BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 65, No. 2, 2017
DOI: 10.1515/bpasts-2017-0024

Parallel anisotropic mesh refinement with dynamic load balancing for transonic flow simulations

S. GEPNER*, J. MAJEWSKI, and J. ROKICKI

Warsaw University of Technology, Institute of Aeronautics and Applied Mechanics, 24 Nowowiejska St., 00-665 Warsaw, Poland

Abstract. The present paper discusses an effective adaptive methods suited for use in parallel environment. An in-house, parallel flow solver based on the residual distribution method is used for the solution of flow problems. Simulation is parallelized based on the domain decomposition approach. Adaptive changes to the mesh are achieved by two distinctive techniques. Mesh refinement is performed by dividing element edges and a subsequent application of pre defined splitting templates. Mesh regularization and derefinement is achieved through topology conserving node movement (**r**-adaptivity). Parallel implementations of an adaptive use the dynamic load balancing technique.

Key words: parallel CFD, adaptation, dynamic load balancing, mesh refinement, parallel efficiency.

1. Introduction

Adaptive techniques allow, in principle, for accurate computation of localized phenomena (boundary layers, shock waves, etc.). This is achieved by enriching the mesh where necessary and by limiting the resolution in the less interesting regions. This approach increases the effectiveness of computations, as the number of mesh cells and therefore the computational cost are kept limited. Further benefits can be achieved if the adaptive procedure takes into account the anisotropy of the flow phenomena. [1–4]. The adaptation may thus be regarded as a generic tool either to reduce the computational cost or to increase the quality of solution at a given numerical cost.

On the other hand the large scale simulations require highly efficient parallelization in order to converge in a reasonable time. Commonly used domain decomposition approach requires that the flow field is partitioned into subdomains, prior to the simulations. To maximize the performance of parallelization the partitioning should minimize both the processor idle time as well as the volume of interprocessor communication. This is achieved by equidistribution of the numerical load (represented by the distribution of mesh entities), and at the same time, by minimization of the number of cut edges. To this end, a range of partitioning methods, based on graph partitioning algorithms, have been developed [5–8]. Since adaptivity delivers a dynamic change to the distribution of mesh entities among processors a dynamic load balancing strategy must be used to maintain high performance of parallelization [9–16].

An adaptive algorithm requires some driving mechanism to enforce mesh modification. Often this is achieved through the use of a metric field defining an optimal mesh spacing. Commonly, and quite successfully this metric field is generated by the Hessian of the current solution [1, 3, 4, 17]. An alternative is the application of the Zienkiewicz-Zhu mechanism based on the reconstructed gradient [1, 18–22]. Still another possibility is a goal oriented method striving to improve only the accuracy of a selected integral value of interest (e.g., drag of the wing, see [23–27]).

The adaptive changes can be achieved by different means. Remeshing methods employ global reconstruction of the mesh in the entire computational domain, making use of the standard meshing techniques [28–32]. In general, this approach yields high quality meshes for arbitrarily complex geometries [4, 33–36]. Nevertheless this approach is difficult to implement in parallel simulations because of the global nature of the remeshing operation.

An alternative approach is to apply modifications to the computational mesh only in the regions of interest. Either by local mesh modification operators (edge splits, edge collapses and edge swaps) [37–40] or element bisection (e.g., based on template refinement) [10, 41–47]. Locality of these methods makes them especially attractive for application in parallel computing, as the necessary communication volume and frequency become limited, boosting parallel performance.

Another possibility is brought by the **r**-adaptive methods, in which grid nodes are allowed to move without changing the topology of the mesh. This technique is used either to augment local refinement methods [48, 49], or as an adaptive tool in itself [50–52]. Application of **r**-adaptive algorithms might be beneficial in parallel computations, as neither the grid topology nor the load balance become affected.

Although adaptive methods are well recognised in the academic community, the broader application of adaptation for industrial problems has been limited. This paper presents an approach towards fully parallel and automatic mesh adaptation method that can be used for 3D industrial cases. The **h-r** adaptive method is developed, suitable for parallel simulation of transonic flows. Adaptivity is accomplished by anisotropic template-based element bisection method. The coarsening step is carried out via an elastic analogy node movement. Anisotropic mesh modifications are driven by a Hessian based approach.

*e-mail: sgepner@meil.pw.edu.pl

Due to the template based refinement, there is no need to calculate the metric field directly, instead an adaptive indicator [12, 53] is used to trigger edge splits. The dynamic load balancing strategy (DLB) [11, 16] is applied to maximize the parallel effectiveness of calculations.

This paper is organized as follows. Section 2 briefly specifies an in-house residual distribution based flow solver used for solution of the flow problem. Section 3 recalls notion of the error estimation process and specifies an adaptive indicator used to drive the adaptive process. Section 4 outlines the template splitting approach and provides descriptions of the possible refinement templates used in this work. Section 5 discuses an **r**-adaptive method used. Section 6 illustrates application of the application of the DLB on to the overall parallel performance. Section7 presents computational results calculated with the the adaptive methods described in this paper. Conclusions are presented in Section 8.

2. The flow solver

In the present paper adaptation is applied to stationary, advection dominated problems discretized with a node centred residual distribution scheme [54–58]. In all cases unstructured simplex meshes are used.

Parallelization of the solver is achieved through mesh partitioning with a single element overlap between neighbouring subdomains. The inter-process communication is achieved through the exchange of the corresponding nodal data. Each process stores communication data structure containing information on the local nodes shared with a neighbouring processes. A similar data structure describes nodes that the process receives from its neighbours.

For the node centred approach, the ownership of mesh nodes is well defined and results directly from the partitioning (the mesh entities are owned by the process with the lowest parallel rank index). The notion of ownership is important for calculation of the adaptive indicator as well as in the refinement step.

The present implementation of the residual distribution scheme requires that the computational meshes have no hanging nodes. Similarly the elements connected through the face in 3D or the edge in 2D must connect through a single (edge or face) entity. These requirements impose additional constrains on the refinement step, in particular in the parallel overlap regions.

3. An adaptive indicator

In general, the adaptive procedure establishes the desired element spacing basing on the information recovered from the numerical solution obtained on the current mesh. Often, and quite successfully the Hessian of the solution is used to reconstruct the interpolation error on a given mesh (see e.g., [3, 59]). Indeed for the second order discretization the leading term of the error contains the norm of the Hessian.For the grid cell E this error can be estimated as:

$$\varepsilon_E \sim \max_{\mathbf{x}\in E}(\mathbf{x}-\mathbf{x}_c)^T|\mathscr{H}|(\mathbf{x}-\mathbf{x}_c) \tag{1}$$

where $\mathscr{H}$ is the Hessian and $\mathbf{x}_c$ stands for the center of the cell.

The refinement procedure presented in this work (see Section 4) operates by first splitting the edges of mesh elements, and then by applying appropriate split templates to divide elements. These are local operations and their application is beneficial in a parallel simulation. At the same time, error indicator (based on (1)) is also locally calculated for each edge.

The key problem in the evaluation of such adaptive indicator (or a metric tensor) consists in the reconstruction of the Hessian [60]. The present approach is based on a very robust and efficient Green formula.

The function u, on which adaptation is based, is known only at the grid nodes. Using the Green formula, the gradient can be calculated as:

$$\nabla u = \frac{1}{\Omega}\oint_{\partial\Omega} u\,\mathbf{n}\,dS \tag{2}$$

where Ω denotes the control volume consisting of cells neighbouring to a node at which Hessian is being estimated. This procedure is repeated twice, first to estimate $\nabla(\nabla u)$, and subsequently to estimate $\mathscr{H} = \nabla(\nabla u)$.

The Hessian matrix is symmetric and therefore it is diagonalisable and has real eigenvalues λ_1, λ_2 and λ_3. The matrix $|\mathscr{H}|$ is defined as:

$$|\mathscr{H}| = R\cdot\begin{bmatrix} |\lambda_1| & 0 & 0 \\ 0 & |\lambda_2| & 0 \\ 0 & 0 & |\lambda_3| \end{bmatrix}\cdot R^{-1} \tag{3}$$

where matrix R consists of eigenvectors of $\mathscr{H}$ as its columns. The error indicator (for an edge e_{ij} connecting nodes i and j) is defined now as:

$$I_{\mathscr{H}e_{ij}} = C_h\mathbf{e}_{ij}^T|\mathscr{H}_{e_{ij}}|\mathbf{e}_{ij} \tag{4}$$

where the arbitrary, positive parameter C_h allows to control the threshold of adaptivity. In the current approach C_h is set to one and edges selected for refinement are these, for which the indicator $I_{\mathscr{H}e_{ij}}$ exceeds a predefined threshold value, calibrated so that the 25% of edges are split in the refinement step.

In case of a parallel simulation calculation of the adaptive indicator (4) is only modified in the parallel overlap region. Each process decides only on splitting edges it owns. The relevant information is then communicated to the neighboring processes, which share a given edge.

