DLR-IB-AS-BS-2022-125

TAU Hyperflex Report

Forschungsbericht

Autor
Stefan Langer, Axel Schwöppe

DLR Deutsches Zentrum für Luft- und Raumfahrt

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

DLR-IB-AS-BS-2022-125

TAU Hyperflex Report

Stefan Langer, Axel Schwöppe

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

ISSN 1614-7790

Stufe der Zugänglichkeit: 1
Braunschweig, im September 2022

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

Verfasser:
Stefan Langer, Axel Schwöppe

Abteilung: Center of Computer Applications in Aerospace Science and Engineering

Abteilungsleiter:
Prof. Dr. S. Görtz

Der Bericht enthält:

73	Seiten
36	Bilder
5	Tabellen
37	Literaturstellen

Outline

Introduction

Computational meshes used in practical aerodynamic applications may possess various properties. For example, structured or unstructured ordering may be used to utilize efficient data structures or simplify mesh generation; high-aspect-ratio cells may be used to resolve boundary layers of high-Reynolds-number flows; the cell orientation may align with specific flow directions to maximize benefits of certain discretization techniques, such as approximated Riemann solvers. The HyperFlex version of the DLR TAU-code provides a suit of computational algorithms and solution techniques that can be canonically switched depending on mesh characteristics to optimize accuracy, efficiency, and robustness of flow solutions. The current HyperFlex implementation derives from experiences gained during previous efforts concerned with enabling structured data and algorithms in an unstructured code environment, developing an hierarchy of preconditioning techniques embedded into a multistage Runge-Kutta method, and implementing alternative multigrid techniques. The practical aspects of code design, such as maintainability and expendability, are also taken into account.

Hyperflex formulation of the spatial discretization

1 Introduction

This report describes the spatial discretization used in TAU HyperFlex and emphasizes differences from the baseline TAU discretizations. The report is organized as follows. The differences between structured and unstructured addressing of mesh components are explained. An extension of the unstructured edge-based data structure, which enables a canonical switch between structured and unstructured second order discretizations, is considered. Solutions obtained with structured discretizations in TAU and FLOWer [4] are compared. Additionally, a modified artificial matrix dissipation which improves the accuracy and robustness of current dissipation schemes and is suitable for the HyperFlex preconditioned Runge-Kutta scheme is introduced. Solutions obtained with the new matrix dissipation are compared with solutions obtained with current dissipation schemes in TAU.

2 Structured and unstructured addressing

In the context of finite volume methods, a mesh is identified as a structured mesh if it contains only quadrilateral elements in 2D or only hexahedra in 3D, and all mesh points are uniquely identified in a linear address space by discrete coordinates, denoted as i,j,k coordinates, corresponding to the x, y, z coordinates of the physical space. The advantage of this discrete computational space is that it allows to store local information, such as flow variables, in a structured data structure. With this property, a (structured) finite volume code can efficiently address neighboring points by increasing and decreasing the i,j,k indices (Fig. 2.1).

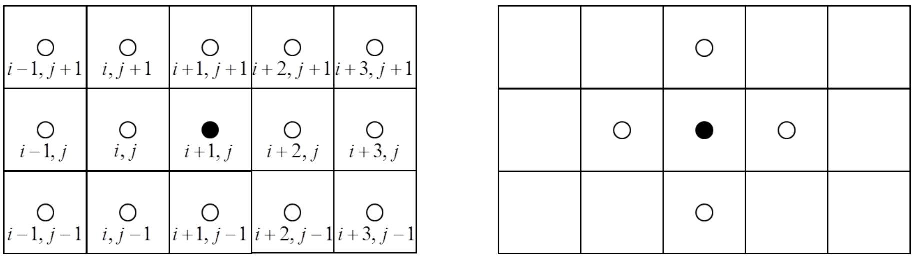

Figure 2.1: Identification of surrounding and neighboring points (white circles) from one point (black circle) in an structured and an unstructured data structure

An unstructured mesh may contain several different polyhedral elements, typically tetrahedra, hexahedra, prisms and pyramids in 3-D. The main advantages of unstructured meshes are nearly automatic mesh generation and a relatively easy mesh adaptation, i.e., solution dependent refinement and coarsening. On the other hand, the points of an unstructured mesh cannot be identified directly by mesh indices as in structured meshes (Fig. 2.1). Hence unstructured meshes are stored usually by an edge-based data structure [1].

In general, a structured mesh can be considered as a specific unstructured mesh with an ordering of the hexahedrons in the discrete computational space. Therefore, on the same mesh, a structured code differs from the corresponding unstructured code only in algorithms utilizing the specific ordering in the discrete computational space. Implicit temporal discretizations widely use the discrete computational space. Discretization of implicit schemes on structured meshes results in well-ordered systems which can be handled more efficiently by structured techniques rather than purely unstructured techniques. This property can also be exploited in unstructured codes by storing the i,j,k addresses for the structured parts of a mesh in addition to the edge-based data structure [8]. Generalized implicit techniques [7], [5] represent more sophisticated alternatives and are used in TAU HyperFlex.

Spatial discretizations in structured and unstructured finite volume codes use different sets of points (stencils) to compute terms of the discrete equations. For example, a second-order finite volume code (e.g., FLOWer or TAU) must identify neighbors of neighbors to discretize the convective terms.

The computational stencils can vary depending on available point ordering and hence the solution accuracy can vary as well. Again this variability can be overcome if an unstructured code stores the i,j,k coordinates for the structured parts of a mesh in addition to the edge-based data structure [8].

If only second-order discretizations are needed, there is another way to take advantage of structured stencils on unstructured meshes. One can enhance the edge-based data structure by so-called face neighbors in the structured parts of the mesh. Face neighbors are added to an edge, if the edge end points and their neighbors are located on a smooth line (Fig. 2.2).

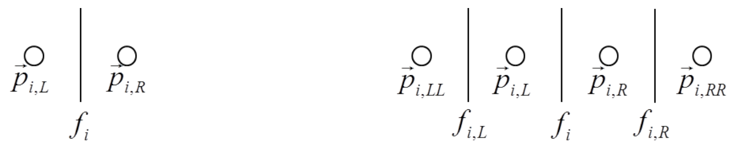

Figure 2.2: Identification of edge-points in an edge-based data structure (left) and of edge-faces in an enhanced edge-based data structure (right)

With this enhanced edge-based data structure, second-order stencils can be identical to those of structured codes on smooth meshes Fig. 2.3. TAU already switches canonically to this data structure in structured-mesh regions to use the same stencils as structured codes for the MUSCL reconstruction of upwind schemes [3]. The following chapter shows that a combination of the enhanced edge-based data structure and a central scheme with artificial dissipation used for discretizing the convective terms is capable to reproduce accuracy of a structured code.

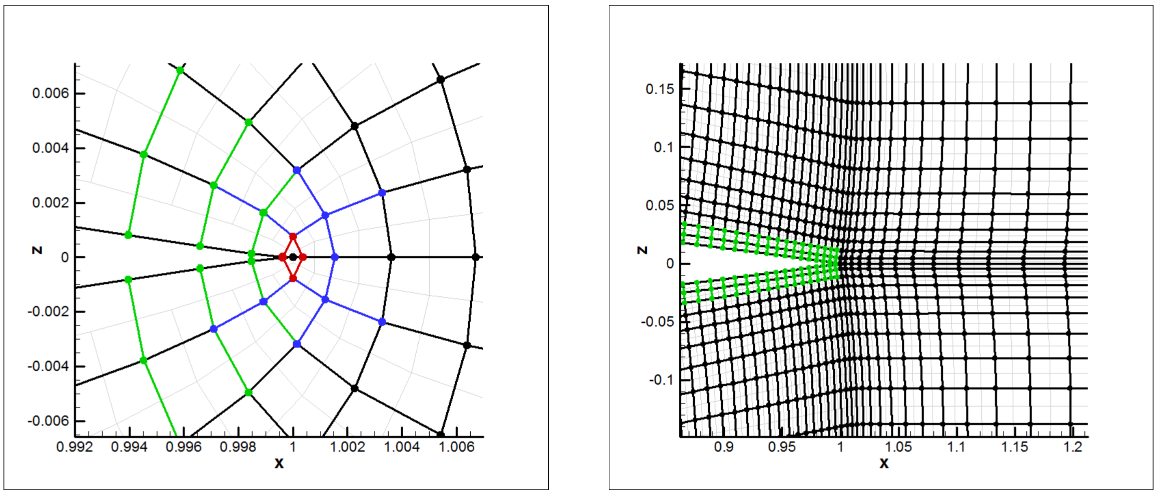

Figure 2.3: Identified structured inner stencils (black), structured boundary stencils (green) and unstructured stencils (blue, red) in a non-smooth (left) and smooth (right) structured mesh

It should be noted that the storage of discrete i,j,k addresses in addition to the common edge-based data structure in an unstructured code environment implies additional memory resources, computation and communication time, and efforts to develop and maintain the code. Redundant storage of data and implementation of schemes and algorithms cannot be avoided. Hence it is recommended to avoid the application of these extensions unless clear advantages compared to purely unstructured discretizations cannot be observed.

3 Structured and unstructured spatial discretization

Spatial discretizations determine the accuracy of a finite volume code. A common opinion is that structured discretizations are more accurate than unstructured discretizations due to availability of the i,j,k addresses in the discrete computational space. Hence TAU HyperFlex is supposed to switch to a structured discretization in structured mesh regions to obtain better solution accuracy. In this section, accuracy of structured and unstructured spatial discretizations implemented in TAU HyperFlex is compared. Different structured and unstructured discretization stencils are mainly occurred for two terms: (1) second-order discretization of the convective fluxes and (2) the discretization of viscous fluxes. The latter is expected to be less important for accuracy of high Reynolds number flow solutions and therefore is not considered in this report.

The structured finite volume code FLOWer [4] is used as reference to verify implementation of the structured discretization in TAU HyperFlex and to compare with the structured and unstructured TAU solutions. The structured version of the central scheme with artificial dissipation of FLOWer has already been compared in detail to the unstructured version of TAU [8]. The effects of different discretizations have been shown. Solutions obtained with a prototype code called TAUijk, which includes the additional i,j,k addressing in structured mesh regions, have also been compared to FLOWer solutions. This study indicated that the solution differences become less important for more complex flows of practical interest. To confirm this observation the structured discretization was directly implemented in TAU HyperFlex.

The TAU HyperFlex structured discretization is implemented by using the enhanced edge-based data structure and is identical to the structured discretization of FLOWer. The artificial dissipation of the central scheme is a combination of the first and third differences of flow variables. Its unstructured version is defined with respect to an edge (or face) between nodes i and j by

$$\boldsymbol{d}_{ij} = \phi_{ij}\left|\boldsymbol{A}_{ij}\right|\left\{\varepsilon_{ij}^{(2)} sc_{ij}^{(2)}\left(\boldsymbol{W}_j - \boldsymbol{W}_i\right) - \varepsilon_{ij}^{(4)} sc_{ij}^{(4)}\left(L_j(\boldsymbol{W}) - L_i(\boldsymbol{W})\right)\right\}, \tag{3.1}$$

where $\boldsymbol{W}$ is the vector of conservative variables. The third difference of the structured scheme [1] is approximated using the undivided Laplacian operator

$$L_i(\boldsymbol{W}) = \sum_{k\in N(i)} \left(\boldsymbol{W}_j - \boldsymbol{W}_i\right), \tag{3.2}$$

where the set $N(i)$ denotes the neighbors of point i; $\varepsilon_{ij}^{(2)}$ and $\varepsilon_{ij}^{(4)}$ are adaptive coefficients designed to switch between the first and second order dissipation. The coefficients are adapted to the local flow gradients

$$\varepsilon_{ij}^{(2)} = k_2 max\left(\psi_i, \psi_j\right), \varepsilon_{ij}^{(4)} = k_4 - \varepsilon_{ij}^{(2)}, \tag{3.3}$$

where the pressure sensor ψ_i is defined as

$$\psi_i = \left|\frac{\sum\limits_{j\in N(i)} \left(p_j - p_i\right)}{\sum\limits_{j\in N(i)} \left(p_j + p_i\right)}\right| \tag{3.4}$$

and k_2 and k_4 are constants, which can be defined by the user to control the amount of dissipation. The scaling factors

$$sc_{ij}^{(2)} = \frac{3}{n_i} + \frac{3}{n_j}, sc_{ij}^{(4)} = \frac{9}{n_i\left(1+n_j\right)} + \frac{9}{n_i\left(1+n_j\right)}, \tag{3.5}$$

where n_i denotes the number of neighbors of cell i, are introduced in order to avoid a strong dependency of the face dissipation on the number of neighbors surrounding the adjacent cells [3]. The cell stretching coefficient, ϕ_{ij}, is computed as

$$\phi_{ij} = 4\frac{r_i r_j}{r_i + r_j}, r_i = \left(\frac{0.5 \sum\limits_{k\in N(i)} \lambda_{ik} - \lambda_{ij}}{2\lambda_{ij}}\right)^{\omega}, \tag{3.6}$$

where ω is typically set to 0.5 and λ_{ik} is the spectral radius of the matrix $\boldsymbol{A}_{ij}$. The influence of the cell stretching coefficient is explained in detail in Ref. [2]. The matrix $\boldsymbol{A}_{ij}$ [1] denotes Jacobian of the convective flux evaluated at the face ij. In this form, the dissipation is called matrix dissipation [1]. If the spectral radius of $\boldsymbol{A}_{ij}$ is used instead, the scheme is reduced to the so-called scalar dissipation scheme [1].

Three terms of the artificial dissipation scheme of FLOWer are discretized differently. The sums of the first differences in the pressure switch (3.4) are calculated by the directional second differences in the computational space,

$$\nu_{i,j,k} = \frac{p_{i+1,j,k} - 2p_{i,j,k} + p_{i-1,j,k}}{p_{i+1,j,k} + 2p_{i,j,k} + p_{i-1,j,k}}. \tag{3.7}$$

The cell stretching coefficient is implemented as

$$\phi^I_{i,j,k} = 1 + \max\left(\left(\frac{\lambda^J_{i,j,k}}{\lambda^I_{i,j,k}}\right)^{\omega}, \left(\frac{\lambda^K_{i,j,k}}{\lambda^I_{i,j,k}}\right)^{\omega}\right), \tag{3.8}$$

where I, J, K denotes the computational direction. Instead of the Laplacian 3.2 the directional second-difference approximation is used. Additionally, the structured scheme is independent of scaling concerning the number of neighbors (3.5).

The structured discretization of FLOWer is implemented into a developer version of TAU HyperFlex using the stencils provided by the enhanced edge-based data structure. Additionally, the implementation uses the cell stretching coefficient suggested in [8] that is equal to (3.8) in 2-D. This version of HyperFlex TAU is denoted as canonical below, in particular in figures. The implementation is tested and compared to FLOWer and TAU using several test cases.

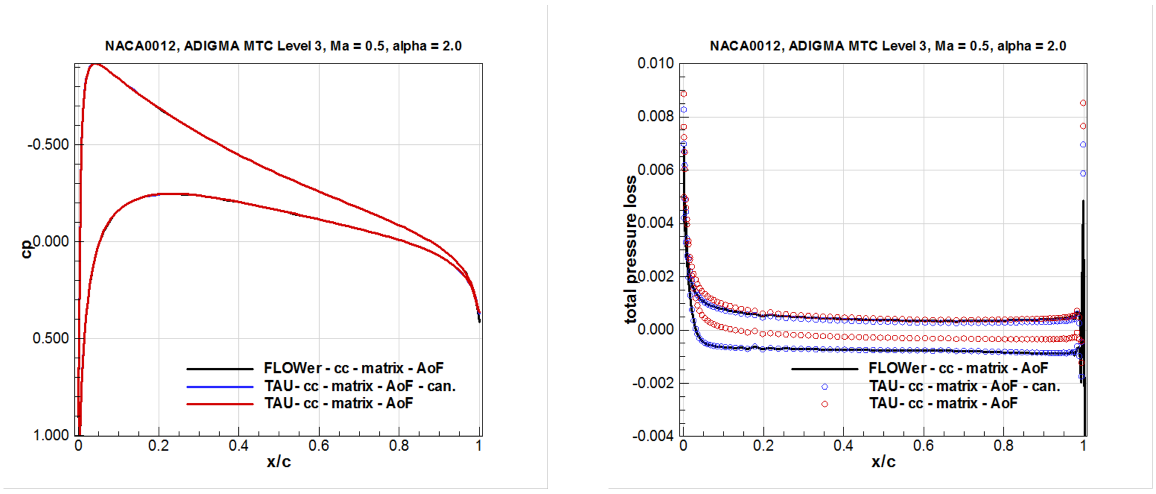

Figure 3.1: Comparison of results concerning a subsonic Euler flow based on structured and unstructured discretizations of the convective terms

A subsonic Euler test case is used to compare the central second-order discretizations of the convective terms with the matrix dissipation. A structured 224x32 element mesh is defined around a NACA0012 profile. To guaranty that other discretization terms are the same, FLOWer and TAU solutions are computed on the cell-centered grid metric (cc) and use the average-of-flux formulation of the central scheme (AoF). Fig 3.1 shows the distribution of pressure coefficient and total pressure loss of FLOWer, TAU, and TAU HyperFlex (canonical) results.

The comparison shows that the structured solutions of FLOWer and TAU HyperFlex are almost identical. This close similarity indicates that other discretization components, such as boundary conditions, are implemented in a similar way as expected. As expected [8], there are some differences between the structured solutions and the unstructured solution of TAU, but they are small.

A transonic RANS test case is used to verify the combination of first- and second-order dissipation at shocks and to evaluate the influence of the viscous terms. A 184x40 structured element mesh is defined around an RAE2822 profile. The flow conditions are $Ma = 0.734$, $\alpha = 2.79$ and $Re = 6.5e6$. Fig 3.2 compares the pressure distribution of FLOWer and TAU HyperFlex (canonical) using the cell-centered grid metric, average of flux discretization of the central difference and a turbulence model. The solutions are nearly identical.

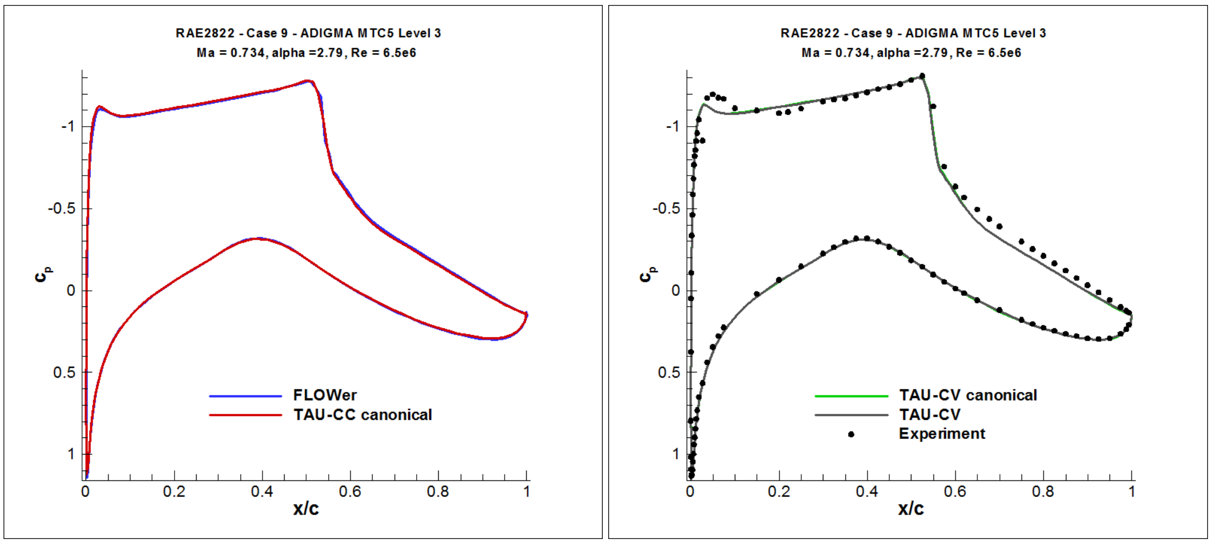

Figure 3.2: Comparison of results concerning a transonic flow based on structured and unstructured discretizations of the convective terms of the RANS equations

The cell-vertex grid metric is the default metric of TAU. Fig. 3.2 (right) compares the cell-vertex solutions obtained with the enhanced edge-based data structure (TAU-CV canonical) and with the default edge-based data structure (TAU-CV); no significant difference has been observed. These results indicate that the differences between the structured and the unstructured discretizations of the convective terms and specifics of the viscous term discretizations have negligible effects on accuracy and that the combination of first and second-order dissipation terms at shocks using the enhanced edge-based data structure is implemented correctly and lead to expected results.

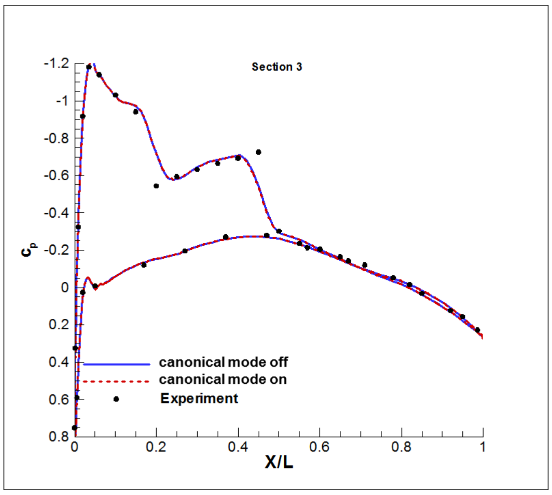

Figure 3.3: Influence of the canonical switch (Onera M6, cut plane at section 3)

A 3-D RANS test is used to compare the influence of the canonical switch between structured and unstructured spatial discretizations. A mixed-element mesh with 265276 points is generated around an Onera-M6 airfoil; hexahedra are used to resolve the boundary layer, prisms are used at the wing tip to simplify the mesh generation, and tetrahedral are used in the farfield. The flow conditions are $Ma = 0.84$, $\alpha = 3.06$ and $Re = 11.72e6$. This case is known to be very sensitive to small changes in the artificial dissipation. Fig 3.3 shows two pressure distribution computed with the cell-vertex formulation of TAU HyperFlex at the surface cut at section 3. The blue distribution represents the standard TAU discretization using the unstructured central dissipation in each mesh region. The dashed red distribution is calculated using the canonical switch which uses the structured discretization based on the enhanced edge-based data structure in mesh regions of hexahedra and prisms and the unstructured discretization in the rest of the mesh. Again the differences between the structured and the unstructured solutions are negligible.

It can be concluded that the difference between the structured and unstructured implementations of the central discretization scheme and the viscous terms has a little influence on discrete solutions on the same mesh. Taking into account efforts required for code development, maintenance and expansion, redundant implementation of algorithms, and additional communication in parallel computations, it is not recommended to implement a canonical switch for the central scheme. It appears that better accuracy can be obtained by improving unstructured discretizations rather than by switching to structured discretizations.

4 Improved matrix dissipation

This section presents a modified dissipation scheme developed for TAU HyperFlex. The default discretization scheme of TAU for the convective terms of the RANS equations is the central scheme with scalar artificial dissipation [3]. A matrix dissipation scheme can reduce the unnecessary artificial dissipation and improve solution accuracy. The common matrix dissipation scheme implemented in TAU has two important shortcomings. (1) It is less robust in particular on meshes with highly stretched hexahedra and tends to generate large wiggles in a vicinity of a shock. (2) It does not perform well in combination with the preconditioning techniques of TAU HyperFlex [7], [5]. A modified matrix dissipation scheme suggested in [7], [5] can be defined as

$$\boldsymbol{d}_{ij} = |\boldsymbol{A}_{ij}| \left\{ \frac{1}{2}\varepsilon_{ij}^{(2)} \left(\boldsymbol{W}_j - \boldsymbol{W}_i\right) - \varepsilon_{ij}^{(4)} \left(L_j(\boldsymbol{W}) - L_i(\boldsymbol{W})\right) \right\}, \tag{4.1}$$

where

$$\varepsilon_{ij}^{(2)} = min\left(k_2 max\left(\psi_i, \psi_j\right), 1\right), \varepsilon_{ij}^{(4)} = k_4\left(1 - \varepsilon_{ij}^{(2)}\right). \tag{4.2}$$

This formulation becomes nearly a first order Roe scheme near a shock and usually does not exhibit near-shock oscillations [3]. However, this formulation is known to degrade accuracy in some cases, specifically in computations performed for the Airbus HyperFlex workshop and the 5th Drag Prediction Workshop [9]. Fig. 4.1 shows a mesh convergence study for the NASA Common Research Model. Two results in this figure correspond to the matrix dissipation (4.1). The dark blue curve represents a parameter combination of the scheme leading to robust but less accurate results. The orange curve represents a parameter combination of the scheme leading to more accurate results but the corresponding solutions tend to diverge on finer meshes.

Two modifications are introduced in the matrix-dissipation formulation to combine advantages of the schemes (4.1) and (3.1) and to enhance the accuracy and robustness of flow solutions. The first modification is concerned with the cell stretching coefficient. A modification of 3.6,

$$\phi_{ij} = 1 + 2\frac{r_i r_j}{r_i + r_j}, r_i = \left(\frac{0.5 \sum\limits_{k \in N(i)} \lambda_{ik} - \lambda_{ij}}{\lambda_{ij}}\right)^{\omega}, \tag{4.3}$$

is introduced for the second-order dissipation. A previous study [2] recommended a cell stretching coefficient to balance accuracy and stability within a matrix dissipation scheme. The original TAU formulation of the cell stretching coefficient (3.6) serves the same goal and is well suited for unstructured meshes with relatively small aspect ratios in the boundary layer. For high aspect ratio cells

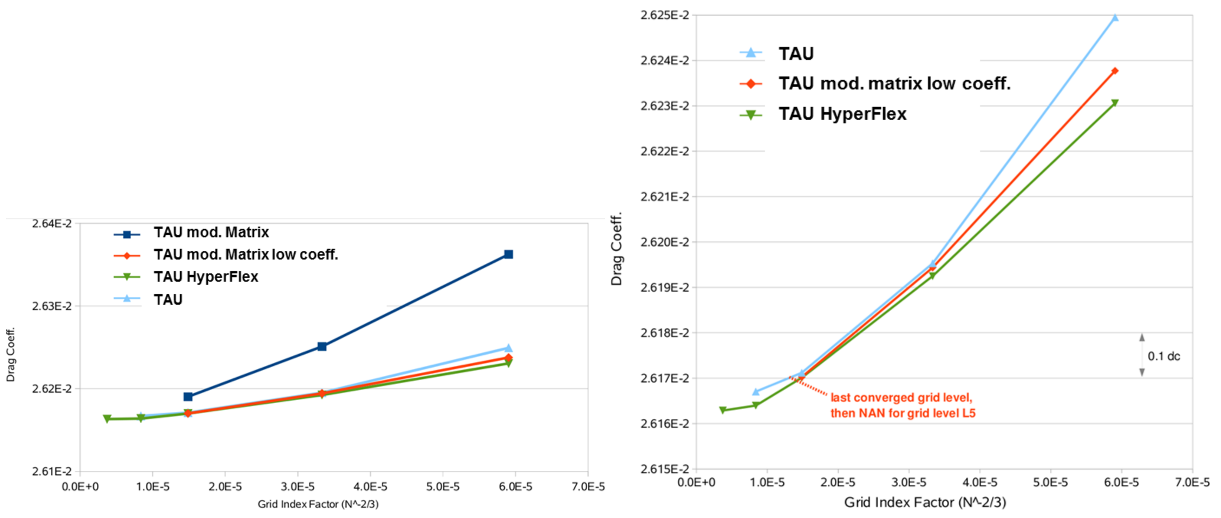

Figure 4.1: NASA Common Research Model, common grids of 5th Drag Prediction Workshop, mesh convergence study (left) in detail (right)

that are typical within structured boundary layers, this coefficient tends to zero (Fig. 4.2, red curve). Because this coefficient scales the whole artificial dissipation, insufficient dissipation introduced for high-aspect-ratio cells can trigger solution divergence.

