DLR-IB-AS-BS-2016-333

Investigation of Relaxation parameters and residual based truncation criteria for Reynolds averaged Navier-Stokes equations

Interner Bericht

Autor
Manickam Somasundaram

Deutsches Zentrum
DLR für Luft- und Raumfahrt

Bericht des Instituts für Aerodynamik und Strömungstechnik
Report of the Institute of Aerodynamics and Flow Technology

DLR-IB-AS-BS-2016-333

Investigation of Relaxation parameters and residual based truncation criteria for Reynolds averaged Navier Stokes equations

Manickam Somasundaram

Herausgeber:
Deutsches Zentrum für Luft- und Raumfahrt e.V.
Institut für Aerodynamik und Strömungstechnik
Lilienthalplatz 7, Germany, 38108 Braunschweig

ISSN 1614-7790

Stufe der Zugänglichkeit: 1
Braunschweig, im Dezember 2016

Institutsdirektor:
Prof. Dr.-Ing. habil. C.-C. Rossow

Verfasser:
Manickam Somasundaram

Abteilung: Center of Computer Applications in Aerospace Science and Engineering
Abteilungsleiter:
Prof. Dr.-Ing. N. Kroll

Der Bericht enthält:
57 Seiten
37 Bilder
7 Tabellen
27 Literaturstellen

Abstract

To solve the RANS equation and the transport equation based on one equation Spalart Allmaras model, an agglomerated non-linear multigrid scheme with implicit smoothing, based on first order approximation of the derivative of the residual function, exists. The implicit smoothing leads to a linear system of equation that needs to be solved for every stage of the smoothing algorithm. To improve this solution method grid anisotropy is treated with the Gauss-Seidel iteration in such a way that lines with strong coupling in linear system are resolved by tridiagonal systems constructed along the direction of strong coupling. The existing methodology of solving this linear system with a line symmetric Gauss-Seidel method restricts finer meshes to reach higher CFL numbers. Therefore relaxation parameter is incorporated into the existing line symmetric Gauss-Seidel methods to investigate the change in the solution methodology.
The current practice is to set the number of Gauss-Siedel sweeps to a certain number and this may lead to over-solving. Thus the extent to which the inner linear equations that needs to be solved can be truncated after the linear residual is reduced by a certain order of magnitude. In this regard too, numerical experiments demonstrate that choosing a low GS sweeps may lead to delay in converging whereas a high GS sweep value may lead to over solving thereby making it computationally expensive. Thus a truncation criteria based on a linear residual is implemented and the change is investigated. Eigenvalue analysis is done after implementation of the linear based truncation criteria to compare the asymptotic convergence of the solution method.

Investigation of Relaxation parameters and residual based truncation criteria for Reynolds averaged Navier Stokes equations

Master of Science Thesis

For obtaining the degree of Master of Science in at TU Braunschweig

Manickam Somasundaram. Advisor:Stefan Langer

November 3, 2016

Faculty of Computational Science in Engineering · TU Braunschweig

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The incompressible and compressible Navier-Stokes equations form the heart of modeling any fluid flow. Ways to solve these equations have been a research topic for many years now. $Navier - Stokes\ existence\ and\ smoothness$ is among the unsolved ones in the set of seven most important problems in Mathematics called Millennium Prize problems.This is partly because its solution often includes turbulence, which parallely remains one of the unsolved problems in physics.
For industrial applications arising in aircraft industry and research, these equations are optimally modeled. Numerical methods are used to approximate the solution. Two ways are possible, either we iterate in time or we try to find a steady state solution. Throughout this thesis we consider the compressible Navier-Stokes equation. This branch of fluid dynamics which makes use of mathematical models to simulate fluid flows is called Computational fluid dynamics(CFD). With recent advancements in high performance computing and algorithm development, computational fluid dynamics is becoming widely accepted design and analysis tool. Numerical simulations in CFD have matured to a point where they are used as practical engineering tools in industrial design process. Experimental techniques and numerical simulations complement each other as experimental data helps us validate the solution methodology or a solver and CFD on the other hand helps to narrow down on optimal configurations for experimental setups. The design of high-speed, low-noise, and safe aircraft is the major challenge of aircraft industry for the following years to come. Numerical simulations that are comfortable simulating flows with high Reynolds number is going to play a vital role in the design process.
With respect to turbulence modeling, solvers can be classified into Direct numerical simulation(DNS), Large -Eddy simulations(LES) and Reynolds averaged Navier-Stokes solvers(RANS). Among the three, RANS are generally known to be computationally less expensive compared to the other two as it involves modeling of the turbulence rather than computing it. In RANS there are Reynolds stress models(RSM) and Eddy Viscosity models (EVM). RSM follows a second order closure and EVM follows first order closure. Eddy viscosity models are based on the Boussinesq hypothesis, which assumes that the turbulent shear stress depends linearly on the mean rate of strain and the proportionality constant is the eddy viscosity. Each of the

Figure 1-1: Eurofighter Typhoon at ILA2016

turbulence models has its own strengths and weaknesses, so it's always important to investigate which model works for which type of flows. In this thesis we restrict ourselves to find the steady state solution for the RANS equation in combination with a one equation Eddy viscosity model.
When it comes to computing viscous flows with high CFL numbers, a drastic loss of efficiency and reliability for compressible RANS equations can be seen. As anyone conversant with creating CFD meshes for high Reynolds number flows will know that, cells in the boundary layer have high aspect ratio and are stretched in the direction of flow to capture the steep gradients occurring here. One such cell is illustrated in Figure.1-2. These cells result in a stiffness of the discrete system of governing equations. This stiffness leads to a difficulty in removing some error modes when computing solutions. A detailed analysis of stiffness in discrete systems of equation and difficulty in solving the equations due to it can be seen in [1].

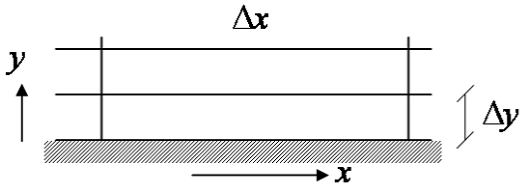


Figure 1-2: Stretched cell

Multigrid methods are well known convergence acceleration techniques. It is the basic idea of multigrid methods that low frequency errors on a fine mesh become high-frequency errors on a coarse mesh. In the framework of an unstructured code agglomeration techniques can be used to create coarser meshes. Now an efficient method to smooth the errors is required. A straightforward introduction of these methods couldn't ensure improvement in efficiency of

high Reynolds number viscous flows when explicit Runge-Kutta smoothers were used. The explicit smoothers figured out to be not suitable for stiff-problems as a consequence, often either a slowdown in convergence history can be observed or convergence cannot be obtained. The number of iterations to converge typically increases significantly. This may be practical for approximating steady-state solution but the order of magnitude increases further as we want to solve unsteady problems where a steady state problem is solved in each time step. Thus it is necessary to improve the solution procedure by implementing adequately more efficient time integration techniques.
It can be seen from Rossow, Swanson and Turkel [2, 3, 4] that when an explicit multistage smoother is supplemented by an implicit one there is a significant improvement in both reliability and efficiency. This was demonstrated for $2D$ structured flows and in [3] the scheme was successfully applied for a $3D$ wing. In [5] implicit Runge-Kutta methods with point relaxation is implemented and significant improvement in convergence rate over explicit methods to approximate steady state solution of inviscid flows can be seen. In [6] an agglomerated FAS multigrid for unstructured meshes to approximate steady state solution of the RANS equation is presented. The smoother in this multigrid method is derived by a Diagonally-Implicit-Runge-Kutta (DIRK) method and is based on the first order approximation of the Jacobian. Here the implicit Runge-Kutta method is interpreted as a preconditioned explicit Runge-Kutta method. The hierarchy of preconditioning techniques for the multistage Runge-Kutta method can be seen in [7]. It can be used to derive other solution techniques like the ones mentioned in [2, 4], line relaxation methods [8, 9, 10], point relaxed techniques [11, 5]. Main differences between these methods are approximation of the Jacobian and the method which is used to solve the arising linear system of equation.
One of the focus in this thesis is to explore and investigate efficient ways to improve the solution methods to solve these linear system of equations. Line relaxation [10] techniques exploit the strong coupling between anisotropic cells. The direction of strong coupling are integrated into the linear solution methods. This allows using of much larger CFL numbers in comparison with pure point and line relaxation techniques. This information of directional coupling is generated using line-search algorithm [10]. In [9] they used the technique only on the finest grid level within the multigrid algorithm and they considered it to be a line only if number of points with strong coupling was more than a certain number, where as in [10] any number of point more than 1 is considered to be a line and the method is applied uniformly on all multigrid levels. In [6], for solving the linear solution method arising from every Runge-Kutta stage of the smoother algorithm, (Block) Gauss-Seidel methods are used. The number of Gauss-Seidel sweeps an inner linear loop needs is also an user defined parameter, which is usually taken as 3 or 5 as seen in [6]. The inner loop is just used to give an optimum update to the outer non-linear loop and thus may lead to over-solving.
Main challenges that are omnipotent to deployment of any CFD tool in industry or in research has been time consumption and complexity of solver parameters. Ideally a CFD code should be robust enough to simulate wide range of fluid flows without the user selecting large number of solver parameters. A solver can be made more robust by making it comfortable to run at higher CFL numbers and also reducing the solver parameters that the user needs to select to get a meaningful and converged result. Our focus is to make our solution method more robust by improving methods in such a way that it is comfortable with higher CFL numbers and eliminate dependent variables, which the user needs to select, by making the program adaptable.
The baseline code, in which the developments reported in this thesis are made, is the develop-

ers code for the TAU code. Which are based on compressible Reynolds averaged Navier-Stokes equations. TAU code is developed by the DLR, German Aerospace Center Institute of Aerodynamics and Flow Technology. It is well-established and widely used general-purpose tool for a broad range of aerodynamic problems.
In this thesis we mainly focus on the implicit smoother in the multistage Runge-Kutta method. For every stage of the Runge-Kutta method a linear system of equation needs to be solved. This is elaborated in Chapter 4. From the results presented in [6], it can be seen that for DPW5 meshes the maximum CFL number that can be reached is lowered as we move on to finer meshes. This points out a lack of robustness and reliability at higher CFL numbers and there is room to explore methods to overcome this. On the other hand a relaxation parameter can be implemented on the Gauss-Seidel methods, which are used to solve the linear set of equations. It is one of the goal of this work to implement a relaxation parameter into this family of Gauss-Seidel methods and investigate in which way it changes the efficiency and robustness of the whole solution methodology.
The number of Gauss-Seidel sweeps required for the inner-linear loop is also a parameter that needs to be selected. This inner-linear loop is an update given to the outer-non linear loop. So an inner truncation criteria can eliminate this parameter as the loop would be truncated after the linear-residual is truncated by a certain order of magnitude. As an attempt to eliminate a parameter, number of Gauss-Seidel steps needed for the inner-linear loop, in this thesis we implement an inner truncation criteria. The aim is to investigate if this criteria changes the outer non-linear residual and in extension the whole steady state solution. Turbulence is modeled using the Spalart-Allmaras one equation model [12, 13]. After discretizing we get a system of ordinary differential equation, which can be split into two: one for mean flow and other for turbulence. The system of equations are sequentially solved in a weakly coupled manner ,i.e. while solving mean flow equations the turbulent variable is assumed to be constant and while solving the turbulent flow equation the conservative variables are assumed to be constant.
The development of these solution methodologies is often based on several heuristics and a way to judge the power and the deficiencies of methods is missing. The work presented in [14] presents a tool based on Arnoldi's method to investigate approximations to the spectrum of the linearized equations. In this work we make use of this to show how the changes made to the baseline code suggested in this work have altered the approximate eigenvalue spectrum. This eigenvalue analysis shall be done to see the changes on the method after implementing the linear-truncation criteria. The thesis is organized as follows: In Chapter 2, we present the governing equation written in Integral form along with the one-equation Spalart-Allmaras model. The discretization strategy for both mean and turbulent flow equation are dealt with in Chapter 3. In Chapter 4, we present the FAS multigrid method applied to unstructured meshes and agglomeration used in the code. The Diagonally Implicit Runge-Kutta method based smoother is also explained here. Towards the end of this chapter we get an implicit Runge-Kutta method, which can be interpreted as a preconditioned explicit Runge-Kutta method. This preconditioning gives rise to a linear system of equations which is given as an update to the outer-non-linear loop. A linear system needs to be solved for every Runge-Kutta stage and therefore efficient and computationally less expensive methods need to be implemented. So in the next chapter, the current solution methods available in the baseline code is discussed and the relaxation parameter that is implemented is explained. In Chapter 6, the suggested methods are applied to several 2D as well as 3D benchmark cases and the results including convergence histories are presented. The truncation criteria implemented is

explained and results are given. Eigenvalue spectrum to show the effects of truncation criteria is given too. How the number of linear Gauss-Seidel sweeps, required for linear residual to drop by one order of magnitude, varies with outer non-linear iteration is presented. At the end all results are considered and Conclusion is given in Chapter 7.

Chapter 2

Governing equation

Navier Stokes equations form the fundamental basis to describe the motion of viscous fluid. It is derived from the conservation laws of mass, momentum and energy in computational fluid dynamics problems. The integral form of these equations are derived based on a finite control volume, which is fixed in space. This form of the equations which is called the conservation form. If the equations were based on a finite volume that was moving with the flow it would be called non conservation form [15]. All the three conservation laws can be collected into one system of equation so as to get a good overview of all terms. This system of equation is considered for an open domain $\Omega \subset \mathbb{R}^3$ and the three dimensional Reynolds averaged Navier Stokes equation in conservative variables $\mathbf{W} := (\rho, \rho u_1, \rho u_2, \rho u_3, \rho E)$ in it's integral form is written as

$$\frac{d}{dt}\int_{\Omega} \mathbf{W} d\mathbf{x} + \int_{\partial\Omega} (\mathbf{f}_c \cdot \mathbf{n} - \mathbf{f}_v \cdot \mathbf{n}) dS = 0 \tag{2-1}$$

where Ω represents the control volume, $\partial\Omega$ is its surface and dS is a surface element of Ω. The vector of convective fluxes($\mathbf{f}_c \cdot \mathbf{n}$) is obtained as

$$\mathbf{f}_c \cdot \mathbf{n} = \begin{bmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ \rho v V + n_z p \\ \rho H V \end{bmatrix} \tag{2-2}$$

where V is the *contravariant velocity* V, which is the velocity normal to the surface element dS. It is the scalar product of the velocity vector,$\mathbf{u}$ =(u,v,w), and the unit normal vector, $\mathbf{n}$ =(n_x,n_y,n_z) , i.e.,

$$V \equiv \mathbf{u} \cdot \mathbf{n} = n_x u + n_y v + n_z w \tag{2-3}$$

The total enthalpy H is given by the formula:

$$H = E + \frac{p}{\rho} \tag{2-4}$$

where pressure is given by the equation of state

$$p = (\gamma - 1)\rho\left(E - \frac{\|\mathbf{u}\|_2^2}{2}\right) \tag{2-5}$$

E is the specific total energy, and γ is the gas dependent ratio of specific heats, which is taken as 1.4 for air. Now, moving on to the vector of viscous fluxes.

$$\mathbf{f}_v \cdot \mathbf{n} = \begin{bmatrix} 0 \\ n_x\tau_{xx} + n_y\tau_{xy} + n_z\tau_{xz} \\ n_x\tau_{yx} + n_y\tau_{yy} + n_z\tau_{yz} \\ n_x\tau_{zx} + n_y\tau_{zy} + n_z\tau_{zz} \\ n_x\Theta_x + n_y\Theta_y + n_z\Theta_z \end{bmatrix} \tag{2-6}$$

where

$$\begin{aligned} \Theta_x &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + k_{eff}\frac{\partial T}{\partial x} \\ \Theta_y &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + k_{eff}\frac{\partial T}{\partial y} \\ \Theta_z &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + k_{eff}\frac{\partial T}{\partial z} \end{aligned} \tag{2-7}$$

The viscous stresses in the above equation (τ_{ij}) originate from the friction between fluid and the surface of an element. It is described by a tensor (τ) given by

$$\tau = \begin{bmatrix} \tau_{xx} & \tau_{xx} & \tau_{xx} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{bmatrix} \tag{2-8}$$

The normal stresses on the diagonal of the matrix in Equation(2-8) try to displace the faces of the element in the three mutually perpendicular direction whereas the shear stresses, which are the non diagonal elements of the matrix, try to shear the element. Since they depend on how a fluid deforms, the viscous stresses depend on the dynamical properties of the fluid medium. The type of the fluids like air or water, Sir Issac Newton stated that the shear stress is proportional to the velocity gradient and these mediums are called as Newtonian fluids. Thus we have the shear stresses

$$\begin{aligned} \tau_{xy} = \tau_{yx} &= \mu_{eff}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial z}\right) \\ \tau_{xz} = \tau_{zx} &= \mu_{eff}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\ \tau_{yz} = \tau_{zy} &= \mu_{eff}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \end{aligned} \tag{2-9}$$

and the normal stress are got after incorporating the hypothesis introduced by Stokes that $\lambda_{eff} = -\frac{2}{3}\mu_{eff}$. λ is called the second viscosity coefficient and is usually eliminated with the help of this hypothesis by Stokes.

$$\begin{aligned} \tau_{xx} &= 2\mu_{eff}\left(\frac{\partial u}{\partial x} - \frac{1}{3}\mathrm{div}\mathbf{u}\right) \\ \tau_{yy} &= 2\mu_{eff}\left(\frac{\partial v}{\partial y} - \frac{1}{3}\mathrm{div}\mathbf{u}\right) \\ \tau_{zz} &= 2\mu_{eff}\left(\frac{\partial w}{\partial z} - \frac{1}{3}\mathrm{div}\mathbf{u}\right) \end{aligned} \tag{2-10}$$

μ_{eff}, the effective dynamic viscosity is the sum of laminar viscosity and eddy viscosity,$\mu_{eff} = \mu_l + \mu_t$. The laminar viscosity is described by Sutherland's Law

$$\mu_l := \mu_{ref} \left(\frac{T}{T_{ref}} \right)^{3/2} \frac{T_{ref} + S}{T + S} \tag{2-11}$$

where S is the Sutherland's Constant for Air is $S = 110.4K$ and μ_{ref} is reference viscosity at T_{ref}. Similarly the effective conductivity in is evaluated as $k_{eff} = k_l + k_t$. We get k_l from

$$\begin{aligned} k_l &= \frac{c_p \mu_l}{Pr_l} \\ c_p &= \mathfrak{Re} \frac{\gamma}{\gamma - 1} \end{aligned} \tag{2-12}$$

where $\mathfrak{Re} = 8.314\ J/\ mol/\ K$ is universal gas constant and Pr_l, the Prandlt number is given as $Pr_l = 0.72$. The eddy viscosity μ_t and k_t shall be defined in next part where the turbulent equations are explained.

The one equation turbulence model introduced by Spalart and Allmaras([12]) is used to describe turbulent flow effects. The transport equation for an kinematic viscosity variable ($\tilde{\nu}$) that needs to be solved is written as

$$\frac{d}{dt} \int_\Omega \tilde{\nu} dx + \int_{\partial\Omega} (\mathbf{f}_{c,turb} \cdot \mathbf{n} - \mathbf{f}_{v,turb} \cdot \mathbf{n}) dS = \int_\Omega Q_T d\Omega \tag{2-13}$$

where Ω, $\partial\Omega$, and dS is same as in Equation(2-1). The convective flux is $\mathbf{f}_{c,turb} \cdot \mathbf{n} = \tilde{\nu} V$, with V being the contra variant velocity as defined in Equation(2-3). And viscous flux is defined as

$$\mathbf{f}_{v,turb} \cdot \mathbf{n} = n_x \left(\frac{\nu_L + \tilde{\nu}}{\rho} \right) \frac{\partial \tilde{\nu}}{\partial x} + n_y \left(\frac{\nu_L + \tilde{\nu}}{\rho} \right) \frac{\partial \tilde{\nu}}{\partial y} + n_z \left(\frac{\nu_L + \tilde{\nu}}{\rho} \right) \frac{\partial \tilde{\nu}}{\partial z} \tag{2-14}$$

Viscous flux is actually a inner dot product of unit normal vector and normal viscous stresses. Finally the source term is given by $Q_T = Production\ term + Diffusion\ Term + Wall\ destruction\ Term$

$$Q_T = C_{b1}(1 - f_{t2}) \tilde{S} \tilde{\nu} + \frac{C_{b2}}{\sigma} \left[\left(\frac{\partial \tilde{\nu}}{\partial x} \right)^2 + \left(\frac{\partial \tilde{\nu}}{\partial y} \right)^2 + \left(\frac{\partial \tilde{\nu}}{\partial z} \right)^2 \right] - \left[C_w f_w - \frac{C_{b1}}{k_2} f_{t2} \right] \left(\frac{\tilde{\nu}}{d} \right)^2 \tag{2-15}$$

where

$$f_{t2} := c_{t3} exp(-c_{t4} \chi^2), f_w := g \left(\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6} \right)^{1/6} \tag{2-16}$$

The above equations are valid only when $\tilde{\nu}$ is greater than 0. When $\tilde{\nu}$ is less than 0, the below equations are valid. Viscous flux is

$$\mathbf{f}_{v,turb} \cdot \mathbf{n} = n_x \left(\frac{\nu_L + f_n \tilde{\nu}}{\rho} \right) \frac{\partial \tilde{\nu}}{\partial x} + n_y \left(\frac{\nu_L + f_n \tilde{\nu}}{\rho} \right) \frac{\partial \tilde{\nu}}{\partial y} + n_z \left(\frac{\nu_L + f_n \tilde{\nu}}{\rho} \right) \frac{\partial \tilde{\nu}}{\partial z} \tag{2-17}$$

A new term f_n is there in this equation, which is defined as

$$f_n := \frac{c_{n1} + \chi^3}{c_{n1} - \chi^3}, \qquad c_{n1} = 16 \tag{2-18}$$

And the source term is

$$Q_T = C_{b1}(1 - c_{t3})S\tilde{\nu} + \frac{C_{b2}}{\sigma}\left[\left(\frac{\partial\tilde{\nu}}{\partial x}\right)^2 + \left(\frac{\partial\tilde{\nu}}{\partial y}\right)^2 + \left(\frac{\partial\tilde{\nu}}{\partial z}\right)^2\right] + C_w\left(\frac{\tilde{\nu}}{d}\right)^2. \tag{2-19}$$

The eddy viscosity μ_t is calculated by

$$\mu_t = \begin{cases} \rho\tilde{\nu}f_{v1} & \tilde{\nu} \geq 0, \\ 0 & \tilde{\nu} < 0 \end{cases} \tag{2-20}$$

$$\kappa_t := \frac{c_p\mu_t}{Pr_t}, \qquad f_{v1} := \frac{\chi^3}{\chi^3 + c_{v1}^3}, \qquad \chi = \frac{\tilde{\nu}}{\nu_l}, \qquad \nu_l := \frac{\mu_l}{\rho} \tag{2-21}$$

All terms and constants that feature in the turbulent equation above are consolidated and defined below

$$S := \|curl(\mathbf{u})\|_2, \qquad \bar{S} := \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}$$

and

$$\tilde{S} := \begin{cases} S + \bar{S} & , \bar{S} \geq -c_{v2}S, \\ S + \frac{S(c_{v2}^2 S + c_{v3}\bar{S})}{(c_{v3} - 2c_{v2})S - \bar{S}} & , \bar{S} < -c_{v2}S, \end{cases}$$

$$g := r + c_{w2}r(r^5 - 1), \qquad r := min\left\{\frac{\tilde{\nu}}{\kappa^2 d^2 \tilde{S}}, 10\right\}, \qquad f_{v2} := 1 - \frac{\chi}{1 + \chi f_{v1}}$$

$$c_{b1} := 0.1355, \qquad c_{b2} := 0.622, \qquad \sigma = \frac{2}{3}, \qquad \kappa := 0.41$$

$$c_w := \frac{c_{b1}}{\kappa^2} + \frac{1 + c_{b2}}{\sigma}, \qquad c_{w2} := 0.3, \qquad c_{w3} := 2$$

$$c_{t2} := 1.2, \qquad c_{t4} := 0.5, \qquad c_{v1} := 7.1, \qquad c_{v2} := 0.7, \qquad c_{v3} := 0.9$$

Non dimensionalization of the governing equation is usually done by means of reference values for length $\hat{x}_{ref}$, density $\hat{\rho}_{ref}$ and velocity $\hat{v}_{ref}$. These are got from the underlying physical flow field. The remaining reference values are got from equations: $\hat{p}_{ref} = \hat{\rho}_{ref} \cdot \hat{v}_{ref}^2$ and $\hat{E}_{ref} = \hat{p}_{ref}/\hat{\rho}_{ref}$. Introducing these values into the governing equation yields the non-dimensional formulations. Generally speaking, for any quantity $\hat{\varphi}$, introducing the expression $\hat{\varphi} = \varphi \cdot \hat{\varphi}_{ref}$ into the governing equation gives a non-dimensional formulation.

