UNITED KINGDOM · CHINA · MALAYSIA

GAS TURBINE AND TRANSMISSIONS RESEARCH CENTRE

Improving accuracy and efficiency of CFD simulations of complex geometry and high-Re turbine blade cooling systems.

ZACHARZEWSKI, Piotr

MEng, AMIMechE

Thesis submitted to the University of Nottingham in partial fulfilment of the requirements for the award of Doctor of Philosophy in Mechanical Engineering.

October 2021

To dad.

Acknowledgments

Having spent seven years completing this work, I was fortunate to be acquainted with many brilliant minds all of whom contributed to my learning, progress of this work as well as many great memories that will be with me forever. Firstly I would like to thank my supervisor Dr. Richard Jefferson-Loveday for offering me a scholarship and all the time and patience he had for me. Although we did not always agree I feel I went from not knowing any programming language or numerical methods at all to creating my own CFD code from scratch under Richard's guidance. He has also given me ample opportunities to attend national and international workshops, conferences and technical meetings and always found time to explain a topic to me. I am very grateful for being able to learn from Richard and can only hope to return professional kindness in the future.

I had the privilege to meet Prof. Henry Power early on in my doctoral studies. Henry never denied a meeting to explain a complex numerical method to me and did so in a fantastically clear and technical way. His depth and breadth of knowledge was very humbling and he will be dearly missed.

Thank you to all members of the G2TRC who made the office a great and vibrant place to work; Jee, Dafne, Keith, Stephen, Bruce, Evgenia, Ben and many more with whom I interacted less often - thank you for creating an atmosphere of friendship and collaboration despite all the deadlines and other pressures we were all under. Although most people I met have now graduated or moved on to other roles I have many fond memories, including one particular Friday lunch where a nearby pub offered a Flaming Challenge in which we duly took part.

I am grateful to the head of group Prof. Herve Morvan and recently Dr. Carol Eastwick for steering the group and always being supportive and understanding . I thank Dr. Donald Giddings, not only for assuming the role of my internal assessor but all the productive discussions and reviews we had. I also thank Donald for all the Air-con and fluid flow labs when I finally understood in depth the inner workings of my fridge.

Last but not least a massive thank you to my colleague in Rolls-Royce, Dr. Luigi Capone and manager Tony Phipps for being very supportive of my PhD and allowing me the flexibility to complete the doctoral work alongside a full time job.

During this work I used several computational resources and it would be very hard to manage without them. Large portion of simulations and code development was done on the University of Nottingham HPC Minerva and its G2TRC segment, both administered by the always helpful Colin Bannister. As the work progressed I also used the MidPlus HPC which provided a good alternative and different hardware to test the robustness of my code.

Thank you to Dr. David Henty and the whole ARCHER team at the University of Edinburgh for giving me access to the ARCHER HPC and organising splendid courses and documentation that allowed me to make the most of this resource.

Finally, many of the turbine blade simulations and meshing were done on the CFMS HPC, courtesy of Rolls-Royce.

Abstract

Flow as well as geometry inside turbomachinery components such as turbine blades is complex and difficult to handle accurately. Computationally affordable RANS simulations are often not suitable and at least partly resolving simulations such as LES or hybrid RANS-LES are needed for sufficient accuracy in the area. Within industrial turbine design, these are not deployed routinely, if at all, due to their presently unaffordable computational cost and time-consuming grid generation for realistic complex geometries. Often even minor changes in geometry lead to relatively substantial changes in the CFD result.

GPUs and other modern heterogeneous hardware offer computational capability that is order of magnitude cheaper and takes significantly less space, enabling small scale 'HPC-like' workstations at every engineer's desk. However so far they remain mostly unharnessed due to difficulty of creating structured datasets in memory required to utilise the GPUs effectively. Those difficulties are amplified by the need to avoid even minor changes to geometry. While unstructured or hybrid grids can be used on massively parallel platforms, it is believed the inter-thread communication usually prohibits effective scaling and GPU remains mostly idle, negating the benefits. Within CFD, structured grids naturally lead to structured datasets hence provide much faster, linear, memory access making them an excellent candidate for GPU platforms.

In addition to computational and grid generation issues, accuracy needs to be improved in turbomachinery internal flows. LES is usually still too expensive as a turbulence treatment for the design process and hybrid RANS-LES methods such as DES are explored for daily industrial engineering applications. A relatively new model is the SST-SAS which aims to improve accuracy of lower cost URANS simulations by forcing the turbulent viscosity lower where the grid is sufficiently refined to resolve the largest turbulence fluctuations, as judged by the model's 'Q' factor. This results in the simulation 'adapting' to local scales of grid and turbulence hence generalise on DES concepts, applying the resolving switch in the entire domain, not just near the wall. To further improve the SST-SAS model concept of localised artificial forcing is tried. The idea is to achieve finer control over the SAS-SST model and force turbulent viscosity lower in user-defined areas on top of the model detected ones. This aims to address some model deficiencies found in previous works. Overall, the SST-SAS simulations

should allow to only refine the grid in areas where resolving of smaller scales is judged to be necessary for accuracy while performing a time-accurate unsteady simulation. The current work presents an assessment of this model using standard LES metrics such as two point correlations and energy spectra as well as experimental measurements. DES and LES was also ran as reference on the same SST-SAS-suitable grids for reference. It is found that for the present application the artificial forcing significantly improves the revealed deficiencies of the SST-SAS model and simulating flow with sufficiently high turbulence, such as a rotating ribbed channel, can similarly trigger the resolving mode. Unfortunately however it is also found that given the same grid, DES or even ’coarse grid LES’ produces more accurate solution than SAS-SST with artificial forcing. If the grid has to be refined more than it is in the present study to obtain satisfactory solution with the SST-SAS model, the author concludes LES would most likely produce better result as refinement would make the grid conform to the strict LES requirements rendering SAS-SST approach obsolete.

In the current work a GPU-accelerated IBM code is presented, verified and then validated on classical test cases. It is shown that the combination of Immersed Boundary Method with an implicit time scheme using the high-level OPS parallelisation library and novel ADI tridiagonal solver provides geometric flexibility as well as computational performance. Additional benefit of the OPS library is that single source code exists for major hardware platforms and the parallel implementation is decoupled from the scientific codebase, making the code scalable and easily adaptable to any emerging, future architectures.

Preface

The work described in this thesis was carried out in the Gas Turbine and Transmissions Research Center (G2TRC), Department of Mechanical, Materials and Manufacturing (M3), Faculty of Engineering, University of Nottingham between September 2015 and May 2018 full time and continued until October 2021 part time. This document is the result of my own work and includes nothing that is result of collaboration unless explicitly stated otherwise. No part of this work has been submitted elsewhere other than the stated peer-reviewed venues. The thesis contains 7 chapters with a total of approx. 57,000 words and 97 figures.

[illegible]

Piotr Zacharzewski

List of publications

Current work resulted in the following peer-reviewed publications:

1. Zacharzewski P, Simmons K, Jefferson-Loveday R, Capone L. Evaluation of the SST-SAS Model for Prediction of Separated Flow Inside Turbine Internal Cooling Passages. ASME. Turbo Expo: Power for Land, Sea, and Air, Volume 5B: Heat Transfer ():V05BT11A003. doi:10.1115/GT2016-56117.

2. Zacharzewski P., Jefferson-Loveday R. J. and Morvan H. (2017) SAS – SST simulations of the flow and heat transfer inside a square ribbed duct with artificial forcing. In: 23rd ISABE Conference, 3-8 Sep 2017, Manchester, UK

3. Rouse, J.P., Zacharzewski P., Hyde C. J., Jefferson-Loveday R., Morris A., Kyaw S. T., Case study investigation into the effects of spatially and temporally dependent convection coefficients on the fatigue response of a power plant header component. Multi-scale Fatigue, Fracture & Damage of Materials in Harsh Environments, IUTAM, 2017.

4. Implicit GPU accelerated Numerical simulations via the use of Ghost Cell Immersed Boundary Method. ASME turbo expo 2020: Turbomachinery technical conference & exposition. September 21-25 2012, Virtual, Online. GT2020-15844.

5. Rouse, J.P., Zacharzewski P., Hyde C. J., Jefferson-Loveday R., Morris A., Kyaw S. T., A case study investigation into the effects of spatially dependent convection coefficients on the fatigue response of a power plant header component, Int. J. Fatigue, 2017.

Posters and presentations:

1. Presentation: SST-SAS unsteady simulations for internal cooling passages, ASME Turbo Expo, June 2016, Seoul.

2. Presentation: SAS – SST simulations of the flow and heat transfer inside a square ribbed duct with artificial forcing, ISABE, september 2017, Manchester.

3. Presentation: GPU parallel Immersed Boundary CFD code, Research Software Engineering meeting, July 2017, Nottingham.
4. Presentation: Implicit GPU parallel Immersed Boundary CFD code, HPC conference, May 2017, Nottingham.
5. Presentation: Improving computer simulations for aerospace, LINK '16 student conference, june 2016, Nottingham.
6. Presentation: GPU accelerated computations of turbulent, high-Re internal turbomachinery flows, Multicore and Manycore Algorithms to Tackle Turbulence user meeting, march 2018, London.
7. Poster: Improving accuracy of CFD of complex geometry and high-Re turbine blade cooling systems, March 2016, Rolls-Royce Annual Review, Derby.
8. Poster: SAS – SST simulations of the flow and heat transfer inside a square ribbed duct with artificial forcing, March 2016, Rolls-Royce Annual Review, Derby.
9. Poster: Implicit GPU parallel Immersed Boundary CFD code, Engineering Research Showcase, April 2016, Nottingham.

Contents

i List of Figures

ii List of Tables

iii Nomenclature

Put a note about einstein notation adopted throughout, give an example small equation

Acronyms

ACARE	Advisory Council for Aviation Research in Europe
ADI	Alternating Direction Implicit
BCG	Body Conformal Grid
BI	Body Intercept
B.C.	Boundary Condition(s)
CAA	Computational Aero Acoustics
CAD	Computer Aided Design
CFD	Computational Fluid Dynamics
CHT	Conjugate Heat Transfer
CPU	Central Processing Unit
DES	Detached Eddy Simulation
EIP	Extra Image Point
FE	Finite Element
GCIBM	Ghost Cell IBM
GN	Ghost Node
GPGPU	General Purpose GPU
GPU	Graphics Processing Unit
HPC	High Performance Computing
HPT	High Pressure Turbine
IBM	Immersed Boundary Method
IDDES	Improved Delayed DES
IIBM	Improved IBM
IP	Image Point
LBM	Lattice Boltzmann Method
LES	Large Eddy Simulation
MPI	Message Passing Interface
N-S	Navier-Stokes

OPS	Oxford Parallel library for Structured applications
PDE	Partial Differential Equation
PIP	Point In Polyhedron
RANS	Reynolds Averaged Navier Stokes
SAS	Scale Adaptive Simulation
SFC	Specific Fuel Consumption
SRS	Scale Resolving Simulation
SST	Shear Stress Transport
TET	Turbine Entry Temperature
TKE	Turbulent Kinetic Energy
URANS	Unsteady RANS
VM	Vortex Method

Greek symbols

ε	Turbulence dissipation rate $[Jkg^{-1}s^{-1}]$
ε_{ijk}	Permutation symbol, Levi-Cita
η_i^n, ξ_i^n, d_i^n, ω^n	Random numbers
η, ζ, ξ	Transformed curvilinear coordinates
κ	Von Karman constant, 0.41
$\vec{r}$	Relative displacement vector $[m]$
V	Vorticity $[s^{-1}]$
τ_t	Turbulence time scale $[s]$
ρ	Density $[kgm^{-3}]$
σ_ϕ	SAS model constant, 2/3
γ	Resolved to total TKE
Ω	Rotational speed $[rads^{-1}]$
ω	Turbulence specific dissipation rate $[s^{-1}]$
ζ_2	SAS model constant, 1.47

Roman symbols

C	SAS model constant, 0.11
C_F	Q criterion constant, 0.5
D_h, L	Reference hydraulic diameter, 0.149 [m]
$F_{,i}$	Forcing source term
h	Rib height
H	Axisymmetric hill height
k, TKE	Turbulence Kinetic Energy [m^2s^{-2}]
L_t, L	Modelled turbulence length scale [m]
L_{VK}	Von Karman length scale [m]
N	Number of harmonic modes, 100
Nu_0	Reference Nusselt number
Pr	Prandtl number
$R_{AB}(\vec{r})$	Normalised correlation coefficient
$Re_{L,u}$	Reynolds number based on length L and velocity u
Ro	Rotation number
S	Strain rate [s^{-1}]
Δh	Maximum cell extent [m]
Δt	Timestep [s^{-1}]
$u^{'}$	Fluctuating x velocity component [ms^{-1}]
$\overline{u}$, $\overline{u}_{in}$	Mean or bulk velocity [ms^{-1}]
$u_{f,i}$	Forcing velocity
$\vec{x_A}$	Absolute displacement vector [m]
x	Streamwise coordinate
y	Wall-normal coordinate
z	Cross-stream coordinate

Chapter 1

1 Introduction

The goal of this chapter is to provide a wider societal, economic and industrial context of the current work, present clear objectives and motivation for research as well as introduce structure of the thesis. Work on this project started in September 2015. Majority of research work and much of this chapter was completed before the global corona virus pandemic of 2020/2021 and a subsection is added to reflect the impact of events on the present work.

1.1 Aerospace industry outlook pre-2020

According to the two major airframe manufacturers, US based Boeing [6] and Europe based Airbus [7] global air traffic is expected to increase by 4% – 5% annually in the next 20 years. This corresponds to doubling today's air traffic in 15 years. Such growth will require major investments in new ground infrastructure, aircraft operating strategies as well as new aircraft themselves. To limit, or even eliminate altogether, the increasing environmental impact on the planet of the global aircraft fleet efficiency is ever more important. There is also the aspect of increasing public awareness of the environmental as well as medical concerns. Aircraft noise has been linked to human cardiovascular diseases [8, 9, 10] in both the US and the UK. All these form what is sometimes referred to as "aviation related nuisances" and resulted in increased legislation in the field. It has been established as early as in 2000 that evolutionary, incremental changes in technology will be far outpaced by the aviation-related nuisances if the 5% annual air traffic increase is sustained [11] [12]. Both the European and North American governmental agencies set out regulations that ensure the aerospace industry must introduce step-change innovations. They are the National Aeronautics Research and Development Plan by the US National Science and Technology Council [13] and the report titled "Creating innovative air transport technologies for Europe" set by the Advisory Council for Aviation Research in Europe (ACARE) [14]. They both set similar, aggressive targets limiting aviation-related nuisances by 2020, namely:

- Increasing lift to drag ratio by 25% (US target)

- Reduce fuel burn by 70% per passanger-kilometer compared to Boeing 737/CFM56 (US target)
    - reduce fuel consumption and CO2 emissions by 50% per passenger kilometre compared to 2000 average values (EU target)
- reduce noise by 62 dB cumulative below current Federal Aviation Authority (FAA) standard for large subsonic jet aircraft (US target)
    - reduce perceived noise by 50% compared to 2000 average values (EU target)
- reduce NOx emissions by 80% below current international standard (US target)
    - reduce NO_x emissions by 80% compared to 2000 average values (EU target).

The European and American target values are not dissimilar and both require significant improvements in current technologies (i.e. physics-based breakthroughs) as well as the way aircraft are operated. The Cleansky project [15] estimates up to 25% contribution to 2020 targets will come from better engine or aircraft design and approximately 10% from more efficient aircraft operation. All of these are part of a longer term strategy and vision named “Flightpath 2050” [14]. Apart from environmental concerns and targets imposed by the governments there is the industrial business case of reducing cost of operation per aircraft if the machine is made more efficient. A whole new issue is the time it takes for the investment in new technologies to pay off.

Many different technologies contribute to designing an efficient aircraft. Structural, Finite Element (FE) simulations predicting stresses and deformations or Computational Aero Acoustics (CAA) predicting noise are only two examples. The key role however in physics based breakthroughs is played by the use of Computational Fluid Dynamics (CFD) in both industry and academia; CFD is often the starting point and input to other models such as FE as well as performance. Improvements of CFD methods are necessary [16] [17] to achieve substantial progress in both the whole aircraft and aeroengine efficiency. The present work focuses on aeroengine components, specifically turbine blade cooling technologies.

Some of the key practical metrics aeroengine designers use to measure efficiency include Specific Fuel Consumption (SFC) or thermal efficiency [18, 19]. The main ways these metrics are improved are by increasing Turbine Entry Temperature (TET) or reducing pressure losses associated with turbine blade cooling [20]. As the first few stages of a typical turbine operate at temperatures beyond their melting point relatively “cool” air at 600-700°C is taken from compressors [21] to cool turbines reducing the overall air that is combusted and hence used to directly propel the aircraft. Losses associated with this bleed air must be minimised for a more efficient aircraft. There is also the benefit of increased component life if accurate simulation is performed; even 30°C (2% with respect to 1700°C present) difference in operating temperature makes significant difference in creep or fatigue life of turbine components. It is interesting to note here how much the TET has changed; 20 years ago TET was about 1300°C [22] while now it can be as high as 2200°C. It has been established that improvements in turbine technologies will be the major contributor towards increasing overall aeroengine efficiency, both in terms of SFC and thermal efficiency [23]. Overall, improvements in turbine blade cooling technology give high impact for the given investment and accurate prediction of blade temperature has direct impact on aeroengine efficiency [24].

Another very important aeroengine metric is the Bypass Ratio (BPR). It is ratio of air that that travels around the core and is used solely for propulsion to air that is ingested by the core. While core exhaust also creates thrust it tends to be significantly less than the ’bypassed’ air. Increasing BPR is always the goal of designers of commercial aeroengines and results in higher thrust per fuel burnt (SFC) and lower fan speed hence noise. Increasing BPR however imposes higher temperatures and stresses inside the core of the engine, among other practical challenges such as the engine being simply too large for taxiing. Increasing BPR is always a multi-faceted engineering trade-off but almost always requires better cooling or heat resistance of the core components and that is the contribution of the current work to the BPR.

1.2 Aerospace industry outlook post-2020

The recent widespread travel restrictions have drastically changed the aerospace industry dynamic and likely affected the long term prognosis made pre-2020. Some analysis has been done to understand impact of the recent events on the current work of fluid flow inside turbine blade cooling systems.

The first observation is that the CFD fundamentals such as insufficient RANS accuracy, cost of LES and time consuming mesh generation for the present application are mostly unchanged. One may even hazard a statement that cost and time efficiency of simulations is even more important in the current economic climate.

However, as the societies around the world are moving to less carbon-producing technologies at increasing pace, demand hence funding for combustion-based technologies may appear to be in rapid decline. There are even calls to drastically reduce air travel altogether, for instance in favour of electrical trains or simply virtual meetings. While electrical or hydrogen propulsion systems are viable to be commercialised for shorter range travel, it does not appear the same will be true for 'long haul' journeys. This is due to the comparatively low energy density of current batteries or hydrogen storage systems; for instance the battery weight required for fully electric journey from London to New York would effectively prohibit such flight. The argument is similar for hydrogen storage; either weight or container pressure would have to be enormous with current hydrogen storage technologies for long haul flights. While those energy mediums are rapidly developing, is it not expected they will not be mature enough for long haul commercial aviation use for at least another 40 years, if not more.

A social argument can also made, that while air travel does indeed account for 2% of global $C0_2$ emissions (and increasing % as other industries decarbonise much faster than aerospace) it is unlikely that our society will choose to avoid this type of transportation in the future. The global society is more connected than ever before, people have been used to the comforts of fast international travel and the aerospace, travel, airport and other related industries are a major contributor to the economy. There are approximately 25,000 aircraft currently on the planet, with that number expected to double in the next 20 years due to increasing demand for air travel.

One of the very few commercially viable options to decarbonise long haul air travel are synthetic Sustainable Aviation Fuels (SAF). While there are many different methods of creating such fuels, they are generally obtained by capturing carbon dioxide from the atmosphere with the use of electricity and can then be stored for combustion during flight. If the electricity used to produce such fuels is sustainable, no $C0_2$ is added to the atmosphere as a result. The synthetic fuel can then be used on existing aircraft with very little modification to the engines.

It is however expected that the cost of SAF will be at least double that of current kerosene-based fuels. It is the cost argument that leads the author to believe investment in increasing aeroengine efficiency is likely to increase in the mid to long term if SAF fuels are more widely adopted or mandated by governments around the planet.

In summary, while the world is rethinking priorities in the short term and aero-engine efficiency may temporarily not be on top of the list, the author believes the present work will still be very much relevant for our society in the future.

1.3 Computational and numerical matters in CFD

Generally, there are only two ways efficiency and effectiveness of CFD can be improved. One is improving algorithms/mathematics (be it core numerical schemes, meshing algorithms or using Machine Learning) or their implementation (hardware, libraries, automated meshing).

Numerical schemes tend to take long time to be tested and mature enough for commercial use. The majority of CFD codes nowadays use the Finite Volume scheme devised largely in 1970s [25] and RANS turbulence modelling originating with Reynolds in the early 20 century with more recent modifications such as that by Menter 2008 [26]. The progress in this area is generally slow, despite high overall investment in both scientific and industrial communities.

Hardware and libraries on the other hand tend to develop rapidly, largely due to their more widespread use and non-specialised nature (e.g. CPUs). An example can be increasing reliance of hardware on vector processing. The problem with non-specialised approaches like this is that not all advances are applicable to specialised CFD application and often to fully utilise the 'flop' rating of the latest HPCs one must maximise the use of approaches such as vector processing, which is not always possible or efficient due to the CFD algorithms.

An argument is also made that a typical lifespan of an industrial CFD code is approximately 40-50 years and there are likely to be several major computational advances over such a time frame. Some of the advances will be efficiency improvements, others will be an entire industry wide paradigm shift to new hardware. For a CFD solver to remain viable or remotely competitive the program must be able to use those new advances. This typically requires major rewrite, re-validation and re-certification of the code with relevant aviation authorities which usually has a very high cost. Cost of engineering simulation software rewrites can be so high that is effectively prohibitive, leading to that software being abandonded in favour of something else that utilises the latest computational capabilities. A new code then must be chosen and for a safety critical application such as aeroengines, all the routines and algorithms must be verified and validated anew, again resulting in large cost. A proposed solution to this issue is a 'future-proof' high level library where scientific and computational elements are relatively independent of each other and can be upgraded in a 'modular' way. Some even argue this is not an optional requirement for new CFD codes [27, 28] and the use of such approach will be investigated here

It should also be mentioned that the specific type of internal flow targeted in the present work is not uncommon in other parts of an aeroengine or indeed other industries, such as steam flow inside a ground powerplant element [29]. It is therefore expected that the present work will be applicable to a variety of practical industrial analysis.

1.4 Motivation for research

The current problem is multi-faceted and requires trade off between three key matters:

1. solution's accuracy (RANS-LES, SAS-SST, LES)
2. engineer's time (e.g. manipulating and clean-up of geometry, creating grid of sufficient quality for the method chosen)
3. computational cost

The balance between these trade offs is ever changing due to development of underlying technologies. They are not unique here and are present in many CFD applications. However from the author experience of working for a major aeroegnine manufacturer what is unique in the present application is that the balance is very delicate and getting it even slightly wrong can easily result in overwhelming cost (either computational due to very large meshes or engineer time manipulating the geometries and meshes) or significantly underperforming or even unsafe aeroengine components.

This work will attempt to advance the turbine fluid flow simulation technology in two ways; one is testing of new and promising turbulence models in a range of common conditions, the other is an entirely new approach that will aim to drastically improve computational as well as human time needed for analysis.

While it is challenging for one doctorate research project to solve the problem entirely it is hoped useful conclusions and pointers for how better to optimise the trade-offs in the long term will be provided.

1.5 Objectives of research

The aim of the present work is to increase accuracy of CFD simulations of internal cooling passages without increasing, or even while reducing the total cost of the simulations for the same accuracy. Ultimately the flow field is necessary to compute Heat Transfer through the blade and predict metal temperatures. Good knowledge of the thermal field in turn allows one to make more informed design choices hence turbine efficiency and life of the blade are improved. Both fluid flow and heat transfer simulations will be performed.

In order, objectives of the work are:

- Perform simulations using SST-SAS model on increasingly complex and realistic geometries and conditions. Evaluate SST-SAS performance against LES and DES. **(research)**
- Recognising a step change in methods is required, to create and parallelise a CFD code used for faster computation on several massively parallel platforms where performance and portability is maximised. **(research & implementation)**
- To research, select and implement the most appropriate at the time Immersed Boundary Method (IBM) to significantly ease grid generation for complex geometries and allow wider use of structured grids **(research & implementation)**
- To validate the GPU IBM work using standard test cases of increasing complexity and with experimental or analytical data, with one being of significant complexity and relevance to the turbine blade internal cooling passages flow **(verification & validation)**
- To investigate how to apply the current techniques with Conjugate Heat Transfer simulations needed for internal cooling **(research)**
- Find one other industrial application to which the present research may apply and perform at least a preliminary study **(exploitation)**

1.6 Thesis structure

Chapter 2 presents a review of techniques, computational matters, turbulence modelling as well as numerical methods suitable to achieve the goals set out, with emphasis on in-

ternal cooling of turbine blades. A broad survey of existing works will be conducted to underpin the reasoning and choice of methods for implementation.
In Chapter 3 the final choice of methods and techniques is presented in more detail and theory explained before implementation. It is also the intention to expand the formulations in ’as-implemented’ form for clarity and future reference.
Chapter 4 contains results of hybrid RANS-LES model testing on channel geometries with rotation and comparison with experimental measurements. Models tested include SST-SAS, DES variants and LES. Standard metrics to measure resolved vs. modelled turbulence content were used and recommendations made for study in further chapters. Content of this chapter is mostly drawn from two ASME peer-reviewed conference papers published during this PhD.
Chapter 5 is an attempt to create a hybrid RANS-LES simulation on near-realistic yet free of sensitive Intellectual Property or Export Control considerations turbine blade geometry with internal cooling passages using conclusions from the previous chapter.
Chapter 6 presents work done developing and validating GPU Immersed Boundary Method (IBM) code, including validation results of the necessary prerequisite algorithms such as nearest wall distance calculation.
Recommendations for future work and final conclusions are drawn in Chapter 7.

Recognising that digesting a technical thesis is a substantial task each chapter will begin with a short introduction and set out goals and questions it intends to answer. Every chapter will also conclude how these goals were achieved, brief conclusions and set background for the next chapter.

Chapter 2

2 Literature review

2.1 Chapter introduction

The aim of this chapter is to explore existing scientific and technical literature in detail, provide rationale for the chapters ahead and build a foundation for the present work. Since the current problem of internal cooling, as indeed the whole field of CFD, draws from multiple disciplines, several different topics will be explored. While this unfortunately results in a lengthy review of past works, a summary will be provided. For the same reason it is sometimes challenging to explore all the literature in as much depth as a keen scientific mind would strive to.

It should also be mentioned this work was started in October 2015 and most of the results were obtained before the end of 2018. Given rapid advancements in some areas, in particular computer hardware or high level libraries, every effort was made to stay up to date. However there had to be a point at which the Doctoral research work stops and writing up the results begins. For that reason some of the conclusions which were based on state of art at the time may appear odd or out of place at the time of reading this work. A good example is the use of Improved IBM method published in 2017 by Chi et.al. [30]. Originally the Ghost Cell IBM of [31] was implemented and even some results obtained however after reading the new IIBM work of Chi [30] in 2017 it was clear that this method is more promising, accurate and likely more stable hence some of the code was rewritten to accommodate the IIBM. In reality there is a limited scope for such modifications as the work nears completion and a discrete cut-off point must at some point be decided.

2.2 Immersed Boundary Methods

2.2.1 Introduction

Most of the commonly used CFD tools require a body-conformal grid (BCG). Generating a suitable structured BCG is time consuming and from the author experience is often a major bottleneck in industrial simulations preventing regular use of a structured solver. It is often not possible at all due to geometry complexity. Creation of a suitable unstructured BCG can take little user time, it can be computed in parallel and is often even automated, but usually results in a grid with massive cell count, often an order of magnitude more than a suitable structured BCG grid. Additionally quality of unstructured grids can easily lead to lower fidelity of the solution or stability issues during convergence. Additionally if the designers wish to use advanced CFD tools, e.g. LES in design optimisation grid must really be kept mostly structured and as small as possible and meshing bottlenecks must be eliminated in the structured grid generation space. There are a myriad of BCG grid-generation tools, both commercial and open source, structured, unstructured but none of them solve the two problems simultaneously. It is usually a trade off between time a user spends generating grid, grid quality and cell count.

The present research looks at grid generation from a different perspective. Instead of generating a Body Conformal Grid to impose Boundary Conditions, two grids are generated. One is a fully structured non-BCG volume grid that spans over the entire fluid and solid domain as shown on Figure 2.1; the other is a Body Conformal Grid, but only surface or perimeter in 2D must be meshed; the surface (line in 2D) grid may be structured or unstructured and is usually relatively easy to generate for any geometry even in an automated non-interactive fashion (e.g. stereo-lithography (STL)). The solution is stored and progressed on the Cartesian grid while the surface grid is only used to impose BCs implicitly via forcing terms or suitable modification of near-wall cells, e.g. to set velocity to 0 at certain points. The surface grid is only needed in the preprocessing stage to identify on the Cartesian grid exactly which nodes are solid, fluid, or IB nodes and compute distances from the BCG to the appropriate nearest nodes. This approach, termed the Immersed Boundary Method (IBM) effectively allows one to use unstructured grids to impose BCs and simulate the flow with a fully structured solver. Easy generation of a structured mesh is combined with speed and ease of solving of the structured code.

The key benefits of the IBM are as follows:

- Relatively easy and quick grid generation, even for complex geometries. To generate a high quality surface grid, one doesn't need well-cleaned CAD geometry. CFD grid generation tools tend to require a “clean” and well defined CAD geom-

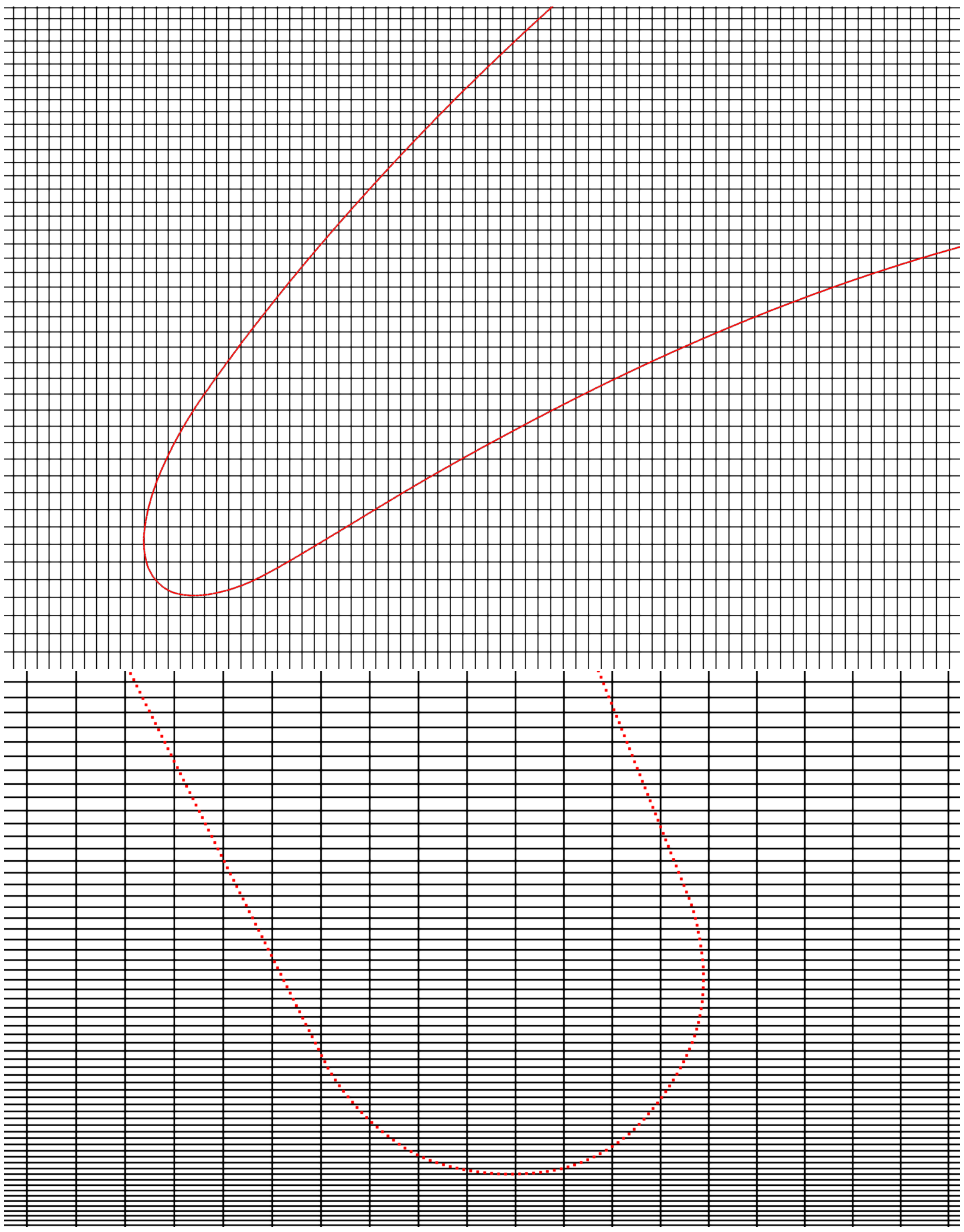

Figure 2.1: Grid of the T106A 2D turbine cascade representation.

etry without any, even smallest holes to create a suitable volume grid. It is often the case that a CFD analyst receives CAD geometry from designers and must invest significant time to ensure it is well defined in the CAD package. This is true for generating volume grid only; to generate a surface mesh in grid generation packages one can usually get away without a well defined geometry.

- Allows one to use a structured solver with even very complex geometries easily. This brings a number of benefits as structured solvers scale excellently on parallel platforms and are much more prone to sophisticated algorithms. For instance one may use a true high order ($>$2) accurate in space method on a structured grid more easily than on unstructured grids and there is no theoretical limit to order of accuracy. Also in general, structured grids usually have much lower node count for the same suitability and quality of grid compared to unstructured counterparts. This effect manifests itself the most in highly turbulent, hence nonlinear, flows.
- The N-S equations are exactly the same as without an IBM, with an extra term added only immediately near the boundaries. This means any existing resolving or modelling approach may be used without additional development time.
- Since the grid on which the numerical solution progresses is independent of the actual geometry being simulated, one may accommodate moving meshes relatively easily. Doing this would indeed require a pre-processing step each time the BCs imposing grid (surface) is changed, but for a structured grid this is significantly less effort than re-generating the entire volume grid at each step and that makes the IBM much more prone to automation.

The challenges of IBM are also summarised:

- The presently pencilled combination of tools and methods has has never been attempted with a high level parallelisation library, e.g. the Oxford Parallell library for Structured grid applications(OPS) used here. High level libraries such as OPS tend to be inflexible in how the data must be arranged and accessed in memory in order to use a library. This might prove to be a severe limitation when combining structured and unstructured datasets.
- Several additional interpolation procedures near the boundaries are needed as well as many algorithms to identify and mark Solid, Fluid and IB nodes are needed. All of these need to be thoroughly tested for accuracy and robustness.

- Ghost Cell Immersed Boundary Method (GCIBM) requires at least one node inside the solid domain to be present and solved on to correctly impose BCs on the surfaces. This requires an algorithm to interpolate between real, physical solution in the fluid domain and the imaginary Ghost Cell inside the solid domain. This also requires that one must invest additional effort to ensure the solution order of accuracy is as expected, in particular in the near-wall regions.
- Grid refinement must be done on a full structured grid as hanging nodes are not possible with the OPS library, often leading to excessive refinement away from the walls. Computational grid must also exist inside the solid domain, regardless whether it is actually used for physical computations or no. These issues can be addressed with approaches such as overset or specific LES models but introduce another layer of complexity and instabilities and are beyond scope of the present work.
- The method is in early development for high Re., turbulent flows. IBM was originally designed for low Re and flexible boundaries [32]. It has then been extended to rigid boundaries and high Re. Not many studies have been performed on IB with high Re and compressible flows and while results are promising so far [33], [34] this is an immature technology with many gaps in understanding.
- Stability of the solution is highly influenced by the normal vectors at Ghost Nodes. This appears to be a simple mathematical problem and accurate normals are not difficult to obtain however there are several peculiarities which can easily destabilise the solution leading to numerical divergence. This effect is likely to be even more pronounced in 3D [30].

Historically, the Immersed Boundary Method was first developed by Peskin [35] to simulate blood flow and heart mechanics. The geometry was changing during solution, Re was low (up to ˜5000) and the boundaries were elastic. The fluid used by Peskin, blood, is considered non-Newtonian. Since Peskin, many variants of IB were developed to adapt the basic principles to other applications, such as viscous unsteady flows or rigid non-elastic boundaries. Paper of Mittal and Iaccarino [36] is an excellent and often quoted review of the IB methods since they were first developed. Another, somewhat more up to date high level review of IB methods is by Gornak[37]. They both also provide a clear explanation of the principles and an outline of the procedures required.

The two papers review all the main variants of IB developed since Peskin. As the two papers really provide a comprehensive and detailed, yet easy to understand IBM background only a high level summary with reasoning for choices of specific IBM will be provided here.

2.2.2 Continous forcing vs. Discrete forcing

IB methods are usually catalogued on a high level as on Figure 2.2. The first level of categorisation is direct or continuous forcing. Continuous forcing means the extra Immersed Boundary “IB” terms are derived analytically and added to the main flow equations (or other equations being solved) before discretisation of the equations. Direct forcing means the terms are added numerically, after the equations have been discretised, on each node.

Continuous forcing, also sometimes referred to as Indirect Approach is independent of discretisation method and found to be good for elastic boundaries [38, 39]. However when modelling sharp rigid boundaries (such as walls of a Titanium alloy turbine blade) the method was found to be unstable near the wall and produce significant error [36, 40, 38]. This is mostly due to the forcing imposing the BC being spread over

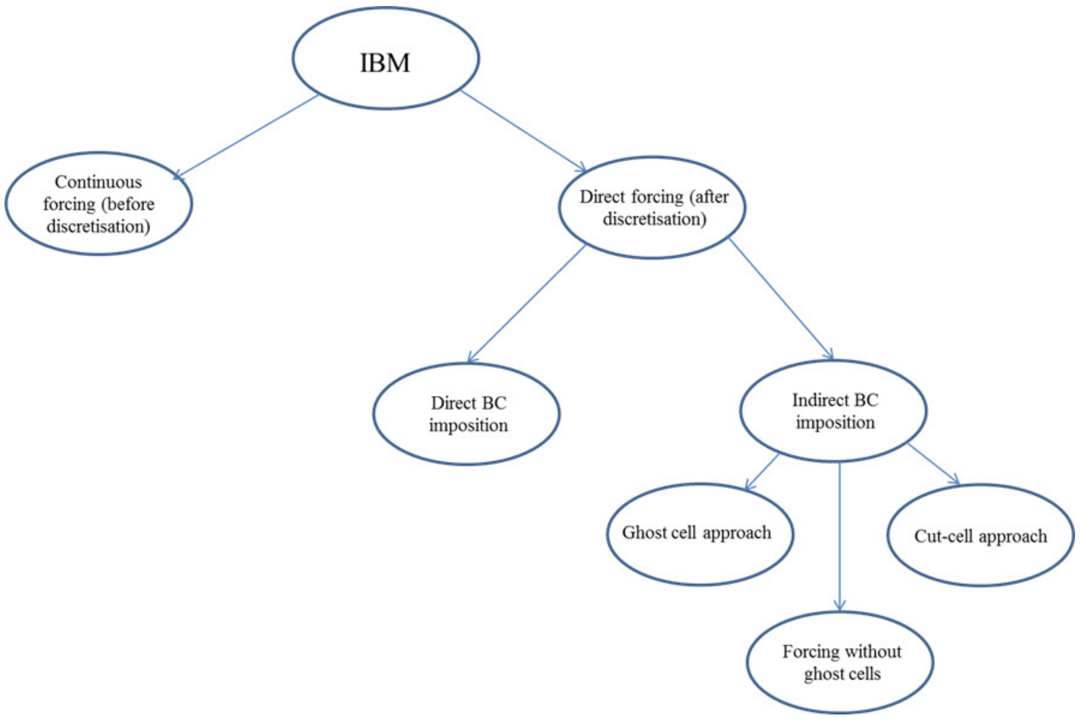
IBM
Continuous forcing (before discretisation)
Direct forcing (after discretisation)
Direct BC imposition
Indirect BC imposition
Ghost cell approach
Cut-cell approach
Forcing without ghost cells

Figure 2.2: High level categorisation of the Immersed Boundary techniques

several nodes near the boundary. It is not a problem for elastic, low Re boundaries but becomes more significant with increasing Re when accurate resolution of the boundary layer is critical. Another disadvantage of indirect approach is that the equations must be solved inside the solid as well as the fluid domain and there is no way to effectively disable the solid domain. The time step was also found to be relatively restricted to achieve numerical stability [41].

Direct forcing approaches can be divided into two main categories; direct and indirect BC imposition; while terminology is similar, the ideas however are very different when it comes to the way Boundary Conditions are imposed. Direct BC imposition, while applying forcing terms after discretisation still spreads application of the boundary over several nodes; a feature undesirable at higher Re. It also requires implicit set of equations to be solved at each step as exact forcing terms are not known; this can cause problems for parallelisation. The advantage here however is absence of any user-defined parameters and no stability constraints on the solution.

Indirect BC imposition on the other hand was designed from the beginning to handle high Re flows and is the most suitable for this purpose. There are three main categories of indirect BC imposition methods; Ghost Cell, forcing without ghost cells and Cut Cell approach. From the reviews it is clear the Cut Cell was designed for Finite Volume only [42]. It was also found this method is not easily extensible to 3D and actually creates unstructured datasets that will likely be highly problematic on GPU architectures. It also has several other problems associated with cutting cells, e.g. creating very small, sharp edged cells. A cut-cell method will not be of any use in the present work with finite difference approach.
There are also attempts to further customise the IBM coefficients such as having different interpolation coefficients in each direction direction (x, y, z) as in Gautier et.al. [43]. This is promising and can result in improved stability and accuracy of the IBM method but will likely require significant investment to mature the technique and make it robust for a larger range of cases.

2.2.3 Ghost Cell IBM

The IB method chosen for implementation in the present work is Direct Forcing with Indirect BC imposition with the Ghost Cell approach. The specific method, originally designed for the Finite Difference method by Tseng et.al. [44], was also implemented by various other researchers, but not in internal cooling or similar fields [45], also applying it to high Re flows over airfoils, e.g. NACA0012 [46] and other relatively simple geometries such a ribbed channels [47], however all at incompressible flow conditions. A variant of it was also attempted to address local grid refinement issue [33]. The reference method in the present work however is that of Ghias et.al. [31] and further improved by Nam et.al. [41], due to their use of compressible flows. In the ghost cell method the solution doesn't need to be solved inside the solid domain and only one or two (depending on the stencil length) layers of "Ghost Nodes" are required inside the solid domain. The solution is then interpolated appropriately to obtain values as if the boundary was present on the Cartesian grid.

Along with the Ghost Cell IBM is another method [40, 38, 45, 48, 49, 42, 50], which is similar but has some important differences. It has no formal name and the procedures are different to the Ghost Cell technique. The dominating motivation for development of this alternative was Conjugate Heat Transfer. The key difference between these methods is the layer of IB cells used to reconstruct the solution near the wall, hence impose Boundary Conditions. In the GCIBM the solution is reconstructed at the ghost cells, inside the solid domain, while in the reconstruction method one layer of nodes closest to the boundary in the fluid domain is used. While seemingly simple modification, this changes notably the numerical behaviour of the scheme near the wall and can likely affect near-wall accuracy at high-Re, which is precisely the topic of the present research.

The interpolation (no GCIBM) version of IBM shares the general procedure with several other researchers' work, e.g He and Tafti [18] or Nagendra and Tafti [42]. The method by Tafti is an extension of the IBM developed by Gilmanov [48] to use curvilinear generalised coordinates. One of the key benefits of this method is that it only requires modification of the solution in the immediate vicinity (up to 2 nodes near the boundaries) near the Immersed Boundary allowing any modelling or resolving to be used in the bulk of the domain (LES/RANS etc.). It has been done with Combined Heat Transfer (CHT) simulation.

The reason Ghost Cell method was chosen is compressibility and turbulence treatment. GCIBM was done and validated with compressible boundary conditions whilst

the interpolation method was not. It is likely possible to extend the interpolation method to compressible B.C.s however this is beyond the scope of present research. It was also found that turbulence is modelled best and easiest with the GCIBM as the interpolation method requires additional considerations for turbulent viscosity at the walls.
There are several key considerations for the IBM procedure to work.

- **A Search and locate algorithm choice.** Algorithm of Allevi and Bermejo [51] is recommended by most researchers with Indirect BC imposition and direct forcing. The algorithm relates an arbitrary lagrangian point to the background Cartesian grid.
- **Interpolation procedures.** An interpolation procedure is necessary to interpolate from the background Cartesian grid onto an arbitrary Lagrangian point in the vicinity of the Cartesian nodes. The two most quoted and recommended such algorithms are by Roman et.al. [52, 53] and Majumdar et.al. [54]. They also provide detailed discussions on the use and performance of the algorithms as well as review of others available.
- **Ghost nodes.** Different stencil lengths will require different number of layers of Ghost Cells. Stencil length depends mostly on the differencing scheme used and will likely be longer for higher order techniques. This issue can be eliminated by using backwards difference near the boundaries and still achieving the required accuracy.

2.2.4 Improved GCIBM

Improved Immersed Boundary Method (IIBM) was devised and published in early 2017 by Chi et.al. [30]. Their method notably improves the previously used compressible flow version of GCIBM [31, 41] in two ways:

- it involves using an Extra Image Point (EIP) further away in the fluid domain, along the normal from the Ghost Node to the Body Intercept, as shown on Figure 2.3.
- the base Image Point is moved further away into the fluid domain if it isn't fully enclosed by fluid cells (a check is performed for each cell and modification performed according to algorithm). This is critical for stability and that the problem is numerically well posed.

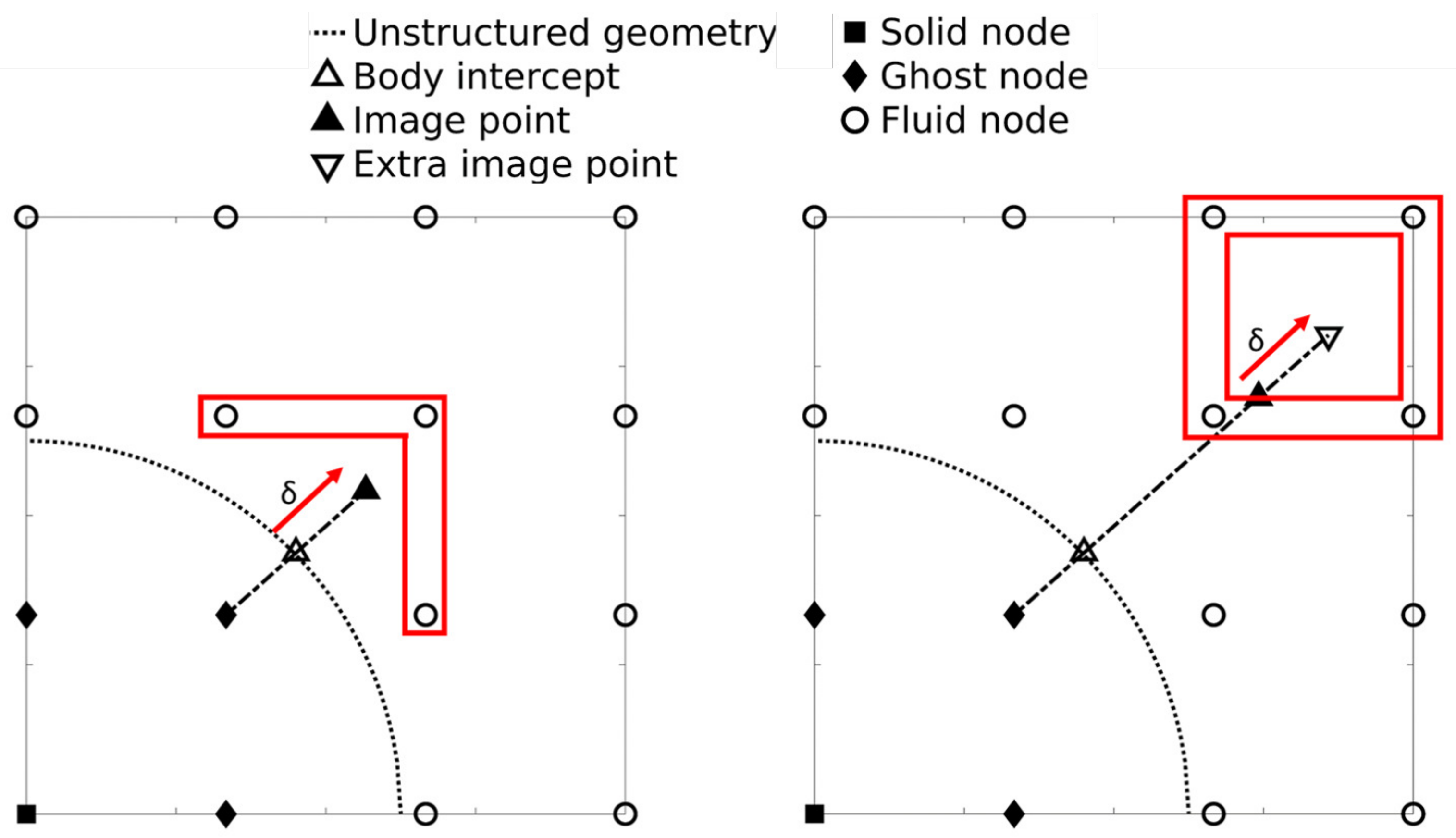

Figure 2.3: Schematic of the baseline Ghost Cell IBM versus Improved IBM. If the distance δ is smaller than a control threshold (constant), the IP and EIP are moved further away into the fluid domain. In both cases, distance δ between IP and EIP is preserved and equal to distance from BI to GN.

The two modifications make the IBM more stable and accurate for boundary definition. It also ensures the problem is always well posed. One extra model constant is introduced to control the distance of IPs from the boundary. Neumann boundaries are imposed easier via the use of EIP and cumbersome handling of incomplete interpolation is eliminated.

Regarding new and relevant insights about the previous IBM, PhD Thesis by Adam Preece [55] presents a GCIBM code. The code has very interesting and useful insights however uses an old version of the IBM and low Re (up to 100) is only studied. Tyagi et.al. [56] use IBM to simulate a multi-phase flow inside an impeller stirred tank, with LES. The impeller is simulated via moving grid and updated with IBM every timestep. Verzicco et.al. [57] studied very similar configuration, and again with $Re < 2000$. Yang et.al. [58] investigate and develop sophisticated interpolation and smoothing techniques for use with IBM that aim to reduce near-boundary oscillations and stabilise the solution in that region while improving accuracy. These stabilising functions are complex and will not be used in the present work before the baseline solver is complete and validated. Also, it appears Roman and Napoli et.al. [52] are the first to use an IBM formulation with implicit discretisation, albeit with indirect B.C. imposition.

It is worth noting that all the papers discussed in this section use an indirect B.C. imposition version of IBM where forcing terms are added to the main N-S equations. Publications using the GCIBM, where the original formulations are preserved and the IBM is suitable for high Re flows, are very scarce and this is also the motivation behind the current work.

2.2.5 Preprocessing algorithms necessary

An important practical consideration that emerged when developing the Immersed Boundary Method solver was an efficient algorithm to determine whether a given point lies inside or outside of an arbitrary two/three dimensional convex/concave shape. While a trivial task for humans, machines struggle to solve this problem efficiently. Additionally, an algorithm for computing distance and vector normal to the shape of said point is needed for IBM. This problem overwhelmingly becomes the bottleneck for simulations with moving geometry as the Point-In-Polyhedron (PIP) problem must be solved at the end of each time step. Although for stationary geometries the PIP must only be solved prior to the solution and never dominates the overall solution time, an algorithm is needed nevertheless.

The simplest solution is a family of build in MATLAB libraries of two dimensional point in polygon. There exist many implementations and the computation is straightforward. The difficulty with those libraries is that they are computationally inefficient and realistically only work for two dimensional geometries. Due to these limitations this is not a solution that could be deployed in an industrial or business context without significant development. It is however suitable for initial research and proof-of-concept stage.

The main approaches widely used in research are:

- Ray tracing
- Angle counting
- K-D tree (more involved and not very often used technique with IBM)
- 'Normals' technique
- Transformation to computational space (η, ζ, ξ)

An excellent source of the above geometry algorithms is book by O'Rourke [59]. A ray tracing algorithm with IBM has been implemented by Iaccarino et.al. who contributed significant amount to development of IBM techniques [60]. Tafti and Nagendra et.al. [42, 50] extensively use the search and locate algorithm developed by Allievi [51]. The problem with these approaches is that no standard computational library is available publicly. Gilmanov et.al. [48] use transformation to the computational space technique, but again it is custom developed and not available publicly. Roman and Napoli [52] and Kim et.al. [61] also use the ray tracing algorithm developed themselves and provide a good overview of the methods available.

There are several MATLAB [62] implementations of the ray tracing as well as angle counting algorithms as well as an efficient K-D tree technique implemented in the CGAL library [63]. While a custom written technique may be more efficient and integrated, a standard library is used at present.

2.2.6 Alternative non-body conformal techniques

The most common type of imposing Boundary Conditions at present is via a body conformal grid, where edges of the grid closely coincide with edges of the geometry being simulated. This way enforcing the boundary conditions is relatively straightforward as the algorithm simply imposes wall velocity (zero or non-zero) or any other required conditions at all extremes of the existing grid, which is physically meaningful for coinciding nodes. This however comes at a cost of time spent ensuring the grid curvatures and cell distributions are created appropriately and are of sufficient quality. As the geometries of turbine blades' internal cooling become more complex and flow more difficult to simulate (hence requiring high quality grid), the task of creating a mesh can easily become so overwhelming it completely dominates the entire simulation process, as will be shown in Chapter 5. It is also often not possible at all to create fully structured grids for some geometries in design timescales, which limits the numerical methods and often results in greatly elevated cell count to achieve the necessary accuracy.

An alternative to body conformal grids are non-body conformal grids where geometry does not coincide with edges of the grid and boundary conditions are imposed via a complex numerical or mathematical (depending on the method) manipulation of the underlying equations being solved. This spawns an entire cache of issues while developing the code but the task is done only once during development and Verification and Validation (V&V) and benefits can be exploited for decades over the lifespan of a CFD code. From a business investment point of view such 'front loading' of simulations cost makes sense and can save alot of money in the long run.

Among the more popular non-body conformal techniques are flux reconstruction, overset grids, cartesian cut cell or discontinuous galerkin. There are also particle based methods such as SPH which do not use a grid at all. Some of them such as Smooth Particle Hydrodynamics (SPH) are notoriously difficult to parallelise on General Purpose Graphics Processing Units (GPGPU) platforms, while others such as Flux Reconstruction (FR) provide added benefits such as easy ability to implement higher order schemes.

2.2.7 Conclusions

While it appears several methods might be potentially be suitable there are usually much higher chances of success if all attention is diverted to one method. There also does not appear to be a CFD code in existence that combines all the computational and scientific benefits into one. There are no universal 'silver bullet' solutions in the CFD world and it is the engineering judgment of the author at a time that Immersed Boundary Method will yield the best compromise between all the requirements of the field of turbine blade internal cooling. Another judgment the author has made at this stage is that even a moderate success in combining the set of chosen techniques together should provide a good contribution to industrial, engineering and scientific communities as an actionable intelligence to guide further research investments. For this reason a higher level of risk is acceptable. However the author acknowledges that as both algorithms and hardware evolve and mature a different combination of methods may become the most optimal in future.

2.3 Parallelisation & computational approaches

2.3.1 Introduction

Until about 2005 computational power was increased mostly by increasing CPU clock frequency [64, 65, 28]. This gave rise to almost free performance increase for developers and did not require the CFD programmer to consider any parallelisation strategies as the code could be fully built on a new CPU without any modifications. Since about 2005 however the CPU clock frequency has stagnated as further increases are unsustainable due to non-linear relationship between CPU clock frequency and power requirements. The HPC stakeholders in fact often prefer to have more CPUs with lower clock frequency as the power cost for a CPU over its lifespan is comparable with its purchase [64] and similar performance can be achieved with lower clock frequency for lower cost this way. This has created industry-wide shift from single core processing to initially multi (up to 10), then many (hundred thousands) and eventually exascale (milions of cores) core processing in CFD simulations [66, 67]. Other hardware vendors also responded with a huge variety of heterogeneous hardware which all can be used to accelerate a CFD code [68]. Examples are General Purpose Graphics Processing Units (GPGPUs) or Intel XeonPhis.

There are many different variants of each hardware, for instance Fermi or Kepler architectures offered by NVIDIA or different AMD boards. Each requires different low-level programming and often significant performance can be gained if a code is restructured or even a numerical method changed to be suitable on a particular hardware.

Several programming models gained a widespread acceptance over the past 10 years; by far the most popular on HPC systems today is Message Passing Interface (MPI) due to its versatility and broadness of use. It can be used on both shared and distributed memory machines. There are general frameworks such as OpenMP, OpenCL or vendor specific such as CUDA. It is also very likely that new coding models, or hardware will emerge in future that will require a code to be substantially changed. It is not certain which of these, or even what combination of these will give the best performance or even exist in practice in years to come.

Once a parallelisation strategy has been chosen many years of development are required to get the code to work reliably in industrial settings, validate and maintain it. Once in use, lifespan of CFD codes is of the order of tens of years and significant resources are committed to each CFD code in the long run. Hence CFD developers are now faced with a question which of the parallelisation strategies to adopt. Realistic answer is all of them and neither at the same time. Ideally one doesn't want to be limited to a particular hardware, but also doesn't want to spend significant resources porting code to another parallel framework. There is also the issue of performance; much expertise is required to create a parallel CFD code that performs best for a given application.

2.3.2 CPU vs. GPU

As a final note, CPU vs. GPU debate will be mentioned. Among many others, Lee et.al. argue [69] that developers of future codes will not be choosing between CPU or GPU, but combining them together to maximise codes' performance. GPUs perform significantly (orders of magnitude) better for raw calculation than CPUs but are less flexible in terms of what they can do. In addition GPUs provide significantly reduced power cost of simulation per flop [70] - something that will be ever more important as global focus shifts to sustainability.

2.3.3 High level libraries

In the field of CFD recently multiple researchers have independently concluded that a high-level abstraction is recommended and necessary [71, 72, 28] for the CFD codes to be able to keep up to date with rapidly evolving hardware and programming models. High level abstractions allow the code to be written once, only in terms of the abstraction and compiled on multiple architectures using a source-to-source translation. The key benefits to such approach are that the CFD developer doesn't need the extensive expertise in parallel framework such as MPI. Instead the high level abstractions are designed to be simple and allow the developer to focus on the physics and the application. The code is also written only once and support for new programming models, e.g. CUDA, can be easily added as they emerge in the future. This is opposed to completely rewriting a code for new model. Due to these factors, the performance can be maximised easier as the back-end code is written by experts in this particular framework, e.g. MPI and the code is said to be 'performance-portable.

This sounds like an excellent way forward for CFD developers however such high-level abstractions only exist for specific applications, for example OP2 for unstructured codes or OPS for structured multiblock. Other abstractions such as MAGMA or BLAS only provide a set of building blocks [64] which restricts their usability and actually lowers the level of abstraction, requiring the developer to have more expertise to use them correctly.

The code developed in the present work uses the OPS high level library for both performance and portability between architectures. The starting point for current development, DOLPHIN, is a multiblock structured grid application and as such is a perfect candidate for OPS. It has been demonstrated that both OPS and OP2 give the same or sometimes better performance as hand-coded parallel implementations without an abstraction library. The applications on which it was demonstrated are small 2D app CloverLeaf and full 3D inhouse production code HYDRA, used by Rolls-Royce [73]. All the low-level intricacies of a parallel implementation are handled by OPS and as such even user with minimal knowledge of e.g. MPI is able to have maximum performance easily and quickly. A disadvantage of using the OPS is its immaturity. The framework has only been published in 2014 and has only few users applying it to real codes [74]. The documentation is developing, there are bugs in the code and some important features are to be implemented. These disadvantages however are expected to diminish with time and the present work also contributes to OPS development by testing its capabilities and reporting issues to OPS developers. Ultimately developing a CFD code is a very long term strategy and the shortcomings of the OPS library are all judged to be resolvable without major infrastructure difficulties.

2.4 Internal cooling in turbomachinery

Much research has been carried to improve turbine blade cooling in the past 20 years, both experimental and numerical. The aerospace industry went from empirically driven designs with occasional simulations 30 years ago to complex CFD-led design with minimal experimental testing now [19] and this is also partly reflected in turbine blade cooling technologies. However at the moment primarily RANS models are used to design the internal channels and other cooling systems for turbines [21, 75]. Mostly empirically established 1D corrections are used [76, 77] in design process of turbine blades to get the necessary accuracy. It is argued by Tucker and Tyacke [78, 79] variance up to 100% in predicting the heat transfer can be encountered between different RANS approaches in the present methods. In the application where even small variations of temperature (2%) give significantly different life of components such error is unacceptable for design. There are very few reliable numerical studies on engine representative geometries [75] and these are usually guarded as a commercial advantage. Additionally due to speeds and temperatures involved high quality experimental data is very hard to obtain and it is not uncommon for measurements to have $>$ 10% uncertainty. Moving to LES or resolving methods will be especially important for cooled turbine blades [19, 78] due to the flow being dominated by larger scale structures and highly transient in nature. Resolving methods however are more computationally intensive and this is the primary reason they haven't been adopted as standard in the design process. Increasing computational power allows CFD users to perform simulations that haven't been possible before, or allows to use more CPU-demanding calculations to be used routinely in design [80, 64]. In the field of internal cooling passages, both apply. In the next 10 years one will likely be able to simulate a full turbine blade with all the details of a cooling system, a calculation with amount of nodes of the order of several billion cells; these will likely be occasional simulations, but frequent enough to be useful for designers. On the other hand smaller simulations, of the order of ~100m cells will be possible in short, design timeframes, contributing directly to significant improvements of engine efficiency. There will also likely be a shift from RANS to resolving methods in internal cooling passages design.

Previous studies in the field deal mainly with simplified geometries and/or body fitted grids. However as shown on Figure 2.4 the geometry is rather complex and consists

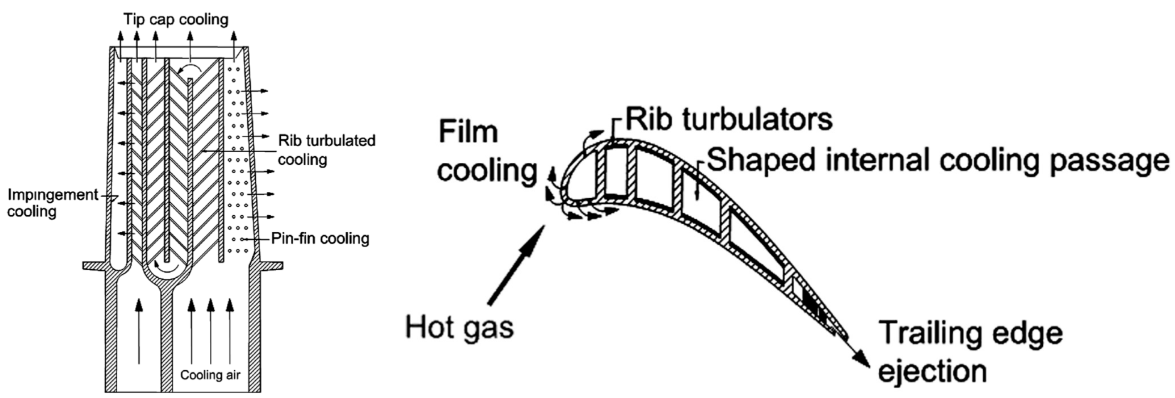

Figure 2.4: Schematic of a typical HP turbine blade

of several different features designed to increase cooling. The main features are: pin fin cooling, ribs turbulators, impingement cooling, tip cap cooling, film cooling and trailing edge ejection. All these features are important, likely interact with each other and the blade is rotating in operation which adds another layer of complexity. It would be of great value to turbomachinery designers to simulate a complete system with all the features.

The discussion of experimental and numerical work on turbine blade internal cooling starts from a historical perspective and will move on to more modern cases. One of the earliest numerical studies on rotating turbine internal ducts is performed by Majumdar in 1977 [81] and then series of smooth and rotating rectangular ducts, computations with RANS $k-\varepsilon$ model by Iacovides and Launder as early as in 1991 [82]. Similar work to Majumdar et.al. was performed later by Dutta in 1996 [83]. The researchers investigated smooth ducts and mainly the effects of rotation were looked at. These are probably some of the earliest studies which compares experimental data with RANS calculations which were relatively rare for a 3D geometry in that time. Additional terms in $k-\varepsilon$ equation are introduced based on experiments to obtain "satisfactory predictions" of Nusselt numbers. Using the same geometry multiple experimental studies using different techniques are done by Han et.al. [84]. Up to approximately 2000 the geometries and flow investigated numerically were mostly either two dimensional or very simple, boundary conditions were constant and no ribs were introduced. Realistic engine representative internal cooling passages were investigated only experimentally and only experimental papers exist from that period, such as by Han et.al.[85, 86]. The studies investigate effects of ribs non-orthogonal to the flow, and of different aspect ratio, all at high Reynolds numbers (>500,000) as well as early studies investigating passages inside the blade leading edge such as that by Taslim et.al. [87]. Other studies attempting to optimise the heat transfer via a ribbed triangular channel were done experimentally by Zhang et.al.[88] in 1994. Ralabandi et.al. [89] provide a more detailed review of general experimental and numerical work done prior to 2009.