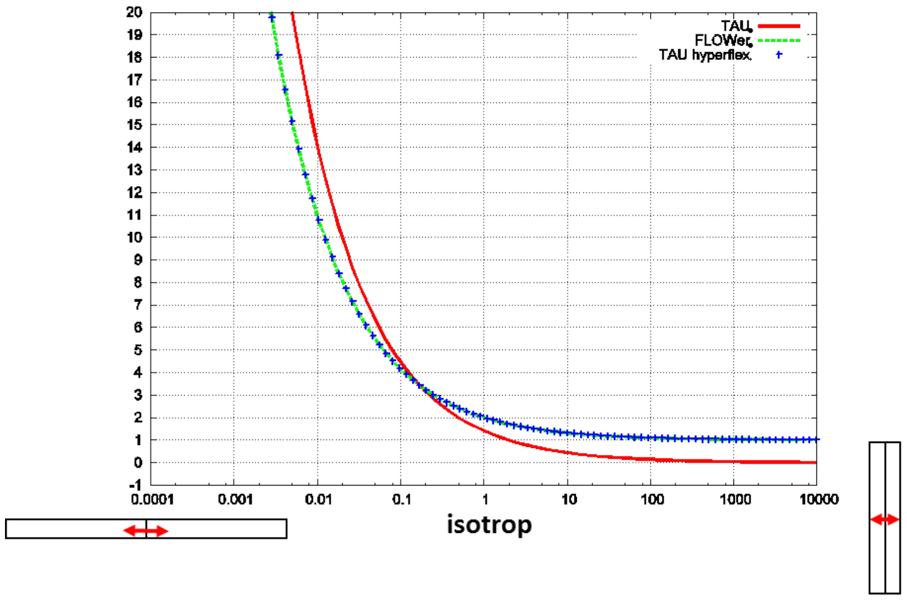

Figure 4.2: Versions of cell stretching coefficient

In contrast, the TAU HyperFlex cell stretching coefficient (4.3) tends to one (Fig. 4.2, blue dots) in the limit of high aspect ratio, similar to the structured version (3.6). It also corresponds to the structured version of the cell stretching coefficient (Fig. 4.2, green curve) on a 2-D mesh.

The second modification concerns with the scaling factors (3.5). Scaling of the dissipation with the number of neighbors is introduced again to achieve comparable accuracy levels similar to the standard TAU version. The HyperFlex cell stretching coefficient and neighbor-dependent scaling are used only for second-order terms to keep the scheme usable for the preconditioned Runge-Kutta scheme

$$\boldsymbol{d}_{ij} = |\boldsymbol{A}_{ij}| \left\{ \frac{1}{2}\varepsilon_{ij}^{(2)} (\boldsymbol{W}_j - \boldsymbol{W}_i) - \phi_{ij} sc_{ij}^{(4)} \varepsilon_{ij}^{(4)} (L_j(\boldsymbol{W}) - L_i(\boldsymbol{W})) \right\}. \tag{4.4}$$

The obtained formulation reduces the oscillations in the vicinity of shocks (similarly to the formulation 4.1) and does not lose accuracy. These properties are illustrated by the computations of a transonic flow around a NACA0012 profile (Fig. 4.3). The new formulations preserves robustness and improves the accuracy of the standard TAU dissipation scheme (Fig. 4.1).

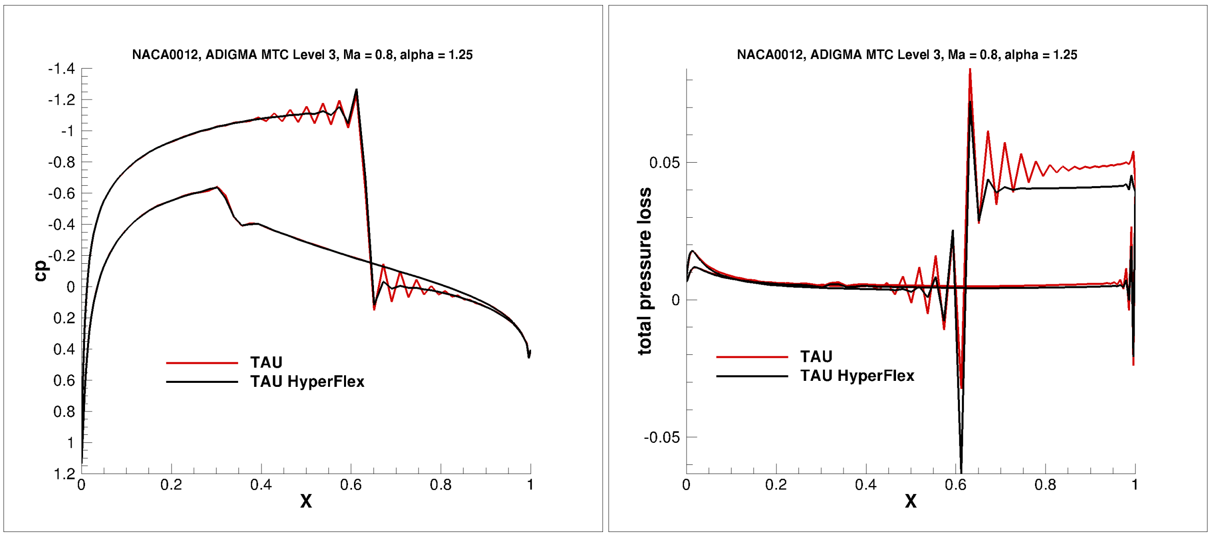

Figure 4.3: Comparison of results concerning a transonic Euler flow based on the standard and HyperFlex matrix dissipation of TAU

5 Conclusion

The report has demonstrated that implementation of a structured version of the central scheme with artificial dissipation in an unstructured code does not improve accuracy. Additionally, it has been shown that unstructured implementations of basic boundary conditions and viscous terms have similar accuracy compared to structured implementations. Experience gained in the process of implementing the enhanced edge-based data-structure and evaluating structured discretizations in TAU indicates that robustness and code design can suffer from a canonical switch. It has also been shown that improved accuracy can be obtained with some modifications of the existing unstructured matrix-dissipation scheme. Overall it is recommended to avoid redundancy associated with structured discretization segments of the central scheme in TAU. To improve accuracy and robustness, the presented improved matrix dissipation scheme is recommended for TAU HyperFlex.

The presented central scheme with matrix dissipation is available in the current TAU release 2012.2.0 by using the parameters 'Modified matrix dissipation (0/1): 1' and 'Version of cell stretching coefficient: HyperFlex', if matrix dissipation is chosen as central dissipation scheme. This parameter combination is going to become default for the matrix dissipation with the upcoming release.

Bibliography

[1] Blazek J. Computational Fluid Dynamics: Principles and Applications. *Elsevier Science, ISBN 0-08-043009-0* 2001.

[2] Brodersen O. Untersuchung einer Matrix-Dissipation in einem Zelleneckpunkt-Finite-Volumen-Schema zur Lösung der Navier-Stokes-Gleichungen. *DLR-Forschungsbericht 92-33, Diplomarbeit,* 1992.

[3] Institute of Aerodynamics and Flow Technology. Technical documentation of the DLR TAU-Code. *Institute of Aerodynamics and Flow Technology*

[4] Institute of Aerodynamics and Flow Technology. FLOWer - Installation and User Handbook. *Institute of Aerodynamics and Flow Technology*

[5] Langer S. Application of a line implicit method to fully coupled system of equations for turbulent flow problems. *submitted to Journal of Computer & Fluids* 2012;

[6] Langer S. Hierarchy of preconditioning techniques for the solution of the Navier-Stokes equations discretited by 2nd order unstructured finite volume methods. *submitted to Journal of Computer & Fluids* 2012;

[7] Langer S. Investigation and application of point implicit Runge-Kutta methods to inviscid flow problems. *ECCOMAS, J, Eberhardsteiner et.al. (eds.), Austria* 2012; **69**(2):332–352.

[8] Schwöppe A, Heinrich R. Flow Solver Algorithms and Data Structures for TAUijk. *DLR-Interner Bericht, DLR-IB 124-2006/006,* 2006.

[9] 5th AIAA CFD Drag Prediction Workshop, `http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/`.

Robust schemes based on improved preconditioning technqiues

1 Introduction

The development of accurate, reliable and efficient numerical algorithms to approximate a solution of the Navier-Stokes equations is an enduring challenge in the field of computational fluid dynamics (CFD).
For example, the simulation of an unsteady full high-lift configuration of an airplane requires a robust and efficient solver. The well-known and established multigrid techniques for unstructured meshes for solving large-scale high-Reynolds number viscous flows, which are usually based on agglomeration techniques with explicit Runge-Kutta methods as smoothers (see [13, 17]), often show a slowdown and significant deterioration of the observed convergence rate. Moreover, often the reliability of these kinds of flow solvers is a severe problem and substantial user experience is required to choose an appropriate set of parameters not only to overcome the start up problem of the algorithm but also to generate a converged solution.
Important reasons for the observed deterioration of the convergence rates and the poor reliability of a flow solver are (see [20] for a detailed discussion)

a) stiffness introduced by large anisotropic mesh cells required for high-Reynolds number viscous flows near the solid boundary,

b) stiffness introduced by the flow equations themselves, in particular introduced by the additional equations arising from the turbulence model.

In general, solution algorithms for the Navier-Stokes equations are based on approximations to the full Jacobian of the system of equations. For example, the traditional explicit Runge-Kutta methods can be viewed as a solution algorithm, where all block off-diagonal terms of the full Jacobian are deleted and the remaining block diagonal terms are approximated by their largest eigenvalue. To improve the reliability and efficiency of solution algorithms better suited approximations to the full Jacobian need to be constructed. These approximations are included as preconditioners into the basic Runge-Kutta scheme. Such a procedure increases the degree of implicitness of the overall scheme (see e.g. [27, 26, 3]).
However, for a reliable and efficient algorithm an appropriate measure for the simplification of the full Jacobian is required. For example, a full second order Jacobian requires a significant increase in memory requirements, and it augments substantially the computational effort per iteration to solve the linear system. Instead, one can consider only terms in the Jacobian which require next neighbor information (1.st order Jacobian). This reduces the memory requirement significantly. Implementing this technique as a smoother into an agglomerated multigrid, the resulting large scale linear systems do not need to be solved exactly. Usually only a few Gauss-Seidel or Krylov subspace iterations are enough for an efficient and reliable multigrid method (see e.g [27, 26]).

In this article we are going to contribute to a so-called line implicit ansatz (see [16, 6]). This ansatz is guided by the following observations:

Stiffness is introduced into the discrete set of equations by anisotropic cells in the mesh. Therefore, a further simplification of the Jacobian can be considered by extracting relevant mesh information.
To this end regions characterized by large anisotropies in the mesh need to be identified in a preprocessing step. This information is given by lines along the anisotropic cells. In isotropic parts of the mesh a line degenerates canonically into a point. Then, the 1.st order Jacobian is only constructed along the lines. The resulting linear systems are small-scale block tridiagonal and can be easily and efficiently solved by direct solution algorithms such as block LU-decomposition. When a line degenerates to a point only a small-scale block system needs to be inverted.
The line implicit ansatz has two advantages when compared with the ansatz based on the 1.st order Jacobian:

a) The required memory for the line implicit ansatz is significantly less than for the 1.st order Jacobian.

b) The resulting linear systems can be efficiently and exactly solved.

2 Governing equations

To describe turbulent flow effects we consider for an open domain $\Omega \subset \mathbb{R}^3$ the Reynolds-averaged Navier-Stokes (RANS) equations fully coupled with a one-equation Spalart-Allmaras (SA) turbulence model [23] for a three-dimensional flow

$$\mathbf{u}(\mathbf{x},t) = (u_1(\mathbf{x},t), u_2(\mathbf{x},t), u_3(\mathbf{x},t)), \qquad (\mathbf{x},t) \in \Omega \times (0,\infty)$$

in conservative variables

$$\mathbf{W} := (\rho, \rho u_1, \rho u_2, \rho u_3, \rho E, \rho\tilde{\nu}) \tag{1}$$

written as

$$\frac{d}{dt}\int_\Omega \mathbf{W}\,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} (\mathbf{f}_c \cdot \mathbf{n} - \mathbf{f}_v \cdot \mathbf{n})\,\mathrm{d}s(y) = \int_\Omega \mathbf{Q}\,\mathrm{d}\mathbf{x}. \tag{2}$$

Here the convective and viscous terms are given by

$$\mathbf{f}_c \cdot \mathbf{n} = \begin{pmatrix} \rho V \\ \rho u_1 V + p n_1 \\ \rho u_2 V + p n_2 \\ \rho u_3 V + p n_3 \\ \rho H V \\ \rho V \tilde{\nu} \end{pmatrix}, \qquad \mathbf{f}_v \cdot \mathbf{n} = \begin{pmatrix} 0 \\ n_1\tau_{11} + n_2\tau_{12} + n_3\tau_{13} \\ n_1\tau_{21} + n_2\tau_{22} + n_3\tau_{23} \\ n_1\tau_{31} + n_2\tau_{32} + n_3\tau_{33} \\ n_1\theta_1 + n_2\theta_2 + n_3\theta_3 \\ n_1\tau_1^{\mathrm{Tu}} + n_2\tau_2^{\mathrm{Tu}} + n_3\tau_3^{\mathrm{Tu}} \end{pmatrix}$$

where $V = \langle \mathbf{u}, \mathbf{n} \rangle$ describes the normal velocity and

$$H(\mathbf{x},t) = E(\mathbf{x},t) + p(\mathbf{x},t)/\rho(\mathbf{x},t) \tag{3}$$

the total Enthalpy. The pressure is defined by the state equation

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

where E is the specific total energy, and γ is the gas dependent ratio of specific heats, which is given by 1.4 for air. The viscous terms are defined by

$$\tau_{ii} := 2\mu_{\mathrm{eff}}\frac{\partial u_i}{\partial x_i} + \lambda_l \operatorname{div}\mathbf{u}, \qquad i = 1,2,3, \tag{4a}$$

$$\tau_{ij} := \mu_{\mathrm{eff}}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \qquad \tau_{ij} := \tau_{ji}, \qquad 1 \le i < j \le 3 \tag{4b}$$

$$\tau_i^{\mathrm{Tu}} := \frac{\mu_{\mathrm{eff}}}{\sigma}\frac{\partial\tilde{\nu}}{\partial x_i}, \qquad i = 1,2,3, \tag{4c}$$

and

$$\begin{aligned} \theta_j &:= \left(\sum_{k=1}^{3} u_k \tau_{jk}\right) + q_j, \qquad j = 1,2,3, \\ \mathbf{q} &:= \kappa_{\mathrm{eff}} \operatorname{grad} T, \qquad \mathbf{q} = (q_1, q_2, q_3). \end{aligned} \tag{5}$$

In formulas (4a) – (4c)

$$\mu_{\text{eff}} := \mu_l + \mu_t, \qquad \kappa_{\text{eff}} := \kappa_l + \kappa_t. \tag{6}$$

denote the effective viscosity and effective conductivity. Sutherland's law gives us,

$$\mu_l := \mu_{l,\infty} \left(\frac{T}{T_\infty}\right)^{3/2} \frac{T_\infty + \bar{T}}{T + \bar{T}}, \qquad \kappa_l := \frac{c_p \mu_l}{Pr_l} \qquad \text{and} \qquad c_p := \Re \frac{\gamma}{\gamma - 1}, \tag{7}$$

whereby $\bar{T} := 110.4\text{K}$ is Sutherland's constant, $\Re$ is the universal gas constant and the laminar Prandtl number is given by $Pr_l := 0.72$.

Following [23] the turbulent conductivity κ_t and the eddy viscosity μ_t and are defined by

$$\kappa_t := \frac{c_p \mu_t}{Pr_t}, \quad \mu_t := \rho \tilde{\nu} f_{\text{v}_1}, \quad f_{\text{v}_1} := \frac{\chi^3}{\chi^3 + c_{\text{v}_1}^3}, \quad \chi := \frac{\tilde{\nu}}{\nu_l}, \quad \nu_l := \frac{\mu_l}{\rho}. \tag{8}$$

The turbulent Prandtl number is given by $Pr_t := 0.92$. Finally, the missing source terms occurring on the right hand side of (2) are given by

$$\begin{aligned}
\mathbf{Q}(\mathbf{x}, t) &:= (0, 0, 0, 0, 0, \mathbf{Q}_6(\mathbf{x}, t)), \\
\mathbf{Q}_6(\mathbf{x}, t) &:= \mathbf{Pr}(\mathbf{x}, t) - \mathbf{De}(\mathbf{x}, t) - \mathbf{Di}(\mathbf{x}, t), \\
\mathbf{Pr}(\mathbf{x}, t) &:= c_{\text{b}_1} \left(1 - f_{\text{t}_2}(\mathbf{x}, t)\right) \tilde{S}(\mathbf{x}, t) \tilde{\nu}(\mathbf{x}, t), \\
\mathbf{De}(\mathbf{x}, t) &:= \left(c_{\text{w}_1} f_{\text{w}}(\mathbf{x}, t) - \frac{c_{\text{b}_1}}{\kappa^2} f_{\text{t}_2}(\mathbf{x}, t)\right) \left(\frac{\tilde{\nu}(\mathbf{x}, t)}{d}\right)^2, \\
\mathbf{Di}(\mathbf{x}, t) &:= \frac{c_{\text{b}_2}}{\sigma} \sum_{j=1}^{3} \left(\frac{\partial \tilde{\nu}(\mathbf{x}, t)}{\partial x_j}\right)^2,
\end{aligned}$$

where we denote by **Pr** the production term, by **De** the destruction term and by **Di** the diffusion term. Terms required for transition given in [23] are neglected. Further terms are given by

$$\begin{aligned}
\tilde{S}(\mathbf{x}, t) &:= \|\text{curl } \mathbf{u}(\mathbf{x}, t)\|_2 + \frac{\tilde{\nu}(\mathbf{x}, t) f_{\text{v}_2}(\mathbf{x}, t)}{\kappa^2 d^2(\mathbf{x})}, \qquad f_{\text{t}_2}(\mathbf{x}, t) := c_{\text{t}_3} \exp\left(-c_{\text{t}_4} \chi(\mathbf{x}, \text{t})^2\right) \\
f_{\text{v}_2}(\mathbf{x}, t) &:= 1 - \frac{\chi(\mathbf{x}, t)}{1 + \chi(\mathbf{x}, t) f_{\text{v}_1}(\mathbf{x}, t)}, \qquad f_{\text{w}}(\mathbf{x}, t) := g(\mathbf{x}, t) \left[\frac{1 + c_{\text{w}_3}^6}{(g(\mathbf{x}, t))^6 + c_{\text{w}_3}^6}\right]^{1/6}, \\
g(\mathbf{x}, t) &:= r(\mathbf{x}, t) + c_{\text{w}_2} \left((r(\mathbf{x}, t))^6 - r(\mathbf{x}, t)\right), \qquad r(\mathbf{x}, t) := \frac{\tilde{\nu}(\mathbf{x}, t)}{\kappa^2 d^2(\mathbf{x}) \tilde{S}(\mathbf{x}, t)}.
\end{aligned}$$

Here $d(\mathbf{x})$ describes the distance to the wall. The constants of the model are

$$\begin{aligned}
c_{\text{b}_1} &:= 0.1355, \quad c_{\text{b}_2} := 0.622, \quad \sigma := \frac{2}{3}, \quad \kappa := 0.41, \\
c_{\text{w}_1} &:= \frac{c_{\text{b}_1}}{\kappa} + \frac{1 + c_{\text{b}_2}}{\sigma}, \quad c_{\text{w}_2} := 0.3, \quad c_{\text{w}_3} := 2, \\
c_{\text{t}_3} &:= 1.2, \quad c_{\text{t}_4} := 0.5, \quad c_{\text{v}_1} := 7.1.
\end{aligned}$$

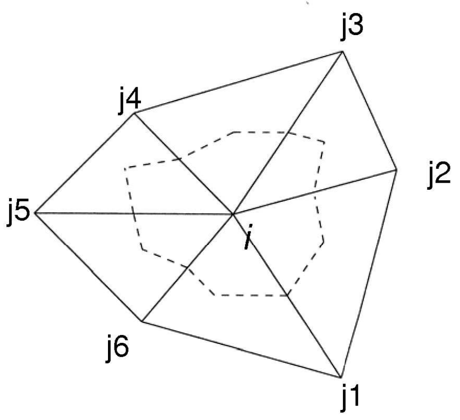

Figure 1: Example of a dual cell; the solid lines represent the primary grid, the dashed line the corresponding dual cell

3 Discretization

The CFD solver used is the TAU code developed at the Deutsches Zentrum für Luft- und Raumfahrt e.V. TAU is based on a finite volume formulation where a median dual grid forms the control volumes with the unknowns at vertices of the primary grid. The dual grid is created in a preprocessing step. Figure 1 shows an example of a dual cell. An agglomeration multigrid algorithm is used to approximate a steady state solution of the governing equations. The description of the multistage preconditioned Runge-Kutta scheme, which is used as a multigrid smoother, is topic of this report.
Throughout this report the set of points in the dual grid where the unknowns are located is denoted by $G := \{1, \dots, N\}$ and the unknowns are given by $\mathbf{W}_1, \dots, \mathbf{W}_N$. The neighbors of some point $i \in G$ are described by $\mathcal{N}(i)$, and the corresponding face between the points i and j is denoted by ij.

3.1 The Inviscid Contribution

For an inner volume Ω_i the inviscid part $\mathbf{f}_c \cdot \mathbf{n}$ of (2) is approximated by

$$\begin{aligned}\int_{\partial\Omega_i} \mathbf{f}_c \cdot \mathbf{n} \, \mathrm{d}s(y) \;\approx\; & \sum_{j \in \mathcal{N}(i)} \frac{1}{2} \left\{ \left(\mathbf{f}_c \cdot \mathbf{n}_{ij}\right)\left(\mathbf{W}_i\right) + \left(\mathbf{f}_c \cdot \mathbf{n}_{ij}\right)\left(\mathbf{W}_j\right) \right\} \\ & - \frac{1}{2} \left|\mathbf{A}_{ij}\right| \mathbf{D}\left(\mathbf{W}_i, \mathbf{W}_{j, j \in \mathcal{N}(i)}, \mathbf{W}_{k, k \in \mathcal{N}(j)}\right). \qquad (9)\end{aligned}$$

As a basic definition for the dissipative term $\mathbf{D}$ we use

$$\begin{aligned} & \mathbf{D}^{P_t,L_z,s(\mathbf{W}),\xi}\left(\mathbf{W}_i, \mathbf{W}_{j,j\in\mathcal{N}(i)}, \mathbf{W}_{k,k\in\mathcal{N}(j)}\right) \\ := & \left\{\psi_{ij}^{(t)}\left(\mathbf{W}_j - \mathbf{W}_i\right) - \xi s_{ij}(\mathbf{W})\left(1-\psi_{ij}^{(t)}\right)\left(L_j^{(z)}(\mathbf{W}) - L_i^{(z)}(\mathbf{W})\right)\right\}, \end{aligned} \tag{10}$$

where P_t denotes the dependency on some pressure switch to weight first and second order terms, L_z represents some construction of second order terms and s_{ij} and ξ are some factors. A more detailed construction of these terms is topic of Sections 3.3–3.5.

3.2 Construction of Roe matrix

Going into the details of the construction of the dissipation we start with the computation of the Roe matrix $|\mathbf{A}_{ij}|$ arising in (9). To implement $|\mathbf{A}_{ij}|$ for a fully coupled solver an eigendecomposition of the derivative $\frac{\partial(\mathbf{f}_c\cdot\mathbf{n})}{\partial\mathbf{W}}$ with respect to the six conservative variables (1) is required. Following line by line the analysis in [10] the derivative can be assembled as

$$\frac{\partial(\mathbf{f}_c\cdot\mathbf{n})(\mathbf{W})}{\partial\mathbf{W}} = V\mathbf{I} + \mathbf{a}_1(\mathbf{b}_1)^T + \mathbf{a}_2(\mathbf{b}_2)^T \tag{11}$$

where

$$\begin{aligned} \mathbf{a}_1 &:= (1, u_1, u_2, u_3, H, \tilde{\nu})^T, \\ \mathbf{a}_2 &:= (0, n_1, n_2, n_3, V, 0)^T, \\ \mathbf{b}_1 &:= (-V, n_1, n_2, n_3, 0, 0)^T, \\ \mathbf{b}_2 &:= (\gamma-1)\left(\frac{\|\mathbf{u}\|_2^2}{2}, -u_1, -u_2, -u_3, 1, 0\right)^T. \end{aligned}$$

It can be shown that the vectors

$$\begin{aligned} \mathbf{y}_1 &:= \left(1, u_1, u_2, u_3, \frac{\|\mathbf{u}\|_2^2}{2}, 0\right)^T, \\ \mathbf{y}_2 &:= (0, 0, n_3, -n_2, u_2n_3 - u_3n_2, 0)^T, \\ \mathbf{y}_3 &:= (0, -n_3, 0, n_1, u_3n_1 - u_1n_3, 0)^T, \\ \mathbf{y}_4 &:= (0, n_2, -n_1, 0, u_1n_2 - u_2n_1, 0)^T \end{aligned}$$

satisfy the orthogonality relations $\mathbf{b}_1^T\mathbf{x} = \mathbf{b}_2^T\mathbf{x} = 0$. However, the vectors $\mathbf{y}_2, \mathbf{y}_3$ and $\mathbf{y_4}$ are linear dependent since $n_1\mathbf{y}_2 + n_2\mathbf{y}_3 + n_3\mathbf{y}_4 = 0$. Therefore, three linear independent eigenvectors may be obtained as

$$\begin{aligned} \mathbf{g}_1 &:= n_1\mathbf{y}_1 + a\mathbf{y}_2, \\ \mathbf{g}_2 &:= n_2\mathbf{y}_1 + a\mathbf{y}_3, \\ \mathbf{g}_3 &:= n_3\mathbf{y}_1 + a\mathbf{y}_4, \end{aligned}$$

where $a := \sqrt{\gamma p/\rho}$ denotes the speed of sound. Obviously, $\mathbf{g}_6 := (0,0,0,0,0,1)^T$ is another eigenvector with corresponding eigenvalue V. To identify the remaining eigenvectors we use the relations

$$\mathbf{b}_1^T\mathbf{a}_1 = \mathbf{b}_2^T\mathbf{a}_2 = 0, \quad \mathbf{b}_1^T\mathbf{a}_2 = A^2 \quad \text{and} \quad \mathbf{b}_2^T\mathbf{a}_1 = a^2,$$

where $A := \sqrt{n_1^2 + n_2^2 + n_3^2}$ denotes the surface area. Then we compute using (11)

$$\begin{aligned}
\frac{\partial(\mathbf{f}_c \cdot \mathbf{n})(\mathbf{W})}{\partial \mathbf{W}}(A\mathbf{a}_1 + a\mathbf{a}_2) &= (V + aA)(A\mathbf{a}_1 + a\mathbf{a}_2), \\
\frac{\partial(\mathbf{f}_c \cdot \mathbf{n})(\mathbf{W})}{\partial \mathbf{W}}(A\mathbf{a}_1 - a\mathbf{a}_2) &= (V - aA)(A\mathbf{a}_1 - a\mathbf{a}_2).
\end{aligned}$$

Hence, we found the two remaining eigenvalues $V \pm aA$ with corresponding eigenvectors $\mathbf{g}_{4/5} := A\mathbf{a}_1 \pm a\mathbf{a}_2$.

Theorem 3.1 *The derivative matrix of the convective flux $\mathbf{f}_c \cdot \mathbf{n}$ for a one-equation turbulence model given by (11) has the following set of eigenpairs*

$$\{(V, \mathbf{g}_1), (V, \mathbf{g}_2), (V, \mathbf{g}_3), (V + aA, \mathbf{g}_4), (V - aA, \mathbf{g}_5), (V, \mathbf{g}_6)\}.$$

Proof: The assertion follows by the computations above.

$\square$

Theorem 3.2 *Let $\mathbf{G} := (\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_4, \mathbf{g}_5, \mathbf{g}_6)$ denote the matrix corresponding to the eigenvectors of $\frac{\partial(\mathbf{f}_c \cdot \mathbf{n})(\mathbf{W})}{\partial \mathbf{W}}$. We define the vectors*

$$\begin{aligned}
\mathbf{p}_1 &:= \frac{\gamma - 1}{a^2}(H - \|\mathbf{u}\|_2^2, u_1, u_2, u_3, -1, 0)^T, && (12a) \\
\mathbf{p}_2 &:= (u_2 n_3 - u_3 n_2, 0, -n_3, n_2, 0, 0)^T, && (12b) \\
\mathbf{p}_3 &:= (u_3 n_1 - u_1 n_3, n_3, 0, -n_1, 0, 0)^T, && (12c) \\
\mathbf{p}_4 &:= (u_1 n_2 - u_2 n_1, -n_2, n_1, 0, 0, 0)^T && (12d)
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{q}_1 &:= \frac{1}{A^2}\left(n_1\mathbf{p}_1^T - \frac{1}{a}\mathbf{p}_2^T\right) && (13a) \\
\mathbf{q}_2 &:= \frac{1}{A^2}\left(n_2\mathbf{p}_1^T - \frac{1}{a}\mathbf{p}_3^T\right) && (13b) \\
\mathbf{q}_3 &:= \frac{1}{A^2}\left(n_3\mathbf{p}_1^T - \frac{1}{a}\mathbf{p}_4^T\right) && (13c) \\
\mathbf{q}_4 &:= \frac{1}{2a^2A}\left(\mathbf{b}_2^T + \frac{a}{A}\mathbf{b}_1^T\right) && (13d) \\
\mathbf{q}_5 &:= \frac{1}{2a^2A}\left(\mathbf{b}_2^T - \frac{a}{A}\mathbf{b}_1^T\right) && (13e) \\
\mathbf{q}_6 &:= \frac{\tilde{\nu}(\gamma - 1)}{a^2}\left(-\frac{\|\mathbf{u}\|_2^2}{2}, u_1, u_2, u_3, -1, \frac{a^2}{\tilde{\nu}(\gamma - 1)}\right) && (13f)
\end{aligned}$$

Then the inverse of $\mathbf{G}$ *is given by*

$$\mathbf{J} := \begin{pmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \\ \mathbf{q}_4 \\ \mathbf{q}_5 \\ \mathbf{q}_6 \end{pmatrix}, \qquad i.e. \qquad (\mathbf{G})^{-1} = \mathbf{J}.$$

Proof: The theorem follows by straightforward computations (see [10, Theorem 3] for more details.

