Adjoint Based Optimisation for Coupled Conjugate Heat Transfer

by

Oluwadamilare Rahman Imam-Lawal

Supervisor: Dr. Jens-Dominik Müller

Technical advisor: Prof. Tom Verstraete

Submitted in partial fulfilment of the requirements of the Degree of Doctor of Philosophy

School of Engineering and Material Science

Queen Mary, University of London

April 29, 2020

Oju eni maa la, a ri iyonu

-Yoruba proverb

Acknowledgements

The end of this journey is an observation point that allows me to compute a unique set of gradients. Who are the individuals that have positively influenced my ability to complete this work? Unfortunately, I have no adjoint to bail me out, only words which do not do justice...

Firstly, I would like to thank my supervisor Dr. Jens-Domink Müller who has inspired and guided me from my first year as an undergraduate student learning the fundamentals of fluid dynamics, to my final year undergraduate project introducing me to the fascinating world of research and CFD, and finally to giving me the opportunity to undertake this thesis. You have helped me mature a lot as a student and a person and I cannot thank you enough.

The support of Professor Tom Verstraete has been invaluable and his expertise has helped shape this thesis. Thank you for adopting me as your student and unofficially becoming my second supervisor. I am grateful for all your advice and for always making time out to discuss my progress (or lack off). I have learnt a lot under your supervision and will be forever grateful.

I am also grateful to QMUL for funding and for the opportunity to conduct this work. To my colleagues Mateusz, Kumar, Rejish, Orest, Anam, Awais, Stephen, Francesco, and more at QMUL, you have all enriched my university experience in various ways.

Finally, I would like to thank my family and friends for their love, support, and constant encouragement. I would also like to thank Bidemi for her support and understanding, especially during the difficult moments.

Statement of Originality

I, Oluwadamilare Rahman Imam-Lawal, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Signature: Oluwadamilare Rahman Imam-Lawal
Date: April 29, 2020

Abstract

Conjugate Heat Transfer (CHT) problems are typically solved using a partitioned approach where separate solvers for the fluid and structure are loosely coupled through boundary conditions. These boundary conditions need to be updated iteratively until the temperature and heat flux are continuous between the two domains. In this thesis, CHT problems are solved by coupling the in-house flow solver with the open-source heat conduction solver, CalculiX. Three coupling algorithms are used, which involve a combination of Neumann, Dirichlet, and Robin boundary conditions.

The interest in shape optimisation has increased the need for efficient optimisation techniques. Consequently, gradient based optimisation using adjoint methods are preferred due to the reduced computational cost of obtaining gradients. Currently, the use of adjoint methods in CHT shape optimisation problems mostly favours the continuous adjoint method which suffers from high developmental costs. This thesis advocates for the use of the discrete adjoint via Automatic Differentiation (AD) as a cost-effective alternative to the continuous adjoint. This is done by differentiating the fluid and solid solvers with respect to the coupling boundary conditions using AD. A fully differentiated partitioned coupling approach is achieved by differentiating the three coupling algorithms used. The differentiation of the Robin boundary conditions results in two new differentiated coupling algorithms and the accuracy of the differentiated coupling algorithms is demonstrated by comparing with gradients obtained through finite differences.

The efficacy of the developed methods is then demonstrated on three CHT optimisation problems: An inverse design problem related to a flat plate and a thermal optimisation of the MarkII turbine blade and an internal cooling channel U-Bend. The gradient verification and optimisation results revealed that the use of Robin boundary conditions in the flow solver reduces computational runtime.

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

Greek Symbols

α	Design variable
δ_{ij}	Kronecker delta
γ	Specific heat ratio
λ_f	Fluid thermal conductivity
λ_s	Solid thermal conductivity
μ	Dynamic viscosity
ν	Kinematic viscosity
Ω	Domain
ρ	Density
$\overline{\tau}$	Friction force tensor
σ	Stefan-Boltzmann constant

General Symbols

Bi	Biot number
c	Speed of sound
Cr	Corrective coefficient
d	Diameter
D_h	Hydraulic diameter
e	Internal energy
$\mathbf{F}$	Flow solver
F_c	Convective flux
F_v	Viscous flux
g	Constraint
h	Convective heat transfer coefficient
H	Enthalpy
J	Objective/cost function
k	Thermal conductivity

K	Stiffness matrix
L	Characteristic length
M	Mach number
n	Normal direction
$\vec{n}$	Normal direction vector
N	Shape function
Nu	Nusselt number
p	Pressure
P	Static pressure
Pr	Prandtl number
q	Heat flux
Q	Source term
$\vec{R}$	Residual vector
Re	Reynolds number
Res_T	Coupling (Temperature) residual
Res_J	Coupling (Objective function) residual
$\mathbf{S}$	Heat solver
T	Temperature
t	Time
$\vec{v}$	Velocity vector

Subscripts

b	Bottom
f	Fluid
fw	Fluid wall
j	Node index
L	Laminar
s	Solid
$sink$	Sink/ambient temperature for Robin boundary condtions

sw	Solid wall
T	Turbulent
$wall$	Wall quantiity
∞	Freestream quantitiy

Superscripts

i	Iteration index

Abbreviations

1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
AD	Automatic Differentiation
BC	Boundary conditions
BFGS	Broyden-Fletcher-Goldfarb-Shanno (optimiser)
CAD	Computer Aided Design
CD	Central-Differences
CFD	Computational Fluid Dynamics
CHT	Conjugate Heat Transfer
CSM	Computational Structural Mechanics
FD	Finite-Differences
FEM	Finite Element Method
FSI	Fluid-Structure Interaction
FVM	Finite Volume Method
hFRB	heat transfer coefficient Forward solid coefficient Back (coupling algorithm)
IDW	Inverse Distance weighting
MDA	Multidisciplinary Design Analysis
MDO	Multidisciplinary Design Optimisation
NSPCC	NURBS-based Parameterisation with Continuity Constraints (CAD kernel)
NURBS	Non-uniform Rational Basis Spline
PDE	Partial Differential Equation
RANS	Reynolds Averaged Navier-Stokes
SA	Spalart-Allmaras
TFFB	Temperature Forward Flux Back (coupling algorithm)
TFRB	Temperature Forward solid coefficient Back (coupling algorithm)

Chapter 1

Introduction

1.1 Multidisciplinary Design Analysis

When a fluid at a certain temperature flows past a solid of differing temperature, heat is transferred between both media due to the temperature difference. The nature of the heat transfer is dependent on factors such as the fluid velocity, flow conditions (e.g. laminar or turbulent), the material properties of the fluid and solid (e.g. solid thermal conductivity), and the temperature difference between both media.

As the hotter medium, e.g a solid plate, heats up the cooler medium, e.g the air flowing past, the hotter medium is simultaneously being cooled down by the cooler medium. Consequently, the final temperature at the boundary between these two media is a result of the heat transfer between the media. This means a simultaneous solution of the heat transfer between both media is required to accurately predict the temperature at the fluid-solid interface. This simultaneous solution, which accounts for the interaction and dependency between both the solid plate and the air flowing past, is known as Conjugate Heat Transfer (CHT).

Conjugate Heat Transfer has several important applications in the modern world. For example, the electrical loads in power transformers and in the micro electronics in modern computers result in high temperatures in these devices. Similarly, environmental requirements and policy changes in the aerospace industry result in the need for more efficient jet engines and higher temperatures in their turbines. Consequently, some form of cooling is needed in these components and CHT is required to accurately predict the local temperature and heat transfer.

CHT is an example of a multidisciplinary problem and considers the transfer of heat between a fluid and solid. Another example is Fluid-Structure Interaction (FSI) which considers the interaction of aerodynamics and structural mechanics, e.g. how fluid flow bends/deforms the wings of an aircraft. The maturity of tools for Computational Fluid Dynamics (CFD) and Computational Structural Mechanics (CSM) enable the solution to multidisciplinary problems to be obtained through the use of numerical simulations. This is known as Multidisciplinary Design Analysis (MDA). MDA is used to provide insight on the fitness for purpose of a given design and provide information about how each discipline interacts with the other [1].

The accuracy of MDA is important as incorrect analysis will hamper the ability to obtain good designs. MDA can be done using a monolithic or partitioned approach. The monolithic approach uses a single numerical solver for the equations of all disciplines while the partitioned approach uses separate solvers for each discipline. The partitioned approach (also called segregated approach) is more common due to its flexibility and requires the iterative exchange of boundary conditions between numerical solvers.

1.2 Multidisciplinary Design Optimisation

The use of numerical simulation tools for MDA has helped to shorten and cheapen the design cycle by reducing the number of prototypes which are built and tested. Although MDA enables one to test designs in a few hours, it does not solve the problem of the iterative nature of the design process. The classical process of combining experiments or numerical analysis with trial and error or educated guesses from experienced designers is highly inefficient and provides no guarantee that the final design is truly optimal. This process can be improved upon through the use of numerical optimisation.

As the optimum is quantifiable in most engineering design problems, numerical optimisation can be used to rationally and objectively find optimal designs [1]. Numerical optimisation is more efficient than trial and error as it makes use of logical methods to evaluate designs and enables the automation of the design process. This helps to further reduce the cost and time of the design process.

The optimisation goal is to provide the best design which meets a given requirement. For example, minimising an objective function, J, (e.g. drag), with respect to some design variables, α, (e.g. the shape of the wing), subject to some constraints, X, such as maintaining the cross-sectional area of the wing.

Multidisciplinary Design Optimisation (MDO) is the application of numerical optimisation methods to solve multidisciplinary design problems [2]. Examples of MDO include, for instance the aero-structural optimisation of aircrafts in [3] [4] [5] and the aero-thermal optimisation of turbine blades [6] and microelectronic coolers [7].

In order to optimise numerous engineering problems of interest, several disciplines need to be considered if a true optimum is to be attained. However, the consideration of interacting systems greatly increases the complexity of the MDO problem. By considering the individual performance of the disciplines as well as their interactions, MDO results in better designs than those obtained by optimising each discipline sequentially [8]. This approach is more efficient, does not require simplification of the constrains imposed by other disciplines, and is more likely to attain the best design (global minimum) [9][6].

1.3 Gradient based optimisation

As the number of design variables increases, the ability of a designer to rely on intuition and experience diminishes. Conversely, the use of a large number of design variables is often more desirable in numerical optimisation as it provides a rich design space. Consequently, the use of optimisation algorithms is essential for MDO. Optimisation algorithms can broadly be grouped into either gradient free or gradient based methods.

The output of MDA is a quantitative assessment of performance which is known as the objective function. Gradient free methods rely solely on the results of the analysis to drive the optimisation. The disadvantage of gradient free methods is that as the number of design variables increases, the required number of function evaluations increases significantly and soon becomes prohibitive [3]. Some examples of gradient free optimisation in literature include grid searching, evolutionary algorithms, and particle swarm optimisation. Although gradient free methods are easier to use, gradient based methods are the most feasible way to solve problems with 10^2 or more design variables [1].

Gradient based methods use the objective function value and gradient (with respect to design variables) to drive the optimisation process. They generally require fewer evaluations than gradient free methods, especially for large design spaces, hence they are preferred. However, their effectiveness is limited to functions which vary smoothly, and they only guarantee convergence to a local minimum/optimum [3].

Gradient based MDO requires the computation of gradients of the objective function with respect to the design variables which is also known as sensitivity analysis (see Figure

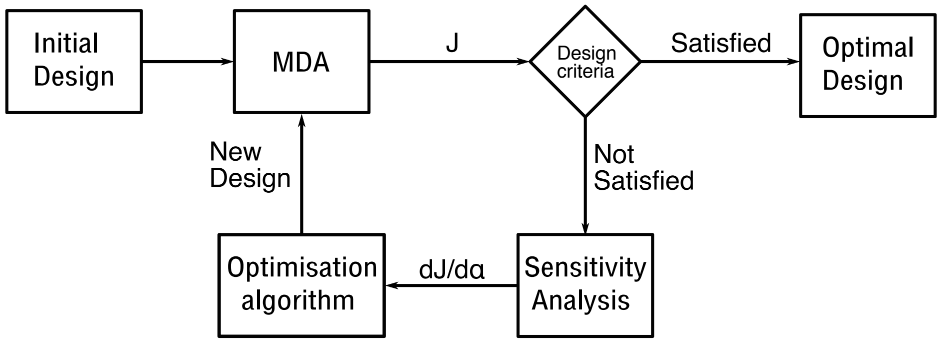


Figure 1.1: A generic gradient based MDO procedure.

1.1). Obtaining accurate derivatives is of paramount importance for gradient based methods. This is because the accuracy of the obtained gradients affects the path to obtaining an optimal design. For example, inaccurate gradients may lead to less direct routes to the optimum being taken, which may also require more design iterations, thereby elongating the design process.

Although gradient based methods are computationally cheaper to use, their cost can still be prohibitive in optimisation problems with a large number of design variables. Therefore it is necessary develop methods which result in efficient derivative computations, however, this requires some level of expertise as gradient computations are rarely included in design analysis tools.

1.4 Adjoint methods

The main methods for computing derivatives include finite-differences, complex step methods, tangent linearisation, and adjoint methods. Finite-difference methods are the easiest to use, however accuracy issues arise due to their dependence on the step-width and large computational effort is required for a large number of design variables. Complex step methods and tangent linearisation eliminate the accuracy issues faced by finite-differences but they still require large amount of computational resources for a large number of design variables [10]. This is because, for these three methods, the cost of the gradient calculation scales linearly with the number of design variables.

Adjoint methods are particularly attractive because accurate derivatives can be computed at a cost which is almost independent of the number of design variables. Therefore, adjoint methods are the most efficient way to obtain derivatives when the number of design variables is greater than the number of objective functions. Adjoint methods can be grouped into continuous or discrete methods. In the continuous method the adjoint equations are analytically derived before being discretised while the discrete method discretises the state equations before formulating the discrete adjoint equations.

Both the continuous and discrete adjoint have their advantages, however, the continuous approach involves lengthy hand derivation of the adjoint Partial Differential Equations (PDEs) which can be error-prone. In contrast, the use of a Automatic Differentiation (AD) for developing discrete adjoint codes eases the burden of code development and maintenance. Automatic Differentiation works by differentiating each line of computer code and using the chain rule to calculate the required gradients. This results in analytically accurate gradients with significantly less effort. Furthermore, discrete adjoint gradients are able to be verified using the analytically accurate tangent linearisation, an option unavailable for the continuous adjoint [11]. This makes the discrete adjoint a more attractive option and this is what is used in the current work.

The use of adjoint methods in CHT optimisation is recent field of study and has been dominated by the continuous adjoint formulation [12] [13] [14] [15] [16]. On the other hand, the use of discrete adjoints for CHT is more rare and recent [17] [18], and the application of AD in the development of adjoint codes for CHT has not been explored.

1.5 Motivation of thesis

Gradient free methods are currently the most popular methods used for CHT optimisation. For example, they have been successfully applied to turbo-machinery optimisation problems in [6] [19] [20] [21]. However, these methods are not the most efficient as they can be computationally expensive, especially when large design spaces are being considered. The use of adjoint methods for the optimisation of CHT problems is relatively new and requires further development.

Moreover, when adjoint methods have been used for CHT shape optimisation problems, they have been limited to the continuous adjoint formulation. Furthermore, these often include several simplifications to the problem such as, not developing the solid adjoint Partial Differential Equations (PDEs) [12], not solving the solid governing equations [13], not developing the fluid adjoint PDEs [15], or not differentiating the turbulence models [14]. Only recently a fully coupled continuous adjoint was developed without any simplifications made in both domains [16]. This may be a consequence of the labour-intensive nature of developing the continuous adjoint as it requires derivation of the adjoint PDEs by hand. The continuous adjoint method also makes code maintenance tedious as newly added features (such as boundary conditions and objective functions) also need to be derived.

The use of the discrete adjoint for CHT optimisation is yet to be explored and developed in depth. Recently, the discrete adjoint was used in a CHT optimisation by He et al [17]. However, a partitioned approach using separate solvers for each domain was not used and finite differences were used to calculate gradients. Burghardt and Gauger [18] also use the discrete adjoint but with a monolithic solver. To the author's knowledge, no work has been done using discrete adjoint methods for partitioned CHT optimisation. Moreover, even when partitioned continuous adjoint formulations are considered [15] [16], a limited selection of coupling algorithms have been considered.

The advantages provided by the use of Automatic Differentiation in the development of discrete adjoint codes is yet to be fully exploited in CHT problems. The application of AD significantly alleviates the difficulty of code maintenance as new boundary conditions and objective functions are automatically differentiated by the AD tool. Consequently, the following problems can be identified:

1. Adjoint methods for CHT problems require developments free from simplifications and which consider all aspects of the coupled problem.

2. The effort of developing and maintaining adjoint codes needs to be reduced and the continuous adjoint is unable to solve this as it requires the hand derivation of adjoint PDEs.

3. The selection of boundary conditions used for CHT optimisation needs to be expanded to keep abreast of recent advancements in CHT coupling algorithms. Current work has not focused enough on the coupling boundary conditions. Particularly, how the multidisciplinary derivatives are obtained and how these derivatives are exchanged between domains.

4. Adjoint methods for CHT should enable the use of partitioned coupling methods and should be generic enough to be applied to most numerical solvers.

To this end, the current work addresses these issues through the following contributions:

1. The development of a coupled discrete adjoint method for partitioned coupling approaches which fully considers both the solid and fluid governing equations.

2. The application of Automatic Differentiation (AD) to open-source codes to obtain adjoint solvers without pain-staking hand derivation of the adjoint PDEs. This also includes the automation of the process and accounts for the easy addition of new boundary conditions and objective functions.

3. The consideration and differentiation of three coupling algorithms, including two recently developed coupling algorithms.

4. The development of a framework for CHT optimisation, using to two open-source solvers, to highlight the generic nature of the proposed methods.

5. Application of developed methods and solvers to challenging optimisation problems to highlight the effectiveness of the proposed methods.

1.6 Thesis outline

This thesis is organised as follows: Chapter 2 discusses Multidisciplinary Design Analysis with a focus on Conjugate Heat Transfer and methods of solving coupled CHT problems are discussed. The coupling algorithms and numerical solvers used as well as implementation details are presented. The selected coupling methods and solvers are then evaluated in Chapter 3 where MDA is performed on three CHT problems and the results are compared to analytical and experimental data.

Chapter 4 discusses the derivation of adjoint equations for single disciplines, the adjoint solvers developed in this work are presented, and the gradients verified. Chapter 5 focuses on obtaining accurate gradients and the coupling algorithms used are differentiated. The accuracy of the differentiated algorithms is then evaluated by comparing with finite differences.

In Chapters 6 & 7, the developed adjoint method is then used to solve three CHT optimisation problems. Finally, the results and contributions are summarised in Chapter 8 and recommendations are made for further research.

Chapter 2

Governing equations and Numerical methods

Conjugate Heat Transfer (CHT) refers to situations in which strong thermal interactions between fluids and solids occur [22]. An example of this is in electrical components such as transformers used for power distribution, and in the microchips/micro-elctronics used for modern computing [7]. The electrical loads in these devices result in energy losses in the form of heat and cooling fluids may be used in order to keep temperatures down to a reasonable value. This requires a CHT problem to be solved in order to accurately predict the thermal interaction between the cooling fluid and hot components.

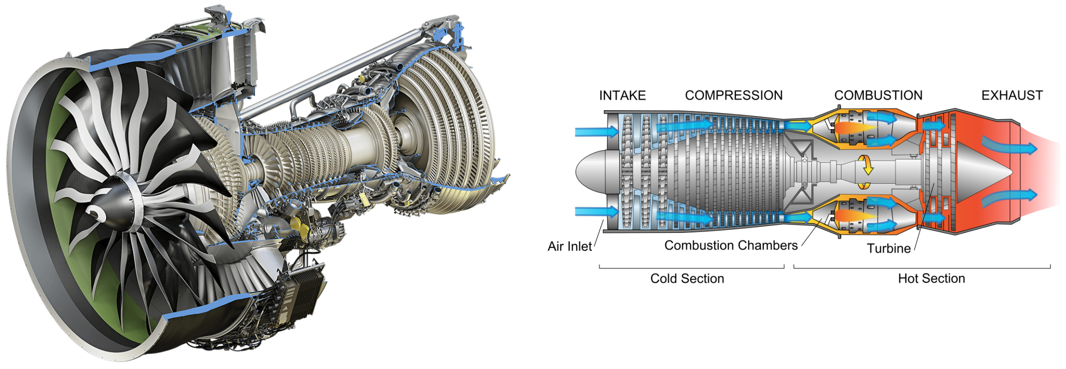


Figure 2.1: Left: GE9X engine (source: GE Aviation). Right: Cross sectional view of gas path.

Another interesting application of CHT is in the engines of modern aircrafts. In order to provide thrust, jet engines typically function by sucking in and compressing air through the use of a fan and several compressor stages. The compressed air is then mixed with fuel and ignited in the combustion chamber. The ignited gas then expands outwards from the combustion chamber towards the turbine blades and exist through the exhaust nozzle to provide the required thrust (see Figure 2.1).

In order to improve efficiency, higher turbine inlet temperatures are favourable because this improves the specific thrust and specific fuel consumption of the engine. Consequently, turbine blades are subjected to turbine inlet temperatures up to 2000 K which often exceeds the melting point of the materials they are made of [23] [24]. As a result, cooling is required to preserve the integrity of the blades. A common method of cooling involves passing cooler air through the inside of the blades and is known as convection cooling. This results in a CHT problem involving heat transfer between the turbine blade, cooling internal flow, and hot external flow [25] [6].

Analytic methods for solving CHT problems are limited to simple configurations, while experiments, although essential for validation, are expensive. Moreover, in the design of new components, it is not always clear how new designs may affect the thermal interactions between the fluid and solid. Consequently, numerical solutions using MDA are key to solving CHT problems [22].

This chapter discusses the governing equations and numerical methods for CHT analysis. The numerical solvers used to solve the governing equations are introduced and details of the implementation of the necessary boundary conditions in the flow solver are provided. The three coupling algorithms used for solving CHT problems are also described.

2.1 Conjugate Heat Transfer

Heat transfer is the transit of thermal energy and can be categorised into three different modes namely: Conduction, Convection, and Radiation.

Conduction is the transfer of thermal energy from higher to lower temperature regions within a medium due to atomic and molecular activity. Conduction is governed by Fouriers law:

$$q = -\lambda \cdot \nabla T, \tag{2.1}$$

where q is the heat flux, λ is the material thermal conductivity, and T is the temperature.

Convection is energy transfer due to a combination of the bulk motion of a fluid and the diffusion of molecules. It can be modelled globally by Newton's law of cooling:

$$q = h(T_w - T_\infty), \tag{2.2}$$

where h is the heat transfer coefficient, while T_w and T_∞ are the wall and fluid freestream temperatures respectively. The heat transfer coefficient represents the thermal resistance of the boundary layer and can be influenced by the fluid properties and the nature of the flow.

Radiation is energy emission by matter with non zero temperature. Radiation energy is transported in the form of electromagnetic waves and does not require a material medium to be present. Radiation is governed by the Stefan-Boltzmann law:

$$q = \sigma T^4 \tag{2.3}$$

where σ is the Stefan-Boltzmann constant $\sigma = 5.67 \cdot 10^{-8}$ W/m^{2}K^4, q is the heat flux, and T is the absolute temperature [26].

This thesis focuses on Conjugate Heat Transfer meaning problems in which only conduction and convection are considered.

2.2 Solid governing equations and numerical solver

The solid domain is governed by the steady state heat conduction equation

$$\lambda \nabla^2 T = 0. \tag{2.4}$$

The heat conduction solver used is CalculiX (2.13) developed by Dhondt and Wittig [27], written in a mix of C and Fortran 77. CalculiX uses the Finite Element Method (FEM) to solve the heat equation and supports the use of a wide variety of element types such as wedge, hexahedral, and tetrahedral elements. The FEM discretises the solid domain (Ω_s) into finite elements which are all connected at the nodes. The temperature in each element is interpolated between the nodal values through the use of shape functions

$$T(x,y,z) = \sum_{i=1}^{M} N_i(x,y,z) T_i, \tag{2.5}$$

$$T = [N]\{T\}, \tag{2.6}$$

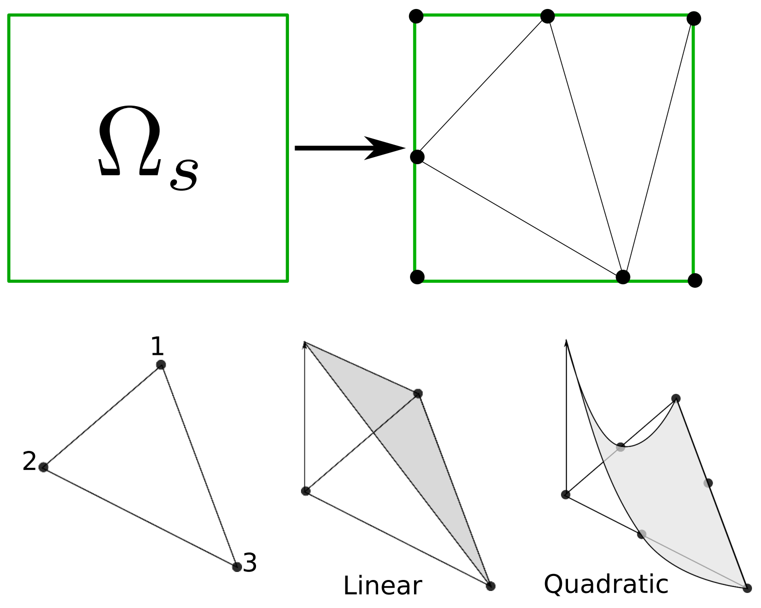


Figure 2.2: Solid domain discretisation and a three node element with linear and quadratic shape functions.

where M is the number of mesh points, $\{T\}$ is the vector of nodal temperatures and $[N]$ is the matrix of shape functions (see figure 2.2). Similarly, the temperature gradient interpolation is obtained though the temperature gradient matrix (which is analogous to the strain matrix for FEM stress calculations). The x component of temperature gradient can be calculated as

$$\left[\frac{dT}{dx}\right] = \left[\frac{\partial N}{\partial x}\right]\{T\}, \tag{2.7}$$

$$\left[\frac{dT}{dx}\right] = [B]\{T\}, \tag{2.8}$$

where $[B]$ is the temperature gradient interpolation matrix. In this work, only steady state analysis is performed, therefore by multiplying (Equation 2.4) by the shape functions, the weak form of the heat equation can be represented as

$$\int_{\Omega_s} \left(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z}\right) N_i d\Omega_s = 0, \tag{2.9}$$

Applying integration by parts transforms the volume integral into surface integral

$$\int_{\Omega_s} \Big(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z}\Big) N_i d\Omega_s = \int_s \{q\}^T \{n\} N_i dS, \tag{2.10}$$

where $\{n\}$ represents the three components of the surface normal and and $\{q\}^T = [q_x, q_y, q_z]$. Introducing the boundary conditions yields the governing set of finite element equations

$$\int_{\Omega_s} \Big(\frac{\partial N_i}{\partial x} + \frac{\partial N_i}{\partial y} + \frac{\partial N_i}{\partial z}\Big)\{q\} d\Omega_s = -\underbrace{\int_{s1} \{q\}^T \{n\} N_i dS}_{Dirichlet} + \underbrace{\int_{s2} q_{fw} N_i dS}_{Neumann} + \underbrace{\int_{s3} h(T_{sw} - T_{sink}) N_i dS}_{Robin}, \tag{2.11}$$

where q_{fw} is the heat flux from the fluid domain Ω_f into the solid domain Ω_s, T_{sw} is the temperature on the solid side of the interface, and T_{sink} is the ambient fluid temperature. The integrals over surfaces s_1, s_2, and s_3 refer to Dirichlet (temperature), Neumann (heat flux), and Robin (mixed) boundary conditions respectively as shown in Figure 2.3. The final boundary condition is a zero temperature gradient (adiabatic) boundary condition which is imposed by setting the $s1$ surface integral to zero. Equation (2.11) can be numerically

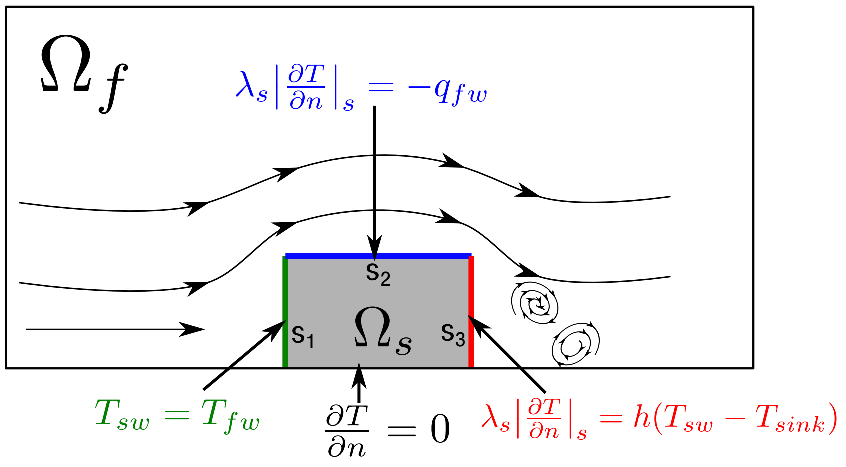

Ω_f
$\lambda_s \left|\frac{\partial T}{\partial n}\right|_s = -q_{fw}$
s_2
s_1
Ω_s
s_3
$T_{sw} = T_{fw}$
$\frac{\partial T}{\partial n} = 0$
$\lambda_s \left|\frac{\partial T}{\partial n}\right|_s = h(T_{sw} - T_{sink})$

Figure 2.3: Boundary conditions in the solid domain Ω_s.

integrated using Gaussian quadrature to obtain

$$[K]\{T\} = \{Q\}, \tag{2.12}$$

where K is the conduction matrix (consisting of the conduction and convective coefficients) and Q is the driving flux from the boundary conditions. CalculiX offers a choice of linear solvers to solve the system of equations in Equation (2.12) and the SPOOLES linear solver is used.

In CalculiX, the Dirichlet condition is imposed by specifying the nodal values of the temperature while the Robin boundary condition requires facial values of the sink temperature T_{sink} and heat transfer coefficient $\tilde{h}$ to be specified by the user [28].

2.3 Fluid governing equations and flow solver

The fluid domain (Ω_f in Figure 2.3) is governed by the Reynolds-Averaged Navier-Stokes (RANS) equations given by

$$\frac{\partial}{\partial t}\int_{\Omega_f}\vec{U}d\Omega + \oint_{\partial\Omega_f}(\vec{F}_c - \vec{F}_v)\cdot dS \quad = \quad \int_{\Omega_f}\vec{Q}d\Omega_f, \tag{2.13}$$

where t denotes pseudo time and Q the source term. The vector of conservative variables U, and the convective and viscous flux vectors $\vec{F}_c$ and $\vec{F}_v$ are defined as

$$\vec{U} = \begin{bmatrix}\rho \\ \rho\vec{v} \\ \rho e \\ \tilde{\nu}\end{bmatrix}, \vec{F}_c = \begin{bmatrix}\rho\vec{v}\cdot\vec{n} \\ (\rho\vec{v}\vec{v}+p)\cdot\vec{n} \\ \rho(e+p)\vec{v}\cdot\vec{n} \\ \tilde{\nu}\vec{v}\cdot\vec{n}\end{bmatrix}, \vec{F}_v = \begin{bmatrix}0 \\ \overline{\overline{\tau}}\cdot\vec{n} \\ \vec{\Theta}\cdot\vec{n} \\ \frac{1}{\sigma}(\nu_L+\tilde{\nu})(\nabla\nu\cdot\vec{n})\tilde{\nu}\end{bmatrix}. \tag{2.14}$$

Where, ρ, p, and $\vec{v}$ are the fluid density, pressure, and velocity vector respectively, $\vec{n}$ the surface normal vector, e the internal energy per unit mass, $\tilde{\nu}$ is the modified eddy viscosity, and

$$\begin{aligned}\vec{\Theta} &= \overline{\overline{\tau}}\cdot\vec{v} + \kappa\nabla T, && (2.15)\\ \overline{\tau} &= \mu(\nabla\vec{v} + \nabla\vec{v}^T) + \delta_{ij}\lambda\nabla\cdot\vec{v}, && (2.16)\\ k &= k_L + k_T = cp\left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T}\right), && (2.17)\\ \mu &= \mu_L + \mu_T, && (2.18)\end{aligned}$$

where laminar dynamic viscosity μ_L is calculated using Sutherland's Law, k_L and k_T are the laminar and turbulent thermal conductivities respectively, μ_T the turbulent eddy viscosity,

δ_{ij} is the Kronecker delta, and the ideal gas law and the Spalart-Allmaras turbulence model are used to close the fluid system of equations [29].