4. Template based mesh refinement

In the current approach the mesh is refined through the use of predefined splitting templates. As only simplex elements are considered, the templates are limited to triangles and tetrahedra. Both the boundary and the internal mesh elements are considered for splitting. The refinement process consists of the following steps:

1. Select the mesh edges for splitting, basing on the value of the error indicator (4).
2. Enrich the mesh with the new nodes inserted at the edges centres.
3. Use the appropriate template to split each internal or boundary mesh element, which has at least one edge divided.
4. Remove all split mesh elements from the mesh data structure.
5. Add elements resulting from the splitting to the mesh data structure.
6. In case of a parallel simulations, negotiate the inter-processor communication.

The possible splitting templates are grouped in categories, based on the number of edges being divided (1–3 for triangles and 1–6 for tetrahedra). Within each category different, variants are identified by the indices of the split edges. The index of an edge within an element is based on the nodal indices within an element. The numbering convention is shown in Fig. 1.

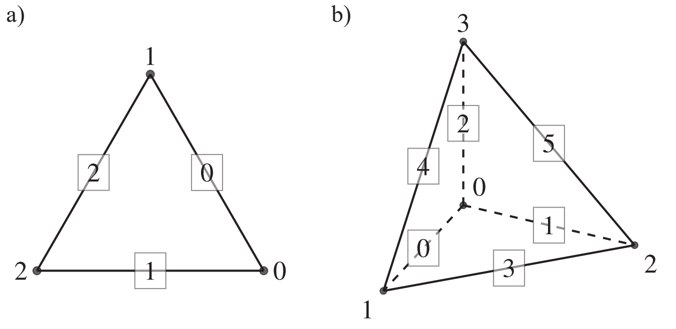

Fig. 1. The numbering of edges, allowing to distinguish between different splitting templates

Figure 2 illustrates two possible divisions of a triangular element $[N_i, N_j, N_k]$, with edges E_0 and E_1 selected for splitting. The splitting templates for triangular elements are shown in Fig. 3.

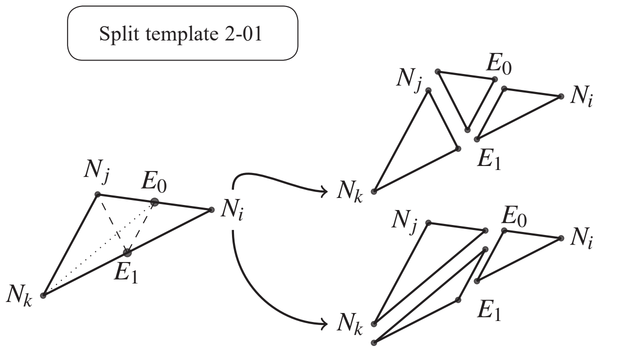

Fig. 2. Two variants of element splitting according to template 2-01 (two edges, 0 and 1 are selected for splitting)

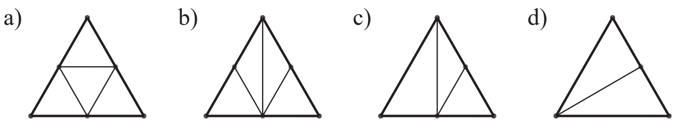

Fig. 3. Triangle splitting templates

4.1. Tetrahedron refinement templates. Similar logic is applied to the splitting templates for tetrahedra. Six basic division categories are used. Within each category different variants are possible, depending on the relative location of the split edges.

In contrast to the 2D meshes, the refinement of the 3D meshes may lead to unsplittable (sometimes referred to as Schoenhardt) polyhedra [61, 62], necessitating the use of additional Steiner nodes. This happens for some templates from the third category (three edges being split, variants: 012, 135, 245, 345) and for almost all category four and five divisions.

Another difficulty lies in the requirement that the neighbouring elements should share the same face. To illustrate this problem, consider splitting of tetrahedral elements connected by a triangular face shown in Fig. 2. Should the tetrahedral splitting produce incompatible faces (by applying both variants of template 2–01), the resulting mesh would becomes unusable.

To guarantee compatibility of the neighbouring elements the variant with the shorter edge is always selected for splitting. In case both edges are equal, the decision is based on the nodal indices of the edges.

In case of a parallel application the modification of this procedure is required in the overlap region. First, the process owning an element must be identified. At the present implementation it is the one with the lowest parallel rank index. The overlapping elements are split by their owning processes only. The results of splitting are then communicated to the neighbouring processes, should any changes be required. The compatibility of elements in the overlap region is ensured by the use of global nodal numbering.

Category one templates result in refinement into two new tetrahedra. Figure 4 illustrates application of 1–1 template.

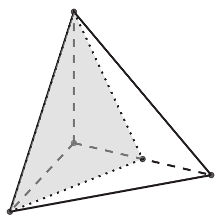

Fig. 4. Category-one element splitting

The second category contains fifteen division variants. Templates corresponding to the splitting of opposite edges (2–05, 2–14, 2–23; Fig. 5c and 5d) result in four new tetrahedra. The remaining variants produce three (Fig. 5a and 5b) new elements.

Twenty variants exist in the third category. These variants can be grouped into three subcategories. Firstly, when the edges

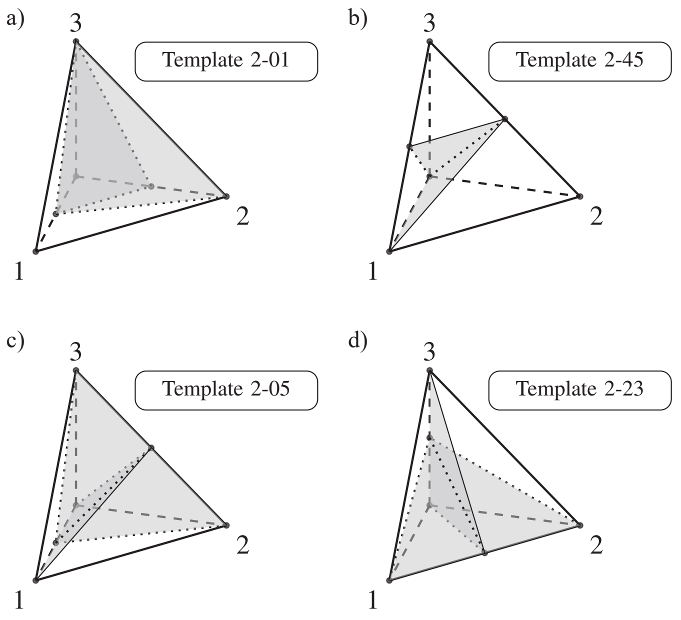

Fig. 5. Selected splitting templates of the second category. a)–b) formation of three new elements. c)–d) formation of four elements

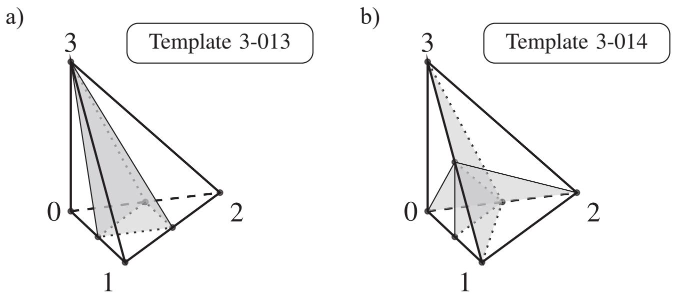

Fig. 7. Variants of category 3 splitting templates

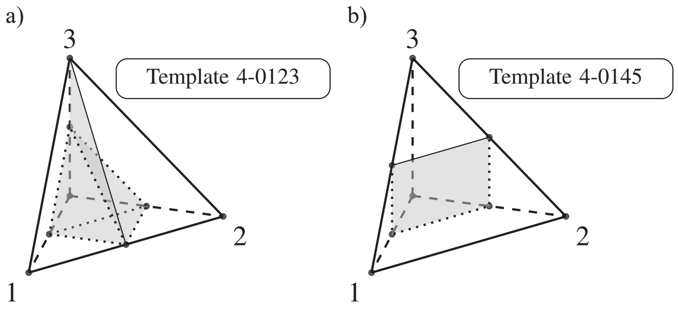

Fig. 8. Two possible variants of the fourth category

after splitting connect through a common node (variants 3–012, 3–034, 3–135, 3–245) – see Fig. 6. In this case a tetrahedron and a possibly unsplittable polyhedron [61, 62], (shown in Fig. 6c) can apprear. This happens when splitting regular tetrahedra, and requires an additional, internal node, which has to be used to enable tetrahedralization (Fig. 6c and 6d). The total of nine tetrahedra are then produced. If this is not the case, three new tetrahedra are produced as shown in Fig. 6b.

Secondly when the split edges belong to a common face (variants 3–013, 3–024, 3–125, 3–345) four new tetrahedra are formed. This is illustrated in Fig. 7a.

Thirdly, the remaining splitting variants of this category result in five new tetrahedra. An example is presented in Fig. 7b.

Fifteen possible variants belong to the fourth category. These fall into two possible cases. The first takes place when three of four split edges connect through a common node (Fig. 8a). This results in six new tetrahedra. The second case occurs when each of the faces has exactly two split edges (Fig. 8b), resulting in the necessity to split the tetrahedron by a plane. The new polyhedra presented in Fig. 6c-d may require additional Steiner points to enable further splitting.

