Solution Strategies for Integration of Semi-Discretized Flow Equations in elsA and CEDRE.

C. Marmignon, V. Couaillier, B. Courbet

C. Marmignon, V.Couaillier, B. Courbet

(Onera)

E-mail: claude.marmignon@onera.fr

Solution Strategies for Integration of Semi-Discretized Flow Equations in *elsA* and CEDRE

This paper is devoted to the presention of a selection of time integration methods used in the Onera elsA and Cedre software dedicated to the resolution of the compressible Navier-Stokes equations. The selected methods are given in the framework of steady and unsteady flow simulations. Emphasis is put on methods for the resolution of algebraic systems associated with time integration methods for space-discretized equations.

Introduction

Onera is developing two new-generation computational programs for the solution of the compressible Navier-Stokes equations, elsA and Cedre. elsA is more particularly dedicated to the aerodynamics of the flows and Cedre is well adapted to fluid mechanics phenomena in energetics. Both codes are based on similar time integration methods, which allows a unified presentation of the schemes used. We consider here a formulation where the time discretization is separated from the space one. The choice of a time integration algorithm is an essential criterion to ensure efficiency and robustness of numerical simulations and depends on the nature of the flow to compute. In this paper, we are going to classify the methods as steady or unsteady flow.
After space discretization, a system of ordinary differential equations is obtained and a large number of methods are available for the solution this system. Two important families of time integration are developed in our codes. The first one is the family of Runge-Kutta multistage methods and the second one is the family of multi step ones. Only these last ones are investigated in this paper. In the framework of stationary flows, we are interested in the theme of linearization and techniques of approximate factorization. The linear algebraic system is solved by iterative methods, direct methods being prohibitive in terms of memory requirements. A very large number of iterative methods are available in the literature. A presentation of the two main methods implemented in elsA (LU Relaxation) or Cedre (GMRES resolution) is done. The convergence acceleration techniques based on multigrid methods that have been used at Onera for many years for block-structured meshes are also presented.

In the framework of unsteady flow calculations, a dual time-stepping approach in which a steady state with respect to the dual time is approximately reached at each physical time-step is described. Calculation between two physical instants leads to the resolution of a system of pseudo-unsteady equations in dual time and is carried out using convergence acceleration techniques developed within the framework of the steady problems (multi-grid method, local time step, implicit phase with respect to dual time).

ODE system

The purpose of this paper is to present the time integration of the flow equations for numerical simulations of compressible flows as used in the Cedre or Elsa software [19],[6]. We shall describe here only some of the present methods in this software. We are interested in laminar or turbulent flows as as well as in reacting flows. The governing equations are solved in their integral conservation law form using a cell-centered finite volume formulation.

Let Ω_h be a polygonal approximation of the physical domain $\Omega \subset R^d$ (d=1,2,3) made of non-overlapping and non-empty polyhedra. The set of the faces of a polyhedron K is denoted by ∂K and for each face on K, $n_{e,K} \in R^d$ represents the outward unit normal to the face e. Given a face e of K, K_e is the unique polyhedra in Ω_h which shares the same face e with K. Discretizing the conservation equations in Ω_h leads to

$$\int_K \frac{\partial W}{\partial t} dV + \int_{\partial K} \left[F_c(W) - F_v(W, \nabla W) \right].n \, dS = \int_K T(W) dV \qquad (1)$$

where W represents the state vector of conservative variables, F_c and F_v are respectively the convective and diffusive fluxes and T is the source term. The fluxes are discretized in space using some

suitable approximation. The convective flux is discretized for example with flux vector or difference splitting schemes, or space centered scheme plus artificial dissipation. A central differencing is generally used for the viscous terms. The source term is evaluated using variables at cell centres.
We obtain a system of ordinary differential equations, the so-called semi-discrete problem:

$$\frac{d}{dt}\left(W_K|K|\right)+R_K=0 \tag{2}$$

where R_K denotes the residual vector and is defined by

$$R_K=\sum_{\partial K_e}R_{\partial K_e}-T_K|K|=\sum_{\partial K_e}\left(F_c-F_v\right)_{\partial K_e}-T_K|K| \tag{3}$$

If all the mesh point values W_K are gathered in a column vector U, the system can be rewritten

$$\frac{dMU}{dt}+R(U)=0 \tag{4}$$

In the general ALE case, M depends on time, $M=M(t)$. In finite volumes on a motionless mesh, M is typically a constant diagonal or block diagonal matrix. For simplicity, most methods will be described in this particular case.

Time discretization

Explicit and implicit methods

Both explicit and implicit methods are available to integrate the system (4) numerically. The choice of a time integration algorithm is an essential criterion to ensure efficiency and robustness of numerical simulations and depends on the nature of the flow to be computed. For a general system of ordinary differential equations, $R(U)$ is a non-linear function of U. To compute stationary flows, a first-order accurate implicit Euler method is often chosen since time accuracy is not required to reach steady state. On the other hand, fast unsteady flows involving high frequency phenomena require explicit time integration of Runge-Kutta type, for example [10], and do not allow the use of large time steps. For slow unsteady problems, the numerical cost of an unsteady cycle is strongly reduced by the use of implicit methods of integration in time, increasing the numerical stability domain of the schemes and thus allowing the use of large time steps. The difference between explicit and implicit time integration scheme is the time step at which the residual vector R is evaluated.