More current work on turbine blade cooling technologies, numerical as well as experimental are the review paper by Iacovides and Launder [90], experimental papers by Schuler et.al. [91], Lee et.al. [92], Ralabandi et.al. [89] and numerical work by Tafti et.al. [18].

Moving on to the use of Immersed Boundary Method (IBM) to investigate turbine cooling the only known IBM approach on such geometries is done by Tafti and He [18]. In terms of latest experimental data the EU project ERICKA [93], much data has been produced related specifically to internal cooling for realistic geometries and flows with many major companies in the sector involved. Data is numerical, but largely also experimental at real conditions as the aim was to serve as a base for further CFD development. The data appears to be guarded as commercial advantage.

2.4.1 Heat Transfer

The primary purpose for which CFD is used in internal cooling passages design is to predict metal temperatures and decide on geometrical features of the blade to be manufactured. This requires accurate prediction of heat transfer from the metal to the fluid. Nusselt number is defined as the ratio of convective to conductive heat transfer and is the one of the main practical measures of effectiveness of fluid cooling when designing cooling passages. The turbulators and ribs are designed to promote mixing and maximise convective (forced convection) heat transfer in the fluid. Hence it is important not only to predict the flow accurately, but also the heat transferred to the fluid. Ultimately, Conjugate Heat Transfer simulations provide the complete transient thermal analysis directly useful in the design process. It is the objective of the present work to investigate, if time permits, Conjugate Heat Transfer simulations with IB method after the IBM is validated. However the primary focus will be on Heat Transfer without inclusion of inside of the metal. Four key papers dealing with heat transfer augmentation in internal cooling passages were identified. They provide reviews of the field and a detailed discussion of the main two CHT strategies widely used.

Amaral et.al. [76] provide a good review on CHT application in internal cooling as well as review of different CHT computational strategies. Bell et.al. [77] provide similar discussion with practical, design oriented discussion with description of the 1D empirical correlations presently used by industrial designers. Ligrani et.al. [94] and Gupta et.al. [95] provide more general discussion and review of heat transfer augmentation techniques. Although general it is still specific to turbine internal cooling and is an excellent overview of the specific field and work done in the past.

2.5 High order schemes

Through detailed and broad studies in both academia and industry there is a consistently established consensus that a numerical scheme's order is considered to be 'high' if it is proven to be 3 or higher [96, 8, 97] via specific and targeted numerical tests. This consensus has been reached in the wider CFD community without doubt. The first question one may ask here is why bother with higher order schemes in CFD at all.

Higher order schemes offer lower computational cost of achieving the same accuracy [96, 97, 98]. While more computation is required per grid point, fewer nodes are overall required as each cell is able to simulate non-linearities more accurately with increasing scheme order. It was found the node count savings are up to 50% larger than extra computations of solving equations to achieve the same accuracy. This will become increasingly important with transition to exascale computing where communication between the partitions is a critical limiting factor. Higher order schemes also tend to have significantly reduced diffusivity than their 1st or 2nd counterparts. This is necessary for some applications such as flow over a helicopter [96] or noise prediction in the field of Computational Aero Acoustics (CAA) [99, 100]. First and second order methods can strongly dissipate unsteady vortices and one needs relatively refined mesh to have an accurate simulation with lower order methods in certain specific applications [101].

There are however limitations and disadvantages to higher order schemes, notably they are only really possible on structured grids. Methods such as Flux Reconstruction can be used however they introduce an entirely different field of challenges and requirements. It was proved by Leonard [102, 103] that an unstructured FV scheme does not achieve “true” high order accuracy even using all the interpolators of higher order. High order methods provide the most benefit with scale resolving techniques such as LES, DES or LES-RANS. While not entirely fruitful there is significantly less benefit in applying high order schemes to steady RANS computations. Moreover, they can be more numerically unstable than lower order, require extra attention to achieve stability and are often more difficult to implement than lower order methods.

The present work on internal cooling provides a good ground for higher order methods as the code used uses Finite Difference schemes on strictly structured grids with the aim of performing Large Eddy Simulations. It has also been established flows present in internal cooling passages of turbines would benefit greatly from the use of higher order methods due to their decreased dissipative properties [104]. Given that massive (of the order of 1b nodes) simulations are expected to be performed regularly in this area even small percentage performance increase will give substantial benefits. In addition, it is almost a certainty that industrial CFD practitioners will have routine access to exa scale computing facilities in the next decade.

2.5.1 Compact high order

A significant subset of high order schemes are compact high order schemes. They reduce dispersion and anisotropy errors introduced by the Boussinesq approximation due to their spectral-like resolution [105, 106, 107]. They also require fewer halo points in the vicinity of the current node to perform computation to a given order of accuracy, reducing communication for parallel calculations. The latter point becomes increasingly important as more computational units are used that communicate with each other (or ’ranks’ in MPI speak). It is already becoming a ’common knowledge’ among HPC practitioners that it is the communication that is the costly element and computation can even be considered ’free’ by comparison.

The major drawback of compact higher order methods is they tend to require simultaneous systems of equations to be solved as will be explained in Section 2.7.2 in more detail. In effect more computation, but less communication between threads is needed and that shifts the CFD code to more compute-bound than communication-bound. However in practice this requires solving a tri-diagonally dominant matrix via the usual methods such as inversion or iterative solves. For instance in principle a

PETSC [108] library could be used for this purpose. PETSC has powerful and scalable iterative matrix solving routines but is judged too complex for a simple Tri-Diagonal matrix present here. It is certain that only TriDiagonal matrices will be present and there is no risk of utilizing a simpler solution with less overhead in terms of both maintenance and implementation. Several standard simpler TDMA algorithms exist, for example the Thomas algorithm [109] however they are inherently sequential in operation, as the value at any node depends on the calculation outcome from all the previous nodes, i.e. there is order-dependency leading to race conditions in parallel implementations. This can be somewhat parallelised by a directional split but would be difficult and far from general, likely resulting in poor scalability and long term suitability. A TriDiagonal Matrix Algorithm (TDMA) by Laszlo [110] was chosen to be used here due to its parallel-first design and the method specifically targetting Tri-Diagonal matrix solve. Even though the OPS library was not designed to work directly with order-dependent schemes such as TDMA, external functions can be integrated to do the implicit computation as will be explained in more detail in the parallelisation chapter. All of this unfortunately adds complexity (hence cost) to the overall implementation however is judged to be highly beneficial as it enables a range of compact high order schemes to be used efficiently and in a scalable manner.

2.6 Turbulence modelling

Commonly established CFD methods in industry often fail to predict strongly unsteady flow with sufficient accuracy (such as RANS) or are too expensive to use routinely throughout the design process (LES). Scale Resolving Simulations (SRS) such as Large Eddy Simulation (LES) or Detached Eddy Simulation (DES) or its variants such as Improved Delayed Detached Eddy Simulation (IDDES) are only few examples of the expensive tools. In response to this hybrid Reynolds Averaged Navier Stokes LES (RANS-LES) methods have emerged as a balance of cost and accuracy but are still largely in experimental phases [111, 112, 113]. Other SRS approaches include Scale adaptive Simulation - Shear Stress Transfer (SAS-SST) [114, 115, 116], which can be considered a generalisation of DES, or explicit hybrid RANS-LES by Davidson et.al. [117, 118] which offers very fine grained control over which areas are designated as 'resolved'. As mentioned before, it has been consistently established that internal cooling passages will benefit greatly from the use of scale resolving approaches such as LES or even the hybrid methods. SAS-SST for instance promises to relax significantly the strict near-wall grid requirements imposed by LES in a manner similar to DES, but also generalise that concept to the entire fluid domain further reducing cost while still providing improvement in accuracy vs. URANS. While there are various other turbulence approaches such as $k-\sqrt{kl}$, the hybrid RANS-LES techniques remain the primary choice of most researchers due to their performance.

It is an objective of the current work to demonstrate which turbulent models offer the best compromises in the framework of Immersed Boundary Method with high order discretisation. DES, SAS, LES all with different modelling strategies will be tested.

On a final note, it is worth pointing out that majority of CFD performed in industrial design environment, including all simulations in the present work is based on the Navier-Stokes equations. This has become the most common way of simulating fluid flows. There are however other possibilities to obtain a fluid flow solution, for instance the Lattice Boltzmann equations. A good comparison of the two can be found in Elhadidi and Khalifa [119] or Marie et.al. [120]. The Lattice Boltzmann Method (LBM) is used widely in the automotive industry (e.g. ExaCorp [121]). It is best suited for bluff body flows, incompressible, non-newtonian flows or multiphase simulations as the idea behind the LBM is statistical averaging of points over time. The LBM algorithms are simpler relative to NS and give accurate predictions of incompressible or moderately compressible flows. This technique was also applied to the field of Computational Aero Acoustics [120]. In addition LBM is perfectly suited for parallelisation including GPU cores and it was found LBM works well with IBM [122] and can be implemented relatively easily. However, in the present application the flow is strongly compressible in regions of interest [18] and NS equations are more precise being a continuum. It is also difficult to obtain an averaged solution such as RANS with LBM as it is inherently an unsteady technique. The main argument against using LBM for the present purpose however is the fact LBM does not represent the energy equation well, especially at high Re. Since heat transfer is critical in turbine blade internal cooling LBM is not recommended. Using LBM technique would also add another layer of complexity to an already complex set of requirements for a CFD code; NS equations are better understood and documented for the present application. It is also worth noting the N-S equations can be recovered from the more general LBM equations, given certain assumptions [123].

2.6.1 Nearest Wall distance

The very first equation that was solved with the present solver on a GPU was the Poisson equation [4]. It was used to compute distance to the nearest wall, quantity necessary for majority of turbulence models. There are several methods to compute the wall distance [124], Poisson [4] and Hamilton-Jacobi [125] methods being the most popular, however the Poisson method is the most simple to implement. It was found that the present solver does not require any smoothers or artificial stabilisers to solve this equation without any oscillations.

2.7 Numerical methods

2.7.1 High order schemes

The use of high order spatial discretisation techniques with Immersed Boundary was investigated as they seem to offer great benefits in the present application. Parnaudeau et.al. [126, 5] use 6th order discretisation with an indirect B.C. imposition version of the IBM. Although this IBM is not suited for high Re. flows, it is demonstrated on flow over cylinder that high order methods with IBM are possible and have the expected benefits of a non-IBM high-order approach. Laizet and Li [127] demonstrated via a DNS code that although formally IBM can only be made 2nd order near the wall, using higher order schemes overall does make the solution more accurate. To achieve higher order of accuracy near the walls the IBM technique itself would need to be modified, which would also likely lead to more instabilities and need for more complex filtering schemes at the wall. This is of course a possible and important improvement to the method but creating a robust baseline solver that works must be a priority.
It must be mentioned that although simulations with Incompact3d code are of 6th order and use compact stencils, they have maximum Re of 22,000 [128, 129, 130, 131]. This is moderately high, but not high enough as needed for internal cooling simulations, where Reynolds number an order of magnitude higher is expected. This is in line with the fact that indirect BC imposition method of IB, which incompact3d uses, is best

suited for low to medium Re range. In addition, Flageul et.al. [129] performed IBM simulation including Conjugate Heat Transfer through the Immersed Boundary wall. Another investigation of high order techniques with GCIBM is made by Xia, Luo et.al. [132, 133], although again with Re up to 200. It is proven that even for moving geometry, multiphase flow and heat transfer calculations the high order techniques with IBM exhibit excellent potential and do indeed work as expected, even with wall formally updated only to 2nd order accuracy.

IBM schemes' accuracy order of the solution can be increased in the bulk of the domain by modifying the primary spatial discretisation method (e.g. going from 2nd order to 4th order central difference) or in the near-wall IBM interpolation itself where the Boundary Conditions are imposed via GNs. Majumdar et.al [54] and Roman et.al. [52] tested higher interpolation schemes for the Immersed Boundary and found that they do not appear to be making any significant, or often even noticeable difference. Tseng et.al. [44] also came to conclusion that higher order interpolation technique for Image Points does not improve overall solution accuracy but can make the solution less stable, as the IBM even with 2nd order interpolators already exhibits some amount of ever-present oscillations.

The higher order boundary interpolations have not been tested on the significantly modified version of Improved IBM by Chi et.al. [30] with EIPs and the conclusions about accuracy may not hold. However it is concluded from the present review that increasing order of discretisation in the bulk domain will most likely yield the greatest accuracy and efficiency benefits.

2.7.2 Implicit discretisation and compact schemes

It is well established in both industry and academia that implicit discretisation is more stable as well as computationally efficient [96, 134, 107, 135]. A discretisation, whether in time or space domain, is usually considered 'implicit' when advancement or updating of variables is done via solving a set of simultaneous equations. In implicit discretisation there is no direct formula to compute for updated variables and a set of simultaneous equations must be solved for the current time step. This is in contrast to 'explicit' methods where updated variable is directly obtained from existing one via mathematical operations with known coefficients. The updating process can be complex such as in

RK4 scheme but the variable is always obtainable directly without a need for additional equation solvers.

For time discretisation, implicit techniques usually allow significantly larger time step and increased stability of solution. In theory the stability is unconditional but in practice there is an upper limit of timestep at which the solution becomes unstable and leads to divergence. That limit however is often up to two orders of magnitude higher than explicit time discretisation. For that reason even extra computations resulting from solving a set of equations quickly yield computational savings over time.

Interestingly a similar matter arises with spatial discretisation. Explicit 4th order differencing uses stencil spanning 2 nodes (hence 8 total in 2D and 12 in 3D) in each direction while implicit methods are able to achieve the same resolution with stencil only 1 node long (so 4 in 2D and 6 in 3D). Such implicit spacial discretisation methods are termed 'compact' schemes and while they usually provide increased stability and decreased dissipation the main benefit computationally is significantly reduced communication 'halo'. By using half the amount of neighbouring nodes the inter-thread communication is reduced in favour of more computation (solving a set of equations) which almost always has the effect of improved scalability. Majority of commercial CFD softwares offer implicit spatial and temporal schemes and while they are not always suitable they tend to be the 'workhorse' in the field.

The main difficulty of implicit schemes lies in implementation cost and limitation they place on parallelisation. Implicit schemes tend to require a more complex set of equation to be implemented than explicit schemes and need a specialised equation solver not normally present in CFD codes. Such solvers, or matrix inversion schemes must be interwoven and communicate with a primary parallel numerical solver and that can create significant bottlenecks in scalability. This matter will be explored further throughout the thesis.

2.8 Existing relevant CFD codes

Embarking on a development of new CFD code it is necessary to review the existing programs; this is to avoid duplication but also draw knowledge from them and even create opportunities for collaboration. The review begins by comprehensive list of requirements that became clearer after carrying out the literature survey:

- use of a high level library for performance portability
- ability to run on multiple hardware efficiently, e.g. CPU, GPU and adapt to future hardwares without major code rewrite. Emphasis is placed on ability to run on cost efficient GPUs
- ability to solve tri diagonal systems of equations for implicit time stepping and high order compact methods
- ability to handle generalised geometries without significant time spent on meshing
- ability to accurately simulate complex, compressible, high Re flows with rotation
- ability to accurately simulate fluid heat transfer with potential to extend to conjugate heat transfer simulations

A list of relevant codes is compiled:

- incompact3d [127]
- GenIDLEST [136]
- ASL is interesting, though last update was in 2015: http://asl.org.il/
- LBM open source code Palabos: http://www.palabos.org/
- Adam Preece 2008 Thesis IBM code: [55]
- Kang 2008 Thesis has IBM code: [137]
- Ikram thesis 2011 ibm code: [138]
- Flux reconstruction based code [139]; https://hifiles.stanford.edu/
- IMPACT: http://www.ifd.mavt.ethz.ch/research/group-kleiser/impact.html, massively parallel, high order, high Re, compact FD
- Witherden Thesis [140], PyFR, massively parallel, GPU, FR code

Not all of the softwares will be commented on or reviewed in detail here as that task alone could easily become a major chapter. Most of the above have features that satisfy majority of requirements here, there appear to be the most relevant due to their unique choices of numerical and computational methods that satisfy all but one or two requirements listed previously:

- Incompact3d
- GenIDLEST

- HifiLES

Incompact3d appears to be highly scalable and the developers proved it scales well for 260,000 cores [127]. It can perform implicit operations with 6th order compact schemes for space, RK explicit and AB 2nd for time, code is incompressible. It is also using a high level library similar to OPS, but only MPI is supported, no CUDA etc. Code is written in fortran, used primarily for DNS/LES at Re up to 20,000. IBM variant used is with direct forcing, but indirect BC imposition (i.e. adding a forcing term to momentum equations and switching it on/off inside/outside of the solid body). This indirect BC imposition method is described in all other IB papers as inaccurate at high Re due to local error near the wall spreading in to solution too much. The authors of incompact3d mention nothing about this. It may not be a big problem for DNS/LES where grid near the wall is refined significantly. Present work is using Ghost Cell IBM, compressible with a high level portable library that supports multiple platforms including GPUs. Targeting higher Re (internal cooling application has Re in the range 20,000 – 500,000)

GenIDLEST: Incompressible, parallel with MPI. 2nd order. Immersed Boundary with very similar forcing to Ghost Node. Applied the IBM to internal cooling, simple geometries and promising for more complex ones. No high-level library is used and the code is a hand crafted MPI application.
HifiLES: Although this code uses the Flux Reconstruction (FR) method and is aimed primarily at LES for aeroacoustics it is probably the closest to meeting the current requirements that the author was able to find. It is developed for use with high order methods and use with GPU architectures and is capable of compressible flow simulations. Due to the use of FR method, unstructured grids can be used for high order computations but it is not clear how the efficiency is achieved on GPUs with the datasets that result in such approach. Primary drawback of this code is lack of any implicit computations; focusing primarily on LES this is not a major issue but it might become a significant problem if RANS or hybrid RANS-LES is performed.

In summary while all the existing codes appear to be suitable for the present application, each lacks at least one critical component necessary (e.g. compressibility, high Re, high level library for portability or sharp IBM boundaries). The author is now satisfied that the current work will not be a duplication but an extension of the existing academic software portfolio.

2.9 Chapter conclusions

While reviewing literature is not a task that can ever be truly ’complete’, broad survey of existing works has been carried out covering both academic and commercial space. Requirements for an ’ideal’ CFD code for simulating flows related to turbine blades were iterated and clarified. A set of solutions is proposed and its benefits and drawbacks are understood. It is also clear there are likely other combinations of techniques that could satisfy the requirements and as technologies progress there will probably be even more. This is unavoidable however to make any progress at all one must set sails on a single set of techniques and focus on their implementation.

Chapter 3

3 Methods & Theory

3.1 Chapter introduction

This chapter presents details of all the numerical and computational approaches used in the present work. The methods cover development of the GPU IBM solver as well as testing of the hybrid RANS-LES. Governing equations, discretisation as well as parallel approach will be presented where possible in long-handed, fully expanded version as they were actually implemented in the solver. An attempt will also be made to list all the principles and assumptions with the aim of providing a solid reference for clarity and future works.

The chapter is arranged into three subsections; first are the core flow equations, discretisations and procedures necessary for physical simulation of a flow using Navier-Stokes and their parallelisation. Second subsection contains details of the modifications needed to use the IBM method and third presents the commercial codes (Fluent and CFX) approach used in some of the simulations for turbulence models testing.

3.2 Governing Equations: GPU IBM code

3.2.1 List of physical assumptions

1. Newtonian fluid, with Stokes assumptions:
    - The stress tensor is a linear function of the strain tensor.
    - The fluid is isotropic.
    - for a fluid at rest, $\vec{\nabla} \bullet \vec{\tau}$ must be zero
    - τ is proportional to $\frac{du}{dy}$
2. Stokes Hypothesis
   $\lambda = -\frac{2}{3}\mu$
3. Boussinesq approximation for compressible Favre Averaged Navier Stokes (FANS) equations

4. Single phase
5. Perfect gas
6. Not reacting chemically
7. Thermally and calorifically perfect gas
8. Sutherland law for viscosity(temperature) dependence
   $\mu = \mu_0 T^{\frac{3}{2}} \frac{1+\frac{110.4}{T_{ref}}}{T+\frac{110.4}{T_{ref}}}$

Where $\vec{\tau}$ is shear stress of the fluid, λ is kinematic viscosity, μ is dynamic viscosity and T is total temperature.

Boussinesq approximation is a widely used assumption that results in isotropic turbulence formulation and constant of proportionality μ_T. No anisotropic turbulence model such as the Reynolds Stress Model (RSM) will be used in the present work as the aim is to resolve majority of the turbulent energy by the grid directly and turbulence models are only used at subgrid.

Although the present code uses the full N-S formulation without the thin-layer approximation, it is worth pointing out that this simplification is best avoided with IBM techniques. The thin-layer approximation that eliminates several terms and is commonly used in CFD codes does not hold if using non-body fitted grid [141] and may reduce accuracy of the near-wall solution as non-body fitted grids are the workhorse of the IBM. It could be a potentially significant source of error in IBM but also a notable computational cost saving, however more investigation is needed before using that assumption with IBM. At present no papers exist that examine the issue of the thin layer approximation with IBM and to what degree does it affect the solution. It should be noted at this point that in the field the IBM is known for its near-wall inaccuracies in comparison with body-fitted grids and investigation into thin-layer approximation with IBM could prove a valuable future research.

3.2.2 Governing flow equations

Starting from a common vector form and moving on to the most expanded forms are focused on as they are most useful for practical implementation.

Continuity:

$$\frac{\partial \rho}{\partial t} + \vec{\nabla}(\rho \vec{V}) = 0 \tag{1}$$

$$\boxed{\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} + \frac{\partial \rho w}{\partial z} = 0} \tag{2}$$

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_j}{\partial x_j} = 0 \tag{3}$$

Where $\partial \rho$ is density and $\vec{V}$ is the velocity vector.

Momentum:

$$\frac{\partial \rho u}{\partial t} + \vec{\nabla}(\rho u \vec{V}) = -\frac{\partial p}{\partial x} + \frac{\tau_{xx}}{\partial x} + \frac{\tau_{yx}}{\partial x} + \frac{\tau_{zx}}{\partial x} + \rho f_x \tag{4}$$

$$\frac{\partial \rho v}{\partial t} + \vec{\nabla}(\rho v \vec{V}) = -\frac{\partial p}{\partial y} + \frac{\tau_{xy}}{\partial y} + \frac{\tau_{yy}}{\partial y} + \frac{\tau_{zy}}{\partial y} + \rho f_y \tag{5}$$

$$\frac{\partial \rho w}{\partial t} + \vec{\nabla}(\rho w \vec{V}) = -\frac{\partial p}{\partial z} + \frac{\tau_{xz}}{\partial z} + \frac{\tau_{yz}}{\partial z} + \frac{\tau_{zz}}{\partial z} + \rho f_z \tag{6}$$

Where f_x, f_y, f_z are body forces and t is time.

Note that the Right Hand Side of momentum must be multiplied by $\frac{1}{Re}$ factor due to non-dimensionalisation.

But

$$\vec{\nabla}(\rho u \vec{V}) = \frac{\partial}{\partial x}(\rho uu) + \frac{\partial}{\partial y}(\rho uv) + \frac{\partial}{\partial z}(\rho uw) \tag{7}$$

$$\vec{\nabla}(\rho v \vec{V}) = \frac{\partial}{\partial x}(\rho vu) + \frac{\partial}{\partial y}(\rho vv) + \frac{\partial}{\partial z}(\rho vw) \tag{8}$$

$$\vec{\nabla}(\rho w \vec{V}) = \frac{\partial}{\partial x}(\rho wu) + \frac{\partial}{\partial y}(\rho wv) + \frac{\partial}{\partial z}(\rho ww) \tag{9}$$

Full momentum in strong conservative form is then:

$$\boxed{\begin{aligned}
\frac{\partial \rho u}{\partial t} + \frac{\partial}{\partial x}(\rho uu) + \frac{\partial}{\partial y}(\rho uv) + \frac{\partial}{\partial z}(\rho uw) &= -\frac{\partial p}{\partial x} + \frac{\tau_{xx}}{\partial x} + \frac{\tau_{yx}}{\partial y} + \frac{\tau_{zx}}{\partial z} + \rho f_x && (10)\\
\frac{\partial \rho v}{\partial t} + \frac{\partial}{\partial x}(\rho vu) + \frac{\partial}{\partial y}(\rho vv) + \frac{\partial}{\partial z}(\rho vw) &= -\frac{\partial p}{\partial y} + \frac{\tau_{xy}}{\partial x} + \frac{\tau_{yy}}{\partial y} + \frac{\tau_{zy}}{\partial z} + \rho f_y && (11)\\
\frac{\partial \rho w}{\partial t} + \frac{\partial}{\partial x}(\rho wu) + \frac{\partial}{\partial y}(\rho wv) + \frac{\partial}{\partial z}(\rho ww) &= -\frac{\partial p}{\partial z} + \frac{\tau_{xz}}{\partial x} + \frac{\tau_{yz}}{\partial y} + \frac{\tau_{zz}}{\partial z} + \rho f_z && (12)
\end{aligned}}$$

Momentum eqs in the Einstein notation: summing on j ($i = 1,2,3$)

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} + \frac{\tau_{ji}}{\partial x_j} + \rho f_i \tag{13}$$

The Energy Equation:

$$\frac{\partial}{\partial t}(\rho e) + \vec{\nabla}(\rho e \vec{V}) = \rho \dot{q} \tag{14}$$

$$+\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right) \tag{15}$$

$$-\frac{\partial}{\partial x}(up) - \frac{\partial}{\partial y}(vp) - \frac{\partial}{\partial z}(wp) \tag{16}$$

$$+\frac{\partial}{\partial x}(u\tau_{xx}) + \frac{\partial}{\partial y}(u\tau_{yx}) + \frac{\partial}{\partial z}(u\tau_{zx}) \tag{17}$$

$$+\frac{\partial}{\partial x}(u\tau_{xy}) + \frac{\partial}{\partial y}(u\tau_{yy}) + \frac{\partial}{\partial z}(u\tau_{zy}) \tag{18}$$

$$+\frac{\partial}{\partial x}(u\tau_{xz}) + \frac{\partial}{\partial y}(u\tau_{yz}) + \frac{\partial}{\partial z}(u\tau_{zz}) \tag{19}$$

$$+\rho \vec{f}\vec{V} \tag{20}$$

Energy equation expanded:

$$\boxed{\begin{aligned}\frac{\partial}{\partial t}(\rho e) + \frac{\partial}{\partial x}(\rho eu) + \frac{\partial}{\partial y}(\rho ev) + \frac{\partial}{\partial z}(\rho ew) &= \rho \dot{q}\\ &+\frac{\partial}{\partial x}(-q_x) + \frac{\partial}{\partial y}(-q_y) + \frac{\partial}{\partial z}(-q_z)\\ &-\frac{\partial}{\partial x}(up) - \frac{\partial}{\partial y}(vp) - \frac{\partial}{\partial z}(wp)\\ &+\frac{\partial}{\partial x}(u\tau_{xx}) + \frac{\partial}{\partial y}(u\tau_{yx}) + \frac{\partial}{\partial z}(u\tau_{zx})\\ &+\frac{\partial}{\partial x}(u\tau_{xy}) + \frac{\partial}{\partial y}(u\tau_{yy}) + \frac{\partial}{\partial z}(u\tau_{zy})\\ &+\frac{\partial}{\partial x}(u\tau_{xz}) + \frac{\partial}{\partial y}(u\tau_{yz}) + \frac{\partial}{\partial z}(u\tau_{zz})\\ &+\rho\left(uf_x + vf_y + wf_z\right)\end{aligned}} \tag{21}$$

Energy equation in the Einstein notation

$$\frac{\partial}{\partial t}(\rho e) + \frac{\partial}{\partial x_j}(\rho e u_j - k\frac{\partial T}{\partial x_j} + u_j p + u_i \tau_{ji}) - \rho \dot{q} = 0 \tag{22}$$

Where:

$$e = e_0 + \frac{u^2+v^2+w^2}{2} = e_0 + \frac{u_j u_j}{2} \tag{23}$$

$$\rho \vec{f}\vec{V} = \rho\left(u f_x + v f_y + w f_z\right) \tag{24}$$

$$\vec{\nabla}(\rho e\vec{V}) = \frac{\partial}{\partial x}(\rho e u) + \frac{\partial}{\partial y}(\rho e v) + \frac{\partial}{\partial z}(\rho e w) \tag{25}$$

$$\vec{f} = \begin{bmatrix} fx \\ fy \\ fz \end{bmatrix} \tag{26}$$

$$q_j = -k\frac{\partial T}{\partial x_j} = -C_p \frac{\mu}{Pr}\frac{\partial T}{\partial x_j} \tag{27}$$

$$Pr = \frac{C_p \mu}{k} \tag{28}$$

$$\gamma = \frac{C_p}{C_v} \tag{29}$$

$$p = \rho R T \tag{30}$$

$$e_0 = C_v T \tag{31}$$

$$C_p - C_v = R \tag{32}$$

$$a = \sqrt{\gamma R T} \tag{33}$$

And

$$p = (\gamma - 1)\left[e - \frac{\rho}{2}(u^2+v^2+w^2)\right] \tag{34}$$

Where γ, C_p, C_v, R are constants. Eq 34 is from Pulliam 2014 [141] , p.61.

Combined the equations read. Subscript indicates viscous terms, no subscript are inviscid terms.

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} - \frac{\partial F_v}{\partial x} - \frac{\partial G_v}{\partial y} - \frac{\partial H_v}{\partial z} = B_f \tag{35}$$

Where:

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{bmatrix} \tag{36}$$

$$F = \begin{bmatrix} \rho u \\ \rho uu + p \\ \rho vu \\ \rho wu \\ \rho eu + q_x + up \end{bmatrix} \tag{37}$$

$$G = \begin{bmatrix} \rho v \\ \rho uv \\ \rho vv + p \\ \rho wv \\ \rho ev + q_v + vp \end{bmatrix} \tag{38}$$

$$H = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho ww + p \\ \rho ew + q_w + wp \end{bmatrix} \tag{39}$$

$$F_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u(\tau_{xx} + \tau_{xy} + \tau_{xz}) \end{bmatrix} \tag{40}$$

$$G_v = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ v(\tau_{yx} + \tau_{yy} + \tau_{yz}) \end{bmatrix} \tag{41}$$

(42)

$$H_v = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ v(\tau_{zx}+\tau_{zy}+\tau_{zz}) \end{bmatrix} \tag{43}$$

$$B_f = \begin{bmatrix} 0 \\ f_x \\ f_y \\ f_z \\ \rho(uf_x+vf_y+wf_z) \end{bmatrix} \tag{44}$$

(45)

Shear Stresses τ are computed from:

$$\tau_{x_ix_j} = \frac{1}{Re_L}\left[\mu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)+\lambda\frac{\partial u_u}{\partial x_u}\partial_{ij}\right] \tag{46}$$

Where ∂_{ij} is the standard Kronecker delta. The original equation includes $\frac{M_\infty}{Re_L}$ term instead of $\frac{1}{Re_L}$. Also, when simulating turbulent flow, μ is defined as $\mu_p+\mu_T$, i.e. viscosity is sum of physical and numerical turbulent 'viscosity'.

A value for turbulent viscosity μ_T is now needed at each computational node. It is a major field of study and Reynolds Averaged Navier Stokes (RANS) and Favre Averaged Navier Stokes (FANS) methodology will be used in the present work. While the full derivation of both is rather complex and can be found in [142], an extract used is shown here. The methodology inside the code essentially boils down to several relatively simple manipulations of the Navier Stokes equations, as follows.

Begin by replacing all variables by their time-mean, removing time dependent terms, τ_{ji} in the above full N-S equations is replaced by: (derivation is common in books)

$$\bar{\tau}_{ji} - \overline{\rho u_i'' u_j''} \tag{47}$$

$$R_{ij} = -\overline{\rho u_i'' u_j''} \tag{48}$$

In continuity, only the time term is ommited and values replaced by mean

$$\frac{\partial\bar{\rho}\bar{u}}{\partial x}+\frac{\partial\bar{\rho}\bar{v}}{\partial y}+\frac{\partial\bar{\rho}\bar{w}}{\partial z}=0 \tag{49}$$

In momentum, the following Reynolds stresses are added to x, y, z equation respectively

$$\boxed{\begin{aligned} R_x &= \frac{\partial}{\partial x}\left(-\overline{\rho u''u''}\right) + \frac{\partial}{\partial y}\left(-\overline{\rho v''u''}\right) + \frac{\partial}{\partial z}\left(-\overline{\rho w''u''}\right) && (50)\\ R_y &= \frac{\partial}{\partial x}\left(-\overline{\rho u''v''}\right) + \frac{\partial}{\partial y}\left(-\overline{\rho v''v''}\right) + \frac{\partial}{\partial z}\left(-\overline{\rho w''v''}\right) && (51)\\ R_z &= \frac{\partial}{\partial x}\left(-\overline{\rho u''w''}\right) + \frac{\partial}{\partial y}\left(-\overline{\rho v''w''}\right) + \frac{\partial}{\partial z}\left(-\overline{\rho w''w''}\right) && (52) \end{aligned}}$$

Resulting in 9 extra terms, 6 of which are unique.

For the energy equation, in addition to the above τ_{ji} modification there is also Reynolds heat flux added to the flux term:

$$\frac{\partial}{\partial x_j}\left(-k\frac{\partial T}{\partial x_j} + \overline{\rho u_j{}''h''}\right) \tag{53}$$

Where $Q_{ij} = \overline{\rho u_j{}''h''}$ is the Reynolds heat flux. Overall there are 7 extra unknowns introduced by Reynolds/Favre averaging.

The key assumption now is the Boussinesq approximation:

$$R_{ij} \propto S_{ij} \tag{54}$$

Where S_{ij} is the strain rate tensor and constant of proportionality is turbulent viscosity μ_t.

$$\boxed{R_{ij} = \mu_t\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\partial_{ij}\right) - \frac{2}{3}\rho k \partial_{ij}} \tag{55}$$

Where k is turbulence kinetic energy on which most two equation turbulence models are based.

$$k = \frac{1}{2}\overline{v_i'v_i'} \tag{56}$$

The Reynolds heat flux is replaced by:

$$Q_{ij} = -k_t \frac{\partial T}{\partial x_j} \tag{57}$$

and:

$$k_t = \frac{c_p \mu_t}{Pr_t} \tag{58}$$

Pr_t is turbulent Prandtl number, typically 0.71 and considered constant for a material. Now its *only* a question how to compute k and μ_t.

The full τ_{ji} for RANS with Boussinesq approximation is: (see τ_{ji} and R_{ji} on previous pages:

$$\overline{\tau_{ji}} = (\mu + \mu_t)\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\partial_{ij}\right) - \frac{2}{3}\rho k \partial_{ij} \tag{59}$$

All in terms of averaged quantities.
q_j, the total heat flux term becomes:

$$q_j = -k\frac{\partial T}{\partial x_j} - k_t\frac{\partial T}{\partial x_j} = -\left(k + k_t\right)\frac{\partial T}{\partial x_j} \tag{60}$$

$$q_j = -c_p\left(\frac{\mu_t}{\mathrm{Pr}_t} + \frac{\mu}{\mathrm{Pr}}\right)\frac{\partial T}{\partial x_j} \tag{61}$$

Again in terms of averaged quantities.

In practice, 'mechanistically' turning the 'original' DNS equations to Favre Averaged equations boils down to the following steps:

- remove the time dependent terms $\frac{\partial}{\partial t}$
- add μ_t to the real kinematic viscosity
- add k_t to the real conductivity coefficient
- add equations to solve for k and μ_t via a turbulence model. $k = 0$ for some models

- add the $-\frac{2}{3}\rho k \partial_{ij}$ term to shear stresses based on turbulence kinetic energy, or omit depending on the model

This is not a strictly a 'scientific' approach but rather a mechanical one, however that clarity is important when developing a software from the ground up.

3.2.3 Non-dimensionalisation

All the variables are iterated in their non-dimensional form. After reading the grid from disk the variables are non-dimensionalised as follows. The three reference values (ρ, u, l, T) - reference velocity, reference length scale, reference density and reference temperature are user-provided in the input file and all the other reference values are derived from them:

$$u_{ref} = input \tag{62}$$

$$T_{ref} = input \tag{63}$$

$$\rho_{ref} = input \tag{64}$$

$$l_{ref} = input \tag{65}$$

$$p_{ref} = \rho_{ref} R_{gas} T_{ref} \tag{66}$$

$$\mu_{ref} = Sutherland\ law\ based\ on\ T_{ref} \tag{67}$$

Before writing the variables for post-processing they are scaled back with the above. Non-dimensionalisation also means that the right hand side of the momentum equation must be divided by $Re = \frac{\rho u L}{\mu}$ to reflect the units correctly.

3.2.4 SA turbulence model

Spalart Almaras RANS turbulence model [143] was implemented in the GPU IBM code as a starting point due to its relative simplicity. The model was originally designed for aerospace applications and is judged suitable for initial testing. More complex models such as $k-\omega$ will be implemented as the code matures.

3.2.5 Spatial discretisation

Primarily a combination of the standard explicit central 2nd and 4th order Finite Difference discretisation is used for the Navier-Stokes equations for both viscous and the inviscid terms.

$$\frac{\delta^2\phi}{\delta x_i^2} = \phi_{i+1,j,k} - 2\phi_{i,j,k} + \phi_{i-1,j,k} \tag{68}$$

$$\frac{\delta^4\phi}{\delta x_i^4} = \phi_{i+2,j,k} - 4\phi_{i+1,j,k} + 6\phi_{i,j,k} - 4\phi_{i-1,j,k} + \phi_{i-2,j,k} \tag{69}$$

An upwinding technique was also attempted to stabilise the solution but yielded very limited results here and was eventually abandoned. Nodes near the non IBM boundaries are modified to lower order or forced to a one-sided formulation. It was possible to preserve the order of discretisation nodes near the IBM boundaries due to the Ghost Nodes approach and it is expected that this will be the case for up to 4th order for non-compact methods and 6th for compact high order. It also appears possible to use more GNs deeper inside the non-fluid domain for an even higher order near wall formulation however this investigation was beyond the scope of the current work and no works were found that examine this possibility in more detail.
The geometric metric coefficients of generalised curvilinear coordinate transformation must be computed numerically also as no analytical solution is available. Pure 2nd order central difference is used for this purpose although it was found that higher order metric calculation can change the solution notably towards more stable or accurate one. At present, this will not be investigated in more detail but could potentially be critical to achieve a stable numerical solution. As mentioned earlier, a combination of 2nd and 4th order high frequency filtering is used to stabilise the numerical solution.

3.2.6 Temporal discretisation

To discretise the time derivatives initially the explicit Runge Kutta 4th order integration algorithm is used as described in detail by Yu [144]. Jameson's [145] dual time stepping is used for unsteady solution. This allows steady-like sub-iterations in time to be performed (in pseudo-time space) between each physical time-step, hence the same time stepping algorithms can be used for steady and unsteady solution, including any multi-grid or other convergence accelration strategy. Without the dual time-stepping these aren't applicable.

$$RHS = \frac{\delta\phi}{\delta\tau} \tag{70}$$

Where τ is pseudotime, t is real time, RHS is set of residuals obtained by current (at this point of RK stage) field values and Q are the actual field variables. For steady state

simulations where no physical temporal terms are present this can also be used to iterate to 'infinity' or 'steady state' where Right Hand Side of the N-S is zero.
The time boundary (the first time step) where previous time step data is not yet available is discretised with one sided difference.

$$\frac{\delta\phi}{\delta\tau} \approx \frac{\phi^{m+1} - \phi^m}{\Delta\tau} \tag{71}$$

$$\phi^{m+1} = \phi^m + \Delta\tau RHS^m = \phi^m + \Delta\tau R^m = \phi^m + \Delta\tau R(Q^m) \tag{72}$$

Where $m+1$ is the next pseudotime level (not the next RK level) The overall procedure is as follows. Q^m is the previous iteration value, or from initialisation. Q^1, Q^2, Q^3, Q^{m+1} are obtained using algorithms with Under relaxation factors (URFs)

$$Q^1 = Q^m + \frac{\Delta\tau}{2} R(Q^m) \tag{73}$$

$$Q^2 = Q^1 + \frac{\Delta\tau}{2} R\left(Q^1\right) \tag{74}$$

$$Q^3 = Q^2 + \Delta\tau R\left(Q^2\right) \tag{75}$$

$$Q^{m+1} = Q^3 + \frac{\Delta\tau}{6}\left(R^m + 2R^1 + 2R^2 + R^3\right) \tag{76}$$

Q^m values are saved in a separately allocated memory before the RK iterations and used through RK procedure. When updating primitive variables after each RK stage, the residual from the current RK stage along with the pre-RK values are used. The RK terms (e.g. $\frac{\Delta\tau}{2} R\left(Q^1\right)$) are calculated the same way in both cases.

$$\frac{\delta\phi}{\delta\tau} + \frac{\delta\phi}{\delta t} = R(Q^m) \tag{77}$$

Similar first order backward FD for the physical time t. m is pseudotime n is real time

$$\frac{\phi^{m+1} - \phi^m}{\Delta\tau} + \frac{\phi^{n+1} - \phi^n}{\Delta t} = R(Q^m) \tag{78}$$

$$\frac{\phi^{m+1} - \phi^m}{\Delta\tau} = R(Q^m) - \frac{\phi^{n+1} - \phi^n}{\Delta t} \tag{79}$$

$$\phi^{m+1} = \phi^m + \Delta\tau \left(R(Q^m) - \frac{\phi^{n+1} - \phi^n}{\Delta t} \right) \tag{80}$$

$$\phi^{m+1} = \phi^m + \Delta\tau (R(Q^m)) - \Delta\tau \left(\frac{\phi^{n+1} - \phi^n}{\Delta t} \right) \tag{81}$$

Compare this to the steady equation

$$\phi^{m+1} = \phi^m + \Delta\tau R(Q^m) - steady \tag{82}$$

$$\phi^{m+1} = \phi^m + \Delta\tau \left(R(Q^m) - \frac{\phi^{n+1} - \phi^n}{\Delta t} \right) - unsteady \tag{83}$$

Effectively, an extra unsteady term is added to residual each time it is calculated

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} == \frac{\phi^n - \phi^{n-1}}{\Delta t} \tag{84}$$

Δt is known and variables from previous iterations are saved (ϕ^{n-1}) before updating.

Several pseudo-time iterations are then performed on the following, until the values between the pseudotimesteps no longer change (i.e. convergence)

$$\phi^{m+1} = \phi^m + \Delta\tau \left(R(Q^m) - \frac{\phi^{n+1} - \phi^n}{\Delta t} \right) \tag{85}$$

$$\frac{\phi^{m+1} - \phi^m}{\Delta\tau} \approx 0 \tag{86}$$

In practice, before each iteration, values of the current physical time step are saved in separate memory as ”old”.

Then at the end of the residual calculation routine term related to the time derivative is computed and added to the total residual. The total residual is then used in the pseudotime-RK scheme as previously.

3.2.7 Distance to the nearest wall

As explained in Section 2.6.1 to compute distance to the nearest wall required by the majority of the turbulence models, the Poisson equation was solved as a precursor calculation before each turbulent simulation. The procedure and full set of equations is explained in more detail in Tucker et.al. [4]:

$$\nabla^2\phi = -1 \tag{87}$$

Standard second order central differences were used to discretise the equation. It was found that no artificial dissipation was required here and the solution was stable up to much higher Courant number than the Navier-Stokes equations.

While there exist many other algorithms to compute the wall distances Poisson was chosen for simplicity of implementation but also as the first equation that will be solved numerically. This will allow the author to debug software and numerical issues on a relatively simple algorithm while still performing useful calculations.

3.2.8 Metric transformation

To enable practical computations with the Finite Difference method, the equations must be transformed to generalised curvilinear coordinates, where each variable is transformed from real to computational space. Metric transformation is not strictly necessary for the Immersed Boundary Method however the code is capable of performing non-IBM multiblock computations. In addition one may create outer curvilinear grid with classical B.C.s and only use the IBM to impose B.C. on the complex inner shape to enhance computational efficiency and reduce complexity hence room for error. All the variables are iterated in their non-dimensional form but stored and written out as dimensional for ease of postprocessing.

$$\hat{\Phi} = \frac{\Phi}{J} \tag{88}$$

$$\xi = \xi(x,y,z,t) \tag{89}$$

$$\eta = \eta(x,y,z,t) \tag{90}$$

$$\zeta = \zeta(x,y,z,t) \tag{91}$$

$$\tau = \tau(t) \tag{92}$$

$$J = \frac{\partial(\xi,\eta,\zeta,t)}{\partial(x,y,z,t)} \tag{93}$$

Where J is Jacobian and contravariant velocity components are computed as follows wherever needed:

$$U = \xi_x u + \xi_y v + \xi_z w \tag{94}$$

$$V = \eta_x u + \eta_y v + \eta_z w \tag{95}$$

$$W = \zeta_x u + \zeta_y v + \zeta_z w \tag{96}$$

More detailed derivation and full formulation is available in [141] but it is also common in literature and will not be presented here.

3.2.9 Oxford Parallel library for Structured applications (OPS)

One of the key novel elements in the present work is the approach to parallelisation and performance portability. Novel set of libraries is used where the scientific code is clearly separated from the parallel implementation. Single source code can be translated into various parallel implementations easily. This is achieved by expressing the problem to be solved in OPS's custom syntax and the library parsing it to produce requested parallelisation automatically. The parsers are provided by OPS and are its key feature in scientific-computational decoupling.
One of the key requirements for the parallelisation approach was performance portability, hence a level of 'future-proof' design. This narrows down the search to mainly high level domain specific libraries. Oxford Parallel library for Structured grid applications (OPS) was chosen as it is very specifically designed for the type of computations appearing in a structured CFD code and integrates an efficient novel tridiagonal solver for implicit-type operations. The author also believes that efficient parallelisation requires significant expertise in the field and is best left to computation experts as opposed to scientific code developers. The fact that a code can be manually implemented on a target platform more efficiently (e.g. CUDA, MPI) than via high level library does not mean the engineering software developer will have the necessary expertise to achieve such results. It was indeed found [146] that often times due to that it is the high level library is more effective as the experts take time and focus on its back-end rather than individual instances. By utilising a high level library even a novice in parallelism can tap into the expertise coded into it.
An example loop structure is shown below:

```
#include "stdlibs.h"
#define OPS_3D
#include "ops_seq.h"
#include "function_headers.h"
#include "def_globals.h"

```

```
#include " update_visc_kernel .h"

void CALL_UPDATE_VISC( ) {

ops_par_loop ( update_visc_kernel , " update_visc_kernel ", dolphin_block , 3,
              range_full ,
                  ops_arg_dat (MU      , 1, S3D_000, "double", OPS_WRITE),
                  ops_arg_dat (T       , 1, S3D_000, "double", OPS_READ ),
                  ops_arg_dat (IBLANK_I, 1, S3D_000, "int"   , OPS_READ ),
                  ops_arg_idx ()
                  );

}
```

And the implementation itself:

```
#ifndef UPDATE_VISC_KERNEL_H
#define UPDATE_VISC_KERNEL_H

#include " stdlibs .h"
#include " function_headers .h"
#include " def_globals .h"

void update_visc_kernel ( double *MU, const double *T, const int *IBLANK_I,
    const int *idx ) {

if (((INPUTI[44] == 1) && (IBLANK_I[OPS_ACC2(0, 0, 0)] !=60)) ||
    (INPUTI[44] == 0)) {

MU[OPS_ACC0(0, 0, 0)] = pow(T[OPS_ACC1(0, 0,
    0)],3.0/2.0) *((1.0+(110.4/ REF[2]))/(T[OPS_ACC1(0, 0, 0)]+(110.4/REF[2])));

}

}

#endif
```

Which differs from serial implementation relatively little:

```
#include " stdlibs .h"
#include " function_headers .h"
#include " def_globals .h"

void LOOP_KJI(double **q, int **qi ) {
int i, j, k;

qi[PNT][I1] = 1; // i increment
qi[PNT][I2] = qi[MESH][I]; // j increment
qi[PNT][I3] = qi[MESH][I]*qi[MESH][J]; //k increment
qi[PNT][DIR] = 1; // Direction
qi[PNT][ID1MN] = 0;
qi[PNT][ID1MX] = qi[MESH][I] - 1;
qi[PNT][ID2MN] = 0;
qi[PNT][ID2MX] = qi[MESH][J] - 1;
qi[PNT][ID3MN] = 0;
qi[PNT][ID3MX] = qi[MESH][K] - 1;

if (qi[MESH][K]==1) { qi[PNT][I3]=0; }

for (k = 0; k < qi[MESH][K]; k++) {
for (j = 0; j < qi[MESH][J]; j++) {
for (i = 0; i < qi[MESH][I]; i++) {

qi[PNT][ID3] = k;
qi[PNT][ID2] = j;
qi[PNT][ID1] = i;
qi[PNT][IN] = (k*qi[MESH][J] + j)*qi[MESH][I] + i;
UPDATE_VISC(q,qi);

}
}
}

}
```

And the body of update visc:

```
#include "stdlibs.h"
#include "function_headers.h"
#include "def_globals.h"

void UPDATE_VISC(double **q, int** qi) {
 int in;

in = qi[PNT][IN];

q[MU][in] = pow(q[T][in],3.0/2.0) *
 ((1.0+(110.4/q[REF][2]))/(q[T][in]+(110.4/q[REF][2])));

}
```

Figure 3.1 shows a quick comparison of a typical hand-implemented structured block computation in C with three nested loops with the equivalent OPS operation. A full sample of one routine is also presented. In the OPS code, the problem is always expressed in two parts, one is the 'kernel' where computations are performed, the other is the actual parallel loop request via 'par_loop'. The kernel contains bulk of the calculations while par_loop is effectively how are they to be performed, how the data is to be accessed, dimensionality and other parameters. OPS can then parse these and produce

```
ops_par_loop(update_visc_kernel,"update_visc_kernel",
             dolphin_block,3,range_full,
             ops_arg_dat(MU      ,1,S3D_000,"double",OPS_WRITE),
             ops_arg_dat(T       ,1,S3D_000,"double",OPS_READ ),
             ops_arg_dat(IBLANK_I,1,S3D_000,"int"   ,OPS_READ ),
             ops_arg_idx()
             );
```

```
void update_visc_kernel(double *MU,double *T,int *IBLANK_I,int *idx) {
  MU[OPS_ACC0(0, 0, 0)] = pow(T[OPS_ACC1(0, 0, 0)],3.0/2.0)*
  ((1.0+(110.4/REF[2]))/(T[OPS_ACC1(0, 0, 0)]+(110.4/REF[2])));
}
```

```
for (k=0;k<kdim;k++) {
  for (j=0;j<jdim;j++) {
    for(i=0;i<idim;i++) {
      q(i,j,k)=pow(T(i,j,k),3.0/2.0)*
      ((1.0+(110.4/REF[2]))/(T(i,j,k)+(110.4/REF[2])));
    }
  }
}
```

Figure 3.1: Comparison of the serial C code with parallel OPS implementation.

an optimised version of all the parallel implementation (e.g. CUDA, MPI, OpenMP, etc.)

3.2.10 Alternating Direction Implicit (ADI) solver

While in conventional tridiagonal solution physical walls (and grid edge) coincide with boundaries of the matrix, this is not the case with the IBM here and effective "wall" nodes are typically surrounded by other nodes, solid or fluid. A fundamental limitation of OPS is that entire data structures are operated on and the libraries make no distinction as to which node is solid and which is fluid; all are computed equally. To eliminate the impact of values inside the solid to the fluid solution a modification to the lower and upper diagonal coefficients was devised in the vicinity of the IBM geometry, as shown on Figure 3.2. This proved to be very important to stability as well as accuracy of the scheme. Several configurations were tried and the one shown on the figure proved to be the most stable.

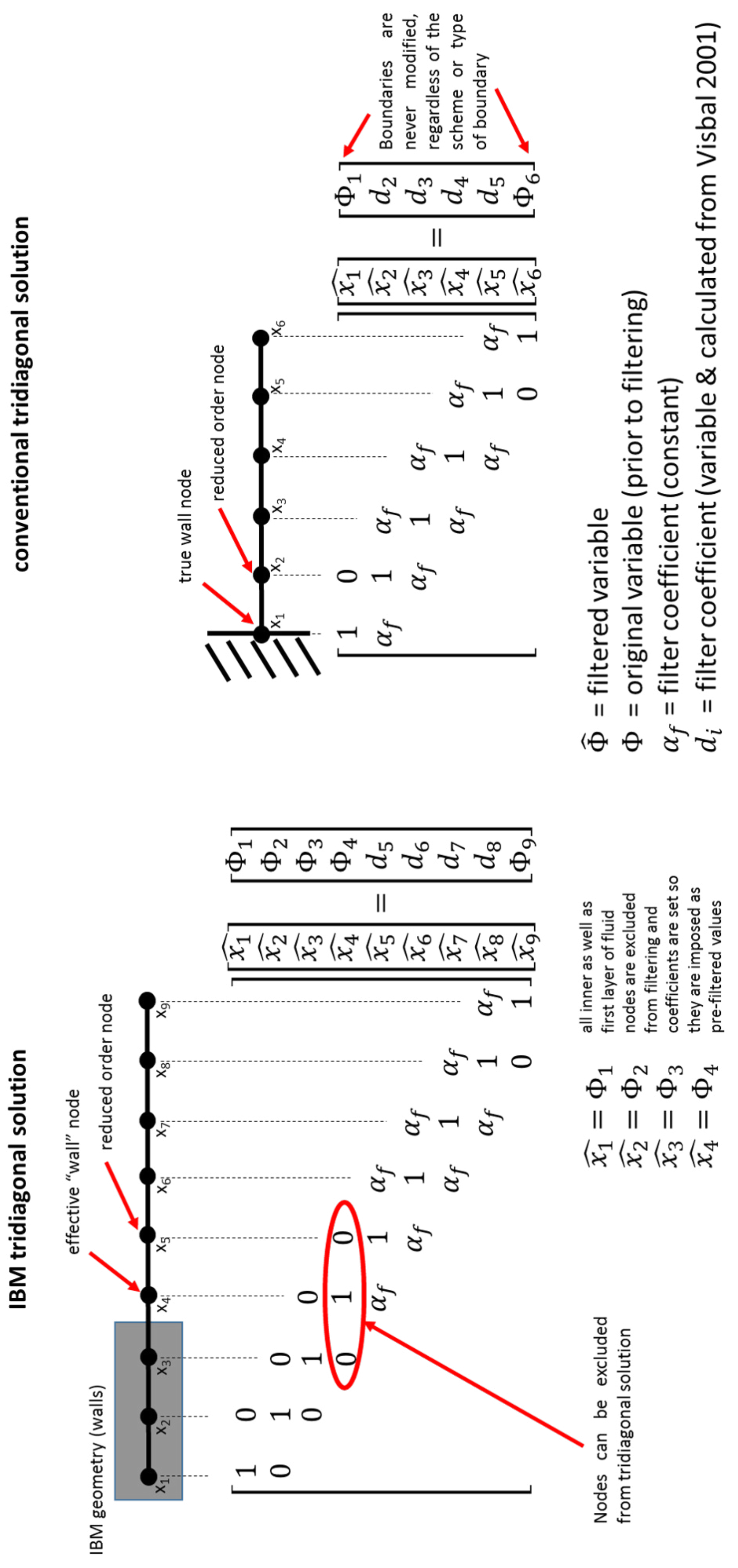
IBM tridiagonal solution
IBM geometry (walls)
effective "wall" node
reduced order node
Nodes can be excluded from tridiagonal solution
all inner as well as first layer of fluid nodes are excluded from filtering and coefficients are set so they are imposed as pre-filtered values
$\widehat{x_1} = \Phi_1$
$\widehat{x_2} = \Phi_2$
$\widehat{x_3} = \Phi_3$
$\widehat{x_4} = \Phi_4$
conventional tridiagonal solution
true wall node
reduced order node
Boundaries are never modified, regardless of the scheme or type of boundary
$\hat{\Phi}$ = filtered variable
Φ = original variable (prior to filtering)
α_f = filter coefficient (constant)
d_i = filter coefficient (variable & calculated from Visbal 2001)

Figure 3.2: Schematic of modification required near the walls for the ADI in the OPS framework.

3.2.11 Artificial dissipation

As the code has non-staggered variable arrangement, an odd-even decoupling of pressure and velocity is likely to impact the solution. Combined with pure central differencing and inherent instabilities and non-linearities present in engineering turbulent flows, the solution is prone to developing oscillations that eventually lead to divergence, in particular for more complex flows or geometries. To remedy this problem and achieve stability, some amount of artificial dissipation, or viscosity must be added, or the equations otherwise stabilised numerically by other means. More detailed study as to why this is the case is presented by Chen et.al. p.76. [147].

It was found in the present work that the problem of oscillations and odd-even decoupling is major and no single scheme tested gave satisfactory results, hence the literature search was extended significantly to find a suitable artificial dissipation scheme. It was also found that only the inviscid, or convective Navier-Stokes terms require some stabilising treatment; the viscous or diffusive terms appear to be numerically stable with very little if any additional modifications.

An inherent instability and unboudedness of pure central differencing at Peclet number $Pe > 2$ should also be noted. This adds to development of numerical oscillations. Upwind schemes, which are unconditionally bounded were also considered. Boundedness comes from the idea of solving hyperbolic PDEs and direction from which information propagates. Upwinding is bounded due to stencil being biased towards where the information originates.

It also became apparent via extensive numerical experiments that varying the smoothing coefficients even by relatively small amounts changed the solution significantly as shown on Figure 3.3, frequently to the point where the solution was too viscous to predict any separation at all. The search for an appropriate artificial dissipation scheme began and this chapter summarises its findings. The goal is to include the minimum amount of dissipation possible to eliminate oscillations, while not affecting the accuracy with artificial viscosity.

In an ideal scenario, a sophisticated and well understood implicit smoother such as that presented by Visbal [107] would be used. Its order is easily variable and can be increased up to 10 for use with high order discretisation techniques. In addition such implicit filtering scheme is less sensitive to user input coefficients, reducing scope for

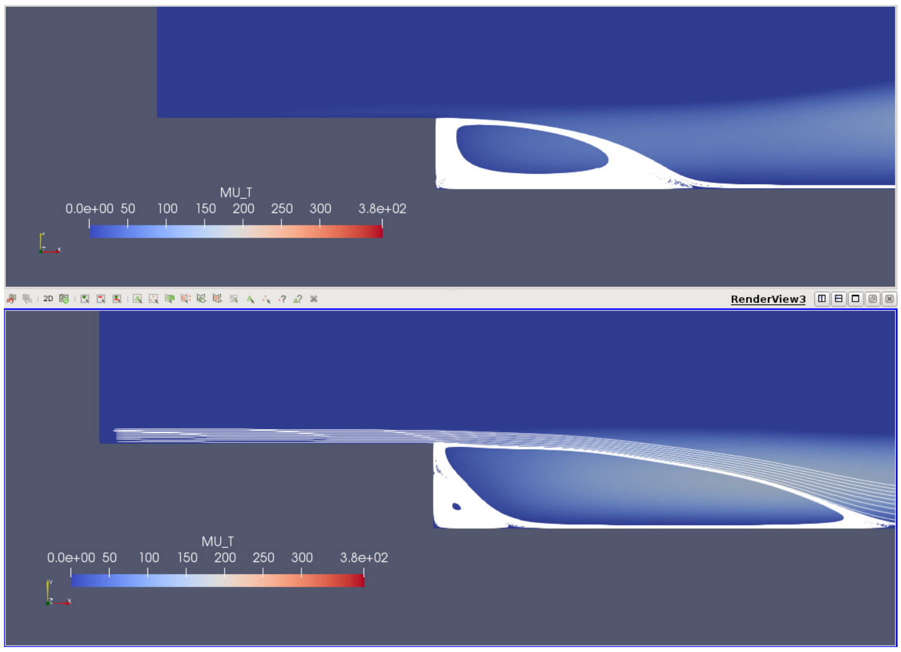
MU_T
0.0e+00 50 100 150 200 250 300 3.8e+02
RenderView3
MU_T
0.0e+00 50 100 150 200 250 300 3.8e+02

Figure 3.3: Result obtained with different filtering coefficients. Excessive damping on top, correct solution on the bottom. Non-dimensionalised μ_t shown.

error and increasing scope for automation and robustness. It is recommended by Visbal, Wang [8, 97] as well as generally accepted in the field that the filtering operation should be at least two orders of accuracy higher than the main N-S discretisation scheme. The filter by Visbal [107] however requires a tridiagonal system of equations to be solved, capability which is relatively immature within the code in the parallel OPS framework.

Explicit, larger stencil or less sophisticated smoothers must therefore be employed at least initially and these are generally much less optimised for providing the right amount of smoothing. It was also found via numerical experiments in the present work that explicit smoothers tend to be more sensitive to input coefficients. Excellent reviews of the smoothers, both explicit and implicit with reasoning and derivations are provided by Ekaterinaris [148], Pulliam [149] and more recently in 2016 by Maulik et.al. [150].

The first smoother investigated was that of Turbostream code, by Brandvik and Pullam [65]. A dissipation term D_i is added to the right hand side of the Navier-Stokes equations on a per-direction basis in each direction:

$$D_i = \varepsilon_i^{(2)} \left(\frac{\delta^2 \phi}{\delta x_i^2} \right) + \varepsilon_i^{(4)} \left(\frac{\delta^4 \phi}{\delta x_i^4} \right) \tag{97}$$

Where ϕ is a variable to be filtered (typically the very minimum is velocity vector components) and x_i is the direction along which the filtering is done. Both constant and variable non-dimensional coefficients $\varepsilon_i^{(2)}$, $\varepsilon_i^{(4)}$ were tried. Where variable, the definition reads:

$$\varepsilon_i^{(2)} = \kappa^{(2)}\alpha_i \tag{98}$$

$$\varepsilon_i^{(4)} = max\left(0, \kappa^{(4)} - \varepsilon_i^{(2)}\right) \tag{99}$$

Where $\kappa^{(4)}$ and $\kappa^{(2)}$ are again user supplied non-dimensional coefficients based on past experience and α_i is a pressure oscillation sensor and reads:

$$\alpha_i = \frac{\left|p_{i+1,j,k} - 2p_{i,j,k} + 2_{i-1,j,k}\right|}{p_{i+1,j,k} + 2p_{i,j,k} + 2_{i-1,j,k}} \tag{100}$$

Discretisation of the derivatives is performed to 2nd order of accuracy via standard central differences:

$$\frac{\delta^2\phi}{\delta x_i^2} = \phi_{i+1,j,k} - 2\phi_{i,j,k} + \phi_{i-1,j,k} \tag{101}$$

$$\frac{\delta^4\phi}{\delta x_i^4} = \phi_{i+2,j,k} - 4\phi_{i+1,j,k} + 6\phi_{i,j,k} - 4\phi_{i-1,j,k} + \phi_{i-2,j,k} \tag{102}$$

With standard one sided differences near the boundaries.

This smoother was tried with and without the pressure sensor, with and without 4th derivatives and proved to be unsuccessful as with the current numerical setup the oscillations were not removed even with very high values of coefficients. The solution always lead to divergence via global oscillations, regardless of initial conditions and search for further stabilising solution continued.

Ciardi et.al. propose a self-adapting 2nd order explicit smoother [151]. The smoother is used successfully in the framework of Finite Volume, staggered code Hydra, developed by Rolls-Royce. The idea is that smoothing coefficients are set based on 'wiggles' or local oscillations of any variable, not just pressure and the smoothing coefficient is varied locally, on a per-cell basis rather than based on an input value. Only the minimum and maximum allowed values of coefficients are specified.

Chen in his PhD thesis, p.112 [147] uses relatively simple explicit method of smoothing primitive flow variables and finds it is not overly sensitive to user input coefficients:

$$\phi^{new} = \phi^{old} + \omega\left(\frac{\delta^2\phi^{old}}{\delta x^2} + \frac{\delta^2\phi^{old}}{\delta y^2}\right) \tag{103}$$

Where ω is an input coefficient. The smoother is effectively a two dimensional (three in 3D) sum of derivatives in each direction that overwrites variables with those of increased viscosity. ω in the range of 0.05 and 0.2 was used by Chen and he concluded that it provides similar levels of smoothing as widely used implicit smoother as described by Anderson [152]. This smoother was implemented in its original form and found to damp the solution so drastically that any separation from the Immersed Boundary Surface was impossible, no matter the Reynolds number or value of the coefficient. If the coefficient was reduced to 0.0001 the solution was not damped however began to exhibit oscillations leading eventually to convergence. This may only be due to the use of IBM and was not investigated further.

Instead, the following two modifications were made to this smoother:

- Fourth order derivatives were included and again discretised with standard central differences as explained above.

$$\phi^{new} = \phi^{old} + \omega^{(2)} \left(\frac{\delta^2 \phi^{old}}{\delta x^2} + \frac{\delta^2 \phi^{old}}{\delta y^2} \right) + \omega^{(4)} \left(\frac{\delta^4 \phi^{old}}{\delta x^4} + \frac{\delta^4 \phi^{old}}{\delta y^4} \right) \tag{104}$$

- The modified scheme was combined with the Self-adaptive Discretization Scheme (SDS) technique for coefficients as described in Jefferson-Loveday PhD thesis, p.55 [153]

The SDS scheme locally varies the value of coefficients with minimum/maximum bounds, similar to that of Ciardi [151], however a combination of second and fourth order derivatives is employed and the variables are overwritten instead of a term added. At the end of each time step, or pseudotimestep for steady iterations, each cell is probed for neighbouring oscillations and the local smoothing coefficient is increased if the oscillation is detected, decreased if it isn't. This proved the most effective technique to date and is the state of art in the present work.

