EFFICIENCY IMPROVEMENTS OF RANS-BASED ANALYSIS AND OPTIMIZATION USING IMPLICIT AND ADJOINT METHODS ON UNSTRUCTURED GRIDS

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2006

Richard P. Dwight
School of Mathematics

Contents

54,000 words total

Abstract

The efficiency of an unstructured grid finite volume RANS solver is significantly improved using two implicit methods based on differing philosophies. The LU-SGS multigrid method aims to improve performance, while maintaining the low memory requirements and robustness of an explicit scheme. The First-Order Krylov Implicit (FOKI) method sacrifices these to some extent, in order to achieve high convergence rates and also avoid the use of a multigrid method, whilst care is taken that the method remains practical for large 3d cases. The speeds of the two schemes are compared with that of an existing, highly-tuned Runge-Kutta multigrid method, and it is seen that a factor of two speed-up can be obtained with no additional memory overhead using LU-SGS, and a factor of ten with FOKI. Attention is then turned to the efficiency of aerodynamic design optimization using gradient-based methods. Use of the Jacobian from the implicit methods allows construction of the adjoint of the flow solver. This adjoint is exact in the sense of being based on the full linearization of all terms in the solver, including all turbulence model contributions. From this starting point various approximations to the adjoint are derived with the intention of simplifying the development and reducing the memory requirements of the method. The effect of these approximations on the accuracy of the resulting design gradients, and the convergence and final solution of optimization problems is studied. The result is a tool for extremely rapid sensitivity evaluations.

Declaration

Copyright

Acknowledgements

The task of writing a thesis is not one which can be performed in isolation, and there are many people who have contributed to this work in ways both tangible and intangible. Foremost amongst the former group must be my friend and colleague Joël Brezillon, who has made the chapter on the adjoint method possible through his insight and experience in optimization, not to mention his dedicated hard work.

Deserving of particular mention are Jens Faßbender and Markus Widhalm for patiently answering uncountable questions on finite volume schemes, Axel Schwöppe for providing the crash course in Galerkin methods that was a prerequisite for the (abortive) chapter on deformation, as well as Bernard Eisfeld, Antonio Fazzolari, Ralf Hartmann, Cord Rossow, Joachim Held and Ralf Heinrich for many inspiring and productive discussions. Advice from Ken Badcock and Mark Woodgate from the University of Glasgow was extremely helpful during initial efforts in constructing the Jacobian.

To Norbert Kroll and all my colleagues in the Numerical Methods department at the DLR Braunschweig I owe thanks for the opportunity to work in such a stimulating environment, as well as for forbearance in the face of my awful German. To my professor, Peter Duck I am grateful for proposing this Ph.D. in the first place, and for performing all the functions of a doctoral supervisor.

Last but not least I would like to thank Krystyna Kurpiers, to whom I owe so much of the happiness of the last three years, and who was also responsible for the catchy acronym “FOKI”.

I never knew before what eternity was made for. It is to give some of us a chance to learn German. — Mark Twain

Chapter 1

Introduction

The efficiency of a Computational Fluid Dynamics (CFD) solver for the compressible Navier-Stokes (NS) equations is critical to its success as an engineering tool. As finite volume based NS methods have become established in the aircraft design process, they are subject to ever increasing demands on their abilities, from three principal directions. Firstly there is a requirement for modelling of increasingly complex geometries, which is met through the use of unstructured and hybrid grids. Secondly there is a demand for more accurate physical modelling, whereby the main area of interest is advanced turbulence models such as Reynolds Stress Models (RSM) and unsteady simulation in the form of Detached- and Large-Eddy Simulation (DES and LES). Finally there is a desire to use the existing flow solvers within inner loops for the purposes of optimization, trimming, and stability and control analysis.

The first two of these requirements have the effect of increasing the cost of individual flow calculations. Experience shows that flow solvers based on structured grids can be made significantly more efficient than unstructured grid solvers, partly because solution methods, in particular multigrid, are more effective on structured grids (Wild, 2004). The cost of improving the physical modelling is even greater. Use of RSM for compressible flow in three-dimensions requires 12 unknown variables per grid point, as compared to 6 for a one-equation turbulence model, with correspondingly increased costs. In addition the resulting equations are extremely poorly conditioned. On the other hand using DES on a complete aircraft configuration has only recently become possible as a research exercise, alone due to the computing resources required (Spalart & Bogue, 2003); its use in design may be considered still to be distant (Johnson *et al.*, 2003).

However high performance is demanded even from stationary RANS calculations with simple turbulence models, when large numbers of such analyses must be performed within an outer optimization loop, for example. For calculations on unstructured grids, performance of the solver is often the bottleneck which prevents the use of CFD for more complex applications. While computer performance is still improving exponentially, improvements in algorithmic convergence acceleration are also essential for achieving the desired modelling complexity. In fact some sources note that improvements in algorithms over the past 50 years have kept pace with improvements in computer power over the same period (Mavriplis, 1998)[1].

Efficiency of solver algorithms is therefore one of the most critical questions in

[1]For Poisson's equation.

modern CFD, and it will be addressed in the context of finite volume RANS methods using one-equation turbulence models in this thesis. In particular the application of the class of *implicit* time stepping methods is considered, see Section 1.1.

For the particular case of aerodynamic design, the high cost of the flow analysis makes gradient-based optimization algorithms attractive. However the evaluation of the design gradients is also an expensive operation; this difficulty will be tackled using the *adjoint* method, see Section 1.2. The resulting performance improvement, in combination with the improvement in the efficiency of the flow solver itself via implicit algorithms, are two essential parts of a system which can very rapidly optimize aerodynamic shapes. This thesis ends with some closing remarks in Chapter 6.

1.1 Implicit Time Stepping Methods

We are concerned with the solution to steady-state of ordinary differential equations of the form

$$\frac{\mathrm{d}W_i}{\mathrm{d}t} + R_i(W) = 0,$$

where W are the unknowns and R is the non-linear residual resulting from the spatial discretization of the RANS equations. In particular we use pseudo time stepping and are interesting in optimizing the convergence of the resulting iteration.

Implicit methods for the above equation are characterized by the presence of the residual evaluated at the unknown time-level, $R(W^{n+1})$, creating in general a non-linear algebraic system of equations to be solved for W^{n+1}. The method of resolving this system by choosing some linearization of the residual R and thus reducing the non-linear system to a linear algebraic equation, forms the class of implicit methods studied here (the compliment being implicit methods solved at each step using a non-linear sub-iteration, e.g. the dual-time method). A further distinction can be made between methods that perform the necessary linearization on the continuous governing equations, and those that linearize the discretized equations. The later approach is much more widely used in CFD, perhaps because it has the advantage of decoupling the time and space discretizations (known as the *method of lines*), thereby allowing relatively independent investigation of each. It is also the approach adopted here.

Implicit methods of this sub-class can in turn be classified by three important choices made during their formulation:

(a) Temporal discretization formula, e.g. Backward-Difference Formula (BDF).

(b) Choice of approximation of the Jacobian of R.

(c) Solution of the resulting linear system: which solver, to what accuracy etc.?

Consequently, the most well-known implicit method, Newton's method, is specified by: (a) the backward-Euler formula with $\Delta t \to \infty$, (b) use of the exact Jacobian of R, and (c) exact solution of the linear system, whereby the choice of particular linear solver is of secondary importance.

Evaluation of the exact Jacobian, and solution of the linear system to machine accuracy tend to be two extremely expensive operations; as such Newton's method is one

	Jacobian 1st-Order		Jacobian 2nd-Order	
	Approximate	Exact	Approximate	Exact
LU-SGS	Y	N	N/S	N/S
SGS	N	Y	N	N
Jacobi	N	N	N/S	N/S
Line solver	???	???	???	???
Krylov	N	N	N/S	N/S
Precon. Krylov	N	Y	Y	Y
Findiff. Krylov	N	N	N	N
Linear Multigrid	???	???	???	???

Table 1.1: Schematic of implicit methods for CFD. Key: Y = Examined in this thesis and found to be useful; N = Examined in this thesis and found to be inferior to other comparable methods; N/S = Nonsensical, unlikely to be useful because of the LHS accuracy-linear solver accuracy mismatch; ??? = Performance unknown.

of the most computationally expensive implicit methods per step. As is well known, in the limit of close proximity to the exact solution, the convergence is quadratic, and the method in this region is therefore superior to all other methods that show purely linear convergence. However in practical aerodynamics, solution of the non-linear equations to such high accuracy is not required. Rather, most effort is expended in reducing the norm of the residual by 3 to 5 orders of magnitude. In this critical region, a more efficient method may be obtained by suitable choice of *approximate* Jacobian and *inexact* linear system solver, as will be shown. For convergence to a steady state, the solution is independent of the choice of temporal discretization, and the simplest possible option is taken: backward Euler.

Consequentially there is a two-parameter family of schemes to investigate, shown schematically in Table 1.1. For a second-order accurate spatial discretization, a natural Jacobian approximation is a Jacobian based on a first-order accurate convective flux discretization (henceforth denoted by the *first-order Jacobian*). The Jacobian may be the exact derivative of the first-order fluxes or may contain additional simplifying approximations. The *second-order Jacobian* is considerably more expensive to formulate and store, but may also be evaluated exactly or approximately. Linear systems formed with these Jacobians may then be solved with any of the methods shown, which represent the main classes of iterative linear solvers for sparse systems. All the methods represented in Table 1.1 are investigated in this thesis, with the exception of those based on the line-implicit solver and linear multigrid.

The various combinations of linear solver and Jacobian are classified according to their performance and practicability, as explicitly examined herein. Question marks indicate schemes which have not been examined here and therefore have unknown performance.

Starting at the top of Table 1.1: LU-SGS will here be shown to be an effective scheme when combined with a suitably chosen approximate first-order Jacobian, but much less effective when the exact first-order Jacobian is used, see Chapter 3. Further, the alternative fixed-point iterations, symmetric Gauss-Seidel (SGS) and Jacobi are shown to be inferior to LU-SGS for the problems considered, unless a much more accurate first-order Jacobian is taken. In that case SGS is significantly more efficient than Jacobi. A line solver has not been investigated here, but is known to be effective, especially for viscous flows (Mavriplis, 1998). All these methods however are ineffective in combination with a second-order Jacobian due to the increased stiffness and reduced diagonal-dominance of the system, as well as the more expensive matrix-vector product. For such a system fixed-point iterations must generally be used in combination with either a Krylov-subspace solver or multigrid on the linear system.

Although it is possible to apply a Krylov method directly to the linear system, indicated by the line "Krylov" in Table 1.1, this technique is generally only effective for relatively well-conditioned systems. The only systems that can be described as well-conditioned are some approximate first-order Jacobian systems, which are later seen to be more efficiently solved with LU-SGS. Krylov methods preconditioned using one of the fixed-point iterations described above on the other hand is very generally applicable, and widely used in CFD with both first-order Jacobians (Dubuc *et al.*, 1996) and second-order Jacobians (Chisholm & Zingg, 2002). Explicit evaluation and storage of the Jacobian can be avoided using a finite difference approximation, however this tends to be more costly in CPU time than the explicit formulation due to the number of residual evaluations involved.

The two methods examined in Chapter 4 both use the exact first-order Jacobian, one using an SGS iteration (FOGSI), the other a preconditioned Krylov iteration (FOKI). As can be seen from the table, both prove to be effective methods.

Finally, multigrid in CFD is most often used in the Full-Approximation Storage (FAS) form (Jameson & Baker, 1984), but may also be used inside an implicit method as a linear solver, a possibility that shows some considerable promise (Mavriplis, 2002), but which is not investigated here.

1.1.1 Literature Review

One of the most widely used convergence acceleration algorithms in CFD today was described in all significant details more than 20 years ago (Jameson & Baker, 1984). The method uses a particularly simple explicit Runge-Kutta (RK) method, whereby the dissipative part of the convective fluxes is evaluated only at certain RK stages. In addition local time stepping, directional implicit smoothing of the residuals, and Full-Approximation Storage (FAS) multigrid combine to make an extremely effective scheme. A variant of this method formed the only time stepping scheme of the DLR unstructured RANS solver *TAU*, up until the work presented in this thesis, and is a highly tuned method, experience in its use having been accumulated over many years. The scheme will henceforth be referred to as the Runge-Kutta (RK) method, and will be used as a reference scheme throughout.

Implicit methods with approximate first-order Jacobians and weak linear solvers in CFD were first proposed in the context of structured grid methods with implicit treatment of lines in the direction normal to the wall, as presented in (Venkatakrishnan, 1998) and (Turkel *et al.*, 1999), and the Alternating Direction Implicit (ADI) scheme, in which implicit line treatment in each grid direction is performed (Peaceman & Rachford, 1955). These methods are well known to be unconditionally unstable in three-dimensions however, leading to a common modification, Diagonally Dominant

(DDADI) schemes (Faßbender, 2003).

The previous methods are restricted to structured grids, but have been used on structured parts of mixed-element (hybrid) grids with great success (Mavriplis, 1998), and even time-accurately (Yoh & Zhong, 2004). General unstructured grids require an alternative solution method, proposed first in CFD for aerodynamics in (Jameson & Turkel, 1981), where a symmetric Gauss-Seidel (GS) sweep was used on a structured grid to solve a heavily approximated linear system. This became known as the LU-SGS (or LU-SSOR) method, and was further developed in (Yoon & Jameson, 1986b; Yoon & Jameson, 1988) and more recently in (Luo *et al.*, 1998; Sharov *et al.*, 2000). A method incorporating a modified form of LU-SGS has also been shown, for simple configurations and a particular discretization, to allow convergence of Euler computations within 10 multigrid cycles (Jameson & Caughey, 2001). Also reported by several sources is that by using a single Jacobi sweep rather than GS, for discretizations including matrix dissipation, a scheme resembling a matrix preconditioner may be effective (Pierce, 1997; Pierce *et al.*, 1997). Generally all these methods are used as multigrid smoothers.

Implicit methods with exact first- or second-order Jacobians are less common, possibly due to the considerably greater development effort, and increased memory requirements and parallelization problems (Cai *et al.*, 1995). A means of avoiding storage of the Jacobian is via Jacobian Free Newton-Krylov (JFNK) methods (Knoll & Keyes, 2004), which approximate Jacobian-vector products using finite differences. Zingg and coworkers use an explicitly stored first-order Jacobian to precondition a JFNK method, allowing the use of the very effective ILU method as a preconditioner (Chisholm & Zingg, 2002; Wong & Zingg, 2005). Others avoid exact Jacobians completely by using a first-order Jacobian in the Newton method (Cantariti *et al.*, 1999; Cantariti *et al.*, 1999), whereby both storage, and effort in the linear system solution are saved. The effects of various Krylov solvers on Newton problems resulting from CFD has been studied (Meister, 1998), as have the effects of various preconditioners. A recently proposed scheme (Rossow, 2005) is one of few to use accurate Jacobians and not a Krylov method, but a GS iteration, thereby reducing memory requirements.

1.1.2 Overview

Two novel variants of implicit methods are proposed for a spatial discretization involving the JST (Jameson *et al.*, 1981) scheme. Throughout this thesis we take two distinct attitudes to the question of memory requirements of the algorithms. Initially we attempt to devise an implicit scheme that uses no more memory than that of the Runge-Kutta, thus allowing it to be a slot-in replacement for that scheme in every situation. Later we recognize that some increase in memory requirements may be acceptable, and necessary for further improvement in solver performance. Hence the trade-off between memory and efficiency is explored in some detail.

The first method resembles the LU-SGS method of (Yoon & Jameson, 1988) used as a multigrid smoother is examined, Chapter 3. The goal is to devise a scheme with all the advantages of Runge-Kutta, i.e. low memory requirements, low computational effort per iteration, easy parallelizability, and easy implementation, but that additionally admits a high CFL number. The former allows the scheme to function as a slot-in replacement for Runge-Kutta, and thereby admits application to very large test cases. The later will allow faster convergence rates than seen with Runge-Kutta. This is achieved by noting that the Jacobian of the JST scheme takes a very simple form in the interior of the field, in particular its block diagonal at each point is a multiple of the identity matrix. By simplifying the Jacobians of boundary conditions and viscous fluxes this property is preserved, and the approximate Jacobian block diagonal can be stored with a single floating-point number at each grid point. Inversion is then trivial, and off-diagonal entries may be rapidly calculated on the fly. The turbulence model treatment follows a similar pattern, whereby the mean flow and turbulence equations are fully decoupled in the Jacobian but calculated simultaneously, allowing rapid matrix and residual evaluation as well as separate treatment.

Comparisons with the highly tuned explicit Runge-Kutta method already described are undertaken, and the method is found to converge 10-50% faster in terms of iterations, while one LU-SGS iteration costs approximately 90% of one RK iteration.

Secondly in Chapter 4 the priorities are changed; a scheme with significantly greater memory requirements is allowed, but it should perform well without multigrid. The novel scheme developed is denoted the First-Order Jacobian, Krylov Implicit (FOKI) scheme, which is similar to the scheme considered in a structured context for upwind discretizations in (Cantariti *et al.*, 1999), but differs in its application to unstructured grids and the JST scheme, and in the treatment of the turbulence equations. The exact Jacobian of the first-order discretization is considered, including boundary conditions and viscous fluxes. Because the turbulence discretization includes only immediate point neighbours in its stencil it is linearized exactly, and is solved decoupled from and independently of the mean flow problem.

The solution method consists of ILU(0) preconditioned GMRES, and convergence is compared with the LU-SGS scheme of the previous chapter. A factor of 4-5 improvement in CPU time over that method is recorded for turbulent cases at high Reynolds numbers.

1.2 The Adjoint Method of Flow Sensitivities

Given the considerable effort required to evaluate the exact Jacobian of the full finite volume discretization in order to build a Newton method, as described in Section 4, it is worth considering whether this construction - which amounts to a linearization of the entire flow solver - may be useful in other contexts. Indeed there are several potential applications, and one in particular that promises to be of very considerable use in the aerodynamic design process, the *adjoint method*, which has been further developed in the context of this thesis, and which is the subject of this chapter.

In aerodynamic design one typically starts from a *baseline* geometry, a parameterization of the shape of the geometry, and a quantity of interest such as aerodynamic drag on the geometry (the *cost function*). The objective is to find the choice of coefficients of the parameterization (the *design variables*), such that the cost-function is minimized. Additionally the problem may be subject to one or more constraints.

Two features distinguish aerodynamic design from other design problems. Firstly the evaluation of the cost-function is typically very expensive, one evaluation corresponds to a solution of the Navier-Stokes equations on the given geometry. Secondly, because shapes must be parameterized, the problem is often characterized by very large numbers of design variables. For two-dimensional design of a profile, 10-30 design variables are typical, and when designing a three-dimensional wing several profiles and a wing planform may be parameterized, routinely leading to of the order of 100 design variables. The optimization problem then consists of a search in a 100-dimensional design space, which combined with the expense of cost-function evaluations, means that only gradient-based optimization methods are admissible.

Gradient-based optimization characterized by the steepest descent method requires two basic steps: first the evaluation of the search direction - the gradient of the cost-function with respect to the design variables - which results in the most rapid improvement of the design locally; and second a one-dimensional search in this direction, consisting of repeated evaluations of the cost-function until a minimum is found in this one-dimensional subspace; this basic process is repeated until no further improvement is obtained. The derivative of the cost-function with respect to a large number of design variables is therefore required. The adjoint method provides a means of performing this with an effort only weakly dependent on the number of design variables; the technique is described in detail in Section 5.3.

One of the earliest applications of adjoint methods to aerodynamics problems is found in the works of Pironneau, who devised optimality conditions for drag on two-dimensional bodies, first in Stokes flow (Pironneau, 1973), then for convection dominated flow (Pironneau, 1974). The work was shortly thereafter applied numerically to aerodynamic design (Glowinski & Pironneau, 1975). Effort has since been applied to the treatment of increasingly complex problems. Jameson popularized the method in the aerodynamic community with design of profiles using a continuous adjoint of the Euler equations (Jameson, 1988). Since then a contentious issue has been the choice of continuous or discrete adjoint (Sirkes & Tziperman, 1997). The former involves adjointing the continuous equations before discretizing them in order to solve them numerically (Gauger & Brezillon, 2003; Brezillon & Gauger, 2004); the later adjoints the already discretized equations. Each has advantages, but the discrete has gained dominance recently, due to its straightforward formulation, and its ability to treat general viscous problems (Nadarajah & Jameson, 2001; Kim *et al.*, 2002). As a result it has become a relatively mature technique (Giles *et al.*, 2003). However recent work suggests a generalization of the continuous adjoint for viscous problems may be possible (Castro *et al.*, 2006).

Recently (Mavriplis, 2006), building on previous work (Nielsen & Park, 2005; Mavriplis, 2005), showed that by adjointing not only the flow solver, but the entire optimization chain in a discrete manner, including surface mesh parameterization and grid deformation, an optimization of the wing of a transport aircraft with an extremely large number of design variables could be performed in less than 6 hours on a standard 16 processor cluster. However, the effort required to develop a discrete adjoint of a given flow solver is very high, as it involves differentiating all parts of the discretization, and often storing the resulting Jacobian matrix (Brezillon & Dwight, 2005). One effort to avoid this overhead uses Automatic Differentiation (AD) tools (Griewank, 2000; Griewank & Walther, 2002), but these are not yet mature enough to be applied to complete flow solvers. Another uses a modified form of finite differences in complex variables, and has been applied to large test cases (Nielsen & Kleb, 2005).

Here we consider a third approach, which involves using an approximation to the Jacobian; a modification which must influence the resulting gradients. Despite the fact that the idea is widely used (Löhner *et al.*, 2003; Soto *et al.*, 2004; Reuther *et al.*, 1999), there have been few studies into its effect on the resulting optimization. Nielsen compared the gradients of several approximations (Nielsen, 1998), and Kim et al. examined the effect on optimization of a constant eddy-viscosity assumption (Kim *et al.*, 2003).

In Chapter 5 an exact adjoint method is constructed, and then five different simplifying approximations are made, each with the aim of reducing the development and computational effort involved. Optimizations are then performed on two test problems using two optimization strategies, and the optima achieved and the convergence behaviour are compared with those of the exact adjoint. It is seen that the Jacobian may be simplified significantly without seriously damaging the optimization result, see also (Dwight & Brezillon, 2006).

Chapter 2

Discretization and Jacobians

2.1 Introduction

It is our objective to significantly improve the efficiency of the unstructured finite volume Navier-Stokes solver of the DLR, the *TAU*-Code, which is widely used in industry and research, and consequently has been validated against experimental results and other numerical methods for a large variety of applications (Gerhold *et al.*, 1997; Rudnik *et al.*, 2004; Kroll & Fassbender, 2005).

We therefore adopt the philosophy that the spatial discretization is given and immutable, and that our objective is purely to improve the efficiency of the solution of the resulting discrete equations. This has the disadvantage of allowing little flexibility - and it is often the case that a small change in discretization can result in a considerable simplification of the Jacobian, see for example Section 2.8. On the other hand it has the considerable advantage of eliminating the need for further verification and validation work on the numerical results. Provided that the equations are fully converged, their solution is independent of the convergence method used (neglecting the possibility of multiple solutions, which are rarely observed in practice[1]). For this reason the previous works (Kroll & Fassbender, 2005) are considered sufficient validation of the spatial discretization described below, and no comparison of experimental with numerical results is given herein.

The solver *TAU* includes a wide variety of spatial discretizations. In this chapter a complete and accurate description of one particular spatial discretization, including boundary conditions and turbulence model, is given. This is the discretization most commonly used for transonic aerodynamics applications, and that which is used to obtain the majority of numerical results given in this thesis.

The focus of this thesis is on implicit methods, an important component of which - the Jacobian - is derived directly from the spatial discretization. For this reason derivatives are presented alongside discretizations for certain elements of the scheme. It is not the intention of the author to give a complete Jacobian for the scheme, which would run to at least a hundred pages; rather to give an impression of the process, the necessary steps and effort required. Where Jacobians of certain elements of the scheme are particularly simple, or where a suitable approximation can simplify the Jacobian significantly, these are given.

[1]With the notable exception of inviscid calculations on aerofoils with blunt trailing edges and no explicit enforcement of the Kutta condition.

Whereas the spatial discretization is certainly not original to this thesis, having been developed in *TAU* principally by others (Gerhold *et al.*, 1997; Galle, 1995; Galle, 1999), and bearing a strong resemblance to many well known schemes in the literature (Jameson *et al.*, 1981; Mavriplis, 1997; Pierce *et al.*, 1997); and the derivation of the Jacobian of a finite volume method is also nothing new (Woodgate *et al.*, 1997; Nielsen *et al.*, 1995; Meister, 1998), the details of the efficient constuction of the Jacobian given here, for example where a suitable choice of spatial discretization or derivative approximation leads to particularly simple expressions for the derivatives, are unique to this thesis. Also an original theoretical justification is presented for the common practice of neglecting the derivatives of the dissipation coefficients in the JST scheme, Section 2.6.2, which demonstrates that the terms which are neglected are of higher order in the grid spacing Δx, than the remaining terms.

2.2 The Navier-Stokes Equations

The governing equations considered are the compressible Euler and Navier-Stokes equations. We consider first instantaneous equations, which implicitly contain the physics of turbulence, and then average them in time to eliminate turbulence fluctuations, whose effect will instead be modelled.

2.2.1 The Instantaneous Equations

The compressible Navier-Stokes equations in conservation form are

$$\frac{\partial W}{\partial t} + \nabla \cdot (f^c(W) - f^v(W)) = 0, \tag{2.1}$$

or equivalently

$$\frac{\partial W}{\partial t} + \frac{\partial}{\partial x_i} f_i^c(W) - \frac{\partial}{\partial x_i} f_i^v(W) = \frac{\partial W}{\partial t} + \mathfrak{R}(W) = 0, \tag{2.2}$$

where summation convention is applied on the index i, and where W is the conservative state vector,

$$W = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix}, \tag{2.3}$$

and the convective and viscous flux tensors f^c and f^v are composed of the inviscid and viscous flux vectors f_i^c and f_i^v in the three coordinate directions, $i \in \{x, y, z\}$.

In 3D they are

$$\begin{aligned}
f_x^c &= \begin{bmatrix} \rho u \\ \rho uu + p \\ \rho uv \\ \rho uw \\ \rho H u \end{bmatrix}, &
f_x^v &= \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ \tau_{xi} U_i + q_x \end{bmatrix}, \\
f_y^c &= \begin{bmatrix} \rho v \\ \rho vu \\ \rho vv + p \\ \rho vw \\ \rho H v \end{bmatrix}, &
f_y^v &= \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ \tau_{yi} U_i + q_y \end{bmatrix}, \\
f_z^c &= \begin{bmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho ww + p \\ \rho H w \end{bmatrix}, &
f_z^v &= \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ \tau_{zi} U_i + q_z \end{bmatrix},
\end{aligned} \tag{2.4}$$

where $U = (u,\, v,\, w)^T$ is the velocity vector, τ is the viscous shear stress tensor and q is the heat flux vector. The solution of the Euler equations will also be considered; they are obtained by neglecting the viscous fluxes in (2.1).

For a calorically perfect gas, pressure is defined by the state equation

$$p = (\gamma - 1)\rho \left\{ E - \frac{1}{2} U^2 \right\}, \tag{2.5}$$

where E is the specific total energy per unit mass, and γ is the gas dependent ratio of specific heats, which is taken to be 1.4 for air, and additionally the total enthalpy is defined as

$$H = E + \frac{p}{\rho}. \tag{2.6}$$

The viscous shear stresses are given by

$$\tau = \mu_l \left(\nabla U + \nabla U^T \right) + \lambda_l \nabla \cdot U\, I \tag{2.7}$$

(where I is the identity matrix), or equivalently by

$$\begin{aligned}
\tau_{xx} &= 2\mu_l \frac{\partial u}{\partial x} + \lambda_l \nabla \cdot U, \\
\tau_{yy} &= 2\mu_l \frac{\partial v}{\partial y} + \lambda_l \nabla \cdot U, \\
\tau_{zz} &= 2\mu_l \frac{\partial w}{\partial z} + \lambda_l \nabla \cdot U, \\
\tau_{xy} &= \tau_{yx} = \mu_l \left\{ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right\}, \\
\tau_{xz} &= \tau_{zx} = \mu_l \left\{ \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right\}, \\
\tau_{yz} &= \tau_{zy} = \mu_l \left\{ \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right\},
\end{aligned} \tag{2.8}$$

where local laminar bulk viscosity by Stokes hypothesis for a monatomic gas is

$$\lambda_l = -\frac{2}{3}\mu_l. \tag{2.9}$$

The heat fluxes are given by Fourier's law,

$$q = \kappa_l \nabla T, \tag{2.10}$$

with the thermal conductivity and the temperature defined by

$$\kappa_l = \frac{c_p \mu_l}{\mathrm{Pr}_l}, \qquad T = \frac{p}{\rho \Re}, \tag{2.11}$$

where $\Re$ is the universal gas constant, which is set to unity when non-dimensionalizing the equations. The local variation of molecular viscosity with temperature is modelled by Sutherland's Law,

$$\mu_l = \mu_{l,\infty} \cdot \left(\frac{T}{T_\infty}\right)^{1.5} \cdot \frac{T_\infty + \bar{T}}{T + \bar{T}}, \tag{2.12}$$

whereby $\bar{T} = 110.4\text{K}$ is Sutherland's constant. The law is used to model the local variation in thermal conductivity in exactly the same way, so that

$$\frac{\mu_l}{\mu_{l,\infty}} = \frac{\kappa_l}{\kappa_{l,\infty}}, \tag{2.13}$$

and the Prandtl number

$$\mathrm{Pr}_l = \frac{c_p \mu_{l,\infty}}{\kappa_{l,\infty}}, \qquad c_p = \Re \frac{\gamma}{\gamma - 1}, \tag{2.14}$$

is constant everywhere.

2.2.2 The Favre Averaged Equations

In order to respect the influence of turbulence without resolving every turbulent eddy, the flow equations are Favre averaged, i.e. time-averaged with mass weighting. The instantaneous flow quantities in (2.2) are substituted for Favre averaged quantities plus a time-dependent fluctuation, i.e.

$$W = \tilde{W} + W'', \tag{2.15}$$

where the mass-average is defined as

$$\tilde{W}(x) = \frac{1}{\bar{\rho}} \lim_{t' \to \infty} \frac{1}{t'} \int_t^{t+t'} \rho(x,s) W(x,s)\, ds, \tag{2.16}$$

where $\bar{\rho}$ is the conventional Reynolds averaged density. Thus the procedure rests on the assumption that the time scale of turbulent motion is much shorter than that of the mean motion. By mass-averaging the result, the Favre averaged Navier-Stokes equations are obtained.

The equations are substantially identical to the instantaneous flow equations with instantaneous replaced by mean quantities, except for the introduction of the turbulence correlations

$$\overline{\rho U'' \otimes U''}, \quad \overline{\rho U'' \otimes U''} \cdot \tilde{U}, \quad \overline{\rho h'' U''}, \quad \overline{\tau \cdot U''}, \quad \frac{1}{2}\overline{\rho U''(U'' \cdot U'')}, \tag{2.17}$$

which are modelled using some closure approximations. All models under consideration here use the Boussinesq *eddy viscosity assumption* (Boussinesq, 1877), which states that the Reynolds stress tensor may be modelled as

$$-\overline{\rho U'' \otimes U''} = \mu_t \left(\nabla U + \nabla U^T - \frac{2}{3} \nabla \cdot U\, I \right) - \frac{2}{3} \bar{\rho} k\, I, \tag{2.18}$$

for a suitable turbulent viscosity μ_t and turbulent kinetic energy k defined as

$$k = \frac{1}{2} \frac{\overline{\rho U'' \cdot U''}}{\bar{\rho}}. \tag{2.19}$$

Thereupon the momentum equations reduce to the instantaneous equations with a modified *effective viscosity* μ_e,

$$\mu_e = \mu_l + \mu_t, \tag{2.20}$$

and a modified pressure

$$p^* = p - \frac{2}{3} \bar{\rho} k, \tag{2.21}$$

and the second correlation in (2.17) results in an extra term in the energy equation. Similarly $\overline{\rho h'' U''}$ is approximated as a heat flux, giving an *effective thermal conductivity* κ_e,

$$\kappa_e = \kappa_l + \kappa_t, \tag{2.22}$$

which replaces κ_l in the viscous terms. Typically

$$\kappa_t = \frac{c_p \mu_t}{\mathrm{Pr}_t}, \tag{2.23}$$

and the turbulence Prandtl number Pr_t is a constant. The last two correlations of (2.17) may be interpreted as diffusion of k, and are therefore included as an extra k diffusion term in the energy equation. See (Wilcox, 1998) for a more complete discussion.

The purpose of a turbulence model is then to provide a value for μ_t and possibly k. One-equation models such as Spalart-Allmaras (Spalart & Allmaras, 1992) consist of a transport equation for some modified eddy-viscosity $\tilde{\nu}_t$, and terms involving k are typically neglected. Two-equation models such as Wilcox $k - \omega$ (Wilcox, 1998) provide transport equations for k and one other quantity, μ_t is then modelled as some function of these.

2.3 Flow Regime

The equations of the previous section display radically different behaviour depending on the flow regime, therefore it is also necessary to indicate the values of the three

Quantity	Symbol	Typical value
Mach number	M_∞	$\approx 0.2 - 1.5$
Reynolds number	Re	$\approx 1 \times 10^6 - 100 \times 10^6$
Prandtl number	Pr_l	≈ 0.72
Turb. Prandtl number	Pr_t	≈ 0.9
Reynolds Length	x_{Re}	$\approx 0.1 - 10.0\,\mathrm{m}$
Ratio of specific heats	γ	≈ 1.4
Temperature	T	$\approx 273.15\,\mathrm{K}$
Pressure	p	$\approx 15000 - 200000\,\mathrm{Nm}^{-2}$
Velocity	$\|U\|$	$\approx 10 - 500\,\mathrm{ms}^{-1}$
Density	ρ	$\approx 0.2 - 2.0\,\mathrm{kg\,m}^{-3}$

Table 2.1: Dimensional flow parameters and typical values for flow quantities in SI units, representing flow over large transport aircraft at standard atmospheric conditions.

flow parameters: the Mach number M_∞, the Reynolds number Re, and the Prandtl number Pr. Typical values representing flow over transport aircraft at standard atmospheric conditions are given in Table 2.1.

Of particular consequence is the large Mach number range, which demands the consideration of compressibility effects at the high end, while making standard methods for compressible fluids stiff at the low end, where the speed of sound dominates the speed of convection. Of even more importance are the exceptionally high Reynolds numbers, which cause thin boundary-layers to form on no-slip walls, which in turn require high grid resolution in the wall normal direction. The large discrepancy between the wall-normal and wall-tangent grid length scales is the dominant source of stiffness in the problems considered. The Prandtl numbers have a relatively minor influence on the flow. The equations are non-dimensionalized before being discretized (Le Chuiton, 2004).

2.4 Finite Volume Discretization

The governing equations may be derived by considering conservation of mass, momentum and energy within an arbetrary stationary *control volume* Ω. Because these quantities are conserved the rate of change of each within Ω must be equal to the flux of each through the walls of Ω. In integral form this statement may be written

$$\frac{\partial}{\partial t}\int_{\Omega} W\,d\Omega + \oint_{\partial\Omega}\Big\{f_T^c(W) - f_T^v(W)\Big\}\cdot n\,d(\partial\Omega) + \int_{\Omega} \mathcal{S}_T(W)\,d\Omega = 0, \tag{2.24}$$

where $\partial\Omega$ is the boundary of Ω, with outer normal n, and where $\mathcal{S}(W)$ represents some non-conservative source term vector, that may arise in turbulence models and which is zero for the mean-flow equations, and the subscript T indicates that the flux tensors have been extended to include the - as yet unspecified - turbulence transport equations. All other quantities are as in Section 2.2.1.

The problem domain is divided into a *grid* of non-overlapping control-volumes (or *cells*) Ω_i, each with an associated central node, chosen such that the integrals of (2.24) are easy to approximate numerically. In particular *TAU* works on the *dual grid* of approximate Voronoï volumes of a *primary grid* consisting in 3D of tetrahedra, triangular prisms, pyramids and hexahedra, whereby no hanging nodes or hanging edges are allowed. See Figures 2.1 and 2.2 for depictions of the dual-grid metric cells and nodes. Also Figure 3.13 depicts some primary grids.

The full discretization of the second and third integrals in (2.24) on the given grid is denoted the *residual* R, so that

$$\frac{\partial}{\partial t}\int\limits_{\Omega_i} W_i \, d\Omega + R_i(W) = 0, \tag{2.25}$$

is the semi-discrete version of (2.24). The Jacobian will be denoted $\partial R/\partial W$.

In this thesis we are concerned solely with second-order accurate methods. The values of the flow variables are approximated as constants on the control-volumes[2], resulting in discontinuities at the cell interfaces. Hence $f^c \cdot n$ and $f^v \cdot n$ must be approximated by numerical fluxes $\hat{f^c}$ and $\hat{f^v}$ in the surface integral of (2.24). The obvious central discretization for the convective terms

$$\hat{f^c_{ij}} = \frac{1}{2}\left(f^c(W_i) + f^c(W_j)\right) \cdot n_{ij}, \tag{2.26}$$

is unstable because the stencil of the scheme at node i does not include node i itself, resulting in the decoupling of neighbouring points. The instability may be shown by applying Fourier analysis for regular structured grids (Jameson, 1995), or using numerical experiments on unstructured grids.

A stable method may be achieved by adding artifical dissipation terms to the central discretization (Section 2.6), or by setting the numerical flux to be an approximate Riemann problem solver (also called *upwind fluxes*) (Radespiel & Kroll, 1995; Quirk, 1994), such as van Leer (Van Leer, 1982) or Roe (Roe, 1986), which may be seen to implicitly contain dissipation terms. In the latter case, the method as described is 1st-order accurate in Δx the grid spacing; this can be increased to Δx^2 by calculating an approximation to the gradients of the flow quantities at the nodes, and reconstructing values onto the cell faces using these gradients.

Given that the method is implemented conservatively, and that numerical viscosity is present, shocks are captured by the scheme automatically. On the other hand for normally 2nd-order accurate methods in space and higher, the order must be reduced near shocks to avoid solution oscillations. This takes the form of slope limiters for the upwind fluxes, and mixed 2nd- and 4th-order dissipation for central fluxes.

Discretization of the viscous terms is not as problematic, see Section 2.8, and the turbulence model equations are typically only discretized to $\mathcal{O}(\Delta x)$, though their source terms can cause stability problems.

At this point some unstructured grid terminology must be introduced: in the following a quantity with a single index, e.g. $|\Omega_i|$, indicates a quantity evaluated at a node (in this case the volume of the cell of node i), and a quantity with a double

[2]And consequently the fluxes are assumed to be constant on the individual faces of the control-volumes.

index, e.g. n_{ij}, indicates a quantity evaluated on the grid face connecting two nodes (in this case the normal vector of said face). The set $\mathcal{N}(i)$ of neighbours of i contains indices of all control-volumes that share a face with node i. Similarly $\mathcal{B}(i)$ contains indices of all faces of i that lie on the boundary of the computational domain, and is the empty set if i is not a boundary point. Sometimes it is useful to consider the first node that lies normal to the boundary at point i. The set $\mathcal{B}_{\text{near}}(i)$ contains this point if i is on the boundary, and is otherwise the empty set. The stencil of the discrete residual R is denoted $\mathcal{M}$.

Henceforth the indices i, j, k, etc. refer to the indices of grid nodes/control-volumes, unless otherwise stated.

2.5 Construction of the Jacobian

In addition to constructing the discrete residual of the scheme R, the Jacobian of the discrete residual $\partial R/\partial W$ is required for implicit methods.

Consider the structure of the Jacobian. R is a vector of size $n \times N$, where n is the number of nodes in the grid, and N is the number of equations per node. In principle R may be a function of all W, and W is a vector of the same size as R. Then $\partial R/\partial W$ is a matrix with dimensions $(n \times N) \times (n \times N)$. The structure of the matrix is dependent on the ordering of the degrees of freedom. It is most convenient to consider orderings of the form

$$(\rho_0, u_0, v_0, p_0, \rho_1, u_1, \cdots, \rho_{n-1}, u_{n-1}, v_{n-1}, p_{n-1}),$$

in which case the Jacobian may be written as an $n \times n$ matrix of $N \times N$ blocks. Then the notation $\partial R_i/\partial W_j$ refers to the block matrix obtained by differentiating the N components of R_i with respect to the N components of W_j.

However, R_i is not a function of W_j for all j, but only of a small number of W_j in the vicinity of i. The set of such j corresponds to the *stencil* of R_i, denoted $\mathcal{M}(i)$, which is almost always either (a) only i, (b) i and the immediate neighbours of i, or (c) i and the immediate and next-neighbours of i. Figure 2.1 shows these sets for a simple grid.

If a point j is not in the stencil of R_i then $\partial R_i/\partial W_j \equiv 0$, otherwise it is non-zero. Hence the Jacobian is sparse and the amount of fill-in is determined by the size of $\mathcal{M}$. The Jacobian is typically very large, even accounting for its sparsity, and therefore computationally intensive to calculate and store. The problem of efficient handling of the Jacobian is the main issue in algorithms involving it, and hence forms one of the principal themes of this thesis.

To see the importance of stencil size consider Table 2.2 which gives the number of immediate neighbours and next-neighbours for several grid types. In three dimensions the stencil size increases by at least a factor of four between neighbours and next-neighbours, resulting in a corresponding increase in the fill-in of the Jacobian.

Remark 2.1. *It is often the case that we consider the contribution of a flux over a face to the Jacobian. If the flux $\hat{f}_{ij}$ has a stencil of $\{i, j\}$ only, then contributions are made to the Jacobian at four points. Consider the case that the flux modifies R as*

$$\begin{aligned} R_i &:= R_i + \hat{f}_{ij}, \\ R_j &:= R_j - \hat{f}_{ij}, \end{aligned}$$

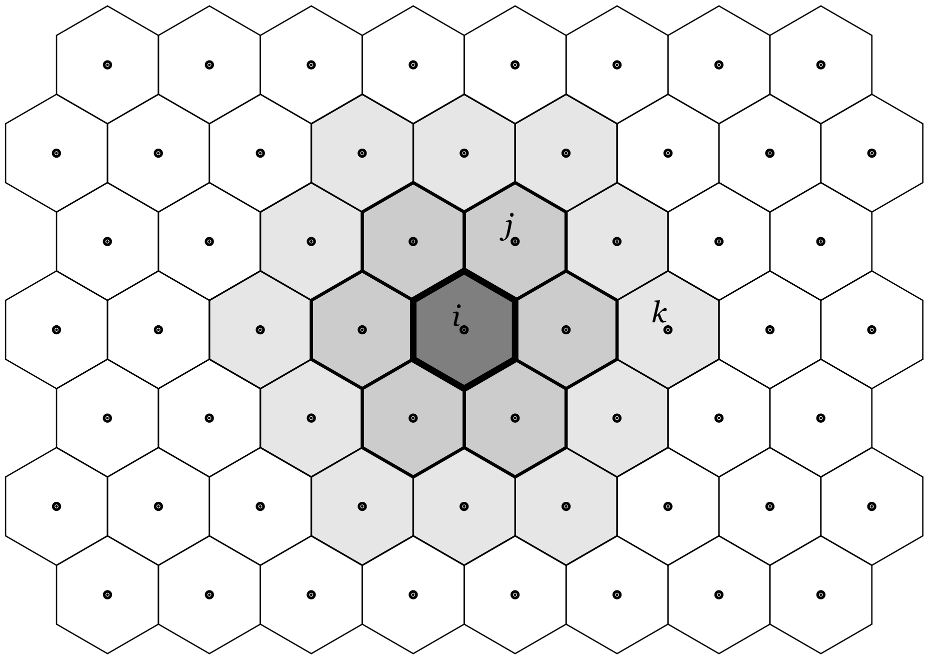


Figure 2.1: A stylized example of an unstructured dual-grid resulting from the Voronoï volumes of a primary grid of equilateral triangles. The nodes of the dual volumes are the same as the nodes of the primary grid. The shaded control-volumes show possible stencils of parts of the residual R calculated at the point i. For example the stencil of a 1st-order upwind flux includes i and the immediate neighbours of i, while the stencil of a 2nd-order upwind flux includes additionally the next-neighbours of i, e.g. k.

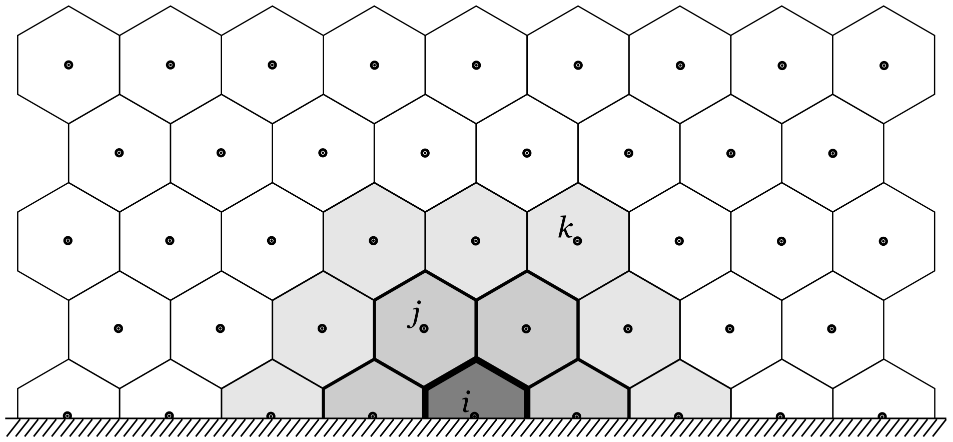


Figure 2.2: As for Figure 2.1, but in the region of a boundary. Important to note is that some nodes lie directly on the boundary. The values of the flow quantities at these nodes are taken to represent the state on the boundary (e.g. for a no-slip wall, zero velocity).

Dimensions	Grid Type	Neighbours	Next-neighbours
2	Structured	5	13
2	Unstructured	7	19
3	Structured	7	33
3	Semi-Structured	9	35
3	Unstructured	15	≈77

Table 2.2: The number of nodes neighbouring any given node on regular 2d and 3d grids. Here "Structured" indicates a regular square or hexahedral mesh, "Unstructured" a regular triangular or tetrahedral mesh, and "Semi-Structured" a mesh of triangular prisms. The node counts are inclusive: "Neighbours" includes the point itself and "Next-neighbours" includes neighbours. The values provide an indicator of the relative sizes of Jacobians on the various meshes, with the two stencil sizes.

then it modifies the Jacobian as

$$\frac{\partial R}{\partial W} := \frac{\partial R}{\partial W} + \begin{bmatrix} \ddots & & & & \\ & \frac{\partial \hat{f}_{ij}}{\partial W_i} & \cdots & \frac{\partial \hat{f}_{ij}}{\partial W_j} & \\ & \vdots & \ddots & \vdots & \\ & -\frac{\partial \hat{f}_{ij}}{\partial W_i} & \cdots & -\frac{\partial \hat{f}_{ij}}{\partial W_j} & \\ & & & & \ddots \end{bmatrix}. \tag{2.27}$$

2.6 Central Convective Fluxes

The most commonly used convective flux in *TAU* is a central flux with blended 2nd- and 4th-*undivided differences* representing 2nd- and 4th-order artificial dissipation, and is an unstructured generalization of the well-known Jameson-Schmitt-Turkel (JST) scheme (Jameson *et al.*, 1981). The term *undivided difference* simply refers to a standard finite difference approximation to a derivative, but without a denominator; for example the LHS of

$$W_{i+1} - 2W_i + W_{i-1} \approx \Delta x^2 \frac{\mathrm{d}^2 W}{\mathrm{d}x^2}.$$

The 4th-order dissipation is used in the majority of the field as the dissipation terms are of order Δx^3 and therefore do not detract from the order of accuracy of the method, which is dominated by the error incurred when approximating the fluxes as constant on faces (Δx^2). However the operator is unstable at discontinuities and introduces overshoots, so 2nd-order dissipation terms of order Δx are used there instead. This reduction of order means it is important to have higher grid resolution near shocks than elsewhere. The detection of discontinuities is performed with a pressure gradient sensor.

The scheme may be written

$$\hat{f}_{ij}^{\mathrm{JST}} = \frac{1}{2}\left(f^c(W_i) + f^c(W_j)\right) \cdot n_{ij} - \frac{1}{2}\bar{D}_{ij}, \tag{2.28}$$

where f^c are the exact convective fluxes as given in (2.4), and $\bar{D}$ contains the dissipation terms. The derivatives of $f^c \cdot n$ are given in various variables in Appendix A.

2.6.1 Scalar Dissipation for the Central Scheme

We give the exact form of the dissipation $\bar{D}$ of (2.28) as implemented in *TAU*. For a control-volume i the total contribution to the residual R is

$$D_i = \sum_{j \in \mathcal{N}(i)} \bar{D}_{ij}, \tag{2.29}$$

whereby the dissipation on a face is a sum of 1st- and 3rd-undivided differences

$$\bar{D}_{ij} = \lambda^c_{ij} \left[\bar{\varepsilon}^{(2)}_{ij} (W_j - W_i) - \bar{\varepsilon}^{(4)}_{ij} (L_j(W) - L_i(W)) \right], \tag{2.30}$$

and $\bar{\varepsilon}^{(2)}$ and $\bar{\varepsilon}^{(4)}$ control the levels of the two types of dissipation, which themselves consist of three terms:

$$\bar{\varepsilon}^{(2)}_{ij} = \varepsilon^{(2)}_{ij} \, s^{c2}_{ij} \, \bar{\phi}_{ij}, \tag{2.31}$$

$$\bar{\varepsilon}^{(4)}_{ij} = \varepsilon^{(4)}_{ij} \, s^{c4}_{ij} \, \bar{\phi}_{ij}. \tag{2.32}$$

The $\varepsilon^{(2)}$ and $\varepsilon^{(4)}$ act as the shock switch, s^{c2} and s^{c4} are intended to make the level of dissipation independent of the number of neighbours of a cell, and $\bar{\phi}$ is intended to increase the amount of dissipation across the larger faces of anisotropic cells and decrease it across the smaller faces. In particular

$$\bar{\phi}_{ij} = \frac{4 \phi^{(i)}_{ij} \phi^{(j)}_{ji}}{\phi^{(i)}_{ij} + \phi^{(j)}_{ji} + \epsilon}, \tag{2.33}$$

where $\epsilon = 10^{-16}$ is a constant chosen simply to prevent a divide-by-zero condition in the arithmetic. The $\phi^{(i)}_{ij}$ are defined by

$$\phi^{(i)}_{ij} = \left(\frac{\max_0 \left(\frac{1}{2} \lambda^t_i - \lambda^c_{ij} \right)}{2 \lambda^c_{ij}} \right)^{\frac{1}{2}}, \tag{2.34}$$

where $\max_0(\cdot) = \max(\cdot, 0)$ and λ^t_i is the sum of the maximum convective eigenvalues over all faces of volume i. The maximum convective eigenvalue denotes is defined in (2.45). Given that $\phi^{(i)}_{ij}$ and $\phi^{(j)}_{ji}$ are approximately the same - which is the case in the absence of rapid changes in cell size - (2.33) reduces to $\bar{\phi} \approx 2\phi^{(i)}_{ij}$. Then (2.34) causes the dissipation over the face of a cell with the larger eigenvalue λ^c to be relatively increased, and that with the smaller eigenvalue to be decreased. In particular, in anisotropic boundary-layer cells, the eigenvalue of the long side dominates that of the short side and so extra dissipation normal to the wall is included.

