Adaptive grid refinement for hydrodynamic flows

Jeroen Wackers, Ganbo Deng, Alban Leroyer, Patrick Queutey, Michel Visonneau

Adaptive grid refinement for hydrodynamic flows

Jeroen Wackers *, Ganbo Deng, Alban Leroyer, Patrick Queutey, Michel Visonneau

Laboratoire de Mécanique des Fluides, CNRS UMR 6598, Ecole Centrale de Nantes, 1 rue de la Noë, B.P. 92101, 44321 Nantes Cedex 03, France

An adaptive grid refinement method is presented for hydrodynamic flow simulation. It is meant for appli-cation to a wide range of realistic flow problems, so generality and flexibility of the method are essential. Directional refinement is developed to be used with unstructured hexahedral meshes, tensor-based refinement allows the implementation of many different refinement criteria. Good grid quality is assured by creating buffers of refined cells around relevant flow features. Tests are performed with two refine-ment criteria, based on the free surface and the Hessian matrix of the solution respectively; these show great increases in efficiency with respect to non-refined grids.

1. Introduction

Automatic grid refinement, the adaptation of a computational grid to a flow solution by locally dividing the grid cells into smaller cells, is an ideal way to efficiently solve flow problems that have strong local structures whose position is not known a priori. Many such problems appear in marine hydrodynamics. For example, marine flows often have a free water surface. If this surface is numerically modelled with any sort of capturing technique, a good local resolution is needed to accurately resolve its position. Furthermore, steep and breaking waves usually appear in a few specific areas only. Finally, a ship's hull and appendages often create coherent, highly localised vortical structures. To simulate the flow phenomena in each of these examples, automatic grid refinement is a very interesting technique.

Grid refinement has been used in fluid dynamics simulations for a long time. It was used first on cartesian grids, see for example [1–7]. These methods typically feature special interpolation techniques on the boundaries between coarser and finer cells, to circumvent the problem of the so-called ''hanging nodes'', i.e. nodes that are present in the fine cells but not in the coarse cells. Later, the focus shifted towards the refinement of unstructured tetrahedral meshes. For local grid refinement, their great advantage is that by adding extra tetrahedra around the zones of grid refinement, hanging nodes can be eliminated. Thus, flow solvers do not have to be strongly modified to accommodate the refined grids. Nowadays, refinement of tetrahedral grids has reached a high level of maturity [8,9].

* Corresponding author. Tel./fax: +33 2 40 37 16 81.
E-mail address: jeroen.wackers@ec-nantes.fr (J. Wackers).

Recently, interest in hexahedral and quadrangular grids has rekindled within the finite-volume community with the emergence of unstructured hexahedral grid generators. The first of these was the HEXPRESS™ grid generator from Numeca Inc. [10], recently other grid generators have appeared like the snappyHexMesh generator of OpenFOAM [11]. The resulting meshes (see Fig. 1) contain large parts which have a structured character and thus produce the accuracy associated with structured meshes; these include body-fitted boundary layer meshes. On the other hand, meshes in complex domains can be generated as easily as with all other unstructured grid generators. The associated flow solvers usually accept arbitrary cell topologies, with any number of faces per cell and any number of nodes per face. Thus, the concept of hanging nodes no longer exists: all nodes are considered the same, the flow solver does not even need to know that the grid is hexahedral.

This paper discusses automatic grid refinement for unstructured hexahedral grids. These meshes are ideal for adaptive refinement, as their original form already contains fine cells laying next to coarser neighbour cells. Thus, when some cells of an unstructured hexahedral grid are refined into smaller hexahedra, the result is ... an unstructured hexahedral grid. Therefore, the solvers that accept these meshes can use locally refined grids without any modification.

