FLEXMG: a new library of multigrid preconditioners for a spectral/finite-element incompressible flow solver*

M. Rasquin[12], H. Deconinck[12] and G. Degrez[12]

[1] Department of Aero-Thermo-Mechanics, Université Libre de Bruxelles
CP 165/41, Avenue F. D. Roosevelt 50, 1050 Brussels, Belgium
[2] Aeronautics and Aerospace Department, von Karman Institute
Chaussée de Waterloo/Waterloosesteenweg 72, 1640 Rhode-St-Genèse/Sint-Genesius-Rode, Belgium
E-mail: michel.rasquin@ulb.ac.be

Abstract

A new library called FLEXMG has been developed for a spectral/finite-element incompressible flow solver called SFELES. FLEXMG allows to use various types of iterative solvers preconditioned by algebraic multigrid methods. Two families of algebraic multigrid preconditioners have been implemented, of smooth aggregation-type and non nested finite-element-type. Unlike gridless multigrid, both of these families use the information contained in the initial fine mesh. Our aggregation-type multigrid is smoothed with either a constant or a linear least square fitting function while the non nested finite-element-type multigrid is already smooth by construction. All these multigrid preconditioners are tested as stand-alone solvers or coupled to a GMRES method. After analyzing the accuracy of our solvers on a typical test case in fluid mechanics, their performance in terms of convergence rate, computational speed and memory consumption are compared with the performance of a direct sparse LU solver as a reference. Finally, the importance of using smooth interpolation operators is also underlined in the study.

Keywords: algebraic multigrid; smooth aggregation; non nested finite-element interpolation; GMRES; CFD

1 INTRODUCTION

Computation of turbulent flows requires the unsteady Navier-Stokes equations to be solved on highly refined meshes over a considerable number of small time steps. For applications of practical interest, requirements in terms of computation time and memory consumption remain one of the key aspects for such simulations, which can be accomplished only by using state-of-the-art numerical methods and supercomputing facilities. Indeed, one of the main challenges of contemporary CFD is to solve ever larger problems in a scalable manner (i.e. both memory and computation time requirements scale linearly with problem size). Direct solvers such as exact LU factorizations work well for small problems only. For larger problem sizes, poor scalability is observed, especially in terms of memory [2]. Traditional fixed-point iterative methods (e.g. Jacobi, Gauss-Seidel, ILUT) scale well

*This is the pre-peer reviewed version of [1], which has been published in final form at doi.wiley.com/10.1002/nme.2808.

in terms of memory consumption, but not at all in terms of computation time. The reason for this is that such methods are only effective for damping out high wave number errors. As soon as these have been removed, convergence stagnates on the remaining low wave number error components. Krylov accelerators [3–6] represent a significant improvement over fixed-point methods. Indeed, unlike fixed-point methods, they look for an optimized solution that minimizes the residual at each iteration. Nevertheless, unless scalable preconditioners are used, even these algorithms fail to provide adequate scaling.

The missing ingredient is multigrid, a technique which relies on a series of increasingly coarser approximations of the original 'fine' problem. The underlying concept is always the same: low wave number errors on fine grids become high wave number errors on coarser levels. Hence, they may be effectively removed by recursively applying fixed-point methods on coarser levels [7,8]. In this context, a new library of iterative solvers preconditioned by algebraic multigrid has been developed for CFD applications and is presented in this paper.

2 BACKGROUND

In this study, our library FLEXMG is tested within the combined spectral/finite-element code SFELES [9–11]. This unsteady and incompressible Navier-Stokes solver is based upon a combination of finite-element in-plane discretization and a spectral discretization (i.e. a truncated Fourier series) in the transverse direction (or azimuthal if working in axisymmetric coordinates). Together with a pseudo-spectral treatment of the nonlinear terms, this approach allows to transform 3D problems into a series of N 2D linear problems in Fourier space that are decoupled within each time step. This pseudo-spectral treatment implies that the nonlinear terms are computed in physical space and transfered back to Fourier space using a Fast Fourier Transform.

At each time step, N sparse linear systems (corresponding to N Fourier modes) need to be solved with eight unknowns per node (i.e. real and imaginary components of u, v, w and p). For each system $A_m x = b_m$ (associated to a Fourier mode m), only the linear terms of the Navier-Stokes equations (time derivative, pressure, diffusion) are contained in the stiffness matrix A_m while all non-linear terms (convection and stabilization terms) are contained in the right hand side b_m. Moreover, the size of each system A_m is equal to eight times the number of nodes, except for modes 0 and $\frac{N}{2}$ which are purely real and have therefore only four unknowns per node. However, the two systems associated to these real modes are grouped together in SFELES so that the size of all the systems is always identical, which ensures a good load balancing when computing in parallel.

The considered linear systems are in general very large, nearly symmetric, constant in time and can be very stiff due to highly anisotropic stretched meshes needed to resolve viscous boundary layers. Exact LU factorization is quite efficient for this code, provided that the number of nodes per finite-element plane is not too large. Indeed, since the stiffness matrices are constant in time, LU factorizations are performed only once at the first time step and stored for the whole computation. Nevertheless, the number of unknowns per finite-element plane is limited by the memory consumption of this direct method. The possibility of solving these systems using various types of algebraic multigrid methods is therefore currently explored [2, 12]. Although FLEXMG is originally designed for $2D$ problems arising from the combined spectral/finite-element discretization used in SFELES, all the concepts presented in this paper are easily extendable to $3D$ problems.

3 MULTIGRID PRECONDITIONERS

Let

$$Ax = b \tag{1}$$

be the problem to solve. Multigrid is an iterative procedure which relies on a series of increasingly coarser approximations of the original 'fine' problem. The underlying concept is the following: low wave number errors on fine grids become high wave number errors on coarser levels which can be effectively removed by applying fixed-point (also called relaxation) methods on coarser levels [7,8]. A typical multilevel correction scheme with two grid levels is described in Fig. 1.

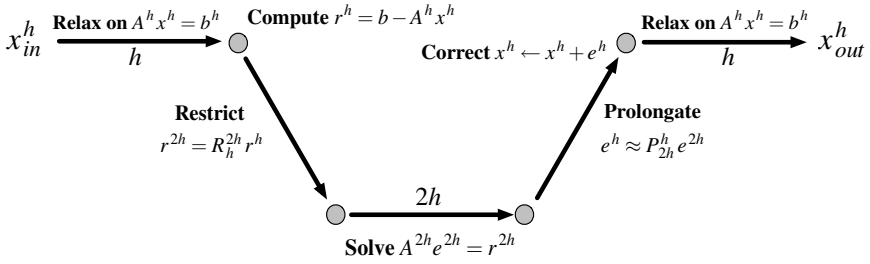


Figure 1: Multilevel correction scheme.

A typical V multigrid cycle starts at the finest level with an arbitrary initial guess as a solution. Some relaxation iterations are performed on the fine grid to remove the high wave number errors. The residual on the finest level is then computed and transferred from the fine level to the next coarser level. The remaining low wave number errors on the fine grid appears now as high wave number errors on the coarse grid and can thus be eliminated efficiently with some additional relaxation iterations at the coarse level. The transfer from a fine level to the next coarser one is called 'restriction'. After some relaxation cycles on the coarse level, the residual of the coarse problem is then restricted again to next coarser level and so forth until the coarsest level is reached.

At the coarsest level, the system is usually solved exactly with a direct solver since the size of the problem at that level has decreased significantly. The solution obtained at the coarsest level is than interpolated back to the finer level. The transfer from a coarse level to the next finer one is called 'prolongation'. The solution of the fine grid problem is then corrected and interpolated to the next finer level after some relaxation iterations, called post multigrid sweeps. The solution is finally prolongated until the finest level is reached. In conclusion, each grid is responsible for eliminating a particular wave number of the bandwidth of errors.

