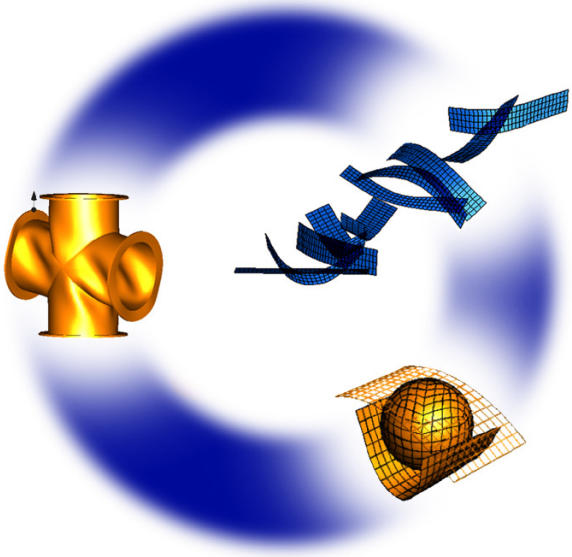
Technische Universität München

Analytical and Numerical Approaches for the Computation of Aeroelastic Sensitivities Using the Direct and Adjoint Methods

Lukas Scheucher, B.Sc.

Master Thesis



SUPERVISOR:
Dr.-Ing. Alexander Popp, Dr.-Ing. Georg Hammerl
popp@lnm.mw.tum.de

Institute for Computational Mechanics
Prof. Dr.–Ing. W. A. Wall
Technische Universität München
Boltzmannstraße 15
85748 Garching b. München (Germany)

Own work declaration

Hereby I confirm that this is my own work and I referenced all sources and auxiliary means used and acknowledged any help that I have received from others as appropriate.

(location, date) (signature)

Abstract

This is the most often read section. It will determine if someone actually bothers to read more of what you've written. Only write this section after you have completed your report. The abstract consist of only one paragraph, mostly limited to 200-300 words. Literally, a summary of your work. Write a sentence or two about each of the main sections of your report, as discussed in this section. When summarizing results, make the reader aware of the most important results (including numbers when applicable) and important conclusions or questions that follow from these.

Zusammenfassung

german version of your abstract.

Contents

1 Fluid mechanics **1**
1.1 Eulerian and Lagrangian view . . . 1
1.1.1 Material derivative . . . 1
1.1.2 Eulerian derivative . . . 1
1.2 Basic equations of fluid mechanics . . . 2
1.2.1 Incompressible Navier Stokes Equations . . . 2
1.2.2 Compressible Navier Stokes Equations . . . 3
1.2.3 Euler equations . . . 3
1.2.4 Conservative Form . . . 4
1.2.5 Euler equations . . . 6
1.2.6 Equations of State . . . 6
1.2.7 Reynolds Averaged Navier-Stokes Equations . . . 7

2 Numerical solution **9**
2.1 General . . . 9
2.2 Finite Volume method for fluid mechanics . . . 10
acro:FVE 2.3 Mixed Finite Volumes (FV)-Finite Elements (FE) formulation . . . 10
2.4 Derivation of the mixed FV-FE formulation . . . 13
2.4.1 Geometric considerations . . . 14

3 The immersed boundary method **15**

4 Fluid structure interaction **17**
4.1 Coupled three-field formulation . . . 17
4.1.1 Staggered algorithm . . . 19

5 Optimization **23**
5.1 Optimization Model . . . 24
5.2 Design Model . . . 26
5.3 Analysis Model . . . 27

acro:SA **6 Fluid Sensitivity Analysis (SA)** **31**
6.1 Connection between discretization and differentiation . . . 31
6.2 Sensitivity derivation . . . 31
6.2.1 Calculation of $\frac{\partial q_j}{\partial s_i}\Big|_{\boldsymbol{w}_0}$ . . . 32
6.2.2 Calculation of $\frac{\partial q_j}{\partial \boldsymbol{w}}\Big|_{\boldsymbol{w}_0}$ . . . 33
6.2.3 Calculation of $\frac{d\boldsymbol{w}}{ds_i}\Big|_{\boldsymbol{w}_0}$ . . . 33
6.3 Full sensitivity equation . . . 35

7 Derivatives of the fluid residual **37**
7.1 Fluid Jacobian $\frac{\partial \mathbf{R}}{\partial \mathbf{w}}$ . . . 37
7.1.1 Derivative of the convective Jacobian $\frac{\partial \mathbf{R}^c}{\partial \mathbf{w}}$ . . . 37
7.1.2 Derivative of the viscous Jacobian $\frac{\partial \mathbf{R}^c}{\partial \mathbf{w}}$ . . . 38
7.2 Analytic Jacobian of the Roe flux . . . 39
7.3 Derivative with respect to the abstract variable $\frac{\partial \mathbf{R}}{\partial s}$ . . . 47
7.3.1 Convective contribution $\frac{\partial \mathbf{R}^c}{\partial s}$ . . . 47
7.3.2 Viscous contribution $\frac{\partial \mathbf{R}^v}{\partial s}$ . . . 48

8 Aero-elastic Sensitivity Analysis **49**
8.1 Direct vs. adjoint approach . . . 50
8.1.1 Direct Sensitivity Analysis for the Euler equations . . . 51
8.1.2 Adjoint Sensitivity Analysis for the Euler equations . . . 53

9 The Embedded Boundary Method **55**
9.1 Setup . . . 55
9.2 Evaluation of the inviscid term at the interface . . . 55
9.2.1 FIVER-1 . . . 56
9.2.2 FIVER-2 . . . 57
9.3 Evaluation of the viscous terms at the interface . . . 57
9.3.1 Reconstruction of the fluid-state at ghost-nodes . . . 58
9.4 Evaluation of forces . . . 58
9.5 Level-set method . . . 59

10 Verification **61**
10.1 Verification approach . . . 61
10.2 Verification of Mach-sensitivity . . . 62
10.3 Verification shape sensitivity . . . 71
10.4 Work in progress . . . 81

List of Figures

1 FV semi-discretization of an unstructured mesh. Vertex i is the center of dual cell C_i. The primal element is drawn in thick black lines, the dual cell in dashed lines. The boundary of the dual cell is denoted as ∂C_{ij} . . . 11
2 One Triangle T out of the set of triangles $\sum_T$, that are connected to vertex i. C denotes the dual cell related to vertex i. . . . 14
3 Visualization of immersed boundary method versus embedded boundary method . . . 16
4 Staggered algorithm scheme . . . 20
5 Optimization block-diagram . . . 23
6 Design-model: geometrical approach . . . 28
7 Design element visualization . . . 29
8 Sensitivity Analysis approaches . . . 32
9 aa . . . 48
10 FIVER setup . . . 56
11 Riemann Sketch . . . 57
12 Ghost Point population . . . 59
13 Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ Euler equations body-fitted . . . 63
14 Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ Laminar equations body-fitted . . . 64
15 Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ RANS equations body-fitted . . . 65
16 Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ Euler equations embedded . . . 66
17 Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ Laminar equations embedded . . . 67
18 Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ RANS equations embedded . . . 68
19 Comparison of analytic $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ for all equation types body-fitted . . . 69
20 Comparison of analytic $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ for all equation types embedded . . . 70
21 Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ Euler equations body-fitted . . . 72
22 Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ Laminar equations body-fitted . . . 73
23 Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ RANS equations body-fitted . . . 74
24 Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ Euler equations embedded . . . 75
25 Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ Laminar equations embedded . . . 76
26 Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ RANS equations embedded . . . 77
27 Comparison of analytic $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ for all equation types embedded-fitted . . . 78
28 Comparison of analytic $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ for all equation types embedded-fitted . . . 79
29 Validation of the Lift and Drag results for mach-sensitivity: body-fitted Eulerequations . . . 82
30 Validation of the Lift and Drag results for mach-sensitivity: body-fitted Laminarequations . . . 82
31 Validation of the Lift and Drag results for mach-sensitivity: body-fitted RANSequations . . . 83
32 Validation of the Lift and Drag results for mach-sensitivity: body-fitted Eulerequations . . . 83

33 Validation of the Lift and Drag results for mach-sensitivity: body-fitted Laminarequations 84
34 Validation of the Lift and Drag results for mach-sensitivity: body-fitted RANSequations 84

Nomenclature

sec:nomenclature

Operators

Algebraic operations

A^T	Transpose of a tensor	`T`
A^{-1}	Inverse of a tensor	`inv`
$\bar{a}$	Average component of a quantity	`av`
a'	Fluctuating component of a qunatity	`fluc`
$\|a\|$	Norm of a quatntity	`norm`

Analytic operations

$\ddot{A}$	Second time derivative at a fixed refrence position	`dderivtime`
$\dot{A}$	First time derivative at a fixed reference position	`derivtime`
$\frac{\partial A}{\partial B}$	partial derivative of one argument with respect to the other	`pdfrac`
$\frac{\partial^2 A}{\partial B^2}$	Second partial derivative of one argument with respect to the other	`ppdfrac`
$\frac{DA}{DB}$	Material time derivative	`mfrac`
$\frac{\mathrm{d}A}{\mathrm{d}B}$	First total derivative of one argument with respect to the other	`tfrac`

Representation of scalars, tensors and other quantities

a	Scalar quantity	`s`
$\boldsymbol{a}$	Vector quantity	`v`
$\boldsymbol{A}$	Matrix quantity	`m`
A, a	continous quantity	`c`
A, a	discrete quantity	`d`
$a^{(k)}$	Iteration index in the optimization loop	`it`
$a^{(k)}$	Iteration index in the optimization loop	`ito`
$a^{(n)}$	Iteration index in thestaggered algorithm	`its`
$\bar{A}$	Fictious entity	`fic`
$\mathcal{O}(A)$	Order of magnitude	`order`

Symbols

3-field formulations

$\mathbf{T}_p$	Interface projection matrix from fluid to structure mesh	`ifaceprojFt`
$\boldsymbol{H}_2$	Second order Jacobian	`jactwo`
$\mathbf{S}$	Turbulence term	`turbmat`
$\mathbf{T}_u$	Interface projection matrix from structure to fluid mesh	`ifaceprojSt`
$\mathbf{R}^c$	Convective part of the flux matrix	`fluxmatconv`
$\mathbf{w}$	Discrete fsluid state vector	`dfstate`
$\mathcal{D}_{gov}$	State equation of the mesh	`EOSmesh`
$\boldsymbol{P}$	Fluid load TODO	`load`
$\boldsymbol{a_u}$	Adjoint structure displacement	`structdispa`
$\boldsymbol{w}$	Fluid state vector	`fstate`
$\boldsymbol{P}_F$	Fluid load	`fload`
$\mathcal{S}_{gov}$	State equation of the structure	`EOSstruct`
$\boldsymbol{a_w}$	Adjoint fluid state vector	`fstatead`
$\tilde{\boldsymbol{w}}$	Primitive ffluid state vector	`fstateprim`
$\dot{\boldsymbol{x}}$	Fluid mesh motion	`mms`
$\mathbf{G}$	diffusive part of the flux matrix	`fluxmatdiff`
$\mathcal{F}_{gov}$	State equation of the fluid	`EOSfluid`
S	Source term in the Reynolds-Averaged Navier-Stokes (RANS) equations	`turbulences`
$\boldsymbol{a_x}$	Adjoint fluid mesh motion	`mmsad`
$\boldsymbol{w}_{RANS}$	Augumentes fluid state vector in the RANS formulation	`fstaterans`
$\boldsymbol{P}_T$	Structure load	`sload`
$\boldsymbol{u}$	Structure displacement	`structdisp`
$\mathbf{A}$	Diagonal matrix with vell volumes	`cellvolmat`
$\boldsymbol{\chi}$	Additional fluid state variable introduced by the turbulence model	`turbparamve`

Fluid Structure Interaction

$\mathcal{P}$	State equation of the structure	`strucstateq`
$\mathcal{D}$	State equation of the mesh motion	`mmsstateeq`
$\mathcal{F}$	State equation of the fluid	`fluidstateq`

Sturctural Analysis

$\mathbf{K}$	FE stiffness matrix	`stiffmat`
u	Displacement vector	`disp`
$\mathbf{d}$	Interface displacement	`ifacedispvec`
$\mathbf{u}$	Discrete displacement vector	`dispvec`
$\boldsymbol{x}$	Mesh motion	`motion`

Optimization

s	Abstract optimazation variable	`absvar`
q	Optimization criterium	`optcrit`
ϵ^{SA}	Specified tolerance in the Sensitivity analysis	`tolsa`
$\boldsymbol{\eta}$	Lagrange multipliers of the equality constraints	`lagmultseq`
L	Lagrangian function of the optimization problem	`Lagfunc`
$\boldsymbol{s}$	Vector of abstract optimization variables	`absvars`
$\boldsymbol{g}$	Non-equality constraints	`neqctr`
$\boldsymbol{\gamma}$	Lagrange multipliers of the inequality constraints	`lagmultsneq`
$\boldsymbol{q}$	Vector of optimization criteria	`optcrits`
$\boldsymbol{d}$	Physical design parameters	`physvars`
z	Target cost function	`costfunc`
$n_{\boldsymbol{g}}$	Number of non-equality constraints	`numneqctr`
$\boldsymbol{h}$	Equality constraints	`eqctr`
$\boldsymbol{a}$	Adjoint solutions	`adjoints`
$n_{\boldsymbol{h}}$	Number of equality constraints	`numeqctr`

Fluid Mechanics

p	Pressure	`pres`
$\mathcal{F}$	Convective fluxes	`fluxesconv`
$\boldsymbol{P}$	Matrix that contains the eigenvectors of the jacobian matrix of $\mathbf{R}^c$	`jaceigvecs`
T	Fluid temperature	`temp`
v_3	Fluid velocity in z-direction	`fluidvelz`
$\boldsymbol{\Lambda}$	Diagonal matrix that contains the eigenvalues of the jacobian matrix of $\mathbf{R}^c$	`jaceigvals`
$\mathcal{H}$	Jacoian matrix	`jac`
ρ	Density	`dens`
μ	Dynamic viscosity	`viscosdyn`
$\mathbf{I}$	Identity matrix	`eye`
ν	Kinematic viscosity	`viscoskin`
$\mathcal{A}$	Flux Jacobian	`fluxjac`
τ	Deviatoric fluid stress tensor	`fluidstressc`
$\mathcal{G}$	Diffusive fluxes	`fluxesdiff`
q	Heat flux comopnenent	`heatfluxcomp`
k	Thermal conductivity of the fluid	`thermcond`
E	Total energies	`energytot`
R_e	Reynolds number	`reynolds`
$\boldsymbol{q}$	Heat flux vector	`heatflux`
γ	Specific heat ratio	`specheatrati`
v_2	Fluid velocity in y-direction	`fluidvely`
e	Internal energy	`energyint`
v	Fluid velocity vector	`fluidvelcomp`
v_1	Fluid velocity in x-direction	`fluidvelx`
$\boldsymbol{M}$	Averaging function associated with the Roe flux	`roeavgfunc`

Abbrevations

SQP Sequential Quadratic Programming

GSE Global Sensitivity Equations

SA Sensitivity Analysis

VOF Volume of Fluid Method

GFM Ghost Fluid Method

GFMP Ghost Fluid Method of the Poor

EOS Equation of State

JWL Jones-Wilkins-Lee

PG Perfect Gas

SG Stiffened Gas

MUSCL Monotonic Upwind scheme for Conservation Laws

FIVER Finite Volume Method with exact two-phase Riemann Integrals

RANS Reynolds Averaged Navier-Stokes

FV Finite Volumes

FD Finite Differences

FE Finite Elements

SA Sensitivity Analysis

LNM Lehrstuhl für Numerische Mechanik(Institute for Computational Mechanics)

TUM Technical University of Munich

CFD Computational Fluid Dynamics

ALE Arbitrary Lagrangian Eulerian

VOF Volume of Fluid Methods

DFP Davidon-Fletcher Powell formula

BFGS Broyden-Fletcher-Goldfarb-Shanno algorithm

LDR Lift over Drag Ratio

NSE Navier Stokes Equations

NSME Momentum Equation of the Navier Stokes Equations

NSCE Continuity Equation of the Navier Stokes Equations

NSEE Energy Equation of the Navier Stokes Equations

PDE Partial Differential Equation

CG Conjugate Gradient

PCG Preconditioned Conjugate Gradient

SA Sensitivity Analysis

CFD Computational Fluid Dynamics

FSI Fluid Structure Interaction

FRG Farhat Research Group

1 Fluid mechanics

:fluid_mechanics

1.1 Eulerian and Lagrangian view

erian_lagrangian

In time dependent analytical physics there are two basic concepts of how to view a system of interest: the Eulerian and the Lagrangian approach. It shall be stressed here, that Eulerian view and Euler equations are two unrelated concepts..
The issue of Lagrangian versus Eulerian perspective can be explained in many ways, in the end it boils down to the interpretation of the time derivative. Since we will be using the different derivatives frequently in this thesis, a short introduction shall be provided.

1.1.1 Material derivative

The material derivative of a property, is a derivative(rate of change), that follows a specific particle 'p'. That means that at every time instance the material derivative gives as the current value of the property of a specific particle. Since in general, a particle will move its position in time quite a lot, some mean is required to track the particles motion. The material derivative is also known as the Lagrangian concept, and is commonly used in solid mechanics, since the motion of the solid particles is typically little, or at least nearly uniform over the body, which makes position tracking easy.

1.1.2 Eulerian derivative

The Eulerian derivative takes a different approach. It refers to a fixed frame of reference in the domain, and regards the rate of change of a quantity of interest at that specific position. Due to the particle movement, it typically refers to different particles at different times.
Material and Eulerian derivatives can be linked via the so-called convective rate of change as:

$$\underbrace{\frac{D\boldsymbol{G}}{Dt}}_{\text{Lagrangian rate of change}} = \underbrace{\frac{\partial \boldsymbol{G}}{\partial t}}_{\text{Eulerian rate of change}} + \underbrace{\boldsymbol{v} \cdot \nabla \boldsymbol{G}}_{\text{Convective rate of change}} \tag{1}$$

In fluid mechanics the Eulerian concept is typically preferred.
However, when looking at fluid-structure interaction problems, the interface is typically moving. To account for that motion in the fluid, one either has to come up with some cell-intersection approach, or the fluid-mesh has to be deformed too.
acro:ALE An appealing and straightforward approach to do so, is the so called Arbitrary Lagrangian Eulerian (ALE) approach. As the name suggest, it is a hybrid between Eulerian and Lagrangian point of view. The ALE approach allows to utilize the best of both approaches. The mesh can either be fixed, moved with the continuum in an Eulerian manner, or be moved in some arbitrarily specified manner in between. The ALE method can handle larger distortions than a pure Eulerian would, while still allowing to keep interface continuity between the structure and the fluid, such

that no intersection of fluid cells is necessary.
In this thesis, both Eulerian and ALE methods will be covered. In general, no recommendation on to prefer one or the other method can be given. Instead, one should always be aware of the pros and cons of either method and choose the appropriate one for a specific problem accordingly.
For a more in depth-analysis, the interested reader is referred to [8]

1.2 Basic equations of fluid mechanics

ations_basic
acro:NSE
The physics of fluid motion are governed by the so-called Navier Stokes Equations (NSE). These equations were derived independently of one another by Claude-Louis Navier and George Gabriel Stokes as a generalization of the Euler-equations by including viscosity effects. In general one distinguishes the incompressible and the compressible NSE. The validity of their application depends on the problem setup. As a rule of thumb, flows with Mach-numbers lower than 0.3 can usually be safely approximated by the incompressible NSE.
acro:NSME
acro:NSCE
acro:NSEE
The Navier stokes equation consist of the Momentum Equation of the Navier Stokes Equations (NSME), the Continuity Equation of the Navier Stokes Equations (NSCE) and, if the compressible equations are solved, the Energy Equation of the Navier Stokes Equations (NSEE). In the literature, the term NSE is often synonymously used for the NSCE, in this thesis however, with NSE we will always refer to the full set of equations and will explicitly name the momentum, continuity or energy equation otherwise.

1.2.1 Incompressible Navier Stokes Equations

The incompressible NSE are derived by putting a few assumptions on the Cauchy stress tensor:

- Galilean invariance of the fluid stress tensor
- Isotropy of the fluid
- The stokes stress constitutive equation applies: $\boldsymbol{\tau} = 2\mu\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} = \frac{1}{2}(\nabla\boldsymbol{v} + (\nabla\boldsymbol{v})^T)$

In convective form, the incompressible NSME can be written as

$$\overbrace{\underbrace{\frac{\partial \boldsymbol{v}}{\partial t}}_{\text{Variation}} + \underbrace{(\boldsymbol{v}\cdot\nabla)\boldsymbol{v}}_{\text{Convection}}}^{\text{Inertia per volume}} - \overbrace{\underbrace{\nu\nabla^2\boldsymbol{v}}_{\text{Diffusion}} = -\underbrace{\nabla \boldsymbol{w}}_{\text{Internal source}}}^{\text{Divergence of stress}} + \underbrace{g}_{\text{External source}} \tag{2}$$

eq:nsg_incompressible
eq:nsg_incompress

where $\boldsymbol{w}$ is the specific thermodynamic work, which satisfies

$$\nabla \boldsymbol{w} = \frac{1}{\rho}\nabla p \tag{3}$$

And the incompressible NSCE can be written as

$$\nabla \cdot \boldsymbol{v} = 0 \tag{4}$$

It shall be noted, that in the incompressible NSE, the pressure can only be solved up to a certain constant. The pressure at a certain point therefore has to be fixed, or the system will be singular.

1.2.2 Compressible Navier Stokes Equations

nsg_compressible

Compared to the incompressible NSE, the compressible NSE introduce one additional unknown, since the density can now vary. To close the system, one additional equation is required: the energy equation.
The compressible NSE are derived by making the following assumptions:

- Galilean invariance of the stress tensor
- Linearity of the stress in the velocity gradient: $\boldsymbol{\tau}(\nabla \boldsymbol{v}) = \boldsymbol{C} : (\nabla \boldsymbol{v})$, where $\boldsymbol{C}$ is called the viscosity or elasticity tensor.
- Isotopy of the fluid

The compressible NSME can be written as

$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{v} = -\frac{\partial 1}{\partial \rho} \nabla p + \nu \nabla^2 \boldsymbol{v} + \frac{1}{3} \nu \nabla (\nabla \cdot \boldsymbol{v}) + g \tag{5}$$

eq:nsg_compressible

eq:nsg_compre

where it shall be noted that for the special case of an incompressible flow, the volume of the fluid elements is constant, resulting in a solenoidal velocity field. Thus $\nabla \cdot \boldsymbol{v} = \boldsymbol{0}$ and $\nabla p = 0$, which gives equation 2.
The NSCE now takes into accounts for the variable density

$$\frac{\partial \rho}{\partial t} = \nabla \cdot (\rho \boldsymbol{v}) = 0 \tag{6}$$

And finally the NSEE writes

$$\frac{\partial s}{\partial t} = -\boldsymbol{v} \cdot s + \frac{Q}{T} \tag{7}$$

where s is the entropy per unit mass, Q is the heat transferred, and T is the temperature.

1.2.3 Euler equations

euler_convective

The Euler equations can be regarded as a special case of the Navies stokes equations where the diffusive(viscous) contributions are neglected. Dimension analysis [2] reveals that this is usually justified for high Mach numbers and distant from walls. In fact, boundary Layer theory suggests that a lot of problems can be well approximated by solving the full, diffusive equations only near the wall boundary and using the

Euler equations everywhere else. Although the Euler equations can be derived for both incompressible and compressible flows, the statement above suggests that only the compressible ones give actual, physically correct results. We therefore restrict our considerations to the compressible Euler equations.
Summing up, these equations can be written in convective form as

$$\begin{cases} \frac{\partial \rho}{\partial t} + \boldsymbol{v} \cdot \nabla \rho + \rho \nabla \cdot \boldsymbol{v} = 0 & \text{Continuity Equation} \\ \frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{v} + \frac{p}{\rho} = \boldsymbol{g} & \text{Momentum Equation} \\ \frac{\partial e}{\partial t} + \boldsymbol{v} \cdot \nabla e + \frac{p}{\rho} \nabla \cdot \boldsymbol{v} = 0 & \text{Energy Equation} \end{cases} \tag{8}$$

1.2.4 Conservative Form

ive_form_nsg

For the purpose of FV discretization one usually brings the equations into the so-called conservative form, which can be written generally as

$$\frac{\mathrm{d}\boldsymbol{\xi}}{\mathrm{d}t} + \nabla \cdot f(\boldsymbol{\xi}) = \mathbf{0} \tag{9}$$ eq:conservative_form_general eq:conservative_f

where $\boldsymbol{\xi}$ is the conserved quantity of interest and $f(\boldsymbol{\xi})$ is denoted as "flux function". This equation can then be transformed into an integral form using the divergence theorem

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_V \boldsymbol{\xi} dV + \int_{\partial V} f(\boldsymbol{\xi}) \cdot \boldsymbol{n} \, dS \tag{10}$$ eq:conservation_form_integral eq:conservation_f

This equations states that the rate of change of the integral of the quantity $\boldsymbol{\xi}$ over an arbitrary control volume V is equal to the negative of the "flux" through the boundary of the control volume ∂V.
A simple choice for f would be $f(\xi) = \xi \boldsymbol{v}$, which means that the quantity $\boldsymbol{\xi}$ follows the fluid field and gives the so-called "advection equation".
Although one form is derived from the other, these two are not equivalent. Particularly, it is possible to find solutions to the integral equations that are non-differentiable and therefore not a solutions to the differential form. This leads to so called "weak solutions" and is the cause for many numerical difficulties in the finite volume simulation of such problems.

This thesis focuses on the compressible NSE only. They can be brought to conservative form as:

$$\frac{\partial \boldsymbol{w}}{\partial t} + \nabla \cdot \mathcal{F}(\boldsymbol{w}) + \nabla \cdot \mathcal{G}(\boldsymbol{w}) = \mathbf{0} \tag{11}$$ eq:nsg_conservative_form eq:nsg_conservati

where, comparing to the general conservative form, one can see that $\boldsymbol{\xi} = \boldsymbol{w}$ and $f(\boldsymbol{\xi}) = \mathcal{F}(\boldsymbol{w}) + \mathcal{G}(\boldsymbol{w})$

In the conservative form of the NSE the so-called "fluid state vector" $\boldsymbol{w}$ is defined as

$$\boldsymbol{w} = \begin{bmatrix} \rho \\ \rho\boldsymbol{v} \\ E \end{bmatrix} \tag{12}$$ eq:fstate_definition

with

$$\boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{13}$$ eq:fluidvel_definition

denoting the fluid velocity vector.

The total energy E can be computed as a sum of the so called internal energy and the kinematic energy due to fluid velocity as

$$\underbrace{E}_{\text{total energy}} = \underbrace{\rho e}_{\text{internal energy}} + \underbrace{\frac{1}{2}\boldsymbol{v}^T\boldsymbol{v}}_{\text{kinematic energy}} \tag{14}$$ eq:energytot

Bringing the convective form of the compressible NSE (5) into the conservative form (11), one can derive the convective fluxes $\mathcal{F}$ as

$$\mathcal{F} = \boldsymbol{w}\boldsymbol{v}^T + p \begin{bmatrix} 0 \\ \mathbf{I} \\ \boldsymbol{v}^T \end{bmatrix} \tag{15}$$ eq:fluxesconv

$$= \begin{bmatrix} \rho\boldsymbol{v}^T \\ \rho(\boldsymbol{v}\boldsymbol{v}^T) + p\mathbf{I} \\ (E+p)\boldsymbol{v}^T \end{bmatrix} \tag{16}$$

$$= \begin{bmatrix} \rho v_1 & \rho v_2 & \rho v_3 \\ p + \rho {v_1}^2 & \rho v_1 v_2 & \rho v_1 v_3 \\ \rho v_2 v_1 & p + \rho {v_2}^2 & \rho v_2 v_3 \\ \rho v_3 v_1 & \rho v_3 v_2 & p + \rho {v_3}^2 \\ v_1(E+p) & v_1(E+p) & v_1(E+p) \end{bmatrix} \tag{17}$$

depending on the algorithm, one or another form can be advantageous.

The diffusive fluxes can be written as

$$\mathcal{G} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau} \\ \boldsymbol{\tau v} + \boldsymbol{q} \end{bmatrix} \tag{18}$$ eq:fluxes_diff

$$= \begin{bmatrix} 0 & 0 & 0 \\ -\tau_{xx} & -\tau_{yx} & -\tau_{zx} \\ -\tau_{xy} & -\tau_{yy} & -\tau_{zy} \\ -\tau_{xz} & -\tau_{yz} & -\tau_{zz} \\ -q_x - v_1\tau_{xx} - v_1\tau_{yx} - v_1\tau_{zx} & -q_y - v_1\tau_{xy} - v_1\tau_{yy} - v_1\tau_{zy} & -q_z - v_1\tau_{xz} - v_1\tau_{yz} - v_1\tau_{zz} \end{bmatrix}^T \tag{19}$$

The definition of the stress tensor depends of the fluid model used. For the simple case of a Newtonian fluid, it can be written as

$$\boldsymbol{\tau} = \mu(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T) + \lambda(\nabla \cdot \boldsymbol{v})\mathbf{I} \tag{20}$$ eq:fluidstress

For the relation between Temperature and heat flux an assumption has to be made. Most often, a simple Fourier's law is assumed

$$\boldsymbol{q} = -k\nabla T \tag{21}$$ eq:heatflux

1.2.5 Euler equations

The euler equations are a simplified form of the Navier-Stokes Equations (2)(5), where the viscous effects are neglected by setting $\mathcal{G} = \mathbf{0}$.

$$\frac{\partial \boldsymbol{w}}{\partial t} + \nabla \cdot \mathcal{F}(\boldsymbol{w}) = \mathbf{0} \tag{22}$$ eq:euler

The Euler equations are appropriate for a wide range of applications. A typical indicator is the Reynolds number, which describes the ratio between inertial forces and viscous forces:

$$R_e = \frac{\rho \boldsymbol{v} L}{\mu} = \frac{\boldsymbol{v} L}{\nu} \tag{23}$$ eq:reynolds

where L describes a characteristic length.
A high Reynolds number thus indicates that the flow is dominated by inertial forces, thus the Euler Equations should give satisfying results. However, an Euler flow lacks the ability to represent stick wall boundary conditions, thus it is unable to represent boundary layers.

1.2.6 Equations of State

Looking at the above Equations, one might notice that the number of unknowns is greater than the number of equations. Particularly, the pressure only appears in

Equation (15) and is not linked to the other formulas. This problem is solved by introducing an Equation of State (EOS) that relates pressure, internal energy and density. The EOS depends on the fluid model, some well-known ones are: Perfect Gas (PG), Stiffened Gas (SG), Jones-Wilkins-Lee (JWL).
For the simplest one, PG, the EOS can be written as

$$p = (\gamma - 1)\rho e \tag{24}$$ eq:eos_pg

1.2.7 Reynolds Averaged Navier-Stokes Equations

sec:rans

The Reynolds Averaged Navier-Stokes (RANS) Equations are time-averaged equations of motion for the fluid.

$$\boldsymbol{w} \to \bar{\boldsymbol{w}} = \lim_{T\to\infty} \frac{1}{T} \int_{t^0}^{t^0+T} \boldsymbol{w} dt \tag{25}$$ eq:rans_averaging

The main idea of the approach is to decompose an instantaneous quantity into time-averaged and fluctuating components

$$\boldsymbol{w} = \underbrace{\bar{\boldsymbol{w}}}_{time-average} + \underbrace{\boldsymbol{w}'}_{fluctuation} \tag{26}$$ eq:rans_decomposition

When substituting this decomposition back into the NSE(and injecting several other approximations), a closure problem induced by the arising non-linear Reynolds stress term $R_{eij} = -\bar{v_i'v_j'}$ arises. Additional modeling is therefore required to close the RANS equations, which has led to many different turbulence models.

Whatever turbulence model is chosen, the fluid-state vector is augmented by the m parameters of the turbulence model

$$\boldsymbol{w}_{RANS} \leftarrow \begin{bmatrix} \boldsymbol{w} \\ \chi_1 \\ \vdots \\ \chi_m \end{bmatrix} = \begin{bmatrix} \rho \\ \rho \boldsymbol{v}^T \\ E \\ \chi_1 \\ \vdots \\ \chi_m \end{bmatrix} \tag{27}$$

The RANS equations can then be written as

$$\frac{\partial \bar{\boldsymbol{w}}}{\partial t} + \nabla \cdot \mathcal{F}(\bar{\boldsymbol{w}}) + \nabla \cdot \mathcal{G}(\bar{\boldsymbol{w}}) = \mathsf{S}(\bar{\boldsymbol{w}}, \chi_1, \cdots, \chi_m) \tag{28}$$ eq:rans_eqautions

Spalart Allamars turbulence model As explained in the previous section, the RANS equations necessitate additional modeling to close the system of equations. This is typically performed by applying the Boussinesq assumption and assuming

that the fluid stress tensor can be written as

$$\boldsymbol{\tau}_{ij} = 2\mu_t \left(\bar{S}_{ij} - \frac{1}{3}\frac{\partial \bar{v}_k}{\partial x_k}\delta_{ij} \right) - \frac{2}{3}\bar{\rho}k\delta_{ij} \tag{29}$$ eq:rans_turbulent_stress

eq:rans_turbulent

$$\bar{S}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{v}_i}{\partial x_j} + \frac{\partial \bar{v}_j}{\partial x_i} \right) \tag{30}$$

$$k = \frac{1}{2}\left(v_i' v_i' \right) \tag{31}$$

where μ_t denotes the turbulent viscosity and k is the mean turbulent kinetic energy. Both μ_t and k need to be modeled, thus an additional two equations are required to close the system. A variety of different turbulence models have been proposed in the literature, this thesis however, focuses on the very particular case of the Spalart-Allmaras model, introduce in [18].
Herein, the last term of equation (29) is being neglected, thus a single additional equation is sufficient to close the system. Specifically, Spalart-Allmaras introduces the single, scalar, viscosity-like turbulent variable

$$\nu^\star = \frac{\mu_t}{\bar{\rho} f_{v1}} \tag{32}$$

and the corresponding transport equation

$$\frac{\partial}{\partial t}\left(\bar{\rho}\nu^\star\right) + \frac{\partial}{\partial x_j}\left(\bar{\rho}\bar{v}_j\nu^\star\right) = \frac{1}{\sigma}\left[\frac{\partial}{\partial x_j}\left(\bar{\rho}(\nu + \nu^\star)\frac{\partial \nu^\star}{\partial x_j} \right) + c_{b2}\bar{\rho}\left(\frac{\partial \nu^\star}{\partial x_j}\frac{\partial \nu^\star}{\partial x_j}\right) \right] + c_{b1}(1 - f_{t2})S^\star \bar{\rho}\nu^\star - \bar{\rho}\left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right)\left(\frac{\nu^\star}{d}\right)^2 \tag{33}$$

with

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \quad \chi = \frac{\nu^\star}{\nu}, \quad S^\star = \omega + \frac{\nu^\star}{k^2 d^2} f_{v2} \tag{34}$$

where ω is the vorticity magnitude and d is the nearest distance to the wall.

te_volume_method

2 Numerical solution

2.1 General

acro:FD
acro:PDE
Like Finite Differences (FD) or Finite Elements (FE), the Finite Volumes (FV) method is a mean of solving a Partial Differential Equation (PDE) by transforming it into a discrete algebraic form. In the field of fluid mechanics, the finite volume method is the most popular approach. Other than FD or FE, the FV is conservative by construction. In contrast to FD it can easily be implemented for unstructured grids. Compared to the FE method, where boundary conditions come naturally from the formulation, this causes some difficulties in FV.
It shall also be mentioned that the introduction of stabilization schemes(like streamline upwinding), is much easier in an FV formulation. Overall, for Computational Fluid Dynamics (CFD), FV has so far shown to be the best compromise between accuracy, stability and efficiency.
acro:CFD

Finite volume methods are typically derived from the so called conservative form of a PDE. It is shown in Section 1.2.4, that the NSE equation can be brought to the conservative form. In general, the conservative form can be written as:

$$\frac{\mathrm{d}\boldsymbol{\xi}}{\mathrm{d}t} + \nabla \cdot f(\boldsymbol{\xi}) = \mathbf{0} \tag{35}$$ eq:general_conservative_form
eq:general_co

(36)

where $\boldsymbol{\xi}$ represents a vector of states and f is the so-called flux tensor.
After subdividing the domain into finite volumes, also called cells, one can write for each particular cell i

$$\int_{V_i} \frac{\mathrm{d}\boldsymbol{\xi}}{\mathrm{d}t} dV_i + \int_{V_i} \nabla \cdot f(\boldsymbol{\xi}) dV_i \tag{37}$$

It shall be stressed again that reformulating the equations in integral form has already changed the space of admissible solutions. Particlularly, the integral form is capable of capturing discontinuities, like shocks whereas the derivative term would be undefined at a shock in equation (35).
After applying the divergence theorem to the second term this gives:

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{V_i} \boldsymbol{\xi} dV_i + \int_{\partial V_i} f(\boldsymbol{\xi}) \cdot \boldsymbol{n} \, dS_i \tag{38}$$

And after integration the first term to get the volume average

$$V_i \frac{\mathrm{d}\bar{\boldsymbol{\xi}}}{\mathrm{d}t} + \int_{\partial V_i} f(\boldsymbol{\xi}) \cdot \boldsymbol{n} \, dS_i \tag{39}$$

So that finally, the equation can be written as

$$\frac{\mathrm{d}\bar{\boldsymbol{\xi}}}{\mathrm{d}t} + \frac{1}{V_i} \int_{\partial V_i} f(\boldsymbol{\xi}) \cdot \boldsymbol{n} \, dS_i \tag{40}$$ eq:final_fv_general
eq:final_fv_g

which can easily be interpreted: The cell average of the conserved properties changes according to the total flux through the cells surface. Of course, the conservative value is defined as being constant within one cell, so there will be different values on faces or edges, depending on which side one is looking at. There are different approaches on how to choose an appropriate value, and the choice may greatly effect the numerical properties. In a FV solver, the numerical flux at the interface is typically constructed such, that upwinding is achieved.