Finally, on the subject of filtering it must also be mentioned that in the present numerical setup values of filtering coefficients appear to have great effect on the final solution. Typical values for second and fourth order coefficients used by other researchers were investigated [154, 155, 156] and found to be:

$$\varepsilon^{(2)} = 0.2 - 0.5 \tag{105}$$

$$\varepsilon^{(4)} = 0.004 - 0.0156 \tag{106}$$

Typical values, limits and change increment used in the present work are:

$$\varepsilon_{max} = 0.001 \tag{107}$$

$$\varepsilon_{min} = 0.000001 \tag{108}$$

$$\Delta\varepsilon = 0.000001 \tag{109}$$

The exact values very much depend on a range of practical factors such as non-dimensionalisation techniques for N-S, overall combination of numerical schemes, typical grid resolutions, Re to name few. The goal however with any type of coefficients like this in filtering schemes is to arrive at the minimum possible values where numerical stability of solution is achieved. Filtering schemes are hugely important and very few if any CFD codes would be numerically stable without them but they are fundamentally unphysical and not derived from conservation laws or other physical assumptions.
The values change significantly depending on the numerical and computational methods employed by different researchers and based on numerical experiments performed here it is suspected that the IBM and the filtering procedure may be strongly interdependent. The values in the SDS scheme are also time and space dependent. This would be an excellent area to deploy a Machine Learning algorithm in the future.
Another variation and discussion of artificial viscosity is provided by Anderson p.238 [152]. This smoother, combined with varying SDS coefficients is used as baseline.

Another common technique tested was upwinding as described by Patankar [157]. This method is widely used by commercial Finite Volume solvers such as ANSYS Fluent or StarCCM. A limiter with variable coefficient is usually employed to blend inherently unstable pure central differencing [149] with the upwind-discretised value to achieve maximum stability with minimum artificial dampening of the solution. Third order upwind formulae are used to discretise convective terms for all variables:

$$u^{-} = \frac{2u_{i-1} + 3u_i - 6u_{i-1} + u_{i-1}}{6\Delta x} \tag{110}$$

$$u^{+} = \frac{-u_{i+2} + 6u_{i+1} - 3u_i - 2_{i-1}}{6\Delta x} \tag{111}$$

Where either u^- or u^+ are used depending on local velocity sign. As the velocities are stored in memory in the real space, they must be transformed to computational space to obtain contravariant velocities as described in Visbal [107]:

$$U = \xi_x u + \xi_y v + \xi_z w \tag{112}$$

$$V = \eta_x u + \eta_y v + \eta_z w \tag{113}$$

$$W = \zeta_x u + \zeta_y v + \zeta_z w \tag{114}$$

Where ξ, η, ζ denote metric terms in each spatial direction respectively. Lowercase are real space, uppercase are computational space.

A promising explicit smoother is presented by Pulliam et.al. [141], p. 96-101. The smoother is sophisticated and simpler ones will be tested first before proceeding with this one. In two dimensions, dissipation terms D_ξ and D_η are added to ξ and η directions of the inviscid flux derivative respectively (added to the R.H.S. of the N-S, so essentially as they were viscous terms)

$$\left(D_\xi\right)_{j,k} = \nabla\xi \left(\varepsilon^{(2)}\widehat{|A|}J^{-1}\right)_{j+1/2,k} \Delta\xi Q_{j,k} \tag{115}$$

$$-\nabla\xi \left(\varepsilon^{(4)}\widehat{|A|}J^{-1}\right)_{j+1/2,k} \Delta\xi\nabla\xi\Delta\xi Q_{j,k} \tag{116}$$

$$\left(D_\eta\right)_{i,j} = \nabla\eta \left(\varepsilon^{(2)}\widehat{|A|}J^{-1}\right)_{i+1/2,k} \Delta\eta Q_{i,k} \tag{117}$$

$$-\nabla\eta \left(\varepsilon^{(4)}\widehat{|A|}J^{-1}\right)_{i+1/2,k} \Delta\eta\nabla\eta\Delta\eta Q_{i,k} \tag{118}$$

Where $\widehat{|A|}$ is matrix that is usually calculated via implicit techniques by $\widehat{|A|} = \frac{\delta\widehat{E}}{\delta\widehat{Q}}$ and is computationally demanding and difficult to solve. It can be approximated by the spectral radius of $\widehat{A}$ given by σ:

$$\widehat{|A|} \approx \sigma = |U| + a\sqrt{\xi_x^2 + \xi_y^2} \tag{119}$$

$$U = \xi_x u + \xi_y v + \xi_z w \tag{120}$$

Similarly for the other directions:

$$\widehat{|B|} \approx \sigma = |V| + a\sqrt{\eta_x^2 + \eta_y^2} \tag{121}$$

$$V = \eta_x u + \eta_y v + \eta_z w \tag{122}$$

Jacobian inverse J^{-1} is given by:

$$J^{-1} = \frac{1}{Vol} \tag{123}$$

Midpoint values of the second and fourth order terms can be avaluated by: (similarly for other directions)

$$\left(\varepsilon^{(2)}\widehat{|A|}J^{-1}\right)_{j+1/2,k} = \frac{1}{2}\left[\left(\varepsilon^{(2)}\widehat{|A|}J^{-1}\right)_{j,k} + \left(\varepsilon^{(2)}\widehat{|A|}J^{-1}\right)_{j+1,k}\right] \tag{124}$$

$$\left(\varepsilon^{(4)}\widehat{|A|}J^{-1}\right)_{j+1/2,k} = \frac{1}{2}\left[\left(\varepsilon^{(4)}\widehat{|A|}J^{-1}\right)_{j,k} + \left(\varepsilon^{(4)}\widehat{|A|}J^{-1}\right)_{j+1,k}\right] \tag{125}$$

Pressure sensor is used to control the second order terms. The sensor is capable of detecting shock waves and is similar to previously used one. The difference is that sensor coefficients are taken at neighbouring nodes and maximum value of the sensor taken:

$$\varepsilon^{(2)}_{j,k} = \kappa_2 max\left(\Gamma_{j+1,k},\Gamma_{j,k},\Gamma_{j-i,k}\right) \tag{126}$$

$$\Gamma = \left|\frac{p_{j+1,k}-2p_{j,k}+p_{j-1,k}}{p_{j+1,k}+2p_{j,k}+p_{j-1,k}}\right| \tag{127}$$

$$\varepsilon^{(4)}_{j,k} = max\left(0, \kappa_4 - \varepsilon^{(2)}_{j,k}\right) \tag{128}$$

Differencing operators Δ and ∇ required for the smoother are defined as:

$$\nabla\phi_j = \phi_j - \phi_{j-1} \tag{129}$$

$$\Delta\phi_j = \phi_{j+1} - \phi_j \tag{130}$$

Typical value of constants in this model are:

$$\kappa_2 = 0.5 \tag{131}$$

$$\kappa_4 = 0.02 \tag{132}$$

And finally, as the dissipation terms are applied to all the variables, Q is the solution vector as presented before:

$$Q = \begin{bmatrix}\rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e\end{bmatrix} \tag{133}$$

Based on the compounded differencing operators, terms can be expanded to:

$$\nabla\xi\Delta\xi Q_{j,k} = Q_{j-1,k} - 2Q_{j,k} + Q_{j+1,k} \tag{134}$$

$$-\nabla\xi\Delta\xi\nabla\xi\Delta\xi Q_{j,k} = -Q_{j-2,k} + 4Q_{j-1,k} - 6Q_{j,k} + 4Q_{j+1,k} - Q_{j+2} \tag{135}$$

And the final form in 2D reads:

$$D_i = \frac{1}{2}\left(Q_{i-1,j} - 2Q_{i,j} + Q_{i+1,j}\right)\left[\left(\varepsilon^{(2)}\widehat{|A|}J^{-1}\right)_{i,j} + \left(\varepsilon^{(2)}\widehat{|A|}J^{-1}\right)_{i+1,j}\right] \tag{136}$$

$$+\frac{1}{2}\left(-Q_{i-2,k} + 4Q_{i-1,j} - 6Q_{i,j} + 4Q_{i+1,j} - Q_{i+2}\right)\left[\left(\varepsilon^{(4)}\widehat{|A|}J^{-1}\right)_{i,j} + \left(\varepsilon^{(4)}\widehat{|A|}J^{-1}\right)_{i+1,j}\right] \tag{137}$$

As the fourth differential terms include $-/+2$ stencil, definition of the fourth order term near boundaries must be modified and replaced with the following relation:

$$-\nabla\xi\Delta\xi\nabla\xi\Delta\xi Q_{j,k} = -1\times Q_{j-2,k} + 4\times Q_{j-1,k} - 5\times Q_{j,k} + 2\times Q_{j+1,k} + 0\times Q_{j+2,k} \tag{138}$$

$$-\nabla\xi\Delta\xi\nabla\xi\Delta\xi Q_{j,k} = \quad 0\times Q_{j-2,k} + 2\times Q_{j-1,k} - 5\times Q_{j,k} + 4\times Q_{j+1,k} - 1\times Q_{j+2,k} \tag{139}$$

Where the top formula is applied near j_{max} wall and the bottom near j_{min} wall. Similar approach is done for the other directions. This artificial dissipation scheme is formally third order and capable of detecting and handling shocks with stability and appropriate level of dissipation. The definition near the walls is first order and proved to be dissipative and stable. Clearly this filtering scheme is rather complex and implementation would be time consuming, however it is judged such sophisticated techniques will be necessary to achieve a good balance of accuracy and dissipation with the IBM method. In final summary, there are five major artificial dissipation schemes tested and implemented in the code:

1. Simple explicit smoother as described by Anderson et.al. [152, 158]
2. 2nd order derivative technique as used by Chen [147]
3. A variant of the upwinding method
4. Pulliam advanced explicit shock capturing [141]
5. Self-adaptive Discretisation Scheme (SDS) [153]

The implicit high order filter by Visbal has not been implemented due to lack of a robust ADI solver at the time but is very much recommended in future works. The filters can be combined with each other and with careful consideration one should be able to arrive at a better balance of dampening and accuracy this way. In the future it is also recommended to try midpoint discretisation with all the central differences and investigate staggered arrangement for storing variables. The latter will likely require re-design of majority of the code however most commercial CFD codes appear to be using this technique.

One final problem to mention in this section is the well known fact low Mach (approx. < 0.3) impairs convergence in compressible solvers [159]. While there are specific schemes to handle this, they will not be implemented initially and flows of $Ma > 0.3$ will largely be used. Where that is not possible it will have to be accepted that convergence rates are suboptimal before a scheme to handle that issue is implemented.

3.2.12 Code procedures

A brief high level schematic of the iteration loops inside the program is shown below. Not all the routines are shown for obvious reasons.

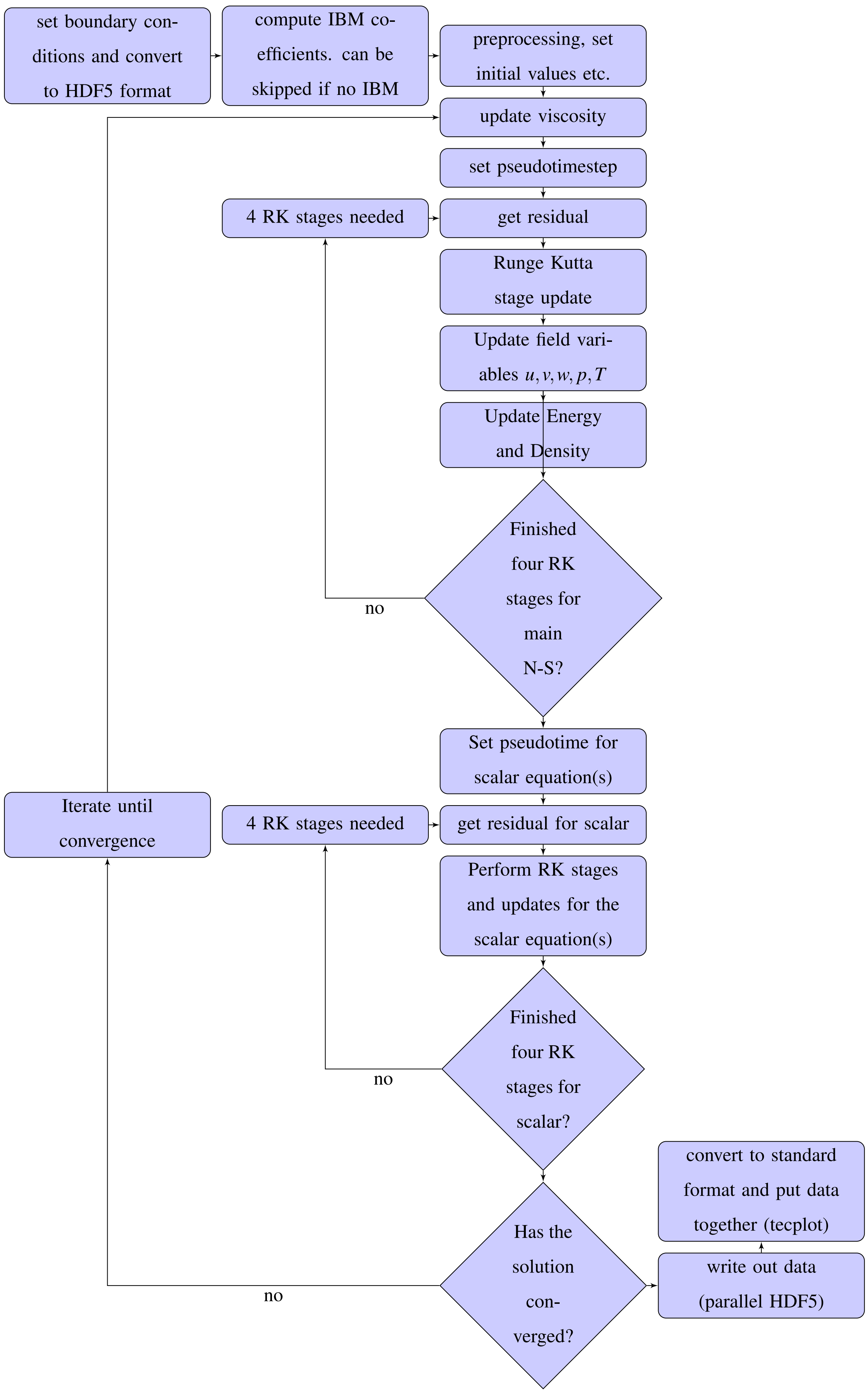
set boundary conditions and convert to HDF5 format
compute IBM coefficients. can be skipped if no IBM
preprocessing, set initial values etc.
update viscosity
set pseudotimestep
4 RK stages needed
get residual
Runge Kutta stage update
Update field variables u, v, w, p, T
Update Energy and Density
Finished four RK stages for main N-S?
no
Set pseudotime for scalar equation(s)
Iterate until convergence
4 RK stages needed
get residual for scalar
Perform RK stages and updates for the scalar equation(s)
Finished four RK stages for scalar?
no
convert to standard format and put data together (tecplot)
Has the solution converged?
no
write out data (parallel HDF5)

Figure 3.4: Solution schematic

3.3 IBM methodology

3.3.1 Immersed Boundary Method fundamentals

The IBM method implemented here is described well in the papers of Ghias et.al. [31], Nam et.al. [41] and finally the IIBM improvements by Chi et.al. [30] and those three papers underpin the present work. A summary of implementation will be given here as well as the necessary modifications to use the IBM method with the ADI scheme.

In general, any IBM simulation begins with creation of two grids; structured background grid on which the solution is performed and unstructured grid of an arbitrary geometry of interest. The grid generation steps can be done in any standard meshing package and the packages need not be the same for each task. Then nodes inside/outside of the geometry are identified by the preprocessing algorithm and solid (inside) nodes within computational stencil of the nearest fluid nodes are marked as Ghost Nodes. Normal distances from the Ghost Nodes to the unstructured boundary are computed. This also allows computation of normal vectors to the boundary, required to impose Boundary Conditions correctly. Further Image Points (IP) are computed by extending via normal vector inside the fluid domain. If the IP does not fall completely within a cell made entirely of fluid cells, distance is extended until it does. This forms the basis of the Improved IBM as opposed to the classic Ghost Cell IBM where non-complete cells are allowed and accounted for as will be explained ahead. Interpolation coefficients for each IP are then computed so that field variables can be quickly interpolated to IP during the iterations. All the values are written out as Hierarchical Data Format 5 (HDF5) for the solver which then reads all the data and meshes and iterates as outlined in previous chapters. Given the overall process there are several important algorithms that must be devised, implemented and tested as follows.

To interpolate from 4 neighbouring points on the Structured Grid to the Image Point simple bilinear interpolation is used:
ϕ_n $(n = 1,2,3,4)$ are the 4 neighbours. ϕ is be a flow variable to be interpolated, pressure, velocity etc., C_n are the interpolation coefficients to be found and x, y, z are spatial coordinates of a node.

$$\left[\frac{\delta\phi}{\delta n}\right]_{BI} = C_1\left(y_{BI}n_x + x_{BI}n_y\right) + C_2\left(n_x\right) + C_3\left(n_y\right) + C_4 \tag{140}$$

$$\phi(x,y) = C_1(xy) + C_2(x) + C_3(y) + C_4 \tag{141}$$

$$\tag{142}$$

There are 4 sets of (x_n, y_n) around the Image Point, therefore:

$$\begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{Bmatrix} = \begin{bmatrix} x_1y_1 & x_1 & y_1 & 1 \\ x_2y_2 & x_2 & y_2 & 1 \\ x_3y_3 & x_3 & y_3 & 1 \\ x_4y_4 & x_4 & y_4 & 1 \end{bmatrix} \begin{Bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{Bmatrix} \tag{143}$$

And of course:

$$\left\{C\right\} = \left[V\right]^{-1}\left\{\phi\right\} \tag{144}$$

Let us define a_nn as the solution to matrix V:

$$\left[V\right]^{-1} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} x_1y_1 & x_1 & y_1 & 1 \\ x_2y_2 & x_2 & y_2 & 1 \\ x_3y_3 & x_3 & y_3 & 1 \\ x_4y_4 & x_4 & y_4 & 1 \end{bmatrix}^{-1} \tag{145}$$

$$\begin{Bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{Bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{Bmatrix} \tag{146}$$

Now C's are a function of ϕ's, they will change each iteration as variables update. $\left[V\right]^{-1}$ however won't change and can be precomputed based on grid and geometry before solution. This step, however simple, saves a great deal of computation over time.

Then

$$C_1 = a_{11}\phi_1 + a_{12}\phi_2 + a_{13}\phi_3 + a_{14}\phi_4 \tag{147}$$

$$C_2 = a_{21}\phi_1 + a_{22}\phi_2 + a_{23}\phi_3 + a_{24}\phi_4 \tag{148}$$

$$C_3 = a_{31}\phi_1 + a_{32}\phi_2 + a_{33}\phi_3 + a_{34}\phi_4 \tag{149}$$

$$C_4 = a_{41}\phi_1 + a_{42}\phi_2 + a_{43}\phi_3 + a_{44}\phi_4 \tag{150}$$

Substituting $C's$ into ϕ to compute value at the Image Point: (Capital I for Image Point)

$$\phi(x_I, y_I) = C_1(x_I y_I) + C_2(x_I) + C_3(y_I) + C_4 \tag{151}$$

$$\phi(x_I, y_I) = (a_{11}\phi_1 + a_{12}\phi_2 + a_{13}\phi_3 + a_{14}\phi_4)(x_I y_I) \tag{152}$$

$$+ (a_{21}\phi_1 + a_{22}\phi_2 + a_{23}\phi_3 + a_{24}\phi_4)(x_I) \tag{153}$$

$$+ (a_{31}\phi_1 + a_{32}\phi_2 + a_{33}\phi_3 + a_{34}\phi_4)(y_I) \tag{154}$$

$$+ a_{41}\phi_1 + a_{42}\phi_2 + a_{43}\phi_3 + a_{44}\phi_4 \tag{155}$$

Grouping by a variable ϕ_n

$$\phi(x_I, y_I) = \phi_1(a_{11}x_I y_I + a_{21}x_I + a_{31}y_I + a_{41}) \tag{156}$$

$$+ \phi_2(a_{12}x_I y_I + a_{22}x_I + a_{32}y_I + a_{42}) \tag{157}$$

$$+ \phi_3(a_{13}x_I y_I + a_{23}x_I + a_{33}y_I + a_{43}) \tag{158}$$

$$+ \phi_4(a_{14}x_I y_I + a_{24}x_I + a_{34}y_I + a_{44}) \tag{159}$$

Defining the final interpolation coefficient α for simplicity:

$$\phi(x_I, y_I) = \sum_{i=1}^{n=4} \alpha_i \phi_i \tag{160}$$

Where $\alpha's$ are

$$\alpha_1 = a_{11}x_I y_I + a_{21}x_I + a_{31}y_I + a_{41} \tag{161}$$

$$\alpha_2 = a_{12}x_I y_I + a_{22}x_I + a_{32}y_I + a_{42} \tag{162}$$

$$\alpha_3 = a_{13}x_I y_I + a_{23}x_I + a_{33}y_I + a_{43} \tag{163}$$

$$\alpha_4 = a_{14}x_I y_I + a_{24}x_I + a_{34}y_I + a_{44} \tag{164}$$

And the $\phi's$ are the solution retrieved from the neighbouring nodes after each iteration. $\alpha's$ are computed in Matlab before the solution begins, hence the solver doesn't have to explicitly have access to the boundary grid, only the cartesian solution grid. This is another very important reason for performing these pre computations as accessing the unstructured grid from a structured solver point of view would be expensive either memory wise or communication wise.

Then after the value at the Image Point is interpolated, there are two types of Boundary Conditions that can be applied at the Ghost Node to which the IP belongs:
Dirichlet:

$$\phi_{GN} = -\phi_{IP} \tag{165}$$

and Neumann:

$$\phi_{GN} = \phi_{IP} + \Delta l \psi \tag{166}$$

Where Δl is distance from the Image Point to the Ghost Node and ψ is the boundary condition applied for the given variable. For pressure and density it will most likely be

$$\psi_p = \psi_\rho = 0 \tag{167}$$

If using nonzero heat flux, it would be

$$\psi_{\dot{q}} = \dot{q_B} \tag{168}$$

For adiabatic B.C. either zero heat flux or fixed temperature T at the boundary can be used.
With the standard non-Improved GCIBM [31] a second look at these matrices is needed to address the following edge cases:

- One or more points C_n is located inside the solid, hence physical values of variables not available.
- Interpolation point is the Ghost Node itself.

These cases must be somehow handled or will result in undefined behaviour most likely leading to immediate divergence or at best strongly inaccurate solution. The points can be identified in pre-processing and are replaced with nearest points on the boundary and the boundary's coordinates as well as the value taken (variable at the boundary).
This was the first formulation implemented and while is not preferred for majority of simulations it can likely be useful in certain scenarios:

$$\begin{Bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{Bmatrix} = \begin{bmatrix} x_1y_1 & x_1 & y_1 & 1 \\ x_2y_2 & x_2 & y_2 & 1 \\ x_3y_3 & x_3 & y_3 & 1 \\ x_4y_4 & x_4 & y_4 & 1 \end{bmatrix}^{-1} \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{Bmatrix} \tag{169}$$

If the last point lies inside solid (order doesn't matter, just a convention to write it this way). Need to create two sets of modified matrices $[V]$, one for Neumann one for Dirichlet B.C.

Dirichlet interpolation matrix:

$$\begin{Bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{Bmatrix} = \begin{bmatrix} x_1y_1 & x_1 & y_1 & 1 \\ x_2y_2 & x_2 & y_2 & 1 \\ x_3y_3 & x_3 & y_3 & 1 \\ x_By_B & x_B & y_B & 1 \end{bmatrix}^{-1} \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_B = 0 \end{Bmatrix} \tag{170}$$

Neumann interplation matrix:

$$\begin{Bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{Bmatrix} = \begin{bmatrix} x_1y_1 & x_1 & y_1 & 1 \\ x_2y_2 & x_2 & y_2 & 1 \\ x_3y_3 & x_3 & y_3 & 1 \\ y_Bn_x + x_Bn_y & n_x & n_y & 0 \end{bmatrix}^{-1} \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \psi_B = 0 \end{Bmatrix} \tag{171}$$

Consequently, there will be two sets of α_i constants, one for Neumann one for Dirichlet. In the bilinear equation

$$\phi(x_I, y_I) = \sum_{i=1}^{n=4} \alpha_i \phi_i \tag{172}$$

Either $(\alpha_i)_{VN}$ or $(\alpha_i)_D$ will be used, depending whether ϕ is Dirichlet type BC (u, v, w, T) or Von Neumann type BC (ρ, p, q).

At each of node of the background structured grid the following quantities are exported from the preprocessing script to the solver:

- 2 x 4 sets of (i, j) locations - these will be used to specify from which nodes to access and retrieve the solution ϕ_i - this is 16 integers per node for a 2D case.
- 2 sets of α_i - this is 8 doubles
- Δl, r_{ghost} - length between IP and GN - one double. Not necessary initially until non-zero gradients at the wall are required.

This is 10 doubles and 16 ints for 2D case. On a 64-bit machine integer is 4 bytes, double is 8 bytes. Total of 144 bytes per grid point are necessary for the IBM in the main solver. This results in 140.625 mb of extra 'IBM' data per milion grid points needed. The considerations are important when dealing with GPUs as it is ideal if the entire simulation is stored inside dedicated GPU memory; time consuming memory transfers can otherwise occur and bottleneck the simulation.

With the Improved IBM, the constants should be set so that it is guaranteed that *all* interpolation points are fluid nodes in all geometric situations. The problem is shown on Figure 2.3 or Figure 3.5. The IIBM avoids incomplete interpolation and using two sets of matrices and was in fact a main motivation for the Improved IBM version by Chi [30]. This is achieved by comparing the distance of the Ghost Node to the boundary with a factor δ and increasing that distance for Image Point purpose as shown on image 2.3. By adjusting the constant appropriately it is possible to optimise so that complete

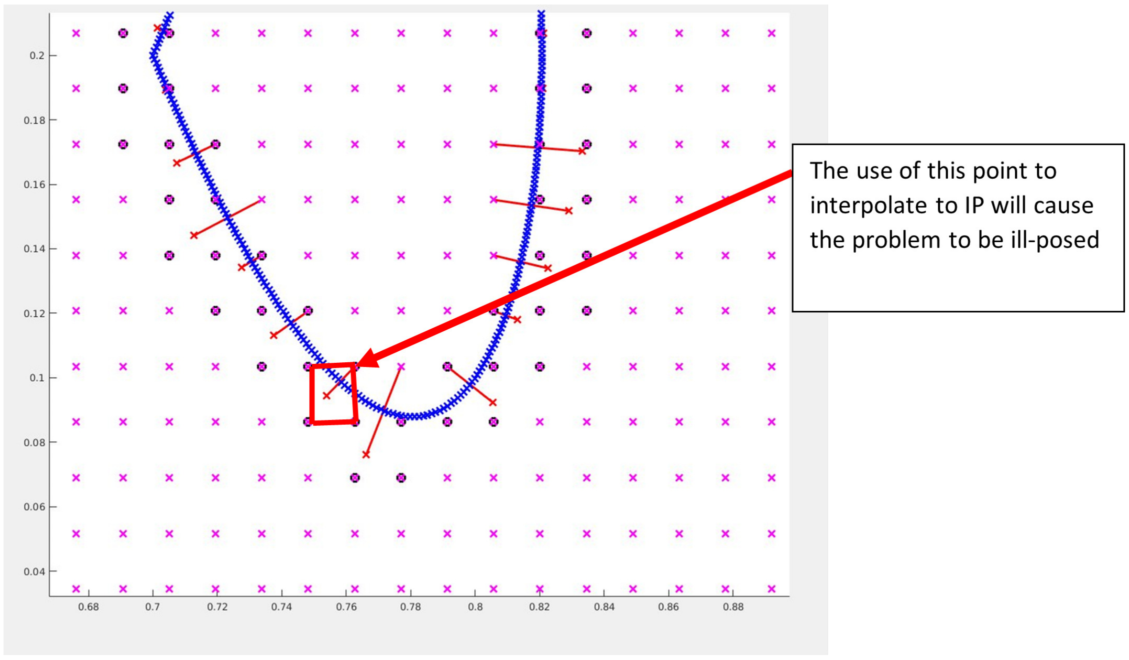

Figure 3.5: Demonstration of problematic point in GC IBM.

interpolation is achieved but the nodes do not lie too far away from the geometry, making the solution unstable and less accurate. The following factors were tested:

$$\delta = \phi\sqrt{2}\, max\,(\Delta x, \Delta y, \Delta z) \tag{173}$$

$$\delta = \phi\sqrt{2}\, min\,(\Delta x, \Delta y, \Delta z) \tag{174}$$

$$\delta = \phi\sqrt{2}\,\frac{\Delta x, \Delta y, \Delta z}{3} \tag{175}$$

Where ϕ is a case-dependent 'tweak' factor introduced by the author. It was found it improves the stability and varies from $0.8 - 1.2$ depending on the grid refinement used. It was also found that altogether different definitions of δ were suitable for different grid refinements and geometric combinations.

Good example of this issue is shown on Figure 3.6. It is evident from Figure 3.6 that definition of δ based on maximum cell extent would produce a very unstable IIBM set of coefficients as the majority of the interpolation points lie very far away from the boundary. Due to rapid changes of aspect ratio in this case, δ_{min} is the most suitable. For grids with more uniform aspect ratio however, average of even maximum δ is the most suitable as it produces coefficients well away from the boundary (which is desired) but not too far to cause instabilities. In case the grid does not fit into one of the three categories of δ (min, avg, max), the closest match is used and tweak factor ϕ adjusted away from the default value of 1 to achieve maximum stability. It is clearly a manual process for each grid and some automated checking could be introduced in the future to detect the grid and set the scheme appropriately.

The above procedure generally worked well however some deviation was required to achieve stable solution in certain situations. For instance average grid size rather than maximum was used if the standard deviation of cell size was large, or $\sqrt{3}$ factor was used to push the Image Points further into the fluid domain. The author did not manage to arrive at a consistent prescription that worked for all situations equally well. It would certainly be an area of further study and potentially an application for a Machine Learning algorithm to set factor δ individually in each cell based on past data.

Additionally the IIBM simulations with adjustment factor were only performed on a 2D plane here. It is expected that complexity of manually tweaking the δ factor will be significantly greater for a 3D simulation, as will likely be the stability issues.

Next, with IIBM, additional treatment is needed to handle the Extra Image Point resulting from the adjustment factor δ.

To impose Dirichlet B.C.: (assuming $\frac{\delta u}{\delta x} = 0$, non-zero gradients require additional treatment)

$$u_{ghost} = u_{EIP} - \left(1 + \frac{2r_{ghost}}{\delta}\right) u_{IP} \tag{176}$$

And Neumann B.C.:

$$u_{ghost} = u_{IP} - (u_{EIP} - u_{IP}) \frac{\delta - r_{ghost}}{\delta} \tag{177}$$

The procedure is explained in more detail in Chi et.al. [30].

In addition, in the vicinity of any IBM walls (Fluid Nodes with one or more Ghost Nodes in their stencil), special treatment can be applied, similarily to the vicinity of standard walls. Stencil can be modified to use one-sided differences or order of discretisation lowered near the boundaries. This is not strictly necessary and higher order formulations can be used however it was found it improves stability of the solution in near-wall IBM region very notably. *All* discretisation routines are modified and it is carefully ensured that the stencil uses valid nodes where physical values are present (i.e. nothing uses Solid Node values, where variables are undefined but memory is still allocated and accessible correctly at each rank).

One final practical issue to mention was when a Ghost Node distance to the nearest wall was very small relative to cell size as shown on Figure 3.7. Apart from this most likely leading to problems with the IIBM tweak factor δ it also caused non-insignificant inaccuracies in normal vector computation hence eventually the location of the Image Points for interpolation. The only satisfactory solution the author managed to devise for this problem under time pressure was local grid refinement to prevent this situation from occurring in the first place. An extra check was put in place in the code to detect it and the user was alerted if this problem arose. It is not ideal and adds an extra manual step but was a robust temporary workaround needed for development. A long term solution would be a double precision accurate normal computation and extra tweaks of the δ factor specifically designed to handle this situation.

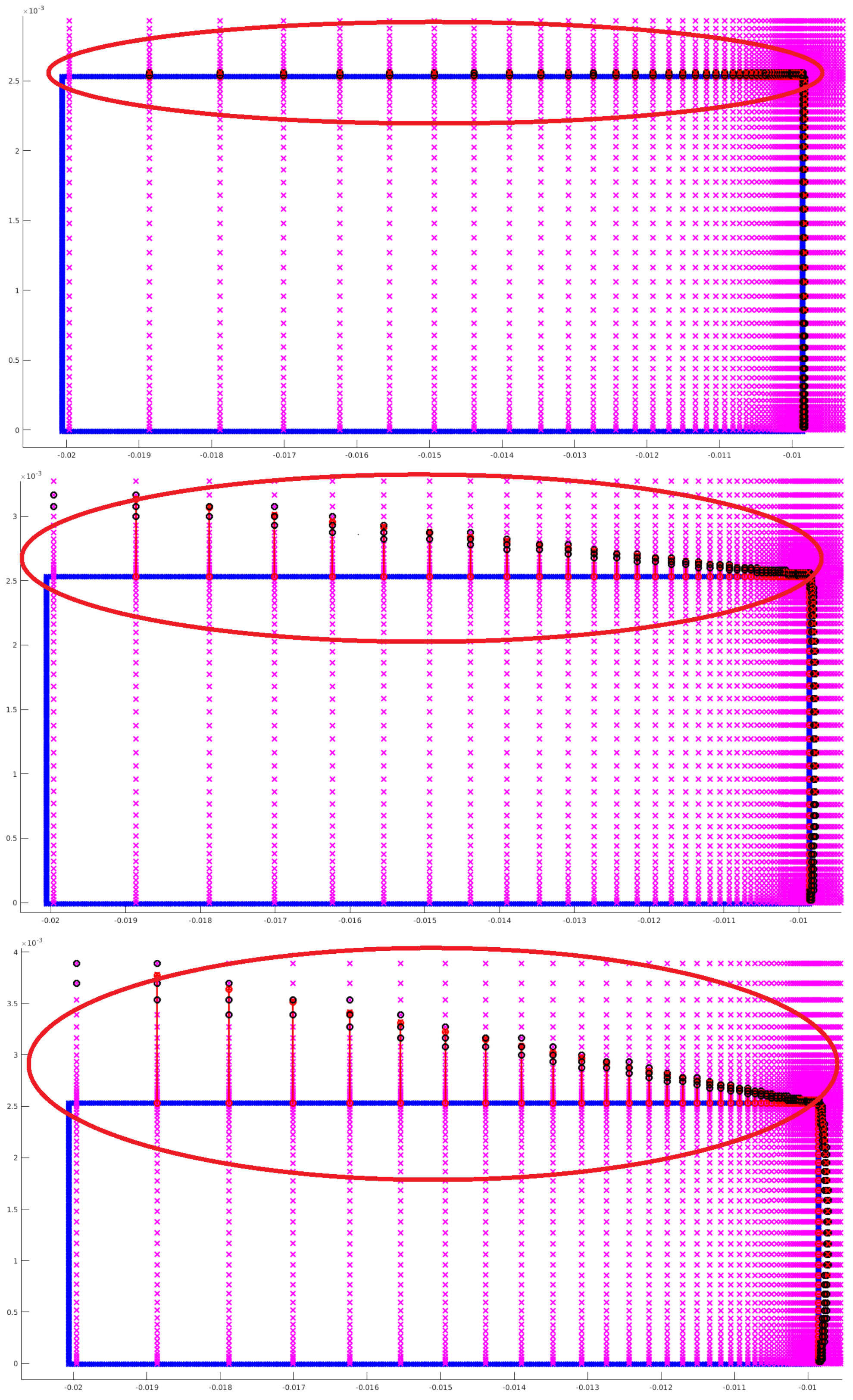

Figure 3.6: Tweak factor δ investigated. From the top, $\delta = min$, middle $\delta = avg$, bottom $\delta = max$. Not how the tweak factor makes the IBM interpolations use nodes much further from the boundary, leading to instabilities.

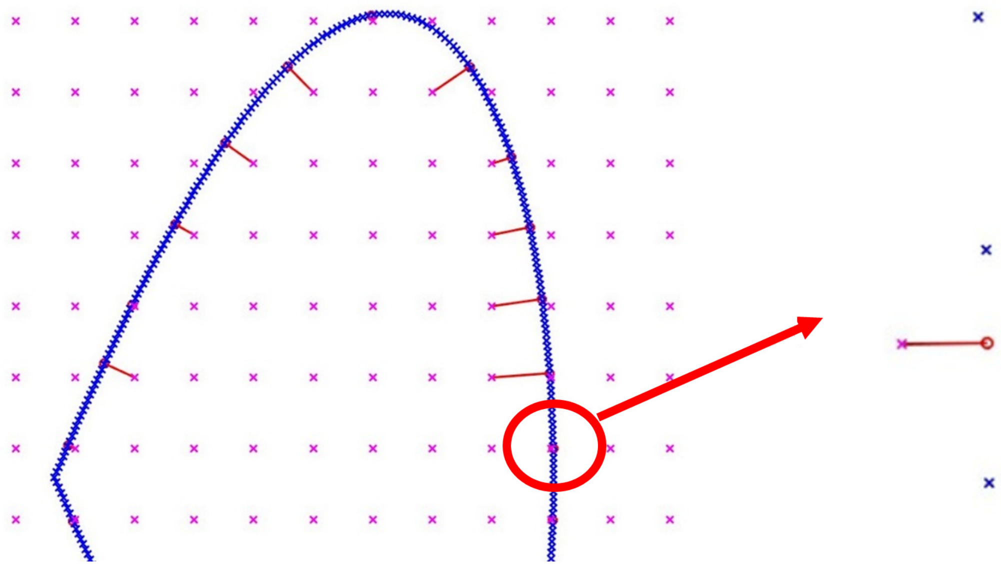

Figure 3.7: Demonstration of extremely small relative distance of Ghost Node to the wall.

3.3.2 Auxiliary relations

When performing unsteady calculations, it is still the mean and Root Mean Square (RMS) field that are usually of most interest. A routine of cumulative average was implemented to sample the flow at the end of each time-step and add the values to the overall average as follows:

$$\overline{\phi^{n+1}} = \frac{(n-1)\,\overline{\phi^n} + \phi}{n} \tag{178}$$

$$\overline{\phi^{n+1}} = \frac{(n-1)\,\overline{\phi^n} + \phi^2}{n} \tag{179}$$

$$\tag{180}$$

Where the first formula samples for mean, the second for mean of the sum of squares, required to compute the R.M.S. To prove correctness, consider $n = 1$, i.e. the first time step. The mean value is simply the current value of the variable. In subsequent time steps, the next values are weighted appropriately based on time step and added to the overall mean.

$$R.M.S. = \sqrt{\frac{a_1^2 + a_2^2 + a_3^2 + ... + a_n^2}{n}} \tag{181}$$

Running mean of sum of squares is kept in memory and square root taken just before writing out the results. The unsteady sampling can be switched on or off in the code, or set to sample after a user-specified value of time step.
Another way:

$$\overline{\phi^{n+1}} = \overline{\phi^n} + \frac{\phi_{n+1} - \overline{\phi^n}}{n+1} \tag{182}$$

3.3.3 Boundary Conditions for the Immersed Boundary Method

Although the boundaries of the inner shape of interest are imposed via the IBM method, the outer domain is defined by conventional boundary conditions. The B.C.s are defined as follows.

Inflow:

$$\frac{\delta p}{\delta n} = 0 \tag{183}$$

$$T = T_{input} \tag{184}$$

$$U = U_{input} \tag{185}$$

$$V = V_{input} \tag{186}$$

Outflow:

$$p = p_{input} = \frac{p_{ref}}{\rho_{ref} u_{input}^2} \tag{187}$$

$$\frac{\delta T}{\delta n} = 0 \tag{188}$$

$$\frac{\delta U}{\delta n} = 0 \tag{189}$$

$$\frac{\delta V}{\delta n} = 0 \tag{190}$$

Viscous walls:

$$\frac{\delta p}{\delta n} = 0 \tag{191}$$

$$\frac{\delta T}{\delta n} = 0 \tag{192}$$

$$U = 0 \tag{193}$$

$$V = 0 \tag{194}$$

Inviscid walls:

$$\frac{\delta p}{\delta n} = 0 \tag{195}$$

$$T = T_{input} \tag{196}$$

$$\frac{\delta U}{\delta n} = 0 \tag{197}$$

$$\frac{\delta V}{\delta n} = 0 \tag{198}$$

Finally translational periodic boundaries without pressure gradient can be imposed by simply copying all the fundamental variables between the corresponding edges. More complex periodic boundaries will need much more careful consideration and involve adding extra terms to N-S equations to balance out the 'looped' periodic energy, for instance as in Patankar [160] or the SST-SAS computations in Chapter 4, Figure 4.12. However during the initial development only the simplest translational periodicity will be considered for ease of debugging and verification.

3.3.4 Near wall scheme modifications

As more simulations were performed more edge cases arose and some nodes near the wall must be modified for the scheme to remain viable. Apart from the usual near-wall treatment where no nodes are available right at the grid edges for central differencing schemes there was the issue of Ghost Nodes located on the boundaries of the grid such as shown on Figure 6.13. This only happens if the IBM imposed geometry is directly connected to the edge of the structured mesh and the IBM geometry is directly perpendicular to the non-IBM wall. It is a very specific situation but it does happen with a certain regularity. The crux of the issue is how to handle the very edge nodes which have multiple designations: one ID coming from the wall and one coming from the IBM (i.e. the Ghost Node designation). Several ways to handle this were devised:

- keep original (non-GN) node designation and treat as wall inflow etc.
- force interpolation along a wall line and use effectively only two nodes not four to interpolate onto Image Points
- force the IP and EIP to shift slightly away from the wall, it will cause minor error but proved to be the most stable of all options.

The list is not exhaustive and there are likely other more numerically sophisticated methods of solving this issue however the last option proved to be sufficient for the time being.

3.4 Commercial codes FV methodology

3.4.1 SST – SAS

The primary model investigated to simulate the high Re, turbulent flow in the present paper was the Shear Stress Transport-Scale Adaptive Simulation (SST-SAS) [161]. From a mathematical point of view it was the standard $k-\omega$ URANS model with an extra Q_{SAS} term added to the specific turbulence dissipation rate ω equation:

$$Q_{SAS} = max\left[\zeta_2 \kappa S^2 \left(\frac{L}{L_{VK}}\right)^2 - C\frac{2k}{\sigma_\phi} max\left(\frac{|\nabla\omega|^2}{\omega^2}, \frac{|\nabla k|^2}{k^2}\right), 0\right] \tag{199}$$

Physically, the model and its functioning is discussed in more detail by the authors previously in [162] as well as by its creators Menter et.al. [161] [115] and various other authors mentioned previously. In essence, the Q_{SAS} term is based on ratio of turbulent length scale to the von Karman length scale and is only significant in regions of high strain and unsteadiness. Where the grid is refined sufficiently to resolve the flow as judged by the term, ω is increased, eddy viscosity is lowered and more of the energy spectrum should be resolved. In this case, once the resolving mode is triggered the underlying $k-\omega$ SST formulation begins to act as a subgrid scale model. Eddy viscosity is limited further by the use of the wall adapting local eddy viscosity (WALE) subgrid model. It is possible to simply use an artificial numerical viscosity instead of a subgrid model for LES [163] but this is another topic altogether and a standard approach of an explicitly defined subgrid model will be used.

3.4.2 Artificial forcing for the SST – SAS model

As the original SST-SAS formulation was deemed insufficient to allow the grid to resolve largest scale turbulent regions correctly, a synthetic turbulence generator was added in the hope to improve this. The formulation is based on the work of Kraichnan(1969), Batten(2004), Smirnov(2001) and Keating(2004) and was formally devised by Menter et.al. [161]. Primary motivation for trying this technique was that it could allow the cost effective SAS-SST to be used while avoiding the more expensive DES. If the synthetic turbulence generator does not improve the SST-SAS accuracy for a given application it would appear the SAS is not suitable to be used. Additionally, the formulation was implemented in ANSYS Fluent manually via the User Defined Function (UDF) functionality. This was not necessary in ANSYS CFX as the option was natively available.

As previously, the long handed, expanded version, will be presented. The formulation of the artificial forcing for the SST-SAS model is as follows: Let $factor = f$ be:

$$f = \sqrt{\frac{2}{3}k}\sqrt{\frac{2}{N}} \tag{200}$$

$$\vec{u}(\vec{x},t) = f\sum_{n=1}^{N}\left(\vec{p}\,\cos\left(arg^{n}\right)+\vec{q}\,\sin\left(arg^{n}\right)\right) \tag{201}$$

Where:

$$\vec{p}^n = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}^n = p_i^n = \varepsilon_{ijk}\eta_j^n d_k^n = \begin{bmatrix} \eta_2 d_3 - \eta_3 d_2 \\ \eta_3 d_1 - \eta_1 d_3 \\ \eta_1 d_2 - \eta_2 d_1 \end{bmatrix}^n \tag{202}$$

$$\vec{q}^n = \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix}^n = q_i^n = \varepsilon_{ijk}\xi_j^n d_k^n = \begin{bmatrix} \xi_2 d_3 - \xi_3 d_2 \\ \xi_3 d_1 - \xi_1 d_3 \\ \xi_1 d_2 - \xi_2 d_1 \end{bmatrix}^n \tag{203}$$

With $i = 1,2,3$, ε_{ijk} is the alternating Levi-Cita symbol and $\vec{\eta}^n$, $\vec{\xi}^n$, $\vec{d}^n$ and ω^n are random numbers (N = N(ϕ, ψ)) from 3D or 2D or scalar gaussian distribution of mean ϕ and standard deviation ψ. Number of modes is usually $N = 200$ and each mode has its own unique random set of values. n denotes mode, not the power of variable.

$$\vec{\eta}^n = \begin{bmatrix} \eta_x \\ \eta_y \\ \eta_z \end{bmatrix}^n ; \vec{\xi}^n = \begin{bmatrix} \xi_x \\ \xi_y \\ \xi_z \end{bmatrix}^n ; \vec{d}^n = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}^n \tag{204}$$

Mean and standard deviation are: $\eta_i^n = N(0,1)$, $\xi_i^n = N(0,1)$, $d_i^n = N(0,0.5)$, $\omega^n = N(1,1)$. Note ω^n is scalar, not a vector

arg^n is:

$$arg^n = 2\pi\left(\frac{\vec{d}^n\vec{x}}{L_t} + \frac{\omega^n t}{\tau_t}\right) = 2\pi\left(\frac{d_i^n x_i}{L_t} + \frac{\omega^n t}{\tau_t}\right) \tag{205}$$

$$= 2\pi\left(\frac{d_x^n x + d_y^n y + d_z^n z}{L_t} + \frac{\omega^n t}{\tau_t}\right) \tag{206}$$

Where $\vec{x}$ is the standard cartesian position vector and L_t, τ_t are length and time scale of turbulence coming from the RANS model given as:

$$L_t = C_L\frac{\sqrt{k}}{C_\mu \omega} ; \tau_t = \frac{L_t}{\sqrt{k}} \tag{207}$$

Where $C_L = 0.5$, $C_\mu = 0.09$, Δt is the timestep and $\Delta h = max(\Delta x, \Delta y, \Delta z)$ is the maximum cell extent in any direction.

Also Nyquist limiter is imposed on the length and time scales:

$$\tau_t = max\left(2\Delta t\ \omega^n, \tau_t\right) \tag{208}$$

$$L_t = max\left(2\Delta h\ |\vec{d}^n|, L_t\right) \tag{209}$$

Where $|\vec{d}^n|$ is magnitude of the wave number:

$$|\vec{d}^n| = \sqrt{d_i^n d_i^n} = \sqrt{d_x^{n2} + d_y^{n2} + d_z^{n2}} \tag{210}$$

This can now be expanded into the three components in 3D, or two in 2D:

$$u_x = u_x\left(\vec{x}, t\right) = f\sum_{n=1}^{N}\left(p_x^n\ \cos\left(arg^n\right) + q_x^n\ \sin\left(arg^n\right)\right) \tag{211}$$

$$u_y = u_y\left(\vec{x}, t\right) = f\sum_{n=1}^{N}\left(p_y^n\ \cos\left(arg^n\right) + q_y^n\ \sin\left(arg^n\right)\right) \tag{212}$$

$$u_z = u_z\left(\vec{x}, t\right) = f\sum_{n=1}^{N}\left(p_z^n\ \cos\left(arg^n\right) + q_z^n\ \sin\left(arg^n\right)\right) \tag{213}$$

Note arg^n is constant across directions but changes with modes.

Having computed $\vec{u}\left(\vec{x}, t\right)$ it can be used in the momentum source:

$$F_{x-mom} = \frac{\rho u_x}{\Delta t} \tag{214}$$

$$F_{y-mom} = \frac{\rho u_y}{\Delta t} \tag{215}$$

$$F_{z-mom} = \frac{\rho u_z}{\Delta t} \tag{216}$$

And turbulent kinetic energy:

$$F_{TKE} = -0.5\frac{\rho\left(\vec{u}\right)^2}{\Delta t} \tag{217}$$

Where $\vec{u}^2$ probably is:

$$\vec{u}^2 = u_x^2 + u_y^2 + u_z^2 \tag{218}$$

The algorithm in practice is as follows:

- generate $\vec{\eta}^n$; $\vec{\xi}^n$, $\vec{d}^n$ and ω^n with given mean and standard deviations
- compute $\vec{p}^n$ and $\vec{q}^n$ from above
- compute the time and length scale of turbulence with the Nyquist limiter
- compute arg^n for a given mode
- compute the velocity fluctuation $\vec{u}$ components u_x, u_y and u_z with all the above
- apply to momentum and TKE equation as constant source terms

3.4.3 Assessment of resolution of RANS-LES methods

One of the primary methods of quantifying the resolved content of a hybrid RANS-LES simulation (or any simulation really, it is equally applicable) is concept of two-point correlation as described in the work of Gaitonde et.al. [164]. The paper has good overall description of the method used here and provides more detailed mathematical formulation.

There are two main ways two point correlation can be done:

- 1) spatial two-point correlation
- 2) temporal self-correlation

Technique 1) involves sampling a set of equidistant points over time and creating a time-averaged product between the starting point and all the subsequent points as explained below but also in Gaitonde et.al. [164]. A 'correlation' factor is then obtained for each point with respect to the starting location. The plotted set of factors then allows to assess how strongly the flow is correlated in space hence how much of the recirculating region has been explicitly captured by the grid.

Technique 2) involves sampling only a single point in space over a long period of time to capture both the high and low frequency temporal oscillations. By fourier analysis and processing of this data, an energy spectrum can be plotted and compared against standard LES-like distributions. This allows a quantified comparison between the flow in question and a 'typical' resolved simulation in a given application. For practical purposes all points used for 1) will have their self-correlation computed but also three additional points in other parts of the domains will be used to cross check for any dis-

crepancies. It is worth noting that the points choice for self correlation is just as important as the set chosen for two-point technique - the location must be in the appropriate region of interest as this is where the 'resolved' content will be measured.
Both methods, while quantitative, carry a range of assumptions and errors. For instance, homogenous turbulence is assumed and the choice of sampling points for both 1) and 2) can strongly influence the result. While majority of turbulence is indeed homogenous and multiple sets of points can be sampled to get an idea of variations these are inaccuracies that cannot be avoided. Nevertheles the two-point correlations are widely used in the hybrid RANS-LES community and the quantitative result provides excellent base for comparison between other researchers' simulations.
Spatial two-point correlation as in 1) will now be explained:

Let A and B be two points in space and:

$u = \overline{u} + u'$

$\overline{u_A}$	mean velocity component at point A
$\overline{u_B}$	mean velocity component at point B
u'	fluctuating velocity component
u	instantaneous velocity

The two-point correlation is the time-averaged product of instantaneous velocities at two points. Velocities do not have to be the same, i.e. we might have u_A correlated with v_B, or any variable not necessarily velocity

$$R_{AB} = \overline{u_A u_B} = \overline{\left(u_A + u'_A\right)\left(u_B + u'_B\right)} = \overline{\overline{u_A u_B} + 2u_A u'_B + u'_A u'_B} \tag{219}$$

But

$$\overline{u_A u'_B} = 0 \tag{220}$$

So:

$$\overline{\left(u_A + u'_A\right)\left(u_B + u'_B\right)} = \overline{\overline{u_A u_B} + u'_A u'_B} \tag{221}$$

The procedure is as follows:

- define a line in space, based on two points
- create N equispaced points
- first point on a line (on either end, but it must be chosen) is going to be the reference $u_A = u_{ref} = u_0$and all the other values (u_i where i is point number along the line) are going to be with respect to that point
- On each of the points, monitor and export the variables of interest to a file during the transient run (full time history along a line)
- Process as follows
  - obtain product of the two variables of interest $u_A u_B$ at each time step at each point where always $u_A = u_{ref} = u_0$
  - obtain mean of the products $\overline{u_A u_B}$ at each point i
  - obtain means of each variable along a line individually $\overline{u_0}, \overline{u_1} ... \overline{u_n}$
  - obtain product of the averages $\overline{u_0}\ \overline{u_i}$ at each point, keeping u_0 as reference always
  - the two-point correlation is: $\overline{u_A u_B} - \overline{u_0}\ \overline{u_i}$
  - normalise the correlation with respect to the first point so it starts with 1

The energy spectra used for self-correlation metric in resolving simulations mathematically are a Power Spectral Density (PSD) which estimate the energy distributions at each frequency of simulation output (in this case x,y or z velocities). The spectra were created using MATLAB's in built *pwelch* function [165], with the sample rate Fs being the unchanging time step of simulation. There are other techniques which manually obtain and process a Fourier transform however the *pwelch* function was found to be ideally suitable for the present purpose containing all the postprocessing required here.

3.5 Chapter conclusions

An attempt to provide complete and detailed set of equations, procedures and approaches was made. Many of the well-known numerical tools were only referenced to avoid duplication as excellent and clear sources of documentation already exist. All the key methods and equations were stated in as-implemented form for clarity but also ease of debugging as the work progresses. It is hoped this chapter will provide a reference for other researchers developing their own CFD code in the future.

Chapter 4

4 Results: SST-SAS model testing

4.1 Introduction

On top of the computational issues explored so far there is also a matter of accuracy. The question is posed whether the necessary level of accuracy can be achieved with lower computational demand, e.g. coarser grid leading to lower cell count, simpler turbulence model leading to fewer or less computationally demanding equations. This chapter presents testing of a new approach termed Scale Adaptive Simulation and its most common and initial variant based on $k-\omega$ SST model. The overall idea is that one can refine the grid in the regions where LES-like resolving of smaller scales is required and keep all other areas in URANS mode. There is no dependency on wall proximity in the equations or other necessities typical of fixed boundary RANS-LES hybrids, such as treatment of the RANS-LES interface to preserve energy correctly. The aim is to achieve required accuracy in critical regions and minimise the computational cost. Only the results are presented here; more detailed discussion of the SST–SAS model as well as other techniques used in this chapter can be found in Section 3.4.2, 3.4.3 and 3.2.12.

This chapter structure has three key components. Firstly, 3D axisymmetric rib and ribbed channel geometries are used to initially assess the SST–SAS model against RANS, URANS, LES and DES and establish a baseline comparison for further study - aim was to highlight the differences between SAS–SST and other standard hybrid resolving models and identify areas of focus for further research. The second part is an investigation into flow inside square ribbed channel with 9 ribs with and without artificial forcing or rotation. More detailed assessment of the differences with other models is made with this channel, including standard LES metrics such as two point correlations and energy spectra. The work is largely driven by findings of the first part. The last part of this chapter presents simulations of flow inside a more realistic blade. It is a strongly turbulent and three dimensional flow with inlets positioned at various irregular locations. It is believed the high levels of turbulence and strong three dimensionality and asymetry of the boundary conditions should be sufficient to trigger the resolving of scales in the SAS–SST model with respect to a URANS model. Should the smaller scales not appear there seems to be little hope for the SAS–SST model in the present internal cooling application as few applications will exhibit stronger turbulence.

While the test cases appear simple in geometry, they were chosen carefully to contain the key flow features commonly present in turbomachinery. The test cases are also known to expose majority of flaws and inaccuracies in numerical schemes without the complexities of real geometries. Using the latter in initial stages of code development not only slow down the progress but can sometimes hide issues as complex effects cancel each other out.

Four peer-reviewed publications resulted from this work. Two papers covering the first two parts of the chapter and two papers investigating the header component. The latter was collaborative work and the CFD model was used as input to novel fatigue models.

4.2 Numerical details

Two solvers were used to obtain the present results: ANSYS Fluent v17.2 and ANSYS CFX v17.2. The simulations without artificial forcing were performed in ANSYS Fluent using the incompressible Finite Volume cell-centred solver with implicit bounded second order time formulation. The Semi-Implicit Method for Pressure Linked Equations (SIMPLE) was used for pressure-velocity coupling. No upwind (smoothing) method was used and all equations were spatially discretised with 2nd order formulations. The bounded second order modification was used for momentum equations as it was found to produce more realistic results and was also recommended for use with the SST-SAS model [166].

Simulations including the forcing were performed in ANSYS CFX with an incompressible finite element-based finite volume formulation and a vertex-based discretisation strategy as explained in detail in [167]. Rhie-Chow [168] coupling was used for the pressure-velocity link as the pressure and velocity is collocated. All equations were discretised with second order accuracy formulations without special treatment.

The SAS-SST forcing term Q was implemented in ANSYS Fluent 17.2 via the use of User Defined Function, however it was found by testing that ANSYS CFX has a faster and more robust implementation of the forcing and differs from the authors own implementation only in the number of harmonic modes used. For this reason, the latter was used for simulations with forcing.

All simulations were performed on the University of Nottingham HPC and utilised 64 to 112 cores. The time step range used in the simulations for all unsteady models is 0.00014s to 0.00012s for the most refined grid. This corresponds to mean Courant number of 0.5-0.8 in the critical regions of interest around the ribs and 1.6 in the streamwise centre of the channel, where the SST-SAS model is expected, and observed, to operate in the modelling mode. Small regions with higher courant number exhibited no unsteadiness or turbulence as they were far away from the ribs. After developing the flow for at least three through flows based on mean inlet velocity, arithmetic averaging was performed on the transient data for another three to five through flows minimum.

Convergence was achieved in both solvers by ensuring that the RMS of residuals drops by at least three orders of magnitude after the time step was incremented. This was achieved in 3-6 iterations per time step.

The guidelines used for setting the cell edge lengths for the SST–SAS model were taken from Davidson[169] [118] and proved to be very similar to recommendations for DES grid design by Spalart [170]. Given that scale resolving SAS is regarded as a generalisation of DES it makes sense for the guidelines to be similar and applicable. For completeness, $\Delta x^+ \approx 100-300$ and $\Delta z^+ \approx 100-600$ were used as recommended. For instance, the current 3D hill grid sizes can be compared to grids of up to 9.6 million cells used in LES simulations by Tessicini *et.al.* [171], 1.7 million cells by Davidson [169] and also 1.7 million cells by Rodi *et.al.* [172]. For completeness the periodic channel grids used by Liu [173] are 0.3 to 3 million cells.

The time step was optimised based on the CFL criterion and varied during the grid sensitivity study. Δt varied from $3.5\times10^{-5}s$ to $8\times10^{-5}s$ for the 3D hill and $1.6\times10^{-5}s$ to $2.9\times10^{-5}s$ for the ribbed channel. Values of $7.3\times10^{-5}s$ and $2\times10^{-5}s$ were used for the final grids resulting from sensitivity studies for the hill and rib, respectively. Three sub iterations per time step were found sufficient in the present study for both the 3D hill and the ribbed channel, based on convergence histories of multiple runs. Four sub iterations were used to enforce the streamwise periodicity for the ribbed channel case.

All transient simulations were started from an appropriate precursor RANS simulation and ran for at least two through-flows based on freestream velocity to develop the flow. Data was then cumulatively sampled for three through flows or until the averaged quantity stabilised, whichever was longer.

Several more parameters were monitored as standard for all simulations in addition to residuals: the streamwise component of velocity, turbulent kinetic energy and ω where available. The reason for monitoring ω is that it is critical to the SAS-SST model as the ω-equation is where the Q_{sas} term is added.

4.3 3D axisymmetric hill

4.3.1 Computational setup

The computational setup for the 3D hill employed here is similar to that of Davidson *et.al.* [118] [169]. The geometry used is based on that investigated experimentally by Simpson *et.al.* [1] [174] and Davidson *et.al.* The hill is axisymmetric and the setup is three-dimensional. The hill height, H, is 78mm and is used as the normalizing factor for most of the plots. The shape of the hill is defined by:

$$y(r) = -H\frac{1}{6.04844}\left[J_0(A)\,I_0\left(A\frac{r}{a}\right) - I_0(A)\,J_0\left(A\frac{r}{a}\right)\right] \tag{222}$$

where $y(r)$ is the hill height and is shown plotted on the vertical axis of Figure 4.1. The value of A is constant and equal to 3.1926; $a = 2H$ is the radius of the base of the hill. The radius of the hill base is equal to twice the height as indicated on Figure 4.1. J_0 is the Bessel function of the first kind and I_0 is the modified Bessel function of the first kind. The constant ν used in both the Bessel functions was set to zero. The origin of the coordinate system is located on the axis of the hill, at height $y = 0$, as demonstrated on Figure 4.1. X is the streamwise coordinate, z is the cross-stream coordinate and y is the wall-normal coordinate.

The full computational domain is $11.6H \times 3.2H \times 19.7H$, which is the same as in Davidson [169]. The inlet is located $4.1H$ ahead of the hill axis along X coordinate. Similar computational domains were used by Tessicini & Leschiziner [171] (domain $3.7H$ shorter in x-direction), Rodi [172] ($0.3H$ longer in x-direction), Garcia-Villalba *et.al.* [172] and Persson [175] ($7.7H$ shorter in x-direction). Several other lengths were investigated by the mentioned authors and it was concluded by Tessicini [171] that approximately $19.7H$ is likely the most optimal length for computational analysis.

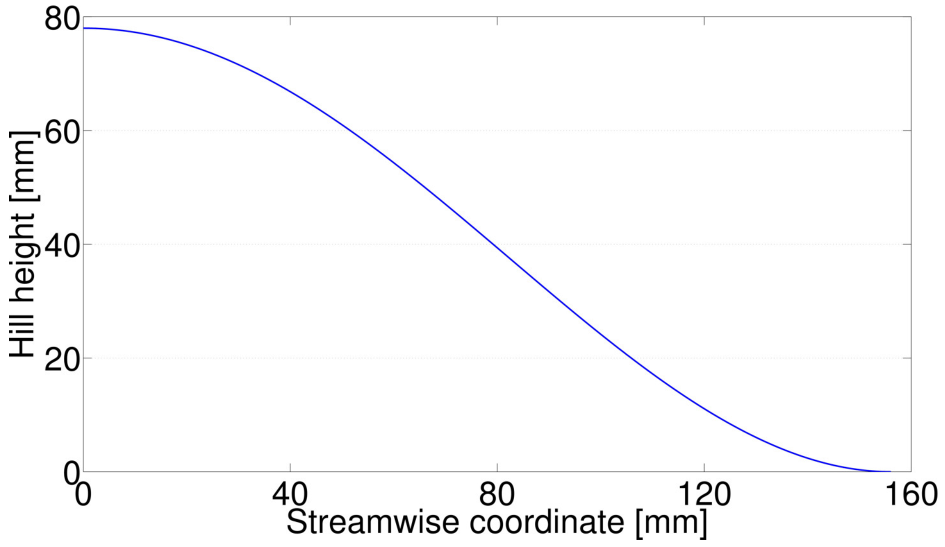

Figure 4.1: Outline of the curve used to create the hill geometry

Grid	Size	Δx^+ range	Δz^+ range
$154 \times 76 \times 197$	2.3m	100-200	100-200
$178 \times 82 \times 226$	3.3m	80-160	80-150
$193 \times 90 \times 248$	4.3m	60-135	70-140
$223 \times 100 \times 294$	6.5m	50-110	60-120

Table 4.1: Grids used to establish mesh sensitivity for 3D hill. Average $y^+ = 1$ for all grids. Δz^+ and Δx^+ are $y^+ = 1$ equivalent concepts in the other two dimensions useful for measuring the grid with respect to the flow.

ICEM CFD software was used to create all the grids in present study. Figure 4.3 shows one of the grid configurations used with every second grid point hidden for clarity. Table 4.1 summarises the grids used in the mesh sensitivity study for the 3D hill. A wall value of $y^+ \approx 1$ through the domain was achieved with an average first cell size of $y = 0.025mm$ and this value was used for all 3D hill grids. Typical grid densities used in industry for turbine blade problems range from 0.5 million cells to 3 milion cells for a steady calculation of a single turbine blade at the time of writing.

The boundary conditions are set as in [172], [171] or [118]. The z-direction normal walls are slip walls (zero shear stress prescribed). The outlet is set to the Von Neumann condition with the gradient of all variables set to zero; including pressure. Left and right (Z-normal) walls are standard slip walls.

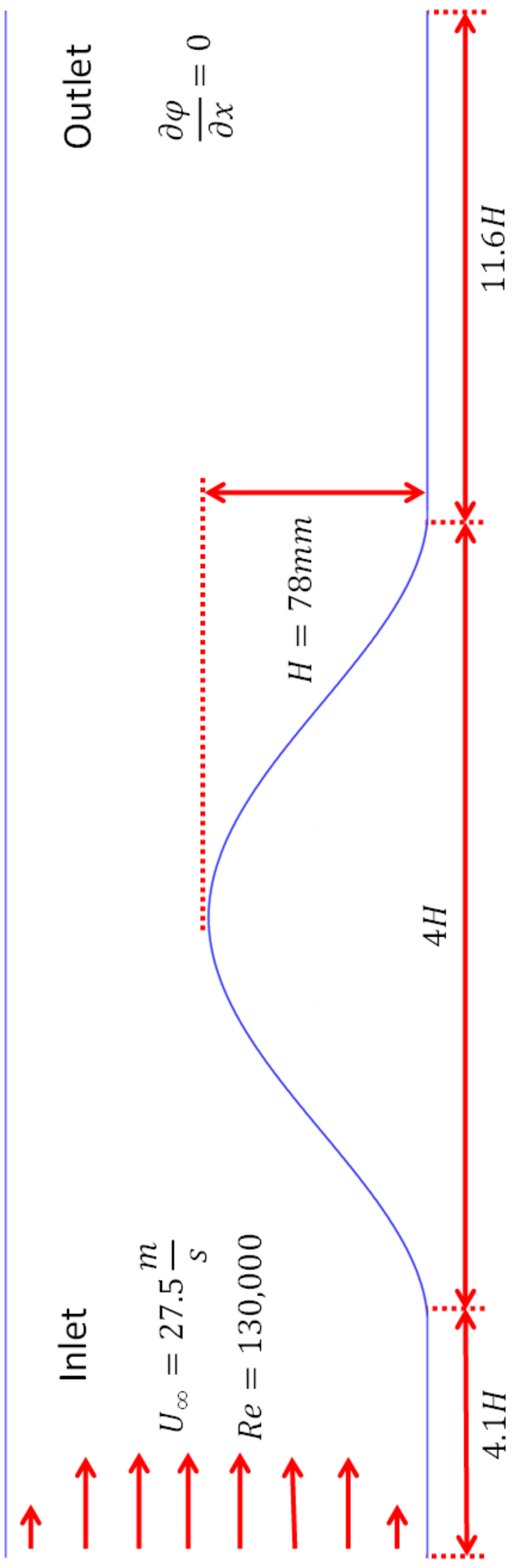
Inlet
$U_\infty = 27.5\frac{m}{s}$
$Re = 130{,}000$
Outlet
$\frac{\partial \varphi}{\partial x} = 0$
$H = 78mm$
$4.1H$
$4H$
$11.6H$

Figure 4.2: Schematic of the boundary conditions used for 3D axisymmetric hill.