The expressions chosen in order to attempt to remove dependence on the number of faces of a control volume are

$$s^{c2}_{ij} = \frac{3(N_i + N_j)}{N_i N_j}, \tag{2.35}$$

$$s^{c4}_{ij} = \frac{9\left[(1 + N_i)N_i + (1 + N_j)N_j\right]}{(1 + N_i)N_i(1 + N_j)N_j}, \tag{2.36}$$

where N_i is the number of faces of cell i,

$$N_i = \sum_{j\in\mathcal{N}(i)} (1) + \sum_{b\in\mathcal{B}(i)} (1). \tag{2.37}$$

The coefficients $\varepsilon^{(2)}$ and $\varepsilon^{(4)}$ are more familiar, being taken directly from Jameson (Jameson *et al.*, 1981),

$$\begin{aligned} \varepsilon_{ij}^{(2)} &= k^{(2)}\max(\Psi_i, \Psi_j), & (2.38)\\ \varepsilon_{ij}^{(4)} &= \max{}_0(k^{(4)} - \varepsilon_{ij}^{(2)}), & (2.39)\end{aligned}$$

where $k^{(2)}$ and $k^{(4)}$ are constants allowing specification of absolute levels of dissipation, typically 1/2 and 1/64 respectively, and the remaining terms control the relative levels of 2nd- and 4th-dissipation using the estimate of the pressure gradient

$$\Psi_i = \left|\frac{p_i^{\text{dif}}}{p_i^{\Sigma}}\right|, \tag{2.40}$$

where

$$\begin{aligned} p_i^{\Sigma} &= \sum_{j\in\mathcal{N}(i)} (p_i + p_j) + \sum_{m\in\mathcal{B}_{\text{near}}(i)} (3p_i - p_m), & (2.41)\\ p_i^{\text{dif}} &= \sum_{j\in\mathcal{N}(i)} (p_j - p_i) + \sum_{m\in\mathcal{B}_{\text{near}}(i)} (p_i - p_m), & (2.42)\end{aligned}$$

so that for a smooth solution Ψ and so $\varepsilon^{(2)}$ are of order Δx^2, while at a shock both are of order unity.

The 3rd-difference is constructed as a difference of two 2nd-undivided differences,

$$L_i(W) = \sum_{j\in\mathcal{N}(i)} (W_j - W_i) + \sum_{m\in\mathcal{B}_{\text{near}}(i)} (W_i - W_m), \tag{2.43}$$

where the use of the boundary near points here is intended to avoid asymmetry of the Laplacian on boundaries. The total maximum eigenvalue for a cell is

$$\lambda_i^t = \sum_{j\in\mathcal{N}(i)} \lambda_{ij}^c + \sum_{b\in\mathcal{B}(i)} \lambda_b^c, \tag{2.44}$$

whereby the maximum eigenvalues on the faces are

$$\begin{aligned} \lambda_{ij}^c = \max_m \left[\lambda_m\left(\frac{\partial f_{ij}^c}{\partial W}\right)\right] &= \tfrac{1}{2}|(U_i + U_j)\cdot n_{ij}| + \tfrac{1}{2}(a_i + a_j)\|n_{ij}\|, & (2.45)\\ \lambda_b^c = \max_m \left[\lambda_m\left(\frac{\partial f_b^c}{\partial W_b}\right)\right] &= |U_b\cdot n_b| + a_b\|n_b\|, & (2.46)\end{aligned}$$

where $\lambda_m(\cdot)$ returns the mth eigenvalue of the matrix argument, thus completing the scheme.

Remark 2.2. *This scheme is derived from the JST method (Jameson* et al.*, 1981) which has proven extremely effective on structured grids. The chief difficulty in the extension to unstructured grids is the unidirectional nature of the coefficients of the dissipation in the original scheme. For example the pressure sensor given by Jameson to construct the shock switch is unidirectional,*

$$\Psi_{IJ} = \frac{|p_{I+1,J} - 2p_{I,J} + p_{I-1,J}|}{|p_{I+1,J} + 2p_{I,J} + p_{I-1,J}|}, \tag{2.47}$$

and repeated in each coordinate direction; here I and J are the structured grid cell indices. Equation (2.40) *is an attempt to model this expression without direction information. Similarly* (2.33) *is an attempt to reproduce the commonly used structured grid anisotropic cell scaling*

$$\bar{\phi}_{I+\frac{1}{2},J,K} = 1 + \max\left(\frac{\lambda^c_{I,J+\frac{1}{2},K}}{\lambda^c_{I+\frac{1}{2},J,K}}, \frac{\lambda^c_{I,J,K+\frac{1}{2}}}{\lambda^c_{I+\frac{1}{2},J,K}}\right)^{\frac{1}{2}}, \tag{2.48}$$

in three dimensions, where e.g. $\lambda^c_{I+\frac{1}{2},J,K}$ is the average of the eigenvalues of the faces of cell I, J, K with face normals pointing in the I direction. The coefficients s^{c2} and s^{c4} have no equivalent in the structured scheme and are chosen such that the unstructured scheme on a regular hexahedral dual grid has the same level of dissipation as the structured scheme on a regular structured grid.

2.6.2 Jacobian of Dissipation under a Constant Coefficient Approximation

As seen in Section 2.6.1 the full dissipation operator is rather complex, and the exact derivatives thereof are therefore also very complex. By assuming that the derivatives of the coefficients of the difference operators in the scheme - namely $\bar{\varepsilon}^{(2)}$, $\bar{\varepsilon}^{(4)}$ and λ^c - may be treated as constants with respect to W, a considerable simplification in the Jacobian is achieved.

Remark 2.3. *We attempt to justify this approximation: consider the relative magnitudes of the terms that are neglected in the derivative, and the remaining terms. For concise presentation consider only the second difference operator without the shock switch,*

$$\bar{D}^{\text{2nd}}_{ij} = \lambda^c_{ij}(W_j - W_i). \tag{2.49}$$

The full derivative of this may then be written

$$\frac{\partial \lambda^c_{ij}}{\partial W_k}(W_j - W_i) + \lambda^c_{ij}\frac{\partial}{\partial W_k}(W_j - W_i), \tag{2.50}$$

however, under the assumption $\lambda^c = \text{const.}$, only the second term appears. Consider the magnitude of these quantities in terms of Δx the grid spacing.

If $k \notin \{i, j\}$ then all derivatives in (2.50) *are zero, so consider the case $k \in \{i, j\}$. For a smooth solution $(W_j - W_i)$ is of order Δx, whereas its derivative is of order unity. Also λ^c_{ij} and its derivate are always order Δx (due to the presence of the face normal n). Therefore the first term in* (2.50) *has order Δx^2 and the second term* *order Δx. Hence the first term may be neglected for smooth solutions on sufficiently fine grids, and the approximation $\lambda^c = \text{const.}$ is justified.*

Extending this argument to the full dissipation operator, is complicated by the differences present in p^{dif}, whose derivatives are also of order unity, but a similar result is eventually achieved. Experiences using the approximate Jacobian in Chapter 5 bear out the conclusions given here.

The derivatives may then be written

$$\frac{\partial D_i}{\partial W_k} = \sum_{j\in\mathcal{N}(i)} \frac{\partial \bar{D}_{ij}}{\partial W_k} \tag{2.51}$$

$$= \sum_{j\in\mathcal{N}(i)} \lambda^c_{ij}\left[\bar{\varepsilon}^{(2)}_{ij}\frac{\partial}{\partial W_k}(W_j - W_i) - \bar{\varepsilon}^{(4)}_{ij}\frac{\partial}{\partial W_k}(L_j(W) - L_i(W))\right], \tag{2.52}$$

whereby choosing the conservative variables for the differentiation pays off in a particularly simple form for the difference derivatives:

$$\frac{\partial}{\partial W_k}(W_j - W_i) = \begin{cases} -I & k = i \\ I & k = j \\ 0 & \text{otherwise} \end{cases}, \tag{2.53}$$

where I is the identity matrix, so that the derivative of their sum is

$$\frac{\partial}{\partial W_k}\sum_{j\in\mathcal{N}(i)}(W_j - W_i) = \begin{cases} \sum_{j\in\mathcal{N}(i)}(-I) & k = i \\ I & k \in \mathcal{N}(i) \\ 0 & \text{otherwise} \end{cases}. \tag{2.54}$$

The pseudo-Laplacian derivatives are similarly

$$\frac{\partial L_i(W)}{\partial W_k} = \begin{cases} \sum_{j\in\mathcal{N}(i)}(-I) + \sum_{m\in\mathcal{B}_{\text{near}}(i)}(I) & k = i \\ I & k \in \mathcal{N}(i) \cap \overline{\mathcal{B}_{\text{near}}(i)} \\ 0 & \text{otherwise} \end{cases}, \tag{2.55}$$

and the expression for the Jacobian is complete.

Comparing this result with the exact Jacobian given in Section 2.6.3 highlights the enormous potential benefits of well-chosen approximations.

2.6.3 Full Jacobian of Scalar Dissipation

The full expression for the exact Jacobian of the scalar dissipation operator is given in the following. Note that extensive use is made of the chain rule to divide the operation into manageable parts. Each expression of Section 2.6.1 is differentiated in turn, writing the derivative in terms of derivatives of the other quantities. No attempt is made to collect terms in an effort to reduce the number of expressions. This helps reduce the likelihood of an error and allows the scheme derivative to be easily verified against the scheme statement.

Starting with the dissipation contribution to the residual at a node,

$$\frac{\partial D_i}{\partial W_k} = \sum_{j\in\mathcal{N}(i)} \frac{\partial \bar{D}_{ij}}{\partial W_k} \tag{2.56}$$

$$\begin{aligned} &= \sum_{j\in\mathcal{N}(i)} \left\{ \frac{\partial \lambda^c_{ij}}{\partial W_k} \left[\bar{\varepsilon}^{(2)}_{ij}(W_j - W_i) - \bar{\varepsilon}^{(4)}(L_j - L_i) \right] \right. \\ &+ \lambda^c_{ij} \left[\frac{\partial \bar{\varepsilon}^{(2)}_{ij}}{\partial W_k}(W_j - W_i) + \bar{\varepsilon}^{(2)}_{ij} \frac{\partial}{\partial W_k}(W_k - W_i) \right. \\ &- \left. \left. \frac{\partial \bar{\varepsilon}^{(4)}_{ij}}{\partial W_k}(L_j - L_i) - \bar{\varepsilon}^{(4)}_{ij} \frac{\partial}{\partial W_k}(L_j - L_i) \right] \right\}; \end{aligned} \tag{2.57}$$

comparing with (2.52) the extra effort required is already apparent. The individual fluxes are then

$$\frac{\partial \bar{\varepsilon}^{(2)}_{ij}}{\partial W_k} = s^{c2}_{ij} \left(\frac{\partial \varepsilon^{(2)}_{ij}}{\partial W_k}\bar{\phi}_{ij} + \varepsilon^{(2)}_{ij} \frac{\partial \bar{\phi}_{ij}}{\partial W_k} \right), \tag{2.58}$$

$$\frac{\partial \bar{\varepsilon}^{(4)}_{ij}}{\partial W_k} = s^{c4}_{ij} \left(\frac{\partial \varepsilon^{(4)}_{ij}}{\partial W_k}\bar{\phi}_{ij} + \varepsilon^{(4)}_{ij} \frac{\partial \bar{\phi}_{ij}}{\partial W_k} \right), \tag{2.59}$$

whereby

$$\frac{\partial \bar{\phi}_{ij}}{\partial W_k} = \frac{4\left[\left(\frac{\partial \phi^{(i)}_{ij}}{\partial W_k}\phi^{(j)}_{ji} + \phi^{(i)}_{ij}\frac{\partial \phi^{(j)}_{ji}}{\partial W_k} \right)(\phi^{(i)}_{ij} + \phi^{(j)}_{ji} + \epsilon) - \phi^{(i)}_{ij}\phi^{(j)}_{ji}\left(\frac{\partial \phi^{(i)}_{ij}}{\partial W_k} + \frac{\partial \phi^{(j)}_{ji}}{\partial W_k} \right) \right]}{(\phi^{(i)}_{ij} + \phi^{(j)}_{ji} + \epsilon)^2}. \tag{2.60}$$

Note that the ϵ used to prevent a divide-by-zero condition in the arithmetic of the flux, prevents this condition in the derivative as well.

The appearance of $\max(\cdot, \cdot)$ in the expression for $\phi^{(i)}_{ij}$ (and similarly, the appearance of $|\cdot|$ in the expression for Ψ), leads to the derivative of $\phi^{(i)}_{ij}$ being undefined at $(\frac{1}{2}\lambda^t_i - \lambda^c_{ij})$. This problem will be discussed further later; for the moment differentiate the function correctly where possible, and choose the limit from one side for the derivative at the discontinuity:

$$\frac{\partial \phi^{(i)}_{ij}}{\partial W_k} = \begin{cases} \frac{\partial}{\partial W_k}\left(\frac{\frac{1}{2}\lambda^t_i - \lambda^c_{ij}}{2\lambda^c_{ij}} \right)^{\frac{1}{2}} & \frac{1}{2}\lambda^t_i - \lambda^c_{ij} \geq 0 \\ 0 & \frac{1}{2}\lambda^t_i - \lambda^c_{ij} < 0 \end{cases}. \tag{2.61}$$

Experience shows that such effects are not harmful to the linearization. Continuing the derivation

$$\begin{aligned} \frac{\partial}{\partial W_k}\left(\frac{\frac{1}{2}\lambda^t_i - \lambda^c_{ij}}{2\lambda^c_{ij}} \right)^{\frac{1}{2}} &= \left(\frac{\left(\frac{1}{2}\frac{\partial \lambda^t_i}{\partial W_k} - \frac{\partial \lambda^c_{ij}}{\partial W_k} \right) 2\lambda^c_{ij} - \left(\frac{1}{2}\lambda^t_i - \lambda^c_{ij} \right) 2\frac{\partial \lambda^c_{ij}}{\partial W_k}}{(\lambda^c_{ij})^2} \right) \\ &\cdot \frac{1}{2}\left(\frac{\frac{1}{2}\lambda^t_i - \lambda^c_{ij}}{2\lambda^c_{ij}} \right)^{-\frac{1}{2}}, \end{aligned} \tag{2.62}$$

and the shock switch introduces another discontinuity,

$$\frac{\partial \varepsilon_{ij}^{(2)}}{\partial W_k} = k^{(2)} \frac{\partial}{\partial W_k}\Big[\max(\Psi_i, \Psi_j)\Big] = \begin{cases} k^{(2)} \frac{\partial \Psi_i}{\partial W_k} & \Psi_i > \Psi_j \\ k^{(2)} \frac{\partial \Psi_j}{\partial W_k} & \Psi_i \le \Psi_j \end{cases}, \tag{2.63}$$

$$\frac{\partial \varepsilon_{ij}^{(4)}}{\partial W_k} = \frac{\partial}{\partial W_k}\Big[\max_0(k^{(4)} - \varepsilon_{ij}^{(2)})\Big] = \begin{cases} -\frac{\partial \varepsilon_{ij}^{(2)}}{\partial W_k} & k^{(4)} - \varepsilon_{ij}^{(2)} > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{2.64}$$

and again,

$$\frac{\partial \Psi_i}{\partial W_k} = \frac{\partial}{\partial W_k}\left|\frac{p_i^{\text{dif}}}{p_i^{\Sigma}}\right| = \begin{cases} \frac{\partial}{\partial W_k}\left(\frac{p_i^{\text{dif}}}{p_i^{\Sigma}}\right) & \frac{p_i^{\text{dif}}}{p_i^{\Sigma}} \ge 0 \\ -\frac{\partial}{\partial W_k}\left(\frac{p_i^{\text{dif}}}{p_i^{\Sigma}}\right) & \frac{p_i^{\text{dif}}}{p_i^{\Sigma}} < 0 \end{cases}, \tag{2.65}$$

whereby

$$\frac{\partial}{\partial W_k}\left(\frac{p_i^{\text{dif}}}{p_i^{\Sigma}}\right) = \frac{\frac{\partial p_i^{\text{dif}}}{\partial W_k} p_i^{\Sigma} + p_i^{\text{dif}} \frac{\partial p_i^{\Sigma}}{\partial W_k}}{(p_i^{\Sigma})^2}. \tag{2.66}$$

The derivatives of the pressure differences are comparatively straightforward,

$$\frac{\partial p_i^{\Sigma}}{\partial W_k} = \begin{cases} \sum_{j\in\mathcal{N}(i)}(\frac{\partial p_i}{\partial W_i}) + \sum_{m\in\mathcal{B}_{\text{near}}(i)} 3\frac{\partial p_i}{\partial W_i} & k = i \\ \frac{\partial p_k}{\partial W_k} & k \in \mathcal{N}(i) \cap \overline{\mathcal{B}_{\text{near}}(i)} \\ 0 & \text{otherwise} \end{cases}, \tag{2.67}$$

$$\frac{\partial p_i^{\text{dif}}}{\partial W_k} = \begin{cases} \sum_{j\in\mathcal{N}(i)}(-\frac{\partial p_i}{\partial W_i}) + \sum_{m\in\mathcal{B}_{\text{near}}(i)} \frac{\partial p_i}{\partial W_i} & k = i \\ \frac{\partial p_k}{\partial W_k} & k \in \mathcal{N}(i) \cap \overline{\mathcal{B}_{\text{near}}(i)} \\ 0 & \text{otherwise} \end{cases}, \tag{2.68}$$

and can be further simplified by performing the differentiation in primitive variables. The pseudo-Laplacians are exactly as in Section 2.6.2

$$\frac{\partial L_i(W)}{\partial W_k} = \begin{cases} \sum_{j\in\mathcal{N}(i)}(-I) + \sum_{m\in\mathcal{B}_{\text{near}}(i)}(I) & k = i \\ I & k \in \mathcal{N}(i) \cap \overline{\mathcal{B}_{\text{near}}(i)} \\ 0 & \text{otherwise} \end{cases}, \tag{2.69}$$

and finally the derivatives of the maximum eigenvalues on the faces are

$$\frac{\partial \lambda_{ij}^c}{\partial W_k} = \begin{cases} \frac{1}{2}\frac{\partial}{\partial W_k}\left[(U_i + U_j)\cdot n_{ij} + (a_i + a_j)\|n_{ij}\|\right] & (U_i + U_j)\cdot n_{ij} \ge 0, \\ \frac{1}{2}\frac{\partial}{\partial W_k}\left[-(U_i + U_j)\cdot n_{ij} + (a_i + a_j)\|n_{ij}\|\right] & (U_i + U_j)\cdot n_{ij} < 0, \end{cases} \tag{2.70}$$

whereby

$$\frac{\partial}{\partial W_k}\left[\tfrac{1}{2}(U_i + U_j)\cdot n_{ij} + \tfrac{1}{2}(a_i + a_j)\|n_{ij}\|\right] = \begin{cases} \frac{1}{2}\frac{\partial U_i\cdot n_{ij}}{\partial W_i} + \frac{1}{2}\frac{\partial a_i}{\partial W_i}\|n_{ij}\| & k = i \\ \frac{1}{2}\frac{\partial U_j\cdot n_{ij}}{\partial W_j} + \frac{1}{2}\frac{\partial a_j}{\partial W_j}\|n_{ij}\| & k = j \\ 0 & \text{otherwise} \end{cases},$$

$$\frac{\partial}{\partial W_k}\left[-\tfrac{1}{2}(U_i + U_j)\cdot n_{ij} + \tfrac{1}{2}(a_i + a_j)\|n_{ij}\|\right] = \begin{cases} \text{similarly,} \end{cases}$$

completing the expression for the Jacobian.

2.7 Gradient Approximation

In the construction of the viscous fluxes and turbulence source terms, the gradients of the flow quantities in space are needed. These may be obtained by a least-squares method, i.e. fitting a plane to a local collection of nodes, but a particularly elegant and efficient gradient on a general grid is obtained with the Gauss integral theorem (Blazek, 2001). Consider the identity

$$\int_{\Omega_i} \nabla W_i \,\mathrm{d}\Omega = \oint_{\partial\Omega_i} W_i \, n \cdot \mathrm{d}(\partial\Omega), \tag{2.71}$$

whereby approximating these integrals numerically gives

$$\Omega_i \nabla W_i^{GG} := \sum_{j\in\mathcal{N}(i)} \frac{1}{2}\left(W_i + W_j\right) n_{ij} + \sum_{m\in\mathcal{B}(i)} W_i \, n_m, \tag{2.72}$$

where including the integral over boundary faces ensures that the approximate surface integral is closed. This procedure is known as the *Green-Gauss* method for the gradient.

2.7.1 Green-Gauss Jacobian

Another advantage of Green-Gauss gradients is that their derivatives are particularly simple. In particular

$$\Omega_i \frac{\partial}{\partial W_k} \nabla W_i^{GG} = \begin{cases} \sum_{j\in\mathcal{N}(i)} \frac{1}{2} n_{ij} + \sum_{m\in\mathcal{B}(i)} n_m & k = i \\ \frac{1}{2} n_{ik} & k \in \mathcal{N}(i) \\ 0 & \text{otherwise} \end{cases}. \tag{2.73}$$

Here it is also apparent that the stencil of the gradient consists of immediate neighbours only.

2.8 Viscous Flux Modelling

The modelling of viscous fluxes is not as critical as that of the convective fluxes, as there are no related stability problems. The exact expressions for the viscous fluxes are used to model the fluxes on cell faces, and therefore all that is required are the values of the flow variables and their gradients on the face. The flow variables are always averaged from the two neighbouring cells,

$$U_{ij} := \frac{1}{2}\left(U_i + U_j\right), \tag{2.74}$$

but there are several approaches to obtain the gradient on the face.

Most simply the average of the gradients in the neighbouring cells are taken,

$$\overline{\nabla U_{ij}} := \frac{1}{2}\left(\nabla U_i + \nabla U_j\right), \tag{2.75}$$

where ∇U is approximated by Green-Gauss, Section 2.7. On the other hand it seems clear that the most accurate and stable approximation to the gradient normal to the face is the difference

$$(\nabla U_{ij} \cdot n_{ij}) \approx \frac{U_j - U_i}{\|x_j - x_i\|}, \tag{2.76}$$

where x_i is the coordinate of node i, so that a better full gradient approximation might be

$$\widetilde{\nabla U_{ij}} = \frac{U_j - U_i}{\|x_j - x_i\|}\widetilde{\Delta x} + \left\{\overline{\nabla U_{ij}} - \left(\overline{\nabla U_{ij}} \cdot \widetilde{\Delta x}\right) \widetilde{\Delta x}\right\}, \tag{2.77}$$

where $\widetilde{\Delta x} = (x_j - x_i)/\|x_j - x_i\|$.

Both these expressions for the gradient have the disadvantage of having stencils consisting of all immediate neighbours of both i and j - leading to a next-neighbour fill-in in the viscous Jacobian. On the other hand by neglecting completely face-tangential gradient components we have

$$\nabla U_{ij}^{\mathrm{TSL}} \approx \frac{U_j - U_i}{\|x_j - x_i\|}, \tag{2.78}$$

which also has a simple expressions for its derivatives. Viscous fluxes based on this gradient will be denoted the *Thin Shear-Layer* (TSL) fluxes, due to their similarity to a method in structured codes in which only viscous fluxes normal to the wall are considered. Here, however, fluxes in all directions are considered.

It may be shown that the use of the TSL gradient results in a consistent viscous flux discretization, and numerical tests show that the influence on the solution is very minor, even for cases sensitive to viscous effects such as high-lift configurations. Further, it is used in some unstructured codes, on the basis that it improves robustness (Mavriplis, 1998).

2.8.1 TSL Viscous Flux Derivatives

The numerical viscous fluxes for a given face in conservative variables are

$$\hat{f}_{ij}^{v} = \begin{pmatrix} 0 \\ n_l \tau_{lx} \\ n_l \tau_{ly} \\ n_l \tau_{lz} \\ n_l(\tau_{lm} U_m + q_l) \end{pmatrix},$$

where τ and q are given in Section 2.2.1. If TSL gradients are used, derivatives of μ and κ are neglected, and if further the differentiation is performed with respect to the alternative primitive variables, Appendix A.2, then the Jacobian takes a particularly simple form. With respect to the alternative primitive variables at nodes i and j

respectively it is

$$\frac{\partial \hat{f}^v_{ij}}{\partial W_{i/j}} = \mp\frac{\mu}{\Delta x}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \theta_x & \eta_z & \eta_y & 0 \\ 0 & \eta_z & \theta_y & \eta_x & 0 \\ 0 & \eta_y & \eta_x & \theta_z & 0 \\ \phi^{\pm}_\rho\theta & \frac{\mp\Delta x}{2\mu}n_l\tau_{lx}+\pi_x & \frac{\mp\Delta x}{2\mu}n_l\tau_{ly}+\pi_y & \frac{\mp\Delta x}{2\mu}n_l\tau_{lz}+\pi_z & \phi^{\pm}_p\theta \end{pmatrix}, \tag{2.79}$$

where

$$\theta = n_x^2 + n_y^2 + n_z^2$$

$$\theta_x = \frac{4}{3}n_x^2 + n_y^2 + n_z^2, \qquad \theta_y = n_x^2 + \frac{4}{3}n_y^2 + n_z^2, \qquad \theta_z = n_x^2 + n_y^2 + \frac{4}{3}n_z^2,$$

$$\eta_x = \frac{1}{3}n_y n_z, \qquad \eta_y = \frac{1}{3}n_x n_z, \qquad \eta_z = \frac{1}{3}n_x n_y,$$

and further

$$\phi^+_\rho = -\frac{\kappa T_L}{\mu\rho_L}, \qquad \phi^-_\rho = -\frac{\kappa T_R}{\mu\rho_R},$$
$$\phi^+_p = \frac{\kappa}{\mu\rho_L}, \qquad \phi^-_p = \frac{\kappa}{\mu\rho_R},$$

$$\begin{aligned} \pi_x &= u\theta_x + v\eta_z + w\eta_y, \\ \pi_y &= u\eta_z + v\theta_y + w\eta_x, \\ \pi_z &= u\eta_y + v\eta_x + w\theta_z. \end{aligned}$$

2.9 Solid Wall Boundary Conditions

With regard to numerical implementation there are two fundamental types of boundary conditions to be considered, weak and strong. The type also has a significant effect on the manner in which the Jacobian must be formulated. Note that all boundary conditions described in this thesis are specific to discretizations with control points lying on the boundary, i.e. typically cell-vertex methods.

2.9.1 Slip-Wall

A numerically weak boundary condition is here defined as a boundary condition imposed by setting only the flux over the boundary face. As an example take the slip-wall condition, which demands that the velocity component normal to the wall must be zero. This is imposed using a flux of

$$\hat{f}^{\text{s-w}}_b = \begin{pmatrix} 0 \\ p_b\, n_{b,x} \\ p_b\, n_{b,y} \\ p_b\, n_{b,z} \\ 0 \end{pmatrix}, \tag{2.80}$$

over the slip-wall faces with associated node b.

2.9.2 Slip-Wall Jacobians

The Jacobian due to this flux written in conservative variables is simply

$$\frac{\partial \hat{f}_b^{\text{s-w}}}{\partial W_c} = \frac{\partial \hat{f}_b^{\text{s-w}}}{\partial W_p} \cdot \frac{\partial W_p}{\partial W_c} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & n_{b,x} \\ 0 & 0 & 0 & 0 & n_{b,y} \\ 0 & 0 & 0 & 0 & n_{b,z} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot \frac{\partial W_p}{\partial W_c}, \tag{2.81}$$

where initially computing the derivatives with respect to the primitive variables W_p, simplifies the calculation considerably. Note that this Jacobian appears on the block diagonal of the full Jacobian, as the flux makes a contribution to the residual at node b, using only flow quantities from node b. The stencil therefore consists of only the node itself, and hence implementation is particularly simple.

2.9.3 No-Slip Wall

A numerically strong boundary condition is here defined as a condition imposed by setting the values of the flow variables on the boundary directly. For example a no-slip isothermal wall condition, which demands that the flow velocity at the wall is zero, and the temperature is a given constant, may be numerically enforced simply by setting the velocity vector on the surface to zero, and the temperature to the specified value.

2.9.4 No-Slip Wall Jacobians

Strong boundary conditions represent a challenge when building the full Jacobian of the scheme. When the velocity vector is set to zero for a node on a viscous wall, the momentum components of all fluxes passing into that boundary cell are discarded. This alters the Jacobians of these fluxes, and so it is not only the diagonal of the Jacobian that must be modified for this condition. This holds for all strongly implemented boundary conditions.

The situation may be clarified by rewriting the equation system $R(W) = 0$ in the form

$$(I - B) \cdot R(W) = 0, \quad B \cdot W = 0, \tag{2.82}$$

where B is a projection matrix that extracts the component of the residual to be discarded, which in the case of the no-slip wall is the boundary velocity (Giles *et al.*, 2003).

By considering a small perturbation ΔW, (2.82) may be linearized as

$$(I - B) \cdot \frac{\partial R}{\partial W} \cdot \Delta W = 0, \quad B \cdot \Delta W = 0, \tag{2.83}$$

which formally specifies that all derivatives of the momentum components of the boundary cell residual are zero, and that all local perturbations on the boundary are disallowed, $B \cdot \Delta W = 0$. Note that the first condition implies that the Jacobian will be singular, as

$$\frac{\partial R_b^{\rho U}}{\partial W_i} = 0, \quad \forall i, \tag{2.84}$$

where $R^{\rho U}$ are the momentum components of the residual, so the Jacobian will have three rows of zeros for each viscous wall node. This is the direct result of having three fewer degrees of freedom in the problem for each no-slip boundary node, while maintaining the same number of equations.

Invertability may be recovered either by removing the superfluous degrees of freedom, or by merging the two equations of (2.83) into

$$\Big[B + (I - B) \cdot R\Big] \cdot \Delta W = 0, \tag{2.85}$$

whereupon a diagonal block of the Jacobian lying on a no-slip wall has the form

$$\frac{\partial R_b}{\partial W_b} = \begin{pmatrix} \frac{\partial R_b^\rho}{\partial \rho_b} & \frac{\partial R_b^\rho}{\partial (\rho u)_b} & \frac{\partial R_b^\rho}{\partial (\rho v)_b} & \frac{\partial R_b^\rho}{\partial (\rho w)_b} & \frac{\partial R_b^\rho}{\partial (\rho E)_b} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{\partial R_b^{\rho E}}{\partial \rho_b} & \frac{\partial R_b^{\rho E}}{\partial (\rho u)_b} & \frac{\partial R_b^{\rho E}}{\partial (\rho v)_b} & \frac{\partial R_b^{\rho E}}{\partial (\rho w)_b} & \frac{\partial R_b^{\rho E}}{\partial (\rho E)_b} \end{pmatrix}. \tag{2.86}$$

Further since ΔW on a no-slip wall must be zero, $\partial R/\partial(\rho U)_b$ may also always consistently be taken to be zero.

2.10 Permeable Boundary Conditions

The computational domain is a finite grid, whereas a typical problem in aerodynamics is an aerofoil moving at a constant speed through an infinite domain. Hence at the edges of the grid an external flow or *farfield* boundary condition must be imposed. At such a boundary some flow information is transported into the computational domain from the ambient flow, whilst other information leaves the domain. In order to specify a physically consistent boundary condition it is necessary to know exactly which flow quantities are transported in which direction. This information is obtained by considering characteristic theory for the compressible Euler equations. Viscous effects are small in the farfield, except possibly in the wake of a body, and are neglected.

The derivation of the characteristic variables will be performed with respect to the primitive variables to simplify the algebra. The Euler equations of (2.2), locally linearized with respect to a state W_0 are

$$\frac{\partial W_i}{\partial t} + A_{ij,k}(W_0) \frac{\partial W_j}{\partial x_k} = 0, \tag{2.87}$$

where the subscripts are no longer node indices, but tensor indices, and the summation convention has been used on j and k. In primitive variables

$$A \cdot \kappa = \begin{pmatrix} U \cdot \kappa & \kappa_x \rho & \kappa_y \rho & \kappa_z \rho & 0 \\ 0 & U \cdot \kappa & 0 & 0 & \kappa_x/\rho \\ 0 & 0 & U \cdot \kappa & 0 & \kappa_y/\rho \\ 0 & 0 & 0 & U \cdot \kappa & \kappa_z/\rho \\ 0 & \kappa_x \rho a^2 & \kappa_y \rho a^2 & \kappa_z \rho a^2 & U \cdot \kappa \end{pmatrix}, \qquad W = \begin{pmatrix} \rho \\ u \\ v \\ w \\ p \end{pmatrix}, \tag{2.88}$$

and κ is an arbitrary vector which fixes the direction in which we desire to know the information transport properties of the equations. When deriving boundary conditions κ is the surface normal vector. The eigenvalues of $A \cdot \kappa$ describe the speed of propagation of the various waves in the direction κ, the *characteristic speeds*, and are

$$\lambda = \left(\lambda^+,\, \lambda^-,\, \lambda^0,\, \lambda^0,\, \lambda^0\right),$$

where

$$\lambda^+ = U \cdot \kappa + a, \quad \lambda^- = U \cdot \kappa - a, \quad \lambda^0 = U \cdot \kappa.$$

The corresponding eigenvectors describe the quantities transported with these waves in terms of the primitive variables; they are the rows of the matrix X, where

$$X = \begin{pmatrix} 0 & \kappa_x & \kappa_y & \kappa_z & 1/(\rho a) \\ 0 & -\kappa_x & -\kappa_y & -\kappa_z & 1/(\rho a) \\ \kappa_x & 0 & \kappa_z & -\kappa_y & -\kappa_x/a^2 \\ \kappa_y & -\kappa_z & 0 & \kappa_x & -\kappa_y/a^2 \\ \kappa_z & \kappa_y & -\kappa_x & 0 & -\kappa_z/a^2 \end{pmatrix},$$

and result in the *characteristic variables* of transported quantities Λ,

$$\Lambda = \begin{pmatrix} \Lambda^+ \\ \Lambda^- \\ \Lambda^0 \end{pmatrix} = X(W_0, \kappa) \cdot W \;=\; \begin{pmatrix} p/(\rho_0 a_0) + U \cdot \kappa \\ p/(\rho_0 a_0) - U \cdot \kappa \\ (\rho - p/a_0^2)\kappa + U \times \kappa \end{pmatrix}$$
$$= \begin{pmatrix} p/(\rho_0 a_0) + u\kappa_x + v\kappa_y + w\kappa_z \\ p/(\rho_0 a_0) - u\kappa_x - v\kappa_y - w\kappa_z \\ (\rho - p/a_0^2)\kappa_x + v\kappa_z - w\kappa_y \\ (\rho - p/a_0^2)\kappa_y + w\kappa_x - u\kappa_z \\ (\rho - p/a_0^2)\kappa_z + u\kappa_y - v\kappa_x \end{pmatrix}, \tag{2.89}$$

and $X(W_0, \kappa)$ again refers to the fact that the result is only valid linearly local to W_0, and with respect to the direction κ.

That the Λ are the transported quantities, with transport speeds λ may easily be seen by substituting (2.89) into the Euler equation in primitive variables (2.87), to obtain the Euler equations in characteristic variables:

$$\frac{\partial \Lambda^i}{\partial t} + \lambda^i \delta_{ij,k} \frac{\partial \Lambda^j}{\partial x_k} = 0,$$

where $\delta_{ij,k}$ is the Kronecker-Delta, repeated three times (on the index k), once for each coordinate direction, and there is no summation on i above.

Having now complete information about the characteristic speeds and variables, it is possible to derive permeable boundary conditions as follows: by considering the signs of the characteristic speeds, determine how many flow quantities are transported onto the boundary, and how many are transported away. For example in the case of subsonic flow out of the domain λ^+ and λ^0 are positive while λ^- is negative, implying that Λ^+ and Λ^0 are transported onto the boundary from the field, and Λ^- is determined at the boundary. Hence one may choose a single flow variable to be set on the boundary termed the *physical boundary condition*; in general one must choose a set of variables corresponding to the number of negative eigenvalues[3].

Having chosen a variable, or set of variables to specify on the boundary, the remaining flow quantities are found as solutions of the *numerical boundary conditions.* Since the characteristic variables are transported directly onto the boundary, we must have

$$\Lambda_b^i(W_c) = \Lambda_e^i(W_c)$$

for all those Λ^i for which λ^i is positive. Here b signifies a point on the boundary, and e an imaginary point immediately outside the domain near b. The linearization in the computation of the characteristic variables is performed with respect to the state at b.

In *TAU* the farfield boundary condition is implemented as follows: for each boundary point λ are found, and so the condition type determined, subsonic or supersonic, inflow or outflow. For subsonic outflow the pressure at e is specified as the physical condition, for subsonic inflow pressure and velocity at e are specified. The boundary is then regarded as a Riemann problem and solved as such for the flux through the associated boundary face, resulting in a weak boundary condition. The approximate Riemann solver used bears no relation to that used for the interior fluxes.

The derivation of the numerical boundary condition for the subsonic outflow case is given as an example in the following section.

2.10.1 Subsonic Outflow Condition

Consider the case of specifying the pressure p_{BC} on a subsonic outflow surface, whereby κ is chosen to be the surface normal vector n. From the previous section the conditions at the boundary are

$$\begin{aligned} p_b &= p_{BC} && \text{- one physical condition} \\ \Lambda_b^+(W_b) &= \Lambda_e^+(W_b) && \text{- one numerical condition} \\ \Lambda_b^0(W_b) &= \Lambda_e^0(W_b) && \text{- three numerical conditions} \end{aligned}$$

and so the numerical conditions expanded read

$$\begin{aligned} (U_e \cdot n) + p_e/(\rho_b a_b) &= \Lambda_b^+(W_b), \\ (\rho_e - p_e/a_b^2)n + U_e \times n &= \Lambda_b^0(W_b). \end{aligned}$$

By substituting p_{BC} for p_e in both equations, and taking the dot product of the second equation with n, a simple expression for ρ_e is obtained. An expression for U_e then follows directly giving:

$$W_e = \begin{pmatrix} \rho_e \\ U_e \\ p_e \end{pmatrix} = \begin{pmatrix} \Lambda_b^0 \cdot n + p_{BC}/a_b^2 \\ \left(\Lambda_b^+ - p_{BC}/(\rho_b a_b)\right) n + n \times \Lambda_b^0 \\ p_{BC} \end{pmatrix}, \tag{2.90}$$

[3]The choice of variables to set is not completely free however; it is obviously not possible to set Λ^+ on a subsonic outflow boundary for example. See (Hirsch, 1989) for a complete discussion.

whereby it has been noted that the tangential velocity component U_t satisfies

$$U_t = n \times \Lambda^0 = n \times (U \times n) = U - (U \cdot n)n.$$

Equation (2.90) may be simplified further by substituting in the expressions for Λ_b, and is the desired boundary condition.

2.11 Turbulence Model Discretization

Two classes of turbulence models are used in this thesis: one-equation models typified by Spalart-Allmaras (SA) (Spalart & Allmaras, 1992), introduce a transport equation for a quantity closely related to the turbulence eddy viscosity, denoted $\tilde{\nu}_t$. Two-equation models introduce transport equations for the turbulent kinetic energy k and one other quantity, typified by the Wilcox $k - \omega$ model.

In both cases the transport equations are given by

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla \cdot (\rho U \phi - \mu_e^t \nabla \phi) - S, \tag{2.91}$$

where $\rho\phi$ is the transported quantity, convected with the flow, μ_e^t is a turbulence effective viscosity, and S is a source term acting as a volume contribution to the residual rather than a conservative flux.

The source terms in all models considered here are discretized such that the source term at node i requires flow quantities and gradients at node i only, i.e. they have a stencil of immediate neighbours. Thus use of only 1st-order accurate convective fluxes together with the choice of TSL gradients in the diffusion terms, means that the stencil of the entire turbulence model discretization may consist only of immediate neighbours. This reduces the fill-in of the Jacobian, and hence allows the effective use of implicit methods for the turbulence equations.

In this section the SA and closely related Spalart-Allmaras-Edwards (SAE) (Edwards & Chandra, 1996) models are fully described as they are implemented in *TAU*.

2.11.1 Expression for the Eddy-Viscosity

Given the SA/SAE transported quantity $\tilde{\nu}_t$ at a given point, there is an explicit algebraic expression for the eddy-viscosity ν_t, there:

$$\nu_t = \frac{\chi^3}{\chi^3 + c_{v1}^3} \rho \max(\tilde{\nu}_t, 0), \tag{2.92}$$

$$\chi = \frac{\tilde{\nu}_t}{\nu_l}, \tag{2.93}$$

where c_{v1} is a turbulence model constant taken as 7.1.

2.11.2 Convective Fluxes

In *TAU* the convective terms occurring in all turbulence models are discretized using the same fully upwinded scheme with piecewise constant face reconstruction (regardless of the discretization of the mean-flow convection), having therefore order of accuracy Δx in space. If the turbulent transported quantity is $(\rho\phi)$ then the flux function may be written

$$\hat{f}_{ij}^{c,t} = \tfrac{1}{2} q_{ij}(\rho_i\phi_i + \rho_j\phi_j) - \tfrac{1}{2}|q_{ij}|(\rho_j\phi_j - \rho_i\phi_i), \tag{2.94}$$

where

$$q_{ij} = \tfrac{1}{2}(U_i + U_j)\cdot n_{ij}, \tag{2.95}$$

so that if $q_{ij} > 0$, (2.94) is independent of W_j and vice versa.

2.11.3 Diffusion Fluxes

Similarly to the convective fluxes, all turbulence model transport equations share a diffusive flux discretization. Again using $\rho\phi$ for the transported quantity,

$$\begin{aligned} \hat{f}_{ij}^{v,t} &= \mu_e^t \nabla\phi\cdot n & (2.96)\\ &\approx \mu_e^t \frac{\phi_j - \phi_i}{\|\Delta x\|}, & (2.97) \end{aligned}$$

i.e. TSL gradients, where the effective *turbulent* viscosity μ_e^t is different from the effective viscosity of the mean-flow equations μ_e, and is defined differently for one- and two-equation models. For SA/SAE it is simply

$$\begin{aligned} \mu_e^t &= \frac{\mu_l + \mu_t^t}{\sigma_m}, & (2.98)\\ \mu_l &= \tfrac{1}{2}(\mu_{l,i} + \mu_{l,j}), & (2.99)\\ \mu_t^t &= \tfrac{1}{2}(\rho_i\tilde{\nu}_{t,i} + \rho_j\tilde{\nu}_{t,j}) & (2.100) \end{aligned}$$

where $\sigma_m = \frac{2}{3}$ is a model constant.

2.11.4 Spalart-Allmaras Source Terms

The Spalart-Allmaras source term at a given node (indices dropped for simplicity) is

$$\begin{aligned} S^{SA} &= P - D + d,\\ P &= c_{b1}\cdot\rho\tilde{\nu}_t\cdot\left(|\omega| + f_{v2}\cdot\frac{\rho\tilde{\nu}_t}{\rho\bar{\kappa}^2 d_{wall}^2}\right),\\ D &= c_{w1}\cdot\frac{(\rho\tilde{\nu}_t)^2}{\rho d_{wall}^2}\cdot f_w,\\ d &= \frac{c_{b2}}{\sigma_m}\rho\|\nabla\tilde{\nu}_t\|^2, \end{aligned}$$

where P, D and d are production, destruction and diffusion terms respectively, d_{wall} is the distance to the nearest wall, and

$$f_{v2} = 1 - \frac{\chi}{1+\chi\cdot f_{v1}}, \qquad f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \qquad \chi = \frac{\tilde{\mu}_t}{\mu_l} = \frac{\rho\tilde{\nu}_t}{\mu_l},$$

$$f_w = g\cdot g_{lim}^{1/6}, \qquad g = r + c_{w2}\cdot\left(r^6 - r\right), \qquad r = \frac{r_s}{\omega + f_{v2}\cdot r_s},$$

$$r_s = \frac{\rho\tilde{\nu}_t}{\rho\bar{\kappa}^2 d_{wall}^2}, \qquad g_{lim} = \frac{1 + c_{w3}^6}{g^6 + c_{w3}^6},$$

where ω is the vorticity,

$$\omega^2 = \left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial z} - \frac{\partial w}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial x} - \frac{\partial u}{\partial z}\right)^2, \tag{2.101}$$

$\bar{\kappa} = 0.4100$ is the Karman constant, and all quantities remaining undefined are other model constants. In particular

$$\begin{aligned} c_{b1} &= 0.1355, & c_{b2} &= 0.6220, & c_{v1} &= 7.1000, \\ c_{w1} &= \frac{c_{b1}}{\bar{\kappa}^2} + \frac{1+c_{b2}}{\sigma_m}, & c_{w2} &= 0.3000, & c_{w3} &= 2.0000, \\ & & \sigma_m &= 0.6667. \end{aligned}$$

2.11.5 Spalart-Allmaras-Edwards Source Terms

The only difference between the SA and SAE models lies in the source term. The SAE source term for a single node is

$$\begin{aligned} S^{SAE} &= P - D + d, \\ P &= c_{b1} \cdot \rho\tilde{\nu}_t \cdot f_{rot} \cdot s_{fac} \cdot \sigma, \\ D &= c_{w1}\rho\tilde{\nu}_t \cdot r_s \cdot f_w, \\ d &= \frac{c_{b2}}{\sigma_m}\rho\|\nabla\tilde{\nu}_t\|^2, \end{aligned}$$

where once again P, D and d are production, destruction and diffusion terms respectively, d_{wall} is the distance to the nearest wall, and

$$\sigma = \sqrt{\max(v_\sigma, 0)}, \qquad s_{fac} = f_{v1} + \frac{1}{\max(\chi, \epsilon)}$$

and $f_{rot} \equiv 1$ if no rotation correction is used. In addition

$$\begin{aligned} v_\sigma &= 2\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial z}\right)^2\right] \\ &+ \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right)^2 \\ &- \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)^2, \end{aligned}$$

and

$$\begin{aligned} s &= \max(s_{fac} \cdot \sigma, \epsilon) & f_{v1} &= \frac{\chi^3}{\chi^3 + c_{v1}^3}, & \chi &= \frac{\tilde{\mu}_t}{\mu_l} = \frac{\rho\tilde{\nu}_t}{\mu_l}, \\ f_w &= g \cdot g_{lim}^{1/6}, & g &= r + c_{w2} \cdot \left(r^6 - r\right), & r &= \frac{\tanh(r_s/s)}{\tanh(1)}, \\ r_s &= \frac{\rho\tilde{\nu}_t}{\rho\bar{\kappa}^2 d_{wall}^2}, & g_{lim} &= \frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}, \end{aligned}$$

where $\epsilon = 10^{-16}$ prevents a divide by zero condition, and where all remaining undefined quantities are model constants, taking values identical to those given in Section 2.11.4.

2.11.6 Boundary Conditions

Farfield boundary conditions are trivial, as there is only one convective speed in the turbulence equations, the flow speed, and no production or destruction. For fully turbulent calculations some fraction of the laminar viscosity at the farfield is taken for $\tilde{\nu}_{t,\infty}$, i.e.

$$\tilde{\nu}_{t,\infty} = \sigma_\infty \nu_l,$$

where σ_∞ is some turbulence model constant. On viscous walls $\tilde{\nu}_t$ is set to zero, corresponding to the absence of turbulent eddies very near to the wall.

2.12 Summary

A complete finite volume scheme on unstructured grid has been described for the Favre averaged Navier-Stokes equations, including a description of certain one-equation turbulence models. In parallel with the description of the discretization, the derivatives of the terms involved have been calculated and described, insofar as this is possible in limited space. For those terms of the discretization for which a particularly compact Jacobian exists, or for which through a suitable simplicifaction a compact Jacobian can be produced, the Jacobians have been given and the simplification justified. In particular, an original argument has been given for a common simplification of the derivatives of the JST scheme (that of the assumption of constant coefficients). No justification for this approximation has been seen previously in the literature by the author.

The solution of the discretized equations presented in this chapter is considered in the remainder of the thesis. The Jacobians developed are used as part of an implicit method, whereby the complexity of these Jacobians, demonstrated here in Section 2.6.3, limits the extent to which the exact derivatives can be applied. On the other hand the exact Jacobians without approximation will be seen to be useful in other contexts, in particular in aerodynamic design, Chapter 5.

Chapter 3

Approximately Factored Implicit Schemes

3.1 Introduction

An implicit method for convergence acceleration for stationary solutions of the RANS equations is developed based on the requirements that (i) the memory usage should not significantly exceed that of Runge-Kutta, (ii) implementation should be relatively easy, (iii) parallelization should be possible and efficient, (iv) the method should be at least as robust as RK. A method satisfying these conditions would then be a slot-in substitute for Runge-Kutta in any application. If in addition it achieves better convergence performance in terms of CPU time, then the performance of the solver will have been improved with no drawbacks.

The method developed in this chapter is a novel variant of the *Lower-Upper Symmetric Gauss-Seidel* (LU-SGS) scheme, which has its origins in CFD in (Jameson & Turkel, 1981), which considers a Newton method with a very lax Jacobian approximation, resulting in a linear system that is inexactly solved using a single step of a symmetric Gauss-Seidel method. The name LU-SGS implies in particular the use of a first-order Jacobian, introduced in (Yoon & Jameson, 1986a), and the use of the resulting scheme as a FAS multigrid smoother (Yoon & Jameson, 1986b; Yoon & Jameson, 1988). The novel aspects of the scheme presented here are described in Section 1.1.2.

The body of this chapter is divided into three sections. Section 3.2 contains a full description of the modifed LU-SGS scheme as it applies to finite volume discretizations of the RANS equations on unstructured grids, including details such as implicit handling of dual-time iterations. Section 3.3 contains a detailed theoretical analysis of the scheme; principally an assessment of the stability of the Gauss-Seidel iteration by means of a generalized theory of diagonal dominance, Fourier analysis, and a demonstration of the time-accuracy of the scheme. Section 3.4 is then concerned with numerical results, the cornerstones of which are an investigation of the effects of the various approximations made in Section 3.2, parallel performance results, and finally a comparison of the relative efficiencies of LU-SGS and Runge-Kutta as multigrid smoothers for large engineering test cases. In the following "Runge-Kutta" refers to the explicit Runge-Kutta method with residual smoothing, local time stepping and FAS multigrid, as described in (Jameson & Baker, 1984).

3.2 Description of the Scheme

Starting with the governing equations in a semi-discrete form, from (2.25):

$$\frac{\partial}{\partial t}\int_{\Omega_i} W_i \, d\Omega + R_i(W) = 0, \tag{3.1}$$

and assuming that the node i is at the baricenter of the control volume Ω_i, and further that W varies linearly within Ω, we have

$$|\Omega_i|\frac{\mathrm{d}W_i}{\mathrm{d}t} + R_i(W) = 0, \tag{3.2}$$

where $|\Omega_i|$ is the volume of Ω_i. The system is further discretized in time. A somewhat general three-level discretization may be written

$$\frac{|\Omega_i^n|\hat{M}_i^n}{\Delta t^n}\Delta W_i^n = -\frac{\beta}{1+\alpha}R_i^{n+1} - \frac{1-\beta}{1+\alpha}R_i^n + \frac{\alpha}{1+\alpha}\frac{|\Omega_i^{n-1}|\hat{M}_i^{n-1}}{\Delta t^{n-1}}\Delta W_i^{n-1}, \tag{3.3}$$

Here $\Delta W^n = W^{n+1} - W^n$, $R^n = R(W^n)$, and $\hat{M}$ is the mass matrix. The superscripts denote the time level, with all quantities up to and including time level n known, and W^{n+1} unknown. Since only the steady state solution is of interest, the mass matrix is lumped: $\hat{M} = I$, and we set $\alpha = 0$, which eliminates dependence on the solution at time t^{n-1}, giving

$$\frac{|\Omega_i^n|}{\Delta t^n}\Delta W_i^n = -\beta R_i^{n+1} - (1-\beta)R_i^n. \tag{3.4}$$

Equation (3.4) is a non-linear algebraic system for ΔW^n, which may be solved by application of Newton's method as follows. The system is linearized about the solution at time t^n, so that

$$\begin{aligned} R_i(W^{n+1}) &= R_i(W^n) + \frac{\partial R_i(W^n)}{\partial t}\Delta t_i^n + \mathcal{O}(\Delta t^2), \\ &= R_i(W^n) + \sum_{j\in\mathcal{M}(i)} \frac{\partial R_i(W^n)}{\partial W_j}\frac{\partial W_j^n}{\partial t}\Delta t_i^n + \mathcal{O}(\Delta t^2), \end{aligned} \tag{3.5}$$

where the time step Δt may vary within the grid such that the CFL number is constant, and where $\mathcal{M}(i)$ is the set of grid points in the stencil of R_i, emphasising that contributions from $\partial R_i/\partial W_j$ where $j \notin \mathcal{M}(i)$ are zero. Substituting (3.5) into (3.4), and applying

$$\frac{\partial W_j^n}{\partial t}\Delta t^n = \Delta W_j^n + \mathcal{O}(\Delta t^2), \tag{3.6}$$

results in the linear algebraic system

$$\left\{\frac{|\Omega_i|}{\Delta t_i}\delta_{ij} + \beta\frac{\partial R_i(W^n)}{\partial W_j}\right\} \cdot \Delta W_j^n = -R_i(W^n), \tag{3.7}$$

$$A(W^n)\cdot\Delta W^n = -R(W^n), \tag{3.8}$$

where A represents the implicit system matrix. Equation (3.8) encapsulates the implicit scheme: the right-hand side (RHS) of this equation is known as the *RHS of* *the scheme* and contains all elements of the spatial discretization. However if $\hat{W}$ is the exact solution of the discrete system, then $R(\hat{W}) \equiv 0$ and therefore by (3.8), $\Delta W = 0$ independently of A, provided only that it is non-singular. Hence the left-hand side (LHS), of (3.8) has no effect on the converged solution, and may be freely chosen to improve convergence.