The novelty of the adaptive grid refinement method described in this paper is, that it is powerful and flexible enough to be used for large-scale realistic computations on unstructured hexahedral meshes, in marine hydrodynamics. The paper describes the need for anisotropic grid refinement in such cases and the way to use refinement criteria to control the locations for this grid refinement. It introduces the application of metric tensors as refinement

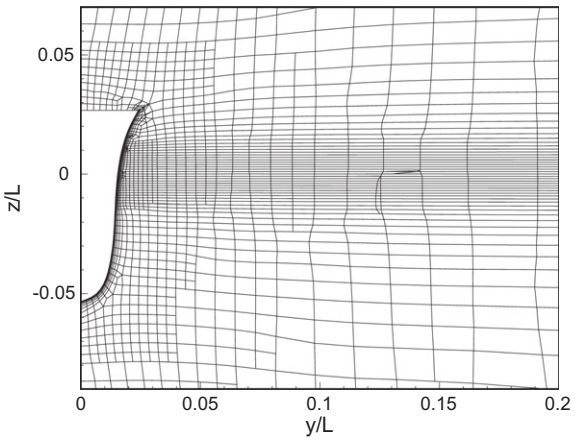


Fig. 1. An example of an unstructured hexahedral grid: x-cross section through the grid around a Series 60 ship, at the bow ($x/L = 0.1$).

criteria, the way we adapted this technique for use with hexahedral grid refinement. Furthermore, refinement criteria are constructed which are well suited for the computation of hydrodynamic flows and the best way to use these criteria is investigated. The refinement method described has been developed for ISIS-CFD, the unstructured finite-volume RANS code created by the Numerical Modelling group of the Laboratoire de Mécanique des Fluides.

Section 2 briefly describes the ISIS-CFD flow solver and Section 3 gives an overview of the refinement algorithm. Then Section 4 focuses on the choice of which cells to refine: the way to evaluate a refinement criterion on unstructured hexahedral grids and the way in which good grid quality is assured during grid refinement. Section 5 introduces the two refinement criteria used in this paper. Finally, Section 6 illustrates the different aspects of the method and its practical application with several numerical examples from marine hydrodynamics. The paper ends with a conclusion.

2. The ISIS-CFD flow solver

Before starting the discussion of grid refinement, we introduce the methods and techniques that form the basis of the ISIS-CFD flow solver. ISIS-CFD, available as a part of the FINE^{TM}/Marine computing suite, is an incompressible unsteady Reynolds-averaged Navier–Stokes (URANS) method. The solver is based on the finite-volume method to build the spatial discretisation of the transport equations. The velocity field is obtained from the momentum conservation equations and the pressure field is extracted from the mass conservation constraint, or continuity equation, transformed into a pressure equation. These equations are similar to the Rhie and Chow SIMPLE method [12], but have been adapted for flows with discontinuous density fields. In the case of turbulent flows, transport equations for the variables in the turbulence model are added to the discretisation. Free-surface flow is simulated with a multi-phase flow approach: the water surface is captured with a conservation equation for the volume fraction of water, discretised with specific compressive discretisation schemes. A detailed description of the solver is given by [13,14].

The unstructured discretisation is face-based, which means that there is no limitation to the type of the cells. Fluxes are computed face by face. The reconstructions of the cell-centred state variables to the face centres are made with interpolations that use the two cells next to a face and their neighbour cells. However, these interpolations use no a priori assumptions about the cell topologies. And while the linearised systems, used to solve iteratively the momentum and pressure equations, are systems of equations in the cell-centred unknowns, these systems are constructed by summing the contributions of the faces to each cell. Thus, no cell topology restrictions apply anywhere, which means that cells with an arbitrary number of arbitrarily shaped faces are accepted.

The solver is mostly used with unstructured hexahedral grids. The grid in Fig. 1 shows the typical features of such meshes: several semi-structured regions, with fully unstructured parts and body-fitted boundary grids near the walls, as well as sudden changes in cell size when the grid goes from one large cell to two or four smaller neighbour cells. Due to its face-based nature, the ISIS-CFD solver needs no specific changes in order to use these meshes, they are treated just the same as any other type of mesh.