The fluid analysis is performed using the in-house mgOpt flow solver which is written in Fortran 90 and has been developed during previous PhD programs. mgOpt is a vertex centred, finite volume solver, which solves the 3-D compressible RANS equation using unstructured grids. Median dual control volumes are constructed from the original grid and a second order accurate spatial discretisation is used with JT-KIRK implicit time stepping [30]. mgOpt also uses geometric multigrid algorithms for convergence acceleration in which the fluid equations are solved on a hierarchy of coarsened grid levels. This allows the use of larger time steps on the coarser grids and for low frequency solution errors to be reduced more efficiently [11] [31] [32].

The spatial integral of Equation (2.13) is represented by a residual vector

$$\vec{R}(\vec{U}) = \oint_{\partial\Omega_i} (\vec{F}_c - \vec{F}_v) \cdot dS_i - \int_{\Omega_i} \vec{Q} d\Omega_i \tag{2.19}$$

where Ω_i refers to a control volume (see Figure 2.4). The convective flux is obtained using

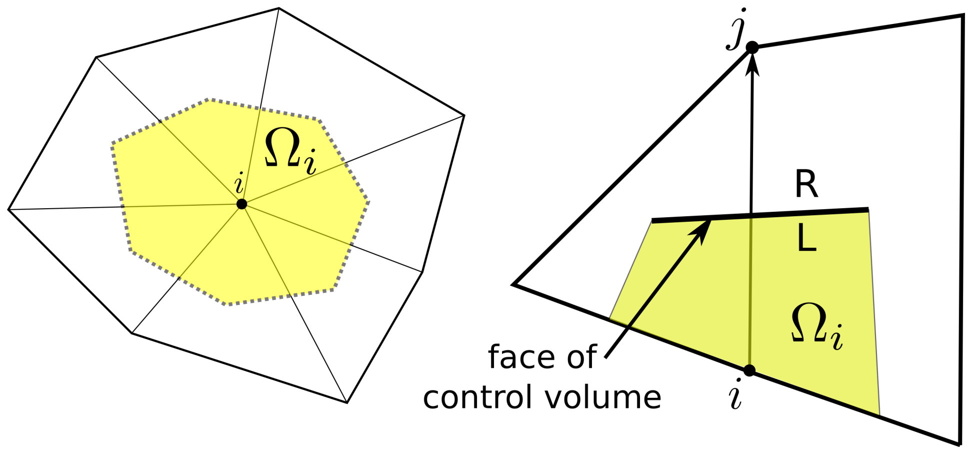


Figure 2.4: Left: 2D dual control volume, Right: Left and Right faces of a control volume.

the Roe approximate Reimann solver [33]. The total convective flux is obtained by summing the fluxes through the faces of the control volumes

$$\vec{F}_{face,i} = \frac{1}{2}[\vec{F}_{c,i}(\vec{U}_R) + \vec{F}_{c,i}(\vec{U}_L) - |A_{roe}|(\vec{U}_R - \vec{U}_L)], \tag{2.20}$$

where A_{roe} is the Roe dissipation matrix. The left (U_L) and right (U_R) states are reconstructed using

$$\begin{aligned} U_L &= U_i + (\nabla U_i) \cdot \vec{r}_{i,c} \\ U_R &= U_j + (\nabla U_j) \cdot \vec{r}_{j,c} \end{aligned}$$

where i, j refer to the indices of the left and right nodes, $\vec{r}_{i,c}$ and $\vec{r}_{j,c}$ are the edge vectors pointing to the face centre, and Green-Gauss theorem is used to calculate the gradients. Harten's entropy correction is used to deal with non-physical solutions which may occur. mgOpt also has the AUSM$_{+}^{up}$ convective flux scheme implemented [34].

For viscous walls, the relative velocity between the wall and fluid in contact with the wall is zero. Therefore the contribution to the convective flux is reduced to the pressure terms of the momentum equation

$$\vec{F}_c = \begin{bmatrix} 0 \\ \vec{n} \cdot p \\ 0 \\ 0 \end{bmatrix} \cdot dS. \tag{2.21}$$

For a strongly specified no-slip boundary condition, the momentum equations do not need to be solved at the wall and no wall fluxes need to be calculated. However, the viscous flux contribution cannot be treated in a similar way for CHT problems. The development of the heat transfer capabilities of the solver is the main contribution of the author. This involves extending the solver to accommodate non-adiabatic walls. On the fluid side of CHT problems, the main equation of concern is the RANS energy equation which is given as

$$\frac{\partial}{\partial t} \int_{\Omega_f} \rho e d\Omega_f + \oint_{d\Omega_f} \rho H (\vec{v} \cdot \vec{n}) dS - \oint_{d\Omega_f} k (\nabla T \cdot \vec{n}) dS - \oint_{d\Omega_f} (\overline{\overline{\tau}} \cdot \vec{v}) \cdot \vec{n} dS = \int_{d\Omega_f} \rho \vec{f}_e d\Omega_f, \tag{2.22}$$

where, ρ, p, and $\vec{v}$ are the fluid density, pressure, and velocity respectively, e the internal energy per unit mass, H the total enthalpy, $\overline{\overline{\tau}}$ the viscous stress tensor, $\vec{f}_e$ is the body force which is set as zero. At the fluid-solid interface (dS_f), the viscous flux component of Equation (2.22) reduces to

$$\oint_{dS_f} k (\nabla T \cdot \vec{n}) dS \tag{2.23}$$

where k is the fluid thermal conductivity which is the sum of the laminar (k_L) and turbulent (k_T) conductivities

$$\begin{aligned} k &= k_L + k_T, \\ k_L &= c_p \frac{\mu_L}{Pr_L}, \\ k_T &= c_p \frac{\mu_T}{Pr_T}, \end{aligned} \tag{2.24}$$

and $Pr_L = 0.713$ and $Pr_T = 0.85$. For adiabatic walls, the temperature gradient normal to the wall is zero therefore viscous fluxes do not need to be calculated. For CHT problems however, the fluid-solid interface would be non-adiabatic meaning the wall normal temperature gradient would be non-zero. The heat flux through the fluid-solid interface is then specified using either Neumann or Robin boundary conditions and the implementation of these boundary conditions are described next.

2.3.1 Neumann boundary implementation

Neumann boundaries refer to boundaries at which the solid interface heat flux (q_{sw}) is imposed as a boundary condition in the fluid domain. The heat conduction solver outputs the heat flux in cartesian coordinates and the wall normal component is added to the energy residual of Equation (2.19)

$$\int_{dS_f} k(\nabla T \cdot \vec{n}) = q_{sw}, \tag{2.25}$$

$$q_{sw} = q_x \cdot n_x + q_y \cdot n_y + q_z \cdot n_z, \tag{2.26}$$

$$R(e) = R(e) + q_{sw} \cdot ds. \tag{2.27}$$

2.3.2 Robin boundary implementation

This boundary condition requires a solid ambient temperature ($\tilde{T}_s$), and the solid conductivity ($\tilde{R}$) as inputs. The contribution to the energy residual (Equation (2.19)) is then calculated as

$$\int_{dS_f} k(\nabla T \cdot \vec{n}) = \tilde{R}(T_{fw} - \tilde{T}_s), \tag{2.28}$$

$$R(e) = R(e) + \tilde{R}(T_{fw} - \tilde{T}_s) \cdot ds, \tag{2.29}$$

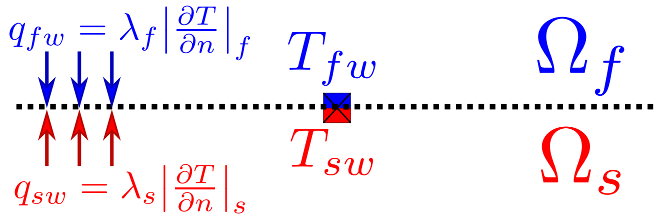

$q_{fw} = \lambda_f \left| \frac{\partial T}{\partial n} \right|_f$
T_{fw}
Ω_f
T_{sw}
Ω_s
$q_{sw} = \lambda_s \left| \frac{\partial T}{\partial n} \right|_s$

Figure 2.5: Definition of boundary condition terms at the Fluid-Solid interface.

where T_{fw} is the fluid wall temperature at the current flow solver iteration.

2.4 Coupling algorithms

CHT problems may be solved using a monolithic or partitioned approach. In the monolithic approach, both fluid and solid equations are solved simultaneously by one numerical solver and the continuity of temperature and heat flux is imposed implicitly. The main advantage of this approach is that no iterations have to be carried out between the solution of the governing equations of the fluid and solid domains. However, monolithic codes are only limited to the particular combination of problems for which they were developed and require a high level of expertise to develop [35] [36].

In the segregated or partitioned approach, separate solvers are used for the fluid and structure which are loosely coupled through interface boundary conditions. These boundary conditions need to be updated iteratively until the temperature and heat flux are continuous between the two domains (see Figure 2.5). That is until

$$\begin{aligned} T_{sw} &= T_{fw}, \\ \underbrace{\lambda_s \left| \frac{\partial T}{\partial n} \right|_s}_{q_{sw}} &= \underbrace{\lambda_f \left| \frac{\partial T}{\partial n} \right|_f}_{q_{fw}}, \end{aligned} \tag{2.30}$$

where T_{fw} is the interface temperature in the fluid domain Ω_f, T_{sw} the interface temperature in the solid domain Ω_s, λ_f and λ_s the thermal conductivity of the fluid and solid respectively, n the surface normal, and q_{fw} and q_{sw} the interface heat flux in the fluid and solid domains respectively.

One advantage of the partitioned approach is the flexibility of using different existing solvers for both domains [37] [38]. The partitioned approach also allows the use of non-matching grids at the fluid-solid interface. However, this leads to a need for interpolation of interface values between domains [36] [37] [39]. In this thesis, only matching grids are used at the interface, in order to avoid the need for interpolation. Moreover, it is computationally inexpensive to solve the solid domain and fine meshes can be used without issue.

Different coupling algorithms exist depending on which boundary conditions are exchanged between both domains. Boundary conditions in both domains could either be Dirichlet (Equation (2.31)), Neumann (Equation (2.32)), or Robin (Equation (2.33)).

$$T = g_1, \tag{2.31}$$
$$\lambda\frac{\partial T}{\partial n} = g_2, \tag{2.32}$$
$$\lambda\frac{\partial T}{\partial n} = h \cdot (T - T_{sink}), \tag{2.33}$$

where n is the surface normal and T_{sink} is the ambient temperature in the domain. A Dirichlet boundary conditions refers to an imposed temperature, while a Neumann condition is an imposed heat flux, and a Robin boundary condition is a mixture of the Dirichlet and Neumann. These three boundary conditions can be combined to form a variety of coupling algorithms as shown in Table 2.1.

Fluid-Solid	Name	Nomenclature
Dirichlet-Neumann	Flux Forward Temperature Back	FFTB
Dirichlet-Robin	heat transfer coefficient Forward Temperature Back	hFTB
Neumann-Dirichlet	Temperature Forward Flux Back	TFFB
Neumann-Robin	heat transfer coefficient Forward Flux Back	hFFB
Robin-Dirichlet	Temperature Forward solid coefficient Back	TFRB
Robin-Neumann	Flux Forward solid coefficient Back	FFRB
Robin-Robin	heat transfer coefficient Forward solid coefficient Back	hFRB

Table 2.1: Types of coupling algorithms. Forward refers to the boundary condition sent by the fluid domain while Back refers to the boundary condition it receives.

The easiest algorithm for partitioned coupling approaches is the fixed-point/Gauss-Seidel iteration in which solvers exchange boundary conditions iteratively until convergence. That is, until there is no change in the value of exchanged boundary conditions between subsequent iterations and the heat flux and temperature is continuous between domains. This is also known as the primal loop/primal coupling iteration. However, fixed-point methods converge slowly and may even fail to converge sometimes.

The stability and convergence behaviour of each fixed-point algorithm is different and several stability analysis studies for partitioned methods have been done. Giles [40] performs a stability analysis on a 1D heat transfer problem using Dirichlet-Neumann coupling concluding that the FFTB is required for stability. However, the conclusions do not apply to steady state problems. Verstraete et al. provide a stability criteria based on Biot number for four coupling algorithms [6]. The work of Verstraete et al. is extended by Scholl et al. who add the Biot number stability criteria for the final three coupling algorithms [7].

The Biot number gives the ratio between the conductive and convective thermal resistance and is defined as

$$Bi = \frac{hL}{\lambda_s} = \frac{\text{Conductive resistance}}{\text{Convective resistance}}, \tag{2.34}$$

where L is the characteristic length of the solid. The Biot number provides insight into the relationship of the heat transfer between both domains. A low Biot number implies a larger temperature gradient in the fluid while a high Biot number implies a larger temperature gradient in the solid.

Other stability studies have been performed in [41] [42] [39]. Based on the work of Verstraete and Scholl, the current work uses the Biot number to ensure the stability of the partitioned method where possible.

A common approach to speed up the convergence time of fixed-point methods is to only partially converge the flow equations. However, this requires more coupling iterations between solvers. Verstraete and Van den Braembussche [43] show that when this is done, imposing Dirichlet conditions in the fluid domain should be avoided and that the stability of coupling algorithms which impose Neumann conditions in the fluid domain will be improved. This is because imposing a Dirichlet (temperature) boundary condition in the fluid domain leads to a sudden change in the wall heat flux between coupling iterations. This leads to an over-prediction of the heat flux value as the thermal boundary layer is not fully converged. This over-prediction of the heat flux could then lead to instability of the coupling algorithm. Alternatively, if a Neumann boundary condition is imposed in the fluid domain, the wall temperature will be under-predicted resulting in an increase in the stability of the coupling algorithm [38]. A similar advantage of partial convergence of the flow equation using Robin boundary conditions was recently demonstrated by Scholl [7].

Alternative to Gauss-Seidel iterations, other coupling algorithms used to accelerate the convergence of partitioned methods include Aitken relaxation, Newton-Krylov methods, and Anderson mixing methods [37]. Despite the stability and convergence limitations of fixed-point iterations, their ease of implementation remains an advantage. Furthermore, time saving benefits from partial fluid convergence ensure that fixed-point coupling algorithms remain a competitive option with regard to partitioned methods for CHT problems.

Consequently, this thesis focuses on the use of fixed-point coupling algorithms which use Neumann or Robin boundary conditions in the fluid domain. Computational time savings are obtained by partially converging the flow equations while the solid is solved to full convergence at each coupling iteration. The three coupling algorithms used for the primal solve in the present work are described below.

2.4.1 TFFB

In the Temperature Forward Flux Back (TFFB) method [44], the solid interface heat flux distribution, q_{sw}^i (where i is the current coupling iteration), is imposed as a boundary condition to the fluid domain. The fluid solver $\mathbf{F}$ solves the flow equations resulting in a fluid interface temperature distribution, T_{fw}^i. This temperature is then imposed as boundary condition for the solid domain and the solid conduction solver, $\mathbf{S}$, provides an updated heat flux distribution q_{sw}^{i+1}. This loop is continued until there is no change in the boundary conditions exchanged by both solvers (see Figure 2.6).

$$
\begin{aligned}
T_{fw}^i &= \mathbf{F}(q_{sw}^i), && (2.35)\\
q_{sw}^{i+1} &= \mathbf{S}(T_{fw}^i). && (2.36)
\end{aligned}
$$

2.4.2 TFRB

The Temperature Forward solid coefficient Back (TFRB) coupling algorithm imposes a Dirichlet boundary condition in the solid domain and uses a Robin boundary condition in the fluid domain. The method also requires a virtual conductivity $\tilde{R}$ to be specified. Where $\tilde{R} = \frac{\tilde{\lambda}}{L}$, $\tilde{\lambda}$ is the virtual solid conductivity, and L is a solid length scale. The values of $\tilde{\lambda}$ and L are non-physical quantities chosen by the user which affect the speed of convergence and stability of the method [7].

$$\begin{aligned} q_{sw}^i &= \mathbf{S}(T_{sw}^i), && (2.37)\\ \tilde{T}_s^i &= \frac{q_{sw}^i}{\tilde{R}} + T_{sw}^i, \implies \text{(calc. } \tilde{T}_s\text{)} && (2.38)\\ T_{fw}^i &= \mathbf{F}(\tilde{T}_s^i, \tilde{R}), && (2.39)\\ q_{fw}^i &= \tilde{R}(\tilde{T}_s^i - T_{fw}^i), \implies \text{(Robin BC in fluid)} && (2.40)\\ T_{sw}^{i+1} &= T_{fw}^i. && (2.41) \end{aligned}$$

Assuming the algorithm begins in the solid domain, an initial guess for the wall temperature T_{sw} is imposed on the solid and used to obtain the heat flux q_{sw}. Next, the virtual solid sink temperature $\tilde{T}_s$ is calculated using Equation (2.38). The virtual conductivity and solid sink temperature are used for a robin boundary condition in the fluid domain. The flow solver then retunes an update of the interface temperature which is given to solid as a Dirichlet boundary condition (see Figure 2.6).

2.4.3 hFRB

The heat transfer coefficient Forward solid coefficient Back (hFRB) method uses Robin boundary conditions in both domains and is the same as TFRB on the fluid side (see Figure 2.6). Similar to the TFRB method, the algorithm can start with an initial guess of the interface temperature on the solid side. Next, the virtual solid temperature $\tilde{T}_s$ can be calculated and passed to the flow solver along with the virtual conductivity $\tilde{R}$.

$$\begin{aligned} q_{sw}^i &= \mathbf{S}(T_{sw}^i), && (2.42)\\ \tilde{T}_s^i &= \frac{q_{sw}^i}{\tilde{R}} + T_{sw}^i, \implies \text{(calc. } \tilde{T}_s\text{)} && (2.43)\\ T_{fw}^i, q_{fw}^i &= \mathbf{F}(\tilde{T}_s^i, \tilde{R}), && (2.44)\\ T_{sink}^i &= T_{fw}^i - \frac{q_{fw}^i}{\tilde{h}}, \implies \text{(calc. } T_{sink}\text{)} && (2.45)\\ q_{sw}^{i+1}, T_{sw}^{i+1} &= \mathbf{S}(T_{sink}^i, \tilde{h}). && (2.46) \end{aligned}$$

The outputs from the flow solver are the interface temperature and heat flux (T_{fw}, q_{fw}). These are used in Equation (2.45) to calculate the ambient fluid temperature (T_{sink}^i) for a given value of the virtual heat transfer coefficient $(\tilde{h})$. The solid solver then returns an

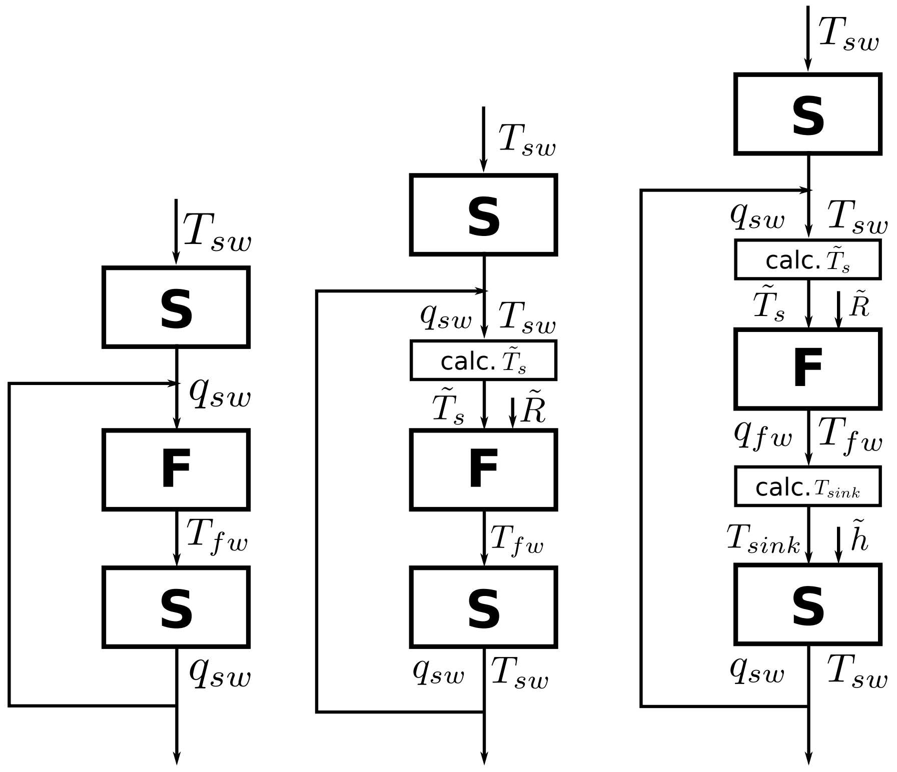


Figure 2.6: CHT coupling algorithms. Left:TFFB, Centre: TFRB, Right:hFRB.

update on the interface temperature and heat flux and the exchange is continued until convergence [7].

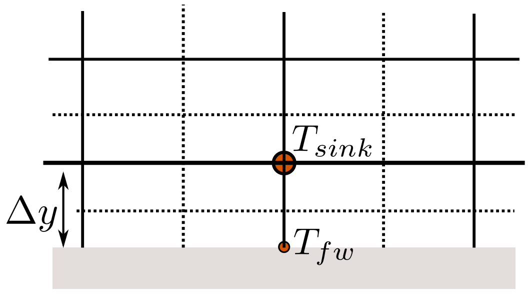


Figure 2.7: Interface (T_{fw}) and ambient/sink (T_{sink}) temperatures on the fluid side. Solid lines represent the mesh while dashed lines represent the fluid control volumes.

$$T^i_{sink} = T^i_{fw} + \frac{q^i_{fw}}{\tilde{h}}. \tag{2.47}$$

Although any user specified value of $\tilde{h}$ can be used, it is difficult to estimate appropriate values of $\tilde{h}$. Therefore, in the present work, the first off-wall node (see Figure 2.7) is used as the ambient fluid temperature (T_{sink}). $\tilde{h}$ is then calculated using the fluid thermal conductivity λ_f as

$$\tilde{h} = \frac{\lambda_f}{\Delta y}. \tag{2.48}$$

This makes $\tilde{h}$ closer to the actual value of the heat transfer coefficient. As CalculiX requires facial values for the Robin boundary conditions, the facial values are obtained by averaging the values of T_{sink} and $\tilde{h}$ at the nodes forming the face.

2.5 Summary

This chapter describes the fluid and solid governing equations for CHT and methods for solving CHT problems. CHT problems can be solved using either a monolithic or partitioned approach, however, the partitioned approach is preferred as it allows the use of separate solvers for each domain. The partitioned approach results in the need for coupling iterations between solvers to ensure the continuity of heat flux and temperature between the fluid and solid domain. In this work, fixed-point coupling iterations will be used to iteratively exchange boundary conditions until this continuity is achieved.

The convergence speed of fixed-point methods can be improved by only partially converging the flow equations. Previous studies have shown that imposing Dirichlet boundary conditions in the fluid domain undermines the stability of coupling algorithms when the flow equations are partially converged. Alternatively, when the flow equations are partially converged, Neumann or Robin boundary conditions in the fluid domain improve the stability of the coupling algorithm. Consequently, three fixed-point coupling algorithms used in this work avoid the use of a Dirichlet boundary condition in the fluid domain.

Finally, the numerical solvers to be used for MDA were introduced and the implementation of Neumann and Robin boundary conditions in the in-house CFD solver was described. The described coupling algorithms can now be combined with the numerical solvers to solve CHT problems and this is shown in the next chapter.

Chapter 3

Numerical analysis and Solver validation

In this chapter, Multidisciplinary Design Analysis (MDA) is carried out on a variety of CHT problems. As MDA is used to determine the fitness of purpose of a design, it is essential that the solvers and coupling algorithms used are accurate. Moreover, the accuracy of the gradients required for gradient based optimisation is dependent on the accuracy of the coupling algorithms and their solver implementation. Consequently, the coupling algorithms described in Chapter 2 as well as the numerical solvers used are validated using test cases with analytic or experimental results.

3.1 Flat plate

The first problem considered is laminar flow over a flat plate with finite thickness. The free stream flow is at temperature T_∞, while the bottom of the plate is maintained at a cooler temperature T_b. Due to the temperature difference between the solid plate and the hot air, there is heat transfer between the solid plate and the fluid.

The aim is to accurately compute the wall temperature T_w at the interface between the fluid and solid, which is unknown a priori and can only be computed by considering the coupled problem. This test case is chosen as a benchmark to verify the numerical solvers and all coupling algorithms by comparing the results with the Luikov analytic solution [45]. The thermal conductivity of the plate is determined through the average Biot number over

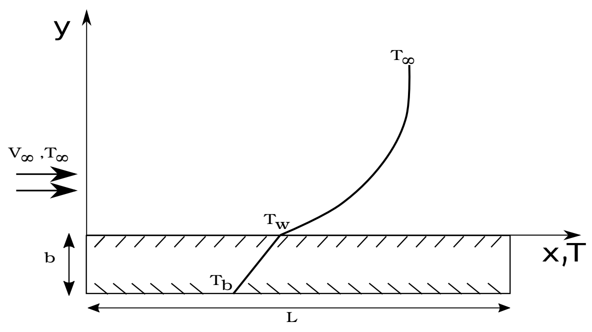


Figure 3.1: Flat plate problem description.

Parameter	Value	units
b	0.01	m
L	0.2	m
M_∞	0.1	
P_∞	$1.03 \cdot 10^5$	Pa
T_∞	1000	K
T_b	600	K
λ_s	0.2222	W/mK
λ_f	0.05568	W/mK
μ	$3.95 \cdot 10^{-5}$	Pa · s
Re_L	$1.132 \cdot 10^5$	
Pr	0.713	

Table 3.1: Flat plate test case table of parameters.

the plate which is calculated as

$$\overline{Bi} = \frac{\overline{h} \cdot b}{\lambda_s}. \tag{3.1}$$

Where $\overline{h}$ is the average heat transfer coefficient over the length of the plate calculated as

$$\overline{h} = \frac{1}{L} 0.664 \lambda_f \sqrt[3]{Pr} \sqrt{Re_L}. \tag{3.2}$$

Using the values in Table 3.1, the average heat transfer coefficient, $\overline{h}$, over the plate is 55.55 W/m^{2}K. The thermal conductivity of the solid can then be calculated for a user defined value of the average Biot number as

$$\lambda_s = \frac{0.5556}{\overline{Bi}}. \tag{3.3}$$

An average Biot number of 2.5 is chosen leading to a solid thermal conductivity of 0.2222 W/mK. However, the actual value of the Biot number varies along the length of the plate. Matching grids are used in both the fluid and solid domains and the flow boundary conditions are shown in Figure 3.2. The fluid domain is discretised using a grid of 137x97 nodes with a total of 226 interface nodes while the solid domain is discretised with 1808 nodes. For the TFRB and hFRB coupling algorithms, $\tilde{R}$ is taken as $\frac{\lambda_s}{b}$.

The Luikov differential heat transfer equation is given as

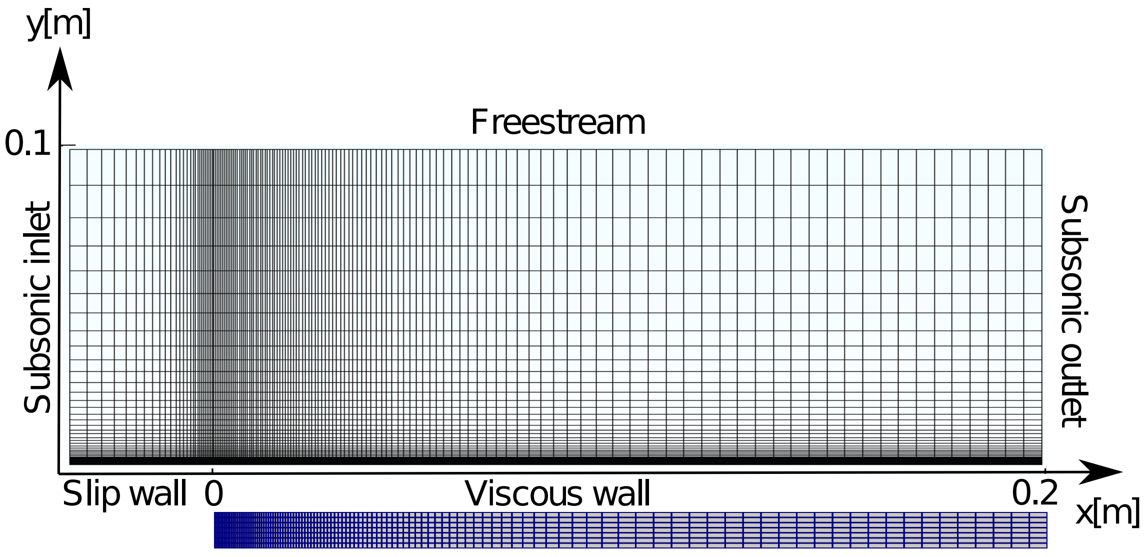


Figure 3.2: Flat plate fluid and solid meshes and computational setup.

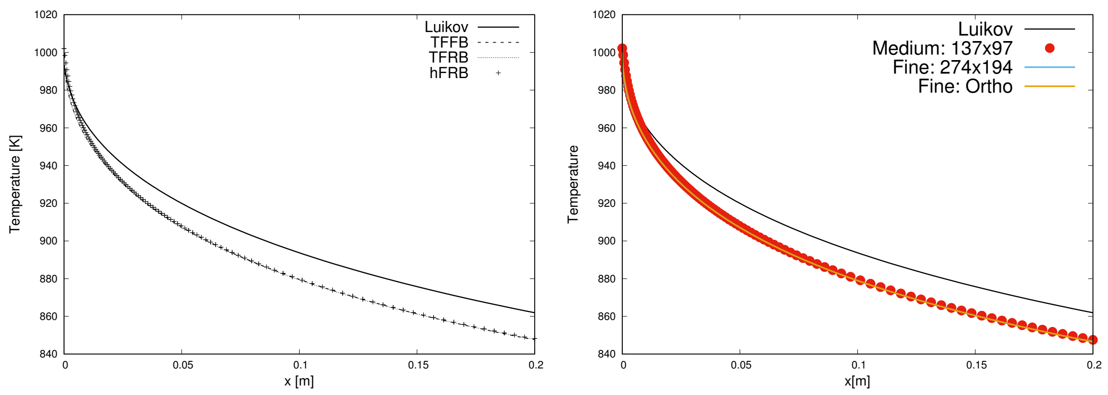
$$\begin{aligned}\frac{T_{f(x,y)}-T_b}{T_\infty-T_b} \quad = \quad & \frac{K-\frac{1}{2}B}{K-B}\exp\Big[(K^2-BK)+\frac{\lambda_s}{\lambda_s}\frac{y}{b}\Big]\cdot\operatorname{erfc}\Big(K-\frac{1}{2}B+\frac{1}{2}\frac{\lambda_s}{\lambda_f}\frac{y}{Kb}\Big) \\ & +\frac{1}{2}\operatorname{erfc}\Big(\frac{1}{2}B-\frac{1}{2}\frac{\lambda_s}{\lambda_f}\frac{y}{Kb}\Big) \\ & -\frac{1}{2}\frac{\exp\Big[\frac{B}{K}\frac{\lambda_s}{\lambda_s}\frac{y}{b}\Big]}{1-\frac{B}{K}}\cdot\operatorname{erfc}\Big(\frac{1}{2}B+\frac{1}{2}\frac{\lambda_s}{\lambda_f}\frac{y}{Kb}\Big). \qquad (3.4)\end{aligned}$$

where erfc is the error function, $K=\frac{\lambda_s}{\lambda_f}\frac{x}{b}Pr^{0.5}\overline{Re}_x^{-0.5}$, $B=\frac{\bar{v}_y}{\bar{v}_x}\sqrt{Pr\overline{Re}_x}$, $\bar{v}_y=0.43v_\infty(\frac{xv_\infty}{\nu})^{-0.5}$, $\bar{v}_x=0.66v_\infty$, and $\overline{Re}_x=\frac{\bar{v}_x x}{\nu}$.

3.1.1 Results

The obtained temperature distribution from all three coupling algorithms is shown in Figure 3.3a. All three algorithms obtain the same solution however the CHT solutions slightly under-predict the interface temperature compared to the analytic profile.

To further investigate the discrepancy between the analytic and CHT temperature distribution, a mesh convergence study was performed. Figure 3.3b shows the results for a



(a) Results for all coupling algorithms. (b) hFRB: Mesh convergence study.

Figure 3.3: Flat plate interface temperature distribution.

medium mesh with 137x97 points and the fine mesh with 274x194 point. Both meshes obtain the same results showing that the discrepancy is not a result of the discretisation.

