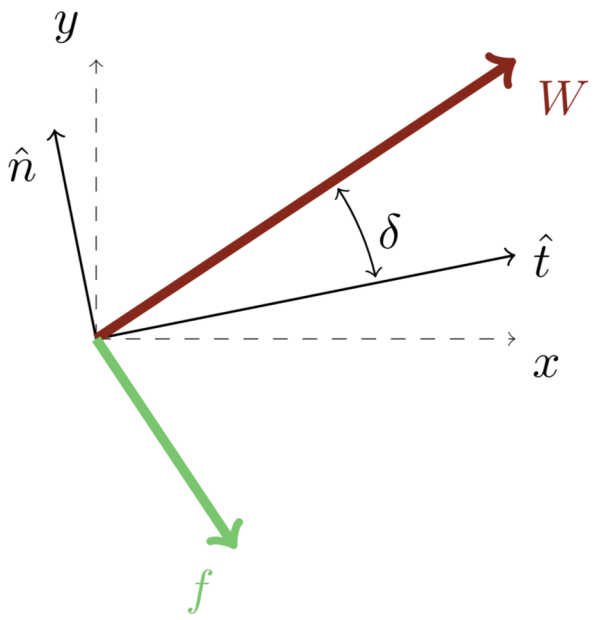
Body force modeling of axial turbomachinery for analysis and design optimization

by:

M.T. Latour

Delft University of Technology



TU Delft

This page is intentionally left blank.

Body force modeling of axial turbomachinery for analysis and design optimization

by:

M.T. Latour

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Wednesday April 1st, 2020 at 14:30.

Cover image: Visualization of Hall's body force model for an arbitrary point on a camber line with camber normal ($\hat{n}$) and tangent ($\hat{t}$) axes. The velocity vector is denoted by W and the force by f while the meridional and tangential axes are denoted by x and y respectively.

Student number: 4203607
Project duration: March 18, 2019 – April 1, 2020

TU Delft thesis committee:

Dr. ir. M. Pini	*Propulsion & Power, Assistant Professor*	Supervisor
Prof. dr. ir. P. Colonna di Paliano	*Propulsion & Power, Professor*	Chair
Dr. ir. A.H. van Zuijlen	*Aerodynamics, Assistant Professor*	External Committee Member

TU Delft

This page is intentionally left blank.

Preface

The last seven-and-a-half years have defined me as a person and have been unforgettable. Not only have I changed as a person, the world around me has changed as well. We are constantly faced and informed of a changing climate. Besides the practical knowledge and skills that I have gained during these last twelve months, I hope that my work will be a cornerstone for future conceptual turbomachinery design, with the goal being to bring more efficient designs to the market quicker. As not everyone may know, turbomachinery is everywhere, so not only could this have potential in aviation, but also in the energy, automotive, and industrial sectors.

Starting this thesis would not have been possible without Matteo Pini. During the turbomachinery course his teaching was excellent and enthusiasm infectious, which nudged me towards a thesis in this area of research. The practical experience that I have gained in C++ as well as the knowledge in optimization methods are invaluable and will stay with me throughout my career. Besides being helpful our meetings were fun, and organizing a hack-a-thon together was a valuable experience. I was lucky enough to participate in an exchange to the Univeristy of Windsor during this thesis, which would not have been possible without Jeff Defoe. Without his guidance as well as his and his team's help I would not have made as much progress as I did in the three months that I was there. Thank you to Palak, Mehran, and Majed for the company during my exchange. A special thank you goes to Nitish, who helped me countless times with C++ and the SU2 repo, be it teaching or debugging.

Mom, Dad, Soph, you are awesome and helped me get through tough times this year. Robert, Britt, AJ, Margriet, and Sjors, thanks for all the support and fun times. There are countless others who have made my time as a student amazing, so thank you for that! It's a weird feeling to wrap up this part of my life, but I am excited to apply the knowledge and skills that I have gained to tackling some of the world's current pressing issues.

M.T. Latour
Amsterdam, March 2020

This page is intentionally left blank.

Abstract

Next-generation propulsion systems will likely take advantage of boundary layer ingestion (BLI), which requires simulation and design tools to perform optimization at an affordable cost while taking airframe interaction into account. A solution for this problem is the reduced-order body force model (BFM). It allows for computationally inexpensive flow analysis, and when combined with the adjoint method accelerates conceptual design optimization. In this thesis, Hall's inviscid body force model is implemented into the direct and adjoint solvers of the open-source software SU2. Direct solver runs of the BFM show results consistent with what is expected from theory for a range of flow characteristics. Furthermore, fourteen-fold reductions in grid size as well as one to two orders of magnitude reduction in computational time are observed while retaining flow behavior. In the adjoint solver, registration of source terms is successful, which is confirmed by changes in the adjoint flow over the body force domain. Physical interpretations of the adjoint vector agree with theory. Total gradient computation is attempted but unsuccessful. It is therefore recommended for future work, as showing this capability is a large step towards faster turbomachinery conceptual design optimization.

This page is intentionally left blank.

Contents

List of Figures

List of Tables

This page is intentionally left blank.

Nomenclature

List of acronyms

Acronym	Definition	Chapter
AD	Algorithmic Differentiation	*2, 3*
BFM	Body Force Model	*1-4*
BLI	Boundary Layer Ingestion	*1, 2*
BWB	Blended Wing Body	*1*
CFD	Computational Fluid Dynamics	*2, 3*
CoDiPack	Code Differentiation Package	*3*
FD	Finite Difference	*3, 5*
GBD	GNU Debugger	*4*
GMRES	Generalized Minimal Residual Method	*3*
HB	Harmonic Balance	*2*
IDE	Integrated Development Environment	*4*
LE	Leading Edge	*4*
LES	Large Eddy Simulation	*3*
NURBS	Non-Uniform Rational B-Spline	*6*
PDE	Partial Differential Equations	*2*
RANS	Reynolds-Averaged Navier Stokes	*2*
SU2	Stanford University Unstructured	*1-6*
TE	Trailing edge	*4*
UMG2	Unstructured Mesh Generator 2D	*3*
URANS	Unsteady Reynolds-Averaged Navier Stokes	*1, 2*

List of symbols

Symbol	Description
α	Angle of local camber line tangent $\hat{t}$ with respect to the meridional axis
$\boldsymbol{\alpha}$	Vector of design variables
B	Number of blades
δ	Angle of flow deflection between camber tangent $\hat{t}$ and relative velocity vector W
e_t	Specific total energy
$\boldsymbol{F^c}$	Convective flux vector
$\boldsymbol{F^{vk}}$	Viscous flux vector
f	Body force magnitude
f_t	Tangential component of the body force
f_{t_x}	X-component of the tangential component of the body force
f_{t_y}	Y-component of the tangential component of the body force
f_n	Normal component of the body force
f_{n_x}	X-component of the normal component of the body force
f_{n_y}	Y-component of the normal component of the body force
$\boldsymbol{G}$	Fixed-point operator
h_t	Total enthalpy
$h_{t_{rel}}$	Total relative enthalpy
θ	Angle between camber normal vector $\hat{n}$ and relative velocity W
J	Objective function
K	Compressibility factor
M	Mach number
μ^{vk}	Viscosity

$\hat{n}$	Camber normal unit vector
$\hat{n}_\theta$	Tangential component of camber normal unit vector
$\hat{n}_x$	X-component of camber normal unit vector
$\hat{n}_y$	Y-component of camber normal unit vector
$\hat{n}_z$	Z-component of camber normal unit vector
p	Pressure
p_t	Total pressure
$p_{t_{rel}}$	Total relative pressure
$\boldsymbol{Q}$	Generic source term vector
R	Radius
$\boldsymbol{R}$	Residual vector
ρ	Density
s	Local blade pitch
$\hat{t}$	Camber tangent unit vector
$\hat{t}_x$	X-component of camber tangent unit vector
$\hat{t}_y$	Y-component of camber tangent unit vector
$\hat{t}_z$	Z-component of camber tangent unit vector
U	Tangential velocity
$\boldsymbol{U}$	Conservative variables
$\bar{\boldsymbol{U}}$	Adjoint vector
V	Absolute velocity
$\boldsymbol{V}$	Velocity vector
W	Relative velocity
$\boldsymbol{X}$	Grid points of the volumetric mesh
$\bar{\boldsymbol{X}}$	Mesh node sensitivity
$\boldsymbol{X}_{surf}$	Grid points of the surface mesh
$\boldsymbol{\Psi}$	Adjoint vector
Ω	Rotational speed
ω	Angular velocity

1 Introduction

As aviation continues to grow, engines must become more efficient to reduce fuel burn to acceptable levels. Incremental upgrades and novel designs to reach these goals are not completed overnight; design of engines can take years or even decades. Accelerating the launch of these engines to the market requires faster turbomachinery analysis as well as design optimization. Turbomachinery's inherent unsteadiness demands time-consuming, full-annulus, unsteady Reynolds Averaged Navier Stokes (URANS) simulations to resolve all relevant flow behavior. Analysis of novel concepts such as boundary layer ingesting (BLI) fans can take upwards of two weeks [Gunn and Hall, 2014]. Even incremental design upgrades require high computational power. The design process requires a large number of these runs to first reduce the design space and subsequently find an optimal design. Instead of solely resorting to high-fidelity simulations, a computationally inexpensive method is necessary to narrow down the design space. This relieves conceptual design from more tedious simulations while retaining them for detailed analysis, producing results in a much shorter time span. A body force model (BFM) meets these requirements. Design optimization can be shortened further by combining it with the adjoint method, a method for computing gradients that is less computationally expensive than conventional methods. This thesis focuses on using a body force model in an adjoint solver to reduce the time required for design optimization.

A BFM represents the forces exerted by a blade row on the flow using volumetric source terms added to the governing equations. It does not require the physical blade to be in the computational domain, which allows considerable reduction in mesh refinement. In 2015, [Hall, 2015] developed an inviscid, incompressible BFM that has been confirmed to accurately represent a fan in studies by [Hall et al., 2017], [Defoe et al., 2018a], and [Hill and Defoe, 2018]. Much of the work related to body force models is in the field of BLI fans, however it can be applied to all types of axial turbomachinery. Hall's BFM has been implemented into a number of direct solver but has never been extended for use with the adjoint solver. The open-source computational fluid dynamics (CFD) solver SU2 makes this possible. SU2 is a multi-physics partial differential equation (PDE) solver with a modular code structure. Its built-in adjoint solver performs computationally inexpensive optimization using the adjoint method [Jameson, 1988]. Partial derivatives of the flow with respect to the design variables are substituted out of the gradient computation, leading to a method whose computational time is *independent* of the number of design variables [Wang and He, 2010].

The goal of this thesis is to implement Hall's BFM in SU2's direct solver and subsequently extend it for use with the adjoint solver. Body force simulations reduce computational requirements for blade row analysis, while an extension to the adjoint solver will shorten conceptual design by optimizing a camber line representation. Quick investigation of preliminary design concepts are then possible, while detailed design is left to higher-fidelity methods. This capability has never been proven before, and when successful will be a step towards faster design of incremental upgrades and novel design concepts for aircraft engines.

1.1. Research Question

After a study on the literature of body force models a research question is defined. This shapes the work in this thesis and is the driving force behind the results presented throughout this report.

Research Question: *What is the computational performance and accuracy of a body force model compared to full-bladed computations when used for analysis and design optimization of 2D axial fans in SU2?*

(a) Is the software's source code able to support the addition of body forces?
(b) What are the key parts of SU2's source code that need to be adapted to ensure successful implementation of a body force model in the direct and adjoint solvers?
(c) Does the resulting direct flow obtained using a body force model resemble that of conventional simulations run in SU2?
(d) Do the resulting adjoint flow and sensitivities obtained using a body force model resemble what is expected from theory?
(e) Can the model be implemented such that it supports all geometries and configurations (for turbomachinery applications)?
(f) What are the benefits and drawbacks of using a body force model in SU2?

1.2. Originality of Work

The output of a research project should be original and relevant to the scientific community. The points below highlight how the work presented in this thesis meets those criteria.

- Hall's body force model is implemented in the direct solver of SU2, which has up until this point never been done.
- The model is verified according to theory, implementation of Hall's model in another CFD solver, and by comparing it with physical blade simulations of a NACA0012 stator and the Whittle fan.
- Computational time reduction when using body forces as compared to physical blade simulations is confirmed. Although this has been done before in other solvers it has never been performed in SU2. An analysis into the flow accuracy as mesh refinement is reduced is included as well.
- The body force model is extended to SU2's adjoint solver. Mesh sensitivity is calculated with body force source terms registered, which has never been done before in any CFD solver.

1.3. Limitations of the Work

Two factors have hindered obtaining all the results necessary to answer both research questions fully. The time constraint associated with a thesis made calculation of the total gradient in SU2's adjoint solver unobtainable; a segmentation fault lasting six weeks being a key driver for this. Furthermore, limited experience in C++ slows down the coding process and lengthens the time required for implementation and troubleshooting. A substantial amount of time, however, was spent understanding the code's structure; therefore knowledge on how to adapt the code to compute total gradient is available. To speed up the learning process for the person following up on this research, the methodology for total gradient computation is included in this report while the results are not.

1.4. Structure

Background information on body force models, SU2's code structure, and the adjoint method are presented in Chapter 2. Next, Chapter 3 presents the methodology, which covers the body force model's implementation and its linking with the adjoint solver. It will elaborate on the calculations behind the model and camber normal representation, meshing of the test cases, and how SU2's source code is adapted for both the direct and adjoint solver. Following this, Chapter 4 elaborates on the results of the direct implementation. A number of test cases are presented, which are compared to theory, their physical blade simulation counterpart, or the implementation of Hall's model in openFOAM. Also included are an evaluation of computational time reduction when using body forces. Chapter 5 dives into the sensitivities and total gradients obtained from running the adjoint solver with body forces. Finally, Chapter 6 presents conclusions of the thesis, proposes recommendations for technical improvement of the implementation, and suggests a number of future research directions.

2

Background

This chapter starts by going over the literature on body force models, which is presented in Section 2.1. A new computational fluid dynamics (CFD) solver called SU2 is covered in Section 2.3. Some relevant case studies are presented in Section 2.4 that look into previous implementations of body force models in CFD solvers. Finally, in Section 2.5 the adjoint method is explained and a background on the relevant literature presented.

2.1. Introducing Body Force Models for Fan Analysis

Current developments in aviation are aligned with the mindset of reducing (fuel) costs. One way this goal will be achieved is through novel engine design. This includes incremental improvements in blade design as well as completely new concepts such as boundary layer ingestion (BLI). Conceptual design is usually the starting point, which uses lower fidelity analyses to narrow down the design space. Following this is detailed design. Full-annulus unsteady Reynold's Averaged Navier-Stokes (URANS) simulations are usually run for both, which resolves many flow characteristics of turbomachinery but generally makes it a tedious process. While accuracy with this type of simulation has been shown to qualitatively reflect experimental behavior, [Gunn and Hall, 2014] notes that runs on a unit cluster take on the order of two weeks to complete. While its necessity for detailed design cannot be denied, having to run calculations such as these for initial fan design is time consuming and expensive. Reducing computational time while maintaining a sufficient level of accuracy for (conceptual) design is possible when substituting physical blades with a body force model. Many studies on the topic of body force modeling is related to BLI fans. Besides the reduction in mesh refinement, a steady simulation can "effectively capture the principal features of fan response to BLI distortion" [Hall et al., 2017]. This does not mean that a BFM cannot be run in an unsteady simulation, only that it allows a steady simulation to be run even with the unsteady effects of distorted inlet flow. Despite this focus on BLI fans, body forces can be used to model conventional fans as well, which will be the focus of this thesis.

2.2. Body Force Modeling

The concept of body force modeling started with [Marble, 1942], who represented blades by splitting them into an infinite number of sections and approximating flow characteristics using forces. Since its inception the approach has not varied much, with the main idea being that flow turning, enthalpy rise, and viscous losses are approximated using a volumetric source term field. Over the years adaptations and developments were made to the concept, with the most notable ones being done by [Gong, 1999], [Peters, 2014] and [Hall, 2015]. Gong's model was innovative in that the body forces responded to local flow characteristics, allowing it to react to physical changes in the flow [Thollet, 2017]. Changes to the flow were made using a pair of forces: while a component normal to the streamline approximated flow turning and enthalpy rise, a parallel component simulated viscous losses [Gong, 1999]. Fifteen years later [Peters, 2014] adapted Gong's model by (1) reformulating the parallel force to accurately capture off-design losses and (2) including a radial component of the force to account for blade lean. Most recently, [Hall, 2015] formulated an inviscid, incompressible body force model that does not require calibration using single-passage RANS computations. The general concept behind body force modeling is shared among all authors and is depicted in Figure 2.1, which visualizes the difference in flow over a blade row for a physical blade and its body force representation.

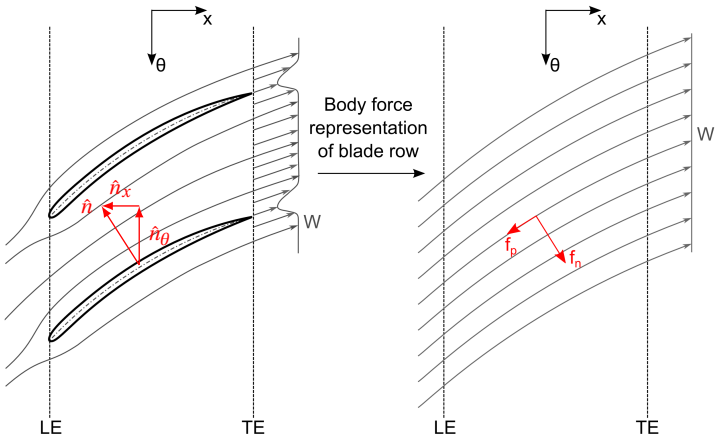


Figure 2.1: *Flow through a physical blade passage (left) and a body force model approximation of the flow (right). A normal (f_n) and parallel (f_p) force approximate flow turning and drag/losses. The camber normal ($\hat{n}$) is used for computation of the body force and is split in tangential ($\hat{n}_\theta$) and meridional ($\hat{n}_x$) components. Without physical blades deflecting flow around the surface it is uniform over the length of the domain, exiting without tangential variations in velocity [Peters, 2014].*

Hall's model has been used in studies on BLI by [Hill and Defoe, 2018], [Defoe et al., 2018a], [Defoe et al., 2018b], [Hill, 2017], and [Thollet, 2017], with its popularity attributed to an accurate representation of flow distribution and lacking the need for calibration. It is for these reasons as well as its simplicity that it is chosen as the model used in this thesis. This does not mean the model can replace full-annulus URANS simulations, as it will not resolve secondary flow to that level. It does, however, "allow the estimation of flow fields using steady CFD methods, and has been compared to experimental data and high-fidelity computations and shown to effectively capture the principal features of fan response to BLI distortion" [Hall et al., 2017]. Using a BFM is therefore useful in conceptual design to narrow down the design space, but higher-fidelity methods do not lose their importance for detailed design. Section 3.2 explores Hall's model and its definitions as well as how it is implemented into the governing equations. Confirmed by [Defoe et al., 2018a] and supported by other research, the use of body force models reduces computational cost by 2 to 3 orders of magnitude. This significantly reduces the required computational power and simulation run time for blade design.

In Hall's body force model, compressibility and viscous effects are neglected. Previous models used a pair of forces: a normal one for flow turning and a parallel one for entropy generation (drag). Hall does not include the parallel term because at the design point "the contribution of the parallel force to flow turning, pressure rise, and enthalpy rise is much smaller than the normal force" [Hall et al., 2017]. Furthermore, because it is only used to approximate flow and not perfectly resolve it, a viscous term is not necessarily required.

Running viscous simulations would have to be performed using either Gong's or Peters's models, or an adaptation of Hall's. Ideally the development of an entropy term based on first principles could simplify viscous simulations. This is incredibly complicated however, as the mechanisms for loss generation are extremely complex and probably impossible to determine using first principles [Thollet, 2017].[1] To focus the scope of the thesis Hall's inviscid model will be implemented, allowing efforts to be directed at one method. This work will concentrate on a two-dimensional blade row, as this will limit complexity and allow all effort to be directed to implementation of the model.

Instead of looking at the blade passage as in Gong's case, Hall looks directly at the blade camber distribution and generates a force using the deviation of the flow with respect to the blade's camber. The magnitude of the force is calculated as shown in Equation 2.1. When present the force acts normal to the direction of the streamline and acts to reduce the deviation angle δ between the flow velocity W and the camber tangent $\hat{t}$ [Hall, 2015]. A visualization of the model is presented in Figure 3.4. The model can be applied to both stators and rotors.

$$|f| = 2\pi\delta \frac{1}{2} W^2 \frac{1}{s} \frac{1}{|\hat{n}_\theta|} \tag{2.1}$$

[1] A calculation from first principles is one starting from established laws of nature without using assumptions or models. In this case Thollet defines first principles as primitive flow characteristics that do not require calibration.

Where W is the magnitude of the relative (rotor) or absolute (stator) velocity; δ is the angle between the camber tangent $\hat{t}$ and velocity W; s is the local blade pitch, which is equivalent to $\frac{2\pi R}{B}$ (with B being the number of blades and R the radius); and $\hat{n}_\theta$ is the tangential component of the camber normal unit vector.

2.3. SU2

Some CFD solvers that have already been used with body force models are ANSYS CFX [Hill, 2017], Numeca FINE/Turbo [Peters, 2014], elsA [Benichou et al., 2019], and openFOAM.[2] Of the four software packages openFOAM is the only one that is open-source and therefore cheaper for academia, however its usability can be tedious, especially for developers. Further development and understanding of the body force model requires an open-source solver with code that is modular and easily adaptable. A software package that meets these requirements is SU2. SU2 solves a set of partial differential equations (PDEs), which are used to model multidimensional problems such as aerodynamics, heat transfer, and sound propagation. It encourages development from the community, an example being a team from the TU Delft who recently implemented a discrete adjoint solver for turbomachinery. To ensure code development does not become fractured if not coordinated properly [Economon et al., 2016], safety nets such as code review (Git pull requests), bug reporting, and a structured approach to development are in place.

Even with numerous universities working on developing SU2's capability, a body force model has never been implemented into the code. Other (proprietary) software such as ANSYS CFX, elsA, and Turbostream do *have* Hall's BFM implemented. Adding this capability to SU2 would allow (conceptual) turbomachinery design to be carried out using open-source software by capitalizing on the adjoint method. Modeling of fluid dynamics for viscous and compressible flow in SU2 is governed according to the Navier-Stokes equations. The PDE form is:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F^c} - \nabla \cdot (\mu^{vk} \boldsymbol{F^{vk}}) = \boldsymbol{Q}, \tag{2.2}$$

where $\boldsymbol{U}$, $\boldsymbol{F^c}$, $\boldsymbol{F^{vk}}$, and $\boldsymbol{Q}$ are the conservative variables, convective fluxes, viscous fluxes, and generic source terms respectively [Palacios et al., 2014]. Implementation of the body force model into a program such as SU2 requires the addition of source terms ($\boldsymbol{Q}$ in Equation 2.2) to the momentum and energy equations. This is covered in Section 3.3.

Written in C++, SU2's source code is based on object-oriented programming, which uses classes throughout the code for modularity (see Figure 2.2). It should be noted that a disadvantage of this object-oriented structure is "the encapsulation of data in multiple classes, which can lead to performance losses" [Economon et al., 2016]. After compilation a number of executables are created, the two most relevant for this thesis being `SU2_CFD` and `SU2_CFD_AD`.[3] These run the direct or adjoint solver respectively, each with numerous options for the type of flow, numerical method, and boundaries used. A configuration file is necessary to set up a simulation, which contains all settings related to the run.

Starting from the top, SU2 has a *CDriver* class that handles the process of running a (multi-) physics simulation. It instantiates a number of classes that set the process in motion. For `SU2_CFD` these are, amongst others, *CConfig*, *CSolver*, *CIntegration*, and *CIteration*. The *CSolver* class contains a number of child classes that can be instantiated when a certain type of solver is needed, for example Euler, Navier-Stokes, or adjoint [Palacios et al., 2013]. Each solver contains subroutines that compute, amongst others, convective fluxes, viscous fluxes, and source terms over the mesh. These are part of the *CNumerics* class. Figure 2.2 shows the class structure employed by the software, with clear links between classes that interact with each other by either instantiating or being instantiated.

This work focuses on implementation feasibility, meaning solver complexity will only make the process more time consuming. That is why a compressible Euler solver is used in this thesis. Turbulence models and viscosity makes for unnecessary complexity, and are irrelevant when an inviscid body force model is used. An implicit formulation of the Euler equation is used with the generalized minimum residual method (GMRES) as the linear solver [Saad and Schultz, 1986]. To compute numerical spatial gradients the Green-Gauss method is used. The finite volume method is used to discretize the PDE's. Besides Roe [Roe, 1981] and JST [Jameson, 2017], a wide range of numerical methods are available for discretization of the convective fluxes.

[2] Jeff Defoe's Turbomachinery and Unsteady Flows group at the University of Windsor has successfully implemented a 3D version of Hall's body force model in openFOAM but has not yet published their results.

[3] All of the executables created by SU2 are described in detail in [Palacios et al., 2013]

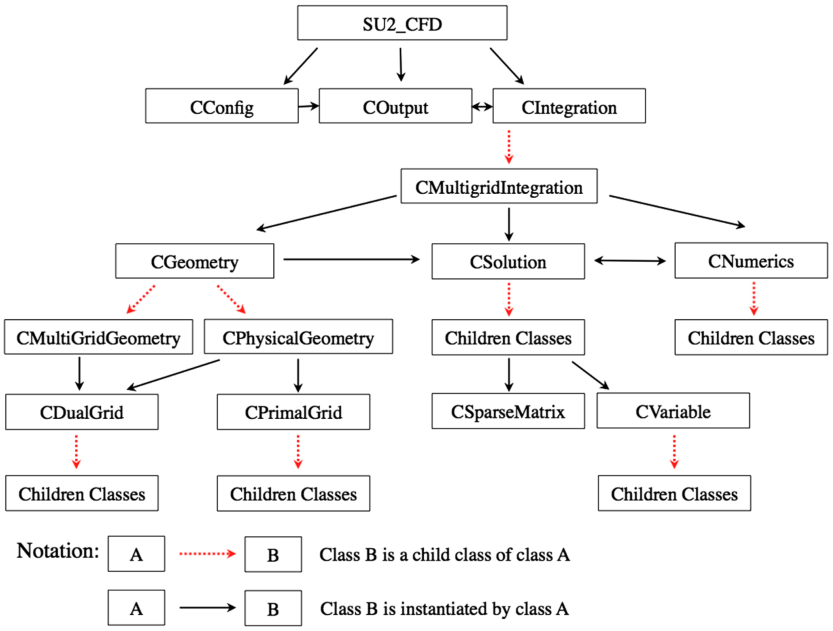


Figure 2.2: *Source code structure of the open-source software SU2. A complex class structure allows for easy adjustment of the source code to meet the needs of future changes to the software [Economon et al., 2016].*

While upwind schemes are generally less dissipative, the central scheme JST is used for this work due to its robustness. This is because early testing showed convergence issues when an upwind scheme was used. Second and fourth order dissipation coefficients are taken as 0.5 and 0.02 respectively.

