Study of high-order energy-stable discretization techniques for compressible flows

G. Giangaspero

Study of high-order energy-stable discretization techniques for compressible flows
G. Giangaspero

Thesis University of Twente, Enschede, The Netherlands
April 2016

ISBN: 978-90-365-4100-8

DOI: 10.3990/1.9789036541008

URL: <http://dx.doi.org/10.3990/1.9789036541008>

Copyright ©2016 by G. Giangaspero

STUDY OF HIGH-ORDER ENERGY-STABLE
DISCRETIZATION TECHNIQUES
FOR COMPRESSIBLE FLOWS

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus
Prof.dr. H. Brinksma
on account of the decision of the graduation committee,
to be publicly defended
on Friday 22 April 2016 at 12:45

by

Giorgio Giangaspero

born on 16 October 1985
at Castel San Pietro Terme (BO), Italy

This dissertation has been approved by the supervisor:
prof. dr. ir. H. W. M. Hoeijmakers

and by the co-supervisor:
dr. ir. E. T. A. van der Weide

SUMMARY

The aerodynamic design of several products of the aerospace and power generation industry relies heavily on the numerical simulations of fluid flows. In the design process of such products, like gas-turbines, wind turbines and aircraft, the accuracy of the numerical results is as crucial as the time needed to obtain them. Thus, the research for more computationally efficient algorithms, i.e. algorithms that yield a result with a given accuracy in less time, has never stopped. The present thesis describes the development of a computational method for the numerical simulation of compressible flows in aerodynamics. The main goal is to study high-order discretization schemes that can potentially deliver a better computational efficiency.

The method of choice for the discretization of the spatial part of the conservation equation is the finite difference method. In particular, Summation by Parts (SBP) finite difference operators and boundary condition treatment via Simultaneous Approximation Terms (SAT) have been investigated. An SBP finite difference operator is essentially a (high-order) centered finite difference scheme with a specific closure at the boundary. The SAT are penalty-like terms that enforce the boundary conditions weakly and are used to augment the SBP schemes. The combination of SBP operators with SAT boundary terms constitutes schemes that, for the linearized equations governing a smooth flow, are provably stable (and thus convergent) schemes. In fact, thanks to the energy stability property of the SBP-SAT schemes, a significantly reduced amount of artificial dissipation is needed compared to schemes which do not possess this (or a similar) property. This leads to more accurate numerical solutions and, potentially, a higher efficiency. In the present work, SBP-SAT schemes of design order of accuracy of 2 to 5 have been employed.

In the field of turbulent flow simulations with high-order finite-difference SBP-SAT schemes, little experience has been gained to date. So far these schemes have been confined either to academic test cases, or, when applied to realistic configurations, only the second order scheme has been used. Thus, a systematic comparison between the efficiency of low and high-order SBP-SAT schemes, especially when used to solve the Navier-Stokes (NS) and Reynolds-Averaged Navier-Stokes (RANS) equations, is lacking. The main goal of the present work is bridge this gap and to assess whether or not these high-order schemes are more computationally efficient than the standard second-order scheme for aerodynamic applications.

The discretization in time is carried out with high-order schemes as well. Both implicit and explicit time integration schemes have been investigated. The system of non-linear equations that arises when using the implicit time integration schemes is solved via a (damped) Newton method. On the target grid, the initial guess required for Newton's method is obtained by performing grid sequencing

combined with a number of explicit relaxation iterations on each grid level. The flow equations and the equation of the turbulence model (if present) are solved in a fully-coupled manner. Domain decomposition is used for the parallelization of the computational method, which has been tested on several machines.

The computational method developed in the present work has been applied to a wide range of compressible flows. Numerical simulations for inviscid flow, laminar flow, as well as turbulent simulations based on Direct Numerical Simulation (DNS), Large Eddy Simulation (LES) and RANS have been performed. In particular, we mention the DNS of the Taylor-Green vortex, a laminar-to-turbulent transition problem tackled with the LES approach, and two engineering applications solved with the RANS approach: a three-element airfoil in high-lift configuration and a transonic axial compressor blade.

Results show that the computational method developed in this work is particularly well-suited for smooth external flows about relatively simple geometries, solved with the DNS and LES approach. For these types of flows the gain in efficiency obtained by using high-order schemes is clear (for instance, more than a factor of 5 for the Taylor-Green vortex case). For RANS, however, the comparison with a state-of-the-art second order finite volume code suggests that solving the turbulence and the flow equations fully coupled with implicit schemes may not be the most efficient iteration strategy.

On the other hand, the computational method developed in the present thesis is not mature enough for non-smooth/internal flows. The numerical simulations of these types of flows highlighted the need of further development of the SBP-SAT framework. Within this framework, three main building blocks are still missing: consistent shock capturing operators, generalized interpolation operators for non-matching grids, and boundary conditions for internal flows. Should these limitations be overcome, then the high-order accurate solution of more complex problems, like turbomachinery flows, will be at reach.

to my little niece Bianca

CONTENTS

SUMMARY i

ACRONYMS ix

NOMENCLATURE xi

1	INTRODUCTION	1
1.1	Background: the COPA-GT project	2
1.2	High-order discretization in space	4
1.2.1	Finite volume methods	6
1.2.2	Finite element methods	7
1.2.3	Finite difference methods	8
1.3	SBP-SAT finite difference	9
1.4	Motivation and goal	10
1.5	Outline of this work	11
2	THE MATHEMATICAL MODEL	13
2.1	Conservation Equations	13
2.1.1	The Navier-Stokes equations	13
2.1.2	Equations of state	14
2.1.3	Constitutive equations	14
2.1.4	Non-inertial frame of reference and Arbitrary Lagrangian-Eulerian formulations	15
2.2	Transformation to curvilinear coordinates	16
2.2.1	Metric terms	17
2.2.2	The transformed equations	19
2.2.3	Free-stream preservation and Geometric Conservation Law	21
2.3	Turbulence modeling	22
2.3.1	URANS	22
2.3.2	LES	27
2.4	Boundary conditions	29
3	NUMERICAL SOLUTION	31
3.1	The method of lines	31
3.2	Spatial discretization	32
3.2.1	Stability proofs using the energy method	33
3.2.2	SBP first-derivative operators	36
3.2.3	Boundary conditions (SAT)	38
3.2.4	Artificial dissipation	45
3.3	Temporal discretization	49

3.3.1	Unsteady flows	50
3.3.2	Steady flows	52
3.4	Solution method: description of the solution procedure	54
3.4.1	Newton's method	54
3.4.2	Initial solution and globalization strategy	56
3.4.3	Efficient computation of the Jacobian	58
3.4.4	Solution of the system of linear equations	69
3.4.5	Backtracking line-search	72
3.5	Aspects of the implementation	73
3.5.1	Load balancing	73
3.5.2	Computation of the wall distance and sliding mesh donor search procedure	74
3.5.3	Computation of forces	81
3.6	Summary of the solution strategy	83
4	NUMERICAL RESULTS	85
4.1	Euler vortex	86
4.1.1	Static grid	87
4.1.2	Moving grid	89
4.2	A rotor-stator interaction problem	90
4.3	NACA0012	94
4.4	Analytical 3D slender body	98
4.5	DNS of Taylor-Green Vortex	101
4.6	LES: Transitional flow over the SD7003 wing	105
4.7	Three-element airfoil	113
4.8	Rotor37	118
4.8.1	Operating and boundary conditions	119
4.8.2	Computational domain	120
4.8.3	Results	120
4.9	Efficiency of the parallel computation	126
5	CONCLUDING REMARKS AND RECOMMENDATIONS	129
5.1	Concluding remarks	129
5.2	Recommendations	131
BIBLIOGRAPHY		135
A	SUMMARY OF THE CONSERVATION EQUATIONS AND CLOSURE RELATIONS	147
B	SBP-SAT OPERATORS	149
B.1	Norms	149
B.2	First derivative operators	149
B.3	Artificial dissipation	154
B.4	SAT terms	155
B.4.1	Euler wall viscous penalty state	155
B.4.2	Block interface viscous stability term ($\tilde{B}_{11}$)	156

B.4.3	Sixth and eighth order interpolation operators	158
C	INVISCID FLUX JACOBIAN	159
C.1	Generic inviscid flux	159
C.2	Eigenvalues and eigenvectors	160
C.3	Transformation to primitive variables	161
C.4	The conservative inviscid flux Jacobian	163
C.5	Commented Maxima code	163
D	BUTCHER TABLEAU FOR THE ESDIRK SCHEMES	169
E	VERIFICATION OF THE IMPLEMENTATION OF THE SPALART-ALLMARAS TUR- BULENCE MODEL	171
E.1	Bump in a channel	171
E.2	NACA4412	172
	ACKNOWLEDGEMENTS	179

ACRONYMS

ADI	Alternating Direction Implicit
ADT	Alternating Digital Tree
ALE	Arbitrary Lagrangian-Eulerian
AoA	angle of attack
ASM	Additive Schwarz Method
BILU	Block Incomplete Lower-Upper
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrich-Lewy
COPA-GT	COupled PArallel simulations of Gas Turbines
CPU	Central Processing Unit
DNS	Direct Numerical Simulation
ESDIRK	first stage Explicit, Singly Diagonally Implicit Runge-Kutta
GCL	Geometric Conservation Law
GMRES	Generalized Minimal RESidual
ILES	Implicit Large Eddy Simulation
ILU	Incomplete Lower-Upper
JST	Jameson-Schmidt-Turkel
LES	Large Eddy Simulation
MPI	Message Passing Interface
NS	Navier-Stokes
PDE	partial differential equation
PETSc	Portable, Extensible Toolkit for Scientific computation
RANS	Reynolds-Averaged Navier-Stokes
RK	Runge-Kutta
SA	Spalart-Allmaras

ACRONYMS

SAT	Simultaneous Approximation Terms
SBP	Summation by Parts
SER	Switched Evolution-Relaxation
ODE	ordinary differential equation
TVD	Total Variation Diminishing
URANS	Unsteady Reynolds-Averaged Navier-Stokes
WALE	Wall-Adapting Local Eddy-viscosity

NOMENCLATURE

The most recurring symbols used in the present thesis are listed here. The main convention is that **bold** is used for vectors, whereas standard non-bold characters refer to either scalars or matrices. The only exception is the viscous stress tensor $\boldsymbol{\tau}$, which is in bold to distinguish it from the (pseudo-)time τ .

Greek Symbols

$\boldsymbol{\omega}$	Vorticity vector $= \nabla \times \mathbf{u} = (\omega_x, \omega_y, \omega_z)^T$ [1/s]
ξ	Curvilinear coordinates vector $(\xi, \eta, \zeta)^T$
ϵ	Turbulent dissipation rate [m^2/s^3]
γ	Ratio of specific heats $\frac{c_p}{c_v} = 1.4$
κ	Heat conduction coefficient [$\text{W}/(\text{m K})$], or Von Kármán constant
μ	Dynamic viscosity [$\text{kg}/(\text{m s})$]
μ_t	Eddy viscosity (turbulent dynamic viscosity) [$\text{kg}/(\text{m s})$]
ν	Kinematic viscosity [m^2/s]
ω	Specific turbulent dissipation rate $= \epsilon/k$ [1/s]
ρ	Density [kg/m^3]
τ	Time or pseudo-time [s]
$\boldsymbol{\tau}$	Viscous stress tensor of components τ_{ij} [$\text{kg}/(\text{m s}^2)$]
$\boldsymbol{\tau}^t$	Reynolds stress tensor of components τ_{ij}^t [$\text{kg}/(\text{m s}^2)$]

Roman Symbols

$\dot{\mathbf{q}}$	Heat flux vector $(\dot{q}_x, \dot{q}_y, \dot{q}_z)^T$ [W/m^2]
Ma	Mach number $ \mathbf{u} /c$
Pr	Prandtl number $(c_p \mu)/(\kappa) = 0.72$ for air
Re	Reynolds number $\rho U L / \mu$
$\mathbf{Q}$	Conserved variables vector $(\rho, \rho \mathbf{u}, \rho E)^T$
$\mathbf{u}$	Velocity vector $(u, v, w)^T$ [m/s]

NOMENCLATURE

$\mathbf{x}$	Coordinates vector $(x, y, z)^T$ [m]
c	Speed of sound $\sqrt{\gamma R T}$ [m/s], or chord length [m]
C_d	Drag coefficient
C_f	Skin friction coefficient
C_l	Lift coefficient
C_p	Pressure coefficient
d	Distance to the nearest wall [m]
E	Specific total internal energy [m^2/s^2]
e	Specific internal energy [m^2/s^2]
H	Specific total enthalpy [m^2/s^2]
h	Specific enthalpy [m^2/s^2], or grid spacing [m]
J	Determinant of the metric Jacobian [$1/\text{m}^3$]
k	Turbulent kinetic energy [m^2/s^2]
p	Pressure [$\text{kg}/(\text{m s}^2)$]
R	Air specific gas constant = 287.87 [J/(kg K)]
T	Temperature [K]
t	Time [s]
U	Contravariant velocity in the ξ direction [1/s]
V	Contravariant velocity in the η direction [1/s]
W	Contravariant velocity in the ζ direction [1/s]

Subscripts/Superscripts

∞	Free-stream
c	Convective, or characteristic
eff	Effective (laminar plus turbulent)
t	Turbulent
w	Wall

1

INTRODUCTION

Computational Fluid Dynamics (CFD) has become an indispensable tool in the design process of a large variety of industrial products, in particular in the aeronautical and power-generation industry. During the past four decades, the ever increasing computational power has continuously pushed CFD towards more and more challenging applications. While the improvement of the hardware for numerical simulations has certainly given a boost to the growth of CFD, the quest for efficient algorithms has played an important role as well. In fact, the development of advanced algorithms has given a comparable increase in the capabilities of numerical simulations [103]. The present thesis contributes to this. The goal is to study potentially more efficient discretization/solution methods for the prediction of compressible flows in aerodynamics.

In numerical simulations of complex physical phenomena, a key trade-off is between the accuracy of the method and its computational cost. Reduced-order models typically introduce simplifications in the mathematical representation by neglecting/approximating a significant part of the physics. On the contrary, the use of high-fidelity models is usually constrained by the available computing resources (or by the available time). Being aware of a method's strengths and weaknesses is essential for an effective use of the algorithms.

In the realm of fluid dynamics, turbulence is a very complex physical phenomenon that clearly shows the necessity of this trade-off. The physical complexity of turbulence is reflected in the numerical complexity of solving its mathematical representation, the Navier-Stokes (NS) equations. While the NS equations are a precise mathematical model of fluid flows, the Direct Numerical Simulation (DNS)¹ of high Reynolds number turbulent flows still exceeds by far the computing power of current supercomputers. Thus, for high Reynolds numbers, the computational cost associated with a very accurate numerical method as DNS is not acceptable. Unfortunately, in engineering applications laminar (turbulence free) flows are rather an exception, while turbulent flows are quite common. Then, in order to reduce the prohibitive computational requirements of such simulations, the only option is to model the effects of turbulence on the flow.

The most common approaches in this sense are performing Large Eddy Simulation (LES) and solving the Reynolds-Averaged Navier-Stokes (RANS) equations, the latter being the cheapest but least high-fidelity modeling approach. Yet RANS methods have become the high-fidelity method of choice in the aerospace industry. While the accuracy of this method has certainly improved in the past decade thanks to progress in turbulence modeling, the main advances have been mostly due to the continually increasing hardware performance for a given price, allowing

¹ That is the numerical solution of the NS equations resolving all the time and spatial scales.

larger meshes, more complex geometries, and more runs [103]. However, RANS methods have well-known limitations for separated flows and laminar-to-turbulent transition. This has limited the reliable use of CFD to a relatively small part of the operating design space. LES would overcome part of these limitations but their computational cost has only recently become affordable for lower-Reynolds-number applications, like for some relevant components of turbomachinery [120].

1.1 BACKGROUND: THE COPA-GT PROJECT

The present work is part of the Marie Curie Initial Training Network project COPA-GT (COupled PArallel simulations of Gas Turbines²), which focuses on the numerical simulation of gas turbines. Gas turbines are versatile engines that are employed for the production of both mechanical and electrical power. Their use is widespread in the air transport industry as large commercial aircrafts are propelled by turbofan gas turbines, while smaller aircrafts and helicopters rely on the turboprop and turboshaft variants, respectively. Gas turbines are also employed for land-based applications, mostly for the generation of electricity and as pump drives for gas or liquid pipe lines. Their main advantages over piston engines are a higher power/weight ratio and the absence of reciprocating parts, the latter simplifies maintenance and increases reliability [20].

The three main components of a gas turbine engine are a compressor, a combustor and a turbine, connected together by one or more shafts, see fig. 1.1. If the engine is operating in an open cycle (as is the case for the vast majority of the applications), fresh atmospheric air is drawn into the compressor continuously. The compressor increases the pressure of the air which is then directed to the combustion chamber. There, energy is added at (ideally) constant pressure by the combustion of fuel and the working fluid itself. The (hot) products of combustion are then expanded through the turbine and exhausted to atmosphere. Depending on the application, the propulsion/work is extracted either through a nozzle that accelerates the flow and gives thrust (in turbojet/turbofan gas turbines), or through an additional turbine (in turboshaft gas turbines). In the latter case, the additional turbine may drive the main rotors of an helicopter, or the alternator to produce electrical power.

The design of such engines is a challenging multi-physics problem, where phenomena like heat-transfer, aerodynamics, combustion, vibrations and noise production, must all be taken into account. To complicate things further, each physical phenomenon is not easy to predict, even when considered alone. In fact, restricting the following discussion to aerodynamics, the flow within these engine is particularly complex as it is characterized by successive accelerations and decelerations, changes in pressure and temperature, static and moving parts. In the compressor, for instance, the working fluid initially flows through the rotor blades, where its absolute velocity is increased. Then the working fluid goes through the stator blade

² see <http://copagt.cerfacs.fr/>.

Figure 1.1: Schematics of a (turbojet) gas turbine engine (modified from the original available at https://upload.wikimedia.org/wikipedia/commons/4/4c/Jet_engine.svg).

passages where the kinetic energy transferred by the rotor is converted to static pressure: the velocity of the fluid is decreased and its pressure is increased. This compression process is carried out in as many stages as are needed to obtain the design overall pressure ratio. Because of the adverse pressure gradient in compressors, the boundary layers along the annulus walls become progressively thicker as the flow goes through the machine. The thicker boundary layers limit the effective area available for the core of the flow thus altering the axial (and relative) velocity locally, in a way that is difficult to predict with simple 1D or 2D models. Moreover, due to adverse pressure gradients, the boundary layer on the suction sides of the compressor blades is also more prone to separation. This limits the diffusion in each stage of the compressor, which can only provide a relatively small compression ratio. These phenomena combined with effect of the body forces due to rotation, the effect of tip clearance, and the unsteady nature of the machine, make the aerodynamic design of the compressor extremely difficult [20].

Furthermore, manufactures of gas turbines cannot ignore the increasingly stringent environmental constraints that push towards higher efficiencies, lower emissions of pollutants and smaller noise signature. The faster, more accurate and reliable the results of CFD are, the more chances the designer of gas turbines has to develop a more efficient and less pollutant engine. These considerations clearly show the importance of CFD as a design and optimization tool in a broader context, not only for gas turbine engines. In fact, the same reasoning holds, for instance, for aircrafts (an optimized aircraft with a smaller drag force consumes less fuel) and for wind turbines (the higher the aerodynamic efficiency, the more electrical energy is produced).

The main objective of the COPA-GT project was to advance the current state-of-the-art of the numerical tools and of the methodologies for the numerical simulations of gas turbines. Currently, gas turbines are typically designed considering (parts of) one single component (compressor, combustor, turbine) and one single physical phenomenon at the time. However, designers face the problem that the behavior of an isolated single component is different from that of the same component when coupled with the other components of the engine [86]. Furthermore, current industrial turbomachinery simulations often solve the RANS or Unsteady Reynolds-Averaged Navier-Stokes (URANS) equations which, as mentioned before,

have known limitations for complex flows. The COPA-GT project addressed these issues and, pushing the boundaries of the current capabilities of CFD for turbomachinery, envisioned two scenarios: LES of relatively small parts of the gas-turbine engine (combustor, combustor/first turbine stage), and RANS simulation of larger parts of the engine such as multiple stages, full wheel of a compressor/turbine, interaction between compressor/combustor/turbine. Due to the huge computational requirements of both these scenarios, an efficient discretization method is certainly desirable. The focus of this thesis is on high-order discretization methods, which can potentially lead to substantial savings in computing costs.

1.2 HIGH-ORDER DISCRETIZATION IN SPACE

High-order schemes are receiving constantly growing attention thanks to their potential in delivering a better computational efficiency, which means less computational work for a given numerical accuracy [131]. In particular, the development of high order-accurate, stable and efficient discretization methods is a very active topic in the field of CFD of compressible flows for aerodynamic applications. In the present work we refer to high-order methods as methods with order of accuracy of at least three. A discretization method is of order p if the discretization error behaves like $\mathcal{O}(h^p)$, where h is the mesh size. Thus, when halving the mesh size h (by doubling the number of degrees of freedom) the discretization error decreases, for instance, by a factor of 16 for a fourth-order scheme and by a factor of only 4 for a second-order scheme. As a consequence, a high-order scheme can provide a solution with a given accuracy on coarser meshes (with less degrees of freedom) compared to a second order scheme.

However, the gain of a high-order scheme is only a potential gain because the discretization error alone does not give enough information to assess a method's efficiency. For the efficiency of the method we need to take into account the computational work (time needed to solve a particular problem on a computer) as well. In fact, for a given number of degrees of freedom, high-order schemes are typically more computationally expensive than second-order schemes. This is mainly due to two reasons. Firstly, a high-order scheme has a larger stencil, meaning that in order to compute the solution in one vertex (or cell) the scheme requires the solution at more neighboring vertices (cells)³. This increases the required number of floating point operations (and more information needs to be communicated in parallel). Secondly, a high-order scheme typically poses a tighter restriction on the time step necessary for stability. This latter issue can partially be circumvented using implicit time integration schemes, although the discretized problem arising from a high-order scheme is typically more computationally expensive to solve. It is only when the computational work and the accuracy are considered together that the efficiency of a method can be assessed. This is one of the major topics of this thesis.

³ For the moment we are not considering finite element discretization methods.

Figure 1.2: Helicopter rotor detached eddy simulation showing the blade-vortex interaction (from <http://www.nasa.gov/centers/ames/images?id=342264>).

Figure 1.3: Wake vortices generated by a landing aircraft (trimmed from the original available at <https://flic.kr/p/hQHwMh>).

Second-order (finite volume) methods are at the moment the *de facto* industry standards and they have been extremely successful (see for example [124]). Yet for some applications second-order accuracy is not the best choice. In comparison to second-order methods, on the same mesh higher-order methods allow a significantly improved resolution of flow features like vortices and wakes. This is clearly important in applications for which these flow features play a substantial role, for example:

- blade-vortex interaction is observed for helicopter blades (fig. 1.2);
- wake-vortices can be seen behind a transport aircraft. At an airport it is necessary to wait until these vortices have dispersed before the next aircraft in line can take off or land (fig. 1.3);
- blade-wake interactions influence the performance of consecutive rows of stator and rotor blades in a turbomachinery.

Current second-order flow solvers tend to be relatively dissipative leading to strong damping of flow features. For example, in numerical simulations this results in a premature dissipation of vortices while in reality the vortices persist. On the contrary, higher-order methods can accurately track the vortices for a significantly longer time/distance. This is important also for the prediction of noise caused by the interaction of vortices with solid surfaces.

When a very high level of accuracy is needed (and enough computational resources are available), this kind of applications are analyzed employing the LES and sometimes DNS approach. High-order schemes are well-suited for these approaches, due to the improved capability of resolving gradients and wave numbers compared to standard second-order schemes [65]. High-order methods for the RANS equations are rarely used in real-life applications. One reason is that the turbulence model required in the RANS equations introduces a modeling error difficult to quantify; actually it might be greater than the discretization error

introduced by the spatial and time integration schemes. In that case, trying to minimize the discretization error with high-order schemes would not pay off since the modeling error would be the leading error term. However, one may argue that modeling error and discretization error should be considered separately, and that a fair comparison between models can only be performed if the discretization error is minimized. In practical terms, minimizing the discretization error translates into finding a so-called grid-converged solution. In this respect, it makes sense to look for more efficient schemes that, by definition, provide a solution with a given accuracy for less computational work. Another reason is stability: the equations governing the turbulence model introduce severe non-linearities that may affect the stability of the overall scheme. This problem is typically solved by adding artificial (numerical) dissipation that, however, undermines the accuracy of the scheme and in turn forces the user to increase the number of grid points. As will be seen later, there are high-order schemes that possess an inherent stability property that allows a reduction of the amount of numerical dissipation, which leads to an improvement of the accuracy of the result.

All these considerations indicate that high-order schemes might allow a significant reduction of mesh sizes, thus making feasible either larger-scale applications, or smaller turnaround times employing the same computing resources.

CFD has matured significantly over the past few decades. The research efforts carried out to achieve this have led to many discretization methods. In the next section we give a brief description of the most popular ones, with particular attention to their high-order versions. Several options exist, each with their specific strengths and weaknesses.

While the following discussion is limited to space discretization methods, it is noted that in this work high-order schemes will also be used for the discretization in time. In this context, all kinds of time integration schemes are available: implicit, explicit, low- and high-order⁴. In this work, a combination of all these schemes will be used and they are described in detail in section 3.3. In the present study, the discretization of the spatial part of the conservation equations is essentially decoupled from that of the temporal part (see section 3.1). Therefore, the time integration schemes are fairly general and may be used in combination with any of the spatial discretization methods described below.

1.2.1 Finite volume methods

The most popular discretization method for industrial aerodynamic applications is the finite volume method (see for instance [66]). The starting point of the finite volume method is the integral form of the conservation laws. The computational domain is discretized into cells, whose mean values constitute the degrees of freedom of the problem. In order to satisfy the conservation law, the fluxes through the

⁴ Another class of time integration schemes, called time-spectral schemes, exists which, however, we do not employ. Time-spectral methods are only suited for periodic problems and they are very efficient when only a few harmonic frequencies are important. The interested reader is referred to [123] for more details.

cells' boundaries have to be evaluated. The finite volume method is well-suited for conservation laws; it can be used for both structured and unstructured grids and therefore it is able to tackle flows around complex geometries. However, the extension to high order schemes is very complicated, especially in multiple space dimensions on unstructured meshes. The main reason is that the higher order versions of the method are achieved by employing a so-called reconstruction procedure. This reconstruction consists of the calculation of the values of the flow variables at the interface from the known cell-averaged values. This procedure determines the scheme's order of accuracy. Once the values at the interface between two adjacent cells are known, a Riemann problem must be solved at the interface and the flux through the common boundary can be calculated with so-called numerical flux functions. While a second-order accurate reconstruction can be achieved utilizing a piecewise linear interpolation from the cell-average values of the adjacent cells (direct neighbors), a third order reconstruction is cumbersome, specifically on unstructured meshes, as it requires more than one layer of neighboring cells. Additionally, higher order quadrature rules must be used. These difficulties limit the order of numerical computations in industrial applications to second order. Yet second order finite volume solvers have become the most popular workhorse of CFD industrial practitioners for many reasons: simplicity, efficiency, availability and the fact that lots of experience has been accumulated in the past three decades (shock capturing techniques, overset-grid methods, boundary conditions for sliding interfaces and internal flows are readily available). The main drawback of the method is the inherent high diffusivity, which is beneficial for stability but detrimental for accuracy.

1.2.2 Finite element methods

Another class of popular spatial discretization schemes is the finite element method. Finite element methods are based on the weak form of the conservation equations. The weak form is obtained by multiplying the differential equations by arbitrary test functions and then integrating by parts. The discrete solution is constructed as a linear combination of the so-called Ansatz functions, which typically are piecewise continuous polynomials. The choice of the test and Ansatz functions space determines the type of finite element method that is obtained. For instance, if test functions and Ansatz functions belong to the same class, the method is referred to as a Galerkin finite element method. Structured and unstructured grids can be handled, which makes this method applicable to complex geometries. Furthermore, two major classes of the schemes can be defined: continuous and discontinuous. In the former, the solution (i.e. the test and Ansatz functions) is assumed to be continuous across the boundary of the elements which constitute the triangulation of the domain. Thus the weak form must be satisfied over the entire domain. This class has become the method of choice for elliptic problems (structural mechanics) but, for hyperbolic problems, it loses part of its attractiveness as it needs stabilization terms, even for smooth solutions. In contrast, in the discontinuous version the continuity of the test functions is not required across the boundary of the elements,

and the weak form is satisfied on each element (the test functions have local compact support). The coupling between adjacent elements is accomplished via the boundary terms which are integrals over the element boundaries (the boundary terms arise from the integration by parts of the conservation equations). Similarly to the finite volume method, since at the interfaces between adjacent elements the solution is discontinuous, a Riemann problem must be solved and a numerical flux function is used⁵. However, no reconstruction procedure is needed, which makes the scheme very compact and thus easily parallelized. The order of the method (for smooth solutions) depends only on the degree of the approximating polynomials, which can be increased relatively easily and which can be changed for each element independently (p-refinement). The method also allows for local mesh refinement (h-refinement).

Thanks to these properties, the discontinuous Galerkin finite element method is receiving more and more attention by the academic community working on high-order methods. However, even though the discontinuous Galerkin method is well-suited for conservation laws, there are two main reasons why it is not yet widely used for complex aerodynamic simulations of compressible flows. Firstly, at least in its standard form [44], for high-order accuracy it becomes extremely expensive in three dimensions, since the degrees of freedom at the interfaces between elements are duplicated and high-order quadrature rules must be used. Secondly, it requires an accurate representation of the solid boundaries for high-order accuracy: the elements close to curved boundaries need to be curved as well. This is not straightforward for complex geometries and the vast majority of commercial grid generators uses piecewise linear approximations only. Typically these piecewise linear meshes are used to construct curved (high order) meshes [63].

1.2.3 Finite difference methods

Finite difference methods are based on the differential form of the conservation equations. The derivatives present in the differential equations are approximated with discrete, finite difference, operators. The application of the discrete operators results in algebraic equations whose unknowns, the degrees of freedom of the described problem, are the flow variables at the mesh points of the domain. Finite difference approximations of low and high order accuracy can be derived via Taylor expansion, see for instance [19]. The main strengths of the finite difference schemes are the following: they are relatively straightforward to program; they are well-suited for conservation laws and they are very efficient in terms of computational cost. The accuracy of the method is determined by the accuracy of the operator that approximates the spatial derivatives. High-order finite difference operators are relatively easy to construct (at least away from the boundaries of the computational domain) and they are obtained by just adding more grid points to the stencil. A major drawback of the method is that it is restricted to (multiblock)

⁵ In fact, the first order discontinuous Galerkin scheme is equivalent to the first order upwind finite volume scheme.

smooth structured grids. As a consequence, although overset grids may circumvent this problem (see [101] for an application), the finite difference method is not suited for very complex geometries as high effort has to be put into the decomposition of the computational domain in suitable blocks. In this respect, it must be noted that even today the generation of such block-structured grids is still a major bottleneck in the numerical simulations of real-life aeronautical applications (to a lesser extent, this applies also to unstructured grids), regardless of the discretization method employed [103, 63]. Since the finite difference method is the method of choice for the present thesis, it will be described in more detail in the next section.

1.3 SBP-SAT FINITE DIFFERENCE

Although it is easy to derive high-order finite difference schemes in the interior of the computational domain, it is non-trivial to find stable *and* accurate schemes close to its boundaries. Boundaries create two types of difficulties. Firstly, the stencils of the finite difference operators need to be modified. Secondly, boundary conditions have to be imposed. In this work we describe the development of a robust high-order CFD method that tries to overcome these issues by using Summation by Parts (SBP) finite difference operators and treatment of boundary conditions via Simultaneous Approximation Terms (SAT). As will be described in more detail in section 3.2, an SBP finite difference operator is essentially a (high-order) centered finite difference scheme with a specific closure at the boundary. SBP operators were first introduced in [60, 61] proposing approximate coefficients. Exact coefficients were derived only twenty years later, in 1994, [104]. In the same year, the first paper describing the use of SAT boundary terms was published [10]. The SAT are penalty-like terms that enforce the boundary conditions weakly and are used to augment the SBP schemes. They have been instrumental for the growth of the applicability of the SBP-SAT framework as the introduction of the SAT made it possible to derive stability proofs for smooth flow solutions. Thanks to this, the framework has been developed considerably during the last two decades. Consistent artificial dissipation operators were derived in [73]. Then, the foundations of the framework, which the present work builds upon, were laid in a series of papers [73, 81, 110, 114, 80], which present the derivation and the stability proofs of several boundary conditions for the Euler and Navier-Stokes equations. Since the stability proofs are obtained via the energy method (see section 3.2.1), these schemes are also called energy-stable. Interestingly, SBP-SAT schemes are not confined to the finite difference method. As shown in a recent review paper [115], the same ideas can be applied to the finite volume method and to the discontinuous Galerkin method; other applications, like wave propagation problems, can be tackled as well.

In the present research high-order SBP-SAT finite-difference schemes have been investigated and employed for the numerical simulation of a wide range of types of compressible flows. For the linearized equations governing a smooth flow,

these schemes are provably stable (and thus convergent) schemes. This is their main strength. In fact, thanks to the energy stability property of the SBP-SAT schemes, a significantly reduced amount of artificial dissipation is needed compared to schemes which do not possess this (or a similar) property; this leads to more accurate numerical solutions [115]. Furthermore, the proofs of convergence constitute a solid mathematical foundation that ensures that the discrete solution obtained in the numerical simulation is an approximation of the true mathematical solution. This makes it possible to distinguish between modeling errors and numerical errors. Additionally, compared to standard second order finite difference schemes, the high-order SBP-SAT schemes have improved capability of resolving wave numbers [65], which makes them well-suited for LES and DNS.

As any other class of discretization schemes, they are not free from drawbacks. They require multiblock structured grids, which gives both advantages and disadvantages. They are very efficient for relatively simple geometries because they require very limited connectivity information and the grid resolution can be easily tuned. Nevertheless, complex geometries are notoriously difficult to handle. However, this issue is less severe than for other methods since for the SBP-SAT schemes the mesh at the interface between blocks needs only to be C^0 continuous and no high-order representation of curved boundaries is needed. On the other hand, since the main assumption at the root of the design of these schemes is that the solution is smooth, they are not well-suited for resolving flows with shocks. As will be seen in section 3.2.3, not all boundary conditions available for finite-volume schemes are available for the SBP-SAT schemes yet.

1.4 MOTIVATION AND GOAL

In the field of turbulent flow simulations with high-order finite-difference SBP-SAT schemes, very little experience has been gained to date. Yet, given the favorable characteristics of these schemes, the potential gain is still substantial. So far these schemes have been confined either to academic test cases, or, when applied to realistic configurations, only the second order scheme has been used. In fact, in the papers describing their mathematical foundations, [110, 114, 80], only academic test cases have been investigated (for instance, the Euler vortex and the vortex shedding of a cylinder in cross-flow), and the emphasis has been on the accuracy of the schemes. More recently the SBP-SAT schemes have been applied to engineering aerodynamic applications: they have been used to predict the main aerodynamic coefficients of realistic geometries solving the RANS equations in [83, 85]. However, only second order solutions have been presented. Thus, a systematic comparison between the efficiency of low and high-order SBP-SAT schemes, especially when used to solve the NS and RANS equations, is lacking.

The main goal of the present work is to bridge this gap. The high-order SBP-SAT schemes are used to solve a wide range of types of compressible flow problems, ranging from inviscid flows to viscous (turbulent) flows. The intention of the work

is not the development of new physical flow models or new numerical schemes, but to assess the applicability of the SBP-SAT framework on classical flow models and engineering aerodynamic applications. The accuracy, the computational cost, the solution strategy and the robustness of the SBP-SAT schemes will be discussed in detail. The assessment of the efficiency of the method will be a focal point of the thesis.

1.5 OUTLINE OF THIS WORK

In chapter 2 the equations that model the motion of a viscous flow are introduced, with particular attention to the turbulence modeling. In chapter 3, the numerical solution of these equations is discussed: the high-order schemes used for the discretization of the equations are described together with the algorithms (the solution strategies) that have been implemented. Results for several test cases are presented in chapter 4. Among them we mention the classical test case of the Euler vortex, which is used to verify the design accuracy of the schemes, and engineering aerodynamic applications such as a multi-element airfoil in high-lift configuration and the transonic axial compressor Rotor37. Finally, chapter 5 presents the conclusions and future prospects.

2

THE MATHEMATICAL MODEL

This chapter introduces the set of conservation equations used as mathematical model for compressible flows. The purpose of this chapter is to show clearly which model is adopted, with particular attention to turbulent flows. The following discussion is restricted to non-reacting, mono-species, one-phase flows of a calorically perfect gas, for which the continuum hypothesis is valid, and the effect of body forces such as gravity as well as volumetric heat sources can be neglected.

A summary of the main equations described in this chapter can be found in appendix A.

2.1 CONSERVATION EQUATIONS

2.1.1 The Navier-Stokes equations

This section introduces the NS equations, the mathematical model of compressible viscous flows. This set of equation describes the conservation of mass, momentum and energy. The conservative form of the three-dimensional NS equations in index notation reads [48]:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_j)}{\partial x_j} = 0 \quad (2.1a)$$

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_j} \delta_{ij} + \frac{\partial \tau_{ij}}{\partial x_j} \quad (2.1b)$$

$$\frac{\partial(\rho E)}{\partial t} + \frac{\partial(\rho H u_j)}{\partial x_j} = \frac{\partial(\tau_{ij} u_i)}{\partial x_j} - \frac{\partial \dot{q}_j}{\partial x_j} \quad (2.1c)$$

or equivalently in a more compact form:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial}{\partial x_j} (\mathbf{F}_j^c - \mathbf{F}_j^v) = 0. \quad (2.2)$$

The vector $\mathbf{Q}$ represents the conserved variables, $\mathbf{F}^c$ and $\mathbf{F}^v$ are the inviscid (convective) and viscous flux vectors, respectively, i. e. :

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u_i \\ \rho E \end{bmatrix}, \quad \mathbf{F}_j^c = \begin{bmatrix} \rho u_j \\ \rho u_i u_j + p \delta_{ij} \\ \rho H u_j \end{bmatrix}, \quad \mathbf{F}_j^v = \begin{bmatrix} 0 \\ \tau_{ij} \\ u_i \tau_{ij} - \dot{q}_j \end{bmatrix} \quad (2.3)$$

Furthermore

$$E = e + \frac{1}{2} |\mathbf{u}|^2, \quad H = h + \frac{1}{2} |\mathbf{u}|^2, \quad h = e + \frac{p}{\rho} \quad (2.4)$$

where e represents the specific internal energy and h the specific enthalpy.

Note that, since the momentum equation (2.1b) has still the free index i , eq. (2.2) represents a system of 5 partial differential equations. However, the unknowns, which are $(\mathbf{Q}, p, \boldsymbol{\tau}, \dot{\mathbf{q}})$, represent more than five unknown quantities. Therefore additional equations are needed to close the system; these are the equation of state and the constitutive equations.

2.1.2 Equations of state

A perfect gas obeys the ideal gas law

$$p = \rho RT \quad (2.5)$$

with the gas constant equivalent to

$$R = c_p - c_v. \quad (2.6)$$

For a calorically perfect gas, the specific heats c_p and c_v are constant, i. e. :

$$e = c_v T, \quad h = c_p T \quad (2.7)$$

and therefore

$$p = (\gamma - 1)\rho e, \quad \gamma = \frac{c_p}{c_v} \quad (2.8)$$

$$T = \frac{\gamma - 1}{R} e, \quad h = \gamma e \quad (2.9)$$

2.1.3 Constitutive equations

The viscous stress tensor τ_{ij} ($\boldsymbol{\tau}$, in vector notation) and the heat flux vector $\dot{q}_i$ need to be defined.

Viscous stress

A common assumption is to model the air as a Newtonian fluid, for which the viscous stress and the strain rate S_{ij} are linearly related:

$$\tau_{ij} = 2\mu S_{ij} + \lambda S_{kk} \delta_{ij}, \quad S_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

Throughout this work we are going to employ the Stokes' assumption, for which $3\lambda + 2\mu = 0$, and therefore

$$\tau_{ij} = \mu \left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right]. \quad (2.10)$$

The molecular dynamic viscosity μ depends on the temperature and can be determined from the semi-empirical relation by Sutherland:

$$\mu(T) = \mu_0 \left(\frac{T}{T_0} \right)^{3/2} \frac{T_0 + S}{T + S} \quad (2.11)$$

where S is the Sutherland's constant, and μ_0 and T_0 are reference values. For air the following values are used:

$$\mu_0 = 1.716 \cdot 10^{-5} \text{ kg/(m s)}, \quad T_0 = 273.15 \text{ K}, \quad S = 110.55 \text{ K}. \quad (2.12)$$

Heat conduction

Fourier's law is used to model the heat flux vector $\dot{q}_i$ [53]:

$$\dot{q}_i = -\kappa \frac{\partial T}{\partial x_i} \quad (2.13)$$

where κ is the thermal conduction coefficient. It is convenient to rewrite $\dot{q}_i$ in terms of the specific internal energy e , making use of the Prandtl number, Pr , which is defined as

$$Pr = \frac{\mu/\rho}{\kappa/(\rho c_p)} = \frac{c_p \mu}{\kappa}. \quad (2.14)$$

Since $T = h/c_p = \gamma e/c_p$, and assuming Pr is constant, we can rewrite eq. (2.13) as:

$$\dot{q}_i = -\frac{\gamma \mu}{Pr} \frac{\partial e}{\partial x_i}. \quad (2.15)$$

Equation (2.14) indirectly expresses the dependence of the thermal conduction coefficient κ on the temperature via Sutherland's law. For air, $Pr = 0.72$.

2.1.4 Non-inertial frame of reference and Arbitrary Lagrangian-Eulerian formulations

For problems characterized by a constant rate of rotation (e.g. rotor applications), it is convenient to transform the conservation laws (2.2) from an inertial frame of reference to a non-inertial frame of reference rotating at constant angular velocity $\mathbf{\Omega}$. It is assumed that the computational domain is non-deformable and fixed to the rotating frame. With this transformation, a flow field that is unsteady when viewed from the inertial frame can be solved for as a steady flow problem, and thus more efficiently, in the non-inertial frame.

First define the rotational velocity $\mathbf{u}^{\text{rot}}$ as

$$\mathbf{u}^{\text{rot}} = \mathbf{u} - \mathbf{u}^{\text{rel}} = \mathbf{\Omega} \times \mathbf{r} \quad (2.16)$$

which is the velocity at which the non-inertial frame of reference is rotating. Here $\mathbf{r}$ is the position vector pointing from the rotation center $\mathbf{r}_0$ to a point $\mathbf{x}$ in the flow domain, $\mathbf{r} = \mathbf{x} - \mathbf{r}_0$. The vector $\mathbf{u}$ is again the absolute velocity, and $\mathbf{u}^{\text{rel}}$ is the relative velocity, i.e. the velocity seen by an observer moving with the rotating frame of reference. Then, the NS equations (eq. (2.2)) in a non-inertial frame of reference rotating at constant rotational speed, written in terms absolute quantities, become (see [21] for the full derivation):

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial}{\partial x_j} (\mathbf{F}^{\text{c,rot}} - \mathbf{F}_j^v) = \mathbf{S} \quad (2.17)$$

where $\mathbf{Q}$ and $\mathbf{F}^v$ are defined in (2.3), and $\mathbf{S}$ is the source term due to rotation:

$$\mathbf{S} = \begin{bmatrix} 0 \\ -\boldsymbol{\Omega} \times (\rho \mathbf{u}) \\ 0 \end{bmatrix}; \quad (2.18)$$

the flux vector $\mathbf{F}_j^{c, \text{rot}}$ reads:

$$\mathbf{F}_j^{c, \text{rot}} = \mathbf{F}_j^c - \mathbf{Q} \mathbf{u}_j^{\text{rot}} = \begin{bmatrix} \rho(u_j - u_j^{\text{rot}}) \\ \rho u_i(u_j - u_j^{\text{rot}}) + p \delta_{ij} \\ \rho H u_j - \rho E u_j^{\text{rot}} \end{bmatrix}. \quad (2.19)$$

The source terms $\mathbf{S}$ are the density multiplied by the cross product of the rotation rate vector $\boldsymbol{\Omega}$ and the absolute velocity $\mathbf{u}$; they are a combination of the centrifugal force and the Coriolis force. Since these forces are perpendicular to the direction of motion, they do not perform any work and thus their contribution to the energy equation is zero.

This solution approach cannot be applied to all simulations of rotating bodies. The flow field must be steady in the rotating frame, and some conditions or geometric features, such as relative surface motion (rotor/stator interactions), can cause unsteadiness for the flow around rotating bodies. Furthermore, many unsteady flows of interest, such as pitching, plunging, or rotating surfaces, require solutions on arbitrarily moving grids. For this reason, an Arbitrary Lagrangian-Eulerian (ALE) [25] formulation has also been implemented. In this formulation, the inviscid flux $\mathbf{F}^c$ appearing in (2.2) is modified as:

$$\mathbf{F}_j^{c, \text{ale}} = \begin{bmatrix} \rho(u_j - s_j) \\ \rho u_i(u_j - s_j) + p \delta_{ij} \\ \rho H u_j - \rho E s_j \end{bmatrix}, \quad (2.20)$$

where $\mathbf{s}$ is the mesh velocities vector, which in this work is computed analytically. The viscous fluxes $\mathbf{F}^v$ again remain the same as in (2.2) and (2.17) but, unlike in the formulation for the rotating frame of reference (2.17), source terms do not appear.

2.2 TRANSFORMATION TO CURVILINEAR COORDINATES

Let us rewrite the equations of motion (2.2) in a more detailed way; with $\mathbf{F}^c = (\mathbf{E}^c, \mathbf{F}^c, \mathbf{G}^c)^T$ and $\mathbf{F}^v = (\mathbf{E}^v, \mathbf{F}^v, \mathbf{G}^v)^T$, eq. (2.2) becomes:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}^c}{\partial x} + \frac{\partial \mathbf{F}^c}{\partial y} + \frac{\partial \mathbf{G}^c}{\partial z} = \frac{\partial \mathbf{E}^v}{\partial x} + \frac{\partial \mathbf{F}^v}{\partial y} + \frac{\partial \mathbf{G}^v}{\partial z} \quad (2.21)$$

where

$$\mathbf{Q} = [\rho, \rho u, \rho v, \rho w, \rho E]^T$$

Figure 2.1: Transformation from physical to computational space (reproduced from [93]).

and

$$\begin{aligned}
 \mathbf{E}^c &= \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho Hu \end{bmatrix}, & \mathbf{E}^v &= \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - \dot{q}_x \end{bmatrix}, \\
 \mathbf{F}^c &= \begin{bmatrix} \rho v \\ \rho v^2 + p \\ \rho v u \\ \rho v w \\ \rho Hv \end{bmatrix}, & \mathbf{F}^v &= \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{yx} + v\tau_{yy} + w\tau_{yz} - \dot{q}_y \end{bmatrix}, \\
 \mathbf{G}^c &= \begin{bmatrix} \rho w \\ \rho w^2 + p \\ \rho w u \\ \rho w v \\ \rho Hw \end{bmatrix}, & \mathbf{G}^v &= \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u\tau_{zx} + v\tau_{zy} + w\tau_{zz} - \dot{q}_z \end{bmatrix}.
 \end{aligned}$$

The NS equations (2.21) are now transformed from the physical space (t, x, y, z) to the arbitrary computational space (τ, ξ, η, ζ) , see fig. 2.1, by the following relations:

$$\tau = t \quad (2.22a)$$

$$\xi = \xi(t, x, y, z) \quad (2.22b)$$

$$\eta = \eta(t, x, y, z) \quad (2.22c)$$

$$\zeta = \zeta(t, x, y, z). \quad (2.22d)$$

2.2.1 Metric terms

Here it is convenient to introduce a compact notation for the derivative: $(\cdot)_x = \frac{\partial \cdot}{\partial x}$, and similarly for the other coordinates $y, z, t, \xi, \eta, \zeta, \tau$. Note that this notation

only applies to the metric terms defined below¹. With this notation, the Cartesian derivatives are to be expanded in (τ, ξ, η, ζ) space via chain-rule-of-differentiation relations such as:

$$\frac{\partial u}{\partial t} = \xi_\tau \frac{\partial u}{\partial \xi} + \eta_\tau \frac{\partial u}{\partial \eta} + \zeta_\tau \frac{\partial u}{\partial \zeta} + \frac{\partial u}{\partial \tau} \quad (2.23a)$$

$$\frac{\partial u}{\partial x} = \xi_x \frac{\partial u}{\partial \xi} + \eta_x \frac{\partial u}{\partial \eta} + \zeta_x \frac{\partial u}{\partial \zeta} \quad (2.23b)$$

$$\frac{\partial u}{\partial y} = \xi_y \frac{\partial u}{\partial \xi} + \eta_y \frac{\partial u}{\partial \eta} + \zeta_y \frac{\partial u}{\partial \zeta} \quad (2.23c)$$

$$\frac{\partial u}{\partial z} = \xi_z \frac{\partial u}{\partial \xi} + \eta_z \frac{\partial u}{\partial \eta} + \zeta_z \frac{\partial u}{\partial \zeta}. \quad (2.23d)$$

From equations (2.23) it is clear that, to compute Cartesian derivatives, the so-called metric terms $\xi_\tau, \xi_x, \xi_y, \xi_z, \eta_\tau, \eta_x, \eta_y, \eta_z, \zeta_\tau, \zeta_x, \zeta_y$, and ζ_z must be provided. Following closely [49], first consider the differential of the independent variables (t, x, y, z) :

$$dt = d\tau \quad (2.24)$$

$$dx = x_\tau d\tau + x_\xi d\xi + x_\eta d\eta + x_\zeta d\zeta \quad (2.25)$$

$$dy = y_\tau d\tau + y_\xi d\xi + y_\eta d\eta + y_\zeta d\zeta \quad (2.26)$$

$$dz = z_\tau d\tau + z_\xi d\xi + z_\eta d\eta + z_\zeta d\zeta \quad (2.27)$$

which can be rewritten in matrix form as:

$$\begin{bmatrix} dt \\ dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ x_\tau & x_\xi & x_\eta & x_\zeta \\ y_\tau & y_\xi & y_\eta & y_\zeta \\ z_\tau & z_\xi & z_\eta & z_\zeta \end{bmatrix} \begin{bmatrix} d\tau \\ d\xi \\ d\eta \\ d\zeta \end{bmatrix}. \quad (2.28)$$

Switching the role of the independent and dependent variables, similar relations can be obtained for the differentials of (τ, ξ, η, ζ) :

$$\begin{bmatrix} d\tau \\ d\xi \\ d\eta \\ d\zeta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \xi_t & \xi_x & \xi_y & \xi_z \\ \eta_t & \eta_x & \eta_y & \eta_z \\ \zeta_t & \zeta_x & \zeta_y & \zeta_z \end{bmatrix} \begin{bmatrix} dt \\ dx \\ dy \\ dz \end{bmatrix}. \quad (2.29)$$

The metric terms are then obtained by comparing eq. (2.28) and eq. (2.29), which leads to:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ \xi_t & \xi_x & \xi_y & \xi_z \\ \eta_t & \eta_x & \eta_y & \eta_z \\ \zeta_t & \zeta_x & \zeta_y & \zeta_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ x_\tau & x_\xi & x_\eta & x_\zeta \\ y_\tau & y_\xi & y_\eta & y_\zeta \\ z_\tau & z_\xi & z_\eta & z_\zeta \end{bmatrix}^{-1} \quad (2.30)$$

¹ For the sake of clarity, the compact notation for the derivatives applies only to the terms present in equations (2.31).

and hence finally

$$\begin{aligned}
\xi_x &= J(y_\eta z_\zeta - y_\zeta z_\eta) & \zeta_x &= J(y_\xi z_\eta - y_\eta z_\xi) \\
\xi_y &= J(x_\zeta z_\eta - x_\eta z_\zeta) & \zeta_y &= J(x_\eta z_\xi - x_\xi z_\eta) \\
\xi_z &= J(x_\eta y_\zeta - x_\zeta y_\eta) & \zeta_z &= J(x_\xi y_\eta - x_\eta y_\xi) \\
\eta_x &= J(y_\zeta z_\xi - y_\xi z_\zeta) & \xi_t &= -(x_\tau \xi_x + y_\tau \xi_y + z_\tau \xi_z) \\
\eta_y &= J(x_\xi z_\zeta - x_\zeta z_\xi) & \eta_t &= -(x_\tau \eta_x + y_\tau \eta_y + z_\tau \eta_z) \\
\eta_z &= J(x_\zeta y_\xi - x_\xi y_\zeta) & \zeta_t &= -(x_\tau \zeta_x + y_\tau \zeta_y + z_\tau \zeta_z)
\end{aligned} \tag{2.31}$$

where

$$\begin{aligned}
J^{-1} = \det \left[\frac{\partial(x, y, z)}{\partial(\xi, \eta, \zeta)} \right] &= +x_\xi(y_\eta z_\zeta - y_\zeta z_\eta) - x_\eta(y_\xi z_\zeta - y_\zeta z_\xi) \\
&\quad + x_\zeta(y_\xi z_\eta - y_\eta z_\xi).
\end{aligned} \tag{2.32}$$

The variable J is the determinant of the Jacobian of the transformation. The computational space (τ, ξ, η, ζ) is chosen such that the step-sizes $\Delta\xi, \Delta\eta, \Delta\zeta$ are uniformly spaced and equal to 1. Thus, the terms $x_\xi, x_\eta, x_\zeta, y_\xi$, etc., can be computed by simple finite difference approximations and the metric terms (2.31) are easily formed. In particular, the finite difference approximations that we use for the computation of the metric terms are the same as the ones we use for the discretization of the spatial derivatives of the NS equations. These finite difference schemes are described in section 3.2.