Multistep Method

Two important families of time integration methods are developed in our codes. The first one is the family of Runge-Kutta multistage methods which allow high orders in time and are explicit. The reader is referred to [11] for details. Another important family of time integration methods is the family of multi step methods, which allow implicit time integration options. The high accuracy can then be achieved by involving multiple time steps. In this framework, we can introduce the particular two-step method (three time levels) in the following form:

$$M\frac{\left[(1+\xi)U^{n+1}-(1+2\xi)U^n+\xi U^{n-1}\right]}{\Delta t}= -\left[\theta R^{n+1}+(1-\theta+\varphi)R^n-\varphi R^{n-1}\right] \tag{5}$$

The incremental form is written:

$$M\frac{\left[(1+\xi)\Delta^nU-\xi\Delta^{n-1}U\right]}{\Delta t}=-\left[\theta R^{n+1}+(1-\theta+\varphi)R^n-\varphi R^{n-1}\right] \tag{6}$$

with $\Delta^nU=U^{n+1}-U^n$. θ,ξ and φ are three parameters allowing control respectively of the implicitness of the method, the order of the finite difference of dU/dt and the number of time levels for R. In our context, we mainly applied the particular schemes with $\varphi=0$. In this case, it can be shown that the schemes are second order accurate for $\xi=\theta-1/2$. For $\xi=0$ and $\theta=1/2$, the resulting scheme is known as the Crank-Nicholson method and is therefore second-order accurate in time.

Classic explicit time integration ($\theta=0$) may be written as

$$M\frac{(1+\xi)\Delta^nU-\xi\Delta^{n-1}U}{\Delta t}=-R^n \tag{7}$$

where R is evaluated at time step n.

Methods for steady flows

The theme of linearization and approximate factorization for the development of implicit methods for compressible Navier-Stokes equation solution has been tackled by many authors. The reader can refer to the retrospective overview by Briley and Mac Donald [5] for more details.

Approximate Jacobian linearizations

In this chapter, we are mainly interested in the numerical simulation of steady flow.

If $\theta\neq0$, $R^{n+1}=R(U^{n+1})$ is evaluated using a Jacobian linearization around the state U^n thanks to a second order Taylor expansion,

$$R^{n+1}=R^n+\Delta^nR=R^n+\frac{\partial R}{\partial U}\Delta^nU+\mathrm{O}\left(\Delta U^2\right). \tag{8}$$

Up to second order,

$$R^{n+1}=R^n+\frac{\partial R}{\partial U}\left(U^n\right)\Delta^nU \tag{9}$$

and the equation (6) becomes

$$\left[M\frac{(1+\xi)}{\Delta t}+\theta\left(\frac{\partial R}{\partial U}\right)^n\right]\Delta^nU=-\left[R^n+\varphi\Delta t\left(R^n-R^{n-1}\right)\right] +M\frac{\xi}{\Delta t}\Delta^{n-1}U. \tag{10}$$

With the exception of Δ^nU, all the quantities are known at time step n.

The left-hand side of (10) is generally a large sparse and non symmetric matrix. This means that many of elegant algorithms for positive definite matrices will not work on equation (10). Also, since equation (10) will be solved many thousands of times, speed is prominent. Since a steady state solution is wanted, it may not be necessary to solve (10) very accurately, since only the converged solution is of any interest. Therefore, an approximate Jacobian is usually applied in the linearization. An approximate Jacobian associated to a first order upwind discretization is often selected. In

the framework of steady flow simulations, the loss in consistency between the space-discretized operators and the approximate Jacobian is no problem. On the other hand, it can have as consequence a reduction of stability range. By choosing a small enough time step, we can note that the matrix in (10) may always be made block diagonal dominant.

For simplicity, the present development uses first-order time differencing ($\xi = 0$) and a two time level space-discretized term R ($\varphi = 0$)

$$\left[M \frac{1}{\Delta t} + \theta \left(\frac{\partial R}{\partial U} \right)^n \right] \Delta^n U = -R^n \tag{11}$$

If $\theta = 1$, the scheme is the backward Euler method. For later discussion, we write

$$A \delta U = -R^n \tag{12}$$

with

$$A = \left[M + \theta \Delta t \left(\frac{\partial R}{\partial U} \right)^n \right] \tag{13}$$

and

$$\delta U = \frac{\Delta^n U}{\Delta t} \tag{14}$$

A is a block-diagonal matrix of 5x5 blocks for tridimensional Euler equations for instance. In practice, the most usual simplifications of the implicit operator are diagonalization of all or part of the blocks of the implicit matrix or factorization (ADI or LU methods).

Diagonalization consists in transforming, wholly or partly, the blocks of the Jacobian matrices occurring in the implicit phase into diagonal matrices. It is generally associated with a factorization.

Approximate factorization methods

The purpose of the techniques of approximate factorization is to simplify the inversion of the matrix system. Indeed, the matrix system cannot be tridiagonal anymore when implicit schemes are applied to multidimensional problems. The factorizations include alternating (ADI) implicit, Lower-Upper (LU) and line relaxation schemes.

The ADI alternate direction technique consists in substituting for the implicit operator a factorized operator along the direction of the grid [17],[18],[2], [4]. Due to the difference between the original matrix and the resulting matrix, upon remultiplication, this procedure introduces errors that can cause a reduction in convergence speed. A modified approximate factorization (MAF) procedure, also called diagonally dominant alternate direction implicit (DDADI), that can regain a part of the convergence rate loss caused by the standard AF is sometimes proposed.

