Research Article

Parallel Mesh Adaptive Techniques for Complex Flow Simulation: Geometry Conservation

Angelo Casagrande,¹ Pénélope Leyland,¹ and Luca Formaggia²

¹ GR-SCI-IAG, STI, EPFL, Station 9, 1015 Lausanne, Switzerland

² MOX, Department of Mathematics “F. Brioschi”, Politecnico di Milano, Piazza L. Da Vinci 32, 20133 Milano, Italy

Correspondence should be addressed to Angelo Casagrande, angelo.casagrande@epfl.ch

Received 27 April 2012; Revised 16 October 2012; Accepted 22 October 2012

Academic Editor: Hing Kai Chan

Copyright © 2012 Angelo Casagrande et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Dynamic mesh adaptation on unstructured grids, by localised refinement and derefinement, is a very efficient tool for enhancing solution accuracy and optimising computational time. One of the major drawbacks, however, resides in the projection of the new nodes created, during the refinement process, onto the boundary surfaces. This can be addressed by the introduction of a library capable of handling geometric properties given by a CAD (computer-aided design) description. This is of particular interest also to enhance the adaptation module when the mesh is being smoothed, and hence moved, to then reproject it onto the surface of the exact geometry.

1. Introduction

The accuracy of a numerical simulation is strongly dependant on the distribution of grid points in the computational domain. For this reason grid generation remains a topical task in CFD (computational fluid dynamics) applications. Prior knowledge of the flow solution is usually required for a grid to be efficient, that is, matching the features in the flow field with appropriate grid resolution. This, however, may not be available, requiring human intervention in analysing the results of an initial solution, going back to the preprocessing stage, and taking an educated guess at how the mesh should be modified (Figure 1). Alternatively, a generally fine grid over most part of the domain is generated to obtain a relatively good solution. Both, the above cases however require excessive time, effort, and computational resources.

Let us consider the case with manual intervention by the user. This step can be automated by adaptation, whereby the flow solution is analysed automatically, following some predefined criteria, and the grid resolution adjusted to the problem. The use of such techniques allows for computationally precise distribution of grid points (rather than

eye precision), and for extremely reduced user intervention, thus addressing the time and effort issues. It also resolves problems related to computational time and costs, as the adapted grid can have fewer overall points, with similar resolution in areas of interest, than an unadapted fine mesh.

The adaptive method used here is that of h-refinement as described in [1], with *a posteriori* error estimation and physical adaptation criteria.

One of the major drawbacks in most of the adaptation techniques resides in the projection of the new nodes created, during the refinement process, onto the boundary curves and surfaces defining the original geometry. Without this step, the adaptation process will be limited in its potential use. It will still be able to capture flow features such as shocks and adapt to geometries with little or no gradient, but will need a “final” mesh resolution where geometric features are most important. This last point clearly defeats an important aspect of mesh adaptation, as it requires prior knowledge of the importance of such features, and of what mesh size should be considered as “final.” Ideally, the best solution would be to integrate the CAD system used to generate the original geometry, which is unfeasible since this may change from one project to the next. This can be overcome by the introduction

FIGURE 1: Manual adaptation process.

of a library capable of handling geometric properties given by a CAD description.

For large-scale problems, these algorithms require an efficient and scalable implementation of calculation techniques on unstructured grids, therefore parallel codes have to be developed. Data, including the grid's geometry description, the successive adapted grid structures, the solution solver's Jacobian matrix, the solution state variable, and work vectors, must be partitioned among the processors in order to minimise the amount of intraprocessor communications. The definition of a distributed data structure, well-suited for all the computational phases, is the first problem to be addressed. The meshes change during the solution progress, hence a dynamic partitioning is required to rebalance the separate partitions on the chosen number of processors [2].

Similar techniques have been mainly implemented for mesh adaptation by either global remeshing, where the entire mesh is re-generated through the use of a mesh metric, or local remeshing, where only a part of the mesh is re-generated [3]. Other works in recent years have also used CAD description to conserve geometry representation during the adaptation procedure [4, 5]. Other techniques where the surface is approximated by means of interpolation have also been experimented in recent years [6, 7]. Parallel mesh adaptation has also been proficiently implemented for various problems; however, geometry approximation is the most common approach [8].