2.2.2 The transformed equations

Once the transformation is carried out, the equations can be recast in conservation form, see [49]. The end result, taking into account the possible source term (2.18), is:

$$\frac{\partial \tilde{\mathbf{Q}}}{\partial \tau} + \frac{\partial \tilde{\mathbf{E}}^c}{\partial \xi} + \frac{\partial \tilde{\mathbf{F}}^c}{\partial \eta} + \frac{\partial \tilde{\mathbf{G}}^c}{\partial \zeta} = \frac{\partial \tilde{\mathbf{E}}^v}{\partial \xi} + \frac{\partial \tilde{\mathbf{F}}^v}{\partial \eta} + \frac{\partial \tilde{\mathbf{G}}^v}{\partial \zeta} + \tilde{\mathbf{S}} \tag{2.33}$$

where

$$\tilde{\mathbf{Q}} = J^{-1} \mathbf{Q} \tag{2.34a}$$

$$\tilde{\mathbf{E}}^c = J^{-1} (\xi_x \mathbf{E}^c + \xi_y \mathbf{F}^c + \xi_z \mathbf{G}^c + \xi_t \mathbf{Q}) \tag{2.34b}$$

$$\tilde{\mathbf{F}}^c = J^{-1} (\eta_x \mathbf{E}^c + \eta_y \mathbf{F}^c + \eta_z \mathbf{G}^c + \eta_t \mathbf{Q}) \tag{2.34c}$$

$$\tilde{\mathbf{G}}^c = J^{-1} (\zeta_x \mathbf{E}^c + \zeta_y \mathbf{F}^c + \zeta_z \mathbf{G}^c + \zeta_t \mathbf{Q}) \tag{2.34d}$$

$$\tilde{\mathbf{E}}^v = J^{-1} (\xi_x \mathbf{E}^v + \xi_y \mathbf{F}^v + \xi_z \mathbf{G}^v) \tag{2.34e}$$

$$\tilde{\mathbf{F}}^v = J^{-1} (\eta_x \mathbf{E}^v + \eta_y \mathbf{F}^v + \eta_z \mathbf{G}^v) \tag{2.34f}$$

$$\tilde{\mathbf{G}}^v = J^{-1} (\zeta_x \mathbf{E}^v + \zeta_y \mathbf{F}^v + \zeta_z \mathbf{G}^v) \tag{2.34g}$$

$$\tilde{\mathbf{S}} = J^{-1} \mathbf{S} \tag{2.34h}$$

The terms $\xi_t \mathbf{Q}$, $\eta_t \mathbf{Q}$ and $\zeta_t \mathbf{Q}$ can be incorporated in the inviscid fluxes using the definition of ξ_t, η_t and ζ_t given in (2.31), and the mesh velocities vector $\mathbf{s}$:

$$\mathbf{s} = (x_\tau, y_\tau, z_\tau)^T = (s_x, s_y, s_z)^T \quad (2.35)$$

Note that if the mesh is rotating with constant angular velocity $\boldsymbol{\Omega}$, then, from equation (2.16), $\mathbf{s} = \mathbf{u}^{\text{rot}}$.

For example, the ξ -flux $\tilde{\mathbf{E}}^c$ can be rewritten as:

$$\tilde{\mathbf{E}}^c = \mathbf{J}^{-1} \begin{bmatrix} \rho[\xi_x(u - s_x) + \xi_y(v - s_y) + \xi_z(w - s_z)] \\ \rho u[\xi_x(u - s_x) + \xi_y(v - s_y) + \xi_z(w - s_z)] + p\xi_x \\ \rho v[\xi_x(u - s_x) + \xi_y(v - s_y) + \xi_z(w - s_z)] + p\xi_y \\ \rho w[\xi_x(u - s_x) + \xi_y(v - s_y) + \xi_z(w - s_z)] + p\xi_z \\ \rho H[\xi_x(u - s_x) + \xi_y(v - s_y) + \xi_z(w - s_z)] - p\xi_\tau \end{bmatrix}$$

Now define the contravariant velocities (U, V, W) as:

$$\begin{aligned} U &= \xi_\tau + \xi_x u + \xi_y v + \xi_z w = \\ &= \xi_x(u - s_x) + \xi_y(v - s_y) + \xi_z(w - s_z); \end{aligned} \quad (2.36a)$$

$$\begin{aligned} V &= \eta_\tau + \eta_x u + \eta_y v + \eta_z w = \\ &= \eta_x(u - s_x) + \eta_y(v - s_y) + \eta_z(w - s_z); \end{aligned} \quad (2.36b)$$

$$\begin{aligned} W &= \zeta_\tau + \zeta_x u + \zeta_y v + \zeta_z w = \\ &= \zeta_x(u - s_x) + \zeta_y(v - s_y) + \zeta_z(w - s_z); \end{aligned} \quad (2.36c)$$

where the terms $(u - s_x)$, $(v - s_y)$ and $(w - s_z)$ are the relative Cartesian velocity components u^{rel} , v^{rel} and w^{rel} , respectively. The inviscid curvilinear ξ -flux becomes:

$$\tilde{\mathbf{E}}^c = \mathbf{J}^{-1} \begin{bmatrix} \rho U \\ \rho u U + p\xi_x \\ \rho v U + p\xi_y \\ \rho w U + p\xi_z \\ \rho H U - p\xi_\tau \end{bmatrix} \quad (2.37)$$

and similarly for the η and ζ direction:

$$\tilde{\mathbf{F}}^c = \mathbf{J}^{-1} \begin{bmatrix} \rho V \\ \rho u V + p\eta_x \\ \rho v V + p\eta_y \\ \rho w V + p\eta_z \\ \rho H V - p\eta_\tau \end{bmatrix}, \quad \tilde{\mathbf{G}}^c = \mathbf{J}^{-1} \begin{bmatrix} \rho W \\ \rho u W + p\zeta_x \\ \rho v W + p\zeta_y \\ \rho w W + p\zeta_z \\ \rho H W - p\zeta_\tau \end{bmatrix}. \quad (2.38)$$

2.2.3 Free-stream preservation and Geometric Conservation Law

In deriving the strong conservation form of the flow equations, (2.33), the following metric identities have been implicitly invoked:

$$\left(\frac{1}{J}\right)_\tau + \left(\frac{\xi_t}{J}\right)_\xi + \left(\frac{\eta_t}{J}\right)_\eta + \left(\frac{\zeta_t}{J}\right)_\zeta = 0 \quad (2.39a)$$

$$\left(\frac{\xi_x}{J}\right)_\xi + \left(\frac{\eta_x}{J}\right)_\eta + \left(\frac{\zeta_x}{J}\right)_\zeta = 0 \quad (2.39b)$$

$$\left(\frac{\xi_y}{J}\right)_\xi + \left(\frac{\eta_y}{J}\right)_\eta + \left(\frac{\zeta_y}{J}\right)_\zeta = 0 \quad (2.39c)$$

$$\left(\frac{\xi_z}{J}\right)_\xi + \left(\frac{\eta_z}{J}\right)_\eta + \left(\frac{\zeta_z}{J}\right)_\zeta = 0 \quad (2.39d)$$

The first relation, (2.39a), is referred to in literature as the Geometric Conservation Law (GCL) [118]. Relations (2.39) must be verified numerically in order to guarantee free-stream preservation.

Obviously, the GCL identity (2.39a) is fulfilled for non-moving meshes. For moving (and/or deforming) meshes, however, it can be satisfied by first splitting the first term of (2.33) as follows:

$$\frac{\partial \tilde{\mathbf{Q}}}{\partial \tau} = \frac{\partial J^{-1} \mathbf{Q}}{\partial \tau} = J^{-1} \frac{\partial \mathbf{Q}}{\partial \tau} + \mathbf{Q} \frac{\partial J^{-1}}{\partial \tau} \quad (2.40)$$

and then, as suggested in [129], by invoking the GCL identity (2.39a) to evaluate the last term of (2.40):

$$\mathbf{Q} \frac{\partial J^{-1}}{\partial \tau} = \mathbf{Q} \left(\frac{1}{J}\right)_\tau = -\mathbf{Q} \left[\left(\frac{\xi_t}{J}\right)_\xi + \left(\frac{\eta_t}{J}\right)_\eta + \left(\frac{\zeta_t}{J}\right)_\zeta \right]. \quad (2.41)$$

The term $\mathbf{Q} \frac{\partial J^{-1}}{\partial \tau}$ is then another source term to be added to the residual evaluation in case of a moving grid.

In [118], it was proposed to rewrite equations (2.31) in an equivalent "conservative" form:

$$\xi_x = J[(y_\eta z)_\zeta - (y_\zeta z)_\eta] = J[(yz_\zeta)_\eta - (yz_\eta)_\zeta] \quad (2.42a)$$

$$\eta_x = J[(y_\zeta z)_\xi - (y_\xi z)_\zeta] = J[(yz_\xi)_\zeta - (yz_\zeta)_\xi] \quad (2.42b)$$

$$\zeta_x = J[(y_\xi z)_\eta - (y_\eta z)_\xi] = J[(yz_\eta)_\xi - (yz_\xi)_\eta] \quad (2.42c)$$

and similarly for the remaining metric terms. Note that there are two options, which are equally valid. Since there is no reason to prefer one over the other, the metric terms in this so-called conservative formulation can be obtained by averaging the two options:

$$\xi_x = J[(y_\eta z)_\zeta - (y_\zeta z)_\eta + (yz_\zeta)_\eta - (yz_\eta)_\zeta]/2 \quad (2.43a)$$

$$\eta_x = J[(y_\zeta z)_\xi - (y_\xi z)_\zeta + (yz_\xi)_\zeta - (yz_\zeta)_\xi]/2 \quad (2.43b)$$

$$\zeta_x = J[(y_\xi z)_\eta - (y_\eta z)_\xi + (yz_\eta)_\xi - (yz_\xi)_\eta]/2. \quad (2.43c)$$

As shown in [129], the metric terms calculated as in (2.43) ensure proper cancellation of the metric terms and hence free-stream preservation for 3D problems. For 2D problems, the free-stream preserving metrics (2.43) are not necessary and the standard metric terms (2.31) can be used.

2.3 TURBULENCE MODELING

The time-dependent NS equations (2.2) contain all the physics that is needed to obtain the solution of any type of flow, given appropriate initial and boundary conditions. The numerical solution of these equations, a so-called DNS, gives -in principle- a numerically-accurate solution of the exact equations of motions and are therefore of great value. However, the DNS of turbulence requires huge computational costs and memory usage, even at low Reynolds number. For flows with Reynolds numbers $Re = \mathcal{O}(10^{5-6})$ or higher, the computational resources required by a DNS exceed by far the capacity of the currently available supercomputers [90]. In order to reduce the prohibitive computational costs of resolving all the turbulent time-dependent structures, the effects of turbulence on the flow need to be modeled. In this work two different modeling approaches will be used, namely the approach of the URANS equations and the approach of the LES.

2.3.1 URANS

The derivation of the URANS equations (also termed the Favre-averaged Navier-Stokes equations) starts with the decomposition of the various flow variables in a mean and a fluctuating part. Two different decompositions have to be introduced, the Reynolds decomposition and the Favre decomposition, which are defined as:

$$\phi(\mathbf{x}, t) = \underbrace{\overline{\phi}(\mathbf{x}, t) + \phi'(\mathbf{x}, t)}_{\text{Reynolds decomp.}} = \underbrace{\hat{\phi}(\mathbf{x}, t) + \phi''(\mathbf{x}, t)}_{\text{Favre decomp.}}. \quad (2.44)$$

Here the overbar indicates a conventional time-average mean, with the averaging time scale assumed to be long compared to that of the turbulent fluctuations, and short compared to the time scale of the unsteadiness of the mean flow. The hat represents the Favre (density-weighted) average: $\hat{\phi} = \overline{\rho\phi}/\bar{\rho}$. Note that the mean values $\overline{\phi}$ and $\hat{\phi}$ are still dependent on time, but their characteristic time scales are much larger than those of the fluctuations ϕ' and ϕ'' .

Next apply the Reynolds decomposition to p , ρ and $\dot{q}_j$, and the Favre decomposition to u_i , h , e and T ; then, upon substitution in the NS equations and time-averaging, the conservation equations for the mean flow variables are obtained

with additional unknown terms due to the nonlinearity of the equations. The resulting conservation equation can be written as follows (see for example [133, 90]):

$$\begin{aligned} \frac{\partial \bar{p}}{\partial t} + \frac{\partial(\bar{p}\hat{u}_j)}{\partial x_j} &= 0 \\ \frac{\partial(\bar{p}\hat{u}_i)}{\partial t} + \frac{\partial(\bar{p}\hat{u}_i\hat{u}_j)}{\partial x_j} &= -\frac{\partial \bar{p}}{\partial x_j}\delta_{ij} + \frac{\partial(\bar{\tau}_{ij} - \overline{\rho u_j''u_i''})}{\partial x_j} \\ \frac{\partial(\bar{p}\hat{E})}{\partial t} + \frac{\partial(\bar{p}\hat{H}\hat{u}_j)}{\partial x_j} &= \frac{\partial(\bar{\tau}_{ij} - \overline{\rho u_j''u_i''})\hat{u}_i}{\partial x_j} \\ &\quad - \frac{\partial(\bar{q}_j + c_p\overline{\rho u_j''T''} - \bar{\tau}_{ij}u_i'' + \frac{1}{2}\overline{\rho u_i''u_i''u_j''})}{\partial x_j}. \end{aligned}$$

The equation of state becomes:

$$\bar{p} = (\gamma - 1)[\bar{p}\hat{E} - \frac{1}{2}\bar{p}(\hat{u}^2 + \hat{v}^2 + \hat{w}^2) - \bar{p}k] = \bar{p}R\hat{T}$$

where $\bar{p}k$ is kinetic energy per unit volume of the turbulent fluctuations defined as: $\bar{p}k = \overline{\rho u_i''u_i''}/2$. The quantities $\hat{E}$ and $\hat{H}$ are the total energy and total enthalpy per unit mass, and include the turbulent kinetic energy, i.e. :

$$\hat{E} = \hat{e} + \frac{1}{2}\hat{u}_i\hat{u}_i + k, \quad \hat{H} = \hat{h} + \frac{1}{2}\hat{u}_i\hat{u}_i + k. \quad (2.45)$$

The additional terms $\overline{\rho u_j''u_i''}$, $c_p\overline{\rho u_j''T''}$, $\bar{\tau}_{ij}u_i''$ and $\frac{1}{2}\overline{\rho u_i''u_i''u_j''}$ describe the influence of the fluctuations on the mean flow quantities and they need to be modeled in order to close the flow model.

Most of the turbulence models focus on the first of the indicated additional terms, $\overline{\rho u_j''u_i''}$. This term is traditionally called the Reynolds stress tensor τ_{ij}^t and is defined as:

$$\tau_{ij}^t = -\overline{\rho u_j''u_i''}. \quad (2.46)$$

The trace of the Reynolds stress tensor is proportional to the turbulent kinetic energy as $\tau_{ii}^t = -\overline{\rho u_i''u_i''} = -2\bar{p}k$. In this work we employ the Spalart-Allmaras (SA) model, which is explained in section 2.3.1.1. Typically less attention is given to the other fluctuating terms. The common practice, adopted here, is to model the turbulent heat flux using a Reynolds analogy:

$$\bar{q}_j^t = c_p\overline{\rho u_j''T''} \approx -\frac{c_p\mu_t}{Pr_t} \frac{\partial \hat{T}}{\partial x_j} \quad (2.47)$$

where μ_t is the eddy viscosity obtained by the turbulence model, and Pr_t is the turbulent Prandtl number, assumed constant and for air equal to 0.9. Finally, the remaining fluctuating terms are modeled together as:

$$\bar{\tau}_{ij}u_i'' - \frac{1}{2}\overline{\rho u_i''u_i''u_j''} \approx \left(\mu + \frac{\mu_t}{\sigma_k}\right) \frac{\partial k}{\partial x_j}. \quad (2.48)$$

where σ_k is a coefficient related to the modeling equation for k .

Incorporating these last relations in the conservation equations we obtain the final system of URANS equations in Cartesian coordinates:

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial(\bar{\rho}\hat{u}_j)}{\partial x_j} = 0 \quad (2.49a)$$

$$\frac{\partial(\bar{\rho}\hat{u}_i)}{\partial t} + \frac{\partial(\bar{\rho}\hat{u}_i\hat{u}_j)}{\partial x_j} = -\frac{\partial \bar{p}}{\partial x_j}\delta_{ij} + \frac{\partial \tau_{ij}^{eff}}{\partial x_j} \quad (2.49b)$$

$$\frac{\partial(\bar{\rho}\hat{E})}{\partial t} + \frac{\partial(\bar{\rho}\hat{h}\hat{u}_j)}{\partial x_j} = \frac{\partial \tau_{ij}^{eff}\hat{u}_i}{\partial x_j} - \frac{\partial \dot{q}_j^{eff}}{\partial x_j} + \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\mu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right] \quad (2.49c)$$

where

$$\tau_{ij}^{eff} = \bar{\tau}_{ij} + \tau_{ij}^t \quad (2.50)$$

$$\dot{q}_j^{eff} = \bar{q}_j + \dot{q}_j^t. \quad (2.51)$$

NOTE From now on, for readability reasons, the hat and overbar notations of the Favre- and Reynolds averaged variables are dropped. It is understood that in the context of URANS the symbols $(\rho, u_i, p, h, e, T, \dot{q}_j)$ indicate the mean flow variables $(\bar{\rho}, \hat{u}_i, \bar{p}, \hat{h}, \hat{e}, \hat{T}, \bar{q}_j)$.

As for the NS equations, the expressions in curvilinear coordinates of the URANS equations can be obtained using the method described in section 2.2.

2.3.1.1 The Spalart-Allmaras model

The SA model is a linear eddy-viscosity model, for which the Reynolds stresses τ_{ij}^t are evaluated using the Boussinesq assumption (note the analogy with eq. (2.10)):

$$\tau_{ij}^t = \mu_t \left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right] - \frac{2}{3} \rho k \delta_{ij}. \quad (2.52)$$

The last term in (2.52) guarantees that the trace of τ^t is $-2\rho k$. However, that term is ignored for SA because the turbulent kinetic energy k is not readily available. For the same reason, all terms of the URANS equations (2.49) that involve k are neglected when using this turbulence model. The version of the model employed in this work is based on the *conservative, negative* version of the model proposed in [4] (also available in [97]), modified with a few simplifications explained below.

The turbulent eddy viscosity μ_t is given by:

$$\mu_t = \rho \nu_t = \rho \tilde{\nu} f_{\nu 1}, \quad f_{\nu 1} = \frac{\chi^3}{\chi^3 + c_{\nu 1}^3}, \quad \chi = \frac{\tilde{\nu}}{\nu} \quad (2.53)$$

where ν is the kinematic viscosity. $\tilde{\nu}$ is the SA working variable and obeys the following transport equation, written in conservative formulation [4]:

$$\begin{aligned} \frac{\partial \rho \tilde{\nu}}{\partial t} + \frac{\partial \rho \tilde{\nu} u_j}{\partial x_j} = & \rho(P - D + T) + \frac{1}{\sigma} \frac{\partial}{\partial x_j} \left[\rho(\nu + \tilde{\nu}) \frac{\partial \tilde{\nu}}{\partial x_j} \right] \\ & + \frac{c_{b2}}{\sigma} \rho \left(\frac{\partial \tilde{\nu}}{\partial x_j} \right)^2 - \frac{1}{\sigma} (\nu + \tilde{\nu}) \frac{\partial \rho}{\partial x_j} \frac{\partial \tilde{\nu}}{\partial x_j} \end{aligned} \quad (2.54)$$

The production, wall destruction and trip terms are

$$P = c_{b1}(1 - f_{t2})\tilde{S}\tilde{\nu}, \quad D = \left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2} \right) \left[\frac{\tilde{\nu}}{d} \right]^2, \quad T = f_{t1} \left(\frac{\partial}{\partial x_j} \frac{\partial u}{\partial x_j} \right)^2, \quad (2.55)$$

respectively. The trip term is shown here for the sake of completeness, however, it is actually neglected throughout this work. The $\tilde{S}$ in the production term is the modified vorticity

$$\tilde{S} = \Omega + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}, \quad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}}, \quad (2.56)$$

where Ω is the vorticity magnitude,

$$\Omega = \sqrt{2\Omega_{ij}\Omega_{ij}}, \quad \Omega_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i} \right), \quad (2.57)$$

κ is the Von Kármán constant and d is the distance to the nearest wall. The laminar suppression term f_{t2} is evaluated as:

$$f_{t2} = c_{t3} \exp(-c_{t4}\chi^2), \quad (2.58)$$

and the near-wall destruction term is given by the following relations:

$$f_w = g \left[\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6} \right]^{1/6}, \quad g = r + c_{w2}(r^6 - r), \quad r = \min \left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, r_{lim} \right). \quad (2.59)$$

The constants of the model are given in table 2.1.

To avoid possible numerical problems, the term $\tilde{S}$ must never be allowed to reach zero or go negative. The approach adopted here, which avoids the often used simple clipping, is the one suggested in [4]:

$$\tilde{S} = \begin{cases} \Omega + \bar{S} & \text{if } \bar{S} \geq -c_2\Omega \\ \Omega + \frac{\Omega(c_2^2\Omega + c_3\bar{S})}{(c_3 - 2c_2)\Omega - \bar{S}} & \text{if } \bar{S} < -c_2\Omega \end{cases} \quad \text{with } \bar{S} = \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}. \quad (2.60)$$

The negative version of the model was developed primarily to address issues with under-resolved grids and non-physical transient states reached during the solution process. The model is the same as the "standard" version when the turbulence variable $\tilde{\nu}$ is positive, but when $\tilde{\nu}$ is negative the production, destruction and diffusion

Table 2.1: Constants of the SA model.

Constant	Value
c_{b1}	0.1355
σ	2/3
c_{b2}	0.622
κ	0.41
c_{w1}	$c_{b1}/\kappa^2 + (1 + c_{b2})/\sigma$
c_{w2}	0.3
c_{w3}	2
c_{v1}	7.1
c_{t1}	1
c_{t2}	2
c_{t3}	1.2
c_{t4}	0.5
r_{lim}	10
c_2	0.7
c_3	0.9
c_{n1}	16

terms are modified. The resulting turbulent eddy viscosity (μ_t) is set to zero when $\tilde{v}$ is negative ($\tilde{v}$ itself becomes a passive scalar).

Putting together the conservative formulation (2.54) and the negative version proposed in [4], the actual implementation of the SA model in Cartesian coordinates used in this work reads:

$$\begin{aligned}
\frac{\partial \rho \tilde{v}}{\partial t} + \frac{\partial \rho \tilde{v} u_j}{\partial x_j} = & \rho(P - D) \\
& + \frac{1}{\sigma} \frac{\partial}{\partial x_j} \left[(v + (f_n + c_{b2})\tilde{v}) \frac{\partial \rho \tilde{v}}{\partial x_j} \right] \\
& - \frac{c_{b2}}{\sigma} \tilde{v} \left(\frac{\partial}{\partial x_j} \frac{\partial \rho \tilde{v}}{\partial x_j} \right)
\end{aligned} \tag{2.61}$$

with

$$\mu_t = \begin{cases} \rho \tilde{v} f_{v1} & \text{if } \tilde{v} \geq 0 \\ 0 & \text{if } \tilde{v} < 0, \end{cases} \tag{2.62}$$

$$P = \begin{cases} c_{b1}(1 - f_{t2})\tilde{S}\tilde{v} & \text{if } \tilde{v} \geq 0 \\ c_{b1}(1 - c_{t3})\Omega\tilde{v} & \text{if } \tilde{v} < 0, \end{cases} \tag{2.63}$$

$$D = \begin{cases} \left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2} \right) \left[\frac{\tilde{v}}{d} \right]^2 & \text{if } \tilde{v} \geq 0 \\ -c_{w1} \left[\frac{\tilde{v}}{d} \right]^2 & \text{if } \tilde{v} < 0 \end{cases} \tag{2.64}$$

and

$$f_n = \begin{cases} 1 & \text{if } \tilde{v} \geq 0 \\ \frac{c_{n1} + \chi^3}{c_{n1} - \chi^3} & \text{if } \tilde{v} < 0. \end{cases} \quad (2.65)$$

Note that the non-conservative diffusion term $\frac{c_{b2}}{\sigma} \rho \left(\frac{\partial \tilde{v}}{\partial x_j} \right)^2$ has been rewritten as:

$$\frac{c_{b2}}{\sigma} \rho \left(\frac{\partial \tilde{v}}{\partial x_j} \right)^2 = \frac{c_{b2}}{\sigma} \rho \left[\frac{\partial}{\partial x_j} \left(\tilde{v} \frac{\partial \tilde{v}}{\partial x_j} \right) - \tilde{v} \frac{\partial}{\partial x_j} \frac{\partial \tilde{v}}{\partial x_j} \right]. \quad (2.66)$$

Additionally, all the diffusion terms have been expressed in terms of $\frac{\partial \rho \tilde{v}}{\partial x_j}$ rather than $\rho \frac{\partial \tilde{v}}{\partial x_j}$, and all the (resulting) terms containing the gradient of density are then neglected. Lastly, the trip term T is ignored.

The transformation to curvilinear coordinates of eq. (2.61) is analogue to the one described for the flow equations in section 2.2. Referring to eq. (2.33) and eq. (2.34), the conserved variable, the Cartesian fluxes and source term of the SA model are the following:

$$Q = \rho \tilde{v}, \quad E^c = u \rho \tilde{v}, \quad F^c = v \rho \tilde{v}, \quad G^c = w \rho \tilde{v}, \quad (2.67a)$$

$$E^v = \frac{1}{\sigma} (v + (f_n + c_{b2}) \tilde{v}) \frac{\partial \rho \tilde{v}}{\partial x}, \quad (2.67b)$$

$$F^v = \frac{1}{\sigma} (v + (f_n + c_{b2}) \tilde{v}) \frac{\partial \rho \tilde{v}}{\partial y}, \quad (2.67c)$$

$$G^v = \frac{1}{\sigma} (v + (f_n + c_{b2}) \tilde{v}) \frac{\partial \rho \tilde{v}}{\partial z}, \quad (2.67d)$$

$$S = \rho(P - D) - L, \quad (2.67e)$$

where $L = \frac{c_{b2}}{\sigma} \tilde{v} \left(\frac{\partial}{\partial x_j} \frac{\partial \rho \tilde{v}}{\partial x_j} \right)$. The Laplacian term L is non-conservative and will be treated formally as another source term. It may be written in curvilinear coordinates as [68]:

$$L = \frac{c_{b2}}{\sigma} \tilde{v} \left(\frac{\partial}{\partial x_j} \frac{\partial \rho \tilde{v}}{\partial x_j} \right) = \frac{c_{b2}}{\sigma} \tilde{v} \left[J \frac{\partial}{\partial \xi_j} \left(\frac{1}{J} \frac{\partial \xi_j}{\partial x_i} \frac{\partial \xi_m}{\partial x_i} \frac{\partial \rho \tilde{v}}{\partial \xi_m} \right) \right], \quad i, j, m = 1, 2, 3. \quad (2.68)$$

The appropriate boundary conditions will be discussed in section 2.4.

2.3.2 LES

The LES approach consists in performing a computation in which the large eddies of the turbulence are resolved and the effects of the the smallest eddies are modeled. The formal derivation of the conservative equation involves the application of a spatial filter to the NS equations such that only the large-scale motions are resolved, while the effect of the small scales, also called sub-grid scales, on the resolved ones is modeled. The underlying premise is that the small scale turbulence is more

nearly isotropic, has a more universal character, and therefore it is more adapt for being modeled.

A filtering and a mass-weighted Favre filtering are introduced according to ([33]):

$$\bar{\phi}(\mathbf{x}, t) = \int_V \phi(\mathbf{x}', t) G(\mathbf{x}') d\mathbf{x}', \quad \bar{\rho}\hat{\phi}(\mathbf{x}, t) = \int_V \rho\phi(\mathbf{x}', t) G(\mathbf{x}') d\mathbf{x}' \quad (2.69)$$

where $G(\mathbf{x}')$ is the filter, which does not depend on time. Note that similar notations ($\bar{\phi}$ and $\hat{\phi}$) are used in both URANS and LES. As will be clearer later, this is done to show the similarity of the equations of the two approaches from an implementation point of view (see also appendix A). Another similarity is that also in LES there exists a decomposition $\phi(\mathbf{x}, t) = \hat{\phi}(\mathbf{x}, t) + \phi''(\mathbf{x}, t)$, where $\hat{\phi}$ denotes the filtered, resolved, quantity and ϕ'' denotes the unresolved part, the subgrid scale part. However, we remind that in the context of URANS $\bar{\phi}$ and $\hat{\phi}$ represent *time*-averages, while in LES they represent (weighted) *volume*-averages. This has important consequences regarding the grid and time resolution requirements of the two approaches.

The conservation equations for LES are obtained by filtering the instantaneous conservation equations, i.e. the NS equations. This process gives rise to similar closure requirements as encountered for URANS. The LES equations used in this work are cast in a form similar to (2.49) and read ([50]):

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial(\bar{\rho}\hat{u}_j)}{\partial x_j} = 0 \quad (2.70a)$$

$$\frac{\partial(\bar{\rho}\hat{u}_i)}{\partial t} + \frac{\partial(\bar{\rho}\hat{u}_i\hat{u}_j)}{\partial x_j} = -\frac{\partial \bar{p}}{\partial x_j} \delta_{ij} + \frac{\partial \tau_{ij}^{eff}}{\partial x_j} \quad (2.70b)$$

$$\frac{\partial(\bar{\rho}\hat{E})}{\partial t} + \frac{\partial(\bar{\rho}\hat{H}\hat{u}_j)}{\partial x_j} = \frac{\partial \tau_{ij}^{eff}\hat{u}_i}{\partial x_j} - \frac{\partial \dot{q}_j^{eff}}{\partial x_j} \quad (2.70c)$$

where

$$\tau_{ij}^{eff} = \bar{\tau}_{ij} + \tau_{ij}^t = (\mu + \mu_t) \left[\left(\frac{\partial \hat{u}_i}{\partial x_j} + \frac{\partial \hat{u}_j}{\partial x_i} \right) - \frac{2}{3} \frac{\partial \hat{u}_k}{\partial x_k} \delta_{ij} \right] \quad (2.71)$$

$$\dot{q}_j^{eff} = \bar{q}_j + \dot{q}_j^t = - \left(\frac{c_p \mu}{Pr} + \frac{c_p \mu_t}{Pr_t} \right) \frac{\partial \hat{T}}{\partial x_j}. \quad (2.72)$$

Again, the Boussinesq assumption has been used to model the Reynolds stresses and a gradient assumption has been used for the subgrid scale heat flux. Finally, a subgrid scale model for μ_t has to be chosen. Here we use the Implicit Large Eddy Simulation (ILES) model and the Wall-Adapting Local Eddy-viscosity (WALE) model.

2.3.2.1 ILES

ILES is based on the hypothesis that the action of subgrid scales on the resolved scales is equivalent to a strictly dissipative action [33]. No explicit modeling of the

subgrid scales is adopted ($\mu_t = 0$). Instead, their dissipative nature is accounted for by the artificial dissipation of the numerical scheme (see section 3.2.4). In other words, the artificial dissipation filters the subgrid scales, thus the filter is implicitly defined by the numerical scheme. In practical terms, the ILES approach adopted here is equivalent to an under-resolved DNS.

2.3.2.2 The WALE model

In the WALE model [78] the subgrid viscosity is defined as:

$$\mu_t = \mu_{sgs} = \hat{\rho}(C_w \Delta)^2 \frac{(s_{ij}^d s_{ij}^d)^{3/2}}{(\hat{S}_{ij} \hat{S}_{ij})^{5/2} + (s_{ij}^d s_{ij}^d)^{5/4}} \quad (2.73)$$

with $C_w = 0.325$, $\Delta = J^{-1/3}$ represents the local length scale,

$$s_{ij}^d = \frac{1}{2} (\hat{g}_{ij}^2 + \hat{g}_{ji}^2) - \frac{1}{3} \hat{g}_{kk}^2 \delta_{ij}, \quad \hat{g}_{ij} = \frac{\partial \hat{u}_i}{\partial x_j}, \quad (2.74)$$

and $\hat{S}_{ij}$ represents the strain rate tensor for the resolved scale defined by

$$\hat{S}_{ij} = \frac{1}{2} \left(\frac{\partial \hat{u}_i}{\partial x_j} + \frac{\partial \hat{u}_j}{\partial x_i} \right) \quad (2.75)$$

This model guarantees the correct asymptotic behavior $\nu_{sgs} \propto y^{+3}$, with y^+ the distance to the wall expressed in wall units.

2.4 BOUNDARY CONDITIONS

The mathematical problem consisting of the conservation equations described in this chapter would not be well-posed without proper initial and boundary conditions. Unless a better approximation of the solution is available, the initial condition will be a uniform field with conserved variables equal to the ones of the undisturbed far-field state. Boundary conditions require more attention. They can be divided into two categories: physical and artificial boundary conditions. As the names suggest, the former arise from physical considerations, while the latter arise from numerical considerations and are typically needed to obtain a finite computational domain. However, regardless of the category of the type of boundary, the number of boundary conditions to be specified has to be such that the problem is well-posed. The number of allowed boundary conditions depends on the local *characteristics* of the flow, see for example [81], and is reported in table 2.2. The practical implementation of the different boundary conditions strongly depends on the numerical treatment, which will be discussed in chapter 3: section 3.2.3 presents the details of the actual implementation. Here we only give the general description.

The boundary condition for a viscous flow at a solid wall is the no-slip condition, which states that the relative velocity at the wall is zero:

$$\mathbf{u} - \mathbf{u}^{\text{wall}} = 0. \quad (2.76)$$

Eq. (2.76) gives three boundary conditions; the fourth condition is usually the temperature $T = g_1$, or the component normal to the wall of the gradient of the temperature, $\mathbf{n} \cdot \nabla T = \frac{\partial T}{\partial \mathbf{n}} = g_2$, where g_1 and g_2 are known and bounded functions. For an inviscid wall and symmetry plane instead, the slip wall condition

$$(\mathbf{u} - \mathbf{u}^{\text{wall}}) \cdot \mathbf{n} = 0 \quad (2.77)$$

is imposed, and the normal component of the viscous flux is set to zero (by applying a mirror boundary condition to the stress tensor and heat flux). At a far-field, the target state $\mathbf{Q} = (\rho_\infty, \rho_\infty \mathbf{u}_\infty, \rho_\infty E_\infty)$ is that of the undisturbed uniform flow, possibly modified by a vortex correction [117], with negligible viscous effects, i. e. zero viscous flux normal to the boundary. Regarding the SA model, the boundary conditions for $\tilde{v}$ are ([4]):

$$\text{no-slip wall: } \tilde{v} = 0 \quad (2.78)$$

$$\text{farfield: } \left. \frac{\tilde{v}}{v} \right|_\infty = 3 - 5 \quad \left(\left. \frac{\mu_t}{\mu} \right|_\infty \approx 0.2 - 1.3 \right) \quad (2.79)$$

$$\text{symmetry plane: } \frac{\partial \tilde{v}}{\partial \mathbf{n}} = 0. \quad (2.80)$$

Unless stated otherwise, the default value used for $\tilde{v}$ at the farfield boundary follows from choosing $\mu_t/\mu = 0.5$.

Table 2.2: Number of boundary conditions to be specified for different flow cases for the three-dimensional Navier-Stokes and Euler equations [81].

	Navier-Stokes	Euler
Supersonic inflow	5	5
Subsonic inflow	5	4
Subsonic outflow	4	1
Supersonic outflow	4	0
Wall	4	1

3

NUMERICAL SOLUTION

The preceding chapter concerned the mathematical model, i.e. the equations, that govern compressible viscous flows. These equations are a set of coupled partial differential equations which depend on both space and time. The topic of this chapter is how these equations are solved numerically. In the present work we adopt the common method of lines ([43]), which is described in section 3.1. This method consists of two steps, the first of which involves the discretization of the spatial derivatives present in the conservation equations. For this we use the high-order energy stable finite difference SBP-SAT schemes described in section 3.2. The second step of the method of lines is the integration in time of the discretized conservation equations. The different quadrature rules used for this are described in section 3.3. Sections 3.4 and 3.5 deal with aspects of the practical implementation of the described schemes. Finally, a summary of the solution strategy is given in section 3.6.

3.1 THE METHOD OF LINES

Consider the system of partial differential equations (PDEs)

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}_j}{\partial x_j} = \mathbf{S}.$$

The most common method in CFD to solve such a system of PDEs is the method of lines, which involves two steps. The first step consists of discretizing the spatial derivatives with algebraic approximations. Typical spatial discretizations in CFD are the finite volume, the finite element and the finite difference method. In the present work we focus on the finite difference method and we describe it in detail in section 3.2. Once this is achieved, only the discretization with respect to time remains. In other words, with only one remaining independent variable, we obtain a system of ordinary differential equations (ODEs)

$$\frac{d\mathbf{Q}(t)}{dt} + \mathbf{Res}(t, \mathbf{Q}(t)) = 0$$

that approximates the original system of PDEs. Here $\mathbf{Res}(t, \mathbf{Q}(t))$ is the discretized version of $\frac{\partial \mathbf{F}_j}{\partial x_j} - \mathbf{S}$. Then, the second step is the application of any integration algorithm for initial value problems to compute an approximate numerical solution to the original PDEs. There is a wide variety of integration algorithms available in literature. They can be divided in two broad categories: explicit and implicit

algorithms. Each algorithm has its own specific strengths and weaknesses, the choice is driven by physical, mathematical and computer hardware considerations. In the present study schemes belonging to both categories have been implemented, and they are described in section 3.3.

3.2 SPATIAL DISCRETIZATION

In the present work the spatial derivatives present in the conservation equations are discretized with high-order finite difference SBP operators ([105]). These operators are employed in combination with SAT terms (originally proposed in [10]); these are penalty-like terms that are used to impose boundary conditions in a weak formulation. Combinations of SBP derivative operators and SAT for boundary conditions constitute high-order schemes that are energy stable (see for example, [115, 29]). In fact, as will be shown in section 3.2.1, the SBP-SAT schemes possess an energy stability property similar to the advection equation. Thanks to this property, a significantly reduced amount of artificial dissipation is needed compared to that required for schemes which do not possess this (or a similar) property.

The key assumption of the design of these schemes is that the solution is *smooth*. On the one hand, smoothness limits the application of the schemes since it precludes flows with shocks, and turbulence must not be too far from resolved for the scheme to become unstable. On the other hand, smoothness allows to prove linear stability. In fact, the theoretical foundation for designing linearly stable schemes is that if the solution is smooth, one can linearize the solution around some point. Linear well-posedness of the continuous problem is achieved when it is possible to derive a bound for the perturbed solution [122]. This can be accomplished using the method of frozen coefficients, which allows to derive bounds for the entire problem including boundary conditions, non-square domains, etcetera (see for example [110] for an application of this method to the NS equations). Once the proper form of the boundary conditions is established, the discrete, numerical, problem is obtained by applying the SBP operators in conjunction with SAT terms. Then, it is possible to derive the corresponding bound on the numerical solution, including boundary conditions, curvilinear grids, multiblock grids and artificial dissipation. The result is a provably energy-stable, and for smooth problems, a convergent, high-order finite difference scheme.

In this section, we first give an example of the procedure for proving stability for a simple problem, the advection equation. Then, we describe the building blocks of the SBP-SAT schemes as used in the present work. These are the SBP derivative operators, section 3.2.2, the SAT terms, section 3.2.3, and the artificial dissipation operators, described in section 3.2.4.

3.2.1 Stability proofs using the energy method

Consistency and stability of a numerical scheme are key properties for a robust and accurate numerical solution. Consistency of a high-order finite-difference scheme, i. e. the requirement that the discretized partial differential equation approximates the correct partial differential equation as the mesh size tends to zero, is straightforward to verify (truncation error analysis). On the contrary, proving stability of schemes on bounded domains is a highly non-trivial task. If consistency and stability can be proven, then the Lax equivalence theorem [64] ensures that the finite difference method is also convergent, i. e. that for a fixed time T the numerical solution converges to the analytical solution as the mesh size decreases to zero. One successful way to obtain stability -and hence convergence- proofs is to employ the SBP-SAT schemes combined with the energy method. This combination has been used in *all* the works this thesis builds upon, from the first application to the advection equation [10], to the coupling of multiple domains in non-conforming meshes [69] (other related references are [73, 81, 110, 114, 80]).

The energy method is used to demonstrate the well-posedness of a continuous problem. This is accomplished by showing that the energy of the solution does not grow unbounded in time. The SBP-SAT schemes mimic this behavior in a discrete sense. In fact, these schemes are designed such that stability proofs can be obtained in the same way well-posedness of the continuous problem can be proven.

With a simple example we will briefly describe how stability proofs can be obtained. While the following example, reproduced from [125], is for a one-dimensional problem, the extension of the method to several space dimensions is not that much more complicated and examples are available in [111, 110, 74, 114, 115]. Most of the theory in the remainder of this section can be found in [115].

Consider the 1D linear advection equation with initial and boundary condition:

$$\begin{aligned} u_t + au_x &= 0, & 0 < x < 1, & \quad 0 < t \leq T \\ u(x, 0) &= f(x) \\ a^+ u(0, t) &= a^+ g_l(t), & a^+ &= \max(a, 0) \\ a^- u(1, t) &= a^- g_r(t), & a^- &= \min(a, 0) \end{aligned} \quad (3.1)$$

and assume $f(x)$ and $g_{r,l}(t)$ are bounded in $L^2[0, 1]$. Note that the advection velocity a is constant, it can be either positive or negative and that the boundary condition follows accordingly (once the sign of a is chosen, only one boundary condition is imposed). To demonstrate well-posedness of the continuous problem, we employ the energy method. Multiply eq. (3.1) by u and then integrate by parts in space:

$$\int_0^1 uu_t dx + a \int_0^1 uu_x dx = 0 \quad (3.2)$$

$$\int_0^1 uu_t dx + a u^2(x, t) \Big|_{x=0}^{x=1} - a \int_0^1 u_x u dx = 0. \quad (3.3)$$

From eq. (3.2) we obtain $a \int_0^1 uu_x dx = - \int_0^1 uu_t dx$, and we substitute this expression in eq. (3.3) which leads to:

$$\int_0^1 uu_t dx + \int_0^1 u_t u dx + a u^2(x, t) \Big|_{x=0}^{x=1} = 0. \quad (3.4)$$

We introduce the inner product between two real-valued functions $u, v \in L^2[0, 1]$ as $\int_0^1 uv dx$, and the corresponding norm

$$\int_0^1 uu dx = \|u\|^2. \quad (3.5)$$

Noting that $\int_0^1 uu_t dx + \int_0^1 u_t u dx = \|u\|_t^2$, and invoking the boundary conditions, eq. (3.4) can now be written as:

$$\begin{aligned} \|u\|_t^2 &= au^2(0, t) - au^2(1, t) \\ &= a^+ g_l(t)^2 - a^+ u^2(1, t) + a^- u^2(0, t) - a^- g_r(t)^2 \\ &\leq a^+ g_l(t)^2 - a^- g_r(t)^2 \end{aligned} \quad (3.6)$$

Integrating in time gives the bound:

$$\|u(\cdot, T)\|^2 \leq \|f\|^2 + a^+ \int_0^T g_l(t)^2 dt - a^- \int_0^T g_r(t)^2 dt \quad (3.7)$$

where T is an arbitrary but finite time. For a linear PDE such a bound is sufficient to prove well-posedness [115, 41].

Next we discretize the continuous problem (3.1) using SBP-SAT schemes and show that an analog energy estimate of the numerical solution is obtained. We introduce a uniform computational grid, $x_i = ih$, $i \in \{0, 1, 2, \dots, N\}$ where $h > 0$ is the grid spacing. For the moment, the time is left continuous. With each grid point x_i we associate the value $v_i(t)$, which represents the discrete approximate of the solution u , i.e.

$$u(x, t) \approx \mathbf{v}(t) = (v_0(t), \dots, v_N(t))^T.$$

We also need the following vectors that pick the solution at the left and right boundary, respectively:

$$\mathbf{e}_0 = (1, 0, \dots, 0)^T, \quad \mathbf{e}_N = (0, \dots, 1)^T.$$

We now introduce the SBP first-derivative operator D , $D\mathbf{v} \approx u_x$. The SBP operator D is a matrix with the following properties:

- $D = H^{-1}Q$ where H and Q are two matrices;
- $H = H^T > 0$, H is (symmetric) positive definite, i.e. $\mathbf{x}^T H \mathbf{x} > 0 \quad \forall \quad \mathbf{x} \neq \mathbf{0}$;
- $Q + Q^T = \text{diag}(-1, 0, \dots, 0, 1)$.

The matrix H defines a weighted (1D) L^2 -equivalent norm, i. e. for any $N \times 1$ vector $\mathbf{w}$, we have (note the analogy with eq. (3.5))

$$\|\mathbf{w}\|^2 = \mathbf{w}^T H \mathbf{w}. \quad (3.8)$$

These properties are all that is needed to proceed with the stability proofs. More details about the operators are given later in section 3.2.2.

We define the semi-discrete scheme for (3.1) as:

$$\mathbf{v}_t + \alpha D \mathbf{v} = \underbrace{\sigma_l H^{-1} \mathbf{a}^+ \mathbf{e}_0 (v_0 - g_l(t))}_{\text{SAT}_{\text{left}}} + \underbrace{\sigma_r H^{-1} \mathbf{a}^- \mathbf{e}_N (v_N - g_r(t))}_{\text{SAT}_{\text{right}}}. \quad (3.9)$$

The right-hand side are the SAT terms, which impose the boundary conditions weakly. They “drive” the numerical solution at the boundary v_0 and v_N , which are still unknowns, to the target values g_l and g_r , respectively. The SAT name comes from the idea of simultaneously approximating both the equation and the boundary condition. The two scalar parameters σ_l and σ_r are to be determined by the stability analysis: they are chosen such that $\|\mathbf{v}\|$ is bounded. The discrete analogy of the continuous energy estimate is obtained by multiplying (3.9) by $\mathbf{v}^T H$ and by adding its transpose:

$$\begin{aligned} \mathbf{v}^T H \mathbf{v}_t + \alpha \mathbf{v}^T Q \mathbf{v} &= \sigma_l \mathbf{a}^+ \mathbf{v}^T \mathbf{e}_0 (v_0 - g_l(t)) + \sigma_r \mathbf{a}^- \mathbf{v}^T \mathbf{e}_N (v_N - g_r(t)) \\ \|\mathbf{v}\|_t^2 + \alpha \mathbf{v}^T (Q + Q^T) \mathbf{v} &= 2\sigma_l \mathbf{a}^+ v_0 (v_0 - g_l(t)) + 2\sigma_r \mathbf{a}^- v_N (v_N - g_r(t)). \end{aligned}$$

Recalling that $Q + Q^T = \text{diag}(-1, 0, \dots, 0, 1)$, we obtain:

$$\|\mathbf{v}\|_t^2 - \alpha (v_0^2 - v_N^2) = 2\sigma_l \mathbf{a}^+ v_0 (v_0 - g_l(t)) + 2\sigma_r \mathbf{a}^- v_N (v_N - g_r(t)). \quad (3.10)$$

Note that the result of the summation of the flux terms, $\mathbf{v}^T (Q + Q^T) \mathbf{v}$, is simply two boundary terms. As this is the direct analogue of integrating by parts, the discrete process is called summation by parts (SBP). In order to prove the stability of the semi-discretization, it is sufficient to obtain a bound with $g_{r,l} = 0$, see [115]. In that case, eq. (3.10) becomes

$$\|\mathbf{v}\|_t^2 - \alpha (v_0^2 - v_N^2) = 2\sigma_l \mathbf{a}^+ v_0^2 + 2\sigma_r \mathbf{a}^- v_N^2. \quad (3.11)$$

Hence, in order to obtain a bounded growth $\|\mathbf{v}\|$ ($\|\mathbf{v}\|_t^2 \leq 0$), we must have $\sigma_l \leq -1/2$ and $\sigma_r \geq 1/2$. The more general case for which the boundary data is inhomogeneous leads to so-called strong stability, see [115, 41]¹. For strong stability, it can be shown that strict inequalities are necessary, i. e.

$$\sigma_l < -1/2, \quad \sigma_r > 1/2.$$

For instance, the standard choice $\sigma_l = -1$, $\sigma_r = 1$ yields

$$\|\mathbf{v}\|_t^2 - \alpha (v_0^2 - v_N^2) = -2\mathbf{a}^+ v_0 (v_0 - g_l(t)) + 2\mathbf{a}^- v_N (v_N - g_r(t))$$

¹ If a semi-discretization is strongly stable rather than (only) stable, then the boundary data $g_{r,l}$ need not to be differentiable in time. Thus, strong stability allows for more general boundary conditions.

or

$$\|\mathbf{v}\|_t^2 \leq -\mathbf{a}^+(v_0 - g_l(t))^2 + \mathbf{a}^+ g_l(t)^2 + \mathbf{a}^-(v_N - g_r(t))^2 - \mathbf{a}^- g_r(t)^2. \quad (3.12)$$

If $v_0 = g_l, v_N = g_r$, i. e. if the numerical solution satisfies exactly the boundary data, then (3.12) is the same as (3.6). However, this is not the case due to the weak enforcement of the boundary conditions and the extra terms introduce a small additional damping. Note that the SAT terms vanish upon substitution of the analytical solution. Thus, the SAT terms are accurate as they do not contribute to a truncation error in the scheme. The integration in time of (3.12) gives an energy estimate corresponding to (3.7), which proves stability of the semi-discrete problem. Additionally, if the time derivative in the semi-discrete stable problem is discretized with Runge-Kutta (RK) schemes, then the corresponding fully discrete problem is stable as well, see [62, 67].

