Multigrid Strategies Coupled with Anisotropic Mesh Adaptation

Victorien Menier, Adrien Loseille, Frédéric Alauzet

Multigrid Strategies Coupled with Anisotropic Mesh Adaptation

Victorien Menier*, Adrien Loseille† and Frédéric Alauzet‡
Gamma3 Team, INRIA Paris-Rocquencourt, 78153 Le Chesnay, France

This paper is an attempt to combine a full multigrid (FMG) algorithm with the classical anisotropic mesh adaptation loop. The FMG algorithm aims at improving the convergence rate of the flow solver, while mesh adaptation tends to improve the tradeoff between CPU time and accuracy of the solution. Coupling these two methods consists, at each stage of the adaptation loop, in running a multigrid simulation using the previously adapted coarser meshes. The main benefits of this coupling are that (i) the convergence of mesh adaptation is improved thanks to the properties stated by the FMG theory, and (ii) anisotropic coarsening is automatically handled by mesh adaptation. First, the implementation and the validation of the multigrid procedures are described, and then the coupling is detailed. Results are presented in 2D and 3D for various flow prescriptions, including a real-life configuration.

Introduction

Multigrid methods have been widely used for algebraic problems since their original development over thirty years ago.[1,2] Interest in these methods has since become even greater, thanks to their ability to efficiently solve problems arising from partial differential equations. For instance, during a CFD simulation using an implicit time integration, a linear system is solved at each iteration of the flow solver. The Newton method used for solving this linear system can be dramatically accelerated using multigrid algorithms, thus improving the convergence rate of the whole simulation.[3,4,5]

Multigrid simulations require a sequence of meshes of different resolutions, and methods of generating these meshes can be classified into three main categories. The simplest manner to generate coarser meshes is to build a hierarchical set of embedded meshes, which presents serious limitations, one of which is that bad quality elements are created during the process, which badly impacts the numerical solution.[7] Another method is the volume agglomeration technique, which consists in agglomerating the finite volume cells of the dual mesh.[8,9] In this paper, we use a third approach: the generation of non-nested unstructured coarse meshes.[10,11,7,12] The metric corresponding to the initial mesh is scaled, and used as an input by an adaptive remesher. Note that stiff problems (shock waves, boundary layers, etc.) can cause a breakdown in efficiency of multigrid methods,[6] due to high anisotropy. This is faced using specific anisotropic coarsening strategies.[13,8,6,11,14]

The full multigrid (FMG) algorithm[7,3] consists in several stages. At each stage, a multigrid simulation is performed, the output solution is linearly interpolated from one stage to the next and used as a restart solution. The complexity of the finest mesh used is increased at each stage. This algorithm presents similarities with the classical mesh adaptation loop, which has proved its efficiency for improving the accuracy of the solution while decreasing the computational wall clock time.[27,28,29,30,21]

This paper presents a coupling of the mesh adaptation loop and the FMG algorithm, both of which can benefit from it. Multigrid algorithms use non-adapted meshes (except in the boundary layer regions), which

*PhD student, Gamma3 Team, recipient of a PhD grant from the Airbus Group Foundation
†Researcher, Gamma3 Team
‡Researcher, Gamma3 Team

are not optimal in terms of sizes and directions, for capturing physical phenomena which are anisotropic or located in small areas of the computational domain. Moreover, anisotropic coarsening can automatically be handled using mesh adaptation. According to the FMG theory,[15] multigrid schemes can improve the convergence of the solution in the mesh adaptation loop. Indeed, it states that, if the solution is fully converged at stage 1, then it is sufficient to converge the solution by one order of magnitude at stages 2, 3, etc. in order to achieve the global convergence on the finest mesh. This is of main interest for the adaptive process, since it states that it is not always necessary to fully converge the solution at intermediary steps of the mesh adaptation loop. Thus, it provides clues on how to reduce the number of solver iterations at intermediary steps, without affecting the accuracy of the final solution.

The paper is organized as follows. In Section I, the implementation of the non-adapted multigrid procedure is presented, together with its validation using 2D and 3D cases. First, the generation of coarser non-nested meshes using isotropic and anisotropic coarsening is discussed. Then, the spatial discretization and the implicit time integration of the modeling equations are described, which lead to solving a linearized system at each flow solver iteration. The multigrid acceleration of the Newton method is then presented, and several multigrid cycles are introduced.

In Section II, the FMG algorithm and the classical mesh adaptation are introduced, and their coupling is detailed. It consists in replacing each single-grid computation in the classical mesh adaptation loop by a multigrid procedure using the previously adapted meshes as coarser meshes. Finally, Section II.4 presents adaptative multigrid simulations, including a 3D industrial configuration.

I. Validation of the Implicit Multigrid Procedure

This section describes how we have implemented and validated the uniform multigrid procedure in `Wolf`, our in-house flow solver. We start by describing the generation of non-nested coarser meshes, which consists in scaling the metric field and using the usual adaptive remesher. The need for anisotropic coarsening is discussed, as well as how it can be done naturally using mesh adaptation. Then, the implementation of the different multigrid schemes used is detailed. Starting from the modeling equations, we introduce the linearized system that is solved after each time iteration and how the Newton method is accelerated and robustified using multigrid. Finally, several numerical examples are provided for validation purposes.

I.1. Uniform Coarse Meshes Generation

This section describes the generation of a hierarchical set of coarser meshes, in the uniform case (i.e. non adapted meshes). Starting from an initial finest mesh $\mathcal{H}_h$ (whose representative edge size is h), we want to generate coarser meshes $\mathcal{H}_{2h}$, $\mathcal{H}_{4h}$, $\mathcal{H}_{8h}$ etc. suitable for a multigrid computation. Various algorithms exist to generate such meshes, including agglomeration techniques[9,8,16] and non-nested methods[10,17,6] (see Figure 2).

We chose to generate unstructured meshes containing simplicial elements (triangles in 2D, surface triangles and tetrahedra in 3D), which are not necessarily nested (see Section I.1.4), i.e. the set of vertices of a coarse mesh $\mathcal{H}_{2h}$ is not necessarily included in the set of vertices of $\mathcal{H}_h$. The use of nested meshes in a multigrid computation is investigated in Section II.4. The coarse mesh generation process is the following: (i) the geometric metric field of $\mathcal{H}_h$ is computed, (ii) a scaling factor is applied to the metric and (iii) an adaptive remesher is used with respect to a set of constraints such as the conservation of the curvature of a surface.

I.1.1. Isotropic Scaling of the Geometric Metric of $\mathcal{H}_h$

At each vertex of a mesh, a metric tensor field prescribes sizes and directions of elements for the mesh generation procedure. More formally, a metric tensor $\mathcal{M}$ in $\mathbb{R}^n$ is an $n \times n$ symmetric definite positive matrix, according to which the scalar product of two vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^n$ is defined as:

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{M}} = \langle \mathbf{u}, \mathcal{M}\mathbf{v} \rangle = {}^t\,\mathbf{u}\mathcal{M}\mathbf{v} \in \mathbb{R}.$$

So, the associated norm of a vector in $\mathbb{R}^n$ is defined as:

$$\|\mathbf{u}\|_{\mathcal{M}} = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle_{\mathcal{M}}} = \sqrt{{}^t\mathbf{u}\mathcal{M}\mathbf{u}},$$

which measures the length of the vector $\mathbf{u}$ in the metric $\mathcal{M}$.

Adapting a mesh $\mathcal{H}$ according to a metric tensor field $\mathcal{M}$ consists in generating a unit mesh $\mathcal{H}_{new}$ with respect to $\mathcal{M}$. A mesh $\mathcal{H}_{new}$ is unit with respect to $\mathcal{M}$, if every one of its edge $\mathbf{ab}$ has a length $\ell_{\mathcal{M}}(\mathbf{ab})$ equal to 1 in $\mathcal{M}$, and if every one of its tetrahedra K has a volume $|K|_{\mathcal{M}}$ in $\mathcal{M}$ equal to $\sqrt{2}/12$. The length and the volume in $\mathcal{M}$ are given by:

$$\begin{aligned} \ell_{\mathcal{M}}(\mathbf{ab}) &= \int_0^1 \sqrt{{}^t\mathbf{ab}\, \mathcal{M}(\mathbf{a}+t\mathbf{ab})\, \mathbf{ab}}\, \mathrm{d}t \\ |K|_{\mathcal{M}} &= \int_K \sqrt{\det \mathcal{M}}\, \mathrm{d}K \end{aligned}$$

The first step of mesh coarsening is to compute the metric $\mathcal{M}_{geo}(\mathcal{H}_h)$, according to which $\mathcal{H}_h$ is unit. In order words, $\mathcal{M}_{geo}(\mathcal{H}_h)$ reflects the sizes and anisotropy of the elements of $\mathcal{H}_h$. $\mathcal{M}_{geo}(\mathcal{H}_h)$ is then multiplied by a scaling factor: 1/4 for $\mathcal{H}_{2h}$, 1/16 for $\mathcal{H}_{4h}$ etc. Using a local mesh operator, the set of coarse meshes is then obtained with respect to these scaled metrics.

I.1.2. Adaptive Remesher

Given $\mathcal{H}_h$ and a scaled metric $\mathcal{M}_{nh}$, vertex insertions and edge collapses are performed recursively until all the edges have a length in the interval $[\sqrt{2}, 1/\sqrt{2}]$ with respect to $\mathcal{M}_{nh}$. Then, an optimization procedure is applied, based on vertex smoothing, edges and faces swapping. During this optimization step, we try to improve the overall quality of the mesh (as opposed to the unit mesh step where the convergence to a prescribed length distribution is the main concern). The quality in a metric $\mathcal{M}$ is given by:

$$Q_{\mathcal{M}}(K) = \frac{36}{3^{\frac{1}{3}}} \frac{|K|_{\mathcal{M}}^{\frac{2}{3}}}{\sum_{i=1}^{6} \ell_{\mathcal{M}}^2(\mathbf{e}_i)} \in [0,1],$$

and a "perfect" element has a quality equal to 1.
All the aforementioned mesh modification operators are based on a unique cavity-based operator revisited in an anisotropic context.[18] This operator also performs surface-based operations. In such a case, a local surface approximation is computed based on a user-provided background surface mesh and metric. This allows us to limit, according to an input parameter, the approximation error of the surface.

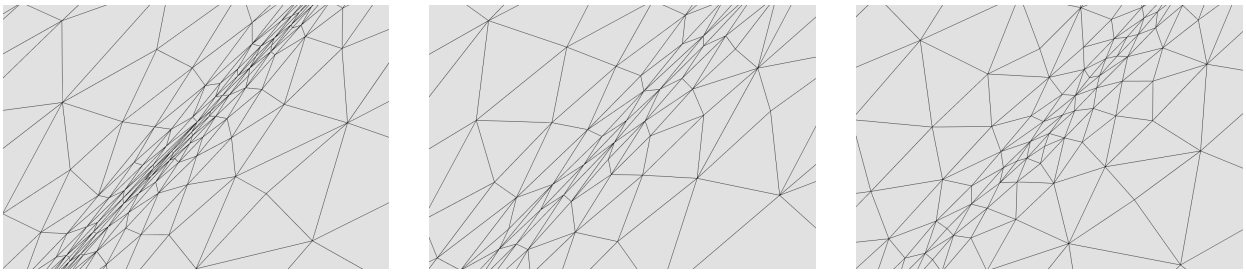

Figure 1. Example of mesh coarsening. Left: initial mesh. Middle: Isotropic coarsening. Right: Anisotropic coarsening.

