RAPID DEVELOPMENT OF DISCRETE ADJOINT SOLVERS WITH APPLICATIONS TO MAGNETOHYDRODYNAMIC FLOW CONTROL

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

André Calado Marta

June 2007

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Juan J. Alonso) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Robert W. MacCormack)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Ilan M. Kroo)

Approved for the University Committee on Graduate Studies.

To my beloved wife, Inês.

Abstract

The motivation for this work arose from initial efforts in magnetohydrodynamics (MHD) flow control when plasma generated at high Mach numbers allows for electromagnetic actuators to change the characteristics of the flow. In the context of gradient-based optimization, this dissertation focuses on the problem of efficiently estimating the sensitivity of a given function of interest with respect to a large number of variables, in environments modeled by complex equations.

The discrete adjoint approach emerges as the best suitable option to deal with such complex equations and, in addition, allows for the use of automatic differentiation (AD) tools in the derivation of the adjoint solver. The selective application of AD is the central idea behind the *Automatic Differentiation adjoint* (*ADjoint*) approach. This approach has the advantages that it is applicable to arbitrary sets of governing equations and cost functions, and it is exactly consistent with the gradients that would be computed by exact numerical differentiation of the original solver. Furthermore, the approach is largely automatic, thus avoiding the lengthy development times usually required to develop discrete adjoint solvers for partial differential equations. It takes days, not years, to construct the *ADjoint* solver.

Sensitivities of aerodynamic coefficients with respect to several types of parameters, totaling over a half million variables, are computed and successfully validated against finite-difference approximations. The overall performance and accuracy of the method is shown to be better than conventional continuous adjoint approaches. The increased memory requirements can be eliminated at the expense of larger computational times for the *ADjoint*, that would bring the computational performance roughly on par with that of the best continuous adjoint solvers.

Acknowledgments

The years I have spent at Stanford have been simply extraordinary. Besides the state-of-the-art research done at this institution, the camaraderie with individuals whose origins span the whole globe, in the relaxed Northern Californian atmosphere, made it a truly unique and unforgettable experience.

Studying in the USA and living in the heart of Silicon Valley have only been made possible with the generous contributions from several Portuguese institutions. My first quarter at Stanford University was sponsored by the *Fundação Luso-Americana para o Desenvolvimento*, and I acknowledge Idalina Salgueiro for her support. During the following four years, I was fortunate to have been granted a scholarship from the *Fundação para a Ciência e a Tecnologia*. I am also thankful for the funding provided by the *Fundação Calouste Gulbenkian*, that allowed me to complete my doctoral work. In addition, I am grateful for Air Force Office of Scientific Research contract No. AF04-009-0078 under tasks monitored by Jonathan Poggie and John Schmisseur.

My deepest thanks go to Juan Alonso for being my academic and research adviser, guiding me throughout my doctoral degree, and providing invaluable advises when I needed the most. We all have role models in our lives, and Juan is definitely one I certainly admire. Not only does he possess a strong background in aeronautics and a clear vision of future challenges in this field, but he is also able to interact with his students in a very open and trustful way. I was extremely luck for being part of his Aerospace Design Laboratory, where I was presented with the opportunity of conducting challenging research under his direction.

I would like to thank my dissertation readers, Ilan Kroo and Robert MacCormack, for their comments and suggestions that contributed for the completeness of the final

version of this manuscript. A note of appreciation goes to Robert MacCormack, who has always welcomed me to his office where we had long discussions about research and life in general. He is a remarkable individual, both at academic and personal levels, and I greatly appreciate his kind, warm and encouraging words.

The fellow students at the department of Aeronautics and Astronautics have contributed tremendously to my welfare. Interacting with such brilliant and self-driven people has enriched me greatly. Among the several laboratory colleagues, I want to thank, in particular, to Edwin van der Weide for his assistance. Edwin has always lent me all the support I could possibly desire, regardless of the time, and I was also lucky to share his superb sense of humor.

Life at Stanford extended far beyond research. Countless hours were spent with Chinmay Patel, building aircraft models, playing with electronics, or flying at lake Lagunita. Friends from the Qualifying Exam study group, Hiroyuki Konno and Sei Higuchi, were always a pretext to have periodic social gatherings. Those long ethnic dinners will have a special place in my memory. Gabriela Caraccia and Pablo Jadzinsky have been a dear couple whom I deeply thank for all the moments shared. They are some of the best friends I have, and I wish we will continue this way for ever.

The Portuguese roots were kept strong with the precious help of the Stanford Portuguese Student group. This group experienced a steady growth since my arrival at Stanford, and it offered me a getaway that allowed me to cope with the separation from home. Jorge Lage, Maria Correia, Daniel Barros, Rita Lopez, Francisco Santos and Rita Oliveira were all important players in my social life that I feel in debt.

But the true meaning of life comes from those that love you just for being who you are. My mother and father, my sister, and my grandparents have always believed in me, and encouraged me to pursue my dreams at the expense of an arduous separation. Finally, a word of appreciation for the person who made all this possible, my wife Inês. If there is anyone who passionately trusts me and blindly supports my decisions, it is her. I was blessed to have her by my side for the most part while at Stanford, and she granted me love, comfort and assistance at a level I could never envision possible. I just wish we spend a long life together to have the opportunity to offset all she has done for me.

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

α	Angle of attack.
β	Angle of side-slip.
Δ	Perturbation step.
ϵ_0	Electric permittivity of free space.
γ	Ratio of specific heats.
κ	Thermal conductivity coefficient.
λ	Spectral radius.
μ	Molecular viscosity coefficient.
μ_m	Magnetic permeability of free space ($= 4\pi \times 10^{-7}$ H/m)
ψ	Adjoint variables, Lagrange multipliers.
ρ	Density.
ρ_e	Electric charge density.
σ	Electrical conductivity.
τ	Shear stress tensor.
ξ, η, ζ	Computational coordinates.

Roman symbols

$\mathbf{B}$	Magnetic induction field.
C	Constraint function.
c	Speed of sound, speed of light.
c_a	Alfvèn wave speed.
C_D	Coefficient of drag.
c_f	Fast magneto-acoustic wave speed.
C_L	Coefficient of lift.
C_M	Coefficient of moment.
C_p	Specific heat at constant pressure.
c_s	Slow magneto-acoustic wave speed.
C_v	Specific heat at constant volume.
$\mathbf{D}$	Electric displacement.
$\mathbf{E}$	Electric field.
E	Total energy.
e	Internal energy.
$\mathbf{H}$	Magnetic field strength.
$\mathbf{I}$	Identity matrix.
I	Objective, cost or merit function.
$\mathbf{J}$	Total electric current density.
$\mathbf{j}$	Conduction current density.

Kn Knudsen number.

L_{ref} Characteristic length.

$\mathbf{M}$ Magnetization.

M Mach number.

N_c Number of cells (or nodes) in the computational mesh.

N_I Number of functions of interest (cost and constraints).

N_s Number of cells (or nodes) in the stencil.

N_v Number of conservation variables/equations (conservation laws).

N_w number of variables of the state vector.

N_x Number of design variables.

$\mathcal{O}$ Truncation error.

$\mathbf{P}$ Polarization.

P MHD pressure.

p Pressure.

Pr Prandtl number.

Q Magnetic interaction parameter.

q Total electric charge.

$\mathcal{R}$ Residual vector.

R Specific gas constant (air= $287.05 J/(kg \cdot K)$).

R_b Magnetic force (or pressure) number.

Re Reynolds number.

Re_σ Magnetic Reynolds number.

T Temperature.

t Time.

$\mathbf{u}$ Velocity vector.

U Velocity magnitude.

u, v, w Velocity Cartesian components.

$\boldsymbol{w}$ Conservative variables, flow solution.

$\boldsymbol{x}$ Vector of design variables.

x, y, z Cartesian coordinates.

Z Total MHD energy.

Subscripts

0 Imposed (or intrinsic) magnetic field.

∞ Free-stream condition.

i Induced (or deviation) magnetic field.

i, j, k Computational indexes.

n Normal component.

x, y, z Cartesian components.

ref Reference condition.

Chapter 1

Introduction

1.1 Motivation

The history of the evolution of computers has been an impressive one, in particular during the last 50 years [116]. While the abacus was invented more than two thousand years ago by the Chinese, it was only with the advent of World War II that significant progress in computational capabilities started. The early computers were room-sized, relay-based calculators that performed binary arithmetic. The move from analog machines using relays and vacuum tubes to electronic computers, progressed through diode logic and transistor technology in the 50's. In the late 60's, the transition from discrete transistors to integrated circuits took place, and machines like the CRAY supercomputer with a power capable of 30 Mflop/s were produced. As of June of 2007, the list of supercomputers in service [154] keeps growing and it is currently topped by the IBM BlueGene/L system, installed at DOEs Lawrence Livermore National Laboratory (LLNL) in the USA, with an overwhelming performance of 280.6 Tflop/s. Even today's personal computers, with their multi-core processors, offer such a staggering computational power that any engineer or researcher can easily have access to such capabilities.

Obviously, the evolution of Computational Fluid Dynamics (CFD) followed that of the digital computer. While the early numerical solutions were restricted to the

potential flow equations on grids with a few thousand nodes, Direct Numerical Simulations (DNS) are now performed using more than 30 billion nodes [148]. At an academic level, graduate students now code their own Euler or Navier–Stokes flow solvers on their laptops. At an industrial level, Reynolds-Averaged Navier–Stokes (RANS) are commonplace.

Among all the phenomena in fluid mechanics, the one that triggered this dissertation was that of magnetohydrodynamics (MHD). A significant amount of work in the analysis of high-speed MHD has been carried out during the past decade. The governing equations for this class of problems can be extremely complex, especially if chemical reactions and turbulence effects are taken into account. Nevertheless, to date, accurate computational models have already been used to predict some of these complex flows, such the flow through a scram-jet that powers a hypersonic vehicle [31].

Despite reasonably matured MHD analysis tools, the fact is that the designs are still arrived at using very rudimentary computational design tools. There has even been some work done to control the flow in scram-jet inlets using MHD [45, 36, 145, 44], but none of these efforts employ formal design methods. The most common practice still relies on parametric studies [96]. More significant is the lack of publications on hypersonic design, demonstrating that very little effort has been devoted to design applications or to automate these flow control processes.

The main reasons for this lack of design focus have been that the analysis tools are still in the process of maturing and that the cost of the simulations is sufficiently large that design applications are beyond the reach of existing computing resources.

However, given that the analysis of MHD flows (with all the appropriate simplifications) has reached a certain level of maturity and that computational power has grown and become accessible, it is now important that an efficient design framework be built around high-speed MHD prediction capabilities that can be used in multi-disciplinary optimization (MDO) applications.

1.2 Generic design problem

In the context of optimization, a generic design problem can be posed as the minimization of a function of interest, I, (also called cost function or figure of merit) with respect to a vector of design variables, $\boldsymbol{x}$, while satisfying a set of linear or non-linear constraints. The cost function depends directly on the design variables and on the state of the system, $\boldsymbol{w}$, that may result from the solution of the governing equations of the problem. Thus we can write the vector-valued function I as

$$I = I\left(\boldsymbol{x}, \boldsymbol{w}(\boldsymbol{x})\right) . \tag{1.1}$$

For a given input vector $\boldsymbol{x}$, the solution of the governing equations subject to appropriate boundary conditions yields a state vector, $\boldsymbol{w}$, thus establishing the dependence of the state of the system on the design variables. We denote these governing equations by

$$\mathcal{R}\left(\boldsymbol{x}, \boldsymbol{w}(\boldsymbol{x})\right) = 0 . \tag{1.2}$$

In mathematical terms, this design problem can be expressed as

$$\begin{aligned} \text{Minimize} \quad & I\left(\boldsymbol{x}, \boldsymbol{w}(\boldsymbol{x})\right) \\ \text{w.r.t.} \quad & \boldsymbol{x} , \\ \text{subject to} \quad & \mathcal{R}\left(\boldsymbol{x}, \boldsymbol{w}(\boldsymbol{x})\right) = 0 \\ & C_i\left(\boldsymbol{x}, \boldsymbol{w}(\boldsymbol{x})\right) = 0 \quad i = 1, ..., m , \end{aligned} \tag{1.3}$$

where $C_i\left(\boldsymbol{x}, \boldsymbol{w}(\boldsymbol{x})\right) = 0$ represents m additional constraints that may or may not involve the flow solution.

When using a gradient-based optimizer to solve the design problem (1.3), the sensitivity of both the cost function I and the constraints C_i with respect to the design variables $\boldsymbol{x}$ are required. That is, $\frac{\mathrm{d}I}{\mathrm{d}\boldsymbol{x}}$ and $\frac{\mathrm{d}C_i}{\mathrm{d}\boldsymbol{x}}$ have to be determined. This optimization process is illustrated in figure 1.1.

This dissertation focuses on efficiently estimating the gradient (also designated as sensitivity or first derivative) of the function of interest (or vector of functions)

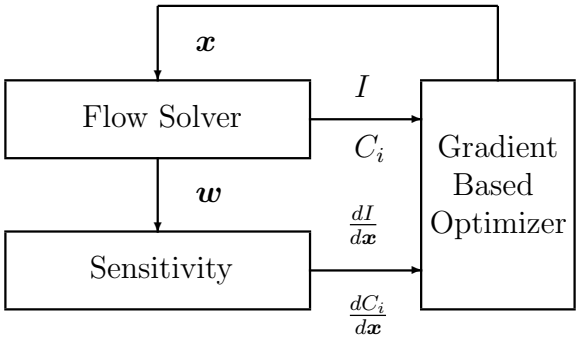


Figure 1.1: Schematic of a generic gradient-based optimization algorithm.

with respect to a very large number of design variables, in flow problems modeled by complex governing equations.

1.3 Control theory

The control theory approach, also called the adjoint method since the necessary gradients are obtained through the solution of the adjoint equations of the governing equations, emerges as an excellent candidate in achieving that goal. The adjoint method has already been mathematically well documented by Giles [55] and Lewis [84] and, being a semi-analytic method, it is capable of computing derivatives with the same precision as the quantity that is being differentiated and can potentially also be very efficient.

Adjoint methods have been used to perform sensitivity analysis of partial differential equations (PDEs) for over three decades. These methods were first applied to optimal control problems and thereafter used to perform sensitivity analysis of linear structural finite element models. The first application to fluid dynamics was due to Pironneau [128]. The method was then extended by Jameson to perform airfoil shape optimization [73] and since then it has been used to design laminar flow airfoils [34] and to optimize airfoils suitable for multi-point operation [122]. The adjoint method has also been extended to three-dimensional problems, leading to applications such as aerodynamic shape optimization of wings [77, 81] and complete aircraft configurations [138, 139, 74], as well as aero-structural design [106, 105]. The adjoint

theory has since been generalized for multi-disciplinary systems [107] and for MHD problems, using both the ideal model [97] and the low magnetic Reynolds number approximation [99, 98].

The adjoint method is extremely valuable because it provides a very efficient method to compute the sensitivity of a given function of interest with respect to many parameters by solving a system of equations of size equivalent to the governing equations of the flow. When using gradient-based optimization algorithms, the efficiency and accuracy of the sensitivity computations have a significant effect on the overall performance of the optimization. Thus, having an efficient and accurate sensitivity analysis capability is very important to constructing high-fidelity (and possible multi-disciplinary) design frameworks.

Given the value of adjoint methods, it seems odd that their application to aerodynamic shape optimization is not more ubiquitous. In fact, while adjoint methods have already found their way into commercial structural analysis packages, they have yet to proceed beyond research CFD solvers. One of the main obstacles is the complexity involved in the development and implementation of adjoint methods for nonlinear PDEs. In addition, since the adjoint equations are typically derived by hand, several simplifications are often done in order to make the derivation possible or just to expedite the code development phase. Even so, the development of an approximate, continuous adjoint solver for the RANS equations can require well over a year of tedious work for highly-qualified researchers with experience in this field. This is true even with significant approximations that have been shown to have a detrimental effect on the accuracy of the sensitivity derivatives, as demonstrated by Dwight [35].

1.4 Automatic differentiation

The solution to this problem might be *Automatic Differentiation* (AD) [60]. This approach relies on a tool that, given the original solver, creates code capable of evaluating the derivatives of quantities computed by the solver with respect to a number of parameters (that are typically inputs to the solver).

There are two different modes of operation for automatic differentiation tools: the

forward and the reverse modes. The forward mode propagates the required sensitivity at the same time as the solution is being computed. To use the reverse mode, the solver has to be run to convergence first, with intermediate variable values stored for every iteration. These intermediate variables are then used by the reverse version of the code to find the sensitivities. The forward mode is analogous to the finite-difference method, but without problems of step-size sensitivity. The reverse mode is similar to the adjoint method and is also efficient when computing the sensitivity of a function with respect to many parameters.

One drawback of the reverse mode is that the memory requirements can be prohibitively expensive in the case of iterative solvers, such as those used in CFD, because they require a large number of iterations to achieve convergence and many intermediate results may need to be stored.

Although efforts have been pursued to minimize the increase in memory requirements arising from iterative solvers [37], the fact remains that given typical parallel computing resources, it is still very difficult to apply reverse mode ideas to large-scale problems. The reverse mode of automatic differentiation has been applied to iterative PDE solvers by a few researchers with limited success [28, 54, 67, 82]. The main problems in each of these applications were the prohibitive runtime and memory requirements for the solution of three-dimensional problems.

In order to tackle these deficiencies, an efficient approach that uses AD tools to derive the corresponding adjoint equations is adopted, that follows the work that has been recently described by Martins *et al.* [101] and requires very little coding effort. This approach has already been successfully tested for both the Euler [102] and MHD equations [98].

1.5 Outcome of the research project

The objective of this dissertation is to turn the development of adjoint solvers into a routine and quick task that only requires the use of pre-existing code to compute the residuals of the governing equations (including boundary conditions) and the functions of interest.

This work employs a discrete adjoint formulation, that emerges as the best suitable option to deal with the complex equations, such those that govern MHD, and with the nature of the functions of interest that may be used in relevant design problems.

The end goal of this effort is to enable the development of discrete adjoint solvers for arbitrary PDEs in days, rather than years, so that the adjoint information may be used in a variety of applications including sensitivity analysis and, possibly, error estimation. To achieve this goal, and taking advantage of using a discrete adjoint formulation, the *Automatic Differentiation adjoint* (*ADjoint*) approach is proposed, in which automatic differentiation is used to compute only certain terms of the discrete adjoint equations, and not to differentiate the entire solver. These terms can then be used, together with standard techniques for the iterative solution of large linear systems, such as the preconditioned Generalized Minimum Residual (GMRES) algorithm [144], to carry out sensitivity analysis.

The major advantages of this method are that it is generic (applicable to any PDE solver), largely automatic, and exactly consistent with the flow solver. Because the process of automatic differentiation allows us to treat arbitrary expressions exactly, the sensitivities produced are perfectly consistent with those that would be obtained from an exact numerical differentiation of the original solver. Thus, typical approximations made in the development of adjoint solvers by hand differentiation (such as neglecting contributions from the variations resulting from turbulence models, spectral radii, artificial dissipation and upwind formulations) are not made here.

While the *ADjoint* method does not constitute a fully automatic way of obtaining sensitivities like pure automatic differentiation, it is much faster in terms of execution time and drastically reduces the memory requirements. In addition, when compared to the conventional continuous adjoint method, the proposed approach requires a much shorter implementation time and can be used to develop the discrete adjoint of an arbitrary solver with a greatly reduced probability of programming errors.

This methodology then aims to provide an important component, the efficient sensitivity analysis capability, toward the advance in the use of automated optimization using the control theory approach, applied to arbitrary complex flows, such as hypersonics and MHD.

1.6 Dissertation outline

This dissertation is structured in such a way that all the relevant background material is first reviewed, followed by a detailed description of implementation of the proposed methodology. Results obtained from the application to several test cases are then presented and final conclusions are drawn.

Starting with chapter 2, a brief review of the literature highlights the extreme complexity that might arise from current computational fluid dynamics (CFD) problems, in particular those dealing with hypersonic flow and magnetohydrodynamics. The complexity of the PDEs governing such flows is shown and light is shed on the possibility of performing flow control using magnetic effects.

Next, within the context of optimal control using a gradient-based optimizer, the available methods to estimate gradients of functions of interest with respect to control (design) variables are revisited in chapter 3, together with their advantages and disadvantages. Among the several analytic sensitivity analysis methods, special interest is given to the adjoint and automatic differentiation methods. This chapter ends with a discussion of the proposed novel hybrid approach — the *ADjoint* approach — that combines the desired good features of both the adjoint and automatic differentiation methods.

Then, a mathematical description of the physical models used in this work, in particular the governing equations of hypersonic flow under the influence of magnetic fields, is presented in chapter 4.

The most significant contribution of this dissertation is given in chapter 5, that covers the formulation of the discrete adjoint equations using the *ADjoint* approach. There, the various components of the design method developed are presented in a generic form applicable to any set of governing equations, regardless of their complexity.

That is followed by the results, in chapter 6, that cover the progress made since the early test cases using relatively small problems and hand-differentiated functions to larger and more complex applications using automatically differentiated functions,

involving up to half million design variables. The precision of the computed adjoint-based sensitivities is established and the performance of the hybrid *ADjoint* method is analyzed. Each of the adjoint solvers constructed is ultimately used in sample design problems, demonstrating their potential capabilities of making part of a larger optimization framework.

Lastly, in chapter 7, the conclusions drawn from the present research as well as some remarks concerning future work are made. This dissertation ends with some forecasts about the possible generalized use of the presented methodology in state-of-the-art high-fidelity design frameworks across not only academia but mainly industry.

Chapter 2

Magnetohydrodynamics

2.1 Hypersonic flow

Since the moment the Wright brothers debuted the first powered, heavier-than-air machine that achieved sustained and controlled flight, back in 1903, a quest for faster and faster aircraft took place. More than one hundred years have gone by and the transonic and supersonic frontiers have been crossed. However, the hypersonic regime still poses challenging problems that include fluid mechanics, propulsion, materials and structures, and stability and control.

Whereas the term supersonic speed is well defined as being greater than the speed of sound (Mach>1), the term hypersonic speed is somewhat nebulous, but often accepted as speeds greater than five times the speed of sound (Mach 5+).

During the last few years, there has been renewed interest in hypersonic flight, leading to an extensive number of conceptual studies [63, 62]. In July 2002, the world's first experimental flight of an air-breathing supersonic ramjet (scramjet) engine took place in Australia, as a result of the *HyShot* research project led by the Center for Hypersonics at the University of Queensland [24], demonstrating the possibility of supersonic combustion under flight conditions. This success attracted widespread interest in Australia's hypersonics research and has been succeeded by the Australian Hypersonics Initiative [3]. However, it was not until March 2004 that significant advances in hypersonic flight were made, in particular with the successful flight test of

the NASA X-43 [119], a hypersonic scramjet-powered research aircraft. The following year, NASA released an integrated hypersonic technology demonstration roadmap, as part of the Next Generation Launch Technology (NGLT) program, which triggered a feasibility study on the X-43D by the Future Hypersonic Flight Demonstration Office, with the objective of developing a baseline conceptual design, assessing its performance, and identifying the key technical issues [79].

More recently, the Air Force Research Laboratory (AFRL), Pratt & Whitney Rocketdyne (PWR), and NASA successfully tested the hypersonic Ground Demonstration Engine (GDE-2) [165] using hydrocarbon fuels and a thermally-balanced setup. The descendant of this effort is the current hypersonics program, the X-51, involving the AFRL, the Defense Advanced Research Projects Agency (DARPA), NASA, PWR and The Boeing Company. NASA just completed tests of the X-1 engine (April of 2007), the successor of the GDE-2 built by PWR, and will continue to test the X-2 engine which will be the flight engine for X-51 [166]. The airframe is being designed by Boeing and the demonstrator is scheduled to fly by 2009. Simultaneously, DARPA is proceeding along with the Falcon program [30], a scramjet reusable missile, and a number of other efforts are being pursued at NASA and abroad.

There are still many technical and scientific obstacles to overcome but the community is getting closer and closer to the stage where hypersonic flight may be possible. With that in mind, it is now the time to start looking at ways to optimize such designs in an efficient manner.

2.2 Chemically reacting flow and thermal non-equilibrium

When air flows at hypersonic speeds around vehicles, strong shock waves emerge in the regions of intense flow deceleration. There, much of the mean flow kinetic energy is converted into internal energy, namely translational and vibrational energy, which causes the temperature of the air to increase dramatically. In turn, this temperature increase leads to the dissociation and ionization of the air and a plasma is generated.

Under these conditions, the air is no longer in thermodynamic equilibrium, as the chemical processes occur in a time-scale comparable to the flow time-scale. The reacting air is said to be in non-equilibrium and must be described by finite-rate chemical reaction relations. This thermal and chemical non-equilibrium makes the analysis of such flows extremely complex and computationally expensive due to the high stiffness of the chemical equations that need to span several time scales.

There have been several efforts modeling these hypersonic chemically reacting flows. The usual flow governing equations, such as the Navier–Stokes (NS) equations, have to be solved coupled to finite-rate chemistry models. Back in 1985, Park made one of the first computations, considering a one-dimensional flow through a duct, using a complex model of air comprised of eleven species and seventeen reactions [127]. In the early 90's, the axisymmetric NS equations were strongly coupled to a five-species, five-reaction model, in the study of a nozzle flow by Walters *et al.* [160] or a blunt body by Josyula *et al.* [80]. Ten years later, the first numerical MHD simulations of chemically reacting flows were made, but were still restricted to two-dimensions: Damevin and Hoffmann [29] used a five-species, seventeen-reaction chemical model, loosely coupled to the ideal MHD equations, to analyze the flow around a blunt body; the viscous MHD equations together with seven-species, twelve-reaction model were used by Deb and Agarwal [31] in the performance study of MHD-bypass scramjet inlets.

Even though the loose coupling, in contrast to strong, between the NS or MHD equations and the chemical model, and the use of implicit time-integration schemes have helped to mitigate the computational cost, the complexity of this set of equations still poses a challenge today.

2.3 Flow control using magnetic effects

Since the medium, in a hypersonic flow, can be locally charged, the possibility opens up to use electromagnetic actuators to optimally control the flow. These actuators can generate magnetic fields that induce an acting force on the flow, as given by the

Lorentz force,

$$\mathbf{F}_{Lorentz} = q(\mathbf{E} + \mathbf{u} \times \mathbf{B}) \,, \tag{2.1}$$

where $\mathbf{F}_{Lorentz}$ is the force exerted on a fluid particle moving with velocity $\mathbf{u}$, whose electric charge is q, subject to an electric field $\mathbf{E}$ and a magnetic field $\mathbf{B}$. Considering the magnetic field alone, the flow particle will experience a force as illustrated in figure 2.1. Thus, the intensity and direction of the applied magnetic field determines the force exerted on the flow.

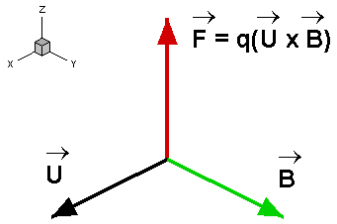


Figure 2.1: Lorentz (magnetic) force.

Several studies have already been made to evaluate the potential use of this phenomenon in hypersonic flow control applications. As early as 1967, experimental observations were made by Nowak *et al.* [124], revealing that the bow shock stand-off distance and drag increased, in re-entry vehicles, with the application of moderately strong magnetic fields. In that same year, Porter and Cambel [131] made a theoretical investigation on this same subject using similarity analysis, corroborating those observations. Four years later, one of the first numerical analysis of the MHD blunt body problem was performed by Coakley and Porter [26]. This problem has been revisited recently but some of the results are contradictory with respect to the possible to drag reduction, as seen on the work of Agarwal and Augustinus [2] and Gaitonde and Poggie [46]. Since then, many other applications have been developed. The concept of MHD energy-bypass scramjet inlets was proposed by Gaitonde and Poggie [49, 43], and MacCormack [94], among others. The potential heat transfer mitigation through magnetic control in a hypersonic blunt body flow has also been analyzed by Poggie [129, 130]. Shock-shock interaction and boundary-layer separation suppression have also been shown possible by Gaitonde [44], and Gaitonde and

Miller [45]. Another possible flow control utility was tackled by Brian *et al.* [36], who studied Mach reflection for double-fin scram-jet inlet configurations for different magnetohydrodynamic forces. Even the replacement of movable control surfaces by the use of glow discharges in plasma flows was evaluated in the work of Shang *et al.* [145].

As mentioned in section 1.1, all of these experiments have been based on either trial-and-error or parametric studies and no formal design approach has been employed yet. However, any of these problems can be viewed as an optimization problem in which a cost function must be minimized by varying a set of control variables, while satisfying a specific set of constraints. One alternative to carry out these optimizations is to use a gradient-based non-linear optimizer, following the algorithm shown in figure 1.1.

To tackle this kind of optimization problem, two important conditions must be met: the flow must be accurately predicted and the sensitivities of the cost and constraint functions must be computed at reasonable expense, with respect to a significant number of parameters. As for the first condition, in order to model the flow, one must couple the flow governing equations (typically the Navier–Stokes equations) to the Maxwell equations, leading to a new set of equations called the magnetohydrodynamic (MHD) equations. Details of these models are reviewed in section 2.4. The second condition poses the most interesting challenge since, despite all the work that has been done trying to control hypersonic flow, no true effort has appeared to pose this problem from the point of view of nonlinear programming. The methodology presented in this dissertation focuses on mitigating this gap.

2.4 Magnetohydrodynamic analysis models

Continuum mechanics

The Knudsen number (Kn) is a dimensionless number defined as the ratio of the molecular mean free path length to a representative physical length scale,

$$Kn = \frac{\lambda}{L} = \frac{k_B T}{\sqrt{2}\pi\sigma^2 P L}\,, \tag{2.2}$$

where k_B is the Boltzmann's constant, T is the temperature, σ is the particle diameter and P is the total pressure. This parameter is useful for determining whether statistical mechanics or the continuum mechanics formulation of fluid dynamics should be used. If the Knudsen number is near or greater than one, the mean free path of a molecule is comparable to a length scale of the problem, the continuum assumption of fluid mechanics is no longer a good approximation, and a statistical model like direct simulation Monte Carlo has to be used. For Knudsen numbers considerably less than one, the fluid can be treated as a continuum where the field variables are considered averages over the microscopic behavior within an infinitesimal control volume. In this case, the traditional equations governing a flow field, such as the Navier–Stokes equations with no-slip boundary conditions, are valid.

In terms of aircraft or spacecraft flying at hypersonic speeds in the atmosphere, Sugimura showed that the minimum altitude at which the free-molecule flow assumption is valid is approximately 120 km and the flow-field can be characterized as near-continuum up until 70 km [151]. As such, the hypersonic flow simulations in this work will be restricted to the stratosphere, as pictured in figure 2.2, so that the Knudsen number remains everywhere small and the *continuous mechanics* assumption holds.

Complexity and completeness of MHD models

The analysis of plasma flows under the presence of magnetic fields is designated as magnetohydrodynamics (MHD) and the scientific community has already devoted much effort into its analysis.

In general, the set of governing equations that model MHD flows are obtained by

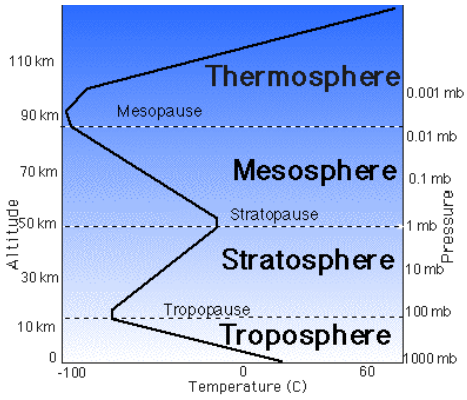


Figure 2.2: Regions of the atmosphere.

coupling the Navier–Stokes to the Maxwell equations. This coupling is strong, in the sense that the velocity field given by the former equations shows in the convective terms of the latter, and the electro-magnetic field modeled by the latter appears as additional force and energy terms in the former equations. The resulting equations are called the full MHD model [42].

Many interesting and important problems arise in astro-physical, solar, magnetospheric, and thermonuclear research which can be described by the MHD equations. The complexity of these problems often prohibits an analytical investigation and/or only some of the variables can be observed or measured experimentally; thus the researcher has to rely on numerical simulations.

In order to cope with the extreme complexity of the full MHD analysis, several simplified models have been developed. The problem might first be approached by ignoring the viscous effects and the heat transfer, as well as no electrostatic force, no displacement current and no resistivity. Under these assumptions, the MHD equations reduce to the ideal MHD formulation [9, 132]. Also, low magnetic Reynolds number approximations make the solution of the magnetic field unnecessary [130], since the magnetic field is assumed to be decoupled from the velocity field.

The completeness of the MHD models can be extended with the addition of turbulence models and chemical reactions. The inclusion of turbulence models, such as the Baldwin–Lomax and the κ-ϵ model, into ideal MHD formulations have been done by Dietiker and Hoffmann [33], and Gaitonde and Poggie [47, 50]. It is interesting to

notice that the former authors found evidence that the presence of a magnetic field decreases the skin friction in turbulent regions, suggesting that a re-laminarization process takes place. If chemical reactions are taken into account the chemical model of the reacting gas has to be included in the governing equations, which dramatically increases the computational cost, as mentioned in section 2.2. This has been shown in the work of Damevin and Hafmann [29], where a complete comparison of the ideal MHD and the Euler solutions, considering frozen, equilibrium and non-equilibrium chemical states has been done.

Another good example of the increasingly added accuracy and difficulty in analyzing the flow at hypersonic speeds with magnetic effects can be seen from MacCormack's work [90, 91, 92, 93], in which the two-dimensional ideal MHD approach evolved into a three-dimensional viscous model with chemical reactions, dealing with both external and internal flows.

In this dissertation, the fluid flow under the influence of magnetic fields has been modeled with both the ideal and low Re_σ approximation MHD equations. These are described in detail in sections 4.1 and 4.2, respectively.

Numerical schemes

In many situations, MHD flows develop features, such as steep gradients, shock waves, contact discontinuities, and shear layers, that require the use of modern high resolution numerical schemes to be resolved.

Many of the well established numerical schemes were initially developed for aerodynamics [69] and only recently have they been extended to the system of MHD equations. The main reason for this delay is that the structure of the MHD equations is much more complex so those schemes become rather complicated and often need modifications to handle accurately the degeneracies and instabilities of the MHD equations.

The stringent stability criteria and the need for efficient algorithms led to the development of modern shock capturing schemes based on upwind methods. These algorithms take into account the direction of signal propagation for the purpose of splitting

and differencing the inviscid fluxes. Among the many upwind schemes, the most popular are based on the Roe's approximate linearized Riemann solver [134, 132] and on the flux splitting techniques of Steger–Warming or Van Leer [156, 40, 41, 80]. While the Riemann solvers usually produce sharp and monotonic profiles, they often require entropy fixes and switches to other methods. Other widely used schemes include the Jameson–Schmidt–Turkel [78] and the MacCormack [88] differencing schemes.

The most common higher-order extension of upwind schemes are the Total Variation Diminishing (TVD) schemes and these can be extended to MHD equations as well. If an upwind MHD numerical scheme satisfies the TVD conditions, it enables a stable computation with high-resolution and effective shock capturing, with sharper profiles, while avoiding spurious oscillations. The Monotonic Upwind Schemes for conservation Laws (MUSCL) approach efficiently constructs TVD schemes from Roe's or flux-splitting schemes. A thorough comparison of some Flux Corrected Transport (FCT) and TVD Numerical schemes for MHD problems is shown in the work of Tóth and Odstrčil [155].

In 1988, Brio and Wu [20] showed a method to construct an upwind finite-difference scheme for the one-dimensional ideal MHD equations adopting Roe's linearization procedure. Four year later, Tanaka [153] extended it to three dimensions, using a finite-volume method based on the conservation laws, by recognizing that the ideal MHD equations are symmetric with the rotation of the space. Shang and Gaitonde [146] extended Van Leer's MUSCL approach to higher-order accuracy using an upwind-biased third-order scheme in 1993

Later, in 2001, Gaitonde proposed a high-resolution scheme for the 3-D full MHD equations using 4^{th}- and 6^{th}-order compact (or Padé-type) schemes [39, 42].

As far as time integration schemes are concerned, the explicit classical multi-stage Runge–Kutta (RK) schemes are the most widespread, even if the time step is restricted by the Courant-Friedrichs-Lewy (CFL) condition (4.41), because of their straightforward implementation. However, this time-step restriction gets even worse for the system of MHD equations since its fastest propagation wave, the fast magneto-acoustic wave (see figure B.1), is even faster than the conventional acoustic wave,

which translates to smaller allowable time steps as the magnetic field intensity increases. Examples of its use can be found in the work of Powell *et al.* [133], where upwind schemes based on an approximate Riemann solver for MHD are integrated in time using an explicit, two-stage, RK time-stepping scheme.

The alternative to explicit schemes is the use of implicit schemes to overcome time-step limitations, at the expense of a significantly more elaborate programming implementation. MacCormack has been a long time proposer of implicit discretization schemes using approximate factorization procedures and has already successfully applied these to MHD problems [87, 89]. Gaitonde and Poggie [47] have developed a loosely coupled, approximately-factored, implicit method, also to overcome time-step limitations of explicit classical Runge–Kutta schemes.

When selecting a numerical scheme among all the currently available ones, properties like robustness, numerical diffusion, production of spurious oscillations and computational efficiency should be taken into account. Regardless, the fact is that the complexity of some of the latest MHD schemes make their implementation extremely, if not prohibitively, difficult for the non-specialized researcher.

Chapter 3

Sensitivity analysis methods

The designation *sensitivity analysis* refers to the estimation of the first directional derivative of one or more functions with respect to one or more independent variables.

This field of study becomes relevant when we recall the optimization problem (1.3). As mentioned in section 1.2, the solution of a design problem using a gradient-based optimizer requires to evaluation of the sensitivity of both the objective and constraint functions with respect to the design variables, as depicted in figure 1.1. Let f be a generic scalar quantity representing the function of interest that is function of a set of parameters x_i,

$$f = f(x_i)\,, i = 1, \ldots, N_x\,, \tag{3.1}$$

where N_x is the number of parameters. Then, it becomes necessary to calculate the derivatives

$$\frac{df}{dx_i}\,, i = 1, \ldots, N_x\,. \tag{3.2}$$

More often than not, these calculations can be the costliest step in the optimization cycle. As such, it is extremely important to devote special care to the way this step is achieved, taking into account not only the number of design variables, N_x, we might be interested in but also how expensive it is to evaluate the function f and how important the accuracy of the estimate is.

Several methods are available to estimate the derivatives (3.2) and they are revisited in the following sections, together with their respective advantages, disadvantages

and applicability. Further details about these standard methods can be found in the book by Griewank [60].

This chapter ends with the introduction of a hybrid method, the *ADjoint* method, that addresses the requirements to accomplish the targeted dissertation goals.

3.1 Finite-differences

The most widely used method to compute derivatives is the finite-differencing (FD) method. This method makes use of the Taylor's series expansion of the function f about a given point x,

$$f(x+\Delta x) = f(x) + \Delta x \frac{\partial f(x)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f(x)}{\partial x^2} + \cdots + \frac{\Delta x^n}{n!}\frac{\partial^n f(x)}{\partial x^n} + \cdots , \tag{3.3}$$

where Δx is the perturbation step, to derive formulas for derivatives with arbitrary order of accuracy.

For instance, solving equation (3.3) for the first derivative results in the 1^{st}-order forward-difference formula,

$$\frac{\partial f}{\partial x} \approx \frac{f(x+\Delta x) - f(x)}{\Delta x} + \mathcal{O}(\Delta x)\,. \tag{3.4}$$

If the expansion (3.3) is done for $(x - \Delta x)$, the resulting formula is the backward-difference formula. On the other hand, a 2^{nd}-order accurate formula can be obtained by writing the expansion for both $(x + \Delta x)$ and $(x - \Delta x)$, and combining the two. The result is the central-difference formula,

$$\frac{\partial f}{\partial x} \approx \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + \mathcal{O}(\Delta x^2)\,. \tag{3.5}$$

A detailed look at these approximations, as well as higher-order approximations, can be found in Lomax *et al.* [85].

From the above expressions, it follows immediately that the implementation of this method is very easy, thus its wide-spread use, as it only requires the function evaluation capability. This is usually embedded in the flow solver itself so all is left

is to modify the input (or boundary) flow conditions to account for the perturbed design variable, corresponding to the step $x_i + \Delta x_i$, and re-evaluate the flow solution. If either a 1^{st}-order forward- or back-difference was selected, it would require $N_x + 1$ flow solver evaluations to compute the full gradient vector, whereas the 2^{nd}-order estimate (3.5) would need as many as $2N_x + 1$. This is to say that the computational cost of this method is proportional to the number of design variables.

However, this need for flow re-evaluations makes this method impractical for problems with a large number of design variables, since the large number (N_x) of expensive function evaluations translates into a prohibitively costly computational process, thus inappropriate for the intent of this dissertation.

Moreover, this method also suffers from a large sensitivity to the choice of step size. If Δx is chosen too large, the derivative estimate might be inaccurate because of the large truncation error; if it is made too small, then subtractive cancellation might occur and the estimate is again inaccurate. A thorough discussion about subtractive cancellation issues can be found in the work of Martins [104]. Finding the sweet spot of Δx is often problem dependent so prior effort expended before using the resulting estimated derivative values.

3.2 Complex-step derivative

The complex-step derivative approximation can also be derived using a Taylor series expansion, similar to the finite-difference approximation, but instead of using a real perturbation step, it uses a pure imaginary step.

Even though it was pioneered back in the late 60's by Lyness and Moler [86], only recently has this method been rediscovered by Squire and Trapp [149] and used to obtain a very simple formula for estimating the first derivative of a real function. One of the first applications to CFD happened just at the turn of the 21^{st} century in the work of Martins *et al.* [109], who performed sensitivity analysis on a two-dimensional, cell-centered, finite-volume solver for the Euler equations. Around the same time, Anderson *et al.* [7] carried out sensitivity analysis for the three-dimensional Navier-Stokes equations.

The first derivative formula can be easily recovered by considering a real function, f, and expand it about a real point x using the Taylor's series expansion (3.3),

$$f(x + i\Delta x) = f(x) + i\Delta x \frac{\partial f(x)}{\partial x} - \frac{\Delta x^2}{2!} \frac{\partial^2 f(x)}{\partial x^2} + \cdots + \frac{(i\Delta x)^n}{n!} \frac{\partial^n f(x)}{\partial x^n} + \cdots , \quad (3.6)$$

where the imaginary unit is defined as $i = \sqrt{-1}$, according to complex calculus. The function f has now become complex valued so taking the imaginary parts of both sides of expression (3.6), dividing it by Δx and solving for the first derivative, yields the desired expression,

$$\frac{\partial f}{\partial x} \approx \frac{\mathrm{Im}[f(x + i\Delta x)]}{\Delta x} + \mathcal{O}(\Delta x^2) \, . \quad (3.7)$$

Comparing the approximation (3.7) to the FD central-difference formula (3.5), it can be seen that even though it retains 2^{nd}-order accuracy, it only requires $N_x + 1$ complex function evaluations.

This estimate is not subject to subtractive cancellation error, since it does not involve a difference operation, which constitutes a tremendous advantage over the finite-difference approximations (3.4) and (3.5). In fact, the derivative estimate is practically insensitive to small step sizes so they can be made extremely small, where the lower limit depends only on the type of finite-precision arithmetic being used.

To implement this method in a flow solver coded in a language that supports complex-arithmetic, it is necessary to make a few changes to the code, namely substitute all real type variable declarations with complex declarations and define all functions and operators that are not defined for complex arguments. Only then can a desired input variable be perturbed by a complex-step and the derivative estimated by the formula (3.7). A point worth noting is that the complex-step method is equivalent to the forward-mode of automatic differentiation using operator overloading [111].

This method has already been proved to be very accurate, extremely robust and surprisingly easy to implement in design problems [112]. Nevertheless, the cost of estimating the derivative using this method is still proportional to the number of design variables, which is further aggravated by the fact that the run time of the

complex-valued code might take up to three times longer to run when compared to the original real-valued code. Consequently, this method is also unsuited for the class of design problems targeted by this dissertation.

3.3 Symbolic differentiation

The use of a symbolic differentiation commercial packages such as *Maple* [161], *Matlab*(with Maple's symbolic toolbox) [162] or *Mathematica* [163] can solve the problem of obtaining expressions for the derivatives. This method avoids truncation errors but usually these packages have problems in handling large expressions and the time/space usage for computing derivatives can be enormous. In the worst case it can even cause the program to crash. Furthermore, common sub-expressions are usually not identified in the expressions and this leads to unnecessary computations during the evaluation of the derivatives.

As an illustration, given the vector function $\boldsymbol{f}$,

$$\left\{ \begin{array}{c} f_1 \\ f_2 \end{array} \right\} = \left\{ \begin{array}{c} 2x_2(4x_1 + 3cos(x_2)) \\ (x_1 + 5x_2^2)/2 \end{array} \right\}, \tag{3.8}$$

the exact Jacobian (matrix of sensitivities) can be easily calculated, resulting

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \left[\begin{array}{cc} 8x_2 & 8x_1 + 6cos(x_2) - 6x_2 sin(x_2) \\ 1/2 & 5x_2 \end{array} \right]. \tag{3.9}$$

Using *Maple* to perform the task through symbolic manipulation produces the results shown in figure 3.1. As expected, the results match the analytic expressions (3.9). However, it is easy to understand that the use of such tools is restricted explicit functions of low dimensionality and they cannot be applied to large iterative solvers, such as the ones employed in CFD, to compute the gradients of the form (3.2).

```
> with(linalg):

Warning, the protected names norm and trace have been redefined and
unprotected
> x := vector( [x1, x2] );
> f := vector( [2*x2*(4*x1+3*cos(x2)), (x1+5*x2^2)/2 ] );
```

$$x := [x1,\ x2]$$

$$f := \left[2\,x2\,(4\,x1 + 3\cos(x2)),\ \frac{1}{2}\,x1 + \frac{5}{2}\,x2^2\right]$$

```
> df_dx := jacobian(f, x);
```

$$df\_dx := \begin{bmatrix} 8\,x2 & 8\,x1 + 6\cos(x2) - 6\,x2\sin(x2) \\ \frac{1}{2} & 5\,x2 \end{bmatrix}$$

Figure 3.1: Symbolic differentiation using *Maple*.

3.4 Automatic differentiation

Automatic differentiation, also known as computational or algorithmic differentiation, is a sensitivity analysis method based on the systematic application of the chain rule of differentiation to computer programs. One of the implementations of the method relies on tools that automatically produce a program that computes user-specified derivatives based on the original program.

The sequence of operations in any computational algorithm can be cast in the form

$$t_i = f_i(t_1, t_2, \ldots, t_{i-1})\,, \quad i = n+1, n+2, \ldots, m\,, \tag{3.10}$$

where each function f_i is either a unary or binary operation. $t_1, t_2, \ldots, t_n$ are the *independent* variables, which in this work assume the role of design variables $\boldsymbol{x}$, and $t_{n+1}, t_{n+2}, \ldots, t_m$ are the *dependent* variables, that include all the intermediate variables in the algorithm, among which we can find the outputs of interest, I. Applying the chain rule to the algorithm (3.10) yields

$$\frac{\partial t_i}{\partial t_j} = \sum_{k=1}^{i-1} \frac{\partial f_i}{\partial t_k}\frac{\partial t_k}{\partial t_j}\,, \quad j = 1, 2, \ldots, n\,. \tag{3.11}$$

The result is an automatically generated new code, corresponding to the differentiated

version of the original one.

Although this approach is as accurate as an analytic method, it is potentially much easier to implement since everything is done automatically. However, the run time of the algorithmic differentiated version compared with the real-valued code is approximately two times longer.

3.4.1 Forward and reverse modes

There are two different modes of operation for automatic differentiation – the forward and the reverse modes – depending on how the derivatives given by the chain rule (3.11) are propagated.

The forward mode propagates the required sensitivity at the same time as the solution is being computed. In terms of index notation, one independent variable is selected, choosing j and keeping it fixed, and then the expression is worked forward in the index i until the desired derivative is obtained. Thus, this mode is well-suited when evaluating the sensitivity of many functions with respect to one parameter.

On the other hand, the reverse mode requires the function to be computed first, with intermediate variable values stored. These intermediate variables are then used by the reverse version of the code to find the sensitivities. This mode works by fixing i, corresponding to the desired output to be differentiated, and working the way backward in the index j all the way down to the independent variables. As such, it is the desired mode to compute the sensitivity of one function with respect to many parameters.

Further details about the two different modes of differentiation, illustrated with examples, can be found in the work of Mader *et al.* [95].

In the context of CFD, the code that evaluates the function of interest, f, is typically an iterative solver. The two modes are directly related to the direct and adjoint methods of sensitivity analysis. The counterparts of the state variables in the semi-analytic methods are the intermediate variables, and the residuals are the lines of code that compute those quantities. As such, the use of AD in reverse mode requires the solver to be run to convergence first, with intermediate variable values

stored for every iteration. Consequently, the memory requirements easily become prohibitive because the flow solvers generally require a large number of iterations to achieve convergence.

3.4.2 Source transformation or operator overloading

There are two methods for implementing automatic differentiation – *source code transformation* and *operator overloading*.

The automatic differentiation implementation using source transformation requires the whole source code to be processed by a parser. The parser introduces additional lines of code corresponding to all the derivative calculations while generating the differentiated version of the source code. Therefore, the resulting code is greatly enlarged and it becomes practically unreadable. This fact might constitute an implementation disadvantage as it becomes impractical to debug this new extended code. Also, every time the original code is changed, it is necessary to re-run the parser to obtain the updated corresponding differentiated version. Despite the fact that this method generates new code that is unmaintainable, it brings the important advantage that it tends to yield considerably faster code.

The automatic differentiation implementation using operator overloading defines a new user-defined type that is used instead of real numbers. This new type includes not only the value of the original variable, but the derivative as well. Therefore, to use this method, it is required that the source code be written in a modern programming language, such as Fortran 90 or C++, that supports derived data types. All the intrinsic operations and functions have to be redefined (overloaded) for the new type in order for the derivative to be computed together with the original computations. The new operator has exactly the same behavior as before for the value part of the new type, but uses the definition of the derivative of the operator to calculate the derivative portion. This results in a very elegant implementation since very few changes are required in the original code, but it is usually less efficient.

3.4.3 Tools for automatic differentiation

There are a number of software tools available for automatic differentiation, supporting different programming languages and implementing either the source code or the operator overloading methods. These tools have been extensively developed and provide the user with great functionality, including the calculation of higher-order derivatives and reverse mode options.

The most well known tools that automatically differentiate Fortran 77 codes using the source transformation method are ADIFOR [17, 18, 21], TAMC [58, 52], DAFOR [15, 14], GRESS [72, 71] and Odyssée [38]. More recently, some tools that support Fortran 90 have been made available, namely, TAF [51, 53](the sucessor of TAMC) and Tapenade [64, 65, 32](the sucessor of Odyssée). All these tools make the necessary changes to the source code automatically.

Tools supporting the operator overloading method are AD01 [135], ADOL-F [147], IMAS [141, 140] and OPTIMA90 [12]. All these require at least Fortran 90 so that the operator overloading feature can be used. Although it is, in theory, possible to have a script make the necessary changes in the source code automatically, none of these tools have this facility and the changes must be done manually.

There are also a few established automatic differentiation tools for C/C++. These include ADIC [16], an implementation mirroring ADIFOR also using the source transformation method, and FADBAD++ [13, 150] and ADOL-C [59] (the C counterpart of ADOL-F), that use the operator overloading method.

3.4.4 Example using Tapenade

Among the several tools for automatic differentiation mentioned above, Tapenade was chosen in this dissertation because it is the only non-commercial tool that supports Fortran 90, which was a requirement taking into account the programming language used in the flow solver implementations.

Tapenade is the successor of Odyssée and it has been developed at the *Institut National de Recherche en Informatique et an Automatique* (INRIA) in Sophia-Antipolis,

France. It uses source transformation and can perform differentiation in either forward or reverse mode. Furthermore, the Tapenade team is actively developing their software and has been very responsive to a number of suggestions toward completing their full support of the Fortran 90 standard. At the time this dissertation was written, the major restriction when using Tapenade was that it did not support pointer variables in the path of differentiation. However, this drawback can be easily circumvented by simply copying the relevant variables instead of pointing to them. Also, variables that remain constant within the section of code that is differentiated do not present this problem.

Similarly to what was done for the symbolic differentiation tool subsection, the same example (3.8) is used here, running Tapenade in both the forward and reverse modes to compute the derivatives. In this case, there are two independent variables (n=2) – t_1 and t_2. The Fortran routine that computes the vector function $\boldsymbol{f}$ is shown in figure 3.2.

```
subroutine myResidual(t1,t2,t5,t6)

real(8), intent(in)  :: t1, t2
real(8), intent(out) :: t5, t6
real(8)              :: t3, t4

t3 = 4 * t1 + 3 * cos(t2)
t4 = t1 + 5 * t2**2
t5 = 2 * t2 * t3
t6 = t4 / 2

end subroutine
```

Figure 3.2: Original Fortran routine.

Running the Tapenade parser in forward mode of differentiation automatically generates the code included in figure 3.3. New variables are introduced, `tjd, j=1,2`, which are the seed vector and `tid, i=5,6`, which are the gradient of all `ti` in the direction specified by the seed vector. The seed `tjd` determines the independent variable t_j with respect to which the gradient of $\boldsymbol{f}$ is to be computed. Choosing `j=1,2` and setting the corresponding seed to unit value `tjd=1` (`tkd=0, k`$\neq$`j`), then

```
C        Generated by TAPENADE     (INRIA, Tropics team)
C  Tapenade - Version 2.2 (r1239) - Wed 28 Jun 2006 04:59:55 PM CEST
C
C  Differentiation of myresidual in forward (tangent) mode:
C   variations  of output variables: t5 t6
C   with respect to input variables: t1 t2
      SUBROUTINE MYRESIDUAL_D(t1, t1d, t2, t2d, t5, t5d, t6, t6d)
      IMPLICIT NONE
C
      REAL*(IN)*(8) t1, t1d, t2, t2d
      REAL*(OUT)*(8) t5, t5d, t6, t6d
      REAL*8 t3, t3d, t4, t4d
      INTRINSIC COS
C
      t3d = 4*t1d - 3*t2d*SIN(t2)
      t3 = 4*t1 + 3*COS(t2)
C
      t4d = t1d + 5*2*t2*t2d
      t4 = t1 + 5*t2**2
C
      t5d = 2*(t2d*t3+t2*t3d)
      t5 = 2*t2*t3
C
      t6d = t4d/2
      t6 = t4/2
C
      END
```

Figure 3.3: Fortran routine automatically differentiated using the forward mode.

the differentiated routine outputs the gradients of all the outputs with respect to t_j,

$$\frac{\partial t_i}{\partial t_j} = \texttt{tid}, \texttt{i} = 5, 6\,. \tag{3.12}$$

For example, the derivative with respect to t_1, we would set $t1d = 1$ and all other `tjd`s to zero. Notice that only one direction can be chosen at a time.

