Solution methods and acceleration techniques in CFD

R. P. Dwight\textsuperscript{1}, H. Bijl\textsuperscript{1}

\textsuperscript{1}Aerodynamics Group, Aerospace Engineering, T.U. Delft, P.O. Box 5058, 2600GB Delft, The Netherlands.

ABSTRACT

Methods for solving the discretized equations of fluid motion are discussed, with particular reference to efficient methods for solving compressible flow problems to a steady state. A spectrum of algorithms of widely varying efficiency and complexity are considered, from explicit single-stage methods, through Runge-Kutta, implicit methods, defect correction, to multigrid. They are classified with respect to cost per iteration, convergence rate and memory requirements, and compared for a concrete high-Reynolds number test-case. Time-resolved calculations are then considered using the dual-time framework.

key words: Pseudo-time integration, Runge-Kutta, implicit methods, multigrid.

1. INTRODUCTION

After performing the spatial discretization of a continuous partial differential equations (PDE) governing the motion of a fluid, one is left with a large coupled system of ordinary differential equations (ODEs) describing the fluid motion, with a single independent variable, time $t$. For example the continuous equations of compressible inviscid fluid flow (the Euler equations) are

$$\frac{\partial w}{\partial t} + \frac{\partial}{\partial x_i} f_i(w) = 0, \quad \text{on } \Omega,$$

where $w = (\rho, \rho u, \rho v, \rho E)$ is the vector of conservative variables, $f$ is the Euler flux, and $\Omega$ is the physical domain. Applying any spatial discretization of these equations, they may be written as an ODE known as the semi-discrete form:

$$\frac{dW_i}{dt} + R_i(W) = 0, \quad \forall i,$$

where $W_i$ are now the discrete dependant variables, one for each degree-of-freedom (DoF) of the spatial discretization. E.g. a finite volume method would give 4 DoFs at each mesh point, one corresponding to each of the 4 unknowns. The details of the entire spatial discretization, including boundary conditions and $\Omega$, are contained in the residual $R$, which is a (generally non-linear) algebraic operator. It is the resolution of this large ODE that is of concern when we talk about solution methods, as its efficient solution is the deciding factor in the efficiency of a numerical code.
Two cases may be distinguished, that of solving (2) to a stationary state, $R_i(W) = 0$, discussed in Section 2; and solving it with resolution of transient motion, see Section 3. In the following we concentrate on techniques typically used for compressible flow, as characterized by (1), but much of the discussion will be relevant to incompressible regimes too. We also restrict ourselves to methods that apply for general solvers, ignoring e.g. alternating-direction implicit (ADI) methods which can be very effective, but only on structured grids.

2. STEADY STATE SOLUTION METHODS

We are interested in finding the vector $W$ of size $N$ satisfying the $N$ non-linear algebraic equations $R_i(W) = 0$. This represents a stationary solution of our discrete flow problem. The classical approach to zero-finding of non-linear systems is Newton’s method, whose iteration may be written:

$$W^{n+1} = W^n - \left[ \frac{\partial R}{\partial W} \right]_{W^n}^{-1} \cdot R(W^n),$$

where $W^n$ is the known solution at iteration/time-level $n$, and $W^{n+1}$ is unknown. Newton’s method converges quadratically in only a few steps, provided $W^n$ is sufficiently close to the exact solution. However it has many disadvantages that make it impractical for use in large-scale CFD problems. Firstly the Jacobian, the sparse $N \times N$ matrix $\partial R/\partial W$, (a) is difficult to implement, as $R$ is usually a complex function of $W$ which must be differentiated, (b) consumes large amounts of memory, often comparable to several times the memory required for the remainder of the code (Dwight, 2006), (c) is difficult and expensive to invert. Furthermore in practice the Newton iteration fails to converge unless the initial condition $W(0)$ is extremely close to the correct solution — most particularly for the very stiff problems that arise in CFD.

There exist techniques for mitigating all these issues: (a) and (b) may be circumvented either by using finite differences to obtain the product of the Jacobian with a vector, or by using automatic differentiation tools. Start-up problems may be treated using some kind of continuation method (i.e. using Newton to solve a simpler problem first, and then using the solution as an initial condition for a slightly more difficult problem etc.). Often the continuation parameter is the grid, which starts coarse and becomes increasingly fine. See Knoll and Keyes, 2004 for discussion of all these methods. Issue (c) is less tractable - essentially one has converted a stiff non-linear problem into a stiff linear problem, the essential solution difficulty remains, and one must solve several of these linear problems to obtain a non-linear solution.