The method for solving the steady state system $R(W) = 0$ might then proceed as follows:

- Linearize (3.4) about W^n; in particular evaluate $A(W^n)$ to form (3.8).
- Solve (3.8) to obtain ΔW^n and hence W^{n+1}.
- Repeat until $\Delta W^n < \epsilon_{\text{tol}}$ or $\|R(W^n)\| < \epsilon_{\text{tol}}$.

For $1/\Delta t = 0$ and $\beta = 1$ the method reduces to Newton's method which has the properties of *quadratic convergence* for W^0 in the domain of *approximate zeros* of R (Butcher, 1987). In practice the region of W^0 for which the method converges tends not to contain the typical initial condition, which is taken as constant flow values, everywhere conforming to the farfield condition (while other boundary conditions are violated), which is possible as time accuracy is not required. Thus some distinct *start-up* technique is required.

This chapter considers only methods which use approximate Jacobians (known as *approximate Newton methods*), inexact linear system solvers (*inexact Newton methods*), and finite values of Δt. Such methods tend not to have the undesirable properties of Newton methods, and in addition are much easier to formulate, as the Jacobian need not be calculated exactly. Further the additional flexibility in the construction of such methods allows the avoidance of many problems traditionally associated with implicit schemes, namely large memory requirements, high computational cost per iteration, and difficulty of parallelization.

There are two critical choices to be made when defining an implicit method, the approximation to the Jacobian, and the method for solution of the resulting linear system. These choices cannot be made independently, as discussed in Section 1.1, it making little sense to solve a very approximate system to extremely high accuracy, and just as little to spend a great deal of effort building an exact Jacobian, and then incurring a large error by solving very approximately. Since it is the aim of this chapter to develop a method of very low memory requirements, and it is expected that many approximations to the Jacobian must be made in order to achieve this goal, only low-powered linear solvers are candidates. There are only two main possibilities, Jacobi and Gauss-Seidel, with the option perhaps to use a line solver in structured regions of the grid. Symmetric Gauss-Seidel has the advantage of achieving communication between every pair of nodes in the grid, within one iteration, and is therefore the starting point. This method is described in the next section, followed by details of the approximate Jacobian.

3.2.1 Symmetric Gauss-Seidel (SGS) Solution

The Gauss-Seidel method may be written as follows: given a non-singular linear system $A \cdot x = b$, decompose the system matrix A into a purely lower triangular part

L, a diagonal part D and and upper triangular part U such that

$$A \cdot x = (L + D + U) \cdot x = b. \tag{3.9}$$

Of course in the context of the implicit method A is the implicit system matrix, x is the update ΔW, and b is the residual $-R$. Then two possible Gauss-Seidel iterations with unknown x^n are

$$\begin{aligned} (D + L)x^{n+1} &= b - U \cdot x^n, \quad \text{and} \\ (D + U)x^{n+1} &= b - L \cdot x^n, \end{aligned}$$

whereby if either of these iterations converges, then $x^{n+1} = x^n = x$ the solution of the linear system. These iterations shall be known as the *forward sweep* and *backward sweep* respectively. Note that for the forward sweep x_i^{n+1} is a function of all x^n as well as x_j^{n+1} for $j < i$, while for the backwards sweep x_i^{n+1} is a function of all x^n and x_j^{n+1} for $j > i$. Hence for a composite of the two sweeps, for instance

$$\begin{aligned} (D + L)x^* &= b - U \cdot x^n, \\ (D + U)x^{n+1} &= b - L \cdot x^*, \end{aligned} \tag{3.10}$$

x_i^{n+1} will be a function of all x^n and x^{n+1}. This corresponds to the Courant-Friedrichs-Lewy (CFL) condition being automatically satisfied. This is the *Symmetric Gauss-Seidel* (SGS) iteration.

In the case of a single SGS iteration, which will be seen to be the optimal number of iterations in our case, and by imposing the restriction that $x^0 = 0$, (3.10) reduces to

$$\begin{aligned} (D + L)x^* &= b, \\ (D + U)x^1 &= Dx^*, \end{aligned} \tag{3.11}$$

which may be written

$$(D + L) \cdot D^{-1} \cdot (D + U)x^1 = b, \tag{3.12}$$

which is the so-called *LU-SGS* iteration. Equation (3.12) highlights an alternative interpretation of the method, as the system $A \cdot x = b$ has been replaced by an alternative system $\bar{A} \cdot x = b$, the two systems being related by

$$\begin{aligned} A &= (L + D + U) \\ &= (D + L) \cdot D^{-1} \cdot (D + U) - L \cdot D^{-1} \cdot U \\ &= \bar{A} - L \cdot D^{-1} \cdot U, \end{aligned} \tag{3.13}$$

i.e. an approximate factorization of A with factorization error $L \cdot D^{-1} \cdot U$.

The SGS method introduces an iteration on a linear system in addition to the iteration of the Newton method. In the following an iteration on a linear system, by SGS or some other iterative linear solver, will be referred to as an *inner* or *linear iteration.* An iteration on the non-linear system will be referred to as an *outer* or *non-linear iteration.* A non-linear iteration can be characterized by the need to re-evaluate the non-linear residual $R(W)$ at least once.

For the purposes of comparison, the Jacobi method for solution of the linear system is also examined in Section 3.4, written

$$Dx^{n+1} = b - (U + L) \cdot x^n, \tag{3.14}$$

whereby Jacobi(n) will refer to a scheme with n Jacobi iterations per non-linear step. In constrast to SGS, two nodes in the grid with a shortest connecting path of m grid edges, will only communicate with each other after m Jacobi iterations. Hence the CFL condition is limited by m times the local cell dimension.

3.2.2 Inviscid Flux Jacobians

Next the construction of the Jacobian of the method is described. Consider the inviscid flux balance on an control volume using a numerical flux $\hat{f}$, and a numerical boundary flux $\hat{f}_b$. Let the values on the left and right sides of the face W_L, W_R be approximated by piecewise constant reconstruction of the cell values, so that $W_L = W_i$ and $W_R = W_j$, then

$$\begin{aligned} R_i &= \sum_{j\in\mathcal{N}(i)} \hat{f}(W_L, W_R; n_{ij}) + \sum_{m\in\mathcal{B}(i)} \hat{f}_b(W_L; n_m) \\ &= \sum_{j\in\mathcal{N}(i)} \hat{f}(W_i, W_j; n_{ij}) + \sum_{m\in\mathcal{B}(i)} \hat{f}_b(W_i; n_m), \end{aligned} \tag{3.15}$$

where as before $\mathcal{N}(i)$ is the set of all immediate neighbours of point i in the grid, and $\mathcal{B}(i)$ is the set of all neighbouring boundary faces. Assuming that $\hat{f}$ may be written in dissipation form

$$\hat{f}(W_L, W_R; n_{ij}) = \frac{1}{2}\left(f^c(W_L) + f^c(W_R)\right) \cdot n_{ij} - \frac{1}{2} D(W_L, W_R; n_{ij}), \tag{3.16}$$

where f^c is the exact convective flux tensor given in (2.4), and further noting that the central part of this flux consists of terms of the form of the product $f^c \cdot n$, allows rewriting (3.15) as

$$\begin{aligned} R_i &= \frac{1}{2} f^c(W_i) \cdot \sum_{j\in\mathcal{N}(i)} n_{ij} + \frac{1}{2} \sum_{j\in\mathcal{N}(i)} f^c(W_j) \cdot n_{ij} \\ &- \frac{1}{2} \sum_{j\in\mathcal{N}(i)} D(W_i, W_j; n_{ij}) + \sum_{m\in\mathcal{B}(i)} \hat{f}_b(W_i; n_m). \end{aligned} \tag{3.17}$$

For a closed control volume i, not touching any boundaries

$$\sum_{j\in\mathcal{N}(i)} n_{ij} = 0, \tag{3.18}$$

and thus for such a control volume the dependence of R_i on W_i occurs only over the dissipation D. Differentiating R_i with respect to W_i results in the diagonal of the Jacobian matrix, which for a non-boundary control volume is now just

$$\frac{\partial R_i}{\partial W_i} = \frac{1}{2} \sum_{j\in\mathcal{N}(i)} \frac{\partial D(W_i, W_j; n_{ij})}{\partial W_i}. \tag{3.19}$$

Thus choosing a flux function with a particularly simple dissipation component derivative, results in a particularly simple Jacobian matrix diagonal. For example taking the first order *Lax-Friedrichs* numerical flux

$$\begin{aligned}\hat{f}_{LF}(W_L, W_R; n) &= \frac{1}{2}\left(F(W_L; n) + F(W_R; n)\right) \\ &- \frac{1}{2}\left|\alpha\right|(W_R - W_L), \qquad (3.20)\\ |\alpha| &= \left|\rho\left(\frac{\partial F}{\partial W}\right)\right| \\ &= |U \cdot n| + \|n\| a, \qquad (3.21)\end{aligned}$$

and treating α as constant with respect to W when differentiating, gives the Jacobian of the convective residual

$$\frac{\partial D_{LF}(W_i, W_j; n_{ij})}{\partial W_i} = \frac{1}{2}|\alpha| I. \qquad (3.22)$$

So that the derivative of the dissipation, and by (3.19) the diagonal of the Jacobian matrix, is a positive scalar multiple of the identity matrix.

In contrast off-diagonal blocks of the Jacobian matrix are in general dense, and are constructed exactly for the Lax-Friedrichs scheme as

$$\frac{\partial R_i}{\partial W_j} = \frac{1}{2}\frac{\partial f_j^c}{\partial W_j} - \frac{1}{2}|\alpha| I, \qquad (3.23)$$

where the convective flux-Jacobian $\partial f / \partial W$ is given explicitly in Appendix A.

Note that for the LU-SGS method, the only part of the Jacobian that must be stored and inverted directly is the block diagonal, see (3.11). Thus the property described by (3.22) is very desirable: firstly it implies that rather than having to store an $N \times N$ matrix at each grid point, it is only necessary to store a scalar, reducing the memory cost by 25 times for the Euler equations. Secondly the cost of inverting the diagonal is reduced from that of an $N \times N$ matrix inversion to a single division.

Of course these savings are only preserved if all other components of the LHS discretization ($\hat{f}_b$, viscous fluxes, turbulence fluxes and sources, etc.), have Jacobians whose diagonal blocks are also diagonal. This will be discussed in the following sections.

3.2.3 Viscous Flux Jacobians

The construction of the viscous Jacobian proceeds in the same manner as that of the inviscid. First the operator is simplified such that the stencil of R_i contains only the immediate neighbours of the point i. In the case of the viscous fluxes this is done by replacing the full, Green-Gauss gradient based flux by the TSL approximation of Section 2.8.

The off-diagonal entries of the Jacobian may be taken directly from the formulae given in Section 2.8.1, but the Jacobian block diagonal is not diagonal itself, damaging the simplicity and memory requirements of LU-SGS. This is corrected by approximating the viscous Jacobian in diagonal blocks by the largest eigenvalue of the operator. The eigenvalues of (2.79) are

$$\lambda\left(\frac{\partial \hat{f}^v_{ij}}{\partial W_i}\right) = \begin{pmatrix} 0 \\ -\frac{(\gamma-1)\kappa A^2}{\rho \Delta x} \\ -\frac{4}{3}\frac{\mu A^2}{\rho \Delta x} \\ -\frac{\mu A^2}{\rho \Delta x} \\ -\frac{\mu A^2}{\rho \Delta x} \end{pmatrix}, \tag{3.24}$$

so that the proposed Jacobian approximation is

$$\frac{\partial \hat{f}^v_{ij}}{\partial W_i} \approx \lambda_{\max}\left(\frac{\partial \hat{f}^v_{ij}}{\partial W_i}\right) \cdot I = \max\left\{\left|\frac{4}{3}\frac{\mu \mathcal{A}^2}{\rho \Delta x}\right|, \left|\frac{(\gamma-1)\kappa \mathcal{A}^2}{\rho \Delta x}\right|\right\} \cdot I, \tag{3.25}$$

which can be simplified further using $(\gamma - 1)\kappa = \gamma\mu/\mathrm{Pr}$ (for non-dimensionalized quantities such that $\Re = 1$). For turbulent flow the viscosity μ, is replaced by its effective value.

An analogous treatment of boundary conditions is employed. As already seen, boundary conditions contribute only to the block diagonal of the Jacobian, and in general produce non-diagonal blocks which are here approximated by their spectral radii. For example the slip-wall Jacobian of Section 2.9.1 is approximated as

$$\frac{\partial \hat{f}^{s-w}_b}{\partial W_c} \approx \rho\left(\frac{\partial \hat{f}^{s-w}_b}{\partial W_c}\right) \cdot I = (\gamma - 1)\left|u n_x + v n_y + w n_z\right| \cdot I. \tag{3.26}$$

Boundary conditions whose spectral radii are too complex to evaluate conveniently are approximated using the maximum eigenvalue of the local flow, which is particularly appropriate for the farfield condition. Neglecting the contribution of a boundary condition completely would artificially reduce the diagonal dominance of the Jacobian, leading to a stiffer linear system, and is therefore avoided. The no-slip wall condition may be accounted for simply by allowing no updates of the momentum equations at the wall nodes.

3.2.4 Turbulence Jacobians

In this work only the Spalart-Allmaras (SA) and Wilcox $k - \omega$ turbulence models are considered explicitly (Wilcox, 1998). Given the number of available models, an individual treatment of each one is impractical. Rather numerical evidence suggests that most 1-equation models are sufficiently similar to SA to converge well with a Jacobian obtained from SA, and similarly for 2-equation models and $k - \omega$. In the context of implicit methods this may be viewed as the use of different models on the RHS and LHS, similar to the treatment of the convective fluxes.

Mean-Flow Turbulence Coupling

A critical design decision is that of either treating the turbulence and mean-flow equations as a coupled system, or to artificially decouple them. Decoupling is by far the most widely used strategy, for two reasons:

- The mean-flow equations have a very different character to the turbulence equations. For mean-flow, stability considerations are dominated by acoustic waves, while the stability of the turbulence equations is dominated by source terms.
- The two sets of equations are coupled over the turbulent viscosity μ_t in the momentum and energy equations, a turbulent contribution to the energy equation, and for models involving the turbulent kinetic energy k, also over the perfect gas relations. Further, the expressions for source terms and μ_t tend to be complex. Accounting for all this coupling in the Jacobians in an accurate manner for each and every model is difficult.

The unknowns and convective fluxes are hence spilt such that

$$f^c = \bar{f}^c + \tilde{f}^c = \bar{f}^c(\bar{W}, \tilde{W}) + \tilde{f}^c(\bar{W}, \tilde{W}), \tag{3.27}$$

where for a 2-equation $k - \omega$-like model

$$\bar{f}^c \cdot n = \begin{pmatrix} \rho V \\ \rho V u + p n_x \\ \rho V v + p n_y \\ \rho V H \\ 0 \\ 0 \end{pmatrix}, \quad \tilde{f}^c \cdot n = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \rho V k \\ \rho V \omega \end{pmatrix}, \tag{3.28}$$

and similarly for the unknown vector. Hence the Jacobian may be written

$$\frac{\partial f^c}{\partial W} = \begin{pmatrix} \dfrac{\partial \bar{f}^c}{\partial \bar{W}} & \dfrac{\partial \bar{f}^c}{\partial \tilde{W}} \\ \dfrac{\partial \tilde{f}^c}{\partial \bar{W}} & \dfrac{\partial \tilde{f}^c}{\partial \tilde{W}} \end{pmatrix}, \tag{3.29}$$

and a decoupling of the equations is accomplished by setting

$$\frac{\partial \bar{f}^c}{\partial \tilde{W}} = 0, \quad \frac{\partial \tilde{f}^c}{\partial \bar{W}} = 0. \tag{3.30}$$

For convenience all turbulent quantities are treated as constants within the mean flow equations, e.g. μ_t which is also a function of mean-flow variables.

The implicit scheme of (3.8) based on this decoupling may then be written

$$\begin{aligned} \left(\frac{|\Omega|}{\Delta \bar{t}} + \frac{\partial \bar{R}}{\partial \bar{W}} \right) \Delta \bar{W} &= -\bar{R}, \\ \left(\frac{|\Omega|}{\Delta \tilde{t}} + \frac{\partial \tilde{R}}{\partial \tilde{W}} \right) \Delta \tilde{W} &= -\tilde{R}, \end{aligned} \tag{3.31}$$

whereby it must not generally be the case that $\Delta \bar{t} = \Delta \tilde{t}$. The submatrix $\partial \tilde{R} / \partial \tilde{W}$ is simplified in a similar manner to $\partial \bar{R} / \partial \bar{W}$, and for the same reasons. In particular the sum of convective flux-Jacobians over an internal cell is a diagonal matrix[1].

[1]In *TAU* convective fluxes for the turbulence equations are modelled using a first-order upwind scheme, and so the LHS and RHS of the implicit system match much better for the turbulence equations than for the mean-flow equations.

On an implementational note: it is not necessary to solve the two systems of (3.31) separately; in fact it is more efficient in terms of operation counts to formulate and solve them simultaneously, as many quantities (e.g. the coefficient of artificial viscosity) appear in the fluxes and Jacobians of both systems.

Turbulence Diffusion Fluxes

Referring to Section 2.11.3 for the definition of the turbulence diffusion fluxes based on TSL flow gradients, the Jacobian can be seen to be

$$\frac{\partial \hat{f}_{ij}^{v,t}}{\partial W_{[ji]}} = \pm \frac{\mathcal{A}_{ij}\mu_e^t}{\Delta x_{ij}}, \tag{3.32}$$

with respect to primitive variables. For 2-equation turbulence models this derivation proceeds independently for each turbulence variable, and hence the resulting 2×2 matrix is diagonal.

Turbulent wall boundary conditions are treated strongly, therfore the discussion of Section 3.2.3 with regard to the no-slip wall applies. It remains to consider source terms.

Source terms for Spalart-Allmaras

The sources of SA contain terms of the form $\partial u/\partial x$. A full differentiation of these terms with respect to all variables would imply differentiation of the gradient of U, resulting in off-diagonal terms in the Jacobian. In contrast, differentiation of the sources with respect to the turbulent variables alone results in a block diagonal Jacobian; hence decoupling the mean-flow and turbulence equations has simplified the scheme dramatically.

Even so, the complexity of the expressions for the source terms encourages further approximations to the Jacobian. In particular the exact differentiation is complicated by the terms r_s and g in Section 2.11.4, so several considerable approximations are made (e.g. constant χ) to obtain the approximate Jacobians of the production, destruction and cross-flow terms for LU-SGS:

$$\frac{\mathrm{d}P}{\mathrm{d}(\rho\tilde{\nu}_t)} \approx c_{b1}\left(|\omega| + 2f_{v2}\frac{\rho\tilde{\nu}_t}{\rho\kappa^2 d_{wall}^2}\right), \tag{3.33}$$

$$\frac{\mathrm{d}D}{\mathrm{d}(\rho\tilde{\nu}_t)} \approx 2c_{w1k}\frac{\rho\tilde{\nu}_t}{\rho\kappa^2 d_{wall}^2} f_w, \tag{3.34}$$

$$\frac{\mathrm{d}d}{\mathrm{d}(\rho\tilde{\nu}_t)} \approx -c_{b2s}\frac{2}{\rho}(\nabla\tilde{\nu}_t \cdot \nabla\rho). \tag{3.35}$$

It is common practice in turbulence modelling to ensure that the source term Jacobian is positive thereby avoiding the use of a limiter (Spalart & Allmaras, 1992). Since the source term is subtracted from the residual, e.g. in 2.91, only negative terms are kept:

$$\frac{\mathrm{d}S_{SA}}{\mathrm{d}(\rho\tilde{\nu}_t)} \approx -\frac{\mathrm{d}D}{\mathrm{d}(\rho\tilde{\nu}_t)} + \min\left\{\frac{\mathrm{d}d}{\mathrm{d}(\rho\tilde{\nu}_t)}, 0\right\} \le 0. \tag{3.36}$$

This treatment has a serious deficiency: it may be the case that the production and destruction Jacobians are both very large, but almost cancel each other out. But from

(3.36) a large contribution will be made to the Jacobian, which may be interpreted as a reduction of the time step. In such a situation very poor convergence will occur. However in this case stability considerations outweigh efficiency considerations.

3.2.5 Dual-Time Treatment

Dual-time is a time-accurate stepping method consisting of solving an implicit time discretization, such as a backward difference formula (BDF), by writing the solution at each time step as a steady state problem, and solving this using known, time-inaccurate methods, see (Jameson, 1991).

In this case the time derivative of (3.2) is discretized with a BDF, and the resulting algebraic system is solved by progressing the equation

$$\frac{\partial W^*}{\partial \tau} = -\left\{ \frac{|\Omega|(3W^* - 4W^n + W^{n-1})}{2\Delta t} + R(W^*) \right\} = -R^*(W^*) \tag{3.37}$$

to a steady state in τ. Here τ is an artificially introduced pseudo time, W^* is the iterate, and the iterations are known as *inner iterations*. At a steady state $\partial W^*/\partial \tau = 0$ and hence $W^* = W^{n+1}$. This system strongly resembles the original problem (3.2) for the steady state case, and so LU-SGS may be applied here. Note that the modification to the residual modifies the Jacobian in the scheme as

$$\frac{\partial R^*}{\partial W^*} = \frac{\partial R}{\partial W^*} + \frac{3}{2\Delta t} \cdot I, \tag{3.38}$$

so that the correct treatment of the additional term increases the diagonal dominance of the system. Numerical experiments show that when using this technique, LU-SGS is more stable than usual.

3.3 Analysis of the Scheme

Though it is easy to quantify the performance of a scheme in practical applications, simply by implementing and testing it, such a procedure offers few insights into why the scheme behaves as it does, where it is particularly weak, and how it may be improved. For such insight theoretical results are vital, even involving - as they generally do - substantial simplifications.

The convergence of the LU-SGS scheme is first investigated using the theory of stability of linear fixed-point iterations extended to account for block-structured matrices, Section 3.3.1. It is shown that the exact Jacobian of the Euler spatial discretization is not block diagonally dominant (a weaker condition than diagonal dominance), nor is the approximate Jacobian of LU-SGS. Thereupon a modified Jacobian is proposed that is only just diagonally dominant, and therefore the SGS iteration is guaranteed to converge. The new scheme (LU-DD) is implemented and it is seen numerically that while the inner SGS iteration converges much better than the original scheme, the non-linear convergence is much poorer due to the greater mismatch between the residual and Jacobian. Hence it is concluded that diagonal-dominance is much too strong a condition on the Jacobian, and hence the corresponding stability results are not useful in this context.

Section 3.3.2 performs a Fourier analysis of LU-SGS applied to a scalar convection-diffusion-source term equation. The Fourier analysis necessitates also the assumptions of a Cartesian grid with constant-spacing and periodic boundary conditions. The Jacobian is based on the derivatives of a first-order upwind discretization while the spatial discretization is central, modelling the residual-Jacobian mismatch of the scheme. Despite this and the approximate LU factorization, the Fourier analysis continues to show that the scheme behaves similarly to an exact implicit scheme, damping low frequency error modes strongly. This result is somewhat inconsistent with the numerical results of Section 3.4, suggesting that some feature of the method not treated by the analysis, most likely coupled equations or boundary conditions have an important influence on the method. The effect of the source term on the scheme is also examined and it is seen that the stability of a forward-Euler step is strongly dependent on its magnitude, but - at least in one dimension - LU-SGS is at least as stable as the model equation itself, independently of the source term and the time step.

Finally in Section 3.3.3 the time-accuracy of the scheme is examined, and on the basis of numerical results it is concluded that the time-accurate form may be useful for DES and LES simulations (Dwight, 2004). Based on these results the scheme has since been used for DES calculations of aerofoils on unstructured grids (Van der Ven & Weinman, 2004; Soda *et al.*, 2005).

3.3.1 Theoretical Stability Conditions for Fixed-Point Iterations

A well known sufficient condition for Jacobi and Gauss-Seidel iterations of a linear system to be stable is that the system matrix be diagonally dominant. Here this result is extended to block-Jacobi and Gauss-Seidel iterations, and the corresponding block diagonal dominance condition is examined for the scheme of Section 3.2.

General Convergence Conditions

For any splitting N, M of a matrix A, where M is invertible, a linear fixed-point iteration

$$M\psi^{n+1} = b - N\psi^n, \tag{3.39}$$

converges to the solution of $A\psi = b$ for all initial conditions ψ^0 and RHSs b if and only if $\rho(M^{-1}N) < 1$, where $\rho(A)$ represents the spectral radius of A. This result may be immediately seen by writing ψ^n in terms of ψ^0:

$$\psi^n = \left\{ \sum_{j=0}^{n-1} (-1)^j \left(M^{-1}N\right)^j \right\} M^{-1}b + (-1)^n \left(M^{-1}N\right)^n \psi^0, \tag{3.40}$$

where the sum represents a Taylor series for the inverse of $I + M^{-1}N$, and hence the first term tends to $A^{-1}b$ as $n \to \infty$ if the series converges. Further the rate of convergence also depends upon of the magnitude of $\rho(M^{-1}N)$.

However, $\rho(A)$ is difficult to evaluate. We might use the fact that

$$\rho(A) \leq \|A\|, \quad \forall \text{ matrix norms } \|\cdot\|, \tag{3.41}$$

to obtain another sufficient condition for convergence, but matrix norms are also costly to evaluate and difficult to simplify algebraically. A simpler and popular (but relatively weak) bound is obtained via Gerschgorin's Circle Theorem for the eigenvalues of a general matrix (Saad, 2003). The final result is that if

$$|a_{ii}| > \sum_{j=1;j\neq i}^{n} |a_{ij}|, \quad \forall i \in \{1,\dots n\}, \tag{3.42}$$

then Jacobi and Gauss-Seidel iterations converge for any ψ^0. If a matrix A satisfies (3.42) then it is said to be *strictly diagonally dominant*.

The Jacobian of the NS equations however has a block structure,

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ A_{n1} & \cdots & \cdots & A_{nn} \end{pmatrix}, \tag{3.43}$$

where each A_{ij} is a 5×5 block matrix for the Euler equations, or a 6×6 matrix if a 1-equation turbulence model is used, etc. Let the particular decomposition into blocks be called a *partitioning* and let the dimension of each A_{ij} be N. Applying condition (3.42) to A would then result in N separate conditions to be satisfied.

A more natural approach is to use a block-generalized version of the above results. Such a generalization (Feingold & Varga, 1962), defines *block strict diagonal dominance* relative to a given partitioning, as

$$\left(\|A_{ii}^{-1}\|\right)^{-1} > \sum_{j=1;j\neq i}^{n} \|A_{ij}\|, \quad \forall i \in \{1,\dots n\}, \tag{3.44}$$

which in the special case of 1×1 blocks reduces to (3.42). Note that a matrix that is not diagonally dominant, may be block diagonally dominant under some suitable partitioning.

For this definition of block diagonal dominance there exists a result corresponding to Gerschgorin's Theorem: for the partitioned matrix A of (3.43), each eigenvalue λ of A satisfies

$$\left(\left\|(A_{ii} - \lambda I_i)^{-1}\right\|\right)^{-1} > \sum_{j=1;j\neq i}^{n} \|A_{ij}\|, \tag{3.45}$$

for at least one $i \in \{1,\dots n\}$ (Feingold & Varga, 1962). Again this reduces to the original theorem for $N = 1$, and may be extended to convergence conditions for Jacobi and Gauss-Seidel. However it holds only for matrix norms defined by

$$\|A\| = \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}, \tag{3.46}$$

for some vector norm $\|\cdot\|$. Note that under this definition

$$(\|A^{-1}\|)^{-1} = \inf_{x\neq 0} \frac{\|Ax\|}{\|x\|}, \tag{3.47}$$

whenever A is non-singular.

Given this, a sufficient convergence condition for the Jacobi iteration may be derived as follows: consider $M^{-1}N = D^{-1}(L+U)$ whose eigenvalues satisfy

$$\lambda D x = (L+U)x, \tag{3.48}$$

where x is partitioned to correspond to the partitioning of A. Now there exists a block element of x with largest norm: $\|x_i\| \geq \|x_j\|$, $\forall j$. Normalize x such that $\|x_i\| = 1$ and $\|x_j\| \leq 1$, $\forall j \neq i$. Equation (3.48) for block i is then

$$\lambda A_{ii} x_i = \sum_{j=0;j\neq i}^{n} A_{ij} x_j, \tag{3.49}$$

implying directly that

$$|\lambda| \|A_{ii} x_i\| \leq \sum_{j=0;j\neq i}^{n} \|A_{ij}\| \|x_j\| \leq \sum_{j=0;j\neq i}^{n} \|A_{ij}\|, \tag{3.50}$$

where the triangle inequality and consistency of the matrix norm to the vector norm have been applied to obtain the first inequality. Furthermore by defining $z_i = A_{ii} x_i$,

$$\|A_{ii} x_i\| = \frac{\|A_{ii} x_i\|}{\|x_i\|} = \frac{\|z_i\|}{\|A_{ii}^{-1} z_i\|} \geq \left(\|A_{ii}^{-1}\|\right)^{-1}, \tag{3.51}$$

from the definition of the matrix norm (3.46) and (3.47). Combining (3.50) and (3.51) we have

$$|\lambda| \leq \sum_{j=0;j\neq i}^{n} \|A_{ii}^{-1}\| \cdot \|A_{ij}\|, \tag{3.52}$$

so that a sufficient condition for convergence of the Jacobi iteration is

$$\sum_{j=0;j\neq i}^{n} \|A_{ii}^{-1}\| \cdot \|A_{ij}\| < 1. \tag{3.53}$$

Exactly the same result holds for Gauss-Seidel, and has a similar derivation with a little more algebra.

Non-Diagonal Dominance of LU-SGS (LU-Original)

The general results of the previous section are applied to the algorithm described in Section 3.2, with particular reference to the simplification of Section 3.2.2. In the following the time step is taken to be infinite. Considering only internal control volumes, by (3.22) the diagonal blocks of the Jacobian are multiples of the identity matrix, so

$$\|A_{ii}^{-1}\| = \left\|(\sigma_i I)^{-1}\right\| = \frac{1}{\sigma_i}, \tag{3.54}$$

where

$$\sigma_i = \frac{1}{2} \sum_{j \in \mathcal{N}(i)} |\alpha_{ij}|. \tag{3.55}$$

For the off-diagonal blocks inequality (3.41) gives

$$\|A_{ij}\| \geq \rho(A_{ij}) = \rho\left(\frac{1}{2}\frac{\partial F_j}{\partial W_j} - \frac{1}{2}|\alpha_{ij}|I\right), \tag{3.56}$$

which value may be easily found, since if λ is an eigenvalue of A then $\lambda + \mu$ is an eigenvalue of $A + \mu I$. In this case the eigenvalues of A_{ij} are

$$\lambda\left(A_{ij}\right) = \begin{pmatrix} V^- - a \\ V^- \\ V^- - \frac{1}{2}a \\ V^- - \frac{1}{2}a \\ V^- - \frac{1}{2}a \end{pmatrix} \tag{3.57}$$

where

$$V^- = \frac{1}{2}(V - |V|), \tag{3.58}$$

from which it may be seen that

$$\rho(A_{ij}) = |\alpha_{ij}^-| = -V^- + a. \tag{3.59}$$

A graph of $|\alpha^-|$ and $\frac{1}{2}|\alpha|$ against V is plotted in Figure 3.1. It may be immediately seen that for $V < a$, i.e. for a face Mach number less than 1, that

$$|\alpha_{ij}^-| > \frac{1}{2}|\alpha_{ij}|,$$

and combining this with (3.56) gives

$$\|A_{ij}\| > \frac{1}{2}|\alpha_{ij}|, \quad \forall i,j \text{ for } V < a,$$

but this implies that

$$\sum_{j\in\mathcal{N}(i)} \|A_{ij}\| > \frac{1}{2}\sum_{j\in\mathcal{N}(i)} |\alpha_{ij}| = \frac{1}{\|A_{ii}^{-1}\|},$$

and hence the system matrix is not diagonally dominant for face Mach numbers less than one, and infinite Δt.

From this derivation a condition on Δt may be obtained that guarantees the diagonal dominance of A; however the condition of diagonal dominance is only sufficient for convergence, and as will be seen it is too strict, and hence the bound on Δt is too strict as well.

Diagonal Dominance for a Modified LU-SGS (LU-DD)

Consider now a modified version of LU-SGS in which the observation of Section 3.2.2 is not made, namely that the Jacobians of the exact fluxes on summed over the faces

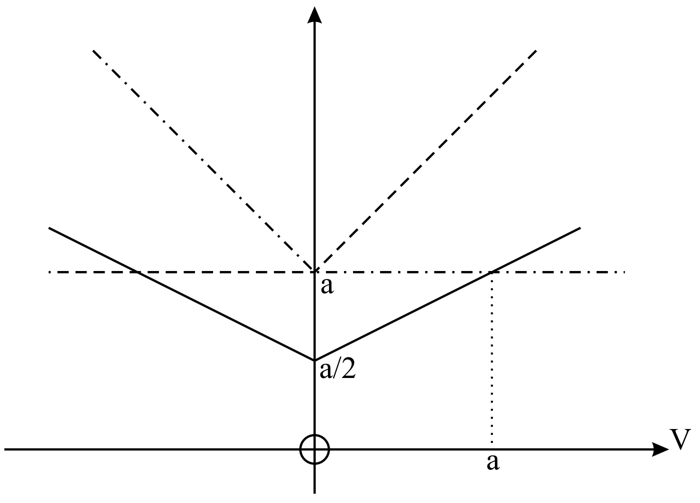


Figure 3.1: Comparison of the magnitudes of the spectral radii of diagonal and off-diagonal Jacobian blocks, as a function of V.

of a cell cancel. Instead approximate each flux Jacobian on each face by its spectral radius, thus diagonalizing the diagonal of the Jacobian as follows

$$\begin{aligned}\frac{\partial R_i}{\partial W_i} = \sum_{j\in\mathcal{N}(i)} \frac{\partial h_{ij}}{\partial W_i} &= \frac{1}{2}\sum_{j\in\mathcal{N}(i)}\left(\frac{\partial F_i}{\partial W_i} + |\alpha_{ij}|I\right)\\ &\approx \frac{1}{2}\sum_{j\in\mathcal{N}(i)}\rho\left(\frac{\partial F_i}{\partial W_i} + |\alpha_{ij}|I\right)I \\ &= \frac{1}{2}\sum_{j\in\mathcal{N}(i)}|\alpha^+_{ij}|I,\end{aligned} \tag{3.60}$$

where the spectral radius has been calculated as before, resulting in

$$|\alpha^+_{ij}| = V^+ + a, \quad V^+ = \frac{1}{2}(V + |V|). \tag{3.61}$$

If in addition the off-diagonal blocks are approximated by their spectral radii, so that

$$\frac{\partial R_i}{\partial W_j} \approx |\alpha^-_{ij}|I = (-V^- + a)I, \tag{3.62}$$

then the diagonal dominance condition of (3.53) reduces to (neglecting the influence of $\Omega/\Delta t$ again):

$$\frac{\sum_{j\in\mathcal{N}(i)}(-V^- + a)}{\sum_{j\in\mathcal{N}(i)}(V^+ + a)} < 1, \tag{3.63}$$

whereby (dropping the index on the sums for readability)

$$\begin{aligned}\sum(V^+ + a) - \sum(-V^- + a) &= \frac{1}{2}\sum(V + |V| + V - |V|)\\ &= \sum V\\ &= \sum U_{ij}\cdot n_{ij}\\ &= \sum(U^0 + \Delta U_{ij})\cdot n_{ij}\\ &\approx \mathcal{O}(\Delta U \Delta x) \approx \mathcal{O}(\Delta x^2),\end{aligned} \tag{3.64}$$

	Number of SGS iterations		Number of non-linear iterations	
Test case	LU-Original	LU-DD	LU-Original	LU-DD
NACA0012	45.4	6.7	285 (402)	885 (1373)
M6 Wing	67.3	5.7	445 (610)	996 (1607)

Table 3.1: Average number of symmetric Gauss-Seidel iterations required to converge the inner residual to 1×10^{-8} for two transonic Euler test cases and the two LU-SGS scheme Jacobians. Average is over non-linear iterations. Also number of non-linear iterations of the LU-SGS schemes need to converge the outer residual to $1 \times 10^{-3}(1 \times 10^{-4})$ (with only one SGS iteration per non-linear step).

so that the system is exactly on the boundary of diagonal dominance for infinite Δt and tends to a weakly diagonally dominant system as $\Delta x \rightarrow 0$. In this case any Jacobi or Gauss-Seidel iteration is likely to be stable for any Δt.

Discussion

In constructing an efficient implicit scheme, two convergence and two stability results must be considered: those of the non-linear outer iteration, and those of the linear inner iteration.

In general, the fewer approximations involved in the formulation of the system matrix, the more the scheme resembles a Newton method, and the better is the non-linear convergence. On the other hand, the more accurate the system matrix is, the less diagonally dominant it is likely to be, and therefore the less likely the linear solver is to converge.

Two versions of LU-SGS have been presented: the first (which we shall name LU-Original) has been shown to result in a system matrix that is never diagonally dominant (for subsonic flow), while the second (LU-DD) has been shown to be almost diagonally dominant, and may be made diagonally dominant - guaranteeing convergence of the linear system - given some weak restriction on Δt. Therefore LU-DD will outperform LU-Original on the inner iteration, and no concern needs to be made over the stability of LU-DD. This result may be verified numerically for particular test cases, by performing a large number of symmetric Gauss-Seidel iterations for each non-linear step, and examining the inner convergence. As expected LU-DD converges in all cases tested, but so does LU-Original, although the convergence of LU-DD is much faster and therefore presumably much more stable, see the first two columns of Table 3.1.

On the other hand, in terms of non-linear iterations needed for convergence, LU-Original is much faster than LU-DD, as may be seen in the second pair of columns of Table 3.1, this time with only one SGS iteration per non-linear step, i.e. LU-SGS. Thus LU-Original is a much more desirable scheme than LU-DD, its poor linear convergence being unimportant. For all these computations, $1/\Delta t = 0$, and no multigrid has been used.

These results suggest that block diagonal dominance is too stringent a condition on the Jacobian for fixed point iterations for the Euler equations. At the point at which diagonal dominance is achieved, the Jacobian is too far removed from the true Jacobian for the non-linear iteration to be effective. Furthermore block diagonal dominance is generally a significantly looser condition than simple diagonal dominance (Feingold & Varga, 1962), and hence it may be asserted that diagonal dominance in general represents too stringent a requirement on the Jacobian for these methods.

Viscous Jacobian Approximation

Consider now the effect on the diagonal dominance of the approximation of (3.25), i.e. replacing the viscous Jacobians by their largest eigenvalue. This approximation increases the diagonal dominance of the system matrix, under both definitions given in Section 3.3.1, definition (3.42) trivially, and definition (3.44) since for a general matrix A

$$\frac{1}{\|A^{-1}\|} \leq \lambda_i \leq \|A\|, \quad \forall \lambda_i \text{eigenvalues of } A, \tag{3.65}$$

for all matrix norms $\|\cdot\|$. Hence under the arguments of Section 3.3.1 the inner SGS iteration is more likely to be stable, and to converge more rapidly. The diagonal dominance may be further increased by approximating the off-diagonal viscous Jacobians by their spectral radii as in (3.25), where here the second inequality of (3.65) is applied. However making such an approximation has the associated cost of worsening the approximation of the Jacobian and therefore also worsening the rate of convergence of the non-linear iteration. The influence of this approximation on linear and non-linear convergence is evaluated numerically in Section 3.4.3.

Influence of Sum of Jacobians on Diagonal Dominance

One additional difficulty arising from the extension of the scheme to viscous terms is that each block matrix of the Jacobian is the sum of at least two matrices: $\partial f^c/\partial W$ and $\partial f^v/\partial W$. The explicit expression for the spectral radius of this sum is too complex to be useful analytically, and so the spectral radius is approximated by the sum of the individual spectral radii in the scheme of Section 3.2. This approximation may artifically *reduce* the diagonal dominance of the scheme; this is noteworthy as all the other approximations we have made *increase* diagonal dominance. The problem is as follows:

For *Hermitian* matrices A and B with real eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, the inequality

$$\lambda_k(A) + \lambda_n(B) \leq \lambda_k(A+B) \leq \lambda_k(A) + \lambda_1(B), \tag{3.66}$$

holds for $k = 1, 2, \cdots, n$, see e.g. (Bhatia, 1997). A special case of this is

$$\rho\,(A+B) \leq \rho\,(A) + \rho\,(B)\,, \tag{3.67}$$

which is exactly what is needed to ensure that the sum of the spectral radii of the convective and viscous Jacobians dominates the spectral radii of their sum. Result (3.66) is also true for any matrices A and B forming part of a real vector space whose elements are matrices with real eigenvalues, and an analogous result holds for normal matrices but no such result exists in general. The Jacobians under consideration satisfy none of these conditions.

One approach to bounding the eigenvalues of

$$A + B = \frac{\partial f^c}{\partial W_c} + \frac{\partial f^v}{\partial W_c},$$

is to introduce a set of variables in which the convective and viscous Jacobians are both symmetric (and hence normal). The *parabolic symmetrizing variables* $W_{s'}$, and associated change of basis matrices, are given in Appendix A.5. Let $N = \partial W_{s'}/\partial W_c$, and A' and B' be the Jacobians in the variables $W_{s'}$, then

$$\begin{aligned} \rho(A+B) \le \|A+B\| &= \left\|N^{-1}A'N + N^{-1}B'N\right\| \\ &\le \|N^{-1}\| \cdot \|N\| \left\{\|A'\| + \|B'\|\right\} \\ &= \kappa(N)\left\{\rho(A') + \rho(B')\right\} \\ &= \kappa(N)\left\{\rho(A) + \rho(B)\right\} \end{aligned}$$

where the definition of the condition number $\kappa(A) = \|A^{-1}\| \cdot \|A\| \ge 1$ has been used, as well as the fact that $\|A\| = \rho(A)$ for a normal matrix, and that the eigenvalues of similar matrices are identical. Unfortunately $\kappa(N)$ can be very large, in particular

$$\kappa(N) \ge \rho(N)\rho(N^{-1}) = \frac{\max\left\{\sqrt{\frac{\rho^3}{p}}, \rho, \sqrt{\frac{\rho}{(\gamma-1)p}}\right\}}{\min\left\{\sqrt{\frac{\rho^3}{p}}, \rho, \sqrt{\frac{\rho}{(\gamma-1)p}}\right\}},$$

where the eigenvalues here have been read directly from the diagonal of the triangular matrix $\partial W_{s'}/\partial W_c$. An alternative may be to formulate the implicit scheme entirely in the variables $W_{s'}$ whereupon $\kappa(N) = 1$.

Further study is required to determine if the reduction of diagonal dominance due to this approximation has an effect on the stability of the scheme.

3.3.2 Von Neumann Analysis of the Scheme

Von Neumann analysis, also known as *Fourier analysis*, is a technique for determining the damping behaviour of a numerical scheme applied to a linear equation on an infinite periodic domain by considering each frequency component of the solution separately. Despite neglecting effects occurring in non-linear equations with boundary conditions, Fourier analysis has proven to be a useful tool in the study of the discretization of such equations. A basic knowledge of von Neumann analysis has been assumed in the following, see any introductory text on numerical analysis, recommended is (Hirsch, 1989). In this section von Neumann analysis is applied to a convection-diffusion equation with LU-SGS time stepping.

Modelling of the Residual

A model problem for the Navier-Stokes equations in two dimensions is the linear, scalar equation

$$\frac{\partial \psi}{\partial t} + u\frac{\partial \psi}{\partial x} + v\frac{\partial \psi}{\partial y} = \mu\left(\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2}\right) + S \cdot \psi, \tag{3.68}$$

i.e. a convection-diffusion equation with constant coefficients and a source term. Consider a finite uniform rectangular mesh aligned with the coordinate axes, with a cell spacing Δx and Δy. A discretization of (3.68) on this mesh with a general formulation of the convective terms and central differencing for the viscous terms is

$$\begin{aligned} Z \cdot \psi_{ij} &= Z_C \cdot \psi \\ &- \Lambda_{\mu x}(\psi_{i-1j} - 2\psi_{ij} + \psi_{i+1j}) \\ &- \Lambda_{\mu y}(\psi_{ij-1} - 2\psi_{ij} + \psi_{ij+1}) \\ &- S \cdot \psi_{ij}, \end{aligned} \tag{3.69}$$

where Z is the Fourier symbol corresponding to the residual R of (3.68), and Λ are the eigenvalues of the spatial operators:

$$\begin{aligned} \Lambda_x &= u\Delta y, & \Lambda_y &= v\Delta x, \\ \Lambda_{\mu x} &= \mu\frac{\Delta y}{\Delta x}, & \Lambda_{\mu y} &= \mu\frac{\Delta x}{\Delta y}. \end{aligned} \tag{3.70}$$

It is expected that the combination of LHS and RHS discretizations plays an important role in the damping properties of the implicit scheme, hence several RHS discretizations are considered. A second-order upwind scheme is modelled by

$$\begin{aligned} Z_{CU} \cdot \psi_{ij} &= \frac{1}{2}\Lambda_x(3\psi_{ij} - 4\psi_{i-1j} + \psi_{i-2j}) \\ &+ \frac{1}{2}\Lambda_y(3\psi_{ij} - 4\psi_{ij-1} + \psi_{ij-2}), \end{aligned} \tag{3.71}$$

but a discretization that corresponds better to the JST scheme is central differencing with a 4th-order dissipation term,

$$\begin{aligned} Z_{CC} \cdot \psi_{ij} &= \frac{1}{2}\Lambda_x(\psi_{i+1j} - \psi_{i-1j}) + \frac{1}{2}\Lambda_y(\psi_{ij+1} - \psi_{ij-1}) \\ &+ \Lambda_x d_x(\psi_{i+2j} - 4\psi_{i+1j} + 6\psi_{ij} - 4\psi_{i-1j} + \psi_{i-2j}) \\ &+ \Lambda_y d_y(\psi_{ij+2} - 4\psi_{ij+1} + 6\psi_{ij} - 4\psi_{ij-1} + \psi_{ij-2}), \end{aligned} \tag{3.72}$$

where d_x, d_y are the dissipation coefficients. A deficiency of this model is that the dissipation is calculated independently for each coordinate direction, which is typically not the case for unstructured flow solvers. The *Tau*-code for example uses a *non-divided Laplacian* to approximate a second-order dissipation, and a Laplacian of a Laplacian for fourth-order dissipation. A corresponding operator in this context is

$$D_{2nd} \cdot \psi_{ij} = \Delta y(\psi_{i+1j} - 2\psi_{ij} + \psi_{i-1j}) + \Delta x(\psi_{ij+1} - 2\psi_{ij} + \psi_{ij-1}), \tag{3.73}$$

from which a fourth-order difference operator may be constructed

$$\begin{aligned} D_{4th} \cdot \psi_{ij} &= \Lambda_x D_{2nd} \cdot (\psi_{i+1j} - 2\psi_{ij} + \psi_{i-1j}) \\ &+ \Lambda_y D_{2nd} \cdot (\psi_{ij+1} - 2\psi_{ij} + \psi_{ij-1}), \end{aligned} \tag{3.74}$$

resulting in a convective flux of

$$\begin{aligned} Z_{CC,iso} \cdot \psi_{ij} &= \frac{1}{2}\Lambda_x(\psi_{i+1j} - \psi_{i-1j}) + \frac{1}{2}\Lambda_y(\psi_{ij+1} - \psi_{ij-1}) \\ &+ \frac{1}{2}d_{xy} D_{4th} \cdot \psi_{ij}, \end{aligned}$$

where the single dissipation coefficient is d_{xy}.

The Fourier symbols of these operators are obtained by substituting for a harmonic solution with amplitude $\hat{\psi}$

$$\psi = \hat{\psi}\mathrm{e}^{I(i\phi_x + j\phi_y)}, \tag{3.75}$$

where the phase angles ϕ_x, ϕ_y vary in the range $[-\pi, \pi)$, and thereby cover all frequencies up to the highest frequency representable on the mesh. The Fourier symbols of the upwind and directional central scheme are then

$$\begin{aligned} \tilde{Z}_{CU} &= \Lambda_x \left\{ I \sin\phi_x (2 - \cos\phi_x) + (1 - \cos\phi_x)^2 \right\} \\ &+ \Lambda_y \left\{ I \sin\phi_y (2 - \cos\phi_y) + (1 - \cos\phi_y)^2 \right\}, \qquad (3.76) \\ \tilde{Z}_{CC} &= \Lambda_x \left\{ I \sin\phi_x + 4d_x (1 - \cos\phi_x)^2 \right\} \\ &+ \Lambda_y \left\{ I \sin\phi_y + 4d_y (1 - \cos\phi_y)^2 \right\}, \qquad (3.77) \end{aligned}$$

while the Fourier symbol of the non-directional central scheme may be constructed by considering the symbol of the second-order dissipation (3.73) with an offset (λ_x, λ_y)

$$\begin{aligned} \tilde{D}_{2nd}^{(\lambda_x,\lambda_y)} &= \Delta y \left\{ \mathrm{e}^{I((\lambda_x+1)\phi_x + \lambda_y\phi_y)} - 2\mathrm{e}^{I(\lambda_x\phi_x + \lambda_y\phi_y)} + \mathrm{e}^{I((\lambda_x-1)\phi_x + \lambda_y\phi_y)} \right\} \\ &+ \Delta x \left\{ \mathrm{e}^{I(\lambda_x\phi_x + (\lambda_y+1)\phi_y)} - 2\mathrm{e}^{I(\lambda_x\phi_x + \lambda_y\phi_y)} + \mathrm{e}^{I(\lambda_x\phi_x + (\lambda_y-1)\phi_y)} \right\}, \qquad (3.78) \end{aligned}$$

using which the symbol of the fourth-order dissipation (3.74) is

$$\begin{aligned} \tilde{D}_{4th} &= \Lambda_x \left(\tilde{D}_{2nd}^{(1,0)} - 2\tilde{D}_{2nd}^{(0,0)} + \tilde{D}_{2nd}^{(-1,0)} \right) \\ &+ \Lambda_y \left(\tilde{D}_{2nd}^{(0,1)} - 2\tilde{D}_{2nd}^{(0,0)} + \tilde{D}_{2nd}^{(0,-1)} \right), \qquad (3.79) \end{aligned}$$

and therefore the symbol of the central scheme with isotropic dissipation is

$$\tilde{Z}_{CC,iso} = I(\sin\phi_x + \sin\phi_y) + \frac{1}{2} d_{xy} \tilde{D}_{4th}. \tag{3.80}$$

Modelling of LU-SGS

Note that because the model equation is linear, the error satisfies the same equation as the solution. Any explicit or implicit scheme used to solve the discretized model equation will be stable if the amplitude $\hat{\psi}$ of every harmonic of the error does not grow in time, that is if

$$|g| = \left| \frac{\hat{\psi}^{n+1}}{\hat{\psi}^n} \right| \leq 1 \quad \forall \phi_x, \phi_y, \tag{3.81}$$

where g is known as the *amplification factor* or *damping factor* of the combined time-space discretization.

