Implementation and Validation of the Spalart-Allmaras turbulence model

Selim Belhaouane

Master of Engineering

Department of Mechanical Engineering

McGill University
Montreal, Quebec
June 2017

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Engineering

Abstract

In the field of computational fluid dynamics (CFD), the most popular method to resolve the effects of turbulence in a numerical simulation is to employ the Reynolds-averaged Navier-Stokes Equations (RANS) and capture the turbulence effects using a turbulence model. Among the many turbulence models that exist, the Spalart-Allmaras (S-A) model has proven to be reliable for attached and moderately separated flows. In this work, the S-A model has been implemented in two CFD frameworks : an in-house academic structured finite-volume cell-centered code and MAYA HTT's commercial unstructured vertex-centered code. The implementation details are presented along with validation results on NASA Turbulence Model Resource benchmark cases and on aerodynamic cases.

Abrégé

Dans le domaine de la mécanique des fluides numérique (MFN), l'approche la plus populaire pour prédire le comportement d'un écoulement turbulent est de faire usage à une méthode de moyennage temporel des équations de Navier-Stokes. Les effets turbulents sont donc pris en compte par un modèle de turbulence. Dans le cadre de cette thèse, le modèle conçu par Spalart et Allmaras, lequel est reconnu pour sa fiabilité pour les écoulements externes, est implémenté dans deux logiciels MFN, l'un étant académique et l'autre étant commercial. Les détails de l'implémentation ainsi que des résultats de validation sont présentés.

Acknowledgements

I would first like to thank Maya Heat Transfer Technologies, specifically Chris Jackson and Christian Ruel, for sponsoring my Master's degree. I am also grateful for the funds received by the FQRNT and CRSNG as part of the Industrial Innovation Scholarship.

I would also like to express my gratitude towards my academic supervisor, Prof. Siva Nadarajah, for his patience and immense knowledge. His continued guidance has helped me stay on the right path throughout the last two years.

I have also benefited greatly from the support and knowledge of my colleagues at Maya, namely Peyman Khayatzadeh, Roland Rivard, Kaveh Mohamed, Jérémie Bisson, Mohsen Karimian and especially Lee Betchen. My fellow student colleagues at McGill's Computational Aerodynamics Group have also been very helpful, and I would like to thank Jeremy Schembri, Yao Jiang, Phillip Zwanenburg, Farshad Navah and Aditya Kashi for their insight. From the same group, my friend Doug Shi-Dong has truly been a pleasure to work next to and I am grateful for all of our fruitful discussions.

I am forever grateful to my girlfriend, Gabrielle Denis-Larocque, for her moral and emotional support. I would also like to express my sincere gratitude towards her parents, Josée Larocque and André Denis.

Last but not least, this work would never have been possible without my parents, Adel Belhaouane and Olga Weyhaeghe, and brother, Hakim Belhaouane. I owe them everything.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This work was done in collaboration with Maya Heat Transfer Technologies, a company developing its own flow solver named NX Flow, part of the Siemens PLM portfolio. The aim of this thesis is the implementation and validation of the Spalart-Allmaras turbulence model in NX Flow as well as the in-house academic code at McGill's Computational Aerodynamics Group, named syn3D.

1.1 Computational fluid dynamics

Computational fluid dynamics (CFD) is a branch of fluid mechanics that uses numerical analysis to solve problems that involve fluid flows. The fundamental basis to all CFD problems is the Navier-Stokes equations, a set of mathematical equations governing the dynamics of any fluid flow. Surprisingly, numerical methods for CFD were first developed in the 1910's by Lewis Fry Richardson even prior to the first computers and were carried out by hand. Then, a team at the Los Alamos National Lab in 1957 developed the first functional CFD computer simulation model. Due to a lack of computational resources, only a simplified set of equations could be solved.

As computational resources became more abundant and the processor speed and available memory increased at a rapid rate, the full Navier-Stokes could finally be solved.

1.2 Turbulence modelling

In theory, the Navier-Stokes equations can be solved numerically, provided initial and boundary conditions. This is what is called Direct Numerical Simulation (DNS). DNS for turbulent flows requires that all turbulent effects, which occur at very small temporal and spatial scales, be resolved. This is further discussed in Section 1.2.1. Common solutions to this problem are given in Section 1.2.2, namely large eddy simulation and the Reynolds-averaged Navier-Stokes (RANS) or Favre-averaged Navier-Stokes (FANS) equations. An introduction to the latter is given in Section 1.2.3. The law of the wall, an important concept in turbulence modelling, is given Section 1.2.4. Finally, an overview of the Spalart-Allmaras model [1], which is the model to be implemented and validated, is given in Section 1.2.5.

1.2.1 Random nature of turbulence

The Reynolds number Re is a dimensionless quantity commonly used to characterize fluid flows and is mathematically expressed as:

$$Re = \frac{\rho U l}{\mu},$$

where l is a characteristic length and U the velocity. The Reynolds number is a measure of the ratio of inertial forces to viscous forces. At low Reynolds numbers, fluid particles may follow relatively straight lines, assuming the flow does not involve much rotation such as in a turbine or compressor, and minimal mixing occurs. For instance, a flow composed of a heterogeneous mixture may remain heterogeneous. This flow regime is referred to as laminar

flow.

On the other hand, flow at high Reynolds numbers is rather chaotic and random. Fluid particles no longer follow straight lines and mixing is significantly increased. A heterogeneous mixture will quickly become homogeneous if it can. Flow properties such as velocity vary randomly in time and space, even if boundary conditions are constant. This regime is called turbulent flow. It also happens that many flows in engineering are turbulent [2], as are all cases mentioned in the present work.

Turbulent flows are easily identifiable by the presence of vortices, or eddies, in the flow. Such a phenomenon was observed and documented as far back as the 15th century, by none other than Leonardo Da Vinci. Figure 1.1 depicts his studies of vortices in turbulent fluid motion. The lower drawing in this figure shows the formation of eddies of various sizes that is typical of turbulent flows. These eddies typically manifest themselves in the vicinity of velocity gradients, such as in free shear flows or near solid boundaries. The physical reason for which DNS is so costly is that it must resolve the production and decay of eddies in order to accurately simulate the flow.

The number of mesh points N required to resolve turbulent effects can be related to the Reynolds number by [3]:

$$N = Re^{2.25}.$$

Hence, memory storage requirements and CPU time grow significantly with the Reynolds number. In addition, the time step size must be reduced such that a fluid particle moves only a fraction of the mesh spacing. Thus, computational costs involved in DNS are too large for even simple problems in the turbulent regime. For instance, resolving the flow field over an airfoil at $Re = 5$ million, a typical flight Reynolds number, would require 10^{15} mesh points. For comparison, simulations that are considered "large" today are typically comprised of 10^8 mesh points, and those can only be solved on massively parallel supercomputers.

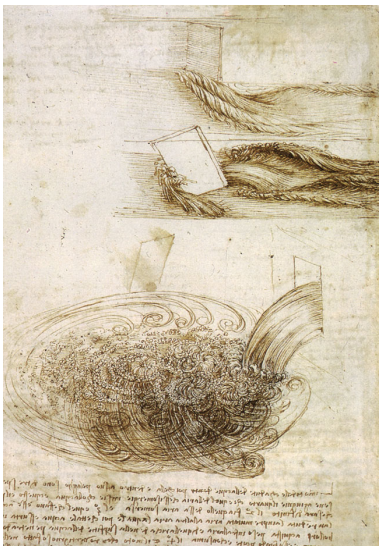


Figure 1.1 – Studies of vortices in turbulent fluid motion by Leonardo da Vinci [4]. Depicts formation of eddies in turbulent flows.

Readers are referred to [3], [5] for a more thorough discussion of turbulence.

1.2.2 Alternatives to DNS

As discussed in the previous section, it is extremely costly to numerically resolve turbulent effects, which makes it impossible to perform simulations on real-life engineering cases using DNS. At the expense of accuracy, it is possible to instead model these effects. A common approach is to employ an averaging technique called Reynolds averaging and solve the Reynolds-averaged Navier-Stokes equations, which yield a solution for the mean flow. This approach is further detailed in Section 2.2.

Another approach, large eddy simulation (LES), which is more costly but typically yields a more accurate solution than RANS, resolves most of the turbulence effects while modelling the smaller scales through low-pass filtering functions. Such a filtering effectively removes small-scale information from the numerical solution, allowing a coarser mesh than DNS to be used.

1.2.3 Reynolds Averaging

For most engineering purposes it is unnecessary to resolve every detail of the turbulent fluctuations. In other words, merely solving for the so-called mean flow would be satisfactory. This would require a set of equations that is only concerned with the mean quantities of the flow. The most common, and probably simplest, way to obtain the mean flow equations is to write all conserved quantities appearing in the conservation equations as a sum of a fluctuating component and a mean component. Taking for instance an instantaneous conserved quantity ϕ, one can write:

$$\begin{aligned}\phi &= \overline{\phi} + \phi' \\ \overline{\phi} &= \frac{1}{\mathrm{T}} \int_{\mathrm{T}} \phi(t)\ \mathrm{d}t,\end{aligned}$$

where $\overline{\phi}$ is the mean component, ϕ' is the fluctuating component, and T is a time interval longer than the characteristic time scale of the turbulence. The mean component is then a time-averaged value, which is only relevant in statistically stationary (steady) flows. This work is only concerned with such flows. This technique is called a Reynolds decomposition.

To allow for compressibility effects, a mass weighted average must also be introduced. The Favre decomposition is defined as:

$$\begin{aligned}\phi &= \widetilde{\phi} + \phi'' \\ \widetilde{\phi} &= \frac{\overline{\rho\phi}}{\overline{\rho}}.\end{aligned}$$

It is important to note that $\overline{\phi'} = 0$ but $\overline{\phi''} \neq 0$.

In order to obtain a set of mean governing equations, a Reynolds decomposition for pressure and density and a Favre decomposition for velocity, internal energy and temperature is taken and substituted into the original (instantaneous) Navier-Stokes equations. Next, a

time average of the resulting equations is taken. This leads to the Favre-averaged Navier-Stokes equations. As shown in Section 2.2.1, this introduces six new unknowns, which are the components of the symmetric Reynolds stress tensor. Since no additional equations are introduced, the system of equations is under-determined. This is the closure problem of turbulence.

A common solution to this problem, which is further detailed in Section 2.2.2, is to approximate the Reynolds stress tensor as a function of density, velocity and a new quantity μ_T called the *turbulent eddy viscosity*. This solution is referred to as the Boussinesq approximation.

1.2.4 Law of the wall

The total shear stress is the sum of viscous and Reynolds stresses. The Reynolds stresses being null at a wall, since $\mathbf{u} = 0$, the total shear stress at the wall τ_w is due entirely to the viscous contribution. It was found through experimentation and simulation that the Reynolds stresses tend to dominate as one moves away from the wall [5]. In fact, the relative importance of both sources of stress can be determined through a non-dimensional wall distance y^+ defined by:

$$y^+ = \frac{u_\tau d}{\nu},$$

where d is the distance to the nearest wall and u_τ is the friction velocity defined as:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}}.$$

It is possible to identify the following layers in the near-wall flow based on the y^+ value:

Viscous sublayer $(y^+ < 5)$: Viscous stresses dominate

Buffer layer $(5 < y^+ < 30)$: Neither stress dominates

Log-law layer $(y^+ > 30)$: Turbulent stresses dominate

Another quantity of interest is the non-dimensional velocity $u^+ = U/u_\tau$ where U is the velocity parallel to the wall, which can be related to y^+ through a function $f_w(y^+)$. For fully turbulent flows, the function f_w has been found to be universal and to vary depending on the layer as follows:

Viscous sublayer : $u^+ = y^+$

Log-law layer : $u^+ = \frac{1}{\kappa} \ln y^+ + C^+$

where $\kappa \approx 0.41$ is the Von Kármán constant and $C^+ \approx 5.0$ for a smooth wall [5]. There is no clear relationship between u^+ and y^+ in the buffer layer, as the contributions to the shear stress are mixed.

In CFD packages that solve the averaged equations, it is recommended to have at least a few points within the viscous sublayer in order to resolve the flow correctly. Thus, in practice, the computation of y^+ is useful in determining whether the mesh requires refinement.

1.2.5 Spalart-Allmaras model

Solving the RANS equations requires evaluation of the eddy viscosity, the quantity that models the effects of turbulence. This quantity is solved for through a turbulence model. One such turbulence model is called the Spalart-Allmaras (SA) model [1], and is the focus of this work. This particular model was designed for aerospace applications and has been shown to give good results for boundary layers subject to adverse pressure gradients [1]. It also involves solving only one additional transport equation, as opposed to other turbulence models such as the $k - \epsilon$ and $k - \omega$ models which require solving two equations. Thus, the Spalart-Allmaras model offers a good trade-off between computational cost and solution accuracy and is well suited for the cases discussed in this work. Another reason for the choice of this model is that a customer of NX Flow requested it. Details and variants of the SA model are given in Section 2.2.3.

The SA equation depends on the distance to the nearest wall, which can be calculated in various ways, as shown in Section 3.2. While the Navier-Stokes equations do not depend on such a quantity, it is important to accurately compute this distance in order to correctly predict the behavior of turbulent flows using the Spalart-Allmaras model. In fact, this value is required by most turbulence models, including the two-equation models mentioned above.

1.3 Thesis overview

The governing equations and an introduction to turbulence modelling are presented in Chapter 2. Numerical methods and their implementation in both solvers are given in Chapter 3, which shows the differences in implementation between both codes.

Results are presented in Chapter 4 and compared against either experimental data or other established CFD codes. The goal is to demonstrate that the implementation in both codes is correct and also to highlight the differences in results that arise due to the different implementations. Special importance is given to the boundary conditions, since they can significantly affect the solution. Specifically, this work aims to determine if the implemented boundary conditions are suitable for external flows and to assess their impact on grid sensitivity.

For some cases, a grid study is performed, in which the same problem is solved on successively finer computational grids. Such a study aims to determine whether grid is sufficiently refined such that the solution lies in the asymptotic range of convergence [6], which is obtained when the grid size no longer significantly affects the result. This type of study will be used to assess the sensitivity of the solution on grid density for both codes. The effects of grid size on the convergence of the residual will also be investigated.

Problems to be solved involve both compressible and incompressible cases; the accuracy of each solver for these types of flow will be discussed.

Chapter 2

Governing Equations

This chapter introduces the equations governing fluid flow. These are shown in their compressible form in Section 2.1 and the turbulence model is given in Section 2.2.

2.1 Conservation laws

The dynamical behavior of any fluid is determined by the following physical conservation laws [7]:

1. Conservation of mass
2. Conservation of momentum
3. Conservation of energy

Given equations of state connecting the thermodynamic variables, these laws form a system of equations that is mathematically closed, i.e. it can theoretically be solved using methods from calculus (analytically) alone if suitable assumptions, initial, and boundary conditions are supplied. An analytic solution has yet to be found and is currently one of the seven Millenium Prize problems [8]. On the other hand, it is possible to solve approximations to

these equations using computers (numerically) by discretizing the equations. This area of study is called Computational Fluid Dynamics (CFD).

The conservation laws impose conditions on the rate of change of mass, momentum and energy per unit volume, and can be expressed mathematically in a three-dimensional Cartesian coordinate system as:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{2.1}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p + \nabla \cdot \tau + \mathbf{S}_M \tag{2.2}$$

$$\frac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho \mathbf{u} H) = \nabla \cdot (\tau \cdot \mathbf{u}) + \nabla \cdot (k \nabla T) + S_E, \tag{2.3}$$

where ρ is the density, $\mathbf{u}$ the velocity vector, τ is the viscous stress tensor, p is the fluid pressure, $\otimes$ is the outer product operator, E is the specific energy of a fluid, H is the specific total enthalpy, T is the temperature, k is the thermal conductivity coefficient, $\mathbf{S}_M$ is the momentum source term and S_E is the energy source term. Readers are referred to [9] for a thorough derivation of these equations.

Assuming the fluid is Newtonian, the stress tensor τ can be written as:

$$\tau_{ij} = \mu \left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right] + \lambda \left[\frac{\partial u_k}{\partial x_k} \right] \delta_{ij}.$$

where μ is the dynamic viscosity, a material property, and the strain-rate tensor S_{ij} is defined as:

$$S_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right).$$

The specific energy is defined as:

$$E = \hat{\mathbf{u}} + \frac{1}{2}(u^2 + v^2 + w^2),$$

where $\hat{\mathbf{u}}$ is the internal thermal energy. The specific total enthalpy and specific energy are

related by the following:

$$H = E + \frac{p}{\rho}.$$

Eqs. (2.1) to (2.3) form an under-determined system of five equations with seven unknowns: density, three components of velocity, pressure, temperature and specific energy. Equations of state are necessary to close the system. If the fluid is assumed to be a perfect gas, which is the case in most aerodynamic problems and all problems in this work, the following equations can be used:

$$p = \rho RT \tag{2.4}$$

$$i = c_v T, \tag{2.5}$$

where R is the specific gas constant and c_v is the specific heat at constant volume – both are material properties.

Finally, any problem definition in CFD requires specification of conditions at boundaries of the physical domain. There currently exists a wide range of available boundary conditions in the literature as well as commercial software. It is the user's responsibility to apply boundary conditions that would most accurately reflect the real behavior of the flow. In the scope of this work specifically, different CFD codes offer different choices of boundary conditions. For this reason, boundary conditions are discussed in the sections pertaining to the different flow solver implementations.

2.2 Turbulence modelling

Section 2.1 introduced the governing equations of the time-dependent three-dimensional fluid flow of a compressible Newtonian fluid. As mentioned in Section 1.2.1, solving these equations directly is nearly impossible in practice. Instead, syn3D and NX Flow solve the

Favre-averaged Navier-Stokes equations, which are detailed in the following sections along with the Spalart-Allmaras turbulence model.

2.2.1 Favre-averaged Navier-Stokes equations

As presented in Section 1.2.3, the Favre-averaged Navier-Stokes equations are obtained by substituting a Favre decomposition for the flow variables and averaging the equations in time. The resulting equations are detailed below.

Conservation of mass

Averaging of Eq. (2.1) results in:

$$\frac{\partial \overline{\rho}}{\partial t} + \nabla \cdot (\overline{\rho}\widetilde{\mathbf{u}}) = 0. \tag{2.6}$$

This new equation is virtually identical to the original one, only with averaged values. Solving the original conservation of mass equation for instantaneous quantities is then no different than solving for the Favre-averaged quantities in the same time-averaged equation. As will be shown in the following sections, the same cannot be said of the remaining conservation laws.

Conservation of momentum

Averaging of Eq. (2.2), ignoring source terms, yields:

$$\frac{\partial \left(\overline{\rho}\widetilde{\mathbf{u}}\right)}{\partial t} + \nabla \cdot (\overline{\rho}\widetilde{\mathbf{u}} \otimes \widetilde{\mathbf{u}}) = -\nabla \overline{p} + \nabla \cdot \overline{\tau} - \nabla \cdot \overline{\rho \mathbf{u}'' \otimes \mathbf{u}''}. \tag{2.7}$$

This equation is also identical to Eq. (2.2), except for the additional term $\tau^T = -\overline{\rho \mathbf{u}'' \otimes \mathbf{u}''}$. This term mathematically comes from the nonlinear convective term and represents the

transfer of momentum due to turbulent fluctuations. Despite that, τ^T is known as the Reynolds-stress tensor. For clarity, the individual components of the Reynolds-stress tensor are defined as:

$$\tau_{ij}^T = \overline{\rho u_i'' u_j''}$$

Moreover, the turbulent kinetic energy K is defined as the sum of the normal stresses (diagonal of the tensor) divided by density:

$$K = \frac{1}{2}\overline{u_k'' u_k''} = \frac{1}{2}\left[\overline{(u'')^2)} + \overline{(v'')^2)} + \overline{(w'')^2)}\right]$$

Because the Reynolds stress tensor is symmetric, this adds a total of six unknowns to the three-dimensional momentum equations, leading to the *turbulence closure problem*. In other words, the Favre-averaged, as well as Reynolds-averaged, conservation laws cannot be solved unless the Reynolds stresses are determined.

Conservation of energy

Averaging of Eq. (2.3), again ignoring source terms, yields an equation with six additional terms. In practice, some of these terms are often ignored because they are assumed to be negligible [7]. After these assumptions are applied, the equation is written as:

$$\frac{\partial\left(\overline{\rho}\widetilde{E}\right)}{\partial t} + \nabla\cdot\left(\overline{\rho}\widetilde{\mathbf{u}}\widetilde{H}\right) = \nabla\cdot\left[\overline{\tau}\cdot\widetilde{\mathbf{u}} + k\nabla\widetilde{T} - \tau^T\cdot\widetilde{\mathbf{u}} - \overline{\rho\mathbf{u}''h''}\right] \tag{2.8}$$

Moreover, the total energy and total enthalpy can be expressed as:

$$\overline{\rho}\widetilde{E} = \overline{\rho}\widetilde{e} + \frac{1}{2}\overline{\rho}\widetilde{u_k}\widetilde{u_k} + K$$
$$\overline{\rho}\widetilde{H} = \overline{\rho}\widetilde{h} + \frac{1}{2}\overline{\rho}\widetilde{u_k}\widetilde{u_k} + K$$

An investigation of Eq. (2.8) reveals two additional terms as compared to Eq. (2.3). These are the last two terms on the right-hand side, and represent the work done by Reynolds stresses and turbulent transport of heat, respectively. In contrast, the first two terms on the right-hand side represent the work done by viscous stresses and diffusion of heat.

Averaging of the conservation of energy equation adds another three unknowns to the system, these are the components of the heat-flux vector $\overline{\rho \mathbf{u}'' h''}$.

2.2.2 Approximations to turbulent terms

The temporal averaging process detailed in the previous subsection introduces new terms to the system of equations. The challenge, which has yet to be completely overcome even today, is to correctly *model* the additional turbulent terms τ^T and $\overline{\rho \mathbf{u}'' h''}$.

In 1877, Boussinesq observed that the momentum transfer in a turbulent flow is dominated by the mixing caused by large energetic turbulent eddies [7]. The Boussinesq approximation then states that the turbulent stress is proportional to the mean strain rate, where the proportionality factor is the so-called eddy viscosity μ_T. This can be written as:

$$\tau_{ij}^T = 2\mu_T \left(\widetilde{S} - \frac{1}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right) - \frac{2}{3} \widetilde{\rho} K \delta_{ij} \tag{2.9}$$

Whereas the dynamic viscosity is a property of the fluid, the eddy viscosity is, or should be, entirely a property of the flow. This is the reason why it is difficult to correctly approximate the term. Nonetheless, this has effectively removed five unknowns from the Favre-averaged conservation laws, since none of the other quantities in τ_T are new.

A Reynolds analogy can be used to model the turbulent heat flux as such:

$$\overline{\rho \mathbf{u}'' h''} = -c_p \frac{\mu_T}{Pr_T} \nabla \widetilde{T} \tag{2.10}$$

where Pr_T is the turbulent Prandtl number, which is assumed to be constant and is equal to 0.9 for air.

Now that both turbulent terms have been modelled, it is convenient to combine the turbulent and laminar components into *effective* components:

$$\mu_{\text{eff}} = \mu + \mu_T$$
$$k_{\text{eff}} = c_p \left(\frac{\mu}{Pr} + \frac{\mu_T}{Pr_T} \right)$$

The sum of viscous and Reynolds stresses can then be written as:

$$\tau_{ij} + \tau_{ij}^T = \mu_{\text{eff}} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \frac{\partial u_k}{\partial u_k} \delta_{ij} \right) - \frac{2}{3} \rho K \delta_{ij}$$

It is then possible to rewrite the Favre-averaged Navier-Stokes equations, noting that the average symbols have been dropped, i.e. every quantity is effectively an averaged quantity.

$$\frac{\partial(\rho)}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{2.11}$$
$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p + \nabla \cdot (\tau + \tau^T) \tag{2.12}$$
$$\frac{\partial \rho E}{\partial t} + \nabla \cdot (\rho \mathbf{u} H) = \nabla \cdot \left[(\tau + \tau^T) \cdot \mathbf{u} \right] - \nabla \cdot (k_{\text{eff}} \nabla T) \tag{2.13}$$

Thus, only the specification of the eddy viscosity is needed in order to solve the Favre-averaged Navier-Stokes equations. It is crucial to remember that the eddy viscosity does not exist, it is nothing but a convenient mathematical invention whose purpose is to *model* the production and decay of eddies in the flow.

2.2.3 Spalart-Allmaras model

The Spalart-Allmaras model adds one partial differential equation for an artificial variable called the *modified eddy viscosity*, identified by $\hat{\nu}$. It is related to the eddy viscosity through a simple algebraic equation:

$$\begin{aligned} \mu_T &= \rho\hat{\nu} f_{v1} \\ f_{v1} &= \frac{\chi^3}{\chi^3 + c_{v1}^3} \\ \chi &= \frac{\hat{\nu}}{\nu}, \end{aligned}$$

where $\nu = \mu/\rho$ is the kinematic viscosity.

The modified eddy viscosity is governed by the following transport equation:

$$\frac{\partial\hat{\nu}}{\partial t} + \mathbf{u}\cdot(\nabla\hat{\nu}) = c_{b1}(1-f_{t2})\hat{\mathbf{S}}\hat{\nu} - \left[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\hat{\nu}}{d}\right)^2 + \frac{1}{\sigma}\nabla\cdot[(\nu+\hat{\nu})\nabla\hat{\nu}] + \frac{c_{b2}}{\sigma}(\nabla\hat{\nu})^2, \quad (2.14)$$

where d is the distance to the nearest solid wall. This is the standard form of the SA model [1], [10]. Comparison of Eq. (2.14) with the Navier-Stokes equations reveals some similarities. The left-hand side is comprised of the time rate of change of a quantity, in this case the modified eddy viscosity, and the advection of the stated quantity. The terms on the right-hand side represent the production, destruction, diffusion and anti-diffusion of $\hat{\nu}$, respectively.

Additional definitions are given by:

$$\begin{gathered} f_{t2} = c_{t3}\exp(-c_{t4}\chi^2), \quad \hat{\mathbf{S}} = \Omega + \frac{\hat{\nu}}{\kappa^2 d^2}f_{v2}, \quad f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}}, \\ \Omega = \sqrt{2W_{ij}W_{ij}}, \quad W_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \quad f_w = g\left[\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right]^{1/6}, \\ g = r + c_{w2}(r^6 - r), \quad r = \min\left[\frac{\hat{\nu}}{\hat{\mathbf{S}}\kappa^2 d^2}, 10\right]. \end{gathered}$$

The non-dimensional function f_w accelerates the decay of the destruction term in the outer region of the boundary layer [1], Ω is the magnitude of the vorticity, $\hat{\mathbf{S}}$ is a modified vorticity term that maintains log-law behavior all the way to the wall and f_{t2} is a term delaying transition from laminar to turbulent regimes [10].

The constants are:

$$c_{b1} = 0.1355, \quad \sigma = 2/3, \quad c_{b2} = 0.622, \quad \kappa = 0.41$$
$$c_{w2} = 0.3, \quad c_{w3} = 2, \quad c_{v1} = 7.1, \quad c_{t3} = 1.2$$
$$c_{t4} = 0.5, \quad c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1 + c_{b2}}{\sigma}.$$

The boundary conditions are:

— $\hat{\nu} = 0$ at solid walls.

— $\hat{\nu} = 3\nu_\infty$ to $5\nu_\infty$ in the far-field.

There exists several variants of the model, which are enumerated in [10]. A common variant involves neglecting the f_{t2} term. Based on studies in [11], use of this form makes little difference in the solution at reasonably high Reynolds numbers, provided that the far-field $\hat{\nu}$ value is large enough. Other variants involve changes to the production term to account for rotation and curvature effects [12], [13], [14]. Some other variants slightly alter the model so as to prevent values of $\hat{\nu}$ from being negative, such as the negative SA model [15] which provides an alternative equation to be solved in regions of negative $\hat{\nu}$. Finally, [16] presents a nonlinear model that does not rely on the traditional linear Bousinnesq relation and instead writes τ^T as a nonlinear function.

It should be emphasized that this is a model, as opposed to a mathematical expression of a law like the Navier-Stokes equations. Most turbulence models involve the calculation of the wall distance d, the accuracy of which can have a significant impact on the solution.

Finally, while some models involve the computation of the turbulent kinetic energy,

the Spalart-Allmaras model does not. Thus, the last term in Eq. (2.9) involving k is ignored. In practice, this term is small compared to the others, especially in non-supersonic flows [10].

Chapter 3

Numerical Methods and Implementation

In this chapter the discretization methods which approximate the differential equations established in Chapter 2 are presented. The discretization methods used in both NX Flow and syn3D is the finite volume (FV) method, which is described in Section 3.1 in a general sense. Methods of calculating the wall distance are discussed in Section 3.2. Sections 3.3 and 3.4 discuss the specifics of the implementation and chosen numerical schemes for syn3D and NX Flow respectively.

3.1 Finite Volume Method

The finite volume method requires the specification of a *computational grid*, a discrete representation of the geometric domain on which the problem is to be solved. Another name for computational grid is *mesh*. This essentially divides the domain into a finite number of subdomains. In the case of the finite volume method, these subdomains are referred to as *control volumes* (CVs). The governing equations must then be solved over each of these discrete control volumes, which results in a solution field at discrete locations, as opposed to an analytical function.

3.1.1 Integral form of the conservation laws

A special property of the finite volume method is that it uses the integral form of the conservation equations. Moreover, it employs the Gauss theorem, also known as divergence theorem, to express volume integrals as surface integrals. Specifically, let $\mathbf{F}$ be a continuously differentiable vector field, Ω be a volume in three-dimensional space, $\partial\Omega$ be the boundary of the volume and $\mathrm{d}S$ be a surface element along with its associated unit outward normal vector $\mathbf{n}$, the divergence theorem states that:

$$\iiint_{\Omega} (\nabla \cdot \mathbf{F}) \; d\Omega = \oiint_{\partial\Omega} (\mathbf{F} \cdot \mathbf{n}) \; dS.$$

The integral on the left-hand side is a volume integral, whereas the integral on the right-hand side is a surface integral. For the remainder of this work, the triple integral and closed integral notations will be replaced with simple integrals, where the type can be inferred from the limit (Ω or $\partial\Omega$).