If Tapenade is run in reverse mode of differentiation, it produces the differentiated code shown in figure 3.4. In this case, selecting the output variable `i=5,6` and setting the corresponding seed to unit value `tib=1` (`tkb=0, k`$\neq$`i`), the new code would

```
C        Generated by TAPENADE     (INRIA, Tropics team)
C  Tapenade - Version 2.2 (r1239) - Wed 28 Jun 2006 04:59:55 PM CEST
C
C  Differentiation of myresidual in reverse (adjoint) mode:
C   gradient, with respect to input variables: t1 t2 t5 t6
C   of linear combination of output variables: t5 t6
      SUBROUTINE MYRESIDUAL_B(t1, t1b, t2, t2b, t5, t5b, t6, t6b)
      IMPLICIT NONE
C
      REAL*(IN)*(8) t1
      REAL*8 t1b, t2b
      REAL*(IN)*(8) t2
      REAL*8 t5, t5b, t6, t6b
      REAL*8 t3, t3b, t4, t4b
      INTRINSIC COS
C
      t3 = 4*t1 + 3*COS(t2)
C
      t4b = t6b/2
      t3b = 2*t2*t5b
      t2b = 5*2*t2*t4b - 3*SIN(t2)*t3b + 2*t3*t5b
      t1b = 4*t3b - t4b
      t5b = 0.0
      t6b = 0.0
C
      END
```

Figure 3.4: Fortran routine automatically differentiated using the reverse mode.

compute the gradient of the selected output with respect to all independent variables,

$$\frac{\partial t_i}{\partial t_j} = \mathtt{tjb}, \mathtt{j} = 1, 2\,. \tag{3.13}$$

The cost of calculating the derivative of one output to many inputs is not proportional to the number of inputs but, rather, to the number of outputs. Since, when using the reverse mode, all the intermediate variables need to be stored, the amount of memory that is necessary increases dramatically. In the case of three-dimensional iterative CFD solver, the cost of using this mode can be prohibitive.

In summary, the application of AD tools directly to the flow solver generates differentiated code that becomes too expensive to be evaluated. A good example of the application of AD tools directly to flow solvers can be found within the German

MEGADESIGN project [82], where TAF was used to differentiate *FLOWer*, a finite-volume, block-structured, flow solver for the RANS equations. The resulting differentiated solver took up to ten times longer to run than original flow solver. Therefore, the AD method does not provide, by itself, the features necessary to achieve the goals of this dissertation.

3.5 Semi-analytic methods

Among all the methods available for sensitivity analysis, the semi-analytic methods are the most accurate and efficient. They are, however, more involved than the other methods since they require the knowledge of the governing equations and the algorithm that is used to solve those equations.

In this case, it is convenient to express the function of interest as I and recognize that it depends, in general, not only on the design variables $\boldsymbol{x}$ themselves directly, but also on the physical state $\boldsymbol{w}$ of the system,

$$I = I(\boldsymbol{x}, \boldsymbol{w}(\boldsymbol{x}))\,. \tag{3.14}$$

For a given set of parameters, $\boldsymbol{x}$, the solution of the governing equations yields a solution $\boldsymbol{w}$, thus establishing the dependence of the state of the system on the design variables. The governing equations can be posed as

$$\mathcal{R}(\boldsymbol{x}, \boldsymbol{w}(\boldsymbol{x})) = 0\,, \tag{3.15}$$

where the first instance of $\boldsymbol{x}$ indicates the fact that the residual of the governing equations may depend explicitly on design variables. The discretization of the governing equations results in a system of ODEs, as detailed in chapter 4, which is here represented by equation (3.15).

As a first step toward obtaining the derivatives (3.2), it is convenient to use the chain-rule of differentiation to write the total sensitivity of the vector-valued function

I as

$$\frac{dI}{d\boldsymbol{x}} = \frac{\partial I}{\partial \boldsymbol{x}} + \frac{\partial I}{\partial \boldsymbol{w}} \frac{d\boldsymbol{w}}{d\boldsymbol{x}} \,. \tag{3.16}$$

The sizes of the sensitivity matrices are

$$\frac{\partial I}{\partial \boldsymbol{x}} \quad (N_I \times N_x) \,, \qquad \frac{\partial I}{\partial \boldsymbol{w}} \quad (N_I \times N_w) \,, \qquad \frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{x}} \quad (N_w \times N_x) \,, \tag{3.17}$$

where N_I is the number of functions of interest, N_x the number of design variables and N_w the size of the state vector, which for the solution of a large, three-dimensional problem involving a system of conservation laws, can be very large. The size of the state vector depends on the number of governing equations, N_v, and the size of the computational mesh, N_c, in each they have been discretized, according to the relation $N_w = N_v \times N_c$.

It is important to distinguish the total and partial derivatives in equation(3.16). The partial derivatives can be directly evaluated by varying the denominator and re-evaluating the function in the numerator *with everything else held constant*. The total derivatives, however, require the solution of the governing equations. Thus, for typical functions of interest, all the terms in the total sensitivity equation (3.16) can be computed with relatively little effort except for $\frac{d\boldsymbol{w}}{d\boldsymbol{x}}$.

Since the governing equations must always be satisfied, the total derivative of the residuals (3.15) with respect to any design variable must also be zero. Expanding the total derivative of the governing equations with respect to the design variables results

$$\frac{d\mathcal{R}}{d\boldsymbol{x}} = \frac{\partial \mathcal{R}}{\partial \boldsymbol{x}} + \frac{\partial \mathcal{R}}{\partial \boldsymbol{w}} \frac{d\boldsymbol{w}}{d\boldsymbol{x}} = 0 \,. \tag{3.18}$$

This expression provides the means for computing the total sensitivity of the state variables with respect to the design variables, $\frac{d\boldsymbol{w}}{d\boldsymbol{x}}$. To this end, rewriting equation (3.18) as

$$\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}} \frac{d\boldsymbol{w}}{d\boldsymbol{x}} = -\frac{\partial \mathcal{R}}{\partial \boldsymbol{x}} \,, \tag{3.19}$$

where the following sensitivity matrices have been defined,

$$\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}} \quad (N_w \times N_w) \,, \qquad \frac{\partial \mathcal{R}}{\partial \boldsymbol{x}} \quad (N_w \times N_x) \,, \tag{3.20}$$

solving for $\frac{d\boldsymbol{w}}{d\boldsymbol{x}}$, and substituting the result into the total derivative equation (3.16), yields

$$\frac{dI}{d\boldsymbol{x}} = \frac{\partial I}{\partial \boldsymbol{x}} - \frac{\partial I}{\partial \boldsymbol{w}} \left[\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}\right]^{-1} \frac{\partial \mathcal{R}}{\partial \boldsymbol{x}} \,. \tag{3.21}$$

Depending on the problem size – number of functions of interest, N_I, and number of design variables, N_x –, the evaluation of the expression for the total derivative (3.21) may be tackled in two different ways, leading to either the direct- or the adjoint-sensitivity equations. These methods are covered in the sub-sections that follow.

3.5.1 Direct method

If the direct method is used, the equation (3.19) is solved first for $\frac{d\boldsymbol{w}}{d\boldsymbol{x}}$ and then the result is substituted in the expression for the total sensitivity (3.16). The resulting direct sensitivity equations are then given by

$$\begin{aligned} &\frac{dI}{d\boldsymbol{x}} = \frac{\partial I}{\partial \boldsymbol{x}} + \frac{\partial I}{\partial \boldsymbol{w}} \frac{d\boldsymbol{w}}{d\boldsymbol{x}} \,, \\ &\text{such that} \quad \frac{\partial \mathcal{R}}{\partial \boldsymbol{w}} \frac{d\boldsymbol{w}}{d\boldsymbol{x}} = -\frac{\partial \mathcal{R}}{\partial \boldsymbol{x}} \,. \end{aligned} \tag{3.22}$$

This method implies that the matrix equation (3.19) has to be solved N_x times for $\frac{d\boldsymbol{w}}{d\boldsymbol{x}}$, one for each design variable x_i. Since the design variables only affect the right-hand side of the equation, solving for multiple right-hand-side vectors would be relatively inexpensive if the matrix $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ could be explicitly factorized and stored.

However, for large problems – such as the ones encountered in CFD – the inverse of the Jacobian matrix $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ is never factorized explicitly and the system of equations requires an iterative solution which is usually as costly as solving the governing equations.

Consequently, the factorization cost, in conjunction with a large number of design variables, makes the total cost for calculating the total sensitivity vector (3.16) unacceptable. In other words, this method is not suitable for achieving the goal statement of this dissertation.

3.5.2 Adjoint method

An alternative method for computing the total sensitivity $\frac{dI}{d\boldsymbol{x}}$ can be derived by returning to the total sensitivity equation (3.21) and defining an auxiliary vector ψ as

$$\psi^T = \frac{\partial I}{\partial \boldsymbol{w}} \left[\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}\right]^{-1} \quad (N_I \times N_w) \;, \tag{3.23}$$

which upon rearrangement yields the adjoint equations,

$$\left[\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}\right]^T \psi = \left[\frac{\partial I}{\partial \boldsymbol{w}}\right]^T \;. \tag{3.24}$$

The vector ψ is usually called the adjoint vector and it is substituted into equation (3.21) to find the total sensitivity. The resulting adjoint sensitivity equations, also called the *dual problem*, are then given by

$$\begin{aligned} &\frac{dI}{d\boldsymbol{x}} = \frac{\partial I}{\partial \boldsymbol{x}} - \psi^T \frac{\partial \mathcal{R}}{\partial \boldsymbol{x}} \,, \\ &\text{such that} \quad \left[\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}\right]^T \psi = \left[\frac{\partial I}{\partial \boldsymbol{w}}\right]^T \;. \end{aligned} \tag{3.25}$$

In contrast with the direct method, the adjoint vector does not depend on the design variables, $\boldsymbol{x}$, but instead depends on the function of interest, I.

As mentioned before, the choice of the solution procedure (direct vs. adjoint) to obtain the total sensitivity (3.21) has a substantial impact on the cost of sensitivity analysis. Although all the partial derivative terms are the same for both the direct and adjoint methods, the order of the operations is not.

The most computationally intensive step for both of these problems (3.22,3.25) is the solution of the corresponding linear systems. In the case of the original problem (3.22) – the *direct method* – a linear system of N_w equations has to be solved N_x times. Notice that once $\frac{d\boldsymbol{w}}{d\boldsymbol{x}}$ is computed, it is valid for any function I, but must be recomputed for each design variable $\boldsymbol{x}$. For the dual problem (3.25) – the *adjoint method* – the linear system has the same size, however, it has to solved N_I times because ψ is valid for all design variables, but must be recomputed for each function.

Thus the choice of which of these methods to use depends largely on how the number of design variables, N_x, compares to the number of functions of interest, N_I.

A comparison of the cost of computing sensitivities with the direct versus adjoint methods is shown in table 3.1. Both methods require the factorization of the same

Step	Direct	Adjoint
Factorization	same	same
Back-solve	N_x times	N_I times
Multiplication	same	same

Table 3.1: Approximate comparison of the relative cost of the semi-analytic methods.

matrix, $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$. The difference in the cost comes form the back-solve step for solving equations (3.19) and (3.24), respectively. The direct method requires this step to be performed for each design variable, x_i, while the adjoint method requires this to be done for each function of interest, I. The multiplication step is simply the calculation of the final sensitivity expressed in equations (3.19) and (3.24), respectively, and its cost is the same for both methods. Consequently, when the number of design variables is greater than the number of functions of interest, $N_x \gg N_I$, the adjoint approach of equation (3.25) is the logical choice. If, instead, the number of functions to be differentiated is greater than the number of design variables, $N_I \gg N_x$, then the direct method should be used.

The cost involved in calculating sensitivities using the adjoint method is therefore practically independent of the number of design variables. After having solved the governing equations, the adjoint equations are solved only once for each I. Moreover, the cost of solution of the adjoint equations is similar to that of the solution of the governing equations since they are of similar size and complexity. The major drawback of using adjoint-based sensitivities has always been the necessity of an additional solver – the adjoint system of equations solver. Therefore, the adjoint approach enables large computational savings, at the expense of a more complex implementation [105, 104], when compared to traditional sensitivity analysis methods such as finite differences.

This method is by far the best suited for this work and, as such, it will constitute the foundation upon which the methodology to develop, in a timely fashion, an efficient sensitivity analysis tool for an arbitrarily complex flow solver will be built. The issue of rapidly developing the adjoint solver will be tackled in section 3.6 and its implementation covered in detail in chapter 5.

Duality viewpoint

The derivation of the adjoint equations found in the previous subsection is often referred as the dual problem, since the adjoint vector takes the role of the dual of the state vector. In that sense, the set of governing equations is called the primal problem, whereas the adjoint system of equations is referred as the dual problem.

Lagrangian viewpoint

An alternative way to derive the adjoint equations is the Lagrangian viewpoint. This approach follows the method of Lagrange multipliers for the solution of a constrained minimization problem, in which the augmented cost function is defined as

$$\tilde{I}(\boldsymbol{w}, \boldsymbol{x}) = I(\boldsymbol{w}, \boldsymbol{x}) - \psi^T \mathcal{R}(\boldsymbol{w}, \boldsymbol{x}), \tag{3.26}$$

where ψ is the vector of Lagrange multipliers that takes the role of the adjoint variables. From the definition of augmented cost function, the constraints are naturally enforced by the optimal solution, thus the governing equations are automatically satisfied.

The sensitivity of the augmented cost function is then

$$\frac{d\tilde{I}}{d\boldsymbol{x}} = \frac{dI}{d\boldsymbol{x}} - \psi^T \frac{d\mathcal{R}}{d\boldsymbol{x}}\,, \tag{3.27}$$

which can be expanded as

$$\frac{d\tilde{I}}{d\boldsymbol{x}} = \left(\frac{\partial I}{\partial \boldsymbol{w}}\frac{d\boldsymbol{w}}{d\boldsymbol{x}} + \frac{\partial I}{\partial \boldsymbol{x}}\right) - \psi^T \left(\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}\frac{d\boldsymbol{w}}{d\boldsymbol{x}} + \frac{\partial \mathcal{R}}{\partial \boldsymbol{x}}\right)\,. \tag{3.28}$$

Rearranging equation (3.28) results

$$\frac{d\tilde{I}}{d\boldsymbol{x}} = \left(\frac{\partial I}{\partial \boldsymbol{w}} - \psi^T \frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}\right) \frac{d\boldsymbol{w}}{d\boldsymbol{x}} + \left(\frac{\partial I}{\partial \boldsymbol{x}} - \psi^T \frac{\partial \mathcal{R}}{\partial \boldsymbol{x}}\right) . \tag{3.29}$$

In order to eliminate the dependence on the flow variables, the term involving $d\boldsymbol{w}/d\boldsymbol{x}$ must vanish, which is achieved by choosing ψ such that it satisfies the adjoint equation

$$\frac{\partial I}{\partial \boldsymbol{w}} - \psi^T \frac{\partial \mathcal{R}}{\partial \boldsymbol{w}} = 0 \qquad \Longrightarrow \qquad \left[\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}\right]^T \psi = \left[\frac{\partial I}{\partial \boldsymbol{w}}\right]^T . \tag{3.30}$$

Once ψ is obtained by solving the system of equations (3.30), the sensitivity of the augmented cost function is simply given by

$$\frac{d\tilde{I}}{d\boldsymbol{x}} = \frac{\partial I}{\partial \boldsymbol{x}} - \psi^T \frac{\partial \mathcal{R}}{\partial \boldsymbol{x}} . \tag{3.31}$$

The final equations (3.30) and (3.31) are exactly the same as those derived by considering duality (3.25); only the mathematical description differs.

The advantage of the adjoint approach can be seen from equation (3.31), which is independent of $\delta\boldsymbol{w}$, meaning that the gradient of I with respect to an arbitrarily large vector of design variables $\boldsymbol{x}$ can be determined without the need for additional solutions of the PDE. This capability to effectively handle design problems involving a large number of design variables is what makes the adjoint methods well known for.

Continuous and discrete approaches

When it comes to implementation, there are two main ways of obtaining the adjoint equations (3.25) for a given system of PDEs, as illustrated in figure 3.5. These two adjoint formulations can be classified into continuous [138, 139] or discrete [123, 121]. The *continuous adjoint approach* forms a continuous adjoint problem from the governing PDEs and then discretizes this problem to solve it numerically. The *discrete adjoint approach* first discretizes the governing PDE and then derives an adjoint system for these discrete equations. As such, there is freedom as how to discretize the adjoint PDE using the continuous approach, whereas the adjoint implementation

in the discrete approach is fixed by the primal discretization.

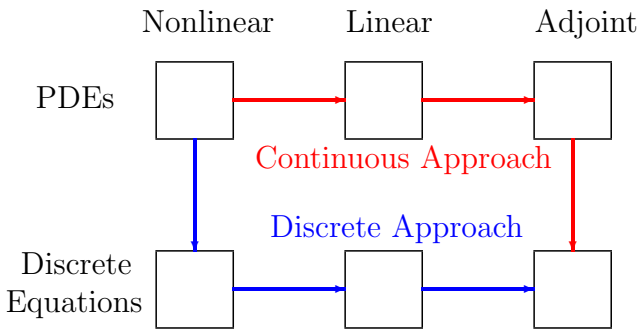


Figure 3.5: Ways of deriving the discretized adjoint equations.

Each of these approaches results in different systems of linear equations that, in theory, converge to the correct analytic value of the gradient of the objective function in the limit of infinite grid resolution, if all the solutions are sufficiently smooth. There have been some studies comparing these two approaches such those done by Nadarajah and Jameson [117, 118].

The discrete approach formulation has the advantage that it can be applied to *any* set of governing equations and, being derived from the discretized form of the flow governing equations, produces gradients that are consistent with the flow solver. If this is not guaranteed then a numerical optimizer cannot converge to a local minimum since the numerical gradient might not be zero at that location due to those inconsistencies. In addition, it can treat *arbitrary* functions of interest, whereas the continuous adjoint can only treat specific forms of integral functions [77, 84]. Another advantage of this formulation is that the boundary conditions are handled seamlessly since the adjoint solver is derived from the discretized flow residual equations that already implement them. But the most interesting feature is that it allows to use automatic differentiation (AD) tools in its derivation, expediting considerably the process of obtaining the differentiated form of the discretized governing equations necessary to assemble the adjoint system of equations.

Differentiating the continuous governing equations first is usually more involved and a number of shortcuts must be included when the governing equations are complicated, such as when using the RANS equations with turbulence models [118] or MHD

models. The viscous effects might not be modeled in their entirety, as in the reference just mentioned, and even when they are modeled (for purposes of the adjoint solver), the flow is typically assumed laminar and no turbulence model is used [77]. In the latter case, the viscosity and heat transfer ratio are usually assumed to be independent of the flow and kept constant when deriving the adjoint. This is true even if the flow solver uses the RANS equations because the derivation of the adjoint typically assumes a constant eddy viscosity thus ignoring the turbulence equations. In addition, applying boundary conditions to the differentiated equations can be non-intuitive because some of these boundary conditions are non-physical. The biggest feature might be the reduced memory requirements (at the same level as the flow solver), but all the other drawbacks when compared to the discrete approach make it inappropriate to meet the needs to achieve the previously stated goal of this dissertation.

When considering high-speed flows in the context of MHD, both turbulence and electromagnetic effects play an important role, and the variation of μ and κ, as well as the magnetic permeability μ_m and the electrical conductivity σ, must be taken into account to accurately model the phenomena. Since these variations can be naturally included in the discrete adjoint formulation, this approach seems to be the most suitable for the problem at hand. This is particularly important as not deriving the full adjoint can result in incorrect sensitivities, particularly for viscous flow, as shown in the previous work of Dwight [35].

Additional details about the continuous and discrete adjoint approaches and their implementations can be found in the work of Giles[55].

3.6 Hybrid approach: ADjoint

An interesting comparison of the computational accuracy and cost among the several sensitivity analysis methods described previously can be found in the work of Martins [103], conveniently replicated here in table 3.2. This table shows the gradient of a single function of interest with respect to a single input parameter evaluated using a finite-element structural model solver. Both the run-time and memory requirements of the complex-step solution were taken as references in the comparison to the other

methods.

Method	Sample Sensitivity	Time	Memory
FD	39.049724352820375	0.88	0.72
Complex	39.049760045804646	1.00	1.00
ADIFOR	39.049760045809059	2.33	8.09
Analytic	39.049760045805281	0.58	2.42

Table 3.2: Run-time and memory requirements for different sensitivity methods.

The data in table 3.2 shows that all methods except finite-differences achieve the precision of the solver. The complex-step approximation might seem a good compromise, but since its cost is proportional to the number of parameters (design variables), it has to be ruled out. The automatic differentiation method, implemented quite easily using ADIFOR, proved to be not only slower but mainly quite memory intensive. On the other hand, the analytic method – a discrete adjoint – looks as the best but its implementation is rather difficult.

Even though the ratios are problem-dependent, it is clear that the features of accuracy and independence of the number of parameters that the adjoint methods have, together with the ease of implementation that the automatic differentiation methods exhibit, are quite desirable to retain. This was the reasoning behind the purposed method of this dissertation – an hybrid method, called the *ADjoint* [108, 110].

3.6.1 Merging the adjoint and AD methods

The *ADjoint* relies on the discrete adjoint method to compute the sensitivities, making use of the adjoint (3.30) and the total sensitivity (3.31) equations to compute the gradients of the function of interest with respect to the design variables. However, rather than using AD to differentiate the entire source code of the CFD solver, AD is selectively applied to produce only the code that computes the individual entries in the flux Jacobian matrix and the other partial derivatives – $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$, $\frac{\partial I}{\partial \boldsymbol{w}}$, $\frac{\partial I}{\partial \boldsymbol{x}}$ and $\frac{\partial \mathcal{R}}{\partial \boldsymbol{x}}$ – that are necessary to compute sensitivities using an adjoint method.

Constructing the Jacobian $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$

The residual calculation of an iterative CFD flow solver is typically done in a subroutine that loops through the three-dimensional domain and accumulates the several fluxes and boundary conditions contributions in the residual $\mathcal{R}$. However, the residual at each cell (or node) only depends on the flow variables at that cell and at the cells adjacent to it, which define the stencil of dependence. As such, applying a purely automatic approach to the nested-loop residual code would translate into enormous computational inefficiencies. If the forward mode were used, then the cost of computing $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ would be roughly $(N_c \times N_w)$ times the cost of the original residual computation. If the reverse mode were used, then there would be a large memory penalty associated with the storage of all the intermediate variables generated by the series of nested loops, which is exactly what needs to be avoided.

Therefore, to avoid the automatic differentiation of nested loops over the whole computational domain, a re-engineered set of routines that mimic the original computation of the residual, but only at a given a cell (or node) location in the domain, must be created. That code can be easily constructed from the original residual evaluation routines in the flow solver by removing the loops over all the cells (or nodes) in the domain and making necessary adjustments so that the boundary conditions are handled properly. The new residual routine computes the N_v residuals at a specified cell (or node), getting contributions from all $(N_v \times N_s)$ flow variables in the stencil, where N_s denote the number of cells (or nodes) of the stencil.

Thus, there are $N_v \times (N_v \times N_s)$ sensitivities to be computed for each cell (or node), corresponding to N_v rows in the Jacobian adjoint matrix, $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$, where each of these rows contains no more than $(N_v \times N_s)$ non-zero entries.

At this stage, the only point that remains to be considered is that of the choice of the mode of automatic differentiation. In principle, because $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ is a square matrix, neither mode should have an advantage over the other in terms of computational time.

In the reverse mode, all of the $N_v \times (N_v \times N_s)$ derivatives in the stencil can be calculated from a single call to the differentiated residual routine at a given cell, since all of the information for that calculation is contained in a single stencil. These would

provide all of the non-zero entries of N_v rows in the Jacobian matrix.

On the other hand, in forward mode, all of the residuals affected by the states in a single cell (or node) would have to be considered. Theoretically, the derivatives of the $(N_v \times N_s)$ residuals affected by a single state can be computed, thus for a single cell (or node), a total of $N_v \times (N_v \times N_s)$ derivatives would again be obtained. However, because of the one-way dependence of the residual on the states, this does not prove to be the case. Therefore, it would be necessary to calculate all of the $N_v \times N_s$ residuals in the inverse stencil as well so, rather than having a single calculation with $N_v \times N_s$ states involved to get $N_v \times (N_v \times N_s)$ derivative values, it would require N_s differentiated residual calculations and many extra states to get the same number of derivative components.

Therefore, due to the way residuals are computed, the reverse mode is much more efficient in this case and, on this basis, it was used to produce adjoint code for the set of residual evaluation routines.

Constructing the vector $\frac{\partial I}{\partial \boldsymbol{w}}$

The right-hand side (RHS) vector of the adjoint equations (3.30) (or matrix, in the case of multiple functions of interest I) represents the direct effect of the flow variables on the function of interest.

In general, it would be necessary to obtain modified versions of the original functions to compute the derivatives, following the same procedure described above for the residual equations.

However, because the functions of interest considered in this dissertation are the inviscid aerodynamic coefficients, C_L, C_D, C_{Mx}, C_{My} and C_{Mz}, that are simple integrations of the pressure over the solid surfaces of the mesh for inviscid flow, these terms were evaluated analytically. These simplified functions have been chosen due to modeling limitations inherited from the MHD flow solvers used. In high-fidelity MHD simulations, however, not only the viscous effects should be taken into account, but also the reactions of the dipoles to the flow magnetic field must be considered when evaluating the total forces exerted on a vehicle.

Constructing the other partial derivative terms

As with the residual equations, modified versions of the original functions were used to compute the derivatives $\frac{\partial \mathcal{R}}{\partial \boldsymbol{x}}$ and $\frac{\partial I}{\partial \boldsymbol{x}}$ and procedures identical to the described for the partial derivatives with respect to the flow variables $\boldsymbol{w}$ are employed.

3.6.2 Benefits of the hybrid approach

It follows from the exposition made previously that the use of *ADjoint* method in the context of CFD brings several advantages compared to any other sensitivity analysis method. In summary, the major benefits gained by using this hybrid method are that it is:

- *Largely automatic*: Given the solver source code, it creates the necessary code to compute all the necessary terms in the discrete adjoint formulation.
- *Exactly consistent*: Because the process of automatic differentiation allows arbitrarily complex expressions for the computation of the residuals, boundary conditions, and cost functions to be treated exactly, the sensitivities produced are perfectly consistent with those that would be obtained with an exact numerical differentiation of the original solver. In other words, typical approximations made in the development of adjoint solvers (such as neglecting contributions from the variations resulting from turbulence models, spectral radii, artificial dissipation and upwind formulations) are not made here.
- *Generic*: It can be quickly applied to new formulations of the governing equations or even new governing equations themselves.

All the above supports the choice of using the *ADjoint* method in the present dissertation. Although this automated way of constructing discrete adjoint solvers will usually require more memory than the continuous adjoint, in particular if the full Jacobian matrix $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ is pre-assembled, the author believes that the advantages mentioned earlier greatly outweigh this disadvantage. In addition, alternative free-matrix adjoint computations are also possible that considerably mitigate this issue.

In that case, the entries of the matrix are evaluated on a row basis, on every GMRES iteration, which causes an overhead in run-time. Depending on the hardware resources available, the trade-off between computational cost and memory requirements must be considered when choosing the appropriate approach to handle the adjoint system matrix. As discussed in more detail in chapter 7, it is possible to reduce the memory requirements of the discrete adjoint approach to levels similar to those of continuous adjoint approaches, at the expense of increased CPU cost, though.

Refer to chapter 5 for the detailed implementation of this method in different flow solvers.

Chapter 4

Flow governing equations

The equations governing the three-dimensional flow of a compressible, conducting fluid with an externally imposed magnetic field are obtained by coupling the Navier–Stokes equations to the Maxwell equations. The resulting set of equations is designated as the magnetohydrodynamics (MHD) equations.

As detailed in chapter 2, there are plenty of references regarding MHD flow analysis, covering a large spectrum of model complexity, ranging from the low magnetic Reynolds number assumption, the ideal MHD, the full MHD, the inclusion of turbulence models, up to thermal and chemical non-equilibrium models.

Even though the MHD analysis can be extremely laborious and detailed, the MHD models used in this work are somewhat simpler and do not intend to reproduce the existing state-of-the-art solvers. This path was taken because there was not a readily available MHD solver, and the focus of this research was not specifically on the MHD field, but rather on the methodology for rapid development of discrete adjoint solvers. Also, the author believes that the results shown here are clearly demonstrative that the methodology developed can be seamlessly transfered to any set of governing equations, regardless of their complexity and completeness.

Thermodynamic and chemical models

In this work, the gas is assumed to be in thermodynamic equilibrium and to be calorically perfect, thus the equation of state for perfect gases $p = \rho RT$ holds, and

the internal energy and the enthalpy relate to the gas temperature by $e = C_v T$ and $h = C_p T$, respectively. In addition, finite-rate chemistry models are not included, meaning that the flow is assumed to be in frozen chemical equilibrium, with no chemical reactions accounted for.

4.1 Ideal MHD model

The equations governing the three-dimensional flow of an inviscid, compressible, perfectly conducting fluid in a magnetic field are called the ideal MHD equations.

Their derivation follows that of the full MHD equations, obtained by coupling the Navier–Stokes equations to the Maxwell equations (refer to appendix B), but additional assumptions are made. In particular, denoting the characteristic density, speed, length and time scales for the problem as ρ, U, L and τ, respectively, and the dielectric constant and conductivity of the fluid as ϵ and σ, the ideal MHD model assumes that

$$\frac{\mu}{\rho U L} \ll 1 \quad \text{and} \quad \frac{\epsilon}{\tau \sigma} \ll 1 \,. \tag{4.1}$$

The implications are that the flow might be considered inviscid, thus the viscous terms of the Navier–Stokes equations vanish, and that the fluid is perfectly conducting, implying that the dispersive terms of the induction equations can be neglected.

In the literature, several versions of the ideal MHD equations can be found, all resulting from the inclusion of additional magnetic terms in the Navier–Stokes equations – the momentum equation gets an additional force per unit volume of matter (Lorentz force) given by expression (B.11), and the additional power delivered to matter given by expression (B.12) is included in the energy equation – and the additional magnetic field transport equation, that is derived from the Faraday's law (B.4). Examples of these are seen in the work of Powell *et al.* [133] and Gaitonde [42].

In the present work, however, a different formulation, that could not be found in any reference, is derived. It combines two features: the inclusion of additional source terms that enforce the solenoidal condition of the magnetic field, and the decomposition of the total magnetic field into the background imposed field and the

induced field to achieve better numerical accuracy.

Among the various possible mathematical forms, the resulting MHD equations are expressed in conservation-law form which is often preferred for numerical models.

4.1.1 Enforcement of the $\nabla \cdot \mathbf{B} = 0$ condition

According to the Gauss' law for magnetism (B.2), the magnetic field has to satisfy the divergence-free condition,

$$\nabla \cdot \mathbf{B} = 0\,. \tag{4.2}$$

In addition, the transient evolution of the magnetic field is governed by Faraday's law,

$$\frac{\partial \mathbf{B}}{\partial t} = -c\nabla \times \mathbf{E}\,, \tag{4.3}$$

where c is the speed of light and E is the electric field. Recalling the vector identity (A.4), the electric field satisfies the conditions $\nabla \cdot (\nabla \times \mathbf{E}) = 0$. The application of this identity to equation (4.3) implies that $\nabla \cdot \mathbf{B}$ is independent of time, as cited by Ramshaw [137]. Therefore, the exact solution of the ideal MHD equations keeps the condition (4.2) valid indefinitely, if it is satisfied initially.

However, in numerical MHD simulations, not only round-off errors but also the use of artificial dissipation schemes lead to a finite divergence of the magnetic field, thus violating that condition and not preserving the differential property of $\nabla \cdot \boldsymbol{B}$. Among others, Brackbill and Barnes [19] have shown that even very small errors in satisfying equation (4.2) cause large errors in the solution of the MHD equations.

There are different approaches to enforce the solenoidal condition (4.2) but in the present work, the inclusion of additional source terms in the MHD equations has been used. Usually, the derivation of the ideal MHD equations is done assuming that terms proportional to $\nabla \cdot \mathbf{B}$ are zero analytically. Had this not been done, there would have been extra source terms (4.4), on the right-hand side of the equations, as shown in

the derivations of both Panofsky [126] and Vinokur [159]

$$\mathbf{S} = -\nabla \cdot \mathbf{B} \begin{pmatrix} 0 \\ R_b \mathbf{B}/\mu_m \\ R_b \left(\mathbf{u} \cdot \mathbf{B}\right)/\mu_m \\ \mathbf{U} \end{pmatrix}. \tag{4.4}$$

Both Powell *et al.* [134], and Tóth and Odstrčil [155] have found that including these corrective terms stabilizes and improves the solution. It should be noted that the terms are non-conservative; thus conservation of momentum, energy, and magnetic flux are not strictly enforced any longer.

4.1.2 Magnetic field decomposition

Besides the issue of satisfying the solenoidal condition (4.2), there is also the problem of having large imposed magnetic fields. Under these circumstances, the ratio of induced to imposed components of the magnetic field becomes extremely small and the magnetic terms can dominate the system. Small errors in the magnetic field solution can cause severe difficulties in the energy equation, because the magnetic energy becomes much greater than the kinetic energy.

Following the work of Tanaka [153], this problem can be mitigated by decomposing the magnetic field $\mathbf{B}$ into two components, the background imposed field, $\mathbf{B}_0$, and the induced field, $\mathbf{B}_i$, as indicated by expression (B.47).

4.1.3 Flux Vector Form

Consequently, the conservative system of ideal MHD equations used in this work was custom derived in order to obtain a stable and accurate system, and its derivation is detailed in appendix B. Neglecting the viscous and dispersive effects, as well as heat transfer, under the assumptions (4.1), the full MHD equations (B.55) simplify to the ideal MHD equations. This set of equations explicitly represent the conservation of mass, momentum, total energy, and magnetic induction of a flow field under the

presence of a magnetic field and it is given by

$$\frac{\partial \boldsymbol{w}_i}{\partial t} + (\nabla \cdot \mathcal{F}_i) + (\nabla \cdot \mathcal{F}_m) = \mathbf{S}_i \,, \tag{4.5}$$

where

$$\boldsymbol{w}_i = \begin{pmatrix} \rho \\ \rho\mathbf{u} \\ \rho Z_i \\ \mathbf{B}_i \end{pmatrix} \,, \quad \mathcal{F}_i = \begin{pmatrix} \rho\mathbf{u} \\ \rho\mathbf{u}\mathbf{u} + p\mathbf{I} \\ (\rho E + p)\mathbf{u} \\ 0 \end{pmatrix} \,,$$

$$\mathcal{F}_m = \begin{pmatrix} 0 \\ R_b(\mathbf{B}_i \cdot \mathbf{B}_i)\mathbf{I}/(2\mu_m) - R_b(\mathbf{B}_i\mathbf{B}_i)/\mu_m + R_b(\mathbf{B}_0 \cdot \mathbf{B}_i)\mathbf{I}/\mu_m - R_b(\mathbf{B}_0\mathbf{B}_i + \mathbf{B}_i\mathbf{B}_0)/\mu_m \\ R_b(\mathbf{B}_i \cdot \mathbf{B}_i)\mathbf{u}/\mu_m - R_b(\mathbf{u} \cdot \mathbf{B}_i)\mathbf{B}_i/\mu_m + R_b(\mathbf{B}_0 \cdot \mathbf{B}_i)\mathbf{u}/\mu_m - R_b(\mathbf{u} \cdot \mathbf{B}_i)\mathbf{B}_0/\mu_m \\ (\mathbf{u}\mathbf{B}_i - \mathbf{B}_i\mathbf{u}) + (\mathbf{u}\mathbf{B}_0 - \mathbf{B}_0\mathbf{u}) \end{pmatrix}$$

and

$$\mathbf{S}_i = -\left(\nabla \cdot \mathbf{B}_i\right) \begin{pmatrix} 0 \\ R_b\mathbf{B}/\mu_m \\ R_b\left(\mathbf{u} \cdot \mathbf{B}_i\right)/\mu_m \\ \mathbf{u} \end{pmatrix} .$$

The inviscid and magnetic flux vectors are obtained as $\mathcal{F}_i = \mathbf{E}_i\hat{e}_x + \mathbf{F}_i\hat{e}_y + \mathbf{G}_i\hat{e}_z$ and $\mathcal{F}_m = \mathbf{E}_m\hat{e}_x + \mathbf{F}_m\hat{e}_y + \mathbf{G}_m\hat{e}_z$, respectively, with the fluxes along the Cartesian directions given in detail by expressions (B.57)-(B.59) and (B.60)-(B.62), that are included in appendix B.

The conservative variables composing $\boldsymbol{w}$ are the density, ρ, the momentum density, $\rho\mathbf{u}$, the total energy density, ρZ_i, and the induced magnetic field, $\mathbf{B}_i$. It should be noted that the MHD total energy per unit volume, ρZ_i, is composed of the usual total energy, ρE, augmented by the induced magnetic energy contribution, as expressed in equation (B.51). The pressure, P_i, is defined as sum of the static pressure, p, and the magnetic pressure, as given by equation (B.52). Combining these equations under the ideal gas assumption, the energy ρZ_i, density ρ, momentum $\rho\mathbf{u}$ and magnetic field

$\mathbf{B}_i$ are related to pressure p by

$$p = (\gamma - 1)\left(\rho Z_i - \frac{1}{2}\rho u^2 - R_b \frac{\boldsymbol{B}_i \cdot \mathbf{B}_i}{2\mu_m}\right), \tag{4.6}$$

where γ is the ratio of the constant pressure and constant volume heat coefficients, R_b is the magnetic force number (B.38) and μ_m is the magnetic permeability.

This ideal MHD model allows for environments characterized by a high magnetic force number, where the magnetic field induced by the current is of comparable magnitude to the one imposed on the flow, since the three induction equations are solved in the governing equations, as opposed to the low magnetic Reynolds number model (refer to section 4.2), as referred by Poggie and Gaitonde [130].

It is relevant to mention that the splitting (B.47) makes no assumption about the relative size of $\mathbf{B}_0$ and $\mathbf{B}_i$; the only requirement is that the imposed magnetic field satisfies (B.48).

4.2 Low magnetic Reynolds number MHD model

In external aerodynamics, many of the encountered hypersonic flow fields are characterized by relatively low levels of electrical conductivity, σ. The pertinent non-dimensional parameter determining the relative magnitude of σ is the magnetic Reynolds number, Re_σ (B.37).

In the limit of $Re_\sigma \to 0$, the energy (B.43) and magnetic induction (B.46) equations of the full MHD model are well behaved analytically even though Re_σ appears in the denominator of some terms. In each of these, the numerator approaches zero more rapidly. For example, the derivation of eqn.(B.43) shows that the last terms are actually proportional to j^2/σ which by virtue of the non-dimensionalization of the Ohm's law are proportional to Re_σ.

From the numeric standpoint, however, terms involving Re_σ are problematic since their proper evaluation requires ratios of relatively small quantities, some of which are obtained discretely. Terms involving R_b on the other hand are numerically well behaved in the limit $Re_\sigma \to 0$ and decouple the magnetic field from the fluid flow by

virtue of the fact that the divergence of the Maxwell's stress tensor vanishes. However, accuracy considerations continue to be important if R_b is large, since small errors in the evaluation of the Maxwell's stresses can translate into large body forces.

To alleviate these problems, considerable simplification can be achieved by recognizing that the induced magnetic field is negligible when Re_σ is small. In this case, Ampère's law (B.13) is suppressed and the current is obtained directly from the generalized Ohm's law (B.10). The electric current $\mathbf{j}$ can be non-dimensionalized using the reference values (B.33), resulting in the generalized Ohm's law

$$\mathbf{j} = Re_\sigma \sigma \left(\mathbf{E} + \mathbf{u} \times \mathbf{B} \right) . \tag{4.7}$$

If the environment of interest is characterized by a low magnetic Reynolds number, then the magnetic field induced by the current is much smaller than that imposed on the flow and, therefore, it can be neglected [48, 44]. In this way, there is no need to solve the three induction equations in the governing equations and the electromagnetic forces and energy show up as source terms in the Navier–Stokes equations.

Besides neglecting the distortion of the magnetic field by the flow and only assuming that the imposed field has a significant influence on the flow, even simpler models can be used. Namely, if the viscous effects and heat transfer are negligible, the Navier–Stokes equations reduce to the Euler equations.

The derivation of the non-dimensional ideal MHD equations governing the flow under these conditions is made in the sections that follow.

Continuity equation

The continuity equations remains unchanged from the Navier–Stokes equations, as expressed in non-dimensional form in eqn. (B.40),

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \, . \tag{4.8}$$

Momentum equation

The Navier–Stokes momentum equation (B.17) can be made non-dimensional using the expressions (B.33), yielding

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u} + p\mathbf{I}) - \frac{1}{Re} \nabla \cdot \vec{\tau} = \mathbf{f}\,, \tag{4.9}$$

with the non-dimensional magnetic source term defined as

$$\mathbf{f} = \mathbf{j} \times \mathbf{B} = Q\sigma \left(\mathbf{E} + \mathbf{u} \times \mathbf{B}\right) \times \mathbf{B}\,, \tag{4.10}$$

where Q is the magnetic interaction parameter, given by expression (B.39).

Energy equation

The Navier–Stokes energy equation (B.26) can also be expressed in non-dimensional form as

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot \left[(\rho E + p)\, \mathbf{u} - \frac{1}{Re} \mathbf{u} \cdot \vec{\tau} - \frac{\mu}{RePr(\gamma - 1)M^2} \nabla T \right] = P\,, \tag{4.11}$$

where the non-dimensional additional magnetic energy source term can be written as

$$P = \mathbf{E} \cdot \mathbf{j} = Q\sigma \left(\mathbf{E} + \mathbf{u} \times \mathbf{B}\right) \cdot \mathbf{E}\,. \tag{4.12}$$

It can be seen from the expression (4.12) that if the magnetic induction field $\mathbf{B}$ is steady and there is no imposed electrostatic field $\mathbf{E}$, then the source term of the energy equation vanishes.

4.2.1 Flux Vector Form

Adopting the low Re_σ model approximation, the complete set of governing equations is given by equations (4.8) to (4.12). This set of non-dimensional equations may be

written in flux vector form as

$$\frac{\partial \boldsymbol{w}}{\partial t} + \frac{\partial \mathbf{E}_i - \mathbf{E}_v}{\partial x} + \frac{\partial \mathbf{F}_i - \mathbf{F}_v}{\partial y} + \frac{\partial \mathbf{G}_i - \mathbf{G}_v}{\partial z} = \mathbf{S}\,, \tag{4.13}$$

where the vector of conservative variables is $\boldsymbol{w} = (\rho, \rho u, \rho v, \rho w, \rho E)$, and the inviscid – $\mathbf{E}_i$, $\mathbf{F}_i$, $\mathbf{G}_i$ – and viscous – $\mathbf{E}_v$, $\mathbf{F}_v$, $\mathbf{G}_v$ – flux vectors in the x,y and z-directions contain only the first five rows of the full governing equations fluxes given by expressions (B.57)– (B.59) and (B.66)– (B.68), and can be derived by setting $R_b = 0$.

The source term $\mathbf{S}$ includes the magnetic field effects and it is given by

$$\mathbf{S} = \begin{pmatrix} 0 \\ Q\sigma\left[B_z\left(E_y + wB_x - uB_z\right) - B_y\left(E_z + uB_y - vB_x\right)\right] \\ Q\sigma\left[B_x\left(E_z + uB_y - vB_x\right) - B_z\left(E_x + vB_z - wB_y\right)\right] \\ Q\sigma\left[B_y\left(E_x + vB_z - wB_y\right) - B_x\left(E_y + wB_x - uB_z\right)\right] \\ Q\sigma\left[E_x\left(E_x + vB_z - wB_y\right) + E_y\left(E_y + wB_x - uB_z\right) + E_z\left(E_z + uB_y - vB_x\right)\right] \end{pmatrix}. \tag{4.14}$$

In compact vector form, the low Re_σ MHD model (4.13) is expressed as

$$\frac{\partial \boldsymbol{w}}{\partial t} + (\nabla \cdot \mathcal{F}_i) - (\nabla \cdot \mathcal{F}_v) = \mathbf{S}\,, \tag{4.15}$$

where

$$\boldsymbol{w} = \begin{pmatrix} \rho \\ \rho\mathbf{u} \\ \rho E \end{pmatrix}, \quad \mathcal{F}_i = \begin{pmatrix} \rho\mathbf{u} \\ \rho\mathbf{u}\mathbf{u} + p\mathbf{I} \\ (\rho E + p)\mathbf{u} \end{pmatrix},$$

$$\mathcal{F}_v = \tfrac{1}{Re}\begin{pmatrix} 0 \\ \vec{\vec{\tau}} \\ \mathbf{u}\cdot\vec{\vec{\tau}} + \frac{\mu}{Pr(\gamma-1)M^2}\nabla T \end{pmatrix} \text{ and } \quad \mathbf{S} = Q\begin{pmatrix} 0 \\ \sigma\left(\mathbf{E} + \mathbf{u}\times\mathbf{B}\right)\times\mathbf{B} \\ \sigma\left(\mathbf{E} + \mathbf{u}\times\mathbf{B}\right)\cdot\mathbf{E} \end{pmatrix}.$$

As previously stated, the fluxes in this model are exactly the same as those of the original Navier–Stokes equations, where the only difference is in the additional source term, $\mathbf{S}$.

Regarding the governing equations in the low Re_σ approximation, since the influence of the magnetic field is restricted exclusively to the source term, the eigensystem

of the resulting set is not afflicted by the degenerate eigenvalues found in the ideal or full MHD models [143].

4.3 Imposed magnetic field

In all the MHD simulations made in this work, the magnetic field is externally imposed by considering the existence of a set of electromagnetic circuits that create a dipole-like magnetic field on the flow. This approach is commonly used in MHD simulations, as seen in the work of Gaitonde and Poggie [46].

The field created by such a dipole is axisymmetric and, as such, it is conveniently expressed in spherical coordinates, as shown in figure 4.1, where ϕ and θ designate the azimuthal and polar angles, respectively.

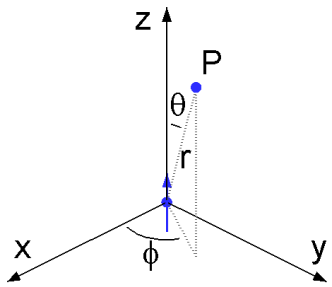


Figure 4.1: Magnetic dipole.

The magnetic field generated by a single dipole is given by the vector function

$$\mathbf{B} = \frac{\mu_m m}{4\pi r^3} \left(2\cos\theta \hat{e}_r + \sin\theta \hat{e}_\theta\right) , \tag{4.16}$$

where $\mathbf{B}$ is the imposed magnetic field at a point located at a distance r from the origin of the dipole, of strength m, at an angle θ with respect to the dipole direction. The field $\mathbf{B}$ is given by expression (4.16) in terms of local dipole coordinates. When a set of dipoles is used, it is required to change from each local reference frame to a global frame using coordinate transformations, which include both translations and rotations, so that the total imposed magnetic field can be evaluated at each point in the simulation domain.

It is important to guarantee that the imposed magnetic field satisfies the magnetic field equations, in particular the Gauss's law (4.2), especially if either the low magnetic Reynolds number approximation model or the ideal MHD governing equations are used. It can be easily shown that the dipole field (4.16) satisfies that divergence-free condition. Consequently, because of the additive property of the divergence operator, $\nabla\cdot$, the total imposed field given by the superimposition of the individual contributions still satisfies that condition.

4.4 Spatial discretization

The most common spatial discretization formulations in CFD are the finite-volume method (FVM) and the finite-difference method (FDM). Even though their mathematical description may differ, the resulting algebraic equations might be equivalent, if special care is taken to guarantee that both formulations are conservative.

In this dissertation, two different CFD solvers have been used – one uses a finite-volume formulation, while the other uses finite-differences –, in order to demonstrate the general applicability of the sensitivity analysis method pursued. Both formulations and solvers are briefly described in the sections that follow.

4.4.1 Finite-volume formulation

To use a finite-volume formulation, the physical domain of interest is first divided into a large number of small sub-domains, often called cells. After this spatial discretization, the conservation laws, expressed by either (4.5) or (4.15), are applied in integral form to each sub-domain. The use of the integral form allows for discontinuities in the solution, which arise naturally in hypersonic flow with the presence of shock waves.

Let Ω be the volume of an arbitrary sub-domain. Then, integrating equation (4.5) over Ω and making use of the Gauss' theorem yields

$$\iiint_{\Omega} \frac{\partial \boldsymbol{w}}{\partial t}\,\mathrm{d}\Omega + \iint_{\partial\Omega} \boldsymbol{\mathcal{F}}\cdot\hat{\mathbf{n}}\,\mathrm{d}\partial\Omega = \iiint_{\Omega} \mathbf{S}_i\,\mathrm{d}\Omega\,, \tag{4.17}$$

where $\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{F}}_i + \boldsymbol{\mathcal{F}}_m$.

Let the cells in the computational domain be denoted by the subscripts (i, j, k). A set of coupled ordinary differential equations (ODEs) is then obtained by applying equation (4.17) separately to each computational cell and it can be written in the form

$$\frac{d}{dt}(V_{ijk}\boldsymbol{w}_{ijk}) + \mathcal{Q}_{ijk} = V_{ijk}\mathbf{S}_{\mathbf{i}\,ijk}\,, \tag{4.18}$$

where V_{ijk} is the cell volume, $\mathcal{Q}_{ijk}$ is the net flux out of the cell over its sides, including boundary conditions, and $\mathbf{S}_{\mathbf{i}\,ijk}$ are the source terms contributions.

4.4.2 Finite-volume solver

For purposes of rapid prototyping of the implementation of a discrete adjoint, a custom single-processor solver for the Euler, low Re_σ and ideal MHD equations was built.

The spatial discretization for this solver was done on a single computational block basis, surrounded by halo cells to facilitate the implementation of the boundary conditions. Even though the finite-volume formulation can be applied to arbitrary subdomains, a three-dimensional discretization composed of a body-fitted, structured mesh of hexahedral cells was used.

A cell-centered scheme was selected, in which the dependent variables are assumed to be known at the center of each cell. As such, each quantity is evaluated as the average of the values in the cells on either side of a face. For instance, the momentum in the x-direction at face $(i + \frac{1}{2})$ is evaluated as

$$(\rho u)_{i+1/2,j,k} = \frac{1}{2}[(\rho u)_{i,j,k} + (\rho u)_{i+1,j,k}]\,. \tag{4.19}$$

This scheme reduces to a central-difference scheme on a Cartesian grid, and it is second-order accurate provided that the mesh is sufficiently smooth.

A two-dimensional illustration of the fluxes $\mathcal{F}$ is depicted in figure 4.2. Although this solver implementation was fully three-dimensional, the exposition that follows of the several terms that comprise the residual, $\mathcal{R}_{ijk}$, is restricted to a single computational coordinate for clarity.

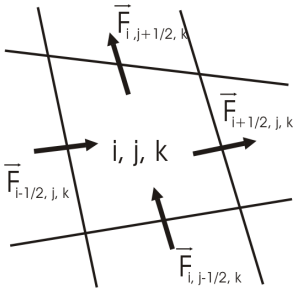


Figure 4.2: Finite-volume fluxes.

Inviscid and magnetic fluxes

Let the area of the right face of cell i be denoted by $\mathbf{S}_{i+\frac{1}{2}}$. The inviscid and magnetic residual contributions, $\mathcal{Q}(\boldsymbol{w})$, can be expressed in terms of the fluxes as

$$\mathcal{Q}(\boldsymbol{w}) = (\boldsymbol{\mathcal{F}} \cdot \mathbf{S})_{i+\frac{1}{2}} - (\boldsymbol{\mathcal{F}} \cdot \mathbf{S})_{i-\frac{1}{2}} \ , \tag{4.20}$$

where the flux at the cell face is evaluated as

$$(\boldsymbol{\mathcal{F}} \cdot \mathbf{S})_{i+\frac{1}{2}} = \frac{1}{2} \left(\boldsymbol{\mathcal{F}}_{i+1} + \boldsymbol{\mathcal{F}}_i\right) \cdot \mathbf{S}_{i+\frac{1}{2}} \, , \tag{4.21}$$

and

$$\begin{aligned} \boldsymbol{\mathcal{F}}_i \cdot \mathbf{S}_{i+\frac{1}{2}} &= \begin{pmatrix} \mathbf{E}_i + \mathbf{E}_m \\ \mathbf{F}_i + \mathbf{F}_m \\ \mathbf{G}_i + \mathbf{G}_m \end{pmatrix}_i \cdot \mathbf{S}_{i+\frac{1}{2}} = \\ &= (\mathbf{E}_i + \mathbf{E}_m)_i S_{x\, i+\frac{1}{2}} + (\mathbf{F}_i + \mathbf{F}_m)_i S_{y\, i+\frac{1}{2}} + (\mathbf{G}_i + \mathbf{G}_m)_i S_{z\, i+\frac{1}{2}} \, . \end{aligned} \tag{4.22}$$

The inviscid and magnetic flux vectors in each Cartesian direction – $\mathbf{E}_i$, $\mathbf{F}_i$, $\mathbf{G}_i$ $\mathbf{E}_m$ $\mathbf{F}_m$ $\mathbf{G}_m$, respectively –, are given by expressions (B.57)-(B.59) and (B.60)-(B.62), found in appendix B.

Artificial dissipation

In order to increase the numerical stability of the scheme, namely to suppress the odd and even point coupling, and the overshoots near shock waves, the scheme is augmented by a dissipative term $\mathcal{D}_{ijk}$. The equation (4.18) then becomes

$$\frac{d}{dt}(V_{ijk}\boldsymbol{w}_{ijk}) + \mathcal{Q}_{ijk} - \mathcal{D}_{ijk} = V_{ijk}\mathbf{S}_{\mathbf{i}\,ijk}\,. \tag{4.23}$$

Along each computational direction, the artificial dissipation has the form

$$\mathcal{D}(\boldsymbol{w}_i) = d_{i+\frac{1}{2}} - d_{i-\frac{1}{2}}\,, \tag{4.24}$$

where $d_{i+\frac{1}{2}}$ is the dissipative flux contributed by the face between cells i and $i+1$.

Several dissipation algorithms have been constructed for central-difference schemes, and their comparison has been performed by Swanson *et al.* [152]. Among the most common ones are the scalar and matrix dissipation (MATD) schemes. This solver uses the Jameson–Schmidt–Turkel (JST) scalar dissipation scheme [76, 78].

The JST scheme uses a scalar artificial dissipation term, which is a blend of first- and third-order-accurate terms to provide good numerical stability properties while keeping its implementation relatively easy and computationally inexpensive. This scheme defines a switching function based on a blending of the second and forth differences, so that it is of third order in smooth regions of the flow but reverts to first-order in regions of high pressure gradients. The JST artificial dissipation flux is defined, along direction i, as

$$d_{i+\frac{1}{2}} - d_{i-\frac{1}{2}} = (D^2 - D^4)\boldsymbol{w}_i \tag{4.25}$$

where the difference operators are

$$D^2\boldsymbol{w}_i = \Delta_B\left[\left(\lambda_{i+\frac{1}{2}}\varepsilon^{(2)}_{i+\frac{1}{2}}\right)\Delta_F\right]\boldsymbol{w}_i \quad \text{and} \tag{4.26}$$

$$D^4\boldsymbol{w}_i = \Delta_B\left[\left(\lambda_{i+\frac{1}{2}}\varepsilon^{(4)}_{i+\frac{1}{2}}\right)\Delta_F\Delta_B\Delta_F\right]\boldsymbol{w}_i\,, \tag{4.27}$$

with Δ_B and Δ_F being the backward and forward difference operators. The scaling factor $\lambda_{i+\frac{1}{2}}$ is a function of the spectral radius of the governing equations and the coefficients $\varepsilon^{(2)}$ and $\varepsilon^{(4)}$ use the pressure as a sensor for sharp gradients. Their detailed definition can be found in reference [78]. The pressure sensor guarantees that the flux (4.25) is of second-order except in regions containing a steep pressure gradient. The fourth differences provide background dissipation throughout the domain. In the neighborhood of a shock wave, however, the second differences become the dominant dissipative terms.