For the problem on the structured grid the LHS of the implicit system (3.8) may be written

$$\left\{ \frac{I}{\Delta t} + \frac{\partial R}{\partial \psi} \right\} = \left\{ \frac{I}{\Delta t} + A + B + A_\mu + B_\mu + T \right\}, \tag{3.82}$$

where A, B are the convective Jacobians, A_μ, B_μ the viscous Jacobians and T the source term Jacobian. The discrepancy between the fluxes and the Jacobians in the true scheme is taken into account by using a first-order upwind discretization for

A and B, as well as an implicit damping coefficient ω_{relax} for the dissipation. The viscous Jacobians are taken to be exact with respect to the RHS. Corresponding to (3.23) it may be shown that

$$\begin{aligned} A^{\pm} &= \frac{1}{2}\left(\Lambda_x \pm \omega_{relax}|\Lambda_x|\right), \\ B^{\pm} &= \frac{1}{2}\left(\Lambda_y \pm \omega_{relax}|\Lambda_y|\right), \\ A_\mu &= \Lambda_{\mu x} \\ B_\mu &= \Lambda_{\mu y} \\ T &= S \end{aligned} \tag{3.83}$$

where $+$ and $-$ denote differentials with respect to ψ at points $i+1$ / $j+1$ and $i-1$ / $j-1$ respectively. If the grid points are lexicographically ordered then the matrix operators are

$$\begin{aligned} L\cdot\psi_{ij} &= -A^{+}\psi_{i-1j} - B^{+}\psi_{ij-1} - A_\mu\psi_{i-1j} - B_\mu\psi_{ij-1}, \\ U\cdot\psi_{ij} &= A^{-}\psi_{i+1j} + B^{-}\psi_{ij+1} - A_\mu\psi_{i+1j} - B_\mu\psi_{ij+1}, \\ D\cdot\psi_{ij} &= \left\{\frac{1}{\Delta t} + \omega_{relax}\left(|\Lambda_x| + |\Lambda_y|\right) + 2(A_\mu + B_\mu) + T\right\}\psi_{ij}, \end{aligned} \tag{3.84}$$

with Fourier symbols

$$\begin{aligned} \tilde{L} &= -A^{+}\mathrm{e}^{-I\phi_x} - B^{+}\mathrm{e}^{-I\phi_y} - A_\mu\mathrm{e}^{-I\phi_x} - B_\mu\mathrm{e}^{-I\phi_y}, \\ \tilde{U} &= A^{-}\mathrm{e}^{I\phi_x} + B^{-}\mathrm{e}^{I\phi_y} - A_\mu\mathrm{e}^{I\phi_x} - B_\mu\mathrm{e}^{I\phi_y}, \\ \tilde{D} &= \left\{\frac{1}{\Delta t} + \omega_{relax}\left(|\Lambda_x| + |\Lambda_y|\right) + 2(A_\mu + B_\mu) + T\right\}. \end{aligned} \tag{3.85}$$

By noting that the Fourier symbol of an inverse operator is the reciprocal of the Fourier symbol of the original operator,

$$\widetilde{L^{-1}} = \frac{1}{\tilde{L}}, \tag{3.86}$$

the amplification factor of LU-SGS may be deduced directly from (3.12), using (3.85):

$$|g| = \left|1 - \frac{\tilde{Z}}{(\tilde{L}+\tilde{D})\tilde{D}^{-1}(\tilde{U}+\tilde{D})}\right|. \tag{3.87}$$

Influence of the Source Term

The presence of the source term S in the model equation (3.68) is an attempt to represent the source terms occurring in the turbulence equations, and has been studied extensively in the context of von Neumann analysis (Faßbender, 2003). The coefficient S is taken to be constant, i.e. uniform production or destruction over the entire domain. This equation differs slightly from the typical convection diffusion model equation, and so we consider firstly some of its properties.

The exact solution of linear equation on a periodic domain may be found by substituting in a Fourier mode and solving for the coefficient a of t in the exponent. A two parameter family of exact solutions of the scalar model equation is then

$$\begin{aligned} \psi &= \hat{\psi} \exp\left\{ I\left(\frac{x}{\Delta x}\phi_x + \frac{y}{\Delta y}\phi_y \right) + at \right\}, \\ a &= -I\left(u\frac{\phi_x}{\Delta x} + v\frac{\phi_y}{\Delta y} \right) - \mu\left(\frac{\phi_x^2}{\Delta x^2} + \frac{\phi_y^2}{\Delta y^2} \right) + S, \end{aligned}$$

where x and y have been normalized using Δx and Δy such that when $\phi_x = \pi$ or $\phi_y = \pi$ the mode's frequency corresponds to the highest frequency representable on the mesh. By Fourier's Theorem these are *all* solutions - up to linear superposition - of the model equation. As may be immediately seen, a mode grows exponentially in time if

$$\frac{S}{\mu} > \frac{\phi_x^2}{\Delta x^2} + \frac{\phi_y^2}{\Delta y^2}, \tag{3.88}$$

i.e. for an ellipse in the phase plane, and is otherwise stable. A consequence is that for positive S the solution always grows unboundedly for sufficiently low frequencies. Note that due to the linearity of the model equation, any error in an approximate solution satisfies the same equation as the solution itself, and hence also grows exponentially for the frequencies specified by condition (3.88).

Given that the model equation *itself* is "unstable" for these low frequencies, it is unreasonable to expect numerical methods to be stable there, and indeed Fourier analysis shows that they are typically not. Figure 3.2 shows the damping factor $|g|$ for a 4-stage Runge-Kutta scheme for the model equation with source term; in particular $S = 1/20$, $\mu = 1/10$, $\Delta x = 1$ and $\Delta y = 2/5$. In the centre of the contour plot it can be seen that the amplification factor increases above 1, taking a maximum value at $(\phi_x, \phi_y) = (0, 0)$. The central, bold-dashed ellipse shows the limit of the stability region of the model equation, as given by (3.88). It can be seen that the method is only unstable when the model equation itself has an unbounded solution.

In this case the natural criteria for judging the stability of a numerical method, namely that the amplifaction factor is always less than or equal to one, is no longer suitable. The only alternative is to suggest that a method be considered stable if the amplification factor in the region of stability of the model problem is always less than one. Such a property we will call *compatibly stable.* Then the method of Figure 3.2 is not stable, but it is compatibly stable, as the scheme only amplifies the error where the model equation does.

It is of interest to determine for which values of S, μ and Δt the methods described above are compatibly stable, which may be done by determining at which frequencies the amplification factor passes through 1. For compatible stability in one dimension we require that

$$|g(\phi)| \leq 1 \quad \text{for all} \quad \phi \in \left[\Delta x\sqrt{\frac{S}{\mu}},\, \pi \right].$$

Since in the cases above $|g|$ is a continuously differentiable function of ϕ, and varies continuously with S, μ and Δt, it can only change from being stable to unstable when

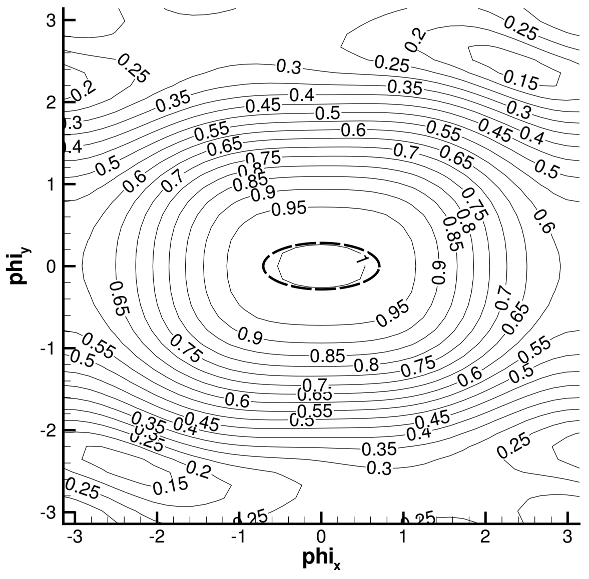


Figure 3.2: Damping factor for a 4-stage Runge-Kutta scheme applied to the model equation with a source term. The Runge-Kutta coefficients are $(\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1)$, with $\Delta t = 3/10$, and $S = 1/20$, $\mu = 1/10$, $\Delta x = 1$, $\Delta y = 2/5$, and $u = v = 1$. The central ellipse shows the limit of the stability region of the model equation.

one of its end points passes through 1, i.e.

$$\left| g\left(\Delta x\sqrt{\frac{S}{\mu}}\right)\right| = 1 \quad \text{or} \quad |g(\pi)| = 1, \tag{3.89}$$

or when a maxima or minima passes through 1, i.e. when for some $\phi = \phi_0$

$$\frac{\mathrm{d}|g|}{\mathrm{d}\phi_0} = 0 \quad \text{and} \quad |g(\phi_0)| = 1. \tag{3.90}$$

By solving these equations for e.g. Δt given S and μ, the boundaries of the stability region may be found, and hence the domain divided into compatibly stable and unstable regions. It remains then only to test a single point in each region, to determine which are stable. Unfortunately even for the simplest time stepping scheme these stability boundaries cannot be evaluated analytically and so numerical evaluation for special cases is performed.

Firstly forward Euler integration with second-order central convective fluxes, viscous fluxes with $\mu = 0.05$ and source term is tested. The amplification factor for this scheme is $|g| = |1 + \Delta t Z|$. Figure 3.3 shows the region of the $S - \Delta t$ plane where the scheme is compatibly stable and compatibly unstable, showing the amount to which the time step is limited by the influence of the source term. The lines shown are solutions of (3.89), the solutions of (3.90) are always complex in this case. Note that for $S \geq \pi^2\mu/\Delta x^2$ the region of instability of the model equation covers the entire numerical frequency domain, and therefore every numerical method is compatibly stable, here the critical value being $S \approx 0.49$.

The most striking feature of Figure 3.3 is that a small source term limits the time step more than a large source term. This is due to compatible stability being

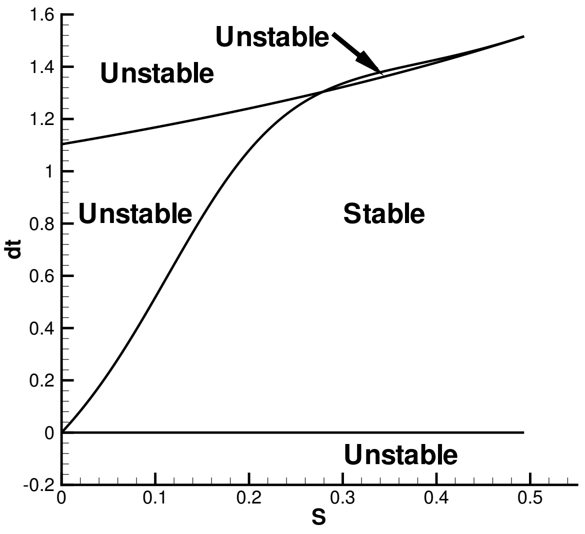


Figure 3.3: Compatible stability region of the forward Euler time stepping scheme for a range of source terms S with respect to the time step Δt. The coefficient of viscosity μ is 0.05, the convective fluxes are modelled with a central scheme.

a stronger condition for small source terms, as the region of ϕ for which the scheme must be stable is larger. At any rate the source term has a limiting effect on the time step.

In contrast, the compatible stability plot for LU-SGS shows stability for all values of the source term and all positive Δt, Figure 3.4, indicating that the source term has no bearing on the choice of time step. Hence for the model equation LU-SGS is a significantly better choice of scheme.

It remains to be seen how this analysis may be extended to two dimensions, for which the stability of LU-SGS is not assured. However algebraic determination of the stability regions may not be possible and must probably be performed numerically. There remains also the question of the applicability of these results to non-linear equations with source terms, which are of course not guaranteed to be unstable, as are linear equations. These are topics for future work.

Behaviour of LU-SGS

In Figure 3.5 the amplification factor of LU-SGS with $\omega_{relax} = 1.0$ and a purely convective RHS is plotted against the frequency components of the mode. The most remarkable feature of this result is that low frequencies are well damped, and high frequencies poorly damped. This behaviour is in stark contrast to the known behaviour of fixed-point iterations for the Euler equations (Blazek, 2001), and is a first indication that the approximations made for the Fourier analysis - in particular the reduction to a *scalar* model problem - introduce discrepancies in this case. If this result were correct there would be two surprising consequences of this behaviour:

- LU-SGS should be a poor smoother in combination with multigrid. The performance of a scheme as a multigrid smoother is determined by the amplification

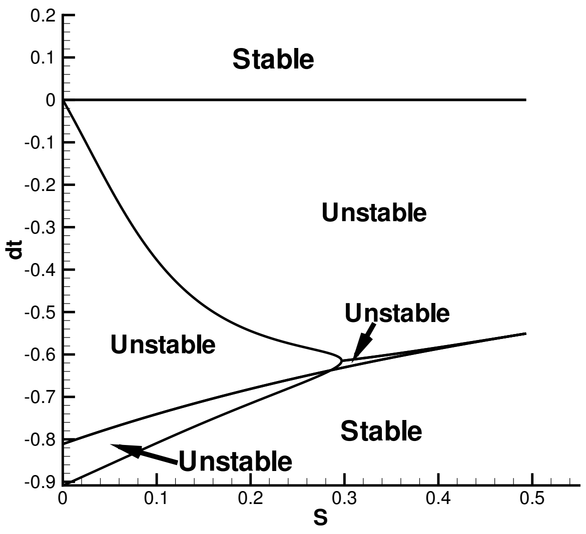


Figure 3.4: Compatible stability region of the LU-SGS time stepping scheme for a range of source terms S with respect to the time step Δt. The coefficient of viscosity μ is 0.05, the convective fluxes are modelled with a central scheme.

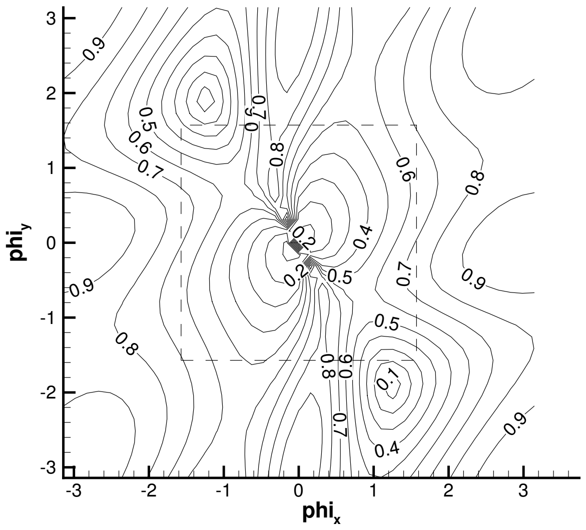


Figure 3.5: Damping factor for LU-SGS with 2nd-order upwind RHS, $\omega_{relax} = 1.0$, $\Lambda_x/\Lambda_y = 2.0$, $\Delta t = 1 \cdot 10^5$.

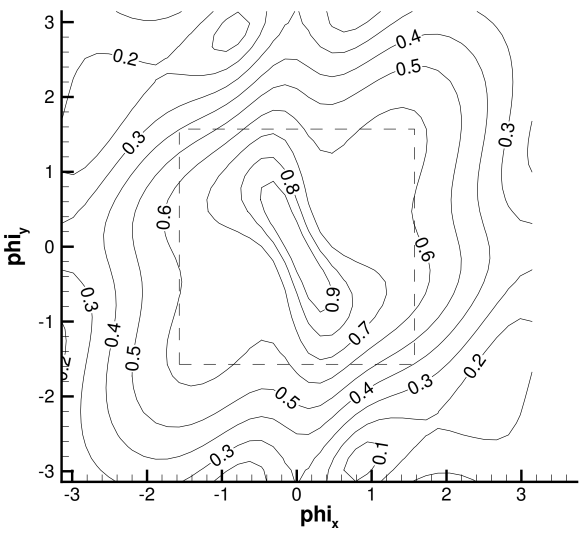


Figure 3.6: Damping factor for LU-SGS with 2nd-order upwind RHS, $\omega_{relax} = 1.5$, $\Lambda_x/\Lambda_y = 2.0$, $\Delta t = 1 \cdot 10^5$.

factor of modes lying outside the central box drawn in Figure 3.5, i.e. modes with a high frequency in at least one direction. Modes within the box will be damped at the next multigrid level where they become high frequency modes. LU-SGS seems to damp exactly the wrong modes for multigrid. This statement is borne out by Fourier analysis of a two-grid multigrid cycle with an LU-SGS smoother in (Blazek, 1993).

- By alternating iterations of LU-SGS with another method, e.g. Runge-Kutta, that damps high frequencies well, *all* frequencies should be damped well. In fact LU-SGS with $\omega_{relax} = 1.5$ has exactly this property, see Figure 3.6. The amplification factor of the combined scheme is just the product of the individual factors, and the result is shown in Figure 3.7. As can be seen the damping is excellent almost everywhere; multigrid would be superfluous in this case.

In fact it will be seen in Section 3.4 that LU-SGS for the Euler equations is at least as good a multigrid smoother as Runge-Kutta. With regard to alternating iterations of LU-SGS($\omega_{relax} = 1.0$) and LU-SGS($\omega_{relax} = 1.5$), the resulting scheme performs better than the former but worse than the latter, as might be expected if there was no significant difference in the structure of the damping properties of the two schemes.

The Fourier analysis of LU-SGS with a centrally discretized RHS produces similar results, Figure 3.8, with the best damping occurring also at low frequencies, a result that is again inconsistent with the numerical results of Section 3.4. The dashed lines in Figure 3.8 show the corresponding damping factor for isotropic artificial dissipation, which has no significant effect on the results, as might have been deduced from the strong similarity of the Fourier footprints of the two central schemes, shown in Figure 3.9 for $\Lambda_x/\Lambda_y = 2$.

In summary, the von Neumann analysis as presented here completely fails to predict the properties of LU-SGS as applied to the Euler equations. This implies that

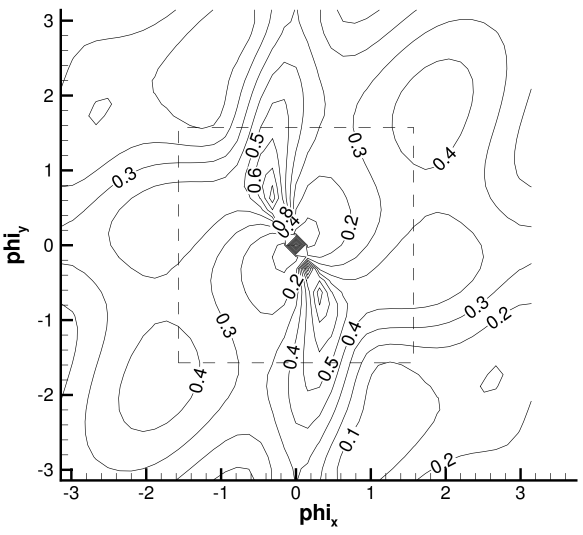


Figure 3.7: Damping factor for combined LU-SGS scheme with alternating steps of $\omega_{relax} = 1.0$, and $\omega_{relax} = 1.5$. RHS is 2nd-order upwind with $\Lambda_x/\Lambda_y = 2.0$, $\Delta t = 1 \cdot 10^5$.

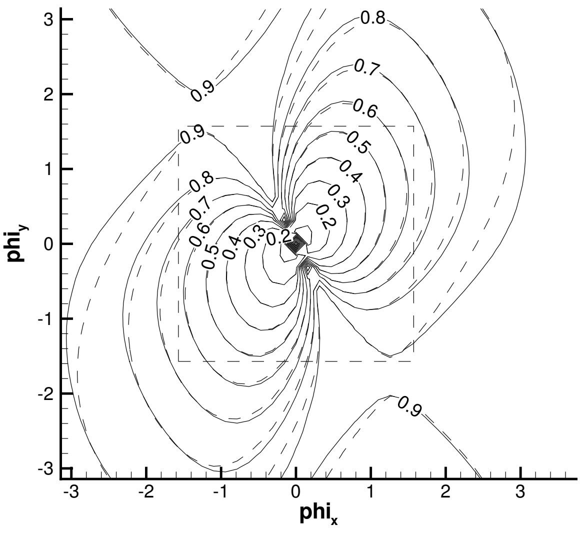
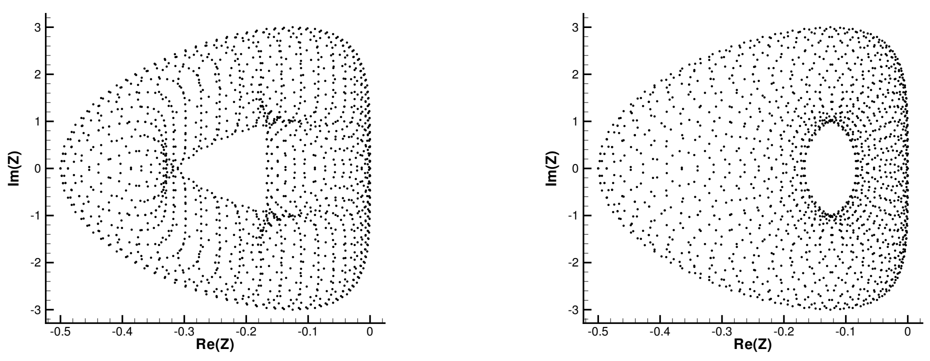


Figure 3.8: Damping factor for LU-SGS with central scheme with 3rd-order dissipation; both directional dissipation (solid contours), and isotropic dissipation (dashed contours), $d_x = d_y = d_{xy} = 1/96$.



(a) Directional dissipation. (b) Isotropic dissipation.

Figure 3.9: Footprints of central operators with 3rd-order dissipation, $d_x = d_y = d_{xy} = 1/96$.

some aspect of the discretization which is not modelled by the analysis is responsible for a significant change in the behaviour of the scheme. It is well known that boundary conditions can have a significant impact, as can irregular grids and multiple coupled equations. The latter may be treated in the context of Fourier analysis (Pierce *et al.*, 1997), whereas the analysis of the first two aspects requires another approach, for example eigenspectrum analysis (Roberts & Swanson, 2005).

Another cause of the discrepency might be the approximation of the Jacobian. For the Euler equations the Jacobian is not constant, and although the error introduced by using a different flux for the LHS than the RHS has been accounted for in (3.83), other sources of error have not. An interesting experiment is to add a constant error term ϵ_L to L in (3.84). The phase-damping plot is shown in Figure 3.10 for $\epsilon_L = 10.0$. As can be seen the damping pattern has been reversed with LU-SGS now behaving as a high-frequency smoother, consistent with the observed behaviour. However the result is sensitive to the magnitude of ϵ_L and cannot therefore be taken as a successful analysis without a method for quantifying this value.

A method of analysis that could take into account all these effects, was recently proposed in (Roberts & Swanson, 2005), whereby the complete eigensystem of a discretization is computed for a simple model problem, the unstable modes are identified and their properties examined. Such a technique will be touched upon in Section 4.3.2.

In the interests of completeness, but with no suggestion of accuracy, the damping factors for the LU-SGS scheme as applied to the full model equation (3.68) including source term, diffusion, and an anisotropic mesh, with a central RHS, are shown in Figure 3.11. Note that the central peak contains an amplification factor greater than unity due to the source term, but this peak lies entirely within the ellipse of instability of the model equation.

3.3.3 Time-Accuracy of the Scheme

Since large time steps are possible with LU-SGS, it is of interest to know whether the code could be used directly for time-accurate computations. Given the large number

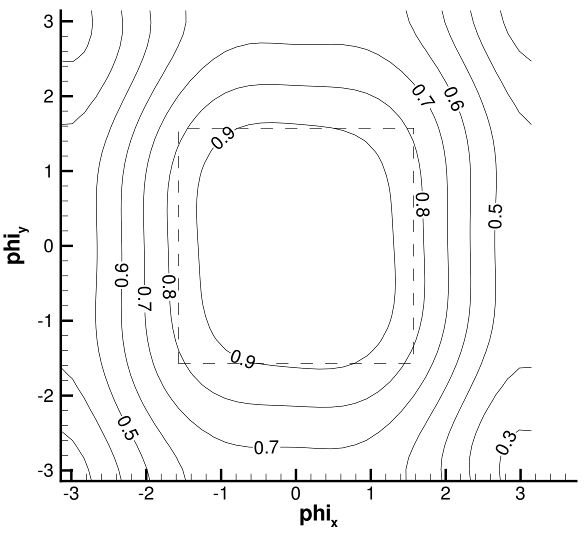


Figure 3.10: Damping factor for LU-SGS with an artifically introduced constant error of $\epsilon_L = 10$ in L and $\omega_{relax} = 1.0$. The RHS is 2nd-order upwind with $\Lambda_x/\Lambda_y = 2.0$, $\Delta t = 1 \cdot 10^5$.

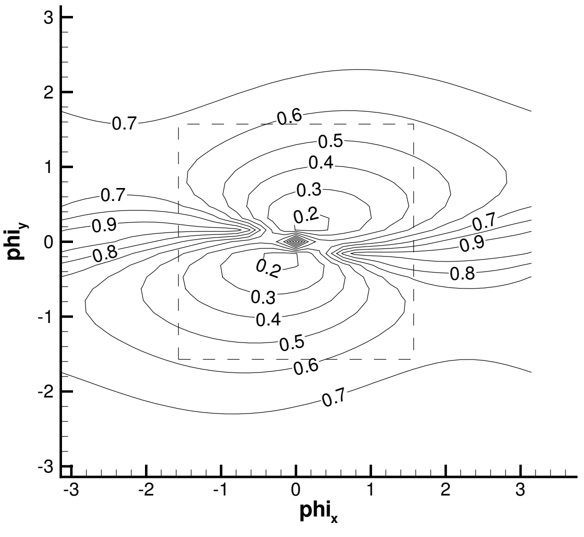


Figure 3.11: Damping factor for LU-SGS with $\omega_{relax} = 1.0$. The RHS is central with isotropic dissipation, $d_{xy} = 1/96$, $\Lambda_x/\Lambda_y = 0.2$, $\Lambda_{\mu x} = 4 \cdot 10^{-3}$, $\Lambda_{\mu y} = 1 \cdot 10^{-1}$, $S = 1 \cdot 10^{-4}$, $\Delta t = 1 \cdot 10^5$.

of approximations made in the course of the derivation, it might be supposed that the scheme is zero-th order in time, in fact it will be shown that it is first-order provided that scalar preconditioning is not used. The following discussion is taken from (Dwight, 2004), which also investigates a hybrid LU-SGS/dual-time scheme for efficient time-accurate calculations of higher order.

Time-Accuracy - Theoretical Results

The time accuracy of the scheme may be derived by considering the order of error in time introduced by the discretization and by each subsequent approximation thereof.

The time discretization may be chosen to be second-order by setting $\beta = \frac{1}{2}$ in (3.4), the backward difference formula on which the scheme is based. The error incurred by the discretization itself is then $\mathcal{O}(\Delta t^2)$. The term $R_i(W^{n+1})$ in (3.4) is approximated by a Taylor expansion, of which terms of order Δt^2 and higher are neglected. Similarly the substitution of (3.6) incurs only an $\mathcal{O}(\Delta t^2)$ error.

The error due to the LU-SGS approximation is the factorization error multiplied by the unknown update (3.13)

$$\epsilon_{LU-SGS} = \left(L \cdot D^{-1} \cdot U\right) \Delta W, \tag{3.91}$$

where L, D and U are the lower, upper and diagonal entries of

$$A = L + D + U = \left\{ \frac{\Omega_i}{\Delta t}\delta_{ij} + \frac{1}{2}\frac{\partial R_i(W^{(n)})}{\partial W_j} \right\}. \tag{3.92}$$

Since R has no Δt dependence, $\partial R/\partial W$ has no Δt dependence, and hence all terms are $\mathcal{O}(1)$ in time with the exception of $\Omega/\Delta t$ which appears only on the diagonal. Hence L and U are both $\mathcal{O}(1)$, and D is $\mathcal{O}(1/\Delta t)$. It may be shown that D^{-1} is therefore $\mathcal{O}(\Delta t)$. By (3.6) again ΔW is $\mathcal{O}(\Delta t)$, and therefore ϵ_{LU-SGS} is $\mathcal{O}(\Delta t^2)$. This provides some justification for the LU-SGS approximation itself: as the time step tends to zero the factorization error also disappears.

The final approximation is to substitute the exact Jacobian of (3.7) with the first-order Jacobian, an $\mathcal{O}(1)$ approximation. This error is then multiplied by the $\mathcal{O}(\Delta t)$ unknown update, giving a composite $\mathcal{O}(\Delta t)$ error.

In summary there are four sources of error in the time discretization:

- the discretization error of the trapezoidal scheme,
- the truncation error of the linearization of $R(W^{(n+1)})$,
- the factorization error of the LU-SGS scheme $LD^{-1}U\Delta W$,
- the error due to the first-order approximation of the flux Jacobian.

All these errors, with the exception of the last, are nominally of size $\mathcal{O}(\Delta t^2)$, but the cost of increasing the last to second-order would seem to be the cost of formulating a scheme with the exact second-order Jacobian, which leads to an entirely different class of schemes, as already discussed. Hence the method is first-order accurate, with leading-order error in time controlled by the approximation of the Jacobian.

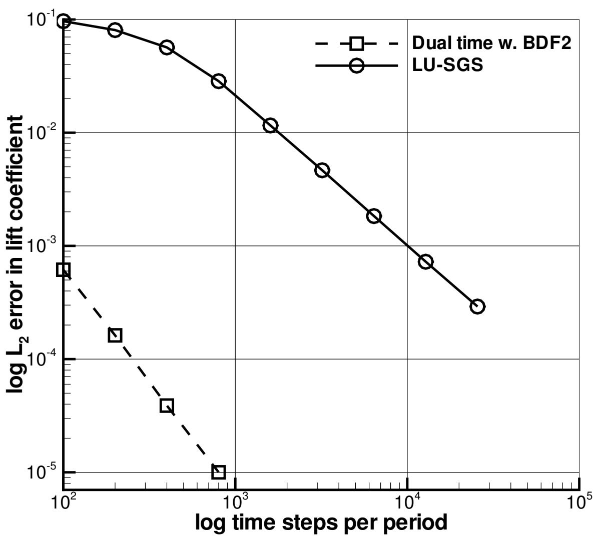


Figure 3.12: Comparison of the time accuracy of second-order dual time with that of LU-SGS.

Time-Accuracy - Numerical Results

A convergence study on the time step Δt is undertaken to numerically measure the time accuracy of the LU-SGS method. A turbulent case is chosen to reflect all approximations made to the Jacobian.

A harmonically oscillating RAE2822 aerofoil is chosen, with pitch angle $\alpha = 3^\circ \pm 2.51^\circ$, at a reduced frequency of $k = \omega l_k / V_\infty = 0.163$. An onflow Mach number of $M_\infty = 0.74$ and Reynolds number of $Re = 6 \times 10^6$ were used, and the turbulence model is Menter SST. Under these conditions the flow remains fully attached at all stages of the motion. A third-order accurate dual time method, with 200 inner iterations per time step and 1024 time steps per period (TSPP), was used to obtain a reference solution. Unsteady solutions obtained with LU-SGS, and with a second-order accurate dual time, for varying Δt were compared with the reference solution using the L_2 norm of the difference in calculated lift coefficients over one period, once transients had disappeared. The results are plotted in Figure 3.12.

Both the second-order behaviour of dual time and the first-order behaviour of LU-SGS are readily apparent, confirming the theory of Section 3.3.3. However the breakdown of the time accurate behaviour of LU-SGS occurs at relatively large values of TSPP (about 300), which is probably a result of the Δx dependence of the factorization error in very thin viscous wall cells. Further note that, in terms of absolute error, LU-SGS reaches the accuracy of 100 TSPP dual time with 10,000 TSPP. Since one dual time inner iteration costs roughly as much as one LU-SGS step, the two methods have roughly equal efficiency (assuming 100 dual time inner iterations per step) for this level of accuracy.

Given this behaviour it is suggested that LU-SGS in time accurate form may be an efficient method in applications where very small time steps are necessary anyway due to physical considerations, e.g. LES. However it is clear that in any such application, care must be taken that the time step is small enough for LU-SGS to behave in a time accurate manner.

3.4 Numerical Results

The purpose of this section is twofold. Firstly it is demonstrated that, after the two major approximations of LUSGS: (a) considering only a first-order Jacobian, and (b) solving the linear system with one SGS iteration, the influence of the subsequent approximations made in Section 3.2 (in particular the use of the Lax-Friedrichs flux and the replacement of the viscous flux Jacobians by their eigenvalues) on the non-linear convergence is minor.

Secondly it is demonstrated that the algorithm thus derived may be applied to a large case of practical engineering interest. Runge-Kutta with residual smoothing and multigrid serves as a reference algorithm throughout, representing as it does the next-best available algorithm.

Within this section the convergence of various algorithms must be compared. In order that performance may be uniformly judged - and to avoid numerous convergence plots - some metric for speed of convergence must be decided upon. A common metric is the convergence rate, which is typically constant as the solution asymptotically approaches the exact solution. However, given that cases of engineering interest typically involve convergence of no more than four orders of magnitude, and that convergence behaviour in this range tends to be highly non-uniform, the convergence rate varies considerably. Instead the following measure is used: the number of iterations (or normalized CPU time) from the beginning of the calculation required for the normalized residual to drop and remain below 1×10^{-3} and 1×10^{-4}. The residual in all cases is normalized against the residual calculated on the first step.

A second metric of the algorithm is taken, which has already been used in Section 3.3.1: the number of linear iterations of SGS needed to drive the (also normalized) linear residual below 1×10^{-8}. This provides a measure of the relative diagonal dominance of the Jacobian, or the relative stability of the Gauss-Seidel iteration, for two different methods. It will be used to verify the theoretical results of Section 3.3.

3.4.1 Approximations of the Jacobian

In Section 3.2 it was discussed how the use of a Jacobian derived from first-order fluxes reduces the memory requirements and complexity of the scheme. In contrast some of the other approximations made are useful but not essential. For example the choice of the Lax-Friedrichs flux, which reduces the diagonal-block entries of the Jacobian to diagonal matrices, could be replaced by a Roe flux, whose Jacobian contains significantly more information about the flow. Hence each such approximation is evaluated numerically for its effect on the linear and non-linear convergence rates.

First however it is useful to observe the effect the reduction of the order of the Jacobian from second to first has on the convergence of the non-linear system, independently of other Jacobian approximations and given exact solution of the intermediate linear systems. Refer to Section 1.1 for an overview of the types of implicit methods discussed here.

Two implicit schemes, one using exact 1st-order Jacobians, and the other using exact 2nd-order Jacobians are constructed in a Jacobian-Free manner using finite differences on the residual vector,

$$\frac{\partial R}{\partial W}\Delta W \approx \frac{R(W+\varepsilon\Delta W)-R(W)}{\varepsilon},$$

where ε is chosen small enough that the perturbation to R is approximately linear, while large enough that the machine rounding error incurred through the subtraction of two similar floating-point quantities is small. In general this quantity must be set carefully in order to obtain quadratic convergence, but for the following simple cases the choice

$$\varepsilon = \frac{1\times 10^{-8}}{\|\Delta W\|},$$

was sufficient. This method also makes it easy to compare various LHS Jacobians for a given RHS discretization, without evaluating the Jacobians explicitly by hand. The systems are solved using a GMRES Krylov solver, preconditioned with the LU-SGS method, and converged to machine accuracy for each Newton step.

The non-linear convergence of these two schemes applied to an unstructured grid about a 2d Euler NACA0012 aerofoil shown in Figure 3.13, is plotted in Figure 3.14, whereby second-order van Leer flux-vector splitting (Van Leer, 1982) is used on the RHS.

The method with the exact van Leer second-order Jacobian is a pure Newton's method, and therefore converges quadratically, and achieves machine accuracy in 5 iterations. Reducing the Jacobian to one based on first-order van Leer fluxes has a dramatic effect on the convergence, as a factor 20 more iterations are required. Compared to this, the effect of introducing a discrepancy between the particular flux used on the LHS and RHS is insignificant, even when a central scheme (JST) rather than an upwind scheme is used.

On the other hand a mismatch between flux functions with a second-order Jacobian does play a significant role, seen in Figure 3.15 this time for a viscous NACA-64A010 aerofoil calculation, as here the method is reduced from a Newton method with quadratic convergence, to an approximate Newton method with merely linear convergence. As a result the number of non-linear iterations required nearly trebles.

These results are intended to demonstrate that the approximation of the Jacobian to first-order has a significantly greater effect on the non-linear convergence of the method, than subordinate approximations such as the choice of flux function. Hence many useful approximations may be made to the Jacobians without fear of damaging convergence too much.

Note that the finite difference-based Newton method described here is not competitive with the LU-SGS method based on the first-order Jacobian, principally because of the necessity of repeatedly evaluating the non-linear residual, which is one of the most expensive solver operations.

3.4.2 Influence of Linear Solver

LU-SGS is now compared to two other linear solvers: SGS(n) and the block-Jacobi iteration, Jacobi(n), where in both cases n is the number of inner iterations. The SGS scheme is described in Section 3.2.1. An identical Jacobian - as described in

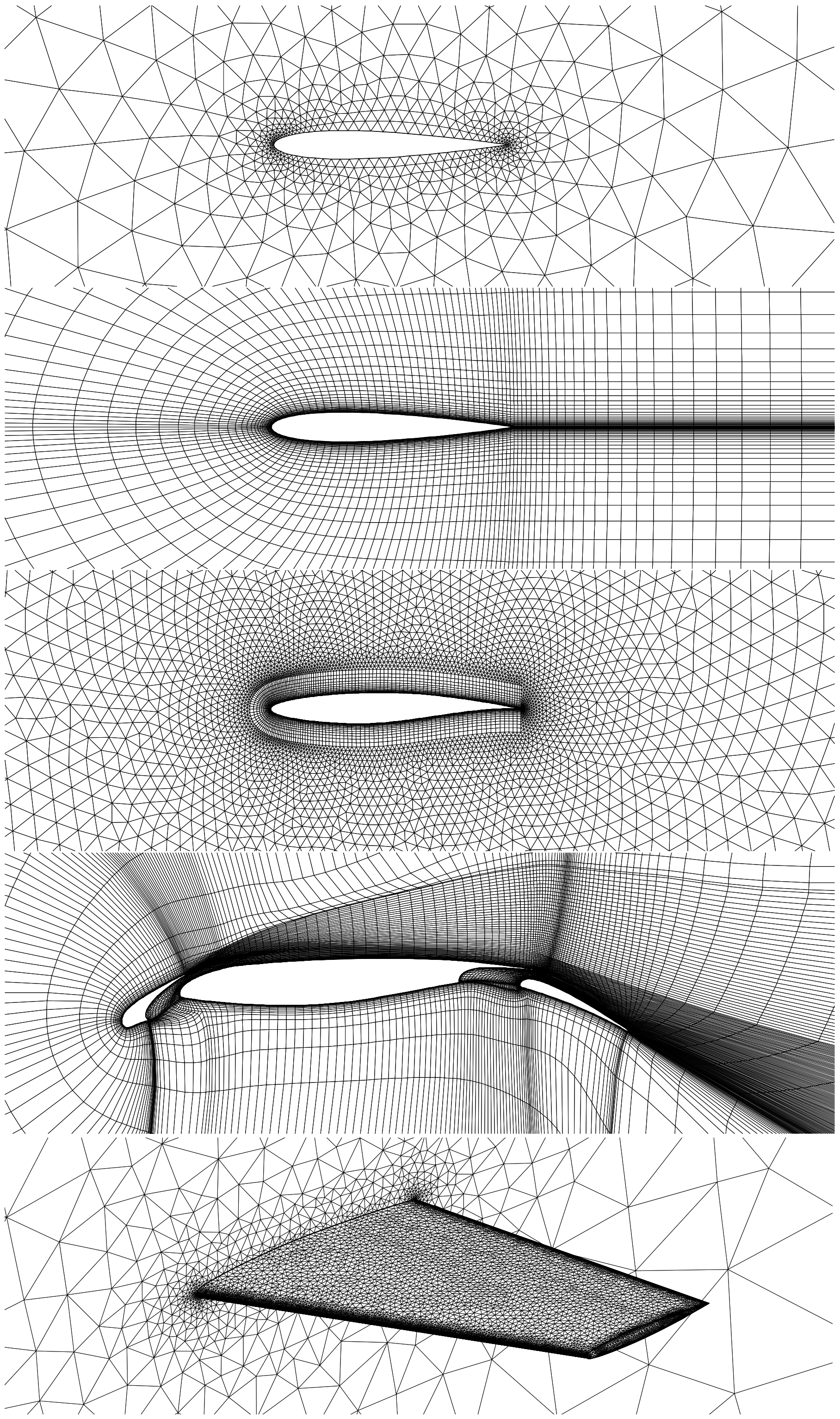


Figure 3.13: Some of the grids used in this thesis. From top to bottom: a NACA0012 Euler grid, a NACA0012 grid for laminar flow at Re = 5000, a RAE2822 grid for turbulent flow at Re = 6×10^6, a EUROLIFT II high-lift configuration grid, and a ONERA M6 wing Euler grid.

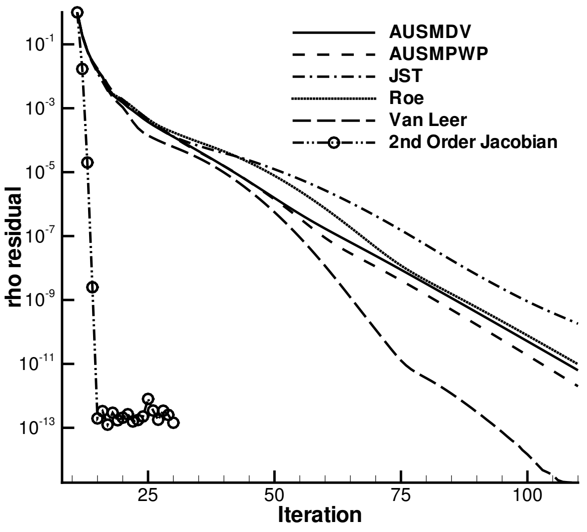


Figure 3.14: Convergence behaviour of a Newton algorithm with 2nd-order van Leer LHS and RHS (labeled *2nd-Order Jacobian*), and an approximate Newton algorithm with a 1st-order van Leer LHS, for various second-order RHS fluxes including van Leer. The linear systems at each non-linear step are converged to machine accuracy. The case is an inviscid NACA0012 aerofoil.

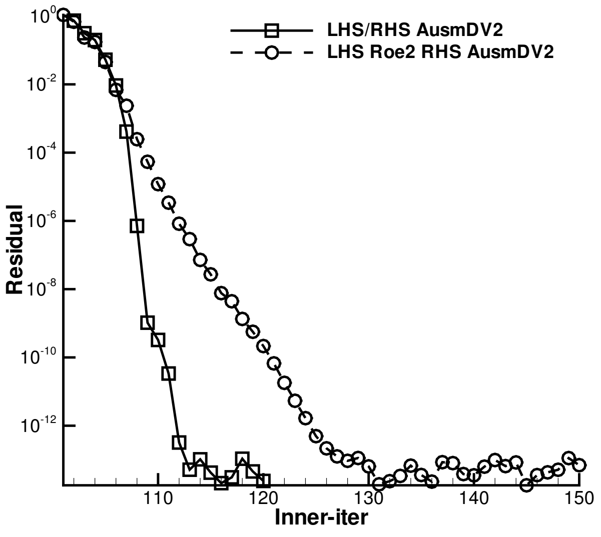


Figure 3.15: Convergence behaviour of an implicit scheme with second-order van Leer LHS, for second-order van Leer and Roe flux RHS. The linear systems at each non-linear step are converged to machine accuracy. The case is a viscous NACA64A010 aerofoil.

Linear Solver	Non-Linear Iterations	Normalized Time for One Iteration	Normalized Time for Iterations
Runge-Kutta	553 (775)	1.09	2.12 (2.97)
LU-SGS	284 (401)	1.00	1.00 (1.41)
SGS(1)	284 (401)	2.19	2.19 (3.09)
SGS(2)	177 (239)	3.98	2.48 (3.35)
SGS(3)	129 (178)	5.13	2.33 (3.22)
SGS(4)	104 (152)	7.78	2.84 (4.16)
SGS(6)	92 (127)	9.52	3.08 (4.26)
SGS(10)	85 (135)	14.64	4.38 (6.87)
SGS(20)	97 (144)	16.21	5.54 (8.22)
Jacobi(1)	725 (1019)	0.92	2.35 (3.30)
Jacobi(2)	567 (816)	1.88	3.75 (5.40)
Jacobi(3)	475 (589)	2.91	4.87 (6.04)
Jacobi(4)	399 (610)	3.72	5.22 (7.99)
Jacobi(6)	315 (434)	4.67	5.18 (7.14)
Jacobi(10)	253 (344)	7.50	6.68 (9.08)
Jacobi(20)	210 (315)	9.82	7.26 (10.9)

Table 3.2: For a variety of linear solvers, and given Jacobian formulation, the number of non-linear iterations required to reach a residual of $1 \times 10^{-3}(1 \times 10^{-4})$, the normalized CPU time for each such iteration, and the normalized CPU time for the iterations required to reach a residual of $1 \times 10^{-3}(1 \times 10^{-4})$. The normalization is performed with respect to the LU-SGS scheme. This for a 2d inviscid NACA0012 aerofoil calcuation in a transonic regime.

Section 3.2 - is used for all methods, in order to compare the methods fairly[2].

Compared are the number of non-linear iterations required for convergence, the CPU time required for one non-linear iteration (normalized), and the CPU time required for convergence (again normalized). The results for the inviscid 2d NACA0012 transonic aerofoil calculation are given in Table 3.2. No multigrid is applied in order to observe the effects of the schemes in isolation.

Note that - as expected - the iteration counts of LU-SGS and SGS(1) are identical, however the cost of the single SGS iteration is approximately twice that of LU-SGS. This effect is due to the extra triangular matrix multiplication required per iteration for SGS; compare (3.10) and (3.11). If this extra multiplication were not necessary, the reduction in the number of non-linear iterations required by SGS(n) might make it more efficient overall than LU-SGS for some n; as it is, it is alway less efficient.

Consider now the relative cost per iteration of Runge-Kutta and LU-SGS. The overhead of Runge-Kutta is principally the re-evaluation of the residual at each stage,

[2]Of course it may be the case that the combination of a particular Jacobian approximation with a particular solver produces the optimal time-efficiency for the non-linear convergence. In fact, in the absence of theory, all combinations of solver and Jacobian must be tested, and this brute-force investigation has been performed. The results of this chapter are intended to be illustrative of the most significant tradeoffs that occur.

whereas the SGS algorithms require only one evaluation of the residual for each non-linear iteration. On the other hand the SGS algorithms must compute the Jacobian and perform two Gauss-Seidel sweeps per inner iteration. If the residual is very expensive to compute, it will drive up the cost of Runge-Kutta, and LU-SGS will be comparatively efficient[3]. As it happens, the two extra residual evaluations are more expensive than the LU-SGS overhead, and hence LU-SGS is more efficient per step. Here the cheapest available flux is used (the JST flux), and hence the difference in CPU time per step is relatively small; for matrix dissipation with preconditioning LU-SGS is considerably faster per step than Runge-Kutta.

Jacobi(1) is slightly cheaper than LU-SGS, but Jacobi has poorer convergence behaviour than Gauss-Seidel and the CFL condition limits the time step. Jacobi(2) is already twice as expensive as LU-SGS, but to get the same convergence behaviour with Jacobi, between 6 and 10 inner steps are required, at which point the expense per step is prohibitive.

The reason that Jacobi iterations perform so poorly for this case is that the diagonal-blocks of the Jacobian are simply multiples of the identity matrix, and hence one step of (3.14) is equivalent to a forward-Euler method with a local time step. In constrast, Jacobi iterations have been shown to be effective when applied to a spatial discretization with matrix dissipation (Swanson *et al.*, 2005; Mavriplis, 1998; Pierce *et al.*, 1997), where the coefficient of the dissipative fluxes contains the matrix $|\partial f^c/\partial W|$ rather than simply the maximum eigenvalue thereof. In this case the diagonal-block of the Jacobian contains significantly more flow information, and as might be expected the method is therefore superior[4]. The use of such Jacobians with scalar dissipation is however ineffective due to the mismatch between the residual and Jacobian.

In conclusion LU-SGS is at least twice as efficient as the next best method tested for the given Jacobian and the central discretization with scalar dissipation.

3.4.3 Influence of Viscous Flux Approximation

Three formulations of the viscous TSL Jacobian proposed in Section 3.2.3 are numerically compared, from most accurate to least accurate they are:

Exact TSL No approximations are made, the TSL block Jacobian contributions are exactly those given in Section 2.8.1.

Diagonalized Diagonal The TSL block Jacobians that occur on the diagonal are replaced by the sum of their spectral radii, thus diagonalizing the block diagonal as described in Section 3.2.3.

Fully Diagonalized All TSL block Jacobians occurring in the system matrix are replaced by their spectral radii, thus removing all coupling of the equations over the viscous terms.

[3]Runge-Kutta has here the advantage that the dissipation in the JST scheme is only calculated on the first stage.

[4]This method is also known as *matrix-preconditioning* in reference to local time stepping as a type of scalar preconditioning.

Jacobian approx.	SGS Iterations	Non-Linear Iterations	Normalized Time/Iter.	Normalized Time for Iterations
Runge-Kutta	-	1473 (2369)	1.18	1.81 (2.91)
Fully Diagonalized	80.2	959 (1639)	1.00	1.00 (1.71)
Diagonalized Diagonal	96.1	821 (1436)	1.07	0.92 (1.60)
Exact TSL	122.9	707 (1193)	3.12	2.30 (3.88)

Table 3.3: For three different formulations of the viscous components of the Jacobian, the number of SGS iterations required to reduce the linear residual below 1×10^{-8}, the number of non-linear LU-SGS iterations required to reduce the non-linear residual below $1 \times 10^{-3}(1 \times 10^{-4})$, the normalized time for one iteration, and the normalized time for the required iterations. The normalization is performed with respect to the Fully Diagonalized scheme. The multigrid cycle is 3W, and the turbulence model is the NLR TNT model. This for the NACA64A010 aerofoil at transonic conditions.

Jacobian approx.	SGS Iterations	Non-Linear Iterations	Normalized Time/Iter.	Normalized Time for Iterations
Runge-Kutta	-	1921 (2708)	1.18	1.80 (2.54)
Fully Diagonalized	108.4	1259 (2123)	1.00	1.00 (1.69)
Diagonalized Diagonal	120.1	1063 (1817)	1.09	0.92 (1.57)
Exact TSL	156.8	953 (1666)	3.27	2.48 (4.33)

Table 3.4: As for Table 3.3 but for the ONERA M6 wing with $\alpha = 2.0°$, $M_\infty = 0.73$, and $\mathrm{Re} = 6.2 \times 10^6$.

