Master of Science Thesis

Goal-oriented mesh adaptation using mesh sensitivities as an indicator

A theoretical and numerical evaluation

FWJ Vonck

December 13, 2016

Aerodynamics

Faculty of Aerospace Engineering

TU Delft

Delft University of Technology

Goal-oriented mesh adaptation using mesh sensitivities as an indicator

A theoretical and numerical evaluation

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

FWJ Vonck

December 13, 2016

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

DELFT UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF AERODYNAMICS

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance the thesis entitled **"Goal-oriented mesh adaptation using mesh sensitivities as an indicator"** by **FWJ Vonck** in fulfillment of the requirements for the degree of **Master of Science**.

Dated: December 13, 2016

Supervisors:

Prof. Dr. S. Hickel

Dr. R.P. Dwight

Dr. A. Gangoli Rao

Summary

Numerical simulations are a widely used method to evaluate the aerodynamic performance of a design. One of the sources of error is the discretization error. This type of error is affected by the choice for numerical scheme as well as the discretization of the domain. In practice, the latter part is performed by making use of a mesh. The chosen mesh has an influence of the error in the obtained Quantity of Interest(QoI), however its size also directly influences the computational cost of the solution. Different spatial locations in the domain have different influence on the results. Therefore, it is inefficient to use global refinement. Instead, a method must be derived which can estimate the influence of a spatial location in the domain, in order to be able to efficiently refine the mesh on areas of large influence on the solution. This is done my first computing a flow solution on a rather rough starting mesh and computing an error indicator to identify regions which should be targeted for refinement. This research aims to assess the effectivity and efficiency of one such method, which is based on an error indicator created by using mesh sensitivities. Mesh sensitivities are defined as the derivative of the QoI with respect to the spatial location of a mesh node.

This method has been developed and its potential has been shown in literature. However, it efficiency and effectivity has not been proven. The goal of this research is draw a comparison of the mesh sensitivity based mesh adaptation with a well proven goal-oriented mesh adaptation method. For this purpose the adjoint-weighted residual method is chosen. For finite volume application the method is developed by Venditti and Darmofal, for clarity this method will be referred to as the V&D method. The V&D method has a very strong link to estimating the actual error in the QoI. The efficiency of the method has been shown for a wide variety of flows and geometries.
Both methods use an error indicator to identify spatial regions which require refinement of the grid. The error indicators from both methods are compared from a theoretical point of view. It is found the same adjoint variables are used. The V&D method consists of two parts, being a computable correction term and an error in the computable correction term. The mesh adaptation error indicator is rather similar compared to the first part of the V&D error indicator. However, the second part is not estimated.

In order to compare the efficiency, both methods are implemented and use the same flow

and adjoint solver. Also, the remeshing procedure is performed in the same manner. Some problems are encountered due to the nature of the solver which requires a small change in the V&D method causing it to lose some of its efficiency. Furthermore, the flow solver requires smooth meshes so both methods require a smoothing operator for newly created meshes. This is performed in the same manner for both methods.

The implementation of the V&D is evaluated by using a flow case from literature. The results show a lower efficiency as compared to the literature. However, part of this might also be due to a difference in flow solver. Both goal-oriented methods do show huge improvement compared to global refinement. A total of three flow cases are examined for lift and drag values. The results from the V&D method obtain slightly better accuracy in the solution for comparable number of nodes. Both methods require a very comparable computational effort for an adaptation cycle for creating a new mesh. However, for the V&D method uses a correction term, which requires the adjoint equations to be solved on the last mesh, which is not required for the mesh sensitivity based method.

The aim of this research is to evaluate the mesh sensitivity based mesh adaptation method. The error indicator from this method is not based on the error in the flow, however a comparison with the error indicator from the V&D method shows the indicator is very similar to the so-called computable correction term from this method. This term is directly linked to the error in flow. This clearly supports the use of mesh sensitivities as basis of an error indicator. From numerical studies it is found results are slightly less efficient compared to the V&D method. However, also at sightly lower numerical costs.

Table of Contents

List of Figures

Chapter 1

Introduction

In aerospace engineering numerical simulations are often used to assess the aerodynamic performance of a design. In this framework very small errors can lead to large influences on the operational costs of the aircraft. Therefore, a high accuracy in the simulations is required. In in the field of computational fluid dynamics(CFD) the error originates from three sources: modelling error, discretization error and convergence error. The first is the discrepancy between the real world flow and the exact solution of the used model, e.g. there is a discrepency between reality and how Euler equation describe the airflow. The latter is affected by machine precision, which in practise is multiple orders of magnitude smaller than the discretization error. For fully converged solutions the second type is the difference between the exact solution of the flow model and the solutions to the discretized equations. This error is influenced by two choices, the numerical scheme and the discretization of the computational domain. This second part is performed by making use of a mesh. The mesh not only influences the error but its size has a direct link to the computational costs of the numerical simulation. If global refinement would be applied in a two dimensional setting with a second order accurate solver, decreasing the size of cells by a factor of two leads to a solution four times as accurate in the asymptotic regime, but also four times as much computational work per iteration step, often convergence takes more steps for smaller meshes and the computational work increases even more. Therefore, more effective methods to refine the grid are required.

For mesh adaptation, the areas of higher interest must be determined. Two methodologies can be identified here: (a) flow-feature based adaptation, where regions of interest in the flow can be identified by for instance looking at the derivative of the residuals, and (b) goal-oriented adaptation. These method use *a posteriori* indicators to define areas of high influence in the computational domain with regard to a specific quantity of interest(QoI). In aerospace related problems this QoI is often defined as the lift, drag, moment coefficient or the aerodynamic efficiency. As the mesh is optimized for the specific goal-function of interest, the latter method can lead to a smaller error in the QoI while at the same

computational effort as compared to meshes generated by flow-feature based adaptation. In the literature multiple approaches have been defined in order to create an appropriate indicator for mesh refinement, which will be discussed in chapter two. This research focusses on the approach by using mesh sensitivities as an indicator for goal-oriented mesh adaptation. This method has been defined, but has yet to be evaluated for unstructured grids. The current research will aim to evaluate how effective and efficient mesh sensitivity based adaptation is compared to a currently known and proven method. This will be done by implementing both methods using the same flow solver in order to compare results.

In order to properly compare and assess the method it is required to obtain general knowledge for goal-oriented mesh adaptation. In chapter two a literature review will be given in which the current goal-oriented mesh adaptation strategies are discussed, as well as their results and applicability. Then, in chapter three, a theoretical evaluation of the indicator will be performed by studying the nature of the indicator and drawing a theoretical comparison to the indicator with the strongest theoretical background. In chapter four, a numerical implementation chapter is included. This covers the flow simulation and both approaches for mesh adaptation. Followed by chapter five in which the numerical results are given en discussed, this shows the both the differences in adapted meshes between the methods as well as the efficiency with which the indicator of interest is determined. Finally, in conclusions a wrap up will be made and the effectiveness and efficiency of the method will be discussed.

Chapter 2

Overview of goal-oriented error estimators

Goal-oriented mesh adaptation requires an error estimator which connects the quantity of interest(QoI) to spatial locations in the domain of the partial differential equation. Multiple methods have been defined in literature. All methods make use of a dual problem, yielding an adjoint solution for a chosen QoI to the flow field. First, the adjoint weighted residual method will be discussed for both finite element and finite volume discretizations. Secondly, a method derived using creation of dissipation as in indicator will be used. Followed by the approach using entropy as an indicator. Finally, the method using grid node location sensitivity will be discussed as error indicator. The focus in this chapter lies in identiying the mesh adaptation strategy which can be used as a reference method for evaluation of the mesh sensitivity based mesh adaptation. Ideally, the chosen reference method must have a strong theoretical foundation as well as proven numerical efficient results.

2.1 Adjoint weighted residual error estimator

The theoretical framework of this method is extensively described in the work of Becker and Rannacher [2], followed by Pierce and Giles [3], who prove the super-convergence of the goal function. Local residuals values in the flow represent the extent to which the partial difference equations are not resolved locally. This method introduces the adjoint problem to provide the link between the local residuals in the flow and the goal-function of interest. The product of the residual and the adjoint vector gives a correction to the initial value, leading to an increased accuracy. In order to further improve the mesh using this corrected value, the remaining error must be estimated. This is performed by increasing the order of the computation for both the primal and the adjoint problem. The method can be applied within both the finite volume and finite element approach. Both will be discussed in this section.

2.1.1 Finite element application

The method as described in Section 2.1 has been applied in finite element sense for the two dimensional Euler equations by Hartmann and Houston [4]. Here a transsonic flow through a converging-diverging channel and both subsonic and supersonic flow over an airfoil are regarded. For the channel mesh adaptation is performed with regard to a point value for the pressure just before the shock. The discontinuous Galerkin finite element analyses uses first order solutions to solve the set of equations. For the approximation of the error, second order elements are used as the higher order estimate of the solution. The effectiveness of adapted meshes in computing the goal functions are compared to meshes adapted using unweighted residuals as indicator, this does not require an adjoint solution. In order to compare the results, a fixed number of nodes to be added in each adaptation step for both methods is defined. The reference method targets the shock, while the weighted method also characteristic lines crossing the point of interest are targetted for refinement. It is found that the adjoint-weigthed method gives results for which the error is around an order of magnitude lower compared to the residual based method for the transsonic channel case. The subsonic case also gives improved results for the adjoint-weighted method however the difference is confined to being around one to twice as effective in terms of both computational effort and mesh size for the same error level. For the supersonic case, the meshes are again an order of magnitude better than the reference method.

2.1.2 Finite volume application

In a series of papers the method is applied within a finite volume setting by Venditti and Darmofal, first in a one dimensional case [5], whereafter it is expanded to two dimensional Euler equations[1] and finally, viscous effect are added by treating the Navier-Stokes equations [6]. Nemec and Aftosmis performed work utilizing the same mathod but with more complex geometries in three dimensions on a launch abort vehicle [7]. In contrast to finite element computations where the order of approximation of the elements can be increased, the finite volume method requires an increase in number of volumes to improve the accuracy of the computation. This is achieved by creating a fine mesh from the original coarse mesh by splitting the cells. This is illustrated in Figure 2.1. The coarse mesh is shown in thick black lines, while the fine mesh is shown with thin red lines.

The adjoint and flow solution is then interpolated onto the fine grid by making use of linear least-squares fitting. In order to estimate the error in the computable correction a second-order fitting is used to estimate the unknowns and adjoint. The estimated error in the fine mesh is defined as the product of the residual and adjoint value per cell. Finally, the estimation of the error in the coarse grid cell is computed by taking the sum of the absolute values of the error in the fine cells per coarse cell.

The results for the Venditti and Darmofal series of papers are compared to the flow-based features based adaptation. Here the second derivatives of pressure and Mach number multiplied

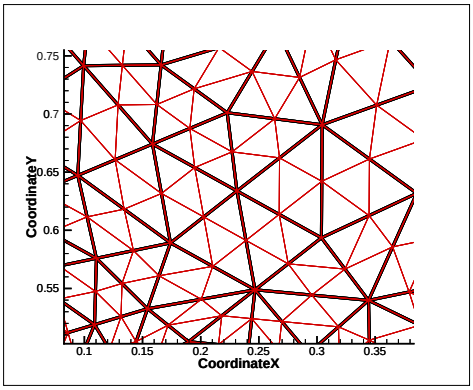

Figure 2.1: Construction of fine mesh(red) from the coarse mesh(black)

by the cell size is used as mesh refinement indicators. The results show much quicker convergence for the lift and drag-based adaptations compared to the flow-feature based adaptation. Also, the goal-oriented method show convergence towards steady values, while in some cases the flow-feature based methods do not due to excessive refinement around a shock.

In the paper by Nemec and Aftosmis [7], the method has been applied in a three dimensional setting, applied to pressure measurements. The papers shows the method is robust and that values converge towards correct limiting values. This study thus shows also complex flows with many different length scales can be accurately captured by goal oriented residual-weighted mesh adaptation.