Boundary conditions

Using a body conforming mesh, the boundary condition implementation becomes greatly simplified.

At solid walls defining the body surface, because the physical models considered are inviscid, the impermeability or tangent velocity condition (4.28) holds. Consequently, the velocity vector at the halo cells is mirrored, such that the normal fluxes at the boundary are zero.

$$\mathbf{u} \cdot \hat{\mathbf{n}} = 0\,. \tag{4.28}$$

As a result, the inviscid flux through a cell face at the wall simplifies to

$$(\boldsymbol{\mathcal{F}} \cdot \mathbf{S})_{wall} = \begin{pmatrix} 0 \\ p_w S_x \\ p_w S_y \\ p_w S_z \\ 0 \end{pmatrix}\,, \tag{4.29}$$

where p_w is the pressure at the wall. In turn, the pressure at the wall can be either extrapolated from interior cells or estimated using the value of normal pressure gradient $\frac{\partial p}{\partial n}$ at the wall. This gradient can be estimated from the manipulation of the body streamline equation (4.28) and the inner product of the momentum equation

with the normal $\hat{\mathbf{n}}$, that leads to the relation given by Rizzi [142] as

$$\rho \mathbf{u} \cdot (\mathbf{u} \cdot \nabla) \hat{\mathbf{n}} = \hat{\mathbf{n}} \cdot \nabla p \,, \tag{4.30}$$

which relates the density, velocity, and body geometry (through $\hat{\mathbf{n}}$) to the normal pressure gradient, all located at the body.

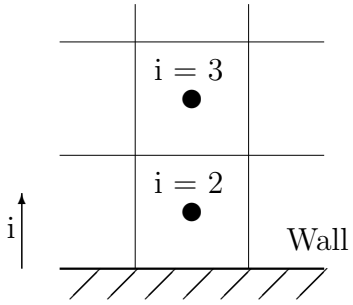


Figure 4.3: Wall boundary conditions.

For simplicity of implementation, and with reference to figure 4.3, the pressure at the wall is estimated using first-order extrapolation as

$$p_w = p_{(i=2)} - \frac{p_{(i=3)} - p_{(i=2)}}{2} = \frac{3}{2} p_{(i=2)} - \frac{1}{2} p_{(i=3)} \,. \tag{4.31}$$

The imposed magnetic field remains fixed at the halo cells, but the induced component is mirrored to get to zero at the wall, assuming the body is dielectric.

The far-field boundary conditions are, in general, imposed using the Riemann invariants, as found in the book of Chung [25]. In conventional aerodynamic problems, the local speed of sound, c, is the measure against which the flow speed is compared to determine whether the characteristic waves travel downstream or upstream. However, the wave speeds in MHD are different, as seen in figure B.1 found in section B.6. Under MHD conditions, the fast magneto-acoustic wave (B.73), c_f, is the fastest wave and determines the direction of propagation of information. In this case, the Riemann invariants are pictured in figure 4.4. At inflow boundaries, $U_n = \boldsymbol{u} \cdot \hat{\boldsymbol{n}} < 0$, the flow is considered supersonic if $U_n < -c_f$, and subsonic if $U_n > -c_f$. Conversely, at outflow

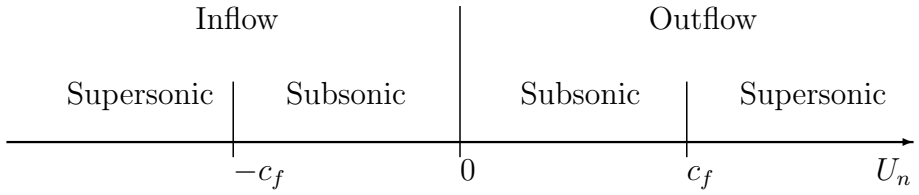


Figure 4.4: MHD Riemann invariants.

boundaries, $U_n = \boldsymbol{u} \cdot \hat{\boldsymbol{n}} > 0$, it is supersonic for $U_n > c_f$, and subsonic for $U_n < c_f$. In the subsonic case, non-reflecting boundary conditions have to be carefully defined to account for the waves traveling upstream, as mentioned by Hedstrom [66]. However, since all the problems simulated in this work have hypersonic free-stream velocities and relatively small magnetic field intensities, the flow remains supersonic (in MHD terms) at the far-field boundaries. These boundary conditions then get considerably simpler than in the subsonic case. As such, inflow and outflow conditions are trivially imposed at hypersonic boundaries, as all the characteristics of the governing equations (4.5) or (4.15) travel downstream, greatly simplifying the Riemann problem. As such, the complete vector of conservation variables, $\boldsymbol{w}$, is fixed at the halo cells of supersonic incoming boundaries, whereas the values at the supersonic outflow boundary halos are extrapolated from the interior. It is important to notice that while the imposed magnetic field is always fixed, the induced portion is set to zero at inflow boundaries, assuming the boundary is located far enough from the magnetic sources.

Summary of the cell-centered residual computation

The computation of the residual, $\mathcal{R}_{ijk}$, at a given cell, can be summarized as follows:

- Compute inviscid fluxes: For the present inviscid flux discretization, the only flow variables, $\boldsymbol{w}$, that influence the residual at a cell are the flow variables at that cell and at the six cells that are adjacent to the faces of the cell.

- Compute artificial dissipation fluxes: In this case, since the second-order JST scheme was used, the residual at a cell gets contributions from the flow variables

in that cell and from the first- and second-level adjacent cells in each of the three coordinate directions.

- Compute magnetic fluxes: This operation has a similar stencil to the inviscid flux computation.

- Apply boundary conditions: The boundaries are enforced by altering the states in the halo cells, thus they only contribute to the residual on immediately adjacent cells.

According to the residual computation just described, the inviscid and magnetic flux terms only require one level of adjacent cells, while the artificial dissipation fluxes require two levels of adjacent cells. Therefore, the stencil of influence contains thirteen cells, as illustrated in figure 4.5.

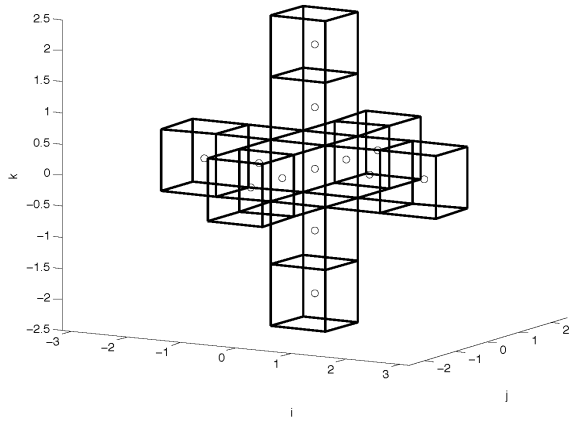


Figure 4.5: Stencil of dependence for the cell-centered residual computation.

As usual for iterative solvers, the residuals over the domain are computed using three nested loops (one in each of the three computational directions) sweeping the faces in a computational block. The total residual for any given cell is only obtained at the end of the loops, when all contributions have been accounted for. This approach translates into significant computational savings since the flux residual at each cell face is shared between the neighboring cells.

4.4.3 Finite-difference formulation

The MHD governing equations (4.5) and (4.15) can also be solved using a finite-difference discretization scheme. For this purpose, a coordinate transformation from physical coordinates (x, y, z) to computational coordinates (ξ, η, ζ) is used, resulting in

$$\frac{\partial \boldsymbol{w}}{\partial t} + \frac{\partial \bar{\mathbf{E}}}{\partial \xi} + \frac{\partial \bar{\mathbf{F}}}{\partial \eta} + \frac{\partial \bar{\mathbf{G}}}{\partial \zeta} = \bar{\mathbf{S}}\,, \tag{4.32}$$

where the state, the fluxes vectors and the source terms in the computational coordinates are given by

$$\begin{cases} \bar{\boldsymbol{w}} = \frac{\boldsymbol{w}}{J} \\ \bar{\mathbf{E}} = \frac{1}{J}(\xi_x \mathbf{E} + \xi_y \mathbf{F} + \xi_z \mathbf{G}) \\ \bar{\mathbf{F}} = \frac{1}{J}(\eta_x \mathbf{E} + \eta_y \mathbf{F} + \eta_z \mathbf{G}) \\ \bar{\mathbf{G}} = \frac{1}{J}(\zeta_x \mathbf{E} + \zeta_y \mathbf{F} + \zeta_z \mathbf{G}) \\ \bar{\mathbf{S}} = \frac{1}{J}\mathbf{S} \end{cases}, \tag{4.33}$$

where J is the coordinate transformation Jacobian. More details about generalized coordinate transformations can be found in appendix C.

After the spatial discretization of the governing equations (4.32) has been carried out, a system of ordinary differential equations (ODE) is obtained,

$$\frac{d}{dt}\left(\frac{\boldsymbol{w}_{ijk}}{J_{ijk}}\right) + \mathcal{R}_{ijk} = 0\,, \tag{4.34}$$

where the triad ijk represents each and every node in the mesh. This equation is of the same form as the one obtained using a finite-volume discretization (4.23), and, consequently, its time integration can be carried out using the some algorithm.

4.4.4 Finite-difference solver

The flow solver used in this case was the *Navier–Stokes Stanford University Solver* (*NSSUS*) flow solver. This solver is a new finite-difference, higher-order solver that has been developed at Stanford University under the Advanced Simulation and Computing (ASC) program sponsored by the Department of Energy [8].

It is a generic node-centered, multi-block, multi-processor solver, currently only

for the Euler equations, but soon to be extended to the Reynolds-Averaged Navier–Stokes equations. The finite-difference operators and artificial dissipation terms follow the work of Mattson and Nordström [113, 114] and the boundary conditions are implemented by means of penalty terms, according to the work of Carpenter *et al.* [22, 23]. The additional magnetic induction equations (4.5) and source terms (4.14) had to be included in this solver so that MHD computations could be performed.

Despite being capable of performing computations up to eight-order accuracy, the implementation of the adjoint solver was restricted to second-order for simplicity. The extension to higher-order accuracy should be straightforward to accomplish.

Inviscid, magnetic and artificial dissipation fluxes

The spatial part of equation (4.32) is discretized on a block-by-block basis.

The internal discretization is simple and only requires the first neighbors in each coordinate direction for the inviscid and magnetic fluxes, and the first and second neighbors for the artificial dissipation fluxes, as shown in figure 4.8. The boundary treatment needs to be explained in more detail, though.

Boundary conditions

As the finite-difference scheme only operates on the nodes of a block, one-sided difference formulae are used near block boundaries (be it a physical or an internal boundary).

Consequently, the nodes on the interface of internal boundaries are multiply defined, as illustrated in figure 4.6. These multiple instances of the same physical node

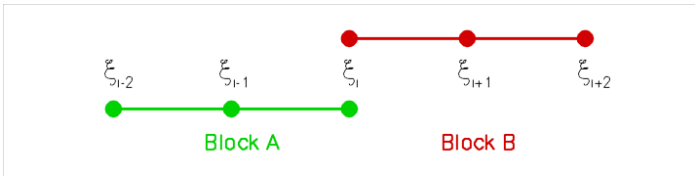


Figure 4.6: Block-to-block boundary stencil.

are driven to the same value (at convergence) by means of a penalty term, i.e., an

additional term is added to the residual, $\mathcal{R}$, that is proportional to the difference between the instances. This reads

$$\mathcal{R}^i_{\text{blockA}} = \mathcal{R}^i_{\text{blockA}} + \tau(w^i_{\text{blockB}} - w^i_{\text{blockA}})\,, \tag{4.35}$$

and a similar expression can be derived for $\mathcal{R}^i_{\text{blockB}}$. In equation (4.35), the parameter τ controls the strength of the penalty and is a combination of a user-defined parameter and the local flow conditions (see reference [113] for more details). Hence, the residual of a node that lies on an internal block boundary is a function of its local neighbors in the block and the corresponding instance in the neighboring block.

The treatment of physical boundaries is very similar to the approach described above except that the penalty term used in equation (4.35) is now determined by the boundary conditions.

Only when runtime block-splitting occurs, do halo nodes get used. This happens when any original computational block is too large and has to be split into different processors for load balancing. The stencils need not to be adjusted close to the these newly created boundaries, as seen in figure 4.7.

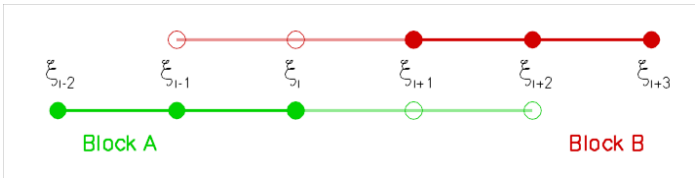


Figure 4.7: Block-splitting boundary stencil.

All of these boundary contributions require special attention when taking care of the assembly of the flux Jacobian matrix in the adjoint system of equations (3.30). This is ensured with the proper treatment of the global node numbering, that handles all the node connectivities.

Summary of the node-centered residual computation

In the present case, the residual $\mathcal{R}_{ijk}$ in equation (4.34) includes the inviscid and magnetic fluxes, the artificial dissipation fluxes, the penalty terms for the boundary

conditions and the magnetic source terms, similarly to equation (4.23).

The routines in the *NSSUS* flow solver that, for each iteration, compute the residuals based on the flow variables $\boldsymbol{w}$ use the following steps:

- Compute inviscid fluxes: For the given inviscid flux discretization the only flow variables, $\boldsymbol{w}$, that influence the residual at a node are the flow variables at that node and at the six nodes that are first neighbors of the node along each computational direction.

- Compute artificial dissipation fluxes: For each of the N_c nodes in the domain, compute the contributions of the flow variables to the residual at that node. For this portion of the residual, the flow variables in the current node and in the first- and second-neighbor nodes in each of the three coordinate directions need to be considered.

- Compute magnetic source terms: These only depend on the flow variables at the current node.

- Apply boundary conditions: Additional penalty terms are added to enforce the boundary conditions. Note that internal block boundaries are also considered as boundaries.

The stencil that affects the residual at given node, when considering the fluxes just mentioned, contains thirteen nodes, as shown in figure 4.8.

4.5 Time-integration

Since the cell volume, V_{ijk}, is independent of time, then the set of coupled ODEs (4.23) or (4.34) can be re-written in semi-discrete form as

$$\frac{d\boldsymbol{w}_{ijk}}{dt} + \mathcal{R}_{ijk}(\boldsymbol{w}) = 0\,, \tag{4.36}$$

where $\boldsymbol{w}$ is the vector of the flow variables at the cell center or vertex (corresponding to the FVM or FDM formulations, respectively). The vector of residuals, $\mathcal{R}(\boldsymbol{w})$,

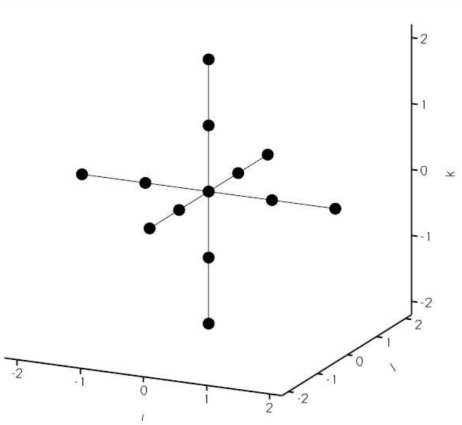


Figure 4.8: Stencil of dependence for the node-centered residual computation.

consists of all the physical and artificial fluxes, boundary conditions and residual terms and can be written as

$$\mathcal{R}_{ijk} = \frac{1}{V_{ijk}}(\mathcal{Q}_{ijk} - \mathcal{D}_{ijk}) - \mathbf{S}_{\mathbf{i}ijk}\,. \tag{4.37}$$

Since the primary objective was to obtain a steady-state solution, the time-marching scheme was selected for its simplicity, stability and damping properties. Therefore, an explicit multi-stage, Runge–Kutta scheme was used to integrate equation (4.36) in time.

A general m-stage Runge–Kutta scheme is of the form

$$\begin{aligned} \boldsymbol{w}^{(n+1,0)} &= \boldsymbol{w}^{n} \\ \boldsymbol{w}^{(n+1,1)} &= \boldsymbol{w}^{(0)} - \alpha_1 \Delta t \mathcal{R}^{(0)} \\ &\cdots \\ \boldsymbol{w}^{(n+1,m)} &= \boldsymbol{w}^{(0)} - \alpha_m \Delta t \mathcal{R}^{(m-1)} \\ \boldsymbol{w}^{n+1} &= \boldsymbol{w}^{(n+1,m)}, \end{aligned} \tag{4.38}$$

where $\boldsymbol{w}^n$ is the solution vector value at time step n, and the indexes ijk have been dropped out.

In a conventional scheme, the residual at the $(l-1)^{th}$ stage is evaluated as

$$\mathcal{R}^{(l)} = \frac{1}{V}\left(\mathcal{Q}(\boldsymbol{w}^{(l)}) - \mathcal{D}(\boldsymbol{w}^{(l)})\right). \tag{4.39}$$

However, better convergence is obtained if the dissipative part is a blend of new and old stage values

$$\mathcal{D}^{(l)} = \beta_l \mathcal{D}(\boldsymbol{w}^{(n+1,l)}) + (1-\beta_l)\mathcal{D}(\boldsymbol{w}^{(n+1,l-1)}). \tag{4.40}$$

Moreover, computational savings can be obtained by setting some of the β_l to zero.

In the present case, a five-stage, modified Runge–Kutta scheme was selected, using the standard coefficient values found in the literature [75, 100, 158] – $\alpha = (1/4, 1/6, 3/8, 1/2, 1)$, $\beta = (1, 0, 14/25, 0, 11/25)$.

Local Time Step

In order to increase the convergence rate of the algorithm, an adaptive local time step was used [5]. The time step is computed for each cell according to

$$\Delta t = \frac{CFL}{\lambda_\xi + \lambda_\eta + \lambda_\zeta}, \tag{4.41}$$

where CFL is the Courant–Friedrichs–Lewy number and λ_ξ , λ_η , λ_ζ are the maximum speed of propagation of information in each of the three computational directions, that corresponds to the maximum eigenvalue of the hyperbolic system of governing equations, also designated by local spectral radii.

The eigenvalues of the system of equations (4.5) can by found in appendix B. The spectral radius corresponds to the fast magneto-acoustic wave,

$$\lambda_{max} = |U_n| + c_f, \tag{4.42}$$

where U_n is the normal fluid velocity and c_f is the speed of the fast-mode MHD wave,

relative to the fluid which, according to equation (B.73), is given by

$$c_f = \sqrt{\frac{1}{2}\left[c^2 + \frac{B^2}{\rho\mu_m} + \sqrt{\left(c^2 + \frac{B^2}{\rho\mu_m}\right)^2 - 4\frac{c^2 B_n^2}{\rho\mu_m}}\right]}. \tag{4.43}$$

The speed of sound, assuming a perfect gas, is defined as $c = \sqrt{\frac{\gamma p}{\rho}}$.

Convergence criterion

The system of ODEs (4.36) is integrated in time until the solution evolves to a steady-state. As a criterion of convergence, the density residual was used,

$$\delta\rho^n = \sqrt{\sum_{ijk}\left(\frac{(\rho_{ijk}^n - \rho_{ijk}^{n-1})}{\Delta t\ V_{ijk}}\right)^2}, \tag{4.44}$$

and convergence was assumed when the residual dropped 8 to 10 orders of magnitude.

4.6 Boundary and initial conditions

Physical boundary conditions are applied after each update of the interior solution vector for the cell-centered flow solver, or they are introduced as penalty terms in the residual computation for the node-centered flow solver. Depending on the type of boundary, these conditions might be of the Dirichlet (e.g., no induced magnetic field) or Neumann (e.g., $\partial p/\partial n = 0$, extrapolation or symmetry) type. The types of boundaries handled have already been described in previous sections of this chapter.

For the multi-block solver *NSSUS*, the data exchange between computational blocks (halos and penalty terms) must also be done after each iteration.

As for the initial conditions, the density, pressure and velocity field are initialized with the free-stream values, while the induced magnetic field (if computed) is initialized to zero. The imposed magnetic field is chosen such that it satisfies Gauss's law, $\nabla \cdot \mathbf{B} = 0$ (4.2). This is easily guaranteed since it is obtained by superimposing dipoles, that individually satisfy the divergence-free condition.

Chapter 5

Discrete adjoint equations

The sensitivities required to solve the design problem schematically illustrated in figure 5.3 are computed by first assembling the discrete adjoint equations (3.30), solving them, and then using the total sensitivity equation (3.31).

Automatic differentiation tools are used, whenever deemed necessary and possible, to generate code that computes the several matrices of partial sensitivities present in these equations, according to the hybrid *ADjoint* approach described in section 3.6.

The next sections describe each of the steps required to implement the discrete adjoint solver for the two different MHD flow solvers described in sections 4.4.2 and 4.4.4. First, the discrete adjoint solver was developed for the FVM solver using a conventional approach, meaning that all the partial derivative terms required in the adjoint and total sensitivity equations were derived by manually differentiating the original flow code. On the other hand, after that first acquired experience with discrete adjoint solvers, the proposed hybrid approach for fast discrete adjoint solver implementation was tested on the sophisticated FDM solver: *NSSUS*.

5.1 Assembly of the adjoint matrix

The discrete adjoint system of equations (or flux Jacobian) matrix, $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$, is independent of the choice of the function of interest or the design variables – it is simply a function of the governing equations, their discretization and the problem boundary

conditions. To compute it, only the routines in the flow solver that evaluate the residuals, $\mathcal{R}_{ijk}$ (4.37), for each iteration, based on the values of the flow variables, $\boldsymbol{w}$, within the discretization need to be considered.

The residual at a computational cell (or node) depends only on a restricted set of cells (or nodes), usually designated by stencil. According to the three-dimensional spatial discretization used in the CFD solvers tested, described in section 4.4, the stencils of dependence for either the cell-centered solver (refer to figure 4.5) or the node-centered solver (refer to figure 4.8) have thirteen cells (or nodes). As such, the residual at a given cell (or node) of the discretized governing equations can, in general, be expressed as

$$\begin{aligned}\mathcal{R}_{ijk} = \mathcal{R}(&\boldsymbol{w}_{i-2}, \boldsymbol{w}_{j-2}, \boldsymbol{w}_{k-2}, \boldsymbol{w}_{i-1}, \boldsymbol{w}_{j-1}, \boldsymbol{w}_{k-1}, \boldsymbol{w}_{ijk}, \\ &\boldsymbol{w}_{i+1}, \boldsymbol{w}_{j+1}, \boldsymbol{w}_{k+1}, \boldsymbol{w}_{i+2}, \boldsymbol{w}_{j+2}, \boldsymbol{w}_{k+2}) \,.\end{aligned} \tag{5.1}$$

Because the residual applies to every computational cell (or node), it follows that the matrix $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ of the adjoint system of equations has dimensions $N_w^2 = (N_c \times N_v)^2$, where N_c is the number of cells/nodes and N_v the number of flow variables.

The adjoint matrix can then be assembled by taking the derivative of the residual $\mathcal{R}$ (5.1), that comprises all the numerical fluxes, with respect to the flow variables $\boldsymbol{w}$. Since a structured computational mesh is used, the result is a multi-diagonal matrix whose entries are block matrices of dimension N_v^2, making the global matrix very sparse, of known structured, and thus easily stored.

The number of non-zero block diagonals matches the dimension of the stencil used in the flow solver, N_s. These block matrices that are entries of the global adjoint matrix $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ for cell (or node) (i, j, k) are computed as

$$\begin{aligned} BB_{[m,n]} &= \frac{\partial \mathcal{R}_{ijk}(m)}{\partial \boldsymbol{w}_{i-2}(n)} \quad B_{[m,n]} = \frac{\partial \mathcal{R}_{ijk}(m)}{\partial \boldsymbol{w}_{i-1}(n)} \\ C_{[m,n]} &= \frac{\partial \mathcal{R}_{ijk}(m)}{\partial \boldsymbol{w}_{i+1}(n)} \quad CC_{[m,n]} = \frac{\partial \mathcal{R}_{ijk}(m)}{\partial \boldsymbol{w}_{i+2}(n)} \end{aligned} \,, \tag{5.2}$$

$$DD_{[m,n]} = \frac{\partial \mathcal{R}_{ijk}(m)}{\partial \boldsymbol{w}_{j-2}(n)} \quad D_{[m,n]} = \frac{\partial \mathcal{R}_{ijk}(m)}{\partial \boldsymbol{w}_{j-1}(n)}$$
$$E_{[m,n]} = \frac{\partial \mathcal{R}_{ijk}(m)}{\partial \boldsymbol{w}_{j+1}(n)} \quad EE_{[m,n]} = \frac{\partial \mathcal{R}_{ijk}(m)}{\partial \boldsymbol{w}_{j+2}(n)} \quad , \tag{5.3}$$

$$FF_{[m,n]} = \frac{\partial \mathcal{R}_{ijk}(m)}{\partial \boldsymbol{w}_{k-2}(n)} \quad F_{[m,n]} = \frac{\partial \mathcal{R}_{ijk}(m)}{\partial \boldsymbol{w}_{k-1}(n)}$$
$$G_{[m,n]} = \frac{\partial \mathcal{R}_{ijk}(m)}{\partial \boldsymbol{w}_{k+1}(n)} \quad GG_{[m,n]} = \frac{\partial \mathcal{R}_{ijk}(m)}{\partial \boldsymbol{w}_{k+2}(n)} \quad , \tag{5.4}$$

$$\text{and} \qquad A_{[m,n]} = \frac{\partial \mathcal{R}_{ijk}(m)}{\partial \boldsymbol{w}_{i}(n)} \, , \tag{5.5}$$

where equation (5.5) is the contribution from the cell at where the residual is being evaluated, and equations (5.2), (5.3) and (5.4) result from the contributions from the neighboring cells in the ξ,η, and ζ computational directions, respectively. In these expression, m spans the number of governing equations (N_v), and n spans the number of conservative variables (N_v).

In this work, the block Jacobians, (5.2) – (5.5), were obtained by both manual and automatic differentiation of the residual routines of the flow solver, as described in sections 5.1.1 and 5.1.2, respectively.

5.1.1 Manually differentiated Jacobian

As mentioned at the beginning of this chapter, manual differentiation was used to derive the discrete adjoint solver corresponding to the cell-centered flow solver 4.4.2.

The adjoint matrix can be constructed by adding up the contributions of each term, obtained by applying expressions (5.2) – (5.5) to the different residual contributions as expressed in equation (4.37).

Even though an adjoint for the second-order, three-dimensional flow solver was developed, the first-order case is shown here for illustration purposes. This means that a seven-cell stencil is assumed, with only a single level of neighboring cells, in the exposition that follows.

Inviscid fluxes

The contributions of the inviscid fluxes $\mathcal{Q}$ to the Jacobian are computed from expressions (4.20), (4.21) and (4.22), resulting in the seven non-zero block terms that follow.

The cell at which the residual is being evaluated, contributes to the adjoint Jacobian as

$$\begin{aligned} A_{ijk} = \frac{\partial \mathcal{Q}_{ijk}}{\partial \boldsymbol{w}_{ijk}} &= \frac{1}{2}\left[\frac{\partial E_{ijk}}{\partial \boldsymbol{w}_{ijk}}(Sx_{i+\frac{1}{2}} - Sx_{i-\frac{1}{2}})\right. \\ &\quad \left. + \frac{\partial F_{ijk}}{\partial \boldsymbol{w}_{ijk}}(Sy_{i+\frac{1}{2}} - Sy_{i-\frac{1}{2}}) + \frac{\partial G_{ijk}}{\partial \boldsymbol{w}_{ijk}}(Sz_{i+\frac{1}{2}} - Sz_{i-\frac{1}{2}})\right] \\ &+ \frac{1}{2}\left[\frac{\partial E_{ijk}}{\partial \boldsymbol{w}_{ijk}}(Sx_{j+\frac{1}{2}} - Sx_{j-\frac{1}{2}})\right. \\ &\quad \left. + \frac{\partial F_{ijk}}{\partial \boldsymbol{w}_{ijk}}(Sy_{j+\frac{1}{2}} - Sy_{j-\frac{1}{2}}) + \frac{\partial G_{ijk}}{\partial \boldsymbol{w}_{ijk}}(Sz_{j+\frac{1}{2}} - Sz_{j-\frac{1}{2}})\right] \\ &+ \frac{1}{2}\left[\frac{\partial E_{ijk}}{\partial \boldsymbol{w}_{ijk}}(Sx_{k+\frac{1}{2}} - Sx_{k-\frac{1}{2}})\right. \\ &\quad \left. + \frac{\partial F_{ijk}}{\partial \boldsymbol{w}_{ijk}}(Sy_{k+\frac{1}{2}} - Sy_{k-\frac{1}{2}}) + \frac{\partial G_{ijk}}{\partial \boldsymbol{w}_{ijk}}(Sz_{k+\frac{1}{2}} - Sz_{k-\frac{1}{2}})\right]. \end{aligned} \tag{5.6}$$

The neighbor cells along the ξ-direction computational direction (index i) contribute with

$$\begin{aligned} B_{ijk} &= \frac{\partial \mathcal{Q}_{ijk}}{\partial \boldsymbol{w}_{i-1,j,k}} \\ &= -\frac{1}{2}\left[\frac{\partial E_{i-1,j,k}}{\partial \boldsymbol{w}_{i-1,j,k}} Sx_{i-\frac{1}{2}} + \frac{\partial F_{i-1,j,k}}{\partial \boldsymbol{w}_{i-1,j,k}} Sy_{i-\frac{1}{2}} + \frac{\partial G_{i-1,j,k}}{\partial \boldsymbol{w}_{i-1,j,k}} Sz_{i-\frac{1}{2}}\right] \end{aligned} \tag{5.7}$$

and

$$\begin{aligned} C_{ijk} &= \frac{\partial \mathcal{Q}_{ijk}}{\partial \boldsymbol{w}_{i+1,j,k}} \\ &= \frac{1}{2}\left[\frac{\partial E_{i+1,j,k}}{\partial \boldsymbol{w}_{i+1,j,k}} Sx_{i+\frac{1}{2}} + \frac{\partial F_{i+1,j,k}}{\partial \boldsymbol{w}_{i+1,j,k}} Sy_{i+\frac{1}{2}} + \frac{\partial G_{i+1,j,k}}{\partial \boldsymbol{w}_{i+1,j,k}} Sz_{i+\frac{1}{2}}\right]. \end{aligned} \tag{5.8}$$

Considering the neighbor cells in the η-direction results in

$$\begin{aligned} D_{ijk} &= \frac{\partial \mathcal{Q}_{ijk}}{\partial \boldsymbol{w}_{i,j-1,k}} \\ &= -\frac{1}{2}\left[\frac{\partial E_{i,j-1,k}}{\partial \boldsymbol{w}_{i,j-1,k}} Sx_{j-\frac{1}{2}} + \frac{\partial F_{i,j-1,k}}{\partial \boldsymbol{w}_{i,j-1,k}} Sy_{j-\frac{1}{2}} + \frac{\partial G_{i,j-1,k}}{\partial \boldsymbol{w}_{i,j-1,k}} Sz_{j-\frac{1}{2}}\right] \end{aligned} \tag{5.9}$$

and

$$\begin{aligned} E_{ijk} &= \frac{\partial \mathcal{Q}_{ijk}}{\partial \boldsymbol{w}_{i,j+1,k}} \\ &= \frac{1}{2}\left[\frac{\partial E_{i,j+1,k}}{\partial \boldsymbol{w}_{i,j+1,k}} Sx_{j+\frac{1}{2}} + \frac{\partial F_{i,j+1,k}}{\partial \boldsymbol{w}_{i,j+1,k}} Sy_{j+\frac{1}{2}} + \frac{\partial G_{i,j+1,k}}{\partial \boldsymbol{w}_{i,j+1,k}} Sz_{j+\frac{1}{2}}\right]. \end{aligned} \tag{5.10}$$

Similarly in the ζ-direction, the contribution of the neighbor cells yields

$$\begin{aligned} F_{ijk} &= \frac{\partial \mathcal{Q}_{ijk}}{\partial \boldsymbol{w}_{i,j,k-1}} \\ &= -\frac{1}{2}\left[\frac{\partial E_{i,j,k-1}}{\partial \boldsymbol{w}_{i,j,k-1}} Sx_{k-\frac{1}{2}} + \frac{\partial F_{i,j,k-1}}{\partial \boldsymbol{w}_{i,j,k-1}} Sy_{k-\frac{1}{2}} + \frac{\partial G_{i,j,k-1}}{\partial \boldsymbol{w}_{i,j,k-1}} Sz_{k-\frac{1}{2}}\right] \end{aligned} \tag{5.11}$$

and

$$\begin{aligned} G_{ijk} &= \frac{\partial \mathcal{Q}_{ijk}}{\partial \boldsymbol{w}_{i,j,k+1}} \\ &= \frac{1}{2}\left[\frac{\partial E_{i,j,k+1}}{\partial \boldsymbol{w}_{i,j,k+1}} Sx_{k+\frac{1}{2}} + \frac{\partial F_{i,j,k+1}}{\partial \boldsymbol{w}_{i,j,k+1}} Sy_{k+\frac{1}{2}} + \frac{\partial G_{i,j,k+1}}{\partial \boldsymbol{w}_{i,j,k+1}} Sz_{k+\frac{1}{2}}\right]. \end{aligned} \tag{5.12}$$

Notice that there is not any inviscid contribution to BB_{ijk}, CC_{ijk}, DD_{ijk}, EE_{ijk}, FF_{ijk} or GG_{ijk} because a central-difference scheme is used, as detailed in expression (4.20).

As seen in expression (4.22), the fluxes $\mathbf{E}_{ijk}$, $\mathbf{F}_{ijk}$ and $\mathbf{G}_{ijk}$ are decomposed in the inviscid and ideal magnetic contributions as $\mathbf{E} = \mathbf{E}_i + \mathbf{E}_m$, $\mathbf{F} = \mathbf{F}_i + \mathbf{F}_m$ and $\mathbf{G} = \mathbf{G}_i + \mathbf{G}_m$, respectively, where the cell indexes have been dropped for convenience. Therefore, the block Jacobians are computed by parts as

$$\frac{\partial \mathbf{E}}{\partial \boldsymbol{w}} = \frac{\partial \mathbf{E}_i}{\partial \boldsymbol{w}} + \frac{\partial \mathbf{E}_m}{\partial \boldsymbol{w}}, \tag{5.13}$$

where the inviscid Jacobian matrix, $\frac{\partial \mathbf{E}_i}{\partial \boldsymbol{w}}$, and the ideal magnetic Jacobian matrix $\frac{\partial \mathbf{E}_m}{\partial \boldsymbol{w}}$, are given by expressions (B.85) and (B.86), respectively, that are found in section B.7 in appendix. Similar results apply to the fluxes in the $y-$ and $z-$directions, with the Jacobian matrices given by expressions (B.89), (B.90), (B.93) and (B.94).

Artificial dissipation

Similarly to what was done for the convective residual $\mathcal{Q}$ of the governing equations, the Jacobian of the artificial dissipation residual $\mathcal{D}$ with respect to the conservative variables $\boldsymbol{w}$ has also to be evaluated. Plugging in the JST scheme (4.25), (4.26) and (4.27) into the artificial dissipation residual defined by (4.24) results in

$$
\begin{aligned}
\mathcal{D}_{ijk}(\boldsymbol{w}) &= d_{i+\frac{1}{2},j,k} - d_{i-\frac{1}{2},j,k} + d_{i,j+\frac{1}{2},k} - d_{i,j-\frac{1}{2},k} + d_{i,j,k+\frac{1}{2}} - d_{i,j,k-\frac{1}{2}} \\
&= \lambda_{i+\frac{1}{2},j,k} \Big[\varepsilon^{(2)}_{i+\frac{1}{2},j,k} (\boldsymbol{w}_{i+1,j,k} - \boldsymbol{w}_{i,j,k}) \\
&\qquad + \varepsilon^{(4)}_{i+\frac{1}{2},j,k} (\boldsymbol{w}_{i+2,j,k} - 3\boldsymbol{w}_{i+1,j,k} + 3\boldsymbol{w}_{i,j,k} - \boldsymbol{w}_{i-1,j,k}) \Big] \\
&- \lambda_{i-\frac{1}{2},j,k} \Big[\varepsilon^{(2)}_{i-\frac{1}{2},j,k} (\boldsymbol{w}_{i,j,k} - \boldsymbol{w}_{i-1,j,k}) \\
&\qquad + \varepsilon^{(4)}_{i-\frac{1}{2},j,k} (\boldsymbol{w}_{i+1,j,k} - 3\boldsymbol{w}_{i,j,k} + 3\boldsymbol{w}_{i-1,j,k} - \boldsymbol{w}_{i-2,j,k}) \Big] \\
&+ \lambda_{i,j+\frac{1}{2},k} \Big[\varepsilon^{(2)}_{i,j+\frac{1}{2},k} (\boldsymbol{w}_{i,j+1,k} - \boldsymbol{w}_{i,j,k}) \\
&\qquad + \varepsilon^{(4)}_{i,j+\frac{1}{2},k} (\boldsymbol{w}_{i,j+2,k} - 3\boldsymbol{w}_{i,j+1,k} + 3\boldsymbol{w}_{i,j,k} - \boldsymbol{w}_{i,j-1,k}) \Big] \\
&- \lambda_{i,j-\frac{1}{2},k} \Big[\varepsilon^{(2)}_{i,j-\frac{1}{2},k} (\boldsymbol{w}_{i,j,k} - \boldsymbol{w}_{i,j-1,k}) \\
&\qquad + \varepsilon^{(4)}_{i,j-\frac{1}{2},k} (\boldsymbol{w}_{i,j+1,k} - 3\boldsymbol{w}_{i,j,k} + 3\boldsymbol{w}_{i,j-1,k} - \boldsymbol{w}_{i,j-2,k}) \Big] \\
&+ \lambda_{i,j,k+\frac{1}{2}} \Big[\varepsilon^{(2)}_{i,j,k+\frac{1}{2}} (\boldsymbol{w}_{i,j,k+1} - \boldsymbol{w}_{i,j,k}) \\
&\qquad + \varepsilon^{(4)}_{i,j,k+\frac{1}{2}} (\boldsymbol{w}_{i,j,k+2} - 3\boldsymbol{w}_{i,j,k+1} + 3\boldsymbol{w}_{i,j,k} - \boldsymbol{w}_{i,j,k-1}) \Big] \\
&- \lambda_{i,j,k-\frac{1}{2}} \Big[\varepsilon^{(2)}_{i,j,k-\frac{1}{2}} (\boldsymbol{w}_{i,j,k} - \boldsymbol{w}_{i,j,k-1}) \\
&\qquad + \varepsilon^{(4)}_{i,j,k-\frac{1}{2}} (\boldsymbol{w}_{i,j,k+1} - 3\boldsymbol{w}_{i,j,k} + 3\boldsymbol{w}_{i,j,k-1} - \boldsymbol{w}_{i,j,k-2}) \Big] \, .
\end{aligned}
\tag{5.14}
$$

From the expression above, the Jacobian of the artificial dissipation residual for each cell of the stencil centered at (i, j, k) can be computed. The spectral radius λ is assumed to be constant, since its contribution is considered negligible, making the hand-differentiation of expression (5.14) considerably simplified. This is, however, an

approximation that is made in hand-differentiating the terms of a discrete adjoint solver. This and other kinds of approximations are not necessary when the ADjoint methodology is followed.

Since the artificial dissipation stencil extends five cells in each computational direction, and recognizing from the JST algorithm [78] that $\varepsilon^{(2)}_{i+\frac{1}{2}} = f(\boldsymbol{w}_{i+2}, \boldsymbol{w}_{i+1}, \boldsymbol{w}_i, \boldsymbol{w}_{i-1})$ and $\varepsilon^{(4)}_{i+\frac{1}{2}} = g(\varepsilon^{(2)}_{i+\frac{1}{2}})$, there are 13 non-zero contributions to the Jacobian. The contribution from the cells along the ξ-direction are shown next¿ The remaining ones along the η- and ζ-directions are easily derived from the former.

The center cell contributes to the artificial residual with

$$\begin{aligned} A_{ijk} &= \frac{\partial \mathcal{D}_{ijk}}{\partial \boldsymbol{w}_{i,j,k}} \\ &= \lambda_{i+\frac{1}{2}} \left[\left(-\varepsilon^{(2)}_{i+\frac{1}{2}} + 3\varepsilon^{(4)}_{i+\frac{1}{2}} \right) \mathbf{I} + \frac{\partial \varepsilon^{(2)}_{i+\frac{1}{2}}}{\partial \boldsymbol{w}_i} \Delta_F \boldsymbol{w}_i + \frac{\partial \varepsilon^{(4)}_{i+\frac{1}{2}}}{\partial \boldsymbol{w}_i} \Delta_F \Delta_B \Delta_F \boldsymbol{w}_i \right] \\ &- \lambda_{i-\frac{1}{2}} \left[\left(\varepsilon^{(2)}_{i-\frac{1}{2}} - 3\varepsilon^{(4)}_{i-\frac{1}{2}} \right) \mathbf{I} + \frac{\partial \varepsilon^{(2)}_{i-\frac{1}{2}}}{\partial \boldsymbol{w}_i} \Delta_F \boldsymbol{w}_{i-1} + \frac{\partial \varepsilon^{(4)}_{i-\frac{1}{2}}}{\partial \boldsymbol{w}_i} \Delta_F \Delta_B \Delta_F \boldsymbol{w}_{i-1} \right] . \end{aligned} \tag{5.15}$$

The contributions to the residual at (i, j, k) coming from the cells to the left are

$$\begin{aligned} BB_{ijk} &= \frac{\partial \mathcal{D}_{ijk}}{\partial \boldsymbol{w}_{i-2,j,k}} \\ &= -\lambda_{i-\frac{1}{2}} \left[\left(-\varepsilon^{(4)}_{i-\frac{1}{2}} \right) \mathbf{I} + \frac{\partial \varepsilon^{(2)}_{i-\frac{1}{2}}}{\partial \boldsymbol{w}_{i-2}} \Delta_F \boldsymbol{w}_{i-1} + \frac{\partial \varepsilon^{(4)}_{i-\frac{1}{2}}}{\partial \boldsymbol{w}_{i-2}} \Delta_F \Delta_B \Delta_F \boldsymbol{w}_{i-1} \right] \end{aligned} \tag{5.16}$$

and

$$\begin{aligned} B_{ijk} &= \frac{\partial \mathcal{D}_{ijk}}{\partial \boldsymbol{w}_{i-1,j,k}} \\ &= \lambda_{i+\frac{1}{2}} \left[\left(-\varepsilon^{(4)}_{i+\frac{1}{2}} \right) \mathbf{I} + \frac{\partial \varepsilon^{(2)}_{i+\frac{1}{2}}}{\partial \boldsymbol{w}_{i-1}} \Delta_F \boldsymbol{w}_i + \frac{\partial \varepsilon^{(4)}_{i+\frac{1}{2}}}{\partial \boldsymbol{w}_{i-1}} \Delta_F \Delta_B \Delta_F \boldsymbol{w}_i \right] \\ &- \lambda_{i-\frac{1}{2}} \left[\left(-\varepsilon^{(2)}_{i-\frac{1}{2}} + 3\varepsilon^{(4)}_{i-\frac{1}{2}} \right) \mathbf{I} + \frac{\partial \varepsilon^{(2)}_{i-\frac{1}{2}}}{\partial \boldsymbol{w}_{i-1}} \Delta_F \boldsymbol{w}_{i-1} + \frac{\partial \varepsilon^{(4)}_{i-\frac{1}{2}}}{\partial \boldsymbol{w}_{i-1}} \Delta_F \Delta_B \Delta_F \boldsymbol{w}_{i-1} \right] , \end{aligned} \tag{5.17}$$

whereas the contributions from the right neighbor cells leads to

$$\begin{aligned} C_{ijk} &= \frac{\partial \mathcal{D}_{ijk}}{\partial \boldsymbol{w}_{i+1,j,k}} \\ &= \lambda_{i+\frac{1}{2}} \left[\left(\varepsilon^{(2)}_{i+\frac{1}{2}} - 3\varepsilon^{(4)}_{i+\frac{1}{2}} \right) \mathbf{I} + \frac{\partial \varepsilon^{(2)}_{i+\frac{1}{2}}}{\partial \boldsymbol{w}_{i+1}} \Delta_F \boldsymbol{w}_i + \frac{\partial \varepsilon^{(4)}_{i+\frac{1}{2}}}{\partial \boldsymbol{w}_{i+1}} \Delta_F \Delta_B \Delta_F \boldsymbol{w}_i \right] \\ &- \lambda_{i-\frac{1}{2}} \left[\left(\varepsilon^{(4)}_{i-\frac{1}{2}} \right) \mathbf{I} + \frac{\partial \varepsilon^{(2)}_{i-\frac{1}{2}}}{\partial \boldsymbol{w}_{i+1}} \Delta_F \boldsymbol{w}_{i-1} + \frac{\partial \varepsilon^{(4)}_{i-\frac{1}{2}}}{\partial \boldsymbol{w}_{i+1}} \Delta_F \Delta_B \Delta_F \boldsymbol{w}_{i-1} \right] \end{aligned} \tag{5.18}$$

and

$$\begin{aligned} CC_{ijk} &= \frac{\partial \mathcal{D}_{ijk}}{\partial \boldsymbol{w}_{i+2,j,k}} \\ &= \lambda_{i+\frac{1}{2}} \left[\left(\varepsilon^{(4)}_{i+\frac{1}{2}} \right) \mathbf{I} + \frac{\partial \varepsilon^{(2)}_{i+\frac{1}{2}}}{\partial \boldsymbol{w}_{i+2}} \Delta_F \boldsymbol{w}_i + \frac{\partial \varepsilon^{(4)}_{i+\frac{1}{2}}}{\partial \boldsymbol{w}_{i+2}} \Delta_F \Delta_B \Delta_F \boldsymbol{w}_i \right] . \end{aligned} \tag{5.19}$$

Source terms

The magnetic field source term contribution $\mathcal{S}$ to the adjoint system, because it is a volume and not a flux term, as given by equation (4.14), it only contributes to the adjoint matrix with

$$A_{ijk} = \frac{\partial \mathcal{S}_{ijk}}{\partial \boldsymbol{w}_{ijk}} . \tag{5.20}$$

Boundary conditions

Special care is taken at the boundaries, where the cells have stencils that extend outside the internal computational domain. There, the chain rule is used to take into account the dependence of the halo cells on the interior cells, according to the type of boundary condition used in the flow solver. This happens because, similarly to the flow solver, the adjoint system is only solved for the interior cells and so all the exterior boundary cell values have to be written as functions of the interior domain.

As an illustration, consider a cell at boundary $i_{min}(i = 2)$. In this case, the value of the conservative vector, $\boldsymbol{w}$, at the halo cell (corresponding to $i = 1$) is a function of the interior cells,

$$\boldsymbol{w}_{i=1} = f\left(\boldsymbol{w}_{i=2}, \boldsymbol{w}_{i=3}\right) . \tag{5.21}$$

Therefore, there is an additional contribution to the block matrices corresponding to the Jacobian of the residual at cell i_{min}, given by

$$B_{i=2}[m,n] = \frac{\partial \mathcal{R}_{i=2}(m)}{\partial \boldsymbol{w}_{i=1}(n)} = 0 \tag{5.22}$$

$$A_{i=2}[m,n] = \frac{\partial \mathcal{R}_{i=2}(m)}{\partial \boldsymbol{w}_{i=2}(n)} = \frac{\partial \mathcal{R}_{i=2}(m)}{\partial \boldsymbol{w}_{i=1}(l)} \frac{\partial \boldsymbol{w}_{i=1}(l)}{\partial \boldsymbol{w}_{i=2}(n)} \tag{5.23}$$

$$C_{i=2}[m,n] = \frac{\partial \mathcal{R}_{i=2}(m)}{\partial \boldsymbol{w}_{i=3}(n)} = \frac{\partial \mathcal{R}_{i=2}(m)}{\partial \boldsymbol{w}_{i=1}(l)} \frac{\partial \boldsymbol{w}_{i=1}(l)}{\partial \boldsymbol{w}_{i=3}(n)} , \tag{5.24}$$

where the auxiliary partial derivatives $\frac{\partial \boldsymbol{w}_{i=1}}{\partial \boldsymbol{w}_{i=2}}$ and $\frac{\partial \boldsymbol{w}_{i=1}}{\partial \boldsymbol{w}_{i=3}}$ are obtained from the flow solver boundary condition routines.

Taking the case of an Euler (inviscid) solid wall boundary (refer to figure 4.3), the residual at a cell $i = 2$ can be expressed as

$$R_{(i=2)} = R^0 - R^{BC}_{i=2} , \tag{5.25}$$

where R^0 is the residual as if it were an internal cell, provided that permeability is used to cancel it out at these cells close to the walls. The additional term $R^{BC}_{i=2}$ reflects

the effect of the wall on the numerical flux, as given by expression (4.29),

$$R_{i=2}^{BC} = (\boldsymbol{\mathcal{F}} \cdot \mathbf{S})_{wall} = \begin{pmatrix} 0 \\ p_w S_x \\ p_w S_y \\ p_w S_z \\ 0 \end{pmatrix} . \tag{5.26}$$

The effect of the wall boundary on the adjoint matrix is then obtained by differentiating the residual (5.26) with respect to the conservative variable vector, resulting in

$$\frac{\partial R_{i=2}^{BC}}{\partial \boldsymbol{w}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{\partial p_w}{\partial w_1} S_x & \frac{\partial p_w}{\partial w_2} S_x & \frac{\partial p_w}{\partial w_3} S_x & \frac{\partial p_w}{\partial w_4} S_x & \frac{\partial p_w}{\partial w_5} S_x \\ \frac{\partial p_w}{\partial w_1} S_y & \frac{\partial p_w}{\partial w_2} S_y & \frac{\partial p_w}{\partial w_3} S_y & \frac{\partial p_w}{\partial w_4} S_y & \frac{\partial p_w}{\partial w_5} S_y \\ \frac{\partial p_w}{\partial w_1} S_z & \frac{\partial p_w}{\partial w_2} S_z & \frac{\partial p_w}{\partial w_3} S_z & \frac{\partial p_w}{\partial w_4} S_z & \frac{\partial p_w}{\partial w_5} S_z \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} . \tag{5.27}$$

Since $p_w = p_w(p_{i=2}, p_{i=3})$, the only non-zero contributions to the Jacobian are given by $\frac{\partial R_{i=2}^{BC}}{\partial \boldsymbol{w}_{i=2}}$ and $\frac{\partial R_{i=2}^{BC}}{\partial \boldsymbol{w}_{i=3}}$.

Recalling the approximation of the pressure at the wall (4.31), results

$$\frac{\partial p_w}{\partial \boldsymbol{w}} = \frac{\partial}{\partial \boldsymbol{w}} p_w = \frac{3}{2} \frac{\partial p_{(i=2)}}{\partial \boldsymbol{w}} - \frac{1}{2} \frac{\partial p_{(i=3)}}{\partial \boldsymbol{w}} . \tag{5.28}$$

Therefore, the Jacobian (5.27) leads to two contributions to the adjoint matrix,

$$A_{i=2}^{BC} = \frac{\partial R_{i=2}^{BC}}{\partial \boldsymbol{w}_{i=2}} = \frac{3}{2} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{\partial p_{i=2}}{\partial w_1} S_x & \frac{\partial p_{i=2}}{\partial w_2} S_x & \frac{\partial p_{i=2}}{\partial w_3} S_x & \frac{\partial p_{i=2}}{\partial w_4} S_x & \frac{\partial p_{i=2}}{\partial w_5} S_x \\ \frac{\partial p_{i=2}}{\partial w_1} S_y & \frac{\partial p_{i=2}}{\partial w_2} S_y & \frac{\partial p_{i=2}}{\partial w_3} S_y & \frac{\partial p_{i=2}}{\partial w_4} S_y & \frac{\partial p_{i=2}}{\partial w_5} S_y \\ \frac{\partial p_{i=2}}{\partial w_1} S_z & \frac{\partial p_{i=2}}{\partial w_2} S_z & \frac{\partial p_{i=2}}{\partial w_3} S_z & \frac{\partial p_{i=2}}{\partial w_4} S_z & \frac{\partial p_{i=2}}{\partial w_5} S_z \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.29}$$

and

$$E^{BC}_{i=2} = \frac{\partial R^{BC}_{i=2}}{\partial \boldsymbol{w}_{i=2}} = -\frac{1}{2}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{\partial p_{i=3}}{\partial w_1} S_x & \frac{\partial p_{i=3}}{\partial w_2} S_x & \frac{\partial p_{i=3}}{\partial w_3} S_x & \frac{\partial p_{i=3}}{\partial w_4} S_x & \frac{\partial p_{i=3}}{\partial w_5} S_x \\ \frac{\partial p_{i=3}}{\partial w_1} S_y & \frac{\partial p_{i=3}}{\partial w_2} S_y & \frac{\partial p_{i=3}}{\partial w_3} S_y & \frac{\partial p_{i=3}}{\partial w_4} S_y & \frac{\partial p_{i=3}}{\partial w_5} S_y \\ \frac{\partial p_{i=3}}{\partial w_1} S_z & \frac{\partial p_{i=3}}{\partial w_2} S_z & \frac{\partial p_{i=3}}{\partial w_3} S_z & \frac{\partial p_{i=3}}{\partial w_4} S_z & \frac{\partial p_{i=3}}{\partial w_5} S_z \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad (5.30)$$

where the pressure gradients are given by expression (B.79) evaluated at cells $i = 2$ and $i = 3$, respectively.

Assembled matrix $\partial\mathcal{R}/\partial\boldsymbol{w}$

Gathering all the Jacobian contributions stated previously, the adjoint matrix $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ gets finally assembled, as indicated in figure (5.1).

5.1.2 Automatically differentiated Jacobian

While the derivation of the non-zero block matrix entries of $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ of the discrete adjoint system of equations (3.30) were previously accomplished by manual differentiation of the CFD solver routines that evaluated the residual $\mathcal{R}$ of the discretized governing equations (4.23) or (4.34), this task is here performed by using *Automatic Differentiation* (AD) tools, taking advantage of using a discrete adjoint approach.

In this case, the adjoint solver implementation was done on the sophisticated, node-centered, flow solver *NSSUS*, described in section 4.4.4. Obviously, all the lessons learned from the previous manual implementation were re-used, namely the infrastructure necessary to assemble and solver the adjoint equations.