The main difficulty of this multilevel correction scheme lies in the construction of the coarse grid stiffness matrices and the grid transfer operators to interpolate the solution from a coarse level to a fine level (prolongators) and vice versa (restrictors). FlexMG provides two main procedures of building the prolongators which are described in the next section. The inverse operation is ensured by the restrictors which are simply taken as the transpose of the prolongators in all cases. At each level (except the finest), the stiffness matrices are built according to a Galerkin formulation [8]:

$$A_{coarse} = R_{fine \to coarse} \, A_{fine} \, P_{coarse \to fine} \tag{2}$$

The multilevel correction scheme described in this section can be used alone or coupled to a well-known GMRES method (Generalized Minimum RESidual [3,5,6]). When used as a stand-alone solver, the whole process illustrated in Fig. 1 is repeated until satisfactory convergence is reached. One of the main drawbacks of such a method is its lack of robustness. Indeed, nothing guarantees that this method will converge in all cases, or at least not diverge. To obviate this problem, the multilevel correction scheme can also be used as a preconditioner of a GMRES method.

Considering a right preconditioning of the system described in Eq. (1)

$$(A\,M^{-1})\,u = b \quad \text{and} \quad x = M^{-1}\,u \tag{3}$$

and letting

$$K_k = \text{span}\{\, r^0,\, A\,M^{-1}\,r^0,\, \ldots,\, (A\,M^{-1})^{k-1}\,r^0 \,\} \tag{4}$$

be its Krylov subspace of dimension k, with r^0 the initial residual (i.e. $r^0 = b - A\,x^0$) and M^{-1} a cheap approximation of A^{-1}, the basic idea of GMRES is to look for a solution x^k (at iteration k) that belongs to the subspace K_k and minimizes the residual r^k.

The solution for this method can then be written as

$$u^k = u^0 + V_k\,y_k \tag{5}$$
$$x^k = M^{-1}\,u^k \tag{6}$$

where V_k is an orthonormal basis for the Krylov subspace of dimension k already defined in Eq. (4) and y_k some coefficients that minimize the residual r^k at iteration k. At each iteration of the GMRES procedure, a new search vector for the solution is thus determined by computing an optimal linear combination of all the previous search vectors. Optimal here means which minimizes the residual, as mentioned above [3, 5, 6]. It should also be mentioned that M^{-1} is assumed to be fixed for the traditional GMRES procedure.

Back to the multilevel correction scheme, the last paragraph of this section explains how a multigrid V cycle can be used as a preconditioner of a GMRES procedure. After deriving the whole multilevel correction scheme for k iterations (k V cycles) and provided that the initial guess $x^0 = 0$, the solution of the problem $A\,x = b$ can be written as $x^k = P_k^{-1}\,b$ with P_k^{-1} an explicit approximation of A^{-1} which depends on k. If k is fixed (usually one when coupled to GMRES) and if the number of relaxation steps is kept constant at each level of the multilevel correction scheme (usually one, two or three), the multigrid preconditioner P_k^{-1} remains constant from step to step and can be used as well as a preconditioner for the traditional GMRES method when computing a new search vector in the Krylov subspace (take $M^{-1} = P_k^{-1}$ in Eq. 3 to Eq. 6). If P_k^{-1} is not constant from one step to another, the traditional GMRES procedure will not converge and cannot be used any more. In order to handle variations in the preconditioner, a number of variants have been proposed in the literature such as 'flexible' and 'recursive' GMRES methods (FGMRES [5], GMRESR [6, 13]). FGMRES and GMRESR are also implemented in FLEXMG but the influence of varying multigrid preconditioners has not been addressed in this study. However, when coupled to fixed preconditioners, FGMRES and GMRESR produce, in exact arithmetic, identical results as the traditional GMRES method [12] but with a higher memory consumption since the equivalent of two Krylov subspaces need to be stored for FGMRES and GMRESR instead of one for GMRES.

4 CONSTRUCTION OF THE PROLONGATORS

As it was mentioned in the previous section, all multigrid methods require the definition of a succession of coarse grids based on the original fine grid in order to construct the prolongation and restriction operators. Two distinct families of multigrid preconditioners have been implemented in FLEXMG for that purpose and are presented in the next three subsections.

4.1 Piecewise constant aggregation-type multigrid (MG - Agg)

The prolongators of this first family are based on a clustering procedure of nodes into aggregates or 'super-nodes' [14–16]. These aggregates correspond to a combination of several nodes from the original fine grid and will constitute new degrees of freedom on the coarser grid levels.

In a classical gridless aggregation-type multigrid, strongly coupled nodes are clustered together according to a neighborhood function defined at each node and based on the system matrix (e.g. [15–19]). Instead of subdividing the system matrix using a neighborhood function, another option consists in using the information contained in the initial fine mesh if it is available. In our implementation, the original fine mesh is therefore subdivided recursively using a bottom-up strategy [20]. To do so, the nodal connectivity graph of the fine mesh is first constructed. Then, a graph partitioning algorithm is called to obtain a small number of sub-graphs. This first set of sub-graphs corresponds to the coarsest level in the multigrid correction scheme. Finally, each sub-graph is recursively subdivided in order to build the other (less) coarse levels. This recursive algorithm stops once the number of sub-graphs corresponds to the desired number of unknowns on the first coarse level. The graph partitioning algorithm coupled to FLEXMG comes from the METIS package [21, 22]. The coarse grids are therefore generated automatically and only the initial fine mesh is required. This method is illustrated in Fig. 3, with the traditional circular cylinder test case in $2D$.

From the construction of these aggregates, it is possible to build piecewise constant prolongation operators for which the solution of each aggregate at a coarse level is simply prolongated to each of its fine nodes (or fine aggregates). Every row of these piecewise constant prolongators has therefore one and only one non zero entry equal to one [16, 23]. This procedure is illustrated in Fig. 2 and Eq. 7 with a simple example for which five nodes are clustered into two aggregates (Fig. 2) and two unknowns per node are considered with a nodal ordering (Eq. 7).

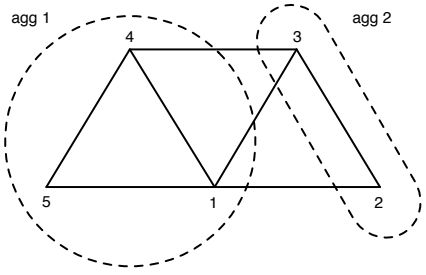
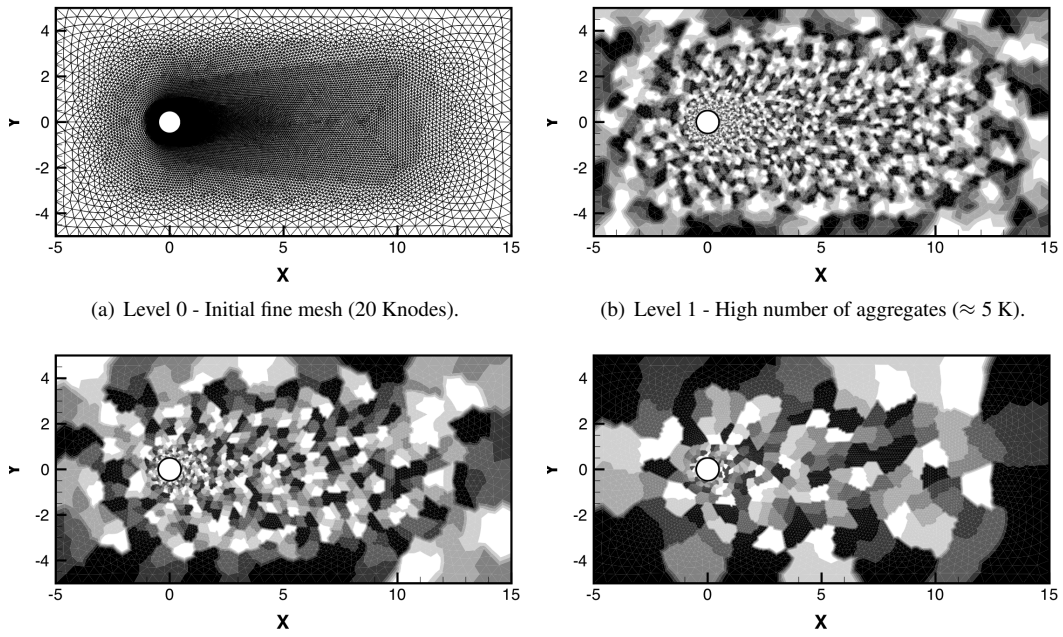