2.2 Finite Volume method for fluid mechanics

id_mechanics

It has already been shown in Section 1.2.4, that the NSE can be brought into the conservative form (11). The flux has been shown to consist of a convective and a diffusive part(15)(18). For this thesis, a Monotonic Upwind scheme for Conservation Laws (MUSCL) type FV framework for unstructured three-dimensional grids, as described in [9] has been used.

acro:MUSCL

In equivalent manner as in equations (35) to (40) one can now write the conservative form of the NSE as

$$\frac{\partial \bar{\boldsymbol{w}}_i}{\partial t} + \frac{1}{\|\Omega_i\|}\int_{\partial\Omega_i} \mathcal{F}(\boldsymbol{w})\cdot dS + \frac{1}{\|\Omega_i\|}\int_{\partial\Omega_i} \mathcal{G}(\bar{\boldsymbol{w}}, \nabla\bar{\boldsymbol{w}})\cdot dS = \mathbf{0} \tag{41}$$

eq:NSE_full_FV formulation

eq:NSE_full_FV fo

2.3 Mixed FV-FE formulation

_formulation

It would be totally feasible to now solve both terms with classical FV methods. However, we note two things. First, there is no need for upwinding in the diffusive term due to its elliptic nature. Secondly, the integration of the second term also becomes much more cumbersome than the first one due to the gradient of the fluid state vector demanding at least a first order representation.

For this reasons, we introduce the following, mixed FV-FE formulation

$$\frac{\partial \boldsymbol{w}_i}{\partial t} + \int_{\partial\mathcal{C}_i} \mathcal{F}(\boldsymbol{w})\cdot dS - \int_{\sum_{T_i}} \mathbb{K}\boldsymbol{w}\nabla\phi_i dx = \mathbf{0} \tag{42}$$

eq_mixed_FV_FE

eq_mixed_FV_FE

where $\mathbb{K}$ is the diffusive tensor introduced in **TODO** and ϕ_i is a linear P1 shape-function over the triangle. Please also note that the convective term is now integrated over the dual cell related to vertex i, whereas the diffusive term integrated over all primal triangles connected to vertex i.

We will justify this re-formulation in section 2.4. See also figure 1.

As can be seen from Figure 1, the dual cells themselves can have very random shapes. This makes the integration over the boundary of the second term in Equation (42) more cumbersome than in a primal approach. Also, the volume integral in the FE-like expression has to be splitted into regular shaped subdomains, e.g. tetrahedra, such that standard integration rules, like gauss-rule, can be applied.

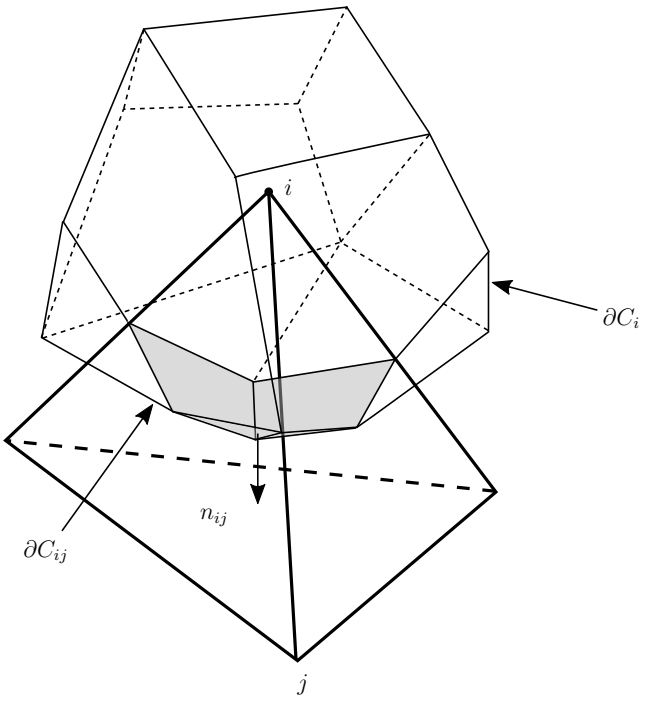


Figure 1: FV semi-discretization of an unstructured mesh. Vertex i is the center of dual cell C_i. The primal element is drawn in thick black lines, the dual cell in dashed lines. The boundary of the dual cell is denoted as ∂C_{ij}

ell_unstructured

For a closer look into the convective fluxes integral, we decompose the boundary as $\partial\Omega_i = \sum_{i\in\kappa(i)} \partial\Omega_{ij}$, where $\kappa(i)$ denotes the set of vertices connected by an edge to vertex i.
In this thesis, the surface integral in equation (42) is then approximated using a Riemann solver [14] and a MUSCL [19] technique. This approximation can be written as

$$\int_{\partial\Omega_i} \mathcal{F}(\boldsymbol{w}) \cdot dS \approx \sum_{j\in\kappa(i)} \boldsymbol{\phi}_{ij}(\boldsymbol{w}_{ij}, \boldsymbol{w}_{ji}, \boldsymbol{n}_{ij}) \tag{43}$$

eq:convectiveterm_approximation

eq:convective

where $\boldsymbol{\phi}_{ij}$ denotes the chosen numerical flux function along the edge $i-j$ and the two extrapolated fluid states at the i and the j-side of the the intersection of the cell boundary $\partial\Omega_{ij}$ and edge $i-j$ are denoted by $\boldsymbol{w}_{ij}$ and $\boldsymbol{w}_{ji}$ respectively. The area-weighted normal of edge $i-j$ is denoted as $\boldsymbol{n}_{ij}$.
A popular choice for the numerical approximation of the convective fluxes is the so-called Roe flux. Appropriate upwinding is directly built into the definition of the Roe flux [14]. In 1D it is defined as:

$$\boldsymbol{\phi}_{ij} = \frac{1}{2}\left(\mathcal{F}(\mathbf{w}_j) + \mathcal{F}(\mathbf{w}_i)\right) - \frac{1}{2}|\mathbf{A}|\left(\mathbf{w}_j - \mathbf{w}_i\right) \tag{44}$$

where $\mathbf{A}$ is the flux Jacobian.
To evaluate a roe-flux between two points in 2D or 3D, a normal vector is required to project the higher, dimensional flux on the connecting line.

As for the volume integral of the diffusive term in Equation (42), it shall be noted that the shape function is still defined in the primal cell. Since the gradient of the test function is constant, as is the diffusive flux itself, the integral becomes a summation of the primal sub-tetrahedral contributions.
The final numerical approximation can therefore be summarized as

$$\frac{\partial \bar{\mathbf{w}}_i}{\partial t} + \sum_{j \in \kappa(i)} \boldsymbol{\phi}_{ij}(\mathbf{w}_{ij}, \mathbf{w}_{ji}, \boldsymbol{\nu}_{ij}) - \sum_{T_i \in \lambda(i)} \int_{T_j} \mathbb{K} \nabla \tilde{\mathbf{w}} \nabla \phi_j dx = \mathbf{0} \tag{45}$$

eq:nse_final_discretized

eq:nse_final_disc

where $\kappa(i)$ is the set of vertices connected to vertex i by an edge, and $\lambda(i)$ is the set of primal elements connected to vertex i.

s_term_treatment

2.4 Derivation of the mixed FV-FE formulation

The equivalence of the finite volume representation of the viscous term, with the finite element-like representation provided in Equation (42) might not be obvious to the reader. This section is therefore denoted to a FE discretization of the governing equations, that can then be reformulated such that the mixed FV-FE formulation is obtained.
First, we start with the full NSE in conservative form, as provided in Equation (11). Next, a test function is defined. We chose the function $(\chi_i)_{i\in\mathcal{P}_h}$ where $\mathcal{P}_h$ represents the primal mesh, as step functions over the dual cells of $\mathcal{D}_h$. A multiplication with the test-functions and subsequent integration thus gives the following weak form

$$\int_\Omega \frac{\partial \boldsymbol{w}}{\partial t}\chi_i dx + \int_\Omega \nabla\cdot\mathcal{F}(\boldsymbol{w})\chi_i dx + \int_\Omega \nabla\cdot\mathcal{G}(\boldsymbol{w},\nabla\boldsymbol{w})\chi_i dx = \mathbf{0} \tag{46}$$

fv_weak_form_felike

fv_weak_form_

where by using the divergence theorem, we obtain

$$\begin{aligned}\int_\Omega \nabla\cdot\mathcal{F}(\boldsymbol{w})\chi_i dx &= \int_{\partial C_i}\mathcal{F}(\boldsymbol{w})\cdot\boldsymbol{n}ds\\ \int_\Omega \nabla\cdot\mathcal{G}(\boldsymbol{w},\nabla\boldsymbol{w})\chi_i dx &= \underbrace{\int_{\partial C_i}\mathbb{K}\nabla\boldsymbol{w}\cdot\boldsymbol{n}ds}_{II}\end{aligned} \tag{47}$$

eq:divtheorem_felike

eq:divtheorem

Where the first term is already equivalent to what we have derived by the FV approach in (41).

It the diffusive tensor $\mathbb{K}$ is approximated to be constant over the triangle, which is justified in section 2.4.1, the product $\mathbb{K}\boldsymbol{w}$ is constant, one can now write the above integral as

$$II = \sum_{T\in\mathcal{D}_i}\mathbb{K}\nabla\boldsymbol{w}\int_{\partial C_i\cap T}\boldsymbol{n}_C ds \tag{48}$$

Now, we use a particular geometric relation, that holds for primitive elements like triangles and tetrahedra, and is derived in Section 2.4.1

$$\int_{C_i\cap T}\boldsymbol{n}ds = \frac{\boldsymbol{n}_i^T}{2} \tag{49}$$

where $\boldsymbol{n}_i^T$ is the normal to the primal mesh triangle T on the edge opposite of node i. This is where the geometric considerations of section 2.4.1 come into place. As shown in equation (58), the integral over the dual face edge can be written via the normal of the primal mesh triangle as

$$-\sum_{T\in\mathcal{D}_i}\mathbb{K}\nabla\boldsymbol{w}\frac{\boldsymbol{n}_i^T}{2} \tag{50}$$

Also, we have shown in Section 2.4.1, that for the case of a simple triangle (but equally applicable to tetrahedra), the last term of the above equation can be written as

$$-\sum_{T\in\mathcal{D}_i}\mathbb{K}\nabla\boldsymbol{w}\int_T\nabla\phi_i dx \tag{51}$$

where ϕ_i is the P1 shape-function defined in (53). Finally we can go the opposite way, and join the summations over integrals over triangular contributions back to an overall integral over the whole dual cell. Doing this finally leads to

$$II=-\int_{\mathcal{D}_i}\mathbb{K}\nabla\boldsymbol{w}\cdot\nabla\phi_i dx \tag{52}$$

which is exactly the term provided in equation (45). Since we used the same step-function based test-functions for all terms in equation (46), this approach is not only consistent, for the special case, of linear triangles we have shown that our finite element term is equivalent to a FV approximation.

2.4.1 Geometric considerations

nsiderations

This section is dedicated to the derivation of a particular link between the shape-function of a linear triangle/tetrahedra and its outward facing normals that we use in chapter 2.4 to proof the equivalence of the FE-formulation of the viscous term with the FV formulation.

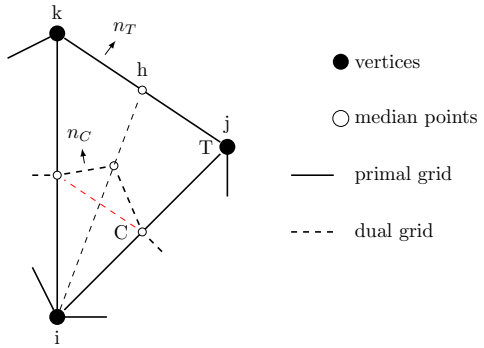


Figure 2: One Triangle T out of the set of triangles $\sum_T$, that are connected to vertex i. C denotes the dual cell related to vertex i.

ape_function

We first look at one triangle of the set of triangles $\sum_T$ that belong to node i.
The setup is visualized in Figure 2.
The reader can easily verify, that a linear shape-function of node i can be written as:

$$\phi_i(\boldsymbol{x})=\frac{\boldsymbol{hx}\cdot\boldsymbol{hi}}{\|\boldsymbol{hi}\|^2} \tag{53}$$

eq:p1_triangle_eq1

eq:p1_triangle_eq

where $\boldsymbol{hx}$ is the vector from point h to any point x in the triangle, and $\boldsymbol{hi}$ is the vector from point h to point i. For this particular representation of the shape function, the gradient can be elegantly written as

$$\nabla\phi_i(\boldsymbol{x}) = \frac{\boldsymbol{hi}}{\|\boldsymbol{hi}\|^2} = \frac{\hat{\boldsymbol{hi}}}{\|\boldsymbol{hi}\|} \tag{54}$$

eq:p1_triangle_eq2 eq:p1_triangl

where $\hat{\boldsymbol{hi}}$ is the normalized vector in $h-i$ direction. Moreover, simple geometry tells us that

$$\|\boldsymbol{hi}\|\,\|\boldsymbol{kj}\| = 2|T| \tag{55}$$

eq:p1_triangle_eq3 eq:p1_triangl

where $|T|$ is the area of the triangle T
Substituting equation 55 into equation 54 therefore leads to

$$\nabla\phi_i(\boldsymbol{x}) = \boldsymbol{hi}\cdot\frac{\|\boldsymbol{kj}\|}{2|T|} = -\frac{\boldsymbol{\nu}}{2|T|} \tag{56}$$

eq:p1_triangle_eq4 eq:p1_triangl

where $\boldsymbol{\nu}$ is the weighted outward facing normal to edge jk.

$$\boldsymbol{\nu} = -\hat{\boldsymbol{hi}}\,\|kj\| = -\int_{[jk]} \boldsymbol{n}_T d\Gamma \tag{57}$$

Referring to previous chapter we remind the reader that in the vertex based FV approach, an integration over the dual face interface (dashed line in Figure 2) via the weighted interface normal is typically performed.
By simple geometrical considerations we can now determine, that the following relation holds

$$\int_{\Gamma_C\cap T} const.\ \boldsymbol{n}_c = \frac{1}{2}\int_{\Gamma_T} const.\ \boldsymbol{n}_T \tag{58}$$

eq:final_relation eq:final_rela

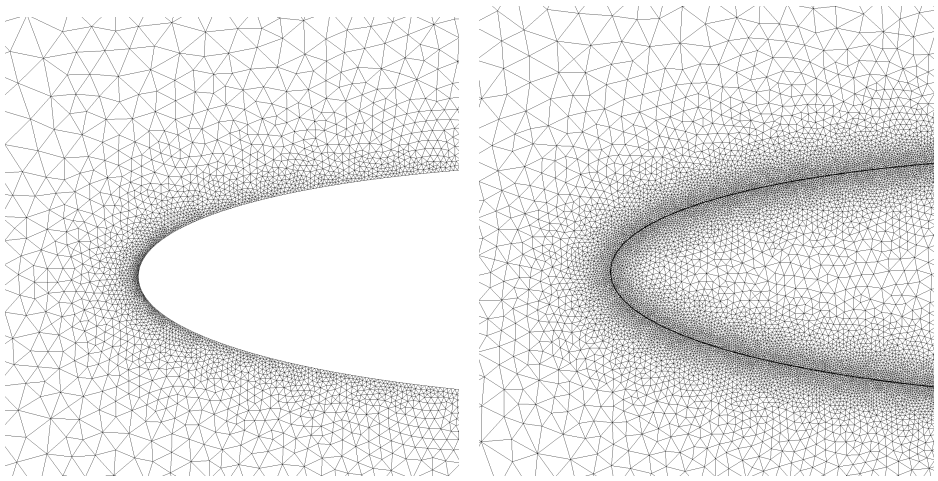
$$= \frac{\boldsymbol{\nu}_i^{\boldsymbol{T}}}{2} \tag{59}$$

3 The immersed boundary method

ersed_boundaries

Traditionally internal fluid boundarys, like physical walls imposed by a structure, are represented by meshing the fluid such that it coincides with the structure surface. This approach, that shall be denoted by "explicit boundary method" here, is intuitive and the imposition of boundary conditions comes naturally. However, in the case of fluid structure interaction, where we expect the structure to move and deform, an explicit boundary treatment, as described above, prohibits the use of an Eulerian formulation, as described in section 1.1. Instead every move or shape-change of the structure requires an appropriate update of the fluid mesh. Therfore, Fluid Structure Interaction (FSI) problems with explicit boundaries are typically solved via an ALE formulation. Details can be found in section 1.1.

acro:FSI



(a) classical wall boundary *(b) immersed wall boundary*

Figure 3: NACA 0012 profile meshed with a classical wall boundary on the left versus an immersed boundary on the right. This geometry is relatively simple, for more complex examples, however, not requiring a conforming fluid grid on the interface can be a huge advantage.

_vs_embedded

However, using an ALE-formulation has some severe disadvantages. First and most importantly for the context of shape-optimization, only minor changes to the topology are admissible. The mesh-motion or mesh-update algorithm is only capable of moving the fluid nodes in a spatial manner, the main connectivity of the mesh does not change, however. Therefor, the method can neither cover large distortions, nor is it possible to fundamentally change the structure shape, e.g. removing material in a non-critical region of the structure.
Additionally, creating body-fitted meshed can be cumbersome and time consuming, especially for complex structure geometries.
An elegant solution to this problem is the so-called "immersed boundary method", first described in the context of flow-computations in the human heart in [12]. In an immersed boundary method, the structure surface does not coincide with fluid faces. Therefore, no mesh-movement algorithm is required, and the fluid mesh no longer needs to be specifically designed with a certain structure in mind. In fact, combing this approach with an apdaptive mesh refinement completely eliminates the cost in fluid mesh generation.
The two different approaches are visualized in Figure **fig:ale**$_v s_e mbedded. Duetotheattractivenessoftheir$

4 Fluid structure interaction

Fluid Structure Interaction (FSI) describes the aeroelastic response of a mechanical system under fluid load, as well as the interaction between fluid and structure itself. Synonymously the term "Aeroelasticity" is often used.
The ultimate goal of the Sensitivity Analysis (SA) derived in this thesis is shape-optimization. For many challenging applications, the interaction of the flow with the structure can not be neglected for the sensitivity computation. Thus, this chapter shall provide a quick overview on the fluid-structure sensitivity analysis.
For this thesis, we first will focus on an ALE formulation, a closer look into this topic in the context of en Embedded formulation will follow.

4.1 Coupled three-field formulation

ield_formulation

In an aeroelastic ALE problem, three solution fields have to be considered. Firstly, the usual fluid problem has to be solved. Over the FSI interface, the fluid now interacts with the structure, imposing a Neumann problem on it. Finally, since we are considering ALE here, a governing equation for the fluid mesh motion has to be introduced, which closes the system.
Generally speaking, there exist two main approaches for solving this problem. Firstly, one can solve all three problems in one joint system, leading to a so-called monolithic algorithm. The advantage is, that the formulation is consistent and compared to the subsequently introduced staggered algorithm it does not introduce errors in the coupling. However, the solution of the monolithic system is cumbersome, since completely different physics are involved.
In practice, using a so-called "staggered algorithm" has shown to be a more robust choice. In a staggered algorithm, fluid- and structure problem are updated from time step n to $n+1$ one after the other. Therefore, consistency is lost, and (if no correction is applied) accuracy drops to first order. On the other hand each problem by itself can be treated by well established, optimized, existing solvers so the solution procedure is more robust.
This thesis utilizes the staggered approach. We have coupled a three-dimensional second order finite volume code [5] with a second order accurate finite element code[6] according to the popular three-field formulation of [4].

The three-field formulation of ALE aeroelasticity can be written as:

eq:3field_structure
eq:3field_str

$$\text{governing equation of structure:} \quad \mathcal{S}_{gov}(\boldsymbol{s}, \boldsymbol{u}, \dot{\boldsymbol{x}}, \dot{\boldsymbol{w}}) = \boldsymbol{0} \tag{60}$$

eq:3field_mesh
eq:3field_mes

$$\text{governing equation of meshmotion:} \quad \mathcal{D}_{gov}(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{x}) = \boldsymbol{0} \tag{61}$$

eq:3field_fluid
eq:3field_flu

$$\text{governing equation of the fluid:} \quad \mathcal{F}_{gov}(\boldsymbol{s}, \dot{\boldsymbol{x}}, \dot{\boldsymbol{w}}) = \boldsymbol{0} \tag{62}$$

(63)

Governing equation of the structure: For a simple linear case, that is assumed in this thesis, the structure state equation can be written as

$$\mathcal{S}_{gov} = \mathbf{K}\boldsymbol{u} - \boldsymbol{P}(\dot{\boldsymbol{x}}, \boldsymbol{w}) \tag{64}$$ eq:eos_struct

where $\mathbf{K}$ represents the finite element stiffness matrix and $\boldsymbol{P}$ denotes the external load vector that combines aerodynamic loads $\boldsymbol{P}_F$ that are inflected to the structure by the fluid, and gravity loads $\boldsymbol{P}_0$

$$\boldsymbol{P} = \boldsymbol{P}_0 + \boldsymbol{P}_F \tag{65}$$

Governing equation of the mesh motion: In an Arbitrary Lagrangian Eulerian formulation, the fluid mesh deforms with the structure. A governing equation is thus required to describe that deformation. Typically a simple linear pseudo finite element approach or a spring analogy method [3] is used to do so. A Dirichlet problem is solved to move the mesh.

$$\mathcal{D}_{gov} = \bar{\mathbf{K}}\dot{\boldsymbol{x}} \quad \text{with} \quad \dot{\boldsymbol{x}} = \boldsymbol{u} \text{ on } \Gamma_{F|S} \tag{66}$$ eq:eos_mesh

Governing equation of the fluid: The state equation of the fluid depends on the flow model used, as well as on whether a Eulerian or Lagrangian approach is used. A quick overview is provided in Section 1.2.
Please note, that the frame of reference(Eulerian, Lagrangian, Arbitrary Lagrangian Eulerian) has nothing to do with the fluid equation type.
The state equation of the fluid (62) can be expressed as

$$\mathcal{F}_{gov} = \begin{array}{|c|l|l|}\hline \multirow{3}{*}{Eulerian} & \text{Euler} & \frac{\partial \boldsymbol{w}}{\partial t} + \nabla\mathcal{F}(\boldsymbol{x}, \boldsymbol{w}) \\ \hline & \text{NSE} & \frac{\partial \boldsymbol{w}}{\partial t} + \nabla\mathcal{F}(\boldsymbol{x}, \boldsymbol{w}) + \nabla\mathcal{G}(\boldsymbol{x}, \boldsymbol{w}) \\ \hline & \text{RANS} & \frac{\partial \boldsymbol{w}}{\partial t} + \nabla\mathcal{F}(\boldsymbol{x}, \boldsymbol{w}) + \nabla\mathcal{G}(\boldsymbol{x}, \boldsymbol{w}) - \mathcal{S}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\chi}) \\ \hline \multirow{3}{*}{ALE} & \text{Euler} & \frac{\partial \boldsymbol{J}\boldsymbol{w}}{\partial t} + \boldsymbol{J}\nabla_\xi\mathcal{F}^c(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}) \\ \hline & \text{NSE} & \frac{\partial \boldsymbol{J}\boldsymbol{w}}{\partial t} + \boldsymbol{J}\nabla_\xi\mathcal{F}^c(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}) + \boldsymbol{J}\nabla_\xi\mathcal{G}^c(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}) \\ \hline & \text{RANS} & \frac{\partial \boldsymbol{J}\boldsymbol{w}}{\partial t} + \boldsymbol{J}\nabla_\xi\mathcal{F}^c(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}) + \boldsymbol{J}\nabla_\xi\mathcal{G}^c(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}) - \mathcal{S}^c(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}, \boldsymbol{\chi}) \\ \hline \end{array} = \mathbf{0} \tag{67}$$

where

$$\mathcal{F}^c(\boldsymbol{w}) = \mathcal{F}(\boldsymbol{w}) - \dot{\xi}\boldsymbol{w} \tag{68}$$

After FV discretization of the convective part, and FE discretization of the diffusive part, as explained in section 2.2, the discrete system can be written as

$$\mathcal{F}_{gov} = \begin{array}{|c|l|l|}\hline \multirow{3}{*}{Eulerian} & \text{Euler} & \frac{\partial \mathbf{w}}{\partial t} + \mathbf{R}^c(\mathbf{x},\mathbf{w}) \\ \hline & \text{NSE} & \frac{\partial \mathbf{w}}{\partial t} + \mathbf{R}^c(\mathbf{x},\mathbf{w}) + \mathbf{G}(\mathbf{x},\mathbf{w}) \\ \hline & \text{RANS} & \frac{\partial \mathbf{w}}{\partial t} + \mathbf{R}^c(\mathbf{x},\mathbf{w}) + \mathbf{G}(\mathbf{x},\mathbf{w}) - \mathbf{S}(\mathbf{x},\mathbf{w},\chi) \\ \hline \multirow{3}{*}{ALE} & \text{Euler} & \frac{\partial \mathbf{A}(\mathbf{x})\mathbf{w}}{\partial t} + \mathbf{R}^c(\mathbf{x},\dot{\mathbf{x}},\mathbf{w}) \\ \hline & \text{NSE} & \frac{\partial \mathbf{A}(\mathbf{x})\mathbf{w}}{\partial t} + \mathbf{R}^c(\mathbf{x},\dot{\mathbf{x}},\mathbf{w}) + \mathbf{G}(\mathbf{x},\dot{\mathbf{x}},\mathbf{w}) \\ \hline & \text{RANS} & \frac{\partial \mathbf{A}(\mathbf{x})\mathbf{w}}{\partial t} + \mathbf{R}^c(\mathbf{x},\dot{\mathbf{x}},\mathbf{w}) + \mathbf{G}(\mathbf{x},\dot{\mathbf{x}},\mathbf{w}) - \mathbf{S}(\mathbf{x},\dot{\mathbf{x}},\mathbf{w},\chi) \\ \hline \end{array} = \mathbf{0} \quad (69)$$

eq:full_governing_equation

eq:full_gover

Where we remind the reader that, as explained in section 2.2, for a FV discretization, the fluid state $\mathbf{w}$ has to be interpreted as average value over the cell. We often take the cell average to be equal to the value at the cell center, an approximation that is second order accurate.
In the above equations $\mathbf{R}^c$ denotes the vector of Roe fluxes resulting from a second-order finite volume discretization of the convective part of the NSE, see section 2.2. Likewise, $\mathbf{G}$ denotes an approximation of the diffusive part. As explained in section 2.3 we chose a FE approximation over a FV one for this term.
$\mathbf{A}$ denotes the matrix of cell volumes and $\mathbf{S}$ denotes the matrix resulting from the additional turbulence closures in the RANS equations (see Section 1.2.7).
The above table also gives a very nice overview over the different equation types and the used simplifications. The Euler equations are obviously obtained by neglecting the diffusive part of the NSE. One can also see that the RANS equations are more difficult than the NSE themselves. However, as explained in 1.2.7, the advantage is that RANS can operate on much coarser meshes that do not have to resolve the small scale turbulence phenomena, which makes them more numerically efficient overall in many applications.

4.1.1 Staggered algorithm

ggered_algorithm

The three-field coupled system of equations (60)-(62), can be solved very efficiently with an iterative staggered procedure, such as the one proposed by by [4]. This work utilizes the second-order staggered algorithm described in Figure 4.

① Determining structural response For a given external load $\boldsymbol{P}^{(n)}$, the structural response $\boldsymbol{u}^{(n)}$ is determined by solving the state equation of the structure (60). To increase stability, and under-relaxation is typically performed:

$$\boldsymbol{u}^{(n)} = (1-\theta)\boldsymbol{u}^{(n-1)} + \theta\tilde{\boldsymbol{u}} \quad (70)$$

$$\tilde{\boldsymbol{u}} = \mathbf{K}^{-1}\boldsymbol{P}^{(n)} \quad (71)$$

Typically, the relaxation factor is chosen as $0.5 \leq \theta \leq 0.9$

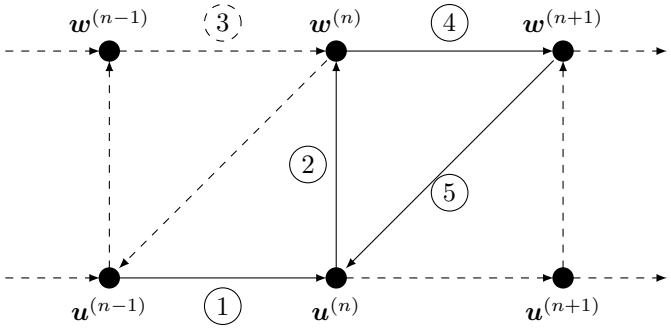


Figure 4: Scheme for the staggered algorithm. The picture depicts the sequential structure($\boldsymbol{u}$) and fluid($\boldsymbol{w}$) solves. First, the structure is updated to a new time step. With this information, a corresponding fluid solution is obtained. After the fluid is progressed in time, the structure solutions is being corrected with that information. A detailed description of all steps follows below.

ed_algorithm

② Motion transfer to wet surface The motion of the wet surface of the structure has to be transformed to the fluid.

$$\boldsymbol{u}_T^{(n)} = \mathbf{T}_u \boldsymbol{u}^{(n)} \tag{72}$$

where $\mathbf{T}_u$ is a transfer matrix that also accounts for potentially non-matching meshes [10].

③ Fluid mesh motion update The fluid mesh motion is then updated by solving an auxiliary pseudo-Dirichlet problem on the fluid mesh. This step is at the very heart of the ALE algorithm.

$$\bar{\mathbf{K}}\dot{\boldsymbol{x}}^{(n)} = 0 \qquad \dot{\boldsymbol{x}}^{(n)} = \boldsymbol{u}_T^{(n)} \text{ on } \Gamma_{F|S} \tag{73}$$

where the Dirichlet conditions are introduced into the system via a decomposition:

$$\bar{\mathbf{K}} = \begin{bmatrix} \bar{\mathbf{K}}_{\Omega\Omega} & \bar{\mathbf{K}}_{\Omega\Gamma} \\ \bar{\mathbf{K}}_{\Gamma\Omega} & \bar{\mathbf{K}}_{\Gamma\Gamma} \end{bmatrix}, \quad \dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{x}}_\Omega \\ \dot{\boldsymbol{x}}_\Gamma \end{bmatrix} \tag{74}$$

Here, the subscript Γ denotes the fluid grid points at the FSI interface and the subscript Ω denotes the remainder. Hence, the fluid mesh is updated in two steps, by first transferring the structure motion to the fluid interface, and then solving the auxiliary pseudo Dirichlet problem:

$$\dot{\boldsymbol{x}}_\Gamma^{(n)} = \mathbf{T}_u \boldsymbol{u}^{(n)} \tag{75}$$

eq:mms_update_1

eq:mms_update_1

$$\bar{\mathbf{K}}_{\Omega\Omega}\dot{\boldsymbol{x}}_\Omega^{(n)} = -\bar{\mathbf{K}}_{\Omega\Gamma}\dot{\boldsymbol{x}}^{(n)} \tag{76}$$

eq:mms_update_2

eq:mms_update_2

It shall be noted that an exact solution of the preceding equation is not required. In fact it is an arbitrarily chosen auxiliary pseudo-problem anyway. Therefor, a valid update of the mesh motion, that does not produce crossovers is enough. Such an update can be obtained at low cost by apllying a few Preconditioned Conjugate Gradient (PCG) iterations to Equation (76).

acro:PCG

④ Update fluid state vector After the mesh has been deformed, the fluid state $\mathbf{w}$ vector can be updated by apllying a singe Newton Rhapsodon iteration to equation (69), which leads to.

Eulerian	Euler	$\mathbf{R}^c(\mathbf{x}^{(n+1)}, \boldsymbol{w}^{(n)}) + \frac{\partial \mathbf{R}^c}{\partial \boldsymbol{w}}(\mathbf{x}^{(n+1)}, \boldsymbol{w}^{(n+1)} - \boldsymbol{w}^{(n)})$
	NSE	$<$ term above $> + \frac{\partial \mathbf{G}}{\partial \boldsymbol{w}}(\mathbf{x}^{(n+1)}, \boldsymbol{w}^{(n+1)} - \boldsymbol{w}^{(n)})$
	RANS	$<$ term above $> - \frac{\partial \mathbf{S}}{\partial \boldsymbol{w}}(\mathbf{x}^{(n+1)}, \boldsymbol{w}^{(n+1)} - \boldsymbol{w}^{(n)}, \boldsymbol{\chi})$
ALE	Euler	$\mathbf{R}^c(\mathbf{x}^{(n+1)}, \dot{\mathbf{x}}^{(n+1)}, \boldsymbol{w}^{(n)}) + \frac{\partial \mathbf{R}^c}{\partial \boldsymbol{w}}(\mathbf{x}^{(n+1)}, \dot{\mathbf{x}}^{(n+1)}, \boldsymbol{w}^{(n+1)} - \boldsymbol{w}^{(n)})$
	NSE	$<$ term above $> + \frac{\partial \mathbf{G}}{\partial \boldsymbol{w}}(\mathbf{x}^{(n+1)}, \dot{\mathbf{x}}^{(n+1)}, \boldsymbol{w}^{(n+1)} - \boldsymbol{w}^{(n)})$
	RANS	$<$ term above $> - \frac{\partial \mathbf{S}}{\partial \boldsymbol{w}}(\mathbf{x}^{(n+1)}, \dot{\mathbf{x}}^{(n+1)}, \boldsymbol{w}^{(n+1)} - \boldsymbol{w}^{(n)}, \boldsymbol{\chi})$

$= 0$ (77)

is this correct? What do I do with the turbulence parameter?