2. Geometry Conservation

The insertion of a library capable of making use of the geometric data from a commercial CAD package is essential for the accuracy of calculations. However, this requires particular care and attention from the initial design stages. It will be seen later in the description of the methodology for implementation, how every step from the conceptual design

FIGURE 2: Brief schematic of the geometric library placement.

down to the mesh generation is important for the success of the adaptive projection procedure.

2.1. CAD Integration. In order to be as general as possible in the capability of gathering the data from CAD/CAM packages, it is important to have a library that is able to make full use of CAD definitions. Hence, the use of B-splines and NURBS (non-uniform rational B-splines) must be an integrating part of such a library.

The possibility of linking directly with a commercial design software has been discarded as it would create a dependency to it. This would be disadvantageous both financially and in case the distribution would cease. It may also be more complex to implement than one would think, as source codes are generally unavailable, which implies an external call to carry out the point projection.

On the other hand, the development of such a library is not a trivial task. It was therefore chosen to address the problem with the use of a library of C functions to perform the necessary operations on NURBS geometries. This consists in the SINTEF Spline Library (SISL), developed at the Scandinavian research group SINTEF, and available under the GNU General Public License. It is particularly suitable since it allows also the handling of interaction between implicit geometric representations such as planes, and tori, and NURBS, which is very popular in modern CAD/CAM systems. A general view of where the geometric toolkit interacts with the adaptation module is shown in Figure 2.

Although NURBS have increased precision, they also have disadvantages when compared to B-splines, it is therefore desirable to use the latter when the option is available. Due to the vast amount of literature on the subject, for an in-depth explanation and background of curves and surfaces in CAGD we refer the reader to [9–11]. Other systems to interact with the geometry, such as CAPRI [12], could also be considered for type of task.

2.2. Implementation. The addition of such a library is not a simple task, not only in terms of programming and data structures, but primarily in terms of different cases that can and will occur, such as the intersection between surfaces. A more detailed schematic view of where the projection algorithm is placed inside the source code is shown in Figure 3. Here, we can see how and when the projection

FIGURE 3: Brief schematic of the reprojection algorithm placement.

algorithm comes into play, and also some of the data it needs in order to work. Since this addition has been carried out on a parallelised source code [2], no specific intervention was necessary for parallelisation. Each parallel computing node will call the projection algorithm either after the addition of new nodes, or after the smoothing procedure (reprojection), for its part of the mesh. It should be noted that the sizes of these data files are extremely compact and have no significant impact in disk usage, unlike grid files and, therefore, can be easily copied to each computing node at the beginning of the computation. In the next paragraphs, we shall describe in more detail the algorithm developed for the projection decision making, and all the necessary conditions for this to work.

First of all we consider the patch structure that is inherent in the adaptation module. The boundary description is defined at the lowest level with the help of faces (or segments in 2D), which are grouped in patches. Therefore, each patch contains a specified number of faces and corresponds in fact to a surface in the CAD definition of the geometry. This is a fundamental aspect of the projection algorithm, since if more than a surface belonged to a patch, it would be much more complex to tell which surface to project the new node is onto. Patches work in both three dimensions and two dimensions. In the former they described the finite discretisation of a surface, in the latter segments describe the boundary.

The CAD geometry descriptions are divided in a curves and surface data files, which are linked to two other files with the number of entities in each. These last two files also

contain the individual names of each geometric entity. The surface file contains also information regarding the curves that define each surface, in particular the number of curves and the identifying number of each.

At this point curves, in 2D, and surfaces, in 3D, are matched to patches with the use of boundaries file information. Segments that belong to the surface and that are marked for refinement are then identified. Note that this step is straightforward in 2D, where a segment may only belong to one boundary and is defined as a face itself. Hence, it will be associated to a distinct single curve.

In 3D things become a little more complex since a segment is the interface of two surface faces. This limitation is assumed to be always true, since only a structure of zero thickness, at the point of intersection with other two surfaces, may produce an ambiguous segment belonging to more than two faces. Moreover, it would be undesirable to model an unrealistic feature of this type. The complexity resides in the parenting of the segment, since the faces it belongs to could lie on two different surfaces, in which case a decision must be made onto which surface the new node should be projected.

The problem can be clearly seen in Figure 4. Here the new nodes have been projected to the surface defining the exit, which is flat; hence, they lie in the plane at the exact midpoint of the segment they refined.