The split variant of the category five is shown in Fig. 9. Similar to categories three and four, the possible Schoenhardt

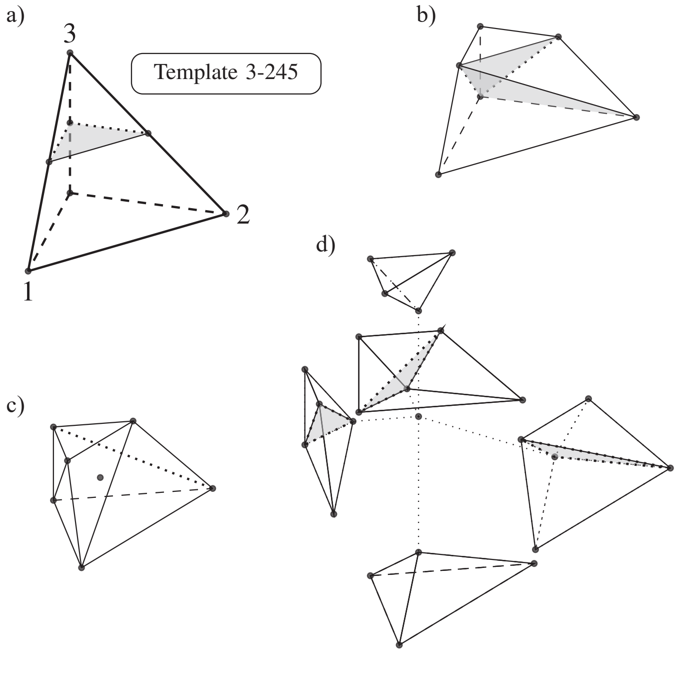

Fig. 6. Splitting template 3–245. a) The initial split results in a tetrahedron and a polyhedron requiring further processing. b) Splitting of the resulting polyhedron, if possible. c) In case of the Schoenhardt polyhedron an additional Steiner node is inserted. d) Eight additional tetrahedra, resulting from splitting

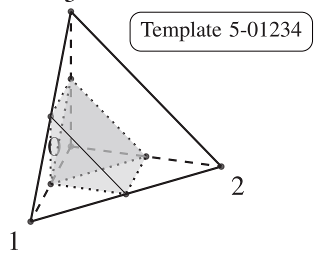

Fig. 9. Category five split template 5–01234

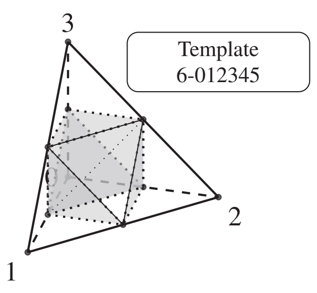

Fig. 10. Single split template of the sixth category

polyhedron is dealt with by insertion of an additional Steiner node, and split analogous to the one shown in Fig. 6d.

Finally, a single variant of category six is presented in Fig. 10. The splitting results in formation of eight new tetrahedra.

An example of application of the splitting templates for the generic 3D mesh is illustrated in Fig. 11. The consecutive refinements, of the initially coarse mesh, are performed by dividing edges located in the predefined region. In this case an adaptive indicator is evaluated for an artificial discontinuity having the shape of a unit sphere.

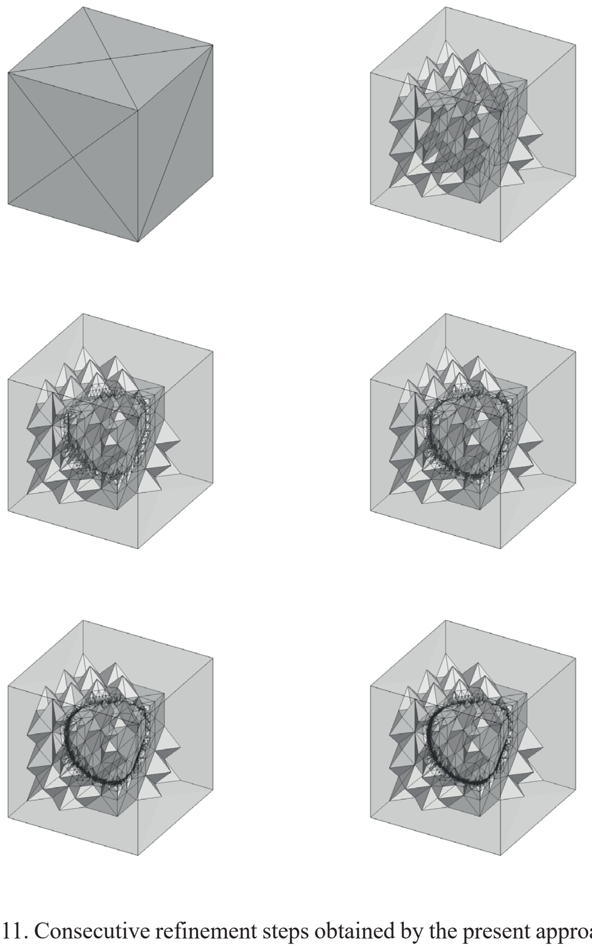

Fig. 11. Consecutive refinement steps obtained by the present approach. The adaptation indicator is driven by an artificial discontinuity in the form of a unit sphere

5. R adaptivity

The adaptive strategy requires also a mechanism to coarsen the mesh and at the same time to eliminate short edges generated during the refinement step [42], as well as to break the influence of the initial mesh onto the adapted mesh.

Instead of a derefinement procedure, the present approach uses a variant of **r** adaptivity, in which the mesh is regarded as elastic and deformable. The deformations are controlled by equations of linear elasticity, as well as by the appropriate boundary conditions:

$$\begin{cases}(\lambda+\mu)\nabla(\nabla\cdot\mathbf{u})+\mu\nabla^2\mathbf{u}+\mathbf{F}=0\\ \mathbf{u}|_{\partial\Omega}=0\end{cases}\tag{5}$$

where **u** denotes the displacement field, λ and μ are local Lamé parameters, while **F** stands for a prescribed vector field. The position of all boundary nodes is fixed through a Dirichlet boundary condition.

The solution of eq. (5) is obtained via the standard finite element method [63], which leads to a symmetric linear algebraic problem:

$$K\mathbf{u}=\mathbf{b}\tag{6}$$

The application of this form of adaptivity is especially attractive in a parallel framework, since the necessary numerical tools (mesh and solution data structure, partitioning, assembly and communication, algebraic solvers, etc.) are already used by the flow solver.

While the assembly of the stiffness matrix K (6) follows a well known procedure, the derivation of the right hand side requires explanation.

In the present work the value of an edge error indicator (4) is used to calculate nodal forces applied to the mesh nodes. For the i-th node of the mesh, with N other nodes as neighbors, the nodal force is calculated as the sum of contributions of corresponding edges:

$$\mathbf{F}_i=C_r\sum_{k=1}^{N}I_{\mathscr{H}e_k}\widehat{\mathbf{e}}_k\tag{7}$$

where $I_{\mathscr{H}e_k}$ denotes the error indicator calculated by (4), C_r is a scaling constant while $\widehat{\mathbf{e}}_k$ stands for the versor parallel to the edge, and pointing outwards from the central node i. By applying (7) to all mesh nodes, the right hand side of (6) is evaluated. The particular form of (7) is justified by the observation that the adaptive method should modify the mesh towards the equidistribution of the estimated error.

6. The load balancing

To obtain a high parallel efficiency the proper balancing of the numerical load is required. Even if the load is balanced at the beginning of the simulation, the dynamic changes caused by the adaptive refinement will significantly lower the efficiency [10, 16]. This makes direct application of adaptivity unfeasible for the automated parallel simulations. It is thus necessary to apply the dynamic load balancing (DLB) strategy throughout the entire parallel simulation.

In order to quantify the quality of the load balancing suppose that the total computation effort is $W = \sum_i^p w_i$ (p denotes the number of processors used and w_i is the workload of the i-th processor). In the homogeneous, parallel computational environment, an optimal load distribution is such that all computational processors are equally loaded. Therefore optimal processor load is $w_{opt} = W/p$. The quality of partitioning can be then expressed [64] by the coefficient β

$$\beta = \max_{0<i\leq p} \frac{w_i}{w_{opt}} \tag{8}$$

Values, of β close to one, indicate a well balanced partitioning. Bigger values of β indicate that the load is un-balanced. It should be noted, that (8) disregards the influence of communication overhead, resulting from a given partitioning.

Various approaches are available to obtain a balanced partition of the computational domain/grid between processors. In this work we make use of the approach described earlier in [12] and [16]. The impact of this particular DLB implementation on the load balance indicator β, and in consequence on the parallel performance is summarized below.

Figures 12 and 13 illustrate parallel speed-up and scaling plots obtained for a parallel simulation of supersonic flow through a Scramjet like geometry. Additionally values of β registered for each case are added to the scaling plot (Fig. 13). One might notice that increase in the value of β results in the deterioration of performance.