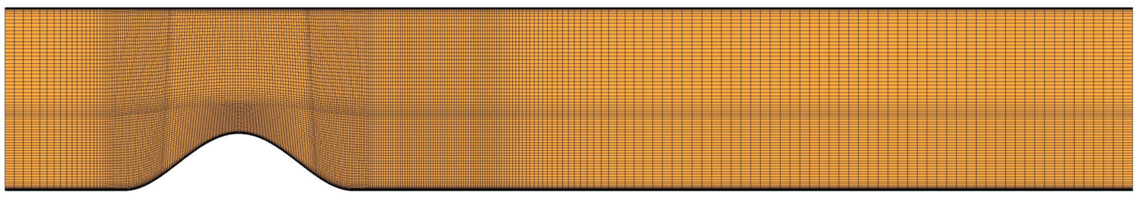

Figure 4.3: Streamwise cross-section of 3D hill mesh resulting from sensitivity study; every second grid point is shown

The inlet for the 3D hill is the most challenging and important boundary here, especially given the inherently unsteady nature of the flow. The approach boundary layer has been experimentally found to be of thickness $\delta = 0.5H$ and was initially set to match the 1/7th law without any time-varying quantity. The freestream velocity used for the 1/7th law is 27.5 m/s as measured by Simpson [1]; it was also confirmed in the study that the actual flow matches the profile well. The Reynolds number based on hill height and freestream velocity is 130,000.

In order to fully evaluate the SAS model another approach used in this study was to impose velocity fluctuations on top of the 1/7th profile. The method chosen was the synthesized turbulence as defined by Kraichnan [176] and modified by Smirnov *et.al* [177]. As will be later explained there is slight improvement in using the unsteady inlet boundary condition however the sensitivity was found to be low for SST-SAS simulations and consequently a constant velocity profile was applied at the inlet for most simulations.

Mean turbulence intensity at inlet was found experimentally [1] to be 0.1% and this value is used here. A value for viscosity ratio of approximately 10 is recommended by Tessicini [171] and Rodi [172]. It is also judged that viscosity ratio should not affect the result strongly as the turbulence intensity is very low.

Figures 4.4 and 4.5 show the key results used to judge grid dependence. While the streamwise velocity component shows relatively strong grid independence, spanwise velocity component indicates that at z/h=-0.16 the finest grid of 6.5 million cells results in higher accuracy than the other grids, however the difference is significant on only one of the lines where data was extracted. For prediction of streamwise velocity field the finest grid made negligible difference, apart from at z/h=-0.16. The mesh with 2.3m cells was chosen for all further investigation based on this study.

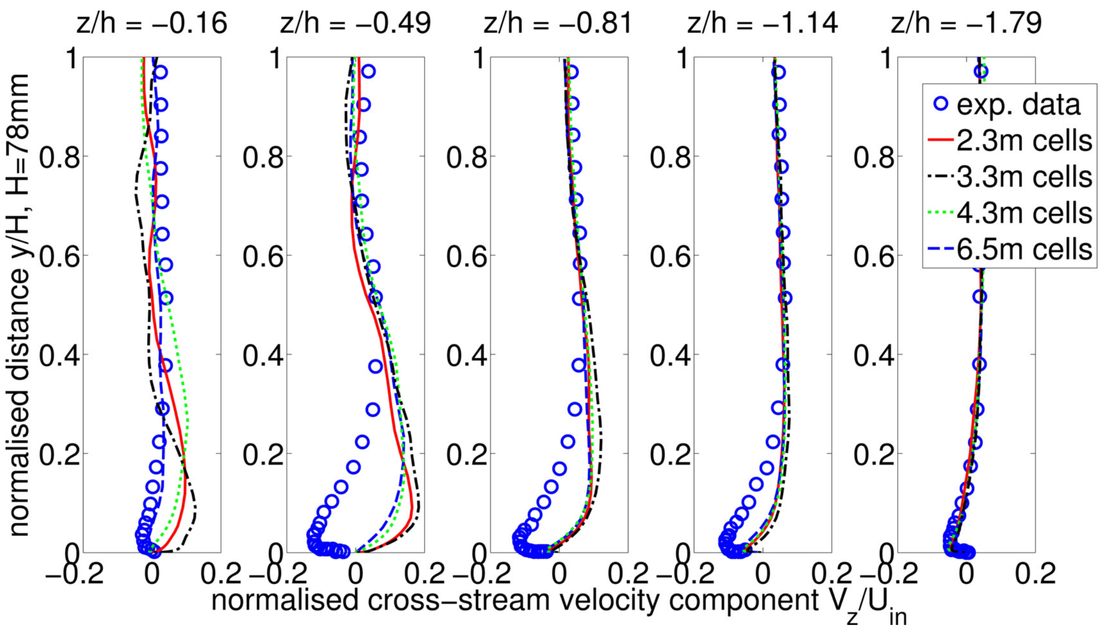

Figure 4.4: Spanwise component of velocity, 3D hill. $x/H = 3.69$.

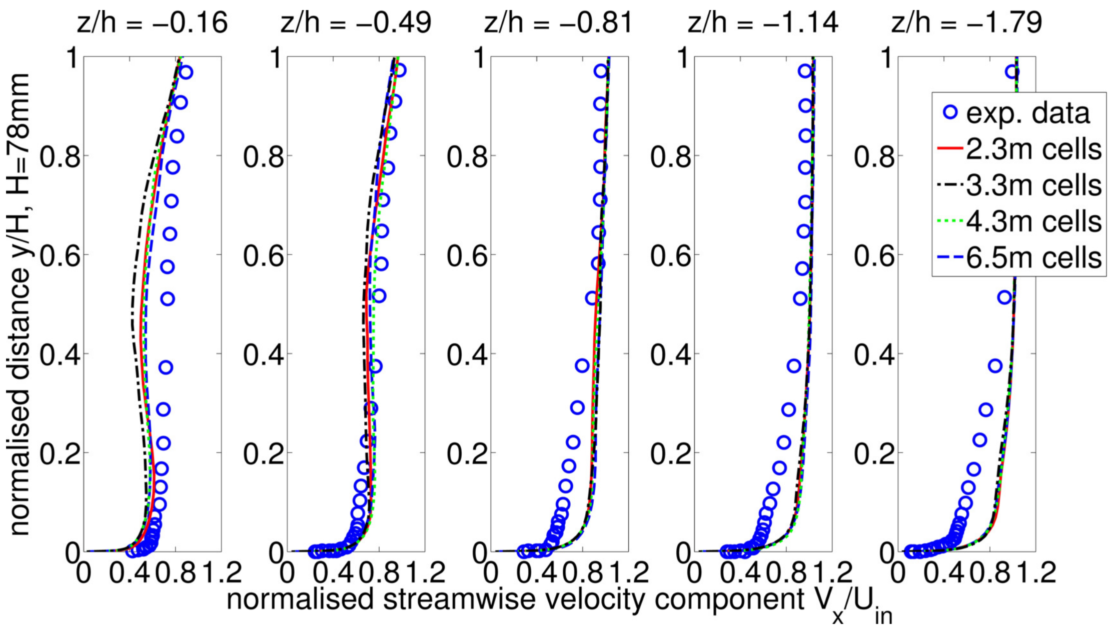

Figure 4.5: Streamwise component of velocity, 3D hill. $x/H = 3.69$

4.3.2 Results

There is comparison and validation data available for the present study, both computational and experimental. Figure 4.6a shows the vector field obtained experimentally by Simpson [1] and this can be compared to the SAS CFD data from the current study shown in Figures 4.6(b) and (c). As is immediately apparent there are significant differences between the two with the SAS model showing a large recirculation region not present experimentally.

Figure 4.7 and Figure 4.8 show cross-stream and streamwise velocity profiles obtained along side the experimental data of Simpson [1] and the computational results from Davidson & Dahlstrom [169] [118] obtained using hybrid LES-RANS. Both Figure 4.7 and Figure 4.8 also show the LES data is significantly closer to the experimental data than other computations run in the present study. It is both surprising and noteworthy that even with a very coarse grid the LES model appears to be much closer to experimental measurements than any of the others investigated including SAS. It is also shown by Tessicini *et.al*. [171] that even on grid of 1.6 million cells (less than the 2.3m cells used here) LES is still significantly closer to the experimental data than the SAS model in the current study. Pure LES however is not as accurate as the hybrid LES-RANS method used by Davidson *et.al*. [118] and the line plots of Figure 4.7 and 4.8 confirm this finding.

Figure 4.9 shows time averaged streamlines predicted by different modelling approaches. The recirculation region behind the hill is predicted by SAS, RANS and DES (with $k-\omega$ model) with a very similar size in each case of approximately H. The size predicted by LES is close to experimental data; while no streamline plots are available from the experimental study, the vectors shown on Figure 4.6 show a relatively thin layer ($\approx 0.1H$) of recirculation, in agreement with LES. The grid used in this study was not designed for LES and is approximately two orders of magnitude coarser in terms of x^+ and z^+ than normally is required for LES computation.

The general structure of the flow and the main separation point are approximately similar for all the turbulence models used, as indicated by Figure 4.9. Surface streamlines shown on Figure 4.10 suggest two eddy-like structures, one on each side of the hill, in agreement with experimental data shown on Figure 4.11. The streamwise size of these eddy structures is predicted to be far larger than found experimentally by any model investigated other than LES. It is well established by all the authors [169, 112, 178, 171, 172, 179, 175] as well as by comparison with experimental data [1] here that the RANS approach completely fails to predict the flow in this case.

Another interesting numerical feature of the flow, although not found experimentally is the small "bubble" at the foot of the hill, upstream of the hill axis (see Figure

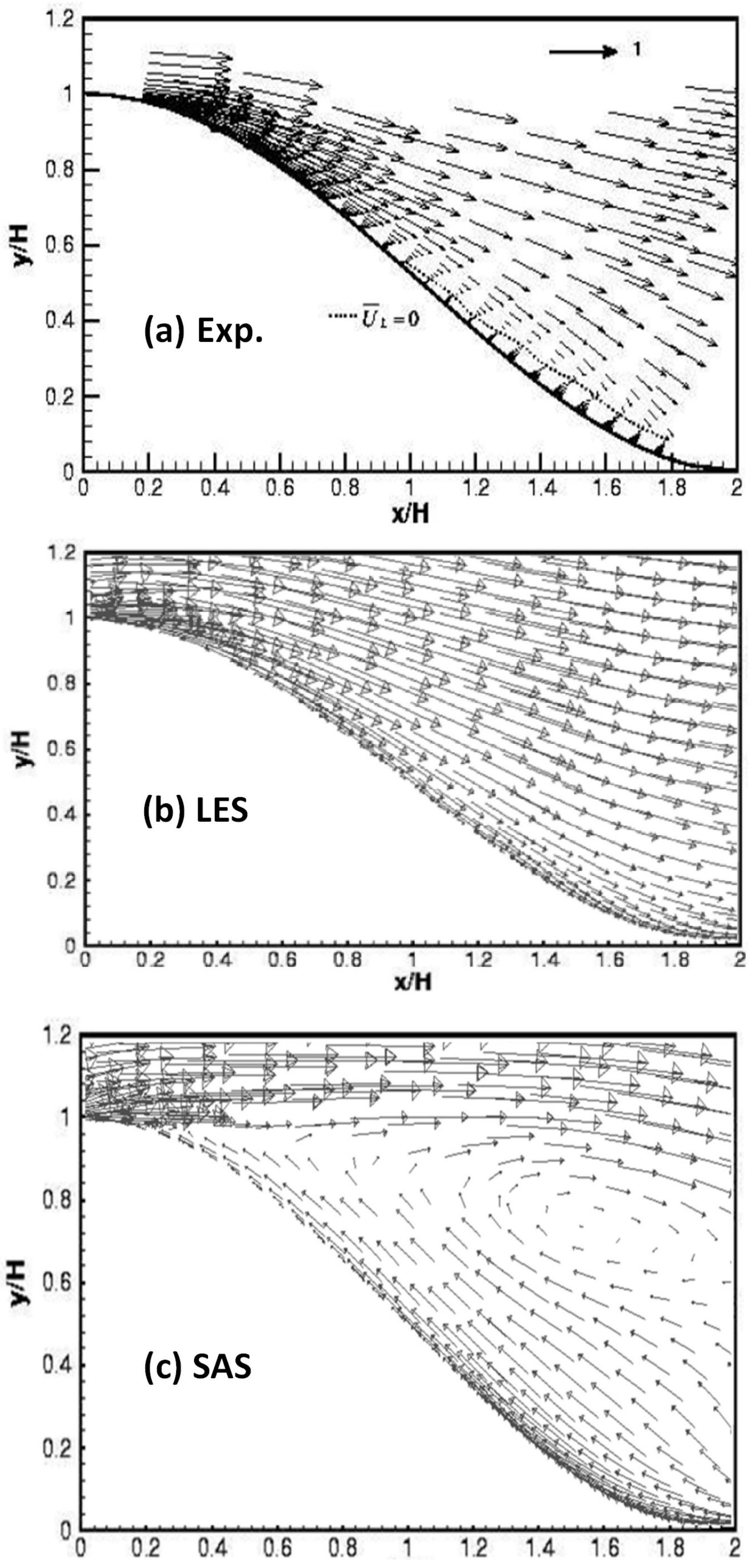

Figure 4.6: streamwise velocity component vectors, experimental data, Simpson $et.al.$[1] (a) Experimental data, (b) LES, (c) SAS

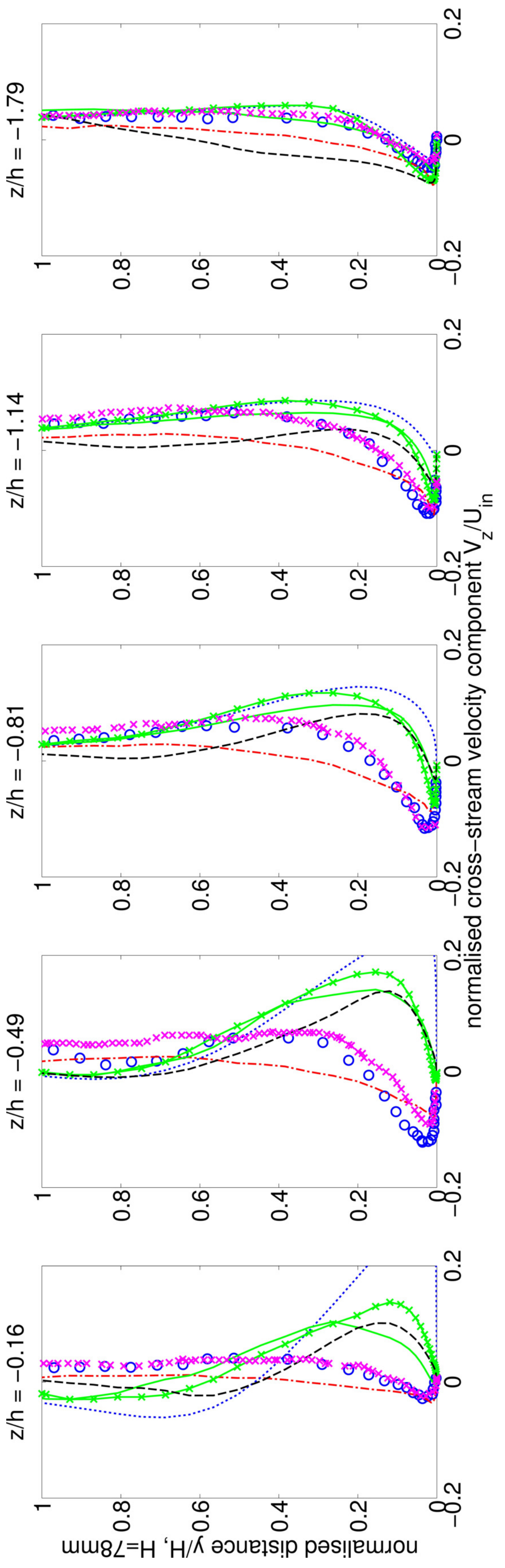

Figure 4.7: Cross-stream velocity components. LES-RANS data with forcing at interface by Davidson, all at x/H=3.69. Experimental data by Simpson. Legend: — SAS, ···· LES, - - - DES, · · · RANS, -×-× SAS-synthetic inlet, × × × hybrid LES-RANS by Davidson

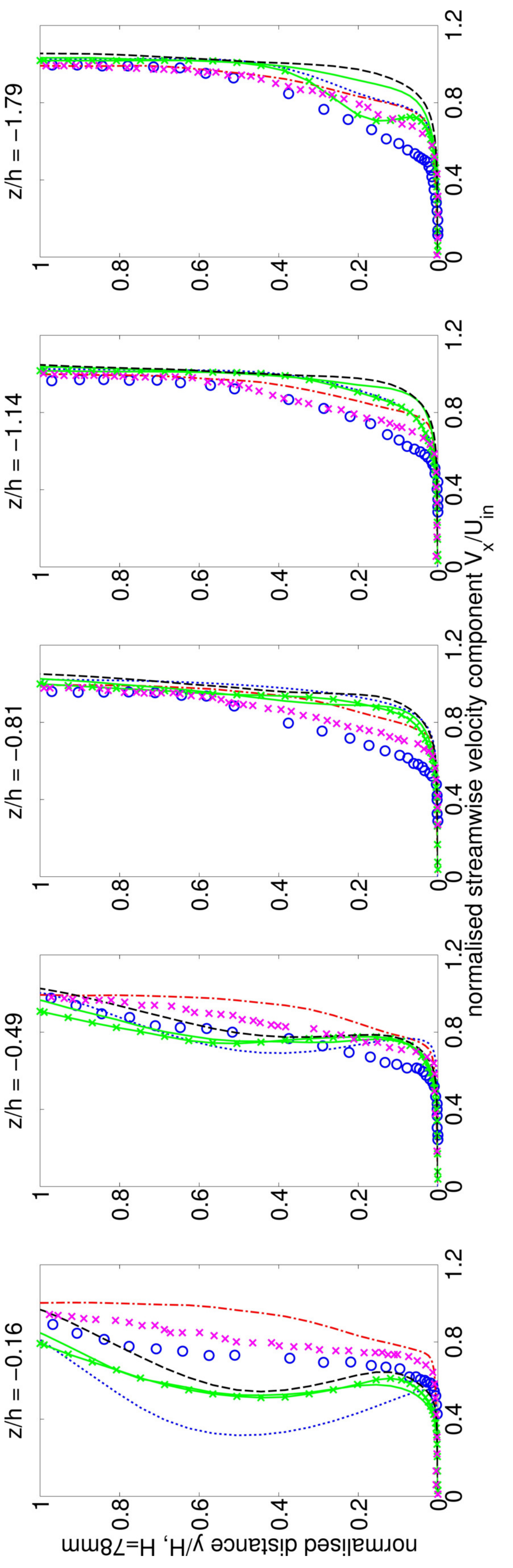

Figure 4.8: Streamwise velocity components. LES-RANS data with forcing at interface by Davidson, all at x/H=3.69. Experimental data by Simpson. Legend: — SAS, -·-· LES, - - - DES, · · · RANS, -×-× SAS-synthetic inlet, × × hybrid LES-RANS by Davidson

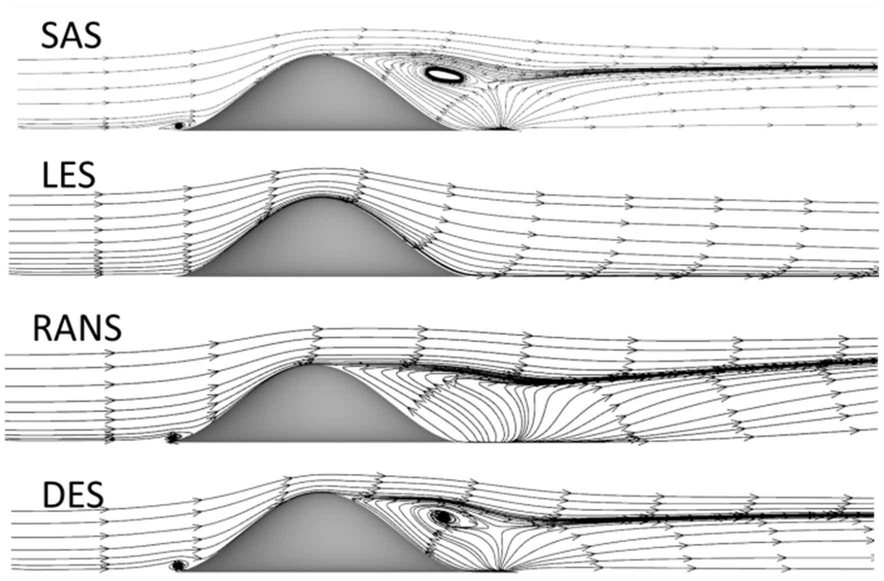

Figure 4.9: streamlines in streamwise direction along the 3d Hill predicted by different turbulence models; significant recirculation in cross-stream direction is present causing the lines to appear to be “flowing” away from the walls

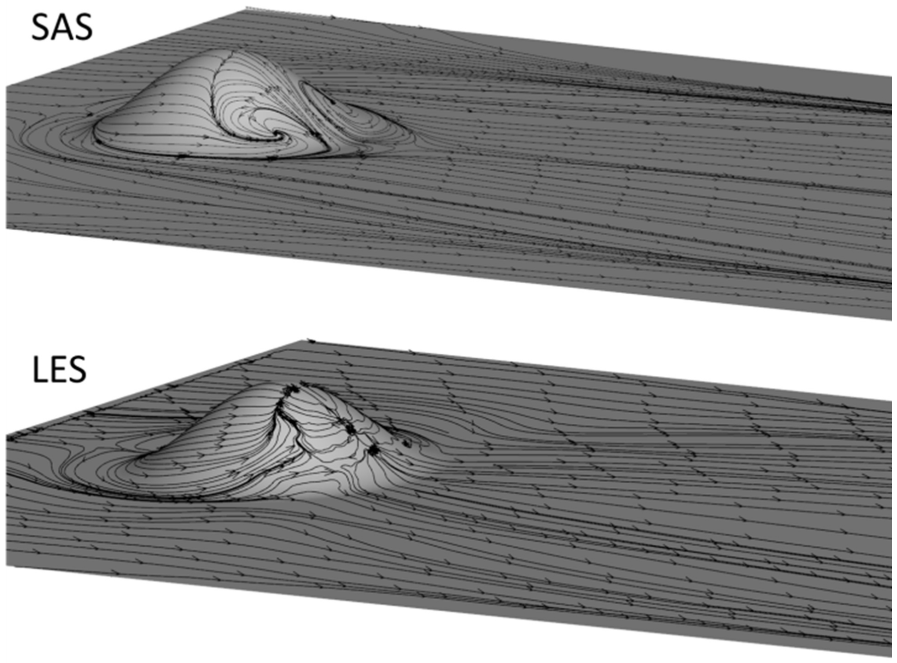

Figure 4.10: Surface streamlines for SAS and LES

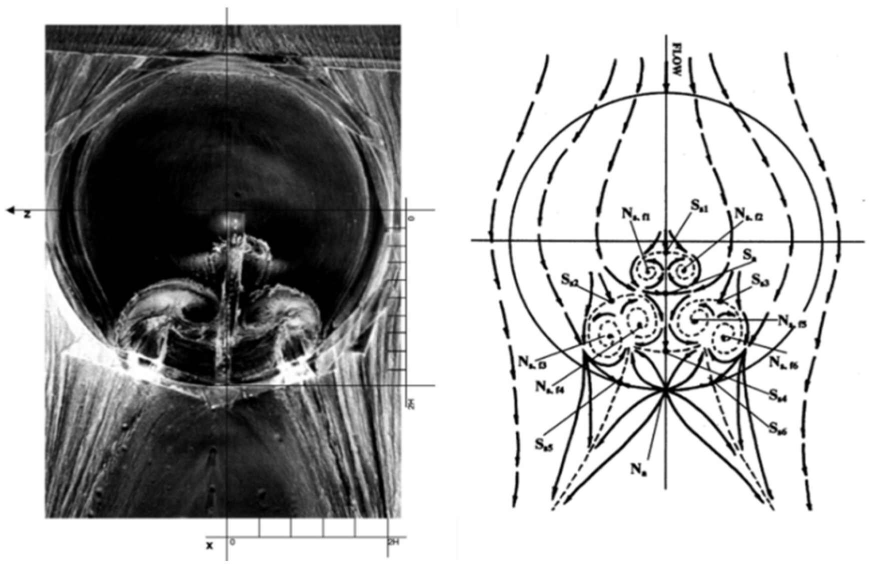

Figure 4.11: Top view of the hill and flow structures. On the left the experiment on the right a schematic of the flow. The image is of oil-flow on a wooden hill and illustrates streamlines obtained experimentally. It confirms the flow is symmetric and there are two main turbulent features around the symmetry plane. Both from Simpson et.al. [1]

4.9). The feature is predicted by all computational approaches tested, except for LES and is also not present in the Hybrid LES-RANS results [118], or LES [172]. Such features are a good test of the accuracy of the method as their presence becomes an immediate indicator.

SAS with synthetic inlet turbulence [177][176] imposed on the mean velocity profile was also tested, and Figure 4.7 and Figure 4.8 suggest there is little difference in results using this method. The main differences between SAS with and without synthetic turbulence can be seen near the inlet at lower z/H and the velocity profile is further away from the experimental data in the synthetic turbulence case. It is interesting to see both DES ($k-\omega$ subgrid model, without synthetic turbulent inlet) and SAS producing very similar result for the most part, including streamlines.

4.4 Ribbed channel - periodicity

4.4.1 Computational setup

The computational setup used for the ribbed channel is identical to the experimental setup of Acharya [2] and computational setup of Liu [173]. Figure 4.12 shows a

Re_b	14200
$L \times H \times W$	$0.127 \times 0.061 \times 0.3m^3$
U_b	$3.6\ m/s$
e	$6.35\ mm$
D_h	$101.6\ mm$
q''_w	$280\ \frac{W}{m^2}$

Table 4.2: Flow parameters for the ribbed channel flow. Nomenclature as per Figure 4.12

Grid	Size	Δx^+_{max}	Δz^+_{max}
$204 \times 94 \times 120$	2.3m	9	30
$186 \times 88 \times 112$	1.8m	10	33
$170 \times 82 \times 106$	1.5m	12	35
$158 \times 76 \times 98$	1.2m	13	39
$138 \times 68 \times 94$	0.9m	14	42

Table 4.3: Grids used to establish mesh sensitivity for ribbed channel. Average $y^+ = 1$ for all grids. Δz^+ and Δx^+ are $y^+ = 1$ equivalent concepts in the other two dimensions useful for measuring the grid with respect to the flow.

schematic diagram of the ribbed channel geometry and Table 4.2 shows the input values used for calculations.

A structured multiblock grid was created, as for the 3D Hill. Table 4.3 summarises grid sizes used to establish grid sensitivity for ribbed channel. Figure 4.13 shows an example mesh used in the present study. $y^+ < 1$ was achieved in the entire domain with first grid node being 0.11mm away from the wall. For comparison the grid density used by Liu [173] varies from $139 \times 21\,(x,y)$ to $199 \times 142\,(x,y)$.

Standard no-slip conditions are prescribed on all the walls. Since experimental data involved a channel with nine ribs [2] the inlet conditions for this case was streamwise periodic flow with prescribed massflow. Although Liu [173] used pressure to enforce the streamwise periodicity, in the present work massflow of $\dot{m} = 0.080703$ was applied based on a Reynolds number of 14,200 and a length scale of 101.6 mm as used by Liu

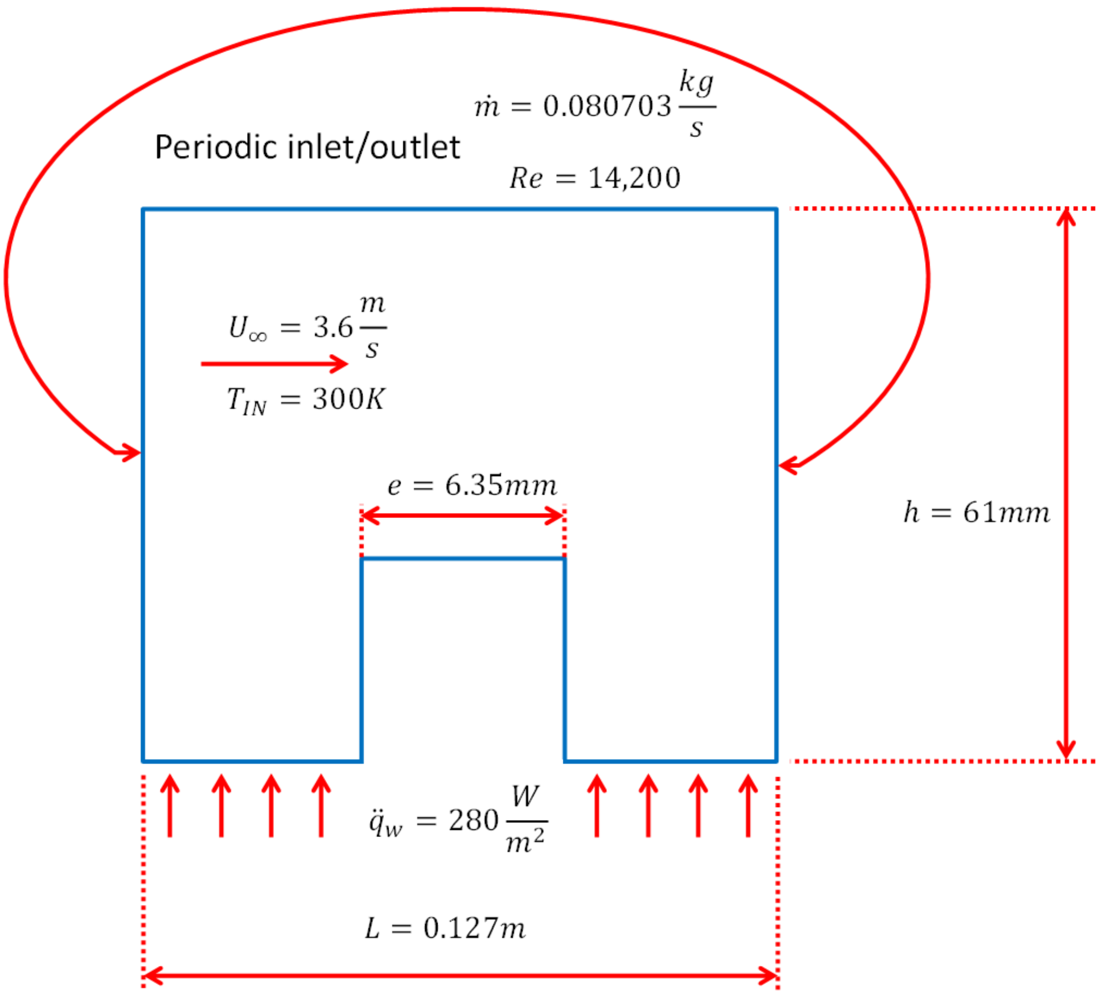
$\dot{m} = 0.080703\frac{kg}{s}$
Periodic inlet/outlet
$Re = 14{,}200$
$U_\infty = 3.6\frac{m}{s}$
$T_{IN} = 300K$
$e = 6.35mm$
$h = 61mm$
$\ddot{q}_w = 280\frac{W}{m^2}$
$L = 0.127m$

Figure 4.12: Schematic of the ribbed channel flow.

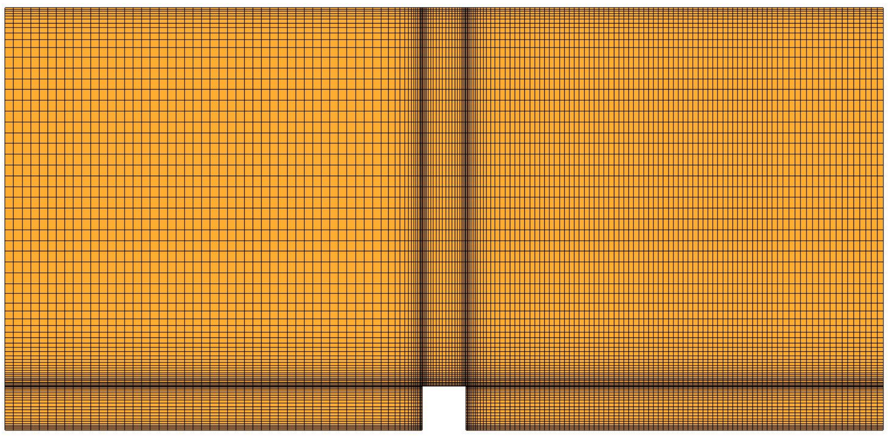

Figure 4.13: Cross-section of the ribbed channel mesh resulting from sensitivity study; every second grid point is shown

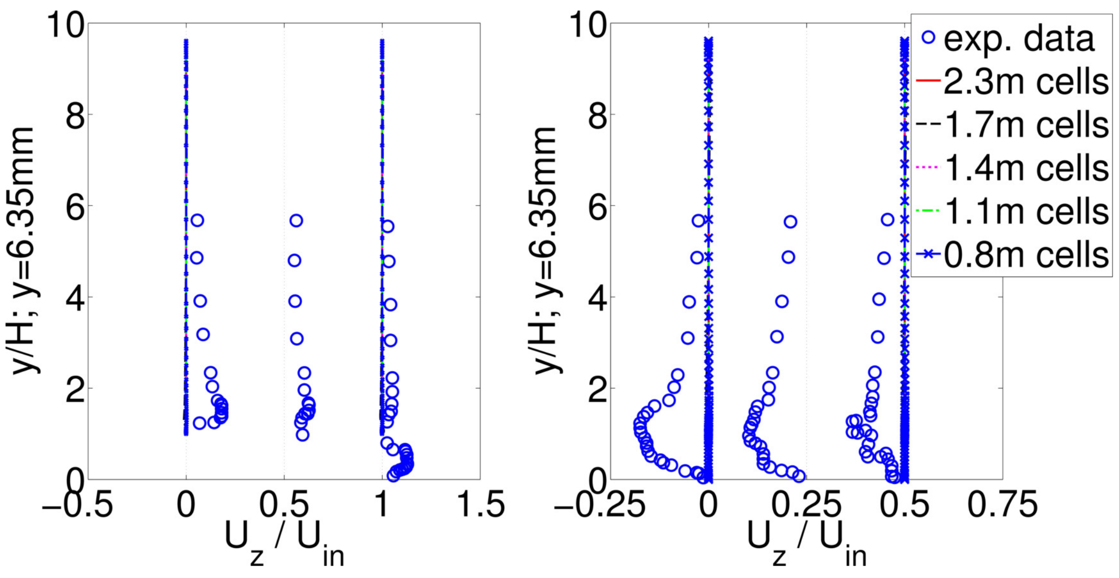

Figure 4.14: spanwise velocity component plot used to establish grid sensitivity for ribbed channel. The lines are x/h 10, 10.5, 11.1, 13.6, 16.2, 17.6 left to right, respectively.

[173] and Acharya [2], see also Table 4.2. A uniform heat flux of 280 Wm^{-2} was applied on the bottom channel surface, excluding the rib surface, as shown on Figure 4.12. A constant temperature of 300K was applied for the inflow. No heat flux condition was prescribed for all remaining walls. Heat transfer results were extracted from the plane mid-way in the spanwise direction. Turbulent Prandtl number was set to 0.9 for RANS calculations and 0.4 for resolved (LES) simulations.

Figure 4.14 and 4.15 show streamwise and spanwise velocity components, respectively. Figure 4.16 shows Nusselt number plots for different size meshes used in the current work. It is apparent that grids of 2.3m cells and 1.7m cells perform similarly and refinement above 1.7m cells gives little benefit for the extra computational time required. Velocities appear to have much less pronounced grid sensitivity than heat transfer results and even on a grid as coarse as 0.8m cells the velocity field is almost identical. As to heat transfer, while the results have 200% – 500% error with respect to experimental data, meshes of 2.3m and 1.7m cells appear to be performing best. The grid of 2.3m cells was used for further computations as the cells‘ aspect ratio was becoming too high for the grid of 1.7m cells, given that $y^+ < 1$ was kept for all the grids.

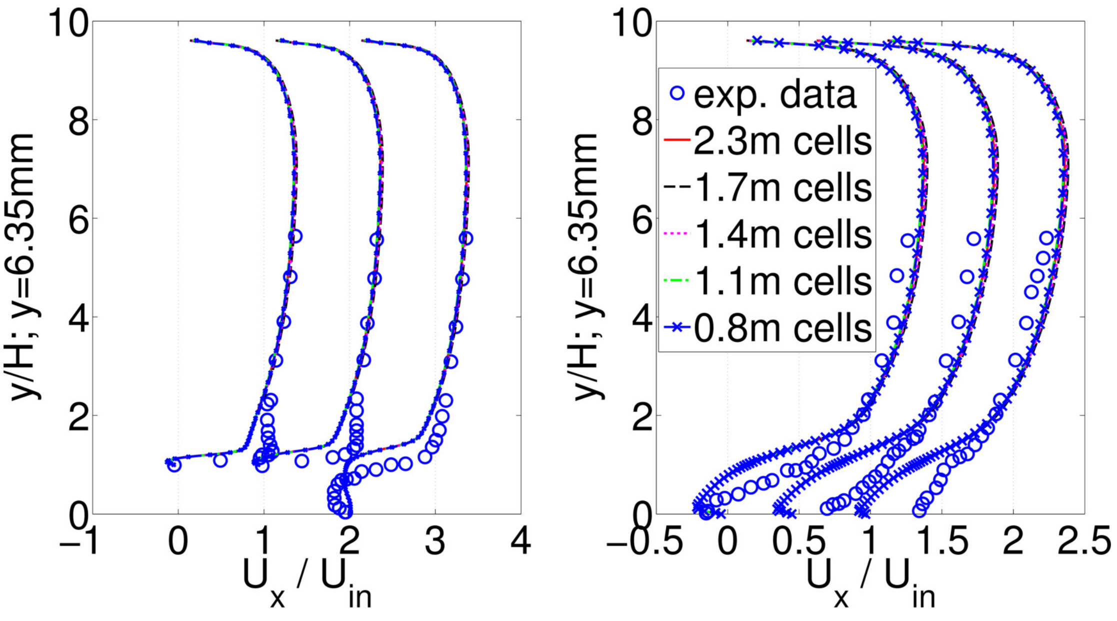

Figure 4.15: streamwise velocity component plot used to establish grid sensitivity for ribbed channel. The lines are x/h 10, 10.5, 11.1, 13.6, 16.2, 17.6 left to right, respectively.

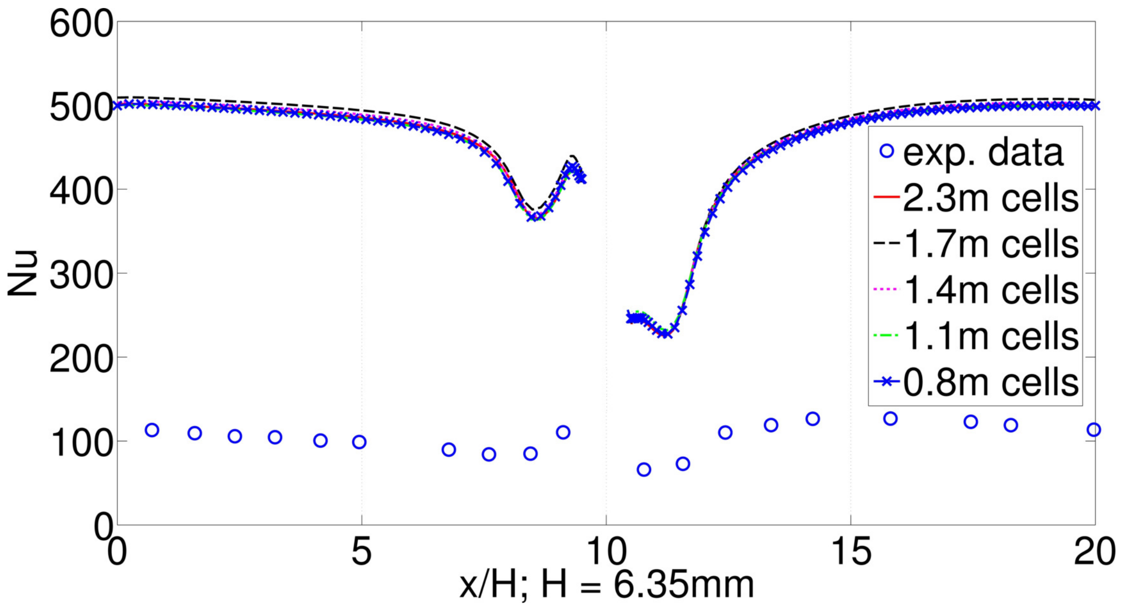

Figure 4.16: Nusselt number plot used to establish grid sensitivity for the ribbed channel. Obtained with the SAS–SST model.

4.4.2 Results

Figures 4.18 and 4.19 compare velocity field with that obtained experimentally by Acharya [2]. Similarly as for the hill case, streamwise velocities are predicted well by most turbulence models, including the RANS. LES is for the most part the most accurate for velocity prediction, however this accuracy is not consistent and there are regions, especially away from the wall, where all other models exhibit much higher accuracy than LES.

The ribbed channel case is another example of flow with well established experimental data that's used widely for validation purposes. Figure 4.20 shows streamlines with the z component of velocity set to zero to display the recirculating regions midway between the channel walls. There is one main recirculating region upstream of the rib and three smaller ones, each adjacent to either of the three walls of the rib. The geometry is sharp-edged and the separation point is well established for all the models and grids tested here. The reattachment length however is different for each model.

It is interesting that all models apart from LES, and to an extent DES predict almost zero cross-stream velocity.

Figure 4.20 shows time averaged streamlines predicted by different methods. A characteristic feature of such flows is the length of the recirculating region, L_R, summarised in Table 4.4. From the two it is apparent that DES and LES predict the length of the recirculation region closest to experimental values. DES however predicts a larger height for the region in comparison with experiment. While there is some improvement in SAS over RANS in prediction of velocity field, it is still below the accuracy offered by LES.

As to the heat transfer results, LES also appears to be predicting Nusselt numbers closest to the experimental data, as shown on Figure 4.17. The DES and SAS predictions are comparable with RANS.

Method	Value of L_R	% error w.r.t. exp.
LES	8.0	14
SAS	9.5	36
RANS $k-\omega$	11.0	57
DES	9.0	28
experiment	7.0	–

Table 4.4: Lengths of the recirculating region upstream of the rib summarised

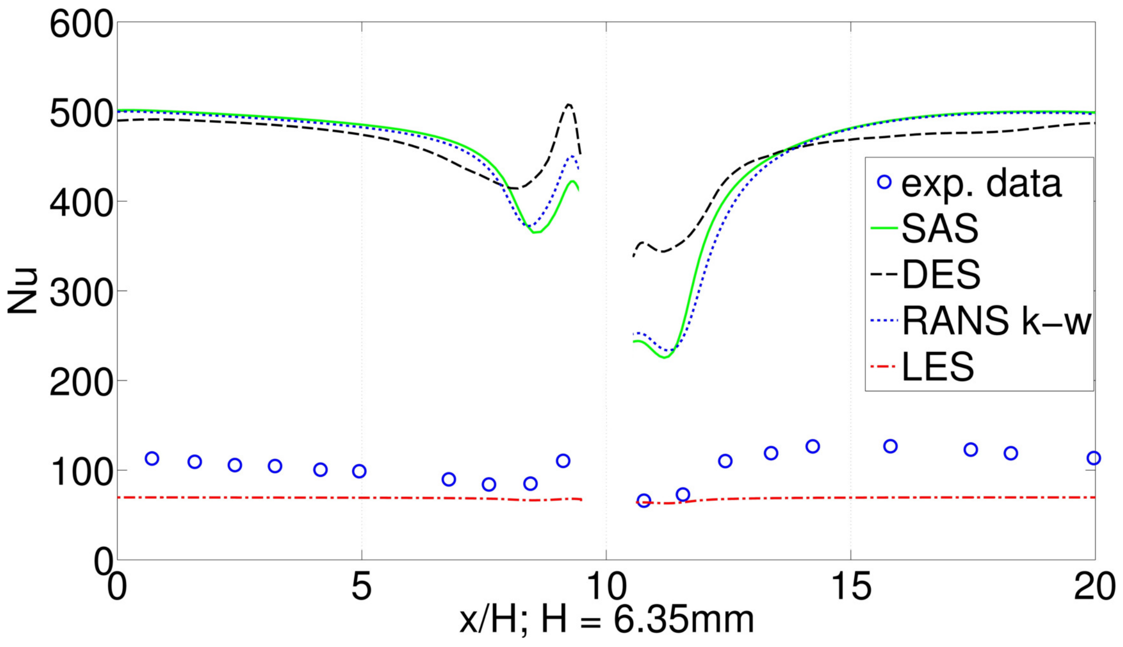

Figure 4.17: Nusselt number plot for the ribbed channel. Experimental data by Acharya[2]

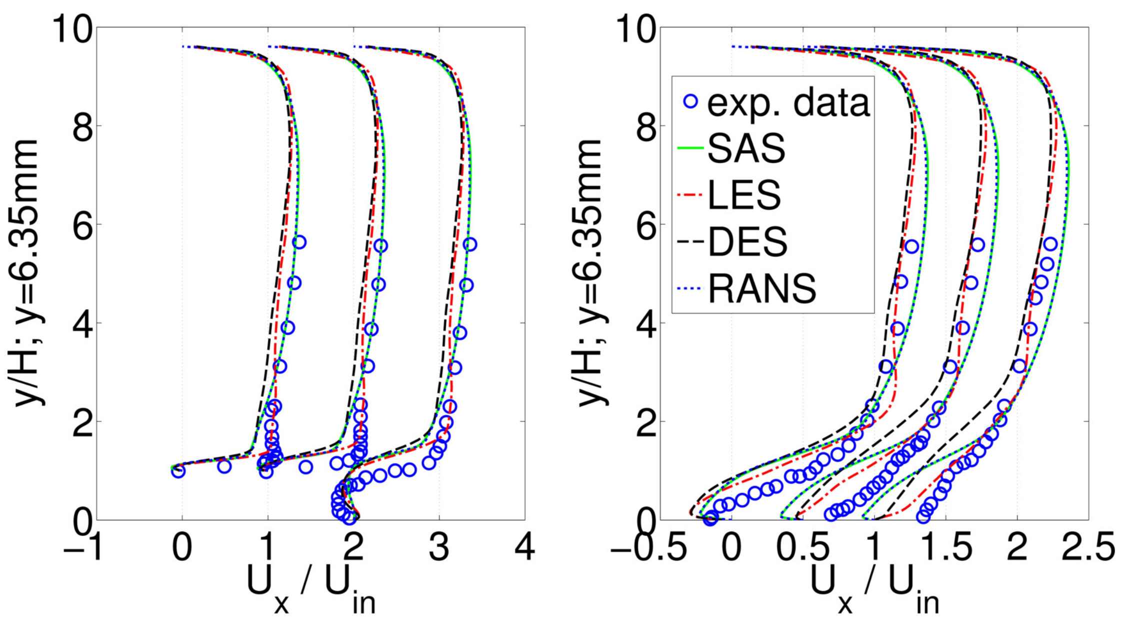

Figure 4.18: Normalised streamwise velocity component. The lines are x/h 10, 10.5, 11.1, 13.6, 16.2, 17.6 left to right, respectively.

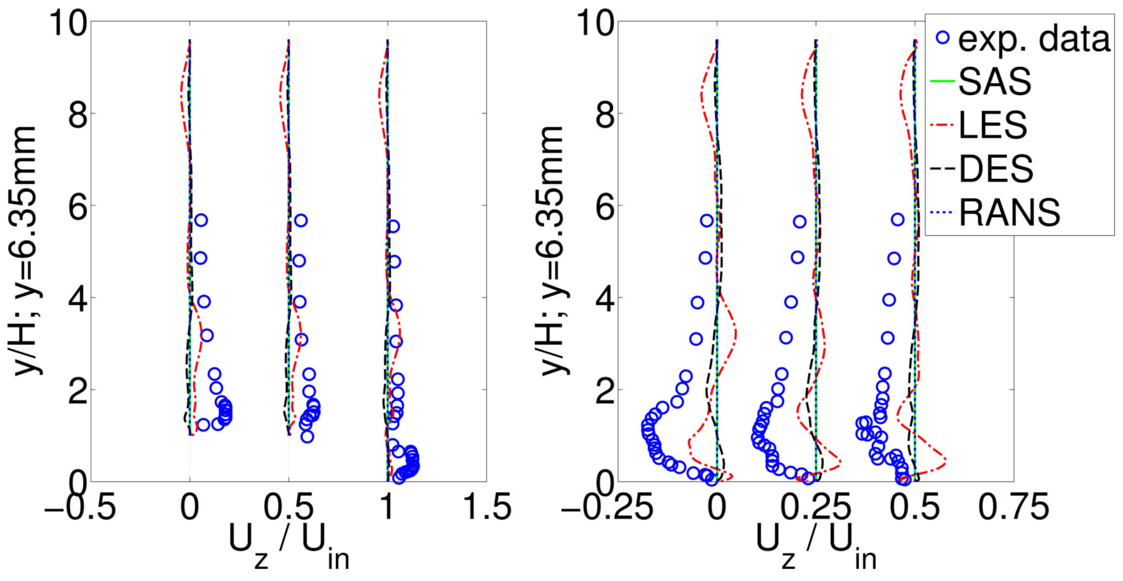

Figure 4.19: Normalised cross-stream velocity component. The lines are x/h 10, 10.5, 11.1, 13.6, 16.2, 17.6 left to right, respectively.

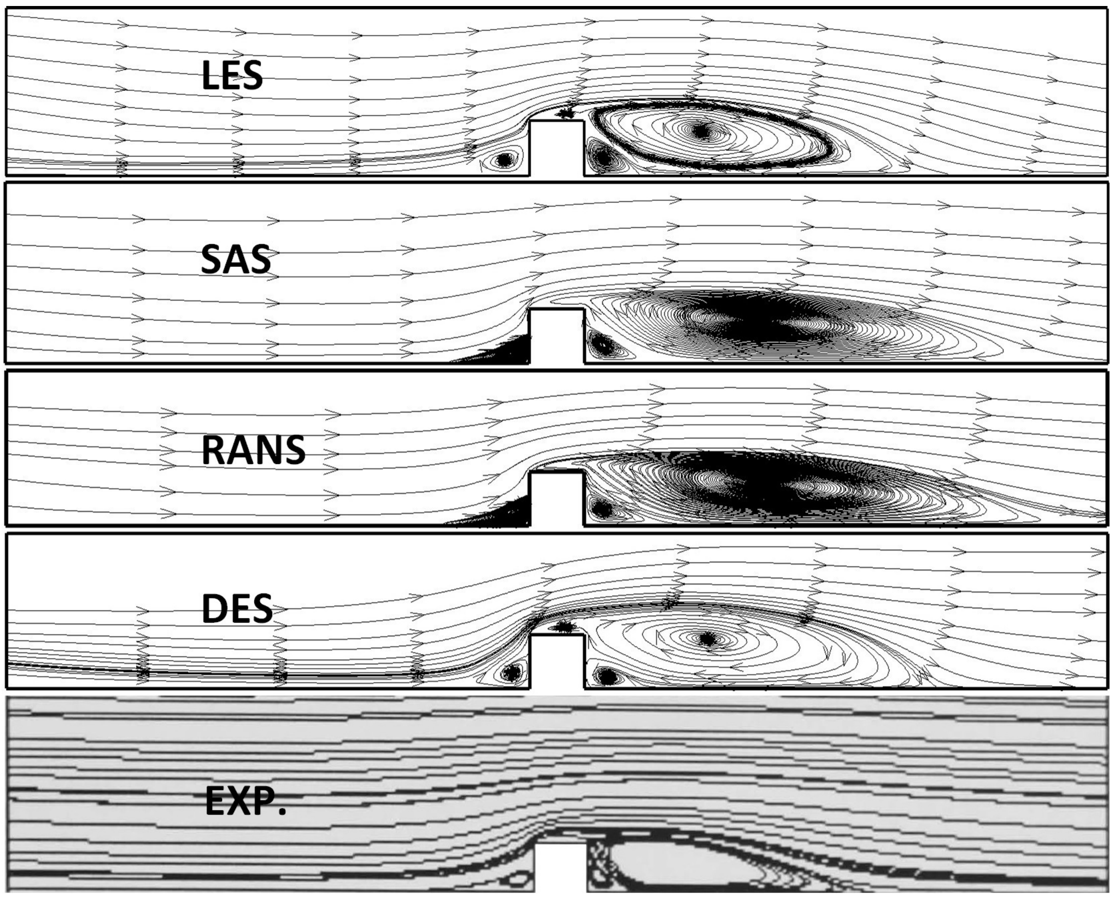

Figure 4.20: streamlines in streamwise direction along the ribbed channel predicted by different turbulence models. Exp. data from [2]

4.5 Conclusions of the initial results

While the idea behind the SAS modelling is generally favourable, in the present case leads to the unfavourable situation when the equations operate neither in resolving LES-like mode, nor in the modelling RANS mode, but somewhere in between. This is likely due to too low 'unsteadiness' levels for the SAS-SST model. This situation was also mentioned by Menter *et.al.* [166] and was the motivation behind creating the forcing terms.

The turbulent viscosity in the SAS-SST simulations is likely too high for a resolving simulation and the simulation differs little w.r.t. RANS, in comparison with LES which is closer to the experimental data. This conclusion is also stated by Davidson [169].

Going back to the original application of turbine blade cooling passages, based on

the results so far it is concluded that more testing should be done on the SAS-SST model to more conclusively state whether or not this approach can be recommended instead of LES or other methods. The results obtained here suggest LES or Hybrid LES-RANS with fixed LES y^+ boundary is likely a more feasible approach even if the grid must be kept coarse. If, however, explicit modelling of several bumps is required a mesh even coarser than the present grid would likely be used as computational cost would increase massively otherwise. In such cases SAS-SST may be more attractive and it is recommended that this should also be further investigated. Additionally inside a turbine blade there are multiple hill-shaped obstacles following one another and turbulent structures generated grow cumulatively to a certain size; this will increase overall turbulence intensity and the resulting field may be more suitable for the SAS-SST computation.
The Scale Adaptive Simulation computational approach was investigated through application to a three-dimensional axisymmetric geometry and ribbed channel and compared with other researchers' simulations as well as experimental data. Both the hill and the channel used are representative of features used inside turbine blade cooling passages to promote heat transfer.

It was found that while RANS approaches fail to predict the flow completely, LES appears to produce a reasonable momentum solution even on coarser meshes. The fact that LES appears to be doing well is somewhat surprising as the computational grids are not sufficiently refined as required by standard LES practice. Using SAS was found to offer only minor improvement over RANS for velocities calculation and much larger improvement for heat transfer results. The error in heat transfer however was still large for SAS with respect to experimental data ($> 200\%$).

The size of the recirculation region predicted by LES is similar to that found experimentally in both cases tested here whereas the size predicted by SAS is significantly larger than that of LES, and similar to that of RANS.

In addition, SAS predicts a small recirculating region upstream at the foot of the 3D hill, not found experimentally. The DES and RANS models predict a similar result. The flow separation point on the hill occurs at approximately the same point ($\approx 0.1H$ past hill peak) for all models, in agreement with experimental data. Due to geometry the ribbed channel has clearly defined separation point which is similar for all models.

The ribbed channel study confirms the findings from the 3D hill. The LES approach also appears to be the better choice of model for the present problem, for both velocity prediction and heat transfer.

Overall the evidence suggests for the present geometries there is little improvement in using SAS over RANS, DES, LES or hybrid LES-RANS while incurring noticeable computational penalty for solving the additional terms in the SAS model. LES on the same grid (coarse for LES) produced velocity results significantly closer to experimental data than SAS. Heat transfer is also predicted better by LES.

Also the SAS model however was found to be lacking resolved content even on geometries that are well known to be unsteady and strongly separated such as an axisymmetric hill [169] or sharp edged rib [162]. It was found that the resolving mode is not triggered even under those strongly turbulent conditions. To remedy the problem, Menter et. al. proposed artificial forcing [166] to trigger the LES-like behaviour of the SST-SAS model. The forcing is based on ideas of Kraichnan et. al. [176], Smirnov [177] and Keating [180] and was proven to work for a reverse-facing step [166]. Forcing will be discussed in more detail later.

4.6 Stationary channel flow

4.6.1 Computational setup

The geometry consists of a development section with eight equi-spaced ribs and a longer outflow region, as shown in Figure 4.21. The channel edge is taken as the hydraulic diameter. The square rib height to hydraulic diameter ratio is 0.1 and pitch to rib ratio is 10. The inflow is positioned at half a rib pitch upstream of the first rib and the outflow boundary is sufficiently coarse and far downstream for all the generated recirculation to be dissipated before exiting.

The grid used in the present study is similar to Viswanathan et.al. [181] in and is suitable for DES simulations as shown in Figure 4.22 and as described by Spalart et. al. [170]. The non-dimensional wall distance y^+ is less than 1.5 in the near-rib region in all cases and varies locally up to $y^+ \approx 4$ The motivation for using the present grid was to test the SST – SAS model on a DES grid and benchmark its resolving performance against DES. From the present simulations, SAS required approximately 15% more computational time than LES and 5% more than DES per iteration on the same grid. If

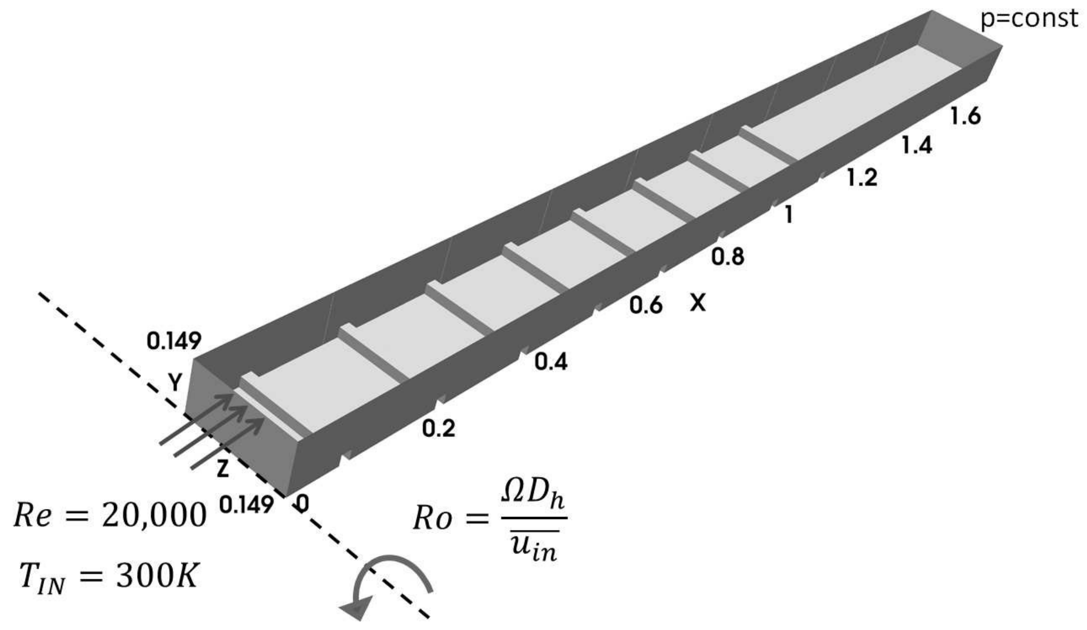

Figure 4.21: 3D representation of the geometry. Only lower half is shown (there is a mirrored top cap on the long section). For simulations including rotation, the axis of revolution is always the z-axis with right-hand rule to determine direction.

the grid must be refined to LES requirements, the LES approach is more advantageous as it is faster and already proven to produce good results [182]. The study therefore should be beneficial to industrial users of the model as it assesses SST – SAS performance under realistic, cost-efficiency driven environment where grids must be coarse and run times kept to minimum.

The Reynolds number used in the present study is 20,000 based on the hydraulic diameter. The Prandtl number is 0.7 and the Rotation number is 0.3 for the case with rotation examined in the later section. Non-rotation simulations are performed for benchmarking the models together as well as with experimental data. The inlet definition is a constant 'plug flow' definition, as described in experimental following Sewall et. al. [182]. In the numerical simulations of [182] it was found in the paper that imposing a non-constant velocity inlet is inconsequential after the second rib and very minor after the first. A constant velocity inlet was used for simplicity but also to be consistent with other numerical results. It is also shown by Davidson et. al. [169] who performed SST – SAS simulations with a fluctuating, realistic inlet without forcing that it seemed to have very little effect on the solution. This is likely due to the fluctuations being insuffi-

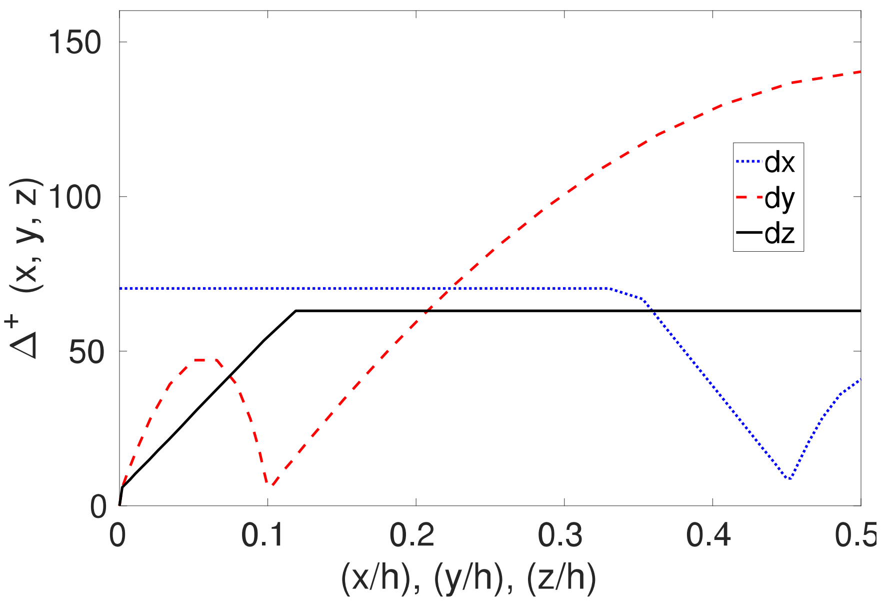

Figure 4.22: Non-dimensional grid characteristics.

cient to switch the SAS model to resolving mode and hence being damped out. Finally, in the LES simulations of Tyacke and Tucker [183] of the same geometry it was also found that turbulent inlet fluctuations of 10% did not have a strong impact on the flow development.

All the walls are prescribed a temperature of 330K, while the inflow temperature is 300K. This latter value is also used as reference in the calculations. Both heat flux and temperature thermal boundary conditions were tried and it was found that very minor differences in heat transfer predictions were observed. Hence the isothermal wall boundary conditions were used for consistency with other available numerical data.

All heat transfer plots are normalised with the hydraulic diameter D_h and Dittus-Boelter correlation:

$$Nu_0 = 0.023 Re_b^{(0.8)} Pr^{0.4} \tag{223}$$

A grid refinement study was performed using the steady state $k-\omega$ SST model and the SST-SAS model. While in general a resolving simulation such as SAS is expected to exhibit grid sensitivity as it approaches DNS with grid refinement, it is still of value to examine the solution sensitivity to the grid.

As shown on Figure 4.24, without introducing any forcing terms, the transient SST –

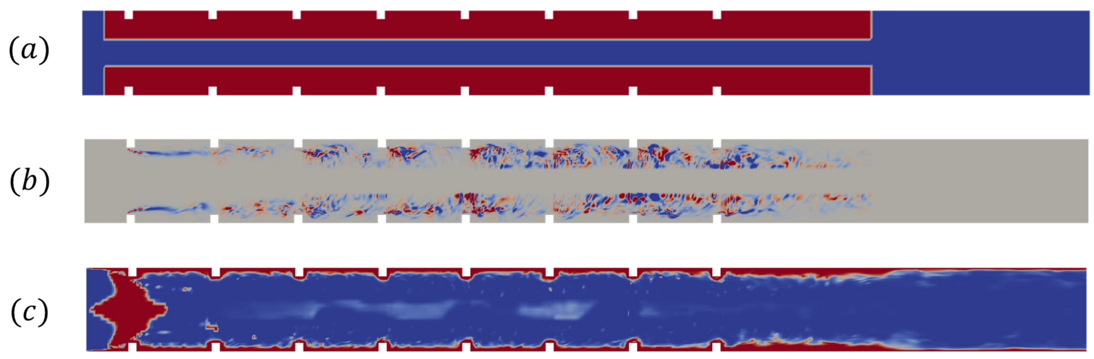

Figure 4.23: contour plots showing where the forcing terms are active. (a) is the prescribed area where forcing terms were active in the simulation, (b) is magnitude of the source forcing terms and (c) is the $k-\omega$-SST first blending function F1 magnitude indicating modelled near wall region

SAS simulation performed similarly to RANS on the refined grid (approximately $0.5\Delta x$ in any direction). The grid is of LES requirements and has 9.3m cells. Even with such refinement there appears to be very little resolved content.