Investigation of Eqs. (2.1) to (2.3) reveals commonalities between the equations. Again, let ϕ be a general conserved property, the so-called transport equation for ϕ can be written in integral form as:

$$\frac{\partial}{\partial t} \int_{\Omega} \rho\phi \; d\Omega + \int_{\partial\Omega} (\rho\phi\mathbf{u}) \cdot \mathbf{n} \; dS = \int_{\partial\Omega} (\Gamma\nabla\phi) \cdot \mathbf{n} \; dS + \int_{\Omega} Q_{\phi} \; d\Omega, \tag{3.1}$$

where Γ is a general diffusion coefficient and Q_{ϕ} is the source term. Reading from left to right, the terms are labelled as: temporal derivative term, convective term, diffusive term and source term. The convective and diffusive terms being surface integrals, are also referred to as *flux* terms. All conservation equations, specifically mass, momentum and energy, can be written in this form, where Q takes care of the special terms, e.g. the pressure gradient term in the momentum equation.

The finite volume method is attractive due to its simplicity and its ability to accom-

modate any type of grid. The method is also *conservative* by construction, so long as fluxes for CVs sharing a boundary are identical.

Again, Eq. (3.1) is to be solved over all CVs. Within each CV lies a *computational node*, also known as field point, at which the dependent quantities are to be calculated.

3.1.2 Numerical approximations

The integral transport equation in Eq. (3.1) remains analytical; no numerical approximations have been made thus far. To enable computer implementation, the equation must be discretized, so as to yield a finite set of algebraic equations in the form of:

$$a_P \phi_P + \sum_{nb} a_{nb} \phi_{nb} = b_P,$$

where P is the control volume of interest, a's and b's are constant coefficients, and nb represents the neighboring field points involved in the discretization. One speaks of a *stencil* when referring to these neighbors. Discretization inevitably introduces discretization errors, which is why it is always an approximation. The goal is to reduce these errors to a negligible level.

Discretization of the finite volume equations can be split in two steps:

Spatial discretization : Numerical approximation of integrals and spatial derivatives

Temporal discretization : Numerical approximation of time derivative

Spatial discretization

Every term in Eq. (3.1) is an integral. To calculate either the volume or surface integrals exactly would require knowledge of the integrand everywhere on the volume or surface;

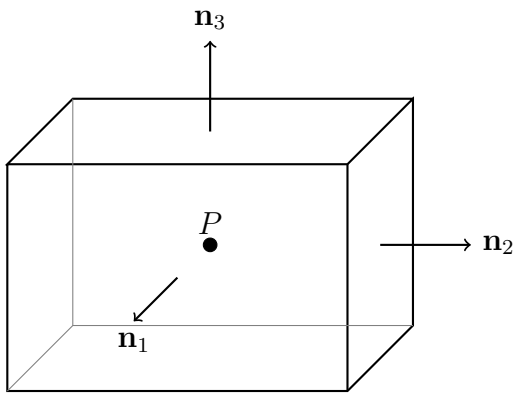


Figure 3.1 – A hexahedron element. Field point is shown as solid black circle.

however such information is not available. Discretization of the integrals in Eq. (3.1) can be done using three levels of approximation: integration, interpolation and differentiation.

Let $\mathbf{F}$ be a flux vector, the closed surface integrals for an arbitrary polyhedron, which is the only type of three-dimensional control volume that is used in most finite volume codes, then the integral of the normal flux over the control surface can be written as:

$$\int_{\partial\Omega} (\mathbf{F} \cdot \mathbf{n})\, dS = \sum_{k=1}^{N_f} \int_{S_k} (\mathbf{F} \cdot \mathbf{n}_k)\, dS, \tag{3.2}$$

where N_f is the number of faces. Taking for instance a hexahedron, a commonly used element in CFD, like the one depicted in Figure 3.1, the closed integral is simply the sum of the integral over all $N_f = 6$ faces.

The individual face integrals then need to be numerically integrated. The most commonly used numerical integration method (quadrature) is the midpoint rule, for which there is only one integration point placed at the face center. Application of the midpoint rule to a single surface integral in Eq. (3.2) yields:

$$\int_S \mathbf{F} \cdot \mathbf{n}\, dS = (\mathbf{F}_{S_c} \cdot \mathbf{n})S + \mathcal{O}(\delta^2),$$

where S is the surface area, $\mathbf{F}_{S_c}$ is $\mathbf{F}$ evaluated at the face center and δ is the characteristic

length. The approximation is of second-order accuracy, as denoted by the second term on the right-hand side, provided that $\mathbf{F}_{S_c}$ is also evaluated with at least second-order accuracy. Then, a closed surface integral may be written as a sum over all integration points (ip)

$$\int_{\partial\Omega} (\mathbf{F} \cdot \mathbf{n}) \; dS = \sum_{ip} (\mathbf{F}_{ip} \cdot \mathbf{n}_{ip}) S_{ip}.$$

The volume integral can also be numerically integrated using the midpoint rule. A volume integral over a CV can then be approximated as:

$$\int_{\Omega} G \; d\Omega = G_P V + \mathcal{O}(\delta^2),$$

where V is the volume and G_P is G at the field point P. This method is of second-order accuracy provided that P lies at the CV centroid.

Application of the integral approximations to Eq. (3.1), ignoring the $\mathcal{O}$ terms, results in:

$$\frac{\partial(\rho\phi V)_P}{\partial t} = \sum_{ip} [-(\rho\phi\mathbf{u})_{ip} \cdot \mathbf{n}_{ip}] S_{ip} = \sum_{ip} [(\Gamma\nabla\phi)_{ip} \cdot \mathbf{n}_{ip}] S_{ip} + Q_P V. \tag{3.3}$$

Whereas all quantities at point P are readily available, quantities and gradients at the integration points must be expressed in terms of the field point values via interpolation and differentiation schemes respectively. Interpolation and differentiation schemes are discussed for each solver separately.

Temporal discretization

Application of spatial discretization schemes leads to an ordinary differential equation (ODE) for each CV of the form:

$$\frac{\partial(\rho\phi V)_P}{\partial t} = R_P, \tag{3.4}$$

where R is the so-called residual of the equation and generally includes quantities depending on neighboring field points.

Given the solution field ϕ^n at time t, a new solution ϕ^{n+1} at time $t + \Delta t$ is sought. This is typically done even for steady-state problems using the concept of pseudo-time [7], which is necessary due to the non-linearity of the equations. Integrating in time on both sides of Eq. (3.4) results in:

$$\int_t^{t+\Delta t} \frac{\partial(\rho\phi V)_P}{\partial t}\, dt = -\int_t^{t+\Delta t} R_P\, dt. \tag{3.5}$$

Since the only unknown field in this equation is ϕ, it is possible to say:

$$\frac{\partial(\rho\phi V)_P}{\partial t} = (\rho V)_P \frac{\partial \phi_P}{\partial t}.$$

The left-hand side Eq. (3.5) can then be evaluated as:

$$\int_t^{t+\Delta t} \frac{\partial(\rho\phi V)_P}{\partial t}\, dt = (\rho V)_P(\phi^{n+1} - \phi^n)_P = (\rho V)_P \Delta\phi_P^n. \tag{3.6}$$

Integration of the right-hand side of Eq. (3.5) can be done in various ways: one can use the solution at time t, time $t + \Delta t$ or a mix of both to calculate the integral. Certain codes even use the solution at time $t - \Delta t$, although this is not done in either solver presented in this work. A family of methods can be generalized by introducing a weighting parameter θ which is allowed to vary between 0 and 1 and approximating the integral of the residual as:

$$\int_t^{t+\Delta t} R_P\, dt = \left[\theta R_P^{n+1} + (1-\theta) R_P^n\right] \Delta t. \tag{3.7}$$

The forward Euler, backward Euler and Crank-Nicolson methods can be recovered by letting $\theta = 0$, $\theta = 1$ and $\theta = 0.5$ respectively. Substitution of Eqs. (3.6) and (3.7) in Eq. (3.5) and

dividing by Δt on both sides results in:

$$\frac{(\rho V)_P}{\Delta t}\Delta\phi_P^n = -\theta R_P^{n+1} - (1-\theta)R_P^n. \tag{3.8}$$

Evaluation of R_P^{n+1} in Eq. (3.8) requires knowledge of the solution field at the new time level ϕ^{n+1}, which is not known a priori. To complicate matters, the residual in the Navier-Stokes equations is non-linear and its discretized form cannot be expressed as a linear combination of the field points. However, it is possible to linearise the residual at the new time level with a first-order Taylor series expansion:

$$R_P^{n+1} \approx R_P^n + \left(\frac{\partial R}{\partial \phi}\right)_P \Delta\phi_P^n,$$

where the partial differential term $\partial R/\partial\phi$ is commonly referred to as the flux Jacobian and depends on the chosen spatial discretization schemes. Substituting the linearisation in Eq. (3.8) results in:

$$\left[\frac{(\rho V)}{\Delta t} + \theta\left(\frac{\partial R}{\partial \phi}\right)\right]_P \Delta\phi_P^n = -R_P^n. \tag{3.9}$$

Thus, provided an initial solution field, it is possible to advance ϕ in pseudo-time until the solution is *converged*, i.e. a steady-state solution is reached. Convergence is typically monitored by calculating $||R||_k$ where k is a specified vector norm, which is typically either the Euclidean norm or the infinity norm. A converged solution is then attained when the norm of the residual falls below a user-specified tolerance ϵ.

Regardless of the chosen scheme, the result is a linear system of equations of the familiar form:

$$Ax = b,$$

where A is a sparse matrix, x is the vector of unknowns and b is a vector. The linear system can be solved with a direct or iterative method.

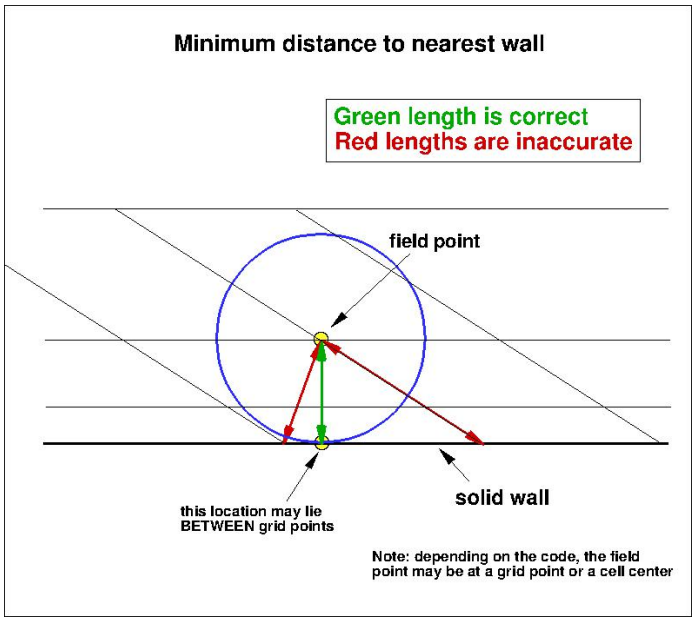


Figure 3.2 – Common approximation made in brute force wall distance calculations [18].

3.2 Wall distance computation methods

Calculation of the wall distance d is required in order to solve the Spalart-Allmaras equation. In the literature, it is typically computed in one of two ways:

1. by a brute force approach.
2. by solving an additional PDE.

The brute force approach is straight-forward and consists of looping through every CV and finding the nearest wall, which itself is done by looping over every face lying on the solid boundary. Depending on the mesh, this approach can be very costly [17]. Moreover, some codes approximate the wall distance for a field point P as the distance between P and the nearest vertex or face center lying on a solid boundary, as opposed to calculating the perpendicular distance between a point and a plane, which is more costly. This approximation and the potential error are illustrated in Figure 3.2.

The other approach, solving an additional PDE, may seem like a lot of extra work. However, CFD codes already provide functions and data structures required in solving such

equations. There exists many different equations for approximating the wall distance, which are detailed and compared in [17], [19], [20]. The technique used in syn3D and NX Flow consists of solving a Poisson equation followed by a normalization, which can mathematically be written as:

$$\begin{aligned} \nabla^2\phi &= -1 \\ d &= ||\nabla\phi||_2 + \sqrt{||\nabla\phi||_2 + 2\phi} \end{aligned}. \tag{3.10}$$

A homogeneous Dirichlet BC is to be imposed on ϕ at solid walls and a homogeneous Neumann BC at all other boundaries. The approximation of d by Eq. (3.10) is only accurate close to walls, however that is the only location where an accurate d is needed when it comes to turbulence modelling.

3.3 Syn3D

The academic code, named syn3D, is owned by Professor Siva Nadarajah of McGill University. The program has the following characteristics:

— Non-dimensional

— Structured

— Uses coordinate transformation to generalized curvilinear coordinates

— Cell-centered

Non-dimensionalization is explained in Section 3.3.1, the mesh structure is given in Section 3.3.2, the coordinate transformation method is summarized in Section 3.3.3, the discretization of the fluid flow and turbulence governing equations are given in Sections 3.3.4 and 3.3.5 respectively and the wall distance calculation details are given in Section 3.3.6.

3.3.1 Non-dimensionalization

Round-off errors can be reduced by non-dimensionalizing the governing equations such that all quantities remain in the same order of magnitude. While arbitrary reference quantities can be used, it is convenient to use significant quantities. External flow problems being usually described by free-stream values, the following scaling parameters are used: the airfoil chord length L, the free-stream density ρ_∞, the free-stream pressure p_∞, the free-stream dynamic viscosity μ_∞ and a reference temperature T_{ref}. Usage of these scales leads to the following relations between dimensional quantities, now denoted by the superscript *, and non-dimensionalized quantities:

$$\rho = \frac{\rho^*}{\rho_\infty}, \qquad u_i = \frac{u_i^*}{\sqrt{p_\infty/\rho_\infty}}, \qquad p = \frac{p^*}{p_\infty}, \qquad i = \frac{i^*}{p_\infty/\rho_\infty},$$
$$x_i = \frac{x_i^*}{L}, \qquad t = \frac{t^*\sqrt{p_\infty/\rho_\infty}}{L}, \qquad \Omega = \Omega^*\frac{c}{\sqrt{p_\infty/\rho_\infty}},$$
$$\mu = \frac{\mu^*}{\mu_\infty}, \qquad \mu_T = \frac{\mu_T^*}{\mu_\infty}, \qquad \hat{\nu} = \frac{\hat{\nu}^*}{\mu_\infty/\rho_\infty}, \qquad d = \frac{d^*}{L}.$$

It also should be noted that the dynamic viscosity is computed from the temperature using Sutherland's law

$$\mu = \frac{C_1 T^{3/2}}{T + S},$$

where $C_1 = 1.461E - 6$, $S = 110.3$. The temperature in the previous equation is computed from

$$T = T_{ref}\frac{p}{\rho},$$

The free-stream dynamic viscosity is computed in the same manner using T_{ref} in the place of T.

Substitution of the non-dimensional relations into Eqs. (2.11) to (2.13), the Favre-

averaged Navier-Stokes equations, and rearranging yields:

$$\frac{\partial(\rho)}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{3.11}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p + \sqrt{\gamma}\frac{M_\infty}{Re}\nabla \cdot (\tau + \tau^T) \tag{3.12}$$

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot (\rho \mathbf{u} H) = \sqrt{\gamma}\frac{M_\infty}{Re}\nabla \cdot \left[(\tau + \tau^T) \cdot \mathbf{u}\right] - \sqrt{\gamma}\frac{M_\infty}{Re}\nabla \cdot (k_{\text{eff}} \nabla T)\,. \tag{3.13}$$

The equations are unchanged except for the additional scaling factor $\sqrt{\gamma}\frac{M_\infty}{Re}$ multiplying viscous terms in the momentum and energy equations, where M_∞ is the free-stream Mach number. This also shows that an increase in the Reynolds number will cause a decrease in magnitude of the viscous terms, which is in agreement with the definition of the Reynolds number.

Repeating the process above for the Spalart-Allmaras equation and rearranging the production and destruction terms so as to group terms with common factors results in:

$$\begin{aligned}\frac{\partial \hat{\nu}}{\partial t} + \mathbf{u} \cdot (\nabla \hat{\nu}) = c_{b1}(1 - f_{t2})\Omega\hat{\nu} + \sqrt{\gamma}\frac{M_\infty}{Re}\left\{C_{b1}[(1 - f_{t2})f_{v2} + f_{t2}]\frac{1}{\kappa^2} - c_{w1}f_w\right\}\left(\frac{\hat{\nu}}{d}\right)^2 \\ + \sqrt{\gamma}\frac{M_\infty}{Re}\frac{1}{\sigma}\nabla \cdot [(\nu + \hat{\nu})\nabla\hat{\nu}] + \sqrt{\gamma}\frac{M_\infty}{Re}\frac{c_{b2}}{\sigma}\left(\nabla\hat{\nu}\right)^2.\end{aligned} \tag{3.14}$$

The auxiliary functions require the same treatment:

$$r = \min\left[\sqrt{\gamma}\frac{M_\infty}{Re}\frac{\hat{\nu}}{\hat{\mathbf{S}}\kappa^2 d^2}, 10\right]$$

$$\hat{\mathbf{S}} = \Omega + \sqrt{\gamma}\frac{M_\infty}{Re}\frac{\hat{\nu}}{\kappa^2 d^2} f_{v2}.$$

Unmentioned functions as well as constants remain unchanged.

3.3.2 Computational Grid

Syn3D solves problems on three-dimensional multiblock-structured meshes, making the hexahedron the only acceptable element type. Moreover, a field point is located at the centroid of each element – it follows that the CVs are all hexahedrons. A general hexahedron CV is shown in Figure 3.1. In structured grids, each field point, or CV, is uniquely identified by its i, j and k indices. Its neighbors can also be identified by incrementing and decrementing the i, j and k indices, which is the main advantage of this method. A more detailed discussion of grid generation is given in [7].

3.3.3 Coordinate transformation

The governing equations are written in generalized curvilinear coordinates through a transformation from physical space (Cartesian coordinates) to a more convenient computational space (generalized curvilinear coordinates). A curvilinear grid can be represented through a transformation from the Cartesian coordinates x_i (x, y, z in three dimensions) to curvilinear coordinates ξ_i (ξ, η, ζ in three dimensions), where each grid line is a line of constant coordinate ξ_i. In three dimensions, ξ, η and ζ correspond to the index-directions i, j, k mentioned in Section 3.3.2. Figure 3.3 illustrates the concept in two dimensions. The transformation is performed by setting the new coordinates to be functions of the Cartesian coordinates, and vice versa:

$$\xi_i = \xi_i(x_j), \quad x_i = x_i(\xi_j), \quad j = 1, 2, 3.$$

Transformation from physical to computational space is then defined by the metrics:

$$K_{nm} = \left[\frac{\partial x_n}{\partial \xi_m}\right], \quad J = \det(K), \quad K^{-1}_{nm} = \left[\frac{\partial \xi_n}{\partial x_m}\right], \quad n, m = 1, 2, 3.$$

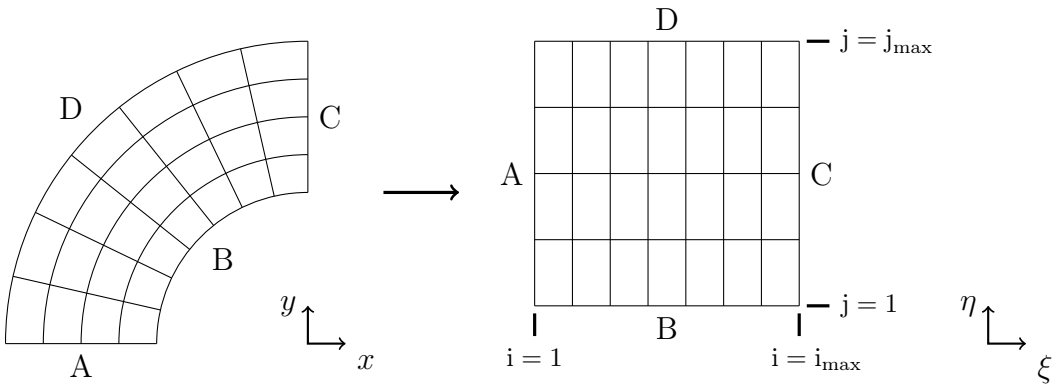


Figure 3.3 – Mapping from physical space (x, y) to computational space (ξ, η)

Cartesian derivatives can then be expressed in terms of the curvilinear coordinates using the chain rule:

$$\frac{\partial \phi}{\partial x_j} = K_{ij}^{-1} \frac{\partial}{\partial \xi_i}, \tag{3.15}$$

where implicit summation is used.

A geometrical interpretation of these metric terms can be made. The Jacobian J corresponds to the volume of the cell and the vector $\nabla \xi_k J$ is the directed area of the cell interface normal to the ξ_k direction. The following notation can then be introduced

$$\nabla \xi_k J = \mathbf{S}_k = (S_{k1}, S_{k2}, S_{k3})$$
$$\frac{\mathbf{S}_k}{|\mathbf{S}_k|} = \mathbf{n},$$

where the $\mathbf{n}$ for the face in the ξ_k direction is chosen and in this case $\mathbf{n}$ always points towards increasing ξ_k.

Then, given the metric terms, derivatives can be evaluated using simple finite difference methods, which is the primary goal of performing a coordinate transformation. For instance, one can obtain an approximation to $\partial \phi / \partial \xi$ using centered differences:

$$\frac{\partial \phi}{\partial \xi} = \frac{\phi_{i+1,j,k} - \phi_{i-1,j,k}}{2\Delta \xi},$$

while $\Delta\xi$ is arbitrary, it is typically chosen to be equal to one.

Finally, usage of a curvilinear grid also allows the closed surface integral, where the midpoint method is used as the quadrature, to be written for the CV at coordinate (i, j, k) as:

$$\begin{aligned}\int_{\partial\Omega} (\mathbf{F}\cdot\mathbf{n})\, dS = &(\mathbf{F}\cdot\mathbf{S}_1)_{i+\frac{1}{2},j,k} - (\mathbf{F}\cdot\mathbf{S}_1)_{i-\frac{1}{2},j,k} \\ &+ (\mathbf{F}\cdot\mathbf{S}_2)_{i,j+\frac{1}{2},k} - (\mathbf{F}\cdot\mathbf{S}_2)_{i,j-\frac{1}{2},k} \\ &+ (\mathbf{F}\cdot\mathbf{S}_3)_{i,j,k+\frac{1}{2}} - (\mathbf{F}\cdot\mathbf{S}_3)_{i,j,k-\frac{1}{2}}.\end{aligned} \tag{3.16}$$

The integration points are at the midpoint of each face enclosing the hexahedron CV, which are represented by the $\pm\frac{1}{2}$ increments to indices. The three lines in Eq. (3.16) on the right-hand side can be seen as the fluxes in the ξ, η and ζ directions respectively.

3.3.4 Discretization of the fluid flow equations

This section describes the numerical discretization of the Favre-averaged Navier-Stokes equations, namely: temporal discretization, discretization of convective fluxes, discretization of viscous fluxes, artificial dissipation, boundary conditions and convergence acceleration.

Temporal discretization

Syn3D uses the modified Runge-Kutta approach introduced by Jameson *et al.* [21] to march the solution in pseudo-time. Let $\mathbf{W}$ be the vector of conserved variables defined as:

$$\mathbf{W} = \begin{Bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \end{Bmatrix}, \tag{3.17}$$

and $\mathbf{R}$ be the residual vector, the modified Runge-Kutta scheme advances the solution at the current time level $\mathbf{W}^{(n)}$ to the solution at the next time level $\mathbf{W}^{(n+1)}$ in the following manner:

$$\begin{aligned} \mathbf{W}^{(0)} &= \mathbf{W}^{(n)} \\ \mathbf{W}^{(k)} &= \mathbf{W}^{(0)} - \alpha_k \Delta t \mathbf{R}(\mathbf{W}^{(k-1)}), \quad k = 1, ..., m \\ \mathbf{W}^{(n+1)} &= \mathbf{W}^{(m)}, \end{aligned}$$

where m is the total number of stages and $\mathbf{R}(\mathbf{W}^{(k-1)})$ is the residual evaluated using the solution from the previous stage $\mathbf{W}^{(k-1)}$. Compared to the classical Runge-Kutta methods where the solution at each stage is kept to obtain the solution at the next time level, the modified Runge-Kutta only requires storage of the latest residual and the zeroth solution, which reduces memory requirements.

The dissipative fluxes, which includes viscous fluxes and artificial dissipation, and convective fluxes are treated separately when computing the residual. The residual at each

stage can be defined as:

$$\begin{aligned}
\mathbf{R}^{(k)} &= \mathbf{R}_c^{(k)}) + \mathbf{R}_d^{(k)} \\
\mathbf{R}_c^{(k)} &= \mathbf{R}_c(\mathbf{W}^{(k)}) \\
\mathbf{R}_d^{(k)} &= \beta_k \mathbf{R}_d(W^{(k)}) + (1-\beta_k)\mathbf{R}_d^{(k-1)},
\end{aligned}$$

where $\mathbf{R}_c$ and $\mathbf{R}_d$ are the convective and dissipative contributions to the residual respectively. The stage coefficients α_k and blending coefficients β_k are chosen such that numerical stability is maximized. In this work, a five stage scheme is employed along with the following coefficients:

$$\alpha_1 = 0.25 \quad \alpha_2 = 0.1667 \quad \alpha_3 = 0.375 \quad \alpha_4 = 0.5 \quad \alpha_5 = 1.0$$

$$\beta_1 = 1,0 \quad \beta_2 = 0.0 \quad \beta_3 = 0,56 \quad \beta_4 = 0.0 \quad \beta_5 = 0.44.$$

Discretization of convective fluxes

Computation of convective fluxes require evaluation of the conserved quantities at cell faces. This is accomplished through simple arithmetic averaging. For instance, let $\mathbf{F}_c$ be the convected quantity, then its value on the $i+\frac{1}{2}$ face is approximated as:

$$\mathbf{F}_{c_{i+\frac{1}{2},j,k}} = \frac{1}{2}\left(\mathbf{F}_{c_{i+1,j,k}} + \mathbf{F}_{c_{i,j,k}}\right). \tag{3.18}$$

This leads to a three-point stencil in each grid direction. Discretization of the convective fluxes in this manner then results in a seven-point stencil for each CV.

Discretization of pressure

The pressure gradient in the momentum equations is typically written as a source term. However, the term can be treated conservatively by treating it as a surface force and applying the divergence theorem as such:

$$\int_{\Omega} \nabla p \; d\Omega = \int_{\partial\Omega} p\mathbf{n} \; dS.$$

The above equation results in a three-dimensional vector with each component belonging to either the x-momentum, y-momentum and z-momentum equations. The integral for the momentum equation in the m Cartesian direction is then discretized according to Eq. (3.16) with $\mathbf{F} = p\boldsymbol{i}_m$ and the pressure at integration points is evaluated using Eq. (3.18).

Discretization of viscous fluxes

In addition to the evaluation of quantities at each integration point, computation of viscous fluxes also requires computation of derivative terms, which are not available either at the field points or integration points. This is done using two levels of approximation: interpolation and differentiation.

The value of the flux on a given face is approximated by the arithmethic average of the flux quantities at the four vertices sharing this face. For instance, let $\mathbf{F}_v$ be the viscous flux, then its value on the $i + \frac{1}{2}$ face is approximated as:

$$\mathbf{F}_{v_{i+\frac{1}{2},j,k}} = \frac{1}{4}\left(\mathbf{F}_{v_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}} + \mathbf{F}_{v_{i+\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}}} + \mathbf{F}_{v_{i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}} + \mathbf{F}_{v_{i+\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}}}\right).$$

An auxiliary control volume is formed at each vertex by joining the centroids of all eight elements that share that vertex. This is illustrated in Figure 3.4 for the two-dimensional

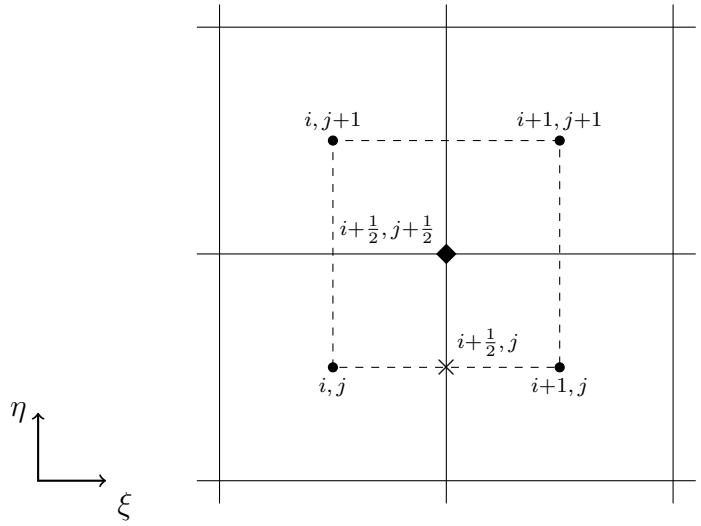


Figure 3.4 – Auxiliary control volume (dashed) for the viscous fluxes in two dimensions.

case. For the sake of conciseness, let the viscous flux $\mathbf{F}_v$ be of the form $\Gamma\nabla\phi$ – even though the viscous term in the momentum and energy equations contain more derivatives, they are all calculated in the same manner. The diffusion coefficient value Γ at a vertex is approximated by an arithmethic average of the values at field points, which form the vertices of the auxiliary control volume. This can mathematically be written for the vertex at $i+\frac{1}{2}, j+\frac{1}{2}, k+\frac{1}{2}$:

$$\begin{aligned}\Gamma_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} = \frac{1}{8}(&\Gamma_{i,j,k} + \Gamma_{i+1,j,k} + \Gamma_{i,j+1,k} + \Gamma_{i+1,j+1,k} \\ &+\Gamma_{i,j,k+1} + \Gamma_{i+1,j,k+1} + \Gamma_{i,j+1,k+1} + \Gamma_{i+1,j+1,k+1}).\end{aligned}$$

Gradients $\nabla\phi$ at the vertices are calculated through a transformation to curvilinear coordinates. For example, the k-component of the gradient at the vertex mentioned above is written as:

$$\left[\frac{\partial\phi}{\partial x_k}\right]_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} = \frac{1}{J_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}}\left[\frac{\partial\hat{\phi}_{1n}}{\partial\xi} + \frac{\partial\hat{\phi}_{2n}}{\partial\eta} + \frac{\partial\hat{\phi}_{3n}}{\partial\zeta}\right]_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}, \tag{3.19}$$

where J is the volume of the auxiliary control volume and is approximated in the same way as Γ. The gradient components in the computational domain are calculated by taking an

average of the centered differences in each direction, which can be written as:

$$\left[\frac{\partial\hat{\phi}_{1n}}{\partial\xi}\right]_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} = \frac{(\hat{\phi}_{1n,i+1,j,k} - \hat{\phi}_{1n,i,j,k}) + (\hat{\phi}_{1n,i+1,j+1,k+1} - \hat{\phi}_{1n,i,j+1,k+1})}{4} + \frac{(\hat{\phi}_{1n,i+1,j+1,k} - \hat{\phi}_{1n,i,+1j,k}) + (\hat{\phi}_{1n,i+1,j,k+1} - \hat{\phi}_{1n,i,j,k+1})}{4},$$

where $\hat{\phi}$ is the quantity multiplied with the area-directed normal as such:

$$\hat{\phi}_{1n,i,j,k} = S_{1n,i,j,k}\phi_{i,j,k},$$

and the directed area at the field point is obtained by an arithmetic average:

$$S_{1n,i,j,k} = \frac{S_{1n,i+\frac{1}{2},j,k} + S_{1n,i-\frac{1}{2},j,k}}{2}.$$

This formulation produces a second-order accurate scheme and the stencil extends over twenty-seven cells since it depends on all field points that share at least one vertex with the CV.