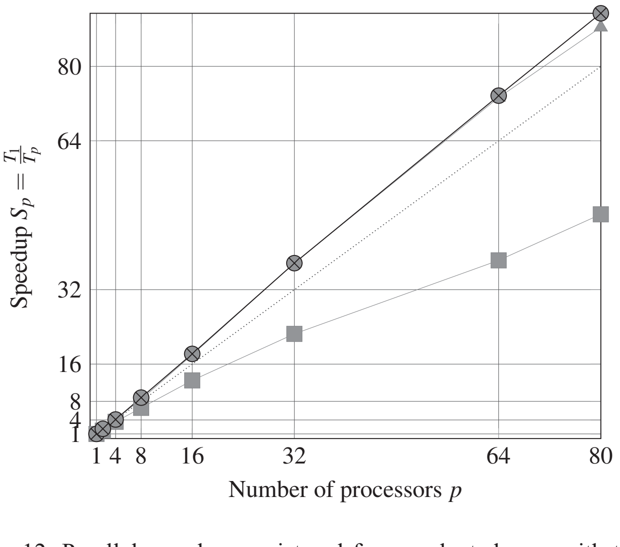

Fig. 12. Parallel speed-up registered for an adapted case with the Dynamic Load Balancing (⊗), compared with the adapted but unbalanced case (■). Results measured using unadapted meshes are provided for reference (▲)

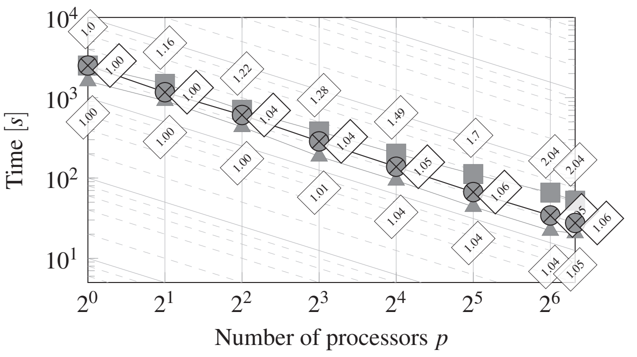

Fig. 13. Parallel scaling registered for an adapted case with load balancing (⊗), compared with the results for the adapted, but unbalanced case (■). For the reference the unmodified case is shown (▲). The load balance indicator β defined with (8) is provided for each of the cases (see the numerical boxes). The plot's background grid corresponds to the linear speed-up case

The lines marked with (▲) correspond to the results obtained for the original, unadapted mesh, resulting in a problem containing $8.0 \cdot 10^5$ DOFs [16]. For this particular configuration superlinear speed-up has been achieved (smaller tasks better fit into the processor cache memory). Influence of mesh refinement, with no load balancing, results in a significant performance drop. This is illustrated by (■). The reader should also notice that, as the parallel performance drops, the values of β increase. The results registered for the adaptation coupled with load balancing are represented by (⊗).

The load re-balancing process is triggered when the parallel efficiency is expected to drop below the accepted level (this usually takes place after the refinement step). As a result the improved domain partitioning is established. The full algorithm consists then of the following steps:

1. The elements of the mesh receive a global numbering throughout the whole domain. This makes mesh elements easily distinguishable within the parallel environment.
2. The improved partitioning is found using procedures of the ParMETIS [65] graph library.
3. According to the new partitioning, the mesh entities (nodes, elements, boundary faces) are marked to be sent/deleted from the process data structure.
4. The mesh and the solution data is migrated between processors.
5. Finally, the parallel overlap layer is reconstructed and the simulation process is restarted.

7. Computational examples

To investigate the applicability of the presented approach computational examples of increasing complexity are introduced. These include (i) the 2D Burgers equation for which the performance of error indicator is examined, (ii) the 3D wedge flow for which 3D aspects of the method are verified and (iii) the DLR-F6 case which represents the industrial multiscale 3D geometry.

7.1. Burgers equation. First we consider the scalar non linear advection problem, the Burgers equation, solved in the (x, y) space-time domain.

$$\begin{cases} \frac{\partial u}{\partial y} + \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) = 0 \\ u(x, y=0) = \begin{cases} 1.5 \text{ for } x \le 0 \\ 1.5 - 2x \text{ for } x \in (0,1\rangle \\ -0.5 \text{ for } x > 1 \end{cases} \end{cases} \quad \Omega = [0,1] \times [0,1] \tag{9}$$

The problem (9) is often used to test the performance of advection solvers. As the solution of (9) contains strong discontinuity (for $y > y^*$ – see Fig. 14) it mimics the behaviour of the more complex non linear hyperbolic systems. The existence of the analytical solution to (9) allows for quantitative verification of convergence of the numerical scheme. In this work we use the L_2 norm of the solution error:

$$\varepsilon_{L_2} = \frac{1}{\|\Omega\|}\sqrt{\int_\Omega |u - u_n|^2 d\Omega} \tag{10}$$

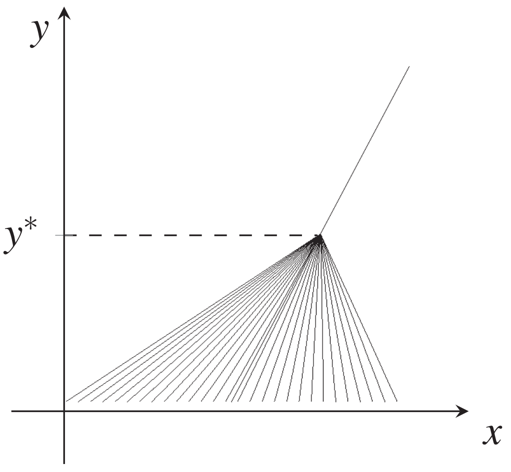

Fig. 14. Characteristics and the strong discontinuity of the analytical solution of the Burgers equation (9)

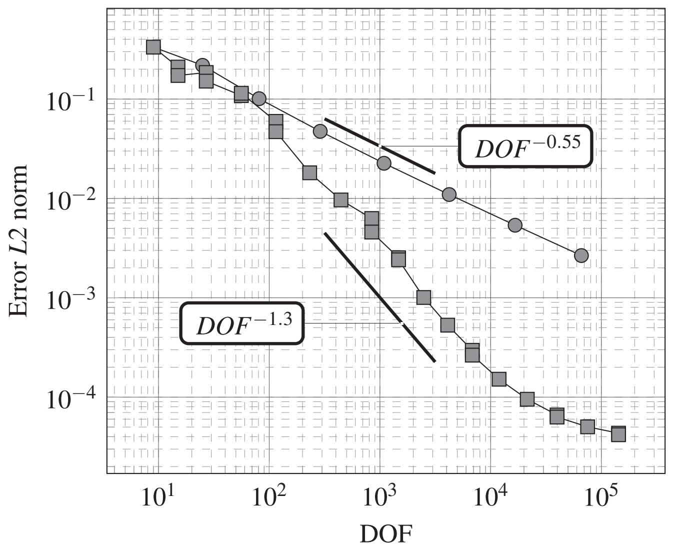

Fig. 16. L_2 error of the numerical solution of (9) as a function of the number of degrees of freedom.
–●– – the uniform refinement
–■– – the adaptive refinement and the mesh movement

where $u(\mathbf{x})$ is given by (9), and $u_n(\mathbf{x})$ is the numerical approximation.

The solutions on a sequence of uniformly and adaptively refined meshes, have been computed using the Residual Distribution method PSI scheme [54, 55, 57]. Each simulation was performed using four concurrent computational processes. For adaptive cases no user interaction was necessary. The automatic adaptive simulation cycle incorporated a DLB algorithm.

The meshes generated during the parallel adaptive simulation are shown in Fig. 15. The L_2 norm of error as a function of the number of degrees of freedom is plotted in Fig. 16 for the case of uniform refinement and for the adaptive algorithm. One may observe that application of adaptive methods leads to a much faster convergence. For the L_2 error uniform refinement converges proportionally to $DOF^{-0.55}$, while application of adaptivity leads to convergence proportional to around $DOF^{-1.3}$. Occasionally the error is decreased without changing the number of DOFs. This results solely from the mesh movement step (**r**-adaptation).

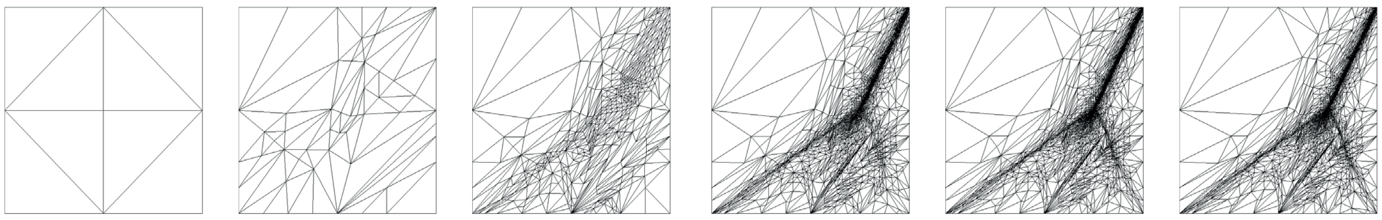

Fig. 15. Computational meshes generated throughout the parallel adaptive simulation. The top row presents, from the left the initial mesh and the mesh obtained after third and sixth adaptation. The bottom row meshes are obtained at the eighth, twelfth and fifteenth iteration

7.2. 3D channel-wedge. Much more challenging test consists in simulation of a supersonic flow through the 3D channel-wedge geometry. Flow conditions correspond to Mach 2 at the inlet. The resulting solution contains discontinuities in a form of a sequence of reflected oblique shock waves. Additionally in the proximity of the upper wall the triple point (line in 3D) is formed with a contact discontinuity appearing behind it.