Chapter 3

Discretization

Space discretization is based on a node centered, finite volume space discretization. In a pre-processing stage the computational mesh, also called the dual grid, is made from the primary grid. The dual grid is formed by creating control volumes in such a way that we have unknowns at the vertices of the primary grid. Shown in Figure(3-1) is one dual mesh cell(the polygon $LMNOPQ$) constructed from primary mesh. The darkened area is this control volume with the unknowns at X, which is a vertex of the primary grid. Similarly A, B, C, D, E and F are vertices with unknowns of the control volumes, which are the neighbors to this control volume shown here.

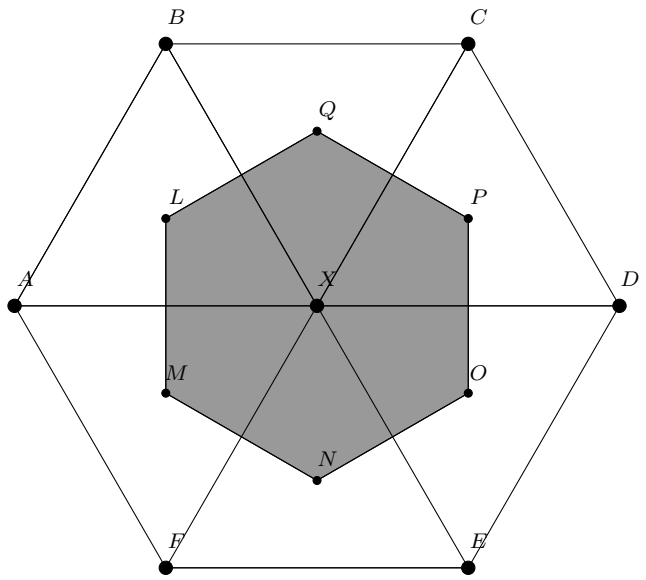


Figure 3-1: Construction of dual grid from primary grid

The inviscid part of the (2-1) is discretized by central difference scheme with an added matrix valued artificial viscosity([16],[17]). For the construction and stability properties of central

difference schemes with an added artificial viscosity we refer to ([18],[19]). The inviscid part $(\mathbf{f}_c \cdot \mathbf{n})$ for an inner volume Ω_i is approximated by

$$\int_{\partial\Omega_i} \mathbf{f}_c \cdot \mathbf{n} ds \approx \sum_{j \epsilon N(i)} \frac{1}{2} \left((\mathbf{f}_c \cdot \mathbf{n}_{ij}) (\mathbf{W}_i) + ((\mathbf{f}_c \cdot \mathbf{n}_{ij}) (\mathbf{W}_j)) - \mathbf{D}_{ij}(\mathbf{W}) \right) \tag{3-1}$$

$\mathbf{D}_{ij}$ is

$$\mathbf{D}_{ij}(\mathbf{W}) = \frac{1}{2} \left| \mathbf{A}_{ij}^{Roe} \right| \left\{ \Psi_{ij}(\mathbf{W}_j - \mathbf{W}_i) - \xi s_{ij}(\mathbf{W})(1 - \Psi_{ij})(\mathbf{L}_j(\mathbf{W}) - \mathbf{L}_i(\mathbf{W})) \right\} \tag{3-2}$$

where

$$\mathbf{L}_j(\mathbf{W}) := \sum_{j \in N(i)} (\mathbf{W}_j - \mathbf{W}_i) \tag{3-3}$$

$$\mathbf{A}_{ij}^{Roe} := \frac{\partial(\mathbf{f}_c \cdot \mathbf{n}_{ij})}{\partial \mathbf{W}} [\mathbf{W}_{Roe}] \tag{3-4}$$

$$\psi_{ij} := min \left\{ \varepsilon_\psi \psi_{ij}^*, 1 \right\}, \qquad \varepsilon_\psi = 8 \tag{3-5}$$

$$\psi_{ij}^* := \frac{(p_j - p_i)^2}{(p_j + p_i)^2} \tag{3-6}$$

$N(i)$ is used to describe the direct neighbors of a point i. $\mathbf{L}_i$ represents an undivided Laplacian operator, $\mathbf{A}_{ij}^{Roe}$ is the derivative of the convective flux evaluated on the face ij using Roe-averaged variables. In order to handle shocks, a pressure switch(ψ) is added in the dissipative part. In the neighborhood of the shock this variable helps to reduce to first order, otherwise a higher order on shocks can lead to oscillations. The term $s_{ij}(\mathbf{W})$ is incorporated to deal with highly stretched cells and is called cell stretching coefficient. This coefficient is based on the absolute value of the largest convective eigenvalue. For highly stretched cells there is significant increase in dissipation in the direction of cell stretching due to the incorporation of cell stretching coefficient.
There are different types of Upwind spatial discretization that can be found in literature. Which can be roughly classified into $flux - vector\ splitting$, $flux\ difference\ splitting$, $total\ variation\ diminishing(TVD)$, and $fluctuation\ splitting\ schemes$. $Flux\ difference\ splitting$ is based on the solution of the locally one-dimensional Euler equation for discontinuous states at the interface. This leads to solving Riemann problem at the interfaces and approximate Riemann solvers were developed. TVD scheme's principle condition is at preventing generation of new extremes. Thus this methodology helps in resolving a shock wave without any spurious oscillations of the solution. For detailed discussion of all schemes one may refer to ([15])

A first order Roe scheme is used to discretize the convective part of the Spalart-Allmaras Turbulence equation.

$$\int_{\partial\Omega_i} \mathbf{f}_{c,turb} \cdot \mathbf{n} ds \approx \sum_{j \epsilon N(i)} \frac{1}{2} \left((\mathbf{f}_{c,turb} \cdot \mathbf{n}_{ij}) (\mathbf{W}_i) + ((\mathbf{f}_{c,turb} \cdot \mathbf{n}_{ij}) (\mathbf{W}_j)) - \frac{1}{2} |V| (\tilde{\nu}_j - \tilde{\nu}_i) \right. \tag{3-7}$$

The viscous stresses in the viscous part of the RANS equations as well as the turbulence equation have derivatives. These derivatives may either be computed by Green-Gauss method

or by a so-called thin shear layer approximation method. The discretization using both methods for a general variable w is given below. Firstly, for the green Gauss method

$$\left(\frac{\partial w_i}{\partial x_k}\right)^{GG} := \frac{1}{vol(\Omega_i)} \sum_{j \epsilon N(i)} svol(\Omega_i) \frac{n_{k,ij}}{2}(w_i + w_j), k = 1, 2, 3 \tag{3-8}$$

where $svol(\Omega_i)$ is the surface are of the control volume. Thin shear layer approximation is done as follows

$$\left(\frac{\partial w_{ij}}{\partial x_k}\right)^{TSL} := n_{k,ij} \frac{(w_i - w_j)}{\|\mathbf{x}_j - \mathbf{x}_i\|_2} \tag{3-9}$$

x_j denotes the coordinate with respect to a Cartesian axis. TSL method gives us the derivatives on a face directly where as Green Gauss equation yields the derivative at a point. The derivative on a face is computed by

$$\frac{\partial w_{ij}}{\partial x_k} := \frac{1}{2}\left[\left(\frac{\partial w_i}{\partial x_k}\right)^{GG} + \left(\frac{\partial w_j}{\partial x_k}\right)^{GG}\right] \tag{3-10}$$

Averaging velocities, viscosity and conductivity in viscous terms, both in $\mathbf{f}_v \cdot \mathbf{n}$ and $\mathbf{f}_{v,turb} \cdot \mathbf{n}$ are discretized using

$$\int_{\partial\Omega_i} \mathbf{f}_v \cdot \mathbf{n} ds \approx \sum_{j \epsilon N(i)} (\mathbf{f}_v \cdot \mathbf{n})(\mathbf{W}_i, \mathbf{W}_j, \text{grad}\mathbf{W}_i, \text{grad}\mathbf{W}_j) \tag{3-11}$$

The source term in turbulence equation is discretized using

$$\int_{\Omega_i} Q d\mathbf{x} \approx vol(\Omega_i) Q(\mathbf{W}_i, \text{grad}\mathbf{W}_i) \tag{3-12}$$

After incorporating all discretizations into the governing equations we obtain a system of ordinary differential equations which looks like

$$\frac{d}{dt}\begin{pmatrix}\mathbf{W}(t) \\ \tilde{\nu}(t)\end{pmatrix} = \begin{pmatrix}-\mathbf{M}_{mean}^{-1}\mathbf{R}_{mean}(\mathbf{W}(t), \tilde{\nu}(t)) \\ -\mathbf{M}_{turb}^{-1}\mathbf{R}_{turb}(\tilde{\nu}(t), \mathbf{W}(t))\end{pmatrix} \tag{3-13}$$

Where $\mathbf{M}_{mean} := \text{diag}(\text{diag}(\text{vol}(\Omega_i))) \in \mathbb{R}^{5N \times 5N}$ and $\mathbf{M}_{turb} := \text{diag}(\text{vol}(\Omega_i)) \in \mathbb{R}^{N \times N}$ are the mass matrix for mean and turbulent flow equations respectively. This looks like a coupled system of equation as $\mathbf{W}$ and $\tilde{\nu}$ feature in both equations We assume that mean flow equations are dependent only on the set of conservative variable $\mathbf{W}$ and $\tilde{\nu}$ merely acts as a parameter while mean flow equations are solved and vice versa. The system of equations are solved in a weakly coupled manner. Thus, the two sets of Ordinary differential equations are.

$$\frac{d}{dt}\mathbf{W}(t) = -\mathbf{M}_{mean}^{-1}\mathbf{R}_{mean}(\mathbf{W}(t); \tilde{\nu}(t)) \tag{3-14}$$

$$\frac{d}{dt}\tilde{\nu}(t) = -\mathbf{M}_{turb}^{-1}\mathbf{R}_{turb}(\tilde{\nu}(t); \mathbf{W}(t)) \tag{3-15}$$

In the solver code point of view the turbulent equation is solved prior to the first iteration of non linear loop is computed. Then on both equations are solved sequentially.

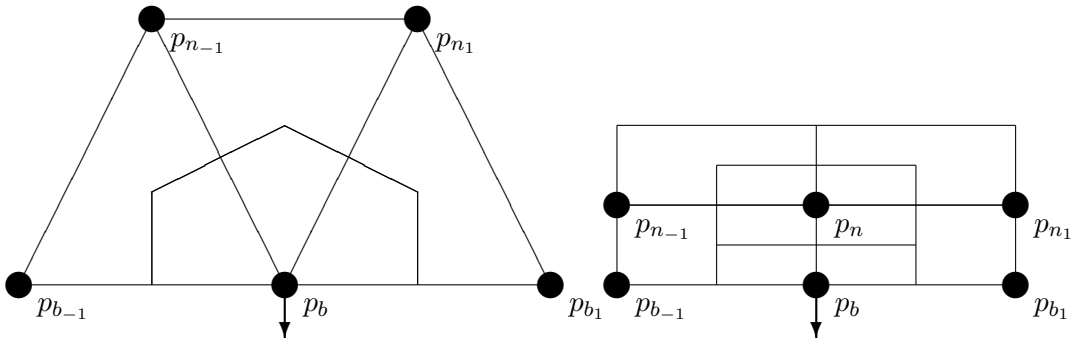


Figure 3-2: Dual mesh for a boundary point

Discretizing Boundary conditions

Boundary conditions are implemented as flux formulations. Considering Figure (3-2), the velocity gradient to discretize Equation (3-11) can be computed by considering zero velocity in Equation (3-9) at the boundary,

$$\frac{\partial u_{p_b}^{(\ell)}}{\partial x_k} \approx \left(\frac{\partial u_{p_b}^{(\ell)}}{\partial x_k}\right)_{\text{bdry}}^{\text{TSL}} := n_{k,p_n p_b} \frac{-u_{p_n}^{(\ell)}}{\|x_{p_b} - x_{p_n}\|_2}, \qquad \ell, k = 1, 2, 3. \tag{3-16}$$

The velocity gradients in the viscous term of turbulent equation can be computed by taking zero velocities at the boundary points and substituting on velocity in thin shear layer discretization (TSL) equation as 0. If elements near the boundary are triangular then the using TSL for velocity gradients can be misleading. The other option is to use the Green Gauss method. Both these method can be used but usually the elements in the boundary layer are structured so usual option is to use TSL. Temperature gradient at boundary points is taken as zero and the pressure is computed assuming zero velocity.
Moving on to the turbulent equation, the $\tilde{\nu}_{\mathbf{p}_b}$ is assumed as zero and all the source terms at the viscous wall are assumed as zero too. Far field boundary conditions like free-stream conditions are realized by the methods of characteristics. In [5] and [10] the eigenvalue decomposition of the convective fluxes are found.

Chapter 4

Numerical Solution methodology

The integral equations (2-1) were taken and discretized to get a set of ordinary differential equations (3-14 and 3-15), which needs to be numerically solved to approximate a steady state solution. An agglomeration multigrid algorithm is used to solve the set of ordinary differential equations. Inside this multigrid method we use a multistage preconditioned Runge-Kutta scheme as a smoother. The smoother can be implicit or explicit one, we will explain later what we understood by an explicit or an implicit smoother. Figure 4-1 gives an overview of the Numerical solution methodology needed to solve a non linear operator equation. With respect to this thesis, the non linear system of equation is Equation.3-13. The next layer is the Runge-Kutta smoother and then the Linear preconditioner, that needs to be solved at every stage of the smoothing algorithm. This preconditioner requires an efficient iterative linear solver and in this thesis we shall be focusing on this part, by implementing a relaxed method and a linear residual based truncation criteria. In the coming sections all layers will be explained.

4-1 Multigrid Methods

Multigrid methods are very powerful acceleration techniques. In this thesis an agglomerated multigrid algorithm [20] is used to approximate a steady state solution of the governing equation 2-1. Now within this multigrid algorithm, as you can see from Figure.(4-1) a multistage Runge-Kutta scheme is used as a smoother. First step would be to construct a hierarchy of coarser to finer meshes. Here we use agglomeration techniques, which is in simple terms joining control volumes to form coarser meshes. Next is formulation of non linear multigrid with projection and interpolation operators are required. We makes use of coarser grids to drive the solution on the finest grid quickly. An effect that helps this convergence acceleration is that majority of the time stepping and iterative schemes reduces the high frequency components of the error and the low-frequency ones are not reduced. So without multigrid, after the largest errors are eliminated in the initial phase the convergence to the steady state becomes slow. With the implementation of multigrid the low-frequency components in finer

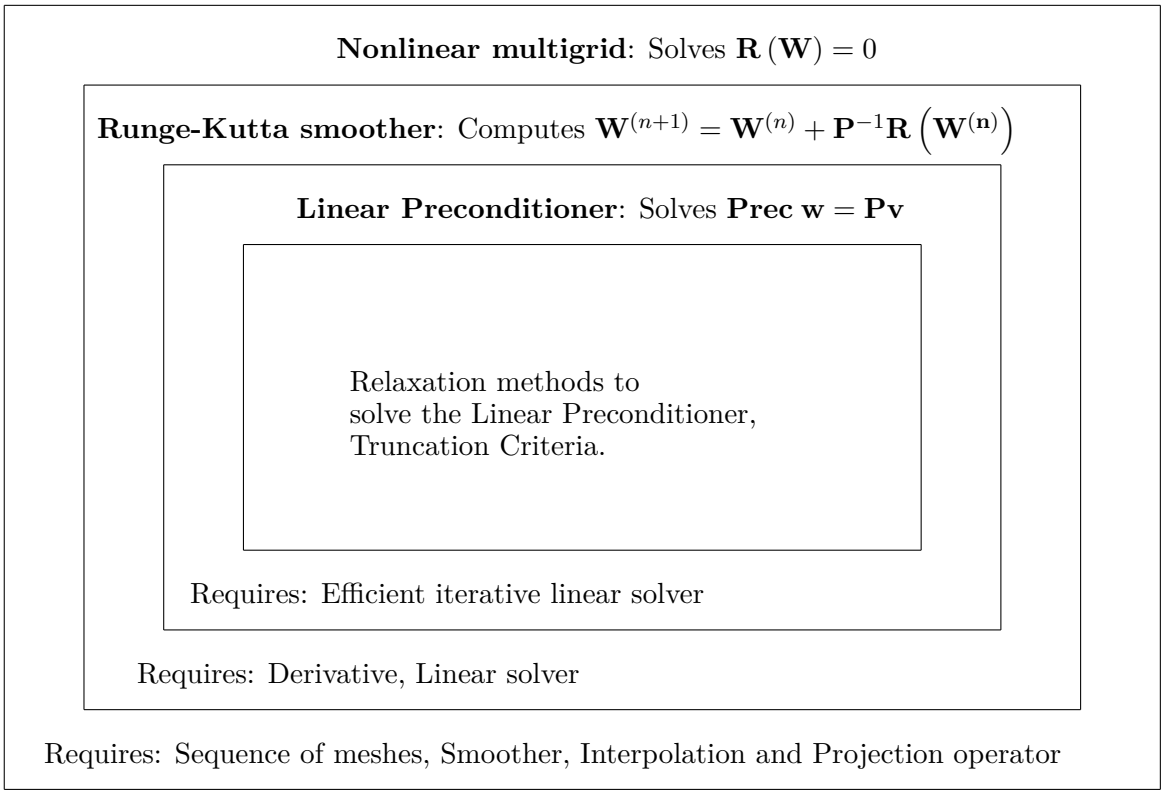


Figure 4-1: Algorithmic structure of nonlinear solution method

grids become high-frequency components in coarser grids and are eliminated and thereby reducing the overall error. This leads to accelerating the convergence. For detailed description of Multigrid schemes we refer to the textbook [21].
The coarser grids need to be generated and there are different ways to do it. The methodology employed here shall be explained in next subsection. Generally multigrid methods have the following steps:

Restriction The residual is transferred to a coarser grid, so that the error components that were low-frequency in fine level can be smoothed as they become high frequency in coarse levels.

Smoothing Reducing the error using few iterations of methods like Gauss Seidel methods. In our context the smoother is the implicit Runge-Kutta method.

Prolongation Interpolating the grid-correction computed on a coarser grid into a finer grid as shown in Equation(4-2).

The order in which these steps occur depend on the multigrid cycle selected.

4-1-1 Agglomeration Techniques

The agglomeration technique used here is a weighted graph algorithm in the library MGridGen[22]. Agglomeration is a coarsening method for unstructured meshes. Control volumes of finer grids

are fused with their neighbors to form coarse grids of irregularly shaped cells. An example is shown in Figure(4-8). Meshes for high Reynolds number viscous flow have structured boundary layers and unstructured far field. this structured boundary layer have highly stretched cells. Regions of high grid stretching are identified using a line search algorithm. This line search algorithm is explained in Chapter 5. It identifies cells with strong coupling. Along this line of strong coupling, a predetermined number of points can be fused into a coarse cell, usually two points are fused. In the part of the mesh with no line information MGridGen is applied, fusing approximately a 4:1 in 2D and 8:1 in 3D. This part where no line information is available is usually the isotropic part. Figure.(4-2 and 4-3) shows four grid levels at the anisotropic boundary layer. Blue mesh is the agglomerated mesh. Since meshes have structured, highly stretched boundary layer and isotropic far field, we combine two kinds of coarsening strategies. MGridGen is used to coarsen the isotropic part but it can not touch the anisotropic part. So a pseudo mesh in which a line is represented as a point is constructed. This pseudo mesh is coarsened using MGridGen. After that is done the points in pseudo mesh, that represent the lines with strong coupling are unpacked and agglomerated by the relation 2:1. Figure (4-4,4-5) shows how coarsening at every level looks like in the isotropic part. The isotropic agglomeration near the airfoil is shown in Figure.(4-6,4-7).