The above example shows the general procedure to derive energy estimates for an SBP-SAT scheme. The same procedure can be applied to more complex problems, like systems of PDEs with additional diffusion terms, in three dimensions, with curvilinear grids and several types of boundary conditions. Clearly, in these cases the algebra for proving stability becomes more involved. Semi-discrete energy estimates have been derived for the linearized Euler and linearized Navier-Stokes equations, see [110, 114, 80, 71] and references therein. Artificial dissipation and different types of boundary conditions, including far-field, walls and grid block interfaces, are included in the theory. The actual SAT terms used in this thesis for these boundary types are given in section 3.2.3. The extension to non-linear problems is possible since, for flows with smooth solutions, linear stability implies convergence of the non-linear equation as the grid size tends to zero, see [105].

3.2.2 SBP first-derivative operators

The first derivative is approximated by the matrix $D = H^{-1}Q$, i. e.

$$\partial u / \partial x \approx H^{-1}Q\mathbf{v} = D\mathbf{v}.$$

The matrix H is used to define an L^2 -equivalent norm, $\|\mathbf{v}\|_H^2 = \mathbf{v}^T H \mathbf{v}$. Thus, as done by many authors, we will refer to this matrix as the norm matrix. The truncation error of SBP operators is not uniform in space. Let w denote a smooth function and define a grid function $\bar{w} = (w(x_0), \dots, w(x_N))^T$ and its derivative $\bar{w}_x = (w_x(x_0), \dots, w_x(x_N))^T$. We have

$$D\bar{w} = \bar{w}_x + \bar{T}$$

where $\bar{T}$ is the truncation error. In general, for the SBP operators it takes the form,

$$\bar{T}^T = \underbrace{(\mathcal{O}(h^s), \dots, \mathcal{O}(h^s))}_{\text{left boundary}} \underbrace{(\mathcal{O}(h^p), \dots, \mathcal{O}(h^p))}_{\text{interior}} \underbrace{(\mathcal{O}(h^s), \dots, \mathcal{O}(h^s))}_{\text{right boundary}}. \quad (3.13)$$

Figure 3.1: Uniform 1D grid.

where $s < p$ and the lower accuracy is confined to a few (finite) number of points close to the boundary. SBP operators exist with various orders of accuracy, [104]. In particular, if H is a diagonal matrix, there are SBP operators with p even and $p \leq 8$, and $s = p/2$. In this case the global order of accuracy in the L^2 -norm for PDEs is $p/2 + 1 = s + 1$ [113]. If H is allowed to have off-diagonal elements (so-called block norm) for a few points near the boundary, then $s = p - 1$ can be achieved. However, in the present work we use diagonal norms because they preserve energy estimates on curvilinear grids [109, 111]. On the contrary, block norms do not guarantee stability on curvilinear grids unless they are modified as recently proposed in [70].

For the NS equations, we use the same approximation of the first derivative to compute the viscous fluxes and to differentiate the fluxes (non-compact schemes)². The schemes we use in the present work have an *interior order-boundary order* of accuracy of 8-4, 6-3, 4-2, and 2-1. Hence, since we employ diagonal norms, their global convergence rates are theoretically 5, 4, 3, and 2, respectively. From now on, we will refer to the schemes by their global order of accuracy.

As an example, here we report the operator for second and third order accurate discretization of the first derivative (see fig. 3.1) together with the corresponding norms. The fourth and fifth order operators with their norms can be found in appendix B.

$$\frac{\partial u}{\partial x} \approx D^{2\text{nd}} \mathbf{v} = \frac{1}{h} \begin{bmatrix} -1 & 1 & & & \\ -\frac{1}{2} & 0 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{2} & 0 & \frac{1}{2} \\ & & & -1 & 1 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix}$$

$$H_2 = h \cdot \text{diag} \left[\frac{1}{2}, 1, \dots, 1, \frac{1}{2} \right],$$

² For compact schemes see [72, 74, 69].

$$\frac{\partial \mathbf{u}}{\partial \mathbf{x}} \approx \mathbf{D}^{3\text{rd}} \mathbf{v} = \frac{1}{h} \begin{bmatrix} -\frac{24}{17} & \frac{59}{34} & -\frac{4}{17} & -\frac{3}{34} & 0 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{4}{43} & -\frac{59}{86} & 0 & \frac{59}{86} & -\frac{4}{43} & 0 & 0 \\ \frac{3}{98} & 0 & -\frac{59}{98} & 0 & \frac{32}{49} & -\frac{4}{49} & 0 \\ 0 & 0 & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ \vdots \end{bmatrix}$$

$$\mathbf{H}_3 = h \cdot \text{diag} \left[\frac{17}{48}, \frac{59}{48}, \frac{43}{48}, \frac{49}{48}, 1, \dots, 1, \frac{49}{48}, \frac{43}{48}, \frac{59}{48}, \frac{17}{48} \right],$$

Note that in the interior of the domain the usual anti-symmetric finite difference stencils are used and the norm reduces to the identity matrix. However, the stencils are modified at the boundaries in order to prove stability, at the price of reduced accuracy. Finally, we remark that, when solving the conservation equations, the derivative operators are applied in the computational space (ξ, η, ζ) , where the grid spacing is uniform and $h = 1$ in all directions. Nevertheless, the derivative operators used in the three directions may be different.

3.2.3 Boundary conditions (SAT)

The boundary conditions are enforced in a weak fashion via the SATs. The SATs are penalty terms that are added to the conservation equations on those block boundaries where a halo treatment (see section 3.5.1) is not used. The boundaries where the SATs are applied can be physical, like a solid wall or a far-field, where the solution is assumed to be known. Or they can be numerical boundaries, which have been introduced for mesh generation reasons, like an internal interface between computational blocks, or to separate parts of the computational domain in relative motion, as is the case of sliding interfaces.

The purpose of this section is to show the actual implementation used in the present work. All SAT terms that follow are added to the right-hand side of the conservation equations (see eq. (A.1)). Flow and turbulence equations are considered in a fully coupled manner.

3.2.3.1 Far-field

The treatment of the far-field boundary condition is based on the analysis carried out in [110]. The SAT term for the vertices residing on a far-field boundary $\xi = \xi_{\min}$ (see fig. 3.2), is

$$\text{SAT}_{\text{far-field}} = h_{\xi,1}^{-1} \left[-\sigma^i A_{\xi}^+ \left(\mathbf{Q} - \mathbf{g}^i \right) \mathbf{J}^{-1} + \sigma^v \left(\tilde{\mathbf{E}}^v - \mathbf{g}^v \right) \right] \quad (3.14)$$

where

$$A_{\xi}^+ = \frac{A_{\xi} + |A_{\xi}|}{2}, \quad A_{\xi} = \frac{\partial \tilde{\mathbf{E}}^c}{\partial \widetilde{\mathbf{Q}}}, \quad (3.15)$$

Figure 3.2: Definition of the min and max boundaries for a 2D computational block. These boundaries are lines (in 3D, they are surfaces) where one of the curvilinear coordinates is constant.

$\mathbf{Q} = \tilde{\mathbf{J}}\tilde{\mathbf{Q}}$ contains the local conserved variables, and $\tilde{\mathbf{J}}$ is the determinant of the metric Jacobian. The scalar $h_{\xi,1}^{-1}$ is the inverse of the first element of the norm $\mathbf{H}_\xi$. A_ξ is the Jacobian of the curvilinear inviscid flux in the ξ direction, $\tilde{\mathbf{E}}^c$. The matrix $|A_\xi|$ is defined as:

$$|A_\xi| = \mathbf{R}|\Lambda|\mathbf{L}^T, \quad (3.16)$$

where the rows of $\mathbf{R}$ are the right eigenvectors of A_ξ , and $\mathbf{L}^T = \mathbf{R}^{-1}$. The matrix Λ contains the eigenvalues of A_ξ on its diagonal (the matrices $\mathbf{L}^T$ and $\mathbf{R}$ form together a similarity transformation that diagonalizes the matrix A_ξ , see appendix C). Thus, A_ξ^+ captures the incoming characteristics. As suggested in [110], the matrix $\mathbf{A}$ is evaluated at the Roe-averaged state between $\mathbf{Q}$ and $\mathbf{g}^i$ (see below). The constants σ^i and σ^v multiply the inviscid and viscous part, respectively, of the penalty term and they are chosen based on stability considerations. In the present work, we choose $\sigma^{i,v} = 1$ (for stability, $\sigma^i \geq 0.5$).

The vectors $\mathbf{g}^{i,v}$ represent the target penalty states. For a far-field boundary,

$$\mathbf{g}^i = [\rho_\infty, \rho_\infty \mathbf{u}_\infty, \rho_\infty E_\infty, \rho_\infty \tilde{v}_\infty]^T \quad (3.17)$$

is the state of the undisturbed flow at infinity, in 2D possibly modified by a vortex correction [117]. The vector $\mathbf{g}^v$ is the target viscous flux that is subtracted from the current numerical viscous flux in the ξ direction, $\tilde{\mathbf{E}}^v$. For a far-field boundary, the target viscous flux is set to zero: $\mathbf{g}^v = \mathbf{0}$. In case of inviscid flow, the viscous term is clearly not considered.

As for all SAT to follow, for a $\xi_{\max}$ boundary, the sign of the penalty terms is reversed and $A_\xi^- = (A_\xi - |A_\xi|)/2$ is used instead of A_ξ^+ . Furthermore, when dealing with boundaries normal to the other two coordinate directions, A_ξ is replaced by either A_η or A_ζ . Accordingly, $h_{\xi,1}^{-1}$ is replaced by either $h_{\eta,1}^{-1}$ or $h_{\zeta,1}^{-1}$ and the fluxes $\tilde{\mathbf{E}}^{c,v}$ are replaced by either $\tilde{\mathbf{F}}^{c,v}$ or $\tilde{\mathbf{G}}^{c,v}$.

In [110] it is shown that, for a viscous subsonic outflow, the SAT term (3.14) introduces one constraint too many. In inviscid flow, the characteristic boundary condition is recovered and the correct number of constraints for well-posedness

is enforced. The authors of [110] propose a modification to the matrix Λ which, however, we do not adopt. As shown in chapter 4 where the numerical results are presented, we did not experience problems using the simple form (3.14).

As will be seen in the remainder of this section, the matrix-vector product $A_{\xi,\eta,\zeta}^{\pm} \cdot (\mathbf{Q} - \mathbf{g}^i)$, which constitutes the inviscid penalty term, is common to all boundary conditions. From an implementation point of view, it is therefore convenient to express A (in either ξ , η or ζ direction) in terms of its eigenvalues and to find a closed form of the matrix-vector product. In appendix C, the derivation of the Jacobian and the matrix-vector product written in such terms is carried out for the extended system of the RANS equations. Appendix C also provides the matrices A , Λ , R and L .

3.2.3.2 Adiabatic viscous wall

The SAT term for a viscous wall is

$$\text{SAT}_{\text{vis. wall}} = h_{\xi,1}^{-1} \left[-\sigma^i A_{\xi}^+ (\mathbf{Q} - \mathbf{g}^i) J^{-1} - \Sigma^{v,1} (\mathbf{Q} - \mathbf{g}^i) + \sigma^{v,2} (\tilde{\mathbf{E}}^{\text{hf}} - \mathbf{g}^v) \right] \quad (3.18)$$

where

$$\sigma^i = 1, \quad \sigma^{v,2} = 1, \quad (3.19)$$

$$\mathbf{g}^i = \begin{bmatrix} \rho \\ \rho(-\mathbf{u} + 2\mathbf{u}^{\text{wall}}) \\ (\rho E)_{\text{pen}} \\ 0 \end{bmatrix}, \quad (3.20)$$

$$\Sigma^{v,1} = \frac{\xi_x^2 + \xi_y^2 + \xi_z^2}{J\rho} \begin{bmatrix} 0 & & & \\ & \mu^{\text{eff}} I_3 & & \\ & & \frac{\gamma\mu}{\text{Pr}} + \frac{\gamma\mu^t}{\text{Pr}^t} & \\ & & & \text{diff}_{\tilde{v}} \end{bmatrix}, \quad (3.21)$$

and

$$\tilde{\mathbf{E}}^{\text{hf}} = J^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -\xi_x \dot{q}_x - \xi_y \dot{q}_y - \xi_z \dot{q}_z \\ 0 \end{bmatrix}, \quad \mathbf{g}^v = J^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \dot{q}_{\text{target}} \\ 0 \end{bmatrix}. \quad (3.22)$$

I_3 is the 3 by 3 unity matrix. Note that in $\mathbf{g}^i$ the penalty state for the velocity vector is defined as a mirror of the local vector. Another option, adopted in [114], would be to penalize the local velocity against its normal component (and to set $\sigma^i = 2$). Furthermore, it is assumed that the density and pressure of the penalty state are the same as for the current solution. The penalty state for the energy equation, $(\rho E)_{\text{pen}}$, follows from this assumption and by considering the penalty velocity vector defined above. As mentioned in section 2.4, the target value for $\rho\tilde{v}$ at the wall is zero.

The last two terms in eq. (3.18) represent the viscous contributions. The term multiplied by $\Sigma^{v,1}$ enforces the no-slip condition, while the one multiplied by $\sigma^{v,2}$ ensures that the wall is adiabatic by choosing a target zero heat flux, $\dot{q}_{\text{target}} = 0$.

The variable $\text{diff}_{\tilde{v}}$ represents the diffusion coefficient for the transported variable of the SA turbulence model, i. e. (see eq. (2.61) and also [114, 85])

$$\text{diff}_{\tilde{v}} = (\mu + (f_n + c_{b2}))\rho\tilde{v}/\sigma. \quad (3.23)$$

3.2.3.3 Inviscid wall and symmetry boundary conditions

Inviscid wall and symmetry boundary conditions are treated in the same way. For these boundaries the penalty term is the following:

$$\text{SAT}_{\text{inv. wall}} = h_{\xi,1}^{-1} \left[-\sigma^i A_{\xi}^+ (\mathbf{Q} - \mathbf{g}^i) \mathbf{J}^{-1} + \sigma^v (\tilde{\mathbf{E}}^v - \mathbf{g}^v) \right]. \quad (3.24)$$

Here the target penalty state $\mathbf{g}^i$ is constructed assuming zero normal (relative) velocity, such that the target velocity vector is purely tangential. We define the normal velocity vector $\mathbf{u}_n$ as

$$\mathbf{u}_n = u_n \mathbf{n}, \quad u_n = \mathbf{u} \cdot \mathbf{n} \quad (3.25)$$

where $\mathbf{n}$ is the unit normal vector that, for a $\xi = \text{constant}$ boundary, is given by:

$$\mathbf{n} = [n_x, n_y, n_z]^T = [\xi_x, \xi_y, \xi_z]^T / \sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}. \quad (3.26)$$

Then, the inviscid penalty state is:

$$\mathbf{g}^i = \begin{bmatrix} \rho \\ \rho(\mathbf{u} - (u_n - u_n^{\text{wall}})\mathbf{n}) \\ (\rho E)_{\text{pen}} \\ \rho\tilde{v} \end{bmatrix} \quad (3.27)$$

As for the adiabatic wall, the pressure and density are the same for the penalty data and for the current solution $\mathbf{Q}$. The total energy of the penalty data, $(\rho E)_{\text{pen}}$, follows from this assumption and by considering the velocity of the penalty state $(\mathbf{u} - (u_n - u_n^{\text{wall}})\mathbf{n})$, see eq. (3.27). Furthermore, no inviscid penalty is directly applied to $\rho\tilde{v}$, which is assumed to be the same for the penalty data as well.

For a viscous flow problem an additional penalty term must be added to the residual. The viscous penalty state for an inviscid wall (or symmetry plane) is obtained by applying a mirror boundary condition to the stress tensor and the heat flux. For $\rho\tilde{v}$, the target target viscous flux is simply set to zero. Thus, the vector $\mathbf{g}^v$ is

$$\mathbf{g}^v = \mathbf{J}^{-1} \begin{bmatrix} 0 \\ g_2^v \\ g_3^v \\ g_4^v \\ g_5^v \\ 0 \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} 0 \\ \xi_x \tau_{xx}^{\text{pen}} + \xi_y \tau_{xy}^{\text{pen}} + \xi_z \tau_{xz}^{\text{pen}} \\ \xi_x \tau_{xy}^{\text{pen}} + \xi_y \tau_{yy}^{\text{pen}} + \xi_z \tau_{yz}^{\text{pen}} \\ \xi_x \tau_{xz}^{\text{pen}} + \xi_y \tau_{yz}^{\text{pen}} + \xi_z \tau_{zz}^{\text{pen}} \\ u g_2^v + v g_3^v + w g_4^v - \xi_x \dot{q}_x^{\text{pen}} - \xi_y \dot{q}_y^{\text{pen}} - \xi_z \dot{q}_z^{\text{pen}} \\ 0 \end{bmatrix} \quad (3.28)$$

where

$$\dot{\mathbf{q}}^{\text{pen}} = [\dot{q}_x - 2\dot{q}_n n_x, \dot{q}_y - 2\dot{q}_n n_y, \dot{q}_z - 2\dot{q}_n n_z]^T, \quad \dot{q}_n = \dot{\mathbf{q}} \cdot \mathbf{n}. \quad (3.29)$$

The explicit formulation of the terms τ_{ij}^{pen} is given in appendix B.4.1.

3.2.3.4 Isothermal wall

The isothermal wall is similar to the two preceding boundary conditions and is again based on the analysis carried out in [114]. The penalty term is:

$$\text{SAT}_{\text{isot. wall}} = h_{\xi,1}^{-1} \left[-\sigma^i A_{\xi}^{+, \text{lim}} (\mathbf{Q} - \mathbf{g}^i) \mathbf{J}^{-1} - \Sigma^{v,1} (\mathbf{Q} - \mathbf{g}^v) \right]. \quad (3.30)$$

The inviscid penalty state $\mathbf{g}^i$ is defined using the fact that pressure remains the same while the density is computed such that the prescribed wall temperature, T_{wall} , is obtained. Furthermore, $\sigma^i = 2$, and the target penalty velocity is purely tangential, as for the inviscid wall.

$$\mathbf{g}^i = \begin{bmatrix} \rho_{\text{pen}} \\ \rho \mathbf{u} - \rho_{\text{pen}} (\mathbf{u}_n - \mathbf{u}_n^{\text{wall}}) \mathbf{n} \\ \rho_{\text{pen}} E_{\text{pen}} \\ 0 \end{bmatrix}, \quad \rho_{\text{pen}} = \frac{p}{RT_{\text{wall}}}. \quad (3.31)$$

Note that the inviscid flux Jacobian is defined in a slightly different way. The matrix $A_{\xi}^{+, \text{lim}}$ is defined as:

$$A_{\xi}^{+, \text{lim}} = \mathbf{R} A^{+, \text{lim}} \mathbf{L}^T, \quad (3.32)$$

$$\Lambda^{+, \text{lim}} = \text{diag} [\max(r \epsilon_{ac}, \lambda_1), \max(r \epsilon_{ac}, \lambda_2), \max(r \epsilon_{sh}, \lambda_3), \\ \max(r \epsilon_{sh}, \lambda_4), \max(r \epsilon_{sh}, \lambda_5), \max(r \epsilon_{sh}, \lambda_6)], \quad (3.33)$$

$$\epsilon_{ac} = 0.25, \quad \epsilon_{sh} = 0.025, \quad (3.34)$$

where r is the spectral radius of A_{ξ} . For this particular boundary condition, it was found that preventing the eigenvalues of the inviscid Jacobian from becoming smaller than a certain fraction of the spectral radius improved the convergence behavior considerably. This limiting of the eigenvalues is usually done in the context of matrix dissipation, but other authors employ it for SAT terms as well ([85]).

The viscous penalty state $\mathbf{g}^v$ is computed assuming zero relative velocity (and not zero *normal* relative velocity as for $\mathbf{g}^i$). The term $\rho_{\text{pen}} E_{\text{pen}}^v$ follows from this assumption.

$$\mathbf{g}^v = \left[\rho_{\text{pen}}, \rho_{\text{pen}} \mathbf{u}^{\text{wall}}, \rho_{\text{pen}} E_{\text{pen}}^v, 0 \right]^T. \quad (3.35)$$

The matrix $\Sigma^{v,1}$ is the same as in eq. (3.21) with the only difference that it is evaluated at the Roe-averaged state between $\mathbf{Q}$ and $\mathbf{g}^v$. This is due to the fact that $\rho \neq \rho_{\text{pen}}$. Finally, as this is an isothermal viscous wall boundary, the heat flux will follow as part of the solution and therefore no penalty is applied to the energy equation in terms of heat flux.

Figure 3.3: Interfaces between computational blocks. Nodes on the interface are duplicated. One set belongs to the left block, the other set to the right block.

3.2.3.5 Internal block interfaces

Internal interfaces are treated as described in [80]. The SAT term is the following:

$$\text{SAT}_{\text{internal}} = h_{\xi,1}^{-1} \left[-A_{\xi}^{\pm} \left(\mathbf{Q} - \mathbf{g}^i \right) \mathbf{J}^{-1} - \sigma^{v,1} \tilde{\mathbf{B}}_{11} \left(\mathbf{Q} - \mathbf{g}^i \right) + \sigma^{v,2} \left(\tilde{\mathbf{E}}^v - \mathbf{g}^v \right) \right]. \quad (3.36)$$

The target inviscid penalty state, $\mathbf{g}^i$, is the state of the coincident node in the adjacent block (see fig. 3.3a). Note that the inviscid term is defined such that both the incoming and outgoing characteristics may be considered. The matrix $A_{\xi}^{\pm}$ reads:

$$A_{\xi}^{\pm} = \sigma_{\text{in}}^i A_{\xi}^{+} + \sigma_{\text{out}}^i A_{\xi}^{-} \quad (3.37)$$

where

$$\sigma_{\text{out}}^i = 1 - \sigma_{\text{in}}^i. \quad (3.38)$$

For stability $\sigma_{\text{in}}^i \geq 0.5$, and we typically use $\sigma_{\text{in}}^i = 1.0$, leading to $\sigma_{\text{out}}^i = 0$.

The viscous part of $\text{SAT}_{\text{internal}}$ is composed of two terms. Using the same terminology and symbols as in [80], the viscous stability term is the one that multiplies the difference in the state vector. In the present work, the constant that multiplies this term is given by

$$\sigma^{v,1} = \frac{\alpha + \beta}{16\alpha\beta}, \quad \alpha = \beta = h_{\xi,1}. \quad (3.39)$$

The matrix $\tilde{\mathbf{B}}_{11}$ is related to the viscous Jacobian, and is derived based on the work carried out in [80]; it can be found in appendix B.4.2.

The last term in eq. (3.36), is the standard viscous penalty term that multiplies the difference in the viscous flux vectors. Thus, $\mathbf{g}^v$ represents the curvilinear viscous flux calculated in the coincident node in the adjacent block. For this type of boundary, we chose $\sigma^{v,2} = 0.5$.

3.2.3.6 Sliding interfaces

The SAT term for the boundary condition at a sliding interface is analogous to eq. (3.36). However, in this case the penalty states $\mathbf{g}^{i,v}$ are determined based on

Figure 3.4: Stencil for 4th order interpolation operator.

interpolation. In fact, in the case of the internal boundary condition, there is a 1-to-1 matching between the vertices at the interface, so the target penalty state is clearly defined: it is the solution of the overlapping vertex on the other side of the interface. For a sliding interface, however, such 1-to-1 relations do not exist and interpolation is needed (see fig. 3.3b). The interpolation operators that we employ are described in [35] (see also [71]). They are obtained subject to the following simplifying assumptions:

1. the problem is 2D, thus the interpolation is 1D
2. the mesh spacing is constant along the interface on both sides of the interface
3. the same number of points on either side of the interface (1:1 compression ratio)
4. the problem is periodic in the direction parallel to the interface.

Assumptions 2 and 3 imply that the sliding interface can reside on one face only of the computational block. Assumption 4 implies that there is no boundary closure and therefore the same stencil can be used throughout the interface. The procedure to construct the operators for a more general case, for which assumptions 3 and 4 are removed, can be found in [71] where, however, only stationary interfaces are considered.

Here we present the 2nd and 4th order accurate interpolation operators resulting from the assumptions outlined above. Figure 3.4 shows the stencil for the 4th order operator. There we introduce a general notation for all stencils: P is the vertex on the one side of the interface where we wish to calculate the penalty state; the penalty state in P is a weighted sum of the solution in the neighboring points N_i , which reside on the other side of the interface. The 2nd order stencil uses two neighbors and their interpolation weights are given in equation 3.40, which is the standard linear interpolation:

$$\text{Sol}(P) = w_0 \text{Sol}(N_0) + w_1 \text{Sol}(N_1) \quad (3.40a)$$

$$w_0 = (h - h_0)/h, \quad w_1 = h_0/h \quad (3.40b)$$

The 4th order stencil uses four neighbors, and their interpolation weights are given in equation 3.41. The 6th and 8th order stencils are presented in appendix B.4.3.

$$\text{Sol}(P) = \sum_{i=-1}^2 w_i \text{Sol}(N_i), \quad \alpha = (h - h_0)/h \quad (3.41a)$$

$$w_{-1} = -1/6(\alpha - 2)(\alpha - 1)(\alpha), \quad w_0 = +1/2(\alpha - 2)(\alpha - 1)(1 + \alpha), \quad (3.41b)$$

$$w_{+1} = -1/2(\alpha - 2)(\alpha)(1 + \alpha), \quad w_{+2} = +1/6(\alpha - 1)(\alpha)(1 + \alpha). \quad (3.41c)$$

In order to have a consistent discretization, we use the interpolation operator that corresponds to the accuracy of the interior stencil of the scheme. Therefore, the 4th order interpolation operator is used with the 3rd order scheme (which is 2nd order accurate at the boundary and 4th order accurate in the interior); the 6th order interpolation operator is used with the 4th order scheme, and the 8th order interpolation operator is used with the 5th order scheme.

Referring again to [71], for stability to be proven, the following condition must be met:

$$H_R^\eta I_{L2R} = I_{R2L}^\top H_L^\eta \quad (3.42)$$

where H_L^η and H_R^η are the norms in the direction parallel to the interface in the left and right domain, respectively, and I_{L2R} and I_{R2L} are the interpolation operators. Because of assumption 4, H_L^η and H_R^η reduce to the identity matrix. It remains to show that $I_{L2R} = I_{R2L}^\top$, which is easily verified by computing the weights w_i on both sides of the interface.

3.2.4 Artificial dissipation

As shown in section 3.2.2, an SBP operator is essentially a centered finite difference scheme with a specific boundary treatment. It is well-known that centered finite difference schemes are inherently non-dissipative, thus for non-linear convection problems they require the addition of artificial dissipation in order to absorb the energy of the unresolved modes. This may be accomplished by adding dissipation operators that are constructed with high order undivided differences. We employ operators based on the ones proposed in [73]. These operators have been designed such that they possess the following properties:

1. they efficiently reduce spurious oscillations;
2. they preserve the accuracy of the finite difference scheme;
3. their computational work is comparable to that of the original scheme;
4. they preserve the stability properties of the original scheme.

In 1D, given global order of accuracy $p + 1$ of the original scheme, the artificial dissipation operators have the form ([122])

$$AD_{p+1} = -\tilde{H}_{(p+1)}^{-1} (\tilde{D}_p^\top (\alpha \Lambda) B_p \tilde{D}_p) \mathbf{v}, \quad (p = 1, 2, 3, 4), \quad (3.43)$$

and they are added to the right hand side of the conservation equations. For a 3D problem, artificial dissipation must be added in all three coordinate directions (ξ, η, ζ) . Thus, in this case, the residual in a grid point receives three contributions of the type (3.43), one for each direction.

In eq. (3.43), the matrix $\tilde{H}_{(p+1)}$ is defined as

$$\tilde{H}_{(p+1)} = H_{(p+1)}/h, \quad (3.44)$$

and $H_{(p+1)}$ is the norm of the original scheme (which is of order of accuracy $2p$ in the interior and p at the boundary). $\tilde{D}_p/h^p$ is a consistent approximation of $\partial^p/\partial x^p$, i. e.

$$\frac{1}{h^p} \tilde{D}_p(\cdot) \approx \frac{\partial^p(\cdot)}{\partial x^p}. \quad (3.45)$$

Note that $\tilde{D}_p$ is an undivided difference. α is a non-negative constant (see below) and $\mathbf{v}$ represents the numerical solution vector. The tilde in $\tilde{D}_p$ and $\tilde{H}_{(p+1)}$ indicates that there is no dependence on the grid spacing h . The matrices $\tilde{D}_p$ can be found in appendix B.3. Note that in the present work we use $\tilde{D}_2$ and B_2 for the 2nd order scheme (which is second order accurate in the interior and first order accurate at the boundary) instead of $\tilde{D}_1$ and B_1 . Thus, $\tilde{D}_2$ and B_2 are used for both the 2nd and the 3rd order scheme. The matrices B_p are symmetric positive diagonal matrices that read:

$$B_1 = \text{diag}[1, 1, 1, \dots, 1, 1, 0] \quad (3.46a)$$

$$B_2 = \text{diag}[0, 1, 1, \dots, 1, 1, 0] \quad (3.46b)$$

$$B_3 = \text{diag}[0, 1, 1, \dots, 1, 0, 0] \quad (3.46c)$$

$$B_4 = \text{diag}[0, 0, 1, \dots, 1, 0, 0]. \quad (3.46d)$$

Λ is a diagonal matrix whose elements are (the average of) the local spectral radius of the inviscid flux Jacobian, divided by the determinant of the metric Jacobian J . In case an average is used, it is to guarantee that a symmetric dissipation operator is obtained, i. e. the residual is the same when the index direction is reversed. For instance, when the artificial dissipation is applied in the ξ direction, indexed with i , the spectral radius is (see eq. (C.14)):

$$\lambda_i = |\mathbf{U}| + c \sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}$$

where $\mathbf{U}$ is the contravariant velocity component and c is the local speed of sound. If B_1 or B_3 are used, then an average must be taken, i. e. the elements of Λ become

$$\frac{\lambda_i J_i^{-1} + \lambda_{i+1} J_{i+1}^{-1}}{2}.$$

Note that the artificial dissipation operators in eq. (3.43), are based on a centered, second order accurate undivided difference operator of order $2p$ for a $2p^{\text{th}}$ order accurate method. They are derived from even-order derivatives, which damp high frequency components of the solution more efficiently than odd-order derivatives [70] (property 1). Furthermore, a centered second order accurate difference approximation of $\partial^{2n}/\partial \xi^{2n}$ includes n neighboring points on each side. A centered finite difference scheme approximating the first derivative to $2p^{\text{th}}$ order accuracy in the

interior will include p neighbors on each side. Hence, the stencil of the artificial dissipation operators has the same width as the one of the original scheme³ (property 3). Additionally, applying $\tilde{D}_{2p}$ to a smooth function ψ gives $h^{2p}(\psi_{(2p)x} + \mathcal{O}(h^2))$. Hence, $\tilde{D}_{2p}$ is of order $\mathcal{O}(h^{2p})$ (property 2, in the interior). However, note that, due to the actual form of the dissipation operator AD_{p+1} , eq. (3.43), the order of accuracy in the interior of AD_{p+1} will be $(2p - 1)$. The B_p matrices constitute a modification to the stencil such that accuracy is preserved also at the boundaries. The scaling with the undivided norm $\tilde{H}$ is done such that stability can be proven (property 4).

As can be seen from eq. (3.43), the artificial dissipation is scalar. An alternative would be to use matrix dissipation, but it turned out that for the high-order schemes the differences between results obtained with scalar dissipation and the ones obtained with matrix dissipation are very small. Furthermore, the scalar dissipation schemes require less Central Processing Unit (CPU) time, partly because the application of a matrix dissipation operator requires more floating point operations than a scalar dissipation operator, and, most importantly, because a matrix operator makes the Jacobian of the residual evaluation function less diagonally dominant and thus more difficult to invert (see section 3.4.1). Hence scalar dissipation is used throughout the present work.

Unless stated otherwise, when solving the NS and Euler equations, the following values for the coefficient α , see eq. (3.43), are used: $\alpha = 0.02$ for the second order scheme, $\alpha = 0.01$ for the third order scheme, $\alpha = 0.005$ for the fourth order scheme and $\alpha = 0.0025$ for the fifth order scheme. When solving the (U)RANS equations, these values are doubled. These values have been chosen favoring stability and robustness over accuracy. Although not done by default, when simulating smooth flows and when using sufficiently fine grids, these values can be halved. Furthermore, note that the values of the artificial dissipation coefficient α used in the present work are generally smaller than those typical of a standard second-order finite volume scheme, as the Jameson-Schmidt-Turkel (JST) scheme [55]. In fact, the (high-order) dissipation term in the JST scheme is multiplied by a constant whose typical values are $\alpha_{JST} \in [1/64, 1/32] \approx [0.016, 0.031]$ ([21]).

3.2.4.1 Shock Capturing

Although the schemes used in the present work have been designed for smooth solutions, an attempt has been made to introduce some shock capturing capabilities. The approach adopted here is rather heuristic and unfortunately we are not able to prove that it preserves the stability of the original scheme. Its main advantage, though, is its simplicity and easiness of implementation. We mention that, more recent and elegant, techniques [112, 31] have been developed that combine shock capturing and entropy conservation.

³ In our case, the only exception is the second order scheme, for which the stencil of the artificial dissipation operator is larger than the stencil of first derivative operator. This is due to the choice of using D_2 instead of D_1 .

We introduce a modified artificial dissipation operator:

$$AD_{p+1}^* = -\tilde{H}_{(p+1)}^{-1} (\tilde{D}_1^T (c_2 \epsilon_2 \Lambda) B_1 \tilde{D}_1 + \tilde{D}_p^T (\alpha^* \Lambda) B_p \tilde{D}_p) \mathbf{v}. \quad (3.47)$$

The ϵ_2 term is the pressure-switch term firstly introduced in [55], i.e.

$$\phi_i = \frac{|p_{i-1} - 2p_i + p_{i+1}|}{|p_{i-1} + 2p_i + p_{i+1}|} \quad (3.48)$$

$$\epsilon_2|_i = \max(\phi_{i-1}, \phi_i, \phi_{i+1}) \quad (3.49)$$

and

$$\alpha^* = \max(0, \alpha - c_2 \epsilon_2). \quad (3.50)$$

Again, $\tilde{D}_2$ is used for the 2nd order scheme. An example of this approach in the context of SBP-SAT schemes can be found in [24]. The idea of this modified operator is to blend two different artificial dissipation operators ([55]): a high-order one that acts in case the solution is smooth, and a first-order dissipation operator that is activated only in the vicinity of a shock, where a discontinuity is detected via the pressure switch ϕ . Close to the shock, the high-order operator is turned off to avoid the oscillations that are typical for such operators; conversely, the first-order operator should provide a monotonic solution.

Implementation details

It turned out that the non-linearities introduced with the modified operator hampered the convergence behavior of our computational method (described in section 3.4). In order to circumvent this issue the following modifications have been devised:

- ϕ_i is slightly different, we compute it as

$$\phi_i = \left[\frac{|p_{i-1} - 2p_i + p_{i+1}|}{|p_{i-1} + 2p_i + p_{i+1} + p_{lim}|} \right]^{1.1} \quad \text{with } p_{lim} = 0.001 p_\infty$$
- $\epsilon_2 = \min(\epsilon_2, 0.25)$
- close to zero α^* is blended to avoid the clipping of the max function:

$$\alpha^* = \begin{cases} \alpha - c_2 \epsilon_2 & \text{if } (\alpha - c_2 \epsilon_2) \geq a \\ \frac{a^2 - b(\alpha - c_2 \epsilon_2)}{(-b + 2a) - (\alpha - c_2 \epsilon_2)} & \text{if } (\alpha - c_2 \epsilon_2) < a \end{cases} \quad (3.51)$$

with $a = 1.e - 5$ and $b = 1.e - 12$. a is the value of $(\alpha - c_2 \epsilon_2)$ where the blending begins and b is the asymptotic value of α^* when $c_2 \epsilon_2 \rightarrow \infty$, see fig. 3.5.

- ϵ_2 is frozen after 2 non-linear iterations, and the non-linear system is solved twice to machine precision.

Figure 3.5: Value of α^* in the modified artificial dissipation operator, eq. (3.47).

The last modification is the most important one for stability. Freezing the scheme turned out to be necessary in order to obtain a converged solution. As will be seen in section 4.3, this approach produces satisfactory results for the standard test case of the NACA0012 airfoil in transonic flow. However, it is noted that, even though the non-linear system is solved twice, the final solution still depends on the initial solution. A truly converged solution is obtained only when the system is solved four or five times. This issue was deemed too severe to consider further applications of this method.

3.3 TEMPORAL DISCRETIZATION

The second step of the method of lines is the integration with respect to time of the semi-discrete equations. Once the discretization of the spatial derivatives has been carried out, the set of conservation equations results in a system of ODEs in time that can be written as

$$\frac{d\mathbf{Q}(t)}{dt} + \mathbf{Res}(t, \mathbf{Q}(t)) = 0 \quad (3.52)$$

where the residual function $\mathbf{Res}(t, \mathbf{Q}(t))$ results from the discretization of the derivatives with respect to space present in the conservation equations, including boundary conditions, artificial dissipation and source term. Note that we take into account time-dependent boundary condition by explicitly setting the time as independent variable of the residual function. Now suppose that the solution at time t^n is $\mathbf{Q}^n = \mathbf{Q}(t^n)$. Integrating eq. (3.52) from $t = t^n$ to $t = t^{n+1}$ results in

$$\begin{aligned} \mathbf{Q}^{n+1} - \mathbf{Q}^n &= - \int_{t^n}^{t^{n+1}} \mathbf{Res}(t, \mathbf{Q}(t)) dt \\ \mathbf{Q}^{n+1} &= \mathbf{Q}^n - \int_{t^n}^{t^{n+1}} \mathbf{Res}(t, \mathbf{Q}(t)) dt \end{aligned} \quad (3.53)$$

Hence, if the integral in eq. (3.53) can be approximated with an appropriate quadrature rule, then solution $\mathbf{Q}^{n+1}$ is known. The particular choice of the quadrature rule leads to explicit or implicit schemes. Both types of schemes have been implemented and are described below.

For steady-state problems the time derivative should vanish, however, most of the algorithms that we use for the computation of steady flows are derived from unsteady, time accurate, schemes. Thus, the schemes for unsteady flow problems are described first.

3.3.1 Unsteady flows

We employ RK methods, which are multi-stage schemes. Defining $\mathbf{Q}^j = \mathbf{Q}(t^j)$ as the solution at the generic intermediate time level

$$t^j = t^n + c_j \Delta t, \quad j = 1, 2, \dots, s \quad (3.54)$$

where t^n is the time at the given time step n , $0 \leq c_j \leq 1$ and s is the number of stages, then the RK schemes for eq. (3.52) can be written as:

$$\mathbf{Q}^k = \mathbf{Q}^n - (\Delta t) \sum_{j=1}^{k^*} a_{kj} \mathbf{Res}(t^j, \mathbf{Q}^j), \quad k = 1, 2, \dots, s \quad (3.55)$$

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n - (\Delta t) \sum_{j=1}^s b_j \mathbf{Res}(t^j, \mathbf{Q}^j) \quad (3.56)$$

where a_{kj} , b_j and c_j are the Butcher coefficient of the scheme. The coefficients c_j denote the point in the time interval $t^n \rightarrow t + \Delta t = t^{n+1}$ at which the intermediate stage is evaluated. Again, the vectors $\mathbf{Q}^{n+1}$ and $\mathbf{Q}^n$ represent the solution at time level $n+1$ and n , respectively. The parameter k^* indicates whether the scheme is implicit or explicit. If $k^* \leq k-1$, then the solution at each stage k depends only on the solutions at the previous stages, which have already been determined, therefore the scheme is explicit. On the other hand, if $k \leq k^* \leq s$, the solution at stage k depends also on itself and possibly (if $k < k^* \leq s$) on the solution at the next stages; therefore the scheme is implicit. In this case, the complexity of the algorithm increases considerably since at each implicit stage k , it is necessary to solve a system of coupled non-linear differential equation. In order to do so, we employ Newton's method, which is described in detail in section 3.4.1.

The available explicit schemes are the classical 4th order Runge Kutta scheme (RK4, [91]) and Total Variation Diminishing (TVD) Runge Kutta schemes up till 3rd order [102, 38]. Explicit schemes are relatively easy to implement and they require a relatively small amount of memory. They suffer, however, from tight restrictions on the maximum allowed time step, i.e. they are conditionally stable. In fact, the maximum allowed Courant-Friedrich-Lewy (CFL) number for the TVD schemes is 1, while the classical RK4 scheme allows CFL numbers up to $2\sqrt{2}$, for inviscid flow problems [55].

Table 3.1: Available implicit schemes.

accuracy	name	implicit stages
1 st order	backward Euler	1
2 nd order	trapezoidal rule	1
3 rd order	ESDIRK ₃	3
4 th order	ESDIRK ₄	5
5 th order	ESDIRK ₅	7

Table 3.2: Butcher tableau for an ESDIRK scheme ($s = 4$).

0	0	0	0	0
c ₂	a ₂₁	a ₄₄	0	0
c ₃	a ₃₁	a ₃₂	a ₄₄	0
1	b ₁	b ₂	b ₃	a ₄₄
	b ₁	b ₂	b ₃	a ₄₄

On the contrary, implicit schemes require a substantial coding effort and have high-memory requirements. Their advantage is that they are unconditionally stable and therefore the choice of the time step is based on accuracy requirements only. They become preferable over explicit schemes when dealing with stiff problems, for which the physical time scales are very small. Typically, this happens in the context of URANS simulations. The implemented implicit schemes have an order of accuracy that varies from 1 to 5 and they are summarized in table 3.1, which also lists the number of implicit stages. The higher-order schemes are of the first stage Explicit, Singly Diagonally Implicit Runge-Kutta (ESDIRK) type [58, 7]. The Butcher tableau for the ESDIRK schemes has the form shown in table 3.2.

Note that: the first row contains all zeros, therefore the first stage is explicit; $k^* = k$, thus the remaining $(s - 1)$ stages are implicit; $U^s = U^{n+1}$, since the a_{sj} coefficients are equal to the b_j coefficients and $c_s = 1$. Therefore the general RK scheme of eqs. (3.55) and (3.56) can be written more explicitly for the ESDIRK schemes as

$$\mathbf{Q}^1 = \mathbf{Q}^n \quad (3.57)$$

$$\mathbf{Q}^k = \mathbf{Q}^n - (\Delta t) \sum_{j=1}^k a_{kj} \mathbf{Res}(t^j, \mathbf{Q}^j), \quad k = 2, 3, \dots, s \quad (3.58)$$

$$\mathbf{Q}^{n+1} = \mathbf{Q}^s \quad (3.59)$$

In practice, this same form is used also for the backward Euler and for the trapezoidal rule. The Butcher coefficients for the ESDIRK schemes used in the present work are listed in appendix D.

3.3.2 Steady flows

If flow problems are considered for which a steady-state solution is assumed to exist, then the time derivative in the semi-discrete conservation equations (3.52) would become zero. In this case, the system of conservation equations (3.52) reduces to:

$$\mathbf{Res}(\mathbf{Q}) = \mathbf{J} \left[\frac{\partial(\tilde{\mathbf{E}}^c - \tilde{\mathbf{E}}^v)}{\partial \xi} + \frac{\partial(\tilde{\mathbf{F}}^c - \tilde{\mathbf{F}}^v)}{\partial \eta} + \frac{\partial(\tilde{\mathbf{G}}^c - \tilde{\mathbf{G}}^v)}{\partial \zeta} - \tilde{\mathbf{S}} \right] = \mathbf{0}. \quad (3.60)$$

It is to be understood that all flux and source terms depend on the solution vector $\mathbf{Q}$, i.e. $\tilde{\mathbf{E}}^c = \tilde{\mathbf{E}}^c(\mathbf{Q})$, $\tilde{\mathbf{E}}^v = \tilde{\mathbf{E}}^v(\mathbf{Q})$, ..., $\tilde{\mathbf{S}} = \tilde{\mathbf{S}}(\mathbf{Q})$, and in turn the solution vector depends on the independent space variable ξ , i.e. $\mathbf{Q} = \mathbf{Q}(\xi, \eta, \zeta)$.

In order to solve (3.60) two approaches may be taken:

1. integration in pseudo-time is performed as a means of relaxation using an explicit RK method (see eqs. (3.55) and (3.56));
2. solve (3.60) directly using Newton's method; in analogy with the schemes for unsteady flows, we will refer to this as the implicit method.

For the first approach we use standard low storage RK schemes. These have the form:

$$\mathbf{Q}^0 = \mathbf{Q}^n \quad (3.61)$$

$$\mathbf{Q}^k = \mathbf{Q}^n - (\Delta t) a_{k,k-1} \mathbf{Res}(\mathbf{Q}^{k-1}), \quad k = 1, 2, \dots, s \quad (3.62)$$

$$\mathbf{Q}^{n+1} = \mathbf{Q}^s. \quad (3.63)$$

We employ two schemes of this type, one with four stages and one with five stages. The coefficients $a_{k,k-1}$ have been optimized for stability and not for accuracy. These coefficients for the four and five stage scheme read [54]:

$$a_{1,0} = \frac{1}{3}, \quad a_{2,1} = \frac{4}{15}, \quad a_{3,2} = \frac{5}{9}, \quad a_{4,3} = 1,$$

and

$$a_{1,0} = \frac{1}{4}, \quad a_{2,1} = \frac{1}{6}, \quad a_{3,2} = \frac{3}{8}, \quad a_{4,3} = \frac{1}{2}, \quad a_{5,4} = 1,$$

respectively. Furthermore, since time accuracy of the solution is not relevant, a global time step size is not required and therefore Δt can be different for each vertex of the grid, i.e. local time-stepping can be used. The magnitude of the local time-step Δt is determined based on the sum of the CFL condition in the three directions, as follows:

$$\frac{1}{\Delta t} = \frac{\sigma_\xi \lambda_\xi + \sigma_\eta \lambda_\eta + \sigma_\zeta \lambda_\zeta}{\text{CFL}} + \left(\frac{\sigma_{\max} \lambda_{\text{viscous}}}{\text{CFL}} + 1 \right) \quad (3.64)$$

where

$$\begin{aligned}\lambda_\xi &= |\mathbf{U}| + c \sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2} \\ \lambda_\eta &= |\mathbf{V}| + c \sqrt{\eta_x^2 + \eta_y^2 + \eta_z^2} \\ \lambda_\zeta &= |\mathbf{W}| + c \sqrt{\zeta_x^2 + \zeta_y^2 + \zeta_z^2}\end{aligned}$$

and

$$\lambda_{\text{viscous}} = (\mu^{\text{eff}}/\rho)(\xi_x^2 + \xi_y^2 + \xi_z^2 + \eta_x^2 + \eta_y^2 + \eta_z^2 + \zeta_x^2 + \zeta_y^2 + \zeta_z^2).$$

The variable c is the local speed of sound and the $\sigma_{\xi,\eta,\zeta}$ are dimensionless constants that depend on the discretization scheme in the corresponding direction. For the 1st and 2nd order scheme, $\sigma = 1$; for the 3rd order scheme $\sigma = 1.4$; for the 4th order scheme $\sigma = 2.0$, and for the 5th order scheme $\sigma = 2.5$. These values of σ have been determined from an eigenvalues analysis of the SBP operators. In particular, the σ 's are equal to the spectral radius of the given derivative operator (the SAT terms and the artificial dissipation have not been taken into account). The three $\lambda_{\xi,\eta,\zeta}$ are the spectral radii of the inviscid flux Jacobian in the corresponding direction (see appendix C) and they constitute the constraint of the time step due to the acoustic part of the equations. The term between brackets in eq. (3.64) is the contribution due to the viscous fluxes and the constant σ_{max} is the maximum of the three $\sigma_{\xi,\eta,\zeta}$. The default value of CFL is 0.8.

Explicit schemes are easy to implement and require low storage. Furthermore, when used as a relaxation method for steady flow problems, they are also very efficient in smoothing the high frequency errors [132]. Their convergence rate, however, becomes very slow once only the low frequency errors are present in the solution⁴. On the other hand, Newton's method, for which we refer again to section 3.4.1, should provide quadratic convergence *if* the initial solution is sufficiently close to the final one.

In the effort of taking advantage of the most desirable features of both approaches, a combination of the two is taken. First explicit pseudo-time integration is performed for a few iterations (typically between 10 and 100). When running RANS, during this initial explicit phase the turbulence variable $\tilde{\nu}$ is kept constant, so no relaxation is applied to the turbulence equation. Subsequently, Newton's method is used to converge the solution to the final solution. The aim of the explicit pseudo-time integration is to provide a suitable (and cheap) initial condition for Newton's method to start. The details of the solution method are given in the next section.