2.2 Dissipation based adaptation

A different approach focusses on numerical dissipation. This method is described by Dwight in [8] and [9]. It focusses on the error due to added numerical dissipation. The method is specifically developed for the Jameson-Schmidt-Turkel (JST) scheme, as described in [10]. This scheme adds second and fourth order dissipation to the solution. A shock switch is present, which determines the amount of fourth and second order dissipation, to try to obtain both stability and accuracy. It is argued that the error due to this additional dissipation is responsible for a largest part of the error. This statement is made after the computation of relative error source for several 2D and 3D examples, including NACA0012 and the ONERA M6 wing. The error due to dissipation contributes to more than 90% of the total error. Furthermore, a link is established between the error in the goal function J due to dissipation and the sensitivity of J with respect to the level of dissipation. This statement is made from the observation that as the cell size in the mesh tends to zero, both the dissipation as the sensitivity to the dissipation coefficients tends to zero. The discrete adjoint equation is then used to establish the sensitivity of J with respect to the dissipation coefficients.

Numerical results of this method are compared against results acquired by mesh adaptation

using local gradient adaptation and to global refinement. For two dimensional test cases it is shown that for a subsonic flow, where the problem can be described as having an elliptic nature, the convergence rate is comparable to that of the flow feature based one. However for the trans- and supersonic test cases, the flow feature based criteria convergences slower and appears to have a systematic error, while this method shows a convergence around 50 times as efficient as global refinement. Furthermore, the meshes of feature based adaptation and dissipation based adaptation are compared and it is shown that the meshes become more irregular and more cells are added upwind for the dissipation-based adaptation. Also, more points are added around the shock rather than on the shock line. As its values of lift and drag are better this might imply the location of the shock being wrongly simulated in the feature-based adaptation method.

For three dimensions only a transonic test is used and it is again shown to behave particularly better as compared to the flow feature based sensor. Solutions are well within engineering accuracy bounds. For lift a very accurate value is found, however for drag there is some more discrepancy possible due to the fact that this method is only taking into account the error due to dissipation.

2.3 Entropy based adaptation

In the third method is not goal oriented, rather a general adaptive indicator is used, as is described by Fidkoski and Roe in [11] and [12]. The indicator targets the spurious creation of entropy. In contrast to the previous methods, this does not use one type of force as a goal function but rather uses the integrated residual of the entropy transport equation as a refinement indicator. The indicator therefore highlights regions where the entropy equation is not sufficiently resolved. Furthermore it is shown that for an inviscid and shock-free flow the entropy variables serve as an adjoint for net entropy flow out of the domain.

Numerical results are generated for both inviscid and viscous two dimensional flows. This method is compared to the results of that of the adjoint-weighted residual sensors. In the subsonic test case, the results for drag are comparable, while for moment and lift coefficient the entropy based adaptation performs slightly better. With the additional advantage that it is not required to compute the adjoint for the different goal function. The lift and moment coefficient oriented methods refine significantly more around the upwind stagnation streamline. It is argued that this is caused by a singularity which scales with the square root of the distance to the stagnation streamline [13]. This causes excessive mesh refinement in these areas. For a transonic test case, the results for lift and drag are still comparable to the reference method, but the more and the stronger the shocks are, the worse the method becomes. The method assumes all entropy creation to be spurious, while shocks physically produce entropy. Therefore, there will be excessive refinement around the shock. Thereby it loses its efficient nature of adapting the mesh. In short this method can be classified as very efficient in subsonic regions, but in transonic regimes its loses its functionality. As do most feature-based adaptivity sensors.

2.4 Mesh sensitivity based adaptation

The last method uses sensitivities in the goal-function with respect to the mesh coordinates. It is the focus of this thesis. The goal of this thesis is to determine its effectivity and efficiency. This method uses the the adjoint equation in order to identify regions of high sensitivity in the mesh, as described in [14]. This method has significantly lowered the computational costs of computing mesh sensitivities as compared to compute derivatives of a goal-function with respect to node locations using linearisation of the mesh. Peter[15] uses the sensitivities, denoted as $\frac{\partial J}{\partial X}$, as a basis for refinement indicator as well as relocations of nodes applied to structured grids. Mesh points on the boundaries of the domain are not free to move, e.g. if the mesh points lie on the arifoil under investigation, moving in certain directions will change the shape of the airfoil, which is is undesirable. A projection $P\left(\frac{\partial J}{\partial X}\right)$ of $\frac{\partial J}{\partial X}$ is used to only maintain projections for allowable degrees of freedom, hence for points on the wall, the normal component is removed. As there is no actual error indicator but rather mesh sensitivity are used, an adaptation strategy has to be determined. It is argued that there are two types of flows, which require two different strategies:

- In the case of a specific flow type, where the integral property of interest is monotonically affected by the numerical dissipation. For instance the stagnation pressure value in Euler flows, is always underestimated due to numerical dissipation.
- In other types of flows, a heuristic method can be used to add nodes in areas of a high derivative.

Several tests are performed using the Euler equations for some two and three dimensional flows for both types of flow and their according strategies. It is found favourable to change the indicator to also incorporate the current cell size, hence the new indicator is defined as: $\theta = hP\left(\frac{\partial J}{\partial X}\right)$, in which h denotes the distance to neighbouring nodes. Also it is concluded that areas where θ is uniform have a higher need for refinement than fast fluctuating areas. In [16], in the light of the last conclusion a spatial mean of $P\left(\frac{\partial J}{\partial X}\right)$, denoted as $\overline{P\left(\frac{\partial J}{\partial X}\right)}$. It is argued that using this spatial mean of the indicator improves the results as, as indicator value will then be suppressed in irregular sensitivity fields, therefore regions with an aligned indicator will be stronger highlighted. Furthermore, the behaviour of the error indicator $\frac{\partial J}{\partial X}$ as the mesh size tends to zero is studied. It is argued that if θ goes to zero as the mesh size tends to zero this indicator will in the limit eliminate the errors in the mesh. A short study of the behaviour of $\frac{\partial J}{\partial X}$ on regular meshes is performed, showing converges towards limiting fields.

Numerical tests are performed to show error convergence for two dimensional examples. However no comparison is made towards its efficiency. In [17] this method is applied to three dimensional RANS flows on an industry size test case. The improvements are not as good as for two dimensional, still goal values where improved using this method, however a conclusion is drawn that re-meshing methods should be improved.
This method has been extended towards unstructured meshes in [18]. However the results of the adaptation process are not assessed yet.

2.5 Research objective

In the previous section a number of different mesh adaptation methods are discussed. Of all the mentioned methods, only the performance of the last method has not been evaluated for unstructured meshes. Also, the method has not been compared to other goal-oriented mesh adaptation strategies. Furthermore, the indicator is not directly linked to error in the computation, hence there is no theoretical proof yet that it will effectively reduce the error in the goal function. This leads to the following research question:

Is goal-oriented mesh adaptation using a mesh sensitivity strategy an efficient and effective method to obtain integral properties of interest for the Euler equations using finite-volume methods for 2D unstructured grids?

In order to be able to answer this question, it is divided into multiple sub-questions. The first question will deal with the theory behind using mesh sensitivities as basis of the refinement indicator:

How are mesh sensitivities related to the error of the goal function?

This question can be split up into an empirical and a theoretical part: What is the convergence behaviour of mesh sensitivities as the mesh size tends to zero for both two dimensional Euler flow on unstructured grids? Is there a mathematical relation between the mesh sensitivities and the error-estimator as defined by Venditti and Darmofal[5]?
The first part will have to provide information on how the indicator behaves as the mesh becomes finer. This gives information on the possibility of using the quantity as refinement indicator. For the second part the error indicator with the strongest theoretical link found in literature is used to compare to. It can give in inside into how the error is estimated. The second subquestion will be posed as follows:

How does the efficiency of mesh adaptation using mesh sensitivity compare with the efficiency of a method using adjoint-weighted residual in a two dimensional Euler flow?

In order to be able to answer this question, it needs to be known how the adaptation chain of mesh sensitivity-based goal-oriented mesh adaptation compares to the adaptation chain of adjoint weighted residual adaptation in terms of numerical costs.
The first part deals with the effort required to obtain the results of both methods. While the second part deals with the accuracy of the results. When both are answered, it will be possible to answer the subquestion on comparing the efficiencies of both methods. In order to perform this task, both methods will have to be implemented using the same flow and adjoint solver. Since, these can be of large influence on the results. For, the adjoint-weighted residual method there are some flow cases available for Euler flows. Therefore, at least one of these known cases will be performed in order to assess how well the adjoint weighted residual method is implemented in this framework. For the mesh sensitivity-based method, there is no literature yet on results in goal-values, hence it cannot be compared.

Chapter 3

Theoretical evaluation of mesh sensitivity as an error indicator

In order to asses whether using mesh sensitivities as an error indicator is a suitable method, a theoretical evaluation of its link with the local error level must be made. From now on the mesh sensitivity based mesh adaptation strategy will be refered to as the *theta* method. In the literature [15] θ is the symbol used to represent the sensor. As stated in the previous chapter, the adjoint-weighted residual method will be used to compare the *theta* sensor as this method shows a very clear link towards the local contribution to the error in the goal-function. As both methods utilize the same adjoint problem, this will first be discussed. Next, the theory of the adjoint weighted residual method will be discussed. Followed by the theory of the mesh sensitivities. Finally a comparison will be drawn in order to show how the error will be estimated.

3.1 Adjoint problem

The adjoint problem is often utilized in the aerodynamical design process as well as for error indication. In both an evaluation of the gradient of a cost function, this case the goal function, with respect to design variables is searched for. In aerodynamic computations computation of the cost function is often very expensive. Furthermore, the problems are often characterized by a very high number of design variables. For such problems, the adjoint method is a very suitable tool as it allows one to compute the derivative of the cost function with respect to the design parameters, while its computation is only weakly dependant on the number of design variables. First, the computations required by the primal approach towards computing the derivatives of a goal-function towards a number of design variables will be shown. Then the adjoint approach is discussed.

3.1.1 Primal approach

The adjoint is always computed with respect to a QoI, in this case, J, which is defined as

$$J := J\left(W\left(X,\alpha\right),X\left(\alpha\right),\alpha\right). \tag{3.1}$$

The direct approach to obtain a derivative $dJ/d\alpha_i$, consists of applying the chain rule to obtain:

$$\frac{dJ}{d\alpha_i} = \frac{\partial J}{\partial W}\frac{dW}{d\alpha_i} + \frac{\partial J}{\partial X}\frac{dX}{d\alpha_i} + \frac{\partial J}{\partial \alpha_i}. \tag{3.2}$$

Here, J represents the goal-function, α_i the design variables, this requires the following total derivatives $\frac{dW}{d\alpha_i}$ and $\frac{dX}{d\alpha_i}$, total derivates also take indirect effects into account and are thus more work to compute compared to partial derivatives. In this case, especially the first term requires a large amount of computational work to be obtained. Instead, this is obtained using the notion of the residual equation, which is by definition required to be zero regardless of a change in design variable leading to the condition of $\frac{dR}{d\alpha_i} = 0$. This gives the following equation.

$$\frac{dR}{d\alpha_i} = \frac{\partial R}{\partial W}\frac{dW}{d\alpha_i} + \frac{\partial R}{\partial X}\frac{dX}{d\alpha_i} + \frac{\partial R}{\partial \alpha_i} = 0. \tag{3.3}$$

In order to obtain $\frac{dW}{d\alpha_i}$, the linear system (3.3), based on a linearization of the discrete flow equations, needs to be solved once for every design variable. This is very computationally expensive.