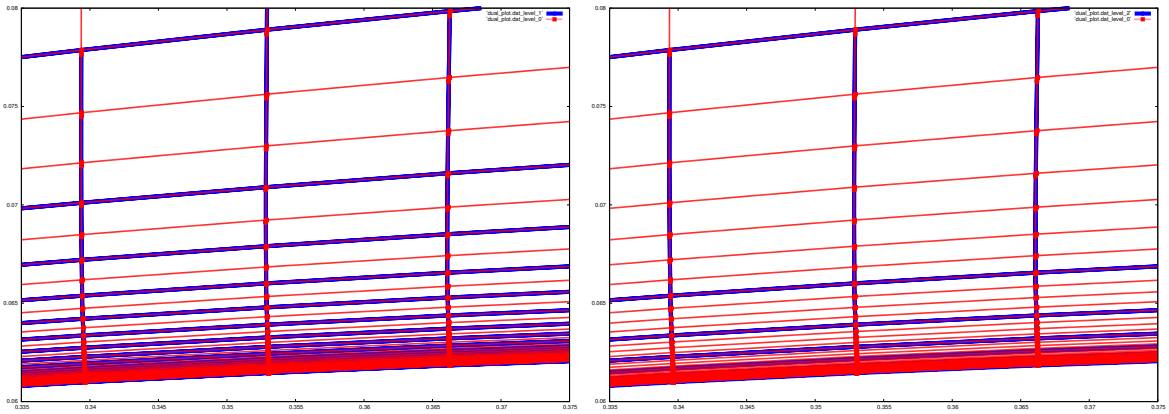


Figure 4-2: Agglomeration Mesh: Boundary Layers

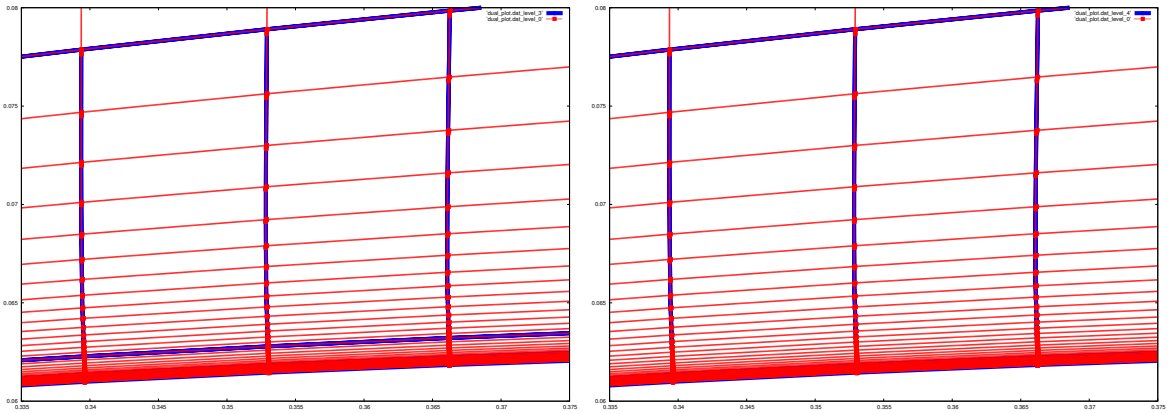


Figure 4-3: Agglomeration Mesh: Boundary Layers

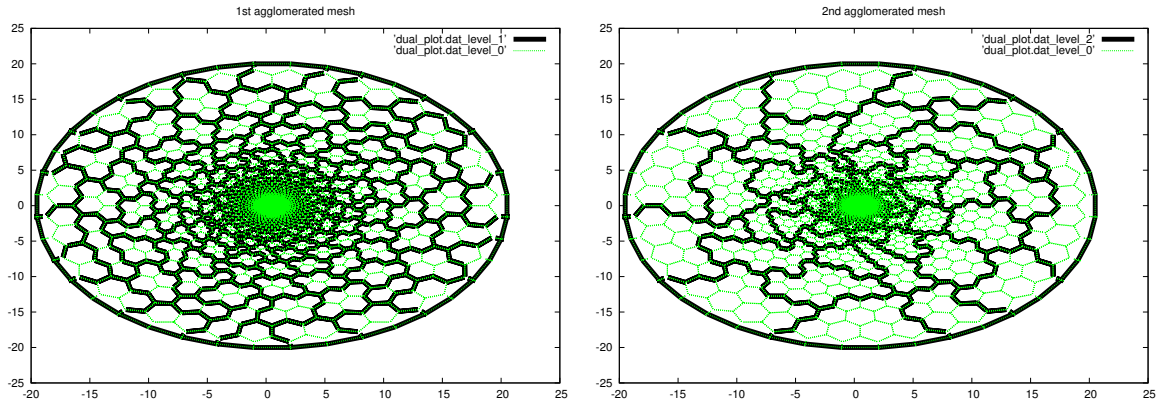


Figure 4-4: Agglomeration Mesh:Far Field

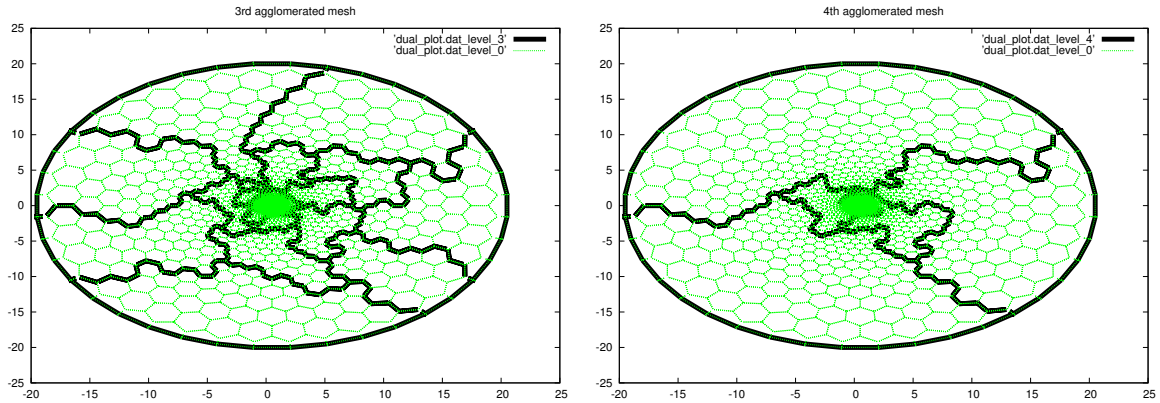


Figure 4-5: Agglomeration Mesh: Far Field

4-1-2 Coarse grid equations

In this section the mathematics for calculating residuals, fluxes for coarse grids and prolongation to finer grids are presented. The figure (4-8) is used to explain which shows a basic dual grid that is got from a triangular mesh before and after one level of agglomeration. $C_i^{(k)}$ denotes the i^{th} cell of level k. So the right figure shows four dual grid cells at level 1 which are fused into one coarse cell at level 2 on the left, the coarsening ratio of this agglomeration would be 4:1. $C_i^{(k)}$ is fused from $C_j^{(1)}$ where j belongs to the set of all children of $C_i^{(k)}$

$$C_j^{(1)} \subset C_i^{(k)}, \qquad \bigcup_{j \epsilon child(C_i^{(k)})} C_i^{(1)} = C_i^{(k)}. \tag{4-1}$$

The flow variables $\mathbf{W}_i^k$ belonging to the i^{th} cell at k^{th} level is prolongated to the finest grid level by a simple injection. Basically, if a finest cell is a child of a coarse cell, it gets the values from that coarse cell.

$$\mathbf{W}_j^1 := \mathbf{W}_i^k, \qquad j \epsilon child(C_i^{(k)}) \tag{4-2}$$

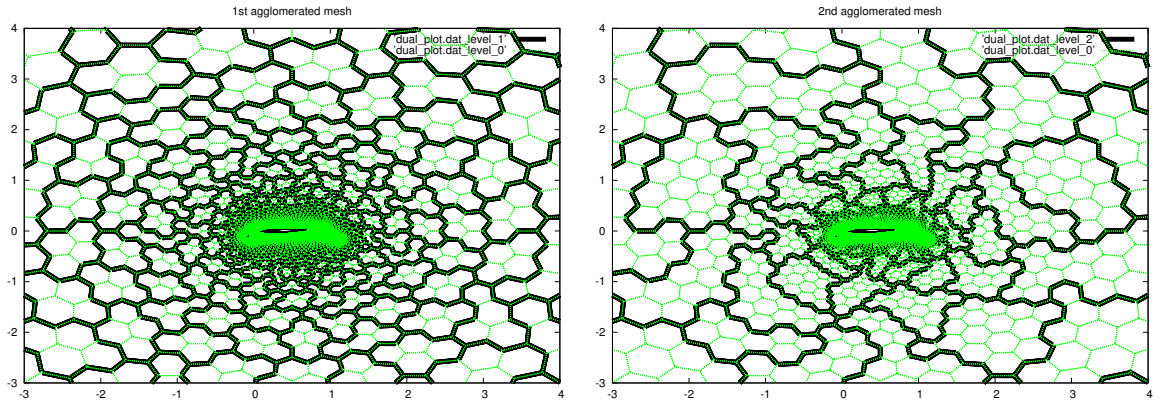


Figure 4-6: Agglomeration Mesh: Around The Wing

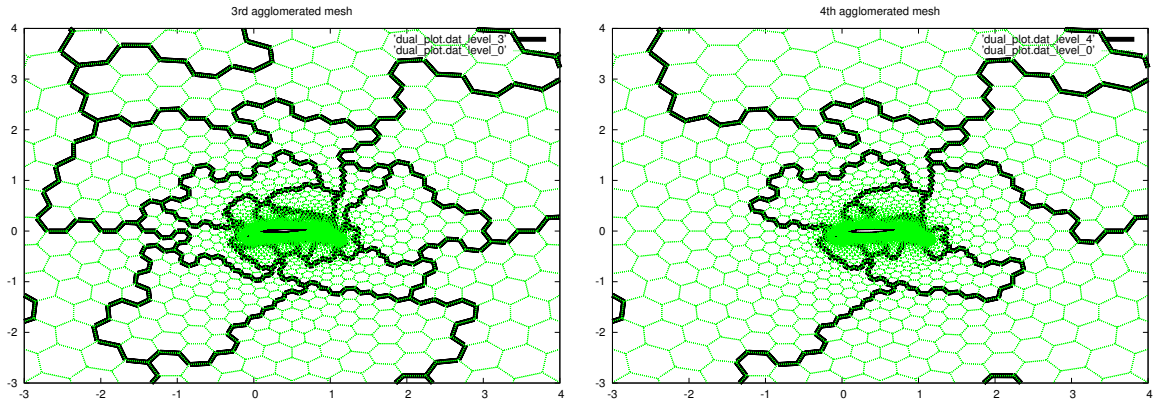


Figure 4-7: Agglomeration Mesh: Around the Wing

Then the fluxes on the edges of the finest grid are computed. The residual in the coarse grid $\mathbf{R}_j^k$ is got by summing up all the fluxes on the edges of all fine grid cells, that it is agglomerated from.

$$\mathbf{R}_i^{(k)} = \sum_{j \epsilon child(C_i^{(k)})} \sum_{k \epsilon N(j)} (\mathbf{f}_{kj}) \cdot (\mathbf{n}_{kj}) \tag{4-3}$$

The inner summation loops around all the neighbors of child elements, but as it can be seen from the figure (4-8) that there are edges in fine level which lie completely inside the corresponding coarse cell, fluxes of these edges cancel out. Thus it is sufficient to loop around all the edges of the coarse grid cell alone.

$$\mathbf{R}_i^{(k)} = \sum_{e \epsilon edges(i)} (\mathbf{f}_e \cdot \mathbf{n}_e) \tag{4-4}$$

This equation can be applied for all terms including convective, viscous and source terms in turbulent equation. Gradients in the viscous and source terms of governing equations are also calculated with the same agglomeration technique. Due to the inconsistency with respect to the viscous terms, they are weighted with a factor of $(1/2)^{k-1}$[20]. The preconditioner and

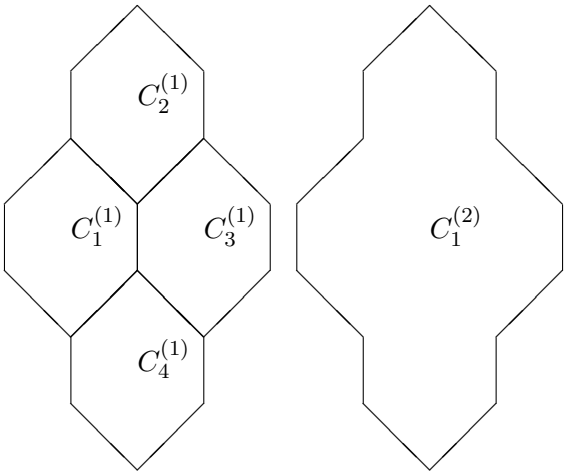


Figure 4-8: Four cells of a dual grid (left) and their agglomerated cell (right)

the ΔT for the coarse grid cells are also calculated in similar manner. Finally the restriction operator is given a volume weighted interpolation

$$\mathbf{W}_i^{(k)} = \frac{1}{vol(C_i^{(k)})} \sum_{j \epsilon child(C_i^{(k)})} vol(C_j^{(1)}) \mathbf{W}_j^{(1)} \tag{4-5}$$

Implementing the boundary conditions in this multigrid approach is straightforward, the boundary conditions for coarser grids simply come from the finest mesh. These operators are implemented in the FAS scheme and the error smoother, which is explained in the next section (4-2), is used on each multigrid levels. On the coarsest grid only the smoothing operation is performed and no solving is done because of the non linear nature. We follow the same methodology as followed in [6].

4-2 Implicit Runge-Kutta Method

Runge-Kutta methods are used for integration in time to get a steady state equation.

To derive a solution method that is used within the multigrid to approximate the discretized flow equations (3-13 and 3-14) a s-stage Runge-Kutta method is used as a smoother. Runge-Kutta schemes can be described by a Butcher scheme, given by $\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$, where

$$A = \begin{pmatrix} \alpha_{11} & 0 & \dots & 0 \\ \alpha_{11} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & \alpha_{s,s-1} & \alpha_{ss} \end{pmatrix} \qquad b = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \alpha_{s+1,s} \end{pmatrix} \qquad c = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \tag{4-6}$$

which would eventually look like

$$\begin{array}{c|cccc} 0 & \alpha_{11} & 0 & \dots & 0 \\ 0 & \alpha_{21} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \alpha_{s-1,s} & \alpha_{s,s} \\ \hline & 0 & \dots & 0 & \alpha_{s+1,s} \end{array} \tag{4-7}$$

If the matrix A has non-zero entries only in its lower-left-triangle then it is explicit. In such a scenario the stage coefficients,(4-8) for every stage can be got directly. But Implicit methods have non zero elements on its diagonal or/and upper-triangle. This makes it complicated to calculate, since a set of non-linear equation needs to be solved for every time step. This implicit methods tend to be more computationally expensive but they are designed to solve stiff system of equation. In the following subscript s denotes the stage of the Runge-Kutta scheme and for every stage a stage coefficient (k_i) needs to be solved. When a s-stage diagonally Implicit Runge-Kutta method is applied on discretized flow equations (3-13 and 3-14) the discrete evolution are given by

$$\begin{aligned} k_1(\mathbf{W}) &= -\mathbf{M}^{-1}\mathbf{R}(\mathbf{W}^{T_n} + \Delta t\alpha_{11}k_1\mathbf{W}) \\ k_2(\mathbf{W}) &= -\mathbf{M}^{-1}\mathbf{R}(\mathbf{W}^{T_n} + \Delta t\alpha_{21}k_1\mathbf{W} + \alpha_{22}\Delta t k_2\mathbf{W}) \\ &\vdots \\ k_s(\mathbf{W}) &= -\mathbf{M}^{-1}\mathbf{R}(\mathbf{W}^{T_n} + \Delta t\alpha_{s,s-1}k_s - 1\mathbf{W} + \alpha_{ss}\Delta t k_s\mathbf{W}) \end{aligned} \tag{4-8}$$

All $k - values$ from all these stages will be used to calculate the variables at next time step.

$$\mathbf{W}^{T_n+1} = \mathbf{W}^{T_n} + \Delta t\alpha_{s+1,s}k_s(\mathbf{W}) \tag{4-9}$$

The sub-indices *mean* and *turb* are skipped because the same method is used for solving both (3-13 and 3-14).

Since the Equations(4-8) are non-linear , the values for next stage can not be computed right away. Firstly we need to solve the non-linear system of equation on k. For this Newton's method is used, which is stopped after one iteration. Here the function whose root needs to be found is

$$g_j() := k_j(\mathbf{W}) + \mathbf{M}^{-1}\mathbf{R}(\mathbf{W}^{T_n} + \Delta t\alpha_{j,j-1}k_{j-1}\mathbf{W} + \alpha_{jj}\Delta t k_j(\mathbf{W})) \tag{4-10}$$

and it's derivative is

$$\frac{\partial g_j(k_j(\mathbf{W}))}{\partial k_j(\mathbf{W})} = \mathbf{I} + \Delta t\alpha_{jj}\mathbf{M}^{-1}\frac{\partial \mathbf{R}}{\partial \mathbf{W}}(\mathbf{W}^{T_n} + \Delta t\alpha_{j,j-1}k_{j-1}\mathbf{W} + \alpha_{jj}\Delta t k_j(\mathbf{W})) \tag{4-11}$$

Now, like the initial explanation of Newtons's method was given , the initial guess is taken as $k_j^{(0)}(\mathbf{W}) = 0$ and the approximate root is found out

$$k_j(\mathbf{W}) = -[\mathbf{P}_j(\mathbf{W})]^{-1}(g_j(k_j^{(0)}(\mathbf{W}))) \tag{4-12}$$

with $\mathbf{P}_j(\mathbf{W})$ given as

$$\mathbf{P}_j(\mathbf{W}) = \mathbf{I} + \Delta t \alpha_{jj} \mathbf{M}^{-1} \frac{\partial \mathbf{R}}{\partial \mathbf{W}} \left[\mathbf{W}^{T_n} + \Delta t \alpha_{j,j-1} k_{j-1} \mathbf{W} \right] \tag{4-13}$$

Now we can solve for $k_1, k_2, ..., k_s$, where the inner Newton iteration to solve the non-linear equations is truncated after one time step. The values of k for every stage in terms of the above defined variable $\mathbf{P}_j$ are

$$\begin{aligned} k_1(\mathbf{W}) &= -[\mathbf{P}_1(\mathbf{W})]^{-1} \mathbf{M}^{-1} \mathbf{R}(\mathbf{W}^{T_n}) \\ k_2(\mathbf{W}) &= -[\mathbf{P}_2(\mathbf{W})]^{-1} \mathbf{M}^{-1} \mathbf{R}(\mathbf{W}^{T_n} + \alpha_{21} \Delta t k_1) \\ &\vdots \\ k_s(\mathbf{W}) &= -[\mathbf{P}_s(\mathbf{W})]^{-1} \mathbf{M}^{-1} \mathbf{R}(\mathbf{W}^{T_n} + \alpha_{s,s-1} \Delta t k_{s-1}) \end{aligned} \tag{4-14}$$

and the conservative variables for the next time step is given by

$$\mathbf{W}^{T_{n+1}} = \mathbf{W}^{T_n} + \alpha_{s+1,s} \Delta t k_s(\mathbf{W}) \tag{4-15}$$

The update is defined as $\mathbf{W}^{(0)} := \mathbf{W}^{T_n}$ and when we consider the $k_1(\mathbf{W})$ and apply it in the $k_2(\mathbf{W})$ before we applied the Newton's method, we could see the next updates as

$$\mathbf{W}^{(j)} := \mathbf{W}^{T_n} - \alpha_{j+1,j} \Delta t [\mathbf{P}_j(\mathbf{W})]^{-1} \mathbf{M}^{-1} \mathbf{R}(\mathbf{W}^{(j-1)}) \tag{4-16}$$

Now by induction, the Runge-Kutta scheme can be formulated as follows

$$\begin{aligned} \mathbf{W}^{(0)} &:= \mathbf{W}^{T_n} \\ \mathbf{W}^{(1)} &:= \mathbf{W}^{(0)} - \alpha_{21} \Delta t [\mathbf{P}_1(\mathbf{W})]^{-1} \mathbf{M}^{-1} \mathbf{R}(\mathbf{W}^{(0)}) \\ &\vdots \\ \mathbf{W}^{(s)} &:= \mathbf{W}^{(0)} - \alpha_{s+1,s} \Delta t [\mathbf{P}_s(\mathbf{W})]^{-1} \mathbf{M}^{-1} \mathbf{R}(\mathbf{W}^{(s-1)}) \end{aligned} \tag{4-17}$$

The above algorithm is the method that we use to solve and one can see that it looks like an explicit method with an additional operator $[\mathbf{P}_j(\mathbf{W})]^{-1}$. This is why the implicit scheme can indeed be interpreted as a preconditioned explicit scheme.

$$\mathbf{h}_j = \alpha_{j+1,j} \Delta t [\mathbf{P}_j(\mathbf{W})]^{-1} \mathbf{M}^{-1} \mathbf{R}(\mathbf{W}^{(j-1)}) \tag{4-18}$$

Thus, for every stage a linear equation in Equation(4-19) needs to be solved.

$$\mathbf{P}_j(\mathbf{W}) \mathbf{h}_j = \alpha_{j+1,j} \Delta t \mathbf{M}^{-1} \mathbf{R}(\mathbf{W}^{(j-1)}) \tag{4-19}$$

and since we have $\mathbf{P}_s(\mathbf{W})$ in hand, substituting it in this will yield us

$$\left((\Delta t)^{-1} \mathbf{M} + \alpha_{jj} \frac{\partial \mathbf{R}}{\partial \mathbf{W}} \right) \mathbf{h}_j = \alpha_{j+1,j} \mathbf{R}(\mathbf{W}^{(j-1)}) \tag{4-20}$$

It needs to be pointed out that the primary aim is not to solve this linear equation to it's maximum accuracy. This linear equation's main function is to deliver a suitable update for the outer non-liner loop. In this work we concentrate on ways to efficiently solve this linear system for making the code more robust and at the same time not indulge in over solving of this system of equation is to obtain a steady state solution to our governing equation.