⑤ Determining structural responses Once the new fluid-state vector has been determined, the pressure $p^{(n+1)}$ can be recovered. The pressure on the fluid interface side, than might need to be transfered to the structure via a transfer matrix $\mathbf{T}_u$ to account for potentially non-matching fluid-structure interfaces.

5 Optimization

sec:optimization

As mentioned in the introduction, the overall motivation for this thesis is optimization. A typicall optimization process, not restricted to fluid problems, is depicted in Figure 5.

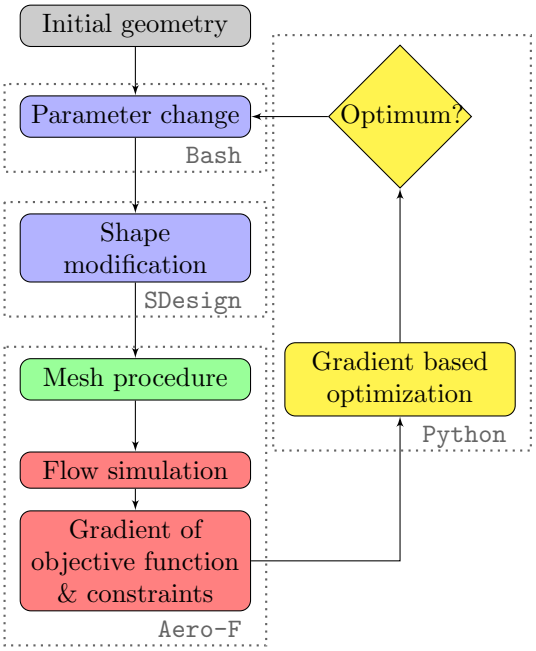


*Figure 5: Flow chart of an optimization process in the context of flow simulations. If one regards the flow-computation block as a black-box to compute gradients of an objective function and the shape-modification block more general as parameter modification, this optimization routine is applicable to all kinds of problems. In our case, we begin with an initial geometry created by a standard mesh-generation software. We also parametrize the initial mesh in terms of design elements in this first step. This is done via SDesign[**?**] in this thesis. All further modifications to the geometry are obtained via the design element concept. The mesh-movement, steady-state siumulations and Sensitivity Analysis (SA) are then performed in AERO-FAerof. For the optimizer itself, python has been chosen in this thesis, more specifically the **TODO** module. The codes communicate via result-files. These files are used to monitor and investigate the optimization process.*

ion_blockdiagram

A generic, aeroelastic optimization problem can be written as

$$q(\boldsymbol{s},\boldsymbol{u},\boldsymbol{w}) \begin{cases} \min\limits_{\boldsymbol{s}} z(\boldsymbol{s}) \\ \boldsymbol{h}(s) = \boldsymbol{0}, \quad \boldsymbol{h} \in \mathbb{R}^{n_h} \\ \boldsymbol{g}(s) > \boldsymbol{0}, \quad \boldsymbol{g} \in \mathbb{R}^{n_g} \end{cases} \quad \text{eq:generic_optimization_problem_1} \tag{78}$$

eq:generic_op

$$\boldsymbol{s} = \{\boldsymbol{s} \in \mathbb{R}^{n_s} \,|\, \boldsymbol{s}_L \leq \boldsymbol{s} \leq \boldsymbol{s}_U\} \quad \text{eq:generic_optimization_problem_2} \tag{79}$$

eq:generic_op

Here, z is an arbitrary cost function that should be minimized. Note that a maximization problem can easily be obtained by multiplying the cost function by a

factor of -1. In aerodynamics, a typical example for a cost function would be the Lift over Drag Ratio (LDR) of an airfoil. The cost-function is described in terms of so-called *abstract variables*. These can have some physical interpretation, but don't necessarily have to(see Section 5.2). Since typically, these optimization problems have no finite solution on an unbounded domain, some restrictions/conditions are introduced. In the airfoil example, we would probably specify a minimum lift that is required to support the configuration. Also a maximum stress for the structure will likely have to be specified. These are classical examples of non-equality constraints, denoted by $\boldsymbol{g}(\boldsymbol{s})$ in the above formulation. Constraints can also be formulated in an equality sense, for instance geometrical restrictions due to the turbine suspensions. Furthermore, the abstract optimization variables $\boldsymbol{s}$ themselves are typically restricted by lower($\boldsymbol{s}_L$) and upper($\boldsymbol{s}_U$) bounds.
The combinations of objective function z and constraints $\boldsymbol{h}$ and $\boldsymbol{g}$ is typically denoted as optimization criteria q.

$$\begin{aligned} q &= q(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{w}) \\ \boldsymbol{u} &= \boldsymbol{u}(\boldsymbol{s}), \quad \boldsymbol{w} = \boldsymbol{w}(\boldsymbol{s}) \end{aligned} \tag{80}$$

This thesis follows the *nested analysis and design approach*, meaning that we assume that $\boldsymbol{u}$ and $\boldsymbol{w}$ always satisfy the aeroelastic state equations. This means that the state equations are not included in the set of constraints, but the structural displacements $\boldsymbol{u}$ and the fluid state variables $\boldsymbol{w}$ are determined in each iteration.
As [10] write, the aeroelastic optimization problem can typically be solved by combining three different numerical approaches:

- Optimization Model
- Design Model
- Analysis Model

The optimization model describes the solution of the generic optimization problem(78)-(79). For this thesis, a Sequential Quadratic Programming (SQP) has been chosen [1].
The design model links abstract optimization variables $\boldsymbol{s}$ to actual shapes, structures, geometries or aerodynamics design parameters. For this purpose SDESIGN, a programm specifically written for SA purpose in the Farhat Research Group (FRG), has been used during this thesis. Its basic concepts an ideas are described in [11].
Finally, the analysis model provides concepts of evaluation the optimization criteria. Typically, the optimization criteria depend on $\boldsymbol{u}$ and $\boldsymbol{w}$ which is why a coupled system of equations has to be solved in every design optimization process. The Sensitivity Analysis (SA) is also part of this model. Aeroelastic analysis and Sensitivity analysis are discussed in Section 8.

5.1 Optimization Model

Optimization problems are typically solved by gradient-based methods. The methods are divided into

- Primal
- Dual
- Penalty-barrier and
- Lagrange approaches

This thesis focuses on Lagrange approaches. For a thorough analysis and comparison of the different approaches, the interested reader is referred to [15].
The Lagrangian approach formulates the optimization problem(78)(79) as an extreme value problem of the Lagrangian:

eq:lagrangian_of_optimization

$$L(\boldsymbol{s}, \boldsymbol{\eta}, \boldsymbol{\gamma}) = z(\boldsymbol{s})) = \boldsymbol{\eta}^T \boldsymbol{h}(\boldsymbol{s}) + \boldsymbol{\gamma}^T \boldsymbol{g}(\boldsymbol{s}) \tag{81}$$

eq:lagrangian

(82)

Here, $\boldsymbol{\eta}$ denote the Lagrange multipliers for the equality constraints and $\boldsymbol{\gamma}$ the Lagrange multipliers for the non-equality constraints. In fact, one can easily see that the original optimization problem can be obtained as saddle point of the Lagrangian:

eq:saddlepoint_optimization

$$\frac{\partial L}{\partial \boldsymbol{s}} = \frac{\partial z}{\partial \boldsymbol{s}} = \boldsymbol{\eta}^T \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{s}} + \boldsymbol{\gamma}^T \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{s}} \tag{83}$$

eq:saddlepoin

$$\frac{\partial L}{\partial \boldsymbol{\eta}} = \boldsymbol{h} = \boldsymbol{0} \tag{84}$$

$$\frac{\partial L}{\partial \boldsymbol{\gamma}} = \boldsymbol{\gamma}^T \boldsymbol{g} = \boldsymbol{0} \tag{85}$$

The SQP method, mention before uses a Newton-Rhapsodon algorithm to solve the above system.

eq:saddlepoint_newtonform

$$\begin{bmatrix} \frac{\partial^2 L}{\partial \boldsymbol{s}^2}^{(k)} & \frac{\partial \boldsymbol{g}}{\partial s}^{(k)} & \frac{\partial \boldsymbol{h}}{\partial s}^{(k)} \\ \boldsymbol{\gamma}^{(k)} \frac{\partial \boldsymbol{g}}{\partial s}^{(k)} & \boldsymbol{h}^{(k)} & \boldsymbol{0} \\ \frac{\partial \boldsymbol{h}}{\partial s}^{(k)} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{s} \\ \Delta \boldsymbol{\gamma} \\ \Delta \boldsymbol{\eta} \end{bmatrix} = - \begin{bmatrix} \frac{\partial L}{\partial \boldsymbol{s}}^{(k)} \\ \boldsymbol{\gamma}^{(k)} \boldsymbol{g}^{(k)} \\ \boldsymbol{h}^{(k)} \end{bmatrix} \tag{86}$$

eq:saddlepoin

Here $(\cdot)^{(k)}$ denotes the iteration index of the optimization loop.

The linear Equations (86) can also be formulated as an equivalent quadratic minimization problem:

eq:quadratic_minimization_problem_3

$$\begin{aligned} \min_{\boldsymbol{s}} \; & (\frac{1}{2} \Delta \boldsymbol{s}^T \frac{\partial^2 L}{\partial \boldsymbol{s}^2}^{(k)} \Delta \boldsymbol{s} + \frac{\partial z^{(k)}}{\partial \boldsymbol{s}}) \\ & \frac{\partial \boldsymbol{g}}{\partial s}^{(k)} \Delta \boldsymbol{s} + \boldsymbol{g}^{(k)} \geq \boldsymbol{0} \\ & \frac{\partial \boldsymbol{h}}{\partial s}^{(k)} \Delta \boldsymbol{s} + \boldsymbol{h}^{(k)} = \boldsymbol{0} \end{aligned} \tag{87}$$

eq:quadratic_

In the above formulation, the evaluation of the second derivative of the Lagrangian(Hessian of L) is the numerically most expensive part. Usually it is preferred to approximate it by a first-order information, for example by the Davidon-Fletcher Powell formula (DFP) or by the Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS) method. However, this simplification introduces an error that one should be aware of. Some correction methods have been proposed trying to minimize the error. In this thesis we adapt the one proposed by [10]:

acro:DFP
acro:BFGS

$$\begin{bmatrix} \boldsymbol{s}^{(k+1)} \\ \boldsymbol{\eta}^{(k+1)} \\ \boldsymbol{\gamma}^{(k+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}^{(k)} \\ \boldsymbol{\eta}^{(k)} \\ \boldsymbol{\gamma}^{(k)} \end{bmatrix} + \alpha^{(k)} \begin{bmatrix} \Delta\boldsymbol{s}^{(k)} \\ \Delta\boldsymbol{\eta}^{(k)} \\ \Delta\boldsymbol{\gamma}^{(k)} \end{bmatrix} \tag{88}$$

eq:correction_step
eq:correction_ste

The appropriate step size $\alpha^{(k)}$ is determined by a line search. AS [10] write, due to this insufficiencies, the Lagrangian can not be used to measure an improvement. Instead, a merit function is introduced that is minimized by the line-search. A local minimum is reached by following a sequence of decreasing merit function values. Convergence of the process can be measured via the $\mathcal{L}_2$-norm of the residual of the Kuhn-Tucker conditions(83).
By construction, the constraints are satisfied only when the optimum point is reached.

5.2 Design Model

design_model

The design model represent the essential link between the described optimization model and the analysis model. Generally speaking, it relates physical design parameters to abstract ones.

$$\boldsymbol{d} = \boldsymbol{d}(\boldsymbol{s}), \quad \boldsymbol{d} \in \mathbb{R}^{n_{\boldsymbol{d}}} \tag{89}$$

eq:physvarsTOabsvars
eq:physvarsTOabsv

Here, $n_{\boldsymbol{d}}$ denotes the number of physical design parameters. To define a relation between the abstract optimization variables and the motion of the nodes, the following design model is introduced:

$$\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}(\boldsymbol{s}) \tag{90}$$

As [10] explain, it is unpractical to identify an abstract variable directly with an increment of the coordinate of a grid point. Instead, two approaches, namely geometrical and mechanical can be adopted for constructing the generic design model.

Mechanical approach In the mechanical approach, the shape variation is identified with a superposition of fictitious structural deformations $\bar{\boldsymbol{u}}_j$ due to fictitious

loads $\mu\bar{\boldsymbol{P}}_j$ and fictions support conditions

$$\dot{\boldsymbol{x}} = \sum_j \bar{\boldsymbol{u}}_j = \sum_j \bar{\mathbf{K}}_j^{-1} \mu_j \bar{\boldsymbol{P}}_j \tag{91}$$ eq:mechanical_approach

eq:mechanical

where $\bar{\mathbf{K}}_j$ is a fictitious pseudo structure stiffness matrix representing the fluid domain compatible to the current fictions support conditions.

Geometrical approach The geometrical approach describes the geometry of the structure or the fluid boundary the design element concept. Here, the shape of a body $\boldsymbol{X}$ is approximated by so-called design elements as follows:

$$\boldsymbol{X} = \sum_j \phi_j(\xi) \hat{\boldsymbol{X}}_j \tag{92}$$ eq:geometrical_approach

eq:geometrica

Here, $\phi_j(\xi)$ is a shape function, $\hat{\boldsymbol{X}}_j$ is the vector of control nodes and ξ represents the reference coordinate. In the design element concept, the variation of the control nodes of the design element is used to vary the shape of the body. The variation of the control nodes position denoted as "mesh-motion" is given as $\dot{\boldsymbol{x}} = \Delta\hat{\boldsymbol{X}}$

$$\dot{\boldsymbol{x}} = \sum_j \phi_j(\xi) \hat{\dot{\boldsymbol{x}}}_j \tag{93}$$ eq:mms

eq:mms

Just as in Finite Elements (FE), where the shape-functions can not only be used to describe the solution field, but also the body's geometry and material parameters, the design element concept can be applied to prescribe parameter distributions and their variations in the optimization process. A simple sketch of the concept is provided in Figure 6

Both approaches have pros and cons, that are quickly discussed in [11], this thesis utilizes the geometrical approach solemnly.

5.3 Analysis Model

c:analysis_model

In the analysis model, the optimization criteria q are determined for a given set of obtimization variables s. Since the optimization algorithm choses in this thesis is based on gradients, the gradients of the optimization criteria with respect to the abstract variables are determined in this step. A main focus on this thesis has been the appropriate evaluation of these derivatives, we therfore denote the subsequent chapter 6 to the issue.

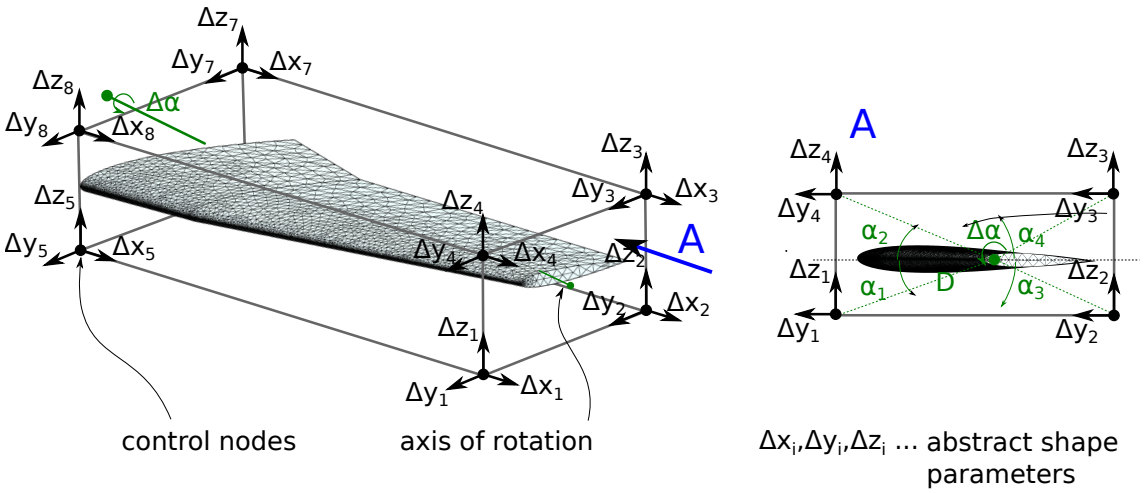


Figure 6: Geometrical approach for the design model as explained in Section 5.2. A NACA type airfoil is considered here as an example. The airfoil is embedded in a bounding box defined by eight so-called control nodes. The position of these control nodes can be varied, which alternates the shape of the airfoil according to some interpolation functions defined on the nodes. The 24 displacement unknowns are denoted as "abstract shape parameters". Abstract shape variables can be arbitrarily defined. As an example, a rotation axis is defined through the wing.

cal_approach

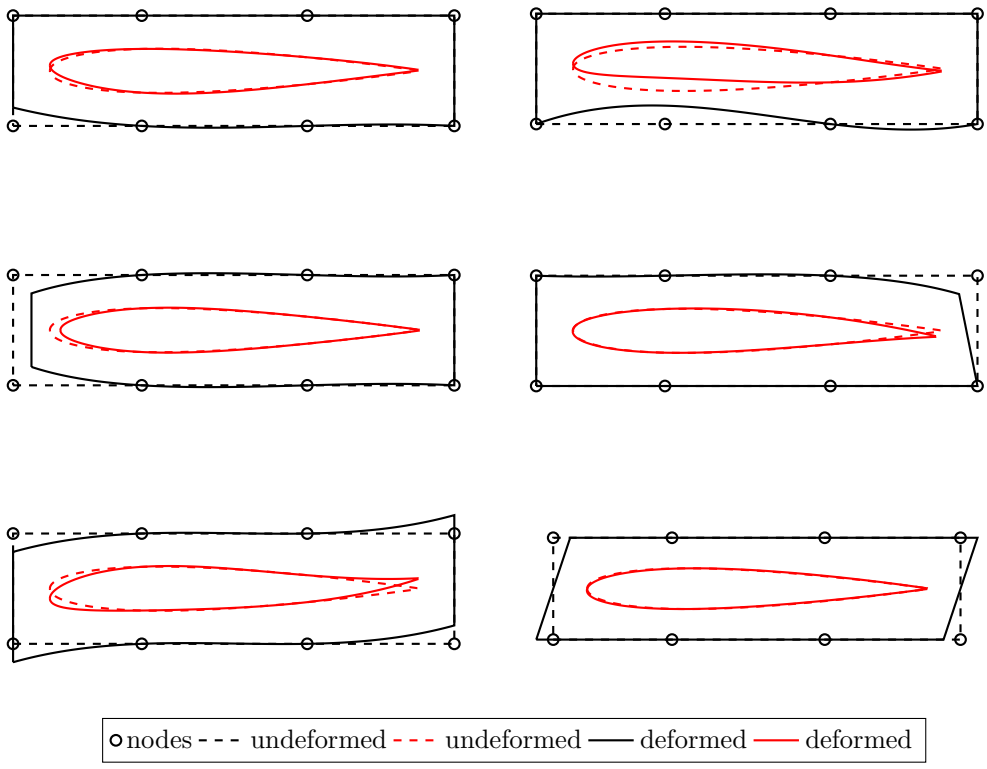


Figure 7: Visualization of the design element concept, exemplarily shown on a NACA0012 airfoil. For simplicity, a single design element(cubic in the horizontal direction and linear in the vertical direction) is used. Both the initial configuration(dashed lines) and the deformed configuration(full lines) are plotted. Although only eight design element nodes are provided, the function space of the design element allows for a great variation of the airfoil shape. Also, the size of the parameter space is now independent of the airfoil mesh size. From this perspective, the design element concept can be regarded as an model reduction for the shape variation. We will use exactly this parametrization of a NACA0012 airfoil here to verify the implemented shape-sensitivities in section 10

`ement_concept_2D`

6 Fluid Sensitivity Analysis (SA)

sec:SA

As explained in section 5, optimization is based on the calculations of gradients, also denoted as Sensitivity Analysis (SA). The derivation and implementation of these gradients constituted the lion-share of this thesis. We will therefore denote this section to the general derivation of fluid-sensitivity equations, before a detailed look into the most important terms within these equations is provided in section 7.

6.1 Connection between discretization and differentiation

_differentiation

When it comes to calculating the gradients of a solution of a PDE, there are two fundamentally different approaches one could take.
Firstly, one could think of first deriving the continuous equations and then applying the discretization scheme on them. On the other hand its equally legitimate to first apply the discretization and compute the gradients of this approximated solution. This thesis focuses on the latter for very particular reasons, see figure 8. If one would take the first approach, getting a steady state solution for the subsequent Sensitivity Analysis (SA) would entail the solution of the derived system of equations. This can not be done with standard CFD software. In addition to the SA routines, one would therefor have to implement an additional steady equation solver. This is not only uneconomical, it increases the risk of creating implementation bugs and also defeats the purpose of decades of research in stable and efficient NSE solvers.

6.2 Sensitivity derivation

When it comes to SA within the context of CFD, one also has to distinguish between SA of the non-coupled fluid problem and the SA in the fully coupled aeroelastic case. The latter case is much more involved, since it requires additional terms from the Structure and mesh motion equations. In the following we will therefore begin with the SA of a regular fluid problem. A generalization to a coupled FSI problem will follow.

As mentioned in Section 5, we typically deal with an optimization criteria q_j, that is in this case dependent on the fluid state variables $\boldsymbol{w}$ that themselves may depend on abstract variables s_i

$$q_j = q_j(\boldsymbol{w}(s_i)) \tag{94}$$

$$\left.\frac{dq_j}{ds_i}\right|_{\boldsymbol{w}_0} = \underbrace{\left.\frac{\partial q_j}{\partial s_i}\right|_{\boldsymbol{w}_0}}_{\substack{\text{directy derived from}\\ \text{the definition of } q}} + \underbrace{\left.\frac{\partial q_j}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}_0}}_{\substack{\text{derived analytically or}\\ \text{by FD}}} \underbrace{\left.\frac{d\boldsymbol{w}}{ds_i}\right|_{\boldsymbol{w}_0}}_{\substack{\text{derived from dynamic}\\ \text{fluid equilibrium}}} \tag{95}$$

eq:sensitivity_startingpoint

eq:sensitivit

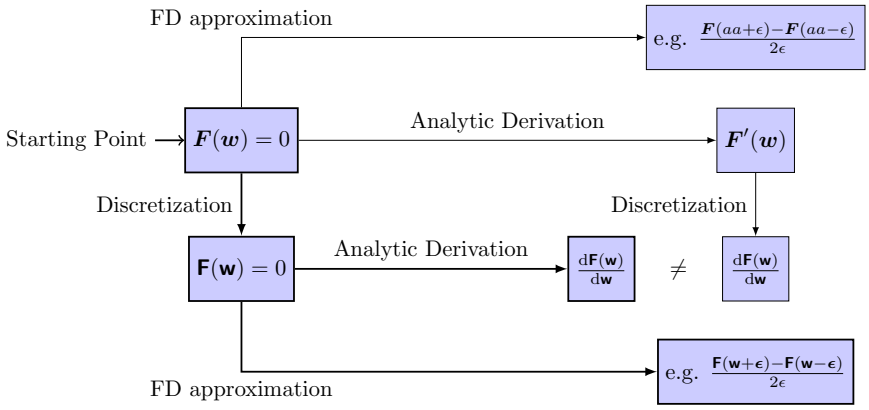


Figure 8: Two different approaches of SA. The question here is whether to discretize the system of equations first and then compute the derivatives of the approximate solution, or whether the continuous system of equation is first derived and discretized afterwards. Both approaches are valid, the final result however will generally not be the same. We have solemnly focused on the case of derivation after discretization in this thesis(thick lines). There are several reason to this. Firstly, if one chose to first derive and then compute a discretized solution, one would solve a completely different equation. If on the other hand the discretization(e.g. Finite Volumes) is done first, standard solvers can be utilized. What is more, approximating the derivative by a Finite difference becomes much easier, since the finite difference evaluation involves only a evaluation of $\mathbf{F}$*, which a standard fluid solver is perfectly capable of, whereas the FD approximation in the other case would involve evaluations of the continuous function and the discretization of the obtained expression which would be much more cumbersome.* ***Think about whether this is correct***

ion_sequence

6.2.1 Calculation of $\left.\frac{\partial q_j}{\partial s_i}\right|_{\boldsymbol{w}_0}$

In this thesis, we restrict ourselves to Lift(L), Drag(D) and combinations of these quantities(e.g. Lift-Drag ration $\frac{L}{D}$) as optimization criteria q. As abstract variables we only consider the free stream mach-number M_∞, the free-stream angle of attack α_∞ and an airfoil shape parameter as explained in section 5.2. The formula for the lift over an airfoil can be written as
If Γ_{FS} denotes the fluid structure interface, which in the ALE context coincides with the airfoil surface, one can formulate the lift and drag of an airfoil in a steady state as:

$$L = \int_{\Gamma_{FS}} p(\boldsymbol{w},\boldsymbol{x})\boldsymbol{n}(\boldsymbol{x})\cdot\boldsymbol{e}_L dS \tag{96}$$

eq:lift_and_drag_integrals

$$D = \int_{\Gamma_{FS}} p(\boldsymbol{w},\boldsymbol{x})\boldsymbol{n}(\boldsymbol{x})\cdot\boldsymbol{e}_D dS \tag{97}$$

eq:lift_and_drag_

where $\boldsymbol{e}_D$ is the unit vector pointing in the direction of the free stream, and $\boldsymbol{e}_L$ is

the unit vector perpendicular to that. We therefor note, that $\frac{\partial L}{\partial s_i}$ and $\frac{\partial D}{\partial s_i}$ are zero for $s_i = M_\infty$ and $s_i s = \alpha_\infty$ and non-zero if s_i is a shape parameter.
Also, having determined the derivatives $\frac{\partial L}{\partial s_i}$ and $\frac{\partial D}{\partial s_i}$, the derivative of the Lift-Drag ratio follows simply as

$$-\frac{\partial(\frac{L}{D})}{\partial s_i}|_{\boldsymbol{w}_0} = -\frac{D_0 \frac{\partial L}{\partial s_i}|_{\boldsymbol{w}_0} - L_0 \frac{\partial D}{\partial s_i}|_{\boldsymbol{w}_0}}{{D_0}^2} \quad \texttt{eq:lifttoddrag\_by\_absvar} \tag{98}$$

6.2.2 Calculation of $\left.\frac{\partial q_j}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}_0}$

The derivative of the optimization criteria with respect to the fluid state vector can again be splitted into Lift and Drag part. For an aeroelastic simulations, the biggest difficulty here is the dependence on the fluid state by the fluid-srtucture interface itself. We are therefore faced with a derivative of an integral quantity, where the integration area itself is dependent on the quantiti of interest.
For this purpose, we recall the leibnitz rule from calculus:

$$\frac{d}{dx}\Big(\int_{a(x)}^{b(x)} f(x,t)dt\Big) = f(x,b(x)) \cdot \frac{d}{dx}b(x) - f(x,a(x)) \cdot \frac{d}{dx}a(x) + \int_{a(x)}^{b(x)} \frac{d}{dx} f(x,t)dt \quad \texttt{eq:leibnitz\_rule} \tag{99}$$

Alternatively, we can directly switch to the discrete form, where the calculation of Lift and drag as outlined in Equation (96) can be performed as

$$L = \sum_{e \in \sum \Gamma_{FS}} \sum_{i=1}^{n_g} g_i \mathsf{p}(\mathbf{w}_i, \mathbf{x}_i) \boldsymbol{n}_e \cdot e_L D = \sum_{e \in \sum \Gamma_{FS}} \sum_{i=1}^{n_g} g_i \mathsf{p}(\mathbf{w}_i, \mathbf{x}_i) \boldsymbol{n}_e \cdot e_D \quad \texttt{eq:discrete\_lift\_and\_drag\_formulas} \tag{100}$$

where $\sum_{\Gamma_{FS}}$ is the set of surface elements on the fluid-structure interface, n_g is the number of gauss-points and g_i are the gauss weights. The dependency of the surface shape, on the fluid state vector is now reflected in non-zero derivatives of the gauss point locations.
Finally, the derivative of the lift-to-drag ratio can be obtained analogously to equation (98) as

$$\frac{\partial(\frac{\mathsf{L}}{\mathsf{D}})}{\partial \mathbf{w}}|_{\boldsymbol{w}_0} = \frac{\mathsf{D}_0 \frac{\partial \mathsf{L}}{\partial \mathbf{w}}|_{\boldsymbol{w}_0} - \mathsf{L}_0 \frac{\partial D}{\partial \mathbf{w}}|_{\mathbf{w}_0}}{{\mathsf{D}_0}^2} \quad \texttt{eq:lifttodrag\_by\_dfstate} \tag{101}$$

6.2.3 Calculation of $\left.\frac{d\boldsymbol{w}}{ds_i}\right|_{\boldsymbol{w}_0}$

Also, we keep in mind, that the state equation of the fluid can be expressed as

$$\mathcal{F}_{gov}(\boldsymbol{w}(s_i), \dot{\boldsymbol{x}}(s_i), s_i) = \frac{\partial \boldsymbol{w}(s_i)}{\partial t} + \nabla \cdot \mathcal{F}(\boldsymbol{w}(s_i)) + \nabla \mathcal{G}(\boldsymbol{w}(s_i)) + S = \mathbf{0} \tag{102}$$

After discretization, we have derived our discrete governing equation as

$$\frac{\partial \bar{\mathbf{w}}_i}{\partial t} + \sum_{j \in \kappa(i)} \boldsymbol{\phi}_{ij}(\mathbf{w}_{ij}, \mathbf{w}_{ji}, \boldsymbol{\nu}_{ij}) - \sum_{T_i \in \lambda(i)} \int_{T_j} \mathbb{K} \nabla \mathbf{w} \nabla \phi_j dx = \mathbf{0} \tag{103}$$ eq:nse_final_discretized_2 eq:nse_final_disc

For the pupose of Sensitivity analysis around a given steady state, the fluid-state vector does not change in time, and can thus be ommitted, also we summaraize the convective and the diffusivee part as $\mathbf{R}$

$$\cancelto{0}{\frac{\partial \bar{\mathbf{w}}_i}{\partial t}} + \underbrace{\underbrace{\sum_{j \in \kappa(i)} \boldsymbol{\phi}_{ij}(\mathbf{w}_{ij}, \mathbf{w}_{ji}, \boldsymbol{\nu}_{ij})}_{\mathbf{R}^i} - \underbrace{\sum_{T_i \in \lambda(i)} \int_{T_j} \mathbb{K} \nabla \mathbf{w} \nabla \phi_j dx}_{\mathbf{R}^v}}_{\mathbf{R}} = \mathbf{0} \tag{104}$$ eq:nse_final_discretized_notationchange eq:nse_final_disc

where $\mathbf{R}^i$ denotes the inviscid residual and $\mathbf{R}^v$ the viscous contribution. We now can express the steady state equation in very compact form as:

$$\mathbf{R} = \mathbf{0} \tag{105}$$ eq:discrete_steady_state eq:discrete_stead

In general, this residual depends on the fluid state solution $\mathbf{w}$ as well as on a potential mesh movement $\mathbf{x}$, both of which can be dependent on the abstract design variable. Additionally, a direct dependence on s might be encountered too.

$$\mathbf{R}(\mathbf{w}(s), \mathbf{x}(s), s) = \mathbf{0} \tag{106}$$

The total derivative of the residual with respect to the design variable can thus be expanded via the chain rule to

$$\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}s_i} = \mathbf{0} = \frac{\mathrm{d}\mathbf{R}}{\mathrm{d}s_i} + \frac{\partial \mathbf{R}}{\partial \mathbf{w}} \frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s_i} + \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s_i} \tag{107}$$

Therefore the total derivative of the fluid state with respect to the shape variable, needed in equation (95), can be obtained by solving

$$\frac{\partial \mathbf{R}}{\partial \mathbf{w}} \frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s_i} = -\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}s_i} - \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s_i} \tag{108}$$ eq:dfstate_by_absvarI eq:dfstate_by_abs

In this equation, $\dfrac{\partial \mathbf{R}}{\partial s_i}$ can be computed analytically or by FD. This part is the most cumbersome part of the whole sensitivity analysis, which is why we denoted a full chapter(7) to it.
The derivative of the mesh motion with respect to the abstract variables is often denoted as "shape gradient". It can be divided into two components:

- The interface component $\dfrac{\mathrm{d}\dot{\boldsymbol{x}}_\Gamma}{\mathrm{d}s_i}$, which is associated with the grid points lying on the fluid boundary

- The interior component $\frac{\mathrm{d}\dot{\boldsymbol{x}}_\Omega}{\mathrm{d}s_i}$, which is associated with the grid points located in the interior Ω of the computational domain.

The interface component is determined by the structure. Having obtained this one, the interior component can be computed by solving an auxiliary, fictitious Dirichlet problem:

$$\frac{\mathrm{d}\dot{\boldsymbol{x}}_\Omega}{\mathrm{d}s_i} = -\left[\bar{\mathbf{K}}_{\Omega\Omega}^{-1}\bar{\mathbf{K}}_{\Omega\Gamma}\right]\frac{\mathrm{d}\dot{\boldsymbol{x}}_\Gamma}{\mathrm{d}s_i} \tag{109}$$

where $\bar{\mathbf{K}}$ is a pseudo stiffness matrix that can be obtained by a simple spring analogy or similar approaches. In general, the mesh-motion algorithm should be chosen consistently with the one used in the calculation of the steady state solution. For the later introduced Embedded framework, $\frac{\mathrm{d}\dot{\boldsymbol{x}}}{\mathrm{d}s_i}$ is the position vector of the embedded discrete surface

6.3 Full sensitivity equation

itivity_equation

After inserting (98), (101) and (108) into (95), one can derive the final sensitivity equations for the special case of a rigid or non-existing structure as

$$\left.\frac{\mathrm{d}q_j}{\mathrm{d}s_i}\right|_{\mathbf{w}_0} = -\left.\frac{\mathrm{d}q_j}{\mathrm{d}\boldsymbol{w}}\right|_{\mathbf{w}_0}\left[\left.\frac{\partial\mathbf{R}}{\partial\mathbf{w}}\right|_{\boldsymbol{w}_0}\right]^{-1}\left(\left.\frac{\partial\mathbf{R}}{\partial s_i}\right|_{\boldsymbol{w}_0} + \left[\alpha\left.\frac{\partial\mathbf{R}}{\partial\dot{\mathbf{x}}_\Omega}\right|_{\mathbf{w}_0}\ \left.\frac{\partial\mathbf{R}}{\partial\dot{\mathbf{x}}_\Gamma}\right|_{\mathbf{w}_0}\right]\begin{bmatrix}\alpha\bar{\mathbf{K}}_{\Omega\Omega}^{-1}\bar{\mathbf{K}}_{\Omega\Gamma}\\ \mathbf{I}\end{bmatrix}\frac{\mathrm{d}\dot{\mathbf{x}}_\Gamma}{\mathrm{d}s_i}\right) \tag{110}$$

eq:full_sa_nostruct

eq:full_sa_no

$$\alpha = \begin{cases}1 \text{ in ALE framework}\\ 0 \text{ in Embedded framework}\end{cases} \tag{111}$$

idual_derivative

7 Derivatives of the fluid residual

This section is denoted to a thorough derivation of the partial derivatives $\frac{\partial \mathbf{R}}{\partial \mathbf{w}}$ and $\frac{\partial \mathbf{R}}{\partial s}$ of the fluid residual equation. These are the main quantities needed in order to compute the sensitivity of the state vector $\frac{\partial \mathbf{w}}{\partial s}$ as shown in equation 110. We need this derivative in order to solve for $\frac{\mathbf{w}}{s}$ in equation (??).

c:fluid_jacobian

7.1 Fluid Jacobian $\frac{\partial \mathbf{R}}{\partial \mathbf{w}}$

According to (104), the total residual is comprised of a convective(inviscid) part and a viscous contributions. Thanks to the distributive property of the derivative operator, we can thus compute the derivative of the total residual as the sum of the derivatives of the components. This is particulary important for us, since we used two different discretization routines for the viscous and the inviscid part.