I.1.3. Anisotropic Coarsening

A breakdown in efficiency of the multigrid methods can be observed when dealing with high anisotropy. Turbulent viscous flow simulations, for instance, require mesh elements whose maximum aspect ratio can be in the order of 10^{-6} in the boundary layer regions. Anisotropic elements are also necessary to accurately

capture shock waves during transonic simulations. In order to prevent this breakdown in efficiency, existing isotropic coarsening techniques[10,11] were extended to the anisotropic case.[13,8,6,11,14] The general idea is to coarsen the mesh only in the directions that are perpendicular to a direction of anisotropy. To do so, only the scaling of the geometrical metric $\mathcal{M}_{geo}(\mathcal{H}_h)$ differs from the isotropic case.

We chose to compute the anisotropic scaled metric tensor field of $\mathcal{M}_{geo}(\mathcal{H}_h)$ as follows[14] (in 3D). Let $\mathcal{M}_k$ be the metric tensor of $\mathcal{M}_{geo}(\mathcal{H}_h)$ associated to a vertex v_k, and $\lambda_1, \lambda_2, \lambda_2$ its eigenvalues, and h_1, h_2, h_3 the corresponding sizes ($h_i = (\lambda_i)^{-1/2}$). We start by ordering these sizes.

$$h_1 \leq h_2 \leq h_3 \,.$$

Then, the new coarsened sizes are computed:

$$h_{i,coarse} = \max(h_i, \min(c \times h_i, h_{i-1})) \,,$$

where c is the coarsening factor. This defines the new coarsened eigenvalues: $\lambda_{i,coarse} = h_{i,coarse}^{-1/2}$. This algorithm will produce the same coarse metric in the isotropic regions of the mesh, as in the anisotropic regions, only the directions where the mesh size is minimal are scaled.

The mesh adaptation process automatically provides a kind of anisotropic coarsening, thanks to the considered L^p error estimate. This is one of the advantages of coupling it with an FMG algorithm. We recall[19] that the convergence order on a discontinuity is $1/p$ in L^p norm. In other words, dividing the mesh size around a discontinuity by a factor s will produce an error on the discontinuity divided by $s^{\frac{1}{p}}$. If, however, the mesh size is divided by a factor $s = 2$ everywhere but around the discontinuity, and divided by a factor $s = 2^{2p}$ in the direction of the discontinuity, then the error will be divided by a factor 4 and for a cost of $N \to 2N + 2^{2p}$.

I.1.4. Generation of Nested Meshes

Generating embedded meshes is a simple way to build a hierarchical nested set of coarse meshes for multigrid, such as depicted in Figure 2. It consists in first generating an initial coarse mesh, which is then refined by element subdivision. This method has a major inconvenience, since the quality of the meshes generated decreases as they are refined. Indeed, as element subdivisions are iteratively performed, patterns corresponding to the coarsest mesh elements appear. These patterns may influence the computation, as they can act as artificial internal boundaries,[7] as illustrated in Figure 3.

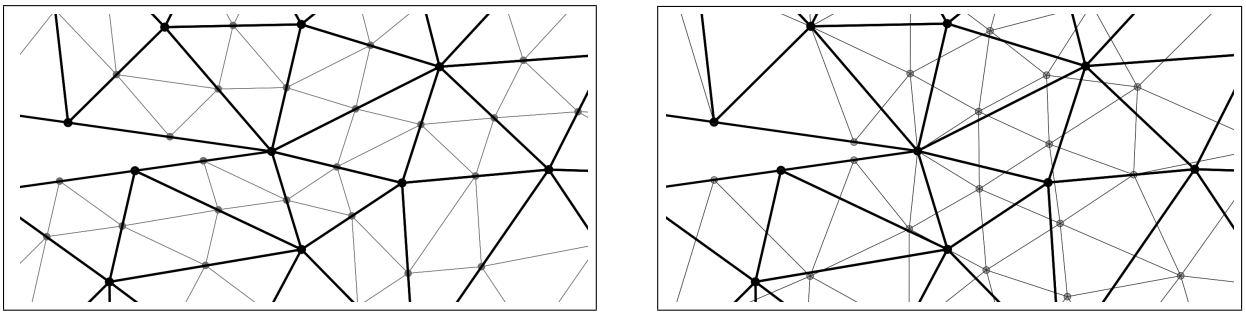

Figure 2. Comparison between nested and non-nested meshes. The coarse mesh $\mathcal{H}_{2h}$ (in black) and the finer mesh $\mathcal{H}_h$ (in grey) are juxtaposed for visualization purposes. Left: $\mathcal{H}_h$ and $\mathcal{H}_{2h}$ are nested. Right: Non-nested meshes.

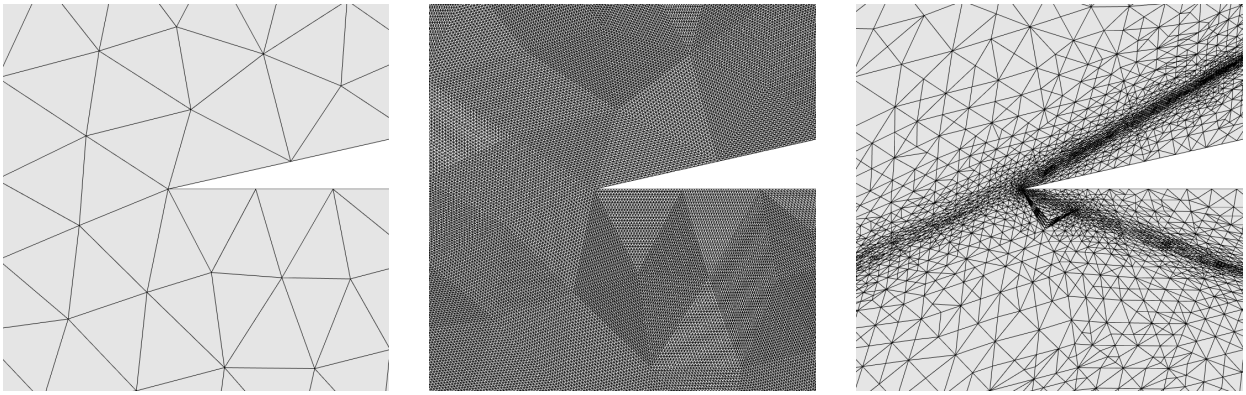

Figure 3. Close-up views of three nested meshes of a 2D scramjet. Left: the initial coarse mesh. Middle: all the edges of the mesh were refined. Right: only some edges were refined. Both nested meshes (middle and right) contain elements of bad quality due to the constraints from the initial discretization.

I.2. Spatial Discretization and Implicit Time Integration in `Wolf`

This section introduces the linearized system that is to be solved after each time iteration. More details are given in.[20] The modeling equations we consider are the Euler equations, which read:

$$\left\{\begin{array}{rcl} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) & = & 0\,, \\ \dfrac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p & = & 0\,, \\ \dfrac{\partial (\rho e)}{\partial t} + \nabla \cdot ((\rho e + p)\mathbf{u}) & = & 0\,, \end{array}\right.$$

where ρ is the density, p the pressure, $\mathbf{u} = (u, v, w)$ the velocity vector, e the total energy per mass unit, and $E = \rho e$ the total energy per volume unit.
This system can be written in vectorial form:

$$W_t + F_1(W)_x + F_2(W)_y + F_3(W)_z = 0 \tag{1}$$

where W is the nondimensionalized conservative variables vector:

$$W = (\rho, \rho u, \rho v, \rho w, \rho E)^T\,,$$

and $\mathbf{F}(W) = (F_1(W), F_2(W), F_3(W))$ are the convective (Euler) flux functions:

$$\begin{array}{rcl} F_1(W) & = & (\rho u,\ \rho u^2 + p,\ \rho uv,\ \rho uw,\ u(\rho E + p))^T \\ F_2(W) & = & (\rho v,\ \rho uv,\ \rho v^2 + p,\ \rho vw,\ v(\rho E + p))^T \\ F_3(W) & = & (\rho w,\ \rho uw,\ \rho vw,\ \rho w^2 + p,\ w(\rho E + p))^T. \end{array}$$

The spatial discretization of the fluid equations is based on a vertex-centered finite element/finite volume formulation on unstructured meshes. It combines HLLC upwind schemes for computing the convective fluxes and second order space accuracy is achieved through a piecewise linear interpolation based on the Monotonic Upwind Scheme for Conservation Law (MUSCL) procedure which uses a particular edge-based formulation with upwind elements. A specific slope limiter is employed to damp or eliminate spurious oscillations that may occur in the vicinity of discontinuities.[21]

The implicit time discretization of (1) reads:

$$\frac{|C_i|}{\delta t_i^n}\left(W_i^{n+1} - W_i^n\right) = R_i(W^{n+1})$$

which is linearized as:

$$\left(\frac{|C_i|}{\delta t_i^n} I_d - \frac{\partial R_i}{\partial W}(W^n)\right)\left(W_i^{n+1} - W_i^n\right) \;=\; R_i(W^n)$$

where $\frac{\partial R_i}{\partial W}(W^n)$ contributes the i^{th} line of the matrix and using a set of linearizations.[20] For the sake of clarity, we rewrite the linearized system in a more compact form:

$$\mathbf{A}^n\,\delta\mathbf{W}^n = \mathbf{R}^n \tag{2}$$

where

$$\mathbf{A}^n = \frac{|C|}{\delta t^n}\mathbf{I} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{W}} \quad \text{and} \quad \delta\mathbf{W}^n = \mathbf{W}^{n+1} - \mathbf{W}^n\,.$$

During an implicit simulation, this linearized system (2) is being solved at each flow solver iteration. Thus, accelerating the Newton method can dramatically decrease the total CPU time of the whole simulation. Moreover, an unsufficient convergence of the Newton method can cause a bad convergence of the whole simulation. We now describe how the linear system is solved using the symmetric Gauss-Seidel (SGS) relaxation, and then we present the implementation of multigrid methods which improve both the convergence speed and the robustness.

I.3. Solving the Linear System Using the SGS Relaxation

The linearized system (2) is solved after each time advancing using the approach based on Lower-Upper Symmetric Gauss-Seidel (LU-SGS) implicit solver initially introduced by Jameson[22] and fully developed by Sharov et al. and Luo et al.[23,24,25,26] The Newton method we use is the symmetric Gauss-Seidel (SGS) relaxation and it is iterated until the residual of the system is reduced by a chosen order of magnitude (usually 0.01) with respect to the initial residual.

We now briefly describe the SGS relaxation. We start by decomposing $\mathbf{A}$ into a strictly lower triangular component $\mathbf{L}$, a diagonal component $\mathbf{D}$, and a strictly upper triangular component $\mathbf{U}$:

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}\,.$$

The linear system (2) can be rewritten :

$$(\mathbf{D}+\mathbf{L})\mathbf{D}^{-1}(\mathbf{D}+\mathbf{U})\,\delta\mathbf{W}^n = \mathbf{R}^n + (\mathbf{L}\mathbf{D}^{-1}\mathbf{U})\,\delta\mathbf{W}^n\,,$$

and approximated by the following system:

$$(\mathbf{D}+\mathbf{L})\mathbf{D}^{-1}(\mathbf{D}+\mathbf{U})\,\delta\mathbf{W}^n = \mathbf{R}^n\,.$$

Using an SGS relaxation to converge the Newton method consists in performing k_{max} SGS sub-iterations using forward and backward sweeps:

$$\begin{aligned}(\mathbf{D}+\mathbf{L})\,\delta\mathbf{W}^{k+1/2} &= \mathbf{R} - \mathbf{U}\,\mathbf{W}^k \\ (\mathbf{D}+\mathbf{U})\,\delta\mathbf{W}^{k+1} &= \mathbf{R} - \mathbf{L}\,\mathbf{W}^{k+1/2}\,.\end{aligned}$$

which can be rewritten point wise:

$$\begin{aligned}\delta W_i^{k+1/2} &= D_{ii}^{-1}\Big(R_i - \sum_{j\in\mathcal{L}(i)} L_{ij}\delta W_j^{k+1/2} - \sum_{j\in\mathcal{U}(i)} U_{ij}\delta W_j^{k}\Big) \\ \delta W_i^{k+1} &= D_{ii}^{-1}\Big(R_i - \sum_{j\in\mathcal{U}(i)} U_{ij}\delta W_j^{k+1} - \sum_{j\in\mathcal{L}(i)} L_{ij}\delta W_j^{k+1/2}\Big)\,.\end{aligned}$$

k_{max} SGS sub-iterations are performed, unless fewer iterations are needed to decrease the residual of the system by the desired order of magnitude.