Boundary conditions are used to specify the physical setup of the problem. Each boundary is identified using a marker, the relevant ones for this work being inflow, outflow, periodic, Giles, mixing plane, and Euler wall. Both the inflow and outflow are defined using a number of flow variables. At the inlet stagnation pressure and temperature are defined along with the velocity unit vector. Static back pressure is defined at the outlet. Periodic boundary conditions are defined using the donor and receiving markers, which ensure flow correctly transfers from one boundary to the next. Giles conditions are used in turbomachinery simulations where flow characteristics are defined according to an averaged value at that location so that reflections at the boundary are neglected [Giles, 1991]. Both this boundary condition and the mixing plane, which is used to average out flow characteristics at a boundary, were used solely in the Whittle fan simulation. Finally, an Euler wall boundary is used for the surfaces of a physical blade (when simulated).

2.4. Relevant Case Studies

The literature on implementation of body force models in CFD solvers is not as extensive as one might think, however there are a number that are worth mentioning. While [Gong et al., 1999] and [Horlock and Marsh, 2007] investigate flow models for turbomachinery, there is no specification as to their implementation in any form of software. In 2014 [Peters, 2014] implemented a body force model in Numeca FINE/Turbo. He adapted the models and used them for an inlet design framework that captures inlet-fan and fan-exhaust interaction as well as flow distortion at the interface. In 2015, [Hall, 2015] came up with his inviscid version of a body force model. It is a widely-used model and in 2017 Hill and Thollet successfully implemented it in CFX and elsA respectively. In the presence of various flow distortion, [Hill, 2017] extracted scaling laws for transonic compressor performance. While this was directly implemented into CFX, [Thollet, 2017] used an independent Python module coupled to the solver. Adaptations were made to the models and comparisons were made between them to analyze how certain additions could increase accuracy of fan-airframe interaction simulations.

Implementation of Hall's model into CFX is used extensively at the University of Windsor by [Defoe et al., 2018a], [Defoe et al., 2018b], [Hill and Defoe, 2018], and [Minaker and Defoe, 2019] for a range of applications. Validation of Hall's model for the Whittle fan using an experimental setup was performed by [Hall et al., 2017]. Furthermore [Benichou et al., 2019] also implemented an adapted version of Hall's model (in

elsA) and compared results to an experimental rig, confirming its capability of accurately representing flow through a fan stage with distorted inlet flow. Numerous solvers have therefore seen Hall's body force model implemented successfully, however it has never been done with SU2. This is relevant because SU2 is open-source and it has a built-in adjoint solver. Never has a body force model been combined with the adjoint method. Blade shape optimization using body force models is a completely innovative solution and has the potential to considerably shorten conceptual blade design.

2.5. Adjoint Method for Optimization

When performing optimization the goal is to minimize an objective function, denoted by J, which usually represents efficiency or some other performance-related metric. The shape to be optimized is defined using a number of design variables, denoted by $\boldsymbol{\alpha}$. These design variables are varied until the optimal value of the objective function is found. This is done using stochastic or gradient-based method, the former performing global searches for an optimum while the latter calculates gradients in multiple directions, chooses the best option, and subsequently moves towards the optimum [Li and Zheng, 2017]. The adjoint method falls under the class of gradient-based methods.

If the set of design variables $\boldsymbol{\alpha}$ is large, derivatives of the objective function with respect to *each* of these need to be computed. If using finite differences, this requires solving the flow field every time a design variable is perturbed, and because the computational cost is proportional to the number of design variables, is extremely costly. The adjoint method solves this problem as its computational cost is *independent* of the number of design variables, which is the reason for its popularity in turbomachinery design.[4] A downside of the method is its complexity, which according to [Wang and He, 2010] has curbed its popularity. There is also the high memory requirement associated with the discrete adjoint method.

Adjoint Equations

An objective function J (Equation 2.3) is defined as a function of the flow $\boldsymbol{U}$ and shape variables $\boldsymbol{\alpha}$. It is supplemented by the flow equation $\boldsymbol{R}$, which is equal to zero in steady-state problems and always has to be satisfied (Equation 2.4). As noted by [Pini, 2013], another interpretation (not used in this thesis) of the flow equation is as an equality constraint for the minimization problem.

$$J = J[\boldsymbol{U}(\boldsymbol{\alpha}), \boldsymbol{\alpha}] \tag{2.3}$$

$$\boldsymbol{R}[\boldsymbol{U}(\boldsymbol{\alpha}), \boldsymbol{\alpha}] = 0 \tag{2.4}$$

Differentiating Equations 2.3 and 2.4 using the chain rule, both can be rewritten as

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \frac{\partial J}{\partial \boldsymbol{U}} \frac{d\boldsymbol{U}}{d\boldsymbol{\alpha}} \tag{2.5}$$

$$\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\alpha}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}} \frac{d\boldsymbol{U}}{d\boldsymbol{\alpha}} = 0 \tag{2.6}$$

The term $\frac{dU}{d\boldsymbol{\alpha}}$ in Equation 2.5 requires solving the flow for each design variable adjustment. This is particularly time consuming when the design variables $\boldsymbol{\alpha}$ are numerous. *However, the calculation of this term can be avoided by using substitution, which is the basic idea of the adjoint method.* First, Equation 2.6 is re-arranged with $\frac{dU}{d\boldsymbol{\alpha}}$ on the left-hand side, which gives Equation 2.7.

$$\frac{d\boldsymbol{U}}{d\boldsymbol{\alpha}} = -\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\alpha}} \left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}} \right)^{-1} \tag{2.7}$$

This can then be substituted into Equation 2.5, eliminating the need to compute the derivative of the flow for each design variable. This gives Equation 2.8:

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} \underbrace{- \frac{\partial J}{\partial \boldsymbol{U}} \left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}} \right)^{-1}}_{\boldsymbol{\Psi}} \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\alpha}} \tag{2.8}$$

[4] Understanding of the adjoint method can be complicated, which is why the reader is referred to [Martins and Hwang, 2013], [Giles and Pierce, 2000], and [Li and Zheng, 2017] for supplementary explanation on its workings.

Where $\boldsymbol{\Psi}$ is the adjoint vector and Equation 2.9 the linear system of adjoint equations

$$\left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right)^T \boldsymbol{\Psi} = \left(\frac{\partial J}{\partial \boldsymbol{U}}\right)^T \tag{2.9}$$

The objective sensitivity can then be re-written as:

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \boldsymbol{\Psi}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\alpha}} \tag{2.10}$$

Computational cost of the gradient is now independent of the number of design variables and requires solving for the adjoint vector (Equation 2.9) and the partial derivatives. Algorithmic Differentiation (AD) of relevant routines in SU2 is performed by Code Differentiation Package (CoDiPack), which computes the partial derivatives required for the adjoint method.

Sensitivities Computation Using an Adjoint Solver

Calculating the objective sensitivity (also called the total gradient) using SU2's adjoint solver is done using a product of three sensitivities. Due to the objective function's dependence on the mesh, this needs to be taken into account in the gradient's computation. By using the differentiation chain rule, the total sensitivity can be written as shown in Equation 2.11. $\boldsymbol{X}$ and $\boldsymbol{X}_{surf}$ are the vectors of grid points of the volumetric mesh and surface mesh respectively.

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{dJ}{d\boldsymbol{X}} \frac{d\boldsymbol{X}}{d\boldsymbol{X}_{surf}} \frac{d\boldsymbol{X}_{surf}}{d\boldsymbol{\alpha}} \tag{2.11}$$

Where $\frac{dJ}{d\boldsymbol{X}}$ is the objective sensitivity, $\frac{d\boldsymbol{X}}{d\boldsymbol{X}_{surf}}$ the mesh sensitivity, and $\frac{d\boldsymbol{X}_{surf}}{d\boldsymbol{\alpha}}$ the geometric sensitivity. The objective and mesh sensitivity can be calculated directly in SU2 [Palacios et al., 2014]. Computation of the geometric sensitivity can be performed using the finite difference (FD) method; while tedious it is still considerably cheaper than calculating the flow and adjoint solutions. AD can also be used for the mesh and geometric sensitivities provided that there is access to the mesh generation and deformation code.

Discrete vs. Continuous

Two main approaches exist for the adjoint method, namely discrete and continuous. In the former the flow equations are first discretized and subsequently linearized, whereas the latter performs these actions in reverse. For both approaches, the final result is a set of discrete adjoint equations. The continuous adjoint method has a lower computational requirement and has a simpler code structure. On the contrary, a discrete adjoint method is straightforward and provides an exact value of the discrete objective function gradient [Li and Zheng, 2017]. In theory, both approaches should lead to the same value of the computed gradient, assuming the flow solutions are smooth [Giles and Pierce, 2000]. For this thesis the discrete adjoint method was chosen. Linearizing the governing equations is a complex process and requires significant analytical work if the equations have not been discretized in advance. Once discretized though, linearization is a simple process that can be automated using Algorithmic Differentiation (AD). "AD was developed based on the observation that any simulation code, regardless of its complexity is merely a sequence of elementary operations whose differentiation rules are well known" [Zhou et al., 2015]. Its benefits include a lack of truncation error, which gives it a much better accuracy than FD methods.[5] CoDiPack is included in SU2 and computes gradients of all relevant routines in SU2 using AD when the source code is compiled.

Applications of the Adjoint Method

While the adjoint method has been around for some time in the field of mathematics, its use for optimization was established by [Jameson, 1988]. He demonstrated the method's capability to perform optimization without depending on large calculations of the flow with respect to design variables. Since then the adjoint method has been popular in the turbomachinery design community, specifically due to the large number of design variables required when defining blades and their profiles. Since the turn of the century numerous

[5] Principles of AD are beyond the scope of this thesis. For a more extensive description of AD and its working principles in SU2 the reader is referred to [Fernández, 2017].

studies have looked into using adjoint in turbomachinery design. Using Bézier curves, [Arens et al., 2005] applied the adjoint method using 2D Euler equations for a stationary case, concluding with the adjoint method's superiority over other optimization methods. In their work, [Florea and Hall, 2001] applied the discrete adjoint method to unsteady, inviscid flow in 2D turbomachinery, likewise noting the effectiveness of the method. A similar analysis was performed by [Yang et al., 2003], but using a continuous method instead. Three-dimensional optimization of a turbine blade was performed by [Luo et al., 2011], they included the application of constraints in the form of a penalty term as well as boundary conditions to ensure thermodynamic accuracy.

More recently [Anand et al., 2018] used the adjoint method to analyze different parameterization approaches and their effect on finding an optimum solution. It has also been combined together with the reduced-order harmonic balance method by [Rubino, 2019] to perform more accurate optimization as compared to using steady state methods. Areas of opportunity are also present, specifically in the area of combining the reduced-order body force model with adjoint optimization. This has the potential for improving in the speed of performing preliminary blade design and optimization.

2.6. Key Takeaways

The following points highlight the principal outcomes of this chapter.

- *Body force models have the potential to reduce computational cost by 2 to 3 order of magnitude with respect to conventional simulations.*
- *SU2's modular code structure allows adding classes and functions for computing body forces.*
- *The cost of the adjoint method is independent of the number of design variables, leading to savings in computational time for the large number of design variables used in turbomachinery.*
- *Up until now, a body force model and its adjoint counterpart has never been implemented in SU2.*

This page is intentionally left blank.

3

Methodology

This chapter covers the methodology applied during this research, which is visualized in Figure 3.1. First mesh generation is covered in Section 3.1, which elaborates on generation of mesh and types used throughout the work. Next, Hall's body force model is covered in Section 3.2 and explained in detail. Its implementation in the direct solver of SU2 is presented in Section 3.3. Finally, extension of the BFM to SU2's adjoint solver is explained in Section 3.4.

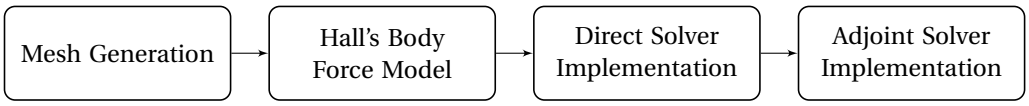


Figure 3.1: *Structure of the methodology.*

3.1. Mesh Generation For Body Force Simulations

Mesh refinement near blade surfaces is unnecessary as boundary layer flow does not need to be resolved. In their work, [Defoe et al., 2018a] showed that a 30-fold decrease in grid count was possible when the reduced order model was used. This retains the level of accuracy of the BFM, however a body force representation will never perfectly represent flow. That is an inherent characteristic of a reduced-order model. Two programs were used for in this thesis for meshing: UMG2 and Pointwise. Unstructured meshes were generated by UMG2 while Pointwise allowed for structured ones to be created. The majority of this work uses unstructured mesh while only a couple of runs are performed on a structured grid for comparative reasons. A short analysis on the differences between the two is presented in Appendix A.2.

3.1.1. UMG2

UMG2 was developed as an unstructured mesh generator at the University of Brescia by [Ghidoni et al., 2006]. It is used for two-dimensional applications and is accessible to the Turbomachinery group at the Aerospace Engineering faculty of the TU Delft.[1] It is run in a shell in the terminal and consists of four files that are edited to meet the user's needs: *options*, *geometry*, *topology*, and *spacingcontrol*. The *options* file allows for some common parameters of the meshing to be varied. In the *geometry* file the mesh is defined using lines (either NURBS or S-curves); this includes shapes in the mesh as well as the size of the domain. The type of boundary is then defined in the *topology* file, with options including inflow, outflow, wall, and periodic. Minimum and maximum cell widths are then defined in *spacingcontrol*, which drives mesh refinement.

3.1.2. Mesh Template

To keep from varying meshes too often a standard mesh is created, which is visualized in Figure 3.2. Zone 2 represents the body force domain and is where the force is applied. When comparing with a physical blade

[1] UMG2 is not publicly accessible. For access the reader is referred to Dr. ir. Matteo Pini, who can grant access to the software and a BitBucket repository containing manuals on mesh creation.

mesh, the width of this cell must match that of the chord of the blade.[2] Zones 1 and 3 contain the upstream and downstream flow respectively. Due to the periodicity and lack of tangential variation in flow, the height of the domain is insignificant. It must not be too narrow, however, as this may cause difficulties due to proximity of the periodic boundary conditions. Dimensions for the domain are not given as they are irrelevant if a comparison with a physical blade is not being made. A domain twice the height would produce the same flow result but with higher computational cost. It is chosen to keep the dimensions the same and vary the grid refinement as this is easy to change in UMG2.

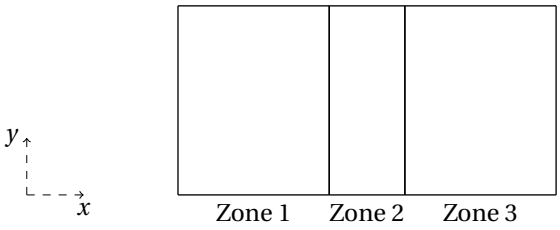
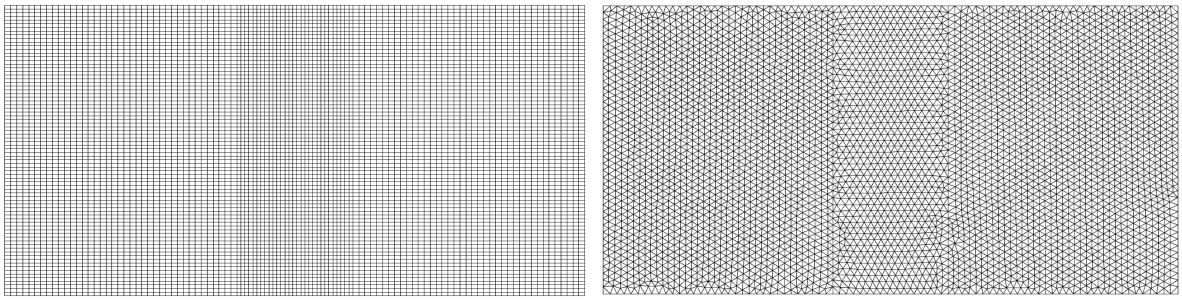


Figure 3.2: *Layout of mesh used for body force model test cases.*

Excluding the meshes used for computational time comparison, all counted the same number of cells in each zone. A total of 8560 cells were used: 3120 in zones 1 and 3 and 2320 in zone 2. The structured mesh in question is presented in Figure 3.3a. Results from Section 4.3 show that further reduction in number of cells is possible while obtaining similar results, but for consistency this mesh is not varied over the entire range of stator and rotor test cases. An unstructured mesh with a similar number of cells was created to test its effect on numerical errors near the interface between the upstream and body force zones. The difference between the two mesh types is negligible and is presented in Appendix A.2. The unstructured mesh is shown in Figure 3.3b.



(a) *Structured mesh created using Pointwise*

(b) *Unstructured mesh created using UMG2*

Figure 3.3: *Meshes used for test case simulations. The structured mesh was constructed using 8560 cells while the unstructured one contained 8378 cells.*

For the comparison between openFOAM and SU2, different domain dimensions are used as these aligned with a test case already present for openFOAM. The size of the domain varies slightly, with a shorter blade zone (0.5 vs. 1) and wider upstream and downstream zones (2.5 vs. 2). Structured mesh is used for the comparison with 4740 cells in the upstream and downstream zones and 1020 in the body force zone.

3.1.3. Single vs. Multi-Zone Mesh

Multiple zones increases complexity of the meshing process. Non-conformal mesh at the interface can introduce errors, which has been shown to happen when using unstructured mesh. A single zone mesh can be used, either with the body force defined in the zone or by specifying a range of x-coordinates over which to apply the body force. Initial tests indicate that specifying coordinates over which to apply the body force model lead to errors and will requires a more complex code structure. Furthermore, specifying both the mesh geometry and domain geometry introduces another source of error; a discrepancy between the two could lead to segmentation faults. The other option is using a single zone mesh with the body force defined in

[2] For three-dimensional setup of the body force model domain the reader is referred to Chapter 3.3 and 3.4 of Hill's dissertation [Hill, 2017]. In this case both the axial and radial direction must match with the physical blade simulation.

that zone. This introduces convergence errors due to specifying both inlet conditions and body forces at the inflow boundary. Due to these complications a multi-zone mesh is used where a single zone is selected over which to apply the body force. This takes into account future applications where a blade row is meshed as a zone, making it easy to specify where to apply the body force model.

3.2. Hall's Body Force Model

Hall's model does not require calibration, a process which would otherwise be time consuming. It also neglects viscosity, which simplifies the simulations that have to be run. This simplicity makes it an ideal model for a first-time implementation, complexity can only lead to more errors. The force is a straightforward computation similar to computing lift over an airfoil; it depends only on a number of local flow properties that are generally available in CFD solvers. Equation 2.1 shows that there are only four variables that the force depends on, which are the following:

- Flow deviation: δ
- Flow absolute (stator) or relative (rotor) velocity: W
- Blade pitch: $s = \frac{2\pi R}{B}$, where R is the radius and B the number of blades
- Tangential component of the blade camber normal: $\hat{n}_\theta$ ($\hat{n}_y$ for 2D)

When defining a blade row geometry, variables such as the blade pitch and blade camber distribution are specified as an input for the simulation. Once a run has started, the flow's velocity and direction can be determined, which are necessary to calculate the magnitude of the body force. Every grid point in the source term field requires computation of the force magnitude, which is then used as input for the equations of motion to determine the flow characteristics at subsequent data points. An iterative process continues until convergence is reached. Figure 3.4 visualizes Hall's body force in the axial-tangential (x-y) plane as well as a decomposition of the force into components to be used in the governing equations.

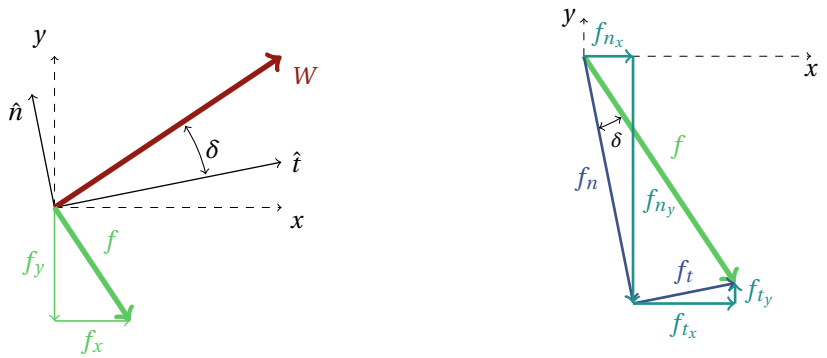


Figure 3.4: *Layout of Hall's approach to generate the body force component of his model. The diagram on the left shows the relative velocity (W) and resultant body force (f) with respect to the blade camber tangent and normal ($\hat{t}$ and $\hat{n}$). On the right a decomposition of the resultant body force is shown, first into camber tangent and normal components (f_t and f_n) and then into their x and y components (f_{t_x}, f_{n_x} and f_{t_y}, f_{n_y}).*

From the diagram it is possible to determine the two components (f_x and f_y,) that make up the body force in the governing equations (as shown in Equation 3.13). Simple geometrical relations are used to determine the camber line normal (f_n) and tangential (f_t) components, and subsequently the flow tangential (f_{t_y}, f_{n_y}) and meridional (f_{t_x}, f_{n_x}) components of each. Equations 3.1 and 3.2 show the initial decomposition, with Equations 3.3 and 3.4 representing the result of the second and third step for the tangential and normal components of the force respectively.

$$f_t = f\sin(\delta) \tag{3.1}$$
$$f_n = -f\cos(\delta) \tag{3.2}$$

$$f_{t_x} = f_t \cdot \hat{t}_x \qquad f_{n_x} = f_n \cdot \hat{n}_x \qquad f_x = f_{t_x} + f_{n_x} \tag{3.3}$$
$$f_{t_y} = f_t \cdot \hat{t}_y \qquad f_{n_y} = f_n \cdot \hat{n}_y \qquad f_y = f_{t_y} + f_{n_y} \tag{3.4}$$

Where $\hat{t}_x$, $\hat{t}_y$, $\hat{n}_x$, and $\hat{n}_y$ are the camber tangent and normal x and y-components respectively. The forces f_x and f_y can then be placed into the steady 2D Euler equations and solved to compute flow values at each grid or time point. In this two-dimensional representation no radial component is present. If a third dimension is included this is also not the case, however it is naturally captured due to the force being projected on the plane shared by the velocity and the camber normal [Thollet, 2017].

Computing flow deviation δ using W · n

Calculating the flow deviation δ requires computing the angle between the velocity vector and x-axis, and then subtracting the local angle of the camber tangent with respect to the x-axis. This step led to issues in the initial implementation, which is why a simplified computation is used. Hall originally recognized the ease of using this method and it has since been used in both CFX and openFOAM by [Hill and Defoe, 2018] at the University of Windsor. The foundations of the concept lie in simple trigonometry and the relationship between two angles at a right-angle to each other. It takes into account the changes in component directions when the angle varies between positive and negative values.

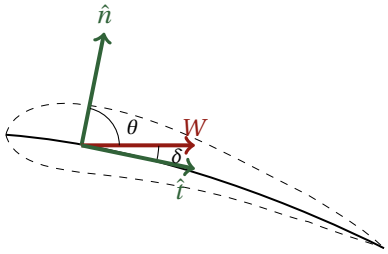


Figure 3.5: *For any arbitrary point on the camber line the angle between the normal and velocity vector θ and the deviation angle δ add up to 90 degrees. This relationship is used to compute the flow deviation δ using only the velocity vector W and camber normal n.*

Figure 3.5 visualizes the camber line of an arbitrary airfoil with the camber normal vector and velocity vector at some point on the chord. The angles θ and δ are shown as well. Coming up with the definition for δ starts with defining $W \cdot n$:

$$W \cdot n = |W||n|\cos(\theta) \tag{3.5}$$

Using the trigonometric relationship between the two angles θ and δ, which together add up to 90° ($\hat{t} \perp \hat{n}$), it is possible to come up with a replacement value for $cos(\theta)$. This can be seen in Equation 3.8.

$$\theta = 90 - \delta \tag{3.6}$$
$$\cos(\theta) = \cos(90 - \delta) \tag{3.7}$$
$$\cos(\theta) = \sin(\delta) \tag{3.8}$$

Doing so and plugging it into the original equation for the dot product $W \cdot n$ gives the final form for calculating the flow deviation δ using only the velocity and camber normal vector (Equation 3.10). This last step uses the knowledge that the magnitude of the normal vector n is equal to 1.

$$W \cdot n = |W||n|\sin(\delta) \tag{3.9}$$
$$\delta = \sin^{-1}\left(\frac{W \cdot n}{|W|}\right) \tag{3.10}$$

3.2.1. Governing Equations

The three-dimensional Euler equations for mass, momentum, and energy in Cartesian coordinates with body forces are written as

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho V_x\\ \rho V_y\\ \rho V_z\\ \rho e_t\end{bmatrix}+\frac{\partial}{\partial x}\begin{bmatrix}\rho V_x\\ \rho V_x^2+p\\ \rho V_y V_x\\ \rho V_z V_x\\ V_x(\rho e_t+p)\end{bmatrix}+\frac{\partial}{\partial y}\begin{bmatrix}\rho V_y\\ \rho V_x V_y\\ \rho V_y^2+p\\ \rho V_z V_y\\ V_y(\rho e_t+p)\end{bmatrix}+\frac{\partial}{\partial z}\begin{bmatrix}\rho V_z\\ \rho V_x V_z\\ \rho V_y V_z\\ \rho V_z^2+p\\ V_z(\rho e_t+p)\end{bmatrix}=\begin{bmatrix}0\\ F_x\\ F_y\\ F_z\\ \bar{F}\cdot\bar{V}\end{bmatrix} \tag{3.11}$$

Initial implementation of the model will be performed for two-dimensional problems only to ease the work load and test its capabilities. Getting rid of the third dimension leaves the x and y planes:

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho V_x\\ \rho V_y\\ \rho e_t\end{bmatrix}+\frac{\partial}{\partial x}\begin{bmatrix}\rho V_x\\ \rho V_x^2+p\\ \rho V_y V_x\\ V_x(\rho e_t+p)\end{bmatrix}+\frac{\partial}{\partial y}\begin{bmatrix}\rho V_y\\ \rho V_x V_y\\ \rho V_y^2+p\\ V_y(\rho e_t+p)\end{bmatrix}=\begin{bmatrix}0\\ F_x\\ F_y\\ \bar{F}\cdot\bar{V}\end{bmatrix} \tag{3.12}$$

Where $\bar{V}$ is the velocity vector with components V_x and V_y and $\bar{F}$ is the body force source term as given by

$$\bar{F}=\begin{bmatrix}F_x\\ F_y\end{bmatrix}=\rho\begin{bmatrix}f_x\\ f_y\end{bmatrix}=\rho\begin{bmatrix}f_{t_x}+f_{n_x}\\ f_{t_y}+f_{n_y}\end{bmatrix} \tag{3.13}$$

Where $\hat{n}_x$, $\hat{n}_y$, $\hat{t}_x$, and $\hat{t}_y$ are the x and y components of the unit vectors for the camber normal and tangent respectively. The magnitude of the body force f for a 2D case (simplified from Equation 2.1) is computed as:

$$f=\frac{\pi\delta W^2}{s|n_y|} \tag{3.14}$$

Where $s=\frac{2\pi R}{B}$ is the local blade pitch. Equation 3.14 can be decomposed into its x and y components using Equations 3.1 through 3.4. Extending the problem to a three-dimensional case is a matter of reverting to the original set of Euler equations in Equation 3.11 and adding the force component in the z-direction F_z. If the turbomachinery geometry is defined using cylindrical coordinates, a transformation to the Cartesian frame of reference is necessary as SU2 is written in Cartesian coordinates. Also necessary is a new decomposition of the body force that includes all three (x, y, z) force components present in the governing equations. Three-dimensional implementation is beyond the scope of this thesis and is presented in Chapter 6 as a recommendation for future work.