7.1.1 Derivative of the convective Jacobian $\frac{\partial \mathbf{R}^c}{\partial \mathbf{w}}$

Also, we will only discuss the special case of intersected elements in the following, since the special case of an non-intersected cell follows trivially from that.
For more details on the evaluation of the inviscid residual the reader is refered to section 2, specifically section 2.2 and section ?? for the incorporation of the immersed boundary conditions.
For the convective part of the Jacobian, $\mathbf{R}^c$ we can be therefor write

$$\mathbf{R}_{ij}^{c,i} = \boldsymbol{\phi}_{ij}^{i}(\tilde{\mathbf{w}}_{ij}, \tilde{\mathbf{w}}_{ij}^{*}, \boldsymbol{n}_{ij}) \tag{112}$$

Since the fluid normal is dependent on the mesh position this can also be written as

$$\mathbf{R}_{ij}^{c,i} = \boldsymbol{\phi}_{ij}^{i}(\tilde{\mathbf{w}}_{ij}, \tilde{\mathbf{w}}_{ij}^{*}, \boldsymbol{x}) \tag{113}$$

If the edge $i-j$ is intersected however, the half-Riemann problem approach as discussed in section REF is applied. Also, the quantity $\mathbf{w}_ij$ on the interface can be reconstructed in several way. In this thesis we applied a MUSCL technique. Of course, the reconstruction and limitation has to be taken into account as well.
Putting this all together, one can write

$$\frac{\partial \mathbf{R}_{ij}^{i}}{\partial \mathbf{w}_k} = \frac{\partial \mathbf{R}_{ij}^{i}}{\partial \tilde{\mathbf{w}}_{ij}^{\star}} \frac{\partial \tilde{\mathbf{w}}_{ij}^{\star}}{\partial \tilde{\mathbf{w}}_k} \frac{\partial \tilde{\mathbf{w}}_k}{\partial \mathbf{w}_k} + \frac{\partial \mathbf{R}_{ij}^{i}}{\partial \tilde{\mathbf{w}}_{ij}} \frac{\partial \tilde{\mathbf{w}}_{ij}}{\partial \tilde{\mathbf{w}}_k} \frac{\partial \tilde{\mathbf{w}}_k}{\partial \mathbf{w}_k} + \underbrace{\frac{\partial \mathbf{R}_{ij}^{c,i}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{n}_{ij}}}_{=\mathbf{0} \text{ for embedded}} \tag{114}$$

- Analytical Jacobian of the (Roe's) centering flux
- Analytical derivative of the solution of the 1D half-Riemann problem
- Analytical derivative of the MUSCL reconstruction and limitation

where, $(\tilde{\sim})$ denotes primitive quantitives, $\frac{\partial \tilde{\mathbf{w}}_k}{\partial \mathbf{w}_k}$ accounts for the conversion between primitive and conservative state vector, $\frac{\partial \mathbf{R}_{ij}^{c,i}}{\partial \tilde{\mathbf{w}}_{ij}^{\star}}$ is the analytical derivative of the MUSCL reconstruction and limitation and a is the analytical Jacobian of the Roe flux. Finally, $\frac{\partial \tilde{\mathbf{w}}_{ij}^{\star}}{\partial \tilde{\mathbf{w}}_k}$ accounts for the analytical derivative of the solution of the 1D half-Riemann problem. This also accounts for the only really new derivative to be considered in comparison to the cells far from the immersed boundaries.
The analytic Jacobian of the Roe flux is provided in section 7.2, the derivative of the solution of the 1D hgalf-Riemann problem is discussed in section **TODO**, the derivative of the MUSCL reconstruction in section **TODO** and finally, the derivative of the conversion to primitive variables is discussed in section **TODO**.

7.1.2 Derivative of the viscous Jacobian $\frac{\partial \mathbf{R}^c}{\partial \mathbf{w}}$

bian_viscous

For the discussion of the viscous Jacobian, we first remind the reader that an FE approach is used to evaluate this term, in contrast to the FV approach used for the inviscid contribution. For a detailed discussion the reader is referred to section 2.3 and the following sections.
As far as the residual evaluation itself goes, we recall that, once the ghost points have been populated(see section **TODO**), the residual is evaluated as though it had never been intersected in the first place. We can therefore write the viscous residual of an intersected cell as

$$\mathbf{R}^c(\tilde{\mathbf{w}}^a, \tilde{\mathbf{w}}^g(\tilde{\mathbf{w}}^a)) = \mathbf{0} \tag{115}$$

where $\tilde{\mathbf{w}}^a$ represents the fluid states of the active node, and $\tilde{\mathbf{w}}^g$ are the fluid states of the ghost nodes, which can themselveS be expressed in terms of the active nodes. An thus the derivative becomes:

$$\frac{\partial \mathbf{R}^v(\tilde{\mathbf{w}}^a, \tilde{\mathbf{w}}^g(\tilde{\mathbf{w}}^a))}{\partial \tilde{\mathbf{w}}} = \underbrace{\frac{\partial \mathbf{R}^v}{\partial \tilde{\mathbf{w}}^a}}_{\substack{\text{can be re-used from ALE} \\ \text{after the ghost-point population}}} + \frac{\partial \mathbf{R}^v}{\partial \tilde{\mathbf{w}}^g} \cdot \overbrace{\frac{\partial \tilde{\mathbf{w}}^g}{\partial \tilde{\mathbf{w}}^a}}^{\substack{\text{obtained during the} \\ \text{population process}}} \tag{116}$$

For the analytic viscous Jacobian contribution we don't have to make a difference between $\frac{\partial \mathbf{R}^c}{\partial \tilde{\mathbf{w}}^a}$ and $\frac{\partial \mathbf{R}^v}{\partial \tilde{\mathbf{w}}^g}$.
We recall the definition of the viscous residual as

eq:derivative_viscous_residual

$$\mathbf{R}_i^v = -\sum_{T_i \in \lambda(i)} \int_{T_j} \mathbb{K} \nabla \mathbf{w} \nabla \phi_j dx \tag{117}$$

eq:derivative_vis

again, we switch to primitive variables for easier evaluation

eq:derivative_viscous_residual_primitives

$$\mathbf{R}_i^v = -\sum_{T_i \in \lambda(i)} \int_{T_j} \mathbb{K} \nabla \mathbf{w} \nabla \phi_j dx \tag{118}$$

eq:derivative_vis

Applying a numerical integration such as gauss rule, this becomes

$$\mathbf{R}_i^v = -\sum_{T_i \in \lambda(i)} \sum_{i=1}^{n_g} w_i \tilde{\mathbb{K}} \nabla \tilde{\mathbf{w}}(\mathbf{x}_i) \nabla \phi_j(\mathbf{x}_i) dx \tag{119}$$ eq:aaa eq:aaa

Thus, due the constant nature of the diffusive tensor $\mathbb{K}$ the derivative can be written straigforwardly as

$$\frac{\partial \mathbf{R}_i^v}{\partial \tilde{\mathbf{w}}} = -\sum_{T_i \in \lambda(i)} \sum_{i=1}^{n_g} w_i \tilde{\mathbb{K}} \nabla \tilde{\mathbf{w}}(\mathbf{x}_i) \nabla \phi_j(\mathbf{x}_i) dx \tag{120}$$ eq:aaa eq:aaa

7.2 Analytic Jacobian of the Roe flux

jacobian_roeflux

This section provides a detailed step by step derivation of the analytic convective Jacobian. This section is lenghty and not necessarily required for understanding the rest of the thesis. For this reason it has been moved at the end of chapter 7. Nonetheless, in the authors opinion, it provides some crucial insight into the basics of SA, and it is certainly of great help and serves as a reference for anyone who wants to implement those derivatives himself.

$$\frac{\partial \mathbf{R}_i^c}{\partial \mathbf{w}} = \sum_{j \in \kappa(i)} \underbrace{\frac{\partial meas(C_{ij})}{\partial \mathbf{w}} \boldsymbol{\phi}_{ij}(\mathbf{w}_{ij}, \mathbf{w}_{ji}, \boldsymbol{\nu}_{ij})}_{=\mathbf{0} \text{ for embedded}} + meas(C_{ij}) \frac{\partial \boldsymbol{\phi}_{ij}(\mathbf{w}_{ij}, \mathbf{w}_{ji}, \boldsymbol{\nu}_{ij})}{\partial \mathbf{w}} \tag{121}$$

where the first term is always zero for embedded simulations.
TODO ask Farhat what to do with the first derivative.
First, we recall the definition of the numerical flux as

$$\boldsymbol{\phi}(\mathbf{w}_i, \mathbf{w}_j, \boldsymbol{n}_{ij}) = \frac{1}{2}\left[\mathcal{F}(\mathbf{w}_i) \cdot \boldsymbol{n}_{ij} + \mathcal{F}(\mathbf{w}_i) \cdot \boldsymbol{n}_{ij}\right] - \frac{1}{2}|\boldsymbol{A}_{\text{Roe}}(\mathbf{w}_i, \mathbf{w}_j, \boldsymbol{n}_{ij})|(\mathbf{w}_j - \mathbf{w}_i) \tag{122}$$ eq:roe_flux eq:roe_flux

Similar to the MUSCL procedure we transform the fluid state vector to primitive form

$$\boldsymbol{w} = \begin{pmatrix} \rho \\ \rho \boldsymbol{v} \\ E \end{pmatrix} \overset{\mathsf{U}}{\mapsto} \tilde{\boldsymbol{w}} = \begin{pmatrix} \rho \\ \boldsymbol{v} \\ p \end{pmatrix} \tag{123}$$

Of course, this transformation involves the EOS of the fluid. We will use $\tilde{(\cdot)}$ from now on top mark any quantity as based on/formulated in primitive variables.
The primitive state vector is also much more appealing when it comes to the implementation of boundary conditions, since the solution variables are separated there.

Next we recall the conservative form of the Euler equations, which was presented in section 1.2.4 as

$$\frac{\partial \boldsymbol{w}}{\partial t} + \nabla \cdot \mathcal{F}(\boldsymbol{w}) = \mathbf{0} \tag{124}$$ eq:t1

Where the convective flux matrix $\mathcal{F}$ is provided in equation (15), and can be written as

$$\mathcal{F}(\boldsymbol{w}) = \begin{pmatrix} \mathcal{F}_x(\boldsymbol{w}) & \mathcal{F}_x(\boldsymbol{w}) & \mathcal{F}_x(\boldsymbol{w}) \end{pmatrix} \tag{125}$$

Therefore, applying the Nabla operator in Equation (124) gives

$$\frac{\partial \boldsymbol{w}}{\partial t} + \underbrace{\frac{\partial \mathcal{F}_x(\boldsymbol{w})}{\partial \boldsymbol{w}}}_{\boldsymbol{A}} \frac{\partial \boldsymbol{w}}{\partial x} + \underbrace{\frac{\partial \mathcal{F}_y(\boldsymbol{w})}{\partial \boldsymbol{w}}}_{\boldsymbol{B}} \frac{\partial \boldsymbol{w}}{\partial y} + \underbrace{\frac{\partial \mathcal{F}_z(\boldsymbol{w})}{\partial \boldsymbol{w}}}_{\boldsymbol{C}} \frac{\partial \boldsymbol{w}}{\partial z} = \mathbf{0} \tag{126}$$ eq:euler_coinservative_splitted

where the matrices $\boldsymbol{A}$,$\boldsymbol{B}$ and $\boldsymbol{C}$ are called the *flux Jacobians*.
In Section 6, we have derived that for any sensitivity calculation, the derivative of the flux matrix with respect to the fluid state vector $\frac{\partial \mathsf{R}^c}{\partial \boldsymbol{w}}$ is required.
We have derived in Section 2.3 that at an interior vertex point of the fluid mesh, we have

$$[\mathbf{R}^c(\mathbf{w}, \mathbf{x}, \dot{\mathbf{x}})]_i = \sum_{j \in \kappa(i)} mes(\partial C_{ij}) \boldsymbol{\phi}(\mathbf{w}_i, \mathbf{w}_j, \boldsymbol{n}_{ij}) \tag{127}$$ eq:convectiveterm _approximation2

Since an equilibrium point is considered, $\dot{\mathbf{x}} = \mathbf{0}$ and is neglected in the following derivations.
In equation (43) we have introduced $\boldsymbol{\phi}$ as the numerical flux function. In this thesis, we consider the popular Roe flux, which is defined as given in equation (122).
Let us remind, that R_i^c is the convective residual at vertex i, $\kappa(i)$ is the set of vertices connected to vertex i by an edge, C_i is the control volume of the dual cell centered at vertex i and ∂C_{ij} is the segment of the boundary that intersects edge (ij) and $\boldsymbol{\nu}_{ij}$ is the outward-facing weighted normal to ∂C_{ij}. See figure 1 for a visualization of the setup.
From Equation (127) it is obvious that when looking for the derivative $\frac{\partial \mathsf{R}^c}{\partial \mathbf{w}}$, the derivative of the numerical flux with respect to the fluid state vector $\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{w}}$ is required. To make things easier we will in the following derive the primitive version of that quantity $\frac{\partial \tilde{\boldsymbol{\phi}}}{\partial \tilde{\mathbf{w}}}$

As [7] demonstrates, primitive version of the numerical Roe flux can be expressed as

$$\tilde{\boldsymbol{\phi}}(\tilde{\mathbf{w}}_i, \tilde{\mathbf{w}}_j, \boldsymbol{n}_{ij}) = \frac{1}{2}\left(\tilde{\mathcal{F}}(\tilde{\mathbf{w}}_i) \cdot \boldsymbol{n}_{ij} + \tilde{\mathcal{F}}(\tilde{\mathbf{w}}_i) \cdot \boldsymbol{n}_{ij}\right) - \frac{1}{2}\boldsymbol{P}^{-1}(\sim)|\boldsymbol{\Lambda}(\sim)|\boldsymbol{P}(\sim)(\tilde{\mathbf{w}}_j - \tilde{\mathbf{w}}_i) \tag{128}$$ eq:primitive_roe_flux

where, for ease of notation we omit the function arguments by simply writing $(\sim)$ as follows

$$\begin{aligned}
\boldsymbol{P}^{-1}(\sim) &= \boldsymbol{P}^{-1}\left(\boldsymbol{M}(\sim), \boldsymbol{n}_{ij}\right) \\
\boldsymbol{M}(\sim) &= \boldsymbol{M}\left(\tilde{\boldsymbol{w}}_i, \tilde{\boldsymbol{w}}_j\right) \\
\boldsymbol{\Lambda}(\sim) &= \boldsymbol{\Lambda}\left(\boldsymbol{M}(\sim), \boldsymbol{n}_{ij}\right) \\
\boldsymbol{P}(\sim) &= \boldsymbol{P}\left(\boldsymbol{M}(\sim), \boldsymbol{n}_{ij}\right)
\end{aligned} \tag{129}$$

eq:split_roeaveragingmatrix

eq:split_roea

Here, $\boldsymbol{M}(\tilde{\boldsymbol{w}}_i, \tilde{\boldsymbol{w}}_j)$ is the averaging function associated with the roe flux. It can be written as

$$\boldsymbol{M}(\tilde{\boldsymbol{w}}_i, \tilde{\boldsymbol{w}}_j) = \frac{1}{\sqrt{\rho_i} + \sqrt{\rho_j}} \left(\sqrt{\rho_i} \begin{bmatrix} \rho_i \\ \boldsymbol{v}_i \\ \mathsf{H}_i \end{bmatrix} + \sqrt{\rho_j} \begin{bmatrix} \rho_j \\ \boldsymbol{v}_j \\ \mathsf{H}_j \end{bmatrix} \right) \tag{130}$$

and $\boldsymbol{P}$ is the matrix whose columns are the right eigenvectors of the Jacobian matrix $\mathcal{A}$ of $\mathcal{F}$. The Jacobi matrix in primitive form can be written as

$$\tilde{\mathcal{A}} = \frac{\partial(\tilde{\mathcal{F}} \cdot \boldsymbol{n})}{\partial \tilde{\boldsymbol{w}}} \tag{131}$$

$$= \begin{bmatrix}
\boldsymbol{v} \cdot \boldsymbol{n} & \rho n_x & \rho n_y & \rho n_z & 0 \\
v_1(\boldsymbol{v} \cdot \boldsymbol{n}) & \rho(\boldsymbol{v} \cdot \boldsymbol{n}) + \rho v_1 n_1 & \rho v_1 n_2 & \rho v_1 n_3 & n_1 \\
v_2(\boldsymbol{v} \cdot \boldsymbol{n}) & \rho v_2 n_1 & \rho(\boldsymbol{v} \cdot \boldsymbol{n}) + \rho v_2 n_2 & \rho v_2 n_3 & n_2 \\
v_3(\boldsymbol{v} \cdot \boldsymbol{n}) & \rho v_3 n_1 & \rho v_3 n_2 & \rho(\boldsymbol{v} \cdot \boldsymbol{n}) + \rho v_3 n_3 & n_2 \\
A_{51} & A_{52} & A_{53} & A_{54} & A_{55}
\end{bmatrix}$$

$$\begin{aligned}
\mathcal{A}_{51} &= H\ (\boldsymbol{v} \cdot \boldsymbol{n}) - \rho \boldsymbol{v} \cdot \boldsymbol{n} \frac{\gamma p}{\rho(\gamma)} \\
\mathcal{A}_{52} &= \rho \boldsymbol{v} \cdot \boldsymbol{n} v_1 + \rho H n_1 \\
\mathcal{A}_{53} &= \rho \boldsymbol{v} \cdot \boldsymbol{n} v_2 + \rho H n_2 \\
\mathcal{A}_{54} &= \rho \boldsymbol{v} \cdot \boldsymbol{n} v_3 + \rho H n_3 \\
\mathcal{A}_{55} &= \rho \boldsymbol{v} \cdot \boldsymbol{n} \frac{\gamma}{\rho(\gamma - 1)}
\end{aligned} \tag{132}$$

For this Jacobian, one can now symbolically derive an eigenvalue decomposition, such that

$$\mathcal{A} = \boldsymbol{P}^{-1}\boldsymbol{\Lambda}\boldsymbol{P} \tag{133}$$

The total specific enthalpy for a perfect gas is given as

$$\mathsf{H} = \frac{\gamma p}{\rho(\gamma-1)} + \frac{1}{2}\boldsymbol{v}^T\boldsymbol{v} \tag{134}$$

where we note the following derivatives

$$\frac{\partial \mathsf{H}}{\partial \rho} = -\frac{\gamma p}{\rho^2(\gamma-1)} \qquad \frac{\partial \mathsf{H}}{\partial v_i} = v_i \qquad \frac{\partial \mathsf{H}}{\partial p} = \frac{\gamma}{\rho(\gamma-1)} \tag{135}$$

Therefore, the derivative of the averaging function with respect to the primitive fluid state vector can be calculated as

$$\begin{aligned}
\frac{\partial \boldsymbol{M}(\sim)}{\partial \rho_i} &= \frac{1}{2\sqrt{\rho_i}(\sqrt{\rho_i}+\sqrt{\rho_j})}\left(-\boldsymbol{M}(\sim) + \begin{bmatrix} 3\rho_i & 2\sqrt{\rho_i}\boldsymbol{v}_i & 2\mathsf{H}_i - 2\rho_i\frac{\gamma p}{\rho^2(\gamma-1)} \end{bmatrix}^T\right) \\
\frac{\partial \boldsymbol{M}(\sim)}{\partial v_i} &= \frac{1}{(\sqrt{\rho_i}+\sqrt{\rho_j})}\sqrt{\rho_i}\begin{bmatrix} 0 & \boldsymbol{e}_i{}^T & v_i \end{bmatrix}^T \\
\frac{\partial \boldsymbol{M}(\sim)}{\partial p_i} &= \frac{1}{(\sqrt{\rho_i}+\sqrt{\rho_j})}\sqrt{\rho_i}\begin{bmatrix} 0 & \boldsymbol{0}^T & \frac{\gamma}{\rho(\gamma-1)} \end{bmatrix}^T \\
\frac{\partial \boldsymbol{M}(\sim)}{\partial \tilde{\boldsymbol{w}}_i} &= \begin{bmatrix}
3\rho_i - \frac{\sqrt{\rho_i}\rho_i+\sqrt{\rho_j}\rho_j}{\sqrt{\rho_i}+\sqrt{\rho_j}} & 0 & 0 & 0 & 0 \\
v_{1i} - \frac{\sqrt{\rho_i}v_{1i}+\sqrt{\rho_j}v_{1j}}{\sqrt{\rho_i}+\sqrt{\rho_j}} & 2\rho_i & 0 & 0 & 0 \\
v_{2i} - \frac{\sqrt{\rho_i}v_{2i}+\sqrt{\rho_j}v_{2j}}{\sqrt{\rho_i}+\sqrt{\rho_j}} & 0 & 2\rho_i & 0 & 0 \\
v_{3i} - \frac{\sqrt{\rho_i}v_{3i}+\sqrt{\rho_j}v_{3j}}{\sqrt{\rho_i}+\sqrt{\rho_j}} & 0 & 0 & 2\rho_i & 0 \\
\frac{\boldsymbol{v}^T\boldsymbol{v}}{2} - \frac{\gamma p}{\rho(\gamma-1)} - \frac{\sqrt{\rho_i}\mathsf{H}_i+\sqrt{\rho_j}\mathsf{H}_j}{\sqrt{\rho_i}+\sqrt{\rho_j}} & 2\rho_i v_{1i} & 2\rho_i v_{1i} & 2\rho_i v_{1i} & \frac{2\gamma}{\gamma-1}
\end{bmatrix}
\end{aligned} \tag{136}$$

The matrix $\boldsymbol{P}$ can be written as

$$\boldsymbol{P} = \begin{bmatrix} \left(\boldsymbol{n} - \frac{(\gamma-1)\boldsymbol{v}^T\boldsymbol{v}}{2c^2}\boldsymbol{n} + \boldsymbol{n}\times\boldsymbol{v}\right)\cdot\boldsymbol{e}_1 & n_1(\gamma-1)\frac{v_1}{c^2} & n_1(\gamma-1)\frac{v_2}{c^2}+n_3 & n_1(\gamma-1)\frac{v_3}{c^2}-n_2 & -n_1\frac{\gamma-1}{c^2} \\ \left(\boldsymbol{n} - \frac{(\gamma-1)\boldsymbol{v}^T\boldsymbol{v}}{2c^2}\boldsymbol{n} + \boldsymbol{n}\times\boldsymbol{v}\right)\cdot\boldsymbol{e}_2 & n_2(\gamma-1)\frac{v_1}{c^2}-n_3 & n_2(\gamma-1)\frac{v_2}{c^2} & n_2(\gamma-1)\frac{v_3}{c^2}+n_1 & -n_2\frac{\gamma-1}{c^2} \\ \left(\boldsymbol{n} - \frac{(\gamma-1)\boldsymbol{v}^T\boldsymbol{v}}{2c^2}\boldsymbol{n} + \boldsymbol{n}\times\boldsymbol{v}\right)\cdot\boldsymbol{e}_3 & n_3(\gamma-1)\frac{v_1}{c^2}+n_2 & n_3(\gamma-1)\frac{v_2}{c^2}-n_1 & n_3(\gamma-1)\frac{v_3}{c^2} & -n_2\frac{\gamma-1}{c^2} \\ (\gamma-1)\frac{\boldsymbol{v}^T\boldsymbol{v}}{c^2} - c\boldsymbol{v}\cdot\boldsymbol{n} & -(\gamma-1)v_1+cn_1 & -(\gamma-1)v_2+cn_2 & -(\gamma-1)v_3+cn_3 & \gamma-1 \\ (\gamma-1)\frac{\boldsymbol{v}^T\boldsymbol{v}}{c^2} - c\boldsymbol{v}\cdot\boldsymbol{n} & -(\gamma-1)v_1-cn_1 & -(\gamma-1)v_2-cn_2 & -(\gamma-1)v_3-cn_3 & \gamma-1 \end{bmatrix} \tag{137}$$

where c is the speed of sound given by $c = \sqrt{\frac{\gamma p}{\rho}}$
The matrix $\boldsymbol{\Lambda}$ derives to

$$\boldsymbol{\Lambda} = diag([\boldsymbol{v}\cdot\boldsymbol{n}, \boldsymbol{v}\cdot\boldsymbol{n}, \boldsymbol{v}\cdot\boldsymbol{n}, \boldsymbol{v}\cdot\boldsymbol{n}+c, \boldsymbol{v}\cdot\boldsymbol{n}-c)]) \tag{138}$$ eq:jaceigvals_definition eq:jaceigvals

$$\frac{\partial \boldsymbol{P}^{-1}}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n})\cdot b = \left[\frac{\partial r_1}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n})\cdot b \quad \cdots \quad \frac{\partial r_5}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n})\cdot b\right] \tag{139}$$ eq:pdfrac_jaceigvalsBYfstate eq:pdfrac_jac

$$\frac{\partial r_1}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \boldsymbol{n}_x & 0 & 0 & 0 \\ 0 & 0 & \boldsymbol{n}_x & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{n}_x & 0 \\ 0 & v_1 n_1 & v_2 n_1 + n_3 & v_3 n_1 - n_2 & 0 \end{bmatrix} \tag{140}$$ eq:pdfrac_jaceigvecsBYfstate_comp1 eq:pdfrac_jac

$$\frac{\partial r_2}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \boldsymbol{n}_y & 0 & 0 & 0 \\ 0 & 0 & \boldsymbol{n}_y & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{n}_y & 0 \\ 0 & v_1 n_2 - n_3 & v_2 n_2 & v_3 n_2 - n_1 & 0 \end{bmatrix} \tag{141}$$ eq:pdfrac_jaceigvecsBYfstate_comp2 eq:pdfrac_jac

$$\frac{\partial r_3}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \boldsymbol{n}_z & 0 & 0 & 0 \\ 0 & 0 & \boldsymbol{n}_z & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{n}_z & 0 \\ 0 & v_1 n_3 + n_2 & v_2 n_3 - n_1 & v_3 n_3 & 0 \end{bmatrix} \quad \texttt{eq:pdfrac\_jaceigvecsBYfstate\_comp3} \quad (142) \quad \texttt{eq:pdfrac\_jaceigv}$$

$$\frac{\partial r_4}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \begin{bmatrix} \frac{1}{2\gamma p} & 0 & 0 & 0 & -\frac{\rho}{2\gamma p^2} \\ \frac{v_1}{2\rho c^2} + \frac{n_1}{4\rho c} & \frac{1}{2c^2} & 0 & 0 & -\frac{v_1}{2pc^2} - \frac{n_1}{4pc} \\ \frac{v_2}{2\rho c^2} + \frac{n_2}{4\rho c} & 0 & \frac{1}{2c^2} & 0 & -\frac{v_2}{2pc^2} - \frac{n_2}{4pc} \\ \frac{v_3}{2\rho c^2} + \frac{n_3}{4\rho c} & 0 & 0 & \frac{1}{2c^2} & -\frac{v_3}{2pc^2} - \frac{n_3}{4pc} \\ \frac{1}{2\gamma} + \frac{\boldsymbol{v}\cdot\boldsymbol{n}}{4\rho c} & \frac{n_1}{2c} & \frac{n_2}{2c} & \frac{n_3}{2c} & -\frac{\boldsymbol{v}\cdot\boldsymbol{n}}{4pc} \end{bmatrix} \quad \texttt{eq:pdfrac\_jaceigvecsBYfstate\_comp4} \quad (143) \quad \texttt{eq:pdfrac\_jaceigv}$$

$$\frac{\partial r_5}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \begin{bmatrix} \frac{1}{2\gamma p} & 0 & 0 & 0 & -\frac{\rho}{2\gamma p^2} \\ \frac{v_1}{2\rho c^2} - \frac{n_1}{4\rho c} & \frac{1}{2c^2} & 0 & 0 & -\frac{v_1}{2pc^2} + \frac{n_1}{4pc} \\ \frac{v_2}{2\rho c^2} - \frac{n_2}{4\rho c} & 0 & \frac{1}{2c^2} & 0 & -\frac{v_2}{2pc^2} + \frac{n_2}{4pc} \\ \frac{v_3}{2\rho c^2} - \frac{n_3}{4\rho c} & 0 & 0 & \frac{1}{2c^2} & -\frac{v_3}{2pc^2} + \frac{n_3}{4pc} \\ \frac{1}{2\gamma} - \frac{\boldsymbol{v}\cdot\boldsymbol{n}}{4\rho c} & -\frac{n_1}{2c} & -\frac{n_2}{2c} & -\frac{n_3}{2c} & \frac{\boldsymbol{v}\cdot\boldsymbol{n}}{4pc} \end{bmatrix} \quad \texttt{eq:pdfrac\_jaceigvecsBYfstate\_comp5} \quad (144) \quad \texttt{eq:pdfrac\_jaceigv}$$

Due to its shape, the derivation of $\boldsymbol{\Lambda}$ can be simplified to a differentiation of the eigenvalues λ_i.

$$\frac{\partial \lambda_1}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \frac{\partial \lambda_2}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \frac{\partial \lambda_3}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \begin{bmatrix} 0 & \boldsymbol{n}^T & 0 \end{bmatrix}^T \quad \texttt{eq:pdfrac\_jaceigvalsBYfstate} \quad (145) \quad \texttt{eq:pdfrac\_jaceigv}$$

$$\frac{\partial \lambda_4}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \begin{bmatrix} -\frac{c}{2\rho} & \boldsymbol{n}^T & \frac{c}{2p} \end{bmatrix}^T \quad (146)$$

$$\frac{\partial \lambda_5}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \begin{bmatrix} \frac{c}{2\rho} & \boldsymbol{n}^T & -\frac{c}{2p} \end{bmatrix}^T \quad (147)$$

$$\frac{\partial \boldsymbol{P}}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) \cdot b = \begin{bmatrix} \frac{\partial l_1}{\partial \tilde{\boldsymbol{w}}}^T(\tilde{\boldsymbol{w}}, \boldsymbol{n}) \cdot b \\ \frac{\partial l_2}{\partial \tilde{\boldsymbol{w}}}^T(\tilde{\boldsymbol{w}}, \boldsymbol{n}) \cdot b \\ \frac{\partial l_3}{\partial \tilde{\boldsymbol{w}}}^T(\tilde{\boldsymbol{w}}, \boldsymbol{n}) \cdot b \\ \frac{\partial l_4}{\partial \tilde{\boldsymbol{w}}}^T(\tilde{\boldsymbol{w}}, \boldsymbol{n}) \cdot b \\ \frac{\partial l_5}{\partial \tilde{\boldsymbol{w}}}^T(\tilde{\boldsymbol{w}}, \boldsymbol{n}) \cdot b \end{bmatrix} \quad \texttt{eq:pdfrac\_jaceigvalsBYfstate\_comp1} \tag{148}$$

$$\frac{\partial l_1}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \begin{bmatrix} -\frac{n_1(\gamma-1)\|\boldsymbol{v}\|^2}{2\rho c^2} & -\frac{n_1(\gamma-1)v_1}{c^2} & -\frac{n_1(\gamma-1)v_2}{c^2} - n_3 & -\frac{n_1(\gamma-1)v_3}{c^2} + n_2 & \frac{n_1(\gamma-1)\|\boldsymbol{v}\|^2}{2\rho c^2} \\ \frac{n_1(\gamma-1)v_1}{\gamma p} & \frac{n_1\rho(\gamma-1)}{\gamma p} & 0 & 0 & -\frac{n_1 v_1\rho(\gamma-1)}{\gamma p^2} \\ \frac{n_1(\gamma-1)v_2}{\gamma p} & 0 & \frac{n_1\rho(\gamma-1)}{\gamma p} & 0 & -\frac{n_1 v_2\rho(\gamma-1)}{\gamma p^2} \\ \frac{n_1(\gamma-1)v_3}{\gamma p} & 0 & 0 & \frac{n_1\rho(\gamma-1)}{\gamma p} & -\frac{n_1 v_3\rho(\gamma-1)}{\gamma p^2} \\ -\frac{n_1\gamma-1}{\rho c^2} & 0 & 0 & 0 & \frac{n_1\gamma-1}{pc^2} \end{bmatrix} \quad \texttt{eq:pdfrac\_jaceigvalsBYfstate\_comp1} \tag{149}$$

$$\frac{\partial l_2}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \begin{bmatrix} -\frac{n_2(\gamma-1)\|\boldsymbol{v}\|^2}{2\rho c^2} & -\frac{n_2(\gamma-1)v_1}{c^2} + n_3 & -\frac{n_2(\gamma-1)v_2}{c^2} & -\frac{n_2(\gamma-1)v_3}{c^2} - n_1 & \frac{n_2(\gamma-1)\|\boldsymbol{v}\|^2}{2\rho c^2} \\ \frac{n_2(\gamma-1)v_1}{\gamma p} & \frac{n_2\rho(\gamma-1)}{\gamma p} & 0 & 0 & -\frac{n_2 v_1\rho(\gamma-1)}{\gamma p^2} \\ \frac{n_2(\gamma-1)v_2}{\gamma p} & 0 & \frac{n_2\rho(\gamma-1)}{\gamma p} & 0 & -\frac{n_2 v_2\rho(\gamma-1)}{\gamma p^2} \\ \frac{n_2(\gamma-1)v_3}{\gamma p} & 0 & 0 & \frac{n_2\rho(\gamma-1)}{\gamma p} & -\frac{n_2 v_3\rho(\gamma-1)}{\gamma p^2} \\ -\frac{n_2\gamma-1}{\rho c^2} & 0 & 0 & 0 & \frac{n_2\gamma-1}{pc^2} \end{bmatrix} \quad \texttt{eq:pdfrac\_jaceigvalsBYfstate\_comp2} \tag{150}$$

$$\frac{\partial l_3}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \begin{bmatrix} -\frac{n_3(\gamma-1)\|\boldsymbol{v}\|^2}{2\rho c^2} & -\frac{n_3(\gamma-1)v_1}{c^2} - n_2 & -\frac{n_3(\gamma-1)v_2}{c^2} + n_1 & -\frac{n_3(\gamma-1)v_3}{c^2} & \frac{n_3(\gamma-1)\|\boldsymbol{v}\|^2}{2\rho c^2} \\ \frac{n_3(\gamma-1)v_1}{\gamma p} & \frac{n_3\rho(\gamma-1)}{\gamma p} & 0 & 0 & -\frac{n_3 v_1 \rho(\gamma-1)}{\gamma p^2} \\ \frac{n_3(\gamma-1)v_2}{\gamma p} & 0 & \frac{n_2\rho(\gamma-1)}{\gamma p} & 0 & -\frac{n_3 v_2 \rho(\gamma-1)}{\gamma p^2} \\ \frac{n_3(\gamma-1)v_3}{\gamma p} & 0 & 0 & \frac{n_2\rho(\gamma-1)}{\gamma p} & -\frac{n_3 v_3 \rho(\gamma-1)}{\gamma p^2} \\ -\frac{n_3\gamma-1}{\rho c^2} & 0 & 0 & 0 & \frac{n_3\gamma-1}{pc^2} \end{bmatrix}$$

eq:pdfrac_jaceigvalsBYfstate_comp3 (151) eq:pdfrac_jaceigv

$$\frac{\partial l_4}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \begin{bmatrix} \frac{c}{2\rho}\boldsymbol{v}\cdot\boldsymbol{n} & (\gamma-1)v_1 - cn_1 & (\gamma-1)v_2 - cn_2 & (\gamma-1)v_3 - cn_3 & -\frac{c}{2p}\boldsymbol{v}\cdot\boldsymbol{n} \\ -\frac{cn_1}{2\rho} & -(\gamma-1) & 0 & 0 & \frac{cn_1}{2p} \\ -\frac{cn_2}{2\rho} & 0 & -(\gamma-1) & 0 & \frac{cn_2}{2p} \\ -\frac{cn_3}{2\rho} & 0 & 0 & -(\gamma-1) & \frac{cn_3}{2p} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

eq:pdfrac_jaceigvalsBYfstate_comp4 (152) eq:pdfrac_jaceigv

$$\frac{\partial l_5}{\partial \tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}, \boldsymbol{n}) = \begin{bmatrix} \frac{c}{2\rho}\boldsymbol{v}\cdot\boldsymbol{n} & (\gamma-1)v_1 - cn_1 & (\gamma-1)v_2 - cn_2 & (\gamma-1)v_3 - cn_3 & -\frac{c}{2p}\boldsymbol{v}\cdot\boldsymbol{n} \\ \frac{cn_1}{2\rho} & -(\gamma-1) & 0 & 0 & -\frac{cn_1}{2p} \\ \frac{cn_2}{2\rho} & 0 & -(\gamma-1) & 0 & -\frac{cn_2}{2p} \\ \frac{cn_3}{2\rho} & 0 & 0 & -(\gamma-1) & -\frac{cn_3}{2p} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

eq:pdfrac_jaceigvalsBYfstate_comp5 (153) eq:pdfrac_jaceigv

Therefor, the numerical roe flux (128) can therefore be derived by the primitive state vector as:

$$\frac{\partial \tilde{\boldsymbol{\phi}}}{\partial \tilde{\boldsymbol{w}}_i} = \frac{1}{2}\mathcal{A}(\tilde{\boldsymbol{w}}_i, \tilde{\boldsymbol{w}}_j) \tag{154}$$

$$- \frac{1}{2}\Bigg[\Big(\frac{\partial \boldsymbol{P}^{-1}}{\partial \tilde{\boldsymbol{w}}}(\sim)\frac{\partial \boldsymbol{M}}{\partial \tilde{\boldsymbol{w}}}(\sim)\Big)|\boldsymbol{\Lambda}(\sim)|\boldsymbol{P}(\sim)(\tilde{\boldsymbol{w}}_j - \tilde{\boldsymbol{w}}_i) \tag{155}$$

$$+ \boldsymbol{P}^{-1}(\sim)\Big(\frac{\partial |\boldsymbol{\Lambda}|}{\partial \tilde{\boldsymbol{w}}}(\sim)\frac{\partial \boldsymbol{M}}{\partial \tilde{\boldsymbol{w}}}(\sim)\Big)\boldsymbol{P}(\sim)(\tilde{\boldsymbol{w}}_j - \tilde{\boldsymbol{w}}_i) \tag{156}$$

$$+ \boldsymbol{P}^{-1}(\sim)|\boldsymbol{\Lambda}(\sim)|\Big(\frac{\partial \boldsymbol{P}}{\partial \tilde{\boldsymbol{w}}}(\sim)\frac{\partial \boldsymbol{M}}{\partial \tilde{\boldsymbol{w}}}(\sim)\Big)(\tilde{\boldsymbol{w}}_j - \tilde{\boldsymbol{w}}_i) \tag{157}$$

$$- \boldsymbol{P}^{-1}(\sim)|\boldsymbol{\Lambda}(\sim)|\boldsymbol{P}(\sim)\Bigg] \tag{158}$$

where the full analytic expression can be obtained by backward substitution of equations (129) to (153).
This derivative, along with the derivative of the viscous term, which will be derived in Section **??**, will be a key component in the Sensitivity calculations as shown in Section 6 and 8. This is due to the fact that will follow the procedere of discretization before derivation as justified in Section 6.1.