This section has described how the linear system (2) is solved using an SGS relaxation in the case of a single-grid computation. We now present how we implemented multigrid methods, which use coarser meshes to accelerate and robustify the convergence of the Newton method.

I.4. Multigrid Methods for Accelerating the Convergence of the Newton Method

This section describes the multigrid procedures for converging the Newton method. Multigrid methods require a sequence of N meshes :

$$\mathcal{H}_h, \mathcal{H}_{2h}, \mathcal{H}_{4h}, \ldots, \mathcal{H}_{2Nh} ,$$

where $\mathcal{H}_h$ is the initial (and finest) mesh, and $(\mathcal{H}_{2ih})_{(i=1,N)}$ are the coarsened versions of $\mathcal{H}_h$ generated as previously explained in Section I.1. For the sake of clarity, we rewrite the linear system (3) :

$$A_h \delta u_h = F_h . \tag{3}$$

where A_h is the matrix of the linearized system built on $\mathcal{H}_h$ and F_h is the right hand side (RHS). The residual of (3) is given by

$$r_h = A_h \delta u_h - F_h .$$

In the case of a single-grid computation, we have seen that k_{max} (or less) SGS sub-iterations are performed on $\mathcal{H}_h$ in order to reduce r_h by a desired order of magnitude. In the case of a multigrid computation, n_{max} multigrid cycles are performed. One multigrid cycle consists in (i) performing one SGS sub-iteration on $\mathcal{H}_h$, (ii) computing a correction using the coarser meshes, and (iii) adding the computed correction to the solution on $\mathcal{H}_h$. The way this correction is computed depends on the number of coarse meshes involved and on the type of the multigrid cycle used.

Although one multigrid cycle is more costly in terms of CPU (than one single-grid sub-iteration), the number of cycles required to reach the targeted residual is expected to be smaller, thanks to the corrections. Note that the smaller the number of vertices of the coarsest mesh is, the quicker the correction is computed. Moreover, coarser meshes have a strong smoothing property, which increases the robustness, i.e. using a multigrid procedure makes it possible to reduce the residual by some orders of magnitude that could not be reached using one single mesh. To summarize, we use a multigrid cycle to compute a correction at each sub-iteration of the Newton method, in order to (i) increase the convergence speed, while (ii) improving the robustness.

We now describe the three different types of multigrid cycles we use to compute the corrections: the V-cycle , the W-cycle and the F-cycle. We start by explaining the case of the two-grid V-cycle, which only requires one coarser mesh. Then, the three types of cycles are introduced in the general case of N meshes.

I.4.1. Basic Two-grid V-cycle

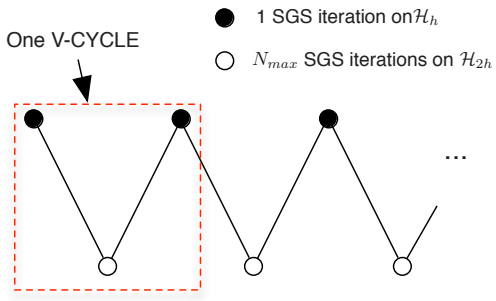

Figure 4. Bigrid V-cycle

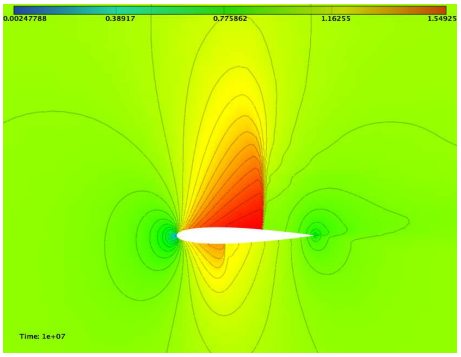

Figure 5. 2D transonic NACA 0012: pressure.

The two-grid V-cycle requires a mesh $\mathcal{H}_h$ and a coarser mesh $\mathcal{H}_{2h}$. We suppose that N time iterations were performed by the flow solver. Let

$$A_h \delta u_h = F_h$$

be the linearized system obtained after the N-th time iteration. In order to accelerate the convergence of the Newton method for solving this linear system, a given number of multigrid cycles can be performed. The

bigrid V-cycle (see Figure 4) consists in computing a correction by performing several SGS iterations on $\mathcal{H}_{2h}$.

A_h, the matrix of the linearized system, was built on $\mathcal{H}_h$ as explained in Section I.3. A_{2h} was built in a similar way on $\mathcal{H}_{2h}$ after $\mathcal{S}_h$, the solution obtained on $\mathcal{H}_h$ after N time iterations, was linearly interpolated to $\mathcal{H}_{2h}$. Starting from an initial solution δu_h^0, a pre-smoothing is performed on $\mathcal{H}_h$, i.e. one SGS iteration. Note that when the multigrid cycle is the first of the current time iteration, δu_h^0 is set to $\mathbf{0}$, and otherwise δu_h^0 is the solution of the previous cycle. Le δu_h^1 be the solution obtained after the pre-smoothing, the residual is computed:

$$r_h = A_h \delta u_h^1 - F_h$$

and is restricted to $\mathcal{H}_{2h}$. The restriction operator $R_{h\to 2h}$ first consists in locating each vertex P_h of $\mathcal{H}_h$ in $\mathcal{H}_{2h}$, i.e. identifying the element $K_{2h} = (P_{2h}(i))_{i=0,3}$ of $\mathcal{H}_{2h}$ containing P_h. Then, the restricted residual is summed to the vertices of K_{2h}:

$$R_{h\to 2h}(r_h)(P_{2h}(i)) += \beta_i \times r_h(P_h) \quad \text{for} \quad i = 0, 3\,,$$

where β_i is the barycentric coordinate of P_h in K_{2h} associated to $P_{2h}(i)$.

The correction c_{2h} is then computed on $\mathcal{H}_{2h}$ by using $R_{h\to 2h}(r_h)$ as the source term. The initial correction is set to 0 : $c_{2h}^0 = \mathbf{0}$ and a given number of SGS iterations is performed:

$$A_{2h} c_{2h}^0 = R_{h\to 2h}(r_h) \tag{4}$$

Then, the resulting correction c_{2h}^1 is linearly interpolated to $\mathcal{H}_h$ and added to the solution:

$$\delta u_h^2 = \delta u_h^1 + I_{2h\to h}(c_{2h}^1)\,.$$

So, at each time iteration of the flow solver, the corrections added after each multigrid cycle are expected to improve the convergence of the Newton method, and thus to improve the convergence of the whole simulation. The number of multigrid cycles required to reach the targeted residual is expected to be smaller than the required number of SGS iterations on $\mathcal{H}_h$ in the single-grid case. The fastest convergence in terms of the number of iterations can be reached using an ideal bigrid V-cycle.

Ideal Bigrid V-cycle. The ideal bigrid V-cycle consists in performing as many SGS iterations on $\mathcal{H}_{2h}$ as are necessary to reduce the residual of the linear system (4) as much as possible, and thus obtain a fully converged correction c_{2h}^1. This ideal bigrid cycle is obviously too costly in terms of CPU to be used during a real-life simulation, but since the linear system is converged to its maximum on $\mathcal{H}_{2h}$, it provides the best correction that can be obtained using a multigrid cycle. The number of iterations needed by an ideal bigrid cycle to converge the linear system on the finest mesh $\mathcal{H}_h$ can thus be targeted when using another multigrid cycle (using more mesh levels). In other words, a "good" multigrid cycle aims at requiring as few iterations as the ideal bigrid cycle to decrease the residual on $\mathcal{H}_h$, while being less costly in terms of CPU thanks to the use of more coarser mesh levels.

I.4.2. N-grid V-Cycle, W-Cycle and F-Cycle

The N-grid V-Cycle is simply the extension of the bigrid V-cycle to N grids. Only one SGS iteration is performed on $\mathcal{H}_{2h}$ and the residual

$$r_{2h}^1 = A_{2h} c_{2h}^1 - r_{2h}$$

is computed and restricted to $\mathcal{H}_{4h}$. $R_{2h\to 4h}$ is then used as the source term to compute a correction C_{4h}^1 on $\mathcal{H}_{4h}$. This is how a correction is computed on each coarse mesh. Once on the coarsest mesh $\mathcal{H}_{2Nh}$, not one but several SGS iterations are used to compute C_{2Nh}^1, which is not costly in terms of CPU due to the low number of vertices. Then, the correction of the coarsest mesh is interpolated and added to the correction of the second coarsest mesh and so on. In the end, the final correction containing all the contributions of the coarser meshes is interpolated on the finest mesh and added to δu_h^1. A post-smoothing (i.e. one SGS iteration) can be performed on each level i after $I_{2(i+1)h\to 2ih}(c_{2(i+1)h})$ has been added to c_{2ih}.

Other types of multigrid cycles may be used, such as the V-, the W- and the F-cycle. The structures of these three cycles is depicted in Figure 6 for the case of four grids, and a 5-grid W-cycle is depicted in Figure 7.

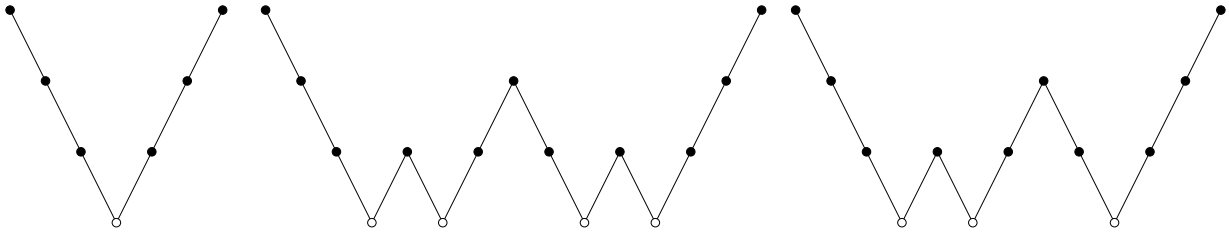

Figure 6. Four-grid methods : V-cycle, W-cycle and F-cycle (• : 1 smoothing SGS iteration, ○ : Several SGS iterations)

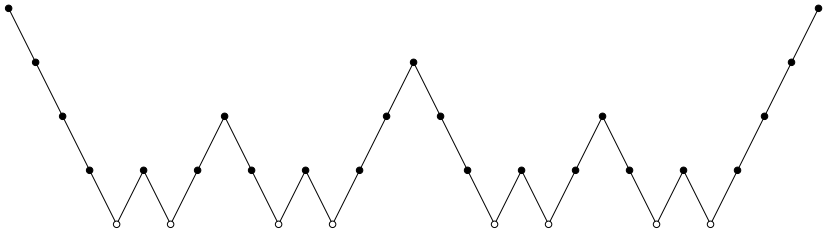

Figure 7. A five-grid method: W-cycle (• : 1 smoothing SGS iteration, ○ : Several SGS iterations)

I.5. Uniform Multigrid Results

Two different comparisons were made in order to measure the impact of multigrid methods. We compared (i) the convergence of the linear system at a given time iteration, and (ii) the convergence of the residual of the whole simulation from the first to the last iteration in time. Three cycles were used (V-cycle, W-cycle and F-cycle), as well as several numbers of coarse levels.