The Euler equations are discretized in a 3D domain using residual distribution LDAN scheme [54, 55, 57]. Parallel simulation employed 412 concurrent computational processes. The mesh was adapted through six automatic adaptive cycles, using mesh refinement and mesh movement techniques outlined in this paper. An adaptive indicator (4) was used to trigger adaptive changes. The necessary DLB was applied, to keep the parallel efficiency at the desired level. Cost of the error estimation, adaptive changes and load balancing is negligible compared to the cost of converging the simulation on a given mesh (less than 3% of the total simulation time).

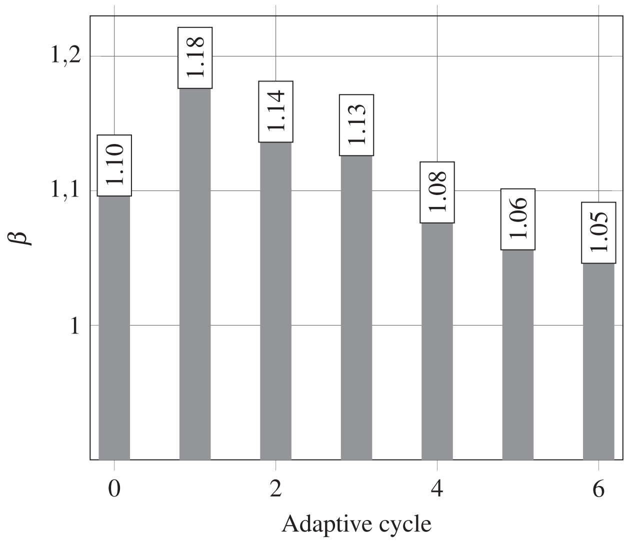

Fig. 17. Load balance indicator β (8), registered after consecutive adaptations/load balancing steps, for the 3D channel-wedge test case (number of processors $p = 412$)

Figure 17 illustrates values of load balance indicator β measured at the consecutive adaptive steps. As a result, the initial mesh of $1.0 \cdot 10^5$ tetrahedral elements and $2.4 \cdot 10^4$ nodes, was refined to a mesh of around $5.9 \cdot 10^6$ elements and $1.2 \cdot 10^6$ nodes. The entire simulation was performed automatically without an interaction with the user.

Figure 18 shows the initial and the final meshes (after the sixth adaptation), as well as the boundaries of mesh partitions for the adapted case. One can observe that partitions, initially evenly distributed, concentrate in the proximity of shocks, where the mesh is refined. Comparison of the initial and the adapted results against evenly refined mesh of $7.9 \cdot 10^6$ elements and $1.3 \cdot 10^6$ nodes, in the form of Mach field contours is shown in Fig. 19. Figure 19a-c shows results obtained using the original mesh with the contact discontinuity zone magnified in Fig. 19c. The improvement in shock resolution is well visible in Fig. 19d-f where solutions obtained on the mesh generated through adaptation process are shown, with Fig. 19f illustrating the contact discontinuity region. As a reference Figs. 19g-i illustrate results obtained using an isotropic, but larger mesh. Comparison of Figs. 19c, 19f and 19i indicates that, although corresponding to a larger number of DOFs, the isotropic large mesh produces less resolved flow field than the anisotropic one obtained through the adaption procedure.

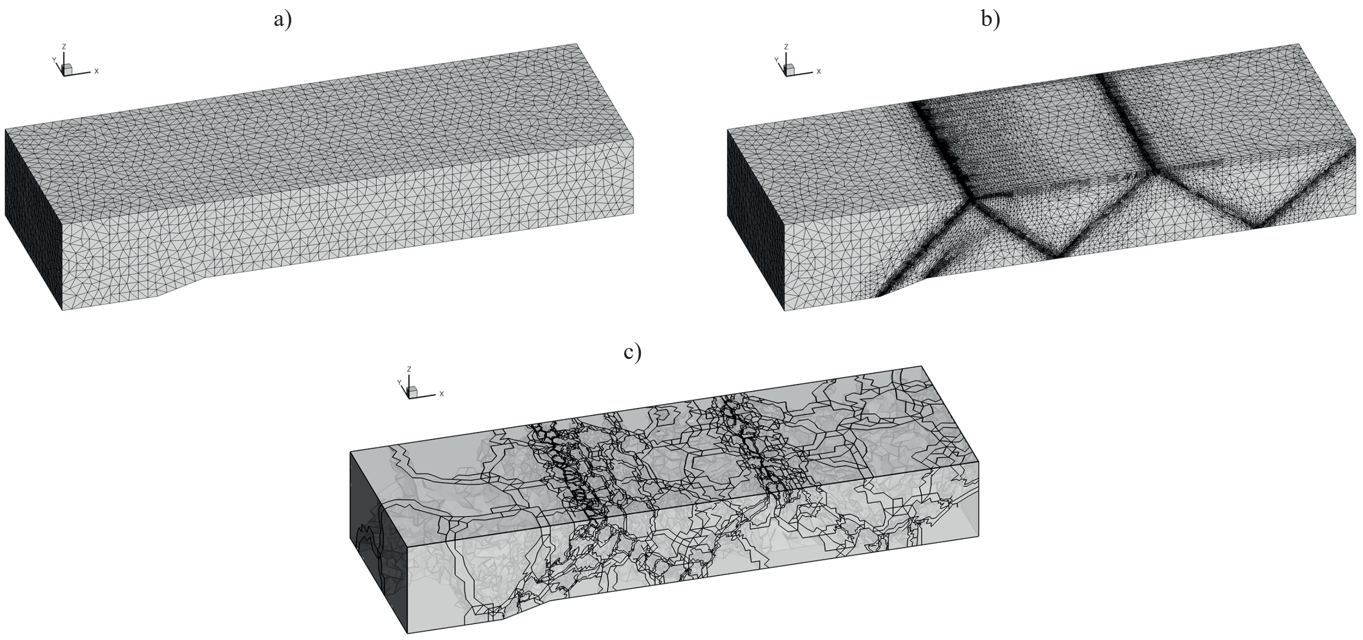

Fig. 18. Initial mesh (a) and the one generated through adaptive simulation (b). Domain partitioning corresponding to the load balanced distribution for an adapted mesh (c)

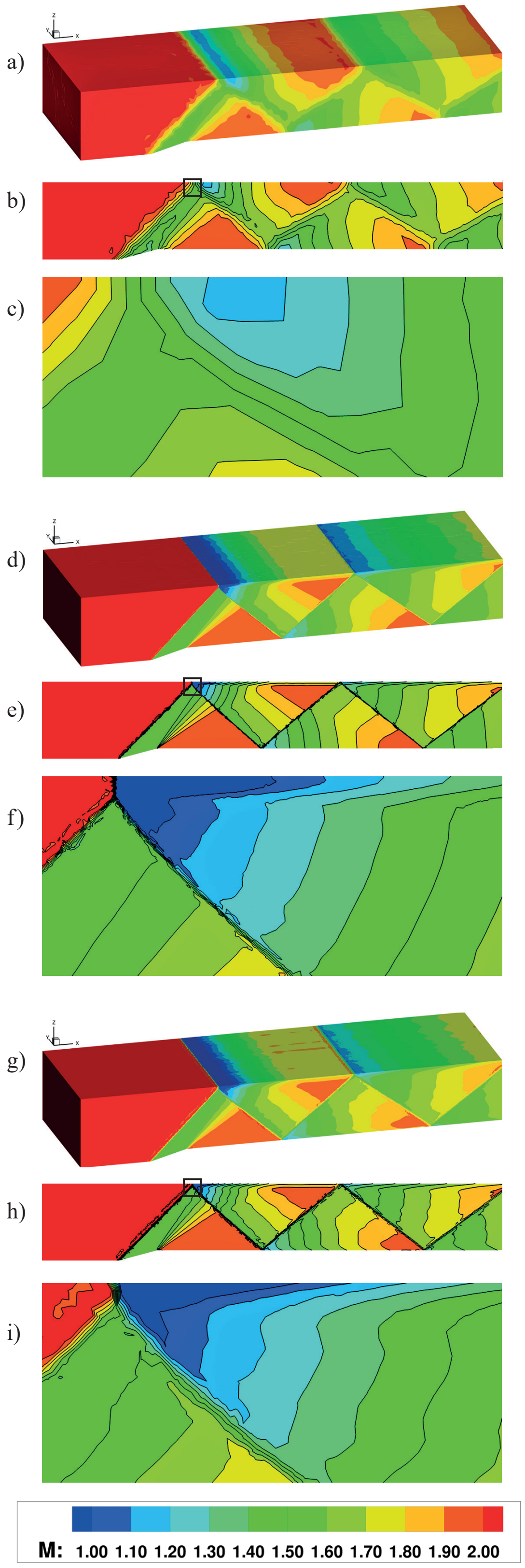

Fig. 19. Mach number contours calculated on the initial mesh (a-c) and using the mesh generated through six adaptive cycles (d-f) compared with results obtained using a large isotropic one (g-i). The planar view (b, e and h) corresponds to the mid-channel crosssection (obtained via interpolation from the 3D terahedral mesh). The marked region (c, f and i) shows the contact discontinuity region