In general the most commonly used solution methods follow a different approach that exploits the physics of the system. Instead of solving directly $R(W) = 0$, we retain the time term, and integrate (2) so far forward in time, that the solution does not change anymore, known as iterating to a steady state. Because we are not interested in the transient behaviour, we are not restricted to considering integration schemes that are time-accurate — and this is the source of much flexibility, leading to a broad zoo of possible methods.

Concretely many of the steady-state iterations to be discussed in the following may be

---

*Notably simple Jacobi and Gauss-Seidel iterations diverge for Jacobians resulting from second-order finite volume discretizations of the compressible Euler equations.*

written in the form:

$$A(W^n) \cdot \left[ W^{n+1} - W^n \right] + R(W^n) = 0,$$

(4)

where $A$ is some *freely chosen* $N \times N$ matrix which must be inverted (possibly approximately) to obtain an explicit expression for $W^{n+1}$ in terms of $W^n$. Note that the left-most term in (4) takes the place of the time discretization. Provided $A$ is invertible, $W$ is a stationary point of the iteration if and only if it is a solution of $R(W) = 0$. The choice of $A$, and means of inversion define the method completely - and by virtue of this freedom of choice efficient and robust methods may be devised. Note that Newton’s method (3) is a special case of (4).

In the following subsections we describe a series of increasingly sophisticated methods, considering explicit schemes, defect correction, and culminating in the non-linear multigrid method. For concreteness we apply each method to transonic flow about the RAE 2822 aerofoil, at Mach 0.73, angle-of-attack $2.8^\circ$ and Reynolds number $6.5 \times 10^6$. A Reynolds-averaged Navier-Stokes formulation is used, with a second-order central finite volume scheme and Spallart-Almaras one-equation turbulence model. The simulations are performed with the DLR TAU-code (Gerhold et al., 1997) on the hybrid grid shown in Figure 1 with $30 \times 10^3$ points. Convergence is assessed by examining the normalized magnitude of the residual $\|R_p(W^{(n)})\|/\|R_p(W^{(0)})\|$, with respect to iteration count and wall-clock time (distributed memory parallel on $8 \times 2.26$GHz cores), shown in Figures 2 and 3 respectively. The drag coefficient computed at transient states is shown in Figure 4, this is one of the desired quantities for which the calculation is performed in the first place. It is not necessarily the case that rapid convergence of the residual implies rapid convergence of drag. This is because error in drag tends to be dominated by low frequency solution error (there is a cancellation effect with high frequency error), and residual error tends to be dominated by high frequency solution error (this is the reason why for most iterative methods the residual converges more quickly initially). Reading these convergence plots is very simple – a guideline is that a solution is sufficiently converged for most purposes when the residual reaches $10^{-4} \rightarrow 10^{-5}$, or the drag coefficient stands still.

In order to give a better overview of the methods presented in relation to each other we offer Figure 5, which broadly and qualitatively characterizes them in terms of rate of convergence, effort per iteration and memory costs. This graphic represents the experience of the authors in aerodynamics, and the positions of the methods will vary somewhat with algorithmic and implementational details. Individual features will be elucidated in the following discussion.

### 2.1. Explicit single-stage methods

The simplest possible choice of $A$ in (4) is

$$A = \frac{1}{\Delta t} I,$$

where $I$ is the $N \times N$ identity matrix and $\Delta t$ is some time-step. The method is an explicit first-order accurate time discretization of (2), and is known as the *forward Euler method*. It is desirable to choose $\Delta t$ as large as possible, so that a steady-state is achieved rapidly. However the iteration is only stable if the time-step satisfies the fundamental *Courant-Friedrichs-Dewy (CFL) condition*. This is a necessary condition for stability which requires that the numerical domain of dependence of every point entirely contains the physical domain of dependence of that point (Courant et al., 1967).