I.5.1. Resolution of the Linear System

The convergence of the linear system at a given time iteration is compared. A solution was previously "almost" converged on the finest mesh by performing N time iterations using any method (single-grid or multigrid) and using an adequate CFL law. The evolution of the residual of the resolution of the linear system obained after the N-th iteration is then compared for (i) the single-grid method, (ii) the ideal bigrid, and (iii) V-, W- and F-cycles using various numbers of mesh levels. A high CFL is prescribed in order to evaluate the robustness of each method.

The first example considers a 2D transonic NACA 0012 airfoil. The Mach number is $M = 0.8$ and the angle of attack $\alpha = 1.25$. A solution (see Figure 5) is precomputed on the finest mesh using 120 time iterations at $\text{CFL}_{max} = 10$. Then, the resulting solution is used as a restart solution and one iteration at CFL = 1000 is performed using the different methods. Views of the four meshes used during multigrid simulations are depicted in Figure 8. Figure 9 presents the convergence rates obtained using one single-grid, an ideal bigrid, and 3 V-cycles (3, 4 and 5 meshes). All the multigrid methods manage to decrease the initial residual by twelve orders of magnitude, while in the same CPU time interval, the single-grid computation fails to decrease it by one order due to the high CFL. As expected, the ideal bigrid shows the fastest convergence in terms of the number of iterations but is also the slowest method in terms of CPU. Figure 10 presents a comparison of the three different 4 grid cycles used (V, W and F). Although both the W-cycle and the F-cycle are really close to the ideal bigrid in terms of the number of iterations, they are slower than the V-cycle. To summarize, the fastest convergence for the transonic NACA is the 4-grid V-cycle in terms of CPU, and the 4-grid F-cycle in terms of the number of iterations.

I.5.2. Impact on the Whole Simulation

The evolution of the residual after each time iteration is compared for the single-grid method and several multigrid cycles. Starting from a uniform solution, the number of time iterations needed to reach a targeted

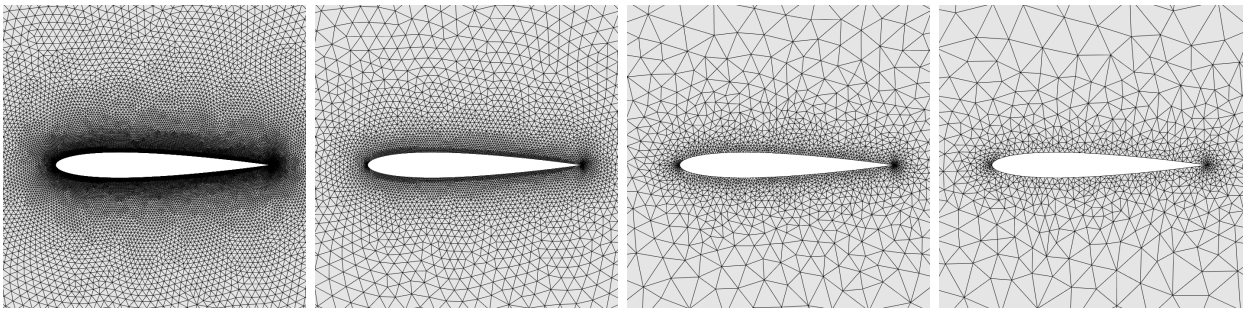

Figure 8. Close-up views of the four meshes used during the multigrid computations of the 2D transonic NACA. Number of vertices, from left to right: 29024, 7379, 2499 and 1305.

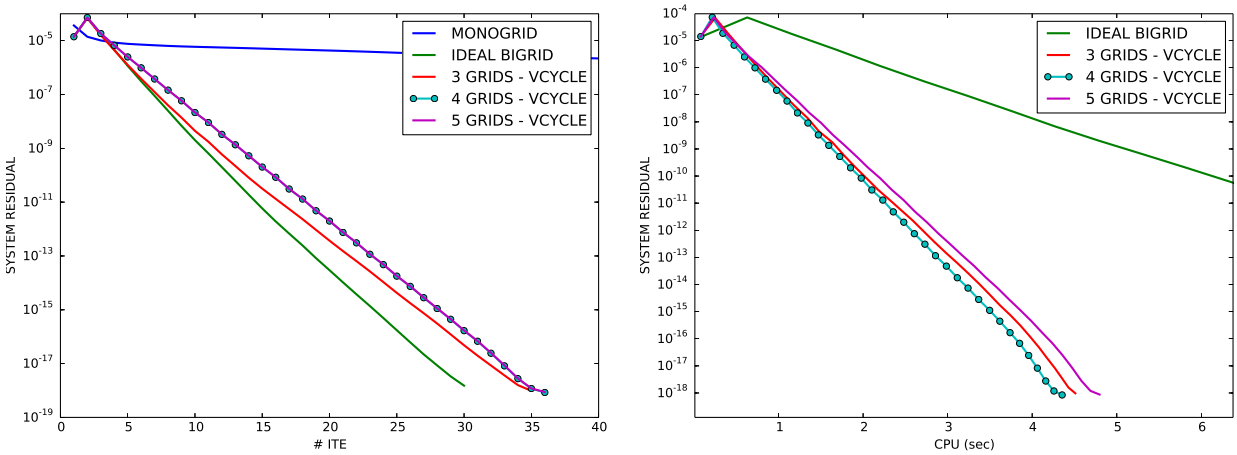

Figure 9. Transonic NACA: Comparison of the V-cycles for the convergence of the Newton method after 120 time iterations.

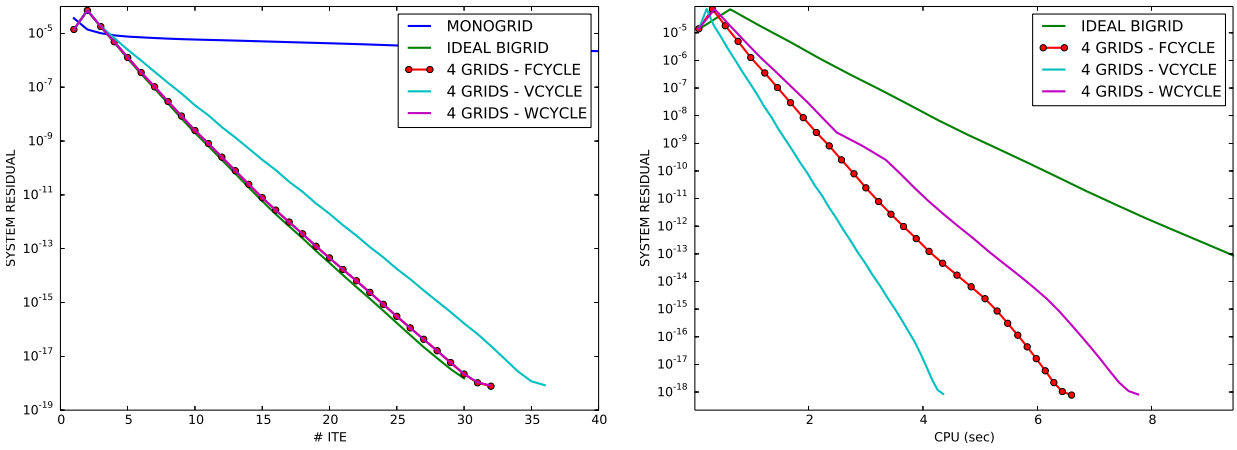

Figure 10. Transonic NACA: Comparison of the 3 cycles (4 grids) for the convergence of the Newton method after 120 time iterations.

residual is compared for three examples : a 2D transonic NACA 0012, a 3D subsonic NACA 0012 and a 3D transonic wing body tails (WBT) configuration.

2D transonic NACA 0012. Section I.5.1 presented how multigrid methods help to converge the Newton method at a given time iteration for the 2D transonic NACA 0012. This time, the convergence of the residual

in terms of the number of flow solver iterations is considered for the same test case. Figure 11 presents a comparison between the single-grid and the multigrid methods. Multigrid methods improve the convergence rate in terms of both the number of iterations and wall clock time. As regards the number of iterations, the best convergence rate is obtained in the single-grid case by performing 40 SGS sub-iterations, and in the multigrid case using a 3-grid V-cycle. The fastest methods in terms of CPU are the 3-grid and 4-grid V-cycles.

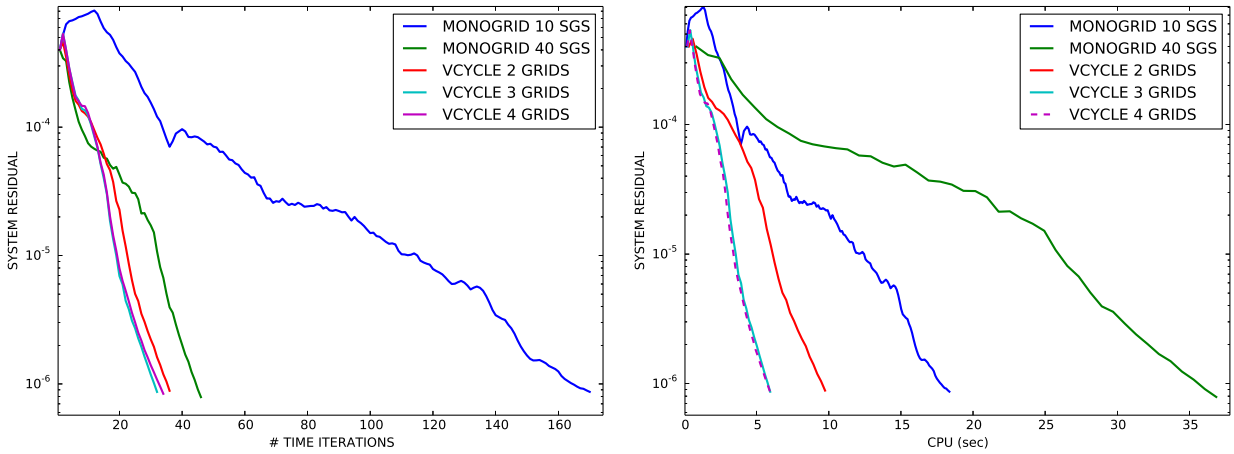

Figure 11. 2D transonic NACA: Convergence of the residual of the whole simulation. Left: number of time iterations. Right: wall clock time (sec).

3D subsonic NACA 0012. This example considers a subsonic flow around an extruded NACA 0012 geometry. The meshes used for the multigrid computation are depicted in Figure 12 and the final solution in Figure 14. This example is interesting because the convergence of the residual of the whole simulation greatly depends on the Newton method. As shown in Figure 13, no fewer than 25 SGS sub-iterations are required in the single-grid case to reduce the residual of the whole simulation by the desired order of magnitude (10^{-9} is the target). Using a multigrid method, however, Figure I.5.2 shows that only one V-cycle is enough, and that performing two cycles is enough to obtain an optimal residual convergence in terms of the number of iterations, i.e. performing more than two cycles does not help to increase the convergence rate. Figure I.5.2 also presents a comparison between the most efficient single-grid method (i.e. 25 SGS-sub-iterations), and the multigrid. As concerns the number of iterations, the residual convergence of the optimal single-grid method (25 SGS sub-iterations) and the optimal multigrid method (2 V-cycles) are identical. The wall clock time, however, drops from 6m50s in the single grid case to 1m54s using one V-cycle.