3.2.2. Compressibility Correction

Hall's model was originally intended to be used for incompressible flow and as such discrepancies may occur when higher pressure ratios or more modern transonic fans are modeled. In his work, [Thollet, 2017] suggested a number of modifications to improve Hall's model. He included a blockage factor to account for metal blockage effects; a parallel force to model viscous losses (drag); and a compressibility correction based on local relative Mach number to include its effect on the lift generated. This compressibility correction requires multiplying the force with a compressibility factor K, as shown in Equation 3.15.

$$f_{n_{comp}}=K\cdot f_{n_{inc}} \tag{3.15}$$

Where the value K is determined using the Prandt-Glauert rule and Ackeret formula for subsonic and supersonic flows respectively (Equation 3.16). To avoid the singularity around $M=1$ a limit of 3 is assigned to the value of K (Equation 3.17), which was shown to be the maximum value that changes the solution significantly [Thollet, 2017].

$$K'=\begin{cases}\frac{1}{\sqrt{1-M^2}}, & \text{if } M<1\\ \frac{2}{\pi\sqrt{M^2-1}}, & \text{if } M>1\\ 3, & \text{if } M=1\end{cases} \tag{3.16}$$

$$K = \begin{cases} K', & \text{if } K' \leq 3 \\ 3, & \text{if } K' > 3 \end{cases} \tag{3.17}$$

The blade blockage factor and parallel force additions created by [Thollet, 2017] and used by [Benichou et al., 2019] are not used in this work as they add complexity to an implementation that is looking at feasibility. While their adoption would improve accuracy of physical blade modeling, this is not the objective of this thesis and it is therefore not taken into account. Once it has been proven that Hall's body force model works in SU2, the addition of viscous effects and blockage has the potential for future work to improve the model's accuracy with respect to physical blade modeling.

3.2.3. Camber Normal Representation of an Airfoil

In a body force simulation, the blade is not physically modeled but is instead approximated using camber normal values. Over the domain in which the physical blade would be present, the camber normal value varies depending on either the x (2D) or x and r (3D) directions. As can be seen in Figure 3.4 and Equation 2.1 the force depends on the camber normal value, which is the unit vector normal to the camber line tangent. A camber normal distribution can be defined using a unit vector distribution over the chord and/or spanwise direction; this is done when creating arbitrary airfoils to simulate. It is also possible to create a camber normal distribution from an existing airfoil by generating the camber line and then calculating the camber normal at every location along the x or r directions.

Arbitrary Camber Line Generation

Generating arbitrary cambers for use in test cases is most easily performed by first defining a leading edge and trailing edge angle. Subsequently the local camber tangent angle is varied between these two values over the distance of the airfoil using an arbitrary distribution. For simplicity the variation in camber tangent angles over the airfoil is limited to linear distributions. Given the local camber tangent angle α (positive in the y-direction), the unit vector camber tangent is determined using the sine and cosine of the angle. In the case of the tangent: $\hat{t}_x = cos(\alpha)$ and $\hat{t}_y = sin(\alpha)$. Camber normal values are then $\hat{n}_x = -sin(\alpha)$ and $\hat{n}_y = cos(\alpha)$. Figure 3.6 visualizes the decomposition of the camber normal and tangent values. With the leading and trailing edge angles defined, it is solely a matter of determining the number of points over the chord at which to compute the camber normal. The higher this value is, the more accurate the distribution. Values of x, $\hat{n}_x$, and $\hat{n}_y$ are stored in arrays and used when calculating the force.

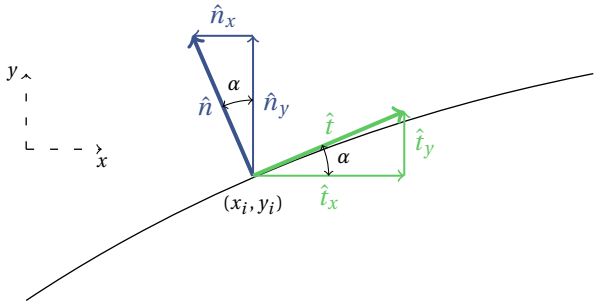


Figure 3.6: *Camber normal and tangent vector decomposition for an arbitrary point on the camber line, with α the angle between the camber tangent and the meridional (positive in the y-direction).*

Every x-coordinate with a node in the body force domain needs to have a camber normal and tangent value assigned to it, which in this case can be done using simple interpolation. For every point it is possible to find two neighboring points (in the x-direction) that possess values of $\hat{n}_x$ and $\hat{n}_y$. A simple linear interpolation (Equation 3.18) depending on the distance between the points and magnitudes of each of the variables will output the camber normal unit vector.

$$y = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \cdot (x - x_i) \tag{3.18}$$

Where y is the value to be computed for any arbitrary point x; y_i and y_{i+1} are the lower and upper boundaries of the value; and x_i and x_{i+1} the lower and upper boundaries of the x coordinate. To determine camber

normal values, y is substituted by $\hat{n}_x$ and $\hat{n}_y$, which can then be used at each node to compute the body force magnitude.

Camber Normal Representation of an Existing Airfoil

Instead of creating arbitrary airfoils, a camber normal distribution can be created that represents a physical airfoil. This allows for a comparison to be made between a physical blade simulation and its body force representation counterpart. From the physical blade, which is usually represented using lower and upper surface coordinates, the goal is to obtain a camber line that can be used to determine the local camber tangent angle. Given a set of coordinates defining an upper and lower surface on the x-y plane, the camber line can be determined by passing over the x-coordinates and at each one computing the middle point between the two y-coordinates (Equation 3.19). It should be noted that this does not take into account a thickness distribution for the airfoil. If this information is present it can be included to ensure a more realistic approximation of the camber line. With a complete camber line the local camber tangent angles can be computed in a similar method as those for the arbitrary cambers explained above. The end result is a set of arrays with x, $\hat{n}_x$, and $\hat{n}_y$ which are then used to determine the camber normal vector at each point using the interpolation method explained above.

$$y_x = \frac{y_{x_{upper}} + y_{x_{lower}}}{2} \tag{3.19}$$

Camber line representations of airfoils will never be as accurate as physical blades due to their lack of thickness distribution. This is an inherent disadvantage of the body force model and is elaborated upon in Section 4.2.5. Nevertheless, the methods presented for camber normal representations are used throughout the thesis because body force simulations are a still a reduced-order method.

Whittle Fan

One of test cases used in the thesis is that of the low-speed Whittle fan, a real-life blade of which a 2D version is created specifically as part of the verification process in this work. The Whittle fan's geometry data consists of a number of constant span fraction blade profiles from the hub to the shroud, which in the 3D case was extrapolated to create a physical blade mesh. To create a 2D blade profile a constant span fraction near the middle of the blade was taken and the set of coordinates projected onto the blade-to-blade (x-y) plane. From this the method described above was used to generate camber tangent and normal values at each x-coordinate. A visualization of the airfoil and its camber line is presented in Figure 3.7.

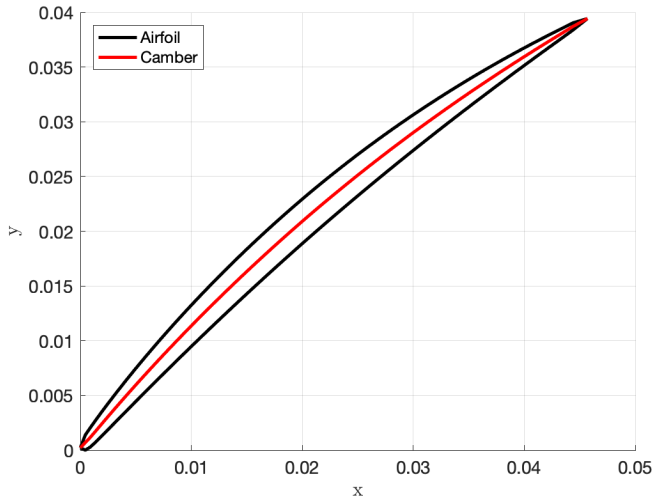


Figure 3.7: *Two-dimensional representation of Whittle fan blade at mid-span. Taken from a constant-span fraction of the 3D version, with coordinates projected onto the x-y plane.*

3.3. Implementing the Body Force Model in SU2

Two options are available for implementation in the direct solver, namely (1) the addition of the source term directly in the *CNumerics* class (while the residual is being calculated), or (2) in the *CSolver* class as the flow is solved for each zone. Besides the computation itself, the former requires the least amount of changes to the current code structure. Its drawback is the lack of "clean" code as compared to the current structure of SU2, which is an essential part of SU2's value proposition. Implementing the computation in the *CSolver* class aligns with this framework but requires adding a number of classes and functions to ensure the value of the force is correctly (re)allocated. Both have been implemented and experimented with, and due to the latter's cleanliness and alignment with SU2's standards it is chosen as the way forward for this thesis.

In this approach the body force computation is "outsourced" to a function in the *CSolver* class (see Algorithm 1), where the value of the force's components are then stored to each node. This function is called in the *CIteration* class after every solve of the flow equations. After being stored at each node, the forces are called in the *CNumerics* class and added to the residuals; thus bypassing the need to compute the force in the discretization step.

Modifying the code to include the body force model follows a structured approach that adapts all the necessary classes and follows it with checks to make sure that the code is structurally sound. Figure 3.8 shows how the *CSolver*, *CIteration*, *CVariable*, *CNumerics*, and *CConfig* are altered. For a more detailed explanation regarding how the classes are altered the reader is referred to Appendix A.3. It is important to mention that hard-coding is present in this implementation. This is because the focus is on showing the software's capability, not on perfecting usage. These hard-coded parts will need to be removed to ensure SU2 works for arbitrary configurations and does not require recompiling before running the executables.

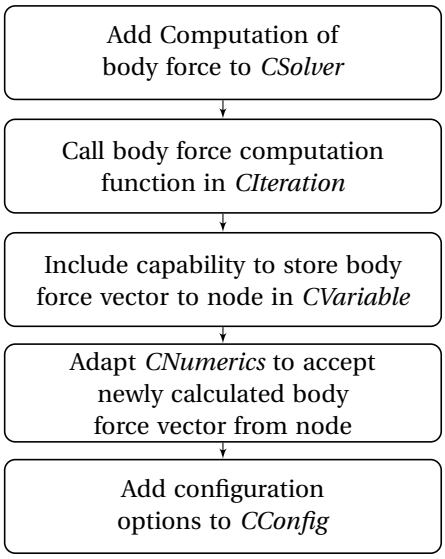


Figure 3.8: *Code adaptation strategy for direct solver.*

After completing the steps in Figure 3.8 all the adapted code is reviewed using the following procedure:

1. Confirm functions and variables are defined and initialized in *_structure.cpp*, *.hpp*, and *.inl* files
2. Check code is conform SU2 standards
3. Add comments clearly stating operations
4. Compile and check for errors

Function Calculating the Body Force

There is one function in particular, in the *CSolver* class, that is most important to get right, namely the computation of the body force itself. Algorithm 1 presents a high-level impression of the function that performs the calculation in SU2. Once the force is computed and stored to the node on which it acts, it is loaded in the *CNumerics* class to calculate the source term residual.

Algorithm 1 Computation of body force in *CEulerSolver::ComputeBodyForce_Turbo*

procedure COMPUTEBODYFORCE_TURBO
  Initialize flow conservative (U_i) and primitive (V_i) variables and node coordinates (x_i)
  Load configuration options γ, R_{gas}, B, Ω, R
  Initialize $\boldsymbol{F_{BF}}$, ωR, s
  Define camber normal arrays $\boldsymbol{x}$, $\hat{\boldsymbol{n}}_{\boldsymbol{x}}$, and $\hat{\boldsymbol{n}}_{\boldsymbol{y}}$
  for all Points **do**
    Load and set flow conservative (U_i) and primitive variables (V_i) and node coordinates (x_i)
    Interpolate to find value of $\hat{n}_x$, $\hat{n}_y$, $\hat{t}_x$, and $\hat{t}_y$
    Compute $f = K\pi\delta\frac{1}{s}V^2\frac{1}{N_y}$
    Decompose into t, n, x, and y components
    Add x and y components to $\boldsymbol{F_{BF}}$
    Set $\boldsymbol{F_{BF}}$ to node at Point
  end for
end procedure

3.4. Extension of SU2's Adjoint Solver to Body Force Modeling

The practical and theoretical workings of the adjoint solver are covered in this section, with emphasis on computation of the mesh sensitivity and total gradient. Firstly, the focus lies on registering body force source terms so that the mesh sensitivity can be calculated. Next, total gradient computation is covered, which entails replacing grid points with the vector of camber normals. How the source code must be adapted is presented, which elaborates on the necessary changes required for the mesh sensitivity and total gradient to be computed when body force source terms are added to the conservative variable vector $\boldsymbol{U}$.

3.4.1. Adjoint Solver in SU2

When performing an AD build of SU2, the external library CoDiPack performs automatic differentiation of the routines in SU2 needed to build the adjoint solver. Besides the standard executables a number of AD ones will be available, such as `SU2_CFD_AD` and `SU2_DOT_AD`. These are then run to perform an adjoint calculation on the problem at hand. Running an adjoint simulation works as follows. First the direct solver is called using `SU2_CFD`, which runs until a converged solution is obtained. Next, `SU2_CFD_AD` runs the adjoint solver to compute the mesh sensitivities $\frac{dJ}{d\boldsymbol{X}}$.[3] Finally, `SU2_DOT_AD` projects the mesh sensitivities to the surface, with the output being $\frac{dJ}{d\boldsymbol{X}_{surf}}$. All three executables can be run individually, however the python script `discrete_adjoint.py` automates this process and performs all three sequentially.

Where direct solver body force implementation focuses on running `SU2_CFD` successfully, the adjoint solver must run `SU2_CFD_AD` correctly now that body forces are included in the simulation. Figure 3.9 shows the steps taken by the discrete adjoint driver class to run a simulation.

Two equations need to be computed, namely the flow adjoint (Equation 3.20) and mesh sensitivity (Equation 3.21) equations, where $\boldsymbol{\Psi}$ is the adjoint flow vector.[4] Starting with $\boldsymbol{U}$ as input, the adjoint solver performs a single iteration starting from the last iteration ($maxiter$) performed by the direct solver. Flow and convergence data is acquired and stored. SU2 then initializes adjoint values in auxiliary solution vectors to be used for the next step. Computation of the adjoint is done using CoDiPack, which performs gradient computations using Algorithmic Differentiation (AD). After the flow adjoint is calculated the mesh sensitivity is computed, which follows the same process. The sensitivities are stored at each node and can be evaluated.

$$\tilde{\boldsymbol{U}} = \boldsymbol{\Psi} = \left[\left(\frac{\partial J}{\partial \boldsymbol{U}}\right)^T + \left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right)^T \boldsymbol{\Psi}\right] \tag{3.20}$$

$$\tilde{\boldsymbol{X}} = \left(\frac{dJ}{d\boldsymbol{X}}\right)^T = \left[\left(\frac{\partial J}{\partial \boldsymbol{X}}\right)^T + \left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right)^T \boldsymbol{\Psi}\right] \tag{3.21}$$

[3] Detailed information regarding the code structure of the adjoint solver in SU2 can be found in [Fernández, 2017].

[4] Both $\tilde{\boldsymbol{U}}$ and $\boldsymbol{\Psi}$ are used interchangeably for the adjoint flow vector. Similarly, $\tilde{\boldsymbol{X}}$ and $\frac{dJ}{d\boldsymbol{X}}$ both represent the mesh sensitivity.

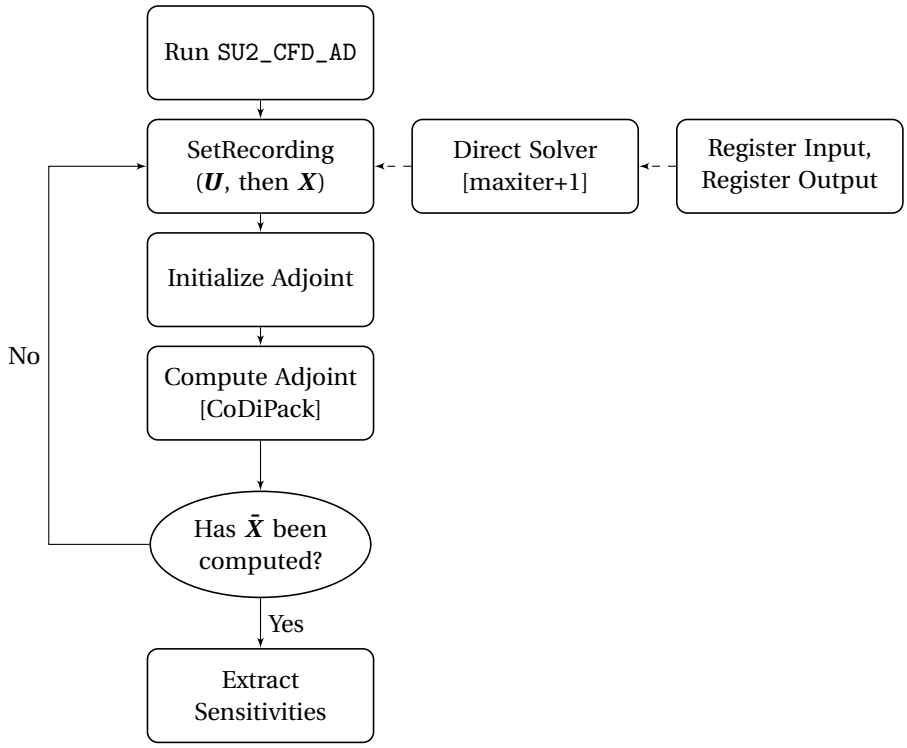


Figure 3.9: *Structure of CDiscAdjFluidDriver::Run, showing the steps that are taken to compute the flow and mesh adjoint. This process follows a converged direct solver run of* maxiter *iterations.*

Calculating Mesh Sensitivity

For an optimization problem in the field of turbomachinery it is the aim to minimize the objective function J, which is a function of the flow $\boldsymbol{U}$ and mesh $\boldsymbol{X}$. Both of these depend on the set of design variables $\boldsymbol{\alpha}$.

$$J(\boldsymbol{U}(\boldsymbol{\alpha}), \boldsymbol{X}(\boldsymbol{\alpha})) \tag{3.22}$$

The problem is subject to:

$$\boldsymbol{U} = \boldsymbol{R}(\boldsymbol{U}, \boldsymbol{X}) \tag{3.23}$$

$$\boldsymbol{X} = \boldsymbol{M}(\boldsymbol{\alpha}) = \boldsymbol{V}(\boldsymbol{S}(\boldsymbol{\alpha})) \tag{3.24}$$

Where $\boldsymbol{V}$ and $\boldsymbol{S}$ are the volume and surface deformation respectively. Given the output of SU2's adjoint solver, namely the mesh sensitivity $\bar{\boldsymbol{X}}$, the total gradient of the objective function with respect to the design variables can be rewritten as shown in Equation 3.25. The term $\frac{d\boldsymbol{M}^T}{d\boldsymbol{\alpha}}$ is the product of the surface mesh and geometric sensitivities.

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{d}{d\boldsymbol{\alpha}} \boldsymbol{M}^T(\boldsymbol{\alpha}) \bar{\boldsymbol{X}} \tag{3.25}$$

A body force simulation does not require mesh deformation because there is no physical blade to model. It is therefore important to focus on computing the mesh sensitivity $\bar{\boldsymbol{X}}$ when the source terms are added to the vector of conservative variables $\boldsymbol{U}$. The adjoint equations must be calculated such that Equations 3.20 and 3.21 output a new mesh sensitivity. Source terms are added to the computation using the *AD::RegisterInput* function (see Figure 3.9). Further code adaptation is necessary and is presented in Section 3.4.2.

Calculating Total Gradient

SU2's adjoint solver is structured in such a way that when *SetRecording* is called, its input defines the calculation that is made. First the conservative variables $\boldsymbol{U}$ are used to implicitly obtain the flow adjoint $\bar{\boldsymbol{U}}$ (Equation 3.20), which is then used to explicitly compute the mesh sensitivity $\bar{\boldsymbol{X}}$ (Equation 3.21) when

the grid points $\boldsymbol{X}$ are taken as input. What the solver essentially does is take this input and use CoDiPack to register each conservative variable or grid point as input for the solver. When `SU2_CFD_AD` is run, the flow adjoint equation is computed first (see Figure 3.9). This is the term $\frac{\partial J}{\partial \boldsymbol{U}}$ in Equation 3.26. With this the mesh sensitivity is computed as the product of the terms $\frac{\partial J}{\partial \boldsymbol{U}} \cdot \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{X}}$, namely $\frac{dJ}{d\boldsymbol{X}}$. The surface mesh sensitivity $\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{X}_{surf}}$ and geometric sensitivity $\frac{d\boldsymbol{X}_{surf}}{d\boldsymbol{\alpha}}$ are computed using AD. The final gradient is a product of these four terms.

Body force simulations have a non-varying mesh, which means the objective function loses its dependence on the volumetric grid points ($J[\boldsymbol{U}(\boldsymbol{\alpha})]$). Any terms in Equation 3.26 relating to the mesh are not taken into account. The flow's sensitivity to the mesh is replaced by the flow's sensitivity to the design variables, which in this case are the camber normal values. Equation 3.27 shows this concept. Similar to before, the flow adjoint is computed, however now it is used to compute the total gradient when α (instead of X) is used as input for *SetRecording*.

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{U}} \cdot \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{X}} \cdot \frac{\partial \boldsymbol{X}}{\partial \boldsymbol{X}_{surf}} \cdot \frac{d\boldsymbol{X}_{surf}}{d\boldsymbol{\alpha}} \tag{3.26}$$

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{dJ}{d\boldsymbol{U}} \cdot \frac{d\boldsymbol{U}}{d\boldsymbol{\alpha}} \tag{3.27}$$

A prerequisite for this to be performed is successful registration of body force source terms and computation of the mesh sensitivities. The code can subsequently be adjusted so that *SetRecording* accepts a vector of camber normal values as input. This requires changing a number of files in the source code, a detailed explanation of which is given in Section 3.4.3.

Choosing an Objective Function

SU2 contains numerous objective functions that can be used for optimization, some examples being entropy generation, total pressure loss, or Eulerian work. Running a simulation with one of these requires a physical blade to be present in the domain. If not, the results of the mesh sensitivity and total gradient computation will be incorrect. In Appendix B.1 a comparison between three objective functions is presented. A custom objective function must be created that evaluates average flow values at a marker (for example at the outflow of the body force domain). To circumvent making structural changes to the source code, one of the current objective functions is altered. In the function *CEulerSolver::Pressure_Forces*, the computation for drag is modified. For the marker specified in the configuration file, the velocity in y-direction is summed over all the points and this value taken as J. This computation is shown in Equation 3.28, where N is the number of points on the marker. Detailed information regarding code adaptation is given in Section 3.4.3.

$$J = \sum_{i=0}^{N} V_{y_i} \tag{3.28}$$

3.4.2. Adjusting Source Code for Sensitivity Calculation

Without any previous work on the implementation of a body force model in an adjoint solver, adaptation of the source code is a challenge. However, [Rubino, 2019] implemented a similar concept in his dissertation, adding a harmonic balance capability to SU2's solver. While the two models differ, their implementation with respect to adding a source term to the mesh sensitivity calculation is comparable. It is therefore used as a basis for the current work. Hard-coding is used for this part as the focus of this thesis is on showcasing a capability rather than code generalization. Essential changes that need to be made are as follows:

Register Input/Output Before computing the adjoint it is necessary to register new source terms as inputs and outputs, which is performed in the *SetRecording* class by the functions *RegisterInput* and *RegisterOutput* (see Figure 3.9). A conditional statement is added to the functions *CDiscAdjSolver::RegisterSolution* and *CDiscAdjSolver::RegisterOutput* for the body force source terms. The function *CVariable::RegisterBFSource* is added, which physically registers the source term from the direct solution to each node. This is possible because the body force is stored to the node in the direct solver, which is shown in the last step of Algorithm 1.