7.3 Derivative with respect to the abstract variable $\frac{\partial \mathbf{R}}{\partial s}$

sidual_by_absvar

Since the fluid Jacobian is also required in every standard fluid solve, the important term, unique to SA is the partial derivative of the residual with respect to the design variable s.
Similar to the previously discussed fluid Jacobian, the computation of $\frac{\partial \mathbf{R}}{\partial s}$ can be splitted into convective and viscous part. We again focus on the special case of an intersected cell in the following two subsections, since the standard case of a standard cell far from the interface follows trivially from that.
It shall also be noted that in most cases, the derivative $\frac{\partial \mathbf{R}}{\partial s}$ is non-zero only at a few fluid nodes. For shape-sensitivity, for example, only the nodes at the interface have a non-zero contribution here. Looking at equation (110) one can therefor interpret the solution of the linear system, or the multiplication with the inverse Jacobian, as a propagation operation.

7.3.1 Convective contribution $\frac{\partial \mathbf{R}^c}{\partial s}$

Generally, one can write the convective fluid residual as

$$\mathbf{R}^{c,i}_{ij} = \boldsymbol{\phi}^i_{ij}(\tilde{\mathbf{w}}_{ij}, \tilde{\mathbf{w}}^*_{ij}(s), \boldsymbol{n}_{ij}) \tag{159}$$

eq:convective_residual_fulldependency

eq:convective

The most important factor here, is of course the dependency of the half Riemann solution on the abstract variable. This dependecy becomes obvious in the case of shape sensitivity, since the intersection point used for the Riemann solution depends

upon the interface shape.
The derivative can be obtained straightforwardly as

$$\mathbf{RR}^{c,i}_{ij}\, s = \frac{\partial \mathbf{R}^{c,i}_{ij}}{\partial \mathbf{w}^{\star}_{ij}} \frac{\partial \mathbf{w}^{\star}_{ij}}{\partial s} \tag{160}$$

here, the analytical derivative of the first term can be taken from section 7.1. The second derivative is more cumbersome. The issue is visualized in Figure 9.

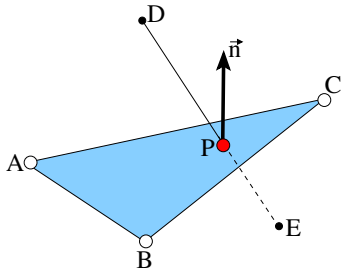


Figure 9: Visualization of the intersection of a fluid mesh edge $(D-E)$ with an element of the embedded surface $(A-B-C)$. For a shape-sensitivity analysis, SDesign returns the derivatives of the embedded surface nodes A, B, C position with respect to the shape parameter. This is done by first computing the intersection point P. Together with the element normal $\boldsymbol{n}$, the intersection point is used to reconstruct the shape derivatives at nodes D and E. ***TODO***

fig:pierce

Based on the consideration above, we find the following dependency for the

$$\frac{\partial \mathbf{w}^{\star}_{ij}}{\partial s} = \frac{\partial \mathbf{w}^{\star}_{ij}}{\partial s}\left(\frac{\partial \mathbf{w}_i}{\partial s}|_{i=A,B,C}, \xi, \boldsymbol{n}\right) \tag{161}$$

The derivatives of the geometrical quantities with respect to s are found as

$$\frac{\partial \xi}{\partial \boldsymbol{s}} = \frac{\partial \xi}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{s}} \tag{162}$$

$$\frac{\partial \boldsymbol{n}_{wall}}{\partial s} = \frac{\partial \boldsymbol{n}_{wall}}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial s} \tag{163}$$

7.3.2 Viscous contribution $\frac{\partial \mathbf{R}^v}{\partial s}$

The derivative of the viscous contibution is derived very simmilarly to the viscous Jacobian. Looking at equation **TODO** one can derive

$$\frac{\partial \mathbf{R}^v(\boldsymbol{s}, \tilde{\mathbf{w}}^a(\boldsymbol{s}), \tilde{\mathbf{w}}^g(\tilde{\mathbf{w}}^a(\boldsymbol{s})), \boldsymbol{x}(\boldsymbol{s}))}{\partial \boldsymbol{s}} = \underbrace{\frac{\partial \mathbf{R}^v}{\partial \tilde{\mathbf{w}}^a} \frac{\partial \tilde{\mathbf{w}}^a}{\partial \boldsymbol{s}}}_{\substack{\text{can be re-used from ALE} \\ \text{after the ghost-point population}}} + \frac{\partial \mathbf{R}^v}{\partial \tilde{\mathbf{w}}^g} \cdot \overbrace{\frac{\partial \tilde{\mathbf{w}}^g}{\partial \tilde{\mathbf{w}}^a}}^{\substack{\text{obtained during the} \\ \text{population process}}} \frac{\partial \tilde{\mathbf{w}}^a}{\partial \boldsymbol{s}} + \underbrace{\frac{\partial \mathbf{R}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{s}}}_{=\mathbf{0} \text{ for embedded}} \tag{164}$$

8 Aero-elastic Sensitivity Analysis

c:aeroelastic_sa

The previous chapter dealt with the SA of a pure fluid problem. Boundary, or respectively mesh-movement in general, was only induced due to shape variation via design variables itself. However, one can easily decieve a system, where the fluid interacts with an elastic structure. This would introduce an additional dependency of the structure shape, and thus fluid-structure interface, on the fluid solution. Also, a fully coupled system introduces a direct connection between the stress resultants in the structure with the fluid solutions. This chapter therfore investigates the SA of such a coupled system of equations. Considerations are build upon the derivations in chapter 6 and the coupled three-field FSI formulation described in chapter 4.1.
The SA approach applied in this thesis is based on the work of [17], for deriving
acro:GSE
the Global Sensitivity Equations (GSE) of coupled systems. As introduced by the authors of[10], we utilize the three-field formulation of [4].
The derivative of the optimization criterion q_j, as introduced in Equation (80), with respect to the optimization variable s_i gives:

$$\frac{dq_j}{ds_i} = \frac{\partial q_j}{\partial s_i} + \frac{\partial q_j}{\partial \boldsymbol{u}}\frac{d\boldsymbol{u}}{ds_i} + \frac{\partial q_j}{\partial \dot{\boldsymbol{x}}}\frac{d\dot{\boldsymbol{x}}}{ds_i} + \frac{\partial q_j}{\partial \boldsymbol{w}}\frac{d\boldsymbol{w}}{ds_i} \tag{165}$$

$$= \frac{\partial q_j}{\partial s_i} + \begin{bmatrix} \frac{\partial q_j}{\partial \boldsymbol{u}} \\ \frac{\partial q_j}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial q_j}{\partial \boldsymbol{w}} \end{bmatrix}^T \cdot \begin{bmatrix} \frac{d\boldsymbol{u}}{ds_i} \\ \frac{d\dot{\boldsymbol{x}}}{ds_i} \\ \frac{d\boldsymbol{w}}{ds_i} \end{bmatrix} \tag{166}$$

where the partial derivatives, $\frac{\partial q_j}{\partial \boldsymbol{u}}$, $\frac{\partial q_j}{\partial \dot{\boldsymbol{x}}}$ and $\frac{\partial q_j}{\partial \boldsymbol{w}}$ can be directly evaluated within the discretized structure and fluid model through the relation between structural, aerodynamic design and abstract optimization parameters defied in the design model 5.2.
The cumbersome part are the derivatives $\frac{d\boldsymbol{u}}{ds_i}$, $\frac{d\dot{\boldsymbol{x}}}{ds_i}$ and $\frac{d\boldsymbol{w}}{ds_i}$ To obtain them, the governing equations (??)**TODO write down governing equations** have to be derived:

$$\begin{bmatrix} \frac{\partial \mathcal{S}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{D}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{F}_{gov}}{\partial s_i} \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{\partial \mathcal{S}_{gov}}{\partial \boldsymbol{u}} & \frac{\partial \mathcal{S}_{gov}}{\partial \dot{\boldsymbol{x}}} & \frac{\partial \mathcal{S}_{gov}}{\partial \boldsymbol{w}} \\ \frac{\partial \mathcal{D}_{gov}}{\partial \boldsymbol{u}} & \frac{\partial \mathcal{D}_{gov}}{\partial \dot{\boldsymbol{x}}} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathcal{F}_{gov}}{\partial \dot{\boldsymbol{x}}} & \frac{\partial \mathcal{F}_{gov}}{\partial \boldsymbol{w}} \end{bmatrix}}_{\boldsymbol{A}} \begin{bmatrix} \frac{\partial \boldsymbol{u}}{\partial s_i} \\ \frac{\partial \dot{\boldsymbol{x}}}{\partial s_i} \\ \frac{\partial \boldsymbol{w}}{\partial s_i} \end{bmatrix} = \mathbf{0} \tag{167}$$

eq:governing_equations_derivative

eq:governing_e

In this equations $\frac{\partial \mathcal{S}_{gov}}{\partial s_i}$ and $\frac{\partial \mathcal{F}_{gov}}{\partial s_i}$ can be again directly evaluated using the relation specified in the design model. The matrix of first derivatives $\boldsymbol{A}$ is from now on denoted as the "Jacobian of the optimization problem".
Combining the previous two equations, it follows that the total derivative of the optimization criterion with respect to the abstract variables can be expressed as:

$$\frac{dq_j}{ds_i} = \frac{\partial q_j}{\partial s_i} - \underbrace{\begin{bmatrix} \frac{\partial q_j}{\partial \boldsymbol{u}} \\ \frac{\partial q_j}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial q_j}{\partial \boldsymbol{w}} \end{bmatrix}^T}_{n_{\boldsymbol{q}} \times n_{eq}} \underbrace{\boldsymbol{A}^{-1}}_{n_{eq} \times n_{eq}} \underbrace{\begin{bmatrix} \frac{\partial \mathcal{S}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{D}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{F}_{gov}}{\partial s_i} \end{bmatrix}}_{n_{eq} \times n_{\boldsymbol{s}}} \quad \texttt{eq:totderiv\_optcritBYabsvar} \tag{168}$$

Where n_{eq} is the total number of equations(e.g. five fluid state equations for the compressible NSG in 3D, three equations of the mesh motions and another three equations for the structure motion), n_q is the number of optimization criteria and n_s is the number of abstract variables.

8.1 Direct vs. adjoint approach

Equation 168 suggests, that there are two alternatives to compute vector-matrix-vector product above.

Direct approach Firstly, one could first compute the derivatives of the aeroelastic response for each abstract variable and perform the matrix product with $\boldsymbol{A}$:

$$\begin{bmatrix} \frac{d\boldsymbol{u}}{ds_i} \\ \frac{d\dot{\boldsymbol{x}}}{ds_i} \\ \frac{d\boldsymbol{w}}{ds_i} \end{bmatrix} = -\boldsymbol{A}^{-1} \begin{bmatrix} \frac{\partial \mathcal{S}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{D}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{F}_{gov}}{\partial s_i} \end{bmatrix} \quad \text{and then} \quad \texttt{eq:direct\_approach} \tag{169}$$

$$\frac{dq_j}{ds_i} = \frac{\partial q_j}{\partial s_i} - \begin{bmatrix} \frac{\partial q_j}{\partial \boldsymbol{u}} \\ \frac{\partial q_j}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial q_j}{\partial \boldsymbol{w}} \end{bmatrix}^T \begin{bmatrix} \frac{d\boldsymbol{u}}{ds_i} \\ \frac{d\dot{\boldsymbol{x}}}{ds_i} \\ \frac{d\boldsymbol{w}}{ds_i} \end{bmatrix} \tag{170}$$

Where the total complexity can be approximated as $\mathcal{O}(n_{eq}^2 n_{\boldsymbol{s}} + n_{\boldsymbol{q}} n_{eq} n_{\boldsymbol{s}})$

Adjoint approach Secondly, one could also first compute the derivatives of the optimization criteria and multiply with the Jacobian before substituting this into Equation (168):

$$\begin{bmatrix} \boldsymbol{a_u} \\ \boldsymbol{a_{\dot{x}}} \\ \boldsymbol{a_w} \end{bmatrix} = \boldsymbol{A}^{-T} \begin{bmatrix} \frac{\partial q_j}{\partial \boldsymbol{u}} \\ \frac{\partial q_j}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial q_j}{\partial \boldsymbol{w}} \end{bmatrix} \tag{171}$$

eq:firststep_adjoint

eq:firststep_

$$\frac{dq_j}{ds_i} = \frac{\partial q_j}{\partial s_i} - \begin{bmatrix} \boldsymbol{a_u} \\ \boldsymbol{a_{\dot{x}}} \\ \boldsymbol{a_w} \end{bmatrix}_j^T \begin{bmatrix} \frac{\partial \mathcal{S}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{D}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{F}_{gov}}{\partial s_i} \end{bmatrix} \tag{172}$$

Where the total complexity can be approximated as $\mathcal{O}(n_{eq}^2 n_{\boldsymbol{q}} + n_{\boldsymbol{q}} n_{eq} n_{\boldsymbol{s}})$

If one or the other approach is to be preferred depends in on the optimization setup, particularly the number of optimization criteria and the number of optimization variables. Looking at the orders above, one can conclude that if the number of abstract parameters $n_{\boldsymbol{s}}$ is smaller than the number of optimization criteria, the direct approach is more efficient, otherwise the adjoint approach is to be preferred. Additionally the on can argue, that the relevant term in the orders above is the one with n_{eq}^2 since it dominates the sum. Therefore, depending on whether n_s or n_q is bigger, the direct or the adjoint method is to be preferred.

8.1.1 Direct Sensitivity Analysis for the Euler equations

sec:direct_sa

The A matrix for the direct approach in ALE formulation looks like

$$\boldsymbol{A} = \begin{bmatrix} \frac{\partial \mathcal{S}_{gov}}{\partial \boldsymbol{u}} & \frac{\partial \mathcal{S}_{gov}}{\partial \dot{\boldsymbol{x}}} & \frac{\partial \mathcal{S}_{gov}}{\partial \boldsymbol{w}} \\ \frac{\partial \mathcal{D}_{gov}}{\partial \boldsymbol{u}} & \frac{\partial \mathcal{D}_{gov}}{\partial \dot{\boldsymbol{x}}} & \boldsymbol{0} \\ \boldsymbol{0} & \frac{\partial \mathcal{F}_{gov}}{\partial \dot{\boldsymbol{x}}} & \frac{\partial \mathcal{F}_{gov}}{\partial \boldsymbol{w}} \end{bmatrix} = \begin{bmatrix} \mathbf{K} & \frac{\partial \boldsymbol{P}_T}{\partial \dot{\boldsymbol{x}}} & \frac{\partial \boldsymbol{P}_T}{\partial \boldsymbol{w}} \\ \begin{bmatrix} \mathbf{K}_{\Omega\Gamma}\mathbf{T}_u \\ \mathbf{T}_u \end{bmatrix} & \begin{bmatrix} \mathbf{K}_{\Omega\Omega} & \boldsymbol{0} \\ \boldsymbol{0} & \mathbf{I} \end{bmatrix} & \boldsymbol{0} \\ \boldsymbol{0} & \frac{\partial \mathbf{F}_2}{\partial \dot{\boldsymbol{x}}_\Omega} & \boldsymbol{H}_2 \end{bmatrix} \tag{173}$$

eq:Amatrix_ALE

eq:Amatrix_AL

Where, $\boldsymbol{H}_2$ is the Jacobian of the second order row flux. It shall be noted that constructing this Jacobian is not a trivial issue and takes up a lot of computational resources, especially for FV, as described in [4]. Investigation into whether this term can be approximated at first order were carried out in [10] and [11].
Furthermore, [13] considered replacing the two mesh motion related matrices $\frac{\partial \boldsymbol{P}_T}{\partial \dot{\boldsymbol{x}}}$ and $\frac{\partial \mathbf{F}_2}{\partial \dot{\boldsymbol{x}}_\Omega}$ by a transpirational boundary condition. The consequences of this approach are also investigated in [10] and [11].
This thesis, however, does not use any of this simplifications.
The derivation of the sensitivities, can be achieved in a staggered scheme, very similar to the one, described in **??TODO**. It consists of five steps.

1) Update the structural displacement sensitivity to a new time step BY differentiating equations (60) and (64) and applying an under relaxation, we can obtain

$$\frac{\mathrm{d}\boldsymbol{u}^{(n)}}{\mathrm{d}s_i} = (1-\theta)\frac{\mathrm{d}\boldsymbol{u}^{(n)}}{\mathrm{d}s_i} + \theta\frac{\mathrm{d}\bar{\boldsymbol{u}}}{\mathrm{d}s_i} \quad \texttt{eq:underrelax\_structdisp} \tag{174}$$

eq:underrelax_str

where $\bar{\boldsymbol{u}}$ is obtained from:

$$\mathbf{K}\frac{\mathrm{d}\bar{\boldsymbol{u}}}{\mathrm{d}s_i} = \frac{\partial TODO}{\partial s_i} + \frac{\partial \mathbf{T}_u{}^{(n)}}{\partial s_i} - \frac{\partial \mathbf{K}}{\partial s_i}\boldsymbol{u} \quad \texttt{eq:fictious\_structdisp} \tag{175}$$

eq:fictious_struc

2) Transfer sensitivity of structure motion to the interface

$$\frac{\mathrm{d}\boldsymbol{u}_T^{(n)}}{\mathrm{d}s_i} = \mathbf{T}_u\frac{\mathrm{d}\boldsymbol{u}^{(n)}}{\mathrm{d}s_i} \quad \texttt{eq:interface\_projections} \tag{176}$$

eq:interface_proj

3) Compute derivative of fluid mesh motion The fluid mesh motion is computed by solving the pseudo Dirichlet problem as described in [4]. By design, the fictions stiffness matrix $\bar{\mathbf{K}}$ does not depend on the abstract optimization variables $\boldsymbol{s}$

$$\bar{\mathbf{K}}_{\Omega\Omega}\frac{\mathrm{d}\dot{\boldsymbol{x}}_\Omega^{(n)}}{\mathrm{d}s_i} = -\bar{\mathbf{K}}_{\Omega\Gamma}\frac{\mathrm{d}\dot{\boldsymbol{x}}_\Gamma^{(n)}}{\mathrm{d}s_i} \quad \texttt{eq:mms\_domain} \tag{177}$$

eq:mms_domain

with

$$\frac{\mathrm{d}\dot{\boldsymbol{x}}_\Gamma^{(n)}}{\mathrm{d}s_i} = \frac{\mathrm{d}\dot{\boldsymbol{x}}_\Gamma^{(n)}}{\mathrm{d}s_i} \tag{178}$$

4) Compute the sensitivity of the fluid state variables The derivatives of the fluid state variables are computed by

$$\boldsymbol{H}_2\frac{\mathrm{d}\boldsymbol{w}^{(n+1)}}{\mathrm{d}s_i} = \frac{\partial \boldsymbol{F}_2}{\partial s_i} - \frac{\partial \boldsymbol{F}_2}{\partial \dot{\boldsymbol{x}}}\frac{\mathrm{d}\dot{\boldsymbol{x}}}{\mathrm{d}s_i}^{(n)} \tag{179}$$

5) Compute the sensitivity of the structure load vector The derivative of the fluid load with respect to the abstract variables can be computed by the third of Equations (169) with the definition of $\boldsymbol{A}$ as specified in (173).

$$\frac{\partial \boldsymbol{P}_F{}^{(n+1)}}{\partial s_i} = \frac{\partial \boldsymbol{P}_F{}^{(n+1)}}{\partial \dot{\boldsymbol{x}}} \frac{\mathrm{d}\dot{\boldsymbol{x}}^{(n)}}{\mathrm{d}s_i} + \frac{\partial \boldsymbol{P}_F{}^{(n+1)}}{\partial \boldsymbol{w}} \frac{\mathrm{d}\boldsymbol{w}^{(n)}}{\mathrm{d}s_i} \tag{180}$$

eq:deriv_floadBYabsvar eq:deriv_floa

and compute project it onto the structure via

$$\frac{\partial \boldsymbol{P}_T{}^{(n+1)}}{\partial s_i} = \mathbf{T}_p \frac{\partial \boldsymbol{P}_F{}^{(n+1)}}{\partial s_i} \tag{181}$$

The convergence of the staggered algorithm can be monitored via

$$\left\| \mathbf{K} \frac{\mathrm{d}\bar{\boldsymbol{u}}^{(n+1)}}{\mathrm{d}s_i} - \frac{\partial TODO}{\partial s_i} - \frac{\partial \boldsymbol{P}_T{}^{(n+1)}}{\partial s_i} + \frac{\partial \mathbf{K}}{\partial s_i} \right\|_2 \leq \epsilon^{SA} \left\| \mathbf{K} \frac{\mathrm{d}\bar{\boldsymbol{u}}^{(0)}}{\mathrm{d}s_i} - \frac{\partial TODO}{\partial s_i} - \frac{\partial \boldsymbol{P}_T{}^{(0)}}{\partial s_i} + \frac{\partial \mathbf{K}}{\partial s_i} \right\|_2 \tag{182}$$

$$\left\| \boldsymbol{H}_2 \frac{\mathrm{d}\boldsymbol{w}^{(n+1)}}{\mathrm{d}s_i} + \frac{\partial \boldsymbol{F}_2}{\partial s_i} + \frac{\partial \boldsymbol{F}_2}{\partial \dot{\boldsymbol{x}}} \frac{\mathrm{d}\dot{\boldsymbol{x}}}{\mathrm{d}s_i}^{(n+1)} \right\|_2 \leq \epsilon^{SA} \left\| \boldsymbol{H}_2 \frac{\mathrm{d}\boldsymbol{w}^{(0)}}{\mathrm{d}s_i} + \frac{\partial \boldsymbol{F}_2}{\partial s_i} + \frac{\partial \boldsymbol{F}_2}{\partial \dot{\boldsymbol{x}}} \frac{\mathrm{d}\dot{\boldsymbol{x}}}{\mathrm{d}s_i}^{(0)} \right\|_2 \tag{183}$$

8.1.2 Adjoint Sensitivity Analysis for the Euler equations

sec:adjoint_sa

The adjoint SA follows the same scheme as the direct one Equation (171) can be written as:

$$\begin{bmatrix} \mathbf{K} & \begin{bmatrix} \frac{\partial \boldsymbol{P}_T}{\partial \dot{\boldsymbol{x}}_\Omega} & \frac{\partial \boldsymbol{P}_T}{\partial \dot{\boldsymbol{x}}_\Gamma} \end{bmatrix} & \frac{\partial \boldsymbol{P}_T}{\partial \boldsymbol{w}} \\ \begin{bmatrix} \mathbf{K}_{\Omega\Gamma}\mathbf{T}_u \\ \mathbf{T}_u \end{bmatrix} & \begin{bmatrix} \mathbf{K}_{\Omega\Omega} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \begin{bmatrix} \frac{\partial \mathbf{F}_2}{\partial \dot{\boldsymbol{x}}_\Omega} & \frac{\partial \mathbf{F}_2}{\partial \dot{\boldsymbol{x}}_\Gamma} \end{bmatrix} & \boldsymbol{H}_2 \end{bmatrix}^T \begin{bmatrix} \boldsymbol{a}_{\boldsymbol{u}} \\ \begin{bmatrix} \boldsymbol{a}_{\dot{\boldsymbol{x}}_\Omega} \\ \boldsymbol{a}_{\dot{\boldsymbol{x}}_\Gamma} \end{bmatrix} \\ \boldsymbol{a}_{\boldsymbol{w}} \end{bmatrix} = \begin{bmatrix} \frac{\partial q_j}{\partial \boldsymbol{u}} \\ \begin{bmatrix} \frac{\partial \boldsymbol{q}}{\partial \dot{\boldsymbol{x}}_\Omega} \\ \frac{\partial \boldsymbol{q}}{\partial \dot{\boldsymbol{x}}_\Gamma} \end{bmatrix} \\ \frac{\partial q_j}{\partial \boldsymbol{w}} \end{bmatrix} \tag{184}$$

eq:adjoint_equation eq:adjoint_eq

TODO check if the index j is really required here! A stated earlier, the matrices $\frac{\partial \boldsymbol{q}}{\partial \dot{\boldsymbol{x}}}$ and $\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{w}}$ can be computed analytically. As for $\boldsymbol{H}_2$, we follow the methodology outlined in [7] for evaluating ans storing it efficiently as the product of flux operators. Again the staggered procedure for solving the adjoint state problem shares the same computational kernels with the partitioned aeroelastic scheme described in [4]

1) Update the adjoint structure displacement to the new time step

$$\boldsymbol{a_u}^{(n+1)} = (1-\theta)\boldsymbol{a_u}^{(n)} + \theta\bar{\boldsymbol{a_u}}^{(n+1)} \tag{185}$$
$$\tag{186}$$

where $\bar{\boldsymbol{a_u}}^{(n+1)}$ is obtained from

$$\mathbf{K}\bar{\boldsymbol{a_u}}^{(n+1)} = \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{u}} - \mathbf{K}_{\Omega\Gamma}\mathbf{T}_u\boldsymbol{a_{x\Omega}}^{(n)} + \mathbf{T}_u\boldsymbol{a_{x\Gamma}}^{(n)} \tag{187}$$

TODO derive this equation

2) Compute the adjoint fluid state by solving

$$\boldsymbol{H_2}^T\boldsymbol{a_w}^{(n+1)} = \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{w}} + \frac{\partial \boldsymbol{P}_T}{\partial \boldsymbol{w}^T}\boldsymbol{a_u}^{(n+1)} \tag{188}$$

TODO derive this equation Again, $\dfrac{\partial \boldsymbol{q}}{\partial \boldsymbol{w}}$ is computed analytically from the relations defined in the design and aeroelastic model.

3) Compute adjoint mesh motion in domain and on the interface

$$\bar{\mathbf{K}}_{\Omega\Omega}\boldsymbol{a_{x\Omega}}^{(n+1)} = \frac{\partial \boldsymbol{q}}{\partial \dot{\boldsymbol{x}}_\Omega} - \frac{\partial \boldsymbol{P}_T}{\partial \dot{\boldsymbol{x}}_\Omega}\boldsymbol{a_u}^{(n+1)} - \frac{\partial \mathbf{F}_1{}^T}{\partial \dot{\boldsymbol{x}}_\Omega}\boldsymbol{a_x}^{(n+1)} \text{ in } \Omega \tag{189}$$

TODO check equations And the adjoint mesh motion on the interface is computed as $\boldsymbol{a_{x\Gamma}}^{(n+1)}$

$$\boldsymbol{a_{x\Gamma}}^{(n+1)} = \frac{\partial \boldsymbol{q}}{\partial \dot{\boldsymbol{x}}_{Gamma}} + \frac{\partial \boldsymbol{P}_T}{\partial \dot{\boldsymbol{x}}_\Gamma}\boldsymbol{a_u}^{(n+1)} - \frac{\partial \mathbf{F}_2}{\partial \dot{\boldsymbol{x}})_\Gamma}^T \text{ on } \Gamma \tag{190}$$

where $\dfrac{\partial \boldsymbol{q}}{\partial \boldsymbol{w}}$ is computed analytically.

The convergence of the staggered adjoint optimization algorithm can be monitored via

$$\left\| \boldsymbol{R_q} - \boldsymbol{A}^T\boldsymbol{a}^{(n+1)} \right\| \leq \epsilon^{SA} \left\| \boldsymbol{R_q} \right\| \tag{191}$$

9 The Embedded Boundary Method

_boundary_method

It was already outlined in Section 1, that using an Eulerian approach int the context of an aeroelastic simulation leads to a so-called embedded formulation, where the interface of the structure mesh no longer coincides with the fluid mesh. This was appealing, since it allowed for the usage of fixed meshes and an Eulerian formulation, on the other hand as explained in Section **??**, an error of the order $\mathcal{O}(\frac{h}{2})$ is introduced. A possible solution to recover second order convergence rate is quickly described in section **TODO**.

9.1 Setup

In this thesis, we consider embedded structure interfaces only. Whether the structure is deformable or rigid does not really make a difference for the subsequent considerations.
The basic setup is depicted in Figure 10.
Several issues have to be addresses in an embedded framework. Firstly, the interface may move during the simulation. In fact, large deformations of the structure have been on on the primary aspects for the development of an embedded framework in the first place. The question thus becomes how to track the interface during the simulation. This is addressed in section 9.5.
Secondly, the evaluation of the inviscid (192) and viscous (202) term becomes cumbersome for cells that are being intersected. We address this issue separate for the inviscid and the viscous term in sections 9.2 and 9.3.
Finally, in an FSI simulation we are typically interested in integral quantities over the structure surface, e.g. the lift and drag values of an airfoil. We now have several possibilities to define the structure surface to perform integration on in an embedded simulation. This issues will is discussed in section 9.4.

9.2 Evaluation of the inviscid term at the interface

er_inviscid_term

For this section as well as for section 9.3 we consider Figure 10.
A question, that arrises when lookin at equation (45) is how to evaluate those terms for node i, where some of the vertices in $\kappa(i)$ are inactive ghost nodes.
We will answer this question for the inviscid term in this subsections and look at the viscous term in the subsequent one.