The figure demonstrates that unforced SST – SAS solution is nearly identical to RANS even with significant grid refinement. In addition very little instantaneous fluctuations were present in the transient SAS solution without forcing. This is indeed the reason for the present work. All unsteady simulations in the present study were started from a well-developed DES solution that exhibited much higher instantaneous unsteadiness than SAS. The final grid used was one consisting of 2.4m cells.

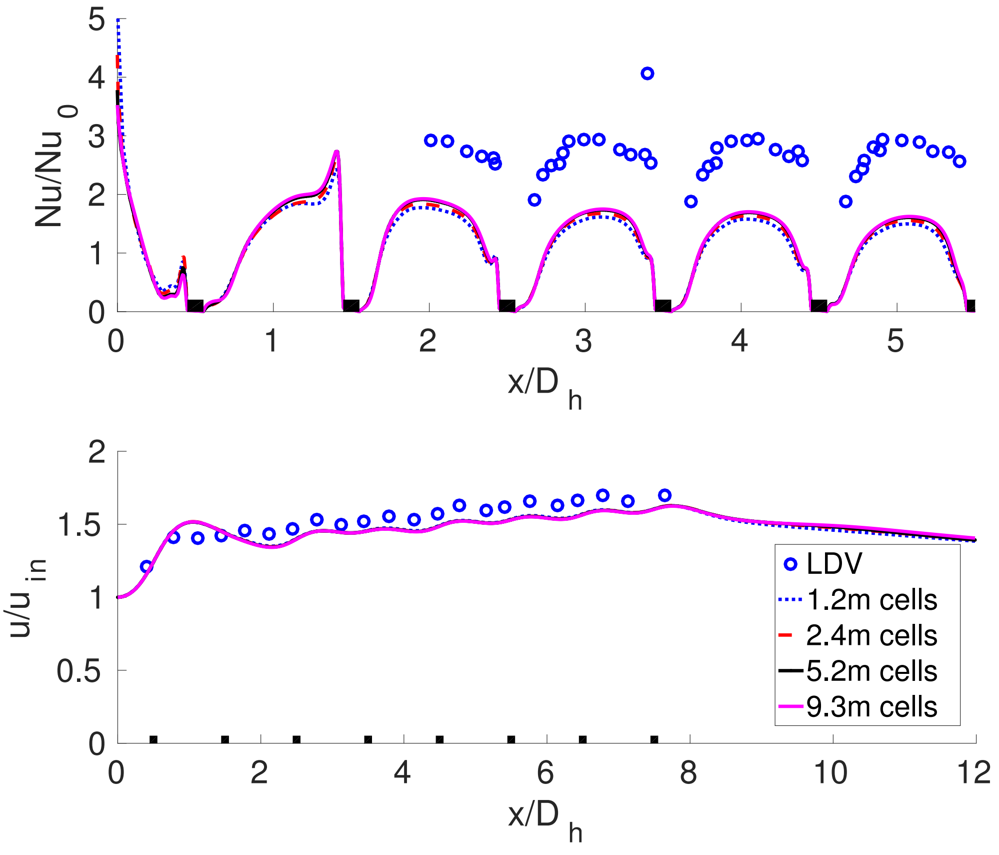

Figure 4.24: Grid refinement test for SST – SAS model without forcing. RANS solution is tested to be grid-insensitive.

4.6.2 Results

As shown on Figure 4.25, the forcing appears to make significant difference for the SST – SAS model. The plots show that switching from modelling mode (where most of the TKE is captured by turbulence model) to resolving mode (where most TKE is modelled explicitly as velocities) is present when the forcing terms are active. Importantly, unsteady content appears outside the boundary of the forcing zone, suggesting the turbulence is self-sustaining and the entire domain does not need to have the forcing applied. On the other hand, by further examination of the forcing function effect on Figure 4.23 and Figure 4.25 it was found that forcing is only significant in the unsteady recirculating regions behind each of the ribs and away from the wall. It is concluded that the forcing could likely be safely applied to the entire domain as it will interact with the SST – SAS model and reduce itself in steady or near-wall regions.

Looking at Q criterion isosurfaces on Figure 4.26, the absolute magnitude of the

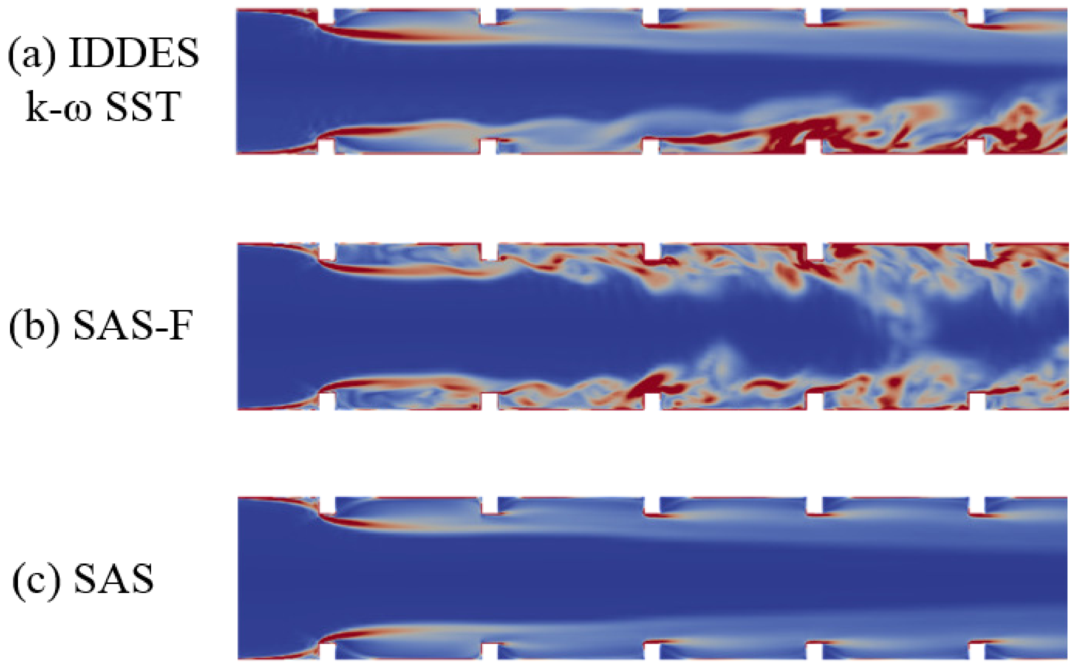

Figure 4.25: Instantaneous vorticity magnitude at the last time step after at least ten through flows of development. Except for the forcing, grid, boundary conditions and numerical setup are identical for all simulations.

Q criterion is not of primary importance, rather it is a useful metric in visualising turbulence. It is again clear from the figure that very little unsteadiness is present in the unforced SST-SAS solution and there is noticeable increase of unsteady content only as a result of the introduction of the forcing terms with identical boundary conditions.

Next, mean velocity profiles and their fluctuations are analysed. Measurements at four locations downstream of the inflow plane are shown on Figure 4.27. The velocity profiles are taken midway between the ribs in the streamwise direction and midway between the walls in the spanwise direction. The measurement locations in the wall-normal direction range from the $y = 0$ to one quarter of the height of the duct. Apart from the first profile location, an improvement of streamwise velocity prediction is observed with the introduction of the SAS forcing term (Q, shown on Figure 4.23) compared with the original SAS approach. It can also be seen that the profiles are very similar to both the SA and $k-\omega$ SST variants of DES.

The velocity fluctuations on Figure 4.27 in the streamwise direction are both underpredicted and over predicted by all models. The SAS-F model dramatically over

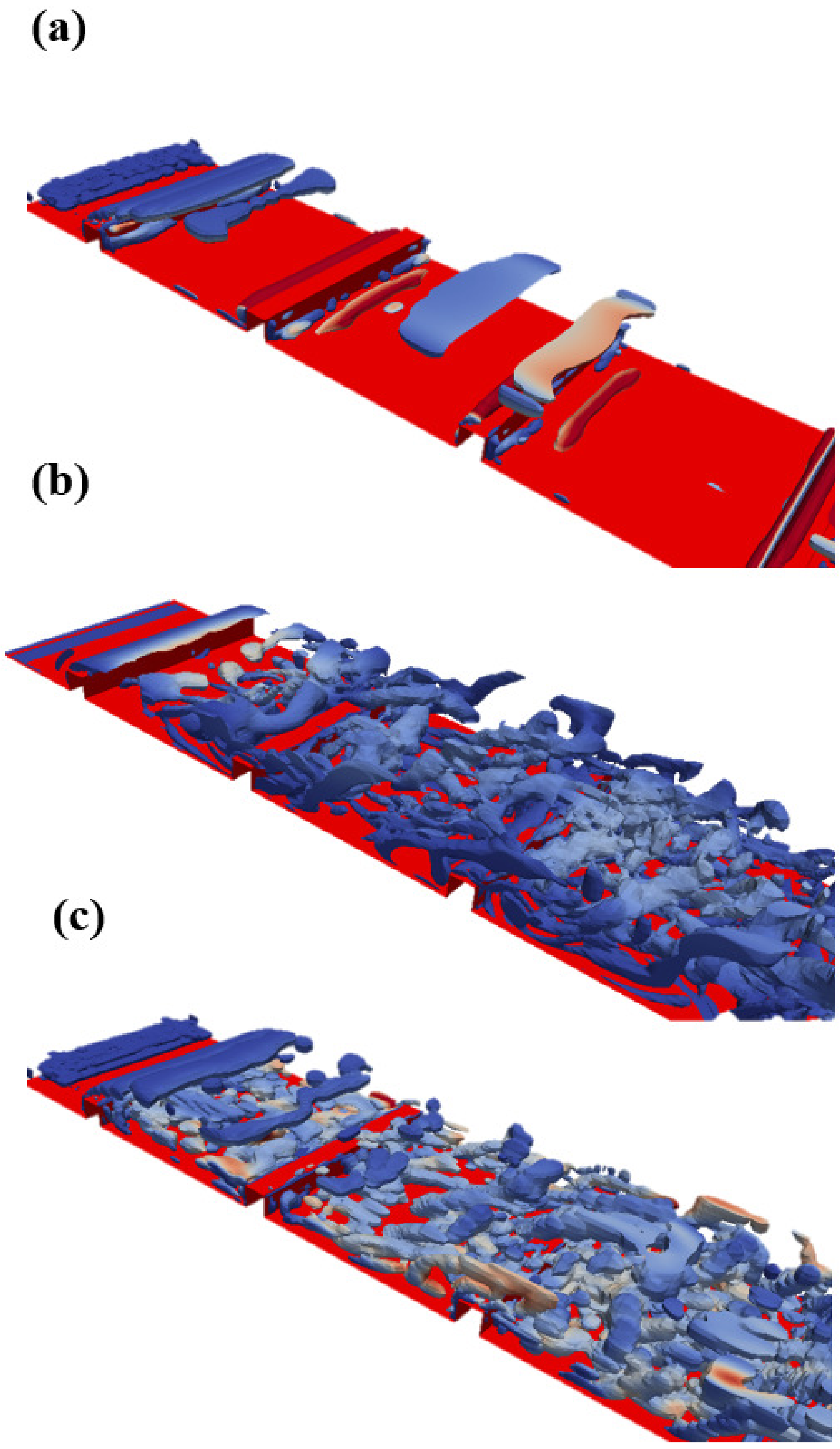

Figure 4.26: Normalised Q criterion isosurfaces of 0.2 value coloured by Turbulent Viscosity Ratio. (a) SST – SAS without forcing, (b) SST – SAS with forcing, (c) IDDES $k-\omega$ SST. Note low eddy viscosity in the unsteady regions.

predicts the fluctuations at the second profile location and at larger values of y/D_h for the third profile location. At the fourth rib it can be seen that the SAS-F and DES variants are similar. It should be noted that very little resolved turbulence was observed for the standard SAS model as can be seen on the plots. The wall normal velocity fluctuations are generally under predicted by all models and interestingly the values predicted by the SAS-F model are approximately 50% below the DES.

To further confirm that the models are operating in resolving mode, ratio of resolved to total turbulent kinetic energy γ is plotted in Figure 4.30. The ratios are nearly identical after the first rib and arithmetic average of γ for all ribs ahead of the first one was taken for simplicity. Pope [184] recommends that at least 80% of energy should be resolved for a simulation to be considered well-resolved. Apart from the region immediately downstream of the first rib this is true for both DES and SST-SAS. Again it can be seen from the figure that very little resolved turbulent kinetic energy is present in unforced SST-SAS model.

Figure 4.28 shows velocity as well as its fluctuations along a geometric centreline of the channel, with streamwise locations at 1, 2, 3 and 4 rib lengths. It is immediately apparent from this, and all other plots that baseline SST-SAS solution behaves nearly identically to RANS and there is little benefit in the SAS-SST simulation while requiring significant extra computational cost with respect to RANS. With introduction of forcing terms the fluctuations in streamwise direction are greatly increased and are overpredicted. The velocity fluctuations in wall-normal direction appear to be under-predicted. Since the forcing terms are statistically isotropic in all directions, it is likely that the SST-SAS model is under-resolving content in the wall-normal direction and over-resolving in streamwise as opposed to not being enough 'stimulation' for the model to do so. It can also be observed that DES, while still exhibiting significant error is much more accurate than the SST-SAS model in predicting the fluctuating component of decomposed velocity. Looking at the total velocity however on both Figure 4.29 and Figure 4.28 the difference in prediction between DES and forced SST-SAS is marginal and greatly improved with respect to RANS or URANS. It is also noteworthy that the first rib is not predicted accurately, likely a consequence of steady inlet definition.

Figure 4.31 shows the normalised Nu plot along the wall centreline of the channel. The forced SST-SAS solution is significantly improved with respect to RANS. Impor-

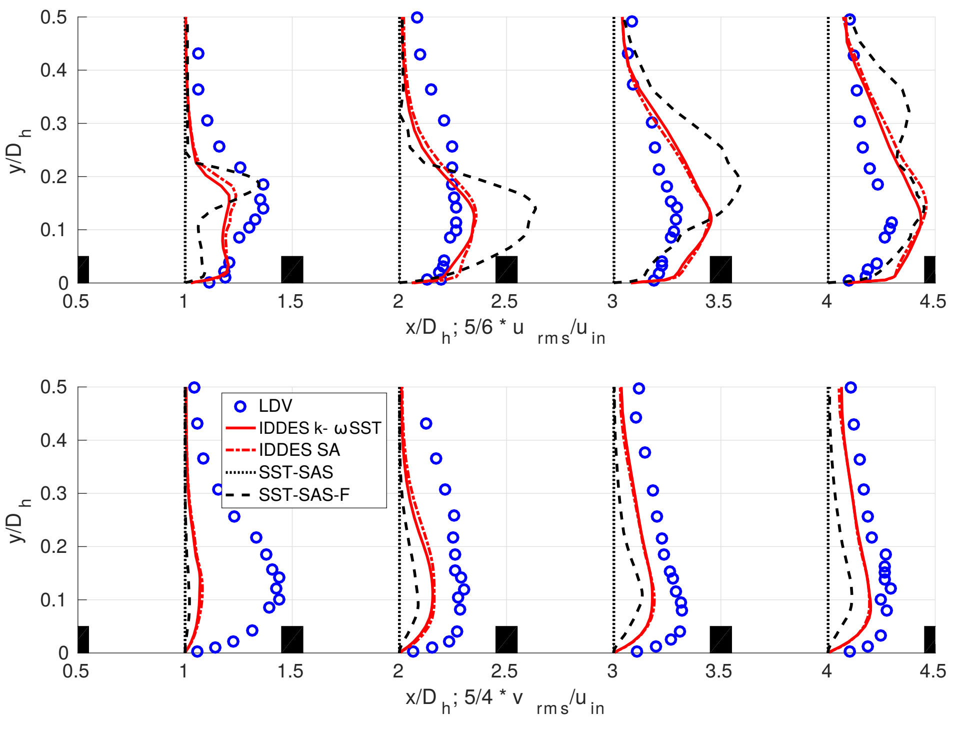

Figure 4.27: Resolved streamwise and wall-normal fluctuations starting from one rib pitch downstream of inflow plane. Note constant velocity inlet in simulations vs. naturally realistic in experimental data.

tantly, the unforced SST-SAS as well as RANS do not predict the sharp peak of cooling efficiency immediately ahead of the ribs and significantly under-predict heat transfer in all cases. The trends themselves predicted by SST-SAS-F and DES are similar to experimentally found ones however the Nusselt number is still significantly underpredicted by all approaches. This suggests deficiency in the near-wall RANS model and that further investigation is required.

Comparing with other authors, the LES of Sewall et.al. [182] is very close to the experimental measurements; the LES predictions are far superior to any other RANS or hybrid RANS-LES performed on the geometry. The hybrid LES-RANS of Tyacke and Tucker [183] is somewhat further away from the experimental solution, but requires significantly less computational effort due to RANS modelling near the wall. Viswanathan and Tafti [181] performed DES and confirmed the present conclusions that DES is much less accurate than LES on the present geometry. They also show that LES exhibits much smaller scale structures than DES, as expected.

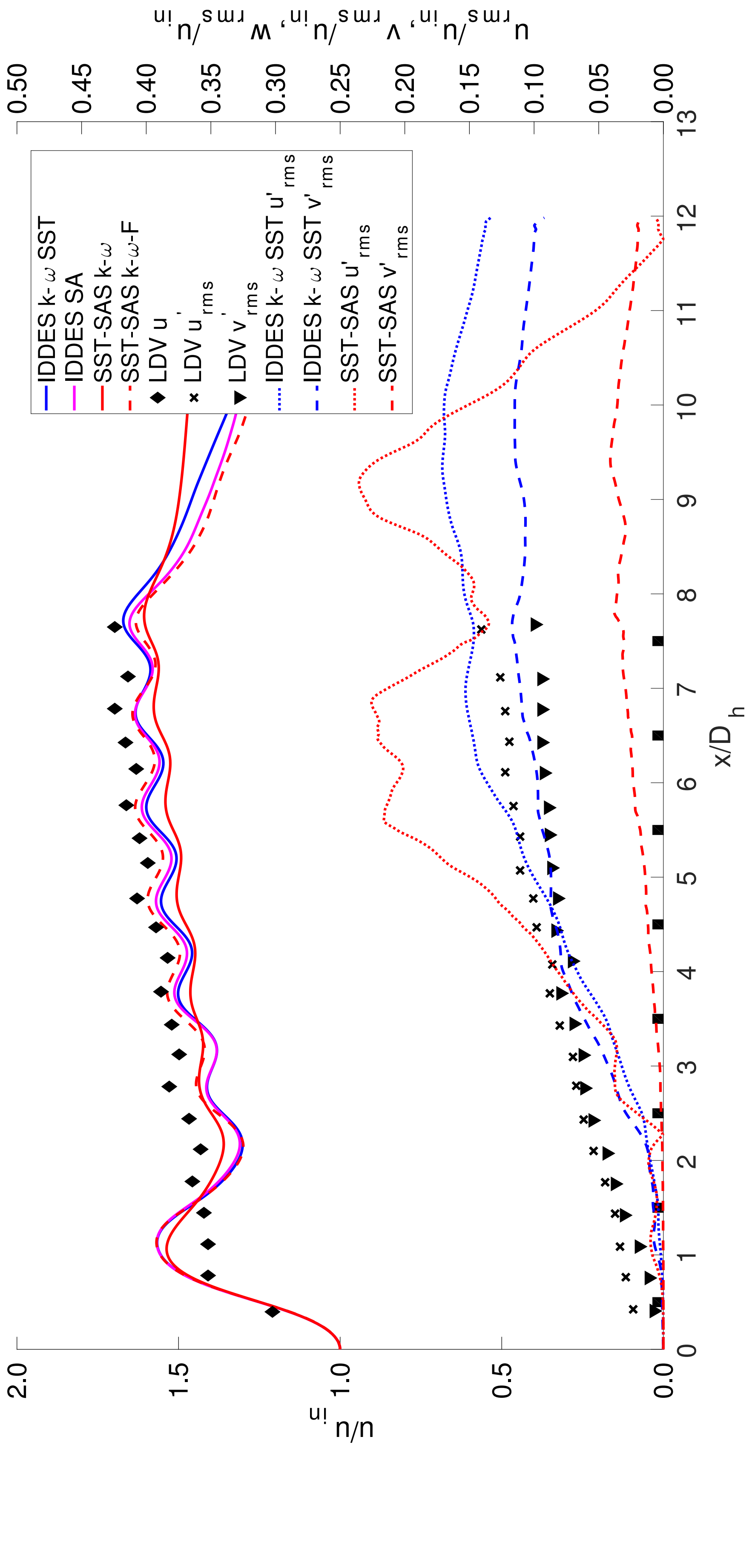

Figure 4.28: streamwise velocity component along a centreline and RMS of velocity fluctuations in streamwise and wall-normal direction for different models. Baseline SAS – SST fluctuations not isible due to near-zero values.

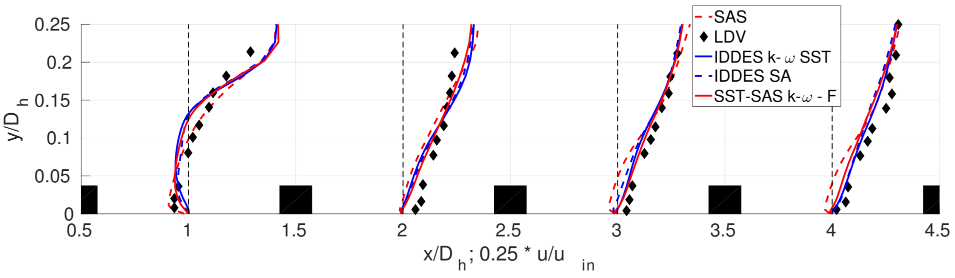

Figure 4.29: Streamwise velocity profile starting from one rib pitch downstream of inflow plane for various models. Note the 0.25 scaling factor on the horizontal axis for the velocity measurements.

Figure 4.33 shows the energy spectra present in the resolving simulations. It should be noted that such plots were generated at various points behind different ribs and all produced very similar results. The standard SST-SAS model is steady and not shown on this plot. For the SAS-F and DES approaches an inertial range is visible beyond which there is a frequency band indicating the existence of an inertial sub-range with a -5/3 slope. The steeper slope beyond this relates to the damping of the energy content of the frequencies due to molecular and turbulent viscosity. Also shown on the plot is the grid cut-off frequency of 320Hz beyond which structures cannot be resolved. Comparing the forced SST-SAS solution with DES, the low frequency eddies for the SAS-F solution contain more energy and the higher frequency eddies less energy than for DES (close to the cut-off frequency).

Figure 4.32 shows streamwise and spanwise two-point cross-correlation plots respectively for SAS-F and DES behind the third rib. The SAS-F results show steeper curves and this would indicate that smaller scale structures are more prevalent in downstream grid locations. The correlation plots are generally smoother for forced SAS solution. As expected, the unforced SAS solution exhibits large scale structures only and two point correlations were not obtained as it is essentially a steady solution.

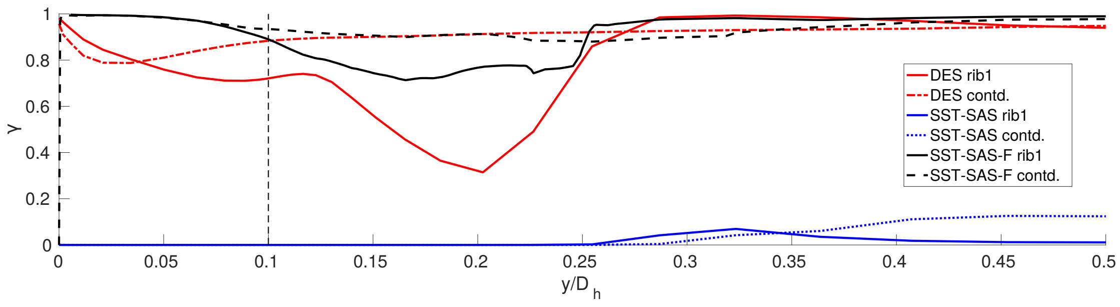

Figure 4.30: Resolved to total Turbulent Kinetic Energy ratio γ. Locations are the same as previously. The dashed vertical line marks height of the ribs. The plots for ribs 2 and beyond are compounded in a ‘continued’ plot as they do not vary greatly. Naphthalene sublimation experimental technique was used.

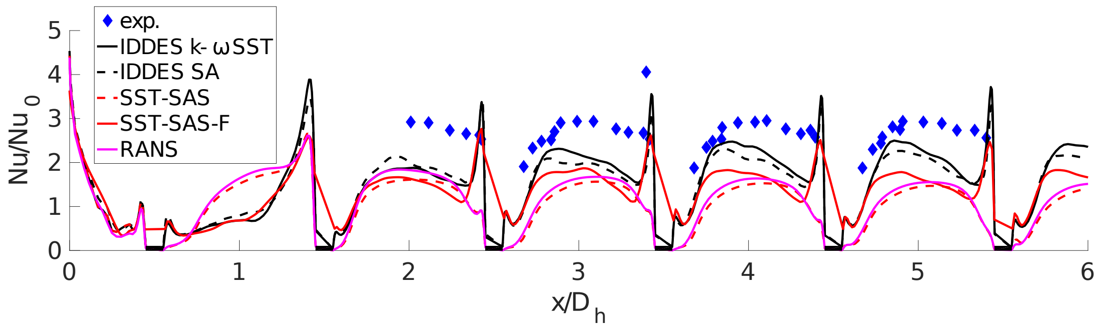

Figure 4.31: Heat transfer measurements and predictions along the wall centreline.

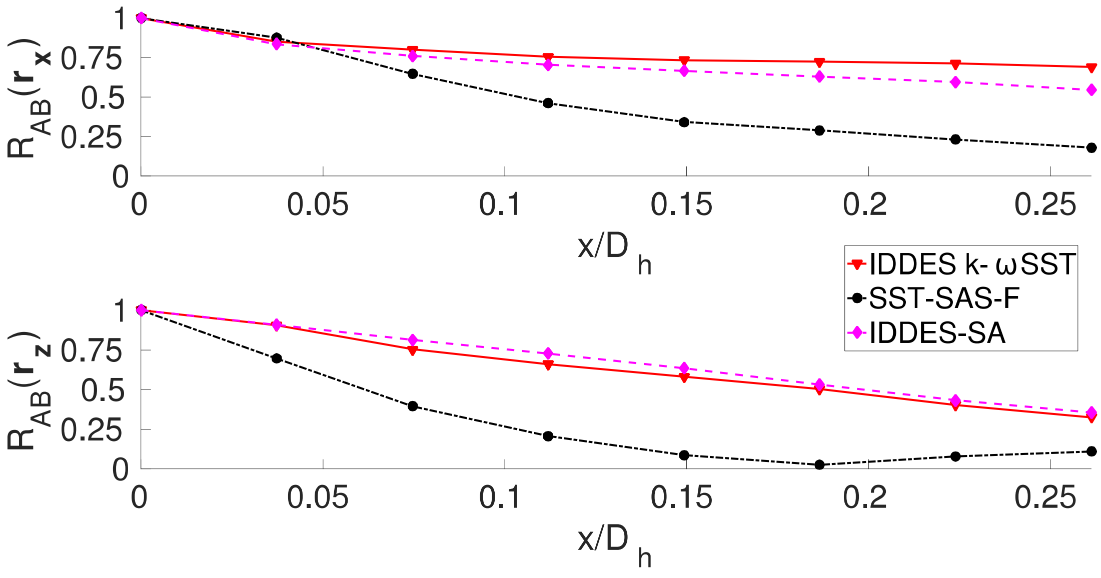

Figure 4.32: Two-point correlation plots. Approximately 5.56mm spacing between the measurement points. The streamwise lines are located in the centre of the channel in z-normal direction and at one rib height in y-direction. Spanwise lines are located two rib heights behind the rib and level with streamwise ones. Grid spacing varies from 2mm to 4mm along the lines.

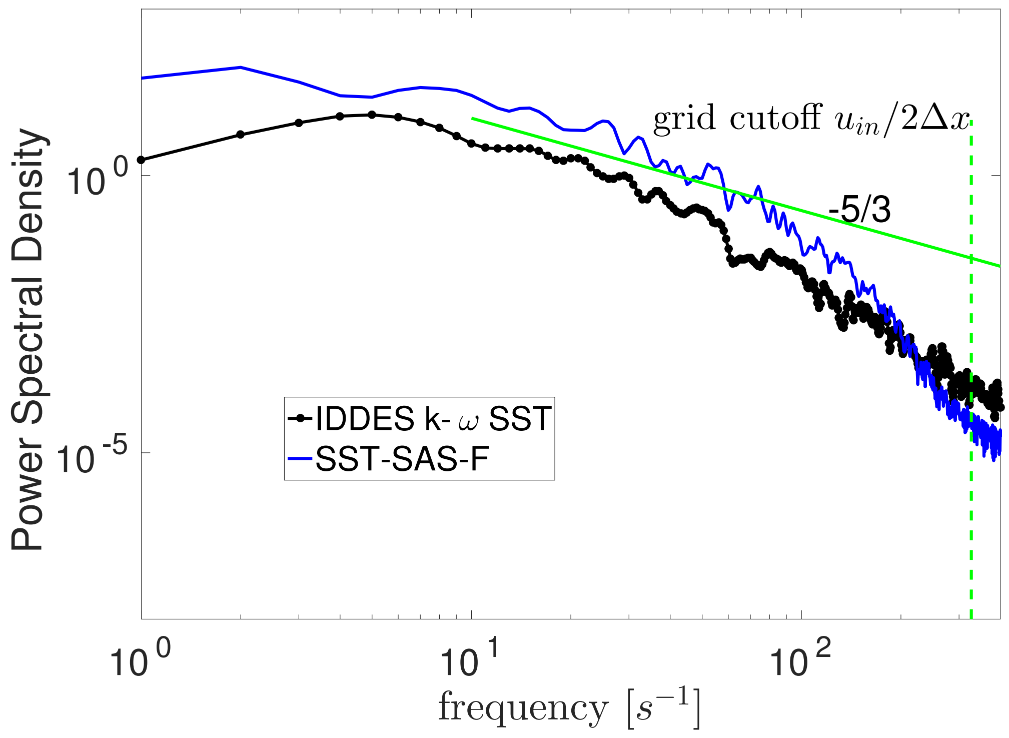

Figure 4.33: Power spectral density for the models tested. The point is located 10mm downstream of 5th rib, one rib height away from the walls and on the z-midplane. The energy spectrum does not appear to be sensitive to location of the point.

4.7 Rotating channel flow

In this preliminary work, a Rotation number of 0.3 was used, which corresponds to 37.3 revolutions per minute, with axis of rotation as shown on Figure 4.21. With added rotations, other boundary conditions are unchanged from the stationary case and similar to LES of Sewall et.al. [185]. Effects of rotation on present type of geometry were also investigated via DES and URANS by Saha et.al. [186]. Experiments were performed

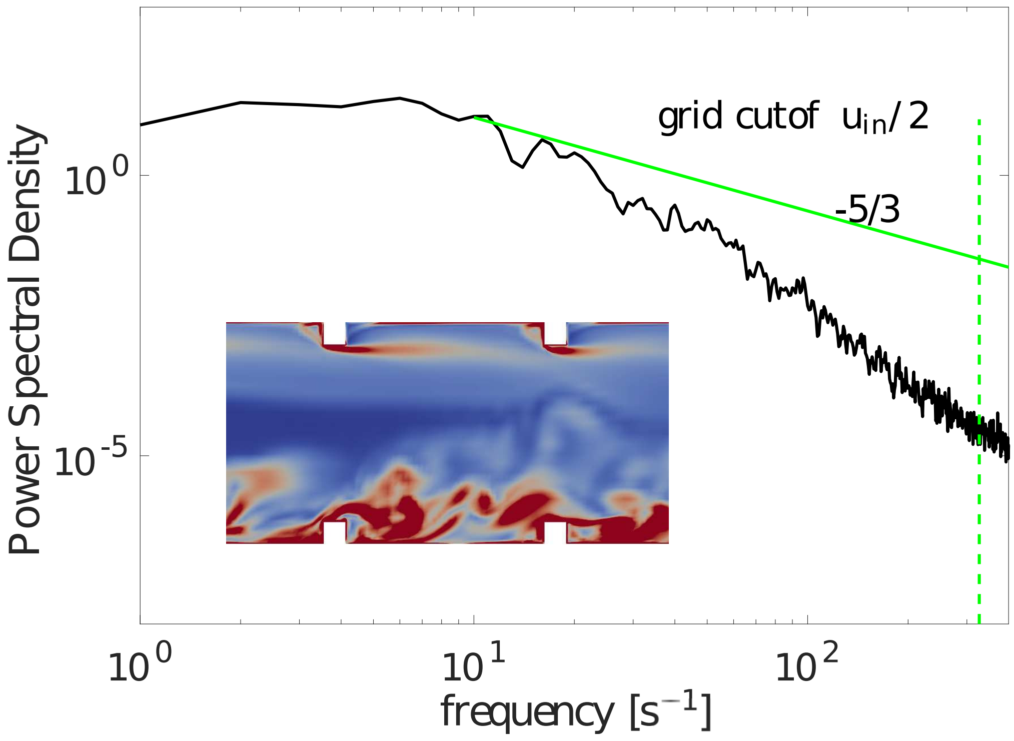

Figure 4.34: Power Spectral Density predicted by the original SST-SAS model with effects of rotation. Contours are of vorticity magnitude at the last time step of simulation for fifth and sixth rib.

by Hibbs et.al. [187]. The following simulations are performed without any additional forcing terms and original SST-SAS formulation.

From Figures 4.34 and 4.35 it can be seen that the solution appears unsteady with a much more realistic energy spectrum even though no forcing terms are active. This confirms the findings of Dobrzynski et.al. [188], who used the unforced SST-SAS model on a cavity flow with rotation and obtained results close to experimental measurements. Figure 4.35 further confirms those observations by comparison with experimental data.

To summarise and compare the findings, Figure 4.36 shows time snapshots of the

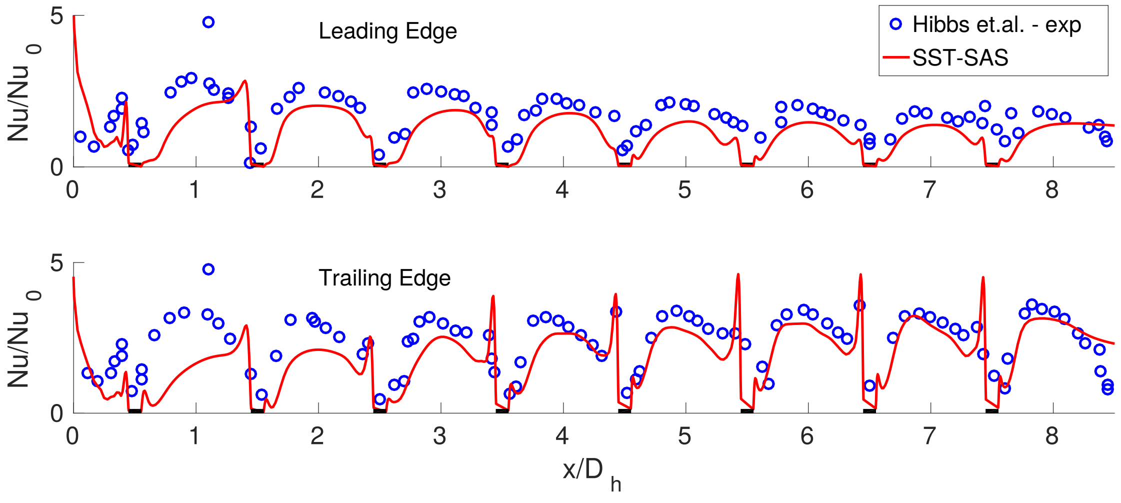

Figure 4.35: Heat Transfer predicted by the original SST-SAS model with effects of rotation.

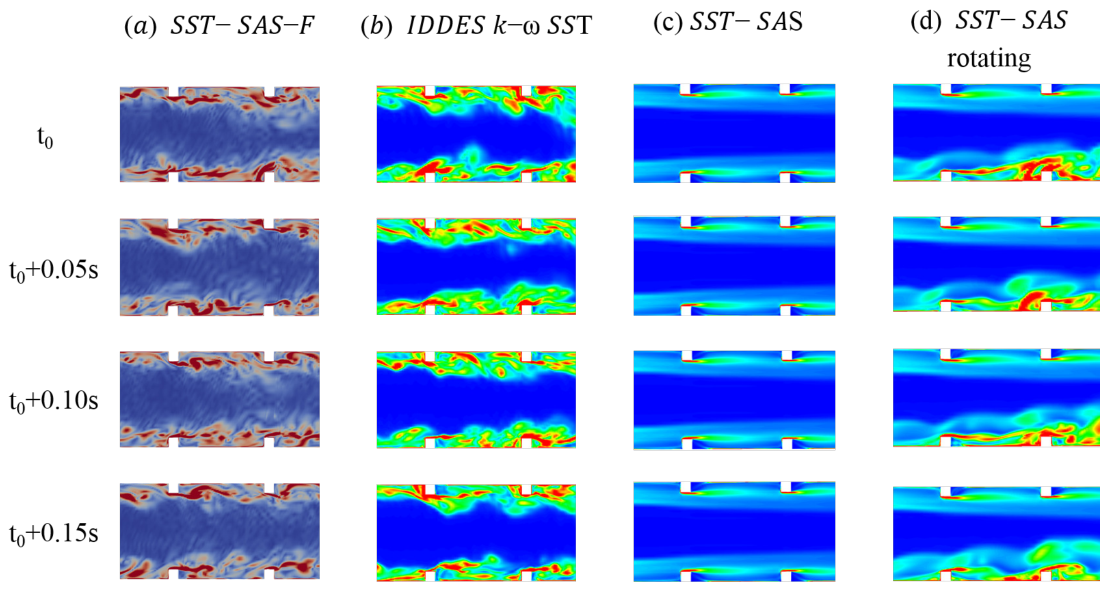

Figure 4.36: Time snapshots of vorticity magnitude for the second and third rib. The initial time varies between the models but is always at least 10 through flows. Time step was 0.00014s in all cases.

flow in intervals of 357 time steps for the different models investigated. The figure shows that unforced SAS solution is essentially steady and does not vary with time. For the case with rotation, the SAS model appears to only be resolving the trailing edge, while leading edge appears as URANS. This is consistent with the expectation that rotation causes turbulence to be attenuated at leading edge and amplified at trailing edge. The forced SAS solution seems to exhibit finer turbulent structures than DES.

4.8 Chapter conclusions

The Scale Adaptive Simulation computational approach was investigated through application to a three-dimensional axisymmetric geometry and ribbed channel and compared with other researchers' simulations as well as experimental data. Both the hill and the channel used are representative of features used inside turbine blade cooling passages to promote heat transfer. Several different CFD models were applied to. It was found that while RANS approaches fail to predict the flow completely, LES appears to produce a reasonable momentum solution even on coarser meshes. The fact that LES appears to be doing well is somewhat surprising as the computational grids are not sufficiently refined as required by standard LES practice. Using SAS was found to offer only minor improvement over RANS for velocities calculation and much larger improvement for heat transfer results. The error in heat transfer however was still large for SAS with respect to experimental data ($> 200\%$).

The size of the recirculation region predicted by LES is similar to that found experimentally in both cases tested here whereas the size predicted by SAS is significantly larger than that of LES, and similar to that of RANS. In addition, SAS predicts a small recirculating region upstream at the foot of the 3D hill, not found experimentally. The DES and RANS models predict a similar result. The flow separation point on the hill occurs at approximately the same point (≈ 0.1H past hill peak) for all models, in agreement with experimental data. Due to geometry the ribbed channel has clearly defined separation point which is similar for all models. The ribbed channel study confirms the findings from the 3D hill. The LES approach also appears to be the better choice of model for the present problem, for both velocity prediction and heat transfer. Overall the evidence suggests for the present geometries there is little improvement in using SAS over RANS, DES, LES or hybrid LES-RANS while incurring noticeable computational penalty for solving the additional terms in the SAS model. LES on the same grid (coarse for LES) produced velocity results significantly closer to experimental data than SAS. Heat transfer is also predicted better by LES.

Moving on to investigate SST-SAS model with forcing terms, with stationary flow the introduction of the forcing terms triggers the resolving capability of the model and the solution is significantly improved, including more realistic spectral content. While the solution is still further away from experiment than DES or LES, it presents a significant improvement over pure SST-SAS. The forcing terms successfully trigger resolving mode with only the modelled time and length scale provided by the base model.

Introducing rotation to the flow seems to trigger the resolving mode of the SAS model similar to that of the SAS-F for the stationary case. This suggests that very strong three-dimensionality of the flow is needed to fully make use of the models' benefits.

Future recommendations include further comparisons of rotating flow against other hybrid RANS-LES models. Friction factors and other more detailed metrics of the flow can also be investigated to further assess applicability of SST-SAS. Test cases where DES struggles to obtain accurate solutions are recommended.

Chapter 5

5 Results: Turbine internal cooling

5.1 Chapter introduction

Now that it was established DES is likely a better overall choice for the present application than SST-SAS, a more complex and realistic geometry is chosen for further evaluations. While the Reynolds numbers and rotational speeds used here are comparable to the geometries in Chapter 4, the geometrical features present in a more realistic blade presented in this chapter create significant, complex and localised turbulence regions. In addition high pressure flow from rotating cooling holes with sharp endings trigger sudden detached flow and mixing into the boundary layer causing complex flow structures to appear close to the wall. It is believed in the industrial turbine blade cooling community these effects can have notable impact on overall blade performance and should be captured accurately. Such high and localised turbulence levels are also the reason SST-SAS was invented and it will be interesting to see if the SST-SAS performs better simulating flow inside the current more complex geometry while it struggled on simpler ones.

Another purpose of this chapter was to explore a single simulation containing all of three key turbine blade flows; the main gas path, film cooling and flow inside the cooling passages. At the moment, all of industrial design process known to the author simulate blade and cooling holes separately and very few simulations include all three together as one continuous domain. While there have been many attempts at unified simulations such as overset, this chapter aims to try a relatively simple approach of a continuous mesh domain. It is believed the CAD and meshing tools have significantly improved to the point where that may just be possible in realistic cost and timescales. Also, much work of this type tends to be commercially guarded as ability to simulate a continuous turbine blade flow domain robustly would give significant advantages to an aeroengine enterprise.

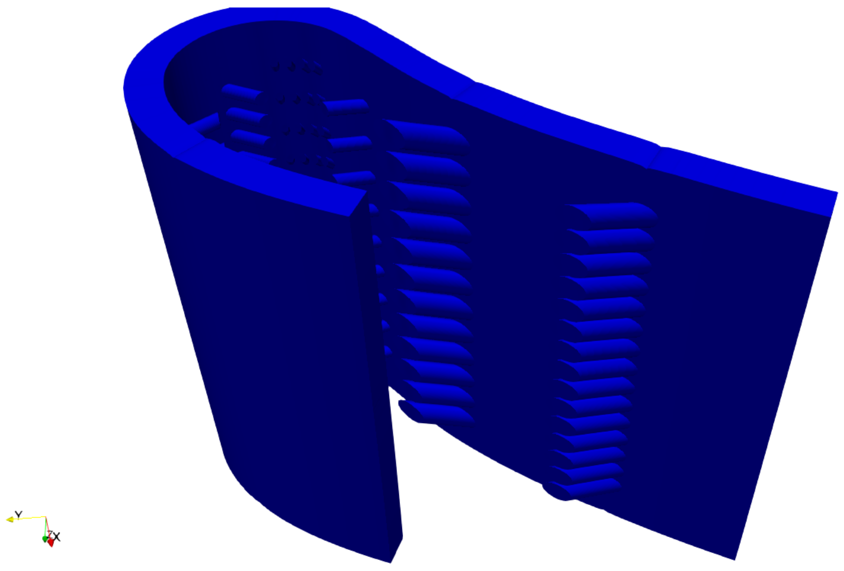

Figure 5.1: Overview of internal cooling hole stubs from the IMB project. This geometry is then superimposed onto a 'blank' blade without cooling holes and the simulations combined using overset-style technique.

5.2 Geometry creation

5.2.1 Cooling holes geometry creation with Siemens NX

After spending several months trying to find the CAD geometry of the internal cooling holes, only an overset-type mesh from a previous project that used the same geometry was found [189, 190]. The technique named Immersed Mesh Boundary (IMB) aimed to mimic the functioning of overset grids where in certain parts of the domain two overlapping volume grids exist and the solution is only progressed on the mesh with higher resolution. A representation of overset-type IMB approach applied to the current geometry is shown on Figure 5.3. The IMB technique was found to be relatively tolerant to CAD and grid imperfections and the mesh obtained needed significant improvements near the boundaries. As a 'watertight' CAD is necessary for most meshing softwares to create high quality meshes with necessary CFD features it was therefore decided to recreate the cooling holes geometry in Siemens NX [191] from the mesh provided.

As can be seen from Figure 5.2 the mesh contains artefacts of IMB technique. While they don't seem to affect the IMB solution as the boundaries in IMB allow flow through freely, they must be first removed for the present purpose as it would interfere with

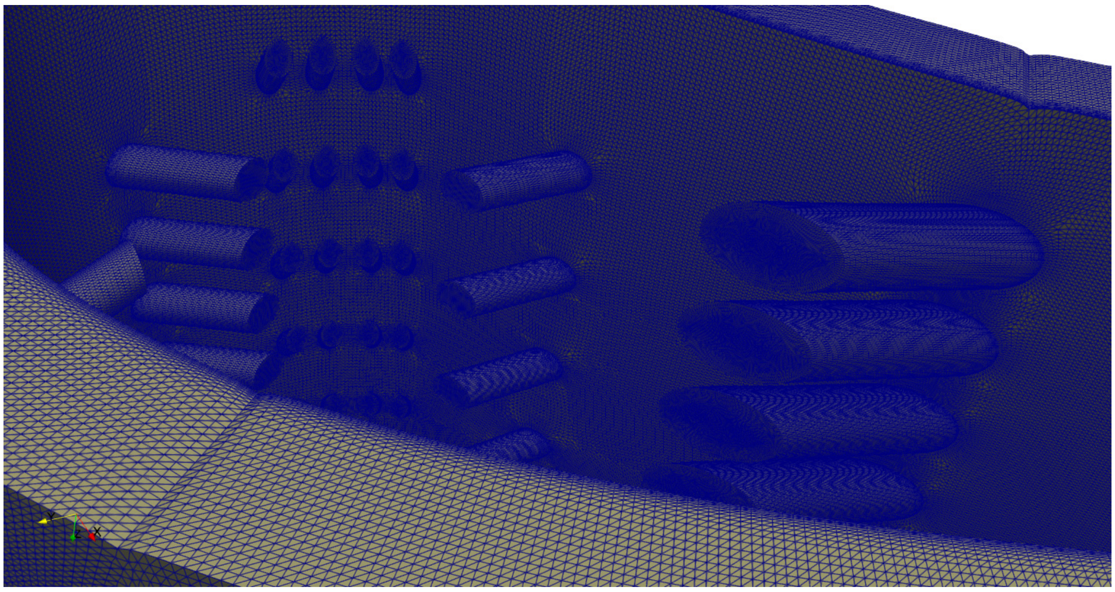

Figure 5.2: Mesh of internal cooling hole stubs from the IMB project.

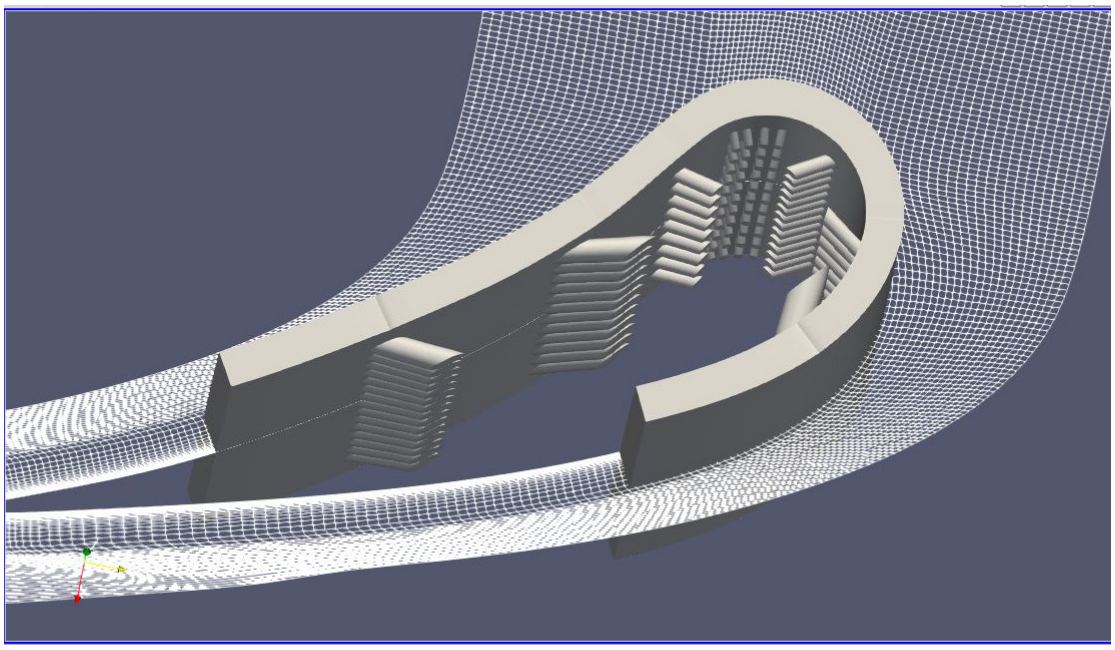

Figure 5.3: Visual representation of IMB technique.

geometry merging and create unnecessary CAD issues.

The cleaned and recreated geometry is presented on Figure 5.4.

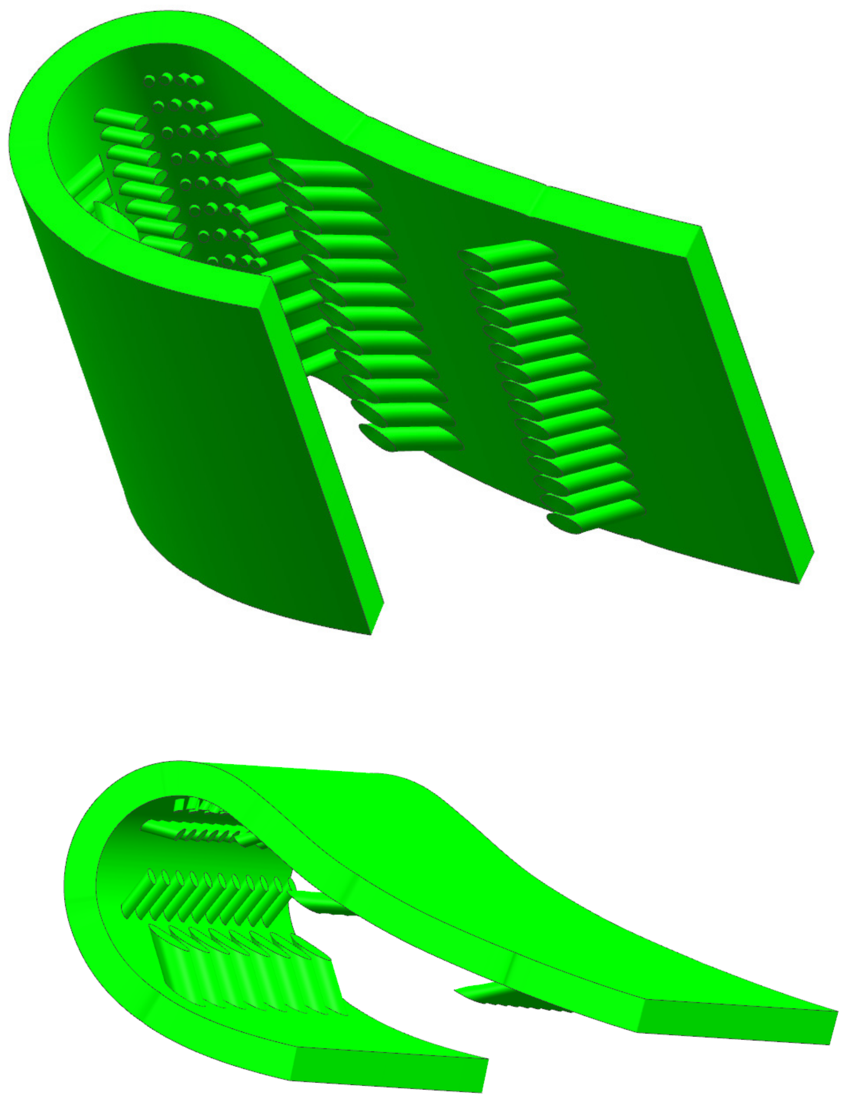

Figure 5.4: Recreated geometry of internal cooling hole stubs.

5.2.2 Blade geometry

Blade geometries are often created and handled entirely within in-house industrial design systems, separate from cooling system and sometimes even other features. The features are often simulated together via 'loosely-coupled' simulations that transfer boundary conditions between two independent simulations. Such was the case with the current blade geometry as it was created in Rolls-Royce program suite 'Parablading'. Meshing and geometry handler within it, PADRAM [192], was used to export the geometry to a standard parasolid format. The full domain is shown on Figure 5.5

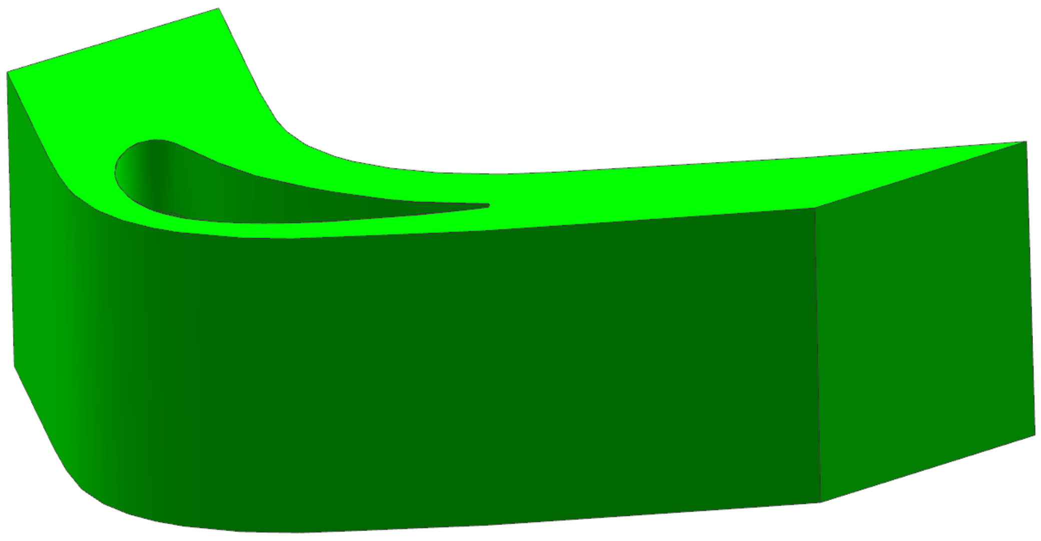

Figure 5.5: CAD geometry of blade without cooling holes.

Next, the geometries must be combined as shown on Figure 5.6. This proved to be not trivial task as the stubs of cooling holes often join the main blade at very sharp angles and each stub is slightly twisted with respect to the previous one, as shown on Figure 5.7. There are 10 rows with a total of 102 cooling holes.

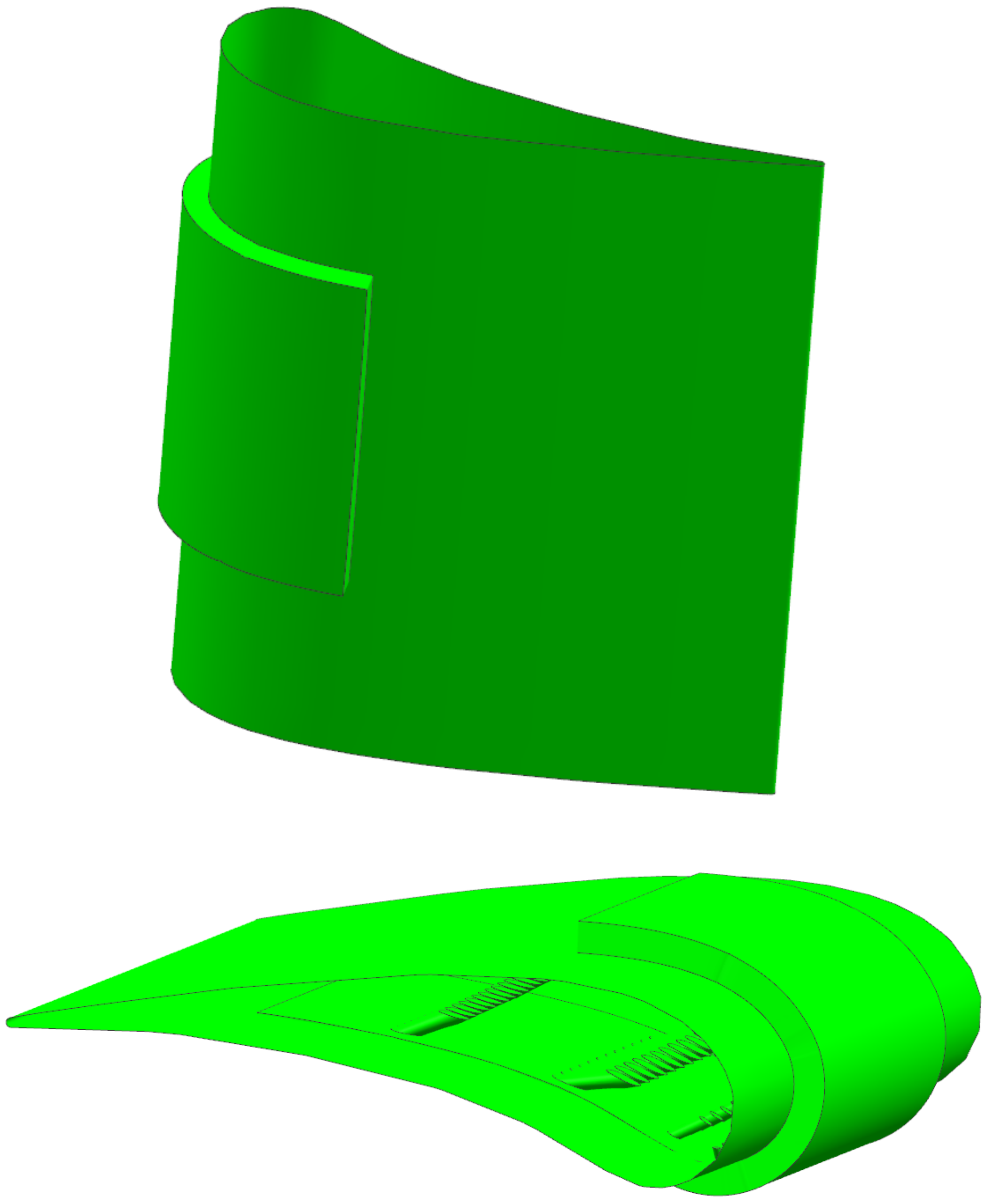
Figure 5.6: Cooling holes geometry superimposed on the main blade.

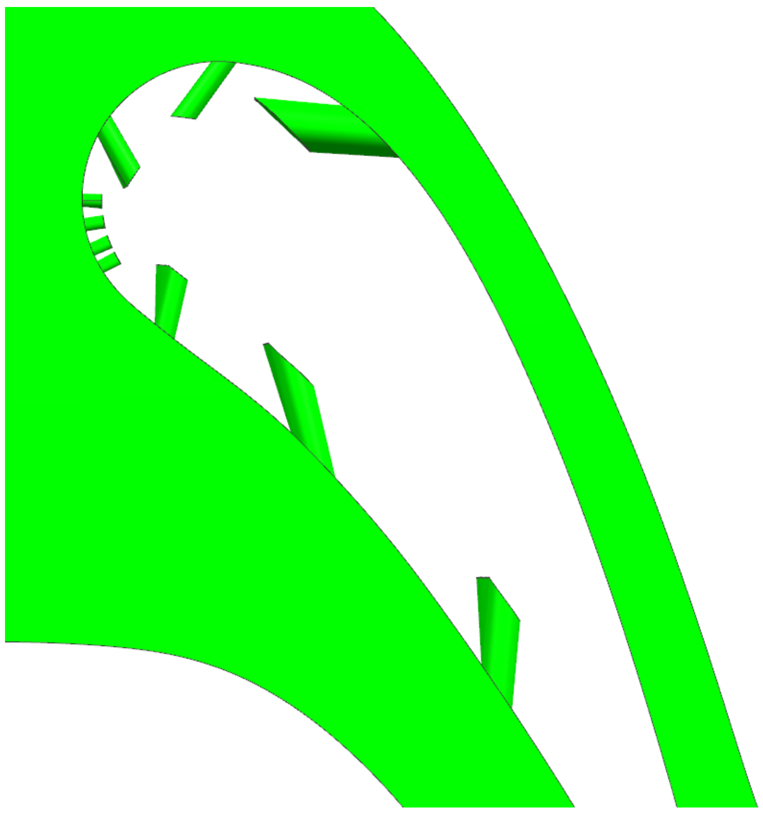

Figure 5.7: Side view into the cooling holes geometry.

The final result is shown on Figure 5.8. Some geometry checks and basic cleanup and surfaces merging was performed in Siemens NX with advice of a CAD expert ahead of meshing.

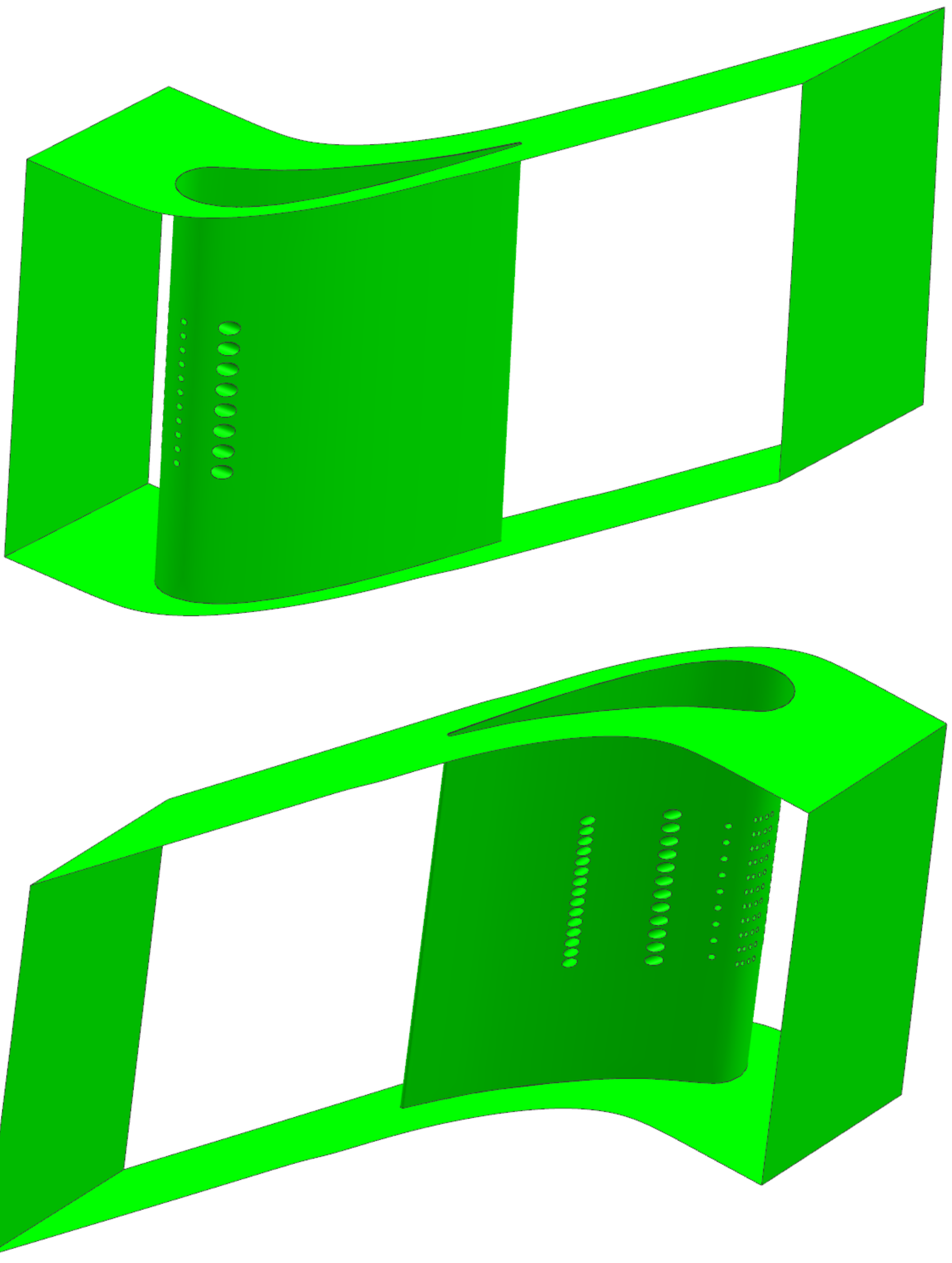

Figure 5.8: CAD definition of the turbine blade complete with 102 cooling holes.

5.2.3 Clean-up with ANSYS SpaceClaim and ANSA

Attempting to create mesh of the fluid domain using ANSYS ICEM as well as ANSYS Fluent Meshing [167] it was quickly evident the geometry is not only non-watertight but also has some undesired CAD characteristic shown on Figure 5.9. For instance the blade surfase was split near the root causing meshing process to fail. Also not all the cooling holes were recognised as cylinders with an axis, making mesh inconsistent and requiring significant manual intervention.