Figure 2: Aggregation process for the construction of the prolongators in MG-Agg. Fives fine nodes are clustered into two aggregates ($2D$ example).

$$P = \begin{pmatrix} \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \end{pmatrix} \begin{matrix} \} \text{ node 1} \\ \} \text{ node 2} \\ \} \text{ node 3} \\ \} \text{ node 4} \\ \} \text{ node 5} \end{matrix} \quad (7)$$

$$\underbrace{\qquad}_{\text{agg 1}} \quad \underbrace{\qquad}_{\text{agg 2}}$$



(a) Level 0 - Initial fine mesh (20 Knodes).

(b) Level 1 - High number of aggregates ($\approx$ 5 K).

(c) Level 2 - Moderate number of aggregates ($\approx$ 1.25 K).

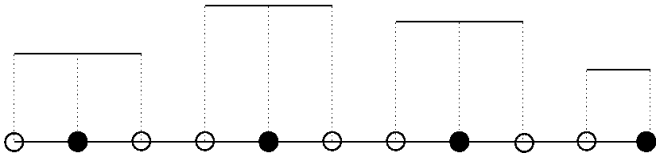
(d) Level 3 - Low number of aggregates ($\approx$ 0.31 K).

Figure 3: Aggregation process for the construction of the prolongators in MG-Agg. Each color zone represents an aggregate of nodes (or aggregate of aggregates).

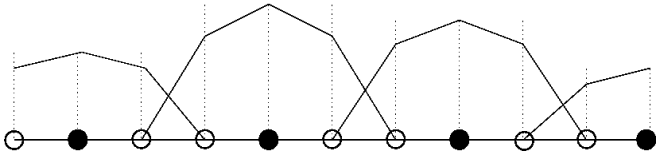
4.2 Smooth aggregation-type multigrid

Piecewise constant prolongators described in Section 4.1 are however not optimal because they introduce substantial high wave number error components by themselves. To improve these piecewise

constant prolongation operators, smoothing operators have to be defined in order to construct smooth prolongators, as illustrated in Fig.4.



(a) Piecewise constant prolongator.



(b) Smooth prolongator.

Figure 4: Piecewise constant and smooth prolongators for the aggregation-type multigrid in $1D$.

Many propositions based on the system matrices have been proposed in the literature for that purpose (e.g. [23–26]). In FLEXMG, two different smoothers based on the geometry of the initial fine mesh have been implemented and are applied after every transfer of the solution from a coarse grid to a fine grid. In practice, the piecewise constant prolongators P_{pc} and smoothing operators S are grouped together to form smooth prolongation operators P_{smth} (i.e. $P_{smth} = SP_{pc}$).

4.2.1 Smooth aggregation-type multigrid based on a constant least square fitting technique (MG-AggSmthCLSF)

For the first smoothing strategy, each node (or aggregate) on a certain level is assigned the value of a constant least square fitting function based on the value of the solution of its direct neighboring nodes (or aggregates). One speaks here about constant least square fitting technique since the local fitting function associated to each node i (or aggregate i) can be written in $2D$ as

$$\tilde{f}_i\,(x,\,y) = a \tag{8}$$

with a a coefficient obtained from a least square method.

After the prolongation of the solution from a coarse level to a fine level, the value of the piecewise constant solution $f(x_i,\,y_i)$ at node i (or aggregate i) is therefore replaced by a. In order to determine the coefficient a, the least square method applied to each node i (or aggregate i) consists in minimizing the function ϕ_i defined by Eq. 9.

$$\begin{aligned} \phi_i &= \sum_{j\,\in\,\mathcal{F}_i} \Big[f(x_j,\,y_j) - \tilde{f}_i(x_j,\,y_j) \Big]^2 \\ &= \sum_{j\,\in\,\mathcal{F}_i} \big[f(x_j,\,y_j) - a \big]^2 \end{aligned} \tag{9}$$

$\mathcal{F}_i$ is a set of points associated to node i and defined according to Fig. 5.

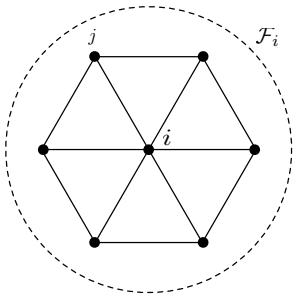


Figure 5: Definition of the set of points $\mathcal{F}_i$ associated to node i in $2D$: $\mathcal{F}_i = \{\text{node } i \cup \text{neighbors of node } i\}$.

To Minimize the function ϕ_i is equivalent to force the derivatives of ϕ_i with respect to a to be equal to zero, which leads to Eq. 10.

$$-\frac{1}{2}\frac{\partial \phi_i}{\partial a} = \sum_{j \in \mathcal{F}_i} \left[f(x_j, y_j) - a \right] = 0 \tag{10}$$

The second equality in Eq. 10 can be further developed in order to obtain an expression for the coefficients a.

$$a = \frac{1}{n_{\mathcal{F}_i}} \sum_{j \in \mathcal{F}_i} f(x_j, y_j) \tag{11}$$

With $n_{\mathcal{F}_i}$ equal to the number of points in the set $\mathcal{F}_i$ defined for node i (or aggregate i), it turns out that this constant least square fitting technique is equivalent to a local average based on the solution of its direct neighboring nodes (or aggregates). For each node i (or aggregate i), the value of its local fitting function evaluated precisely at node i (or aggregate i) can be written as

$$\begin{aligned} \tilde{f}_i(x_i, y_i) &= \frac{1}{n_{\mathcal{F}_i}} \sum_{j \in \mathcal{F}_i} f(x_j, y_j) \\ &= \sum_{j \in \mathcal{F}_i} s_{ij}^f f(x_j, y_j) \quad \text{with} \quad s_{ij}^f = \frac{1}{n_{\mathcal{F}_i}} \end{aligned} \tag{12}$$

The coefficients of the smoothing matrix S introduced in the previous paragraph can be deduced from Eq. 12 and particularly from the terms s_{ij}^f.

This smoother is illustrated in Fig. 6(a) with a $1D$ example and in Fig. 7(b) with a $2D$ example.