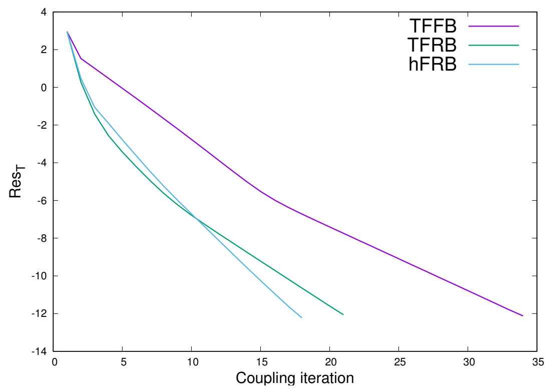
One assumption for this discrepancy is that the Luikov solution is only an approximate solution where lateral heat transfer in the solid is not modelled. Consequently, the solid was modelled as an orthotropic material using a longitudinal conductivity of $\lambda_f|_y = 0.2222$ W/mK and a lateral conductivity of $\lambda_f|_x = 10^{-6}$ W/mK. The results in Figure 3.3b for Fine:Ortho show that no significant difference between modelling the solid as an isotropic or orthotropic material. Although the cause of the discrepancy could not be determined, similar under-prediction of the interface temperature has been reported in [7].

3.2 Convergence of coupling algorithms

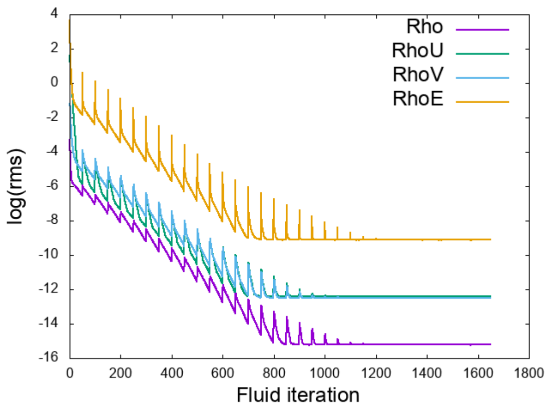
The convergence behaviour of all three algorithms on the flat plate test case was evaluated. A coupling residual which quantifies the change in interface temperature between coupling iterations is defined as

$$Res_T = log_{10}\left(\sqrt{\frac{1}{N}\sum_{j=1}^{N}(T_j^i - T_j^{i-1})^2}\right), \tag{3.5}$$

where i denotes the coupling iteration and N the number of interface nodes. All runs were ended as soon as Res_T fell below -12. Figure 3.4a shows that the methods which use Robin boundary require less coupling iterations to converge with Robin-Robin method requiring the least.



(a) Convergence of coupling algorithms.



(b) TFFB: Partial convergence of fluid equations

Figure 3.4: Comparison of coupling convergence on the Flat plate test case.

Furthermore, the computational runtime of the fixed-point methods was decreased by only partially converging the flow equations between coupling iterations (see Figure 3.4b). For runs with partial fluid convergence, the flow solver was interrupted every 50 fluid iterations to get the fluid interface conditions. The solid solver was then run to full convergence (using the unconverged fluid interface conditions) to produce an update of the solid interface conditions. The fluid solver then used the updated solid interface conditions and continued for the next fifty fluid iterations. This is what causes the spikes in the fluid residual every 50 iterations seen in Figure 3.4b. On the other hand, during runs with full fluid convergence, the flow solver was run to full convergence (typically 350 fluid iterations), before the exchange of boundary conditions.

Coupling method	Full fluid convergence		Partial fluid convergence	
	No its.	Time [min]	No its.	Time [min]
TFFB	37	435.4	34	69.1
TFRB	23	211.1	21	49.9
hFRB	18	144.6	18	30.8

Table 3.2: Comparison of runtimes and number of coupling iterations for all coupling algorithms.

The results presented in Table 3.2 show that the Robin-based coupling algorithms required less wall-clock time to converge with the hFRB algorithm being the fastest. For all three algorithms, the partial convergence of the flow equations leads to a significant reduction in computational time without a compromise in accuracy. Full fluid convergence lead to an increase in the number of coupling iterations required for the TFFB and TFRB algorithms. This might be explained by the results presented by Verstraete and Scholl [38] [7] which show that partial convergence of the fluid solver increases the stability of coupling algorithms.

For the hFRB method, using 30 fluid iterations between boundary condition updates reduced the computational time to 17.2 minutes for a total number of 18 coupling iterations. However, reducing to 10 fluid iterations between updates increased the time to 28.8 minutes for a total of 35 coupling iterations. This suggests there is an optimum number fluid iterations which produces the least computational time and similar results have been reported by Scholl [7]. As a result of the findings in Table 3.2 and Figure 3.4a, the hFRB algorithm is favoured in this thesis and computational time savings are obtained by only partially converging the flow equations between coupling iterations.

Although it was shown by Ganine et al. [37] that significant computational time saving can be obtained through the use of several acceleration techniques, the results were not benchmarked against standard fixed-point methods and the use of Robin boundary conditions in the fluid domain was not considered. Consequently, the advantage, in terms of compute time, of more sophisticated coupling algorithms over fixed-point methods remains unclear.

3.3 C3X & MarkII turbine blades

The C3X turbine blade is chosen as it has been investigated experimentally by Hylton et al. [46]. Similar to modern turbine blades, the blade is convectively cooled by ten cooling channels and a 2D simulation of the problem is carried out. The test case is used to validate the effect of turbulence on the accuracy of the CHT solution. The Spalart-Allmaras one-equation turbulence model is implemented in the flow solver. The MarkII test from the same experimental study as the C3X is also chosen due to the presence of weak and strong shockwaves on the suction side of the blade. A full description of the blade geometries is available in [46].

Both blades are made of ASTM 310 stainless steel and the thermal conductivity is a function of temperature taken as

$$\lambda_s = 6.811 + 0.020176 \cdot T, \tag{3.6}$$

where T is the temperature at a point in the solid. The density and heat capacity are taken as 7900 kgm^{-3} and 586.5 J kg^{-1} K^{-1} respectively.

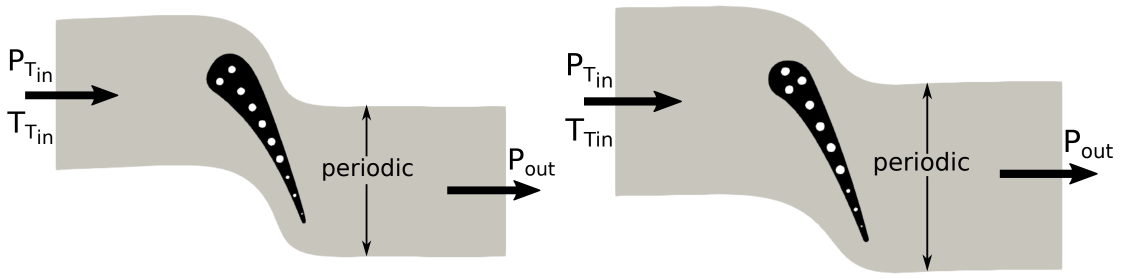


Figure 3.5: Turbine blades geometry and domain (Left: C3X, Right: MarkII).

3.3.1 Boundary conditions

On the fluid side, the total temperature T_T, total pressure P_T, and mach number M are specified at the inlet, while the static pressure is specified at the outlet. The isentropic equations for total pressure (P_T) and total temperature (T_T) ratios are given by

$$\frac{P_T}{P_s} = \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma-1}}, \tag{3.7}$$

$$\frac{T_T}{T_s} = 1 + \left(\frac{\gamma - 1}{2} M^2\right), \tag{3.8}$$

where P_s and T_s are static pressure and static temperature respectively and γ is the heat capacity ratio for air which is taken as 1.4. The fluid boundary conditions for each run are shown in Table 3.3 and the computational domain is shown in Figure 3.5.

Robin boundary conditions are specified on each cooling channel by supplying the heat transfer coefficient, h, and average temperature of the cooling fluid T_c. The heat transfer

Test case	C3X 4521	MarkII 5411
$P_{T_{in}}$[Pa]	413286	337100
$T_{T_{in}}$ [K]	818	788
P_{out} [Pa]	254172	167000
M_{in}	0.17	0.19
M_{out}	0.89	1.04

Table 3.3: Turbine blades fluid boundary conditions

coefficient is calculated from the Nusselt number given by

$$Nu = 0.022 Cr Re_D^{0.8} Pr^{0.5}, \tag{3.9}$$

$$h = \frac{\lambda_f \cdot Nu}{d}, \tag{3.10}$$

$$Re = \frac{\rho v d}{\mu}. \tag{3.11}$$

Where Re is the Reynolds number shown in Equation (3.11), d is the channel diameter, μ and ρ are the viscosity and density, and v is the velocity of the fluid. Cr is the corrective coffecient which is given for each cooling channel while Pr is the Prandtl number taken as 0.7. Combining Equations (3.9) and (3.10), the heat transfer coefficient can then be calculated as

$$h = \frac{\lambda_f}{d} \cdot 0.022 \cdot Cr \cdot Pr^{0.5} \cdot \left(\frac{\dot{m} \cdot \frac{4}{\pi d}}{\mu} \right)^{0.8} \tag{3.12}$$

Where $\dot{m}$ is the mass flow rate through the channel and is given in the experimental data. The viscosity and thermal conductivity for each channel are calculated using the Sutherland formula

$$\mu = \mu_0 \left(\frac{T_c}{T_0} \right)^{\frac{3}{2}} \frac{T_0 + S}{T_c + S}, \tag{3.13}$$

$$\lambda_f = \lambda_0 \left(\frac{T_c}{T_0} \right)^{\frac{3}{2}} \frac{T_0 + S}{T_c + S}. \tag{3.14}$$

Where $\mu_0 = 1.7894 \cdot 10^{-5}$ Pa/s, $T_0 = 273.11$ K, $S = 110.56$, and $\lambda_0 = 0.0261$ W/(mK). The cooling channel boundary conditions for both runs are shown in Table 3.4.

The hFRB coupling algorithm is used and to the best of the authors' knowledge, this

			C3X: 4521			MarkII: 5411		
ID.	d [m]	Cr	$\dot{m}$ [Kg/s]	T_c [K]	h [W/m^2/K]	$\dot{m}$ [Kg/s]	T_c [K]	h [W/m^2/K]
1	0.0063	1.118	0.0222	352.14	1848.925763	0.0246	336.39	1993.570538
2	0.0063	1.118	0.0221	354.54	1844.108669	0.0237	326.27	1926.157139
3	0.0063	1.118	0.0218	345.62	1817.188554	0.0238	332.68	1938.314169
4	0.0063	1.118	0.0228	346.72	1884.462461	0.0247	338.86	2002.237365
5	0.0063	1.118	0.0225	340.7	1859.750452	0.0233	318.95	1893.604378
6	0.0063	1.118	0.0225	366.21	1879.68760	0.0228	315.58	1858.029672
7	0.0063	1.118	0.0216	351.48	1808.338204	0.0238	326.26	1932.647274
8	0.0031	1.056	0.00744	376.24	2635.689007	0.00775	359.83	2705.486664
9	0.0031	1.056	0.00477	406.97	1867.93929	0.00511	360.89	1939.672853
10	0.00198	1.025	0.00256	446.69	2502.441424	0.00334	414.85	3063.69078

Table 3.4: Turbine blades cooling channel conditions (d, Cr, $\dot{m}$, and T_c obtained from experiment).

is the first time Robin-Robin coupling boundary conditions have been used on the MarkII and C3X cases. The solid length scale is taken as 2% of the chord length and the virtual conductivity was calculated as $\tilde{R} = \dfrac{\lambda_s}{0.02 \cdot x_c}$. As the solid conductivity is a function of temperature described in Equation (3.6), the value of $\tilde{R}$ changes with each coupling iteration.

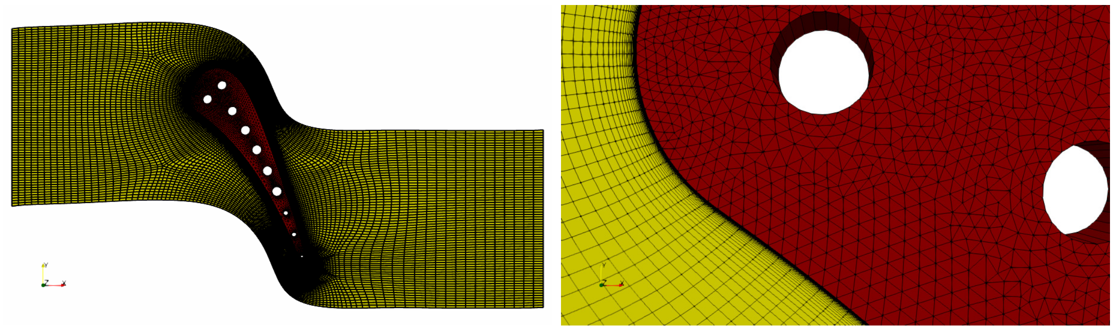


Figure 3.6: C3X mesh with matching grids in both domains.

The C3X domain is discretised using 36,744 nodes in the fluid and 5,412 nodes in the solid. The MarkII fluid domain is discretised using 49,532 nodes while the solid has 5,714 nodes. The solid domains both use wedge (C3D6) elements. A near wall spacing y^+ of less than 1 is used and 2 levels of multigrid meshes are used in the fluid domain to accelerate convergence. Matching grids are used for both domains as shown in Figures 3.6 and 3.7.

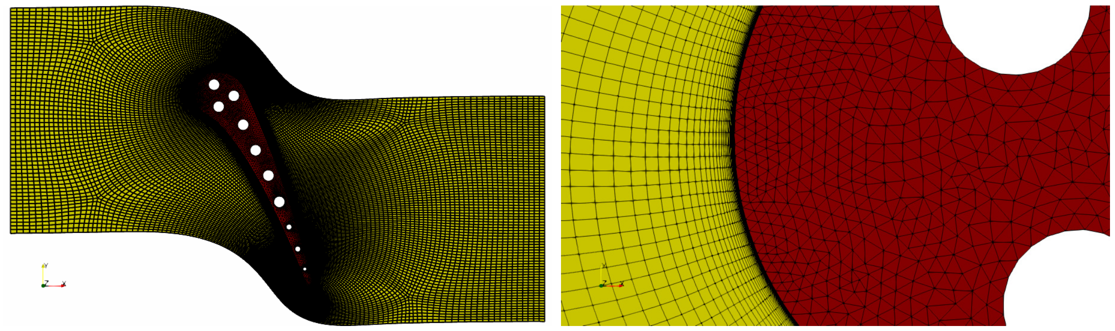


Figure 3.7: MarkII mesh with matching grids in both domains.

3.3.2 Results

The simulation is terminated once Res_T drops below -6 and the obtained Mach number and temperature distribution are shown in Figures 3.8 and 3.9 respectively. The MarkII results show the presence of two shock waves on the suction side of the blade. The C3X case which has a lower exit Mach number only shows a mild shock at the trailing edge of the suction side.

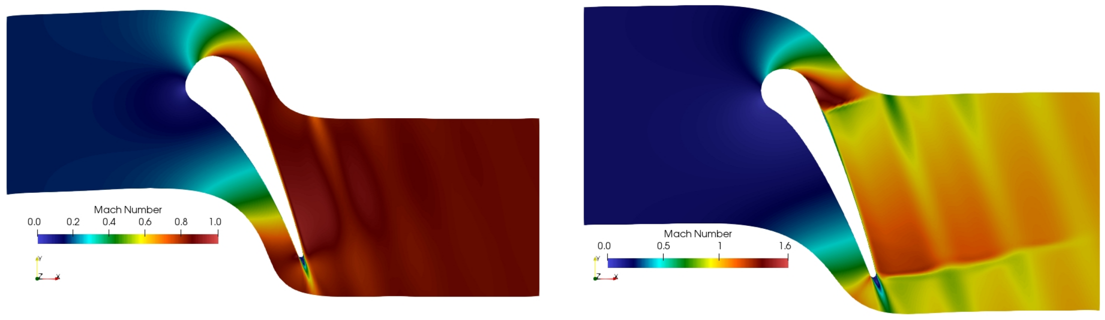


Figure 3.8: Mach number distribution (Left: C3X 4521, Right: MarkII 5411).

Figures 3.10 - 3.12 show plots of the static pressure (P_s), static temperature (T), and heat transfer coefficient (H) distributions. The values are normalised as was done in the experiment in order to compare the numerical and experimental results. Good agreement is seen between the CHT simulations and the experimental data for the surface pressure distribution while the surface temperature and heat transfer coefficient distributions show some discrepancy on the suction side.

The presence of shockwaves can be seen in the sudden increase and decrease of values

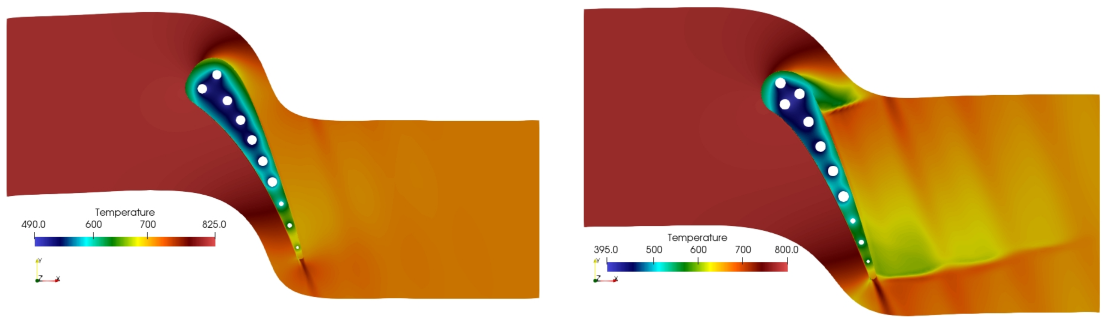


Figure 3.9: Static temperature distribution (Left: C3X 4521, Right: MarkII 5411).

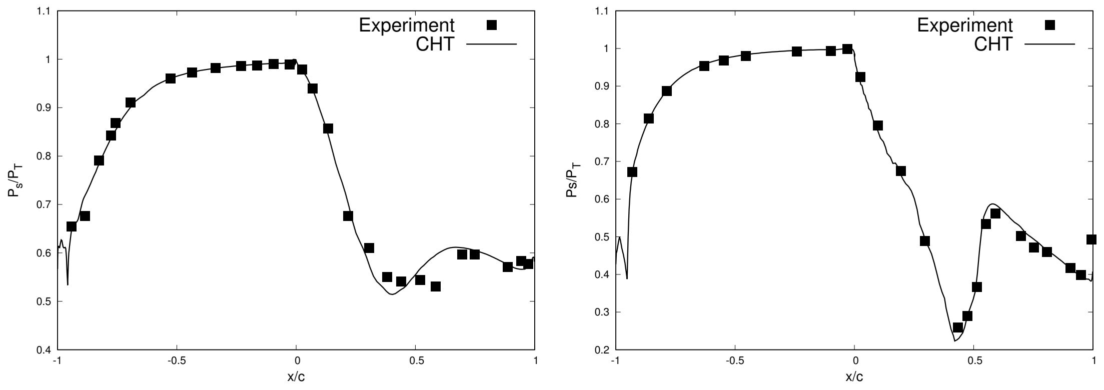


Figure 3.10: Surface pressure distribution (Left: C3X 4521, Right: MarkII 5411).

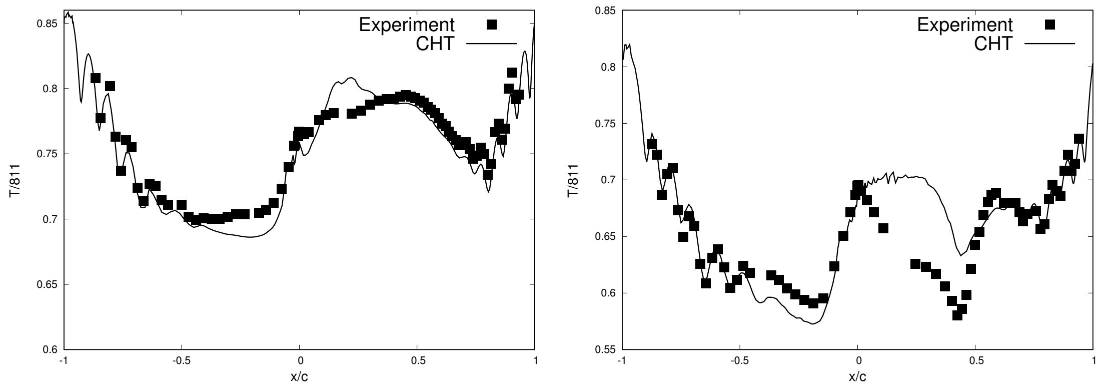
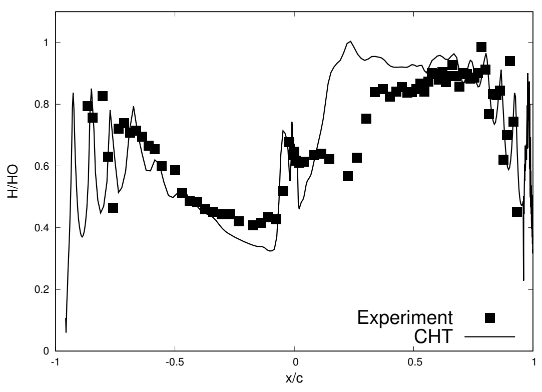


Figure 3.11: Surface temperature distribution (Left: C3X 4521, Right: MarkII 5411).

on the suction side ($x/c > 0$) of the blades. The effect of cooling channels can also be seen in the oscillations of the temperature distribution at the trailing edge of the blades.



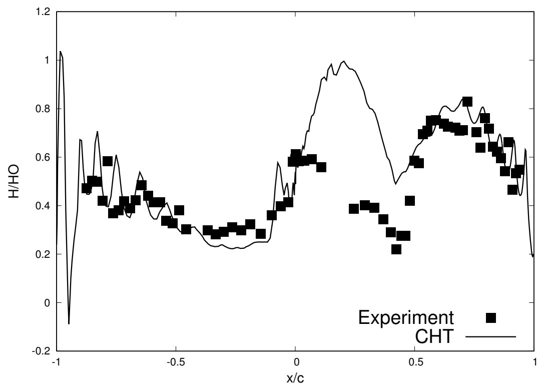


Figure 3.12: Surface heat transfer coefficient (Left: C3X 4521, Right: MarkII 5411).

The obtained CHT results are similar to those obtained by Tetsuya et al. [47] using the Spalart-Allmaras (SA) turbulence model. The literature shows that transition turbulence models are required to accurately capture the flow behaviour on the suction side of the blade [47] [48] [49] [50] [51]. This is because the SA model assumes fully turbulent flow and is unable to predict the point of transition from laminar to turbulent flow.

3.4 Summary

In this chapter the validity and accuracy of the numerical solvers and coupling algorithms used for MDA was evaluated. The obtained results show good agreement between analytic and experimental data and the CHT simulations for all three test cases. The results of the turbine blades match the experiments to the level expected of the Spallart-Allmaras turbulence model as reported in literature. These results could be improved through the use of transition models, however, this is beyond the scope of this work.

The rate of convergence of the three coupling algorithms used was compared on the flat plate test case and it was seen that the Robin-Robin (hFRB) method converges fastest and computational cost savings can be obtained through partial convergence of the flow equations between coupling iterations.

The results of the multidisciplinary CHT analysis give confidence that the numerical solvers and coupling algorithms can effectively be used to asses the fitness of purpose of a design. The solvers and coupling algorithms can now be used for Multidisciplinary Design Optimisation.

Chapter 4

Sensitivity analysis

The main aim of Multidisciplinary Design Analysis (MDA) is to provide an assessment of the performance of a design. This performance is typically measured by a singular value, referred to as the objective function (e.q. the lift produced by an aerofoil), and provides a quantitative way to compare several designs and decide which design performs better. Rather than merely measuring the performance of several designs, it is more desirable to find an optimum design which results in the best performance. This can be done through the use of numerical optimisation.

Optimisation algorithms can broadly be grouped into gradient free or gradient based methods. Gradient free methods rely solely on the value of the objective function to drive the optimisation process. The main disadvantage of gradient free methods is that they require a large number of function evaluations to find the optimum. This quickly becomes prohibitive in cases with a large number of design variables and in cases where the cost of obtaining the objective function is very high (e.g. in multidisciplinary problems).

On the other hand, gradient based methods are ideal in cases where the cost of evaluating the objective function is very high because they require fewer function evaluations. These methods make use of the value and gradient of the objective function w.r.t to design variables (e.g the surface of the aerofoil) to drive the optimisation process.

Sensitivity analysis refers to the process of obtaining gradients by assessing how the inputs to a model affect its outputs. Some methods of obtaining gradients include: finite differences, complex step methods, and tangent linearisation [3]. These three methods obtain gradients at a cost linearly dependent on the number of degrees of freedom, which is often prohibitive for industrial problems. Contrarily, adjoint methods obtaining obtain gradients at a cost which does not scale with the number of design variables and this principle is demonstrated in this chapter. Adjoint methods have been developed and used extensively for shape optimisation of single disciplinary problems [11] [30] [52].

This chapter discusses how analytically accurate derivatives are calculated and the adjoint equations are derived. The discrete adjoint method is used in this thesis as it lends itself to the use of Automatic Differentiation [53]. The main advantage of Automatic Differentiation (AD) is that it eases the effort of obtaining accurate derivatives from numerical solvers, thereby circumventing one of the developmental cost of the continuous adjoint. Finally, AD is applied to the two numerical solvers used for partitioned coupling resulting in two independent adjoint solvers. These solvers are able to efficiently obtain relevant CHT gradients and the accuracy of the gradients are verified.

4.1 Tangent linearisation

Tangent linearisation is one method of obtaining exact derivatives and can be done as follows; Let the parameter to be minimised be represented by an objective function $J(U(\alpha), \alpha)$ where U and α represents the state and design variables respectively. The value of interest for gradient based optimisation is the derivative of the objective function with respect to the design variable which is known as the *sensitivity.* That is

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial \alpha} + \frac{\partial J}{\partial U}\frac{dU}{d\alpha}, \tag{4.1}$$

$$\frac{dJ}{d\alpha} \equiv \frac{\partial J}{\partial \alpha} + g^T u. \tag{4.2}$$

The term $\frac{dU}{d\alpha}$ can be obtained from the equation of state (e.g. Navier-Stokes) represented by a function R.

$$R(U(\alpha), \alpha) = 0. \tag{4.3}$$

Through tangent linearization, the derivative with respect to the design variable is

$$\frac{\partial R}{\partial U}\frac{dU}{d\alpha} + \frac{\partial R}{\partial \alpha} = 0 \tag{4.4}$$

this can be re-written as

$$\mathbf{A}u = f, \tag{4.5}$$

where $\mathbf{A}$ is the Jacobian, u is derivative of the state variables with respect to the design variables, and f is the negative partial derivative of the residual, R,with respect to the design variables.

While the term f is computationally inexpensive to obtain, the term $\frac{dU}{d\alpha}$ requires N solutions of the linear system for N design variables. This quickly becomes computationally prohibitive as the number of design variables increases. Adjoint methods can be used to reduce this cost by avoiding the calculation of this term.

4.2 Discrete Adjoint for single disciplines

It can be shown that the cost of evaluating gradients using the adjoint method is nearly independent of the number of design variables. Consequently, this method is highly recommended for problems with numerous design variables. The two main approaches to deriving adjoint equations are the discrete and continuous methods [11].

The continuous method analytically derives the adjoint equations before being discretised while the discrete method discretises the primal equations before formulating the discrete adjoint equations. This thesis uses the discrete method as it allows the use of Automatic Differentiation in order to avoid the tedious analytic derivation of the adjoint equations. This is discussed in more detail in section 4.3. First, the formulation of the discrete adjoint equations is presented in order to show the advantage of the adjoint method.

Assuming u satisfies Equation(4.5) which is a linear system of equations, the term $g^T u$ (in Equation (4.2)) can be evaluated in dual form as $v^T f$ [54]. Where the adjoint variable, v, is the solution to the set of equations

$$\begin{aligned} \left(\frac{\partial R}{\partial U}\right)^T v &= \left(\frac{\partial J}{\partial U}\right)^T, \quad (4.6) \\ \mathbf{A}^T v &= g. \quad (4.7) \end{aligned}$$

The equivalence of the two terms is shown as:

$$g^T u = (\mathbf{A}^T v)^T u = v^T \mathbf{A} u = v^T f. \quad (4.8)$$

Due to the adjoint equivalence, Equation (4.2) can be made independent of u

$$\frac{dJ}{d\alpha} \equiv \frac{\partial J}{\partial \alpha} + v^T f. \quad (4.9)$$

The advantage of the adjoint approach can be seen as Equation (4.7) only needs to be solved once for all design variables as opposed to solving Equation (4.2), N times for N design variables. Hence, the adjoint method computes the sensitivity at a cost which is linear in the number of cost functions and independent of the number of design variables [10].

4.3 Adjoint via Automatic Differentiation (AD)

Automatic Differentiation is a method for obtaining derivatives from computational models through the application of the chain rule. Consider a cost function ($J^T = [J_1, J_2]$) which is a function of three design variables ($x^T = [x_1, x_2, x_3]$) and which is calculated in a computer program through a series of function calls and intermediate variables. AD obtains the gradient of the cost function with respect to the design variables by applying the chain rule to each line of the program. Each operation can be seen as a function which is dependent on the previous operations and can be analytically differentiated. By assembling each intermediate derivative using the chain rule, the total derivative can then be obtained.

AD can be used to compute gradients in either forward mode or reverse mode. The forward mode computes the gradient as

$$\dot{\mathbf{J}} = \frac{d\mathbf{J}}{d\mathbf{x}}\dot{\mathbf{x}}, \tag{4.10}$$

$$\begin{bmatrix} \dot{J}_1 \\ \dot{J}_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial J_1}{\partial x_1} & \frac{\partial J_1}{\partial x_2} & \frac{\partial J_1}{\partial x_3} \\ \frac{\partial J_2}{\partial x_1} & \frac{\partial J_2}{\partial x_2} & \frac{\partial J_2}{\partial x_3} \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix}. \tag{4.11}$$

The process of retrieving gradients is done through seeding. For example, by seeding $\dot{x}^T = [1, 0, 0]$ we extract the first column of the Jacobian ($\frac{d\mathbf{J}}{d\mathbf{x}}$). Therefore, for N design variables, N number of computations need to be done. This makes Forward mode AD equivalent to the tangent linear method of computing exact derivatives.

On the other hand, Reverse mode A.D. computes the gradient as

$$\overline{\mathbf{x}} = \left(\frac{d\mathbf{J}}{d\mathbf{x}}\right)^T \overline{\mathbf{J}} \tag{4.12}$$

$$\begin{bmatrix} \overline{x}_1 \\ \overline{x}_2 \\ \overline{x}_3 \end{bmatrix} = \begin{bmatrix} \dfrac{\partial J_1}{\partial x_1} & \dfrac{\partial J_2}{\partial x_1} \\ \dfrac{\partial J_1}{\partial x_2} & \dfrac{\partial J_2}{\partial x_2} \\ \dfrac{\partial J_1}{\partial x_3} & \dfrac{\partial J_2}{\partial x_3} \end{bmatrix} \begin{bmatrix} \overline{J}_1 \\ \overline{J}_2 \end{bmatrix}. \tag{4.13}$$

By seeding $\overline{J}^T = [1, 0]$ we extract the gradient of one cost function with respect to all design variables. This makes the reverse mode equivalent to the adjoint method of obtaining derivatives. Therefore, Forward mode AD is more efficient when there are more outputs than inputs while Reverse mode AD is more efficient when there are more inputs than outputs.

In this thesis, the adjoint solvers are obtained using source transformation AD tool Tapenade [55]. Tapenade parses and analyses the original source code and produces a differentiated version. This process can also be automated to ensure that accurately differentiated code can be obtained with minimal code maintenance. Consequently, the developmental cost of obtaining the adjoint solvers can be significantly lower than the continuous adjoint method. As a partitioned coupling method is used, each solver is individually differentiated with respect to the boundary conditions it receives. The accuracy of the gradients obtained from both solvers is also evaluated individually.

The accuracy of the obtained gradients is of great importance as this affects the performance of gradient based methods. Inaccurate gradients can lead to less direct routes being taken to the optimum design, consequently, it is important the gradients obtained through AD are verified. This is done by comparing AD gradients with those obtained through Finite-Difference (FD). Although FD gradients suffer from accuracy issues, they still give a good indication of the expected values for gradients.