Each of two example test cases has been calculated with these three Jacobian formulations, whereby the convergence of the inner SGS iteration was also recorded. The results are given in Tables 3.3 and 3.4. The first case is the NACA64A010 aerofoil at an angle of attack $\alpha = 4.0°$, $M_\infty = 0.789$ and a Reynolds number $\mathrm{Re} = 11.88 \times 10^6$. The grid is similar to the RAE2822 grid shown in Figure 3.13. The second case is the finite swept ONERA M6 wing, with $\alpha = 2.0°$, $M_\infty = 0.73$, and $\mathrm{Re} = 6.2 \times 10^6$. For both cases the two-equation TNT model from the NLR is used, and the multigrid cycle is 3W. The geometry (but not the grid) may be seen in Figure 3.13. As was the case for the different approximations of the convective flux-Jacobian, the CPU time needed to evaluate the Jacobian does not vary significantly for the three schemes. However the Exact TSL scheme is the only scheme considered so far to have a non-diagonal block diagonal. The solution of the linear system therefore involves the inversion of a 5×5 matrix for each grid point for each sweep (the turbulence Jacobian is diagonal), which is performed directly using Gaussian elimination with pivoting. This introduces the extra cost per iteration seen in the penultimate column of Tables 3.3 and 3.4 for this scheme.

Otherwise as expected, the better the Jacobian approximation, the better the non-linear convergence and the worse the linear convergence. However the time-disadvantage incurred by the Exact TSL scheme reduces it to last place, the best

Algorithm	SGS Iterations	Non-Linear Iterations	Normalized Time for One Iteration
Original Grid	67.3	58 (94)	1.00
Cache-block	45.3	58 (94)	0.86
Cuthill-McKee	44.0	60 (98)	0.97
Random	70.1	60 (102)	1.95

Table 3.5: Average number of symmetric Gauss-Seidel iterations needed to converge the inner residual to 1×10^{-8} for an unstructured M6 wing case with various grid-point orderings. Also number of iterations required for convergence to 1×10^{-3} (1×10^{-4}) for LU-SGS with a 4W multigrid cycle.

choice being the Diagonalized Diagonal scheme. It is used henceforth.

3.4.4 Influence of Point Ordering

The ordering of grid points has an effect on the transport of information across the grid within a Gauss-Seidel sweep, and thus on the convergence. Table 3.5 gives performance data for LU-SGS with different grid point orderings, for a transonic flow on a fully unstructured grid Euler about an M6 wing, shown in Figure 3.13. In fact the point ordering has almost no effect on the convergence of LU-SGS, but a large effect on cache performance[5]. The use of *symmetric* Gauss-Seidel apparently reduces the dependence on point ordering considerably, making further research into better point orderings for LU-SGS unattractive.

3.4.5 Parallel Efficiency

LU-SGS is parallelized using the domain decomposition model, whereby the GS sweep is applied to each domain individually, points outside the current domain being accounted for in a Jacobi fashion (Sharov *et al.*, 2000). Hence, unlike Runge-Kutta, the convergence will depend to some extent on the grid partitioning. The magnitude of this effect is measured for the 3d, transonic M6 wing Euler test-case on an unstructured grid with $\approx$160,000 points, see Figure 3.13. The results for 1-32 processors are given in Table 3.6. As can be seen, Runge-Kutta convergence is completely independent of the number of processors, but LU-SGS only has a very weak dependence, indicating that the modifications made to the algorithm were justified.

The parallel efficiency of LU-SGS and Runge-Kutta is also of interest. From examination of the two algorithms it may be seen that for a complete 3-stage Runge-Kutta scheme the flow variables must be communicated to other domains four times, while for LU-SGS they must only be communicated twice. Lower communication

[5]Not directly related to LU-SGS, but as a matter of some interest: the cache-block algorithm for point reordering was designed to reduce cache misses on cache based processors for edge-based codes (Löhner & Galle, 2002). Based on Table 3.5 it gives an performance improvement of 15% on the original grid ordering (which ordering is due to the advancing front algorithm with which the grid was generated), which is nevertheless dwarfed by the awful performance of the random ordering. The lesson? Don't try to do badly and you'll be okay!

	Number of Processors					
Algorithm	1	2	4	8	16	32
LU-SGS 4w	60 (101)	60 (101)	61 (103)	61 (104)	61 (104)	62 (105)
RK 4w	91 (169)	91 (169)	91 (169)	91 (169)	91 (169)	91 (169)

Table 3.6: Number of iterations required for convergence to 1×10^{-3} (1×10^{-4}) for LU-SGS and Runge-Kutta with multigrid. The grid has 160,000 points.

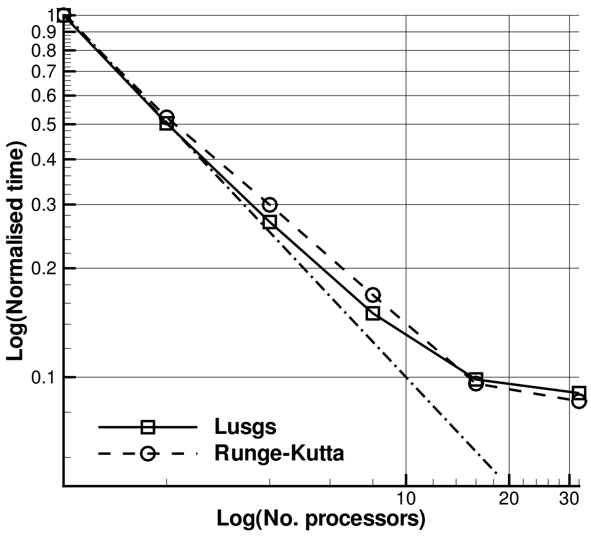


Figure 3.16: Time for one iteration - normalized against time for one iteration on one processor - against number of processors, from 1-32. The diagonal line represents perfect speed-up.

overhead should mean higher parallel efficiency, and this is measured for LU-SGS and Runge-Kutta for the same grid as above on 1-32 processors of a Intel Xeon cluster. The results are plotted in Figure 3.16, demonstrating that indeed LU-SGS shows small benefits. The tail-off at 16-32 processors is due to the small size of the grid.

3.4.6 Generic Delta-Wing

As a representative three-dimensional test case a generic delta-wing is chosen, shown in Figure 3.17. The computation is intended to approximate a wind-tunnel test, and hence the model support is taken into account. The half of the support nearest the model is modelled as a viscous wall, but the other half as a slip-wall to avoid problems at the farfield boundary; the cross-over can be seen in the pressure distribution on the support.

This case was calculated with a farfield Mach number of $M_\infty = 0.5$, with angle of attack $\alpha = 9°$, a side-flow angle $\beta = 0°$, and with zero roll. The Reynolds number is $\mathrm{Re} = 3.5 \times 10^6$. The grid is hybrid, consisting of layers of squat prisms in the

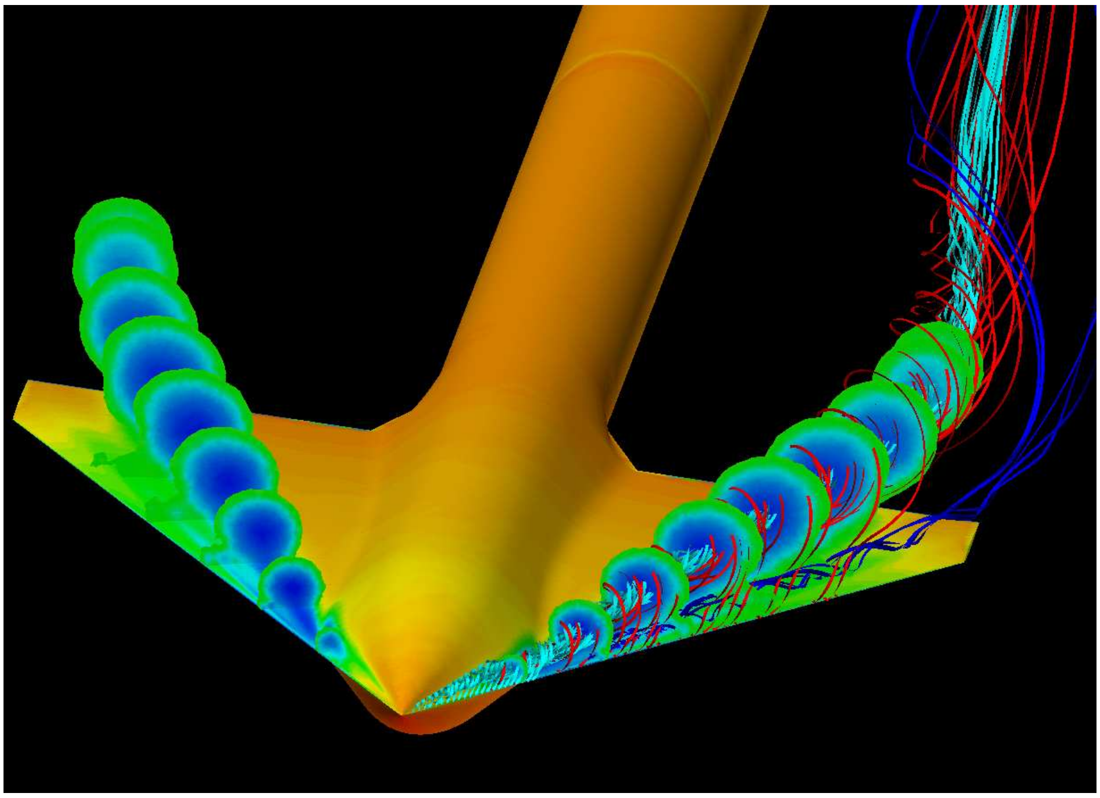


Figure 3.17: Visualization of the flow about a generic delta-wing configuration with supporting strut. Contours show the pressuse distribution on the surface and within the vortices, ribbons show the streamline. Both primary and secondary vortices are visible. (Figure courtesy of A. Schütte, DLR.)

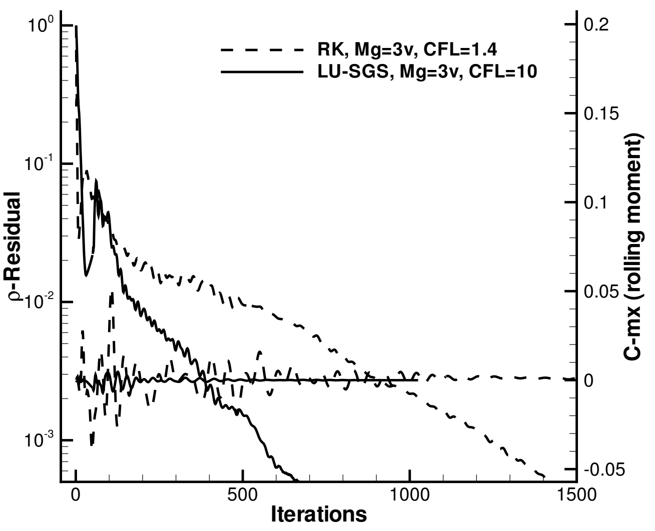


Figure 3.18: Convergence behaviour of the ρ-residual and the rolling moment for the generic delta-wing with two time stepping schemes: Runge-Kutta with CFL number of 1.4, and LU-SGS with a CFL number of 10.0. Both schemes act as smoothers in a $3V$ multigrid cycle.

boundary layer, and an unstructured mesh of tetrahedra in the field, for a total of about 3.4 million points.

The performance of Runge-Kutta and LU-SGS in terms of multigrid cycles are compared in Figure 3.18, whereby a suitable CFL number has been chosen for each method, and both methods act as smoothers in a $3V$ multigrid cycle. The calculation was performed on 48 processors of an Opteron cluster. It may be seen that LU-SGS converges approximately twice as rapidly as Runge-Kutta here, and this may be taken as a typical example of expected speed-up.

Of more interest however is the relative CPU time required for convergence using the two schemes. The time required for one iteration of LU-SGS in relation to Runge-Kutta is shown in Figure 3.19, whereby this relation is dependent on the spatial discretization. The time for one iteration is divided into time required for a single residual evaluation, time required for any residual smoothing sweeps, and time required for remaining operations.

The Runge-Kutta scheme has two main disadvantages in this competition. Firstly it requires a residual smoothing step to be efficient, whereas LU-SGS does not, adding a constant amount to the time required for one step. Secondly a three-stage Runge-Kutta scheme requires three evaluations of the residual, whereas LU-SGS requires only one. As the residual becomes more expensive the "Remainder" part of Runge-Kutta becomes more expensive, whereas for LU-SGS it stays constant[6].

The combination of improved convergence in terms of iterations, and reduced CPU time per iteration implies that for this case LU-SGS requires between $30-40\%$

[6]In fact the Runge-Kutta algorithm implemented in *Tau* evaluates the full residual only once, and on later stages evaluates only the convective terms of the residual. For the central scheme the dissipation is also not reevaluated.

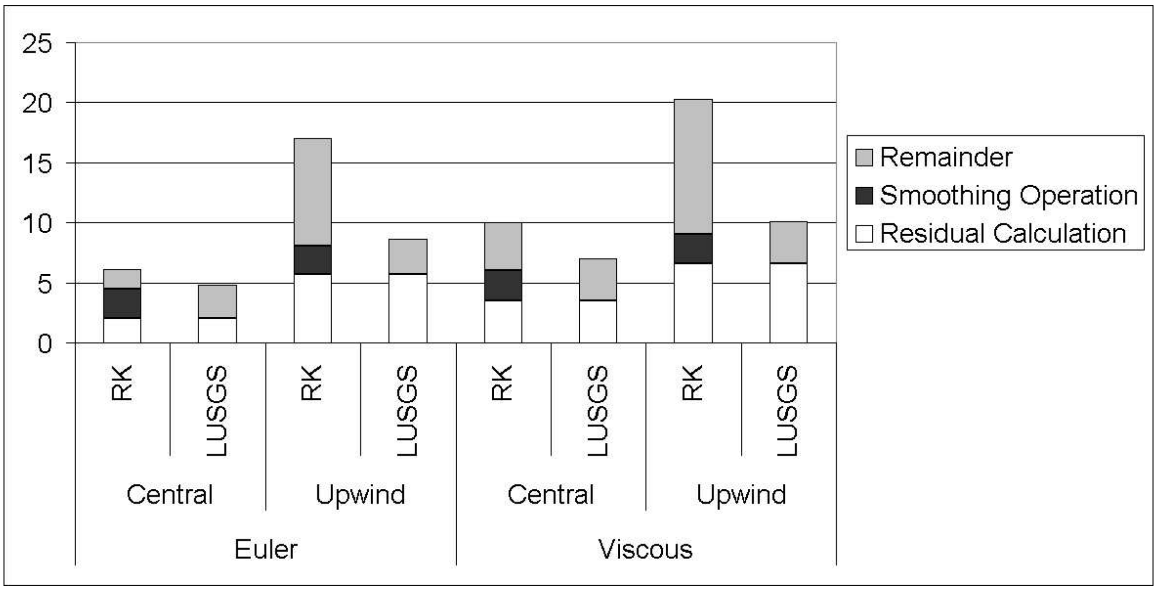


Figure 3.19: Normalized time required for a single iteration of Runge-Kutta and LU-SGS for various spatial discretizations. The time for one iteration is divided into three components: that due to the initial calculation of the spatial residual, that due to any residual smoothing required, and that due to remaining operations. For Runge-Kutta these remaining operations contain additional residual computations necessary for subsequent stages.

of the CPU time of Runge-Kutta for convergence to a residual of 1×10^{-4} for this case.

3.4.7 Wing-Body-Tail Configuration

One key requirement for the implicit algorithm is that it must be applicable to large practical cases that appear in the aerodynamic development of aircraft.

The final and demanding test-case presented is a generic military transport aircraft, with modelling of wing, body and tail, with wheel-house fairings and strake visible in Figure 3.20. The grid has $\approx$ 22 million points, most of which are used in the resolution of the boundary layer and the vortices expected to be produced by the fairings and the strake. The farfield is 30 chord lengths distant and provides an onflow at an angle of attack of $4°$ and Mach number of 0.68. The Reynolds number is 3×10^6.

The discretization consists of JST inviscid fluxes, full viscous fluxes, and the Spalart-Allmaras one-equation turbulence model. The calculation is performed on 16 nodes of an HP IA64 cluster, for Runge-Kutta 3-stage with residual smoothing and LU-SGS, both with a 3V multigrid cycle and local time stepping. The CFL number used with RK was 1.5 and for LU-SGS 3.0, which approaches the empirically tested stability limit of the respective schemes for this case. The convergence histories are shown in Figure 3.21, together with the transient behaviour of the drag. The use of LU-SGS results in an approximate doubling of the convergence rate, at least in the initial stages of convergence. The CPU time required for one LU-SGS iteration is approximately 90% of that of one Runge-Kutta iteration.

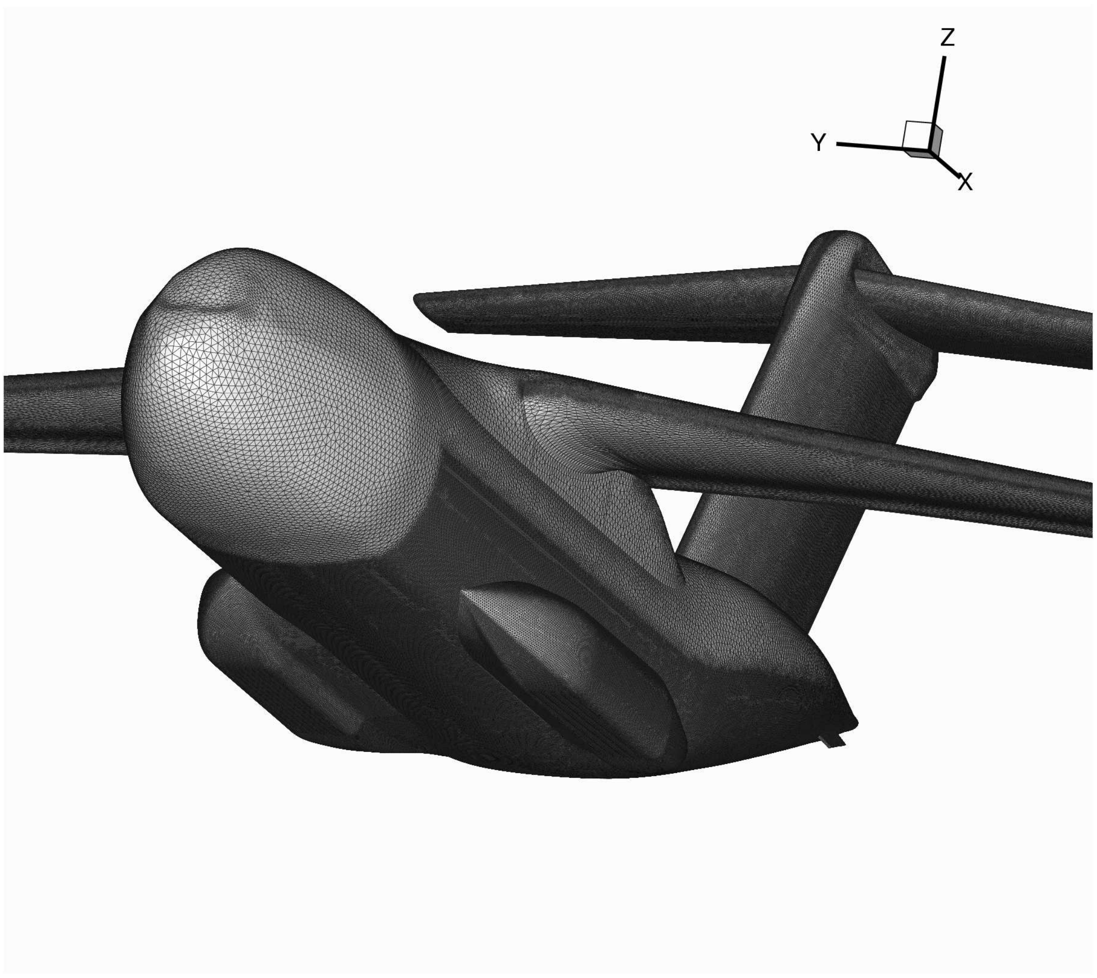


Figure 3.20: Surface mesh of the generic military transport configuration showing tail, fairings and strakes. The grid is composed of prisms in the boundary layer, and tetrahedra elsewhere. (Thanks to Airbus for permission.)

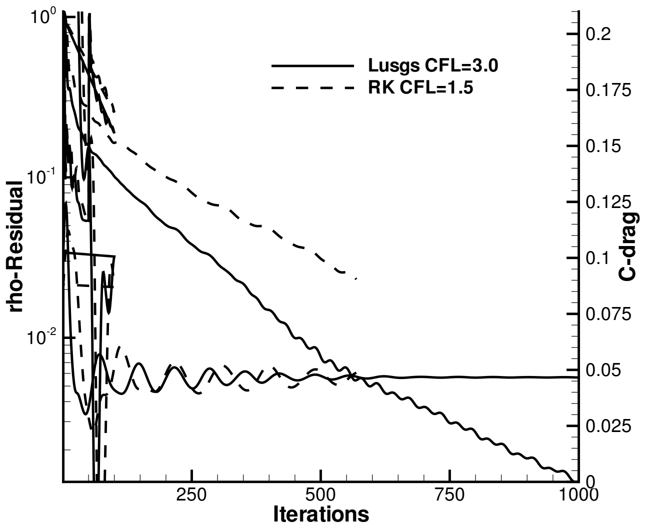


Figure 3.21: Convergence behaviour of Runge-Kutta and LU-SGS on the generic military transport configuration. Shown are the non-linear ρ-residual and the transient drag c_d. CPU time for one LU-SGS iteration is approximately 90% that of one Runge-Kutta iteration.

3.5 Summary

The LU-SGS scheme with multigrid has been investigated as it applies to the solution of the RANS equations discretized using the finite volume method on unstructured grids. In particular a variety of approximations to the Jacobian of the method, and a variety of linear solvers, have been proposed, and the optimal combination thereof chosen. This has been accomplished using results obtained from numerical experiments supported by a theory of the stability of block Gauss-Seidel iterations, and Fourier analysis.

The performance of the resulting method shows a typical improvement on 3-stage Runge-Kutta, in terms of rate of convergence of 10%-50%, and in terms of CPU time per iteration of 10%-40%. Even so the memory requirements are comparable to those of Runge-Kutta, and the scheme converges for all those cases for which Runge-Kutta converges. The method therefore represents a robust, fast, extremely memory efficient, easily implementable and parallelizable implicit method, that works for a variety of convective flux discretizations, but in particular the central scheme with scalar dissipation. The method as described has been demonstrated for two large test cases of practical engineering interest.

With this chapter a range of approximate and inexact Newton methods have been examined based on very simple linear solvers. If the prerequisites for the algorithm are modified slightly, so that much larger memory requirements are allowed, implicit methods based on Krylov subspace linear solvers using more accurate Jacobians become a possibility. These methods are the subject of the next chapter.

Chapter 4

Unfactored Implicit Schemes

4.1 Introduction

The philosophy of the development of the implicit scheme of Chapter 3 was that such a scheme could be an effective alternative to an explicit Runge-Kutta method for large applications in aerospace engineering, if it had similar memory requirements, no start-up difficulties, and could be easily parallelized. This was achieved with the LU-SGS scheme, whereby these restrictions (in particular the first) severely limited the choice of Jacobian, and hence the convergence rate possible.

In this chapter these restrictions are relaxed, and schemes are considered that may have significantly larger memory requirements than explicit methods. This may be justified by noting that for the cluster computers which increasingly dominate the scientific computing landscape, and which typically have large amounts of memory per node, the limiting factor in common engineering applications is cluster CPU time. For a given case and an efficient parallel solver, the total CPU time needed is roughly a constant function of the number of processors used; however the wall-clock time falls in inverse proportion to the number of processors. Engineers require timely results and so use many processors, whereby the full memory capacity of the individual nodes often remains unused. An implicit method, even with significantly larger memory requirements, would be practical under such conditions, provided that it is efficient in parallel. This is not to say that memory considerations may be ignored altogether however, and efforts will be made to produce a memory-efficient method.

Another disadvantage of LU-SGS was that because the implicit operator developed was not accurate enough to provide effective full-domain communication within one iteration on its own, the method was only efficient when used as a multigrid smoother (this is not the case for an exact Newton method, for example). While multigrid on structured grids is extremely effective (Pierce *et al.*, 1997), on unstructured grids it often suffers under poor coarse-grid quality, due to the difficulty of producing a cell agglomeration algorithm that results in smooth convex volumes in 3D. Some selected coarse grid cells from the second multigrid level of a 3d calculation using the *TAU*-code are shown in Fig. 4.1; note that even on the surface the volumes are irregular, while in the field there is even a cell shaped like a banana. The influence of the multigrid correction in this region of the mesh was shown to cause divergence of the entire calculation in this case. One aim of the development of implicit methods in this chapter is therefore the removal of non-linear multigrid from the solution process,

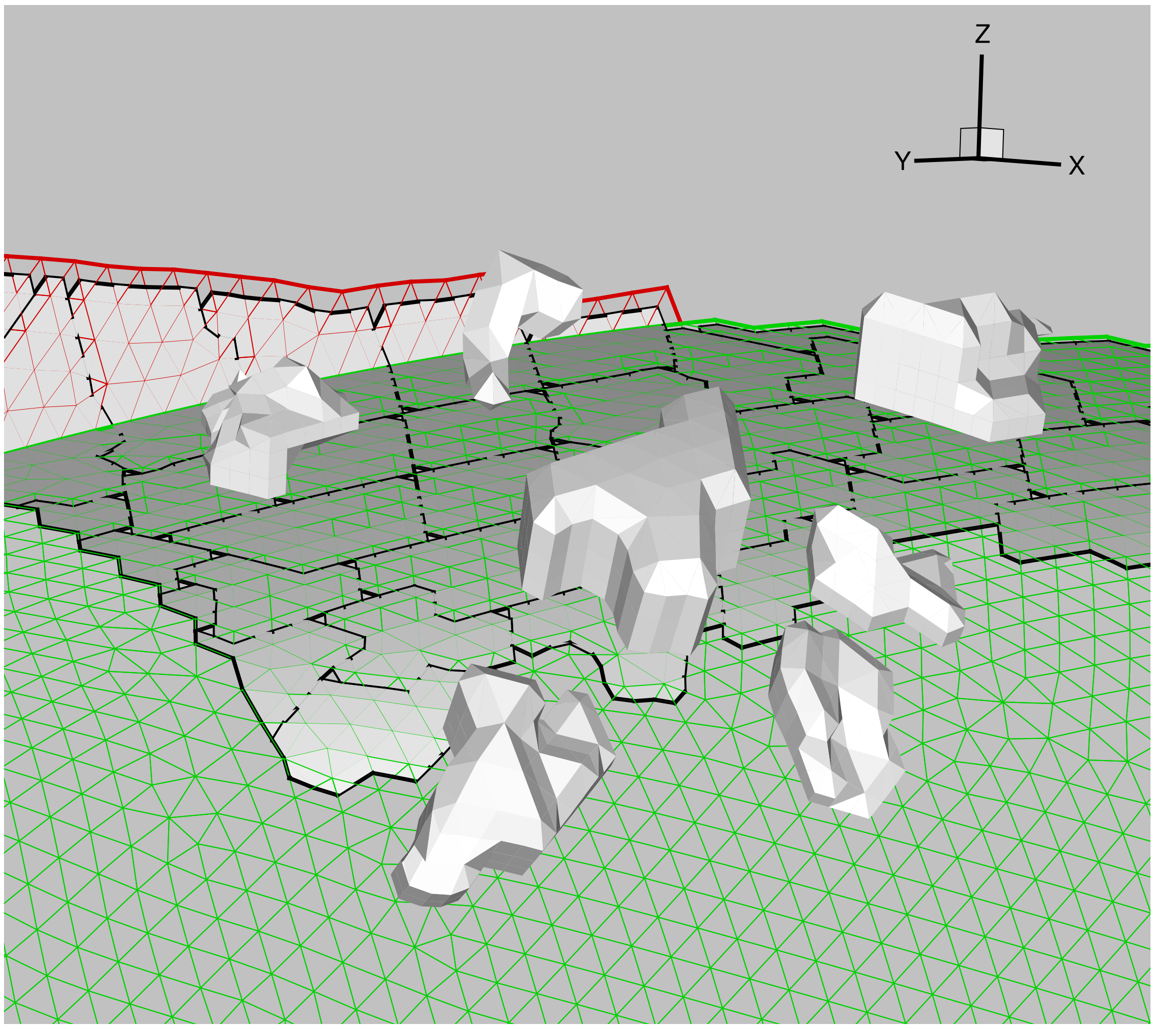


Figure 4.1: Selected coarse dual grid control-volumes produced by a agglomeration algorithm near a corner of a viscous wall. The non-convex shapes and large variation in cell size are typical of coarse grids generated in this manner.

and consequently removal of the dependence on unreliable coarse-grid agglomeration algorithms.

To this end more accurate Jacobians are hand-formulated and explicitly stored, forming the basis for an exact Newton method. From this starting point, simplifications to the Jacobian are made in order to reduce the stiffness of the resulting linear system, and thereby allow their rapid solution. The result of this process is a novel variant of a method using a Jacobian based on first-order convective fluxes, that includes the exact Jacobians of all boundary conditions. The turbulence model equations are decoupled from the mean flow equations, and treated with exact Jacobians with respect to the turbulence variables. The linear inner-iteration is solved with preconditioned Krylov methods.

As a result of the increased complexity of these new algorithms, large configurations such as those shown in Chapter 3 will not be shown, rather the emphasis is on finding methods that perform extremely well for simple cases such as 2d aerofoils, and that have the potential to be extended to more complex configurations.

4.1.1 Overview

This chapter begins with a general description of the methods used for the per-hand implementation and verification of the Jacobian of the finite volume discretization of *TAU*, see Section 4.2. The Jacobian is exact in the sense that each and every component of $R(W)$ is differentiated exactly, including boundary conditions and turbulence models, with *no* simplifications or approximations made. The techniques described are applicable to the differentiation of flow solvers in general.

Corresponding to the stiffness of the non-linear problem the Jacobian is very poorly conditioned, especially for Navier-Stokes problems. Section 4.3 examines the solution of linear systems involving the exact Jacobian, and considers preconditioned Krylov algorithms. In particular Incomplete Lower-Upper (ILU) preconditioned Generalized Minimal Residual Method (GMRES) is examined, and it is seen that for timely convergence very large ILU fill-in levels are required, having correspondingly large memory requirements. It is also seen, as previously reported (Campobasso & Giles, 2002), that for linear systems resulting from Newton iterations based on partially converged non-linear solutions, a necessary condition for convergence of Jacobi or Gauss-Seidel iterations on the linear system is not satisfied. In this situation a Krylov solver is essential, and Jacobi and Gauss-Seidel are poor preconditioners.

Using ILU preconditioned GMRES and an explicitly stored Jacobian, an exact Newton method is constructed and tested on a turbulent RAE2822 test case. Quadratic convergence is observed, and the method is much faster than LU-SGS, but requires ILU(4) preconditioning and may only be started after five orders of magnitude reduction in the residual have been achieved.

Given these significant difficulties with exact Newton methods, attention is again turned to approximate Newton methods. Section 4.5 considers a method based on an explicitly stored, first-order Jacobian. As the reduction to first-order is more natural for upwind schemes, the investigation begins with an implicit method for Roe's scheme. This is briefly compared with another recently proposed and similar method, also for Roe (Rossow, 2005). However the goal is to improve performance using the JST scheme as the spatial discretization. Hence a first-order JST Jacobian is proposed and solved with ILU preconditioned GMRES. The method is shown to be 4-5 times faster than the LU-SGS method for the cases considered, has no reliance on geometric multigrid, and may be easily parallelized.

Throughout this chapter the RAE2822 Test-Case Definitions 9 and 10 will be used, which have come to be standard reference cases in CFD, in particular in the area of turbulence modelling. They are defined in (Cook *et al.*, 1979) along with detailed experimental results. Both cases are transonic flows about the non-symmetric RAE2822 single element aerofoil, with chord Reynolds numbers of 6.5×10^6 and 6.2×10^6, at Mach numbers of 0.73 and 0.75 respectively. The angle-of-attack is $2.8°$. Both cases are transonic with normal shocks on the aerofoil upper surface. Case 9 is fully attached, while Case 10 has a small area of separated flow near the trailing edge. The grid used is shown in Figure 3.13 and has about 14×10^3 points.

4.2 Construction of the Exact Jacobian

Implementation of an exact Newton method requires the ability to formulate and solve linear systems of the form

$$\frac{\partial R}{\partial W}\Delta W = -R(W),$$

with respect to ΔW. The residual R is taken to be readily available here, whereas the Jacobian $\partial R/\partial W$ may be difficult to obtain.

One option is to use finite differences to approximate the Jacobian-vector product, and use solution algorithms that require only this product, and not the Jacobian explicitly. This was examined briefly in Section 3.4.1, and was found not to be competitive with LU-SGS because of the expense of repeated evaluations of R. It may be the case that an alternative Krylov solver/preconditioner combination would require less Jacobian-vector products, but a more promising alternative is to use an explicitly stored Jacobian.

The Jacobian may be evaluated by hand, which is a straightforward exercise as R may be written explicitly in terms of W, while being very time-consuming as R is typically extremely complex, as illustrated in Section 2.6.3. However, as R is a sum of convective fluxes, viscous fluxes, boundary conditions etc., each of these may be differentiated independently, and may be further subdivided into manageable chunks by application of the chain rule. The operation of differentiation is further simplified by choosing primitive variables as working variables. Because the equations themselves remain in conservative form, this choice has no effect on the final solution, however the update ΔW will then be in terms of primitive variables. Strong boundary conditions such as the specification of zero velocity on viscous walls are handled as discussed in Section 2.9.4.

The accuracy of the implementation of the derivatives of the individual fluxes is verified by applying finite differences to the original routines, and comparing with the hand-calculated Jacobian for a variety of inputs. As each flux derivative is calculated, a contribution is made to the explicitly stored Jacobian.

To provide an insight into the implementation, the somewhat personal process developed by the author after much trial and error is offered here, in the hope that it may guide initial efforts of others. It may be summarized in the following six steps:

1. Divide the code for the non-linear residual R into parts that may be differentiated independently - either because R is a simple sum of e.g. inviscid and viscous fluxes, or by means of the product and chain rules. All parts of the code that do not influence the solution of $R(W) = 0$, are superfluous.

2. Copy out the definition for e.g. a particular flux function $\hat{f}$, directly from the code onto paper. Begin with the definition of $\hat{f}$ in terms of intermediate variables at the end of the code fragment, and proceed upwards. This order will aid the application of the chain rule in Step 4. In order that the resulting Jacobian be the exact derivative of R it is the author's experience that it is not sufficient to take flux function definitions from technical reports, published articles etc., as the reality is often substantially different.

	Std. *TAU*	+ Jac. storage	+ Linear sol. storage
Memory (Bytes)	25M	165M	290M
Factor increase	$\times 1.0$	$\times 6.6$	$\times 11.6$
Points in 1GB	2×10^6	300×10^3	170×10^3

Table 4.1: The memory requirements of the linearized code with explicit exact Jacobian storage, compared to the standard *TAU*-code - measured for a two-dimensional unstructured grid with 50×10^3 points. Also given are the maximum number of points that would fit in 1GB of memory. The linear solver used is ILU(4) preconditioned GMRES(30).

3. Decide which parts of the definition of $\hat{f}$ to neglect as insignificant, too time-consuming to differentiate, or as a means of providing a more efficient implementation. This step is optional.
4. Differentiate the simplified version of $\hat{f}$ on paper, which should now be a matter of working top to bottom on the page. Making use of the chain rule wherever possible (i.e. for every intermediate variable) tends to simplify expressions and helps avoid errors. Given a one page definition, a two page derivative can typically be expected.
5. Implement the derivative calculated in Step 4. As a basis use the original function for $\hat{f}$, as many intermediate variables will appear in undifferentiated form in the expression for the derivative; also the inputs should be identical. Write the original expressions and their derivatives together. Comment the derivative with the original function's name, with respect to which variables it is differentiated, and any assumptions made during Step 3.
6. Numerically compare the hand-coded derivative against a finite difference approximation using the original function. It is very likely that at least one mistake was made, very often in the transcription of the original routine from code onto paper, or in the implementation of the derivative.

Storing the full Jacobian explicitly has the disadvantage of requiring approximately six times the memory of the standard code, see Table 4.1, reducing the capacity of a node with 1GB of memory from 2 million points to 300 thousand points. Whereby it is important to emphasize that this result is only valid in 2d, the situation in 3d being worse due to the greatly increased numbers of next-neighbours of a point. Given the linear systems stored explicitly the remaining challenge is their solution.

4.3 Solution of Linear Systems Involving the Exact Jacobian

Although there exist many generally applicable algorithms for the numerical solution of sparse linear systems, see e.g. (Saad, 2003), it is worth considering whether the systems under consideration have any special properties which admit alternative solution methods. This is of particular interest in the case of the linearized Navier-Stokes equations, as the linear systems can be extremely stiff and difficult to solve with the usual preconditioned Krylov methods.

Since the linear equations are derived directly from non-linear equations, one possibility is to apply the non-linear solution method to the linear system. This approach is discussed in Section 4.3.1, and it is seen that it is not guaranteed to be stable, even if the iteration for the non-linear problem converges. The reasons for this are examined in Section 4.3.2, where the preconditioned GMRES method is introduced.

4.3.1 Application of Existing Non-Linear Iteration

One immediate possibility is to use the same iterative method for the linear system as for the non-linear system; in this case the LU-SGS smoothed FAS multigrid method with local time stepping. The convergence rate of the linear system will then be identical to the asymptotic convergence rate of the non-linear system. This can be seen as follows: let $\tilde{W}$ represent the exact solution, and P a completely arbitrary preconditioning operator (including e.g. multigrid), then without loss of generality consider an explicit scheme for the non-linear problem

$$\begin{aligned} P\left(W^{n+1} - W^n\right) &= -R(W^n) \\ &= -R\left(\tilde{W} + (W^n - \tilde{W})\right) \\ &= -R(\tilde{W}) - \left.\frac{\partial R}{\partial W}\right|_{\tilde{W}} (W^n - \tilde{W}) + \mathcal{O}\|W^n - \tilde{W}\|^2, \end{aligned}$$

using $R(\tilde{W}) = 0$ and rearranging gives

$$W^{n+1} = \left(I - P^{-1} \left.\frac{\partial R}{\partial W}\right|_{\tilde{W}}\right)(W^n - \tilde{W}) + \tilde{W}. \tag{4.1}$$

Applying the same scheme to the linear problem arising at a Newton iteration we have

$$P\left(x^{n+1} - x^n\right) = -\left(\frac{\partial R}{\partial W} x^n - b\right),$$

which becomes

$$x^{n+1} = \left(I - P^{-1}\frac{\partial R}{\partial W}\right) x^n + P^{-1} b, \tag{4.2}$$

after rearranging. The coefficients of the solution at time level n in (4.1) and (4.2) are identical provided the Jacobian in the linear problem is based on a sufficiently converged non-linear solution. Hence the convergence rates are identical, and the extensive experience gained in developing iterative schemes for the non-linear problem can be applied to the linear problem.

A numerical demonstration of this property is given in the left-hand plot of Figure 4.2 for the RAE2822 Case 9 with the SAE turbulence model. The convergence history of a linear problem, corresponding to an exact Newton iteration based on the solution of the non-linear problem found after 4000 multigrid cycles, is compared

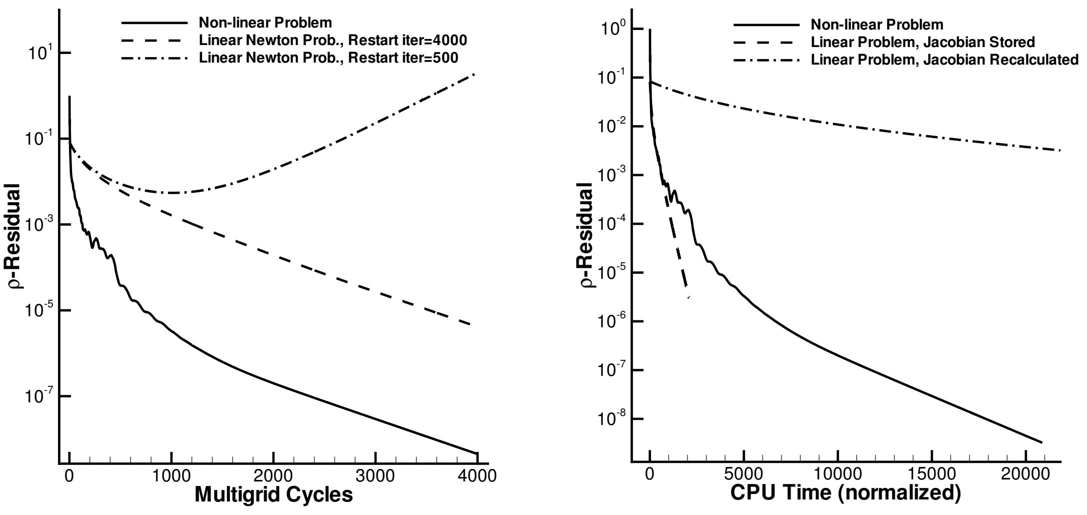


Figure 4.2: Convergence of one non-linear and two different linear problems for a transonic RAE2822 with a one-equation turbulence model. The solution method is identical for all problems, being LU-SGS smoothed FAS multigrid. The two linear problems in the left-hand plot are those obtained from forming a Newton type iteration after 4000 and 500 iterations of the non-linear system respectively. In the right-hand plot, two possible implementations of the linear solver are compared: a low-memory version, with evaluation of the Jacobian at each iteration, and a high-memory version, with one-time calculation of the Jacobian which is explicitly stored thereafter. The divergent linear problem is not shown in the right-hand plot.

with the non-linear convergence history itself. Exactly the same LU-SGS smoothed multigrid algorithm is applied to both problems. It can be seen that the non-linear convergence achieves a constant asymptotic slope after about 2000 iterations, and the linear convergence reproduces this slope exactly.

Whether an iterative scheme that is efficient for the non-linear equations will also be efficient for the linear equations therefore depends most significantly on the relative cost of the linear and non-linear residual evaluations. In our case if the Jacobian is stored explicitly a linear residual corresponds to a matrix-vector product, with a CPU time cost of approximately 10% of that of the non-linear residual, given the one-time cost of forming the matrix. If the matrix is not stored the evaluation of the linear residual is 10 to 15 times more expensive than that of the non-linear residual. Example convergence in terms of CPU time for these two cases is given in the right-hand plot of Figure 4.2.

Another factor to consider is that the asymptotic convergence of the non-linear problem tends to be much slower than the initial convergence. This can be seen in the non-linear convergence in Figure 4.2, with the phenomenon of *multigrid breakdown* occurring at between 500 and 1000 cycles, where the dominant error modes become those that are poorly smoothed by multigrid. Hence for any given number of iterations the linear residual is reduced less than the non-linear. This effect is offset by the fact that the linear residual must only be reduced by about three orders of magnitude for many applications. This is the case for the Newton convergence shown in Section 4.4, as well as for the adjoint problems in the following chapter. One additional disadvantage associated with this approach is that if the convergence rate of the adjoint iteration is to be guaranteed to be the same as the non-linear the solution procedure must be additionally adjointed. For LU-SGS or Runge-Kutta with multigrid this is however a relatively straightforward process (Giles *et al.*, 2003).

A more serious problem with this approach is that even if the non-linear iteration converges, there is no guarantee that the linear problem resulting from a Newton iteration based on a partially converged non-linear solution will converge. This is demonstrated on the left-hand side of Figure 4.2, which shows the convergence curve of the linear system resulting from a Newton iteration after 500 non-linear iterations. The iteration diverges after a small number of steps, and reducing the CFL number in the iteration merely postpones the divergence.

This phenomenon has been observed for a variety of turbulent Navier-Stokes cases but not for the Euler equations. In particular in cases where the non-linear problem converges poorly, or where convergence stalls in a limit cycle, the effect has been observed. An attempt at an explanation is given in the following section.

4.3.2 Application of preconditioned GMRES

The deficits inherent in the method of the previous section suggest considering Krylov methods, some of which are guaranteed to converge for arbitrary linear systems. Previous work shows that the choice of particular Krylov method has a significantly smaller impact on the performance of the linear solver than the choice of preconditioner (Meister, 1998). As such we concentrate on the use of the popular Generalized Minimum Residual (GMRES) algorithm (Saad & Schultz, 1988).

Preconditioned GMRES rests on forming an orthonormal basis of the Krylov space $\mathcal{K}_m$ given by

$$\mathcal{K}_m(P^{-1}A, r_0) = \text{span}\left\{r_0,\, (P^{-1}A)r_0, \cdots,\, (P^{-1}A)^{m-1}r_0\right\},$$

where span$\{\cdots\}$ denotes the space spanned by the vector arguments, and $r_0 = b - P^{-1}A \cdot x_0$ is the initial linear residual. The Arnoldi procedure is applied iteratively to $P^{-1}A$ with initial Arnoldi vector $v_0 = r_0/\|r_0\|$. A standard Gram-Schmitt algorithm, internal to the Arnoldi process, produces an orthonormal basis $V_m = (v_0, \cdots, v_{m-1})$ of $\mathcal{K}_m$, together with an $(m+1) \times m$ upper Hessenberg matrix $\bar{H}_m$ such that

$$P^{-1}A \cdot V_m = V_{m+1} \cdot \bar{H}_m, \tag{4.3}$$

i.e. a progressive reduced factorization. On the m-th iteration, GMRES approximates the solution of $A \cdot x = b$ by a linear combination of the m available v_i, chosen to minimize the 2-norm of the linear residual. This is computationally cheap as it can be reduced to a least-squares problem involving the $(m+1) \times m$ Hessenberg matrix.

A bonus associated with the GMRES method (Campobasso & Giles, 2004) is that it is possible to obtain estimates for the eigenvalues of $P^{-1}A$ as follows. From (4.3) we have

$$\begin{aligned} P^{-1}A \cdot V_m &= V_m \cdot H_m, \\ V_m^T \cdot P^{-1}A \cdot V_m &= H_m, \end{aligned} \tag{4.4}$$

where H_m is the $m \times m$ matrix formed by deleting the last row of $\bar{H}_m$. The eigenvalues of H_m are

$$\begin{aligned} H_m z_i &= \lambda_i z_i, \\ (V_m^T \cdot P^{-1}A \cdot V_m) z_i &= (V_m^T \cdot V_m) \lambda_i z_i, \\ V_m^T \cdot (P^{-1}A - \lambda_i I) \cdot V_m z_i &= 0, \end{aligned} \tag{4.5}$$

where the orthonormality of V_m has been used in (4.5). So the eigenvalues of H_m are approximations to the eigenvalues of $P^{-1}A$, with the error quantified by the *eigensystem residual*:

$$r_{\text{eig}} = (P^{-1}A - \lambda_i I) \cdot V_m z_i,$$

which is orthogonal to the Krylov subspace $\mathcal{K}_m$. It may be shown that r_{eig} depends linearly on the residual of the linear equations. A condition for accurate eigenvalues is therefore the convergence of the GMRES iteration. By applying LU-SGS with multigrid as a preconditioner, it is then possible using this algorithm to approximate the eigenspectrum of the operator.

For a complete exposition refer to (Saad, 2003) for example. In practice rather than implementing the routines privately, the author uses the excellent publicly available Portable, Extensible Toolkit for Scientific Computation (PETSc) library (Balay *et al.*, 1997; Balay *et al.*, 2004; Balay *et al.*, 2006), which implements a variety of Krylov solvers and preconditioners in a parallel environment using the Message Passing Interface (MPI). The availability of this library saved considerable development time during these investigations.

First we use eigenspectrum analysis to investigate the reasons for the failure of LU-SGS smoothed multigrid to solve the linear problem of Figure 4.2. From (4.2) it is clear that the method will converge if and only if all eigenvalues of the operator

$$\left(I - P^{-1} \frac{\partial R}{\partial W} \right) = \left(I - P^{-1}A \right),$$

have magnitude strictly less than 1. Using the Arnoldi procedure, approximations to the eigenvalues of $P^{-1}A$ are evaluated for the two linear systems of Section 4.3.1, which resulted from Newton iterations on partially converged non-linear solutions after 4000 and 500 multigrid cycles. The calculated eigenspectra are shown in Figures 4.3 and 4.4 respectively.

As can be seen, for the Newton iteration started at 4000 cycles, all eigenvalues lie within the unit circle, although some are very close to the boundary, indicating modes that are poorly damped by the scheme. On the other hand for the Newton iteration started at 500 cycles, one complex conjugate pair of eigenvalues have absolute value greater than one, and are amplified by the method. The modes associated with these eigenvalues are the cause of the instability observed in Figure 4.2.

The analysis does not suggest a remedy however, so alternative solution methods are considered. The explicit storage of the system matrix allows the use of the Incomplete Lower-Upper (ILU) preconditioner, which performs Gaussian elimination on A, but drops elements which do not occur in predetermined positions $\mathcal{P}$, thus reducing memory requirements and computational effort. The most common *fill-in* pattern $\mathcal{P}$, is the sparsity pattern of A itself. This method is termed ILU(0). More accurate preconditioners are denoted ILU(n), which roughly speaking allow fill-in

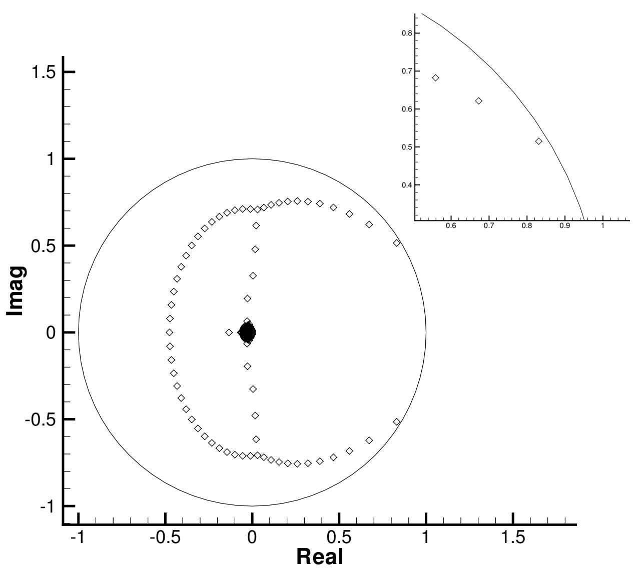


Figure 4.3: Approximate eigenspectrum of the LU-SGS multigrid preconditioned Jacobian of the RAE2822 Case 9 after 4000 non-linear iterations. This corresponds to the convergent linear iteration of Figure 4.2. Spectrum obtained using the Arnoldi procedure with 500 iterations.

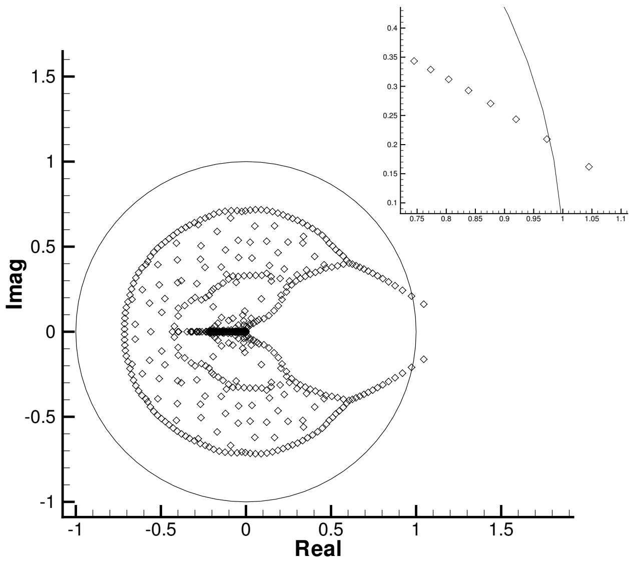


Figure 4.4: Approximate eigenspectrum of the LU-SGS multigrid preconditioned Jacobian of the RAE2822 Case 9 after 500 non-linear iterations. This corresponds to the divergent linear iteration of Figure 4.2. Spectrum obtained using the Arnoldi procedure with 500 iterations.