The flow solver features sophisticated turbulence models: apart from the classical two-equation $k-\varepsilon$ and $k-\omega$ models, the anisotropic two-equation Explicit Algebraic Stress Model (EASM), as well as Reynolds Stress Transport Models, are available, see [15,13]. The technique included for the 6 degree of freedom simulation of ship motion is described by [16]. Time-integration of Newton's laws for the ship motion is combined with analytical weighted or elastic analogy grid deformation to adapt the fluid mesh to the moving ship. Furthermore, the code has the possibility to model more than two phases.

The code is entirely parallelised, based on domain decomposition of the of the mesh. Communication between the processes is achieved with the MPI (Message Passing Interface) toolbox. For brevity, the technical structure of the code is not further described here.

3. Grid refinement technique

The grid refinement method integrated in ISIS-CFD has been developed for use in daily practice for all the applications of this code. Therefore, the method is flexible and general, allowing anisotropic refinement and derefinement for unsteady flows. Like the flow solver, it is fully parallel. The refinement procedure is completely integrated in the flow solver.

Globally, the method works as follows. The flow solver is run on an initial grid for a limited number of time steps. Then the refinement procedure is called; if a refinement criterion, based on the current flow solution, indicates that parts of the grid are not fine enough, these cells are refined and the solution is copied to the refined grid. On this new grid, the flow solver is restarted, again for a limited number of time steps. Then the refinement procedure is called again, to further refine or to derefine the mesh (i.e. to undo earlier refinements). This cycle is repeated several times. When computing steady flows, the procedure eventually converges: once the flow starts to approach a steady state and the grid is correctly adapted to this state according to the refinement criterion, then the refinement procedure keeps being called, but it no longer changes the grids.

From an algorithmic point of view, the computation of the refinement criterion is decoupled from the decision of which cells to refine. Therefore, it is easy to exchange refinement criteria without modifying the remainder of the method. The refinement method contains an automatic adaptive load balancing procedure to keep the grid evenly distributed over the processors. Further details of the algorithm can be found in [17,18].

4. Criterion and decision

The refinement criterion is that part of the algorithm which, based on the flow solution, determines where the grid should be refined or derefined. Our use of unstructured hexahedral grids poses strong requirements on the way this choice is made, and invalidates many classical strategies of refinement criterion

evaluation. In Subsection 4.1, it is explained why directional refinement is mandatory for our unstructured hexahedral grids. Subsection 4.2 presents the criterion evaluation strategy that is, in our opinion, preferable for use with directional refinement. Finally, Subsection 4.3 shows how extra refinement is added in order to guarantee good grid quality.

4.1. Isotropic or directional refinement

The first grid refinement methods were all isotropic (see for example [5]). For isotropic refinement, cells are refined in all their directions at once (a quadrangle is divided in four, a hexahedron in eight, etc.), so the resulting refined small cells have the same shape as the original cells. More recently, some methods have begun to use directional refinement (e.g. [19,20]). Here, cells can be refined in one direction only or in several directions at once, resulting in a more complex algorithm and more possible cell topologies, but greater flexibility in refinement. Fig. 2 gives an illustration of isotropic and directional refinement.

To create refined grids of good quality, it is in principle necessary to control the sizes of the cells in all their directions, in order to guarantee that the cell size in each direction varies smoothly and that a cell fits in well with all its neighbours. In other words, on a general grid, the refinement must control both the size and the aspect ratio of the cells.

Isotropic refinement controls only the size of the cells; their aspect ratios are necessarily the same as those of the original grid. Therefore, isotropic refinement only makes sense when the original mesh is locally isotropic, i.e. when all the cells in a neighbourhood have more or less the same aspect ratio (though local refinement is acceptable, cells may be two times smaller than their neighbours), see Fig. 3a. In that case, cells that are being refined still fit in correctly in the mesh, so the quality of the refined mesh is good.