4.4 Adjoint CalculiX: Heat conduction solver

Based on the types of coupling algorithms used in this thesis, the solid solver outputs the solid heat flux, q_{sw}, and receives the fluid interface temperature, T_{fw}, as a boundary condition for the TFFB and TFRB coupling algorithms. For the hFRB method, the sink temperature, T_{sink}, and heat transfer coefficient, h, are specified as boundary conditions while the solid interface temperature, T_{sw}, and solid heat flux, q_{sw}, are returned.

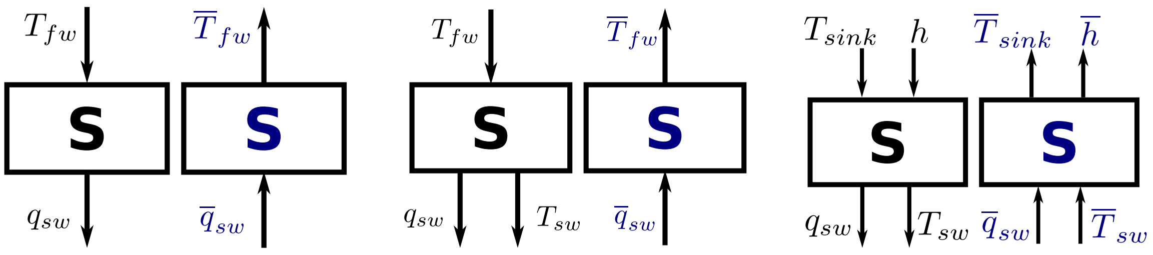


Figure 4.1: Input and output variables for all coupling algorithms (primal = black, adjoint = blue); Left:TFFB, Centre: TFRB, Right:hFRB.

As a result the gradient required from the solid solver is the gradient of the values output to the fluid domain w.r.t. the received boundary conditions. These gradients describe how the solid state is affected by the fluid state. This is summarised in Table 4.1.

Coupling algorithm	Input	Output	Gradient	Tangent seed	Adjoint seed
Neumann-Dirichlet (TFFB)	T_{fw}	q_{sw}	$\frac{\partial q_{sw}}{\partial T_{fw}}$	$\dot{T}_{fw}$	$\overline{q}_{sw}$
Robin-Dirichlet (TFRB)	T_{fw}	q_{sw}	$\frac{\partial q_{sw}}{\partial T_{fw}}$	$\dot{T}_{fw}$	$\overline{q}_{sw}$
Robin-Robin (hFRB)	T_{sink} h	q_{sw} T_{sw}	$\frac{\partial q_{sw}}{\partial T_{sink}}, \frac{\partial q_{sw}}{\partial h}$ $\frac{\partial T_{sw}}{\partial T_{sink}}, \frac{\partial T_{sw}}{\partial h}$	$\dot{T}_{sink}$ $\dot{h}$	$\overline{q}_{sw}$ $\overline{T}_{sw}$

Table 4.1: Gradients obtained from CalculiX.

Automatic Differentiation is used (in both forward and reverse mode) to obtain the gradient matrix in column 4 of Table 4.1 and the required derivatives are obtained using a $1 \times N$ seed vector (where N is the number of nodes/faces in which the boundary conditions are applied). The seed vectors for each coupling algorithm are shown in column 5 for the tangent mode and column 6 for the adjoint mode.

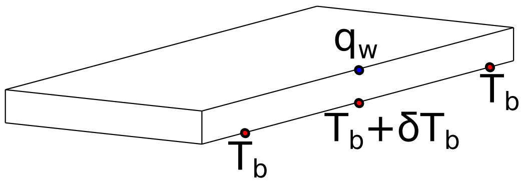
In order to differentiate CalculiX, extensive code preparation was required before Tapenade (version 3.13) could be used. At the current time, the development of Tapenade to handle a mix of C and Fortran is still on going. The C-Fortran code was able to be differentiated in Forward/Tangent mode with several modifications to the original source code. In particular, Multithreading in the solver had to be removed as it was not supported by Tapenade. New macros also needed to be defined for variable allocations and reallocations and files containing #IFDEF directives needed to be preprocessed before calling Tapenade. Furthermore, CalculiX is partly written in Fortran77 which allows the use of arrays of arbitrary length. The sizes of numerous arrays needed to be explicitly specified in order for Tapenade to create differentiated variables and initialise them with zero. Several legacy statements had to be removed and several intrinsic functions had to be differentiated separately after differentiating the main routine.

Despite these modifications, it was not possible to differentiate the C-Fortran code in Reverse/Adjoint mode using Tapenade. These problems would be significantly reduced if the code was written in only one programming language and better yet, if it was written with the intention to be differentiated using AD. Therefore, the top routine and other relevant C routines were converted to Fortran and successfully differentiated using AD. The differentiation of the linear solver was done manually as this is the recommended way of differentiating linear solvers and more details are provided in Appendix A.

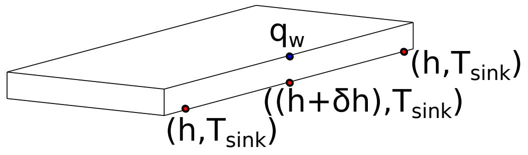
The adjoint code initially consumed a huge amount memory due to the need to store intermediate variables for the gradient calculation. In particular, for some three dimensional arrays where only a subset was required for intermediate calculations, Tapenade stored the whole array each time. This lead to a prohibitive increase in the memory consumption of the code. This problem was resolved by manually editing the differentiated code to store only the subset it required rather than storing the whole array.

4.4.1 Sensitivity verification

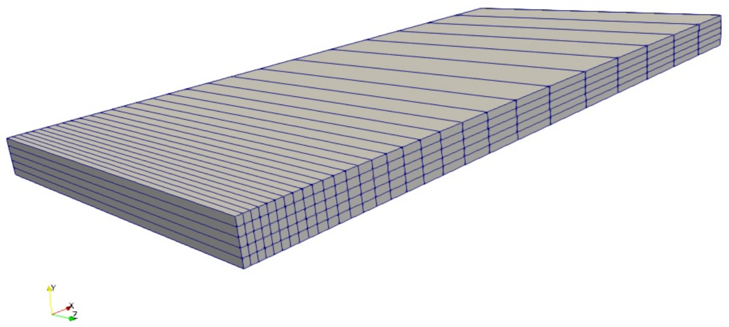
The gradients obtained through AD of CalculiX are verified by comparing the values to those obtained through finite differences. The finite difference gradients are obtained by perturbing the value of the boundary conditions in a few random nodes as a selection of design variables (α) and measuring the change in the heat flux at a point on the interface. Sensitivity verification is carried out on a the flat plate test case (see Figure 4.2).



(a) TFFB and TFRB perturbation.



(b) hFRB perturbation.



(c) Solid mesh with 348 nodes.

Figure 4.2: Perturbation of solid domain design variables for finite differences.

For the TFFB and TFRB coupling algorithms, the boundary condition to be perturbed is the imposed temperature at the bottom of the plate T_b while for the hFRB algorithm, either the heat transfer coefficient h or sink temperature T_{sink} is perturbed. A perturbation of the boundary condition of one node leads to a new temperature and heat flux distribution in the entire solid domain. For example, by perturbing the bottom temperature T_b and measuring the change in heat flux at a point (blue node in figure 4.2), the central difference gradient can then be calculated as

$$\frac{dq_w}{dT_b} = \frac{q_w^{(T_b+\delta T_b)} - q_w^{(T_b-\delta T_b)}}{2 \cdot \delta T_b}, \tag{4.14}$$

where $q_w^{(T_b+\delta T_b)}$ is the interface heat flux obtained for a positive perturbation of magnitude δT_b to the temperature at the bottom of the plate. For the hFRB method, a perturbation in the sink temperature or heat transfer coefficient at one node leads to a new temperature and heat flux distribution and the central difference gradient can then be calculated in a similar way.

CalculiX is also differentiated to obtain the gradients of the boundary conditions w.r.t. coordinates $\vec{x}$, where $\vec{x}$ represents the cartesian coordinates x, y, z. This enables the solver to be used for shape optimisation problems.

The comparison of tangent, adjoint, and central difference gradients for all coupling algorithms is shown in Tables 4.2 - 4.5 below.

Gradient method	Design variable		
	1 [E-07]	2 [E-07]	3 [E-07]
Tangent	3.5485219015784**849**	7.1842117605076**197**	7.4222539856771**185**
Adjoint	3.5485219015784**918**	7.1842117605076**208**	7.4222539856771**386**
CD [$\Delta = 0.6$K]	3.548**439053702168**	7.184**492763675128**	7.422**067938023246**

Table 4.2: TFFB & TFRB: Gradient of heat flux q_w w.r.t temperature $\left(\frac{\partial q_w}{\partial T_w}\right)$.

Gradient method	Design variable		
	1	2	3
Tangent	-6.23540559322260**18**	-0.15740871006555**840**	-0.71552157932**750082**
Adjoint	-6.23540559322260**09**	-0.15740871006555**6009**	-0.71552157932**749982**
CD [$\Delta = 100$ K]	-6.2354055932**540176**	-0.15740871006**850285**	-0.7155215793**1657345**

Table 4.3: hFRB: Gradient of heat flux q_w w.r.t sink temperature $\left(\frac{\partial q_w}{\partial T_{sink}}\right)$.

The adjoint and tangent gradient match to machine precision as expected for two analytically accurate methods. They also show good agreement with Central Difference (CD) gradients. The CD gradients match the adjoint gradients to a precision of 10 decimal places for both the temperature ($\frac{dq_w}{dT_w}$) and sink temperature ($\frac{dq_w}{dT_{sink}}$). Therefore the gradients are considered accurate.

Gradient method	Design variable		
	1 [E-06]	2 [E-07]	3 [E-08]
Tangent	-2.3181592565718**062**	-2.0916730807518**237**	8.259775594282**0205**
Adjoint	-2.3181592565718**172**	-2.0916730807518**353**	8.259775594282**2984**
CD [$\Delta = 10$ W/m^2/K]	-2.3181**609321909490**	-2.09167**77430102231**	8.2597**944128792731**

Table 4.4: hFRB: Gradient of heat flux q_w w.r.t heat transfer coefficient $\left(\frac{\partial q_w}{\partial \tilde{h}}\right)$.

Gradient method	Design variable		
	x	y	z
Tangent	-106418.706322105**34**	-277295.888147918**10**	-8012.90183165392**41**
Adjoint	-106418.706322105**53**	-277295.888147918**51**	-8012.90183165392**32**
CD [$\Delta = 10^{-7}$m]	-106418.706**40029083**	-277295.888**03595107**	-8012.9018**897423521**

Table 4.5: Gradient of heat flux q_w w.r.t coordinates $\left(\frac{\partial q_w}{\partial x}\right)$.

4.5 Adjoint mgOpt: Flow solver

The gradients required from the flow solver are the gradients of the fluid interface variables which are passed to the solid domain (i.e. T_{fw}, T_{sink}, $\tilde{h}$) w.r.t to the received fluid boundary inputs (q_{sw} and $\tilde{T}_s$) as shown in Figure 4.3. These gradients describe how the fluid state is affected by the solid state through the CHT boundary conditions. The gradient of the virtual solid conductivity ($\tilde{R}$) is not considered as this is a user defined number which does not affect the solution. The required gradients from the flow solver are summarised in Table 4.6.

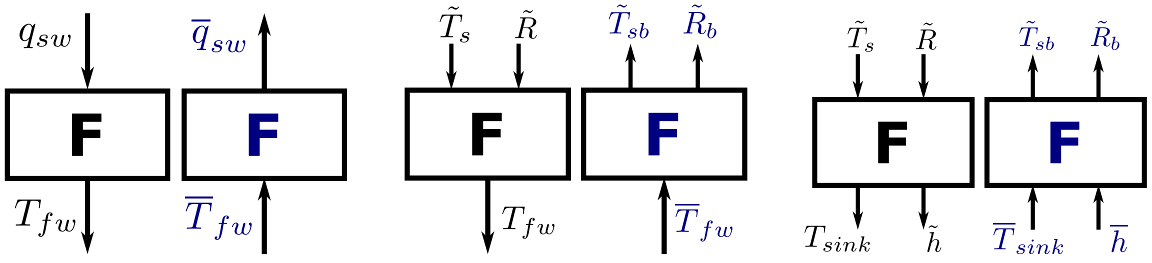


Figure 4.3: Input and output variables for all coupling algorithms (primal = black, adjoint = blue); Left:TFFB, Centre: TFRB, Right:hFRB.

The adjoint flow solver has been developed during previous PhD programs [11] [31] [32]. This thesis extends the capabilities of the flow solver to include CHT applications. This was

Coupling algorithm	Input	Output	Gradient	Tangent seed	Adjoint seed
Neumann-Dirichlet (TFFB)	q_{sw}	T_{fw}	$\frac{\partial T_{fw}}{\partial q_{sw}}$	$\dot{q}_{sw}$	$\overline{T}_{fw}$
Robin-Dirichlet (TFRB)	$\tilde{T}_s$, $\tilde{R}$	T_{fw}	$\frac{\partial T_{fw}}{\partial \tilde{T}_s}$	$\dot{T}_s$	$\overline{T}_{fw}$
Robin-Robin (hFRB)	$\tilde{T}_s$, $\tilde{R}$	T_{sink}, h	$\frac{\partial T_{sink}}{\partial \tilde{T}_s}$, $\frac{\partial h}{\partial \tilde{T}_s}$	$\dot{T}_s$	$\overline{h}$ $\overline{T}_{sink}$

Table 4.6: Gradients obtained from the flow solver.

done by implementing and differentiating the CHT boundary conditions. The flow solver is written in Fortran 90/95 and was developed with the intention to be differentiated using AD. Consequently, extensive code modifications were unnecessary and the development of the tangent and adjoint versions of the solver was successfully automated.

4.5.1 Sensitivity verification

The gradients obtained from the flow solver are also verified by comparing the values to those obtained through Central Difference (CD) and the flat plate test case is used for sensitivity verification.

Gradient method	Design variable		
	1 [E-05]	2 [E-04]	3 [E-05]
Tangent	2.309724896279**3258**	6.350257330617**1983**	5.61072133311**07320**
Adjoint	2.309724896279**2699**	6.350257330617**1387**	5.61072133311**05077**
CD [$\Delta = 10^{-4}$ W/m^2]	2.30**13342342892429**	6.3**306970332632773**	5.6**497242439945694**

Table 4.7: TFFB: Gradient of temperature T_w w.r.t heat flux q_w $\left(\frac{\partial T_w}{\partial q_w}\right)$.

The results in Table 4.7 - 4.10 show that the CD gradients agree with the Tangent and Adjoint gradients, and the Adjoint and Tangent match to machine precision therefore the adjoint gradients are considered accurate.

Gradient method	Design variable		
	1 [E-04]	2 [E-02]	3 [E-04]
Tangent	3.61489759605278**68**	1.064552426883286**7**	6.04930316108196**74**
Adjoint	3.61489759605278**47**	1.064552426883286**4**	6.04930316108196**53**
CD [$\Delta = 10^{-4}$ K]	3.60**91781415355700**	1.06**29397214264221**	6.0**780773007233313**

Table 4.8: TFRB: Gradient of temperature T_w w.r.t heat flux $\tilde{T}_s$ $\left(\frac{\partial T_w}{\partial \tilde{T}_s}\right)$.

Gradient method	Design variable		
	1 [E-04]	2 [E-02]	3 [E-04]
Tangent	3.61601604274501**39**	1.064880358184998**0**	6.05114279872326**29**
Adjoint	3.61601604274501**28**	1.064880358184998**1**	6.05114279872326**40**
CD [$\Delta = 10^{-6}$ K]	3.61**02202708813513**	1.06**32665710848718**	6.0**797826032891555**

Table 4.9: hFRB: Gradient of sink Temperature T_{sink} w.r.t $\tilde{T}_s$ $\left(\frac{\partial T_{sink}}{\partial \tilde{T}_s}\right)$.

Gradient method	Design variable		
	1 [E-02]	2	3 [E-02]
Tangent	1.44045606517038**11**	0.424200398282454**43**	2.41051238573111**73**
Adjoint	1.44045606517038**09**	0.424200398282454**37**	2.41051238573111**69**
CD [$\Delta = 10^{-6}$ K]	1.4**382142884035904**	0.42**355956490306801**	2.4**219237578411896**

Table 4.10: hFRB: Gradient of heat transfer coefficient $\tilde{h}$ w.r.t. $\tilde{T}_s$ $\left(\frac{\partial \tilde{h}}{\partial \tilde{T}_s}\right)$.

4.6 Summary

This chapter discussed the principles and methods of obtaining adjoint gradients. The derivation of the adjoint equations and the computational efficiency of adjoint methods was shown. It was shown that the adjoint method was the most ideal method of obtaining gradients as the cost does not scale with the number of design variables. It was also shown that Automatic Differentiation is a useful tool for the discrete adjoint method and how it was used to differentiate the numerical solvers was discussed.

The adjoint gradients obtained were verified by comparing with finite difference gradients. The two adjoint solvers developed are representative of individual but co-dependent parts of a multidisciplinary system. The two adjoint solvers can now be combined to obtain the full gradients of the multidisciplinary system and how this is done is discussed in the following chapter.

Chapter 5

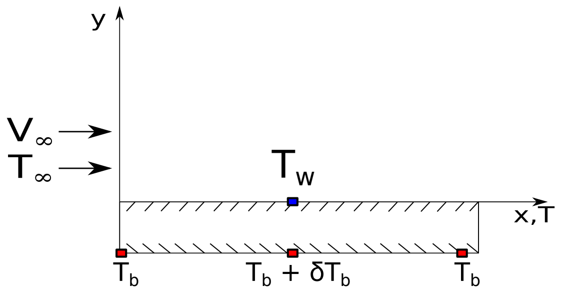
Differentiation of coupling algorithms

In the previous chapter, two independent adjoint solvers, which can be used for single disciplinary optimisation, were developed. For these to be used for Multidisciplinary Design Optimisation (MDO), the fluid and solid adjoint solvers have to be coupled in a similar manner as the primal coupling algorithms. However, rather than exchanging state variables such as temperature and heat flux, gradients need to be exchanged to obtain gradients from the coupled system. This chapter describes the process of coupling two discrete adjoint solvers in order to obtain the gradients of the coupled system w.r.t the design variables.

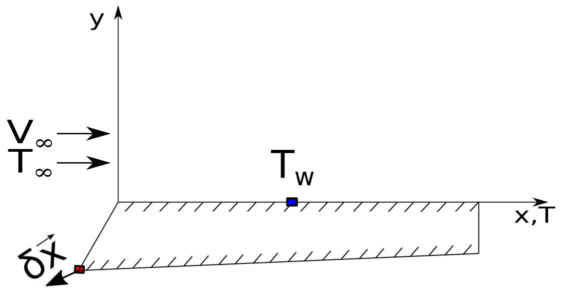
The use of adjoint methods for MDO have been proposed in [9] [10] and Martins [3] [8] uses the continuous adjoint methods for Fluid-Structure Interaction (FSI) which is analogous to CHT. Continuous adjoint methods have also been utilised for CHT optimisation problems in [12] [13] [14] [15] [16]. As discussed in Chapter 1, the discrete adjoint method is adopted in the current work and Automatic Differentiation (as described in Chapter 4) is used to reduce the effort of adjoint solver development.

The previous studies on adjoint CHT optimisation have also been limited to coupling algorithms which combine Dirichlet and Neumann boundary conditions. By differentiating the three coupling algorithms described in Chapter 2, Dirichlet, Neumann, and Robin boundary conditions are all considered. This results in a discrete method for differentiated Neuman-Dirichlet coupling algorithms and 2 newly differentiated Robin-based coupling algorithms. Moreover, the generic manner of the differentiation allows for straightforward extension to other coupling algorithms than the three considered.

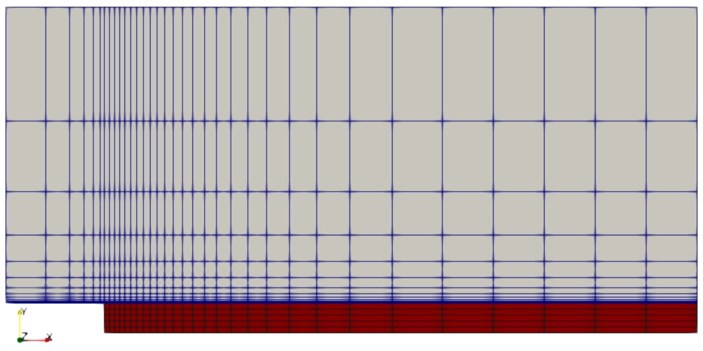
The accuracy of the three differentiated algorithms is tested using the coupled flat plate problem discussed in Section 3.1. The objective is to compute the gradient of the interface temperature with respect to a change in the solid temperature boundary condition.



(a) Temperature perturbation.



(b) $\vec{x}$ coordinate perturbation



(c) Fluid and Solid meshes for gradient verification.

Figure 5.1: Design variable perturbations for central difference.

The plate has a fixed temperature T_b at the bottom and comes in contact with fluid of a different temperature (see Section 3.1). A perturbation in the temperature of a node at the bottom results in a change in the heat flux into the fluid domain. This perturbation travels through the coupling and results in a new interface temperature T_w. Similarly, a coordinate perturbation (x, y, z) changes the volume of the plate and leads to a change in the heat flux at the fluid-solid interface and consequently alters the solution of the coupled problem. Therefore, the effect of these perturbations on the interface temperature is described by two gradients $\frac{dT_w}{dT_b}$ and $\frac{dT_w}{d\vec{x}}$ where $\vec{x} = [x, y, z]$.

5.1 Finite Difference

The effect of a perturbation on the coupled solution (i.e gradient) can be calculated using Central Differences (CD) as

$$\frac{dT_w}{dT_b} = \frac{T_w^{(T_b+\delta T_b)} - T_w^{(T_b-\delta T_b)}}{2\delta T_b}. \tag{5.1}$$

To compute the gradient in Equation (5.1), the temperature at the blue node in Figure 5.1a is selected as the point to measure the interface temperature (T_w) while several nodes at the bottom of the plate are chosen as design variables. The temperature of only one of the design variables (red nodes) is perturbed while all others nodes are are held constant. Equation (5.1) can then be used to calculate the gradient for the perturbed node. Similarly, to calculate the gradients w.r.t to a coordinate perturbation, the red node at the bottom left corner of the plate in Figure 5.1b is perturbed and the central difference is used to calculate $\frac{dT_w}{d\vec{x}}$. However, this has to be done N times for N design variables, which soon becomes computationally expensive.

To highlight this point, in the flat plate test case in Section 3.1, approximately twenty coupling iterations were needed to obtain the interface temperature T_w. Therefore, in order to calculate $\frac{dT_w}{dT_b}$for only five design variables, one would need to run a total of a hundred coupling iterations just to obtain $T_b + \delta T_b$ and another hundred for $T_b - \delta T_b$. For industrial cases where days might be needed to converge the flow equations to a reasonable level and the number of design variables might be of the order 10^2, the use of divided soon becomes unfeasible.

Due to the high computational cost of central differences, and the fact that the accuracy of the obtained gradient suffers from truncation and round-off errors, more efficient methods of computing gradients are required.

5.2 Discrete adjoint for coupled problems

As discussed in Chapter 4, the adjoint method can be used to obtain the desired gradients in one adjoint solution, as opposed to solving N times for N design variables when finite difference or Tangent linear methods are used. In Chapter 4, the derivation of adjoint equations was shown for only single disciplinary problems. This section applies the same principle to coupled problems.

The derivation of adjoint equations for coupled problems is similar to the process described in Section(4.2), the main difference being that scalars become matrices. A coupled CHT optimisation problem will be used to outline the procedure. Let the objective function be denoted as $I(\alpha, U, W)$ where α denotes the design variables, and U and W represent the fluid and solid state variables respectively. The sensitivity of the cost function can then be written as

$$\frac{dI}{d\alpha} = \frac{\partial I}{\partial \alpha} + \frac{\partial I}{\partial U}\frac{dU}{d\alpha} + \frac{\partial I}{\partial W}\frac{dW}{d\alpha}, \tag{5.2}$$

$$\frac{dI}{d\alpha} = \frac{\partial I}{\partial \alpha} + \begin{bmatrix} \frac{\partial I}{\partial U} & \frac{\partial I}{\partial W} \end{bmatrix} \begin{bmatrix} \frac{dU}{d\alpha} \\ \frac{dW}{d\alpha} \end{bmatrix}. \tag{5.3}$$

This can be re-written as

$$\frac{dI}{d\alpha} = \frac{\partial I}{\partial \alpha} + g^T u. \tag{5.4}$$

As mentioned in Chapter 4, the term $g^T u$ is expensive to solve. Hence, it is advantageous to use the adjoint formulation to solve this problem. The derivatives of the state variables for fluid and solid with respect to the design variables $\left(\frac{dU}{d\alpha}, \frac{dW}{d\alpha}\right)$ can be obtained from the state equations

$$F(\alpha, U, W) = 0, \tag{5.5}$$

$$S(\alpha, U, W) = 0. \tag{5.6}$$

Where F is used to represent the RANS equations, and S represents the heat equation. The derivative of the state equations with respect to the design variables which are required to solve Equation (5.2) are obtained from

$$\frac{\partial F}{\partial \alpha} + \frac{\partial F}{\partial U}\frac{dU}{d\alpha} + \frac{\partial F}{\partial W}\frac{dW}{d\alpha} = 0 \tag{5.7}$$

$$\frac{\partial S}{\partial \alpha} + \frac{\partial S}{\partial U}\frac{dU}{d\alpha} + \frac{\partial S}{\partial W}\frac{dW}{d\alpha} = 0. \tag{5.8}$$

This can be re-written as

$$\begin{bmatrix} \frac{\partial F}{\partial U} & \frac{\partial F}{\partial W} \\ \frac{\partial S}{\partial U} & \frac{\partial S}{\partial W} \end{bmatrix} \begin{bmatrix} \frac{\partial U}{\partial \alpha} \\ \frac{\partial W}{\partial \alpha} \end{bmatrix} = \begin{bmatrix} -\frac{dF}{d\alpha} \\ -\frac{dS}{d\alpha} \end{bmatrix}. \tag{5.9}$$

The diagonal terms are the Jacobians of each discipline while the off-diagonal terms show how the states of one discipline affect the state of the other e.g. how the fluid temperature affects the solid flux and vice-versa. Equation (4.5) is written below for convenience.

$$\begin{aligned} \mathbf{A}u &= f, \\ u &= \begin{bmatrix} \frac{\partial F}{\partial U} & \frac{\partial F}{\partial W} \\ \frac{\partial S}{\partial U} & \frac{\partial S}{\partial W} \end{bmatrix}^{-1} \begin{bmatrix} -\frac{\partial F}{\partial \alpha} \\ -\frac{\partial S}{\partial \alpha} \end{bmatrix}. \end{aligned} \tag{5.10}$$

Therefore the sensitivity of the cost function, Equation (5.3), for the coupled problem can be written as

$$\frac{dI}{d\alpha} = \frac{\partial I}{\partial \alpha} + \begin{bmatrix} \frac{\partial I}{\partial U} & \frac{\partial I}{\partial W} \end{bmatrix} \begin{bmatrix} \frac{\partial F}{\partial U} & \frac{\partial F}{\partial W} \\ \frac{\partial S}{\partial U} & \frac{\partial S}{\partial W} \end{bmatrix}^{-1} \begin{bmatrix} -\frac{\partial F}{\partial \alpha} \\ -\frac{\partial S}{\partial \alpha} \end{bmatrix}. \tag{5.11}$$

This can be shortened to

$$\frac{dI}{d\alpha} = \frac{\partial I}{\partial \alpha} + (g^T A^{-1})f. \tag{5.12}$$

The cost of calculating the gradient can be reduced through the adjoint method. The adjoint variables are the solution to the adjoint equation

$$\begin{bmatrix} \frac{\partial F}{\partial U} & \frac{\partial F}{\partial W} \\ \frac{\partial S}{\partial U} & \frac{\partial S}{\partial W} \end{bmatrix}^T \begin{bmatrix} \psi \\ \phi \end{bmatrix} = \begin{bmatrix} \frac{\partial I}{\partial U} \\ \frac{\partial I}{\partial W} \end{bmatrix}, \tag{5.13}$$

$$\mathbf{A}^T v = g. \tag{5.14}$$

Where $v^T = [\psi, \phi]$ represents the fluid and solid adjoint variables respectively.

$$v = (A^T)^{-1} g, \tag{5.15}$$

$$v^T = g^T (A^{-1}). \tag{5.16}$$

Therefore, after only one solve of Equation (5.14) for the adjoint variable v, the sensitivity can be calculated as

$$\frac{dI}{d\alpha} = \frac{\partial I}{\partial \alpha} + v^T f, \tag{5.17}$$

$$\frac{dI}{d\alpha} = \frac{\partial I}{\partial \alpha} + \begin{bmatrix} \psi & \phi \end{bmatrix} \begin{bmatrix} -\dfrac{\partial F}{\partial \alpha} \\ -\dfrac{\partial S}{\partial \alpha} \end{bmatrix}. \tag{5.18}$$

5.3 Differentiating the partitioned approach

In the partitioned coupling approach, the Jacobian of the coupled system in Equation (5.14) is not calculated. Therefore, the adjoint solution is obtained through an iterative approach, similar to the primal coupling, to compensate for the missing off-diagonals. The same approach used by Martins for FSI [3], called the lagged-coupled adjoint, is applied here to solve the coupled adjoint system of equations. The adjoint of system of each discipline is solved using the adjoint solution of the other discipline from the previous iteration.

$$\left(\frac{\partial F}{\partial U}\right)^i \psi^i = \left(\frac{\partial I}{\partial U}\right)^i - \left(\frac{\partial S}{\partial U}\phi\right)^{i-1} \tag{5.19}$$

$$\left(\frac{\partial S}{\partial W}\right)^i \phi^i = \left(\frac{\partial I}{\partial W}\right)^i - \left(\frac{\partial F}{\partial W}\psi\right)^{i-1}, \tag{5.20}$$

where i is the current coupling iteration. The partial derivatives ($\frac{\partial S}{\partial U}$, $\frac{\partial F}{\partial W}$) in Equations (5.19) and (5.20) depend on the type of coupling algorithm used (see Table 5.1) and are obtained by differentiating the solvers w.r.t. the coupling boundary conditions as described in Sections 4.4 and 4.5.

Partial derivative	Coupling algorithm		
	TFFB	TFRB	hFRB
$\frac{\partial S}{\partial U}$	$\frac{\partial q_{sw}}{\partial T_{fw}}$	$\frac{\partial q_{sw}}{\partial T_{fw}}$	$\frac{\partial q_{sw}}{\partial T_{sink}}$, $\frac{\partial q_{sw}}{\partial h}$, $\frac{\partial T_{sw}}{\partial T_{sink}}$, $\frac{\partial T_{sw}}{\partial h}$
$\frac{\partial F}{\partial W}$	$\frac{\partial T_{fw}}{\partial q_{sw}}$	$\frac{\partial T_{fw}}{\partial \tilde{T}_s}$	$\frac{\partial T_{sink}}{\partial \tilde{T}_s}$, $\frac{\partial h}{\partial \tilde{T}_s}$

Table 5.1: Description of multidisciplinary partial derivative terms.

The iterative approach to solving Equations (5.19) and (5.20) requires the differentiation of the coupling algorithms and this is described in the following sections of this chapter. Pseudocode of the differentiation of the three coupling algorithms is also provided in Appendix B. Once the differentiated coupling algorithms converge, the total sensitivity in Equation (5.18) can then be computed.

5.4 TFFB

The Temperature Forward Flux Back tangent run is initialised by seeding a vector of $\dot{T}_b$ (or $\dot{x}$) which has a unitary value only at the node of interest (e.g one of the red nodes in Figure 5.1). The solid tangent solver ($\mathbf{S_d}$) is then run to obtain $\dot{q}_{sw}$ which is the gradient of the wall heat flux q_{sw} w.r.t bottom temperature T_b at the seeded node. $\dot{q}_{sw}$ is then passed to the fluid tangent solver ($\mathbf{F_d}$) which is used to obtain $\dot{T}_{fw}$. $\dot{T}_{fw}$ is the gradient of the interface temperature w.r.t the seeded bottom temperature after a single coupling iteration. The same number of coupling iterations as the primal are then performed.

$$\begin{aligned} \dot{q}^i_{sw} &= \mathbf{S_d}(\dot{T}_b, \dot{T}^i_{fw}), && (5.21) \\ \dot{T}^{i+1}_{fw} &= \mathbf{F_d}(\dot{q}^i_{sw}). && (5.22) \end{aligned}$$

During the tangent coupling iterations, the primal variables are also computed and used to calculate the relevant tangent variables. The final result obtained is the gradient $\frac{dT_w}{dT_b}$ for the converged coupled solution. In order to compute $\frac{dT_w}{d\vec{x}}$, three separate tangent coupling iterations have to be run for the three degrees of freedom (x, y, z) for each perturbed point. This shows the disadvantage of the tangent linear method when there are more inputs than outputs.