This would be correct, if it was not for the fact that the new node splits a segment that lies on the frontier of two patches, and the parent face to which it should have been referred for projection should have been the one describing

FIGURE 4: Projection to the wrong surface for segments on the intersection between patches. (a) Detail of the channel exit with a false step created. (b) Front view of the channel exit. (c) Relative solution with erroneous results due to wrong placement of new nodes.

FIGURE 5: Projection to the defining curve of the patches frontier. (a) Detail of the channel exit. (b) Front view of the channel exit. (c) Relative solution with correct results.

the wall of the channel. In fact, if this had been done to begin with, the problem would go unnoticed.

This is overcome with the information gathered earlier on the curves defining the surfaces. The curve that describes the intersection of the two patches is taken as the geometry definition to project onto. The result of this is shown in Figure 5. It can be noticed how the exit is also less refined. In the previous case, the refinement was due to the flow altered by the coarse mesh, which created an artificial step. If the refinement nodes had been placed correctly, the result would have been a smoother flow due to the exact geometry

FIGURE 6: Blade in cylindrical channel.

FIGURE 7: Detail of the blade from above with half-wall channel blanked.

representation, which presents no such artificial features created by an incorrect (or too coarse) mesh.

Particular care has to be taken when preparing the curves and surfaces however. This method requires each surface to have only one curve in common with another surface. In this way the frontier curve between patches can be clearly found.

Let us take as an example a blade in a cylindrical channel as shown in Figure 6. In the case of inviscid flow this would be twice as much the needed geometry, in fact we could cut along the xz plane and add a symmetry plane. It is a good example though, of the type of problems that can be encountered above, and gives a good idea of the type of thought and preprocessing that must go into the design approach.

From a design point of view it would be natural to draw the blade as two surfaces with the leading edge and trailing edge in common. This however would create the condition described above and, hence, is not viable. Therefore, the blade is split in four surfaces as shown in Figure 7. In this way the problem is solved. There remains a problem with the cylinder walls. If we start from the idea that the walls are defined as two surfaces cut by the xz plane, then each surface would intersect twice with the two vertical blade surfaces on their side of the plane. A solution could be that of splitting again the blade in the xy plane. This would not be the good solution though, as the two wall surfaces would still touch at the top and at the bottom in two distinct places with two intersection curves that have no continuous solution as they are disconnected and parallel.

Hence the walls are split in four surfaces, at the xz and xy planes through the centre the circle defining the cylinder. An attentive eye will immediately notice that the two top wall

FIGURE 8: Curve connecting the top wall surfaces divided by the blade. The black dots are existing segment nodes, and the empty dots are the new nodes to be placed.

FIGURE 9: Bump in a channel—surface initial mesh and patches shown in different colours with one side blanked, viewed from the down side.

surfaces will still have two intersections in common which are separated by the blade. This is a different case from the previous one. Here a single curve can be used to define the two intersections. If we examine Figure 8, we can see that the curve that describes the two intersections is continuous. In this particular case it is allowed since the points are projected onto the same curve, and since the point to project is at the midpoint of the segment there is no risk of projecting it in onto the wrong intersection.

A further feature, that is an important part of the algorithm, is that of being able to specify the surfaces to project onto. This has been implemented only for 3D applications since it has less (if any) appeal in two dimensions. The choice is made through the surfaces information file, where only the selected surfaces and their defining curves are taken into consideration. In the latter stages only on these entities the projection is carried out. It can be of particular interest in cases such as that shown in Figure 9. Here it is obvious that the only surface that will need point projection, to insure the geometry is followed, is the bump, since all other surfaces are planar. The defining curves of the surface will also be needed, hence the information of all other surfaces must be retained, but not the surfaces definitions or those of curves that are not frontiers of the selected surfaces.

2.2.1. Reprojection. Finally, the projection algorithm is generalised for all elements that lie on the boundary. This is used in the case where the smoothing routine is selected. In fact, the smoothing may cause some problems when used, particularly when the starting mesh is very coarse and a few adaptation steps are necessary. In these cases the multiple iterations of the smoothing procedure may cause the boundary nodes to slowly drift away from the geometrical definition, despite all the checks in place to avoid this [1].

(a)

(b)

FIGURE 10: Projection curves and surfaces during the smoothing iterations—(a) without projection of all nodes, (b) with projection after each smoothing iteration.