3.1.2 Adjoint approach

A different approach aims at acquiring the derivative by starting with the Lagrangian:

$$\mathcal{L}\left(W,X,\alpha,\Lambda\right) = J\left(W,X,\alpha\right) + \Lambda^T R\left(W,X,\alpha\right). \tag{3.4}$$

Here Λ is the set adjoint variables. Since $R = 0 \,\forall\, \alpha$, implying $\mathcal{L} = J \,\forall\, alpha, \Lambda$. Therefore in term of derivatives:

$$\frac{d\mathcal{L}}{d\alpha} = \frac{dJ}{d\alpha}. \tag{3.5}$$

Applying the chain rule to (3.4) yields:

$$\frac{d\mathcal{L}}{d\alpha} = \left\{\frac{\partial J}{\partial W}\frac{dW}{d\alpha} + \frac{\partial J}{\partial X}\frac{dX}{d\alpha} + \frac{\partial J}{\partial \alpha}\right\} + \Lambda^T\left\{\frac{\partial R}{\partial W}\frac{dW}{d\alpha} + \frac{\partial R}{\partial X}\frac{dX}{d\alpha} + \frac{\partial R}{\partial \alpha}\right\} \tag{3.6}$$

. Rearranging the terms leads to:

$$\frac{d\mathcal{L}}{d\alpha} = \left\{\frac{\partial J}{\partial W} + \Lambda^T\frac{\partial R}{\partial W}\right\}\frac{dW}{d\alpha} + \left\{\frac{\partial J}{\partial X} + \Lambda^T\frac{\partial R}{\partial X}\right\}\frac{dX}{d\alpha} + \left\{\frac{\partial J}{\partial \alpha} + \Lambda^T\frac{\partial R}{\partial \alpha}\right\}. \tag{3.7}$$

Now the unknown term $\frac{dW}{d\alpha}$ can be removed from this equation by chosen the adjoint variables to be as follows:

$$\left(\frac{\partial R}{\partial W}\right)^T\Lambda = -\left(\frac{\partial J}{\partial W}\right)^T \tag{3.8}$$

This equation needs to be solved once for every goal-function, independently of the number of design variables. This definition of the adjoint gives a connection between the derivative of the local residual with respect to the flow field to the derivative of a specific goal-function with respect to the flow field. Using (3.7), (3.8) and (3.5), yields:

$$\frac{dJ}{d\alpha}\frac{d\mathcal{L}}{d\alpha} = \left\{\frac{\partial J}{\partial X} + \Lambda^T \frac{\partial R}{\partial X}\right\}\frac{dX}{d\alpha} + \left\{\frac{\partial J}{\partial \alpha} + \Lambda^T \frac{\partial R}{\partial \alpha}\right\}. \tag{3.9}$$

In terms of computational costs, the biggest cost is the solving of the system of equations (3.8) once every goal-function. While in the primal approach the system of equations as described in (3.8) must be computed for every design variable. Since in a mesh optimization problem, such as regarded in this research, there are far more design variables as compared to goal-functions, this approach gives the potential for much lower computational costs. The costs of both linear set of equations, seems to be comparable as both sets of equations are equal in size for the same problem.

3.2 Adjoint-weighted residual based error estimation

As is discussed in Chapter 1, this research focusses on the discretization error. This is defined as the differences between the exact solution of the continuous system and the exact solution of the discrete system. A method of quantifying this is by looking at the residual of the discrete system of flow equations, which is found by inserting the discrete solution into the PDE High values of the residual indicate regions where the differential equation are not well-resolved. The product of the aforementioned adjoint and the exact residuals give the error in the output function. Solving for an exact adjoint and residual is not possible and thus a choice needs to be made on what level of discretization both will be evaluated.

In order to estimate the error using the adjoint and residual terms, the starting point is a coarse discretization with N_H degrees of freedom, with computed discrete solution. In case the output function obtained from the coarse grid computation, J_H, is not sufficiently accurate, the error of this function could be estimated by utilizing a finer discretization, denoted by N_h. However, solving the solution on this fine grid is a computational expensive process which will not be done. Instead, an estimation of the output function, J_h, will be made. This is done by making a first order extension around the coarse grid solution:

$$J_h\left(\mathbf{u}_h\right) \approx J_h\left(\mathbf{u}_h^H\right) + \left.\frac{\partial J_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H}\left(\mathbf{u}_h - \mathbf{u}_h^H\right). \tag{3.10}$$

Here, the notation $\mathbf{u}_h^H$ is a projection of the coarse grid solution onto the fine grid, defined as follows:

$$\mathbf{u}_h^H = I_h^H \mathbf{u}_H, \text{ with } \& I_h^H \in \mathbb{R}^{N_h \times N_H}. \tag{3.11}$$

Here, I_h^H is a projection operator, the form of which depends on the type of discretization. In order to estimate the vector $\left.\frac{\partial J_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H}$, the nonlinear residual operator of the considered PDE is considered on the fine grid level:

$$\mathbf{R}_h\left(\mathbf{u}_h\right) = 0. \tag{3.12}$$

Since, the method is applied within a finite volume discretization, $\mathbf{R}_h$ represents an integral statement. Now, linearizing around the coarse grid solution of the residual of the PDE gives:

$$\mathbf{R}_h\left(u_h\right) \approx \mathbf{R}_h\left(\mathbf{u}_h^H\right) + \left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H} \left(\mathbf{u}_h - \mathbf{u}_h^H\right). \tag{3.13}$$

Here, the Jacobian $\left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H}$ contains the sensitifities for the fine grid scales, however it is evaluated at the coarse grid level. Using the condition (3.12) and inverting (3.13), assuming the system is non-singular, the following expression can be obtained:

$$\left(\mathbf{u}_h - \mathbf{u}_h^H\right) \approx -\left[\left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H}\right]^{-1} \mathbf{R}_h\left(\mathbf{u}_h^H\right). \tag{3.14}$$

Inserting this expression into (3.10) yields an expression for estimating the goal-function on the fine grid level:

$$J_h\left(\mathbf{u}_h\right) \approx J_h\left(\mathbf{u}_h^H\right) - \left(\left.\mathbf{\Lambda}_h\right|_{\mathbf{u}_h^H}\right)^T \mathbf{R}_h\left(\mathbf{u}_h^H\right). \tag{3.15}$$

Here, $\left.\mathbf{\Lambda}_h\right|_{\mathbf{u}_h^H}$ represents the discrete adjoint solution, satisfying the following condition:

$$\left[\left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H}\right]^T \left.\mathbf{\Lambda}_h\right|_{\mathbf{u}_h^H} = \left[\left.\frac{\partial J_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H}\right]^T. \tag{3.16}$$

The estimation of the goal-function in (3.19) is based on first-order approximations and will thus be exact in the case of linear residual and goal-function. Which is often not the case, nevertheless it can be used for estimating the value if the grid is sufficiently fine. However, the adjoint vector is required to be computed on the fine grid, which is most likely on the same order of computation effort as computing the flow problem on the fine grid and thereby undesirable. Therefore, a projection operator in analogy to that of the flow variables will be used:

$$\mathbf{\Lambda}_h^H = K_h^H \mathbf{\Lambda}_H. \tag{3.17}$$

Here $\mathbf{\Lambda}_H$ is the adjoint on the coarse grid obtained by solving the following equation:

$$\left[\frac{\partial \mathbf{R}_H}{\partial \mathbf{u}_H}\right]^T \mathbf{\Lambda}_H = \left[\frac{\partial J_H}{\partial \mathbf{u}_H}\right]^T. \tag{3.18}$$

Furthermore, $\mathbf{\Lambda}_h^H$ in (3.17) is the adjoint vector projected from the coarse grid onto the fine grid and this vector shall be used to replace $\left.\mathbf{\Lambda}_h\right|_{\mathbf{u}_h^H}$ in (3.15) to become:

$$\tilde{J}_h\left(\mathbf{u}_H\right) \approx J_h\left(\mathbf{u}_h^H\right) - \left(\mathbf{\Lambda}_h^H\right)^T \mathbf{R}_h\left(\mathbf{u}_h^H\right). \tag{3.19}$$

The notion $\tilde{J}_h$ is now the corrected goal-function value. Now the next step is the estimate the error in this corrected result. This can then be used to highlight areas in the domain to refine the grid. This term is the discrete analogue form of [3] and the same shape as [1]. Then remaining correction term will also follow the steps as presented by the latter paper.

In order to determine the error in the goal-function value, the difference between value based on the interpolated flow variables and that of the fine grid flow variables must be determined:

$$J_h\left(\mathbf{u}_h^H\right) - J_h\left(\mathbf{u}_h\right) \approx \underbrace{\left(\mathbf{\Lambda}_h^H\right)^T \mathbf{R}_h\left(\mathbf{u}_h^H\right)}_{\text{computable correction}} + \underbrace{\left(\mathbf{\Lambda_h}|_{\mathbf{u}_h^H} - \mathbf{\Lambda}_h^H\right)^T \mathbf{R}_h\left(\mathbf{u}_h^H\right)}_{\text{error in computable correction}}. \tag{3.20}$$

Now if the error in computable correction part is isolated, this becomes:

$$E_{cc} \approx \left(\mathbf{\Lambda_h}|_{\mathbf{u}_h^H} - \mathbf{\Lambda_h^H}\right)^T \mathbf{R}_h\left(\mathbf{u}_h^H\right), \tag{3.21}$$

and this in its turn can be rewritten as:

$$E_{cc} \approx \left\{\mathbf{R}_h^{\mathbf{\Lambda}}\left(\mathbf{\Lambda}_h^H\right)\right\}^T \left[\left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H}\right]^{-1} \mathbf{R}_h\left(\mathbf{u}_h^H\right) \tag{3.22}$$

Whereby now the term $\mathbf{R}_h^{\mathbf{\Lambda}}$ represents the adjoint of the residual operator, which is defined as follows:

$$E_{cc} \approx \left\{\mathbf{R}_h^{\mathbf{\Lambda}}\left(\mathbf{\Lambda}_h^H\right)\right\}^T \left(\mathbf{u}_h - \mathbf{u}_h^H\right) \tag{3.23}$$

It is then argued that combining the expressions as given in (3.21) and (3.23), gives the optimal result. However, one of these expressions requires the adjoint to be solved on the fine grid scale, while other one requires the flow solution to be solved as the fine grid scale. Instead, these values will be replace these terms by a quadratic interpolation in order to estimate the error in the computable correction.

3.3 Mesh sensitivity

The method aims to compute the effect of a specific node location towards the goal function. This will be expressed as $\frac{dJ}{dX}$. The dependence of the goal-function on the location of nodes can be obtained from (3.9). The design variable is now set to be a change in mesh location vector X, the effect of moving a mesh node on the boundary of the goal-function for the flow variables however does need to be taken into account. This leads to:

$$\frac{dJ}{dX} = \frac{\partial J}{\partial X} + \frac{\partial J}{\partial u_b}\frac{\partial u_b}{\partial X} + \Lambda^T \frac{\partial R}{\partial X}. \tag{3.24}$$

Whereby u_b denote the state variables at the boundary where the integral of the goal function is applied. The first component from (3.24) is only non-zero if a node lying on the integral property of interest is considered, while the second components is only relevant in direct vicinity or direct on the support of the integral of interest. The third component can be non-zero over the entire domain and utilizes the same definition for the adjoint as in described Section 3.1:

$$\frac{\partial J}{\partial \mathbf{u}} = -\mathbf{\Lambda}^T \frac{\partial \mathbf{R}}{\partial \mathbf{u}}. \tag{3.25}$$

The $\frac{dJ}{dX}$ is now defined in all spatial directions. However on the support of the integral not all movements are allowed. Only directions which do not alter the boundaries of the

domain of the PDE are allowed. This implies that boundary nodes are only allowed to have move tangential to the boundary of the domain. To guarantee this a projection of the mesh sensitivity is defined:

$$P\left(\frac{dJ}{dX}\right) = \begin{cases} \frac{dJ}{dX} \& \text{if outside support of J or boundaries of the domain} \\ \frac{dJ}{dX} - \frac{dJ}{dX} \cdot \vec{n} \& \text{if on support of J or boundaries of the domain} \\ 0 \& \text{if on a corner of the support of J or boundaries of the domain} \end{cases} . \quad (3.26)$$

Here, $\vec{n}$ represents the outward normal of the boundary or support face. The sensitivity of the goal function with respect to change in location of the node is described in (3.24). In [16] the indicator has been extended to $\theta = rP\left(\frac{dJ}{dX}\right)$. Whereby the r term is defined as the allowable mesh deformation. This is equal to half the shortest distance to a neighbouring node. Thereby effectively adding a variable which scales with the cell size into the indicator. This creates a situation in which the derivative of the goal-function w.r.t. a change in spatial location of a node is mutiplied with a spatial quantity connected to the maximum allowable displacement of this same node, thereby this indicator becomes an first-order estimate of the maximum change in the goal-funtion through the movement of this node.

In the previous research aimed at structured meshes [15] and [16], use is made of a space average value of $P\left(\frac{dJ}{dX}\right)$. The argument is made that if the $P\left(\frac{dJ}{dX}\right)$ field is non-regular at a certain area, the effects of a smaller mesh will most likely be rather small, while in an area of regular $P\left(\frac{dJ}{dX}\right)$ the effect of a refined mesh will likely be larger.
In the current research unstructured meshes are regarded. This implies that cells can be refined without having to refine a large surrounding area. Furthermore, a non regular area of the mesh sensitivity indicator might not directly lead to a change in the goal function, it does indicate a high dependence on the used mesh. Therefore, in this research it is chosen to not use a local average of the indicator. Instead, the absolute value of the vector will be used.