⁴ Note that this is the motivation behind all multigrid algorithms which, however, have not been exploited in the present work. We do use grid sequencing, though, to obtain a good initial guess for the target mesh.

3.4 SOLUTION METHOD: DESCRIPTION OF THE SOLUTION PROCEDURE

A general 3D computational method for 3D flows, which can handle multiblock grids around arbitrary configurations, has been developed. This code can run on (massively) parallel platforms, for which for reasons of load balancing, the blocks may be split during runtime in an arbitrary number of sub-blocks. A halo treatment of the newly created interfaces is used instead of an internal interface treatment, such that the results of the parallel algorithm are identical to those of the sequential algorithm. The spatial discretization schemes used are finite difference SBP-SAT schemes of order 2 to 5, described in section 3.2. For the NS equations, we use the same first derivative operator to compute the viscous fluxes and to differentiate the fluxes. Boundary conditions have been described in section 3.2.3. Flow and turbulence equations (if present) are fully coupled and solved simultaneously with the same scheme. The time discretization schemes, described in section 3.3, can be either explicit or implicit, and both types can be used for steady and unsteady flow problems. Note that all explicit schemes used have the same form, they are explicit RK schemes (although some implementation changes can and should be done for the low-storage schemes for the steady flow problems). Analogously, all the implicit schemes are similar since they involve the solution of a system of non-linear algebraic equations via Newton's method (either for eq. (3.58) or eq. (3.60)). Therefore the main difference between the solution of unsteady and steady flow problems is that a global time-step is used for the former, and a local time-step is used for the latter. This is exploited from an implementation point of view.

Explicit schemes are relatively straightforward to implement (see for example [91] for some common implementation strategies), and will not be described further. On the contrary, the solution of the system of non-linear equations resulting from an implicit time discretization via Newton's method requires considerably more attention. It is described in detail in section 3.4.1.

3.4.1 Newton's method

The system of non-linear equations arising from the implicit schemes (eq. (3.58) and eq. (3.60)) can be written generically as:

$$\mathbf{f}(\mathbf{x}) = 0 \quad (3.65)$$

where $\mathbf{f}$ is a vector of non-linear functions and $\mathbf{x}$ is the vector of independent variables. Since the conservation equations are valid for each vertex of the computational grid, system (3.65) has dimensions

$$\text{nDOFs} \times \text{size}(\mathbf{Q}) \quad (3.66)$$

where nDOFs is the total number of degrees of freedom per equation, which equals the number of grid points for a finite difference discretization; $\text{size}(\mathbf{Q})$ denotes the

dimension of the vector of conserved variables per vertex, i.e. 6 for the 3D RANS equations and 5 for all other 3D flow models considered in the present work (for 2D flows it is one less). Additionally, let us assume that each of the functions f_i of the system (3.65) depends on each of the independent variables grouped in $\mathbf{x}$, i.e. :

$$f_i = f_i(\mathbf{x}), \quad i = 1, 2, \dots, \text{nDOFs} \times \text{size}(\mathbf{Q}).$$

This assumption will not be true in practice (since the functions f_i are a result of a finite difference discretization with a discrete stencil), but it is needed for the formal derivation of the method.

The non-linear problem (3.65) can be solved iteratively by means of a Newton (also called Newton-Raphson) algorithm. Defining the current approximate solution $\mathbf{x}^n$ and assuming that this solution is sufficiently close to the root $\mathbf{x}^{\text{exact}}$ of the system of functions $\mathbf{f}$, a Taylor expansion around the root gives

$$\mathbf{f}(\mathbf{x}^{\text{exact}}) = \mathbf{f}(\mathbf{x}^n) + J(\mathbf{x}^{\text{exact}} - \mathbf{x}^n) + \text{h.o.t.} = 0 \quad (3.67)$$

where h.o.t. represents the higher-order terms of the expansion and J is the Jacobian matrix. The element in row i and column j of this square matrix is defined as the partial derivative of the function f_i with respect to the unknown x_j , evaluated at the state $\mathbf{x}^n$:

$$J = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}^n} \rightarrow J_{i,j} = \left. \frac{\partial f_i}{\partial x_j} \right|_{\mathbf{x}^n}. \quad (3.68)$$

The idea of Newton's method is to solve iteratively for an approximate solution $\mathbf{x}^{n+1}$ of the linear problem

$$J(\mathbf{x}^{n+1} - \mathbf{x}^n) = -\mathbf{f}(\mathbf{x}^n), \quad n = 0, 1, 2, \dots \quad (3.69)$$

The linear system eq. (3.69) arises from eq. (3.67) by neglecting the high-order terms and substituting $\mathbf{x}^{n+1}$ for $\mathbf{x}^{\text{exact}}$. Thus, Newton's method solves the non-linear problem via an iterative procedure in which during each step a linearized problem is solved. Note that if $\mathbf{f}$ was linear, the high-order terms of eq. (3.67) would be exactly zero, and only one iteration (3.69) would be necessary. For highly non-linear problems as the ones arising from the discretization of the RANS equations, the number of Newton iterations increases considerably.

In practice, first eq. (3.69) is rewritten in terms of the update

$$\delta \mathbf{x}^n = (\mathbf{x}^{n+1} - \mathbf{x}^n) \quad (3.70)$$

which becomes the unknown of the linear system

$$J \delta \mathbf{x}^n = -\mathbf{f}(\mathbf{x}^n), \quad n = 0, 1, 2, \dots \quad (3.71)$$

Then, after solving (3.71) for $\delta \mathbf{x}^n$, the update is applied to the current solution, possibly with some under-relaxation factor $\omega \leq 1$, as

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \omega \delta \mathbf{x}^n. \quad (3.72)$$

The process is repeated until the norm $\|\mathbf{f}(\mathbf{x}^{n+1})\|$ is sufficiently small.

Still four questions need to be addressed:

1. how to choose a suitable initial solution $\mathbf{x}^0$
2. how to calculate the Jacobian matrix J at each iteration
3. how to solve the linear system (3.71)
4. whether or not to use relaxation; this can be done explicitly via the relaxation parameter ω , and/or implicitly, by adding a term to the diagonal of the Jacobian.

Our approach to these interrelated issues is described in the next sections.

3.4.2 Initial solution and globalization strategy

The choice of the initial solution is crucial for the convergence of Newton's method. While for unsteady flow problems the solution at the current time step is usually a good enough approximation for the next time step, this is clearly not an option for steady flow problems. In fact, for steady problems other than RANS, we have found the following strategy to be very effective. First, the solution is initialized with the far-field state on a coarse grid. Then, the initial guess is obtained via grid sequencing for which we use a combination of explicit and implicit schemes. At each grid level, some iterations (typically 30 to 50) are performed with the explicit relaxation solver in order to get a suitable initial solution for Newton's method, which is used at each level as well. Upon convergence on one grid level, the solution is interpolated to the finer grid, explicit relaxation is performed and then the implicit method is used again to obtain a converged solution. The coarser grids are obtained by deleting every other grid point from the finer grids. Explicit relaxation damps the high-frequency components of the solution very quickly, and in our experience this technique combined with grid sequencing is sufficient to provide a good initial guess for the implicit solver. This is the case for both low and high-order schemes. For high-order schemes we typically use a low-order, not fully-converged, solution as initial solution.

However, for RANS this is simply not adequate and some other techniques must be employed. A popular choice is pseudo-transient continuation [59, 57, 39]. Following the description in [39], pseudo-transient continuation solves the steady flow problem $\mathbf{f}(\mathbf{x}) = 0$ through a series of problems

$$\mathbf{g}^l(\mathbf{x}) = \frac{\mathbf{x}^l - \mathbf{x}^{l-1}}{\Delta\tau^l} + \mathbf{f}(\mathbf{x}^l) = 0, \quad l = 1, 2, \dots, \quad (3.73)$$

which are derived from applying the method of lines and solving the original equation, plus the pseudo-time derivative, with backward Euler:

$$\frac{\partial \mathbf{x}(\tau)}{\partial \tau} + \mathbf{f}(\mathbf{x}(\tau)) = 0. \quad (3.74)$$

Each problem (3.73) is solved approximately for $\mathbf{x}^l$. The physical transient is followed when the timestep $\Delta\tau^l$ is sufficiently small and constant over the grid, and

the problems at each time-step are solved to a sufficiently low norm of the residual. This leads the solution process through a physically feasible sequence of states. Moreover, the Jacobian matrices associated with $\mathbf{g}^l$ are well-conditioned when the pseudo-time-step is small. $\Delta\tau^l$ is advanced from $\Delta\tau^0$ to $\Delta\tau^l \rightarrow \infty$ as $l \rightarrow \infty$, so that $\mathbf{x}^l$ approaches the root of the original problem $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, and Newton's method is recovered. Note that, in contrast to typical line-search globalization strategies (see section 3.4.5), pseudo-transient continuation does not require the reduction of the function norm $\|\mathbf{f}\|$ at each step; it can "climb hills" [39], meaning that this method can escape local minima in a function while searching for its root.

Newton's method applied to (3.73) gives:

$$\left(\frac{\mathbf{I}}{\Delta\tau^l} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \bigg|_{\mathbf{x}^{l,k}} \right) (\mathbf{x}^{l,k+1} - \mathbf{x}^{l,k}) = - \left[\frac{\mathbf{x}^{l,k} - \mathbf{x}^{l-1}}{\Delta\tau^l} + \mathbf{f}(\mathbf{x}^{l,k}) \right] \quad (3.75)$$

for Newton iteration $k = 0, 1, \dots$. Choosing $\mathbf{x}^{l,0} = \mathbf{x}^{l-1}$ (the most straightforward initial iterate), then the first correction step is

$$\left(\frac{\mathbf{I}}{\Delta\tau^l} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \bigg|_{\mathbf{x}^{l,k}} \right) (\mathbf{x}^{l,k+1} - \mathbf{x}^{l,k}) = -\mathbf{f}(\mathbf{x}^{l,k}). \quad (3.76)$$

If we were to solve (3.75) exactly for $\mathbf{x}^l$ ($k \rightarrow \infty$) and $\Delta\tau^l$ was constant over the grid, then we would follow the transient solution implicitly. A common practice, here adopted, is to advance in pseudo-time after only one Newton step (3.76). This means that, with pseudo-transient continuation, the original problem $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ is actually solved as:

$$\left(\frac{\mathbf{I}}{\Delta\tau^l} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \bigg|_{\mathbf{x}^l} \right) (\mathbf{x}^{l+1} - \mathbf{x}^l) = -\mathbf{f}(\mathbf{x}^l), \quad l = 0, 1, 2, \dots \quad (3.77)$$

The algorithm is completed once a criterion for the selection of the time-step is chosen. One choice is Switched Evolution-Relaxation (SER) [126], for which the time step grows in inverse proportion to the progress of the norm of the residual. Since the globalization strategy is formulated in terms of a pseudo-time, meaning that time accuracy is not required, a local pseudo-time step can be used. Therefore, in the present work we apply the SER strategy to define the CFL number:

$$\text{CFL}^l = \text{CFL}^{l-1} \cdot \phi \left(\frac{\|\mathbf{f}(\mathbf{x}^{l-2})\|}{\|\mathbf{f}(\mathbf{x}^{l-1})\|} \right), \quad (3.78)$$

and then determine the magnitude of the local pseudo-time step $\Delta\tau^l$ from eq. (3.64). The function $\phi()$ is a limiter that has been chosen in a rather conservative way favoring stability and robustness over performance:

$$\phi(a) = \begin{cases} 1.1 & \text{if } a > 1.1 \\ a & \text{if } 0.8 \leq a \leq 1.1 \\ 0.8 & \text{if } a < 0.8 \end{cases} \quad (3.79)$$

Note that with the current approach, the pseudo-transient continuation essentially consists in adding (another) time term to the diagonal of the Jacobian. This can easily be seen by comparing eq. (3.69) and eq. (3.77). Therefore, pseudo-transient continuation can be interpreted as an implicit damping method: it does not act on the relaxation parameter ω as an explicit damping would, instead, by adding a positive term to the diagonal of the Jacobian, it effectively reduces the update $\delta \mathbf{x}^n$. In some texts, this approach is referred to as a Damped Newton method.

Finally, we mention that we adopted the common choice of the pseudo-transient continuation for its robustness and simplicity although other globalization techniques are available in literature, like the dissipation-based continuation of [45] or, more recently, a physics-based continuation [14]. Also, SER is a quite popular CFL evolution strategy, but other options exist; for instance: ramping to a constant target CFL [76], exponential growth, the ‘expert’ system proposed in [127] and the residual difference method of [9]. The number of options available in literature is an indication that the evolution of determining an optimal value of CFL is still an open problem.

3.4.3 Efficient computation of the Jacobian

Computing the Jacobian may be a lengthy, time-consuming process. Its terms are the derivatives of the residual functions (possibly with the addition of a time-term) with respect to all conserved variables. The complexity of the Jacobian stems from the fact that the residual functions are expressions that form a challenge to differentiate analytically, even with the help of software for symbolic mathematics. The analytical differentiation is so error prone and cumbersome to implement that it is almost never done for practical problems. Instead, many other techniques for the evaluation of the Jacobian have been devised. Among them we mention the straightforward computation of derivatives via a finite difference approximation, the complex-step finite-difference method, automatic (also known as algorithmic) differentiation, and the dual number method. In the present work we use the dual number method, which is described next. For a comprehensive description and comparison of the aforementioned methods, the interested reader is referred to [30, 56] and citations therein.

3.4.3.1 Dual numbers

Following closely [56], we define first the set of dual numbers $\mathbb{D}$ as:

$$\mathbb{D} := \{a + \epsilon b \quad : \quad a, b \in \mathbb{R}, \quad \epsilon^2 \equiv 0\}. \quad (3.80)$$

Note the similarity with complex numbers, for which $(a + ib) \in \mathbb{C}$. However, one of the main differences is that for dual numbers $\epsilon^2 \equiv 0$, whereas $i^2 \equiv -1$ for complex numbers. As an example, we report some of the computational rules for dual numbers in table 3.3. Note that information always flows in one direction: from the real part to the dual part. The real part is never affected by the dual part. Now define the open subset $\mathbb{E} \subset \mathbb{R}$ and consider the function $f(z) : \mathbb{D} \rightarrow \mathbb{D}$ with

$z = a + \epsilon h$, $a \in \mathbb{E}$, $h \in \mathbb{R}$; and let $f(x) \in \mathbb{R} \forall x \in \mathbb{R}$. Then, the Taylor series expansion of $f(z)$ reads:

$$f(a + \epsilon h) = f(a) + \epsilon \left. \frac{df}{dz} \right|_a h + \frac{\epsilon^2}{2} \left. \frac{d^2 f}{dz^2} \right|_a h^2 + \mathcal{O}(\epsilon^2(\epsilon h^3)) \quad (3.81)$$

Since $\epsilon^2 \equiv 0$ by definition, this Taylor series expansion truncates at the first derivative. Now using the definition of the derivative and considering that the result is not influenced by the direction in the dual plane from which zero is approached, $j \in \mathbb{R}$ can be chosen such that

$$\left. \frac{df}{dz} \right|_a = \lim_{j \rightarrow 0} \frac{f(a + j) - f(a)}{j} \equiv \left. \frac{df}{dx} \right|_a.$$

Since $\mathbb{E}$ is an open set, $(a + j) \in \mathbb{E}$ for $j \rightarrow 0$. This means that both $f(a)$ and $f(a + j)$ are in $\mathbb{R}$, which leads to

$$\left. \frac{df}{dz} \right|_a = \left. \frac{df}{dx} \right|_a \in \mathbb{R} \quad \forall a \in \mathbb{E}.$$

Thanks to these properties, dual numbers can be used for computing derivatives: the derivative of the function f can be determined by evaluation of f with a dual number and considering the dual part only of the result divided by the step h , i. e.

$$\left. \frac{df}{dx} \right|_a \equiv \frac{\mathfrak{D}[f(a + \epsilon h)]}{h} \quad h \in \mathbb{R}.$$

The symbol $\mathfrak{D}[\cdot]$ means taking the dual part of the dual number argument, equivalent to $\mathfrak{I}[\cdot]$ for taking the imaginary part of a complex number. Note that all terms of the Taylor series expansion of order 2 and higher are exactly zero, thus no truncation error occurs and the value of h can be chosen arbitrarily. Choosing $h = 1$, we obtain

$$\left. \frac{df}{dx} \right|_a \equiv \mathfrak{D}[f(a + \epsilon)]. \quad (3.82)$$

Therefore, the derivative of the function f evaluated at the state a is the dual part of the result obtained evaluating the function f at the dual number state $(a + \epsilon)$. Note that eq. (3.82) does not involve any subtraction operation, therefore it is not affected by subtractive cancellation errors. Thus the dual number method yields an *exact* derivative.

So far we have assumed that the function f depends on a single variable only. The Jacobian J of eq. (3.68), however, is defined by means of partial derivatives. In particular, it is the derivative of the functions $f_i(x)$ with respect to all conserved variables x_j . In this situation, it is convenient to introduce the so-called vector mode of dual numbers. In this mode, the dual part of the dual number has more than one component and can be therefore considered as a vector. More explicitly, a dual number d with q dual parts is written as:

$$d = a + \epsilon \mathbf{b}, \quad \mathbf{b} = [b_1, b_2, \dots, b_q]^T. \quad (3.83)$$

Table 3.3: Computational rules for dual numbers [56]. Note that there is no influence from the dual part on real part.

Addition:	$(a + \epsilon b) + (c + \epsilon d) = (a + c) + \epsilon(b + d)$
Multiplication:	$(a + \epsilon b)(c + \epsilon d) = ac + \epsilon(ad + bc)$
Division:	$1/(c + \epsilon d) = 1/c - \epsilon d/c^2$
Math functions:	$\cos(a + \epsilon b) = \cos(a) - \epsilon b \sin(a)$ $\sin(a + \epsilon b) = \sin(a) + \epsilon b \cos(a)$ $\tan(a + \epsilon b) = \tan(a) + \epsilon b(\tan^2(a) + 1)$ $\exp(a + \epsilon b) = \exp(a) + \epsilon b \exp(a)$ $\ln(a + \epsilon b) = \ln(a) + \epsilon b/a$ $(a + \epsilon b)^{(c + \epsilon d)} = \exp[(c + \epsilon d) \ln(a + \epsilon b)]$

This can be expressed formally by introducing an extended definition of dual numbers as:

$$\mathbb{D}^q := \{a + \epsilon \mathbf{b} \quad : \quad a \in \mathbb{R}, \quad \mathbf{b} \in \mathbb{R}^q, \quad \epsilon^2 \equiv 0\}, \quad q \in \mathbb{N}, \quad (3.84)$$

where $\mathbb{N}$ represents the natural numbers (integers) without the zero.

In our case, the dimension of the dual number part will be equal to $\text{size}(\mathbf{Q})$, i. e. equal to the number of conserved variables per vertex, that is $q = 6$ for the 3D RANS equations and $q = 5$ for all other types of 3D flows (for 2D it is 4). Using this vector mode, it is possible to compute multiple derivatives with a single function evaluation. The advantage is that the real part of the result is computed only once, saving CPU time.

In order to show how this fits into the computation of the Jacobian, we need to introduce some more symbols. Define the symbol $\mathfrak{D}_j[\cdot]$ as taking the j^{th} component of the dual part of the dual number argument. In other words, $\mathfrak{D}_j[\cdot]$ extracts the coefficient b_j that multiplies the j^{th} component of the dual part vector in eq. (3.83). Furthermore, let the vectors $\mathbf{e}_j$ be the unit vectors of dimension q , with the j^{th} element equal to one, and the other $(q - 1)$ elements equal to zero:

$$\mathbf{e}_1 = (1, 0, \dots, 0)^T, \quad \mathbf{e}_2 = (0, 1, \dots, 0)^T, \quad \mathbf{e}_3 = (0, 0, 1, \dots, 0)^T \quad (3.85)$$

and so on. Now imagine that all components of the vector of independent variables $\mathbf{x}$ are dual numbers that belong to the set $\mathbb{D}^q$, and that all dual number components are initially set to zero. To explain the ideas, suppose $q = 5$. We focus on one vertex only, which, starting to count from 1, we identify with the 1D index k . Then we

perturb the dual part solution in vertex k , such that the *perturbed* solution vector $\mathbf{x}_p$ becomes

$$\mathbf{x}_p = \begin{bmatrix} \vdots \\ x_{1+5 \cdot (k-1)} + \epsilon \mathbf{0} = \rho_{k-1} \\ x_{2+5 \cdot (k-1)} + \epsilon \mathbf{0} = \rho_{k-1} u_{k-1} \\ x_{3+5 \cdot (k-1)} + \epsilon \mathbf{0} = \rho_{k-1} v_{k-1} \\ x_{4+5 \cdot (k-1)} + \epsilon \mathbf{0} = \rho_{k-1} w_{k-1} \\ x_{5+5 \cdot (k-1)} + \epsilon \mathbf{0} = \rho_{k-1} E_{k-1} \\ \\ x_{1+5 \cdot k} + \epsilon \mathbf{e}_1 = \rho_k + \epsilon \mathbf{e}_1 \\ x_{2+5 \cdot k} + \epsilon \mathbf{e}_2 = \rho_k u_k + \epsilon \mathbf{e}_2 \\ x_{3+5 \cdot k} + \epsilon \mathbf{e}_3 = \rho_k v_k + \epsilon \mathbf{e}_3 \\ x_{4+5 \cdot k} + \epsilon \mathbf{e}_4 = \rho_k w_k + \epsilon \mathbf{e}_4 \\ x_{5+5 \cdot k} + \epsilon \mathbf{e}_5 = \rho_k E_k + \epsilon \mathbf{e}_5 \\ \\ x_{1+5 \cdot (k+1)} + \epsilon \mathbf{0} = \rho_{k+1} \\ x_{2+5 \cdot (k+1)} + \epsilon \mathbf{0} = \rho_{k+1} u_{k+1} \\ x_{3+5 \cdot (k+1)} + \epsilon \mathbf{0} = \rho_{k+1} v_{k+1} \\ x_{4+5 \cdot (k+1)} + \epsilon \mathbf{0} = \rho_{k+1} w_{k+1} \\ x_{5+5 \cdot (k+1)} + \epsilon \mathbf{0} = \rho_{k+1} E_{k+1} \\ \vdots \end{bmatrix}$$

Then, a block column of the Jacobian can be filled out as:

$$J_{i,j} = \left. \frac{\partial f_i}{\partial x_j} \right|_{\mathbf{x}^n} = \mathfrak{D}_{(j \% q)}[f_i(\mathbf{x}_p^n)], \quad (3.86)$$

$$i = 1, 2, \dots, (q \times \text{nDOFs}), \quad 1 + q \cdot (k-1) \leq j \leq q + q \cdot (k-1) \quad (3.87)$$

where $j \% q$ means taking the integer remainder of the integer division j/q (essentially the $\text{mod}(j, q)$ function in many programming languages).

An example is given to clarify this procedure.

EXAMPLE

Consider the grid in fig. 3.6. This grid has 9 vertices in the ξ -direction and 17 in the η -direction for a total of $9 \times 17 = 153$ nDOFs. Let us assume we are solving the 2D Euler equations for a steady flow, therefore $q = \text{size}(\mathbf{Q}) = 4$, and the state vector $\mathbf{x}$ contains $q \times \text{nDOFs} = 4 \times 153 = 612$ elements. The f_i functions are the residual evaluation functions of eq. (3.60). Now starting to count from 1, consider the vertex in $l = 2$ and $m = 3$, where l is the index in the ξ -direction and m is the index in the η -direction. To this vertex we associate the 1D index $k = l + 9 * (m - 1) = 20$. We perturb the solution in

Figure 3.6: Grid with 9 vertices in the ξ -direction and 17 in η -direction, for a total of $9 \times 17 = 153$ nDOFs.

this vertex only, such that the perturbed state vector $\mathbf{x}_p^n$ at the given Newton iteration n reads:

$$\mathbf{x}_p^n = \begin{bmatrix} \vdots \\ x_{77} + \epsilon \mathbf{0} = \rho_{19} \\ x_{78} + \epsilon \mathbf{0} = \rho_{19} u_{19} \\ x_{79} + \epsilon \mathbf{0} = \rho_{19} v_{19} \\ x_{80} + \epsilon \mathbf{0} = \rho_{19} E_{19} \\ x_{81} + \epsilon \mathbf{e}_1 = \rho_{20} + \epsilon \mathbf{e}_1 \\ x_{82} + \epsilon \mathbf{e}_2 = \rho_{20} u_{20} + \epsilon \mathbf{e}_2 \\ x_{83} + \epsilon \mathbf{e}_3 = \rho_{20} v_{20} + \epsilon \mathbf{e}_3 \\ x_{84} + \epsilon \mathbf{e}_4 = \rho_{20} E_{20} + \epsilon \mathbf{e}_4 \\ x_{85} + \epsilon \mathbf{0} = \rho_{21} \\ x_{86} + \epsilon \mathbf{0} = \rho_{21} u_{21} \\ x_{87} + \epsilon \mathbf{0} = \rho_{21} v_{21} \\ x_{88} + \epsilon \mathbf{0} = \rho_{21} E_{21} \\ \vdots \end{bmatrix}.$$

Then, the block column of the Jacobian associated to this vertex, that is the columns going from $1 + 4 \times k = 1 + 4 \times 20 = 81$ to $4 + 4 \times 20 = 84$ of all 612 rows, are determined by:

$$J_{i,j} = \left. \frac{\partial f_i}{\partial x_j} \right|_{\mathbf{x}_p^n} = \mathfrak{D}_{(j \% 4)}[f_i(\mathbf{x}_p^n)], \quad (3.88)$$

$$i = 1, 2, \dots, 612, \quad 81 \leq j \leq 84. \quad (3.89)$$

For instance, the terms $J_{i,81}$, which are the derivatives of the residual functions with respect to the density ρ_{20} of vertex $k = 20$, will be:

$$J_{i,81} = \left. \frac{\partial f_i}{\partial x_{81}} \right|_{x^n} = \left. \frac{\partial f_i}{\partial \rho_{20}} \right|_{x^n} = \mathfrak{D}_1[f_i(x_p^n)], \quad i = 1, 2, \dots, 612. \quad (3.90)$$

To be more specific, if $i = 80$, then the Jacobian entry $J_{80,81}$ would represent the derivative of the energy residual of vertex 19, which is $f_{80} = \text{Res}_{\rho_{19}E_{19}}$, with respect to the density of vertex 20. In other words, eq. (3.90) states that we need to evaluate the functions f_i at the perturbed state x_p^n ; the result will be a dual number with $q = 4$ dual components, the first of which is the Jacobian entry $J_{i,81}$.

This process can be repeated for each vertex present in the computational grid, thus filling the Jacobian one block column at the time. This would mean that all f functions need to be evaluated a number of times equal to the total number of grid points in the grid. Furthermore, here it is assumed that all f functions depend on all x unknowns; this is not true in practice, and therefore a lot of zeros would be inserted in the Jacobian. In summary, although this procedure is correct in principle, it is very inefficient and prohibitively expensive in terms of memory (also it makes no sense to store the zeros). Instead, an efficient way of computing the Jacobian can be achieved with a so-called coloring technique, which is described in section 3.4.3.3. Coloring starts from the observation that the solution in one vertex is not influenced by the solution in all other vertices of the grid. In fact, only a small number of neighboring grid points influences its solution.

3.4.3.2 Extension to matrix-free algorithms

Newton's method requires the solution of the system of linear equations (3.69) at each iteration. Anticipating part of the contents of section 3.4.4, the iterative algorithm (GMRES) that we employ requires only matrix-vector products. This means that the Jacobian matrix need neither be explicitly formed nor stored (a good preconditioner is still needed though). Instead, the product of the Jacobian and an arbitrary vector $\mathbf{v}$ can be calculated with the dual number method as:

$$J\mathbf{v} = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{x^n} \mathbf{v} = \mathfrak{D}[\mathbf{f}(\mathbf{x}^n + \epsilon \mathbf{v})] \quad \text{where } (\mathbf{x}^n + \epsilon \mathbf{v}) = \begin{bmatrix} x_1^n + \epsilon v_1 \\ x_2^n + \epsilon v_2 \\ \vdots \end{bmatrix} \quad (3.91)$$

Note that here we used the first definition of dual number, eq. (3.80), with one dual part only. When a time derivative term is present, the matrix vector product simply becomes:

$$\left(\frac{I}{\Delta t} + J \right) \mathbf{v} = \frac{I}{\Delta t} \mathbf{v} + \mathfrak{D}[\mathbf{f}(\mathbf{x}^n + \epsilon \mathbf{v})] \quad (3.92)$$

Jacobian-free matrix-vector products are not always advantageous in terms of computing time. Without taking into account the preconditioner, if a small number

of inner iterations is needed, then it can actually be faster to perform the matrix-vector products rather than forming the Jacobian matrix and apply it repeatedly. However, a preconditioner is always needed for the problems to be tackled in the present work. If Jacobian and preconditioner are the same matrix, then such matrix has to be formed in any case, thus the use of a matrix-free algorithm is always going to be slower, i. e. of little use. In fact, in this case, the matrix-free algorithm would bring no advantages in terms of maximum memory required either: the memory requirements are dictated by the preconditioner and by its approximate inverse which, again, need to be computed anyway and they are both needed at the same time during the factorization process⁵. On the other hand, if Jacobian and preconditioner are different matrices, typically because they are based on different discretization schemes (higher for the Jacobian), then the matrix-free approach is usually preferable in terms of both computing time and memory requirements. In the present work, the matrix-free version of the algorithm is always used when the Jacobian is computed with higher order ($\geq 3^{\text{rd}}$) discretizations to limit the memory requirements. For the same reason, the preconditioner is always based on a lower order scheme, either the 2nd or the 1st order scheme. The matrix-free approach is used only when Jacobian and preconditioner are different.

3.4.3.3 *Coloring and memory requirements*

An efficient computation of the Jacobian starts by noting that each of the f_i functions does not depend on every element of $\mathbf{x}$. In fact, the set of residual functions in each vertex of the computational grid depend only on a relatively small number of independent variables. In other words, the Jacobian is typically a sparse matrix rather than a dense one. Coloring is a technique of graph theory which exploits this property of the Jacobian in order to minimize the number of times the f_i functions need to be evaluated. Before coloring is described, it is necessary to describe first the stencil and the sparsity pattern.

It is useful to think of the Jacobian in terms of blocks composed of $q \times q$ sub-matrices. These sub-matrices are a single block since they arise from grouping together the q conserved variables and the q conservation equations for each vertex of the grid. If there was only one conservation equation and one conserved variable per vertex, then the corresponding block Jacobian and non-block (scalar) Jacobian would be exactly the same. Note also that all the conservation equations have the same form: in each equation there is an inviscid flux and a viscous flux both of which need to be differentiated. Contrary to the inviscid flux, the viscous flux contains derivatives of the conserved variables, therefore in the differentiation of the viscous flux the derivative operator is used twice (non-compact schemes are used). The only exception is the mass conservation equation which does not have any viscous terms, but this is not exploited. The point is that the same stencil is used for all q conservation equations, thus it is useful to consider the Jacobian in terms of blocks. However, the stencil depends on the discretization scheme and, for

⁵ In-place factorization for sparse matrices is not allowed in the external library that we use to compute the factorization of the preconditioner.

a given scheme, it is not the same for all vertices of the grid. As seen in section 3.2, the SBP operators feature different stencils close to the boundaries. Furthermore, the higher the order of accuracy of the scheme, the wider the stencil.

The importance of the stencil is that it determines which of the neighboring vertices influence the solution of the central vertex, where the stencil is applied. Clearly the vertices outside the stencil do not influence the central vertex: if we perturb the solution in one of the vertices outside the stencil, such perturbation will not be ‘felt’ by the central vertex and its residuals will not change. This means that, on the contrary, the block columns corresponding to the vertices in the stencil will be the block non-zeros of the row of the Jacobian corresponding to the central vertex. Therefore, by evaluating the stencil in each grid point, we can determine the sparsity pattern (i. e. the location of block non-zeros) of the Jacobian matrix and its memory requirements.

As an example, fig. 3.7 shows the 2nd and 3rd order stencil in the interior and at a block boundary. The red dot is the central vertex whose neighbors are visualized. The black dots are the neighbors for the inviscid fluxes whereas the blue dots are the additional neighbors that arise from the discretization of the viscous fluxes. By neighbors we mean the vertices that participate in the stencil centered at the red dot. These are the vertices that influence the solution at the central vertex: a change in the solution in any of the black and blue dots will change the residuals of the red dot. Thus, the dots (all colors) will be the block non-zeros of the row of the Jacobian corresponding to the red central vertex.

Note that also terms of the artificial dissipation contribute to the stencil and should be taken into account when determining the sparsity pattern. However, since we are using non-compact schemes for the viscous terms, the contribution of artificial dissipation in fig. 3.7 is covered by the neighbors contributing to the viscous fluxes, which determine the width of the stencil. Figures 3.7b and 3.7d show the stencil at an internal block boundary. As mentioned in section 3.2.3, the SAT terms at this boundary require the conserved variables and the viscous flux computed in the corresponding vertex on the other side of the interface, the right (R) side in these figures. Thus, the extended stencil. Note that in this case the grid at the interface is conforming, meaning that the vertices residing on both sides of the interface have the same geometrical coordinates. They are, however, logically different degrees of freedom and they are therefore drawn separately. If the central red vertex is on a far-field or a wall boundary, there is simply no right side (R).

If we apply the third order stencil to all vertices of the grid in fig. 3.6, we obtain the sparsity pattern of fig. 3.8. Once the stencil is known and has been translated into non-zeros, the coloring of the Jacobian can be determined. In the context of graph theory, a sparsity pattern can be considered as a graph, that is a collection of objects connected by links. In this case the objects are the non-zeros blocks, and the link is the fact that two non-zeros blocks may belong to the same stencil. A graph coloring of such a collection is a subdivision of objects so that two objects of the same color do not share any common link. A suitable coloring for efficient Jacobian computation is a subdivision of the columns so that two columns of the same color do not share any common rows. In this way, vertices belonging to the same

Figure 3.7: Stencil (dependency graph) for the 2nd and 3rd order scheme in the interior and at a block boundary. The red dot is the central vertex. Black dots are the neighbors contributing to the inviscid fluxes and the blue dots are the neighbors arising from the discretization of the viscous fluxes.

color are independent of each other and their areas of influence, their stencils, do not overlap. In the present work the coloring of the Jacobian is determined using the coloring routines provided by the external library PETSc (Portable, Extensible Toolkit for Scientific computation; for a detailed description of the coloring algorithm see [56] and PETSc's documentation [6]). An example of the output of these routines is given in fig. 3.9, which represents the colors associated to the sparsity pattern of fig. 3.8. Note that the coloring is performed on a block base. For this example, if we were to follow the naive approach explained in section 3.4.3.1, in order to fill the Jacobian we would have to evaluate the f_i functions 153 times. Instead, exploiting the fact that vertices belonging to the same color are independent of each other, we can perturb the solution of the vertices belonging to the same color at the same time and then call the f_i functions. This way the f_i functions need to be evaluated only 17 times, once for each color. This number depends on the scheme, on the type of equations to be solved as well as on the number of dimensions of the problem, but it is independent of the actual grid size.

3.4.3.4 Implementation

This section combines the considerations about dual numbers and coloring and shows how the Jacobian is actually computed in our computational method.

The programming language adopted for the implementation of this method is C++ [106]. This language allows to define so-called template functions, which are general functions that have not yet been specialized for a particular data type. This allows the same template function to be used with several data types. As done in [56], a new data type for dual numbers has been introduced in the method. For this dual number data type (with arbitrary number of q dual parts), all mathematical operations used in the calculation of the residuals have been defined, see section 3.4.3.1. Moreover, all functions in the residual calculations have been defined as template functions. Only these two procedures are needed for the computation of the Jacobian with the dual number method. In this way, the functions used in the calculation of the residuals are defined only once, but they can be evaluated for both a real-valued floating point data type as well as for a dual number data type with q dual parts.

The computation of the Jacobian can be split in two phases, described in algorithms 3.1 and 3.2. In the preprocessing phase, algorithm 3.1, the sparsity pattern and the corresponding coloring of the Jacobian are determined based a dummy scalar Jacobian. This dummy is a scalar representation of the true block Jacobian. Since the coloring of the sparsity pattern is independent of the actual values of the elements of the matrix, at this stage the stencil is a dummy as well, meaning that it simply defines the non-zeros entries by filling the dummy Jacobian with 1. The memory requirements for the true Jacobian are that of the dummy Jacobian with the only difference that each entry of the dummy will be a $q \times q$ block of the true Jacobian. Once the dummy Jacobian has been constructed, it is passed to the coloring routines and the coloring is determined. Note that, if the grid does not contain parts in relative motion, these preprocessing steps need to be done only once.

Figure 3.8: Sparsity pattern of the 3rd order Jacobian corresponding to the grid in fig. 3.6, for an inviscid flow problem. All block non-zeros, which are 1325 in total, are indicated with a black dot.

Figure 3.9: Coloring of the Jacobian of fig. 3.8. Each of the 153 columns is associated with one of the 17 colors.

Algorithm 3.1 Preprocessing Jacobian

-
- 1: Apply stencil to all grid points and determine the block sparsity pattern
 - 2: Assemble the dummy scalar Jacobian
 - 3: Determine the colors n_c using the dummy
 - 4: Allocate the memory for the 'true' Jacobian
-

Algorithm 3.2 Compute Jacobian

-
- 1: **for** $c = 1$ to n_c **do** // Loop over all colors
 - 2: **for all** vertices $\in c$ **do** // Loop over all vertices belonging to this color
 - 3: $\mathbf{Q} = \{\rho + \epsilon \mathbf{e}_1, \rho u + \epsilon \mathbf{e}_2, \rho v + \epsilon \mathbf{e}_3, \rho w + \epsilon \mathbf{e}_4, \rho E + \epsilon \mathbf{e}_5, \rho \tilde{v} + \epsilon \mathbf{e}_6\}^T$ // Perturb the solution in this vertex
 - 4: **end for**
 - 5: Evaluate all $f_i(\mathbf{x}_p)$ // Compute residuals in all vertices of the grid
 - 6: **for all** affected vertices $\in c$ **do** // Loop over vertices with non-zero dual part
 - 7: $J_{i,j} = \mathcal{D}_{(j\%q)}[f_i(\mathbf{x}_p^n)]$ // Store the non-zero result according to eq. (3.86)
 - 8: **end for**
 - 9: **end for**
-

After the preprocessing steps, the actual computation of the Jacobian can take place. As shown in algorithm 3.2, first a loop over all colors is performed. For each vertex in this color, the value of 1.0 is assigned to the different components of the dual part of the dual number, for each of the independent variables with respect to which the derivative needs to be determined (line 3). Then the f_i functions are called and the dual part of the output variables, evaluated at line 7, contains the desired derivative. The loop starting at line 6 needs to be clarified. The coloring technique allows to fill more than one block column of the Jacobian at the same time. However, not all rows of the columns belonging to a given color are non-zeros. Fortunately we can determine which ones are non-zero from the sparsity pattern and, at line 6 of algorithm 3.2, we loop over the corresponding vertices. They are called “affected vertices” because their residual is affected by the perturbed solution of the vertex associated with the color which the column belongs to.

3.4.4 Solution of the system of linear equations

Assuming that the Jacobian has already been constructed, at each Newton iteration the system of linear equations (3.71) or (3.77) needs to be solved, at least approximately. The system of linear equations is solved iteratively with a Krylov subspace method [98]. An iterative method is used rather than a direct method simply because the memory and CPU time required by the latter are prohibitively large for 3D problems. The Krylov method that we employ in this work is the Generalized Minimal RESidual (GMRES) [99]. It is a quite popular choice and many implementations are readily available. One of these is part the library PETSc [6], a library

that provides also solvers for systems of non-linear equations, which we use as well.

A good iterative solver is not enough. The Jacobian matrix resulting from CFD problems is typically badly conditioned and, for RANS, it arises from poorly scaled equations. Regarding this last issue, if no counter measures are taken, the residual norm will be dominated by the worst residual component. For example, the residual of the equation representing the turbulence model can be orders of magnitude larger than the residuals of the mean flow equations. With such large differences, the algorithm will try to reduce only the residual of the turbulence equation while leaving almost untouched, if not letting increase, the residual of the other components of the system. However, in the case of the SA model, a simple scalar scaling of SA's discretized equation is effective in scaling the different components of the residual to similar magnitude [17, 13]. As suggested in [17, 13], we nondimensionalize $\rho\tilde{v}$ with respect to $10^3\mu_\infty$. This corresponds to defining the reference value of $\rho\tilde{v}$ as its estimated maximum value. In fact, a maximum eddy viscosity ratio, μ^t/μ (see eq. (2.53)), of $\approx 10^3$ is typical for aerodynamics applications; more precisely, this value is typical for Reynolds numbers in the range $Re = 10^6 \rightarrow 10^7$ [13]. At the same time, the mean flow variables are nondimensionalized with their corresponding far-field value. Note that the computational method solves the dimensional equations, thus this scaling is carried out while computing the linear system. Once the solution of the linear system is obtained, it is scaled back to dimensional values. However, the CFL number in eq. (3.78) is increased based on the scaled residual norms. This simple scaling strategy was found to greatly improve the robustness of the iterative solution procedure, although some further gains can be expected from more sophisticated techniques.

Additionally, a preconditioner (also called preconditioning matrix) is used to improve the convergence characteristics of the Krylov subspace method. To achieve this improvement, the preconditioner must cluster the eigenvalues of the matrix, leading to a reduction of the condition number of the matrix [56]. In order for such a preconditioner to be efficient, computing its inverse has to be computationally less expensive than computing the inverse of the original Jacobian matrix. Literature offers several strategies for constructing an efficient preconditioner (see for instance [34]). In the present research we use a parallel preconditioner based on the Additive Schwarz Method (ASM), in which the underlying method requires an exact or inexact inversion of local submatrices. Here an inexact inversion is preferred and it is accomplished with the block version of the incomplete lower-upper factorization with level p (BILU(p)) (see, for example, [98]) available in PETSc. The blocks are composed of the flow field unknowns plus, if present, the turbulence unknown at each node. In particular, the general problem

$$Ax = b \quad (3.93)$$

is rewritten as

$$P^{-1}Ax = P^{-1}b \quad (3.94)$$

for left preconditioning. The matrix A denotes the Jacobian whereas P is the preconditioner matrix that is approximately inverted as:

$$P = LU - R \approx LU \quad \rightarrow \quad P^{-1} \approx L^{-1}U^{-1}, \quad (3.95)$$

where R represents the error in the factorization. If zero level of fill is allowed, L and U have the same (block) sparsity pattern as the lower and upper parts of P , respectively. When running in parallel, the problem (3.93) is split into multiple domains, giving for each domain the problem:

$$P_i^{-1} A_i x_i = P_i^{-1} b_i, \quad i = 1, 2, \dots, n_d \quad (3.96)$$

where n_d is the total number of domains. We associate one domain with each processor, such that n_d equals the total number of processors being used. Although not done by default, the user may decide to further divide each domain into sub-domains associated to one processor. Then, the factorization is applied to the sub-domains resulting in a cheaper (both in terms of memory and of computing time) but less accurate preconditioner. Furthermore, Additive Schwarz methods can employ overlapping domains to improve the quality of the preconditioner. The performance of this ASM preconditioner as implemented in PETSc has been studied extensively in [40, 39] in the context of low-order schemes. It has been shown that while increasing the amount of overlap reduces the number of necessary Krylov iterations, the lowest overall CPU time is obtained by using an overlap between 0 and 2. We chose 1. Note that when running on one processor this ASM method reduces to a standard BILU(p) preconditioner.

Before calculating the incomplete factorization, a reordering is applied to the preconditioner in order to reduce the memory requirements and to improve the quality of the preconditioner. In fact, the reordering permutations are designed to reduce fill-ins during the Gaussian elimination, and when this happens it is likely that the ILU factorizations resulting from dropping small terms will be more accurate. Furthermore, reordering is most beneficial when relatively accurate factorizations (high levels of fill) are computed [98]. It was found that the *nested dissection* reordering algorithm, implemented in PETSc, was the most reliable and is used throughout the present work, although the *Reverse Cuthill-McKee* method gave slightly better performance on small grids and low levels of fill (see for example [98] for a description of these reordering methods). Typical values of fill go from 1, for unsteady flow problems, to 8 for steady flow problems.

As can be seen in fig. 3.7, the number of non-zeros per row increases very rapidly as the order of accuracy gets higher and the stencil becomes accordingly wider. This is especially true for 3D problems. In fact, the memory requirements for the higher order schemes are so large that building and storing the higher order preconditioner becomes prohibitively expensive for all but the smallest grid problems. Therefore in the present work the preconditioner is always computed based on the 2nd order scheme. For the same reason, a matrix-free version of the GMRES is employed when the Jacobian is computed with orders of accuracy larger than 2nd. Finally, a summary of typical settings of the linear solution method is presented

Table 3.4: Settings of the linear solver.

solver	GMRES
subspace size	400
initial solution	zero
relative tolerance	10^{-4} to 10^{-8}
BILU level of fill	1 to 8
reordering	nd
ASM overlap	1
sub-domains per processor	1

in table 3.4. In the context of implicit schemes, the choice of the solver parameters is notoriously non-trivial and problem dependent [40]. Nevertheless, the values in table 3.4 were found to be a good starting point for further optimization.

3.4.5 Backtracking line-search

The last aspect that must be addressed for the solution of the linear system (3.71) or (3.77) is the choice of the update parameter ω of eq. (3.72). We employ the backtracking line-search implemented in PETSc. This line search finds the minimum of a polynomial (quadratic or cubic) fitting of the L_2 norm of the function $\mathbf{f}$, $\|\mathbf{f}(\mathbf{x})\|$. If this fit does not satisfy the conditions for progress, the interval shrinks and the fit is reattempted until ω is below a threshold value $\omega_{\min}$ or until a maximum number of attempts (by default 40) has been made. The implementation follows closely the algorithm explained in [22, p. 325].

The line-search can be used in combination with pseudo-transient continuation, however, the former requires the residual norm to decrease at each iteration, while the latter does not. For steady RANS, the initial solution is typically very far from the converged solution, and the residual norm must be allowed to grow in order to escape the local minima. Therefore, for these problems the maximum number of attempts is limited to 1, such that the line-search is almost disabled while pseudo-transient continuation is active. It has been found that this single iteration of the line-search improves the stability of the iterative solution procedure by limiting large changes in the residual norm. Disabling it completely usually brings faster convergence for very well resolved problems, but this is not always the case and, again, robustness is preferred over performance. On the other hand, for Unsteady RANS the solution at the current time-step is a good approximation of the solution at the next time-step and pseudo-transient continuation and line-search can coexist. For laminar flow and for inviscid flow problems, pseudo-transient continuation is usually not necessary (remember that initial guesses are always obtained via grid sequencing) and the (cubic) line-search is fully enabled.

3.5 ASPECTS OF THE IMPLEMENTATION

This section describes some miscellaneous aspects of the implementation of the computational method which have not been addressed so far. These are load balancing, the computation of the distance from solid walls, donor search procedure for sliding mesh interfaces, and the computation of forces.

3.5.1 Load balancing

The computational method that has been developed in the present work can run on (massively) parallel platforms. Load balancing is achieved via domain decomposition: the computational blocks may be split during runtime in an arbitrary number of sub-blocks that are assigned to different processors. A halo treatment of the newly created interfaces (fig. 3.10) is used instead of an internal interface treatment, such that the results of the parallel algorithm are identical to those of the sequential algorithm. When doing this, care must be taken in the computation

Figure 3.10: Standard treatment of multiblock numerical interfaces. Layers of unknowns called halos, here indicated by dashed lines, are transferred between sub-blocks. (Reproduced from [80]).

of the Jacobian in order to avoid duplication of degrees of freedom. The halos must not be included since they are a copy of other vertices. For the same reason, the vertices residing on a newly created interface, which is duplicated and assigned to two different processes, must be considered only once. The approach adopted here is that the process with the lower Message Passing Interface (MPI) rank is responsible for the vertices at the interface. Its responsibility consists in computing the Jacobian entries, the residual and the solution corresponding to these vertices.

3.5.2 Computation of the wall distance and sliding mesh donor search procedure

The distance to the wall is needed for the SA turbulence model and interpolation coefficients must be determined for the sliding mesh interfaces between rotor and stator. Moreover, due to the relative motion of the different parts, these interpolations must be repeated every stage of each time step, at least for the sliding mesh interfaces. Thus an efficient search algorithm is a necessity. Both the calculation of the wall distance and the sliding mesh donor search are minimum distance searches: for a given vertex, find the surface element (quadrilateral) that minimizes the distance to that vertex, and, more precisely, determine the coordinates of the point within the quadrilateral that minimizes such distance. The algorithm employed in the present work to carry out this minimum distance search consists of two steps. First a relatively small set of candidate quadrilaterals is determined via an Alternating Digital Tree (ADT) search algorithm [3] (also known as alternating binary tree search). The search criterion in the ADT is not the actual distance between the given point and a surface element, rather it is an approximation of the surface element that is easier to compute: the search criterion is the distance from the given point to the Cartesian bounding box of the surface element. Then, in the second step of the process, a loop is performed over the candidate quadrilaterals. In this loop the actual distance is calculated and the closest quadrilateral, the one that minimizes such distance, is determined.

Note that in the case of sliding meshes, the minimum distance search algorithm is not optimal. A more efficient algorithm would be a containment search, that is a search that checks whether or not a point is within a set of bounding boxes. An example of this type of search and its implementation can be found in [56] in the context of overset grids. However, this has not been exploited in the present work and the minimum search algorithm is used both for the calculation of the wall distance and for the sliding mesh donor search⁶.

An exhaustive search of N points is one which checks each point in the set against the search criteria. Such a procedure has linear complexity since searching through a set of $2N$ points will take twice as long. The ADT structure allows to search such a point set with $\mathcal{O}(\log N)$ complexity [3]. Clearly, when the search has to be repeated several times, the performance gain is substantial.

3.5.2.1 ADT terminology

The ADT is a data structure that is particularly well suited to search collections of objects. In our case the objects that make up the tree, called the nodes, are Cartesian bounding boxes. The term node generically refers to any storage location in the tree (root, branch, leaf, etc.). As shown in fig. 3.11, a binary tree is a nested structure of nodes in which each node can have at most two children. To each node is associated

⁶ We mention that another (popular) option exists for an approximate and efficient calculation of the wall distance: the so-called equation based wall-distance calculation [121, 119], for which the approximate wall distance is obtained by solving a Poisson-like equation. While proved successful for finite-volume schemes, this option has not been adopted due to numerical difficulties experienced when solving the Poisson-like equation with the SBP-SAT schemes.