4.2.2 Smooth aggregation-type multigrid based on a linear least square fitting technique (MG-AggSmthLLSF)

For the second smoother, each node (or aggregate) on a certain level is assigned the value of a local plane whose coefficients are computed from a least square method based on the value of the solution *and* the position of its direct neighboring nodes (or aggregates). The position of an aggregate is simply

taken as the center of gravity computed from the position of all its nodes. One speaks here about linear least square fitting technique since the local fitting function associated to each node i (or aggregate i) can be written in $2D$ as

$$\tilde{f}_i\,(x,\,y) = a + b\,x + c\,y \quad \text{or} \quad \tilde{f}_i\,(x,\,y) = (1,\,x,\,y)\begin{pmatrix} a \\ b \\ c \end{pmatrix} \tag{13}$$

with (a, b, c) the coefficients of the plane obtained from a least square method. After the prolongation of the solution from a coarse level to a fine level, the value of the piecewise constant solution $f(x_i,\,y_i)$ at node i (or aggregate i) is therefore replaced by $\tilde{f}_i(x_i,\,y_i)$ with $(x_i,\,y_i)$ the position of the considered node i (or aggregate i). In order to determine the coefficients a, b and c, the least square method applied to each node i (or aggregate i) consists in minimizing the function ϕ_i defined by Eq. 14.

$$\begin{aligned} \phi_i &= \sum_{j\,\in\,\mathcal{F}_i} \Big[f(x_j,\,y_j) - \tilde{f}_i(x_j,\,y_j) \Big]^2 \\ &= \sum_{j\,\in\,\mathcal{F}_i} \left[f(x_j,\,y_j) - \big(1,\,x_j,\,y_j\big) \begin{pmatrix} a \\ b \\ c \end{pmatrix} \right]^2 \end{aligned} \tag{14}$$

$\mathcal{F}_i$ is the same set of points as the one defined for the constant least square fitting function in Fig. 5. To minimize the function ϕ_i is equivalent to force its derivatives with respect to a, b and c to be equal to zero, which leads to Eq. 15.

$$-\frac{1}{2}\begin{pmatrix} \frac{\partial \phi_i}{\partial a} \\ \frac{\partial \phi_i}{\partial b} \\ \frac{\partial \phi_i}{\partial c} \end{pmatrix} = \sum_{j\,\in\,\mathcal{F}_i} \begin{pmatrix} 1 \\ x_j \\ y_j \end{pmatrix} \left[f(x_j,\,y_j) - \big(1,\,x_j,\,y_j\big) \begin{pmatrix} a \\ b \\ c \end{pmatrix} \right] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{15}$$

The second equality in Eq. 15 can be further developed in order to obtain an expression for the coefficients a, b and c.

$$\begin{aligned} \left[\sum_{k\,\in\,\mathcal{F}_i} \begin{pmatrix} 1 \\ x_k \\ y_k \end{pmatrix} (1,\,x_k,\,y_k) \right] \begin{pmatrix} a \\ b \\ c \end{pmatrix} &= \sum_{j\,\in\,\mathcal{F}_i} \begin{pmatrix} 1 \\ x_j \\ y_j \end{pmatrix} f(x_j,\,y_j) \\ \underbrace{\left[\sum_{k\,\in\,\mathcal{F}_i} \begin{pmatrix} 1 & x_k & y_k \\ x_k & x_k^2 & x_k y_k \\ y_k & x_k y_k & y_k^2 \end{pmatrix} \right]}_{B_i} \begin{pmatrix} a \\ b \\ c \end{pmatrix} &= \sum_{j\,\in\,\mathcal{F}_i} \begin{pmatrix} 1 \\ x_j \\ y_j \end{pmatrix} f(x_j,\,y_j) \\ \begin{pmatrix} a \\ b \\ c \end{pmatrix} &= B_i^{-1} \sum_{j\,\in\,\mathcal{F}_i} \begin{pmatrix} 1 \\ x_j \\ y_j \end{pmatrix} f(x_j,\,y_j) \end{aligned} \tag{16}$$

From Eq. 13 and Eq. 16, it is now possible to give the expression of $\tilde{f_i}$ evaluated precisely at node i (or aggregate i), which yields

$$\begin{aligned} \tilde{f_i}(x_i, y_i) &= (1, x_i, y_i) \begin{pmatrix} a \\ b \\ c \end{pmatrix} \\ &= (1, x_i, y_i)\, B_i^{-1} \sum_{j \in \mathcal{F}_i} \begin{pmatrix} 1 \\ x_j \\ y_j \end{pmatrix} f(x_j, y_j) \\ &= \sum_{j \in \mathcal{F}_i} (1, x_i, y_i)\, B_i^{-1} \begin{pmatrix} 1 \\ x_j \\ y_j \end{pmatrix} f(x_j, y_j) \end{aligned} \tag{17}$$

Finally, this last equation can be written as

$$\tilde{f_i}(x_i, y_i) = \sum_{j \in \mathcal{F}_i} s_{ij}^f f(x_j, y_j) \quad \text{with} \quad s_{ij}^f = (1, x_i, y_i)\, B_i^{-1} \begin{pmatrix} 1 \\ x_j \\ y_j \end{pmatrix} \tag{18}$$

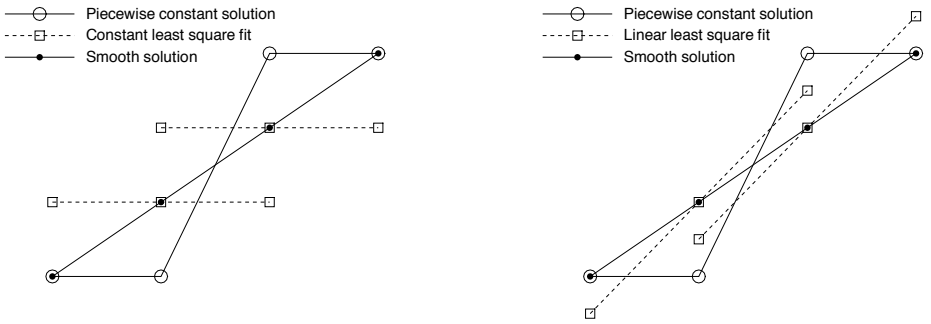
Similarly to the smoother based on a constant least square fit, the coefficients of the smoothing matrix S can be deduced from Eq. 17 and particularly from the terms s_{ij}^f.

For each node i (or aggregate i), the evaluation of the coefficients a, b and c in Eq. 16 requires the computation of B_i^{-1}, which is the inverse of a 3×3 matrix in $2D$. If the determinant of B_i is too small (e.g. for a node i situated in the corner of the initial fine mesh, with only two neighbors), the linear least square fitting function can lead to a bad interpolation. In this case and for this node (or aggregate) only, a constant least square fitting function is rather used.

This smoother is illustrated in Fig. 6(b) with a $1D$ example and in Fig. 7(c) with a $2D$ example. One can observe in Fig. 6 that there is no difference between the solution smoothed with the constant and the linear least square fit, simply because the points are equidistant in this $1D$ example. In Fig. 7, the difference is small and hard to notice. In practice, it will be shown in the next section that the prolongators smoothed with a constant or linear least square method dramatically improve the convergence rate and computation time, compare to the initial non smooth piecewise constant prolongators. Concerning the smooth prolongators, the linear least square fit will lead to slightly better but comparable results on unstructured grids, as shown in the next section as well. Finally, it is worth mentioning that, in $2D$, the rows of both the prolongators smoothed with the constant or linear least square fit have in average 3.8 non zero entries per line whose sum is equal to one.

4.3 Non nested finite-element-type multigrid (MG - FE)

The prolongators of the second family of preconditioners implemented in FLEXMG are based on a finite-element interpolation between the nodes of the coarse grid and the fine grid. Contrary to aggregation-type multigrid, the construction of the subgrids is not automatic and the meshes of the sublevels must be provided by the user, as illustrated in Fig. 8. These meshes are unstructured, non nested and completely independent from each other. Semicoarsening in the boundary layer is then achieved implicitly during the generation of the coarse meshes since their boundary layers is coarsened more in the direction normal to the wall than in the tangential direction.



(a) Solution smoothed at the two middle nodes with a constant least square fit (AggSmthCLSF).

(b) Solution smoothed at the two middle nodes with a linear least square fit (AggSmthLLSF).

Figure 6: Aggregation-type multigrid - Comparison of two smoothing strategies based on a least square fitting technique and applied after the transfer of the solution from a coarse grid with two aggregates to a fine grid with four nodes ($1D$ example).