3.4 Comparison of indicators

The adjoint-weighted residual method estimates the error in the goal-function and is therefore a very logical indicator to use, from now it will be referred to as the V&D method after its developers Venditti and Darmofal. The mesh sensitivity indicator does not estimate the error in the goal-function. However it does highlight locations where the solution is very sensitive to the chosen mesh. Intuitively, if the exact location of the node has a profound influence on the goal function, this calls for a finer mesh at these locations to lower the effect of the utilized mesh. However, in order to make a stronger statement about the theoretical foundation of this indicator, the terms will be compared in greater detail in this section.

In the previous sections the terms in both methods are defined. The V&D error estimator in (3.20) has two parts, being the computable correction and the error in that computable correction. Both parts will be used for this comparison as both are part of the method to estimate the error. For the estimation of the fine grid in the error in the computable correction term, a second order interpolation is used.

First, the computable correction is used as a comparison. This shows that the adjoint, Λ, is the same for both methods. However, the location is defined slightly different as the V&D method uses an interpolated fine grid, whereas the mesh sensitivity method uses a coarse grid nodal based criterion. The larger difference between the two methods however is the part linked to the residual. The V&D method utilizes an interpolated value at the fine grid level from which a residual can be computed. The mesh sensitivity method uses the partial derivative of the residual with respect to the change in location of the node multiplied by the distance r. Since r is defined as half the distance to the closest neighbouring node. This scales with the cell size, while noting that irregular meshes lead to the lowest distance being used. This distance is also equal to the shortest cell size of the fine grid. In case of a very fine mesh with regular $\frac{\partial R}{\partial X}$ field, it can be concluded that this estimates the value of R at the location of equal the size of the fine grid from the node. The computed value of $\frac{\partial R}{\partial X}$-vector is the value the direction of the strongest gradient. So rather than estimating the value over the domain by use of the fine grid, it estimates the highest value in any direction. If a one-dimensional problem would be regarded, the mesh senisitivity method would estimate the value at the very same point, the V&D method estimates the error in the solution.

The error in the computable correction is estimated by utilizing more degrees of freedom. In the setting for finite-volume schemes as used in this research this implies a higher order fitting. Since the mesh sensitivity indicator only utilizes $\frac{\partial R}{\partial X}$, it does not estimate $\frac{\partial^2 R}{\partial X^2}$. Furthermore, even if second order effects would be computed, these would have been summed up with the first order effects and the absolute value in the direction of largest vector would used as an indicator. Thereby part of the information would be lost.

From comparing the two estimators it can thus be concluded that the mesh sensitivity indicator is comparable to the computable correction term, except the highest gradient is utilized. However, these effects will becomes smaller with decreasing mesh size. For the error in the computable correction, this is not estimated with this method. Also, adding second order effects to the indicator is not expected to be very effective since all directional information will be lost, once the norm of the vector is taken to obtain θ.

Chapter 4

Numerical implementation

In this chapter the numerical set-up is described. The numerics are used to generate the results and thereby assessing the effectiveness and efficiency of both methods. As a solver the *elsA* code from ONERA is used. The set-up will be an unstructured mesh. First, the implementation of the Euler equations followed by the adjoint equations are discussed. Then the method of obtaining the mesh refinement indicators for both methods are treated and finally the framework of the complete adaptation chain for both methods is described.

4.1 Euler equations

The equation under consideration are the inviscid Euler equations. The equation in integral form within a defined volume V, surrounded by the surface S is defined as follows

$$\frac{\partial}{\partial t}\int_V U dV + \oint_S F dS = 0. \tag{4.1}$$

The vectors containing the conservative variables, denoted by U, and fluxes, denoted by $\vec{F}\vec{n}$ are defined as follows:

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix}, F = \begin{bmatrix} \rho u & \rho v & \rho w \\ \rho u^2 + p & \rho uv & \rho uw \\ \rho uv & \rho v^2 + p & \rho vw \\ \rho uw & \rho vw & \rho w^2 + p \\ \rho uH & \rho vH & \rho wH. \end{bmatrix} \tag{4.2}$$

Here ρ is the density, u, v and w are the velocity vectors along the x-axis, y and z-axis, respectively. E is the total energy per unit mass and H is the total enthalpy per unit mass, which is a result of the other variables:

$$H = E + \frac{p}{\rho}. \tag{4.3}$$

This leads to 6 unknowns and 5 equations, in order to compute the unknowns, a sixth equation will be added in the form of assuming an ideal gas. This gives the following relation:

$$\rho E = \frac{p}{\gamma - 1} + \frac{\rho}{2}\left(u^2 + v^2 + w^2\right). \tag{4.4}$$

Here, γ is the specific heat ratio. The Euler equation allow for non-physical expansion shocks, therefore a condition to prevent the destruction of entropy must be enforced. This is the second law of thermodynamics:

$$\frac{\partial s}{\partial t} + \left(\vec{u} \cdot \vec{\nabla}\right) s \geq 0. \tag{4.5}$$

The entropy is denoted by s. The Euler equations have been discretized using a second order upwind finite volume scheme. This is done by using the flux difference method, the approach used in the current research is that of Roe approximate Riemann solver, which will be discussed in section 4.1.1. Afterwards the extension towards second order will be made by making use of the MUSCL scheme as will be described in section 4.1.2.

4.1.1 Roe flux difference scheme

The method is first described by Roe [19]. The method is based on using two neighbouring cells, u^L and u^R connected via a face. Then an average Jacobian matrix $\tilde{A}$ is constructed, which needs to satisfy the following constraints:

- As $u^L \to u^R$, then the Jacobian matrix $\tilde{A} \to \frac{\partial U}{\partial F}$.
- For any u^L and u^R, the Jacobian must satisfy $\tilde{A} \times \left(u^L - u^R\right) = F^L - F^R$.
- The eigenvectors of the Jacobian matrix $\tilde{A}$ must be linearly independent.

The transformation from the exact Jacobian matrix into the matrix $\tilde{A}$ is performed by making use of the properties of the left and right cell of a face. The definition is made such that the normal vector points from the left to the right cell. For the definition of the average values, a new variable, R_ρ, will be introduced:

$$R_\rho = \sqrt{\frac{\rho^R}{\rho^L}}. \tag{4.6}$$

From this the average values required for roe method's are defined as:

$$\begin{array}{ccc} \bar{\rho} = R\rho^L & \bar{u} = \frac{u^R \cdot R_\rho + u^L}{1 + R_\rho} & \bar{v} = \frac{v^R \cdot R_\rho + v^L}{1 + R_\rho} \\ \bar{w} = \frac{w^R \cdot R_\rho + w^L}{1 + R_\rho} & \bar{e} = \frac{1}{2}\sqrt{\bar{u}^2 + \bar{v}^2 + \bar{w}^2} & \bar{h} = \frac{h^R \cdot R_\rho + h^L}{1 + R_\rho}. \\ \bar{c} = \sqrt{(\gamma - 1)\left(\bar{h} - \bar{e}\right)} & & \end{array} \tag{4.7}$$

Now the eigenvalues of the matrix $\tilde{A}$ are:

$$\lambda_1 = \lambda_2 = \lambda_3 = \bar{u}_n \tag{4.8}$$
$$\lambda_4 = \bar{u}_n + c \tag{4.9}$$
$$\lambda_5 = \bar{u}_n - c. \tag{4.10}$$

In this framework the variable $\bar{u}_n$ is defined as the velocity in the direction of the face normal. It is thus defined as:

$$\bar{u}_n = \vec{\bar{u}} \cdot \vec{n}, \text{ with } \vec{\bar{u}} = \begin{bmatrix} \bar{u} \\ \bar{v} \\ \bar{w} \end{bmatrix}. \tag{4.11}$$

This method allows for expansion shocks. In order to prevent these from occurring the Harten correction will be applied[20]. If an eigenvalue gets very close to zero it is different method for computing the value is used.

$$|\lambda_i| = \begin{cases} |\lambda_i| & \text{if } |\lambda_i| \geq \delta \\ \frac{\lambda_i^2 + \delta^2}{2\delta} & \text{if } |\lambda_i| < \delta \end{cases}. \tag{4.12}$$

The variable δ is dependent on the jump in velocity scaled by a numerical setting.
The corresponding fluxes can be determined by making use of difference of two adjacent fluxes, which is by definition [21]:

$$F^R - F^L = \sum_j \Gamma_j \lambda_j e_j. \tag{4.13}$$

Here Γ_j is the characteristic strength, λ_j is the wave speed and e_j is the right eigenvector of A, all corresponding the the j-th wave. This leads to a flux at the interface, F^{ROE}, as follows:

$$F^{ROE} = F^L + \sum^{(-)} \Gamma_j \lambda_j e_j. \tag{4.14}$$

Or, alternatively

$$F^{ROE} = F^R - \sum^{(+)} \Gamma_j \lambda_j e_j. \tag{4.15}$$

Here $\sum^{(-)}$ is a sum over all negative wave speeds, while $\sum^{(+)}$ is a sum over all positive wave speeds. Both expressions can be combined by taken the average value, this leads to the final expression:

$$F^{ROE} = \frac{1}{2}\left(F^L + F^R\right) + \frac{1}{2}\left(\sum \Gamma_j \left|\lambda_j\right| e_j\right). \tag{4.16}$$

In *elsA* all interface variables will be expressed in primitive variables. Therefore the fluxes can be defined in the primitive variables, denoted by P, only.

4.1.2 MUSCL extension towards second order

The above described Roe flux scheme is extended to second order by making use of the MUSCL extrapolation formula. This is based on computing left and right states by a limiting

function. In this framework the van Albeda limiter has been chosen [22]. The primitive variables on both sides of the cellface, denoted by P^- and P^+, are defined as:

$$P^- = P_l + \frac{1}{2}\phi_l \tag{4.17}$$

$$P^+ = P_r - \frac{1}{2}\phi_r. \tag{4.18}$$

The $\phi_{l,r}$ variables represent the van Albada limiter in the left and right cells, respectively. These are defined as:

$$\phi_l(\beta_l, \zeta_l) = \frac{\beta_l^2\zeta_l + \beta_l\zeta_l^2}{\beta_l^2\zeta_l^2} \tag{4.19}$$

$$\phi_r(\beta_r, \zeta_r) = \frac{\beta_r^2\zeta_r + \beta_r\zeta_r^2}{\beta_r^2\zeta_r^2}. \tag{4.20}$$

β is a cell-centred term, which is defined as follows:

$$\beta_l = \beta_r = P_r - P_l. \tag{4.21}$$

ζ is an upwind term. This is defined by making use of the vector $\Delta\vec{x}_c$ pointing from the left to the right cell center:

$$\zeta_l = \frac{\partial P_l}{\partial x}\Delta x_c + \frac{\partial P_l}{\partial y}\Delta y_c + \frac{\partial P_l}{\partial z}\Delta z_c \tag{4.22}$$

$$\zeta_r = \frac{\partial P_r}{\partial x}\Delta x_c + \frac{\partial P_r}{\partial y}\Delta y_c + \frac{\partial P_r}{\partial z}\Delta z_c \tag{4.23}$$

Here the gradients of the primitive variables are required. For each cell in the interior of the domain, these are computed as follows:

$$\frac{\partial P}{\partial x_i} = \frac{1}{V}\sum_j \frac{P + P_j}{2} S_j n_{i,j}. \tag{4.24}$$

Here x_i points at the directional component, the sum is taken over the interfaces of the cell, the subscript j indicates the j-th neighbouring cell, while the S_j and $n_{(i,j)}$ denote the surface and the directional component i of the normal vector, respectively of the face under consideration. When a boundary cell is considered, the contribution of the borderface is replaced by the following expression:

$$\left[\frac{\partial P}{\partial x_i}\right]_{\text{borderface}} = \left[P_b - \frac{P + P_{\text{ghost}}}{2}\right] S n_i. \tag{4.25}$$

Here P_b represent the value of the primitive variables at the border, and P_{ghost} represents the value of the primitive variable in the ghost cell associated to this cell.