This move from a tedious, time-consuming, and error-prone hand differentiation technique to the use of an AD tool brings additional advantages, such as: 1) it speeds up the derivation of the adjoint system of equations considerably; 2) it generates the numerically exact Jacobians (with no need for approximations that may impact the accuracy of the sensitivities); and 3) it takes care of the boundary conditions automatically.

$$
\begin{bmatrix}
\ddots & \ddots & 0 & \ddots & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
B_{i,j,k-1} & A_{i,j,k-1} & C_{i,j,k-1} & 0 & E_{i,j,k-1} & 0 & G_{i,j,k-1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \ddots & \ddots & \ddots & 0 & \ddots & 0 & \ddots & 0 & 0 & 0 & 0 & 0 \\
D_{i,j-1,k} & 0 & B_{i,j-1,k} & A_{i,j-1,k} & C_{i,j-1,k} & 0 & E_{i,j-1,k} & 0 & G_{i,j-1,k} & 0 & 0 & 0 & 0 \\
0 & \ddots & 0 & \ddots & \ddots & \ddots & 0 & \ddots & 0 & \ddots & 0 & 0 & 0 \\
F_{i-1,j,k} & 0 & D_{i-1,j,k} & 0 & B_{i-1,j,k} & A_{i-1,j,k} & C_{i-1,j,k} & 0 & E_{i-1,j,k} & 0 & G_{i-1,j,k} & 0 & 0 \\
0 & F_{ijk} & 0 & D_{ijk} & 0 & B_{ijk} & A_{ijk} & C_{ijk} & 0 & E_{ijk} & 0 & G_{ijk} & 0 \\
0 & 0 & F_{i+1,j,k} & 0 & D_{i+1,j,k} & 0 & B_{i+1,j,k} & A_{i+1,j,k} & C_{i+1,j,k} & 0 & E_{i+1,j,k} & 0 & G_{i+1,j,k} \\
0 & 0 & 0 & \ddots & 0 & \ddots & 0 & \ddots & \ddots & \ddots & 0 & \ddots & 0 \\
0 & 0 & 0 & 0 & F_{i,j+1,k} & 0 & D_{i,j+1,k} & 0 & B_{i,j+1,k} & A_{i,j+1,k} & C_{i,j+1,k} & 0 & E_{i,j+1,k} \\
0 & 0 & 0 & 0 & 0 & \ddots & 0 & \ddots & 0 & \ddots & \ddots & \ddots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & F_{i,j,k+1} & 0 & D_{i,j,k+1} & 0 & B_{i,j,k+1} & A_{i,j,k+1} & C_{i,j,k+1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & \ddots & 0 & \ddots & \ddots
\end{bmatrix}
\begin{Bmatrix}
\vdots \\
\boldsymbol{w}_{i,j,k-1} \\
\vdots \\
\boldsymbol{w}_{i,j-1,k} \\
\vdots \\
\boldsymbol{w}_{i-1,j,k} \\
\boldsymbol{w}_{i,j,k} \\
\boldsymbol{w}_{i+1,j,k} \\
\vdots \\
\boldsymbol{w}_{i,j+1,k} \\
\vdots \\
\boldsymbol{w}_{i,j,k+1} \\
\vdots
\end{Bmatrix}
$$

Figure 5.1: Assembled adjoint matrix $\partial \mathcal{R}/\partial \boldsymbol{w}$.

This procedure might be effortlessly used with arbitrarily complex governing equations, provided that a flow solver has already been coded.

As mentioned in chapter 3, Tapenade was chosen as the automatic differentiation tool because it is currently the only non-commercial tool that supports Fortran 90, which is the programming language used in both flow solvers tested.

The baseline implementation of the *NSSUS* flow solver computes the residual using nested loops over the coordinate directions of each computational block on each of the processors in the calculation. To make the implementation of the discrete adjoint solver more efficient, it was necessary to re-write the flow residual routine such that it computed the residual for a single specified node, based on the stencil shown in figure 4.8. These routines were developed by cutting and pasting from the original flow solver routines, a process which turned out to be straightforward and that could be completed in less than a week of work, including the proper boundary condition handling. The re-engineered residual routine is a function with the residuals at a given node, `rAdj`, returned as an output argument, and the stencil of flow variables, `wAdj`, that affect the residuals at that node is provided as an input argument,

$$\texttt{subroutine residualAdj}(\texttt{wAdj}, \texttt{rAdj}, \texttt{i}, \texttt{j}, \texttt{k})\,, \tag{5.31}$$

for each node in each block on each processor. Any required Fortran pointers were set before calling the routine due to the current pointer handling limitations in Tapenade.

The stencil of flow variables `wAdj` that affects the residual `rAdj` in a given node `(i,j,k)` extends two nodes in each direction to allow for a second-order discretization as shown in figure 4.8. In this case, the number of nodes in the stencil whose variables affect the residual of a given node is $N_s = 13$. This re-engineered residual routine `residualAdj` computes N_v residuals in a given node that depend on all $(N_v \times N_s)$ flow variables in the stencil.

The boundary condition penalty terms were moved to a separate routine that also had the boundary sub-face `mm` and the corresponding flow variable that was the donor

to the penalty state `wDonorAdj` as input parameters,

$$\texttt{subroutine residualPenaltyAdj}(\texttt{wAdj}, \texttt{rAdj}, \texttt{mm}, \texttt{wDonorAdj}, \texttt{i}, \texttt{j}, \texttt{k})\,. \qquad (5.32)$$

This penalty residual routine was only called when the given node `(i,j,k)` was located at a boundary face.

In order to properly treat the multi-block grid, it was necessary to setup a global node numbering scheme and preserve the node connectivity. This allowed for the proper block boundary treatment when assembling the flux Jacobian matrix.

Having verified the re-written residual routines by comparing the residual values to the ones computed with the original code, these routines were then fed into Tapenade for differentiation. For the reasons explained in section 3.6, since for this stencil the number of input variables, $\boldsymbol{w}$, is significantly larger than the number of output variables, $\mathcal{R}$, automatic differentiation was performed using the reverse mode on the set of routines `residualAdj` and `residualPenaltyAdj`. The automatic differentiation process produced the differentiated routines

```
subroutine residualAdj_B(wAdj,wAdjB,rAdj,rAdjB,i,j,k)
```

and

```
subroutine residualPenaltyAdj_B(wAdj,wAdjB,rAdj,rAdjB,mm,wDonorAdj,wDonorAdjB,i,j,k) ,
```

which are able to compute, using the reverse mode of automatic differentiation, all the necessary derivatives for the flux Jacobian matrix.

The $N_v \times (N_v \times N_s)$ sensitivities that need to be computed for each node, corresponding to N_v rows in the $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ matrix, are readily computed by these automatically differentiated routines in reverse mode. This is accomplished because all the derivatives in the stencil can be calculated from one residual evaluation `residualAdj_B` since

$$\texttt{wAdjB}(\texttt{ii}, \texttt{jj}, \texttt{kk}, \texttt{n}) = \frac{\partial \mathcal{R}(i,j,k,m)}{\partial \boldsymbol{w}(i+ii, j+jj, k+kk, n)}\,, \qquad (5.33)$$

where the triad `(ii,jj,kk)` spans the stencil and `n` spans the N_v flow variables.

Similarly, the penalty term contributions are given by the outputs `wAdjB` and `wDonorAdjB` of `residualPenaltyAdj_B`. Referring to figure 4.6, the evaluation of the Jacobian of the residual at a boundary node, when running `residualPenaltyAdj_B`

in block A, gets some contributions from block B, corresponding to penalty data donated, `wDonorAdj`. These additional contributions are returned by `wDonorAdjB` and must be taken into account in the adjoint matrix assembly, so that they are included in the rows corresponding to boundary nodes and columns corresponding to the donor nodes.

Since these AD routines evaluate an entire row of the flux Jacobian at one time, setting `rAdjB(m)=1`, they allow for an easy assembly of the matrix $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$, as shown in the routine pseudo-code in figure 5.2.

```
subroutine setupADjointMatrix
(...)
! Loop over the local computational blocks.
do nn=1,nDom
  ! Loop over location of output (R) cell of residual
  do k=1,kl
    do j=1,jl
      do i=1,il
        ! Global node number
        idxmgb = globalNode(i,j,k)
        ! Transfer state w to auxiliar stencil array wAdj(:,:,:,:)
        call copyADjointStencil(wAdj,i,j,k)
        ! Loop over the outputs (R)
        do m=1,nwFlow
          ! Initialize the seed for the reverse mode to return dR(m)/dw
          rAdjB(:)=zero; rAdjB(m)=one; rAdj(:)=zero; wAdjB(:,:,:,:)=zero
          ! Call reverse mode of residual computation
          call residualAdj_B(wAdj,wAdjB,rAdj,rAdjB,i,j,k)
          ! Store block Jacobians (by rows).
          Aad(m,:) = wAdjB(0,0,0,:); Bad(m,:) = wAdjB(-1,0,0,:)
          (...)
        enddo
        ! Transfer block Jacobians to PETSc matrix.
        ! >>> center block A = dR(i,j,k)/w(i,j,k)
        idxngb = idxmgb
        call MatSetValuesBlocked(dRdW,1,idxmgb,1,idxngb,Aad,INSERT_VALUES,ierr)
        ! >>> west block B < w(i-1,j,k)
        idxngb = globalNode(i-1,j,k)
        call MatSetValuesBlocked(dRdW,1,idxmgb,1,idxngb,Bad,INSERT_VALUES,ierr)
        (...)
```

Figure 5.2: Flux Jacobian matrix assembly routine.

5.2 Assembly of the adjoint RHS vector

So far, only the left-hand side of the adjoint system of equations (3.30) has been considered.

The RHS vector of the discrete adjoint system of equations, $\frac{\partial I}{\partial \boldsymbol{w}}$, is constructed by differentiating the discretized version of the function of interest (3.14) with respect to the flow variables $\boldsymbol{w}$. This vector has length $N_w = (N_c \times N_v)$ and, depending on the chosen function I, it might be very sparse. Since the most common functions of interest – such as the aerodynamic coefficients of drag, C_D, lift, C_L and moment, C_M, used in this work – result from surface integral evaluations, that is in fact the case.

The choice of function I is problem-dependent but, regardless of the actual function being use, it can always be expressed in the form $I = I(\boldsymbol{w})$. It is important to understand how the flow solution $\boldsymbol{w}$ influences it to compute the contributions $\frac{\partial I}{\partial \boldsymbol{w}}$. As an illustration, the derivation of the RHS adjoint vector for the inviscid drag coefficient C_D is shown next.

By definition, the drag coefficient is the drag force that acts on a body, made dimensionless by a reference force that is derived from the dynamic pressure and a reference area. The pressure drag force D (inviscid) is computed by integrating the pressure over the body surface in the direction of the flow, resulting in

$$C_D = \frac{D}{q_\infty S_{ref}} = \frac{1}{q_\infty S_{ref}} \iint_S p_w \hat{\mathbf{n}}_D \cdot d\boldsymbol{S}\,, \tag{5.34}$$

where the drag direction is defined as $\hat{\mathbf{n}}_D = -\frac{\boldsymbol{u}_\infty}{\|\boldsymbol{u}_\infty\|} = (n_{Dx}, n_{Dy}, n_{Dz})$ (the minus sign takes into account that positive drag opposes the movement), p_w is the wall pressure, S_{ref} is the reference area and the dynamic pressure is given by $q_\infty = \frac{1}{2}\rho_\infty U_\infty^2 = \frac{\gamma}{2} p_\infty M_\infty^2$.

The discretization of C_D (5.34) in the computational domain, assuming that the body surface occurs at $i = 1$, can be expressed as

$$C_D = \frac{1}{q_\infty S_{ref}} \sum_j \sum_k p_{w_{jk}}\, \hat{n}_D \cdot \boldsymbol{S}_{w_{jk}}\,, \tag{5.35}$$

where $\boldsymbol{S}_{w_{jk}}$ is the normal-oriented surface area of the $(1, j, k)$ cell.

Using the inviscid drag coefficient C_D (5.35) as the cost function, its sensitivity with respect to the conservative variables can then be easily computed as

$$\frac{\partial I}{\partial \boldsymbol{w}} = \frac{\partial C_D}{\partial \boldsymbol{w}} = \frac{1}{q_\infty S_{ref}} \sum_j \sum_k \frac{\partial p_{w_{j,k}}}{\partial \boldsymbol{w}} \hat{\mathbf{n}}_D \cdot \boldsymbol{S}_{w_{jk}} \,, \tag{5.36}$$

where the pressure at the wall p_w is computed using the same boundary condition method as in the flow solver to ensure consistency.

For the cell-centered, custom MHD solver 4.4.2, p_w is computed using the linear extrapolation approximation of the solid wall boundary condition (4.31),then $p_w = p_w(p_2, p_3)$. Consequently, the only non-zero terms are given at cells $i = 2$ and $i = 3$, which are given by

$$\frac{\partial I}{\partial \boldsymbol{w}_{2,j,k}} = \frac{1}{q_\infty S_{ref}} \frac{3}{2} \frac{\partial p_{2,j,k}}{\partial \boldsymbol{w}_{2,j,k}} \hat{\mathbf{n}}_D \cdot \boldsymbol{S}_{w_{jk}} \tag{5.37}$$

and

$$\frac{\partial I}{\partial \boldsymbol{w}_{3,j,k}} = -\frac{1}{q_\infty S_{ref}} \frac{1}{2} \frac{\partial p_{3,j,k}}{\partial \boldsymbol{w}_{3,j,k}} \hat{\mathbf{n}}_D \cdot \boldsymbol{S}_{w_{jk}} \tag{5.38}$$

with the gradients $\frac{\partial p}{\partial \boldsymbol{w}}$, given by the expressions (B.79), evaluated at cells $i = 2, 3$.

For the *NSSUS* solver, because this specific flow solver works with primitive variables, $\boldsymbol{w} = (\rho, u, v, w, p)$, then the derivative $\partial C_D / \partial \boldsymbol{w}$ is always zero except for $w_5 (= p)$. Therefore, it became trivial to derive analytically the expression for this partial derivative from the flow solver routine that calculated that function of interest. The RHS of the adjoint for the ideal MHD model is simply expressed as

$$\frac{\partial C_D}{\partial \boldsymbol{w}} = \left\{ 0, 0, 0, 0, \frac{\partial C_D}{\partial p}, 0, 0, 0 \right\} . \tag{5.39}$$

It is always worth mentioning that the discrete adjoint formulation allows any cost function to be treated in a similar same fashion, independently of its form, in contrast to the continuous adjoint formulation. In general, when using AD differentiated functions of interest, the assembly of the adjoint vector is blind in terms of sparsity. This just follows naturally from the automatically generated differentiated routine evaluation.

5.3 Solution of the adjoint system

The adjoint solver requires the solution of a system of $N_w = (N_c \times N_v)$ equations (3.30) but, as mentioned before, both the flux Jacobian and the right hand side in this system of equations are very sparse.

PETSc

In order to solve this large sparse discrete adjoint problem (3.30), the Portable, Extensible Toolkit for Scientific Computation (PETSc) [11, 10] was used. PETSc has been developed at the Argonne National Laboratory, US Department of Energy, and it is a suite of data structures and routines for the scalable, parallel solution of scientific applications modeled by PDEs. It employs the message passing interface (MPI) standard [115] for all interprocessor communication, it has several linear iterative solvers and preconditioners available and performs very well, provided that a careful object creation and assembly procedure is followed.

The integration of PETSc, natively implemented with MPI, with the multi-processor flow solver was achieved quite smoothly, providing an efficient, parallel (multi-processor), adjoint solver implementation.

All the adjoint and partial sensitivity matrices and vectors – $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$, $\frac{\partial I}{\partial \boldsymbol{w}}$, $\frac{\partial \mathcal{R}}{\partial \boldsymbol{x}}$ and $\frac{\partial I}{\partial \boldsymbol{x}}$ – were created as PETSc's data structures and, due to their structure, stored as sparse entities.

Once the sparse data structures are filled, the adjoint system of equations (3.30) was solved using a PETSc built-in Krylov subspace (KSP) method. More specifically, a Generalized Minimum Residual (GMRES) method was used, preconditioned with the block Jacobi method, with one block per processor, each solved with ILU(0) preconditioning. This large-scale iterative solver proved to be very efficient, exhibiting extreme robustness and fast convergence rates, even with test cases up to $\mathcal{O}(10^6)$ equations, as found in the results included in chapter 6.

5.4 Total sensitivity

Once the adjoint solution, ψ, is found, the gradient of the function of interest is easily obtained from the sensitivity equation (3.31). This expression for the adjoint-based sensitivity requires the differentiation of the flow solver residual $\mathcal{R}$ (4.37) and cost function I evaluation routines with respect to the design variables $\boldsymbol{x}$. The resulting matrix $\frac{\partial \mathcal{R}}{\partial \boldsymbol{x}}$ has dimensions $(N_c \times N_v) \times N_x$, where N_x is the number of design variables, whereas the right hand side vector $\frac{\partial I}{\partial \boldsymbol{x}}$ has length N_x.

Depending on the type of design variable being tested, the terms $\frac{\partial I}{\partial \boldsymbol{x}}$ and $\frac{\partial \mathcal{R}}{\partial \boldsymbol{x}}$ are either computed using automatically differentiated routines or approximated using finite-differences, but being partial derivatives, they are very cheap to compute since no flow re-evaluation is necessary.

When using AD tools, the re-engineered routines used for $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ and $\frac{\partial I}{\partial \boldsymbol{w}}$ earlier were again differentiated automatically, this time with respect to $\boldsymbol{x}$, thus providing $\frac{\partial \mathcal{R}}{\partial \boldsymbol{x}}$ and $\frac{\partial I}{\partial \boldsymbol{x}}$, respectively. However, this required that the routines (5.31) and (5.32), and the routine that computes the functions of interest, had to be modified so that the design variables $\boldsymbol{x}$ became input parameters,`xAdj`. For example, the residual routine should write

$$\texttt{subroutine residualAdj}(\texttt{xaDj}, \texttt{wAdj}, \texttt{rAdj}, \texttt{i}, \texttt{j}, \texttt{k})\,. \tag{5.40}$$

Once the partial derivative terms in the total sensitivity equation (3.31) had been evaluated, and the adjoint solution ψ found, the total sensitivity was then computed using the matrix-vector multiplication and the vector addition built-in operation routines provided in PETSc.

5.5 Adjoint-based optimization

The optimization problem (1.3) is solved by feeding the cost and constraint function values, obtained by the flow solver, and their gradients, obtained by the adjoint solver, into a gradient-based optimizer, following the algorithm depicted in the block diagram shown in figure 5.3. This algorithm constitutes an extension to the general design algorithm previously presented in figure 1.1.

By constructing the adjoint system of equations (3.30) and solving for the vector of adjoint variables, ψ, the sensitivity of the cost function is simply given by equation (3.31). The sensitivity obtained from (3.31) can then be used by the gradient-based optimizer to find the search direction and to determine the step size during the line search.

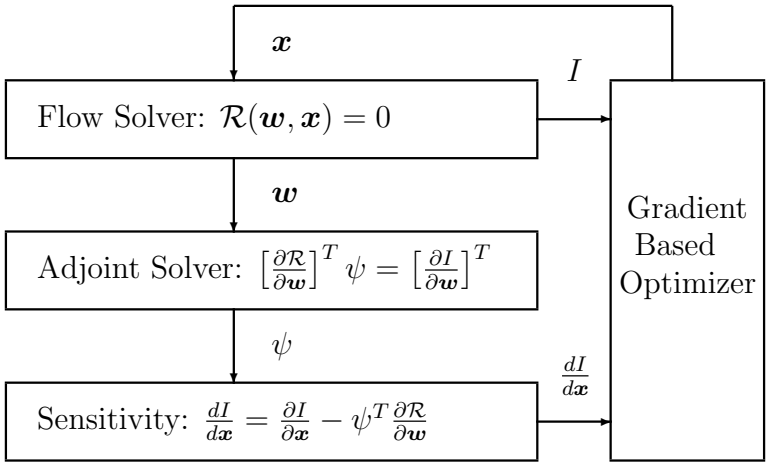


Figure 5.3: Schematic of the adjoint-based optimization algorithm.

As mentioned in chapter 1, if the design problem (1.3) includes m additional constraints to the governing equations, then it is also required to compute their sensitivities. Thus, an additional adjoint system has to be solved for each additional constraint function, C_i, which includes the computation of a new right-hand side vector for the adjoint system (3.30). Notice, however, that the adjoint matrix assembly and its factorization (if ever computed) can be preserved, expediting this process considerably, at the expense of increased memory requirements.

The gradient-based optimizer used in this work is *SNOPT* [56, 57], which is a software package for solving large-scale optimization problems (linear and nonlinear programs), developed in the Systems Optimization Laboratory at Stanford University. Among the several methods available, the Sequential Quadratic Programing (SQP) method was selected.

All the material presented in this work focuses on the adjoint solver and sensitivity modules within the optimization algorithm shown schematically in figure 5.3. The *ADjoint* approach is thought to be an answer to the rapid development of such modules.

Chapter 6

Results and discussion

The results that follow correspond to the implementation of the discrete adjoint approach in two distinct MHD flow solvers: the cell-centered, single-block, single-processor, custom-developed solver described in section 4.4.2, and the vertex-centered, multi-block, multi-processor, *NSSUS* solver described in section 4.4.4.

Four test cases have been used to demonstrate the hybrid *ADjoint* sensitivity analysis method: a blunt body, a transonic airfoil, a simplified hypersonic vehicle and a generic complete hypersonic vehicle. The first test case was used to assess and validate the discrete adjoint-based sensitivity method, whose formulation was derived by hand-differentiation of the the custom built, cell-centered, flow solver. The remaining ones were run on the sophisticated multi-block, vertex-centered, flow and adjoint solvers, that resulted from the implementation of the *ADjoint* approach.

6.1 Symmetric blunt body

In this section, a verification study of the sensitivities provided by the discrete adjoint formulation derived by hand-differentiation of a cell-centered, single block, MHD solver is presented. Both the low Re_σ and the ideal MHD equations have been studied. For this purpose, finite-difference sensitivities obtained from the flow solver are used. Additionally, a sample design case using the sensitivity information obtained with the adjoint approach is also shown.

6.1.1 Problem set-up

The configuration used as a preliminary test case follows that of Gaitonde and Poggie [46]. The body is a blunt cylinder immersed in a hypersonic incoming flow, at an arbitrary angle of attack and side-slip angle. These two angles are considered design (control) variables in the gradient and optimization computations.

A collection of hypothetical electric circuits is placed inside the body which imposes a magnetic field on the flow, calculated by using the dipole expression (4.16), automatically satisfying the condition $\nabla \cdot \mathbf{B} = 0$, as explained in section 4.3. In this way, three additional control variables are inserted in the design problem (strength m and two orientation angles), for each dipole placed inside the body. The location of the dipoles is set and kept fixed.

In addition, a collection of shape-modifying bumps is located on the body nose so that aerodynamic shape control can be performed. These bumps are given by Hicks–Henne functions [68], whose amplitudes can be changed during the design process, and are superimposed to the baseline nose radius. The bumps are equally distributed between the nose tip and the 45° angular location (with respect to the body axis) and their location is fixed. This leads to ring-type shape perturbations on the nose surface.

Using the governing equations written in non-dimensional form, only a limited set of values need to be specified to initialize the flow, namely the Mach number, M, the velocity vector direction, $\hat{\mathbf{u}} = (\hat{u}, \hat{v}, \hat{w})$, the magnetic pressure number, R_b and the magnetic Reynolds number, Re_σ.

This test case models the nose of a atmospheric re-entry vehicle, flying at a cruise altitude of $h = 30,000\,m$ ($\approx 100,000\,ft$, well within the stratosphere layer), at speed $U = 1509.05\,m/s$. Assuming a reference length of $L = 1\,m$, the thermodynamic properties of air were given using the Java script provided in the Aircraft Aerodynamics and Design Group webpage [1], that is based on the 1976 standard atmosphere data. These were found to be: temperature $T = 226.65K$, density $\rho = 0.0180kg/m^3$, pressure $p = 1171.95N/m^2$ and speed of sound (using the ideal gas assumption $c = \sqrt{\gamma RT}$) $c = 301.8m/s^2$. The corresponding free-stream Mach number was $M = U/c = 5$.

The imposed magnetic field magnitude B, which is controlled by the dipole strength m, determines R_b, whereas the medium electrical conductivity σ sets Re_σ. Whereas the electrical conductivity was fixed at $\sigma = 300S/m$, the dipoles strength varied, leading to a tested range of Q.

6.1.2 Flow solver validation

Figure 6.1 shows the Mach number distribution around the body. In this case, there are neither magnetic nor viscous effects, thus corresponding to the solution of the Euler equations, but the Mach number was raised to $M = 16$. The results obtained

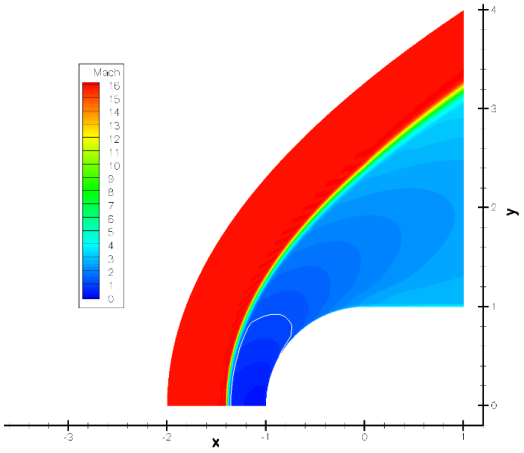


Figure 6.1: Blunt body: Euler solution at M=16.

follow remarkably well others found in the literature [129, 29].

The low magnetic Reynolds number MHD solver was validated by running a simulation of a hypersonic flow over the same blunt cylinder. In this case, an incoming flow at Mach $M = 5$ aligned with the body axis was used, and a single dipole was located inside the body at the nose center point, also aligned with the body axis. Different magnetic field strengths were tested, leading to a range of the magnetic interaction parameter of $Q = 0$ to $Q = 6$. The effect of Q on the shock stand-off distance can be seen in figure 6.2. Being a simple MHD model, and being impossible to compare the values of the non-dimensional parameter directly, one can only argue that the values follow the expected qualitative trend, that is to say, as the magnetic

field gets stronger, the stand-off distance increases, which had been observed in the cited references as well.

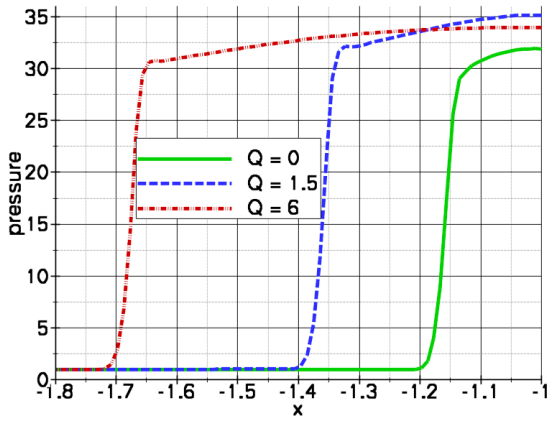


Figure 6.2: Blunt body: shock stand-off distance as function of Q.

6.1.3 Sensitivity verification of the low Re_σ MHD solver

A validation study of the sensitivities obtained from the discrete adjoint formulation was performed with finite-difference approximations computed using the MHD flow solver.

The mesh size used was a modest ($18 \times 16 \times 24$), with a single dipole located at the body nose center and oriented against a Mach $M = 5$ incoming flow at an angle of attack of $26.6°$. The imposed magnetic field was such that the value of the magnetic interaction parameter was $Q = 6$. The electrical conductivity was assumed to be a design variable in each computational cell, leading to a total of $6,912$ design variables. The results in figure 6.3 show the inviscid drag coefficient sensitivity with respect to the electrical conductivity σ on the body surface and mid plane locations.

The results obtained using the discrete adjoint approach were compared with values obtained using a finite-difference solution at three control cells located on the body surface (and shown on figure 6.3 with the dots) and the results are summarized in table 6.1. The agreement is remarkably good and highlights the potential of the

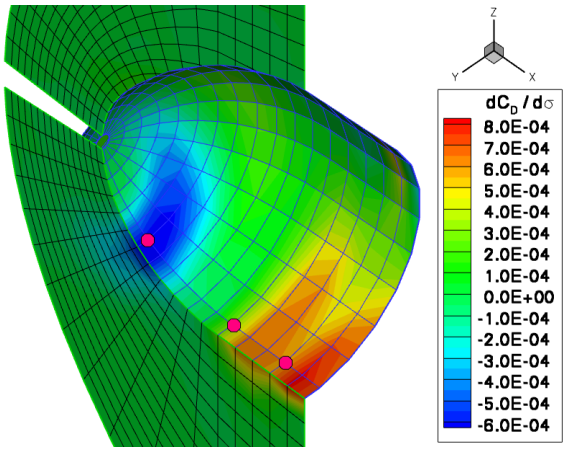


Figure 6.3: Blunt body: drag coef. sensitivity w.r.t. electrical conductivity.

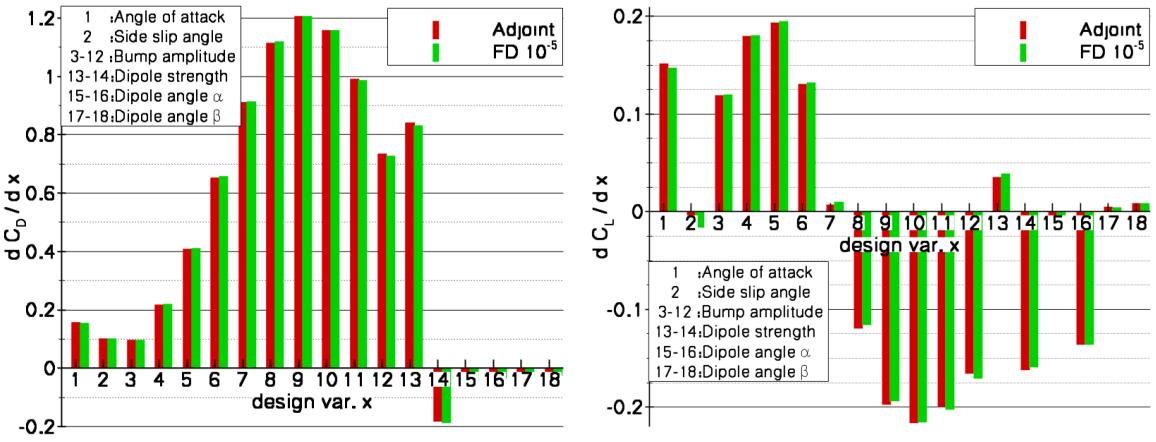
Cell index (i,j,k)	Adjoint	Finite-Differences	Δ
(8,2,19)	-6.6314×10^{-4}	-6.5840×10^{-4}	0.72%
(14,2,19)	2.9937×10^{-4}	2.9319×10^{-4}	2.11%
(17,2,19)	5.8642×10^{-4}	5.8650×10^{-4}	-0.01%

Table 6.1: Blunt body: verification of sensitivity $\partial C_D/\partial\sigma$.

adjoint procedure to be used in problems with a large number of design variables.

Figure 6.4 shows the inviscid drag and the inviscid lift coefficient sensitivities with respect to some other design variables. In these cases, the mesh size was $(32\times48\times64)$, two dipoles were located inside the body, oriented against a Mach $M = 5$ incoming flow at an angle of attack of 10° and side-slip angle of 5° . A total of 18 design variables were considered: angle of attack, side-slip angle, bump amplitudes (10 in total) and dipole strengths (2) and orientations (4). The sensitivity computed from the adjoint solution was also compared against a finite-difference approach. Once again, there is good agreement, with values matching within 1.5% for the inviscid drag coefficient sensitivity, and 3% for the inviscid lift coefficient sensitivity.

These results proved that the discrete adjoint method is a viable, accurate and efficient method to compute sensitivities in problems modeled by the MHD equations, and that the *ADjoint* approach, if implemented, would be feasible.



(a) drag coefficient

(b) lift coefficient

Figure 6.4: Blunt body: aerodynamic coefficients sensitivity for low Re_σ MHD.

6.1.4 Sample design problem using the low Re_σ MHD solver

To demonstrate the design capabilities achieved by using the sensitivity information efficiently obtained by the discrete adjoint approach, a simple design problem of the form (1.3) was solved.

The same blunt body was used, modeling the nose of a re-entry vehicle in the atmosphere. The design problem intends to maximize the inviscid drag coefficient, while keeping the inviscid lift coefficient in a specified range ($0.04 < C_L < 0.05$).

A total of 21 design variables were used, representing three types of design variables: free-stream direction (angle of attack and side slip angle), shape design variables (4 Hicks–Henne bumps distributed on the body surface) and magnetic field characteristics (strength and orientation of 5 dipoles distributed inside the body, as shown in figure 6.5). The initial imposed magnetic field was such that $R_b = 1.747$ and $Re_\sigma = 0.114$, which translated to $Q = 0.2$. In addition, a design was also made without the dipoles to investigate the importance and impact of the latter in the optimal problem solution.

Figure 6.6 shows the convergence history of the design iterations, for both the case with and without dipoles, using *SNOPT*. The initial optimization iterations employ

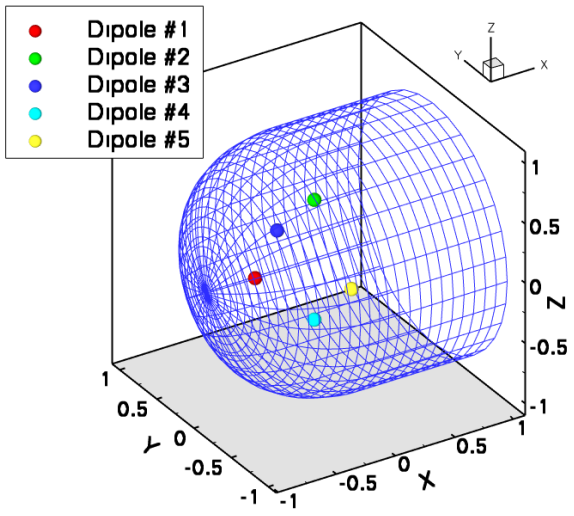


Figure 6.5: Blunt body: dipole locations.

primarily the free-stream and shape design variables, but subsequent iterations perform a fine tuning of the dipole characteristics, which result in a 2.5% improvement over the non-magnetic optimum solution. The optimal dipole strengths were limited by their upper bound of m corresponding to $B_{ref} = 0.06\,T = 600\,Gauss$, or $Q = 0.04$.

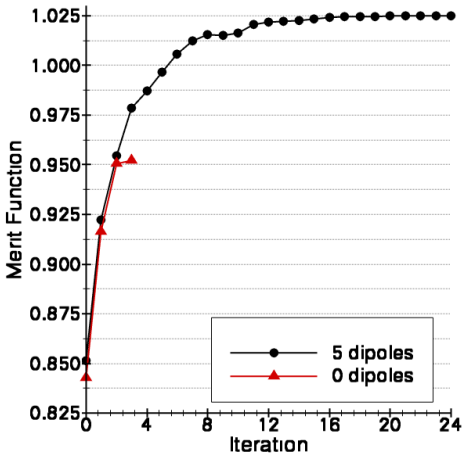


Figure 6.6: Blunt body: convergence history of the design iterations for low Re_σ.

Although the magnetic field imposed by the dipoles is weak, the pressure distribution on the surface body changes significantly compared to the non-magnetic solution. Figure 6.7 shows this distribution along the body centerline, in which the increase of pressure on both the upper and lower surfaces is the cause of the optimal solution shown in figure 6.6.

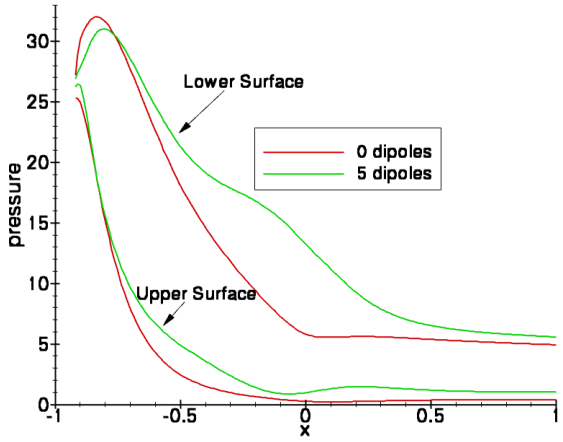
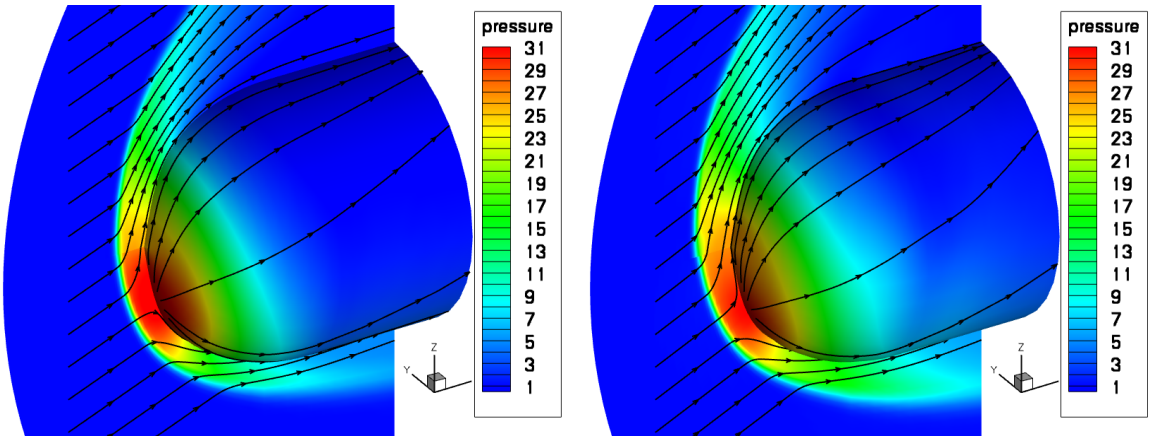


Figure 6.7: Blunt body: pressure distribution along centerline.

It is interesting also to point out that, even though the shock stand-off distance change is barely noticeable, the overall pressure distribution on the body surface changes considerably, with the pressure recovery from the stagnation point taking longer to occur, as seen in figure 6.8.

The optimal magnetic field found is shown in figure 6.9, where a contour of its strength and its vector field are illustrated on the body surface and the center plane.

These results stress the fact that in order to fine tune any MHD control device, a high-fidelity optimization method is necessary. A designer would never have got to such a precise solution if he/she had done it by trial-and-error experiments or manual tuning.



(a) without dipoles (b) with 5 dipoles

Figure 6.8: Blunt body: pressure distribution on body surface.

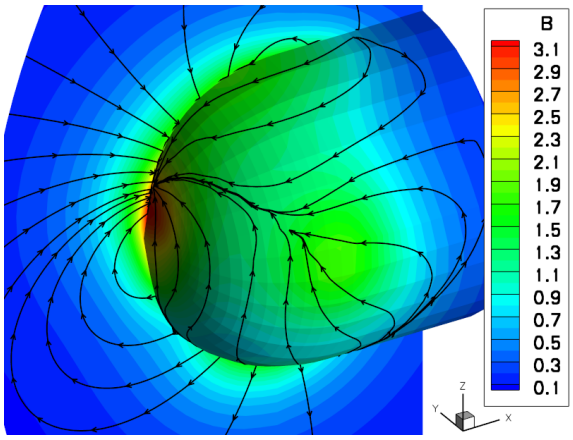


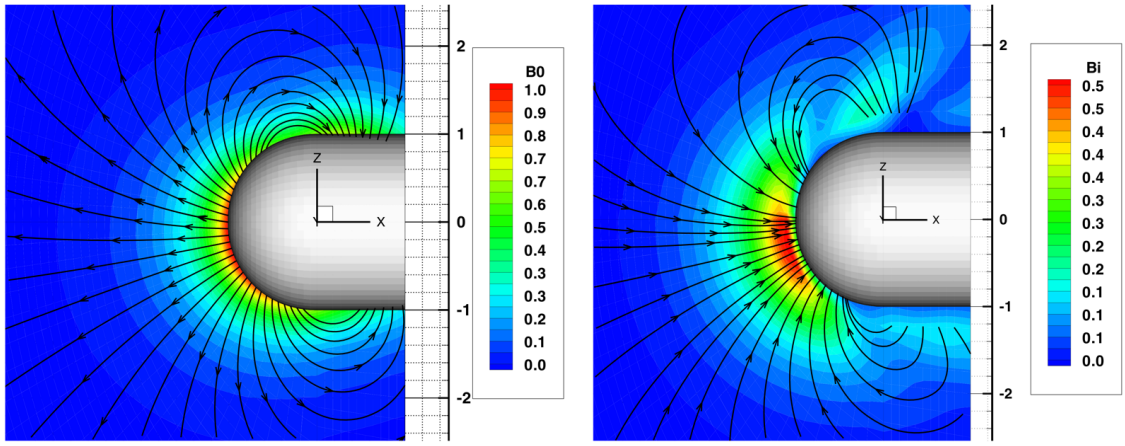
Figure 6.9: Blunt body: optimal magnetic field.

6.1.5 Sensitivity verification of the ideal MHD solver

In this section, the baseline configuration consisting of the blunt body described in the previous section is tested with a single dipole located at the body nose center, oriented against a Mach $M = 5$ incoming flow at an angle of attack of 20° and side-slip angle of 5° . The size of the mesh used was $(32 \times 32 \times 64)$ and the baseline design variable values are included in table 6.5. The flow free-stream conditions were kept identical to the previous case, including the electrical conductivity of the medium (which meant $Re_\sigma = 0.569$), but the imposed magnetic field was such that $R_b = 1.75$,

translating to a magnetic interaction parameter of $Q = 1$.

The plots in figure 6.10 show both the imposed magnetic field, due to the embedded dipole, and the induced magnetic field, as computed by the ideal MHD flow solver, on a vertical plane, where the contour map represents the magnetic field magnitude and the streamlines show the magnetic field vector. The expected strong bow shock is captured in figure 6.11, in which the Mach number is plotted, together with the static pressure distribution.



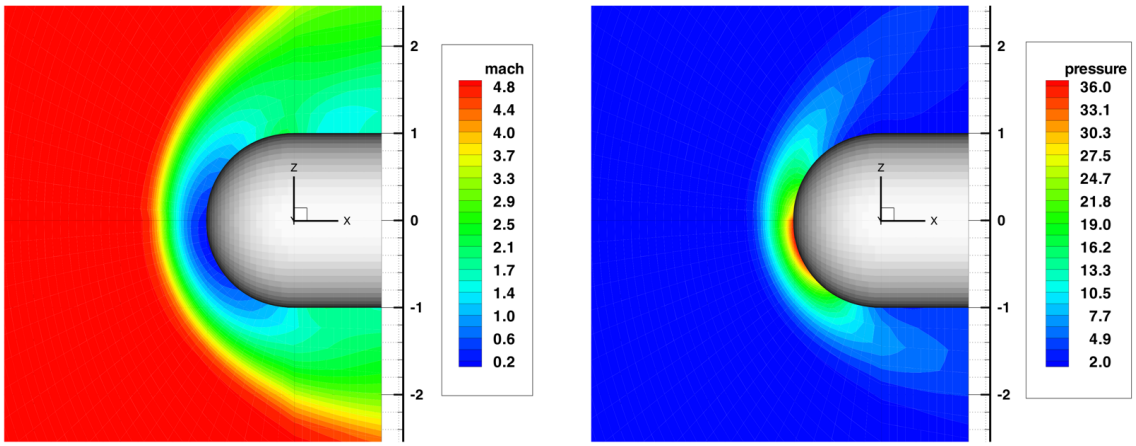
(a) Imposed field (b) Induced field

Figure 6.10: Blunt body: magnetic field of baseline configuration for ideal MHD.

Figure 6.12 shows the inviscid drag and lift coefficient sensitivities with respect to 7 design variables - angle of attack, side-slip angle, bump amplitudes (2) and dipole strength (1) and orientation (2). The sensitivities obtained using the discrete adjoint approach are matched against values obtained using a finite-difference solution, with a perturbation step of 10^{-5} for the design variables, and the results are summarized in tables 6.2 and 6.3.

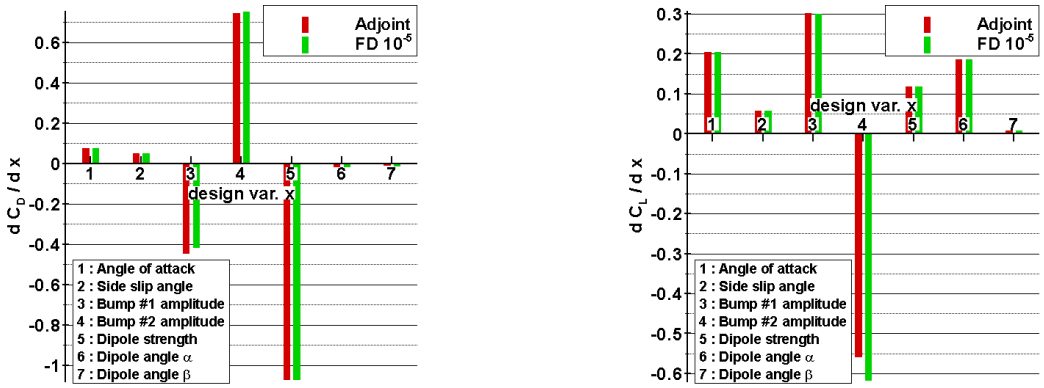
In general, there is an excellent agreement between the two approaches. The differences that sometimes occur are thought to be due to either the inaccuracy of the finite-difference solution itself or to some approximations computing the vector $\frac{dI}{d\boldsymbol{x}}$ and the matrix $\frac{\partial \mathcal{R}}{\partial \boldsymbol{x}}$ in equation (3.31).

The computational times necessary to obtain these verification results are shown in table 6.4, where the CPU time required to obtain and solve the adjoint system



(a) Mach number (b) Static pressure

Figure 6.11: Blunt body: baseline solution for ideal MHD.



(a) drag coefficient (b) lift coefficient

Figure 6.12: Blunt body: aerodynamic coefficient sensitivities for ideal MHD.

of equations is used as reference. The flow solution was obtained from a cold start (free-stream conditions in the whole computational domain), while the subsequent finite-difference solutions used the previously computed flow solution as an initial guess, having to converge only due to the design variable perturbation. The CPU time of the adjoint solution is nearly independent of the number of design variables, but linearly dependent on the number of functions of interest, whereas the CPU time of the finite-difference approach is approximately linearly dependent on the number of design variables and independent of the number of functions. These timings

Design variable	Adjoint	Finite-Diff.	Δ
Angle of attack	7.461×10^{-2}	7.432×10^{-2}	0.4%
Side-slip angle	4.871×10^{-2}	4.863×10^{-2}	0.2%
Bump #1 amplitude	-4.432×10^{-1}	-4.146×10^{-1}	6.9%
Bump #2 amplitude	7.447×10^{-1}	7.504×10^{-1}	-0.8%
Dipole strength	$-1.070 \times 10^{+0}$	$-1.071 \times 10^{+0}$	-0.1%
Dipole angle α	-1.526×10^{-2}	-1.527×10^{-2}	0.0%
Dipole angle β	-1.003×10^{-2}	-1.064×10^{-2}	-5.7%

Table 6.2: Blunt body: drag sensitivity verification for ideal MHD.

Design variable	Adjoint	Finite-Diff.	Δ
Angle of attack	2.044×10^{-1}	2.044×10^{-1}	0.0%
Side-slip angle	5.846×10^{-2}	5.839×10^{-2}	0.1%
Bump #1 amplitude	3.021×10^{-1}	3.005×10^{-1}	0.5%
Bump #2 amplitude	-5.572×10^{-1}	-6.164×10^{-1}	-9.6%
Dipole strength	1.186×10^{-1}	1.186×10^{-1}	0.1%
Dipole angle α	1.869×10^{-1}	1.869×10^{-1}	0.0%
Dipole angle β	7.790×10^{-3}	7.790×10^{-3}	0.0%

Table 6.3: Blunt body: lift sensitivity verification for ideal MHD.

clearly demonstrate the excellent efficiency of the adjoint-based sensitivity approach when compared to the traditional finite-difference approach, here with a 43:1 ratio. Obviously, the finite-difference approach handicap would have been even larger if more design variables had been used.

Solver	Time
Flow	21
Adjoint sensitivity	1
Finite-diff. sensitivities	43

Table 6.4: Blunt body: computational time with 2 functions and 7 variables.

6.1.6 Sample design problem using the ideal MHD solver

Similarly to what was done using the low Re_σ model, a design problem using the same body set-up was solved to assess the design capabilities using discrete adjoint-based MHD sensitivities. The design problem consists of controlling the inviscid drag coefficient, either minimize or maximize it, while keeping the inviscid lift coefficient within a specified range. The range and baseline values of the design variables and constraints are shown in table 6.5.

Figure 6.13 shows the convergence history of the design iterations for both the drag minimization and maximization cases. The optimizer quickly drives the design

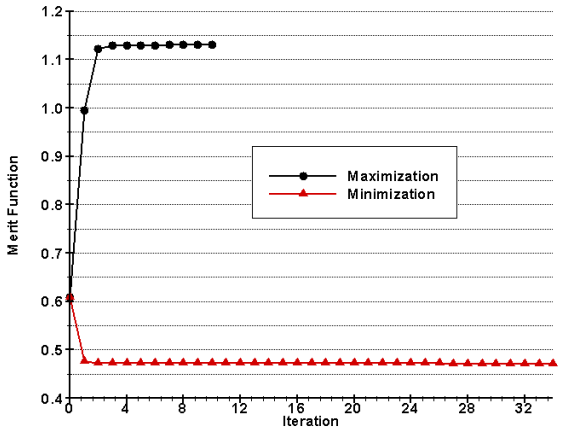


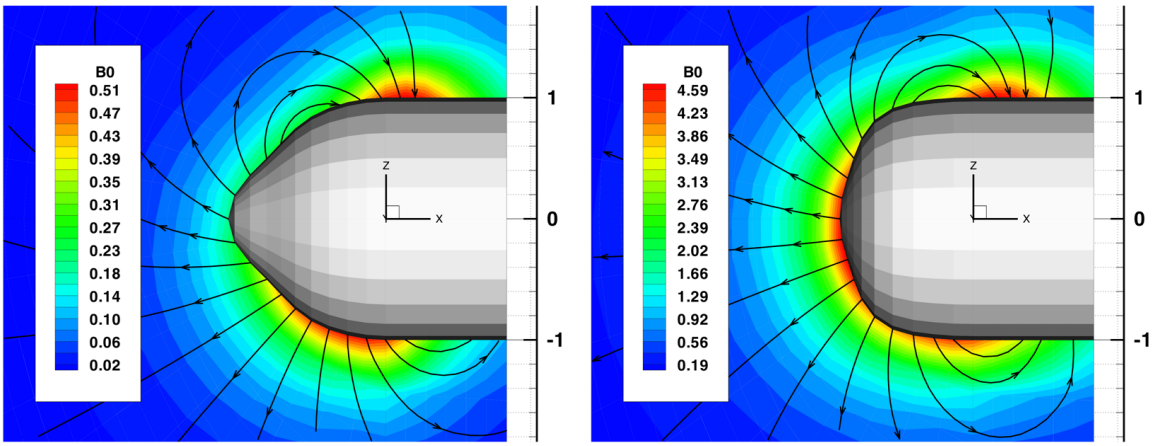
Figure 6.13: Blunt body: merit function convergence history using ideal MHD model.

solution toward the optimum and then further iterates to achieve the desired accuracy.

As expected, the minimum drag optimal solution corresponds to a slender body with a weak imposed magnetic field, while the maximum drag optimal solution corresponds to a more blunt body with a strong imposed magnetic field, as sketched in figure 6.14. The optimal results obtained for both cases are summarized in table 6.5.

The 10 and 34 optimizer iterations shown in figure 6.13 corresponded to 40 and 94 function calls, respectively. Each function call represent one cost and constraint function value and sensitivity evaluations, that is, one flow and two adjoint solutions.

In both design problems, the optimal solutions satisfy all the constraints and significant improvement is achieved over the baseline configuration. In the case of



(a) C_D minimization (b) C_D maximization

Figure 6.14: Blunt body: optimal solutions for ideal MHD.

Design variable	x_{min}	x_0	x_{max}	CD_{min}	CD_{max}
Angle of attack	-0.3491	0.1745	0.3491	0.0761	0.2349
Side-slip angle	-0.1745	0.0858	0.1745	0.0000	0.0272
Bump #1 amplitude	0.1000	0.2000	0.3000	0.3000	0.1000
Bump #2 amplitude	0.0500	0.1000	0.1500	0.0500	0.1500
Dipole strength	-0.1200	-0.0100	-0.0100	-0.0100	-0.1200
Dipole angle α	-0.6981	0.0853	0.6981	0.6981	0.2412
Dipole angle β	-0.3491	0.1745	0.3491	0.0029	0.0372
Drag Coefficient	-	0.6080	-	0.4710	1.1304
Lift Coefficient	0.0350	0.0385	0.0400	0.0350	0.0350

Table 6.5: Blunt body: design variable bounds, initial value and optimal values.

drag maximization, the optimum occurs with the dipole oriented against the incoming flow, and set at its maximum allowable strength. Also, as one would expect, the body shape got blunter compared with the baseline. In contrast, the minimum drag is achieved with a slender body shape and the weakest possible dipole strength. Notice that, due to the way the optimization problem was placed, the optimal solutions are located at the bounds of the design variables.

Once again, the adjoint-based sensitivities computed using a discrete adjoint approach proved to be accurate for design applications.

6.2 Transonic airfoil

The adjoint solver developed for the multi-block vertex-centered CFD solver was initially tested with a simple geometry, namely, using the NACA 0012 airfoil. This test case was meant to verify the *ADjoint* implementation, including the boundary conditions handling in the automatically differentiated routines.

6.2.1 Problem set-up

A quasi-3D wing was modeled using the computational mesh pictured in figure 6.15, consisting of a single-block C-mesh of size $(41 \times 33 \times 7)$. The fluid flow was assumed

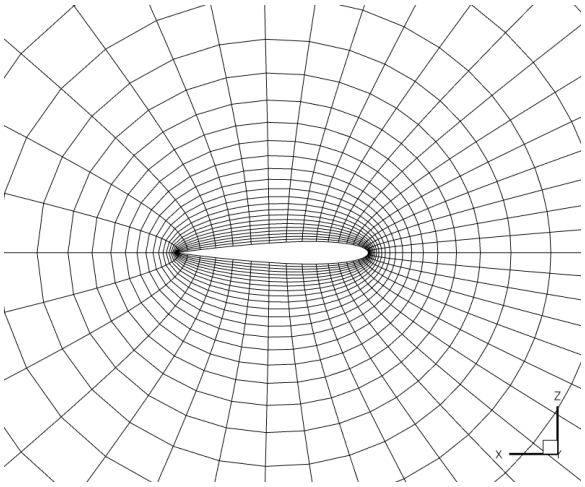


Figure 6.15: Airfoil: computational C-mesh.

to be inviscid, without any magnetic interaction, thus the Euler equations were used as the physical model.

The wing surface was set as an inviscid wall boundary, symmetry boundary conditions were applied at the wing tips in the spanwise direction (side faces of computational block), the wrapping block faces due to the C-mesh topology had internal node-to-node matching conditions applied and far-field boundary was used at the remaining block face.

The free-stream conditions were defined by a Mach number of 0.9 and an angle of attack of 5° .

6.2.2 Verification of the re-engineered residual routine

As mentioned in the automatically derived Jacobian section 5.1.2, in order to reduce the cost of the automatic differentiation procedure, a new set of subroutines that compute the residual at a specified node had to be built. This set of subroutines closely resembles the original code used to compute the residuals over the whole domain, except that it no longer loops over the domain but is based on the specification of a single-node instead.

Although this new code was essentially built by *cut-and-pasting* from the original, it was necessary to verify that the residual evaluation matched. To this end, the differences between the new and original residuals were computed in each node and the evaluated differences were all $\mathcal{O}(10^{-14})$ (i.e. same order as machine zero using double-precision arithmetic), which validated the new residual routine. For illustration purposes, the result for the continuity equation residual difference is shown in figure 6.16 at a slice of the domain corresponding to the mid-plane (computational index:$k = 4$).