The adjoint run of the TFFB algorithm, starts with seeding a vector $\overline{T}^i_{fw}$ which is set to 1 for the blue node in Figure 5.1 while all others are set to 0. $\overline{T}_{fw}$ is used by the adjoint flow solver ($\mathbf{F_b}$) to obtain the adjoint heat flux $\overline{q}_{sw}$. This is then passed directly to the adjoint solid ($\mathbf{S_b}$) solver to obtain an update of the adjoint temperature.

$$\begin{aligned} \overline{q}^i_{sw} &= \mathbf{F_b}(\overline{T}^i_{fw}), && (5.23) \\ \overline{\vec{x}}^i, \overline{T}^i_b, \overline{T}^{i-1}_{fw} &= \mathbf{S_b}(\overline{q}^i_{sw}). && (5.24) \end{aligned}$$

As shown in listing B.5, the reverse coupling requires the intermediate state variables. During the primal run, the solid and fluid states are saved after each iteration. These are then retrieved for the adjoint solvers during each reverse coupling iteration. For example, if 20 primal iterations were run, during reverse iteration 5, the fluid and solid solutions for iteration 15 are retrieved for the adjoint solvers.

The reverse loop is performed for the same number of iterations as for the primal solution and at the end of each coupling iteration, the gradient $\overline{T}_b$ is accumulated. This single adjoint solve obtains the gradient of the interface temperature at the blue node w.r.t to all design variables (red nodes). A block structure representation of the reverse differentiated coupling algorithm is shown in Figure 5.2. In fact, both $\frac{dT_w}{dT_b}$ and $\frac{dT_w}{d\vec{x}}$ are obtained in one reverse solve highlighting the advantage and cost savings of the adjoint method.

Gradient method	Design variable		
	1 [E-04]	2 [E-02]	3 [E-04]
Tangent	3.57424996556775**86**	1.09156391242752**44**	6.2298230582871**555**
Adjoint	3.57424996556775**91**	1.09156391242752**37**	6.2298230582871**479**
CD [$\Delta = 10^{-4}$ K]	3.5**665266295836773**	1.0**898488085331337**	6.2**618937590741552**

Table 5.2: TFFB gradient of temperature T_w w.r.t bottom temperature T_b ($\frac{dT_w}{dT_b}$) after 20 reverse coupling iterations.

Table 5.2 shows gradients for $\frac{dT_w}{dT_b}$ while Table 5.3 shows the gradients for $\frac{dT_w}{d\vec{x}}$. Both Tables show good agreement between the tangent, adjoint, and central difference methods.

Gradient method	Design variable		
	x	y	z
Tangent	16.3537546**63042668**	-13.270758**597756023**	0.410171858**55131066**
Adjoint	16.3537546**47042667**	-13.270758**610753868**	0.410171858**29135394**
CD [$\Delta = 10^{-7}$m]	16.3**21095586135925**	-13.2**44326737549272**	0.4**0858196825865889**

Table 5.3: TFFB gradient of temperature T_w w.r.t coordinates.

5.5 TFRB

To compute $\frac{dT_w}{dT_b}$, the Temperature Forward solid coefficient Back method would also start with a seed vector for $\dot{T}_b$ as described for the TFFB algorithm. However, to compute $\frac{dT_w}{dy}$, a seed 3×N seed matrix (where N is the number of mesh points) is required which has a unitary value only for the $y-$component of one design variable (i.e. red node in Figure 5.1).

This matrix is passed to the tangent solid solver, which returns $\dot{q}_{sw}$. $\dot{q}_{sw}$ is the gradient of the heat flux $\vec{q}_{sw}$ w.r.t to a y perturbation of the design node. $\dot{q}_{sw}$ is then used by the tangent differentiated version of the routine which calculates Equation (2.38) to obtain $\tilde{T}_{sd}$. $\tilde{T}_{sd}$ is then passed to the tangent flow solver to obtain $\dot{T}_{fw}$ which is the gradient of the interface temperature w.r.t to the y perturbation of the design node after one coupling iteration.

$$\begin{aligned}
\dot{q}^i_{sw} &= \mathbf{S_d}(\dot{T}_b|\dot{y}, \dot{T}^i_{sw}), && (5.25)\\
\tilde{T}^i_{sd} &= f(\dot{q}_{sw}, \dot{T}_{sw}, \tilde{R}), \implies \text{(calc. } \tilde{T}_{sd}) && (5.26)\\
\dot{T}^i_{fw} &= \mathbf{F_d}(\tilde{T}^i_{sd}, \tilde{R}), && (5.27)\\
\dot{T}^{i+1}_{sw} &= \dot{T}^i_{fw}. && (5.28)
\end{aligned}$$

This loop is done for the same number of iterations as the primal. The final result is only $\frac{dT_w}{dy}$, and separate tangent solutions are required for $\frac{dT_w}{dx}$, $\frac{dT_w}{dz}$, and $\frac{dT_w}{dT_b}$.

The adjoint TFRB run is started by passing the adjoint fluid temperature $\overline{T}_{fw}$ to the flow solver to obtain $\tilde{T}_{sb}$. The reverse differentiated routine for Equation (2.38) is also required to obtain the adjoint heat flux and temperature as shown in Equations (5.30). The solid solver then uses the adjoint heat flux to obtain $\overline{T}_{sw}$. The two adjoint interface temperatures then summed to provide an update for the next iteration.

$$\begin{aligned}
\tilde{T}^i_{sb} &= \mathbf{F_b}(\overline{T}^i_{fw}), && (5.29)\\
\overline{q}^i_{sw}, \overline{T}_{sw} &= f(\tilde{T}^i_{sb}, \tilde{R}) \implies \text{(calc. } \tilde{T}_{sb}), && (5.30)\\
\overline{\vec{x}}^i, \overline{T}^i_b, \overline{T}^i_{fw} &= \mathbf{S_b}(\overline{q}^i_{sw}), && (5.31)\\
\overline{T}^{i-1}_{fw} &= \overline{T}^i_{fw} + \overline{T}_{sw}. && (5.32)
\end{aligned}$$

The loop is done for the same number of coupling iterations and $\overline{T}_b$ and $\overline{\vec{x}}$ are accumulated. The final result returns $\frac{dT_w}{dT_b}$ and $\frac{dT_w}{d\vec{x}}$ and is shown in Tables 5.4 and 5.5. Good agreement is seen between the tangent, adjoint, and central difference methods.

5.6 hFRB

The tangent heat transfer coefficient Forward solid coefficient Back (hFRB) algorithm follows the same procedure as the TFFB and TFRB algorithms. The algorithm begins with

Gradient method	Design variable		
	1 [E-04]	2 [E-02]	3 [E-04]
Tangent	3.57355505580010**70**	1.091430424038719**9**	6.2333366050062**219**
Adjoint	3.57355505580010**43**	1.091430424038719**6**	6.2333366050062**176**
CD [$\Delta = 10^{-4}$ K]	3.5**665607356349938**	1.0**898481832555262**	6.2**619392338092439**

Table 5.4: TFRB gradient of temperature T_w w.r.t bottom temperature T_b.

Gradient method	Design variable		
	x	y	z
Tangent	16.370061**472907469**	-13.283993**573142387**	0.41096593**285022071**
Adjoint	16.370061**462907479**	-13.283993**581266039**	0.41096593**268774745**
CD [$\Delta = 10^{-7}$m]	16.3**21095586135925**	-13.2**44325032246707**	0.4**0858196825865889**

Table 5.5: TFRB gradient of temperature T_w w.r.t coordinates.

a seed vector for the design variable ($\dot{T}_b$ or $\dot{\vec{x}}$) which is passed to the tangent solid solver to obtain $\dot{q}_{sw}$. This is then used by the differentiated version of the routine which calculates $\tilde{T}_s$ to compute $\tilde{T}_{sd}$. $\tilde{T}_{sd}$ is used by the tangent flow solver to obtain $\dot{T}_{fw}$ and $\dot{q}_{fw}$ which are in turn used to calculate $\dot{T}^i_{sink}$ and $\tilde{h}_d$. If $\tilde{h}$ is a user specified value like $\tilde{R}$, then the gradient $\tilde{h}_d$ would be zero. However, due to the present implementation of the hFRB algorithm (described in section 2.3.2), $\tilde{h}$ is a function of the fluid state ($h = f(\lambda(\mu(T_w)))$) therefore $\tilde{h}_d$ is non-zero. During the primal run, nodal values of h and T_{sink} are converted into facial values for CalculiX and the routine which does this is also differentiated using AD.

$$\begin{aligned}
\dot{q}^i_{sw} &= \mathbf{S_d}(\dot{T}_b), && (5.33)\\
\tilde{T}^i_{sd} &= f(\dot{q}_{sw}, \dot{T}_{sw}, \tilde{R}), \implies (\text{calc. } \tilde{T}_{sd}) && (5.34)\\
\dot{T}^i_{fw}, \dot{q}^i_{fw} &= \mathbf{F_d}(\tilde{T}^i_{sd}, \tilde{R}), && (5.35)\\
\dot{T}^i_{sink}, \tilde{h}_d &= f(\dot{T}_{fw}, \dot{q}_{fw}) \implies (\text{calc. } \dot{T}_{sink}) && (5.36)\\
q^{i+1}_{sw}, T^{i+1}_{sw} &= \mathbf{S_d}(\dot{T}_b, \dot{T}^i_{sink}, \tilde{h}_d) && (5.37)
\end{aligned}$$

For the reverse differentiated mode, the adjoint wall temperature $\overline{T}_{fw}$ is used by the flow solver to obtain the adjoint solid sink temperature $\overline{T}_s$. This is then converted into an adjoint heat flux $\overline{q}_{sw}$ and adjoint temperature $\overline{T}_{sw}$ using the differentiated routine which calculates the Robin parameters T_{sink} and $\tilde{h}$ for the solid domain.

$$\begin{aligned}
\overline{T}_s^i, \overline{R}^i &= \mathbf{F_b}(\overline{T}_{fw}^i) && (5.38)\\
\overline{q}_{sw}^i, \overline{T}_{sw}^i &= f(R, \overline{T}_s^i) \implies (\text{calc. } \tilde{T}_{sb}) && (5.39)\\
\overline{\vec{x}}^i, \overline{T}_b^i, \overline{T}_{sink}^i, \overline{h}^i &= \mathbf{S_b}(\overline{q}_{sw}^i, \overline{T}_{sw}^i), && (5.40)\\
\overline{T}_{fw}^{i-1}, \overline{T}_1^i &= f(\overline{T}_{sink}^i, \overline{h}^i) \implies (\text{calc. } \overline{T}_{sink}) && (5.41)\\
\overline{T}_s^{i-1} &= \mathbf{F_b}(\overline{T}_{fw}^{i-1}, \overline{T}_1^{i-1}). && (5.42)
\end{aligned}$$

The solid solver then returns an adjoint sink temperature $\overline{T}_{sink}$ and adjoint virtual heat transfer coefficient $\overline{h}$. The differentiated Robin preprocessing step uses the virtual heat transfer coefficient $\overline{h}$ to calculate the adjoint wall temperature $\overline{T}_{fw}$ while the adjoint sink temperature $\overline{T}_{sink}$ is assigned to the first off wall node, as shown in Equation (5.41).

Tables 5.6 and 5.7 show good agreement between the gradients of $\frac{dT_w}{dT_b}$ and $\frac{dT_w}{d\vec{x}}$ obtained using the tangent, adjoint, and central difference methods.

Gradient method	Design variable		
	1 [E-04]	2 [E-02]	3 [E-04]
Tangent	2.57621104676445**55**	1.092538001954116**1**	6.26993861582025**40**
Adjoint	2.57621104676445**82**	1.092538001954116**4**	6.26993861582025**84**
CD [$\Delta = 10^{-4}$ K]	2.576**1096367205027**	1.09253**47169177257**	6.2**007757151150145**

Table 5.6: hFRB gradient of temperature T_w w.r.t bottom temperature T_b.

Gradient method	Design variable		
	x	y	z
Tangent	12.7768341**18125210**	-10.5235973**06166273**	0.3238394528**1109172**
Adjoint	12.7768341**08125197**	-10.5235973**14289930**	0.323839452**64861863**
CD [$\Delta = 10^{-7}$m]	12.7**60587537741230**	-10.5**09541539249767**	0.323**02864383382257**

Table 5.7: hFRB gradient of temperature T_w w.r.t coordinates.

5.7 Convergence of differentiated coupling

All the gradients obtained in Tables 5.2 - 5.7 were obtained by fully converging the fluid and solid primal and adjoint equations and looping backwards for the same number of

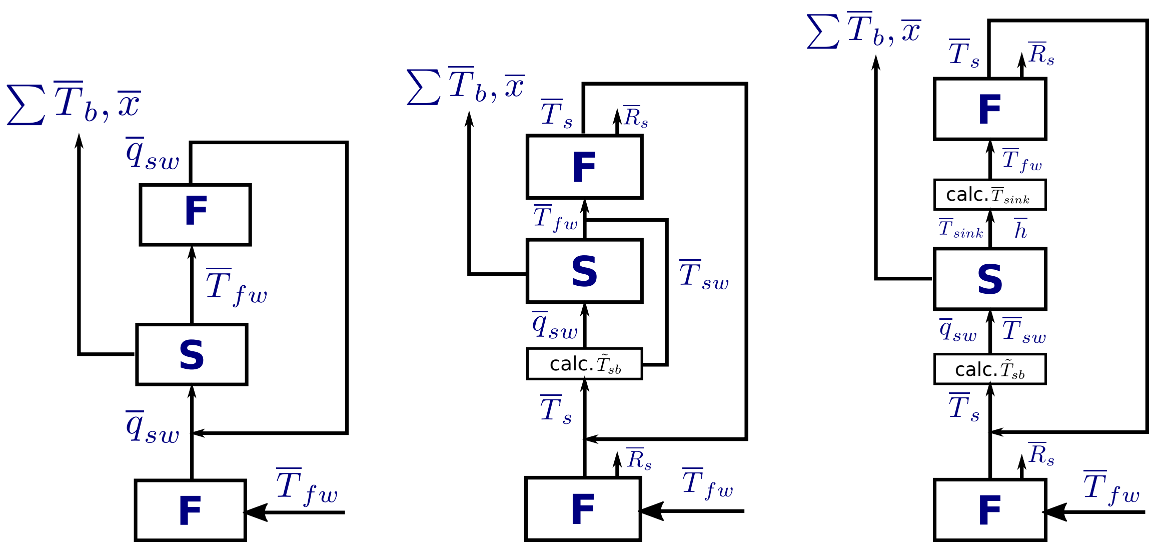


Figure 5.2: Reverse differentiated CHT coupling algorithms; Left:TFFB, Centre: TFRB, Right:hFRB.

iterations as the primal loop. However, significant time savings can be obtained by only partially converging the fluid adjoint equations each time without significantly impacting the accuracy of the gradient. Rather than looping backwards for the same number of iterations as the primal, a fewer number of reverse iterations could be done, at the cost of a slight reduction in gradient accuracy. This is demonstrated with the hFRB coupling algorithm on the same flat plate coupled problem and the results are shown in Table 5.8

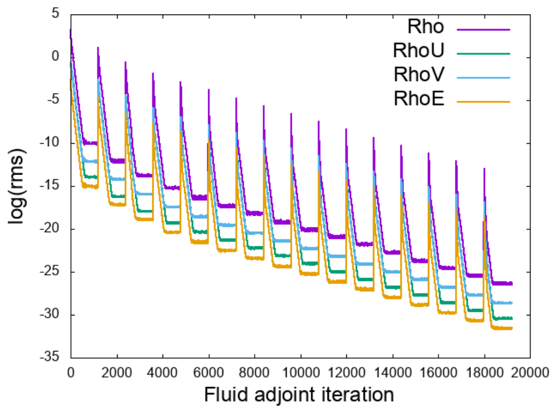
Res_T criteria	No. primal its	No. adjoint its	Fully converged fluid adjoint	$\frac{dT_w}{dT_b}$ [E-04]	$\frac{dT_w}{dx}$
-12	19	19	✓	2.5931768445246677	12.803442391969877
		3	✓	2.5932320190671476	12.803770636795289
		19	✗	2.5927960839112308	12.801561971825787
		3	✗	2.5928512561382180	12.801890193911879
-4	8	8	✓	2.5804449469263107	12.740490830418784
		3	✓	2.5805001251356501	12.740819089239759
		8	✗	2.5800736517173864	12.738657080036727
		3	✗	2.5801288188754076	12.738985273671066

Table 5.8: Adjoint gradients for different coupling convergence criteria.

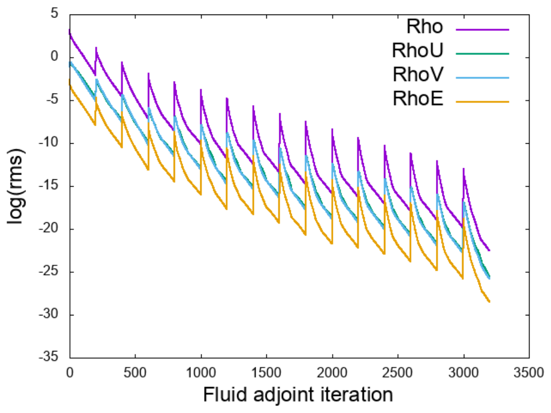
The first column of Table 5.8 refers to the stopping criteria of the coupling based on the coupling residual equation (Equation (3.5)) repeated here for convenience

$$Res_T = \log_{10}\left(\sqrt{\frac{1}{N}\sum_{j=1}^{N}(T_j^i - T_j^{i-1})^2}\right).$$

Res_T is used to quantify the difference in the interface temperature between successive coupling iterations. A Res_T criteria of -12 means the primal coupling was terminated as soon the value of Res_T was less than -12. The second column of the table refers to the number of primal coupling iterations required to meet the stopping criteria in column 1. The third column of the table refers to the number of reverse coupling iterations performed. The fourth column refers to if the fluid adjoint equations were solved to full convergence or not. For partially converged fluid adjoint (represented by crosses), the adjoint equations are converged for a residual drop of approximately 5 orders of magnitude between successive reverse coupling iterations (see Figure 5.3). Columns five and six are the values of the obtained gradients.



(a) Full convergence of fluid adjoint (✓).



(b) Partial convergence of fluid adjoint (✗).

Figure 5.3: Convergence level of fluid adjoint equations between coupling iterations.

The results from Table 5.8 show only minor changes in the value of the gradient even when the fluid adjoint equations are only partially converged. For example, there is only about a 0.5% difference between the first and final values of $\frac{dT_w}{dx}$ despite a difference of 16 reverse coupling iterations and the partial convergence of the fluid adjoint. These results suggest that the number of reverse iterations needed to obtain reasonably accurate gradients does not have to be equal to the number of primal iterations. Consequently, it is possible to reduce the amount of wall clock time taken to obtain the required gradients. However, for optimisation problems, the gradient of the objective function w.r.t design variables could be more sensitive to the convergence criteria and number of reverse iterations. Therefore more investigation is required if this is to be done during numerical optimisation.

5.8 Summary

This chapter discussed how to combine two adjoint solvers to obtain accurate gradients using the adjoint method. The derivation of the adjoint equations for multidisciplinary problems was shown. Furthermore, three coupling algorithms were differentiated and the adjoint gradients were compared to the tangent linear and central difference gradients. It was shown that the adjoint gradients were accurate and have the added advantage of being far less computationally expensive to obtain than the tangent linear and central difference gradients.

The three coupling algorithms considered (i.e. TFFB, TFRB, hFRB) show how gradients are exchanged for Dirichlet, Neumann, and Robin boundaries. Therefore, the presented framework can be extended to other coupling algorithms derived through a combination of any of these three boundary conditions. This represents a generic framework which can be used to obtain gradients for MDO and can then be extended to other coupling algorithms and disciplines.

This enables the use of fully coupled partitioned methods to perform Multidisciplinary Design Optimisation (MDO) which will lead to more robust optimisation results. This is because the inter-disciplinary interaction, which may be lost in sequential single discipline optimisation, are captured and the gradients of these interactions will be used in the optimisation process. Consequently, the presented framework for calculating gradients can be combined with gradient based optimisation algorithms and used for MDO.

Finally, it was seen that partial convergence of the fluid adjoint equations did not lead to significant changes in gradient values and the number of reverse coupling iterations done could be reduced. However, this will be further investigated in the following chapter.

Chapter 6

Numerical optimisation

In this chapter, several CHT optimisation problems are solved. As discussed previously, CHT is a multidisciplinary problem and Multidisciplinary Design Analysis (MDA) is required to assess the performance of a design. The performance of a design is measured quantitatively by an objective function (J) and accurate methods of solving CHT problems are required to obtain accurate values of the objective function. This thesis is focused on the use of numerical optimisation to solve CHT optimisation problems and gradient based optimisation is preferred as fewer objective function evaluation are required to find the optimum. This is particularly essential for multidisciplinary problems as the cost of evaluating the objective function can be high. Furthermore, the adjoint method is used to reduce the computational cost of obtaining the required gradients.

In Chapter 2, methods for performing MDA were discussed and these methods were successfully applied in Chapter 3 to solve three CHT problems. The results of Chapter 3 show that we have the required numerical solvers and algorithms to accurately compute objective functions (J). In Chapters 4 and 5, a sensitivity analysis framework (based on the discrete adjoint via Automatic Differentiation) was developed and verified. In this chapter, the gradients $\left(\frac{dJ}{d\alpha}\right)$ obtained from the differentiated coupling algorithms are used in conjunction with standard gradient based optimisation algorithms and applied to CHT optimisation problems of interest.

6.1 Optimisation algorithms

Given an initial design which does not satisfy certain design or performance criteria, optimisation algorithms can be used to produce new designs which satisfy the requirements or improve the performance of the initial design. The optimiser does this by altering the design variables α to produce a new design. For example, in shape optimisation problems, the design variables could be the shape of an aerofoil and new designs are obtained by changing the shape of the aerofoil to reduce the drag or increase the lift.

Optimisation algorithms can grouped into gradient based or gradient free methods. Gradient based optimisation algorithms obtain new designs by using the slope of the objective function (i.e. gradient w.r.t design variables) as a search direction and moving in the direction of the negative slope in order to reduce the value of the objective. This is done through updating the design variables based on the gradient information. The two optimisation algorithms used in this work are the Steepest Descent and BFGS algorithms. The steepest descent algorithm updates the design variables in the direction of the steepest slope. That is

$$\alpha_{i+1} = \alpha_i - s \cdot \frac{dJ}{d\alpha}, \tag{6.1}$$

where s is the step size chosen. The gradient ($\frac{dJ}{d\alpha}$) provides information on which direction leads to a reduction in the objective function J, while the step size (s) controls the step taken along the descent direction. The steepest descent method is known to converge slowly, however, the choice of step size gives the user control over the amount of change per optimisation iteration which is sometimes desirable.

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm uses an approximation of the Hessian (curvature) which leads to much faster convergence to the optimum than the steepest descent. This comes at the price of greater complexity and the method needs the step size to be safeguarded to avoid divergence. The complexity of the method means that rather than user implementation, external libraries are often used as a black box. This means the user has little control over the step size of the algorithm which sometimes leads to large changes in the design variables which is often a problem in shape optimisation problems. This is because large shape changes lead to large perturbation in the volume mesh which could reduce mesh quality and affect the convergence of the flow solver.

6.2 Mesh deformation

For shape optimisation problems, the change in geometry requires an updated computational mesh which mirrors the geometry perturbation. This can be done by re-meshing the domains but an automatic mesh generator would be required. Furthermore, re-meshing results in topology changes which alter the truncation error in the calculated gradients and could lead to poor performance of gradient based optimisation algorithms.

Alternatively, a mesh deformation algorithm can be use to propagate the displacement of the surface of the new geometry to the interior domain nodes thereby leaving the mesh topology unchanged. Consequently, a mesh deformation algorithm is used rather than re-meshing the solid domain. The mesh deformation algorithm used is the Inverse Distance Weighted interpolation (IDW). IDW obtains the volumetric displacement field (δX) based on a boundary displacement field (X_b) using an inverse distance weighting function (W). That is,

$$\delta X = \frac{\sum_{b=1}^{N} W(X - X_b)\delta X}{\sum_{b=1}^{N} W(X - X_b)}, \tag{6.2}$$

$$W(X - X_b) = \left(\frac{L_{def}}{||X - X_b||}\right)^a + \left(\frac{\alpha L_{def}}{||X - X_b||}\right)^b, \tag{6.3}$$

where $a = 3$, $b = 5$, $\alpha = 0.1$, and L_{def} is the furthest distance of a any mesh node to the element centroid. The mesh deformation algorithm used was implemented in the flow solver and the algorithm has also been reverse differentiated.

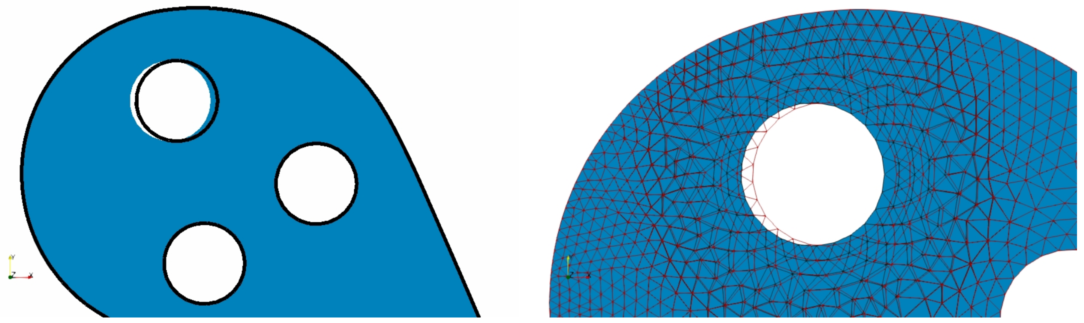


Figure 6.1: Left: Displacement of a cooling channel, Right: Boundary and interior mesh deformation due to displacement (Original grid = black lines, deformed mesh = red lines).

At this point, all the necessary ingredients required to perform numerical optimisation are available.

6.3 Inverse design optimisation

Inverse problems are solved by providing a desired solution and adjusting design variables in order to achieve the desired target. The current inverse problem is related to the flat plate test case described in Section 3.1. In the inverse problem, we seek the bottom temperature (T_b) which results in the best match for a given the interface wall temperature (T_{target}) as shown in Figure 6.2. This simple test case is chosen to evaluate the performance of the differentiated algorithms as the final solution of the problem is known.

Inverse CHT problems are often simplified into either inverse conduction [56] [57] or convection problems [58] [59] [60]. Ahamad and Balaji [61] consider both conduction and convection in their coupled inverse problem however, the problem is solved using artificial neural networks. The use of gradient based optimisation methods to solve fully coupled inverse CHT problems is rare.

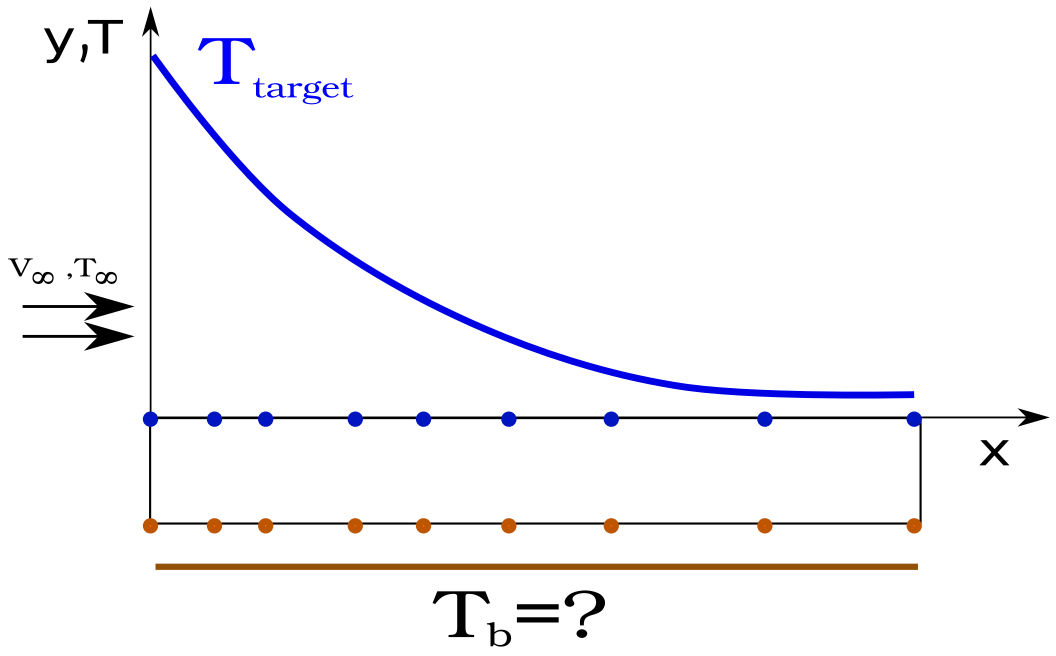


Figure 6.2: Description of flat plate inverse problem

The inverse problem is solved by formulating an optimisation problem, which allows the use of classical direct methods to solve the physics involved. An objective function (J) is defined as the difference between the desired interface temperature (T_{target}) and the obtained interface temperature (T_w) for an estimated bottom temperature $(\tilde{T_b})$.

$$J = \frac{1}{2}\int_0^L (T_{target} - T_w)^2 dx, \tag{6.4}$$

and minimising Equation 6.4 results in an interface temperature which matches the target temperature. The objective function depends implicitly on the bottom temperature T_b through the solution of the coupled problem. Each node at the bottom of the plate has an independent value of T_b specified as a boundary condition, and is used in this work as a design variable (α) that needs to be changed to drive J to zero. As a result, the design variable α in the discrete problem is an array of size N, being 226 in the present work.

The objective is hence to minimise Equation (6.4) subject to the constraints of satisfying both the state equations of both domains and maintain continuity of state variables (T_w, q_w) across the interface.

A gradient based method is used to reduce the deviation of the current interface wall temperature with the desired one. The gradient of the objective function (sensitivity) w.r.t the control variables, α, is given as

$$\begin{aligned} \frac{dJ}{d\alpha} &= \int_0^L (T_{target} - T_w)\frac{dT_w}{d\alpha}dx, \\ &= -\sum_{j=1}^{N}(T_{target} - T_w)\frac{dT_w}{d\alpha}, \\ &= g^T u. \end{aligned} \tag{6.5}$$

The gradients of the temperature w.r.t the design variables, i.e. the temperature specified on the bottom of the flat plate, are computed using the adjoint approach for calculating $\frac{dT_w}{dT_b}$ described in Chapter 5. The target temperature is obtained by solving the primal problem with a bottom temperature of 600K, hence it is guaranteed that a solution to the problem exists. $\tilde{T_b}$ is initially taken as 400K and refers to the estimated bottom temperature that should yield T_{target} .

6.3.1 Gradient verification and Convergence of coupled adjoint

$\frac{dJ}{d\alpha}$ is calculated using the adjoint and central difference methods. The central difference gradient (using the hFRB algorithm) for a step-size of 10^{-4} is calculated as -16.285555575006327. The obtained adjoint gradients for different coupling convergence criteria (Equation (3.5)) are shown in Table 6.1.

Res_T criteria	No. primal its	J	No. adjoint its	Fully converged fluid adjoint	$\frac{dJ}{d\tilde{T}_b}$
-10	14	49574.76672	14	✓	-16.213149771304280
			3	✓	-16.213151850037050
			14	✗	-16.211719505224423
			3	✗	-16.211484618170818
-3	7	49574.62303	7	✓	-16.213151396904671
			3	✓	-16.213153480794599
			7	✗	-16.211484166652358
			3	✗	-16.211486250228454
			1	✗	-16.208558817229790
1	3	48350.78851	3	✓	-15.964738963931770
			1	✓	-15.962456781879856
			3	✗	-15.963124615242574
			1	✗	-15.960842140321112

Table 6.1: Adjoint gradients for different coupling convergence criteria using the hFRB coupling algorithm.

As mentioned in Chapter 5, the number of reverse coupling iterations needed to obtain the adjoint gradients could be less than the number of primal iterations. Looking at the values for a stopping criteria of -3, there is only about a 0.02% difference between 7 reverse iterations with full fluid adjoint convergence and 1 reverse iteration with partial adjoint convergence. This shows that the number of reverse iterations required can be as little as 1 and the fluid adjoint equations do not need to be solved to full convergence.