As an example consider a wave propagating at speed $u$ across an infinite 1d grid with spacing $\Delta x$. The governing equation is

$$\frac{\partial \phi}{\partial t} + u \frac{\partial \phi}{\partial x} = 0,$$

for $u > 0$, which may be discretized as e.g.

$$\frac{\phi_i^{(n+1)} - \phi_i^{(n)}}{\Delta t} + \frac{u}{\Delta x} \frac{\phi_{i+1}^{(n)} - \phi_i^{(n)}}{\Delta x} = 0,$$

where the superscript and subscripts are time and space indices respectively. In time $\Delta t$ information travels a distance $u\Delta t$ in the physical problem, so it should travel at least at that speed in the numerical problem to assure that the scheme can access the information required to form the solution. For the above discretization information is propagated a distance $\Delta t x$ in time $\Delta t$ (in the positive $x$-direction only). Hence the CFL condition stipulates

$$u \Delta t \leq \Delta x. \quad (5)$$

In practice grids with widely varying cell sizes are common. The CFL condition must be satisfied for the smallest cell in the grid, limiting the size of $\Delta t$ everywhere. In our high-Reynolds number RAE 2822, the boundary-layer is resolved with stretched cells with an aspect ratio of about $1 : 1500$, and a height of $1 \times 10^{-5}$ in units of aerofoil chord length. If the global time-step is based on the CFL condition in these cells, then $1 \times 10^5$ time-steps are required to transport information once from the leading- to the trailing-edge. If the convergence of this method were plotted in Figure 2 it would be indistinguishable from a horizontal line. However by recognizing that time-accuracy, and therefore a uniform time-step, is not necessary, we can perform local time-stepping

$$A = \frac{1}{\Delta t_i} f, \quad \Delta t_i = \frac{\Delta x_i}{u_i},$$

where the subscript $i$ indicates a per-cell evaluation of the quantities – thus the worst CFL restrictions are confined to the smallest cells. This is the cheapest and – at the same time – slowest iteration considered and therefore belongs at the extreme lower left of Figure 5, diagonally opposite the most expensive approach, Newton, which also needs the least iterations. In terms of iterations it needs in excess of 50,000 iterations to converge to a residual of $10^{-4}$. In practice local time-stepping is never used alone, only in combination with other methods to be presented.

Note that the maximum speed of information propagation $u$ depends on the local solution and governing equations (for incompressible flow it is infinite, and fully explicit methods are always unstable). Finally the CFL number is defined as ratio of the time-step employed locally to the time-step specified by the CFL condition (5). This may exceed 1.0 if an iterative method is used which communicates information further than one cell within one time-step, for example multistage methods.

### 2.2. Explicit multi-stage methods

**Runge-Kutta** (RK) methods employ multiple evaluations of the residual $R$ per time-step, and were originally developed to obtain high-order accuracy in time. The basis for the use of explicit-RK as a convergence acceleration technique is its excellent error damping properties,
particularly in high-frequency components of the error (Hirsch, 1989; Jameson et al., 1981) – which makes them suitable for use as multigrid smoothers, see Section 2.4. The basic $m$-stage scheme (with local $\Delta t$), may be written
\[
W^{(0)} = W^n, \\
W^{(j)} = W^{(0)} - \alpha_j \Delta t \cdot R(W^{(j-1)}), \quad \forall j \in 1 \ldots m, \\
W^{n+1} = W^{(m)},
\]
where the subscript $i$ has been dropped for simplicity, and $\alpha_j$ are scalar constants. These are chosen to optimize the scheme with respect to stability. The choice is typically based on a von Neumann analysis of the discrete system, and depends on the spatial discretization (Hirsch, 1989). For the centrally discretized Euler equations Jameson et al., 1981 suggested a 3-stage RK method with $\alpha = (\frac{2}{3}, \frac{2}{3}, 1)$, and a 5-stage RK method with $\alpha = (\frac{1}{3}, \frac{1}{3}, \frac{3}{8}, \frac{1}{2}, 1)$. In general Runge-Kutta methods include stages with a dependence on all previous stages, requiring storage of several solution vectors. This is necessary to obtain high-order time-accuracy, but is not essential for good damping properties, and so is avoided.