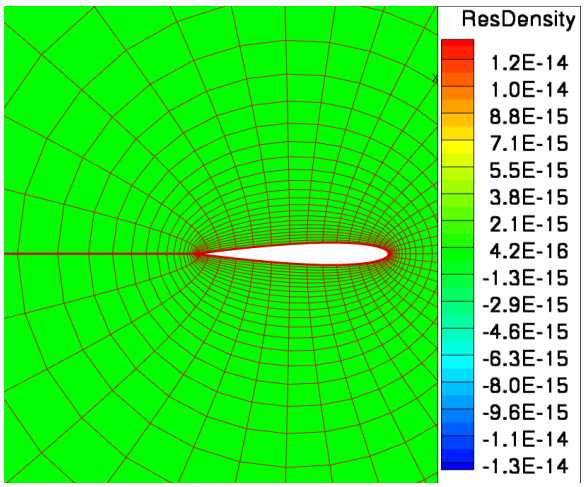


Figure 6.16: Airfoil: residual routine verification for the continuity equation.

6.2.3 Verification of the automatically differentiated residual routine

The re-engineered set of residual routines were then differentiated with Tapenade in reverse mode, for the reasons previously discussed, according to the procedure laid down in section 5.1.2. As mentioned in that section, the resulting differentiated residual code evaluates the sensitivity of one residual of the chosen node with respect to all the flow variables in the stencil of that node, i.e., all the nonzero terms in the corresponding row of $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$. To assemble the complete flux Jacobian matrix, it was necessary to loop over each residual and node indexes, in each computational directions, in every computational domain.

Similarly to what was done to the residual evaluation routines, the flux Jacobian computed with the automatically differentiated code was also verified against values estimated by finite-differencing the re-engineered residual routine and the relative error between the two was used to assess its validity. Figure 6.17 shows the relative difference of the row sum of the block Jacobian $A_{[m,n]}$ (5.5) corresponding to the energy equations,

$$\sum_n A_{[irhoe,n]} = \sum_n \frac{\partial \mathcal{R}_{ijk}(irhoe)}{\partial \boldsymbol{w}_i(n)}\,, \tag{6.1}$$

which, recalling the Tapenade differentiated Jacobian routine output (6.2), corresponds to

$$\sum_n \texttt{wAdjB}(\texttt{0},\texttt{0},\texttt{0},\texttt{n}) = \sum_n \frac{\partial \mathcal{R}(i,j,k,irhoe)}{\partial \boldsymbol{w}(i,j,k,n)}\,, \tag{6.2}$$

where `irhoe` is the index of the energy equation residual. There is a rather good agreement since all of the errors are within an acceptable range, varying between $\mathcal{O}(10^{-10})$ and $\mathcal{O}(10^{-11})$. Of course, the finite differences are most likely less precise than the automatic differentiation results, but there is no way of verifying this claim without analytic results.

Similar results were obtained for the residuals corresponding to every other governing equation (continuity and momentum equations). These results showed that the AD routine was computing the desired Jacobian entries of the adjoint matrix $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$, and that it could be safely and efficiently used to assemble it.

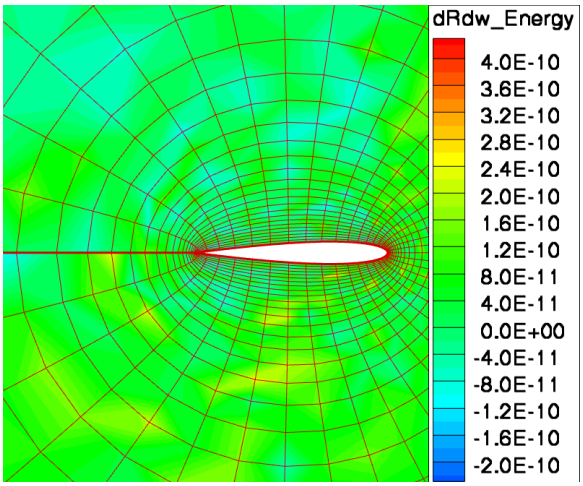


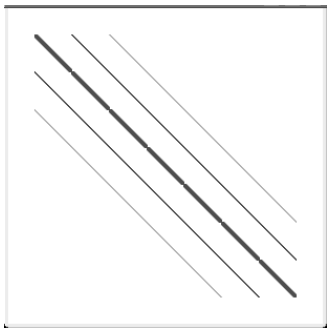
Figure 6.17: Airfoil: verification of the differentiated residual routine.

The non-zero pattern of the assembled adjoint matrix $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ and vector $\frac{\partial I}{\partial \boldsymbol{w}}$ was visualized by calling the PETSc routines

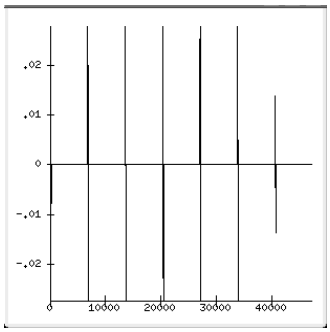
```
call MatView(dRdW,PETSC_VIEWER_DRAW_WORLD,ierr)
call VecView(dJdW,PETSC_VIEWER_DRAW_WORLD,ierr)
```

This allowed for the inspection of their sparsity pattern, which is shown in figure 6.18, where the RHS vector was evaluated considering lift coefficient as the function of interest, $I = C_L$. As expected for this single-block test case, the Jacobian is a multi-diagonal matrix because of the stencil of influence (4.8), whereas the RHS vector is only non-zero at the nodes that are on the body surface since the inviscid lift coefficient is computed from a surface integral (5.36).

The comparison between the run-time for the computation of the residuals in the whole domain using the original set of routines, the re-engineered set of routines and the time for running the corresponding version automatically differentiated in reverse mode is summarized in table 6.6, running on a single node of a 3.2GHz multi-processor workstation. These results show that the re-engineered residual routine, now node-based, takes roughly ten more times to evaluate the residuals compared to the original version, which is a result of not taking advantage of residual flux sharing between nodes that are optimized in the latter residual evaluation with the



(a) Adjoint matrix: $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$



(b) Adjoint vector: $\frac{\partial I}{\partial \boldsymbol{w}}$

Figure 6.18: Airfoil: adjoint matrix and vector sparsity patterns.

Original residual eval.	0.008 sec
Re-engineered residual eval.	0.088 sec
AD Jacobian residual	1.744 sec
Ratio	19.82×

Table 6.6: Airfoil: comparison between the original and differentiated residual evaluations.

nested loops in the three computational directions. This supports the claim that two different residual evaluation routines must be coded, in order to optimize both the flow solver and the adjoint matrix derivation using the *ADjoint* approach.

In addition, it can also be seen that the evaluation of the differentiated version of the re-engineered residual routine was equivalent to approximately 20 residual computations, which is a ratio of the order typically found codes automatically differentiated in reverse mode.

The verification performed previously (fig. 6.16) allowed for the comparison of the time needed to compute the full Jacobian matrix using the automatically differentiated residual routine and a central finite-difference approximation (3.5) approach with the original residual routine. The results are summarized in table 6.7 using different number of processors. Even though the evaluation of the differentiated residual routine takes longer than the original one, the automatic differentiation approach is much more efficient than finite differencing in computing the flux Jacobian due to the fact that the reverse mode calculation does not need to be called as many times

# proc.		1	2	3	4
Automatic differentiation	[s]	1.74	0.84	0.58	0.42
Finite-differences	[s]	21.29	10.29	7.18	5.01
Ratio		12.21×	12.19×	12.39×	11.92×

Table 6.7: Airfoil: run-time comparison between the AD and FD Jacobian evaluation.

as the finite-difference residual evaluations, as detailed in chapter 5. A speed-up of more than 12 times was registered regardless of the number of processors used, which also highlighted the good scalability of the approach.

6.2.4 Flow and adjoint solutions

The Mach number contours obtained by running the flow solver for the conditions described in the set-up section is shown in figure 6.19. As expected, a transonic regime is established as a result of the significantly high free-stream Mach number, with a pocket of significant size of supersonic flow on the upper airfoil surface.

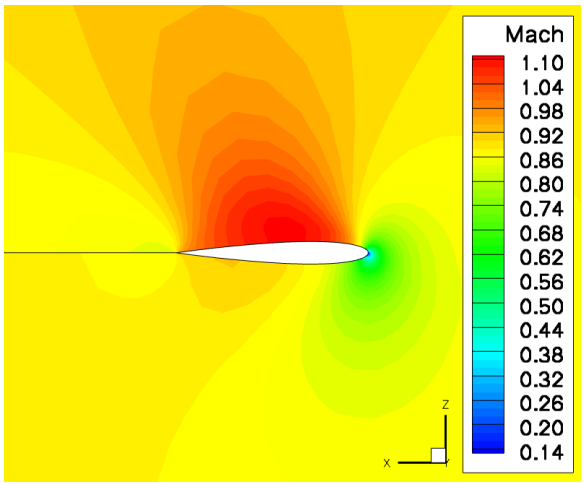
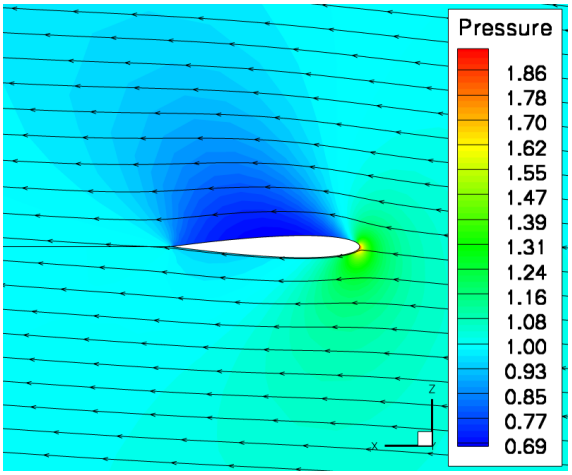


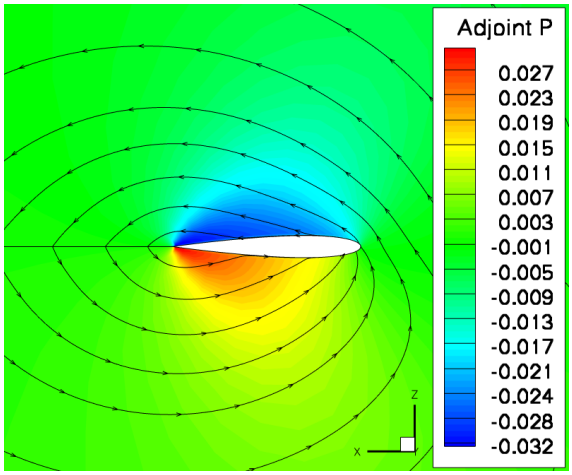
Figure 6.19: Airfoil: Mach number contour.

Once the flow solution was computed, the adjoint solver was run for different aerodynamic coefficients, taking the role of several functions of interest in the sensitivity

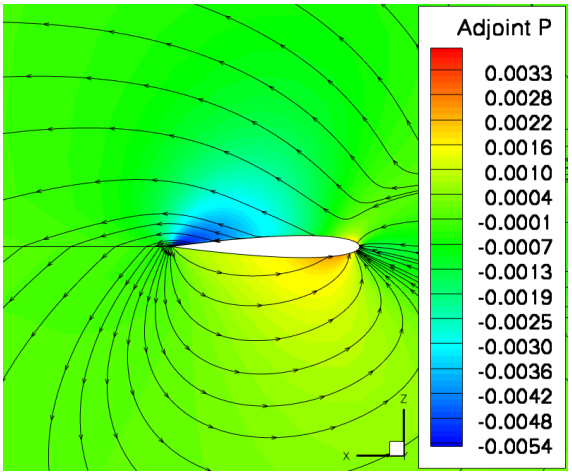
analysis, which corresponded to solving the adjoint system (3.30) for different right-hand side vectors $\frac{\partial I}{\partial w}$. The corresponding adjoint solutions are shown in figure 6.20 for the inviscid lift, drag and pitching moment coefficients, where the flow solution has been also included for reference. These plots show the contours of the flow and



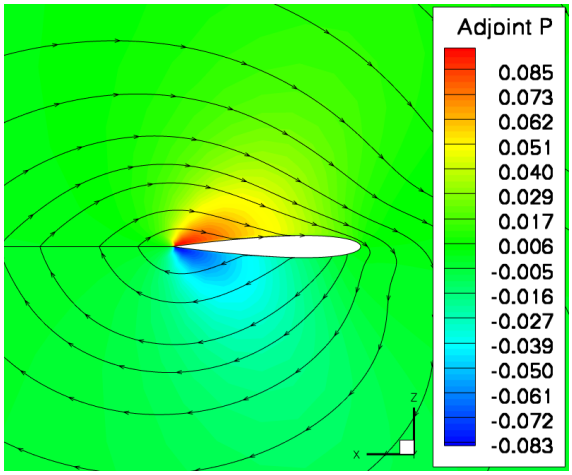
(a) Flow pressure



(b) Adjoint pressure $I = C_L$



(c) Adjoint pressure $I = C_D$



(d) Adjoint pressure $I = C_{My}$

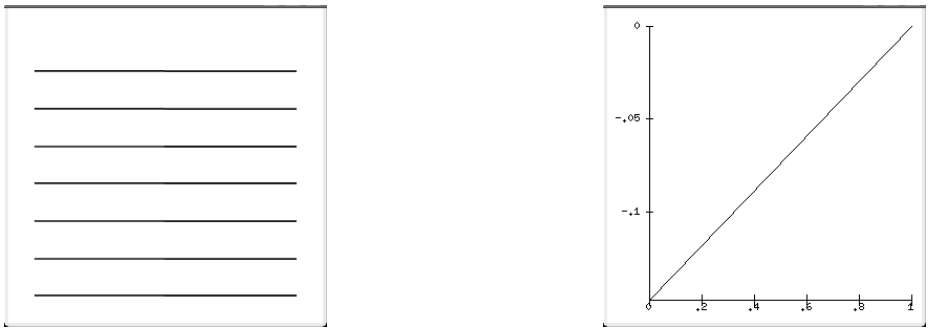
Figure 6.20: Airfoil: flow and adjoint solutions.

adjoint pressure, and the streamlines corresponding to the flow and adjoint velocity fields.

6.2.5 Verification of the adjoint-based sensitivities

After the adjoint system of equations was solved, the gradients of the functions of interest were computed using the total adjoint-based sensitivity expression (3.31) with respect to the angle of attack, α, here taking the role of a design variable.

The sparsity pattern of the auxiliary partial derivatives, obtained using the PETSc built-in visualization functions, are shown in figure 6.21, for $I = C_L$.



(a) Gradient matrix: $\frac{\partial \mathcal{R}}{\partial \alpha}$ (b) Gradient vector: $\frac{\partial I}{\partial \alpha}$

Figure 6.21: Airfoil: partial gradient matrix and vector patterns.

Figure 6.21 actually shown the results for both the angle-of-attack and side-slip, so only the first column of the matrix and the first entry of the vector should be considered. It can be seen that the direct influence of the angle-of-attack on the residual occurs only at the nodes located at the far-field boundary, through the boundary conditions.

The benchmark sensitivity results were obtained using the forward first-order finite-difference derivative approximation (3.4). In this case, the derivative of the lift coefficient is given by

$$\frac{\partial C_L}{\partial \alpha} \approx \frac{C_L(\alpha + \Delta\alpha) - C_L(\alpha)}{\Delta\alpha}\,, \tag{6.3}$$

where the relative magnitude of the perturbation step was taken to be 10^{-3}. This required the flow solver to be run one additional time for the perturbed value of angle-of-attack.

The resulting function values and gradients are summarized in table 6.8. The

reference area and moment arms were taken as $23.1\,m^2$ and $1\,m$, respectively, and the moments were taken about the airfoil leading edge. The excellent agreement between

	Value		Sensitivity		
	Baseline	Perturbed	Adjoint	Finite-diff.	Δ
C_L	0.47567771	0.47619350	5.1985	5.1974	0.021%
C_D	0.14730740	0.14739654	0.8978	0.8982	−0.049%
C_{My}	-0.79783454	-0.79870840	−8.8072	−8.8056	0.018%

Table 6.8: Airfoil: function values and gradients w.r.t. angle-of-attack.

the adjoint-based and finite-difference based sensitivities shown in table 6.8 prove that the adjoint solver has been correctly implemented using the proposed hybrid *ADjoint* approach.

6.3 Simplified hypersonic aircraft

After the successful verification of the *ADjoint* implementation presented in the previous test case, the flow and adjoint solvers were now using the low Re_σ MHD equations.

The results shown in this section include both the flow and adjoint solutions of a very simplified aircraft flying at hypersonic speeds, as well as gradients of aerodynamic coefficients with respect to different types of variables. A verification study of the sensitivities provided by the discrete adjoint formulation is also presented, where finite-difference approximations obtained from the flow solver were once again used.

The goal of this test case was to verify the multi-block implementation of the adjoint solver and the adjoint-based sensitivity analysis, based on the MHD equations.

6.3.1 Problem set-up

For this test case, a simplified hypersonic vehicle geometry was selected, as shown in figure 6.23, in which neither the scramjet propulsion system nor the vertical fins have been included. Since all the simulations were run without any side-slip angle, only half the body had to be modeled, as shown in figure 6.22, where the plane $y = 0$ was set to be the symmetry plane.

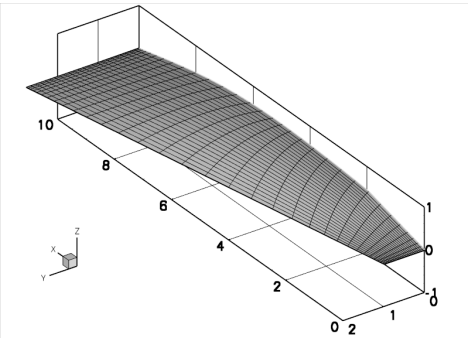


Figure 6.22: Simplified vehicle: half-body configuration modeled.

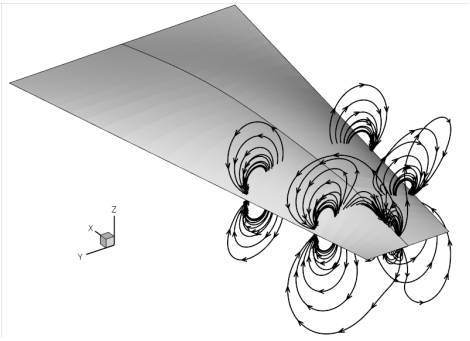


Figure 6.23: Simplified vehicle: imposed magnetic field.

A collection of five hypothetical dipoles is placed inside the body, at the locations indicated in table 6.9, which imposes a magnetic field on the flow. The resulting field given by the superposition of dipoles is shown in figure 6.23. Even though only half domain was simulated, all five dipoles were taken into account to compute the imposed magnetic field.

Dipole #	Location	Orientation
1	$(0.5, 0, 0)$	$(-1, 0, 0)$
2/3	$(1.5, \pm 0.96, 0)$	$(0, \pm 1, 0)$
4/5	$(3.5, \pm 1.24, 0)$	$(0, \pm 1, 0)$

Table 6.9: Simplified vehicle: dipole locations.

The runs were made on a parallel processor workstation, with four 3.2Ghz nodes and 8GB of RAM. The multi-block computational domain is shown in figure 6.24 and had a total of $183,342$ nodes. The inflow and outflow boundary conditions were set as supersonic and the vehicle surface as an Euler (inviscid) wall.

In this test, the inviscid aerodynamic coefficients (lift, drag, moment in x-,y-,z-directions) were used as functions of interest, I, and the electrical conductivity, σ, in each computational node was taken as the design variables. This led to a total of five cost functions and $183,342$ design variables.

The free-stream flow conditions used in this test case replicated those in case 6.1 in terms of cruise conditions, that is, a cruise altitude of $h = 30,000\, m$ and speed

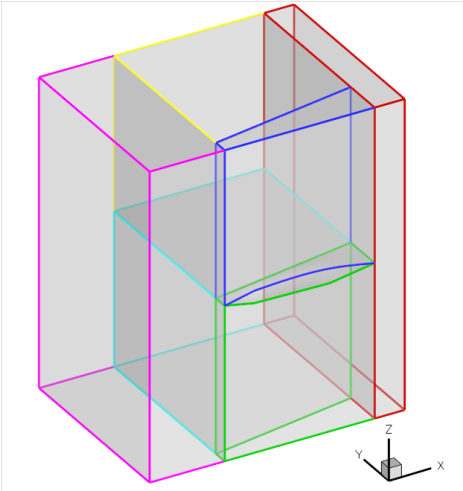


Figure 6.24: Simplified vehicle: multi-block domain.

$U = 1509.7\,m/s$ was used, corresponding to a free-stream Mach number of $M = 5$, and an angle of attack of $5°$ was modeled.

The flow solver was first run without any imposed magnetic field, corresponding to the Euler equations, and then the dipoles were turned on to obtain the MHD solution.

All the dipoles were set to the same strength m, and the baseline electrical conductivity was taken to be $\sigma = 300\,S/m$, such that a magnetic Reynolds number of $Re_\sigma = 0.57$ and a magnetic interaction parameter $Q = 0.3$ were used.

The maximum magnetic flux occurred on the body bottom surface, below dipole #1, and read $B_{max} = 8.28\,T = 8.24 \times 10^4 Gauss$. It is interesting to compare this value with some relevant values summarized in table 6.10, that were obtained from a comprehensive list of magnetic field strengths gathered by the East Tennessee State University Astronomical Observatory [125]. The magnetic intensity modeled is rela-

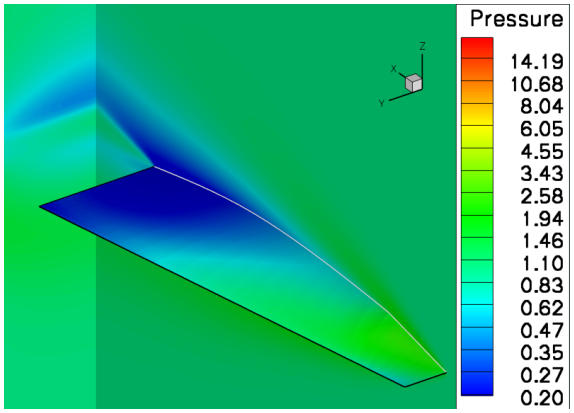
Description	Strength
Earth magnetic field	0.6 Gauss
Common iron magnet	100 Gauss
Strongest field achieved in laboratory (sustained)	4×10^5 Gauss

Table 6.10: Simplified vehicle: magnetic field strengths.

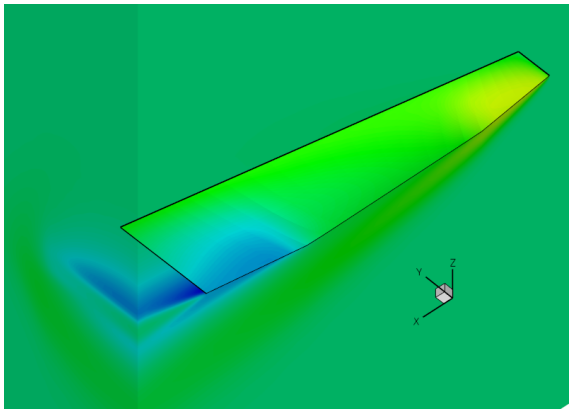
tively high, but yet possible to produce, probably at the expense of heavy electromagnets. These high values simulated were necessary because of the low ionization level considered (which translated into low electrical conductivity). However, there are studies that have shown that seeded air leads to stronger local ionization at relatively low hypersonic speeds, amplifying the magnetic effects, thus decreasing the required applied magnetic field intensity [61].

6.3.2 Flow and adjoint solutions

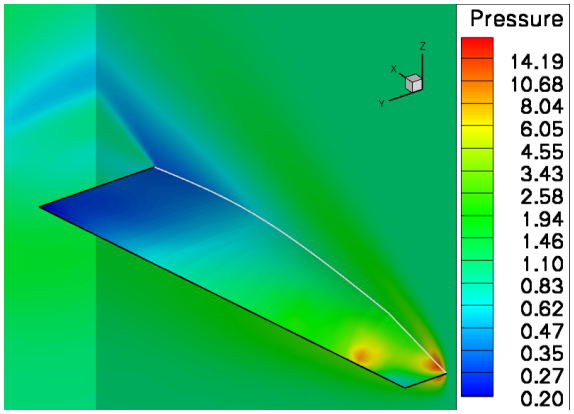
The computed pressure at the body surface and on the plane of symmetry can be seen in figure 6.25. As expected, there is large pressure increase close to the dipoles due



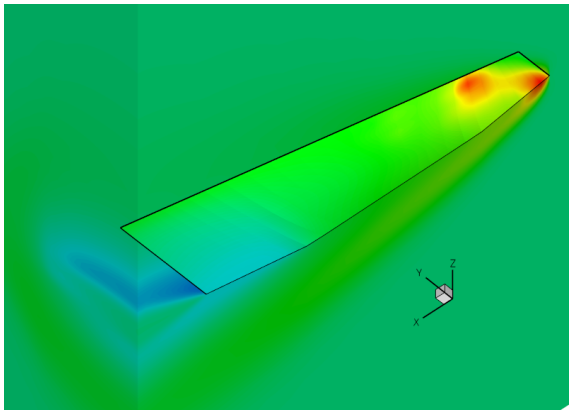
(a) Euler: top view



(b) Euler: bottom view



(c) MHD: top view



(d) MHD: bottom view

Figure 6.25: Simplified vehicle: flow solutions.

to the imposed magnetic field. This effect is caused by the additional source terms in the MHD equations and makes the numerical solution much less stable, being necessary to run at significantly lower CFL numbers. Because of this, while the Euler solution took only 768 seconds for the residual to converge ten orders of magnitude, the equivalent MHD solution took $6,245$ seconds. This clearly rules out the use of finite-differences to compute cost function gradients and highlights the importance of an alternative approach such as the discrete adjoint-based gradients. Moreover, the slower convergence highlights the need for an implicit treatment of the source terms in the MHD solution that should be pursued in the future.

The resulting values of the functions of interest are summarized in table 6.11. The reference area and moment arms were taken as $18.8\,m$ and $1\,m$, respectively.

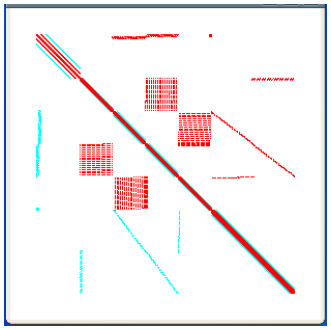
C_L	0.0520750437522571
C_D	0.0179831679282085
C_{Mx}	0.0496412105981379
C_{My}	0.0247952565444227
C_{Mz}	0.0504365242914952

Table 6.11: Simplified vehicle: baseline aerodynamic coefficients.

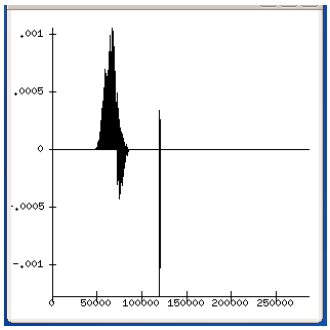
The entries of the adjoint matrix $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ and vector $\frac{\partial I}{\partial \boldsymbol{w}}$ were computed by calling the automatically differentiated routines, and then globally assembled using PETSc. The non-zero patterns were visualized using the PETSc routine calls and they are shown in figure 6.26. The vector $\frac{\partial I}{\partial \boldsymbol{w}}$ is shown for $I = C_L$ but similar results are obtained for the other functions.

Comparing figure 6.26 to figure 6.18, it is interesting to observe the effect of having a multi-block domain, causing the matrix bandwidth to increase due to block-to-block node connectivities. This bandwidth increase, however, does not negatively impact the GMRES solver since PETSc automatically takes care of the optimization of the node ordering. For that reason, no special care was given to the way the global node numbering was done in the adjoint solver pre-processing.

The assembly time of the Jacobian matrix was 10.17 seconds, including all the calls to the automatically differentiated routines and PETSc matrix functions, whereas the



(a) Adjoint matrix: $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$



(b) Adjoint vector: $\frac{\partial I}{\partial \boldsymbol{w}}$

Figure 6.26: Simplified vehicle: adjoint matrix and vector sparsity patterns.

adjoint vector assembly time was negligible.

Once the adjoint system of equations (3.30) was set up, the GMRES solver provided by PETSc was used. To be consistent with the flow solver, the adjoint solution residual convergence criterion was also set to 10^{-10}. The iterative solver consistently shows very good robustness and convergence properties. For all the different functions of interest tested, the convergence was typically achieved after about 65 iterations, which took 55 seconds to run. The residual history of the adjoint solution using PETSc for $I = C_L$ is illustrated in figure 6.27.

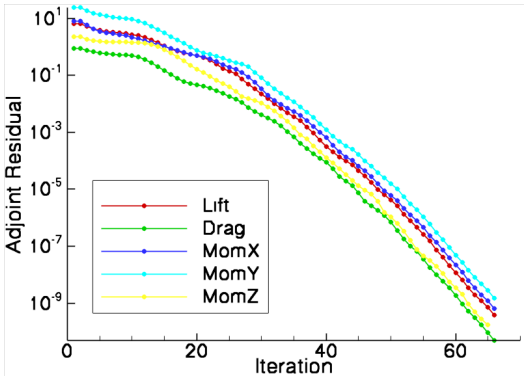


Figure 6.27: Simplified vehicle: adjoint residual convergence history.

Even though both residuals of the flow and adjoint solvers were converged ten orders of magnitude, this was only done to ensure that the solution had fully converged, and that the verification of the adjoint-based sensitivities using FD-based values was

as precise as possible. However, some tests have shown that the gradients computed using the adjoint are not significantly affected if the criteria are relaxed. Converging both solvers only five orders of magnitude still produces function values and gradients accurate enough to be used by a gradient-based optimizer. This is extremely important because a designer ultimately wants the fastest possible turnaround time, using the least computational cost and memory required.

The adjoint solution associated with the flow shown in figure 6.25 is shown in figure 6.28. It is interesting to point out that the adjoint solution, for the functions of interest used in this work, is similar to the flow solution, except that the flow direction is reversed. This occurrence has already been observed in other works involving the adjoint method [83].

In addition, using a coarser grid with only $57,214$ nodes, the Jacobian matrix entries $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$ were computed with both central finite-differences and the automatically differentiated routines. This was primarily done to debug the code but also showed the dramatic performance gain obtained using the AD routines, as shown in table 6.12. It is important to notice that these timings corresponded to a single processor

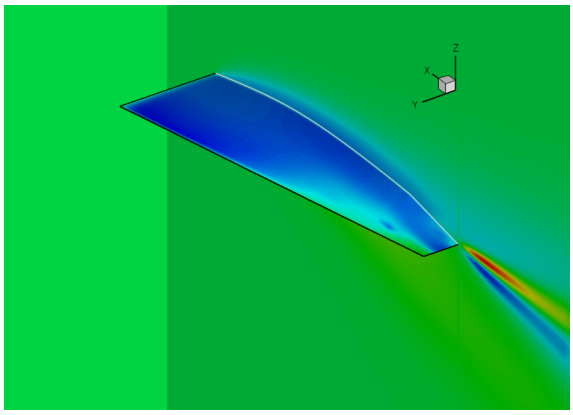
Automatic differentiation	102 sec
Finite-differences (central)	1,856 sec
Performance gain ratio	18.2×

Table 6.12: Simplified vehicle: run-time comparison between the AD and FD Jacobian evaluation.

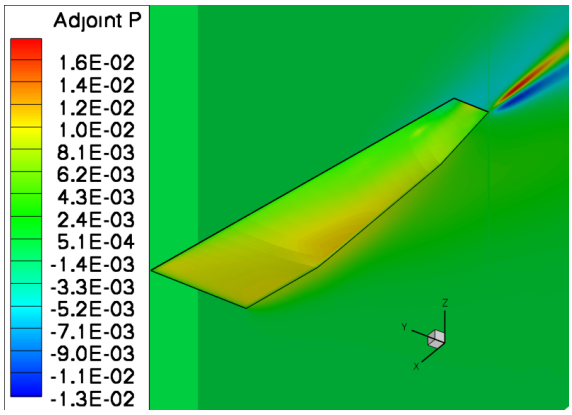
computation using a non-optimized executable, but the ratio is expected to remain similar for other conditions.

6.3.3 Adjoint-based sensitivities

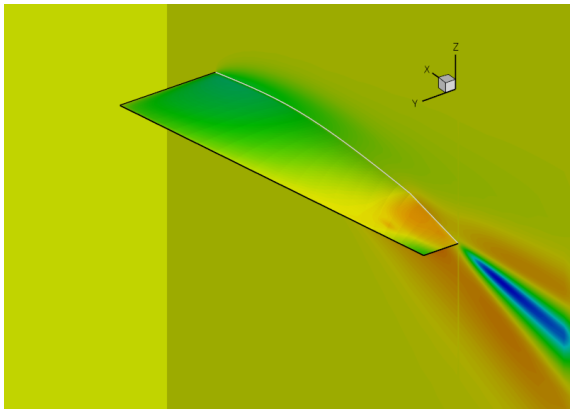
Choosing the electrical conductivity, σ, to be the independent variable in each computational node, it is necessary to evaluate two additional terms to form the total sensitivity equation, $\partial I/\partial\sigma$ and $\partial\mathcal{R}/\partial\sigma$, since the total sensitivity (3.31) is written



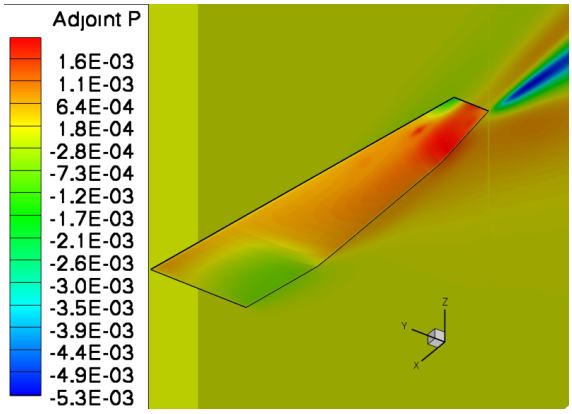
(a) Adjoint pressure $I = C_L$: top view



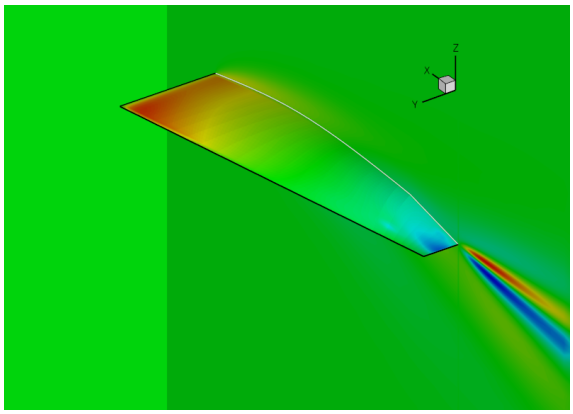
(b) Adjoint pressure $I = C_L$: bottom view



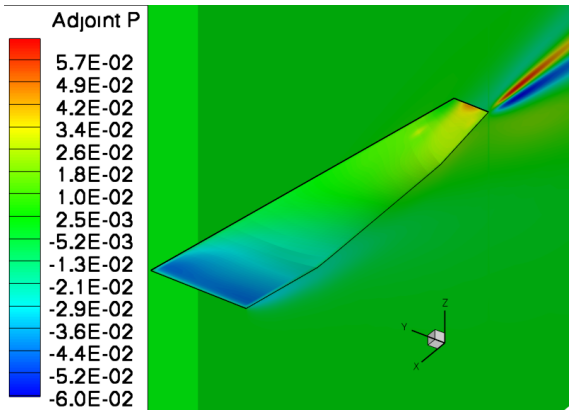
(c) Adjoint pressure $I = C_D$: top view



(d) Adjoint pressure $I = C_D$: bottom view



(e) Adjoint pressure $I = C_{My}$: top view



(f) Adjoint pressure $I = C_{My}$: bottom view

Figure 6.28: Simplified vehicle: adjoint solutions.

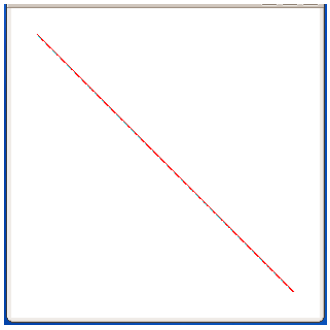
in this case as

$$\frac{\mathrm{d}I}{\mathrm{d}\sigma} = \frac{\partial I}{\partial \sigma} + \psi^T \frac{\partial \mathcal{R}}{\partial \sigma}\,. \tag{6.4}$$

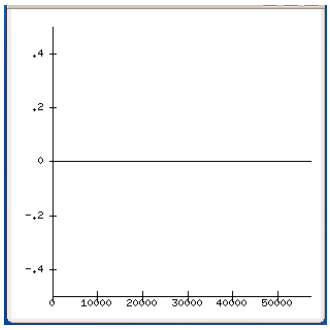
The first term is easy to evaluate since there is no direct dependence of aerodynamic coefficients with respect to the electrical conductivity σ. As such, that term is identically zero.

The second term is also easy to obtain by looking at the low Re_σ MHD governing equations, in particular to the magnetic source term: the residual $\mathcal{R}$ shows a linear dependence of σ through the source term $\mathbf{S}$ (4.14). Thus, this term was computed analytically, the result being a very sparse matrix, with non-zero entries only along the diagonal since the residual at a given node depends only on the conductivity of that same node.

These terms were assembled as PETSc obejcts and their non-zero patterns are shown in figure 6.29.



(a) Gradient matrix: $\frac{\partial \mathcal{R}}{\partial \sigma}$

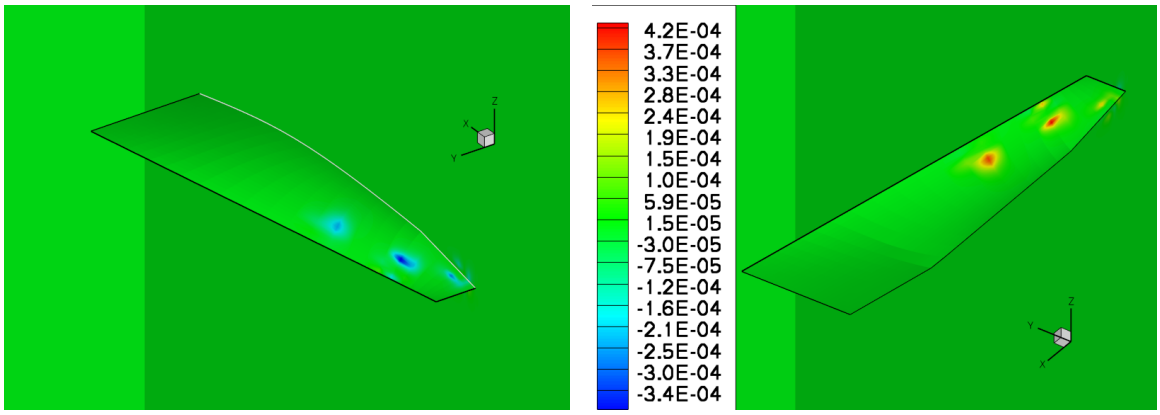


(b) Gradient vector: $\frac{\partial I}{\partial \sigma}$

Figure 6.29: Simplified vehicle: partial gradient matrix and vector pattern.

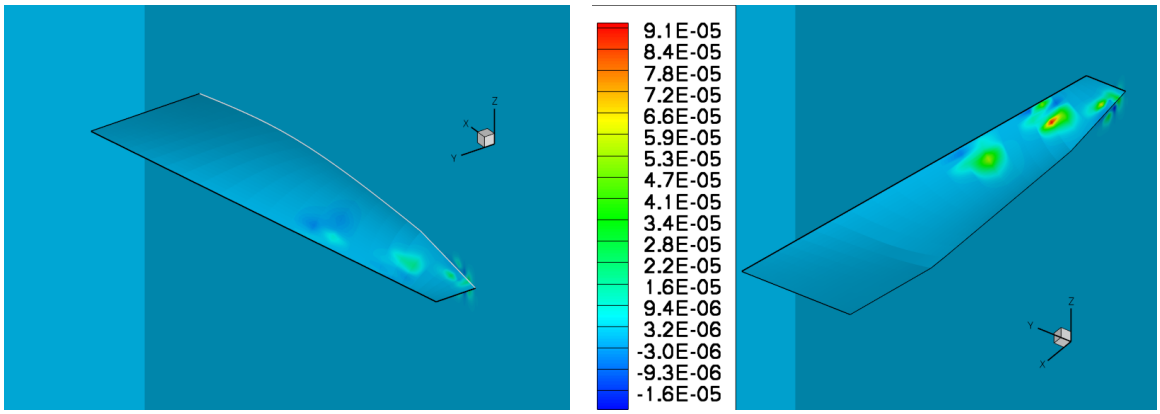
The total sensitivity (6.4) was then computed using the matrix-vector multiplication and the vector addition routines provided in PETSc.

The total sensitivity was computed for the different functions of interest and existed everywhere in the volume since the design variable σ spanned the entire problem domain. For visualization purposes, the values at the body surface and symmetry plane are presented in figures 6.30, 6.31 and 6.32, corresponding to the inviscid lift, drag and pitching moment coefficient sensitivities with respect to the electrical conductivity on these surfaces, respectively.



(a) Top view (b) Bottom view

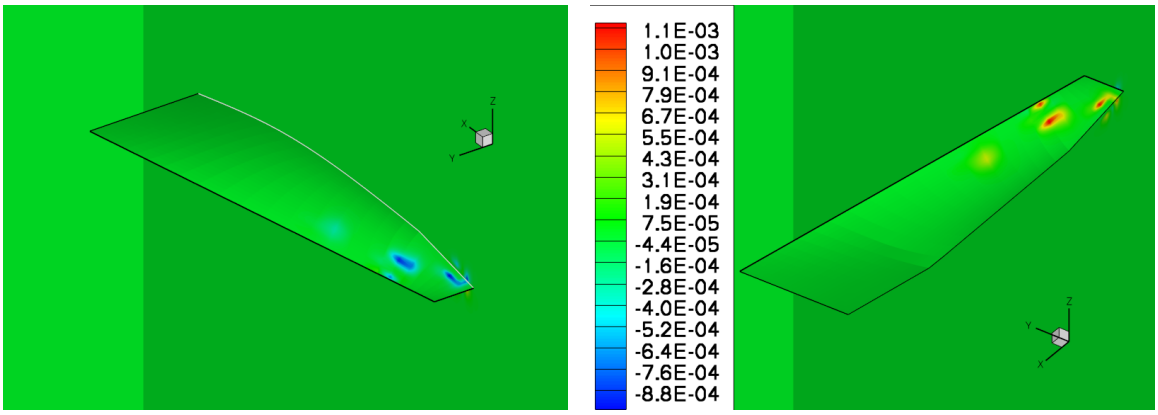
Figure 6.30: Simplified vehicle: $\mathrm{d}C_L/\mathrm{d}\sigma$ on body surface.



(a) Top view (b) Bottom view

Figure 6.31: Simplified vehicle: $\mathrm{d}C_D/\mathrm{d}\sigma$ on body surface.

Naturally, the sensitivities are higher at regions of stronger magnetic field intensity, that is, closer to the dipole locations. Also, since the magnetic field produces a local pressure rise, the sensitivity of the inviscid lift coefficient is negative on the top surface, and positive on the bottom one. Similar reasoning applies to the sensitivity of drag and pitching moment coefficients.



(a) Top view (b) Bottom view

Figure 6.32: Simplified vehicle: $\mathrm{d}C_{My}/\mathrm{d}\sigma$ on body surface.

6.3.4 Verification of the sensitivities

The sensitivities obtained using the discrete adjoint approach were matched against values obtained using the forward finite-difference solution (3.4). The comparison was made using three control nodes located on the body surface, as indicated in table 6.13, and the results are summarized in table 6.14 using two different finite-difference perturbation step sizes. The values in table 6.14 demonstrate two things. Firstly, the

Node #	Description	Zone	Location (i,j,k)
1	body bottom, under dipole 2	B	iMax @ (25,11,10)
2	body top, above dipole 1	C	kMax @ (28,6,25)
3	slightly below body nose	F	iMin @ (1,3,29)

Table 6.13: Simplified vehicle: location of control nodes for spot-checking.

agreement between the two different approaches is excellent, with an accuracy yielding between four and five digits agreement when compared to the finite-difference results, successfully verifying the adjoint-based gradient values. Secondly, it shows how the finite-difference approach can break down when choosing a perturbation step that leads to subtractive cancellation issues, as seen with control node #3. If the complex-step method [109] had been used, these issues would have been overcome.

This verification also revealed that it would have been computationally prohibitive

Control node #	Cost function I	Adjoint	Finite-diff. (step 10^{-3})	Δ	Finite-diff. (step 10^{-4})	Δ
	C_L	2.28079E-5	2.28042E-5	-0.016 %	2.28075E-5	-0.002 %
	C_D	4.90557E-6	4.90473E-6	-0.017 %	4.90548E-6	-0.002 %
1	C_{Mx}	1.03367E-5	1.03348E-5	-0.018 %	1.03365E-5	-0.002 %
	C_{My}	6.25820E-5	6.25715E-5	-0.017 %	6.25809E-5	-0.002 %
	C_{Mz}	6.61905E-6	6.61751E-6	-0.023 %	6.61889E-6	-0.002 %
	C_L	-1.49820E-5	-1.49798E-5	-0.015 %	-1.49820E-5	0.000 %
	C_D	1.22216E-6	1.22198E-6	-0.015 %	1.22215E-6	-0.001 %
2	C_{Mx}	-4.70550E-6	-4.70480E-6	-0.015 %	-4.70551E-6	0.000 %
	C_{My}	-5.37240E-5	-5.37161E-5	-0.015 %	-5.37234E-5	-0.001 %
	C_{Mz}	4.11529E-6	4.11471E-6	-0.014 %	4.11532E-6	0.001 %
	C_L	2.26830E-7	2.26822E-7	-0.003 %	2.24640E-7	-0.966 %
	C_D	5.73601E-8	5.73601E-8	0.000 %	5.71386E-8	-0.386 %
3	C_{Mx}	3.61174E-8	3.61146E-8	-0.008 %	3.41968E-8	-5.318 %
	C_{My}	1.28740E-6	1.28739E-6	-0.001 %	1.28649E-6	-0.071 %
	C_{Mz}	6.13412E-8	6.13439E-8	0.004 %	6.10878E-8	-0.413 %

Table 6.14: Simplified vehicle: verification of $\mathrm{d}I/\mathrm{d}\sigma$.

to compute the sensitivities with respect to such large number of design variables using anything but the adjoint method: using finite-difference approximations, it took roughly 5 minutes to get the flow solver to converge (starting from the baseline solution) every time the electrical conductivity was perturbed in a single node in the domain. Extrapolating to all nodes, corresponding to the $183,342$ design variables, it would have taken almost two years to obtain the same results that took less than a minute for the adjoint method described. The detailed computational costs are summarized in table 6.15. It is important to notice that the flow solver has not been optimized for MHD computations yet.

However, in realistic design problems with optimized flow and adjoint solvers, 5–10 cost functions and about 100 design variables, the automatic discrete adjoint-based gradients are expected to be at least 20–50 times faster compared with finite-difference approximations.

Solver	**Wall clock time**
Flow Solution - Euler (explicit RK5)	13 minutes
Flow Solution - Euler (implicit DDD-ADI)	2.5 minutes
Flow Solution - MHD (explicit RK5)	1 hour
Sensitivities via *ADjoint*	1 minute / function of interest
Sensitivities via finite-diff.	5 minutes / design variable

Table 6.15: Simplified vehicle: cost comparison of *ADjoint* and FD gradients.

6.4 Generic hypersonic aircraft

After assessing the multi-processor, discrete *ADjoint* solver of *NSSUS* using the previous test cases, this last test intented to show off the capabilities acquired to perform realistically sized MHD flows.

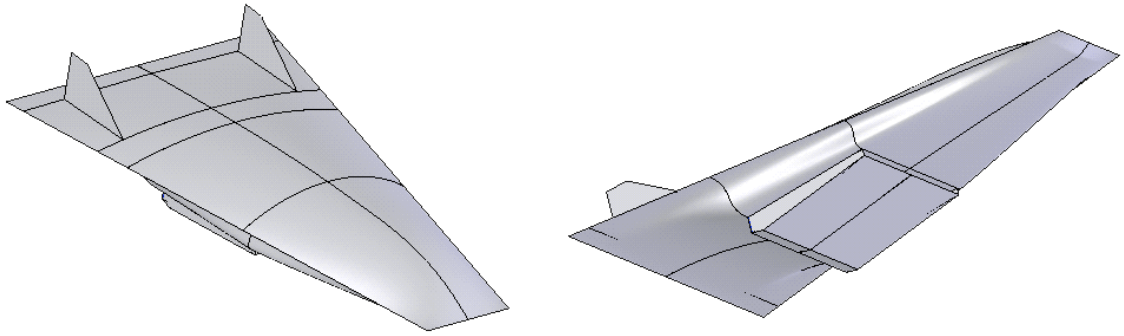
With that in mind, not only the simpler low Re_σ approximation MHD model was demonstrated, but also the more involved ideal MHD equations were tried out. In addition, the problem geometry got more detailed and complex, requiring a larger number of computational blocks and number of nodes.

6.4.1 Problem set-up

The hypersonic vehicle geometry used for this test case is a generic vehicle inspired by the NASA X-43A experimental aircraft [119], which was a scramjet-powered research aircraft capable of flying at Mach 10, and it is shown in figure 6.33. In contrast with the previous test (section 6.3), this aircraft geometry includes most of the principal airframe components, namely the vertical fins and the scramjet duct.

Since the simulation was run without any side-slip angle, only half of the body had to be modeled, with a symmetry boundary condition imposed on the center plane. The body wall was set to be an impermeable Euler wall, while the outer boundaries have non-reflecting boundary conditions imposed on them. The free-stream flow conditions chosen were Mach 5 and an angle of attack of 5° .

To simulate the magnetohydrodynamics interaction, a collection of seven hypothetical dipoles was placed inside the body, at the locations indicated in table 6.16,



(a) Top view (b) Bottom view

Figure 6.33: Generic vehicle configuration.

which imposed a magnetic field on the flow given by expression (4.16). The resulting

Dipole #	Location	Orientation
1	$(0.7, 0, 0)$	$(-1, 0, 0)$
2/3	$(2, \pm 1, 0)$	$(-0.34731, \pm 0.93775, 0)$
4/5	$(4, \pm 1.5, 0)$	$(-0.34731, \pm 0.93775, 0)$
6/7	$(6, \pm 2, 0)$	$(-0.34731, \pm 0.93775, 0)$

Table 6.16: Generic vehicle: dipole locations.

field given by the superposition of dipoles is shown in figure 6.34. Despite the fact that only half of the domain was modeled, all dipoles were taken into account when calculating the imposed magnetic field. All dipoles were set to the same strength, and the baseline electrical conductivity was such that a magnetic Reynolds number of $Re_\sigma = 0.19$ and a magnetic pressure number of $R_b = 0.11$ were used, yielding a magnetic interaction parameter of $Q = 0.02$.

The computational domain consisted of 15 blocks, as illustrated in figure 6.35.

The multi-block mesh for this test case is shown in figure 6.36. Two versions, a coarser mesh with a total of $290,107$ nodes, and a finer one with $550,109$ nodes, have been used for the ideal MHD governing equations and the low Re_σ, respectively. For visualization purposes, the smaller mesh is shown with three levels of coarsening applied.

In this example, the aerodynamic coefficients were used as functions of interest and

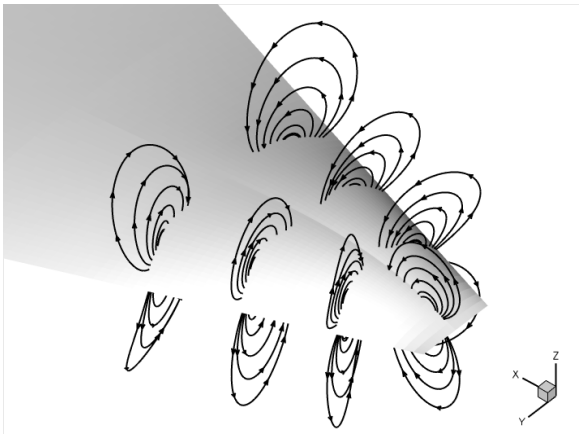


Figure 6.34: Generic vehicle: imposed magnetic field.

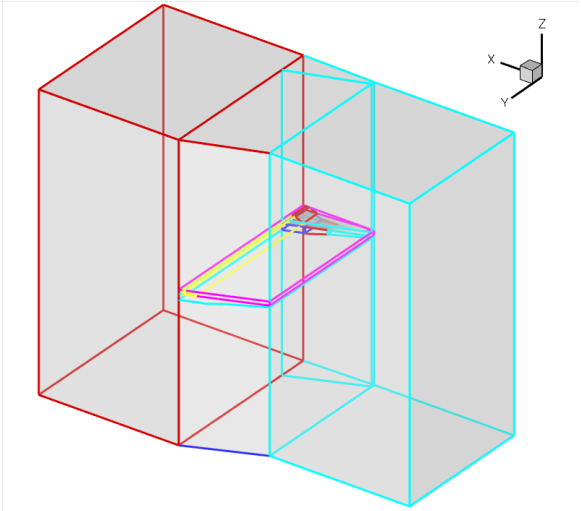


Figure 6.35: Generic vehicle: multi-block domain.

the electrical conductivity, σ, in every computational node was taken as the design variables when running the low Re_σ MHD model. As seen in the results that follow, this led to a total of $550,109$ design variables for the computational mesh used.

Additional design variables were tested as well, specifically, the vehicle attitude, defined by angle of attack, α, and side-slip angle, β, and the properties of each dipole: strength m and orientation angles α and β. These totaled 23 extra variables.

The results showed in the subsequent sections were made on a parallel processor workstation, with four 3.2GHz nodes, 2MB L2 cache and 8GB of RAM.

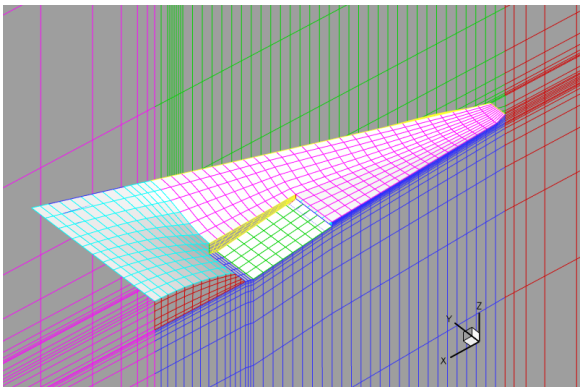


Figure 6.36: Generic vehicle: mesh, bottom view (coarsened 3 levels).