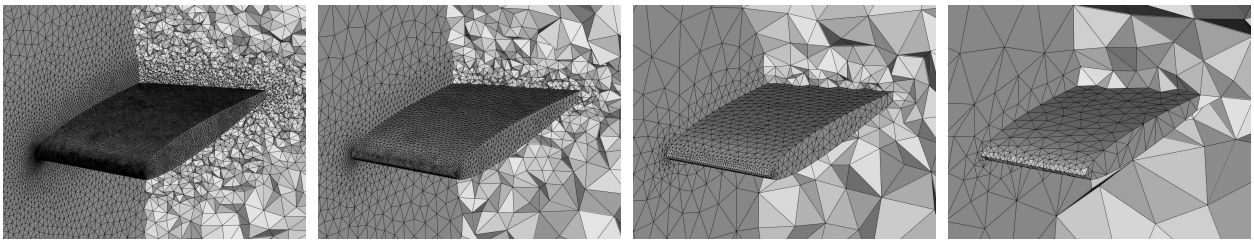

Figure 12. 3D subsonic NACA 0012 : Cuts in the volumes of the four meshes used during the multigrid computations.

3D transonic WBT configuration. A transonic flow is computed over a 3D wing body tails (WBT) configuration geometry depicted in Figure 17, which includes a wing, a body and two tails (horizontal and vertical). The prescribed Mach number is $M = 0.8$ and the angle of attack $\alpha = 1$ degree. The convergence of the residual is compared for a single-grid computation and a 3-grid V-cycle. The three meshes used for the computation are presented in Figure 16.

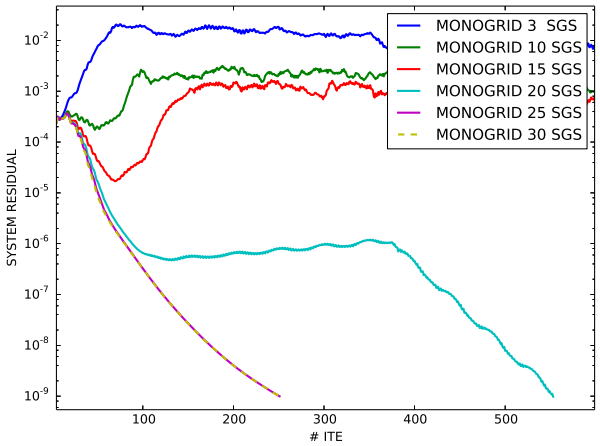

Figure 13. 3D subsonic NACA 0012: At least 25 SGS sub-iterations are required in the single-grid case to converge the residual of the whole simulation.

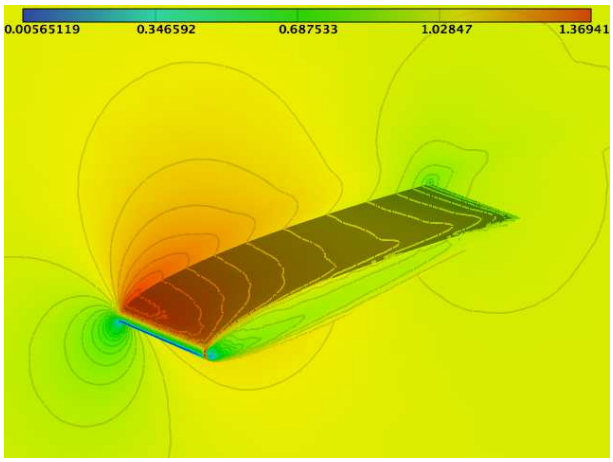

Figure 14. 3D subsonic NACA 0012: Solution (velocity).

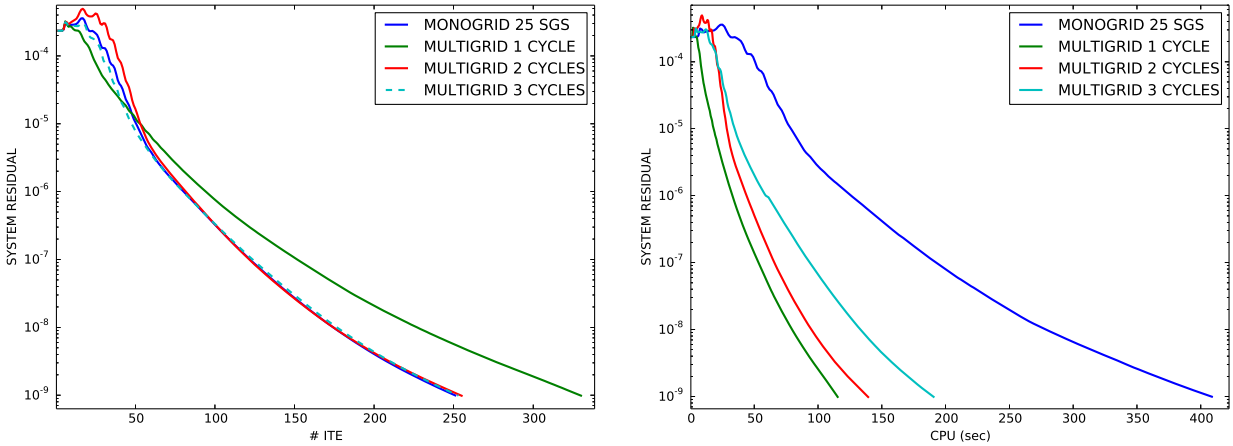

Figure 15. 3D subsonic NACA 0012 : Convergence of the residual of the whole simulation. Left: number of time iterations. Right: wall clock time (sec).

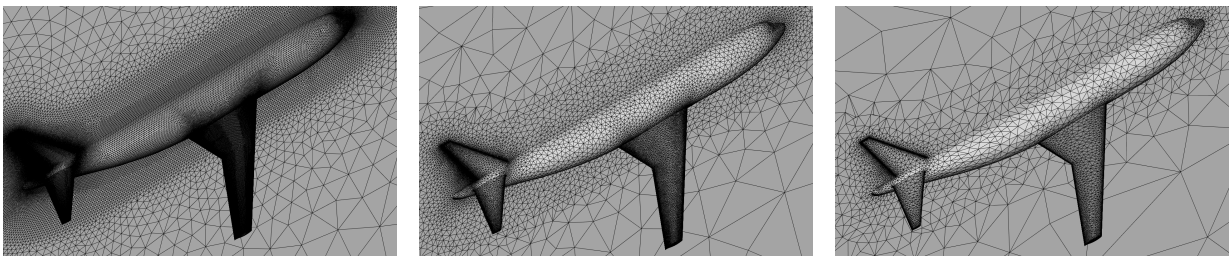

Figure 16. 3D WBT configuration : Close-up views of the three meshes used for multigrid simulations.

II. Full Multigrid (FMG) Coupled with Adaptivity

II.1. Full Multigrid Validation

This section presents the full multigrid (FMG) algorithm and its validation.

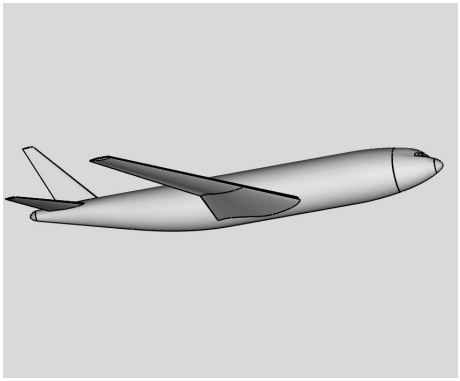

Figure 17. 3D wing body tails (WBT) configuration.

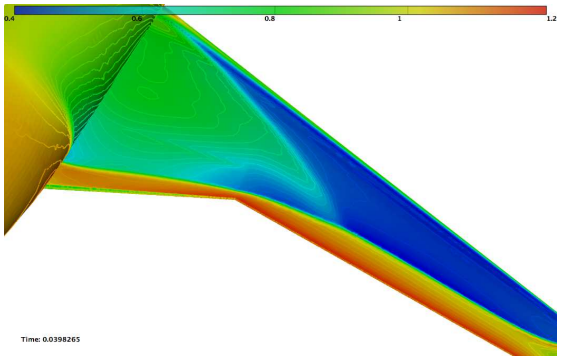

Figure 18. Solution computed on the finest WBT mesh using a 3-grid V-cycle.

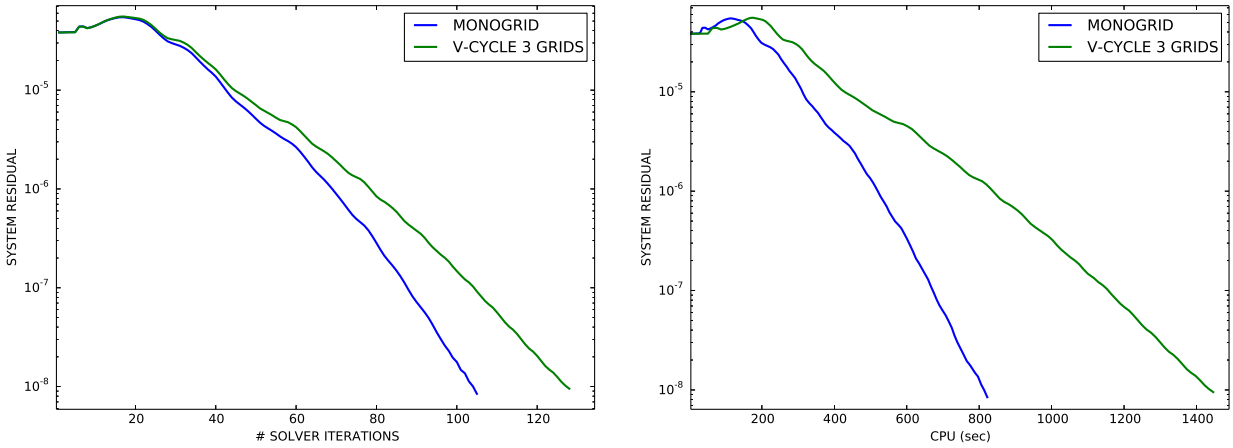

Figure 19. 3D WBT : Residual convergence in terms of the number of solver iterations (left) and wall clock time (right). Comparison between a monogrid computation and a 3-grid V-cycle.

II.1.1. Description of the FMG Method

The full multigrid algorithm is an iterative process (see Figure 23). At each stage, a solution is computed on a mesh, whose complexity of the mesh is increased from one stage to the next. The coarsest mesh is used for the first stage and the finest mesh is used for the final stage. At each stage i ($i >= 2$), the coarser meshes from the previous stages are used to run a multigrid simulation. At the end of each stage, the solution is linearly interpolated to the next (finer) mesh and then used as a restart solution by the flow solver. For the sake of clarity, we describe the FMG algorithm for the case of a sequence of four meshes:

$$\mathcal{H}_h, \mathcal{H}_{2h}, \mathcal{H}_{4h}, \mathcal{H}_{8h} \, ,$$

where $\mathcal{H}_h$ is the finest mesh and $\mathcal{H}_{8h}$ the coarsest. A full multigrid algorithm using these four meshes consists in computing a solution on each mesh, as follows:

1. On $\mathcal{H}_{8h}$: starting from a uniform solution $\mathcal{S}^0_{8h}$, a solution $\mathcal{S}_{8h}$ is computed using a single-grid method. $\mathcal{S}_{8h}$ is then interpolated to $\mathcal{H}_{4h}$.