Figure 3.11: Perfectly balanced binary tree. The numbering of the nodes follows from the generation process. Node 0 is the root node and nodes 7 to 14 are the terminal leaves.

Figure 3.12: Guaranteed minimum distance d_g and possible minimum distance d_p from point P to the Cartesian bounding box (dashed blue line) of a quadrilateral (black solid line).

a Cartesian bounding box that contains the bounding boxes of all its descendants. The tree starts from a single root node whose associated bounding box contains the entire domain spanned by the tree. The tree ends at the terminal leaves that, by definition, do not have any children. The terminal leaves are constituted by the Cartesian bounding boxes of the actual quadrilaterals in the surface mesh that needs to be searched. These are the quadrilaterals that reside on solid walls (for the wall distance computation) or on the sliding mesh interfaces. A Cartesian bounding box $[a, b]$ of a quadrilateral is defined as the smallest Cartesian region (the edges of the box are aligned with axis of the coordinate system) which can contain the quadrilateral (fig. 3.12). Thus it is fully specified by only two vertices, a and b , which can be defined as a point in a 6-dimensional space.

The parallel algorithm in the present work, described in sections 3.5.2.2 and 3.5.2.3, has been derived from the one explained in [124]. The main difference between the approach in [124] and the one adopted here regards the communication strategy. In [124], each processor generates its own tree based on the locally owned grid, and communication is performed during the search. In the present work instead, each processor involved in the search generates and stores a copy of the entire tree. Communication is performed only before the tree is build so that each processor stores the entire surface mesh to be searched and for which the ADT needs to be build. Therefore communication is performed only once in a preprocessing step. Although this causes a memory overhead, the CPU time required for the search is reduced because it does not involve communication. Furthermore, the number of nodes in the tree is small compared to the number of vertices in the grid, hence the memory required to store the entire tree(s) is orders of magnitude smaller than the memory required to store the preconditioner and its factorization for solving the linear systems, see section 3.4.4.

3.5.2.2 *Tree generation*

The generation of the tree starts with building the bounding boxes of the quadrilaterals composing the local surface mesh. These bounding boxes constitute the terminal leaves of the tree and the generation of the tree consists in building their parental relationships. In order to distinguish the bounding boxes associated to the terminal leaves from the larger bounding boxes associated to the parent nodes, from now on we are going to refer to the former group as terminal bounding boxes. Note that at this stage communication has already been performed, so that each processor stores the entire surface mesh needed for the ADT. Once all the terminal bounding boxes have been constructed, the bounding box of the root node is determined by simply looping over all terminal bounding boxes and taking the component-wise maximum and minimum of $\mathbf{b}$ and $\mathbf{a}$, respectively.

We now begin bisecting this bounding box with planes normal to the 6-coordinate axes (3 components of $\mathbf{a}$ plus 3 components of $\mathbf{b}$), changing the direction of the partitioning plane at each level in the binary tree. Before each bisection the terminal bounding boxes associated with the node are sorted such that an ordered set of bounding boxes is obtained; the criterion for the ordering is the coordinate of the direction normal to which the bisection is going to take place. Since the points may not be distributed evenly, each node is logically rather than geometrically bisected to improve the balance of the ADT. This means that the bisection is performed based on the number of elements rather than on their geometrical properties. This partitioning strategy ensures that the left and right children of each node contain approximately the same number of objects (the difference is at most 1). Therefore, the tree remains balanced, even when the data is not uniformly distributed in the search space. The disadvantage of this approach is that the bounding box of each node of the tree must be stored, which doubles the memory requirements of the ADT.

So for the first bisection, the bisection of the root node, the terminal bounding boxes are ordered according to the x -component of $\mathbf{a}$ and a set of ordered terminal bounding boxes is obtained. The first half of the set is assigned to the left child and the second half of the set is assigned to the right child. The bounding box of the left child and right child are determined by looping over the terminal bounding boxes in their sub-set and taking the component-wise maximum and minimum of vectors $\mathbf{b}$ and $\mathbf{a}$, respectively. Level 1 of the tree has just been created. For the second and third bisection, the terminal bounding boxes are ordered according to the y -component and z -component of $\mathbf{a}$, respectively. Then the components of $\mathbf{b}$ will be used for the ordering and subsequently, if the tree allows more than 7 levels (6 plus the root), the components of $\mathbf{a}$ will be used again, and so on in a cyclic manner. Continuing with this process, at each bisection, 2^l new nodes (sub-trees, branches) of the tree are created, in which l is the level in the binary tree that is being created. For each of the 2^l nodes, the bounding box is determined. The sub-set of terminal bounding boxes associated to each node at the current level are now ordered with respect to the new bisection direction. Then the bisection takes place: the first half of the terminal bounding boxes in the sub-set of a node at the current level are assigned to its left children and the second half to its right children. After, the bounding boxes of the newly created nodes are determined, the new bisection takes place, and so on. This process is carried out until three or less terminal bounding boxes are associated to a node. If two terminal bounding boxes are associated to the node, then one is assigned to the left child and one to the right child; the two children become the terminal leaves of the tree since each of them is associated with only one terminal bounding box. Instead, if a node contains three terminal bounding boxes, the left child is assigned two bounding boxes and the previous situation is recovered. The third terminal bounding box is assigned to the right child which will then become a terminal leaf of the tree. The reason for mixing a terminal bounding box and a leaf is that then the number of leaves can be determined a priori. At the end of this process, all bounding boxes of the quadrilaterals in the surface mesh constitute a terminal leaf of the tree and their parental relationships are fully determined.

3.5.2.3 Search procedure

The search for the quadrilateral that minimizes the distance to a given point P is carried out in two steps. The first step is the ADT search. In this step, the search criterion is the distance from P to the bounding box of the given quadrilateral rather than the distance from P to the given quadrilateral itself. The ADT search outputs a relatively small set of terminal leaves, called candidates.

The tree search procedure is outlined in algorithm 3.3 ([8]). First the current value of the minimum distance $d_{\min}$ is initialized to a very large value and the root node is marked for search. Then the search front advances one level at a time, starting from the root. At each level the tree is traversed *horizontally* and the nodes that have been marked for search in the previous level are considered. For each of the children of the considered nodes, the possible minimum distance d_p and

Algorithm 3.3 Determine candidates for minimum distance quadrilateral via ADT.

```

1:  $d_{\min} = 10^{20}$  // Init. current minimum distance
2: Mark the root node for search
3: for  $l = 0$  to  $(n_l - 2)$  do // Loop over all but the last tree levels
4:   for all marked nodes  $\in l$  do // Loop over the marked nodes in this level
5:     for  $c = 1$  to  $2$  do // Loop over this node's two children
6:       Calculate  $d_p$  // Calc. the possible minimum distance from P to this child's BB
7:       Calculate  $d_g$  // Calc. the guaranteed minimum distance
8:       if  $d_p \leq d_{\min}$  then
9:         if (this child is a terminal leaf) then
10:          Store terminal leaf as candidate
11:         else // This child has children itself
12:           Mark this child for search at the next level
13:         end if
14:          $d_{\min} = \min(d_g, d_{\min})$  // Update the current minimum distance
15:       end if
16:     end for
17:   end for
18: end for

```

the guaranteed minimum distance d_g from the point P to the child's bounding box are determined, see fig. 3.12. At line 8 of algorithm 3.3, d_p is checked against the currently stored minimum distance $d_{\min}$. If $d_p \leq d_{\min}$, then the branch corresponding to this child should be further investigated; thus, the child is either marked for search at the next level, or, if the child is a terminal leaf, it is stored as candidate. The nodes -and the corresponding branches- that do not meet the requirement $d_p \leq d_{\min}$ are excluded from the search. At line 14, the current value of the minimum distance is possibly updated with d_g (note that $d_g \geq d_p$).

Once the candidates have been found via the ADT search, the second step of the search procedure can commence. In this step, a loop is performed over all candidates and the actual distance from the point P to the quadrilaterals corresponding to the candidates is determined. This is accomplished by solving a constrained optimization problem which finds the coordinates of the point within the quadrilateral that minimizes the distance to the given point P . Solving a constrained optimization problem is clearly more costly in terms of CPU time than computing the distance between a point and a box. That is why it is convenient to first obtain a small set of candidates via the ADT of bounding boxes. In order to solve the optimization problem, the quadrilateral is mapped to the orthogonal space (u, v) shown in fig. 3.13, in which both u and v range from 0 to 1, thus the constraints for the optimization are: $0 \leq (u, v) \leq 1$. Defining P^* as the point with coordinates (u^*, v^*) within the quadrilateral that minimizes the distance to the given vertex P ,

Figure 3.13: Transformation from the real to the orthogonal space in which the quadrilateral is mapped such that the interpolation weights of the neighbors of P^* can be determined.

the interpolation weights of neighbors N_0 to N_3 are given by the following equations:

$$w_0 = (1 - u^*)(1 - v^*), \quad (3.97)$$

$$w_1 = u^*(1 - v^*), \quad (3.98)$$

$$w_2 = u^*v^*, \quad (3.99)$$

$$w_3 = (1 - u^*)v^*. \quad (3.100)$$

Note that for a sliding interface, P and P^* coincide, and the distance to the quadrilateral is zero. The interpolation weights are needed only in the context of sliding mesh donor search, as they are required for the imposition of the boundary conditions at the sliding interface, see section 3.2.3.6. When rotational rather than translational sliding meshes are used, cylindrical coordinates must be used. Their implementation is explained next.

3.5.2.4 Transformation to cylindrical coordinates

The search algorithm just explained is also used when sliding mesh interfaces are rotational (like in a 3D rotor-stator interaction problem). However, in this case, tree generation and the search procedure have to be carried out in cylindrical coordinates. This is necessary because, when the sliding mesh is rotational, the intended (logical) neighbors might not be the neighbors that define the closest quadrilateral in the Cartesian coordinate system, as can be seen in fig. 3.14. There, a sliding interface composed of 6 quadrilaterals per side is rotating around the z axis. Note the high aspect ratio of the external quadrilaterals; high aspect ratios are particularly common in the sliding mesh interfaces of a turbomachinery stage due to the presence of the hub and casing walls. We wish to determine the interpolation weights of the neighbors of P that lies on the other side of the interface (dashed quadrilaterals). The intended (logical) neighbors of P are vertices A and B because they lie on the same circumferential trajectory as P does. Thus the interpolated solution in P should depend only on the solution in A and B . However, the algorithm described so far would assign the higher interpolation weights to vertices C and D because,

Figure 3.14: Rotational sliding mesh rotating around the z axis. The intended (logical) neighbors of P are vertices A and B because they lie on the same circumferential trajectory as P does. However, in Cartesian coordinates, P is closer to vertices C and D which, as an outcome of the search algorithm described so far, would wrongly become the neighbors of P with the higher interpolation weights. (Reproduced from [128].)

in Cartesian coordinates, P is actually closer to C and D than to A and B . Note that, if the external quadrilaterals had an even higher aspect ratio, P would lie outside the quadrilateral $ABCD$ and yet different neighbors would be found. To avoid this problem, cylindrical coordinates must be used to determine the correct neighbors and the corresponding interpolation weights.

In order to perform the transformation from the Cartesian coordinates (x, y, z) to cylindrical coordinates $(r, \theta, \bar{z})$, the Cartesian coordinates are first transformed to a local system of Cartesian coordinates $(\bar{x}, \bar{y}, \bar{z})$, for which the $\bar{z}$ axis corresponds to the axis of rotation. Then, the cylindrical coordinates $(r, \theta, \bar{z})$ are obtained from the local Cartesian system $(\bar{x}, \bar{y}, \bar{z})$ as follows (see fig. 3.15a):

$$r = \sqrt{\bar{x}^2 + \bar{y}^2}, \quad (3.101)$$

$$\theta = \text{atan2}(\bar{y}, \bar{x}), \quad (3.102)$$

$$\bar{z} = \bar{z}. \quad (3.103)$$

Here atan2 is the C++ more robust implementation of the inverse tangent $\text{atan}(\bar{y}/\bar{x}) = \arctan(\bar{y}/\bar{x})$ ⁷. Note that the cylindrical coordinates must be made non-dimensional by dividing r and $\bar{z}$ by a constant which has the dimension of a length (for instance,

⁷ The atan2 function takes into account the signs of both arguments, and places the angle in the correct quadrant. For instance, the call $\text{atan}(1/0)$ will raise an error due to the division by zero, whereas $\text{atan2}(1, 0)$ gives the correct result of $\pi/2$.

the maximum between $r_{\max} - r_{\min}$ and $\bar{z}_{\max} - \bar{z}_{\min}$), such that the search algorithm described before can be reused; the distances will not be straight lines but they will still be a valid criterion for searching.

Furthermore, the function atan2 produces results in the range $(-\pi, \pi]$. Thus, care must be taken when considering a quadrilateral that crosses the $-\pi, +\pi$ line (see the gray-filled quadrilateral in fig. 3.15a), where the function atan2 is discontinuous. Without any special treatment, such quadrilateral would be misrepresented in the tree. In fact, in the cylindrical space in which the tree lives, the quadrilateral with vertices of both sides of the $-\pi, +\pi$ line would wrongly cover almost the entire region that ranges from $-\pi$ to $+\pi$, as shown in fig. 3.15b. To overcome this problem, the vertices on the $-\pi$ side are duplicated to the $+\pi$ side (by adding 2π) to form a quadrilateral with vertices on the $+\pi$ side, and at the same time, the vertices on the $+\pi$ side are duplicated to the $-\pi$ side (by subtracting 2π) to form a second quadrilateral (fig. 3.15c). In this way there are two quadrilaterals in cylindrical coordinates that represent the same quadrilateral in Cartesian coordinates. However, there will be only one quadrilateral in cylindrical coordinates that minimizes the distance to a given point P , because P is defined only in $(-\pi, \pi]$ region.

3.5.3 Computation of forces

The computation of forces consists in performing an integration of a particular quantity over a surface. For instance, the force on an airfoil in inviscid flow is given by

$$\mathbf{F} = - \int_S (p - p_\infty) \mathbf{n} dS \quad (3.104)$$

where p is the local pressure, p_∞ is the pressure at the far-field and $\mathbf{n}$ is the outward pointing normal to the closed surface S of the airfoil. Since we are working in a discrete framework, the integral is approximated by a summation over the grid points present on the surface, which, due to the transformation to curvilinear coordinates, corresponds to a plane in the computational (ξ, η, ζ) space. In order to obtain the correct result, the value of the integrand in each grid point must be multiplied by an integration weight that depends on the spatial schemes being used and on how close the grid point is to the boundary of the surface. The integration weight depends in fact on the norms of the schemes (see section 3.2) in the given point. The norm matrices are also the integration rules of the appropriate order of accuracy. The scheme can be different for the two directions making up for the surface, thus the weights are obtained by taking the product of the two norms. For example, if the surface resides on a $\xi = \text{constant}$ plane, the force is given by:

$$\mathbf{F} = - \sum_k^{N_\zeta} \sum_j^{N_\eta} P_\zeta^k P_\eta^j (p^{j,k} - p_\infty) \begin{bmatrix} \xi_x \\ \xi_y \\ \xi_z \end{bmatrix} / J \quad (3.105)$$

where N_ζ and N_η are the number of grid points in the ζ and η direction, respectively; P_ζ^k and P_η^j are the norms in the two directions in the vertex (j, k) of the

Figure 3.15: Care must be taken when a quadrilateral crosses the $-\pi, \pi$ line, as the gray-filled quadrilateral in fig. 3.15a does. Without any special treatment, that quadrilateral will be misrepresented in the cylindrical coordinates system, covering almost the entire domain, fig. 3.15b. This is solved by duplicating its vertices and by constructing two corresponding quadrilaterals in the cylindrical system, as in fig. 3.15c.

surface. The components of the vector $[\xi_x, \xi_y, \xi_z]^T$ are the metric terms, which are divided by the determinant of the Jacobian of the transformation, J . Remember that away from the boundaries of the computational block the norm is simply 1. Thus, when an edge of the surface has been duplicated (due to the load balancing algorithm explained in section 3.5.1 and shown in fig. 3.10), the corresponding weights in the direction normal to the duplicated edge are 1 except for the vertices lying on the edge itself, for which the weight is 1/2. In this way the result does not depend on the number of processors. The weights explained above are used whenever an integration of a field quantity must be performed.

3.6 SUMMARY OF THE SOLUTION STRATEGY

For the steady flow test cases the set of non-linear algebraic equations is solved using the non-linear solver library of PETSc [6]. This library requires the Jacobian matrix of the spatial residual, which is computed via dual numbers [30] and appropriate coloring of the vertices of the grid, for which the PETSc routines are used. A matrix-free version of the algorithm has also been implemented, which is always used for the high-order ($\geq 3^{\text{rd}}$) space discretization schemes. In this case, the matrix-vector products are computed using dual numbers as well. Initial guesses are obtained via grid-sequencing, and, for each grid level, some iterations of explicit relaxation are performed before the implicit solver is started. The solution of the systems of linear equations needed by the non-linear solution algorithm is obtained by PETSc's ASM preconditioned GMRES. The ASM preconditioner uses the Block ILU solver on the local domains. Thus, only when running on 1 processor, the ASM preconditioner reduces to a standard Block ILU preconditioner. The preconditioner is always computed with the 2nd order scheme in order to limit the memory requirements. Implicit time integration schemes of the ESDIRK type have been implemented, for which the resulting systems of non-linear equations are solved using a slightly adapted version of the algorithm explained above for the case of steady flow. The non-linear solver of PETSc features a backtracking line-search which we use as well. However, only one iteration of the back-tracking is allowed for RANS for steady flow (the line-search is almost disabled), whereas up to the default 40 iterations are allowed for URANS, and for all other types of flows. For (U)RANS, we employ the common pseudo-transient continuation as a globalization strategy, in which the local-time step is based on a SER algorithm.

In summary, we use a high-order pseudo-transient matrix-free inexact Newton-Krylov-Schwarz algorithm. The main solver settings are given in table 3.5.

Table 3.5: Settings of the implicit solver for the different flows.

	(U)RANS	Inviscid, Laminar
Linear solver	GMRES	GMRES
Preconditioner type	ASM-BILU(p)	ASM-BILU(p)
Preconditioner scheme	2 nd order	2 nd order
Matrix-free	if $\geq 3^{\text{rd}}$ order	if $\geq 3^{\text{rd}}$ order
Pseudo-transient cont.	SER	none
Line-search backtracking	up to 1 iteration for steady, up to 40 iterations for unsteady	up to 40 iterations

4

NUMERICAL RESULTS

This chapter presents numerical results obtained with the discretization schemes and computational method described in the preceding chapters¹. The analyzed test cases have been selected such that the numerical results cover a wide range of flow conditions and modeling approaches. The test cases are ordered with increasing complexity of the governing equations: first inviscid flows, then laminar viscous flows, subsequently turbulent flows analyzed with DNS, then LES, and finally RANS. In particular, in sections 4.1 to 4.3 the method is applied to inviscid flow test cases. A laminar flow test case is analyzed in section 4.4. Section 4.5 shows our results for a popular DNS test case, the Taylor-Green vortex. In section 4.6, the LES modeling approach is applied to a problem characterized by laminar-to-turbulent transition induced by the separation of the laminar boundary layer. Results of numerical simulations with RANS are shown in section 4.7 and section 4.8. Two other RANS test cases are presented in appendix E as verification of the implementation of the SA turbulence model. Finally, in section 4.9 the parallel efficiency of the method is analyzed. Most of the test cases analyzed have been proposed at the second and third International Workshop on High-Order CFD methods, see [1, 2].

Emphasis is put on the accuracy and efficiency of the schemes. In the description to follow, the variable h represents the grid length scale defined as $h = 1/\sqrt{n\text{DOFs}}$ for 2D problems, and $h = 1/\sqrt[3]{n\text{DOFs}}$ for 3D problems; $n\text{DOFs}$ is the total number of degrees of freedom per discretized equation and equals the number of grid points for a finite difference discretization. The grid length scale h gives an indication of the fineness of the grid: the smaller h , the finer the grid. A benchmark code, named TauBench (available at [1]), was run on each particular machine to provide a normalization such that simulations made on different machines can be compared. To this end, the total computational time, multiplied by the number of processors used to run the simulation, is divided by the computational time needed by TauBench to perform a fixed number of multigrid cycles on the given machine². The non-dimensional result is referred to as “work units”, and reads:

$$\text{work units} = n_p t_{\text{run}} / t_{\text{bench}} \quad (4.1)$$

where n_p is the number of processors used to run the simulation, t_{run} is the total computational time of the simulation, and t_{bench} is the computational time of TauBench defined before. The work units provide a normalized measure of the computational cost. Typical values of the computational time of TauBench are between 6 and 10 seconds on state-of-the-art machines.

¹ Part of the content of this chapter is based on references [122, 36, 35].

² TauBench was run using the following parameters: `mpirun -np 1 ./TauBench -n 250000 -s 10`.

4.1 EULER VORTEX

This classic test case aims at characterizing the solver's ability to preserve vorticity in a 2D inviscid flow. Furthermore, this test case is also used here to verify the correct implementation of the computational method by checking that the design accuracy of the schemes is obtained.

The flow is inviscid and is governed by the unsteady 2D Euler equations. The flow consists of a 2D vortex transported by a uniform flow across a rectangular computational domain of dimensions $(x, y) = (0, L_x) \times (0, L_y)$. The initial solution of the vortex, centered in (x_c, y_c) and superimposed onto a uniform flow, is given by the following equations:

$$\rho_0 = \rho_\infty \left(\frac{T_0}{T_\infty} \right)^{1/(\gamma-1)}, \quad u_0 = U_\infty + \delta u, \quad v_0 = \delta v, \quad T_0 = T_\infty - \delta T \quad (4.2)$$

where, for $r \in [0, \infty)$,

$$\begin{aligned} \delta u &= -(\beta U_\infty) \frac{y - y_c}{R} e^{-r^2/2} \\ \delta v &= (\beta U_\infty) \frac{x - x_c}{R} e^{-r^2/2} \\ \delta T &= \frac{0.5}{C_p} (\beta U_\infty)^2 e^{-r^2} \end{aligned}$$

and

$$C_p = \frac{\gamma}{\gamma - 1} R_{gas}, \quad r = \frac{\sqrt{(x - x_c)^2 + (y - y_c)^2}}{R}.$$

The pressure follows from the ideal gas law and reads:

$$p_0 = p_\infty \left(\frac{T_0}{T_\infty} \right)^{\gamma/(\gamma-1)}.$$

The variable R represents the characteristic radius of the vortex while β defines its strength. The subscript 0 indicates the initial solution, whereas the subscript ∞ denotes the solution of the unperturbed (uniform) flow. Furthermore, $\gamma = 1.4$ is the constant specific heat ratio and $R_{gas} = 287.87 \text{ J/(kg K)}$ is the gas constant; the variables $U_\infty = Ma_\infty \sqrt{\gamma R_{gas} T_\infty}$ and $\rho_\infty = p_\infty / (T_\infty R_{gas})$ are the velocity and density of the unperturbed flow, respectively. The (analytic) solution is steady in the frame of reference moving with the free-stream. As the problem is almost linear for small β , no artificial dissipation is necessary in that case.

Since the vortex should be transported without distortion, the L^2 -norm of the error can be defined as

$$L_{\text{error}}^2 = \left[\frac{\sum_{i=1}^{N_{\text{tot}}} \text{error}_i^2 J_i}{\sum_{i=1}^{N_{\text{tot}}} J_i} \right]^{1/2}, \quad (4.3)$$

with

$$\text{error}_i = \phi_i^{\text{numerical}} - \phi_i^{\text{analytic}}, \quad i = 1, 2, \dots, N_{\text{tot}}, \quad (4.4)$$

where ϕ is one of the conserved variables and $\phi^{\text{numerical}}$ is the numerical solution. The analytic solution is computed from eq. (4.2), N_{tot} is the total number of grid points, and J_i is the determinant of the metric Jacobian (see eq. (2.32)). On a uniform Cartesian grid, eq. (4.3) simplifies to

$$L_{\text{error}}^2 = \left[\left(\sum_{i=1}^{N_{\text{tot}}} \text{error}_i^2 \right) / N_{\text{tot}} \right]^{1/2}. \quad (4.5)$$

This test case has been run in two different configurations: on a stationary grid, and on a moving grid.

4.1.1 Static grid

The “slow vortex” configuration is computed (see test case C1.4 in [2]), for which $\text{Ma}_\infty = 0.05$, $\beta = 1/50$ and $R = 0.005$ m. The flow direction is parallel to the positive x -axis. Moreover, we assume $L_x = 0.1$ m, $L_y = 0.1$ m, $x_c = 0.05$ m, $y_c = 0.05$ m, $p_\infty = 10^5$ Pa and $T_\infty = 300$ K. Note that a dimensional problem is solved, see [2]. It is possible to solve the non-dimensional version, which leads to the same conclusions. The flow is periodic in both directions, therefore no boundary closure is needed and the internal discretization is used throughout the domain. Hence, a scheme with global accuracy of order p (with $p > 2$), which features a $2(p-1)$ order discretization in the interior (see section 3.2.2), should provide an $2(p-1)$ order solution and it will be referred to as such in the remainder of this section. The solution is advanced in time with the classical RK4 scheme up to 50 periods ($t_{\text{final}} = 50U_\infty/L_x$). For this case the time step is dictated by stability requirements. Reducing the time step by a factor of two did not change the results significantly. The solution is computed on four structured uniform grids consisting of one square computational block, discretized with 33^2 , 65^2 , 129^2 and 257^2 vertices, respectively.

Figure 4.1 and 4.2 show the L_{error}^2 of the two velocity components (u, v) as a function of h and work units. The 2^{nd} order scheme does not show the design accuracy, in fact the error is almost constant. This indicates that the asymptotic range has not yet been reached and therefore the 2^{nd} order scheme requires much finer grids than the ones used. Note also that there is no difference between the error on the second finest and finest grid for the 8^{th} order scheme when running in double precision. It turns out that the reason for this behavior is finite arithmetic. When running the computation in quadruple precision, the error reduces according to design accuracy of the scheme. Most likely for the same reason the error reduction of the 6^{th} order scheme is lower than expected when going from the second finest to the finest grid. However, a quadruple precision computation is approximately two orders of magnitude slower than the corresponding double precision computation and has therefore only been carried out for the 8^{th} order scheme. As shown in fig. 4.2, the higher-order schemes provide a more accurate solution at a smaller

computational cost. The quadruple precision computation has been omitted from this graph.

(a) x-velocity component u

(b) y-velocity component v

Figure 4.1: Euler vortex on stationary grid: L^2 -norms of the velocity components (u, v) as function of the grid size.

(a) x-velocity component u

(b) y-velocity component v

Figure 4.2: Euler vortex on stationary grid: L^2 -norms of the velocity components (u, v) as function of the work units. The quadruple precision 8th order case is not shown to avoid excessive scaling.

Figure 4.3: Euler vortex problem on sliding meshes. $N_x = 65, N_y = 65$, 5th order solution (density contours).

4.1.2 Moving grid

This test case is used here to verify the design accuracy of the interpolation operators. In this configuration the vortex travels across two rectangular computational domains of dimensions $(x,y) = (-L_x, 0) \times (-0.5L_y, 0.5L_y)$ and $(x,y) = (0, L_x) \times (-0.5L_y, 0.5L_y)$, see figure 4.3. We set $Ma_\infty = 0.5$, $\beta = 1/(2\pi)$, $R = 0.1\text{m}$, $L_x = 1\text{ m}$, $L_y = 1\text{ m}$, $x_c = -0.5\text{ m}$, $y_c = 0.0\text{ m}$. The flow is periodic in the y direction, therefore no boundary closure is needed at the moving interface where we use the interpolation operators. The analytic solution is used as boundary data at the left and right boundaries. While the left block is fixed, the right block is oscillating with a frequency of $f = U_\infty/L_x$ and with an amplitude of $0.5L_y$. Note that, in contrast to the stationary configuration, we did not assume periodicity in the x -direction, thus the boundary closure is needed in the x -direction and the design accuracy of the schemes changes accordingly. The solution is advanced in time with an explicit 3rd order TVD Runge-Kutta scheme for 1 characteristic time ($t_{\text{final}} = L_x/U_\infty$), such that the vortex has to travel across the moving interface once. The time step is chosen such that the error due to the temporal discretization is negligible with respect to the error due to the spatial discretization. In this configuration, 3000 time steps were used.

The convergence rates of the density error are reported in table 4.1; similar values were obtained for the other conserved variables. The global accuracy of the schemes is verified, hence the interpolation operators show their design accuracy.

Table 4.1: Euler vortex crossing a sliding interface: L^2_{error} norms and convergence rates for the density error obtained with the different schemes. N ($= N_x = N_y$) is the number of grid points in the two directions for each block.

N	2 nd order		3 rd order	
	L^2_{error}	Conv. rate	L^2_{error}	Conv. rate
33	1.176e-4	—	3.365e-5	—
65	3.216e-5	1.87	4.128e-6	3.03
129	8.042e-6	2.00	4.971e-7	3.05

N	4 th order		5 th order	
	L^2_{error}	Conv. rate	L^2_{error}	Conv. rate
33	6.372e-5	—	5.623e-5	—
65	5.369e-6	3.57	2.032e-6	4.79
129	3.618e-7	3.89	5.802e-8	5.13

4.2 A ROTOR-STATOR INTERACTION PROBLEM

To further illustrate the applicability of the treatment of the sliding interface, we consider a linear cascade problem. Originally designed at the Duke University [28], the cascade consists of a 1&1/2 stages (stator-rotor-stator) compressor. This is a model two-dimensional compressor in which the flow is inviscid (for testing purposes). It has been selected to demonstrate that the method is capable of simulating such a geometry.

In order to reduce the complexity of the grid and the computational costs, the geometry has been scaled down to a 3-4-5 configuration from the original 16-20-25 configuration. In the scaled configuration the (non-dimensional) pitches for the three rows of blades are 1.0-0.75-0.6 (see fig. 4.4a), whereas for the original configuration they are 1.0-0.8-0.64. The coarse grid, used to obtain an initial solution, is shown in fig. 4.4. The fine mesh consists of 54 computational blocks and approximately 650,000 vertices. The mesh topology has been chosen in order to comply with the simplifying assumption of section 3.2.3.6; that is, only one block on each side of the interface and 1 : 1 matching. However, this forced the introduction of sub-faces in the computational blocks involved in the sliding interfaces, see the red and blue blocks in fig. 4.4b. Sub-faces, also known as T-junctions, are not accounted for in the stability of the SBP-SAT schemes of 3rd order and higher, and are a relatively recent topic of research. A possible solution has been proposed in [79], however, it has not been implemented yet in our code and no ad-hoc treatment has been employed.

Characteristic far-field boundary conditions are used at the inlet and outlet, and the values of the corresponding primitive variables are given in table 4.2³. The

³ As for the Rotor37 test case, see section 4.8, the values of the primitive variables at the inlet and outlet planes have been obtained with the code Sumb [124].

Table 4.2: Boundary conditions at the inlet and outlet planes for the linear cascade problem (non-dimensional values).

	ρ	u	v	p
inlet	0.8895	0.55	0.743	1.2375
outlet	1.31	0.37	0.4	2.0

(a) Coarse mesh **(b)** 1:1 matching at the interface and sub-faces (right boundary of the red block)**Figure 4.4:** Coarse mesh for the linear cascade problem.

flow is assumed to be periodic in the y direction. Furthermore, the rotor travels at a non-dimensional speed of 1.25 in the vertical direction. The flow is inviscid and subsonic, and the solution is advanced in time until the initial disturbances are smoothened out and a periodic solution is obtained.

We computed the 2nd and 3rd order solution. For this test case, the 4th and 5th order schemes were not stable. The cause of this behavior is under investigation but we believe the most likely culprit is the non-stable handling of sub-faces.

The history of the y -component of the forces acting on the first blade of each of the three rows, during the periodic regime of the 2nd order run, is shown in fig. 4.5. The other blades exhibit a similar but phase-shifted behavior (not shown here). The corresponding fast Fourier transform (obtained with the open-source software Octave [27]) is shown in fig. 4.6. As the rotor travels at a speed of 1.25 (note that for this test case all quantities are non-dimensional) and has a pitch of 0.75, a blade of the rotor passes the blades of the two stators with a frequency of

$$f_R = 1.25/0.75 \approx 1.67.$$

At this frequency, the amplitude of the Fourier transform of the signals S1 and S2 is the largest. The harmonics of the blade-passing frequency f_R can be observed as well. The spectrum of the signal of the rotor blade, R, is richer than that of the two

Figure 4.5: History of the y-component of the forces acting on the first blade of each row (non-dimensional values). From top to bottom: first stator (S1), rotor (R), second stator (S2).

Figure 4.6: Fourier transform of the y-component of the forces acting on the first blade of each row (all quantities are non-dimensional), see fig. 4.5. The main frequencies f_{S1} , f_{S2} and f_R , and their harmonics are indicated with dashed vertical lines. The frequencies f_C^i are combination of the harmonics of the main frequencies: $f_C^1 = 2f_{S1} - f_{S2}$, $f_C^2 = f_{S2} - f_{S1}$ and $f_C^3 = 2f_{S1} + f_{S2}$.

stator blades. In fact, an observer moving with the rotor sees the blades of the first stator with a frequency of

$$f_{S1} = 1.25/1 = 1.25,$$

and the blades of the second stator with a frequency of

$$f_{S2} = 1.25/0.6 \approx 2.1.$$

As can be seen in fig. 4.6, the two frequencies f_{S1} and f_{S2} , together with f_R and the harmonics of all three of them, are among the main frequencies of the rotor signal R . Furthermore, in the rotor signal R we can also observe other important frequencies, indicated with f_C^1 , f_C^2 and f_C^3 . These frequencies are a combination of the harmonics of the frequencies f_{S1} and f_{S2} ; in particular,

$$f_C^1 = 2f_{S1} - f_{S2},$$

$$f_C^2 = f_{S2} - f_{S1},$$

$$f_C^3 = 2f_{S1} + f_{S2}.$$

Figure 4.7a shows the instantaneous pressure field at a particular time during the periodic regime for the 3rd order discretization. Figure 4.7b shows the pressure contours and the mesh close to the rotor-stator interface (the same considerations hold for the stator-rotor interface). The mesh is clearly not grid-line continuous, but the solution is.

(a) Global view (animation)

(b) Zoom-in at the sliding interface: the solution is smooth, the mesh is not grid-line continuous at this moment in time.

Figure 4.7: Pressure contours of the 3rd order instantaneous solution of the stator-rotor-stator calculation. The flow is from left to right, the rotor is moving upward.

Figure 4.8: Coarse grid (145×129 vertices) for the transonic inviscid flow over the NACA0012 airfoil.

Figure 4.9: Convergence history (norm of the vector of residuals) for the NACA0012, $Ma_\infty = 0.8$, $AoA = 1.25^\circ$, 5th order scheme with shock capturing, fine grid (577×513). Grid sequencing using 7 grids.

4.3 NACA0012

The NACA0012 airfoil is computed for a transonic inviscid flow, defined by the following far-field Mach number and angle of attack (AoA):

$$Ma_\infty = 0.8, \quad AoA = 1.25^\circ.$$

For this test case (identified with C1.2 in [2]), we generated a fine O-grid of 577×513 vertices using the hyperbolic grid generation capabilities of the commercial software Pointwise [52]. The far-field is located at 1000 chords and the trailing edge is sharp. The vertices are clustered on the suction side, in particular close to the shock region, as shown in fig. 4.8. Initial guesses are obtained via grid sequencing, for which the coarser grids are obtained by deleting every other grid line from the finer grid (regular coarsening). Two sets of results have been computed, one with our “standard” artificial dissipation operators, see section 3.2.4, and one with the modified (for shock capturing) dissipation operators described in section 3.2.4.1. The convergence history for the 5th order scheme on the fine grid, with shock capturing, is presented in fig. 4.9, which clearly shows the grid sequencing (on 7 grid levels) as well as the convergence of the residual to machine zero. Note that for the fine grid, when using the modified artificial dissipation operators, the non-linear system arising from Newton’s method is solved twice. As mentioned in section 3.2.4.1, this is done to limit the dependence of the final solution on the initial one (the modified operator is frozen after two non-linear iterations).

Figure 4.10 shows the pressure coefficient, C_p , as a function of the c_2 constant in eq. (3.47) on a coarse mesh (128×48). The pressure coefficient is a non-dimensional quantity defined as

$$C_p = \frac{p - p_\infty}{\frac{1}{2} \rho_\infty U_\infty^2}, \quad (4.6)$$

where p_∞ is the free-stream static pressure, ρ_∞ is the free-stream density and U_∞ the free-stream velocity. Note that for this test case the artificial dissipation coefficient α in eq. (3.50) is set to 0.04 for all schemes and grids. The oscillations close to the shock, which are typical for the high-order dissipation operators, appear to be effectively reduced, although this effect is more pronounced for the 2nd order scheme. The same behavior is observed for the fine grid, fig. 4.11a. Furthermore, on the fine grid a weaker shock appears on the pressure side of the airfoil.

Figure 4.10: NACA0012, $Ma_\infty = 0.8$, $AoA = 1.25^\circ$: chordwise distribution of the pressure coefficient obtained for different values of the artificial dissipation constant c_2 (equation 3.47), coarse grid (145×129). The black line corresponds to $c_2 = 0$ (no shock capturing).

In fig. 4.11b we show the distribution of the quantity $(\hat{s} - \hat{s}_\infty)/\hat{s}_\infty$, with $\hat{s} = p/\rho^\gamma$, on the suction side of the airfoil. We refer to the quantity $(\hat{s} - \hat{s}_\infty)/\hat{s}_\infty$ as ‘relative entropy’. The entropy, s , is related to the relative entropy by the following equation:

$$\frac{\hat{s} - \hat{s}_\infty}{\hat{s}_\infty} = \exp\left\{\frac{s - s_\infty}{c_v}\right\} - 1, \quad \hat{s} = p/\rho^\gamma. \quad (4.7)$$

Equation (4.7) can be obtained from the entropy changes for a calorically perfect gas:

$$\frac{s - s_\infty}{c_v} = \ln \frac{p/p_\infty}{(\rho/\rho_\infty)^\gamma} \rightarrow \frac{p/p_\infty}{(\rho/\rho_\infty)^\gamma} = \frac{\hat{s}}{\hat{s}_\infty} = \exp\left\{\frac{s - s_\infty}{c_v}\right\}. \quad (4.8)$$

Just upstream of the shock, the 2nd order scheme exhibits the oscillations with the higher amplitude. Right downstream of the shock, all schemes show almost no oscillations. The jump of the relative entropy across the normal shock on the

Figure 4.11: NACA0012, $Ma_\infty = 0.8$, $AoA = 1.25^\circ$: chordwise distribution of the surface pressure coefficient and of the relative entropy on the airfoil, fine grid (577×513), $c_2 = 2.0$ (equation 3.47), 2nd to 5th order scheme.

suction side can be estimated from the jump condition of the entropy ([5]):

$$\frac{s_2 - s_1}{c_v} = \ln \frac{p_2/p_1}{(\rho_2/\rho_1)^\gamma} \quad (4.9)$$

where

$$\frac{p_2}{p_1} = \frac{2\gamma Ma_1^2 - (\gamma - 1)}{\gamma + 1}, \quad (4.10)$$

$$\frac{\rho_2}{\rho_1} = \frac{(\gamma + 1)Ma_1^2}{(\gamma - 1)Ma_1^2 + 2}. \quad (4.11)$$

The subscripts 1 and 2 denote the solution on the upstream and downstream side of the shock, respectively. Substituting eq. (4.7) into eq. (4.9), we obtain the jump for the relative entropy:

$$\frac{\hat{s}_2 - \hat{s}_1}{\hat{s}_1} = \frac{p_2/p_1}{(\rho_2/\rho_1)^\gamma} - 1, \quad (4.12)$$

Assuming that the flow is isentropic before the shock, $s_1 = s_\infty$, and substituting the value of the upstream Mach number $Ma_1 = 1.3841$, which is obtained from the 5th order accurate flow solution, eq. (4.12) yields

$$\frac{\hat{s} - \hat{s}_\infty}{\hat{s}_\infty} = 0.0156.$$

A very similar value is observed in fig. 4.11b.

The contours of the pressure and the relative entropy, obtained for the fine grid with the 5th order scheme and the modified artificial dissipation operators, are shown in fig. 4.12. The 2nd order scheme gives very similar contours (not shown

Figure 4.12: NACA0012, $Ma_\infty = 0.8$, $AoA = 1.25^\circ$: fine grid results (577×513) with shock capturing ($c_2 = 2.0$), 5th order scheme.

here) and differences are almost not discernible at this scale. Some oscillations can still be observed close to the shock on the suction side, which nevertheless appears quite sharp. As can be seen from fig. 4.12b, the weak shock on the pressure side is almost isentropic (for this shock the upstream Mach number $Ma_1 = 1.1$, and the relative entropy jump is $\approx 4 \cdot 10^{-4}$).

In tables 4.3 and 4.4 we report the values of the lift and drag coefficients, with the corresponding work units, on the three grids obtained with the second and fifth order scheme. The lift and drag coefficients are defined as

$$C_l = \frac{\mathbf{F} \cdot \mathbf{k}_\perp}{\frac{1}{2} \rho_\infty U_\infty^2 A_{ref}}, \quad (4.13)$$

$$C_d = \frac{\mathbf{F} \cdot \mathbf{k}_\parallel}{\frac{1}{2} \rho_\infty U_\infty^2 A_{ref}}, \quad (4.14)$$

respectively, where $\mathbf{F}$ is the force acting on the airfoil (see eq. (3.105)), $\mathbf{k}_\perp$ is the unit vector perpendicular to the free-stream, $\mathbf{k}_\parallel$ is the unit vector parallel to the free-stream, and A_{ref} is a reference area (for a two-dimensional problem, A_{ref} is equal to the chord length c). In tables 4.3 and 4.4, the result of Richardson extrapolation [95, 12] is also shown when possible (convergence has to be monotonic). Furthermore, grid converged values of the aerodynamic coefficients obtained by other authors are reported in table 4.5, as reference. Due to the scatter of the reference results and due to the oscillating convergence of the lift coefficient in our results, the accuracy is difficult to measure. For both the second and fifth order scheme, Richardson extrapolation of the drag coefficient gives lower values than those obtained by the other authors. Without a precise measure of the accuracy, it is not possible to compare in a rigorous manner the efficiency of the two schemes.

However, we note the large differences in terms of computational time between the second and fifth order method. For instance, on the fine grid, the latter is almost four times more expensive than the former. At the same time, the second order scheme, on the fine grid, gives a value of C_d closer to the extrapolated ones than the fifth order scheme. Additionally, Richardson extrapolation for C_d shows 2nd order convergence for the 5th order scheme, and 3rd order convergence for the 2nd order scheme. Therefore, these considerations seem to indicate that, for this test case, the high-order scheme does not deliver a better efficiency than the second order scheme. Even though reasonable results have been obtained, the numerical simulation of non-smooth flows, like the flow with a shock, with the SBP-SAT schemes described in this work is still a challenge.

Table 4.3: 2nd order results for NACA0012, $Ma_\infty = 0.8$, $AoA = 1.25^\circ$, with shock capturing.

C_l	C_d	h	work units
3.49137e-01	2.27823e-02	7.31175e-03	11.6
3.52547e-01	2.26357e-02	3.66931e-03	24.2
3.51919e-01	2.26175e-02	1.83803e-03	448.2
-	2.26149e-02	o.o(Richardson)	-

Table 4.4: 5th order results for NACA0012, $Ma_\infty = 0.8$, $AoA = 1.25^\circ$, with shock capturing.

C_l	C_d	h	work units
3.48051e-01	2.28588e-02	7.31175e-03	31.8
3.51929e-01	2.26727e-02	3.66931e-03	212.3
3.51717e-01	2.26320e-02	1.83803e-03	1742.3
-	2.26206e-02	o.o(Richardson)	-

Table 4.5: Grid-converged values for NACA0012, $Ma_\infty = 0.8$, $AoA = 1.25^\circ$, obtained by different authors [1].

group	C_l	C_d
RWTH Aachen	3.52914e-01	2.27464e-02
Univ. of Michigan	3.52022e-01	2.26260e-02
MIT	3.51694e-01	2.26275e-02

4.4 ANALYTICAL 3D SLENDER BODY

The geometry for this case is described analytically in [1] (see test case C2.3). It concerns a slender body based on a 10% thick airfoil with boundaries constructed

by a surface of revolution. Figure 4.13 shows the medium computational grid, which is composed out of 6 blocks with a total of approximately $8 \cdot 10^5$ grid points. It is obtained from the fine grid, which contains about 6.2 million grid points, by applying a regular coarsening once. The surface mesh (figure 4.13a) was generated with Pointwise [52] while for the volume mesh (figure 4.13b) we employed an in-house hyperbolic grid generator inspired by the work of Chan and Steger [15]. Note that fig. 4.13c shows half of the model to highlight the mesh resolution close to the wall. On the fine grid, the minimum height of the cells at the wall is $2 \cdot 10^{-5}c$. Note also that the trailing edge, originally sharp, has been rounded with a radius of $10^{-4}c$, where c is the chord length, to facilitate the mesh generation, see fig. 4.13d.

For the workshop, three flow conditions have been proposed, namely

1. Subsonic inviscid flow with $Ma_\infty = 0.5$ and angle of attack 1° .
2. Subsonic laminar flow with $Ma_\infty = 0.5$ and angle of attack 1° . Reynolds number based on chord length is 5000.
3. Subsonic turbulent flow with $Ma_\infty = 0.5$ and angle of attack 5° . Reynolds number is $10 \cdot 10^6$.

Here we only present the results for the case of laminar flow. We seek a steady solution starting from a uniform free-stream flow and using the (implicit) Newton-Krylov iterative method. The quantities of interest are the lift and drag coefficient. Free-stream preserving metrics, see section 2.2.3, are employed and the default values of the artificial dissipation coefficients are used. The convergence history for the case of laminar flow on the finest grid with the 5th order scheme is shown in figure 4.14. Note the grid sequencing and the quadratic convergence of Newton's method. The CPU time for this simulation was 6.5 hours on 8 cores (of an Intel Xeon X5650 processor, 2.57 GHz), which corresponds to $2.32 \cdot 10^4$ Tau work units. Furthermore, 250 Gb of memory were needed for the implicit algorithm. Figure 4.15 shows the contours of the surface pressure coefficient C_p and the skin-friction streamlines for the fine grid for the 5th order scheme. The flow is smooth and the boundary layer is attached. The computed lift and drag coefficients for the three different grids are presented in table 4.6; the last line shows the values obtained Richardson extrapolation. The solution obtained with the fifth order scheme on the medium grid is more accurate than the second order solution on the fine grid. As the fifth order scheme on the medium grid requires $2.42 \cdot 10^3$ work units and the second order scheme on the fine grid needs $1.10 \cdot 10^4$ work units, the gain in efficiency achieved by the high order scheme is substantial.

Furthermore, the Richardson extrapolation shows 7th order convergence for C_d and C_l for the 5th order scheme. A possible explanation of this super convergence can be found in the dual (also called adjoint) consistency theory formulated by Hicken and Zingg [46, 47]. A discretization is dual consistent if it leads to a discrete dual problem that is a consistent discretization of the continuous dual PDE. In [46, 47] it is shown that a dual-consistent discretization of the *Euler* equations using an SBP-SAT scheme can yield super convergent estimates of discrete functionals, such as the lift and drag force. In particular, if the dual consistent SBP-SAT discretization

Figure 4.13: Medium grid for the slender body. It is obtained from the fine grid by applying a regular coarsening once.

Table 4.6: Laminar flow over the analytical slender body, $Ma_\infty = 0.5$, $Re_c = 5000$, $AoA = 1^\circ$: lift and drag coefficient obtained on the coarse, medium and fine grid with the 2nd and 5th order scheme. The values obtained with the Richardson extrapolation are also shown. “No solution” means that it was not possible to obtain a solution using the same artificial dissipation coefficient used on the finer grids. In this case the extrapolated value is obtained assuming the design order of accuracy of the scheme.

h	2 nd order		5 th order	
	C_l	C_d	C_l	C_d
2.11080e-2	No solution	No solution	2.61558e-3	6.45973e-2
1.07660e-2	2.56728e-3	6.35770e-2	2.58406e-3	6.40381e-2
5.43830e-3	2.57927e-3	6.38830e-2	2.58382e-3	6.40336e-2
0.0 (Richardson)	2.58327e-3	6.39850e-2	2.58382e-3	6.40336e-2

Figure 4.14: Convergence history (L^2 -norm of density residual) for laminar flow over the analytical slender body, $Ma_\infty = 0.5$, $Re_c = 5000$, $AoA = 1^\circ$, 5th order scheme, on the finest grid.

Figure 4.15: Laminar flow over the analytical slender body, $Ma_\infty = 0.5$, $Re_c = 5000$, $AoA = 1^\circ$: contours of the surface pressure coefficient C_p and skin-friction streamlines for the fine grid, 5th order scheme.

is 2p-order accurate in the interior (and p-order accurate at the boundary), and if the primal and dual solutions are sufficiently smooth, then the discrete functional will also be 2p-order accurate.

4.5 DNS OF TAYLOR-GREEN VORTEX

This test case is intended to test the capability of the method to capture turbulence accurately. The initial data is smooth and 3-dimensional and the flow will transition to turbulence. The initial state is given by

$$\begin{aligned}
 u(x, y, z, 0) &= U_0 \sin\left(\frac{x}{L}\right) \cos\left(\frac{y}{L}\right) \sin\left(\frac{z}{L}\right) \\
 v(x, y, z, 0) &= -U_0 \cos\left(\frac{x}{L}\right) \sin\left(\frac{y}{L}\right) \sin\left(\frac{z}{L}\right) \\
 w(x, y, z, 0) &= 0 \\
 p(x, y, z, 0) &= p_0 + \frac{\rho_0 U_0^2}{16} \left(\cos\left(\frac{2x}{L}\right) + \cos\left(\frac{2y}{L}\right) \right) \left(\cos\left(\frac{2z}{L}\right) + 2 \right) \\
 \rho(x, y, z, 0) &= \frac{p(x, y, z, 0)}{RT_0}
 \end{aligned}$$

The flow is governed by the Navier-Stokes equations with a Prandtl number of $Pr = 0.71$, specific heat ratio $\gamma = 1.4$ and the bulk viscosity is assumed to be zero. Furthermore, the Mach number $U_0/c_0 = 0.1$ and the Reynolds number $Re = \frac{\rho_0 U_0 L}{\mu} = 1600$. The initial temperature is uniform, $T_0 = \frac{p_0}{\rho_0 R}$, and the dynamic viscosity μ is as

Table 4.7: Number of time steps, corresponding $\Delta t/t_c$ and computational cost, for Taylor-Green vortex.

grid	scheme	# time steps	$\Delta t/t_c$	work units
65^3	2 nd order	2348	$8.5 \cdot 10^{-3}$	$3.96 \cdot 10^2$
	7 th order	5870	$3.4 \cdot 10^{-3}$	$1.38 \cdot 10^3$
129^3	2 nd order	4696	$4.3 \cdot 10^{-3}$	$6.13 \cdot 10^3$
	7 th order	11739	$1.7 \cdot 10^{-3}$	$1.95 \cdot 10^4$
257^3	2 nd order	9391	$2.1 \cdot 10^{-3}$	$1.11 \cdot 10^5$
	7 th order	23476	$8.5 \cdot 10^{-4}$	$3.59 \cdot 10^5$
513^3	7 th order	50000	$4.0 \cdot 10^{-4}$	$9.55 \cdot 10^6$

sumed to be constant, $\mu(x, y, z, t) = \mu(T_0)$. The solution is computed on the periodic domain $\Omega = \{-\pi L \leq x, y, z \leq \pi L\}$ which is discretized using four uniform structured grids containing 65^3 , 129^3 , 257^3 and 513^3 vertices, respectively. For the 65^3 , 129^3 and 257^3 grids it was possible to use an in-house local Linux cluster, while the 513^3 grid was run on up to 512 processors of the LISA machine of SurfSARA, the Netherlands Supercomputer Center.