Two common techniques for increasing the stability of explicit methods were developed in the context of the compressible Euler equations: (a) applying a low-pass filter $Q$ (typically a Laplacian smoother) to the residual, and (b) updating stabilization terms (such as artificial dissipation) intermittently. If the residual may be written $R = C + D$, where $D$ represents the contribution from stabilization terms, then an improved method is
\[
W^{(0)} = W^n, \quad D^{(0)} = D(W^{(0)}), \\
W^{(j)} = W^{(0)} - \alpha_j \Delta t \cdot Q \cdot \left( C(W^{(j-1)}) + D^{(j)} \right), \quad \forall j \in 1 \ldots m, \\
D^{(j)} = \beta_j D(W^{(j-1)}) + (1 - \beta_j) D^{(j-1)}, \quad \forall j \in 1 \ldots m, \\
W^{n+1} = W^{(m)}.
\]
Both these modifications are motivated by von Neumann analysis (Blazek, 2001). In the RAE 2822 case for a 3-stage scheme with $\beta = (1, 0, 0)$, these modifications increase the stable CFL number from 1.7 to 2.1, and improve the convergence somewhat, Figure 2. In particular a residual of $10^{-4}$ is reached after 4,150 iterations of RK3 and only 3,300 iterations of the improved RK3. The improvement of basic Runge-Kutta over single-stage local time-stepping is much more dramatic however. Also RK with $m$ stages is about $m$ times more costly than local time-stepping, hence its place in Figure 5.

2.3. Implicit methods and defect correction

The main disadvantage of explicit methods is the limitation imposed by the CFL condition. This can be overcome with implicit methods, at the expense of a non-trivial inversion of $A$. Start with an implicit time discretization of (2), and linearize about the known time level:
\[
\frac{W^{n+1} - W^n}{\Delta t} = -R(W^{n+1}) = - \left( R(W^n) + \left[ \frac{\partial R}{\partial W} \right]_{W^n} \Delta W^n + \mathcal{O}(\Delta W^2) \right),
\]
where $\Delta W^n = W^{n+1} - W^n$. Neglecting higher-order terms and rearranging gives
\[
\left( \frac{I}{\Delta t} + \left[ \frac{\partial R}{\partial W} \right]_{W^n} \right) \cdot \Delta W^n = -R(W^n),
\]
where the left-most term may be identified with \( A \). In general \( W^{n+1} \) is a function of \( W^n \) at all mesh points (provided \( A \) is irreducible) and thus the CFL condition is removed. In the limit \( \Delta t \to \infty \) this is Newton’s method, eqn. (3). Having established that Newton’s method is inappropriate, we attempt to devise alternatives that are significantly cheaper, but retain some of the desirable convergence properties.

To this end consider replacing \( \partial R/\partial W \) with some approximation \( \tilde{A} \) (known as approximate Newton), and solving the linear system (6) only roughly at each iteration (inexact Newton). Both of these simplifications save effort per iteration, at the cost of a lower convergence rate. They are also complementary – given \( \tilde{A} \) a very poor approximation to the Jacobian, solving (6) to high accuracy does not buy any convergence improvement, as the “wrong” problem is being solved. Similarly, taking time to construct an accurate \( \tilde{A} \) and then solving (6) very roughly is also a waste of resources. These are examples of what is meant by over-solved and under-solved methods respectively, in the top-left area of Figure 5.

A natural choice for \( \tilde{A} \) arises from low-order discretizations. For example, given \( R^{(1)} \) and \( R^{(2)} \), first- and second-order discretizations of the governing equations in space, let \( \tilde{A} = \partial R^{(1)}/\partial W \) (the aim is to solve \( R^{(2)} = 0 \)). Typically the sparse matrix \( \partial R^{(1)}/\partial W \) will be much better conditioned than \( \partial R^{(2)}/\partial W \) and will have significantly less fill-in. Such first-order problems may be easily solved with Gauss-Seidel, Incomplete Lower-Upper (ILU) factorization, etc., and must be solved repeatedly to iterate toward the solution of the second-order problem. This approach may also be seen as a linearization of the defect correction (DC) method (Koren, 1990):

\[
R^{(1)}(W^{n+1}) = R^{(1)}(W^n) - R^{(2)}(W^n),
\]

where the non-linear problem associated with \( R^{(1)} \) must be solved repeatedly. The defect correction method has many desirable convergence properties. In particular, under certain regularity and consistency conditions on the operators it may be shown that if the lower-order operator is first-order in space, one defect correction iteration gains one order of accuracy in the solution (Koren, 1990). Hence only two iterations are required to obtain second-order accuracy. Numerical evidence in CFD of such theoretical results is not often seen however, and typically 10 – 20 defect correction iterations must be applied to obtain convergence to an acceptable absolute level.