2. On $\mathcal{H}_{4h}$: the interpolated of $\mathcal{S}_{8h}$ is used as a restart solution by the flow solver : $\mathcal{S}^0_{4h} = I_{8h \to 4h}(\mathcal{S}_{8h})$. A two-grid multigrid simulation is then performed on $\mathcal{H}_{4h}$ using $\mathcal{H}_{8h}$ as the coarse mesh. $\mathcal{S}_{4h}$ is then interpolated to $\mathcal{H}_{2h}$.

3. On $\mathcal{H}_{2h}$: $\mathcal{S}^0_{2h} = I_{4h \to 2h}(\mathcal{S}_{4h})$ is used as a restart solution and a 3-grid multigrid simulation is performed on $\mathcal{H}_{2h}$ using $\mathcal{H}_{4h}$ and $\mathcal{H}_{8h}$ as coarser meshes.

4. On $\mathcal{H}_h$: $\mathcal{S}_h^0 = I_{2h\to h}(\mathcal{S}_{2h})$ and a 4-grid multigrid simulation is performed.

This algorithm is naturally extended to N meshes. The FMG theory[15] states that if the solution is fully converged at stage 1, then it is sufficient to converge the solution by one order of magnitude at stages 2, 3, etc. in order to achieve the global convergence on the finest mesh. In other words, fully converging the solution at every stage would not improve the residual on the finest mesh (and would be more CPU comsuming). In our approach, we chose to converge the solution by two orders at stages 2, 3, etc. instead of one order, to be sure to achieve the global convergence on adapted meshes.

II.1.2. FMG Validation

The FMG method is validated using two examples introduced in Section I.5.2: a 2D transonic NACA and a 3D wing body tails (WBT) configuration. We compared the convergence rate of the residual in terms of wall clock time, for the FMG algorithm and a 4 grid V-cycle simulation on the finest mesh, which can be seen as the last stage of the FMG algorithm, but starting from a uniform solution. Results are presented in Figure 20 for the 2D NACA and in Figure 24 for the 3D transonic WBT configuration. In both cases, a faster convergence is obtained using the FMG method.

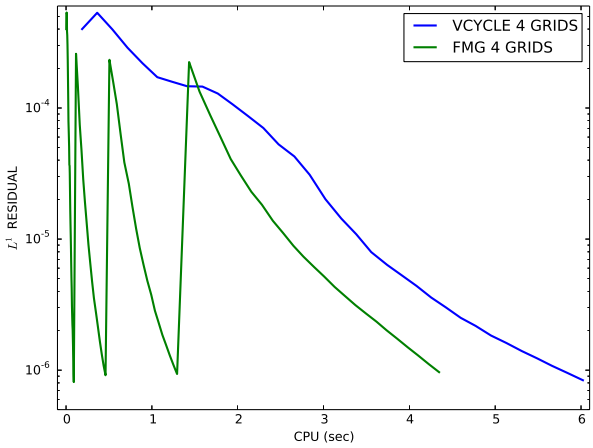

Figure 20. 2D transonic NACA : Comparison of convergence rates with respect to wall clock time. A faster convergence is obtained using an FMG algorithm.

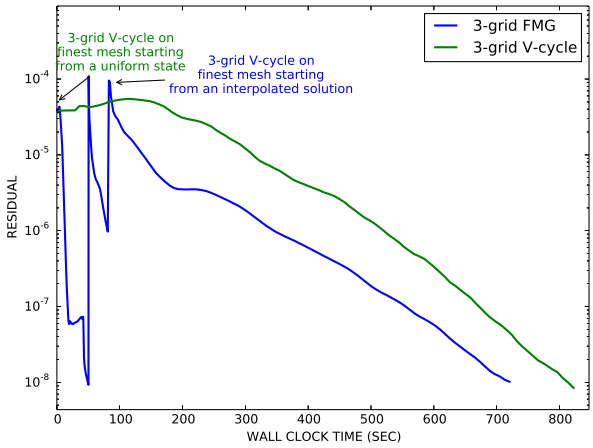

Figure 21. 3D transonic WBT configuration: Comparison of convergence rates with respect to wall clock time. A faster convergence is obtained using an FMG algorithm.

II.2. Steady Mesh Adaptation

Mesh adaptation provides a way to control the accuracy of the numerical solution by modifying the domain discretization according to size and directional constraints. When dealing with real life flow problems, Hessian-based unstructured mesh adaptation has already proved its efficiency to improve the tradeoff between the accuracy of the solution and the number of degrees of freedom (i.e. the complexity of the problem).[27,28,29,30] In addition, as a large number of physical phenomena are anisotropic by nature, anisotropic mesh adaptation improves this ratio even more.

The classical steady mesh adaptation scheme is a fixed-point algorithm. Starting from an initial mesh and solution $(\mathcal{H}_0, \mathcal{S}_0^0)$, both of them are converged to an optimal state based on metric tensor fields and on the concept of unit-mesh.

The classical steady mesh adaptation loop. The main steps of the algorithm are depicted in Figure 22. Given $(\mathcal{H}_i, \mathcal{S}_i)$, a metric tensor $\mathcal{M}_i$ is computed at each vertex of the mesh $\mathcal{H}_i$, according to a given error estimation of the solution (Hessian-based, for instance). $\mathcal{M}_i$ contains information on optimal sizes and

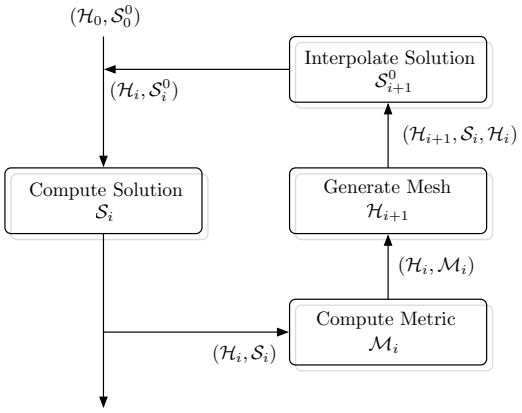

Figure 22. The classical mesh adaptation loop

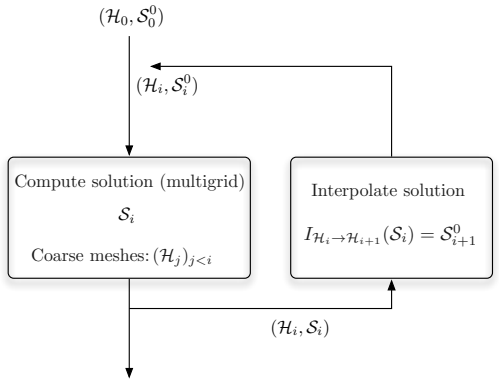

Figure 23. The FMG algorithm

directions. This information given by the metric tensor field $\mathcal{M}_i$ is then used by the adaptive anisotropic remesher (described in Section I.1) to generate a new mesh $\mathcal{H}_{i+1}$. Then, $\mathcal{S}_i$ is interpolated on $\mathcal{H}_{i+1}$: we obtain $\mathcal{S}_{i+1}^0$ which is then used as a restart solution by the solver for the next stage of the mesh adaptation loop. This loop is repeated until convergence, i.e. until there is not too much variation in both the mesh and the solution from one stage to the next.

II.3. Coupling FMG and Mesh Adaptation

There are many similarities between the FMG algorithm and the classical mesh adaptation loop. In both cases, (i) we start from an initial coarse mesh, (ii) the complexity of the current mesh is increased at each stage, and (iii) the solution on the current mesh is linearly interpolated to the next mesh and then used as a restart solution by the flow solver. We start by describing the coupling of the two algorithms, then we explain how both of them can benefit from this coupling.

Description of the coupling. Coupling a FMG algorithm with adaptivity consists in modifying the solution computation step in the classical mesh adaptation loop (see Figure 22). Instead of a single-grid computation, a i-grid multigrid computation is performed at stage i, using the meshes previously adapted as coarser meshes.

Improving the FMG algorithm using a coupling. There are two reasons why the FMG algorithm can benefit from this coupling. First, it is based on uniform meshes which are generated previous to the computation. These non-adapted meshes do not take into account the characteristics of the physical phenomena, which are often anisotropic and located in small areas of the computational domain. Thus, FMG uses meshes which are not optimal in terms of both sizes and directions. As explained in Section II.2, mesh adaptation has proved its efficiency in reducing computational time while improving the accuracy of the solution. The second reason is that the anisotropic meshes generated using mesh adaptation are particularly well-suited to a multigrid computation.

Improving the mesh adaptation loop using a coupling. An example of how mesh adaptation can benefit from multigrid is presented in Figure 24. Two simulations of a transonic flow over a Falcon business jet geometry are considered. First, a classical FMG algorithm is performed using four mesh levels $\mathcal{H}_h$, $\mathcal{H}_{2h}$, $\mathcal{H}_{4h}$ and $\mathcal{H}_{8h}$. The resulting residual convergence is compared to a second simulation, which consists in replacing every multigrid simulation of the FMG algorithm by a single-grid computation. In other words, this second computation is very similar to the classical mesh adaptation loop: mesh complexity is increased at each stage but no multigrid strategy is used. The residual convergences are identical for the two first stages, but a major difference appears during the third: the single-grid computation fails to converge on $\mathcal{H}_{2h}$, while the multigrid procedures used on $\mathcal{H}_{2h}$ and then on $\mathcal{H}_h$ ensure clean convergences on both meshes. So, in addition to a faster convergence in terms of wall clock time, this comparison shows that multigrid

procedures can improve the robustness of the mesh adaptation strategy.

s

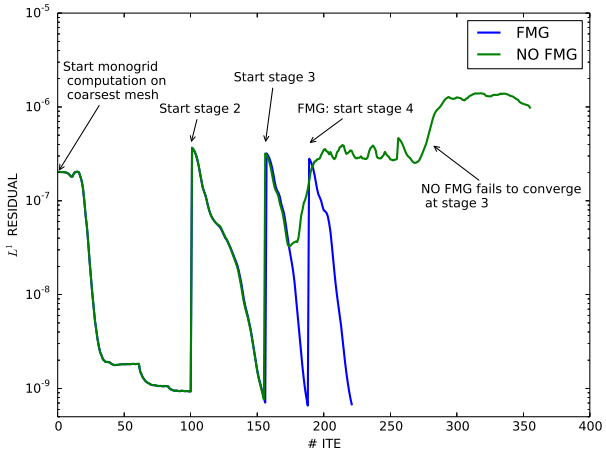

Figure 24. Transonic Falcon: residual convergence in terms of time iterations. Blue: classical FMG algorithm. Green: multigrid computations were replaced by single-grid computations in the FMG algorithm.

II.4. Results

Two cases were considered to validate the adaptive FMG algorithm: a 3D subsonic NACA 0012 airfoil and a 3D transonic WBT configuration, which were both introduced in Section I.5.2. For each case, two simulations were performed:

- A classical mesh adaptation process: the prescribed mesh complexity was increased at each stage, and monogrid simulations were run. As is it usually done, the residual of the solution was fully converged at each stage.
- An adaptive FMG algorithm: the same mesh complexities were prescribed. The residual of the solution was fully converged for the lowest mesh complexity (i.e. stage 1), and then reduced by two orders of magnitude for the other complexities.