A vital term in the above equation which might prove to be computationally expensive if proper way to calculate it is not chosen is the derivative $\frac{\partial \mathbf{R}}{\partial \mathbf{W}}$. Calculating a full Jacobian by considering the full second order stencil .i.e. direct neighbors and their direct neighbors too, increases the storage requirements and every iteration becomes more computationally expensive. So a good approximation for Jacobian needs to be calculated. For this we consider only a first order approximation of the residual function, .i.e. second order terms are neglected. Consider $\mathbf{R}$ represent the stencil of a first order finite volume discretization, each entry of the vector valued function $\mathbf{R}_i$ depends on the unknown $\mathbf{W}_i$ located in the control volume i and their direct neighbors $\mathbf{W}_{j,j\in N(i)}$ and the neighbors of the neighbors $\mathbf{W}_{k,k\in N(j),j\in N(i)}$. For example, on a 3D hexahedral structured mesh we have

$$|\{k, k \in N(j), j \in N(i)\}| = 25 \tag{4-21}$$

$$|\{k, k \in N(i)\}| = 9 \tag{4-22}$$

Therefore each row of a matrix $\frac{\partial \mathbf{R}}{\partial \mathbf{W}}$ has 25 non zero entries in the case of second order and 9, if it is first order. For a mesh with N points locating d degrees of freedom, the number of entries is $N \cdot d \cdot d \cdot 25$ (for second order) and $N \cdot d \cdot d \cdot 9$(for first order). And for the governing equation we have $d = 5$. Considering the Programing language C, a double requires 8 bytes, the memory requirement of matrix are

$$N \cdot 5 \cdot 5 \cdot 25 \cdot 8 = 5000 N bytes \tag{4-23}$$

$$N \cdot 5 \cdot 5 \cdot 9 \cdot 8 = 1800 N bytes. \tag{4-24}$$

For a mesh with $N = 1e06$ nodes about 5GB memory is required to construct the second order matrix $\frac{\partial \mathbf{R}}{\partial \mathbf{W}}$ and 1.8GB for it's first order discretization. By using the first order approximation of the derivative this is the storage requirement that is being saved. The operator $\left|\mathbf{A}_{ij}^{Roe}\right|$, in the discretization of the convective part of governing equation, is assumed to be locally constant. The effective viscosity μ_{eff} and κ_{eff} are assumed to be constant as well. To demonstrate the approximation of the jacobian, a first order discretization of the the inviscid part of governing equation(mean flow) may be represented as

$$\int_{\partial\Omega_i} \mathbf{f}_c \cdot \mathbf{n} ds \approx \sum_{j\epsilon N(i)} \frac{1}{2} \left((\mathbf{f}_c \cdot \mathbf{n}_{ij}) (\mathbf{W}_i) + ((\mathbf{f}_c \cdot \mathbf{n}_{ij}) (\mathbf{W}_j)) - \frac{1}{2} \left|\mathbf{A}_{ij}^{Roe}\right| (\mathbf{W}_j - \mathbf{W}_i). \right. \tag{4-25}$$

The term $\mathbf{D}_{ij}(\mathbf{W})$ in Equation(3-7)is approximated to only the Roe matrix term. Now the derivative of convective flux is approximated to

$$\frac{\partial \mathbf{R}_i^{1st}}{\partial \mathbf{W}_k} = \begin{cases} \sum_{j\in N(i)} \left|\mathbf{A}_{ij}^{Roe}\right| & k = i \\ -\left|\mathbf{A}_{ij}^{Roe}\right| & k \in N_i \\ 0 & k \neq i, k \notin N_i \end{cases} \tag{4-26}$$

On a global scale a row pertaining to a certain cell i of the $\frac{\partial \mathbf{R}^{1st}}{\partial \mathbf{W}}$ may look like

$$\frac{\partial \mathbf{R}_i^{1st}}{\partial \mathbf{W}} = \Bigg(0,\dots,0,\frac{\partial \mathbf{R}_i^{1st}}{\partial \mathbf{W}_{j1}},0,\dots,0,\frac{\partial \mathbf{R}_i^{1st}}{\partial \mathbf{W}_{j2}},0,\dots,0,\frac{\partial \mathbf{R}_i^{1st}}{\partial \mathbf{W}_{j3}},0,\dots,0,\frac{\partial \mathbf{R}_i^{1st}}{\partial \mathbf{W}_{i}},$$
$$0,\dots,0,\frac{\partial \mathbf{R}_i^{1st}}{\partial \mathbf{W}_{j4}},0,\dots,0,\frac{\partial \mathbf{R}_i^{1st}}{\partial \mathbf{W}_{j5}},0,\dots,0,\frac{\partial \mathbf{R}_i^{1st}}{\partial \mathbf{W}_{j6}},0,\dots,0\Bigg) \tag{4-27}$$

As the preconditioner needs to be efficiently stored in order to be quickly accessed, block compressed sparse row(CSR) format is used to store the matrix and corresponding system of linear equations. For further description of the CSR format we refer to the textbook [23].
We are interested only in the steady state computation and the time step is defined as follows.

$$\Delta t_i = CFL.vol(\Omega_i)\Bigg[\sum_{j\in N(i)}\frac{1}{2}(|V_{ij}| + a_{ij}svol(\Omega_{ij}))+$$
$$\frac{C_v(\mu_{eff})_{ij}svol(\Omega_{ij})}{\left\|\mathbf{x}_{\mathbf{p}_i}-\mathbf{x}_{\mathbf{p}_j}\right\|_2\rho_i}\left(max\left\{\frac{4}{3},\frac{(k_{eff})_{ij}(\gamma-1)}{(\mu_{eff})_{ij}}\right\}\right)\Bigg] \tag{4-28}$$

where $C_v := 8$ and in the linear system of equation the Δt is replaced by ΔT , where

$$\Delta T := diag(diag(\Delta t_i)) \in \mathbb{R}^{5N\times 5N} \tag{4-29}$$

this is for mean flow equation where the variable matrix $\mathbf{W}$ comprises of 5 variables, where as for mean flow were only ν is solved and the ΔT looks like

$$\Delta T := diag(\Delta t_i) \in \mathbb{R}^{N\times N} \tag{4-30}$$

After applying the jacobian approximation and time step matrix ΔT the inner linear loop equation looks like

$$\left((\Delta T)^{-1}\mathbf{M} + \varepsilon\alpha_{jj}\frac{\partial \mathbf{R}^{1st}}{\partial \mathbf{W}}\right)\mathbf{h}_j = \alpha_{j+1,j}\mathbf{R}(\mathbf{W}^{(j-1)}) \tag{4-31}$$

In this whole section sub-scripts of 'mean' and 'turb' haven't been used since the methodology is same for both discretized equation. Only difference being the variable passed in will be a vector of variables ($\mathbf{W}$) w.r.t to mean flow equations and single variable when it comes to turbulent (ν).
One of the major challenge lies in finding the approximate solution of this above equation of the form, $Preconditioner \cdot x = b$ and finding the right method to do it. In the next chapter the different Iterative solution methods shall be explained extensively.

Chapter 5

Linear Solution Methods

The linear system that needs to be solved is given in the Equation (4-31). This linear system is an update to the outer non-linear loop. This system needs to be solved for every Runge-Kutta stage. The linear system at hand need not be solved to the maximum accuracy and the aim is to find ways to approximate efficiently the system of equation. If you consider CFD meshes and specifically the high Reynolds number viscous flows, it is known that in order to capture the steep gradients occurring in viscous boundary layers the cells near the boundary of a wing or any outer body have boundary layer cells which are structured, very thin, and their lengths run parallel to the flow direction, leading to cells with high aspect ratios. Mathematically these cells boil down to creating a discrete system that is stiff in nature. Certain error modes while solving discrete equations are difficult to remove when the system has high numerical stiffness. Thus due to the viscous boundary layer, the system is ill-conditioned. This chapter will deal with the linear solution methods at hand to deal with these anisotropies and how to efficiently solve it.

Straightforward iterative solution methods include Jacobi, Gauss-Seidel, symmetric Gauss-Seidel. For mean flow equations the block versions of these methods are used. Now these methods can be significantly improved by making use of the anisotropies representing direction of strong coupling present in these meshes. Yes, these cells tend to have strong coupling and directions of strong coupling can be found out in order to be solved together. These directions of strong coupling are identified using a line-search algorithm, which is explained later in this section. Along this line the Jacobian shall look like a (block) tridiagonal matrix as seen in Equation(5-6). By searching for several lines and assembling the 1st order Jacobian matrices along those lines, what we gain is that it decouples and gives us several small scale block tridiagonal matrices.

To give you a perspective of how the evolution towards these line implicit method may look like, first here is equation for Block Jacobi method(see [14],[7]),

$$\mathbf{h}_i^{(k+1)} = (\mathbf{P}_{i,i})^{-1} \left(\mathbf{b}_i - \sum_{j=i, j\neq i}^{N} \mathbf{P}_{(i,j)} \mathbf{h}_j^{(k)} \right), i = 1, ..., N \tag{5-1}$$

and Block Gauss-Seidel method

$$\mathbf{h}_i^{(k+1)} = (\mathbf{P}_{i,i})^{-1}\left(\mathbf{b}_i - \sum_{j=i}^{i-1}\mathbf{P}_{(i,j)}\mathbf{h}_j^{(k+1)} - \sum_{j=i+1}^{N}\mathbf{P}_{(i,j)}\mathbf{h}_j^{(k)}\right), i = 1, ..., N \tag{5-2}$$

These are conventional methods well known in linear algebra, where $\mathbf{h}$ is the vector of unknowns and $\mathbf{P}$ is the matrix and $\mathbf{b}$ is the right hand side of the equation of type $P \cdot h = b$. The system of equation being described here is the Equation (4-31). $P \cdot h = b$ is approximately solved by a Gauss-Seidel iteration, and the initial guess is taken as $h^{(0)} = 0$.

Now moving on to the Block Jacobi, which is line implicit with line information of lines with strong coupling, and shall be called Block line Jacobi henceforth.

$$\mathbf{h}_{G_i}^{(k+1)} = \mathbf{Tri}_{G_i}^{-1}\left(\mathbf{b}_{G_i} - \sum_{j\in G_1, G_2, ...G_n, j\notin G_i}\mathbf{P}_{G_i,j}\mathbf{h}_j^{(k)}\right), i = 1, ..., n \tag{5-3}$$

and similarly the Block line Gauss-Seidel is given by

$$\mathbf{h}_{G_i}^{(k+1)} = \mathbf{Tri}_{G_i}^{-1}\left(\mathbf{b}_{G_i} - \sum_{j\in G_1, G_2, ...G_{i-1}, j\notin G_i}\mathbf{P}_{G_i,j}\mathbf{h}_j^{(k+1)} - \sum_{j\in\{1,...,N\}\setminus\{G_1,...,G_i\}}\mathbf{P}_{G_i,j}\mathbf{h}_j^{(k)}\right), i = 1, ..., n \tag{5-4}$$

where G_i are lines which have points with strong coupling. $\mathbf{Tri}_{G_i}$ is the tridiagonal matrix along this line. This is explained in Equation (5-5,5-6)

The new terms that feature in line-implicit equations are group of line $G_1, G_2, ..., G_n$ and $\mathbf{Tri}_{G_i}$, which is tridiagonal block matrix pertaining a certain line G_i. How the lines are searched and constructed in a given mesh, and the assembling of the simplified derivative matrix of the residual function, are the two important ingredients when it comes to developing an efficient line-implicit preconditioner.

Line search algorithm is used to find points with strong coupling. This algorithm's main aim is to search for points representing anisotropy and these anisotropies can be interpreted as lines. As mentioned earlier anisotropies can be found in structured boundary layers and elsewhere too in a hybrid mesh. The algorithm implemented here is based on weighted graph algorithm(see [10]), which follows the following steps:

1. For every edge in the mesh a weight is calculated. It is taken as the inverse of the edge length.

2. The ratio of maximum to average weights is used as an indication of the local anisotropy.

3. The vertices are sorted according to these ratios.

4. The first vertex in this sorted list is taken as starting point of a line and the neighboring vertices are searched for strong connection, this neighbor shouldn't be a part of other lines and the edge weights should be greater than a threshold value. This value in itself should be greater than 1.

5. This is continued till a situation is reached where there aren't any other vertices with strong coupling and the search has reached an isotropic region of the mesh.

6. The cycle terminates when no additional vertex can be found.

Lines are denoted as G_i for indices $i = 1, ..., n$ and

$$G_i \subset G, \quad G_j \cap G_i = \varnothing, \quad \bigcup_{j=1}^{n} G_j = G, \quad r_i := \#(G_i) \tag{5-5}$$
$$G_i = \{l_i^1, l_i^2, ..., l_i^{r_i}\}$$

To elaborate, every line is a subset of the mesh, two lines do not have vertex in common and r_i is the number of elements in G_i. Meshes are hybrid with isotropic regions and in these regions these lines reduce to a point. So algorithmically, when $r_i > 1$, G_i is a line, otherwise it is a point. If G_i is a line then its points are denoted by l_i^j. l_i^j is the jth point in the line G_i.

Preconditioner After the lines are found preconditioner needs to be constructed to solve. It is worthwhile to mention again that the preconditioner is a first order approximation to the jacobian of the residual function. The residual only depends on it's neighbors.
Now consider a line $G_i = l_i^1, l_i^1, ..., l_i^{r_i}$, Along this line the derivative $\frac{\partial \mathbf{R}_{G_i}}{\partial \mathbf{W}_{G_i}}$ can be represented by a block tridiagonal matrix

$$\frac{\partial \mathbf{R}_{G_i}}{\partial \mathbf{W}_{G_i}} = \begin{bmatrix} \frac{\partial \mathbf{R}_{l_i^1}}{\partial \mathbf{W}_{l_i^1}} & \frac{\partial \mathbf{R}_{l_i^1}}{\partial \mathbf{W}_{l_i^2}} & & & \\ \frac{\partial \mathbf{R}_{l_i^2}}{\partial \mathbf{W}_{l_i^1}} & \frac{\partial \mathbf{R}_{l_i^2}}{\partial \mathbf{W}_{l_i^2}} & \frac{\partial \mathbf{R}_{l_i^2}}{\partial \mathbf{W}_{l_i^3}} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{\partial \mathbf{R}_{l_i^{r_i-1}}}{\partial \mathbf{W}_{l_i^{r_i-2}}} & \frac{\partial \mathbf{R}_{l_i^{r_i-1}}}{\partial \mathbf{W}_{l_i^{r_i-1}}} & \frac{\partial \mathbf{R}_{l_i^{r_i-1}}}{\partial \mathbf{W}_{l_i^{r_i}}} \\ & & & \frac{\partial \mathbf{R}_{l_i^{r_i}}}{\partial \mathbf{W}_{l_i^{r_i-1}}} & \frac{\partial \mathbf{R}_{l_i^{r_i}}}{\partial \mathbf{W}_{l_i^{r_i}}} \end{bmatrix} \tag{5-6}$$

Along this line the corresponding block tridiagonal matrix $\mathbf{Tri}_{G_i} \in \mathbb{R}^{5r_i \times 5r_i}$ is given by

$$\mathbf{Tri}_{G_i} := (\Delta t_{G_i})^{-1} \mathbf{M}_{G_i} + \alpha_{jj} \frac{\partial \mathbf{R}_{G_i}}{\partial \mathbf{W}_{G_i}} \tag{5-7}$$

If r_i is 1, then it is a point and $\mathbf{Tri}_{G_i}$ is a diagonal block. So if n is the no. of sets into which the line search algorithm divides the points in whole domain into. Then for every Runge-Kutta stage, n block tridiagonal linear systems need to be solved. If the RK method used is a 3-stage one then every outer non-linear iterations need $3 \times n$ block tridiagonal linear systems to be solved. The lines are found to take advantage of the strong coupling in order to improve the linear solution method, and thus this can be made use only along these points. Thus at isotropic points, point relaxation methods are used.

Figure (5-1) shows the structure of how the preconditioner with look like in Equation (4-31). The anisotropic part corresponds to all the lines searched using the line search algorithm.

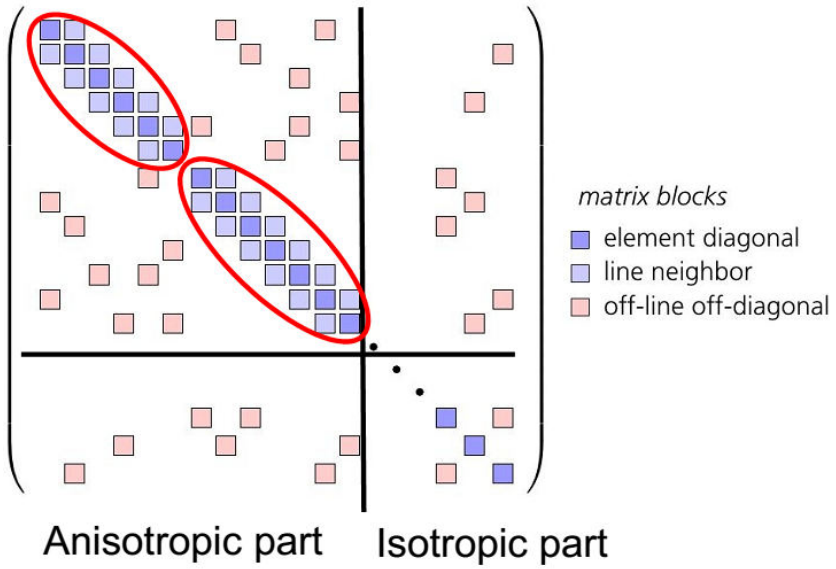


Figure 5-1: Structure of the Preconditioner Matrix

These lines are along the direction of strong coupling. It can be seen that in the anisotropic part it is tridiagonal. The elements those are not there in the diagonal are taken to the right hand side. The resulting block tridiagonal matrix can be used to solve by the algorithm explained later in this chapter. At isotropic part the matrix is block diagonal. To elaborate on hybrid mix of point and line relaxation methods, the Block symmetric line Gauss-Seidel method is taken. In general one sweep is a forward Gauss-Seidel sweep followed by a backward Gauss-Seidel sweep. With Block symmetric line Gauss-Seidel method:

- Gauss-Seidel sweep over lines, tri-diagonal systems are approximated. Exchanged approximate solution between domains in parallel environment.
- Gauss-Seidel sweep over all other points and exchange solution
- Gauss-Seidel sweep over all points in a reverse manner and then exchange solution
- Gauss-Seidel sweep over all lines in a reverse manner and then exchange solution

There are several numerical experiments explained in Chapter 6. Among them one relevant one at this point would be a comparison between symmetric Gauss-Seidel(SGS) and line symmetric Gauss-seidel(LSGS) methodology to solve the preconditioner. The convergence histories of residuals when the L2 Hybrid mesh is computed, one with SGS and other with LSGS are shown in Figure(5-2) and all other settings as shown in Table.(6-5). The line symmetric Gauss-Seidel converges earlier compared to the symmetric Gauss-Seidel.

5-1 Relaxed Methods

With both Jacobi and Gauss-Seidel it can be seen that the solution of previous iteration isn't used in the current iteration. With relation methods those are used and convergence of the

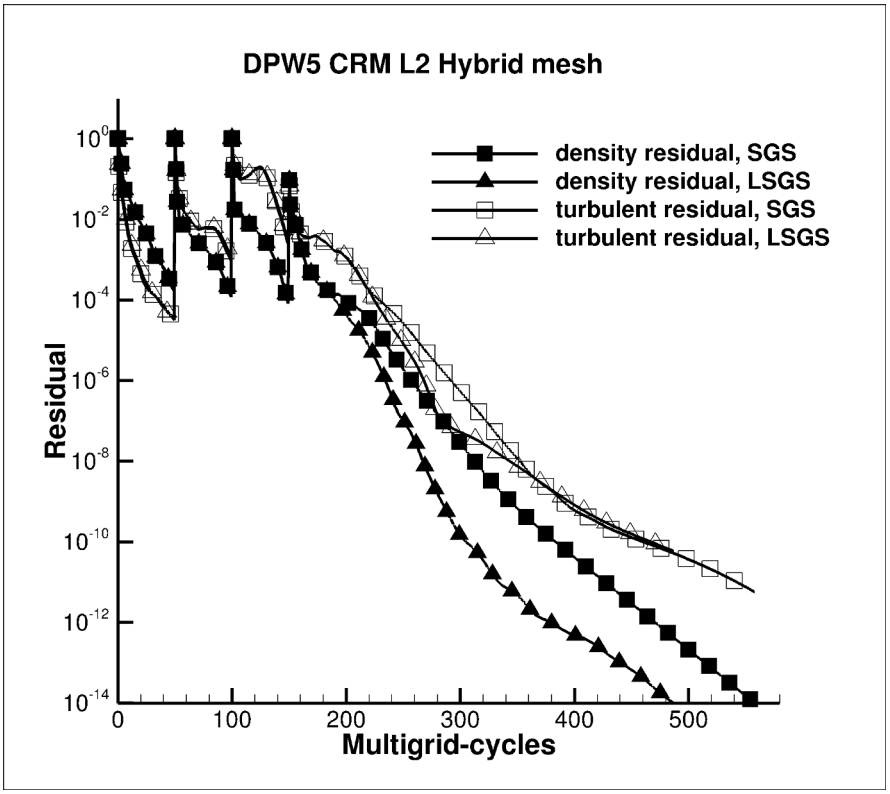


Figure 5-2: Convergence histories: Comparison between Symmetric Gauss-Seidel and line Symmetric Gauss-Seidel

solution can be considerably increased by using a linear combination of old and new solutions. For general description of relaxed Gauss-Seidel methods we refer to the Textbook([23]). So a Relaxed Gauss-Seidel method looks like

$$\mathbf{h}_i^{(k+1)} = \omega\Bigg[(\mathbf{P}_{i,i})^{-1}\Bigg(\mathbf{b}_i - \sum_{j=i}^{i-1}\mathbf{P}_{(i,j)}\mathbf{h}_j^{(k+1)} - \sum_{j=i+1}^{N}\mathbf{P}_{(i,j)}\mathbf{h}_j^{(k)}\Bigg)\Bigg] + (1-\omega)\mathbf{h}_i^{(k)}, i = 1, ..., N \tag{5-8}$$

where ω is the relaxation parameters and $\mathbf{h}_i^{(k)}$ is the solution from previous iteration. Equation 5-8 can be rewritten by

$$\mathbf{h}_i^{(k+1)} = \omega(\mathbf{h}_i^{GS}) + (1-\omega)\mathbf{h}_i^{(k)}, i = 1, ..., N. \tag{5-9}$$

and $\mathbf{h}_i^{GS}$ being the solution we get by solving the Gauss-Seidel method at the current time step, $k+1$. If the ω is set at 1, it can be seen that the old solution term vanishes and we get the Gauss-Seidel method. The Block relaxed symmetric line Gauss-Seidel is given by

$$\mathbf{h}_{G_i}^{(k+1)} = \omega\Bigg[\mathbf{Tri}_{G_i}^{-1}\Bigg(\mathbf{b}_{G_i} - \sum_{j\in G_1,G_2,...G_{i-1},j\notin G_i}\mathbf{P}_{G_i,j}\mathbf{h}_j^{(k+1)} - \sum_{j\in\{1,...,N\}\setminus\{G_1,...,G_i\}}\mathbf{P}_{G_i,j}\mathbf{h}_j^{(k)}\Bigg)\Bigg] + (1-\omega)\mathbf{h}_{G_i}^{(k)}, i = 1, ..., n \tag{5-10}$$

When $\omega > 1$, it's called over relaxation and the method tends to run quicker than Gauss-Seidel([23]) and if it is less than 1, it's called under relaxation and convergence takes more time. Here we have implemented the relaxation parameter in both Gauss-Seidel (turbulent) and Block Gauss-Seidel (mean flow) methods and investigated how it has improved the robustness of the algorithm at high CFL numbers. In the next Chapters we explore how the Relaxed Line Gauss-Seidel performs in comparison with the Line Gauss-Seidel on different test cases.