Such techniques are no longer valid for unstructured grids so that the system has to be solved by one of the two large families of methods available for the resolution of the linear algebraic system (4): the direct and the iterative methods. Direct methods are prohibitive in terms of memory requirements and in practice iterative must be considered.

Iterative methods

A very large number of iterative methods are available in the literature. A presentation of the two main methods implemented in elsA (LU Relaxation) or Cedre (GMRES resolution) is described here.

GMRES resolution

Iterative methods aim at building a sequence of iterates $\delta U_0 = -R^n$, δU_1, $\ldots \delta U_\upsilon$ such as $\delta U_\upsilon \to A^{-1} \delta U_0$ when $\upsilon \to \infty$.

GMRES is a nonlinear Krylov method, i.e. it builds a sequence of iterates δU_υ belonging to the Krylov subspace

$$K_\upsilon = span\left\{ \delta U_0, A\delta U_0, \ldots, A^{\upsilon-1} \delta U_0 \right\} \tag{15}$$

More precisely, GMRES finds $\delta U_\upsilon \in K_\upsilon$ so as to minimize the Euclidian norm of the residual $r_\upsilon = A\delta U_\upsilon - \delta U_0$:

- as the dimension of K_υ cannot be larger than the order m of A, $|r_\upsilon| = 0$ when $\upsilon = m$: in exact arithmetics, the method would converge in a finite (though very large !) number of iterates ;
- as the dimension of K_υ grows with υ, the sequence of residuals r_υ is necessarily decreasing, and no residuals oscillation is possible. The minimum principle underlying the method is thus a very sound basis for iterations, which distinguishes GMRES from other Krylov methods.

GMRES is very efficient in terms of CPU time in that it needs only one matrix-vector product per iterate. Of course, computational effort and storage needs increase with the dimension of the Krylov space, which means that it is generally not advisable to use this method with a large Krylov space ($\upsilon \geq 100$ for instance). Nevertheless, the restarted version of GMRES puts an end to the orthogonalization process at a given number υ_{res} of iterates and Krylov vectors, then resumes iterations with a new sequence of υ_{res} of iterates and Krylov vectors etc. Restarted GMRES thus has a storage requirement of υ_{res} Krylov vectors only although at the price of slower convergence.

GMRES is generally applied to a preconditioned system equivalent to the original one. For instance, left preconditioning amounts to multiplying (...) by a matrix B^{-1} such that $B^{-1}A$ is in some sense closer to the unit matrix. In Cedre, B is chosen as the block-diagonal part of A ; although very simple, this preconditioning has the advantage of solving exactly the local part of the implicit system, for instance the contribution of the sources associated with chemical reactions or turbulence.

GMRES with block-diagonal preconditioning turns out very efficient for solving implicit systems from various solvers (multispecies reactive fluid flow, conduction in solids etc). The convergence mechanism of the nested iterative process including time marching and internal GMRES iterates was carefully studied in a PhD thesis [20] :

- at every time step, a few tens of iterations allow a moderate reduction of r_υ typically, $\frac{|r_\upsilon|}{|r_0|} \sim 0.1$ to 0.001 for $\upsilon = 20$ at very large time step Δt. Of course, convergence is much faster for relatively small time steps;
- spectral analysis of convergence shows a dramatic reduction of high frequency components in the course of GMRES iterates. Even though $|r_\upsilon|$ does not converge to very low values, GMRES practically

kills all high frequencies in r_υ, which is sufficient to preserve the stability of implicit time iterates U^n. This distinguishes GMRES from relaxation methods like Jacobi or Gauss-Seidel, for which the reduction rate tends to be the same for all frequencies: for these methods, a huge number of iterates may be necessary to preserve stability at large time steps, specially in presence of dissipative fluxes.

On the other hand, GMRES does have limitations which must be taken into account:

- the nonlinear iterates depend on a small number of global sensors (scalar products and norms, in particular $|r_\upsilon|$) for a very large number of degrees of freedom, which means that residuals in some regions and for some components of U may not be properly taken into account. For physical soundness, the Euclidian norm in the δU space must of course use dimensionless components; in some cases, the choice of scaling may have a significant effect on the quality of the solution. For instance in Reynolds Averaged Navier-Stokes simulations, improper scaling can lead to a poor solution;
- GMRES works well as long as numerical fluxes and sources are sensible, but convergence can be very awkward otherwise. In a low Mach number flow for instance, using a classical compressible numerical flux formula results in a very poor GMRES resolution, whereas a low-Mach number flux ensures efficient iterations;
- In case of preconditioning, time conservativity is not built-in and is only approximate.

GMRES has been used in Cedre in various contexts:

- its primary use is for finding steady asymptotic states in Reynolds Averaged Navier-Stokes or conductive heat transfer simulations…
- ... but it has also been used as the linear solver for time-dependant simulation with Runge-Kutta type implicit schemes [3].

LU relaxation

After trying several relaxation methods in the context of the elsA solver, the LU relaxation is often used to invert the large matrices. The matrix of the implicit stage is then split into its block diagonal (D), block lower (L) and block upper (U) submatrices so that $A = L + D + U$. Each block of D is associated with its own cell. The implicit LU relaxation approximate factorization can be implemented for 3D hybrid structured or unstructured grid.