In both cases, the adaptive multigrid algorithm showed a significant reduction of the total wall clock time of the simulation. It was ensured that both methods converged to same final solution. To do so, one more stage of the classical mesh adaptation loop was performed using a higher mesh complexity, and the resulting couple mesh/solution was then used as a reference solution to compute spatial errors. Both algorithms showed the same mesh convergence.

II.4.1. 3D Subsonic NACA 0012

The first mesh adaptation considers the 3D subsonic NACA 0012 case that was introduced in Section I.5.2. Six stages of the classical mesh adaptation loop and of the adaptive FMG alorithm were performed, using mesh complexities leading approximately to the following numbers of vertices:

$$8k, 16k, 32k, 64k, 128k, 256k\,.$$

For each mesh complexity, three sub-iterations in the adaptation loop were performed. The residual of the solution was fully converged to 10^{-9} at each stage of the classical adaptation. A slope limiter was used in order to avoid spurious oscillations.[21] A freeze of this limiter is activated in case the limiter itself is oscillating. During the adaptive FMG algorithm, the residual was fully converged at stage 1 (which does not differ from the classical algorithm), and then it was reduced by two orders of magnitude at stages 2, 3, etc. The residual convergence of both simulations in terms of wall clock time is presented in Figure 25. The total wall clock time of the simulation is dramatically improved: 13min9s for the adaptive FMG method,

and 2h25min for the classical adaptation algorithm. As shown in Figure 26, the mesh convergence observed for both methods are similar. The reference couple mesh/solution was computed using one more step in the classical adaptation loop at a mesh complexity leading to $\sim 512k$ vertices. The final adapted mesh and the solution are depicted in Figures 27 and 28.

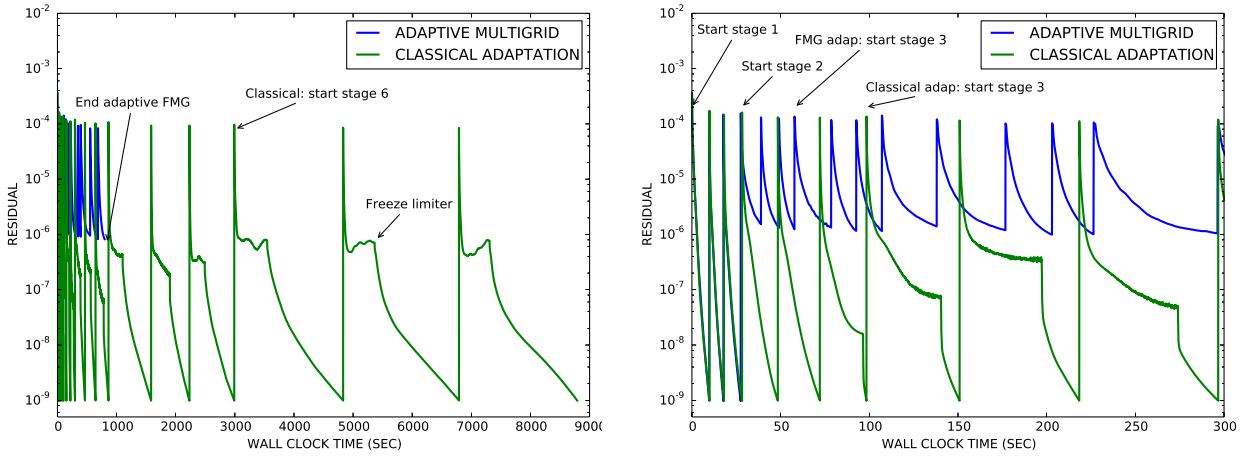

Figure 25. 3D subsonic NACA 0012: comparison of the residual convergence in terms of wall clock time. Left: whole simulation. Right: close-up view of the first stages. Note that each stage corresponds to a mesh complexity, and that three sub-iterations were performed for each one of them.

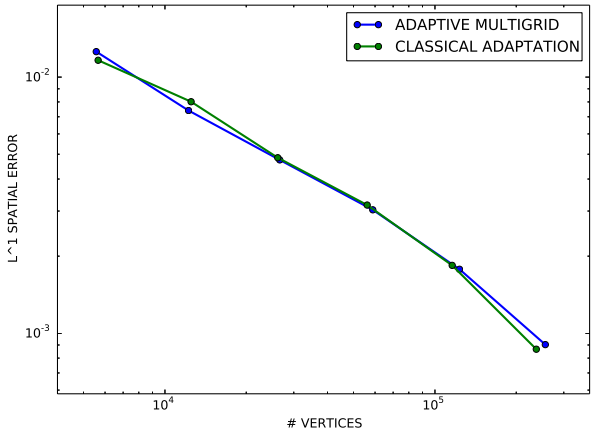

Figure 26. 3D subsonic NACA 0012: mesh convergence to the reference solution.

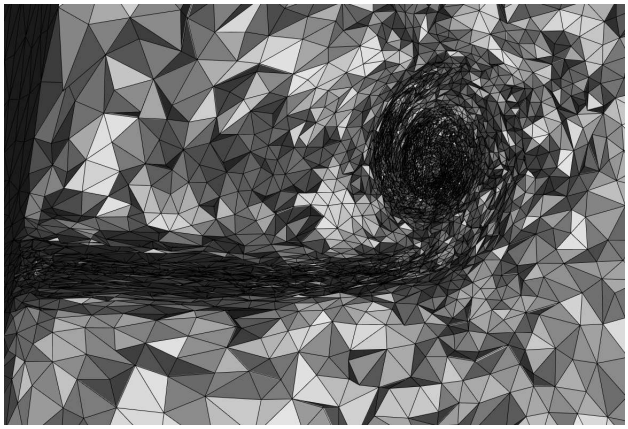

Figure 27. 3D subsonic NACA 0012: Cut in the volume of the final adapted mesh.

II.4.2. 3D Transonic Wing Body Tails Configuration

The second example considers the 3D transonic WBT configuration that was also introduced in Section I.5.2. Four stages of the classical mesh adaptation loop and of the adaptive FMG algorithm were performed. The prescribed mesh complexities (corresponding to each stage) lead approximately to the following numbers of vertices:

$$140k, 210k, 340k, 620k\,.$$

At each stage (i.e. mesh complexity), five sub-iterations in the adaptation loop were performed. A comparison of the residual convergence of both methods in terms of wall clock time is presented in Figure 29. The total wall clock time of the simulation is reduced from 1d3h57m for the classical mesh adaptation loop, to 2h53m for the adaptive FMG algorithm. As shown in Figure 30, converging the residual by two orders of magnitude

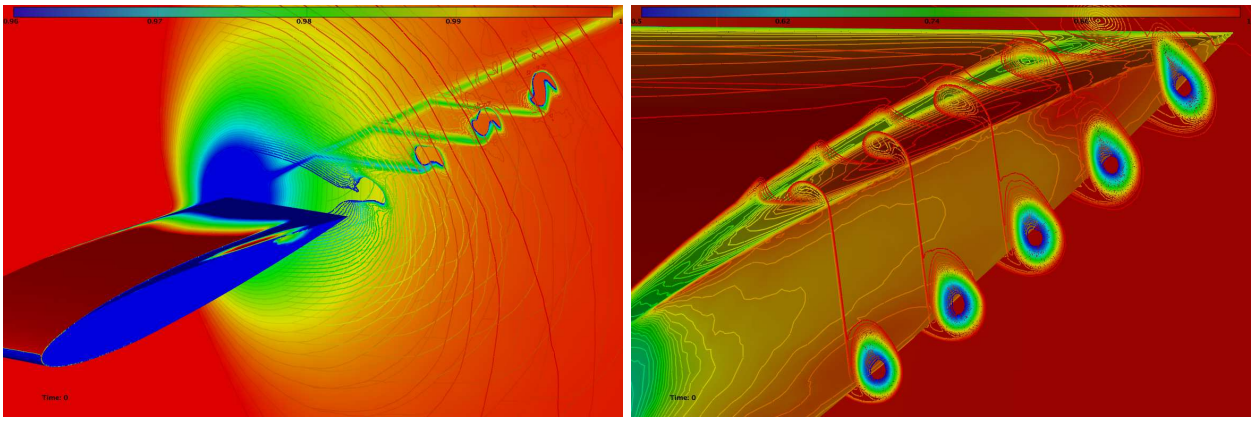

Figure 28. 3D subsonic NACA 0012 : velocity isovalues.

at stages 2, 3 and 4 of the adaptive FMG process does not affect the global convergence to the reference solution. This reference couple mesh/solution was computed by performing one more step of the classical mesh adaptation (∼5M vertices). Various views of the final solution and of the corresponding adapted meshes are depicted in Figures 31, 32 and 33.

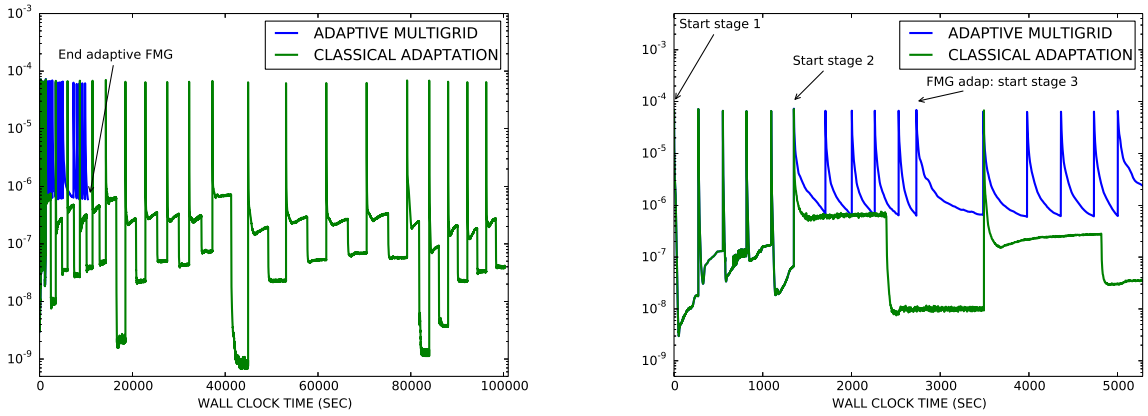

Figure 29. 3D transonic WBT: comparison of the residual convergence in terms of wall clock time. Left: whole simulation. Right: close-up view of the first stages. Note that each stage corresponds to a mesh complexity, and that five sub-iterations were performed for each one of them.

III. Conclusion

The implementation of an implicit multigrid procedure has been described and its validation study has been carried out. A significant improvement of both the convergence speed and the robustness has been observed in the non-adapted case.

The coupling of an FMG algorithm with mesh adaptation has been detailed and compared to the classical mesh adaptation loop for a 3D subsonic and a 3D transonic case. Converging the solution by two orders of magnitude in the adaptive FMG algorithm significantly reduced the total wall clock time of the simulation, while achieving the same global convergence on adapted meshes.

We are working on mesh adaptation strategies for boundary layers, which we hope will allow to extend the adaptive FMG algorithm to turbulent viscous flow simulations.

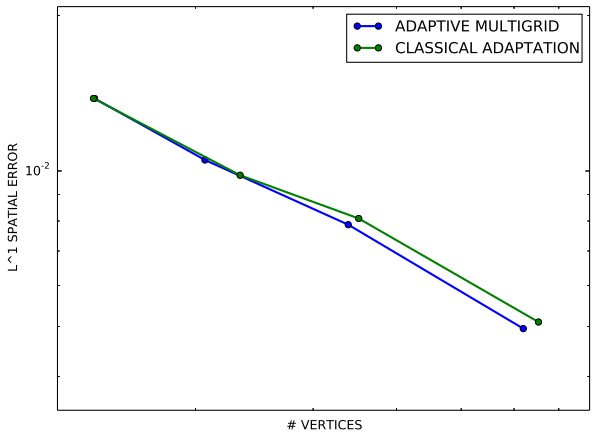

Figure 30. 3D transonic WBT: Mesh convergence to the reference solution.