5-2 Block Tridiagonal linear systems

In this thesis our focus is to efficiently approximate Equation.(4-31). Along with the line information due to the anisotropy in the mesh, the inner-linear loop requires us to solve many Block- tridiagonal linear system, which looks like Figure.(5-1) without the off-diagonal block elements. Here we present the algorithm that is used for solving such systems. LU factorization is used to get a Block lower triangle matrix, with Identity matrix (I) on its diagonals and a Block Upper triangle matrix. The methodology is similar to the Thomas algorithm used extensively for solve linear systems, but since this is a Block system, in the place of solving for a simple variable whilst back-substitution, here a linear system of matrices

are solved (see [24]). The block tridiagonal matrix of the form $A \cdot x = b$

$$\begin{bmatrix} \mathbf{D_1} & \mathbf{F_1} & & \dots & 0 \\ \mathbf{E_1} & \mathbf{D_2} & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \mathbf{F_{N-1}} \\ 0 & \dots & & \mathbf{E_{N-1}} & \mathbf{D_N} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \vdots \\ \vdots \\ \mathbf{x_N} \end{bmatrix} = \begin{bmatrix} \mathbf{b_1} \\ \mathbf{b_2} \\ \vdots \\ \vdots \\ \mathbf{b_N} \end{bmatrix} \tag{5-11}$$

E, D and F are matrices, in our case these are Jacobian matrices of each point in a line. The $N \times N$ Block matrix on the right hand side can be LU decomposed into Lower triangular matrix with I on it's diagonals and Upper triangular matrix.

$$A = \begin{bmatrix} \mathbf{I} & & & \dots & 0 \\ \mathbf{L_1} & \mathbf{I} & & & \vdots \\ & \ddots & \ddots & & \\ \vdots & & \ddots & \ddots & \\ 0 & \dots & & \mathbf{L_{N-1}} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{U_1} & \mathbf{F_1} & & \dots & 0 \\ 0 & \mathbf{U_2} & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \mathbf{F_{N-1}} \\ 0 & \dots & & 0 & \mathbf{U_N} \end{bmatrix} \tag{5-12}$$

The algorithm to get $\mathbf{L_i}$ and $\mathbf{U_i}$ is
$\mathbf{U_1} = \mathbf{D_1}$
for i=2:N **do**
Solve $\mathbf{L_{i-1}} \cdot \mathbf{U_{i-1}} = \mathbf{E_{i-1}}$ for $\mathbf{L_{i-1}}$
$\mathbf{U_i} = \mathbf{D_i} - \mathbf{L_{i-1}F_{i-1}}$
end for

Now we have the system becomes $\mathbf{LU} \cdot x = b$. Here we consider $\mathbf{U} \cdot x = y$ and do a block forward elimination by finding out y from $\mathbf{L} \cdot y = b$. And then we have y and now a backward substition is done easily since $\mathbf{U}$ is an upper diagonal matrix.
$y_1 = b_1$
for i=2:N **do**
$y_i = b_i - \mathbf{L_{i-1}} y_{i-1}$
end for

Solve $\mathbf{U_N} \cdot x_N = y_N$ for x_N

for i=N-1:-1:1 **do**
Solve $\mathbf{U_i} x_i = y_i - \mathbf{F_i} x_{i+1}$for x_i
end for

Chapter 6

Numerical Test Cases

In the previous Chapters the solution methodology, methods to solve the linear systems arising at every stage of the smoother algorithm, and implementation of the relaxation parameter into Gauss-Seidel methods to solve this linear systems were explained. In this Chapter with the help of several test cases available in literature we investigate how the implementation of relaxation parameter has changed the solution methodology. NACA0012 and RAE2822 are the 2D test cases that we have selected. To investigate the effects on 3D configurations we have taken the CRM meshes from the fifth AIAA Drag Prediction workshop. Here both the hexahedral and hybrid meshes of grid levels L1 to L4 is considered. As another 3D test case the NASA TRAP wing configuration is also computed to assess the behavior of the method. CFD solvers in general should reduce the solution parameters to a minimum. In this quest we implement a linear criteria for the inner linear loop's approximate solution. In the Section(6-2) we computed all the meshes after implementing the truncation criteria and present the result. Eigenvalue analysis of the solution method to show the change after truncation criteria for all DPW5 meshes are also presented.

6-1 Relaxed Methods

Algorithm (4-17) is the preconditioned- implicit Runge-Kutta method that we use and the pre-conditioner used is given in equation (4-31). In this work we focus on the linear solution method used to efficiently approximate a solution of the linear system Equation.(4-31). Relaxed Gauss-Seidel methods as shown in equation (5-8 and 5-10) were implemented. In this section we will investigate the influence of this relaxation parameter on the methodology. While comparing, the terminology $No-relax$ or $Non-relaxed$ is used, which is nothing but the normal Gauss-Seidel methods. As mentioned while introducing the relaxation parameter in Chapter 5, when $\omega < 1$ it is called under relaxation and when $\omega > 1$ it is called over relaxation.
All 3D test cases are computed fully turbulent without transition. These test cases are run in parallel and the information about number of domains is given in the Table (6-4-6-5). In this

work for all test cases three stage Runge-Kutta scheme with coefficients $\alpha_{11} = \alpha_{22} = \alpha_{33} = 1$, $\alpha_{21} = 0.15$, $\alpha_{32} = 0.4$ and $\alpha_{43} = 1$.

The CFL number is set low at the beginning and is increased by a factor every iteration after the tenth iteration. For example, the initial CFL number in most of our computations, CFL_{init} is taken as 3, it remains 3 till the 10th iteration and then starts increasing by a factor of γ till it reaches CFL_{max}.

$$CFL_{mean}(n) = min\{CFL_{init} \cdot f_{mean}(n), CFL_{mean,max}\} \tag{6-1}$$

$$CFL_{turb}(n) = min\{CFL_{init} \cdot f_{turb}(n), CFL_{turb,max}\} \tag{6-2}$$

Where $f_{mean}(n)$ is

$$f_{mean}(n) = f_{turb}(n) = \begin{cases} 1 & n < 10 \\ \gamma^{n-10} & n \geq 10 \end{cases} \tag{6-3}$$

These values varies with test cases so they are given in the description of the examples. In all cases a relation of $1:3$ steps between the mean flow and turbulent flow equations is used. This means for every multigrid cycle of mean flow equation three cycles are performed for turbulent equation. The residuals are computed by a volume weighted norm, normalized with respect to the residuals evaluated with free stream values, like done in ([6]).

$$density\ residual(n) := \frac{\sqrt{\sum_{j=1}^{N} \frac{(\mathbf{R}_{j,mean,\rho}(\mathbf{W}^{T_n}))^2}{(vol(\Omega_j))^2}}}{\sqrt{\sum_{j=1}^{N} \frac{(\mathbf{R}_{j,mean,\rho}(\mathbf{W}_\infty))^2}{(vol(\Omega_j))^2}}} \tag{6-4}$$

$$turbulent\ residual(n) := \frac{\sqrt{\sum_{j=1}^{N} \frac{(\mathbf{R}_{j,turb}(\tilde{\nu}^{T_n}))^2}{(vol(\Omega_j))^2}}}{\sqrt{\sum_{j=1}^{N} \frac{(\mathbf{R}_{j,turb}(\tilde{\nu}_\infty))^2}{(vol(\Omega_j))^2}}} \tag{6-5}$$

The computations were stopped after density residual was reduced by 14 orders of magnitude. For all $3D$ test cases a good initial guess was needed, so full multigrid was applied. For this we run 50 iterations each on the coarse meshes. For example, if the mesh is agglomerated into 4 levels of coarse grids, then first 50 iterations are run on the coarsest grid, then next 50 iterations we move to the second coarsest grid and after three sets of 50 iterations the multigrid cycle with all levels gets activated. For all the computations we freeze the preconditioner, which means that Block sparse matrix is built only on the first stage and not for every stage. In all computations presented in this work line symmetric Gauss-Seidel and its relaxed version is being compared.

6-1-1 NACA0012 and RAE2822

Two basic 2D geometries chosen are NACA0012 and RAE2822. NACA0012 is considered as a laminar test case whereas for RAE2822, both the mean flow and turbulent equations are solved.

- NACA0012 test case is computed with Reynolds number(Re)= 5000, Mach number(M)= 0.5 and angle of attack,$\alpha = 0^o$ (see [25]).

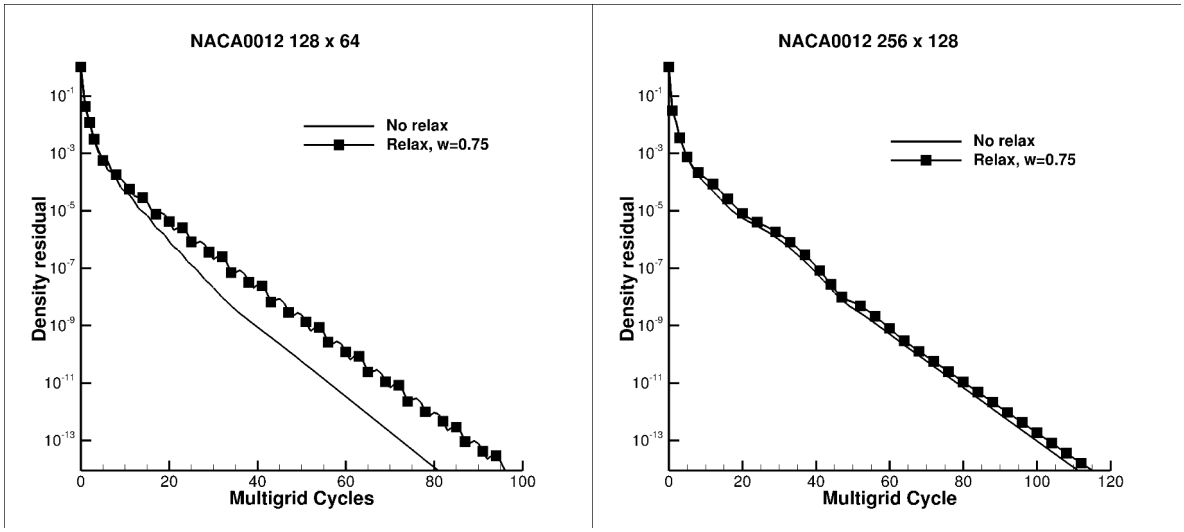
- RAE2822 test case is computed with Reynolds number(Re)= 6.5e5, Mach number(M)= 0.73 and angle of attack,$\alpha = 2.79^o$ (see [26]).

We have considered four grid levels for NACA0012. They are 128×64, 256×128, 512×256 and 1024×512. In [25] one can find a detailed investigation of this test case. The parameter settings were kept constant for all grid levels and they are presented in Table (6-1)

Parameter	Value
Number of Runge-Kutta Stages	3
CFL Number	1000
Number of Gauss Seidel Sweeps	5
Multigrid cycle	4w
Relaxation parameters (ω)	0.75

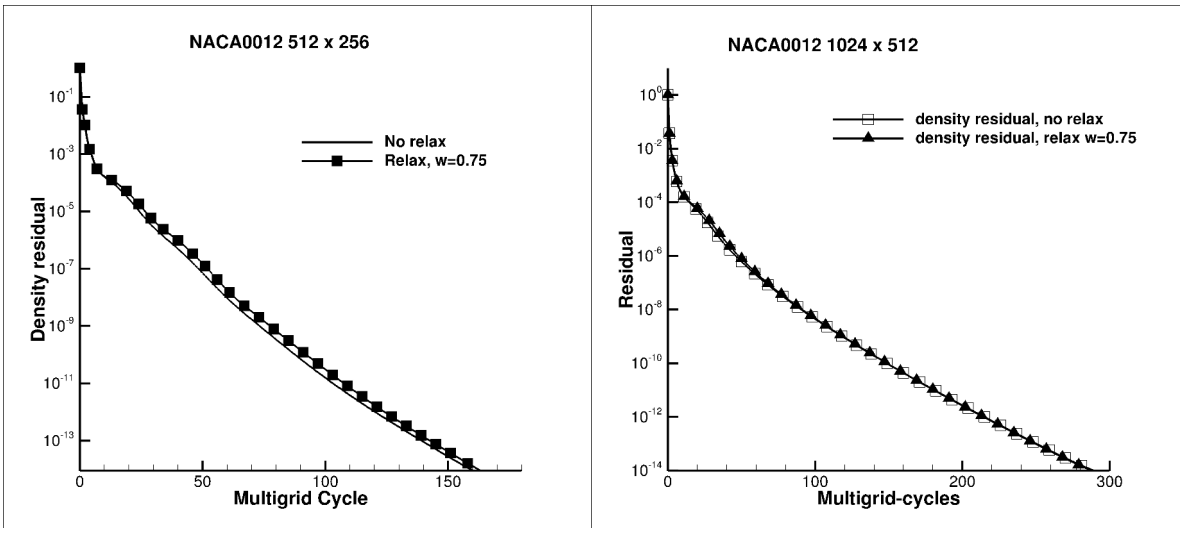
Table 6-1: NACA0012:Solver Parameters

With the settings in Table.(6-1) computations on sequence of structured NACA0012 grids are performed and the relaxed line symmetric Gauss-Siedel are found to converge in similar way compared to line symmetric Gauss-Siedel methods(LSGS). Convergence histories of comparison between relaxed and non-relaxed setting can be seen in Fig.6-1,6-2(left). The relaxation parameter is fixed at $\omega = 0.75$. When the NACA0012 mesh was computed with $\omega > 1$, the computation crashed in start up phase itself.



(a) NACA0012 128×64 Convergence History **(b)** NACA0012 256×128 Convergence History

Figure 6-1: NACA0012:128×64 and 256×128. Comparison between relaxed Gauss-Seidel and Gauss-Seidel



(a) NACA0012 512 × 256 Convergence History **(b)** NACA0012 1024 × 512 Convergence History

Figure 6-2: NACA0012:512 × 256 and 1024 × 512. Comparison between relaxed Gauss-Seidel and Gauss-Seidel

RAE2822 is a classical well known test case in the literature. Computations on three grid levels were performed and results are presented. The mesh details and the algorithmic parameters are shown in Table .6-2

Parameters	Coarse	Medium	Fine
Mesh size	320 × 64	640 × 128	1280 × 256
No. Of quadrilaterals	20 480	81 920	327 680
CFL_{init}	10	10	10
γ	1.5	1.5	1.5
$CFL_{mean,max}$	1000	1000	1000
$CFL_{turb,max}$	1000	1000	1000
Gauss-Seidel sweeps(mean/turb)	3/5	3/5	3/5
Multigrid cycles(mean,turb)	4w,4w	4w,4w	4w,4w
Number of domains	4	24	24

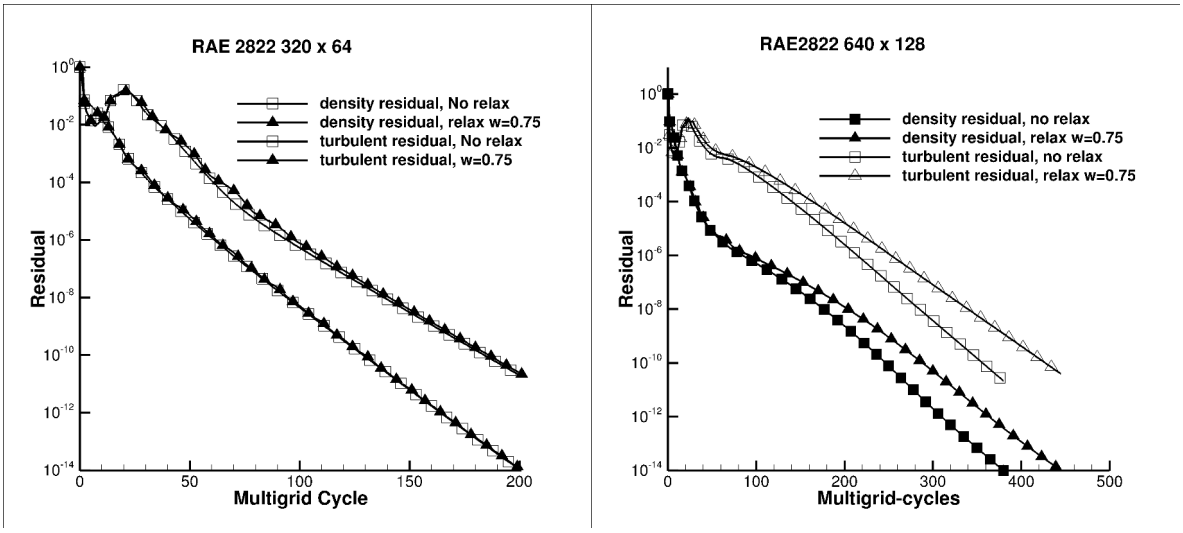
Table 6-2: RAE2822: Input Parameters and mesh data for test case RAE2822

In all computations done in this test case, the relaxation parameter is kept equal for both mean and turbulent flow equations and the value is $\omega = 0.75$. In this test case also any experiments with $\omega > 1$ did not converge. Here unlike the NACA0012 the CFL number is kept low initially and increased with every iteration like shown in Equation.(6-1). Multigrid cycle is selected as 4w for both means and turbulent flow.
Figure.(6-3 and 6-4) shows the convergence history of both density residual and turbulent residual for relaxed LSGS as well as LSGS methods. It can be seen that with the coarse mesh the convergence histories are almost the same for both relaxed and non-relaxed methods. As we move to the medium and fine mesh it can be seen that the relaxed methods take a little longer to reduce the residual to the same order of magnitude. Fig.(6-4 right) shows

the convergence history for lift coefficient and drag coefficient. This plot has results from computation done on different grid level with the relaxed method only. Fig.(6-5) shows the C_f distribution and C_p distribution for RAE2822 1280 $\times$ 256 test case along with the experimental results. Thus from this test case we can conclude that our solution methodology, after the relaxation parameter is implemented, gives converged solution if under relaxation is used.

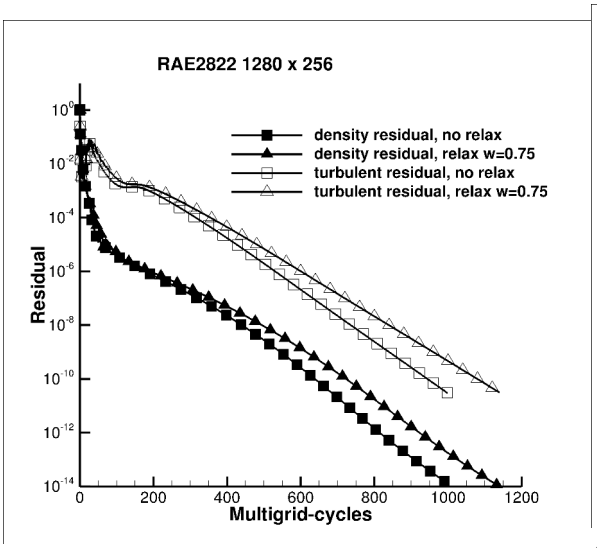
When further experiments were done by keeping the mean flow equations to be solved with a non-relaxed Gauss-Seidel method and turbulent equations to be solved with an over-relaxed Gauss-Seidel method ($\omega = 1.2$), the turbulent residual was found to be NaN(Not a Number) right from the first iteration even when the CFL number was at 10.

But when similar experiments were conducted when mean flow equations were being solved with an over-relaxed Gauss-Seidel method ($\omega = 1.2$) and turbulent flow was set to be solved with non-relaxed method, the simulations did not converge but both residual took a little longer to crash. This might not be a valuable result with respect to getting a steady-state solution but one may infer with this observation that turbulent equations are more sensitive to over relaxation.