However, in unstructured hexahedral meshes, neighbour cells may have very different aspect ratios (Fig. 3b, see also Fig. 1). Thus, sticking to isotropic refinement will produce irregular meshes, as refining a cell to reduce its size in one direction can produce cells that are too fine in another (in the middle grid of Fig. 3b, the horizontal cell size goes from fine to coarse to fine, which is undesirable). So for unstructured hex meshes, isotropic refinement is impossible; here, it is mandatory to have a method that controls both cell size and aspect ratio.

Another reason why directional refinement is essential for us, is the large-scale 3D nature of our grids. Isotropic refinement in three dimensions is very expensive: each time a cell is refined, it is divided in 8. The only way to keep the total number of cells low, while still obtaining sufficient resolution to capture small flow features, is to permit high cell aspect ratios by allowing refinement in one direction only wherever this is possible. A typical example is the refinement around the water surface, see Subsections 5.1, 6.1 and 6.3.

For these two reasons, the only way to properly refine unstructured hex meshes is to control the refinement in each cell direction separately; directional refinement becomes the only choice.

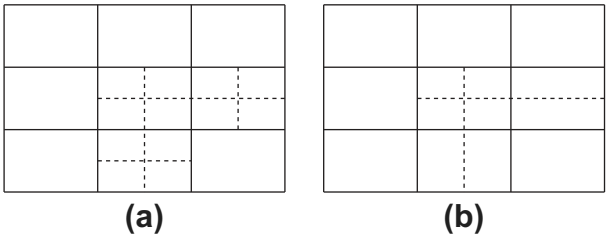


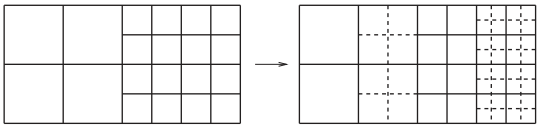
Fig. 2. Illustration of isotropic (a) and directional (b) refinement. 2D examples.

4.2. Metric-based refinement criteria

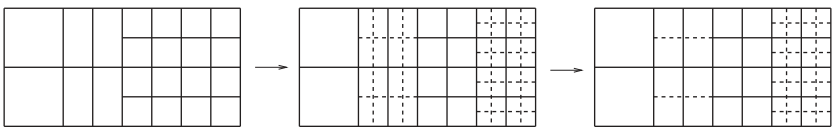
As a result of the choice for directional refinement, the way to compute refinement criteria has to be adapted. Often (see e.g. [21–23,5,7]) for the criterion, a quantity is chosen that decreases when the mesh becomes finer, like an error estimator or a simpler error indicator function. The grid is then refined in those cells where the criterion is largest, with an eventual goal of producing the same value for the criterion in all cells. This procedure is intrinsically isotropic: it chooses which cells to refine, but it cannot choose in which direction to refine them.

Isotropic criteria can be made directional by applying a post-treatment that chooses between isotropic and directional refinement in cells marked for refinement, but we consider this procedure not ideal. First, there is no explicit guarantee that situations like in Fig. 3a are prevented. And second, the procedure is not so flexible; when the refinement criterion is changed, it is neither certain that the existing direction finder works well with the new criterion, nor that a new direction finder for the criterion can be found.

In the authors' opinion, the only generally applicable way to specify refinement criteria for directional refinement is to use a metric: a smooth criterion that is insensitive to the size and orientation of the cells and whose value indicates the locally desired cell size. Cells are then refined to produce actual cell sizes as close as possible to the values demanded by the criterion. Specifically, we use tensorial metrics that allow the specification of desired cell sizes individually in any given direction. Such metrics are in common use for the generation and adaptive refinement of tetrahedral



(a) Locally isotropic grid.



(b) Locally anisotropic grid.

Fig. 3. Isotropic grid refinement is satisfactory on an original grid where all cells have the same aspect ratios (a), otherwise directional refinement is needed (b).

meshes [24–29] and for controlling refinement by mesh deformation [30,31]. We propose to use this technique also for the refinement of hexahedral meshes. With respect to existing methods, there are some differences in the way these metrics are used for hexahedral grid refinement, as will be explained at the end of this subsection.

