element group preconditioned GMRES algorithm for non-symmetric problems arising in finite element analysis*", Computer. *Methods Appl. Mech. Eng.* 87:415–56, 1989.

[145] Chan TF, Mathew TP.," Domain decomposition algorithms", *Acta Numer.* pp. 61–143.

[146] Hassan O, Morgan K, Peraire J., "*An implicit finite element method for high speed flows*", AIAA Pap. 90-0402.

memory requirements to that of the largest region or line (although on parallel machines no savings are realized if each region is assigned to an individual processor). The numerical coupling between regions or lines is lost in these approaches, a loss that in turn degrades overall convergence, as the number of regions increases. However, this degradation can be minimized by judicious choices of the regions based on the character of the problem, a technique that has not yet been fully exploited. If these regions are reduced to individual grid points (i.e. all implicit coupling between grid points is discarded), only the diagonal block matrices of the original Jacobian are retained, and the point-implicit method is obtained [147-148]. These terms represent the coupling between the various fluid dynamic equations at a grid point. While point-implicit methods can offer increases in efficiency over regular explicit schemes, they are also plagued by slow convergence for fine grids. Point-implicit methods are sometimes viewed as preconditioning techniques for explicit schemes, where the point-implicit matrix is the preconditioner.

3.4.2.2 Reflections on the Evolution of Implicit Navier-Stokes Algorithms[149]

The motivation for implicit viscous flow solvers has been the need to solve complex flows requiring highly nonuniform grids and with multiple time and length scales. Such problems can present severe algorithmic challenges when resolving disparate local length scales introduced by geometry, very thin shear layers, and other localized flow structures. In addition, the differing time scales of convection, diffusion, sound propagation, and chemical reaction can result in equation stiffness, a term used for ordinary differential equations whose system matrices have a wide range of eigenvalues. In both instances, the enhanced stability properties available from implicit schemes can help by allowing larger time steps, within an objective of either time accuracy or convergence to a steady solution. We will briefly comment here on the development of our own noniterative time-linearized, block-coupled, ***Alternating-Direction Implicit (ADI) scheme*** reported during 1973-1977, and then give some personal reflections on the subsequent evolution of these ideas and some of the progress achieved through related ideas and innovations introduced by others.

3.5 Listing to CFD Numerical Methods

This section covering the outline of CFD and brought to you by CFD-Wiki and mentioned here as a recap. The user should refer to this page for further reference. Be aware that this is lengthy and general list and not all the items are relevant to the problem in hand. Top level is displayed as Red, next Green, black, Brown, and so on.

- Basic Aspects of Governing Equations as related to CFD
  - Classification of Governing Equations (GE)
    - Integral form of GE
    - Differential form of GE
    - Generic Scalar Transport Equation
  - Mathematical behavior of Partial Differential Equations
    - Classification of physical behavior
      - Equilibrium Problems
      - Marching Problems
    - Mathematical Classification of flows
      - Hyperbolic Flows
      - Parabolic Flows

---

[147] Hassan O, Morgan K, Peraire J., "*An implicit finite element method for high speed flows*", AIAA Pap. 90-0402.
[148] Thareja RR, Stewart JR, Hassan O, Morgan K, Peraire J., "*A point implicit unstructured grid solver for the Euler and Navier-Stokes equations*.", AIAA Pap. 88-0036, 1988.
[149] W. R. Briley and H. McDonald, "*Reflections on the Evolution of Implicit Navier-Stokes Algorithms*", UTC-CECS-SimCenter-2008-04, September 2008.