Artificial dissipation and convergence acceleration

Usage of second-order centered difference schemes is known to admit chequer-board solutions, which leads to odd-even point decoupling [2]. This effect can be prevented by adding artificial dissipation to the equations, which was first shown in [22]. This new term does not show itself in the governing equations. Syn3D uses an artificial dissipation scheme first introduced by Jameson, Schmidt and Turkel in [21], and referred to as the JST scheme. Its implementation is described in detail [23].

Convergence is accelerated through means of multigrid, residual averaging and local time stepping. Implementation of these are also detailed in [23].

Boundary conditions

To facilitate implementation of boundary conditions, syn3D uses the concept of ghost cells, or halo cells, which are additional CVs introduced beyond the physical boundaries. This allows the programmer to treat all interior CVs in exactly the same way when calculating fluxes, irrespective of whether the CV has a face on a boundary. This concept is further explained in [7]. Field points belonging to interior CVs with a face marked as a physical boundary will be sub-scripted with a 2 and field points belonging to ghost CVs will be sub-scripted with a 1.

This work uses three different types of boundary conditions:

1. Solid wall
2. Symmetry plane
3. Far-field

Physically, solid walls require that flow neither enter or leave the domain, which is also referred to as a no-slip boundary condition. This can be expressed mathematically as:

$$\mathbf{u}|_{wall} = \mathbf{0}.$$

This can be achieved without any changes to the fluxes mentioned previously by setting the following values for the first level halos:

$$\rho_1 = \rho_2 \quad \mathbf{u}_1 = -\mathbf{u}_2 \quad p_1 = p_2,$$

and by updating density, velocity, pressure and energy at second level halos according to:

$$\phi_0 = \phi_1 + (\phi_1 - \phi_2)\,,$$

which is simply a linear extrapolation.

Symmetry planes are required if the flow is to be symmetrical with respect to a plane, which requires the following to be true at the interface

$$\mathbf{n} \cdot \nabla \phi = 0$$
$$\mathbf{n} \cdot \nabla (\mathbf{u} \cdot \mathbf{t}) = 0$$
$$\mathbf{t} \cdot \nabla (\mathbf{u} \cdot \mathbf{n}) = 0,$$

where $\mathbf{t}$ is the vector tangential to the boundary and ϕ represents any scalar quantity. This type of boundary condition is easily implemented by setting density, velocity, pressure and energy in the following manner:

$$\phi_1 = \phi_2 \quad \phi_0 = \phi_3,$$

where the subscript 0 denotes a ghost CV that is cell adjacent to the ghost CV 1, meaning that ghost CV 0 shares a face with ghost CV 1, and 3 denotes an interior CV that is adjacent to CV 2.

Finally, all simulations have to be conducted within a finite bounded domain. However, simulating external flows requires the incoming far-field (free-stream) flow to not be affected by the body under consideration. Far-field boundary conditions are used for this purpose. Because this is not obvious in practice, there are various ways to implement such conditions. The method used in syn3D is described in detail in [24] and only briefly summarized here. At far-field boundaries, a characteristic based condition using Riemann invariants is imposed. The Riemann invariants can be expressed as:

$$R_\infty = \mathbf{u}_\infty \cdot \mathbf{n} - \frac{2c_\infty}{\gamma - 1}$$
$$R_e = \mathbf{u}_e \cdot \mathbf{n} + \frac{2c_\infty}{\gamma - 1},$$

where freestream and interior values are denoted by the ∞ and e subscripts and c is the

speed of sound. The velocity normal to the boundary and the speed of sound can then be expressed as:

$$\mathbf{u} \cdot n = \frac{1}{2}(R_\infty + R_e)$$
$$c = \frac{\gamma - 1}{4}(-R_\infty + R_e).$$

The velocity at the boundary can then be found using a velocity triangle:

$$\mathbf{u} = \mathbf{u}_e \cdot \mathbf{t} + \mathbf{u} \cdot \mathbf{n}.$$

Pressure, density and energy are calculated using entropy.

3.3.5 Discretization of turbulent equations

Turbulent models, including the Spalart-Allmaras equation, are discretized differently than the fluid flow equations: are not solved using the finite volume method, but rather the finite difference method. Thus, the differential form, not integration form, of the equation is solved and each derivative is expressed in curvilinear coordinates through application of Eq. (3.15). It should be noted that field points are still located at the centroids of the hexahedrons. Schemes used for each term in Eq. (3.14) are detailed in the following sections.

A fully implicit scheme is used to discretize the time derivative and a segregated solve is used to advance the turbulence and fluid flow equations in time, as will also be detailed in the following sections.

Segregated solve

It can be seen that solving the Favre-averaged fluid flow equations requires that another equation be solved, in this case the SA equation. These equations are inherently

coupled, since the momentum and energy equations depend on the eddy viscosity obtained from solving the turbulence equation and the turbulence equation itself depends on fluid quantities such as velocity. A common way to approach this problem is to solve each set of equations separately at each time step, which is what is done in syn3D. The procedure is described by the following steps:

1. Solve the turbulence equations using the latest solution field.
2. Update the eddy viscosity and modified eddy viscosity $\hat{\nu}$.
3. Solve the fluid flow equations using the same solution field as well as the updated viscosities.
4. Update the solution field.

This type of procedure is commonly referred to as a segregated solve.

Discretization of production and destruction terms

The production and destruction terms do not involve derivatives other than the vorticity Ω, which itself requires computation of velocity gradients. The velocity gradients are calculated at the vertices using Eq. (3.19). The vorticity magnitude is then evaluated at the vertices and interpolated to the hexahedron centroid through an arithmethic average of all eight vertices of the hexahedron. The remaining computations all rely on readily available quantities stored at the field point of interest.

Discretization of advection term

The advection term involves the gradient of the modified eddy viscosity, which can be expressed in curvilinear coordinates as:

$$\begin{aligned}\mathbf{u}\cdot(\nabla\hat{\nu}) &= u_x\frac{\partial\hat{\nu}}{\partial x}+u_y\frac{\partial\hat{\nu}}{\partial y}+u_z\frac{\partial\hat{\nu}}{\partial z}\\ &= u\left[\frac{\partial\xi}{\partial x}\frac{\partial\hat{\nu}}{\partial\xi}+\frac{\partial\eta}{\partial x}\frac{\partial\hat{\nu}}{\partial\eta}+\frac{\partial\zeta}{\partial x}\frac{\partial\hat{\nu}}{\partial\zeta}\right]\\ &+ v\left[\frac{\partial\xi}{\partial y}\frac{\partial\hat{\nu}}{\partial\xi}+\frac{\partial\eta}{\partial y}\frac{\partial\hat{\nu}}{\partial\eta}+\frac{\partial\zeta}{\partial y}\frac{\partial\hat{\nu}}{\partial\zeta}\right]\\ &+ w\left[\frac{\partial\xi}{\partial z}\frac{\partial\hat{\nu}}{\partial\xi}+\frac{\partial\eta}{\partial z}\frac{\partial\hat{\nu}}{\partial\eta}+\frac{\partial\zeta}{\partial z}\frac{\partial\hat{\nu}}{\partial\zeta}\right],\end{aligned}$$

which can be rewritten as

$$\mathbf{u}\cdot(\nabla\hat{\nu}) = +\frac{\partial\hat{\nu}}{\partial\eta}q_\eta+\frac{\partial\hat{\nu}}{\partial\zeta}q_\zeta, \tag{3.20}$$

where

$$q_{\xi_k} = u\frac{\partial\xi_k}{\partial x}+v\frac{\partial\xi_k}{\partial y}+w\frac{\partial\xi_k}{\partial z}.$$

Each term in the summation, which can be seen as the convective flux in each curvilinear direction, is discretized using a first-order upwinding scheme. For instance, the first term on the right-hand side of Eq. (3.20) is approximated as:

$$\frac{\partial\hat{\nu}}{\partial\xi}q_\xi = q^+_{\xi_{i,j,k}}(\hat{\nu}_{i,j,k}-\hat{\nu}_{i-1,j,k})+q^-_{\xi_{i,j,k}}(\hat{\nu}_{i+1,j,k}-\hat{\nu}_{i,j,k}),$$

where

$$\begin{aligned}q^+ &= \frac{1}{2}(q+|q|)\\ q^- &= \frac{1}{2}(q-|q|).\end{aligned}$$

This scheme leads to a three-point stencil in each curvilinear direction.

Discretization of diffusion term

The diffusion term involves the divergence of the gradient. The gradient is first expressed in curvilinear coordinates similarly to the gradient appearing in the convective term. For the sake of conciseness, let δ_x represent only discretization for the x derivative, namely:

$$\frac{\partial \hat{\nu}}{\partial x} = \delta_x = \left[\frac{\partial \xi}{\partial x}\frac{\partial \hat{\nu}}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial \hat{\nu}}{\partial \eta} + \frac{\partial \zeta}{\partial x}\frac{\partial \hat{\nu}}{\partial \zeta} \right].$$

The divergence operator is then also expressed in curvilinear coordinates. Again, looking only at the derivative in the x direction and letting $\Gamma = (\nu + \hat{\nu})$ yields:

$$\frac{\partial}{\partial x}\left[\Gamma \frac{\partial \hat{\nu}}{\partial x} \right] = \frac{\partial \xi}{\partial x}\frac{\partial (\Gamma \delta_x)}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial (\Gamma \delta_x)}{\partial \eta} + \frac{\partial \zeta}{\partial x}\frac{\partial (\Gamma \delta_x)}{\partial \zeta}.$$

Each term on the right-hand side above involves mixed derivatives of the form:

$$\frac{\partial}{\partial \xi_i}\left[\Gamma \frac{\partial \xi_j}{\partial x}\frac{\partial \hat{\nu}}{\partial \xi_j} \right], \tag{3.21}$$

with unequal indices i and j. Such terms vanish for orthogonal grids and their magnitudes relative to terms with equal indices depend on the grid non-orthogonality [25]. To reduce computations and to obtain a smaller stencil, these cross terms are simply ignored. Consequently, the expression can be expanded to:

$$\frac{\partial}{\partial x}\left[\Gamma \frac{\partial \hat{\nu}}{\partial x} \right] = \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi}\left[\Gamma \frac{\partial \hat{\nu}}{\partial \xi}\frac{\partial \xi}{\partial x} \right] + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\left[\Gamma \frac{\partial \hat{\nu}}{\partial \eta}\frac{\partial \eta}{\partial x} \right] + \frac{\partial \zeta}{\partial x}\frac{\partial}{\partial \zeta}\left[\Gamma \frac{\partial \hat{\nu}}{\partial \zeta}\frac{\partial \zeta}{\partial x} \right].$$

A flux-like discretization is then used to approximate the outer derivative with respect

to curvilinear coordinates such that:

$$\begin{aligned}\frac{\partial}{\partial x}\left[\Gamma\frac{\partial\hat{\nu}}{\partial x}\right] = &\left(\frac{\partial\xi}{\partial x}\right)_{i,j,k}\left[\left(\Gamma\frac{\partial\hat{\nu}}{\partial\xi}\frac{\partial\xi}{\partial x}\right)_{i+\frac{1}{2},j,k} - \left(\Gamma\frac{\partial\hat{\nu}}{\partial\xi}\frac{\partial\xi}{\partial x}\right)_{i-\frac{1}{2},j,k}\right] \\ &+ \left(\frac{\partial\eta}{\partial x}\right)_{i,j,k}\left[\left(\Gamma\frac{\partial\hat{\nu}}{\partial\eta}\frac{\partial\eta}{\partial x}\right)_{i,j+\frac{1}{2},k} - \left(\Gamma\frac{\partial\hat{\nu}}{\partial\eta}\frac{\partial\eta}{\partial x}\right)_{i,j-\frac{1}{2},k}\right] \\ &+ \left(\frac{\partial\zeta}{\partial x}\right)_{i,j,k}\left[\left(\Gamma\frac{\partial\hat{\nu}}{\partial\zeta}\frac{\partial\zeta}{\partial x}\right)_{i,j,k+\frac{1}{2}} - \left(\Gamma\frac{\partial\hat{\nu}}{\partial\zeta}\frac{\partial\zeta}{\partial x}\right)_{i,j,k-\frac{1}{2}}\right].\end{aligned}$$

Metrics and the diffusion coefficient at the interfaces are obtained using the usual arithmethic average of the two cells that share the interface and the derivatives are obtained using centered differences. For example, the derivative $\partial\hat{\nu}/\partial\xi$ at $i+\frac{1}{2}$ is written as:

$$\left(\frac{\partial\hat{\nu}}{\partial\xi}\right)_{i+\frac{1}{2},j,k} = \hat{\nu}_{i+1,j,k} - \hat{\nu}_{i,j,k}.$$

Discretization of the diffusion then results in a three-point stencil in each direction.

Discretization of anti-diffusion term

The last term to be discretized is the anti-diffusion, which can be expanded as follows if constant coefficients are ignored:

$$(\nabla\hat{\nu})^2 = \left[\left(\frac{\partial\hat{\nu}}{\partial x}\right)^2 + \left(\frac{\partial\hat{\nu}}{\partial y}\right)^2 + \left(\frac{\partial\hat{\nu}}{\partial z}\right)^2\right].$$

Each derivative, which results in three terms when expressed in curvilinear coordinates, must then be squared. Similar to the diffusion term, products involving unequal directions are

neglected, resulting in:

$$
\begin{aligned}
(\nabla\hat{\nu})^2 = &\left(\frac{\partial\hat{\nu}}{\partial\xi}\right)_{i,j,k}\left[\left(\frac{\partial\xi}{\partial x}\right)^2+\left(\frac{\partial\xi}{\partial y}\right)^2+\left(\frac{\partial\xi}{\partial z}\right)^2\right]_{i,j,k}\\
&+\left(\frac{\partial\hat{\nu}}{\partial\eta}\right)_{i,j,k}\left[\left(\frac{\partial\eta}{\partial x}\right)^2+\left(\frac{\partial\eta}{\partial y}\right)^2+\left(\frac{\partial\eta}{\partial z}\right)^2\right]_{i,j,k}\\
&+\left(\frac{\partial\hat{\nu}}{\partial\zeta}\right)_{i,j,k}\left[\left(\frac{\partial\zeta}{\partial x}\right)^2+\left(\frac{\partial\zeta}{\partial y}\right)^2+\left(\frac{\partial\zeta}{\partial z}\right)^2\right]_{i,j,k}.
\end{aligned}
$$

The metrics are required at the field point itself, thus no interpolation is needed. The derivative with respect to curvilinear directions is, once again, obtained with centered differences. The derivative with respect to ξ can then be written as:

$$
\left(\frac{\partial\hat{\nu}}{\partial\xi}\right)_{i,j,k} = \frac{\hat{\nu}_{i+1,j,k}-\hat{\nu}_{i-1,j,k}}{2}.
$$

Temporal discretization

The time derivative is discretized using the backward Euler approach mentioned in Section 3.1.2, which when applied to the S-A model yields the following equation for each field point:

$$
\left[\frac{1}{\Delta t^n}+\frac{\partial R^n}{\partial\hat{\nu}}\right]\Delta\hat{\nu}^n = -R^n.
$$

Due to the spatial discretization previously described, the residual at each point (i,j,k) depends on values of $\hat{\nu}$ at (i,j,k) as well as at the following six neighboring field points: $(i+1,j,k)$, $(i-1,j,k)$, $(i,j+1,k)$, $(i,j-1,k)$, $(i,j,k+1)$ and $(i,j,k-1)$. Consequently, the flux Jacobian, $\partial R^n/\partial\hat{\nu}$ is a very sparse matrix with at most seven entries on each row, including the diagonal. This system can be solved using the Alternating Direction Implicit (ADI) technique detailed in [7], which factorizes the implicit operator into three factors. Each factor contains the linearisation for one particular curvilinear direction in the computational space. The matrix for each factor is a tridiagonal matrix, which can be inverted directly

at a low computational cost using tridiagonal matrix algorithm, also known as the Thomas algorithm [26].

Boundary conditions

Implementation of boundary conditions for the turbulence model also benefits from the ghost cell structure previously described. Solid wall boundary conditions are implemented by setting the first level ghost cell as:

$$\hat{\nu}_1 = -\hat{\nu}_2,$$

which will result in an arithmethic average of zero at the interface.

Symmetry planes are implemented in exactly the same way as for the fluid flow equations, and far-field boundary conditions are implemented as follows:

$$\begin{aligned} \hat{\nu}_1 &= \hat{\nu}_\infty \quad \text{ingoing flow} \\ \hat{\nu}_1 &= \hat{\nu}_2 \quad \text{outgoing flow,} \end{aligned}$$

where the flow direction is determined by the sign of $\mathbf{u} \cdot \mathbf{n}$ at the boundary.

3.3.6 Wall distance calculation

Syn3D can calculate wall distances either through the brute force approach or the Poisson approach detailed in Section 3.2. With the brute force approach, distances between each cell center and the nearest face center is calculated. As mentioned in Section 3.2, this can be a source of error. However, structured meshes are typically constructed in such a way that grid lines are orthogonal to solid wall boundaries, in which case the error vanishes.

The Poisson equation (Eq. (3.10)) is solved using the Galerkin method, which belongs

to the class of finite element techniques. A detailed discussion on using the Galerkin method for solving PDEs can be found in [27] and only an overview is given here.

The first step is to rewrite the Poisson equation in the following weak form:

$$\int_{\Omega} (\nabla w \cdot \nabla \phi - w) \ d\Omega - \int_{\partial\Omega} [(\mathbf{n} \cdot \nabla \phi) w] \ dS,$$

where w is a "trial" function. Since this particular problem only requires homogeneous Dirichlet conditions to be imposed, the second term vanishes. Linear Lagrange shape functions are used as basis functions for ϕ as well as w. Integrals are approximated using a two-point Gaussian quadrature. This scheme is second-order accurate and results in a linear equation, which is solved using the conjugate gradient method with an algebraic multigrid preconditioner. The PETSc library was used to assemble and solve the linear system [28]. Finally, d is computed at the cell centers by using the shape functions.

3.4 NX Flow

NX™ Flow is a commercial software part of the Simcenter™ solution portfolio [29]. It is developed at Maya Heat Transfer Technologies in Montreal. The program has the following characteristics:

— Dimensional

— Unstructured

— Vertex-centered

— Uses a control-volume based finite element (CVFE) method

— Pressure-based

The mesh structure is explained in Section 3.4.1, an overview of the CVFE method is given in Section 3.4.2, the characteristics of a pressure-based code are given in Section 3.4.3, the discretization is described in Section 3.4.4 and wall distance calculation methods are

presented in Section 3.4.5.

3.4.1 Computational grid

As opposed to syn3D, NX Flow solves the governing equations on a computational domain tessellated with unstructured grids of various element types.. This allows for greater flexibility in adapting the grid to arbitrarily complex domains. The grids are typically composed of hexahedrons and tetrahedrals, although pyramids and wedges are also often used as transition elements. Thus, significant bookkeeping is required in order to keep track of the connectivity compared to structured grids. The reader is referred to [7] for a more detailed discussion on unstructured grids.

Another difference between syn3D and NX Flow is that the latter locates the field points at the vertices, not the element centroids, making it a vertex-centered, or cell-vertex, code. However, it still is a finite volume code, which raises the question of how the CVs are constructed. This is done by generating a so-called dual mesh, which is similar to the auxiliary CV construction in syn3D as described in Section 3.3.4. In two dimensions, as depicted in Figure 3.5, a CV is constructed around a given vertex by looping over the edges containing the vertex and joining the midpoints of those edges to the face centers. Using finite volume terminology, these lines (one-dimensional surfaces) are referred to as *integration surfaces*. It goes without saying that distinguishing between a control volume and an element is crucial. One speaks of a CV sector or element sector when referring to the portion of a CV contained in a single element. It can also be said that a given element is divided into N sectors, where N is the number of vertices this element is made up of.

In three dimensions, sectors are generated by joining the element centroid to the element face centers, in combination with lines joining element face centers to the mid-points of the element edges. Again, the sub-division of an element yields as many sectors as vertices on the element.

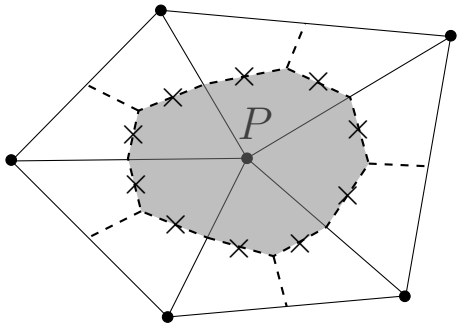


Figure 3.5 – Dual control volume of a cell-vertex scheme in two dimensions. Integration surfaces are shown as dashed lines, the CV around P is filled in gray, elements are separated by solid lines, the field point is shown as a solid black circle and integration points are shown as crosses.

The terminology introduced here will be used for the remainder of this work of Section 3.4.

3.4.2 Control-volume based finite element method

The CVFE method was first proposed by Baliga and Patankar in [30], [31]. This new method borrows two characteristics from the finite element method (FEM):

1. the use of element-based linear Lagrangian interpolation functions (shape functions).
2. an element-by-element compilation of the coefficients in the discretized equations.

These two characteristics are briefly discussed in the following sections. Readers interested in a thorough discussion of the classical FEM applied to fluid flow problems are referred to [32].

Shape functions

The shape functions Ψ represent the variation of solution inside an element. This allows one to calculate the value of a field variable ϕ at an (s, t, u) coordinate within the

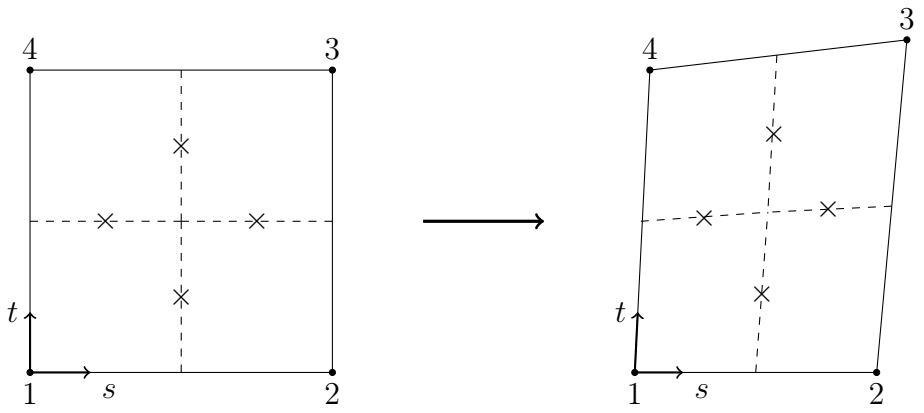


Figure 3.6 – General linear quadrilateral element (right) and its representation in the natural coordinate system (left)

element as follows:

$$\phi(s,t,u) = \sum_{n}^{N} \Psi(s,t,u)_n \phi_n,$$

where N is the number of vertices in the element, ϕ_n is the stored value of ϕ at vertex n. The s,t,u coordinate system is local to each element and is also referred to as the natural coordinate system in the FEM literature. Construction of shape functions in the x,y,z coordinate system results in complex algebraic expressions, which is why it is best to express them in terms of the natural coordinates. Moreover, the integration point locations for a given element type will always be at the same s,t,u coordinates. Values of s,t,u are only allowed to vary between 0 and 1. The natural coordinate system transformation is illustrated in Figure 3.6 for a quadrilateral element (two-dimensional). For such an element, the shape functions can be written as:

$$\Psi_1(s,t) = (1-s)(1-t)$$

$$\Psi_2(s,t) = s(1-t)$$

$$\Psi_3(s,t) = st$$

$$\Psi_4(s,t) = (1-s)t.$$

It is then straightforward to obtain the shape function derivatives with respect to local coordinates, which are given by:

$$\frac{\partial \phi}{\partial s_i} = \sum_n^N \frac{\partial \Psi_n}{\partial s_i} \phi_n,$$

where s_i represents the i-th component in the natural coordinate system.

Similarly to the generalized curvilinear coordinates, a transformation is necessary to calculate the shape function derivatives with respect to the global coordinates x, y, z. The position vector can be expressed as a function of the s, t, u coordinates as:

$$\mathbf{x}(s,t,u) = \sum_n^N \Psi(s,t,u)_n \mathbf{x}_n.$$

This leads to the following metric formulation:

$$\left.\frac{\partial \mathbf{x}}{\partial s_j}\right|_{(s,t,u)} = \sum_n^N \left(\left.\frac{\partial \Psi_n}{\partial s_j}\right|_{(s,t,u)} \mathbf{x}_n \right). \tag{3.22}$$

The derivatives of ϕ with respect to the natural coordinates can be written in the following matrix form using the chain rule:

$$\begin{pmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} & \frac{\partial z}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} & \frac{\partial z}{\partial t} \\ \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} & \frac{\partial z}{\partial u} \end{pmatrix} \begin{bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \\ \frac{\partial \phi}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial \phi}{\partial s} \\ \frac{\partial \phi}{\partial t} \\ \frac{\partial \phi}{\partial u} \end{bmatrix}.$$

This system can be numerically inverted using Kramer's rule. The derivative $\partial\phi/\partial x$ at an integration point in the element can then be written as:

$$\left.\frac{\partial \phi}{\partial x_i}\right|_{ip} = \sum_n^N \left.\frac{\partial \Psi_n}{\partial x_i}\right|_{ip} \phi_n.$$

It should be noted that the evaluation of ϕ and its derivative inside an element can be

computed with second-order accuracy regardless of the shape of the element, which is a significant advantage of using this method – the approximations in Section 3.3 are at most second-order accurate, depending on the skewness and orthogonality of the grid.

Compilation of the coefficients

The CVFE method uses the integral form of the equations as discussed in Section 3.1, and fluxes through the integration surfaces around a control volume must be evaluated. Due to the dual mesh configuration, the integration surfaces as well as the integration points lie completely inside the elements. The surface integrals can then be calculated element-wise and accumulated to each control volume appropriately. The surface integrals are then guaranteed to be conservative. Moreover, because of the way the shape functions are constructed, the value of the flux at a given integration point depends on the field variables at all vertices in that element, regardless of whether the integration point belongs to their control volume. In other words, the stencil involved in the computation of a flux always includes all the vertices of the element in which the flux is computed.

It should be noted that boundaries of the domain require special treatment, since the surfaces are not located inside any element. This is discussed further below.

3.4.3 Pressure-based solver

The main difference between a density-based and a pressure-based solver is the dependent variable in the mass equation, which is density in the former and pressure in the latter. Because a large portion of users of NX Flow want to simulate flows involving liquids, which have a constant density, the developers of NX Flow chose the pressure-based approach. In the case of ideal gases, the density is updated using the ideal gas law at each iteration.

NX Flow employs the collocated variables approach, i.e. pressure and velocity are

stored at the same locations, proposed by Rhie and Chow [33] in order to avoid pressure checkerboarding.

3.4.4 Discretization of the governing equations

This section describes the numerical discretization of the terms in the mass, momentum, energy and turbulence equations.

NX Flow being a commercial code, offers a wide variety of options to its users. For the sake of conciseness, only the options used to solve the problems in this work are described.

Temporal discretization

NX Flow uses the fully-implicit first-order backward Euler approach to discretize the time derivative of all partial differential equations. It solves the mass and momentum equations together, which yields an updated pressure and velocity field.

The turbulence and energy equations are not coupled with the mass and momentum equations; they are solved separately in a segregated manner. The linear systems are solved using an iterative solver such as GMRES.

Discretization of convective fluxes

The convective flux of an arbitrary quantity over an integration surface can be written in the following general form:

$$\int_S (\rho \mathbf{u} \phi) \cdot \mathbf{n} \; dS = \dot{m}_{ip} \phi_{ip},$$

where $\dot{m}_{ip}$ is the mass flow rate through the integration surface and is evaluated using the shape functions along with a pressure correction term [33], [34].

The value of ϕ at the integration point also needs to be approximated in terms of the nodal values of ϕ. The default scheme for this is a first-order upwind scheme, in which ϕ_{ip} is approximated by the value of ϕ at the upstream vertex, i.e.:

$$\phi_{ip} = \phi_i,$$

where the upstream vertex is determined using the sign of the mass flux. This scheme leads to a stable and quick convergence at the expense of reduced accuracy.