First, we notice that we can split the summation as follows

$$\sum_{j\in\kappa(i)} \boldsymbol{\phi}_{ij}(\boldsymbol{w}_i,\boldsymbol{w}_j,\boldsymbol{\nu}_{ij}) = \sum_{j\in\kappa(i)^a} \boldsymbol{\phi}_{ij}(\boldsymbol{w}_i,\boldsymbol{w}_j,\boldsymbol{\nu}_{ij}) + \sum_{j\in\kappa(i)\setminus\kappa(i)^a} \boldsymbol{\phi}_{ij}(\boldsymbol{w}_i,\boldsymbol{w}^*,\boldsymbol{\nu}_{ij}) \tag{192}$$

eq:fiver_inviscid_split

eq:fiver_invi

where $\boldsymbol{w}_*$ is the fluid state at the auxiliary intersection between edge ij and the auxiliary interface. To obtain $\boldsymbol{w}_8$ a one-sided Riemann problem is defined

$$\frac{\partial\tilde{\boldsymbol{w}}}{\partial t} = \frac{\partial\mathcal{F}}{\partial s}(\tilde{\boldsymbol{w}}) = \boldsymbol{0} \tag{193}$$

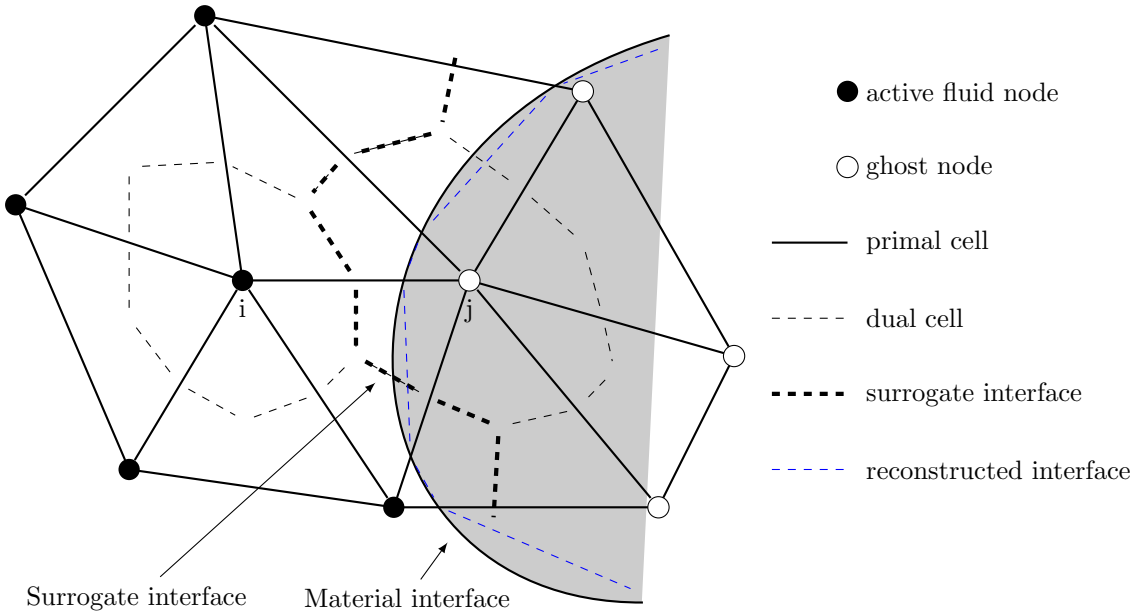


Figure 10: Sketch of an embedded simulation setup. The primal grid is intersected by a material interface. In FIVER, the material interface is replaced by a surrogate interface that is created by connecting the dual-grid interfaces closest to the embedded surface. By connecting the intersection points of the embedded surface with the primal grid, we can also create a so-called, reconstructed surface that is particularly useful for the calculation of forces on the surface.

intersection

where s is the local abscissa along the direction ij, that has its origin in M_{ij}.
The one sided Riemann problem can then be initialized with

$$\tilde{\boldsymbol{w}}_L = \begin{bmatrix} \rho_i & \boldsymbol{v}_i & p_i \end{bmatrix} \tag{194}$$

The exact solution of the one-sided, one-dimensional Riemann problem contains a constatn (in time) state at the fluid-structure interface which is denoted here by

$$\boldsymbol{w}_* = \begin{bmatrix} \rho_* & \boldsymbol{v}_* & p_* \end{bmatrix} \tag{195}$$

which can then be used in equation (192)

9.2.1 FIVER-1

eq:fiver-1

$$\begin{aligned} &\frac{\partial \tilde{\mathbf{w}}^\star}{\partial \tau} + \frac{\partial \widetilde{f}(\tilde{\mathbf{w}}^\star)}{\partial \xi} = \mathbf{0} \\ &\widetilde{f}^\star(\xi, 0) = \tilde{\boldsymbol{w}}_{ij} \qquad \xi < 0 \\ &\boldsymbol{v}(0,\tau) \cdot \boldsymbol{n}_{wall} = \boldsymbol{v}_{wall} \cdot \boldsymbol{n}_{wall} \qquad 0 \le t \le \tau \end{aligned} \tag{196}$$

eq:half_riemann_problem

eq:half_riemann_p

$$\boldsymbol{\phi}_{ij} = \boldsymbol{\phi}_{ij}(\tilde{\boldsymbol{w}}_{ij}, \tilde{\boldsymbol{w}}_b^\star, \boldsymbol{n}_{ij}) \tag{197}$$

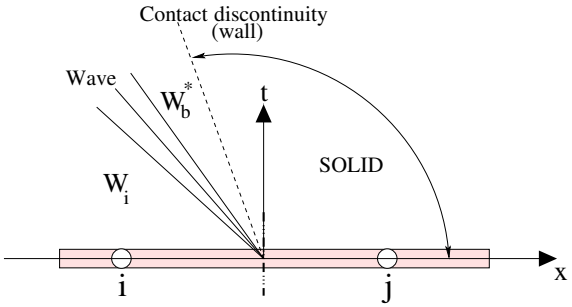


Figure 11: Solution of the 1D half Riemann problem around edge $i-j$ described in equation (196). The obtained fluid state at the interface is then used to evaluate the flux function, see equation (??)

g:riemann_sketch

9.2.2 FIVER-2

eq:fiver-2

The Fluid state is extrapolated to the material interface

$$\mathbf{w}_\Gamma = \mathbf{w}_i + \nabla \mathbf{w}_i (\mathbf{x}_\Gamma - \mathbf{x}_i) \tag{198}$$

The one-dimensional half-Riemann problem is solved at the material interface Γ

$$\tilde{\mathbf{w}}^\star_\Gamma = \tilde{\mathbf{w}}^\star(\tilde{\mathbf{w}}_\Gamma, \boldsymbol{v}_{wall}, \boldsymbol{n}_{wall}) \tag{199}$$

The fluid state is inter/extra-polated at the control volume interface C_{ij}

$$\tilde{\mathbf{w}}^\star_{ij} = \tilde{\mathbf{w}}^\star_{ij}(\mathbf{w}^\star_\Gamma, \mathbf{w}_i) \tag{200}$$

Numerical flux at the control volume interface:

$$\mathbf{R}^c_{ij} = \mathbf{R}^c_{ij}(\mathbf{w}_{ij}, \mathbf{w}^\star_{ij}, \boldsymbol{n}_{ij}) \tag{201}$$

It has been proven in **TODO** that this approach recovers second order convergence int the vicinity of the interface.

9.3 Evaluation of the viscous terms at the interface

ver_viscous_term

If one wants to keep the FE-like evaluation of the second term, (45) can be splitted as

$$\sum_{T_i \in \lambda(i)} \int_{T_i} \mathbb{K} \nabla \boldsymbol{w} \nabla \phi_i dx = \sum_{T_i \in (\lambda(i^a))} \int_{T_i} \mathbb{K} \nabla \boldsymbol{w} \nabla \phi_i dx + \sum_{T_i \in (\lambda(i) \setminus \lambda(i^a))} \int_{T_i} \mathbb{K} \nabla \boldsymbol{w}^R \nabla \phi_i dx = \tag{202}$$

eq:fiver_viscous_split eq:fiver_visc

where $\lambda(i^a)$ is the set of triangles that can be build from the active nodes around node i and $\lambda(i) \setminus \lambda(i^a)$ are all the triangles where at least one node is inactive(ghost).

The only difficulty now becomes the evaluation of the last term, meaning the integration over cutted elements, where at least on node is inactive, and thus does not

have a fluid state solution.
We did not have that problem for the inviscid term of equation (45), since a FV approximation was chosen, and therefore we could construct a flux at the interface thanks to the piston problem. No inactive node had to be considered.
For the viscous part, however, we want to keep the FE like formulations described in 2.3 to avoid re-writing large portions of the code. If a triangle is cut by the interface, one or two nodes will therefor be labeled inactive, and be denoted as ghost-nodes. Clearly, the fluid state vector at this ghost nodes is not defined. The question thus becomes how to evaluate this terms (last part in equation (202)).

The approach we take, that is also outlined in **TODO ask for reference** is to reconstruct a pseudo fluid-state at the ghost points, such that the boundary conditions at the wall are full-filled.

9.3.1 Reconstruction of the fluid-state at ghost-nodes

construction

TODO ask Farhat if this is really how it is done To explain the reconstruction at the ghost nodes, it again helps to consider Figure 10.
We also notice that the viscous flux, as defined in Equation 18, only depends on the fluid velocity and the temperature. Reconstruction of the velocity at the ghost node is straight forward. Assuming a linear interpolation inside the elements, the velocity at node i is reconstructed such, that the stick condition is fulfilled at the interface.
As far as the reconstruction of the temperature is concerned, two different cases have to be considered: adiabatic walls and isothermal walls.
For an adiabatic wall, the temperature gradient at the wall is zeros, which can be achieved by setting the temperature of the ghost note equal to that of node i. An isothermal wall boundary condition enforces a certain, constant temperature at the wall. Similar to the velocity boundary condition, this can be achieved by finding an appropriate ghost point value such that the condition is enforces.

The issue is depicted in detail in figure 12. The figure also reveals that there is no unique solution for the ghost point value. Multiple active node connect to the ghost node and the above described relations can be formulated for every one of them. FIVER therefore solves a least square system to chose an appropriate value.
The elegance of this approach is that once the ghost point state have been found, the evaluation of the viscous contribution can be done with the standard code routines. No adaption is required to make them work in the embedded framework.

9.4 Evaluation of forces

e_evaluation

Another issue that arises for embedded simulations is the appropraite evaluation of the forces on an embedded interface. This is especially important if the embedded framework is used in an FSI setup to get an approriate resultant on the structure. Sensitivity Analysis is another application, where approriate evaluation of Lift and

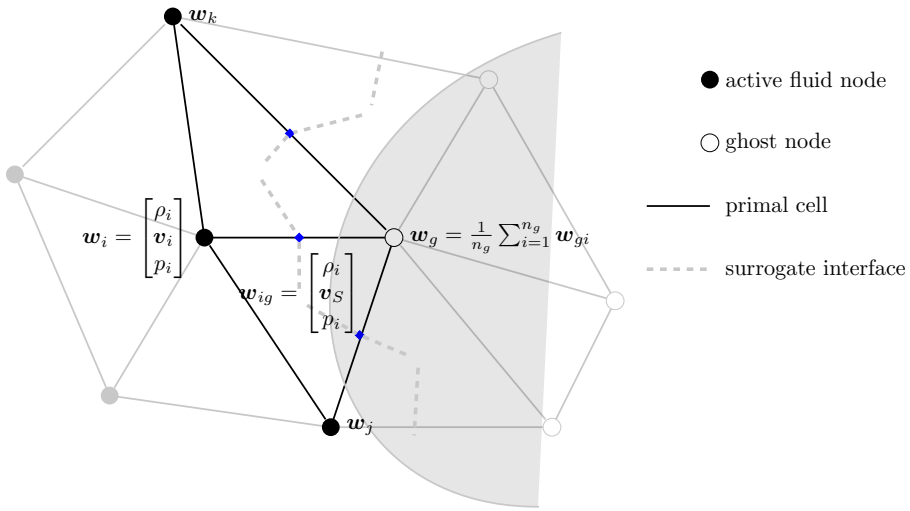


Figure 12: Illustration of the ghost point population process. States at nodes i, j, k are known from computation. States at mid-edges kg, ig, jg are the states we want to impose at the interface. The pressure and density are constantly extrapolated from the fluid states, whereas the velocity is set to the structure velocity at the interface, A linear extrapolation is then performes form the active states over the stated at the intersection points to obtain the interpolated ghost states. Since the interpolated ghost state values are typically ambiguos, due to multiple edges connecting to one ghost node, a least-square algorithm or a simple averaging can be performed to obtain a fianl value for the ghost state $\boldsymbol{w}_g$.

ghost_population

Drag integrals are crucial.
This problem as been analyzed in **REF REF** MOst importantly, the qeustion becomes whether to evaluate the pressure integral **REF** over the ausivilary control volume interface **Ref** or over the so called reconstructed interface, which can be obtained by creating ausiliary surfaces inside the intersected elements through the knowledge ablout intersection points at the edges.

terface_tracking

9.5 Level-set method

The issue of interface tracking is extensively discussed in **CITEWangGretarsson2012** and TODO.

Since the interface in an embedded stimulation typically moves, an appropriate interface tracking approach is required.
For FIVER, the popular level set method [16] was chosen.
The level set approach is characterized by the following equation:

$$\frac{\phi}{t} + \boldsymbol{v} \cdot \nabla\phi = 0 \tag{203}$$

eq:level_set

eq:level_set

where ϕ is a function designed to track the material interface. Particularly, ϕ is initialized such that that the interface is characterized by $\phi = 0$. The interface

can then be tracked via a simple time integration of equation (203). Details of this approach, which was developed for computer visualization purposes, can be found in [16].

TODO

10 Verification

sec:verification

After chapter 1 described the basic equation of fluid mechanics, discretization of this equations with FV and FE schemes in section 2, and an introduction to SA, the expressions for the actual derivatives have been derived in section 6. This has been done for both the body-fitted and the embedded 9 framework of AERO-F.
This section is denoted to a thorough validation of the obtained derivatives. As described in 6, AERO-F handles both sensitivities with respect to far-field variables(e.g. angle of attack) as well as shape sensitivities. Both types require to be handled fundamentally different. While the first type leads to derivatives mainly in the far field boundary conditions, the latter one interferes with the wall treatment. More importantly, as shown in sections 7.1 and 7.3, in the embedded case, doing shape sensitivities leads to additional derivatives with regards to the intersector.
For all those reasons, the verification provided in this chapter deals with both M_∞-sensitivity as well as shape-sensitivity. More over we will provide plots for both the body-fitted and the embedded case.

10.1 Verification approach

ication_approach

The question of how to verify the obtained sensitivities is anything but straightforward.
The first question is: What can be verified anyway?
If we have a look at equation (110), one can see, that there are to main derivatives: $\frac{\partial \mathbf{R}}{\partial \mathbf{w}}$ and $\frac{\partial \mathbf{R}}{\partial s}$. The derivative with respect to the mesh position $\frac{\partial \mathbf{R}}{\partial \mathbf{x}}$ is not validated here, since it is obtained from SDESIGN and taken for granted.
The basic idea that we exploit for verification is to compute a reference solution for the sensitivities by a FD of two steady-state simulation. The advantage is, that the sensitivity-module does not have to be used at all to obtain those references, thus it is ensured that potential bugs are not carried over.
However, for both of the above quantities, computing a finite difference solution is cumbersome. For the Jacobian $\frac{\partial \mathbf{R}}{\partial \mathbf{w}}$ a simple forward difference would look as follows:

$$\left.\frac{\partial \mathbf{R}_i}{\partial \mathbf{w}_j}\right|_{\boldsymbol{w}_0} = \frac{\mathbf{R}_i(\mathbf{w}_0 + \epsilon \boldsymbol{e}_j) - \mathbf{R}_i(\mathbf{w}_0 - \epsilon \boldsymbol{e}_j)}{2\epsilon} \tag{204}$$

which would entail an $\mathcal{O}(N)$ number of residual evaluations.
As far as $\frac{\partial \mathbf{R}}{\partial s}$ goes, a FD verification would be feasible:

$$\begin{aligned} \frac{\partial \mathbf{R}_i}{\partial M_\infty} &= \frac{\mathbf{R}(\mathbf{w}_i^+) - \mathbf{R}(\mathbf{w}_i^-)}{2\epsilon} \\ \mathbf{w}_i^\pm &= \begin{cases} \mathbf{w}_i\big|_0 \text{ for internal nodes} \\ \mathbf{w}_i\big|_{\boldsymbol{w}_i(M_i = M_0 + \epsilon)} \text{ for boundary nodes} \end{cases} \end{aligned} \tag{205}$$

which could be done cheaply, and is in fact done in AERO-F if the parameter `SensitivityAnalysis.SensitivityComputation` is set to `FiniteDifference` however, it requires the fluid-state to be updated according to the Mach-number change at the fluid boundary and thus introduced the potential for new bugs.
Instead, the first quantity to be validated in this thesis is the solution of the linear system described in equation (110): $\frac{\partial \mathbf{w}}{\partial s}$.
this can be done easily by performing a simple FD on the solution of two steady-state simulations;

$$\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s} = \frac{\mathbf{w}(s+\epsilon) - \mathbf{w}(s-\epsilon)}{2\epsilon} \tag{206}$$

which involves no extra coding and can be done with a standard steady state solver.
The validation of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ is provided in figures [13,14,15] for the body fitted framework and figures [16,17,18] for the embedded.
Additionally, since the optimization will run on lift and drag, we also validate them with steady state results.

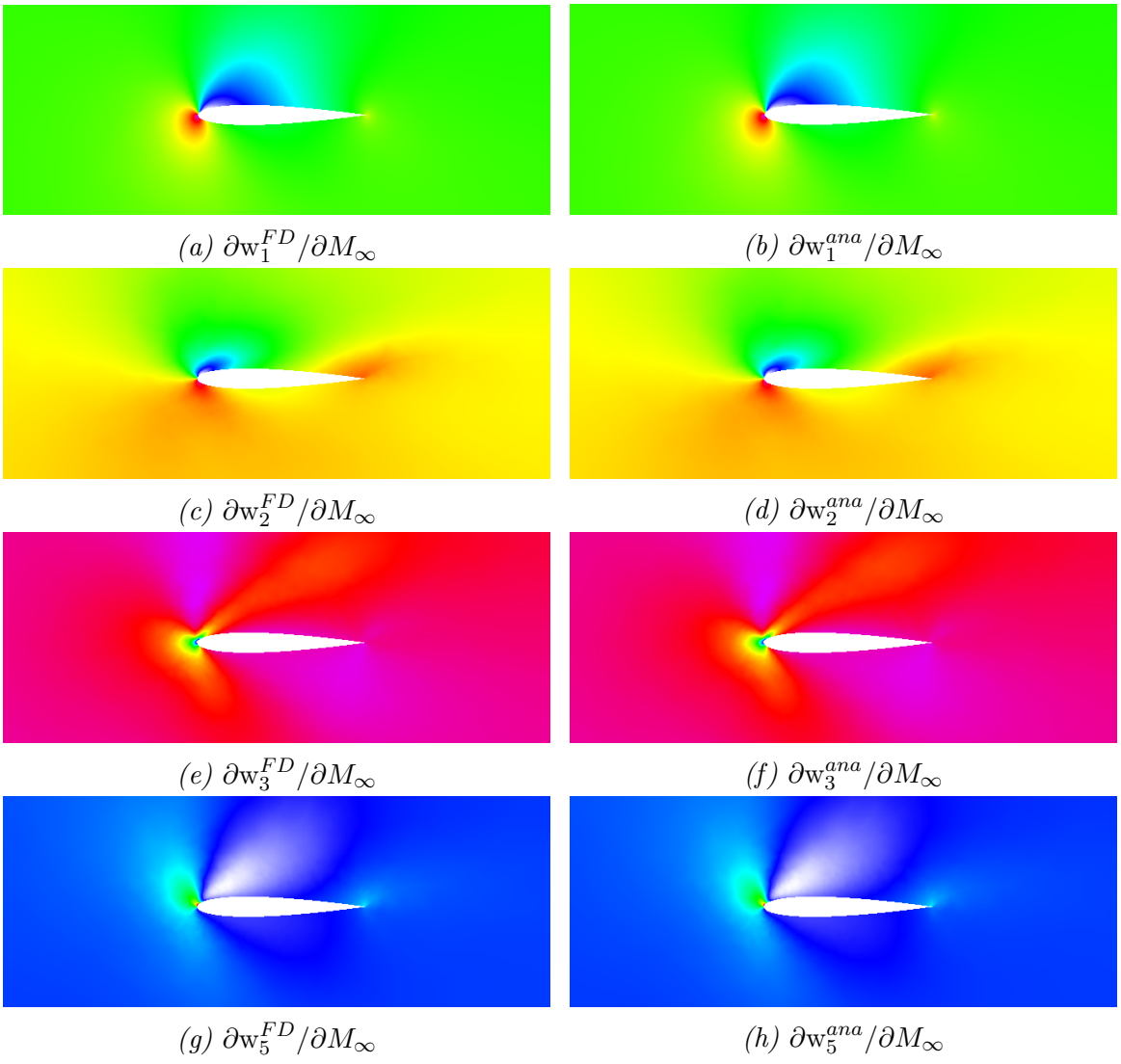
$$\frac{\mathrm{d}q}{\mathrm{d}s} = \frac{q(s+\epsilon) - q(s-\epsilon)}{2\epsilon} \tag{207}$$

The validation of integrated forces is provided in figures [29,30,31] for body-fitted computations and in figures [32,33,34] for embedded.

10.2 Verification of Mach-sensitivity

For the body-fitted case, the verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ is provided in figure 13 for Euler equations, figure 14 for full NSE and in figure 15 for NSE with an additional Spallart-Almares turbulence model. Additionally, a side by side comparison of the analytic results between all three equation types is also provided in figure 19 in order to ensure, that the additional viscous and turbulent terms actually account for a visible difference.
For the embedded case, we provide the verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ in figure 16 for Euler equation, figure 17 for full NSE and in figure 18 for NSE equations with an additional Spallart-Almares turbulence model. Again, a side by side comparison of the analytic results between all three equation types is also provided in figure 20 in order to ensure that the additional viscous and turbulent terms actually account for a visible difference.

Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ for Euler equations and a body-fitted framework



(a) $\partial \mathrm{w}_1^{FD}/\partial M_\infty$ *(b)* $\partial \mathrm{w}_1^{ana}/\partial M_\infty$

(c) $\partial \mathrm{w}_2^{FD}/\partial M_\infty$ *(d)* $\partial \mathrm{w}_2^{ana}/\partial M_\infty$

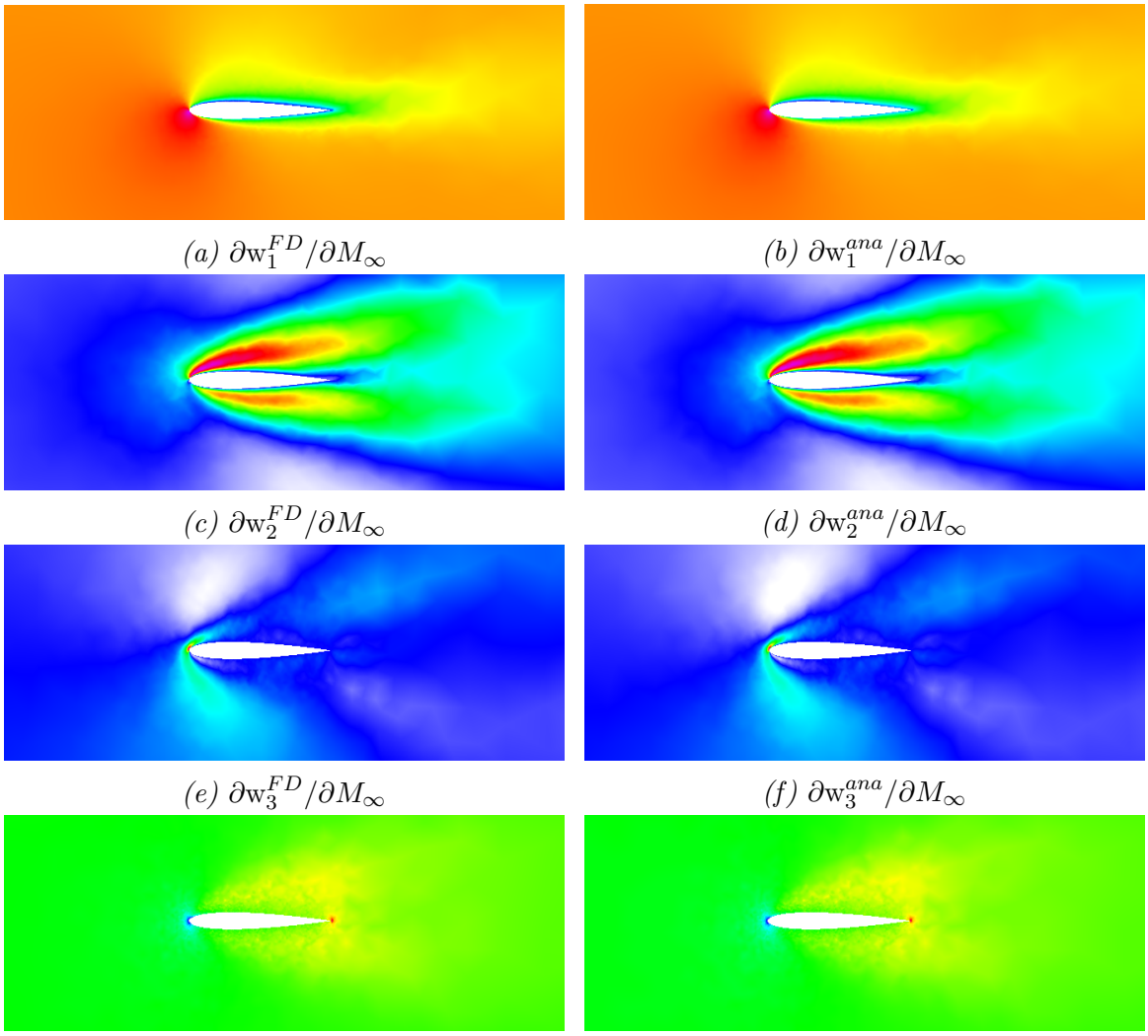
(e) $\partial \mathrm{w}_3^{FD}/\partial M_\infty$ *(f)* $\partial \mathrm{w}_3^{ana}/\partial M_\infty$

(g) $\partial \mathrm{w}_5^{FD}/\partial M_\infty$ *(h)* $\partial \mathrm{w}_5^{ana}/\partial M_\infty$

Figure 13: Verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ *for Euler equations in the body-fitted framework. The FD reference solution as described in section 10.1 is provided in the left column, the newly implemented analytic derivatives are visualized in the right column. FD and analytic solution are plotted using the same color-scheme.*

`_dwdma_ale_Euler`

Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ for Laminar equations and a body-fitted framework



(a) $\partial \mathrm{w}_1^{FD}/\partial M_\infty$ *(b)* $\partial \mathrm{w}_1^{ana}/\partial M_\infty$

(c) $\partial \mathrm{w}_2^{FD}/\partial M_\infty$ *(d)* $\partial \mathrm{w}_2^{ana}/\partial M_\infty$

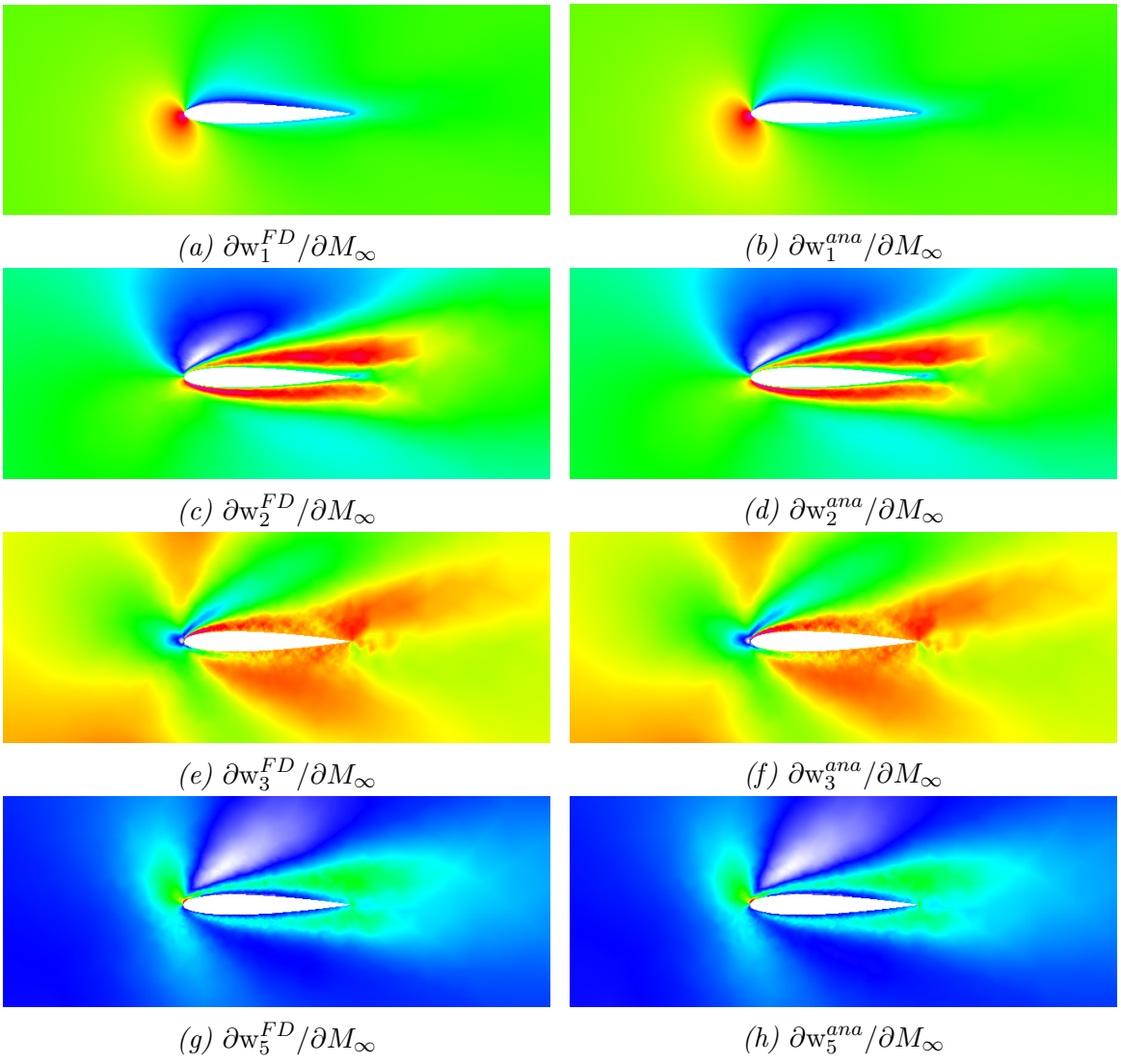
(e) $\partial \mathrm{w}_3^{FD}/\partial M_\infty$ *(f)* $\partial \mathrm{w}_3^{ana}/\partial M_\infty$

(g) $\partial \mathrm{w}_5^{FD}/\partial M_\infty$ *(h)* $\partial \mathrm{w}_5^{ana}/\partial M_\infty$

Figure 14: Verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ *for Laminar equations in the body-fitted framework. The FD reference solution as described in section 10.1 is provided in the left column, the newly implemented analytic derivatives are visualized in the right column. FD and analytic solution are plotted using the same color-scheme.*

_ale_Laminar

Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ for RANS equations and a body-fitted framework



(a) $\partial \mathrm{w}_1^{FD}/\partial M_\infty$ *(b)* $\partial \mathrm{w}_1^{ana}/\partial M_\infty$

(c) $\partial \mathrm{w}_2^{FD}/\partial M_\infty$ *(d)* $\partial \mathrm{w}_2^{ana}/\partial M_\infty$

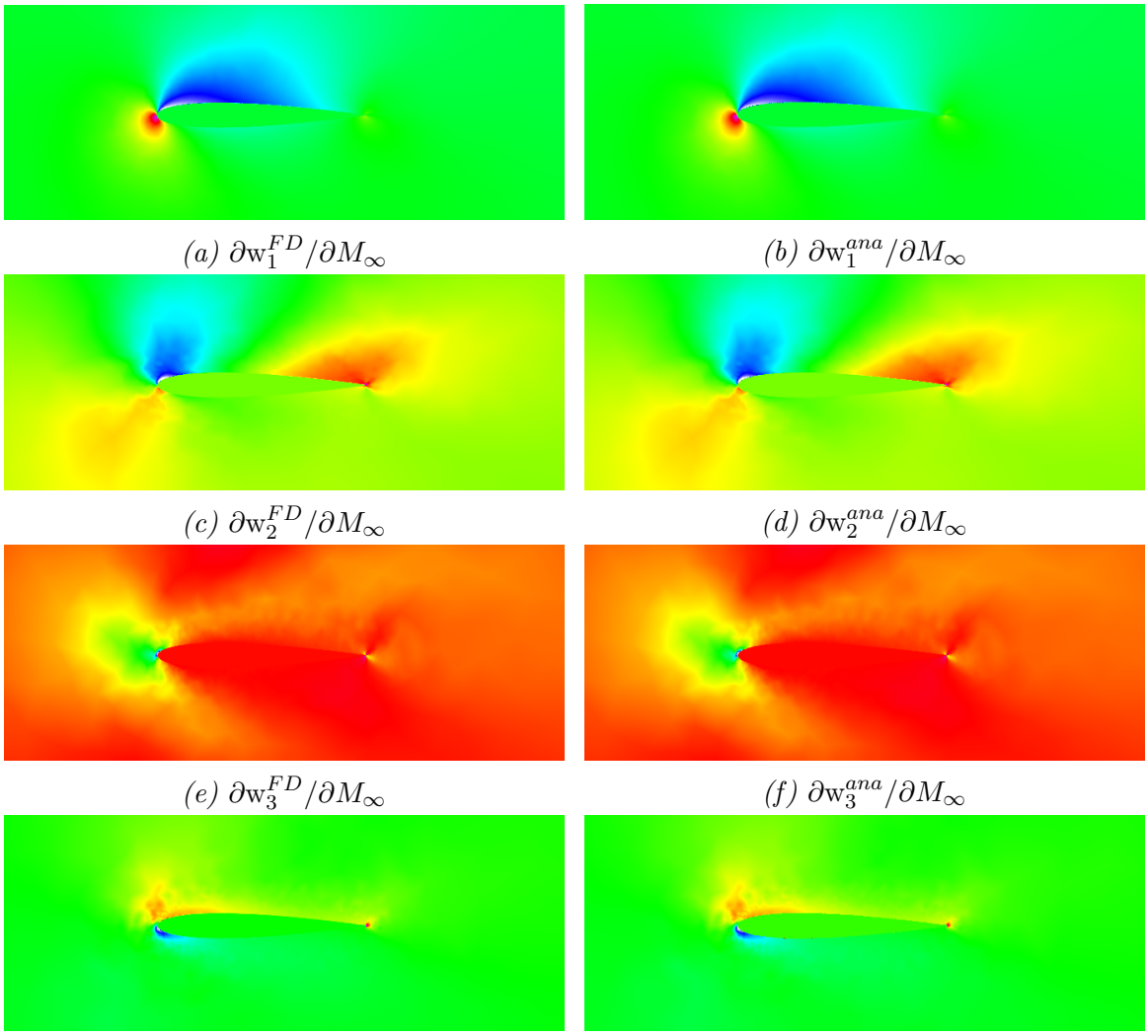
(e) $\partial \mathrm{w}_3^{FD}/\partial M_\infty$ *(f)* $\partial \mathrm{w}_3^{ana}/\partial M_\infty$

(g) $\partial \mathrm{w}_5^{FD}/\partial M_\infty$ *(h)* $\partial \mathrm{w}_5^{ana}/\partial M_\infty$

Figure 15: Verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ *for RANS equations in the body-fitted framework. The FD reference solution as described in section 10.1 is provided in the left column, the newly implemented analytic derivatives are visualized in the right column. FD and analytic solution are plotted using the same color-scheme.*

`n_dwdma_ale_RANS`

Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ for Euler equations and a embedded framework



(a) $\partial \mathrm{w}_1^{FD}/\partial M_\infty$ *(b)* $\partial \mathrm{w}_1^{ana}/\partial M_\infty$

(c) $\partial \mathrm{w}_2^{FD}/\partial M_\infty$ *(d)* $\partial \mathrm{w}_2^{ana}/\partial M_\infty$