4.2 Adjoint solver

In this section the implementation in the numerical scheme is discussed. The adjoint equation as described in Section 3.1. Since the integral property of interest is defined at the border of

the domain, while the u vector is a cell-centered value, the equation is written as follows:

$$-\left(\frac{\partial J}{\partial \mathbf{u}}+\frac{\partial J}{\partial \mathbf{u}_b}\frac{d\mathbf{u}_b}{d\mathbf{u}}\right)^T=\boldsymbol{\lambda}\frac{\partial \mathbf{R}}{\partial \mathbf{u}}^T. \tag{4.26}$$

In order to compute the adjoint vector the inverse of the $\frac{\partial \mathbf{R}}{\partial \mathbf{u}}$ matrix must be computed. However since this is a very large sparse matrix, this has to be approximated rather than be computed directly. In *elsA* a Newton method is used, which is defined as follows:

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right)^T_{(Approximate)}\left[\lambda^{n+1}-\lambda^n\right]=-\left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right)^T_{(Exact)}\lambda^n+\left(\frac{\partial J}{\partial \mathbf{u}}+\frac{\partial J}{\partial \mathbf{u}_b}\frac{d\mathbf{u}_b}{d\mathbf{u}}\right)^T. \tag{4.27}$$

Here, the superscript n denoted the iterative stepping towards a converged solution.

4.3 Mesh sensitivity indicator

In this section the terms in order to arrive at the indicator used for the mesh sensitivity based approach for mesh adaptation are discussed. The code as developed by Todarello [18] is used. The sensitivity of a mesh point to its location is as follows:

$$\frac{dJ}{dX}=\frac{\partial J}{\partial X}+\frac{\partial J}{\partial u_b}\frac{\partial u_b}{\partial X}+\lambda^T\frac{\partial R}{\partial X}. \tag{4.28}$$

The first two terms can be regarded as geometrical derivatives, while the last part is an aerodynamic derivative. The geometrical derivatives are only nonzero if the node considered lies at or next to the support of the function J. The computation of these derivatives is rather straightforward. The aerodynamic derivative consists of two parts, the first is the adjoint vector, which is discussed in the previous section. The second term, regards the change in residual due to a change in location of the node. This is computed by computing the change in fluxes on the associated faces.

$$\frac{\partial R}{\partial X}=\sum_j\frac{\partial F_j}{\partial X}. \tag{4.29}$$

In this computation first- and second-order terms are incorporated. The terms are derived from the way the Euler equations are described in Section 4.1. As a result this term can be written as:

$$\frac{\partial F}{\partial X}=\frac{\partial F}{\partial \vec{S}}\frac{\partial \vec{S}}{\partial X}+\frac{\partial F}{\partial P^{\pm}}\frac{\partial P^{\pm}}{\partial \zeta}\left(\frac{\partial \zeta}{\partial V}\frac{\partial V}{\partial X}+\frac{\partial \zeta}{\partial \vec{S}}\frac{\partial \vec{S}}{\partial X}+\frac{\partial \zeta}{\partial \Delta\vec{x}_c}\frac{\partial \Delta\vec{x}_c}{\partial X}\right). \tag{4.30}$$

In the case a boundary node is regarded, one more contribution, containing the effect of change in the conservative value on the boundary, will be added. Thus (4.30) then becomes:

$$\left(\frac{\partial F}{\partial X}\right)_b=\frac{\partial F}{\partial \vec{S}}\frac{\partial \vec{S}}{\partial X}+\frac{\partial F}{\partial P^{\pm}}\frac{\partial P^{\pm}}{\partial \zeta}\left(\frac{\partial \zeta}{\partial V}\frac{\partial V}{\partial X}+\frac{\partial \zeta}{\partial \vec{S}}\frac{\partial \vec{S}}{\partial X}+\frac{\partial \zeta}{\partial \Delta\vec{x}_c}\frac{\partial \Delta\vec{x}_c}{\partial X}+\frac{\partial \zeta}{\partial U_b}\frac{\partial U_b}{\partial X}\right). \tag{4.31}$$

The application is done within a two dimensional framework. However, the solver is set-up for three dimensional computations. therefore, the computations are performed in pseudo-2d, consequently there are two layers of cell faces around one layer of cells of equal width. The component in the third dimension is neglected in this framework. It is found that very small perturbations in the term $\frac{dR}{dX}$ between the layers of faces occur. In the computations performed, these are found to be smaller than 10^{-7} in magnitude. For both faces around the single layer of two dimensional cells, the components have effects in negative direction to each other. Since the effect is very small, this is only observable in areas where very low other contributions of $\frac{dR}{dX}$ are apparent. In areas of large adjoint values combined with a large r of the cells, these effects can cause regions to be flagged for refinement if one uses just one layer of nodes to obtain the $\frac{dR}{dX}$ values. Therefore, since there are two layers and theoretically in pseudo-2d both should give the same results, the average value between the nodes at the same (x, y) position is used as value for $\frac{dR}{dX}$. This operation cancels this 3d error in the code.

As described in Section 3.3, the next step is to make a projection of this operator to exclude the effects of domain changes on the function of interest. A projection of $\frac{dJ}{dX}$ to the allowable direction gives a new vector of allowable displacement. Now a measure of cell size is included. It is chosen to follow the method as created for structured meshes which uses r, a measure of maximum displacement, defined as half the distance to the closest neighbouring node. This results in the form of:

$$\theta = rP\left(\frac{dJ}{dX}\right). \tag{4.32}$$

This as a final result, yields the indicator which will be used to define the areas in the mesh which will be flagged for refinement.

The method assumes that this mesh sensitivity is a scale of how much the goal-function would change if a mesh point would be located differently. Therefore, it is interesting to check if the goal-function indeed behaves as the gradient computed suggests.

The evaluation is done by comparing the adjoint found value with first order finite difference estimation of the value of interest. The hypothesis is:

$$\left(\frac{dJ}{dX}\right)_{\text{adjoint}} \approx \left(\frac{dJ}{dX}\right)_{\text{finite difference}} = \frac{J\left(X + \delta X_k\right) - J(X)}{\delta X_k} \tag{4.33}$$

In order to test this, computations are performed under transonic conditions, then a node is moved in the direction of $\frac{dJ}{dX}$ for a number of different relative lengths compared to r. Then to evaluate if the derivative $\frac{dJ}{dX}$, as obtained from the adjoint method, does indeed show similar behaviour as a first order estimation, the following coefficient in compared:

$$\psi\left(\delta X_k\right) = \frac{\left(\frac{dJ}{dX}\right)_{\text{adjoint}} - \frac{J(X+X_k)-J(X)}{\delta X_k}}{\frac{J(X+X_k)-J(X)}{\delta X_k}} \tag{4.34}$$

$$= \frac{\left(\frac{dJ}{dX}\right)_{\text{adjoint}} \cdot \delta X_k - \left[J\left(X + X_k\right) - J\left(X\right)\right]}{J\left(X + X_k\right) - J\left(X\right)} \tag{4.35}$$

If for decreasing δX_k, $\psi\left(\delta X_k\right)$ tends to zero, it shows the difference between the adjoint obtained value and the finite difference first order approximation of the derivation tend towards

the same value. In this comparison δX_k is chosen sufficiently large to exclude small distance numerical effects.
As a goal-function the lift coefficient has been chosen. Six points around the airfoil have been taken to obtain the data. The points are chosen such that in both upstream and downstream areas are considered, as well as point in the near vicinity of the shock, the location are plotted in figure 4.1. The results for the λ values are plotted in figure 4.2.

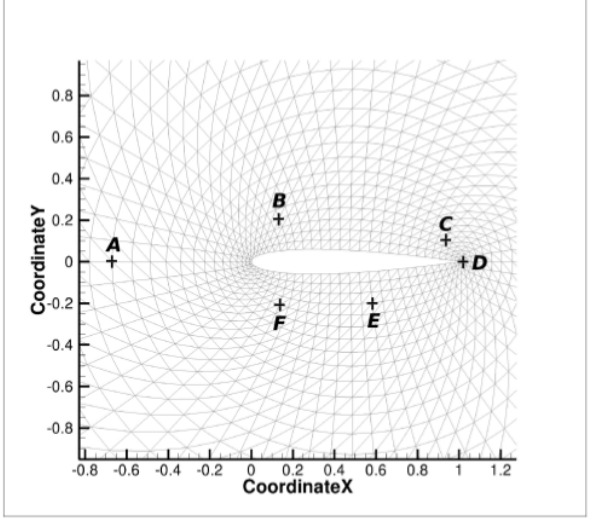

Figure 4.1: Location of the nodes used for validation of $\frac{dJ}{dX}$

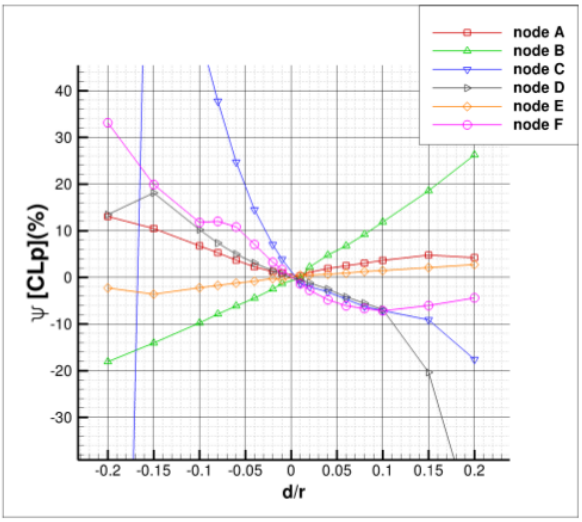

Figure 4.2: λ values for the different nodes

It can be observed that for all nodes, as the δX decreases, ψ tends to zero. Which implies that $\frac{dJ}{dX}$ is showing similar behaviour as a finite difference first order estimate. Thus confirming a correct implementation of the adjoint method to obtain the derivative value.

4.4 Adjoint-weighted residual indicator

This section describes how the error indicator as defined in Section 3.2 is computed. Both the computable correction as well as the estimated error in the computable correction are described here. For both parts the interpolation towards a fine grid is required. The fine grid is defined by splitting each face into two. Thereby effectively creating 4 cells out of every original cell. Is is shown in Figure 2.1 in Section 2.1.2.

The interpolation process is derived from the method as proposed by Venditti and Darmofal [1]. In this research small changes regarding used data points need to be applied due to the described method being focussed at nodal based values, while in the present research cell centred values are regarded. For the linear interpolation, the reference method utilizes a simple linear interpolation within each coarse grid cell. Due to the cell-centred scheme holding only one data point per cell, more data is required in order to make an interpolation.

It is chosen to use the coarse grid neighbouring cells for obtaining the additional information, while for the quadratic interpolation more data is required and therefore also the neighbours of the neighbouring cells are used. A graphical representation of the utilized cells is given in figure 4.3. The red cell represents the cell for which the data needs to be computed, for linear interpolation, data from the blue cells is also taken into account, while for the quadratic interpolation, data from all coloured cells is used. In the case a boundary of the domain is encountered a cell will be missing. However, in *elsA* on the boundary cells, a value in the center of a boundary face is available. This data point will be used replacing the missing data point. This cell centred approach has the disadvantage that it requires data from the surrouding cell, as oppose to the data at the cells corners. In regions of very strong gradients, this makes the method less accurate, as it smears the values. This is mostly apparent around the boundary of the airfoil and around the shock. The smearing of the values in the quadratic interpolation are higher that those of the linear interpolation, as more data points are required. This effect is expected to weaken the method.

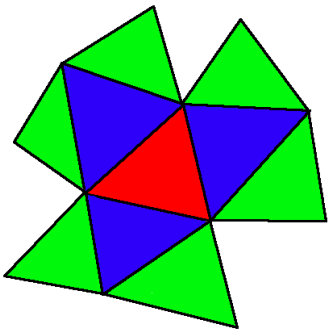

Figure 4.3: Involved cells in the interpolation process

The least-squares problem is cast in order to find the linearly interpolated values $\bar{\phi}$, by using the generic scalar variables, ϕ

$$\bar{\phi} = \sum_{i=1}^{3} N_i \phi_L^i. \tag{4.36}$$

The sum is taken over the three vertices of a triangle. Here N denote the standard shape functions for generic triangles[23]. Now the minimization problem is cast as follows, on the vector $z_L = \left\{\phi_L^1, \phi_L^2, \phi_L^3\right\}^T$.

$$\Lambda_L = \sum_{j}^{4} \left[\bar{\phi}\left(z_L, x_j, y_j\right) - \phi\left(x_j, y_j\right)\right]^2. \tag{4.37}$$

The sum will be taken over the input data points for ϕ, being the associated cells centres.