Hence, the generalised projection routine is called at the end of each smoothing iteration to check that all nodes are correctly placed on the geometry and reposition them when this is not the case. The effect of this can be clearly seen from Figure 10, where the smoothing iterations pushed the nodes away from the surface (a), and how the integration of the projection places them back onto the surface after each iteration (b).

2.2.2. Point Projection. This can be summarised as the search of the minimal distance from the point P in space to the geometric entity onto which it should lie. The possibility of the point being at the same minimal distance between more than one entity is cancelled. This is done by identifying the node as belonging to a specific geometric entity, with the use of its parent nodes, as explained earlier. There remains therefore only the problem of finding the point on the geometric entity for which the distance to the point in space is minimised. This problem can be approached in various ways and is most commonly referred to in the literature as the foot point problem [13]. A detailed description of a few commonly used methods, including the ones implemented in the NURBS library, can be found in [14, 15].

Three possible choices for carrying out such a task are given in the library. The first consists in a Newton iteration on the distance function between the point and the curve/surface, though if a bad initial guess is made then the iteration may give a local rather than a global closest point. However, since in our case the point is already close to the surface, the local minimum problem is not a concern. The second option is based on the first since it makes use of the same algorithm, but rather than having a user input initial guess, it takes the closest control point and estimates the corresponding curve parameter value or pair for the

FIGURE 11: Grids and density solution downstream of the blade—(a) and (b), first step with and without projection, respectively.

surface case. The value, or values, obtained is then used as the starting point for the iteration. The last option in the library is that of entering the equation of the curve/surface into the equation of a circle/sphere having P as its centre and radius zero. This results in a one-dimensional curve/surface for which the minima is found [16].

2.3. Applications. In order to verify the effectiveness of the procedures introduced in this section, three test cases were considered. The first is the earlier mentioned blade in a cylindrical channel (Figure 6). A second test case is represented by a 10% bump in a channel as shown in Figure 9. In both these cases the geometry is created from scratch. The third and final configuration examined is that of the ONERA M6 wing, where the geometry is reconstructed from an available mesh. The details of each calculation are presented hereafter, together with the solutions which were obtained using THOR [17]. The geometries are described with nondimensional units in all cases.

2.3.1. Pipe Blade. This geometry has been taken as example for describing the design approach, and proves a good test case for its simplicity in the design process and yet complexity in the geometry conservation during the adaptation process. The initial flow conditions are set to Mach number 2.0 at zero angle of attack in order to obtain some interesting flow features. Here we look principally at the adaptation projection process rather than the solution, hence only a first-order scheme is used.

The blade is described by two arcs of 4 units chord length and has a maximum width of 0.6, whilst the distance from the tips of the blade to the ends of the pipe is 28 units. Finally

FIGURE 12: Grids and density solution downstream of the blade—(a) and (b), second step with and without projection, respectively.

the diameter of the pipe is of 4 units. The initial coarse grid is uniform and contains 2486 nodes, 10683 tetrahedra, and 2384 triangular surface elements (Figure 6).

Four distinct adaptation steps were then performed in series, each restarting with the previous steps adapted mesh. The adaptation is performed with respect to the difference in density on the segment. An identical procedure was also done without the projection algorithm and used as comparison. This last computation serves also as illustration of initial mesh dependency when no projection is applied.

Considering the steps of the adaptation cycles reported in Table 1, the surface grids show remarkable differences. As expected, the meshes obtained from the algorithm that makes use of the geometry definition are much smoother. In particular, grid-induced flow features that may come from an excessively coarse initial mesh are lost after one or two steps. Whereas in the case of no projection these features are further refined to no avail. Details of this are shown in Figures 11 and 12. The effect is clearly shown with the smoother solution, and only some further refinement, in Figures 11(a) and 12(a), where the projection algorithm is selected. In Figures 11(b) and 12(b) it is noticeable how, despite the refinement, the original grid is still driving the solution, and because of this it is refined also in those areas where an artificial step is created by the mesh coarseness. This effect explains the values in Table 1, which become increasingly higher for the case without projection respect to that projecting, as more refinement steps are carried out.