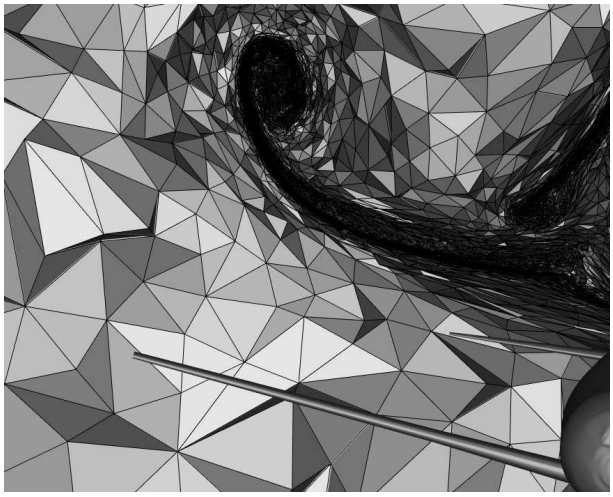

Figure 31. Transonic WBT configuration: cut in the trailing vortices region of the final adapted mesh.

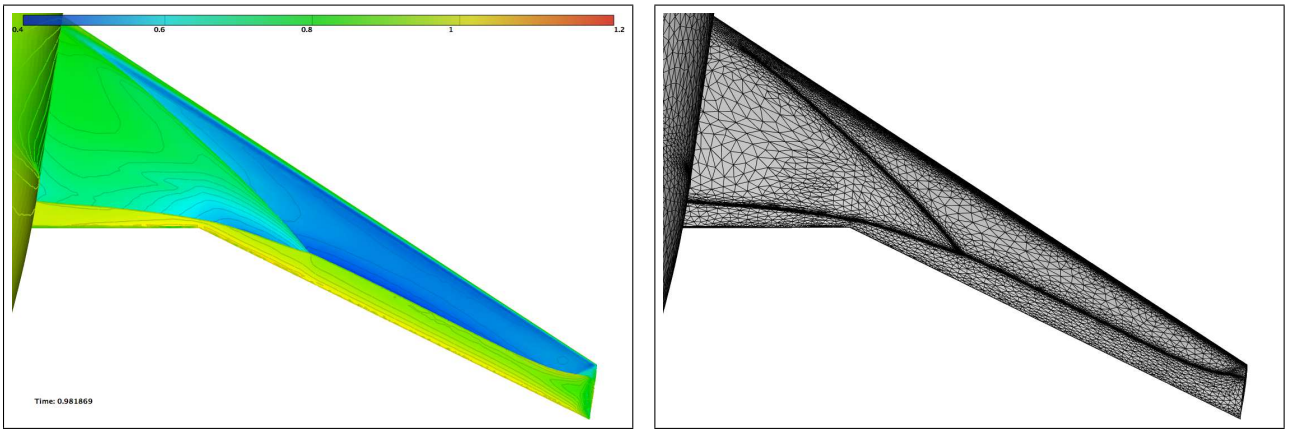

Figure 32. Transonic WBT configuration: pressure on the wing and corresponding adapted surface mesh.

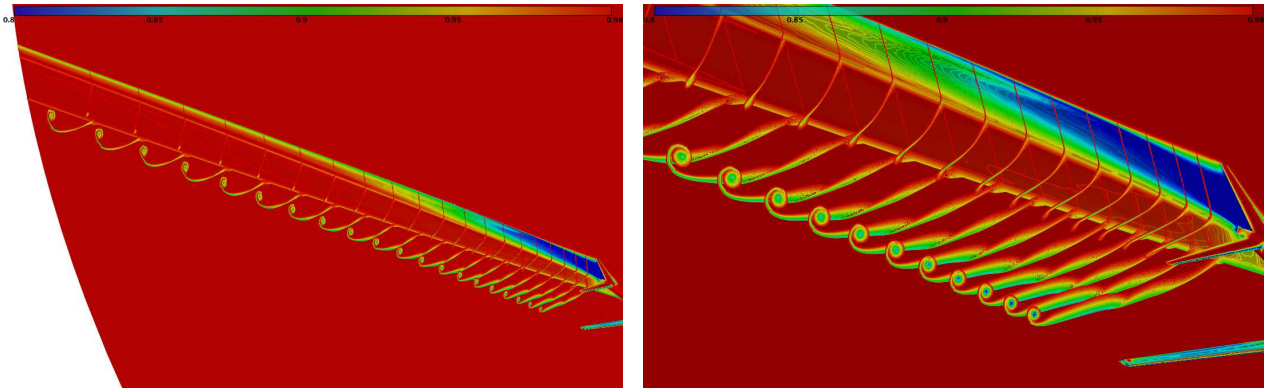

Figure 33. Transonic WBT configuration: velocity isovalues.

IV. Acknowledgements

Thanks to Dave Marcum for providing the initial mesh of his 3D WBT configuration and to Richard James for the proofreading. This work was partially funded by the Airbus Group Foundation.

References

[1]A. Brandt, S. M. and Ruge, J., "Algebraic multigrid (AMG) for sparse matrix equations," *In Sparsity and its applications*, 1983.

[2]Brezina, M., Falgout, R., MacLachlan, S., Manteuffel, T., McCormick, S., and Ruge, J., "Adaptive algebraic multigrid," *SIAM Journal on Scientific Computing*, Vol. 27, No. 4, 2006, pp. 1261–1286.

[3]Trottenberg, U. and Schuller, A., *Multigrid*, Academic Press, Inc., Orlando, FL, USA, 2001.

[4]McCormick, S. F., *Multilevel Adaptive Methods for Partial Differential Equations*, Society for Industrial and Applied Mathematics, 1989.

[5]Sorensen, K. A., Hassan, O., Morgan, K., and Weatherill, N. P., "A multigrid accelerated time-accurate inviscid compressible fluid flow solution algorithm employing mesh movement and local remeshing," *International Journal for Numerical Methods in Fluids*, Vol. 43, No. 5, 2003, pp. 517–536.

[6]Mavriplis, D. J., "Multigrid Strategies for Viscous Flow Solvers on Anisotropic Unstructured Meshes," 1998.

[7]Carré, G., Carte, G., Guillard, H., and Lanteri, S., "Multigrid Strategies for CFD Problems on Non-Structured Meshes," *Multigrid Methods VI*, edited by E. Dick, K. Riemslagh, and J. Vierendeels, Vol. 14 of *Lecture Notes in Computational Science and Engineering*, Springer Berlin Heidelberg, 2000, pp. 1–10.

[8]Francescatto, J. and Dervieux, A., "A Semi-Coarsening Strategy for Unstructured MG with Agglomeration," Tech. Rep. RR-2950, July 1996.

[9]Koobus, B., Lallemand, M.-H., and Dervieux, A., "Unstructured volume-agglomeration MG : solution of the Poisson equation," Research Report RR-1946, 1993.

[10]Guillard, H., "Node-Nested Multi-Grid With Delaunay Coarsening," 1993.

[11]Mavriplis, D. J., "Adaptive meshing techniques for viscous flow calculations on mixed element unstructured meshes," *International Journal for Numerical Methods in Fluids*, Vol. 34, No. 2, 2000, pp. 93–111.

[12]Mller, J.-D., "Anisotropic adaptation and multigrid for hybrid grids," *International Journal for Numerical Methods in Fluids*, Vol. 40, No. 3-4, 2002, pp. 445–455.

[13]Brandt, A. and Livne, O., *Multigrid Techniques*, Society for Industrial and Applied Mathematics, 2011.

[14]Mesri, Y., Guillard, H., and Coupez, T., "Automatic coarsening of three dimensional anisotropic unstructured meshes for multigrid applications," *Applied Mathematics and Computation*, Vol. 218, No. 21, April 2012, pp. pages 10500–10509.

[15]Carré, G. and Dervieux, A., "On the Application of FMG to Variational Approximation of Flow Problems," *International Journal of Computational Fluid Dynamics*, Vol. 12, No. 2, 1999, pp. 99–117.

[16]Guillard, H. and Vanek, P., "An Aggregation Multigrid Solver for Convection-diffusion Problems on Unstructured Meshes." Tech. rep., 1998.

[17]Mavriplis, D. J. and Mavriplis, D. J., "Multigrid techniques for unstructured meshes," Tech. rep., in VKI Lecture Series VKI-LS, 1995.

[18]Loseille, A. and Menier, V., "Serial and Parallel Mesh Modification Through A Unique Cavity-Based Primitive," *Proceedings of the 22nd International Meshing Roundtable*, edited by X. Jiao and J.-C. Weill, 2013.

[19]Dervieux, A., Loseille, A., and Alauzet, F., "High-order adaptive method applied to high-speed flows," *West-East High Speed Flow Field Conference, nov. 19-22 2007, Moscou, Russia*, 2007.

[20]V. Menier, A. L. and Alauzet, F., "CFD validation and adaptivity for viscous flow simulations," *AIAA Paper* , Vol. 2014-2925, 2014.

[21]Alauzet, F. and Loseille, A., "High Order Sonic Boom Modeling by Adaptive Methods," *J. Comp. Phys.*, Vol. 229, 2010, pp. 561–593.

[22]Jameson, A. and Yoon, S., "Lower-Upper implicit schemes with multiple grids for the Euler equations," *AIAA Journal*, Vol. 25, No. 7, 1987, pp. 929–935.

[23]Luo, H., Baum, J., and Löhner, R., "A fast, matrix-free implicit method for compressible flows on unstructured grids," *J. Comp. Phys.*, Vol. 146, 1998, pp. 664–690.

[24]Luo, H., Baum, J., and Löhner, R., "An accurate, fast, matrix-free implicit method for computing unsteady flows on unstructured grids," *Comput. & Fluids*, Vol. 30, 2001, pp. 137–159.

[25]Sharov, D. and Nakahashi, K., "Reordering of hybrid unstructured grids for Lower-Upper Symmetric Gauss-Seidel computations," *AIAA Journal*, Vol. 36, No. 1, 1997, pp. 484–486.

[26]Sharov, D., Luo, H., Baum, J., and Löhner, R., "Implementation of unstructured grid GMRES+LU-SGS method on shared-memory, cache-based parallel computers," *AIAA Paper* , Vol. 2000-0927, 2000.

[27]Castro-Díaz, M. J., Hecht, F., Mohammadi, B., and Pironneau, O., "Anisotropic Unstructured Mesh Adaptation for Flow Simulations," *Int. J. Numer. Meth. Fluids*, Vol. 25, 1997, pp. 475–491.

[28]Frey, P. J. and Alauzet, F., "Anisotropic mesh adaptation for CFD computations," *Comput. Methods Appl. Mech. Engrg.*, Vol. 194, No. 48-49, 2005, pp. 5068–5082.

[29]Gruau, C. and Coupez, T., "3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric," *Comput. Methods Appl. Mech. Engrg.*, Vol. 194, No. 48-49, 2005, pp. 4951–4976.

[30]Li, X. L., Shephard, M. S., and Beall, M. W., "3D anisotropic mesh adaptation by mesh modification," *Comput. Methods Appl. Mech. Engrg.*, Vol. 194, No. 48-49, 2005, pp. 4915–4950.