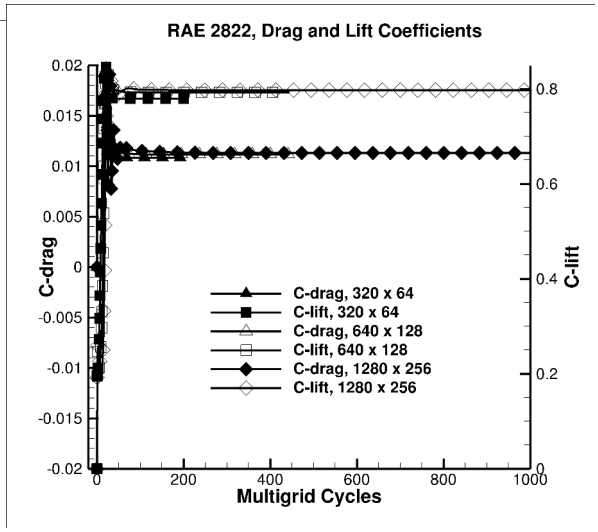


(a) RAE2822 320 $\times$ 64 Convergence History **(b)** RAE2822 640 $\times$ 128 Convergence History

Figure 6-3: RAE2822: 320 $\times$ 64 and 640 $\times$ 128. Comparison between relaxed Gauss-Seidel and Gauss-Seidel

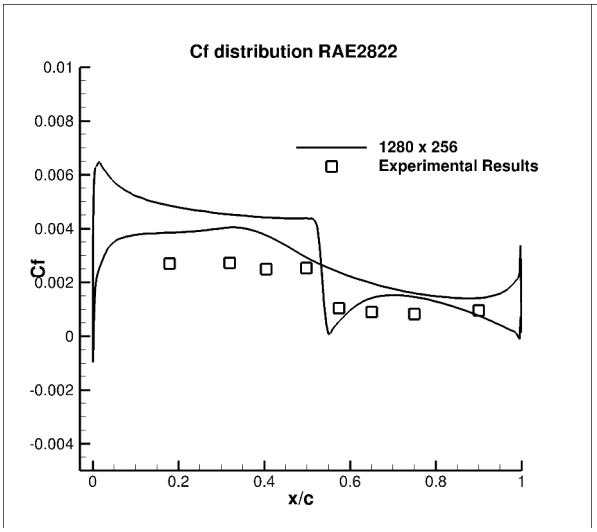


(a) RAE2822 1280 $\times$ 256 Convergence History

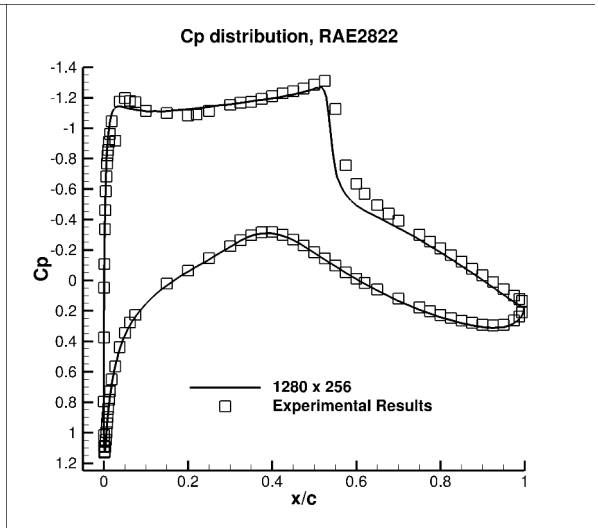


(b) RAE2822 meshes: Drag and Lift coefficients of Relaxed methods

Figure 6-4: RAE2822: Convergence histories and Drag and Lift coefficients of Relaxed methods



(a) Cf distribustion RAE2822 1280 $\times$ 256 with experimental data



(b) C_p distribution RAE2822 1280 $\times$ 256 with experimental data

Figure 6-5: RAE2822:Results

6-1-2 DPW5 CRM

As one of the $3D$ test cases we consider a sequence of meshes that were considered at the fifth AIAA drag prediction workshop (DPW5). It represents a wing body configuration and it's baseline configuration is that of the NASA common research model (CRM). The CRM is representative of a contemporary transonic commercial transport designed to cruise at Mach number , M= 0.85 and $C_L = 0.5$ at a nominal altitude of 37000 ft. ([27]). The considered original meshes are block-structured and the hybrid meshes were generated from pure hexahedral meshes. We refer to ([27]) for detailed description of the meshes.
The details of the different meshes are shown in Table.(6-3). We consider a turbulent flow over the wing-body configuration and angle of attack is different and is given in Table. 6-4 and 6-5. It is motivated by results for the target lift of 0.5. Here we present results of both hexahedral meshes and hybrid meshes of grid level L1-L4. The results seen in ([6]) shows that as the mesh gets finer the maximum CFL number ($CFL_{mean,max}$ and $CFL_{turb,max}$) is seen to be restricted. This limitation to reach higher CFL numbers is seen as a loss of robustness and versatility needed in a method. As shown in [6],with L3 Hybrid meshes there is a significant drop in maximum CFL number can be seen as we can reach only $CFLmean, max$=250 and $CFLturb, max$=50 where as the levels L1 and L2 meshes reaches levels of $CFLturb, max = 1000$. Other limitation that can be seen is the need to reduce the multigrid cycles to 3v when finer meshes like L4 or L5 are computed. It needs to be mentioned that in this work L5 mesh is not dealt with.
Thus there is an opportunity to search for means to make the solution method more robust by searching for ways to make finer meshes run at higher CFL numbers. The preconditioning techniques used till now were Gauss-Seidel methods and with respect to Gauss-Seidel methods too there is room to include a relaxation parameter. In this section we assess the performance of Algorithm (4-17) after the preconditioning technique with a new relaxed Line Gauss-Seidel method is implemented. DPW5 meshes were computed with the parameter settings mentioned in Table(6-4) for hexahedral meshes and Table(6-5) for hybrid meshes. The focus of this work is to investigate the effect, that the relaxation parameter has created on the overall solution methodology. For this purpose we consider the results in [6], where Symmetric Gauss-Seidel methods is used to solve the linear system of equation(4-31). After implementing the relaxation parameter as shown in Equations.(5-8 and 5-10), all test cases were computed with the relaxation parameter and without relaxation paramaters. The CFL number is kept small in the beginning and then increased by a factor of γ till it reaches the maximum.

Level	Hybrid Meshes		Hex Meshes	No of points
	No. of Tetrahedra	No. of prisms	No of Hexahedron	
L1	2555904	425984	638,976	660177
L2	8626176	1437696	2156544	2204089
L3	20766720	3301376	5111808	5196193
L4	69728256	11261952	17252352	17441905

Table 6-3: DPW5 CRM Mesh Data

Level	L1	L2	L3	L4
Drag coeeficient(p)	1.480233e-02	1.37283e-02	1.35147e-02	1.337936e-02
Drag coeeficient(v)	1.085389e-02	1.121219e-02	1.134309e-02	1.144732e-02
Drag coeeficient	2.565622e-02	2.494049e-02	2.485782e-02	2.482667e-02
Angle of Attack	2.364	2.193	2.158	2.133
CFL_{init}	3	3	3	2
γ	1.1	1.1	1.1	1.1
$CFL_{mean,max}$	1000	1000	1000	350
$CFL_{turb,max}$	1000	1000	1000	350
Multigrid Cycle (mean/turb)	4w/3v	4w/3v	4w/3v	3v/2v
No. Of GS Sweeps (mean,turb)	3,3	3,3	5,5	5,5
Number of domains	24	48	72	192
Time taken (in minutes)	161	237	669	1826

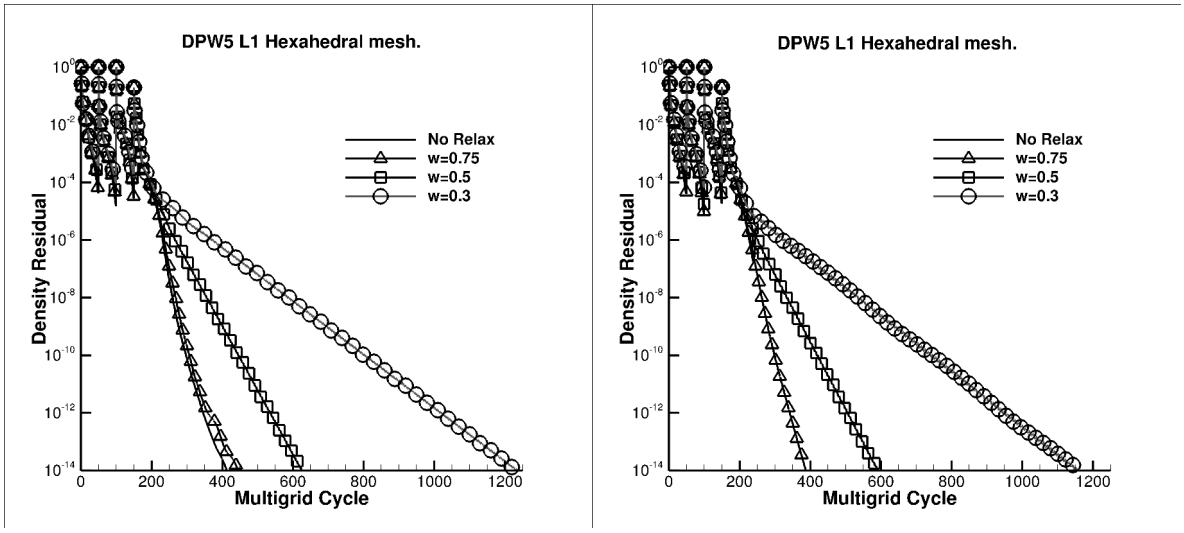
Table 6-4: DPW5 CRM: Input Parameters and computed forces for hexahedral meshes

Level	L1	L2	L3	L4
Drag coeeficient(p)	1.614527e-02	1.455193e-02	1.393222e-02	1.359777e-02
Drag coeeficient(v)	1.037894e-02	1.08365e-02	1.095987e-02	1.122124e-02
Drag coeeficient	2.652421e-02	2.538846e-02	2.489209e-02	2.481901e-02
Angle of Attack	2.314	2.1892	2.168	2.1355
CFL_{init}	3	3	3	3
γ	1.1	1.1	1.1	1.1
$CFL_{mean,max}$	1000	1000	1000	1000
$CFL_{turb,max}$	1000	1000	1000	1000
Multigrid Cycle (mean,turb)	4w/3v	4w/3v	3v/2v	4w/3v
No. Of GS Sweeps (mean,turb)	3,3	3,3	5,5	5,5
Number of domains	24	48	72	192
Time taken (in minutes)	238	581	1542	1976

Table 6-5: DPW5 CRM: Input Parameters and computed forces for hybrid meshes

In this work the goal is to explore the difference a relaxation parameter in the Gauss-Seidel methods bring to the solution methodology. Thus we need to investigate how ω values effect the convergence histories. For this purpose, a numerical experiment with L1 Hexahedral and Hybrid along with L2 Hexahedral test cases were done. The solver parameters are as shown in Table.(6-4). The computation is done for the two test cases with different ω values. Figure.(6-6 and 6-7) shows the convergence histories for L1 and L2 Hexahedral meshes when ω is kept at 0.75,0.5 and 0.3 along with Non relaxed setting for comparison. It can be seen that as ω decreases it takes longer time to reduce the density residual by 14 orders of magnitude. Thus $\omega = 0.75$ is selected as a value that will be used in all our test cases to evaluate the relaxed methods.

After implementing the relaxation parameter in the baseline code, DPW5 test cases L1-L4 were computed and the it was found all test cases converges to give a steady state solution and the computed Drag coefficients are shown in Table (6-4 and 6-5). All computation in this Section are computed with the parameter setting in Table (6-4 and 6-5).



(a) **(b)**

Figure 6-6: DPW5 L1 Hexahedral and hybrid meshes with varying relaxation parameters

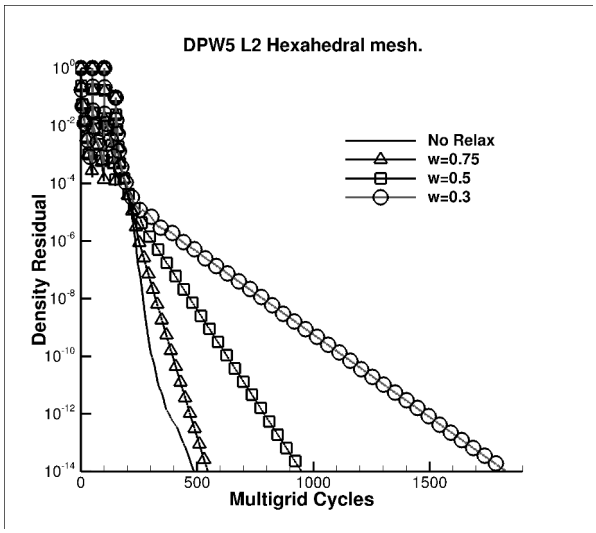
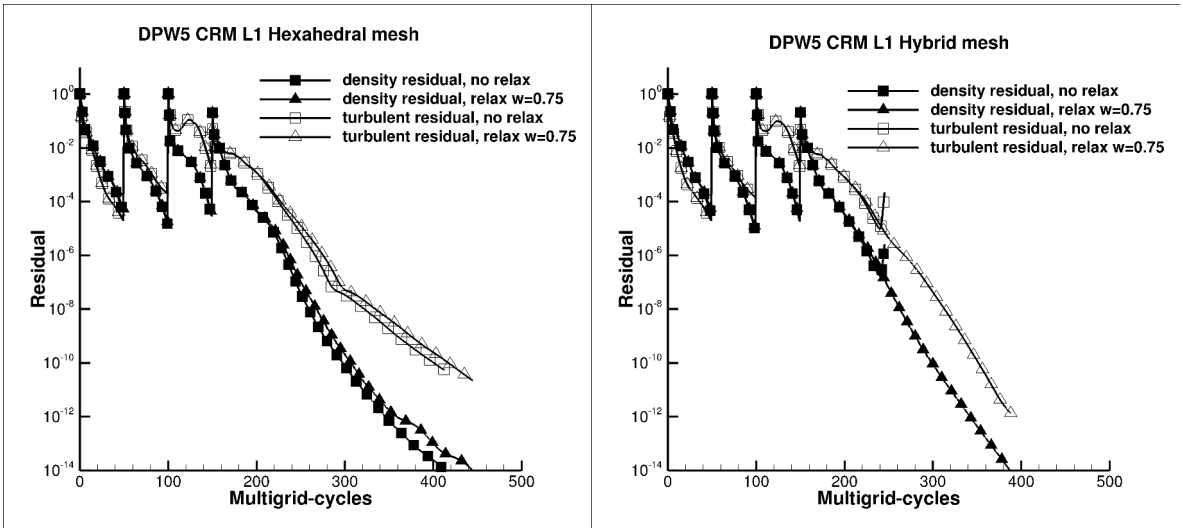


Figure 6-7: DPW5 L2 Hybrid with varying relaxation parameters

As seen in Figure(6-8), relaxed and non-relaxed methods show similar convergence behavior, while the relaxed methods taking a little longer to converge. While only the relaxed method is able to give a steady state solution while computing L1 Hybrid test cases. The non-relaxed method does not converge. However it was seen that the non-relaxed method gave a converged solution when lower multigrid levels of (mean flow:3v, turbulent flow:2v)were chosen. This shows that the relaxation criteria tends to make the solution methodology more robust as it gives a converged solution for L1 hybrid test case, when the non-relaxed method fail to converge at higher multigrid levels.



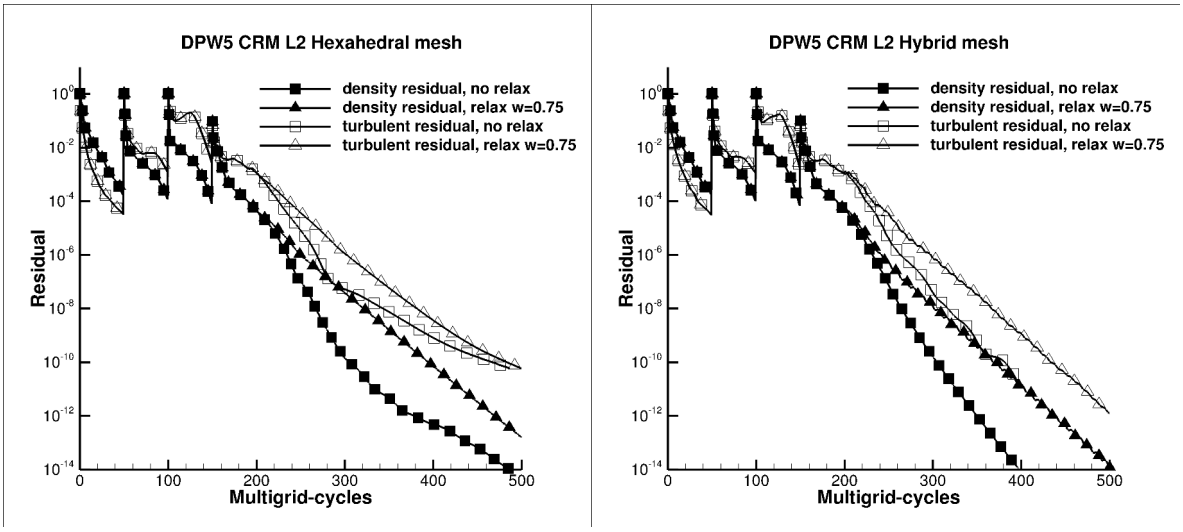
(a) DPW5 CRM L1 Hex Convergence History **(b)** DPW5 CRM L1 Hybrid Convergence History

Figure 6-8: L1 mesh: Comparison between Relaxed Gauss Seidel and Gauss-Seidel

Figure.(6-9) shows the convergence histories of L2 meshes. Here again you can see that for the Hexahedral meshes, relaxed and non-relaxed methods show similar convergence behaviour with the relaxed setting taking approximately 50 iterations more to converge. With L2 hybrid however the gap increases to around 100 iterations. The turbulent residual of relaxed methods while computing the L2 Hybrid meshes reduces further by two orders of magnitude compared to the non-relaxed setting.

Convergence histories of both density and turbulent residual of the L3 Hexahedral test case when computed with relaxed and non-relaxed residuals look similar as seen in Figure.(6-10 left). But the L4 Hybrid meshes do not give us a converged solution with the non-relaxed settings. Figure.(6-10 right) shows us the convergence history of L3 Hybrid mesh with the residuals getting reduced by 14 orders of magnitude only for the relaxed setting. Figure.(6-11) shows a closer look at the same convergence history and it can be seen that the relaxed setting crashes after the density residual is reduced by 5 orders of magnitude. This result also validates that the relaxed methods show more robustness because they run at higher CFL numbers.

Converged solution were obtained for both L4 Hexahedral and L4 Hybrid test case with both relaxed and non-relaxed setting of the solution method. Since this is the finest mesh among the DPW5 Test cases we have chosen, the gap between the relaxed and non-relaxed methods have widened as it can be seen in Figure(6-12).

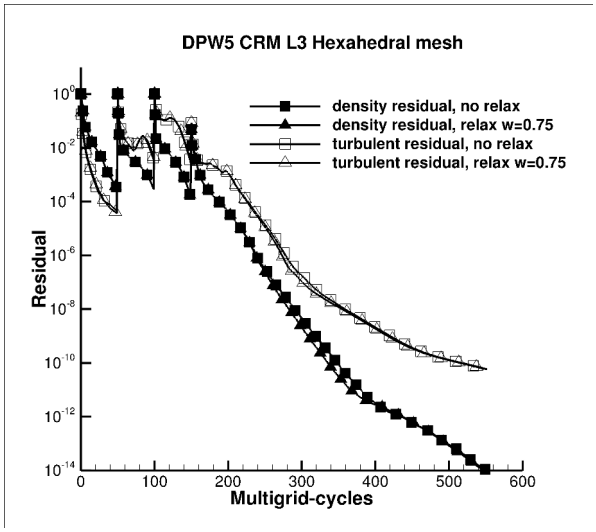


(a) DPW5 CRM L2 Hex Convergence History **(b)** DPW5 CRM L2 Hybrid Convergence History

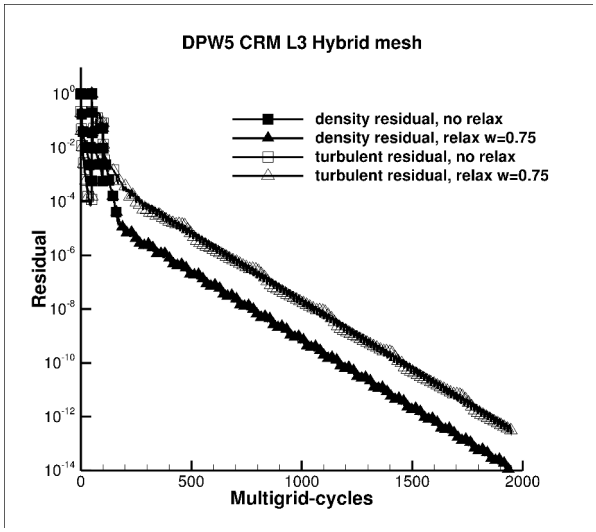
Figure 6-9: L2 mesh: Comparison between Relaxed Gauss Seidel and Gauss-Seidel

Important findings after implementing the relaxation parameter are:

- All DPW5 test cases when computed with over relaxation ($\omega > 1$) did not converge.
- Converged steady state solutions were obtained for all test cases with a relaxation parameter of ($\omega = 0.75$).
- With non-relaxed methods there was loss in robustness found as we moved onto the finer meshes as seen in [6] where the L3 Hybrid test case shows a restriction of $CFL_{mean,max} = 200$ and $CFL_{turb,max} = 50$ whereas CFL numbers for other test cases are let to increase till 1000. Fig.6-10(b) shows that by using LSGS the computations doesn't converge but relaxed-LSGS gives a converged result when the CFL number is let to increase till 1000. Fig.6-11 gives a closer look.
- The L2 Hybrid test case also did not converge when LSGS was used with multigrid levels 4w for mean and 3v for turbulent flow, where we were able to find a converged solution when mutigrid levels were reduced to 3v for mean and 2v for turbulent. But when computed with relaxed LSGS, there was no need to reduce the multigrid levels as the test case converged even with higher multigrid levels. Fig. 6-8(right) shows that with the given input parameter setting for L1 Hybrid test case in Table (6-5), converged solution is got only with relaxed LSGS method.
- Convergence histories of all test cases shows us that Multigrid(MG) cycles needed to reduce the density residual by 14 orders of magnitude is higher after we introduce the relaxation parameter. This increase in MG-cycles to converge is amplified as we move on to finer meshes. The gap between number of MG cycles to converge for relaxed and non-relaxed methods is found to increase as the mesh gets finer, as seen in Figure.6-12 (left and right). In Figure.(6-9) also this can be seen.