As an example of implicit methods in Figure 2 convergence of the Lower-Upper Symmetric-Gauss-Seidel (LU-SGS) is presented (Yoon and Jameson, 1988; Dwight, 2006). This uses an \( \tilde{A} = A^{\text{Low}} \) that is a heavily simplified version of the first-order Jacobian. The simplifications are designed to make \( A^{\text{Low}} \) fast to construct while still allowing infinite CFL numbers. The system is solved at each implicit iteration with a single SGS sweep (increasing the number of sweeps does not improve convergence significantly). In total the cost per iteration is lower than that of RK(3), and convergence is more than twice as fast: LU-SGS converges to \( 10^{-4} \) in about 1,200 iterations. In Figure 5 LU-SGS is placed at a similar effort to RK. Using a single Jacobi iteration instead of SGS reduces the convergence a lot, but the cost only a little. Using two SGS sweeps increases the cost without improving convergence significantly. Hence \( A^{\text{Low}} \) and a single SGS iteration seem to complement each other well.

With respect to Newton methods Jacobian-Free Newton-Krylov (JFNK) methods should be mentioned, which have the same convergence properties of Newton methods, but avoid the expense of constructing and storing the Jacobian matrix (Knoll and Keyes, 2004). Instead finite-differences are performed on \( R \) when a Jacobian-vector product is required, and this
product is the only operation on the matrix needed by many Krylov methods (Saad, 2003). Such techniques are most effective in unsteady simulation, where start-up issues are mitigated by initial conditions coming from the previous time-step (Lucas et al., 2009).

2.4. Multigrid methods for CFD

Of all classes of solution methods, only multigrid methods have the property of termination in $O(N)$ operations, where $N$ is the number of degrees-of-freedom of the problem – this is known as grid-independent convergence, and is increasingly important as computer capacity and problem size grows. Although this result may be proven only for elliptic problems, multigrid provides close to grid-independent convergence for a wide range of problems, also in CFD. Multigrid is a complex and subtle framework with many interrelated choices to be made for each problem. There is insufficient space for a satisfactory description of the method here - refer to Trottenberg et al., 2000, for an excellent introduction and detailed overview of this subject.

Briefly, the key observation leading to multigrid is that iterative methods such as Gauss-Seidel and Runge-Kutta are much more effective at smoothing high frequency than low frequency error in the solution. Therefore normally after a few Gauss-Seidel iterations the error is dominated by low frequency components and the solution converges slowly (in particular the drag and lift converge slowly). However low frequencies on a fine mesh are high frequencies on a coarse mesh - because the highest frequency error that the coarse mesh can represent is lower. The idea is to smooth low frequency errors on coarse meshes, where they can be damped effectively, and use this to correct the fine grid solution.

So let the subscripts $h$ and $H$ denote fine and coarse grids respectively, let $I^H_h$ be a restriction operator projecting the residual error on the fine grid to the coarse grid, $I^h_H$ a prolongation operator interpolating the coarse grid correction to the fine grid, and $S_h$ a smoothing operator $W^{n+1} = S_h W^n$, e.g. a single iteration of Gauss-Seidel, and $R_h$, $R^H_H$ spatial discretizations on the respective grids. If $R$ is linear then a complete single two-grid cycle may be written:

$$W^{n+1} = M^H_h \circ W^n,$$

$$M^H_h = S^\nu_2 (I_h - I^H_h R^H H^{-1} I_h R_h) S^\nu_1,$$

where $I_h$ is the identity.

This iteration can be broken into several parts. Firstly $\nu_1$ iterations of $S_h$ are applied on the fine grid - giving a fine grid solution improved in the high frequencies $\bar{W} = S^\nu_1_h W^n$. Next the residual of this solution on the fine grid is computed, and this residual is restricted to the coarse grid: $f = I^H_h R_h \bar{W}$. Thus $f$ (the forcing function) represents the errors in $\bar{W}$ which are low frequency on the fine grid. Note that if $\bar{W}$ is the exact solution of the discrete system then $f = 0$.

Next we solve the defect equation: $R^H_H \psi = f$ for $\psi$ the coarse grid correction. This problem may be seen as the equivalent of the fine grid problem on the coarse grid, but which solves for low frequency error in $\bar{W}$. Once $\psi$ is obtained it is prolonged to the fine grid and subtracted from the solution there. The correction must be prolonged in order to perform the subtraction. In fact the prolongation operator introduces some high-frequency error, so it is wise to smooth the solution after the subtraction of coarse grid correction with $\nu_2$ iterations of $S_h$. 