Initialize Adjoint Once the source terms have been registered, they have to be initialized before they can be differentiated using CoDiPack. In the function *CDiscAdjSolver::SetAdjoint_Output* a conditional statement is included for if a body force is used in the simulation. If this is the case it loops over all points and for each node of the mesh initializes the source term and its derivative value.[5] Two functions are created to perform this process. *CDiscAdjVariable::GetAdjoint_BFSource* is called on an auxiliary solution vector, storing the direct value. *CEulerVariable::SetAdjoint_BFSource* then initializes the derivative. Computation of the adjoint is performed on the auxiliary vector while the original object remains untouched.

Iteration After AD, the resulting derivative is extracted from the auxiliary solution vector and copied to a node-defined vector, allowing the auxiliary solution vector to be used for the next iteration. Similar and opposite to the process during initialization, *CDiscAdjSolver::ExtractAdjoint_Solution* sets the derivative obtained from AD to the auxiliary vector using *CEulerVariable::GetAdjoint_BFSource*. After this the result is copied to the node's body force source adjoint vector using *CDiscAdjVariable::SetAdjoint_BFSource*.

Custom Objective Function In the function *CEulerSolver::Pressure_Forces*, a variable is added to store the velocity in the y-direction. In the conditional loop when the values are calculated at the specified boundary, the velocity's y-component is obtained from the node and stored to this variable. All values of the velocity at the boundary are added together so they are weighted equally. The value of *CD_Inv[iMarker]* is set equal to this cumulative velocity, replacing computation of drag. It is a hard-coded implementation and as such should be reviewed before being used in future work.

3.4.3. Adjusting Source Code for Total Gradient Calculation

Due to the author's limited experience with C++ and knowledge of SU2's code structure, total gradient computation was not successfully implemented during this thesis. However, it is clear what steps need to be taken to make it possible. Conceptually, adapting the source code is a matter of replacing the grid point vector $\boldsymbol{X}$ with an array consisting of camber normal definitions at each x-location (essentially the design variables $\boldsymbol{\alpha}$). Key parts of the code that need to be changed are as follows:

Driver In *CDiscAdjFluidDriver::Run* when the geometrical sensitivities are being computed, the input for the function *SetRecording* should be changed to the design variables vector (ex. CAMB_NORM). This will tell the driver that the sensitivity being computed is taking the camber normals as input.

Option In the file option_structure.hpp under the ENUM_RECORDING function, the option CAMB_NORM has to be defined. This is to make sure that it can be used in the driver class.

Iteration A new condition must be added to *CDiscAdjFluidIteration::RegisterInput* that calls a function registering the camber normal array as input for CoDiPack. Given that the camber normals will be read as an input file (similar to an inlet velocity profile), this function should be placed in the *CConfig* class.

[5] SU2's in-house object type *su2double* includes both a direct and derivative value. This allows derivatives of each value to be stored in the same object, easing adjoint computation.

Config A function must be created that loops over the camber normals array (as read from the file) and registers them as input using *AD::RegisterInput*. Once they are registered the adjoint computation will be performed on this array and a derivative value will be loaded into the derivative part of the *su2double* object. This can then be used to extract the total gradient with respect to each camber normal component.

Solver Once calculated, the sensitivities (in this case the total gradient with respect to each design variable) must be extracted. A function must be added that loops over the design variable vector and extracts the derivative value, which can then be used for evaluation after the solver has finished running.

With this information, further research can be initiated that will eventually lead to performing optimization using the output of this total gradient. Without the need for mesh sensitivity calculation, the speed at which optimization can be performed will be shorter as well. The steps presented above are not exhaustive, and further work on the code may present new challenges that have to be overcome in order to be successful in computing the gradient.

3.5. Key Takeaways

The following points highlight the principal outcomes of this chapter.

- *Multi-zone meshes are used to easily classify in which zone the body forces should be active.*
- *Body forces are determined using local flow characteristics and camber normal values interpolated at every node. They are then decomposed into x and y-components and added as source terms to the governing equations.*
- *Computation of the forces is included in a function in the CSolver class, which then stores them at each node for later use in the CNumerics class of the direct solver. They can then be called in the adjoint solver as well.*
- *Computing mesh sensitivity when using body forces requires registering the forces as input and output, initializing them in an auxiliary solution vector, and then storing the derivative to each node.*
- *Computing the total gradient directly from the solver is possible when body forces are used because the objective function's dependency on the grid is removed. A vector of camber normals replaces the volumetric grid points. Registering the vector as input should output the value of total gradient.*

This page is intentionally left blank.

4

Case Studies

This chapter covers all of test cases in chronological order, which follows their complexity. First, Section 4.1 gives an overview of this as well as describing the simulation setup. Section 4.2 presents all the test cases that have been run. This ranges from a non-cambered stator to a Whittle fan simulation. The chapter is concluded with a computational time comparison between body forces and blade simulations in Section 4.3.

4.1. Test Cases and Simulation Setup

Six test cases, each more complex than the previous one, allows for step-wise complexity increments to test the model's ability to reproduce flow. All of the runs performed are presented in Table 4.1.[1] Once completed the results are obtained by extracting data using Tecplot. For body force simulations a precise line is extracted over the entire domain in the meridional direction, lying at the center of the y-axis. For physical blade simulations a number of points on the meridional axis is taken and at each all the flow values are averaged over the y-direction. Code verification results are presented in the order shown in Table 4.1. For the computational time analysis is presented, the non-cambered stator test case is the same, however an unstructured mesh is used. The Whittle test case is the same as in Table 4.1.

Table 4.1: *Test case overview and specifications for body force implementation in direct solver.*

Test Case	**Mesh Type**	**Specifications**
Stator Non-Cambered	Structured	$R = 1$, $\alpha = [1,2,5,10,20]$, $B = [1,5,10,20]$, $\boldsymbol{V}_{in} = (34.05,0,0)$
Stator Cambered	Structured	$R = 1$, $B = [1,5,10,20]$, $\boldsymbol{V}_{in} = (34.05,0,0)$
Rotor	Structured	$R = 1$, $\alpha = [10,20,30,40,50]$, $\Omega = [100,200,500,1000,1500]$, $\boldsymbol{V}_{in} = (34.05,0,0)$
Stator Comparison	Unstructured	Cell count reduced for BF, $R = 1$, $\boldsymbol{V}_{in} = (34.05,0,0)$
Whittle	Unstructured	$R = 0.1625$, $\Omega = [1803,2362,2898]$, $\boldsymbol{V}_{in} = (44.24,0,0)$, cell count reduced for BF
openFOAM	Structured	Rotor: $B = 23$, $\Omega = 150$, $R = 1.5$, $\boldsymbol{V}_{in} = (17.03,0,0)$ Stator: $B = 23$, $R = 1.5$, $\boldsymbol{V}_{in} = (17.03,0,0)$

For all simulations the configuration options were similar if not the same. Presented in Table 4.2 are a number of options that were identical for all simulations, which can be used to replicate the work presented in this chapter. Detailed configuration options for each test case can be found on the author's GitHub repository (see Section A.1).

4.2. Code Verification

Before being able to run simulations the code must be checked for errors. As SU2 is written in C++, the code cannot be run in sections and the source code has to compile completely before the executable can

[1] Complete configuration files for every simulation run can be found in the author's GitHub repository (see Section A.1), which will be made publicly available.

Table 4.2: *Common simulation configurations for direct solver test cases.*

Setting	Value
Solver	Euler
Type	Compressible
Gradient numerical method	Green Gauss
CFL Number	5
Linear solver	FGMRES
Linear solver error	1e-6
Linear solver iterations	5
Convective numerical method	JST
Time discretization	Implicit Euler

be run again. Compilation and the IDE CLion both help identify errors in syntax before running the code with a configuration file. It is checked if variables are defined, initialized, and are of the correct type. When compilation is successful the code runs with a configuration file. If there is a segmentation fault, which occurred a number of times during the thesis, then a debugger such as GDB and thorough analysis of the code must be performed to find errors. Only once the code runs successfully can verification of the model take place.

Verification of this implementation requires confirming that the model used in SU2 behaves according to the mathematical definition of Hall's body force model. Validation of Hall's model is also required: confirmation is necessary that the model is an accurate representation of reality. Hall's body force model has been used in the past and is assumed validated [Hall, 2015] [Hall et al., 2017] [Benichou et al., 2019] [Hill and Defoe, 2018]. For this reason it is beyond the scope of this thesis to dive deeper into validation of Hall's BFM as a representation of a blade row.[2] What remains is the need to assess the implementation of Hall's body force model in SU2.

While the theoretical concept does not vary between solvers there will always be a discrepancy in results due to varying numerical methods and unique code structure. Criteria used for assessment are:

- Flow angle
- Static pressure
- Total pressure
- Total relative pressure
- Total enthalpy
- Total relative enthalpy
- Conservation of mass

Changing a blade row's variables should coincide with the expected change in flow characteristics: for example a decrease in number of blades will decrease the force and therefore turning of the flow. It is important that these trends are visible in the results when comparing body force and physical blade simulations.

4.2.1. Non-Cambered Stator

Considering the base case as a domain without applying any body forces, this test case is the next step up and is simply a flat plate at an angle to the flow; essentially a stator blade row. It gives an initial indication of the model's ability to deflect the flow and alter its characteristics over the body force domain. First it is essential to visualize the results using post-processing software, which for this thesis was Tecplot. Figure 4.1 shows the visualization of a stator blade row with $\alpha = 5$, $B = 20$, and $R = 1$. An initial inspection of the flow shows that it is being physically adapted similarly to what is expected in practice (a physical blade comparison is presented in Section 4.2.2); the flow deflects in the direction of the flat plate and only varies over the body force domain. This confirms the model's capability of deflecting flow in a manner consistent with theory.

Before evaluating the accuracy of this model's implementation, basic turbomachinery thermodynamics is quickly revisited. Total pressure over a stator stays constant as there is no work being done by the blade row on the flow. As the blade row is non-rotating the total relative pressure is the same as the total pressure and is therefore not analyzed. With a constant total pressure and an increasing velocity magnitude due

[2] For an extensive explanation regarding conception and evaluation of Hall's body force model the reader is referred to [Hall, 2015].

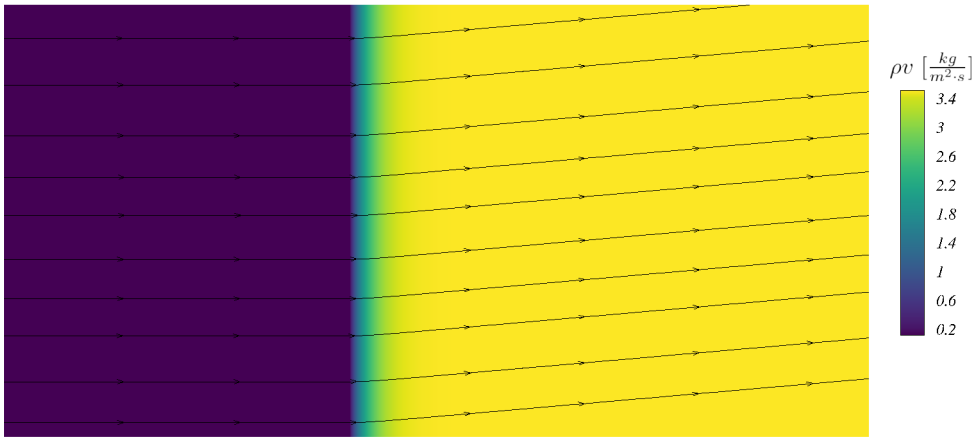


Figure 4.1: *Flow visualization of body force representation of a flat plate stator test case at $\alpha = 5$, $B = 20$, and $R = 1$. Contours represent y-momentum over the entire domain, with largest change occurring in the body force zone. Streamlines indicate the flow direction and coincide with what is expected from theory.*

to flow deflection (and subsequently an increase in the tangential velocity component), the static pressure decreases over the stator domain. Non-dimensionalization is crucial to be able to compare numerous cases: p/p_t is used to compare pressures. Again, as the blades are not rotating no work is being done on the flow, meaning that the total enthalpy and total relative enthalpy is zero. With these behaviors in mind it is possible to investigate the implementation of Hall's body force model for a stator blade row.

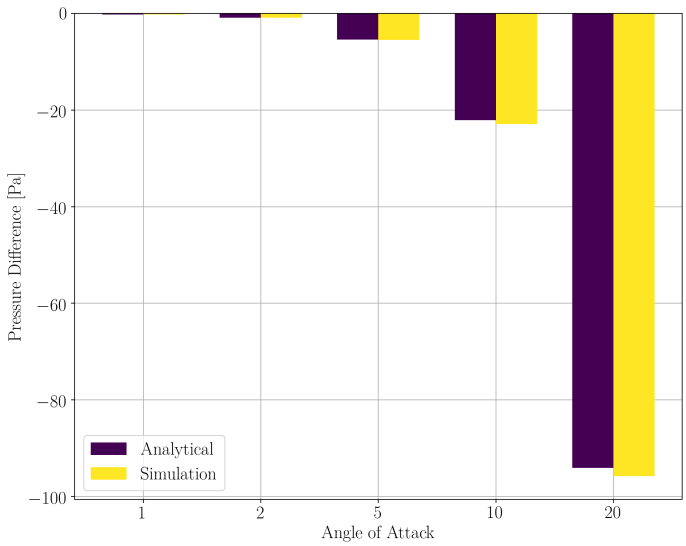


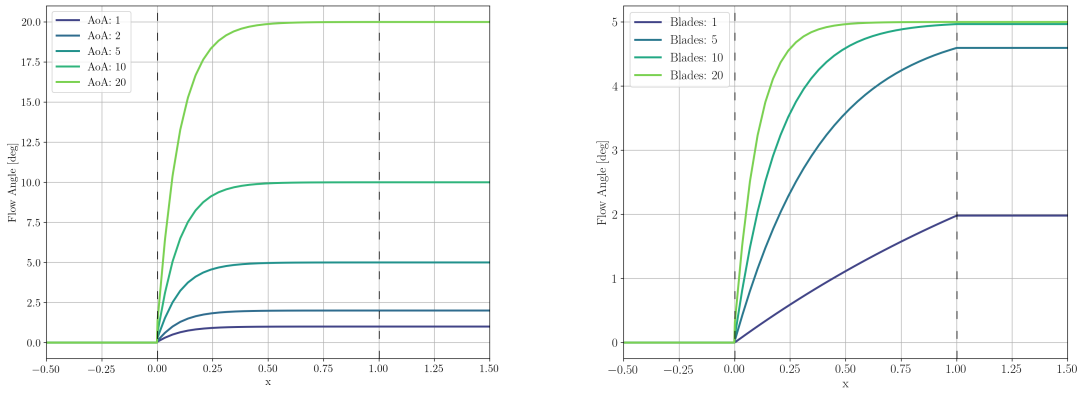
Figure 4.2: *Comparison between analytical calculations (Equation 4.1) and simulation values for static pressure loss for flat plate stator test case ($B = 20$, $R = 1$). As the angle of attack of the plate increases the pressure loss rises, which is consistent with theory. Error percentages are on the order of 1%-6%.*

A sanity check is performed by comparing analytical calculations and simulation values of static pressure change over the body force domain. Equation 4.1 computes the pressure difference over the stator, which assumes incompressible flow. Differences in density are less than 0.01%, which is why this is done. The magnitude of the absolute velocity at the exit V_2 and its tangential component V_{t_2} are given in Equations 4.2 and 4.3 respectively. Figure 4.2 presents the results for the five stator angles, showing that the discrepancy between the two results is on the order of 1%-6%. There is no relationship between error percentage and angle of attack. If larger angles of attack are used this analysis should be performed again to confirm their accuracy.

$$\Delta p = \frac{1}{2}\rho(V_2^2 - V_1^2) \tag{4.1}$$

$$V_2 = \sqrt{V_1^2 + V_{t_2}^2} \tag{4.2}$$

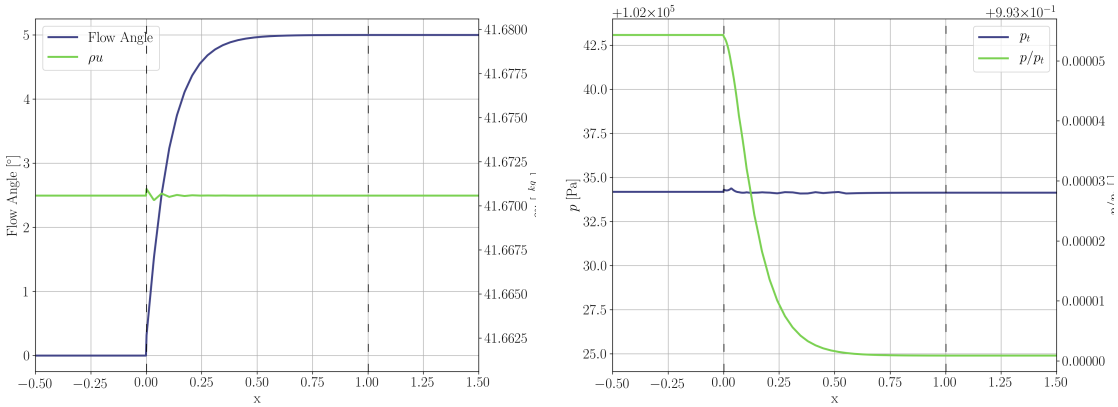
$$V_{t_2} = tan(\alpha_2) \cdot V_1 \tag{4.3}$$



(a) *Varying the angle of attack α*

(b) *Varying the number of blades B, α = 5*

Figure 4.3: *Two figures showing the characteristics of the flow when varying either the angle of attack of the flat plate or the number of blades of the blade row. As the angle of attack or number of blades is increased, the flow deflection becomes larger, corresponding to a larger applied force (Equation 3.14). Vertical dashed lines indicate the body force domain.*



(a) *Flow angle and momentum in x-direction*

(b) *Total pressure and pressure ratio*

Figure 4.4: *A number of flow characteristics for the body force representation of a flat plate stator at α = 5, B = 20, and R = 1. Vertical dashed lines indicate the body force domain.*

When changing variables that the force magnitude depends on, such as B and δ (Equation 2.1), there should be a change in flow behavior. This is confirmed in Figure 4.3, which shows larger flow deflection as the number of blades or the angle of attack of the stator is increased. It is therefore established that the model's implementation is capable of changing the way it varies the flow. Mass conservation is checked in Figure 4.4a, which shows that this is the case for the stator test case. Over the domain there should be no variation in total pressure due to there being no work done on the flow. Flow is deflected tangentially, which increase the magnitude of the velocity with respect to fully-axial inflow, and this leads to an increase in the dynamic pressure portion of the total pressure. Pressure ratio should therefore decrease over the body force domain. Both the total pressure and pressure ratio behaviors are confirmed in Figure 4.4b.

4.2.2. Non-Cambered Stator: Comparing Body Force with Physical Blade

Although validation of the model has already been performed by [Hall, 2015] and [Gunn and Hall, 2014], a trivial test is performed to confirm this for the model's implementation in SU2. A symmetrical (flat camber line) NACA0012 airfoil at $\alpha = 5$ is simulated in SU2 at the same operating conditions as the body force representation. This is visualized in Figure 4.5, showing similar flow behavior between the two cases.

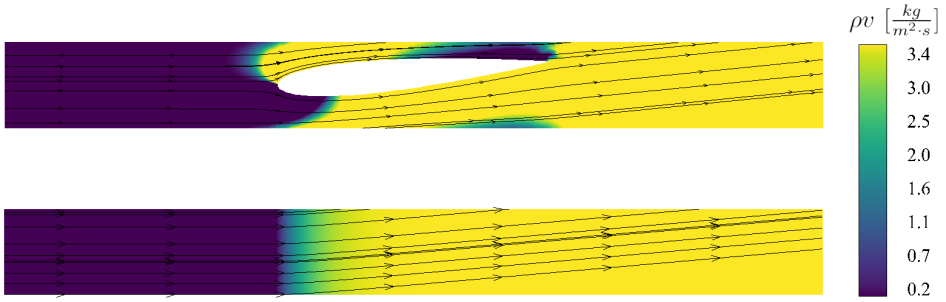
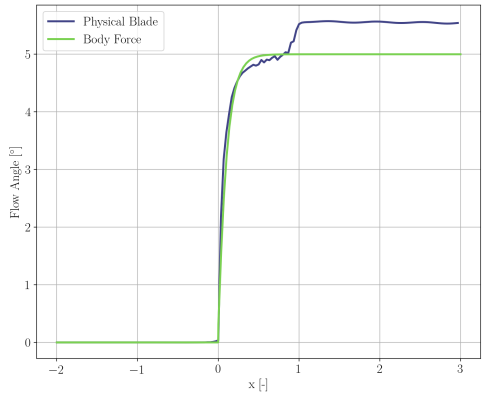
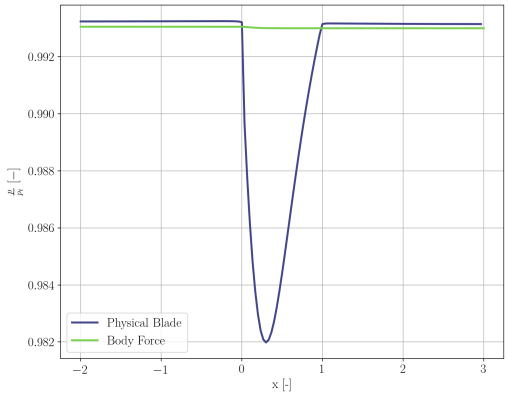


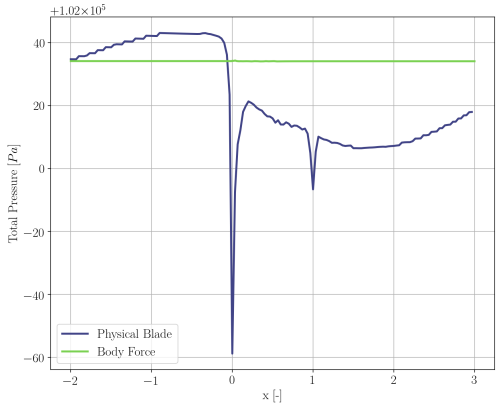
Figure 4.5: *Comparison of flow result between physical blade (top) and body force representation (bottom) for flat plate stator at $\alpha = 5$, $B = 20$, $R = 1$. The body force representation lacks effects such as upstream influence, pressure variation over the blade domain, and trailing edge instability. Flow at the domain exit is highly comparable in terms of flow deflection, pressure, and total pressure, with mass flow being conserved.*



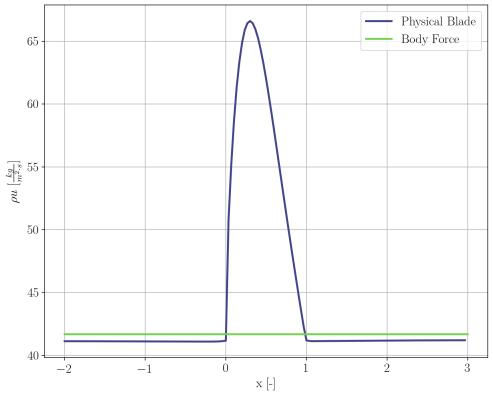
(a) *Flow angle*



(b) *Pressure ratio*



(c) *Total pressure*



(d) *Momentum in x-direction*

Figure 4.6: *Comparison between physical blade and body force simulation of a number of flow characteristics for the non-cambered stator at $\alpha = 5$, $B = 20$, and $R = 1$. The physical blade simulation uses a NACA0012 airfoil while the body force representation uses a flat camber line.*

Results are compared between physical blade and body force representation, which are presented in Figure 4.6. Despite a difference of 10.1%, the trend of the flow angle over the body force domain is accurately represented by the body force, which can be seen in Figure 4.6a. Furthermore the pressure ratio over the domain follows a similar pattern, with a small decrease present over the body force zone. Noticeable is the "spike" present for the physical blade, which is due to the large variations in pressure between the upper and lower surfaces of the airfoil. A difference in 0.02% is observed between the two runs. As Figure 4.6b shows, the most important aspects are the differences between physical blade and body force values in the upstream and downstream zones.

Except for the numerical error present in the physical blade simulation, leading to a difference of 0.02%, the total pressure stays constant (note the scale of the y-axis). Noteworthy is the flatness of the total pressure recorded in the body force simulation when compared to the physical blade (Figure 4.6c). Section 4.2.7 will present the presence of numerical error, however its magnitude is so low that it is difficult to make out when compared to the physical blade results. A final characteristic is the conservation of mass flow, which is conserved and visualized in Figure 4.6d. There is a spike in the graph that is attributable to the variation in velocity and density over the airfoil. The value analyzed in this graph is the difference between the body force and physical blade values in the upstream and downstream zone, which is only 1.1%. Mass flow is indeed conserved for both simulations and the difference between the two is negligible given the scale of the y-axis.

4.2.3. Cambered Stator

The geometry used in the last two cases was elementary and does not represent the complex shapes seen in turbomachinery rotors or stators, which is why the next step is to show the model's ability to vary flow over a cambered airfoil. The camber line of the cambered stator used for this analysis is presented in Figure 4.7. Figures 4.8a and 4.8b present a comparison between the cambered and non-cambered stator simulations run in SU2.

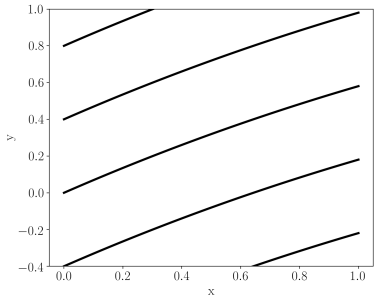
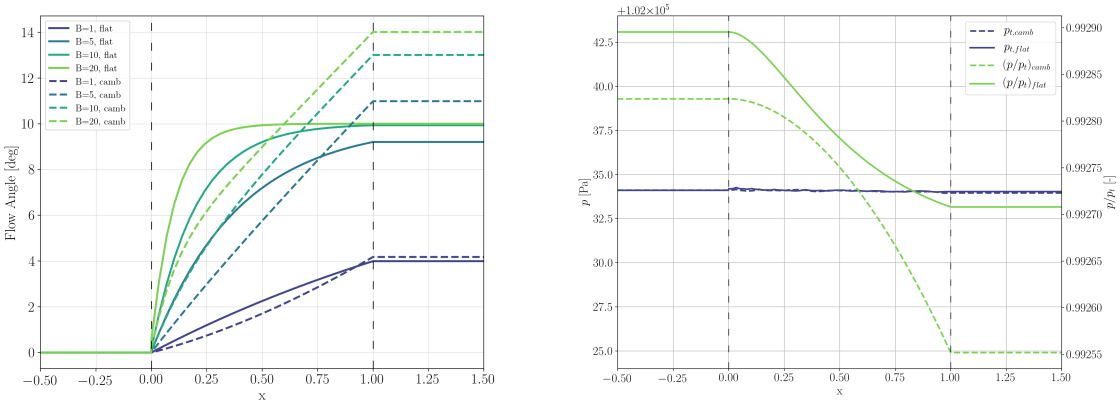


Figure 4.7: *Camber line representation of the blade row used for the cambered stator test case.*



(a) *Flow Angle* ***(b)*** *Total pressure and pressure ratio*

Figure 4.8: *Flow deflection, total pressure, and pressure ratio comparison between a cambered (dashed) and non-cambered (solid) stator. Cambered stator α varies from 5° at LE to 10° at TE. Non-cambered stator is at α = 10. Behavior aligns with what is expected from Hall's model.*

Flat plates at a constant angle with respect to the meridional will initially make the flow turn fast (as the body force model works to decrease the flow deviation δ to zero) and subsequently slower as δ is decreased. Solid lines in Figure 4.8a show this. With varying camber, δ starts closer to zero but has an almost constant (but lower) force over the body force domain. This leads to a constant change in flow angle, which is visualized by the dashed lines in Figure 4.8a. A direct effect of this is a slower increase in the y-component of the velocity, which changes the rate at which the dynamic pressure increases. Figure 4.8b sees the pressure ratio p/p_t vary more slowly over the first quarter of the domain as compared to a flat stator. The absolute difference in value is due to both the difference in camber as well as a higher TE angle for the cambered airfoil. This behavior confirms the code and model's capability to work with cambered blades.