        - Elliptic Flows
        - Mixed Flow
    - Stability Analysis
        - Fourier Stability Analysis
        - von Neuman Stability Analysis
        - Courant–Friedrichs–Lewy (CFL) condition
- Properties of Numerical Solution Methods
    - Consistency
    - Stability
    - Conservation
    - Boundedness
    - Reliability
    - Accuracy
- Solution of Euler Equation
    - Method of Characteristics
    - Shock Capturing Techniques
        - Explicit Mac Cormack Method
        - Flux Splitting (Stegar-Warming)
        - Total Variation Diminishing (TVD) scheme
        - Artificial Dissipation
    - The SCM
    - Beam & Warming Approximate Factorization scheme
- Solution of Poisson's Equation
    - Type-Dependent Differencing
- Solution of Navier-Stokes Equations
    - Explicit schemes
    - Mac Cormack Predictor-Corrector scheme
    - Beam & Warming (AF)
    - Vorticity-Stream Function
    - Velocity-Pressure Coupling
        - Rhie-Chow Interpolation
    - Fully Coupled Methods - FC
        - Density Methods
    - De-Coupled Methods - DC
        - Pressure Methods
            - SIMPLE
            - SIMPLEC - SIMPLE Consistent
            - SIMPLEM – SIMPLE Modified
            - SIMPLEX
            - SIMPLEST - SIMPLE Shortened
            - SIMPLER - SIMPLE Revised
            - PISO - Pressure Implicit with Split Operator
            - PRIME - Pressure Implicit Momentum Explicit
            - MSIMPLEC, MPISO , SIMPLESSEC , SIMPLESSE
            - CPC - Compressible Pressure Correction algorithm
        - MCBA - Mass Conservation Based Algorithms
        - GCBA - Geometric Conservation Based Algorithms

- IPSA - Inter-Phase Slip Algorithm
- CPI - Consistent Physical Interpolation
- MWIM - Momentum Weighted Interpolation Method
- PWIM - Pressure Weighted Interpolation Method
- ACM - Artificial Compressibility Method
- CIP - Cubic Interpolating Polynomial Method
- Discrete Operator Splitting

- Solution of Boltzmann Equation CFD Related algorithms
  - Wall Distance Calculations
    - Moore's k-tree algorithm
    - Transport equation based wall distance calculation
- Geometrical Calculations
  - Area calculations
  - Volume calculations
  - Calculation on non-orthogonal curvilinear structured grids, finite-volume method
- General Discretization
  - Domain (Field) Discretization
    - Grid or Mesh Generation
      - Structured
        - Algebraic
        - PDE
        - Adaptive Meshes
      - Unstructured
        - Advancing Front
        - Delaney Triangulation
        - Octree Decomposition
        - Unstructured Hexahedral Meshes
        - Hybrid Meshes
        - Overset Meshes
        - Cartesian Meshes
        - Polyhedral Meshes
        - Adaptive Meshes
  - Governing Equations Discretization
    - Finite Difference
    - Finite Element
    - Spectral Volume
    - Discontinuous Galerkin
    - Finite Volume
      - Generic scalar transport equation
      - Source term linearization
      - Gradient computation
      - Special Discretization
        - Discretization of the diffusion term
        - Discretization of the convection term
        - Discretization of the transient term
      - Time Discretization
        - Euler Method

        - Crank-Nicolson Method
        - Predictor-Corrector Methods
        - Runge Kutta Methods
        - Adams Bashforth Method
        - Adams Moulton Method
        - Implicit second order Method

- Linear Systems of Equations
    - Direct Methods
        - Gaussian Elimination
        - LU Decomposition
        - Tri-Diagonal Matrix Algorithm - (TDMA -Thomas Algorithm)
    - Iterative Methods
        - Gauss-Seidel Method
        - Jacobi Method
        - Successive 0ver-Relaxation Method - SOR
        - Stone's Method
        - Alternating Direction Implicit (ADI) Method
        - Conjugate Gradient Methods
            - Conjugate Gradient Method of Golub and van Loan
            - Bi-Conjugate Gradient Method
            - Bi-Conjugate Gradient Stabilized Method
    - Matrix Factorization and Preconditioning
        - Incomplete LU Factorization - ILU
        - Incomplete Cholesky Factorization
    - Multigrid Methods
        - Geometric Multigrid – Full Approximation Storage (FAS)
        - Algebraic Multigrid - AMG
- Efficiency and Stability
    - Limiters
        - Flux Limiters
        - Gradient Limiters