6.4.2 Flow and adjoint solutions

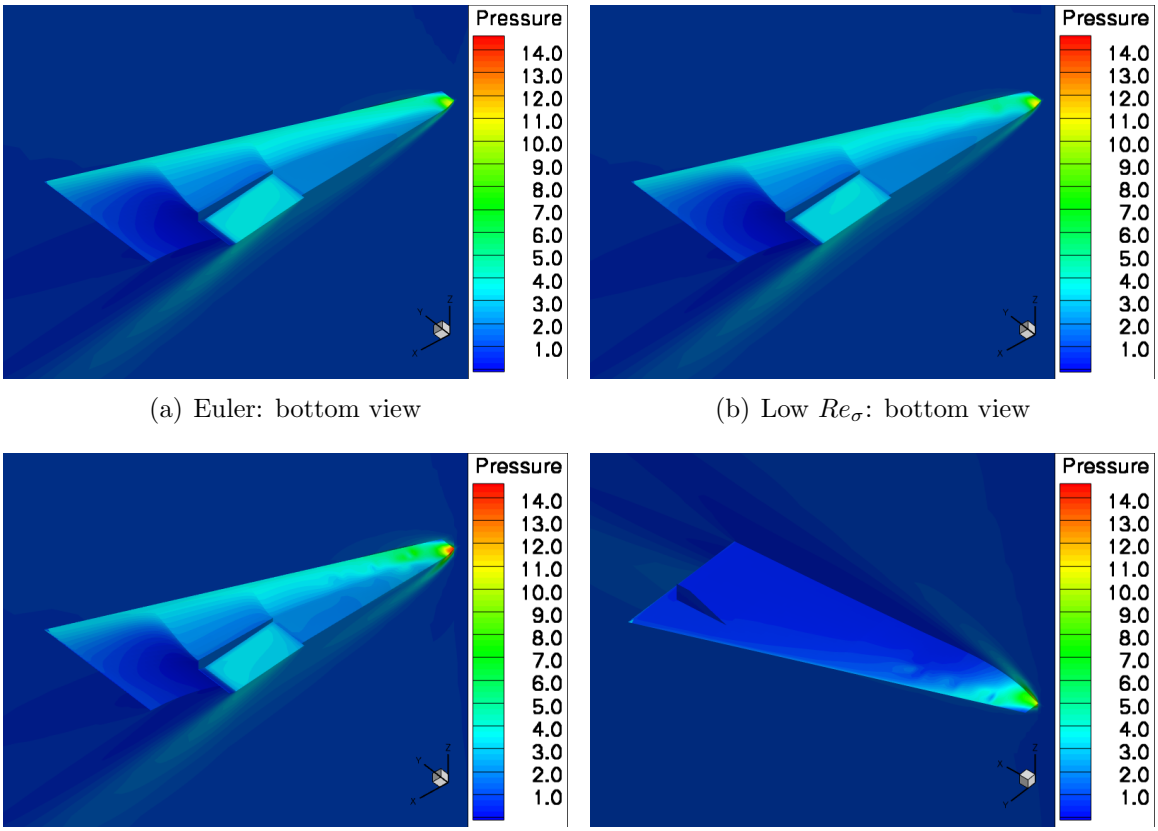
Figure 6.37 shows the pressure contours of the flow solution on the body surfaces, as well as at the plane of symmetry, for the Euler (no imposed magnetic field), the low Re_σ and the ideal MHD models. As expected, there is a large pressure increase close to the dipoles due to the imposed magnetic field. This effect is caused by the additional magnetic terms in the MHD equations.

The baseline cost function values are summarized in table 6.17, where the reference area was taken as 95.2 m^2.

Model	C_L	C_D	C_{My}
Euler	0.08309374	0.02109627	-0.10428562
Low Re_σ	0.08265143	0.02118411	-0.10308334
Ideal MHD	0.07809434	0.02112645	-0.10086555

Table 6.17: Generic vehicle: baseline aerodynamic coefficients.

The assembly time of the Jacobian matrix was 32.53 seconds for the low Re_σ solver running on the finer mesh, and 70.76 seconds for the more expensive eight-equation ideal MHD solver on the coarser mesh. These included all the calls to the automatically differentiated routines and PETSc matrix functions according to the *ADjoint* approach. The assembly times of the adjoint vectors were negligible for both models. The detailed timings can be examined in table 6.22.



(a) Euler: bottom view

(b) Low Re_σ: bottom view

(c) Ideal MHD: bottom view

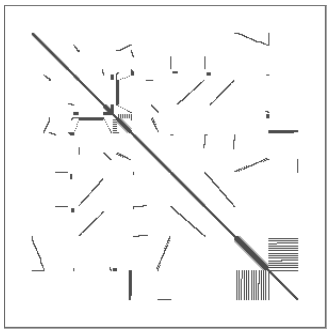
(d) Ideal MHD: top view

Figure 6.37: Generic vehicle: pressure contours.

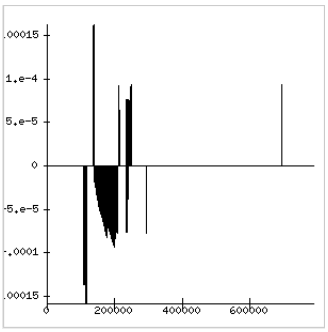
The non-zero patterns of the components of the assembled adjoint system of equations $\left[\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}\right]^T \psi = \left\{\frac{\partial I}{\partial \boldsymbol{w}}\right\}^T$ corresponding to the ideal MHD flow equations are shown in figure 6.38.

As already experienced with the simpler 6-block test case in section 6.3, the increase in the number of computational blocks causes the adjoint matrix to have a more complex structure with larger bandwidth, due to the off-diagonal blocks that result from all the block-to-block connectivities.

Once the adjoint system of equations (3.30) was set up, the GMRES solver provided by PETSc was used. To be consistent with the flow solver, the adjoint solution residual convergence criterion was also set to 10^{-10}. The iterative solver consistently showed very good robustness and convergence properties and, as in the previous test



(a) Adjoint matrix: $\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}$



(b) Adjoint vector: $\frac{\partial I}{\partial \boldsymbol{w}}$

Figure 6.38: Generic vehicle: adjoint matrix and vector sparsity pattern.

cases, the larger matrix bandwidth did not have any measurable repercussion on the efficiency of the solution of the adjoint system of equations.

For the different function of interest tested, convergence was typically achieved after about 120 iterations, which took 263.12 and 383.55 seconds to run the low magnetic Reynolds number and ideal MHD models, respectively. It is relevant to note that no restart was used in the GMRES solver, meaning all 120 Krylov subspaces were stored during the iterative procedure. The residual convergence history of the adjoint solution using PETSc for the different aerodynamic coefficients is plotted in figure 6.39 for the ideal MHD solver. This figure corroborates the claimed efficiency

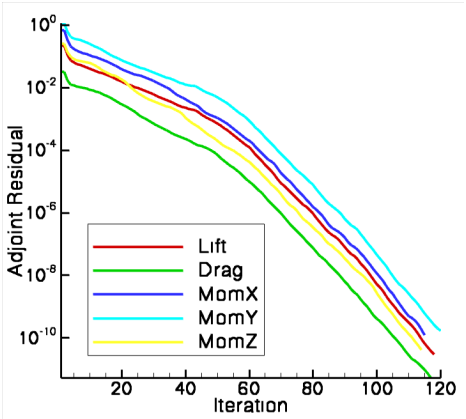
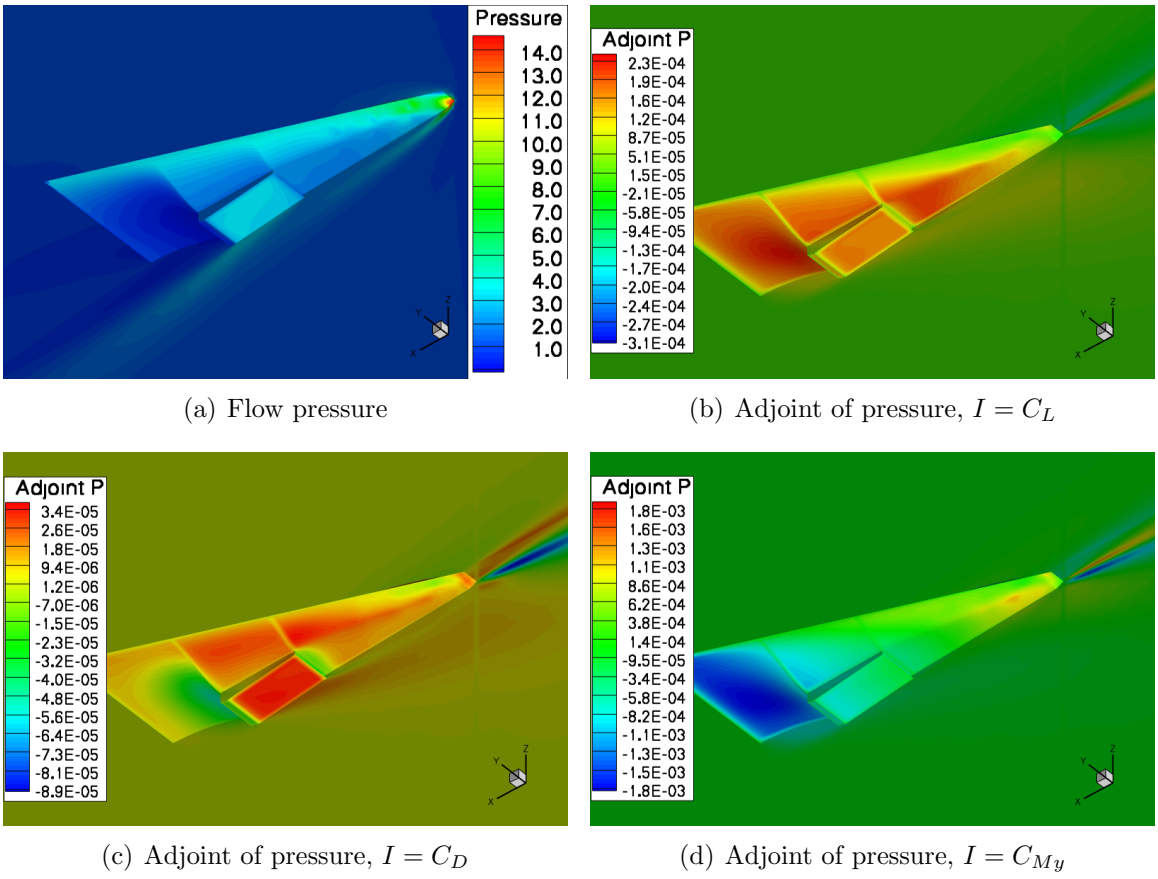


Figure 6.39: Generic vehicle: adjoint residual history (ideal MHD model).

of the preconditioned GMRES iterative solver that is provided by PETSc, displaying

convergence rates that are faster than even to best flow solvers.

The adjoint solutions corresponding to the flow pressure for different functions of interest, I, are shown in figure 6.40 for the ideal MHD solver. Similarly to other



(a) Flow pressure

(b) Adjoint of pressure, $I = C_L$

(c) Adjoint of pressure, $I = C_D$

(d) Adjoint of pressure, $I = C_{My}$

Figure 6.40: Generic vehicle: adjoint solutions (ideal MHD model).

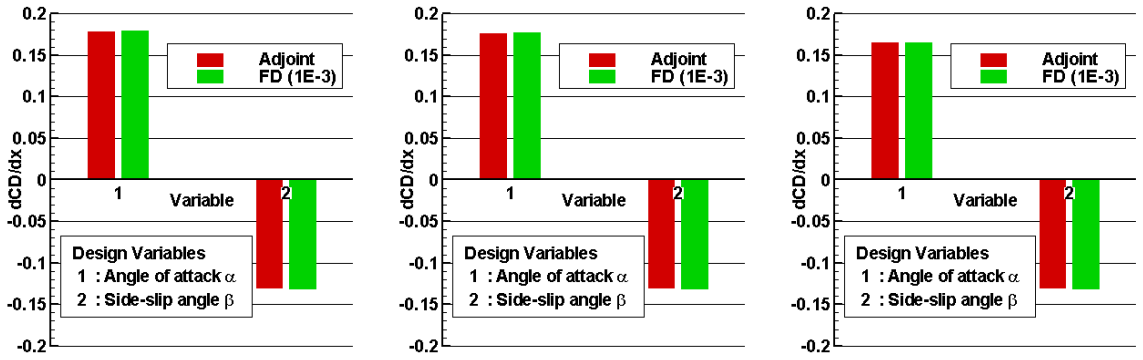
results previously presented, the adjoint solution, for the cost functions used in this work, resembles the flow solution, except that the flow direction looks as if it had been reversed. This is indeed the physical meaning of the primal and dual problem formulations outlined in section 3.5.

6.4.3 Adjoint-based sensitivities

Once the adjoint solution is obtained, the total sensitivity can then quickly be computed using equation (3.31), thus requiring the calculation of the partial derivatives

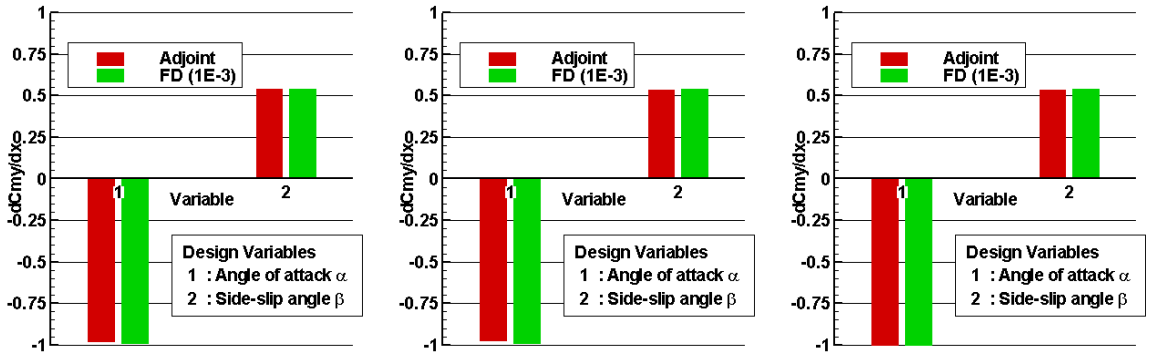
$\frac{\partial \mathcal{R}}{\partial \boldsymbol{x}}$ and $\frac{\partial I}{\partial \boldsymbol{x}}$.

The sensitivity of the inviscid drag and pitching moment coefficients with respect to vehicle attitude are shown in figures 6.41 and 6.42, respectively, for different physical models. The adjoint-based values are compared with forward-FD values using a perturbation step of 10^{-3} The agreement is always within 2.2% for the C_D and



(a) Euler model (b) Low Re_σ MHD (c) Ideal MHD

Figure 6.41: Generic vehicle: sensitivity $\mathrm{d}C_D/\mathrm{d}\boldsymbol{x}$.



(a) Euler model (b) Low Re_σ MHD (c) Ideal MHD

Figure 6.42: Generic vehicle: sensitivity $\mathrm{d}C_{My}/\mathrm{d}\boldsymbol{x}$.

C_{My} sensitivities, which is deemed acceptable for the accuracy expected from the finite-difference approximations.

Additional sensitivities of the inviscid lift coefficient with respect to the dipole properties are summarized in table 6.18, for the low Re_σ MHD model. A brief per-

Dipole #	DV $\boldsymbol{x}$	Adjoint	Finite-diff. (step 10^{-3})	Δ	Finite-diff. (step 5×10^{-3})	Δ
	str.	-3.498E-2	-3.556E-2	-1.6 %	-3.564E-02	-1.9 %
1	α	-2.337E-4	-2.343E-4	-0.2 %	-2.345E-04	-0.3 %
	β	1.354E-4	1.364E-4	-0.7 %	1.358E-04	-0.3 %
	str.	-1.065E-2	-1.121E-2	-5.2 %	-1.123E-02	-5.4 %
2	α	-2.233E-5	-2.193E-5	1.8 %	-2.195E-05	1.7 %
	β	5.589E-5	5.646E-5	-1.0 %	5.661E-05	-1.3 %
	str.	4.882E-4	4.609E-4	5.6 %	4.610E-04	5.6 %
3	α	2.427E-5	2.451E-5	-1.0 %	2.427E-05	0.0 %
	β	4.370E-6	4.437E-6	-1.5 %	4.444E-06	-1.7 %
	str.	-5.439E-3	-5.632E-3	-3.5 %	-5.643E-03	-3.7 %
4	α	-1.732E-5	-1.715E-5	1.0 %	-1.716E-05	0.9 %
	β	5.851E-6	6.178E-6	-5.6 %	6.318E-06	-8.0 %
	str.	3.444E-4	3.354E-4	2.6 %	3.364E-04	2.3 %
5	α	1.008E-5	1.058E-5	-4.9 %	1.015E-05	-0.7 %
	β	2.511E-6	2.435E-6	3.1 %	2.457E-06	2.2 %
	str.	-1.912E-2	-1.928E-2	-0.8 %	-1.931E-02	-1.0 %
6	α	-6.968E-5	-6.981E-5	-0.2 %	-6.985E-05	-0.2 %
	β	2.070E-5	2.105E-5	-1.7 %	2.129E-05	-2.8 %
	str.	7.433E-5	7.160E-5	3.7 %	7.250E-05	2.5 %
7	α	3.515E-6	4.001E-6	-13.8 %	3.610E-06	-2.7 %
	β	1.611E-6	1.556E-6	3.5 %	1.574E-06	2.3 %

Table 6.18: Generic vehicle: sensitivity of C_L w.r.t. magnetic field (low Re_σ MHD).

turbation size study was conducted for the finite-difference approximations, which showed how dependent the gradient estimates can be on their proper choice. The overall comparison lead to very satisfactory results; the small discrepancies are mainly attributed to the lack accuracy of the finite-difference approximations.

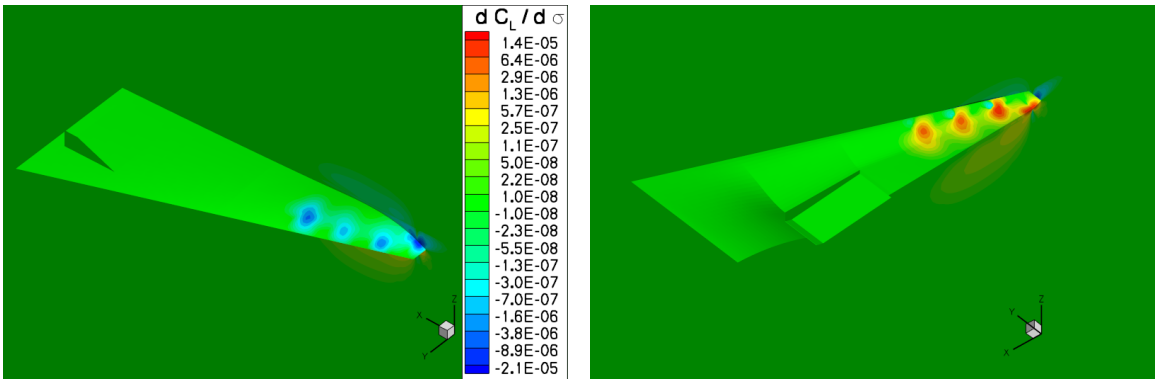
Similar results were also obtained using the ideal MHD model, and have been included in table 6.19. Once again, the matching is very good, validating the adjoint-based sensitivities computed using the *ADjoint* applied to the ideal MHD flow solver.

When running the low Re_σ solver, the electrical conductivity, σ, was also taken as a design variable in each computational node. In this case, the partial derivatives $\frac{\partial \mathcal{R}}{\partial \sigma}$ and $\frac{\partial I}{\partial \sigma}$ required to evaluate the total sensitivity $\frac{\mathrm{d}I}{\mathrm{d}\sigma}$ could be computed analytically: $\mathcal{R}$ depends linearly on σ due to its MHD source term and the inviscid aerodynamic coefficients used as cost function I do not depend explicitly on σ. The total sensitivity was then computed for the different functions of interest and existed everywhere in

Dipole #	DV $\boldsymbol{x}$	Adjoint	Finite-diff. (step 10^{-3})	Δ	Finite-diff. (step 5×10^{-3})	Δ
	str.	-2.576E-1	-2.608E-1	-1.6 %	-2.609E-01	-1.3 %
1	α	-3.322E-4	-3.246E-4	2.3 %	-3.254E-04	2.0 %
	β	1.091E-3	1.091E-3	0.0 %	1.100E-03	-0.9 %
	str.	-1.344E-1	-1.367E-1	-1.8 %	-1.369E-01	-1.9 %
2	α	-1.439E-4	-1.412E-4	1.9 %	-1.417E-04	1.5 %
	β	-9.153E-4	-9.098E-4	0.6 %	-9.051E-04	1.1 %
	str.	-3.734E-3	-3.853E-3	-3.2 %	-3.850E-03	-3.1 %
3	α	1.763E-5	1.838E-5	-4.3 %	1.794E-05	-1.8 %
	β	8.434E-6	7.048E-6	16.4 %	6.843E-06	18.9 %
	str.	-6.715E-2	-6.790E-2	-1.1 %	-6.803E-02	-1.3 %
4	α	-3.764E-4	-3.757E-4	0.2 %	-3.753E-04	0.3 %
	β	-5.535E-5	-5.578E-5	-0.8 %	-5.250E-05	5.2 %
	str.	-4.637E-3	-4.612E-3	0.5 %	-4.615E-03	0.5 %
5	α	-3.293E-5	-3.312E-5	-0.6 %	-3.277E-05	0.5 %
	β	-8.589E-6	-8.780E-6	-2.2 %	-9.028E-06	-5.1 %
	str.	-1.043E-1	-1.053E-1	-0.9 %	-1.054E-01	-1.0 %
6	α	-6.896E-4	-6.909E-4	-0.2 %	-6.906E-04	-0.1 %
	β	-2.577E-4	-2.617E-4	-1.6 %	-2.606E-04	-1.1 %
	str.	-2.349E-3	-2.334E-3	0.6 %	-2.337E-03	0.5 %
7	α	-2.277E-5	-2.314E-5	-1.6 %	-2.271E-05	0.2 %
	β	-1.660E-5	-1.652E-5	0.5 %	-1.666E-05	-0.3 %

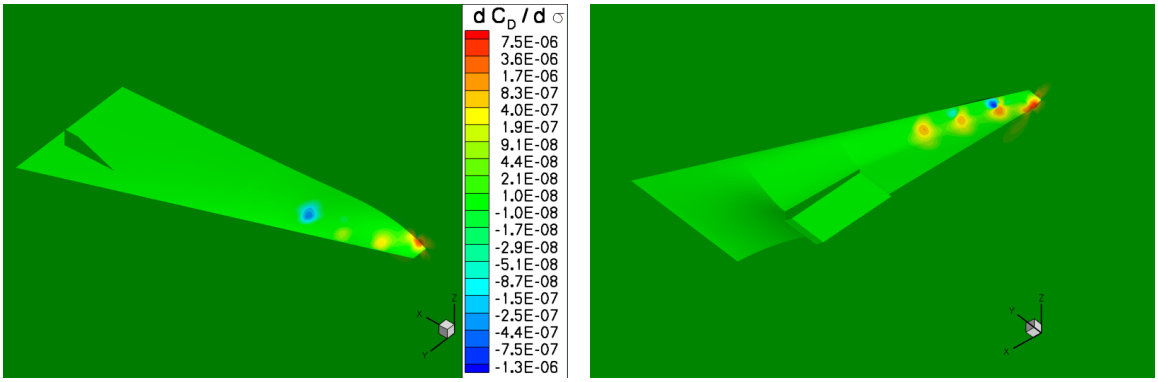
Table 6.19: Generic vehicle: sensitivity of C_L w.r.t. magnetic field (ideal MHD).

the volume since the design variable σ spanned the entire problem domain. For visualization purposes, the values are only shown at the body surface and symmetry plane. Figures 6.43, 6.44 and 6.45 show the sensitivity of lift, drag and pitching moment coefficients with respect to the electrical conductivity, respectively. As expected, these sensitivities are greatest close to the location of the dipoles, where the imposed magnetic field intensity is stronger, and their sign depends on the function of interest I. Since locally increasing the electrical conductivity σ generates stronger magnetic effects, for a given imposed magnetic field $\boldsymbol{B}$, then it also causes the local pressure to increase. Consequently, the lift sensitivity with respect to σ is positive on the bottom surface and negative on the top, the drag sensitivity is positive on the surface regions facing the incoming flow and negative on the other ones, and the pitching moment sensitivity is positive on the bottom surface behind the reference moment point and negative otherwise.



(a) top view (b) bottom view

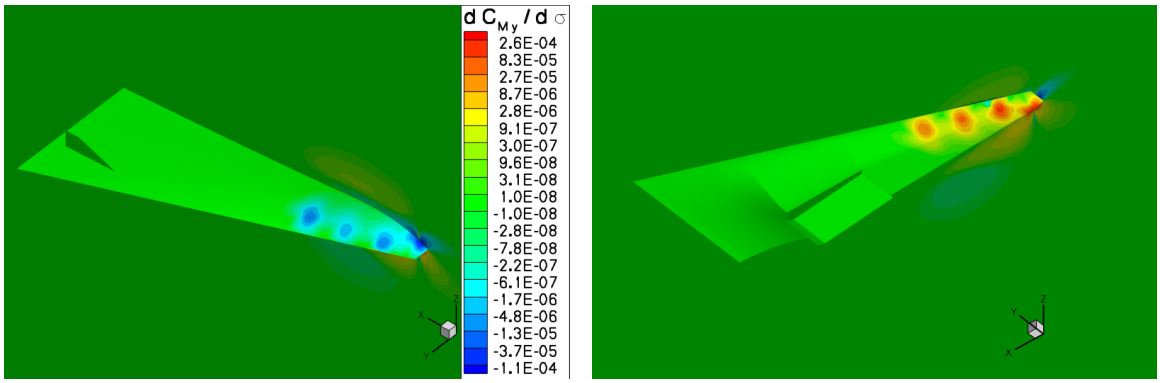
Figure 6.43: Generic vehicle: $\mathrm{d}C_L/\mathrm{d}\sigma$ (low Re_σ model).



(a) top view (b) bottom view

Figure 6.44: Generic vehicle: $\mathrm{d}C_D/\mathrm{d}\sigma$ (low Re_σ model).

These insights can be extremely useful if local flow seeding is considered in conjunction with the imposed magnetic field. Flow seeding consists in the addition of substances to the flow that have a considerably lower ionization temperature than air. As such, stronger ionization levels (translating into higher local electrical conductivity) can be made possible, maximizing the magnetic effects for the same imposed magnetic field intensity. These sensitivities give the designers the tools to find where, when and how much seeding should be injected to accomplish the desired flow control.



(a) top view (b) bottom view

Figure 6.45: Generic vehicle: $\mathrm{d}C_{My}/\mathrm{d}\sigma$ (low Re_σ model).

6.4.4 Verification of the sensitivities

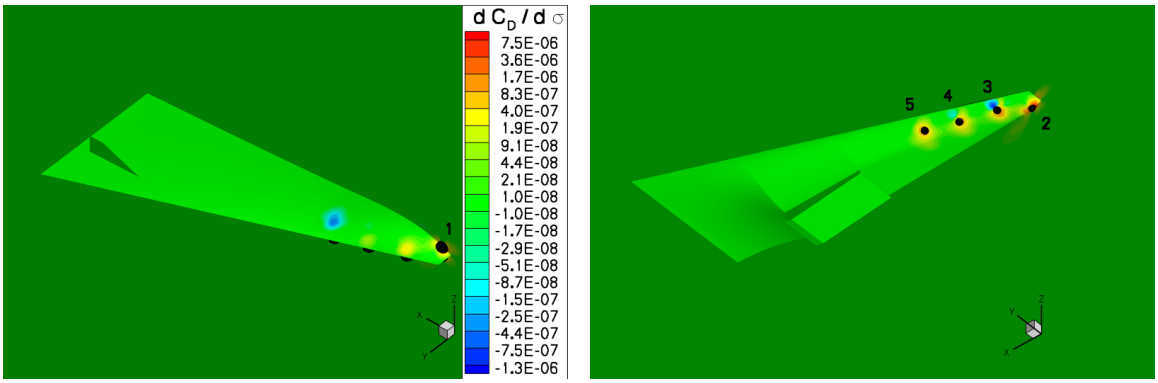
Besides the comparisons performed for the gradients of the functions of interest with respect to the vehicle attitude and dipole properties included in tables 6.18 and 6.19, the sensitivities relative to the electrical conductivity were also verified.

Because the adjoint-based sensitivities of the aerodynamic coefficients with respect to the electrical conductivity showed in figures 6.43, 6.44, and 6.45 covered the whole computational domain, the results were just spot-checked against finite-differences, which also included a FD step size study.

The comparison was made using five control nodes located on the body surface over the magnetic dipoles location, as graphically shown in figure 6.44, whose coordinates are stated in table 6.20.

Control Node	x	y	z
1	0.5457	0.0484	0.2726
2	0.5481	0.0486	-0.2535
3	1.9845	1.1352	-0.3072
4	3.9908	1.7485	-0.4047
5	5.7942	2.3447	-0.4059

Table 6.20: Generic vehicle: location of control nodes for spot-checking.



(a) top view (b) bottom view

Figure 6.46: Generic vehicle: spot-check of $\mathrm{d}C_D/\mathrm{d}\sigma$ (low Re_σ model)

The comparison results are summarized in table 6.21 using two different finite-difference perturbation step sizes: 0.1% and 0.5% of the baseline electrical conductivity, σ. The values in table 6.21 demonstrate two things: firstly, the agreement

Control node #	Cost function I	Adjoint	Finite-diff. (step 1×10^{-3})	Δ	Finite-diff. (step 5×10^{-3})	Δ
1	C_L	-2.0799E-5	-2.0744E-5	0.3 %	-2.0835E-05	-0.2 %
	C_D	3.0044E-6	3.0118E-6	-0.2 %	3.0007E-06	0.1 %
	C_{My}	-1.0495E-4	-1.0526E-4	-0.3 %	-1.0462E-04	0.3 %
2	C_L	1.4367E-5	1.3365E-5	7.0 %	1.3960E-05	2.8 %
	C_D	7.5052E-6	7.4545E-6	0.7 %	7.4295E-06	1.0 %
	C_{My}	2.5843E-4	2.6616E-4	-3.0 %	2.6109E-04	-1.0 %
3	C_L	1.3471E-5	1.2759E-5	5.3 %	1.3223E-05	1.8 %
	C_D	1.3559E-6	1.3117E-6	3.3 %	1.2954E-06	4.5 %
	C_{My}	1.5455E-4	1.6023E-4	-3.7 %	1.5620E-04	-1.1 %
4	C_L	4.1276E-6	3.6142E-6	12.4 %	4.0171E-06	2.7 %
	C_D	6.6961E-7	6.7518E-7	-0.8 %	6.5858E-07	1.6 %
	C_{My}	3.5045E-5	3.9367E-5	-12.3 %	3.5971E-05	-2.6 %
5	C_L	4.0768E-6	3.8087E-6	6.6 %	4.0112E-06	1.6 %
	C_D	8.9999E-7	9.0168E-7	-0.2 %	8.9586E-07	0.5 %
	C_{My}	2.5903E-5	2.8220E-5	-8.9 %	2.6500E-05	-2.3 %

Table 6.21: Generic vehicle: verification of $\mathrm{d}I/\mathrm{d}\sigma$.

between the two different approaches is excellent, successfully verifying the adjoint-based gradient values; secondly, it shows how the finite-difference approach is sensitive to the chosen a perturbation step. This verification also revealed that it would have

been computationally prohibitive to compute the sensitivities with respect to such large numbers of design variables using anything but the adjoint method: to get the flow solver to converge (starting from the baseline solution) every time the electrical conductivity was perturbed in a single node in the domain, took roughly one and half hours. Extrapolating to all nodes, corresponding to 550, 109 design variables, it would have taken almost 95 years to obtain the same results that took less than six minutes (per function of interest) for the *ADjoint* method.

6.4.5 Run-time and memory requirements

A performance analysis was conducted for this multi-block *ADjoint* implementation and the detailed computational costs for the different MHD models are summarized in table 6.22. It is important to notice that the flow solver has not been optimized for MHD computations yet and all solutions started from a free-stream condition throughout the domain.

	Wall clock time[1] [s]		
	Euler ($550k$)	Low MHD ($550k$)	Ideal MHD ($290k$)
Flow solver[2]	14,677	15,353	20,614
ADjoint solver	293.72	322.90	476.15
Breakdown:			
Setup PETSc variables	1.14	1.48	0.38
Assemble matrix $\frac{d\mathcal{R}}{dw}$	32.25	32.53	70.76
Assemble vector $\frac{dI}{dw}$	0.01	0.01	0.01
Solve ADjoint system	258.89	263.12	383.55
Compute sensitivity	1.43	25.76	21.45
$\frac{\text{ADjoint system}}{\#\text{GridNodes}\times\#\text{FlowVars}^2}$	0.0188	0.0191	0.0207

Table 6.22: Generic vehicle: *ADjoint* computational cost breakdown.

The additional magnetic terms in the MHD equations make the numerical solution much less stable, and because an explicit, 5-stage, Runge–Kutta time integration scheme was used, the runs had to be made at significantly lower CFL numbers (0.1). Consequently, it took almost six hours for the ideal MHD flow solver residual to

converge ten orders of magnitude. This clearly rules out the use of finite-differences to compute cost function gradients and highlights the importance of an alternative approach such as discrete adjoint-based gradients. Moreover, the slow convergence highlights the need for an implicit treatment of the source terms in the MHD solution that should be pursued in the future.

The solution of the adjoint equations was the component that took most of the time in the adjoint solver, whereas the automatic differentiation sections represented less than 15% of the time, proving its efficiency. The total cost of the adjoint solver, including the computation of all the partial derivatives and the solution of the adjoint system, is less than 3% of the cost of the flow solution for this case. Again, this is not truly representative of reality as the flow solver can still be optimized for MHD, but it clearly shows once again that the *ADjoint* approach is very efficient.

Looking at the bottom line of table 6.22, it can be inferred that the *ADjoint* equation solver runtime is proportional to the number of grid nodes N_c and the number of flow variables N_v squared, as expected from the full adjoint matrix Jacobian structure (refer to figure 5.1 and expressions 5.2–5.5).

The memory usage of the flow and adjoint solvers while running this multi-processor test case using both MHD models (low Re_σ and ideal MHD) was assessed by monitoring the memory used by each processor, and the information is summarized in table 6.23. These measurements show that the memory required for the

	Virtual memory [MB]		
	Euler ($550k$)	Low MHD ($550k$)	Ideal MHD ($290k$)
Flow solver	682	697	602
ADjoint solver	6,568	7,349	9,782
Ratio	9.6×	10.5×	16.3×
$\frac{\text{Flow memory (B)}}{\#\text{GridNodes}\times\#\text{FlowVars}}$	248	253	259
$\frac{\text{ADjoint memory (B)}}{\#\text{GridNodes}\times\#\text{FlowVars}^2}$	478	534	527

Table 6.23: Generic vehicle: memory usage comparison (in MB).

ADjoint code is approximately ten times that required for the original flow solver, when solving only for five governing equations (low Re_σ model), and increases to a sixteen fold for the eight equation model (ideal MHD). The ratios at the bottom lines of table 6.23 show that the memory usage of flow solver is proportional to the number of grid nodes N_c and flow variables N_v, whereas the adjoint solver depends on the number of grid nodes and the number of flow variables squared. This is in line with the explicit flow solver treatment, and the full discrete adjoint matrix handling used in the adjoint solver.

The main point to highlight is that current flow computations run by designers use no more than 1/10 of the memory available because of the desired faster turnaround time.

However, if larger problems ought to be run, there are other possible options to accommodate them, namely, the adjoint system of equations might be handled as a matrix-free system in PETSc. In this case, the entries of the Jacobian are evaluated on a row-by-row basis for every iteration of the GMRES solver, leading to smaller memory requirements at the expense of a larger CPU cost. The *ADjoint* approach still retains all of its advantages and can trade the higher memory requirements for increased CPU time in the solution of the discrete adjoint problem. Even if the matrix-free version of PETSc were to be used, the cost of a single *ADjoint* solution is estimated to be lower than that of the flow solution: typical continuous adjoint solvers require computational times for solutions that are very close to the cost of a single flow solution.

Another comparison was made for the computational cost of the *ADjoint*- and FD-based sensitivities. The values summarized in table 6.24 were gathered while performing the comparison of the adjoint-based sensitivities with FD approximations, that have been shown previously. Following the previously shown test cases, the efficiency of the adjoint solver is again evident, outperforming tremendously the traditional finite-difference sensitivity method. In this case, using a non-optimized MHD flow solver, the adjoint-based sensitivity is roughly 30 times faster, per function of interest and design variable, than the FD sensitivity. Obviously the final ratio depends on the number of functions of interest (the larger this is, the less advantageous the

	Wall clock time			
	Euler ($550k$)	Low MHD ($550k$)	Ideal MHD ($290k$)	
Flow solution	244.6	255.9	343.6	[minutes]
Sensitivities via *ADjoint*	4.9	5.4	7.9	[minutes/cost function]
Sensitivities via finite-diff.	160.9	146.3	210.2	[minutes/design var]

Table 6.24: Generic vehicle: cost comparison of *ADjoint* and FD gradients.

adjoint becomes), and the number of design variables (the larger it gets, the more efficient the adjoint method becomes).

In realistic design problems with optimized flow and adjoint solvers, having 5–10 functions of interest and on the order of 200 design variables, the automatic discrete adjoint-based gradients are expected to be obtained 50–100 times faster compared to finite-difference approximations.

6.4.6 Sample design problem using the low Re_σ MHD solver

The *ADjoint*-based sensitivity analysis module developed for the low Re_σ MHD solver was put into practice on a gradient-based optimization application. The ideal MHD model was not tested in this environment because of limitations in the available computer time.

The design problem was a re-entry hypersonic vehicle in the atmosphere, in which both the vehicle attitude and dipole properties were taken as control variables, $\boldsymbol{x}$, with the objective of maximizing the inviscid drag coefficient, C_D,

$$\begin{aligned} Maximize \quad & C_D \\ w.r.t. \quad & x \qquad (6.5) \\ s.t. \quad & C_{Lmin} \leq C_L \leq C_{Lmax} \end{aligned}$$

(6.6)

In order to make the flow turnaround time faster, a coarser mesh with $98,288$ nodes was used, and the relative L2-norm for convergence was lowered to 1×10^{-6}.

A total of 23 design variables, x, were considered: angle of attack, side-slip angle, dipole strengths (7) and dipole orientations (14). Their upper and lower bounds, and the initial and optimum values are compiled in table 6.25. The problem had a constraint on the inviscid lift coefficient, as indicated in the bound values found in the previously mentioned table.

	Variable	Lower bound	Upper bound	Baseline	Optimized
Vehicle attitude	Angle-of-attack	-0.1745	0.1745	0.0349	0.0971
	Side-slip angle	0.0000	0.0000	0.0000	0.0000
Dipole #1	m_1	-0.0150	-0.0001	-0.0050	-0.0150
	α_1	-0.6981	0.6981	0.0000	-0.5357
	β_1	-0.3491	0.3491	0.0000	-0.3491
Dipole #2	m_2	0.0001	0.0150	0.0050	0.0150
	α_2	-0.6981	0.6981	0.0000	0.6981
	β_2	1.5708	2.2689	1.9251	1.5708
Dipole #3	m_3	0.0001	0.0150	0.0050	0.0150
	α_3	-0.6981	0.6981	0.0000	-0.4893
	β_3	-2.2689	-1.5708	-1.9251	-1.9112
Dipole #4	m_4	0.0001	0.0150	0.0050	0.0150
	α_4	-0.6981	0.6981	0.0000	0.4662
	β_4	1.5708	2.2689	1.9251	1.5708
Dipole #5	m_5	0.0001	0.0150	0.0050	0.0001
	α_5	-0.6981	0.6981	0.0000	0.0000
	β_5	-2.2689	-1.5708	-1.9251	-1.9251
Dipole #6	m_6	0.0001	0.0150	0.0050	0.0150
	α_6	-0.6981	0.6981	0.0000	0.6981
	β_6	1.5708	2.2689	1.9251	1.5794
Dipole #7	m_7	0.0001	0.0150	0.0050	0.0001
	α_7	-0.6981	0.6981	0.0000	0.0000
	β_7	-2.2689	-1.5708	-1.9251	-1.9251
Lift coef.	C_L	0.0750	0.0900	0.0420	0.0900
Drag coef.	C_D	-/-	-/-	0.0133	0.0224

Table 6.25: Generic vehicle: baseline and optimized design variables.

The baseline flow conditions corresponded to a Mach number of $M = 5$, a magnetic force number of $R_b = 0.11$, and a magnetic Reynolds number of $Re_\sigma = 0.19$. Thus, the magnetic interaction parameter was $Q = 0.02$.

The design was performed using the SNOPT optimizer and its convergence history is shown in figure 6.47.

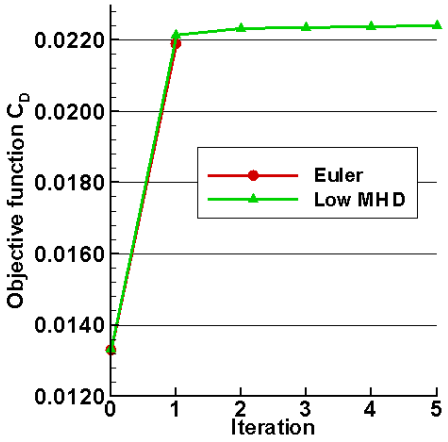


Figure 6.47: Generic vehicle: optimization problem history.

The optimal design found after the optimizer converged to the specified tolerances is described in the last column of table 6.25. It can be seen that the dipole strengths had their bounds active, and that the lift constraint was satisfied at its upper bound.

The number of SNOPT major iterations and functions calls done by the optimizer are included in table 6.26, where the values obtained while running the problem without any magnetic effects (degenerating to the Euler equations model) were also included as reference. Notice that a function call corresponds to one flow and two adjoint (cost function C_D and constraint on C_L) solutions.

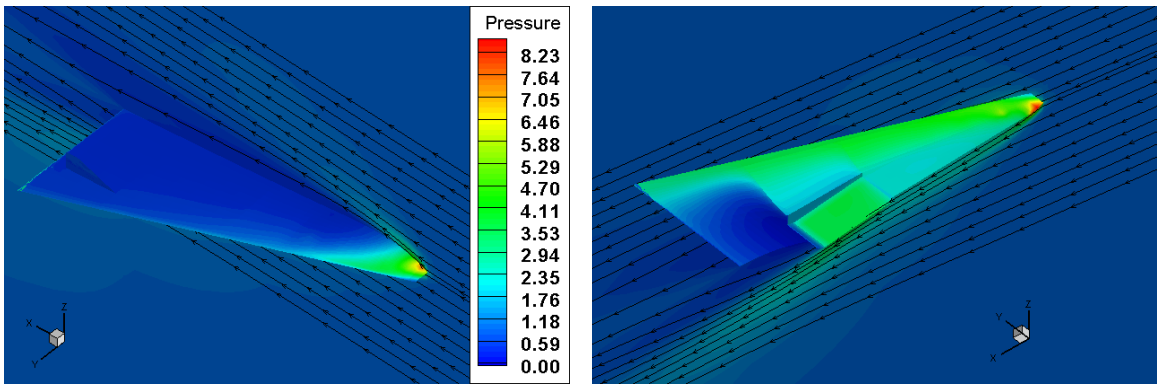
Case	Iter.	F.Call	C_D	C_L	Time[sec]
(Baseline)			0.0133	0.0420	-/-
Euler	1	3	0.0218	0.0894	3,278
Low	5	9	0.0224	0.0900	8,649

Table 6.26: Generic vehicle: functions of interest of design problem.

Table 6.26 also shows the influence of the magnetic control, where an improvement of 6 drag counts was obtained using MHD. It also demonstrates the additional

computational cost incurred by solving the less stable MHD flow equations, compared to the simpler Euler model, more than doubling the required wall clock time.

The flow field pressure and the velocity streamlines for the optimized configuration are shown in figure 6.48 for the low Re_σ model. Had the Euler model been presented, the differenced would have been almost imperceptible, with the exception of localized pressure increase close to the dipoles in the magnetic case shown.



(a) top view (b) bottom view

Figure 6.48: Generic vehicle: pressure contours on optimized vehicle.

The results shown for this test case clearly attest to the validity and efficiency of the *ADjoint* approach to estimate gradient information in large, complex problems modeled by the MHD equations. This provides a solid ground to even more complex MHD flow control problems that have not been possible to grasp by the scientific community yet.

Chapter 7

Conclusions and future developments

This dissertation outlines an effective method to develop a scalable tool to compute sensitivities of functions of interest that arise in large-scale CFD design problems, regardless of the complexity of the governing equations or boundary conditions.

The method paves the way to implement a discrete adjoint solvers for arbitrary governing equations making use of available software tools, namely *Automatic Differentiation* tools. The approach described in this dissertation is particularly well suited to compute the gradients of any function of interest in an optimization problem involving any set of governing equations that can be cast in the form $\mathcal{R}(\boldsymbol{x}, \boldsymbol{w}(\boldsymbol{x})) = 0$, when the number of design variables considerably outnumbers the number of cost functions or when the solution of the governing equations is computationally too expensive to allow the use of finite-differences.

The implementation of the adjoint solver has been largely automated, thus the use of the name *ADjoint* (Automatic Differentiation adjoint), eliminating the need for hand-differentiation of the governing equations . In addition, this approach does not require any simplifications in the derivation of the adjoint equations nor any special treatment of the boundary conditions.

But the major advantage of the *ADjoint* approach when compared to traditional approaches is the fact that it drastically reduces the implementation time: while the

typical development of a continuous adjoint solver could take up to a year of work for a well trained researcher, the *ADjoint* can take as little as a week, provided that a flow solver is already available. Besides expediting the development time, this method also produces gradients that are exactly consistent with the flow solver discretization and permits the use of arbitrary cost and constraint functions, and design variables.

This work has pioneered the extension of the discrete adjoint approach to the control of a hypersonic flow in the presence of magnetic fields, and successfully demonstrated its feasibility in simple design problems governed by MHD governing equations and using up to a half million design variables. The *ADjoint* approach was successfully applied to two distinct MHD flow solvers (a cell-centered, single-block solver, and a vertex-centered, multi-block solver), and the total sensitivities obtained from the corresponding discrete adjoint solvers showed excellent agreement with the results produced by finite-difference methods. It must be noted that the *ADjoint* derivation was presented here for the MHD equations, but since this approach is only based on the existence of a computer program that evaluates the *residual* of the governing equations (3.15), the procedure can be extended to any arbitrary set of arbitrary governing PDEs without modification.

This approach has the advantage that it uses the reverse mode of differentiation on the code that computes the residuals on a cell-per-cell (or node-per-node) basis for the governing equations and, therefore, it is highly time-efficient.

Compared pure automatic differentiation, the timings presented show that this hybrid approach is significantly faster and, by far, less memory intensive. However, the memory usage is still considerably higher than that of the corresponding flow solver. This drawback is not even significant given the fact that not only the hardware resources keep growing and getting more accessible, but also because the designers often use relatively smaller flow problems (through domain decomposition in a parallel computer) in order to get reasonable turn-around times in the flow solution.

Although it can be argued that the penalty in storage that this method incurs is largely outweighed by the substantial benefits over current methodologies used to develop adjoint solvers, this handicap can by addressed with the way the adjoint matrix is handled in the solution of the adjoint system of equations. In this dissertation, the

matrix is fully assembled prior to calling the iterative solver, however, this approach maximizes the memory requirements. Another option, instead of pre-assembling the matrix, would be running the iterative solver in matrix-free mode, that is, the entries of the matrix would be evaluated as needed (on-the-fly), with an added overhead in run-time. Understandably, the latter mode incurs in maximum computational cost because the matrix would have to be evaluated as many times as the number of iterations needed to converge the solution. This trade-off between CPU cost and memory requirements in the adjoint system assembly is quantitatively depicted in figure 7.1. The extreme cases are highlighted: compute once and store all; always

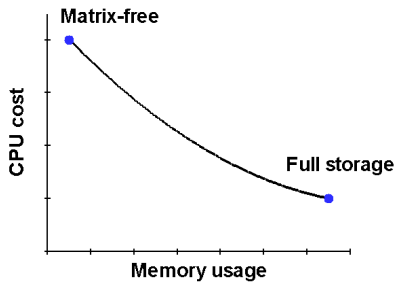


Figure 7.1: Adjoint matrix handling: trade-off between CPU cost and memory usage.

recompute and never store. A compromise is also possible, that is, some elements of the matrix would be pre-computed, and the rest computed in each GMRES iteration. In addition, further memory savings could be achieved if not all Krylov subspaces were stored during the iterative GMRES procedure. All these options can bring the memory requirements of this approach down to a level similar to the conventional continuous adjoint approach.

Currently, few research groups have been able to develop adjoint codes, largely due to the sheer effort required. The *ADjoint* approach is meant to facilitate the implementation of adjoint methods and it can, potentially, become a popular choice among researchers wishing to perform efficient sensitivity analysis and optimization. The adjoint technique might finally become accessible to engineering design at an industrial level, and not only restricted to highly specialized academic research groups, as it has been so far.

7.1 *ADjoint* approach

The hybrid sensitivity analysis method, *ADjoint*, has been the focal point of this dissertation. It is based on several key theories and components that, when put together, produce this easy to implement, generic and efficient method. The next sub-sections synthesize the features of each component in a systematic way, starting from the general adjoint-based sensitivity analysis, and ending at the automatically derived discrete adjoint solver.

Adjoint method

In the context of gradient-based optimization, the use of adjoint methods is an efficient and accurate way to estimate sensitivity information in problems where the number of variables largely exceeds the number of functions of interest or the governing equations are computationally too expensive to allow finite-difference approximations.

The adjoint method is characterized by:

- Additional solver required, with development and run-time costs that are similar to those of the flow solver;
- Allows for the computation of the sensitivity of one function with respect to a set of design variables by solving both the flow and the adjoint solvers one time only, *independently* of the number of design variables.

Discrete adjoint approach

The adjoint formulation can be classified in continuous or discrete. The discrete adjoint is thought to be the most suitable approach to follow because of the following properties:

- Well defined procedure to derive adjoint equations *independently* of the complexity of the governing equations;
- Ability to treat *arbitrary* cost functions and constraints (unlike continuous formulation that can only deal with certain classes of integral functions);

- Gradients computed are *consistent* with the flow solver;
- Allows for the use of *automatic* differentiation tools (but at the expense of increased *memory* requirements).

Automatic differentiation tools

The use of automatic differentiation (AD) tools in the derivation of the discrete adjoint equations brings tremendous advantages, namely:

- *Effortlessly* computes the adjoint system of equations of arbitrary complex governing equations, provided that the flow solver has already been coded, dramatically reducing the development time;
- Allows faster execution times and reduced memory requirements, with the *selective* application of AD tools;
- Allows the derivation of the adjoint equations with *no simplifications* of the actual flow residual terms.

7.2 Future developments

This work represents the first step toward an automatic design framework for problems involving hypersonic flow control using electromagnetic effects.

Future work might include the incorporation of non-ideal MHD effects in the governing equations, corresponding to the full MHD formulation, specifically the viscous Navier–Stokes terms and the magnetic dispersive terms.

In addition, the flow solver efficiency can be improved, in particular, by substituting the explicit time-integration scheme by an implicit method, since the former performs poorly when large magnetic fields produce extremely fast magneto-acoustic waves. With the current scheme, the permissible time step becomes extremely small due to the CFL condition, limiting the usefulness of the numerical model. By doing so, the disadvantage of the finite-difference approach over the discrete adjoint would

not be so large, but still orders of magnitude worse, and the overall design framework efficiency would improve significantly.

As far as the adjoint solver implementation is concerned, the option of running the adjoint solver in matrix-free mode should be implemented to reduce memory requirements so that they do not exceed those of the flow solver.

The last step in the development of an automatic design framework is to integrate the *ADjoint* solver as a module to compute sensitivities, together with an array of other components, such as other multi-disciplinary analysis modules (of which the MHD flow solver is part), and grid generators with perturbation capabilities, just to name a few. The final product is aimed to be a high-fidelity MDO environment, offering to a designer very high-level functionalities through the use of scripting languages, such as Python [136].

Another topic of future research might also be the handling of design problems involving not only a large number of design variables but also a large number of constraints. Posing the design problem in the conventional form (1.3), this would require to compute as many adjoint solutions as the number of constraints, every time the gradient-based optimizer had to evaluate the derivatives. However, some ideas can be brought from the structural optimization field, such the use of penalty functions [157] or lumped constraints [4].

Once the flow solver is extended and optimized, and the *ADjoint*-based sensitivity module is integrated in a design framework, the investigation of meaningful MHD design problems and the definition of significant cost functions can finally be tackled.

Appendix A

Vector calculus

Some definitions and vector identities useful in the derivation of the magnetohydrodynamic equations are listed in the sections below.

A.1 Dyadic product

$$\mathbf{P} = \mathbf{uv} = \mathbf{u} \otimes \mathbf{v} = \left\{ \begin{array}{c} u_1 \\ u_2 \\ u_3 \end{array} \right\} \otimes \{v_1 v_2 v_3\} = \left[\begin{array}{ccc} u_1 v_1 & u_1 v_2 & u_1 v_3 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 \\ u_3 v_1 & u_3 v_2 & u_3 v_3 \end{array} \right] , \tag{A.1}$$

or in Einstein's (index) notation,

$$P_{ij} = u_i v_j \, . \tag{A.2}$$

A.2 Vector identities

$$\nabla \times (\nabla \phi) = 0 \tag{A.3}$$

$$\nabla \cdot (\nabla \times \mathbf{u}) = 0 \tag{A.4}$$

$$\mathbf{u} \times (\mathbf{v} \times \mathbf{w}) = (\mathbf{u} \cdot \mathbf{w})\, \mathbf{v} - (\mathbf{u} \cdot \mathbf{v})\, \mathbf{w} \tag{A.5}$$

$$\mathbf{u}\cdot(\mathbf{v}\times\mathbf{w})=\mathbf{v}\cdot(\mathbf{w}\times bfu)=\mathbf{w}\cdot(\mathbf{u}\times\mathbf{v}) \tag{A.6}$$

$$(\mathbf{u}\times\mathbf{v})\cdot(\mathbf{w}\times\mathbf{x})=(\mathbf{u}\cdot\mathbf{w})(\mathbf{v}\cdot\mathbf{x})-(\mathbf{u}\cdot\mathbf{x})(\mathbf{v}\cdot\mathbf{w}) \tag{A.7}$$

$$\nabla\cdot(\mathbf{u}\times\mathbf{v})=\mathbf{v}\cdot(\nabla\times\mathbf{u})-\mathbf{u}\cdot(\nabla\times\mathbf{v}) \tag{A.8}$$

$$\begin{aligned}\nabla\times(\mathbf{u}\times\mathbf{v}) &= (\nabla\cdot\mathbf{v})\,\mathbf{u}-(\nabla\cdot\mathbf{u})\,\mathbf{v}+(\mathbf{v}\cdot\nabla)\,\mathbf{u}-(\mathbf{u}\cdot\nabla)\,\mathbf{v}\\ &= \nabla\cdot(\mathbf{vu}-\mathbf{uv})\end{aligned} \tag{A.9}$$

$$(\nabla\times\mathbf{u})\times\mathbf{u}=(\mathbf{u}\cdot\nabla)\,\mathbf{u}-\frac{1}{2}\nabla(\mathbf{u}\cdot\mathbf{u})$$

or equivalently,

$$(\mathbf{u}\cdot\nabla)\,\mathbf{u}=\nabla\left(\frac{1}{2}u^2\right)-\mathbf{u}\times(\nabla\times\mathbf{u}) \tag{A.10}$$

$$\nabla\cdot(\mathbf{uv})=(\mathbf{u}\cdot\nabla)\,\mathbf{v}+\mathbf{v}(\nabla\cdot\mathbf{u}) \tag{A.11}$$

$$\nabla(\mathbf{u}\cdot\mathbf{v})=(\mathbf{u}\cdot\nabla)\,\mathbf{v}+(\mathbf{v}\cdot\nabla)\,\mathbf{u}+\mathbf{u}\times(\nabla\times\mathbf{v})+\mathbf{v}\times(\nabla\times\mathbf{u}) \tag{A.12}$$

$$\nabla\cdot(\phi\mathbf{u})=\nabla\phi\cdot\mathbf{u}+\phi\nabla\cdot\mathbf{u} \tag{A.13}$$

$$\nabla\times(\phi\mathbf{u})=\nabla\phi\times\mathbf{u}+\phi\nabla\times\mathbf{u} \tag{A.14}$$

$$\nabla\times(\nabla\times\mathbf{u})=\nabla(\nabla\cdot\mathbf{u})-\nabla^2\mathbf{u} \tag{A.15}$$

Appendix B

Full MHD equations with magnetic field decomposition

B.1 Maxwell's equations

The Maxwell's equations represent one of the most elegant and concise ways to state the fundamentals of electricity and magnetism. From them one can develop most of the working relationships in the field.