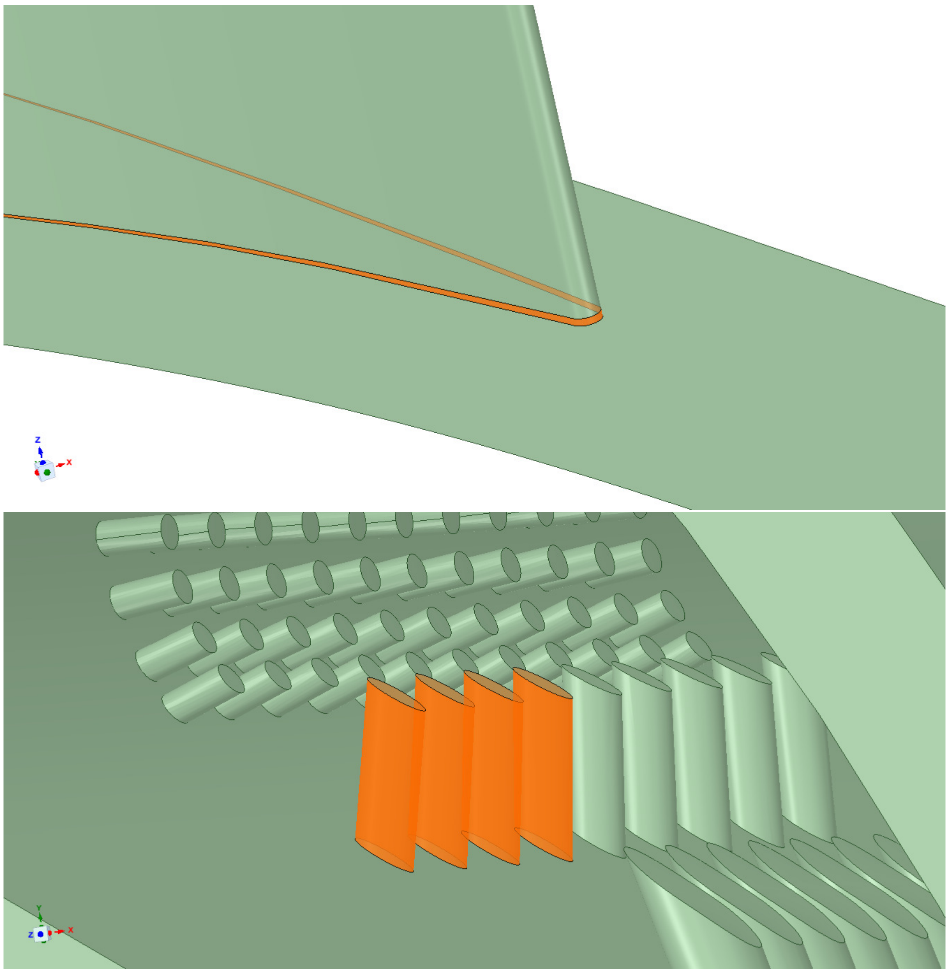

Figure 5.9: Example of geometrical issues with the CAD.

It was quickly evident both manual and automatic geometry cleanup operations with siemens NX will not be sufficient, even with the help of a CAD expert. Some cleanup was performed in ANSYS Space Claim Design Modeller [193] as well as CADFix [194]. After several unsuccessful attempts with ICEM [195] it was decided to try Fluent meshing engine as it offered a robust non-watertight geometry handler. A mesh was finally obtained as shown on Figure 5.10 however it still seemed to have geometry-related issues and not all the cooling holes were meshed in a satisfactory way. It became clear at this stage that cooling holes are going to be an issue and most of the 102 stubs will need to be handled manually for meshing purposes. It was also found at this stage that Fluent Meshing offers very convenient and powerful meshing capabilities but is relatively sensitive to CAD definition imperfections.

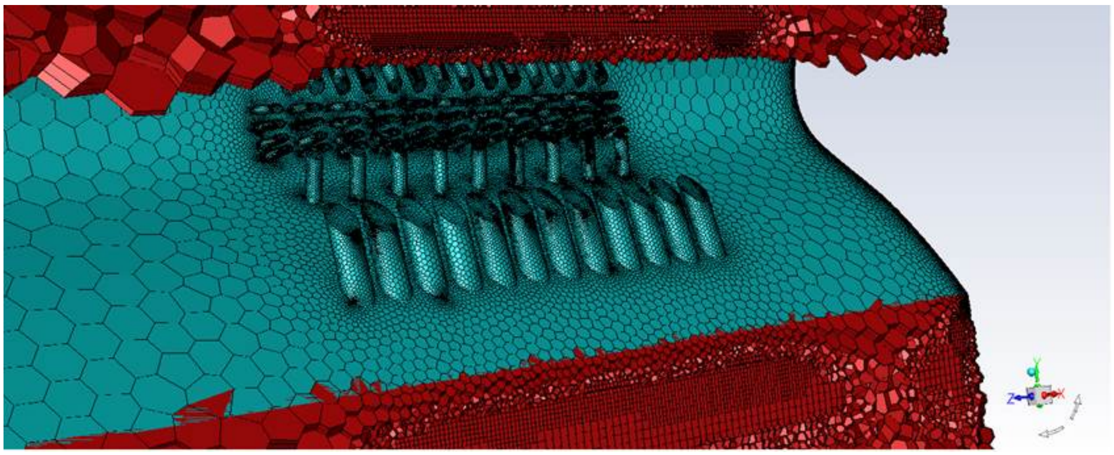

Figure 5.10: Example of ANSYS Fluent ’mosaic’ mesh.

An opportunity to participate in a commercial trial of Beta CAE Ansa [196] meshing tool arose and it was decided to work with representatives of Beta CAE to try mesh the problematic geometry with their software. After preliminary work the resulting mesh still doesn’t have all the necessary features for CFD simulation; however it doesn’t appear to have any obvious geometry issues or extremely high aspect ratio elements either. An example is shown on Figure 5.11. It can also be observed that some cooling hole tubes have different surface refinement than others. This is due to the software not recognising many of the tubes as cylinders with an axis but treating them as generalised surfaces. There are two solutions to this, both unfortunately very manual; one is to remedy that at the CAD level, one is to set each tubes’ refinements during meshing. However, now that a verifiably watertight geometry was obtained, a meshing can be performed in any software.

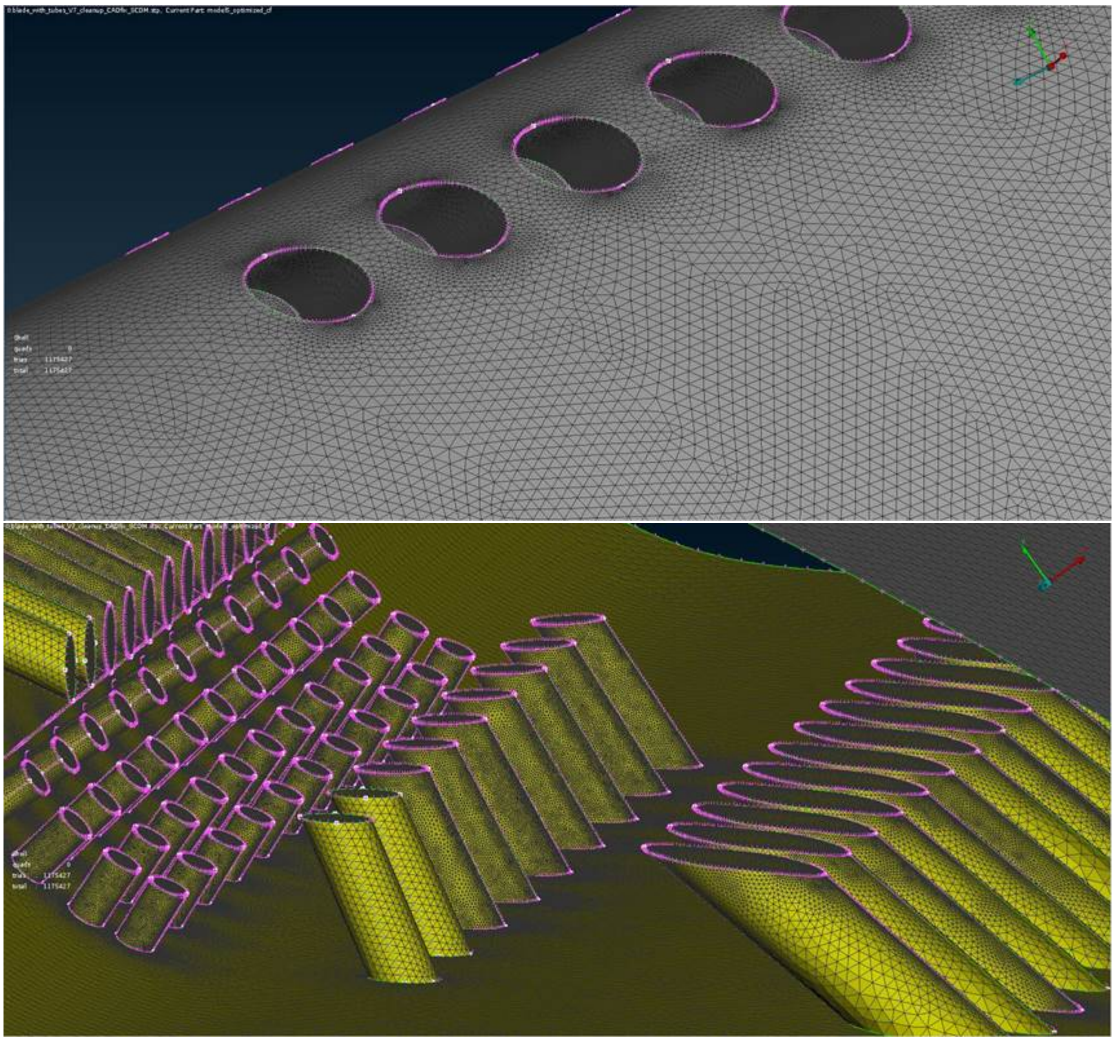

Figure 5.11: Example of ANSA produced mesh.

5.2.4 Meshing

Using ANSA pre-processor, mesh with average first cell value of $y^+ = 4$ in areas of interest was finally obtained and is shown on Figure 5.12 and Figure 5.13. The mesh has a total of 34 million cells. It is likely possible to reduce the grid size further by introducing a hexa blocked structure in the volume domain however it was judged to be very manual and too time consuming. Structured layers are already present in the near wall region via an in-built automated wall layers extrusion tool.

For purposes of grid independence this mesh was then refined by a factor of 1.8 in all directions resulting in grid size of 150 million cells and theoretical $y^+ = 2$. One coarsened grid was also generated, resulting in 16 million cells and theoretical $y^+ = 10$.

In an attempt to further reduce computational cost and take an opportunity to test this

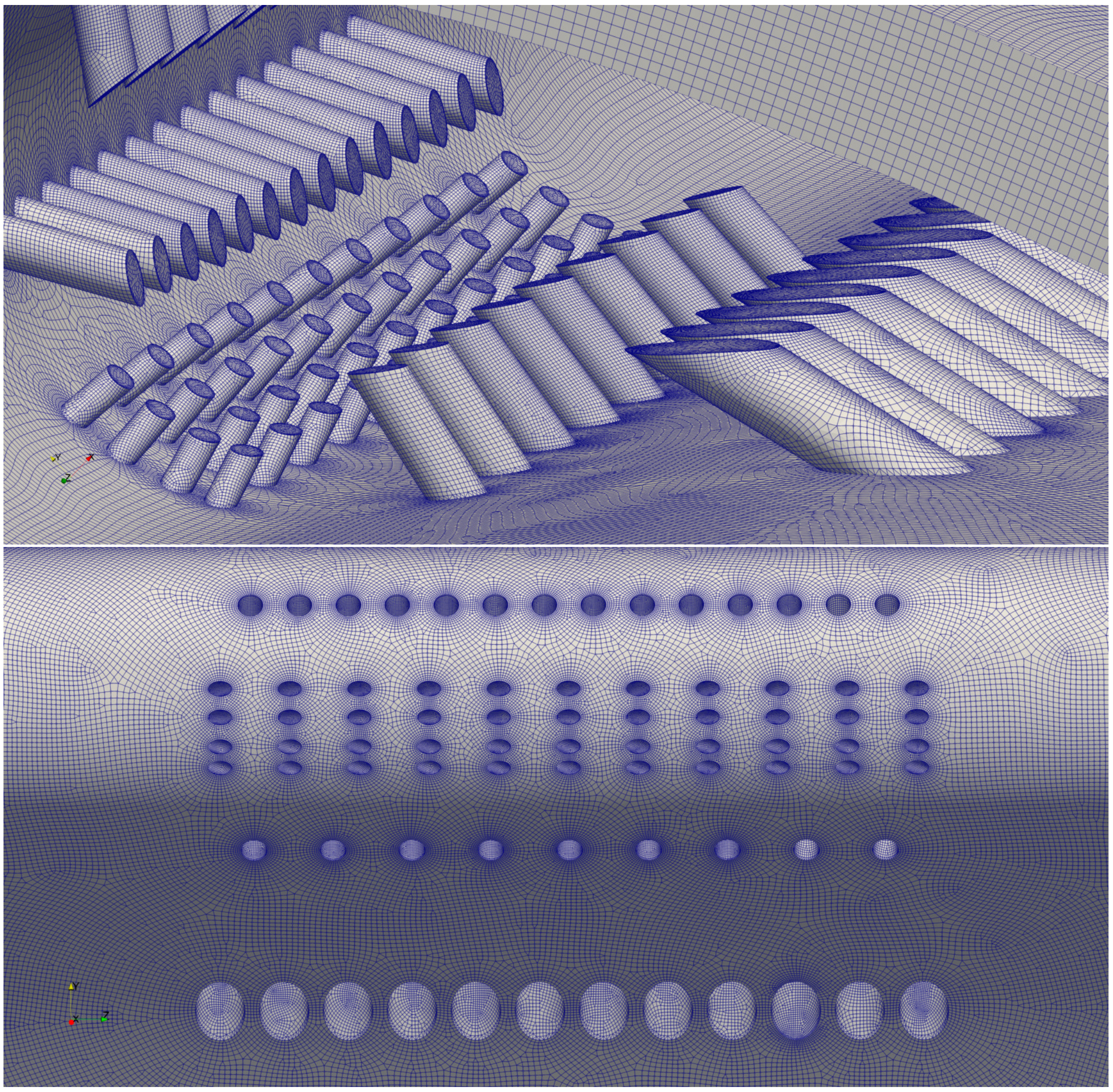

Figure 5.12: Final ANSA produced mesh.

feature, polyhedral conversion was applied to the 'base' grid of 34 million cells. This was done using ANSYS Fluent and driven by the CFL number and pressure field obtained from the RANS solution presented in the following subsection. This resulted in 18 million cell grid. While the polyhedral conversion seemed to not produce different result to the base grid in steady RANS simulation it was not expected for this grid to perform well with a hybrid RANS-LES technique such as DES. The polyhedra exercise was done primarily to test the feature for future use if necessary.

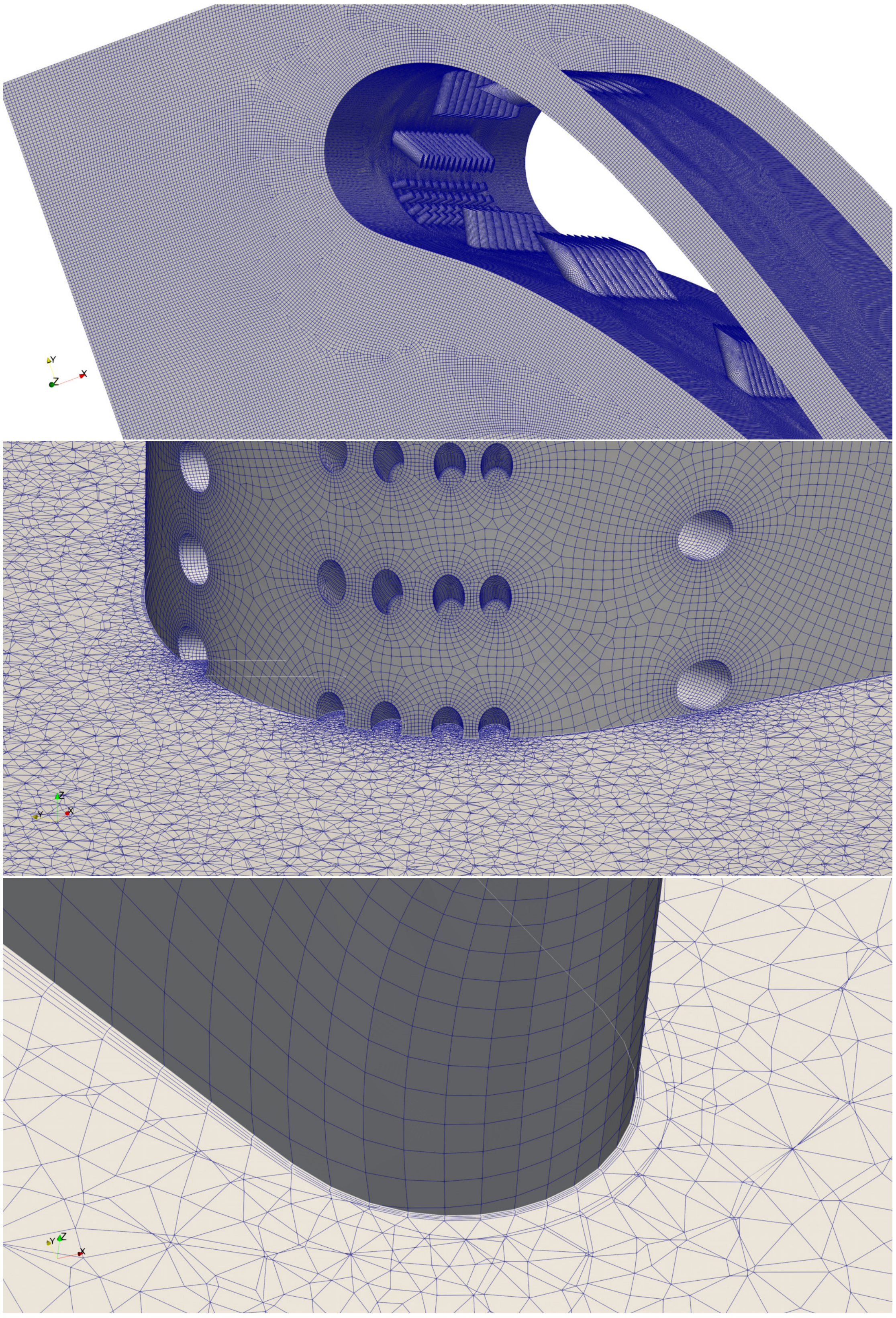

Figure 5.13: Final ANSA produced mesh.

5.3 Computational setup

5 compute nodes with 24 physical cores each were used resulting in 120 ranks for parallel ANSYS Fluent. Spanwise-averaged quantities were obtained in the same way as in [3] as well as the experiment. The area is shown on Figure 5.14. It is slightly different for heat flux and mach number averaging and is not exactly in the middle of the cooling hole region but the region is consistent with experiment and calculations of [3]. The procedure for obtaining the averages was to first, using Paraview, eliminate all but the relevant region of the blades surface (shown in blue), then using the slice filter segment the averaging area into spanwise lines (bottom of the figure). All flow quantities were interpolated onto those lines and the data exported to Matlab. From there it was a simple matter of taking an arithmetic average of all the points on each of the spanwise lines and plotting them as a function of perimeter of the blade.

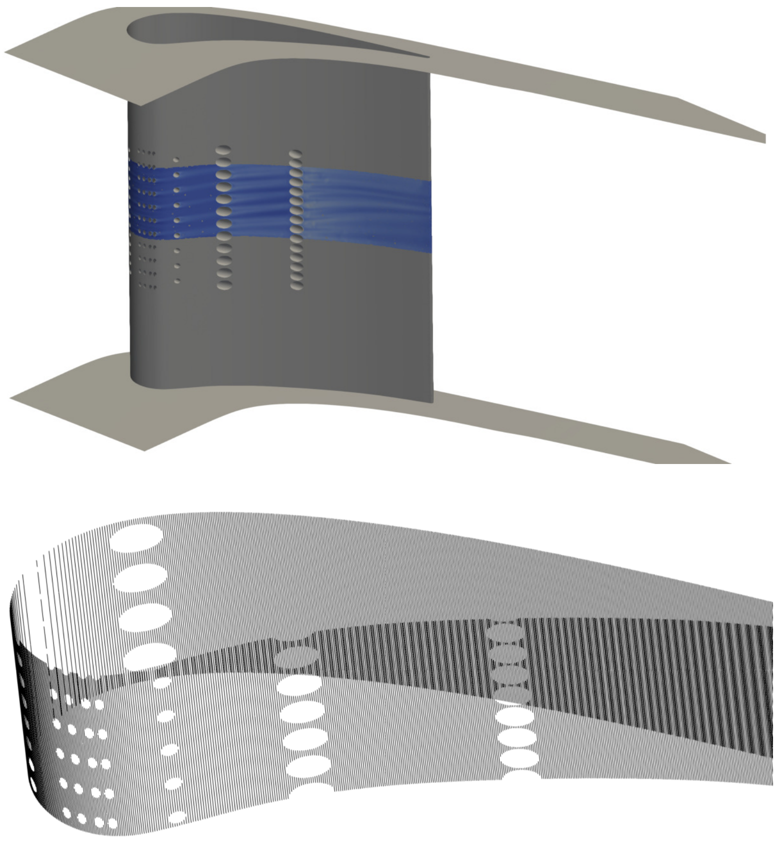
Figure 5.14: Representation of area used for spanwise averaging.

5.3.1 Boundary Conditions

Majority of the boundary conditions were replicated from Chardonnier et.al. [3] and are consistent with experimental setup. The main difference in present work is lack of plenum as shown on Figure 5.15. Constant pressure was therefore applied at the entrances of the cooling hole stubs. This is in contrast to a more realistic distribution as shown on the Figure 5.15. While the internal passages are not engine-representative their existence alone makes considerable difference to the flow, as advised by industrial internal cooling experts and also confirmed by the paper [3]. Unfortunately passages' geometry was not available and there was not enough information available to replicate them easily. Combined with time pressure, it was decided to simulate the geometry without the passages as this would still likely provide a meaningful comparison between the hybrid RANS-LES models.

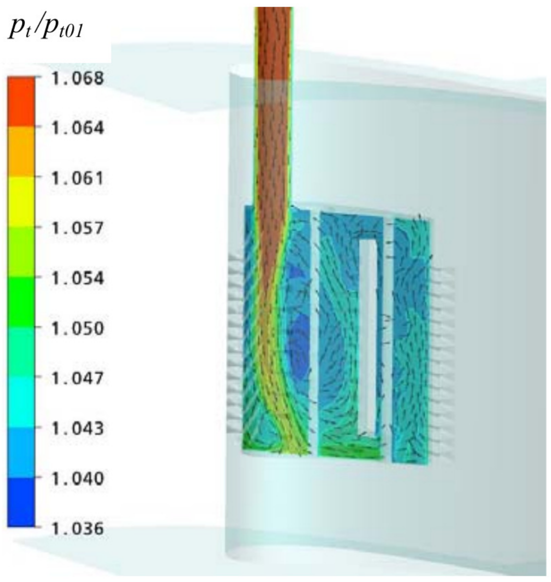

Figure 5.15: Plenum representation from paper of Charbonnier et. al. [3].

Another, likely less significant, difference is that the coolant used in [3] and EPFL experiment is Carbon Dioxide. CO_2 has 65% higher density as well as different thermal properties and this has been accounted for by adjusting the mass flow into the cooling holes. To simplify the simulation however no multi-phase calculations were performed and it was assumed that mass of CO_2 is insignificant with respect to air in main gas path. This assumption is not ideal and was not validated but allowed the author to perform the simulation in realistic timescales.

5.4 RANS results

RANS result using the $k-\omega$ SST model was obtained as a starting solution to further unsteady simulations but also to validate the current computational configuration. Due to the Boundary Conditions differences mentioned previously this result, especially the thermal equation, cannot be expected to be identical to experiment. It does however show consistencies in surface distributions and trends on spanwise averaged quantities as shown on Figure 5.16, Figure 5.17 and Figure 5.18. It will also provide a good foundation on which to perform the further unsteady simulations.

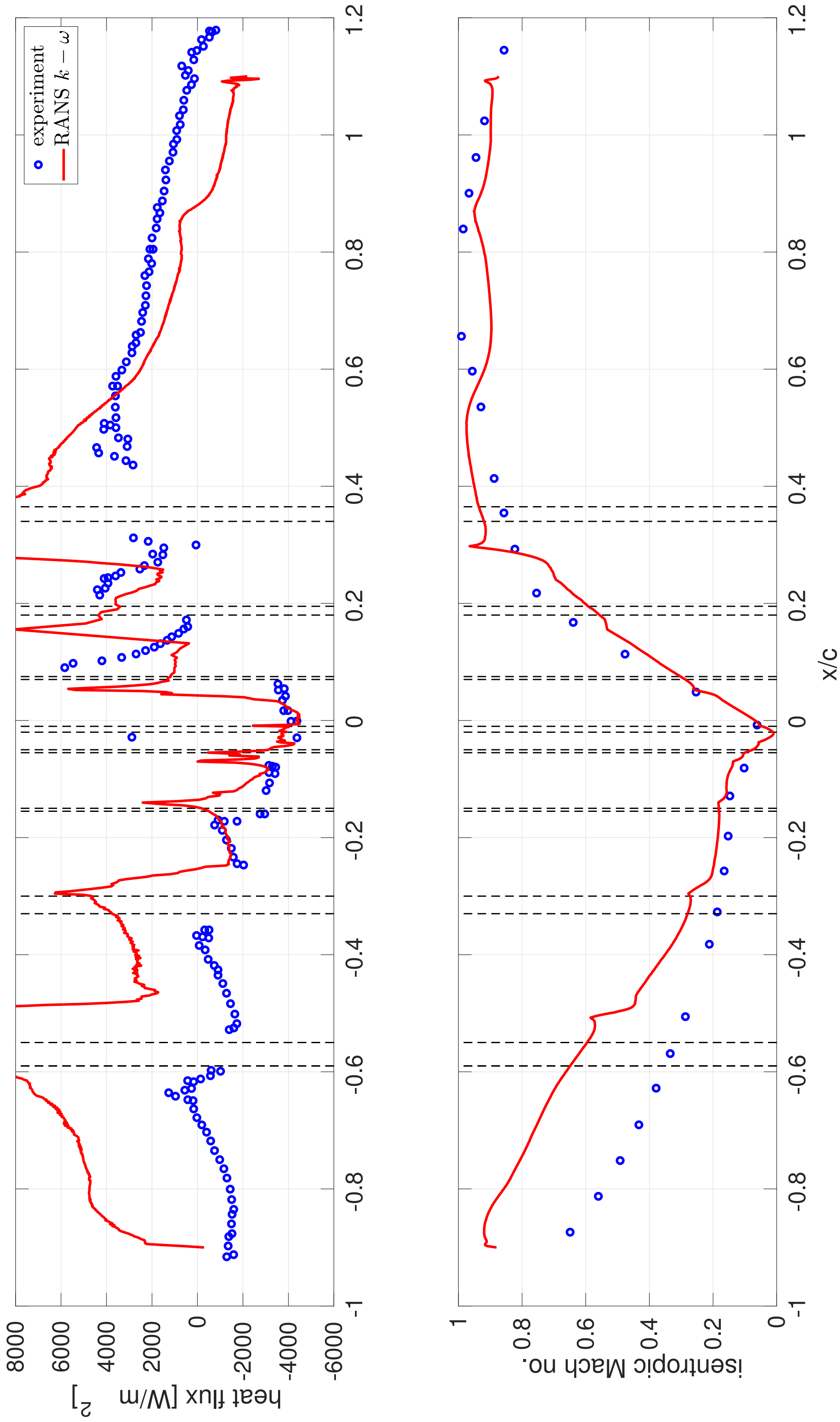

Figure 5.16: RANS results vs. experiment. Vertical dashed lines represent locations of the cooling holes.

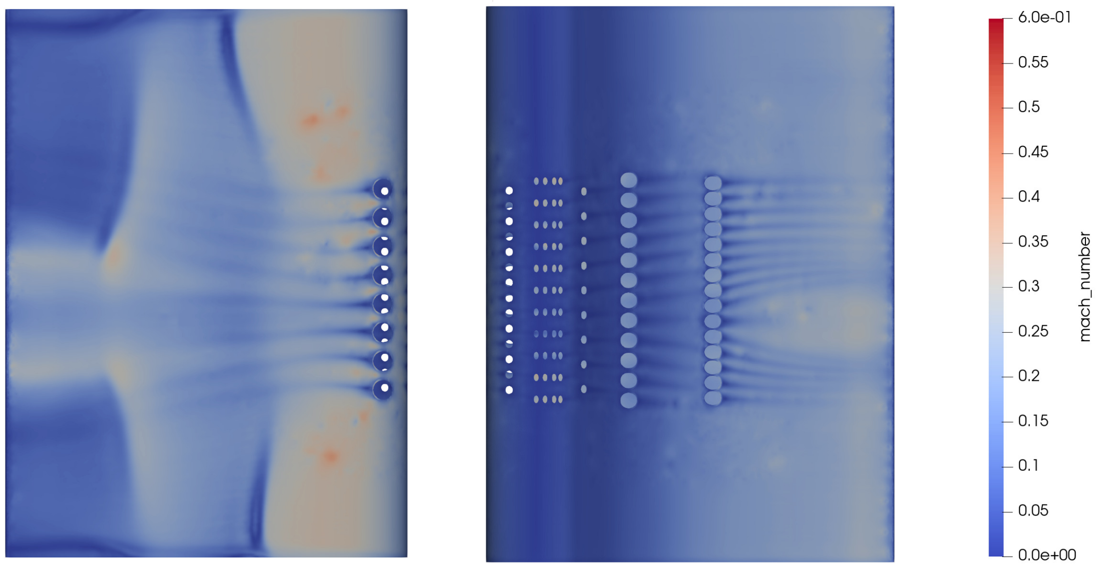

Figure 5.17: Mach number contours.

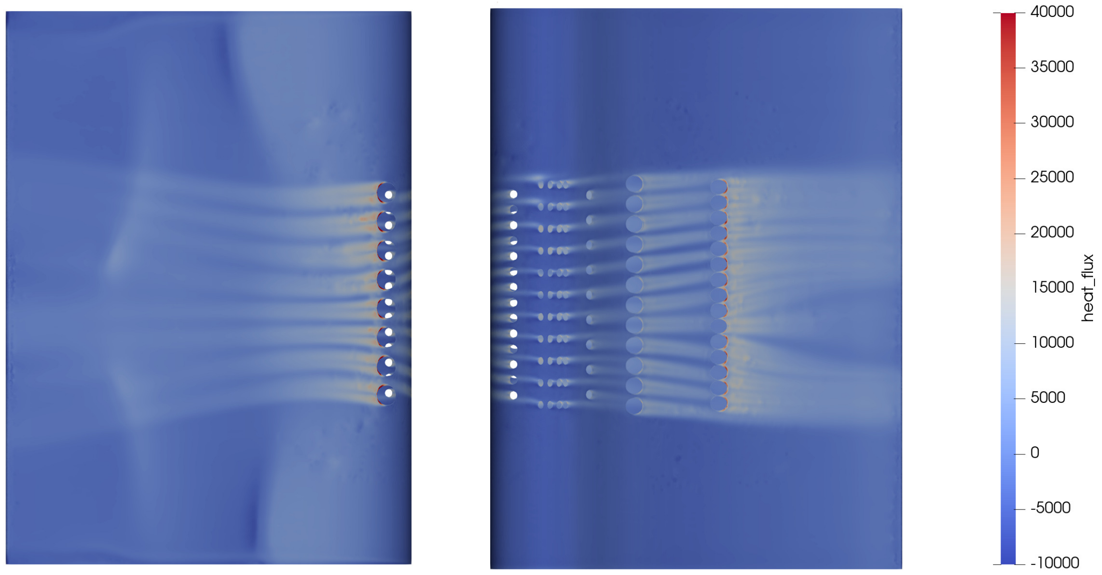

Figure 5.18: Heat flux contours.

5.5 DES

Computational benchmarking shown based on current grid and required timestep of 5.4×10^{-5} it would take 4 weeks on 120 cores on CFMS HPC for the flow to go ten times through the domain. At least three 'flow-through' times are usually required for the flow to develop a fully unsteady characteristics when starting from a RANS solution and at least another three to six flow 'travels' for temporal averaging to settle. While these timescales don't guarantee a well averaged solution, from experience they are usually the minimum for this type of simulations. The bulk of the mesh, as is typical, is in the near wall region where refinement to low y+ is done. This can be somewhat reduced but it is well known that present application of turbine cooling is sensitive to this and y+ should be kept to ideally 1.

After many attempts to obtain a sustained unsteady solution, the simulation was found to not be stable enough and the solution always ended at convergence issues and unphysical flow. After detailed inspection most problematic areas were found to be those where the cooling holes meet the boundary layers of the main flow. Cause of the issue is likely twofold; flow in this area is highly turbulent, three dimensional and irregular or mesh with high aspect ratio and sharp edges. While Reynolds Averaged simulation appears to be able to handle sharp mesh near hole inlets DES clearly struggles to converge there. Is is likely the meshing step has to be revisited to improve quality and structure of the mesh and potentially refine further to obtain stable DES solution. The author eventually ran out of allocated time and the work had to cease.

5.6 Conclusions

Majority of work shown in this chapter has been the preprocessing legwork of CFD. While RANS results were obtained, unfortunately the work has not resulted in a satisfactory DES simulation in the time frame given. The work did however highlight that meshing and geometry handling is a significant issue for present application and this difficulty is only going to increase with realistic internal cooling passages or more complex blade features. Despite there being 102 cooling holes of different shapes, current geometry is still very simplified with respect to components commonly certified for commercial aeroengines. Other works such as overset and IMB [189, 190] simplify greatly the meshing but introduce other issues and are a different technique fulfilling different requirements altogether.

Achieving stable numerical solution was also difficult, judged largely due to locally inadequate mesh exacerbated by the high speed flow.

Another, perhaps less scientific, conclusion is that obtaining or recreating geometry from previous works is less than trivial and must not be underestimated. Any contribution to knowledge rests on such practical points which sometimes even prevent any contribution at all.

But even work like this results in useful insights; it is now clear that a radically different approach to simulating flow around complex internal geometries is needed. Such rethink and development of alternative method to simulate problematic geometries in environment of fast paced component design timescales will be offered in the next chapter.

Chapter 6

6 Results: GPU IBM program

6.1 Chapter introduction

As demonstrated in the previous chapters, our current computational tools appear to struggle to address the challenges of turbine blade design well. While it is not impossible to simply use the current industrial state of art, obtaining quality solution requires large amount of time and expertise, making it impractical and preventing widespread use. Work presented in this chapter aims to create a path towards unique, step-change, solution to a unique set of problems encountered in the field.

The solution proposed is a General Purpose Graphical Processing Unit (GPGPU) Immersed Boundary Method (IBM) – based program where a standard square 'background' grid is generated with minimal user intervention and a user-specified geometry is imposed onto the Navier-Stokes solution via Ghost Nodes. The Navier-Stokes solver only operates on the fully structured 'background grid' and each node is checked for proximity to the user provided geometry. The numerical solver is appropriately modified to make it behave as if the geometry was physically there. This introduces a number of benefits, such as easy grid generation and ability to solve any arbitrary geometry on a structured grid. The latter enables the code to be readily portable to different architectures, such as GPGPUs, with little performance penalty and without introducing numerical or computational complexities. Accuracy of the method will also be addressed via high order schemes and verification and validation exercises. This chapter will present results of two relatively distinct projects required to make the technology work.

One will be a set of preprocessing algorithms that will detect location of the nodes relative to geometry, compute Immersed Boundary coefficients for the Ghost Nodes and ensure the problem is well posed for the solver. Methodology of obtaining distance to the nearest wall must also be arrived at and tested as most turbulence models require wall distance parameter. This is less trivial task using IBM then with geometry-conformal grids and accuracy must be preserved over even very small distances near the walls.

Second is development of the solver itself that will be adapted to make use of these coefficients with the finite difference method. The usual issues of computational efficiency and stability need to be addressed. While it is possible, and likely better long-term, to create the two steps in one software, separating them allows experimentation with different types of Immersed Boundary methods via the standard interface between the preprocessing and the solver. It also allows the use of different underlying languages and scripting before the final optimised configuration is found.

In addition, a practical consideration of code portability leads to the use of OPS library where the computational and scientific issues are treated separately. A single source code is written in a high level domain language and automatically translated to all required hardware platforms with the use of OPS language.

This chapter will present development of the method, rationales behind the methods' choices and validation with analytical and experimental data.

Summary of the key results and highlights of the methods' development from this chapter have been peer reviewed and published in ASME Turbo Expo conference with paper number GT2020-15844. The work was also attempted to be published in ASME 2019 as GT2019-90265 however it had to be withdrawn as no author was able to attend the conference in the end.

6.2 IBM specific preprocessing algorithms

This section will present a number of algorithms that must be implemented and validated prior to solving the Navier Stokes equations. Turbulence models require distance to the nearest wall at each node, Immersed Boundary Method requires interpolation coefficients, accurate identification of nodes inside and outside of the solid domain, Ghost Nodes need to be identified, vectors normal to the boundary computed and Image Points identified. The complete list is long and all the steps will be shown. In this section results of all the prerequisite algorithms and calculations will also be presented. Poisson equations will be solved and validated.

Additionally since central second order spatial discretisation is used some numerical dissipation must be introduced to stabilise the numerical solution of the Navier Stokes equations. This chapter will also present comparison of artificial dissipation schemes.

Finally, parallel scaling will be shown and compared to serial computations as well as various different GPUs.

This chapter is essentially intended to validate all the algorithms necessary to be able to solve the Navier Stokes equations on realistic geometries using the Ghost Cell Improved Immersed Boundary Method of Chi [30].

6.2.1 Meshes generation

The procedure starts by generating two grids in a common coordinate system, as shown on Figures 6.10 and 6.11 or 2.1. Firstly, a geometry representation is created as shown by dashed line on the figures; in practice this is a csv list of (x,y) pairs for a two dimensional simulation or a commonly used Stereolithography (STL) representation in three dimensions. While certain level of points density is required for the node identification algorithm to work correctly this step usually does not tend to pose a major challenge regardless of geometry complexity. STL representation were generated by Creo CAD suite [197] but can even be created by a postprocessing software such as ParaView [198]. The two dimensional csv files were created using the slicing and projection tools in Paraview for complex shapes and Matlab for simpler ones where analytical expressions exist.
The second step is to generate the background grid for Navier-Stokes computations. Those grids must be given more thought as the usual recommendations of y^+ and general best practices apply. Ansys ICEM CFD [195] and Pointwise [199] software were used for this task and the geometry generated in previous step was first imported as shown on Figure 6.5. The reason for this was to aid generating suitable, but not excessive, wall refinements; as the grids are generated independently, not having a clear geometry representation while creating the background grid would make the task problematic and result in suboptimal mesh. As can be seen on 6.10 the IBM grids closely resemble body conforming type grids in terms of near wall refinement and expansion ratios. One notable difference is that grid cells and points exist inside the geometry as for instance shown on Figure 6.5. There is no flow inside the geometry but those cells are still parallelised across compute ranks and computations, however unphysical, are still performed there. This creates a penalty in the present application however presents an excellent opportunity for conjugate heat transfer computations as a natural extension of the method.

6.2.2 Nodes in/out and GN identification

Having both meshes generated, the very first task on which all else is built is to identify two types of nodes:

- nodes of the background grid located inside the geometry defined by the user. This step will assign either a 'solid domain' or 'fluid domain' tag to each node of the grid.
- Ghost Nodes, here defined as nodes of the solid domain which have at least one neighbour in the fluid domain. The neighbour can be either +i, +j, or a combination of them. The reason for including diagonal nodes in the search is that some more advanced smoothing / filtering algorithms use it to determine dissipation coefficients. Only one layer of Ghost Nodes is required for most 2nd order schemes however expanding to increase more 'layers' of Ghost Nodes is straightforward once the program is written.

The main difficulty here was node identification. There are many algorithms to accomplish this, ranging from ray tracing, kd-tree or by 'brute force' simply computing distance to curve at each of the nodes. Brute force here was chosen for simplicity of implementation and debugging and performance was found to be acceptable. The structured grid can easily be parallelised, user geometry is small so can be stored on each core and no communication is necessary between the grid nodes. This makes the brute force method attractive for early research where speed of implementation is prioritised over 'production grade' optimisations. In addition, MATLAB was used to implement the method, which further eased the parallelisation via its in-built shared memory approach.

Results of the computations on a sample cylinder and array of such cylinders are shown on Figures 6.1, 6.2 and 6.3. It was found that density of at least 4 points of the user defined geometry within one background grid cell were necessary to achieve robust result. Overall the program appears to be able to handle any number of geometries of any arbitrary shape once the user geometry is sufficiently refined, i.e. if multiple shapes exist they do not have to be explicitly defined separately, the algorithm is general enough to recognise when a node is enclosed.

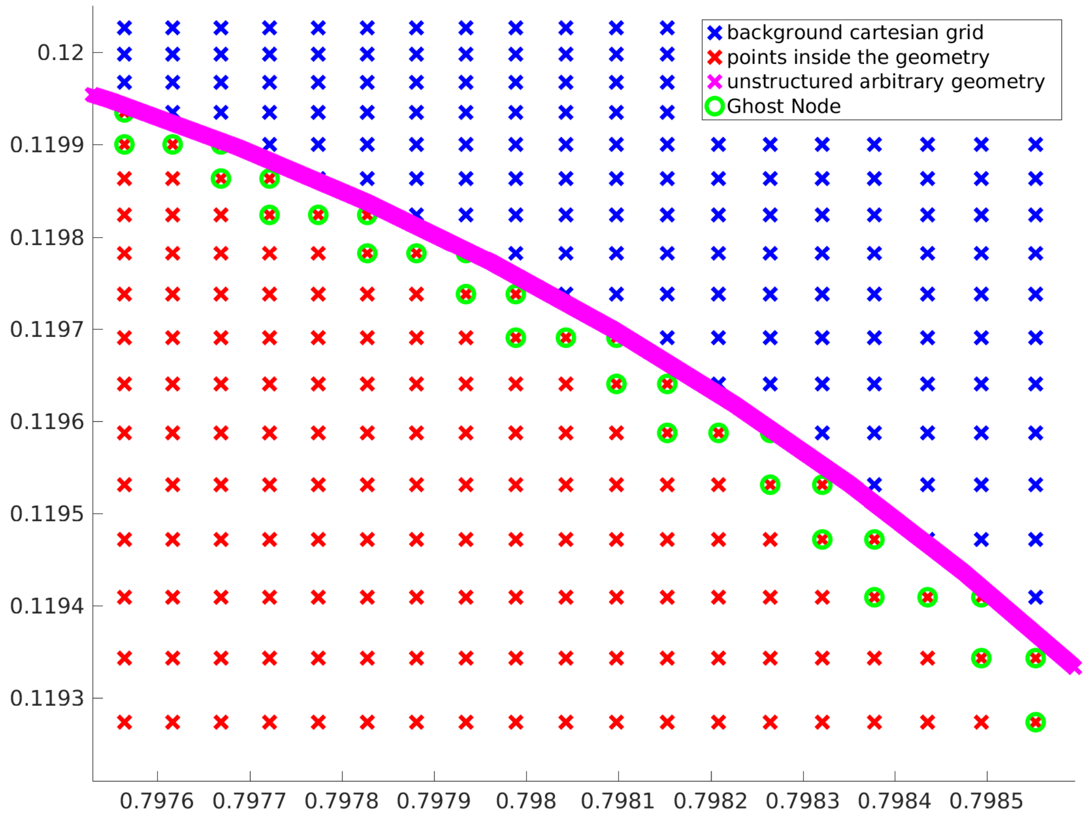

Figure 6.1: Example result of node identification algorithm on a cylinder section: detailed view.

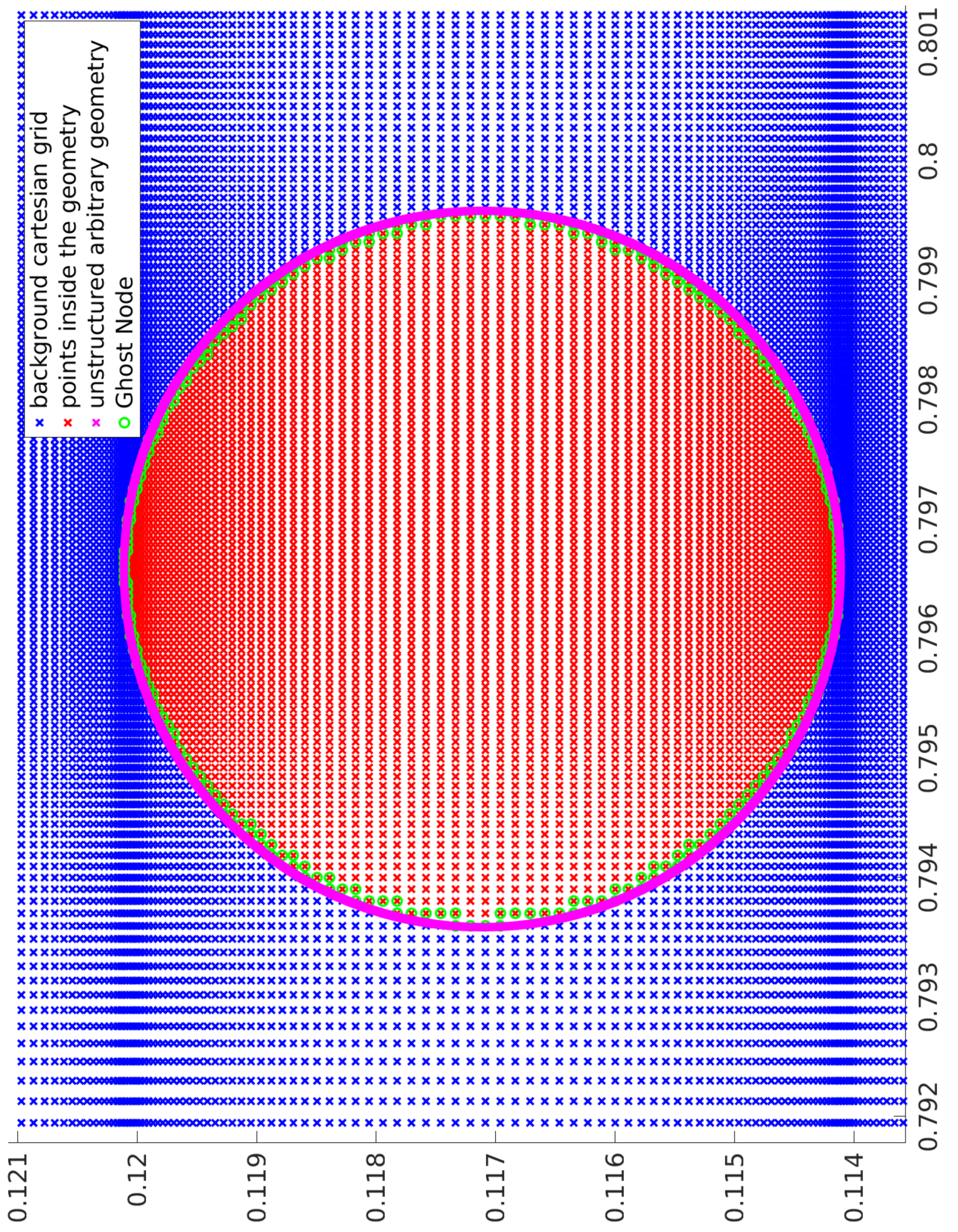

Figure 6.2: Example result of node identification algorithm: single pin.

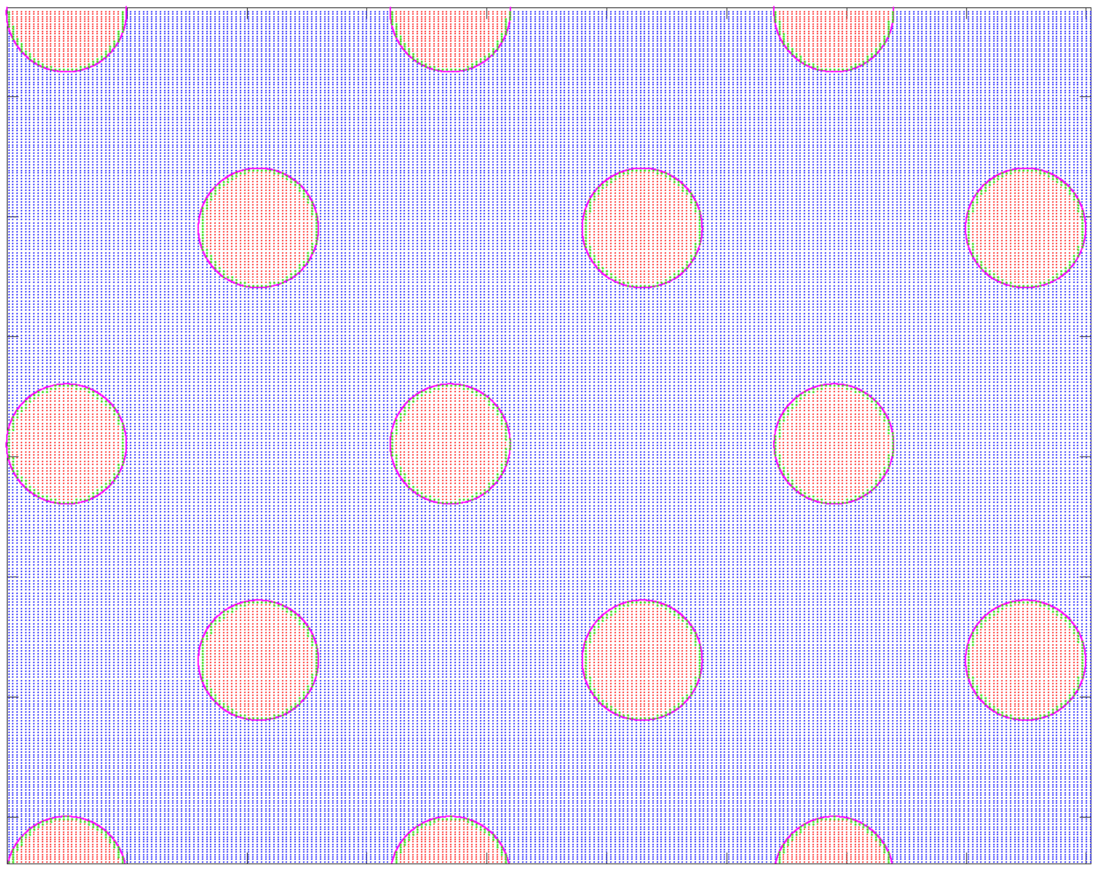

Figure 6.3: Example result of node identification algorithm: single: array of pins.

6.2.3 Normals, IP, EIP and coefficients

Having identified the IBM tags of the nodes, for the IIBM technique used here each Ghost Node must now have a corresponding set of points as also shown on Figure 2.3 and explained in detail previously:

- Body Intercept (BI) - the algorithms search for it as the point on the user geometry that is closest to the Ghost Node
- normal unit vector - unit vector along the line between GN and BI.
- Image Point (IP) - produced by extending a fixed distance δ along the normal unit vector, from the BI towards the fluid domain.
- Extra Image Point (EIP) - produced by extending a distance of δ along the normal unit vector, from IP. This point only exists in Improved IBM technique [30] and is used primarily in ensuring the gradients imposition is more stable and accurate along the IBM line.

There were some caveats found when computing the above coefficients. If the grid is too coarse or there are concave areas this might produce unphysical results or algorithm fail entirely for Boundary Intercept. The simplest solution was to refine the background grid such that the curvature radius was relatively high per Ghost Node and no concave areas are present - a concave area is a series of smaller convex areas.

Another observation here was that any number of shapes can be used, each consisting of any arbitrary amount of points. The technique is entirely local and detects each GN and handle them on node-by-node basis without assumptions on the unstructured shape or their quantity. This aspect was one of the rare surprises where practical reality did not pose any unexpected challenges at all.

Figure 6.4 shows an example of the computations results on part of the cylinder where all edge cases with varying δ are visible on the image. As a reminder, the δ in the present context is the distance between Image Point and Extra Image point and is a local indicator of how likely the image points are to form a ’complete’ interpolation (where all 4 nodes are in the fluid domain). The way it’s used is to move the IP and EIP slightly further away from the boundary if δ is below a prescribed threshold.

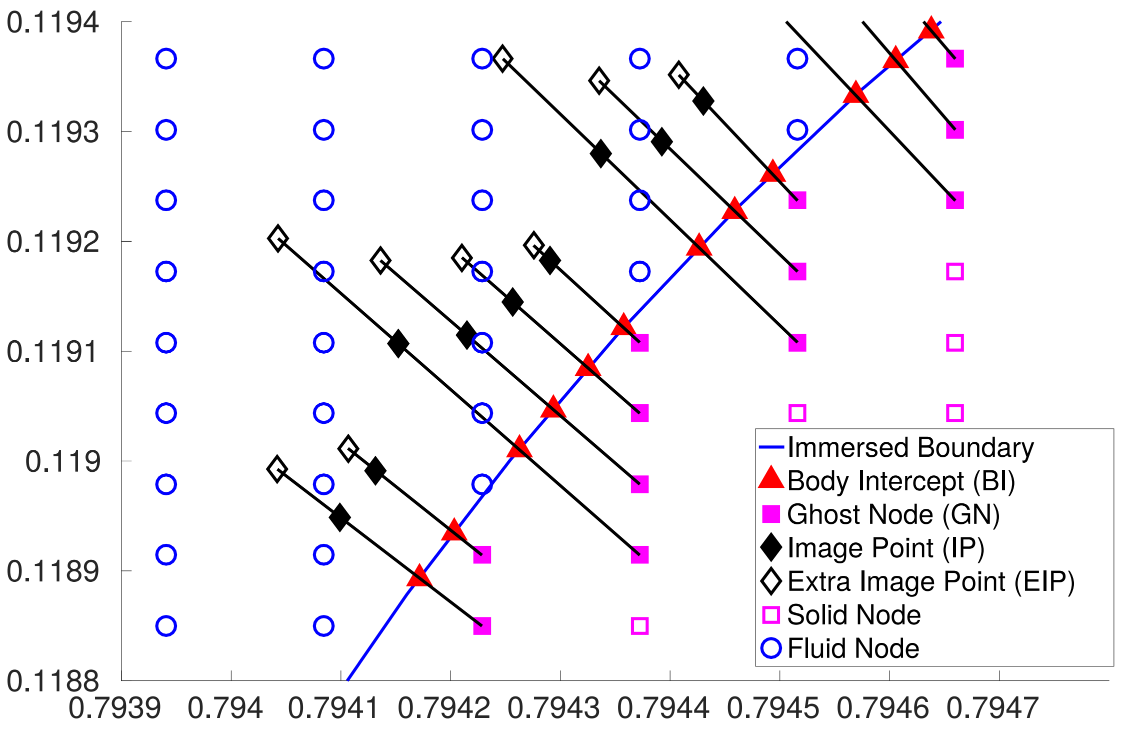

Figure 6.4: Actual plot of the normals computation.

6.2.4 Wall distance - Poisson equation

As explained in sections 2.6.1 and 3.2.7 the method of Tucker et.al. [4] was used to compute distance to the nearest user-defined wall at each fluid node of the background grid. This relatively simple Poisson equation allowed much easier debugging than more elaborate schemes. The following two dimensional geometries were used to validate the algorithm:

- turbulent channel
- cylinder
- backward facing step

Several other cases were also prepared but not simulated on with the N-S solver due to the stability issues uncovered with the code.

A brief scalability and memory use investigation then will be carried out to understand what can be expected from the present solver and to allow planning for hardware for when the full Navier-Stokes solver is completed.

The first test of the code, although without the IBM, was performed on Poiseuille flow [200, 201]. Next, using the IBM the calculations with the Navier-Stokes solver were performed on backward facing step [202, 203], cylinder at canonical Re=3900 [204] and NACA0012 [205, 206, 207].

6.2.5 Cylinder

Figure 6.5 shows the cylinder grid with mean y^+ of approximately 4 based on ghost node distance to nearest fluid node. Arguably a better metric would be y^+ based on distance of fluid node to user geometry and it will be implemented later however in the initial stages of development this option was judged to be a good first approximation given the cells sizes do not change drastically around the geometry.

Figure 6.6 shows implicit convergence history. Explicit pseudo-iteration history is not shown as it would be a flat line and the solution is not converged with explicit anyway. 64,000 iterations took 853.502 seconds (approximately 75 iterations per second) with implicit time integration and CFL of 40,000. Such high CFL is acceptable for Poisson equation but also because of high aspect ration and flow parallel to the wall. Explicit solution was not fully converged for practical reasons as it was not possible to obtain a stable solution with CFL higher than 0.2 with the present solver. There is no convergence acceleration such as multigrid or residual smoothing currently implemented in the code and physical time requirement to converge a solution using explicit time integration would be impractical. 4,270,000 explicit iterations were performed, taking 61,257 seconds (around 69 iterations per second) with CFL of 0.2 and to the author surprise appear to be the same or slower as the implicit solver. All CFLs were tried first and set to the highest values that could produce a stable non-diverging solution.

Analysing the output of the program, contour of the obtained solution is shown on Figure 6.7. The contours are somewhat skewed to the right as the inlet (on the left) is further away than outlet and the image is cropped. Figure 6.8 shows comparison of the result against an exact geometric node-by-node measurement, taken from the rightmost point on the middle of the cylinder, moving downstream. Until approximately 20% of cylinder diameter the solution is relatively accurate and then begins to diverge notably. This is a known limitation of the present method and the Hamilton-Jacobi approach is reported to be significantly more accurate as well as more computationally efficient. However, implementation of Hamilton-Jacobi method is much more complex while accuracy of the distance to nearest wall in turbulence equations only matters in the

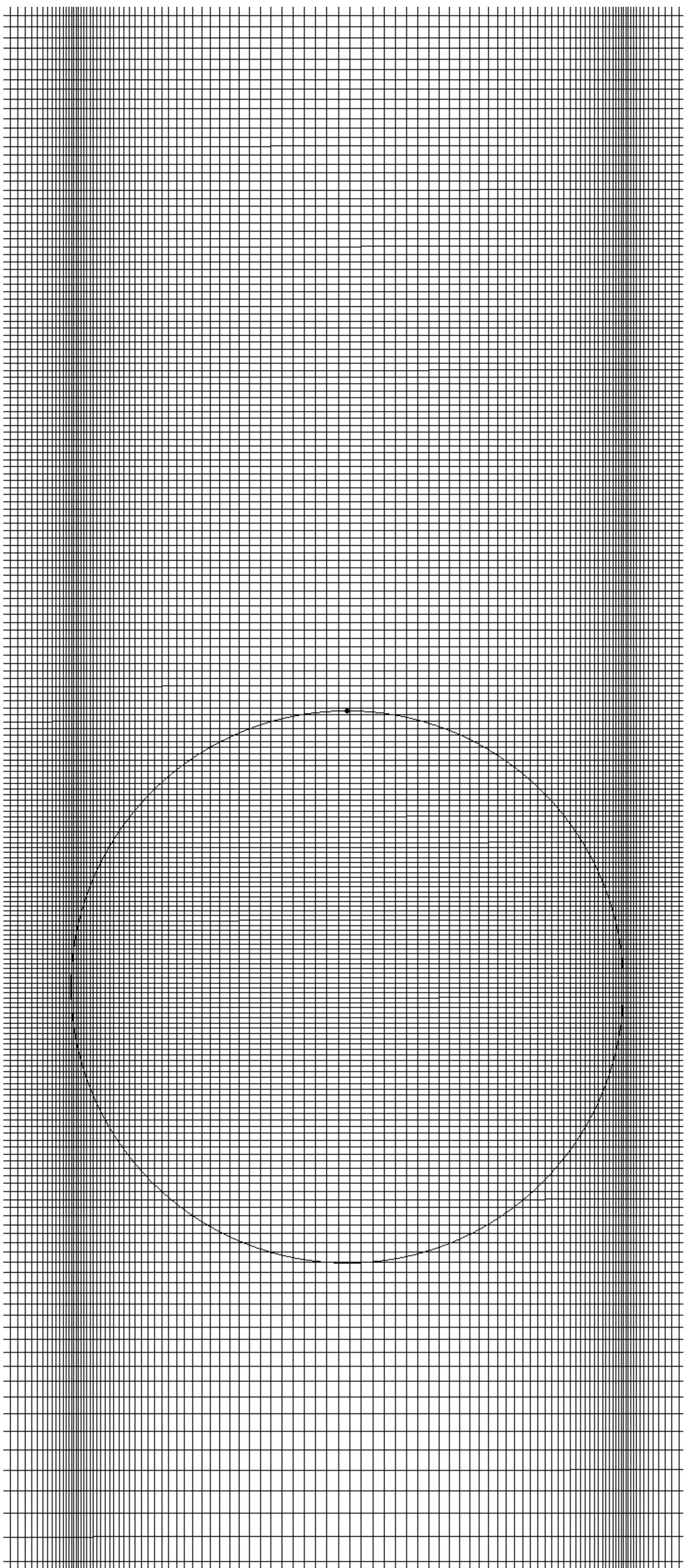

Figure 6.5: Magnification of the IBM grid of cylinder, both structured and unstructured.

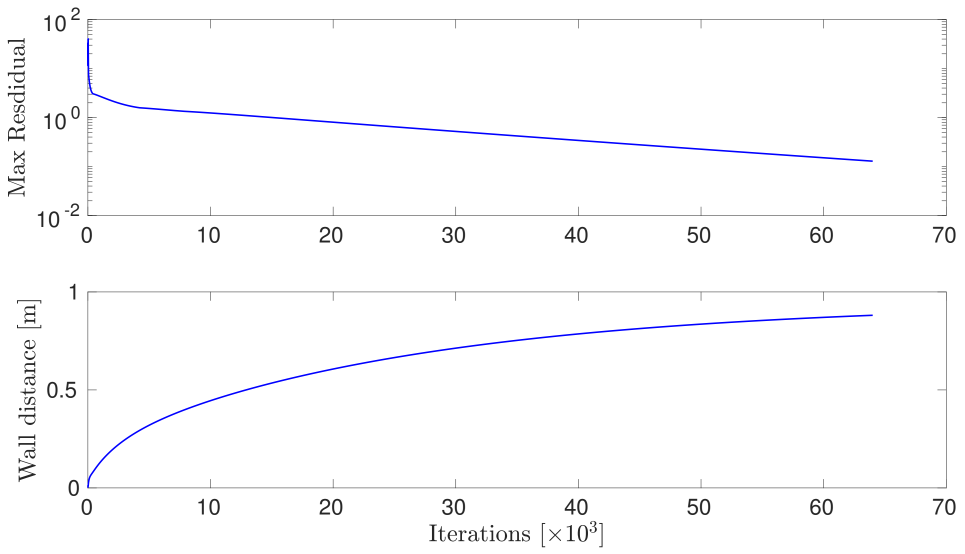

Figure 6.6: Convergence of the implicit diagonal scheme

immediate vicinity of the wallwhere y^+ is low. Away from the wall terms that use wall distance are insignificant and the error has less impact. For these reasons the present method of computing distance to the nearest wall was judged to be sufficiently accurate to move on with further development of this experimental solver.

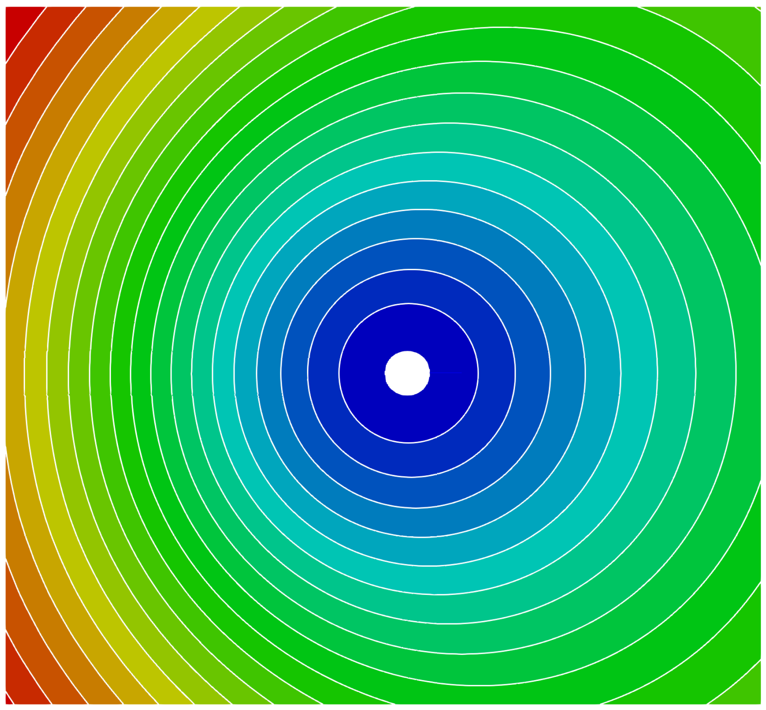

Figure 6.7: Contours of distance to the wall of 2D cylinder. Slightly skewed to the right as inlet (on the left) is further away than outlet and the image is cropped.