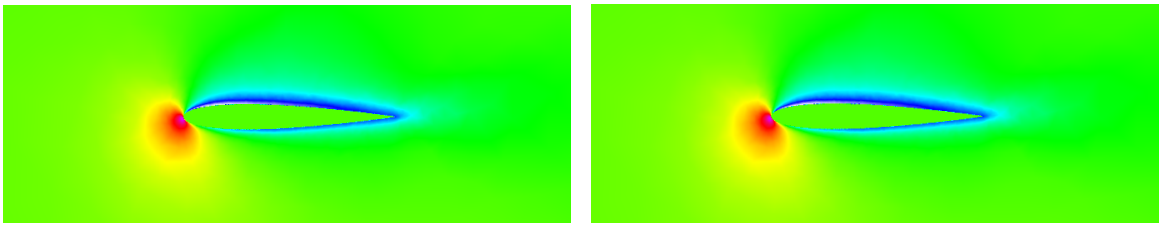
(e) $\partial \mathrm{w}_3^{FD}/\partial M_\infty$ *(f)* $\partial \mathrm{w}_3^{ana}/\partial M_\infty$

(g) $\partial \mathrm{w}_5^{FD}/\partial M_\infty$ *(h)* $\partial \mathrm{w}_5^{ana}/\partial M_\infty$

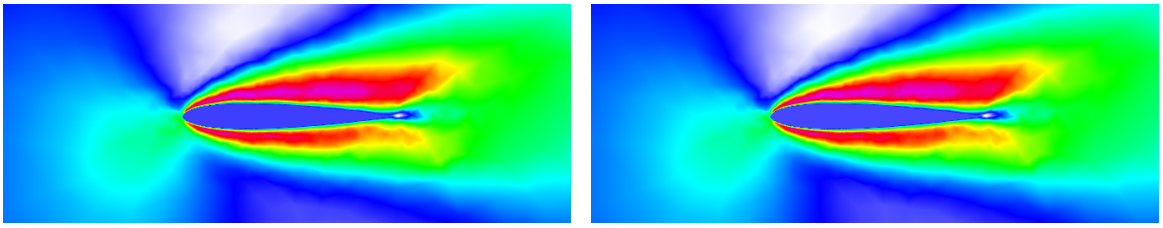
Figure 16: Verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ *for Euler equations in the embedded framework. The FD reference solution as described in section 10.1 is provided in the left column, the newly implemented analytic derivatives are visualized in the right column. FD and analytic solution are plotted using the same color-scheme.*

`ma_emb_Euler`

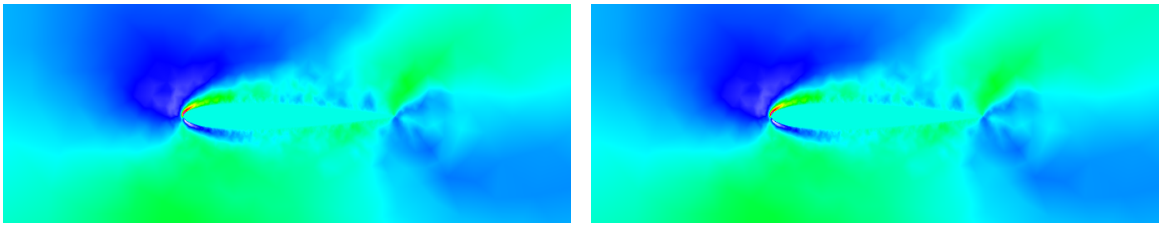
Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ for Laminar equations and a embedded framework



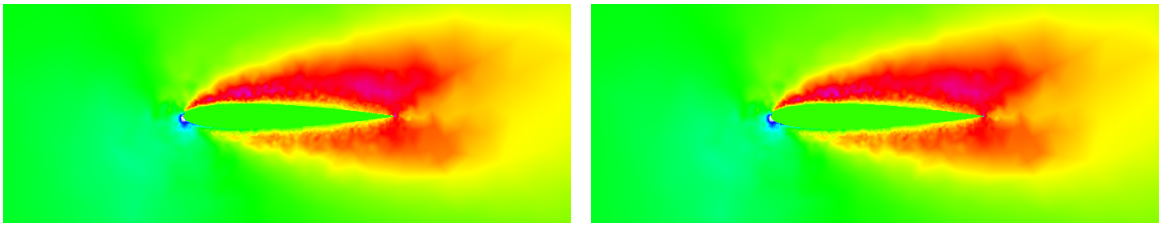
(a) $\partial \mathrm{w}_1^{FD}/\partial M_\infty$ (b) $\partial \mathrm{w}_1^{ana}/\partial M_\infty$



(c) $\partial \mathrm{w}_2^{FD}/\partial M_\infty$ (d) $\partial \mathrm{w}_2^{ana}/\partial M_\infty$



(e) $\partial \mathrm{w}_3^{FD}/\partial M_\infty$ (f) $\partial \mathrm{w}_3^{ana}/\partial M_\infty$

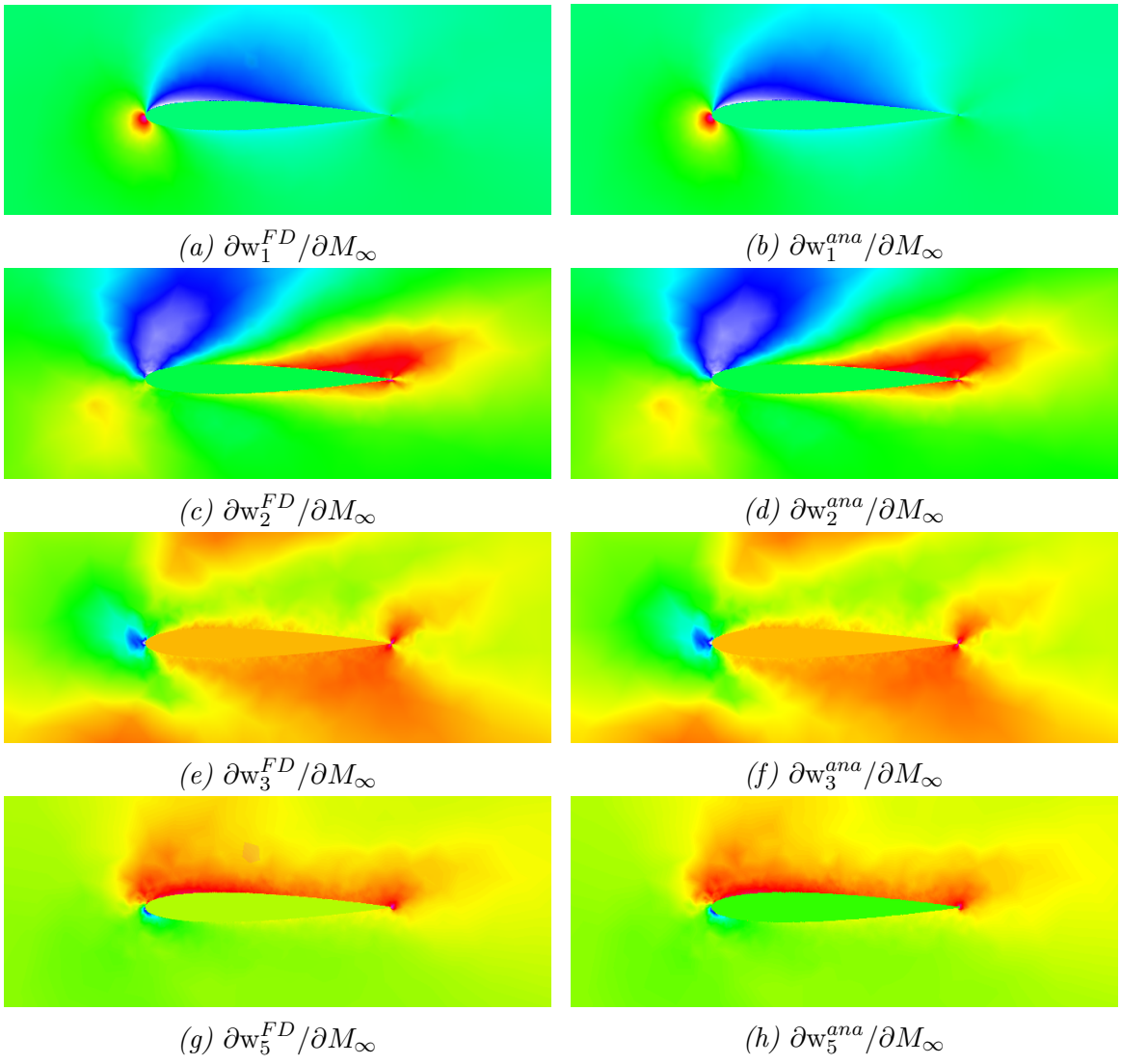


(g) $\partial \mathrm{w}_5^{FD}/\partial M_\infty$ (h) $\partial \mathrm{w}_5^{ana}/\partial M_\infty$

Figure 17: Verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ for Laminar equations in the embedded framework. The FD reference solution as described in section 10.1 is provided in the left column, the newly implemented analytic derivatives are visualized in the right column. FD and analytic solution are plotted using the same color-scheme.

`wdma_emb_Laminar`

Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ for RANS equations and a embedded framework



(a) $\partial \mathrm{w}_1^{FD}/\partial M_\infty$ *(b)* $\partial \mathrm{w}_1^{ana}/\partial M_\infty$

(c) $\partial \mathrm{w}_2^{FD}/\partial M_\infty$ *(d)* $\partial \mathrm{w}_2^{ana}/\partial M_\infty$

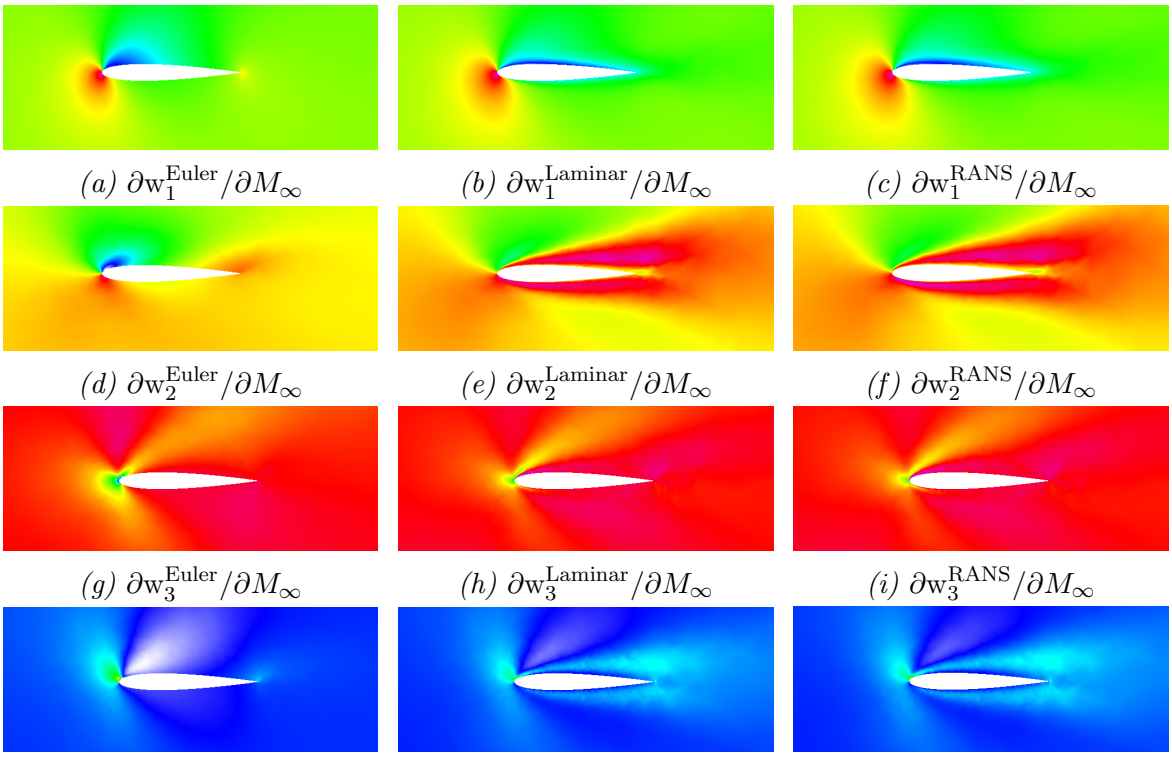
(e) $\partial \mathrm{w}_3^{FD}/\partial M_\infty$ *(f)* $\partial \mathrm{w}_3^{ana}/\partial M_\infty$

(g) $\partial \mathrm{w}_5^{FD}/\partial M_\infty$ *(h)* $\partial \mathrm{w}_5^{ana}/\partial M_\infty$

Figure 18: Verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ *for RANS equations in the embedded framework. The FD reference solution as described in section 10.1 is provided in the left column, the newly implemented analytic derivatives are visualized in the right column. FD and analytic solution are plotted using the same color-scheme.*

dma_emb_RANS

Comparison of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ in the body-fitted framework for all 3 equation types



(a) $\partial \mathrm{w}_1^{\mathrm{Euler}}/\partial M_\infty$ *(b)* $\partial \mathrm{w}_1^{\mathrm{Laminar}}/\partial M_\infty$ *(c)* $\partial \mathrm{w}_1^{\mathrm{RANS}}/\partial M_\infty$

(d) $\partial \mathrm{w}_2^{\mathrm{Euler}}/\partial M_\infty$ *(e)* $\partial \mathrm{w}_2^{\mathrm{Laminar}}/\partial M_\infty$ *(f)* $\partial \mathrm{w}_2^{\mathrm{RANS}}/\partial M_\infty$

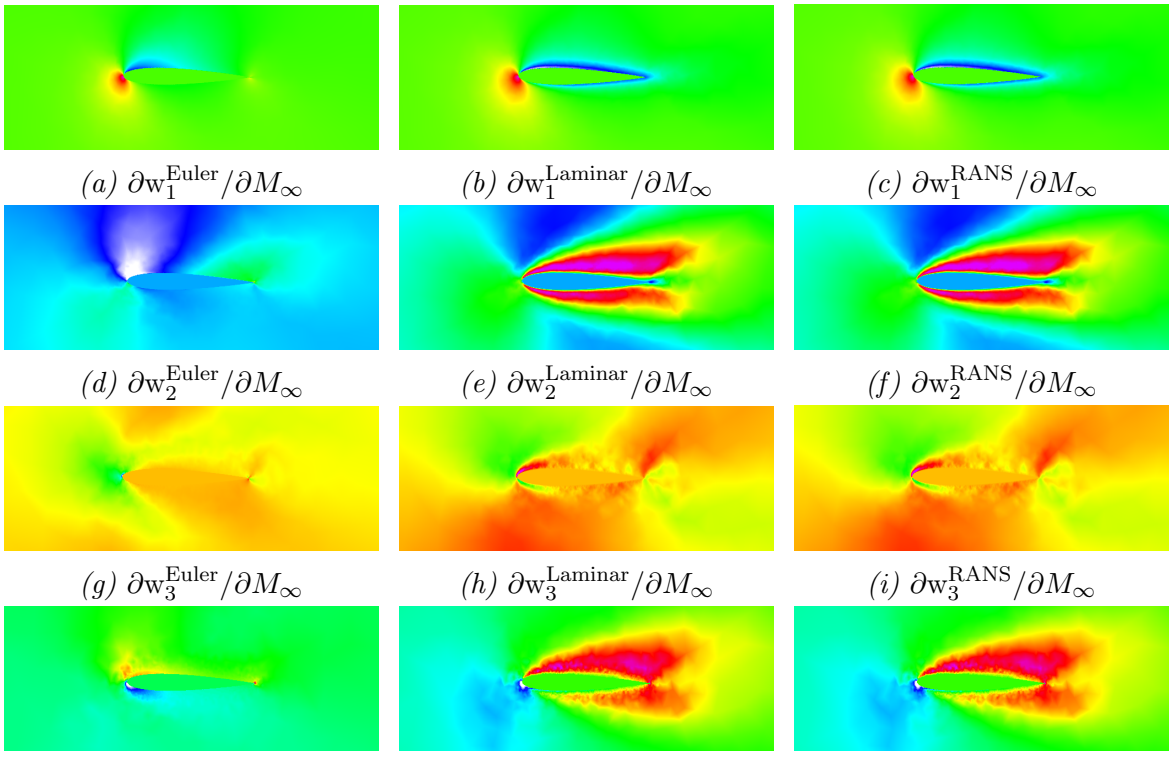
(g) $\partial \mathrm{w}_3^{\mathrm{Euler}}/\partial M_\infty$ *(h)* $\partial \mathrm{w}_3^{\mathrm{Laminar}}/\partial M_\infty$ *(i)* $\partial \mathrm{w}_3^{\mathrm{RANS}}/\partial M_\infty$

(j) $\partial \mathrm{w}_5^{\mathrm{Euler}}/\partial M_\infty$ *(k)* $\partial \mathrm{w}_5^{\mathrm{Laminar}}/\partial M_\infty$ *(l)* $\partial \mathrm{w}_5^{\mathrm{RANS}}/\partial M_\infty$

Figure 19: Comparison of analytic $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ *for Euler equations(left column), laminar NSE equations (center column) and NSE equations with turbulence models (right column) in the body-fitted framework.*

a_ale_comparison

Comparison of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ in the embedded framework for all 3 equation types



(a) $\partial \mathrm{w}_1^{\mathrm{Euler}}/\partial M_\infty$ *(b)* $\partial \mathrm{w}_1^{\mathrm{Laminar}}/\partial M_\infty$ *(c)* $\partial \mathrm{w}_1^{\mathrm{RANS}}/\partial M_\infty$

(d) $\partial \mathrm{w}_2^{\mathrm{Euler}}/\partial M_\infty$ *(e)* $\partial \mathrm{w}_2^{\mathrm{Laminar}}/\partial M_\infty$ *(f)* $\partial \mathrm{w}_2^{\mathrm{RANS}}/\partial M_\infty$

(g) $\partial \mathrm{w}_3^{\mathrm{Euler}}/\partial M_\infty$ *(h)* $\partial \mathrm{w}_3^{\mathrm{Laminar}}/\partial M_\infty$ *(i)* $\partial \mathrm{w}_3^{\mathrm{RANS}}/\partial M_\infty$

(j) $\partial \mathrm{w}_5^{\mathrm{Euler}}/\partial M_\infty$ *(k)* $\partial \mathrm{w}_5^{\mathrm{Laminar}}/\partial M_\infty$ *(l)* $\partial \mathrm{w}_5^{\mathrm{RANS}}/\partial M_\infty$

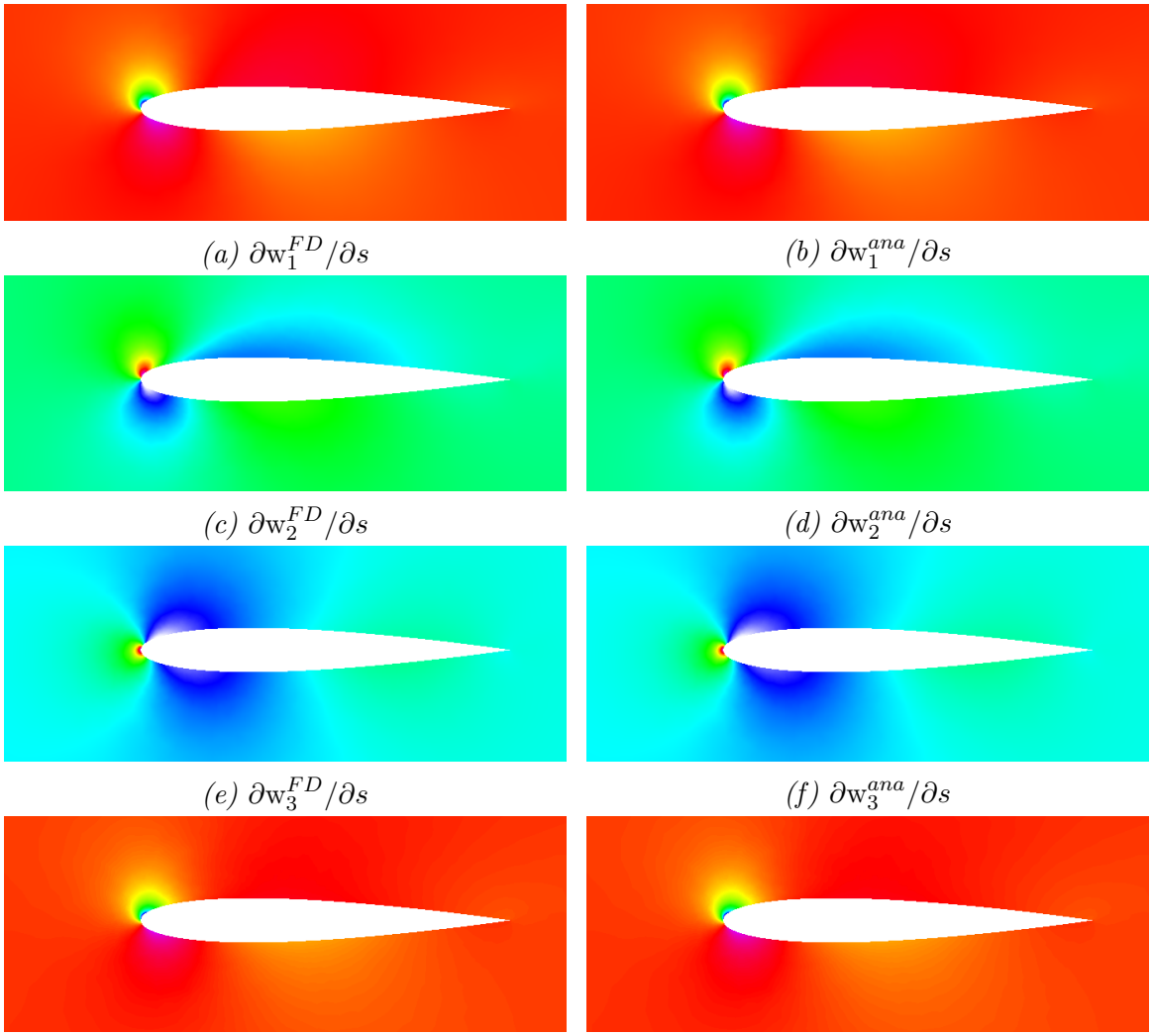
Figure 20: Comparison of analytic $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ for Euler equations(left column), laminar NSE equations (center column) and NSE equations with turbulence models (right column) in the embedded framework.

b_comparison

10.3 Verification shape sensitivity

The verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ for ALE simulations is provided in figure 21 for Euler equation, figure 22 for full NSE and in figure 23 for NSE equations with an additional Spallart-Almares turbulence model. Additionally, a side by side comparison of the analytic results between all three equation types is also provided in figure 27 in order to ensure that the additional viscous and turbulent terms actually account for a visible difference.
For the embedded case, we provide the verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ in figure 24 for Euler equation, figure 25 for full NSE and in figure 26 for NSE equations with an additional Spallart-Almares turbulence model. Again, a side by side comparison of the analytic results between all three equation types is also provided in figure 28 in order to ensure that the additional viscous and turbulent terms actually account for a visible difference.

Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ for Euler equations and a body-fitted framework



(a) $\partial \mathrm{w}_1^{FD}/\partial s$ (b) $\partial \mathrm{w}_1^{ana}/\partial s$

(c) $\partial \mathrm{w}_2^{FD}/\partial s$ (d) $\partial \mathrm{w}_2^{ana}/\partial s$

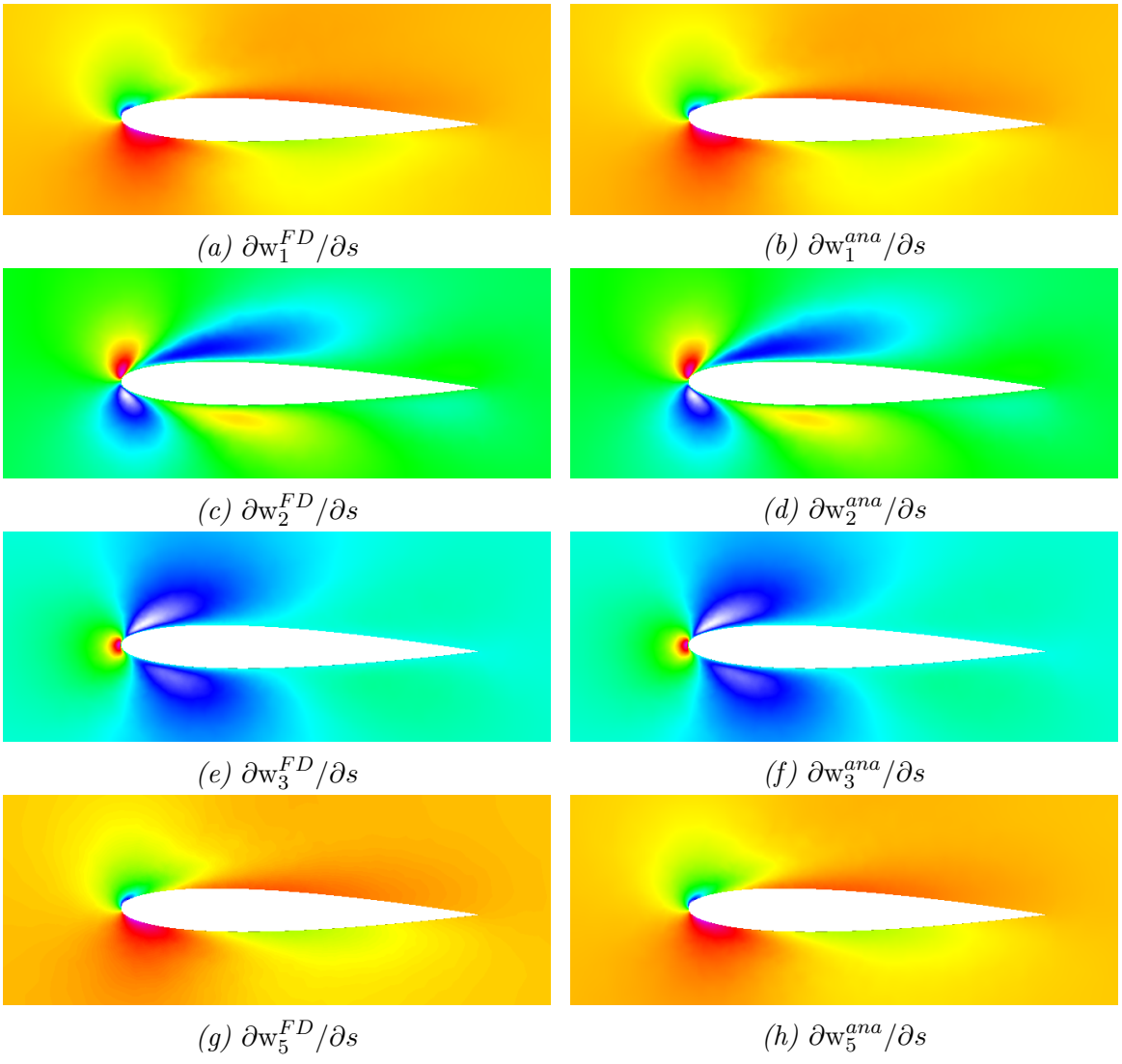
(e) $\partial \mathrm{w}_3^{FD}/\partial s$ (f) $\partial \mathrm{w}_3^{ana}/\partial s$

(g) $\partial \mathrm{w}_5^{FD}/\partial s$ (h) $\partial \mathrm{w}_5^{ana}/\partial s$

Figure 21: Verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ for Euler equations in the body-fitted framework. The FD reference solution as described in section 10.1 is provided in the left column, the newly implemented analytic derivatives are visualized in the right column. FD and analytic solution are plotted using the same color-scheme.

`ds_ale_Euler`

Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ for Laminar equations and a body-fitted framework



(a) $\partial \mathrm{w}_1^{FD}/\partial s$ *(b)* $\partial \mathrm{w}_1^{ana}/\partial s$

(c) $\partial \mathrm{w}_2^{FD}/\partial s$ *(d)* $\partial \mathrm{w}_2^{ana}/\partial s$

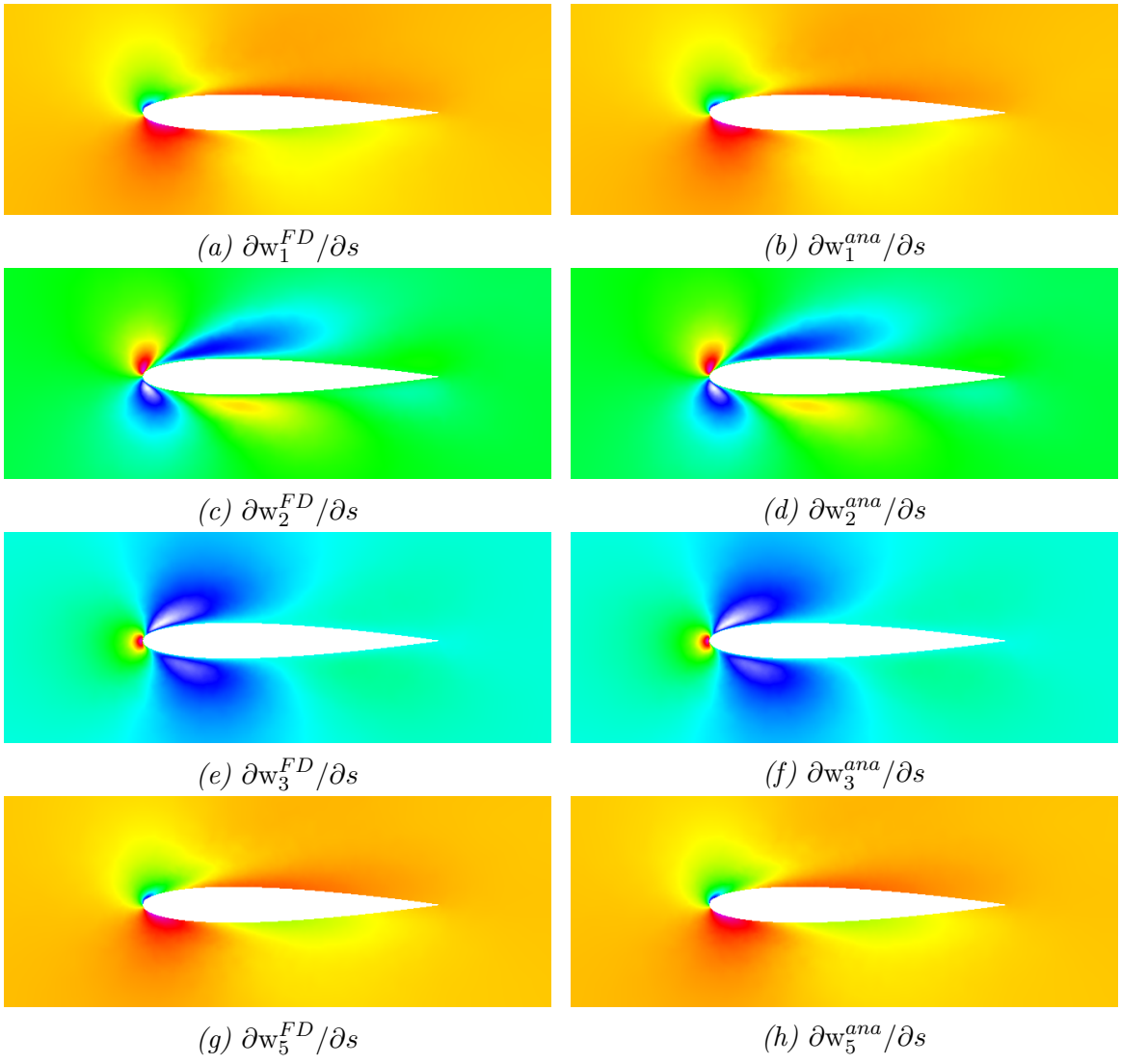
(e) $\partial \mathrm{w}_3^{FD}/\partial s$ *(f)* $\partial \mathrm{w}_3^{ana}/\partial s$

(g) $\partial \mathrm{w}_5^{FD}/\partial s$ *(h)* $\partial \mathrm{w}_5^{ana}/\partial s$

Figure 22: Verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ *for Laminar equations in the body-fitted framework. The FD reference solution as described in section 10.1 is provided in the left column, the newly implemented analytic derivatives are visualized in the right column. FD and analytic solution are plotted using the same color-scheme.*

`dwds_ale_Laminar`

Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ for RANS equations and a body-fitted framework



(a) $\partial \mathrm{w}_1^{FD}/\partial s$ *(b)* $\partial \mathrm{w}_1^{ana}/\partial s$

(c) $\partial \mathrm{w}_2^{FD}/\partial s$ *(d)* $\partial \mathrm{w}_2^{ana}/\partial s$

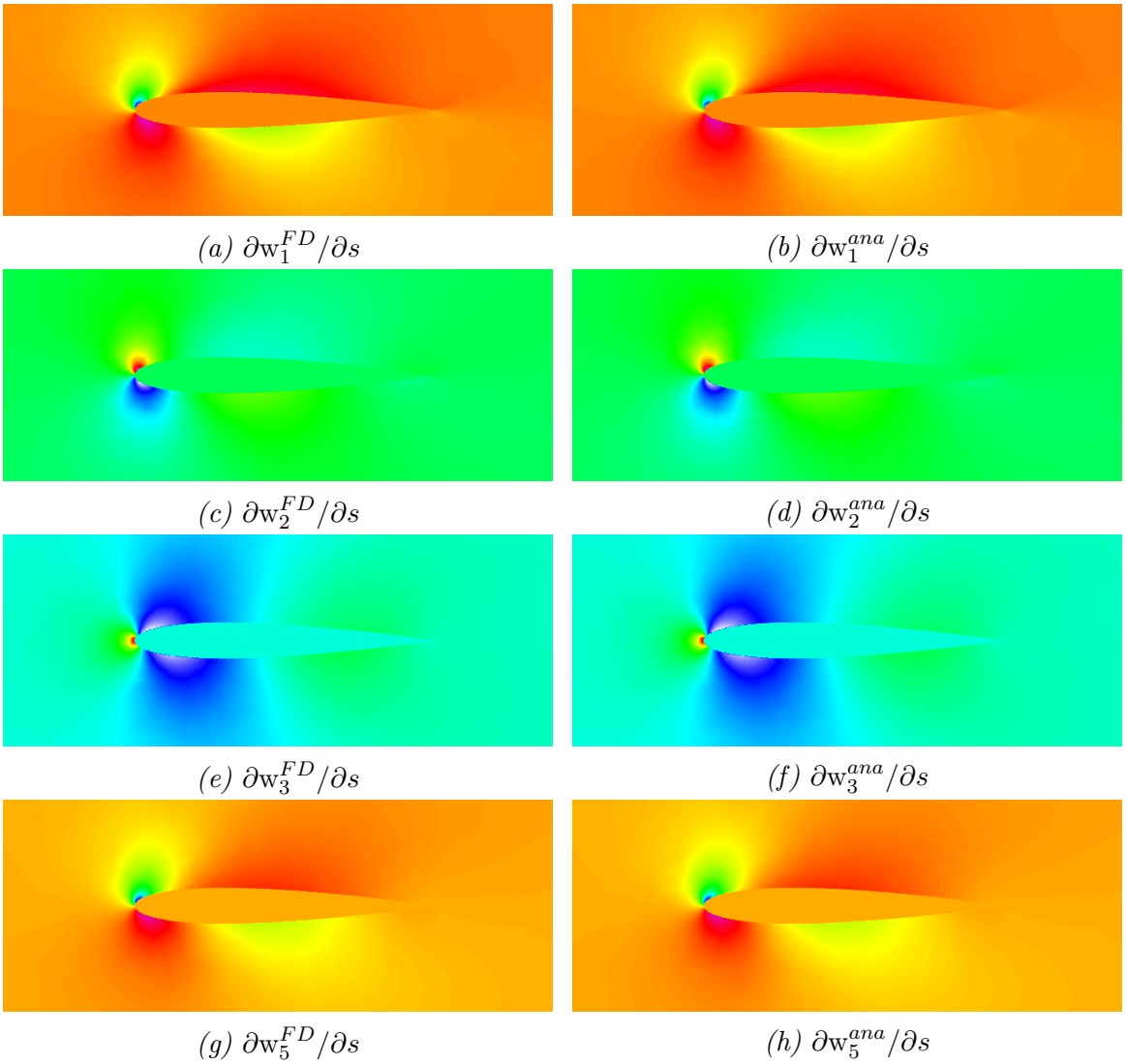
(e) $\partial \mathrm{w}_3^{FD}/\partial s$ *(f)* $\partial \mathrm{w}_3^{ana}/\partial s$