The percentage change was taken as $\% \text{ change} = (N_{wp} - N_p)/N_{wp}$, where wp and p stand for without and with projection, respectively, and N is the number of the characteristic size of the mesh being compared, that is, nodes, elements, and surface elements. This is expressed graphically

TABLE 1: Grid sizes for adapted pipe blade calculations: with and without projection algorithm.

Adaptation step	With Projection			Without Projection		
	N_{nodes}	$N_{elements}$	N_{surf_el}	N_{nodes}	$N_{elements}$	N_{surf_el}
First	6 024	29 110	4 458	6 011	29 077	4 440
Second	20 755	110 287	9 790	21 711	115 361	10 326
Third	86 876	493 771	22 264	95 565	538 420	27 318
Fourth	314 652	1 854 594	42 718	381 353	2 219 641	69 152

TABLE 2: Grid sizes for adapted channel bump calculations and relative flow solver step—with and without projection algorithm.

Adaptation/ flow-step	Projection			No projection		
	N_{nodes}	$N_{elements}$	N_{surf_el}	N_{nodes}	$N_{elements}$	N_{surf_el}
First-flow 1	600	2 058	908	600	2 058	908
Second-flow 2	1 731	7 390	1 804	1 717	7 318	1 800
Third-flow 3	5 104	25 204	3 506	4 952	24 373	3 430
Fourth-flow 5	13 362	71 143	6 388	15 625	84 313	6 716
Fifth-flow 6	31 933	179 573	10 236	48 158	273 635	12 830

FIGURE 13: Percentage difference between mesh characteristic sizes of adaptations with and without projection.

in Figure 13, where it is clear to see the effect of the projection algorithm, particularly on the number of surface elements, which is where the procedure has the greatest influence.

Furthermore, by overlapping the original grid to the adapted grid after only two steps, there is a remarkable difference between the two. This is clearly shown in Figure 14, with only two halves of one blade wall selected. From the internal side viewpoint of the blade only the initial grid is visible, with a few segments from the adapted grid, and vice versa. The reason for the visibility of some segments is due to the fact that some of the original segments may lie on the exact geometry. A detail of the gap between original and adapted grid at the top is also shown in Figure 14(c).

Although the solution itself might not be of particular interest, it is important to notice how and if this is improved by the projection process in order to verify its effectiveness.

A closeup of the zone at the location of the blade in the pipe is shown in Figures 15 and 17, with a further detail of the area behind the blade in Figure 16. In particular from Figure 16 it is clear to see the projection on the channel wall and its effects. The solution examined is the fourth and final step of the case considered here, and shows the variation in density of the fluid flow. As expected, a much smoother and precise solution is obtained with the projection algorithm, confirming the method to effectively ameliorate the result.

2.3.2. Bump in a Channel. This is a classical test case which offers the possibility to test all aspects of the adaptation process including the solution. The initial grid is that shown previously in Figure 9, and it also allows us to test whether the feature of projecting only onto selected surfaces works correctly.

The geometry consists in a 10% circular bump in a channel, of unit chord length and 0.1 maximum thickness. Height, length and width of the channel are respectively 1, 3 and 0.2 units. The free-stream velocity is set to Mach number 0.675 at zero angle of attack, which causes the flow to be transonic. The initial grid consists of 184 nodes, 426 tetrahedra, and 364 triangular surface elements.

At a first glance it is clear to see the differences with the prior test case from Table 2. Here the main variance is between the number of nodes and elements, but only slightly and at the last stage of adaptation for the surface elements. This is normal since only a slight part of the surface mesh is projected onto the geometry. Also, unlike the pipe blade where only a first order scheme was used, here the approach was changed.

For compressible CFD, when performing a steady state or transient solution computation, the following steps are usually carried out.

- (1) On the initial grid, perform a first-order spatial accurate solution technique to obtain a provisional approximate solution. This solution will have an initial position of shock waves and discontinuities,

FIGURE 14: Detail of the difference between adapted projected mesh and the original grid of one side of the blade, view from—(a) outside, (b) inside and (c) top with gap detail.

FIGURE 15: Details of the solution in proximity of the blade and downstream, viewed from underneath—(a) with projection, (b) without.

but will be very dissipative and not capture finer details.

- (2) This first solution can then be used for a first step of adaptation and/or serve as the initial condition for a further number of iterative cycles using a second-order scheme, which will resolve more precisely the solution. At this point the mesh can be further adapted.