7.3. Onera M6 wing. Finally we show the results of parallel and adaptive simulations of a transonic flow around Onera M6 wing [66]. Flow conditions correspond to Mach number 0.8395 and angle of attack 3.06°. As before Euler equations are discretized using Residual Distribution LDAN scheme. Computational mesh was adapted through four automatic adaptive cycles, using mesh refinement and mesh movement techniques outlined in this paper. Adaptive changes resulted from application of an adaptive indicator (4). The necessary DLB was applied, to keep the parallel efficiency at the desired level. Cost of the error estimation, adaptive changes and load balancing is negligible compared to the cost of converging the simulation on a given mesh (as earlier it is less than 3% of the total simulation time). As a result the initial mesh of $3.2 \cdot 10^4$ nodes, was refined to a mesh of around $4.6 \cdot 10^5$ nodes. The entire simulation was performed automatically without an interaction with the user.

Figure 20 shows Mach field contours obtained using the initial (Fig. 20a) and adapted (resulting from the fourth adaptation)

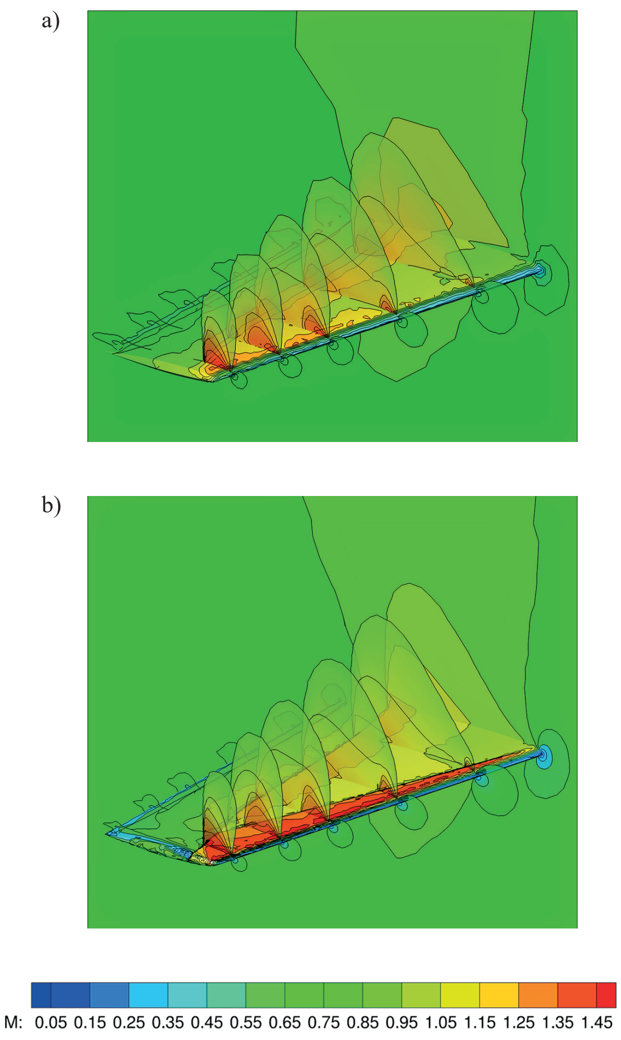

Fig. 20. Mach number contours on the initial (a) and adaptively generated mesh (b)

meshes. Adaptively generated meshes and corresponding, load balanced partitionings are shown in Fig. 21.

Figure 22 illustrates distribution of the C_p coefficients calculated at selected positions throughout the wing. Results obtained on the initial mesh are marked with red colour, while results obtained on adaptively generated mesh are shown in black. As reference we use experimental results marked with (■), and results obtained using a different adaptive strategy (global re-meshing) calculated by Majewski [34] (green line), using a full Navier–Stokes solver with turbulence modelling.

8. Conclusions

In this paper we have presented an approach to automatic mesh adaptivity, applicable to parallel simulation. Two techniques, suitable for application in parallel simulation, used in this work for mesh modification, are (i) the elastic mesh movement and (ii) the element bisection method.

We have proposed an alternative to the commonly used, metric based, mesh spacing definition, in the form the of edge adaptive indicator. This form is more suitable to allow for the required mesh modification. Proposed method allows for a natural linear interpolation of solution to an adapted mesh.

Applicability of the methods described in this paper was illustrated on a sequence of advection dominated problems. It was shown that, for Burgers equation, significant improvement in the rate of convergence is achievable using the adaptive approach. For the 3D flow problems the resolution of important flow features (shocks, slip lines) was significantly improved. Application of DLB strategy allowed to maintain low values of the β indicator, and in consequence high parallel performance.

Promising initial results obtained for complex curvilinear 3D geometries have been presented. The robust boundary reconstruction is still required to augment the refinement process.

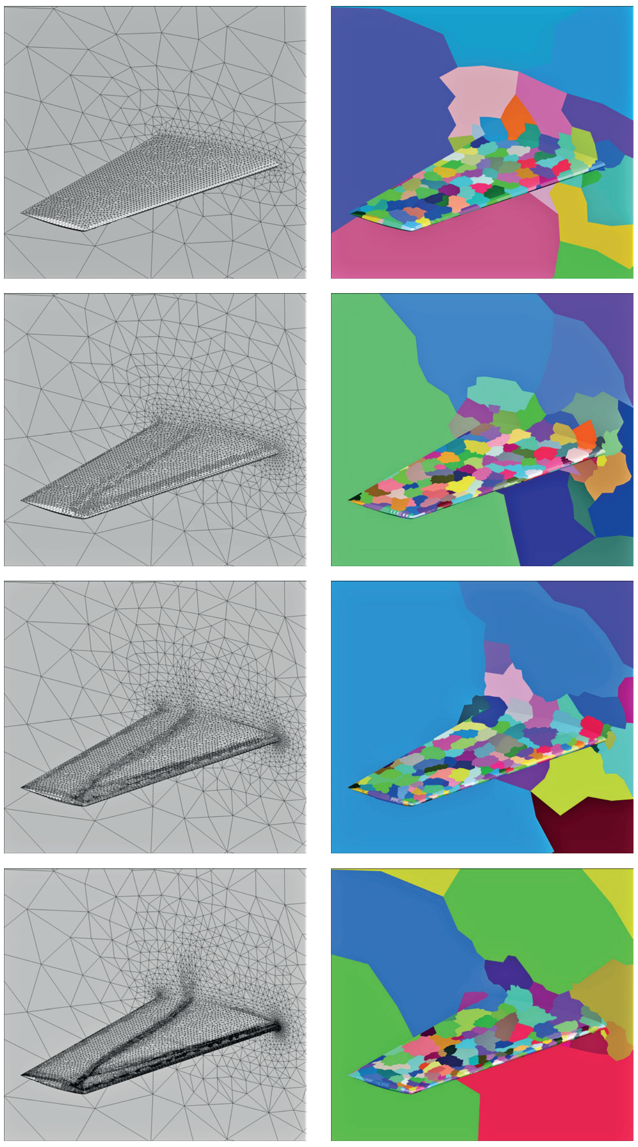
Fig. 21. Meshes at selected adaptive steps with corresponding, load balanced domain decompositions