With a convective time scale $t_c = L/U_0$, the final time in the simulation is $t_{\text{final}} = 20t_c$. The classical 4th order Runge-Kutta scheme is used for the time-derivative of the governing equations. The spatial part is computed with the 2nd and 5th order scheme on all grids but the finest, for which only the 5th order solution has been generated. Since the domain is periodic in all three directions, the internal discretization is used for the entire domain. Hence, the 5th order scheme, which features an 7th order discretization in the interior (without artificial dissipation it would be 8th order accurate), should provide a 7th order solution and it will be referred to as such in the remainder of this section. The computational cost, the number of time steps of the numerical simulation and the corresponding Δt , are presented in table 4.7 for each of the cases. Again, the computational cost is expressed in terms of work units, that is the CPU time scaled to one processor and divided by the cost of TauBench. The number of time steps is chosen based on stability considerations. Reducing the time step by a factor of four did not change the results significantly. Considering table 4.7, note that, for a given grid, the high-order scheme requires a time step approximately 2.5 times smaller than the time step needed for stability by the second-order scheme. Furthermore, the computational cost of the evaluation of one time step using the high-order scheme is 30-40% larger than the computational cost of one time step using the second-order scheme.

The temporal evolution of the flow field is shown in figure 4.16, which presents the iso-surfaces of the Q-criterion $Q = 0.1(U_0/L)^2$, colored with the velocity magnitude U/U_0 (see [77] for comparison). The Q-criterion is defined as ([51]):

$$Q = \frac{1}{2} (\Omega_{ij}\Omega_{ij} - S_{ij}S_{ij}) \quad (4.15)$$

where

$$S_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \quad \Omega_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i} \right). \quad (4.16)$$

The Q-criterion defines a vortex in the regions of the domain where $Q > 0$.

The initial large eddies (fig. 4.16a) gradually evolve towards smaller turbulent structures without reaching an isotropic state, symmetries are still present in the last solution (fig. 4.16f). The peak of the dissipation of kinetic energy occurs at $t/t_c \approx 8$, when the maximum of the velocity magnitude is observed. Then, a decay phase starts during which dissipation takes place at increasingly lower rates.

Our results are compared to an available reference flow solution (see test case C3.5 in [1]). This reference solution was obtained with a dealiased pseudo-spectral method run on a 513^3 grid; time integration was performed with a low-storage 3-steps Runge-Kutta scheme and a non-dimensional time step of $1.0 \cdot 10^{-3}$. The solution consists of the temporal evolution of the following non-dimensional mean quantities:

- the kinetic energy $E_k = \frac{1}{\rho_0 \Omega u_0^2} \int_{\Omega} \rho \frac{\mathbf{u} \cdot \mathbf{u}}{2} d\Omega$;
- the dissipation of kinetic energy $\epsilon = -t_c \frac{dE_k}{dt}$;
- the enstrophy $\mathcal{E} = \frac{t_c^2}{\rho_0 \Omega} \int_{\Omega} \rho \frac{\boldsymbol{\omega} \cdot \boldsymbol{\omega}}{2} d\Omega$.

Furthermore, the iso-contours of the magnitude of the vorticity on the periodic face $x/L = -\pi$ at $t/t_c = 8$ were provided as well.

A study of the influence of the artificial dissipation, see section 3.2.4, has been performed on the 257^3 grid using the 7th order scheme. Figure 4.17 shows the evolution of the enstrophy obtained for different artificial dissipation coefficients down to the stability limit. It is clear that reducing the amount of artificial dissipation leads to a better solution. The evolution of dissipation of kinetic energy and enstrophy are presented in figure 4.18 for the different schemes and grids used. The dissipation of kinetic energy at time step i , ϵ_i , has been computed with a simple centered second order finite difference scheme, i. e.

$$\epsilon_i = -(E_{k,i+1} - E_{k,i-1})t_c / (2 \Delta t).$$

The iso-contours of the magnitude of the vorticity are shown in fig. 4.19. Our solution for the fine grid is extremely close to the reference solution.

From fig. 4.18b and from table 4.7 it is clear that the 7th order scheme outperforms the 2nd order scheme. For instance, the 7th order solution for the 129^3 grid is more accurate (closer to the reference) than the 2nd order solution for the 257^3 grid; at the same time, the computational cost of the former is almost six times smaller than that of the latter ($1.95 \cdot 10^4$ and $1.11 \cdot 10^5$ work units, respectively). The same applies to the other grids. Therefore the 7th order scheme gives a more accurate solution at a lower computational cost than the 2nd order scheme, although the difference

Figure 4.16: Taylor-Green vortex: evolution of iso-surfaces of the Q -criterion ($Q = 0.1(U_0/L)^2$) colored with the non-dimensional velocity magnitude $|\mathbf{u}|/U_0$. The solution has been computed with the 7th order scheme on the 257^3 grid.

Figure 4.17: Taylor-Green vortex: evolution of the enstrophy for different artificial dissipation coefficients α (eq. (3.43)), for 257^3 grid, 7th order scheme.

between the two is smaller than expected from a truncation error analysis (not reported here). However, design order refers to the asymptotic slope of the error of a smooth solution that is well resolved. With the exception of the 7th order solution for the finest grid, the present DNS solutions are far from being resolved. For the present resolution the solution can be viewed as non-smooth, which explains the sub-optimal convergence rates. Nevertheless, the error is still smaller for the high-order scheme due to its superior ability to resolve sharp gradients.

4.6 LES: TRANSITIONAL FLOW OVER THE SD7003 WING

This test case is aimed at assessing the numerical method's capability of capturing mixed laminar, transitional and turbulent flows encountered in low-Reynolds problems (test case C3.3 in [1]). At an angle of attack of 8 degrees, free-stream Mach number of 0.1 and Reynolds number equal to $60 \cdot 10^3$, the unsteady flow over an SD7003 wing section is characterized by laminar flow separation and the formation of a transitional zone, followed by the reattachment of the turbulent boundary layer. When considering a time-averaged solution, the transitional zone appears as a laminar separation bubble on the airfoil surface.

For this flow, the full Navier-Stokes equations for 3D unsteady, compressible and viscous flow are solved with the ILES approach (i.e. no explicit modeling of the subgrid-scale term), and with the WALE model (see section 2.3.2). The ratio of specific heats and the Prandtl number are constant and equal to 1.4 and 0.72, respectively. The far-field is located approximately 95 times the chord length c away from the airfoil, where no synthetic turbulence is injected. The flow is considered homogeneous in the spanwise direction (z) and periodic boundary conditions are imposed at $z/c = 0$ and $z/c = 0.2$. The surface of the airfoil is treated as an adiabatic no-slip wall. The computational domain is discretized with an O-grid topology, see

(a) Dissipation of kinetic energy

(b) Enstrophy

Figure 4.18: Evolution of the dissipation of kinetic energy and evolution of enstrophy for different schemes and grids used for the Taylor-Green vortex.

Figure 4.19: Iso-contours (levels: [1,5,10,20,30]) of the dimensionless magnitude of vorticity on the periodic face $x/L = -\pi$ at $t/t_c = 8$. Comparison with the Reference data for the 513³ and 257³ grids.

figure 4.20, which is generated with Pointwise [52] and contains approximately 4 million grid points ($385 \times 129 \times 81$).

Because of the large computational requirements, only the 5th order scheme has been employed for this test case. A coarser mesh has been used only to obtain the initial solution. Due to the choice of an explicit time integration scheme, the classical RK4, the time step is limited by stability considerations and not by accuracy. For the given conditions a non dimensional time step of $\Delta t/t_c = \Delta t \cdot U_\infty/c = 4.3 \cdot 10^{-6}$ has been used. As this time step is mainly restricted by either the acoustic or viscous contributions, which are (almost) constant, this time step is kept constant throughout the simulation. It was requested to compute the time-averaged solution over an interval of 10 convective times ($10t_c = 10c/U_\infty$), upon the decay of the initial transients. The simulation was first run with the ILES approach. The stationary state (in a time-averaged sense) was reached after 7.2 convective times, therefore the simulation was performed until $17.2t_c$, for a total of approximately 4 million time steps, see fig. 4.21. Then, another 10 convective times were computed using the WALE model. The ILES simulation was run for almost a month on 128 processors of the LISA machine of SARA, the Netherlands Supercomputer Center, and the Hexagon machine of the University of Bergen, Norway, for a total of approximately $28 \cdot 10^6$ work units. The simulation with the WALE model needed 40% less time since the initial transients were already eliminated during the ILES part of the numerical simulation.

Once again, our solution is compared to a reference solution. The reference (grid-converged) solution was obtained on a fine grid of approximately $54 \cdot 10^6$ vertices ($971 \times 448 \times 125$), with a 6th order compact finite-difference discretization and with the ILES approach, see [32].

Figure 4.22 shows the iso-surfaces of the instantaneous Q-criterion [51] ($Q = 500 \text{ s}^{-2}$) at time $t/t_c = 16$ during the (time averaged) stationary state. The transi-

Figure 4.20: O-Grid for the SD7003 wing ($385 \times 129 \times 81$).

Figure 4.21: History of lift and drag coefficient for the ILES of the SD7003 wing; $Ma_\infty = 0.1$, $Re_c = 6 \cdot 10^4$, $AoA = 8^\circ$.

tion to turbulence is apparent. The final averaged solution is obtained by averaging in the spanwise direction, which gives a 2D (time and space) averaged solution. Based on the averaged solution, the distribution of the surface pressure and skin friction coefficients (C_p , C_f) are computed. Both coefficients are plotted in figure 4.23a and 4.23b, respectively. The skin friction coefficient is defined as

$$C_f = \frac{\tau_w}{\frac{1}{2} \rho_\infty U_\infty^2}, \quad (4.17)$$

where τ_w is the wall shear stress. The two approaches lead to very similar results. As expected, the WALE model is slightly more diffusive. In the region $x/c \in [0.25, 0.3]$ a sharp adverse pressure gradient is observed. The separation occurs at approximately $0.03 x/c$ on the suction side, where the skin friction coefficient becomes negative. The separation bubble ends at approximately $0.3 x/c$, where the skin friction coefficient becomes positive again. Compared to the reference solution, on the suction side the magnitude of the skin-friction coefficient is slightly over-predicted in the region between the reattachment zone and the trailing edge.

The laminar separation bubble is shown in detail in figure 4.24, where the contours of the average x -component of the velocity ($\bar{u}/U_\infty$) are plotted. The resulting aerodynamic coefficients, together with the details of the bubble, are given in table 4.8.

Figure 4.25 shows the y^+ distribution for the vertices on the airfoil. The quantity y^+ is the non-dimensional first grid spacing normal to the wall defined here as

$$y^+ = \frac{\rho u_\tau \Delta y}{\mu}, \quad (4.18)$$

Figure 4.22: Animation of the iso-surfaces of the Q-criterion ($Q = 500 \text{ s}^{-2}$) colored with pressure, $8 < t/t_c < 16$; SD7003 airfoil, $Ma_\infty = 0.1$, $Re_c = 6 \cdot 10^4$, $AoA = 8^\circ$.

Figure 4.23: Pressure and skin-friction coefficient on the SD7003 airfoil, $Ma_\infty = 0.1$, $Re_c = 6 \cdot 10^4$, $AoA = 8^\circ$.

Figure 4.24: SD7003 airfoil, $Ma_\infty = 0.1$, $Re_c = 6 \cdot 10^4$, $AoA = 8^\circ$: contours of the non-dimensional average x-component of velocity $\bar{u}/U_\infty$ in the region of the separation bubble.

Table 4.8: SD7003 airfoil, $Ma_\infty = 0.1$, $Re_c = 6 \cdot 10^4$, $AoA = 8^\circ$: comparison between results from ILES, WALE and Reference ([32]).

	C_l	C_d	Separation point (x/c)	Reattachment point (x/c)
ILES	0.9182	0.0447	0.034	0.308
WALE	0.9234	0.0453	0.032	0.308
Reference	0.917	0.0447	0.031	0.303

where Δy is the distance between the vertex at the wall and its closest neighbor in the direction normal to the wall, and $u_\tau = \sqrt{|\tau_w|/\rho}$ is the friction velocity; ρ and μ are the density and the dynamic viscosity at the wall, respectively. For most of the vertices, y^+ is between 0.1 and 0.5. In particular, we mention that at $x/c = 0.8$ on the suction side the value of y^+ is 0.3.

Profiles of the non-dimensional average x-component of velocity, $\bar{u}/U_\infty$, and profiles of the average fluctuations of the x-component of velocity, $(\overline{u'u'})^{1/2}/U_\infty$, are shown in fig. 4.26. They have been determined at vertical lines every tenth of the chord on the suction side of the airfoil. The average velocity fluctuations $\overline{u'u'}$ are defined as

$$\overline{u'u'} = \frac{1}{T} \int_0^T u'(t)u'(t) dt \approx \frac{1}{n_T \Delta t} \sum_{i=1}^{n_T} u'_i u'_i \Delta t \quad (4.19)$$

Figure 4.25: Distribution of y^+ for the vertices on the SD7003 airfoil, $Ma_\infty = 0.1$, $Re_c = 6 \cdot 10^4$, $AoA = 8^\circ$.

where we have used the classical Reynolds decomposition

$$u(t) = \bar{u} + u'(t) \quad (4.20)$$

for which

$$\int_0^T u(t) dt = T\bar{u}, \quad \int_0^T u'(t) dt = 0. \quad (4.21)$$

Here $u(t)$ is the instantaneous solution while $\bar{u}$ and $u'(t)$ are the average and fluctuating part of the solution, respectively. T represents the final integration time and n_T is the total number of time steps used to discretize the interval $[0, T]$ (eq. (4.19)). Dropping from now on the dependence on time for readability reasons, the average of the squared velocity can be computed as

$$\begin{aligned} \overline{u^2} &= \frac{1}{T} \int_0^T u^2 dt = \frac{1}{T} \int_0^T (\bar{u} + u')^2 dt \\ &= \frac{1}{T} \int_0^T (\bar{u}^2 + 2\bar{u}u' + u'u') dt \\ &= \bar{u}^2 + \frac{2\bar{u}}{T} \int_0^T u' dt + \overline{u'u'} \\ &= \bar{u}^2 + \overline{u'u'}. \end{aligned} \quad (4.22)$$

Thus, the average fluctuations can be derived from the average of the velocity and the average of its square as

$$\overline{u'u'} = \overline{u^2} - \bar{u}^2. \quad (4.23)$$

(a) x-component velocity $\bar{u}/U_\infty$

(b) fluctuations x-component velocity $((u'u')^{1/2}/U_\infty)$

Figure 4.26: SD7003 airfoil, $Ma_\infty = 0.1$, $Re_c = 6 \cdot 10^4$, $AoA = 8^\circ$: average x-component of velocity (fluctuations), plotted at every $0.1x/c$ along the suction side of the airfoil. Note that in fig. (b) the scale is exaggerated in order to highlight the small differences between the results.

Therefore, in order to compute $\overline{u'u'}$ two running time-averages of the 3D solution have been stored, one for u and one for u^2 . The running average of u at the n -th time step, $\bar{u}_n$, is determined as

$$\bar{u}_n = \bar{u}_{n-1} + (u_n - \bar{u}_{n-1})/n, \quad n > 0, \bar{u}_0 = 0, \quad (4.24)$$

where u_n is the instantaneous value of u at the n -th time step. Similarly $\bar{u^2}$ is computed. Note that eq. (4.24) assumes that Δt is kept constant⁴. Note also that the actual values $\overline{u'u'}$ as plotted in fig. 4.26b, have been obtained by using eq. (4.23) *after* the spanwise average was performed. In other words, as suggested at the workshop [1], the terms appearing in the right hand side of eq. (4.23) are the spanwise average of the time average values of u^2 and u .

As can be seen in fig. 4.26, ILES and WALE results are very close to each other. Consistently with the skin-friction distribution, for $x/c > 0.4$ the derivative of the velocity at the wall $\partial \bar{u}/\partial y$ is slightly larger for ILES and WALE than for the reference solution, see fig. 4.26a. Small discrepancies can also be observed in the velocity fluctuations, fig. 4.26b. Nevertheless, considering also the very large differences in grid resolution between our results and the reference results, a very good agreement is observed.

4.7 THREE-ELEMENT AIRFOIL

This test case consists of the turbulent flow over a three-element airfoil in high-lift configuration (see test case C3.1 in [2]). It was chosen to test the high-order schemes for a two-dimensional turbulent flow, governed by the RANS equations, for a complex configuration. The target quantities of interest are the lift and drag coefficients at one free-stream condition: free stream Mach number $Ma_\infty = 0.2$, angle of attack $\alpha = 16^\circ$, and Reynolds number $Re = 9 \times 10^6$ based on the reference chord of $c_{ref} = 0.5588m$.

This test case has been run on three different grids, see fig. 4.27. The coarser grids are obtained by removing every other point from the finer grid (regular coarsening). The far field is located at 5000 chords away from the airfoil and the 2D fine grid contains 1×10^6 grid points distributed over 47 computational blocks. However, the numerical method is 3D and therefore the actual grid size is 2×10^6 with 2 vertices in the third dimension. As can be seen in fig. 4.27 where each computational block is drawn with a different color, the grid topology that had to be used for this test case is quite complex. Since we are using a structured multi-block method, such complexity is nevertheless unavoidable for this non-trivial geometry, unless overset grids are used. The main drawback of the present mesh is the high number of degrees of freedom in regions below the second and third element; there the

⁴ For a non-constant time step, the running average of u at time $t = \sum_{i=1}^n (\Delta t)_i$, $\bar{u}_t$, can be determined as

$$\bar{u}_t = \bar{u}_{(t-(\Delta t)_n)} + \frac{(\Delta t)_n}{t} (u_n - \bar{u}_{(t-(\Delta t)_n)}), \quad t > 0, \bar{u}_{t=0} = 0.$$

Figure 4.27: Coarse 2D grid (62.5×10^3 vertices) for the three-element airfoil. 47 blocks in total.

flow will not vary very much and such resolution is unnecessary. Furthermore, it turns out that on the coarse grid there are vertices for which the Jacobian of the transformation (see eq. (2.32)), computed with the 5th order metrics, is negative. This is a clear indication that the smoothness of the mesh needs to be improved.

The solution has been sought utilizing three different schemes, namely second, third and fourth order. In order to establish a reference with the current state-of-the-art finite volume methods, the same test case was run with the code `SUmb` [124]. `SUmb` uses a multi-block structured finite-volume method. For this simulation the discretization of the inviscid fluxes was carried out with Roe’s upwind scheme [96], linear reconstruction and no limiter. The viscous fluxes were computed using a central discretization and the classical SA model was used. Flow and turbulence equations were solved in a segregated manner: a RK smoother is used for the flow equations (see section 3.3.2), whereas the SA turbulence equation is solved with Alternating Direction Implicit (ADI) splitting technique [130]. `SUmb` provides a second order cell-centered solution.

Figure 4.28 shows the contours of the Mach number resulting from the 4th order solution on the fine grid. The flow is subsonic and there is no separation on the suction side of either element. The wake behind the trailing edge of the first element is clearly visible.

In fig. 4.29 we plot the lift and drag coefficients against the grid size and the non-dimensional CPU time. The lift coefficient tends to increase together with the order of accuracy and with the number of degrees of freedom. The drag coefficient has the opposite tendency. We assume that the most accurate results are those of the 4th order scheme on the fine grid. In fact, these values of lift and drag are

Figure 4.28: Contours of the Mach number of the three-element airfoil, $Ma_\infty = 0.2$, $Re_c = 9 \cdot 10^6$, $AoA = 16^\circ$; fine grid, 4th order solution.

very close to the findings of other researchers as can be seen in [2]⁵. The CPU time shown is the one necessary for a relative reduction of the residuals of four orders of magnitude. Tests on the coarse and medium grids showed that the difference between converging 4 and 7 orders of magnitude is ± 0.001 for the lift coefficient and $\pm 10^{-4}$ for C_d . In our experience, the most reliable solution strategy is to perform a grid and order sequencing. This means that on a given grid, the second order solution is used as initial guess for the third order solution, which, in turn, is used as initial guess for the fourth order scheme. Subsequently, the fourth order solution is interpolated to the finer grid and used as initial guess for the second order solution on so on. While this may not be the fastest way to obtain a certain solution, it is always a safe approach for non-trivial problems like this test case, thus it is recommended. However, in order to make a fair comparison between low and high order schemes and trying to minimize the computational time, for the second order scheme only grid sequencing was used.

Considering figs. 4.29a and 4.29c, it is clear that the high-order schemes give more accurate results on all grids. In particular, the fourth-order scheme consistently gives more accurate results, whereas the difference between the accuracy of the third and second order scheme is not that pronounced. These two schemes give very similar values of the drag coefficient on the finer grids, yet the third order scheme predicts a higher lift coefficient. The efficiency in terms of computational time can be evaluated analyzing figs. 4.29b and 4.29d. Regarding the results of the SBP-SAT schemes, the fourth order scheme is again slightly more efficient. For the drag coefficient, however, this is true only by a small margin. Furthermore, the

⁵ For instance, the DLR group reports the following values of lift and drag coefficients: $C_l = 4.16634$; $C_d = 0.0472271$.

(a) lift coefficient vs grid scale h

(b) lift coefficient vs work units

(c) drag coefficient vs grid scale h

(d) drag coefficient vs work units

Figure 4.29: Three-element airfoil, $Ma_\infty = 0.2$, $Re_c = 9 \cdot 10^6$, $AoA = 16^\circ$: lift and drag coefficient versus the grid scale h and the normalized computational time.

Figure 4.30: Three-element airfoil, $Ma_\infty = 0.2$, $Re_c = 9 \cdot 10^6$, $AoA = 16^\circ$: convergence history for the three grids, 4th order scheme.

second order scheme is slightly more efficient than the third order scheme. Nevertheless, the finite-volume method Sumb clearly achieves the best efficiency.

Considering once more figs. 4.29a and 4.29c, we note that the 2nd order finite-volume code Sumb provided results somewhat in between the results of the 3rd and 4th order SBP-SAT schemes. Thus, the actual accuracy (at least in terms of integrated quantities as the lift and drag coefficient) of the SBP-SAT schemes is not as high as expected. A possible explanation is the very high number of blocks of the grid (47) and, associated with that, the high number of internal boundaries. In fact, at these boundaries, where a halo treatment is not used, the order of accuracy of the schemes reduces to half of the order of accuracy in the interior and the SAT terms add some dissipation. The high number of these interfaces, combined with the non-optimal smoothness of the mesh, might have caused the reduction in accuracy.

Regarding efficiency, while figs. 4.29b and 4.29d indicate that the fourth order scheme is overall more efficient than the second order (finite-difference) scheme, it must be noted that the comparison has been carried out for runs converged 4 orders of magnitude⁶. That was necessary because a very slow convergence was observed for the fine grid, especially for the third and fourth order schemes. As an example, in fig. 4.30 we show the convergence history of the 4th order scheme for the three grids. The number of non-linear iterations required for the residual to reduce 4 orders of magnitude, increases very rapidly and so does the CPU time. Since always a second order preconditioner is used, it is expected that the solution

⁶ Nevertheless, the same comparison for the coarse and medium grids, with residual reduced by 7 orders of magnitude (not shown here), leads to the same conclusion.

of the linear systems with a higher order Jacobian requires more iterations. This clearly results in more CPU time. In order to limit this, one may increase the levels of fill-in of the preconditioner, or decrease/fix manually the CFL number used for damping. The first strategy will increase the time needed for the factorization as well as the memory requirements, but will also yield a better preconditioner thus helping the convergence of the linear systems. The second strategy will not increase the memory requirements, it will reduce the time required for the solution of the linear systems (the preconditioner becomes more diagonally dominant and easier to invert), but will negatively affect the convergence of the non-linear system. On the fine grid, however, the amount of fill-in was limited by the hardware (amount of memory), thus the only option was to limit/reduce manually the CFL. This was detrimental for the speed of convergence for the fine grid, especially for the higher-order schemes. Furthermore, we recall that all runs have been performed using the present method for 3D flows, and we do not take advantage of the particular structure of the Jacobian/preconditioner that results from having only 2 vertices in the third dimension.

Finally, we note that, contrary to all other RANS test cases, the problem of the three-element airfoil required the intervention of the user to change manually the CFL used for the damping. In particular, the CFL number was divided by 10 every time the solution of the linear system required more than 5000 iterations, or it was set to a constant value. Furthermore, initial CFL numbers of 10 to 100 were used. That manual intervention was required is an indication that the simple SER strategy (see section 3.4.2) is not optimal for this test case and other options should be investigated. In addition, the comparison of the CPU time with that required by SUMB suggests that our iterative technique (implicit, fully coupled, see section 3.4.1) may not be the most efficient choice for RANS problems.

4.8 ROTOR37

This rotor has originally been designed by Reid and Moore [94] as the rotating part of Stage37, one of four inlet stages for an eight-stage core compressor with 20 : 1 pressure ratio. Experimental results for the isolated rotor were performed later with both aerodynamic probes and laser anemometry [107, 108]. Radial distributions of static and total pressure, total temperature, and flow angle were measured at the two axial stations labeled stations 1 ($z = -4.19\text{cm}$) and 4 ($z = 10.67\text{cm}$) in fig. 4.31. The relative Mach number was measured at stations 2 ($20\%x/c$), 3 and 4 and in the five spanwise planes also shown in fig. 4.31. The main design parameters are reported in table 4.9.

The experimental data were used for a CFD blind test conducted by the International Gas Turbine Institute (IGTI) of the American Society of Mechanical Engineers (ASME) in 1994 (unpublished), and for a more detailed but non-blind test conducted later in 1998. The main results of the latter test are available in [26]. Since then, Rotor37 has been a popular test case for CFD validation and, as already

Figure 4.31: Cross section of the NASA Rotor37 [107].

Table 4.9: Rotor37 design parameters [107, 26].

Number of rotor blades	36
Tip radius at leading edge	252 mm
Inlet Hub-Tip radius ratio	0.7
Blade aspect ratio	1.19
Tip clearance height	0.0356 cm
Tip solidity	1.29
Rotational speed	17188.7 rpm
Tip speed	454 m/s
Total pressure ratio	2.106
Mass flow	20.19 kg/s
Choked mass flow	20.93 kg/s
Adiabatic efficiency	0.877

reported in [23], it has been proven quite challenging for RANS and, more recently, for LES methods as well. In fact, discrepancies between the experimental and numerical data from both RANS and LES simulations can still be found in recent papers (see [16, 37, 42]). The flow in Rotor37 is quite complex: it is transonic and shock interaction with the tip leakage vortex and shock-boundary layer interaction have been observed.

4.8.1 Operating and boundary conditions

Most of the available data focus on one operating point, which is 98% of the maximum mass flow, at nominal rotation speed. We focus on this operating point as well and we solve the RANS equations. The available boundary conditions for the SBP-SAT discretization technique are far-field, slip/no-slip wall and internal interface, see section 3.2.3. These boundary conditions are appropriate for external flows. However, typically a turbomachinery problem (like Rotor37) is an internal flow problem, for which the velocity at the inlet and the velocity at the outlet

is not known a priori. The common approach in these cases is to prescribe the total conditions (total pressure and total enthalpy/temperature) and the velocity direction at the inlet, combined with prescribed static pressure at the outlet. Unfortunately, such kind of boundary conditions are not yet available in the SBP-SAT framework. Within this framework, the most appropriate boundary condition for the inlet and outlet surfaces is the far-field condition which, however, requires the entire state vector in each vertex of these surfaces. To use this boundary condition, the following approach is taken. The simulation is first carried out with the already mentioned finite-volume code `SUmb` [124], which allows to prescribe the total conditions at the inlet combined with prescribing the static pressure at the outlet. Then, upon convergence, the solution is extracted at the inlet and outlet planes, and imposed as far-field state in the SBP-SAT method described in the present thesis. The simulation with `SUmb` has been performed with the JST scheme ([55]) and with the original version of the SA turbulence model ([4]). The total inlet conditions are given in [26] and the outlet pressure has been varied in order to obtain the target mass flow rate. The results shown here have been obtained with an outlet static pressure of 108500Pa.

4.8.2 Computational domain

The computational domain and the corresponding surface mesh are shown in fig. 4.32 and fig. 4.33, respectively. The axial direction of the flow is the in z -axis and the blade rotates around the $-z$ axis. The volume mesh has approximately $1.5 \cdot 10^6$ vertices distributed over 10 computational blocks. It is available as part of the Numeca software suite ([82]). The tip gap is discretized using 21 vertices, there are 201 vertices around the blade and 77 vertices in the span-wise direction. The target y^+ is 1.0. In order to simplify the mesh generation process, sub-faces were introduced. Due to the presence of the tip gap, removing such sub-faces by splitting the corresponding blocks turned out to be troublesome: a large number of very small blocks was obtained. Therefore, it has been decided to keep the sub-faces and to use only the second order scheme, for which the sub-faces do not pose stability problems. Furthermore, the standard artificial dissipation operators have been used.

4.8.3 Results

The solution obtained with `SUmb` is used as initial solution and for this test case grid sequencing is not used. The convergence history is presented in fig. 4.34. In particular, fig. 4.34a shows the convergence histories for the norms of the dimensional components of the residuals; fig. 4.34b shows the norm of the corresponding scaled residual vector together with the CFL number used for the damping of the Newton method (see section 3.4.2). Since grid sequencing has not been used, a very tight convergence criterion was set. In the first 20 iterations the turbulence equation dominates the L^2 -norm of the scaled residual vector. It is only after this

Figure 4.32: Blade of Rotor37.

Figure 4.33: Surface mesh used for Rotor37. The corresponding volume mesh has $\approx 1.5 \cdot 10^6$ vertices.

initial phase that the scaled residual vector becomes representative of all equations, which then are effectively solved in a coupled manner. Such initial phase is typically avoided when using grid sequencing, which allows to choose a more relaxed convergence criterion (for instance, 4-5 orders of magnitude, as for the multi-element airfoil). The CPU time required for this test case was 24 hours on 8 processors, and 150 Gb of memory were needed.

The pitchwise-(mass-weighted) average of the total pressure and the total temperature at the outlet are plotted versus spanwise location in fig. 4.35, which also shows the experimental data as well as the numerical results obtained by other researchers with RANS simulations (yet other numerical results can be found in [42, 26]). The trends of total pressure and total temperature are generally correctly predicted, with the notable difference of the temperature close to the casing, see fig. 4.35b. The total temperature in the region 90 to 100% of the span is over-predicted, up to a factor of 2% with respect to the experiments. Such discrepancies, however, are found in *all* of the aforementioned studies. On the other hand, the distribution of the total pressure has the correct trend but quantitatively generally under-predicted; the maximum deviation from the experiments is of 12% (in the tip region). Near the hub the measurements of total pressure show a region of low total pressure at 20% of the blade span. This total pressure deficit has been subject of debate in the research community and several theories have been proposed. Shabbir [100] provided evidence, both numerically and experimentally, that the total pressure loss is caused by the hub leakage flow close to the leading edge of the blade. This leakage flow consists of a flow entering the primary flow passage through the gap between the disc rotor and the (non-moving) annulus upstream of the rotor. This gap is present in the actual machine but is usually not modeled in CFD simulations. In the present work, the gap and the corresponding leakage is

(a) Dimensional components of the residual vector

(b) Normalized, scaled residual, and damping CFL number

Figure 4.34: Rotor37: convergence history. (a) L^2 -norm of the dimensional (unscaled) components of residual. (b) L^2 -norm of the corresponding normalized, scaled residual vector and the damping CFL (see eq. (3.78)).

Figure 4.35: Rotor37: radial distributions at outlet. Comparison between experimental and numerical results obtained with RANS simulations.

not modeled either. Chima [16] noted that central difference schemes tend to smear out the distribution of the total pressure in that region. Hah [42] showed the presence of flow separation in the corner hub close to the trailing edge on the suction side of the blade, both in his RANS and LES results. According to Hah, this flow separation causes the total pressure deficit. The presence of flow separation in the corner hub is reported also in [26]. An indication that the corner flow separation might cause the total pressure deficit could come from the LES results of Gomar *et al.* [37] (not reproduced here). In fact, in [37] the flow separation in the corner is not present, and the total pressure deficit is not predicted either. Nevertheless, a general consensus is yet to be found.

The mid-span region is characterized by a strong shock originating at the blade leading edge. This shock interacts with the boundary layer on the suction side of the blade. As can be seen in fig. 4.36, which shows the skin friction stream lines, this interaction induces the separation of the boundary layer at $\approx 40\%$ of the chord. Moreover, it is observed that there is a radial flow from the hub to the tip in the region with separated flow. The shock can be seen in fig. 4.38, which shows the relative Mach number at 95% of span as well as a comparison with the experiments. This comparison shows a relatively good agreement, although, as expected since the SBP-SAT schemes have not been designed for shock-capturing

Figure 4.36: Rotor37: skin friction lines on the suction side.

and no ad-hoc treatment has been used, some oscillations are clearly visible close to the shock, fig. 4.39a. These oscillations are not present in the SUMb solution, see fig. 4.39b, which employs the JST scheme with an active shock capturing term. The tip leakage vortex highlighted by the experiments (fig. 4.38b) is shown in fig. 4.37.

Figure 4.37: Rotor37: tip leakage vortex.

(a) Present study, 2nd order solution.

(b) Experimental results. The dashed line indicates the trajectory of the tip leakage vortex.

Figure 4.38: Rotor37: comparison between computational and experimental results [107, 108]. Contours of the relative Mach number at 95% of span.

(a) Present study, 2nd order solution.

(b) SUMb, JST scheme.

Figure 4.39: Rotor37: contours of the relative Mach number at 95% of span, zoom-in at leading edge. (a) present study results; (b) SUMb [124] with JST scheme.

4.9 EFFICIENCY OF THE PARALLEL COMPUTATION

Scalability tests have been performed with explicit and implicit time-stepping algorithms. Figure 4.40 shows the strong scalability of the explicit algorithm for the simulation of the Taylor-Green vortex, on the finest grid. Note that the eight order scheme is used, thus a relatively large amount of data is exchanged between the processors. This test, which consists of the comparison of the computing time for 100 residual evaluations, has been performed on the JUQUEEN cluster, the Blue Gene/Q machine of the Jülich supercomputing center (Germany). The scalability is excellent up to 4096 processors; then, the performance deteriorates probably due to communication overhead. The results of the scalability tests for the implicit algorithm are shown in fig. 4.41. The tests have been carried out for the Rotor37 test case and for the three-element airfoil test case, which have been run on the Cartesius machine of SurfSARA, the Netherlands Supercomputer Center. The test consists of the solution of one linear system up to a predefined accuracy; the computation and the factorization of the preconditioner have been taken into account. Preprocessing steps, like memory allocation and the computation of the dummy Jacobian are excluded. Note that while the computation of the Jacobian/preconditioner is part of the present method, the factorization of the preconditioner and the solution of the linear system is carried out via the external library PETSc. As seen for the explicit algorithm the scalability reduces after a certain number of processors. The performance of the parallel algorithm is better in the case of the three-element airfoil for which a slight super-linear scalability is observed. Overall, satisfactory results have been obtained.

Figure 4.40: Taylor-Green vortex, $1.35 \cdot 10^8$ nDOFs, 8th order scheme: strong scalability with explicit time stepping (t.s.) on JUQUEEN, Jülich, Germany.

(a) Rotor37, fine grid ($\approx 1.5 \cdot 10^6$ nDOFs). (b) Three-element airfoil, fine grid ($\approx 2 \cdot 10^6$ nDOFs).

Figure 4.41: Strong scalability with implicit time stepping (t.s.) on Cartesius of SurfSara, Netherlands.

5

CONCLUDING REMARKS AND RECOMMENDATIONS

High-order ($\geq 3^{\text{rd}}$) schemes are receiving a constantly growing attention thanks to their potential in delivering a higher computational efficiency, which means less computational work for a given accuracy. In the present research high-order energy-stable finite-difference (SBP-SAT) schemes have been investigated and employed for the numerical simulation of a wide range of types of compressible flows. For the linearized equations governing a smooth flow, these schemes are provably stable schemes. Numerical simulations for inviscid flow, laminar flow, as well as simulations based on DNS, LES and RANS have been performed using a combination of high-order implicit and explicit time-stepping techniques, which have been developed in a computational method for parallel computing. The next sections presents the conclusions of this investigation and the recommendations for future research.

5.1 CONCLUDING REMARKS

For smooth external flows, throughout this work the SBP-SAT high-order schemes have consistently provided more accurate solutions than low-order schemes (for a given number of degrees of freedom). The high-order schemes were also more efficient in terms of required CPU time for all types of simulations, from inviscid flows to simulations based on RANS. Very good results have been obtained, especially for the Euler vortex (section 4.1), and for the more complicated 3D test cases, i. e. the analytical slender body (section 4.4) and the DNS of the Taylor Green vortex (section 4.5), which clearly show the gain in efficiency when using high-order schemes. Also the results of high order schemes applied to (I)LES of transitional flow over the SD7003 wing (section 4.6) are of high quality, but for this case no comparisons have been made with a standard 2nd order scheme due to the excessive computational requirements for this case.

For the simulation based on RANS of the flow over the three-element airfoil in high-lift configuration (section 4.7), however, the gain in efficiency achieved by the high-order schemes in combination with implicit time-stepping is not that clear. On the one hand, the higher order SBP-SAT schemes yield a more accurate solution than the second order scheme. The fourth order scheme, in particular, turned out to be also more computationally efficient than the second and third order scheme. (The convergence rates of the lift and drag coefficients were below the design orders, though.) On the other hand, the comparison with an industry-standard second-order finite-volume method suggests the following two consider-

ations. Firstly, the overall accuracy of the SBP-SAT schemes, at least measured in terms of integrated quantities as lift and drag, was not as high as expected. In fact, the second-order finite volume method yielded results in between the results of the third and those of the fourth order SBP-SAT method (on a given grid). Given the results of all other test cases, the most likely reasons for this discrepancy are the large number of computational blocks and the sub-optimal smoothness of the grid. Both are known to deteriorate the accuracy of high-order finite difference schemes. Secondly, solving the turbulence and the flow equations fully coupled with implicit schemes (section 3.4), may not be the most efficient iteration strategy for RANS problems. Nevertheless, we believe there is still potential for improvement (see the recommendations in the next section). The finite-volume method, which solves the flow and the turbulence equations in a decoupled manner with semi-implicit schemes, turned out to be computationally more efficient than the present SBP-SAT method. The highly non-linear nature of the turbulence equations forces to introduce quite some damping in Newton's method, used to solve the fully coupled system. This damping deteriorates the convergence rate of the algorithm and slow convergence was observed on the finer grids. However, it is stressed that the same iteration strategy was very efficient for inviscid flow and laminar (steady-state) viscous flow test cases. For these types of flows, in fact, typically no damping is needed and grid-sequencing, combined with some explicit smoothing at the beginning of the iterative procedure on each grid level, provides an adequate initial guess for Newton's method. Moreover, note that the present research code does not perform at peak efficiency. When a thorough optimization is carried out, it is likely that at least a factor of two can be gained, in terms of CPU time.

The numerical simulation of non-smooth/internal flows (the linear cascade in section 4.2, the NACA0012 in section 4.3 and Rotor37 in 4.8) highlighted the need of further development of the SBP-SAT framework. Within this framework, three main building blocks are still missing: consistent shock capturing operators, generalized interpolation operators, and boundary conditions for internal flows. Regarding the first missing building block, an attempt has been made to incorporate some shock-capturing capabilities in the schemes. Following the approach presented in [24] (inspired by the work in [55]), modified artificial dissipation operators were implemented and tested for the transonic flow over the NACA0012 airfoil. While reasonable results were obtained, the solid mathematical base of linear-stability was lost and convergence issues were observed. It is worth mentioning that some research in this direction is under development and techniques combining shock capturing and entropy conservation showed promising results (see the recommendations in the next section). Nevertheless, applications to non-academic problems are yet to be demonstrated. About the second missing building block, let us define generalized interpolation operators as operators able to handle both sub-faces (interfaces in conforming meshes) and sliding interfaces (i. e. non-conforming meshes). Both these interfaces pose stability problems for all but the second order scheme, which can be used in a relatively straightforward manner. The extension to high-order schemes is ongoing research. Most notably, in [79] sub-faces are treated in a

consistent way, although applications are again limited to academic test cases. As seen for the Rotor37 test case, avoiding sub-faces complicated the mesh generation process so much that it was decided to keep the sub-faces and to only compute the second order solution. In sections 4.1.2 and 4.2, high-order interpolation operators (derived in [35], see also [71]) have been applied at sliding mesh interfaces and promising results have been obtained. The solution was smooth across the interface and the design accuracy was verified. However, the assumptions used to construct such operators pose severe limitations on the computational mesh. For instance, in order to comply with these limitations, sub-faces were introduced in the mesh for the linear cascade (section 4.2). Thus, for turbomachinery test cases, which typically consist of complex geometries with parts in relative motion, this becomes a vicious circle, for which the lack of generalized interpolation operators in practice precludes the use of high-order finite difference SBP-SAT schemes and restricts the accuracy to second order. Additionally, the absence of boundary conditions for internal flows, the third missing building block, forces the user to first obtain the solution at inlet and outlet with another method (that uses a different discretization technique), and then to compute the solution with the SBP-SAT schemes. This process hampers the efficiency and the stability of the whole approach. Only once these limitations will be overcome, real-life turbomachinery problems will be within reach.

5.2 RECOMMENDATIONS

The best results have been obtained for smooth external flows about relatively simple geometries. Therefore, for these types of flows, the use of the method developed in the present work is recommended and encouraged. The method is particularly well suited for inviscid flow, laminar viscous flow, as well as for methods based on DNS and (I)LES, for which implicit and explicit time-stepping can be used. While numerical simulations based on RANS have been successfully performed, it is for this kind of simulations that we believe there is the most room for improvement. In particular, the iteration technique may be revised and the following suggestions are forwarded:

- investigate more advanced scalings of the linear system in Newton's method. The scaling significantly affects the robustness of the solution process. For instance, as suggested in [17], at each Newton iteration the linear system could be scaled such that the maximum difference, in terms of norm of the residual, of the mean-flow equations and the equation of the turbulence model is within an order of magnitude. A better scaling of the linear system would also allow to reduce the level of fill-in of the ILU preconditioner, thus lowering the memory requirements.
- investigate other globalization and CFL evolution strategies. In the present work pseudo-transient continuation and a simple SER strategy were chosen

but other options are available in literature, see the dissipation-based continuation in [45] and the physics-based continuation in [14].

- investigate other iterative techniques. For example, one could decouple the solution of the flow and the turbulence equations, and/or use a low-order scheme for the equation of the turbulence model and a high-order scheme for the mean flow equations.

When solving the URANS equations with the high-order implicit time integration schemes (ESDIRK), the iterative technique developed in this thesis gives satisfactory results (not shown in this work). Contrary to steady-state flows for which an appropriate initial guess is difficult to obtain, the discretization of the equations for unsteady flows leads to relatively well-conditioned systems. Nevertheless, the suggestions given for RANS can be beneficial for URANS too. Furthermore, since only preliminary results have been obtained with the ESDIRK schemes, further testing is encouraged.

It is believed that in the future, thanks to the ever increasing computational power, the modeling approach of LES will be applied to larger and larger problems on massively parallel architectures [103]. In this respect, it may be useful to implement wall functions for the computation of wall-modeled LES (see [89] for a description of various wall functions). When using standard second order schemes, for the wall-resolved LES the grid requirements scale proportionally to $Re^{13/7}$ [18]; not too far from DNS for which the scaling factor is $Re^{9/4}$ in a unconfined flow [90]. For wall-modeled LES, instead, the number of required grid points is proportional to Re [18]. Hence, for relatively high Reynolds number, wall functions could significantly lower the turn-around time of a LES simulation. At lower Reynolds number, the differences in cost of wall-modeled and wall-resolved LES becomes small. However, this estimate does not take into account the use of high-order schemes nor the laminar-to-turbulent transition zone, which may both have significant effects [103]. An indication of the possible savings in terms of CPU time deriving from the use of high-order schemes might come from our DNS results of the Taylor-Green vortex (section 4.5). For that test case, the higher order scheme on a coarse mesh provided a more accurate solution than the second order scheme on the finer mesh, leading to a 5 times smaller computational cost. Note that the higher order scheme was *more* accurate than (and not as accurate as) the second-order scheme, thus a gain of a factor of more than 5 could be expected.

The computational method developed in the present thesis is not mature enough for non-smooth/internal flows. The absence of the three building blocks, mentioned above, poses too severe limitations for real-life applications. Thus, further development in the SBP-SAT framework is needed. Regarding consistent shock capturing techniques, Fisher and Carpenter [31] have shown promising results using SBP-SAT schemes that prevent entropy from growing unbounded (consequently called entropy stable schemes). Therefore, it is suggested to test these schemes on non-academic test cases. Note, however, that even though entropy stable boundary conditions have been developed (slip/no-slip walls in [87], discontinuous interfaces in [88]; an entropy stable far-field boundary condition is available only for the Eu-

ler equations, [116]), the line of research of entropy stable schemes has moved from finite-difference schemes to spectral collocation schemes (see also [11] and references therein).

Generalized, consistent, high-order interpolation operators are another building block that we would like to have in the SBP-SAT framework. This kind of high-order interpolation operators, besides being necessary for turbomachinery problems, would have the added benefit of allowing a more flexible computational mesh for complex geometries in general. Being able to handle sub-faces in a stable way would be already beneficial. Thus it is suggested to implement the approach presented in [79] for this kind of interfaces. An alternative approach to alleviate the burden of mesh generation for complex geometries would be the use of overset meshes and high-order interpolation, see for instance [101]. This, however, would most likely lead to non-provably stable schemes and hence more artificial dissipation would be needed. The question is how much and how this will affect the accuracy of the solution.

Finally, researchers are encouraged to pursue the development of consistent boundary conditions for internal flows. So far these types of boundary conditions have not been considered in the SBP-SAT framework, but they are essential in order to extend the range of applicability of these schemes to turbomachinery flow problems.