According to Panofksy [126], the complete Maxwell's equations can be expressed as

$$\nabla \cdot \mathbf{D} = \rho_e \quad \text{(Gauss' Law for Electricity)} \tag{B.1}$$

$$\nabla \cdot \mathbf{B} = 0 \quad \text{(Gauss' Law for Magnetism)} \tag{B.2}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \quad \text{(Faraday's Law of Induction)} \tag{B.3}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \quad \text{(Ampère-Maxwell Law)}\,, \tag{B.4}$$

where $\mathbf{D}$ is the electric displacement, $\mathbf{B}$ is the magnetic induction field, $\mathbf{E}$ and $\mathbf{H}$ are the electric and magnetic field strength vectors, respectively, $\mathbf{J}$ is the total electric current density vector, and ρ_e is the charge density.

The relations between $\mathbf{D}$ and $\mathbf{E}$, and between $\mathbf{B}$ and $\mathbf{H}$, are called constitutive

equations. In general, these constitutive relations are

$$\mathbf{D} = \epsilon_0 \mathbf{E} + \mathbf{P} \tag{B.5}$$

$$\mathbf{B} = \mu_m \left(\mathbf{H} + \mathbf{M} \right) , \tag{B.6}$$

where $\mathbf{P}$ is the polarization and $\mathbf{M}$ is the magnetization. The electric permittivity ϵ_0 and magnetic permeability μ_m are related by the speed of light as $c = 1/\sqrt{\mu_m \epsilon_0}$.

In the absence of magnetic or polarizable media, the total electric current density is given by

$$\mathbf{J} = \rho_e \mathbf{u} + \mathbf{j} , \tag{B.7}$$

where $\rho_e \mathbf{u}$ is the convection current density, with $\mathbf{u}$ denoting the velocity field of the medium, and $\mathbf{j}$ is the conduction current density. By definition, these assumptions imply that both $\mathbf{M}$ and $\mathbf{P}$ are zero since the medium is not magnetic nor polarizable, respectively¿ The constitutive relations (B.5) and (B.6) for a linear isotropic medium simplify, and allow an easy substitution into equations (B.1) and (B.4).

The force per unit volume of matter (Lorentz force) exerted on a particle with charge q is expressed as

$$\mathbf{F}_{Lorentz} = q\mathbf{E} + \mathbf{J} \times \mathbf{B} , \tag{B.8}$$

and the power delivered to matter by the field is

$$P = \mathbf{E} \cdot \mathbf{J} . \tag{B.9}$$

The generalized Ohm's Law is given by

$$\mathbf{j} = \sigma(\mathbf{E} + \mathbf{u} \times \mathbf{B}) \quad \text{(Ohm's Law)} , \tag{B.10}$$

where σ is the electrical conductivity and the convection, polarization and Hall current components have been neglected.

According to Gaitonde and Poggie [46], significant simplifications can be made under the assumptions that 1) the flow time scales are larger than the reciprocal of the plasma frequency (i.e., the system under consideration is a good conductor),

$\frac{\epsilon\omega}{\sigma} \ll 1$, where ω is a representative frequency of interest; and 2) the flow velocities are much less than the speed of light, $(\frac{U}{c})^2 \ll 1$, where U is the velocity of the conducting medium. The first of these assumptions implies that the displacement current, $\frac{\partial \mathbf{D}}{\partial t}$, in the Ampère-Maxwell law (B.4) can be neglected. The resulting set of equations is sometimes termed "pre-Maxwell". The second assumption permits relativistic effects to be ignored. These assumptions also allow to neglect the convection current density when compared to the conduction current density in equation (B.7), resulting in $\mathbf{J} \approx \mathbf{j}$.

In addition, since the charge separation is small, the force due to the electric field might also be neglected. As such, the Lorentz force (B.8) can be reduced to

$$\mathbf{F}_{Lorentz} \approx \mathbf{j} \times \mathbf{B}\,, \tag{B.11}$$

and the power delivered to matter by the field (B.9) simplifies to

$$P \approx \mathbf{E} \cdot \mathbf{j}\,. \tag{B.12}$$

The current density may then be expressed from the Ampère's law (B.4) as

$$\mathbf{j} = \nabla \times \mathbf{H} = \nabla \times \frac{\mathbf{B}}{\mu_m}\,. \tag{B.13}$$

More details about the material presented in this section can be found in references [164] and [120].

B.2 Coupling of the Maxwell's and Navier–Stokes equations

The governing equations of magnetohydrodynamics are obtained by coupling the "pre-Maxwell" equations to the Navier–Stokes equations through the momentum and energy equations. An additional equation — the magnetic induction field transport — is derived from the Faraday's law of induction.

B.2.1 Continuity equation

The conservation of mass expressed by the continuity equation retains the same form as given by the Navier–Stokes equations, namely,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \,. \tag{B.14}$$

B.2.2 Momentum equations

The momentum equation contains an extra electromagnetic body force that results from the Lorentz force created by the presence of a magnetic field. Combining equations (B.11) and (B.13) leads to

$$\mathbf{F}_{em} = \left(\nabla \times \frac{\mathbf{B}}{\mu_m} \right) \times \mathbf{B} \,. \tag{B.15}$$

Making use of the vector identities (A.10) and (A.11), the electromagnetic force may be expressed in the form of Maxwell's stresses as

$$\mathbf{F}_{em} = (\mathbf{B} \times \nabla) \frac{\mathbf{B}}{\mu_m} - \nabla \left(\frac{\mathbf{B} \cdot \mathbf{B}}{2\mu_m} \right) = \nabla \cdot \left(\frac{\mathbf{B}\mathbf{B}}{\mu_m} \right) - \frac{\mathbf{B}}{\mu_m} (\nabla \cdot \mathbf{B}) - \nabla \left(\frac{\mathbf{B} \cdot \mathbf{B}}{2\mu_m} \right) \,. \tag{B.16}$$

The second term, even though is physically zero — because of the divergence-free condition of the magnetic field — it is retained for numerical stability and implicit enforcement of that condition.

This additional force acting on the flow is included in the Navier–Stokes momentum equation,

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\nabla p + \nabla \cdot \vec{\vec{\tau}} + \mathbf{F}_{em} \,, \tag{B.17}$$

which upon substitution of (B.16) for $\mathbf{F}_{em}$, and rearranging terms, yields

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot \left(\rho \mathbf{u}\mathbf{u} - \frac{\mathbf{B}\mathbf{B}}{\mu_m} + P\mathbf{I} - \vec{\vec{\tau}} \right) = -\frac{\mathbf{B}}{\mu_m} (\nabla \cdot \mathbf{B}) \,, \tag{B.18}$$

where $\mathbf{I}$ is the identity tensor, $\vec{\vec{\tau}}$ is the shear stress tensor and P is static MHD pressure

given as the sum of the static and magnetic pressures,

$$P = p + \frac{\mathbf{B} \cdot \mathbf{B}}{2\mu_m} \,. \tag{B.19}$$

B.2.3 Energy equation

The energy equation is modified with the addition of the electromagnetic energy term E_{em} given by expression (B.12).

Solving Ohm's law (B.10) for $\mathbf{E}$ yields

$$\mathbf{E} = \frac{\mathbf{j}}{\sigma} - \mathbf{u} \times \mathbf{B} \,, \tag{B.20}$$

where the term $\frac{\mathbf{j}}{\sigma}$ is zero for ideal MHD. Substituting (B.13) into (B.20) results

$$\mathbf{E} = \frac{1}{\sigma} \left(\nabla \times \frac{\mathbf{B}}{\mu_m} \right) - \mathbf{u} \times \mathbf{B} \,. \tag{B.21}$$

On the other hand, substituting (B.13) into (B.12), and recalling the identity (A.8), the electromagnetic energy can be written as

$$E_{em} = \frac{\mathbf{B}}{\mu_m} \cdot (\nabla \times \mathbf{E}) + \nabla \cdot \left(\frac{\mathbf{B}}{\mu_m} \times \mathbf{E} \right) \,, \tag{B.22}$$

Expressing Faraday's law (B.3) for a moving medium as given by Vinokur's [159],

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} + \mathbf{u} \, (\nabla \cdot \mathbf{B}) = 0 \,, \tag{B.23}$$

and rearranging it, yields

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} - \mathbf{u} \, (\nabla \cdot \mathbf{B}) = 0 \,. \tag{B.24}$$

Substituting (B.21) and (B.24) into (B.22), manipulating the terms and making

use of equality (A.5), the electromagnetic energy can then be expressed as

$$\begin{aligned} E_{em} &= -\frac{\partial}{\partial t}\left(\frac{\mathbf{B}\cdot\mathbf{B}}{2\mu_m}\right) - \left(\mathbf{u}\cdot\frac{\mathbf{B}}{\mu_m}\right)\nabla\cdot\mathbf{B} + \\ &+ \nabla\cdot\left[\nabla\frac{\mathbf{B}}{\mu_m}\cdot\frac{\mathbf{B}}{\mu_m\sigma} - \frac{\mathbf{B}}{\mu_m\sigma}\cdot\nabla\frac{\mathbf{B}}{\mu_m}\right] - \\ &- \nabla\cdot\left[\left(\frac{\mathbf{B}\cdot\mathbf{B}}{2\mu_m}\right)\mathbf{u} + \left(\frac{\mathbf{B}\cdot\mathbf{B}}{2\mu_m}\right)\mathbf{u} - \mathbf{B}\left(\mathbf{u}\cdot\frac{\mathbf{B}}{\mu_m}\right)\right]. \end{aligned} \tag{B.25}$$

Recalling the Navier–Stokes energy equation, and including the additional electromagnetic energy contribution, it writes

$$\frac{\partial\rho E}{\partial t} + \nabla\cdot\left[(\rho E + p)\,\mathbf{u} - \mathbf{u}\cdot\vec{\vec{\tau}} + \mathbf{Q}\right] = E_{em}\,, \tag{B.26}$$

where the total energy E can be expressed, if the ideal gas assumption is made, as

$$\rho E = \frac{p}{\gamma - 1} + \rho\frac{\mathbf{u}\cdot\mathbf{u}}{2}\,, \tag{B.27}$$

and the heat transfer rate, using the Fourier's hypothesis, is $\mathbf{Q} = -\kappa\nabla T$, where κ is the thermal conductivity coefficient and T is the temperature.

Substituting (B.25) into (B.26) and rearranging terms leads to

$$\begin{aligned} \frac{\partial}{\partial t}\left(\rho E + \frac{\mathbf{B}\cdot\mathbf{B}}{2\mu_m}\right) &+ \nabla\cdot\left[\left(\rho E + \frac{\mathbf{B}\cdot\mathbf{B}}{2\mu_m} + p + \frac{\mathbf{B}\cdot\mathbf{B}}{2\mu_m}\right)\mathbf{u} - \mathbf{B}\left(\mathbf{u}\cdot\frac{\mathbf{B}}{\mu_m}\right) - \mathbf{u}\cdot\vec{\vec{\tau}} + \mathbf{Q} + \right. \\ &+ \left.\left(\frac{\mathbf{B}}{\mu_m\sigma}\cdot\nabla\frac{\mathbf{B}}{\mu_m} - \nabla\frac{\mathbf{B}}{\mu_m}\cdot\frac{\mathbf{B}}{\mu_m\sigma}\right)\right] = -\left(\mathbf{u}\cdot\frac{\mathbf{B}}{\mu_m}\right)\nabla\cdot\mathbf{B}\,. \end{aligned} \tag{B.28}$$

Defining the MHD total energy per unit volume, ρZ, as being composed of the usual total energy, ρE, increased by the magnetic energy contribution,

$$\rho Z = \rho E + \frac{\mathbf{B}\cdot\mathbf{B}}{2\mu_m}\,, \tag{B.29}$$

and recalling the MHD static pressure (B.19) definition, the energy equation (B.28)

can be written in a more compact form as

$$\begin{aligned}\frac{\partial \rho Z}{\partial t} &+ \nabla \cdot \left[(\rho Z + P)\,\mathbf{u} - \mathbf{B}\left(\mathbf{u}\cdot\frac{\mathbf{B}}{\mu_m}\right) - \mathbf{u}\cdot\vec{\vec{\tau}} + \mathbf{Q} + \right. \\ &+ \left. \left(\frac{\mathbf{B}}{\mu_m\sigma}\cdot\nabla\frac{\mathbf{B}}{\mu_m} - \nabla\frac{\mathbf{B}}{\mu_m}\cdot\frac{\mathbf{B}}{\mu_m\sigma}\right)\right] = -\left(\mathbf{u}\cdot\frac{\mathbf{B}}{\mu_m}\right)\nabla\cdot\mathbf{B}\,. \end{aligned} \tag{B.30}$$

B.2.4 Magnetic induction equations

The magnetic induction field transport equation is based on Faraday's law using Vinokur's expression (B.23). Upon substitution of the electric field **E** using Ohm's law combined with Ampère-Maxwell law (B.21), it leads to the equation describing the evolution of the magnetic induction field, **B**,

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \times \left[\frac{1}{\sigma}\nabla\times\left(\frac{\mathbf{B}}{\mu_m}\right) - \mathbf{u}\times\mathbf{B}\right] = -\mathbf{u}\left(\nabla\cdot\mathbf{B}\right)\,. \tag{B.31}$$

Expanding terms and using vector equality (A.9), results

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla\cdot(\mathbf{uB} - \mathbf{Bu}) + \nabla\times\left[\frac{1}{\sigma}\nabla\times\left(\frac{\mathbf{B}}{\mu_m}\right)\right] = -\mathbf{u}\left(\nabla\cdot\mathbf{B}\right)\,. \tag{B.32}$$

Equation (B.32) exhibits a great deal of similarity with the vorticity equation, and includes phenomena associated with convection, stretching and diffusion of the magnetic induction field.

A detailed description of the units and dimensions of each variable found in the magnetohydrodynamic equations can be found in Cramer's book [27].

A complete characterization of the full MHD equation terms can be found in Gaitonde's work [42].

B.3 Non-dimensionalization of the equations

The governing equations – (B.14), (B.18), (B.30) and (B.32) – may be non-dimensionalized. By doing so, it is possible to provide conditions upon which flow similarity may be obtained for geometrically similar cases. In addition, the solution of the equations

would be of unitary order of magnitude.

Using dimensional analysis (see see Anderson [6]), it is possible to identify the number of independent variables, which ought to span the dimensions present in the equations, and all the other dependent variables, whose dimensions can be obtained by combination of the previous variables. A set of parameters can be identified and several reference conditions may be selected to accomplish this task.

Several reference conditions may be selected to accomplish this task but, for the present case, a characteristic length, L_{ref}, density, ρ_{ref}, velocity, U_{ref}, molecular viscosity, μ_{ref}, temperature, T_{ref}, thermal conductivity, κ_{ref}, magnetic field, B_{ref}, magnetic permeability, μ_{mref} and electrical conductivity, σ_{ref}, are used.

The non-dimensional variables are then

$$\begin{array}{lllll} t^* = \frac{tU_{ref}}{L_{ref}} & x^* = \frac{x}{L_{ref}} & y^* = \frac{y}{L_{ref}} & z^* = \frac{z}{L_{ref}} & \mu^* = \frac{\mu}{\mu_{ref}} \\ \rho^* = \frac{\rho}{\rho_{ref}} & u^* = \frac{u}{U_{ref}} & v^* = \frac{v}{U_{ref}} & w^* = \frac{w}{U_{ref}} & T^* = \frac{T}{T_{ref}} \\ T^* = \frac{T}{T_{ref}} & p^* = \frac{p}{\rho_{ref}U_{ref}^2} & E^* = \frac{E}{U_{ref}B_{ref}} & \kappa^* = \frac{\kappa}{\kappa_{ref}} & \mu_m^* = \frac{\mu_m}{\mu_{mref}} \\ B_x^* = \frac{B_x}{B_{ref}} & B_x^* = \frac{B_x}{B_{ref}} & B_x^* = \frac{B_x}{B_{ref}} & \sigma^* = \frac{\sigma}{\sigma_{ref}} & \nabla^* = L_{ref}\nabla \end{array} \tag{B.33}$$

The non-dimensional parameters found in this process are the Mach number,

$$M = \frac{U_{ref}}{c_{ref}}\,, \tag{B.34}$$

where $c_{ref} = \sqrt{\frac{\gamma p_{ref}}{\rho_{ref}}}$ for perfect gas, the Reynolds number,

$$Re = \frac{\rho_{ref}U_{ref}L_{ref}}{\mu_{ref}}\,, \tag{B.35}$$

the Prandtl number,

$$Pr = \frac{\mu C_p}{\kappa}\,, \tag{B.36}$$

the magnetic Reynolds number,

$$Re_\sigma = L_{ref}U_{ref}\mu_{mref}\sigma_{ref}\,, \tag{B.37}$$

the magnetic force (or pressure) number,

$$R_b = \frac{B_{ref}^2}{\rho_{ref} U_{ref}^2 \mu_{mref}} \tag{B.38}$$

and the magnetic interaction parameter,

$$Q = R_b Re_\sigma = \frac{\sigma_{ref} B_{ref}^2 L_{ref}}{\rho_{ref} U_{ref}} \,. \tag{B.39}$$

For notational convenience, the superscript $(*)$ in the non-dimensional variables in the equations presented next is dropped. The presence of any non-dimensional parameter implies that the equations are already in non-dimensional form.

Continuity equation

Using the reference values (B.33), the conservation of mass equation (B.14) can be made nondimensional. It can be easily shown that it remains unchanged as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \,. \tag{B.40}$$

Momentum equation

Upon substitution of the reference values (B.33), and after some algebraic manipulation, the conservation of momentum equation (B.18) is written in non-dimensional form as

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot \left(\rho \mathbf{u}\mathbf{u} - R_b \frac{\mathbf{B}\mathbf{B}}{\mu_m} + P\mathbf{I} - \frac{1}{Re} \vec{\vec{\tau}} \right) = -R_b \frac{\mathbf{B}}{\mu_m} (\nabla \cdot \mathbf{B}) \,, \tag{B.41}$$

where the non-dimensional static MHD pressure is

$$P = p + R_b \frac{\mathbf{B} \cdot \mathbf{B}}{2\mu_m} \,. \tag{B.42}$$

Energy equation

Similarly, the energy equation (B.30) is transformed into

$$\begin{aligned}\frac{\partial \rho Z}{\partial t} &+ \nabla \cdot \left[(\rho Z + P)\,\mathbf{u} - R_b \mathbf{B} \left(\mathbf{u} \cdot \frac{\mathbf{B}}{\mu_m} \right) - \frac{1}{Re} \mathbf{u} \cdot \vec{\vec{\tau}} + \frac{\mu}{RePr(\gamma-1)M^2} \mathbf{Q} + \right. \\ &+ \left. \frac{R_b}{Re_\sigma} \left(\frac{\mathbf{B}}{\mu_m \sigma} \cdot \nabla \frac{\mathbf{B}}{\mu_m} - \nabla \frac{\mathbf{B}}{\mu_m} \cdot \frac{\mathbf{B}}{\mu_m \sigma} \right) \right] = -R_b \left(\mathbf{u} \cdot \frac{\mathbf{B}}{\mu_m} \right) \nabla \cdot \mathbf{B}\,, \end{aligned} \tag{B.43}$$

where

$$\rho Z = \rho E + R_b \frac{\mathbf{B} \cdot \mathbf{B}}{2\mu_m} \tag{B.44}$$

and

$$\mathbf{Q} = -\nabla T\,. \tag{B.45}$$

Induction equation

Lastly, the magnetic induction equation (B.32) may be expressed as

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{uB} - \mathbf{Bu}) + \frac{1}{Re_\sigma} \nabla \times \left[\frac{1}{\sigma} \nabla \times \left(\frac{\mathbf{B}}{\mu_m} \right) \right] = -\mathbf{u} \left(\nabla \cdot \mathbf{B} \right)\,. \tag{B.46}$$

B.4 Magnetic field decomposition

The magnetic field $\mathbf{B}$ may be decomposed into two components, the background imposed (or intrinsic) field $\mathbf{B}_0$ and the induced (or deviation) field $\mathbf{B}_i$,

$$\mathbf{B} = \mathbf{B}_0 + \mathbf{B}_i\,. \tag{B.47}$$

This decomposition allows for the derivation of MHD governing equations that

avoid the direct inclusion of the imposed components of the magnetic field as dependent variables, when the imposed magnetic field satisfies the conditions

$$\begin{aligned} \frac{\partial \mathbf{B}_0}{\partial t} &= 0 \quad , \\ \nabla \cdot \mathbf{B}_0 &= 0 \quad , \text{and} \\ \nabla \times \mathbf{B}_0 &= 0 \quad . \end{aligned} \tag{B.48}$$

These conditions mean that the imposed magnetic field $\mathbf{B}_0$ is time-invariant (steady), it satisfies Gauss' law for magnetism (meaning it is a solenoid field, without monopoles), and it is produced outside of the flow domain (no current sources in the domain), respectively.

Momentum equation

Introducing the decomposition (B.47) in the nondimensional momentum equation (B.41), expanding the terms, and using equalities (B.48), (A.10) and (A.11), it can be shown that the momentum equation can be alternatively expressed as

$$\begin{aligned} \frac{\partial \rho \mathbf{u}}{\partial t} &+ \nabla \cdot \left[\rho \mathbf{u}\mathbf{u} + P_i \mathbf{I} - R_b \frac{\mathbf{B}_i \mathbf{B}_i}{\mu_m} \right] + \\ &+ \nabla \cdot \left[R_b \frac{\mathbf{B}_0 \cdot \mathbf{B}_i}{\mu_m} \mathbf{I} - R_b \left(\frac{\mathbf{B}_0 \mathbf{B}_i}{\mu_m} + \frac{\mathbf{B}_i \mathbf{B}_0}{\mu_m} \right) \right] - \frac{1}{Re} \nabla \cdot \vec{\vec{\tau}} = \\ &= -R_b \frac{\mathbf{B}_0 + \mathbf{B}_i}{\mu_m} \left(\nabla \cdot \mathbf{B}_i \right) \ , \end{aligned} \tag{B.49}$$

where $P_i = p + R_b \frac{B_i^2}{2\mu_m}$.

Energy equation

The additional terms relative to the ideal MHD decomposed form presented by Powell *et al.* [133] are the Navier–Stokes viscous terms, $f = f(\mu)$, and the magnetic dispersion terms, $f = f(\sigma)$. Only the latter are affected by the decomposition of $\mathbf{B}$ (B.47).

Again, making the decomposition, and simplifying the equation by the use of

equalities (B.48), results

$$\begin{aligned}\frac{\partial \rho Z_i}{\partial t} &+ \nabla\cdot\left[(\rho Z_i + P_i)\,\mathbf{u} - R_b\mathbf{B}_i\left(\mathbf{u}\cdot\frac{\mathbf{B}_i}{\mu_m}\right) - \frac{1}{Re}\mathbf{u}\cdot\vec{\vec{\tau}} + \frac{\mu}{RePr(\gamma-1)M^2}\mathbf{Q} + \right.\\ &+ \frac{R_b}{Re_\sigma}\left(\frac{\mathbf{B}_i}{\mu_m\sigma}\times\frac{1}{\sigma}\left(\nabla\times\frac{\mathbf{B}_i}{\mu_m}\right)\right) + R_b\frac{\mathbf{B}_0\cdot\mathbf{B}_i}{\mu_m}\mathbf{u} - R_b\mathbf{B}_0\left(\mathbf{u}\cdot\frac{\mathbf{B}_i}{\mu_m}\right) + \\ &+ \left.\frac{R_b}{Re_\sigma}\left(\frac{\mathbf{B}_0}{\mu_m\sigma}\times\frac{1}{\sigma}\left(\nabla\times\frac{\mathbf{B}_i}{\mu_m}\right)\right)\right] = -R_b\left(\mathbf{u}\cdot\frac{\mathbf{B}_i}{\mu_m}\right)\nabla\cdot\mathbf{B}_i\,,\end{aligned} \tag{B.50}$$

where

$$\rho Z_i = \rho E + R_b\frac{\mathbf{B_i}\cdot\mathbf{B_i}}{2\mu_m}\,, \tag{B.51}$$

and similarly for the static pressure,

$$P_i = p + R_b\frac{\mathbf{B_i}\cdot\mathbf{B_i}}{2\mu_m}\,. \tag{B.52}$$

Induction equation

Applying the decomposition (B.47) to the magnetic induction equation (B.46), and subsequently simplifying the equation using equalities (B.48), leads to

$$\frac{\partial\mathbf{B}_i}{\partial t} + \nabla\cdot(\mathbf{u}\mathbf{B}_i - \mathbf{B}_i\mathbf{u}) + \frac{1}{Re_\sigma}\nabla\times\left[\frac{1}{\sigma}\nabla\times\left(\frac{\mathbf{B}_i}{\mu_m}\right)\right] + \nabla\cdot(\mathbf{u}\mathbf{B}_0 - \mathbf{B}_0\mathbf{u}) = -\mathbf{u}\,(\nabla\cdot\mathbf{B}_i)\,. \tag{B.53}$$

B.5 Decomposed flux vector form

Introducing the set of dependent variables $\boldsymbol{w}_i = (\rho, \rho\boldsymbol{u}, \rho Z_i, \boldsymbol{B}_i) = (\rho, \rho\boldsymbol{u}, \rho Z - (\boldsymbol{B}_i\cdot\boldsymbol{B}_0)/(\mu_m) - B0^2/(2\mu_m), \boldsymbol{B}-\boldsymbol{B}_0)$, the set of equations (B.40), (B.50), (B.51) and (B.53) can be expressed in flux vector form in Cartesian coordinates as

$$\frac{\partial\boldsymbol{w}}{\partial t} + \frac{\partial\mathbf{E}}{\partial x} + \frac{\partial\mathbf{F}}{\partial y} + \frac{\partial\mathbf{G}}{\partial z} = \mathbf{S}\,. \tag{B.54}$$

The fluxes can be separated according to the different physical contributions into

$$\begin{cases} \mathbf{E} = \mathbf{E}_i + \mathbf{E}_m + \mathbf{E}_d - \mathbf{E}_v \\ \mathbf{F} = \mathbf{F}_i + \mathbf{F}_m + \mathbf{F}_d - \mathbf{F}_v \\ \mathbf{G} = \mathbf{G}_i + \mathbf{G}_m + \mathbf{G}_d - \mathbf{G}_v \end{cases} , \tag{B.55}$$

where $\mathbf{E}_i$,$\mathbf{F}_i$, and $\mathbf{G}_i$ are the inviscid fluxes, $\mathbf{E}_m$,$\mathbf{F}_m$, and $\mathbf{G}_m$ contain terms relevant to perfectly conducting media (ideal MHD), while $\mathbf{E}_d$,$\mathbf{F}_d$, and $\mathbf{G}_d$ contain the effects due to finite electrical conductivity (full MHD), and $\mathbf{E}_v$,$\mathbf{F}_v$, and $\mathbf{G}_v$ include the viscous effects.

The solution vector $\boldsymbol{w}$ is then

$$\boldsymbol{w} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho Z_i \\ B_{ix} \\ B_{iy} \\ B_{iz} \end{pmatrix} , \tag{B.56}$$

and the various flux vectors of eq.(B.54) are given in the following sections.

Inviscid flux (Euler or NS equations)

$$\mathbf{E}_i = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (\rho E + p)u \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{B.57}$$

$$\mathbf{F}_i = \begin{pmatrix} \rho v \\ \rho v u \\ \rho v^2 + p \\ \rho v w \\ (\rho E + p) v \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{B.58}$$

$$\mathbf{G}_i = \begin{pmatrix} \rho w \\ \rho w u \\ \rho w v \\ \rho w^2 + p \\ (\rho E + p) w \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{B.59}$$

Magnetic flux (ideal or full MHD equations)

$$\mathbf{E}_m = \begin{pmatrix} 0 \\ R_b \frac{\mathbf{B}_i \cdot \mathbf{B}_i}{2\mu_m} - R_b \frac{B_{ix}^2}{\mu_m} + R_b \frac{\mathbf{B_0} \cdot \mathbf{B_i}}{\mu_m} - R_b \frac{B_{0x} B_{ix}}{\mu_m} - R_b \frac{B_{ix} B_{0x}}{\mu_m} \\ -R_b \frac{B_{ix} B_{iy}}{\mu_m} - R_b \frac{B_{0x} B_{iy}}{\mu_m} - R_b \frac{B_{ix} B_{0y}}{\mu_m} \\ -R_b \frac{B_{ix} B_{iz}}{\mu_m} - R_b \frac{B_{0x} B_{iz}}{\mu_m} - R_b \frac{B_{ix} B_{0z}}{\mu_m} \\ R_b \frac{\mathbf{B}_i \cdot \mathbf{B}_i}{\mu_m} u - R_b B_{ix} \frac{\mathbf{u} \cdot \mathbf{B}_i}{\mu_m} + R_b \frac{\mathbf{B}_0 \cdot \mathbf{B}_i}{\mu_m} u - R_b B_{0x} \frac{\mathbf{u} \cdot \mathbf{B}_i}{\mu_m} \\ 0 \\ (u B_{iy} - v B_{ix}) + (u B_{0y} - v B_{0x}) \\ (u B_{iz} - w B_{ix}) + (u B_{0z} - w B_{0x}) \end{pmatrix} \tag{B.60}$$

$$\mathbf{F}_m = \begin{pmatrix} 0 \\ -R_b\frac{B_{iy}B_{ix}}{\mu_m} - R_b\frac{B_{0y}B_{ix}}{\mu_m} - R_b\frac{B_{iy}B_{0x}}{\mu_m} \\ R_b\frac{\mathbf{B}_i\cdot\mathbf{B}_i}{2\mu_m} - R_b\frac{B_{iy}^2}{\mu_m} + R_b\frac{\mathbf{B_0}\cdot\mathbf{B_i}}{\mu_m} - R_b\frac{B_{0y}B_{iy}}{\mu_m} - R_b\frac{B_{iy}B_{0y}}{\mu_m} \\ -R_b\frac{B_{iy}B_{iz}}{\mu_m} - R_b\frac{B_{0y}B_{iz}}{\mu_m} - R_b\frac{B_{iy}B_{0z}}{\mu_m} \\ R_b\frac{\mathbf{B}_i\cdot\mathbf{B}_i}{\mu_m}v - R_bB_{iy}\frac{\mathbf{u}\cdot\mathbf{B}_i}{\mu_m} + R_b\frac{\mathbf{B}_0\cdot\mathbf{B}_i}{\mu_m}v - R_bB_{0y}\frac{\mathbf{u}\cdot\mathbf{B}_i}{\mu_m} \\ (vB_{ix} - uB_{iy}) + (vB_{0x} - uB_{0y}) \\ 0 \\ (vB_{iz} - wB_{iy}) + (vB_{0z} - wB_{0y}) \end{pmatrix} \tag{B.61}$$

$$\mathbf{G}_m = \begin{pmatrix} 0 \\ -R_b\frac{B_{iz}B_{ix}}{\mu_m} - R_b\frac{B_{0z}B_{ix}}{\mu_m} - R_b\frac{B_{iz}B_{0x}}{\mu_m} \\ -R_b\frac{B_{iz}B_{iy}}{\mu_m} - R_b\frac{B_{0z}B_{iy}}{\mu_m} - R_b\frac{B_{iz}B_{0y}}{\mu_m} \\ R_b\frac{\mathbf{B}_i\cdot\mathbf{B}_i}{2\mu_m} - R_b\frac{B_{iz}^2}{\mu_m} + R_b\frac{\mathbf{B_0}\cdot\mathbf{B_i}}{\mu_m} - R_b\frac{B_{0z}B_{iz}}{\mu_m} - R_b\frac{B_{iz}B_{0z}}{\mu_m} \\ R_b\frac{\mathbf{B}_i\cdot\mathbf{B}_i}{\mu_m}w - R_bB_{iz}\frac{\mathbf{u}\cdot\mathbf{B}_i}{\mu_m} + R_b\frac{\mathbf{B}_0\cdot\mathbf{B}_i}{\mu_m}w - R_bB_{0z}\frac{\mathbf{u}\cdot\mathbf{B}_i}{\mu_m} \\ (wB_{ix} - uB_{iz}) + (wB_{0x} - uB_{0z}) \\ (wB_{iy} - vB_{iz}) + (wB_{0y} - vB_{0z}) \\ 0 \end{pmatrix} \tag{B.62}$$

Dispersive magnetic flux (full MHD)

$$\mathbf{E}_d = \frac{1}{Re_\sigma}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ R_b\left[\frac{B_{0y}+B_{iy}}{\mu_m\sigma}\left(\frac{\partial}{\partial x}\frac{B_{iy}}{\mu_m} - \frac{\partial}{\partial y}\frac{B_{ix}}{\mu_m}\right) - \frac{B_{0z}+B_{iz}}{\mu_m\sigma}\left(\frac{\partial}{\partial z}\frac{B_{ix}}{\mu_m} - \frac{\partial}{\partial x}\frac{B_{iz}}{\mu_m}\right)\right] \\ 0 \\ \frac{1}{\sigma}\left(\frac{\partial}{\partial x}\frac{B_{iy}}{\mu_m} - \frac{\partial}{\partial y}\frac{B_{ix}}{\mu_m}\right) \\ \frac{1}{\sigma}\left(\frac{\partial}{\partial x}\frac{B_{iz}}{\mu_m} - \frac{\partial}{\partial z}\frac{B_{ix}}{\mu_m}\right) \end{pmatrix} \tag{B.63}$$

$$\mathbf{F}_d = \frac{1}{Re_\sigma}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ R_b\left[\frac{B_{0z}+B_{iz}}{\mu_m\sigma}\left(\frac{\partial}{\partial y}\frac{B_{iz}}{\mu_m}-\frac{\partial}{\partial z}\frac{B_{iy}}{\mu_m}\right)-\frac{B_{0x}+B_{ix}}{\mu_m\sigma}\left(\frac{\partial}{\partial x}\frac{B_{iy}}{\mu_m}-\frac{\partial}{\partial y}\frac{B_{ix}}{\mu_m}\right)\right] \\ \frac{1}{\sigma}\left(\frac{\partial}{\partial y}\frac{B_{ix}}{\mu_m}-\frac{\partial}{\partial x}\frac{B_{iy}}{\mu_m}\right) \\ 0 \\ \frac{1}{\sigma}\left(\frac{\partial}{\partial y}\frac{B_{iz}}{\mu_m}-\frac{\partial}{\partial z}\frac{B_{iy}}{\mu_m}\right) \end{pmatrix} \tag{B.64}$$

$$\mathbf{G}_d = \frac{1}{Re_\sigma}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ R_b\left[\frac{B_{0x}+B_{ix}}{\mu_m\sigma}\left(\frac{\partial}{\partial z}\frac{B_{ix}}{\mu_m}-\frac{\partial}{\partial x}\frac{B_{iz}}{\mu_m}\right)-\frac{B_{0y}+B_{iy}}{\mu_m\sigma}\left(\frac{\partial}{\partial y}\frac{B_{iz}}{\mu_m}-\frac{\partial}{\partial z}\frac{B_{iy}}{\mu_m}\right)\right] \\ \frac{1}{\sigma}\left(\frac{\partial}{\partial z}\frac{B_{ix}}{\mu_m}-\frac{\partial}{\partial x}\frac{B_{iz}}{\mu_m}\right) \\ \frac{1}{\sigma}\left(\frac{\partial}{\partial z}\frac{B_{iy}}{\mu_m}-\frac{\partial}{\partial y}\frac{B_{iz}}{\mu_m}\right) \\ 0 \end{pmatrix} \tag{B.65}$$

Viscous flux (NS equations)

$$\mathbf{E}_v = \frac{1}{Re}\begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx}+v\tau_{yx}+w\tau_{zx}+\frac{\mu}{Pr(\gamma-1)M^2}\frac{\partial T}{\partial x} \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{B.66}$$

$$\mathbf{F}_v = \frac{1}{Re}\begin{pmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{xy} + v\tau_{yy} + w\tau_{zy} + \frac{\mu}{Pr(\gamma-1)M^2}\frac{\partial T}{\partial y} \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{B.67}$$

$$\mathbf{G}_v = \frac{1}{Re}\begin{pmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u\tau_{xz} + v\tau_{yz} + w\tau_{zz} + \frac{\mu}{Pr(\gamma-1)M^2}\frac{\partial T}{\partial z} \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{B.68}$$

Magnetic source terms (ideal or full MHD equations)

$$\mathbf{S} = -\left(\nabla \cdot \mathbf{B}_i\right)\begin{pmatrix} 0 \\ R_b\frac{B_{0x}+B_{ix}}{\mu_m} \\ R_b\frac{B_{0y}+B_{iy}}{\mu_m} \\ R_b\frac{B_{0z}+B_{iz}}{\mu_m} \\ R_b\left(\mathbf{u}\cdot\frac{\mathbf{B}_i}{\mu_m}\right) \\ u \\ v \\ w \end{pmatrix} \tag{B.69}$$

B.6 MHD Eigenvalues

The eigenvalues of a set of flow governing equations define the speed at which the information waves travel. For the case of the Euler equations, these are given by Chung [25] as

$$\lambda = (U_n, U_n, U_n, U_n + c, U_n - c) \ . \tag{B.70}$$

The maximum eigenvalue can then be found to be $\lambda_{max} = |U_n| + c$.

However, the MHD equations are considerably more complex than the Euler equations because of the additional magnetic terms and induction equations. According to Powell *et al.* [133], the ideal MHD system of equations (B.54) has eight distinct eigenvalues – the entropy, magnetic-flux, Alfvèn (incompressible magnetic wave), fast and slow magneto-acoustic waves (compressible, magnetic field and pressure coupled):

$$\lambda = (U_n, U_n, U_n \pm c_a, U_n \pm c_f, U_n \pm c_s) \ , \tag{B.71}$$

where U_n is the normal fluid velocity, c_a is the Alfvèn wave speed given by

$$c_a = \frac{B_n}{\sqrt{\mu_m \rho}} \, , \tag{B.72}$$

and c_f and c_s are the fast and slow magneto-acoustic wave speeds, respectively, expressed as

$$c_{f,s} = \sqrt{\frac{1}{2}\left[c^2 + \frac{B^2}{\rho\mu_m} \pm \sqrt{\left(c^2 + \frac{B^2}{\rho\mu_m}\right)^2 - 4\frac{c^2 B_n^2}{\rho\mu_m}}\right]} \, . \tag{B.73}$$

The speed of sound is defined as $c = \sqrt{\frac{\gamma p}{\rho}}$, under perfect gas assumption.

The ideal MHD eigenvalues are illustrated in terms of characteristic waves in figure B.1. Therefore, if $U_n < c_f$, then some of the information propagates upstream of the flow direction. Otherwise, everything travels downstream.

The maximum eigenvalue corresponds to the fast magneto-acoustic wave,

$$\lambda_{max} = |U_n| + c_f \, . \tag{B.74}$$

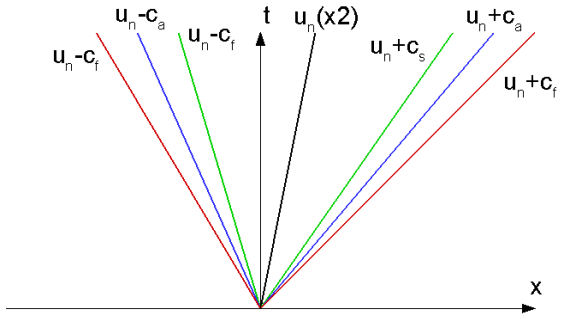


Figure B.1: Ideal MHD waves.

B.7 Flux Jacobians

Let $\boldsymbol{w}_i$ be the vector of conservation variables defined as

$$\boldsymbol{w}_i = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \end{pmatrix} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho Z_i \\ B_{ix} \\ B_{iy} \\ B_{iz} \end{pmatrix} . \tag{B.75}$$

To simplify the computations of the Jacobians that follow, it is convenient to pre-compute the derivative of energy, E, and pressure, p, with respect to the conservation vector $\boldsymbol{w}$. Expressing the energy E in terms of the conservation variables $\boldsymbol{w}_i$, recalling definition (B.29),

$$\rho E = \rho Z_i - \frac{1}{2\mu_m}\left(B_{ix}^2 + B_{iy}^2 + B_{iz}^2\right) = w_5 - \frac{1}{2\mu_m}\left(w_6^2 + w_7^2 + w_8^2\right) , \tag{B.76}$$

it then follows that

$$\begin{cases} \frac{\partial \rho E}{\partial w_1} = 0 \\ \frac{\partial \rho E}{\partial w_2} = 0 \\ \frac{\partial \rho E}{\partial w_3} = 0 \\ \frac{\partial \rho E}{\partial w_4} = 0 \end{cases} \qquad \begin{cases} \frac{\partial \rho E}{\partial w_5} = 1 \\ \frac{\partial \rho E}{\partial w_6} = -\frac{w_6}{\mu_m} = -\frac{B_{ix}}{\mu_m} \\ \frac{\partial \rho E}{\partial w_7} = -\frac{w_7}{\mu_m} = -\frac{B_{iy}}{\mu_m} \\ \frac{\partial \rho E}{\partial w_8} = -\frac{w_8}{\mu_m} = -\frac{B_{iz}}{\mu_m} \, . \end{cases} \tag{B.77}$$

The pressure p can be expressed in terms of the conservation variables $\boldsymbol{w}_i$, assuming perfect gas, as

$$\begin{array}{rcl} p & = & \rho\,(\gamma-1)\,C_v T = \rho\,(\gamma-1)\,e = \rho\,(\gamma-1)\left[E - \frac{1}{2}\,(u^2+v^2+w^2)\right] \\ & = & (\gamma-1)\left[w_5 - \frac{1}{2w_1}\,(w_2^2+w_3^2+w_4^2) - \frac{1}{2\mu_m}\,(w_6^2+w_7^2+w_8^2)\right] \end{array} \, , \tag{B.78}$$

leading to

$$\begin{cases} \frac{\partial p}{\partial w_1} = \frac{1}{2}\frac{(\gamma-1)(w_2^2+w_3^2+w_4^2)}{w_1^2} = \frac{1}{2}(\gamma-1)(u^2+v^2+w^2) \\ \frac{\partial p}{\partial w_2} = -\frac{(\gamma-1)w_2}{w_1} = -(\gamma-1)u \\ \frac{\partial p}{\partial w_3} = -\frac{(\gamma-1)w_3}{w_1} = -(\gamma-1)v \\ \frac{\partial p}{\partial w_4} = -\frac{(\gamma-1)w_4}{w_1} = -(\gamma-1)w \\ \frac{\partial p}{\partial w_5} = (\gamma-1) \\ \frac{\partial p}{\partial w_6} = -\frac{(\gamma-1)w_6}{\mu_m} = -\frac{(\gamma-1)B_{ix}}{\mu_m} \\ \frac{\partial p}{\partial w_7} = -\frac{(\gamma-1)w_7}{\mu_m} = -\frac{(\gamma-1)B_{iy}}{\mu_m} \\ \frac{\partial p}{\partial w_8} = -\frac{(\gamma-1)w_8}{\mu_m} = -\frac{(\gamma-1)B_{iz}}{\mu_m} \, . \end{cases} \tag{B.79}$$

Also, defining the magnetic pressure as

$$p_{mag} = \frac{1}{2\mu_m}\left(B_{ix}^2 + B_{iy}^2 + B_{iz}^2\right) = \frac{1}{2\mu_m}\left(w_6^2 + w_7^2 + w_8^2\right) , \tag{B.80}$$

results in

$$\begin{cases} \frac{\partial p_{mag}}{\partial w_1} = 0 \\ \\ \frac{\partial p_{mag}}{\partial w_2} = 0 \\ \\ \frac{\partial p_{mag}}{\partial w_3} = 0 \\ \\ \frac{\partial p_{mag}}{\partial w_4} = 0 \end{cases} \qquad \begin{cases} \frac{\partial p_{mag}}{\partial w_5} = 0 \\ \\ \frac{\partial p_{mag}}{\partial w_6} = \frac{w_6}{\mu_m} = \frac{B_{ix}}{\mu_m} \\ \\ \frac{\partial p_{mag}}{\partial w_7} = \frac{w_7}{\mu_m} = \frac{B_{iy}}{\mu_m} \\ \\ \frac{\partial p_{mag}}{\partial w_8} = \frac{w_8}{\mu_m} = \frac{B_{iz}}{\mu_m} \end{cases} . \tag{B.81}$$

Based on the definitions of the flux vectors in terms of the conservation variables, (B.57), (B.58) and (B.59), the flux Jacobians can be readily computed using the definition

$$\mathcal{A}_{ij} = \frac{\partial \mathcal{F}_i}{\partial w_j} . \tag{B.82}$$

Since in the present dissertation the full MHD equations were used in a simpler form, corresponding to the ideal MHD equations, only the inviscid and perfectly conducting magnetic flux Jacobians are evaluated in the following sections.

B.7.1 Flux Jacobian in the x-direction

The flux Jacobian in the x-direction is given by

$$\mathcal{A}_n^m = \frac{\partial \mathcal{F}^m(x)}{\partial w_n} = \frac{\partial E^m}{\partial w_n} \tag{B.83}$$

Using the decomposition (B.55), the total Jacobian can be given as

$$\mathcal{A}_n^m = \frac{\partial}{\partial w_n}\left(E_i^m + E_m^m + E_d^m - E_v^m\right) \tag{B.84}$$

Inviscid flux Jacobian (Euler or NS equations)

$$\frac{\partial E_i^m}{\partial w_n} = \frac{\partial}{\partial w_n}\begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (\rho E + p)u \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} w_2 \\ \frac{w_2^2}{w_1} + p \\ \frac{w_2 w_3}{w_1} \\ \frac{w_2 w_4}{w_1} \\ (\rho E + p)\frac{w_2}{w_1} \\ 0 \\ 0 \\ 0 \end{pmatrix} = \tag{B.85}$$

$$
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{w_2^2}{w_1^2}+\frac{\partial p}{\partial w_1} & \frac{2w_2}{w_1}+\frac{\partial p}{\partial w_2} & \frac{\partial p}{\partial w_3} & \frac{\partial p}{\partial w_4} & \frac{\partial p}{\partial w_5} & \frac{\partial p}{\partial w_6} & \frac{\partial p}{\partial w_7} & \frac{\partial p}{\partial w_8} \\
-\frac{w_2 w_3}{w_1^2} & \frac{w_3}{w_1} & \frac{w_2}{w_1} & 0 & 0 & 0 & 0 & 0 \\
-\frac{w_2 w_4}{w_1^2} & \frac{w_4}{w_1} & 0 & \frac{w_2}{w_1} & 0 & 0 & 0 & 0 \\
\frac{w_2^2}{w_1^2}\left(\frac{\partial p}{\partial w_1}-\rho E-p\right) & \frac{1}{w_1}\left(\frac{\partial p}{\partial w_2}+\rho E+p\right) & \frac{w_2}{w_1}\frac{\partial p}{\partial w_3} & \frac{w_2}{w_1}\frac{\partial p}{\partial w_4} & \frac{w_2}{w_1}\left[\frac{\partial(\rho E+p)}{\partial w_5}\right] & \frac{w_2}{w_1}\left[\frac{\partial(\rho E+p)}{\partial w_6}\right] & \frac{w_2}{w_1}\left[\frac{\partial(\rho E+p)}{\partial w_7}\right] & \frac{w_2}{w_1}\left[\frac{\partial(\rho E+p)}{\partial w_8}\right] \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u^2+\frac{1}{2}(\gamma-1)U^2 & (3-\gamma)u & (1-\gamma)v & (1-\gamma)w & \gamma-1 & \frac{(1-\gamma)B_{ix}}{\mu_m} & \frac{(1-\gamma)B_{iy}}{\mu_m} & \frac{(1-\gamma)B_{iz}}{\mu_m} \\
-uv & v & u & 0 & 0 & 0 & 0 & 0 \\
-uw & w & 0 & u & 0 & 0 & 0 & 0 \\
-\left(E+\frac{p}{\rho}\right)u+ & E+\frac{p}{\rho}+ & (1-\gamma)uv & (1-\gamma)uw & \gamma u & -\frac{\gamma B_{ix}u}{\mu u_m} & -\frac{\gamma B_{iy}u}{\mu u_m} & -\frac{\gamma B_{iz}u}{\mu u_m} \\
+\frac{1}{2}(\gamma-1)uU^2 & +(1-\gamma)u^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Magnetic flux Jacobian (ideal or full MHD)

$$\frac{\partial E_m^m}{\partial w_n} = \frac{\partial}{\partial w_n} \begin{pmatrix} 0 \\ \frac{R_b}{\mu_m}\left(\frac{\mathbf{B}_i\cdot\mathbf{B}_i}{2} - B_{ix}^2 + \mathbf{B_0}\cdot\mathbf{B_i} - B_{0x}B_{ix} - B_{ix}B_{0x}\right) \\ -\frac{R_b}{\mu_m}\left(B_{ix}B_{iy} + B_{0x}B_{iy} + B_{ix}B_{0y}\right) \\ -\frac{R_b}{\mu_m}\left(B_{ix}B_{iz} + B_{0x}B_{iz} + B_{ix}B_{0z}\right) \\ \frac{R_b}{\mu_m}\left(\mathbf{B}_i\cdot\mathbf{B}_i u - B_{ix}\mathbf{u}\cdot\mathbf{B}_i + \mathbf{B}_0\cdot\mathbf{B}_i u - B_{0x}\mathbf{u}\cdot\mathbf{B}_i\right) \\ 0 \\ (uB_{iy} - vB_{ix}) + (uB_{0y} - vB_{0x}) \\ (uB_{iz} - wB_{ix}) + (uB_{0z} - wB_{0x}) \end{pmatrix} =$$

$$= \begin{pmatrix} 0 \\ \frac{R_b}{\mu_m}\left(-\frac{1}{2}w_6^2 + \frac{1}{2}w_7^2 + \frac{1}{2}w_8^2 - B_{0x}w_6 + B_{0y}w_7 + B_{0z}w_8\right) \\ -\frac{R_b}{\mu_m}\left(w_6w_7 + B_{0x}w_7 + w_6B_{0y}\right) \\ -\frac{R_b}{\mu_m}\left(w_6w_8 + B_{0x}w_8 + w_6B_{0z}\right) \\ \frac{R_b}{\mu_m w_1}\left[(w_6^2 + w_7^2 + w_8^2)w_2 - w_6(w_2w_6 + w_3w_7 + w_4w_8) + \right. \\ \left. +(B_{0x}w_6 + B_{0y}w_7 + B_{0z}w_8)w_2 - B_{0x}(w_2w_6 + w_3w_7 + w_4w_8)\right] \\ 0 \\ \frac{1}{w_1}\left(w_2w_7 - w_3w_6 + w_2B_{0y} - w_3B_{0x}\right) \\ \frac{1}{w_1}\left(w_2w_8 - w_4w_6 + w_2B_{0z} - w_4B_{0x}\right) \end{pmatrix} = \tag{B.86}$$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
=
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
=
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

B.7.2 Flux Jacobian in the y-direction

The flux Jacobian in the y-direction is given by

$$\mathcal{B}_n^m = \frac{\partial \mathcal{F}^m(y)}{\partial w_n} = \frac{\partial F^m}{\partial w_n} \tag{B.87}$$

Using the decomposition (B.55), the total Jacobian can be given as

$$\mathcal{B}_n^m = \frac{\partial}{\partial w_n}\left(F_i^m + F_m^m + F_d^m - F_v^m\right) \tag{B.88}$$