□

Theorems 3.1 and 3.2 allow a straightforward generalization of the Matrix Dissipation operator to turbulent flows via

$$\left| \frac{\partial(\mathbf{f}_c \cdot \mathbf{n})(\mathbf{W})}{\partial \mathbf{W}} \right| := \sum_{j=1}^{6} |\alpha_j| \mathbf{g}_j \mathbf{q}_j, \tag{14}$$

where the scalars α_j are given by the eigenvalues

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_6 = V, \qquad \alpha_4 = V + aA, \qquad \alpha_5 = V - aA.$$

Formula (14) is evaluated on a face $k\ell$ using Roe-averaged variables [21] and some entropy fix to avoid zero dissipation. The exact proceeding can be found in [10, Section 3.4]. Additionally the entropy fix needs to be incorporated into the additional turbulent equation, which is straightforward.

3.3 General dissipative operators

To derive a dissipative scheme for convective dominated flows we consider for a domain $U \subset \Omega \times [0,T)$, $T > 0$, $\Omega \subset \mathbb{R}^m$ the heat equation

$$\begin{aligned} \frac{\partial}{\partial t} u(x,t) &= \Delta u(x,t), \qquad t \geq 0, \quad x \in \Omega, & \text{(15a)} \\ u(x,0) &= f(x), \qquad x \in \partial\Omega. & \text{(15b)} \end{aligned}$$

Integration of (15a) and the application of Green's theorem yields

$$\int_\Omega \frac{\partial}{\partial t} u(x,t)\,\mathrm{d}x = \int_\Omega \Delta u(x,t)\,\mathrm{d}x = \int_{\partial\Omega} \frac{\partial u(x,t)}{\partial n}\,\mathrm{d}s(y). \tag{16}$$

Here n denotes the outward facing normalized normal vector on $\partial\Omega$. Considering a dual grid approach a possible discretization of (16) may read

$$\mathrm{vol}(\Omega_i) \left(\frac{d\mathbf{u}_i(t)}{dt} \right) = \sum_{j \in \mathcal{N}(i)} \mathbf{u}_{ij}(t) \mathrm{vol}(\Omega_{ij}) \tag{17}$$

where the term $\mathbf{u}_{ij}(t)$ is approximated by

$$\begin{aligned}\frac{\partial u(x,t)}{\partial n} &= \langle \text{grad } u(x,t), n\rangle \approx \left\langle \begin{pmatrix} \frac{(\mathbf{u}_j(t)-\mathbf{u}_i(t))n_1}{d(\mathbf{p}_j,\mathbf{p}_i)} \\ \frac{(\mathbf{u}_j(t)-\mathbf{u}_i(t))n_2}{d(\mathbf{p}_j,\mathbf{p}_i)} \\ \frac{(\mathbf{u}_j(t)-\mathbf{u}_i(t))n_3}{d(\mathbf{p}_j,\mathbf{p}_i)} \end{pmatrix}, n \right\rangle \\ &= \frac{\mathbf{u}_j(t)-\mathbf{u}_i(t)}{d(\mathbf{p}_j,\mathbf{p}_i)},\end{aligned}$$

$\text{vol}(\Omega_{ij})$ denotes the volume of the face connecting the elements around the node $\mathbf{p}_i$ and $\mathbf{p}_j$ and

$$d(\mathbf{p}_j,\mathbf{p}_i) := \|\mathbf{p}_j - \mathbf{p}_i\|_2.$$

Neglecting boundaries (17) can be written in matrix form

$$\mathbf{M}\frac{d\mathbf{u}(t)}{dt} = \mathbf{A}\mathbf{u}(t), \qquad \mathbf{M} := \text{diag}(\text{vol}(\Omega_i)). \tag{18}$$

To understand the shape of the matrix $\mathbf{A} = (a_{ij})$ we rearrange

$$\begin{aligned}\sum_{j\in\mathcal{N}(i)} \mathbf{u}_{ij}(t)\text{vol}(\Omega_{ij}) &= \sum_{j\in\mathcal{N}(i)} \left(\frac{\mathbf{u}_j(t)-\mathbf{u}_i(t)}{d(\mathbf{p}_j,\mathbf{p}_i)}\right)\text{vol}(\Omega_{ij}) \\ &= \sum_{j\in\mathcal{N}(i)} \frac{\mathbf{u}_j(t)}{d(\mathbf{p}_j,\mathbf{p}_i)}\text{vol}(\Omega_{ij}) - \mathbf{u}_i(t)\sum_{j\in\mathcal{N}(i)}\frac{1}{d(\mathbf{p}_j,\mathbf{p}_i)}\text{vol}(\Omega_{ij}).\end{aligned}$$

From this we conclude that

$$a_{ii} = -\sum_{j\in\mathcal{N}(i)} \frac{\text{vol}(\Omega_{ij})}{d(\mathbf{p}_j,\mathbf{p}_i)} \tag{19a}$$

$$a_{ij} = \frac{\text{vol}(\Omega_{ij})}{d(\mathbf{p}_j,\mathbf{p}_i)}, \qquad j\in\mathcal{N}(i), \qquad a_{ij}=0, \quad i\neq j, j\notin\mathcal{N}(i). \tag{19b}$$

To show that this discretization is stable one needs to show that it satisfies some energy estimate. The key property for this is that all eigenvalues of $\mathbf{A}$ are nonpositive. For further details on this topic we refer to [25, 24]. Here we only prove the nonpositivity of $\mathbf{A}$.

Theorem 3.3 *The matrix* $\mathbf{A}$ *defined through (19a) and (19b) is symmetric and negative semidefinite, that is*

$$\mathbf{v}^T\mathbf{A}\mathbf{v} \leq 0 \qquad \textit{for all} \quad \mathbf{v}\in\mathbb{R}^N.$$

Proof: Since $\mathrm{vol}(\Omega_{ij}) = \mathrm{vol}(\Omega_{ji})$ and $d(\mathbf{p}_j, \mathbf{p}_i) = d(\mathbf{p}_i, \mathbf{p}_j)$ we conclude that $a_{ij} = a_{ji}$ for all i, j. Therefore $\mathbf{A}$ is symmetric. From (19b) and (19a) it is obvious that $a_{ij} > 0$, $j \in \mathcal{N}(i)$ and $a_{ii} < 0$. Moreover, for the off diagonal elements of the ith row of $\mathbf{A}$ we compute

$$\sum_{j\in\mathbb{N}, j\neq i} a_{ij} = \sum_{j\in\mathcal{N}(i)} a_{ij} = -a_{ii}.$$

Hence, the Gerschgorin circles of $\mathbf{A}$ are given by

$$G_i = \left\{ z \in \mathbb{R} : |z - a_{ii}| \leq \sum_{j\in\mathcal{N}(i)} |a_{ij}| \right\} = \{z \in \mathbb{R} : |z - a_{ii}| \leq |a_{ii}|\} \subset (-\infty, 0],$$

because $a_{ii} \leq 0$. From Gerschgorin's theorem it follows that all eigenvalues λ_i of $\mathbf{A}$ satisfy $\lambda_i \leq 0$.

□

Now we incorporate a boundary term into (17). Denoting a possible boundary point by $\mathbf{p}_b$ a possible discretization for a boundary point may read

$$\mathrm{vol}(\Omega_b)\left(\frac{d\mathbf{u}_b(t)}{dt}\right) = \sum_{j\in\mathcal{N}(b)} \mathbf{u}_{bj}(t)\mathrm{vol}(\Omega_{bj}) + \mathbf{u}_{b,N}(t)\mathrm{vol}(\Omega_b). \tag{20}$$

Here $\mathbf{u}_{b,N}(t)$ denotes some so far unknown construction of the normal derivative for a boundary point b. Before constructing this term we extend the matrix $\mathbf{A}$ to boundary points. To do so we rewrite (20) as

$$\begin{aligned}
&\sum_{j\in\mathcal{N}(b)} \mathbf{u}_{bj}(t)\mathrm{vol}(\Omega_{bj}) + \mathbf{u}_{b,N}(t)\mathrm{vol}(\Omega_b) \\
= &\sum_{j\in\mathcal{N}(b)} \frac{\mathbf{u}_j(t)}{d(\mathbf{p}_j, \mathbf{p}_b)}\mathrm{vol}(\Omega_{bj}) - \mathbf{u}_b(t) \sum_{j\in\mathcal{N}(b)} \frac{\mathrm{vol}(\Omega_{bj})}{d(\mathbf{p}_j, \mathbf{p}_b)} + \mathbf{u}_{b,N}(t)\mathrm{vol}(\Omega_b) \\
= &\sum_{j\in\mathcal{N}(b)} a_{bj}\mathbf{u}_j(t) + a_{bb}\mathbf{u}_b(t) + \mathbf{u}_{b,N}(t)\mathrm{vol}(\Omega_b)
\end{aligned}$$

where

$$\begin{aligned}
a_{bb} &= -\sum_{j\in\mathcal{N}(b)} \frac{\mathrm{vol}(\Omega_{bj})}{d(\mathbf{p}_j, \mathbf{p}_b)} \\
a_{bj} &= \frac{\mathrm{vol}(\Omega_{bj})}{d(\mathbf{p}_j, \mathbf{p}_b)}, \qquad j \in \mathcal{N}(b), \quad a_{bj} = 0, \quad b \neq j, j \notin \mathcal{N}(b).
\end{aligned}$$

Note that the matrix $\mathbf{A}$ with this extension is still symmetric and negative semidefinite. Symmetry is a consequence of $a_{bj} = a_{jn}$. As above for a boundary off diagonal row sum we compute

$$\sum_{j\in\mathbb{N}, j\neq b} a_{bj} = \sum_{j\in\mathcal{N}(b)} a_{bj} = -a_{bb}.$$

Therefore the additional Gerschgorin circles are given by

$$G_b = \{z \in \mathbb{R} : |z - a_{bb}| \leq |a_{bb}\} \subset (-\infty, 0],$$

since $a_{bb} < 0$. Hence, all eigenvalues of the extension of $\mathbf{A}$ to the boundary point are nonpositive.

To construct the term $\mathbf{u}_{b,N}(t)$ we introduce the following sets:

Definition 3.4 *Let $\mathbf{p}_b$ denote a boundary point. We define*

$$\begin{aligned} B_b &:= \{\mathbf{p}_i : i \in \mathcal{N}(b) \cap B\}, \\ M_b &:= \mathcal{N}(b) \setminus B_b, \\ I_b &:= M_b \cup \{b\} \\ L_{bi} &:= \mathcal{N}(i) \setminus I_b. \end{aligned}$$

Lemma 3.5 *For a point $\mathbf{p}_b$, $b \in B$ we have*

$$\sum_{i \in I_b} \sum_{j \in \mathcal{N}(i) \cap I_b} \frac{\mathbf{u}_j(t) - \mathbf{u}_i(t)}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}) = 0. \tag{21}$$

Proof: By the Definition 3.4 we have negelecting time level t

$$\begin{aligned} &\sum_{i \in I_b} \sum_{j \in \mathcal{N}(i) \cap I_b} \frac{\mathbf{u}_j - \mathbf{u}_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}) \\ = &\sum_{i \in M_b} \sum_{j \in \mathcal{N}(i) \cap I_b} \frac{\mathbf{u}_j - \mathbf{u}_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}) + \sum_{i \in \{b\}} \sum_{j \in \mathcal{N}(i) \cap I_b} \frac{\mathbf{u}_j - \mathbf{u}_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}) \\ = &\sum_{i \in M_b} \left(\sum_{j \in \mathcal{N}(i) \cap M_b} \frac{\mathbf{u}_j - \mathbf{u}_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}) + \sum_{j \in \mathcal{N}(i) \cap \{b\}} \frac{\mathbf{u}_j - \mathbf{u}_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}) \right) \\ &+ \sum_{j \in \mathcal{N}(b) \cap M_b} \frac{\mathbf{u}_j - \mathbf{u}_b}{d(\mathbf{p}_j, \mathbf{p}_b)} \mathrm{vol}(\Omega_{bj}) + \sum_{j \in \mathcal{N}(b) \cap \{b\}} \frac{\mathbf{u}_j - \mathbf{u}_b}{d(\mathbf{p}_j, \mathbf{p}_b)} \mathrm{vol}(\Omega_{bj}). \end{aligned}$$

Since $\mathcal{N}(b) \cap \{b\} = \emptyset$ and $\mathcal{N}(b) \cap M_b = M_b$ we have

$$\begin{aligned} \sum_{j \in \mathcal{N}(b) \cap \{b\}} \frac{\mathbf{u}_j - \mathbf{u}_b}{d(\mathbf{p}_j, \mathbf{p}_b)} \mathrm{vol}(\Omega_{bj}) &= 0 \\ \sum_{j \in \mathcal{N}(b) \cap M_b} \frac{\mathbf{u}_j - \mathbf{u}_b}{d(\mathbf{p}_j, \mathbf{p}_b)} \mathrm{vol}(\Omega_{bj}) &= \sum_{j \in M_b} \frac{\mathbf{u}_j - \mathbf{u}_b}{d(\mathbf{p}_j, \mathbf{p}_b)} \mathrm{vol}(\Omega_{bj}). \end{aligned} \tag{22}$$

Moreover, by $\mathcal{N}(i) \cap \{b\} = \{b\}$ for $i \in M_b$ we conclude

$$\sum_{i \in M_b} \sum_{j \in \mathcal{N}(i) \cap \{b\}} \frac{\mathbf{u}_j - \mathbf{u}_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}) = \sum_{i \in M_b} \frac{\mathbf{u}_b - \mathbf{u}_i}{d(\mathbf{p}_b, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ib}). \tag{23}$$

Note that the sum of (22) cancels the sum of (23). Thus, we have proven so far

$$\sum_{i \in I_b} \sum_{j \in \mathcal{N}(i) \cap I_b} \frac{\mathbf{u}_j - \mathbf{u}_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}) = \sum_{i \in M_b} \sum_{j \in \mathcal{N}(i) \cap M_b} \frac{\mathbf{u}_j - \mathbf{u}_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}).$$

Using that $\mathcal{N}(i) \cap M_b = M_b \setminus \{i\}$ for $i \in M_b$ and extending $\frac{\mathrm{vol}(\Omega_{ii})}{d(\mathbf{p}_i, \mathbf{p}_i)} = 1$ we get

$$\begin{aligned}
& \sum_{i \in M_b} \sum_{j \in \mathcal{N}(i) \cap M_b} \frac{\mathbf{u}_j - \mathbf{u}_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}) \\
= & \sum_{i \in M_b} \sum_{j \in M_b \setminus \{i\}} \frac{\mathbf{u}_j - \mathbf{u}_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}) \\
= & \sum_{i \in M_b} \left(\sum_{j \in M_b \setminus \{i\}} \frac{\mathbf{u}_j - \mathbf{u}_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}) + \frac{\mathbf{u}_i - \mathbf{u}_i}{d(\mathbf{p}_i, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ii}) \right) \\
= & \sum_{i \in M_b} \sum_{j \in M_b} \frac{\mathbf{u}_j - \mathbf{u}_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}) = 0,
\end{aligned}$$

which proves the assertion.

□

So far we have left open the construction of the term $\mathbf{u}_{b,N}(t)$ in (20). To close this gap we define a new dual volume for some boundary point $\mathbf{p}_b$ by

$$\Omega_b' := \Omega_b \cup \left(\bigcup_{j \in M_b} \Omega_j \right) = \bigcup_{j \in I_b} \Omega_j. \tag{24}$$

Using (24) we define

$$\mathrm{vol}\left(\Omega_b'\right) \left(\frac{d\mathbf{u}_b(t)}{dt} \right) := \sum_{i \in I_b} \sum_{j \in L_{bi}} \frac{\mathbf{u}_j(t) - \mathbf{u}_i(t)}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ji}) + \mathbf{u}_{b,N}(t) \mathrm{vol}(\Omega_b), \tag{25}$$

which can be rewritten using (20)

$$\begin{aligned}
& \mathrm{vol}\left(\Omega_b'\right) \left(\frac{d\mathbf{u}_b(t)}{dt} \right) - \sum_{i \in I_b} \sum_{j \in L_{bi}} \frac{\mathbf{u}_j(t) - \mathbf{u}_i(t)}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ji}) \\
= \; & \mathrm{vol}\left(\Omega_b\right) \left(\frac{d\mathbf{u}_b(t)}{dt} \right) - \sum_{j \in \mathcal{N}(b)} \frac{\mathbf{u}_j(t) - \mathbf{u}_b(t)}{d(\mathbf{p}_j, \mathbf{p}_b)} \mathrm{vol}(\Omega_{jb})
\end{aligned} \tag{26}$$

We may now prove the following:

Lemma 3.6 *Equation (26) is, at least, a first order approximation of (15a) at some boundary point* $\mathbf{p}_b$.

Proof: Using Lemma 3.5 we subtract (21) from the left side of (26) to obtain

$$\begin{aligned}
\mathrm{vol}\,(\Omega_b') \left(\frac{d\mathbf{u}_b(t)}{dt}\right) &- \sum_{i\in I_b}\sum_{j\in L_{bi}} \frac{\mathbf{u}_j(t)-\mathbf{u}_i(t)}{d(\mathbf{p}_j,\mathbf{p}_i)}\mathrm{vol}(\Omega_{ji}) \\
&- \sum_{i\in I_b}\sum_{j\in\mathcal{N}(i)\cap I_b} \frac{\mathbf{u}_j(t)-\mathbf{u}_i(t)}{d(\mathbf{p}_j,\mathbf{p}_i)}\mathrm{vol}(\Omega_{ij}) \\
&= \mathrm{vol}\,(\Omega_b)\left(\frac{d\mathbf{u}_b(t)}{dt}\right) - \sum_{j\in\mathcal{N}(b)} \frac{\mathbf{u}_j(t)-\mathbf{u}_b(t)}{d(\mathbf{p}_j,\mathbf{p}_b)}\mathrm{vol}(\Omega_{jb}). \quad (27)
\end{aligned}$$

Since $L_{bi} = \mathcal{N}(i) \setminus I_b = \mathcal{N}(i) \setminus (\mathcal{N}(i) \cap I_b)$ we have

$$L_{bi} \cup (\mathcal{N}(i) \cap I_b) = \mathcal{N}(i). \quad (28)$$

Putting together (27) and (28) we get for the left hand side of (27)

$$\begin{aligned}
\mathrm{vol}\,(\Omega_b') \left(\frac{d\mathbf{u}_b(t)}{dt}\right) &- \sum_{i\in I_b}\sum_{j\in\mathcal{N}(i)} \frac{\mathbf{u}_j(t)-\mathbf{u}_i(t)}{d(\mathbf{p}_j,\mathbf{p}_i)}\mathrm{vol}(\Omega_{ji}) \\
&= \mathrm{vol}\,(\Omega_b)\left(\frac{d\mathbf{u}_b(t)}{dt}\right) - \sum_{j\in\mathcal{N}(b)} \frac{\mathbf{u}_j(t)-\mathbf{u}_b(t)}{d(\mathbf{p}_j,\mathbf{p}_b)}\mathrm{vol}(\Omega_{jb}). \quad (29)
\end{aligned}$$

Using $I_b \setminus \{b\} = M_b$

$$\begin{aligned}
\mathrm{vol}\,(\Omega_b) \left(\frac{d\mathbf{u}_b(t)}{dt}\right) &= \mathrm{vol}\,(\Omega_b')\left(\frac{d\mathbf{u}_b(t)}{dt}\right) - \sum_{i\in I_b}\sum_{j\in\mathcal{N}(i)} \frac{\mathbf{u}_j(t)-\mathbf{u}_i(t)}{d(\mathbf{p}_j,\mathbf{p}_i)}\mathrm{vol}(\Omega_{ji}) \\
&\quad + \sum_{j\in\mathcal{N}(b)} \frac{\mathbf{u}_j(t)-\mathbf{u}_b(t)}{d(\mathbf{p}_j,\mathbf{p}_b)}\mathrm{vol}(\Omega_{jb}) \\
&= \mathrm{vol}\,(\Omega_b')\left(\frac{d\mathbf{u}_b(t)}{dt}\right) - \sum_{i\in M_b}\sum_{j\in\mathcal{N}(i)} \frac{\mathbf{u}_j(t)-\mathbf{u}_i(t)}{d(\mathbf{p}_j,\mathbf{p}_i)}\mathrm{vol}(\Omega_{ji}) \quad (30)
\end{aligned}$$

The sum over M_b is a sum of the approximate Laplacians of interior points, the boundary point excluded. Solving (30) for $\frac{d\mathbf{u}_b(t)}{dt}$ using (24) and (17)

$$\begin{aligned}
\frac{d\mathbf{u}_b(t)}{dt} &= \frac{1}{\sum_{j\in M_b}\mathrm{vol}(\Omega_j)}\left(\sum_{i\in M_b}\sum_{j\in\mathcal{N}(i)} \frac{\mathbf{u}_j(t)-\mathbf{u}_i(t)}{d(\mathbf{p}_j,\mathbf{p}_i)}\mathrm{vol}(\Omega_{ji})\right) \\
&= \frac{1}{\sum_{j\in M_b}\mathrm{vol}(\Omega_j)}\left(\sum_{i\in M_b}\mathrm{vol}(\Omega_i)\frac{d\mathbf{u}_i(t)}{dt}\right). \quad (31)
\end{aligned}$$

Since

$$\frac{d\mathbf{u}_i(t)}{dt} = \frac{1}{\mathrm{vol}\Omega_i}\sum_{j\in\mathcal{N}(i)}\left(\frac{\mathbf{u}_j(t)-\mathbf{u}_i(t)}{d(\mathbf{p}_j,\mathbf{p}_i)}\right)\mathrm{vol}(\Omega_{ij})$$

is an approximation of $(\Delta\mathbf{u})_i$ one can show using for example Taylor's expansion that

$$\frac{d\mathbf{u}_i(t)}{dt} = (\Delta\mathbf{u})_\mathbf{b} + O(d(\mathbf{p}_b, \mathbf{p}_i)), \qquad i \in \mathcal{N}(b).$$

Let $h_b := \max\{d(\mathbf{p}_b, \mathbf{p}_i)\}$, $i \in M_b$, we finally obtain using (31)

$$\frac{d\mathbf{u}_b(t)}{dt} = \frac{1}{\sum_{j\in M_b} \operatorname{vol}(\Omega_j)} \left(\sum_{i\in M_b} \operatorname{vol}(\Omega_i) \frac{d\mathbf{u}_b(t)}{dt} \right) + O(h_b),$$

which represents a first order approximation of the Laplacian at $\mathbf{p}_b$.

□

3.4 Construction of dissipative operators for flow problems

Equation (17) serves as a prototype to define a dissipative operator to stabilize a convective dominated partial differential equation. By constructing a dissipative operator it is our goal not to influence the properties of the operator proven in Theorem 3.3. To this end, in a first step we generalize (17) to

$$\operatorname{vol}(\Omega_i) \left(\frac{d\mathbf{u}_i(t)}{dt} \right) = \sum_{j\in\mathcal{N}(i)} \frac{\mathbf{u}_j(t) - \mathbf{u}_i(t)}{d(\mathbf{p}_j, \mathbf{p}_i)} \operatorname{vol}(\Omega_{ij}) w_{ji}, \tag{32}$$

where $w_{ji} > 0$ denotes some weight satisfying $w_{ji} = w_{ij}$. In a further step we scale the approximate Laplacian with some grid dependent value. For example, to obtain a classical dissipative operator we scale the right hand side operator of (32) with some grid size h, which may be locally given by $h_{ji}^{(0)} = d(\mathbf{p}_j, \mathbf{p}_i)/\operatorname{vol}(\Omega_{ij})$. Then (32) becomes

$$\begin{aligned}
\frac{d\mathbf{u}_i(t)}{dt} &= h \frac{1}{\operatorname{vol}(\Omega_i)} \sum_{j\in\mathcal{N}(i)} \frac{\mathbf{u}_j(t) - \mathbf{u}_i(t)}{d(\mathbf{p}_j, \mathbf{p}_i)} \operatorname{vol}(\Omega_{ij}) w_{ji} \\
&\approx \frac{1}{\operatorname{vol}(\Omega_i)} \sum_{j\in\mathcal{N}(i)} h_{ji}^{(0)} \frac{\mathbf{u}_j(t) - \mathbf{u}_i(t)}{d(\mathbf{p}_j, \mathbf{p}_i)} \operatorname{vol}(\Omega_{ij}) w_{ji} \\
&= \sum_{j\in\mathcal{N}(i)} \left[\mathbf{u}_j(t) - \mathbf{u}_i(t)\right] w_{ji}. \qquad (33)
\end{aligned}$$

However, note that $h_{ji}^{(0)}$ is only one possible choice. Another one suggested in [24] is for example $h_{ji}^{(1)} = d(\mathbf{p}_j, \mathbf{p}_i)^2$. Then we obtain

$$\begin{aligned}
\frac{d\mathbf{u}_i(t)}{dt} &= h \frac{1}{\operatorname{vol}(\Omega_i)} \sum_{j\in\mathcal{N}(i)} \frac{\mathbf{u}_j(t) - \mathbf{u}_i(t)}{d(\mathbf{p}_j, \mathbf{p}_i)} \operatorname{vol}(\Omega_{ij}) w_{ji} \\
&\approx \frac{1}{\operatorname{vol}(\Omega_i)} \sum_{j\in\mathcal{N}(i)} \left[\mathbf{u}_j(t) - \mathbf{u}_i(t)\right] d(\mathbf{p}_j, \mathbf{p}_i) \operatorname{vol}(\Omega_{ij}) w_{ji}. \qquad (34)
\end{aligned}$$

Let us shortly analyze the dissipation operators (33) and (34) on a 2d cartesian grid. Let h denote the grid spacing and assume that the unknowns are ordered in a matrix $\mathbf{u}_{i,j}$. Then we have $V_i = h^2$, $d(\mathbf{p}_j, \mathbf{p}_i) = h$ and $\mathrm{vol}(\Omega_{ij}) = h$. Choosing the weights as normal directions, we obtain for (34)

$$\begin{aligned}\frac{d\mathbf{u}_{k,k}(t)}{dt} &= \frac{1}{h^2}\big\{(\mathbf{u}_{k+1,k}(t) - 2\mathbf{u}_{k,k}(t) + \mathbf{u}_{k-1,k}(t))\, h^2 \langle e_1, e_1\rangle \\ &\quad + (\mathbf{u}_{k,k+1}(t) - 2\mathbf{u}_{k,k}(t) + \mathbf{u}_{k,k-1}(t))\, h^2 \langle e_2, e_2\rangle\big\} \\ &= (\mathbf{u}_{k+1,k}(t) + \mathbf{u}_{k,k+1}(t) - 4\mathbf{u}_{k,k}(t) + \mathbf{u}_{k,k-1}(t) + \mathbf{u}_{k-1,k}(t))\,,\end{aligned}$$

which represents the undivided standard stencil of (15a) in 2d. Note that (33) gives exactly the same result. Assuming that $u(.,t) \in C^{2,1}(U)$ by Taylor's expansion the approximate second derivatives satisfy

$$\begin{aligned}\frac{1}{h^2}(\mathbf{u}_{k+1,k}(t) - 2\mathbf{u}_{k,k}(t) + \mathbf{u}_{k-1,k}(t)) &= o(h), \\ \frac{1}{h^2}(\mathbf{u}_{k,k+1}(t) - 2\mathbf{u}_{k,k}(t) + \mathbf{u}_{k,k-1}(t)) &= o(h).\end{aligned}$$

Therefore, for a cartesian mesh the dissipation operators defined by the right hand side of (33) and (34) are of order $o(h)$. In general, for a given unstructured mesh we cannot expect better than this.