Discretization of viscous fluxes

Computation of the diffusive fluxes requires evaluation of the diffusion coefficient and spatial derivatives at the integration points. This is simply done using the finite element shape functions. The gradient and diffusion coefficient at an integration point are then evaluated as:

$$\Gamma_{ip} = \sum_{n}^{N} \Psi_n|_{ip} \, \Gamma_n$$
$$\nabla\phi|_{ip} = \sum_{n}^{N} \nabla\Psi_n|_{ip} \, \phi_n.$$

Discretization of pressure term

The pressure gradient term is treated as a surface force via the divergence theorem, similarly to syn3D. The value of pressure must then be evaluated at all integration points, which is done using the shape functions.

Discretization of nodal gradients

Source terms sometimes require the computation of gradients at field points, as opposed to at an integration point. Such nodal gradients can also be computed using a form of the divergence theorem. The gradient of ϕ at vertex P can then be computed as follows:

$$(\nabla\phi)_P = \frac{1}{V}\sum_{ip}(\phi\mathbf{n})_{ip}.$$

The formula requires that ϕ be evaluated at integration points, which is done using the shape functions.

Boundary conditions

NX Flow was unfortunately not developed with external flows in mind and does not offer far-field boundary conditions in its catalogue. The user must then use internal flow BCs such as inlets and openings. The following boundary conditions are used in this work:

— Solid wall

— Inlet

— Opening

— Symmetry plane

Treatment of boundary conditions in a vertex-centered code is quite different from that of a cell-centered code since some field points may lie on the boundary, as depicted in Figure 3.7. Such points have integration surfaces corresponding to segments of the boundary. It should be noted that evaluation of fluxes over the domain then requires two loops: one over the elements and the interior integration surfaces as well as one over the boundary integration surfaces.

Symmetry planes require that there be no flux through the boundary. The implementation of such a boundary condition simply requires not calculating any fluxes through the

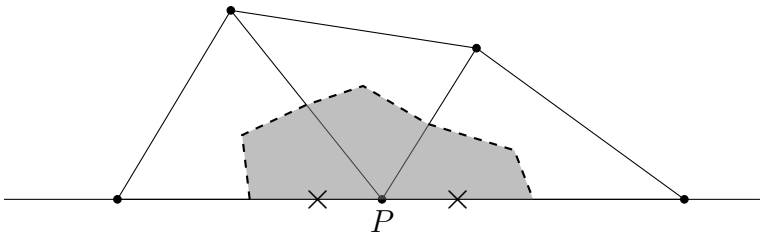


Figure 3.7 – Control volume for a boundary vertex P. Only boundary integration points are shown.

boundary integration surfaces.

In an inlet BC, the user specifies values for velocity and its direction as well as pressure, temperature and turbulence quantities. Since the integration point is on the boundary, convective and diffusive fluxes can be calculated using the user-specified values as the *ip* values, i.e. no interpolation is required. The gradients are calculated using the shape function derivatives, except that the vertex values are replaced with the integration point values for vertices on the boundary. The gradient of ϕ at a boundary integration point ip, b can then be expressed as a sum over the interior vertices, the ones not on a boundary, and boundary vertices:

$$\nabla\phi|_{ip,b} = \sum_{interior} \nabla\Psi_n|_{ip,b}\,\phi_n + \sum_{boundary} \nabla\Psi_n|_{ip,b}\,\phi_{ip,b}.$$

Opening BCs specify pressure, temperature and turbulence quantities. The velocity at boundary integration points is obtained by shape function interpolation, which determines whether the opening is an inflow or outflow. The advection term requires no special treatment. On the other hand, the diffusion flux is only calculated if the flow is outgoing. All openings used in this work should be treated as outflows.

Viscous wall boundary conditions can be implemented strongly by directly imposing prescribed values at the vertices (Dirichlet) or weakly by instead imposing a condition on the flux through specification of $\phi_{ip,b}$. A detailed discussion can be found in [35]. This BC is currently implemented weakly for the mass and momentum and strongly for the turbulence

equations.

3.4.5 Wall distance calculation

NX Flow also offers both the brute force approach and Poisson approach to calculate the wall distance. The brute force approach finds the nearest vertex for every field point, and the Poisson equation is solved with the finite volume method, exactly like the Navier-Stokes equations.

Chapter 4

Results

This chapter assesses the computational implementation accomplished by the author. The aim is to determine if the programming and computational implementation of the model is correct. The implementation in syn3D and NX Flow are validated on the following cases:

— Flat plate (syn3D & NX Flow)

— Two-dimensional bump (syn3D)

— Three-dimensional bump (syn3D)

— NACA 0012 (NX Flow)

— RAE 2822 (syn3D & NX Flow)

— High-lift configuration 30P30N (syn3D)

The first four cases are taken from the Turbulence Modeling Resource (TMR) [18], which provides a set of grids as well as results from established CFD codes, namely CFL3D and FUN3D, which are developed at NASA.

It should be noted that the difference in results using either wall distance methods (brute force and Poisson) was negligible, below 0.01%, and a comparison between the results obtained with the two methods is not shown.

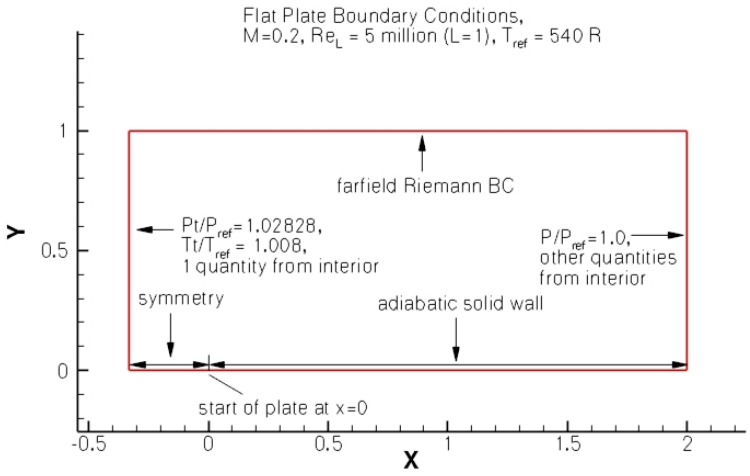


Figure 4.1 – Flow conditions and problem domain for the turbulent flat plate case [18].

4.1 Flat plate

The turbulent flow over a flat plate is investigated. All grids and results are available in [18] under the name: "2D Zero pressure gradient flat plate". The problem domain and flow conditions are shown in Figure 4.1. Notably, the Reynolds number for this case is five million. NX Flow does not offer far-field BCs and uses an inlet BC on the leftmost face and openings on the top and rightmost face. Moreover, the problem was run with a fixed density in NX Flow so as to reduce computational cost, since this is essentially an incompressible case.

The Turbulence Modeling Resource provides five grids, each finer than the next, in order to compare so-called mesh independent results and to perform a grid study. The goal of a grid study is to establish that the obtained results will not vary significantly by further refining the mesh, and thus that the obtained results are as accurate as possible.

The skin friction coefficient C_f at $x = 0.97$ and drag coefficient C_D are compared to CFL3D, a cell-centered finite volume code developed by NASA, in Table 4.1. These

Table 4.1 – Flat Plate (syn3D & NX Flow): Comparison of force coefficients on the finest grid.

Solver	C_D	C_f at $x = 0.97$
CFL3D	0.00286	0.00270
NX Flow	0.00288	0.00272
syn3D	0.00280	0.00269

coefficients are calculated as:

$$C_f = \frac{2\tau_w}{\rho_\infty U_\infty^2}$$
$$C_D = \frac{2D}{\rho_\infty U_\infty^2 A},$$

where D is the drag force and A the reference area. For this case, the reference area is the length of the plate times its width, i.e. the length of the mesh in the z direction. Table 4.1 tabulates the skin friction coefficient and drag coefficient values. All values are within 2% of each other. Reasons for discrepancies are given below. Figure 4.2 shows the residual convergence of the density residual and turbulence residual for all grids with syn3D and NX Flow. While the y-axis is scaled the same for all plots, the difference in number of iterations between NX Flow and syn3D requires the x-axis to be scaled differently between the two solvers; this is further discussed below. As expected, the coarser grids show faster convergence than the finer ones – this phenomenon is detailed in [7]. While the residual was never driven down to machine precision, differences in the obtained solution field were found to be marginal by further reducing the residual.

Table 4.2 compares the number of iterations and CPU time between syn3D and NX Flow for the medium grid, which shows that syn3D requires approximately two thousand times more iterations to converge than NX Flow, even though syn3D only converged the residual to 10^{-8}, as compared to NX Flow's 10^{-10}. While there exists many differences between the two solvers, the disparity can still largely be attributed to the time stepping

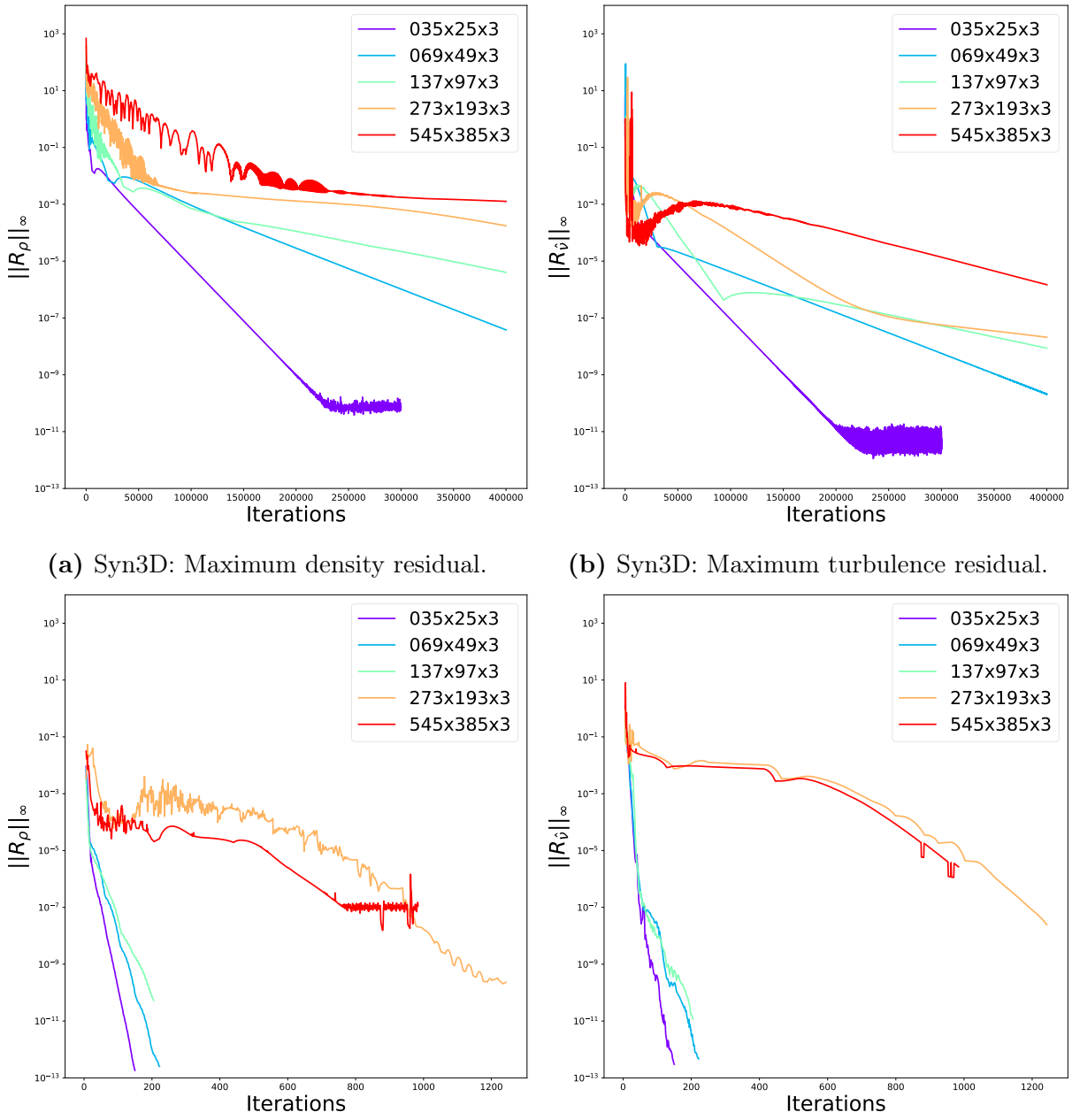
Table 4.2 – Flat Plate (syn3D & NX Flow): Convergence metrics on the medium grid (137x97).

Solver	Processors	Iterations	Time taken (min)	Time per iteration (sec)	Residual reduction
NX Flow	4	205	34	≈ 10	10^{-10}
syn3D	8	400000	156	≈ 0.023	10^{-8}

scheme. In other words, usage of an implicit scheme drastically reduces the number of iterations taken to converge. On the other hand, explicit iterations are much less computationally expensive and require less storage. It should also be noted that NX Flow does not solve the conservation of energy equation for this particular case, since it uses a fixed density. Moreover, multigrid was not used with syn3D for this case, which significantly accelerates convergence in most cases.

Simulations were run using eight cores with syn3D, since the program lies on a supercomputer. On the other hand, NX Flow simulations were run on a regular Windows desktop machine. While it is difficult to say how much time would have been taken by syn3D if it were run on the same Windows machine, reducing the number of CPU cores by half typically doubles the time taken, assuming the load was well balanced to begin with. In any case, the number of iterations would remain approximately the same.

Several plots have been generated to compare the results between all solvers. Unless otherwise noted, all results were obtained with the finest grid. Figure 4.3 compares the skin friction along the plate, which is maximal at the leading edge and then gradually decreases. The skin friction coefficient is a measure of the velocity gradient at the wall. NX Flow and syn3D display slight over-prediction and under-prediction respectively. It should be noted that the leading edge is in fact a singularity, and different codes handle this differently due to the varying boundary condition implementations, as explained in [18]. Thus, the sharp increase and decrease exhibited by the syn3D curve should be disregarded for the sake of comparison.



(a) Syn3D: Maximum density residual.

(b) Syn3D: Maximum turbulence residual.

(c) NX Flow: Maximum density residual.

(d) NX Flow: Maximum turbulence residual.

Figure 4.2 – Flat Plate (syn3D & NX Flow): Convergence of flow and turbulence variables on various grid sizes.

Figure 4.4 shows contour plots of the dimensionless eddy viscosity $\frac{\mu_t}{\mu_\infty}$ and Figure 4.5 shows line plots of the dimensionless eddy viscosity. As expected, the eddy viscosity increases gradually as the wall distance increases, as turbulence effects are more significant in the log-law region, and then diminishes as it reaches the far-field region. This is in accordance

with the law of the wall, which states that turbulent stresses are minimal in the viscous sublayer and maximal in the log-law layer. Moreover, the eddy viscosity, initially close to zero, increases with x due to production near the wall as well as advection.

Figures 4.6 and 4.7 show profiles of the dimensionless velocity u/u_∞ and u^+ respectively. The velocity profile shows a sharp gradient near the wall, typical of turbulent flows. Moreover, the height of the boundary layer, the region where velocity is reduced as compared to the far-field, grows with increasing x. It can also be seen that the u^+ profile agrees with the theory: u^+ follows the $u^+ = y^+$ curve in the viscous sublayer and the log-law curve in the log-law region.

Overall, there is excellent agreement between NX Flow and CFL3D and some level of discrepancy between syn3D and CFL3D. The source of discrepancy is easily seen in Figure 4.5b: the eddy viscosity jumps abruptly right at the leading edge for syn3D but not for the other two. It should be noted that the spike occurs at the interface between block boundaries, which could be the cause of this phenomenon. These blocks thus have different boundary conditions at their $y = 0$ face. Moreover, the jump does not occur if the domain only includes the plate, i.e. starts at $x = 0$.

Figure 4.8 shows the dependence of C_D and C_f on the grid spacing $h = 1/N^2$, where N is the number of grid points. It can be seen that there is still significant differences in results between the second finest and finest grid for syn3D as compared to CFL3D and FUN3D – the latter is a vertex-centered code also developed by NASA. Thus, syn3D may not have reached the asymptotic range of convergence. Moreover, the importance of a grid study can be seen from these plots: whereas results on the fine grid are mostly similar, they are quite different on the coarser grids, with some being over-predicted and others under-predicted compared to the asymptotic values.

The skin friction coefficient obtained by NX Flow is closer to that of FUN3D run in incompressible mode, which is higher than the compressible value. This explains the

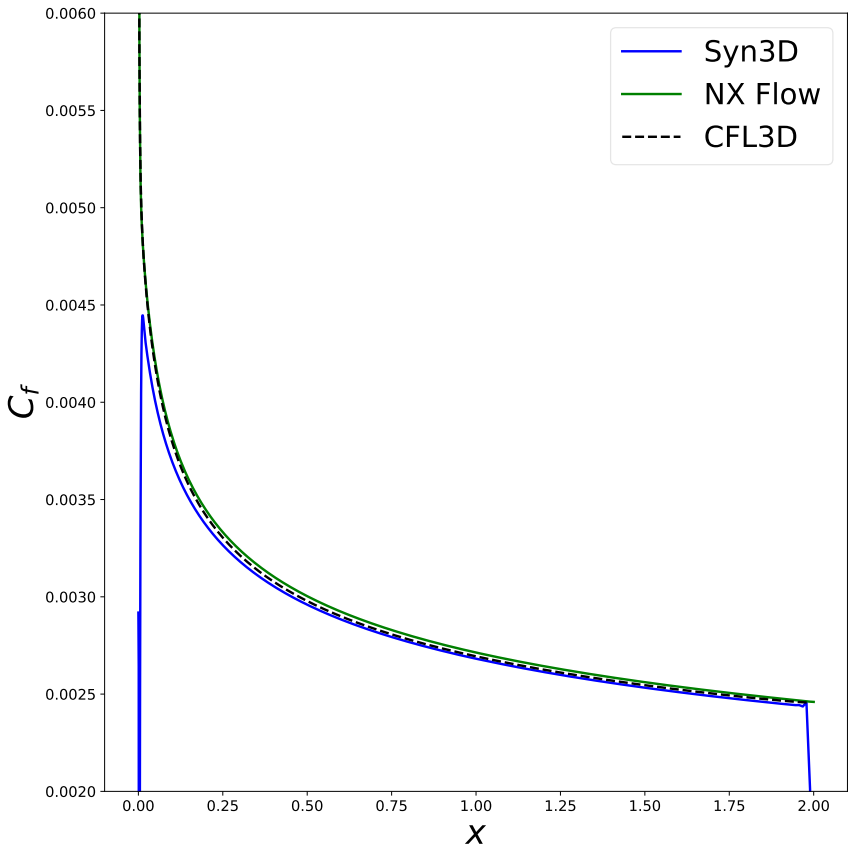
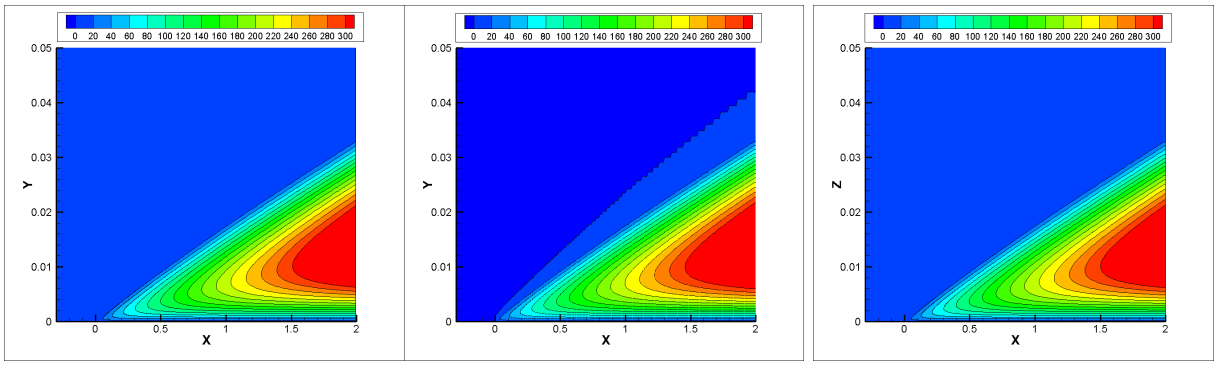


Figure 4.3 – Flat Plate (syn3D & NX Flow): Coefficient of skin friction along the plate.

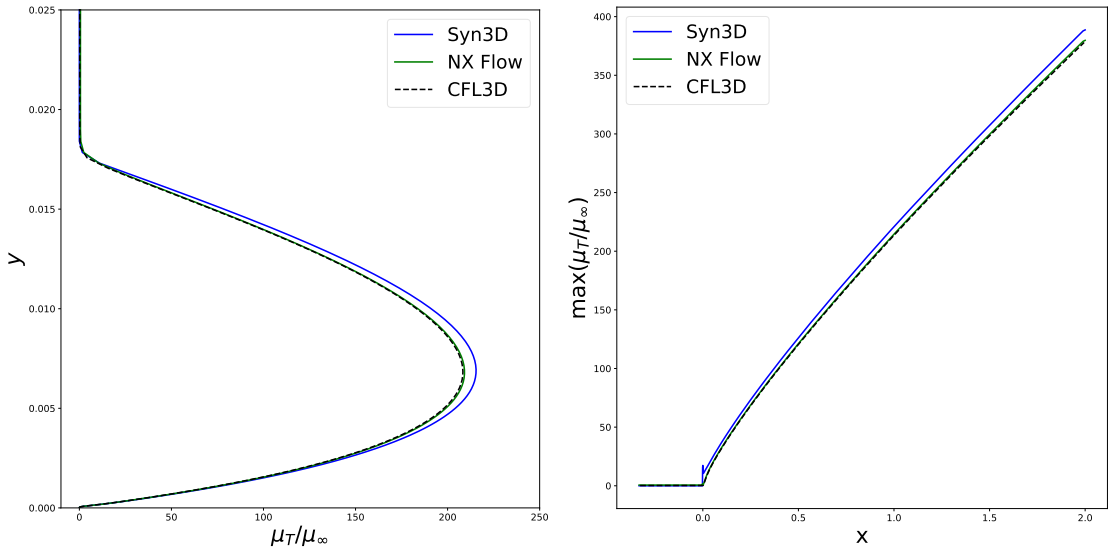


(a) CFL3D **(b)** syn3D **(c)** NX Flow

Figure 4.4 – Flat Plate (syn3D & NX Flow): Contours of non-dimensionalized eddy viscosity

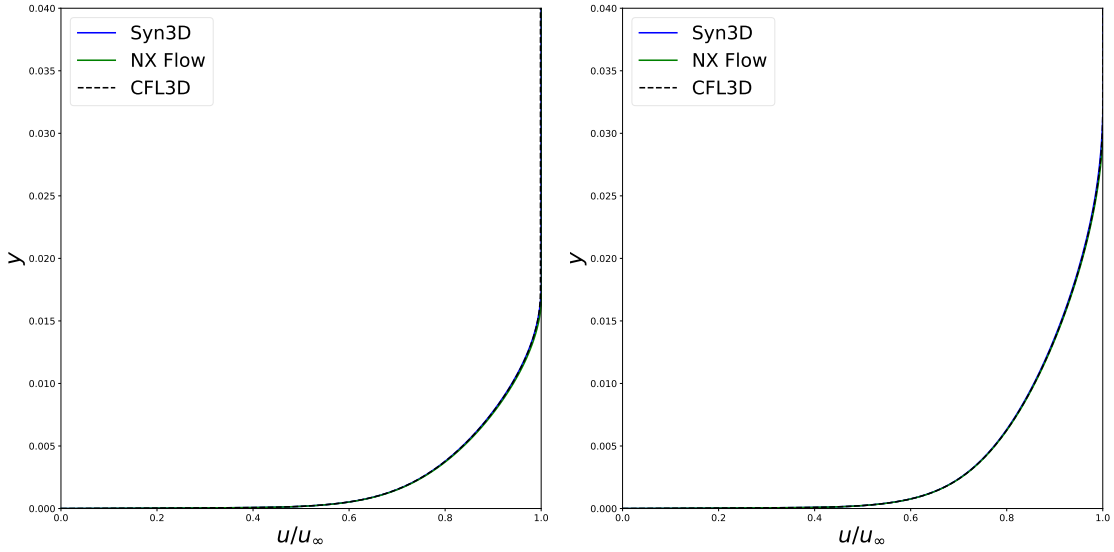
over-prediction in the skin friction coefficient as well as drag.

Figures 4.9 and 4.10 show the variation of skin friction coefficient along the plate and profiles of eddy viscosity and velocity with grid size respectively with syn3D and Figures 4.11



(a) Nondimensional eddy viscosity at x=0.97 **(b)** Maximum $\frac{\mu_t}{\mu_\infty}$ in the boundary layer

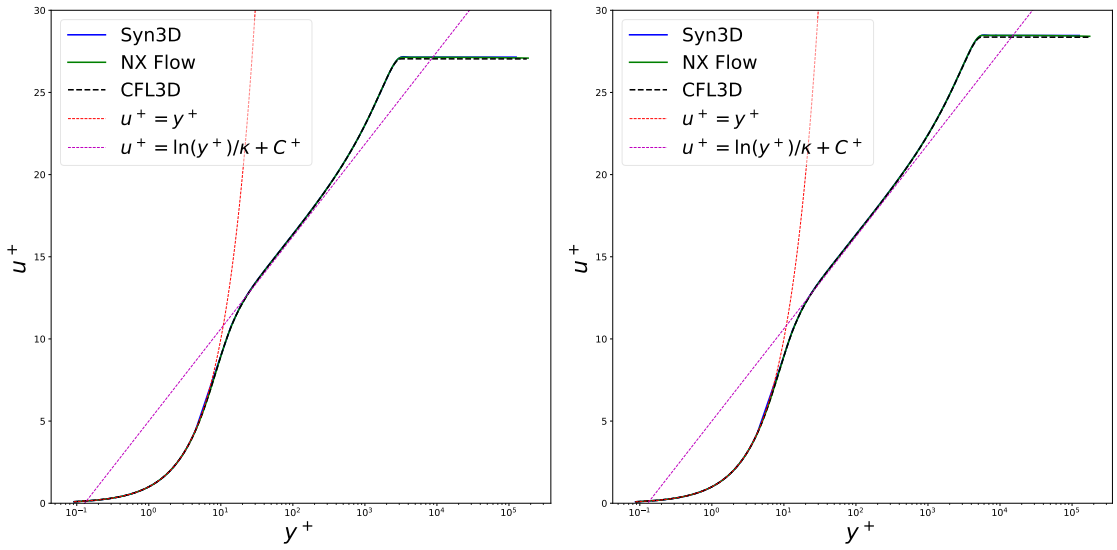
Figure 4.5 – Flat Plate (syn3D & NX Flow): Dimensionless eddy viscosity line plots.



(a) $x = 0.97$ **(b)** $x = 1.90$

Figure 4.6 – Flat Plate (syn3D & NX Flow): $\frac{U}{U_\infty}$ profiles in the boundary layer.

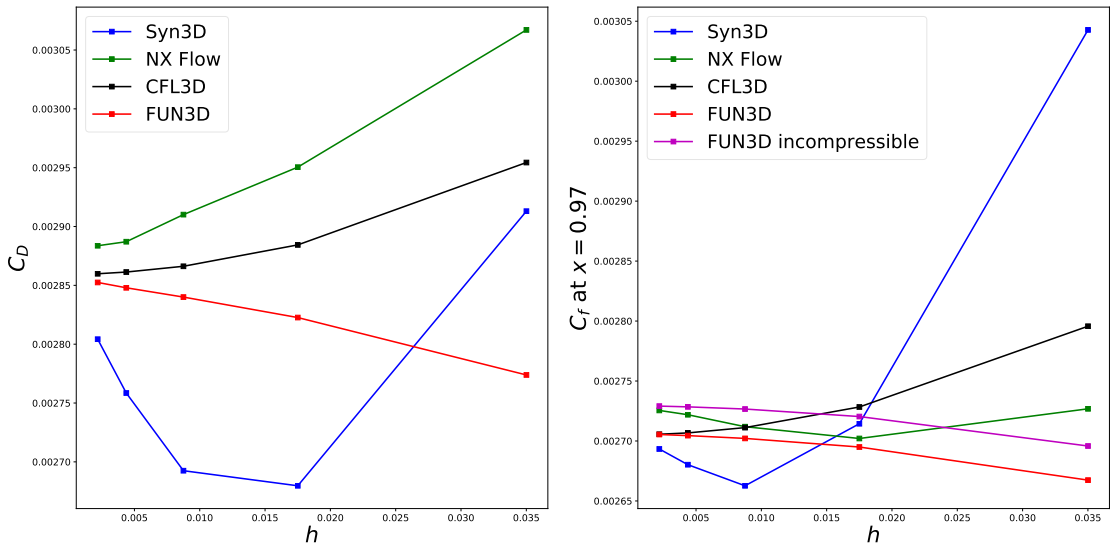
and 4.12 show the variation with NX Flow. These plots show that the solution field seems to converge to a unique solution, as opposed to oscillating back and forth between different values. As expected, solutions on coarser grids display increased dissipation; the discretization errors scale with the grid spacing and typically lead to increased diffusion. Moreover,



(a) $x = 0.97$ **(b)** $x = 1.90$

Figure 4.7 – Flat Plate (syn3D & NX Flow): u^+ vs. y^+ profiles in the boundary layer. Law of the wall is also shown for comparison.

the artificial dissipation parameters in syn3D were not tuned for each individual grid, and they lead to significantly increased dissipation on coarser grids. Nevertheless, the grid study provides an ability to determine the minimum required grid size to ensure sufficiently accurate values of the drag coefficient to within five drag counts. In the case of NX Flow, the 35×25 grid was required, while syn3D demanded a grid of 69×49. A study of the choice of artificial dissipation may reduce the grid density requirement, however, this was not part of the work.