Inviscid flux Jacobian (Euler or NS equations)

$$\frac{\partial F_i^m}{\partial w_n} = \frac{\partial}{\partial w_n}\begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ (\rho E + p)v \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} w_3 \\ \frac{w_2 w_3}{w_1} \\ \frac{w_3^2}{w_1} + p \\ \frac{w_3 w_4}{w_1} \\ (\rho E + p)\frac{w_3}{w_1} \\ 0 \\ 0 \\ 0 \end{pmatrix} = \tag{B.89}$$

$$
\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
-\frac{w_2 w_3}{w_1^2} & \frac{w_3}{w_1} & \frac{w_2}{w_1} & 0 & 0 & 0 & 0 & 0 \\
-\frac{w_3^2}{w_1^2}+\frac{\partial p}{w_1} & \frac{\partial p}{w_2} & \frac{2w_3}{w_1}+\frac{\partial p}{w_3} & \frac{\partial p}{w_4} & \frac{\partial p}{w_5} & \frac{\partial p}{w_6} & \frac{\partial p}{w_7} & \frac{\partial p}{w_8} \\
-\frac{w_3 w_4}{w_1^2} & 0 & \frac{w_4}{w_1} & \frac{w_3}{w_1} & 0 & 0 & 0 & 0 \\
\frac{w_3}{w_1^2}\left(\frac{\partial p}{\partial w_1}w_1-\rho E-p\right) & \frac{w_3}{w_1}\frac{\partial p}{\partial w_2} & \frac{1}{w_1}\left(\frac{\partial p}{\partial w_3}w_3+\rho E+p\right) & \frac{w_3}{w_1}\frac{\partial p}{\partial w_4} & \frac{w_3}{w_1}\left[\frac{\partial(\rho E+p)}{\partial w_5}\right] & \frac{w_3}{w_1}\left[\frac{\partial(\rho E+p)}{\partial w_6}\right] & \frac{w_3}{w_1}\left[\frac{\partial(\rho E+p)}{\partial w_7}\right] & \frac{w_3}{w_1}\left[\frac{\partial(\rho E+p)}{\partial w_8}\right] \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
-uv & v & u & 0 & 0 & 0 & 0 & 0 \\
-v^2+\frac{1}{2}(\gamma-1)U^2 & (1-\gamma)u & (3-\gamma)v & (1-\gamma)w & \gamma-1 & \frac{(1-\gamma)B_{ix}}{\mu_m} & \frac{(1-\gamma)B_{iy}}{\mu_m} & \frac{(1-\gamma)B_{iz}}{\mu_m} \\
-vw & 0 & w & v & 0 & 0 & 0 & 0 \\
-v(\frac{E+p}{\rho}-\frac{1}{2}(\gamma-1)vU^2 & (1-\gamma)uv & \frac{E+p}{\rho}+(1-\gamma)v^2 & (1-\gamma)vw & \gamma v & -\frac{\gamma B_{ix}v}{m u_m} & -\frac{\gamma B_{iy}v}{m u_m} & -\frac{\gamma B_{iz}v}{m u_m} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Magnetic flux Jacobian (ideal or full MHD)

$$\frac{\partial F_m^m}{\partial w_n} = \frac{\partial}{\partial w_n} \begin{pmatrix} 0 \\ -\frac{R_b}{\mu_m}\left(B_{iy}B_{ix} + B_{0y}B_{ix} + B_{iy}B_{0x}\right) \\ \frac{R_b}{\mu_m}\left(\frac{\mathbf{B}_i\cdot\mathbf{B}_i}{2} - B_{iy}^2 + \mathbf{B_0}\cdot\mathbf{B_i} - B_{0y}B_{iy} - B_{iy}B_{0y}\right) \\ -\frac{R_b}{\mu_m}\left(B_{iy}B_{iz} + B_{0y}B_{iz} + B_{iy}B_{0z}\right) \\ \frac{R_b}{\mu_m}\left(\mathbf{B}_i\cdot\mathbf{B}_i v - B_{iy}\mathbf{u}\cdot\mathbf{B}_i + \mathbf{B}_0\cdot\mathbf{B}_i v - B_{0y}\mathbf{u}\cdot\mathbf{B}_i\right) \\ (vB_{ix} - uB_{iy}) + (vB_{0x} - uB_{0y}) \\ 0 \\ (vB_{iz} - wB_{iy}) + (vB_{0z} - wB_{0y}) \end{pmatrix} =$$

$$= \begin{pmatrix} 0 \\ -\frac{R_b}{\mu_m}\left(w_6w_7 + B_{0x}w_7 + w_6B_{0y}\right) \\ \frac{R_b}{\mu_m}\left(\frac{1}{2}w_6^2 - \frac{1}{2}w_7^2 + \frac{1}{2}w_8^2 + B_{0x}w_6 - B_{0y}w_7 + B_{0z}w_8\right) \\ -\frac{R_b}{\mu_m}\left(w_7w_8 + B_{0y}w_8 + w_7B_{0z}\right) \\ \frac{R_b}{\mu_m w_1}\left[(w_6^2 + w_7^2 + w_8^2)w_3 - w_7(w_2w_6 + w_3w_7 + w_4w_8) + \right. \\ \left. +(B_{0x}w_6 + B_{0y}w_7 + B_{0z}w_8)w_3 - B_{0y}(w_2w_6 + w_3w_7 + w_4w_8)\right] \\ \frac{1}{w_1}\left(w_3w_6 - w_2w_7 + w_3B_{0x} - w_2B_{0y}\right) \\ 0 \\ \frac{1}{w_1}\left(w_3w_8 - w_4w_7 + w_3B_{0z} - w_4B_{0y}\right) \end{pmatrix} = \tag{B.90}$$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{R_b}{\mu_m}(w_7+B_{0y}) & 0 & -\frac{R_b}{\mu_m}(w_6+B_{0x}) & 0 \\
0 & 0 & 0 & 0 & \frac{R_b}{\mu_m}(w_6+B_{0x}) & 0 & -\frac{R_b}{\mu_m}(w_7+B_{0y}) & \frac{R_b}{\mu_m}(w_8+B_{0z}) \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{R_b}{\mu_m}(w_8+B_{0z}) & -\frac{R_b}{\mu_m}(w_7+B_{0y}) \\
-\frac{R_b}{\mu_m}\frac{w_3}{w_1^2}\left[w_3(w_6^2+w_8^2)-\right. & -\frac{R_b w_6(w_7+B_{0y})}{\mu_m w_1} & \frac{R_b}{\mu_m}\left(w_6^2+w_8^2+\right. & -\frac{R_b w_8(w_7+B_{0y})}{\mu_m w_1} & 0 & \frac{R_b}{\mu_m w_1}\left[w_3(2w_6+B_{0x})-\right. & -\frac{R_b(w_2w_6+w_4w_8)}{\mu_m w_1} & \frac{R_b}{\mu_m w_1}\left[w_3(2w_8+B_{0z})-\right. \\
-w_7(w_2w_6+w_4w_8)+ & +B_{0x}w_6+B_{0z}w_8) & & & -w_2(w_7+B_{0y})] & & & -w_4(w_7+B_{0y})] \\
+w_3(B_{0x}w_6+B_{0z}w_8)- & & & & & & & \\
-B_{0y}(w_2w_6+w_4w_8)] & & & & & & & \\
\frac{w_2(w_7+B_{0y})-w_3(w_6+B_{0x})}{w_1^2} & -\frac{w_7+B_{0y}}{w_1} & \frac{w_6+B_{0x}}{w_1} & 0 & 0 & -\frac{w_2}{w_1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{w_3(w_8+B_{0z})-w_4(w_7+B_{0y})}{w_1^2} & 0 & \frac{w_8+B_{0z}}{w_1} & -\frac{w_7+B_{0y}}{w_1} & 0 & -\frac{w_4}{w_1} & 0 & \frac{w_3}{w_1}
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -\frac{R_b}{\mu_m}(B_{iy}+B_{0y}) & -\frac{R_b}{\mu_m}(B_{ix}+B_{0x}) & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{R_b}{\mu_m}(B_{ix}+B_{0x}) & -\frac{R_b}{\mu_m}(B_{iy}+B_{0y}) & \frac{R_b}{\mu_m}(B_{iz}+B_{0z}) \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{R_b}{\mu_m}(B_{iz}+B_{0z}) & -\frac{R_b}{\mu_m}(B_{iy}+B_{0y}) \\
-\frac{R_b}{\mu_m\rho}\left[v(B_{ix}^2+B_{iz}^2)-\right. & -\frac{R_b B_{ix}(B_{iy}+B_{0y})}{\mu_m\rho} & \frac{R_b}{\mu_m\rho}\left(B_{ix}^2+B_{iz}^2+\right. & -\frac{R_b B_{iz}(B_{iy}+B_{0y})}{\mu_m\rho} & 0 & \frac{R_b}{\mu_m}\left[v(2B_{ix}+B_{0x})-\right. & -\frac{R_b(uB_{ix}+wB_{iz})}{\mu_m} & \frac{R_b}{\mu_m}\left[v(2B_{iz}+B_{0z})-\right. \\
-uB_{iy}B_{ix}+wB_{iz}B_{iy})+ & & +B_{0x}B_{ix}+B_{0z}B_{iz}) & & -u(B_{iy}+B_{0y})] & & & -w(B_{iy}+B_{0y})] \\
+v(B_{0x}B_{ix}+B_{0z}B_{iz})- & & & & & & & \\
-B_{0y}(uB_{ix}+wB_{iz})] & & & & & & & \\
\frac{u(B_{iy}+B_{0y})-v(B_{ix}+B_{0x})}{\rho} & -\frac{B_{iy}+B_{0y}}{\rho} & \frac{B_{ix}+B_{0x}}{\rho} & 0 & 0 & v & -u & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{v(B_{iz}+B_{0z})-w(B_{iy}+B_{0y})}{\rho^2} & 0 & \frac{B_{iz}+B_{0z}}{\rho} & -\frac{B_{iy}+B_{0y}}{\rho} & 0 & 0 & -w & v
\end{bmatrix}
$$

B.7.3 Flux Jacobian in the z-direction

The flux Jacobian in the z-direction is given by

$$\mathcal{C}_n^m = \frac{\partial \mathcal{F}^m(z)}{\partial w_n} = \frac{\partial G^m}{\partial w_n} \tag{B.91}$$

Using the decomposition (B.55), the total Jacobian can be given as

$$\mathcal{C}_n^m = \frac{\partial}{\partial w_n}\left(G_i^m + G_m^m + G_d^m - G_v^m\right) \tag{B.92}$$

Inviscid flux Jacobian (Euler or NS equations)

$$\frac{\partial G_i^m}{\partial w_n} = \frac{\partial}{\partial w_n}\begin{pmatrix} \rho w \\ \rho u w \\ \rho v w \\ \rho w^2 + p \\ (\rho E + p) w \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} w_4 \\ \frac{w_2 w_4}{w_1} \\ \frac{w_3 w_4}{w_1} \\ \frac{w_4^2}{w_1} + p \\ (\rho E + p)\frac{w_4}{w_1} \\ 0 \\ 0 \\ 0 \end{pmatrix} = \tag{B.93}$$

$$
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
-\frac{w_2 w_4}{w_1^2} & \frac{w_4}{w_1} & 0 & \frac{w_2}{w_1} & 0 & 0 & 0 & 0 \\
-\frac{w_3 w_4}{w_1^2} & 0 & \frac{w_4}{w_1} & \frac{w_3}{w_1} & 0 & 0 & 0 & 0 \\
-\frac{w_4^2}{w_1^2}+\frac{\partial p}{\partial w_1} & \frac{\partial p}{w_2} & \frac{\partial p}{w_3} & \frac{2w_4}{w_1}+\frac{\partial p}{w_4} & \frac{\partial p}{w_5} & \frac{\partial p}{w_6} & \frac{\partial p}{w_7} & \frac{\partial p}{w_8} \\
\frac{w_4}{w_1^2}\left(\frac{\partial p}{\partial w_1}w_1-\rho E-p\right) & \frac{w_4}{w_1}\frac{\partial p}{\partial w_2} & \frac{w_4}{w_1}\frac{\partial p}{\partial w_3} & \frac{1}{w_1}\left(\frac{\partial p}{\partial w_4}w_4+\rho E+p\right) & \frac{w_4}{w_1}\left[\frac{\partial(\rho E+p)}{\partial w_5}\right] & \frac{w_4}{w_1}\left[\frac{\partial(\rho E+p)}{\partial w_6}\right] & \frac{w_4}{w_1}\left[\frac{\partial(\rho E+p)}{\partial w_7}\right] & \frac{w_4}{w_1}\left[\frac{\partial(\rho E+p)}{\partial w_8}\right] \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-uu & u & 0 & 0 & 0 & 0 & 0 & 0 \\
-vu & v & 0 & 0 & 0 & 0 & 0 & 0 \\
-w^2+\frac{1}{2}(\gamma-1)U^2 & (1-\gamma)u & (1-\gamma)w & (1-\gamma)v & (3-\gamma)w & \gamma-1 & \frac{(1-\gamma)B_{ix}}{\mu_m} & \frac{(1-\gamma)B_{iy}}{\mu_m} & \frac{(1-\gamma)B_{iz}}{\mu_m} \\
-\frac{1}{2}(\gamma-1)wU^2+\left(E+\frac{p}{\rho}\right)w & 0 & 0 & 0 & E+\frac{p}{\rho}+(1-\gamma)w^2 & \gamma w & -\frac{\gamma B_{ix}w}{\mu_m u_m} & -\frac{\gamma B_{iy}w}{\mu_m u_m} & -\frac{\gamma B_{iz}w}{\mu_m u_m} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Magnetic flux Jacobian (ideal or full MHD)

$$\frac{\partial G_m^m}{\partial w_n} = \frac{\partial}{\partial w_n} \begin{pmatrix} 0 \\ -\frac{R_b}{\mu_m}\left(B_{iz}B_{ix} + B_{0z}B_{ix} + B_{iz}B_{0x}\right) \\ -\frac{R_b}{\mu_m}\left(B_{iz}B_{iy} + B_{0z}B_{iy} + B_{iz}B_{0y}\right) \\ \frac{R_b}{\mu_m}\left(\frac{\mathbf{B}_i\cdot\mathbf{B}_i}{2} - B_{iz}^2 + \mathbf{B_0}\cdot\mathbf{B_i} - B_{0z}B_{iz} - B_{iz}B_{0z}\right) \\ \frac{R_b}{\mu_m}\left(\mathbf{B}_i\cdot\mathbf{B}_i w - B_{iz}\mathbf{u}\cdot\mathbf{B}_i + \mathbf{B}_0\cdot\mathbf{B}_i w - B_{0z}\mathbf{u}\cdot\mathbf{B}_i\right) \\ (wB_{ix} - uB_{iz}) + (wB_{0x} - uB_{0z}) \\ (wB_{iy} - vB_{iz}) + (wB_{0y} - vB_{0z}) \\ 0 \end{pmatrix} =$$

$$= \begin{pmatrix} 0 \\ -\frac{R_b}{\mu_m}\left(w_6w_8 + B_{0x}w_8 + w_6B_{0z}\right) \\ -\frac{R_b}{\mu_m}\left(w_7w_8 + B_{0y}w_8 + w_7B_{0z}\right) \\ \frac{R_b}{\mu_m}\left(\frac{1}{2}w_6^2 + \frac{1}{2}w_7^2 - \frac{1}{2}w_8^2 + B_{0x}w_6 + B_{0y}w_7 - B_{0z}w_8\right) \\ \frac{R_b}{\mu_m w_1}\left[(w_6^2 + w_7^2 + w_8^2)w_4 - w_8(w_2w_6 + w_3w_7 + w_4w_8) + \right. \\ \left. +(B_{0x}w_6 + B_{0y}w_7 + B_{0z}w_8)w_4 - B_{0z}(w_2w_6 + w_3w_7 + w_4w_8)\right] \\ \frac{1}{w_1}\left(w_4w_6 - w_2w_8 + w_4B_{0x} - w_2B_{0z}\right) \\ \frac{1}{w_1}\left(w_4w_7 - w_3w_8 + w_4B_{0y} - w_3B_{0z}\right) \\ 0 \end{pmatrix} = \tag{B.94}$$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{R_b}{\mu_m}(w_8+B_{0z}) & 0 & 0 & -\frac{R_b}{\mu_m}(w_6+B_{0x}) \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{R_b}{\mu_m}(w_8+B_{0z}) & -\frac{R_b}{\mu_m}(w_7+B_{0y}) \\
0 & 0 & 0 & 0 & 0 & \frac{R_b}{\mu_m}(w_6+B_{0x}) & \frac{R_b}{\mu_m}(w_7+B_{0y}) & -\frac{R_b}{\mu_m}(w_8+B_{0z}) \\
0 & 0 & 0 & 0 & 0 & \frac{R_b}{\mu_m w_1}[w_4(2w_6+B_{0x})- & \frac{R_b}{\mu_m w_1}[w_4(2w_7+B_{0y})- & -\frac{R_b(w_2w_6+w_3w_7)}{\mu_m w_1} \\
 & & & & & -w_2(w_8+B_{0z})] & -w_3(w_8+B_{0z})] & \\
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{w_2}{w_1} \\
0 & 0 & 0 & 0 & 0 & \frac{w_4}{w_1} & \frac{w_4}{w_1} & -\frac{w_3}{w_1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
$$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{R_b}{\mu_m}(B_{ix}+B_{0x}) \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{R_b}{\mu_m}(B_{iz}+B_{0z}) & -\frac{R_b}{\mu_m}(B_{iy}+B_{0y}) \\
0 & 0 & 0 & 0 & 0 & \frac{R_b}{\mu_m}(B_{ix}+B_{0x}) & \frac{R_b}{\mu_m}(B_{iy}+B_{0y}) & -\frac{R_b}{\mu_m}(B_{iz}+B_{0z}) \\
0 & 0 & 0 & 0 & 0 & \frac{R_b}{\mu_m}[w(2B_{ix}+B_{0x})- & \frac{R_b}{\mu_m}[w(2B_{iy}+B_{0y})- & -\frac{R_b(uB_{ix}+vB_{iy})}{\mu_m} \\
 & & & & & -u(B_{iz}+B_{0z})] & -v(B_{iz}+B_{0z})] & \\
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{R_b}{\mu_m}(B_{ix}+B_{0x}) \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{R_b}{\mu_m}(B_{iz}+B_{0z}) & -\frac{R_b}{\mu_m}(B_{iy}+B_{0y}) \\
0 & 0 & 0 & 0 & 0 & \frac{R_b}{\mu_m}(B_{ix}+B_{0x}) & \frac{R_b}{\mu_m}(B_{iy}+B_{0y}) & -\frac{R_b}{\mu_m}(B_{iz}+B_{0z}) \\
0 & 0 & 0 & 0 & 0 & w & 0 & -u \\
0 & 0 & 0 & 0 & 0 & 0 & w & -v \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
$$

Appendix C

Generalized coordinate transformation

The equations that govern fluid flow are typically derived in physical space, which result from the application of fundamental laws, such as conservation of mass, momentum and energy. However, if one wishes to solve them numerically, it is often desired to express those equations in computational space, in particular when using finite-difference formulations, to enhance the efficiency and accuracy of the numerical schemes, and to simplify the implementation of the boundary conditions.

A brief description of the mathematics necessary to accomplish that transformation is described in this chapter, where further details and application to specific equations were described by Hoffmann [70].

C.1 Governing equations in physical space

The three-dimensional equations of motion can be written in flux vector form, expressed in Cartesian coordinates, as

$$\frac{\partial \boldsymbol{w}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} = \mathbf{S}\,, \tag{C.1}$$

where $\boldsymbol{w}$ is the vector of conservative variables, and $\mathbf{E}$, $\mathbf{F}$ and $\mathbf{G}$ are the fluxes (inviscid, viscous, magnetic, artificial dissipation,...) in the $x-$, $y-$ and $z-$directions, respectively, and $\mathbf{S}$ represents the source terms.

C.2 Coordinate transformation

A coordinate transformation of the governing equations expressed in a Cartesian coordinate system $x_i = (x, y, z)$ from physical space to computational space $\xi_j = (\xi, \eta, \zeta)$ can be generically expressed in the form

$$\begin{cases} \xi = \xi(x, y, z) \\ \eta = \eta(x, y, z) \\ \zeta = \zeta(x, y, z) \end{cases} . \tag{C.2}$$

This transformation defines the metrics

$$K_{ij} = \left[\frac{\partial x_i}{\partial \xi_j}\right], \qquad K_{ij}^{-1} = \left[\frac{\partial \xi_i}{\partial x_j}\right], \tag{C.3}$$

$$J = \det(K^{-1}) \quad \text{and} \qquad S_{ij} = \frac{1}{J} K_{ij}^{-1}, \tag{C.4}$$

where S_{ij} represents the area of the face of a cell in the ξ_i direction, projected onto each physical coordinate direction x_j. The Jacobian of the transformation J C.4 can be expanded to

$$J = \frac{1}{x_\xi (y_\eta z_\zeta - y_\zeta z_\eta) - x_\eta (y_\xi z_\zeta - y_\zeta z_\xi) + x_\zeta (y_\xi z_\eta - y_\eta z_\xi)}, \tag{C.5}$$

where the metrics $x_\xi, x_\eta, x_\zeta, y_\xi, y_\eta, y_\zeta, z_\xi, z_\eta$, and z_ζ are easily computed numerically by some finite-difference approximations since the step sizes are equally spaced in the computational domain.

C.3 Governing equations in computational space

The gradients in physical space can be expressed in terms of gradients in computational space by applying the chain rule of differentiation to the inverse of the transformation C.2. These can then be written as

$$\left\{\begin{array}{l} \frac{\partial}{\partial x} = \frac{\partial}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial}{\partial \zeta}\frac{\partial \zeta}{\partial x} = \xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta} + \zeta_x \frac{\partial}{\partial \zeta} \\ \frac{\partial}{\partial y} = \frac{\partial}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial}{\partial \eta}\frac{\partial \eta}{\partial y} + \frac{\partial}{\partial \zeta}\frac{\partial \zeta}{\partial y} = \xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta} + \zeta_y \frac{\partial}{\partial \zeta} \\ \frac{\partial}{\partial z} = \frac{\partial}{\partial \xi}\frac{\partial \xi}{\partial z} + \frac{\partial}{\partial \eta}\frac{\partial \eta}{\partial z} + \frac{\partial}{\partial \zeta}\frac{\partial \zeta}{\partial z} = \xi_z \frac{\partial}{\partial \xi} + \eta_z \frac{\partial}{\partial \eta} + \zeta_z \frac{\partial}{\partial \zeta} \end{array}\right. . \tag{C.6}$$

Applying the relationships C.6 to the governing equations C.1 yields

$$\frac{\partial \boldsymbol{w}}{\partial t} + \xi_x \frac{\partial \mathbf{E}}{\partial \xi} + \eta_x \frac{\partial \mathbf{E}}{\partial \eta} + \zeta_x \frac{\partial \mathbf{E}}{\partial \zeta} \tag{C.7}$$

$$+ \xi_y \frac{\partial \mathbf{F}}{\partial \xi} + \eta_y \frac{\partial \mathbf{F}}{\partial \eta} + \zeta_y \frac{\partial \mathbf{F}}{\partial \zeta} \tag{C.8}$$

$$+ \xi_z \frac{\partial \mathbf{G}}{\partial \xi} + \eta_z \frac{\partial \mathbf{G}}{\partial \eta} + \zeta_z \frac{\partial \mathbf{G}}{\partial \zeta} = \mathbf{S}\,, \tag{C.9}$$

which, upon division by the transformation Jacobian J (assumed here invariant with t), and some algebraic rearrangements, can be written as

$$\frac{\partial}{\partial t}\left(\frac{\boldsymbol{w}}{J}\right) + \frac{\partial}{\partial \xi}\left[\frac{1}{J}\left(\xi_x \mathbf{E} + \xi_y \mathbf{F} + \xi_z \mathbf{G}\right)\right] \tag{C.10}$$

$$+ \frac{\partial}{\partial \eta}\left[\frac{1}{J}\left(\eta_x \mathbf{E} + \eta_y \mathbf{F} + \eta_z \mathbf{G}\right)\right] \tag{C.11}$$

$$+ \frac{\partial}{\partial \zeta}\left[\frac{1}{J}\left(\zeta_x \mathbf{E} + \zeta_y \mathbf{F} + \zeta_z \mathbf{G}\right)\right] = \frac{1}{J}\mathbf{S}\,. \tag{C.12}$$

The governing equations can then be expressed in computational coordinates as

$$\frac{\partial \bar{\boldsymbol{w}}}{\partial t} + \frac{\partial \bar{\mathbf{E}}}{\partial \xi} + \frac{\partial \bar{\mathbf{F}}}{\partial \eta} + \frac{\partial \bar{\mathbf{G}}}{\partial \zeta} = \bar{\mathbf{S}}\,, \tag{C.13}$$

where the state, the fluxes vectors, and the source terms are given in computational

coordinates by

$$\begin{cases} \bar{\boldsymbol{w}} = \frac{1}{J}\boldsymbol{w} \\ \bar{\mathbf{E}} = \frac{1}{J}\left(\xi_x\mathbf{E} + \xi_y\mathbf{F} + \xi_z\mathbf{G}\right) \\ \bar{\mathbf{F}} = \frac{1}{J}\left(\eta_x\mathbf{E} + \eta_y\mathbf{F} + \eta_z\mathbf{G}\right) \\ \bar{\mathbf{G}} = \frac{1}{J}\left(\zeta_x\mathbf{E} + \zeta_y\mathbf{F} + \zeta_z\mathbf{G}\right) \\ \bar{\mathbf{S}} = \frac{1}{J}\mathbf{S} \end{cases} . \tag{C.14}$$

Bibliography

[1] Aircraft Aerodynamics and Design Group. Standard atmosphere computations. `http://aero.stanford.edu/StdAtm.html`, 2007.

[2] Ramesh K. Agarwal and Justin Augustinus. Numerical simulation of compressible viscous MHD flows for reducing supersonic drag of blunt bodies. *AIAA Paper* 1999-0601, January 1999. Proceedings of the 37^{th} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[3] AHI. Australian Hypersonics Initiative. `http://www.spacedaily.com/news/rocketscience-04p.html`. Accessed in November of 2006.

[4] Mehmet A. Akgun, Raphael T. Haftka, K. C. Wu, and Joanne L. Walsh. Sensitivity of lumped constraints using the adjoint method. *AIAA Paper* 1999-1314, April 1999. Proceedings of the 40^{th} AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, St. Louis, MO.

[5] Juan J. Alonso. *Parallel Computation of Unsteady and Aeroelastic Flows Using an Implicit Mulrigrid-Driven Algorithm*. Ph.D. Dissertation, Princeton University, June 1997.

[6] John D. Anderson. *Fundamentals of Aerodynamics*. Series in Aeronautical and Aerospace Engineering. Mcgraw-Hill, 4^{th} edition, 2005.

[7] W. Kyle Anderson, James C. Newman, David L. Whitfield, and Eric J. Nielsen. Sensitivity analysis for Navier-Stokes equations on unstructured meshes using complex variables. *AIAA Journal*, 39(1):56–63, January 2001.

[8] ASC. Advanced simulation and computing. `http://www.llnl.gov/asc`. Accessed in March of 2007.

[9] Justin Augustinus, Klaus A. Hoffmann, and Shigeki Harada. Numerical solutions of ideal MHD equations for a symmetric blunt body at hypersonic speeds. *AIAA Paper* 1998-0850, January 1998. Proceedings of the 36^{th} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[10] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.3.0, Argonne National Laboratory, 2004.

[11] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc: Web page. `http://www.mcs.anl.gov/petsc`. Accessed in April of 2006.

[12] M. Bartholomew-Biggs. OPFAD - a users guide to the OPtima forward automatic differentiation tool. Technical report, Numerical Optimisation Centre, University of Hertfordshire, England, 1994.

[13] C. Bendtsen and O. Stauning. FADBAD, a flexible C++ package for automatic differentiation using the forward and backward methods. Technical Report Technical Report IMM-REP-1996-17, Technical University of Denmark, DK-2800 Lyngby, Denmark, 1996.

[14] M. Berz. DAFOR: Differential algebraic extension of Fortran. `http://bt.pa.msu.edu/index_cosy.htm`. Accessed in April of 2007.

[15] M. Berz. The differential algebra precompiler version 3 reference manual. Technical Report MSUCL-755, Michigan State University, East Lansing, MI 48824, 1991.

[16] C. H. Bischof, L. Roh, and A. J. Mauer-Oats. ADIC: an extensible automatic differentiation tool for ANSI-C. *Software Practice and Experience*, 27(12):1427–1456, 1997. `http://www-new.mcs.anl.gov/adic/`.

[17] Christian Bischof, Alan Carle, Paul Hovland, Peyvand Khademi, and Andrew Mauer. ADIFOR 2.0 user's guide (revision D). Technical Report Technical Memorandum ANL/MCS-TM-192, Argonne National Laboratory, 1998.

[18] Christian Bischof, Alan Carle, Peyvand Khademi, and Andrew Mauer. The ADIFOR 2.0 system for the automated differentiation of Fortran 77 programs. Technical Report Technical Report ANL-MCS-P481-1194, Argonne National Laboratory, 1998.

[19] J. U. Brackbill and D. C. Barnes. The effect of nonzero $\nabla \cdot \mathbf{B}$ on the numerical solution of the magnetohydrodynamic equations. *Journal of Computational Physics*, 35(3):426–430, May 1980.

[20] M. Brio and C. C. Wu. An upwind differencing scheme for the equations of ideal magnetohydrodynamics. *Journal of Computational Physics*, 75(2):400–422, April 1988.

[21] Alan Carle and Mike Fagan. ADIFOR 3.0 overview. Technical Report CAAM-TR-00-02, Rice University, 2000.

[22] Mark H. Carpenter, David Gottlieb, and Saul Abarbanel. Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: Methodology and application to high-order compact schemes. *Journal of Computational Physics*, 111(2):220–236, April 1994.

[23] Mark H. Carpenter, Jan Nordström, and David Gottlieb. A stable and conservative interface treatment of arbitrary spatial accuracy. *Journal of Computational Physics*, 148(2):341–365, January 1999.

[24] Australia Centre for Hypersonics, University of Queensland. HyShot. `http://www.uq.edu.au/hypersonics/index.html?page=19501`. Accessed in November of 2006.

[25] T. J. Chung. *Computational Fluid Dynamics*. Cambridge University Press, 2002.

[26] James F. Coakley and Robert W. Porter. Time-dependent numerical analysis of MHD blunt body problem. *AIAA Journal*, 9(8):1624–1626, August 1971.

[27] Kenneth R. Cramer. *Magnetofluid dynamics for engineers and applied physicists*. Washington, Scripta Pub. Co., 1973.

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

[29] Henri-Marie Damevin and Klaus A. Hoffmann. Numerical magnetogasdynamics simulations of hypersonic, chemically reacting flows. *AIAA Paper* 2001-2746, June 2001. Proceedings of the 32^{nd} AIAA Plasmadynamics and Lasers Conference, Anaheim, CA.

[30] DARPA. Falcon program, Tactical Technology Office. `http://www.darpa.mil/tto/programs/falcon.htm`. Accessed in October of 2006.

[31] Prasanta Deb and Ramesh K. Agarwal. A performance study of MHD-bypass scramjet inlets with chemical nonequilibrium. *AIAA Paper* 2001-2872, June 2001. Proceedings of the 32^{nd} AIAA Plasmadynamics and Lasers Conference, Anaheim, CA.

[32] Alain Dervieux, Laurent Hascoët, Valérie Pascual, Bruno Koobus, and Mariano Vazquez. TAPENADE: web page. `http://www-sop.inria.fr/tropics/tapenade.html`. Accessed in December of 2005.

[33] Jean-Francois Dietiker and Klaus A. Hoffmann. Numerical simulation of turbulent magnetohydrodynamic flows. *AIAA Paper* 2001-2737, June 2001. Proceedings of the 32^{nd} AIAA Plasmadynamics and Lasers Conference, Anaheim, CA.

[34] J. Driver and D. W. Zingg. Optimized natural-laminar-flow airfoils. In *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, 2006. AIAA 2006-0247.

[35] Richard P. Dwight and Joel Brezillon. Effect of various approximations of the discrete adjoint on gradient-based optimization. *AIAA Paper* 2006-0690, January 2006. Proceedings of the 44^{th} AIAA Aerospace Sciences Meeting & Exhibit, Reno, NV.

[36] Brian E. Earp, Datta V. Gaitonde, and Richard J. McMullan. Numerical simulation of magnetogasdynamic Mach reflection flow control. *AIAA Paper* 2004-2558, June 2004. Proceedings of the 35^{th} AIAA Plasmadynamics and Lasers Conference, Portland, OR.

[37] Mike Fagan and Alan Carle. Reducing reverse-mode memory requirements by using profile-driven checkpointing. *Future Generation Comp. Syst.*, 21(8):1380–1390, 2005.

[38] C. Faure and Y. Papegay. *Odyssée Version 1.6: The Language Reference Manual*. INRIA, 1997. Rapport Technique 211.

[39] Datta Gaitonde. Development of a solver for 3-D non-ideal magnetogasdynamics. *AIAA Paper* 1999-3610, June 1999. Proceedings of the 30^{th} AIAA Plasmadynamics and Lasers Conference, Norfolk, VA.

[40] Datta Gaitonde and J. S. Shang. The performance of flux-split algorithms in high-speed viscous flows. *AIAA Paper* 1992-0186, January 1992. Proceedings of the 30^{th} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[41] Datta Gaitonde and J. S. Shang. Accuracy of flux-split algorithms in high-speed viscous flows. *AIAA Journal*, 31(7):1215–1221, July 1993.

[42] Datta V. Gaitonde. Higher-order solution procedure for three-dimensional non-ideal magnetogasdynamics. *AIAA Journal*, 39(11):2111–2120, November 2001.

[43] Datta V. Gaitonde. Three-dimensional flow-through scramjet simulation with MGD energy-bypass. *AIAA Paper* 2003-0172, January 2003. Proceedings of the 41^{st} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[44] Datta V. Gaitonde. Simulation of local and global high-speed flow control with magnetic fields. *AIAA Paper* 2005-0560, January 2005. Proceedings of the 43^{rd} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[45] Datta V. Gaitonde and James H. Miller. Numerical exploration of shock interaction control with plasma-based techniques. *AIAA Paper* 2003-3483, June 2003. Proceedings of the 34^{th} AIAA Plasmadynamics and Lasers Conference, Orlando, FL.

[46] Datta V. Gaitonde and Jonathan Poggie. Simulation of magnetogasdynamic flow control techniques. *AIAA Paper* 2000-2326, June 2000. Proceedings of the AIAA Fluids 2000 Conference and Exhibit, Denver, CO.

[47] Datta V. Gaitonde and Jonathan Poggie. An implicit technique for 3-D turbulent MGD with the generalized Ohm's law. *AIAA Paper* 2001-2736, June 2001. Proceedings of the 32^{nd} AIAA Plasmadynamics and Lasers Conference, Anaheim, CA.

[48] Datta V. Gaitonde and Jonathan Poggie. Elements of a numerical procedure for 3-D MGD flow control analysis. *AIAA Paper* 2002-0198, January 2002. Proceedings of the 40^{th} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[49] Datta V. Gaitonde and Jonathan Poggie. Simulation of 3-D scramjet flowpath with MGD control. *AIAA Paper* 2002-2134, May 2002. Proceedings of the 33^{rd} AIAA Plasmadynamics and Lasers Conference, Maui, HI.

[50] Datta V. Gaitonde and Jonathan Poggie. Implicit technique for three-dimensional turbulent magnetoaerodynamics. *AIAA Journal*, 41(11):2179–2191, November 2003.

[51] R. Giering. TAF FastOpt, transformation of algorithms in Fortran. `http://www.FastOpt.com`. Accessed in April of 2007.

[52] R. Giering. TAMC tangent linear and adjoint model compiler. `http://www.autodiff.com/tamc`. Accessed in April of 2007.

[53] Ralf Giering and Thomas Kaminski. Applying TAF to generate efficient derivative code of Fortran 77-95 programs. In *Proceedings of GAMM 2002, Augsburg, Germany*, 2002.

[54] Ralf Giering, Thomas Kaminski, and Thomas Slawig. Generating efficient derivative code with TAF: Adjoint and tangent linear Euler flow around an airfoil. *Future Generation Comp. Syst.*, 21(8):1345–1355, 2005.

[55] Michael B. Giles and Niles A. Pierce. An introduction to the adjoint approach to design. In *Flow, Turbulence and Combustion*, volume 65, pages 393–415. Kluwer Academic Publishers, 2000.

[56] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4):979–1006, December 2002.

[57] Philip E. Gill, Walter Murray, and Michael A. Saunders. Users guide for SNOPT (version 6): A Fortran package for large-scale nonlinear programming. Report sol, December 2002. Department of Operations Research, Stanford University, Stanford, CA.

[58] Mark S. Gockenbach. Understanding Code Generated by TAMC. IAAA Paper TR00-29, Department of Computational and Applied Mathematics, Rice University, Texas, USA, 2000.

[59] A. Griewank, D. Juedes, H. Mitev, J. Utke, O. Vogel, and A. Walther. *ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++.* Institute of Scientific Computing, Technical University Dresden, Germany, 1998. User Manual.

[60] Andreas Griewank. *Evaluating Derivatives*. SIAM, Philadelphia, 2000.

[61] Sumeet Gupta, John C. Tannehill, Unmeel B. Mehta, and David W. Bogdanoff. Simulation of 3-D nonequilibrium seeded airflow in the NASA Ames MHD channel. *Journal of Thermophysics and Heat Transfer*, 21(2):276–283, 2007.

[62] Evgeniy P. Gurijanov and Philip T. Harsha. AJAX: New directions in hypersonic technology. *AIAA Paper* 1996-4609, November 1996. Proceedings of the 7^{th} AIAA International Space Plane and Hypersonic Conference, Norfolk, VA.

[63] Richard P. Hallion. The history of hypersonics: or, back to the future – again and again. *AIAA Paper* 2005-0329, January 2005. Proceedings of the 43^{rd} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[64] Laurent Hascoët and Valérie Pascual. TAPENADE 2.1 user's guide. Technical report 300, INRIA, 2004.

[65] Laurent Hascoët and Valérie Pascual. Extension of TAPENADE towards Fortran 95. In H. M. Bücker, G. Corliss, P. Hovland, U. Naumann, and B. Norris, editors, *Automatic Differentiation: Applications, Theory, and Tools*, Lecture Notes in Computational Science and Engineering. Springer, 2005.

[66] G. W. Hedstrom. Nonreflecting boundary conditions for nonlinear hyperbolic systems. *Journal of Computational Physics*, 30:222–237, August 1979.

[67] Patrick Heimbach, Chris Hill, and Ralf Giering. An efficient exact adjoint of the parallel MIT general circulation model, generated via automatic differentiation. *Future Generation Comp. Syst.*, 21(8):1356–1371, 2005.

[68] Raymond M. Hicks and Preston A. Henne. Wing design by numerical optimization. *AIAA Journal*, 15(7):407–412, July 1978.

[69] Charles Hirsch. *Numerical Computation of Internal and External Flows*. Wiley, New York, 1990.

[70] Klaus A. Hoffmann. *Fundamental Equations of Fluid Mechanics*. Engineering Education System, 1996.

[71] J. E. Horwedel. GRESS, a preprocessor for sensitivity studies of fortran programs. In *SIAM Workshop of Automatic Differentiation of Algorithms*, pages 243–250, Breckenridge, Colorado, jan 1991. CO. Soc. Ind. Appl. Math. Also available as ICASE technical report 97–61.

[72] J. E. Horwedel. GRESS version 2.0 user's manual. Technical Report ORNL/TM-11951, Oak Ridge National Laboratory, 1991.

[73] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260, sep 1988.

[74] Anthony Jameson, Sriram, and Luigi Martinelli. Aerodynamic shape optimization of transonic and supersonic aircraft configurations. *AIAA Paper* 2005-1013, January 2005. Proceedings of the 43^{rd} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[75] Antony Jameson. Multigrid algorithms for compressible flow calculations. In W. Hackbush and U. Trottenberg, editors, *Princeton University, MAE report 1743*, volume 1228 of *Lecture Notes in Mathematics*, pages 166–201. Springer-Verlag, 1986. also in Proceedings of the 2^{nd} European Conference on Multigrid Methods, Cologne, October 1985.

[76] Antony Jameson and Timothy J. Baker. Solution of the Euler equations for complex configurations. *AIAA Paper* 1983-1929, July 1983. Proceedings of the 6^{th} AIAA Computational Fluid Dynamics Conference, Danvers, MA.

[77] Antony Jameson, Niles A. Pierce, and Luigi Martinelli. Optimum aerodynamic design using the Navier–Stokes equations. In *Theoretical and Computational Fluid Dynamics*, volume 10, pages 213–237. Springer-Verlag GmbH, January 1998.

[78] Antony Jameson, Wolfgang Schmidt, and Eli Turkel. Numerical solution of the Euler equations by finite volume methods using Runge–Kutta time-stepping schemes. *AIAA Paper* 1981-1259, June 1981. Proceedings of the 14^{th} AIAA Fluid and Plasma Dynamic Conference, Palo Alto, CA.

[79] Donald B. Johnson and Jeffrey S. Robinson. X-43D conceptual design and feasibility study. *AIAA Paper* 2005-3416, May 2005. Proceedings of the 13^{th} AIAA/CIRA International Space Planes and Hypersonics Systems and Technologies, Capua, Italy.

[80] Eswar Josyula, Datta Gaitonde, and Joseph Shang. Nonequilibrium hypersonic flow solutions using the Roe flux-difference split scheme. *AIAA Paper* 1991-1700, June 1991. Proceedings of the 22^{nd} AIAA Fluid Dynamics, Plasma Dynamics and Lasers Conference, Honolulu, HI.

[81] Hyoung-Jin Kim and Kazuhiro Nakahashi. Discrete adjoint method for unstructured Navier–Stokes solver. *AIAA Paper* 2005-0449, January 2005. Proceedings of the 43^{rd} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[82] N. Kroll, K. Becker, H. Rieger, and F. Thiele. Ongoing activities in flow simulation and shape optimization within the German MEGADESIGN project. In *25^{th} ICAS Congress*, Hamburg, Germany, September 2006.

[83] Ki-Hwan Lee, Juan J. Alonso, and Edwin van der Weide. Mesh adaptation criteria for unsteady periodic flows using a discrete adjoint time-spectral formulation. *AIAA Paper* 2006-0692, January 2006. Proceedings of the 44^{th} AIAA Aerospace Sciences Meeting & Exhibit, Reno, NV.

[84] Robert Michael Lewis. Numerical computation of sensitivities and the adjoint approach. In Jeff Borggaard, John Burns, Eugene Cliff, and Scott Schreck, editors, *Computational Methods for Optimal Design and Control*, pages 285–302. Birkhäuser, 1998. Also available as ICASE technical report 97–61.

[85] H. Lomax, Thomas H. Pulliam, and David W. Zingg. *Fundamentals of Computational Fluid Dynamics*. Springer, 2003.

[86] J. N. Lyness and C. B. Moler. Numerical differentiation of analytic functions. *SIAM Journal on Numerical Analysis*, 4(2):202–210, 1967.

[87] Robert W. MacCormack. A new implicit algorithm for fluid flow. *AIAA Paper* 1997-2100, June 1997. Proceedings of the 13^{th} AIAA Computational Fluid Dynamics Conference, Snowmass Village, CO.

[88] Robert W. MacCormack. A fast and accurate method for solving the Navier–Stokes equations. In *21^{st} ICAS Congress*, Melbourne, Australia, September 1998.

[89] Robert W. MacCormack. A CFD procedure for aerospace applications. In *JSASS 13^{th} International Sessions, 37^{th} Aircraft Symposium*, Tokyo, Japan, October 1999.

[90] Robert W. MacCormack. An upwind conservation form method for magnetofluid dynamics. *AIAA Paper* 1999-3609, June 1999. Proceedings of the 30^{th} AIAA Plasmadynamics and Lasers Conference, Norfolk, VA.

[91] Robert W. MacCormack. A computational method for magnetofluid dynamics. *AIAA Paper* 2001-2735, June 2001. Proceedings of the 32^{nd} AIAA Plasmadynamics and Lasers Conference, Anaheim, CA.

[92] Robert W. MacCormack. Conservation form method for magnetofluid dynamics. *AIAA Paper* 2001-0195, January 2001. Proceedings of the 39^{th} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[93] Robert W. MacCormack. Three dimensional magneto-fluid dynamics algorithm development. *AIAA Paper* 2002-0197, January 2002. Proceedings of the 40^{th} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[94] Robert W. MacCormack. Simulation of hypersonic flow with strong magnetic field interaction. *AIAA Paper* 2006-0970, January 2006. Proceedings of the 44^{th} AIAA Aerospace Sciences Meeting & Exhibit, Reno, NV.

[95] Charles A. Mader, Joaquim R. R. A. Martins, Juan J. Alonso, and Edwin van der Weide. ADjoint: An approach for the rapid development of discrete adjoint solvers. *AIAA Journal*, XX(X):1–43, November 2006. Submitted.

[96] N. Malmuth, S. Zakharov, and A. Fedorov. Conical Navier–Stokes modeling of forebody vortex symmetry plasma control. *AIAA Paper* 2007-0219, January 2007. Proceedings of the 45^{th} AIAA Aerospace Sciences Meeting & Exhibit, Reno, NV.

[97] Andre C. Marta and Juan J. Alonso. Discrete adjoint formulation for the ideal MHD equations. *AIAA Paper* 2006-3345, June 2006. Proceedings of the 3^{rd} AIAA Flow Control Conference, San Francisco, CA.

[98] Andre C. Marta and Juan J. Alonso. High-speed MHD flow control using adjoint-based sensitivities. *AIAA Paper* 2006-8009, November 2006. Proceedings of the 14^{th} AIAA/AHI International Space Planes and Hypersonic Systems and Technologies Conference, Canberra, Australia.

[99] Andre C. Marta, Juan J. Alonso, and Lei Tang. Automatic magnetohydrodynamic control of hypersonic flow using a discrete adjoint formulation. *AIAA Paper* 2006-0370, January 2006. Proceedings of the 44^{th} AIAA Aerospace Sciences Meeting & Exhibit, Reno, NV.

[100] Luigi Martinelli. *Calculation of Viscous Flows with a Multigrid Method*. Ph.D. Dissertation, Princeton University, 1987.

[101] J. Martins, J. Alonso, and E. van der Weide. An automated approach for developing discrete adjoint solvers. *AIAA Paper* 2006-1608, May 2006. Proceedings of the 47^{th} AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Newport, RI.

[102] J. Martins, Charles A. Mader, and J. Alonso. ADjoint: An approach for rapid development of discrete adjoint solvers. *AIAA Paper* 2006-7121, September 2006. Proceedings of the 11^{th} AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, VA.

[103] Joaquim R. R. A. Martins. AER1415: Optimization concepts and applications. `http://mdolab.utias.utoronto.ca/teaching/optimization`. Accessed in February of 2007.

[104] Joaquim R. R. A. Martins. *A Coupled-Adjoint Method for High-Fidelity Aero-Structural Optimization*. Ph.D. Dissertation, Stanford University, October 2002.

[105] Joaquim R. R. A. Martins, Juan J. Alonso, and James J. Reuther. Complete configuration aero-structural optimization using a coupled sensitivity analysis method. *AIAA Paper* 2002-5402, September 2002. Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA.

[106] Joaquim R. R. A. Martins, Juan J. Alonso, and James J. Reuther. High-fidelity aerostructural design optimization of a supersonic business jet. *Journal of Aircraft*, 41(3):523–530, 2004.

[107] Joaquim R. R. A. Martins, Juan J. Alonso, and James J. Reuther. A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design. *Optimization and Engineering*, 6(1):33–62, March 2005.

[108] Joaquim R. R. A. Martins, Juan J. Alonso, and Edwin van der Weide. An automated approach for developing discrete adjoint solvers. In *Proceedings of the 2nd AIAA Multidisciplinary Design Optimization Specialist Conference*, Newport, RI, 2006. AIAA 2006-1608.

[109] Joaquim R. R. A. Martins, Ilan M. Kroo, and Juan J. Alonso. An automated method for sensitivity analysis using complex variables. *AIAA Paper* 2000-0689, January 2000. Proceedings of the 38th Aerospace Sciences Meeting, Reno, NV.

[110] Joaquim R. R. A. Martins, Charles A. Mader, and Juan J. Alonso. Adjoint: An approach for rapid development of discrete adjoint solvers. In *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Portsmouth, VA, 2006. AIAA 2006-7121.

[111] Joaquim R. R. A. Martins, Peter Sturdza, and Juan J. Alonso. The connection between the complex-step derivative approximation and algorithmic differentiation. In *Proceedings of the 39th Aerospace Sciences Meeting*, Reno, NV, 2001. AIAA 2001-0921.

[112] Joaquim R. R. A. Martins, Peter Sturdza, and Juan J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, 2003.

[113] Ken Mattsson and Jan Nordström. Summation by parts operators for finite difference approximations of second derivatives. *Journal of Computational Physics*, 199(2):503–540, September 2004.

[114] Ken Mattsson, Magnus Svärd, and Jan Nordström. Stable and accurate artificial dissipation. *Journal of Scientific Computing*, 21(1):57–79, August 2004.

[115] MPI. A message-passing interface standard. *The International Journal of Supercomputer Applications and High Performance Computing*, 8(3/4):159–416, 1994.

[116] Computer History Museum. Timeline of computer history. `http://www.computerhistory.org`. Accessed in April of 2007.

[117] Siva K. Nadarajah and Antony Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. *AIAA Paper* 2000-0667, January 2000. Proceedings of the 38^{th} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[118] Siva K. Nadarajah and Antony Jameson. Studies of the continuous and discrete adjoint approaches to viscous automatic aerodynamic shape optimization. *AIAA Paper* 2001-2530, June 2001. Proceedings of the 15^{th} AIAA Computational Fluid Dynamics Conference, Anaheim, CA.

[119] NASA. X-43A hypersonic scramjet-powered research aircraft. `http://www.nasa.gov/missions/research/x43-main.html`. Accessed in October of 2006.

[120] C. R. Nave. Maxwell's equations. `http://hyperphysics.phy-astr.gsu.edu/hbase/electric/maxeq.html`. Accessed in June of 2005.

[121] Marian Nemec, Michael J. Aftosmis, Scott M. Murman, and Thomas H. Pulliam. Adjoint formulation for and embedded-boundary Cartesian method. *AIAA Paper* 2005-0877, January 2005. Proceedings of the 43^{rd} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[122] Marian Nemec and David W. Zingg. Multipoint and multi-objective aerodynamic shape optimization. *AIAA Journal*, 42(6):1057–1065, June 2004.

[123] Eric J. Nielsen and Michael A. Park. Using an adjoint approach to eliminate mesh sensitivities in computational design. *AIAA Paper* 2005-0491, January 2005. Proceedings of the 43^{rd} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[124] R. Nowak, S. Krank, R. W. Porter, M.C. Yuen, and Ali Bulent Cambel. Magnetogasdynamic re-entry phenomena. *Journal of Spacecraft*, 4(11):1538–1542, November 1967.

[125] Department of Physics Astronomy and Geology at East Tennessee State University. ETSU Astronomical observatory - magnetic field strengths. `http://www.etsu.edu/physics/etsuobs/starprty/102498mwc/magtable.htm`. Accessed in December of 2005.

[126] Wolfgang K.H. Panofsky and Melba Phillips. *Classical Electricity and Magnetism*. Addison-Wesley, 1962.

[127] Chul Park. On convergence of computation of chemically reacting flows. *AIAA Paper* 1985-0247, January 1985. Proceedings of the 23^{rd} AIAA Aerospace Sciences Meeting, Reno, NV.

[128] O. Pironneau. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64:97–110, 1974.

[129] J. Poggie and D. V. Gaitonde. Magnetic control of hypersonic blunt body flow. *AIAA Paper* 2000-0452, January 2000. Proceedings of the 38^{th} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[130] J. Poggie and D. V. Gaitonde. Computational studies of magnetic control in hypersonic flow. *AIAA Paper* 2001-0196, January 2001. Proceedings of the 39^{th} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[131] Robert W. Porter and Ali Bulent Cambel. Magnetic coupling in flight magnetoaerodynamics. *AIAA Journal*, 5(4):803–805, April 1967.

[132] Kenneth G. Powell. An approximate Riemann solver for magnetohydrodynamics (that works in more than one dimension). ICASE Report 94-24, Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA, 1994.

[133] Kenneth G. Powell, Philip L. Roe, Timur J. Linde, Tamas I. Gombosi, and Darren L. De Zeeuw. A solution-adaptive upwind scheme for ideal magnetohydrodynamics. *Journal of Computational Physics*, 154(2):284–309, September 1999.

[134] Kenneth G. Powell, Philip L. Roe, Rho Shin Myong, Tamas Gombosi, and Darren De Zeeuw. An upwind scheme for magnetohydrodynamics. In *12^{th} AIAA Computational Fluid Dynamics Conference, San Diego, CA, June 19-22, 1995*, Collection of Technical Papers. Pt. 1 (A95-36501 09-34), pages 661–674. American Institute of Aeronautics and Astronautics, Washington, DC, 1995. AIAA Paper 1995-1704.

[135] J. D. Pryce and J. K. Reid. AD01, a Fortran 90 code for automatic differentiation. Technical Report Report RAL-TR-1998-057, Rutherford Appleton Laboratory, Chilton, Didcot, Oxfordshire, OX11 OQX, U.K., 1998.

[136] Python. Python programming language – official website. `http://www.python.org/`, 2007.

[137] John D. Ramshaw. A method for enforcing the solenoidal condition on magnetic field in numerical calculations. *Journal of Computational Physics*, 52(3):592–596, December 1983.

[138] James J. Reuther, Antony Jameson, Juan J. Alonso, Mark J. Rimlinger, and David Saunders. Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 1. *Journal of Aircraft*, 36(1):51–60, 1999.

[139] James J. Reuther, Antony Jameson, Juan J. Alonso, Mark J. Rimlinger, and David Saunders. Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 2. *Journal of Aircraft*, 36(1):61–74, 1999.

[140] A. Rhodin, U. Callies, and D.P. Eppel. IMAS: Integrated modeling and analysis system. `http://w3g.gkss.de/G/imashome`. Accessed in April of 2007.

[141] A. Rhodin, U. Callies, and D.P. Eppel. IMAS user manual: release 1.1. Technical Report GKSS 96/E/28, GKSS Research Center, Geesthacht, Germany, 2002.

[142] A. Rizzi. Numerical implementations of solid-body boundary conditions for the Euler equations. *ZAMM Journal of Applied Mathematics and Mechanics*, 58:T301–T304, 1978.

[143] P. L. Roe and D. S. Balsara. Notes on the eigensystem of magnetohydrodynamics. *SIAM Journal on Applied Mathematics*, 56(1):57–67, February 1996.

[144] Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal for Scientific and Statistical Computing*, 7(3):856–869, July 1986.

[145] J. S. Shang, Datta V. Gaitonde, and G. A. Updike. Modeling magneto-aerodynamic actuator for hypersonic flow control. *AIAA Paper* 2004-2657, June

2004. Proceedings of the 35^{th} AIAA Plasmadynamics and Lasers Conference, Portland, OR.

[146] Joseph Shang and Datta Gaitonde. Characteristic-based, time-dependent Maxwell equations solvers on a general curvilinear frame. *AIAA Paper* 1993-3178, July 1993. Proceedings of the 24^{th} AIAA Plasmadynamics and Lasers Conference, Orlando, FL.

[147] D. Shiriaev, A. Griewank, and J. Utke. *A User Guide to ADOL-F: Automatic Differentiation of FORTRAN Codes*. Institute of Scientific Computing, Technical University Dresden, Germany, 1995. Software Engineering in Scientific Computing: Preprint IOKOMO-04-95.

[148] Lee Shunn, Ali Mani, Steve Jones, and Gianluca Iaccarino. A new planet in 7 days: A job for BlueGene/L. Technical report, Center for Integrated Simulations, Stanford University, 2006.

[149] W. Squire and G. Trapp. Using complex variables to estimate derivatives of real functions. *SIAM Review*, 40(1):110–112, 1998.

[150] Ole Stauning and Claus Bendtsen. FADBAD++: Flexible automatic differentiation using templates and operator overloading in ANSI C++. Technical report, Technical University of Denmark, DK-2800 Lyngby, Denmark, 2007. `http://www2.imm.dtu.dk/~km/FADBAD`.

[151] Takashi Sugimura and Frank W. Vogenitz. Monte Carlo simulation of ion collection by a Rocket-Borne mass spectrometer for collisionless and transitional flowfields. Final report AD0767419, July 1973. TRW systems group, Redondo Beach, CA.

[152] R. C. Swanson, R. Radespiel, and E. Turkel. Comparison of several dissipation algorithms for central difference schemes. *AIAA Paper* 1997-1945, June 1997. Proceedings of the 13^{th} AIAA Computational Fluid Dynamics Conference, Snowmass Village, CO.

[153] T. Tanaka. Finite volume TVD scheme on an unstructured grid system for three-dimensional MHD simulation of inhomogeneous systems including strong background potential fields. *Journal of Computational Physics*, 111(2):381–389, April 1994.

[154] Top500. Top 500 supercomputing sites. `http://www.top500.org/lists/2006/11`. Accessed in April of 2007.

[155] Gábor Tóth and Dušan Odstrčil. Comparison of some flux corrected transport and total variation diminishing numerical schemes for hydrodynamic and magnetohydrodynamic problems. *Journal of Computational Physics*, 128(1):82–100, October 1996.

[156] Bram van Leer, James L. Thomas, Philip L. Roe, and Richard W. Newsome. A comparison of the numerical flux formulas for the Euler and Navier–Stokes equations. *AIAA Paper* 1987-1104, June 1987. Proceedings of the 8^{th} AIAA Computational Fluid Dynamics Conference, Honolulu, HI.

[157] Garret Vanderplaats. Very large scale continuous and discrete variable optimization. *AIAA Paper* 2004-4458, August 2004. Proceedings of the 10^{th} AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY.

[158] V. Venkatakrishnan and Antony Jameson. Computation of unsteady transonic flows by the solution of Euler equations. *AIAA Journal*, 26(8):974–981, August 1988. also AIAA Paper 1985-1514, July 1985.

[159] Marcel Vinokur. A rigorous derivation of the MHD equations based only on Faraday's and Ampères's laws. NASA Ames Research Center, presentation at the LANL MHD Workshop, 1996.

[160] Robert W. Walters, Pasquale Cinnella, David C. Slack, and David Halt. Characteristic-based algorithms for flows in thermo-chemical nonequilibrium. *AIAA Paper* 1990-0393, January 1990. Proceedings of the 28^{th} AIAA Aerospace Sciences Meeting, Reno, NV.

[161] MapleSoft: web page. Maple 11. http://www.maplesoft.com. Accessed in April of 2007.

[162] MathWorks: web page. Matlab 7.4. http://www.mathworks.com. Accessed in April of 2007.

[163] Wolfram: web page. Mathematica 6.0. http://www.wolfram.com. Accessed in April of 2007.

[164] Eric W. Weisstein. Maxwell's equations from Eric W. Weisstein's World of Physics. http://scienceworld.wolfram.com/physics/MaxwellEquations.html. Accessed in June of 2005.

[165] Pratt & Whitney. Pratt & Whitney Rocketdyne completes Mach 5 testing of worlds first closed-loop hydrocarbon-fueled hypersonic propulsion system. Press release, July 2006.

[166] Pratt & Whitney. Pratt & Whitney Rocketdyne's revolutionary scramjet engine successfully powers first X-51A simulated flight. Press release, April 2007.