The results in Table 6.1 also show that the convergence of the value of the objective function is an essential requirement for accurate gradients. The objective function and gradient values for a stopping criteria (Res_T) of -3 and -10 are similar. However, for a stopping criteria of 1, the deviation in the value of the gradients is much larger and is probably due to the incorrect value of the objective function. Consequently, it appears better to formulate the coupling convergence criteria based on the change in the value of the objective function rather than the change in the interface temperature. As a result, the new coupling stopping criteria for numerical optimisation is chosen as

$$Res_J = \log_{10}\Big(abs\Big(\frac{J_i - J_{i-1}}{J_i}\Big)\Big), \tag{6.6}$$

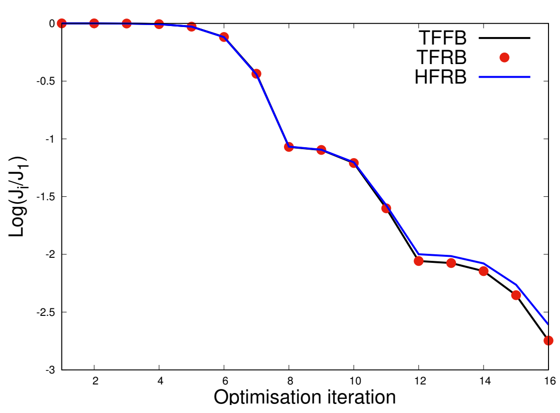
where i is the current coupling iteration. Using the new stopping criteria, it can be observed that the TFRB algorithm also converges in a few reverse iterations as shown in Table 6.2. For the TFFB algorithm however, more adjoint coupling iterations are required for the gradient to converge. This is most likely due to the slower convergence of the algorithm as seen in Figure 3.4a. Consequently, the coupling algorithms using Robin boundary conditions offer significant computational cost savings during both the primal and reverse runs due the the fewer number of coupling iterations required to converge the primal and adjoint coupling.

Res_J	Method	No. primal its	No. adjoint its	Fully converged fluid adjoint	$\frac{dJ}{d\tilde{T}_b}$
-4	hFRB	7	7	✓	-16.213123464117896
			3	✓	-16.213125551733963
			3	✗	-16.211458326021756
			1	✗	-16.208558817229790
-4	TFRB	7	7	✓	-16.276452487010975
			3	✓	-16.276461676406644
			3	✗	-16.274786846455068
			1	✗	-16.275029595997751
-4	TFFB	11	11	✓	-16.497409395306306
			6	✓	-16.501610373037391
			3	✓	-17.062164288946512
			11	✗	-16.486675440154741
			3	✗	-17.050383199509540

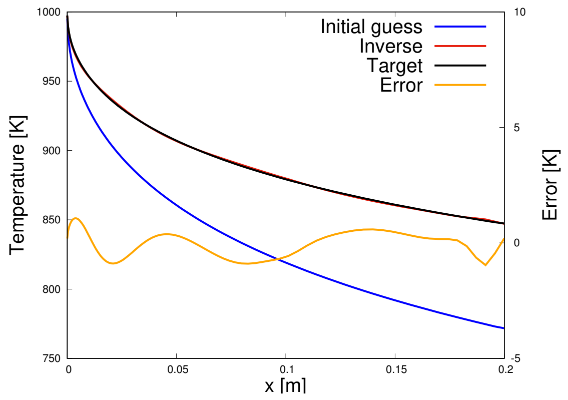
Table 6.2: Adjoint gradients for different coupling algorithms.

6.3.2 Results

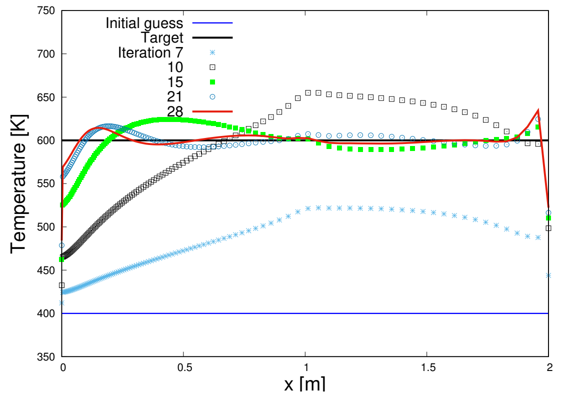
The objective is minimised using the BFGS optimisation algorithm from the SciPy library [62]. The difference between the temperatures obtained from the direct and inverse solution



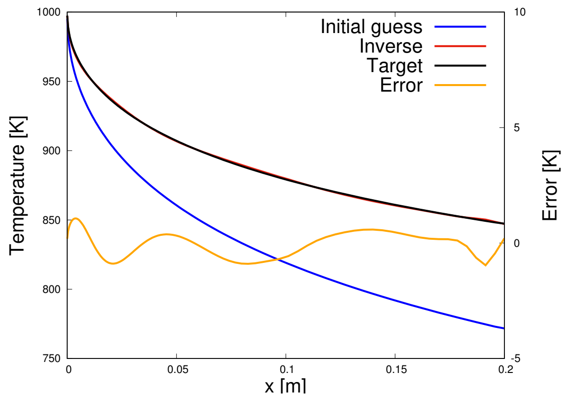
(a) Evolution of the objective function.



(b) T_w distribution after 28 optimisation iterations (TFRB).



(c) Evolution of the bottom temperature (TFRB).



(d) Final T_b distribution (TFRB).

Figure 6.3: Results of the inverse solution after 28 optimisation iterations.

is defined as

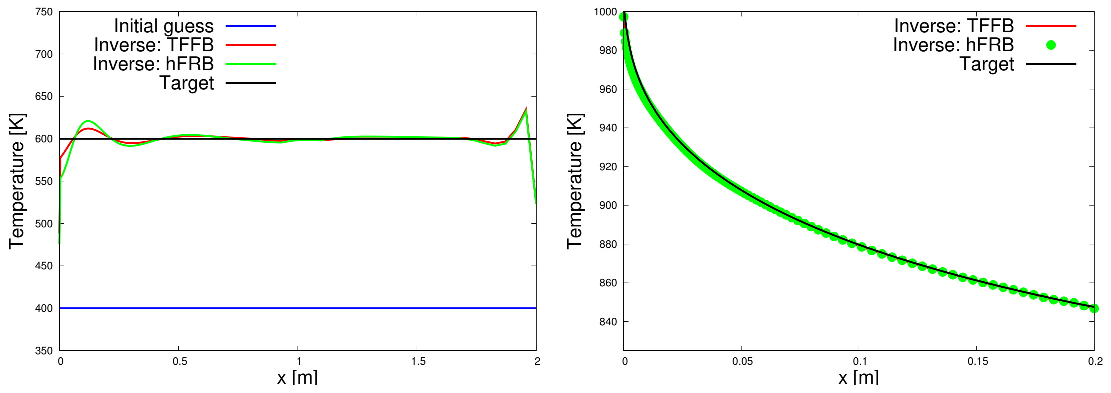
$$\text{Error} = T_{\text{Target}} - T_{\text{Inverse}}. \tag{6.7}$$

The optimisation is solved using all three coupling algorithms and the coupling stopping criteria (Res_J) is set as -4. Only two reverse coupling iterations are done for the hFRB and TFRB algorithms while the TFFB uses the same number of reverse iterations as the primal. For all algorithms, fluid adjoint is converged about 5 orders of magnitude each time. The optimisation is run for 28 optimisation steps and the results obtained are shown in Figure 6.3.

The TFRB optimisation finished in 8.3 hours, while the hFRB took 9.3 hours, and the

TFFB finished in 66.7 hours. The difference in runtimes shows the advantage of using coupling algorithms which use Robin boundary conditions as they require less primal and reverse coupling iterations.

Figure 6.3a shows that the reduction in the objective function for all three coupling algorithms. The results for the TFRB algorithm in Fig. 6.3b show that the inverse solution is significantly closer to the target than the initial guess. Similar results were obtained for the TFFB and hFRB coupling algorithms (see Figure 6.4). The successful solution of the inverse problem shows that gradient computation procedure described Chapter 5 can be successfully utilised by optimisation algorithms to solve multidisciplinary design optimisation problems.



(a) Final T_b distribution (TFFB and hFRB). (b) Final T_w distribution. (TFFB and hFRB).

Figure 6.4: Results of the inverse solution for TFFB and hFRB algorithms.

6.4 Thermal load optimisation

The turbine inlet temperatures of modern gas turbines often exceed the maximum permissible temperature of turbine blade materials. One approach to address the overheating problem is to provide internal cooling to preserve the integrity of the blade. However, this causes large temperature gradients in the blade and may adversely affect the blade lifetime due to the thermal stresses generated [63].

The overheating and thermal fatigue caused by the combination of high inlet temperatures and turbine cooling is a common cause of turbine blade failure as seen in Figure 6.5 [64] [65]. Hence, the blade lifetime could be extended by reducing the thermal gradients in

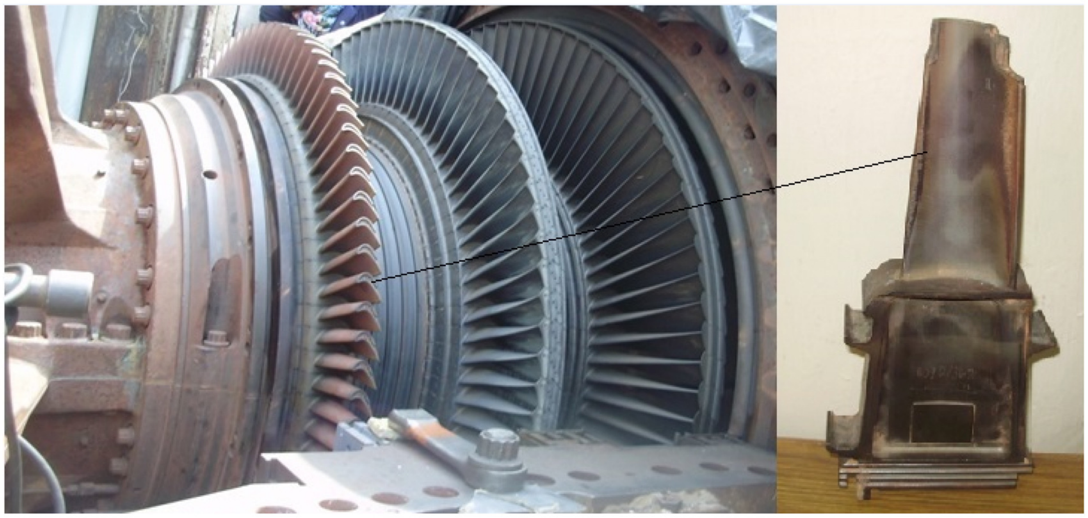


Figure 6.5: Thermally damaged turbine blade [64].

the blade. Consequently, an objective function is defined to represent this effect.

The thermal objective function (J) is defined as the weighted sum of the temperature and temperature gradient in the solid domain.

$$J = c_1 \cdot I_1 + c_2 \cdot I_2, \tag{6.8}$$

where c_1 and c_2 are weight factors ($c_1 + c_2 = 1$) and the sum of the temperature in the solid domain is defined as

$$I_1 = \frac{1}{2}\sqrt{\frac{1}{\Omega_s}\sum_{j=1}^{N} T_j^2}, \tag{6.9}$$

while the non-uniformity of the temperature field is defined as

$$I_2 = \frac{1}{2}\sum_{j=1}^{N}(\nabla T_j)^2, \tag{6.10}$$

where N is the number of nodes in the solid domain. The term I_1 serves to avoid designs which produce a uniform high magnitude temperature field (i.e. low temperature gradient, high temperature), while the I_2 term serves to reduce the temperature gradient and consequently the thermal stress. Both I_1 and I_2 are normalised using the values from the initial design.

The objective function is implemented in CalculiX and is differentiated w.r.t. the coordinates and CHT boundary conditions in CalculiX. The objective function depends implicitly on the solid coordinates because a coordinate perturbation ($\delta\vec{x}$) results in a volume change ($\delta\Omega_s$). Therefore, the gradient of the objective w.r.t coordinates is evaluated as

$$\frac{dJ}{d\vec{x}} = \frac{\partial J}{\partial \vec{x}} + \frac{\partial J}{\partial T}\frac{dT}{d\vec{x}}, \tag{6.11}$$

where $\vec{x}$ represents the cartesian coordinates x, y, z and $\frac{dT}{d\vec{x}}$ can be obtained using the differentiated coupling algorithms described in Chapter 5.

6.4.1 Gradient verification and Convergence of coupled adjoint

To verify the accuracy of the adjoint gradients obtained for Equation (6.11), the flat plate test case in Section 3.1 is used. The weighting constants $c_1 \& c_2$ are taken as 0.5 and the coupled solution stopping criteria (Res_J) is set as -7. The node at the bottom left corner of the plate is used as the design variable which is perturbed to obtain central difference gradients (see Figure 6.6). The adjoint hFRB procedure is also shown in Figure 6.6.

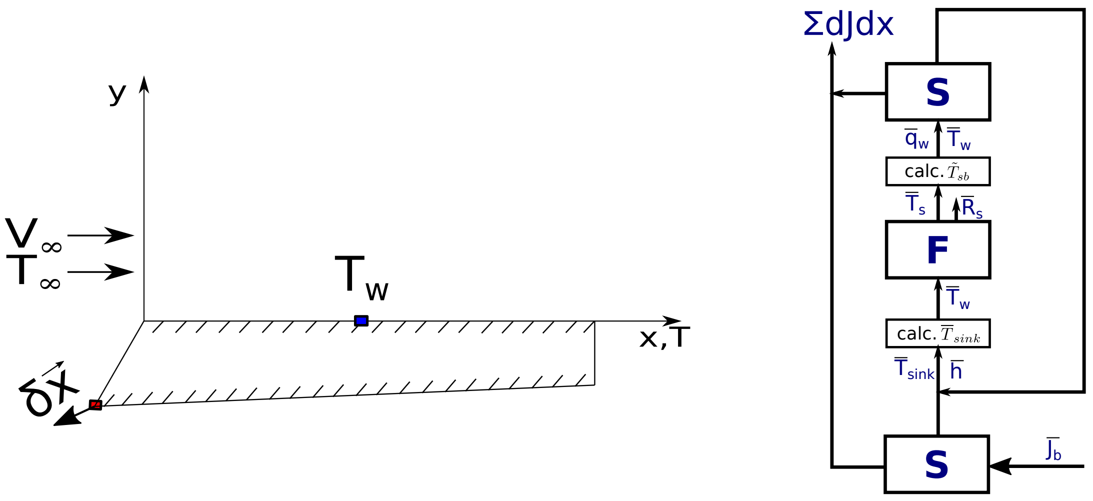


Figure 6.6: Left: Perturbation of the design variable, Right: hFRB adjoint procedure.

Table 6.3 shows good agreement between the adjoint and central difference gradients. Similar to the results obtained in Table 6.2, the TFFB algorithm requires more reverse coupling iterations than the TFRB and hFRB algorithms to converge to an accurate gradient value.

Res_J	Method	Central Diff $[\Delta_y = 10^{-8}\text{m}]$	No. adjoint its	Fully converged fluid adjoint	$\frac{dJ}{dy}$
-7	hFRB	2.763951245654539	8	✓	2.7640087476819333
			3	✓	2.7640086766814211
			3	✗	2.7640116201616292
			1	✗	2.7608959386278791
-7	TFRB	2.303537188774740	8	✓	2.3036154270081126
			3	✓	2.3036150655530117
			3	✗	2.3036189193159049
			1	✗	2.3035295370861868
-7	TFFB	2.303536139613982	13	✓	2.3034514495830405
			3	✓	2.3016725906081996
			13	✗	2.3034768792365168
			3	✗	2.3017001159515513

Table 6.3: Adjoint gradients for different coupling algorithms.

6.4.2 Mark II turbine blade optimisation

An optimisation problem is formulated to minimise the thermal objective function (Equation (6.8)) by changing only the location of each cooling channel in the MarkII turbine blade. Similar work has been done by Ferlauto [15] where an inverse design optimisation of the MarkII was performed using channel coordinates and radii as design variables. However, only the conduction in the solid was considered and the flow equations were ignored. Wang et al. [66] minimise the maximum temperature in the C3X vane by changing the shape and location of the cooling channels, however the gradients are calculated using the finite-differences. In [14] the entropy generation of an internally cooled turbine blade is minimised by changing the shape of the blade wall. Mousavi [12] performs a temperature gradient optimisation while Gkaragkounis [16] minimises the mean temperature in the C3X turbine blade by changing the shape of the blade wall and location of the cooling channels.

However, in [12] , [14], and [16], the continuous adjoint method is used to obtain the gradients and only a combination of Dirichlet and Neumann boundary conditions is used for the coupling between solvers. In the present work, the coupled discrete adjoint method is used to calculate the gradients while the Robin-Robin (hFRB) coupling algorithm is used to obtain the coupled solution. A 2D simulation of the case is performed and the boundary conditions used are shown in Section 3.3

The x, y coordinates of each cooling channel is taken as a design variable (α) while the

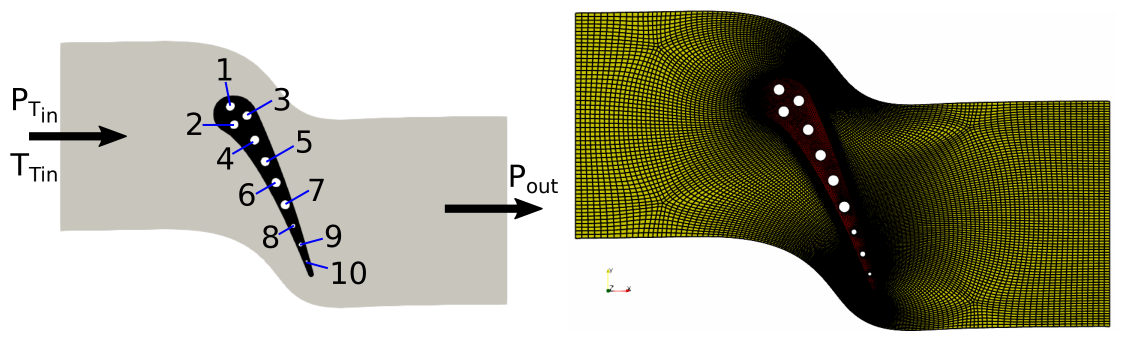


Figure 6.7: MarkII Geometry and mesh.

radii of the channels are kept fixed. As the solver inputs are the grid (x, y) coordinates, the reverse differentiation only gives us the gradient of the cost function w.r.t grid coordinates. However, by summing the $x\&y$ gradients at the grid nodes of a particular cooling channel, we are able to obtain the gradient w.r.t the channel coordinates, and by displacing all the nodes by the same magnitude, the channel radius is kept constant.

Consequently, the number of design variables is equal to the number of channels (10) times two degrees of freedom, leading to 20. The ten cooling channels are numbered as shown in Figure 6.7. The objective function gradient is first verified in the solid domain alone and the weight constants are chosen as $c_1 = c_2 = 0.5$. The central difference and adjoint gradients obtained are shown in Table 6.4

α	Adjoint	CD [$\Delta = 10^{-7}$m]
x	-4.027759**607145722**	-4.027759**575864209**
y	3.453533**744189160**	3.453533**725661373**

Table 6.4: Gradient of J w.r.t to x, y location of cooling channel 1 (solid domain only).

The coupled adjoint gradients are also compared to central difference gradients with the stopping criteria (Res_J) set as -6.

Table 6.5 shows that the adjoint and central difference gradients match up to 3 decimal places. Therefore, the adjoint gradients can now be used in conjunction with a gradient based optimiser to solve the optimisation problem.

dJ/d	Channel	Adjoint	CD [$\Delta = 10^{-7}$m]
x	1	0.814**184396519775**	0.814**074160859768**
y	1	-0.257**961749813784**	-0.257**498785627419**
x	4	-0.045**413004345721**	-0.045**279308635848**
y	4	-0.1079**93044634841**	-0.1079**42127458571**
x	10	-0.210**540327245528**	-0.210**956274115937**
y	10	-0.114**930459095038**	-0.114**789205363230**

Table 6.5: Coupled gradient of J w.r.t to x, y location of 3 cooling channels

6.4.3 Optimisation results

Three optimisations are run with varying values of c_1 and c_2 as shown in Table 6.6. An unconstrained optimisation is performed and the steepest descent algorithm is used in order to avoid large displacements to the cooling channels between optimisation iterations. For the L^2 temperature norm optimisation (Opt1), a constant step-size of 7E-03 was used while Opt2 and Opt3 were run with a step-size of 1E-03. The coupling stopping criteria is set as $Res_J = -4$ and 3 reverse coupling iterations are used to calculate the gradients.

	Opt1	Opt2	Opt3
c1	1.0	0.0	0.5
c2	0.0	1.0	0.5

Table 6.6: Values of weights for each optimisation run.

Figure 6.8 shows the displacement of the cooling channels. The black lines represent the channels of the optimised blades while the solid surface is the initial geometry. For the L^2 temperature norm optimisation (Opt1), the channels at the leading edge are displaced upwards towards the high temperature regions while the trailing edge channels are displaced towards the suction surface (x/c > 0).

Opt2 is the exact opposite of Opt1 with the leading edge channels being displaced downwards towards the blade interior while the trailing edge channels are displaced towards the pressure side of the blade (x/c < 0). Opt3 is a less amplified version of Opt2.

The optimisations are stopped prematurely due to excessive mesh deformation. The final values for the objective function are shown Figure 6.9 and Table 6.7. Figure 6.10 shows the change in temperature distribution between the initial and optimised blades, where ΔT is the difference between the optimised and baseline temperature.

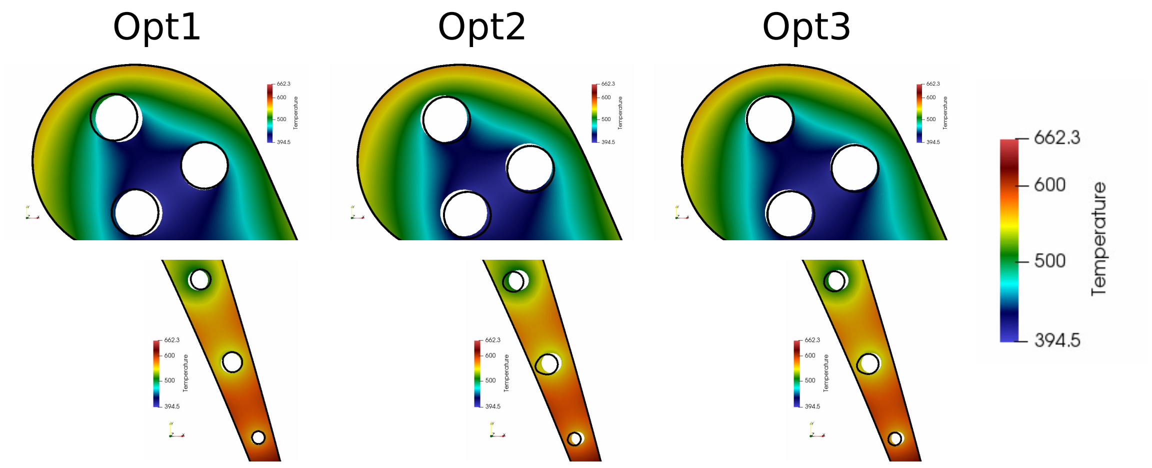
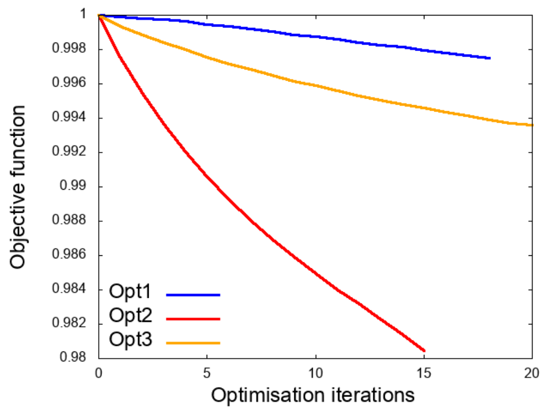


Figure 6.8: Channel displacements at the leading and trailing edges.



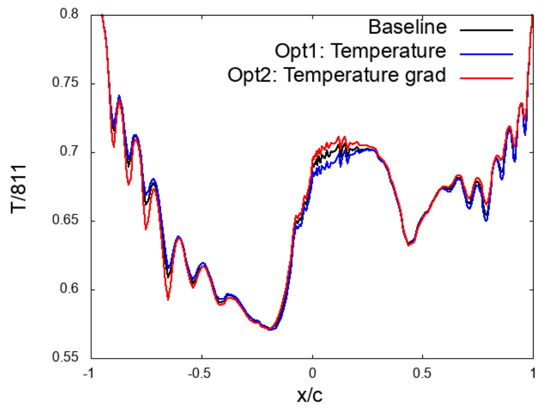
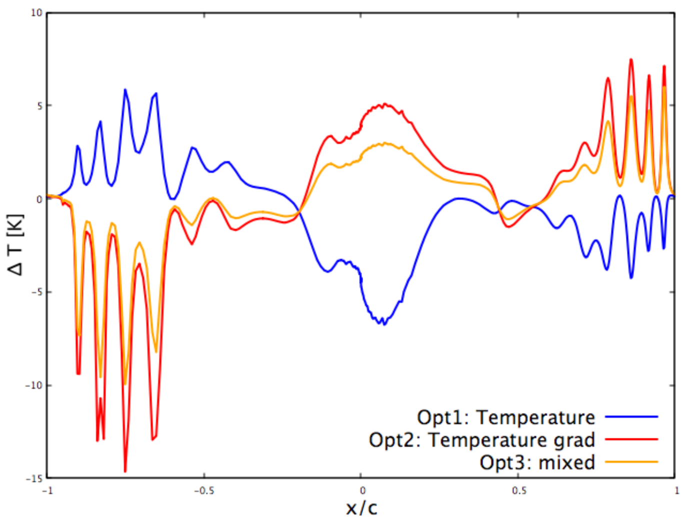


Figure 6.9: Cost function evolution (right) and interface temperature distribution (left).

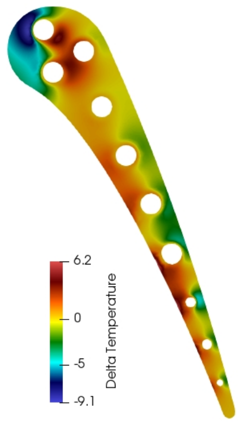
	Opt1	Opt2	Opt3
No. Opt iterations	18	15	20
$I_1/I_{1,ref}$	0.99752	1.00276	1.00180
$I_2/I_{2,ref}$	1.01373	0.99813	0.98536

Table 6.7: Final Values of I_1 and I_2 after optimisation.

For the L^2 temperature norm optimisation, the temperature around the leading edge and the suction side of the blade are reduced. This is evident in Figure 6.10a which shows negative values for ΔT over a significant portion of the blade perimeter. Figure 6.10b also shows how the internal temperature is reduced in the direction of the channel displacement.



(a) Change in interface temperature distribution.



(b) Opt1 change in internal temperature.

Figure 6.10: Comparison of initial and optimised temperature distribution.

The greatest reduction in temperature is seen around the leading edge of the blade (blue region in Figure 6.10b).

Conversely, for Opt2 and Opt3, the temperature is reduced on the pressure side and increased on the suction side and the results for Opt1 are inverted.

6.5 Summary

In this Chapter, the developed coupled discrete adjoint framework has been used in conjunction with optimisation algorithms to solve multidisciplinary optimisation problems. It was seen that the convergence of the objective function is required for accurate gradients. It was also seen that coupling algorithms using Robin boundary conditions require less reverse coupling iterations to obtain reasonably accurate gradients.

All three differentiated coupling algorithms were successfully used to solve an inverse problem. The TFRB and hFRB algorithms required less wall clock time than the TFFB algorithm to solve the inverse problem. The time savings can be attributed to the Robin-based algorithms requiring less primal and reverse coupling iterations. This shows that using Robin boundary conditions in the fluid domain can speed up the optimisation process.

A thermal objective function was also introduced and used in a turbine blade optimisation problem. The thermal objective function is a weighted sum of the solid temperature and temperature gradient. Three optimisation runs were performed using different weighting values for the objective function. For all three optimisations, there was a 0.2-2% decrease in the objective function. In the following chapter, a more challenging 3D optimisation using the thermal objective function is performed.

Chapter 7

Internal cooling channel optimisation

Modern gas turbines are equipped with internal cooling channels which cool the internal structure and provide cooling air for other forms of cooling such as film cooling as shown in Figure 7.1. A prominent feature at the ends of the channels is the presence of a U-Bend. The U-Bend directs the flow through a 180 degree turn which leads to significant pressure losses in the internal cooling scheme and results in a lower cycle efficiency of the gas turbine plant. Consequently, the majority of the U-Bend shape optimisations in literature have focused on the pressure loss optimisation of the U-Bend section [67] [68] [69] [70].

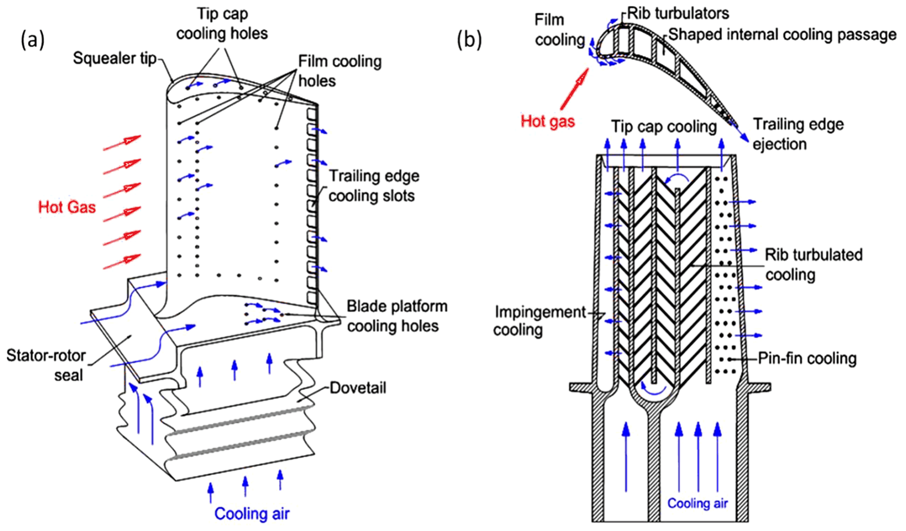


Figure 7.1: External turbine cooling and gas path through internal cooling channels [71].

While the pressure loss optimisation is an important and interesting problem, it is a single disciplinary optimisation problem which does not take into account the effect of shape changes in the bend on the temperature and heat transfer in the both domains. As the primary aim of the internal channels is to provide cooling for the blades and preserve structural integrity, the effect of the channel flow on the solid also needs to be considered in an optimal design.

He et al. [17] perform an adjoint based aerothermal optimisation using weighted sums of the pressure loss coefficient (C_{PL}) and the Nusselt number (Nu) as the objective function. The wall temperature was fixed at 10 K higher than the inlet temperature, and the partial derivatives for the adjoint equations were approximated using finite differences. The results showed that an increased weighting for C_{PL} lead to an expansion of the duct which decreased the flow velocity and friction losses. On the other hand, an increased weighting for Nu lead to a shrinking of the duct which increased the flow velocity and convective heat transfer.

Previous optimisation studies on the U-Bend have not considered the conduction or deformation in the solid domain. Optimisations investigating the solid temperature or thermal gradients generated as a result of the cooling are lacking and the present works aims to address this. In this chapter, a multidisciplinary CHT optimisation of thermal objective function (Equation (6.8)) is performed. A partitioned coupling approach is used to determine the interface temperature and the adjoint equations are solved using analytically calculated derivatives.

The geometry of the U-Bend used in this work is based on the experimental study of Colleti et al. [72], however, the length of the inlet and outlet sections are shortened as shown in Figure 7.2.

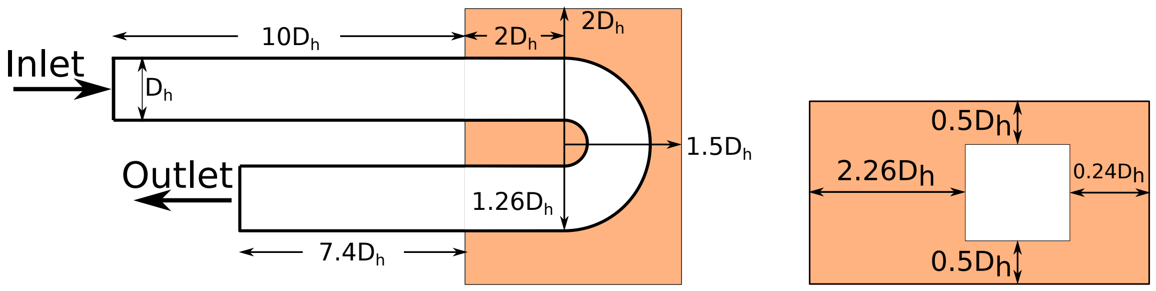


Figure 7.2: Left: Top cross sectional view of the U-Bend, Right: Side cross sectional view.