To increase the order of accuracy of the dissipation operator we define

$$(\Delta\mathbf{u}(t))_i := \sum_{j\in\mathcal{N}(i)} \left(\frac{\mathbf{u}_j(t) - \mathbf{u}_i(t)}{d(\mathbf{p}_j, \mathbf{p}_i)}\right) \mathrm{vol}(\Omega_{ij}) w_{ji}. \tag{35}$$

According to (17) we now apply the construction of the dissipation operator above to $(\Delta\mathbf{u}(t))_i$ and obtain

$$\mathrm{vol}(\Omega_i)\left(\frac{d(\Delta\mathbf{u}(t))_i}{dt}\right) = \sum_{j\in\mathcal{N}(i)} \frac{\mathrm{vol}(\Omega_j)^{-1}(\Delta\mathbf{u}(t))_j - \mathrm{vol}(\Omega_i)^{-1}(\Delta\mathbf{u}(t))_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij}),$$

which can be viewed as the application of the right hand side of (16) twice. Again, introducing some weight $\tilde{w}_{ij} > 0$, $\tilde{w}_{ij} = \tilde{w}_{ji}$ a dissipative operator may be given by

$$\begin{aligned}\frac{d(\Delta\mathbf{u}(t))_i}{dt} &= \frac{1}{\mathrm{vol}(\Omega_i)} \sum_{j\in\mathcal{N}(i)} \frac{\mathrm{vol}(\Omega_j)^{-1}(\Delta\mathbf{u}(t))_j - \mathrm{vol}(\Omega_i)^{-1}(\Delta\mathbf{u}(t))_i}{d(\mathbf{p}_j, \mathbf{p}_i)} \mathrm{vol}(\Omega_{ij})\tilde{w}_{ji} \\ &= \frac{1}{\mathrm{vol}(\Omega_i)} \sum_{j\in\mathcal{N}(i)} \frac{\mathrm{vol}(\Omega_{ij})\tilde{w}_{ji}}{d(\mathbf{p}_j, \mathbf{p}_i)} \\ &\quad \left\{\frac{1}{\mathrm{vol}(\Omega_j)} \sum_{k\in\mathcal{N}(j)} \left(\frac{\mathbf{u}_k(t) - \mathbf{u}_j(t)}{d(\mathbf{p}_k, \mathbf{p}_j)}\right) \mathrm{vol}(\Omega_{kj}) w_{jk} \right. \\ &\quad \left. -\frac{1}{\mathrm{vol}(\Omega_i)} \sum_{k\in\mathcal{N}(i)} \left(\frac{\mathbf{u}_k(t) - \mathbf{u}_i(t)}{d(\mathbf{p}_k, \mathbf{p}_i)}\right) \mathrm{vol}(\Omega_{ki}) w_{ik} \right\},\end{aligned}$$

which as above may be scaled with some grid dependent value. Following the idea above we multiply with $d(\mathbf{p}_k, \mathbf{p}_j)^2$,

$$\begin{aligned} & \frac{1}{\mathrm{vol}(\Omega_j)} \sum_{k \in \mathcal{N}(j)} \left(\frac{\mathbf{u}_k(t) - \mathbf{u}_j(t)}{d(\mathbf{p}_k, \mathbf{p}_j)} \right) \mathrm{vol}(\Omega_{kj}) w_{jk} \\ \sim \; & \frac{1}{\mathrm{vol}(\Omega_j)} \sum_{k \in \mathcal{N}(j)} \left(\mathbf{u}_k(t) - \mathbf{u}_j(t) \right) d(\mathbf{p}_k, \mathbf{p}_j) \mathrm{vol}(\Omega_{kj}) w_{jk} \end{aligned}$$

and get

$$\begin{aligned} \frac{d(\Delta \mathbf{u}(t))_i}{dt} \sim \; & \frac{1}{\mathrm{vol}(\Omega_i)} \sum_{j \in \mathcal{N}(i)} \mathrm{vol}(\Omega_{ij}) \tilde{w}_{ji} d(\mathbf{p}_j, \mathbf{p}_i) \\ & \left\{ \frac{1}{\mathrm{vol}(\Omega_j)} \sum_{k \in \mathcal{N}(j)} \left(\mathbf{u}_k(t) - \mathbf{u}_j(t) \right) d(\mathbf{p}_k, \mathbf{p}_j) \mathrm{vol}(\Omega_{kj}) w_{jk} \right. \\ & \left. - \frac{1}{\mathrm{vol}(\Omega_i)} \sum_{k \in \mathcal{N}(i)} \left(\mathbf{u}_k(t) - \mathbf{u}_i(t) \right) d(\mathbf{p}_k, \mathbf{p}_i) \mathrm{vol}(\Omega_{ki}) w_{ik} \right\}. \end{aligned} \tag{36}$$

Considering as above a 2d cartesian grid with grid size h and assuming again that the unknowns are ordered in a matrix $\mathbf{u}_{i,j}$ we have $\mathrm{vol}(\Omega)_i = h^2$, $d(\mathbf{p}_j, \mathbf{p}_i) = h$

and vol$(\Omega_{ij}) = h$. Then the dissipation operator (36) can be expressed by

$$\begin{aligned}
\frac{d(\Delta \mathbf{u}(t))_i}{dt} &\sim \sum_{j\in\mathcal{N}(i)} \tilde{w}_{ji} \left\{ \sum_{k\in\mathcal{N}(j)} (\mathbf{u}_k(t) - \mathbf{u}_j(t))\, w_{jk} \right. \\
&\quad \left. - \sum_{k\in\mathcal{N}(i)} (\mathbf{u}_k(t) - \mathbf{u}_i(t))\, w_{ik} \right\} \\
&= \sum_{j\in\mathcal{N}(i)} \tilde{w}_{ji} \left\{ (\mathbf{u}_{j+1,j}(t) + \mathbf{u}_{j,j+1}(t) - 4\mathbf{u}_{j,j}(t) + \mathbf{u}_{j,j-1}(t) + \mathbf{u}_{j-1,j}(t)) \right. \\
&\quad \left. - (\mathbf{u}_{i+1,i}(t) + \mathbf{u}_{i,i+1}(t) - 4\mathbf{u}_{i,i}(t) + \mathbf{u}_{i,i-1}(t) + \mathbf{u}_{i-1,i}(t)) \right\} \\
&= -4\,(\mathbf{u}_{i+1,i}(t) + \mathbf{u}_{i,i+1}(t) - 4\mathbf{u}_{i,i}(t) + \mathbf{u}_{i,i-1}(t) + \mathbf{u}_{i-1,i}(t)) \\
&\quad + (\mathbf{u}_{i,i-2}(t) + \mathbf{u}_{i+1,i-1}(t) - 4\mathbf{u}_{i,i-1}(t) + \mathbf{u}_{i,i}(t) + \mathbf{u}_{i-1,i-1}(t)) \\
&\quad + (\mathbf{u}_{i+1,i-1}(t) + \mathbf{u}_{i+2,i}(t) - 4\mathbf{u}_{i+1,i}(t) + \mathbf{u}_{i+1,i+1}(t) + \mathbf{u}_{i,i}(t)) \\
&\quad + (\mathbf{u}_{i,i}(t) + \mathbf{u}_{i+1,i+1}(t) - 4\mathbf{u}_{i,i+1}(t) + \mathbf{u}_{i,i+2}(t) + \mathbf{u}_{i-1,i+1}(t)) \\
&\quad + (\mathbf{u}_{i-1,i-1}(t) + \mathbf{u}_{i,i}(t) - 4\mathbf{u}_{i-1,i}(t) + \mathbf{u}_{i-1,i+1}(t) + \mathbf{u}_{i-2,i}(t)) \\
&= -(\mathbf{u}_{i,i+2}(t) - 4\mathbf{u}_{i,i+1}(t) + 6\mathbf{u}_{i,i}(t) - 4\mathbf{u}_{i,i-1}(t) + \mathbf{u}_{i,i-2}(t)) \\
&\quad - (\mathbf{u}_{i-2,i}(t) - 4\mathbf{u}_{i-1,i}(t) + 6\mathbf{u}_{i,i}(t) - 4\mathbf{u}_{i+1,i}(t) + \mathbf{u}_{i+2,i}(t)) \\
&\quad + \{(\mathbf{u}_{i-1,i-1}(t) - 2\mathbf{u}_{i-1,i}(t) + \mathbf{u}_{i-1,i+1}(t)) \\
&\quad -2\,(\mathbf{u}_{i,i-1}(t) - 2\mathbf{u}_{i,i}(t) + \mathbf{u}_{i,i+1}(t)) \\
&\quad (\mathbf{u}_{i+1,i-1}(t) - 2\mathbf{u}_{i+1,i}(t) + \mathbf{u}_{i+1,i+1}(t))\} \\
&\quad + \{(\mathbf{u}_{i-1,i-1}(t) - 2\mathbf{u}_{i,i-1}(t) + \mathbf{u}_{i+1,i-1}(t)) \\
&\quad -2\,(\mathbf{u}_{i-1,i}(t) - 2\mathbf{u}_{i,i}(t) + \mathbf{u}_{i+1,i}(t)) \\
&\quad (\mathbf{u}_{i-1,i+1}(t) - 2\mathbf{u}_{i,i+1}(t) + \mathbf{u}_{i+1,i+1}(t))\}\,. \qquad (37)
\end{aligned}$$

By (37) it is obvious that on a 2d cartesian mesh the dissipation operator (36) is an approximation of the differential operator

$$h^4\Delta(\Delta u) = h^4\left(\frac{\partial^4 u}{\partial x_1^4} + \frac{\partial^4 u}{\partial x_1^2 \partial x_2^2} + \frac{\partial^4 u}{\partial x_2^2 \partial x_1^2} + \frac{\partial^4 u}{\partial x_2^4}\right).$$

This analyis can be extended to 3d, and it can be shwon that in 3d on a cartesian grid (36) approximates the differential operator

$$\begin{aligned}
h^4\Delta(\Delta u) &= h^4\left(\frac{\partial^4 u}{\partial x_1^4} + \frac{\partial^4 u}{\partial x_2^2 \partial x_1^2} + \frac{\partial^4 u}{\partial x_3^2 \partial x_1^2}\right. \\
&\quad + \frac{\partial^4 u}{\partial x_1^2 \partial x_2^2} + \frac{\partial^4 u}{\partial x_2^4} + \frac{\partial^4 u}{\partial x_3^2 \partial x_2^2} \\
&\quad \left. + \frac{\partial^4 u}{\partial x_1^2 \partial x_3^2} + \frac{\partial^4 u}{\partial x_2^2 \partial x_3^2} + \frac{\partial^4 u}{\partial x_3^4}\right). \qquad (38)
\end{aligned}$$

Hence, at least on a cartesian mesh we conclude that the construction (36) yields for an inner point to an operator of order ($o(h^2)$).

We want to emphasize that on a given, unstructured mesh one cannot expect to get this order of accuracy for the operator above. Moreover, (38) also shows a major difference of the construction of the dissipative operator when compared with a structured code. In general, in a structured code having determined directions in hand, the dissipative operator is constructed by considering only the diagonal terms of (38), that is $\frac{\partial^4 u}{\partial x_i^4}$, $i = 1, 2, 3$, whereas the construction (36) also leads to cross derivative terms, as shown in (38).

3.5 Construction of blended dissipation

In Sections 3.3 and 3.4 above we have presented some general view on the construction of a dissipative operator. As we have seen there is quite some freedom to choose proper weightings for the dissipation operators to ensure both

a) accuracy,

b) reliablity.

On the other hand it is clear that if the weights are not chosen adequately both accuracy and reliablity can be severely deteriorated.

It is now our goal to define and investigate required terms for the dissipative term $\mathbf{D}$ given in (10). Starting with the pressure switch $\Psi_{ij}^{(t)}, t = 1, 2$, we investigate the two formulations

$$\Psi_{ij}^{(1)} := \min\{\varepsilon_\psi \max\{\Psi_i, \Psi_j\}, 1\}, \qquad \Psi_i := \left| \frac{\sum_{j\in\mathcal{N}(i)}(p_j - p_i)}{\sum_{j\in\mathcal{N}(i)}(p_j + p_i)} \right|, \tag{39a}$$

$$\Psi_{ij}^{(2)} := \min\{\varepsilon_\psi \Psi_{ij}, 1\}, \qquad \Psi_{ij} := \frac{(p_j - p_i)^2}{(p_j + p_i)^2}. \tag{39b}$$

Note that (39a) and (39b) only differ in the formulation of the scaled pressure difference over the face ij. Whereas (39a) takes into account neighbor information which increases the stencil, (39b) only evaluates the difference across some face, yielding a nice compact stencil. The formulation of the pressure sensors (39a) and (39b) are such that across a shock the pressure difference is large, and therefore the first order terms in (10) are activated, whereas in smooth regions of the flow the first order terms should be negligible. The unknown value ε_ψ deals as weight for the scaled pressure difference. The choice $\varepsilon_\psi = 0$ switches off the first order terms, the choice $\varepsilon_\psi = \infty$ gives a purely first order scheme assuming that either $\max\{\psi_i, \psi_j\} > 0$ or $\Psi_{ij} > 0$ for all i, j. In applications we use in general $\varepsilon_\psi \in [1, 8]$.

The complete first order dissipative term for the point i is given by

$$\begin{aligned}
\mathbf{D}^{1st}\left(\mathbf{W}_i, \mathbf{W}_{j,j\in\mathcal{N}(i)}\right) &= \frac{1}{2}\sum_{j\in\mathcal{N}(i)} \psi_{ij}^{(t)} |\mathbf{A}_{ij}| \left(\mathbf{W}_j - \mathbf{W}_i\right) \\
&= \frac{1}{2}\sum_{j\in\mathcal{N}(i)} \psi_{ij}^{(t)} \mathbf{G}_{ij}|\Lambda_{ij}|\mathbf{G}_{ij}^{-1} \left(\mathbf{W}_j - \mathbf{W}_i\right),
\end{aligned}$$

which corresponds to (33). The weight satisfying $w_{ji} = w_{ij}$ is given by

$$w_{ji} = \psi_{ij}^{(t)} \mathbf{G}_{ij}|\Lambda_{ij}|\mathbf{G}_{ij}^{-1}.$$

Following the analysis of Section 3.3 we conclude that $\mathbf{D}^{1st}$ can be written as

$$\mathbf{D}^{1st}\left(\mathbf{W}_i, \mathbf{W}_{j,j\in\mathcal{N}(i)}\right) = \mathbf{B}\mathbf{W},$$

where

$$\begin{aligned}
b_{ii} &= -\sum_{j\in\mathcal{N}(i)} \psi_{ij}^{(t)} \mathbf{G}_{ij}|\Lambda_{ij}|\mathbf{G}_{ij}^{-1}, \\
b_{ij} &= \psi_{ij}^{(t)} \mathbf{G}_{ij}|\Lambda_{ij}|\mathbf{G}_{ij}^{-1}.
\end{aligned}$$

Note that each b_{ij} is a block matrix. By similar arguments we may conclude that all eigenvalues of $\mathbf{B}$ are nonpositive.
The remaining term on the right hand side of (10) given by

$$\begin{aligned}
&\mathbf{D}^{2nd}\left(\mathbf{W}_i, \mathbf{W}_{j,j\in\mathcal{N}(i)}, \mathbf{W}_{k,k\in\mathcal{N}(j)}\right) \\
= &-\frac{1}{2}\sum_{j\in\mathcal{N}(i)} s_{ij}(\mathbf{W})\left(1-\psi_{ij}^{(t)}\right)|\mathbf{A}_{ij}|\left(L_j^{(z)}(\mathbf{W}) - L_i^{(z)}(\mathbf{W})\right) \qquad (40)
\end{aligned}$$

is more complicated. First of all we define

$$\begin{aligned}
L_i^{(1)}(\mathbf{W}) &:= \sum_{j\in\mathcal{N}(i)} \left(\mathbf{W}_j - \mathbf{W}_i\right), \\
L_i^{(2)}(\mathbf{W}) &:= \frac{1}{\#\mathcal{N}(i)}\sum_{j\in\mathcal{N}(i)} \left(\mathbf{W}_j - \mathbf{W}_i\right), \\
L_i^{(3)}(\mathbf{W}) &:= \left(\sum_{j\in\mathcal{N}(i)} \frac{1}{d(\mathbf{p}_j,\mathbf{p}_i)}\right)^{-1} \sum_{j\in\mathcal{N}(i)} \frac{\left(\mathbf{W}_j - \mathbf{W}_i\right)}{d(\mathbf{p}_j,\mathbf{p}_i)}.
\end{aligned}$$

Then we conclude that the construction of (40) corresponds to (36) and (37) respectively. Note that on a cartesian mesh we have $L_i^{(2)} = L_i^{(3)}$ and therefore $L_i^{(3)}$ can be viewed as generalization of $L_i^{(2)}$. The operator $L_i^{(2)}$ has been introduced

dh	$(k-2,\ell+2)$	$(k-1,\ell+2)$	$(k,\ell+2)$	$(k+1,\ell+2)$	$(k+2,\ell+2)$
dh	$(k-2,\ell+1)$	$(k-1,\ell+1)$	$(k,\ell+1)$	$(k+1,\ell+1)$	$(k+2,\ell+1)$
dh	$(k-2,\ell)$	$(k-1,\ell)$	(k,ℓ)	$(k+1,\ell)$	$(k+2,\ell)$
dh	$(k-2,\ell-1)$	$(k-1,\ell-1)$	$(k,\ell-1)$	$(k+1,\ell-1)$	$(k+2,\ell-1)$
dh	$(k-2,\ell-2)$	$(k-1,\ell-2)$	$(k,\ell-2)$	$(k+1,\ell-2)$	$(k+2,\ell-2)$
	dH	dH	dH	dH	dH

Figure 2: Example of an anisotropic cartesian mesh

in [5]. The scaling by grid distances has been suggested in [18]. Such a scaling is important when working with highly stretched meshes as required in boundary layers for RANS computations.

When choosing $L_i^{(2)}$ or $L_i^{(3)}$ we set $s(\mathbf{W}) = 1$ and use the resulting scheme as dissipative operator. Another idea to introduce some proper scaling for the dissipative term $\mathbf{D}^{2nd}$ is to use values arising from the scheme itself. To this end we construct a scalar $s^{(t)}$ based on the largest convective eigenvalues. Using the notation

$$\begin{aligned} \lambda_{ij}^{max} &:= |V_{ij}| + \mathrm{vol}(\Omega_{ij})a_{ij}, \qquad \lambda_i^{max} := \sum_{j\in\mathcal{N}(i)} \lambda_{ij} \\ z_i^{(1)} &:= \frac{\lambda_i - \lambda_{ij}}{\lambda_{ij}}, \qquad z_i^{(2)} := \frac{\max\{\frac{1}{2}\lambda_i - \lambda_{ij}, 0\}}{\lambda_{ij}} \end{aligned}$$

we define

$$s_{ij}^{(t)}(\mathbf{W}) := 1 + 2\frac{\sqrt{z_i^{(t)}}\sqrt{z_j^{(t)}}}{\sqrt{z_i^{(t)}} + \sqrt{z_j^{(t)}}}. \tag{41}$$

To understand the construction (41) we consider the boundary layer of an anisotropic mesh, represented in Figure 2. Assuming that the magnitude of the speed of sound is much greater than the streamwise normal velocities in the viscous boundary layer, that the speed of sound is constant over neighboring cells and that $dH \gg dh$, we approximate for a vertical face for example between the cells (k,ℓ) and $(k+1,\ell)$

$$\begin{aligned} z_i^{(1)} &= \frac{\sum_{k\in\mathcal{N}(i)} (|V_{ik}| + \mathrm{vol}(\Omega_{ik})a_{ik}) - |V_{ij}| + \mathrm{vol}(\Omega_{ij})a_{ij}}{|V_{ij}| + \mathrm{vol}(\Omega_{ij})a_{ij}} \\ &\approx \frac{2dH + dh}{dh} \approx 2\frac{dH}{dh} \end{aligned}$$

and therefore

$$s_{ij}^{(1)}(\mathbf{W}) \approx 1 + 2\frac{2\frac{dH}{dh}}{2\sqrt{2\frac{dH}{dh}}} = 1 + \sqrt{2}\sqrt{\frac{dH}{dh}}. \tag{42}$$

Exchanging the role of dh and dH we get for a horizontal face (e.g. the face between the cells (k,ℓ) and $(k,\ell+1)$

$$z_i^{(1)} \;\approx\; \frac{2dh + dH}{dH} \approx 1$$

and therefore

$$s_{ij}^{(1)}(\mathbf{W}) \;\approx 1 + 2\tfrac{1}{2} = 2.$$

Following the arguments for $z_i^{(1)}$ we may also conclude for a vertical face for example between the cells (k,ℓ) and $(k+1,\ell)$

$$z_i^{(2)} \approx \frac{\frac{1}{2}\,(2dH + dh)}{dh} \approx \frac{dH}{dh} \tag{43}$$

and therefore

$$s_{ij}^{(2)}(\mathbf{W}) \;\approx 1 + 2\frac{\frac{dH}{dh}}{2\sqrt{\frac{dH}{dh}}} = 1 + \sqrt{\tfrac{dH}{dh}}.$$

The approximations (42) and (43) show that the for both $z_i^{(1)}$ and $z_i^{(2)}$ the factor s_{ij} is designed to increase the dissipation significantly for highly stretched cells in direction of the cell stretching. This is a desired effect to improve the reliability of the algorithm.

Let us still shortly discuss the difference between $z_i^{(1)}$ and $z_i^{(2)}$. As seen above from (42) and (43) $z_i^{(1)}$ produces finally slightly bit more dissipation for highly stretched meshes. However, the advantage of $z_i^{(1)}$ is that it satisfies $z_i^{(1)} > 0$. Therefore the expression $s_{ij}^{(1)}$ is well defined. Although $z_i^{(2)}$ produces slightly bit less dissipation it happens that $z_i^{(2)} = 0$. Hence, $s_{ij}^{(2)}$ is not necessarily well defined for all states and all faces. In particular in the critical starting phase of the algorithm in general this happens. Therefore, for $z_i^{(2)}$ some fix needs to be employed and the the term $s_{ij}^{(2)}$ must be exchanged by for example

$$\tilde{s}_{ij}^{(2)}(\mathbf{W}) := 1 + 2\frac{\sqrt{z_i^{(2)}}\sqrt{z_j^{(2)}}}{\sqrt{z_i^{(2)}} + \sqrt{z_j^{(2)}} + \varepsilon}, \qquad \varepsilon > 0$$

or

$$\bar{s}_{ij}^{(2)}(\mathbf{W}) := \begin{cases} 1, & z_i^{(2)} = 0 \;\text{ and }\; z_j^{(2)} = 0, \\ s_{ij}^{(2)}(\mathbf{W}), & \textit{else}. \end{cases}$$

In summary: To construct the dissipative part of the convective flux we can choose the pressure sensor either by (39a) or by (39b). These pressure sensors can be combined with $L_i^{(t)}$, $t = 1,2,3$, where for the choice $L_i^{(1)}$ again between $z_i^{(1)}$ or $z_i^{(2)}$ may be chosen. Hence, totally we have 8 different formulations of a dissipative term and it is hard to evaluate which to prefer.

3.6 Construction of dissipation at the boundary

For the implementation of the dissipation at the boundary we have not tried yet the rather complicate formulation in formula (25). Instead we construct the following operator at the boundary. Using the notation of Figure 3, which shows a dual cell around a boundary point $\mathbf{p}_b$ given by a half cell, we close this boundary element by mirroring the values with respect to the nearest boundary point $\mathbf{p}_n$. Then, for a quadrilateral element as given in the right of Figure 3 we obtain using (33)

$$\begin{aligned}\frac{d\mathbf{u}_b(t)}{dt} &= \Big(\mathbf{u}_{\mathbf{p}_{b_{-1}}}(t) - \mathbf{u}_{\mathbf{p}_b}(t)\Big) + \Big(\mathbf{u}_{\mathbf{p}_{b_1}}(t) - \mathbf{u}_{\mathbf{p}_b}(t)\Big) \\ &\quad + (\mathbf{u}_{\mathbf{p}_n}(t) - \mathbf{u}_{\mathbf{p}_b}(t)) + (\mathbf{u}_{\mathbf{p}_b}(t) - \mathbf{u}_{\mathbf{p}_n}(t)) \\ &= \mathbf{u}_{\mathbf{p}_{b_{-1}}}(t) - 2\mathbf{u}_{\mathbf{p}_b}(t) + \mathbf{u}_{\mathbf{p}_{b_1}}(t). \end{aligned} \tag{44}$$

Obviously this constructions yields for a quadrilatel mesh an undivided Laplacian on the boundary. Unfortunately this construction does neither carry over straightforward to triangular or tetrahedral meshes nor to curved boundaries. On the one hand for a general mesh as can be seen by the left of Figure 3 a nearest neighbor point is not well defined i.e. unique, on the other hand the projection is in general not directly in direction of the normal. Both properties are only satisfied for special cases.

Hence, to generalize the construction (44) to unstructured meshes we neglect in the boundary formulation of the dissipation the faces connecting the boundary point and the neighboring inner points. Then for both the triangular and quadrilateral mesh in Figure 3 we obtain (44) as boundary operator.

To obtain the fourth difference operator we simply apply the described idea above to $\frac{d\mathbf{u}_b(t)}{dt}$. Then we get a typical fourth difference on the boundary. For example for a one dimensional boundary surface as given in Figure 3 we obtain using (44)

$$\frac{d(\Delta u(t))_{\mathbf{b}}}{dt} = \mathbf{u}_{\mathbf{p}_{b_{-2}}}(t) - 4\mathbf{u}_{\mathbf{p}_{b_{-1}}}(t) + 6\mathbf{u}_{\mathbf{p}_b}(t) - 4\mathbf{u}_{\mathbf{p}_{b_1}}(t) + \mathbf{u}_{\mathbf{p}_{b_2}}(t). \tag{45}$$

Both (44) and (45) generalize straightfoward to 3d, where the surface boundary is represented by a two dimensional manifold.

3.7 Viscous contribution

The derivatives of the flow variables required to discretize the viscous terms $\mathbf{f}_v \cdot \mathbf{n}$ are computed by a Green-Gauss ansatz,

$$\frac{\partial w}{\partial x_k} \approx \left(\frac{\partial w}{\partial x_k}\right)^{\mathrm{GG}} := \frac{1}{\mathrm{vol}\,(\Omega_i)} \sum_{j \in \mathcal{N}(i)} \frac{n_k}{2}\left(w^{(i)} + w^{(j)}\right), \qquad k = 1,2,3, \tag{46}$$

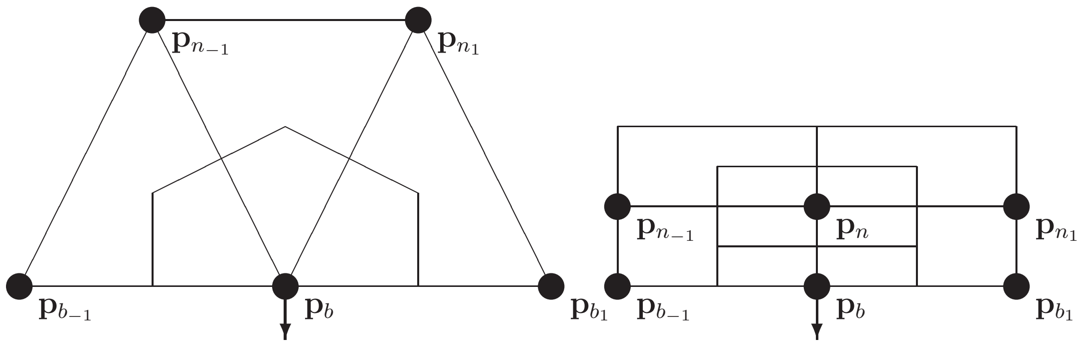

Figure 3: Example of an anisotropic cartesian mesh

where n_k describes the kth entry of the normal vector. Using an averaging of the velocities, the viscosity and the conductivity on the face $k\ell$

$$u_i := \frac{1}{2}\left(u_i^{(k)} + u_i^{(\ell)}\right), \qquad \mu_{\text{eff}} := \frac{1}{2}\left(\mu_{\text{eff}}^{(k)} + \mu_{\text{eff}}^{(\ell)}\right), \qquad \kappa_{\text{eff}} := \frac{1}{2}\left(\kappa_{\text{eff}}^{(k)} + \kappa_{\text{eff}}^{(\ell)}\right) \quad (47)$$

formulae (46) and (47) allow a straightforward implementation of the viscous terms.

3.8 Source terms

The source terms are discretized in a straightforward manner using the approximation

$$\int_{\Omega_i} \mathbf{Q}\,\mathrm{d}\mathbf{x} \approx \mathrm{vol}(\Omega_i)\mathbf{Q}\left(\mathbf{W}_i, \mathrm{grad}\ \mathbf{W}_i\right).$$

The dependency of the source terms with respect to grad $\mathbf{W}_i$ is due to curl $(\mathbf{u})$ and the diffusion term $\mathbf{Di}$. Hence, the source term also depends on the variables of the direct neighbors of point i. All the other terms occurring in the source terms only require direct information of the variables $\mathbf{W}_i$.