This procedure is standard and removes some stiffness from the problem resolution. It is also known as “Blending” first and second order solutions.

FIGURE 16: Detail of the area behind the blade of Figure 15, showing initial grid (a) dependency with artificial step created (b), and no dependency with projection (c).

FIGURE 17: Detail of the solution in proximity of the blade and downstream, viewed from the side—(a) with projection, (b) without.

Initially a first order scheme is used to advance the solution and adapt the mesh, up until the third adaptation step (adaptation and flow steps 1–3). After this refinement the solution and grid obtained are used to start the second order solution, without adaptation (flow step 4). Once a solution is obtained, it is then used to start the following adaptation steps with the second-order solutions (adaptation steps 4, 5, and flow steps 5, 6). In all steps the adaptation is performed with respect to the difference in density on the segment.

A detail of the difference in the projected part of the grid is shown in Figure 18. In particular the gap between the original coarse mesh and the adapted projected grid can be clearly seen. As for the previous case, in the proximity of segments of the original grid that lie on the defining geometry, adapted segments are also visible.

The grids and solutions produced from the various steps are shown in Figures 19 and 20, with solution field for the Mach number and C_p isolines of the final step shown in detail in Figure 21. Here similarities in the grids can be seen until the second order solution adaptation steps

FIGURE 18: Details of the differences between original and adapted projected grid for the channel bump—(a) view from the side, (b) tilted view from underneath the bump.

FIGURE 19: Adaptation/flow steps with (left) and without (right) projection—(a) first, (b) second, and (c) third. Solution field is C_p .

in (Figure 20(b)), however the solution shows discrepancies between the adapted projected and that without, already from the second step (Figure 19(b)). In particular, since without projection the original grid drives the solution, two regions of shock seem to appear in the second step, of which the first lies at the interface of the original coarse elements, which define a sharp feature rather than a smooth curvature of the bump. Although this is greatly reduced when switching to the second order solution (Figure 20(a)), large differences in both the adapted grids and the solution are

FIGURE 20: Adaptation/flow steps with (left) and without (right) projection—(a) only flow calculation step 4, (b) adapt fourth-flow 5, and (c) adapt fifth-flow 6. Solution field is C_p .

clear in the final steps of the computation, with a correct shock placement for the one using the projection algorithm, and a corrupted one for the standard midpoint scheme.

2.3.3. ONERA M6 Wing. Unlike the previous test cases, an exploitable mesh is used to recover a usable set of curves to reconstruct the geometry. Although the geometrical information of the ONERA M6 wing can be easily obtained, this test case is also used as basis to check the feasibility of recovering a usable geometry from an available mesh. The mesh used to recover the initial set of curves is shown in Figure 22, in which the different patches of the mesh are also shown.

The procedure consisted first in recovering the nodes, at the sections described by the patches borders, which are then used as points to construct the curves. The curves are then split accordingly to the guidelines given earlier in Section 2.2, in such a way that no two surfaces have more than one curve in common. The recovered geometry consists of 8 surfaces and 15 curves, the root chord length is 10 units long with the centre of the domain placed at the tip of the wing section on the symmetry plane.

Figure 23 shows the new patch structure on the wing, which coincides with that of the surfaces, with details of the tip and tail patches with mesh. Finally, the far field is described by a half-sphere of radius 12.5 times the root chord. This leads to the initial mesh shown in Figure 24.

FIGURE 21: Final solutions with (left) and without (right) projection. (a) Mach number solution field and C_p isolines and (b) details from the side.

FIGURE 22: Original ONERA M6 wing grid and patches.

The free-stream velocity is set to Mach 0.84 and the angle of attack at 3.06° , causing the flow to be transonic. The initial grid consists of 91 379 nodes, 477 262 tetrahedra, and 38 926 triangular surface elements.

For all the following examples, an initial first-order solution is carried out to start off the computation, and used in the second stage as a restart for a second-order solution. This solution then becomes the base for all subsequent meshes in the first adaptation step. The first and second order solutions are shown in Figure 25.