The resolution of the implicit operator is based on an approximation of the exact matrix $(\mathcal{L}+\mathcal{D}+\mathcal{U})$ by $(\mathcal{L}+\mathcal{D})\mathcal{D}^{-1}(\mathcal{U}+\mathcal{D})$. The system is approximated by a method of relaxation with forward and backward sweeps through the domain. The method sweeps through the mesh from the lower left corner to the right upper corner during the forward sweep. In order to avoid a bias in the iteration scheme and some error accumulations, alternating sweeps in both direction is used. The backward sweep starts at the end-point of the first step. Each relaxation cycle writes in the form of two stages:

$$(\mathcal{L}+\mathcal{D})\Delta U^{(p+1/2)} = -\mathcal{R}^n - \mathcal{U}\Delta U^{(p)} \quad (16)$$

$$(\mathcal{U}+\mathcal{D})\Delta U^{(p+1)} = -\mathcal{R}^n - \mathcal{L}\Delta U^{(p+1/2)} \quad (17)$$

where $p \in [0, p_{\max}]$ and indicates the number of the relaxation cycle.

These two sweeps are repeated several times and $U^{n+1} = U^n + U^{(p_{\max}+1)}$, $p_{\max}$, corresponding to the maximum number of the relaxation cycle. The choice of the number of relaxation cycles must lead to a satisfactory speed of convergence, and a weak value such as two ($p_{\max} = 1$) can prove to be a good compromise. When this number tends towards $+\infty$, the method is convergent for an unfactored matrix with a strictly dominant diagonal and this whatever the initial vector.

In the case of an unstructured grid, a grid reordering algorithm is necessary for efficiency of the LU relaxation method. For structured grids, the LU relaxation sweeps are usually performed by using hyper planes $i + j + k = C^{te}$. The main interest of these sweeps by hyper planes is to order the matrix in lower and upper triangular matrices.

Forward sweep updates point (i, j, k) using already updated values at $(i-1, j, k)$, $(i, j-1, k)$, $(i, j, k-1)$ while backward sweep uses $(i+1, j, k)$, $(i, j+1, k)$ and $(i, j, k+1)$. However, this procedure is particularly well adapted to structured grids and does not extend easily to unstructured grids. In order to get a similar LU relaxation algorithm for unstructured grids, a special grid re-ordering procedure is required. This re-ordering procecedure was proposed by Soetrisno, Imlay and Roberts in [21].

For unstructured grid, the previous reordering of cells allows a clear definition of the lower and upper matrices $\mathcal{L}$ and $\mathcal{U}$ and the proposed algorithm can be applied.

Multigrid method

The multigrid method using a sequence of fine to coarse grids, thus denoted H-multigrid, has been extensively used for practical 3D turbulent flow configurations for many years in block-structured finite volume codes in the CFD community.

This part presents convergence acceleration techniques based on multigrid methods, which have been used at Onera for many years for block-structured meshes. The first developments for the solution of Euler and Navier-Stokes system have been coupled with Lax-Wendroff type schemes [8],[9] based on the multigrid method proposed by Ni [16]. Then cell centered Jameson type schemes were used for complex configurations typical of industrial-type problems, and new multigrid methods proposed by Jameson [12] were implemented at Onera [7] and especially in the elsA software.

The time integration of this system of ordinary differential equations is carried out using multi-stage Runge-Kutta scheme (the Backward Euler scheme corresponds to a one-step scheme). To enhance convergence to steady state, local time stepping as well as implicit residual smoothing is used. In general classical iterative approaches are well adapted for rapidly damping high frequency error components on a given grid. The remaining errors, associated with the smoother low frequency error components, are responsible for the slow convergence. These low frequency error components on the fine grid appear as higher frequencies on the coarser grid. Thus, to enhance faster convergence of the solution to steady state on the fine grid, the multigrid idea is to use the coarser grids to smooth the fine grid low frequency errors on the coarse grids.

Multigrid : Description of the FAS algorithm

The multigrid technique uses a sequence of successively coarser grids to efficiently damp the perturbations. Denoting the grid level by

a subscript, a sequence of grids $h_1, \ldots, h_m, \ldots, h_M$ are then defined, where h_1 denotes the finest grid and h_M represents the coarsest grid.

The multigrid strategy employed is the Full Approximation Storage (FAS) scheme in conjunction with Runge-Kutta time stepping proposed by Jameson. This strategy is used to improve the convergence rate of a multi-block solver for the solution of Euler and Reynolds-Averaged Navier-Stokes equations. The Jameson's FAS algorithm for a simple V-Cycle can then be summarised as follows:

• Compute the residual R_{h_1} and start with a q-stage Runge-Kutta time stepping to update the solution on the finest level h_1.

The following steps are repeated right up to the coarsest level $m = 2, \ldots, M$ which corresponds to the Restriction step :

• Recompute the residual $R_{h_{m-1}}\left(u_{h_{m-1}}\right)$ on the previous level and calculate the modified residual to be transferred from grid level h_{m-1} to the level h_m :

$$R^{(*)}_{h_{m-1}} = R_{h_{m-1}}\left(u_{h_{m-1}}\right) + P_{h_{m-1}} \tag{18}$$

where $P_{h_{m-1}}$ is the added forcing function defined below with P_{h_1} on the finest level.