BIBLIOGRAPHY

- [1] 2nd International Workshop on High-Order CFD Methods, Cologne, Germany, May 2013. <http://www.dlr.de/as/hiocfd>. [Cit. on pages 85, 98, 103, 105 and 113]
- [2] 3rd International Workshop on High-Order CFD Methods, Kissimmee, Orlando, USA, January 2015. <https://www.grc.nasa.gov/hiocfd/>. [Cit. on pages 85, 87, 94, 113 and 115]
- [3] M. J. Aftosmis. Solution adaptive cartesian grid methods for aerodynamic flows with complex geometries. *VKI Lecture Series*, March 1997. [Cit. on page 74]
- [4] S. R. Allmaras, F. T. Johnson, and P. R. Spalart. Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. In *7th International Conference on Computational Fluid Dynamics*, number ICCFD7-1902, Big Island, Hawaii, 9-13 July 2012. [Cit. on pages 24, 25, 26, 30 and 120]
- [5] J. D. Anderson. *Fundamentals of Aerodynamics*. McGraw-Hill Education, fifth edition, 2010. [Cit. on page 96]
- [6] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, B. Smith, and H. Zhang. PETSc users manual. Technical Report Revision 3.2, Argonne National Laboratory, 2011. [Cit. on pages 67, 69 and 83]
- [7] H. Bijl, M. Carpenter, and V. Vatsa. Time integration schemes for the unsteady Navier-Stokes equations. In *AIAA paper 2001-2612, 15th AIAA Computational Fluid Dynamics Conference*, pages 1-11, Anaheim, CA, 11 2001. [Cit. on page 51]
- [8] J. Bonet and J. Peraire. An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems. *International Journal for Numerical Methods in Engineering*, 31(1):1-17, 1991. [Cit. on page 77]
- [9] H. M. Bücker, B. Pollul, and A. Rasch. On CFL evolution strategies for implicit upwind methods in linearized euler equations. *International journal for numerical methods in fluids*, 59(1):1-18, 2009. [Cit. on page 58]
- [10] M. H. Carpenter, D. Gottlieb, and S. Abarbanel. Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: Methodology and application to high-order compact schemes. *Journal of Computational Physics*, 111(2):220 - 236, 1994. [Cit. on pages 9, 32 and 33]

- [11] M. H. Carpenter, M. Parsani, T. C. Fisher, and E. J. Nielsen. Entropy stable staggered grid spectral collocation for the burgers' and compressible Navier-Stokes equations. Technical Memorandum NASA/TM-2015-218990, NASA, December 2015. [Cit. on page 133]
- [12] I. B. Celik, U. Ghia, P. J. Roache, C. J. Freitas, H. Coleman, and P. E. Raad. Procedure for estimation and reporting of uncertainty due to discretization in CFD applications. *Journal of Fluids Engineering*, 130(7):078001–078001, Jul 2008. [Cit. on page 97]
- [13] M. Ceze and K. J. Fidkowski. Drag prediction using adaptive discontinuous finite elements. *Journal of Aircraft*, 51(4):1284–1294, 2014. [Cit. on page 70]
- [14] M. Ceze and K. J. Fidkowski. Constrained pseudo-transient continuation. *International Journal for Numerical Methods in Engineering*, 102(11):1683–1703, 2015. [Cit. on pages 58 and 132]
- [15] W. M. Chan and J. L. Steger. Enhancements of a three-dimensional hyperbolic grid generation scheme. *Applied Mathematics and Computation*, 51(2-3):181 – 205, 1992. [Cit. on page 99]
- [16] R. V. Chima. SWIFT code assessment for two similar transonic compressors. Technical report, NASA Glenn Research Center, 2009. Also AIAA paper 2009-1058. [Cit. on pages 119 and 123]
- [17] T. T. Chisholm and D. W. Zingg. A jacobian-free Newton–Krylov algorithm for compressible turbulent fluid flows. *Journal of Computational Physics*, 228(9):3490–3507, 2009. [Cit. on pages 70 and 131]
- [18] H. Choi and P. Moin. Grid-point requirements for large eddy simulation: Chapman's estimates revisited. *Physics of Fluids*, 24(1), 2012. [Cit. on page 132]
- [19] T. J. Chung. *Computational Fluid Dynamics*. Cambridge University Press, 2002. [Cit. on page 8]
- [20] H. Cohen, G. F. C. Rogers, and H. I. H. Saravanamuttoo. *Gas Turbine Theory*. Longman Group Limited, fourth edition, 1996. [Cit. on pages 2 and 3]
- [21] H. de Vries. *On Synthetic Jet Actuation for Aerodynamic Load Control*. PhD thesis, University of Twente, 2013. [Cit. on pages 15, 47 and 160]
- [22] J. Dennis and R. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, Philadelphia, 1996. [Cit. on page 72]
- [23] J. Denton. Lessons from Rotor 37. *Journal of Thermal Science*, 6:1–13, 1997. 10.1007/s11630-997-0010-9. [Cit. on page 119]

- [24] S. C. Dias and D. W. Zingg. A high-order parallel Newton–Krylov flow solver for the Euler equations. In *19th AIAA Computational Fluid Dynamics*, June 22–25 2009. AIAA paper 2009-3657. [Cit. on pages 48 and 130]
- [25] J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodríguez-Ferran. *Arbitrary Lagrangian-Eulerian methods*. John Wiley & Sons, Ltd, 2004. [Cit. on page 16]
- [26] J. Dunham. CFD validation for propulsion system components (la validation CFD des organes des propulseurs). AGARD Advisory Report 355, Advisory Group for Aerospace Research and Development, 1998. [Cit. on pages 118, 119, 120, 121 and 123]
- [27] J. W. Eaton, D. Bateman, and S. Hauberg. *GNU Octave version 3.0.1 manual: a high-level interactive language for numerical computations*. CreateSpace Independent Publishing Platform, 2009. ISBN 1441413006. [Cit. on page 91]
- [28] K. Ekici and K. C. Hall. Nonlinear analysis of unsteady flows in multistage turbomachines using harmonic balance. *AIAA journal*, 45(5):1047–1057, 2007. [Cit. on page 90]
- [29] D. C. D. R. Fernández, J. E. Hicken, and D. W. Zingg. Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations. *Computers & Fluids*, 95:171–196, 2014. [Cit. on page 32]
- [30] J. Fike, S. Jongsma, J. Alonso, and E. van der Weide. Optimization with gradient and hessian information calculated using hyper-dual numbers. In *AIAA Paper 2011-3807*, 2011. [Cit. on pages 58 and 83]
- [31] T. C. Fisher and M. H. Carpenter. High-order entropy stable finite difference schemes for nonlinear conservation laws: Finite domains. *Journal of Computational Physics*, 252(0):518 – 557, 2013. [Cit. on pages 47 and 132]
- [32] D. J. Garmann and M. R. Visbal. C3.3: Implicit Large Eddy-Simulations of transitional flow over the SD7003 airfoil using compact finite-differencing and filtering. http://www.dlr.de/as/Portaldata/5/Resources/dokumente/veranstaltungen/2013_05-hiocfd/contrib/C33_Garmann.pdf, 2013. Accessed: January 20, 2016. [Cit. on pages 107 and 110]
- [33] E. Garnier, N. Adams, and P. Sagaut. *Large eddy simulation for compressible flows*. Springer Science & Business Media, 2009. [Cit. on page 28]
- [34] J. Gatsis. *Preconditioning Techniques for a Newton-Krylov Algorithm for the Compressible Navier-Stokes Equations*. PhD thesis, University of Toronto, 2013. [Cit. on page 70]
- [35] G. Giangaspero, M. Almquist, K. Mattsson, and E. van der Weide. Unsteady simulations of rotor stator interactions using SBP-SAT schemes: status and challenges. In R. M. Kirby, M. Berzins, and J. S. Hesthaven, editors, *Spectral*

- and High Order Methods for Partial Differential Equations - ICOSAHOM '14*, volume 106 of *Lecture Notes in Computational Science and Engineering*. Springer International Publishing, 2015. [Cit. on pages 44, 85 and 131]
- [36] G. Giangaspero and E. van der Weide. A stable and conservative high-order solver for the Reynolds-Averaged Navier-Stokes equations. In *3rd ECCOMAS Young Investigators Conference*, Aachen, Germany, July 2015. [Cit. on page 85]
 - [37] A. Gomar, N. Gourdain, and G. Dufour. High-fidelity simulation of the turbulent flow in a transonic axial compressor. In *European Turbomachinery Conference (ETC9)*, 2011. [Cit. on pages 119 and 123]
 - [38] S. Gottlieb, C. Shu, and E. Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Review*, 43(1):89–112, 2001. [Cit. on page 50]
 - [39] W. Gropp, D. Keyes, L. C. McInnes, and M. Tidriri. Globalized Newton-Krylov-Schwarz algorithms and software for parallel implicit CFD. *International Journal of High Performance Computing Applications*, 14(2):102–136, 2000. [Cit. on pages 56, 57 and 71]
 - [40] W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith. High-performance parallel implicit CFD. *Parallel Computing*, 27(4):337–362, 2001. [Cit. on pages 71 and 72]
 - [41] B. Gustafsson, H.-O. Kreiss, and J. Oliger. *Time dependent problems and difference methods*. John Wiley & Sons, Inc., 1995. [Cit. on pages 34 and 35]
 - [42] C. Hah. Large eddy simulation of transonic flow field in NASA Rotor 37. NASA/TM 2009-215627, NASA, September 2009. [Cit. on pages 119, 121 and 123]
 - [43] S. Hamdi, W. E. Schiesser, and G. W. Griffiths. Method of lines. *Scholarpedia*, 2(7):2859, 2007. revision #124335. [Cit. on page 31]
 - [44] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Texts in Applied Mathematics. Springer New York, first edition, 2008. [Cit. on page 8]
 - [45] J. E. Hicken, H. Buckley, M. Osusky, and D. W. Zingg. Dissipation-based continuation: a globalization for inexact-Newton solvers. In *20th AIAA Computational Fluid Dynamics Conference*, page 3237, 2011. [Cit. on pages 58 and 132]
 - [46] J. E. Hicken and D. W. Zingg. Superconvergent functional estimates from summation-by-parts finite-difference discretizations. *SIAM Journal on Scientific Computing*, 33(2):893–922, 2011. [Cit. on page 99]
 - [47] J. E. Hicken and D. W. Zingg. Dual consistency and functional accuracy: a finite-difference perspective. *Journal of Computational Physics*, 256:161–182, Jan. 2014. [Cit. on page 99]

- [48] K. A. Hoffmann and S. T. Chiang. *Computational Fluid Dynamics*, volume 1. Engineering Education System, fourth edition, 2000. [Cit. on page 13]
- [49] K. A. Hoffmann and S. T. Chiang. *Computational Fluid Dynamics*, volume 2. Engineering Education System, fourth edition, 2000. [Cit. on pages 18 and 19]
- [50] K. A. Hoffmann and S. T. Chiang. *Computational Fluid Dynamics*, volume 3. Engineering Education System, fourth edition, 2000. [Cit. on page 28]
- [51] J. C. Hunt, A. Wray, and P. Moin. Eddies, streams, and convergence zones in turbulent flows. In *Proceedings of the 1988 Summer Program*, volume 1, pages 193–208. Center for Turbulence Research, Stanford University, 1988. [Cit. on pages 102 and 107]
- [52] P. Inc. *Pointwise User Manual*, 2011. www.pointwise.com. [Cit. on pages 94, 99 and 107]
- [53] F. Incropera, D. DeWitt, T. Bergman, and A. Lavine. *Fundamentals of Heat and Mass Transfer*. Wiley, 6th edition, 2007. [Cit. on page 15]
- [54] A. Jameson. Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. *AIAA paper*, 91(1956):1991, June 1991. [Cit. on page 52]
- [55] A. Jameson, W. Schmidt, and E. Turkel. Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes. In *AIAA 14th Fluid and Plasma Dynamic Conference*, Palo Alto, CA, June 23-25 1981. AIAA paper 1981-1259. [Cit. on pages 47, 48, 50, 120 and 130]
- [56] S. H. Jongsma. *On a Method for Simulation-Based Wind Turbine Blade Design*. PhD thesis, University of Twente, 2014. [Cit. on pages 58, 60, 67, 70 and 74]
- [57] C. T. Kelley and D. E. Keyes. Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis*, 35(2):508–523, 1998. [Cit. on page 56]
- [58] C. A. Kennedy and M. H. Carpenter. Additive Runge–Kutta schemes for convection–diffusion–reaction equations. *Applied Numerical Mathematics*, 44:139–181, 2003. [Cit. on pages 51 and 169]
- [59] D. Knoll and D. Keyes. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004. [Cit. on page 56]
- [60] H.-O. Kreiss and G. Scherer. Finite element and finite difference methods for hyperbolic partial differential equations. *Mathematical Aspects of Finite Elements in Partial Differential Equations.*, Academic Press, Inc., 1974. [Cit. on page 9]

- [61] H.-O. Kreiss and G. Scherer. On the existence of energy estimates for difference approximations for hyperbolic systems. Technical report, Dept. of Scientific Computing, Uppsala University, 1977. [Cit. on page 9]
- [62] H. O. Kreiss and L. Wu. On the stability definition of difference approximations for the initial boundary value problem. *Applied Numerical Mathematics*, 12(1-3):213 – 227, 1993. [Cit. on page 36]
- [63] N. Kroll, T. Leicht, C. Hirsch, F. Bassi, C. Johnston, K. Sorensen, and K. Hillewaert. Results and conclusions of the european project IDIHOM on high-order methods for industrial aerodynamic applications. In *53rd AIAA Aerospace Sciences Meeting*, AIAA SciTech. American Institute of Aeronautics and Astronautics, Jan 2015. AIAA 2015-0293. [Cit. on pages 8 and 9]
- [64] P. D. Lax and R. D. Richtmyer. Survey of the stability of linear finite difference equations. *Communications on Pure and Applied Mathematics*, 9(2):267–293, 1956. [Cit. on page 33]
- [65] S. K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, 1992. [Cit. on pages 5 and 10]
- [66] R. J. LeVeque. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002. [Cit. on page 6]
- [67] D. Levy and E. Tadmor. From semidiscrete to fully discrete: Stability of Runge–Kutta schemes by the energy method. *SIAM Review*, 40(1):40–73, 1998. [Cit. on page 36]
- [68] V. Liseikin. *Grid Generation Methods*. Scientific computation. Springer, 2009. [Cit. on page 27]
- [69] K. Mattsson. Summation by parts operators for finite difference approximations of second-derivatives with variable coefficients. *Journal of Scientific Computing*, 51(3):650–682, 2012. [Cit. on pages 33 and 37]
- [70] K. Mattsson and M. Almquist. A solution to the stability issues with block norm summation by parts operators. *Journal of Computational Physics*, 253(0):418 – 442, 2013. [Cit. on pages 37 and 46]
- [71] K. Mattsson and M. H. Carpenter. Stable and accurate interpolation operators for high-order multiblock finite difference methods. *SIAM Journal on Scientific Computing*, 32(4):2298–2320, 2010. [Cit. on pages 36, 44, 45 and 131]
- [72] K. Mattsson and J. Nordström. Summation by parts operators for finite difference approximations of second derivatives. *Journal of Computational Physics*, 199(2):503 – 540, 2004. [Cit. on pages 37 and 149]
- [73] K. Mattsson, M. Svärd, and J. Nordström. Stable and accurate artificial dissipation. *Journal of Scientific Computing*, 21(1):57–79, 2004. [Cit. on pages 9, 33, 45 and 154]

- [74] K. Mattsson, M. Svärd, and M. Shoneybi. Stable and accurate schemes for the compressible Navier-Stokes equations. *Journal of Computational Physics*, 227(4):2293 – 2316, 2008. [Cit. on pages 33 and 37]
- [75] Maxima. Maxima, a computer algebra system. version 5.34.1, 2014. [Cit. on pages 159 and 163]
- [76] G. May, F. Iacono, and A. Jameson. Efficient algorithms for high-order discretizations of the Euler and Navier-Stokes equations. In *47th AIAA Aerospace Sciences Meeting*, number 2009-0182, 2009. [Cit. on page 58]
- [77] C. Morris. Studies of inviscid flux schemes for acoustics and turbulence problems. *AIAA paper 2013-0075*, 2013. [Cit. on page 102]
- [78] F. Nicoud and F. Ducros. Subgrid-scale stress modelling based on the square of the velocity gradient tensor. *Flow, Turbulence and Combustion*, 62(3):183–200, 1999. [Cit. on page 29]
- [79] A. Nissen, K. Kormann, M. Grandin, and K. Virta. Stable difference methods for block-oriented adaptive grids. *Journal of Scientific Computing*, pages 1–26, 2014. [Cit. on pages 90, 130 and 133]
- [80] J. Nordström, J. Gong, E. van der Weide, and M. Svärd. A stable and conservative high order multi-block method for the compressible Navier-Stokes equations. *Journal of Computational Physics*, 228(24):9020 – 9035, 2009. [Cit. on pages 9, 10, 33, 36, 43, 73 and 156]
- [81] J. Nordström and M. Svärd. Well-posed boundary conditions for the Navier-Stokes equations. *SIAM J. Numer. Anal.*, 43:1231–1255, March 2005. [Cit. on pages 9, 29, 30 and 33]
- [82] NUMECA. www.numeca.com, 2012. [Cit. on page 120]
- [83] M. Osusky, P. D. Boom, and D. W. Zingg. Results from the fifth AIAA Drag Prediction Workshop obtained with a parallel Newton-Krylov-Schur flow solver discretized using summation-by-parts operators. In *31st AIAA Applied Aerodynamics Conference*, June 24-27 2013. AIAA 2013-2511. [Cit. on page 10]
- [84] M. Osusky, J. E. Hicken, and D. W. Zingg. A parallel Newton-Krylov-Schur flow solver for the Navier-Stokes equations using the SBP-SAT approach. In *48th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, Florida, AIAA 2010-116*, 2010. [Cit. on page 156]
- [85] M. Osusky and D. W. Zingg. A parallel Newton-Krylov-Schur flow solver for the Reynolds-Averaged Navier-Stokes equations. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, January 2012. AIAA 2012-0442. [Cit. on pages 10, 41 and 42]

- [86] D. Papadogiannis. *Coupled Large Eddy Simulations of combustion chamber-turbine interactions*. PhD thesis, Institut National Polytechnique de Toulouse, 2015. [Cit. on page 3]
- [87] M. Parsani, M. H. Carpenter, and E. J. Nielsen. Entropy stable wall boundary conditions for the compressible Navier-Stokes equations. Technical Memorandum NASA/TM-2014-218282, NASA, December 2014. [Cit. on page 132]
- [88] M. Parsani, M. H. Carpenter, and E. J. Nielsen. Entropy stable discontinuous interfaces coupling for the three-dimensional compressible Navier-Stokes equations. *Journal of Computational Physics*, 290:132 – 138, 2015. [Cit. on page 132]
- [89] U. Piomelli and E. Balaras. Wall-layer models for large-eddy simulations. *Annual Review of Fluid Mechanics*, 34(1):349–374, 2002. [Cit. on page 132]
- [90] S. B. Pope. *Turbulent flows*. Cambridge University press, 2000. [Cit. on pages 22, 23 and 132]
- [91] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007. [Cit. on pages 50 and 54]
- [92] T. Pulliam and D. Chaussee. A diagonal form of an implicit approximate-factorization algorithm. *Journal of Computational Physics*, 39(2):347 – 363, 1981. [Cit. on page 159]
- [93] T. Pulliam and J. Steger. Implicit finite-difference simulations of three-dimensional compressible flow. *AIAA Journal*, 18(2):159–167, 1980. [Cit. on page 17]
- [94] L. Reid and R. D. Moore. Design and overall performance of four highly loaded, high speed inlet stages for an advanced high-pressure-ratio core compressor. *NASA-TP-1337*, 1:1–119, 1978. [Cit. on page 118]
- [95] P. J. Roache. Quantification of uncertainty in computational fluid dynamics. *Annual Review of Fluid Mechanics*, 29(1):123–160, 1997. [Cit. on page 97]
- [96] P. Roe. Fluctuations and signals, a framework for numerical evolution problems. *Numerical Methods for Fluid Dynamics*, 11, 1982. [Cit. on page 114]
- [97] C. L. Rumsey, B. R. Smith, and G. P. Huang. Description of a website resource for turbulence model verification and validation. Technical report, Tech. Rep. AIAA 2010-4742, NASA Langley Research Center, 2010. See also: <http://turbmodels.larc.nasa.gov/>. [Cit. on pages 24 and 171]
- [98] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial Mathematics, 2nd edition, 2003. [Cit. on pages 69, 70 and 71]

- [99] Y. Saad and M. H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986. [Cit. on page 69]
- [100] A. Shabbir, M. L. Celestina, J. J. Adamczyk, and A. J. Strazisar. The effect of hub leakage flow on two high speed axial flow compressor rotors. *ASME Paper 97-GT-346*, 1997. [Cit. on page 121]
- [101] S. E. Sherer and M. R. Visbal. Multi-resolution implicit large eddy simulations using a high-order overset-grid approach. *International Journal for Numerical Methods in Fluids*, 55(5):455–482, 2007. [Cit. on pages 9 and 133]
- [102] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439 – 471, 1988. [Cit. on page 50]
- [103] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis. CFD vision 2030 study: a path to revolutionary computational aerosciences. Contractor Report NASA/CR-2014-218178, NASA, March 2014. [Cit. on pages 1, 2, 9 and 132]
- [104] B. Strand. Summation by parts for finite difference approximations for d/dx . *Journal of Computational Physics*, 110(1):47 – 67, 1994. [Cit. on pages 9, 37, 149 and 151]
- [105] G. Strang. Accurate partial difference methods II. non-linear problems. *Num. Math.*, 6:37–46, 1964. [Cit. on pages 32 and 36]
- [106] B. Stroustrup. *The C++ Programming Language*. Addison-Wesley, 1997. [Cit. on page 67]
- [107] K. L. Suder. Experimental investigation of the flow field in a transonic, axial flow compressor with respect to the development of blockage and loss. Technical Report NASA Technical Memorandum 107310, NASA, 1996. [Cit. on pages 118, 119, 123 and 125]
- [108] K. L. Suder and M. L. Celestina. Experimental and computational investigation of the tip clearance flow in a transonic axial compressor rotor. *Journal of Turbomachinery*, 118(2):218–229, 1996. [Cit. on pages 118 and 125]
- [109] M. Svärd. On coordinate transformations for summation-by-parts operators. *J. Sci. Comput.*, 20:29–42, February 2004. [Cit. on page 37]
- [110] M. Svärd, M. H. Carpenter, and J. Nordström. A stable high-order finite difference scheme for the compressible Navier-Stokes equations, far-field boundary conditions. *Journal of Computational Physics*, 225(1):1020 – 1038, 2007. [Cit. on pages 9, 10, 32, 33, 36, 38, 39 and 40]
- [111] M. Svärd, K. Mattsson, and J. Nordstrom. Steady-state computations using summation-by-parts operators. *Journal of Scientific Computing*, 24(1):79–95, 2005. [Cit. on pages 33, 37, 149 and 151]

- [112] M. Svärd and S. Mishra. Entropy stable schemes for initial-boundary-value conservation laws. *Zeitschrift für angewandte Mathematik und Physik*, 63(6):985–1003, 2012. [Cit. on page 47]
- [113] M. Svärd and J. Nordström. On the order of accuracy for difference approximations of initial-boundary value problems. *Journal of Computational Physics*, 218(1):333 – 352, 2006. [Cit. on page 37]
- [114] M. Svärd and J. Nordström. A stable high-order finite difference scheme for the compressible Navier-Stokes equations: No-slip wall boundary conditions. *Journal of Computational Physics*, 227(10):4805 – 4824, 2008. [Cit. on pages 9, 10, 33, 36, 40, 41 and 42]
- [115] M. Svärd and J. Nordström. Review of summation-by-parts schemes for initial-boundary-value problems. *Journal of Computational Physics*, 268:17–38, 2014. [Cit. on pages 9, 10, 32, 33, 34 and 35]
- [116] M. Svärd and H. Özcan. Entropy stable schemes for the Euler equations with far-field and wall boundary conditions. *online first, Journal of Scientific Computing*, 2013. [Cit. on page 133]
- [117] J. L. Thomas and M. D. Salas. Far-field boundary conditions for transonic lifting solutions to the Euler equations. *AIAA Journal*, 24(7):1074–1080, Jul. 1986. [Cit. on pages 30 and 39]
- [118] P. D. Thomas and C. K. Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA Journal*, 17(10):1030–1037, Oct. 1979. [Cit. on page 21]
- [119] P. Tucker. Differential equation-based wall distance computation for DES and RANS. *Journal of Computational Physics*, 190(1):229 – 248, 2003. [Cit. on page 74]
- [120] P. Tucker. Computation of unsteady turbomachinery flows: Part 2—LES and hybrids. *Progress in Aerospace Sciences*, 47(7):546 – 569, 2011. [Cit. on page 2]
- [121] P. G. Tucker, C. L. Rumsey, P. R. Spalart, R. B. Bartels, and R. T. Biedron. Computations of wall distances based on differential equations. *AIAA journal*, 43(3):539–549, 2005. [Cit. on page 74]
- [122] E. van der Weide, G. Giangaspero, and M. Svärd. Efficiency benchmarking of an energy stable high-order finite difference discretization. *AIAA Journal*, 57(7):1845–1860, July 2015. doi: 10.2514/1.J053500. [Cit. on pages 32, 45 and 85]
- [123] E. van der Weide, A. Gopinath, and A. Jameson. Turbomachinery applications with the time spectral method. In *35th AIAA Fluid Dynamics Conference and Exhibit, Toronto, Ontario, Canada*, 2005. [Cit. on page 6]

- [124] E. van der Weide, G. Kalitzin, J. Schluter, and J. J. Alonso. Unsteady turbomachinery computations using massively parallel platforms. *AIAA paper*, 421:2006, 2006. [Cit. on pages 5, 76, 90, 114, 120 and 125]
- [125] E. van der Weide and M. Svärd. Abstract for the 1st international workshop on high-order CFD methods. http://dept.ku.edu/~cfdku/hio CFD/abstracts/C11_Weide.pdf, 2012. Accessed: January 20, 2016. [Cit. on page 33]
- [126] B. Van Leer and W. A. Mulder. Relaxation methods for hyperbolic conservation laws. In *Numerical Methods for the Euler equations of fluid dynamics*, pages 312–333. SIAM Philadelphia, PA, 1985. [Cit. on page 57]
- [127] D. Vanderstraeten, C. A., and D. Rose. An expert-system to control the CFL number of implicit upwind methods. Technical report, Universiteit Leuven, 2000. [Cit. on page 58]
- [128] M. Versteeg. Time Spectral Method for periodic problems using vertex centered spatial discretization. Master’s thesis, University of Twente, 2013. [Cit. on page 80]
- [129] M. R. Visbal and D. 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. [Cit. on pages 21 and 22]
- [130] P. Walsh and T. Pulliam. The effect of turbulence model solution on viscous flow problem. In *Proceedings of the AIAA Aerospace Sciences Conference*, number AIAA Paper 2001-1018, Reno, Nevada, January 2001. [Cit. on page 114]
- [131] Z. J. Wang et al. High-order CFD methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72:811–845, 2013. [Cit. on page 4]
- [132] P. Wesseling. *An introduction to multigrid methods*. Pure and applied mathematics. J. Wiley, 1992. [Cit. on page 53]
- [133] D. C. Wilcox. *Turbulence Modeling for CFD*. DCW Industries, Incorporated, third edition, 2006. [Cit. on page 23]

A

SUMMARY OF THE CONSERVATION EQUATIONS AND CLOSURE RELATIONS

Reported here are the main equations discussed in chapter 2, which the reader is referred to for more details.

We start with the extended system of URANS equations (mean flow plus the SA turbulence model). For readability reasons, the hat and overbar notations of the Favre- and Reynolds averaged variables are dropped. For the same reasons, in eq. (A.6) to (A.11), the viscous stress τ_{ij} and the heat flux $\dot{q}_i$ are meant to be the effective ones: $\tau_{ij} = \tau_{ij}^{eff}$ and $\dot{q}_i = \dot{q}_i^{eff}$.

$$J^{-1} \frac{\partial \mathbf{Q}}{\partial \tau} + \frac{\partial \tilde{\mathbf{E}}^c}{\partial \xi} + \frac{\partial \tilde{\mathbf{F}}^c}{\partial \eta} + \frac{\partial \tilde{\mathbf{G}}^c}{\partial \zeta} = \frac{\partial \tilde{\mathbf{E}}^v}{\partial \xi} + \frac{\partial \tilde{\mathbf{F}}^v}{\partial \eta} + \frac{\partial \tilde{\mathbf{G}}^v}{\partial \zeta} + \tilde{\mathbf{S}} \quad (\text{A.1})$$

$$\mathbf{Q} = [\rho, \rho u, \rho v, \rho w, \rho E, \rho \tilde{v}]^T \quad (\text{A.2})$$

$$\tilde{\mathbf{E}}^c = J^{-1} \begin{bmatrix} \rho U \\ \rho u U + p \xi_x \\ \rho v U + p \xi_y \\ \rho w U + p \xi_z \\ \rho H U - p \xi_\tau \\ \rho \tilde{v} U \end{bmatrix} \quad (\text{A.3})$$

$$\tilde{\mathbf{F}}^c = J^{-1} \begin{bmatrix} \rho V \\ \rho u V + p \eta_x \\ \rho v V + p \eta_y \\ \rho w V + p \eta_z \\ \rho H V - p \eta_\tau \\ \rho \tilde{v} V \end{bmatrix} \quad (\text{A.4})$$

$$\tilde{\mathbf{G}}^c = J^{-1} \begin{bmatrix} \rho W \\ \rho u W + p \zeta_x \\ \rho v W + p \zeta_y \\ \rho w W + p \zeta_z \\ \rho H W - p \zeta_\tau \\ \rho \tilde{v} W \end{bmatrix} \quad (\text{A.5})$$

$$\tilde{\mathbf{E}}^v = J^{-1} \begin{bmatrix} 0 \\ \xi_x \tau_{xx} + \xi_y \tau_{xy} + \xi_z \tau_{xz} \\ \xi_x \tau_{yx} + \xi_y \tau_{yy} + \xi_z \tau_{yz} \\ \xi_x \tau_{zx} + \xi_y \tau_{zy} + \xi_z \tau_{zz} \\ \xi_x \beta_x + \xi_y \beta_y + \xi_z \beta_z \\ \xi_x \phi_x + \xi_y \phi_y + \xi_z \phi_z \end{bmatrix} \quad (\text{A.6})$$

$$\tilde{\mathbf{F}}^v = \mathbf{J}^{-1} \begin{bmatrix} 0 \\ \eta_x \tau_{xx} + \eta_y \tau_{xy} + \eta_z \tau_{xz} \\ \eta_x \tau_{yx} + \eta_y \tau_{yy} + \eta_z \tau_{yz} \\ \eta_x \tau_{zx} + \eta_y \tau_{zy} + \eta_z \tau_{zz} \\ \eta_x \beta_x + \eta_y \beta_y + \eta_z \beta_z \\ \eta_x \phi_x + \eta_y \phi_y + \eta_z \phi_z \end{bmatrix} \quad (\text{A.7})$$

$$\tilde{\mathbf{G}}^v = \mathbf{J}^{-1} \begin{bmatrix} 0 \\ \zeta_x \tau_{xx} + \zeta_y \tau_{xy} + \zeta_z \tau_{xz} \\ \zeta_x \tau_{yx} + \zeta_y \tau_{yy} + \zeta_z \tau_{yz} \\ \zeta_x \tau_{zx} + \zeta_y \tau_{zy} + \zeta_z \tau_{zz} \\ \zeta_x \beta_x + \zeta_y \beta_y + \zeta_z \beta_z \\ \zeta_x \phi_x + \zeta_y \phi_y + \zeta_z \phi_z \end{bmatrix} \quad (\text{A.8})$$

with

$$\beta_x = u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - \dot{q}_x \quad (\text{A.9})$$

$$\beta_y = u\tau_{yx} + v\tau_{yy} + w\tau_{yz} - \dot{q}_y \quad (\text{A.10})$$

$$\beta_z = u\tau_{zx} + v\tau_{zy} + w\tau_{zz} - \dot{q}_z \quad (\text{A.11})$$

$$\phi_x = \frac{1}{\sigma} (\nu + (f_n + c_{b2})\tilde{\nu}) \frac{\partial \rho \tilde{\nu}}{\partial x} \quad (\text{A.12})$$

$$\phi_y = \frac{1}{\sigma} (\nu + (f_n + c_{b2})\tilde{\nu}) \frac{\partial \rho \tilde{\nu}}{\partial y} \quad (\text{A.13})$$

$$\phi_z = \frac{1}{\sigma} (\nu + (f_n + c_{b2})\tilde{\nu}) \frac{\partial \rho \tilde{\nu}}{\partial z} \quad (\text{A.14})$$

and finally

$$\tilde{\mathbf{S}} = \mathbf{J}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \rho(P - D) - L \end{bmatrix} + \mathbf{Q} \cdot \text{GCL}, \quad \text{GCL} = \left[\left(\frac{\xi_t}{J} \right)_{\xi} + \left(\frac{\eta_t}{J} \right)_{\eta} + \left(\frac{\zeta_t}{J} \right)_{\zeta} \right]. \quad (\text{A.15})$$

For non-moving meshes, $\text{GCL} = 0$. If the problem is solved in the non-inertial frame of reference under the assumptions outlined in section 2.1.4, then the additional source term (2.18), multiplied by $\mathbf{J}^{-1}$, is added to $\tilde{\mathbf{S}}$ as well. The source terms P , D and L of the turbulence model are given in eq. (2.63), (2.64) and (2.68), respectively. For 2D problems, the metric terms are calculated as in eq. (2.31). For 3D problems, instead, the free-stream preserving metrics eq. (2.43) are used.

The set of equations solved for LES are easily obtained from the previous ones. For LES, $\mathbf{Q}$ represents the filtered variables and the transport equation for $\rho \tilde{\nu}$ is ignored. The eddy viscosity μ_t is either set to zero for ILES, or computed according to eq. (2.73) if the WALE model is being used. Additionally, for DNS and laminar flow, $\mu_t = 0$ and $\mathbf{Q}$ is meant as the vector of instantaneous solution without any decomposition/filter. Finally for inviscid flow, all viscous quantities are neglected.

B | SBP-SAT OPERATORS

B.1 NORMS

The subscript refers to the global order of accuracy of the corresponding scheme (see for instance [104, 72]). For space reasons, H_4 and H_5 are written only for the ‘left’ boundary; at the ‘right’ boundary the sequence is reversed (see H_2 and H_3). They are all diagonal matrices.

$$H_2 = h \cdot \text{diag} \left[\frac{1}{2}, 1, \dots, 1, \frac{1}{2} \right], \quad (\text{B.1})$$

$$H_3 = h \cdot \text{diag} \left[\frac{17}{48}, \frac{59}{48}, \frac{43}{48}, \frac{49}{48}, 1, \dots, 1, \frac{49}{48}, \frac{43}{48}, \frac{59}{48}, \frac{17}{48} \right], \quad (\text{B.2})$$

$$H_4 = h \cdot \text{diag} \left[\frac{13649}{43200}, \frac{12013}{8640}, \frac{2711}{4320}, \frac{5359}{4320}, \frac{7877}{8640}, \frac{43801}{43200}, 1, \dots \right], \quad (\text{B.3})$$

$$H_5 = h \cdot \text{diag} \left[\frac{1498139}{5080320}, \frac{1107307}{725760}, \frac{20761}{80640}, \frac{1304999}{725760}, \right. \\ \left. \frac{299527}{725760}, \frac{103097}{80640}, \frac{670091}{725760}, \frac{5127739}{5080320}, 1, \dots \right]. \quad (\text{B.4})$$

B.2 FIRST DERIVATIVE OPERATORS

The first derivative operators are reproduced here from [104] and [111].

Second and third order SBP derivative operators. Note that for the third order operator the boundary closure on the max (right) boundary is obtained by reversing the sequence and switching the sign of the constants (see $D^{2\text{nd}}$).

$$D^{2\text{nd}} \mathbf{v} = \frac{1}{h} \begin{bmatrix} -1 & 1 & & & \\ & \frac{1}{2} & 0 & & \\ & & \ddots & \ddots & \ddots \\ & & & -\frac{1}{2} & 0 & \frac{1}{2} \\ & & & & -1 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ v_n \end{bmatrix},$$

$$D^{3rd} \mathbf{v} = \frac{1}{h} \begin{bmatrix} -\frac{24}{17} & \frac{59}{34} & -\frac{4}{17} & -\frac{3}{34} & 0 & 0 & 0 & \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & \\ \frac{4}{43} & -\frac{59}{86} & 0 & \frac{59}{86} & -\frac{4}{43} & 0 & 0 & \\ \frac{3}{98} & 0 & -\frac{59}{98} & 0 & \frac{32}{49} & -\frac{4}{49} & 0 & \\ 0 & 0 & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ \vdots \end{bmatrix}$$

Fourth order operator D^{4th} , interior stencil:

$$v_i = \frac{1}{h} \left(-\frac{1}{60}v_{i-3} + \frac{9}{60}v_{i-2} - \frac{45}{60}v_{i-1} + \frac{45}{60}v_{i+1} - \frac{9}{60}v_{i+2} + \frac{1}{60}v_{i+3} \right) \quad (B.5)$$

Fourth order operator D^{4th} , boundary discretization. Constants are given for a one-sided difference on the min boundaries. On the max boundaries the sequence needs to be reversed and the constants negated.

$$x_6 = 342523.0/518400.0$$

$$d_{1,1} = -(21600.0/13649.0)$$

$$d_{1,2} = (43200.0/13649.0) \cdot x_6 - (7624.0/40947.0)$$

$$d_{1,3} = -(172800.0/13649.0) \cdot x_6 + (715489.0/81894.0)$$

$$d_{1,4} = (259200.0/13649.0) \cdot x_6 - (187917.0/13649.0)$$

$$d_{1,5} = -(172800.0/13649.0) \cdot x_6 + (735635.0/81894.0)$$

$$d_{1,6} = (43200.0/13649.0) \cdot x_6 - (89387.0/40947.0)$$

$$d_{2,1} = -(8640.0/12013.0) \cdot x_6 + (7624.0/180195.0)$$

$$d_{2,3} = (86400.0/12013.0) \cdot x_6 - (57139.0/12013.0)$$

$$d_{2,4} = -(172800.0/12013.0) \cdot x_6 + (745733.0/72078.0)$$

$$d_{2,5} = (129600.0/12013.0) \cdot x_6 - (91715.0/12013.0)$$

$$d_{2,6} = -(34560.0/12013.0) \cdot x_6 + (240569.0/120130.0)$$

$$d_{3,1} = (17280.0/2711.0) \cdot x_6 - (715489.0/162660.0)$$

$$d_{3,2} = -(43200.0/2711.0) \cdot x_6 + (57139.0/5422.0)$$

$$d_{3,4} = (86400.0/2711.0) \cdot x_6 - (176839.0/8133.0)$$

$$d_{3,5} = -(86400.0/2711.0) \cdot x_6 + (242111.0/10844.0)$$

$$d_{3,6} = (25920.0/2711.0) \cdot x_6 - (182261.0/27110.0)$$

$$d_{4,1} = -(25920.0/5359.0) \cdot x_6 + (187917.0/53590.0)$$

$$d_{4,2} = (86400.0/5359.0) \cdot x_6 - (745733.0/64308.0)$$

$$d_{4,3} = -(86400.0/5359.0) \cdot x_6 + (176839.0/16077.0)$$

$$d_{4,5} = (43200.0/5359.0) \cdot x_6 - (165041.0/32154.0)$$

$$d_{4,6} = -(17280.0/5359.0) \cdot x_6 + (710473.0/321540.0)$$

$$d_{4,7} = (72.0/5359.0)$$

$$\begin{aligned}
d_{5,1} &= (34560.0/7877.0) \cdot x_6 - (147127.0/47262.0) \\
d_{5,2} &= - (129600.0/7877.0) \cdot x_6 + (91715.0/7877.0) \\
d_{5,3} &= (172800.0/7877.0) \cdot x_6 - (242111.0/15754.0) \\
d_{5,4} &= - (86400.0/7877.0) \cdot x_6 + (165041.0/23631.0) \\
d_{5,6} &= (8640.0/7877.0) \cdot x_6 \\
d_{5,7} &= - (1296.0/7877.0) \\
d_{5,8} &= (144.0/7877.0)
\end{aligned}$$

$$\begin{aligned}
d_{6,1} &= - (43200.0/43801.0) \cdot x_6 + (89387.0/131403.0) \\
d_{6,2} &= (172800.0/43801.0) \cdot x_6 - (240569.0/87602.0) \\
d_{6,3} &= - (259200.0/43801.0) \cdot x_6 + (182261.0/43801.0) \\
d_{6,4} &= (172800.0/43801.0) \cdot x_6 - (710473.0/262806.0) \\
d_{6,5} &= - (43200.0/43801.0) \cdot x_6 \\
d_{6,7} &= (32400.0/43801.0) \\
d_{6,8} &= - (6480.0/43801.0) \\
d_{6,9} &= (720.0/43801.0)
\end{aligned}$$

Fifth order operator $D^{5^{\text{th}}}$, interior stencil:

$$\begin{aligned}
v_i = \frac{1}{h} \left(\frac{1}{280} v_{i-4} - \frac{4}{105} v_{i-3} + \frac{1}{5} v_{i-2} - \frac{4}{5} v_{i-1} \right. \\
\left. + \frac{4}{5} v_{i+1} - \frac{1}{5} v_{i+2} + \frac{4}{105} v_{i+3} - \frac{1}{280} v_{i+4} \right) \quad (\text{B.6})
\end{aligned}$$

Fifth order operator $D^{5^{\text{th}}}$, boundary discretization. Constants are given for a one-sided difference on the min boundaries. On the max boundaries the sequence needs to be reversed and the constants negated. Note that the parameters r_{67}, r_{68}, r_{78} that we use are the ones proposed in [111], which guarantee a smaller spectral radius than the original parameters in [104]¹.

$$r_{67} = 0.649, \quad r_{68} = -0.104, \quad r_{78} = 0.755.$$

¹ The parameters r_{67}, r_{68}, r_{78} for the original (minimal-bandwidth) operator [104] are the following:

$$\begin{aligned}
r_{67} &= 1714837/4354560 \approx 0.394, & r_{68} &= -1022551/30481920 \approx -0.034, \\
r_{78} &= 6445687/8709120 \approx 0.740.
\end{aligned}$$

$$\begin{aligned}
d_{1,1} &= -(2540160.0/1498139.0) \\
d_{1,2} &= -(142642467.0/5992556.0) + (50803200.0/1498139.0) \cdot r_{78} \\
&\quad + (5080320.0/1498139.0) \cdot r_{67} + (25401600.0/1498139.0) \cdot r_{68} \\
d_{1,3} &= (705710031.0/5992556.0) - (228614400.0/1498139.0) \cdot r_{78} \\
&\quad - (25401600.0/1498139.0) \cdot r_{67} - (121927680.0/1498139.0) \cdot r_{68} \\
d_{1,4} &= -(3577778591.0/17977668.0) + (381024000.0/1498139.0) \cdot r_{78} \\
&\quad + (50803200.0/1498139.0) \cdot r_{67} + (228614400.0/1498139.0) \cdot r_{68} \\
d_{1,5} &= (203718909.0/1498139.0) - (254016000.0/1498139.0) \cdot r_{78} \\
&\quad - (50803200.0/1498139.0) \cdot r_{67} - (203212800.0/1498139.0) \cdot r_{68} \\
d_{1,6} &= -(32111205.0/5992556.0) + (25401600.0/1498139.0) \cdot r_{67} \\
&\quad + (76204800.0/1498139.0) \cdot r_{68} \\
d_{1,7} &= -(652789417.0/17977668.0) + (76204800.0/1498139.0) \cdot r_{78} \\
&\quad - (5080320.0/1498139.0) \cdot r_{67} \\
d_{1,8} &= (74517981.0/5992556.0) - (25401600.0/1498139.0) \cdot r_{78} \\
&\quad - (5080320.0/1498139.0) \cdot r_{68} \\
d_{2,1} &= (142642467.0/31004596.0) - (7257600.0/1107307.0) \cdot r_{78} \\
&\quad - (725760.0/1107307.0) \cdot r_{67} - (3628800.0/1107307.0) \cdot r_{68} \\
d_{2,3} &= -(141502371.0/2214614.0) + (91445760.0/1107307.0) \cdot r_{78} \\
&\quad + (10886400.0/1107307.0) \cdot r_{67} + (50803200.0/1107307.0) \cdot r_{68} \\
d_{2,4} &= (159673719.0/1107307.0) - (203212800.0/1107307.0) \cdot r_{78} \\
&\quad - (29030400.0/1107307.0) \cdot r_{67} - (127008000.0/1107307.0) \cdot r_{68} \\
d_{2,5} &= -(1477714693.0/13287684.0) + (152409600.0/1107307.0) \cdot r_{78} \\
&\quad + (32659200.0/1107307.0) \cdot r_{67} + (127008000.0/1107307.0) \cdot r_{68} \\
d_{2,6} &= (11652351.0/2214614.0) - (17418240.0/1107307.0) \cdot r_{67} \\
&\quad - (50803200.0/1107307.0) \cdot r_{68} \\
d_{2,7} &= (36069450.0/1107307.0) - (50803200.0/1107307.0) \cdot r_{78} \\
&\quad + (3628800.0/1107307.0) \cdot r_{67} \\
d_{2,8} &= -(536324953.0/46506894.0) + (17418240.0/1107307.0) \cdot r_{78} \\
&\quad + (3628800.0/1107307.0) \cdot r_{68} \\
d_{3,1} &= -(18095129.0/134148.0) + (3628800.0/20761.0) \cdot r_{78} \\
&\quad + (403200.0/20761.0) \cdot r_{67} + (1935360.0/20761.0) \cdot r_{68} \\
d_{3,2} &= (47167457.0/124566.0) - (10160640.0/20761.0) \cdot r_{78} \\
&\quad - (1209600.0/20761.0) \cdot r_{67} - (5644800.0/20761.0) \cdot r_{68} \\
d_{3,4} &= -(120219461.0/124566.0) + (25401600.0/20761.0) \cdot r_{78} \\
&\quad + (4032000.0/20761.0) \cdot r_{67} + (16934400.0/20761.0) \cdot r_{68} \\
d_{3,5} &= (249289259.0/249132.0) - (25401600.0/20761.0) \cdot r_{78} \\
&\quad - (6048000.0/20761.0) \cdot r_{67} - (22579200.0/20761.0) \cdot r_{68} \\
d_{3,6} &= -(2611503.0/41522.0) + (3628800.0/20761.0) \cdot r_{67} \\
&\quad + (10160640.0/20761.0) \cdot r_{68} \\
d_{3,7} &= -(7149666.0/20761.0) + (10160640.0/20761.0) \cdot r_{78} \\
&\quad - (806400.0/20761.0) \cdot r_{67} \\
d_{3,8} &= (37199165.0/290654.0) - (3628800.0/20761.0) \cdot r_{78} \\
&\quad - (806400.0/20761.0) \cdot r_{68}
\end{aligned}$$

$$\begin{aligned}
d_{4,1} &= (3577778591.0/109619916.0) - (54432000.0/1304999.0) \cdot r_{78} \\
&\quad - (7257600.0/1304999.0) \cdot r_{67} - (32659200.0/1304999.0) \cdot r_{68} \\
d_{4,2} &= - (159673719.0/1304999.0) + (203212800.0/1304999.0) \cdot r_{78} \\
&\quad + (29030400.0/1304999.0) \cdot r_{67} + (127008000.0/1304999.0) \cdot r_{68} \\
d_{4,3} &= (360658383.0/2609998.0) - (228614400.0/1304999.0) \cdot r_{78} \\
&\quad - (36288000.0/1304999.0) \cdot r_{67} - (152409600.0/1304999.0) \cdot r_{68} \\
d_{4,5} &= - (424854441.0/5219996.0) + (127008000.0/1304999.0) \cdot r_{78} \\
&\quad + (36288000.0/1304999.0) \cdot r_{67} + (127008000.0/1304999.0) \cdot r_{68} \\
d_{4,6} &= (22885113.0/2609998.0) - (29030400.0/1304999.0) \cdot r_{67} \\
&\quad - (76204800.0/1304999.0) \cdot r_{68} \\
d_{4,7} &= (158096578.0/3914997.0) - (76204800.0/1304999.0) \cdot r_{78} \\
&\quad + (7257600.0/1304999.0) \cdot r_{67} \\
d_{4,8} &= - (296462325.0/18269986.0) + (29030400.0/1304999.0) \cdot r_{78} \\
&\quad + (7257600.0/1304999.0) \cdot r_{68} \\
\\
d_{5,1} &= - (203718909.0/2096689.0) + (36288000.0/299527.0) \cdot r_{78} \\
&\quad + (7257600.0/299527.0) \cdot r_{67} + (29030400.0/299527.0) \cdot r_{68} \\
d_{5,2} &= (1477714693.0/3594324.0) - (152409600.0/299527.0) \cdot r_{78} \\
&\quad - (32659200.0/299527.0) \cdot r_{67} - (127008000.0/299527.0) \cdot r_{68} \\
d_{5,3} &= - (747867777.0/1198108.0) + (228614400.0/299527.0) \cdot r_{78} \\
&\quad + (54432000.0/299527.0) \cdot r_{67} + (203212800.0/299527.0) \cdot r_{68} \\
d_{5,4} &= (424854441.0/1198108.0) - (127008000.0/299527.0) \cdot r_{78} \\
&\quad - (36288000.0/299527.0) \cdot r_{67} - (127008000.0/299527.0) \cdot r_{68} \\
d_{5,6} &= - (17380335.0/1198108.0) + (10886400.0/299527.0) \cdot r_{67} \\
&\quad + (25401600.0/299527.0) \cdot r_{68} \\
d_{5,7} &= - (67080435.0/1198108.0) + (25401600.0/299527.0) \cdot r_{78} \\
&\quad - (3628800.0/299527.0) \cdot r_{67} \\
d_{5,8} &= (657798011.0/25160268.0) - (10886400.0/299527.0) \cdot r_{78} \\
&\quad - (3628800.0/299527.0) \cdot r_{68} \\
d_{5,9} &= - (2592.0/299527.0) \\
\\
d_{6,1} &= (1529105.0/1237164.0) - (403200.0/103097.0) \cdot r_{67} \\
&\quad - (1209600.0/103097.0) \cdot r_{68} \\
d_{6,2} &= - (3884117.0/618582.0) + (1935360.0/103097.0) \cdot r_{67} \\
&\quad + (5644800.0/103097.0) \cdot r_{68} \\
d_{6,3} &= (2611503.0/206194.0) - (3628800.0/103097.0) \cdot r_{67} \\
&\quad - (10160640.0/103097.0) \cdot r_{68} \\
d_{6,4} &= - (7628371.0/618582.0) + (3225600.0/103097.0) \cdot r_{67} \\
&\quad + (8467200.0/103097.0) \cdot r_{68} \\
d_{6,5} &= (5793445.0/1237164.0) - (1209600.0/103097.0) \cdot r_{67} \\
&\quad - (2822400.0/103097.0) \cdot r_{68} \\
d_{6,7} &= (80640.0/103097.0) \cdot r_{67} \\
d_{6,8} &= (80640.0/103097.0) \cdot r_{68} \\
d_{6,9} &= (3072.0/103097.0) \\
d_{6,10} &= - (288.0/103097.0)
\end{aligned}$$

$$\begin{aligned}
d_{7,1} &= (93255631.0/8041092.0) - (10886400.0/670091.0) \cdot r_{78} \\
&\quad + (725760.0/670091.0) \cdot r_{67} \\
d_{7,2} &= - (36069450.0/670091.0) + (50803200.0/670091.0) \cdot r_{78} \\
&\quad - (3628800.0/670091.0) \cdot r_{67} \\
d_{7,3} &= (64346994.0/670091.0) - (91445760.0/670091.0) \cdot r_{78} \\
&\quad + (7257600.0/670091.0) \cdot r_{67} \\
d_{7,4} &= - (158096578.0/2010273.0) + (76204800.0/670091.0) \cdot r_{78} \\
&\quad - (7257600.0/670091.0) \cdot r_{67} \\
d_{7,5} &= (67080435.0/2680364.0) - (25401600.0/670091.0) \cdot r_{78} \\
&\quad + (3628800.0/670091.0) \cdot r_{67} \\
d_{7,6} &= - (725760.0/670091.0) \cdot r_{67} \\
d_{7,8} &= (725760.0/670091.0) \cdot r_{78} \\
d_{7,9} &= - (145152.0/670091.0) \\
d_{7,10} &= (27648.0/670091.0) \\
d_{7,11} &= - (2592.0/670091.0)
\end{aligned}$$

$$\begin{aligned}
d_{8,1} &= - (3921999.0/1079524.0) + (25401600.0/5127739.0) \cdot r_{78} \\
&\quad + (5080320.0/5127739.0) \cdot r_{68} \\
d_{8,2} &= (536324953.0/30766434.0) - (121927680.0/5127739.0) \cdot r_{78} \\
&\quad - (25401600.0/5127739.0) \cdot r_{68} \\
d_{8,3} &= - (334792485.0/10255478.0) + (228614400.0/5127739.0) \cdot r_{78} \\
&\quad + (50803200.0/5127739.0) \cdot r_{68} \\
d_{8,4} &= (296462325.0/10255478.0) - (203212800.0/5127739.0) \cdot r_{78} \\
&\quad - (50803200.0/5127739.0) \cdot r_{68} \\
d_{8,5} &= - (657798011.0/61532868.0) + (76204800.0/5127739.0) \cdot r_{78} \\
&\quad + (25401600.0/5127739.0) \cdot r_{68} \\
d_{8,6} &= - (5080320.0/5127739.0) \cdot r_{68} \\
d_{8,7} &= - (5080320.0/5127739.0) \cdot r_{78} \\
d_{8,9} &= (4064256.0/5127739.0) \\
d_{8,10} &= - (1016064.0/5127739.0) \\
d_{8,11} &= (193536.0/5127739.0) \\
d_{8,12} &= - (18144.0/5127739.0)
\end{aligned}$$