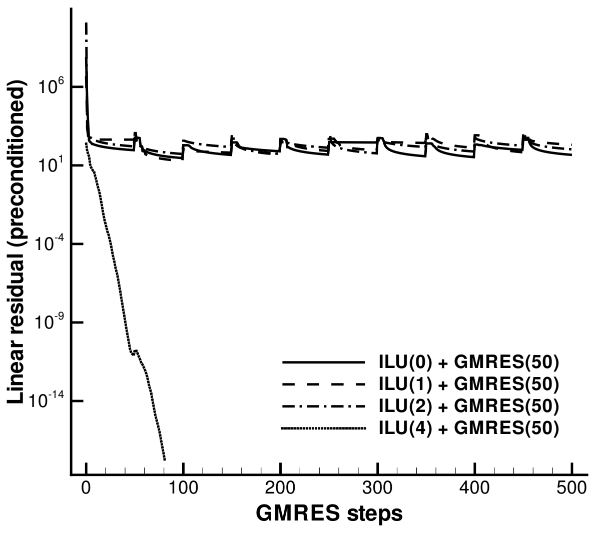


Figure 4.5: Convergence of the (preconditioned) linear residual for a Newton inner iteration starting from a partially converged non-linear state, for the ILU(n) preconditioner with a variety of levels of fill-in.

of elements that are reached within n steps of the Gaussian elimination, for details see (Saad, 2003).

For Euler problems ILU(0) preconditioned GMRES has been seen to be sufficient. However for the viscous RAE2822 case of the previous section ILU(0) often results in convergence that is too slow. Convergence of several ILU(n) preconditioned GMRES methods applied to the Newton iteration restarted after 500 non-linear multigrid cycles, are shown in Figure 4.5. In order to reduce the amount of storage needed for the Krylov vector, *restarted* GMRES (GMRES(m)) was used, which resets the Krylov space every m iterations. Unlike pure GMRES, restarted GMRES is not guaranteed to converge. Here a restart length of 50 is specified, whereas 10 to 30 is typical in practical applications.

The method completely fails to converge for anything less than ILU(4), which is undesirable due to the significantly greater amount of storage needed, detailed in the last column of Table 4.1. The complete linearized code requires more than ten times the memory of the standard code.

The convergence problems may be traced back to the extreme stiffness of the linear systems, which may be characterized by their condition numbers. In particular in the case just examined the condition numbers for the various preconditioned operators are given in Table 4.2, compared to an unpreconditioned condition number calculated for a simple Euler case of about 1×10^5. High condition numbers may be interpreted as evidence of the transfer of the stiffness inherent in the non-linear problem to the linear problem.

Another deficit of the ILU preconditioner is that the convergence rate of the method (used alone) decreases as the problem size increases (corresponding to grid-dependent convergence), as is also the case for Jacobi and Gauss-Seidel iterations, indicating that its effectiveness as a preconditioner will also decrease. This effect has

Preconditioner	Condition Number
None	1.2×10^{15}
ILU(0)	7.7×10^{8}
ILU(1)	9.3×10^{7}
ILU(2)	6.2×10^{6}
ILU(4)	4.1×10^{6}

Table 4.2: Condition numbers of preconditioned Jacobians for the RAE2822 case from a Newton iteration started after 500 multigrid cycles.

not been observed in practice in this study, but future work will investigate the use of preconditioners with grid-independent convergence, i.e. multigrid.

4.4 Example of an Exact Newton Method

An exact Newton method was implemented and tested in Section 3.4.1, using a finite difference approximation of the Jacobian, and was found to be less efficient than the LU-SGS scheme in terms of CPU time, partly due to the expensive residual evaluation necessary to approximate each Jacobian-vector product. Here the Newton method is implemented using the explicitly formulated and stored Jacobian and, based on the results of the previous section, an ILU(4) preconditioned GMRES solver is used for the solution of the linear system at each time step.

In addition to evaluating the efficiency of the method, this test also serves to validate the accuracy of the Jacobian as implemented, as if it in any way differs from the true Jacobian, the Newton method will not give quadratic convergence in the limit as the error tends to zero.

The test case is the same RAE2822 Case 9 as in the previous section. The Newton iteration needs a partially converged solution to begin, as previously discussed, and this is provided by LU-SGS smoothed multigrid cycles; the convergence is shown in Figure 4.6. Through trial and error it was discovered that it was not possible to start the Newton iteration before about 1200 multigrid cycles, or before a residual of about 1×10^{-6} had been reached. Once started however the iteration converges to machine accuracy in less than 15 steps, and is 6-8 times faster than the multigrid method in terms of CPU time. Also while the convergence of the Newton method cannot be claimed to be quadratic, it is clearly super-linear.

This case must be considered to be one of the simplest transonic high-Reynolds number cases possible, a simple two-dimensional geometry with no regions of separation. Even so the difficulties encountered in applying a Newton method are considerable; from the necessity of storage of the Jacobian, to the problems associated with solving the linear system, to the start-up issues. In particular in Figure 4.6 it can be seen that the drag is fully converged before the Newton method can even be started. From an engineering perspective the method as it stands is therefore useless, despite being much faster in the zero-error limit.

While the start-up problem may be mitigated with a variety of *continuation* techniques (Knoll & Keyes, 2004), the difficulties involved already make the prospects

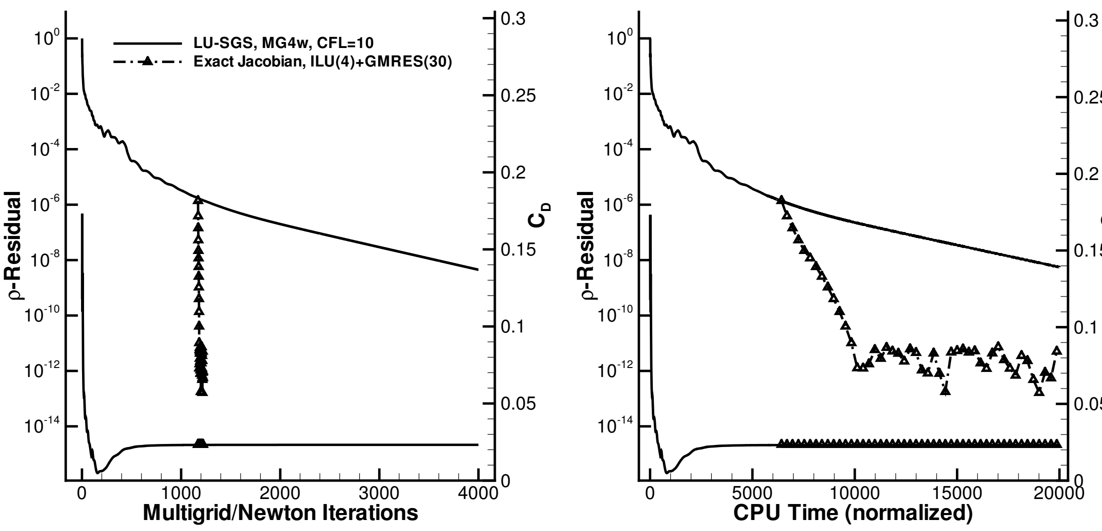


Figure 4.6: Convergence of the RAE2822 Case 9 with the SAE turbulence model, using an exact Newton method restarted from an LU-SGS multigrid iteration.

of the method for large three-dimensional cases poor. Therefore attention is again turned in the remainder of this chapter to approximations of the Jacobian, which have the potential to reduce memory requirements and linear solution time.

Note that this conclusion does not necessarily apply to unsteady simulations, where a good starting solution may be available (from the previous time step), and where the linear system tends to be better conditioned due to the presence of the discretized time derivative in the Jacobian, see e.g. (3.38).

4.5 Approximate Newton Methods

By approximating the Jacobian, it is relatively easy to achieve schemes with much lower memory requirements, and well-conditioned inner linear systems; an extreme example being the LU-SGS scheme of Chapter 3. Here we develop another scheme in this manner, whereby we explicitly allow the scheme to have significantly greater storage requirements than Runge-Kutta, and we attempt to achieve high convergence rates without the use of multigrid.

The resulting scheme, which bears a resemblance to work in implicit methods on structured grids undertaken in the Department of Aerospace Engineering at Glasgow University (Woodgate *et al.*, 1997; Cantariti *et al.*, 1997; Badcock *et al.*, 1999; Cantariti *et al.*, 1999), but differs in that it is applied on unstructured grids, thereby introducing Jacobians with unstructured sparsity patterns. Also while previous work has considered only upwind schemes, here we investigate the extension to the JST convective flux discretization.

For the purposes of comparison another recently developed scheme is described in Section 4.5.2, and both schemes are tested numerically, in comparison with LU-SGS.

4.5.1 1st-Order Jacobian Krylov Implicit Method (FOKI)

The method developed here will be referred to as the First-Order Jacobian, Krylov Implicit (FOKI) method, as it uses an explicitly calculated and stored Jacobian based on first-order convective fluxes, the resulting system being solved using a preconditioned Krylov method (in the following GMRES). The turbulence equations are decoupled from the mean flow equations, and solved using a separate Krylov iteration. It is always used without multigrid.

Approximation of the Jacobian

The chief aim of this approximation is to reduce the stencil of the discretization to immediate neighbours of a node only, leading to a corresponding reduction in the sparse fill-in of the Jacobian. Such a simplification results in a reduction of the memory requirements of the solver with Jacobian storage to less than three-times those of the non-linear solver, and the system can be solved in a time equivalent to less than 1% of that required for the non-linear system. Further the immediate neighbour fill-in allows the storage of the off-diagonal entries of the Jacobian on the edges of the unstructured grid (as opposed to being stored in a special sparse matrix data structure), making a Jacobian-vector multiplication a familiar loop over all grid edges.

This stencil reduction may be achieved by constructing the Jacobian from derivatives of first-order convective fluxes, and TSL viscous and turbulence diffusion fluxes (see Section 2.8.1). The stencil of the turbulence discretization consists only of immediate neighbours anyway, and therefore is treated without approximation.

The most sensitive issue is the simplification of the convective fluxes. For an upwind discretization the first-order generalization is natural; the use of constant face-reconstruction rather than affine reconstruction reduces the stencil in the required manner. In order to improve the Jacobian approximation somewhat, once the expression for the flux derivative has been obtained from the first-order upwind flux, values reconstructed onto the cell faces are used instead of cell-centered values in the expression.

For the JST scheme used here the situation is not as clear. For convenience the JST flux is repeated here, from (2.28):

$$\begin{aligned}\hat{f}_{ij}^{\mathrm{JST}}(W; n_{ij}) &= \frac{1}{2}\Big(f^c(W_i) + f^c(W_j)\Big) \cdot n_{ij} \\ &- \frac{1}{2}\left|\lambda_{ij}^c\right| \Big\{\bar{\varepsilon}_{ij}^{(2)}(W_j - W_i) - \bar{\varepsilon}_{ij}^{(4)}(L_j(W) - L_i(W))\Big\}, \qquad (4.6)\end{aligned}$$

where in addition

$$L_i(W) = \sum_{k \in \mathcal{N}(i)} (W_k - W_i), \qquad (4.7)$$

in the interior of the field.

	x_{i-2}	x_{i-1}	x_i	x_{i+1}	x_{i+2}
2nd difference		$+1$	-2	$+1$	
4th difference	$+1$	-4	$+6$	-4	$+1$
4th chopped		-4	$+6$	-4	

Table 4.3: Second and fourth difference operators in one-dimension. Also a fourth difference that has been "chopped" in order to reduce its stencil.

There are two principal possibilities to reduce the stencil: (a) use purely second-order dissipation, neglecting the term involving $L(W)$ in (4.6), so that

$$\begin{aligned}\hat{f}^c_{ij}(W_i, W_j; n_{ij}) &= \frac{1}{2}\Big(f^c(W_i) + f^c(W_j)\Big)\cdot n_{ij} \\ &- \frac{1}{2}\left|\lambda^c_{ij}\right|\left\{\chi(\bar{\varepsilon}^{(2)}_{ij}, \bar{\varepsilon}^{(4)}_{ij})(W_j - W_i)\right\},\end{aligned} \tag{4.8}$$

is the flux to be differentiated, as in LU-SGS, or (b) explicitly neglect derivatives of the dissipation with respect to next-neighbours, whereby derivatives with respect to fourth-order dissipation terms are included for the immediate neighbours.

Given that the solution is smooth almost everywhere, the true residual will contain a 4th difference operator almost everywhere, and neither of these simplifications is satisfactory. Consider Table 4.3, which gives the weights of 2nd differences, 4th differences, and the "chopped" 4th differences of option (b) in 1D. The operator of option (a) is completely at variance with the 4th difference operator, with even the signs of the corresponding weights differing. On the other hand choosing option (b) (the last row of Table 4.3) results in a system that is considerably more difficult to solve than the full unapproximated system, even though it is more diagonally dominant. This could be understood to be a result of fact that the chopped fourth-order dissipation operator contains a large second-order anti-dissipation component. In the absence of a really satisfactory approximation, the pure second-order dissipation operator is used in the following.

It remains to choose the function χ, which is necessary as both $\bar{\varepsilon}^{(2)}$ and $\bar{\varepsilon}^{(4)}$ are zero in particular regions of the solution, but are never zero together due to the shock switch. Simply using $\chi = k^{(2)}$, the constant coefficient of second-order dissipation, is equivilent to the Jacobian of a Lax-Friedrichs scheme as for LU-SGS. Better non-linear convergence can be obtained with the choice of (Wong & Zingg, 2005), namely

$$\chi(\bar{\varepsilon}^{(2)}, \bar{\varepsilon}^{(4)}) = \bar{\varepsilon}^{(2)} + \sigma\bar{\varepsilon}^{(4)}. \tag{4.9}$$

Numerical experiments suggest that a value of σ in the range $10 - 20$ is appropriate. In the remainder of this thesis $\sigma = 15$ is used.

Decoupled Turbulence Treatment

As in the case of LU-SGS, the turbulence equations are decoupled from the mean-flow equations in the FOKI method. However, in order to increase the flexibility of the turbulence treatment the turbulence equations are also formulated and solved

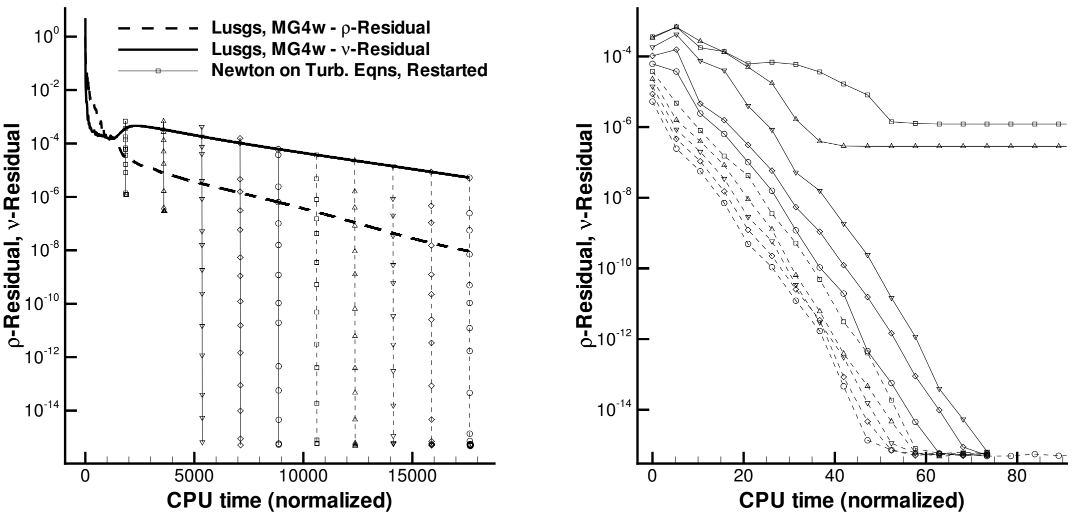


Figure 4.7: Convergence of the turbulence equation alone, using an exact Newton method with ILU(0)+GMRES solution of the linear system, restarted from several points along a normal convergence history. The right-hand shows a zoom of the same iterations, shifted to zero. Note that all histories are in terms of CPU time.

separately. This is particularly important for FOKI, as the Jacobian of the turbulence equations is exact, therefore an exact Newton method is possible, whereas for the mean-flow equations the Jacobian includes a sizeable approximation. The discrepancy suggests that the turbulence equations will converge much faster than the mean-flow equations, and therefore a balanced method should spend more effort on the mean-flow convergence.

In order to solve the two systems separately, the evaluation of the turbulence residual and Jacobian, without necessarily computing the mean-flow equivalents, and vice versa, is implemented. The speed with which the turbulence equations can be solved is demonstrated in Figure 4.7. The case is the RAE2822 Case 9 with a Spalart-Allmaras Edwards one-equation turbulence model. The baseline convergence is obtained with an LU-SGS multigrid method, and the residual of both the mean-flow and turbulence equations are shown. Every 200 iterations of the baseline convergence a turbulence equation Newton method is started, consisting of the turbulence equation part of FOKI.

Note that all convergence histories in Figure 4.7 are shown in terms of CPU time; it can be seen that the exact solution of the turbulence equation for a given mean flow can be performed in a time equivalent to approximately one-four hundredth of the time needed for the solution of the full system. This is partly due to the very high convergence rate of the Newton method, and partly due to the very cheap turbulence residual and Jacobian evaluation, as a result of it being a scalar equation. For example the Jacobian in this case is 25 times smaller than the Jacobian of the mean flow equations.

The optimal combination of mean-flow and turbulence iterations is investigated numerically in Section 4.5.3.

	Std. *TAU*	+ Jac. storage	+ Linear sol. storage
Memory (Bytes)	25M	82M	122M
Factor increase	$\times 1.0$	$\times 3.3$	$\times 4.9$
Points in 1GB	2.0×10^6	0.61×10^6	0.41×10^6

Table 4.4: The memory requirements of the FOKI method with a one-equation turbulence model and explicit and separate storage of mean-flow and turbulence Jacobians, as compared to the standard *TAU*-code - measured for a *t*wo-dimensional unstructured grid with 50×10^3 points. Also given are the maximum number of points that would fit in 1GB of memory. The linear solver used is ILU(0) preconditioned GMRES(20).

Linear Solution Method

The resulting linear equation is solved with ILU(0) preconditioned GMRES, which was sufficient in all cases tested. The method is easy to parallelize: as the Jacobian construction is first-order it can be performed in a loop over all grid faces, exactly as for the non-linear residual. ILU is performed per domain (analogously to the GS iteration in LU-SGS), and parallelization of GMRES consists merely in providing a parallel version of the Jacobian-vector product. Parallelization of the Krylov solver and preconditioner is provided by PETSc (Balay *et al.*, 2006), although only sequential results are given here.

Given a ILU(0) preconditioned GMRES(20) linear solver, and separate explicit storage of the first-order mean-flow and turbulence Jacobians, the memory requirements of the FOKI method for typical two-dimensional turbulent cases are given in Table 4.4 (as compared with Table 4.1 for the full linearized code). Apart from the significant reduction in memory requirements attributed to the reduction in size of the Jacobian (through separation of the turbulence equations, as well as the reduction to first order), the ILU(0) algorithm requires little additional memory, the partial factorization being stored in the entries of the original matrix.

4.5.2 Alternative 1st-Order Implicit Method (FOGSI)

For the purposes of comparison a method similar to FOKI is implemented, denoted as the First-Order Jacobian, Gauss-Seidel Implicit (FOGSI) method, which has been recently proposed by Rossow for high-Reynolds number problems (Rossow, 2005; Roberts & Swanson, 2005). The method also uses an approximate Jacobian, which is based only on the first-order Roe scheme, but in contrast to FOKI the Jacobian is not stored, rather the Jacobian-vector product is recalculated each time it is required, the resulting system being solved with SGS iterations. FOGSI(n) then denotes the method with n SGS iterations on the linear system per step, and it is used as an FAS multigrid smoother. In the original paper it is also used within an explicit Runge-Kutta method, but this was found to be less efficient than using it directly in this case.

The coefficient of the dissipation in the Roe flux is taken to be constant when obtaining the Jacobian, so that the convective terms of the Jacobian take the form

$A \pm |A|$ (where $A = \partial f^c / \partial W$), which is evaluated using flow variables on the grid faces with either constant or affine reconstruction.

The core idea of the method is to make the product $(A \pm |A|) \cdot \Delta W$, for some flow state update ΔW, very computationally efficient, thereby making the entire method efficient. This is achieved by firstly working entirely in primitive variables, and secondly rewriting the above product in terms of the Mach number as follows (in two-dimensions):

$$A \pm |A| = \begin{pmatrix} V^{\pm} & \rho n_x M^{\pm} & \rho n_y M^{\pm} & \pm \frac{1}{a} \hat{M} \\ 0 & V^{\pm} \pm n_x^2 a \hat{M} & \pm n_x n_y a \hat{M} & \frac{n_x}{\rho} M^{\pm} \\ 0 & n_x n_y a \hat{M} & V^{\pm} \pm n_y^2 a \hat{M} & \frac{n_y}{\rho} M^{\pm} \\ 0 & n_x \rho a^2 M^{\pm} & n_y \rho a^2 M^{\pm} & V^{\pm} \pm a \hat{M} \end{pmatrix}, \tag{4.10}$$

where $V = U \cdot n$, $V^{\pm} = V \pm |V|$, $M^{\pm} = (1 \pm M_0)$, $\hat{M} = 1 - |M_0|$, and

$$M_0 = \begin{cases} 1 & M > 1 \\ M & -1 \leq M \leq 1 \\ -1 & M < -1 \end{cases} .$$

This is not an approximation to the expression $A \pm |A|$, purely a rewrite, in a similar vein to the efficient calculation of $|A|$ due to Turkel in conservative variables (Turkel, 1988). The values $V^{\pm}$, $M^{\pm}$ and $\hat{M}$ are calculated once at the beginning of a non-linear step and stored on the faces of the grid, further reducing the computational cost.

In the implementation described here the viscous Jacobians are based on TSL fluxes assuming constant viscosity, and the convective Jacobians are based on flow values reconstructed on the grid faces. Also a one-equation turbulence model is used, and differentiated exactly, in contrast to (Rossow, 2005) where an algebraic model is used. Another difference to previous results involves the boundary conditions, which are also treated exactly in the Jacobian in this case. The inverse of the block diagonal of the Jacobian is calculated explicitly using Gaussian elimination once, and stored until the Jacobian is recalculated. This was found by the present author to be faster than the method proposed by the originator, of performing a few GS iterations on the block diagonal at each linear iteration. It has the disadvantage of increasing the memory requirements of the scheme.

In summary the distinguishing differences from FOKI are the absence of any storage of the Jacobian, the avoidance of Krylov methods in favour of simpler GS iterations, the use as an FAS multigrid smoother, and the lack of explicit separation of mean-flow and turbulence equations (although this latter is of course possible in FOGSI too). Being based on a Roe flux, FOGSI performs well with upwind and central with matrix-dissipation schemes on the RHS, but poorly (in fact it is unstable in most cases) with the scalar dissipation scheme as currently used in *TAU*.

4.5.3 Numerical Comparison of LU-SGS, FOKI and FOGSI

In order to initially separate convergence behaviour due to the turbulence equations from that involving only the mean flow equations, we consider a laminar test case,

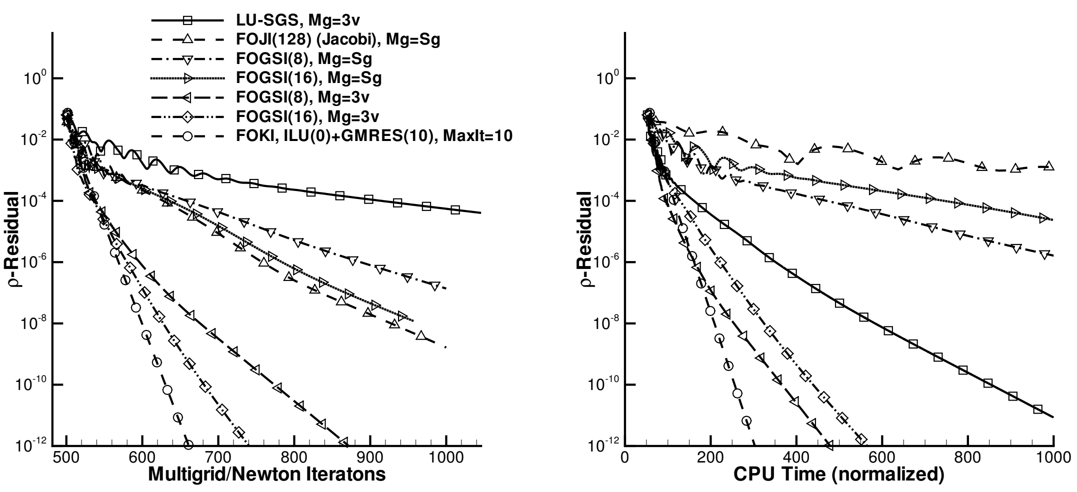


Figure 4.8: Convergence histories for the NACA0012 laminar case with Re = 5000, and a variety of time stepping methods, all started after 500 iterations of a single-grid method. The convective flux discretization is Roe. `Mg=Sg/3v` refers to the multigrid cycle type; `Sg` stands for single-grid, i.e. no multigrid. All calculations used CFL=100, except for FOKI which used an infinite CFL number.

a NACA0012 at a chord Reynolds number of 5000, a Mach number of 0.73, and an angle-of-attack of 0°. The grid, shown in Figure 3.13, is a structured single-block grid with about 20,000 points, and the first layer above the wall is chosen to achieve a *first-cell Reynolds number*, y^+, of the order of 1. The Reynolds number is just below the value at which the flow becomes turbulent, and the field is entirely subsonic, as well of course remaining attached to the aerofoil over its entire surface. In order to accommodate the FOGSI method, which is only formulated for Roe convective fluxes, Roe is used in the spatial discretization. Viscous fluxes are TSL and are therefore their implicit treatment is exact in both FOGSI and FOKI. In order to avoid start-up problems calculations are started from a partially converged solution, obtained after 500 iterations of an alternative method, in this case LU-SGS without multigrid.

First the appropriate choice of the number of SGS iterations in FOGSI is investigated. Figure 4.8 compares the convergence of several possible methods, in particular FOGSI(8) and FOGSI(16) both with and without multigrid. Also investigated is the same method, but with a Jacobi rather than a SGS linear system iteration (FOJI). A Jacobi iteration is significantly computationally cheaper than an SGS iteration, however, as seen in the figure, about 128 Jacobi iterations per step are needed to reach the same convergence rate as 16 SGS iterations, at which point FOJI is significantly slower. This is consistent with well known performance results, as well as the investigations of Section 3.4.2. The best performing FOGSI method used 8 SGS iterations per step and a 3V multigrid cycle, and results in a improvement on the efficiency of LU-SGS of a factor of ≈ 2.5 in terms of CPU time. In all of the above calculations the CFL number was fixed at 100.

The FOKI method with a Jacobian based on the Roe flux is also applied to this case. ILU(0) preconditioned GMRES(10) was used, and the linear iteration was stopped after a factor of 100 reduction in the preconditioned residual, or after 10

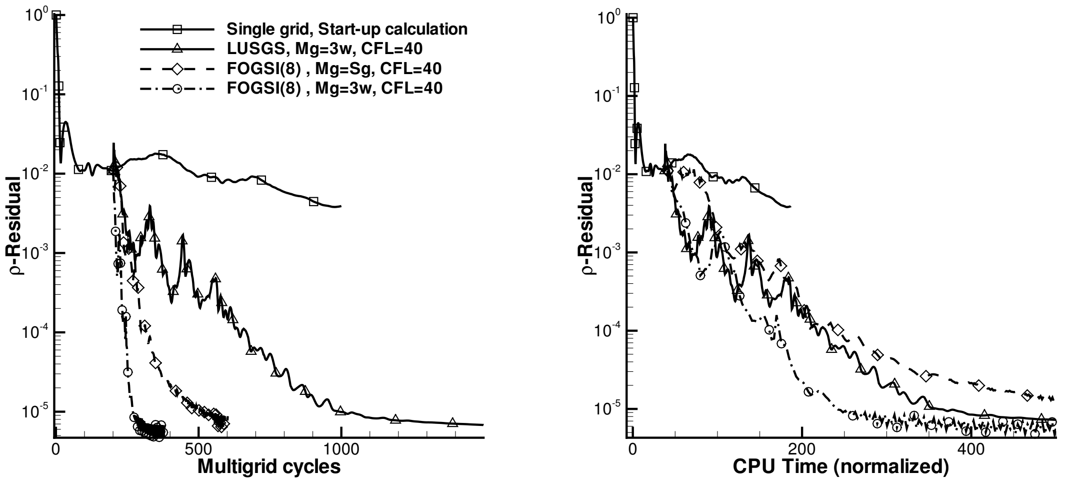


Figure 4.9: Convergence for RAE2822 Case 9, with Roe fluxes and Spalart-Allmaras turbulence. The values chosen for CFL and multigrid cycle were close to optimal values for each of the schemes considered.

iterations. In practice, after a short initial convergence phase, the method achieved the required factor of 10 reduction within $2-5$ GMRES steps. As a result of the small number of iterations the dominant cost of the calculation lay with the formulation the Jacobian and the ILU(0) preconditioner. To mitigate this effect the Jacobian was henceforth only calculated every 10 non-linear iterations, a modification that is very simple to implement given its explicit storage. The resulting convergence is shown in Figure 4.8 whereby the CFL number is taken to be infinite.

For this case the Jacobians of FOKI and FOGSI are substantially identical, the principal difference between the methods being the choice of CFL number, and the accuracy of linear system solution. For CFL numbers higher than 100 FOGSI was not stable for the number of SGS iterations used in this case. However given exact inner system solutions, FOKI and FOGSI showed identical convergence, verifying the implementation. With the modifications proposed here the FOKI scheme is approximately a factor of 4 faster in terms of CPU time, than LU-SGS with multigrid in terms of CPU time.

The next case considered is the RAE2822 Case 9 with the Spalart-Allmaras Edwards (SAE) turbulence model, again using Roe fluxes to cater to the FOGSI scheme. The grid is mixed-element with about 28,000 points. Convergence for LU-SGS and FOGSI is shown in Figure 4.9. None of the methods used converged beyond a residual of about 10^{-5}. Further FOKI could not be brought to converge at all. This is a common effect observed when applying upwind schemes with Green-Gauss gradient reconstruction on heavily anisotropic grids, and is due to oscillation of the gradient limiter. The limiter is designed to be active only in regions of large gradients, and can get caught in a limit cycle where it repeatedly swings between neighbouring cells on subsequent iterations. This can be diagnosed by examining the flow field at each iteration, but is difficult to resolve.

In order to calculate the RAE Case 9 using FOGSI, the central scheme with matrix dissipation is chosen for the spatial discretization. The coefficient of dissipation in

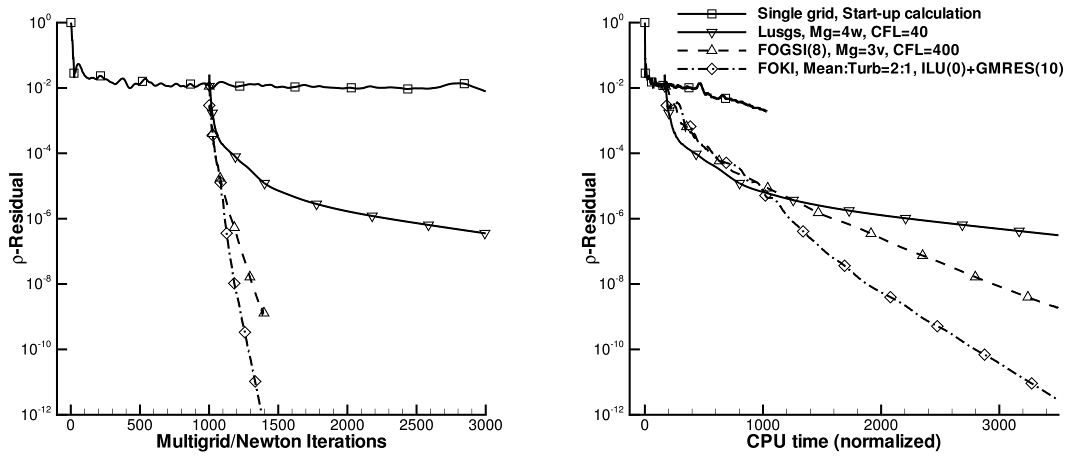


Figure 4.10: Convergence for RAE2822 Case 9, with the central scheme with matrix dissipation and Spalart-Allmaras turbulence. Settings of all methods were chosen to optimize their convergence speed. `Mean:Turb` refers to the number of turbulence equation iterations per mean equation iteration. In the left-hand plot one iteration refers to a complete cycle of (possibly multiple) mean flow iterations and a turbulence iteration.

the scheme is $|\partial f/\partial W|$, i.e. identical to that of the Roe scheme, and therefore the FOGSI Jacobian should be appropriate. However FOGSI is now at a disadvantage to FOKI, as FOKI uses a Jacobian based on the matrix dissipation fluxes in this case, taking into account the pressure switch over χ. The convergence plots are shown in Figure 4.10, compared again against LU-SGS with multigrid.

Immediately obvious is the poor behaviour of LU-SGS for this case, with early multigrid breakdown and very poor convergence thereafter (in particular as compared with Figure 4.11). FOGSI and FOKI both have very high convergence rates, and FOGSI shows no multigrid breakdown. The cost of FOKI and FOGSI iterations turns out to be approximately equal in this case, which gives the advantage to the method with the better convergence rate, FOKI. This difference may be attributed to the higher linear system convergence.

Finally we consider a discretization using the central scheme with scalar dissipation - the ultimate target scheme - in Figure 4.11. FOGSI is not stable with this discretization, a result of the mismatch between the Jacobian and the RHS. This statement can again be verified by using FOKI with a Roe Jacobian for this case, which also diverges. As discussed LU-SGS performs well, with almost no convergence breakdown, in contrast to its use with matrix dissipation in Figure 4.10. These results suggest that multigrid breakdown can be entirely avoided (at least for simple cases) by using an implicit treatment with an accurate Jacobian.

The performance of FOKI is compared against the case where the Jacobian is exact with respect to the RHS, i.e. where the RHS contains only first-order dissipation fluxes. Given an exact linear system solution at each step the method should then show quadratic convergence, and this has been verified. Given a factor of 100 reduction in the linear residual per step, the convergence is as shown in Figure 4.11. Also compared is the method with only one turbulence iteration for each two mean

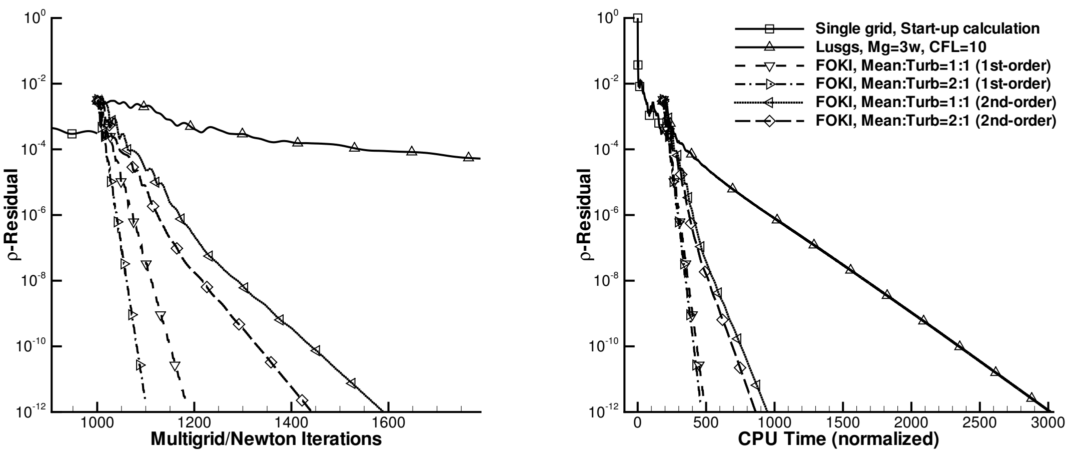


Figure 4.11: Convergence for RAE2822 Case 9, with the central scheme with scalar dissipation and Spalart-Allmaras turbulence. The FOKI scheme uses ILU(0) preconditioned GMRES(10) with a maximum of 10 iterations per linear system. In the left-hand plot one iteration refers to a complete cycle of (possibly multiple) mean flow iterations and a turbulence iteration.

flow iterations.

The choice of mean:turbulence iteration ratio has little effect on the first-order accurate problem, the 2:1 ratio being very slightly more efficient. This might be expected as in this case the Jacobian for all parts of the discretization are exact, and hence the convergence of the mean flow equations should be as good as that of the turbulence equations. Additionally, the extra turbulence iteration is very cheap compared to a mean flow iteration. The effect is somewhat larger for the second-order discretization, but is still not significant. Further study is necessary to determine the optimal use of the mean-turbulence splitting.

The reduction in convergence rate when switching to second-order is significant, as expected, and as seen earlier in Section 3.4.1. However in terms of CPU time per iteration, the difference is very small, even though the second-order residual is much more expensive than the first. This is a consequence of the fact that in FOKI the residual is only a small part of the total calculation cost (the Jacobian being more expensive for example), whereas for LU-SGS and Runge-Kutta the cost of residual evaluation dominates. Thus for more expensive residuals, methods like FOKI become increasingly attractive.

As compared with LU-SGS - which performed very well in this case - the best FOKI method is about 4.4 times faster in terms of CPU time. This is a significant difference, and since the memory requirements of FOKI are reasonable, it may be considered to be a promising scheme for large practical 3D cases in the future.

4.6 Summary

Two methods have been developed about a Jacobian based on first-order convective fluxes, including boundary conditions, viscous fluxes, and in particular an exact Jacobian of the turbulence model. One of the methods uses Gauss-Seidel to solve the linear system at each non-linear step (FOGSI), and the other uses a preconditioned GMRES algorithm, whereby the linear mean flow and turbulence equations are formulated and solved separately, and the method makes no use of multigrid (FOKI).

These schemes were tested on a simple two-dimensional single-element transonic aerofoil configuration, using a hybrid grid consisting of triangles and quadrilaterals, with a high Reynolds number resembling that encountered in practical applications. The FOKI method was seen to consistently out-perform both LU-SGS and FOGSI with multigrid, and resulted in an improvement in CPU time of a factor of 4-5 over the LU-SGS scheme, which in turn showed a performance improvement of a factor of two over Runge-Kutta.

The code with the FOKI scheme has memory requirements that are about five times those of the standard code, but has the advantage that it is not dependent on multigrid, and is therefore independent of the unreliable coarse grid agglomeration algorithm.

It remains to examine the performance, and in particular the robustness of the method for large three-dimensional test cases, such as those that have been considered in the context of LU-SGS. The best use of the separation of turbulence and mean-flow must be examined, as well as the start-up procedure. The development of FOKI into a practically usable scheme represents considerable work, but the method has been shown to be promising.

Chapter 5

The Discrete Adjoint Equations[1]

5.1 Introduction

We consider gradient-based optimization, characterized by the steepest descent method. A critical component is the evaluation of the gradient of the quantity of interest (the cost function) with respect to the parameterization of the problem (the design variables). Aerodynamic optimization problems often involve the very detailed parameterization of shapes, and as such a large number of design variables are required. The derivative of a cost function with respect to many design variables can be computed in a time only weakly dependent on the number of design variables using the *adjoint* method. An overview of the situation has been given in Section 1.2.

An ambitious goal of aerodynamic design is the gradient-based optimization of three-dimensional high-lift transport aircraft configurations using an unstructured RANS code. This is an extremely ambitious objective requiring the resolution of a number of very significant problems before it becomes practicable (Kroll *et al.*, 2004). Some of the difficulties involved are:

(a) The extensive grids necessary in order to accurately resolve the wakes of the individual elements of the wing (which control the onset of separation on the upper surfaces) and the associated high computational costs (Rudnik *et al.*, 2004). This critical and difficult problem has been tackled in Chapters 3 and 4.

(b) Grid deformation which can robustly handle adjacent bodies with large relative motion in large grids.

(c) Calculating gradients of the maximum lift $C_L^{\max}$ (Kim *et al.*, 2002), especially considering that determining the value of $C_L^{\max}$ itself is problematic.

(d) Taking unsteady flow in the non-linear solver and in the design process into account.

(e) The many design variables needed for parameterization of complex 3d shapes (Wild, 2003), which makes an adjoint method for the gradients essential for

[1]The author is very grateful for the support of Joël Brezillon, who performed all optimizations shown in this chapter in the context of (Dwight & Brezillon, 2006). The author performed the remainder of the work, in particular the conception and development of the discrete adjoint discretization and solution method.

efficiency. However adjoint techniques have difficulty handling turbulence modelling, in the case of the continuous formulation (Nadarajah & Jameson, 2000), or are too memory hungry to be applied to large 3d grids, in the case of the discrete formulation (Brezillon & Dwight, 2005).

This chapter tackles solely the last problem for the discrete adjoint in the context of two-dimensional high-lift optimization. There are two principal difficulties associated with the use of the discrete adjoint: firstly formulating the adjoint requires differentiating the corresponding flow solver per hand, including discrete boundary conditions, gradient calculations, turbulence models, etc., which as has been seen in Chapter 4, is a laborious process, and which must be repeated each time the spatial discretization changes. Secondly, depending on the manner of constructing the adjoint residual, storage of the full discrete flux-Jacobian may be required, which limits 3d applications due to memory requirements.

Alternatives to hand-differentiation are being developed, for example complex variable finite differences (Nielsen & Kleb, 2005; Burdyshaw & Anderson, 2005), and algorithmic differentiation (Griewank, 2000), but the solution many authors have used is to perform only an approximate differentiation of a flow solver. For example by treating the coefficient of artificial viscosity in the Jameson-Schmidt-Turkel scheme (Jameson *et al.*, 1981) as constant it is possible to reduce the construction of the adjoint residual to two sweeps over the faces of the grid (Mavriplis, 2004; Mavriplis, 2005). Another example is the practice of assuming that the eddy-viscosity is constant, thereby obviating the differentiation of the turbulence model (Kim *et al.*, 2003).

There have, however, been few studies examining the effect these approximations have on the resulting gradients and optimizations; and this is consequently a matter of pressing interest to the community. It is the purpose of this chapter to determine which of these simplifications are acceptable in the context of aerodynamic optimization, in the sense of how the convergence of the optimization and its result are affected. In particular the approximations considered are:

- Adjoint solution based on a 1st-order accurate discretization (FOA),
- Adjoint solution with Thin Shear-Layer viscous fluxes (TSL),
- Assumption of constant coefficients in the JST scheme (CCA),
- Assumption of constant eddy-viscosity (CEV),
- Adjoint solution with alternative turbulence model (ATM).

Each of these approximations is either based on the assumption that the derivatives of the particular terms are negligible, or that they may be replaced with the derivatives of similar related terms. Two optimization test cases are examined numerically for each approximation: (i) drag reduction of a transonic aerofoil, where the design problem essentially consists of the removal of the shock, and (ii) drag reduction of a high-lift configuration in which a wide variety of flow phenomena are represented, and for which both compressible and viscous effects, as well as the choice of turbulence model are critical. By considering such a wide range of flow phenomena, it is anticipated that the results obtained will be valid for general two-dimensional aerodynamic optimizations, and it is hoped that they will also be representative of the situation in three-dimensions.

With this goal the exact discrete adjoint to the unstructured finite volume RANS solver, the DLR *TAU*-code, is constructed based on the formulation of the Jacobian described in Chapter 4. The gradients obtained from the exact adjoint are verified against those obtained using finite differences on the original non-linear routines. Where the approximate adjoint formulations are the result of using a related discretization, as for the adjoint based on 1st-order fluxes, finite differences have again been used to verify the implementation. Gradient evaluations and thereafter full optimizations are then performed with each of the various adjoint approximations, and variations in gradients, convergence rates and solutions attained are compared.

The solution of the linear adjoint problem is performed using a Krylov subspace method with ILU preconditioning, allowing the solution of the adjoint problem in a CPU time equivalent to about 5% of the time required for the main problem. In addition the adjoint fields for multiple cost-functions may be computed simultaneously, further reducing the CPU-time cost of the gradient evaluation.

5.1.1 Overview

Section 5.2 provides a short formal description of the design problem and introduces the necessary notation. Section 5.3 introduces the adjoint method, and in particular it is shown how it is possible to evaluate the adjoint solution with an effort equivalent to 5% of that needed to evaluate the flow solution itself (Brezillon & Dwight, 2005), making optimization methods that require large numbers of gradient evaluations, such as the Quasi-Newton Trust Region (QNTR) approach, attractive in an aerodynamic design context for the first time.

The simplifying approximations for the Jacobian are described in detail in Section 5.4. In order to determine whether or not the gradients remain accurate enough for use in optimization, and therefore which Jacobian approximations are admissible, each approximation is numerically tested within the conjugate gradient method for the drag reduction of a transonic single element aerofoil and a multi-element high-lift configuration, Section 5.5. It is seen that the Jacobian may be simplified considerably without seriously damaging the accuracy of the gradients, and that resulting optimizations are barely affected (Dwight & Brezillon, 2006).

5.2 Aerodynamic Design Problem

The optimization problem may be stated as follows: minimize $I(W, X, D)$ a cost-function with respect to some set of design variables D, subject to the constraints $R(W, X, D) = 0$ and $\mathcal{G}(X, D) = 0$; whereby W and X are functions of D, and R and $\mathcal{G}$ are general non-linear operators.

Here I is typically an aerodynamic force integrated over the geometry such as lift, drag or pitching moment, W represents the flow variables, X the computational mesh, R the residual resulting from the discretization of the fluid flow equations, and G a mesh deformation operator. In particular R is here the finite volume discretization on an unstructured mesh of the Navier-Stokes equations. Finally D is some parameterization of the geometry and onflow conditions.

For problems with a large number of design variables, the most efficient algorithms are gradient-based, and require the evaluation of $\mathrm{d}I/\mathrm{d}D$ for each design variable. These gradients may in turn be efficiently evaluated using the adjoint approach. Additional constraints on the problem are also common, such as the specification of constant lift and constant pitching moment.

5.2.1 Details of the Discretization

The spatial dicretization is exactly the finite volume discretization of the Navier-Stokes equations described in Chapter 2. The central numerical flux with mixed second- and fourth-order scalar dissipation operators is applied (Jameson *et al.*, 1981), see Section 2.6. Viscous and turbulence diffusion fluxes use Green-Gauss gradients averaged onto the cell faces, as per (2.75), and thereby have a stencil of next-neighbours. Turbulence convective fluxes are 1st-order upwind, sources use gradients from Green-Gauss.

5.3 Gradients via Discrete Adjoint

The adjoint approach allows the rapid evaluation of $\mathrm{d}I/\mathrm{d}D$ for a large number of design variables $|D|$. It can be readily understood by contrasting it with the direct or primal approach.

5.3.1 Primal Approach

The most direct approach to evaluation of the gradient is to apply the chain rule to $\mathrm{d}I/\mathrm{d}D$, to give

$$\frac{\mathrm{d}I}{\mathrm{d}D} = \frac{\partial I}{\partial W}\frac{\mathrm{d}W}{\mathrm{d}D} + \frac{\partial I}{\partial X}\frac{\mathrm{d}X}{\mathrm{d}D} + \frac{\partial I}{\partial D},$$

which is an expression for $\mathrm{d}I/\mathrm{d}D$ in terms of $\mathrm{d}W/\mathrm{d}D$ and $\mathrm{d}X/\mathrm{d}D$ (the remaining quantities being readily calculable). By noting that $\mathrm{d}R/\mathrm{d}D = 0$ - as the condition $R = 0$ should hold for all D - we have

$$\frac{\mathrm{d}R}{\mathrm{d}D} = \frac{\partial R}{\partial W}\frac{\mathrm{d}W}{\mathrm{d}D} + \frac{\partial R}{\partial X}\frac{\mathrm{d}X}{\mathrm{d}D} + \frac{\partial R}{\partial D} = 0, \tag{5.1}$$

a linear system for $\mathrm{d}W/\mathrm{d}D$, based on the linearization of the discretized flow equations. Hence to find the sensitivity of I to $|D|$ design variables it is necessary to solve (5.1) $|D|$ times, and in practice this effort dominates the total cost of the calculation.

5.3.2 Adjoint Approach

Instead of applying the chain rule to I, apply it to the *Lagrangian*:

$$\mathcal{L}(W, X, D, \Lambda) = I(W, X, D) + \Lambda^T R(W, X, D),$$

where Λ are known as the *adjoint variables*. Since $R = 0$ for all D, $\mathcal{L} = I$ for all Λ and all D. Hence

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}D} = \frac{\mathrm{d}I}{\mathrm{d}D}, \qquad \forall\, \Lambda, D.$$

Applying the chain rule to $\mathcal{L}$

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}D} &= \left\{\frac{\partial I}{\partial W}\frac{\mathrm{d}W}{\mathrm{d}D} + \frac{\partial I}{\partial X}\frac{\mathrm{d}X}{\mathrm{d}D} + \frac{\partial I}{\partial D}\right\} + \Lambda^T \left\{\frac{\partial R}{\partial W}\frac{\mathrm{d}W}{\mathrm{d}D} + \frac{\partial R}{\partial X}\frac{\mathrm{d}X}{\mathrm{d}D} + \frac{\partial R}{\partial D}\right\}, \\ &= \left\{\frac{\partial I}{\partial W} + \Lambda^T\frac{\partial R}{\partial W}\right\}\frac{\mathrm{d}W}{\mathrm{d}D} + \left\{\frac{\partial I}{\partial X} + \Lambda^T\frac{\partial R}{\partial X}\right\}\frac{\mathrm{d}X}{\mathrm{d}D} + \left\{\frac{\partial I}{\partial D} + \Lambda^T\frac{\partial R}{\partial D}\right\} \qquad (5.2)\end{aligned}$$

after rearranging. The unknown quantity $\mathrm{d}W/\mathrm{d}D$ may then be eliminated by choosing Λ such that

$$\left(\frac{\partial R}{\partial W}\right)^T \Lambda = -\left(\frac{\partial I}{\partial W}\right)^T, \qquad (5.3)$$

whereby the first bracketed term of (5.2) is zero. This is the *adjoint equation*, and must be solved only once to evaluate the gradient of a single I with respect to any number of design variables. Given Λ, the gradient is

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}D} = \left\{\frac{\partial I}{\partial X} + \Lambda^T\frac{\partial R}{\partial X}\right\}\frac{\mathrm{d}X}{\mathrm{d}D} + \left\{\frac{\partial I}{\partial D} + \Lambda^T\frac{\partial R}{\partial D}\right\},$$

whereby $\partial I/\partial D$ and $\partial R/\partial D$ are zero for shape-based design variables, and the remaining unknown term, $\mathrm{d}X/\mathrm{d}D$, may be reliably evaluated by finite differences,

$$\begin{aligned}\frac{\partial I}{\partial X}\frac{\mathrm{d}X}{\mathrm{d}D}\Delta D_i &\approx \frac{I(W, X(D+\epsilon\Delta D_i), D) - I(W, X, D)}{\epsilon}, \\ \frac{\partial R}{\partial X}\frac{\mathrm{d}X}{\mathrm{d}D}\Delta D_i &\approx \frac{R(W, X(D+\epsilon\Delta D_i), D) - R(W, X, D)}{\epsilon},\end{aligned}$$

as the result is relatively insensitive to choice of ϵ, in contrast to direct approximation of $\mathrm{d}I/\mathrm{d}D$ by finite differences.