4.2.4. Non-Cambered Rotor

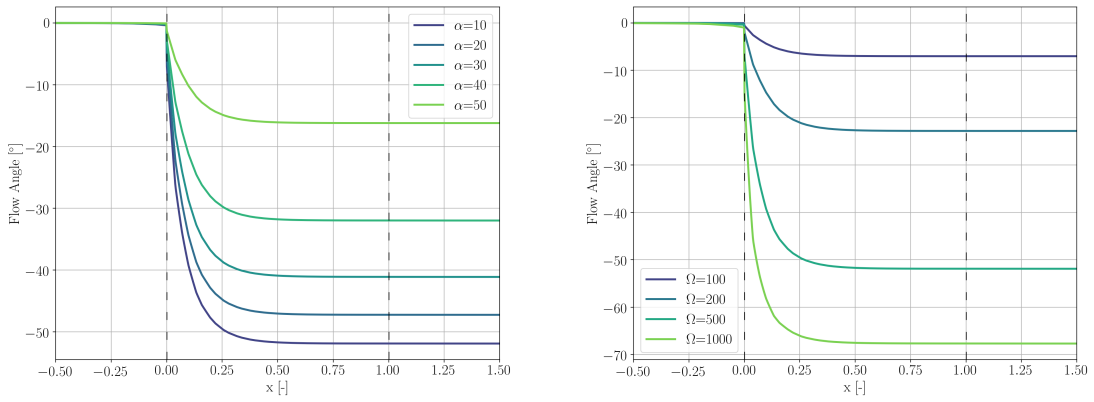
Rotation introduces a tangential velocity component, which needs to be subtracted from the meridional component to produce the relative velocity vector. Equation 4.4 computes the tangential velocity component U, where Ω is the rotational speed in RPM and R is the radius in meters. At every node, the tangential velocity is added to the y-component of the absolute velocity (see Equation 4.5), which is loaded using the conservative variables y-momentum (ρv) and density (ρ). The x-component of the relative and absolute velocity are the same ($W_x = V_x$). This relative velocity W (Equation 4.6) is then used to compute the body force, which takes into account rotation and will reproduce flow through a rotor.

$$U = \frac{\Omega}{60} \cdot 2\pi R \tag{4.4}$$

$$W_y = \frac{\rho v}{\rho} - U \tag{4.5}$$

$$W = \sqrt{W_x^2 + W_y^2} \tag{4.6}$$

At a constant rotational speed, with the tangential velocity U remaining constant, increasing the angle of attack directly influences the relative velocity vector W. Higher angles of attack lead to lower flow deflection, which is confirmed in Figure 4.9a and visualized using velocity triangles in Figure 4.11. Similarly, if W remains constant and U increases due to higher Ω, then the deflection increases; this is confirmed in Figure 4.9b.

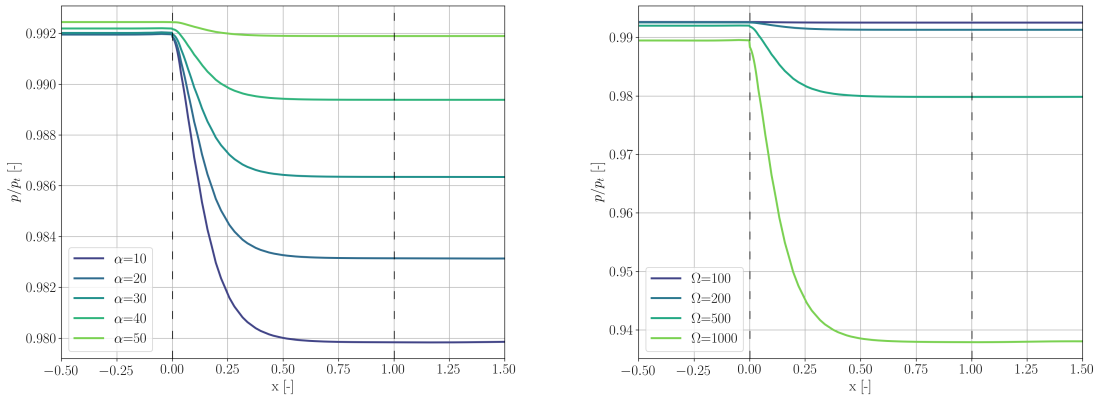


(a) *Varying α, $\Omega = 500$*

(b) *Varying Ω, $\alpha = 10$*

Figure 4.9: *Variation in absolute flow angle over the simulation domain for a body force representing a non-cambered rotor with $B = 20$ and $R = 1$. Angle of attack of the blade and rotational speed are varied.*

Behavior of the pressure ratio follows directly from the variation in flow deflection and total pressure rise for both varying α and Ω. As α is reduced or Ω increased (while the other stays constant), the total pressure rise increases due to larger flow deflection and work done on the flow. The dynamic pressure component increases due to a larger relative velocity magnitude. While the static pressure of the flow does increase, this is smaller than its dynamic counterpart. What this leads to is a proportionally smaller static pressure component, decreasing the pressure ratio over the domain. This behavior is confirmed in the two graphs shown in Figure 4.10.



(a) *Varying α, $\Omega = 500$* ***(b)*** *Varying Ω, $\alpha = 10$*

Figure 4.10: *Two figures showing the variation in pressure ratio over the simulation domain for a body force representing a non-cambered rotor with $B = 20$ and $R = 1$. Angle of attack of the blade and rotational speed are varied and results show behavior similar to that expected from theory.*

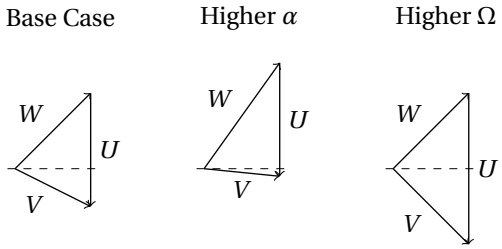
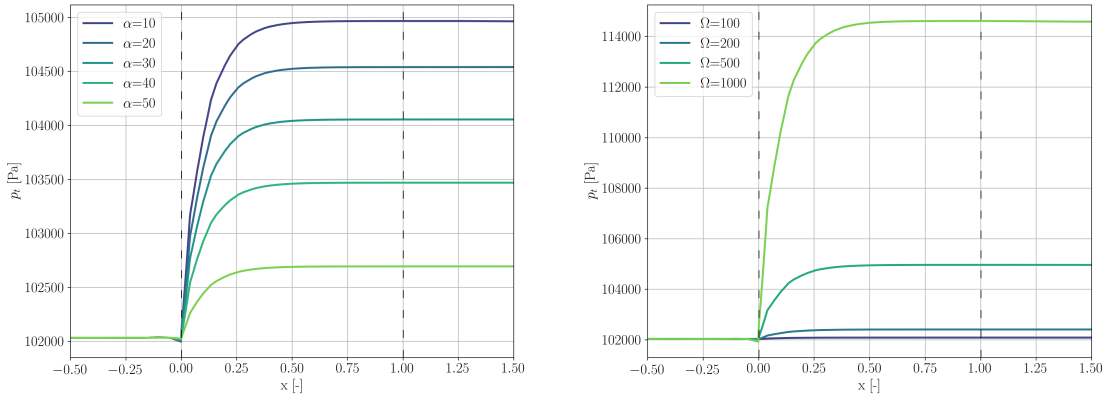


Figure 4.11: *Rotor exit velocity triangles, where an increase in α or Ω lead to different configurations. Higher α leads to lower v_{t_2} as U remains constant. Larger values of Ω increase the magnitude of U and therefore increase v_{t_2}.*

Total pressure is proportional to the work done on the flow, which increases as the tangential component of the absolute velocity vector is increased (Equation 4.7). Figure 4.12 shows that the model does this. Higher α leads to smaller v_{t_2} and lower total pressure rise. Higher Ω leads to higher U and therefore a larger total pressure rise. These two concepts are visualized using velocity triangles in Figure 4.11.

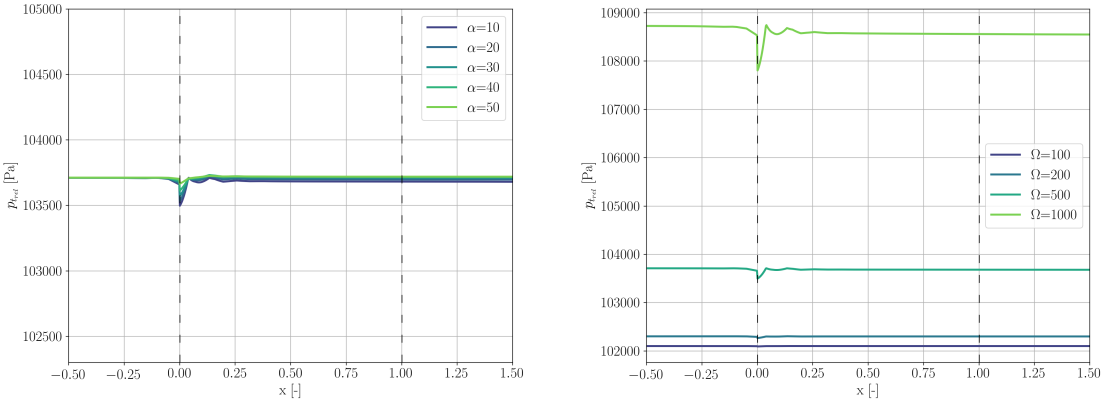


(a) *Varying α, $\Omega = 500$* ***(b)*** *Varying Ω, $\alpha = 10$*

Figure 4.12: *Two figures showing the variation in total pressure over the simulation domain for a body force representing a non-cambered rotor with $B = 20$ and $R = 1$. Angle of attack of the blade and rotational speed are varied.*

$$w = U(v_{t_2} - v_{t_1}) \tag{4.7}$$

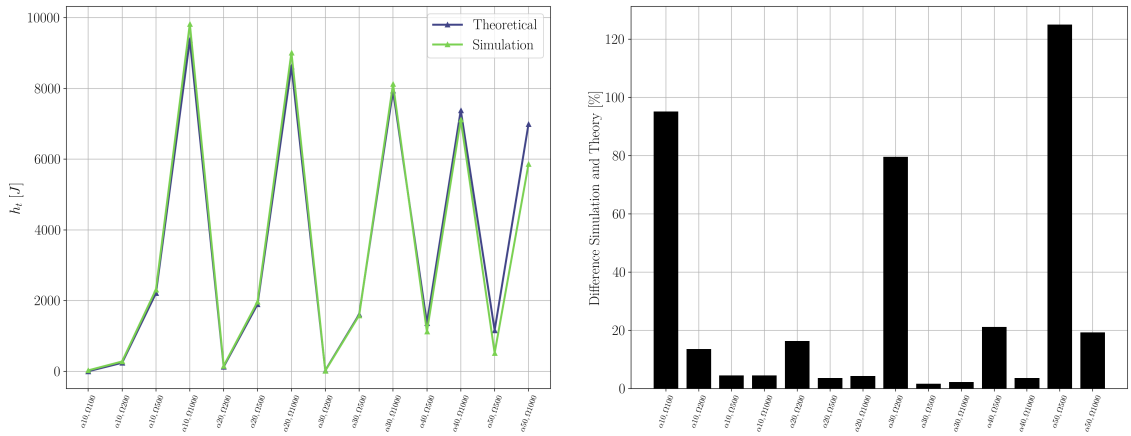
Now that rotation is introduced the flow characteristic total relative pressure becomes relevant. Over a rotor blade row $p_{t_{rel}}$ remains constant, this is because in the relative frame of reference the blade is static. Figure 4.13 shows that for both varying α and Ω this is confirmed. Tangential velocity is the key driver to magnitude of total relative pressure, which is why it varies in Figure 4.13b but not in Figure 4.13a. As is required, mass flow is conserved; these results are presented in Appendix A.4.



(a) *Varying α, $\Omega = 500$*

(b) *Varying Ω, $\alpha = 10$*

Figure 4.13: *Two figures showing the variation in total relative pressure over the simulation domain for a body force representing a non-cambered rotor with $B = 20$ and $R = 1$. Angle of attack of the blade and rotational speed are varied.*



(a) *Theoretical vs. simulation total enthalpy rise*

(b) *Error percentage*

Figure 4.14: *Comparison of analytical and simulations values of total enthalpy for non-cambered rotor test case at $B = 20$. At low rotational speeds the absolute difference in error is smaller but relative difference higher. For all other test cases error percentages are at reasonable levels.*

Contrary to the stator test cases, rotors induce an enthalpy rise due to work being done on the flow. For these test cases a non-cambered rotor was used, meaning that a comparison between analytical and simulation results can be performed. Figure 4.14 shows a comparison of absolute values for each (Figure 4.14a) and the percentage difference between the two (Figure 4.14b) for all the test cases successfully run. At lower rotational speeds, specifically for the test cases $[\alpha = 10, \Omega = 100]$, $[\alpha = 30, \Omega = 200]$, and $[\alpha = 50, \Omega = 500]$, there is a higher error percentage between the two; this is seen by the three outliers in Figure 4.14b. At these conditions, the contribution of numerical error becomes larger as compared to the absolute value of enthalpy rise. Furthermore, low rotational speeds are on the limit of converged simulations. For all test

cases except $\alpha = 10$, a number of low-rotation runs could not achieve convergence. This can be troublesome, however turbomachinery at these rotational speeds are rarely interesting for compressor fan blades, meaning the results should be acknowledged but are of low significance. Neglecting these results, the error of total enthalpy rise is on average about 8%, with the median lying at 4.3%. This is an acceptable result and shows that the model does indeed represent the trend seen in total enthalpy rise.

4.2.5. Cambered Rotor: Whittle Fan

The previous test cases lack complexity that would be similar to a real-world problem. To show that the model's implementation is capable of being applied to complex geometries as well as accurately representing a real-world physical blade, it should be tested with one. The literature contains a number of articles by [Gunn et al., 2013], [Gunn and Hall, 2014], and [Gunn and Hall, 2017] from the Whittle Lab in Cambridge, UK, where they have experimentally tested a BLI fan. Having been tested before and due to the ease with which the blade data can be obtained, the Whittle fan is chosen to perform a validation of Hall's BFM implementation in SU2 (insofar as that is possible without empirical data). Geometrical data for the Whittle fan was obtained and adapted (see Section 3.2.3) for use with the 2D source code by creating a span-wise constant blade profile and subsequently generating a set of camber normals for its use with the body force model.[3] This was then used to run 2D Euler simulations, the results being used to show correlation between the two sets of flow output.

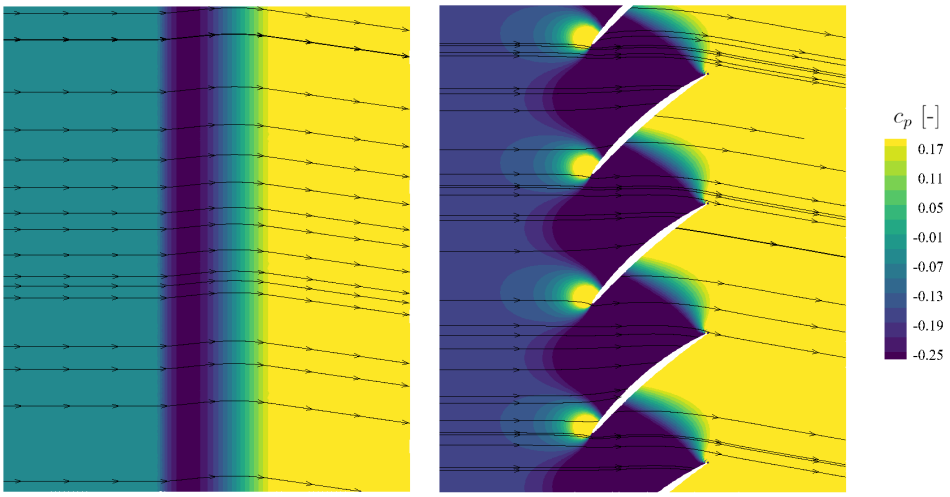


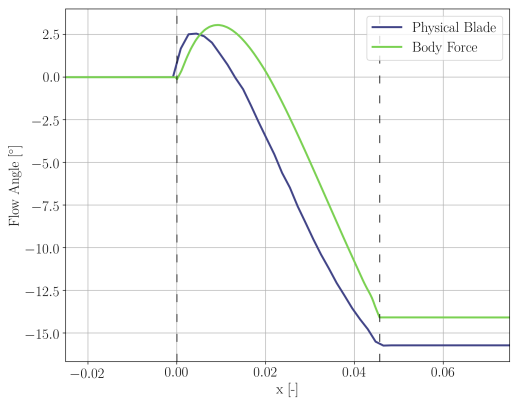
Figure 4.15: *Body force (left) and physical blade (right) representation of pressure coefficient for Whittle fan at operating conditions specified in [Defoe et al., 2018a] (except for* Ω*, which was set to 2362). Absolute velocity streamlines indicate similar flow turning for both cases, with pressure distribution in body force case corresponding to that in the physical blade simulation.*

Figure 4.15 compares the physical blade simulation with its corresponding body force representation. Streamlines and pressure coefficient contours show similar results and indicate the model's ability to portray the flow over a blade row with complex geometry. A number of flow characteristics are analyzed in Figures 4.16 and 4.17, which provide further proof of its accuracy. While a 10.3% difference in flow angle is present in Figure 4.16a, its trend over the domain is similar to that of the physical blade. Analysis for multiple rotational speeds show a systematic error between the two that is attributable to blockage factor, which is covered below. Total relative pressure (Figure 4.16b) shows almost perfect correlation (0.008% difference), with only low numerical error present in the result. Trends of the total pressure are similar but inverse to that seen in the flow angle (Figure 4.17a). Its deviation from the physical blade representation (0.1%) is also due to the blockage factor. Finally, to confirm mass flow conservation, Figure 4.17b shows that this is the case for both simulations. There is a difference between the two, however it should be noted that the y-axis is zoomed, meaning that the error is only 2.4%. From these observations it is concluded that the body force model can accurately represent the complex geometry of a fan blade.

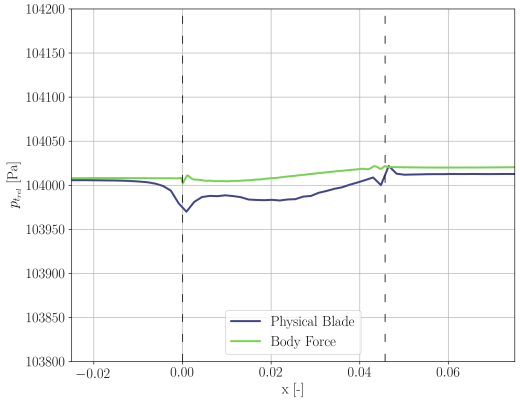
Blockage Factor

The camber line representation presented in Section 3.2.3 essentially neglects the effect of blockage due to an absence of thickness distribution. Where the physical blade representation sees flow diffusion due to

[3] Dr. Jeff Defoe from the University of Windsor supplied the data.

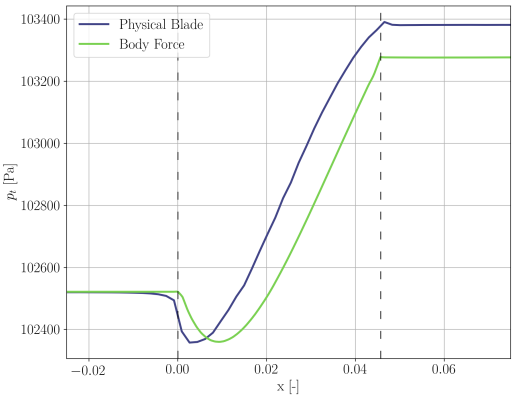


(a) Flow angle

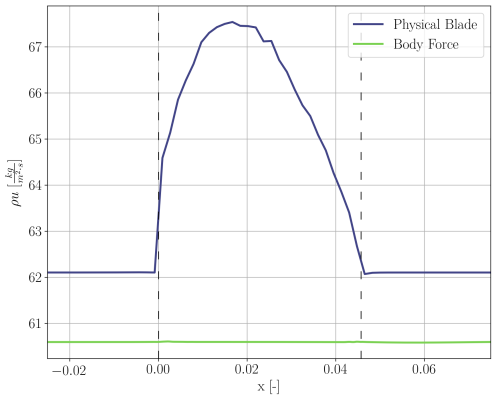


(b) Total relative pressure

Figure 4.16: *Absolute flow angle and total relative pressure comparison between physical blade and body force simulations for Whittle fan at* $\Omega = 2898$*. Both flow characteristics show low discrepancy between the two results. The difference in flow angle over the body force domain is due to a lack of blockage factor in the BFM.*



(a) Total pressure



(b) Momentum in x-direction

Figure 4.17: *Total pressure and x-momentum comparison between physical blade and body force simulations for Whittle fan at* $\Omega = 2898$*. The discrepancy in total pressure is due to a lack of blockage factor in the BFM. Differences in flow over the upper and lower side of the airfoil lead to the peak seen in x-momentum.*

a variation in thickness over the chord of the blade, a camber line does not. Diffusion through the blade passage occurs in the physical blade case, leading to slower flow at the trailing edge. This shortens the relative velocity vector, and with a constant tangential velocity U, increases the tangential component of the absolute velocity vector v_{t_2}. This in turn increases total pressure rise (Figure 4.17a) as it is proportional to these values (Equation 4.8). In Figure 4.18 the flow angle over the domain is plotted for three rotational speeds and a comparison visualized between the body force and physical blade simulations. Noteworthy and consistent with a systematic error is the constant difference in flow angle between both simulations; for each Ω the flow angle differs almost exactly 1.7°. Over the blade domain there is also a consistent difference in the variation of the flow angle; the body force simulation takes longer to react near the LE and peaks at a higher value. These observations are consistent with a systematic error, namely the absence of a thickness distribution. Contrary to [Thollet, 2017] and [Benichou et al., 2019] a blockage factor is not taken into account in this thesis due to increased complexity.

$$\delta p_t \propto v_{t_2} \cdot U \tag{4.8}$$

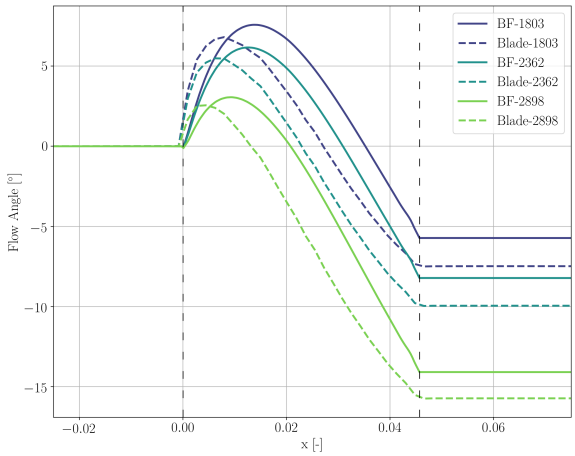


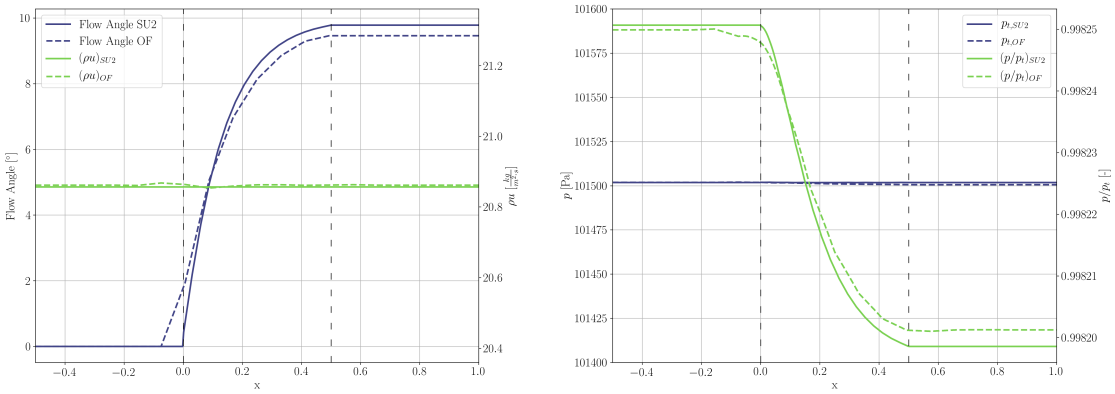
Figure 4.18: *Three simulations run at different rotational speeds (Ω = 1803, Ω = 2362, and Ω = 2898) show a systematic discrepancy between the body force and physical blade absolute flow angle results. A lack of a thickness distribution in the camber line representation eliminates the effect of the blockage factor occurring over a blade row.*

4.2.6. Accuracy Comparison of BFM with openFOAM

There should be a resemblance between implementations of the same model in different solvers as this would indicate that the solver is structurally sound in its implementation. While two solvers will never have the exact same results, similar flow behavior will imply that the model's implementation is accurate. The University of Windsor has a 3D, validated, and fully-working implementation of Hall's BFM in the open-source software openFOAM. Two test cases are used to evaluate similarities between the solver, namely a stator and a rotor.