For the quadratic interpolation, the derivative functions in both directions are added:

$$\bar{\phi_x} = \sum_{i=1}^{3} N_i \phi_x^i \tag{4.38}$$

$$\bar{\phi_y} = \sum_{i=1}^{3} N_i \phi_y^i \tag{4.39}$$

While the quadratic interpolant of ϕ is defined as follows:

$$\tilde{\phi} = \sum_{i=1}^{6} \tilde{N}_i \phi_Q^i. \tag{4.40}$$

The associated vector filled with the input parameters is defined as: $z_Q = \left\{\phi_Q^1, \phi_Q^2, \phi_Q^3, \phi_Q^4, \phi_Q^5, \phi_Q^6\right\}^T$. The minimization problem for the quadratic interpolation process is defined as:

$$\Lambda_{\mathrm{L}} = \int_{\Omega_k} \left[\tilde{\phi}\left(z_Q\right) - \phi\right]^2. \tag{4.41}$$

As described in section 3.2, whereby the adjoint vector for the fine mesh is estimated by the quadratic interpolation of the adjoint vector, the error in the computable correction is estimated as follows.

$$\epsilon = \left(\widetilde{\lambda_h^H} - \overline{\lambda_h^H}\right)^T \cdot R_h\left(\overline{U_h^H}\right). \tag{4.42}$$

Here, $\overline{\bullet_h^H}$ denotes a linear interpolation, while $\widetilde{\bullet_h^H}$ denotes the quadratic interpolation of the values under consideration. By summing up the contribution from the fine cells inside a coarse cell, the contribution for every coarse cell is obtained. In the literature described, a cell-based remeshing tool is utilized. However, the current research uses the program *gmsh*, which requires nodal input, therefore it the nodal value is obtained by simple averaging the values from the surrounding cells.

4.5 Adaptation procedure

In this section the chain of procedures is described. Furthermore, the procedure that convert the error indicator to input values for the mesh refinement will be discussed and ultimately, the remeshing tool will be briefly discussed. The set-up of the adaptation chains for both methods is shown in Figures 4.4 and 4.5.

In the previous sections everything up to and including the step of computing the error indicator is elaborated. The next step is to determine the flag strategy. First, the adjoint-weighted residual one will be discussed as this is chosen to mimic the one used in literature, then the strategy of the mesh sensitivity method will be discussed.

4.5.1 Flag strategy for adjoint-weighted residual method

As a starting point an allowed error level, tol_G, is chosen. This is then transformed for an allowable level per cell, tol_L.

$$\mathrm{tol}_L = \frac{\mathrm{tol}_G}{N_{cell}}. \tag{4.43}$$

Here N_{cell} is the number of cells. Then a cell specific factor is computed.

$$\eta_k = \max\left(\frac{\epsilon_k}{tol_L}, 1\right). \tag{4.44}$$

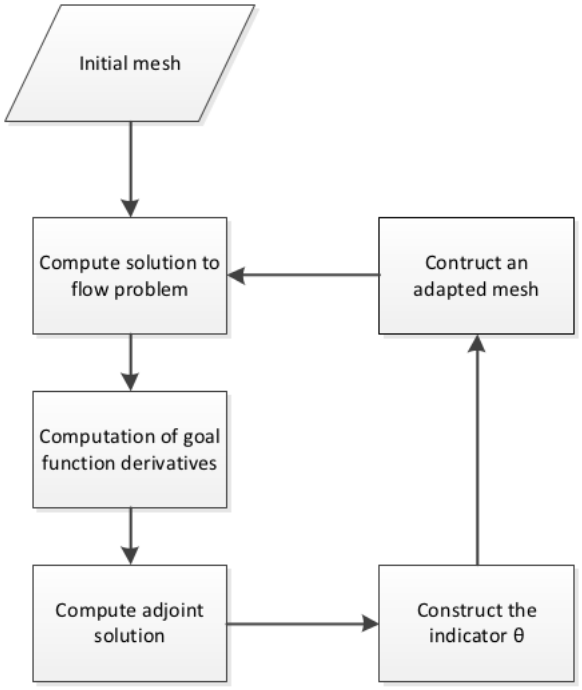

Figure 4.4: Adaptation scheme for mesh sensitivity based indicator

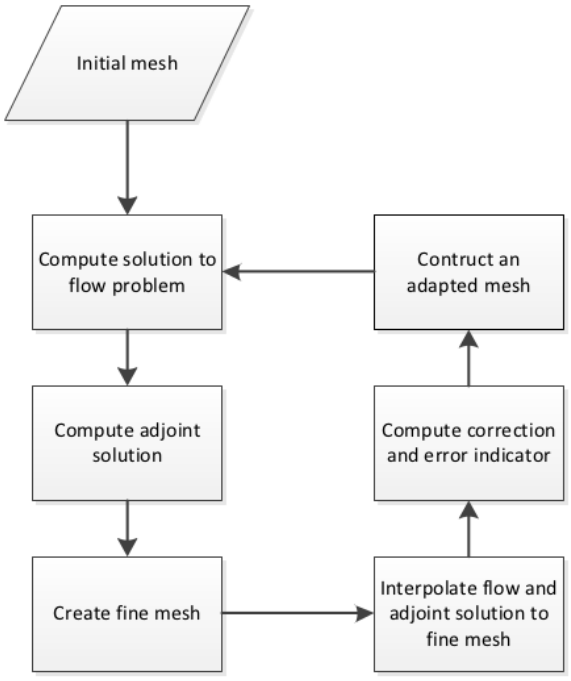

Figure 4.5: Adaptation scheme for mesh adjoint-weighted residual indicator

Following the method as described in [1], if more than half the cells are flagged for refinement, the tol_L is adjusted to that exactly half the number of cells is flagged for refinement. In the original research, also a global factor is used. However, in the current research it is found that this led to ineffective mesh refinement in areas of very low significance to the solution. This is possibly due to the large far-field extension of 150 cords in the current research. The new target cell size is computed as follows:

$$r_{\text{target}} = r_{\text{old}} \left(\frac{1}{\eta_k} \right)^{\omega}. \tag{4.45}$$

This leads to a vector of target cell size for all locations in the computational domain. For all vertices not exceeding the tolerance, this leads to an unchanged grid. The vector ω is taken from literature [1] and is set at $\frac{1}{4}$.

4.5.2 Flag strategy for mesh adaptation

The biggest difference between the V&D method and the theta method, lies in the nature of the estimator. This implies that this indicator does not show a direct link to the error, therefore the quantity of the tolerance level is more difficult to determine. In the current research, the choice is made to target half the number of nodes for refinement by setting this as the tolerance level. This condition is also apparent in the V&D method, with the exception

if a lower number of cells is sufficient to achieve the required accuracy. Next the adaptation vector will be determined in the same manner as for the V&D method:

$$f_k = \max\left(\frac{\theta_k}{\text{tol}}, 1\right). \tag{4.46}$$

The new target size for the mesh is defined as follows.

$$r_{\text{target}} = r_{\text{old}}\left(\frac{1}{f_k}\right)^n. \tag{4.47}$$

The factor n needs to be determined. This is done by evaluating how f_k scales with r. Remember $f_k \propto \theta = rP\left(\frac{dJ}{dX}\right)$, thus the scaling of $\frac{dJ}{dX}$ must be determined. From a theoretical point of view this is performed for structured grids, with an assumed grid type of four rectangular cells per vertex in [16] leading to a scaling of $\frac{dJ}{dX} \propto r^2$. However, these assumptions are not valid for this case, also it is advantageous to check the results from a test. Therefore the relation is empirically determined by utilizing a series of grid with increasing number of cells and thereby decreasing cell sizes. In order to asses the scaling of $P\left(\frac{dJ}{dX}\right)$ with the cell size, the average value, $\frac{1}{N_{\text{nodes}}}\sum_{Nodes} P\left(\frac{dJ}{dX}\right)$, is plotted against the number of nodes, as the domain remains the same, while the shape of the mesh does aswell, this results in decreasing cell size. The results are plotted in figure 4.6.

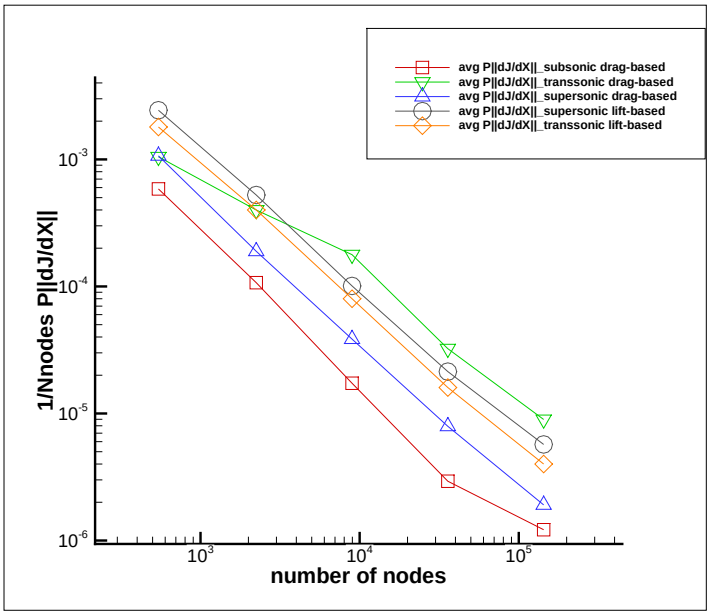

Figure 4.6: Asymptotic behaviour of average $P\left(\frac{dJ}{dX}\right)$ as the cell size decreases

A power fit is used to determine the order of convergence. Over the different flow regimes and goal-functions, the convergences rate ranges from 1.7 to 2.3, therefore the following realtion is assumed: $\frac{dJ}{dX} \propto r^2$. This implies $\theta \propto r^3$ and thus the choice is made for $n = 3$ in (4.47).

To be able to also assume a general scaling of $\frac{dJ}{dX}$ with r, the shape of the fields should remain of constant nature over the decreasing values of r. In order to compare the $\frac{dJ}{dX}$ field normalized for cell size, the field of $\frac{1}{S_l}P\left(\frac{dJ}{dX}\right)$ is plotted for a various number of r. The

associated fields and the meshes on which these are created are plotted in Figure 4.7. The general shape of the field seems to remain constant with decreasing mesh size, although small deviations in the field are apparent.

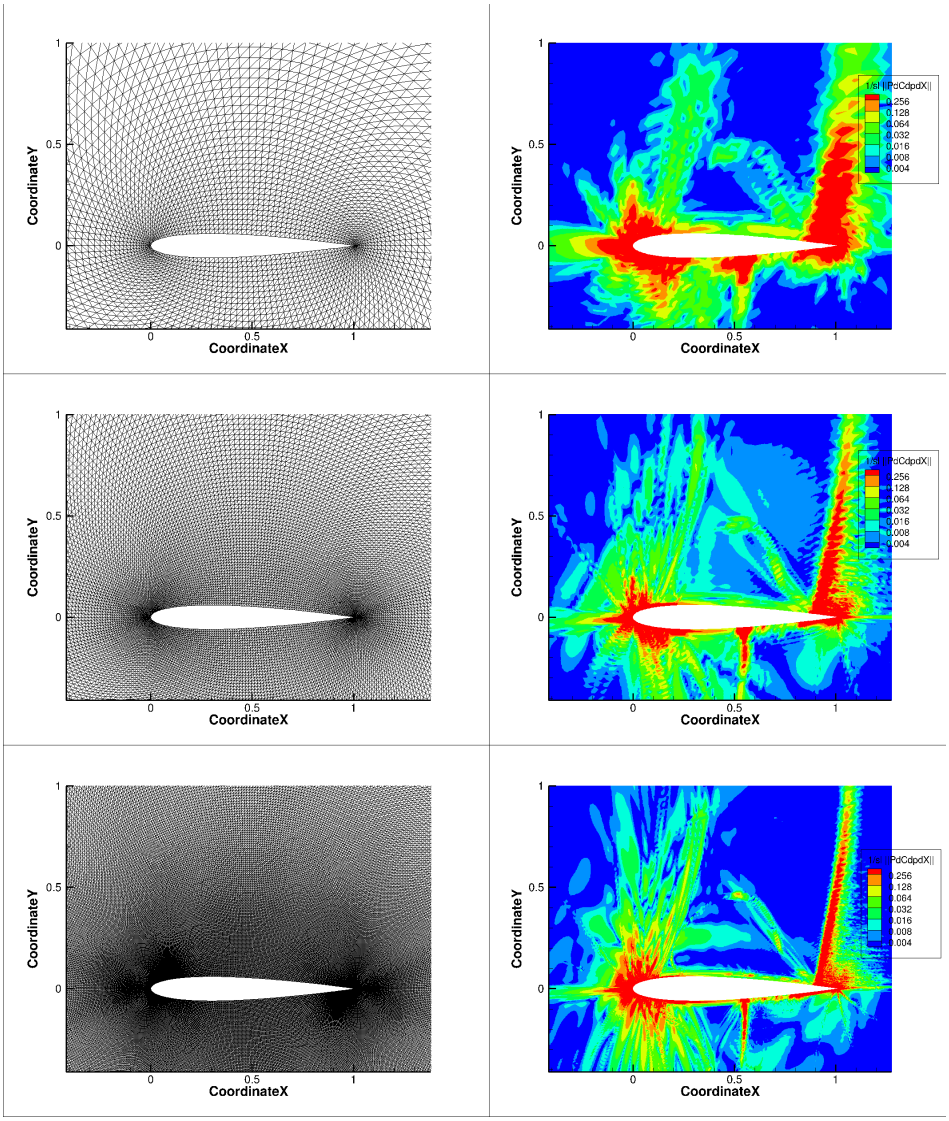

Figure 4.7: Behaviour of the $\frac{dJ}{dX}$ field for decreasing r

4.5.3 Remeshing

For each iteration a new mesh is generated by making use of the program *MMG2D*, which is the two dimensional version of *MMG3D* [24]. As input parameters, the boundaries of the domain are given and furthermore an input for the desired cell size must be provided. For both methods a vector of desirable new mesh size for each location in the field is available. In areas where the tolerance is not surpassed, the mesh remains almost the same.