• Transfer the solution and residual vectors from the previous grid h_{m-1} to the next coarser grid h_m using respectively the fine to coarse transfer operators $T^{h_m}_{h_{m-1}}$ and $\hat{T}^{h_m}_{h_{m-1}}$:

$$\begin{aligned} \overline{u}_{h_m} &= T^{h_m}_{h_{m-1}} u_{h_{m-1}} \\ \overline{R}_{h_m} &= \hat{T}^{h_m}_{h_{m-1}} R^{(*)}_{h_{m-1}} \end{aligned} \tag{19}$$

• Compute the forcing function for the residuals on the grid level h_m which is the difference between aggregated residuals transferred from grid h_{m-1} and the residuals recalculated on h_m :

$$P_{h_m} = \overline{R}_{h_m} - R_{h_m}\left(\overline{u}_{h_m}\right) \tag{20}$$

where $R_{h_m}\left(\overline{u}_{h_m}\right)$ is the residual vector computed on the grid level h_m using the transferred solution vector $\overline{u}_{h_m}$ from the previous grid level h_{m-1}.

• Start Runge-Kutta time stepping on the coarse level h_m using the following reformulated version, to take into account the forcing function as well as to include sub-iterations if necessary, coupled with an implicit smoothing technique (IRS or LU):

$$\begin{cases} u^{(0)}_{h_m} = \overline{u}_{h_m} \; or \; \overline{u}^{(q)}_{h_m} \\ \Delta\tilde{u}^{(1)}_{h_m} = \alpha_1 \dfrac{\Delta t_{h_m}}{\Omega_{h_m}}\left[R_{h_m}\left(u^{(0)}_{h_m}\right) + P_{h_m} \right] \\ \Theta_{h_m} \Delta u^{(1)}_{h_m} = \Delta\tilde{u}^{(1)}_{h_m} \\ u^{\ddot{u}\ddot{u}\ddot{u}\ddot{u}}_{h_m} = u_{h_m} + \Delta u_{h_m} \\ \vdots \end{cases} \tag{21}$$

$$\begin{cases} \vdots \\ \Delta\tilde{u}^{(q)}_{h_m} = \alpha_q \dfrac{\Delta t_{h_m}}{\Omega_{h_m}}\left[R_{h_m}\left(u^{(q-1)}_{h_m}\right) + P_{h_m} \right] \\ \Theta_{h_m} \Delta u^{(q)}_{h_m} = \Delta\tilde{u}^{(q)}_{h_m} \\ u^{(q)}_{h_m} = u^{(0)}_{h_m} + \Delta u^{(q)}_{h_m} \end{cases} \tag{21}$$

Note that upon convergence, when the residual on the finest level goes to zero, the term $R_{h_m}\left(u^{(0)}_{h_m}\right) + P_{h_m}$ in the above equation which can be rewritten as $R_{h_m}\left(u^{(0)}_{h_m}\right) + \left[\overline{R}_{h_m} - R_{h_m}\left(\overline{u}_{h_m}\right) \right]$ goes as well to zero. Thus, no correction is computed on the coarser levels and driven back to the finest level.

• Updated solution on coarse grid h_m :

$$u_{h_m} = u^{(q)}_{h_m} \tag{22}$$

The accumulated corrections from each coarser grid are then successively passed back to finer levels by interpolation $(m = M, \cdots, 2)$. This represents the prolongation step: Transfer the correction from the grid level h_m to the next finer one h_{m-1}.

• Transfer the correction from the grid level h_m to the next finer one h_{m-1}:

$$u^{(+)}_{h_{m-1}} = u_{h_{m-1}} + I^{h_{m-1}}_{h_m}\left(u^{(+)}_{h_m} - \overline{u}_{h_m}\right) \quad \text{avec} \quad u^{(+)}_{h_M} \equiv u_{h_M} \tag{23}$$

where $I^{h_{m-1}}_{h_m}$ is the coarse to fine grid prolongation or interpolation operator from grid h_m to the next finer one h_{m-1} with $u^{(new)}_{h_M} \equiv u_{h_M}$ on the coarsest grid.

The implemented strategies include V and W cycles as well as options for full multigrid (FMG) versions. In the present multigrid approach, only full coarsening algorithms are employed. Thus, a sequence of coarser grids is extracted from the initial given fine grid by deleting every other grid line in each coordinate direction.

Further, the boundary conditions on the coarse grids are treated in the same way as in the fine grid.

Special attention is given to the intergrid transfer operators in the cell-centered formulations in which the variables are located at cell centers. Thus, the transferred variable locations change from one grid to another, which is not the case in a cell vertex or node centered formulation. For the fine to coarse operators, the standard approach is used. The transfer of flow variables conserves mass, momentum and energy by the rule:

$$\overline{u}_{h_m} = \frac{\sum \Omega_{h_{m-1}} u_{h_{m-1}}}{\sum \Omega_{h_{m-1}}} \tag{24}$$

and the residual transferred to grid h_m is the sum of the residuals computed on the eight cells of the fine grid :

$$\overline{R}_{h_m} = \sum R^{(*)}_{h_{m-1}} \tag{25}$$

where the summations range over the cells on the fine grid composing each cell on the coarser grid.

For the coarse to fine operator, in order to damp the high frequency errors, an efficient prolongation operator possessing inherent smoothing properties and well adapted for multiblock computational mesh is introduced. The basic idea is to project the cell centered corrections $\left(u_{h_m}^{(new)} - \overline{u}_{h_m}\right)$, denoted here by the symbol Φ, in a conservative manner to the nodes of the coarse grid by the relation:

$$\Phi_{h_m}^{(Node)} = \frac{\sum_{\{Cell/Node \in Cell\}} \Omega_{h_m}^{(Cell)} \Phi_{h_m}^{(Cell)}}{\sum_{\{Cell/Node \in Cell\}} \Omega_{h_m}^{(Cell)}} \qquad (26)$$

where the numbering of the nodes are given as in figure 1.