The design area is bounded by the solid domain (coloured peach) and shape changes are allowed to only the fluid-solid interface.

7.1 Boundary conditions and Discretisation

The experimental study was conducted at a Reynolds number of 43830 and with a bulk velocity of 8.8m/s. The simulation of the fluid flow with the experimental conditions is challenging due to a combination of low inlet velocity, flow separation, secondary flow motion, and streamwise curvature. This combination of features result in poor convergence of the compressible flow solver due to the large difference in acoustic and convective eigenvalues in low Mach number flows. This problem can be alleviated by having the Mach number in the range 0.1-0.3 which ensures that compressibility effects remain negligible [32]. However, in order to match the experimental results, the Reynolds number needs to be kept constant. Consequently, the inlet velocity for the CFD run needs to be increased while the Reynolds number remains constant. In the present work, higher inlet velocity is achieved by adjusting the ratio between the inlet total pressure and outlet static pressure. The isentropic equations for total pressure (P_T) and total temperature (T_T) ratios are given by

$$\frac{P_T}{P_s} = \Big(1 + \frac{\gamma - 1}{2} M^2\Big)^{\frac{\gamma}{\gamma-1}}, \tag{7.1}$$

$$\frac{T_T}{T_s} = 1 + \Big(\frac{\gamma - 1}{2} M^2\Big), \tag{7.2}$$

where P_s and T_s are static pressure and static temperature respectively, and γ is the heat capacity ratio of air. The inlet total temperature is chosen to match the experimental values in Table 7.1. An inlet Mach number (M_{in}) of 0.1 is chosen and the values of the inlet total pressure and outlet static pressure are adjusted until the experimental Reynolds number is achieved. The inlet and outlet pressure boundary conditions are obtained through the following steps:

1. For $M_{in} = 0.1$, calculate P_s for an initial guess of $P_{T,in}$ using Equation (7.1).
2. With the experimental value for T_T calculate T_s using Equation (7.2).
3. Use the calculated value of P_s and T_s to calculate the density and viscosity using the ideal gas law and Sutherland formula respectively.
4. Calculate the inlet velocity and Reynolds number.

5. If the target Reynolds number is not obtained, return to step 1 and adjust $P_{T,in}$ accordingly.

The final values used for the CFD boundary conditions are shown in Table 7.1.

Experiment		CFD	
Temperature (T)	293.15 K	$T_{T,in}$ [K]	293.19
Pressure (P) [Pa]	1.103×10^5	$P_{T,in}$ [Pa]	27218.2727
Velocity (V) [m/s]	8.8	M_{in}	0.1
Density (ρ) [kgm^{-3}]	1.204	Density (ρ) [kgm^{-3}]	0.32185
Viscosity (μ) [Pa·s]	1.813E-05	Viscosity (μ) [Pa·s]	1.88841E-05
		$P_{S,out}$ [Pa]	27028.5990
D_h [m]	0.075	D_h [m]	0.075 m
Re	43830	Re	43830.00024

Table 7.1: U-Bend fluid boundary conditions.

For the solid domain, the solid material is chosen as stainless steel with a thermal conductivity of 14.2 W/mK, and a density of 7900 kgm^{-3}. The temperature of all outer walls are set as 303 K except the inlet facing wall which is set as adiabatic. A Robin boundary condition is specified at the fluid-solid interface (dashed line in Figure 7.3) with the interface temperature being the result of the coupled solution. The virtual solid conductivity used for the hFRB coupling algorithm is specified as $\tilde{R} = \frac{\lambda_s}{0.01}$.

The fluid domain is discretised with approximately 75,000 nodes while the solid uses approximately 29,000 nodes and matching grids are used between both domains as shown in Figure 7.3.

7.2 Primal results

A 3D simulation is performed using the Spalart-Allmaras turbulence model [29]. The AUSM scheme [34] is used for the convective fluxes while the viscous terms are calculated using an edge-corrected Green-Gauss method. 2 levels of multigrid are generated using Hip [73] and used with JT-KIRK implicit time stepping [30] to accelerate convergence. The hFRB coupling algorithm is used and after 37 coupling iterations the final values of the coupling convergence criteria (Equations (3.5) and (6.6)) are $Res_T = -7.4$ and $Res_J = -10.1$. Figure 7.4 shows that the coupled problem converges slowly. Between the second and final coupling iteration, there is only a Res_T drop of 3.61 and Res_J drop of 3.2.

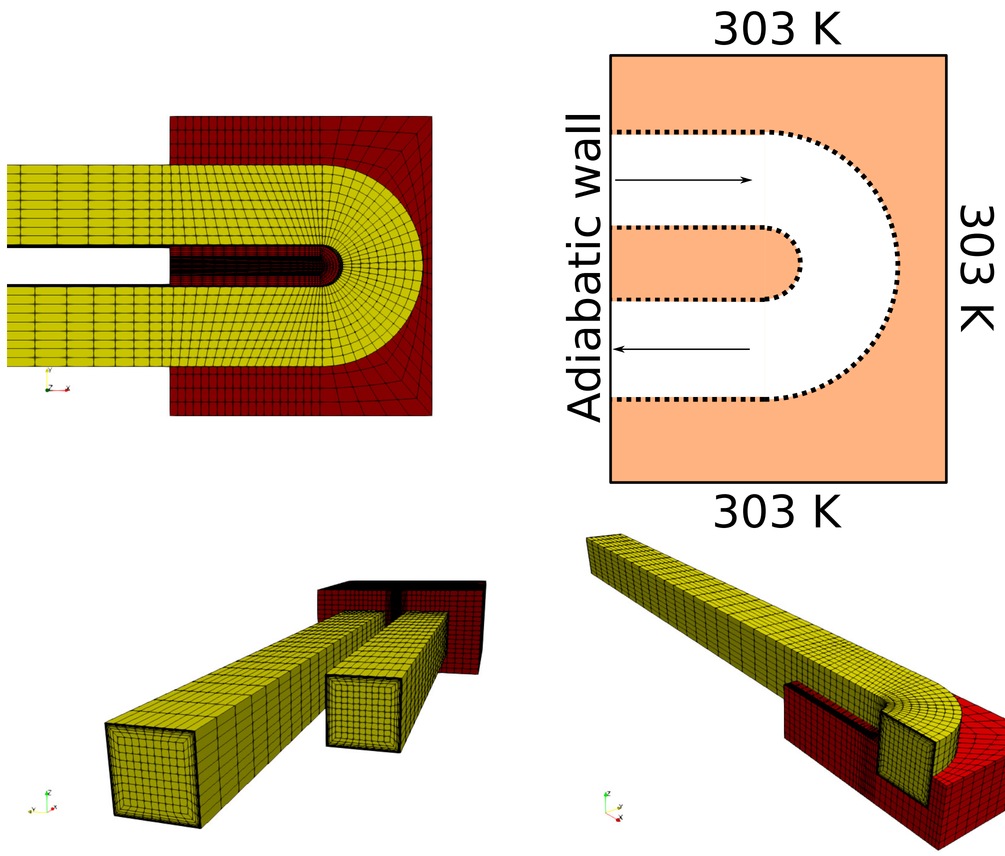


Figure 7.3: Solid boundary temperature and fluid and solid grids.

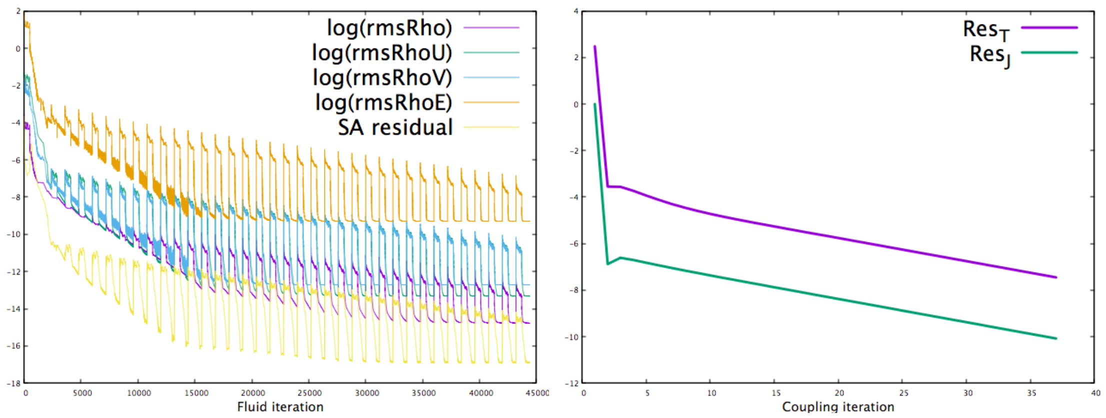


Figure 7.4: Evolution of the fluid residuals and coupling convergence criteria.

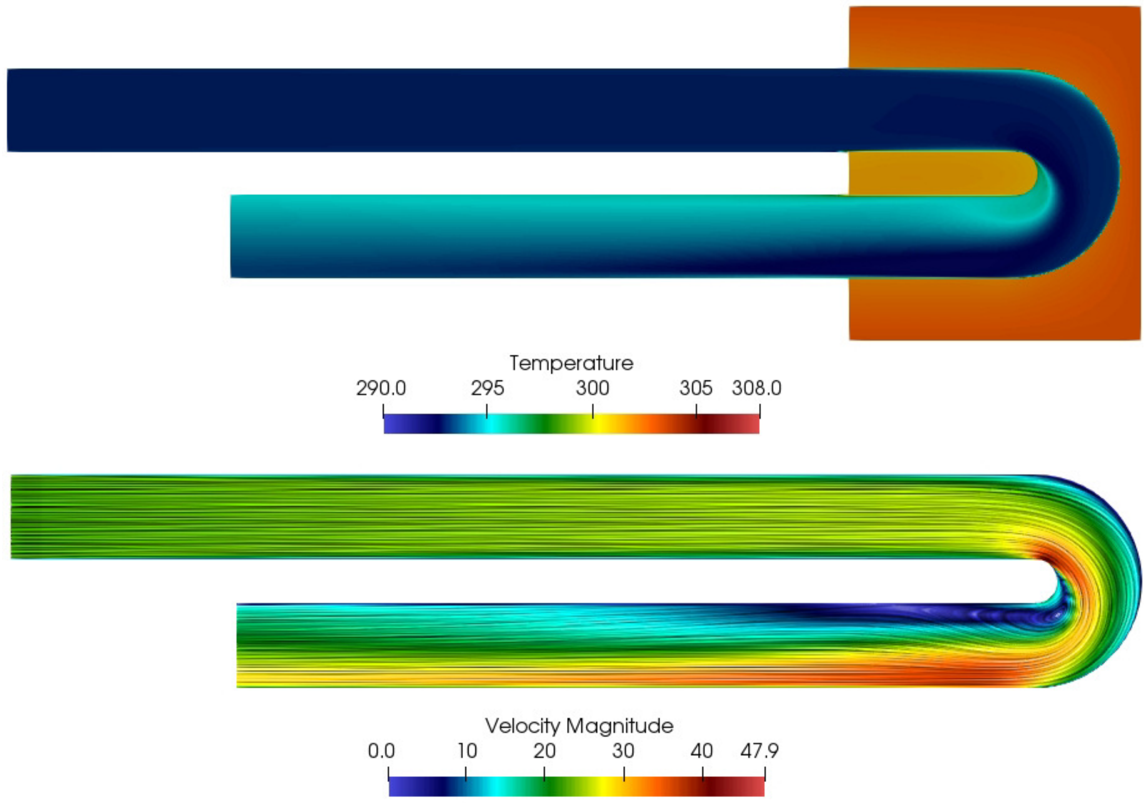


Figure 7.5: Initial temperature (top) and velocity (bottom) on the mid-plane.

The slow convergence is related to the choice of the virtual solid conductivity $\tilde{R}$. Smaller values of $\tilde{R}$ which could have sped up the coupling convergence lead to instabilities in the high velocity region of the inlet leg of the channel (see Figure 7.5).

Figure 7.5 shows the initial temperature and velocity magnitude field. A qualitative comparison with the experiment shows that the current mesh is unable to accurately capture the flow separation and reattachment. However, due to the high runtimes and slow convergence, it is computationally expensive to further discretise the flow. The use of a parallelised flow solver or Quasi-Newton coupling algorithm would eliminate this bottleneck.

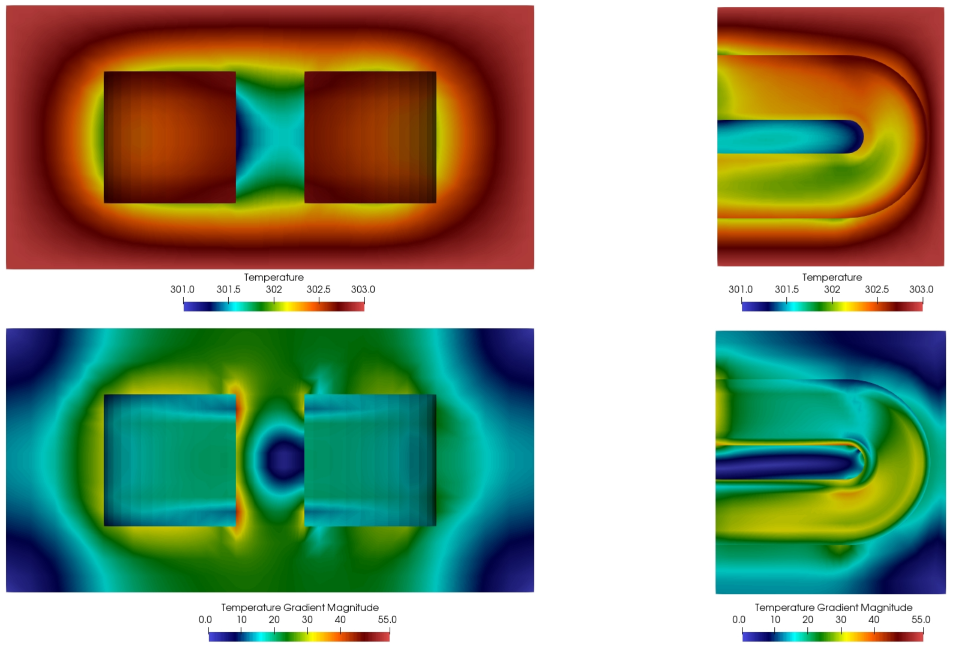


Figure 7.6: Initial temperature (top) and temperature gradient (bottom) field in the solid.

7.3 Parameterisation

The design variables (α) used for the MarkII shape optimisation were the grid coordinates (X) which produced the optimised design as a deformed mesh. Typically, the initial geometry is designed using Computer Aided Design (CAD). CAD models are used for several aspects of the design process such as meshing, numerical analysis, and manufacturing. However, reconstructing the CAD files from the output mesh requires approximations which lead to deterioration of geometric details and reduces the quality of the final design. Consequently, it would be preferable to have a CAD model of the optimised design as this ensures compatibility between several design departments.

Alternatively, the CAD parameters which define the geometry can be used as design variables during the optimisation leading to a CAD model of the optimised design. A CAD based parameterisation makes it is easier to impose geometric constraints. For example, maintaining the channel radius in the MarkII optimisation in the previous chapter.

If the CAD parameters are to be used as design variables for gradient based optimisation, the optimiser requires the gradient of the objective function w.r.t. the CAD parameters. That is

$$\frac{dJ}{\partial\alpha} = \frac{\partial J}{\partial\alpha} + \frac{\partial J}{\partial X}\frac{\partial X}{\partial\alpha}. \tag{7.3}$$

The gradient of the grid coordinates w.r.t. CAD parameters ($\frac{\partial X}{\partial\alpha}$) has to be obtained from the differentiated CAD modelling system. However, most commercial CAD packages lack analytically and efficiently differentiated gradients. Robinson et al. [74] use finite-differences to calculate the CAD sensitivites while Banovic, Mykhaskiv, Auriemma et al. [75] [76] [77] use operator overloading to reverse differentiate the OpenCASCADE CAD kernel. The handicap of the operator overloading approach is that it consumes a lot of memory.

The current work uses the NURBS-based Parameterisation method with Complex Constraints (NSPCC) CAD kernel. The NSPCC kernel is written in Fortran90 and has been revers differentiated using source transformation AD by Jesudasan et al. [78]. NSPCC uses the boundary representation of the CAD model to derive the parameterisation. NSPCC also allows the imposition of geometric constraints between adjacent NURBS patches such as G_0 (no gaps), G_1 for tangency , and G_2 for curvature.

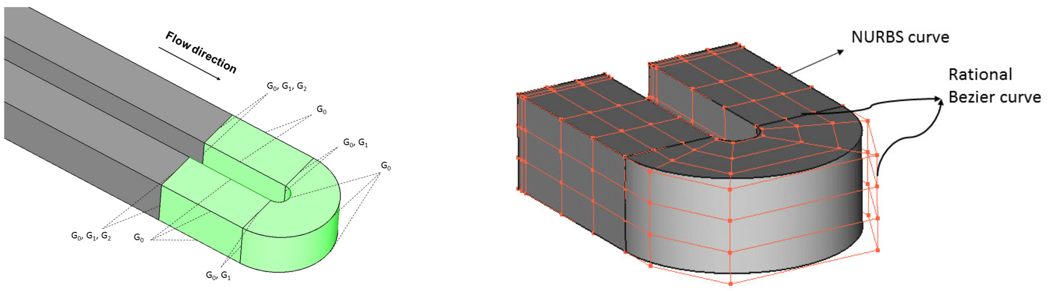


Figure 7.7: Geometric constraints and control net distribution.

The boundary representation of the U-Bend geometry consists of 12 patches of cubic NURBS and rational Bezier curves (see Figure 7.7). Each patch is defined using a 6x4 control net leading total number of 288 control points. The constraint equations are formulated numerically, differentiated with AD, and a constraint Jacobian is computed. The design space (all possible infinitesimal deformations that satisfy the constraints) is in the null space (kernel) of the constraint Jacobian. We compute a basis for this null space (and also for the orthogonal row space) using SVD. The design variables are then not the control points, but the singular vectors from the SVD.

7.4 Optimisation results

The thermal objective function (Equation (6.8)) is used with $c_1 = 1$ and $c_2 = 0$. That is, only the temperature term of the objective which is rewritten here for convenience

$$I_1 = \frac{1}{2}\sqrt{\frac{1}{\Omega_s}\sum_{j=1}^{N} T_j^2}.$$

The coupling stopping criteria used for the optimisation is set as $Res_J = -5$ which results in approximately 6 coupling iterations between optimisation steps. 3 reverse coupling iterations are done to calculate the gradient. The steepest descent optimisation algorithm is used with a fixed step size of 3.E-2 and after five optimisations steps there is a 4.3% reduction in the cost function.

Figure 7.8 shows that the optimiser clearly expands the solid domain and shrinks the fluid in order to reduce the solid temperature. There is a 9% increase in the solid volume and a marked increase in the size of solid region between the upstream and downstream channel.

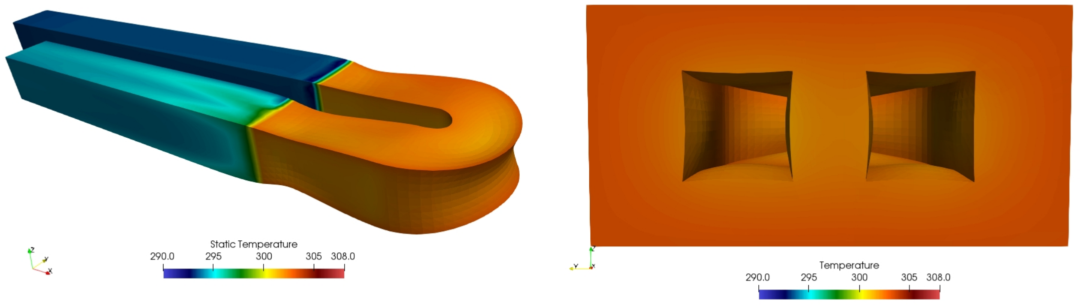


Figure 7.8: Optimised fluid and solid domains.

Figure 7.9 shows a comparison of the temperature distribution between the initial and optimised geometry while Figure 7.10 compares the velocity distribution. Figure 7.10 also shows a change in the position of the high velocity fluid flow. The high velocity region corresponds to lower a lower temperature region in Figure 7.11.This is because the increased fluid velocity results in an increase in the convective heat transfer.

Figure 7.12 shows that the optimiser removes the area of low secondary flow, increases the secondary flow in the channel centre. This will lead to more mixing in the fluid, hence

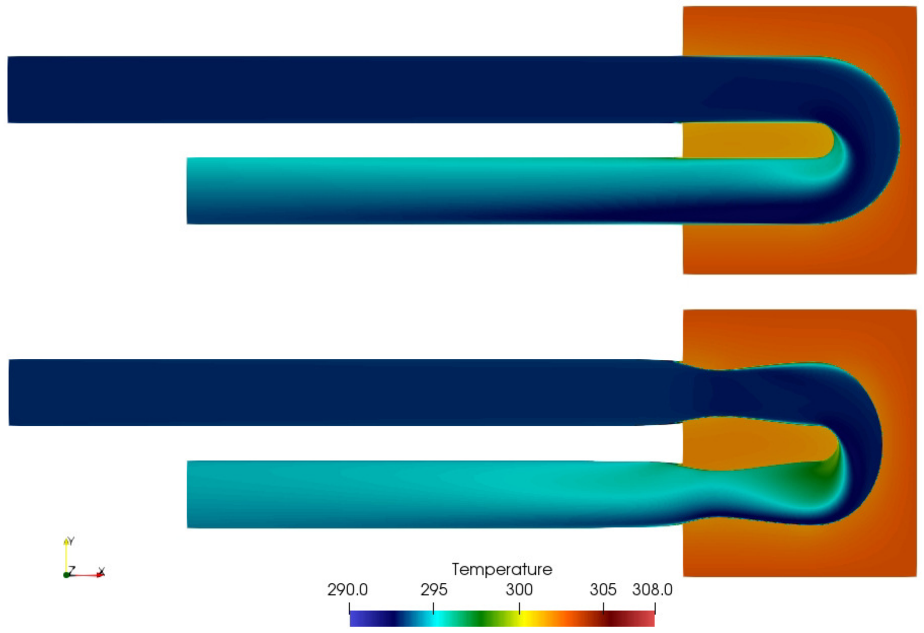


Figure 7.9: Initial vs. optimised temperature on the mid-plane.

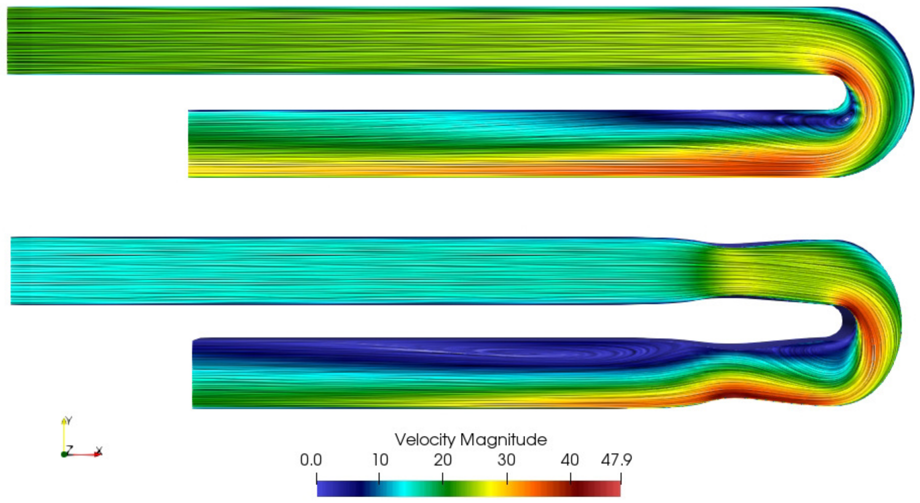


Figure 7.10: Initial vs. optimised velocity on the mid-plane.

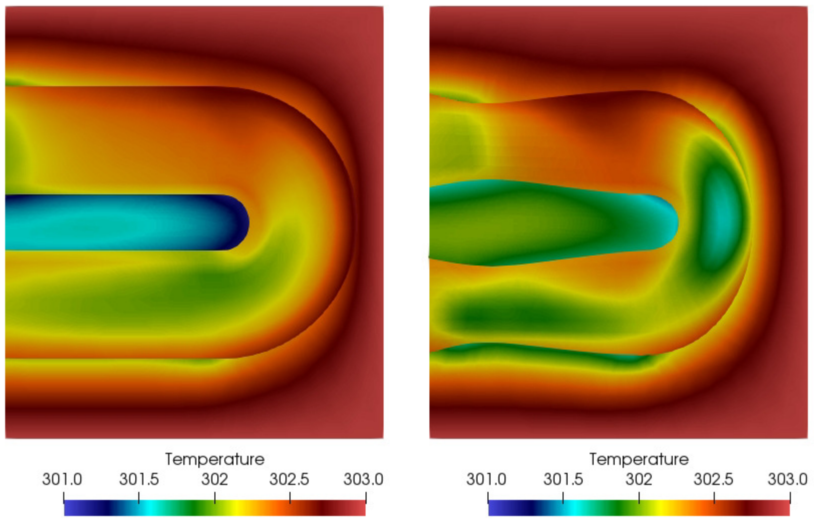


Figure 7.11: Initial vs. optimised solid temperature on the mid-plane.

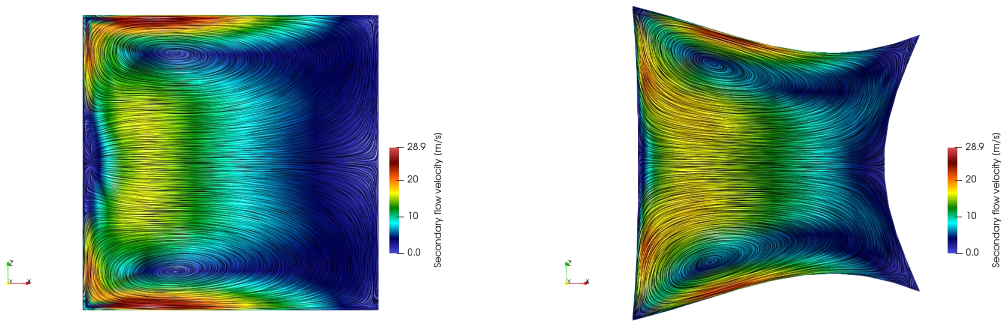


Figure 7.12: Initial vs. optimised velocity tangent to plane normal (at 90° position).

increased heat transfer.

There is a pronounced increase in the flow separation in the downstream channel shown in Figure 7.10. The increased flow separation would result in an undesirable increase in the pressure loss. Consequently, a multi-objective cost function taking the pressure loss into account (similar to the work of He et al. [17]) would be a more appropriate optimisation for this test case.

7.5 Summary

In this chapter, a novel multidisciplinary optimisation was performed on an internal cooling channel U-Bend. For the first time, a solid domain was used to enclose the design area of the U-Bend. The Robin-Robin hFRB coupling method was used to solve coupled problem and it was seen that problem converged slowly. The slow convergence of the coupled solution is caused by a combination of the choice the virtual solid conductivity $\tilde{R}$ and the use of a compressible flow solver.

A multidisciplinary optimisation was performed to reduce the temperature in the solid domain surrounding the U-Bend. The geometry was parameterised using the NSPCC CAD kernel. The optimisation used the differentiated partitioned method described in Chapter 5 to obtain the gradient with respect to grid coordinates. The reverse differentiated CAD kernel was then used to obtain the final derivative of the objective function with respect to the design CAD design variables. This ensures that the optimised shape is available in CAD format.

The optimised geometry produced deformations to both the fluid and solid domains. The optimisation results showed a shrinking of the U-Bend fluid section and an increase in the sold volume. This lead to an increase in the fluid velocity which increased the convective heat transfer and reduced the cost function by 4.3%. The obtained results would lead to an undesirable increase in pressure loss therefore the results could be improved by modifying the objective function to consider the pressure loss.

Chapter 8

Conclusion

The predominant approach to Conjugate Heat Transfer (CHT) optimisation is the use of gradient-free optimisation methods which suffer from high computational costs and limit the number of design variables which could be used. Gradient-based adjoint methods promise to eliminate the high computational cost through efficient gradient calculation which also allows for an increase in the number of design variables used. However, the commonly favoured continuous adjoint method offsets this advantage with a high cost in the development of the adjoint solvers. The discrete adjoint method developed in this work alleviates the developmental cost imposed by the continuous adjoint by building the adjoint solvers using Automatic Differentiation (AD).

In this thesis, a discrete adjoint framework for fully coupled partitioned CHT optimisation was developed. The adjoint was obtained by applying source transformation AD to the in-house flow solver, mgOpt, and the open-source heat conduction solver, CalculiX. This resulted in a lower implementation cost than continuous adjoint and required no simplifications as is often done for CHT optimisation problems. Consequently, the bottleneck of the number of design variables which is faced by gradient-free optimisation methods is eliminated and the cost of developing the adjoint is significantly reduced. Therefore, more interesting and challenging CHT optimisations can be performed, devoid of any simplifications to the governing equations, while using a rich design space. This has the effect of reducing both the computational cost and barrier to entry for gradient-based CHT optimisation and has the potential to significantly shorten the design cycle.

CHT problems are often solved using a partitioned approach in which two different numerical solvers are used to solve the governing equations of the fluid and solid domains.

This allows the use of highly reputable solvers for the separate domains. The partitioned approach requires the use of coupling algorithms which iteratively exchange boundary conditions until temperature and heat flux are continuous between the fluid and solid domains as discussed in Chapter 2. The use of Robin boundary conditions in the fluid domain is relatively new and the results of numerical analysis in Chapter 3 show that Robin boundary conditions lead to much faster convergence of the primal coupling problem.

The core contributions of this thesis are the differentiation of CalculiX for CHT and the differentiation of the partitioned coupling algorithms used. The differentiated coupling algorithms also require coupling iterations to converge the gradient and the exchange of gradients through the differentiated algorithms was clearly outlined. The three coupling algorithms differentiated in Chapter 5 involved a combination of Dirichlet, Neumann, and Robin boundary conditions. Previous work has focused on differentiation of a combination of Dirichlet-Neumann coupling algorithms. The differentiation of recently developed coupling algorithms using Robin boundary conditions is one of the novelties of this thesis.

The convergence of the coupled adjoint gradients was investigated in Chapters 5 & 6 and it was seen that poor convergence of the objective function negatively impacted the accuracy of gradients. Furthermore, in a similar way to the primal flow equations, the fluid adjoint equations could be partially converged between reverse coupling iterations without significant loss of accuracy in the gradients. It was also shown that the number of reverse coupling iterations could be significantly lower than the number of primal coupling iterations without severely impacting gradient accuracy. This advantage was only seen for coupling algorithms which use Robin boundary conditions in the fluid domain and it was shown that accurate gradients can be obtained in as little as one reverse coupling iteration in Chapter 6. This highlights a useful and previously unreported advantage of the use of Robin boundary conditions in CHT problems.

Therefore it is recommended that the use of Robin boundary conditions in CHT problems become more widespread as they speed up the convergence of both the primal and coupled adjoint. This will lead to significant computational cost savings during both the primal and adjoint runs and speed up the design process. This is highlighted by the results of the inverse design optimisation which showed that Robin-based coupling algorithms required less wall clock time to complete the optimisation. Furthermore, it is recommended that where possible, the primal run of CHT optimisation problems need not be solved until temperature and heat flux is continuous between domains, rather, they can be terminated once the cost function is converged to just 4 significant figures.

The differentiated coupling algorithms made use of two adjoint solvers which were obtained via source transformation Automatic Differentiation (AD) in Chapter 4. The use of AD is very effective as it allowed the fluid adjoint solver to be developed automatically through the use of differentiation scripts. Although problems were encountered while differentiating CalculiX, the maturity and further development of AD tools will eliminate the problems related to mixed programming languages faced during this thesis.