4 Line implicit preconditioner

4.1 Determination of lines

Grids given in an unstructured data format have no line information. This information must be generated. To this end we exploit that, in general, meshes for high Reynolds number turbulent flows have a structured boundary layer with strong anisotropies. To detect these anisotropies we formulate in this section a so-called line search algorithm. Similar line search algorithms have been suggested for example in [15, 6]. A line search algorithm based on the boundary surface can be found in [19]. In the context of Discontinuous Galerkin methods there also line search algorithms based on an advection-diffusion equation are established [7].

The line search algorithm presented here is based on a weighted graph method. Each edge of the mesh is assigned a weight $w(e_i)$ representing the degree of coupling in the discretization. The weights are taken as the inverse of the edge length,

$$w(e_i) := \left(\|v_i(\text{left}) - v_i(\text{right})\|_2\right)^{-1}, \tag{48}$$

where $v_i(\text{left})$ and $v_i(\text{right})$ denote the left and the right vertex of the edge e_i. The ratio of maximum to average weight is used as an indication of the local anisotropy in the mesh at each vertex.
In the next step the vertices are sorted according to the ratio of the maximum to the average weight. This is an important issue since it ensures that lines originate in areas of maximum grid stretching and end in isotropic regions.

To construct the lines the first vertex in this ordered list is picked as the starting point for a line. The line is built by adding to the original vertex the neighboring vertex which is most strongly connected to the current vertex, provided this vertex does not already belong to a line, and provided the ratio of maximum to minimum edge weights is greater than some threshold parameter $\gamma_{\text{line}} \geq 1$. The line terminates when no additional vertex can be found.

The line search algorithm can be summarized as follows:

1) For each vertex v_j, construct a list of edges $e_i(j)$ originating from the vertex and determine the weight (48).

2) Compute the minimum weight, maximum weight, the average weight and the ratio of both of them:

$$\begin{aligned}
\text{wmin}(v_j) &:= \min_{i \in \mathcal{N}(j)} \{w(e_i(j)\}, \qquad \text{wmax}(v_j) := \max_{i \in \mathcal{N}(j)} \{w(e_i(j)\} \\
\text{wavg}(v_j) &:= \frac{1}{\#\mathcal{N}(j)} \sum_{i=1}^{\#\mathcal{N}(j)} w(e_i(j)), \qquad \text{rat}(v_j) := \text{wmax}(v_j)/\text{wavg}(v_j).
\end{aligned}$$

3) Sort the vertices v_j with respect to $\mathrm{rat}(v_j)$.

4) Construct the lines:

   a) Set searchOppositeDirection = false.

   b) Pick the first vertex v_k out of the sorted list, delete it from the list and add it to the line.

   c) If the ratio $\mathrm{wmax}(v_k)/\mathrm{wmin}(v_k) \geq \gamma_{\text{line}}$:

      * Find the neighbor vertex v_{neig} corresponding to $\mathrm{wmax}(v_k)$, delete it from the sorted list and add it to the line.
      * Define $v_k := v_{\text{neig}}$ and go back to b).

   d) else if the ratio $\mathrm{wmax}(v_k)/\mathrm{wmin}(v_k) < \gamma_{\text{line}}$ and searchOppositeDirection = false:

      * searchOppositeDirection = true
      * Go back to the first element in the line. (E.g. when a line starts in a wake region, one has to search in both directions from the original element.)
      * Go to c).

   e) else

      * Go to a).

We denote the sets of points determined by the algorithm above by $L_1, \ldots, L_n$. These sets of points satisfy for $j = 1, \ldots, n$:

$$L_j \subset \{1, \ldots, N\}, \qquad L_j \cap L_i = \emptyset, \quad i \neq j, \qquad \cup_{j=1}^{n} L_j = \{1, \ldots, N\}$$

The number of points in the set L_i is given by

$$r_i := \#\{L_i\}, \qquad i = 1, \ldots, n.$$

If $r_i > 1$ we call L_i a line, otherwise we say L_i is a point.

4.2 Line implicit Runge-Kutta method

To derive a line implicit Runge-Kutta method we start with an s-stage diagonally implicit Runge-Kutta method given by the Butcher scheme

$$\mathcal{A} := \begin{pmatrix} \alpha_{11} & 0 & \cdots & 0 \\ \alpha_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & \alpha_{s,s-1} & \alpha_{ss} \end{pmatrix}, \quad b := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \alpha_{s+1,s} \end{pmatrix} \quad \text{and} \quad c := \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{49}$$

The stages of this scheme and the discrete evolution are given by

$$\begin{aligned}
k_1 &= -\mathbf{M}^{-1}\mathbf{R}\left(\mathbf{W}^{\mathrm{T}_n} + \Delta t\alpha_{11}k_1\right)\\
k_2 &= -\mathbf{M}^{-1}\mathbf{R}\left(\mathbf{W}^{\mathrm{T}_n} + \Delta t\alpha_{21}k_1 + \Delta t\alpha_{22}k_2\right)\\
&\vdots\\
k_s &= -\mathbf{M}^{-1}\mathbf{R}\left(\mathbf{W}^{\mathrm{T}_n} + \Delta t\alpha_{s,s-1}k_{s-1} + \Delta t\alpha_{ss}k_s\right)\\
\mathbf{W}^{\mathrm{T}_{n+1}} &= \mathbf{W}^{\mathrm{T}_n} + \Delta t\alpha_{s+1,s}k_s.
\end{aligned} \tag{50}$$

Instead of considering the stages k_j, $j = 1, \ldots, s$, being functions of all variables $\mathbf{W}_i$, $i = 1, \ldots, N$, we consider them as being functions only of the variables along the lines $L_1, \ldots, L_n$. Then the nonlinear equations k_j of the scheme (50) decouple for $i = 1, \ldots, n$ as follows:

$$\begin{aligned}
k_1(\mathbf{W}_{L_i}) &= -\mathbf{M}_{L_i}^{-1}\mathbf{R}_{L_i}\left(\mathbf{W}^{\mathrm{T}_n} + \mathrm{CFL}_{\mathrm{impl}}\Delta t_{L_i}\alpha_{11}k_1(\mathbf{W}_{L_i})\right)\\
k_2(\mathbf{W}_{L_i}) &= -\mathbf{M}_{L_i}^{-1}\mathbf{R}_{L_i}\left(\mathbf{W}^{\mathrm{T}_n} + \mathrm{CFL}_{\mathrm{impl}}\alpha_{12}\Delta t_{L_i}k_1(\mathbf{W}_{L_i}) + \mathrm{CFL}_{\mathrm{impl}}\alpha_{22}\Delta t_{L_i}k_2(\mathbf{W}_{L_i})\right)\\
&\vdots\\
k_s(\mathbf{W}_{L_i}) &= -\mathbf{M}_{L_i}^{-1}\mathbf{R}_{L_i}\left(\mathbf{W}^{\mathrm{T}_n} + \mathrm{CFL}_{\mathrm{impl}}\alpha_{s-1,j}\Delta t_{L_i}k_{s-1}(\mathbf{W}_{L_i}) + \mathrm{CFL}_{\mathrm{impl}}\alpha_{ss}\Delta t_{L_i}k_s(\mathbf{W}_{L_i})\right)\\
\mathbf{W}_{L_i}^{\mathrm{T}_{n+1}} &= \mathbf{W}_{L_i}^{\mathrm{T}_n} + \mathrm{CFL}_{\mathrm{impl}}\alpha_{s+1,s}\Delta t_{L_i}k_s(\mathbf{W}_{L_i}).
\end{aligned} \tag{51}$$

Here the diagonal matrices with respect to the local time steps and the cell volumes along the lines L_i are denoted by

$$\Delta t_{L_i} := \mathrm{diag}\left(\Delta t_{\ell_i^1}, \ldots, \Delta t_{\ell_i^{r_i}}\right) \quad \text{and} \quad \mathbf{M}_{L_i} := \mathrm{diag}\left(\mathrm{vol}\left(\Omega_{\ell_i^1}\right), \ldots, \mathrm{vol}\left(\Omega_{\ell_i^{r_i}}\right)\right).$$

To approximate a solution of the nonlinear systems $k_1, \ldots, k_s$ we use Newton's method to approximate the root of the function

$$\begin{aligned}
g_j(k_j(\mathbf{W}_{L_i})) := \; & k_j(\mathbf{W}_{L_i}) + \mathbf{M}_{L_i}^{-1}\mathbf{R}_{L_i}\left(\mathbf{W}^{\mathrm{T}_n}\right.\\
& \left. + \mathrm{CFL}_{\mathrm{impl}}\alpha_{j-1,j}\Delta t_{L_i}k_{j-1}(\mathbf{W}_{L_i}) + \mathrm{CFL}_{\mathrm{impl}}\alpha_{jj}\Delta t_{L_i}k_j(\mathbf{W}_{L_i})\right).
\end{aligned}$$

Its derivative is given by

$$\begin{aligned}
\frac{\partial g_j(k_j(\mathbf{W}_{L_i}))}{\partial k_j(\mathbf{W}_{L_i})}[k_j(\mathbf{W}_{L_i})] &= \mathbf{I} + \mathrm{CFL}_{\mathrm{impl}}\alpha_{jj}\Delta t_{L_i}\mathbf{M}_{L_i}^{-1}\frac{\partial \mathbf{R}_{L_i}}{\partial \mathbf{W}_{L_i}}\left(\mathbf{W}^{\mathrm{T}_n}\right.\\
&+ \left.\mathrm{CFL}_{\mathrm{impl}}\alpha_{j-1,j}\Delta t_{L_i}k_{j-1}(\mathbf{W}_{L_i}) + \mathrm{CFL}_{\mathrm{impl}}\alpha_{jj}\Delta t_{L_i}k_j(\mathbf{W}_{L_i})\right).
\end{aligned}$$

and Newton's method may be formulated by

$$\begin{aligned}
k_j^{(0)}(\mathbf{W}_{L_i}) &= 0 \qquad \text{(initial guess)}\\
\frac{\partial g_j(k_j(\mathbf{W}_{L_i}))}{\partial k_j(\mathbf{W}_{L_i})}[k_j^{(m)}(\mathbf{W}_{L_i})]h_m(L_i) &= -g(k_j^{(m)}(\mathbf{W}_{L_i})),\\
h_m(L_i) &= k_j^{(m+1)}(\mathbf{W}_{L_i}) - k_j^{(m)}(\mathbf{W}_{L_i}).
\end{aligned}$$

Stopping Newton's iteration after one step we get for $j = 1, \dots, s$ the iterates

$$k_j^{(1)}(\mathbf{W}_{L_i}) = -\left[\mathbf{P}_j(L_i)\right]^{-1}(g_j(k_j^{(0)}(\mathbf{W}_{L_i}))), \qquad i = 1, \dots, n, \tag{52}$$

where

$$\mathbf{P}_j(L_i) = \mathbf{I} + \mathrm{CFL}_{\mathrm{impl}}\alpha_{jj}\Delta t_{L_i}\mathbf{M}_{L_i}^{-1}\frac{\partial \mathbf{R}_{L_i}}{\partial \mathbf{W}_{L_i}}\left[\mathbf{W}^{\mathrm{T}_n} + \mathrm{CFL}_{\mathrm{impl}}\alpha_{j,j-1}\Delta t_{L_i}k_{j-1}^{(1)}(\mathbf{W}_{L_i})\right].$$

Using these formulae the line implicit Runge-Kutta method (51), where the inner Newton iteration is truncated after one step can, be represented by the algorithm

$$\begin{aligned}
k_1(\mathbf{W}_{L_i}) &= -\left[\mathbf{P}_1(L_i)\right]^{-1}\mathbf{M}_{L_i}^{-1}\mathbf{R}_{L_i}\left(\mathbf{W}^{\mathrm{T}_n}\right)\\
k_2(\mathbf{W}_{L_i}) &= -\left[\mathbf{P}_2(L_i)\right]^{-1}\mathbf{M}_{L_i}^{-1}\mathbf{R}_{L_i}\left(\mathbf{W}^{\mathrm{T}_n} + \mathrm{CFL}_{\mathrm{impl}}\alpha_{21}\Delta t_{L_i}k_1\right)\\
&\vdots\\
k_s(\mathbf{W}_{L_i}) &= -\left[\mathbf{P}_s(L_i)\right]^{-1}\mathbf{M}_{L_i}^{-1}\mathbf{R}_{L_i}\left(\mathbf{W}^{\mathrm{T}_n} + \mathrm{CFL}_{\mathrm{impl}}\alpha_{s+1,s}\Delta t_{L_i}k_{s-1}\right)\\
\mathbf{W}_{L_i}^{\mathrm{T}_{n+1}} &= \mathbf{W}_{L_i}^{\mathrm{T}_n} + \mathrm{CFL}_{\mathrm{impl}}\alpha_{s+1,s}\Delta t_{L_i}k_s(\mathbf{W}_{L_i}).
\end{aligned}$$

Defining the updates $\mathbf{W}_{L_i}^{(0)} := \mathbf{W}_{L_i}^{\mathrm{T}_n}$ and

$$\mathbf{W}_{L_i}^{(j)} := \mathbf{W}_{L_i}^{\mathrm{T}_n} - \mathrm{CFL}_{\mathrm{impl}}\alpha_{j+1,j}\Delta t_{L_i}\left[\mathbf{P}_j(L_i)\right]^{-1}\mathbf{M}_{L_i}^{-1}\mathbf{R}_{L_i}(\mathbf{W}^{(j-1)})$$

we notice that the Runge-Kutta scheme above can be formulated equivalently by

$$\begin{aligned}
\mathbf{W}_{L_i}^{(0)} &:= \mathbf{W}_{L_i}^{\mathrm{T}_n}\\
\mathbf{W}_{L_i}^{(1)} &= \mathbf{W}_{L_i}^{(0)} - \mathrm{CFL}_{\mathrm{impl}}\alpha_{21}\Delta t_{L_i}\mathbf{P}_1(L_i)^{-1}\mathbf{M}_{L_i}^{-1}\mathbf{R}_{L_i}\left(\mathbf{W}^{(0)}\right)\\
&\vdots\\
\mathbf{W}_{L_i}^{(s)} &= \mathbf{W}_{L_i}^{(0)} - \mathrm{CFL}_{\mathrm{impl}}\alpha_{s+1,s}\Delta t_{L_i}\mathbf{P}_s(L_i)^{-1}\mathbf{M}_{L_i}^{-1}\mathbf{R}_{L_i}\left(\mathbf{W}^{(s-1)}\right)\\
\mathbf{W}_{L_i}^{\mathrm{T}_{n+1}} &= \mathbf{W}_{L_i}^{(s)}.
\end{aligned} \tag{53}$$

Algorithm (53) indicates that along each line the linear system

$$\mathbf{P}_j(L_i)\mathbf{h}_{L_i} = \mathrm{CFL}_{\mathrm{impl}}\alpha_{j+1,j}\Delta t_{L_i}\mathbf{M}_{L_i}^{-1}\mathbf{R}_{L_i}(\mathbf{W}^{(j-1)})$$

needs to be solved. This can be equivalently formulated by

$$\left(\frac{1}{\mathrm{CFL}_{\mathrm{impl}}}\left(\Delta t_{L_i}\right)^{-1}\mathbf{M}_{L_i} + \alpha_{jj}\frac{\partial \mathbf{R}_{L_i}}{\partial \mathbf{W}_{L_i}}\right)\mathbf{h}_{L_i} = \alpha_{j+1,j}\mathbf{R}_{L_i}(\mathbf{W}^{(j-1)}). \tag{54}$$

The exact construction of the preconditioner $\mathbf{P}_j$ is topic of the next section. Here we want to mention that the actual preconditioner $\mathbf{P}_j$ is given by a simplification of a first order approximation to the Jacobian of the residual function $\mathbf{R}(\mathbf{W}^{(0)})$.

Then the residual at point i only depends on its direct neighbors (i.e., $\mathbf{R}_i = \mathbf{R}_i\left(\mathbf{W}_i, \mathbf{W}_{j,j\in\mathcal{N}(i)}\right)$). Therefore, along a line L_i the extracted part $\partial\mathbf{R}_{L_i}/\partial\mathbf{W}_{L_i}$ of the full Jacobian can be represented by a block tridiagonal matrix,

$$\frac{\partial\mathbf{R}_{L_i}}{\partial\mathbf{W}_{L_i}} = \begin{pmatrix} \frac{\partial\mathbf{R}_{\ell_i^1}}{\partial\mathbf{W}_{\ell_i^1}} & \frac{\partial\mathbf{R}_{\ell_i^1}}{\partial\mathbf{W}_{\ell_i^2}} & & & \\ \frac{\partial\mathbf{R}_{\ell_i^2}}{\partial\mathbf{W}_{\ell_i^1}} & \frac{\partial\mathbf{R}_{\ell_i^2}}{\partial\mathbf{W}_{\ell_i^2}} & \frac{\partial\mathbf{R}_{\ell_i^2}}{\partial\mathbf{W}_{\ell_i^3}} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{\partial\mathbf{R}_{\ell_i^{r_i-1}}}{\partial\mathbf{W}_{\ell_i^{r_i-2}}} & \frac{\partial\mathbf{R}_{\ell_i^{r_i-1}}}{\partial\mathbf{W}_{\ell_i^{r_i-1}}} & \frac{\partial\mathbf{R}_{\ell_i^{r_i-1}}}{\partial\mathbf{W}_{\ell_i^{r_i}}} \\ & & & \frac{\partial\mathbf{R}_{\ell_i^{r_i}}}{\partial\mathbf{W}_{\ell_i^{r_i-1}}} & \frac{\partial\mathbf{R}_{\ell_i^{r_i}}}{\partial\mathbf{W}_{\ell_i^{r_i}}} \end{pmatrix}. \tag{55}$$

Hence, to compute one stage of Algorithm (53) n block-tridiagonal linear systems (54) need to be solved. This can be efficiently done by applying for example a block LU-decomposition (see e.g. [8]).

In the case when a line degenerates to a point, which is usually the case for the isotropic part of the mesh, Algorithm (53) becomes point implicit. Hence, the line implicit preconditioned Runge-Kutta method (53) is a straightforward generalization of the algorithm presented in [10]. Moreover, specifically, a line implicit method is a hybrid method which is

a) line implicit in anisotropic parts of the mesh,

b) point implicit in isotropic parts of the mesh.

Similar line implicit methods are suggested in [1, 6, 14, 15, 16]. However, usually the derivation of these methods in these references are quite specific and do not exploit the general formulation of implicit Runge-Kutta methods.
Finally, note that in our line implicit solution strategy all lines determined by the line search algorithm are used. This remark seems to be trivial. However, for example in [6] it was mentioned that their algorithm only accepts lines of a minimum length of 10 elements. In our algorithm this does not play a role since our algorithm becomes automatically point implicit. Therefore lines of very short length can be handled.

5 Construction of the preconditioner

The construction of the preconditioner is guided by the following rules:

a) We only consider a first order discretization of the residual function, that is we neglect second order terms.

b) The dissipation operator $|\mathbf{A}_{ij}|$ in (9) is assumed to be constant.

c) The effective viscosity μ_{eff} and effective conductivity κ_{eff} (see (6)) are assumed to be constant.

For the exact inviscid contribution to the preconditioner we refer to [10]. A detailed description of the viscous contribution and the contribution with respect to the source terms is given in the Appendix, Section 8.

Therefore, the diagonal entries of the tridiagonal matrix (55) are approximated by

$$\begin{aligned}
\frac{\partial \mathbf{R}_i}{\partial \mathbf{W}_i} &\approx \sum_{j \in \mathcal{N}(i)} \left\{ \frac{1}{2} |\mathbf{A}_{ij}| - \left[\frac{\partial (\mathbf{f}_v \cdot \mathbf{n})(\mathbf{W}_{(\mathbf{p}^{(i)}, \mathbf{p}^{(j)})})}{\partial \mathbf{W}_{\mathbf{p}^{(i)}}} \right]^{\text{TSL}, \mu_{\text{eff}} = const} \right\} \\
&+ \frac{\partial \left(\mathbf{f}_{\text{bdry}} \cdot \mathbf{n} \right) (\mathbf{W}_i)}{\partial \mathbf{W}_i} - \text{vol}\,(\Omega_i) \frac{\partial \mathbf{Q}(\mathbf{W}_i, \mathbf{W}_j)}{\partial \mathbf{W}_i}. \qquad (56)
\end{aligned}$$

For an inner point i the boundary term $\mathbf{f}_{\text{bdry}} \cdot \mathbf{n}$ vanishes. However, since we consider all derivatives of the boundary terms (e.g. characteristic farfield, Euler wall) we mention them here for completeness. For example, for an inviscid calculation the derivative of the Euler boundary condition $\mathbf{f}_{\text{bdry,eul}} \cdot \mathbf{n} := (0, n_1 p, n_2 p, n_3 p, 0)$ given by

$$\frac{\partial \left(\mathbf{f}_{\text{bdry,eul}} \cdot \mathbf{n} \right) (\mathbf{W})}{\partial \mathbf{W}} = (\gamma - 1) \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{n_1 \|\mathbf{u}\|_2^2}{2} & -n_1 u_1 & -n_1 u_2 & -n_1 u_3 & n_1 \\ \frac{n_1 \|\mathbf{u}\|_2^2}{2} & -n_2 u_1 & -n_2 u_2 & -n_2 u_3 & n_2 \\ \frac{n_1 \|\mathbf{u}\|_2^2}{2} & -n_3 u_1 & -n_3 u_2 & -n_3 u_3 & n_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

is added to the preconditioner. (We left out the sixth component in the formula above because turbulence is not present in an inviscid calculation.) Moreover, in a cell vertex code the boundary condition for a boundary point i in general only depends on variables at point i. Hence, derivatives of the boundary fluxes need only be added to the diagonal of the Jacobian.

The approximate derivative of the residual with respect to a neighboring point $j \in \mathcal{N}(i)$ reads

$$\begin{aligned}
\frac{\partial \mathbf{R}_i}{\partial \mathbf{W}_j} &\approx -\frac{1}{2} |\mathbf{A}_{ij}| + \frac{1}{2} \left[\frac{\partial (\mathbf{f}_c \cdot \mathbf{n}_{ij})(\mathbf{W}_{(\mathbf{p}^{(i)}, \mathbf{p}^{(j)})})}{\partial \mathbf{W}_{\mathbf{p}^{(j)}}} \right] \\
&- \left[\frac{\partial (\mathbf{f}_v \cdot \mathbf{n}_{ij})(\mathbf{W}_{(\mathbf{p}^{(i)}, \mathbf{p}^{(j)})})}{\partial \mathbf{W}_{\mathbf{p}^{(j)}}} \right]^{\text{TSL}, \mu_{\text{eff}} = const} - \text{vol}\,(\Omega_i) \frac{\partial \mathbf{Q}_i(\mathbf{W}_i, \mathbf{W}_j)}{\partial \mathbf{W}_j}. \qquad (57)
\end{aligned}$$

The terms on the right hand side of (56) and (57) are used to approximate the block tridiagonal matrix (55) denoted in the following by $\partial \overline{\mathbf{R}}_{L_i}/\partial \mathbf{W}_{L_i}$. The linear system (54) is slightly modified into

$$\left(\frac{1}{\mathrm{CFL}_{\mathrm{impl}}} \left(\Delta t_{L_i}\right)^{-1} \mathbf{M}_{L_i} + \varepsilon_{\mathbf{P}} \alpha_{jj} \frac{\partial \overline{\mathbf{R}}_{L_i}}{\partial \mathbf{W}_{L_i}} \right) \mathbf{h}_{L_i} = \alpha_{j+1,j} \mathbf{R}_{L_i}(\mathbf{W}^{(j-1)}). \qquad (58)$$

The parameter $\varepsilon_{\mathbf{P}}$ can be used for overrelaxation and underrelaxation. It can be similarly interpreted as in [10, Section 5.1] (see also [2]). The estimate for the time step Δt_i is computed by a weighting of the largest convective and viscous eigenvalues (see e.g. [11]),

$$\begin{aligned} \Delta t_i \quad := \quad & \mathrm{vol}(\Omega_i) \left[\sum_{j \in \mathcal{N}(i)} \frac{1}{2} \left(|V_{ij}| + a_{ij} A_{ij} \right) \right. \\ & \left. + \frac{C_{visc} (\mu_{\mathrm{eff}})_{ij} A_{ij}}{\|\mathbf{p}^{(i)} - \mathbf{p}^{(j)}\|_2 \rho^{(i)}} \left(\max \left\{ \frac{4}{3}, \frac{(\kappa_{\mathrm{eff}})_{ij} (\gamma - 1)}{(\mu_{\mathrm{eff}})_{ij}} \right\} \right) \right]^{-1}, \qquad C_{visc} := 4. \end{aligned}$$

6 Numerical results

6.1 Investigation of dissipative scheme

In Section 3 we suggested many different techniques to construct the term (9). Once again we want to emphasize that in particulat this term is responsible for both accuracy and reliability of a flow simulation. Prior to the investigations it should be mentioned that the suggested methods have reached a final end. From our point of view still many things need to be considered to reach an overall satisfying formulation of the dissipative terms. The following investigationy may be viewed as a first step towards such a formulation.

6.1.1 Inviscid flow over smooth bump

We consider the inviscid flow simulation over a smooth bump. The grids are shown in Figures 4– 6. Their generation is completely described in the notes of the High-Order CFD workshop taken place at the AIAA conference January 2012. The flow conditions for the testcase are $Ma = 0.5$ and Angle of attack 0°.

First of all we want to get an idea of the order of accuracy of the discretization defined by (9). To this end first of all note that the argumentation given in (37) and (36) do not hold since we work with triangular meshes. Hence, considering the general theory of Section 3.3 we can only expect a method of order one. However, Figure 7 shows that the considered method yields on the sequence of triangular meshes a method which approximately shows a second order of accuracy. The considered error in Figure 7 is a so-called entropy error

$$\text{Error} := \left(\sum_{j=1}^{N} \text{vol}(\Omega_j)\right)^{-1/2} \left(\sum_{j=1}^{N} \text{vol}(\Omega_j) \left(\frac{\frac{p_j}{\rho_j^{\gamma}} - \frac{p_\infty}{\rho_\infty^{\gamma}}}{\frac{p_\infty}{\rho_\infty^{\gamma}}}\right)^2\right)^{1/2}, \tag{59}$$

which needs to go to 0 for an infinite fine mesh.

6.1.2 Influence of construction of dissipation at the boundary

To give an initial impression of the influence of the construction of dissipation at the boundary we consider again the smooth bump as in Section 6.1.1. Only for the finest mesh (see Figure 6) we have plotted the total pressure loss in Figure 7 on the right. Here the effect described in Section 3.6 can be observed. Dealing with a triangular mesh formulae (44) and (45) are not good approximated by the construction with the nearest neighbor point to the wall when compared with the construction where the faces connecting the boundary point and the next inner points are neglected. For the first construction a significant larger total pressure loss can be observed.

Having Figure 7 we restrict ourselves in the following to the construction of boundary dissipation by neglecting faces connecting the boundary point and the next inner points. Note that so far we have not implemented and investigated the boundary construction using formula (25). Moreover, it is still an open issue to construct a stable artificial dissipation operator in corners such as wing body junction. We assume that many reliability problems of unstructured CFD codes is due to the problem of constrcution of dissipation near the boundary.

6.1.3 Inviscid flow over NACA0012

To investigate the interplay of different parameters we investigate inviscid flow over a NACA0012 airfoil under flow conditions Ma = 0.8, AoA = 1.25°. We consider a sequence of meshes of dimension 640×128, 320×64 and 160×32. An illustration of the meshes is given in Figures 8 – 10.