(a) Coefficient of drag. **(b)** Coefficient of Skin Friction at x=0.97.

Figure 4.8 – Flat Plate (syn3D): Force coefficients for various grid sizes.

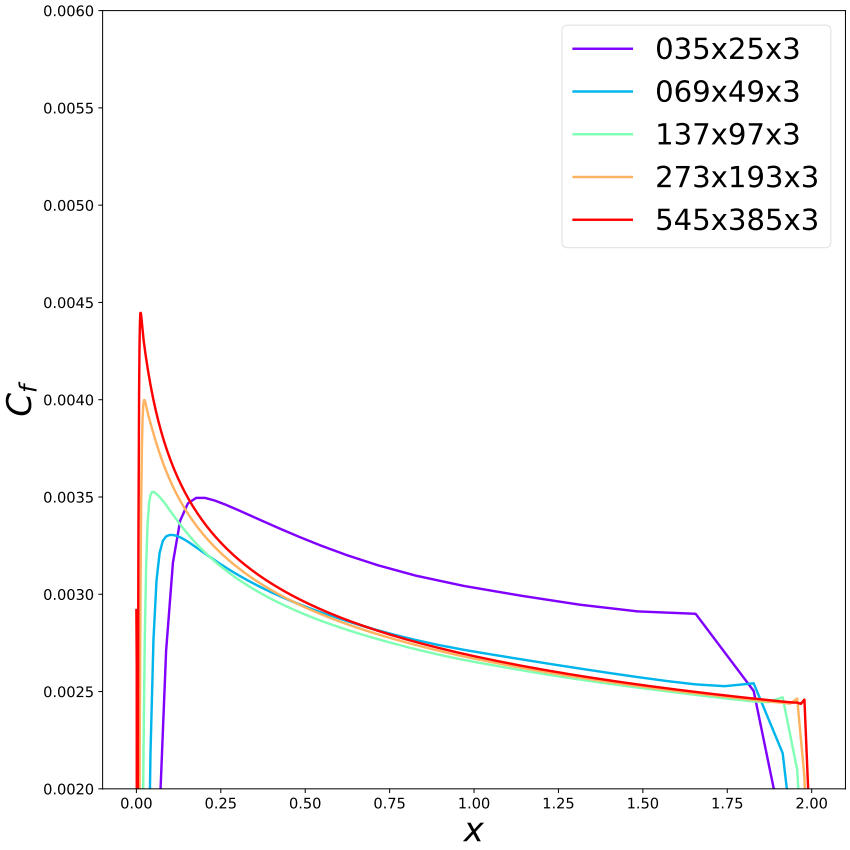
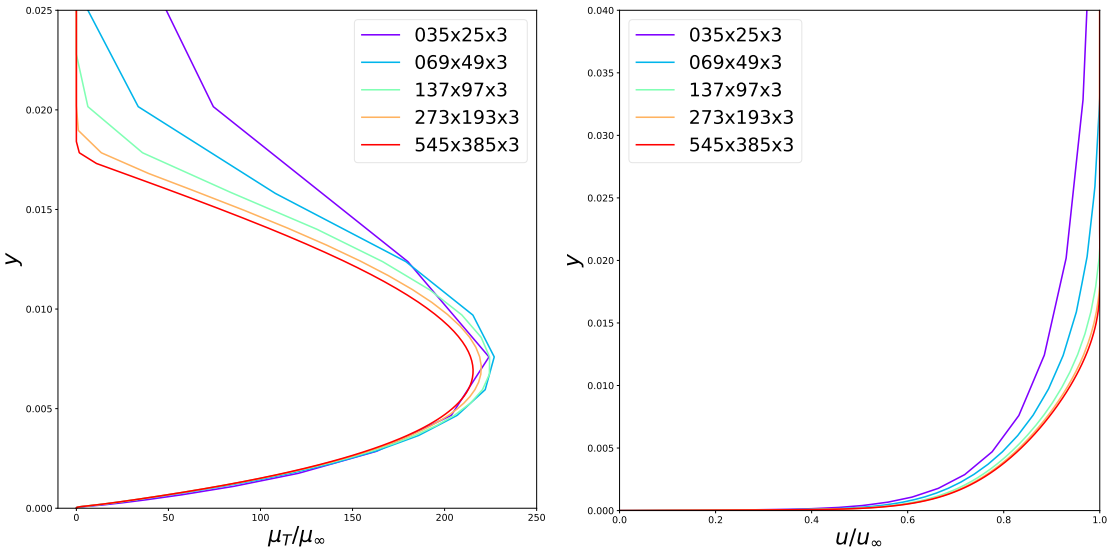


Figure 4.9 – Flat Plate (syn3D): Coefficient of skin friction on various grids.



(a) Dimensionless eddy viscosity **(b)** Dimensionless velocity

Figure 4.10 – Flat Plate (syn3D): Profiles at $x = 0.97$ for various grid sizes.

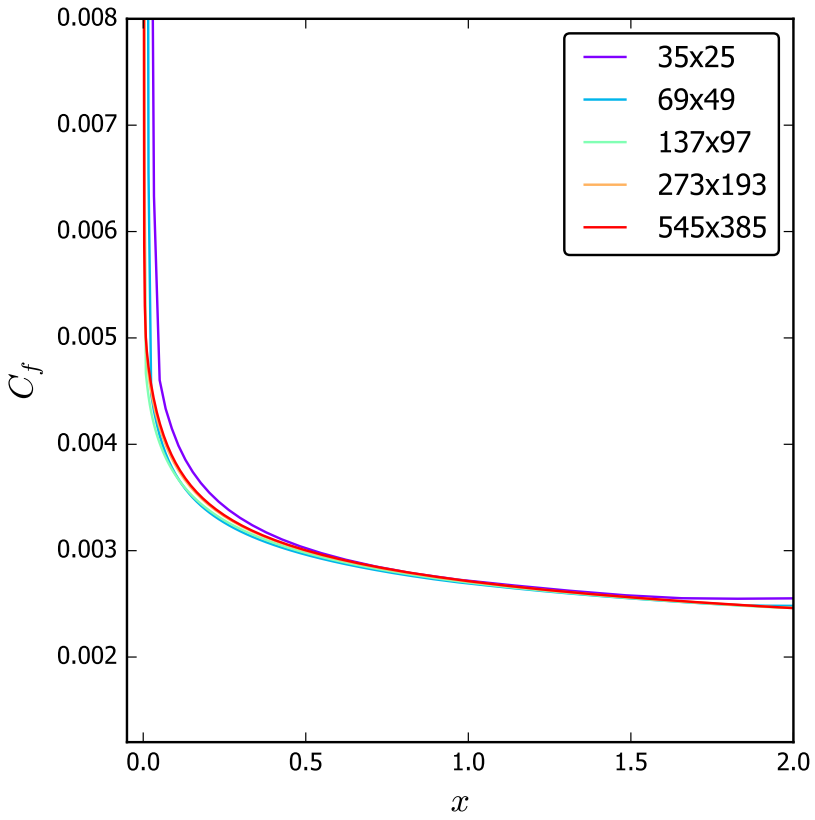
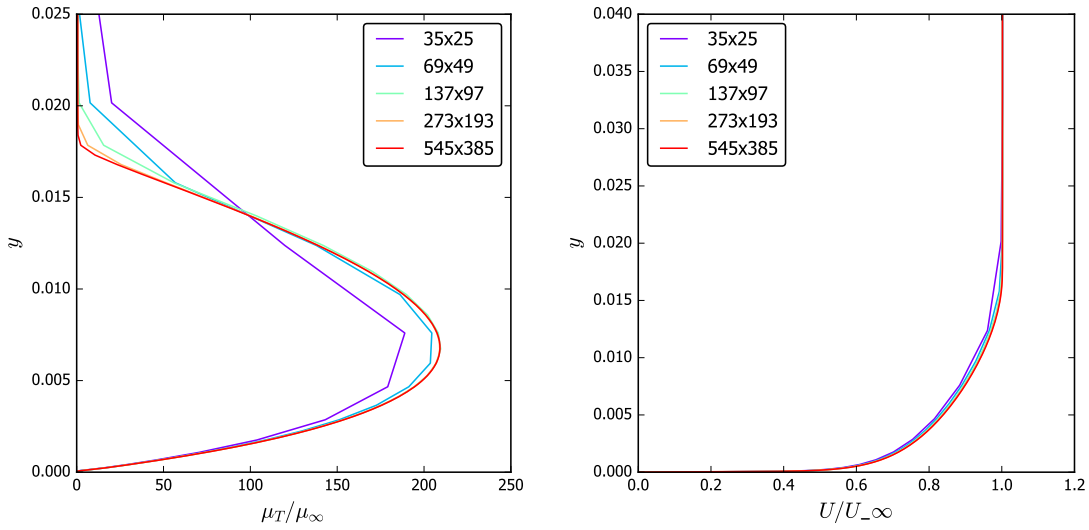


Figure 4.11 – Flat Plate (NX Flow): Coefficient of skin friction on various grids.



(a) Dimensionless eddy viscosity **(b)** Dimensionless velocity

Figure 4.12 – Flat Plate (NX Flow): Profiles at $x = 0.97$ for various grid sizes.

4.2 Two-dimensional bump-in-channel

Upon completion of the validation of the flow over a flat-plate, the turbulent flow on a two-dimensional bump-in-channel case was investigated. This case is also available on [18] under the name "2D Bump-in-channel". This case was only run with syn3D due to time constraints; setting up a simulation and finding the suitable combination of parameters to obtain convergence is non-trivial.

The problem domain and flow conditions are shown in Figure 4.13. The Reynolds number for this case is of three million. It is referred to as two-dimensional due to the shape of the bump not depending upon the z coordinate, as opposed to the three-dimensional bump in Section 4.3.

This case is different from the previous flat plate case because it involves wall curvature, which induces a pressure gradient. The TMR website also provides five grids for this case.

Table 4.3 tabulates the drag coefficient, the lift coefficient C_L as well as the contrib-

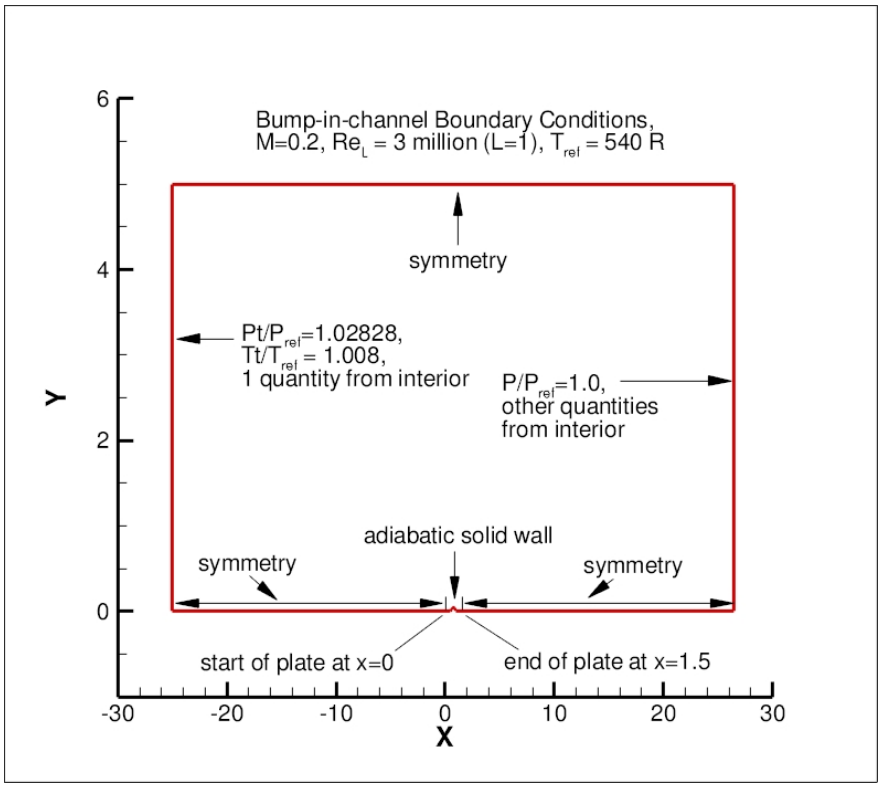


Figure 4.13 – Flow conditions and problem domain for the turbulent two-dimensional bump case [18].

Table 4.3 – 2D Bump (syn3D): Comparison of force coefficients for the finest two-dimensional bump.

Solver	C_L	C_D	C_{Dv}	C_{Dp}
CFL3D	0.0249	0.0036	0.0032	0.0004
syn3D	0.0251	0.0035	0.0031	0.0004

Table 4.4 – 2D Bump (syn3D): Comparison of skin friction coefficient at various locations.

	C_f		
Solver	$x = 0.75$	$x = 0.6321975$	$x = 0.8678025$
CFL3D	0.00615	0.00519	0.002680
syn3D	0.00610	0.00519	0.002835

utors to the drag coefficient C_{Dv} and C_{Dp}, which represent the contributions due to viscous forces and pressure forces respectively. Table 4.4 tabulates the skin friction coefficient probed at various locations. These tables show good agreement between CFL3D and syn3D.

Figure 4.14 shows the convergence for all grids. Reduction of the residual is quite slow for this particular problem, with the finest grid taking over 3 days on 64 processors to complete. In addition, a plateau of the residual at approximately 2×10^{-3} can be observed for the three coarsest grids, which is likely due to the interfaces between the symmetry planes and the solid wall. In fact, the maximum residual occurs at the field points with faces lying on that interface, which is also an interface between block boundaries.

Figure 4.15 compares the C_f distribution over the bump for the finest grid. The skin friction coefficient can be seen to start high, slowly decrease in the flat portion as it did for the flat plate, and then increase on the bump as it accelerates. The value of C_f is under-predicted syn3D until the peak of the bump, after which point it is over-predicted. Again, a discrepancy is seen for syn3D at the leading and trailing edge of the bump, which is due to the boundary condition implementation.

Figure 4.16 shows the C_p distribution. The trend in this plot is similar to the skin friction plot. The pressure decreases as the flow accelerates at the leading edge of the bump.

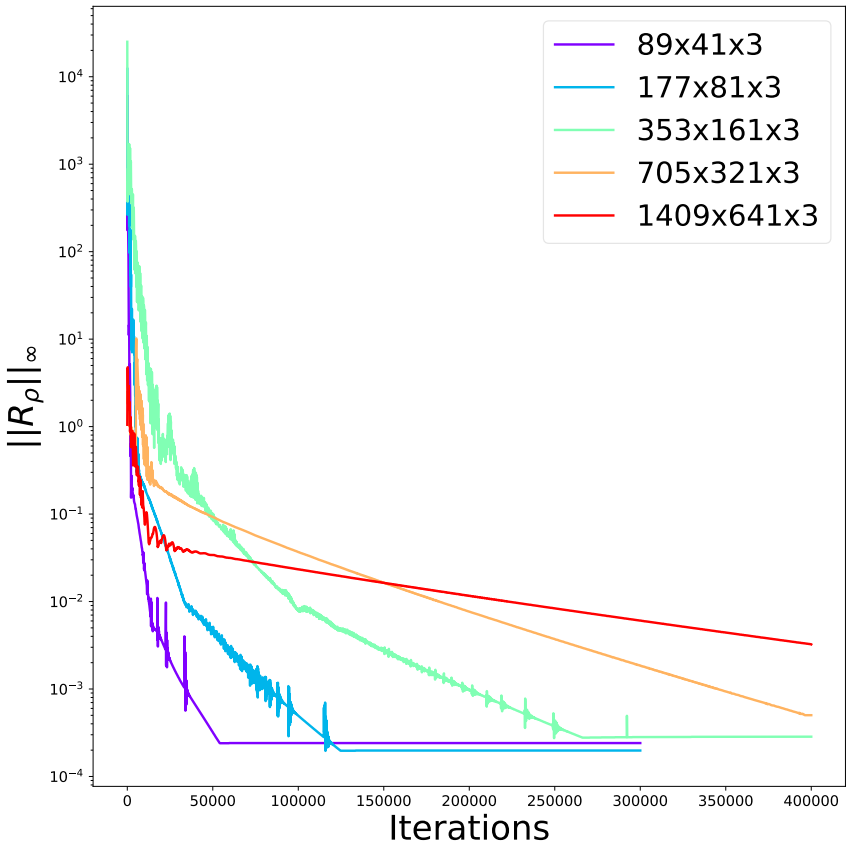


Figure 4.14 – 2D Bump (syn3D): Convergence of maximum density residual on various grid sizes.

The pressure then increases again once the peak is reached. The pressure coefficient is identical between syn3D and CFL3D, as is the drag force due to the pressure component C_{Dp} (see Table 4.3).

Figure 4.17 compares the contour of eddy viscosity over the bump. Both contour plots show similar trends. However, values of μ_t in the downstream region are slightly elevated for syn3D.

Figure 4.18 compares the eddy viscosity through line plots. Similar to the flat plate, μ_t increases with x. Once again, syn3D displays a small peak in eddy viscosity at the leading edge ($x = 0.0$). The difference of the maximum μ_t value between syn3D and CFL3D is, as mentioned previously, greater in the downstream region. In, Figure 4.18a, it can again be

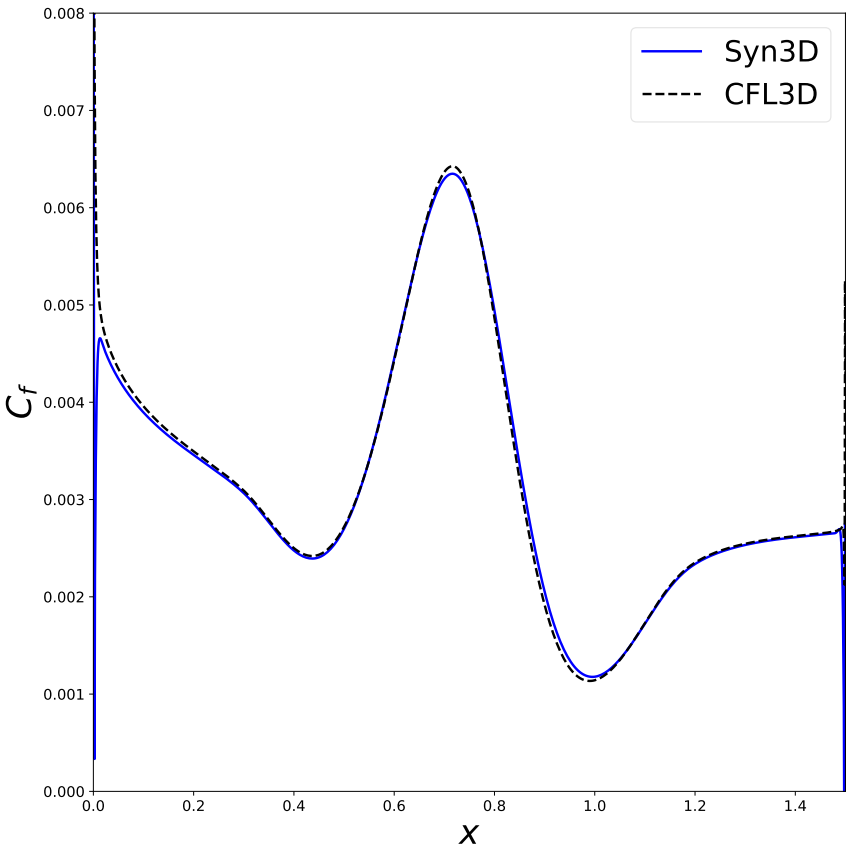


Figure 4.15 – 2D Bump (syn3D): Coefficient of skin friction distribution along the bump.

seen that the eddy viscosity increases with y until it reaches a maximum value and begins to decrease again as it reaches the far-field region. In the region of increasing μ_t ($y < 0.054$), syn3D and CFL3D show very similar results. In the decreasing region, however, syn3D displays a greater gradient and under-predicts the distance needed for μ_t to return to its free-stream value.

Figure 4.19 compares the velocity profile. A sharp velocity gradient near the wall is observed at both locations, although the height of the boundary layer is greater as x increases. Profiles are identical between the two solvers.

Figures 4.20 and 4.21 show C_f and C_p distributions along the bump for all grids. Both plots show over-dissipated results for the coarser grids, which is expected, as discussed for the flat plate case. The difference of C_f between the coarsest and finest grid is much greater

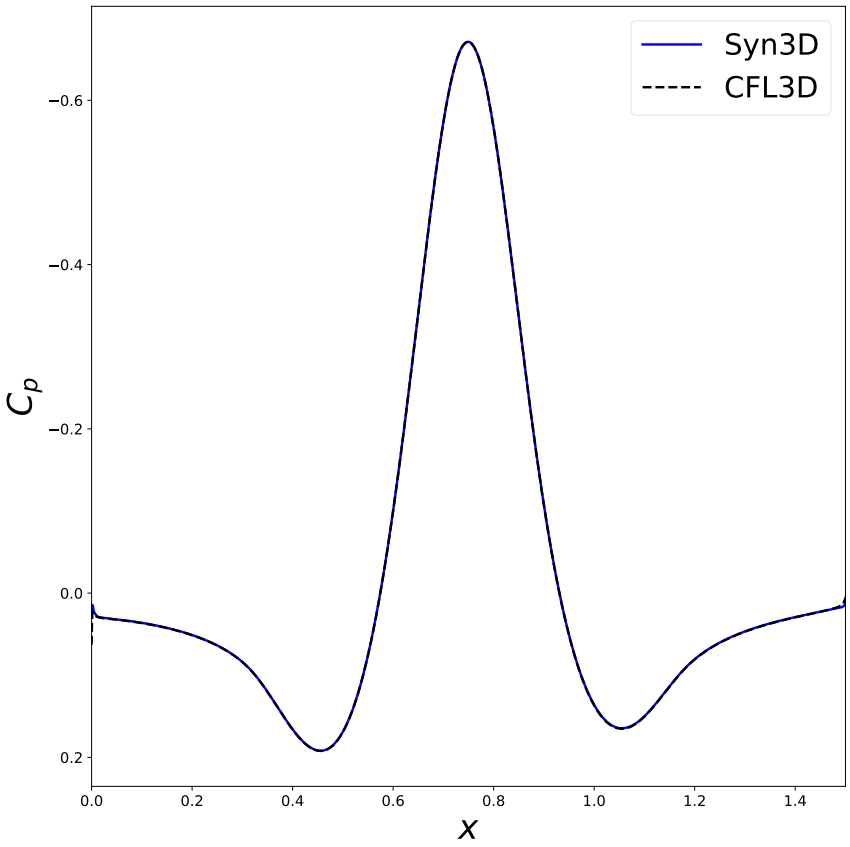
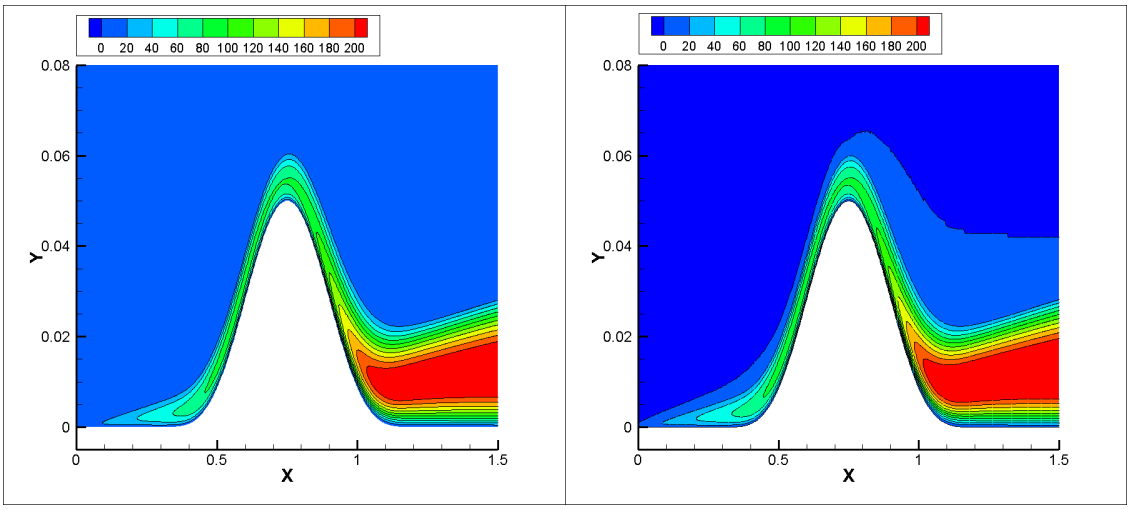
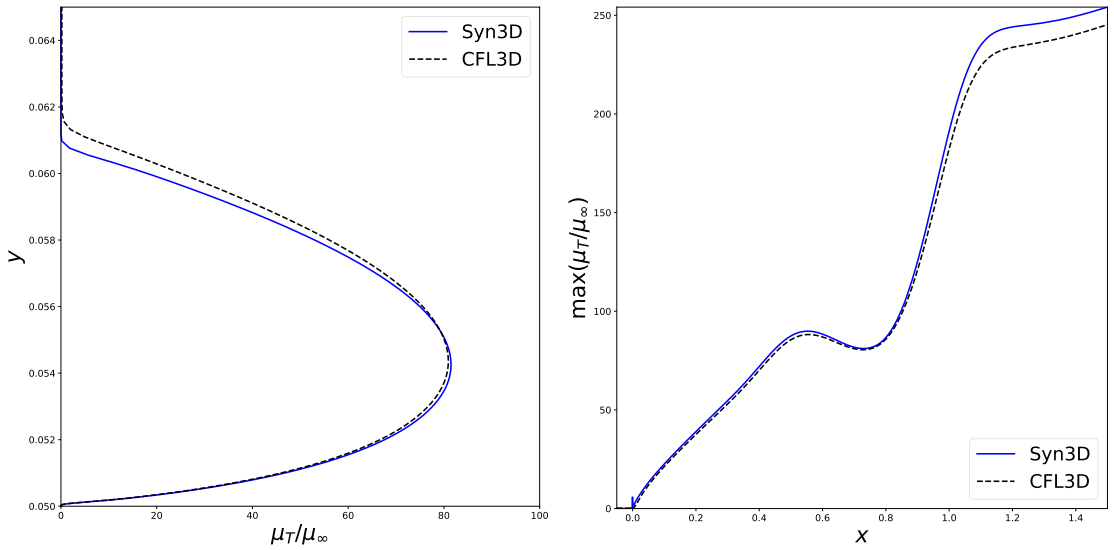


Figure 4.16 – 2D Bump (syn3D): Coefficient of pressure distribution along the bump.



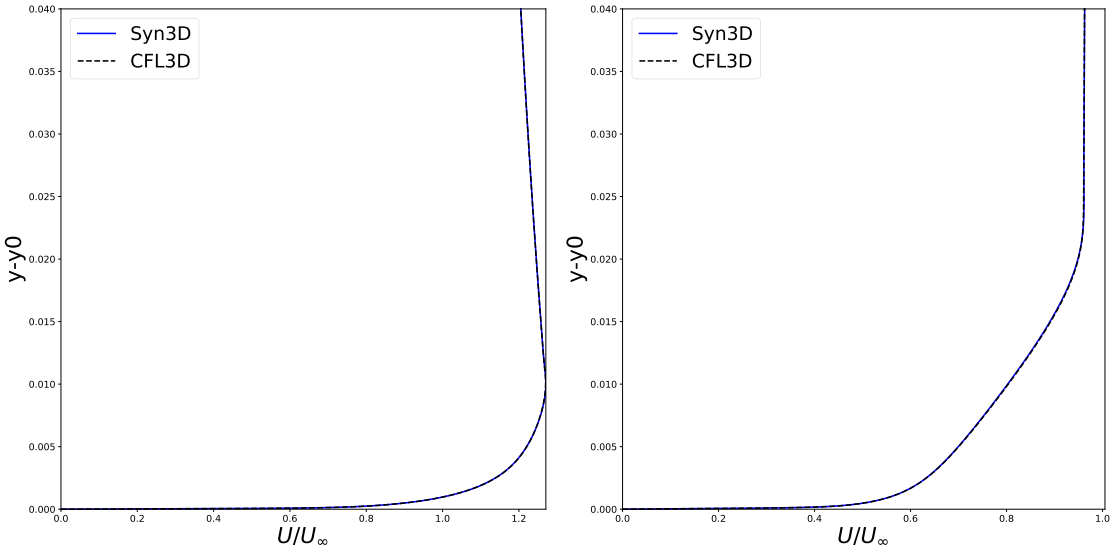
(a) CFL3D **(b)** syn3D

Figure 4.17 – 2D Bump (syn3D): Contours of μ_T/μ_∞.



(a) Profile at $x = 0.75$ **(b)** Maximum value in the boundary layer.

Figure 4.18 – 2D Bump (syn3D): Dimensionless eddy viscosity profiles.



(a) $x = 0.75$ **(b)** $x = 1.20148$

Figure 4.19 – 2D Bump (syn3D): Dimensionless velocity profiles U/U_∞.

than that of C_p. In other words, pressure seems to be less senstivive to mesh refinement than the skin friction. By the same token, while there is seemingly no difference of C_p between the second finest and finest grids, indicating that grid convergence is achieved, there remains a noticeable discrepancy for C_f, which indicates its greater sensitivity to grid refinement.