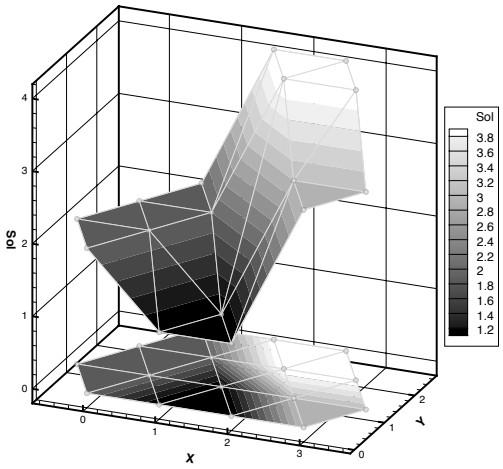
As illustrated in Fig. 9, the prolongated solution in any fine node p is obtained from a linear combination of the solution at the three coarse nodes L_i forming the finite-element T enclosing the fine node p [27]. In order to determine inside which coarse finite-element a fine node lies, a search algorithm inspired from [28] is implemented in FLEXMG. For $2D$ test cases, the rows of the prolongators have therefore three non zero entries whose sum is equal to one, except for the boundary nodes. Moreover, the prolongators of this method do not require any additional smoothing operation since they are already sufficiently smooth by construction.

5 SETTINGS AND RESULTS

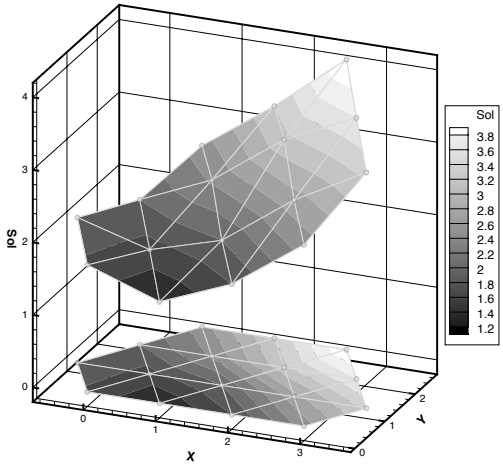
The multigrid preconditioners presented in the previous sections have been tested either as a stand-alone multigrid solver or coupled to a GMRES method. Four multigrid preconditioners are considered here: aggregation-type multigrid without smoothing (MG - Agg), smooth aggregation-type multigrid based on a constant and linear least square fit (MG - AggSmthCLSF and MG - AggSmthLLSF) and finite-element-type multigrid (MG - FE), which makes eight different solvers.

At each level of the multilevel correction scheme, one single pre and post relaxation iteration is applied, using a fixed-point method preconditioned by ILUT. The ILUT matrices are provided by the SPARSKIT package [29] and a $LFIL$ equal to 10 is set. $LFIL$ in this study follows the definition of the SPARSKIT package, i.e. only the largest $LFIL$ elements in every row of L and the largest $LFIL$ elements in every row of U (excluding diagonal elements) are kept. To give a reference, the discretization of the Navier-Stokes equations implemented in SFELES leads to an average number of non-zero entries per line close to 18 in the initial stiffness matrix A.

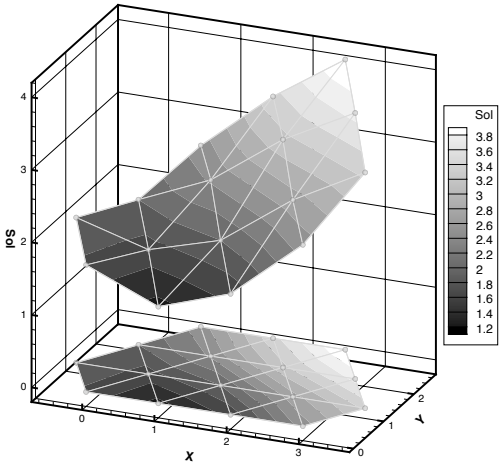
Three sublevels are used in the multilevel correction scheme, dividing each time the number of unknowns by a factor approximately equal to four. At the coarsest level, the problem is solved directly using an exact LU factorization [30]. Finally, an absolute convergence threshold of $\| r^k \| \leq 10^{-12}$ is imposed (computations in double precision).



(a) Non smooth piecewise constant solution (Agg).



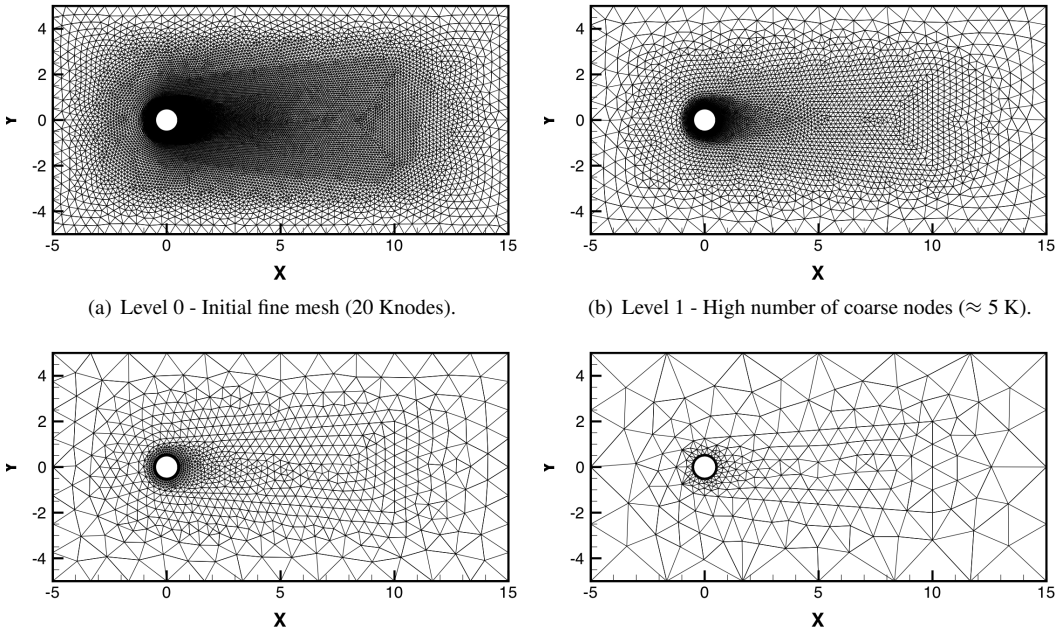
(b) Solution smoothed at each node with a constant least square fit (AggSmthCLSF).



(c) Solution smoothed at each node with a linear least square fit (AggSmthLLSF).

Figure 7: Aggregation-type multigrid - Comparison of two smoothing strategies based on a least square fitting technique and applied after the transfer of the solution from a coarse grid with four aggregates to a fine grid with 16 nodes ($2D$ example).

As mentioned in Section 2, FLEXMG has been linked to a combined finite-element/spectral code called SFELES. This code is able to perform DNS and LES of $3D$ incompressible turbulent flows on unstructured meshes. In this work, the classical test case of the unsteady flow over a circular cylinder at low Re number is considered on a series of increasingly fine anisotropic meshes. First, the accuracy of FLEXMG is checked. Then, the performance of our solvers in terms of convergence rate, computation time and memory consumption is compared with the performance of a direct sparse LU solver called



(a) Level 0 - Initial fine mesh (20 Knodes).

(b) Level 1 - High number of coarse nodes ($\approx$ 5 K).

(c) Level 2 - Moderate number of coarse nodes ($\approx$ 1.25 K).

(d) Level 3 - Low number of coarse nodes ($\approx$ 0.31 K).

Figure 8: Increasingly coarser meshes for the construction of the prolongators in MG-FE.

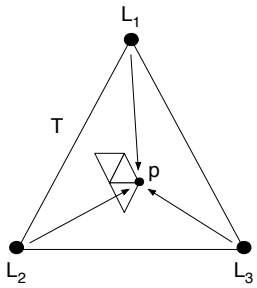


Figure 9: Finite-element interpolation for the construction of the prolongators in MG - FE.

SUPERLU [30] as a yardstick.