Encyclopedia of Aerospace Engineering. Edited by Richard Blockley and Wei Shyy.
© 2010 John Wiley & Sons, Ltd.
In the above the resolution of the defect equation is not necessarily exact, but perhaps some iterative procedure, e.g. another two-grid cycle. In this way the two-grid cycle may be applied recursively until the problem size is small enough that direct inversion is cheap. This is the multigrid method. Note that the two-grid cycle may be thought of as a defect correction iteration with $\tilde{A} = I_h^2R_H I_h^H$. Multigrid occupies an “island” in Figure 5 because its behaviour and performance is qualitatively different from other techniques.

An important issue is how to construct the coarse grids. If the fine grid is a regular structured grid then this is easy: every second grid line is removed in each direction, resulting in a nested grid with one-quarter as many cells (in 2d). This operation is performed recursively until a sufficiently coarse grid is obtained. Structured grids are typically constructed with blocks with $2^n$ points in each direction for exactly this reason. For unstructured grids one typically performs an equivalent procedure; several fine grid cells are agglomerated into a single coarse grid cell, again resulting in a nested pair of grids. The choice of which fine cells to agglomerate is critical, and is based on largely heuristic criteria such as uniformity of coarse grid cell size and shape, coarse grid cell convexity, and strength of connection (e.g. edge length) between neighbouring fine grid cells. The algorithms used tend also to have large heuristic components, but a starting point is often an algorithm from the field of graph partitioning (Mavriplis, 2002).

If $R$ is non-linear then multigrid may be applied directly using the more general Full-Approximation Storage (FAS) multigrid method (Trottenberg et al., 2000), or multigrid may be used within a Newton or implicit iteration, see Mavriplis, 2002 for a comparison of these approaches. FAS MG is the variety of multigrid used for comparison in Figure 2, once with Runge-Kutta 3-stage, and once with LU-SGS as the smoother. The coarse grids are obtained recursively by agglomerating cells from the finer grid in groups of about 4 cells. This can result in poorly shaped cells, so $R^{(1)}$ is used on coarse grids to improve stability. The cost per iteration is roughly equivalent to 2 smoother iterations on the fine grid, and the convergence is dramatically improved. Now only 250 and 150 iterations are needed for residual convergence to $10^{-4}$ for RK MG and LU-SGS multigrid respectively. Interestingly in Figure 4 it can be seen that the drag is not yet visibly stationary at these points, but first at 400 and 200 iterations respectively. This suggests that a convergence criteria of $10^{-4}$ is too lax for these methods, and highlights the importance of examining other convergence metrics.

Finally in this section we mention the distinctive convergence of Newton in Figures 2 and 3. The Newton iteration shown was started with an initial guess based on a partially converged solution from MG. The starting point shown was the earliest starting point which resulted in convergence of Newton – attempting to start Newton from a solution converged to a residual of e.g. $10^{-5}$ resulted in divergence. Note in Figure 4 that by the time it is possible to start Newton in this case, the drag is already fully converged. Once it starts up Newton converges within a few iterations (in the figures each iteration of Newton is a black circle), but the cost per iteration is about 120 times that of MG with a RK smoother. The Newton start-up problems in this case are mostly due to the turbulence model, which is highly non-linear and results in unusually large entries in the Jacobian matrix. A possible hybrid method might be more effective, combining Newton for the mean-flow equations with some other treatment of the turbulence equations - this is a promising avenue of research.
3. TIME-ACCURATE SOLUTION METHODS

Time-accurate simulation has many similarities to steady state simulation - especially as all the methods discussed in the previous section used iteration to a steady state (with the exception of Newton). In particular the discussion of the limitation of the CFL condition applies, so that explicit methods are inefficient when cells of widely varying size are needed - as in Reynolds-averaged Navier-Stokes. For this reason for high-Reynolds number RANS only implicit and semi-implicit (Blazek, 2001) methods are practical (this does not apply to Large Eddy Simulation). Fortunately it is possible to apply the iterative methods developed for steady flow to unsteady flow at each time-step, in a dual-time iteration.