Finally, it is found that very irregular meshes give convergence problems when simulating the flow in *elsA*. The output meshes from *MMG2D* during the adaptation are showing these problems, therefore an additional smoothing is applied. This is performed by averaging the target cell size difference with all surrounding vertices. The disadvantages is that the refinement of exact locations is sacrificed.

Chapter 5

Results

In this chapter a comparison of the results for both mesh adaptation strategies is presented. This is done by comparing the results of the goal-function for several mesh adaptation steps using both approaches. Within this section, one case is used to compare the results for the reference method as implemented in the current framework to the results of the same method as found in literature. Then, a total of three cases are used to compare the results as obtained from the *theta* en V&D method. Finally, the difference computational costs of both methods will be discussed.

5.1 Goal-function comparison

In aeronautical application of CFD simulations, the goal often is to determine the lift and drag coefficients. Therefore, the mesh adaptations in this research are performed with regard to the lift and drag coefficients of a NACA0012 airfoil. The results are tested in three different flow regimes, two of which are transonic and one in the supersonic flow regime. These cases are chosen such that these can be compared to the resulting meshes of the studies performed by Dwight[9] and Venditti and Darmofal[1]. The number of mesh points required to obtain certain goalvalues can however not be compared due to different farfield radius of the utilized mesh. Therefore, the results of the implementation of both methods in the current framework is used to compare the accuracy of the goalvalues versus the number of nodes required. The first case will also be used to make a quick comparison between the reference method as implemented in this research towards the reference method as implemented by its original developers.

5.1.1 Transonic flow $M_\infty = 0.95$, AoA $= 0^\circ$

The first case has a freestream velocity near the speed of sound, with no angle of attack. The case is derived from Venditti and Darmofal. This case will also be used to compare results for the V&D method as obtained implemented in the current research with those as obtained in the literature [1]. The flow accelerates over the airfoil and becomes supersonic. A shock wave is formed at both the upper and lower side around the trailing edge of the airfoil. As the airfoil is symmetric and has zero angle of attack, the lift has a trivial theoretical value of zero.

THe limiting value for the drag has been shown by other flow solvers to be around 1098 drag counts. Unfortunately, it was not possible to verify this number by the use of regularly refined grid, as the grid containing around 80.000 nodes is still at 1110 drag counts. Further refined grid, containing around 320.000 nodes, showed convergence problems.

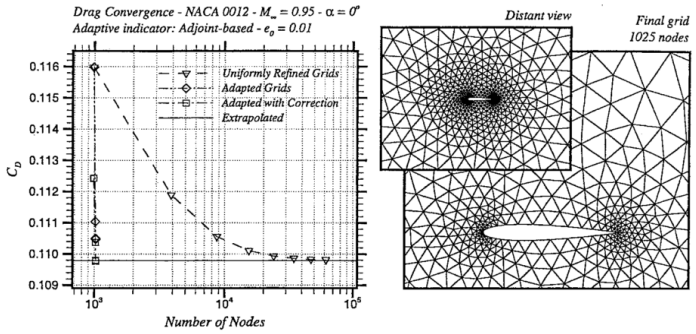

Figure 5.1: Results for C_{d_p} for $M = 0.95$ $AoA = 0^\circ$, as found by Venditti and Darmofal [1].

Assessment of implementation of reference method

The implementation of the reference method differs slightly from the original set-up, as described in section 4.4. Hence, it is expected that the results will differ. Furthermore, as no far field boundary was given in the original research [1] the number of nodes cannot be compared on a one-to-one basis. The number of nodes on the airfoil boundary in the initial step is of equal size. The literature first uses a large maximum error level of 100 drag counts[1]. The results for drag based adaptation are shown in Figure 5.1.It requires only one adaptation step and only 50 nodes added and gives accuracy of higher than one drag count. Also, the adapted mesh without correction requires just two steps and very few additional nodes to obtain an accuracy of less than 10 drag counts from the limiting value. The results from the current research are shown in figure 5.2. The convergence is much slower. The corrected value does not arrive within one drag count after five steps and 5000 nodes added. It can thus be concluded that the results are not as strong as found in the literature. This does not necessary originate from the adaptation procedure alone. It is very well possible the utilized flow solver also influences these results up to a certain extent. This hypothesis is supported by the fact that when the uniformly refined grids are compared, many more nodes are required before comparable goal values are obtained. Furthermore, there have been some convergence problems even with regularly refined grids. The current research also required many more

nodes to arrive at comparable goal values. Possible sources of this difference will be further discussed in chapter 7.

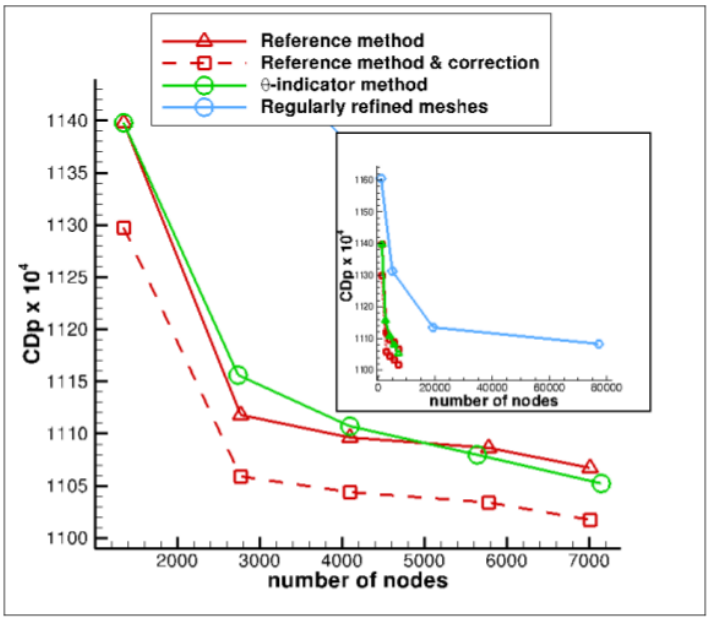

Figure 5.2: Convergence behaviour for C_{d_p} for θ- and reference method based mesh adaptation for $M = 0.95$ $AoA = 0°$

Comparison of methods

It can be observed that the corrected value of the reference method is more accurate compared to the goalvalue from a mesh with a comparable number of nodes as generated by the θ method. From this case it becomes apparent that even though the reference method is not fully efficiently implemented, as found in the previous section, it is still considerable more accurate when compared to the θ method. As is expected, both method show an enormous improvement compared with regularly refined mesh.

5.1.2 Transonic flow $M_\infty = 0.85$, AoA $= 2°$

The second case is within the transonic flow regime, with a small angle of attack. The lift based adaptation shows a decrease in accuracy during the first step. This is probably due to the mesh being too coarse during this first step to simulate dominant flow features. After three adaptation steps both the reference method with correction term and the θ-based method are within 1 lift count of the limiting value, which is considered to be sufficiently accurate in the current research. Both methods use almost the same number of nodes to obtain these results. The θ method seems to be slightly more accurate in this case. This case shows comparable efficiencies towards number of nodes require to obtain a sufficiently accurate goal-function. The adjoint-weighted residual without correction term shows slower convergence, this is due to the mesh adapted towards a optimal computation with the correction term.

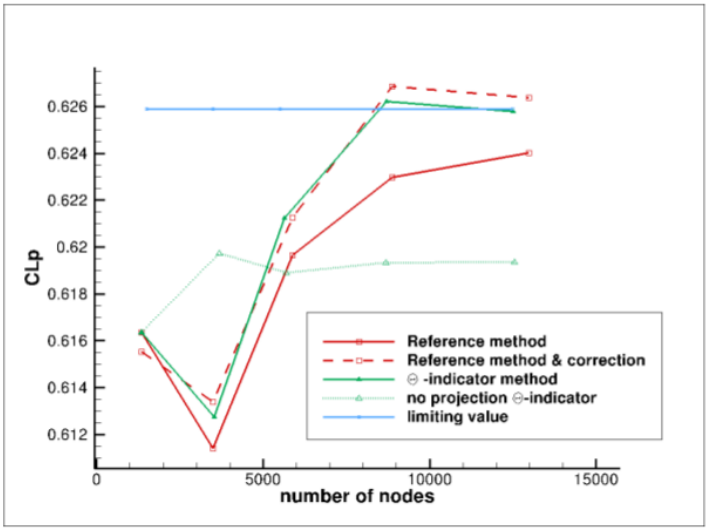

Figure 5.3: Convergence behaviour for C_{l_p} for θ- and reference method based mesh adaptation for $M = 0.85$ $AoA = 2°$

5.1.3 Supersonic flow $M_\infty = 1.5$, AoA $= 1°$

The last case is well within the supersonic flow regime, the case is derived form Dwight [9]. The information only travels in one direction for the largest part of the domain. Due to this hyperbolic nature of the flow, goal-oriented adaptation methods are expected to be very efficient compared to feature based or global refinement methods. The flow solver seems to handle this flow case more easily as compared to the ones in the transsonic flow regime. For both the lift and drag, the regularly refined grids already show convergence behaviour of sufficient quality to enable a Richardson extrapolation to be performed. This gives the following accuracy intervals for the goal-function for lift [0.05468 , 0.0548] and drag [0.09761 , 0.09711].

First, the results for lift based adaptation are shown in figure 5.4. Both methods are much more efficient than global refinement and show nice convergence behaviour. The θ method is a little less accurate compared to the reference method with correction.

Considering the drag-based adaptation as shown in figure 5.5, both goal-oriented approaches are again much more efficient compared to global refinement. However, the results for drag are more difficult to interpret. The drag values from both methods seem to overestimate the drag slightly. It takes until the sixth adapted mesh for the reference method with the correction term to arrive within the accuracy interval, while the θ method at this step with comparable number of nodes is around one lift count above the upper level of the accuracy interval. In this case, the reference method certainly gives better results compared to the theta method.