Mesh: 640×128	C-drag	C-lift
$\Psi^{(1)}$, $\varepsilon_\Psi = 1$, $\xi = 0.015625$, $s^{(1)}$	$2.1425e-02$	$3.3425e-01$
$\Psi^{(1)}$, $\varepsilon_\Psi = 8$, $\xi = 0.015625$, $s^{(1)}$	$2.1484e-02$	$3.3522e-01$
$\Psi^{(1)}$, $\varepsilon_\Psi = 8$, $\xi = 0.03125$, $s^{(1)}$	$2.1502e-02$	$3.3555e-01$
$\Psi^{(2)}$, $\varepsilon_\Psi = 1$, $\xi = 0.015625$, $s^{(1)}$	$2.1417e-02$	$3.3413e-01$
$\Psi^{(2)}$, $\varepsilon_\Psi = 8$, $\xi = 0.015625$, $s^{(1)}$	$2.1426e-02$	$3.3427e-01$
$\Psi^{(2)}$, $\varepsilon_\Psi = 8$, $\xi = 0.03125$, $s^{(1)}$	$2.1458e-02$	$3.3486e-01$

Table 1: C-drag and C-lift for the mesh 640×128

Mesh: 320×64	C-drag	C-lift
$\Psi^{(1)}$, $\varepsilon_\Psi = 1$, $\xi = 0.015625$, $s^{(1)}$	$2.1493e-02$	$3.3525e-01$
$\Psi^{(1)}$, $\varepsilon_\Psi = 8$, $\xi = 0.015625$, $s^{(1)}$	$2.1614e-02$	$3.3671e-01$
$\Psi^{(1)}$, $\varepsilon_\Psi = 8$, $\xi = 0.03125$, $s^{(1)}$	$2.1638e-02$	$3.3707e-01$
$\Psi^{(2)}$, $\varepsilon_\Psi = 1$, $\xi = 0.015625$, $s^{(1)}$	$2.1474e-02$	$3.3500e-01$
$\Psi^{(2)}$, $\varepsilon_\Psi = 8$, $\xi = 0.015625$, $s^{(1)}$	$2.1500e-02$	$3.3538e-01$
$\Psi^{(2)}$, $\varepsilon_\Psi = 8$, $\xi = 0.03125$, $s^{(1)}$	$2.1546e-02$	$3.3612e-01$

Table 2: C-drag and C-lift for the mesh 320×64

To get an impression of the influence of the two different pressure switches (39a) and (39b) we make computations for both. Moreover, to see the influence of the weighting coefficient ε_ψ and ξ we choose either $\varepsilon_\psi = 1$ and $\varepsilon_\psi = 8$ and we consider $\xi = 0.015625$ and $\xi = 0.03125$. The weighting function s_{ij} is always determined by $s_{ij}^{(1)}$.

Mesh: 160×32	C-drag	C-lift
$\Psi^{(1)}$, $\varepsilon_\Psi = 1$, $\xi = 0.015625$, $s^{(1)}$	$2.1624e-02$	$3.3759e-01$
$\Psi^{(1)}$, $\varepsilon_\Psi = 8$, $\xi = 0.015625$, $s^{(1)}$	$2.1969e-02$	$3.3851e-01$
$\Psi^{(1)}$, $\varepsilon_\Psi = 8$, $\xi = 0.03125$, $s^{(1)}$	$2.2007e-02$	$3.3863e-01$
$\Psi^{(2)}$, $\varepsilon_\Psi = 1$, $\xi = 0.015625$, $s^{(1)}$	$2.1587e-02$	$3.3721e-01$
$\Psi^{(2)}$, $\varepsilon_\Psi = 8$, $\xi = 0.015625$, $s^{(1)}$	$2.1633e-02$	$3.3710e-01$
$\Psi^{(2)}$, $\varepsilon_\Psi = 8$, $\xi = 0.03125$, $s^{(1)}$	$2.1706e-02$	$3.3776-e01$

Table 3: C-drag and C-lift for the mesh 320×64

Results for C-drag and C-lift for all computations are given in the Table 1– 3. However, these values seem to carry only little information to evaluate the different discretizations and one should be very careful by evaluating the different discretizations with respect to these values. It is more important to have a closer look at the cp distributions given in Figures 11– 13. From Figure 11 it is obvious that a choice of $\varepsilon_\Psi = 1$ and $\xi = 0.015625$ is not appropriate since the cp distribution shows lots of oscillations in a neighborhood of the shock.

A first explanation for this behavior is given by Figure 14. Despite the theoretical assumption also in a neighbor of the shock neither of both pressure switches (39a) and (39b) is close to 1. Therefore the scheme (10) will never reach a first order scheme in the neighborhood of the shock. It will only deliver a small scaled first order part and will mainly be a second order scheme. Hence, oscillations in a neighborhood of the shock can be expected. Therefore, we increased the $\varepsilon_\Psi = 1$ to $\varepsilon_\Psi = 8$. Then, in a neighborhood of the shock comes closer to a first order scheme. However, by a look at Figure 12 it can be observed that the influence of increasing ε_Ψ to 8 is negligible.

So, to reduce the oscillations we finally increased the weighting factor ξ to 8. This improves the quality of the solution significantly. We want to mention that the final oscillation of Figure 13 cannot be significantly reduced by playing around with ε_Ψ and ξ. It can be avoided by modifying the pressure switch a little. However, since such an oscillation can in general not be observed in viscous computations, no modifications are included into the implementation.

Figure 13 also shows that the shock position is not significantly influenced by the choice of the pressure switch. Following the argumentation in Section 3.5 we therefore recommend to use the more local formulation (39b). Although not shown here in particular for 3d configurations it figured out that this formulation yields better reliability.

6.2 Concluding remarks with respect to construction of dissipation

Both the theory presented in Section 3 and the numerical investigations of Sections 6.1.1, 6.1.2 and 6.1.3 may be viewed as initial data points to construct an accurate and reliable scheme. We have tried to give arguments that the construction yields a stable implementation by estimating the eigenvalues of the dissipative operators. Moreover, we showed which accuracy of the scheme can be expected. However, the theoretical view does not circumvent the problem that in pratice some scaling of first and second order terms is required. Also the construction of dissipation along the boundary is still not answered satisfactory.

So, in future work additional data points are required to get an impression of the behavior of the dissipative scheme. Here in particular turbulent high Reynolds number flow simulations also on complex 3d configurations should be investigated.

Moreover, we think that further significant improvements can only be reached by extending the theory presented in Section 3 to the relevant block systems. Special focus should be in the construction of dissipation along boundaries in particular in junction areas.

6.3 Application of the line search algorithm

To demonstrate the influence of the threshold parameter γ_{line} of the line search algorithm we consider a structured RAE 2822 airfoil grid of dimension 736×176. In Figures 15 – 17 the lines determined in the neighborhood of the airfoil are shown for $\gamma_{\text{line}} = 5$ and $\gamma_{\text{line}} = 20$. Obviously, the choice $\gamma_{\text{line}} = 20$ is far more restrictive and the lines are significantly shorter.
As a second example the determined lines of a three element high lift airfoil with a structured boundary layer and unstructured farfield (see Figure 23) are shown in Figures 18 – 19 for $\gamma_{\text{line}} = 2$. The mesh has 69534 cells. Whereas the lines on the finest grid are as expected, on the coarse grids the lines deteriorate.
As a third example and to demonstrate the applicability of the line search algorithm for 3-D unstructured grids we show results for a 3-D wing-body configuration in Figure 20. The mesh has 2953483 cells. The loss as well as the deterioration of the lines on the agglomerated grid levels happens here as well.

6.4 Application of line implicit method

In all our computations we chose the Butcher schemes
The preconditioner required required for equation (58) is only constructed and inverted on the first stage of the Runge-Kutta scheme. Updating the preconditioner on every stage in general did not show any significant improvement in the conver-

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 0 & 2/3 & 0 & 0 \\ 0 & 0 & 2/3 & 0 \\ \hline & 0 & 0 & 1 \end{array} \quad \text{(expl.RK.)}, \qquad \begin{array}{c|ccc} 0 & 1 & 0 & 0 \\ 0 & 2/3 & 1 & 0 \\ 0 & 0 & 2/3 & 1 \\ \hline & 0 & 0 & 1 \end{array} \quad \text{(Algorithm (53))}.$$

Size	736×176	368×88	184×44	92×22	46×11
Max. aspect ratio	4000	3800	3500	3000	2200
Iter. Res. 10^{-12} expl. RK.	86309	24393	10829	6395	2952
Iter. Res. 10^{-12} line impl.	4852	2372	2639	942	862
CPU time expl. RK.	> 24h	2h	14 min.	120sec.	15sec.
CPU time line impl.	9.5h	1.2h	21 min.	105sec.	30sec.

Table 4: Main results for the RAE2822 testcase

gence rate whereas the CPU time is substantially increased. The parameter $\varepsilon_{\mathbf{P}}$ is always chosen as 2.

6.4.1 RAE 2822 airfoil

We apply Algorithm (53) to a sequence of five meshes for the RAE 2822 airfoil under the following conditions:

a) Onflow Mach number: 0.73

b) Angle of attack: 2.8°

c) Reynolds number: $6.5e6$

To eliminate parallelization effects in the algorithm all computations were performed on a single core. For the line implicit method the parameter $\gamma_{\text{line}} = 5.0$ was chosen. A plot of the convergence histories is given for the finest grid in Figure 21. The drag coefficient shows that it is usually not enough to reduce the density residual several orders of magnitude. For the explicit Runge-Kutta methods after about 20000 iterations the density residual is reduced about eight orders of magnitude. However, the drag coefficient after 20000 iterations is still not converged for this method. This observation again indicates that for highly stretched meshes reliable and efficient solution methods are required which approximate a solution of the discrete equation, that is, which decrease the residual to machine accuracy in an adequate number of iterations. The final drag coefficient of about $C_d = 0.01708$ corresponds to other results given in literature, see e.g. [22]. Figure 22 shows the C_p–distribution compared to measured data ([4]). The agreement is acceptable.

6.4.2 Three element high lift airfoil

Results for the three element high lift airfoil are shown in Figure 23 and Figure 24. The compuations were performed on a single core. For the line implicit method the parameter $\gamma_{\text{line}} = 4.0$ was chosen. The total CPU time was about four hours. The flow conditions of this test case are as follows:

a) Onflow Mach number: 0.22

b) Angle of attack: 21.4°

c) Reynolds number: $4.0e6$

Figure 24 shows the complexity of flow field around a high lift airfoil. On the one hand there are small transonic and supersonic regions in a neighborhoood of the slat whereas the rest of the flowfield is dominated by low Mach numbers. Moreover, one can observe that on the lower side of the slat, on the upper side of the flap and at the end of the main wing there exist regions of separation. The flow solver is required to handle all these flow features. This configuration provides an example for the gain in reliability of the line implicit Algorithm (53). The explicit Runge-Kutta method was not successful at all for this configuration whereas the line implicit method yield to a convergent solution (see Figure 23).

6.5 Helicopter fuselage

The first 3D example we consider is turbulent flow around an analytic streamlined body, which can be viewed as a generic helicopter fuselage (for details see [9, Chapter 33]). The mesh has 425984 elements and 38912 surface elements. A plot of the mesh as well as the lines for $\gamma_{\text{line}} = 10$ is given in Figure 25. The flow conditions of this test case were as follows:

a) Onflow Mach number: 0.8

b) Angle of attack: 5.0°

c) Reynolds number: $1.0e7$

The computations for this testcase were done in a parallel version of the code using twelve processes. The convergence histories of the testcase are given in Figure 26. The results are summarized in the following tabular:
Tabular 5 shows the significant superiority of the line implicit method (53) when compared with the explicit Runge-Kutta method. In CPU time the speed up was about a factor of three.

	Stopping criterion	Iterations	CPU time
expl. RK.	Res. 10^{-12}	12566	3.1h
line impl.	Res. 10^{-12}	875	0.9h

Table 5: Main results for the helicopter fuselage

6.6 ONERA M6 wing

We consider turbulent flow over the ONERA M6 wing. The mesh is purely hexahedral with 1725056 elements. The number of surface quadrilaterals is 79280. A section of the mesh is given in Figure 27. The lines used in the multigrid cycle are given in Figure 28. Moreover, in Figure 28 the convergence histories for the density and the turbulent variable residual as well as the convergence history for the lift are plotted. The flow conditions of this test case were as follows:

a) Onflow Mach number: 0.8395

b) Angle of attack: 3.06°

c) Reynolds number: $11.3e6$

Note that the explicit Runge-Kutta method in combination with a Matrix dissipative scheme (see (9)) was not successful at all. This observation goes along with the theory presented in ([10]). The computation was performed using eight processes. The total computation time to reduce the residual about nine orders of magnitude took about 33 hours.

6.7 DPW4

As last example we consider turbulent flow over a configuration used at the Drag Prediction Workshop 4. It is a wing-body configuration with horizontal tail plane (see Figure 29). For more details on this test case we refer to [28]. The mesh is hybrid with 8565419 elements and 167863 surface elements given by:

a) 5309541 tetrahedrons

b) 18384 prisms

c) 108987 pyramids

d) 3128507 hexahedrals

e) 40888 surface triangles

f) 126975 surface quadrilaterals

The flow conditions of this test case were as follows:

a) Onflow Mach number: 0.85

b) Angle of attack: 2.33548°

c) Reynolds number: $5.0e6$

Figure 29 also shows the convergence history of the density and the turbulent variable residual as well as the convergence history for the drag. Note that also in this testcase the explicit Runge-Kutta method in combination with a Matrix dissipative scheme (see (9)) was not successful at all. The computation was performed using 16 processes. The total computation time to reduce the residual about ten orders of magnitude took about 25 hours.

7 Conclusion and future work

We have presented a line implicit Runge-Kutta method. This technique was applied to fully coupled turbulent flow problems for the one equation turbulence model of Spalart and Allmaras [23]. For all parts of the equations the required approximative derivative terms have been presented as well as the complete construction of the preconditioner. The improved reliability and efficency in particular for meshes with large anisotropies and for critical flow conditions such as high lift have been demonstrated.

In our framework an agglomeration strategy built upon the same idea as the construction of the preconditioner is still missing yielding in particular on coarse grid levels within the multigrid badly shaped lines. Therefore, as one of the next steps it is planned to combine the line implicit method with a directional coarsening strategy [14, 12].

It was presented that the length and the shape of the lines are determined by some user defined parameter input γ_{line}. To make the line implicit method reliable for industrial application the influence of the length and the shape of the lines to the flow solver must be investigated. From an algorithmic point of view lines resolving the complete anisotropies in the given grid are desired. Then the method gets a more implicit behaviour. This problem is in particular from relevance for parallel applications. Domain decomposition methods are required taking care of the lines representing the anisotropies of the grid. To ensure a good load balancing correct weights need to be incorporated into the domain decomposition algorithms since the main complexity of the algorithm is to set up and solve the block tridiagonal linear systems (54).

So far, computational expensive parts namely, the solution of the block tridiagonal linear systems, rely on private implemented routines. It can be assumed that these routines are not implemented in an optimal way and should therefore be exchanged by professional tools such as LAPACK. Hence, the given CPU times in this article may be misleading and the potential of the line implicit method is even greater than indicated in this article.

8 Appendix: Required derivative terms

8.1 The viscous contribution

Instead of considering the full derivative including the derivative of the Green-Gauss gradients, we use a thin shear layer approximation for the construction of the viscous part of the preconditioner.
For a detailed computation of the derivative given below we refer to [11]. For notation only we describe the points i and j corresponding to the face ij by $i = \mathbf{p}^{(0)}$ and $j = \mathbf{p}^{(1)}$. The derivative of the viscous flux $\mathbf{f}_v \cdot \mathbf{n}$ given on the face $\mathbf{p}^{(0)}\mathbf{p}^{(1)}$ with respect to the conservative variables $\mathbf{W}_{\mathbf{p}^{(\ell)}}$, $\ell = 0, 1$ can be assembled as

$$\left[\frac{\partial(\mathbf{f}_v \cdot \mathbf{n})\left(\mathbf{W}_{\mathbf{p}^{(0)}}, \mathbf{W}_{\mathbf{p}^{(1)}}\right)}{\partial \mathbf{W}_{\mathbf{p}^{(\ell)}}}\right]^{\mathrm{TSL},\mu_{\mathrm{eff}}=const} =$$

$$\xi^{(\ell)} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ (-1)^{\ell}(N_1 u_1^{(\ell)} + \frac{1}{3}N_{12,13}^{(\ell)}) & (-1)^{\ell+1}N_1 & \frac{(-1)^{\ell+1}}{3}N_{12} & \frac{(-1)^{\ell+1}}{3}N_{13} & 0 & 0 \\ (-1)^{\ell}(N_2 u_2^{(\ell)} + \frac{1}{3}N_{12,23}^{(\ell)}) & \frac{(-1)^{\ell+1}}{3}N_{12} & (-1)^{\ell+1}N_2 & \frac{(-1)^{\ell+1}}{3}N_{23} & 0 & 0 \\ (-1)^{\ell}(N_3 u_3^{(\ell)} + \frac{1}{3}N_{13,23}^{(\ell)}) & \frac{(-1)^{\ell+1}}{3}N_{13} & \frac{(-1)^{\ell+1}}{3}N_{23} & (-1)^{\ell+1}N_3 & 0 & 0 \\ E_1^{(\ell)} + E_2^{(\ell)} + E_3^{(\ell)} & \frac{1}{\xi^{(\ell)}}\frac{\partial(\mathbf{f}_v \cdot \mathbf{n})^{(5)}}{\partial z_2^{(\ell)}} & \frac{1}{\xi^{(\ell)}}\frac{\partial(\mathbf{f}_v \cdot \mathbf{n})^{(5)}}{\partial z_3^{(\ell)}} & \frac{1}{\xi^{(\ell)}}\frac{\partial(\mathbf{f}_v \cdot \mathbf{n})^{(5)}}{\partial z_4^{(\ell)}} & \frac{\kappa_{\mathrm{eff}}(1-\gamma)}{\mu_{\mathrm{eff}}} & 0 \\ (-1)^{\ell}\frac{\tilde{\nu}^{(\ell)}}{\sigma} & 0 & 0 & 0 & 0 & \frac{(-1)^{\ell+1}}{\sigma} \end{pmatrix}$$

where

$$
\begin{aligned}
E_1^{(\ell)} &:= (-1)^\ell \left\{ u_1 \left[N_1 u_1^{(\ell)} + \frac{1}{3} N_{12} u_2^{(\ell)} + \frac{1}{3} N_{13} u_3^{(\ell)} \right] \right. \\
&+ \left. u_2 \left[\frac{1}{3} N_{12} u_1^{(\ell)} + N_2 u_2^{(\ell)} + \frac{1}{3} N_{23} u_3^{(\ell)} \right] + u_3 \left[\frac{1}{3} N_{13} u_1^{(\ell)} + \frac{1}{3} N_{23} u_2^{(\ell)} + N_3 u_3^{(\ell)} \right] \right\}, \\
E_2^{(\ell)} &:= -\frac{1}{2} \frac{\Delta}{\mu_{\text{eff}} A} \sum_{i=1}^{3} u_i^{(\ell)} \sum_{j=1}^{3} n_j \tau_{ji}, \\
E_3^{(\ell)} &:= \frac{\kappa_{\text{eff}}}{\mu_{\text{eff}}} \left(T^{(\ell)} - \frac{1}{2}(\gamma - 1) \|\mathbf{u}^{(\ell)}\|_2^2 \right) \\
\frac{\partial (\mathbf{f}_v \cdot \mathbf{n})^{(5)}}{\partial z_2^{(\ell)}} &= \frac{\mu_{\text{eff}} A}{\Delta \rho^{(\ell)}} \left\{ \frac{\Delta}{\mu_{\text{eff}} A} \frac{1}{2} \sum_{j=1}^{3} n_j \tau_{j1} + (-1)^{\ell+1} \left[N_1 u_1 + \frac{1}{3} N_{12} u_2 + \frac{1}{3} N_{13} u_3 \right] \right. \\
&\left. + (-1)^\ell \frac{\kappa_{\text{eff}}}{\mu_{\text{eff}}} (\gamma - 1)\, u_1^{(\ell)} \right\}, \\
\frac{\partial (\mathbf{f}_v \cdot \mathbf{n})^{(5)}}{\partial z_3^{(\ell)}} &= \frac{\mu_{\text{eff}} A}{\Delta \rho^{(\ell)}} \left\{ \frac{\Delta}{\mu_{\text{eff}} A} \frac{1}{2} \sum_{j=1}^{3} n_j \tau_{j2} + (-1)^{\ell+1} \left[\frac{1}{3} N_{12} u_1 + N_2 u_2 + \frac{1}{3} N_{23} u_3 \right] \right. \\
&\left. + (-1)^\ell \frac{\kappa_{\text{eff}}}{\mu_{\text{eff}}} (\gamma - 1)\, u_2^{(\ell)} \right\}, \\
\frac{\partial (\mathbf{f}_v \cdot \mathbf{n})^{(5)}}{\partial z_4^{(\ell)}} &= \frac{\mu_{\text{eff}} A}{\Delta \rho^{(\ell)}} \left\{ \frac{\Delta}{\mu_{\text{eff}} A} \frac{1}{2} \sum_{j=1}^{3} n_j \tau_{j3} + (-1)^{\ell+1} \left[\frac{1}{3} N_{13} u_1 + \frac{1}{3} N_{23} u_2 + N_3 u_3 \right] \right. \\
&\left. + (-1)^\ell \frac{\kappa_{\text{eff}}}{\mu_{\text{eff}}} (\gamma - 1)\, u_3^{(\ell)} \right\}, \\
N_{12,13}^{(\ell)} &:= N_{12} u_2^{(\ell)} + N_{13} u_3^{(\ell)}, \qquad N_{12,23}^{(\ell)} := N_{12} u_1^{(\ell)} + N_{23} u_3^{(\ell)}, \\
N_{13,23}^{(\ell)} &:= N_{13} u_1^{(\ell)} + N_{23} u_3^{(\ell)}, \qquad \xi^{(\ell)} := \frac{\mu_{\text{eff}} A}{\Delta\left(\mathbf{p}^{(1)}, \mathbf{p}^{(0)}\right)} \frac{1}{\rho^{(\ell)}}, \\
N_1 &:= \frac{4}{3} \overline{n}_1^2 + \overline{n}_2^2 + \overline{n}_3^2, \qquad N_{12} := \overline{n}_1 \overline{n}_2, \qquad N_{13} := \overline{n}_1 \overline{n}_3, \\
N_2 &:= \overline{n}_1^2 + \frac{4}{3} \overline{n}_2^2 + \overline{n}_3^2, \qquad N_{23} := \overline{n}_2 \overline{n}_3, \\
N_3 &:= \overline{n}_1^2 + \overline{n}_2^2 + \frac{4}{3} \overline{n}_3^2, \qquad \Delta := \Delta\left(\mathbf{p}^{(1)}, \mathbf{p}^{(0)}\right) := \|\mathbf{p}^{(1)} - \mathbf{p}^{(0)}\|_2.
\end{aligned}
$$

8.2 Contribution of the source terms

We rewrite Sutherland's law (7) as

$$
\mu_l = \mu_{l,\infty} \left(\frac{T}{T_\infty} \right)^{3/2} \frac{1 + \frac{\bar{T}}{T_\infty}}{\frac{T}{T_\infty} + \frac{\bar{T}}{T_\infty}}.
$$

Usually $\bar{T}$ and T_∞ are user defined parameters with default values

$$\begin{aligned} \bar{T} &:= 110.4\text{K} \qquad \text{(Sutherland's constant)}, \\ T_\infty &:= 273.15\text{K} \qquad \text{(Reference temperature)}. \end{aligned}$$

Working with non dimensionalized variables the temperature T is replaced by $T_{\text{nd}} := T/T_\infty$ and Sutherland's law reads

$$\mu_l = \mu_{l,\infty} T_{\text{nd}}^{3/2} \left(\frac{1 + C_{\text{suth}}}{T_{\text{nd}} + C_{\text{suth}}} \right), \qquad C_{\text{suth}} = \frac{\bar{T}}{T_\infty}. \tag{60}$$

For the rest of the paper Sutherland's law is always used in its non dimensionalized form (60) and therefore we will skip the subscript on the non dimensionalized temperature T_{nd}.

Note that $\mu_{l,\infty}$ is a predetermined value and therefore constant. The derivative of Sutherland's law can be computed as

$$\frac{\partial \mu_l}{\partial \mathbf{W}} = \mu_{l,\infty} \left\{ \frac{\sqrt{T}}{2} \frac{(1 + C_{\text{suth}})(T + 3C_{\text{suth}})}{(T + C_{\text{suth}})^2} \right\} \frac{\partial T}{\partial \mathbf{W}}.$$

Using the representation $T = p/\rho$ we find

$$\frac{\partial T}{\partial \mathbf{W}} = \frac{\partial (p/\rho)}{\partial \mathbf{W}} = \frac{1}{\rho^2} \left[\rho \frac{\partial p}{\partial \mathbf{W}} - p \frac{\partial \rho}{\partial \mathbf{W}} \right] = \frac{1}{\rho} \frac{\partial p}{\partial \mathbf{W}} - \frac{T}{\rho} \frac{\partial \rho}{\partial \mathbf{W}}.$$

The formulas

$$\frac{\partial p}{\partial \mathbf{W}} = (\gamma - 1) \left(\frac{\|\mathbf{u}\|_2^2}{2}, -u_1, -u_2, -u_3, 1, 0 \right)^T \quad \text{and} \quad \frac{\partial \rho}{\partial \mathbf{W}} = (1, 0, 0, 0, 0, 0)$$

conclude the derivative of Sutherland's law. In the following we will list all required derivatives with respect to the source terms of the turbulence model. We have

$$\frac{\partial \chi}{\partial \mathbf{W}} = \frac{\partial \left(\frac{\rho \tilde{\nu}}{\mu_l} \right)}{\partial \mathbf{W}} = \frac{\mu_l \frac{\partial (\rho \tilde{\nu})}{\partial \mathbf{W}} - \rho \tilde{\nu} \frac{\partial \mu_l}{\partial \mathbf{W}}}{\mu_l^2} = \frac{1}{\mu_l} \left(\frac{\partial (\rho \tilde{\nu})}{\partial \mathbf{W}} - \chi \frac{\partial \mu_l}{\partial \mathbf{W}} \right)$$

and $\frac{\partial (\rho \tilde{\nu})}{\partial \mathbf{W}} = (0, 0, 0, 0, 0, 1)$. Therefore, we get the following derivatives:

$$\begin{aligned} \frac{\partial f_{\text{v}_1}}{\partial \mathbf{W}} &= \frac{1}{(\chi^3 + c_{\text{v}_1}^3)^2} \left[\left(\chi^3 + c_{\text{v}_1}^3 \right) 3\chi^2 \frac{\partial \chi}{\partial \mathbf{W}} - 3\chi^5 \frac{\partial \chi}{\partial \mathbf{W}} \right] = \frac{3c_{\text{v}_1}^3 \chi^2}{(\chi^3 + c_{\text{v}_1}^3)^2} \frac{\partial \chi}{\partial \mathbf{W}}, \\ \frac{\partial f_{\text{v}_2}}{\partial \mathbf{W}} &= -\frac{1}{(1 + \chi f_{\text{v}_1})^2} \left[(1 + \chi f_{\text{v}_1}) \frac{\partial \chi}{\partial \mathbf{W}} - \chi \left(f_{\text{v}_1} \frac{\partial \chi}{\partial \mathbf{W}} + \chi \frac{\partial f_{\text{v}_1}}{\partial \mathbf{W}} \right) \right] \\ &= \frac{1}{(1 + \chi f_{\text{v}_1})^2} \left(\chi^2 \frac{\partial f_{\text{v}_1}}{\partial \mathbf{W}} - \frac{\partial \chi}{\partial \mathbf{W}} \right), \\ \frac{\partial f_{\text{t}_2}}{\partial \mathbf{W}} &= -2c_{\text{t}_3} c_{\text{t}_4} \exp\left(-c_{\text{t}_4} \chi^2 \right) \chi \frac{\partial \chi}{\partial \mathbf{W}} = -2c_{\text{t}_4} f_{\text{t}_2} \chi \frac{\partial \chi}{\partial \mathbf{W}}. \end{aligned}$$

To compute the derivative of $\tilde{S}$ we start with the derivative of the Green-Gauss gradients (46) applied to the velocities. Denoting the conservative variables by $(z_1, z_2, z_3, z_4, z_5, z_6) := (\rho, \rho u_1, \rho u_2, \rho u_3, \rho E, \rho\tilde{\nu})$ we have

$$\begin{aligned}\left(\frac{\partial u_{i-1}^{(k)}}{\partial x_j}\right)^{\mathrm{GG}} &= \frac{1}{\mathrm{vol}(\Omega_k)} \sum_{n\in\mathcal{N}(k)} \frac{(n_{kn})_j}{2}\left(u_{i-1}^{(k)} + u_{i-1}^{(n)}\right)\\ &= \frac{1}{\mathrm{vol}(\Omega_k)} \sum_{n\in\mathcal{N}(k)} \frac{(n_{kn})_j}{2}\left(\frac{z_i^{(k)}}{z_1^{(k)}} + \frac{z_i^{(n)}}{z_1^{(n)}}\right).\end{aligned}$$