3.1. Dual-time iterations

Consider discretizing the time derivate in (2), for example with the implicit second-order Backward Difference Formula (BDF):

\[
\frac{dW}{dt} + R(W) = \frac{W^{(n+1)} - 3 \cdot W^{(n)} + 2 \cdot W^{(n-1)}}{2\Delta t} + R(W^{(n+1)}) + \mathcal{O}(\Delta t^2) = 0,
\]

where \(W^{(n)}\) is the solution at physical time \(t^{(n)}\), and \(W^{(n+1)}\) is unknown. This system can be solved by introducing a fictive pseudo-time \(t^*\), the corresponding state \(W^*\), and deriving a modified residual \(R^*\):

\[
\frac{dW^*}{dt} + W^* - 3 \cdot W^{(n)} + 2 \cdot W^{(n-1)} = 0,
\]

\[
\frac{dW^*}{dt} + R^*(W^*) = 0.
\]
The equation is now in exactly the same form as (2) and all the methods of the previous section can be applied. Given that iteration on $W^*$ reaches a steady state, this will correspond to $W^{(n+1)}$. Note that this approach can be applied for any implicit time discretization.

Although this method seems quite expensive since a stationary non-linear problem is solved for each time-step, in practice it is not so bad as: (a) a good guess at the $W^{(n+1)}$ solution can be made by linearly extrapolating from $W^{(n-1)}$ and $W^{(n)}$, (b) $R^*$ is better conditioned than $R$ because of the addition of a term proportional to $1/\Delta t$ on the diagonal - the smaller $\Delta t$ the better the conditioning and stability, and (c) the non-linear system does not necessarily need to be solved to high accuracy at each physical time-step. On the other hand there are currently no techniques for determining how accurately the non-linear system must be solved at each time-step in order to retain the order-of-accuracy of the baseline scheme - this is a matter of case-by-case experience.

Figure 2. Convergence of relaxation methods in terms of iterations for the RAE 2822.
3.2. Two possible approaches

Given this framework there are two choices to be made: which baseline time-discretization to use, and how to solve the non-linear system. A common choice is to use second- or third-order BDFs with multigrid. This seems a reasonable choice: memory requirements are low because only two previous time-steps must be stored (for BDF(2)), and multigrid is the fastest solver for steady state simulation - suggesting it will be the fastest for the modified problem $R^* = 0$ too.

However recently it has been seen that this choice is significantly sub-optimal (Lucas et al., 2009). There is little penalty to using higher-order accurate time discretizations, and significant accuracy to be gained. Higher-order BDF formulas become rapidly unstable, due to the constant step size between samples. Preferable are time-accurate implicit Runge-Kutta
methods (Butcher, 1987; Carpenter et al., 2005), which have the additional advantages of being higher-order accurate from the first time-step (while BDFs require a special start-up procedure since e.g. $W^{(n-1)}$ may not exist on the zero-th time-step). Runge-Kutta methods also often include embedded error estimators (Butcher, 1987), providing a posteriori information on the time discretization error at no additional cost, and can easily be generalized to use adaptive time-steps based on this information. Multigrid may also be a sub-optimal choice in this context. The often have excellent convergence in the first several iterations, but then can break down due to anisotropic solution modes that are high-frequency in some directions and

Figure 4. Convergence of the drag coefficient corresponding to the convergence of the residual in Figure 2.
low frequency in others, and are not smoothed well on any grid\footnote{Directional grid coarsening can resolve this issue.}. In the case that a very good initial condition is known (from the previous time-step) multigrid is therefore less effective than Newton iterations.

An example of such an approach, using the third-order Explicit first-Step Diagonally Implicit Runge-Kutta (ESDIRK) method, and a JFNK sub-iteration is given in Lucas et al., 2009, from which Figures 6 and 7 are taken. The test-case is periodic vortex shedding behind an aerofoil at a high angle-of-attack, and the ESDIRK scheme is compared to the accuracy of BDF(2),

---

Figure 6. Time-dependent behaviour of drag over two periods of vortex shedding.

where the former consistently achieves a given accuracy at lower cost.

4. CONCLUSIONS

The field of convergence acceleration is very rich, a consequence of the constant engineering demand for more efficient solvers, and the flexibility available in constructing algorithms (especially where time-accuracy is not demanded). The basic ideas of the most common techniques have been considered here, but this is only the tip of the iceberg. They may be combined in many ways, e.g. multi-stage methods (Section 2.2) with an implicit problem (Section 2.3) solved at each stage (Swanson et al., 2007). Multigrid alone admits infinite variation. The interested reader may start from the references below.
Figure 7. Accuracy of BDF and ESDIRK with varying temporal resolution for the vortex shedding case.


REFERENCES