Stator

Both implementations show strong correlations with each other, with Figure 4.19 comparing a number of flow characteristics. Mass flow is conserved in both programs (1.3% difference) and the flow angle varies in the correct direction (Figure 4.19a). In openFOAM some upstream influence is seen as the flow angle starts increasing before reaching the body force domain. This can be attributed to SU2's use of zones to define the body force domain while openFOAM uses coordinates. This difference is only 3.2%. Figure 4.19b shows a similar effect on the pressure ratio (0.0002% difference), however the same trend is captured in both solvers. Total pressure (0.002% difference) is conserved in SU2 and openFOAM.



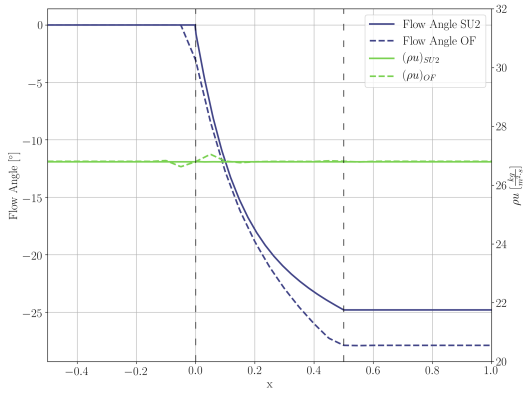
(a) *Flow angle and momentum in x-direction*

(b) *Total pressure and pressure ratio*

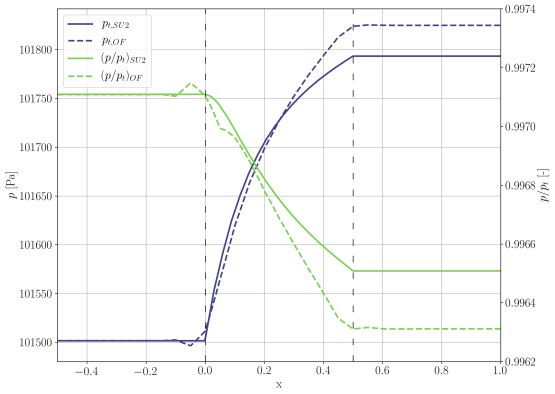
Figure 4.19: *Comparison of flow angle, x-momentum, total pressure, and pressure ratio for stator test case between body force implementations in SU2 (solid line) and openFOAM (dashed line). Results are comparable between the two and show that SU2's implementation is performing similarly to openFOAM's validated model. $M = 0.05$, $\alpha = 10$, $B = 23$, and $R = 1.5$.*

Rotor

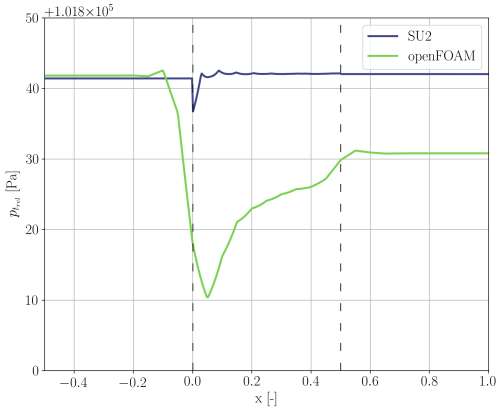
Rotor test case results also show similarities between SU2 and openFOAM in the trends for flow angle, x-momentum, pressure ratio, and total pressure. There is slightly higher flow turning (12%) in openFOAM (Figure 4.20a), which directly influences the total pressure rise (0.03% difference) and pressure ratios (0.02% difference, see Figure 4.20b). Mass flow (0.3% difference) is once again conserved in both solvers and barring some numerical error, so is the total relative pressure (0.01% difference, see Figure 4.20c). Differences in the results between openFOAM and SU2 are minimal but similar trends are captured, which was the aim of the comparison. This does confirm that different solvers never produce the same results. Generally this is due to the approach that a solver takes. Where one solver may be solely built for external flows, another may be specifically developed to optimally resolve internal flow. Differences such as this lead to distinct results for the same simulation setup.



(a) *Absolute flow angle and momentum in x-direction*



(b) *Total pressure and pressure ratio*



(c) *Total relative pressure*

Figure 4.20: *Comparison of flow angle, x-momentum, total pressure, pressure ratio, and total relative pressure for rotor test case between body force implementations in SU2 (solid line) and openFOAM (dashed line). Results are comparable between the two and show that SU2's implementation is performing similarly to openFOAM's validated model. $M = 0.05$, $\Omega = 150$, $B = 23$, $R = 1.5$, and α varies from 35° to 25°.*

4.2.7. Numerical Errors

Figure 4.13b reveals increased numerical error at the interface, where a substantial difference in forces occurs between the upstream zone and body force domain. The difference in total relative pressure between the upstream and downstream zones also increases with higher rotational speed (and therefore the forces in the body force domain). This is because higher gradients lead to larger numerical error. From Table 4.3, starting from $\Omega = 200$ the inter-zonal discrepancy percentage in total relative pressure starts increasing by about an order of magnitude for every doubling of the rotational speed. From this it can be concluded that at $\Omega = 2000$ the difference in total relative pressure between the two zones should be in the order of 2%. This should be

taken into account in future work if high rotational speeds are to be simulated; higher local mesh refinement could be a potential solution.

Table 4.3: *Magnitudes of numerical error present in total relative pressure values for the rotor test case at $\alpha = 10$, $B = 20$, and $R = 1$. As rotational speed increases, both max fluctuation and inter-zonal discrepancy error increases.*

Rotational Speed [RPM]	Max Fluctuation	Inter-zonal Discrepancy
100	0.002%	0%
200	0.005%	0.006%
500	0.2%	0.02%
1000	0.9%	0.2%

Impact of Mesh Refinement

Figure 4.21 shows the total relative pressure when the mesh of the body force domain is refined. Decreasing the width of the cells leads to smaller distances over which to resolve high gradients. This leads to less fluctuation near the interface, but does not have a direct impact on the inter-zonal discrepancy in total relative pressure. Looking at Table 4.4, the error between zones stays constant at 0.01% of the original value, which is negligible. Noting the scale of the y-axis in Figure 4.21, it becomes clear that the impact of mesh refinement may be negligible. As the results in Section 4.3 show, significant decreases in mesh refinement still lead to accurate results with negligible numerical error. This is inherent when performing discretization and is accepted as a consequence of its use. Errors should still be evaluated for every simulation as local mesh refinement could be a logical solution for lower error percentages for certain simulations (such as high gradients near the upstream-body force interface).

Table 4.4: *Error percentages for numerical error of total relative pressure observed at the interface between the upstream zone and body force domain for rotor test case at $\alpha = 40$, $\Omega = 500$, $R = 1$, and $B = 20$. All values are extremely low at under a twentieth of a percent, indicating that numerical error, although present, is negligible for the body force implementation.*

Mesh	Number of Cells	Max Fluctuation	Inter-zonal Discrepancy
Coarse	9560	0.05%	0.01%
Refined	11220	0.02%	0.01%
More Refined	16200	0.009%	0.01%

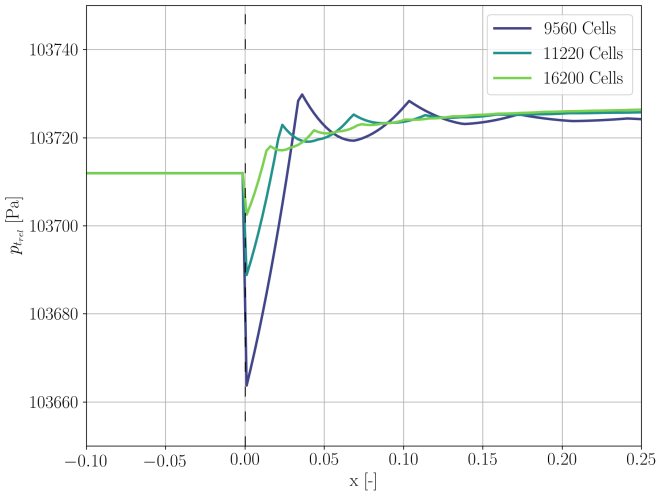


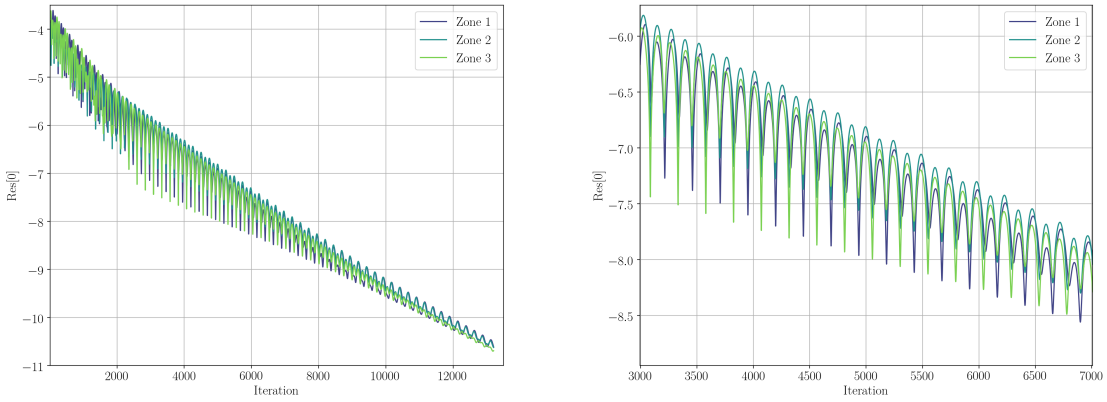
Figure 4.21: *Decrease in numerical error as mesh refinement is increased at the zone interface between the upstream and body force domain. Total relative pressure for rotor ($\alpha = 40$) at $\Omega = 500$. Mesh sizes in x-direction are 9560, 11220, and 16200 cells.*

4.2.8. Convergence

Convergence behavior observed in all simulations show oscillations in residual values along their downward trend. Figure 4.22 shows the mass flow residuals for one of the stator test cases, showing oscillatory behavior

for each of the zones over the entire duration of the simulation. Where three to four orders of magnitude reduction in residual is considered a converged solution, most of the runs were producing in the range of five to six orders. No negative effects of this behavior on convergence are observed, and this is applicable to all test cases that are presented in this thesis. Convergence results for all of these runs are not presented due to the large number that have been performed. It is noteworthy that this behavior is systematic, which most likely means that it can be traced to a source.

Stiffness of the system of equations may be a cause of this behavior, which is due to the adding of source terms in the body force domain. As the equations increase in complexity, the residuals behave more erratically as the solver is having more difficulty finding a solution. Another potential source could be the lack of registering a Jacobian of the source term when using an implicit Euler scheme. The impact of these residuals on the results in not significant, which combined with time constraints of the thesis, mean they have not been investigated. Future work, should it be impacted by this behavior, will have to analyze potential sources.



(a) *All iterations*

(b) *Zoomed*

Figure 4.22: *Mass flow residual behavior for all three zones of the stator test case at* $\alpha = 5$, $B = 20$, *and* $R = 1$.

4.3. Assessment of Computational Performance

Reducing computation time of CFD simulations is valuable, as it is possible to design machines and bring them to the market on shorter time scales. Reaching this goal can be done by reducing required computational power. Benefits of the body force model lie in being able to decrease the number of cells (and shorten computational time) without losing flow accuracy, something a physical blade simulation cannot do due to mesh refinement requirements near surfaces to resolve boundary layer flow. Confirming computational time and mesh size reduction for Hall's BFM implementation in SU2 is essential to exemplify its benefits and relevance for future work. Two test cases are used for this analysis, namely a stator and the Whittle fan. Time per iteration and total computational time is compared for each, with equivalent convergence criteria being used for both the physical blade and body force simulation.[4] This will identify the difference in computational time between the body force model and the physical simulation. Mesh size is reduced for the body force simulation and a comparison of the flow characteristics is made with the physical blade simulation.

4.3.1. Stator

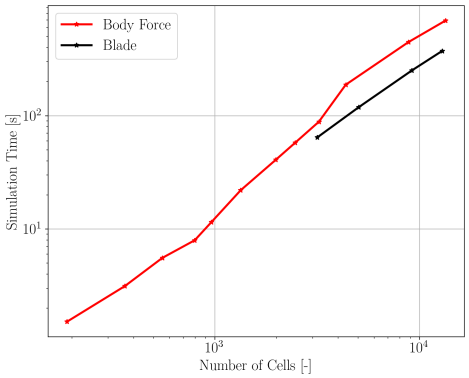
Four simulations were run for the physical blade and twelve for the body force. Mesh size was continually reduced and the accuracy of flow characteristics investigated. Computational time for a converged result is compared to the total number of cells on a log-log plot in Figure 4.23a.[5]

Running a body force simulation on a mesh with the same number of cells as in the physical blade simulation takes much longer to run. The source term has to be added and included in calculations at each node,

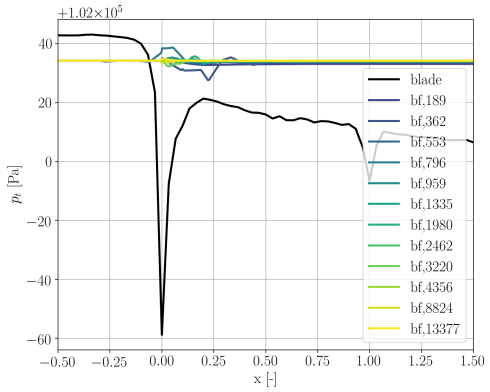
[4] All simulations presented below were run using a single core of a 2.9GHz Intel Quad-Core i7. This is to ensure that an unbiased comparison of total run time is made.

[5] Four orders of magnitude is taken as the minimum requirement for a converged solution.

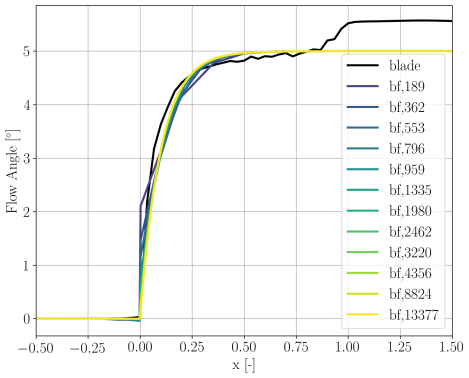
increasing the solver's computational requirements. From Figure 4.23 it is clear that mesh refinement has an impact on the value of flow angle in the body force domain, creating a less smooth curve as the mesh is made coarser (Figure 4.23c). The end result does not vary, with all body force simulations achieving the same value for flow angle before the end of the domain is reached. All simulations have constant total pressure, although similarly to the flow angle, there is more pronounced numerical error when grid refinement is reduced. It is clear from Figure 4.23b that the numerical error is still less than it is for the physical blade simulation, which is impressive. Mass conservation and pressure ratio plots also confirm the body force's ability to accurately represent flow at low cell numbers and can be found in Appendix A.6.



(a) *Computational time*



(b) *Total pressure*



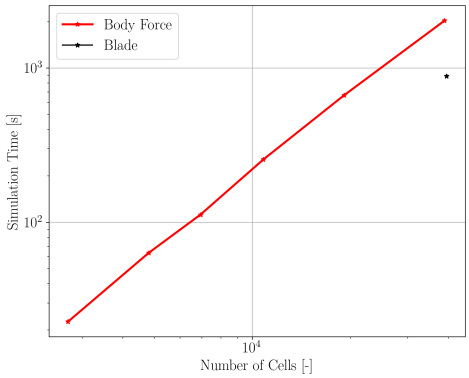
(c) *Flow angle*

Figure 4.23: *Simulation time, total pressure and flow angle are plotted over the domain for a number of body force simulations with descending number of cells. Decreasing the number of cells does not have a large impact on the flow with a 189-cell mesh giving essentially the same results as one with 13377 cells. These runs are for a stator at $\alpha = 5$ and $B = 20$; the number of cells used in the blade computation is 12903.*

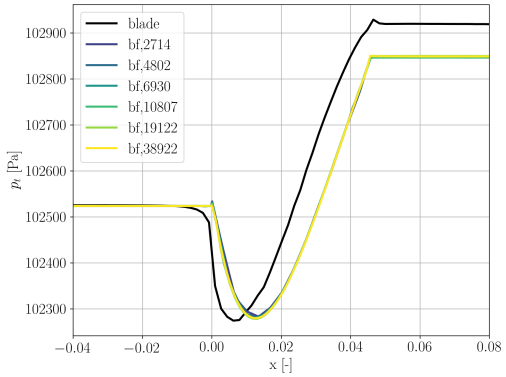
4.3.2. Whittle Fan

Comparing the run times between physical blade and body force simulations for the Whittle fan leads to the same conclusion as that for the stator. Figure 4.24a shows the downward trend in computational time as the number of cells in the body force simulation is decreased. Only a single physical blade simulation was run, however even if its trend would follow that of the body force: there would come a point where the simulations would not be accurate and have trouble converging. Once again the number of cells for the body force simulation is decreased and the results compared to each other and the physical blade result. Similarly to the stator case, there is little difference between the flow properties, even when the number of cells is reduced by a factor of 14. Both total pressure (Figure 4.24b) and total relative pressure (Figure 4.24c) show low variation when compared to the maximum cell number used. Numerical error at 2714 cells for the body force simulation is still lower than that for the physical blade (with 12903 cells). Pressure ratio, x-momentum, and flow angle also show accurate representations independent of cell number and are presented in Appendix

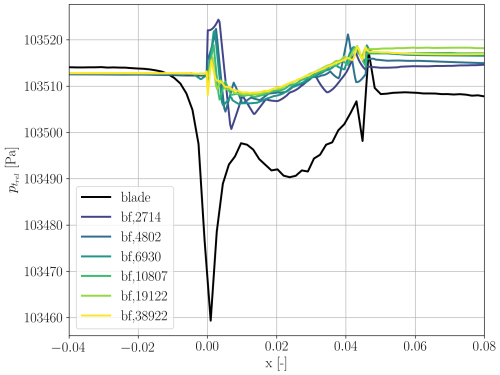
A.6. Further decreases in mesh size are possible, but will impact the resolution of the flow over the body force domain.



(a) *Computational time*



(b) *Total pressure*



(c) *Total relative pressure*

Figure 4.24: *Simulation time, total pressure and total relative pressure are plotted over the domain for a physical blade simulation of the Whittle fan ($\Omega = 2362$) as well as body force simulations with a range of mesh refinement. Decreasing the number of cells has no significant impact on the body force model's representation of the flow, the flow's trends are accurately predicted for the entire range of meshes.*

4.4. Key Takeaways

The following points highlight the principal outcomes of this chapter.

- *Varying α, Ω, B, or N_x and N_y leads to changes in the flow angle, total pressure, total relative pressure, and pressure ratio. Stator and rotor test cases confirm its behavior, verifying the model's implementation.*
- *Numerical errors are present at the interface between the upstream and body force zones, however its magnitude, between 0.002% and 0.9%, is negligible for all test cases presented.*
- *Convergence behavior is consistently oscillatory. Orders of convergence are above five for all test cases, which is higher than three, which is conventionally taken as satisfactory.*
- *Hall's BFM in SU2 shows similar trends in flow characteristics when compared to physical blade simulations of a NACA0012 stator and a Whittle fan. Discrepancies are between 1% and 10%.*
- *A 14-fold reduction in grid size is possible when using Hall's BFM while retaining an almost similar level of flow characteristics accuracy over the body force domain as compared to a physical blade simulation.*
- *Computational time reduction of Hall's BFM with respect to physical blade simulation is between one and two orders of magnitude.*

This page is intentionally left blank.

5

Adjoint-based Optimization using a BFM

In this chapter the test case setup for the adjoint simulations is covered first in Section 5.1. Following this, mesh sensitivity results are presented in Section 5.2. This includes the effect of registration of body force source terms on the adjoint flow and the effect of angle of attack on those results. Finally, Section 5.3 shortly elaborates on the process of computing total gradient using SU2.

5.1. Test Case Setup

The body force representation of an $\alpha = 5$ non-cambered stator (presented in Section 4.1) is used as a test case. The simulation is run on the structured multi-zone mesh presented in Section 3.1. All configuration options are kept the same for the direct solver (which is necessary), while a number of additions are present for the adjoint. A custom objective function is used (see Appendix B.1), namely the sum of the y-velocity at the outlet of the body force zone. In the configuration file a marker is specified at which the objective function is to be evaluated, this is taken as the outflow of the body force zone. Numerical methods for the adjoint flow are taken as similar to the direct solver. The first order central scheme JST is used as a convective numerical method and the same second and fourth order dissipation coefficients are used (0.5 and 0.02 respectively). Furthermore, an implicit Euler formulation is used.

5.2. Mesh Sensitivity

When the body forces are registered as input for CoDiPack, they are added to the symbolically-expressed governing equations $\boldsymbol{R}$. The same set of adjoint calculations is then performed to determine the adjoint flow equation and subsequently the mesh sensitivity. Confirming that the source terms are in fact being registered requires evaluating the adjoint flow as well as the mesh sensitivities. First, a comparison is made between a test case with registered and non-registered body force source terms. In order to confirm that the source terms are registered correctly, a comparison of the adjoint conservative variables ($\boldsymbol{\Psi}$) between the two cases is made. Next, the effect of changing the configuration is investigated by comparing the adjoint conservative variables for three stators at different α. From these results it can be concluded if the implementation of the adjoint BFM in SU2 is correct.

A physical interpretation of the adjoint solution is necessary to be able to analyze the results presented in this section. The value of the adjoint variable indicates how much and in what direction its direct counterpart must change in order to improve the value of the objective function [Marta et al., 2013]. An example being a turbine stator vane with an adjoint density that is negative on the suction side near the trailing edge. This indicates that the density in that region needs to be lowered for the objective function to be improved. Design changes must be induced that ensure this happens; this could be thickness or shape related. Positive values indicate that design changes must be made to vary the flow variable in the positive direction while negative values indicate the opposite. In his article, [Marta et al., 2013] presents further examples of the physical interpretation of the adjoint variables.

5.2.1. Adjoint Flow Field with Body Forces

Body forces are only applied in the specified zone, which means that when they are registered as input for the adjoint solver, there should only be visible changes in the adjoint flow vector in that zone. Figure 5.1 presents the adjoint density for two simulations, one where the source terms are registered and one where they are not. Changes are present between the two cases. In both simulations the upstream and downstream zones (1 and 3) are identical, with adjoint density values in the range of 10^{-19} to 10^{-23}, which is practically machine accuracy and can be neglected. Over the body force domain there is a difference between the two, which confirms that the implementation makes changes to the adjoint vector $\mathbf{\Psi}$ when these source terms are registered.

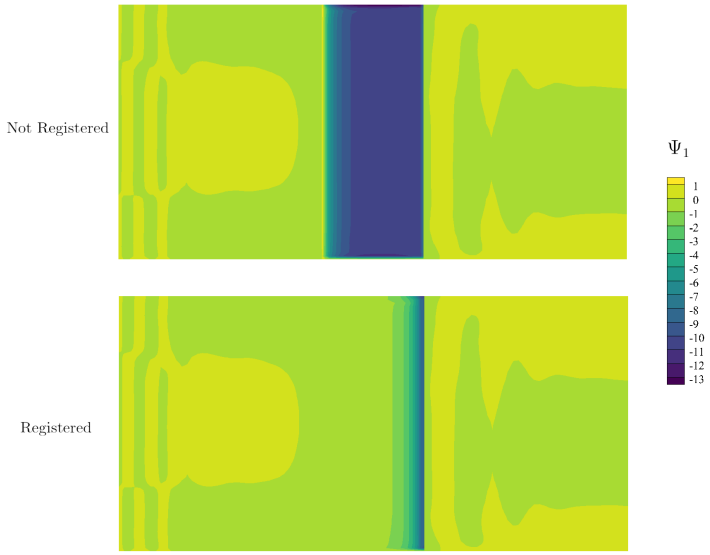
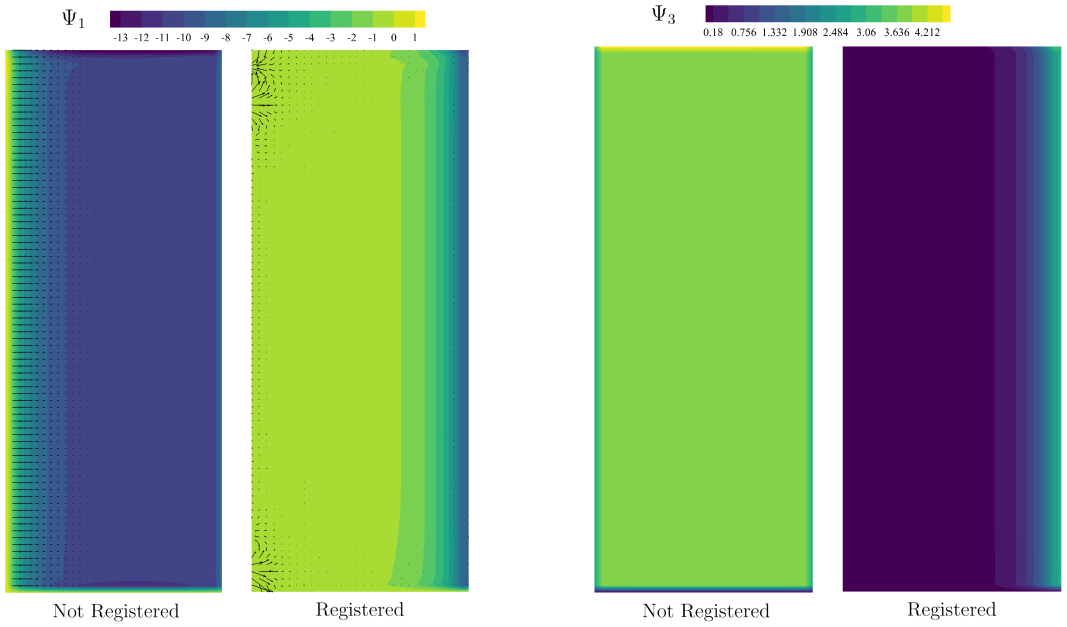


Figure 5.1: *Full domain comparison of adjoint density $\mathbf{\Psi}_1$ for registered and non-registered source terms of non-cambered stator at $\alpha = 5$, $B = 20$, and $R = 1$.*

All four adjoint variables are analyzed. The most prominent differences are visible in the density and y-momentum, which is visualized in Figure 5.2. From these results it is concluded that the registered source terms in fact vary the adjoint flow in the domain over which they are applied. Until two-thirds of the length of the body force zone the adjoint density is near zero. At the trailing edge the adjoint density is lowest, which indicates that design changes that decrease the density would optimize the vertical velocity at the outlet. Varying the angle of the camber line to deflect the flow further in the y-direction would lead to this happening. This is corroborated in Figure 5.2b, which indicates that increasing the y-momentum at the trailing edge has the most impact on vertical velocity at the outlet. Changing the camber in the middle of the domain will have no effect on V_y, while the last point in the domain can significantly increase the vertical velocity. These physical interpretations make sense, which supports the conclusion that the source terms are registered correctly. Adjoint x-momentum and energy are presented in Figure B.2 in Appendix B.2.