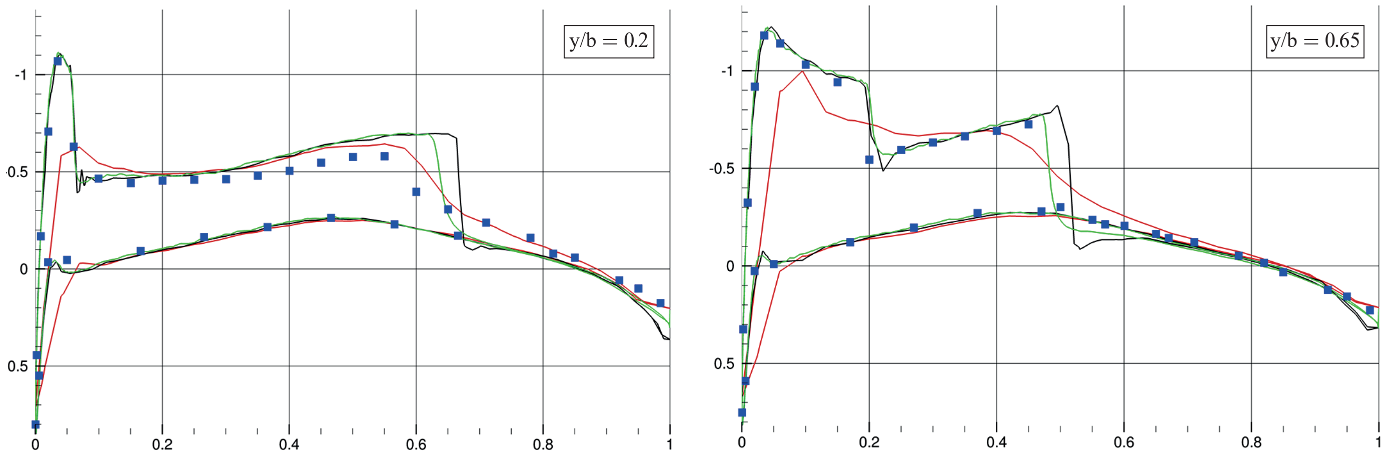

Fig. 22. Comparison of the C_p distribution obtained for Onera M6 wing geometry. ■ – experimental data, **red** – initial mesh results, **black** – final adapted mesh, **green** – reference computations [34], obtained for a viscous case at $M = 0.8395$, $Re = 11.72 \cdot 10^6$, $\alpha = 3.06°$ – Navier–Stokes solver with turbulence modelling

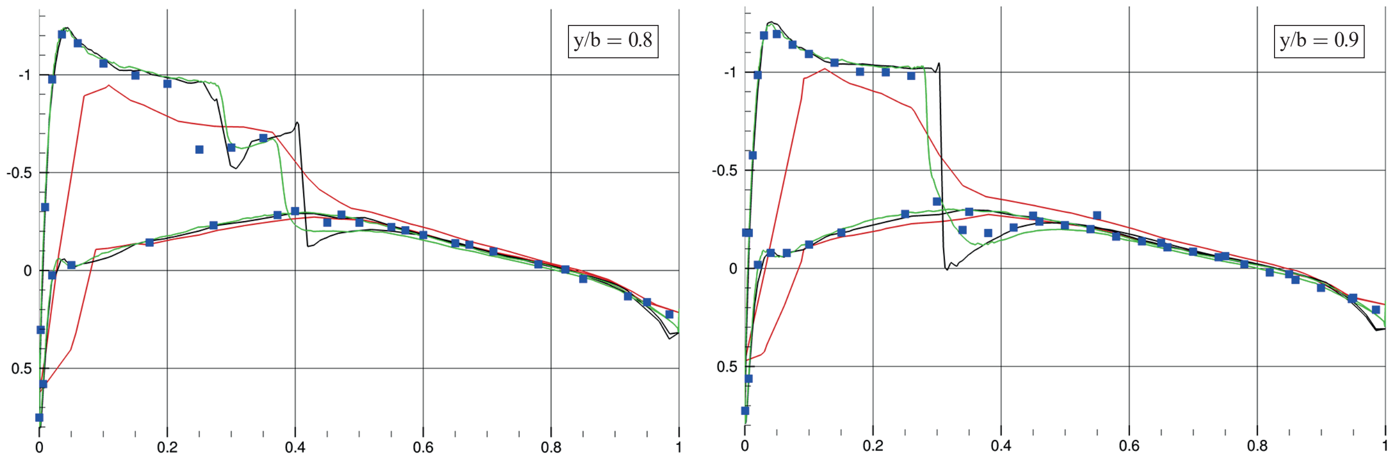

Fig. 23. The same as figure 22

References

[1] Y. Bourgault, M. Picasso, F. Alauzet, and A. Loseille, "On the use of anisotropic a posteriori error estimators for the adaptative solution of 3D inviscid compressible flows", *International Journal for Numerical Methods in Fluids* 59 (1), 47–74 (2009).

[2] F. Alauzet, "Size gradation control of anisotropic meshes", *Finite Elem. Anal. Des*. 46, 181–202 (2010).

[3] M.J. Castro-Diaz, F. Hecht, and B. Mohammadi, "New progress in anisotropic grid adaptation for inviscid and viscous flows simulations", Tech. Rep. RR-2671, INRIA, 1995.

[4] J. Majewski, "An anisotropic adaptation for simulation of compressible flows", *Mathematical Modelling and Analysis* 7, 127–134 (2002).

[5] G. Karypis and V. Kumar, *MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0*, 1995.

[6] S. Patkar and H. Narayanan, "An efficient practical heuristic for good ratio-cut partitioning", *Proceedings of 16th International Conference on VLSI Design, 2003*, 64–69 (2003).

[7] B. Hendrickson and T.G. Kolda, "Graph partitioning models for parallel computing", *Parallel Computing* 26 (12), 1519–1534 (2000).

[8] A.E. Feldmann, "Fast balanced partitioning is hard even on grids and trees", *Theoretical Computer Science* 485, 61–68 (2013).

[9] C. Troyer, D. Baraldi, D. Kranzlmuller, H. Wilkening, and J. Volkert, "Parallel grid adaptation and dynamic load balancing for a CFD solver", *Lecture Notes in Computer Science* 3666, 493–501 (2005).

[10] L. Oliker, R. Biswas, and H.N. Gabow, "Parallel tetrahedral mesh adaptation with dynamic load balancing", *Parallel Comput*. 26 (12), 1583–1608 (2000).

[11] B. Hendrickson and K. Devine, "Dynamic load balancing in computational mechanics", *Computer Methods in Applied Mechanics and Engineering* 184 (2–4), 485–500 (2000).

[12] S. Gepner and J. Rokicki, "Dynamic load balancing for parallelization of adaptive algorithms", in *ADIGMA – A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications*, vol. CXII, pp. 327–338, eds. N. Kroll, H. Bieler, H. Deconinck, V. Couaillier, H. van der Ven, and K. Sorensen, Springer-Verlag, Berlin, 2010.

[13] S. Gepner, J. Majewski, and J. Rokicki, "Dynamic load balancing for adaptive parallel flow problems", in *Parallel Processing and Applied Mathematics. PPAM 2009*, eds. R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Waśniewski, Springer, Berlin, 2010.

[14] K.D. Devine, E.G. Boman, R.T. Heaphy, B.A. Hendrickson, J.D. Teresco, J. Faik, J.E. Flaherty, and L.G. Gervasio, "New challenges in dynamic load balancing", *Applied Numerical Mathematics* 52 (2–3), 133–152 (2005).

[15] J.M.S. Gepner and J. Rokicki, "Parallel performance of adaptive algorithms with dynamic load balancing", *5th European Conference on Computational Fluid Dynamics ECCOMAS CFD 2010*, (2010).

[16] S. Gepner, J. Majewski, and J. Rokicki, "Parallel efficiency of an adaptive, dynamically balanced flow solver", in *Parallel Processing and Applied Mathematics*, pp. 541–550, eds. R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Waśniewski, Springer, Berlin, 2014.

[17] P.J. Frey and F. Alauzet, "Anisotropic mesh adaptation for CFD computations", *Comput. Methods Appl. Mech. Engrg* 194 (48–49), 5068–5082 (2005).

[18] O.C. Zienkiewicz and J.Z. Zhu, "A simple error estimator and adaptive procedure for practical engineering analysis", *International Journal for Numerical Methods in Engineering* 24 (2), 337–357 (1987).

[19] O.C. Zienkiewicz and J.Z. Zhu, "The superconvergent patch recovery and a posteriori error estimates. Part 1: The recovery technique", *International Journal for Numerical Methods in Engineering* 33 (7), 1331–1364 (1992).

[20] O.C. Zienkiewicz and J.Z. Zhu, "The superconvergent patch recovery and a posteriori error estimates. Part 2: Error estimates and adaptivity", *International Journal for Numerical Methods in Engineering* 33 (7), 1365–1382 (1992).

[21] P.E. Farrell, S. Micheletti, and S. Perotto, "An anisotropic Zienkiewicz-Zhu-type error estimator for 3d applications", *International Journal for Numerical Methods in Engineering* 85 (6), 671–692 (2011).

[22] S. Micheletti and S. Perotto, "Reliability and efficiency of an anisotropic Zienkiewicz-Zhu error estimator", *Computer Methods in Applied Mechanics and Engineering* 195 (9–12), 799 – 835 (2006).