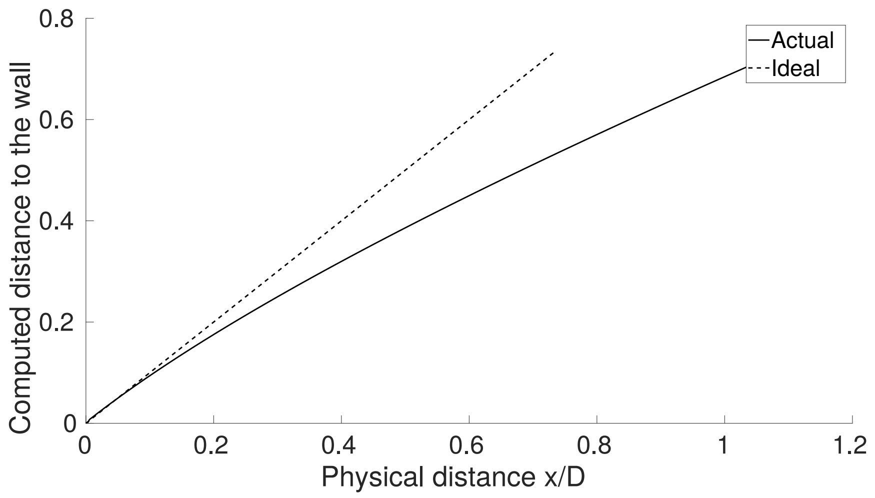

Figure 6.8: Validation of distance to the nearest wall as computed by the differential equation method of Tucker et.al. [4].

6.2.6 Backward facing step

Figures 6.10 and 6.11 show the computational grids used for the backward facing step simulation. Note two grids are shown on Figure 6.11; the 'background' cartesian grid on which computations are actually performed and the dotted 'user-defined' geometry which is used to impose wall boundary conditions for IBM purpose. The IBM grid is extended beyond the computational grid intentionally as it helped make the IBM algorithms more stable. Specifically the search algorithm struggled if every node (including wall nodes) wasn't clearly encapsulated by IBM grid. There is no risk to Navier Stokes computations using the extended grid as the IBM always computes normal to the user defined geometry in this case resulting in stencil along the bottom walls.

Figure 6.9 shows contours of the Poisson wall distance along the step with the values computed as geometrically expected.

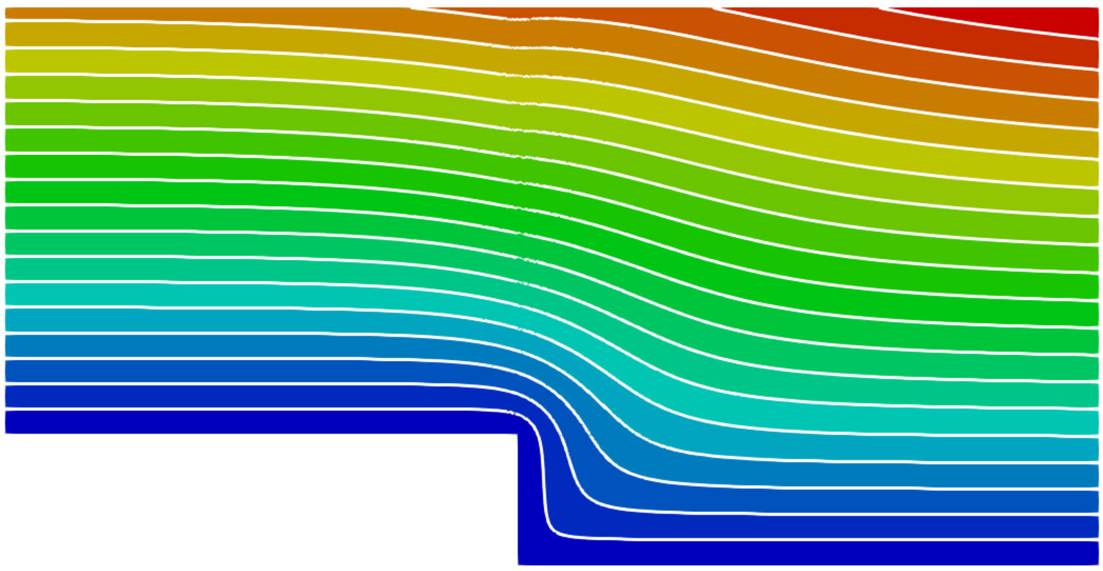

Figure 6.9: Contours of distance to the wall of backward facing step.

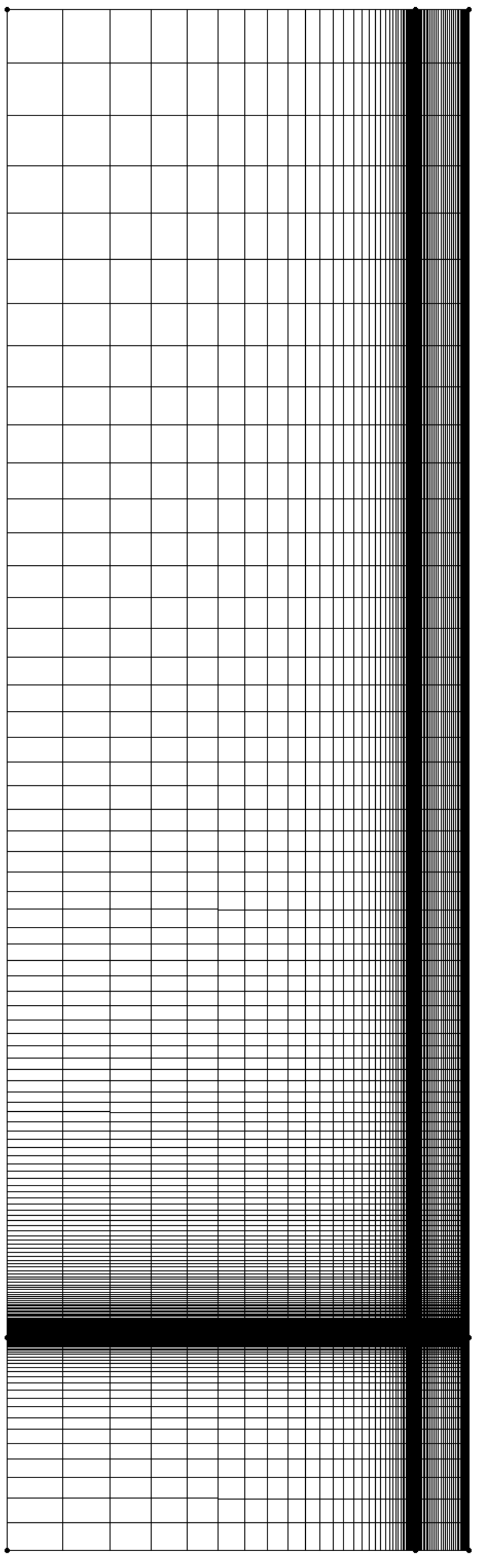

Figure 6.10: Overall view of the backward facing step grid used in the present study.

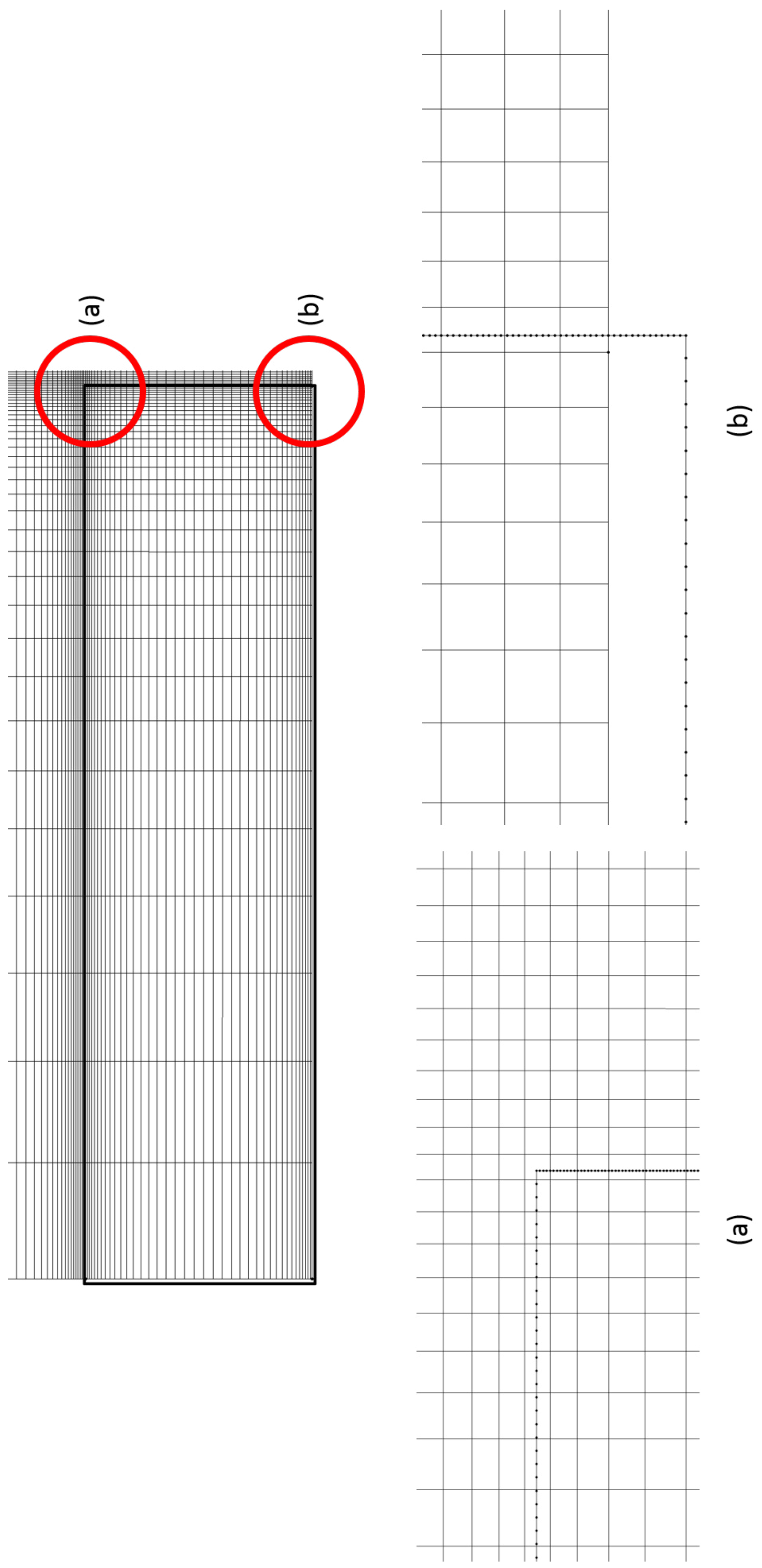

Figure 6.11: Magnification of the backward facing step grid with the IBM geometry superimposed.

6.2.7 Array of cylinders

Figure 6.12 shows the computational as well as IBM grid used for wall distance calculation validation of the array of cylinders. Figure 6.13 shows magnified representation of the pins. Several things are worth noting here. Grid around the full pin is very refined and combined with all the pins around it, the heavy refinement must be present in the entire domain. This is a limitation of the IBM in the present work. Secondly, the IBM geometry of the half cylinder again extends beyond the cartesian computational domain, similarly to the backward facing step. This again ensures the geometry ends well outside of any tolerances or the δ factor and all nodes are properly designated as solid. Also it is worth noting that the boundary conditions imposed by the IBM always are applied on top, and after the boundary conditions applied at the cartesian grid. So in this case the outer edges of the domain are walls but IBM designates them as solid and the flow is not computed there at all. While inconsequential in this test case, such approach is important for a range of edge cases that do occasionally occur elsewhere.

Finally, Figure 6.14 shows results of the Poisson equation computations with distances to the nearest walls as expected. As shown already on Figure 6.8 the solution here tends to be very close to geometrically ideal at close distances to the wall. This domain is fully encapsulated by walls and contains a number of wall boundary conditions inside designating the cylinders and the errors are minimised.

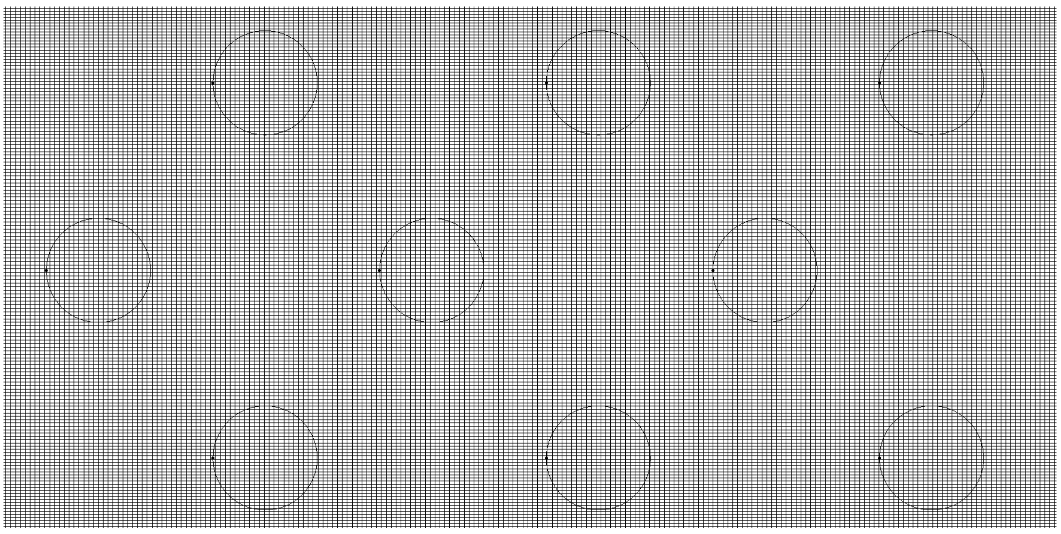
Figure 6.12: Overall view of the the array of pins grid.

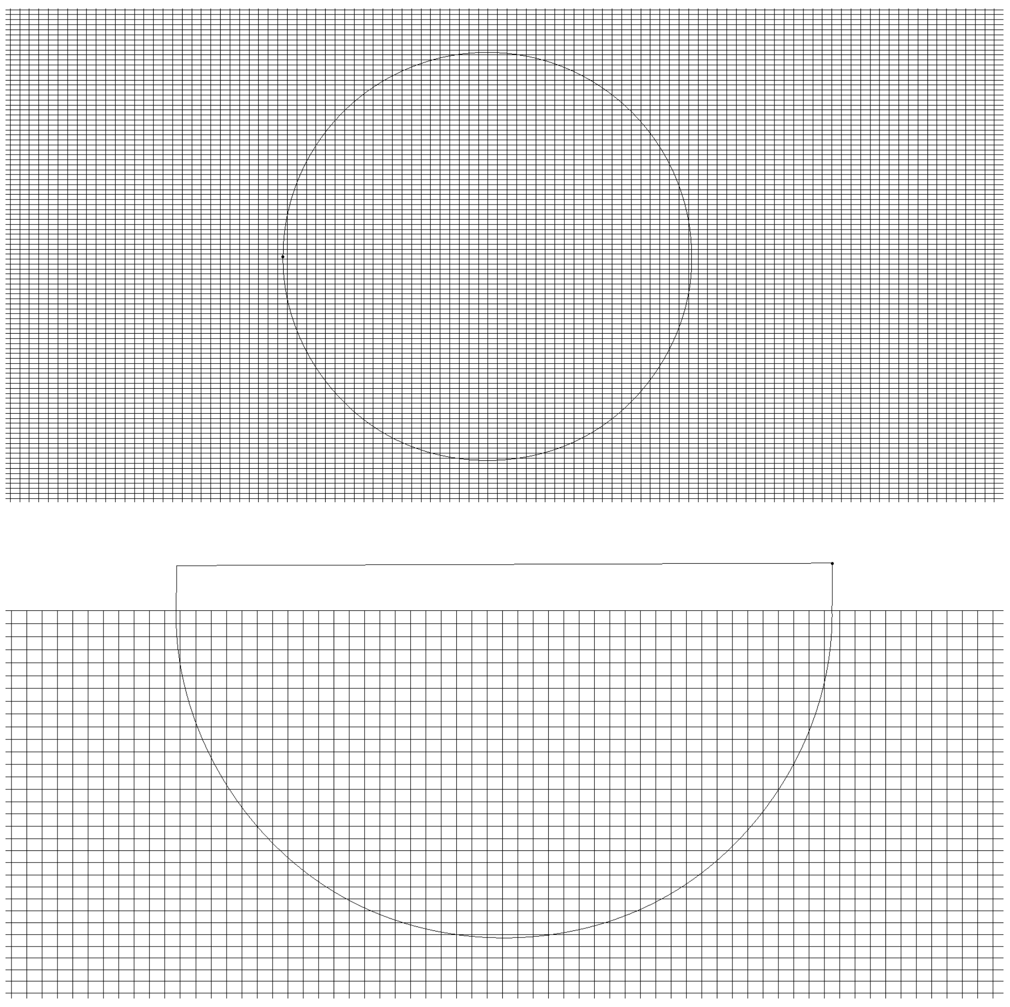

Figure 6.13: Pins representation in the array of pins case. Top figure is a full pin, bottom image demonstrates the way half circumference was imposed via IBM at the edges of the domain.

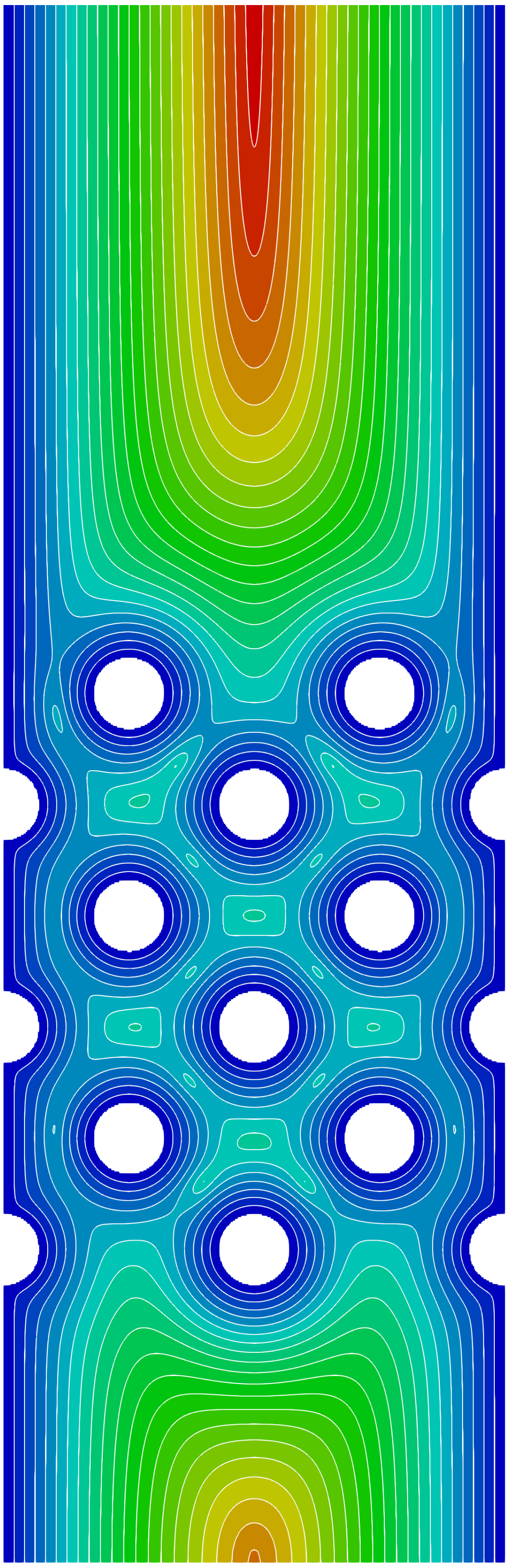

Figure 6.14: Contour plot of the distance to the nearest wall variable for the array of pins geometry.

6.2.8 T106 cascade

Figure 6.15 shows the two dimensional grids (computational and IBM) of the T106A test case. It was necessary to only display every second grid point for the figure to be legible. Note very high refinement near the suction side. This is not strictly necessary for the wall distance computation but it was necessary for any attempts at Navier-Stokes simulation of the geometry and correct capture of the separation point. There are no N-S computations shown here for this geometry as there was not enough time to complete the analysis properly however after few attempts it quickly became clear IBM needs a very high level of grid refinement to capture separation correctly at high Re.

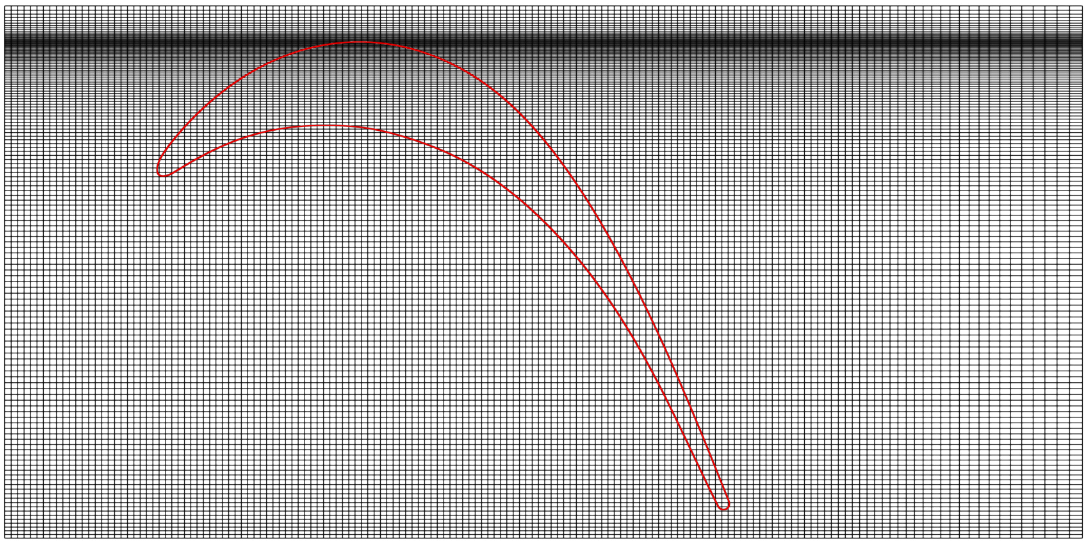

Figure 6.15: Grid of the T106A 2D turbine cascade representation. Every second point is shown.

And finally, Figure 6.16 shows results of the computatons which appear to be as expected. Note the grid oscillations at the periodic boundary. This appears to be only a visual artefact due to the way the results are displayed and no IBM computations are taking place in those areas. However only the simplest translational periodicity was applied there without more advanced filters or checks and this area should be watched closely in any N-S computations.

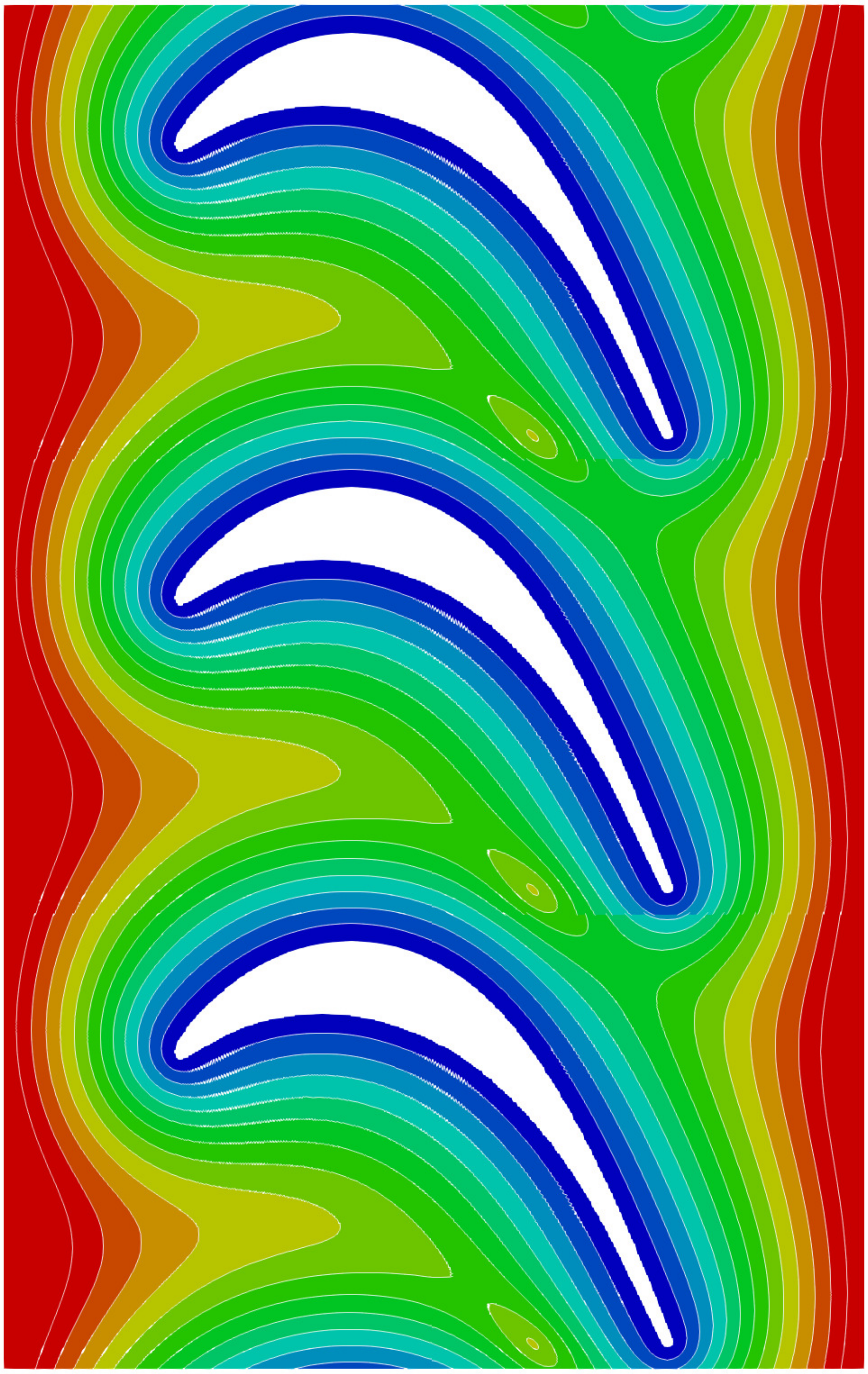

Figure 6.16: Contours of the distance to the nearest wall variable for the T106A 2D cascade.

6.2.9 Parallelisation study

Memory usage was 4.8 GB per million nodes and changing linearly. This value can be greatly optimised by compressing the variables together and removing several arrays. Significant amount of IBM data was also held on per-core basis for debugging. Due to the way the code was initially written with OPS library there is no distinction of memory used for NS or Poisson computations as all memory is always allocated. This was mainly due to the way IBM computations were initially implemented but was improved in later versions of the code.

Figure 6.17 shows a Poisson computation on a grid with 600,000 nodes, ran for 100,000 iterations. Minerva HPC used Intel Xeon E5 Sandy Bridge 2.6GHz CPUs, arranged in 16 core nodes. Tests shown an NVIDIA 1050Ti performed comparably to 56 CPU cores (7 physical 8 core CPUs). This is where cost effectiveness can be demonstrated as at the time of performing the simulations the NVIDIA GPU was about the price of one of the CPU units. Power consumption will also be much lower with GPUs; NVIDIA 1050ti is rated at 75W while the Intel CPU is 130W (910W total for 7 CPUs). This is acceptable for small scale systems but will become a significant benefit as simulations are scaled up to thousands or millions of cores.

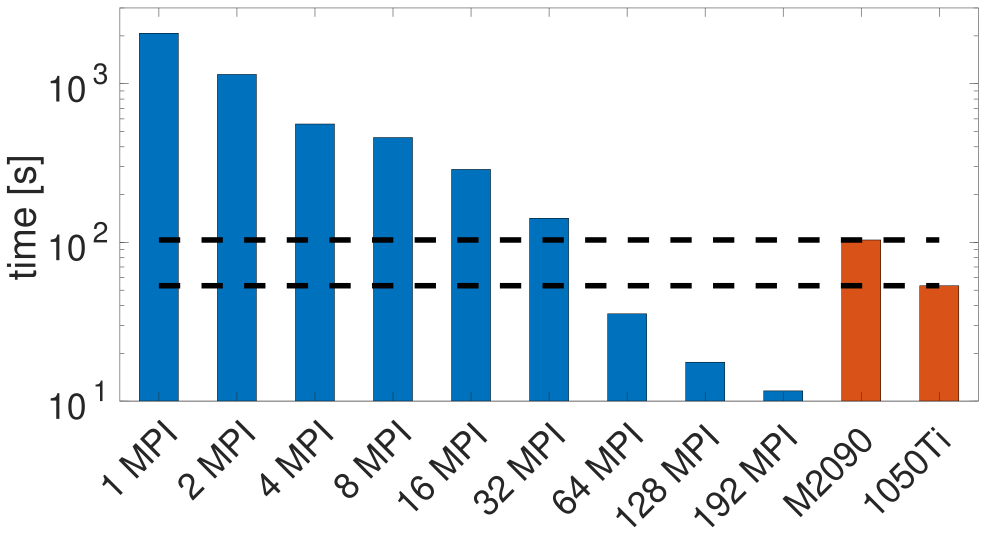

Figure 6.17: Comparison of runtimes on CPU and GPU. NVIDIA 1050Ti performance is equivalent to 56 MPI ranks on Nottingham HPC Minerva. Tests done in 2016 when the 1050Ti was the latest GPU available. Time in wall clock seconds.

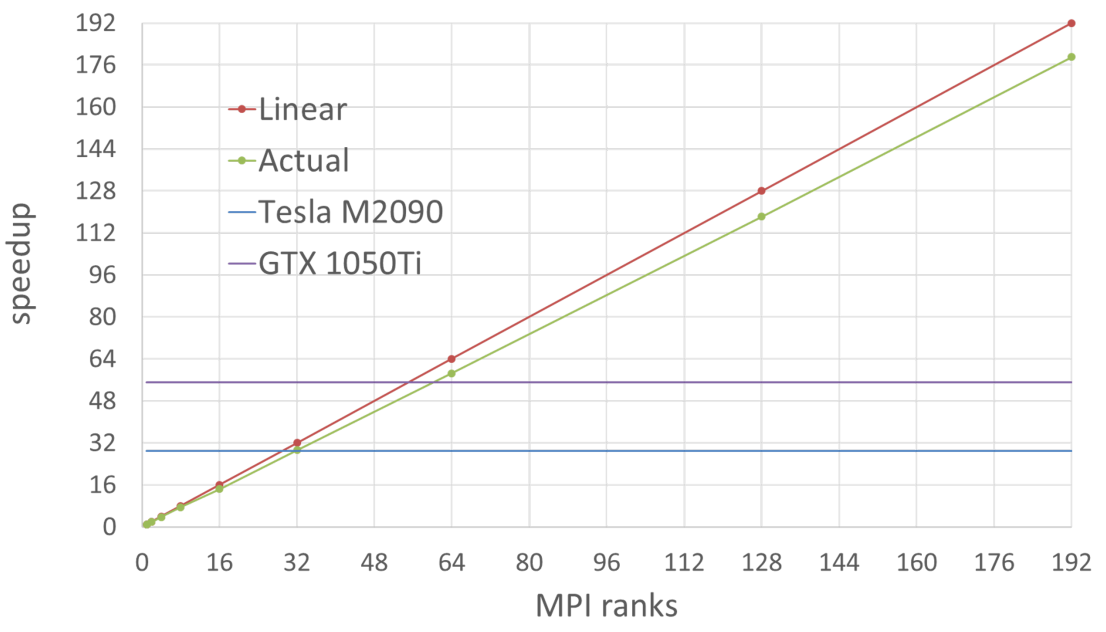

Figure 6.18: CPU scaling study. Scaling remains quite efficient despite two dimensional geometry being used and less than 10,000 nodes per MPI rank for the 192 ranks simulation.

Figure 6.18 shows ’strong’ scaling capability of the present approach. Even on a small two dimensional grid the scaling remained mostly linear up to 192 cores (and 192 MPI ranks). Note this required no involvement of the author in the low level MPI implementation inside the OPS library.

6.2.10 Conclusions of the base algorithms validation

It appears that the implementation of the IIBM using OPS and ADI has so far been successful and produces working set of IBM coefficients that lead to expected result of a simple differential equation on a grid using various hardware. Some instability is observed at the periodic boundary as can be see on Figure 6.16; this is unlikely to be coming from the IBM coefficients but will need to be investigated before running more simulations with periodic boundary conditions.

More detailed verification is still needed however top level validation is sufficient to begin with due to time constraints. It is also still unknown whether the present combination of computational techniques is even viable and detailed V&V study will be conducted once the 3D Navier-Stokes solver is at least operational, however inefficiently that may be. The reason is that three dimensional IBM methods are much less mature or stable than two dimensional work and also implementation is likely going to be significantly more complex to adapt to the ADI algorithm and OPS framework.

6.3 2D N-S solver

Now that the necessary IBM coefficients have been validated, a solution of two dimensional RANS equations can be attempted.

6.3.1 Turbulent channel Re=13,800

The RANS equation validation begins with a very common channel test case as shown on Figures 6.19 and 6.20 to demonstrate the two dimensional parallelised code. Although this particular geometry does not have to be simulated via IBM due to it being strictly rectangular, IBM boundary condition imposition will be used anyway to demonstrate that it results in correct boundary layer. Geometry was imposed in a manner similar show on Figure 6.11 previously.
The boundary conditions and geometry setup are that of Schlater et.al. [208] and AGARD group [209]. The case is also a standard NASA CFD validation exercise [210] although with Re of 80 million for validation of the SA model. There also exist simulations at lower Re_θ and DNS performed by Moser et.al. [211] and Home et.al. [212].
While exact analytical solution does not exist for this exercise a well known 'Law of the Wall' discovered by Von Karman et.al. [213] can be used to assess accuracy of the near wall region outside of the viscous sublayer. The equations behind the Law of the Wall as plotted on Figure 6.21 are as follows:

$$y^+ = \frac{u^* y}{\nu} \tag{224}$$

$$u^* = \sqrt{\frac{\tau_w}{\rho}} \tag{225}$$

$$\tau_w = \mu \left(\frac{\partial u}{\partial y}\right)_{y=0} \approx \mu \frac{\Delta u}{\Delta y} \tag{226}$$

With u^+ defined as:

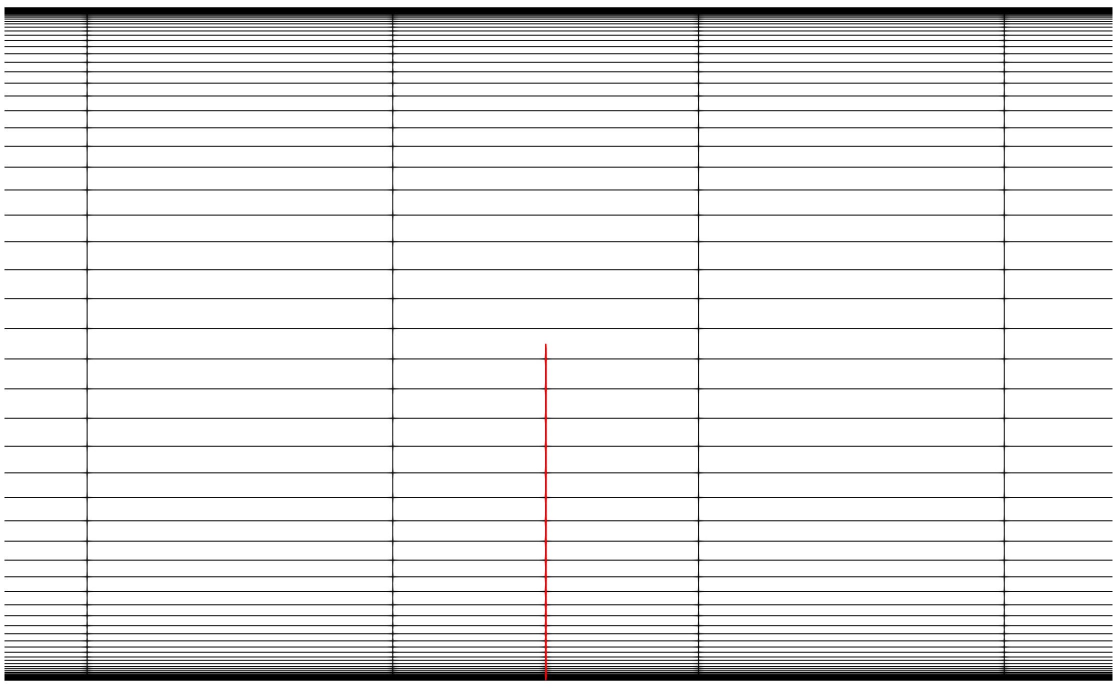

Figure 6.19: Grid of the turbulent channel geometry with data extraction location marked with initial y^+ of 0.5

$$u^+ = \frac{1}{\kappa} ln(y^+) + C^+ \tag{227}$$

$$\kappa = 0.41 \tag{228}$$

$$C^+ = 5.0 \tag{229}$$

Empirical measurements on Figure 6.21 were done by Hussain et.al. [214]. The paper also goes into more detail regarding the analytical solution.

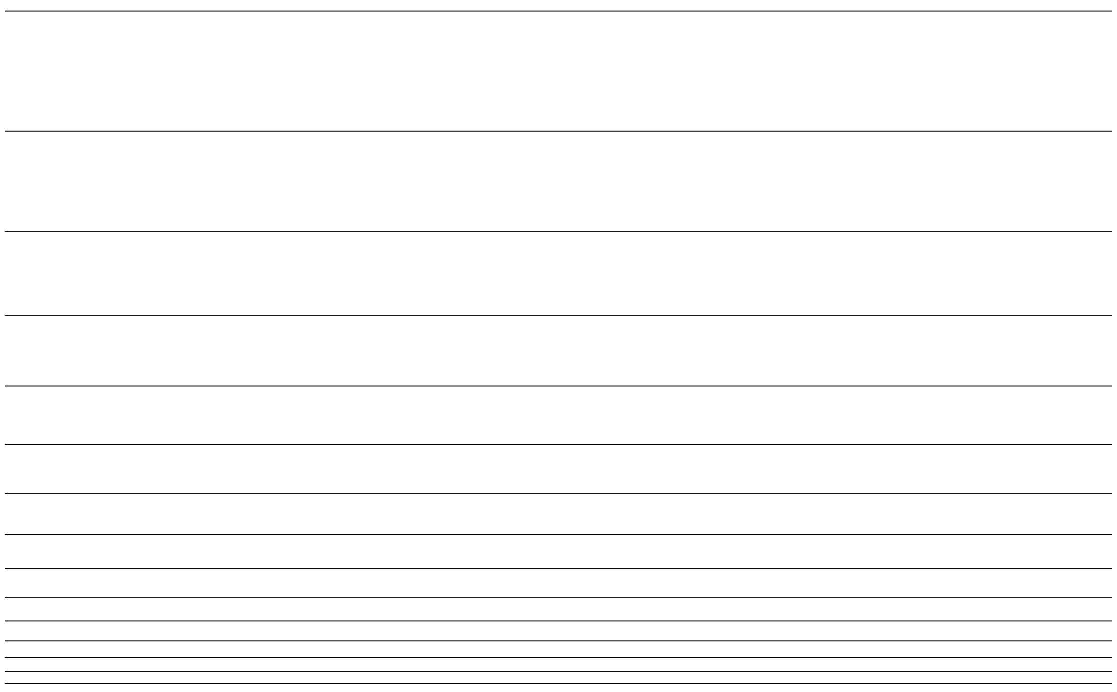

Figure 6.20: Near wall magnified view of the turbulent channel geometry grid.

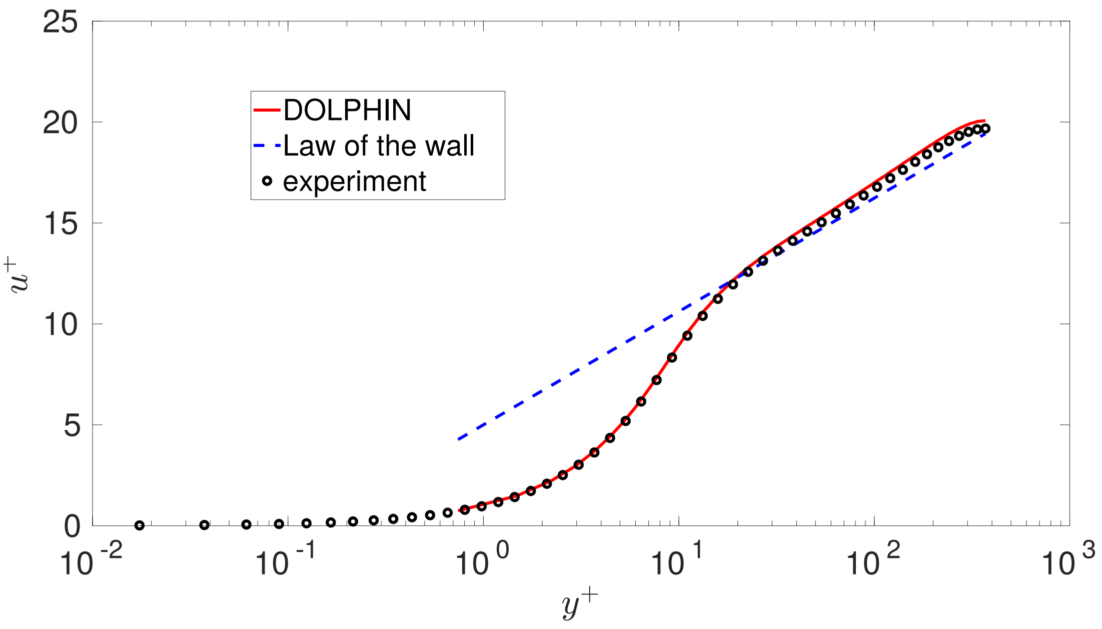

Figure 6.21: Validation of the turbulent channel simulation with experimental data and Law of the Wall. DOLPHIN is University of Nottingham's in-house structured CFD tool developed by Dr. Richard Jefferson-Loveday which inspired the present work.

6.3.2 Cylinder Re=3,900

Next validation case will be a cylinder with grid and geometry as shown on Figure 6.5. The geometry at this particular Reynolds number is commonly used for validation and many authors perform experiments as well as high fidelity simulations such as LES or DNS on the cylinder. Some examples include Fornberg et.al. [215], Taneda et.al. [216], Balabani et.al. [217], Kravchenko et.al. [204] and Parnaudeau et.al. [5]. There are also extremes of Re, both very low Re studies reviewed by Preece [55] and very high Re ($> 10^7$) by Roshko [218]. High order techniques were also investigated on the flow around cylinder [219]. The actual boundary conditions used here are summarised in Table 6.1.

Parameter	Value
Re	3,900
Fluid used in exp.	air, 20 degC
$\overline{u_{bulk}}$	9.81 m/s
I_{inlet}	1% - 5%
$L_{characteristic} = d_{cylinder}$	6mm = 0.006m

Table 6.1: Summary of the conditions of the turbulent flow over cylinder case

Moving on to results of the simulations, data extraction locations is again relatively common in literature and most researchers are using data on lines as shown on Figure 6.22. Figures 6.23 and 6.24 show velocity comparisons on the lines directly across the wake while Figures 6.26 and 6.27 show velocity along the wake and pressure coefficient around the cylinder circumference respectively.

RANS models are generally known to struggle simulating flow around this geometry accurately and it was a challenge to even obtain a stable solution with the present code at all. It is clear from the plots though that even under this conditions the Navier Stokes equations appear to be converging at an approximately correct solution.

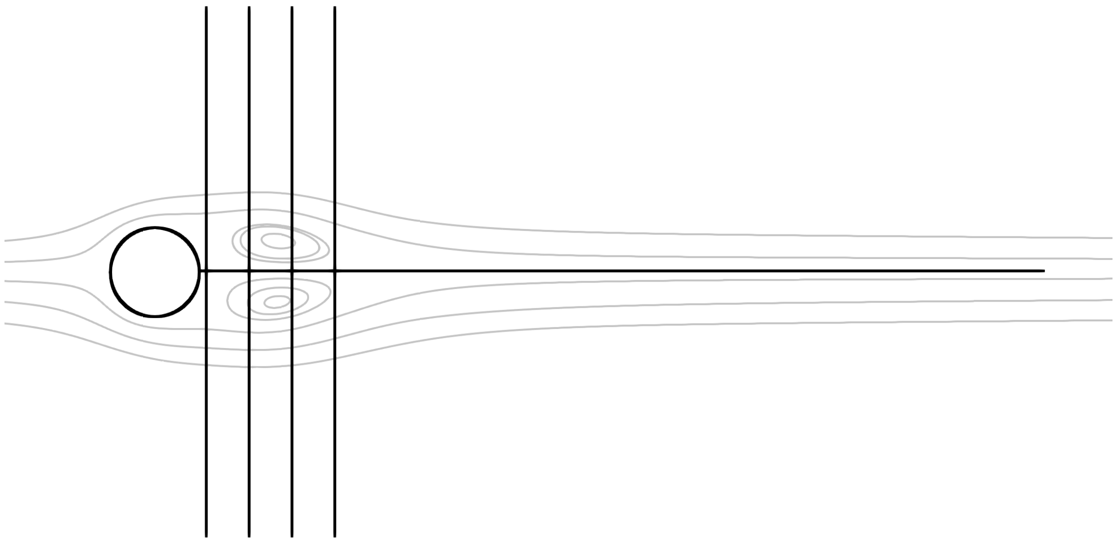

Figure 6.22: In solid black are data extraction lines. The same locations and lengths as experimental data of Parnaudeau et.al. [5]. Streamwise direction is positive ”X”.

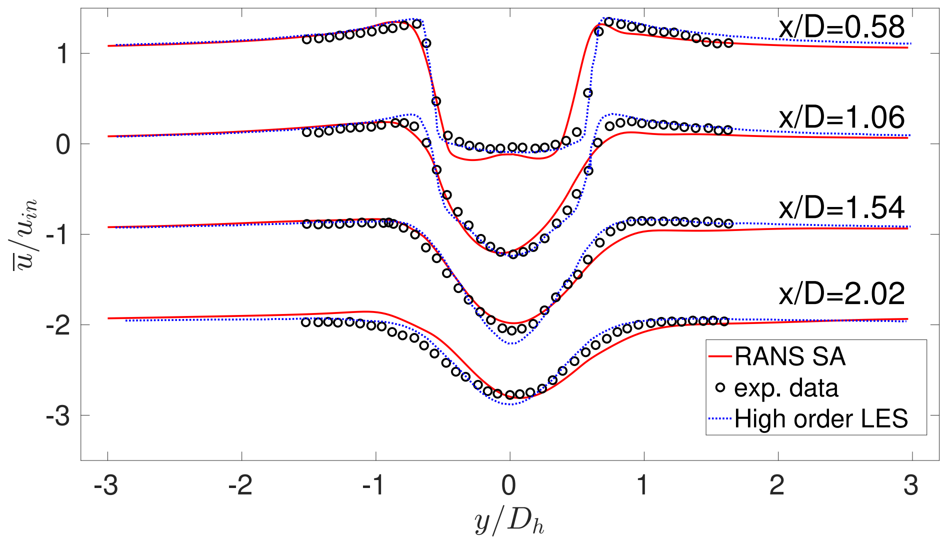

Figure 6.23: Cylinder x-component of velocity comparison with experimental data.

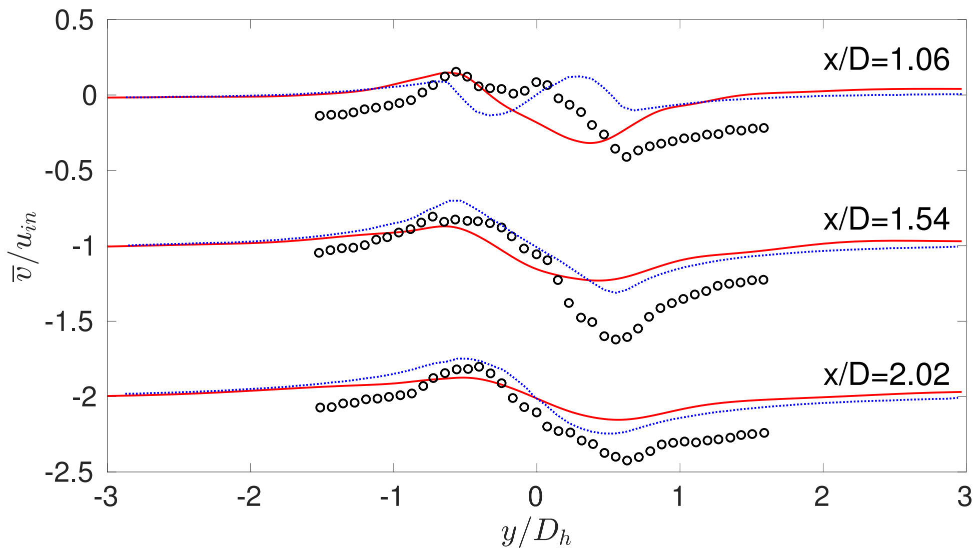

Figure 6.24: Cylinder y-component of velocity comparison with experimental data. Legend as in 6.23

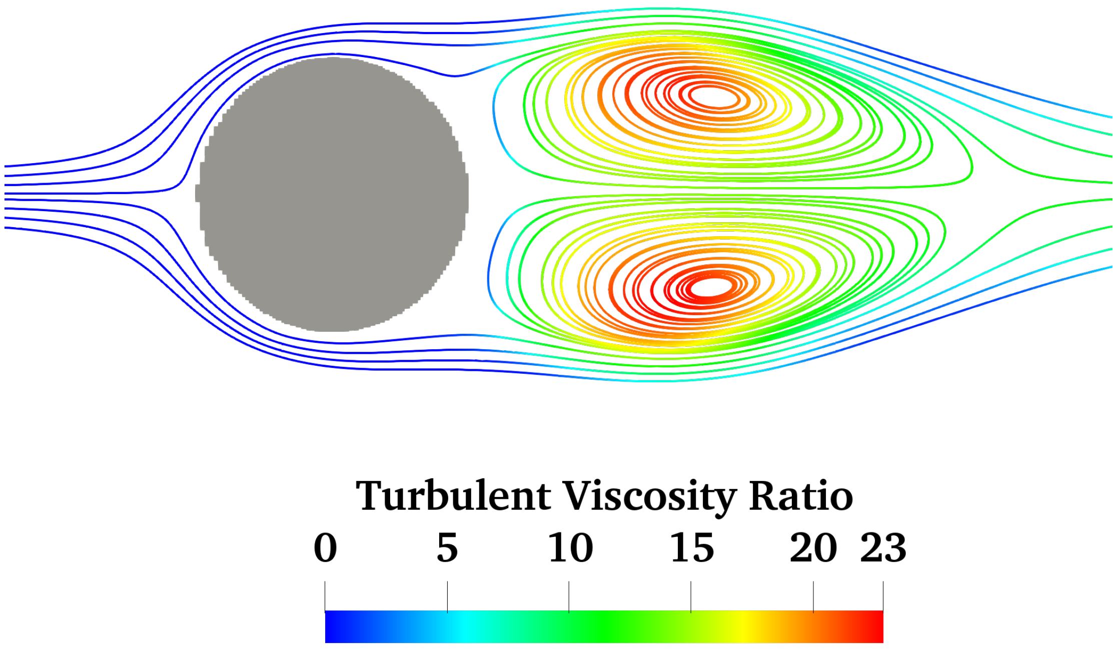

Figure 6.25: Cylinder streamlines with viscosity ratio colours.

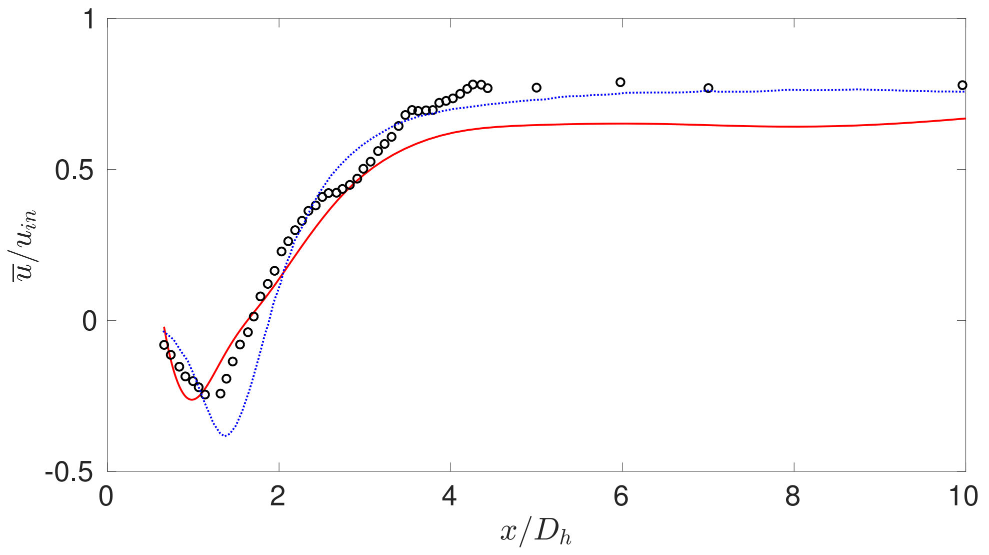

Figure 6.26: Cylinder velocity profile along the wake centerline directly behind and away from the geometry. Legend as in 6.23

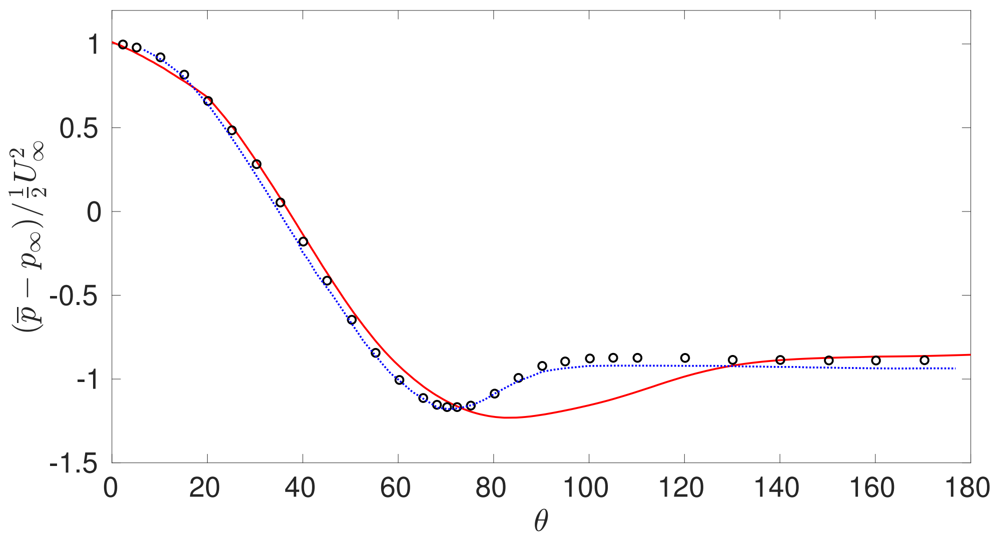

Figure 6.27: Cylinder pressure coefficient around the geometry. Legend as in 6.23

6.3.3 Upstream facing step Re=36,000

The last two dimensional validation case is an upstream facing step as described by Jespersen et.al. in the NASA validation resources [220]. Case details and boundary conditions are the same as Jespersen et.al. [220] and summarised in table 6.2. Flow around this geometry is often simulated at many different Reynolds numbers, such as in paper by Le Moin et.al. [221] with step at Re=5,100 and expansion ratio of 1.2. A NASA report [222] also has this step at Re=5100 for validation of the SA turbulence model. Re=6000 and some more data is in paper by Schafer et.al. [223] but also in Lee et.al. [224] and Vogel et.al. [225]. Overall, similarly to the previous cylinder case, ample high fidelity simulation as well experimental data exist related to this geometry and flow conditions to aid validation of CFD codes.

Parameter	Value
Re_{ref}	36,000
Ma_{ref}	0.22
Fluid used in exp.	air, 20degC
$\overline{u_{bulk}}[m/s]$	75.5
I_{inlet}	$1\% - 5\%$
$L_{characteristic} = h_{step}[m]$	0.0070417

Table 6.2: Summary of the conditions of the upstream facing step case.

Data extraction lines again tend to be common across literature and for clarity are shown on Figure 6.28. As before, the figure is to scale and the length of lines is representative of the data in the four plots that follow.

Figures 6.29 and 6.30 show the streamwise (x) velocity component along the lines, with legend as on 6.23 while Figures 6.31 and 6.32 show streamlines of the recirculating region behind the step with the IBM imposed geometry marked with red dotted lines. Since the model used to obtain the results is RANS it is not expected to agree fully with experimental measurements or high fidelity simulations such as LES but it does appear to be consistent with RANS simulations performed by commercial codes or other researchers which is encouraging.

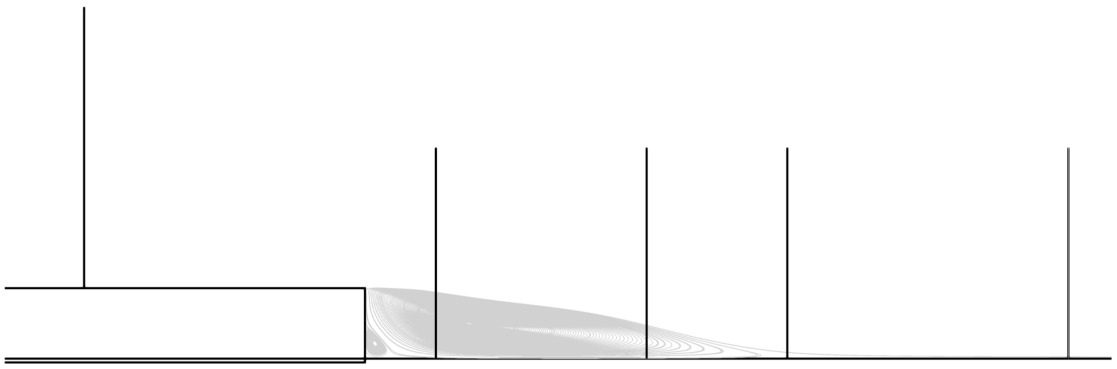

Figure 6.28: Data extraction lines for the upstream facing step. Figure to scale.

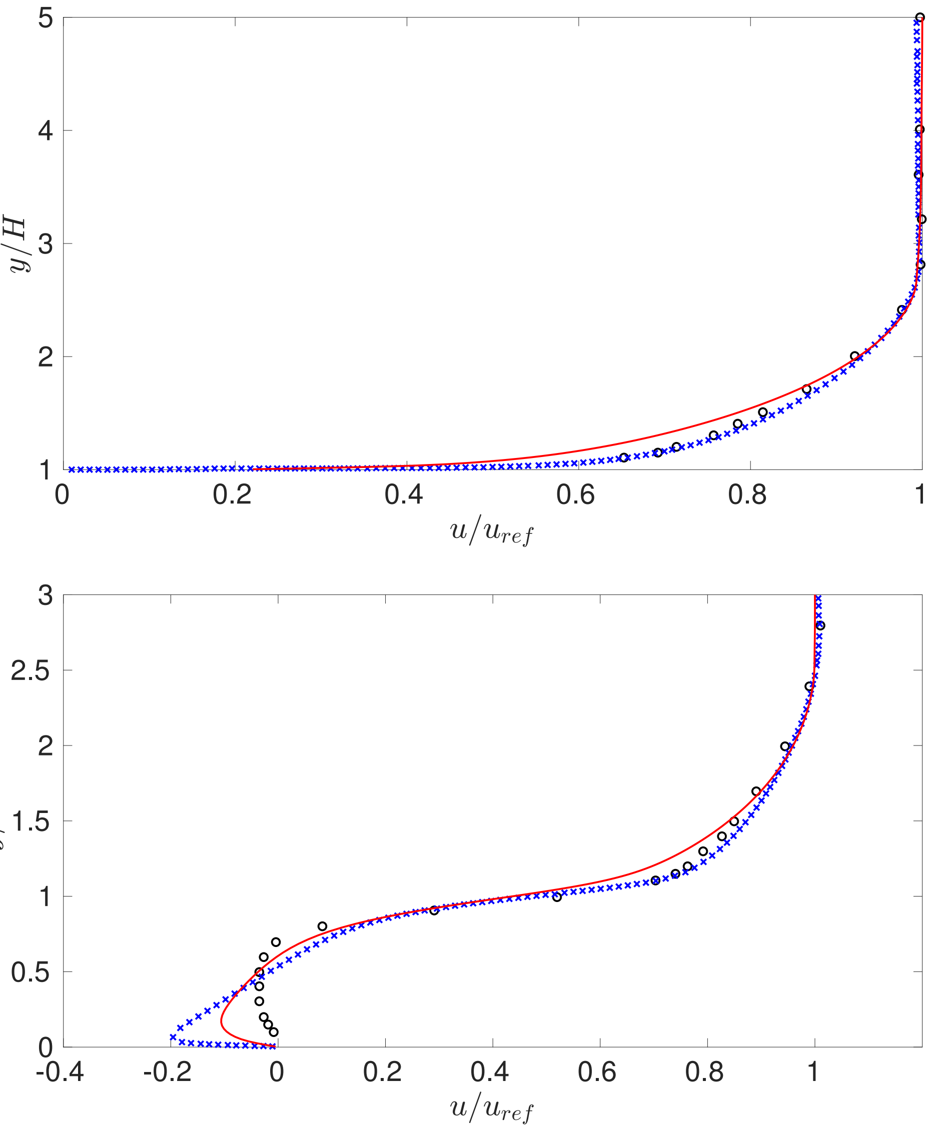

Figure 6.29: Streamwise velocity component on the first two data extraction lines.

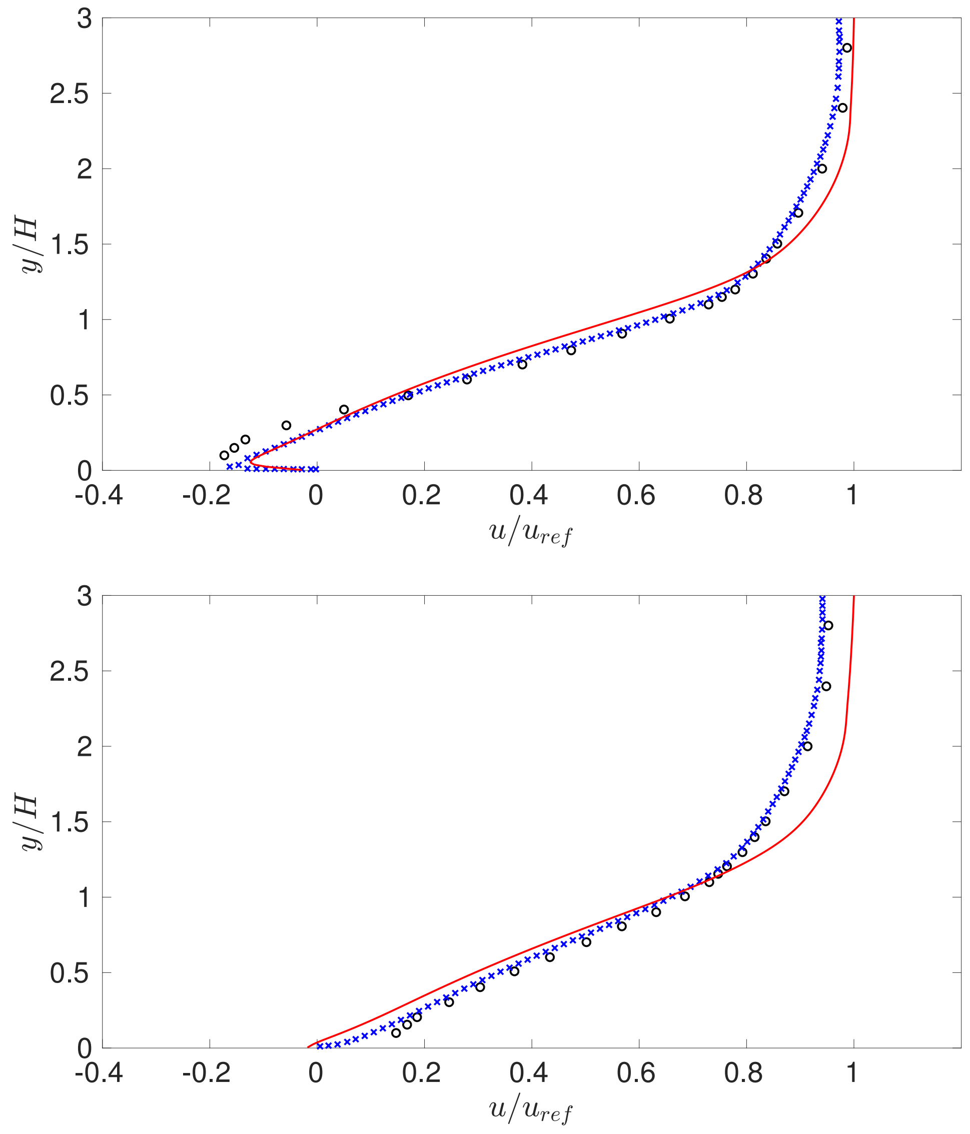

Figure 6.30: Streamwise velocity component on the next two data extraction lines.