In Figure 5.2a the mesh sensitivity at each node is visualized as a vector. This sensitivity indicates in which direction that node would have to move for the specified objective function to increase (or decrease) in magnitude. Compared to the sensitivities computed when the body forces are not registered, there is a pronounced difference between the two, which confirms that registration is successful. However, when body forces are registered the magnitude of this vector should be zero at each node because the mesh does not change. There are two areas of increased sensitivity in the top and bottom left of the body force zone. These discrepancies can be attributed to the interaction between boundaries in that area and the lack of dependency of the objective function on the volumetric grid. Firstly, in these areas there is an interface between two zones (with high gradients) that has numerical error in the direct implementation (see Section 4.2.7) as well as a periodic boundary condition. Secondly, as there is no dependency of the objective function on the mesh there may be a structural complication in determining sensitivities that is deep within SU2's source code. Either way, this "incorrect" sensitivity calculation does not detract from the findings. Due to time constraints this subject was not covered in this thesis and it is advised to study it further for a better grasp of the consequences.

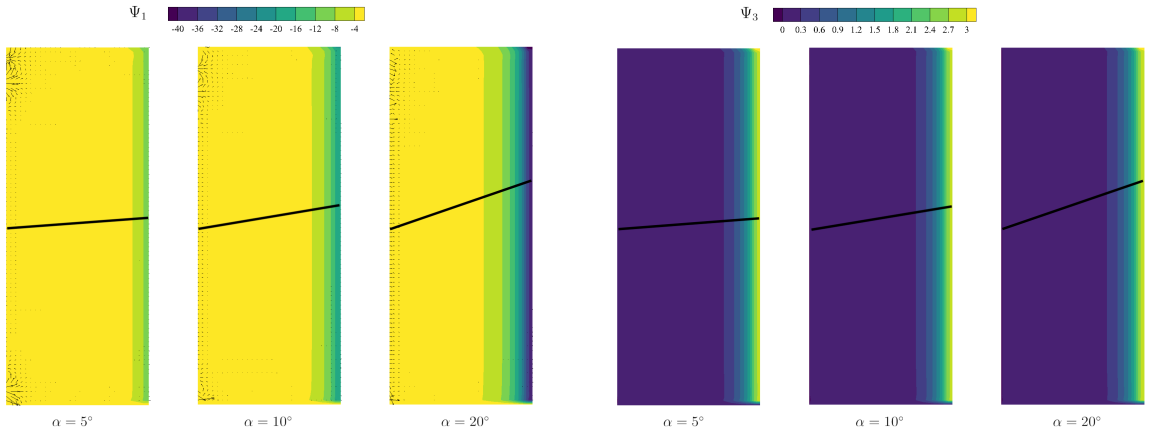


(a) Ψ_1*, including mesh sensitivity as vectors* ***(b)*** Ψ_3

Figure 5.2: *Adjoint density* Ψ_1 *and y-momentum* Ψ_3 *for stator at* $\alpha = 5$, $B = 20$, *and* $R = 1$, *optimized for cumulative* V_y *at outflow of body force zone.*

5.2.2. Effect of Angle of Attack on Adjoint Flow

The adjoint solver is run for three different stator configurations. Except for the angle of attack of the blade, which varies between 5°, 10°, and 20°, the simulations are identical. Figures 5.3 and B.3 compare all four variables of the adjoint vector between the three configurations.



(a) Ψ_1*, including mesh sensitivity and camber lines* ***(b)*** Ψ_3*, including camber lines*

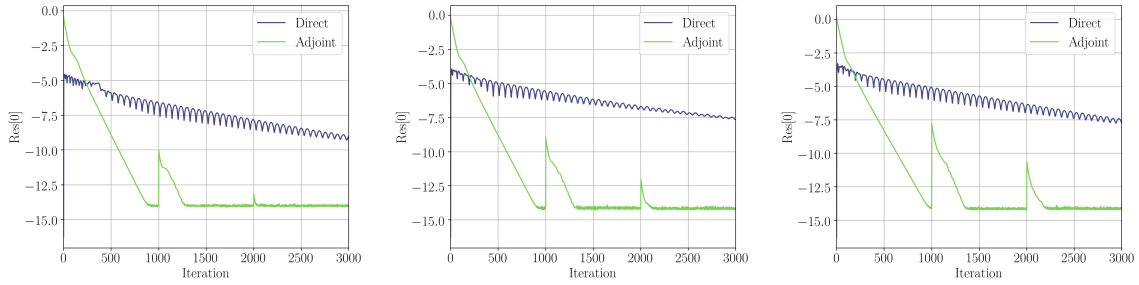
Figure 5.3: *Adjoint density* Ψ_1 *and y-momentum* Ψ_3 *compared for three different values of* α*: 5, 10, and 20. Simulation is optimized for cumulative* V_y *at outflow of body force zone.* $B = 20$ *and* $R = 1$.

As the angle of attack is increased the magnitude of the adjoint density towards the trailing edge of the domain becomes larger. It indicates that design changes that decrease the density will minimize the objective function. Higher adjoint density magnitudes indicate larger decreases in density are needed to increase y-velocity. Design changes that increase the y-momentum of the flow have equal impacts, independent of the angle of attack of the stator. Vertical velocity (V_y) is directly related to the y-momentum, meaning that changes in one will directly impact the other. Varying the camber with a certain value will deflect the flow the same amount, independent of the angle of the flow. For both the adjoint density and y-momentum the biggest changes can be made at the trailing edge; camber line adaptation at the TE will have the largest impact on V_y. Changes at mid-chord will not. Adjoint x-momentum and energy are presented in Figure B.3

in Appendix B.2. This physical interpretation is consistent with reality, confirming that the implementation is successful. The mesh sensitivity vector is present in Figure 5.3a. Changing the angle of attack leads to small variations, however the discrepancies in the bottom and top left of the body force zone are still present. Section 5.2.1 elaborates on this behavior.

5.2.3. Computational Performance of BF Adjoint Solver

Running the adjoint solver for a test case first requires a direct simulation of the configuration, the results of which are then used as input for the adjoint computation. Convergence of the direct solver is presented in Section 4.2.8 and matches with output from the three adjoint test cases. Figure 5.4 shows the body force zone density residual for both the direct and adjoint solver of these runs. Oscillatory convergence is once again seen for all three direct solutions. The adjoint residual, on the other hand, shows a steep and consistent residual drop in the first thousand iterations, reaching almost 14 orders of magnitude convergence in that time. Peaks occur at 1000 and 2000 iterations, which is when the solver writes solution files. Soon thereafter the residual sharply drops and levels off at 10^{-14}. This is sufficient convergence for the adjoint simulation, which confirms that the results are reliable.



(a) $\alpha = 5$ ***(b)*** $\alpha = 10$ ***(c)*** $\alpha = 20$

Figure 5.4: *Convergence for both the direct and adjoint runs for all three test cases ($\alpha = 5$, $\alpha = 10$, $\alpha = 20$). All three show similar behavior, with the direct residual oscillating over a downward trend and the adjoint residual reaching convergence before 1000 iterations.*

Table 5.1 shows the time per iteration and RAM requirements for the direct and adjoint solvers. RAM requirements vary between direct (40MB) and adjoint (127MB) but not per test case. Time per iteration for the adjoint is 32% to 44% lower than the direct solver if all 3000 iterations are taken. It is 14% to 35% higher if only iterations before max convergence of zones 1 and 3 is taken.

Table 5.1: *RAM requirements and time per iteration for direct and adjoint runs with 3000 iterations. Adjoint time per iteration (short) is taken for all iterations until convergence of zones 1 and 3 reaches 10^{-32}. After this iterations are significantly shorter.*

	Direct		**Adjoint**			
Angle of Attack	RAM	Time/Iter	RAM	Time/Iter [max]	Time/Iter [short]	Iterations
5	40MB	0.066s	127MB	0.043s	0.089s	464
10	40MB	0.077s	127MB	0.043s	0.088s	465
20	40MB	0.07s	128MB	0.043s	0.089s	470

5.3. Computation of Total Gradient $\frac{dJ}{d\alpha}$

Due to a number of limitations, calculation of the objective function's sensitivity with respect to camber normal value was not possible in the time frame of this thesis. A segmentation fault delayed the mesh sensitivity computation by six weeks, and given the time limitations of a thesis project and the limited experience in C++ development, the author was not successful in completing this step. Following the methodology presented in Section 3.4.3, the total gradient was attempted to be calculated using body force camber normal as input. While no compilation errors or segmentation faults were experienced, the total gradient values calculated were equal to zero. It is concluded that further work is required to alter the code in such a way that total gradient computation is successful. The current implementation may be faulty or additional work may need to be done. This capability is an essential advancement in proving SU2's ability to perform adjoint optimization with body forces, and as such it is suggested as a recommendation for future work in Section 6.2. A suggested method for validation of the obtained gradients has been worked out and is presented in Appendix B.4.

5.4. Key Takeaways

The following points highlight the principal outcomes of this chapter.

- *The value of the adjoint vector indicates how the direct counterpart must be changed to improve the objective function. Large values of adjoint density therefore indicate that design changes must be made that increase the density of the flow in that region.*
- *Body force source terms are successfully registered as input for the adjoint solver.*
- *Registering body forces as input for the adjoint solver leads to adjoint flow variations in the body force zone. Physical interpretations of the adjoint are logical and confirm successful implementation.*
- *Changing stator angle of attack varies the adjoint flow in the body force domain. Physical interpretation matches reality, confirming successful implementation.*
- *Adjoint convergence is rapid and stable, reaching twelve orders of convergence in under 1000 iterations.*
- *Computation of total gradient was unsuccessful in the time frame of the thesis.*

This page is intentionally left blank.

6

Conclusions

Section 6.1 presents concluding remarks for this thesis. A number of technical recommendations as well as suggestions for future work are presented in Section 6.2.

6.1. Concluding Remarks

Hall's body force model is implemented into the source code of SU2's direct and adjoint solvers. Starting with the direct solver the body force was tested for increasingly complex test cases ranging from a non-cambered stator to the Whittle fan. Using these test cases the implementation was confirmed to accurately represent trends in flow characteristics over the blade row. Mesh refinement of body force simulations was reduced and reductions in grid size as well as computational time was observed with respect to physical blade simulations. Extension of the BFM to the adjoint solver was partially successful. While computation of the total gradient was not achieved, the body forces were successfully registered and the adjoint flow showed expected trends. Obtaining the total gradient directly from SU2 will allow design optimization to be performed quickly, and as such it should be a focus of future work. Combining Hall's BFM with the adjoint method will accelerate conceptual design optimization, making it a relevant goal in the field of turbomachinery. It will be especially useful for design of interacting components, such as a BLI propulsor integrated with a fuselage.

The points below highlight the key takeaways from this thesis:

- *Forces are calculated at each node using Hall's BFM by adding a function to SU2's CSolver class. The CNumerics, CConfig, CVariable, and CIntegration classes are adapted to ensure that the code runs without errors.*
- *Trends in flow angle, pressure ratio, total pressure, total relative pressure, and x-momentum are consistent with theory; varying α, Ω, or B gives expected results. When compared to blade simulation counterparts, openFOAM, or analytical calculations, flow characteristics of body force simulations differ by between 0.1% and 12%.*
- *Oscillatory residual behavior of BFM-based simulations is observed, however its impact on the results is negligible as five to six orders of reduction are obtained. This is likely due to unregistered terms in the Jacobian or stiffness of the equations and is suggested to be investigated further.*
- *A fourteen-fold reduction in grid size while retaining flow characteristics is possible for body force simulations as compared to physical blade simulations. This results in one to two orders of magnitude reduction in computational time.*
- *Registration of body forces as source terms in the adjoint solver is successful. Physical interpretations of the adjoint indicate design changes are necessary at the trailing edge to increase cumulative y-velocity at the outflow, which is consistent with theory.*
- *Total gradient computation returns inconclusive results for preliminary implementation. Given the code structure of the adjoint solver it should be possible, however further work is required for this to be successful.*

6.2. Recommendations for Future Work

There are a number of technical improvements that the author sees as beneficial for performance of the model. They are mostly related to the code itself. Besides this, a number of suggestions for further research is given. They are presented in no particular order.

6.2.1. Technical Improvements

Code Generalization Hard-coding is currently present in the source code of SU2, meaning that it does not work off the bat for any configuration. To ensure that the model works for all configurations and that no errors appear when adapting the code to include a third dimension, this hard-coding should be removed. All functions need to be reviewed to ensure they are not made specifically for a two-dimensional implementation. Furthermore the camber normals should be read from a file, allowing for new configurations to be run without having to recompile. This has been tried without success, and the code should be revised or removed when attempting to include this feature. Finally, an expert on the structure of SU2 should evaluate the current implementation and make sure that it is computationally most effective.

File Reading A function should be created that reads camber normals from a file. Currently, they are generated using a Python script and hard-coded into the *CEulerSolver::ComputeBodyForce_Turbo* function. This is tedious work and requires recompiling the code before being able to run another simulation. A solution to this could be the generation of blades using *ParaBlade*. Generation of a camber line and the respective camber normals could then be output to a file. This could then be used as the standard input used for SU2 to simplify and quicken the process of running new simulations.

SU2 V7.0 Compatibility Recently SU2 was updated to its most recent version, V7.0 Blackbird. The implementation should be tested in the new version to ensure everything still works.

NURBS Representation Camber line generation using Non-Uniform Rational B-Splines (NURBS) (as done by [Anand et al., 2018]) could simplify the design process. Using only two design variables the camber line can be defined, from which the camber normal values at a self-defined number of locations can be determined.

6.2.2. Research Suggestions

Total Gradient Adapting the source code for successful total gradient computation is the next big step in confirming SU2's adjoint solver's compatibility with body forces. For this, the steps in Section 3.4.3 should be followed.

Three-Dimensional Adding a third dimension to the model is a matter of including a z-component to the body force computation and ensuring all code works for an arbitrary number of dimensions. Verification of the model will have to be performed in a similar way as to that presented in the current work. It is recommended to undertake this after successful integration of the BFM with the adjoint solver. A suggested approach for this task is presented in Appendix A.5

Shape Optimization Following the computation of total gradient it would be valuable to look into using it to perform optimization of the camber line. Topics to look into include the kind of optimization to perform, what the objective function is, what the constraints are, and if the code can support this optimization process given that it normally runs using grids that deform. What gains can be achieved using this method and does it perform as expected? Being able to answer this question would be unique, as it would confirm that optimization can be performed on camber normal values using SU2's adjoint solver.

Model Complexity Similar to the work by [Thollet, 2017], a number of additions can be made to Hall's model. Including viscosity in the form of a parallel force term would allow losses to be modeled. Further additions include a blockage factor, which would account for the thickness distribution naturally present in any blade row (see work by [Thollet, 2017] and [Benichou et al., 2019]). A compressibility correction is included in the current code, however it has only been tested on flow at $M = 0.05$, which means testing could be extended to higher Mach number flow. All three of these additions could be pursued, with viscosity being a noteworthy addition to achieve more accurate flow representations when looking into losses.

Distorted Inlet Flow Inlet flow distortion can be simulated in SU2 by specifying a velocity profile at the inlet of the domain. As mentioned in Section 2, BLI fans have the potential for fuel reduction on next-generation aircraft. Two-dimensional testing using Hall's model should be performed first. Once verified it can be extended to a 3D BFM.

More Test Cases For further verification of the solver and its BFM implementation more configurations can be tested. NASA rotor test cases can be tried out or Mach numbers can be increased to evaluate transonic flow. Ideally a number of configurations are tested that have empirical data, which can then be used to validate the model to the fullest extent.

Multiple Blade Rows Adding a second body force zone immediately following the first would be a method of modeling multiple blade rows using a body force model. The current research is focused on using the model solely for fans, however in the future it could be applied to reduce computational time of conceptual compressor design. To the author's knowledge this has not been pursued as of the time of writing.

This page is intentionally left blank.

Bibliography

[Anand et al. 2018] ANAND, N. ; VITALE, S. ; PINI, M. ; COLONNA, P.: Assessment of FFD and CAD-based shape parametrization methods for adjoint-based turbomachinery shape optimization. In: *Proceedings of the Global Power and Propulsion Forum*. Montreal : Global Power & Propulsion Society, 2018, p. 1–8

[Arens et al. 2005] ARENS, K. ; RENTROP, P. ; STOLL, S. O. ; WEVER, U.: An adjoint approach to optimal design of turbine blades. In: *Applied Numerical Mathematics* 53 (2005), Nr. 2-4, p. 93–105. – ISSN 01689274

[Benichou et al. 2019] BENICHOU, E. ; DUFOUR, G. ; BOUSQUET, Y. ; BINDER, N. ; ORTOLAN, A. ; CARBONNEAU, X.: Body force modeling of the aerodynamics of a low-speed fan under distorted inflow. In: *International Journal of Turbomachinery Propulsion and Power* 4 (2019), Nr. 29, p. 1–15

[Defoe et al. 2018a] DEFOE, J. J. ; ETEMADI, M. ; HALL, D. K.: Fan Performance Scaling With Inlet Distortions. In: *Journal of Turbomachinery* 140 (2018), Nr. 7, p. 1–11. – ISSN 0889-504X

[Defoe et al. 2018b] DEFOE, J. J. ; MINAKER, Q. J. ; ALTENHOF, W. J.: Estimation of Unsteady Blade Loading Due To Inlet Distortion Using A Body Force Rotor Model. In: *15th International Symposium on Unsteady Aerodynamics*. Oxford, United Kingdom : ISUAAAT Scientific Committee, 2018, p. 1–11

[Economon et al. 2016] ECONOMON, T. D. ; PALACIOS, F. ; COPELAND, S. R. ; LUKACZYK, T. W. ; ALONSO, J. J.: SU2: An Open-Source Suite for Multiphysics Simulation and Design. In: *AIAA Journal* 54 (2016), Nr. 3, p. 828–846. – ISBN 0001-1452

[Fernández 2017] FERNÁNDEZ, R. S.: *A coupled adjoint method for optimal design in fluid-structure interaction problems with large displacements*, Imperial College London, PhD Dissertation, 2017. – 1–186 p.

[Florea and Hall 2001] FLOREA, R. ; HALL, K. C.: Sensitivity analysis of unsteady inviscid flow through turbomachinery cascades. In: *AIAA Journal* 39 (2001), Nr. 6, p. 1047–1056. – ISSN 00011452

[Ghidoni et al. 2006] GHIDONI, A. ; PELIZZARI, E. ; REBAY, S. ; SELMIN, V.: 3D anisotropic unstructured grid generation. In: *International Journal for Numerical Methods in Fluids* 51 (2006), Nr. 9-10, p. 1097–1115. – ISSN 02712091

[Giles and Pierce 2000] GILES, M. B. ; PIERCE, N. A.: An introduction to the adjoint approach to design. In: *Flow, Turbulence and Combustion* 65 (2000), Nr. 3-4, p. 393–415. – ISSN 13866184

[Giles 1991] GILES, M.B.: Non-reflecting boundary conditions for Euler equation calculations. In: *AIAA Journal* 28 (1991), Nr. 12

[Gong 1999] GONG, Y.: *A Computational Model for Rotating Stall and Inlet Distortion in Multistage Compressors*, Massachusetts Institute of Technology, PhD Dissertation, 1999. – 1–187 p.. – URL https://dspace.mit.edu/handle/1721.1/9733

[Gong et al. 1999] GONG, Y. ; TAN, C. S. ; GORDON, K. A. ; GREITZER, E. M.: A Computational Model for Short-Wavelength Stall Inception and Development in Multistage Compressors. In: *Journal of Turbomachinery* 121 (1999), p. 726–734. – ISSN 0889504X

[Gunn and Hall 2014] GUNN, E. J. ; HALL, C. A.: Aerodynamics of Boundary Layer Ingesting Fans. In: *ASME Turbo Expo 2014*. Duesseldorf, Germany : ASME, 2014, p. 1–13

[Gunn and Hall 2017] GUNN, E. J. ; HALL, C. A.: Non-Axisymmetric Stator Design for Boundary Layer Ingesting Fans. In: *ASME Turbo Expo 2017*. Charlotte, North Carolina, United States : ASME, 2017, p. 1–10

[Gunn et al. 2013] GUNN, E. J. ; TOOZE, S. E. ; HALL, C. A. ; COLIN, Y.: An Experimental Study of Loss Sources in a Fan Operating With Continuous Inlet Stagnation Pressure Distortion. In: *Journal of Turbomachinery* 135 (2013), p. 1–10

[Hall 2015] HALL, D. K.: *Analysis of civil aircraft propulsors with boundary layer ingestion*, Massachusetts Institute of Technology, PhD Dissertation, 2015. – 1–116 p.. – URL https://dspace.mit.edu/handle/1721.1/97353

[Hall et al. 2017] HALL, D. K. ; GREITZER, E. M. ; TAN, C. S.: Analysis of Fan Stage Conceptual Design Attributes for Boundary Layer Ingestion. In: *Journal of Turbomachinery* 139 (2017), p. 1–10. – ISBN 9780791849699

[Hill 2017] HILL, D. J.: *Compressor Performance Scaling in the Presence of Non-Uniform Flow*, University of Windsor, PhD Dissertation, 2017. – 1–124 p.

[Hill and Defoe 2018] HILL, D. J. ; DEFOE, J. J.: Innovations in Body Force Modeling of Transonic Compressor Blade Rows. In: *International Journal of Rotating Machinery* (2018), p. 1–12. – ISSN 1023-621X

[Horlock and Marsh 2007] HORLOCK, J. H. ; MARSH, H.: Flow Models for Turbomachines. In: *Journal of Mechanical Engineering Science* 13 (2007), Nr. 5, p. 358–368. – ISSN 0022-2542

[Jameson 1988] JAMESON, A.: Aerodynamic Design via Control Theory. In: *Journal of Scientific Computing* 3 (1988), Nr. 3, p. 233–260

[Jameson 2017] JAMESON, A.: Origins and further development of the Jameson-Schmidt-Turkel scheme. In: *AIAA Journal* 55 (2017), Nr. 5, p. 1487–1510. – ISSN 00011452

[Li and Zheng 2017] LI, Z. ; ZHENG, X.: Review of design optimization methods for turbomachinery aerodynamics. In: *Progress in Aerospace Sciences* 93 (2017), Nr. July, p. 1–23. – URL http://dx.doi.org/10.1016/j.paerosci.2017.05.003. – ISSN 03760421

[Luo et al. 2011] LUO, J. ; XIONG, J. ; LIU, F. ; MCBEAN, I.: Three-dimensional aerodynamic design optimization of a turbine blade by using an adjoint method. In: *Journal of Turbomachinery* 133 (2011), Nr. 1, p. 1–11. – ISSN 0889504X

[Marble 1942] MARBLE, F. E.: Three-dimensional flow in turbomachines. In: *High Speed Aerodynamics and Jet Propulsion* 10 (1942)

[Marta et al. 2013] MARTA, A. C. ; SHANKARAN, S. ; WANG, Q. ; VENUGOPAL, P.: Interpretation of adjoint solutions for turbomachinery flows. In: *AIAA Journal* 51 (2013), Nr. 7, p. 1733–1744. – ISSN 00011452

[Martins and Hwang 2013] MARTINS, J. R. R. A. ; HWANG, J. T.: Review and unification of methods for computing derivatives of multidisciplinary computational models. In: *AIAA Journal* 51 (2013), Nr. 11, p. 2582–2599. – ISBN 9781600869372

[Minaker and Defoe 2019] MINAKER, Q. J. ; DEFOE, J. J.: Prediction of Crosswind Separation Velocity for Fan and Nacelle Systems Using Body Force Models : Part 1 : Fan Body Force Model Generation without Detailed Stage Geometry. In: *International Journal of Turbomachinery Propulsion and Power* 4 (2019), Nr. 43, p. 1–23

[Palacios et al. 2013] PALACIOS, F. ; COLONNO, M. ; ARANAKE, A. ; DURAISAMY, K. ; COPELAND, S. R. ; ECONOMON, T. D. ; TAYLOR, T. ; LUKACZYK, T. W. ; LONKAR, A. K. ; HICKEN, J. ; ALONSO, J. J. ; CAMPOS, A.: Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design. In: *51st AIAA Aerospace Sciences Meeting*. Grapevine, Texas, United States : AIAA, 2013, p. 1–60. – ISBN 978-1-62410-181-6

[Palacios et al. 2014] PALACIOS, F. ; ECONOMON, T. D. ; ARANAKE, A. ; PADRON, S. ; TRACEY, B. ; MANOSALVAS, D. E. ; COPELAND, S. R. ; VARIYAR, A. ; ALONSO, J. J. ; LUKACZYK, T. W. ; LONKAR, A. K. ; NAIK, K. R.: Stanford University Unstructured (SU2): Analysis and Design Technology for Turbulent Flows. In: *AIAA SciTech Forum - 52nd Aerospace Sciences Meeting*. National Harbor, Maryland, United States : AIAA, 2014, p. 1–33. – ISBN 978-1-62410-256-1

[Peters 2014] PETERS, A.: *Ultra-Short Nacelles for Low Fan Pressure Ratio Propulsors*, Massachusetts Institute of Technology, PhD Dissertation, 2014. – 1–248 p.

[Pini 2013] PINI, Matteo: *Turbomachinery Design Optimization using Adjoint Method and Accurate Equations of State*, Politecnico di Milano, Dissertation, 2013. – 153 p.. – URL https://www.politesi.polimi.it/bitstream/10589/89787/1/2014{_}03{_}PhD{_}Pini.pdf

[Roe 1981] ROE, P. L.: Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. In: *Journal of Computational Physics* 43 (1981), p. 357–372

[Rubino 2019] RUBINO, A.: *Reduced Order Models For Unsteady Fluid Dynamic Optimization of Turbomachinery*, Delft University of Technology, Dissertation, 2019. – 1–112 p.