B.3 ARTIFICIAL DISSIPATION

Referring to eq. (3.43), the $\tilde{D}_p$ operators are essentially the ones in [73] and are reproduced here for the reader's convenience. However, $\tilde{D}_1$ and $\tilde{D}_3$ are modified

as follows for implementation reasons (see also the definition of the B_p matrices given in eq. (3.46)).

$$\tilde{D}_1 = \begin{bmatrix} -1 & 1 & & & \\ & \ddots & \ddots & & \\ & & -1 & 1 & \\ & & & -1 & 1 \\ & & & & -1 & 1 \end{bmatrix} \quad \tilde{D}_2 = \begin{bmatrix} 1 & -2 & 1 & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix} \quad (B.7)$$

$$\tilde{D}_3 = \begin{bmatrix} -1 & 3 & -3 & 1 & & \\ -1 & 3 & -3 & 1 & & \\ -1 & 3 & -3 & 1 & & \\ & -1 & 3 & -3 & 1 & \\ & & \ddots & \ddots & \ddots & \ddots \\ & & & -1 & 3 & -3 & 1 \\ & & & & -1 & 3 & -3 \end{bmatrix} \quad (B.8)$$

$$\tilde{D}_4 = \begin{bmatrix} 1 & -4 & 6 & -4 & 1 & & \\ 1 & -4 & 6 & -4 & 1 & & \\ 1 & -4 & 6 & -4 & 1 & & \\ & 1 & -4 & 6 & -4 & 1 & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots \\ & & & 1 & -4 & 6 & -4 & 1 \\ & & & & 1 & -4 & 6 & -4 \\ & & & & & 1 & -4 & 6 \end{bmatrix} \quad (B.9)$$

B.4 SAT TERMS

B.4.1 Euler wall viscous penalty state

The shear stress normal component should be negated so that a symmetry boundary condition is obtained. The formulae below are the result of the following transformations. First the stress tensor is transformed to a coordinate system where the first axis is aligned with the normal. The other two axis are chosen such that a right handed system is obtained (which defines the latter two up to an arbitrary angle). Next the shear stresses involving the normal component are negated and that re-

sult is transformed back to the Cartesian tensor. The final result only depends on the unit normal vector $\mathbf{n} = [n_x, n_y, n_z]^T$.

$$\tau_{xx}^{\text{pen}} = \tau_{xx} a_1 a_1 + \tau_{yy} a_4 + \tau_{zz} a_5 + \tau_{xy} 4a_1 n_{xy} + \tau_{xz} 4a_1 n_{xz} + \tau_{yz} 8n_x^2 n_{yz} \quad (\text{B.10})$$

$$\tau_{yy}^{\text{pen}} = \tau_{xx} a_4 + \tau_{yy} a_2 a_2 + \tau_{zz} a_6 + \tau_{xy} 4a_2 n_{xy} + \tau_{xz} 8n_y^2 n_{xz} + \tau_{yz} 4a_2 n_{yz} \quad (\text{B.11})$$

$$\tau_{zz}^{\text{pen}} = \tau_{xx} a_5 + \tau_{yy} a_6 + \tau_{zz} a_3 a_3 + \tau_{xy} 8n_z^2 n_{xy} + \tau_{xz} 4a_3 n_{xz} + \tau_{yz} 4a_3 n_{yz} \quad (\text{B.12})$$

$$\begin{aligned} \tau_{xy}^{\text{pen}} = & \tau_{xx} 2a_1 n_{xy} + \tau_{yy} 2a_2 n_{xy} + \tau_{zz} 4n_{xy} n_z^2 + \tau_{xy} (a_4 + a_1 a_2) \\ & + \tau_{xz} (4n_x^2 n_{yz} + 2a_1 n_{yz}) + \tau_{yz} (4n_y^2 n_{xz} + 2a_2 n_{xz}) \end{aligned} \quad (\text{B.13})$$

$$\begin{aligned} \tau_{xz}^{\text{pen}} = & \tau_{xx} 2a_1 n_{xz} + \tau_{yy} 4n_{xz} n_y^2 + \tau_{zz} 2a_3 n_{xz} + \tau_{xy} (4n_x^2 n_{yz} + 2a_1 n_{yz}) \\ & + \tau_{xz} (a_5 + a_1 a_3) + \tau_{yz} (4n_z^2 n_{xy} + 2a_3 n_{xy}) \end{aligned} \quad (\text{B.14})$$

$$\begin{aligned} \tau_{yz}^{\text{pen}} = & \tau_{xx} 4n_{yz} n_x^2 + \tau_{yy} 2a_2 n_{yz} + \tau_{zz} 2a_3 n_{yz} + \tau_{xy} (4n_y^2 n_{xz} + 2a_2 n_{xz}) \\ & + \tau_{xz} (4n_z^2 n_{xy} + 2a_3 n_{xy}) + \tau_{yz} (a_6 + a_2 a_3) \end{aligned} \quad (\text{B.15})$$

where

$$n_{xy} = n_x n_y, \quad n_{xz} = n_x n_z, \quad n_{yz} = n_y n_z, \quad (\text{B.16})$$

$$a_1 = 2n_x^2 - 1, \quad a_2 = 2n_y^2 - 1, \quad a_3 = 2n_z^2 - 1, \quad (\text{B.17})$$

$$a_4 = 4n_x^2 n_y^2, \quad a_5 = 4n_x^2 n_z^2, \quad a_6 = 4n_y^2 n_z^2. \quad (\text{B.18})$$

B.4.2 Block interface viscous stability term ($\tilde{B}_{11}$)

The viscous stability term for an internal interface boundary, see eq. (3.36), is the part of the penalty term that multiplies the difference in the state vector and is a result of the analysis carried out in [80]. In this work the boundary condition is applied using the curvilinear $\tilde{B}_{11}, \tilde{B}_{22}, \tilde{B}_{33}$ viscous Jacobian when the boundary has the normal in the ξ -, η -, ζ -direction, respectively. Each curvilinear $\tilde{B}_{11}, \tilde{B}_{22}, \tilde{B}_{33}$ is a combination of all six Cartesian jacobians $B_{11}, B_{12}, B_{13}, B_{22}, B_{23}, B_{33}$. The difference lies in the metric terms that multiply the Cartesian matrices. Keep in mind that the Cartesian jacobians here are written in terms of conserved variables (and not symmetrized, as in the aforementioned paper). The Cartesian matrices B_{11}, B_{22}, B_{33} can also be found in [84] for the NS equations (the mixed-indices matrix B_{12}, B_{23}, B_{33} are not considered there). Here they are given for the coupled system of URANS equations, assuming a face normal to the ξ direction. Furthermore, note that the following matrices are calculated at the Roe-averaged state between the current state $\mathbf{Q}$ and the target state $\mathbf{g}^i$.

$$\tilde{B}_{11} = \left(\xi_x^2 B_{11} + \xi_y^2 B_{22} + \xi_z^2 B_{33} + 2\xi_x \xi_y B_{12} + 2\xi_x \xi_z B_{13} + 2\xi_y \xi_z B_{23} \right) / J \quad (\text{B.19})$$

$$B_{ii} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -t_1^{ii}u & t_1^{ii} & 0 & 0 & 0 & 0 \\ -t_2^{ii}v & 0 & t_2^{ii} & 0 & 0 & 0 \\ -t_3^{ii}w & 0 & 0 & t_3^{ii} & 0 & 0 \\ b_{51}^{ii} & (t_1^{ii} - t_4)u & (t_2^{ii} - t_4)v & (t_3^{ii} - t_4)w & t_4 & 0 \\ -t_6\tilde{v} & 0 & 0 & 0 & 0 & t_6 \end{bmatrix} \quad (B.20)$$

where

$$\begin{aligned} t_1^{11} &= \frac{4}{3}\nu & t_1^{22} &= \nu & t_1^{33} &= \nu \\ t_2^{11} &= \nu & t_2^{22} &= \frac{4}{3}\nu & t_2^{33} &= \nu \\ t_3^{11} &= \nu & t_3^{22} &= \nu & t_3^{33} &= \frac{4}{3}\nu \end{aligned}$$

and

$$\begin{aligned} \nu &= \mu/\rho \\ t_4 &= \frac{\nu\gamma}{\text{Pr}} + \frac{\nu^t\gamma}{\text{Pr}^t} \\ t_6 &= \frac{1}{\sigma}(\nu + \tilde{v}(f_n + c_{b2})) \\ b_{51}^{ii} &= t_4 \left(\frac{c^2}{\gamma} - H \right) - (t_1^{ii} - t_4)u^2 - (t_2^{ii} - t_4)v^2 - (t_3^{ii} - t_4)w^2 \end{aligned}$$

The mixed-indices Cartesian matrices are given here below. Note that the turbulence equation does not receive any contribution from these matrices, thus they are represented as 5×5 matrices. The value of t_5 follows from the Stokes' hypothesis.

$$B_{12} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -t_5v & 0 & t_5 & 0 & 0 \\ -t_5u & t_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -2t_5uv & t_5v & t_5u & 0 & 0 \end{bmatrix} \quad (B.21)$$

$$B_{13} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -t_5w & 0 & 0 & t_5 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -t_5u & t_5 & 0 & 0 & 0 \\ -2t_5uw & t_5w & 0 & t_5u & 0 \end{bmatrix} \quad (B.22)$$

$$B_{23} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -t_5w & 0 & 0 & t_5 & 0 \\ -t_5v & 0 & t_5 & 0 & 0 \\ -2t_5vw & 0 & t_5w & t_5v & 0 \end{bmatrix} \quad (B.23)$$

where

$$t_5 = \frac{1}{6} \frac{\mu}{\rho} \quad (B.24)$$

B.4.3 Sixth and eighth order interpolation operators

The interpolation weights for the 6th order operator are (see Fig. 3.4):

$$\text{Sol}(P) = \sum_{i=-2}^3 w_i \text{Sol}(N_i), \quad \alpha = (h - h_0)/h$$

$w_{-2} = -1/120$	$(\alpha - 3)(\alpha - 2)$	$(\alpha - 1)(\alpha)$	$(1 + \alpha)$
$w_{-1} = +1/24$	$(\alpha - 3)(\alpha - 2)$	$(\alpha - 1)(\alpha)$	$(2 + \alpha)$
$w_0 = -1/12$	$(\alpha - 3)(\alpha - 2)$	$(\alpha - 1)(1 + \alpha)$	$(2 + \alpha)$
$w_{+1} = +1/12$	$(\alpha - 3)(\alpha - 2)$	$(\alpha)(1 + \alpha)$	$(2 + \alpha)$
$w_{+2} = -1/24$	$(\alpha - 3)(\alpha - 1)$	$(\alpha)(1 + \alpha)$	$(2 + \alpha)$
$w_{+3} = +1/120$	$(\alpha - 2)(\alpha - 1)$	$(\alpha)(1 + \alpha)$	$(2 + \alpha)$

The interpolation weights for the 8th order operator are (see Fig. 3.4):

$$\text{Sol}(P) = \sum_{i=-3}^4 w_i \text{Sol}(N_i), \quad \alpha = (h - h_0)/h$$

$w_{-3} = -1/5040$	$(\alpha - 4)(\alpha - 3)$	$(\alpha - 2)(\alpha - 1)$	$(\alpha)(\alpha + 1)$	$(\alpha + 2)$
$w_{-2} = +1/720$	$(\alpha - 4)(\alpha - 3)$	$(\alpha - 2)(\alpha - 1)$	$(\alpha)(\alpha + 1)$	$(\alpha + 3)$
$w_{-1} = -1/240$	$(\alpha - 4)(\alpha - 3)$	$(\alpha - 2)(\alpha - 1)$	$(\alpha)(\alpha + 2)$	$(\alpha + 3)$
$w_0 = +1/144$	$(\alpha - 4)(\alpha - 3)$	$(\alpha - 2)(\alpha - 1)$	$(\alpha + 1)(\alpha + 2)$	$(\alpha + 3)$
$w_{+1} = -1/144$	$(\alpha - 4)(\alpha - 3)$	$(\alpha - 2)(\alpha)$	$(\alpha + 1)(\alpha + 2)$	$(\alpha + 3)$
$w_{+2} = +1/240$	$(\alpha - 4)(\alpha - 3)$	$(\alpha - 1)(\alpha)$	$(\alpha + 1)(\alpha + 2)$	$(\alpha + 3)$
$w_{+3} = -1/720$	$(\alpha - 4)(\alpha - 2)$	$(\alpha - 1)(\alpha)$	$(\alpha + 1)(\alpha + 2)$	$(\alpha + 3)$
$w_{+4} = +1/5040$	$(\alpha - 3)(\alpha - 2)$	$(\alpha - 1)(\alpha)$	$(\alpha + 1)(\alpha + 2)$	$(\alpha + 3)$

C

INVISCID FLUX JACOBIAN

As seen in section 3.2.3, the application of the SAT terms involves the computation of the matrix vector product $A^\pm \cdot \Delta \mathbf{Q}$. The matrix $A^\pm$ is the inviscid flux Jacobian in curvilinear coordinates which must be constructed retaining either the positive or the negative eigenvalues. The vector $\Delta \mathbf{Q}$ is the difference between the current state and the target penalty state. It is therefore convenient to express A (in either ξ , η or ζ direction) in terms of its eigenvalues. The derivation of the Jacobian written in such terms is carried out for the extended system of the RANS equations, but the Jacobian for the NS and for the Euler equations can be obtained easily from the following derivation simply neglecting all terms containing $\tilde{v}$ (this simpler Jacobian can also be found in [92]). The actual symbolic computations have been performed with the software Maxima [75], and we report the code, with comments, in section C.5.

C.1 GENERIC INVISCID FLUX

From eq. (2.37), (2.38) and (2.67a), the inviscid (convective) flux $\tilde{\mathbf{Z}}^c$ in the generic k curvilinear direction reads:

$$\tilde{\mathbf{Z}}^c = \mathbf{J}^{-1} \begin{bmatrix} \rho\theta \\ \rho u\theta + pk_x \\ \rho v\theta + pk_y \\ \rho w\theta + pk_z \\ \rho H\theta - pk_\tau \\ \rho\tilde{v}\theta \end{bmatrix}, \quad k \in (\xi, \eta, \zeta)$$

where θ is the contravariant velocity

$$\theta = k_\tau + k_x u + k_y v + k_z w.$$

For example, if we are dealing with $\xi = \text{constant}$ boundaries, then $k = \xi$, $\theta = U$ and the inviscid curvilinear flux $\tilde{\mathbf{Z}}^c = \tilde{\mathbf{E}}^c$, as in eq. (2.37). Rewrite the component associated with the conservation of energy as follows:

$$\begin{aligned} \rho H\theta - pk_\tau &= (\rho E + p)\theta - pk_\tau = \rho E\theta + p(\theta - k_\tau) = \\ &= \rho E\theta + p(k_x u + k_y v + k_z w) \\ &= \rho E\theta + p\theta^* \\ &= \left(\frac{p}{(\gamma - 1)} + \rho q \right) \theta + p\theta^* \end{aligned} \tag{C.1}$$

where

$$\theta^* = k_x u + k_y v + k_z w, \quad q = \frac{u^2 + v^2 + w^2}{2}. \quad (\text{C.2})$$

The generic inviscid flux can now be written as:

$$\tilde{\mathbf{Z}}^c = \mathbf{J}^{-1} \begin{bmatrix} \rho\theta \\ \rho u\theta + pk_x \\ \rho v\theta + pk_y \\ \rho w\theta + pk_z \\ \rho E\theta + p\theta^* \\ \rho\tilde{v}\theta \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \rho\theta \\ \rho u\theta + pk_x \\ \rho v\theta + pk_y \\ \rho w\theta + pk_z \\ \left(\frac{p}{(\gamma-1)} + \rho q\right)\theta + p\theta^* \\ \rho\tilde{v}\theta \end{bmatrix} \quad (\text{C.3})$$

We would like to derive a formulation of the Jacobian A of $\tilde{\mathbf{Z}}^c$ with respect to the vector of curvilinear conserved variables $\tilde{\mathbf{Q}} = \mathbf{J}^{-1}\mathbf{Q} = \mathbf{J}^{-1}[\rho, \rho u, \rho v, \rho w, \rho E, \rho\tilde{v}]^T$, i. e.

$$A = \frac{\partial \tilde{\mathbf{Z}}^c}{\partial \tilde{\mathbf{Q}}},$$

in terms of its eigenvalues. We will follow the same procedure employed in [21], although in [21] this is done in the context of finite volume schemes and Cartesian coordinates; here we work in the context of finite difference schemes and in curvilinear coordinates.

C.2 EIGENVALUES AND EIGENVECTORS

We seek the eigenvalues and eigenvectors of A . By definition, an eigenvalue λ_i of matrix A is a scalar which satisfies the condition

$$A\mathbf{r}_i = \lambda_i\mathbf{r}_i \iff (A - \lambda_i I)\mathbf{r}_i = 0, \quad i = 1, \dots, 6 \quad (\text{C.4})$$

where the vector $\mathbf{r}_i$ is defined as the right eigenvector of A corresponding to λ_i . The eigenvalues of A can be found by finding the roots of the characteristic polynomial, i.e. by solving the characteristic equation for λ_i :

$$|A - \lambda_i I| = 0 \quad (\text{C.5})$$

where $|\cdot|$ indicates the determinant. Furthermore, if there is a vector $\mathbf{l}_i$ such that

$$\mathbf{l}_i(A - \lambda_i I) = 0, \quad i = 1, \dots, 6 \quad (\text{C.6})$$

then, $\mathbf{l}_i$ is the left eigenvector of A corresponding to λ_i . If we construct the matrix L^T whose rows are the transposed left eigenvectors $\mathbf{l}_i^T$, and the matrix R whose columns are the right eigenvectors $\mathbf{r}_i$, then

$$L^T A R = \Lambda, \quad (\text{C.7})$$

where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_6)$. The matrix L^T can be obtained from R as

$$L^T = R^{-1}. \quad (\text{C.8})$$

This corresponds to scaling the eigenvectors such that $\mathbf{I}_j^T \mathbf{r}_i = \delta_{ij}$.

The matrices L^T and R form together a similarity transformation that diagonalizes the matrix A .

C.3 TRANSFORMATION TO PRIMITIVE VARIABLES

The problem of finding the eigenvalues and eigenvectors of A is simplified by rewriting it in terms of the primitive variables, $\tilde{\mathbf{P}} = J^{-1} \mathbf{P} = J^{-1} [\rho, u, v, w, p, \tilde{v}]^T$, using the transformation M

$$M = \frac{\partial \tilde{\mathbf{Q}}}{\partial \tilde{\mathbf{P}}} = \frac{\partial \mathbf{Q}}{\partial \mathbf{P}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ u & \rho & 0 & 0 & 0 & 0 \\ v & 0 & \rho & 0 & 0 & 0 \\ w & 0 & 0 & \rho & 0 & 0 \\ q & \rho u & \rho v & \rho w & \frac{1}{\gamma-1} & 0 \\ \tilde{v} & 0 & 0 & 0 & 0 & \rho \end{bmatrix}. \quad (\text{C.9})$$

and its inverse

$$M^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -\frac{u}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 & 0 \\ -\frac{v}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{w}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 & 0 \\ (\gamma-1)q & -(\gamma-1)u & -(\gamma-1)v & -(\gamma-1)w & \gamma-1 & 0 \\ -\frac{\tilde{v}}{\rho} & 0 & 0 & 0 & 0 & \frac{1}{\rho} \end{bmatrix}. \quad (\text{C.10})$$

We now seek the eigenvalues of the matrix

$$\hat{A} = M^{-1} A M = M^{-1} \frac{\partial \tilde{\mathbf{Z}}^c}{\partial \tilde{\mathbf{Q}}} M = M^{-1} \frac{\partial \tilde{\mathbf{Z}}^c}{\partial \tilde{\mathbf{P}}}. \quad (\text{C.11})$$

Due to the similarity transformation, $\hat{A}$ and A will have the same eigenvalues. Their eigenvectors, however, will be different and they are related by the following relations:

$$L^T = \hat{L}^T M^{-1}, \quad R = M \hat{R}. \quad (\text{C.12})$$

First we evaluate the Jacobian $\partial \tilde{\mathbf{Z}}^c / \partial \tilde{\mathbf{P}}$, that is the derivative of the inviscid flux with respect of the primitive variables, which reads:

$$\begin{bmatrix} \theta & \rho k_x & \rho k_y & \rho k_z & 0 & 0 \\ \theta u & \rho k_x u + \rho \theta & \rho k_y u & \rho k_z u & k_x & 0 \\ \theta v & \rho k_x v & \rho k_y v + \rho \theta & \rho k_z v & k_y & 0 \\ \theta w & \rho k_x w & \rho k_y w & \rho k_z w + \rho \theta & k_z & 0 \\ \theta q & \rho(\theta u + q k_x) + p_x^* & \rho(\theta v + q k_y) + p_y^* & \rho(\theta w + q k_z) + p_z^* & \theta^* + \frac{\theta}{\gamma-1} & 0 \\ \theta \tilde{v} & \rho k_x \tilde{v} & \rho k_y \tilde{v} & \rho k_z \tilde{v} & 0 & \rho \theta \end{bmatrix}$$

with

$$p_x^* = \frac{\gamma p k_x}{\gamma - 1}, \quad p_y^* = \frac{\gamma p k_y}{\gamma - 1}, \quad p_z^* = \frac{\gamma p k_z}{\gamma - 1}.$$

Then, we calculate $\hat{A}$ according to eq. (C.11):

$$\hat{A} = \begin{bmatrix} \theta & \rho k_x & \rho k_y & \rho k_z & 0 & 0 \\ 0 & \theta & 0 & 0 & \frac{k_x}{\rho} & 0 \\ 0 & 0 & \theta & 0 & \frac{k_y}{\rho} & 0 \\ 0 & 0 & 0 & \theta & \frac{k_z}{\rho} & 0 \\ 0 & \rho c^2 k_x & \rho c^2 k_y & \rho c^2 k_z & \theta & 0 \\ 0 & 0 & 0 & 0 & 0 & \theta \end{bmatrix} \quad (C.13)$$

where we have introduced the speed of sound $c = \sqrt{\gamma p / \rho}$.

The six eigenvalues of $\hat{A}$ – and of A – can now be evaluated and are:

$$\lambda_1 = \theta - c \sqrt{k_x^2 + k_y^2 + k_z^2} \quad (C.14a)$$

$$\lambda_2 = \theta + c \sqrt{k_x^2 + k_y^2 + k_z^2} \quad (C.14b)$$

$$\lambda_{3,4,5,6} = \theta \quad (C.14c)$$

The eigenvalues are put together in the matrix $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_6)$. Using the right eigenvectors of $\hat{A}$ as the rows of the matrix $\hat{R}$, and evaluating $\hat{L}^T = \hat{R}^{-1}$ we obtain:

$$\hat{R} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ -\frac{c n_x}{\rho} & \frac{c n_x}{\rho} & 0 & 1 & 0 & 0 \\ -\frac{c n_y}{\rho} & \frac{c n_y}{\rho} & 0 & 0 & 1 & 0 \\ -\frac{c n_z}{\rho} & \frac{c n_z}{\rho} & 0 & -\frac{k_x}{k_z} & -\frac{k_y}{k_z} & 0 \\ c^2 & c^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (C.15)$$

$$\hat{L}^T = \begin{bmatrix} 0 & -\frac{\rho n_x}{2c} & -\frac{\rho n_y}{2c} & -\frac{\rho n_z}{2c} & \frac{1}{2c^2} & 0 \\ 0 & \frac{\rho n_x}{2c} & \frac{\rho n_y}{2c} & \frac{\rho n_z}{2c} & -\frac{1}{2c^2} & 0 \\ 1 & 0 & 0 & 0 & -\frac{1}{c^2} & 0 \\ 0 & n_z^2 + n_y^2 & -n_x n_y & -n_x n_z & 0 & 0 \\ 0 & -n_x n_y & n_z^2 + n_x^2 & -n_y n_z & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (C.16)$$

being $\mathbf{n}$ the unit normal vector defined as:

$$\mathbf{n} = \left[\frac{k_x}{\sqrt{k_x^2 + k_y^2 + k_z^2}}, \frac{k_y}{\sqrt{k_x^2 + k_y^2 + k_z^2}}, \frac{k_z}{\sqrt{k_x^2 + k_y^2 + k_z^2}} \right]^T = [n_x, n_y, n_z]^T. \quad (C.17)$$

C.4 THE CONSERVATIVE INVISCID FLUX JACOBIAN

Putting together (C.7), (C.8), (C.12) and (C.14) we can now go back to our original problem. That is, the conservative inviscid flux Jacobian of the augmented system of the URANS equations, written in terms of its eigenvalues, reads:

$$A = \frac{\partial \tilde{\mathbf{Z}}^c}{\partial \tilde{\mathbf{Q}}} = \mathbf{R} \mathbf{\Lambda} \mathbf{L}^T = \mathbf{M} \hat{\mathbf{R}} \mathbf{\Lambda} \hat{\mathbf{L}}^T \mathbf{M}^{-1} \quad (\text{C.18})$$

Note that defining the normal velocity u_n as:

$$u_n = \mathbf{u} \cdot \mathbf{n} = u n_x + v n_y + w n_z \quad (\text{C.19})$$

and abbreviations:

$$a_1 = \frac{\lambda_1 + \lambda_2}{2} \quad (\text{C.20a})$$

$$a_2 = \frac{\lambda_2 - \lambda_1}{2} \quad (\text{C.20b})$$

$$a_3 = a_1 - \lambda_3 \quad (\text{C.20c})$$

$$a_4 = (\gamma - 1)(q(\Delta\rho) - u(\Delta\rho u) - v(\Delta\rho v) - w(\Delta\rho w) + (\Delta\rho E)) \quad (\text{C.20d})$$

$$a_5 = n_x(\Delta\rho u) + n_y(\Delta\rho v) + n_z(\Delta\rho w) - u_n(\Delta\rho) \quad (\text{C.20e})$$

$$a_6 = a_3 a_4 / c^2 + a_2 a_5 / c \quad (\text{C.20f})$$

$$a_7 = a_2 a_4 / c + a_3 a_5 \quad (\text{C.20g})$$

the whole matrix-vector product $A(\mathbf{Q} - \mathbf{g})$ can be expressed conveniently as

$$A(\mathbf{Q} - \mathbf{g}) = A \begin{pmatrix} \Delta\rho \\ \Delta\rho u \\ \Delta\rho v \\ \Delta\rho w \\ \Delta\rho E \\ \Delta\rho \tilde{v} \end{pmatrix} = \begin{pmatrix} \lambda_3(\Delta\rho) + a_6 \\ \lambda_3(\Delta\rho u) + u a_6 + n_x a_7 \\ \lambda_3(\Delta\rho v) + v a_6 + n_y a_7 \\ \lambda_3(\Delta\rho w) + w a_6 + n_z a_7 \\ \lambda_3(\Delta\rho E) + H a_6 + u_n a_7 \\ \lambda_3(\Delta\rho \tilde{v}) + \tilde{v} a_6 \end{pmatrix} \quad (\text{C.21})$$

where again $\mathbf{Q}$ is the vector of the conserved variables, $\mathbf{Q} = [\rho, \rho \mathbf{u}, \rho E, \rho \tilde{v}]^T$, and $\mathbf{g}$ is the vector of the (target) penalty state, i.e. $\mathbf{g} = [\rho_p, \rho_p \mathbf{u}_p, \rho_p E_p, \rho_p \tilde{v}_p]^T$. As mentioned in section 3.2.3, the matrix A is evaluated at the Roe-averaged state between $\mathbf{Q}$ and $\mathbf{g}$.

Equation (C.21) constitutes the main result of this Appendix.

C.5 COMMENTED MAXIMA CODE

In this section we present the commented code used to produce the matrices above. We used Maxima [75], which is an open-source software for symbolic mathematics.

In what follows, the commands to be given to Maxima are printed in *verbatim*. The code can be adapted easily to other turbulence models. Note that the code follows closely the derivation exposed above and the main results have already been presented. For these reasons the output of the computations is not printed here.

Clean up the memory first.

```
kill(all);
```

Construct the convective curvilinear flux $\tilde{\mathbf{Z}}^c$ (here named simply Flux; the subscript `_tmp` is there just to avoid problems later when doing some substitutions).

```
theta_tmp: k_t + k_x*u + k_y*v + k_z*w;
```

```
thetaS_tmp: k_x*u + k_y*v + k_z*w;
```

```
e: p/(rho*(gam-1));
```

```
q_tmp: (u^2 + v^2 + w^2)/2;
```

```
E: e + q_tmp;
```

```
Flux: matrix(
[rho*theta_tmp],
[rho*u*theta_tmp + p*k_x],
[rho*v*theta_tmp + p*k_y],
[rho*w*theta_tmp + p*k_z],
[rho*E*theta_tmp + p*thetaS_tmp],
[rho*nuTilde*theta_tmp]
);
```

Determine the Jacobian of the inviscid flux with respect to the primitive variables, $\frac{\partial \tilde{\mathbf{Z}}^c}{\partial \mathbf{P}}$. Compute one column at the time and then assemble and simplify it.

```
col1: diff(Flux, rho, 1);
col2: diff(Flux, u, 1);
col3: diff(Flux, v, 1);
col4: diff(Flux, w, 1);
col5: diff(Flux, p, 1);
col6: diff(Flux, nuTilde, 1);
```

```
dZdP: matrix(
[col1[1][1],col2[1][1],col3[1][1],col4[1][1],col5[1][1],col6[1][1]],
[col1[2][1],col2[2][1],col3[2][1],col4[2][1],col5[2][1],col6[2][1]],
[col1[3][1],col2[3][1],col3[3][1],col4[3][1],col5[3][1],col6[3][1]],
[col1[4][1],col2[4][1],col3[4][1],col4[4][1],col5[4][1],col6[4][1]],
[col1[5][1],col2[5][1],col3[5][1],col4[5][1],col5[5][1],col6[5][1]],
[col1[6][1],col2[6][1],col3[6][1],col4[6][1],col5[6][1],col6[6][1]]
);
```

```
dZdP: subst(theta, (k_z*w+k_y*v+k_x*u+k_t), dZdP);
dZdP: subst(thetaS, (k_z*w+k_y*v+k_x*u), dZdP);
dZdP: subst(q, ((w^2 + v^2 + u^2)/2), dZdP);
dZdP: ratsimp(dZdP);
```

Matrix to go from primitive to conserved and its inverse.

```
M: matrix(
[1,0,0,0,0,0],
[u,rho,0,0,0,0],
[v,0,rho,0,0,0],
[w,0,0,rho,0,0],
[q, rho*u, rho*v, rho*w, 1/(gam-1),0],
[nuTilde, 0, 0,0,0,rho]
);
```

```
Minverse: factor(ratsimp(invert(M)));
```

```
Minverse[5][1]:q*(gam-1);
```

```
Minverse;
```

Compute $\hat{A}$.

```
A_hat: ratsimp(Minverse.dZdP);
```

```
ratsimp(A_hat[5][2]);
```

and simplify by hand the energy equation components.

```
A_hat[5][1] : 0;
```

```
A_hat[5][2] : rho*c^2*k_x;
A_hat[5][3] : rho*c^2*k_y;
A_hat[5][4] : rho*c^2*k_z;
```

```
A_hat;
```

Compute the eigenvalues and the right eigenvectors, $(\hat{A} - \lambda_i I)\hat{\mathbf{r}}_i = 0$

```
[vals, vecs] : eigenvectors(A_hat);
```

```
for i thru length (vals[1]) do disp (val[i] = vals[1][i],
    mult[i] = vals[2][i], vec[i] = vecs[i]);
```

Construct the matrix $\hat{\mathbf{R}}$ (whose columns are the right eigenvectors) and simplify it.

```
R_hat: matrix(
[vecs[1][1][1],vecs[2][1][1],vecs[3][1][1],vecs[3][2][1],vecs[3][3][1],vecs[3][4][1]],
[vecs[1][1][2],vecs[2][1][2],vecs[3][1][2],vecs[3][2][2],vecs[3][3][2],vecs[3][4][2]],
[vecs[1][1][3],vecs[2][1][3],vecs[3][1][3],vecs[3][2][3],vecs[3][3][3],vecs[3][4][3]],
[vecs[1][1][4],vecs[2][1][4],vecs[3][1][4],vecs[3][2][4],vecs[3][3][4],vecs[3][4][4]],
[vecs[1][1][5],vecs[2][1][5],vecs[3][1][5],vecs[3][2][5],vecs[3][3][5],vecs[3][4][5]],
[vecs[1][1][6],vecs[2][1][6],vecs[3][1][6],vecs[3][2][6],vecs[3][3][5],vecs[3][4][6]]
);
R_hat: subst(area, sqrt(k_z^2+k_y^2+k_x^2), R_hat);
R_hat: factor(ratsimp(R_hat));
R_hat: subst(area^2, k_z^2+k_y^2+k_x^2, R_hat);
```

Calculate the matrix $\hat{\mathbf{L}}^T$ which holds the left eigenvectors in its rows ($\hat{\mathbf{L}}^T = \hat{\mathbf{R}}^{-1}$)

```
Ltranspose_hat: invert(R_hat);
Ltranspose_hat: factor(ratsimp(Ltranspose_hat));
Ltranspose_hat: subst(area^2, k_z^2+k_y^2+k_x^2, Ltranspose_hat);
```

Now go to conservative, i.e. compute $\mathbf{L}^T$ and $\mathbf{R}$ (no hat).

```
Ltranspose: Ltranspose_hat.Minverse;
```

```
R: M.R_hat;
```

Finally the conservative Jacobian $\mathbf{A}$ expressed in terms of its eigenvalues λ_i .

```
EigenDiag: matrix(
  [lambda_1,0,0,0,0,0],
  [0,lambda_2,0,0,0,0],
  [0,0,lambda_3,0,0,0],
  [0,0,0,lambda_3,0,0],
  [0,0,0,0,lambda_3,0],
  [0,0,0,0,0,lambda_3]
);
```

```
A: R.EigenDiag.Ltranspose;
```

The output reveals that this is an extremely long expression. Since it is rather difficult to tell Maxima how to simplify long expressions, we check instead that the simplifications proposed in eq. (C.20) are consistent with what we have just derived. So compute the matrix vector multiplication shown in eq. (C.21) and the product $A \cdot dQ$. They should be the same.

```
dQ: matrix(
  [dr],
  [dru],
  [drv],
  [drw],
  [drE],
  [drNuTilde]
);
```

```
n_x: k_x/area;
n_y: k_y/area;
n_z: k_z/area;
u_n: n_x*u + n_y*v + n_z*w;
H: p*gam/(gam -1)/rho +q;
```

```
a_1: (lambda_1 + lambda_2)/2;
a_2: (lambda_2 - lambda_1)/2;
a_3: a_1 - lambda_3;
a_4: (gam -1)*(q*dr - u*dru -v*drv -w*drw + drE);
a_5: n_x*dru + n_y*drv + n_z*drw - u_n*dr;
a_6: a_3*a_4/c^2 + a_2*a_5/c;
a_7: a_2*a_4/c + a_3*a_5;
```

```
matVecMult: matrix(
  [lambda_3*dr + a_6],
  [lambda_3*dru + a_6*u + n_x*a_7],
```

```
[lambda_3*drv + a_6*v + n_y*a_7],
[lambda_3*drw + a_6*w + n_z*a_7],
[lambda_3*drE + a_6*H + u_n*a_7],
[lambda_3*drNuTilde + a_6*nuTilde]
);
```

```
AdQ: A.dQ;
```

If our simplifications a_i are correct, the last two results should be identical. Check that their difference is zero.

```
zeroVec: matVecMult-AdQ$
```

```
zeroVec: expand(zeroVec)$
```

```
zeroVec: ratsimp(zeroVec);
```

Looking at the last output, it is clear that the density and $\rho\tilde{v}$ components are zero. It remains to show that is also the case for the momentum and energy components. For the momentum components it easy to see that once we recall that $k_z^2 + k_y^2 + k_x^2 = \text{area}^2$. For the energy, we set two λ_i at the time to zero and we check the what's left is zero as well.

```
tmp: subst(0, lambda_1, zeroVec[5][1])$
tmp: subst(0, lambda_2, tmp)$
```

Simplify what we just obtained and note that $\rho c^2 = \rho \gamma p / \rho = \gamma p$ and that $u^2 + v^2 + w^2 = 2q$. This means that what follows is zero.

```
tmp: ratsimp(tmp);
```

Now go back to the energy component of the zeroVec vector and set λ_1 and λ_3 to zero.

```
tmp: subst(0, lambda_1, zeroVec[5][1])$
tmp: subst(0, lambda_3, tmp)$
tmp: ratsimp(tmp);
```

Again, what we just obtained is zero. Finally, set λ_2 and λ_3 to zero in the energy component of zeroVec.

```
tmp: subst(0, lambda_2, zeroVec[5][1])$
tmp: subst(0, lambda_3, tmp)$
tmp: ratsimp(tmp);
```

Again, we only have to remember that $\gamma p = \rho c^2$ to see that this is zero. Therefore the two vectors `matVecMul` and `AdQ` are identical and the derivation holds.

D

BUTCHER TABLEAU FOR THE ESDIRK SCHEMES

Reported here are the Butcher coefficients for the three ESDIRK schemes employed in this work. They are denoted as ESDIRK₃, ESDIRK₄ and ESDIRK₅, where the last digit indicates the order of accuracy. The coefficients are reproduced from [58].

Table D.1: Butcher tableau for the ESDIRK₃ scheme, third order scheme with 4 stages.

0	0	0	0	0
$\frac{1767732205903}{2027836641118}$	$\frac{1767732205903}{4055673282236}$	$\frac{1767732205903}{4055673282236}$	0	0
$\frac{3}{5}$	$\frac{2746238789719}{10658868560708}$	$-\frac{640167445237}{6845629431997}$	$\frac{1767732205903}{4055673282236}$	0
1	$\frac{1471266399579}{7840856788654}$	$-\frac{4482444167858}{7529755066697}$	$\frac{11266239266428}{11593286722821}$	$\frac{1767732205903}{4055673282236}$
	$\frac{1471266399579}{7840856788654}$	$-\frac{4482444167858}{7529755066697}$	$\frac{11266239266428}{11593286722821}$	$\frac{1767732205903}{4055673282236}$

Table D.2: Butcher tableau for the ESDIRK₄ scheme, fourth order scheme with 6 stages.

0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0
$\frac{83}{250}$	$\frac{8611}{62500}$	$-\frac{1743}{31250}$	$\frac{1}{4}$	0	0	0
$\frac{31}{50}$	$\frac{5012029}{34652500}$	$-\frac{654441}{2922500}$	$\frac{174375}{388108}$	$\frac{1}{4}$	0	0
$\frac{17}{20}$	$\frac{15267082809}{155376265600}$	$-\frac{71443401}{120774400}$	$\frac{730878875}{902184768}$	$\frac{2285395}{8070912}$	$\frac{1}{4}$	0
1	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$-\frac{2260}{8211}$	$\frac{1}{4}$
	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$-\frac{2260}{8211}$	$\frac{1}{4}$

Table D.3: Butcher tableau for the ESDIRK5 scheme, fifth order scheme with 8 stages.

0	0	0	0	0	0	0	0	0	0
$\frac{41}{100}$	$\frac{41}{200}$	$\frac{41}{200}$	0	0	0	0	0	0	0
$\frac{2935347310677}{11292855782101}$	$\frac{41}{400}$	$-\frac{567603406766}{11931857230679}$	$\frac{41}{200}$	$\frac{41}{200}$	0	0	0	0	0
$\frac{1426016391358}{7196633302097}$	$\frac{683785636431}{9252920307686}$	0	$-\frac{110385047103}{1367015193373}$	$\frac{41}{200}$	0	0	0	0	0
$\frac{92}{100}$	$\frac{3016520224154}{10081342136671}$	0	$\frac{30586259806659}{12414158314087}$	$-\frac{22760509404356}{11113319521817}$	$\frac{41}{200}$	0	0	0	0
$\frac{24}{100}$	$\frac{218866479029}{1489978393911}$	0	$\frac{638256894668}{5436446318841}$	$-\frac{1179710474555}{5321154724896}$	$-\frac{60928119172}{8023461067671}$	$\frac{41}{200}$	0	0	0
$\frac{3}{5}$	$\frac{1020004230633}{5715676835656}$	0	$\frac{25762820946817}{2526394035340}$	$-\frac{2161375909145}{9755907335909}$	$-\frac{211217309593}{5846859502534}$	$-\frac{4269925059573}{7827059040749}$	$\frac{41}{200}$	0	0
1	$-\frac{872700587467}{9133579230613}$	0	0	$\frac{22348218063261}{9555858737531}$	$-\frac{1143369518992}{8141816002931}$	$-\frac{39379526789629}{19018526304540}$	$\frac{32727382324388}{42900044865799}$	$\frac{41}{200}$	0
	$-\frac{872700587467}{9133579230613}$	0	0	$\frac{22348218063261}{9555858737531}$	$-\frac{1143369518992}{8141816002931}$	$-\frac{39379526789629}{19018526304540}$	$\frac{32727382324388}{42900044865799}$	$\frac{41}{200}$	0

E VERIFICATION OF THE IMPLEMENTATION OF THE SPALART-ALLMARAS TURBULENCE MODEL

This appendix presents the verification of the implementation of the SA turbulence model. Two test cases have been analyzed in detail: the problem of the flow over a bump in a channel, and the problem of the flow over NACA4412 airfoil featuring trailing edge flow separation. All grids, reference data and extensive description of the test cases are available at <http://turbmodels.larc.nasa.gov/> (see also [97]).

E.1 BUMP IN A CHANNEL

The computational domain for this test case is shown in fig. E.1. Note that the reference length for this test case is $L = 1\text{m}$; the grid coordinates (x, y) , in fig. E.1 and in the remainder of this section, are meant to be non-dimensional, i.e. $x = x/L$ and $y = y/L$. The bump extends from $x = 0$ to $x = 1.5$; its maximum height is $(y - y_0) = 0.05$ (at $x = 0.75$), see fig. E.1b. The flow conditions are $Ma = 0.2$ and

Figure E.1: Computational domain for the flow over a bump in a channel (<http://turbmodels.larc.nasa.gov/>). Note that the x and y axis use two independent scales in order to highlight the presence of the bump.

$Re = 3 \cdot 10^6$. We used a far-field boundary condition at the inflow, top and outflow boundary. This 2D test case has been run for three regularly refined grids of di-

mensions 177×81 , 353×161 and 705×231 . Two schemes have been used, namely second and fifth order. Figure E.2 shows the distribution of the non-dimensional eddy viscosity at $x = 0.75$ (at the crest of the bump) obtained with the two schemes on the three grids. Convergence is apparent as well as that the fifth order solution is quite well converged already for the medium grid. Next we compare our 5th order results for the fine grid with the reference data, which are the results obtained with the code FUN3D on a finer grid of dimensions 1409×641 . FUN3D uses a node-centered unstructured-grid method, featuring a second-order accurate discretization for the mean flow equations, and a first-order accurate upwind scheme for the advective terms of the SA turbulence model equation. The boundary layer profile at two locations on the bump is plotted in fig. E.3. In fig. E.4 we present the pressure and skin friction coefficient on the bump. The distributions of the non-dimensional eddy viscosity, (again) at $x = 0.75$, and its maximum value in the boundary layer, are shown in fig. E.5. Finally, we present the contours of the non-dimensional eddy viscosity close to the bump in fig. E.6. All results are in very good agreement with the reference results.

Figure E.2: μ_t/μ_∞ distribution at $x = 0.75$ obtained with the 2nd and 5th order scheme for the three grids. $Ma = 0.2$, $Re = 3 \cdot 10^6$.

Figure E.3: Boundary layer profile at $x = 0.75$ and $x = 1.20148$. Every other solution point of the Reference solution is shown. $Ma = 0.2$, $Re = 3 \cdot 10^6$.

E.2 NACA4412

This test case consists of the subsonic flow around NACA4412 airfoil. The flow conditions are:

$$Ma = 0.09, \quad Re = 1.52 \cdot 10^6, \quad AoA = 13.87^\circ. \quad (E.1)$$

Figure E.4: Pressure and skin friction coefficient along the surface of the bump. Every other solution point of the Reference is shown. $Ma = 0.2$, $Re = 3 \cdot 10^6$.

Figure E.5: μ_t/μ_∞ distribution. Every other solution point of the Reference solution is shown. $Ma = 0.2$, $Re = 3 \cdot 10^6$.

Figure E.6: Contours of μ_t/μ_∞ . $Ma = 0.2$, $Re = 3 \cdot 10^6$.

Experiments have shown that, for this configuration, the boundary layer separates on the upper surface close to the trailing edge, see fig. E.7. We have carried out a numerical simulation for three provided, regularly coarsened, O-grids, the finest of which has dimensions 481×253 . In the context of verification of the implementation of the turbulence model, we are interested in numerical results obtained with trusted codes rather than experimental data. Thus, our results, obtained with the method described in this work (called MOOSE) for the three grids using the 2nd and 5th order scheme, are compared with reference numerical results. The reference results were obtained for the fine grid with CFL3D, a cell-centered finite volume method for structured grids. CFL3D uses a second-order accurate discretization for the mean flow equations, and a first-order accurate upwind scheme for the advective terms of the SA turbulence model equation.

We present in figs. E.8 to E.10 the non-dimensional x - and y -velocity components, indicated with u/U_{ref} and v/U_{ref} , respectively, along the lines normal to the airfoil located at the following locations (see fig. E.7):

$$x/c = [0.06753, 0.7308, 0.7863, 0.8418, 0.8973, 0.9528].$$

Note that the fifth order scheme provides an almost fully converged solution already for the medium grid. Again, very good agreement with the reference data is observed.

Figure E.7: Experimental results for the NACA4412 (<http://turbmodels.larc.nasa.gov/>).
 $Ma = 0.09$, $Re = 1.52 \cdot 10^6$, $AoA = 13.87^\circ$.

(a) x-component velocity, 2nd order scheme (b) y-component velocity, 2nd order scheme

(c) x-component velocity, 5th order scheme (d) y-component velocity, 5th order scheme

Figure E.8: Velocity profiles on the *coarse* grid (CFL3D results are on the fine grid).
NACA4412, $Ma = 0.09$, $Re = 1.52 \cdot 10^6$, $AoA = 13.87^\circ$.

(a) x-component velocity, 2nd order scheme (b) y-component velocity, 2nd order scheme

(c) x-component velocity, 5th order scheme (d) y-component velocity, 5th order scheme

Figure E.9: Velocity profiles on the *medium grid* (CFL3D results are on the fine grid).
NACA4412, $Ma = 0.09$, $Re = 1.52 \cdot 10^6$, $AoA = 13.87^\circ$.

Figure E.10: Velocity profiles on the *fine grid*. NACA4412, $Ma = 0.09$, $Re = 1.52 \cdot 10^6$, $AoA = 13.87^\circ$.

ACKNOWLEDGEMENTS

The research leading to the results presented in this thesis has received funding through the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7 under REA grant agreement No. PITN-GA-2011-290042 for the project COPA-GT. The Netherlands Organisation for Scientific Research (NWO) is acknowledged for funding the computational resources of the SurfSARA machines.

Many times during the past four years I have realized that I am indebted to a large number of people who, in different ways, have contributed to this thesis. Here I would like to express my gratitude to all of them, as this work could not have been completed without their help and support.

I cannot thank Edwin van der Weide, the co-promotor of this thesis and my daily supervisor, enough for his patience, for his open-door policy (that I have exploited many times) and for his continuous, always helpful, support on (not only) technical-related matters. He has been my mentor and I have been his tormentor. I have asked him countless questions and given him few answers. It is hard to find such a large amount of patience and knowledge packed in one person, no matter how tall that person is.

I would like to thank Harry Hoeijmakers, the promotor of the thesis, for his guidance and advices, especially at the beginning and at the end of my PhD journey. His passion for fluid dynamics and for teaching is as profound as contagious. I believe his comments and feedback have made this thesis a much better one.

Many thanks to Kees Venner, who has given me the possibility to finish this work without having to worry about anything else. I remember very clearly a crash-course he gave me about multigrid in his office. Even though I ended up not using multigrid in this work, the learning experience was inspirational.

I would like to thank the members of the graduation committee for the time and effort spent for reviewing my thesis, and for their valuable suggestions. In particular, I would like to thank Piero Colonna for his interesting comments, and Jim Kok for our fruitful discussions. I am also grateful to Magnus Svård for his support and useful answers to my questions about SBP-SAT.

I would like to thank Martin Almquist and Ken Mattsson for our collaboration; working with you has been a pleasure.

Furthermore, I would like to thank the staff and the students of the Engineering Fluid Dynamics Group for making me feel always welcome. The working environment has been more enjoyable than I could ever imagine. I would like to express my gratitude to Wouter den Breeijen for helping me with all the computing-related issues that I encountered.

A sincere thank you to Enrico Sciubba, for always finding the time to hear my story.

There are a lot of ups and downs during a PhD, but I am lucky enough to have many friends who have also given me plenty of support and helped me during this memorable experience. Even if I will not be able to mention everyone, I am grateful to all of them. Thanks to Simon, who quickly became a dear friend, rather than a colleague; your kids have a great dad. Caterina e Federico, thanks for the company and for the awesome food; I am sure the newcomer(s) will enjoy it as well. Aga and Matteo, thank you for all the good times (licorice-vodka shots included) during these years. Thanks to Ludovica and Alberto, Loredana and Tommaso, for making me think that Turin is a city where I could live. Marco, thank you for welcoming me as a flatmate. Giuseppe, thank you for the nights spent playing Mannarino's songs. Thanks to Alex, for remembering me how painful it can be to learn how to snowboard, and how enjoyable it is when you know how to do it. Thanks to Riccardo, for the showing me what hard work in the gym means. Thanks to Mattia, for being my pal during the very first days here in Enschede. Davide, thank you for making me realize that people from Milan can actually be funny, and, more importantly, good friends. Thanks to the entire climbing community in Enschede. Nothing clears my mind and soul like a good two hours of climbing with friends. Thanks to the COPA-GT fellows, in particular to Sebastian and Stefano, for the good times at VKI. A warm thank you goes to my dear friends in Rome. In particular, to Damiano, I know I can always count on you, no matter what; Benedetta, our psycho-talks have always made me feel lighter and happier; Sbracchi, always good ice-creams and even better chats. Thanks to the guys from Bologna, especially to Giampiero, Filippo and Frabs, for remembering me where my journey started.

My warmest thank you is for my family. The support of my father Felice, my mother Mariuccia and my brother Giulio, has always been incredibly important. A special thank you to my aunt Carla, for making me smile since I was a kid. Marina and Bianca, I could not be happier that you have joined this wonderful family. I owe them this experience.

Finally, I would like to express my deepest gratitude to Virginia: thank you for withstanding the weight of my soul, for being there every time I need it, for making me smile when thinking about our future, and, above all, for all the love and happiness you bring in my life. I am a lucky man.

Giorgio Giangaspero, April 2016