5.1 Problem description

A computation of the unsteady laminar $2D$ flow past a circular cylinder at $Re = 100$ is considered in this work [2]. However, the explicit treatment of the nonlinear terms of the Navier-Stokes equations (convection, stabilization and eventually LES turbulence model) in the right-hand side of the linear

system in Eq. 1 preserves the convergence properties of our multigrid preconditioners even for higher Reynolds number flows [12].

Following the description of our code SFELES in Section 2, only mode 0 is required for this $2D$ computation. However, for some implementation reasons, mode $\frac{N}{2}$ is also solved within the same linear system and then discarded. The size of the system associated to this $2D$ problem is thus equal to eight times the number of nodes (real components of u, v, w and p for modes 0 and $\frac{N}{2}$).

A Dirichlet velocity inlet condition is applied upstream of the cylinder while a Neumann outlet condition is applied far downstream. Slip-wall conditions are applied on the sides of the domain (top/bottom). Computations are performed on three meshes (20, 80 and 320 Knodes) to analyze the performance of our solvers. All runs to determine linear solver accuracy and performance were restarted using an already established solution illustrated in Fig. 10.

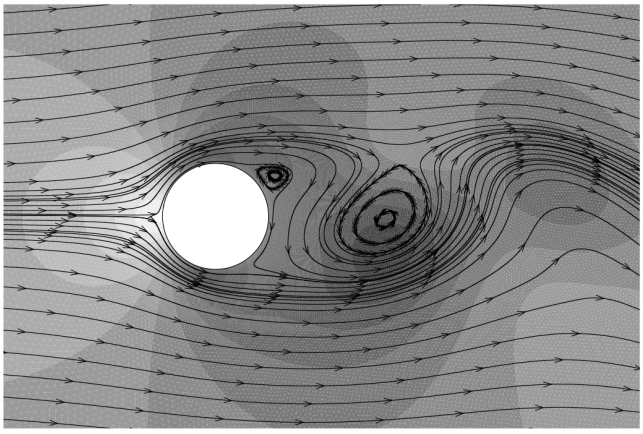
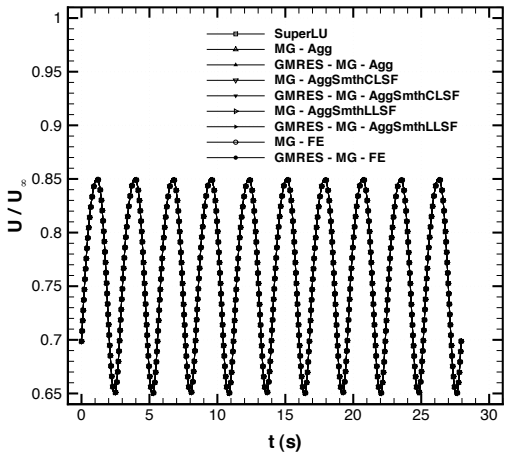


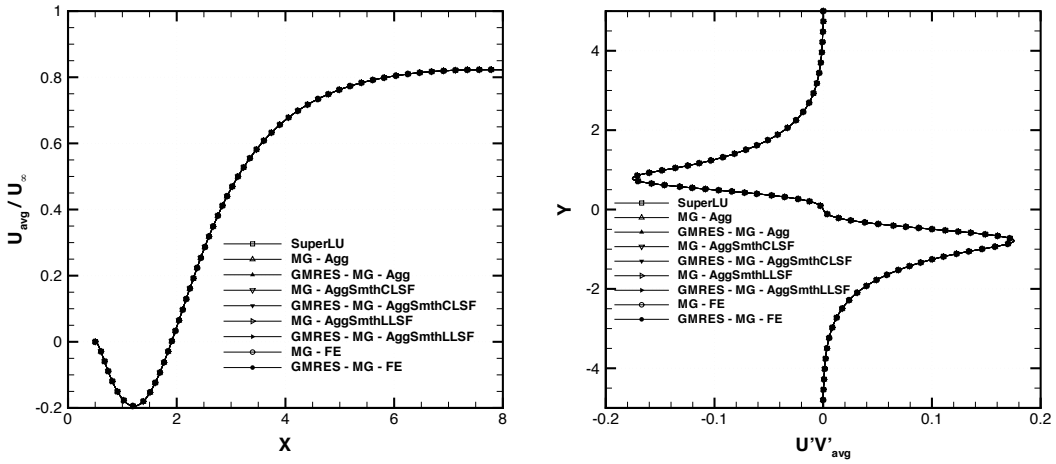
Figure 10: Pressure field and streamlines around a cylinder at $Re = 100$ - 80 Knodes mesh - von Karman vortices.

5.2 Accuracy

First, the solution obtained with the SUPERLU package is compared with the eight iterative solvers preconditioned by multigrid presented in Section 5. The code ran for 55890 time steps with $\Delta t\, U_\infty / D = 0.0005$, which corresponds to 10 complete periodic vortex sheddings. Figure 11(a) shows that all solvers produce the same temporal evolution of the U velocity at a point downstream of the cylinder. In addition to the time-accurate monitoring of a single point, the complete time-averaged flow field is also compared to the direct LU solution. The results both in terms of zeroth-order (mean streamwise velocity, see Fig. 11(b)) and first order (Reynolds stress, see Fig. 11(c)) statistics are found to be identical.



(a) Evolution of U/U_∞ nondimensional velocity at a control point located 5 diameters downstream of the center of the cylinder.



(b) Time-averaged U streamwise velocity along a horizontal line through the center of the cylinder and extending into the wake region.

(c) Time-averaged $U'V'$ Reynolds stress along a vertical line located 1.75 diameters downstream of the center of the cylinder.

Figure 11: Validation of the linear solver accuracy on the 20 Knodes mesh - $Re = 100$.

5.3 Performance

It has been observed that the performance of the eight iterative solvers tested in this work are sensitive to the time step and therefore to the global CFL number used in the simulations. The global CFL number is defined by $\Delta t\, U\, /\, h_{min}$ with U a characteristic velocity of the flow, Δt the time step and h_{min} the smallest characteristic length of the initial finite-element mesh. Decreasing the time

step leads to a better preconditioning (although more iterations are required to simulate the same physical phenomenon). Below a certain value, no more influence of this parameter on the multigrid preconditioning is however noticed. Therefore, a time step equal to $\Delta t \, U_\infty / D = 0.0001$ has been chosen in order to keep the global CFL number around 0.15 for the three considered meshes (20, 80 and 320 Knodes). The final objective is to study the efficiency of the multigrid preconditioners and not the influence of the CFL number on the stability of the temporal discretization used in SFELES. In the next four paragraphs, the convergence rate is analyzed for one typical time step while the computation time is computed over 1000 time steps.

5.3.1 Convergence rate

In addition to the eight iterative solvers preconditioned by multigrid, the convergence rate of a traditional fixed-point method and GMRES method, both preconditioned by ILUT is shown in Fig. 12 (FP - ILUT and GMRES - ILUT). FP - ILUT (our relaxation procedure in the multilevel correction scheme) converges rapidly at the beginning but then stagnates since it fails to remove efficiently the remaining high wave number errors. GMRES - ILUT is already a big improvement over FP - ILUT but still requires too many iterations for this case before reaching the convergence threshold of 10^{-12}. One can also observe that GMRES - ILUT is characterized by accelerations of convergence whenever an eigenvalue of the stiffness matrix is correctly approximated by an eigenvalue of the Hessenberg matrix constructed during the GMRES procedure [3, 5]. Despite its lack of robustness, MG - Agg is already better than GMRES - ILUT but the lack of smoothness of its prolongators is clearly underlined when compared to the other smooth stand-alone multigrid cycles. GMRES - MG - Agg is able to compensate strongly this drawback and illustrates the efficiency of GMRES methods in general. But the smoothing of the prolongators for the stand-alone MG - Agg cycle even leads to a greater improvement (see MG - AggSmthCLSF and MG - AggSmthLLSF). A slight improvement can still be noticed when MG - AggSmthCLSF and MG - AggSmthLLSF are used as a preconditioner of GMRES. As mentioned in Subsection 4.1, few differences can be noted between AggSmthCLSF and AggSmthLLSF in terms of convergence rate, although AggSmthLLSF seems to be slightly better for both our stand-alone multigrid cycle and GMRES method. Finally, the best convergence rate is observed for MG - FE and GMRES - FE. No improvement is brought this time by GMRES, showing the efficiency of the MG - FE preconditioner for this particular test case. It should however be reminded that one of the main shortcomings of stand-alone multigrid cycles is their lack of robustness, which is not underlined in this study when considering the circular cylinder test case.