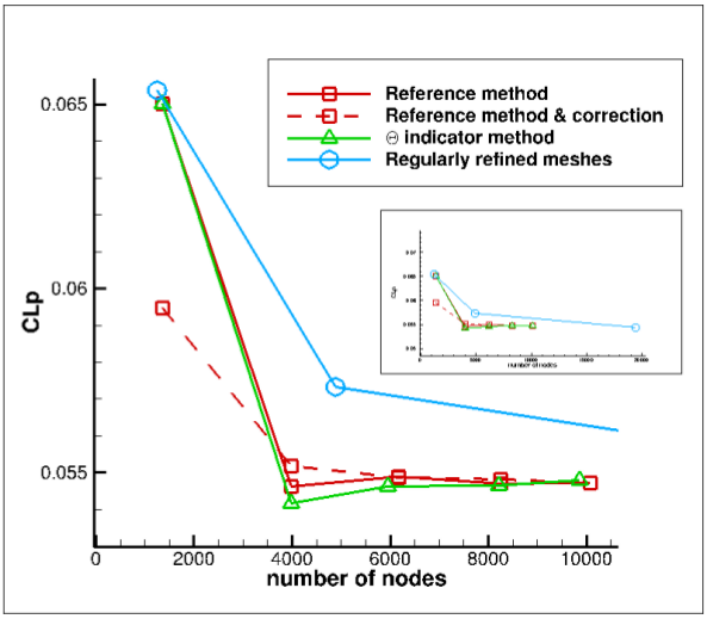

Figure 5.4: Convergence behaviour for C_{l_p} for θ- and reference method based mesh adaptation for $M = 1.5$ $AoA = 1°$

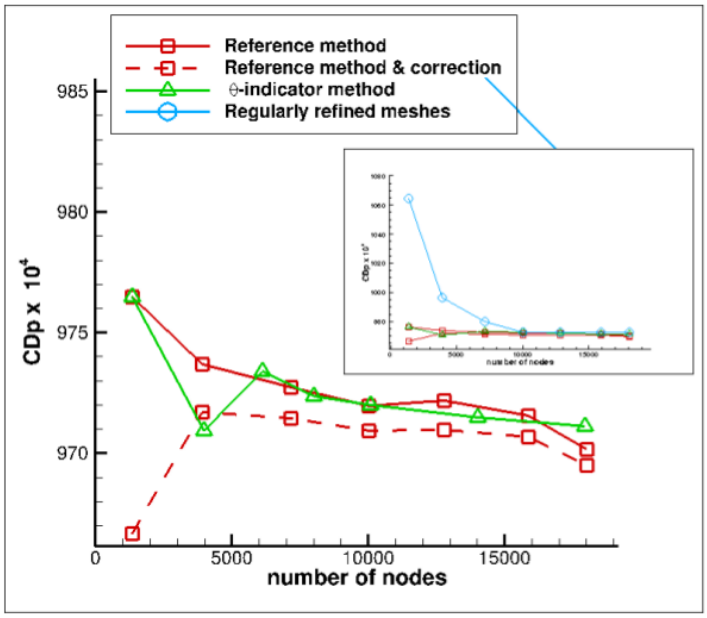

Figure 5.5: Convergence behaviour for C_{d_p} for θ- and reference method based mesh adaptation for $M = 1.5$ $AoA = 1°$

5.2 Computational costs comparison

In order to compare the efficiency of both methods, it is also required to compare the computational costs. Unfortunately, this cannot be done in a rigorous way. As the adaptation

chains consists of many steps and the computational time for some are very dependent on simulated flow it is not possible to present a complete comparison. Furthermore, a large part of the computations are performed on a cluster for larger computational effort. However, this prohibits a comparison of the computational time to be between different operations. Nonetheless, a very rough estimation will be drawn identifying the largest contributors to the computational time.

The adaptation chain for both methods is given in section 4.5. The first step for computing the solution to the flow problem is equal for both. Next, the θ method requires derivatives of goal-functions, this operation has computational costs which are far lower compared to those of computing the flow or adjoint solution. Subsequently, the same adjoint problem has to be solved for both methods. It is found that in the current framework, solving the adjoint equations takes considerably more computational effort as compared to solving the flow problem. This lies in the order of 2 to 10 times more effort. Finally, the step required to compute θ and the interpolation process to estimate the error can be neglected in computational effort compared to solving for the flow and the adjoint problem.

One important note should be taken with the regard of computational time. The value of the corrected value for the reference method requires the adjoint problem to be solved, while the θ method does not require the adjoint to be solved at the final mesh level. Therefore, the reference method requires this last adjoint computational effort additional compared to the θ method. Since solving the adjoint equations in the current framework takes considerably more effort compared to solving the flow equations for the samen number of volumes, this constitutes to considarable more computational effort for this method. It can roughly be compared to half an additional cycle of the adaptation chain.

Chapter 6

Conclusions

This chapter aims to answer the research question as posed in section 2.5:

Is goal-oriented mesh adaptation using a mesh sensitivity strategy an efficient and effective method to obtain integral properties of interest for the Euler equations using finite volume methods for two dimensional unstructured grids?

In this research the first aim is to compare the theoretical basis of both approaches. It is found the same adjoint function is solved. The remaining part for both methods concern the residual, in the reference method a least squares fitting is performed to estimate values for a fine grid. While, the mesh sensitivity method utilizes the partial derivative of the residual with respect to the node location and multiplied by the distance to the nearest midpoint of surrounding cells. This effectively estimates the magnitude of the residual at the aforementioned distance from the node. The reference method utilizes first order effects to correct the computed goalvalue and adepts the mesh by estimating the error due to second order effects, while the mesh sensitivity indicator takes only first order effects into account.

In order for the mesh sensitivity based indicator to be used as mesh refinement indicator it is required to for the indicator to show constant asymptotic behavior as the mesh size tends to zero. A number of flow cases, with decreasing cell size have been tested and the results show reasonable constant decreasing behavior for the indicator. Also, the general shape of the indicator field remains constant. This behavior supports the hypothesis for using the θ as a refinement indicator.

Following the theoretical comparison, an empirical study is performed. The value of the goal-functions from both methods are compared for comparable number of nodes and adaptation steps. It is found that the results from the implemented reference method are not as effective as found in the original literature. This may be due to the following reasons: The method is changed to be applicable to cell-center based values instead of vertex-based values. The altered method of interpolation causes additional diffusion of information for the interpolated

flow field. The farfield boundary of the original method is unknown, hence the exact number of nodes cannot be compared. Also, the flow solver is different, since *elsA* is originally written for structured purposes, there is a possibility of less accurate solution for unstructured purposes. Finally, errors may be induced in implementing the method. Comparing the reference method as implemented in the current framework with the mesh sensitivity-based adaptation chain, two flow cases for which lift is computed show very similar results. While, in the two cases drag is considered, the reference method is clearly more accurate. Here, the results show that often two or even three adaptation steps less are required to obtain comparable accuracy in the results. Finally, it can be concluded both method require very comparable amount of computational effort for every cycle. However, the reference method only obtains its value after an additional half of the cycle, since the solution of the adjoint equations needs to be known in order to obtain the corrected value.

Overall, it found the reference method has a stronger theoretical link with the error in the goal-function. Also, by estimation of second-order effects, it takes into account a higher number of degrees of freedom. This leads to a higher expected efficiency of the method. The small empirical study does seem to support this hypothesis.

Chapter 7

Recommendations

While performing this research a number of choices are made on the approach to answer the research question. The chosen approaches have their advantages and disadvantages. Having the knowledge of their outcome a number of recommendations can be made as how to improve the answer to the research question. Broadly, these will be categorized in three categories. Firstly, improving the implementation of the reference method. Secondly, draw an additional comparison between the methods. Finally, improve the quality of the numerical comparison.

7.1 Improvement of reference method

As indicated in section 4.4 the interpolation process needs to be changed from how it is implemented in the literature. This is due to the current research uses cell centered values rather than node based values. The changes as made in this research lead to some diffusion of information in this process. This likely decreases the strength of the method. A recommendation for future research would be to implement the method on node based refinement.

7.2 Drawing an additional comparison

In section 3.4 it is concluded the mesh sensitivities based adaptation method uses an indicator which takes only first order effects into account. It is found to be comparable with the linear part the reference method. The linear part in the reference method is used as correction term rather than indicator for mesh refinement. It can be interesting to compare the fields of the linear part of the reference method with the mesh sensitivity indicator. This can show a more clear comparison between the methods, because the same number of degrees of freedom is applied for both approaches.

7.3 Improvement of the numerical comparison

The results as found in this research are not very satisfactory. The flow solver has some convergence issues. For some cases it is not possible to compute limiting value using the current set-up of the flow solver. This prevents the use of smaller accuracy intervals. Furthermore, a strong smoothing operator is required in order for the adapted meshes to be able to converge to satisfactory residual levels. This diffuses the effects of the indicator thereby reducing its effect. If a more robust flow solver is utilized for the comparison, a smaller accuracy interval could be used, possibly leading to more information on the effectiveness of both mesh adaptation approaches. Finally, if the amount of smoothing is lowered or possible removed, the effects of both adaptation approaches become stronger and thereby the comparison will become clearer.

Furthermore, the computations should all be performed on the same computer so the adaptation chain can be timed. This will give more information on the required computational effort of both methods.

Finally, as the different flow cases and targeted goal values give a different conclusion it is interesting to increase the number of flow cases. This will strengthen the conclusion towards difference in the efficiency of both methods.

Bibliography

[1] David A. Venditti and David L. Darmofal. Grid adaptation for functional outputs: Application to two-dimensional inviscid flows. *Journal of Computational Physics*, 176(1):40–69, February 2002.

[2] R Becker and R Rannacher. Weighted A Posteriori Error Control in FE Methods. In *ENUMATH-97, Heidelberg*, pages 621 – 637. Citeseer, 1998.

[3] Niles A. Pierce and Michael B Giles. Adjoint Recovery of Superconvergent Functionals from PDE Approximations. *SIAM Review*, 42(2):247–264, 2000.

[4] Ralf Hartmann and Paul Houston. Adaptive Discontinuous Galerkin Finite Element Methods for the Compressible Euler Equations. *Journal of Computational Physics*, 183(2):508–532, December 2002.

[5] David A. Venditti and David L. Darmofal. Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Quasi-One-Dimensional Flow. *Journal of Computational Physics*, 164(1):204–227, October 2000.

[6] David A. Venditti and David L. Darmofal. Anisotropic grid adaptation for functional outputs: Application to two-dimensional viscous flows. *Journal of Computational Physics*, 187(1):22–46, May 2003.

[7] Marian Nemec, Eloret Corp, Moffett Field, Michael J Aftosmis, and Mathias Wintzer. Adjoint-Based Adaptive Mesh Refinement for Complex Geometries. In *AIAA Aerospace Sciences Meeting*, number January, pages 1–23, 2008.

[8] Richard P Dwight. Goal-oriented mesh adaptation using a dissipation-based error indicator. *International journal for numerical methods in fluids*, 56:1–6, 2006.

[9] Richard P Dwight. Heuristic a posteriori estimation of error due to dissipation in finite volume schemes and application to mesh adaptation. *Journal of Computational Physics*, 227(5):2845–2863, 2008.

[10] Antony Jameson, W Schmidt, and E Turkel. Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes. *AIAA paper*, 1259, 1981.

[11] Krzysztof J Fidkowski and P.L Roe. An entropy adjoint approach to mesh refinement. *SIAM Journal on scientific computing*, 32(3):1261–1287, 2010.

[12] Krzysztof J Fidkowski, Marco A Ceze, and P.L Roe. Entropy-Based Drag-Error Estimation and Mesh Adaptation in Two Dimensions. *Journal of Aircraft*, 49(5):1485–1496, September 2012.

[13] Michael B Giles and Niles A. Pierce. Adjoint equations in CFD : duality , boundary conditions and solution behaviour. *AIAA paper*, 1850, 1997.

[14] Eric J Nielsen and Michael Andrew Park. Mesh Sensitivities in Computational Design. *AIAA JOURNAL*, 44(5):948 – 953, 2006.

[15] Jacques E V Peter, Maxime Nguyen-Dinh, and Pierre Trontin. Goal oriented mesh adaptation using total derivative of aerodynamic functions with respect to mesh coordinates With applications to Euler flows. *Computers & Fluids*, 66:194–214, August 2012.

[16] Maxime Nguyen-Dinh, Jacques E V Peter, Renaud Sauvage, Matthieu Meaux, and Jean-Antoine Désidéri. Mesh quality assessment based on aerodynamic functional output total derivatives. *European Journal of Mechanics - B/Fluids*, 45:51–71, May 2014.

[17] Maxime Nguyen-Dinh. *Qualification of numerical simulations by anisotropic mesh adaptation*. Phd, Universit de Nice-Sophia Antipolis, 2014.

[18] Giovanni Todarello. Goal-oriented adaptation of unstructured meshes. Technical report, TU Delft, Delft, 2014.

[19] P.L Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43:357–372, 1981.

[20] Ami Harten. On a class of high resolution total-variation-stable finite-difference schemes. *SIAM Journal on scientific computing*, 21(1):1–23, 1984.

[21] P.L Roe. Characteristic-Based Schemes for the Euler Equations. *Annual Review of Fluid Mechanics*, 18:337–365, 1986.

[22] G. D. Van Albada, B. Van Leer, and W. W. Roberts. A comparative study of computational methods in cosmic gas dynamics. *Astronomy and Astrophysics*, 108:76–84, 1982.

[23] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method: The basis*. 2000.

[24] Cécile Dobrzynski. MMG3D: User Guide REPORT. 2012.