(a) DPW5 CRM L3 Hex Convergence History



(b) DPW5 CRM L3 Hybrid Convergence History

Figure 6-10: L3 mesh: Comparison between Relaxed Gauss Seidel and Gauss-Seidel

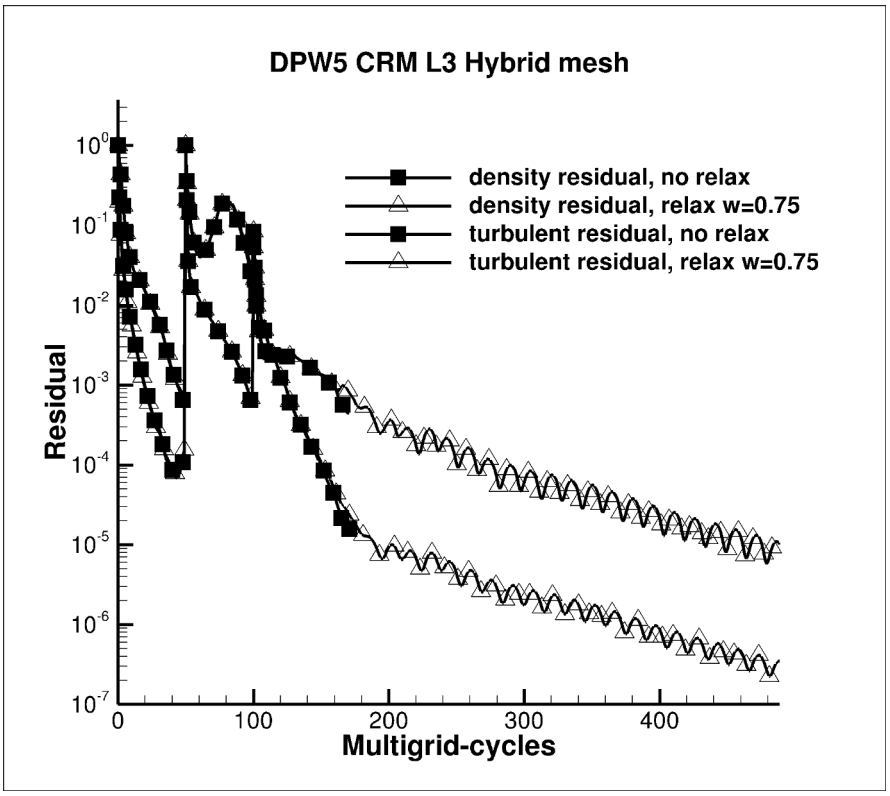
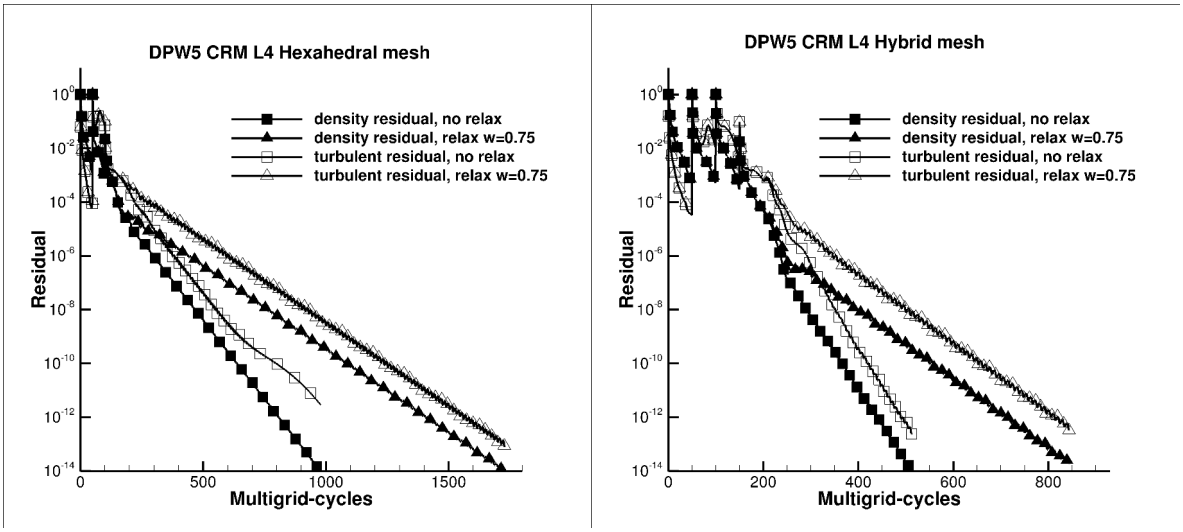


Figure 6-11: DPW5 CRM L3 Hybrid Convergence History (closer look)



(a) DPW5 CRM L4 Hex Convergence History **(b)** DPW5 CRM L4 Hybrid Convergence History

Figure 6-12: L4 mesh: Comparison between Relaxed Gauss-Seidel and Gauss-Seidel

6-1-3 NASA TRAP Wing

The final test case considered is the NASA Trap Wing considered at the first AIAA High-Lift workshop. Here a sequence of three meshes are considered. Details about the three meshes- Coarse, Medium and Fine, are given in Table (6-6)
Computations were done for the angle of attack of 28°. For all the three meshes density residual reduction of 14 orders of magnitude was possible when the test cases were computed with a relaxation parameter of $\omega = 0.75$. Convergence histories for all meshes in comparison to the simulations without the implementation of relaxation parameter are shown in Figures (6-13, 6-14).

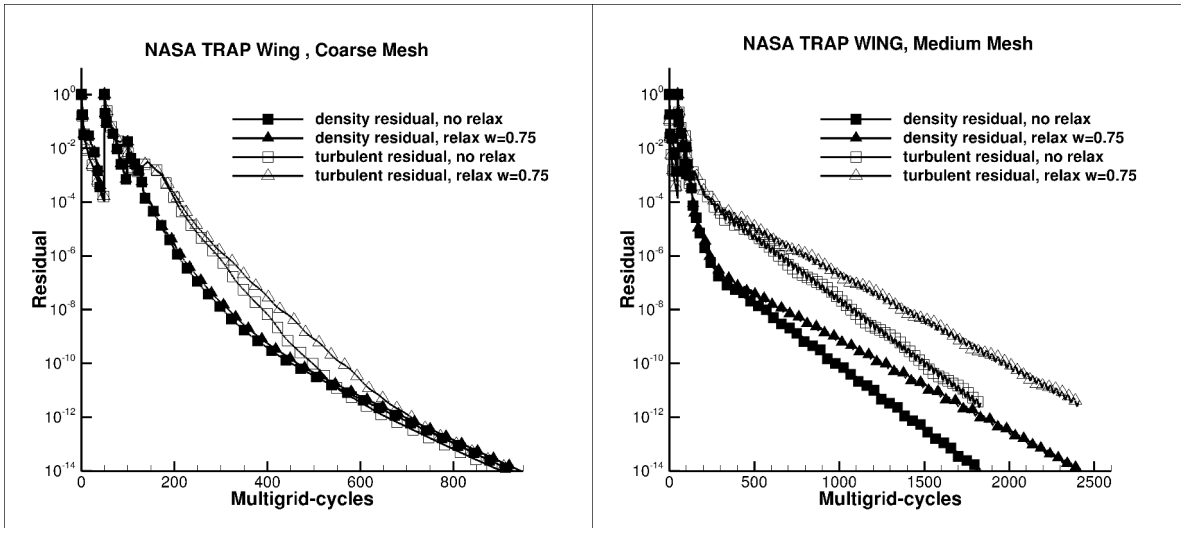
Mesh	No. of points	No. Of elements
Coarse	3727008	10169092
Medium	11047965	380017477
Fine	32445391	127443165

Table 6-6: NASA Trap Wing: Mesh data

Level	Coarse	Medium	Fine
Drag coefficient(p)	6.578072e-01	6.697461e-01	6.769523e-01
Lift coefficient(p)	2.813589	2.900736	2.933248
Angle of Attack	28°	28°	28°
CFL_{init}	3	3	3
γ	1.1	1.1	1.1
$CFL_{mean,max}$	3	3	3
$CFL_{turb,max}$	1000	1000	1000
Multigrid Cycle (mean,turb)	3v,3v	3v,3v	3v,3v
No. Of GS Sweeps (mean,turb)	5,5	5,5	5,5
Number of domains	72	192	384

Table 6-7: NASA Trap Wing: Input Parameters and computed forces

The determined force coefficients of lift and drag are given in Table (6-7). C_p distributions at 58% and 98% wing section is shown in Figure.(6-15). The results computed are compared to the experimental results. The convergence histories when the three meshes were computed after the relaxation parameter was implemented are shown in Figures.(6-13 and 6-14). In the coarse mesh there is no much difference found with respect to iterations needed for density residual to be reduced to 14 orders of magnitude. But as we move on to the medium and finer meshes it is found that the relaxed methods take longer to reach the same order of density residual reduction compared to the non-relaxed methods. It needs to be mentioned that NASA TRAP wing was considered to show that the newly implemented relaxed method is practical enough to be deployable in challenging computations. Since this sequence of meshes are computationally expensive, an attempt to increase the CFL_{max} was not done.



(a) **(b)**

Figure 6-13: NASA TRAP Wing: Coarse and Medium mesh convergence histories

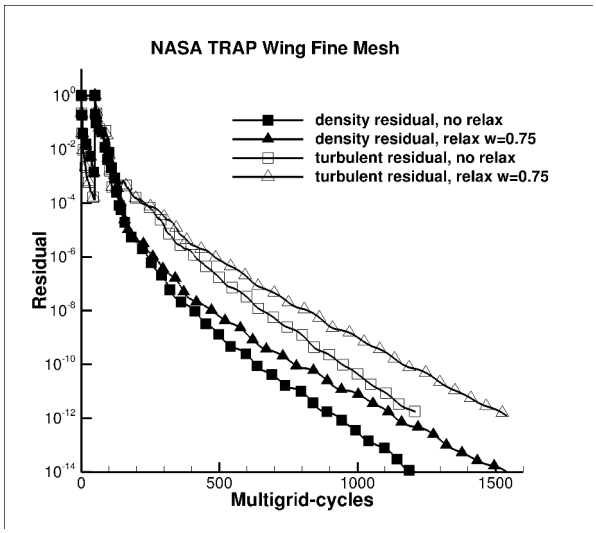
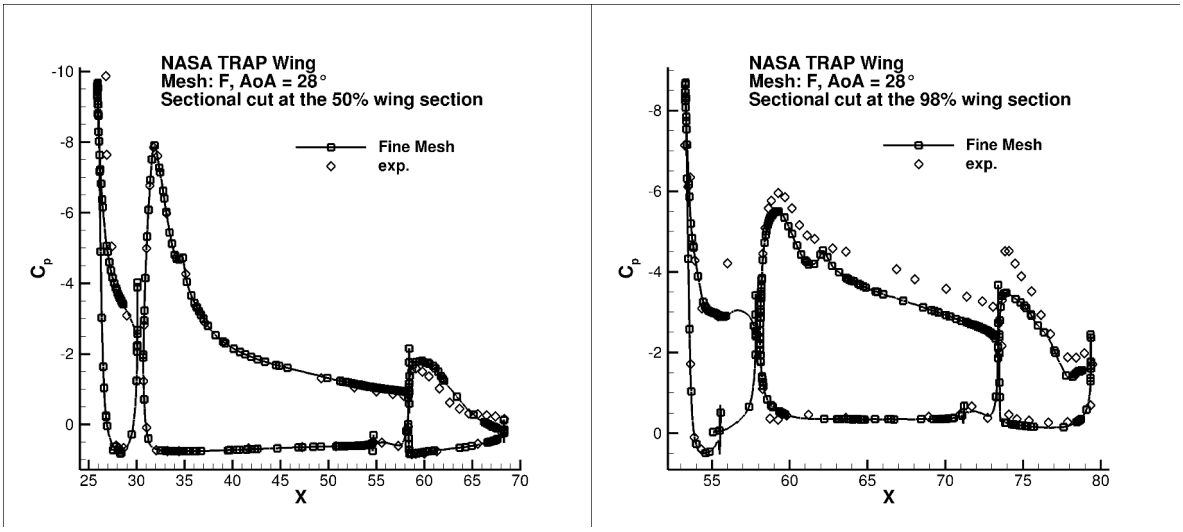


Figure 6-14: NASA TRAP Wing:Fine mesh convergence histories



(a) **(b)**

Figure 6-15: NASA TRAP Wing: Computed C_p and C_f distribution at 50% and 98% wings section

6-2 Truncation Criteria

At every Runge-Kutta stage of the smoother algorithm, Equation.(4-31) needs to be solved. This is solved iteratively and number of Gauss-Seidel sweeps needs to be user-defined. As discussed in the introduction, a CFD code needs to be versatile enough to give a steady-state solution without the user-selecting many solver parameters. In this section by implementing a truncation criteria, we are moving towards eliminating a parameter that solver needs as an input. It can be seen in DPW5 CRM test cases Table.(6-4 and 6-5) that some cases the GS sweeps were chosen as 3 and for some it was chosen as 5. The issue with this value being low is that it may take longer for the computation to converge and if this value is large, it may lead to over solving of the linear system. This linear system is just an update to the outer-non linear loop, thus it needn't be solved to maximum accuracy. For a linear system $A \cdot x = b$, a normalized linear residual can be computed as

$$Linear\ residual = \frac{\|Ax_n - b\|}{\|b\|} \tag{6-6}$$

where x_n is the vector of unknowns at n^{th} iteration. The linear residual is normalized with respect to the outer non-linear loop. This linear residual is calculated at the end of every linear iteration to solve the Equation.(4-31) with an iterative method, for example Relaxed Gauss Seidel method as shown in Equation (5-8). This linear iteration was truncated as soon as the linear residual was reduced by an order of 1.
All the DPW5 test cases were computed with LSGS method to solve the inner-linear loop and the linear loop was truncated after the linear residual reduced by one order of magnitude. The input parameters were taken same as the ones specified in Table.6-4 and 6-5. The convergence history for these computations can be seen in Fig.(6-16 left) and all test cases converged. The values of all Coefficients were found to be exactly the same as the computations, where truncation criteria wasn't implemented. One may conclude with this result that solving the inner

linear loop to the more accuracy is not going to add any value to the steady state solution. Time needed to converge with the truncation criteria for L1 hexahedral mesh is 160 minutes,

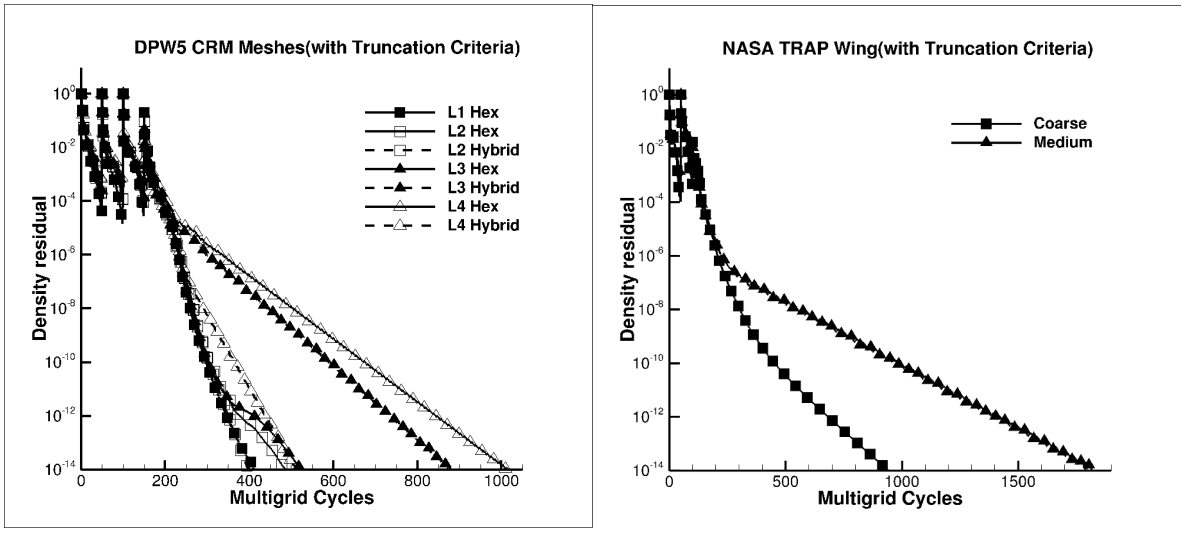


Figure 6-16: Truncation criteria: Convergence histories for DPW5 and NASA TRAP Wing meshes

L2 Hexahedral is 383 minutes, L2 Hybrid is 489 minutes. For the NASA TRAP wing the computation with the truncation criteria converged in 483 minutes compared to 527 minutes for the computation without the truncation criteria. To demonstrate the perils of letting the user to decide the number of Gauss-Seidel sweep can be seen from Figure(6-17 right). Here the L4 Hexahedral test case is computed first with Gauss-Seidel sweeps = 3 and second, after implementing the truncation criteria. Since the number of sweeps was chosen as a low value, it leads to the computation to take longer to converge. This is a clear case of under solving.

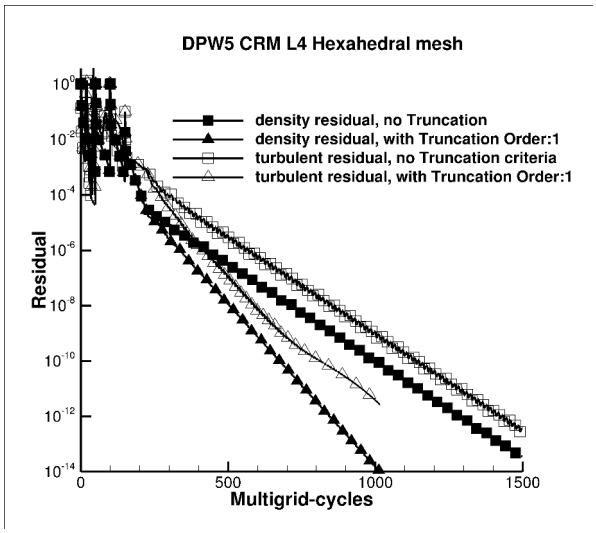


Figure 6-17: DPW5 CRM L3 Hexahedral mesh:Comparison between before and after implementing truncation criteria

6-2-1 Inner Linear loop analysis

After Equation.(6-6) is implemented numerical experiments were conducted by keeping the Number of Gauss-Seidel sweeps as high as 100 and the linear inner loop was truncated after the inner linear residual reduced by one order. After this we were interested to know, how many GS sweeps does the inner linear loop need to reduce the residual by 1/10. Figure.(6-18 and 6-19) shows how the No.of GS sweeps needed by inner linear loop changes with respect to the outer non-linear iteration.

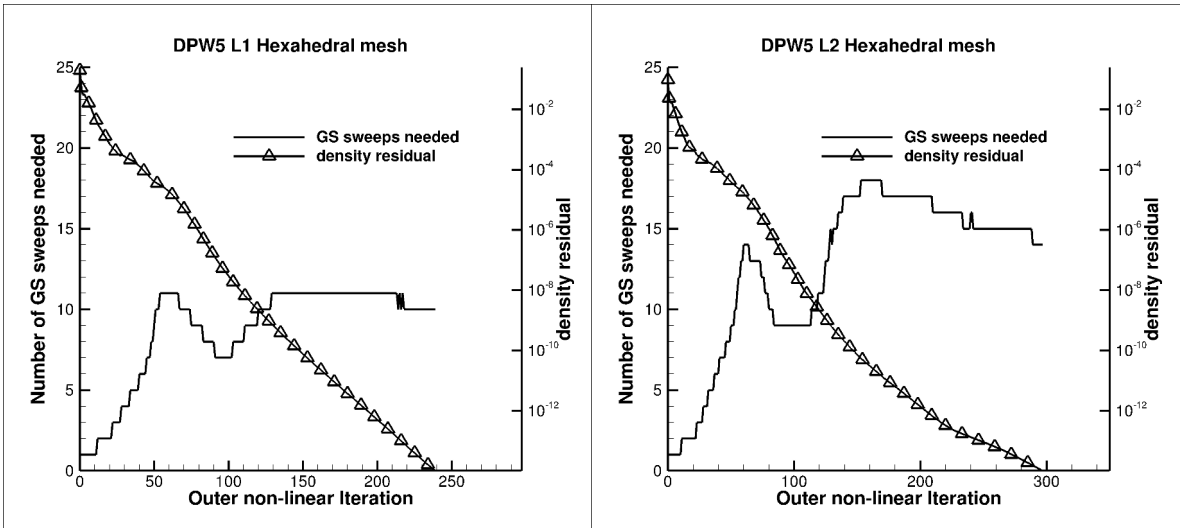


Figure 6-18: DPW5 CRM:No. of inner linear GS sweep needed to reduce linear residual by one order of magnitude vs Outer non-linear loop

It can be seen from Figure.(6-18) that when L1 hexahedral mesh was computed with setting which allowed the inner linear system (Equation-4-31) to get solved till the residual dropped by one order of magnitude. The number of linear Gauss-Seidel sweeps needed for residual to drop by 0.1, is seen to increase initially and after the computation has reached asymptotic convergence it constantly lies in a range. If you take L1 hexahedral mesh, this value is somewhere around 10 GS sweeps. Same can be seen with L2 hexahedral and L3 Hybrid meshes as the no. of inner linear GS sweeps needed for residual to drop by an order of magnitude lies between 14-17 after it reaches asymptotic convergence. It needs to be mentioned that we start monitoring this only after the complete multigrid cycle is activated and not during the first set of 50 iterations, which is computed to get a good initial guess.