A first adaptation step is then performed, with respect to the difference in Mach number. The smoothing factor is kept to a minimum number of iterations and a relatively high boundary constraint in order to avoid as much as possible the unprojected solution to be influenced by excessive smoothing. The results of a computation with and without the projection algorithm are shown in Figure 26, where an initial difference in the solution due to the smoother surface, is already visible in the areas of greater surface gradient, that is, at the leading edge and at the tip. For completeness the grids are also shown, although the differences are minimal

TABLE 3: Grid sizes for CAD projection after 1 adaptation step with 16 processors. ONERA M6 wing.

Projection type	N_{nodes}	$N_{elements}$	N_{surf_el}
No projection	171 219	955 543	57 700
CAD new nodes	171 194	955 793	57 660
CAD new and smoothing	171 192	955 863	57 658

at this stage, and can only be noticed at the leading edge (Figure 27).

Adapting a second time gives us a clearer view of the differences on the solution, due to the effect of projecting the nodes onto the surface. The overall solution gives a relatively good result as we can see from Figure 28, but even so, the “noise” produced by the original grid size is clear to see at the tip and at the leading edge in particular. Finally, Figure 29 illustrates the iso-Mach lines and a detail of the solution at the leading edge. In both cases, we can see a net improvement in the computation where the projection is included.

3. Performance Aspects and Parallelisation

As described in [1] the time for the ensemble of adaptation processes is approximately 2% of the total computational time.

We now examine the execution times of the various adaptation blocks as described in [1] when CAD projection is used. The test case used here is that of the ONERA M6 wing. Here the initial grid is composed by 477 262 tetrahedral elements, 91 379 nodes, and 38 926 boundary faces. Only a set number of processors is used for this computation, which will be 16, since we are only interested in observing the overhead when using CAD projection. Here a single adaptation step was carried out, with respect to the Mach number, using an error estimator [18] for the refinement and derefinement and with the addition of the physical difference [1] criteria for the refinement. Smoothing iterations were set to 10 and optimisation steps to 20. Furthermore the case was run with reprojection of new nodes only in a first computation, and of all nodes in-between smoothing for another one. Table 3 shows the different grids obtained for the different cases, whilst Figure 30 shows the execution times for the three adaptation blocks. Smoothing time in Figure 30, for the case when CAD projection is used during the smoothing operation, can be broken down into two parts to obtain the real smoothing time and the projection time, as shown in Figure 31. As we can notice from this last figure, the projection time becomes more important when carried out for all surface nodes, increasing threefold the whole smoothing time block. However, it is also interesting to notice how the actual smoothing time decreases, due to reprojection, when compared to the smoothing times of Figure 30.

4. Conclusions

When creating a computational discretisation of an object, the precise geometrical definition is required in order to

FIGURE 23: Patches of the reconstructed ONERA M6 wing mesh—(a) side view, (b) detail of the tail (left) and tip (right).

FIGURE 24: Initial ONERA M6 wing mesh.

FIGURE 27: Grid on the ONERA M6 wing with (a) and without (b) projection.

FIGURE 25: First- and second-order solutions (M_∞) of the ONERA M6 wing mesh—(a) initial solution first-order, (b) restarted solution second-orders.

FIGURE 26: Solution (M_∞) and grids on the ONERA M6 wing with (left) and without (right) projection.

FIGURE 28: Solution (M_∞) and grids after a second adaptation step on the ONERA M6 wing with (left) and without (right) projection.

generate a grid. This is also true when refining the mesh in locations where the stresses will be higher, or where flow features are going to appear. However, it is rare to know the exact location of these stresses and flow features a priori, hence the mesh is adapted manually in large portions of a computational domain. The use of dynamic integrated mesh adaptation avoids the need of prior knowledge of the solution features and their locations. Given an initial coarse grid, the mesh adaptation process will automatically detect

FIGURE 29: Iso-Mach lines and solution (M_∞), with detail of the leading edge, after a second adaptation step on the ONERA M6 wing with (left) and without (right) projection.

FIGURE 30: Adaptation blocks times for CAD projection after 1 adaptation step with 16 processors. ONERA M6 wing.

the critical zones of the solution, and adapt the mesh accordingly, enhancing the accuracy and rendering an improved solution.

There remains a problem in the process described above. Just as one cannot discretize geometry without the set of curves and surfaces defining it, the adaptation algorithm cannot place the refined elements on the underlying geometry without its computational representation. This was the first goal achieved here, by inserting a dedicated library for the treatment of the most commonly used geometric representations in present day CAD software. The results presented clearly show the importance of maintaining the geometric representation, with undoubted benefits to the acquisition of a precise solution.