Therefore, we have for $i = 1, 2, 3$, $\ell = 2, 3, 4$,

$$\frac{\partial}{\partial z_1^{(\ell)}}\left(\frac{\partial u_i^{(k)}}{\partial x_j}\right)^{\mathrm{GG}} = \frac{1}{\mathrm{vol}(\Omega_k)}\begin{cases} \sum_{n\in\mathcal{N}(k)} -\frac{(n_{kn})_j}{2}\frac{u_i^{(k)}}{\rho^{(k)}}, & \ell = k,\\ -\frac{(n_{k\ell})_j}{2}\frac{u_i^{(\ell)}}{\rho^{(\ell)}}, & \ell \in \mathcal{N}(k),\\ 0 & \ell \neq k, \ell \notin \mathcal{N}(k),\end{cases}$$

$$\frac{\partial}{\partial z_m^{(\ell)}}\left(\frac{\partial u_i^{(k)}}{\partial x_j}\right)^{\mathrm{GG}} = \frac{1}{\mathrm{vol}(\Omega_k)}\begin{cases} \sum_{n\in\mathcal{N}(k)} \frac{(n_{kn})_j}{2}\frac{1}{\rho^{(k)}}, & \ell = k,\\ \frac{(n_{k\ell})_j}{2}\frac{1}{\rho^{(\ell)}}, & \ell \in \mathcal{N}(k),\\ 0 & \ell \neq k, \ell \notin \mathcal{N}(k),\end{cases}$$

and

$$\frac{\partial}{\partial z_5^{(\ell)}}\left(\frac{\partial u_i^{(k)}}{\partial x_j}\right)^{\mathrm{GG}} = \frac{\partial}{\partial z_6^{(\ell)}}\left(\frac{\partial u_i^{(k)}}{\partial x_j}\right)^{\mathrm{GG}} = 0.$$

Since

$$\begin{aligned}\|\mathrm{curl}\ \mathbf{u}\|_2 &= \left\{\left[\left(\frac{\partial u_3}{\partial x_2}\right)^{\mathrm{GG}} - \left(\frac{\partial u_2}{\partial x_3}\right)^{\mathrm{GG}}\right]^2 + \left[\left(\frac{\partial u_1}{\partial x_3}\right)^{\mathrm{GG}} - \left(\frac{\partial u_3}{\partial x_1}\right)^{\mathrm{GG}}\right]^2\right.\\ &+ \left.\left[\left(\frac{\partial u_2}{\partial x_1}\right)^{\mathrm{GG}} - \left(\frac{\partial u_1}{\partial x_2}\right)^{\mathrm{GG}}\right]^2\right\}^{1/2}\end{aligned}$$

we get

$$\begin{aligned}\frac{\partial\|\mathrm{curl}\ \mathbf{u}\|_2}{\partial \mathbf{W}} &= \frac{1}{\|\mathrm{curl}\ \mathbf{u}\|_2}\left\{\left[\left(\frac{\partial u_3}{\partial x_2}\right)^{\mathrm{GG}} - \left(\frac{\partial u_2}{\partial x_3}\right)^{\mathrm{GG}}\right]\frac{\partial}{\partial \mathbf{W}}\left[\left(\frac{\partial u_3}{\partial x_2}\right)^{\mathrm{GG}} - \left(\frac{\partial u_2}{\partial x_3}\right)^{\mathrm{GG}}\right]\right.\\ &+ \left[\left(\frac{\partial u_1}{\partial x_3}\right)^{\mathrm{GG}} - \left(\frac{\partial u_3}{\partial x_1}\right)^{\mathrm{GG}}\right]\frac{\partial}{\partial \mathbf{W}}\left[\left(\frac{\partial u_1}{\partial x_3}\right)^{\mathrm{GG}} - \left(\frac{\partial u_3}{\partial x_1}\right)^{\mathrm{GG}}\right]\\ &+ \left.\left[\left(\frac{\partial u_2}{\partial x_1}\right)^{\mathrm{GG}} - \left(\frac{\partial u_1}{\partial x_2}\right)^{\mathrm{GG}}\right]\frac{\partial}{\partial \mathbf{W}}\left[\left(\frac{\partial u_2}{\partial x_1}\right)^{\mathrm{GG}} - \left(\frac{\partial u_1}{\partial x_2}\right)^{\mathrm{GG}}\right]\right\}.\end{aligned}$$

Hence, the derivative of $\tilde{S}, r, g$ and f_{w} is computed by

$$
\begin{aligned}
\frac{\partial \tilde{S}}{\partial \mathbf{W}} &= \frac{\partial \|\mathrm{curl}\ \mathbf{u}\|_2}{\partial \mathbf{W}} + \frac{1}{\kappa^2 d^2}\left(f_{\mathrm{v}_2}\frac{\partial \tilde{\nu}}{\partial \mathbf{W}} + \tilde{\nu}\frac{\partial f_{\mathrm{v}_2}}{\partial \mathbf{W}}\right), \\
\frac{\partial r}{\partial \mathbf{W}} &= \frac{1}{\kappa^2 d^2 \tilde{S}^2}\left(\tilde{S}\frac{\partial \tilde{\nu}}{\partial \mathbf{W}} - \tilde{\nu}\frac{\partial \tilde{S}}{\partial \mathbf{W}}\right), \\
\frac{\partial g}{\partial \mathbf{W}} &= \frac{\partial r}{\partial \mathbf{W}} + c_{\mathrm{w}_2}\left[6r^5\frac{\partial r}{\partial \mathbf{W}} - \frac{\partial r}{\partial \mathbf{W}}\right], \\
\frac{\partial f_{\mathrm{w}}}{\partial \mathbf{W}} &= \left[\frac{1+c_{\mathrm{w}_3}^6}{g^6+c_{\mathrm{w}_3}^6}\right]^{1/6}\frac{\partial g}{\partial \mathbf{W}} + \frac{g}{6}\left[\frac{1+c_{\mathrm{w}_3}^6}{g^6+c_{\mathrm{w}_3}^6}\right]^{-5/6}\left(-\frac{6(1+c_{\mathrm{w}_3}^6)g^5\frac{\partial g}{\partial \mathbf{W}}}{(g^6+c_{\mathrm{w}_3}^6)^2}\right).
\end{aligned}
$$

Finally, the derivatives of production and destruction can be computed by

$$
\begin{aligned}
\frac{\partial \mathbf{Pr}}{\partial \mathbf{W}} &= c_{\mathrm{b}_1}\left[-\tilde{S}\tilde{\nu}\frac{\partial f_{\mathrm{t}_2}}{\partial \mathbf{W}} + (1-f_{\mathrm{t}_2})\tilde{\nu}\frac{\partial \tilde{S}}{\partial \mathbf{W}} + (1-f_{\mathrm{t}_2})\tilde{S}\frac{\partial \tilde{\nu}}{\partial \mathbf{W}}\right], \\
\frac{\partial \mathbf{De}}{\partial \mathbf{W}} &= \left(\frac{\tilde{\nu}}{d}\right)^2\left(c_{\mathrm{w}_1}\frac{\partial f_{\mathrm{w}}}{\partial \mathbf{W}} - \frac{c_{\mathrm{b}_1}}{\kappa^2}\frac{\partial f_{\mathrm{t}_2}}{\partial \mathbf{W}}\right) + \left(c_{\mathrm{w}_1}f_{\mathrm{w}} - \frac{c_{\mathrm{b}_1}}{\kappa^2}f_{\mathrm{t}_2}\right)\frac{2\tilde{\nu}}{d^2}\frac{\partial \tilde{\nu}}{\partial \mathbf{W}}.
\end{aligned}
$$

To get a reliable solution algorithm care has to be taken by incorporating parts of the derivative of the source terms into the preconditioner. To get a robust implementation in the production part it was necessary to replace

$$
\rho\tilde{\nu}\frac{\partial \|\mathrm{curl}\ \mathbf{u}\|_2}{\partial \mathbf{W}} \qquad \text{by} \qquad \max\left\{\rho\tilde{\nu}\frac{\partial \|\mathrm{curl}\ \mathbf{u}\|_2}{\partial \mathbf{W}}, 0\right\}
$$

and in the destruction part to replace

$$
\rho\tilde{\nu}^2\frac{\partial f_{\mathrm{w}}}{\partial \mathbf{W}} \qquad \text{by} \qquad \max\left\{\rho\tilde{\nu}^2\frac{\partial f_{\mathrm{w}}}{\partial \mathbf{W}}, 0\right\}.
$$

For the derivative of the diffusion part **Di** we compute

$$
\frac{\partial}{\partial z_1^{(\ell)}}\left(\frac{\partial \tilde{\nu}^{(k)}}{\partial x_j}\right)^{\mathrm{GG}} = \frac{1}{\mathrm{vol}(\Omega_k)}\begin{cases} \sum_{n\in\mathcal{N}(k)} -\frac{(n_{kn})_j}{2}\frac{\tilde{\nu}^{(k)}}{\rho^{(k)}}, & \ell = k, \\ -\frac{(n_{k\ell})_j}{2}\frac{\tilde{\nu}^{(\ell)}}{\rho^{(\ell)}}, & \ell\in\mathcal{N}(k), \\ 0 & \ell\neq k, \ell\notin\mathcal{N}(k), \end{cases}
$$

and

$$
\frac{\partial}{\partial z_6^{(\ell)}}\left(\frac{\partial \tilde{\nu}^{(k)}}{\partial x_j}\right)^{\mathrm{GG}} = \frac{1}{\mathrm{vol}(\Omega_k)}\begin{cases} \sum_{n\in\mathcal{N}(k)} \frac{(n_{kn})_j}{2}\frac{1}{\rho^{(k)}}, & \ell = k, \\ \frac{(n_{k\ell})_j}{2}\frac{1}{\rho^{(\ell)}}, & \ell\in\mathcal{N}(k), \\ 0 & \ell\neq k, \ell\notin\mathcal{N}(k). \end{cases}
$$

The additional derivatives satisfy

$$\frac{\partial}{\partial z_i^{(\ell)}}\left(\frac{\partial \tilde{\nu}^{(k)}}{\partial x_j}\right)^{\mathrm{GG}} = 0, \qquad i = 2, 3, 4, 5.$$

Then the derivative of the diffusion is given by

$$\frac{\partial \mathbf{Di}}{\partial \mathbf{W}} = \frac{2c_{\mathrm{b}}}{\sigma}\sum_{j=1}^{3}\left(\frac{\partial \tilde{\nu}}{\partial x_j}\right)^{GG}\frac{\partial}{\partial \mathbf{W}}\left(\frac{\partial \tilde{\nu}}{\partial x_j}\right)^{GG}.$$

Finally, note that only the terms curl $(\mathbf{u})$ and $\mathbf{Di}$ require gradient information and depend therefore not only on the variables $\mathbf{W}_i$, but also on all neighbor variables $\mathbf{W}_{j,j\in\mathcal{N}(i)}$. This means that off-diagonal terms with respect to the source terms need to be considered in the block tridiagonal matrix (55). But only terms arising from

$$\frac{\partial \tilde{S}(\mathbf{W}_i)}{\partial \mathbf{W}_j} = \frac{\partial \|\text{curl } \mathbf{u}^{(i)}\|_2}{\partial \mathbf{W}_j} \quad \text{and} \quad \frac{\partial \mathbf{Di}(\mathbf{W}_i)}{\partial \mathbf{W}_j}, \qquad j \in \mathcal{N}(i),$$

produce off-diagonal terms.

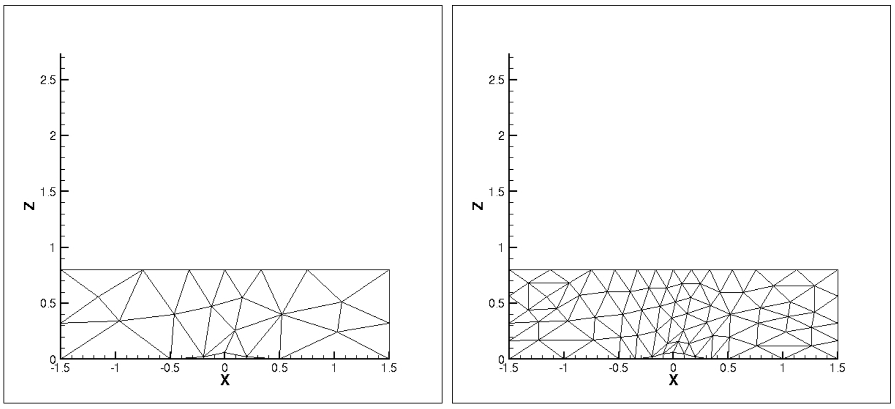

Figure 4: Bump grid: No. of elements 25 (left), No. of elements 81 (right)

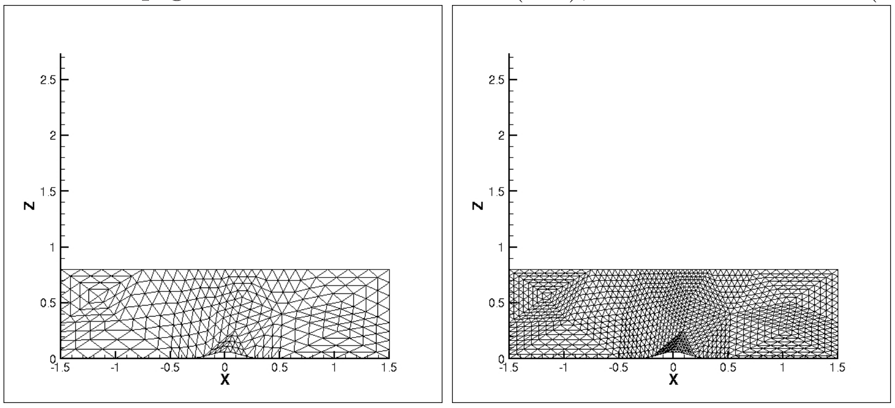

Figure 5: Bump grid: No. of elements 289 (left), No. of elements 1089 (right)

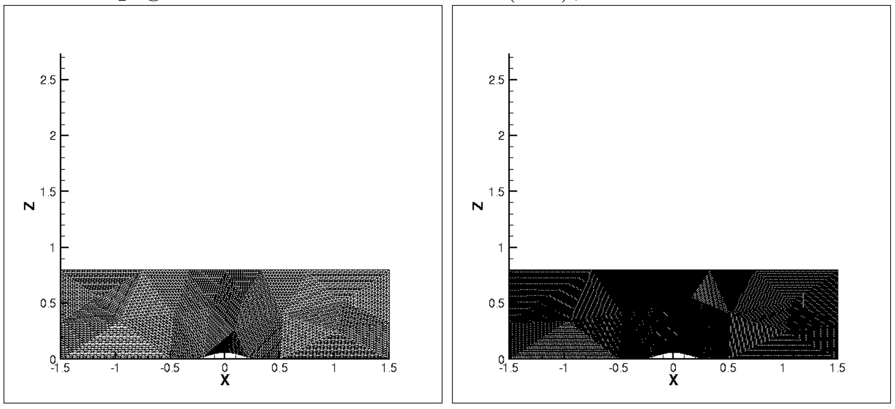

Figure 6: Bump grid: No. of elements 4225 (left), No. of elements 16641 (right)

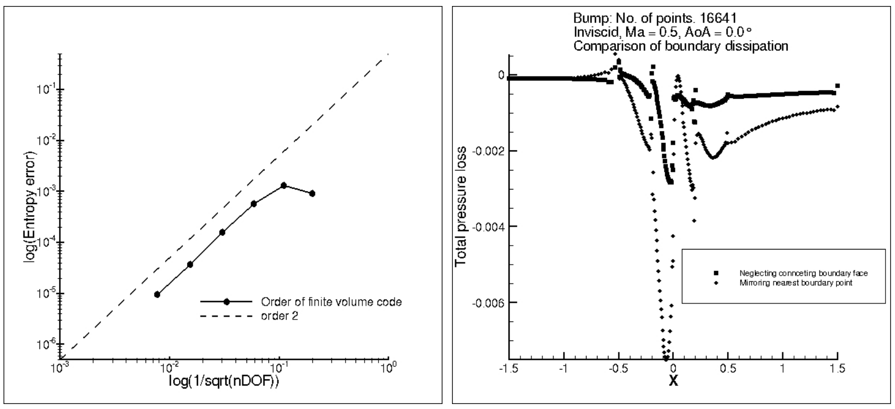

Figure 7: Left: Order of accuracy with respect to entropy error (59), Right: Comparison of total pressure loss for construction of boundary dissipation by nearest point to wall and neglecting of boundary faces

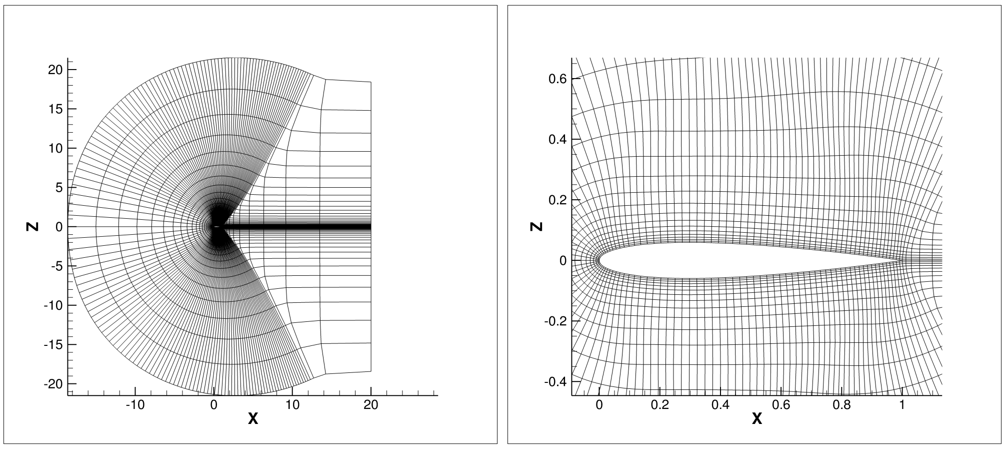

Figure 8: NACA0012 Euler grid 160 × 32: total view (left), neighborhood of the airfoil(right)

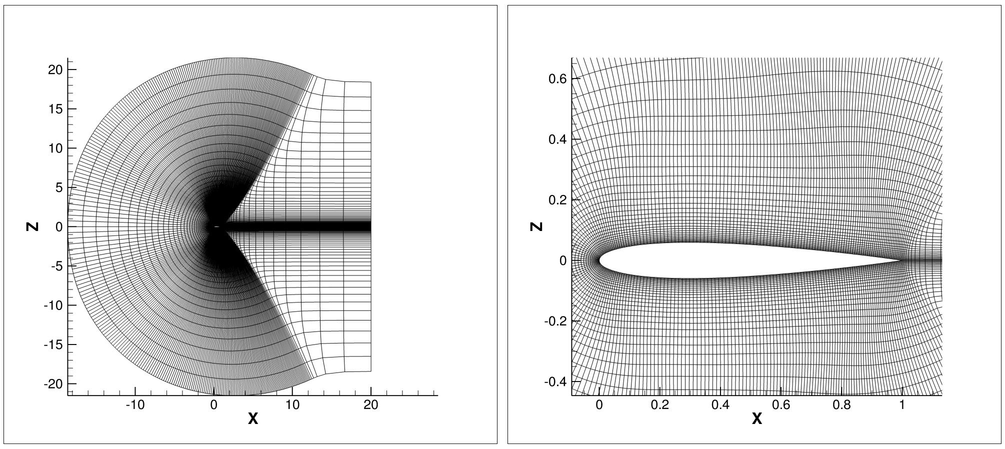

Figure 9: NACA0012 Euler grid 320 × 64: total view (left), neighborhood of the airfoil(right)

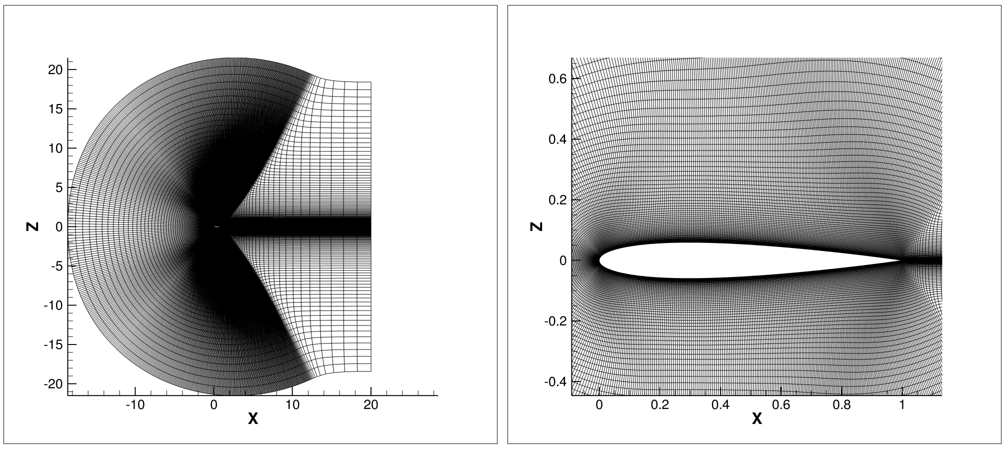

Figure 10: NACA0012 Euler grid 640 × 128: total view (left), neighborhood of the airfoil(right)

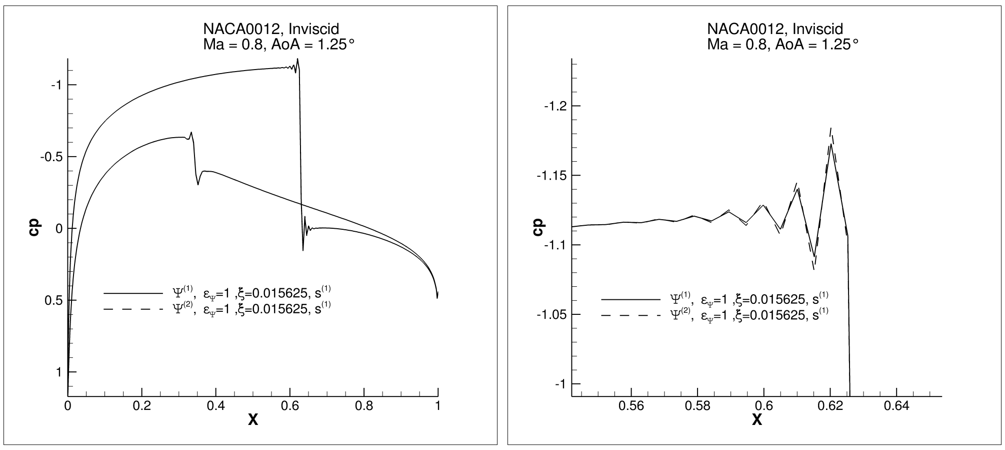

Figure 11: NACA0012 Euler grid 640 × 128: cp distribution, $\varepsilon_\Psi = 1$, $\xi = 0.015625$

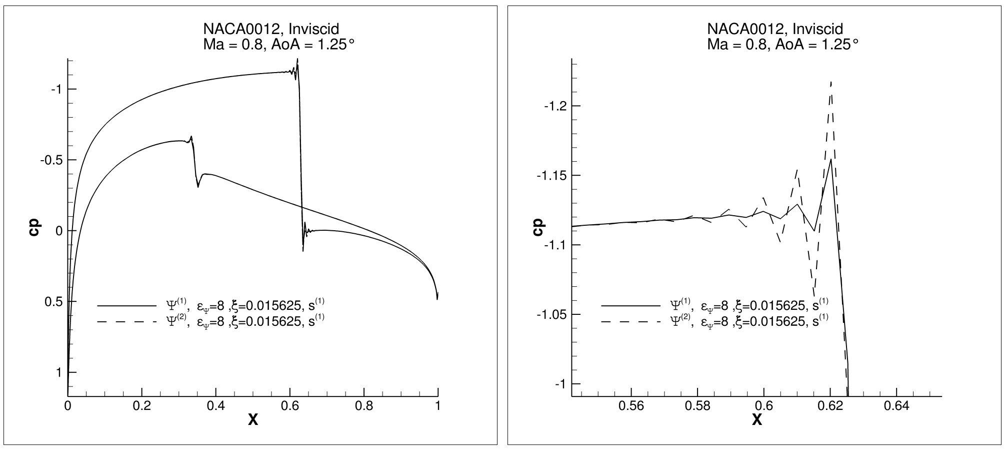

Figure 12: NACA0012 Euler grid 640 × 128: cp distribution, $\varepsilon_\Psi = 8$, $\xi = 0.015625$

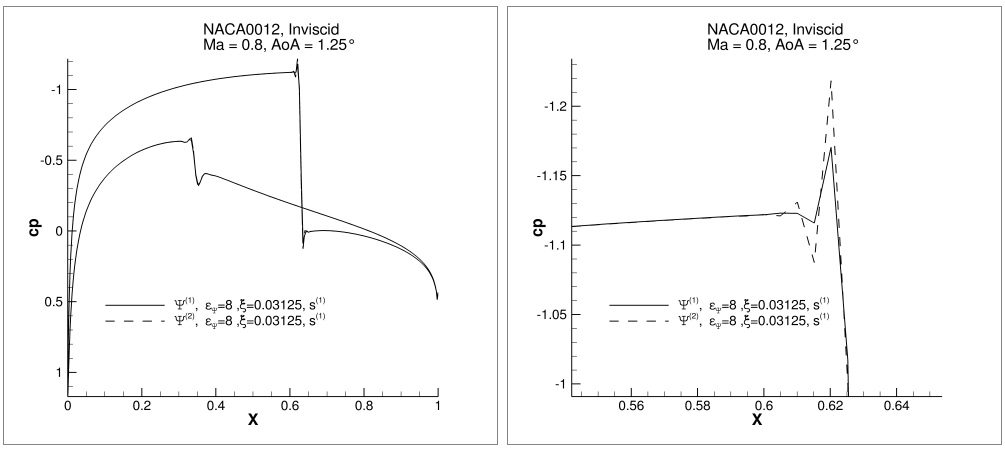

Figure 13: NACA0012 Euler grid 640 × 128: cp distribution, $\varepsilon_\Psi = 8$, $\xi = 0.03125$

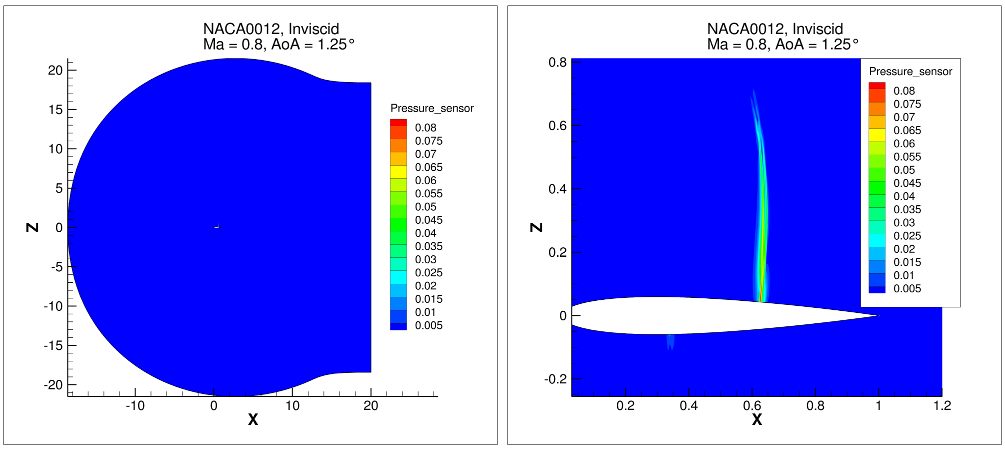

Figure 14: Computed pressure sensor by formula $\Psi_i := \left| \frac{\sum_{j\in\mathcal{N}(i)}(p_j - p_i)}{\sum_{j\in\mathcal{N}(i)}(p_j + p_i)} \right|$

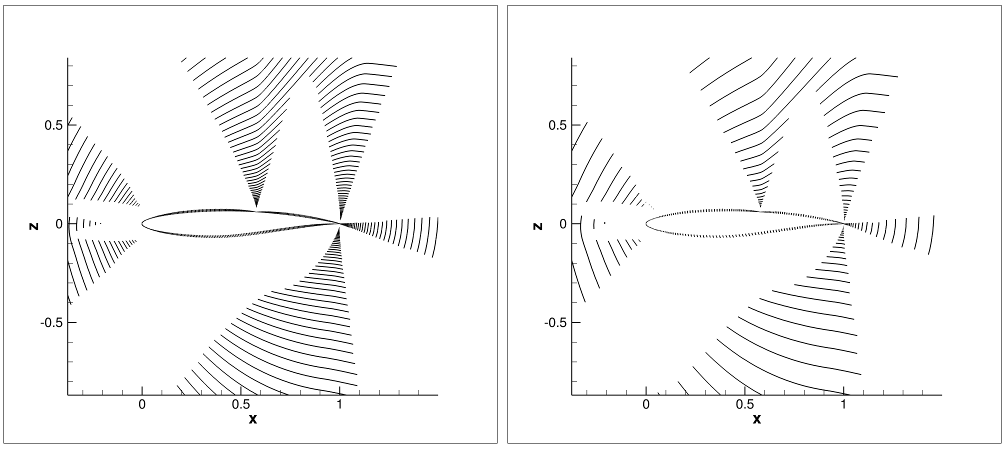

Figure 15: RAE2822: Determined lines in the neighborhood of the airfoil on the finest grid (left) and on the first agglomerated grid (right), $\gamma_{\text{line}} = 5.0$