5.3.2 Computation time

For both SUPERLU and FLEXMG, only the CPU time needed to solve linear systems is shown in Fig. 13. Set-up times (LU factorization, construction of coarse grid levels, prolongators and Galerkin matrices, ...) have not been included as these tasks need to be done at the first iteration of the solver only. All results are normalized with respect to the time required by SUPERLU on the 20 Knodes mesh. SUPERLU is the fastest solver in double precision but its usage is limited by its high memory consumption, as presented in the next paragraph. MG - FE is the fastest iterative solver and could even be faster than SUPERLU for this particular test case if computations were made in single precision with a convergence threshold of 10^{-5} [12].

GMRES - MG - FE is slower than MG - FE because the construction of the Krylov subspace takes some additional time and is not counterbalanced in this case by a lower number of iterations per time

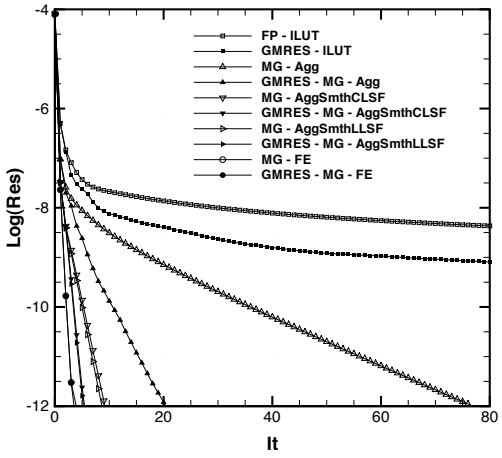


Figure 12: Typical convergence curves of different iterative procedures preconditioned by ILUT and Multigrid (Fixed-Point method, stand-alone multigrid cycle and GMRES) - 80 Knodes mesh.

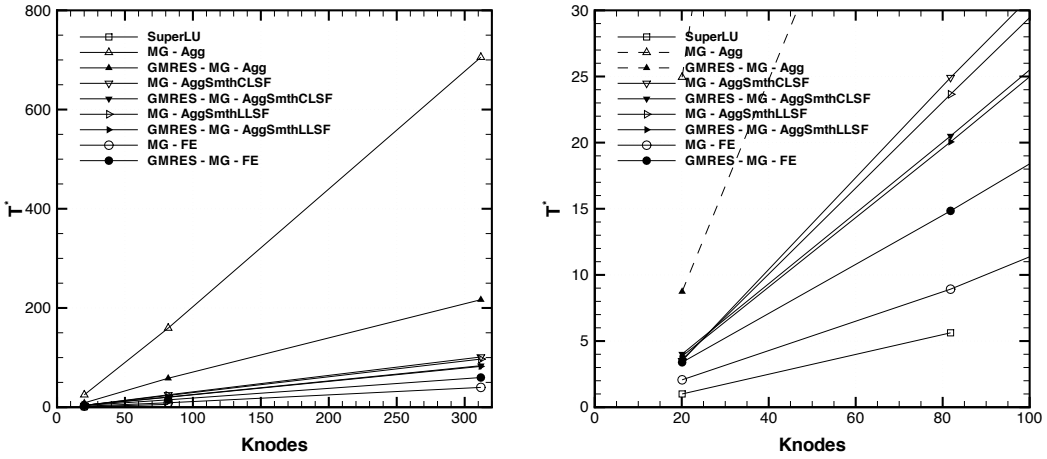
step (see Fig. 12). GMRES preconditioned by either MG - AggSmthCLSF or MG - AggSmthLLSF is faster than their stand-alone multigrid cycle because GMRES improves here the convergence rate and solves the problem in less iterations. As expected, the slowest solver is the stand-alone MG - Agg cycle but nevertheless, it should be pointed out the impressive improvement brought by GMRES for this multigrid preconditioner without smooth prolongators.

5.3.3 Memory consumption

Memory required by FLEXMG and SUPERLU is presented in Fig. 14. All results are normalized w.r.t. the memory required by SUPERLU on the 20 Knodes mesh. One can observe the very high memory consumption of the SUPERLU solver, which prevents to compute the 320 Knodes test case on a machine with 8 GB of RAM. On the other hand, stand-alone multigrid and GMRES solvers exhibit much lower, linear memory requirements. More memory is however required for the GMRES methods compared to the stand-alone multigrid cycles since their Krylov subspace needs to be stored (a maximum of 50 Krylov vectors is set in this study). The MG - Agg preconditioner is also less memory consuming than the three other multigrid preconditioners since its non smooth prolongators and restrictors are sparser. Finally, MG - AggSmthCLSF and MG - AggSmthLLSF both require exactly the same amount of memory but a little bit more than MG - FE because the number of non zero entries per line in their prolongators and restrictors is slightly larger.

6 CONCLUSIONS

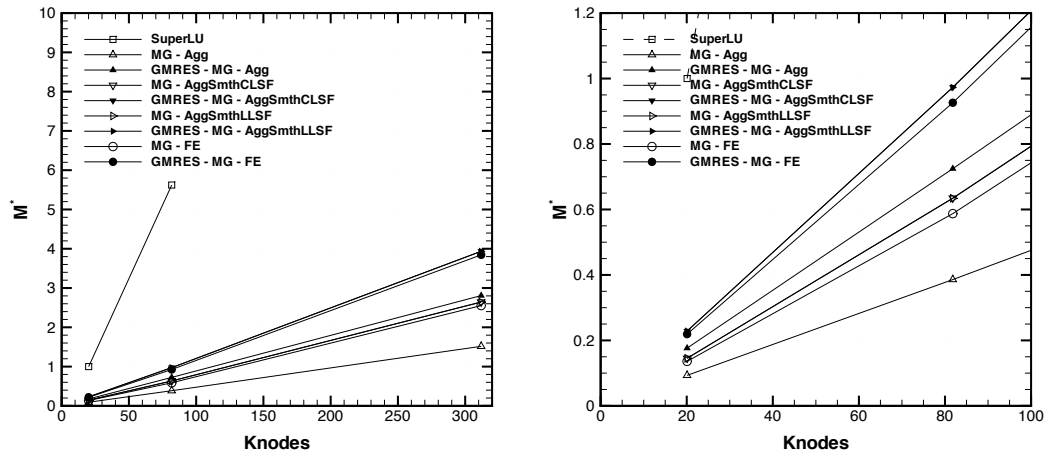
In this contribution, a new library of solvers preconditioned by algebraic multigrid called FLEXMG has been implemented for a spectral/finite-element incompressible flow solver called SFELES. Algebraic multigrid preconditioners can be used as stand-alone solvers or coupled to a GMRES method. Two



(a) Linear system solution time.

(b) Linear system solution time - Zoom.

Figure 13: Linear system solution time normalized with the solution time of the direct sparse LU solver on the 20 Knodes mesh (normalization factor = 0.22 s) - $Re = 100$.



(a) Memory usage.

(b) Memory usage - Zoom.

Figure 14: Memory usage normalized with the memory of the direct sparse LU solver on the 20 Knodes mesh (normalization factor = 766.5 MB) - $Re = 100$.

families of prolongators are available in FLEXMG. The first one is based on smooth aggregation while the second one is based on non nested finite-element interpolations.