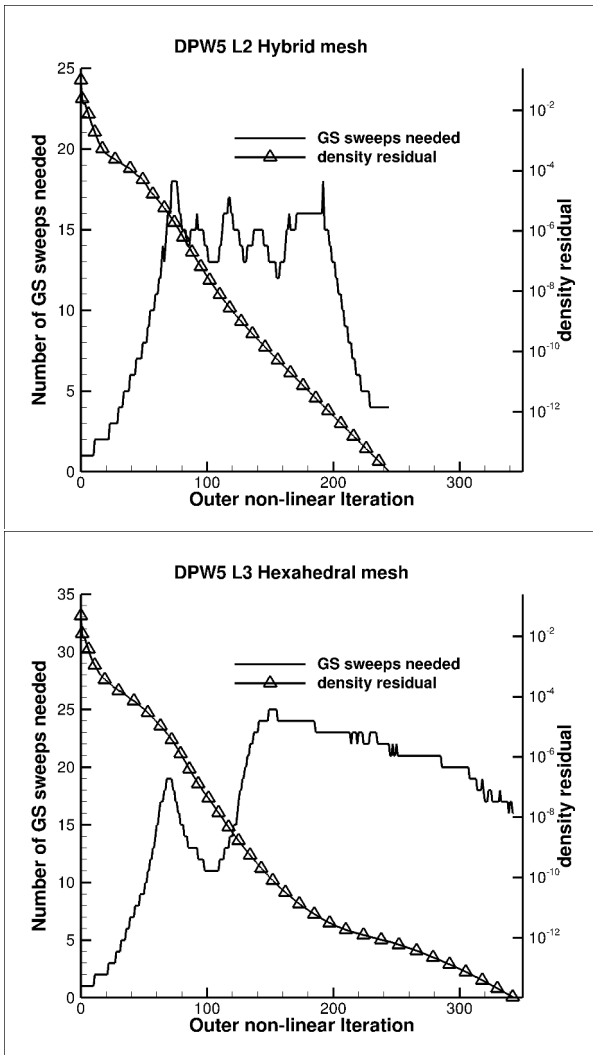


Figure 6-19: DPW5 CRM:No. of inner linear GS sweep needed to reduce linear residual by one order of magnitude vs Outer non-linear loop

6-2-2 Computer-Aided analysis

Eigenvalue analysis is performed on the Algorithm.(4-17) in order to understand it's properties. In this work we calculate the eigenvalue spectrum like it was done in [6]. With this goal Equation.(3-14) is replaced by its linearized counterpart $\frac{d}{dt}\mathbf{W} = -\mathbf{M}^{-1}\mathbf{A}\mathbf{W}$ where $\mathbf{A} := \frac{\partial \mathbf{R}}{\partial \mathbf{W}}$. Algorithm.(4-17) becomes

$$\begin{aligned}
\mathbf{W}^{(0)} &:= \mathbf{W}^{T_n} \\
\mathbf{W}^{(1)} &:= \mathbf{W}^{(0)} - \alpha_{21}\mathbf{P}^{-1,app}\mathbf{A}\mathbf{W}^{(0)} \qquad (6\text{-}7) \\
&\vdots \\
\mathbf{W}^{(s)} &:= \mathbf{W}^{(0)} - \alpha_{s+1,s}\mathbf{P}^{-1,app}\mathbf{A}\mathbf{W}^{(s-1)} \qquad (6\text{-}8) \\
\mathbf{W}^{T_{n+1}} &:= \mathbf{W}^{(s)}
\end{aligned}$$

where $\mathbf{P}$ is the preconditioner. This algorithm can be written by a polynomial expression

$$\begin{aligned}
\mathbf{W}^{T_{n+1}} &= q_s(\mathbf{P}^{-1,app}\mathbf{A})\mathbf{W}^{T_n} \\
q_s(z) &= 1 + \sum_{j=1}^{s}(-1)^j z^j \prod_{i=s-j+1}^{s} \alpha_{i+1,i}. \qquad (6\text{-}9)
\end{aligned}$$

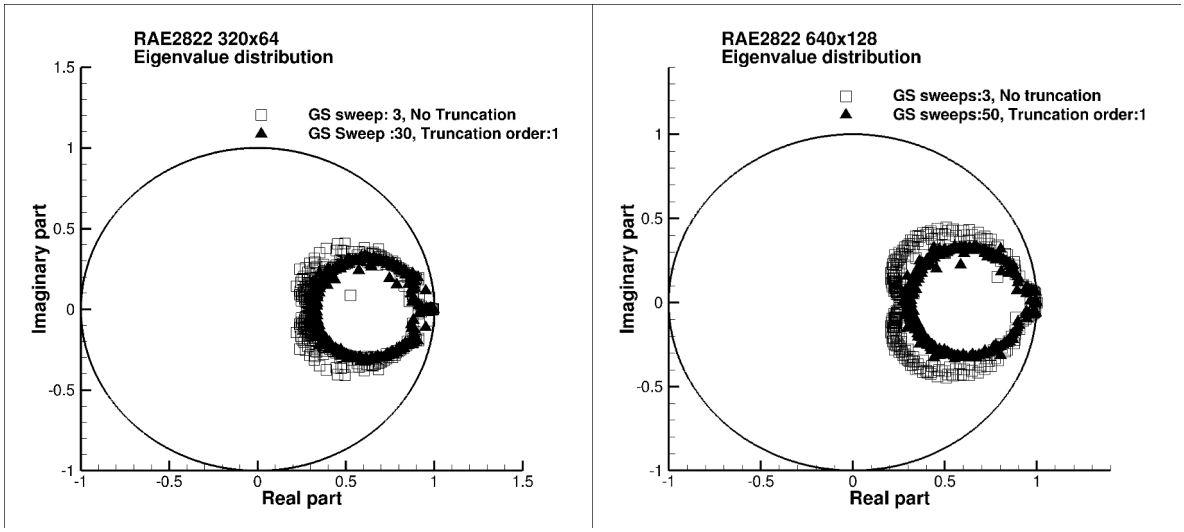
We denote the eigenvalue by λ_i and corresponding normalized eigenvector as $\mathbf{v}_i$ of the matrix $\mathbf{P}^{-1,app}\mathbf{A}$. Then we have

$$\begin{aligned}
\mathbf{P}^{-1,app}\mathbf{A} &= \mathbf{V}\Lambda\mathbf{V}^{-1} \\
\Lambda &= \mathrm{diag}(\lambda_i) \qquad (6\text{-}10) \\
\mathbf{V} &:= (\mathbf{v}_1, \mathbf{v}_2, ...) \qquad (6\text{-}11) \\
\mathbf{W}^{T_{n+1}} &:= \mathbf{V}q_s(\Lambda)\mathbf{V}^{-1}\mathbf{W}^{T_n}. \qquad (6\text{-}12)
\end{aligned}$$

To show that Algorithm.(6-7) converges to the unique limit it is necessary and sufficient [23] to show that the spectral radius of $\mathbf{P}^{-1,app}\mathbf{A}$ satisfies, $\rho(q_s(\mathbf{P}^{-1,app}\mathbf{A})) = \rho(q_s(\Lambda)) < 1$

Now, to find the eigenvalues of $\mathbf{P}^{-1,app}\mathbf{A}$ the Generalized minimal residual (GMRES) method is used(see [14]). The eigenvalue distribution for an approximate solution of the nonlinear problem is computed only after its density residual has been reduced by 14 orders of magnitude. These eigenvalues govern the asymptotic convergence. After it has converged we introduce a small perturbation and assess how the method behaves, thus they can not be used to assess start-up problems.

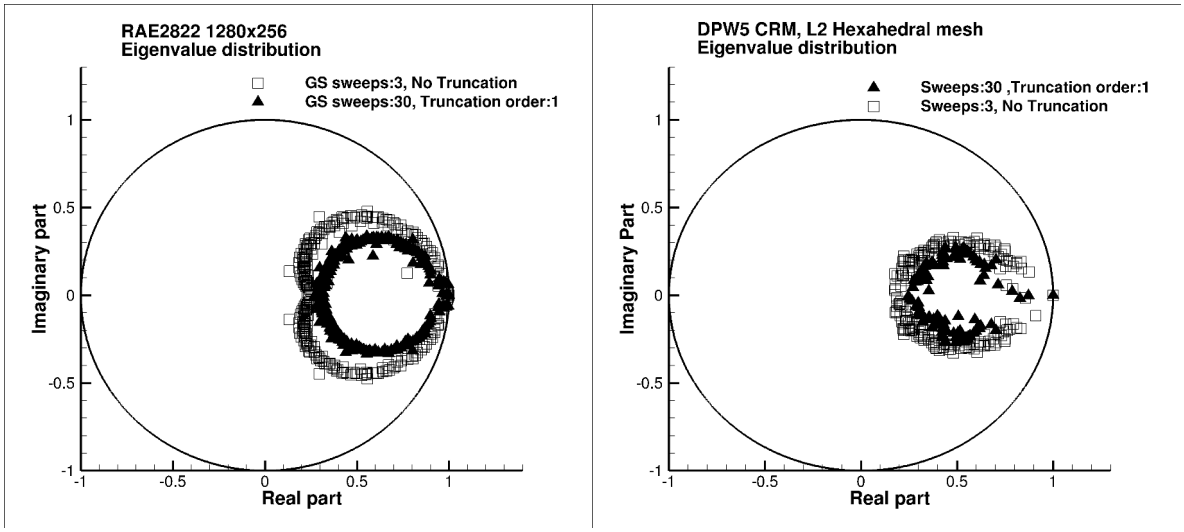
In order to see how the implementation of truncation criteria has changed the solution methodology, we take the converged steady state solutions and do the eigenvalue analysis. Firstly, by setting the Gauss-Seidel sweep to a certain value and without the truncation criteria. Secondly, by increasing the number of Gauss-Seidel sweeps to a large value and letting the truncation criteria stop the iteration. In all our experiments the truncation order is set as 1, which means the inner linear loop truncates as soon as the linear residual drop by a magnitude of 1. In Figures (6-20,6-21,6-22 and 6-23) it can be seen that the setting in which the truncation criteria is set has the eigenvalues move closer towards each other compared to the setting with the fixed Gauss-Seidel sweeps. This indicates that a truncation criteria for the inner linear loop makes the solution methodology more robust.



(a) RAE2822 320 × 64 Eigenvalue spectrum, with and without truncation criteria

(b) RAE2822 640 × 128 Eigenvalue spectrum, with and without truncation criteria

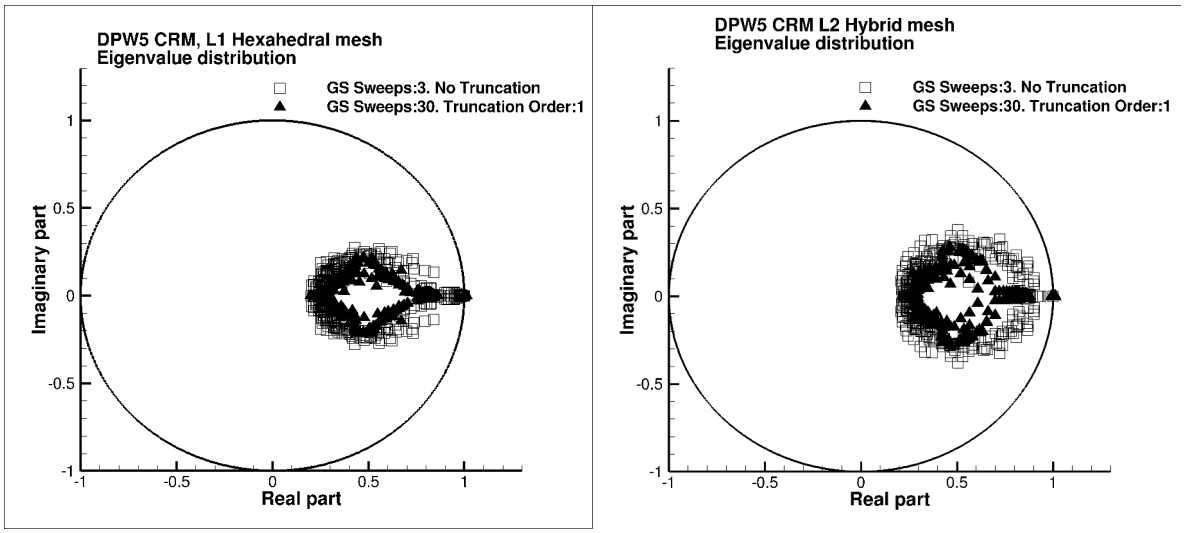
Figure 6-20: Eigenvalue spectrum



(a) RAE2822 1280 × 256 Eigenvalue spectrum, with and without truncation criteria

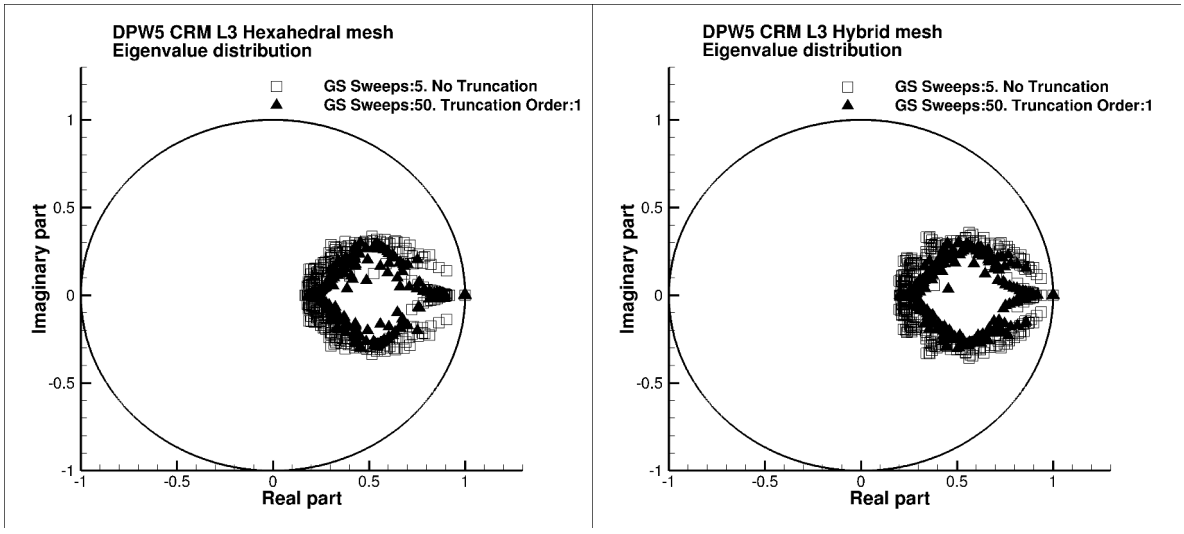
(b) DPW5 CRM L2 Hexahedral: Eigenvalue spectrum,with and without truncation criteria

Figure 6-21: Eigenvalue spectrum



(a) **(b)**

Figure 6-22: DPW5 CRM L1 Hexahedral and L2 Hybrid: Eigenvalue spectrum,with and without truncation criteria



(a) **(b)**

Figure 6-23: DPW5 CRM L3 Hexahedral and Hybrid: Eigenvalue spectrum,with and without truncation criteria

Chapter 7

Conclusion

An agglomerated FAS multigrid scheme with an implicit multitage Runge-kutta smoother to approximately solve the RANS equation and the transport type equation of Spallart Allmaras model in a loosely couple manner, needs to solve a linear system of equation for every stage of Runge-kutta method. Different iterative techniques are used to solve this linear system of equation, which is ill-conditioned and stiff. In this work we have successfully implemented relaxation parameter in Gauss-Seidel methods and tested the algorithm to different kinds of grids and grid families. In this scope of work we are interested in obtaining the steady state solution.

- Over relaxation in Relaxed Gauss-Seidel methods: In case they were used to solve the inner linear loop in the implicit smoother algorithm within the FAS multigrid, computation did not converge for any test Cases used in this thesis.
- Under relaxation in Relaxed Gauss-Seidel methods: In case they were used to solve the inner linear loop in the implicit smoother algorithm within the FAS multigrid, it showed promising results as density residual of all test cases could reach machine accuracy.
- When compared with the normal Gauss-Seidel the relaxed methods were able to run with L2 Hybrid and L4 Hybrid meshes at $CFL_{max} = 1000$, when the former failed to produce converged results. This shows an improved robustness in the solution methodology when relaxed methods were implemented.
- In all test cases relaxed Gauss-Seidel methods took longer to converge compared to standard Gauss-Seidel. For the 2D meshes and coarser meshes on DPW5 test cases the gap between the iterations require to converge were not significant. But as the meshes got finer the gap was seen to widen.
- As an attempt to eliminate a solution parameter: $No: \ of\ Gauss-Seidel\ sweeps$, a linear truncation criteria was introduced. All test cases considered in this thesis were found to produce converged results when the inner linear loop was truncated after the residual dropped by one order of magnitude. Comparing the coefficient result of test

cases with truncation criteria and without truncation criteria it is seen that the values are same. The inner linear loop just needs to be approximated.

- Eigenvalue plots show improvement in robustness when truncation criteria was implemented. In these experiments a higher Gauss-Seidel sweeps were considered and the inner linear loop was truncated after the residual dropped by one order of magnitude.

Even though the relaxed Gauss-Seidel methods may take longer to converge, they have demonstrated more robustness with respect to higher CFL numbers and multigrid cycle. A suitable value for the relaxation parameter ω is not straightforward and for sure case dependent. In the numerical experiments performed in this thesis often a value between $\omega \in \left[\frac{1}{2}, \frac{9}{10}\right]$ was successful. Even though ω is an additional parameter, it is seen from the results that it is more robust, as it can compute finer meshes at higher CFL numbers as shown, in comparison with existing methods.
The practice of setting the number of Gauss-Seidel sweeps as a solver parameter makes the solver less adaptive and versatile. For this purpose a linear truncation criteria is implemented and eigenvalue spectrum show results that support the claim it has made the scheme more stable.

Bibliography

[1] M. B. Niles A.Pierce, "Preconditioned multigrid methods for compressible flow calculations on structured meshes," *Journal of Computational Physics*, 1997.

[2] C.-C. Rossow, "Efficient computation of compressible and incompressible flows," *Journal of Computational Physics*, 2007.

[3] C.-C. R. R.C.Swanson, E.Turkel, "Convergence acceleration of runge-kutta schemes for solving the navier-stokes equations," *Computers and Fluids*, 2007.

[4] C.-C. R. R.C.Swanson, "An efficient solver for rans equation and a one-equation turbulence model," *Computers and Fluids*, 2011.

[5] S. Langer, "Investigation and application of point implicit runge-kutta methods to invicid flow problem," *International Journal for Numerical Methods Fluid*, 2012.

[6] S. Langer, "Agglomeration multigrid methods with implicit runge-kutta smoothers applied to aerodynamic simulations on unstructured grids," *Journal of Computational Physics*, 2014.

[7] S. Langer, "Heirarchy of preconditioning techniques for the solution of the navier-stokes equations discretized by 2nd order unstructured finite volume methods," *in: ECCOMAS 2012 Conference, in:Conference Proceeding Series*, 2012.

[8] A. L. Niles A.Pierce, Michael B.Giles, "Accelerating three-dimentional navier-stokes calculations," *13th AIAA Computational Fluid Dynamics conference, in:Conference Proceeding Series, vol. 1997-1953*, 1997.

[9] J. N. Peter Eliasson, Per Weinerfelt, "Application of a line-implicit scheme on stretched unstructured grids," *47th AIAA Aerospace science meetings, in:Conference Proceeding Series, vol. 2009-0163, AIAA*, 2009.

[10] S. Langer, "Application of line implicit method to fully coupled system of equation for turbulent flow problems," *International Journal for Computational Fluid Dynamics*, 2013.

[11] B. Leer, "Characteristic time-stepping or local preconditioning of euler equations," *AIAA*, 1911.

[12] S. P.R.Spalart, "A one equation turbulence model for aerodynamic flows," *AIAA Computational Fluid Dynamics conference, in:Conference Proceeding Series, vol. 1992-439*, 1992.

[13] P. R. Steven.R.Allmaras, Forrester T.Johnson, "Modification and clarification for the implementation of the spalart-allmaras turbulence model," *in:International Conference on Computational fluid dynamics 7, Hawaii, in Conference Proceeding Series, vol. ICCFD7-1902*, 2012.

[14] S. Langer, "Computer aided analysis of preconditioned multistage runge-kutta methods applied to solve the compressible reynolds averaged navier-stokes equations,"

[15] J. Blazek, "Computational fluid dynamics: Principes and application," 2005.

[16] E.Turkel, "Improving the accuracy of central difference schemes," *11th International conference on Numerical Methods in Fluid Dynamics*, 1989.

[17] E. R.C.Swanson, "On central difference and upwind schemes," *Journal for computational Physics*, 1992.

[18] J. N. Magnus Svard, "Stability of finite volume approximations for the laplacian operator on quadrilateral and triagular grids," *Applied Numerical Mathematics*, 2004.

[19] J. N. Magnus Svard, Jing Gong, "Stable artificial dissipation operators for finite volume schemes on unstructured grids," *Applied Numerical Mathematics*, 2006.

[20] D. J. Mavriplis, "Multigrid techniques for unstructured meshes," *ICASE Report 95-27*, 1995.

[21] A. S. Ulrich Trottenberg, C.W.Oosterlee, "Multigrid, academic press, orlando," 2001.

[22] G. K. Irene Moulitsas, "Multilevel algorithms for generating coarse grids for multigrid methods," *in:Greg Johnson,SC,ACM,p.45*, 2001.

[23] Y.Saad, "Iterative methods for sparse linear systems," *International Thomson Publishing*, 1996.

[24] G. H. Golub and C. F. van Loan, "Matrix computations," 1983.

[25] R.Swanson and S. Langer, "Steady state laminar flow solutions for naca0012 airfoil," *Computers and fluids*, 2016.

[26] M. P.H.Cook, M.A.McDonald, "Aerofoil rae2822 pressure distribution and boundary layer and wake measurements," *AGARD-AR,138*, 1979.

[27] J. C.Vassberg, "A uniform baseline grid about the common research model wing-body for the fifth aiaa cfd drag prediction workshop," *in:Proc.29th AIAA Applied aerodynamics conference,Honolulu*, 2011.

DLR-IB-AS-BS-2016-276

Manickam Somasundaram

Verteiler:

Institutsbibliothek	1 Exemplar
Verfasser	2 Exemplare
Institutsleitung	1 Exemplar
Abteilungsleiter	1 Exemplar
Deutsche Bibliothek in Frankfurt/Main	2 Exemplare
Niedersächsische Landesbibliothek Hannover	1 Exemplar
Techn. Informationsbibliothek Hannover	1 Exemplar
Zentralbibliothek BS	1 Exemplar
Zentralarchiv GÖ	1 Exemplar
Reserve	5 Exemplare