(g) $\partial \mathrm{w}_5^{FD}/\partial s$ *(h)* $\partial \mathrm{w}_5^{ana}/\partial s$

Figure 23: Verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ *for RANS equations in the body-fitted framework. The FD reference solution as described in section 10.1 is provided in the left column, the newly implemented analytic derivatives are visualized in the right column. FD and analytic solution are plotted using the same color-scheme.*

wds_ale_RANS

Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ for Euler equations and a embedded framework



(a) $\partial \mathrm{w}_1^{FD}/\partial s$ *(b)* $\partial \mathrm{w}_1^{ana}/\partial s$

(c) $\partial \mathrm{w}_2^{FD}/\partial s$ *(d)* $\partial \mathrm{w}_2^{ana}/\partial s$

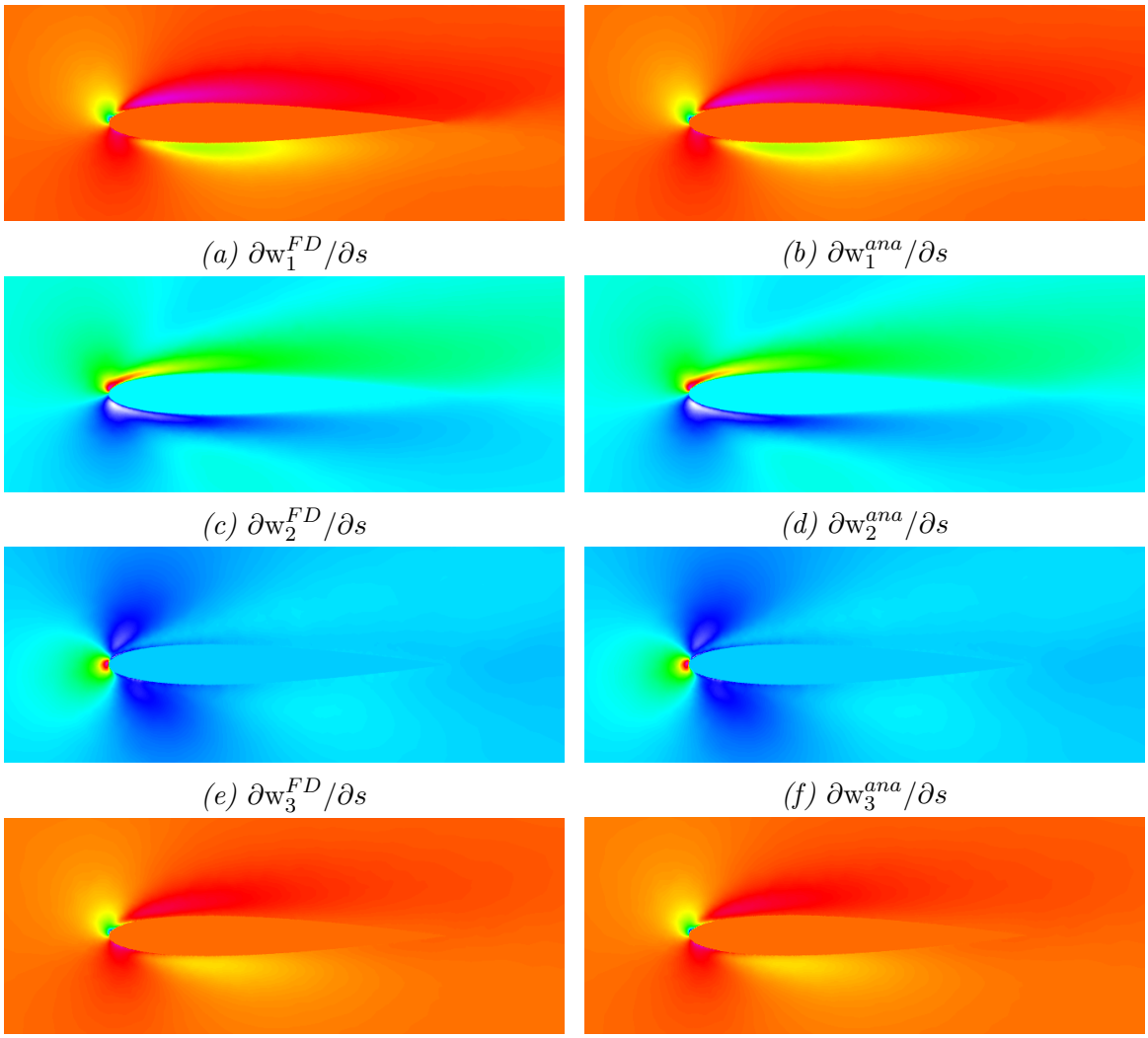
(e) $\partial \mathrm{w}_3^{FD}/\partial s$ *(f)* $\partial \mathrm{w}_3^{ana}/\partial s$

(g) $\partial \mathrm{w}_5^{FD}/\partial s$ *(h)* $\partial \mathrm{w}_5^{ana}/\partial s$

Figure 24: Verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ *for Euler equations in the embedded framework. The FD reference solution as described in section 10.1 is provided in the left column, the newly implemented analytic derivatives are visualized in the right column. FD and analytic solution are plotted using the same color-scheme.*

n_dwds_emb_Euler

Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ for Laminar equations and a embedded framework



(a) $\partial \mathrm{w}_1^{FD}/\partial s$ *(b)* $\partial \mathrm{w}_1^{ana}/\partial s$

(c) $\partial \mathrm{w}_2^{FD}/\partial s$ *(d)* $\partial \mathrm{w}_2^{ana}/\partial s$

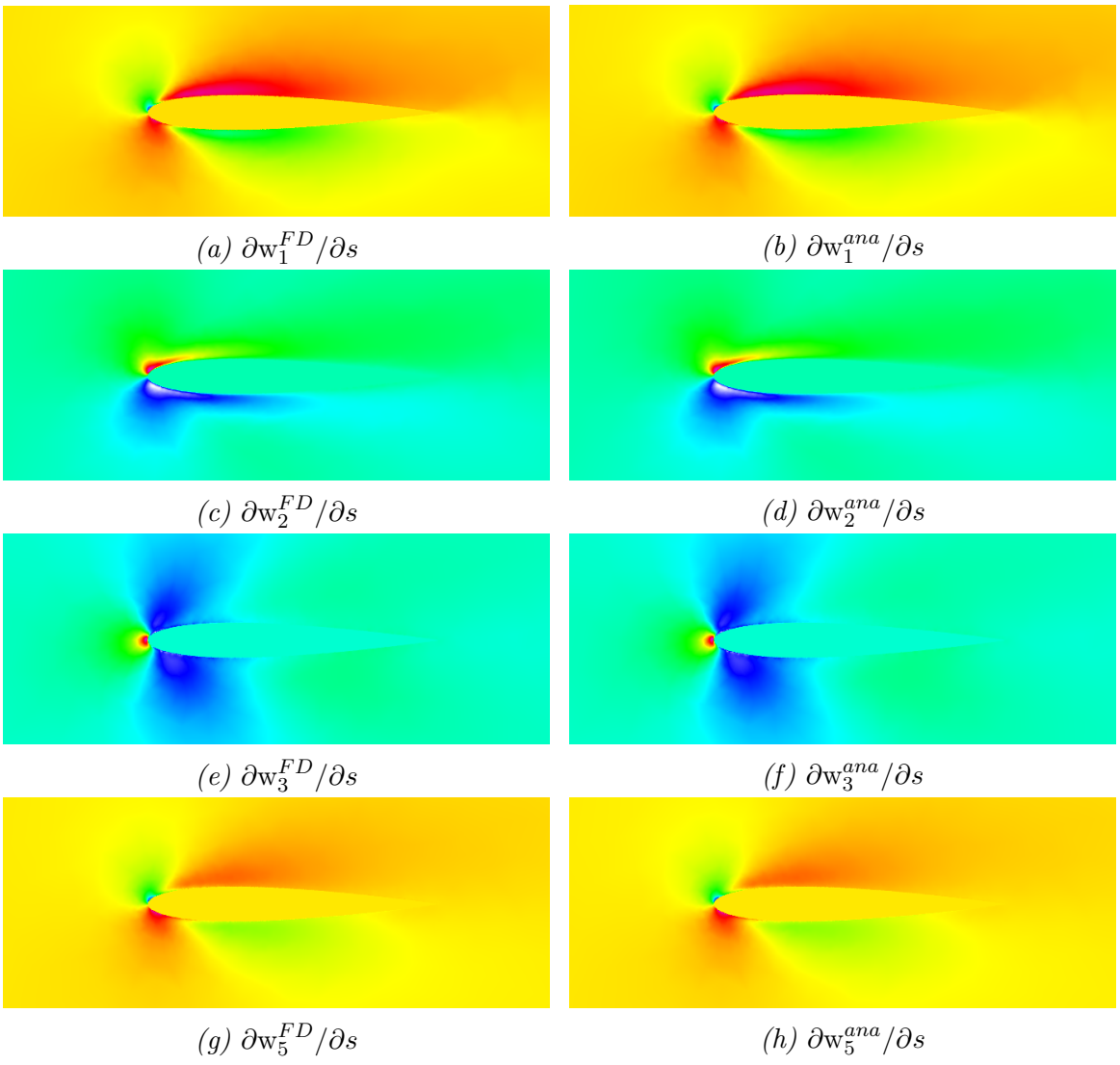
(e) $\partial \mathrm{w}_3^{FD}/\partial s$ *(f)* $\partial \mathrm{w}_3^{ana}/\partial s$

(g) $\partial \mathrm{w}_5^{FD}/\partial s$ *(h)* $\partial \mathrm{w}_5^{ana}/\partial s$

Figure 25: Verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ for Laminar equations in the embedded framework. The FD reference solution as described in section 10.1 is provided in the left column, the newly implemented analytic derivatives are visualized in the right column. FD and analytic solution are plotted using the same color-scheme.

`_emb_Laminar`

Verification $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ for RANS equations and a embedded framework



(a) $\partial \mathrm{w}_1^{FD}/\partial s$ *(b)* $\partial \mathrm{w}_1^{ana}/\partial s$

(c) $\partial \mathrm{w}_2^{FD}/\partial s$ *(d)* $\partial \mathrm{w}_2^{ana}/\partial s$

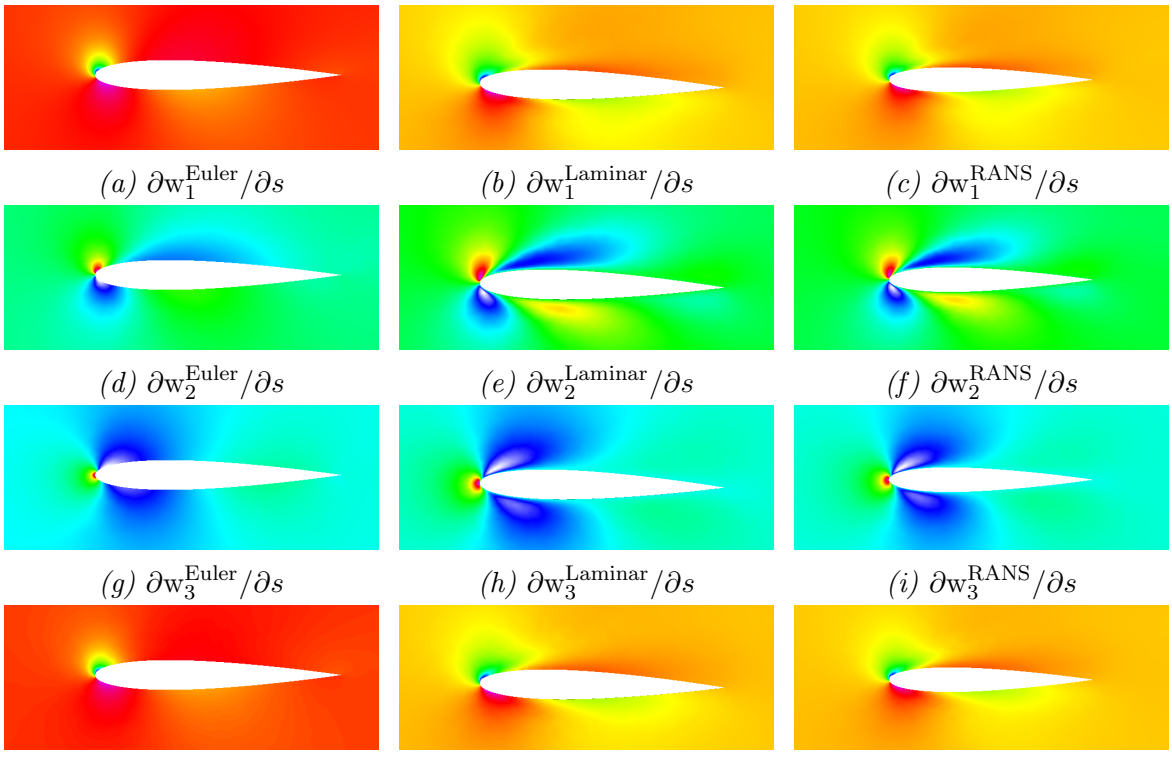
(e) $\partial \mathrm{w}_3^{FD}/\partial s$ *(f)* $\partial \mathrm{w}_3^{ana}/\partial s$

(g) $\partial \mathrm{w}_5^{FD}/\partial s$ *(h)* $\partial \mathrm{w}_5^{ana}/\partial s$

Figure 26: Verification of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ *for RANS equations in the embedded framework. The FD reference solution as described in section 10.1 is provided in the left column, the newly implemented analytic derivatives are visualized in the right column. FD and analytic solution are plotted using the same color-scheme.*

on_dwds_emb_RANS

Comparison of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ in the body-fitted framework for all 3 equation types



(a) $\partial \mathrm{w}_1^{\mathrm{Euler}}/\partial s$ *(b)* $\partial \mathrm{w}_1^{\mathrm{Laminar}}/\partial s$ *(c)* $\partial \mathrm{w}_1^{\mathrm{RANS}}/\partial s$

(d) $\partial \mathrm{w}_2^{\mathrm{Euler}}/\partial s$ *(e)* $\partial \mathrm{w}_2^{\mathrm{Laminar}}/\partial s$ *(f)* $\partial \mathrm{w}_2^{\mathrm{RANS}}/\partial s$

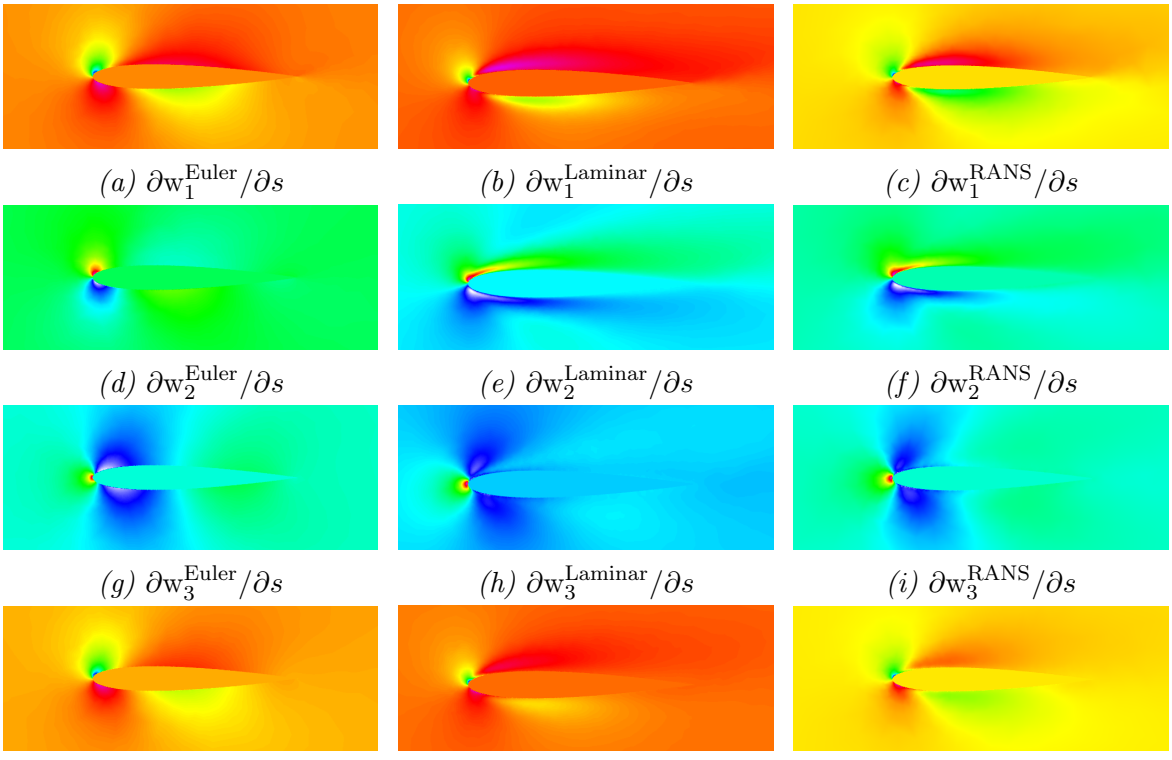
(g) $\partial \mathrm{w}_3^{\mathrm{Euler}}/\partial s$ *(h)* $\partial \mathrm{w}_3^{\mathrm{Laminar}}/\partial s$ *(i)* $\partial \mathrm{w}_3^{\mathrm{RANS}}/\partial s$

(j) $\partial \mathrm{w}_5^{\mathrm{Euler}}/\partial s$ *(k)* $\partial \mathrm{w}_5^{\mathrm{Laminar}}/\partial s$ *(l)* $\partial \mathrm{w}_5^{\mathrm{RANS}}/\partial s$

Figure 27: Comparison of analytic $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ for Euler equations(left column), laminar NSE equations (center column) and NSE equations with turbulence models (right column) in the body-fitted framework.

e_comparison

Comparison of $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s}$ in the embedded framework for all 3 equation types



(a) $\partial \mathrm{w}_1^{\mathrm{Euler}}/\partial s$ *(b)* $\partial \mathrm{w}_1^{\mathrm{Laminar}}/\partial s$ *(c)* $\partial \mathrm{w}_1^{\mathrm{RANS}}/\partial s$

(d) $\partial \mathrm{w}_2^{\mathrm{Euler}}/\partial s$ *(e)* $\partial \mathrm{w}_2^{\mathrm{Laminar}}/\partial s$ *(f)* $\partial \mathrm{w}_2^{\mathrm{RANS}}/\partial s$

(g) $\partial \mathrm{w}_3^{\mathrm{Euler}}/\partial s$ *(h)* $\partial \mathrm{w}_3^{\mathrm{Laminar}}/\partial s$ *(i)* $\partial \mathrm{w}_3^{\mathrm{RANS}}/\partial s$

(j) $\partial \mathrm{w}_5^{\mathrm{Euler}}/\partial s$ *(k)* $\partial \mathrm{w}_5^{\mathrm{Laminar}}/\partial s$ *(l)* $\partial \mathrm{w}_5^{\mathrm{RANS}}/\partial s$

Figure 28: Comparison of analytic $\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}M_\infty}$ *for Euler equations(left column), laminar NSE equations (center column) and NSE equations with turbulence models (right column) in the embedded framework.*

s_emb_comparison

10.4 Work in progress

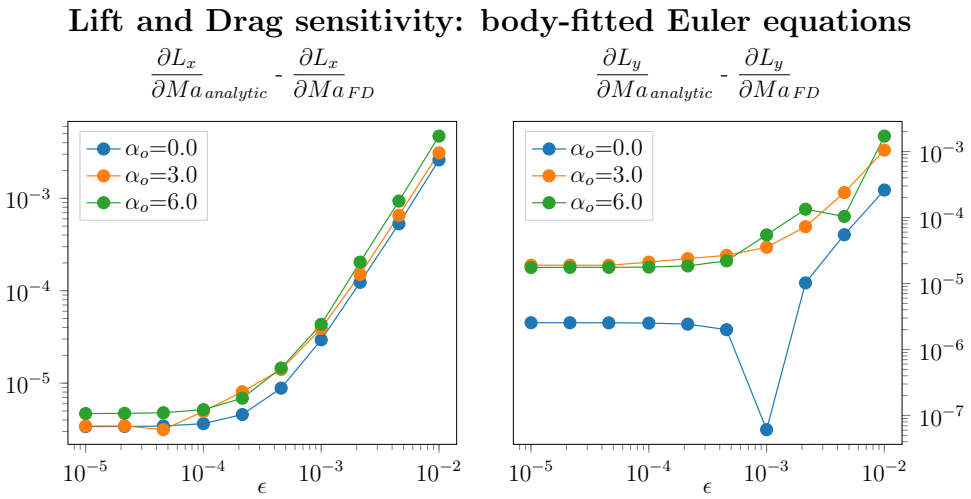


Figure 29: The graph shows the relative error between the Lift and Drag Mach-sensitivities obtained by FD of two steady states, compared to the mach-sensitivity obtained by analytic direct evaluation. L_x denotes the Drag, L_y denotes the lift. Please note that a FD Jacobian has been used here, which of course introduces an approximation error. Furthermore, the finite precision of the obtained steady states is another error source. This explains why the error levels off. For all practical optimization purposes, the obtained accuracy is more than enough.

Ma_Euler_ale

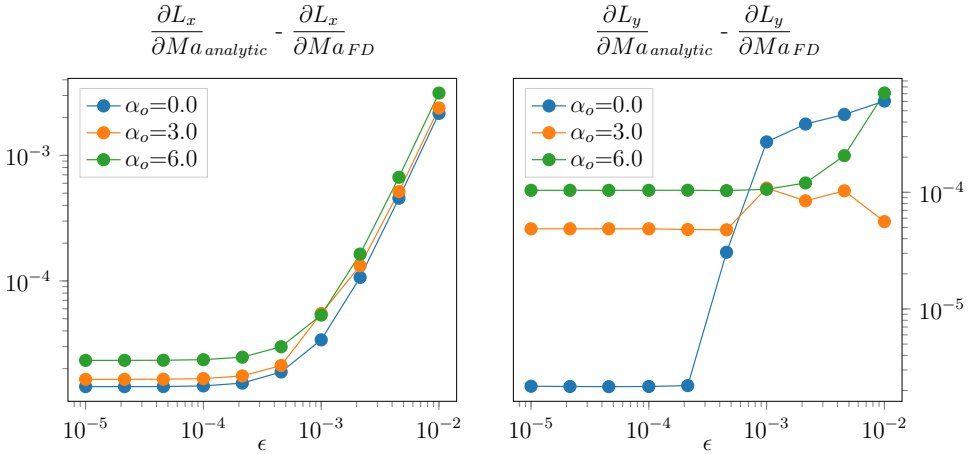


Figure 30: The graph shows the relative error between the Lift and Drag Mach-sensitivities obtained by FD of two steady states, compared to the mach-sensitivity obtained by analytic direct evaluation. L_x denotes the Drag, L_y denotes the lift. Please note that a FD Jacobian has been used here, which of course introduces an approximation error. Furthermore, the finite precision of the obtained steady states is another error source. This explains why the error levels off. For all practical optimization purposes, the obtained accuracy is more than enough.

_Laminar_ale

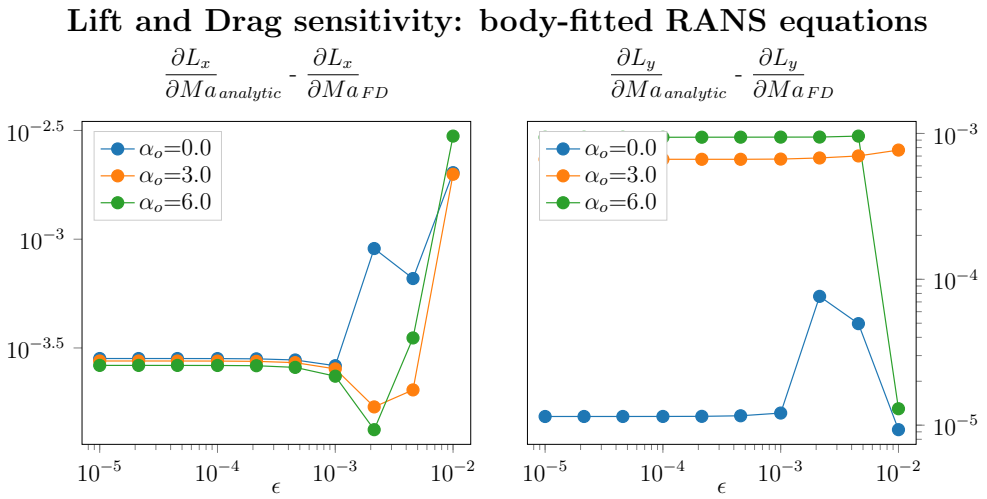


Figure 31: The graph shows the relative error between the Lift and Drag Mach-sensitivities obtained by FD of two steady states, compared to the mach-sensitivity obtained by analytic direct evaluation. L_x denotes the Drag, L_y denotes the lift. Please note that a FD Jacobian has been used here, which of course introduces an approximation error. Furthermore, the finite precision of the obtained steady states is another error source. This explains why the error levels off. For all practical optimization purposes, the obtained accuracy is more than enough.

g:dLdMa_RANS_ale

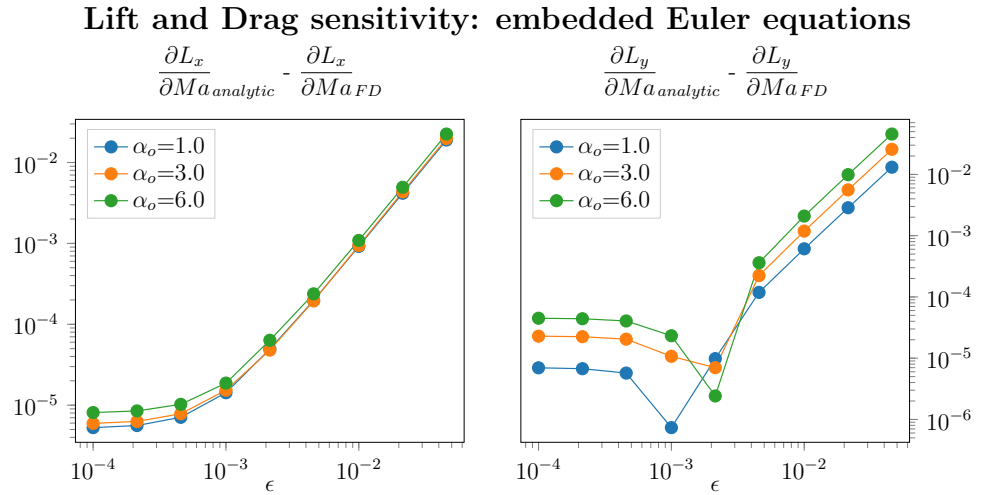


Figure 32: The graph shows the relative error between the Lift and Drag Mach-sensitivities obtained by FD of two steady states, compared to the mach-sensitivity obtained by analytic direct evaluation. L_x denotes the Drag, L_y denotes the lift. Please note that a FD Jacobian has been used here, which of course introduces an approximation error. Furthermore, the finite precision of the obtained steady states is another error source. This explains why the error levels off. For all practical optimization purposes, the obtained accuracy is more than enough.

:dLdMa_Euler_emb

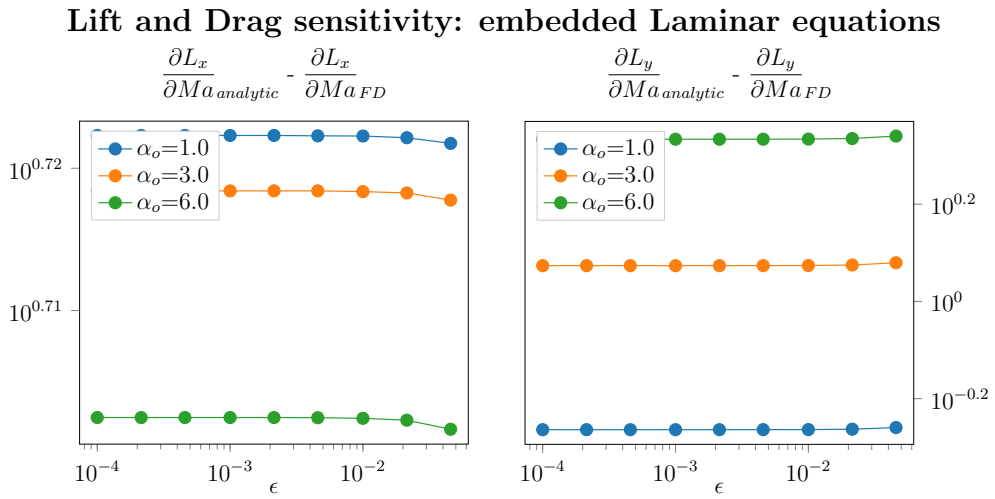


Figure 33: The graph shows the relative error between the Lift and Drag Mach-sensitivities obtained by FD of two steady states, compared to the mach-sensitivity obtained by analytic direct evaluation. L_x denotes the Drag, L_y denotes the lift. Please note that a FD Jacobian has been used here, which of course introduces an approximation error. Furthermore, the finite precision of the obtained steady states is another error source. This explains why the error levels off. For all practical optimization purposes, the obtained accuracy is more than enough.

_Laminar_emb

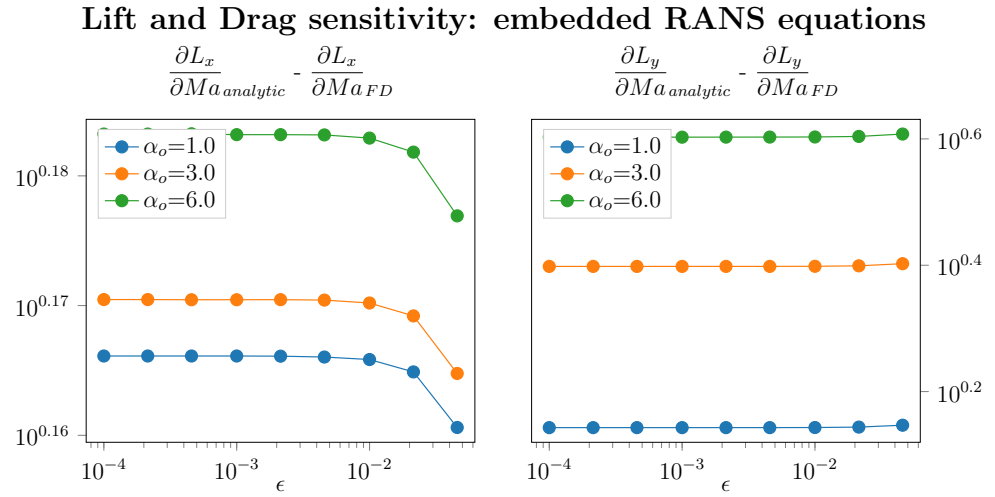


Figure 34: The graph shows the relative error between the Lift and Drag Mach-sensitivities obtained by FD of two steady states, compared to the mach-sensitivity obtained by analytic direct evaluation. L_x denotes the Drag, L_y denotes the lift. Please note that a FD Jacobian has been used here, which of course introduces an approximation error. Furthermore, the finite precision of the obtained steady states is another error source. This explains why the error levels off. For all practical optimization purposes, the obtained accuracy is more than enough.

dMa_RANS_emb

References

[1] Joseph-Frédéric Bonnans, Gilbert Jean Charles, Lemarechal Claude, and Sagastizábal Claudia A. Numerical optimization: Theoretical and practical aspects. *Numerical Optimization: Theoretical and Practical Aspects*, pages 1–494, 2006.

[2] Charles R. Doering and J. D. Gibbon. *Applied analysis of the Navier-Stokes equations*. Cambridge University Press, Cambridge, 1995.

[3] C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering*, 163(1-4):231–245, 1998.

[4] Charbel Farhat, Michel Lesoinne, and Nathan Maman. Mixed explicit/implicit time integration of coupled aeroelastic problems: Three-field formulation, geometric conservation and distributed solution. *International Journal for Numerical Methods in Fluids*, 21(10):807–835, 1995.

[5] Farhat Research Group. Aero-F.

[6] Farhat Research Group. Aero-S.

[7] Michel Lesoinne, Marcus Sarkis, Ulrich Hetmaniuk, and Charbel Farhat. A linearized method for the frequency analysis of three-dimensional uid / structure interaction problems in all ow regimes. 190:1–38, 2001.

[8] Randall J. LeVeque. *Numerical Methods for Conservation Laws*, volume 33. 1992.

[9] Alexander Main. Algorithmic Problems in Social and Geometric Influence a Dissertation Submitted To the Institute for Computational and Mathematical Engineering and the Committee on Graduate Studies of Stanford University in Partial Fulfillment of the Requirements for the. 2014.

[10] K. Maute, M. Nikbay, and C. Farhat. Coupled analytical sensitivity analysis and optimization of three-dimensional nonlinear aeroelastic systems. *AIAA Journal*, 39(11):2051–2061, 2001.

[11] K. Maute, M. Nikbay, and C. Farhat. Sensitivity analysis and design optimization of three-dimensional non-linear aeroelastic systems by the adjoint method. *International Journal for Numerical Methods in Engineering*, 2003.

[12] Charles Peskin. Flow patterns around heart valves: A numerical method. *Journal of Computational Physics*, 271(0021-9991):252–271, 1972.

[13] Serge Piperno and Charbel Farhat. Design of Efficient Partitioned Procedures for the Transient Solution of Aeroelastic Problems. *Revue Européenne des Éléments Finis*, 9(6-7):655–680, jan 2000.

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

[15] K. Schittkowski, C. Zillober, and R. Zotemantel. Numerical comparison of nonlinear programming algorithms for structural optimization. *Structural Optimization*, 7(1-2):1–19, 1994.

[16] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Berkeley, 1999.

[17] Jaroslaw Sobieszczanski-Sobieski. Sensitivity of complex, internally coupled systems. *AIAA Journal*, 28(1):153–160, 1990.

[18] P R Spalart and S R Allmaras. A one-equation turbulence model for aerodynamic flows. *La Recherche Aerospatiale*, 1(1):5–21, 1994.

[19] Bram van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of Computational Physics*, 32(1):101–136, 1979.