Already, the performance of FLEXMG in terms of memory consumption and computation time are promising. Computations that were out of reach because of lack of memory when using an exact

LU factorization, are now allowed in a reasonable computation time. For the $2D$ flow past a circular cylinder at $Re = 100$, an absolute convergence threshold of 10^{-12} can be reached in four to five iterations when considering a stand-alone multigrid cycle or GMRES method preconditioned by finite-element-type or smooth aggregation-type multigrid.

Despite the shortcomings of algebraic multigrid preconditioners for convection-dominated problems, theses preconditioners are particularly well adapted for our CFD code called SFELES since convection is not treated implicitly. Consequently, only elliptic and parabolic terms contribute to the linear systems. However, the systems arising from the discretization in SFELES are much stiffer than for a Stokes problem because of the highly anisotropic stretched meshes required to resolve viscous boundary layers.

As far as the aggregation-type multigrid is concerned, initial piecewise constant prolongators can be smoothed by either a constant or a linear least square fitting functions. The importance of smoothing the prolongators has also been underlined in this work. Indeed, only smooth aggregation-type multigrid is able to compete with the finite-element-type multigrid which is already smooth by construction. The latter is still faster and a little bit less memory consuming than smooth aggregation-type multigrid. However, coarse meshes for the non nested finite-element-type multigrid still need to be provided by the user while sublevels are automatically generated for the aggregation-type multigrid.

Moreover, multigrid and GMRES methods are also thought to cure their respective deficiencies, i.e. lack of robustness for the former and lack of scalability for the latter. This lack of robustness for stand-alone multigrid solvers was not underlined in this study when considering the $2D$ flow past a circular cylinder but this should become more obvious with other more complicated test cases.

Finally, this contribution is a first step in the development of a scalable parallel linear solver and sets the framework for continued study and improvement of multigrid preconditioners for CFD applications.

Acknoledgments

The first author is supported by a FRIA fellowship from the French Community of Belgium. Developments in SFELES were supported by the Higher Education and Scientific Research Division of the French Community Government at ULB (ARC project 02/07-283). Computations are performed amongst other on the Linux clusters ANIC4 and ANIC5, funded by several contracts (`http://anic4.ulb.ac.be` and `http://anic5.ulb.ac.be`). We are also very thankful to Dr. David Vanden Abeele and Mr. Todd White (former researchers at the von Karman Institute for Fluid Dynamics) for their precious advices and to Prof. Yvan Notay (Université Libre de Bruxelles) for his review and comments on this work. Dr. Fabien Delalondre (Rensselaer Polytechnic Institute) is also warmly acknowledged for his suggestions about some possible smoothing strategies for the aggregation-type multigrid.

References

[1] Rasquin M, Deconinck H, Degrez G. FLEXMG: A new library of multigrid preconditioners for a spectral/finite element incompressible flow solver. *International Journal for Numerical Methods in Engineering* 2010; doi:10.1002/nme.2808.

[2] Larson G, Snyder D, Vanden Abeele D, Clees T. Application of single-level, pointwise algebraic, and smoothed aggregation multigrid methods to direct numerical simulations of incompressible turbulent flows. *Computing and Visualization in Science* 2008; **11**(1):27–40.

[3] Saad Y, Schultz M. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal of Scientific and Statistical Computing* 1986; **7**(3):856–869.

[4] Sleijpen GLG, van der Vorst HA. An overview of approaches for the stable computation of hybrid BiCG methods. *Appl. Numer. Math.* 1995; **19**(3):235–254. Special issue on iterative methods for linear equations (Atlanta, GA, 1994).

[5] Saad Y. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2003.

[6] Van der Vorst H. *Iterative Krylov methods for large linear systems*. Cambridge University Press, 2003.

[7] Briggs WL, Henson VE, McCormick SF. *A multigrid tutorial: second edition*. Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2000.

[8] Trottenberg U, Oosterlee C, Schüller A. *Multigrid*. Academic press, 2000.

[9] Snyder DO. A parallel finite-element/spectral les algorithm for complex two-dimensional geometries. PhD Thesis, von Karman Institute for Fluid Dynamics and Utah State University 2002.

[10] Detandt Y. Numerical simulation of aerodynamic noise in low mach number flows. PhD Thesis, von Karman Institute for Fluid Dynamics and Université Libre de Bruxelles 2007.

[11] Vanden Abeele D, Degrez G, Snyder D. Parallel turbulent flow computations using a hybrid spectral/finite-element method on Beowulf clusters. *Proceedings of ICCFD3* 2004; **30**:50–55.

[12] Rasquin M, White T, Degrez G, Deconinck H, Vanden Abeele D. Development of an aggregation/geometric multigrid solver for large-scale cfd calculations. *Proceedings of the 15th Annual Conference of the Computational Fluid Dynamics Society of Canada*, Toronto, 2007.

[13] Van der Vorst H, Vuik C. GMRESR: a family of nested GMRES methods. *Numerical Linear Algebra with Applications* 1994; **1**(4).

[14] Ruge J, Stüben K. Algebraic multigrid. *Multigrid methods* 1987; **3**:73–130.

[15] Vaněk P, Mandel J, Brezina M. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing* 1996; **56**:179–196.

[16] Wagner C, fur wissenschaftliches Rechnen IZ. Introduction to algebraic multigrid. Course notes, Universitat Heidelberg 1998.

[17] Notay Y. Algebraic multigrid and algebraic multilevel methods: a theoretical comparison. *Numerical linear algebra with applications* 2005; **12**(5-6):419–451.

[18] Notay Y. Aggregation-based algebraic multilevel preconditioning. *SIAM Journal on Matrix Analysis and Applications* 2006; **27**(4):998–1018.

[19] Muresan A, Notay Y. Analysis of Aggregation-Based Multigrid. *SIAM Journal on Scientific Computing* 2008; **30**(2):1082–1103.

[20] Wabro M. AMGe–Coarsening Strategies and Application to the Oseen Equations. *SIAM Journal on Scientific Computing* 2006; **27**(6):2077–2097.

[21] Karypis G, Kumar V. METIS a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices version 4.0. *University of Minnesota, Department of Comp* 1998; .

[22] Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing* 1999; **20**(1):359.

[23] Janka A. Smoothed aggregation multigrid for a Stokes problem. *Computing and Visualization in Science* 2008; **11**(3):169–180.

[24] Vaněk P, Brezina M, Mandel J. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematik* 2001; **88**(3):559–579.

[25] Stüben K. A review of algebraic multigrid. *J. Comput. Appl. Math.* 2001; **128**(1-2):281–309, doi:http://dx.doi.org/10.1016/S0377-0427(00)00516-1.

[26] Gee M, Hu J, Tuminaro R. A new smoothed aggregation multigrid method for anisotropic problems. *Numerical Linear Algebra with Applications* 2009; **16**(1).

[27] Issman E. Implicit solution strategies for compressible flow equations on unstructurd meshes. PhD Thesis, von Karman Institute for Fluid Dynamics and Université Libre de Bruxelles 1997.

[28] Löhner R, Ambrosiano J. A vectorized particle tracer for unstructured grids. *J. Comput. Phys.* 1990; **91**(1):22–31, doi:http://dx.doi.org/10.1016/0021-9991(90)90002-I.

[29] Saad Y, for Supercomputing Research C, Development, of Illinois at Urbana-Champaign U. SPARSKIT: A basic tool kit for sparse matrix computation 1994. URL `http://www-users.cs.umn.edu/~saad/software/SPARSKIT/sparskit.html`.

[30] Demmel J, Eisenstat S, Gilbert J, Li X, Liu J. A Supernodal Approach to Sparse Partial Pivoting. *SIAM Journal on Matrix Analysis and Applications* 1999; **20**:720–755.