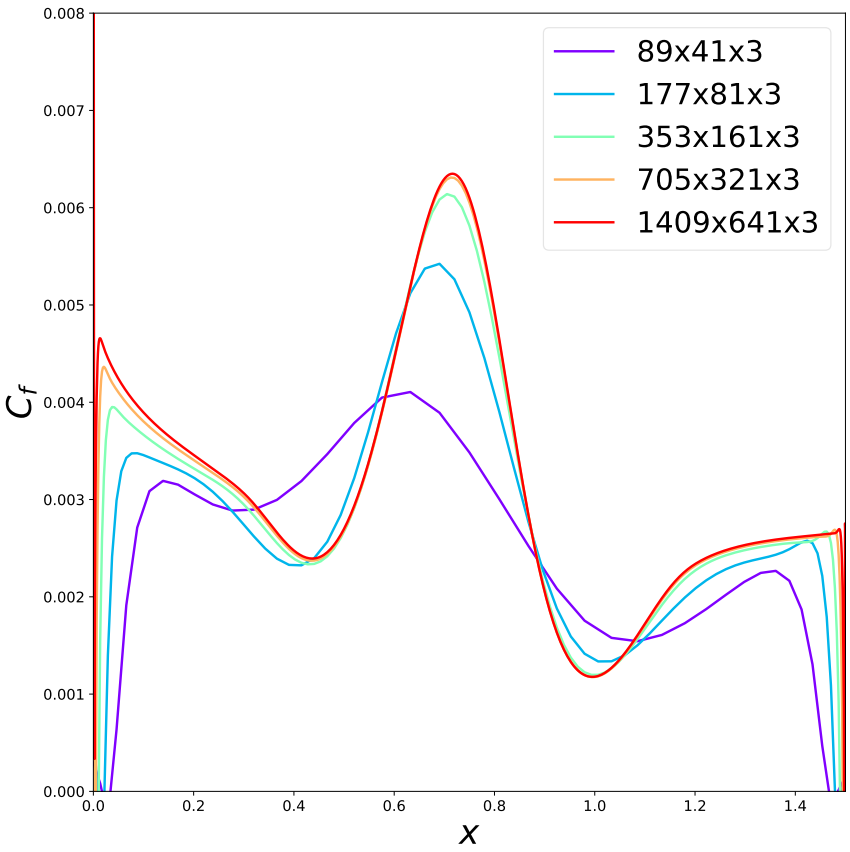


Figure 4.20 – 2D Bump (syn3D): Coefficient of skin friction along the bump for various grids.

Figure 4.22 compares the eddy viscosity distribution between grids. The peak at the leading edge is significantly more pronounced for the coarser grids, and even slightly extends upstream. This results in over-predicted values for most, but not all, of the length of the wall; μ_t is under-predicted after $x \approx 1.0$. Again, this is due to increased dissipation in the coarser grids, where the gradients are not as sharp as they should be.

Figures 4.23 and 4.24 show the convergence of force coefficients and skin friction for various grid sizes, compared with results from CFL3D and FUN3D. Again, it can be seen that there is much more dependence on grid size for syn3D as compared to the NASA solvers, although most values are very similar on the finest grid. The pressure contribution to drag, seems to have reached the asymptotic range of convergence, although the same cannot be said of the viscous contribution. Without a doubt, this increased dependence on grid size is

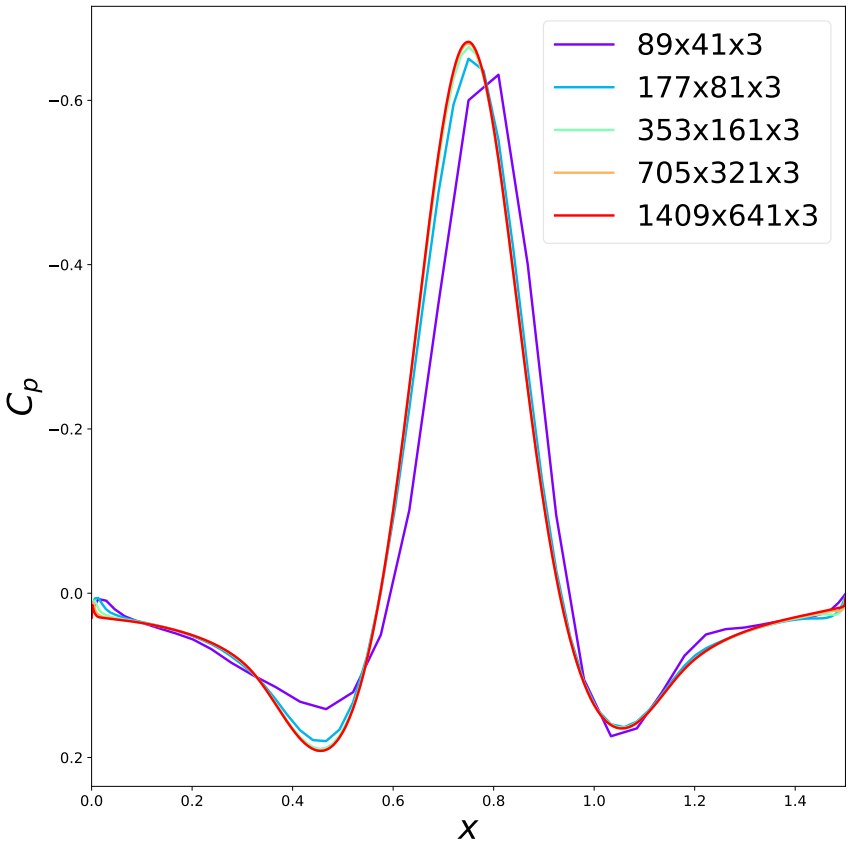
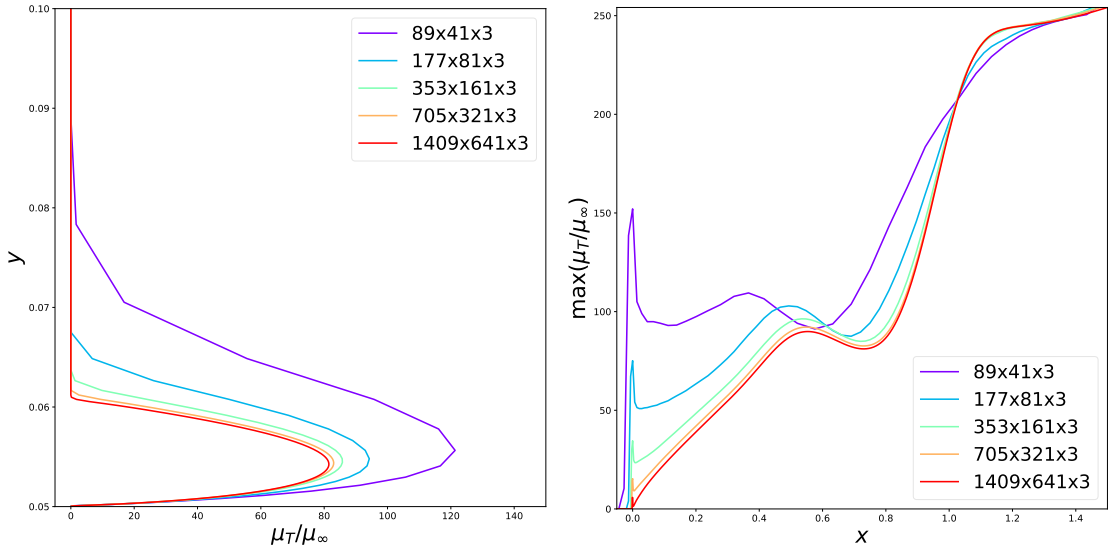


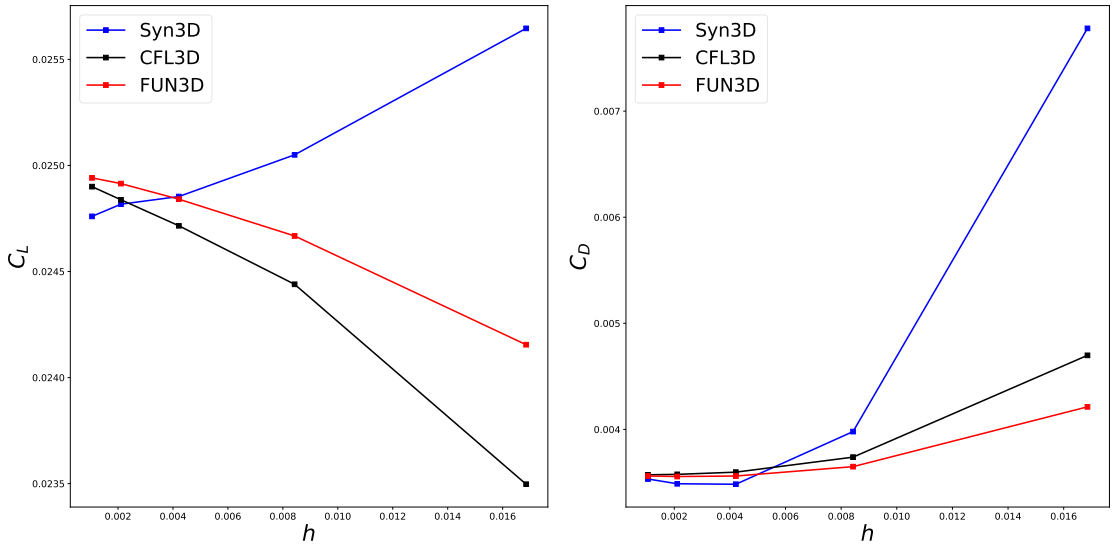
Figure 4.21 – 2D Bump (syn3D): Coefficient of pressure along the bump for various grid.

due to the peak in eddy viscosity at the leading edge, which is an interface between block boundaries.



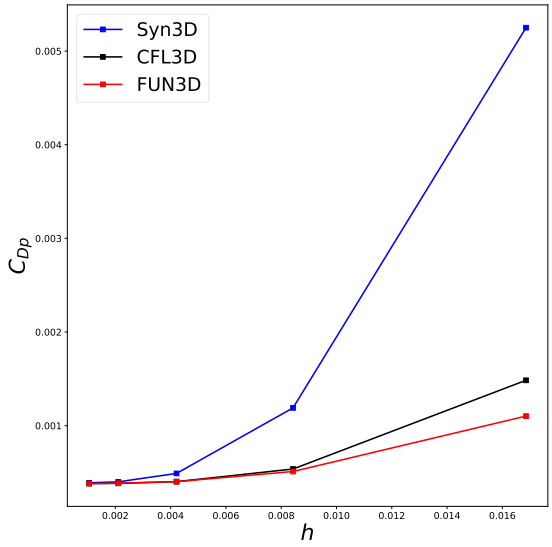
(a) Profile at $x = 0.75$. **(b)** Maximum value in the boundary layer.

Figure 4.22 – 2D Bump (syn3D): Dimensionless eddy viscosity profiles on the bump for various grids.

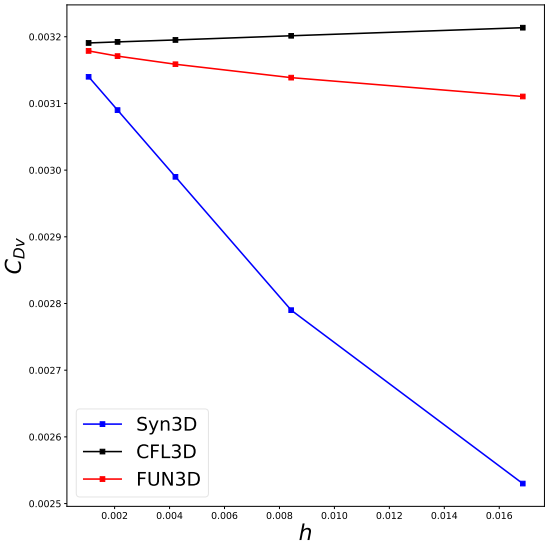


(a) Lift coefficient.

(b) Drag coefficient.

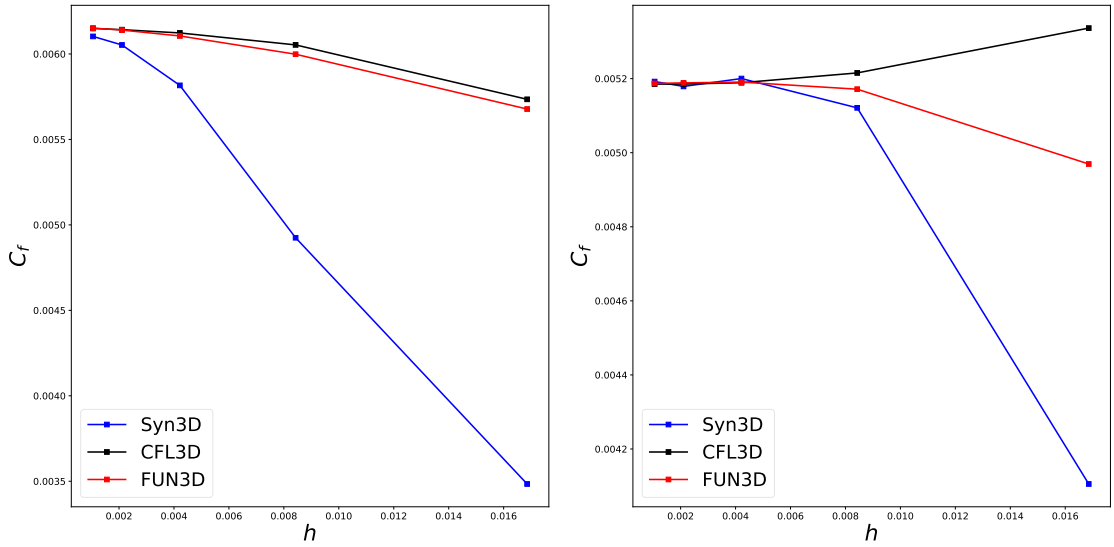


(c) Pressure contribution to drag.



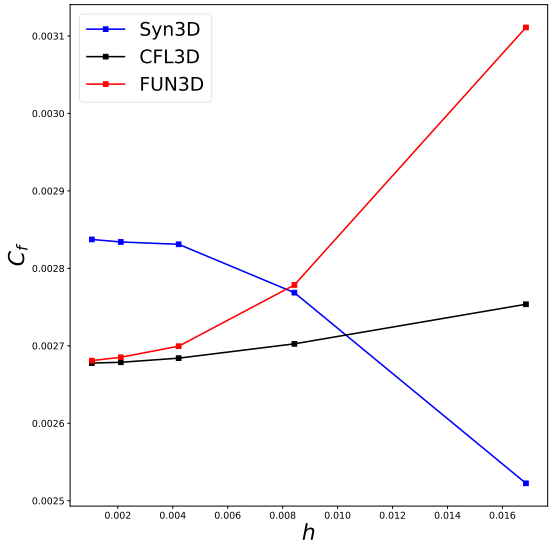
(d) Viscous contribution to drag.

Figure 4.23 – 2D Bump (syn3D): Force coefficients for various grid sizes.



(a) C_f grid study where x=0.75

(b) C_f grid study where x=0.6321975



(c) C_f grid study where x=0.8678025

Figure 4.24 – 2D Bump (syn3D): Skin friction coefficient at specific locations for various grid sizes.

4.3 Three-dimensional bump-in-channel

The third verification case from the TMR website that was investigated was the three-dimensional bump-in-channel, which is available on [18] under the name "3D Bump-in-channel". This case was only simulated with syn3D, for the same reasons as the two-dimensional bump.

The problem domain and flow conditions are shown in Figure 4.25. The Reynolds number for this case is of three million. While the website provides five sets of grids for this case as well, only results from the finest three grids are shown.

The number of elements of the finest available grid exceeds 1.5 million and the residual could not be converged past 10^{-2} after over 6 days of computation with 64 cores, which is typically insufficient for the purposes of validation – results are shown nonetheless since integrated values such as lift and drag coefficients are sufficiently converged.

Table 4.5 compares force coefficients and shows relatively good agreement. Figure 4.26 shows the convergence for the finest three grids.

Table 4.5 – 3D Bump (syn3D): Comparison of force coefficients for the three-dimensional bump.

Solver	C_L	C_D	C_{Dv}	C_{Dp}
CFL3D	0.0250	0.0036	0.0032	0.0004
syn3D	0.0258	0.0035	0.0031	0.0004

Figure 4.27 compares C_p at four fixed values of y. The relative shift between the different curves can be attributed to the grid resolution and the large overshoot in eddy viscosity in the coarser grids. Figures 4.28 and 4.29 show contour plots of the dimensionless eddy viscosity at $x = 0.3$ and $x = 1.2$ respectively. Figure 4.30 compares dimensionless eddy viscosity profiles at various locations. The plots are similar in trend to those of the two-dimensional bump, but vary due to the three-dimensional geometry of the bump. While

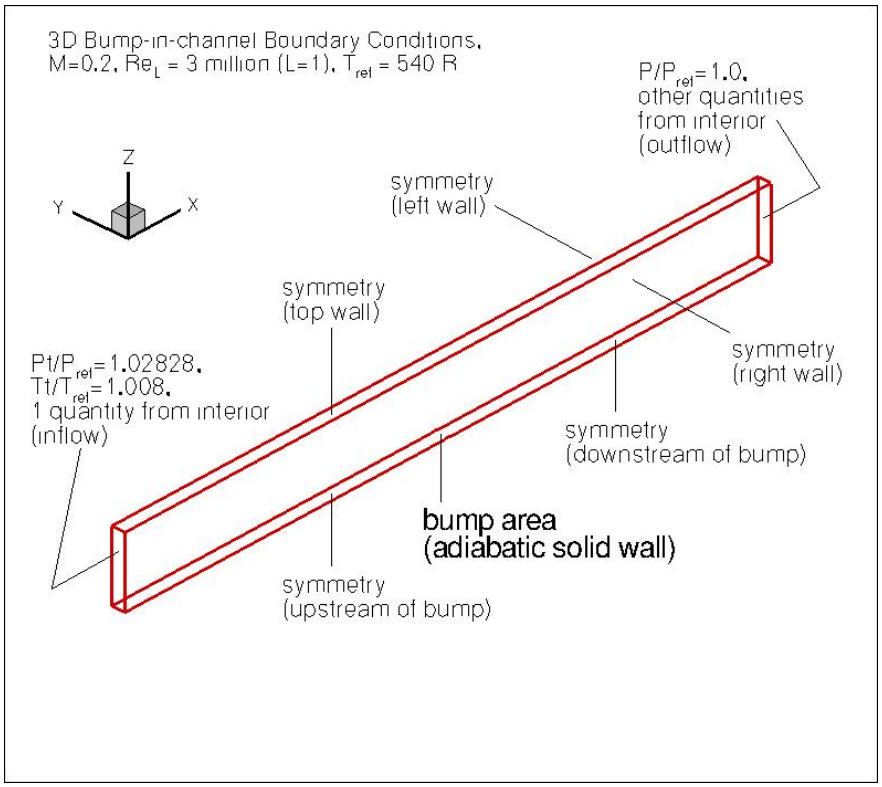
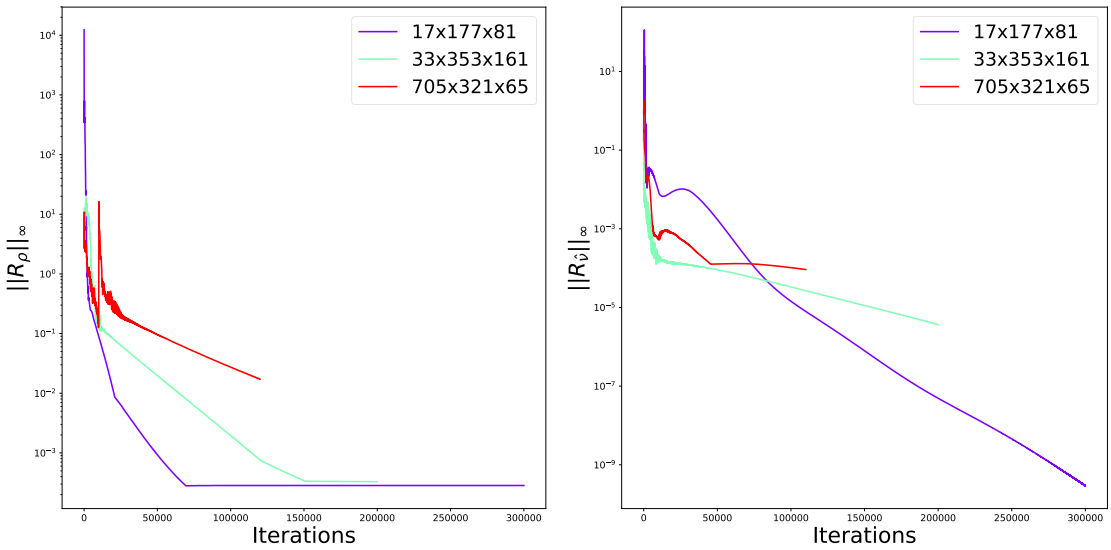


Figure 4.25 – Flow conditions and problem domain for the turbulent three-dimensional bump case [18].

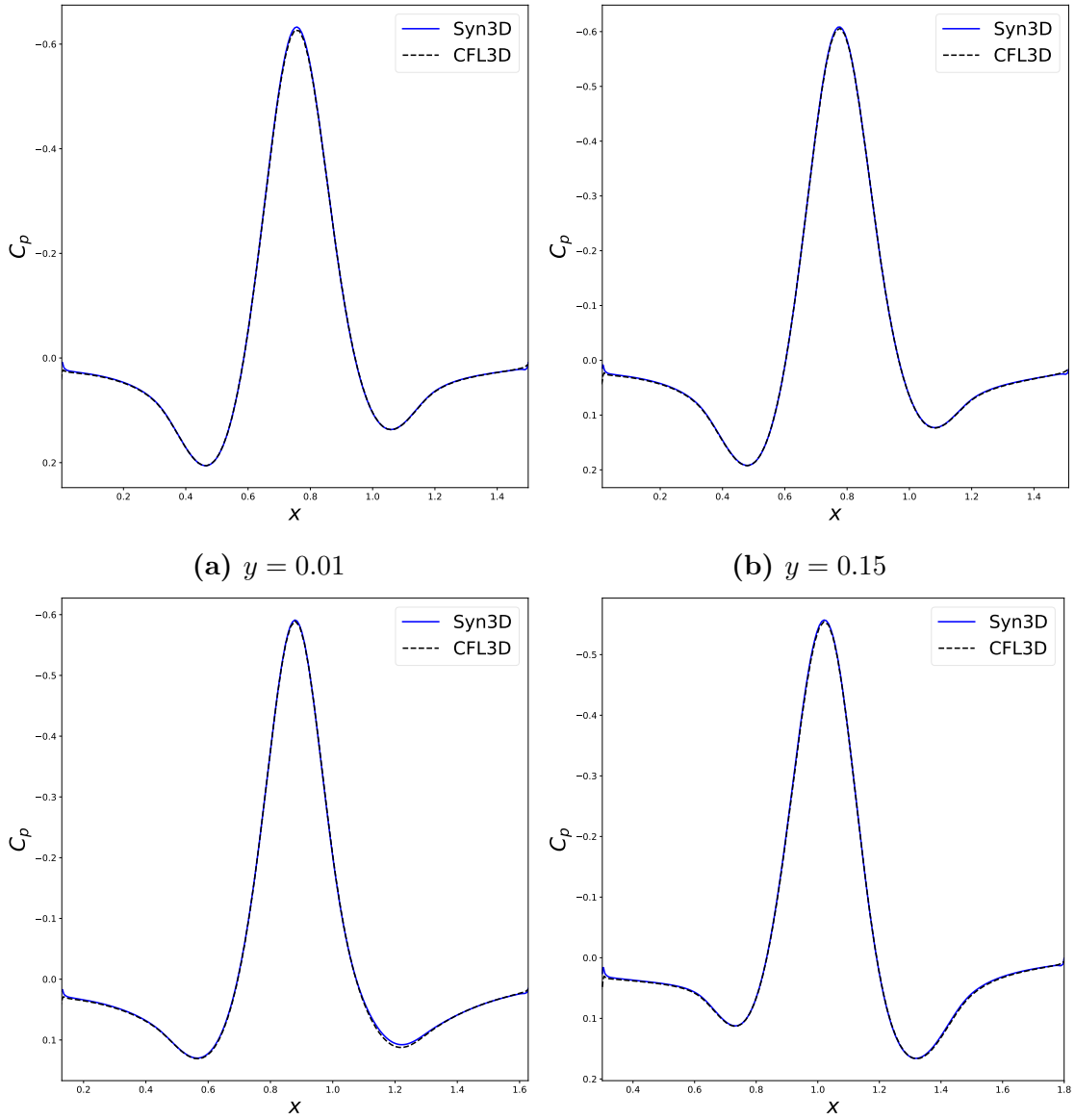


(a) Maximum density residual. **(b)** Maximum turbulent variable residual.

Figure 4.26 – 3D Bump (syn3D): Convergence of flow and turbulence variables.

the eddy viscosity is, as in previous cases, over-predicted by syn3D, Figure 4.30b displays a large discrepancy between syn3D and CFL3D.

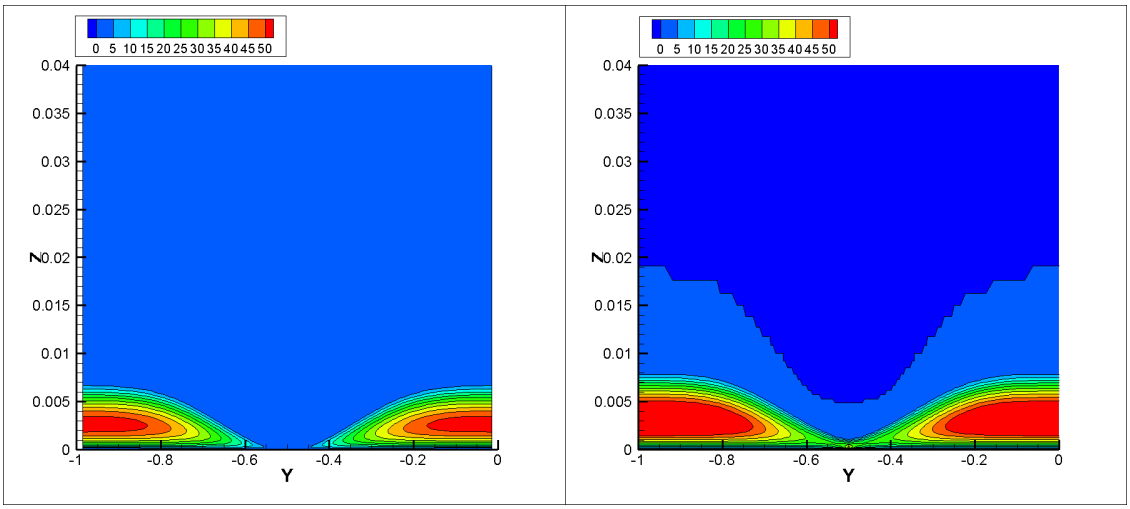
As for the grid study, Figure 4.31 compares the dependence of force coefficients on grid size, Figure 4.32 and Figure 4.33 compare the eddy viscosity and pressure coefficient at select locations. Similarly to the two-dimensional bump, the pressure contribution to drag is less sensitive to mesh refinement than the viscous contribution.



(a) $y = 0.01$ **(b)** $y = 0.15$

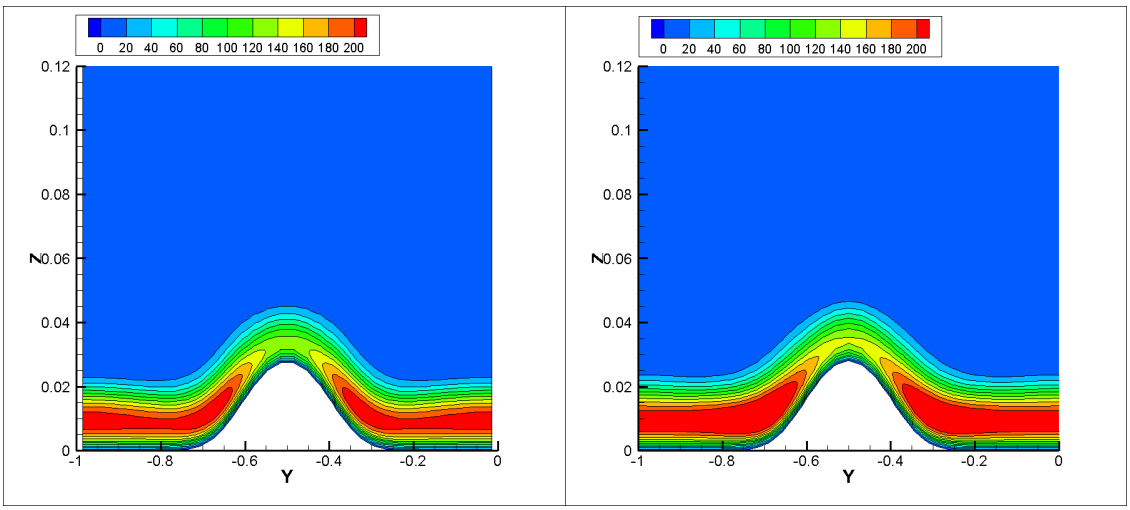
(c) $y = 0.3$ **(d)** $y = 0.50$

Figure 4.27 – 3D Bump (syn3D): Comparison of coefficient of pressure distribution at various cross-sections.