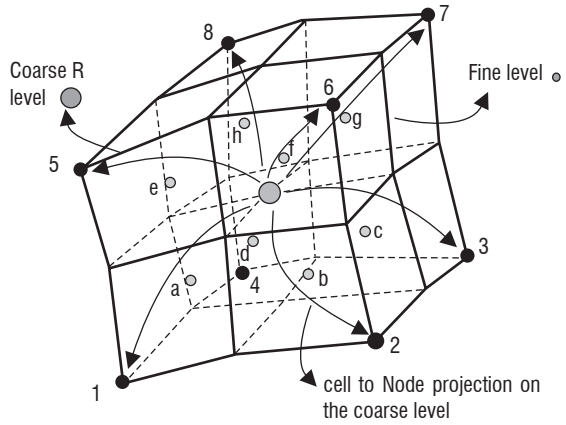


Figure 1 - Coarse to fine inter-grid operator

This conservative smoothing is found to be quite efficient in all our applications. For multi-block computational mesh, the exact interfaces are also taken into account in this cell-to-node projection process. For the four finer (in 2D) cells, which are then exactly included in the coarser grid cell, a volume weighted interpolation is used to compute the cell centered corrections:

$$\Phi_{h_{m-1}}^{(a)} = \frac{\sum_i \Omega_i \Phi_{h_m}^{(i)}}{\Omega} \qquad (27)$$

where Ω, the volume of the coarse grid cell h_m, is the summation of the included fine grid cell volumes

The interpolation coefficients could be based on linear interpolation, inverse distance interpolation or inverse volume interpolation to get better accuracy, but not necessarily better smoothing properties.

For inviscid computations, this leads to an efficient procedure and good convergence properties are obtained for a wide range of 3D applications. Further, to treat complex multi-block configurations with limited number of cells in one direction, the idea of using linear dissipation terms on the coarse grids is also implemented. This consists in using a simple constant coefficient second order dissipation term on the coarser grids instead of the nonlinear artificial dissipation model.

Strategy for turbulent flows

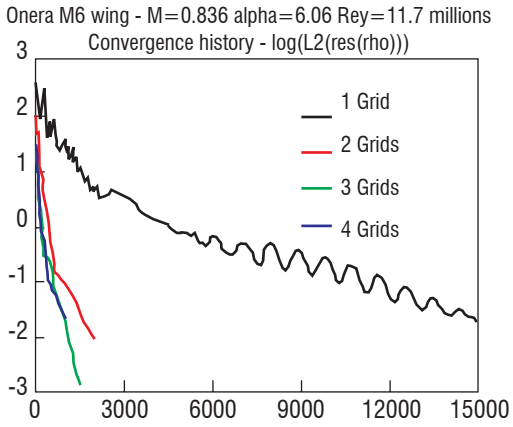
In the case of the RANS equations, the approach adopted is to compute the viscous terms on the coarser grids too. Thus, their influences are also taken into account in the forcing functions on the coarser grids. Different turbulence models are available in the solver, ranging from algebraic models to two equation models. These models are used to compute the turbulent quantities only on the finest grid level. On the coarser grids, they are obtained by interpolating the values from the finest level. This leads to a very direct approach with algebraic models, while with one or two equation models, the corresponding turbulence model equations are solved separately decoupled from the flow equations. In the solver, one Runge-Kutta iteration is carried out to update the turbulent quantities on the fine grid. Thus, different new turbulence models can easily be included in the present environment.

Strategy for multigrid cycles

In order to ensure robustness in V cycles without multigrid on turbulent quantities, sub-iterations are performed on the corresponding equations (let say 2 subiterations on $k-w$ system for a V cycle with 2 or 3 grids). Concerning the present multigrid DG implementation, we have used a P1 approximation on the fine grid and a P0 approximation on the coarse grid, leading to small overcost when using multigrid (20% additional cost per iteration).

Onera M6 wing multigrid computations

a)



b)

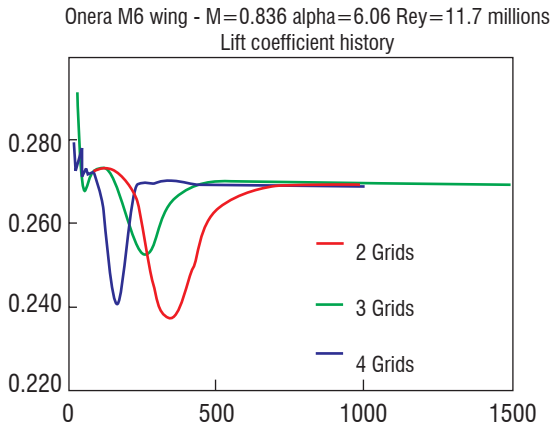


Figure 2 : Onera M6 wing transonic computation – Convergence evolution
a) L2 norm of residual – b) lift coefficient

The Onera M6 wing is a basic 3D test case widely presented in the literature in order to validate numerical methods and turbulence models. The flow field is computed here by the solution of the RANS equation with the Wilcox $k-\omega$ model at a free stream Mach number of 0.836, an angle of attack of 6.06° and a free stream Reynolds number of 11.7 10^6. The C-O mesh used for the computations is

composed of 193x49x65 points, corresponding to $y^+ \approx 1$ almost everywhere on the wing.