In order to carry out all the above processes on realistic simulations with computational meshes of more than one million nodes, mono processor-serial calculations are not

FIGURE 31: Projection and smoothing time breakdown for CAD projection during smoothing. ONERA M6 wing.

feasible. It is hence mandatory that the above techniques be developed in parallel. Numerical investigations on the parallel performances of these methods proved the efficiency of all parts of the adaptation process. Execution time of the adaptation is low compared to that of the physics solver. These were computationally efficient as shown from the results presented.

References

- [1] P. Leyland, A. Casagrande, and Y. Savoy, "Parallel mesh adaptive techniques illustrated with complex compressible flow simulations," *Modelling and Simulation in Engineering*. In press.
- [2] P. Leyland and R. Richter, "Completely parallel compressible flow simulations using adaptive unstructured meshes," *Computer Methods in Applied Mechanics and Engineering*, vol. 184, no. 2–4, pp. 467–483, 2000.
- [3] A. Loseille and R. Löhner, "On 3d anisotropic local remeshing for surface, volume and boundary layers," in *Proceedings of the 18th International Meshing Roundtable*, B. W. Clark, Ed., pp. 611–630, Springer, Berlin, Germany, 2009.
- [4] M. A. Park, "Adjoint-based, three-dimensional error prediction and grid adaptation," *AIAA Journal*, vol. 42, no. 9, pp. 1854–1862, 2004.
- [5] M. S. Shephard, J. E. Flaherty, K. E. Jansen et al., "Adaptive mesh generation for curved domains," *Applied Numerical Mathematics*, vol. 52, no. 2–3, pp. 251–271, 2005.
- [6] T. J. Baker, "Adaptive modification of time evolving meshes," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 48–49, pp. 4977–5001, 2005.
- [7] X. Li, M. S. Shephard, and M. W. Beall, "Accounting for curved domains in mesh adaptation," *International Journal for Numerical Methods in Engineering*, vol. 58, no. 2, pp. 247–276, 2003.
- [8] F. Alauzet, X. Li, E. S. Seol, and M. S. Shephard, "Parallel anisotropic 3D mesh adaptation by mesh modification," *Engineering with Computers*, vol. 21, no. 3, pp. 247–258, 2006.
- [9] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design, A practical Guide*, Computer Science and Scientific Computing, Academic Press, 4th edition, 1997.
- [10] G. Farin, *NURBS, From Projective Geometry to Practical Use*, A. K. Peters, 2nd edition, 1999.
- [11] G. Farin, J. Hoschek, and M. S. Kim, Eds., *Handbook of Computer Aided Geometric Design*, Elsevier Science, 1st edition, 2002.

- [12] R. Haimes and G. J. Follen, “Capri (computational analysis programming interface): a solid modeling based infrastructure for engineering analysis and design simulations,” Tech. Rep. 19990004160, NASA Glenn Research Center, 1998.
- [13] I. J. Anderson, M. G. Cox, A. B. Forbes, J. C. Mason, and D. A. Turner, “An efficient and robust algorithm for solving the foot point problem,” in *Proceedings of the international conference on Mathematical methods for curves and surfaces II Lillehammer 1997*, pp. 9–16, Vanderbilt University, Nashville, Tenn, USA, 1998.
- [14] M. Aigner and B. Jüttler, “Robust computation of foot points on implicitly defined curves,” in *Mathematical Methods For Curves and Surfaces: Tromsø 2004*, pp. 1–10, Brentwood, Tenn, USA, 2005.
- [15] E. Hartmann, “On the curvature of curves and surfaces defined by normalforms,” *Computer Aided Geometric Design*, vol. 16, no. 5, pp. 355–376, 1999.
- [16] T. Dokken, *Aspects of intersection algorithms and approximation [Ph.D. thesis]*, University of Oslo, 1997.
- [17] M. Sala, P. Leyland, and A. Casagrande, “A parallel adaptive Newton- Krylov-Schwarz method for the 3D compressible flow simulations,” *Modelling and Simulation in Engineering*. In press.
- [18] A. Casagrande, P. Leyland, and L. Formaggia, “Parallel mesh adaptive techniques for complex flow simulations,” *Modelling and Simulation in Engineering*. In press.