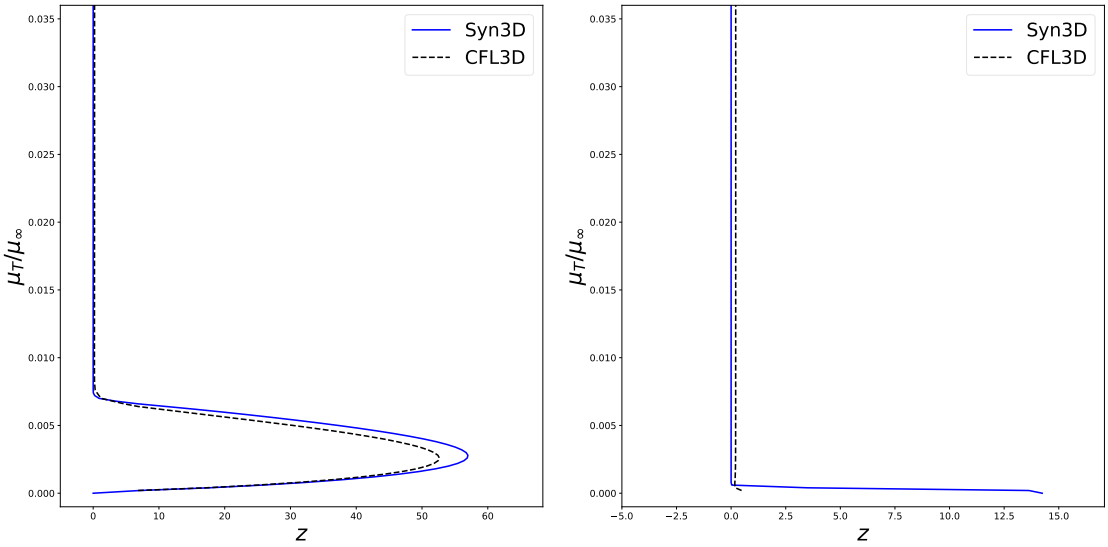
(a) CFL3D **(b)** syn3D

Figure 4.28 – 3D Bump (syn3D): Contours of $\frac{\mu_t}{\mu_\infty}$ at $x = 0.3$

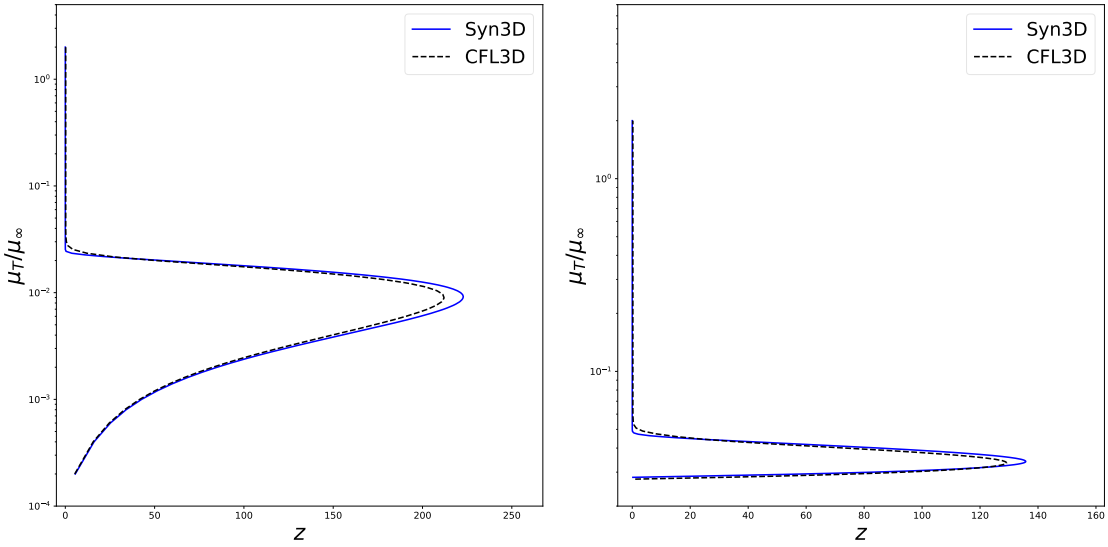


(a) CFL3D **(b)** syn3D

Figure 4.29 – 3D Bump (syn3D): Contours of $\frac{\mu_t}{\mu_\infty}$ at $x = 1.2$

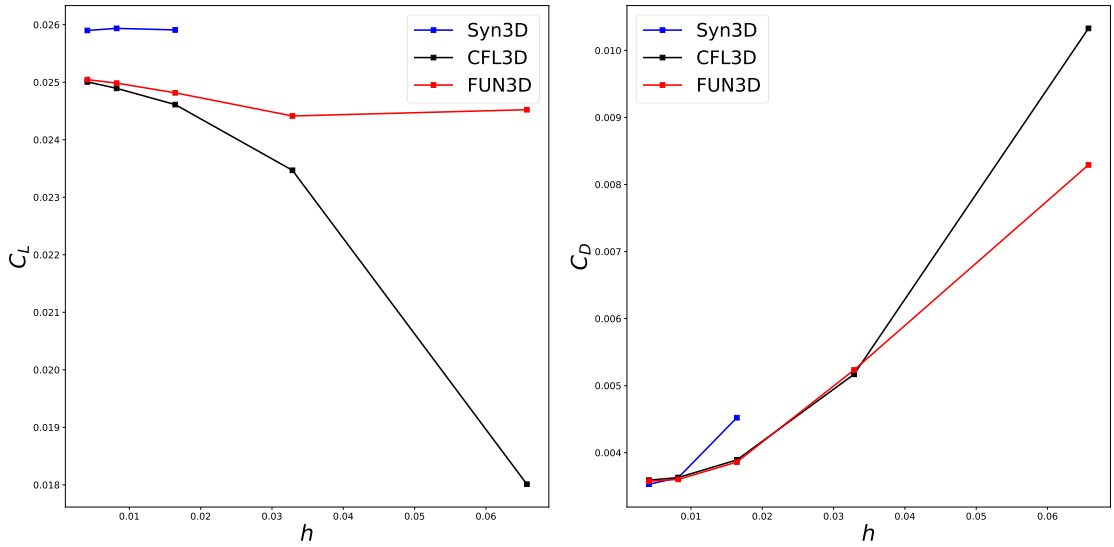


(a) $x = 0.3, y = -0.1$ **(b)** $x = 0.3, y = -0.5$



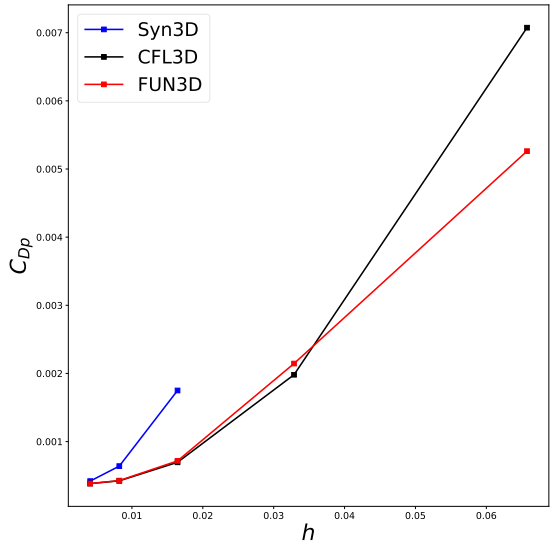
(c) $x = 1.2, y = -0.1$ **(d)** $x = 1.2, y = -0.5$

Figure 4.30 – 3D Bump (syn3D): Comparison of dimensionless eddy viscosity profiles at various locations.

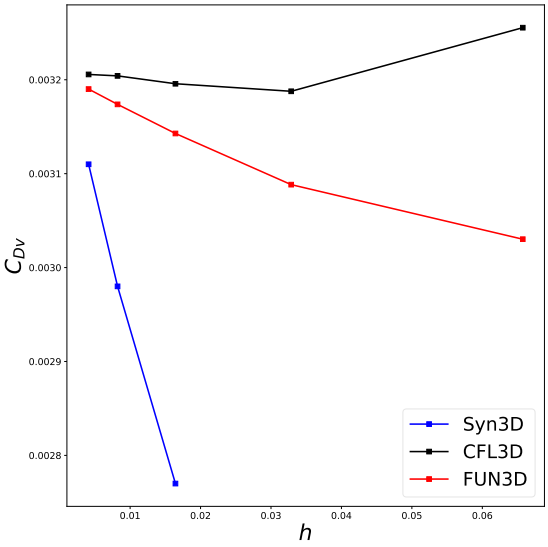


(a) C_L

(b) C_D



(c) C_{Dp}



(d) C_{Dv}

Figure 4.31 – 3D Bump (syn3D): Force coefficients for various grids

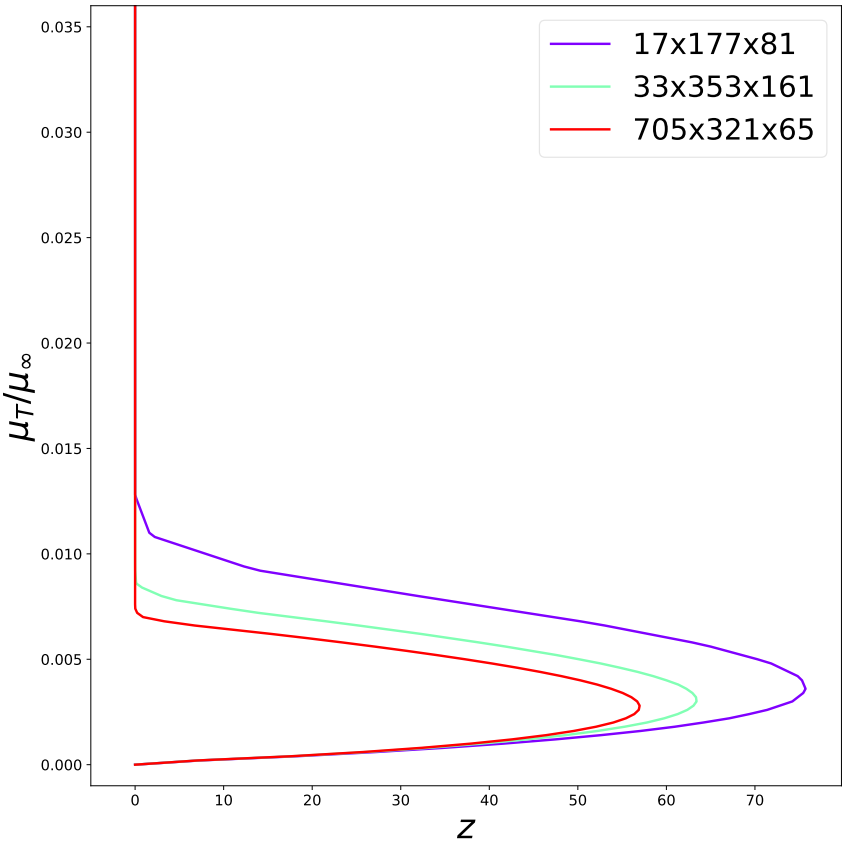


Figure 4.32 – 3D Bump (syn3D): Dimensionless eddy viscosity at $x = 0.3, y = 0.1$ for various grids.

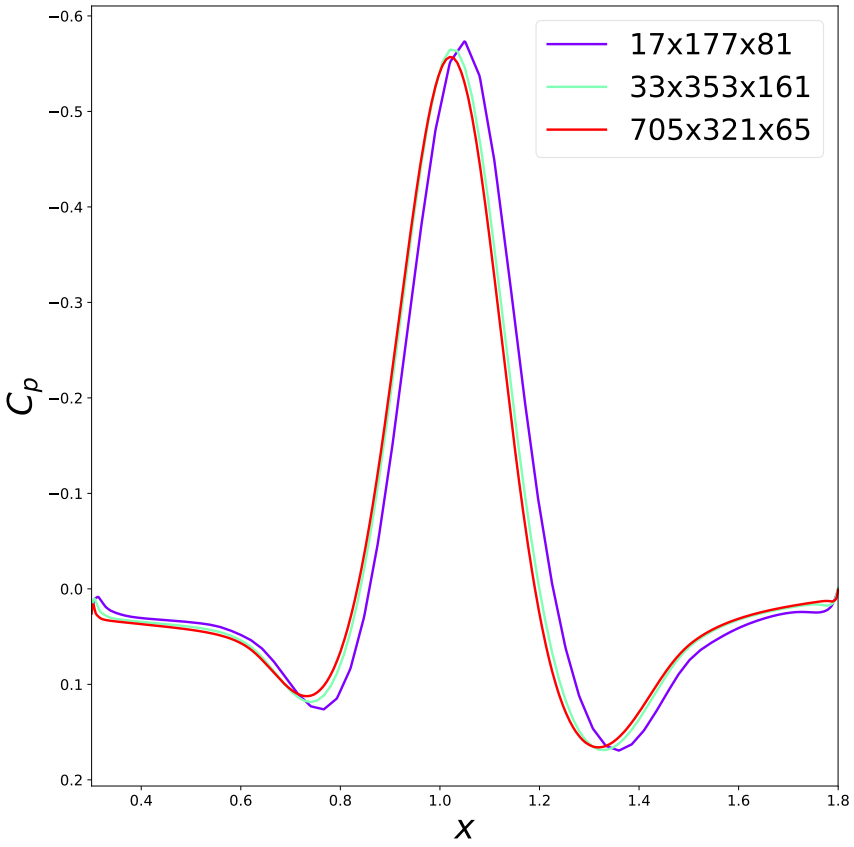


Figure 4.33 – 3D Bump (syn3D): Coefficient of pressure at $y = 0.5$ for various grids.

4.4 NACA0012

The last validation case from the TMR website is the NACA 0012 airfoil case, which was only run with NX Flow due to time constraints. This case is available on [18] under the name "2D NACA 0012 airfoil". While the Turbulence Modeling Resource provides grids for this problem, they are not suitable for the boundary conditions available in NX Flow. The mesh used is shown in Figure 4.34, which is an unstructured mesh. The boundary layer is meshed with hexahedron elements and the far field with tetrahedral elements. The average value of y^+ for the first layer is approximately unity. No grid study was performed since no grid study data is available at [18]. This case was run using a fixed density, as it is an incompressible case. The Reynolds number value is 6 million and three angles of attack were used: 0, 10 and 15 degrees.

Table 4.6 tabulates the lift and drag coefficients at all angles of attack and Figure 4.35 through Figure 4.37 compare the pressure and skin friction coefficient along the airfoil at $\alpha = 0°$, $\alpha = 10°$ and $\alpha = 15°$ respectively. In all cases, there is a high level of suction on the upper surface at the leading edge, as the flow accelerates. The pressure differential between the upper and lower surface is, for the most part, what generates lift. The skin friction distribution is, as expected, qualitatively similar to that of the flat plate: the peak is at the

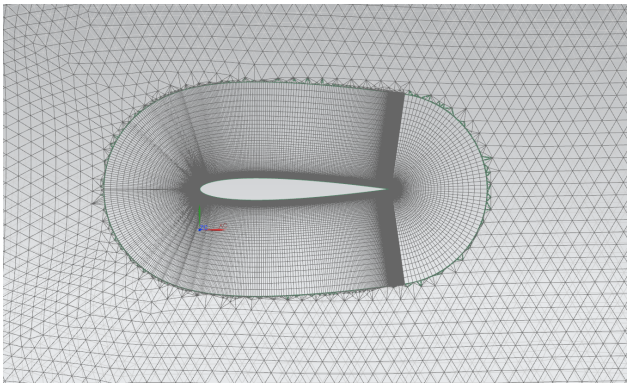


Figure 4.34 – Close-up of the NACA0012 unstructured mesh. Far field extends 3 chords upstream, 7 chords downstream and 3 chords above and below.

leading edge, after which point C_f decreases for the remainder of the wall.

Even though there is decent agreement between the coefficients of pressure and skin friction, the drag values are heavily over-predicted by NX Flow. The pressure at the leading edge is under-predicted by NX Flow. Compounded with the fact that most of the drag (over 90%) is due to the pressure forces in this case, this results in the large discrepancy observed.

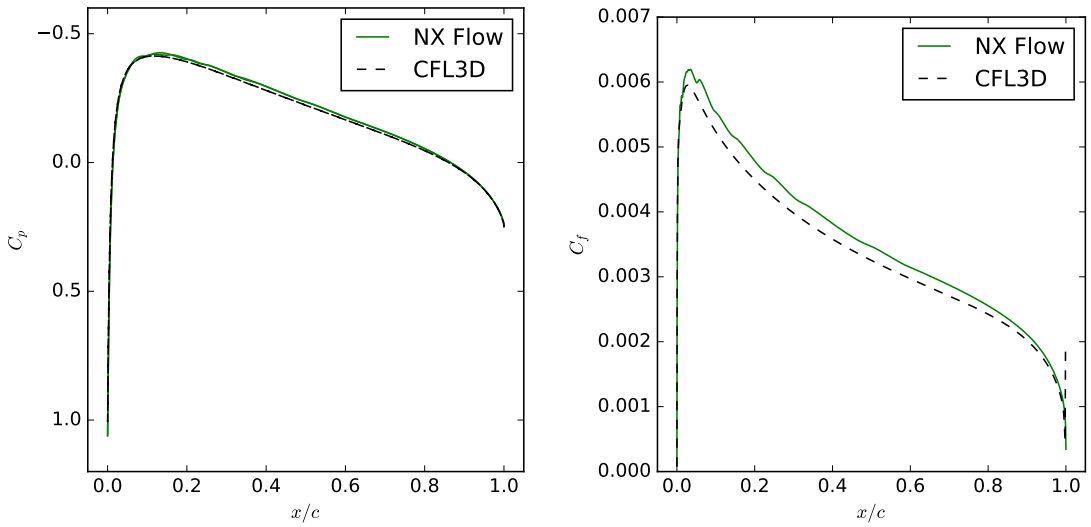
The disagreements may be attributed to many sources, but the chief source of error is that NX Flow is currently meant to be used for internal flows. Consequently, the product lacks far-field boundary conditions, which have been shown to significantly impact the solution, especially when the distance between the far-field boundary and the airfoil surface is as small as it is in this case [36], [37]. Moreover, the y^+ is greater at the leading edge of the upper surface, and reaches values as high as 3.4. For comparison, the highest y^+ value for the coarsest flat plate grid (35×25) is below 0.95. It may be that a lower y^+ value is necessary to capture the required acceleration, especially given that weak boundary conditions are used for solid walls, in which case the physical requirement $\mathbf{u} = 0$ at the wall may not be satisfied on coarser grids.

Thus, both far-field boundary conditions and solid wall boundary conditions must be revisited in order to allow NX Flow to be confidently used for external flows. The far-field BC should preferably employ a circulation correction, as this drastically reduces the dependence of the solution on the size of the computational domain [36].

Finally, as for the flat plate problem, compressibility does have a small influence on the results, although this reason alone is not sufficient to explain the discrepancy in this case.

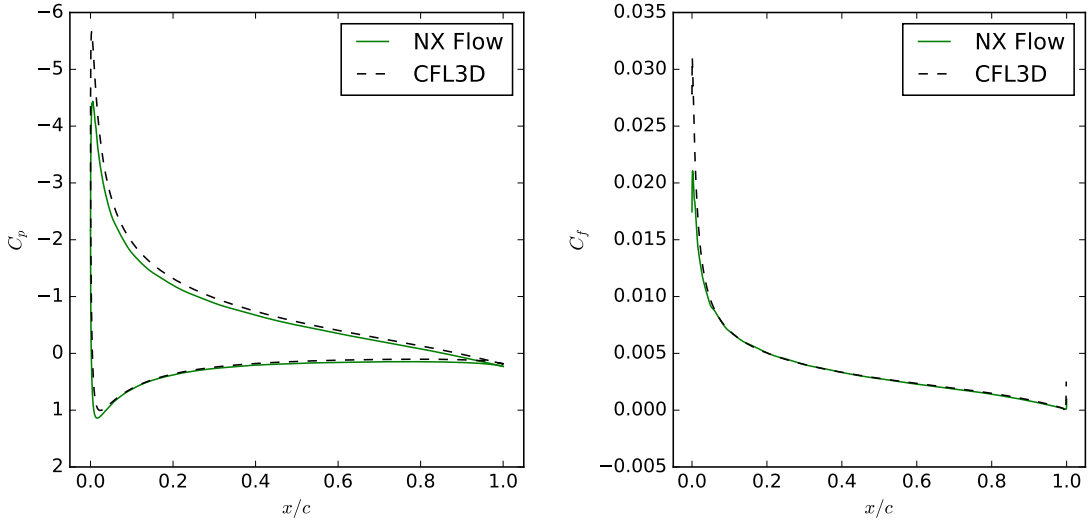
Table 4.6 – NACA0012: Comparison of drag and lift coefficients at all angles of attack.

	$\alpha = 0°$		$\alpha = 10°$		$\alpha = 15°$	
Code	C_L	C_D	C_L	C_D	C_L	C_D
CFL3D	≈ 0	0.00819	1.0909	0.01231	1.5461	0.02124
FUN3D	≈ 0	0.00812	1.0983	0.01242	1.5547	0.02159
NX Flow	7.5E-5	0.01230	1.0175	0.05018	1.4295	0.09959



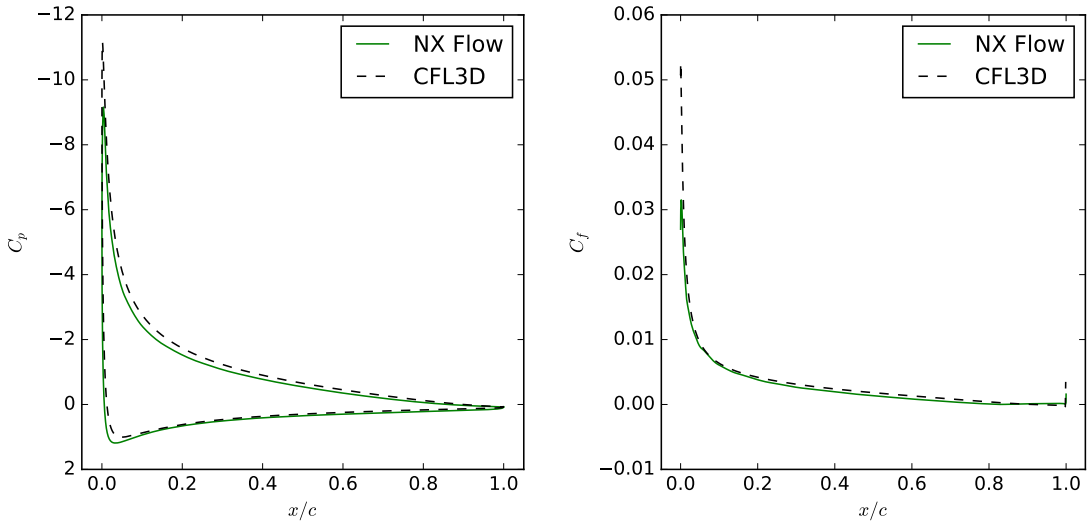
(a) Coefficient of pressure. **(b)** Coefficient of skin friction.

Figure 4.35 – NACA0012 (NX Flow): Force coefficient distribution at $\alpha = 0°$.



(a) Coefficient of pressure. **(b)** Coefficient of skin friction.

Figure 4.36 – NACA0012 (NX Flow): Force coefficient distribution at $\alpha = 10°$.



(a) Coefficient of pressure. **(b)** Coefficient of skin friction.

Figure 4.37 – NACA0012 (NX Flow): Force coefficient distribution at $\alpha = 15°$.

4.5 RAE 2822

Flow over the RAE 2822 airfoil is discussed in this section. Experimental data is obtained from the AGARD advisory report 138 [38]. The report provides results for a large set of conditions, however only two of these are run in this report: cases 3 and 9. The free-stream conditions and the solvers ran are tabulated in Table 4.7. Figures 4.38 and 4.39 show the mesh used in syn3D and NX Flow respectively. The average y^+ of the first layer for both grids is around 1.

4.5.1 Case 3

Figure 4.40 compares the pressure coefficient over the airfoil between NX Flow, syn3D and the experimental data. Figure 4.41 compares the skin friction coefficient only for syn3D and NX Flow, since no experimental data is available. It can be seen that syn3D over-predicts the suction on the upper surface at the leading edge, whereas NX Flow under-predicts it. The skin friction coefficient distribution is different between syn3D and NX Flow, but similar trends can be observed.

Table 4.8 tabulates the lift and drag coefficients. Syn3D slightly over-predicts both the lift and drag. It was shown for the TMR cases that syn3D is quite sensitive to grid density. Since the maximum y^+ is around 3, compared to 0.95 for the the coarsest flat plate grid, it is possible that the grid is not sufficiently refined.

On the other hand, NX Flow under-predicts the lift and significantly over-predicts drag, as it did for the NACA0012 case. Since a mesh very similar to that of the NACA0012

Table 4.7 – Case specifications for the RAE 2822 and which code they were run with.

Case	M_∞	$Re \times 10^6$	α (°)	syn3D	NX Flow
3	0.6	6.3	2.57	Yes	Yes
9	0.730	6.5	3.19	Yes	No

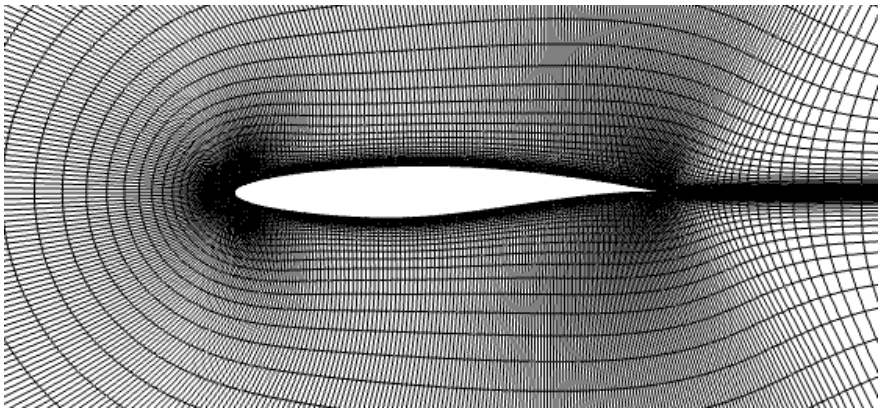


Figure 4.38 – Close-up of the syn3D mesh for the RAE 2822 case.

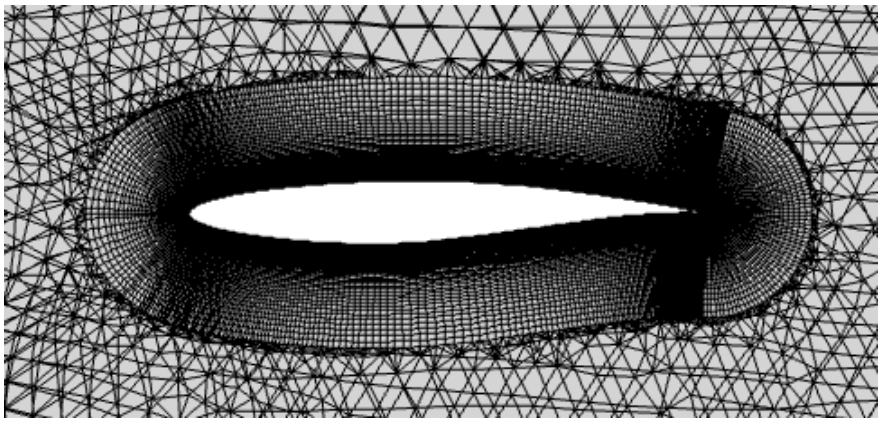


Figure 4.39 – Close-up of the NX Flow mesh for the RAE 2822 case.

Table 4.8 – RAE 2822 case 3 (syn3D & NX Flow): Comparison of drag and lift coefficients with experimental data.

Source	C_L	C_D
Experimental	0.5220	0.0101
syn3D	0.5907	0.0115
NX Flow	0.4865	0.0348

Table 4.9 – RAE 2822 case 9 (syn3D): Comparison of drag and lift coefficients with experimental data.

Source	C_L	C_D
Experimental	0.8030	0.0168
syn3D	0.8186	0.0225

is used for the RAE, the errors may be attributed to the wall boundary condition, the far-field BC and the size of the grid. Additionally, NX Flow has had limited testing and benchmarking on compressible problems, and it could be that the implementation requires improvement. Finally, the code was run with a first-order advection scheme due to its robustness; a second-order scheme would be more accurate, albeit more difficult to converge as well as more computationally expensive.

4.5.2 Case 9

Figure 4.42 compares the pressure coefficient over the airfoil between syn3D and experimental data. A shock can be observed on the upper surface, at around half of the chord length. This shock is well captured by syn3D and the location approximately matches that of the experimental data. Figure 4.43 shows the skin friction coefficient on the upper surface. A shock can be observed in this plot as well, at around half the chord length.

Table 4.9 tabulates the lift and drag coefficients. Once again, syn3D slightly over-predicts both the lift and drag.

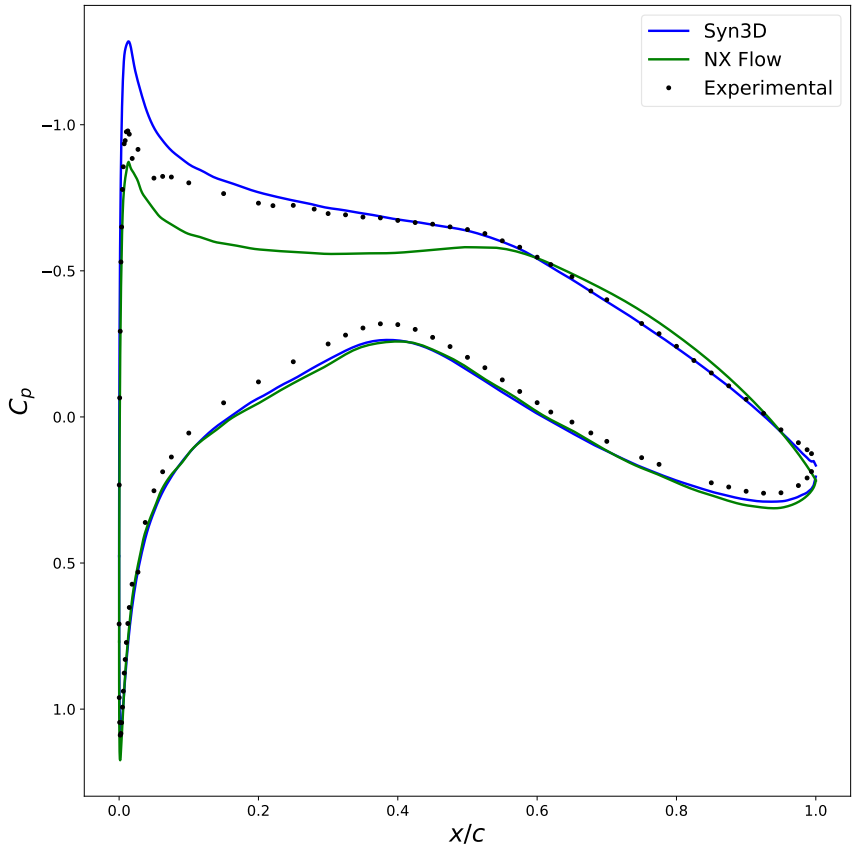


Figure 4.40 – RAE 2822 case 3 (syn3D & NX Flow): Coefficient of pressure along the airfoil.