Figure 2 illustrates convergence histories of computation performed with the implicit LU method respectively without multigrid (1 grid), and with multigrid with different number of coarse grids (2, 3 and 4 grids). For that test case we have a ratio gain of 5 for a 4-grid computation, knowing than a multigrid cycle is about 25% to 35% more expensive than a single iteration (monogrid cycle).

Time integration for unsteady flow simulations

The majority of flow fields encountered in engineering applications are unsteady. In unsteady applications, some of the most widely used methods of the two step family (6) are the explicit method of Adams-Bashworth ($\varphi = 1/2$), and the implicit Crank-Nicholson ($\theta = 1/2$) and Gear methods ($\theta = 1$, $\xi = 1/2$). The main drawback of explicit schemes lies in the numerical stability limit on Δt, but the CPU cost per iteration is low, since no matrices have to be invert. For implicit methods, large time steps can be applied. Nevertheless, for unsteady flow problems, accuracy requirements tend to restrict the maximum time step.

Dual time stepping

Unsteady flow can also be computed by a dual time-stepping approach [13], [14], [15] in which a steady state with respect to the dual time is approximately reached at each physical time-step. Calculation between two physical instants leads to the resolution of a system of pseudo-unsteady equations in dual time and is carried out using convergence acceleration techniques developed within the framework of the steady problems (multi-grid method, local time step, implicit phase with respect to dual time).

In order to use the dual-time stepping method, an additional term is introduced in the equation

$$M\left(\frac{dU}{dt}\right)^{n+1} + R\left(U^{n+1}\right) = 0 \qquad (28)$$

where for simplicity of presentation we assume $\theta = 1$

This term corresponds to derivative in dual time of the conservative variables. The form of the equation relative to dual time then becomes:

$$M\frac{dU^{n,m}}{d\tau} + \hat{R}\left(U^{n,m+1}\right) = 0 \qquad (29)$$

with

$$\hat{R}\left(U^{n,m+1}\right) = M\frac{dU^{n,m+1}}{dt} + R\left(U^{n,m+1}\right) \qquad (30)$$

Indices m and n are respectively attached to dual time and physical time. The sub-iterations in dual time thus relate to the index m. The term $\hat{R}_K\left(U^{n,m+1}\right)$ represents the unsteady residual. With convergence of sub-iterations ($m \to \infty$), the first term of equation is null and the aerodynamic field satisfies:

$$\hat{R}\left(U^{n,\infty}\right) = \hat{R}\left(U^{n+1}\right) = 0 \qquad (31)$$

Thus, the aerodynamic field indeed corresponds to the unsteady solution of the physical problem at the instant $(n+1)\Delta t$.

In order to speed up the convergence to the pseudo-steady state, scheme can be made implicit with respect to τ. Generally, a simple first-order space discretization is retained to build the implicit stage. If no motion of K is considered, the scheme is written:

$$\left[M\frac{1}{\Delta\tau} + M\frac{(1+\xi)}{\Delta t} + \theta\frac{\partial R\left(U^{n,m}\right)}{\partial U}\right]\Delta^m U = -\left[R\left(U^{n,m}\right) + \varphi\Delta t\left(R\left(U^{n}\right) - R\left(U^{n-1}\right)\right)\right] + \frac{\xi}{\Delta t}M\left(U^{n,m} - U^{n-1}\right) - \frac{(1+2\xi)}{\Delta t}M\left(U^{n,m} - U^{n}\right) \qquad (32)$$

with $\Delta^m U = U^{n,m+1} - U^{n,m}$.

The time accuracy of the method depends on the temporal discretization of the physical time derivative.

Implicitation of Runge-Kutta-Heun methods

Implicit methods can also be of interest in unsteady simulations when unresolved time scales which do not participate in the solution can none the less destabilize explicit calculations. A classical example is low Mach number flows for which we may want to capture unsteady advection while acoustics is of no importance. In that case, the ratio of the convective velocity scale v to the speed of sound c is the Mach number, which tends to zero. For mesh size of order h, the time step to resolve advection is $\sim h/v$ and is much larger than the maximum allowable time step for explicit stability $\sim h/c$.

Gear methods are examples of implicit schemes currently used in unsteady applications. Their main drawback is in being multi step methods that need several initial conditions $\ldots U^{-1}, U^0$. That is why it was decided to develop implicit variants of one step Heun and Runge-Kutta methods.

A simple explicit method for the integration of (4) is

$$M\,\delta U^* = -R(U^n)$$
$$M\,\delta U = -\theta R(U^*) - (1-\theta)R(U^n) \qquad (33)$$

where

$\delta U^* = \dfrac{U^* - U^n}{\Delta t}$, and δU is given by (14).

If $\theta = 1/2$, (33) is second order accurate and is known as Heun's method.

Following the model of linearized backward Euler (12), a possible candidate for the implicitation of (33) is the three parameter method,

$$\left[M + \theta_1\left(\frac{\partial R}{\partial U}\right)^n\right]\delta U = -R^n$$

$$\left[M + \theta_2\left(\frac{\partial R}{\partial U}\right)^*\right]\Delta^n U = -\theta R^* - (1-\theta)R^n \qquad (34)$$

which needs two linear system resolutions per time step. In the same way, Runge-Kutta methods with three or four evaluations have implicit variants with additional parameters (34) and variants are investigated in detail in [3], which studies stability and precision for this large class of methods. This work also defines a method for correcting the approximation of the Jacobian matrix in the implicit system, and show applications to Large Eddy Simulation ■

References

[1] J. BARDINA and J, C.K. LOMBARD - *Three Dimensional Hypersonic Flow Simulations with the CSCM Implicit Upwind Navier–Stokes Method.* AIAA Paper No. 87–1114, 1987.