5.3.3 Adjoint of the Grid Deformation

In order to eliminate the use of finite differences for $\mathrm{d}X/\mathrm{d}D$, it is possible to use an adjoint approach, thereby removing the expense of deforming the grid in response to each shape-modifying design variable, which is the dominant cost in the gradient calculation (Nielsen & Park, 2005). Apply the chain rule to a modified Lagrangian

$$\mathcal{L}(W, X, D, \Lambda_R, \Lambda_{\mathcal{G}}) = I(W, X, D) + \Lambda_R^T R(W, X, D) + \Lambda_{\mathcal{G}}^T \mathcal{G}(X, D),$$

where two sets of adjoint variables are used, one for each of the two constraints, to give

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}D} &= \left\{\frac{\partial I}{\partial W}\frac{\mathrm{d}W}{\mathrm{d}D} + \frac{\partial I}{\partial X}\frac{\mathrm{d}X}{\mathrm{d}D} + \frac{\partial I}{\partial D}\right\} + \Lambda_R^T \left\{\frac{\partial R}{\partial W}\frac{\mathrm{d}W}{\mathrm{d}D} + \frac{\partial R}{\partial X}\frac{\mathrm{d}X}{\mathrm{d}D} + \frac{\partial R}{\partial D}\right\} \\ &+ \Lambda_{\mathcal{G}}^T \left\{\frac{\partial \mathcal{G}}{\partial X}\frac{\mathrm{d}X}{\mathrm{d}D} + \frac{\partial \mathcal{G}}{\partial D}\right\}, \\ &= \left\{\frac{\partial I}{\partial W} + \Lambda_R^T\frac{\partial R}{\partial W}\right\}\frac{\mathrm{d}W}{\mathrm{d}D} + \left\{\frac{\partial I}{\partial X} + \Lambda_R^T\frac{\partial R}{\partial X} + \Lambda_{\mathcal{G}}^T\frac{\partial \mathcal{G}}{\partial X}\right\}\frac{\mathrm{d}X}{\mathrm{d}D} \\ &+ \left\{\frac{\partial I}{\partial D} + \Lambda_R^T\frac{\partial R}{\partial D} + \Lambda_{\mathcal{G}}^T\frac{\partial \mathcal{G}}{\partial D}\right\}.\end{aligned}$$

Then as before $\mathrm{d}W/\mathrm{d}D$ may be eliminated by choosing Λ_R to satisfy

$$\left(\frac{\partial R}{\partial W}\right)^T \Lambda_R = -\left(\frac{\partial I}{\partial W}\right)^T, \tag{5.4}$$

but further $\mathrm{d}X/\mathrm{d}D$ may be eliminated by choosing $\Lambda_{\mathcal{G}}$ such that

$$\left(\frac{\partial \mathcal{G}}{\partial X}\right)^T \Lambda_T = -\left(\frac{\partial I}{\partial X}\right)^T - \left(\frac{\partial R}{\partial X}\right)^T \Lambda_R. \tag{5.5}$$

Thus only (5.4) and (5.5), must be solved for any number of design variables, after which $\mathrm{d}I/\mathrm{d}D$ may be written

$$\frac{\mathrm{d}I}{\mathrm{d}D} = \frac{\partial I}{\partial D} + \Lambda_R^T \frac{\partial R}{\partial D} + \Lambda_{\mathcal{G}}^T \frac{\partial \mathcal{G}}{\partial D}. \tag{5.6}$$

The biggest effort involved is the linearization of the grid deformation operator $\mathcal{G}$, and the evaluation of $\partial R/\partial X$ and $\partial I/\partial X$. If however a linear deformation operator is choosen, its linearization is trivial. This approach is not used in this thesis, but is to be the subject of future work.

5.3.4 Implementation of the Method

Implementation of the above procedure requires the ability to evaluate the quantities $(\partial R/\partial W)^T \Lambda$ - the *adjoint residual* - and $\partial I/\partial W$. The Jacobian is the same as the exact Jacobian discussed in Chapter 4, i.e. evaluated by hand and stored explicitly. As it is stored explicitly it is easy to transpose.

The fact that the Jacobian was formulated by differentiating with respect to primitive variables poses no problem as long as the cost functions are also differentiated with respect to primitive variables. The equations remain in conservative form, and therefore this choice has no effect on the final solution. Given the explicitly stored Jacobian - which must only be constructed once per gradient evaluation, even for multiple cost-functions - assessment of the adjoint residual reduces to a matrix-vector product. Further the availability of the matrix allows the application of ILU preconditioned Krylov methods, which have been shown to be very effective if the ILU fill-in level is chosen high enough, in Section 4.3. Again this preconditioner must only be constructed once per calculation.

As the eigenspectrum of the Jacobian and transpose Jacobian are identical, the convergence rates achieved with Krylov subspace methods for the two resulting linear systems are guaranteed to be identical, and so the experience gained applying these methods in implicit schemes in Chapter 4 may be directly carried over. The result is that the calculation of the adjoint solution requires only approximately 5% of the time required for a non-linear flow calculation - and so forms an insignificant component of the total time for the optimization. Given that the gradient is much cheaper than the line search, optimization strategies that rely on many gradient evaluations, such as Quasi-Newton Trust Region (QNTR) (Geiger & Kanzow, 1999), become more attractive. However, the ILU preconditioner and GMRES method have an associated memory cost that further reduces the size of grid that may be calculated in a given memory, in this case the requirements for ILU with 4 levels of fill-in and GMRES(30) were given in Chapter 4 in Table 4.1.

Note that a finite difference evaluation of the matrix-vector product, as was done in the Newton method of Section 3.4.1 to avoid calculating the Jacobian explicitly, is much more difficult to use for the adjoint residual, as the product with the *transposed* Jacobian is needed. It is necessary to split the field into several sets of grid nodes, such that no two nodes within a set are connected to each other by the stencil of the scheme (Nielsen & Kleb, 2005). Then the product of the Jacobian with the vector consisting of ones at the node positions of a given set, and zeros elsewhere, returns a vector containing the non-zero entries of the Jacobian in the columns where the nodes were positioned. This vector is approximated by finite differences. However, because of the block structure of the matrix the finite difference must be performed (number of equations) times for each point set. The method therefore requires 20-30 residual evaluations per adjoint residual, and is thereby more expensive than constructing the Jacobian explicitly.

5.4 Approximations of the Discrete Adjoint

In an attempt to reduce the memory requirements of the scheme, the manual effort required to adjoint new spatial discretizations, and the computational effort required to solve the resulting system, approximations to the adjoint equations are made. The cost is inaccuracies in the resulting gradients. Each approximation is described here and its advantages noted, while its accuracy will be numerically assessed in Section 5.5. Throughout no approximation of the boundary conditions is undertaken, as all are treated numerically with a stencil of a single (boundary) point, and therefore appear only on the diagonal of the Jacobian.

5.4.1 1st-Order Approximation (FOA)

As already noted in reference to simplified Jacobians of the central scheme in the context of implicit methods, see Section 4.5.1, by forming the Jacobian of first-order convective fluxes rather than second-order, the stencil of the Jacobian is reduced to immediate neighbours of a point only, and the resulting linear system is much better conditioned.

Here only simplifications of the JST flux are considered, and the same arguments as in Section 4.5.1 apply. In particular purely second-order dissipation is used:

$$\begin{aligned} \hat{f}^c_{ij}(W_i, W_j; n_{ij}) &= \frac{1}{2}\Big(f^c(W_i) + f^c(W_j)\Big)\cdot n_{ij} \\ &- \frac{1}{2}\left|\lambda^c_{ij}\right|\Big\{k^{(2)}(W_j - W_i)\Big\}, \end{aligned} \tag{5.7}$$

but rather than specifying the dissipation coefficient as a blend of second- and fourth-order coefficients as in (4.9), a constant dissipation factor $\bar{k}^{(2)}$ is used, which is in this case taken to be equal to $k^{(2)}$ in (2.39), i.e. $\bar{k}^{(2)} = 0.5$.

5.4.2 Thin Shear-Layer (TSL) Assumption

The formulation of the viscous fluxes given in Section 5.2.1 has a stencil including next-neighbours as a result of the use of gradients based on Green-Gauss. As in the implicit schemes, this stencil may be reduced to immediate neighbours if the TSL flux is used. However here the approximation has an effect on the converged result of the calculation. That the two viscous flux discretizations agree well for complex flows does not necessarily imply that their gradients will agree well. This will be examined.

5.4.3 Constant JST Coefficients Approximation (CCA)

By making the assumption that the ϵ, θ and $|A|$ in (4.6) are constant, the following simplification of the adjoint residual is possible (Mavriplis, 2004). Let $L(W)$ be as defined in (4.7), and treated in the expression for the convective fluxes as an independent variable, so that the derivative and adjoint of (4.6) may be written respectively

$$\frac{\mathrm{d}\hat{f}^c}{\mathrm{d}W} = \frac{\partial \hat{f}^c}{\partial W} + \frac{\partial \hat{f}^c}{\partial L} \cdot \frac{\partial L}{\partial W}, \tag{5.8}$$

$$\left(\frac{\mathrm{d}\hat{f}^c}{\mathrm{d}W}\right)^T = \left(\frac{\partial \hat{f}^c}{\partial W}\right)^T + \left(\frac{\partial L}{\partial W}\right)^T \cdot \left(\frac{\partial \hat{f}^c}{\partial L}\right)^T, \tag{5.9}$$

whereby all matrices on the right-hand sides above have immediate neighbour fill-in only, $\partial\hat{f}^c/\partial L$ is symmetric and $\partial L/\partial W$ is trivial.

Memory requirements are then approximately 1.6 times those of the FOA scheme if advantage is taken of the symmetry of $\partial\hat{f}^c/\partial L$, and $\partial L/\partial W$ is calculated on-the-fly. Also the matrices may be stored on the edges and nodes of the grid, given which the adjoint residual may be evaluated in two loops over all edges by introducing an intermediate variable Λ^* as follows:

$$\begin{aligned}\Lambda^* &= \left(\frac{\partial \hat{f}^c}{\partial L}\right)^T \cdot \Lambda, \\ \left(\frac{\mathrm{d}\hat{f}^c}{\mathrm{d}W}\right)^T \cdot \Lambda &= \left(\frac{\partial \hat{f}^c}{\partial W}\right)^T \cdot \Lambda + \left(\frac{\partial L}{\partial W}\right)^T \cdot \Lambda^*.\end{aligned}$$

TSL viscous and turbulence diffusion fluxes are used to similarly reduce the memory requirements of the viscous flux Jacobian, while the discrete turbulence Jacobians are formed exactly.

The CCA approximation may be justified by noting that the terms in the derivative that are neglected due to the approximation are of higher order in the grid spacing than the remaining terms. Therefore in the limit of zero cell size the influence of the approximation tends to zero, see Section 2.6.2.

Compared to FOA, this approach seems very favourable; for only sightly more memory the convective fluxes are sensibly approximated. However the resulting system is almost as poorly conditioned as the exact system and therefore similarly powerful linear solvers are required.

5.4.4 Constant Eddy-Viscosity (CEV) Assumption

One of the most demanding parts of the spatial discretization to differentiate by hand is the turbulence model, partly due to the wide variety of blending functions, limiters, vortex corrections etc., partly because of the many coupling points to the mean-flow equations, and partly because of the enormous selection of models available. More seriously it is very difficult to treat turbulence models in a continuous adjoint framework without resorting to continuous-discrete hybrids (Nadarajah & Jameson, 2000), hence some simplifying assumption must be made.

By assuming that derivatives of all turbulence quantities with respect to all flow variables are negligible, all turbulence terms are eliminated. For Spalart-Allmaras the derivatives of the eddy-viscosity are taken to be zero, for $k - \omega$ and $k - \epsilon$ models derivatives of k and eddy-viscosity are taken to be zero.

One place where this assumption might be invalid is in the adverse pressure gradient region following the shock on a transonic airfoil, where the large eddy-viscosity increases the surface shear-stress, directly affecting the aerodynamic forces.

5.4.5 Use of an Alternative Turbulence Model (ATM)

The only significant benefit that the CEV assumption confers is the avoidance of hand differentiating the turbulence model. The similarity in the formulation and results of, for instance, the original Spalart-Allmaras model (SA) (Spalart & Allmaras, 1992), and the modified Spalart-Allmaras-Edwards (SAE) (Edwards & Chandra, 1996), suggest that it may be reasonable to use the Jacobian of one as an approximation to the Jacobian of the other, and that this may be a better approximation than ignoring turbulence derivatives altogether. If so only one or two models of each type must be differentiated in a large code.

SA and SAE differ only in the formulation of the turbulence production term: for SA it is based on a measure of the flow vorticity, see Section 2.11.4, SAE uses a measure of the shear stress, see Section 2.11.5. Within the boundary-layer the source terms are dominated in both models by terms involving the wall-distance, hence the models are expected to differ most significantly in detached shear-layers and vortices.

Another candidate pair might be Wilcox's $k - \omega$ (Wilcox, 1998), and Mentor's $k - \omega$ SST (Menter, 1993), whereby only SA gradients as an approximation to SAE gradients are numerical evaluated in the following.

5.5 Results and Discussion

We wish to determine which, if any, of the previously described approximate Jacobians results in an adjoint gradient evaluation method that is sufficiently accurate for use in aerodynamic optimization. This is a question that is difficult to answer theoretically due to the complexity of the complete optimization process, and the somewhat arbitrary nature of the Jacobian approximations. Instead we investigate concrete optimization problems numerically and examine how each approximate adjoint solver performs in each case. In an attempt to obtain results that have a relatively general validity, two significantly different 2d optimization problems are considered representing a variety of industrially relevant flow physics, and - just as importantly - two different optimization algorithms are applied.

The first case is drag reduction of a transonic RAE2822 single element aerofoil at a Reynolds number of 6.5×10^6, and a Mach number of 0.730, whereby the lift must be held constant at a lift coefficient of 0.8. The computational grid is shown in Fig. 3.13. The geometry is parameterized using 20 design variables which modify the camberline of the aerofoil with Hicks-Henne bump functions (Hicks & Henne, 1978). The thickness of the aerofoil is not permitted to change, and as a result no additional geometrical constraints are necessary. The baseline geometry has a strong shock on the upper surface which is the main source of pressure drag; the optimization problem therefore substantially consists of the removal of the shock. Two gradient-based algorithms, the Conjugate Gradient (CG) method and the Quasi-Newton Trust Region (QNTR) method are applied to the problem.

The lift constraint is enforced explicitly by varying the angle of attack during the evaluation of the drag in the non-linear RANS solver, the so-called *target-lift* mode. Because we wish to minimize the drag at the target lift C_L^*, rather than at the preexisting lift C_L, the objective function must be modified to consider the lift constraint consistently (Reuther *et al.*, 1999),

$$I = C_D - \frac{(\partial C_D/\partial\alpha)}{(\partial C_L/\partial\alpha)}\left(C_L - C_L^*\right), \tag{5.10}$$

a consequence of which is that the accuracy of the gradients of lift are also important for the optimization.

The second optimization test-case is also drag reduction at constant lift, but of the three-element high-lift geometry in take-off configuration. The baseline geometry and flow conditions are derived from a test-case defined in the European project EUROLIFT II (Wild *et al.*, 2005), and operate at a Reynolds number of 14.70×10^6 and a Mach number of 0.17146. Constant lift is ensured in the same manner as for the RAE case. The computational grid is shown in Fig. 3.13 and is structured, whereby the structured topology is not employed by the solvers, all algorithms being implemented for general unstructured grids. Only the flap of the configuration is parameterized, and in such a manner that no modification to the composite clean wing is possible (Brezillon & Wild, 2005). The position and angle of attack of the flap relative to the main element may be changed (the so-called *setting* parameters), as may the sharpness of the flap's nose and the shape of the portion of the flap hidden by the main element (*shape* parameters). Only the conjugate gradient algorithm is examined for this case.

5.5.1 Transonic RAE2822 Aerofoil

Firstly exact adjoint gradients are compared with direct finite difference approximations to $\mathrm{d}I/\mathrm{d}D$; the gradients of total and viscous drag are plotted in Fig. 5.1. Note that the curves in this figure are "smooth" becuase of the indexing of the design variables, which parameterize bumps running around the surface of the aerofoil in order. Both finite differences and the discrete adjoint approximate the gradient of the *discretized* cost-function, and should therefore correspond with each other exactly on any given grid. It may be seen that the agreement is very good, the discrepancies apparent in the viscous drag gradients may be attributed to the rounding error in finite differences as a result of the very small absolute values of the gradients. This discrepancy as a percentage of the total drag gradient is less than 0.5%. The discrete adjoint using the exact Jacobian is thereby taken to be verified for this case.

Next the gradients using the adjoint method with various Jacobian approximations are compared with those of the exact adjoint in Fig. 5.2. TSL and CCA gradients

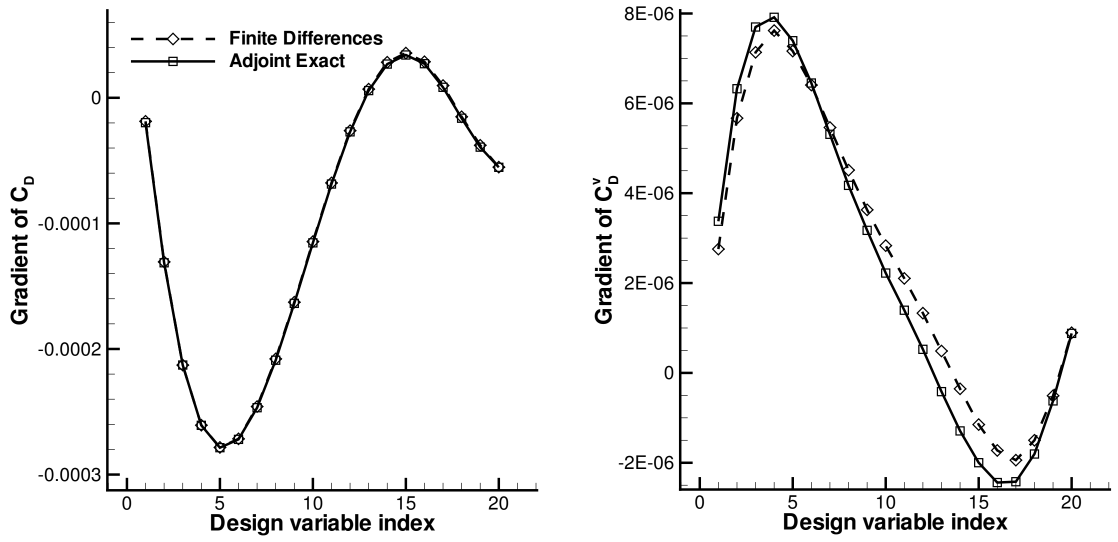


Figure 5.1: Gradients of total and viscous drag obtained using finite differences and the discrete adjoint formulation with an exact Jacobian for the RAE aerofoil. The 20 design variables parameterize the aerofoil camberline from the leading edge to the trailing edge. The trailing edge itself remains fixed.

agree with the exact gradients to within a relative error of 1%, whereby TSL is slightly more accurate than CCA, an expected effect as TSL is a subset of the CCA approximation. The considerable inaccuracy of the FOA gradients might be attributed to the presence of the shock, which would be heavily smeared if first-order fluxes were used in the non-linear calculation. The error in the CEV gradients might be attributed to the importance of turbulence, an effect which is apparently well captured by the ATM model. More detailed explanations might be constructed by comparing the adjoint fields for each approximation, but physically interpreting the adjoint field is a delicate and difficult matter, and rather we move directly to numerical investigations.

Conjugate Gradient Optimization

The conjugate gradient (CG) algorithm, an improvement on the method of steepest descent, uses conjugate gradients rather than the local gradient to determine a direction in which to search for the minimum. Given this search direction, a line-search using repeated evaluations of the cost-function is performed on the resulting one-dimensional subspace. Once the minimum on the line is found, a new search direction is calculated.

If no cost-function improvement is obtained in the CG direction the algorithm is restarted, and the search continues in the gradient direction. This is performed to prevent stalling of the algorithm in the case that the cost-function is not a pure quadratic form, but has the effect of also preventing the accumulation of errors from inaccurate gradients on each iteration. As a result the algorithm is particularly robust, and will only fail if no reduction in the cost function can be found in either the CG

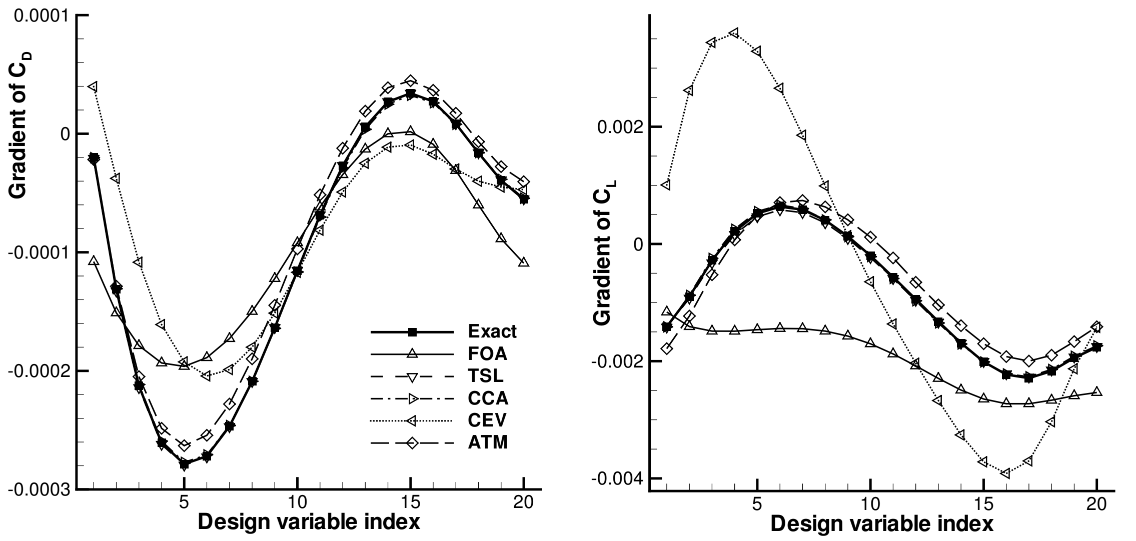


Figure 5.2: Gradients of total drag and total lift for the camberline parameterization of the RAE2822 aerofoil, obtained using the discrete adjoint with the exact Jacobian, as well as with all five Jacobian approximations.

direction or the gradient direction, which could only be the result of a very poor gradient, or a design point close to a local minimum.

This robustness can be seen in the convergence of the RAE case, Figs. 5.3 and 5.4, whereby the force coefficients are plotted against the number of evaluations of the cost-function (i.e. non-linear RANS computations) performed, and therefore approximately represent the cost of the calculation, given that gradient evaluations are relatively cheap and seldom. Gradient evaluations are denoted on the convergence curves by symbols. In convergence plots of α and C_D^v in Fig. 5.4 it is clear that none of the CG methods are fully converged, as the values of α and C_D^v are still changing after 30 iterations, however after this point the change in the cost-function is very small - less than one-tenth of a drag count. As a result full convergence is rarely considered necessary in practical applications, and the partial convergence used here better reflects engineering practice.

The most striking feature of the convergence, seen in the left-hand graph of Fig. 5.3, is that all approximate gradient optimizations converge to approximately the same value of C_D with approximately the same effort. All optimal solutions are within 2.5 drag counts of each other - if FOA is disregarded, 0.5 drag counts - and obtained within 30-40 cost-function evaluations. Apparently large deviations in gradients have little effect on CG for this relatively simple case, although it is the case that the approximations that showed more accurate gradients have better results, TSL and CCA in particular being indistinguishable from the exact gradient optimization.

FOA was the worst performer, at iteration 34 it was unable to find a better solution in the CG direction, and at iteration 35 also in the gradient direction and therefore stopped, at a point far from the optimum found by the other methods, as seen in Fig. 5.4. In order to establish whether this design point was an alternative local

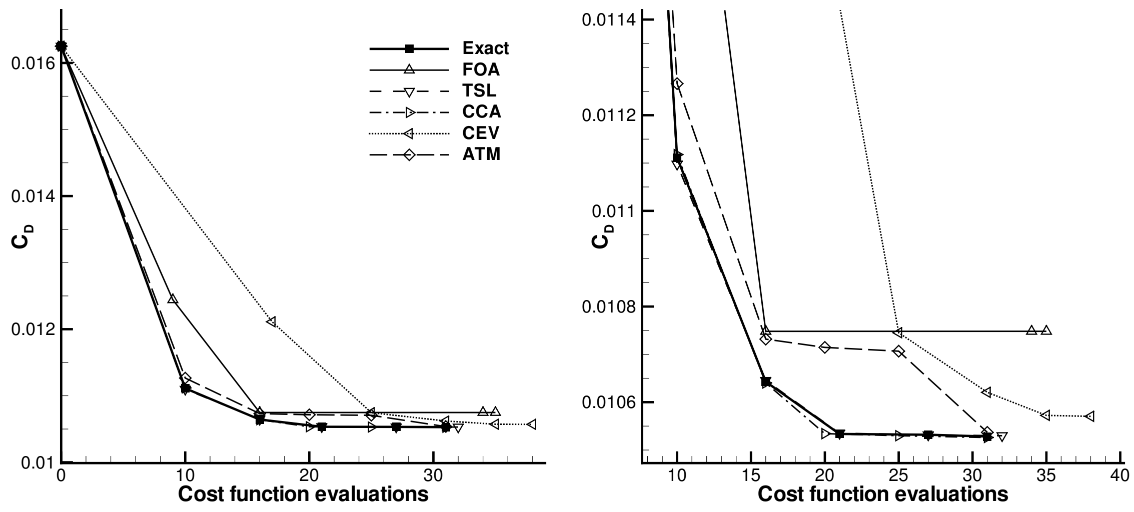


Figure 5.3: Convergence of the conjugate gradient algorithm for drag minimization of the RAE2822 at constant lift, for all discrete adjoint approximations. The convergence is plotted against the number of cost-function evaluations (i.e. non-linear RANS computations) performed, and therefore approximately represents the CPU-time cost of the optimization. Most cost-function evaluations are needed for line searches, whereby symbols denote gradient evaluations. The plot on the right shows details of the plot on the left near the optimum solution.

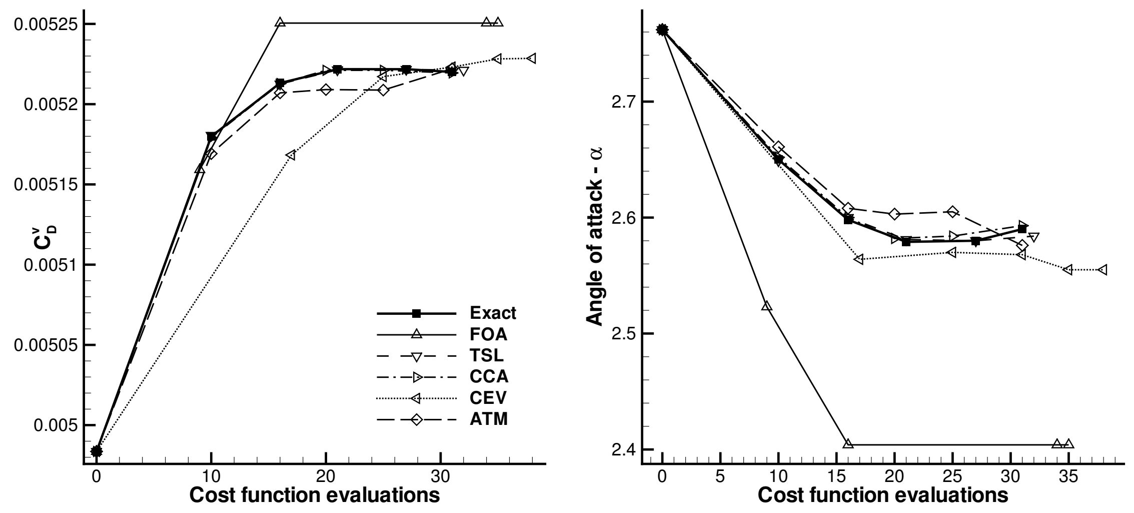


Figure 5.4: Convergence of the viscous drag and angle of attack for the RAE2822 drag reduction optimization of Fig. 5.3. Convergence is shown for each discrete adjoint approximation.

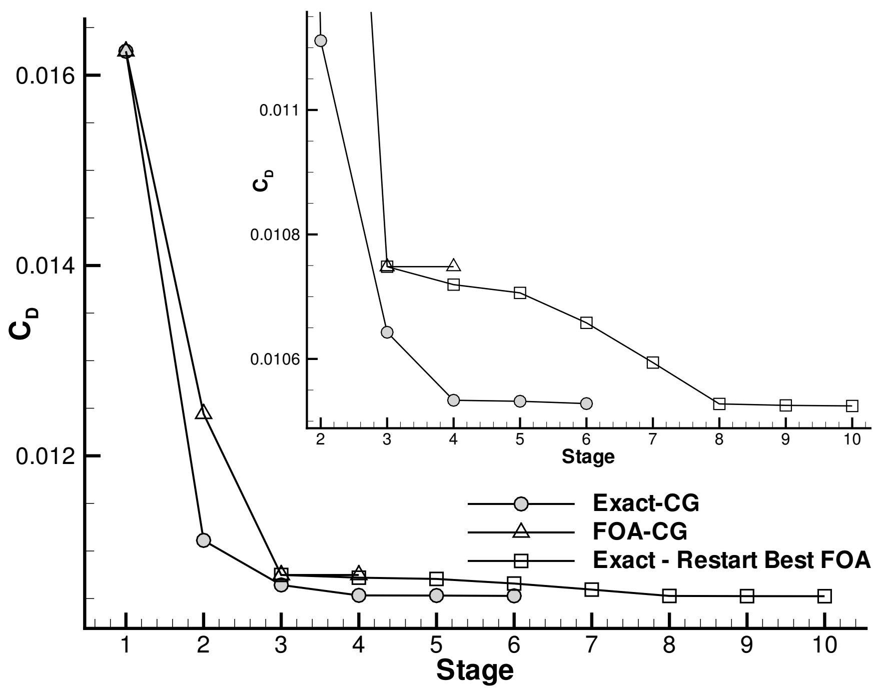


Figure 5.5: Convergence of the optimizations with exact and FOA adjoint gradients for the drag reduction optimization of Fig. 5.3. After the FOA optimization has converged, a restart with the exact adjoint gradients is performed.

minimum, or stalling due to an inaccurate gradient, an optimization was restarted using exact gradients from the iteration at which FOA stalled, the convergence is shown in Fig. 5.5 plotted against number of gradient evaluations. The restarted optimization rapidly finds a better solution, indicating that poor FOA gradients were responsible for the 2.5 drag count deficit.

For completeness the force coefficients of the best design found by each approximate optimization process are given in Table 5.1. Some optimal geometries and pressure distributions are shown in Fig. 5.6, whereby the results of TSL and CCA are indistinguishable from the exact optimization. All approximations removed the shock, as might have been deduced from the drag convergence behaviour, but the resulting pressure distribution for the exact gradient optimization is significantly smoother than that of the poorer approximations.

As described above the constant lift coefficient was enforced explicitly; an alternative method that is not considered here involves restricting the search direction to be in the hyper-plane normal to the gradient of the lift. For sufficiently small updates, the lift then varies in proportion to the square of the magnitude of the update step (if the lift has "wandered too far" after several iterations then an optimization iteration purely for the correction of the lift is performed). This process is of course very sensitive to the accuracy of the gradient, poor gradients implying the need for more correction steps. However the results *directly* reflect the disparities in the gradients themselves, and the performance of the scheme may be roughly judged by considering

	Cost-fn. evals	Gradient evals	C_D (counts)	C_D^v (counts)	C_L (counts)	C_M	α
Baseline	-	-	162.51	49.83	79.99	-0.29327	2.7620
Exact	31	5	105.28	52.20	79.99	-0.29921	2.5899
FOA	35	4	107.48	52.50	79.99	-0.31165	2.4040
TSL	32	5	105.30	52.21	79.99	-0.29956	2.5840
CCA	31	5	105.26	52.19	79.99	-0.29903	2.5929
CEV	38	5	105.70	52.28	80.00	-0.30269	2.5550
ATM	31	5	105.36	52.22	79.99	-0.29995	2.5759

Table 5.1: Results of drag reduction (at constant lift) optimizations of the RAE-2822 aerofoil using the conjugate gradient method. Given are flow coefficients for the best geometry obtained using each gradient approximation, and the number of cost-function and gradient evaluations necessary to achieve that geometry. One drag count corresponds to one-ten thousandth of a drag coefficient unit, and one lift count to one-hundredth of a lift coefficient unit.

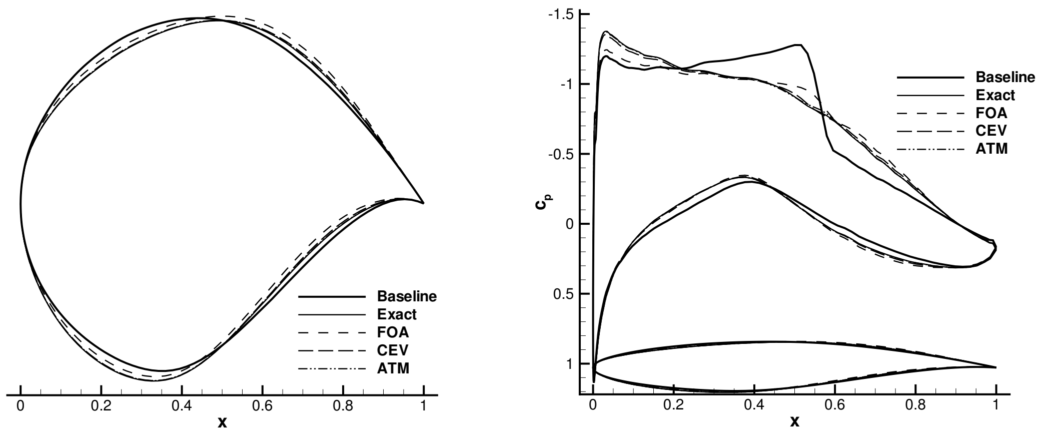


Figure 5.6: Baseline and optimized geometries and pressure distributions for optimizations performed with a variety of approximate discrete adjoint gradients. The results for TSL and CCA optimizations are not shown as they are indistinguishable from the exact optimization result.

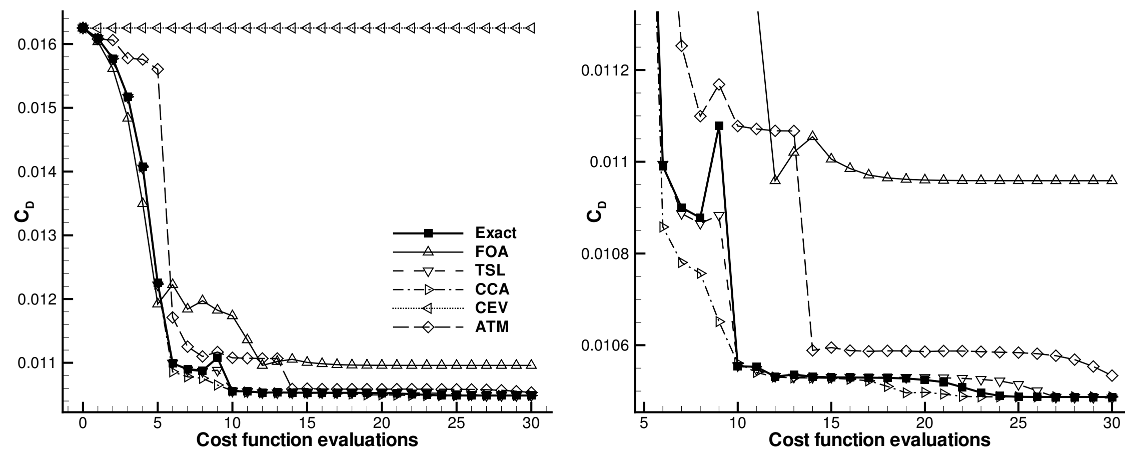


Figure 5.7: Convergence of the Quasi-Newton Trust Region method for drag minimization of the RAE2822 at constant lift. The optimization problem statement is identical to that of Fig. 5.3. Convergence is plotted for the method with various adjoint gradient approximations. As before the plot on the right shows details of the plot on the left near the optimum solution.

the error in the gradient.

Quasi-Newton Trust Region (QNTR) Optimization

The Quasi-Newton Trust Region method attempts to improve on the convergence of the CG method by approximating the cost-function on the design space by a quadratic form (Geiger & Kanzow, 1999). The gradient of the cost-function is computed at every iteration, and based on a BFGS update, an approximation to the Hessian of the cost-function is built. The next design point is chosen as the minimum of this approximation, whereby the minimum must lie within the *trust-region*, effectively a limit on the size of the design step. The size of the trust-region is increased or decreased based upon the accuracy with which the approximation matches the real function, judged using the discrepancy between the previous cost-function evaluation and its corresponding estimate.

In practice this method is less robust than CG, and strongly dependent on the accuracy of the Hessian, which in turn depends upon the gradients at all previous steps. One poor gradient could damage the Hessian approximation, and hence the convergence of the method, significantly. QNTR represents therefore a more demanding test of gradient accuracy than CG.

This method is applied to the same RAE drag reduction case as before and similar optima are achieved; the convergence is shown in Fig. 5.7. With this algorithm the convergence of the exact, TSL and CCA approximations are no longer identical, testifying to the increased sensitivity of the method to the gradient, while being similar enough to have confidence in the use of these gradients in QNTR. The fact that CCA converges slightly faster than the exact method is not significant, but rather noise as a result of the complexity of the system.

The complete lack of convergence of CEV in this case cannot be attributed to the

	Cost-fn. evals	Gradient evals	C_D (counts)	C_D^v (counts)	C_L (counts)	C_M	α
Baseline	-	-	162.51	49.83	79.99	-0.29327	2.7620
Exact	32	32	104.86	51.92	79.99	-0.29142	2.7070
FOA	12	12	109.58	52.51	80.00	-0.31582	2.3499
TSL	32	32	104.84	51.93	79.99	-0.29145	2.7109
CCA	37	37	104.87	51.88	79.99	-0.29038	2.7349
CEV	6	6	162.51	49.83	79.99	-0.29327	2.7620
ATM	36	36	105.28	52.21	79.99	-0.29945	2.5780

Table 5.2: Results of drag reduction (at constant lift) optimizations of the RAE-2822 aerofoil using the Quasi-Newton Trust Region method. Values given are as for Table 5.1. Note that the QNTR algorithm performs a gradient evaluation for every cost-function evaluation.

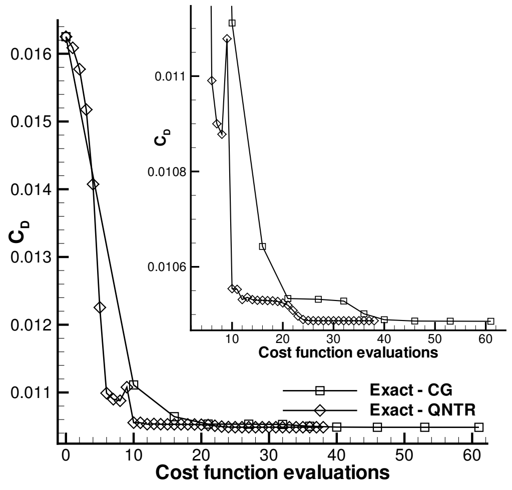
poor initial drag gradient, which is not substantially worse than that of FOA. But upon examining the gradient of the corrected cost function of (5.10) it is seen that while the shape of the gradient is correct, the scaling is completely wrong. This is of no consequence for CG, which uses only directional information from the gradient, but is fatal for QNTR, whose cost-function approximation becomes increasing inaccurate, leading to a reduction in the size of the trust-region on every iteration. Examining the gradient of CEV at each step of the CG method of the previous section, it is apparent that the gradient improves substantially after the first iteration, after which the strength of the shock has been reduced considerably.

The results for the optimal designs using the QNTR method are given in Table 5.2. The best drag coefficients for TSL and CCA show a slight improvement over those of the CG optimization due to the more highly converged state of the system.

Comparison of CG and QNTR

It is interesting to compare the relative performance of CG and QNTR for this adjoint code, which - due to the extremely rapid adjoint solution - makes gradient evaluation considerably cheaper than cost-function evaluation. The CPU time required for full evaluation of the gradient, including (exact) adjoint calculations for C_D and C_L, and deformation of the mesh for each design variable, is approximately 17.5% of the time required for a single non-linear flow solution. Hence the QNTR overhead of a gradient evaluation at every iteration is minimal.

The convergence of CG and QNTR using exact adjoint gradients are shown in Fig. 5.8. Convergence of the drag to within a tenth of a drag count of the optimum occurs for CG and QNTR within 50 and 25 cost function evaluations respectively. However it is apparent from the development of the pitching moment C_M, on the right-hand side of Fig. 5.8, that while the QNTR solution is completely stationary after 30 iterations, CG is still modifying the geometry after 60 iterations. This property of QNTR, while very attractive, is not of immediate practical relevance, as the minimization of drag is the single objective of the optimization. Hence we can say that for this case QNTR delivers the optimal solution in a CPU time equivalent to



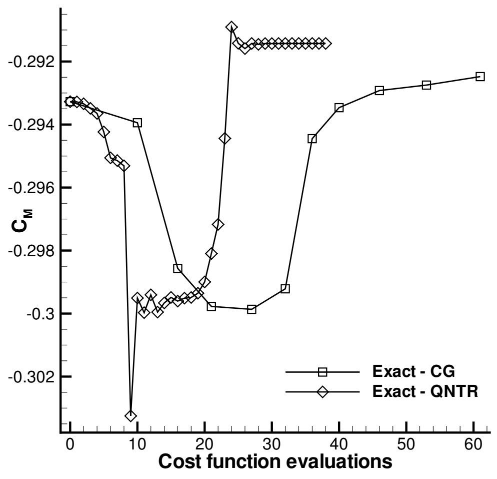


Figure 5.8: Convergence of the drag and pitching moment for the RAE2822 optimization with the CG and QNTR algorithms using gradients from the exact adjoint.

about 50% of that of CG, while being less robust to poor gradients.

5.5.2 Three-Element High-Lift Configuration

Again gradients obtained by the adjoint method are compared with finite difference gradients in Fig. 5.9, whereby the latter are difficult to determine accurately in this case, and convergence of the difference could not be achieved for all design variables. Instead the finite difference gradients are plotted for three distinct step sizes. The agreement with the adjoint is very good, even for C_D^v, and the adjoint is thereby taken to be verified for this case.

As before the adjoint approximations are compared in Fig. 5.10. TSL and CCA are again almost indistinguishable from the exact gradients, CEV agrees well, corroborating the evidence that for subsonic cases its accuracy is good, while FOA and in particular ATM make extremely large systematic errors. The poor performance of ATM here is surprising given its good performance for the RAE aerofoil, but might be explained by the strong influence of detached shear-layers in this case, in which regions the SA and SAE models are most likely to differ - as previously discussed. It seems that neither CEV nor ATM can be considered really reliable adjoint approximations as regards the gradient.

Figure 5.11 shows a typical non-linear solver convergence plot for this case. The time stepping algorithm, an LU-SGS smoothed multigrid iteration (Dwight, 2004), requires about 5000 cycles to achieve a reduction in the residual of the discretization of 6 orders of magnitude, at which point the drag is converged to an accuracy well under a drag count, although changes in the drag are still visible in the right-hand plot of Fig. 5.11. Further convergence of the non-linear solution is desirable, as partially converged solutions are a major source of noise in the optimization process, however CPU time constraints do not permit this.

An optimization using CG is performed for each adjoint gradient approximation, and the convergence histories are plotted in Fig. 5.12. Again convergence is obtained

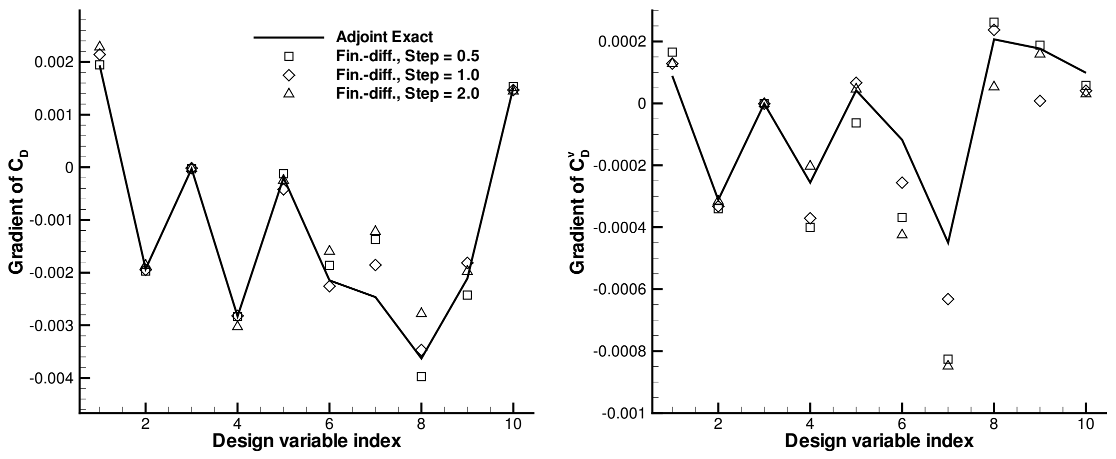


Figure 5.9: Gradients of total and viscous drag obtained using finite differences and the discrete adjoint formulation with an exact Jacobian for the high-lift configuration. The design variables parameterize the geometry of the flap. The first four determine the horizontal and vertical position, the angle, and the nose sharpness respectively. The remaining six modify the shape of the forward-upper surface of the flap. Finite difference gradients are shown for a variety of step-sizes.

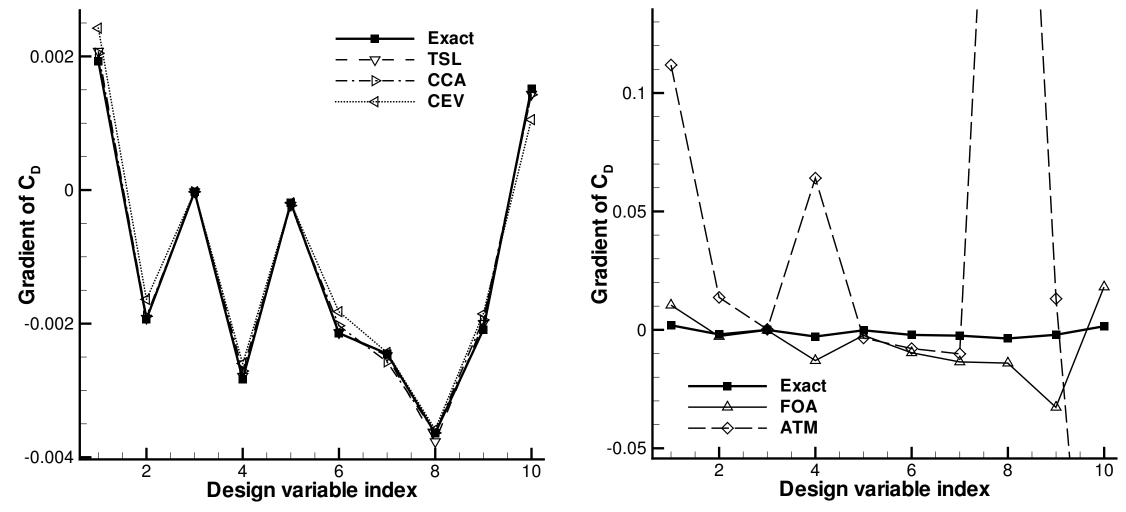


Figure 5.10: Gradients of total drag for the parameterization of the high-lift configuration, calculated with the exact adjoint as well as the various adjoint approximations.

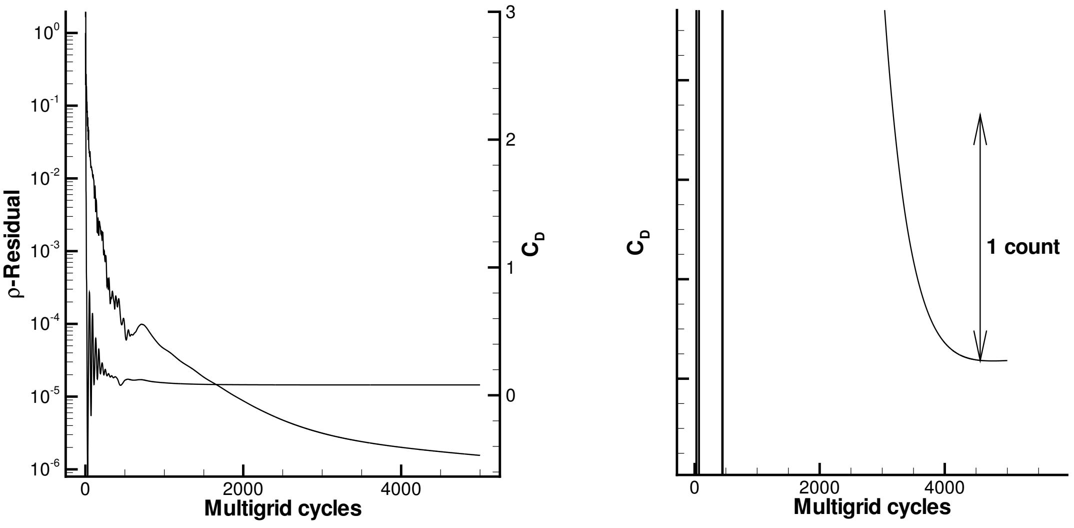


Figure 5.11: Convergence of residual and drag for the non-linear RANS solver for the high-lift configuration.

in all cases, testifying to the robustness of CG. Convergence and optimal solution are very similar for the three approximations that showed accurate gradients: TSL, CCA and CEV. On the other hand FOA and even ATM achieve optima within one drag count of the best solution found; the total drag reduction over the baseline geometry amounts to 66 counts. Whether this discrepancy is significant depends upon the significance of one drag count to the engineering problem, bearing in mind also the limited accuracy of the CFD model.

The details of each of the optimal solutions are given in Table 5.3. The optimal geometry and pressure distribution achieved with the exact gradients are shown in Fig. 5.13. The means by which the drag may be reduced are not as clear as for the RAE case, however an important aspect is certainly the prevention of separation and maintenance of high speed flow over the upper side of the main element near the trailing edge. This may be achieved by, for example, moving the flap downstream until the suction peak at the flap stagnation point is at, or downstream of, the main element trailing-edge. In Fig. 5.13 it can be seen that exactly this effect has been obtained. If on the other hand a constraint is placed on the amount by which the flap may be shifted, then it is sometimes the case that the optimization finds an optimal geometry with a bump on the upper surface, producing a second suction peak in the pressure distribution behind the main element trailing edge. This is undesirable as such a profile would be expected to have a poor maximum lift, maximum lift being vital for a feasible high-lift system.

The optimal geometries and pressure distributions obtained with exact gradients, FOA and ATM are shown in Fig. 5.14, the remaining approximate results being identical to the exact gradient result. The CG method was ultimately able to perform effective optimizations with all the gradient approximations tested here.

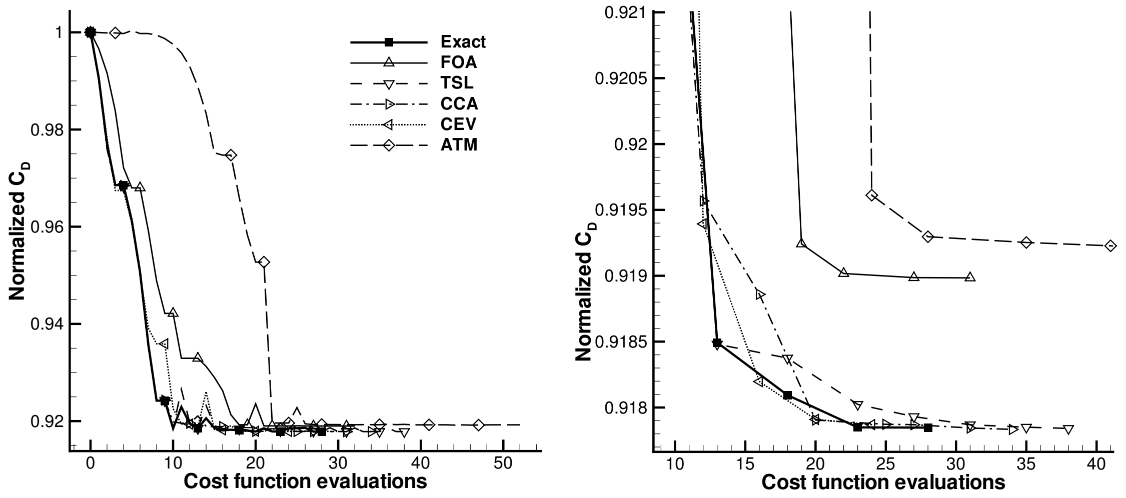


Figure 5.12: Convergence of the conjugate gradient algorithm for drag reduction at constant lift for the high-lift configuration, shown for optimizations based on all adjoint gradient approximations. As before convergence is plotted against non-liner RANS solver calls, thereby representing the approximate cost of the optimization. Symbols represent gradient evaluations, and the drag is normalized against the drag of the baseline geometry.