The differentiated coupling algorithms were used with standard optimisation algorithms to solve several optimisation problems in Chapter 6. At no point were the problems simplified to conduction or convection only problems and partitioned coupling methods were used. The benefits of the author's developments culminated in the possibility of a novel optimisation problem related to an internal cooling channel u-bend in Chapter 7. Furthermore, the number of design variables used in two of the optimisation problems was greater than 10^2 which is not feasible for standard gradient-free optimisation methods. Consequently, it is the author's hope that there will be an increase the use of gradient-based methods in CHT optimisation and more interest in the use of the discrete adjoint with partitioned coupling algorithms using Robin boundary conditions.

8.1 Future work

This thesis has shown the advantage Robin-based coupling algorithms offer in terms of computational time savings. For Robin-based coupling algorithms, the choice of the virtual parameters $\tilde{R}$ and $\tilde{h}$ could be further investigated. Currently, there is no widely accepted method of determining the choice of these parameters which would lead to fast convergence while maintaining stability.

Only the three coupling algorithms used in this work were differentiated. For completeness, the 4 other fixed-point algorithms in Table 2.1 could also be differentiated. The strength of Automatic Differentiation (AD) was also demonstrated in this work. AD could be used to obtain the Jacobian of the coupled problem which can then be used for Newton-like coupling algorithms.

The results obtained in Chapter 3 for the C3X and MarkII turbine blades show that the Spalart-Allmaras turbulence model is unable to accurately predict the transition from laminar to turbulent flow. This inaccuracy certainly has a negative impact on the ability of the optimiser to obtain optimum designs. Therefore the implementation of transition turbulence models will ensure truly optimum designs can be found.

The 2D optimisation of the MarkII turbine blade performed in Chapter 6 only used the location of the internal channels as design variables. Further work could include the deformation to the fluid-solid interface subject to maintaining pressure loss. The radius and shape of the cooling channels could also be changed, subject to some manufacturing constrains. Additionally, A full 3D simulation of the turbine blade which also models the coolant flow would lead to more complete optimisation designs.

Only matching interface grids were used in this work. Although the computational time of solid is significantly less than that of the fluid, the use of non-conforming meshes between domains could reduce the computational time further. This would require the use of interpolation algorithms which also have to be differentiated and the effect of this on the accuracy and convergence of multidisciplinary gradients could be investigated.

The thermal load optimisation could be further developed to include a stress calculation. The results of the CHT simulations could be used for a stress analysis in order to more accurately predict the blade thermal stress and lifetime. CalculiX is already capable of stress analysis, and can therefore be differentiated using AD. Looking further, the centrifugal forces on the blade can also be considered for a more rigorous optimisation study.

For the U-Bend optimisation, the obtained thermal optimisation results would lead to a significant increase in pressure loss. Therefore an objective function taking pressure loss into account would lead to more robust optimisation results. The results were obtained with a relatively coarse mesh. A finer mesh could be used to more accurately capture the flow field, and a parallelised solvers could be used to reduce the computational runtime.

Appendices

Appendix A

Differentiation of CalculiX

CalculiX solves the steady state heat equation as linear system of equations described in Chapter 2. That is,

$$\int_{\Omega_s} \Big(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z}\Big) N_i d\Omega_s \quad = \quad Q, \tag{A.1}$$

$$\mathbf{K}T \quad = \quad Q. \tag{A.2}$$

```
SUBROUTINE nonlingeo_fortran($q_w^{\uparrow}$, $T_w^{\downarrow}$, $T_{sink}^{\downarrow}$, $h^{\downarrow}$, $\vec{x}^{\downarrow}$)
    ...
    ! set boundary conditions
    ! calc. internal forces
    ! build system of equations
    ...
    !solve linear system
    CALL linear_solver($\mathbf{K}$,T,Q)
    ...

END SUBROUTINE nonlingeo_fortran
```

Listing A.1: Sample version of main routine.

The linear system is solved using a black box linear solver (see Listing A.1) therefore the source code is unavailable for Tapenade. Consequently, the linear solver had to be hand differentiated as recommended by Tapenade [79]. By applying the chain rule, the forward differentiated (tangent) code solves the following tangent system of equations

$$\dot{\mathbf{K}}T + \mathbf{K}\dot{T} = \dot{Q}, \tag{A.3}$$

$$\mathbf{K}\dot{T} = \dot{Q} - \dot{\mathbf{K}}T, \tag{A.4}$$

$$\mathbf{K}\dot{T} = RHS, \tag{A.5}$$

where $\dot{K}$ is the derivative of the conduction matrix w.r.t the the coupling boundary conditions Q, $\dot{Q}$ is the tangent seed vector, and $\dot{T}$ is the derivative of the Temperature field w.r.t the coupling boundary conditions. The procedure for solving the tangent system is as follows, First, Equation (A.2) is solved by the linear solver to obtain T, next the RHS term in Equation (A.5) is calculated, then the linear solver then solves for $\dot{T}$. Listiing A.2 shows pseudo-code of this process.

```
SUBROUTINE nonlingeo_fortran_d(..., $\dot{q}_w^{\uparrow}$, $\dot{T}_w^{\downarrow}$, $\dot{T}_{sink}^{\downarrow}$, $\dot{h}^{\downarrow}$, $\dot{\vec{x}}^{\downarrow}$)
    ...
    !solve tangent linear system
    CALL linear_solver($\mathbf{K}^{\downarrow}$,$T^{\uparrow}$,$Q^{\downarrow}$)
    $RHS$ = $\dot{Q} - \dot{\mathbf{K}}T$
    CALL linear_solver($\mathbf{K}^{\downarrow}$,$\dot{T}^{\uparrow}$,$RHS^{\downarrow}$)
    ...
END SUBROUTINE nonlingeo_fortran_d
```

Listing A.2: Forward differentiated linear solver.

The reverse differentiation of the linear system in Equation (A.5) makes use of the dot product equality between the tangent and adjoint [79] which leads to

$$\overline{Q} = (\mathbf{K}^{-1})^T \overline{T} \tag{A.6}$$

$$\overline{K}_{i,j} = -T_j \overline{Q}_i \tag{A.7}$$

where $\overline{T}$ is the adjoint seed vector, and $\overline{K}$ and $\overline{T}$ are the derivatives of the conduction matrix and the temperature field w.r.t coupling boundary condition respectively. As the conductivity matrix is symmetric, the primal and adjoint system matrix is the same. As an iterative scheme requiring several solves of the linear system in Equation A.2 is used to converge the primal solution, the adjoint gradients needs to be accumulated leading to

$$\overline{Q}_j = \overline{Q}_j + \overline{Q}_j \tag{A.8}$$

$$\overline{K}_{i,j} = \overline{K}_{i,j} - T_j \overline{Q}_i \tag{A.9}$$

A pseudocode of the reverse differentiated linear system solve is shown in listing A.3.

```
SUBROUTINE nonlingeo_fortran_b(..., $\bar{q}_w^{\uparrow}$, $\bar{T}_{sink}^{\uparrow}$, $\bar{h}^{\uparrow}$, $\bar{\vec{x}}^{\uparrow}$, $\bar{T}_w^{\downarrow}$)
   ...
   !solve adjoint linear system
   $\mathbf{K}^T$ = $\mathbf{K}$ ! symmetric matrix
   CALL linear_solver($\mathbf{K}^{T\downarrow}$,$\overline{w}^{\uparrow}$,$\overline{T}^{\downarrow}$)
   $\overline{Q}$ = $\overline{Q} + \overline{w}$
   CALL linear_solver($\mathbf{K}^{\downarrow}$,$T^{\uparrow}$,$Q^{\downarrow}$)
  DO i=1,n
   DO j=1,n
    $\overline{\mathbf{K}}_{\mathbf{ij}}$ = $\overline{\mathbf{K}}_{\mathbf{ij}} - T_j \times \overline{w}_i$
   END DO
  END DO
   ...
END SUBROUTINE nonlingeo_fortran_b
```

Listing A.3: Reverse differentiated linear solver.

Appendix B

Differentiating the partitioned approach

In order to illustrate the data flow between solvers during coupling iterations, the exchange of boundary conditions at one interface node is considered. The interface temperatures on the fluid and solid side are denoted as T_{fw} and T_{sw} respectively. Upon convergence of the coupling algorithm, both temperatures will have the same value. The heat flux between domains can be estimated using the conductivity and temperature difference between the interface and first off-wall node. The Robin parameters $\tilde{R}$ and $\tilde{h}$ are user defined numbers.

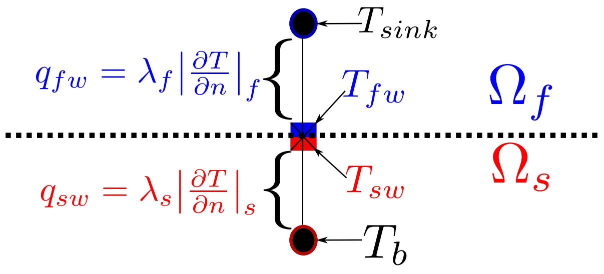

T_{sink}
$q_{fw} = \lambda_f \left|\frac{\partial T}{\partial n}\right|_f$
T_{fw}
Ω_f
$q_{sw} = \lambda_s \left|\frac{\partial T}{\partial n}\right|_s$
T_{sw}
Ω_s
T_b

Figure B.1: Description of boundary terms for one interface node

The solid and fluid solvers can then be represented by the pseudocode shown in listings B.1 and B.2. The fluid and solid solvers are coupled through listing B.3 which mimics the exchange of boundary conditions at a single interface node. The ↓ symbol is used to signify inputs to the routines while the ↑ signifies an output.

```
SUBROUTINE solid($T_{sw}^{\uparrow}$, $q_{sw}^{\uparrow}$, $T_b^{\downarrow}$, $T_{fw}^{\downarrow}$, $\tilde{h}^{\downarrow}$, $T_{sink}^{\downarrow}$)
...
   $q_{sw}$ = $\lambda_s \times$($T_b$ - $T_{fw}$) ! solve for interface heat flux
   IF (HFRB == TRUE) THEN
     $T_{sw}$ = (-$q_{sw}$/$\tilde{h}$) + $T_{sink}$ ! solve for interface temperature
   END IF
END SUBROUTINE solid
```

Listing B.1: Simplified solid solver.

```
SUBROUTINE fluid($T_{fw}^{\uparrow\downarrow}$, $\tilde{h}^{\uparrow}$, $T_{sink}^{\uparrow}$, $q_{sw}^{\downarrow}$, $\tilde{T}_s^{\downarrow}$, $\tilde{R}^{\downarrow}$)
   ...
   IF (TFRB OR HFRB == TRUE) THEN
     $q_w$ = $\tilde{R} \times$($\tilde{T}_s$-$T_{fw}$) ! Robin boundary condition
   END IF
   $T_{fw}$ = ($q_{fw}$/h) + $T_{freestream}$ ! solve for interface temperature
   IF ( HFRB == TRUE) THEN
     $T_{sink}$ = $T_{fw}$ + ($q_{fw}$/$\tilde{h}$)
   END IF
END SUBROUTINE fluid
```

Listing B.2: Simplified fluid solver.

```
SUBROUTINE couple($T_w^{\uparrow}$, $q_w^{\uparrow}$, $T_b^{\downarrow}$)
   $T_{fw} =$ ! initial guess
   do i=1,end  ! loop until convergence
        call solid($T_{sw}^{\uparrow}$, $q_{sw}^{\uparrow}$, $T_b^{\downarrow}$, $T_{fw}^{\downarrow}$, $\tilde{h}^{\downarrow}$, $T_{sink}^{\downarrow}$) ! solve solid
        ! For TFRB & hFRB
        IF ( TFFB == FALSE) THEN
            call calc_ts($\tilde{T}_s^{\uparrow}$, $\tilde{R}^{\downarrow}$, $q_w^{\downarrow}$, $T_w^{\downarrow}$)
        END IF
        Call fluid($T_{fw}^{\uparrow\downarrow}$, $\tilde{h}^{\uparrow}$, $T_{sink}^{\uparrow}$, $q_{sw}^{\downarrow}$, $\tilde{T}_s^{\downarrow}$, $\tilde{R}^{\downarrow}$) ! solve fluid
   end do
END SUBROUTINE couple
```

Listing B.3: An example fixed-point coupling iteration scheme.

B.0.1 Partitioned tangent linear method

To obtain the tangent mode of the fixed-point coupling method, forward mode AD is applied to the coupling subroutine to produce the differentiated version which is shown in listing B.4. The "_d" notation denotes a forward differentiated routine and a dot above a variable implies a seed vector. The symbol f is used to denote some form of computation e.g addition, multiplication, division etc which has been analytically differentiated.

```
SUBROUTINE couple_d(..., $\dot{T}_w^{\uparrow}$, $\dot{q}_d^{\uparrow}$, $\dot{T}_b^{\downarrow}$)
   ! loop for same number as primal
   do i=1,end
        call solid_d(..., $\dot{T}_{fw}^{\uparrow\downarrow}$, $\dot{q}_{sw}^{\uparrow}$, $\dot{T}_b^{\downarrow}$, $\dot{T}_{sink}^{\downarrow}$, $\tilde{h}_d^{\downarrow}$) ! tangent solid solver
        ! For TFRB & hFRB
        IF ( TFFB == FALSE) THEN
            $\tilde{T}_{sd}$ = $f(\dot{q}_w, \dot{T}_w)$ ! $\implies$ calc. $\tilde{T}_{sd}$
        END IF
        Call fluid_d(..., $\dot{T}_{fw}^{\uparrow\downarrow}$, $\dot{T}_{sink}^{\uparrow}$, $\tilde{h}_d^{\uparrow}$, $\dot{q}_{sw}^{\uparrow}$, $\tilde{T}_{sd}^{\downarrow}$) ! tangent fluid solver
   end do
END SUBROUTINE couple_d
```

Listing B.4: Tangent coupling iterations.

The forward differentiated (tangent linear) coupling algorithm shown in listing B.4 follows the same procedure as the primal coupling. The tangent variables flow in the same direction as their primal counterparts and the differentiated routines are also called in the same sequence as the primal. This is an advantageous feature of the tangent mode which makes debugging easier due to the similarity between the tangent and primal codes.

B.0.2 Partitioned adjoint method

Reverse mode AD is applied to the coupling routine to obtain the fixed-point adjoint in listing B.5. The fixed-point adjoint is an inverse of the primal and iterates backwards from the final to first iteration. The calls to the fluid and solid solvers are reversed and the input and output variables are reversed. During the reverse loop, it is necessary to reuse some intermediate values for the gradient calculations. Consequently, the fluid and solid states are stored during the primal run and recalled using "POP" statements shown in Listing B.5.

```
SUBROUTINE couple_b(..., $\overline{T}_b^{\uparrow}$, $\overline{T}_w^{\downarrow}$, $\overline{q}_w^{\downarrow}$)
   ! reverse loop
   do i=end,1
         POP ($T_{fw}^i$, $q_{sw}^i$) ! reuse transient state
        Call fluid_b(..., $\overline{q}_{sw}^{\uparrow}$, $\tilde{T}_{sb}^{\uparrow}$,$\overline{T}_{sink}^{\downarrow}$, $\overline{h}^{\downarrow}$) ! adjoint fluid solver
        ! For TFRB & hFRB
        IF ( TFFB == FALSE) THEN
            POP ($\tilde{T}_s^i$) ! reuse transient state
            $\overline{T}_{sw}$, $\overline{q}_{sw}$ = $f(\tilde{T}_{sb})$ ! $\Longrightarrow$ calc. $\tilde{T}_{sb}$
        END IF
        POP ($q_w^i$) ! reuse transient state
        call solid_b(...,$\overline{T}_{fw}^{\uparrow\downarrow}$, $\overline{q}_{sw}^{\downarrow}$, $\overline{T}_b^{\uparrow}$, $\overline{T}_{sink}^{\uparrow}$,$\overline{h}^{\uparrow}$) ! adjoint solid solver
        IF ( TFRB == TRUE) THEN
            $\overline{T}_w$ = $\overline{T}_{sw}$ + $\overline{T}_{fw}$ ! accumulate gradient
         END IF
        $\overline{T}_b$ += $\overline{T}_b$ ! accumulate gradient
   end do
END SUBROUTINE couple_b
```

Listing B.5: Reverse coupling iterations.

Bibliography

[1] Garret N. Vanderplaats. *Multidiscipline Design Optimization.* VR and D, 2007.

[2] Joaquim R. R. A. Martins and Andrew B. Lambe. Multidisciplinary design optimization: A survey of architectures. *AIAA Journal*, 51(9):2049–75, 2013.

[3] Joaquim R. R. A. Martins. *A coupled-adjoint method for high-fidelity aero-structural optimization.* PhD thesis, Stanford University, October, 2002.

[4] Gaetan K.W. Kenway, Joaquim R. R. A. Martins, and Graeme J. Kennedy. Aerostructural optimization of the common research model configuration. *American Institute of Aeronautics and Astronautics*, 2014.

[5] Stefan Keye, Thomas Klimmek, Mohammad Abu-Zurayk, Matthias Schulze, and Caslac Ilic. Aero-structural optimization of the nasa common research model. *American Institute of Aeronautics and Astronautics*, 2017.

[6] Tom Verstraete. *Multidisciplinary Turbomachinery Component Optimization Considering Performance, Stress, and Internal Heat Transfer.* PhD thesis, von Karman Institute, 2008.

[7] Sebastian Scholl. *Large-Eddy Simulation, Stability and Optimization of the Conjugate Heat Transfer for Cooling Channels.* PhD thesis, RWTH AACHEN, February 2017.

[8] Joaquim R. R. A. Martins, Juan J. Alonso, and James J. Reuther. Complete configuration aero-structural optimization usinf a coupled sensitivity analysis method. *AIAA 2002-5402*, 2002.

[9] Antonio Fazzolari, Nicolas R. Gauger, and Joël Brezillon. Efficient aerodynamic shape optimization in mdo context. *Journal of Computational and Applied Mathematics*, 203:548–560, 2005.

[10] Joaquim R. R. A. Martins and John T. Wang. Review and unification of methods for computing derivatives of multidisciplinary computational models. *American Institute of Aeronautics and Astronautics*, 51, 2013.

[11] F. Christakopoulos. *Sensitivity computation and shape optimisation in aerodynamics using the adjoint methodology and Automatic Differentiation.* PhD thesis, Queen Mary University of London, 2013.

[12] Arash Mousavi and Siva K. Nadarajah. Adjoint-based multidisciplinary design optimization of cooled gas turbine blades. *AIAA 2011-1131*, 2011.

[13] Seyyed Arash Mousavi. *Constrained aerodynamic and heat transfer optimization of gas turbine blades using an adjoint approach.* PhD thesis, McGill University, August 2012.

[14] M. Zeinalpour, K. Mazaheri, and K.C. Kiani. A coupled adjoint formulation for non-cooled and internally cooled turbine blade optimization. *Applied Thermal Engineering*, 105(327-335), 2016.

[15] Michele Ferlauto. An inverse method of designing the cooling passages of turbine blades based on the heat adjoint equation. In *Proceedings of the Institution of Mechanical Engineers. Part A, Journal of Power and Energy*, pages 328–339, 2014.

[16] K.T. Gkaragkounis, E.M. Papoutsis-Kiachagias, and K.C. Giannakoglou. The continuous adjoint method for shape optimization in conjugate heat transfer problems with turbulent incompressible flows. *Applied Thermal Engineering*, 140:351–362, 2018.

[17] Ping He, Charles A. Mader, Joaquim R. R. A. Martins, and Kevin J. Maki. Aerothermal optimization of internal cooling passages using a discrete adjoint method. Technical report, university of Michigan, Ann Arbor, 2018.

[18] O. Burghardt and N.R. Gaugaer. Accurate gradient computations for shape optimization via discrete adjoints in cfd-related multiphysics problems. *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, 2018.

[19] Zhongran Chi, Haiqing Liu, Shusheng Zang, and Guangyun Jiao. Conjugate heat transfer optimization of the nonuniform impingement cooling structure of a hpt 2nd stage vane. In *Proceedings of the ASME Turbo Expo 2015: Turbine Technical Conference and Exposition*, 2015.

[20] K. Mazaheri, M. Zeinalpour, and H.R. Bokaei. Turbine blade cooling passages optimization using reduced conjugate heat transfer methodology. *Applied Thermal Engineering*, 2016.

[21] Stefano Caloni, Sharokh Shahpar, and Vassili V. Toropov. Multi-disciplinary design optimisation of the cooled squealer tip for high pressure turbines. *Aerospace*, 2018.

[22] Emmanuel Radenac, Jérémie Gressier, and Pierre Millan. Methodology of numerical coupling for transient conjugate heat transfer. Technical report, HAL, 2014.

[23] Yousef S.H. Najjar and Ibrahin A.I. Balawneh. Optimization of gas turbines for sustainable turbojet propulsion. *Propulsion and Power Research*, 4:114–121, June 2015.

[24] Barinyima Nkoi, Pericles Pilidis, and Theoklis Nikolaidis. Performance and assessment of simple and modified cycle turboshaft gas turbines. *Propulsion and Power Research*, 2:96–106, June 2013.

[25] Alok Majumdar and S.S. Ravindran. Numerical prediction of conjugate heat transfer in fluid network. *American Institute of Aeronautics and Astronautics*, 2010.

[26] Theodore L. Bergman, Adrienne S. Lavine, Frank P. Incropera, and David P. Dewitt. *Fundamentals of heat and mass transfer.* John Wiley & Sons, seventh edition, 2011.

[27] G. Dhont and K. Wittig. Calculix. http://www.calculix.de/.

[28] Guido Dhont. *CalculiX CrunchiX USER'S MANUAL version 2.10*, March 2016.

[29] Steven R Allmaras and Forrester T Johnson. Modifications and clarifications for the implementation of the spalart-allmaras turbulence model. In *Seventh international conference on computational fluid dynamics (ICCFD7)*, pages 1–11, 2012.

[30] S. Xu, D. Radford, M. Meyer, and J.-D. Müller. Stabilisation of discrete steady adjoint solvers. *Journal of Computational Physics*, 299:175–195, 2015.

[31] Shenren Xu. *CAD-based CFD shape optimisation using discrete adjoint solvers*. PhD thesis, Queen Mary, University of London, September 2015.

[32] Mateusz Gugała. *Output-based mesh adaptation using geometric multi-grid for error estimation.* PhD thesis, School of Engineering and Materials Science, Queen Mary University of London, London, UK, 2019. to be submitted.

[33] P. L. Roe. Approximate reimann solvers parameter vectors and difference schemes. *Journal of Computational Physics*, 43(2):357–372, 1981.

[34] Meng-Sing Liou. A sequel to ausm, part ii: Ausm$_{+}^{up}$ for all speeds. *Journal of Computational Physics*, 214(137-170), 2006.

[35] Joris Degroote. *Development of algorithms for the partitioned simulation of strongly coupled fluid-structure interaction problems*. PhD thesis, Ghent Universiteit, 2010.

[36] Gene Hou, Jin Wang, and Anita Layton. Numerical methods for fluid-structure interaction. *Communications in Computational Physics*, 12(2):337–377, August 2015.

[37] V. Ganine, N.J. Hills, and B.L. Lapworth. Nonlinear acceleration of coupled fluid-structure transient thermal problems by anderson mixing. *International Journal for Numerical Methods in Fluids*, 71:939–959, 2013.

[38] Tom Verstraete and Sebastian Scholl. Stability analysis of partitioned methods for predicting conjugate heat transfer. *International Journal of Heat and Mass Transfer*, 101:852–869, 2016.

[39] William D. Henshaw and Kyle K. Chand. A composite gird solver for conjugate heat transfer in fluid-structure systems. *Journal of Computational Physics*, February 2009.

[40] M. B. Giles. Stability analysis of numerical interface condtions in fluid-structure thermal analysis. *International Journal for Numerical Methods in Fluids*, 25:421–436, 1997.

[41] Rocco Moretti, Marc-Paul Errera, Vincent Couaillier, and Frédéric Feyel. Stability, convergence and optimization of interface treatments in weak and strong thermal fluid-structure interaction. Technical report, HAL, May 2018.

[42] V. Kazemi-Kamyab, A.H. van Zuijlen, and H.Bijl. Accuracy and stability analysis of a second-order time-accurate loosely coupled partitioned algorithm for transient conjugate heat transfer problems. *International Journal for Numerical Methods in Fluids*, pages 113–133, September 2013.

[43] Verstraete Tom and Rene Van den Braembussche. A novel method for the computation of conjugate heat transfer with coupled solvers. In *International symposium on heat transfer in gas turbine systems*, August 2009.

[44] E. Divo, E. Steinthorsson, F. Rodriguez, A. J. Kassab, and J.S. Kappat. Glenn-ht/bem conjugate heat transfer solver for large-scale turbomachinery models. Technical Report NASA/CR-2003-212195, NASA, 2003.

[45] A. V. Luikov. Conjugate convective heat transfer problems. *International Journal of Heat and Mass Transfer*, 17:257–265, 1974.

[46] Hylton L.D., Mihelc M.S., Turner E.R., Nealy D.A., and York R.E. Analytical and experimental evaluation of the heat transfer distribution over the surfaces of turbine vanes. Technical Report NASA CR 168015, NASA, 1983.

[47] Tetsuya Yoshiara, Daisuke Sasaki, and Kazuhiro Nakahashi. Conjugate heat transfer simulation of cooled turbine blades using unstructured-mesh cfd solver. *American Institute of Aeronautics and Astronautics*, January 2011.

[48] Zhang Hongjun, Zou Zhengping, Li Yu, Ye Jian, and Song Songhe. Conjugate heat transfer investigations of turbine vane based on transition models. *Chinese Journal of Aeronautics*, 26:890–897, 2013.

[49] Zeng-Rong Hao, Chun-Wei Gu, and Xiao-Dong Ren. The application of discontinuous galerkin methods in conjugate heat transfer simulations of gas turbines. *Energies*, 2014.

[50] Gang Lin, Karsten Kusterer, Anis Haj Ayed, Dieter Bohn, and Takao Sugimoto. Conjugate heat transfer analysis of convection-cooled turbine vanes using $\gamma - \mathrm{Re}_\theta$ transition model. *International Journal of Gas Turbine, Propulsion and Power Systems*, 6(3), December 2014.

[51] Wang Qiang, Guo Zhaoyuan, Zhou Chi, Feng Guotai, and Wang Zhongqi. Coupled heat transfer simulation of a high-pressure turbine nozzle guide vane. *Chinese Journal of Aeronautics*, 22:230–236, September 2009.

[52] J.-D. Müller, W. Jahn, C.Othmer, M.Megahed, N. Hollette, G. Pierrot, K.-U. Bletzinger, and E. Stavropolou. Goal-based flow optimisation for automotive design. *Procedia: Soc. Beh. Sc.*, 48:3599–3612, 2012.

[53] J.-D. Müller and P. Cusdin. On the performance of discrete adjoint cfd codes using automatic differentiation. *International Journal for Numerical Methods in Fluids*, 47(6-7):939–945, 2005.

[54] M. B. Giles and N. A. Pierce. An introduction to the adjoint approach to design. *Flow, Turb. Comb.*, 65:393–415, 2000. also Oxford University Computing Laboratory NA report 00/04.

[55] L. Hascoët and V. Pascual. The Tapenade Automatic Differentiation tool: Principles, Model, and Specification. *ACM Transactions On Mathematical Software*, 39(3), 2013.

[56] Jianhua Zhou, Yuwen Zhang, J.K. Chen, and Z.C. Feng. Inverse estimation of surface heat condition in a three-dimensional object using conjugate gradient method. *International Journal of Heat and Mass Transfer*, 53:2643–2654, 2010.

[57] Tahar Loulou and Elaine P. Scott. An inverse heat conduction problem with heat flux measurements. *International Journal for Numerical Methods in Engineering*, 67:1587–1616, 2006.

[58] Wen-Lih Chen and Yu-Ching Yang. On the inverse heat conduction problem of the flow over a cascade of rectangular blades. *International Journal of Heat and Mass Transfer*, 51(4184-4194), 2008.

[59] Chen-Hung Huang and Wei-Chung Chen. A three-dimensional inverse forced convection problem in estimating surface heat flux by conjugate gradient method. *International Journal of Heat and Mass Transfer*, 43:3171–3181, 2000.

[60] Fu-Yun Zhao, Di Liu, Li Tang, Yu-Ling Ding, and Gaung-Fa Tang. Direct and inverse mixed convections in an enclosure with ventilation ports. *International Journal of Heat and Mass Transfer*, 52:4400–4412, 2001.

[61] Shaik Imran Ahamad and C. Balaji. Inverse conjugate mixed convection in a vertical substrate with protruding heat sources: a combined experimental and numerical study. *Heat and Mass Transfer*, 52:548–560, 2015.

[62] Eric Jones, Travis Oliphant, and Pearu Peterson. SciPy: open source scientific tools for Python. Technical report, https://www.scipy.org/, 2014.

[63] Majid Rezazadeh Reyhani, Mohammad Alizadeh, Alireza Fathi, and Hiwa Khaledi. Turbine blade temperature calculation and life estimation - a sensitivity analysis. *Propulsion and Power Research*, 2(2):148–161, 2013.

[64] Sushila Rani, Atul k. Agrawal, and Vikas Rastogi. Failure analysis of a fisrt stage in738 gas turbine blade tip cracking in a thermal power plant. *Case studies in Engineering Failure Analysis*, 8:1–10, April 2017.

[65] Mariusz Bogdan, Józef Błachnio, Artur Kułaszka, and Marcin Derlatka. Assessing the condition of gas turbine rotor blades with the optoelectronic and thermographic methods. *Metals*, March 2019.

[66] Bingxu Wang, Weihong Zhang, Gongnan Xie, Yingjie Xu, and Manyu Xiao. Multiconfiguration shape optimization of internal cooling systems of a turbine guide vane based on thermomechanical and conjugate heat transfer analysis. *Journal of Heat Transfer*, March 2015.

[67] Tom Verstraete, Filippo Coletti, Jérémy Bulle, Timothée Vanderwielen, and Tony Arts. Optimization of a U-bend for minimal pressure loss in internal cooling channels — Part I: Numerical method. *Journal of Turbomachinery*, 135(5):051015, 2013.

[68] S. Auriemma, M. Banovic, O. Mykhaskiv, H. Legrand, J.-D. Müller, and A. Walther. Optimisation of a U-bend using CAD-based adjoint method with differentiated CAD kernel. In *ECCOMAS Congress*, 2016.

[69] Xingchen Zhang. *CAD-based geometry parameterisation for shape optimisation using Non-uniform Rational B-splines.* PhD thesis, Queen Mary University of London, September 2017.

[70] Jens-Dominik Muller Tom Verstraete, Lasse Muller. Adjoint based design optimization of an internal cooling channel u-bend for minimized pressure losses. In *Proceedings of the 12th European Conference on Turbomachinery Fluid Dynamics & Thermodynamics*, Stockholm, Sweden, apr 2017.

[71] Je-Chin Han, Sandip Dutta, and Srinath Ekkad. *Gas Turbine Heat Transfer and Cooling Technology.* CRC Press, 2012.

[72] Filippo Coletti, Tom Verstraete, Jérémy Bulle, Timothée Van der Wielen, Nicolas Van den Berge, and Tony Arts. Optimization of a u-bend for minimal pressure loss in internal cooling channels-part ii: Experimental validation. *Journal of Turbomachinery*, 135(5):051016, 2013.

[73] P. Moinier, J.-D. Müller, and M. B. Giles. Edge-based multigrid schemes and preconditioning for hybrid grids. *AIAA Journal*, 40(10):1954–60, 2002.

[74] T.T. Robinson, C.G. Armstrong, H.S. Chua, C. Othmer, and T. Grahs. Optimizing parameterized CAD geometries using sensitivities based on adjoint functions. *Computer-Aided Design & Applications*, 9(3):253–268, 2012.

[75] M. Banovic, O. Mykhaskiv, S. Auriemma, A. Walther, H. Legrand, and J. D. Müller". Automatic differentiation of the Open CASCADE Technology CAD system and its coupling with an adjoint cfd solver. In *Optimization Methods and Software*, 2017.

[76] Auriemma S. and Banovic M. Mykhaskiv O. Applications of differentiated CAD kernel in gradient-based aerodynamic shape optimisation. In *2018 Joint Propulsion Conference*, page 4828, 2018.

[77] O. Mykhaskiv, M. Banovic, S. Auriemma, A. Walther, and J.-D. Mueller. NURBS-based and Parametric-based shape optimisation with differentiated CAD kernel. *Computer-Aided Design and Applications*, pages 1–11, 2018.

[78] R. Jesudasan, M. Gugała, O.R. Imam-Lawal, and Jens-Dominik Müller. CAD-Based parameterisation framework for aerodynamic shape optimisation using adjoint sensitivites. to be submitted, September 2019.

[79] Tapenade. Frequently asked questions on tapenade. https://www-sop.inria.fr/tropics/tapenade/faq, April 2019.