[2] R. BEAM and R. WARMING - *On the Construction and Application of Implicit Factored Schemes for Conservation Laws.* SIAM-AMS Proceedings; 11:85-129, 1978.

[3] N. BERTIER - *Simulation numérique des grandes échelles en maillage non structuré pour l'aérothermique*. Phd Thesis, Université Pierre et Marie Curie, 2006

[4] W.R. BRILEY and H. MC DONALD - *Solution of the Multidimensional Compressible Navier–Stokes Equations by a Generalized Implicit Method*. J Computat Phys, 24:372-397, 1977)

[5] W.R. BRILEY and H. MC DONALD - *An Overview and Generalization of Implicit Navier-Stokes Algorithms and Approximate Factorization*. Computers & Fluids 30:807-828, 2001.

[6] L. CAMBIER, M. GAZAIX, S. HEIB, S. PLOT, M. POINOT, JP. VEUILLOT, J.-F. BOUSSUGE and M. MONTAGNAC - *elsA Software*. Aerospace Lab issue 2, 2011.

[7] R. COLLERCANDY - *Multigrid Strategy for Euler and Navier-Stokes Computations for Flows Around Complex Aerospace Configurations.* Proceedings of the 4th ECCOMAS, 1998

[8] V. COUAILLIER and R. PEYRET - *Theoretical and Numerical Study of Ni's Multigrid Method.* La Recherche Aérospatiale, English Edition, pp. 9-24, 1985

[9] V. COUAILLIER - *Multigrid Method for Solving Euler and Navier-Stokes Equations in Two and Three Dimensions.* Proceedings of the 8th GAMM Conference, Delft, 1989

[10] G.W. GEAR - *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, New Jersey 1971

[11] C. HIRSCH - *Numerical Computation of Internal and External Flows*. Butterworth-Heinemann Elsevier, Second edition 2007

[12] A. Jameson - *Multigrid Algorithms for Compressible Flow Calculations.* Lecture Notes in Mathematics No 1228, Multigrid II – Proc. Cologne-Springer-Verlag,1985

[13] A. JAMESON - *Time Dependent Calculations Using Multigrid with Applications to Unsteady Flows past Airfoils and Wings.* AIAA Paper No. 91–1259, 1991

[14] N. MELSON, M. SANETRIK and H.L. ATKINS - *Time-Accurate Navier-Stokes Calculations With Multigrid Acceleration.* 6th Copper Mountain Conference on Multigrid Methods, 1993

[15] N. MELSON and M. SANETRIK - *Multigrid Acceleration of Time-Accurate Navier-Stokes Calculations*. 7th Copper Mountain Conference on Multigrid Methods, 1995

[16] R.H. NI - *Multigrid Acceleration of Time-Accurate Navier-Stokes Calculations.* 7th Copper Mountain Conference on Multigrid Methods, 1995

[17] D.W. PEACEMAN and H.H. RACHFORD - *The Numerical Solution of Parabolic and Elliptic Differential Equations*. SIAM Journal; 3:28-41, 1955

[18] R. PULLIAM and D. CHAUSSEE - *A Diagonal Form of an Implicit Approximate Factorization Algorithm*. Journal of Computational Physics,39:347-363, 1981

[19] A. REFLOCH, B. COURBET, A. MURRONE, P. VILLEDIEU, C. LAURENT, P. GILBANK, J. TROYES, L. TESSÉ, G. CHAINERAY, J.B. DARGAUD, E. QUÉMERAIS and F. VUILLOT - *CEDRE Software*. Aerospace Lab issue 2, 2011.

[20] G.SELVA - *Méthodes itératives pour l'intégration implicite des équations de l'aérothermochimie sur des maillages non-structurés*. Phd Thesis, Ecole Centrale Paris, 1998.

[21] M. SOETRISNO, S.T. IMLAY and D.W. ROBERTS - *A Zonal Implicit Procedure for Hybrid Structured-Unstructured Grids.* AIAA Paper No. 94–0645, 1994.

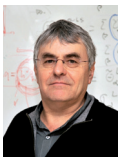
Acronyms

elsA	(Ensemble Logiciel pour la Simulation en Aérodynamique)
CEDRE	(Calcul d'Ecoulements Diphasiques Réactifs pour l'Energétique)
GMRES	(Generalized Minimal RESidual)
LU	(Lower/Upper)
ADI	(Alternating Direction Implicit)

AUTHORS

Claude Marmignon, graduated from ENSH Grenoble. He received a PhD from the Ecole des Mines de Paris in 1987. He is currently senior scientist in charge of the development of numerical methods for complex flows in the CFD and Aeroacoustics Department.

Vincent Couaillier received his dotoral degree Master in Numerical Analysis from the University Pierre et Marie Curie (UPMC) in 1985. He has been working in the CFD & Aeroacoustics Department of Onera for algorithms and software developments. He is presently head of the unit NUMF "Numerical Methods for Fluid Mechanics" in the CFD and Aeroacoustics Department.



Bernard Courbet, graduated from Ecole Centrale de Paris. He is Research engineer, Developer as part of the CEDRE software team.