	Cost-fn. evals	Gradient evals	ΔC_D (counts)	ΔC_D^v (counts)	ΔC_M	$\Delta\alpha$
Baseline	-	-	-	-	-	-
Exact	28	6	-66.68	-5.80	-0.0432	-0.587
FOA	31	7	-65.76	-6.16	-0.0445	-0.518
TSL	38	9	-66.68	-5.85	-0.0433	-0.582
CCA	34	8	-66.69	-5.80	-0.0432	-0.587
CEV	23	6	-66.66	-5.80	-0.0434	-0.584
ATM	44	9	-65.59	-5.94	-0.0452	-0.509

Table 5.3: Results of drag reduction (at constant lift) optimizations of the high-lift configuration using the conjugate gradient method and a variety of adjoint gradient approximations. Values given are changes in the coefficients, the coefficients themselves are as in Table 5.1.

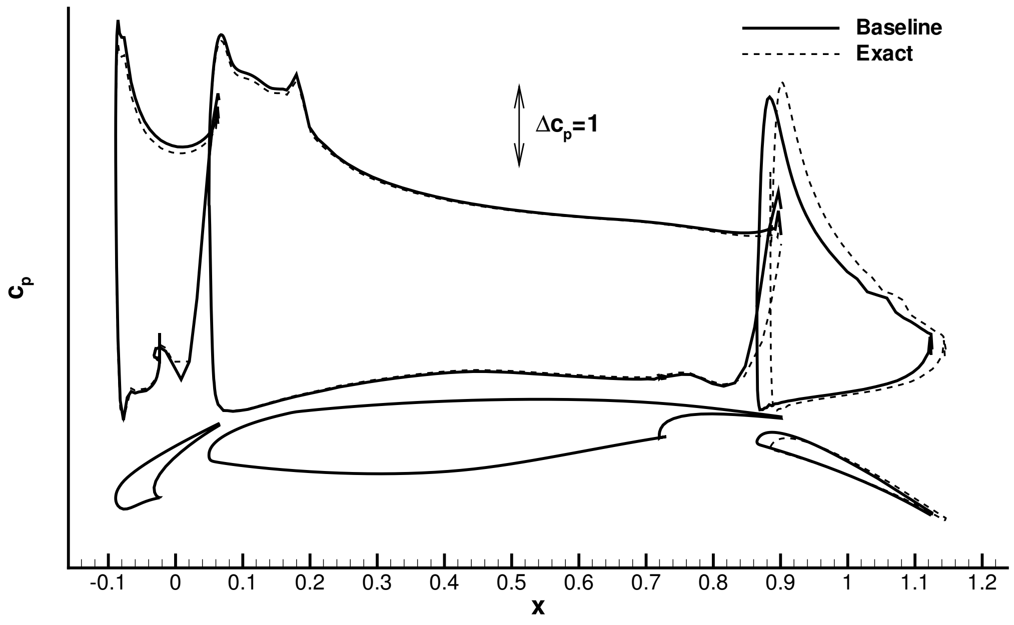


Figure 5.13: Baseline and optimized geometries and pressure distributions for drag minimization of the high-lift configuration with constant lift. The baseline angle of attack is 9.61°, whereas the optimized value is 9.02°.

5.6 Conclusions

A method for construction and solution of the exact adjoint problem has been described which permits convergence within 5% of the CPU time for the non-linear problem, and therefore typical full gradient calculations (including two adjoint solutions) within 20% of the time for a single non-linear problem solution. Memory requirements are however 6-7 times higher than the standard non-linear solver due to the storage of the Jacobian, effectively limiting the method to 2d cases.

In an effort to reduce the memory requirements while maintaining the efficiency of the method, several approximations to the discrete adjoint have been proposed and studied with respect to optimizations on two 2d cases. Two of the approximations studied, TSL and CCA, showed consistently minimal variance from the exact gradients, and as such may be used with confidence as substitutes for the exact adjoint method. On the other hand it has been seen that approximating the adjoint of the turbulence model, either with constant eddy-viscosity (CEV) or the adjoint of a very similar model (ATM), leads to gradients that are good in some cases, but exceptionally poor in other cases. In particular CEV produced poor gradients for the RAE baseline case with a strong shock but excellent gradients in the high-lift case; for ATM the situation was reversed. These results suggest that construction of reliable adjoint method requires exact consideration of the particular turbulence model used.

Optimizations were performed with the conjugate gradient (CG) algorithm, which converged reasonably well for all cases and with all gradients, no matter how poor.

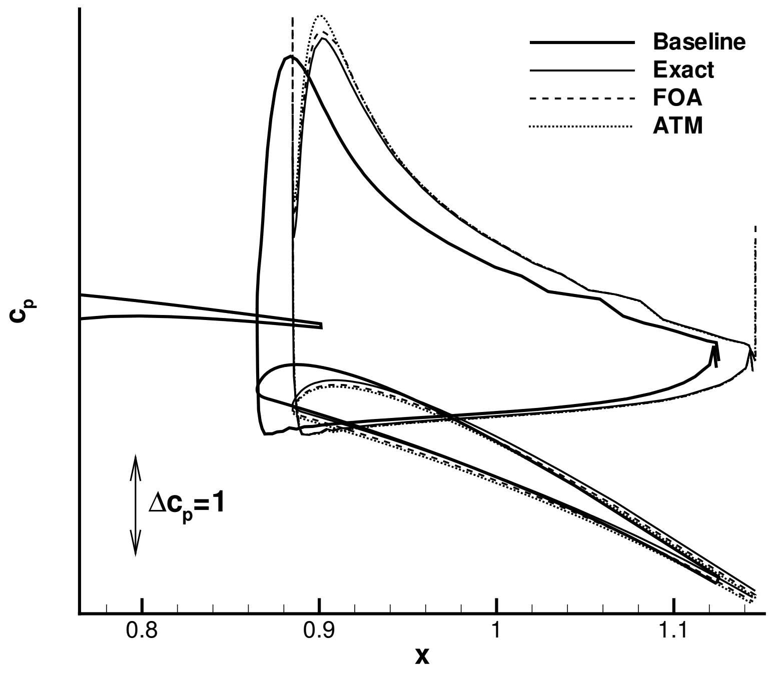


Figure 5.14: Baseline and optimized geometries and pressure distributions for the flap of the high-lift configuration, from optimizations performed with a variety of approximate discrete adjoint gradients. The results for TSL, CCA and CEV optimizations are not shown as they are indistinguishable from the exact optimization result.

Optimal results obtained with variously inaccurate gradients resulted in optimal solutions differing by no more than 2.5 drag counts for the RAE case and 1 drag count for the high-lift case. A consequence is that an extremely poor (and correspondingly cheap) adjoint approximation, such as FOA, could be useful in situations where high accuracy is not required.

It has also been shown that the possibility for rapid gradient evaluation allows efficient use of the Quasi-Newton Trust-Region method, which can reduce the overall optimization time by 50% in comparison to CG, whereby the method is much more sensitive to poor gradients.

Based on the results of this study an adjoint solver using the CCA approximation will be further developed, and a reduction in memory requirements to about three times those of the non-linear solver are expected.

Chapter 6

Conclusions

A complete unstructured grid finite volume method for the RANS equations was presented in Chapter 2, including all details of the JST convective flux discretization, boundary conditions and turbulence models, as implemented in the flow solver the DLR-*TAU*-Code. In addition the exact and some approximations of the Jacobian of the complete spatial discretization were derived and implemented, of which selected parts were described. In particular the full Jacobian of the JST convective flux was derived, and an approximation is proposed (namely the regarding of the coefficient of dissipation as constant) which simplifies the expression for the Jacobian significantly. It was shown that the terms in the derivative neglected through this approximation were of higher order in the grid-spacing Δx than the remaining terms, thereby providing an original justification for this approximation. Further, a method for the treatment in the Jacobian of strongly implemented boundary conditions was detailed.

The implicit solution of the discretized equations was considered next in Chapter 3. It was shown, based on the practical example of a novel formulation of the LU-SGS scheme with FAS multigrid, that it is possible to construct an efficient, approximately factored, implicit method that has identical memory requirements to the standard explicit Runge-Kutta scheme. The method is also cheaper in terms of CPU time per iteration than said scheme. Even so it is able to operate with CFL numbers of the order of 1000 for Euler flow problems, and of the order of 10 for complex 3d high-Reynolds number Navier-Stokes problems, for which it converges in 50%–90% of the iteration count of RK. Finally the scheme is at least as stable in a wide range of situations as RK, and can therefore be used as a true slot-in replacement for that scheme in all situations. The method has been implemented in the *TAU*-Code and at time of writing is widely used within the European aerospace industry.

While the final assessment of the performance of LU-SGS was based on the numerical investigation of the method applied to practical engineering test-cases, the properties of the scheme were also examined in the context of diagonal dominance of the implicit matrix, and Fourier analysis. It has been shown by numerical comparison with related schemes designed to be diagonally dominant, that diagonal dominance is not desirable in terms of non-linear convergence. Fourier analysis of the scheme was unsuccessful in that results fundamentally disagreed with numerical observations, the cause of which was traced back to the oversimplified model of the implicit method necessitated by the linear nature of the model equation under consideration.

Nonetheless these investigations were instrumental in formulating and understanding the LU-SGS scheme. Last, but not least, the time-accuracy of the scheme was examined.

By relaxing the constraint that the memory requirements of the scheme should not be significantly greater than those of Runge-Kutta, explicit storage of the Jacobian, and the subsequent solution via Krylov methods became possible, investigated in Chapter 4. An exact Newton method based on the explicitly stored Jacobian of a second-order central discretization was examined. Solution of the linear system at each step using the existing non-linear time stepping method was investigated. It was shown both theoretically, and on the basis of a turbulent test case, that the resulting linear system convergence is identical to the asymptotic convergence of the non-linear iteration using the same method. However this approach was ultimately found to be inappropriate as part of a Newton method due to its instability for Jacobians based on partially-converged non-linear solutions.

Attention was therefore turned to Krylov methods, which provided convergence in all cases given sufficient preconditioning. However, it was found that the level of preconditioning needed was often excessive. For a simple 2d turbulent test case an ILU(4) preconditioner and a GMRES(50) method was required. The resulting algorithm required about 15 times more memory than the standard solver, which was considered an unacceptable overhead.

Attention was therefore turned to Jacobians based on first-order fluxes, and two methods were proposed, one using a Gauss-Seidel inner iteration (FOGSI), and one using an ILU preconditioned Krylov method (FOKI). In FOKI the turbulence equation iteration was completely separated from the mean flow iteration, and this was seen to save memory, as well as improve the convergence of the method. FOKI was shown to be 4-5 times faster than LU-SGS for the cases considered, while the resulting code required about 5 times more memory. The method was seen to be a very promising candidate for further development, in particular in terms of extension to complex three-dimensional configurations.

These investigations, covering a wide range of Jacobian approximations and linear solvers, provided insights into the relationship between the two critical elements of an implicit scheme. In particular the importance of proper consideration of boundary conditions, even in poor Jacobians, and aspects of the trade-off between linear and non-linear stiffness were demonstrated.

The effort required to evaluate the Jacobian for the exact Newton method of the previous chapter, which amounts to a linearization of the entire discrete flow solver, suggested looking for alternative applications. The adjoint method, in Chapter 5, was seen to be such an application which allows the rapid evaluation of the derivatives of a flow quantity, such as drag, with respect to some design variables which parameterize the aerodynamic shape. These derivatives are then used for gradient-based shape optimization.

The adjoint method suffered under similar problems to the exact Newton method however: considerable storage was required and the resulting linear system was difficult to solve. Using a simplified Jacobian affected the accuracy of the resulting gradients, unlike in the Newton method where it only affected the transient convergence behaviour. The sensitivity of the gradients to approximations of the Jacobian was therefore examined on the basis of two optimization test cases. It was seen that considerable simplifications were possible, without significant modifications to the optimum, and these differences were quantified. Thereby greater understanding of the sensitivity of the optimization to inaccurate gradients, and in particular to approximations of the adjoint equations was won.

6.1 Further Work

This thesis does not represent the last word on the methods discussed; rather it is part of a continuing investigation, especially in the case of the adjoint method, and it is hoped that the results herein will inform and guide future work.

While the LU-SGS method is more-or-less complete, the FOKI approach has so far only been shown for two-dimensional test cases. If it to be applied routinely to practical cases it must be made at least as robust as LU-SGS and Runge-Kutta for complex geometries, poor grids, and flows with separation. Additionally the area of start-up methods should be investigated, including consideration of when it is possible to start the full method and expect convergence. For these questions more experience with the approach is needed in practice.

The principal defect of the discrete adjoint algorithm as described here is its large memory requirements in relation to the non-linear solver, which are a consequence of the explicit storage of the Jacobian and the use of ILU preconditioned GMRES. It is expected that in the near future an accompanying adjoint solver will be demanded of any industrially applied computational aerodynamics code, not only for the purposes of optimization, but also for error estimation and goal-based mesh adaptation. In this case the memory performance of the adjoint solver must be improved significantly if it is to enjoy routine use.

The path to this goal is highlighted by the investigations of Chapter 5, which show that given some approximations which have little effect on the adjoint solution, the construction of the adjoint residual can be simplified and accelerated considerably, while the associated memory requirements are also reduced, see especially Section 5.4.3. Future work will concern the implementation of this formulation of the residual in an efficient manner.

Apart from its high memory requirements, the ILU preconditioner may not longer be used, as the Jacobian is not available explicitly. Instead solution using existing time stepping methods for the non-linear equation will be applied, as was done for the linear problem in Section 4.3.1. This approach had two nice properties: guaranteed convergence of the linear system given convergence of the non-linear system, as well as identical asymptotic convergence rate. If these are to be preserved for the adjoint equation, the time stepping scheme must also be “adjointed” (Giles, 2001). This will be done, and it is expected that the resulting system will have similar stability and convergence properties and similar memory requirements to the non-linear solver.

Given the adjoint approach, the operation that now dominates the cost of an gradient evaluation is the deformation of the grid for each design variable, in order to find the mesh sensitivities. It was shown how this cost may be eliminated in Section 5.3.3, and work is progressing to implement this approach, whereby the most effort needs to be expended in the linearization of the mesh deformation algorithm, and the evaluation of the terms $\partial R/\partial X$ and $\partial I/\partial X$ by hand.

With these tools complete the gradient-based optimization of a three-dimensional wing-body configuration with a mesh of several million points and a parameterization of $100 - 1000$ design variables should be possible in well under one day's computing time on a small PC cluster, and this specific calculation is the immediate goal.

Appendix A

Change of Variables

A variety of variables are useful for studying the Euler- and Navier-Stokes equations; this Appendix is intended as a reference to the flux-Jacobians in some of the more common cases. Consider first the *conservative variables*

$$W_c = (\rho,\, \rho u,\, \rho v,\, \rho w,\, \rho E)^T\,, \tag{A.1}$$

so called because they are quantities conserved by the governing equations. The convective flux across a face n in conservative variables is

$$f^c_c \cdot n = \begin{pmatrix} \rho V \\ \rho u V + p n_x \\ \rho v V + p n_y \\ \rho w V + p n_z \\ \rho V H \end{pmatrix}, \tag{A.2}$$

the subscript of f^c denoting here the variable set used. The derivatives of these fluxes with respect to the convervative variable vector W_c are

$$\frac{\partial f^c_c \cdot n}{\partial W_c} = \begin{pmatrix} 0 & n_x & n_y & n_z & 0 \\ n_x\phi - uV & V - a_3 n_x u & n_y u - a_2 n_x v & n_z u - a_2 n_x w & a_2 n_x \\ n_y\phi - vV & n_x v - a_2 n_y u & V - a_3 n_y v & n_z v - a_2 n_y w & a_2 n_y \\ n_z\phi - wV & n_x w - a_2 n_z u & n_y w - a_2 n_z v & V - a_3 n_z w & a_2 n_z \\ V(\phi - a_1) & n_x a_1 - a_2 u V & n_y a_1 - a_2 v V & n_z a_1 - a_2 w V & \gamma V \end{pmatrix},$$

where

$$a_1 = \gamma E - \phi, \quad a_2 = \gamma - 1, \quad a_3 = \gamma - 2,$$
$$V = (u, v, w)\cdot n, \quad \phi = \frac{1}{2}(\gamma - 1)(u^2 + v^2 + w^2).$$

A.1 Primitive Variables

After the conservative variables the next most commonly used set are some form of *primitive variables*, which for the purposes of this thesis have been chosen as

$$W_p = (p,\, u,\, v,\, w,\, T)^T\,, \tag{A.3}$$

purely to provide a convenient set of variables in which to perform algebraic manipulations. The Jacobian of the exact convective fluxes in primitive-primitive variables (that is the primitive fluxes differentiated with respect to the primitive variables) is

$$\frac{\partial f_p^c \cdot n}{\partial W_p} = \begin{pmatrix} V & \rho a^2 n_x & \rho a^2 n_y & \rho a^2 n_z & 0 \\ \frac{n_x}{\rho} & V & 0 & 0 & 0 \\ \frac{n_y}{\rho} & 0 & V & 0 & 0 \\ \frac{n_z}{\rho} & 0 & 0 & V & 0 \\ 0 & \frac{\gamma-1}{\gamma} n_x a^2 & \frac{\gamma-1}{\gamma} n_y a^2 & \frac{\gamma-1}{\gamma} n_z a^2 & V \end{pmatrix}. \tag{A.4}$$

The change of basis matrices between conservative and primitive variables are

$$\frac{\partial W_c}{\partial W_p} = \begin{pmatrix} \frac{\rho}{p} & 0 & 0 & 0 & -\frac{\rho}{T} \\ \frac{\rho u}{p} & \rho & 0 & 0 & -\frac{\rho u}{T} \\ \frac{\rho v}{p} & 0 & \rho & 0 & -\frac{\rho v}{T} \\ \frac{\rho w}{p} & 0 & 0 & \rho & -\frac{\rho w}{T} \\ \frac{\rho E}{p} & \rho u & \rho v & \rho w & -\frac{\rho q^2}{2T} \end{pmatrix},$$

$$\frac{\partial W_p}{\partial W_c} = \begin{pmatrix} \phi & -(\gamma-1)u & -(\gamma-1)v & -(\gamma-1)w & (\gamma-1) \\ -\frac{u}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{v}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{w}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 \\ \frac{(\gamma-1)(q^2-E)}{\rho} & -\frac{(\gamma-1)u}{\rho} & -\frac{(\gamma-1)v}{\rho} & -\frac{(\gamma-1)w}{\rho} & \frac{(\gamma-1)}{\rho} \end{pmatrix}.$$

where all flow quantities are as in Chapter 2.

There now follows descriptions of several more useful vectors of variables. For each set the convective Jacobians are given, along with change of basis matrices to and from the conservative and primitive variables.

A.2 Alternative Primitive Variables

What are here called the *alternative primitive variables*, which are also often used as convenient variables for hand calculations, are

$$W_{p'} = (\rho,\, u,\, v,\, w,\, p)\,, \tag{A.5}$$

so that in alternative primitive-alternative primitive variables the convective flux Jacobian is

$$\frac{\partial f_{p'}^c \cdot n}{\partial W_{p'}} = \begin{pmatrix} V & n_x\rho & n_y\rho & n_z\rho & 0 \\ 0 & V & 0 & 0 & \frac{n_x}{\rho} \\ 0 & 0 & V & 0 & \frac{n_y}{\rho} \\ 0 & 0 & 0 & V & \frac{n_z}{\rho} \\ 0 & n_x\rho a^2 & n_y\rho a^2 & n_z\rho a^2 & V \end{pmatrix}. \tag{A.6}$$

The change of basis matrices to and from conservative variables are

$$\frac{\partial W_c}{\partial W_{p'}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ u & \rho & 0 & 0 & 0 \\ v & 0 & \rho & 0 & 0 \\ w & 0 & 0 & \rho & 0 \\ \frac{1}{2}q^2 & \rho u & \rho v & \rho w & \frac{1}{\gamma-1} \end{pmatrix}, \tag{A.7}$$

$$\frac{\partial W_{p'}}{\partial W_c} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{u}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{v}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{w}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 \\ \phi & -(\gamma-1)u & -(\gamma-1)v & -(\gamma-1)w & \gamma-1 \end{pmatrix}, \tag{A.8}$$

and the corresponding matrices for primitive variables are particularly simple, being

$$\frac{\partial W_p}{\partial W_{p'}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -\frac{T}{\rho} & 0 & 0 & 0 & \frac{1}{\rho} \end{pmatrix}, \tag{A.9}$$

$$\frac{\partial W_{p'}}{\partial W_p} = \begin{pmatrix} \frac{1}{T} & 0 & 0 & 0 & -\frac{\rho}{T} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{A.10}$$

A.3 Entropy Variables

Particularly useful in considerations of low-speed preconditioning for the Euler equations (Turkel *et al.*, 1994) are the *entropy variables*:

$$W_e = (p,\, u,\, v,\, w,\, S)^T\,, \tag{A.11}$$

where S is the entropy, so that $\mathrm{d}S = \mathrm{d}p - a^2\mathrm{d}\rho$ in differential notation. The convective flux Jacobian in entropy-entropy variables is particularly sparse

$$\frac{\partial f_e^c \cdot n}{\partial W_e} = \begin{pmatrix} V & \rho a^2 n_x & \rho a^2 n_y & \rho a^2 n_z & 0 \\ \frac{n_x}{\rho} & V & 0 & 0 & 0 \\ \frac{n_y}{\rho} & 0 & V & 0 & 0 \\ \frac{n_z}{\rho} & 0 & 0 & V & 0 \\ 0 & 0 & 0 & 0 & V \end{pmatrix}, \tag{A.12}$$

as are the change of basis matrices to and from primitive variables. The change of basis matrices between entropy variables and conservative variables are

$$\frac{\partial W_c}{\partial W_e} = \begin{pmatrix} \frac{1}{a^2} & 0 & 0 & 0 & -\frac{1}{a^2} \\ \frac{u}{a^2} & \rho & 0 & 0 & -\frac{u}{a^2} \\ \frac{v}{a^2} & 0 & \rho & 0 & -\frac{v}{a^2} \\ \frac{w}{a^2} & 0 & 0 & \rho & -\frac{w}{a^2} \\ \frac{1}{2}M^2 + \frac{1}{\gamma} & \rho u & \rho v & \rho w & -\frac{1}{2}M^2 \end{pmatrix},$$

$$\frac{\partial W_e}{\partial W_c} = \begin{pmatrix} \frac{1}{2}\gamma q^2 & -\gamma u & -\gamma v & -\gamma w & \gamma \\ -\frac{u}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{v}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{w}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 \\ \frac{1}{2}q^2\gamma - a^2 & -\gamma u & -\gamma v & -\gamma w & \gamma \end{pmatrix},$$

and those to and from primitive variables are

$$\frac{\partial W_p}{\partial W_e} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{\gamma-1}{\gamma\rho} & 0 & 0 & 0 & \frac{1}{\gamma\rho} \end{pmatrix},$$

$$\frac{\partial W_e}{\partial W_p} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1-\gamma & 0 & 0 & 0 & \gamma\rho \end{pmatrix},$$

where

$$q^2 = u^2 + v^2 + w^2, \quad \phi = \tfrac{1}{2}(\gamma - 1)\, q^2, \quad M^2 = \frac{q^2}{a^2}.$$

A.4 Symmetrizing Variables

Symmetrizing variables, in the context of the equations of gas-dynamics, are variables for which the convective flux-Jacobian is symmetric. They are by no means unique, but a commonly used set, in differential form, is

$$W_s = \left(\frac{\mathrm{d}p}{\rho a}, \mathrm{d}u, \mathrm{d}v, \mathrm{d}w, \mathrm{d}S \right)^T, \tag{A.13}$$

where again S is the entropy and $\mathrm{d}S = \mathrm{d}p - a^2\mathrm{d}\rho$. The flux-Jacobian is then

$$\frac{\partial f_s^c \cdot n}{\partial W_s} = \begin{pmatrix} V & an_x & an_y & an_z & 0 \\ an_x & V & 0 & 0 & 0 \\ an_y & 0 & V & 0 & 0 \\ an_z & 0 & 0 & V & 0 \\ 0 & 0 & 0 & 0 & V \end{pmatrix}, \tag{A.14}$$

which form exposes the wave structure in solutions of the gas-dynamic equations. The change of basis matrices of symmetrizing variables to and from conservative variables are

$$\frac{\partial W_c}{\partial W_s} = \begin{pmatrix} \frac{\rho}{a} & 0 & 0 & 0 & -\frac{1}{a^2} \\ \frac{\rho u}{a} & \rho & 0 & 0 & -\frac{u}{a^2} \\ \frac{\rho v}{a} & 0 & \rho & 0 & -\frac{v}{a^2} \\ \frac{\rho w}{a} & 0 & 0 & \rho & -\frac{w}{a^2} \\ \frac{\rho H}{a} & \rho u & \rho v & \rho w & -\frac{q^2}{2a^2} \end{pmatrix}, \tag{A.15}$$

$$\frac{\partial W_s}{\partial W_c} = \begin{pmatrix} (\gamma-1)\frac{q^2}{2\rho a} & -(\gamma-1)\frac{u}{\rho a} & -(\gamma-1)\frac{v}{\rho a} & -(\gamma-1)\frac{w}{\rho a} & \frac{\gamma-1}{\rho a} \\ -\frac{u}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{v}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{w}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 \\ \phi - a^2 & -(\gamma-1)u & -(\gamma-1)v & -(\gamma-1)w & \gamma-1 \end{pmatrix}. \tag{A.16}$$

The change of basis matrices to and from primitive variables are

$$\frac{\partial W_p}{\partial W_s} = \begin{pmatrix} \frac{1}{\rho a} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -(\gamma-1) & 0 & 0 & 0 & \gamma\rho \end{pmatrix}, \tag{A.17}$$

$$\frac{\partial W_s}{\partial W_p} = \begin{pmatrix} \frac{1}{\rho a} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -(\gamma-1) & 0 & 0 & 0 & \gamma\rho \end{pmatrix}. \tag{A.18}$$

A.5 Parabolic Symmetrizing Variables

Parabolic symmetrizing variables in the context of gas-dynamics are variables which simultaneously symmetrize the convective and viscous flux Jacobians. The set given here is from (Abarbanel & Gottlieb, 1981):

$$W_{s'} = \left(\frac{a \mathrm{d}\rho}{\sqrt{\gamma}\rho}, \mathrm{d}u, \mathrm{d}v, \mathrm{d}w, \frac{a \mathrm{d}T}{\sqrt{\gamma(\gamma-1)T}} \right)^T . \tag{A.19}$$

The convective Jacobian in these variables is then

$$\frac{\partial f^c_{s'} \cdot n}{\partial W_{s'}} = \begin{pmatrix} V & \frac{a}{\sqrt{\gamma}} n_x & \frac{a}{\sqrt{\gamma}} n_y & \frac{a}{\sqrt{\gamma}} n_z & 0 \\ \frac{a}{\sqrt{\gamma}} n_x & V & 0 & 0 & \sqrt{\frac{\gamma-1}{\gamma}} a n_x \\ \frac{a}{\sqrt{\gamma}} n_y & 0 & V & 0 & \sqrt{\frac{\gamma-1}{\gamma}} a n_y \\ \frac{a}{\sqrt{\gamma}} n_z & 0 & 0 & V & \sqrt{\frac{\gamma-1}{\gamma}} a n_z \\ 0 & \sqrt{\frac{\gamma-1}{\gamma}} a n_x & \sqrt{\frac{\gamma-1}{\gamma}} a n_y & \sqrt{\frac{\gamma-1}{\gamma}} a n_z & V \end{pmatrix} . \tag{A.20}$$

The transformation between the conservative and parabolic symmetrizing variables is accomplished with the matrices

$$\frac{\partial W_c}{\partial W_{s'}} = \begin{pmatrix} \frac{\sqrt{\gamma}\rho}{a} & 0 & 0 & 0 & 0 \\ \frac{\sqrt{\gamma}\rho u}{a} & \rho & 0 & 0 & 0 \\ \frac{\sqrt{\gamma}\rho v}{a} & 0 & \rho & 0 & 0 \\ \frac{\sqrt{\gamma}\rho w}{a} & 0 & 0 & \rho & 0 \\ \frac{\sqrt{\gamma}\rho E}{a} & \rho u & \rho v & \rho w & \sqrt{\frac{\gamma}{(\gamma-1)}} \frac{p}{a} \end{pmatrix} , \tag{A.21}$$

$$\frac{\partial W_{s'}}{\partial W_c} = \begin{pmatrix} \frac{a}{\sqrt{\gamma}\rho} & 0 & 0 & 0 & 0 \\ -\frac{u}{\rho} & \rho^{-1} & 0 & 0 & 0 \\ -\frac{v}{\rho} & 0 & \rho^{-1} & 0 & 0 \\ -\frac{w}{\rho} & 0 & 0 & \rho^{-1} & 0 \\ \left(q^2 - E\right) a - \frac{1}{p\sqrt{\frac{\gamma}{(\gamma-1)}}} & -\frac{ua}{p\sqrt{\frac{\gamma}{(\gamma-1)}}} & -\frac{va}{p\sqrt{\frac{\gamma}{(\gamma-1)}}} & -\frac{wa}{p\sqrt{\frac{\gamma}{(\gamma-1)}}} & \frac{a}{p\sqrt{\frac{\gamma}{(\gamma-1)}}} \end{pmatrix} , \tag{A.22}$$

and the corresponding matrices for primitive variables are

$$\frac{\partial W_p}{\partial W_{s'}} = \begin{pmatrix} \frac{\sqrt{\gamma}p}{a} & 0 & 0 & 0 & \frac{\sqrt{\gamma(\gamma-1)}p^2}{\rho T a} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{\sqrt{\gamma(\gamma-1)}p}{\rho a} \end{pmatrix} , \tag{A.23}$$

$$\frac{\partial W_{s'}}{\partial W_p} = \begin{pmatrix} \frac{a}{\sqrt{\gamma}p} & 0 & 0 & 0 & -\frac{a}{\sqrt{\gamma}T} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{\rho a}{\sqrt{\gamma(\gamma-1)}p} \end{pmatrix}. \tag{A.24}$$

References

Abarbanel, S., & Gottlieb, D. 1981. Optimal Time Splitting for Two- and Three-Dimensional Navier-Stokes Equations with Mixed Derivatives. *Journal of Computational Physics*, **41**, 1–33.

Badcock, K., Woodgate, M., Cantariti, F., & Richards, B. 1999. Solution of the Unsteady Euler Equations in Three Dimensions Using a Fully Unfactored Method. *Glasgow University, Department of Aerospace Engineering, Technical Report 9909.*

Balay, S., Gropp, W.D., Curfman-McInnes, L., & Smith, B.F. 1997. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. *Pages 163–202 of:* Arge, E., Bruaset, A. M., & Langtangen, H. P. (eds), *Modern Software Tools in Scientific Computing.* Birkhäuser Press.

Balay, S., Buschelman, K., Eijkhout, V., Gropp, W.D., Kaushik, D., Knepley, M.G., Curfman-McInnes, L., Smith, B.F., & Zhang, H. 2004. *PETSc Users Manual.* Tech. rept. ANL-95/11 - Revision 2.1.5. Argonne National Laboratory.

Balay, S., Buschelman, K., Gropp, W.D., Kaushik, D., Knepley, M.G., Curfman-McInnes, L., Smith, B.F., & Zhang, H. 2006. *PETSc Web page.* http://www.-mcs.anl.gov/petsc.

Bhatia, R. 1997. *Matrix Analysis.* Springer. ISBN 0-387-94846-5.

Blazek, J. 1993. Investigations of the Implicit LU-SSOR Scheme. *DLR Forschungsbericht, DLR-FB-93/51.*

Blazek, J. 2001. *Computational Fluid Dynamics: Principles and Applications.* Elsevier Science. ISBN 0-08-043009-0.

Boussinesq, J. 1877. Theorie de l'Ecoulement Tourbillonant. *Comptes-Rendus de l'Academie les Sciences*, **23**, 46–50.

Brezillon, J., & Dwight, R.P. 2005. Discrete Adjoint of the Navier-Stokes Equaitons for Aerodynamic Shape Optimization. *Evolutionary and Deterministic Methods for Design, EUROGEN.*

Brezillon, J., & Gauger, N.R. 2004. 2D and 3D Aerodynamic Shape Optimization Using the Adjoint Aproach. *Aerospace Science and Technology Journal*, **8**(8), 715–727.

Brezillon, J., & Wild, J. 2005. Evaluation of Different Optimization Strategies for the Design of a High-Lift Flap Device. *EUROGEN 2005, Munich, September.*

Burdyshaw, C., & Anderson, W. 2005. A General and Extensible Unstructured Mesh Adjoint Method. *43th AIAA Aerospace Sciences Meeting and Exhibit, January 10-13, Reno, NV., AIAA-2005-0335.*

Butcher, J. C. 1987. *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods.* Wiley-Interscience.

Cai, X.-C., Gropp, W.D., Keyes, D.E., & Tidriri, M.D. 1995. Newton-Krylov-Schwarz methods in CFD. *Proceedings of the International Workshop on Numerical Methods for the Navier-Stokes Equations, Vieweg, Braunschweig*, 17–30.

Campobasso, M.S., & Giles, M.B. 2002. Effects of Flow Instabilities on the Linear Analysis of Turbomachinery Aeroelasticity. *Proceedings of 38th Joint Propulsion Conference and Exhibit, Indianapolis,AIAA-2002-4085.*

Campobasso, M.S., & Giles, M.B. 2004. Stabilization of a linear flow solver for turbomachinery aeroelasticity by means of the recursive projection method. *AIAA Journal*, **42**(9), 1765–1774.

Cantariti, F., Dubuc, L., Gribben, B., Woodgate, M., Badcock, K., & Richards, B. 1997. Approximate Jacobians for the Solution of the Euler and Navier-Stokes Equations. *Glasgow University, Department of Aerospace Engineering, Technical Report 9705.*

Cantariti, F., Woodgate, M., Badcock, K., & Richards, B. 1999. Solution of the Navier-Stokes Equations in Three Dimensions Using a Fully Unfactored Method. *Glasgow University, Department of Aerospace Engineering, Technical Report 9908.*

Castro, C., Lozano, C., Palacios, F., & Zuazua, E. 2006. A Systematic Continuous Adjoint Approach to Viscous Aerodynamic Design on Unstructured Grids. *44th AIAA ASM Conference, Reno, NV. AIAA-2006-0051.*

Chisholm, T., & Zingg, D.W. 2002. A Fully Coupled Newton-Krylov Solver for Turbulent Aerodynamic Flows. *ICAS 2002 Conference, Toroto, Paper 333.*

Cook, P.H., McDonald, M.A., & Firmin, M.C.P. 1979. *Aerofoil RAE 2822 — Pressure Distributions and Boundary Layer and Wake Measurements.* AGARD–AR, no. 138. Neuilly-sur-Seine, France: AGARD — Advisory Group for Aerospace Research & Development. Chap. 6.

Dubuc, L., Badcock, K., Richards, B., & Woodgate, M. 1996. Implicit Navier-Stokes simulations of unsteady flows. *Proceedings of R.Ae.Soc Conference in Unsteady Aerodynamics, London.*

Dwight, R.P. 2004. Application of Approximately Factored Implicit Schemes to Unsteady Navier-Stokes Calculations. *Proceedings of the ICCFD3 Conference, Toronto, Springer.*

Dwight, R.P., & Brezillon, J. 2006. Effect of Various Approximations of the Discrete Adjoint on Gradient-Based Optimization. *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno NV. AIAA-2006-0690.*

Edwards, J.R., & Chandra, S. 1996. Comparison of Eddy-Viscosity Transport Turbulence Models for Three-Dimensional Shock-Seperated Flowfields. *AIAA Journal*, **34**(4).

Faßbender, J. 2003. *Improved Robustness for Numerical Simulations of Turbulent Flows around Civil Transport Aircraft at Flight Reynolds Numbers.* Ph.D. thesis, Institute of Aerodynamics and Flow Technology, DLR, Brauschweig.

Feingold, D.G., & Varga, R.S. 1962. Block Diagonally Dominant Matrices and Generalizations of the Gerschgorin Circle Theorem. *Pacific J. Math.*, **12**, 1241–1250.

Galle, M. 1995. Solution of the Euler and Navier-Stokes Equations of Hybrid Grids. *AGARD-CP-578*, 31.1–31.9.

Galle, M. 1999. Ein Verfahren zur numerischen Simulation kompressibler, reibungsbehafteter Strömungen auf hybriden Netzen. *DLR Forschungsbericht, DLR-FB-99/04.*

Gauger, N., & Brezillon, J. 2003. Aerodynamic shape optimization using the adjoint method. *Journal of the Aeronautical Society of India*, **54**(3).

Geiger, & Kanzow. 1999. *Numerische Verfarhren zur Loesung unrestringierter Optimierungsaufgaben.* Springer.

Gerhold, T., Galle, M., Friedrich, O., & Evans, J. 1997. Calculation of Complex 3D Configurations Employing the DLR TAU-Code. *AIAA Paper, AIAA-97-0167.*

Giles, M.B. 2001. On the iterative solution of adjoint equations. *Automatic Differentiation: From Simulation to Optimization*, 145–152.

Giles, M.B., Duta, M.C., Muller, J.D., & Pierce, N.A. 2003. Algorithm Developments for Discrete Adjoint Methods. *AIAA Journal*, **41**(2), 198–205.

Glowinski, R., & Pironneau, O. 1975. On the Numerical Compuation of the Minimum-Drag Profile in Laminar Flow. *Journal of Fluid Mechanics*, **72**, 385–389.

Griewank, A., & Walther, A. 2002. On Constrained Optimization by Adjoint Based Quasi-Newton Methods. *Optimization Methods and Software*, **17**(5), 869–889.

Griewank, Andreas. 2000. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation.* Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia. ISBN 0898714516.

Hicks, R.M., & Henne, P.A. 1978. Wing Design by Numerical Optimization. *Journal of Aircraft*, **15**, 407–412.

Hirsch, C. 1989. *Numerical Computation of Internal and External Flows: Fundamentals of Numerical Discretization v. 1.* John Wiley and Sons Ltd. ISBN 0471923850.

Jameson, A. 1988. Aerodynamic Design Via Control Theory. *Journal of Scientific Computing*, **3**, 233–260.

Jameson, A. 1991. Time dependant calculations using multigrid with applications to unsteady flows past airfoils and wings. *AIAA Paper.*

Jameson, A. 1995. Analysis and design of numerical schemes for gas dynamics I: artificial diffusion, upwind biasing, limiters and their effect on accuracy and multigrid convergence. *International Journal of Computational Fluid Dynamics*, **4**(171).

Jameson, A., & Baker, T. 1984. Multigrid Solution of the Euler Equations for Aircraft Configurations. *AIAA Paper, AIAA-84-0093.*

Jameson, A., & Caughey, D.A. 2001. How Many Steps are Required to Solve the Euler Equations of Steady Compressible Flow: In Search of a Fast Solution Algorithm. *15th AIAA Computational Fluid Dynamics Conference, June 11-14, 2001, Anaheim, CA.*

Jameson, A., & Turkel, E. 1981. Implicit schemes and LU-decompositions. *Mathematics of Computation*, **37**, 385–397.

Jameson, A., Schmidt, W., & Turkel, E. 1981. Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes. *AIAA Paper. AIAA-81-1259.*

Johnson, F., Tinoco, E., & Yu, N. 2003. Thirty Years of Development and Application of CFD at Boeing Commercial Airplanes, Seattle. *16th AIAA CFD Conference, Orlando, Florida.*

Kim, C.S., Kim, C., & O.H., Rho. 2003. Feasibility Study of the Constant Eddy Viscosity Assumption in Gradient-Based Design Optimization. *Journal of Aircraft*, **40**, 1168–1176.

Kim, S., Alonso, J., & Jameson, A. 2002. Design Optimization of High-Lift Configurations Using a Viscous Continuous Adjoint Method. *40th AIAA Aerospace Sciences Meeting and Exhibit, Reno. AIAA-2002-0844.*

Knoll, D.A., & Keyes, D.E. 2004. Jacobian-free Newton-Krylov Methods: A Survey of Approaches and Applications. *Journal of Computational Physics*, **193**, 357–397.

Kroll, N., & Fassbender, J. K. (eds). 2005. *MEGAFLOW — Numerical Flow Simulation for Aircraft Design: Results of the second phase of the German CFD initiative MEGAFLOW presented during its closing symposium at DLR, Braunschweig, Germany, December 10th and 11th 2002.* Notes on Numerical Fluid Mechanics and Multidisciplinary Design, vol. 89. Berlin: Springer Verlag.

Kroll, N., Gauger, N.R., Brezillon, J., Becker, K., & Schulz, V. 2004. Ongoing Activities in Shape Optimization within the German Project MEGADESIGN. *ECCOMAS Finland, 24-28 July.*

Le Chuiton, F. 2004. On the Non-Dimensionalization of the Navier-Stokes equations. *DLR Institutsbericht, AS. IB 124-2004/10.*

Löhner, R., & Galle, M. 2002. Minimization of Indirect Addressing for Edge-Based Field Solvers. *40th AIAA Aerospace Sciences Meeting and Exhibit, January 14-17, Reno, NV.*

Löhner, R., Soto, O., & Yang, Chi. 2003. An Adjoint-Based Design Methodology for CFD Optimization Problems. *AIAA Paper, AIAA-03-0299.*

Luo, H., Baum, J.D., & Löhner, R. 1998. A Fast, Matrix-Free Implicit Method for Compressible Flows on Unstructured Grids. *Journal of Computational Physics*, **146**, 664–690.

Mavriplis, D. 2005. Formulation and Multigrid Solution of the Discrete Adjoint for Optimization Problems on Unstructured Meshes. *43th AIAA Aerospace Sciences Meeting and Exhibit, January 10-13, Reno, NV.*

Mavriplis, D. 2006. A Discrete Adjoint-Based Approach for Optimization Problems on Three-Dimensional Unstructured Meshes. *44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada. Paper AIAA-2006-50.*

Mavriplis, D. J. 2002. An assessment of linear versus nonlinear multigrid methods for unstructured mesh solvers. *J. Comput. Phys.*, **175**(1), 302–325.

Mavriplis, D.J. 1997. Adaptive Meshing Techniques for Viscous Flow Calculation on Mixed-Element Unstructured Meshes. *AIAA Paper, AIAA-97-0857.*

Mavriplis, D.J. 1998. On Convergence Acceleration Techniques for Unstructured Meshes. *ICASE Report, No. 98-44.*

Mavriplis, D.J. 2004. Personal communication.

Meister, A. 1998. Comparison of Different Krylov Subspace Methods Embedded in an Implicit Finite Volume Scheme for the Computation of Viscous and Inviscid Flow Fields on Unstructured Grids. *Journal of Computational Physics*, **140**, 311–345.

Menter, F.R. 1993. Zonal two-equation k-w turbulence model for aerodynamic flows. *AIAA Paper. AIAA-1993-2906.*

Nadarajah, S., & Jameson, A. 2000. A Comparison of the Continuous and Discrete Adjoint Approach to Automatic Aerodynamic Optimization. *38th Aerospace Sciences Meeting and Exhibit, Reno. AIAA-2000-0667.*

Nadarajah, S., & Jameson, A. 2001. Studies of the Continuous and Discrete Adjoint Approaches to Viscous Automatic Aerodynamic Shape Optimization. *15th AIAA Computational Fluid Dynamics Conference, Anaheim. AIAA-2001-2530.*

Nielsen, E., & Kleb, B. 2005. Efficient Construction of Discrete Adjoint Operators on Unstructured Grids by using Complex Variables. *43th AIAA Aerospace Sciences Meeting and Exhibit, January 10-13, Reno, NV.*

Nielsen, E., & Park, M. 2005. Using an Adjoint Approach to Eliminate Mesh Sensitivities in Computational Design. *43th AIAA Aerospace Sciences Meeting and Exhibit, January 10-13, Reno, NV., AIAA-2005-0491.*

Nielsen, E., Walters, R., Anderson, W.K., & Keyes, D. 1995. Application of Newton-Krylov Methodology to a Three-Dimensional Unstructured Euler Code. *Proceedings of the 12th AIAA CFD Conference, San Diego. AIAA-95-1733.*

Nielsen, E.J. 1998. *Aerodynamic Design Sensitivities on an Unstructured Mesh using the Navier-Stokes Equations and a Discrete Adjoint Formulation.* Ph.D. thesis, Virginia State University.

Peaceman, D., & Rachford, H. 1955. The Numerical Solution of Parabolic and Elliptic Differential Equations. *Journal of the Society for Industrial and Applied Mathematics*, **3**(1), 28–41.

Pierce, N. 1997. *Preconditioned Multigrid Methods for Compressible Flow Calculations on Stretched Meshes.* Ph.D. thesis, Oxford University.

Pierce, N.A., Giles, M.B., Jameson, A., & Martinelli, L. 1997. Accelerating Three-Dimensional Navier-Stokes Calculations. *AIAA Paper, AIAA-97-1953.*

Pironneau, O. 1973. On Optimum Profiles in Stokes Flow. *Journal of Fluid Mechanics*, **59**, 117–128.

Pironneau, O. 1974. On Optimum Design in Fluid Mechanics. *Journal of Fluid Mechanics*, **64**, 97–110.

Quirk, J.J. 1994. A contribution to the great Riemann solver debate. *International Journal of Numerical Methods in Fluids*, **18**(555).

Radespiel, R., & Kroll, N. 1995. Accurate flux vector splitting for shocks and shear layers. *Journal of Computational Physics*, **121**(66).

Reuther, J., Alonso, J., Rimlinger, M.J., Sanders, D., & Jameson, A. 1999. Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers. *Journal of Aircraft*, **36**, 51–60.

Roberts, T.W., & Swanson, R.C. 2005. A Study of Multigrid Preconditioners Using Eigensystem Analysis. *AIAA Paper, AIAA-2005-5229.*

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

Rossow, C. 2005. Convergence Acceleration for Solving the Compressible Navier-Stokes Equations. *43rd AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, 2005. AIAA-2005-0094.*

Rudnik, R., Heinrich, R., Eisfeld, B., & Schwarz, T. 2004. DLR Contributions to Code Validation Activities within the European High Lift Project EUROLIFT. *DGLR, New Results in Numerical and Experimental Fluid Dynamics IV, Springer Verlag*, 42–49.

Saad, Y. 2003. *Iterative Methods for Sparse Linear Systems - 2nd ed.* SIAM. ISBN 0-89871-534-2.

Saad, Y., & Schultz, M. H. 1988. GMRES: A generalized minimum residual algorithm for solving non-symmetric linear systems. *SIAM Journal of scientific and statistical computing*, **7**(3), 856–859.

Sharov, D., Luo, H., & Baum, J.D. 2000. Implementation of Unstructured Grid GMRES+LU-SGS Method on Shared-Memory, Cache-Based Parallel Computers. *AIAA Paper, AIAA-2000-0927.*

Sirkes, Z., & Tziperman, E. 1997. Finite difference of adjoint or adjoint of finite difference? *American Meteorological Society*, **125**, 3373–3378.

Soda, A., Knopp, T., & Weinman, K. 2005. Numerical Investigation of Transonic Shock Osciilations on Stationary Aerofoils. *Hybrid RANS-LES Symposium, Stockholm, July.*

Soto, O., Löhner, R., & Yang, Chi. 2004. An Adjoint-Based Design Methodology for CFD Problems. *International Journal of Numerical Methods in Heat and Fluid Flow*, **14**(6), 734–759.

Spalart, P.R., & Allmaras, S.R. 1992. A One-Equation Turbulence Model for Aerodynamic Flows. *30th AIAA Aerospace Sciences Meeting and Exhibit, January 6-9, 1992, Reno, NV., AIAA-92-0439.*

Spalart, P.R., & Bogue, D.R. 2003. The Role of CFD in Aerodynamics Off-Design. *The Aeronautical Journal.*

Swanson, R.C., Turkel, E., Rossow, C.C., & Vatsa, V. 2005. Convergence Acceleration for Multistage Time-Stepping Schemes. *AIAA Paper.*

Turkel, E. 1988. Improving the accuracy of central difference schemes. *ICASE Report.*

Turkel, E., Fiterman, A., & van Leer, B. 1994. Preconditioning and the limit to the incompressible flow equations. *Computing the Future: Frontiers of Computational Fluid Dynamics*, 215–234.

Turkel, E., Vasta, V., & Venkatakrishnan, V. 1999. Uni-Directional Implicit Acceleration Techniques for Compressible Navier-Stokes Solvers. *AIAA Paper, AIAA-99-3265.*

Van der Ven, H., & Weinman, K. 2004. NLR/DLR Co-operation in XLES/DES Methods. *NLR/DLR Internal Report.*

Van Leer, B. 1982. Flux-vector splitting for the Euler equations. *Proceedings of 8th International Conference on Numerical Methods in Fluid Dynamics, Aachen. Springer-Verlag*, 507–512.

Venkatakrishnan, V. 1998. Improved Convergence of Compressible Navier-Stokes Solvers. *AIAA Paper, AIAA-98-2967.*

Wilcox, D.C. 1998. *Turbulence Modeling for CFD - 2nd ed.* DWC Industries. ISBN 0-9636051-5-1.

Wild, J. 2003. On the Potential of Numerical Optimization of High-Lift Multi-Element Airfoils based on the Solution of the Navier-Stokes Equations. *Proceedings of the ICCFD II Conference, Sydney, Springer Verlag*, 267–273.

Wild, J. 2004. Acceleration of Aerodynamic Optimization Based on RANS-Equations by using Semi-Structured Grids. *Design Optimization International Conference, Athens.*

Wild, J., Mertins, R., Quagliarella, D., Brezillon, J., Quest, J., Amoignon, O., & Moens, F. 2005. Applying Numerical Optimization to Realistic High-Lift Design of Transport Aircraft - An Overview of the Aerodynamic Design Optimization Investigations within the EUROLIFT II Project. *EUROGEN 2005, Munich, September.*

Wong, P., & Zingg, D.W. 2005. Aerodynamic Computations on Unstructured Grids Using a Newton-Krylov Approach. *AIAA Paper 2005-5231.*

Woodgate, M., Badcock, K., Cantariti, F., & Richards, B. 1997. Solution of the Euler Equations in Three Dimensional Complex Geometries Using a Fully Unfactored Method. *Glasgow University, Department of Aerospace Engineering, Technical Report 9705.*

Yoh, J.J., & Zhong, X. 2004. New Hybrid Runge-Kutta Methods for Unsteady Reactive Flow Simulation. *AIAA Journal*, **42**(8), 1593–1600.

Yoon, S., & Jameson, A. 1986a. Lower-Upper implicit schemes with multiple grids for the Euler equations. *AIAA Paper, AIAA-86-0105.*

Yoon, S., & Jameson, A. 1986b. A multigrid LU-SSOR scheme for approximate Newton-iteration applied to the Euler equations. *NASA Paper, NASA-CR-17954.*

Yoon, S., & Jameson, A. 1988. An LU-SSOR Scheme for the Euler and Navier-Stokes Equations. *AIAA Journal*, **26**, 1025–1026.