[23] F. Chalot, "Goal-oriented mesh adaptation in an industrial stabilized finite element Navier-Stokes code", in *ADIGMA – A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications*, pp. 369–385, eds. N. Kroll, H. Bieler, H. Deconinck, V. Couaillier, H. van der Ven, and K. Sørensen, Springer, Berlin, 2010.
[24] A. Loseille, A. Dervieux, and F. Alauzet, "Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations", *J. Comput. Physics* 229 (8), 2866–2897 (2010).
[25] R. Hartmann, "Adjoint consistency analysis of discontinuous Galerkin discretizations", *SIAM J. Numer. Anal*. 45 (6), 2671–2696 (2007).
[26] L. Tourrette, M. Meaux, and A. Barthet, "Adjoint-based correction of aerodynamic coefficients on structured multiblock grids", in *ADIGMA – A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications*, pp. 355–368, eds. N. Kroll, H. Bieler, H. Deconinck, V. Couaillier, H. van der Ven, and K. Sørensen, Springer, Berlin, 2010.
[27] D.A. Venditti and D.L. Darmofal, "Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows", *J. Comput. Phys* 187, 22–46 (2003).
[28] D.J. Mavriplis, "Adaptive mesh generation for viscous flows using Delaunay triangulation", *J. Comput. Phys*. 90, 271–291 (1990).
[29] J. Peraire, J. Peiró, and K. Morgan, "Adaptive remeshing for three-dimensional compressible flow computations", *Journal of Computational Physics* 103 (2), 269–285 (1992).
[30] P.J. Frey and F. Alauzet, "Anisotropic mesh adaptation for transient flows simulations", *Proc. of 12th Int. Meshing Roundtable*, 335–348 (2003).
[31] M.J. Castro-Diaz, F. Hecht, B. Mohammadi, and O. Pironneau, "Anisotropic unstructured mesh adaptation for flow simulation", *International Journal for Numerical Methods in Fluids* 25, 475–491 (1997).
[32] F. Alauzet, P.L. George, B. Mohammadi, P.J. Frey, and H. Borouchaki, "Transient fixed point-based unstructured mesh adaptation", *International Journal for Numerical Methods in Fluids* 43, 729–745 (2003).
[33] J. Majewski and J. Rokicki, "Anisotropic mesh adaptation in the presence of complex boundaries", in *ADIGMA – A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications*, pp. 441–453, eds. N. Kroll, H. Bieler, H. Deconinck, V. Couaillier, H. van der Ven, and K. Sørensen, Springer, Berlin, 2010.
[34] J. Majewski and P. Szałtys, "High-order 3D anisotropic mesh adaptation for high-Reynolds number flows", in *IDIHOM FP7 – Transport*, AAT.2010.1.1–1., AAT.2010.4.1–1., 2010.
[35] J. Majewski and A. Athanasiadis, "Anisotropic solution adaptive technique applied to simulations of steady and unsteady compressible flows", in *Computational Fluid Dynamics 2006*, eds. H. Deconinck and E. Dick, 2009.
[36] J. Majewski, "Anisotropic adaptation for flow simulations in complex geometries", in *36th Lecture Series on Computational Fluid Dynamics / ADIGMA course on HP-adaptive and HP-multigrid methods*, ed. H. Deconinck, von Karman Institute for Fluid Dynamics, 2009.
[37] M.A. Park and D.L. Darmofal, "Parallel anisotropic tetrahedral adaptation", in *46th AIAA Aerospace Sciences Meeting and Exhibit*, vol. 917, 2008.
[38] F. Alauzet, X. Li, E.S. Seol, and M.S. Shephard, "Parallel anisotropic 3D mesh adaptation by mesh modification", *Engineering with Computers* 21 (3), 247–258 (2006).
[39] A. Loseille and R. Löhner, "On 3D anisotropic local remeshing for surface, volume and boundary layers", in *Proceedings of the 18th International Meshing Roundtable*, pp. 611–630, ed. B.W. Clark, Springer, Berlin, 2009.
[40] G. Compère, J.-F. Remacle, J. Jansson, and J. Hoffman, "A mesh adaptation framework for dealing with large deforming meshes", *International Journal for Numerical Methods in Engineering* 82 (7), 843–867 (2010).
[41] Y.M. Park and O.J. Kwon, "A parallel unstructured dynamic mesh adaptation algorithm for 3D unsteady flows", *International Journal for Numerical Methods in Fluids* 48, 671–690 (2005).
[42] X. Li, M.S. Shephard, and M.W. Beall, "3D anisotropic mesh adaptation by mesh modification", *Computer Methods in Applied Mechanics and Engineering* 194 (48–49), 4915–4950 (2005).
[43] Z. Zhu, P. Wang, and S. Tuo, "An adaptive solution of the 3D Euler equations on an unstructured grid", *Acta Mechanica* 155, 215–231 (2002).
[44] J. Waltz, "Parallel adaptive refinement for unsteady flow calculations on 3D unstructured grids", *International Journal for Numerical Methods in Fluids* 46, 37–57 (2004).
[45] M.-C. Rivara, "New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations", *International Journal for Numerical Methods in Engineering* 40 (18), 3313–3324 (1997).
[46] E. Bänsch, "Local mesh refinement in 2 and 3 dimensions", *IMPACT of Computing in Science and Engineering* 3 (3), 181–191 (1991).
[47] D. Arnold, A. Mukherjee, and L. Pouly, "Locally adapted tetrahedral meshes using bisection", *SIAM Journal on Scientific Computing* 22 (2), 431–448 (2000).
[48] A. Rajagopal and S. Sivakumar, "A combined r-h adaptive strategy based on material forces and error assessment for plane problems and bimaterial interfaces", *Computational Mechanics* 41 (1), 49–72 (2007).
[49] B. Ong, R. Russell, and S. Ruuth, "An h-r moving mesh method for one-dimensional time-dependent PDEs", in *Proceedings of the 21st International Meshing Roundtable*, pp. 39–54, eds. X. Jiao and J.-C. Weill, Springer, Berlin, 2013.
[50] M. Scherer, R. Denzer, and P. Steinmann, "Energy-based r-adaptivity: A solution strategy and applications to fracture mechanics", in *Defect and Material Mechanics*, pp. 117–132, eds. C. Dascalu, G. Maugin, and C. Stolz, Springer, Dordrecht, 2008.
[51] W. Bauer, M. Baumann, L. Scheck, A. Gassmann, V. Heuveline, and S.C. Jones, "Simulation of tropical-cyclone-like vortices in shallow-water icon-hex using goal-oriented r-adaptivity", *Theoretical and Computational Fluid Dynamics* 28 (1), 107–128 (2014).
[52] C.J. Budd, W. Huang, and R.D. Russell, "Adaptivity with moving grids", *Acta Numerica* 18, 111–241 (2009).
[53] S. Gepner, *Adaptacja Siatek i Przetwarzanie Równoległe w Symulacji Przepływów Transonicznych*, PhD thesis, Faculty of Power and Aeronautical Engineering, Warsaw University of Technology, 2014.
[54] H. Deconinck, K. Sermeus, and R. Abgrall, "Status of multidimensional upwind residual distribution schemes and applications in aeronautics", *AIAA Conference Proceedings*, 2000–2328 (2000).

[55] M. Ricchiuto, *Construction and Analysis of Compact Residual Discretizations for Conservation Laws on Unstructured Meshes*, von Karman Institute for Fluid Dynamics, 2005.

[56] M. Ricchiuto, *Construction and Analysis of Compact Residual Discretizations for Conservation Laws on Unstructured Meshes*, PhD thesis, Faculty of Applied Science, Université Libre de Bruxelles, 2005.

[57] K. Sermeus and H. Deconinck, "Solution of steady Euler and Navier-Stokes equations using residual distribution schemes", in *33rd Lecture Series on Computational Fluid Dynamics – Novel Methods for Solving Convection Dominated Systems (LS2003- 05)*, von Karman Institute for Fluid Dynamics, Rhode Saint Genèse, Belgium, 2003.

[58] L.M. Mesaros and J. Matthew, *Multi-Dimensional Fluctuation Splitting Schemes For The Euler Equations On Unstructured Grids*, PhD thesis, University of Michigan, 1995.

[59] P.L. George and H. Borouchaki, *Delaunay Triangulation and Meshing – Application to Finite Elements*, Hermes, 1998.

[60] L. Formaggia and S. Perotto, "Anisotropic error estimation for finite element methods", in *31st Computational Fluid Dynamics*, eds. N.P. Weatherill and H. Deconinck, von Karman Institute for Fluid Dynamics, 2000.

[61] J. Ruppert and R. Seidel, "On the difficulty of triangulating three-dimensional nonconvex polyhedra", *Discrete & Computational Geometry* 7 (1), 227–253 (1992).

[62] E. Schönhardt, "Über die zerlegung von dreieckspolyedern in tetraeder", *Mathematische Annalen* 98 (1), 309–312 (1928), [in German].

[63] O.C. Zienkiewicz and R.L. Taylor, *Finite Element Method: Volume 1 – The Basis*, 5th ed., Butterworth-Heinemann, Oxford, 2000.

[64] J. Rokicki, J. Żółtak, D. Drikakis, and J. Majewski, "Parallel performance of overlapping mesh technique for compressible flows", *Future Generation Computer Systems* 18 (1), 3–15 (2001).

[65] G. Karypis, "ParMETIS – parallel graph partitioning and fill-reducing matrix ordering", http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview.

[66] "Onera M6 Wing", in *NPARC Alliance Verification and Validation Archive*, http://www.grc.nasa.gov/WWW/wind/valid/m6wing/m6wing.html.