[Saad and Schultz 1986] SAAD, Y. ; SCHULTZ, M. H.: GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. In: *SIAM Journal on Scientific and Statistical Computing* 7 (1986), Nr. 3, p. 856–869. – ISSN 0196-5204

[Thollet 2017] THOLLET, W.: *Body force modeling of fan – airframe interactions*, Universite Federale Toulouse Midi-Pyrenees, PhD Dissertation, 2017. – 1–157 p.

[Wang and He 2010] WANG, D. X. ; HE, L.: Adjoint Aerodynamic Design Optimization for Blades in Multistage Turbomachines—Part I: Methodology and Verification. In: *Journal of Turbomachinery* 132 (2010), p. 1–14. – ISSN 0889504X

[Yang et al. 2003] YANG, S. ; WU, H. Y. ; LIU, F. ; TSAI, H.: Aerodynamic design of cascades by using an adjoint equation method. In: *41st Aerospace Sciences Meeting and Exhibit*. Reno, Nevada, United States : AIAA, 2003, p. 1–12. – ISBN 9781624100994

[Zhou et al. 2015] ZHOU, B. Y. ; ALBRING, T. ; GAUGER, N. R. ; ECONOMON, T. D. ; PALACIOS, F. ; ALONSO, J. J.: A discrete adjoint framework for unsteady aerodynamic and aeroacoustic optimization. In: *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. Dallas, Texas : AIAA, 2015, p. 1–11. – ISBN 9781624103681

This page is intentionally left blank.

A

Direct Implementation

This appendix contains supporting information regarding the direct implementation.

A.1. Resources

All adaptations to the source code can be found in the branch "feature_turbo_bodyforce" on the SU2 repository. It can be accessed at:

https://github.com/su2code/SU2/tree/feature_turbo_bodyforce

All other relevant files such as test cases, camber generators, results, and relevant documentation is stored on the a publicly-accessible repository:

https://github.com/mtlatour/Thesis_MarcLatour.git

A.2. Difference Between Structured and Unstructured Grids

Throughout the process of the work both a structured and unstructured meshing program were used to generate grids for simulations. To identify the difference in results between the two a comparison of the same simulation was performed to identify potential variations in numerical error. When visualizing the flow there were apparent differences between the two, specifically when analyzing the flow's x-momentum. For this flow variable the structured grid indicated a vertical line of error in the y-direction at one cell width from the interface between the upstream and body force zones. Unstructured mesh leads to a more spread numerical error over the first half of the body force zone. This is largely due to non-conformal mesh at the interface, which leads to interpolation error. Although present in visualizations of the results, the error is negligible when taking the meridional variation in a flow characteristic. For a stator test case the error between the two mesh types is on the order of 0.005%. While this means it shouldn't even deserve a second glance, for consistency reasons simulations use the same mesh type where possible.

Computational time reduction simulations are performed on unstructured mesh, which is due to a lack of access to the structured mesh generator Pointwise after the author left the University of Windsor. This is no problem as all simulations are consistent with the type of mesh being used. All other simulations could also have been run on unstructured mesh as the increase in error would not lead to any significant difference in the results. The key difference lies at the interface between the upstream and body force zones, where numerical error is largest due to high gradients.

A.3. Source Code Adaptation for the Direct Solver

A more detailed specification of the classes that are adjusted is given below. Each of the adapted files are included in this section, with an explanation given as to what exactly has been changed.

CIteration

iteration_structure.cpp	In the function *CFluidIteration::Iterate*, a call is made to the function computing the body force: *CEulerSolver::ComputeBodyForce_Turbo*. After the flow has been calculated this function is called if the condition that a body force has been specified is met.

CSolver

solver_direct_mean.cpp	The function *CEulerSolver::ComputeBodyForce_Turbo* is created and contains the calculations for Hall's body force. Algorithm 1 elaborates on the steps taken by the function. Once computation is complete the value is stored to the node.
solver_direct_mean.cpp	To ensure that body force residuals are computed correctly *CEulerSolver::Source_Residual* is changed. In this function the source term residuals are computed using their respective functions in the numerics class. The numerics function computing the body force residual is adapted such that it takes as input the body force vector at the node.
solver_structure.hpp/.inl	Both the header and inline files must be adjusted so that they contain newly-defined functions or functions with a new set of inputs.

CVariable

variable_structure.cpp	Defined and initialized the variable *Body_Force_Turbo* so that it can be used to store the body force vector at each node.
variable_structure.hpp	To be able to store and acquire the body force vector at each node, functions are added to this file specifically for this purpose. *SetBodyForceVector_Turbo* and *GetBodyForceVector_Turbo* perform these tasks and allow the vector to be set in the *CSolver* class and grabbed when loading the *CNumerics* class.

CNumerics

numerics_direct_mean.cpp	The function *CSourceBodyForce::ComputeResidual* is adapted so that it takes the value of the body force vector at the node as input and stores it to the vector used in calculating the residual.
numerics_structure.hpp	Added the body force vector pointer as input for the function *CSourceBodyForce::ComputeResidual*.

CConfig

config_structure.cpp	Included a number of options for body force computation in the function *CConfig::SetConfig_Options*. These are BODY_FORCE_ZONE, BODY_FORCE_TYPE, BODY_FORCE_BLADES, BODY_FORCE_ROTATION, BODY_FORCE_RADIUS, and BODY_FORCE_NORMALS. ZONE specifies where the body force should be applied, TYPE can be either constant or varying (Hall), BLADES is the number of blades, ROTATION is the rotational speed in RPM, RADIUS is in meters, and NORMALS is an array with camber normal values. NORMALS has been included as an option but does not work yet.
option_structure.hpp	Added a list of options for the configuration option BODY_FORCE_TYPE, which allows a number of options to be specified.
config.cfg	Configuration options for Hall's body force model are added to the configuration file under the header BODY FORCE DEFINITION.

A.4. Rotor Test Case

Figure A.1 visualizes the flow angle and x-momentum for the rotor test case.

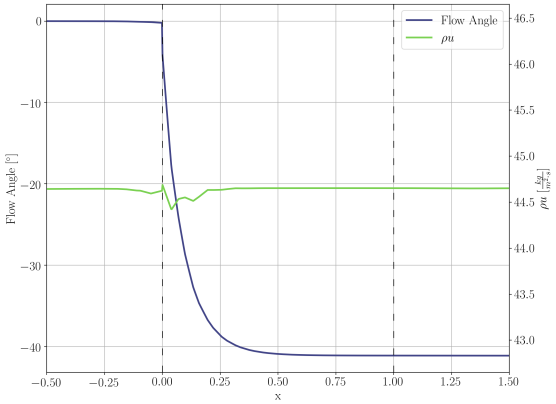


Figure A.1: *Flow angle and x-momentum for the rotor test case at* $\alpha = 30$, $B = 20$, *and* $\Omega = 500$.

A.5. Development into 3D-capable Code

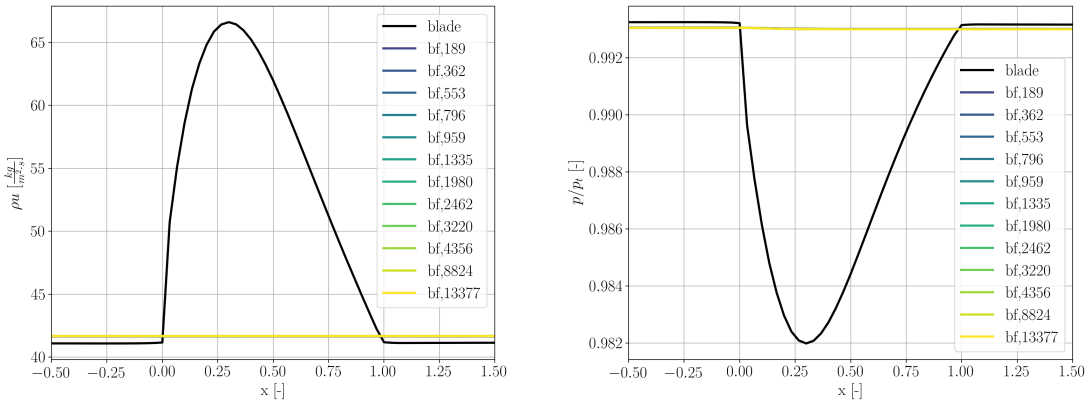
Supporting the recommendations given in Section 6.2 is a high-level action plan for implementing a 3D version of Hall's body force model. The following clarifies what is necessary on both a theoretical and practical level to convert the code into one that works for 3D simulations. If further information is required the reader should contact the author or Dr. Jeffrey Defoe at the University of Windsor's MAME faculty.

- Create a standard template for the arrays x, $\hat{n}_x$, $\hat{n}_y$, and $\hat{n}_z$. Write a piece of code (in Python) that creates this standard template using a number of input variables.
- Add capability for reading camber normal arrays.
- Ensure that code in the function *CEulerSolver::ComputeBodyForce_Turbo* accepts both 2D and 3D arrays.
- Go through all code relevant to body force computation to make sure that it worked independent of the number of dimensions.
- Extend interpolation of camber normals to $\hat{n}_z$ and $\hat{t}_z$.
- Make constant and variable body force different from each other so that they don't mistakenly interact.

Once these changes have been made it is required to perform simulations with test cases to confirm that the implementation behaves in a similar manner to that of physical blade simulations. This ensure that the implementation is consistent with physical behavior.

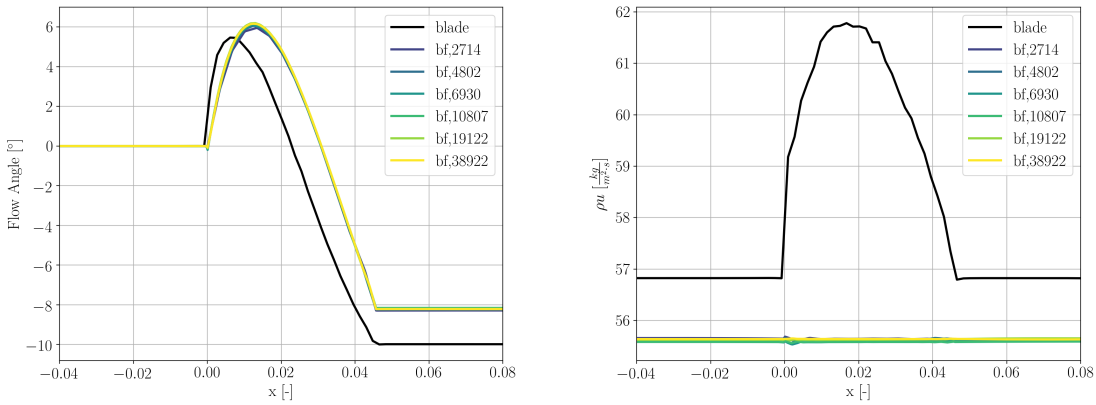
A.6. Computational Time

Figure A.2 and A.3 show a number of flow characteristics for the stator and Whittle fan computational time test cases.



(a) *Momentum in x-direction* ***(b)*** *Pressure Ratio*

Figure A.2: *Momentum in x-direction and pressure ratio are plotted over the domain for a number of body force simulations with descending number of cells. Decreasing the number of cells does not have a large impact on the flow with a 189-cell mesh giving essentially the same results as one with 13377 cells. These runs are for a stator at $\alpha = 5$ and $B = 20$; the number of cells used in the blade computation is 12903.*



(a) *Flow Angle* ***(b)*** *Momentum in x-direction*

Figure A.3: *Flow angle and x-momentum are plotted over the domain for a physical blade simulation of the Whittle fan ($\Omega = 2362$) as well as body force simulations with a range of mesh refinement. Decreasing the number of cells has no significant impact on the body force model's representation of the flow, the flow's trends are accurately predicted for the entire range of meshes.*

B

Adjoint Implementation

This appendix contains supporting information regarding the adjoint implementation.

B.1. Custom Objective Function

Included in SU2 are a number of objective functions that can be used with the adjoint solver, however, these require a physical blade to be present in the simulation to work. When used with body forces the results are meaningless, producing an adjoint flow with values extremely close to zero. Figure B.1 shows the adjoint density for three different objective functions: velocity in y-direction (custom objective function), outlet flow angle, and entropy generation. The second and third are in-house functions and as such produce insignificant results. A novel objective function can be created from scratch in the source code, however this can be time consuming and is not aligned with the goal of this thesis. To show the adjoint capability it is easiest to adapt a current objective function (such as drag, lift, or moment) and replace it with one relevant to the test case. In this case it is the vertical velocity, as it is an indicator of flow turning.

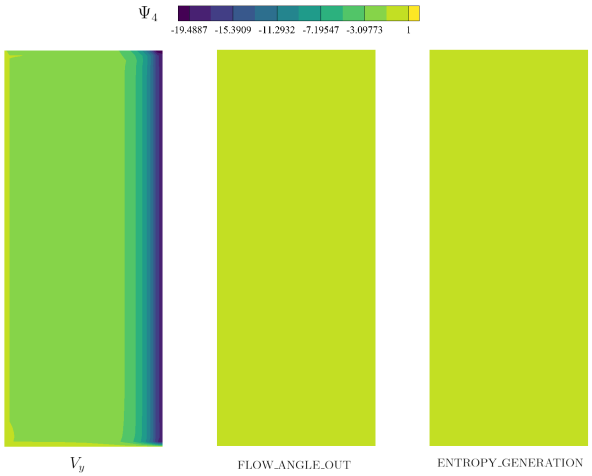
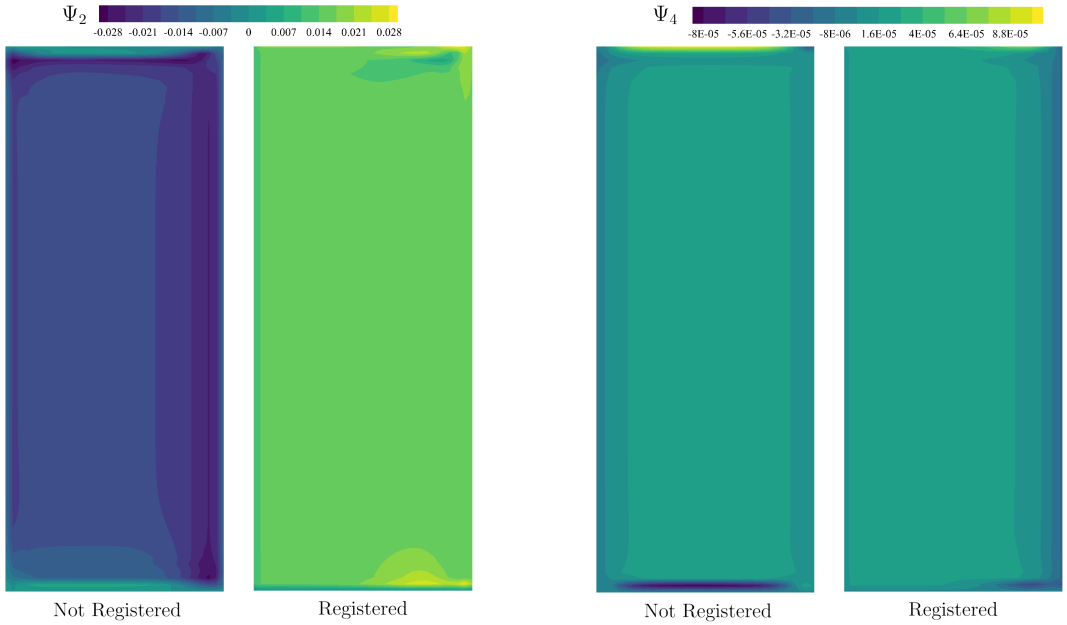


Figure B.1: *Adjoint density Ψ_1 in the body force zone for three different objective functions (for non-cambered stator at $\alpha = 5$). Using the solver's native objective functions leads to erroneous results as they require a blade to be present in the computational domain.*

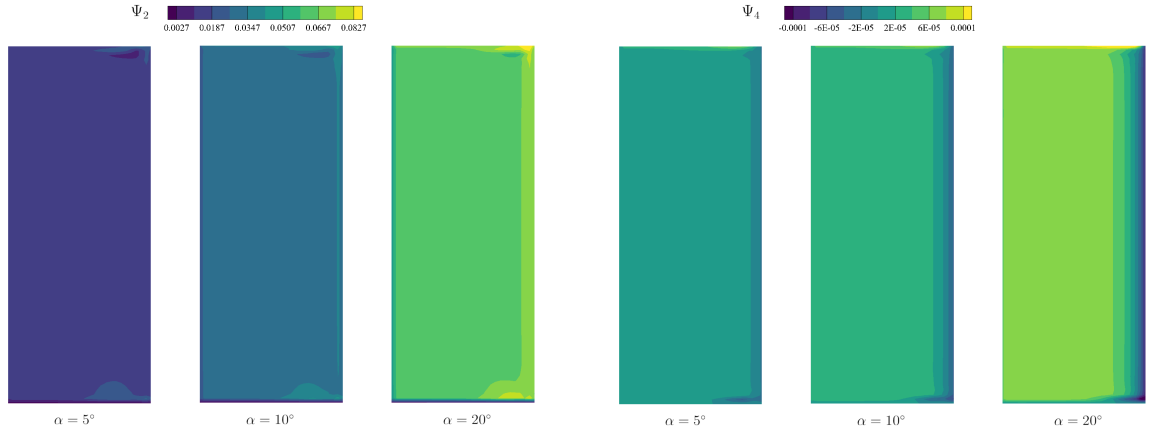
B.2. Mesh Sensitivity

Figures B.2 and B.3 show the adjoint x-momentum and energy results for the adjoint implementation. In Figure B.2 a comparison is made between non-registered and registered source terms. Figure B.3 shows the adjoint flow for varying angle of attack.



(a) Ψ_2 *(b)* Ψ_4

Figure B.2: *Adjoint x-momentum* Ψ_2 *and energy* Ψ_4 *for non-cambered stator at* $\alpha = 5$, $B = 20$, *and* $R = 1$, *optimized for cumulative* V_y *at outflow of body force zone.*



(a) Ψ_2 *(b)* Ψ_4

Figure B.3: *Adjoint x-momentum* Ψ_2 *and energy* Ψ_4 *for non-cambered stator at* α*: 5, 10, and 20. Simulation is optimized for cumulative* V_y *at outflow of body force zone.* $B = 20$ *and* $R = 1$.

B.3. Source Code Adaptation for the Adjoint Solver

Listed below are the files that are adapted for the adjoint solver to compute the mesh sensitivity $\frac{dJ}{dX}$ and total gradient $\frac{dJ}{d\alpha}$ with the body force source term registered.

B.3.1. Mesh Sensitivity

CSolver

solver_adjoint_discrete.cpp	*CDiscAdjSolver::SetRecording* includes a loop resetting the body force source term to the initial converged solution.
solver_adjoint_discrete.cpp	*CDiscAdjSolver::RegisterSolution* includes a loop registering body force source term as input for the adjoint computation
solver_adjoint_discrete.cpp	*CDiscAdjSolver::RegisterOutput* includes a loop registering body force source term as output of the adjoint computation

solver_adjoint_discrete.cpp	*CDiscAdjSolver::SetAdjoint_Output* includes a loop gets body force source term and adds it to the auxiliary solution vector. The derivative value of the auxiliary solution vector is then initialized as well.
solver_adjoint_discrete.cpp	*CDiscAdjSolver::ExtractAdjoint_Solution* includes a loop that gets the direct and derivative values of the body force source term from the auxiliary solution vector and assigns them to the node.

CVariable

variable_adjoint_discrete.cpp	Defines and initializes the arrays BFSource_Direct and Adjoint_BFSource for saving of the direct and adjoint source terms of the body force to the node.
variable_structure.inl	Creates *GetBFSource_Direct* for CVariable and CDiscAdjVariable, *SetAdjoint_BFSource* and *GetAdjoint_BFSource* for CVariable, CEulerVariable, and CDiscAdjVariable, and *CVariable::RegisterBFSource*. These are used throughout the functions added to the solver_adjoint_discrete.cpp to ensure the adjoint solver works with the body force source term as input.
variable_structure.hpp	Adds pointers to the arrays BFSource_Direct and Adjoint_BFSource for use in the CVariable and CDiscAdjVariable classes. Add a number of functions from the variable_structure.inl file that are used to register body force source terms, get them from an array, or set them to an array.

B.3.2. Total Gradient

CConfig

option_structure.hpp	Added option CAMB_NORM to ENUM_RECORDING, which is used in iteration_structure.cpp as a condition if camber normal values are used as input.
config_structure.hpp	Added vectors for default camber normal, actual camber normal, and total gradient. These are used in calculations for storing values. Defined functions to register input and obtain derivative, which are used in config_structure.cpp.
config_structure.inl	Function *CConfig::GetBody_Force_Camb_Norm* created that returns camber normal array.
config_structure.cpp	Defined and initialized default camber normal and total gradient arrays. Added option BODY_FORCE_CAMB_NORM to configuration file, allowing camber normal vector to be specified in config.cfg. Defined function *CConfig::Register_Camb_Norm* that uses *AD::RegisterInput* to register camber normal values as input for adjoint. Defined function *CConfig::TotalGrad_Camb_Norm* that gets derivative of camber normal (total gradient) and assigns the value to the total gradient array.

CDriver

driver_structure.cpp	Changed *SetRecording* input to CAMB_NORM.

CIteration

iteration_structure.cpp	Added conditions statement in *CDiscAdjFluidIteration::RegisterInput* that calls *CConfig::Register_Camb_Norm* if condition CAMB_NORM is met.

CSolver

solver_adjoint_discrete.cpp	Added call to total gradient computation function *CConfig::TotalGrad_Camb_Norm* in *CDiscAdjSolver::SetSensitivity*.

B.4. Total Gradient Validation

Computing the total gradient $\frac{dJ}{d\boldsymbol{\alpha}}$ can be used as an initial indication of its magnitude but also as validation of the output from SU2. Once the capability is included this step is necessary to evaluate the accuracy of the implementation. Presented below are the steps to obtain an analytical computation of the total gradient.

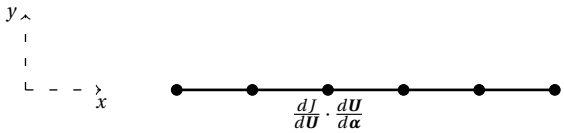


Figure B.4: *Computation of total gradient using flow output at each node. Flow adjoint is taken from the adjoint solver for each node while $\frac{d\boldsymbol{U}}{d\boldsymbol{\alpha}}$ must be computed by hand.*

As shown in Section 3.4.1, once the influence of the mesh is removed from the computation of the total gradient, all that is left is the flow and the design variables. To obtain the total gradient, Equation B.1 must be solved. When running SU2's adjoint solver, its output is the adjoint flow equation in Equation B.2. For each of the conservative variables (four in the 2D case) a value is obtained. For each of these conservative variables the sensitivity with respect to the design variables (camber normal values) must be computed (Equation B.3); this is not an output from SU2 but must be calculated separately.

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{dJ}{d\boldsymbol{U}} \cdot \frac{d\boldsymbol{U}}{d\boldsymbol{\alpha}} \tag{B.1}$$

$$\frac{dJ}{d\boldsymbol{U}} = \boldsymbol{\Psi} = \begin{bmatrix} \frac{dJ}{d\boldsymbol{U}_1} & \frac{dJ}{d\boldsymbol{U}_2} & \cdots & \frac{dJ}{d\boldsymbol{U}_n} \end{bmatrix} \tag{B.2}$$

$$\frac{d\boldsymbol{U}}{d\boldsymbol{\alpha}} = \begin{bmatrix} \frac{d\boldsymbol{U}_1}{d\boldsymbol{\alpha}_1} & \frac{d\boldsymbol{U}_1}{d\boldsymbol{\alpha}_2} & \cdots & \frac{d\boldsymbol{U}_1}{d\boldsymbol{\alpha}_m} \\ \frac{d\boldsymbol{U}_2}{d\boldsymbol{\alpha}_1} & \frac{d\boldsymbol{U}_2}{d\boldsymbol{\alpha}_2} & \cdots & \frac{d\boldsymbol{U}_2}{d\boldsymbol{\alpha}_m} \\ \vdots & & \ddots & \vdots \\ \frac{d\boldsymbol{U}_n}{d\boldsymbol{\alpha}_1} & \frac{d\boldsymbol{U}_n}{d\boldsymbol{\alpha}_2} & \cdots & \frac{d\boldsymbol{U}_n}{d\boldsymbol{\alpha}_m} \end{bmatrix} \tag{B.3}$$

Computing $\frac{d\boldsymbol{U}}{d\boldsymbol{\alpha}}$ and subsequently $\frac{dJ}{d\boldsymbol{\alpha}}$ is simplified by the fact that there is tangential uniformity in the flow. For every node in the x-direction, the flow adjoint can be multiplied by the flow sensitivity to obtain the total gradient (see Figure B.4). At every node $\hat{n}_x$ and $\hat{n}_y$ are synonymous with the design variables α_1 and α_2. By multiplying both matrices with each other a total gradient matrix per node is obtained, which in this case is $\frac{dJ}{d\boldsymbol{\alpha}} = \begin{bmatrix} \frac{dJ}{d\boldsymbol{\alpha}_1} & \frac{dJ}{d\boldsymbol{\alpha}_2} \end{bmatrix}$. An example of this calculation is presented in Equation B.4. Validation can then be performed with this output.

$$\frac{dJ}{d\boldsymbol{\alpha}} = \begin{bmatrix} \frac{dJ}{d\boldsymbol{U}_1} & \frac{dJ}{d\boldsymbol{U}_2} & \frac{dJ}{d\boldsymbol{U}_3} & \frac{dJ}{d\boldsymbol{U}_4} \end{bmatrix} \cdot \begin{bmatrix} \frac{d\boldsymbol{U}_1}{d\boldsymbol{\alpha}_1} & \frac{d\boldsymbol{U}_1}{d\boldsymbol{\alpha}_2} \\ \frac{d\boldsymbol{U}_2}{d\boldsymbol{\alpha}_1} & \frac{d\boldsymbol{U}_2}{d\boldsymbol{\alpha}_2} \\ \frac{d\boldsymbol{U}_3}{d\boldsymbol{\alpha}_1} & \frac{d\boldsymbol{U}_3}{d\boldsymbol{\alpha}_2} \\ \frac{d\boldsymbol{U}_4}{d\boldsymbol{\alpha}_1} & \frac{d\boldsymbol{U}_4}{d\boldsymbol{\alpha}_2} \end{bmatrix} \tag{B.4}$$