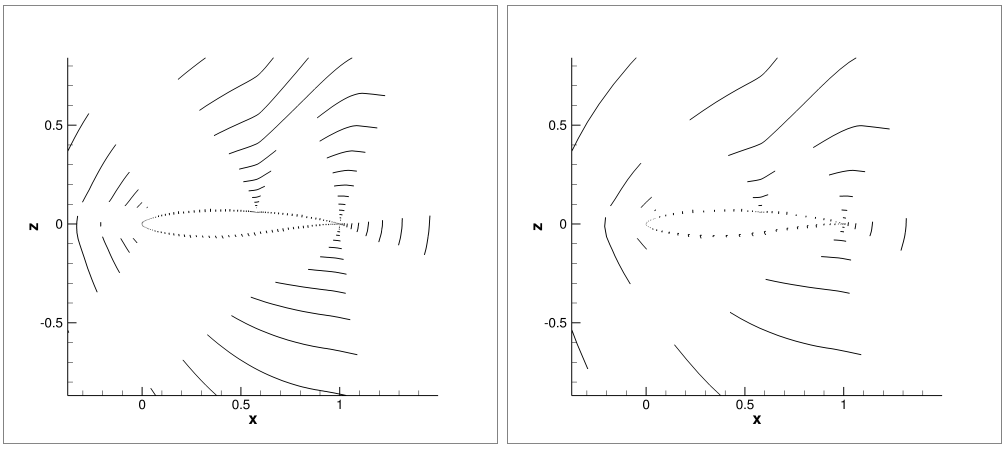

Figure 16: RAE2822: Determined lines in the neighborhood of the airfoil on the second agglomerated grid (left) and on the third agglomerated grid (right) $\gamma_{\text{line}} = 5.0$

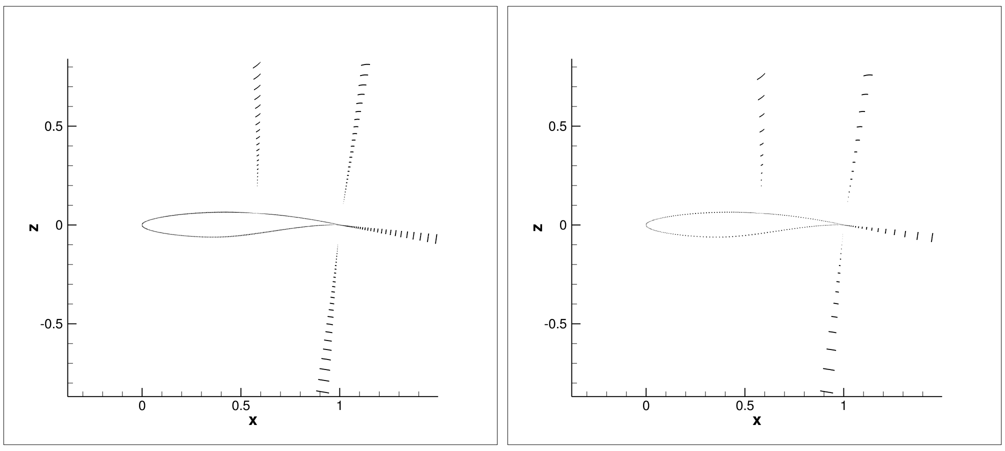

Figure 17: RAE2822: Determined lines in the neighborhood of the airfoil on the finest grid (left) and on the first agglomerated grid (right), $\gamma_{\text{line}} = 20.0$

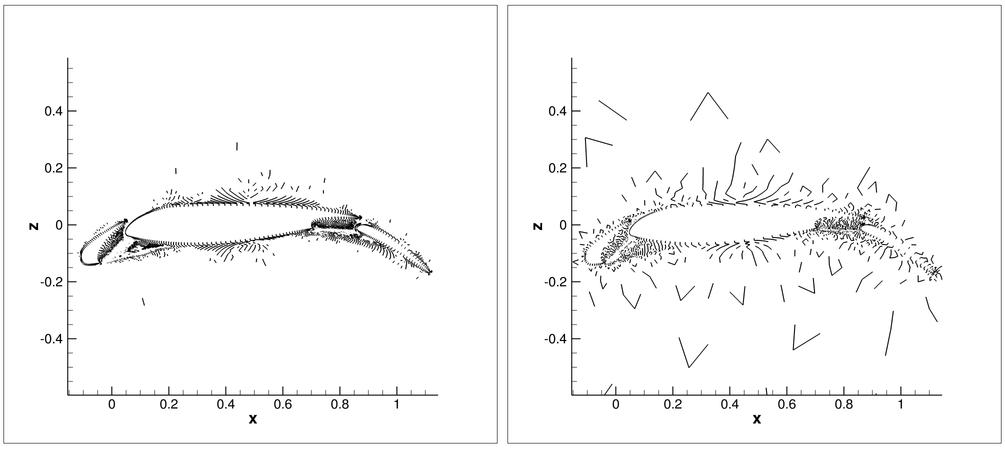

Figure 18: High lift airfoil: Determined lines in the neighborhood of the airfoil on the finest grid (left) and on the first agglomerated grid (right), $\gamma_{\text{line}} = 2.0$

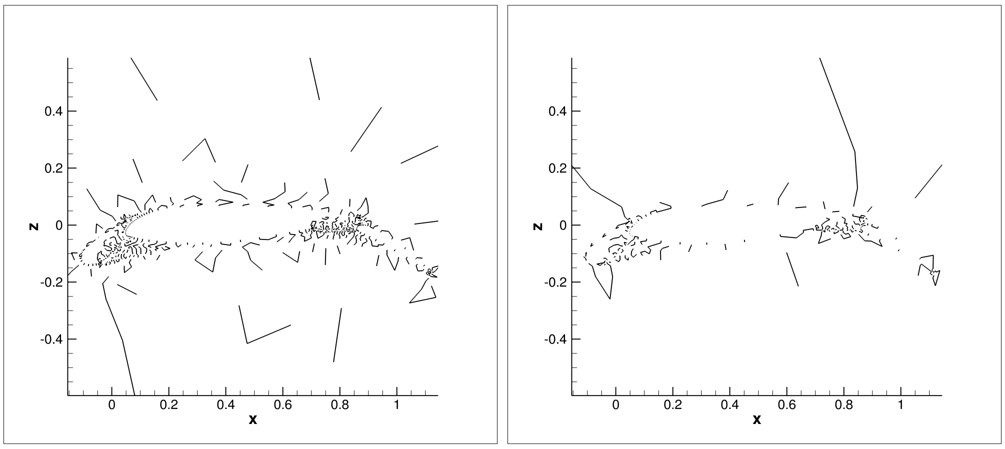

Figure 19: High lift airfoil: Determined lines in the neighborhood of the airfoil on the second agglomerated grid (left) and on the third agglomerated grid (right) $\gamma_{\text{line}} = 2.0$

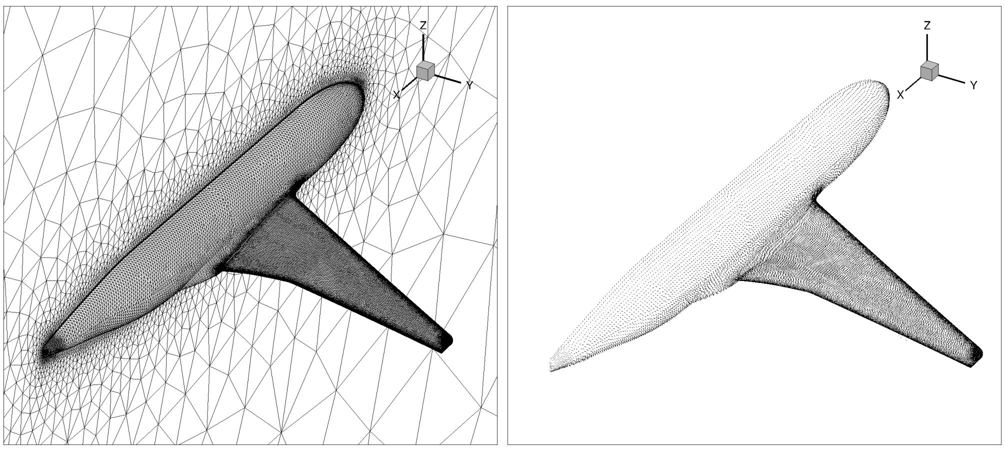

Figure 20: Wing-body configuration: Unstructured mesh with prismatic boundary layer(left), determined lines on the finest grid (right), $\gamma_{\text{line}} = 10.0$

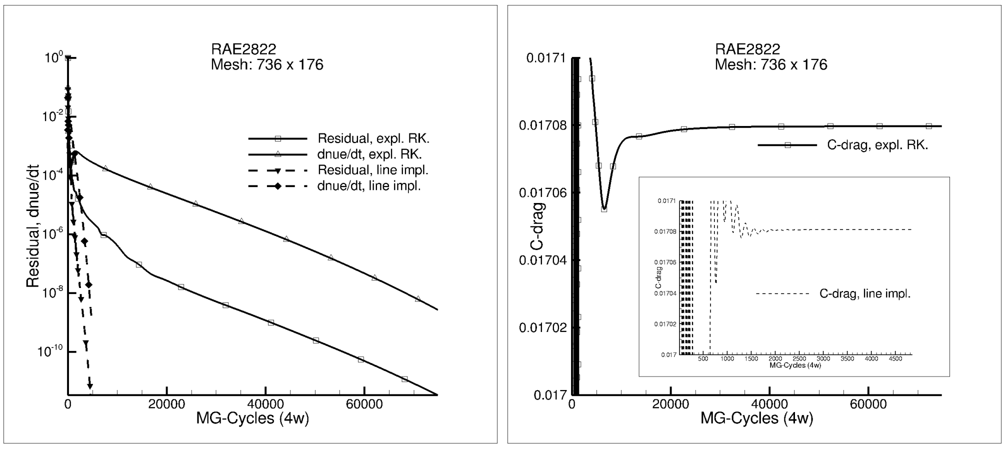

Figure 21: RAE2822: Comparison of density residual and turbulent variable residual (left) and drag (right) for the expl. RK. and the line implicit method. The computation of the line implicit method were performed with $\gamma_{\text{line}} = 5.0$.

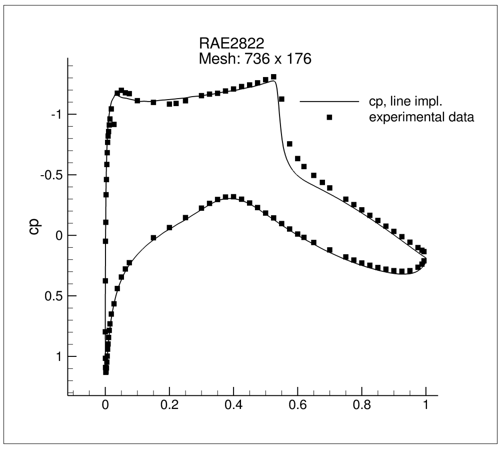

Figure 22: RAE2822: Comparison of computed Cp–distribution on the finest mesh of 736×176 and measured data [4]

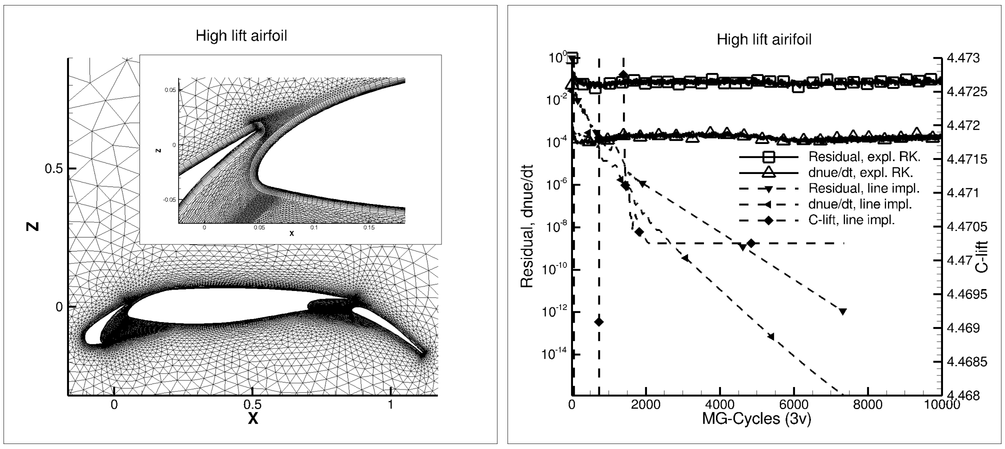

Figure 23: High lift airfoil: Section of the grid (left), Comparison of density residual and turbulent variable residual for the expl. RK. and the line implicit method (right). The computation of the line implicit method were performed with $\gamma_{\text{line}} = 2.0$

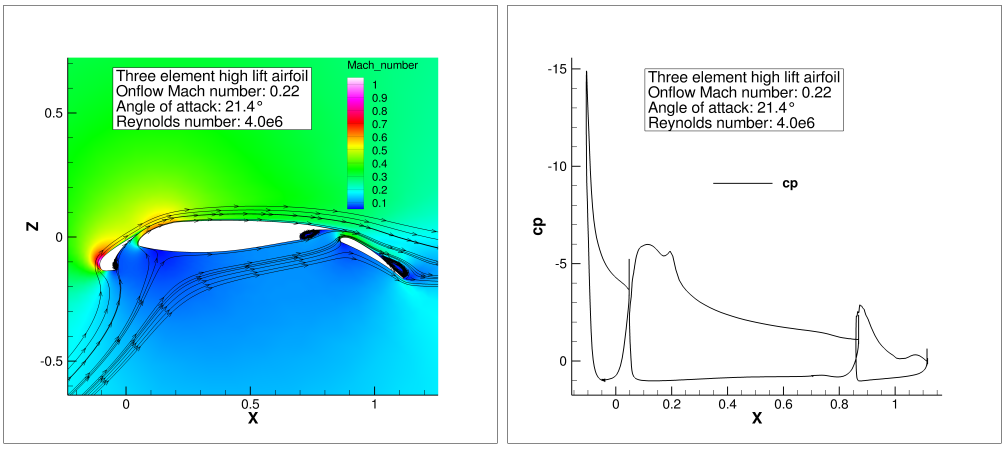

Figure 24: High lift airfoil: Mach number and velocity stream traces (left), C_p distribution (right)

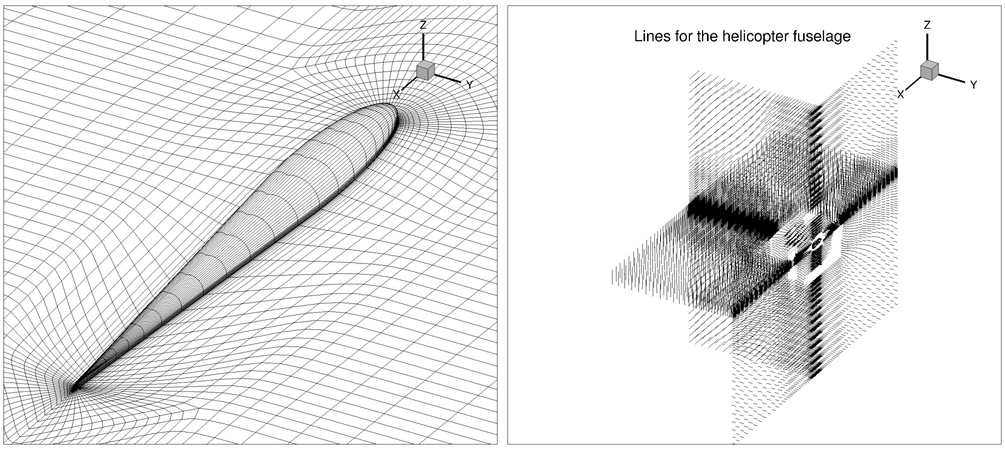

Figure 25: Helicopter fuselage: Section of the grid (left), determined lines on the finest grid $\gamma_{\text{line}} = 10$(right)

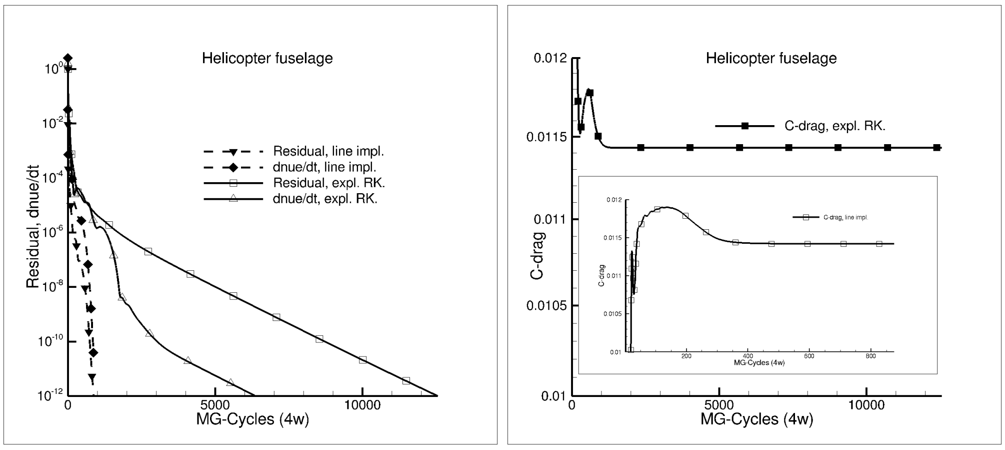

Figure 26: Helicopter fuselage: Comparison of density residual and turbulent variable residual (left) and drag (right) for the expl. RK. and the line implicit method

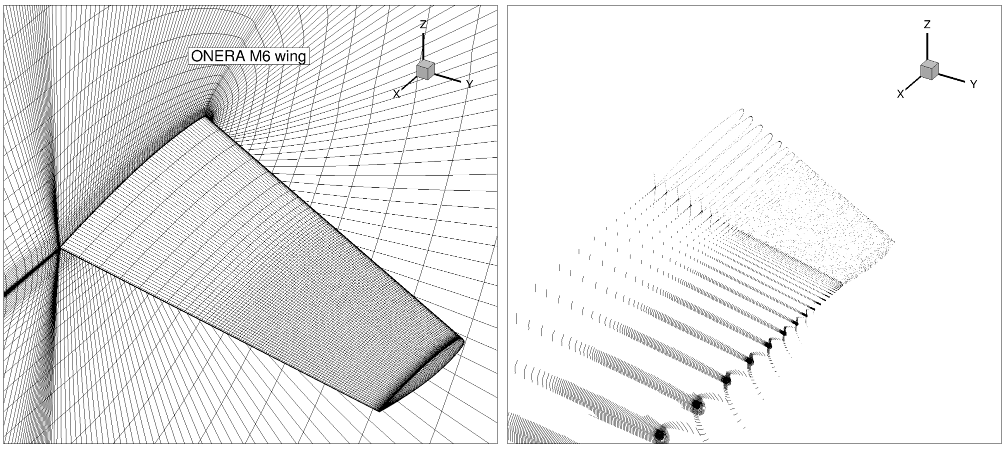

Figure 27: ONERA M6 wing: Section of the grid (left), determined lines on the finest grid in the neighborhood of the wing, $\gamma_{\text{line}} = 100$(right)

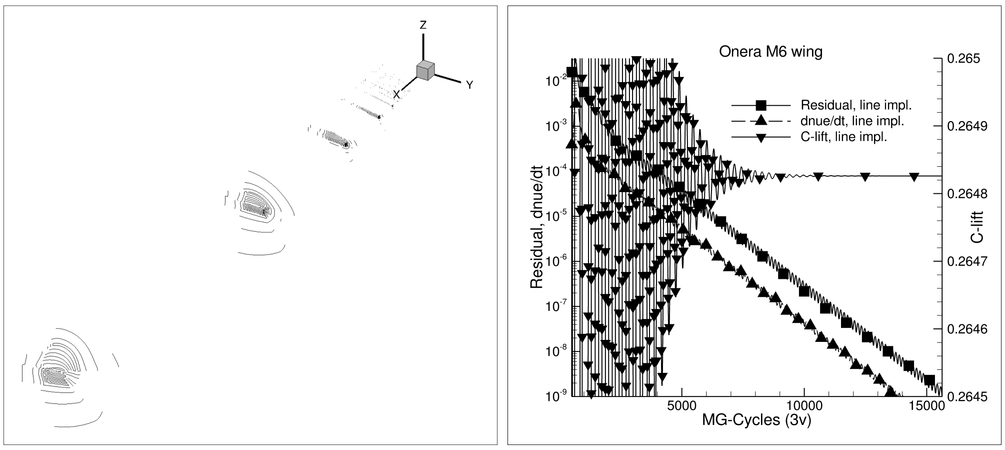

Figure 28: ONERA M6 wing: Determined lines on on the second agglomerated grid, $\gamma_{\text{line}} = 100$, residual, turbulent variable residual and lift coefficient for the line implicit method (right)

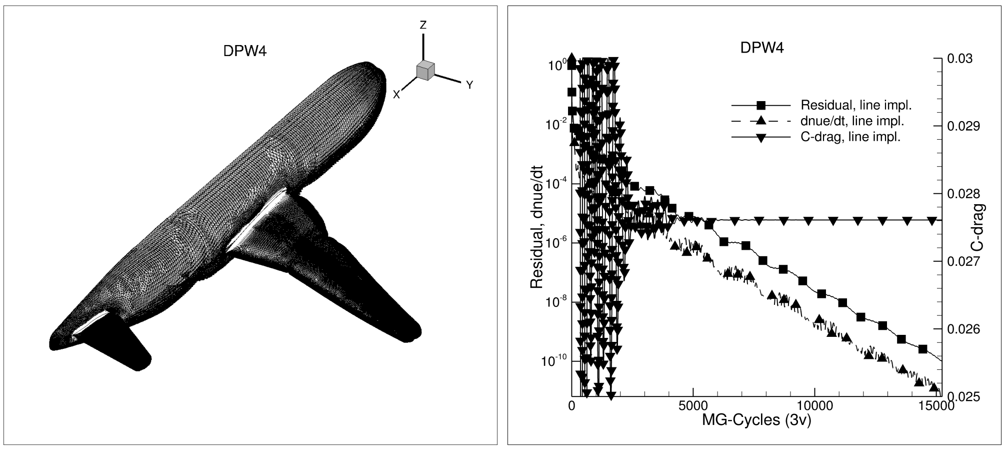

Figure 29: Configuration of the Drag Prediction Workshop 4: Section of the grid (left), density residual, turbulent variable residual and lift coefficient for the line implicit method (right)

References

[1] S. R. Allmaras, *Analysis of semi-implicit preconditioners for multigrid solution of the 2-d compressible navier-stokes equations*, AIAA Paper 95-1651-CP, 95 (1995).

[2] ——, *Multigrid for the 2-d Compressible Navier-Stokes Equations*, AIAA Paper 99-3336, 99 (1999).

[3] J. S. Cagnone, K. Sermeus, S. K. Nadarajah, and E. Laurendeau, *Implicit multigrid schemes for challenging aerodynamic simulations*, Computer & Fluids, 44 (2011), pp. 314–327.

[4] P. H. Cook, M. A. McDonald, and M. C. P. Firmin, *Aerofoil rae 2822 pressure distributions and boundary layer and wake measurements*, AGARD-AR, 138 (1979).

[5] P. Crumpton, *A efficient cell vertex method for unstructured tetrahedral grids*, Technical Report NA 93/09, Oxford University Computing Laboratory, 1997.

[6] P. Eliasson, P. Weinerfelt, and J. Nordström, *Applicaton of a line-implicit scheme on stretched unstructured grids*, AIAA Paper, AIAA-2009-163, (2009).

[7] K. J. Fidkowsky, *A High-Order Discontinuous Galerkin Multigrid Solver for Aerodynamic Applications*, Master's thesis, Massachusetts Institute of Technology, 2004.

[8] G. H. Golub and C. F. van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, second ed., 1983.

[9] N. Kroll, H. Bieler, H. Deconinck, V. Couaillier, H. Ven, and K. Sorensen, *ADIGMA - A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications: Results of a Collaborative Research Project Funded by the European Union, 2006-2009*, Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Springer, 2010.

[10] S. Langer, *Investigation and application of point implicit Runge-Kutta methods to inviscid flow problems*, International Journal for Numerical Methods in Fluids, 69(2) (2012), pp. 332–352.

[11] S. Langer and D. Li, *Application of point implicit Runge-Kutta methods to inviscid and laminar flow problems using AUSM and AUSM$^+$ upwinding*,

International Journal of Computational Fluid Dynamics, 25:5 (2011), pp. 255–269.

[12] J. V. Lassaline and D. W. Zingg, *Development of an Agglomeration Multigrid Algorithm with Directional Corasening*, AIAA paper 99-3338, 1999.

[13] D. J. Mavriplis, *Multigrid Technqiues For Unstructured Meshes*, ICASE Report 95-27, 1995.

[14] ——, *Directional Coarsening and Smoothing for anisotropic Navier-Stokes Problems*, Electronic Transactions on Numerical Analysis, 6 (1997), pp. 182–197.

[15] ——, *Directional Agglomeration Multigrid Techniques for High-Reynolds Number Viscous Flows*, ICASE Report No.98-7, (1998).

[16] ——, *Multigrid Strategies for Viscous Flow Solvers on Anisotropic Unstructured Meshes*, ICASE Report No.98-6, (1998).

[17] D. J. Mavriplis and V. Venkatakrishnan, *A 3D Agglomeration Multigrid Solver For The Reynolds-Averaged Navier-Stokes Equations On Unstructured Meshes*, ICASE Report 95-30, 1995.

[18] P. Moinier, *Algorithm Developments for an Unstructured Viscous Flow Solver*, PhD thesis, University of Oxford, 1999.

[19] E. J. Nielsen, J. Lu, M. A. Park, and D. L. Darmofal, *An implicit, exact dual adjoint solution method for turbulent flows on unstructured grids*, Computer & Fluids, 33 (2004), pp. 1131–1155.

[20] N. A. Pierce, M. B. Giles, A. Jameson, and L. Martinelli, *Accelerating Three-Dimensional Navier-Stokes Calculations*, AIAA 97-1953, (1997).

[21] P. Roe, *Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes*, Journal of Computational Physics, 43 (1981), pp. 357–372.

[22] C.-C. Rossow, *Efficient computation of compressible and incompressible flows*, Journal of Computational Physics, 220 (2007), pp. 879–899.

[23] P. R. Spalart and S. R. Allmaras, *A One-Equation Turbulence Model for Aerodynamic Flows*, AIAA Paper, AIAA-92-439, (1992).

[24] M. Svärd, J. Gong, and J. Nordström, *Stable artificial dissipation operators for finite volume schemes on unstructured grids*, Applied Numerical Mathematics, 56 (2006), pp. 1481–1490.

[25] M. Svärd and J. Nordström, *Stability of finite volume approximations for the laplacian operator om quadrilateral and triangular grids*, Applied Numerical Mathematics, 51 (2004), pp. 101–125.

[26] R. C. Swanson and C.-C. Rossow, *An efficient solver for the RANS equation and a one-equation turbulence model*, Computers & Fluids, 42 (2011), pp. 13–25.

[27] R. C. Swanson, E. Turkel, and C.-C. Rossow, *Convergence acceleration of Runge-Kutta schemes for solving the Navier-Stokes equations*, Journal of Computational Physics, 224 (2006), pp. 365–388.

[28] J. C. Vassberg, E. N. Tinoco, M. Mani, B. Rider, T. Zickuhr, D. W. Levy, O. P. Brodersen, B. Eisfeld, S. Crippa, R. A. Wahls, J. H. Morrison, D. J. Mavriplis, and M. Murayama, *Summary of the fourth aiaa cfd drag prediction workshop*, Tech. Rep. AIAA 2010-4547, June 2010.

DLR-IB-AS-BS-2022-125

TAU Hyperflex Report

Stefan Langer, Axel Schwöppe

<u>Verteiler:</u>

Institutsbibliothek	1 Exemplar
Verfasser	3 Exemplare
Institutsleitung	1 Exemplar
Abteilungsleiter	1 Exemplar
Deutsche Bibliothek in Frankfurt/Main	2 Exemplare
Niedersächsische Landesbibliothek Hannover	1 Exemplar
Techn. Informationsbibliothek Hannover	1 Exemplar
Zentralbibliothek BS	2 Exemplare
Zentralarchiv GÖ	1 Exemplar