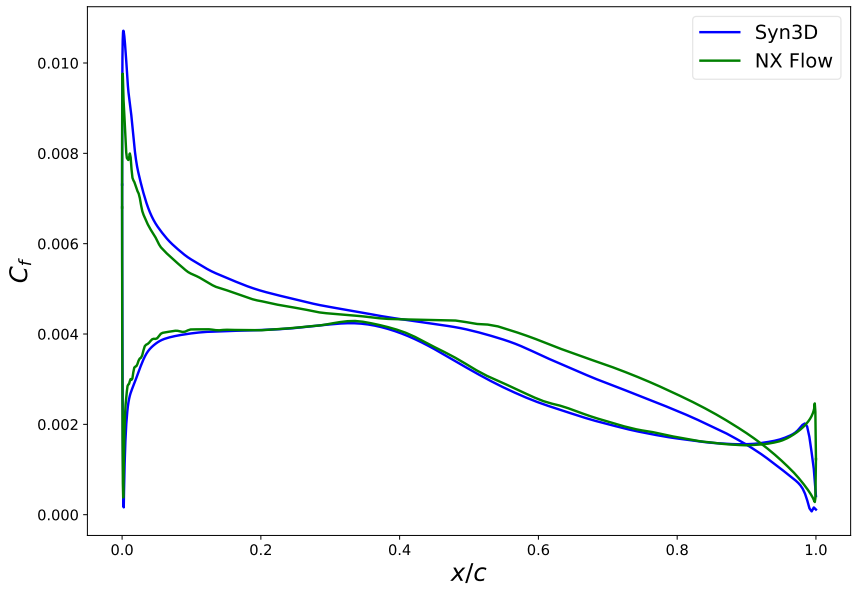


Figure 4.41 – RAE 2822 case 3 (syn3D & NX Flow) : Coefficient of skin friction along the airfoil.

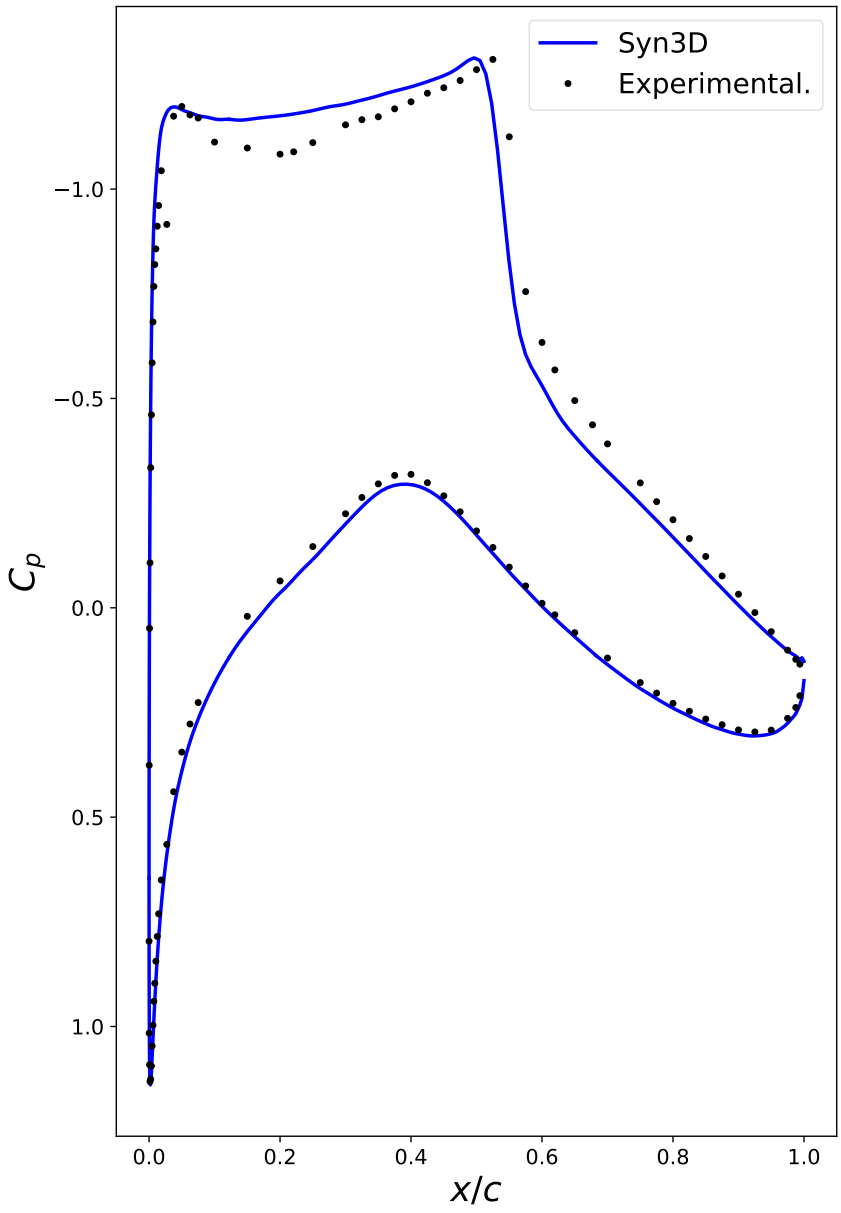


Figure 4.42 – RAE 2822 case 9 (syn3D) : Coefficient of pressure along the airfoil.

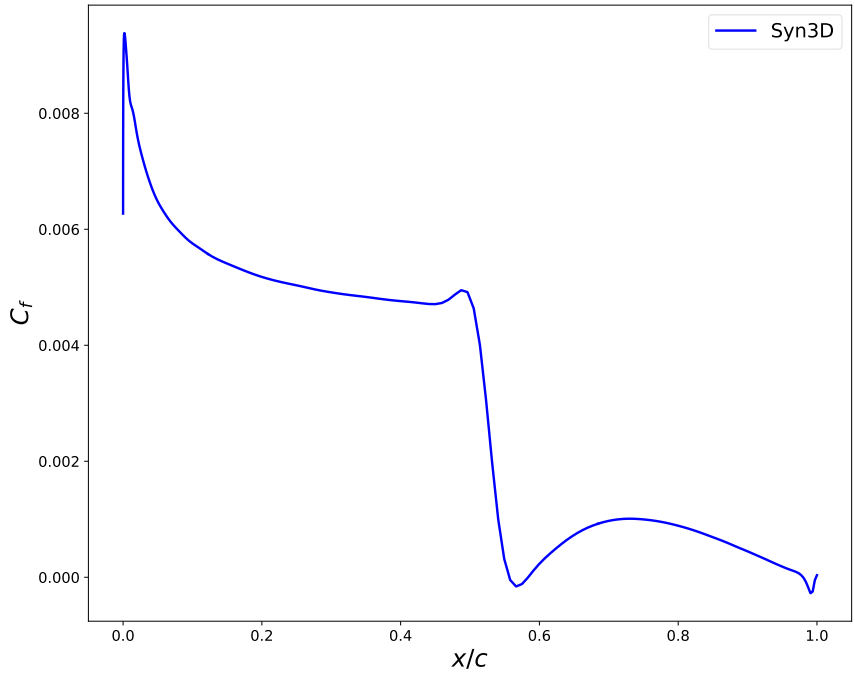


Figure 4.43 – RAE 2822 case 9 (syn3D) : Coefficient of skin friction along the upper surface of the airfoil.

4.6 High-lift configuration: Preliminary Results

The last case that was run is the McDonnel-Douglas 30P/30N three-element high-lift configuration. This case was only run with syn3D and the mesh is shown in Figure 4.44. Table 4.10 tabulates the free-stream conditions.

Unfortunately, this case could only be run on a mesh with an average y^+ of 50 and composed of around 100,000 elements. This is much too coarse of a mesh, especially for such a complex configuration. The results are preliminary but provide the right trend.

Figure 4.45 shows a contour of the wall distance. Figure 4.46 depicts streamlines around the airfoils, which shows the expected recirculation regions.

Table 4.10 – Specification of free-stream flow conditions for the high-lift configuration

M_∞	$Re \times 10^6$	α (°)
0.2	9	19

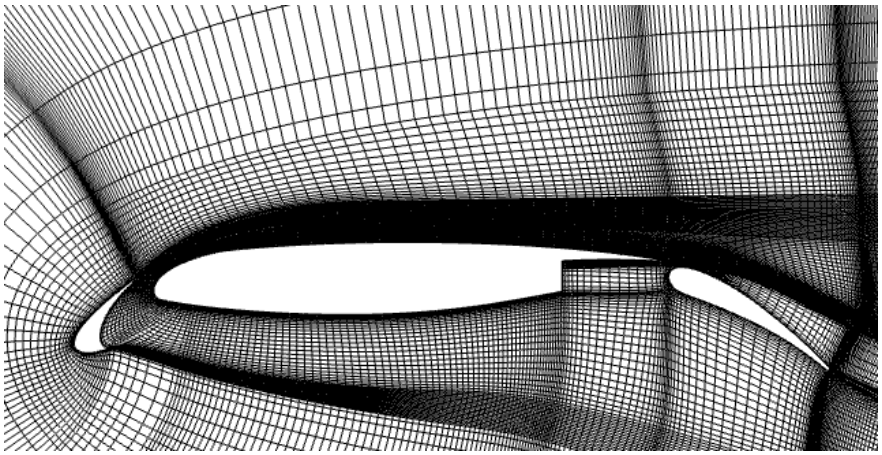


Figure 4.44 – High-lift (syn3D): Close-up of the high-lift mesh.

Table 4.11 – High-lift (syn3D): Comparison of lift and drag coefficients

Code	C_L	$_D$
syn3D	2.98	0.290
ISAAC	≈ 4	≈ 0.08

Figure 4.47 compares the coefficient of pressure distribution with ISAAC [39]. The cited reference also shows experimental data, although the results from ISAAC are virtually identical. It can be seen that the suction on the first element is not as pronounced as it should be for syn3D, which can be attributed to the coarseness of the grid. ISAAC also implements a transitional model, which allows the flow to naturally transition between the laminar regime and turbulent regime. Such a model is currently not implemented in syn3D, and this is another source of error [39].

Finally, Table 4.11 compares the lift and drag coefficients. The lift coefficient is under-predicted and the drag coefficient is over-predicted. This discrepancy may be attributed to the reasons mentioned above.

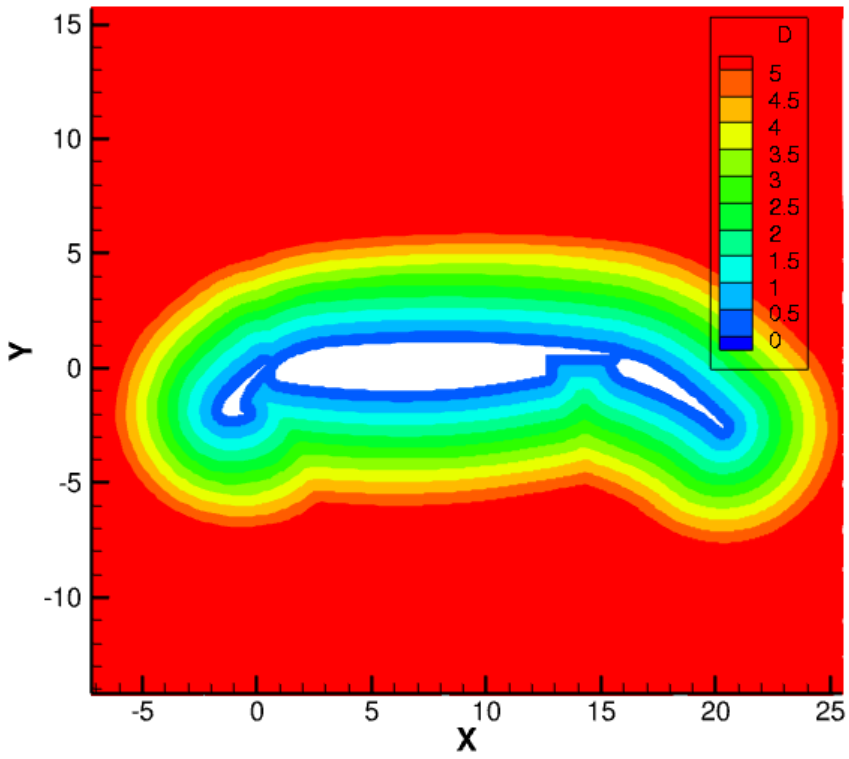


Figure 4.45 – High-lift (syn3D): Wall distance contour.

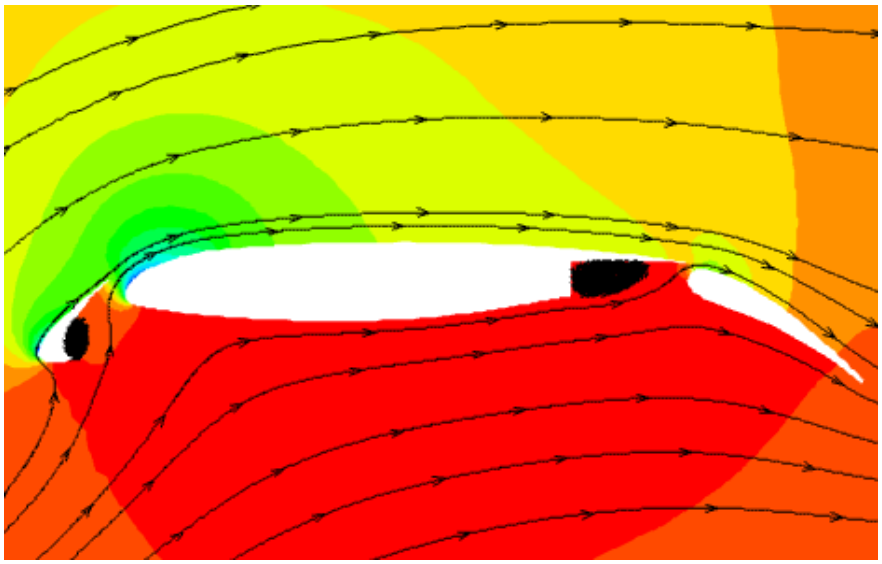


Figure 4.46 – High-lift (syn3D): Streamlines around the high-lift configuration. Recirculation regions are identified by the solid black shapes.

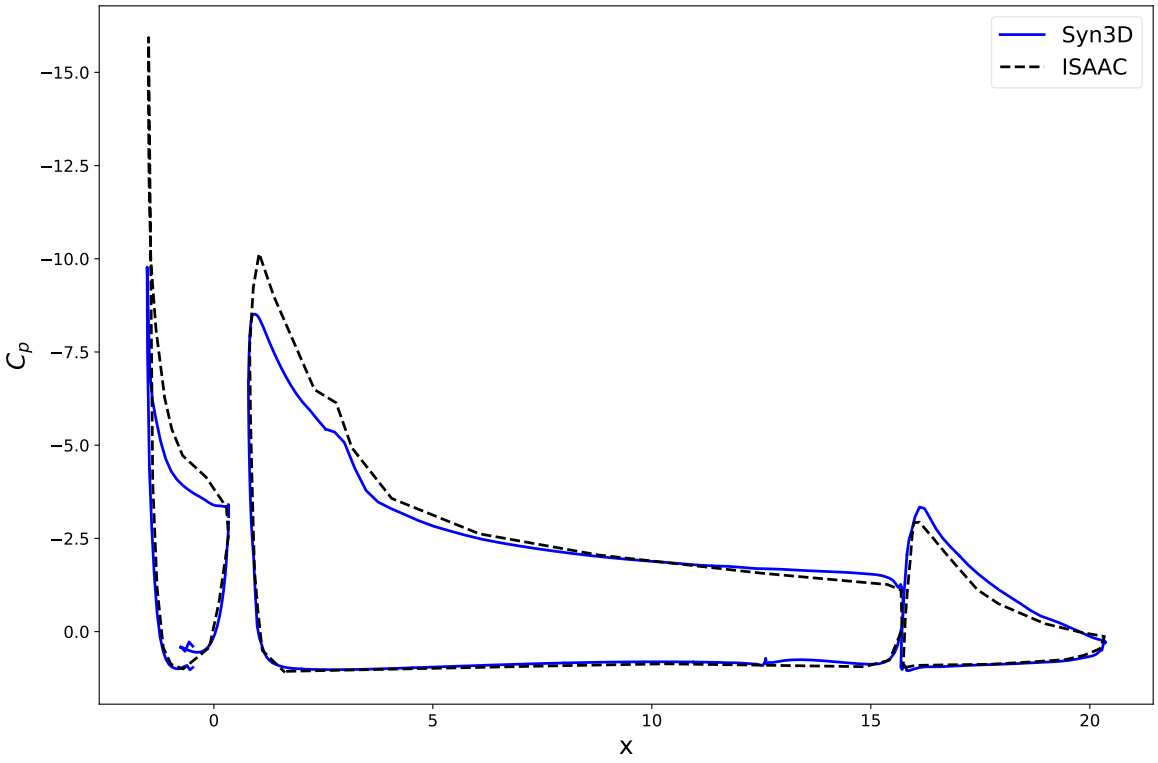


Figure 4.47 – High-lift (syn3D): Comparison of coefficient of pressure distribution.

Chapter 5

Conclusion

This chapter concludes the work. The obtained results are discussed in Section 5.1, future work is given in Section 5.2 and conclusions of the thesis and main contributions are given in Section 5.3 .

5.1 Discussion of results

The results following the implementation of the Spalart-Allmaras turbulence model in NX Flow and syn3D were compared to established CFD codes as well as experimental data. Both codes were run on compressible and essentially incompressible cases. All cases require Reynolds numbers between 2 million and 9 million. The results can only be as accurate as the rest of the existing solver, specifically the implementation and discretization of Navier-Stokes equations as well as its boundary conditions, since the eddy viscosity obtained from the S-A model both feeds into these equations and also relies on the solution of the flow variables.

Both syn3D and NX Flow gave excellent results for the flat plate problem. However, it was found that syn3D predicts a spike in the eddy viscosity at the interface between

symmetry BCs and wall BCs. This effect is more pronounced for coarser grids, which may explain why syn3D exhibits greater sensitivy to grid density as compared to other codes, including NX Flow. Another factor which contributes to this sensitivity is that the artificial dissipation parameters were not tuned for each individual mesh level. Additionally, it was seen that there exists small differences in the drag value between compressible codes and codes using a fixed density, even if that problem is run at essentially incompressible conditions ($M = 0.2$). Finally, it was seen that the temporal discretization significantly affect the residual convergence: whereas the fully-implicit backward Euler method used by NX Flow resulted in expensive but effective iterations, the explicit modified Runge-Kutta scheme employed by syn3D required around a thousand times more iterations, albeit the cost of each iteration is greatly reduced. However, ease of implementation and memory requirements are important factors to consider, both of which favor the explicit approach employed by syn3D.

Another recurring discrepancy between syn3D and the other solvers was the skin friction at the leading and trailing edge of the wall. It is important to remember that the edges are always at an interface between blocks. Since the skin friction is a measure of the velocity gradient normal to the wall, this means that the velocity gradient does not spike fast enough at the interface. It is possible that special treatment of such interfaces would remedy this problem.

Syn3D was also validated on the two-dimensional and three-dimensional bump cases from the Turbulence Modeling Resource. The spike in eddy viscosity, similar to that observed for the flat plate, was made even more apparent on these problems. It was also found that the coefficient of pressure is less sensitive to grid spacing than the skin friction coefficient. Moreover, convergence of the residual was shown to be slow on these cases. Increasing the number of multigrid levels might accelerate convergence, but optimization of convergence acceleration methods is not the focus of this work.

NX Flow was then validated on an essentially incompressible NACA0012 case at

various angles of attack. While the pressure coefficient results were on par with CFL3D for the most part, the integrated drag value was significantly over-predicted due to an inaccurate prediction of the flow acceleration at the leading edge on the upper surface of he airfoil. This discrepancy is due to an unsuitable BC for the far-field boundaries and a grid that may be too coarse considering the weak treatment of the solid wall BC.

Both NX Flow and Syn3D were compared against experimental data for the RAE2822 airfoil, this time in compressible flow. Both codes deviated slightly from the expected results, although NX Flow exhibited the same problems as with the NACA0012. Syn3D was able to accurately predict the location of the shock for the transonic case.

Preliminary results were obtained for a high-lift configuration with syn3D, which are qualitatively correct. However, the computational grid is still much too coarse; improvements to the discretization as well as tuning of the parameters may be necessary in order to run a finer grid.

To conclude, it can be said that the implementation in both codes corresponds to the discretization, i.e. no programming errors have been made. This doesn't mean, however, that improvements cannot be made, either to the underlying solver or to the SA implementation itself.

5.2 Future work

To further validate, several items are suggested. First, what seems to be a boundary condition issue in syn3D, which led to high peaks in the eddy viscosity for the TMR cases, should be investigated. This may decrease the dependence on grid refinement.

Second, a grid study should be performed on the NACA0012 and RAE2822 to see if better results can be obtained.

Third, the effects of strong and weak boundary conditions for viscous walls in NX Flow should be compared.

Fourth, a far-field BC should be implemented into NX Flow, so as to be able to run external flow simulations on structured grids, which are popular for such flows. For best results, this BC should employ a vortex correction in order to reduce influence of the far-field condition on the solution and also reduce dependence on the extent of the computational grid [36], [37].

Finally, the high-lift configuration should be solved on a suitable grid with syn3D, along with a transition model.

5.3 Closing remarks

The primary goal of this project was to add a turbulence model to syn3D and NX Flow. Furthermore, it is important to validate a new feature once it is added to a product. The obvious reason is to ensure that the feature was implemented correctly. This goal has been achieved. However, the main challenge with testing in CFD is that all components of a solver are interlinked, as opposed to a typical software product where *separation of concerns* is a key design principle and where each component serves a single purpose, hence should not be tightly coupled to the rest of the code. In a CFD solver, there exists but one purpose shared by all components: to accurately solve the fluid flow equations. For this reason, it is difficult to pinpoint the exact source of an error or discrepancy when it occurs.

Thus, the secondary goal of this work, especially in the case of NX Flow, was to test the existing software against cases it had not run before. This turned out to be almost as important as the primary goal: many issues, which were previously unknown, were resolved over the course of this project. As discussed in the previous section, there still remains a variety of challenges for the developers to work on.

Bibliography

[1] P. R. Spalart and S. R. Allmaras, "A one equation turbulence model for aerodynamic flows", *AIAA Paper 92-0439*, 1992.

[2] H. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education Limited, 2007, ISBN: 9780131274983. [Online]. Available: `https://books.google.ca/books?id=RvBZ-UMpGzIC`.

[3] D. C. Wilcox *et al.*, *Turbulence modeling for CFD*. DCW industries La Canada, CA, 1998, vol. 2.

[4] Wikimedia Commons, *File:Studies of Water passing Obstacles and falling.jpg — Wikimedia Commons, the free media repository*, `https://commons.wikimedia.org/w/index.php?title=File:Studies_of_Water_passing_Obstacles_and_falling.jpg&oldid=239927287`, Accessed: 2017-06-23, 2017.

[5] S. B. Pope, *Turbulent flows*, 2001.

[6] P. J. Roache, K. N. Ghia, and F. M. White, "Editorial policy statement on the control of numerical accuracy", *Journal of Fluids Engineering*, vol. 108, no. 1, pp. 2–2, 1986.

[7] J. Blazek, *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

[8] J. A. Carlson, A. Jaffe, and A. Wiles, *The millennium prize problems*. American Mathematical Soc., 2006.

[9] B. Munson, A. Rothmayer, and T. Okiishi, *Fundamentals of fluid mechanics, 7th edition*. Wiley, 2012, ISBN: 9781118214596. [Online]. Available: https://books.google.ca/books?id=GQMcAAAAQBAJ.

[10] C. Rumsey, *The Spalart-Allmaras turbulence model*, https://turbmodels.larc.nasa.gov/spalart.html, Accessed: 2017-06-27.

[11] C. L. Rumsey, "Apparent transition behavior of widely-used turbulence models", *International Journal of Heat and Fluid Flow*, vol. 28, no. 6, pp. 1460–1471, 2007.

[12] M. L. Shur, M. K. Strelets, A. K. Travin, *et al.*, "Turbulence modeling in rotating and curved channels: assessing the Spalart-Shur correction", *AIAA Journal*, vol. 38, no. 5, pp. 784–792, 2000.

[13] J. Dacles-Mariani, G. G. Zilliac, J. S. Chow, and P. Bradshaw, "Numerical/experimental study of a wingtip vortex in the near field", *AIAA Journal*, vol. 33, no. 9, pp. 1561–1568, 1995.

[14] J. Dacles-Mariani, D. Kwak, and G. Zilliac, "On numerical errors and turbulence modeling in tip vortex flow prediction", *International Journal for Numerical Methods in Fluids*, vol. 30, no. 1, pp. 65–82, 1999, ISSN: 1097-0363. DOI: 10.1002/(SICI)1097-0363(19990515)30:1<65::AID-FLD839>3.0.CO;2-Y. [Online]. Available: http://dx.doi.org/10.1002/(SICI)1097-0363(19990515)30:1%3C65::AID-FLD839%3E3.0.CO;2-Y.

[15] S. R. Allmaras and F. T. Johnson, "Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model", in *Seventh international conference on computational fluid dynamics (ICCFD7)*, 2012, pp. 1–11.

[16] P. R. Spalart, "Strategies for turbulence modelling and simulations", *International Journal of Heat and Fluid Flow*, vol. 21, no. 3, pp. 252–263, 2000.

[17] P. G. Tucker, C. L. Rumsey, P. R. Spalart, R. B. Bartels, and R. T. Biedron, "Computations of wall distances based on differential equations", *AIAA Journal*, vol. 43, no. 3, pp. 539–549, 2005.

[18] C. Rumsey, *Turbulence Modeling Resource*, https://turbmodels.larc.nasa.gov, Accessed: 2017-06-10.

[19] P. Tucker, "Hybrid Hamilton-Jacobi-Poisson wall distance function model", *Computers & Fluids*, vol. 44, no. 1, pp. 130–142, 2011.

[20] A. G. Belyaev and P.-A. Fayolle, "On variational and PDE-based distance function approximations", in *Computer Graphics Forum*, Wiley Online Library, vol. 34, 2015, pp. 104–118.

[21] A. Jameson, W. Schmidt, and E. Turkel, "Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes", in *14th fluid and plasma dynamics conference*, 1981, p. 1259.

[22] J. VonNeumann and R. D. Richtmyer, "A method for the numerical calculation of hydrodynamic shocks", *Journal of applied physics*, vol. 21, no. 3, pp. 232–237, 1950.

[23] S. K. Nadarajah, "The discrete adjoint approach to aerodynamic shape optimization", PhD thesis, Stanford University, 2003.

[24] A. Jameson and T. Baker, "Solution of the Euler equations for complex configurations", in *6th Computational Fluid Dynamics Conference Danvers*, 1983, p. 1929.

[25] J. Ferziger and M. Peric, *Computational methods for fluid dynamics*. Springer Berlin Heidelberg, 2012, ISBN: 9783642560262. [Online]. Available: https://books.google.ca/books?id=BZnvCAAAQBAJ.

[26] L. Thomas, "Elliptic problems in linear differential equations over a network: Watson scientific computing laboratory", *Columbia Univ., NY*, 1949.

[27] J. N. Reddy, *An introduction to the finite element method*. McGraw-Hill New York, 1993, vol. 2.

[28] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, K. Rupp, B. Smith, S. Zampini, H. Zhang, and H. Zhang, *PETSc Web page*, http://www.mcs.anl.gov/

petsc, Accessed: 2017-07-01, 2016. [Online]. Available: http://www.mcs.anl.gov/petsc.

[29] Siemens PLM, *Simulating fluid flow for complex parts and assemblies*, https://www.plm.automation.siemens.com/en_us/Images/6639_tcm1023-4403.pdf, Accessed: 2017-06-27, 2013.

[30] B. Baliga and S. Patankar, "A new finite-element formulation for convection-diffusion problems", *Numerical Heat Transfer*, vol. 3, no. 4, pp. 393–409, 1980.

[31] ——, "A control volume finite-element method for two-dimensional fluid flow and heat transfer", *Numerical Heat Transfer*, vol. 6, no. 3, pp. 245–261, 1983.

[32] J. Reddy and D. Gartling, *The finite element method in heat transfer and fluid dynamics, second edition*, ser. Computational Mechanics and Applied Analysis. Taylor & Francis, 2000, ISBN: 9780849323553. [Online]. Available: https://books.google.ca/books?id=bF8F1qPhq2kC.

[33] C. Rhie and W. L. Chow, "Numerical study of the turbulent flow past an airfoil with trailing edge separation", *AIAA Journal*, vol. 21, no. 11, pp. 1525–1532, 1983.

[34] G. Schneider and M. Raw, "Control volume finite-element method for heat transfer and fluid flow using colocated variables—1. Computational procedure", *Numerical Heat Transfer, Part A Applications*, vol. 11, no. 4, pp. 363–390, 1987.

[35] J. Nordström, S. Eriksson, and P. Eliasson, "Weak and strong wall boundary procedures and convergence to steady-state of the Navier-Stokes equations", *Journal of Computational Physics*, vol. 231, no. 14, pp. 4867–4884, 2012.

[36] C. Rumsey, *Effect of Farfield Boundary*, https://turbmodels.larc.nasa.gov/naca0012_val_ffeffect.html, Accessed: 2017-06-10.

[37] J. L. Thomas, "Far-field boundary conditions for transonic lifting solutions to the Euler equations", *AIAA Journal*, vol. 24, no. 7, pp. 1074–1080, 1986.

[38] J. Thibert, M. Granjacques, and L. Ohman, "NACA0012 Airfoil AGARD Advisory Report No. 138", *Experimental Data Base for Computer Program Assessment*, A1–9, 1979.

[39] J. H. Morrison, "Numerical Study of Turbulence Model Predictions for the MD 30P/30N and NHLP-2D Three-Element Highlift Configurations", *NASA report NASA/CR-1998-208967*, 1998.