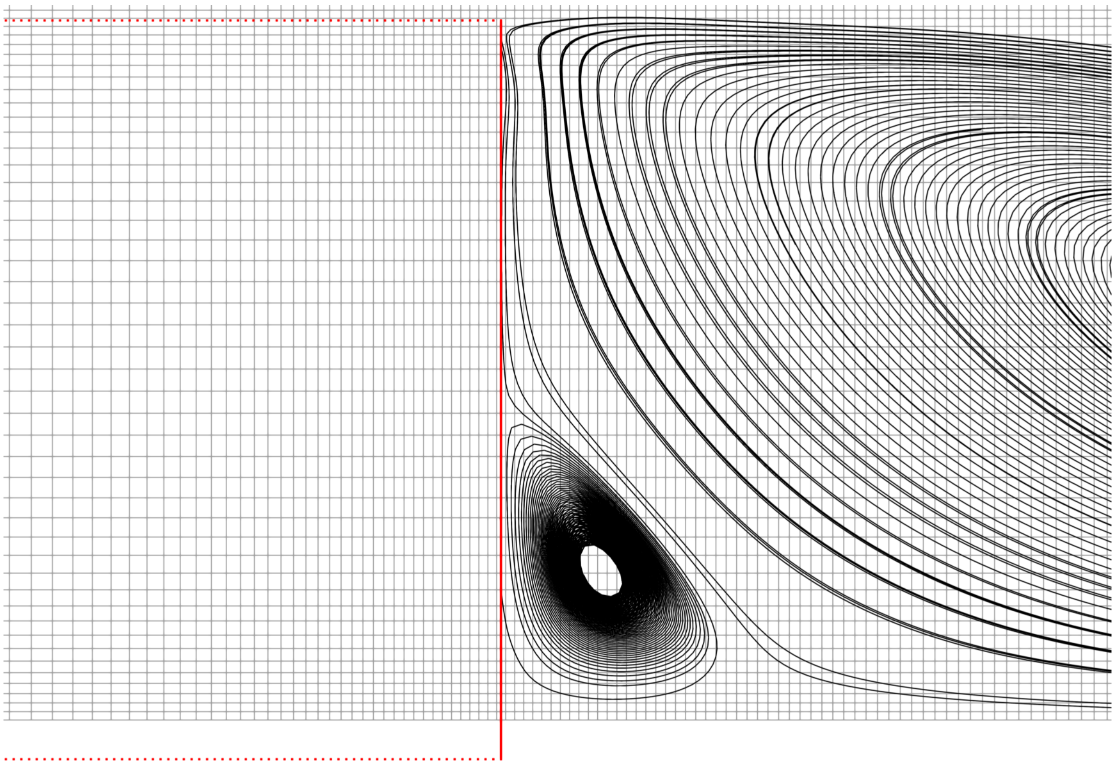

Figure 6.31: Streamlines immediately after the step with the IBM geometry marked.

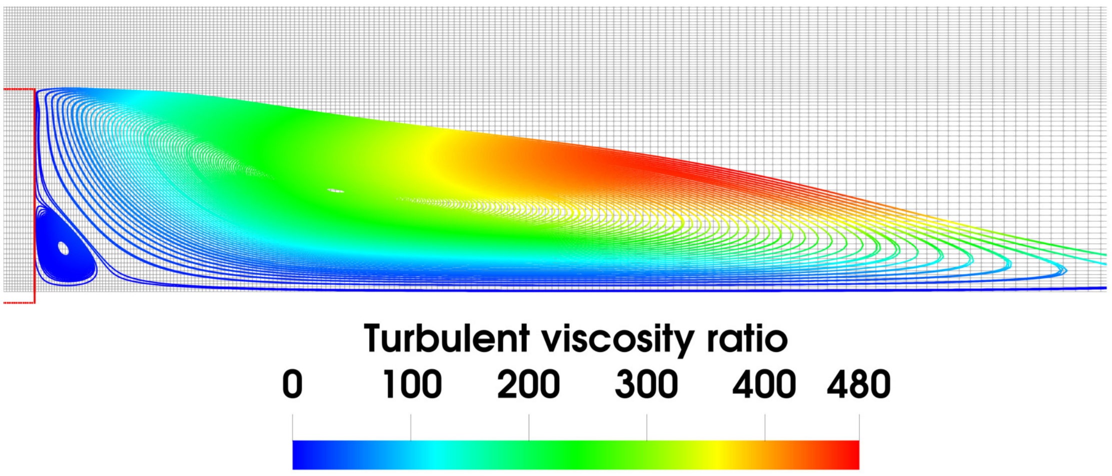

Figure 6.32: Overall view of the streamlines behind the step with turbulent viscosity ratio marked.

6.3.4 Further validation

Three further validation cases were identified that can be used to get more detailed insight into performance of the CFD program:

- Array of cylinders as summarised in Table 6.3
- Rounded rib as used by Gillespie et.al. [226] and Rallabandi et.al. [89]
- T106A 2D turbine cascade as summarised in Table 6.4 and much data including DNS provided by Kalitzin et.al. [227] [228] or Stieger et.al. [229].

Parameter	Value
Re	12,858
Fluid used in exp.	water, 20 degC
$\overline{u_{bulk}}$	0.93 m/s
I_{inlet}	1% - 5%
$L_{characteristic} = d_{cylinder}$	10mm = 0.01m
$\overline{u_{gap}}$	1.2858
amount of cases	3
paper	Balabani, Yianneskis et.al. [217]

Table 6.3: Summary of the conditions of the array of pins case

Re	$148,000$
Chord (characteristic length)	$0.0313\ m$
Inlet flow angle	$37.7°$
Mean flow velocity magnitude	$69.48\ m/s$
Mach number	0.2
Air dynamic viscosity	$1.8 \times 10^{-5}\ Pas$
Air inlet density	$1.225\ kg/m^3$

Table 6.4: Summary of the conditions of the T106A turbine cascade test case.

The data and geometries from the above papers will prove useful once the stability issues with the code are resolved.

6.4 Chapter conclusions

All the necessary equations, including the two dimensional Navier-Stokes equations were implemented. Immersed Boundary Method preprocessing algorithms were modified as necessary, tested and several optimisations performed to achieve a balance between speed, robustness and accuracy.

It was demonstrated that the bulk of the two dimensional algorithm works and has been implemented correctly. There appear to be some instabilities and the filtering scheme needs more work to achieve balance between artificial viscosity (i.e. accuracy) and stability. While full verification was not performed, the results from the test cases shown are encouraging to proceed to implementation of a three dimensional algorithm. Scalability has been shown to be good, even in a scenario with very low node count per rank. Due to this it is likely the scalability will remain good or better as the cell count gets larger.

A 3D algorithm is likely to be more challenging and complex to get right, however there do not appear to be any fundamental technical or practical obstacles as to why it cannot be done. Overall the author believes it was demonstrated that this type of CFD solver is feasible and can meet all the requirements set out earlier.

Chapter 7

7 Conclusions and future work

7.1 Summary

Several threads were explored in this thesis; turbulence modelling, numerical methods as well as computational approaches. Meshing and geometry manipulation was also considered, as is inevitable in any CFD work. As an engineering work, an element of business rationale was also explored and the overarching application was industrial turbine blade simulations with all its complexities such as internal and film cooling. The primary aim of this doctoral work was to advance understanding and simulation capabilities of the field and this goal was achieved in two ways.

One was a comprehensive study of advanced hybrid RANS-LES models, which concluded that LES is often the most optimal type of simulation, even if done on a grid that is coarser than the ideal LES requirement would dictate. DES and SST-SAS variants are still promising and should be explored on engine-representative geometries but for simpler shapes ’coarse grid LES’ appears to be acceptable. Progressively complex geometries were created and simulated on. Attempting to simulate flow around the most complex geometry proved that meshing is indeed a primary difficulty and fundamentally different approach to meshing is required.

The second way to achieve the thesis’ aims was a two dimensional, portable, Immersed Boundary Method program that eases significantly the grid generation woes and has the ability to harness the power of variety of hardware, current and future, efficiently. It was proved possible to use the IBM method with implicit time and spatial discretisation on a GPU and with high order methods. While three dimensional extension will likely be much more complex to develop and achieve stability, there appear to be no fundamental obstacles for making it happen.

7.2 Contribution to knowledge

- It was shown that GPUs can be effectively utilised to simulate flow around realistic geometries, outperforming classical CPU solvers. This is without compromise of numerical methods and the same implicit time and spatial discretisation that are used on CPUs can be used effectively with minor modifications. It was also demonstrated that GPUs offer cheaper computational power and higher power density which makes it possible to run large scale simulations on local machines

- It was shown that structured grids do not have to be inflexible in terms of what geometry can be modelled - in fact the IBM method proves that grid generation is easier than for classical solvers and produces higher quality grids without compromising accuracy even at high, turbulent Re.
- It was shown that high level libraries provide significant and much needed benefits to the scientific developer. Indeed it would not be possible to create the present code within the timescale given if no high level library was used and performance would likely be below that of a dedicated library. The code is also significantly easier to maintain and deemed to be 'future–proof', which is not the case with manual parallel implementation.
- The combination of methods demonstrated to perform well in this work allows easy implementation of high-order spatial discretisation to potentially arbitrary order of accuracy, further increasing simulation efficiency.
- A hybrid RANS–LES turbulence approach termed SAS–SST was thoroughly tested for use with turbomachinery internal cooling applications. It was shown that the artificial forcing remedies some of the original models problems and the model is a significant improvement of URANS with little increase in cost or complexity. However, DES or even coarse grid LES is likely still preferred for the present application.

7.3 Limitations of current work

- Larger simulations on GPU are memory limited
  - One must distribute the grid and solution data across multiple GPUs, which requires MPI + GPU; this is not yet available with OPS and ADI
  - The best GPUs currently have 32gb memory. One would need >20 very expensive GPUs to perform a large scale LES simulation (assuming 1b cell grid)
  - The upside here is that having so many GPUs would result in even 1b cell

calculations being done in design timescales, i.e. < 12h

- Grid must be very refined near the wall for IBM – main limitation for high Re so far Hence IBM has been applied to low Re so far mainly
- The datasets of OPS are relatively inflexible and significant amount of unnecessary information is stored. For instance to store IBM interpolation coefficients an ops_dat datatype is used which allocates memory for each cell but has only useful information near the unstructured boundary comprising of approximately 5 % of cells. This was found to be unavoidable in many places in the code. Similarly some pre-processing routines loop over each point and perform several checks for every node of the domain while it is often apparent when a check is not required. The impact of this is limited as pre-pre-processing is a one-off operation.

7.4 Recommendations for future work

The methods developed in the present work may be improved substantially by introducing some sort of appropriate local grid refinement. At the moment this is major drawback of the present work; the grid must be refined locally and the refinement propagates through the structured domain. Overset grids, or hanging nodes may be used, but were found by other researchers to reduce the accuracy of solution locally, often even to first order. It is also not clear how one may achieve hanging nodes or overset grids with the datasets and halos of OPS. On top of these issues, hanging nodes introduce more communication over computation which may reduce parallel performance. The subject of local grid refinement is recommended as future work to improve the present methods.
It is recommended to investigate applicability of the present research to be used in an optimisation framework.
Conjugate Heat Transfer simulations and higher order discretisation are the next natural step in the present work, utilising the already existing grid inside the solid.

7.5 Unique selling points

- User time to generate a structured grid of a complex geometry is significantly reduced. This is achieved via Ghost Cell Immersed Boundary Method (GCIBM) and all standard turbulence models and equations can be used without modification to them and only minor and easy modification near the IBM boundaries.

- The code is portable to different architectures (CPU, GPU) without modification and single source code exist for all architectures. No new source needs to be created if a new architecture emerges in future (e.g. Xeon PHI), only the back end adapted.
- Simulations can be performed at much lower cost and performance of the code is maximised by decoupling the parallel implementation from the scientific application. GPU computational power is much cheaper per unit FLOP.
- Implicit temporal and spatial discretisation (filters) can be used and scales effectively, including on massively parallel GPU platforms.

8 References

[1] Roger L. Simpson, C. H. Long, and G. Byun. Study of vortical separation from an axisymmetric hill. *International Journal of Heat and Fluid Flow*, 23(5):582–591, 2002. ISBN: 0142-727X.

[2] T Myrum, S Acharya, and S Dutta. Developing temperature and periodically developed flow , and heat transfer in a ribbed duct. *International Journal of Heat and Mass Transfer*, 40(2):2069 – 2082, 1997.

[3] D. Charbonnier, P. Ott, M. Jonsson, Th Köbke, and F. Cottier. Comparison of numerical investigations with measured heat transfer performance of a film cooled turbine vane. In *Proceedings of the ASME Turbo Expo*, volume 4, pages 571–582, 2008. Issue: PART A.

[4] P G Tucker, C L Rumsey, P R Spalart, R E Bartels, and R T Biedron. Computations of wall distances based on differential equations. *AIAA Journal*, 43(3):539–549, 2005. ISBN: 9781624100314.

[5] Philippe Parnaudeau, Johan Carlier, Dominique Heitz, and Eric Lamballais. Experimental and numerical studies of the flow over a circular cylinder at Reynolds number 3900. *Physics of Fluids*, 20(8), 2008.

[6] Boeing inc. http://www.boeing.com/boeing/commercial/cmo/.

[7] Airbus inc. http://www.airbus.com/company/market/forecast/.

[8] Z J Wang and Learned Hall. High-order computational fluid dynamics tools for aircraft design Subject Areas :. 2014.

[9] A Correia, J Peters, J Levy, S Melly, and F Dominici. Residential exposure to aircraft noise and hospital admissions for cardiovascular diseases: multi airport retrospective study. *Br. Med. J. 347, f5561*, 2013.

[10] A L Hansell. Aircraft noise and cardiovascular disease near Heathrow airport in London: small area study. *Br. Med. J. 347, f5432*, 2013.

[11] Ilan M. Kroo and Nicolas E. Antoine. Framework for Aircraft Conceptual Design and Environmental Performance Studies. *AIAA Journal*, 43(10):2100–2109, 2005. ISBN: 978-1-62410-019-2.

[12] J E Penner. *Aviation and the Global Atmosphere*. Cambridge university press, 1999.

[13] U S National Science and Technology Council. http://www.whitehouse.gov/sites/default/files/ microsites/ostp/aero-rdplan-2010.pdf.

[14] ACARE Europe. http://www.acare4 europe.com/sites/acare4europe.org/files/document/ Create-Final-Report-October-2010.pdf.

[15] The Cleansky project. http://www.cleansky.eu/content/homepage/ aviation-environment.

[16] John C Vassberg and Et.al. Summary of the Fourth AIAA CFD Drag Prediction Workshop. *AIAA Paper*, (July):4547, 2010. ISBN: 9781617389269.

[17] D. Levy, R. Wahls, T. Zickuhr, J. Vassberg, S. Agrawal, S. Pirzadeh, and M. Hemsch. Summary of data from the first AIAA CFD Drag Prediction Workshop. *40th AIAA Aerospace Sciences Meeting & Exhibit*, pages 1–31, 2002. ISBN: 978-1-62410-078-9.

[18] Long He and Danesh K. Tafti. Evaluating the Immersed Boundary Method in a Ribbed Duct for the Internal Cooling of Turbine Blades. *ASME Turbo Expo*, pages 1–10, 2015. ISBN: 9780791856710.

[19] Kevin Menzies. Delivering better power: the role of simulation in reducing the environmental impact of aircraft engines. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 372(2022), 2014.

[20] Je-Chin Han. Recent Studies in Turbine Blade Cooling. *International Journal of Rotating Machinery*, 10(6):443–457, 2004. ISBN: 1023-621X.

[21] D K Tafti, L He, and K Nagendra. Large eddy simulation for predicting turbulent heat transfer in gas turbines. *Phil. Trans. R. Soc. A 372: 20130322*, 372(2022), 2015. ISBN: 1364-503x.

[22] J. Schabacker, A. Bolcs, and B. V. Johnson. PIV Investigation of the Flow Characteristics in an Internal Coolant Passage with Two Ducts Connected by a Sharp 180 0 Bend. *International Gas Turbine, Aeroengine Congress and Exhibition*, (x):1–11, 1998. ISBN: 978-0-7918-7865-1.

[23] C Orozco-Pineiro. ERICKA Project final report. Technical report, Rolls-Royce Derby, 2014.

[24] J.C. Tyacke and P.G. Tucker. Future use of Large Eddy Simulation in Aero-engines. *J.Turbomach.*, 137, 2015. ISBN: 978-0-7918-4561-5.

[25] Suhas V. Patankar. Numerical heat transfer and fluid flow. page 218, 1980. ISBN: 9780891165224.

[26] F.R. Menter. Turbulence Modeling for Engineering Flows. *A Technical Paper from ANSYS, Inc*, pages 1–25, 2011.

[27] Tobias Brandvik and Graham Pullan. SBLOCK: A framework for efficient stencil-based PDE solvers on multi-core platforms. *Proceedings - 10th IEEE International Conference on Computer and Information Technology, CIT-2010, 7th IEEE International Conference on Embedded Software and Systems, ICESS-2010, ScalCom-2010*, (3):1181–1188, 2010. ISBN: 9780769541082.

[28] T Brandvik and G Pullan. An accelerated Navier-Stokes solver for flows in turbomachines. *J. Turbomach. 133*, 2011.

[29] J.P. Rouse, P. Zacharzewski, C.J. Hyde, R. Jefferson-Loveday, A. Morris, and S.T. Kyaw. A case study investigation into the effects of spatially dependent convection coefficients on the fatigue response of a power plant header component. *International Journal of Fatigue*, 113, 2018.

[30] Cheng Chi, Bok Jik Lee, and Hong G. Im. An improved ghost-cell immersed boundary method for compressible flow simulations. *International Journal for Numerical Methods in Fluids*, 83(2):132–148, 2017.

[31] R. Ghias, R. Mittal, and H. Dong. A sharp interface immersed boundary method for compressible viscous flows. *Journal of Computational Physics*, 225(1):528–553, 2007. ISBN: 0022112006.

[32] CS Charles S. Peskin. The immersed boundary method. *Acta Numerica*, 11(January 2002):479–517, 2002. ISBN: 1064827500.

[33] M. D. de Tullio, P. De Palma, G. Iaccarino, G. Pascazio, and M. Napolitano. An immersed boundary method for compressible flows using local grid refinement. *Journal of Computational Physics*, 225(2):2098–2117, 2007.

[34] Francesco Capizzano. High Reynolds Number Simulations by Using an Immersed Boundary Technique. *Academy Colloquium Immersed Boundary Methods*, pages 3–5, 2009.

[35] C S Peskin. Flow patterns around heart valves: a numerical method. *J. Comp. Phys. 10(2)*, pages 252–271, 1972.

[36] Rajat Mittal and Gianluca Iaccarino. Immersed Boundary Methods. *Annual Review of Fluid Mechanics*, 37(1):239–261, 2005.

[37] Fraunhofer Itwm and T Gornak. A goal oriented survey on immersed boundary methods. 235(235), 2013.

[38] J Mohd-Yusof. Development of immersed boundary methods for complex geometries. *Center for Turbulence Research Annual Research Briefs*, pages 325–336, 1998.

[39] Roberto Verzicco, Jamaludin Mohd-Yusof, Paolo Orlandi, and Daniel Haworth. Large eddy simulation in complex geometric configurations using boundary body forces. *AIAA Journal*, 38(JANUARY 1998):427–433, 2000.

[40] J. Mohd-Yusof. Combined immersed boundary/b-spline methods for simulation of flow in complex geometries. *Center for Turbulence Research Annual Research Briefs1*, pages 317–328, 1997. arXiv: 1011.1669v3 ISBN: 9788578110796.

[41] J W Nam and F S Lien. A ghost-cell immersed boundary method for large- eddy simulations of compressible turbulent flows. *International Journal of Computational Fluid Dynamics*, 28(1–2):41–55, 2016.

[42] Krishnamurthy Nagendra, Danesh K. Tafti, and Kamal Viswanath. A new approach for conjugate heat transfer problems using immersed boundary method for curvilinear grid based solvers. *Journal of Computational Physics*, 267:225–246, 2014. Publisher: Elsevier Inc.

[43] Rémi Gautier, Sylvain Laizet, and Eric Lamballais. A DNS study of jet control with microjets using an immersed boundary method. *International Journal of Computational Fluid Dynamics*, 28(6):393–410, 2014.

[44] YH Tseng and JH Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics*, 192(2):593–623, 2003. ISBN: 0021-9991.

[45] E.A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics*, 161(1):35–60, 2000. arXiv: 1501.0228 ISBN: 00219991.

[46] S Pantula, M H Lu, and W W Liu. Calculations of turbulent flow around airfoils with Attached Flexible Fin using and Immersed Boundary method. *AIAA 2009-721. Proc. 47th AIAA Aerosp. Scien. Meeting Inc. the New Horizons forum and Aerosp. Exposition. 5-8 Jan 2009, Orlando, USA*, 2009.

[47] By Gianluca Iaccarino, Georgi Kalitzin, and Christopher J Elkins. Numerical and experimental investigation of the turbulent flow in a ribbed serpentine passage. pages 379–387, 2003.

[48] A. Gilmanov, F. Sotiropoulos, and E. Balaras. A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartersian grids. *Journal of Computational Physics*, 191(2):660–669, 2003. ISBN: 0021-9991.

[49] Elias Balaras. Modeling complex boundaries using an external force field

on fixed Cartesian grids in large-eddy simulations. *Computers and Fluids*, 33(3):375–404, 2004. ISBN: 0045-7930.

[50] D K Tafti, L He, and K Nagendra. Large eddy simulation for predicting turbulent heat transfer in gas turbines. *Philosophical Transactions of the Royal Society a-Mathematical Physical and Engineering Sciences*, 372(2022), 2014. ISBN: 1364-503x.

[51] Alejandro Allievi and Rodolfo Bermejo. A generalized particle search-locate algorithm for arbitrary grids. *Journal of Computational Physics*, 132(2):157–166, 1997.

[52] F. Roman, E. Napoli, B. Milici, and V. Armenio. An improved immersed boundary method for curvilinear grids. *Computers and Fluids*, 38(8):1510–1527, 2009. Publisher: Elsevier Ltd ISBN: 0045-7930.

[53] F. Roman, V. Armenio, J. Fröhlich, V. Armenio, J. Fröhlich, and V. Armenio. A simple wall-layer model for large eddy simulation with immersed boundary method. *Physics of Fluids*, 21(June), 2009.

[54] Sekhar Majumdar, Gianluca Iaccarino, and Paul Durbin. RANS solvers with adaptive structured boundary non-conforming grids. *Annual Research Briefs*, pages 353–366, 2001.

[55] Adam Preece. An Investigation into Methods to aid the Simulation of Turbulent Separation Control. *PhD Thesis*, (April), 2008.

[56] Mayank Tyagi, Somnath Roy, Albert D. Harvey, and Sumanta Acharya. Simulation of laminar and turbulent impeller stirred tanks using immersed boundary method and large eddy simulation technique in multi-block curvilinear geometries. *Chemical Engineering Science*, 62(5):1351–1363, 2007.

[57] R. Verzicco, M. Fatica, G. Iaccarino, and P. Orlandi. Flow in an impeller-stirred tank using an immersed-boundary method. *AIChE Journal*, 50(6):1109–1118, 2004. ISBN: 9780080445441.

[58] Xiaolei Yang, Xing Zhang, Zhilin Li, and Guo Wei He. A smoothing technique for discrete delta functions with application to immersed boundary

method in moving boundary simulations. *Journal of Computational Physics*, 228(20):7821–7836, 2009. arXiv: 0911.5187 Publisher: Elsevier Inc. ISBN: 0021-9991.

[59] J. O'Rourke. *Computational Geometry in C (J. O'Rourke)*. Smith College, Massachusetts, 2008.

[60] Gianluca Iaccarino and Roberto Verzicco. Immersed boundary technique for turbulent flow simulations. *Applied Mechanics Reviews*, 56(3):331, 2003. ISBN: 00036900.

[61] J Kim. An Immersed-Boundary Finite-Volume Method for Simulations of Flow in Complex Geometries. *Journal of Computational Physics*, 171(1):132–150, 2001.

[62] MATLAB. https://uk.mathworks.com/matlabcentral/fileexchange/.

[63] CGAL. https://github.com/CGAL/releases/blob/master/examples/AABB_tree/AABB_polyhedro

[64] M. B. Giles and I. Reguly. Trends in high-performance computing for engineering calculations. *Philosophical Transactions of the Royal Society A*, 372(2022):20130319, 2014.

[65] Tobias Brandvik and Graham Pullan. An Accelerated 3D Navier–Stokes Solver for Flows in Turbomachines. In *ASME Turbo Expo 2009: Power for Land, Sea and Air*, volume 133, pages 1–11, 2009. Issue: 2 ISSN: 0889504X.

[66] Stan Posey. Considerations for GPU acceleration of parallel CFD. *Procedia Engineering*, 61:388–391, 2013. ISBN: 1877-7058.

[67] G R Mudalige, I Z Reguly, M B Giles, W Gaudin, J A Herdman, and A Mallinson. High-level Abstractions for Performance , Portability and Continuity of Scientific Software on Future Computing Systems - CloverLeaf 3D. pages 1–18, 2015.

[68] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell. A Survey of General Purpose Computation on Graphics Hardware. 26(1):80–113, 2007. ISBN: 1467-8659.

[69] Seyong Lee and Jeffrey S. Vetter. Early evaluation of directive-based GPU programming models for productive exascale computing. *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, (iii), 2012. ISBN: 9781467308069.

[70] Paul Tucker, Simon Eastwood, Christian Klostermeier, Richard Jefferson-Loveday, James Tyacke, and Yan Liu. Hybrid LES Approach for Practical Turbomachinery Flows: Part 2—Further Applications. *ASME Turbo Expo 2010: Power for Land, Sea and Air*, 134(March):2, 2010. ISBN: 9780791844021.

[71] D Curran, C B Allen, and D Beckingsale. DEVELOPING A FUTURE-PROOF CFD CODE. *international conference on parallel computational fluid dynamics*, pages 4–5.

[72] Istvan Zoltan Reguly, Gihan R. Mudalige, and Michael B. Giles. Design and development of domain specific active libraries with proxy applications. *Proceedings - IEEE International Conference on Cluster Computing, ICCC*, 2015-Octob:738–745, 2015. ISBN: 9781467365987.

[73] Istv??n Z. Reguly, Gihan R. Mudalige, Carlo Bertolli, Michael B. Giles, Adam Betts, Paul H J Kelly, and David Radford. Acceleration of a Full-Scale Industrial CFD Application with OP2. *IEEE Transactions on Parallel and Distributed Systems*, 27(5):1265–1278, 2016. arXiv: 1403.7209 ISBN: 1045-9219.

[74] Oxford O P S main website. http://www.oerc.ox.ac.uk/projects/ops.

[75] D Jackson, P Ireland, and B Cheong. Combined Experimental and CFD Study of a HP Blade Multi-Pass Cooling System. *ASME Conference Proceedings*, 2009(48845):851–862, 2009.

[76] S Amaral et.al. Design and Optimization of the Internal Cooling Channels of a High Pressure TurbineBlade—Part I:Methodology. *J. Turbomach. 132(2)*, 2010.

[77] C P Bell, W N Dawes, J P Jarrett, and P J Clarkson. TURBINE ROTOR BLADE COOLING SYSTEMS 1 Introduction 2 Turbine blade cooling background. pages 1–10, 2005.

[78] P. G. Tucker. Trends in turbomachinery turbulence treatments. *Progress in Aerospace Sciences*, 63:1–32, 2013. arXiv: 1011.1669v3 Publisher: Elsevier ISBN: 0376-0421.

[79] J C Tyacke and P G Tucker. Future use of Large Eddy Simulation in Aero engines. *J. Turbomach 137(8)*, 2015.

[80] P G Tucker and J R DeBonis. Aerodynamics, computers and the environment. *Philosophical Transactions of the Royal Society A*, 372:20130331, 2014.

[81] A K Majumdar, V S Pratap, and D B Spalding. Numerical computation of flow in rotating ducts. *J. Fluids Eng. 99*, pages 148–153, 1977.

[82] H Iacovides and B E Launder. Parametric and numerical study of fully developed flow and heat transfer in rotating rectangular ducts. *J. Turbomach 113(90)*, 1991.

[83] Sandip Dutta, Malcolm J. Andrews, and Je Chin Han. Prediction of turbulent heat transfer in rotating smooth square ducts. *International Journal of Heat and Mass Transfer*, 39(12):2505–2514, 1996. ISBN: 0017-9310.

[84] J C Han and Y M Zhang. Effects of uneven wall temperature on local heat transfer in a rotating square channel with smooth walls and radial outward flow. *J. Heat Transfer 114*, pages 850–858, 1992.

[85] J C Han. Heat Transfer and Friction Characteristics in Rectangular Channels With Rib Turbulators. *J. Heat Transfer 110(2)*, pages 321–328, 1988.

[86] J C Han and J S Park. Developing heat transfer in rectangular channels with rib turbulators. *Int. J. Heat Mass Transfer 31 (1)*, pages 183–195, 1988.

[87] M. E. Taslim, T. Li, and S. D. Spring. Measurements of Heat Transfer Coefficients and Friction Factors in Rib-Roughened Channels Simulating Leading-Edge Cavities of a Modern Turbine Blade. *Journal of Turbomachinery*, 119(3):601, 1997. ISBN: 9780791878811.

[88] Y M Zhang, W. Z. Gu, and J. C. Han. Augmented heat transfer in triangular ducts with full and partial ribbed walls. *Journal of Thermophysics and Heat Transfer*, 8(3):574–579, 1994.

[89] Akhilesh P Rallabandi, Huitao Yang, and Je-Chin Han. Heat Transfer and Pressure Drop Correlations for Square Channels With 45 Deg Ribs at High Reynolds Numbers. *Journal of Heat Transfer*, 131(7):71703, 2009.

[90] H. Iacovides and B. E. Launder. Internal blade cooling: The Cinderella of computational and experimental fluid dynamics research in gas turbines. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, 221(3):265–290, 2007.

[91] M Schüler, S O Neumann, and B Weigand. Experimental investigations of pressure loss and heat transfer in a 180° bend of a ribbed two-pass internal cooling channel with engine-similar cross-sections. *Proc. Inst. Mech. Eng., Part A (J. Pow. Ener.)*, 223:709 – 719, 2009. ISBN: 9783851250367.

[92] S. W. Lee, H. S. Ahn, and S. C. Lau. Heat, Mass Transfer Distribution in a Two-Pass Trapezoidal Channel With a 180 deg Turn. *Journal of Heat Transfer*, 129(11):1529, 2007.

[93] ERICKA project. http://cordis.europa.eu/result/rcn/144025_en.html.

[94] P M Ligrani, M. M. Oliveira, and T. Blaskovich. Comparison of Heat Transfer Augmentation Techniques. *AIAA Journal*, 41(3):337–362, 2003.

[95] S. Gupta, A. Chaube, and P. Verma. Review on heat transfer augmentation techniques: Application in gas turbine blade internal cooling. *Journal of Engineering Science and Technology Review*, 5(1):57–62, 2012.

[96] Z J Wang, Krzysztof Fidkowski, Francesco Bassi, Doru Caraeni, Andrew Cary, Herman Deconinck, Ralf Hartmann, Koen Hillewaert, H T Huynh, Norbert Kroll, Georg May, Per-olof Persson, Bram Van Leer, and Miguel Visbal. High-Order CFD Methods : Current Status and Perspective. *International Journal for Numerical Methods in Fluids*, pages 1–42, 2012.

[97] ZhiJian Wang. A perspective on high-order methods in computational fluid dynamics. *Science China Physics, Mechanics & Astronomy*, 59(1):1–6, 2015. ISBN: 1978646860.

[98] Scott E. Sherer and James N. Scott. High-order compact finite-difference methods on general overset grids. *Journal of Computational Physics*, 210(2):459–496, 2005. ISBN: 0021-9991.

[99] Jan Delfs. An overlapped grid technique for high resolution CAA schemes for complex geometries. *7th AIAA/CEAS Aeroacoustics Conference and Exhibit*, (May):1–11, 2001.

[100] Sanjiva K Lele and Joseph W Nichols. A second golden age of aeroacoustics? *Philosophical Transactions of the Royal Society a-Mathematical Physical and Engineering Sciences*, 372(2022, SI):1–16, 2014. ISBN: 9781461383420.

[101] Donald P. Rizzetta, Miguel R. Visbal, and Philip E. Morgan. A high-order compact finite-difference scheme for large-eddy simulation of active flow control. *Progress in Aerospace Sciences*, 44(6):397–426, 2008. ISBN: 9781563479373.

[102] B P Leonard, M K Macvean, and A P Lock. The flux-integral method for multidimensional convection and diffusion. *Applied Mathematical Modelling*, 19(6):333–342, 1994. ISBN: ICOMP-94-13.

[103] B. P. Leonard. Simple high???accuracy resolution program for convective modelling of discontinuities. *International Journal for Numerical Methods in Fluids*, 8(10):1291–1318, 1988. ISBN: 1097-0363.

[104] James Tyacke, Paul Tucker, Richard Jefferson-Loveday, Nagabushana Rao Vadlamani, Robert Watson, Iftekhar Naqavi, and Xiaoyu Yang. LES for Turbines: Methodologies, Cost and Future Outlooks. *J. Turbomach.*, 136, 2013. ISBN: 978-0-7918-5523-2.

[105] Sanjiva K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, 1992. arXiv: DOI: 10.1002/fld.1 ISBN: 0021-9991.

[106] Miguel R. Visbal and Datta V. Gaitonde. High-order-accurate methods for complex unsteady subsonic flows. *AIAA Journal*, 37(10):1231–1239, 1999.

[107] Miguel R Visbal and Datta V Gaitonde. On the Use of Higher-”Order Finite-Difference Schemes on Curvilinear and Deforming Meshes. *Journal of Computational Physics*, 181(1):155–185, 2002. ISBN: 0021-9991.

[108] PETSc, https://petsc.org/release/.

[109] L.H. Thomas. Elliptic Problems in Linear Differential Equations over a Network. *Report; Columbia University: New York, NY, USA*, 1949.

[110] Endre László, Mike Giles, Jeremy Appleyard, Endre L Aszl, Mike Giles, Jeremy Appleyard, Endre László, Mike Giles, and Jeremy Appleyard. Manycore Algorithms for Batch Scalar and Block Tridiagonal Solvers. *ACM Transactions on Mathematical Software*, 42(4):1–36, 2016.

[111] Lars Davidson and Simon Dahlström. Hybrid LES-RANS: An approach to make LES applicable at high Reynolds number. *International Journal of Computational Fluid Dynamics*, 19(March 2015):415–427, 2005.

[112] Lars Davidson. The SAS model: A turbulance model with controlled modelled dissipation. *20th Nordic Seminar on Computational Mecanics*, 5:6–9, 2007.

[113] P Tucker, S Eastwood, C Klostermeier, R Jefferson-Loveday, J Tyacke, and Y Liu. Hybrid LES Approach for Practical Turbomachinery flows - Part 1: Hierarchy and Example Simulations. *J. Turbomach 134*, 2012.

[114] F Menter, M Kuntz, and R Bender. A Scale-Adaptive Simulation Model for Turbulent Flow Predictions. *AIAA Paper*, 2003-0767(January), 2003. ISBN: 978-1-62410-099-4.

[115] F.R. Menter and Y Egorov. Formulation of the Scale-Adaptive Simulation (SAS) Model during the DESIDER Project. *DESider - A European Effort on Hybrid RANS-LES Modelling*, pages 51–62, 2009.

[116] Florian R. Menter and Yury Egorov. SAS Turbulence Modelling of Technical Flows. *Journal of Chemical Information and Modeling*, 53(9):1689–1699, 2013. arXiv: 1011.1669v3 ISBN: 9788578110796.

[117] L. Davidson and M. Billson. Hybrid RANS-LES using synthetized turbulence for forcing at the interface. *European Congress on Computational Methods in Applied Sciences and Engineering*, pages 1–18, 2004.

[118] L. Davidson and S. Dahlström. Hybrid LES-RANS: Computation of the Flow Around a Three-Dimensional Hill. *Engineering Turbulence Modelling and Experiments 6*, pages 319–328, 2005. ISBN: 9780080445441.

[119] Basman Elhadidi and H. Ezzat Khalifa. Comparison of coarse grid lattice Boltzmann and Navier Stokes for real time flow simulations in rooms. *Building Simulation*, 6(2):183–194, 2013.

[120] Simon Mari??, Denis Ricot, and Pierre Sagaut. Comparison between lattice Boltzmann method and Navier-Stokes high order schemes for computational aeroacoustics. *Journal of Computational Physics*, 228(4):1056–1070, 2009. Publisher: Elsevier Inc. ISBN: 0021-9991.

[121] Exa Corporation. http://exa.com.

[122] Shin K. Kang and Yassin A. Hassan. A comparative study of direct-forcing immersed boundary-lattice Boltzmann methods for stationary complex boundaries. *International Journal for Numerical Methods in Fluids*, 66(9):1132–1158, July 2011.

[123] Hudong Chen, Shiyi Chen, and William H Matthaeus. Recovery of the Navier-Stokes equations using a lattice-gas Soltzmann method. *Physical Review A*, 45(8):5339–5342, 1992. ISBN: 1050-2947.

[124] Jing Lei Xu, Chao Yan, and Jing Jing Fan. Computations of wall distances by solving a transport equation. *Applied Mathematics and Mechanics (English Edition)*, 32(2):141–150, 2011. ISBN: 1048301114018.

[125] P. G. Tucker. Hybrid Hamilton-Jacobi-Poisson wall distance function model. *Computers and Fluids*, 44(1):130–142, 2011. ISBN: 0045-7930.

[126] P Parnaudeau, D Heitz, E Lamballais, and J H Silvestrini. Combination of the immersed boundary method with compact schemmes for DNS of flows in complex geometry. page 10p, 2003.

[127] Sylvain Laizet and Ning Li. Incompact3d: A powerful tool to tackle turbulence problems with up to O(105) computational cores. *International Journal for Numerical Methods in Fluids*, 67(11):1735–1757, 2011.

[128] Alejandro Gronskis and Guillermo Artana. A simple and efficient direct forcing immersed boundary method combined with a high order compact scheme for simulating flows with moving rigid boundaries. *Computers and Fluids*, 124:86–104, 2016. ISBN: 00457930.

[129] Cédric Flageul, Sofiane Benhamadouche, Éric Lamballais, and Dominique Laurence. DNS of turbulent channel flow with conjugate heat transfer: Effect of thermal boundary conditions on the second moments and budgets. *International Journal of Heat and Fluid Flow*, 55(May):34–44, 2015. ISBN: 0142-727X.

[130] Sylvain Laizet and Eric Lamballais. High-order compact schemes for incompressible flows: A simple and efficient method with quasi-spectral accuracy. *Journal of Computational Physics*, 228(16):5989–6015, 2009. Publisher: Elsevier Inc. ISBN: 0021-9991.

[131] Incompact3d. No Title.

[132] Junjie Xia, Kun Luo, and Jianren Fan. A ghost-cell based high-order immersed boundary method for inter-phase heat transfer simulation. *International Journal of Heat and Mass Transfer*, 75:302–312, 2014. Publisher: Elsevier Ltd ISBN: 00179310.

[133] Junjie Xia, Kun Luo, and Jianren Fan. Simulating heat transfer from moving rigid bodies using high-order ghost-cell based immersed-boundary method. *International Journal of Heat and Mass Transfer*, 89:856–865, 2015. Publisher: Elsevier Ltd ISBN: 0017-9310.

[134] M. R. Visbal and D. P. Rizzetta. Large-Eddy Simulation on Curvilinear Grids Using Compact Differencing and Filtering Schemes. *Journal of Fluids Engineering*, 124(4):836, 2002.

[135] D. V. Gaitonde and M. R. Visbal. High-order schemes for Navier-Stokes equa-

tions: algorithm and implementation into FDL3DI. *Air Vehicles Directorate*, page 50, 1998.

[136] E. Sewall and D. Tafti. Large Eddy Simulation of Flow and Heat Transfer in the 180 deg bend region of a stationary gas turbine blade ribbed internal cooling duct. *Journal of Turbomachinery*, 128:763–771, 2006. ISBN: 0-7918-4726-8.

[137] Seongwon Kang. An improved immersed boundary method for computation of turbulent flows with heat transfer. *Ph.D. thesis Stanford*, (June):1–125, 2008. ISBN: 9780549622284.

[138] Z Ikram. Numerical Investigation of the effects of free-surface flow past submerged bluff and streamlined bodies. 2011. Publisher: Queen Mary University of London.

[139] R L Manuel, Jonathan Bull, Jacob Crabill, Joshua Romero, Abhishek Sheshadri, Jerry E Watkins Ii, David Williams, Francisco Palacios, and Antony Jameson. Verification and Validation of HiFiLES : a High-Order LES unstructured solver on multi-GPU platforms. *AIAA Aviation*, (June):1–27, 2014. ISBN: 9781624102882.

[140] Freddie David Witherden. *On the Development and Implementation of High-Order Flux Reconstruction Schemes for Computational Fluid Dynamics by*. PhD thesis, 2015. Issue: September.

[141] Thomas H Pulliam and David W Zingg. *Fundamental Algorithms in Computational Fluid Dynamics*. Springer, 2014.

[142] Todd A Oliver. Turbulence Model Equation Documentation. 2009.

[143] P. Spalart and S. Allmaras. A one-equation turbulence model for aerodynamic flows. In *30th Aerospace Sciences Meeting and Exhibit*, Reno,NV,U.S.A., January 1992. American Institute of Aeronautics and Astronautics.

[144] S.-T. Yu. Center for Modeling of Turbulence Research Briefs - 1991 and Transition 93-15802. *Center for Modeling of Turbulence and Transition Research Briefs, NASA Lewis Research Center*, N93-15802, 1991.

[145] A. Jameson. *Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings*. 1991. Publication Title: AIAA 10th Computational Fluid Dynamics Conference.

[146] István Z. Reguly, Gihan R. Mudalige, Michael B. Giles, Dan Curran, and Simon McIntosh-Smith. The OPS domain specific abstraction for multi-block structured grid computations. *Proc. of WOLFHPC 2014*, pages 58–67, 2014. ISBN: 9781479970209.

[147] Kuo-huey Chen. A primitive variable , strongly implicit calculation procedure for two and three-dimensional unsteady viscous flows : applications to compressible and incompressible flows including flows with free surfaces. *Retrospective Theses and Dissertations*, 9485, 1990.

[148] John A. Ekaterinaris. High-order accurate, low numerical diffusion methods for aerodynamics. *Progress in Aerospace Sciences*, 41(3-4):192–300, 2005. ISBN: 0376-0421.

[149] Thomas H Pulliam. Artificial dissipation models for the Euler equations. *AIAA Journal*, 24(12):1931–1940, 1986.

[150] Romit Maulik and Omer San. Evaluation of explicit and implicit LES closures for Burgers turbulence. pages 1–37, 2016. arXiv: 1604.08649.

[151] M. Ciardi, P. Sagaut, M. Klein, and W. N. Dawes. A dynamic finite volume scheme for large-eddy simulation on unstructured grids. *Journal of Computational Physics*, 210(2):632–655, 2005.

[152] John David Anderson. *Computational fluid dynamics: basics with applications*. New York, NY, 1995. Publication Title: McGraw-Hill International editions.

[153] Richard J. Jefferson-Loveday. *Numerical Simulations of Unsteady Impinging Jet Flows*. PhD thesis, Swansea, 2008. Publication Title: Ph.D. thesis Swansea.

[154] R. C. Swanson, R. Radespiel, and E. Turkel. Comparison of Several Dissipation Algorithms for Central Difference Schemes. *Methods*, pages 357–372, 1996.

[155] R C Swanson and E Turkel. Artificial dissipation and central difference schemes for the Euler and Navier-Stokes equations. *AIAA Comput. Fluid Dynamics Conf.*, (October 2016):AIAA 87–1107–CP, 1987.

[156] Antony Jameson, W Schmidt, and E Turkel. Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes. *AIAA paper*, n/a(n/a):n/a, 1981. ISBN: 9781441976314.

[157] Suhas V. Patankar. Numerical heat transfer and fluid flow. page 218, 1980. ISBN: 9780891165224.

[158] Dale A Anderson, John C Tannehill, Richard H Pletcher, Munipalli Ramakanth, and Vijaya Shankar. *Computational fluid mechanics and heat transfer*. CRC Press, 2020.

[159] I J Keshtiban, F Belblidia, and M F Webster. Compressible flow solvers for low Mach number flows – a review. *Technical Report CSR2, Institute of Non-Newtonian Fluid Mechanics, University of Wales*, pages 1–12, 2004.

[160] S. V. Patankar, C. H. Liu, and E. M. Sparrow. Fully Developed Flow and Heat Transfer in Ducts Having Streamwise-Periodic Variations of Cross-Sectional Area. *Journal of Heat Transfer*, 99(2):180, 1977. ISBN: 0022-1481.

[161] Y Egorov and F Menter. Development and application of SST-SAS turbulence model in the DESIDER project. In *Proc. Num. Fluid Mech. and Multidis. Design*, volume 97, pages 261–270, 2008. ISSN: 16122909.

[162] P. Zacharzewski, K. Simmons, R. Jefferson-Loveday, and L. Capone. Evaluation of the sst-sas model for prediction of separated flow inside turbine internal cooling passages. In *Proceedings of the ASME Turbo Expo, GT2016-56117*, 2016.

[163] Thibault Dairay, Eric Lamballais, Sylvain Laizet, and John Christos Vassilicos. Numerical dissipation vs. subgrid-scale modelling for large eddy simulation. *Journal of Computational Physics*, 337:252–274, 2017. Publisher: Elsevier Inc.

[164] Ulka Gaitonde, Dominique Laurence, and Alistair Revell. Quality Criteria for Large Eddy Simulation. *Test*, (May), 2006.

[165] MATLAB inc. MATLAB pwelch function, https://www.mathworks.com/help/signal/ref/pwelch.html.

[166] F. R. Menter, A. Garbaruk, P. Smirnov, D. Cokljat, and F. Mathey. Scale-adaptive simulation with artificial forcing. *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, 111:235–246, 2010. ISBN: 9783642141676.

[167] inc. ANSYS. ANSYS Fluent R17.1.

[168] W.L. Rhie, C.M and Chow. “Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation.”. *AIAA Journal 21*, 11(11):1525–1532, 1983. ISBN: 0001-1452\r1533-385X.

[169] Lars Davidson. Evaluation of the SST-SAS model: Channel flow, asymmetric diffuser and axi-symmetric hill. *European Conference on Computational Fluid Dynamics (ECCOMAS CFD)*, pages 1–20, 2006.

[170] R Spalart. Young-Person’s Guide Simulation Grids Detached-Eddy. *NASA Technical Note*, 211032(July):1003–1008, 2001. ISBN: 0025600907.

[171] F. Tessicini, N. Li, and M. A. Leschziner. Large-eddy simulation of three-dimensional flow around a hill-shaped obstruction with a zonal near-wall approximation. *International Journal of Heat and Fluid Flow*, 28(5):894–908, 2007.

[172] M. García-Villalba, N. Li, W. Rodi, and M. a. Leschziner. Large-eddy simulation of separated flow over a three-dimensional axisymmetric hill. *Journal of Fluid Mechanics*, 627:55–96, 2009. ISBN: 0022-1120$\$r1469-7645.

[173] Yan Liu. *Numerical Simulations Unsteady of Complex Geometry Flows*. PhD thesis, Warwick, 2004.

[174] G. Byun and R. L. Simpson. Structure of Three-Dimensional Separated Flow on an Axisymmetric Bump. *AIAA Journal*, 44(5):999–1008, 2006. ISBN: 9781600867392.

[175] T Persson and M Liefvendahl. comparison of des and les for separated flow over an axisymmetric hill.

[176] Robert H Kraichnan. Diffusion by a Random Velocity Field. *Physics of Fluids*, 13(1):22, 1970. arXiv: 1011.1669v3 ISBN: 1070-6631.

[177] a. Smirnov, S. Shi, and I. Celik. Random Flow Generation Technique for Large Eddy Simulations and Particle-Dynamics Modeling. *Journal of Fluids Engineering*, 123(2):359, 2001. ISBN: 0098-2202.

[178] Lars Davidson. Inlet boundary conditions for embedded LES. *Proceedings of 1st CEAS European Air and Space Conference Century Perspectives*, (September):1–10, 2007.

[179] Y. Egorov, F. R. Menter, R. Lechner, and D. Cokljat. The scale-adaptive simulation method for unsteady turbulent flow predictions. part 2: Application to complex flows. *Flow, Turbulence and Combustion*, 85(1):139–165, 2010. ISBN: 1386-6184.

[180] Anthony Keating and Ugo Piomelli. Synthetic Generation of Inflow Velocities for Large-Eddy Simulation. *34th AIAA Fluid Dyn. Conf.*, pages 1–13, 2004. ISBN: 978-1-62410-031-4.

[181] Aroon K. Viswanathan and Danesh K. Tafti. Detached eddy simulation of turbulent flow and heat transfer in a two-pass internal cooling duct. *International Journal of Heat and Fluid Flow*, 27(1):1–20, 2006. ISBN: 0046-71380142-727X0961-5539.

[182] Evan A. Sewall, Danesh K. Tafti, Andrew B. Graham, and Karen A. Thole. Experimental validation of large eddy simulations of flow and heat transfer in a stationary ribbed duct. *International Journal of Heat and Fluid Flow*, 27(2):243–258, 2006. ISBN: 0142-727X.

[183] J. Tyacke and P. G. Tucker. Large eddy simulation of turbine internal cooling ducts. *Computers and Fluids*, 114:130–140, 2015. Publisher: Elsevier Ltd ISBN: 9780791856352.

[184] S.B. Pope. *Turbulent Flows.pdf*. 2000.

[185] Evan A Sewall and Danesh K Tafti. Large Eddy Simulation of the Developing Region of a Rotating Ribbed Internal Turbine Blade Cooling Channel. *ASME Conference Proceedings*, 2004(41685):735–747, 2004.

[186] K Saha and Sumanta Acharya. Flow and Heat Transfer in an Internally Ribbed Duct With Rotation: An Assessment of LES and URANS. *ASME Conference Proceedings*, 2003:481–495, 2003. ISBN: 0-7918-3688-6.

[187] R. G. Hibbs, S. Acharya, Y. Chen, and D. E. Nikitopoulos. *Heat/mass transfer distribution in a rotating, two-pass channel with smooth and ribbed walls*. 1996. ISSN: 02725673.

[188] Artur Szymanski and Slawomir Dykas. Unsteady flow field evaluation in labyrinth seals by means of computational fluid dynamics. pages 76 – 85, 2016.

[189] Yuewen Jiang, Lintong He, Luigi Capone, and Eduardo Romero. Investigation of Steady and Unsteady Film-Cooling Using Immersed Mesh Blocks With New Conservative Interface Scheme. page V05CT12A007, June 2016.

[190] Yuewen Jiang, Luigi Capone, Peter Ireland, and Eduardo Romero. A Detailed Study of the Interaction Between Two Rows of Cooling Holes. *Journal of Turbomachinery*, 140, December 2017.

[191] Siemens NX, https://www.plm.automation.siemens.com/global/en/products/nx/, accessed 26 September 2021.

[192] Andrea Milli and Shahrokh Shahpar. PADRAM: Parametric Design and Rapid Meshing System for Complex Turbomachinery Configurations. In *Proceedings of the ASME Turbo Expo*, volume 8, June 2012.

[193] inc. ANSYS. ANSYS Space Claim Design Modelled (SCDM) R17.1.

[194] Wipro ITI Global. CADFix, https://www.iti-global.com/cadfix.

[195] inc. ANSYS. ANSYS ICEM CFD R17.1.

[196] ltd. beta cae. Beta CAE preprocessor. https://www.beta-cae.com/.

[197] inc. PTC. Creo R16.

[198] inc. kitware. ParaView, paraview.org.

[199] inc. pointwise. Pointwise CAE. pointwise.com.

[200] J. PFITZNER. Poiseuille and his law. *Anaesthesia*, 31(2):273–275, 1976. ISBN: 1365-2044.

[201] S P Sutera and R Skalak. The History of Poiseuille's Law. *Annual Review of Fluid Mechanics*, 25(1):1–20, 1993. ISBN: 0199988519410.

[202] M. Biswas, G. Breuer and F. Durst. Backward-Facing Step Flows for Various Expansion Ratios at Low and Moderate Reynolds Numbers. *Journal of Fluids Engineering*, 126(3):362, 2004. ISBN: 0098-2202.

[203] J. Kim, S. J. Kline, and J. P. Johnston. Investigation of a reattaching turbulent shear layer Flow over a backward-facing step. *ASME Journal of Fluids Engineering*, 102(3):302–308, 1980. ISBN: 00982202 (ISSN).

[204] A G Kravchenko and P Moin. Numerical studies of flow over a circular cylinder at Re= 3900. *Physics of Fluids*, 12(2000):403–417, 2000.

[205] Hong-Sik Im and Ge-Cheng Zha. Delayed Detached Eddy Simulation of a Stall Flow Over NACA0012 Airfoil Using High Order Schemes. *Proceedings of the 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exhibition*, (January):1–16, 2011.

[206] W. J. McCroskey. A critical assessment of wind tunnel results for the NACA 0012 airfoil. Technical report, 1987. Publication Title: A Critical Assessment of Wind Tunnel Results for the NACA 0012 Airfoil.

[207] L. K. Loftin Jr. Airfoil Section Characteritics at High Angles of Attack. Technical report, 1954.

[208] Philipp Schlatter and Ramis Örlü. Assessment of direct numerical simulation data of turbulent boundary layers. *Journal of Fluid Mechanics*, 659:116–126, 2010. ISBN: 0022-1120.

[209] A. A. Wray. A Selection of Test Cases for the Validation of Large-Eddy Simulations of Turbulent Flows. (345), 1998.

[210] NASA CFD validation database, https://turbmodels.larc.nasa.gov/channelflow_val.html. accessed on 23 Mar 2016.

[211] Robert D. Moser, John Kim, and Nagi N. Mansour. Direct numerical simulation of turbulent channel flow up to Reτ=590. *Physics of Fluids*, 11(4):943–945, 1999. arXiv: 1410.7809 ISBN: doi:10.1017/jfm.2015.268.

[212] D. Home, M. F. Lightstone, and M. S. Hamed. Validation of DES-SST based turbulence model for a fully developed turbulent channel flow problem. *Numerical Heat Transfer; Part A: Applications*, 55(4):337–361, 2009.

[213] Theodore Von Karman. Technical report no. 611. Mechanical similitude and turbulence.

[214] A. K. M. F. Hussain and W. C. Reynolds. Measurements in fully developed turbulent channel flow. *Journal of Fluids Engineering*, 97(4):568–578, 1975. ISBN: 0098-2202.

[215] Bengt Fornberg. Steady viscous flow past a sphere at high Reynolds numbers. *Journal of Fluid Mechanics*, 190:471, 1988.

[216] Sadatoshi Taneda. Experimental Investigation of the Wakes behind Cylinders and Plates at Low Reynolds Numbers. *Journal of the Physical Society of Japan*, 11(3):302–307, 1956. ISBN: 1100019502.

[217] S Balabani, Dip Eng, and M Yianneskis. An experimental study of the mean flow and turbulence structure of cross-flow over tube bundles. 210, 1996.

[218] Anatol Roshko. Experiments on the flow past a circular cylinder at very high Reynolds number. *J. Fluid Mech*, (1924):345–356, 1961. ISBN: 0022-1120.

[219] Abrar Mohammad, Z.J. Wang, and Chunlei Liang. Large Eddy Simulation of Flow Over a Cylinder Using High-Order Spectral Difference Method. *26th AIAA Applied Aerodynamics Conference*, pages 1–14, 2008. ISBN: 978-1-60086-987-7.

[220] Dennis C Jespersen, Thomas H Pulliam, and Marissa L Childs. Turbulence Modeling Resource Validation Results. 01:174, 2016.

[221] H Le, , P Moin, and J Kim. Direct Numerical Simulation of Turbulent Flow Over a Backward-Facing Step. *J. Fluid Mech.*, 330:349–374, 1997.

[222] David M Driver. Backward-Facing Step Measurements at Low Reynolds Number, Reh=5000. (February 1994), 1994.

[223] F. Schäfer, M. Breuer, and F. Durst. The dynamics of the transitional flow over a backward-facing step. *Journal of Fluid Mechanics*, 623:85–119, 2009. ISBN: 0022-1120.

[224] T. Lee and D. Mateescu. Experimental and numerical investigation of 2-D backward-facing step flow. *Journal of Fluids and Structures*, 12(6):703–716, 1998.

[225] Eaton J.K. Vogel JC. Combined heat transfer and fluid dynamic measurements downstream of a backward facing step. *J Heat Transfer*, 107:922–929, 1985.

[226] D. R. H. Gillespie, J. C. Ryley, and M. Mcgilvray. Stationary Internal Cooling Passage Experiments for an Engine Realistic Configuration. *Osney Thermo-Fluids Lab*, pages 1–10, 2011.

[227] Georgi Kalitzin and Gianluca Iaccarino. Turbulence modeling in an immersed-boundary RANS method. *Center for Turbulence Research, Annual Research Briefs*, 29(6):415–426, 2002. arXiv: 1512.00567 ISBN: 9781617796029.

[228] Georgi Kalitzin, Xiaohua Wu, and Paul A. Durbin. DNS of fully turbulent flow in a LPT passage. *International Journal of Heat and Fluid Flow*, 24(4):636–644, 2003.

[229] Rory Douglas Stieger. *The Effects of Wakes on Separating Boundary Layers in Low Pressure Turbines*. PhD thesis, 2002. Issue: February.

9 Appendices

9.1 Appendix 1: UDF of forcing

```
/*
* This is Fluent UDF routine to add source term to momentum
* and sink term to the kinetic energy equation for k-omega
* type Scale Adaptive Simulation unsteady model. Based on
* paper:
* Menter, F. R., Garbaruk, A., Smirnov, P., Cokljat, D. and
* Mathey, F, 2010. "Scale-Adaptive Simulation with Artificial
* Forcing", Progress in Hybrid RANS-LES Modelling, NNFM 111,
* pp. 235-246, 2010.
*
* Note: worth having a look through the references as the paper
* is not completely clear on some points.
*
* Author:
* Piotr Zacharzewski
* Gas Turbine and Transmissions Research Centre (G2TRC)
* The University of Nottingham
* zacharzewski.p@gmail.com
*/

/* fluent includes */
#include "udf.h"
#include "mem.h"
#include "sg_udms.h"
#include "unsteady.h"
#include "metric.h"

/* standard C includes */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```c
#include <math.h>
#include <stdbool.h>
#include <time.h>

/*model constants (could get some via macros instead of hardode here)*/
#define M_PI     3.14159265358979323846
#define N_MODES 10
#define C_L      0.5
#define C_MU     0.09
#define SF       1       /* scaling factor for imposed fluctuations */

/* need to do in fluent before accessing variables
/define/user-defined/user-defined-memory 6 */

 static int version = 1;
 static int release = 2;

/*----------------------------------------------------------------------*/
/* RANDOM NUMBER WITH GIVEN MEAN AND STANDARD DEVIATION,
    NEEDED FOR FORCING */
/*----------------------------------------------------------------------*/

void set_seed () {
srand(time(NULL));
}

 real gen_rand() {
 return ((real)rand())/(real)RAND_MAX;
}

 real gen_rand_norm(real mean, real std) {
 return (mean + std*sqrt(-2.0*log(gen_rand())) * cos(2.0*M_PI*gen_rand()));
```

```
}
/*-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------*/

DEFINE_EXECUTE_ON_LOADING( init_case, libname) {

Message("\nLoading %s version %d.%d\n",libname,version, release );
Set_User_Memory_Name(0,"turb_length_scale");
Set_User_Memory_Name(1,"turb_time_scale");
Set_User_Memory_Name(2,"source_mom_x");
Set_User_Memory_Name(3,"source_mom_y");
Set_User_Memory_Name(4,"source_mom_z");
Set_User_Memory_Name(5,"source_tke");

 printf ("Setting the seed for random numbers ...\n");
 set_seed ();

/*
* bug: doesn't display names properly. change UDM amount to 0, then back to
   the required amount
* and the names appear
*/

}

DEFINE_ADJUST(gen_sources_store, domain)
{
/* loop over all cell centres to store the x-y-z and tke source */
 cell_t cell;
Thread *thread;
 thread_loop_c (thread, domain)
{
```

```
begin_c_loop(cell, thread)
{

/* physical time and length scale of turbulence, density, timestep */
real L_t        = (C_L/C_MU)*(sqrt(C_K(cell, thread))/C_O(cell, thread));
real tau_t      = C_L/(C_MU*C_O(cell, thread));
real rho        = C_R(cell, thread);
real time_step  = (real)CURRENT_TIMESTEP;

real eta_x, eta_y, eta_z;
real  xi_x,  xi_y,  xi_z;
real   d_x,   d_y,   d_z;

real omega_n;

int cnt;
real cell_char_size , d_mag, limit_t , limit_l ;

real x[3];
C_CENTROID(x, cell, thread);
real arg_n, p_x, p_y, p_z, q_x, q_y, q_z;
real u_fx, u_fy, u_fz;
u_fx = 0.0;
u_fy = 0.0;
u_fz = 0.0;

/* this must be changed to include max(delta_x, delta_y, delta_z) of a cell
   */
/* it is the characteristic cell length */
 cell_char_size = pow(C_VOLUME(cell, thread), 0.33333333333);

/* Sum up all the modes based on random number generator */
```

```
for (cnt = 0; cnt < N_MODES; ++cnt) {

eta_x    = gen_rand_norm((real) 0.0,  (real) 1.0);
eta_y    = gen_rand_norm((real) 0.0,  (real) 1.0);
eta_z    = gen_rand_norm((real) 0.0,  (real) 1.0);

xi_x     = gen_rand_norm((real) 0.0,  (real) 1.0);
xi_y     = gen_rand_norm((real) 0.0,  (real) 1.0);
xi_z     = gen_rand_norm((real) 0.0,  (real) 1.0);

d_x      = gen_rand_norm((real) 0.0,  (real) 0.5);
d_y      = gen_rand_norm((real) 0.0,  (real) 0.5);
d_z      = gen_rand_norm((real) 0.0,  (real) 0.5);

omega_n = gen_rand_norm((real) 1.0,  (real) 1.0);

/*------------------------------------------------------------*/
/*             NYQUIST LIMITER FOR LENGTH AND TIME SCALE */
/*------------------------------------------------------------*/
d_mag = sqrt(d_x*d_x + d_y*d_y + d_z*d_z);

 limit_t  = 2*time_step*omega_n;
 limit_l  = 2* cell_char_size *d_mag;

if ( tau_t < limit_t ) {  /* something is wrong with max() in fluent */
tau_t = limit_t ;
}
if (L_t < limit_l ) {
L_t = limit_l ;
}
/*------------------------------------------------------------*/
/*------------------------------------------------------------*/
```

```
/*---------------------------------------------------------------*/
/*              THE FORCING TERM CALCULATION                      */
/*---------------------------------------------------------------*/
arg_n = 2*M_PI*((d_x*x[0] + d_y*x[1] + d_z*x[2])/L_t + (omega_n*CURRENT_TIME)/tau_t);

p_x = eta_y*d_z - eta_z*d_y;
p_y = eta_z*d_x - eta_x*d_z;
p_z = eta_x*d_y - eta_y*d_x;

q_x = xi_y*d_z - xi_z*d_y;
q_y = xi_z*d_x - xi_x*d_z;
q_z = xi_x*d_y - xi_y*d_x;

u_fx = u_fx + p_x*cos(arg_n) + q_x*sin(arg_n);
u_fy = u_fy + p_y*cos(arg_n) + q_y*sin(arg_n);
u_fz = u_fz + p_z*cos(arg_n) + q_z*sin(arg_n);

/*---------------------------------------------------------------*/
/*---------------------------------------------------------------*/
}

 real fac = ((sqrt(2)/sqrt(3)) * (sqrt(2)/sqrt(N_MODES)))*sqrt(C_K(cell, thread));
u_fx = u_fx * fac;
u_fy = u_fy * fac;
u_fz = u_fz * fac;

/* apply forcing only in specific region */
/* if (x[1] < 0.02 || x[1] > 0.13) {*/
C_UDMI(cell, thread, 2) = (u_fx * rho) / (time_step);   /* x-momentum source */
C_UDMI(cell, thread, 3) = (u_fy * rho) / (time_step);    /* y-momentum source */
C_UDMI(cell, thread, 4) = (u_fz * rho) / (time_step);    /* z-momentum source */
C_UDMI(cell, thread, 5) = ((u_fx*u_fx + u_fy*u_fy + u_fz*u_fz) * rho * (-0.5)) / time_step; /* tke source          */
/*}
else {
C_UDMI(cell, thread, 2) = 0.0;
C_UDMI(cell, thread, 3) = 0.0;
C_UDMI(cell, thread, 4) = 0.0;
C_UDMI(cell, thread, 5) = 0.0;
}*/

/* diagnostics
C_UDMI(cell, thread, 6) = arg_n;
C_UDMI(cell, thread, 7) = pow(C_VOLUME(cell, thread), 0.33333333333);
C_UDMI(cell, thread, 8) = C_VOLUME(cell,thread);
C_UDMI(cell, thread, 9) = q_x;*/
}
end_c_loop (cell, thread)
}

}

/* The main momentum sources passed to fluent */
DEFINE_SOURCE(cell_x_source_mom, cell, thread, dS, eqn)
{
/* derivative of source term w.r.t. x-velocity, always zero in here */
/* forces fluent to treat the source term explicitly */
dS[eqn] = 0.0;

```

```
/*get the source term for x-momentum equation and return*/
/* cell and thread must be passed also */
return C_UDMI(cell, thread, 2);
}

DEFINE_SOURCE(cell_y_source_mom, cell, thread, dS, eqn)
{
/* derivative of source term w.r.t. x-velocity, always zero in here */
dS[eqn] = 0.0;

/*get the source term for y-momentum equation and return*/
/* cell and thread must be passed also */
return C_UDMI(cell, thread, 3);
}

DEFINE_SOURCE(cell_z_source_mom, cell, thread, dS, eqn)
{
/* derivative of source term w.r.t. x-velocity, always zero in here */
dS[eqn] = 0.0;

/*get the source term for z-momentum equation and return*/
/* cell and thread must be passed also */
return C_UDMI(cell, thread, 4);
}

DEFINE_SOURCE(turb_kin_ener_sink, cell, thread, dS, eqn)
{
/* derivative of source term w.r.t. x-velocity, always zero in here */
dS[eqn] = 0.0;

/*get the source term for tke equation and return */
/* cell and thread must be passed also */
```

```
return C_UDMI(cell, thread, 5);
}

/*
 allocate values to UDM memory, various interesting quantities
*/
DEFINE_ON_DEMAND(fill_UDM)
{
Domain *domain;
 cell_t c;
Thread *t;

domain=Get_Domain(1);

 thread_loop_c (t,domain)
{
 begin_c_loop (c,t)
{
C_UDMI(c,t,0) = (C_L/C_MU)*(sqrt(C_K(c, t))/C_O(c, t)); /* L_t   */
C_UDMI(c,t,1) = C_L/(C_MU*C_O(c, t));                    /* tau_t */
}
 end_c_loop (c,t)
}
}
```