4.2.1. Metric-based decision

The metric tensor $\mathcal{C}_i(\mathbb{R}^3 \to \mathbb{R}^3)$ in each cell is seen as a geometric operator that transforms each cell Ω_i in the physical space into a deformed cell $\widetilde{\Omega}_i$ in a modified space. Then in each cell, directional refinement is applied to produce a grid that is as nearly uniform as possible, in the modified space (see Fig. 4). The result is an anisotropically refined grid in the real space.

The refinement decision algorithm works as follows. Let the refinement criterion $\mathcal{C}_i$ be a given positive-(semi)definite 3×3 tensor in each hexahedral cell Ω_i, i.e. a tensor having eigenvalues $\geqslant 0$. Then, for each Ω_i, do:

1. Compute $\mathbf{d}_{k,i}$ ($k = 1 \ldots 3$), the distance vector between the two face centres of the hexahedron in the direction k, which is a measure of its size in the direction k. (If a face of the hexahedron is divided in several smaller faces, the average of their face centres is used.)
2. Compute the transformed distances[1]:

$$\tilde{\mathbf{d}}_{k,i} = \mathcal{C}_i \mathbf{d}_{k,i}. \tag{1}$$

3. For each k, make a decision to refine in that direction if:

$$\|\tilde{\mathbf{d}}_{k,i}\| \geqslant T_r, \tag{2a}$$

for a given, constant refinement threshold T_r.
4. Make a decision to derefine the cell if:

$$\|\tilde{\mathbf{d}}_{k,i}\| \leqslant \frac{T_r}{f_{dr}} \quad \forall k \in [1,3], \tag{2b}$$

where f_{dr} is a factor, usually 2.5, to prevent derefined cells from being refined again directly in the next refinement step. While the refinement is directional, derefinement can only be performed if it is permitted in all directions at once. Furthermore, a group of sister cells that once formed one bigger cell can only be derefined if all sisters have a derefinement decision. If not, possible derefinement decisions in some of the sister cells are removed.

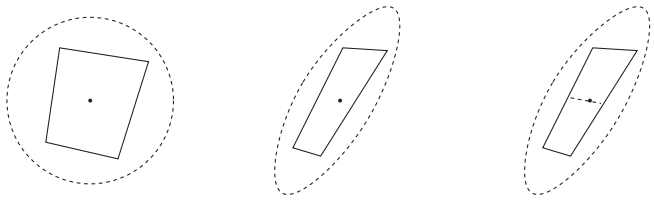
From this procedure, we can see that the resulting grid sizes are inversely proportional to the eigenvalues of the tensors $\mathcal{C}_i$.

4.2.2. Choosing refinement criteria

The procedure is a straightforward and flexible way of giving complete control over the grid refinement to the refinement criterion. By a proper choice of the $\mathcal{C}_i$, desired cell sizes in any given direction can be specified; no separate procedure is needed to determine the direction of the refinement. As a result, it becomes easy to exchange between different criteria, to develop and implement new criteria, as well as to refine based on several criteria at once: the criterion in each cell is then set as the maximum over these criteria (see Subsection 4.3.2 for the computation of the approximate maximum of two tensors).

The combination of metric-based refinement with a fixed original grid gives a very robust procedure. In tetrahedral mesh generation, strong requirements are posed on admissible metric tensors, in order to guarantee good cell orientation, to limit cell skewness and to prevent singular tensors, that would produce infinite-sized cells. For our mesh refinement, the cell skewness and orientations, as well as the maximum cell sizes, are imposed by the original grid. Thus, there is a limitation on the refined meshes that can be generated, but much more freedom in the choice of the metric tensors. For example, singular tensors are not forbidden. On the contrary, we use them to our advantage, to specify refinement in one direction only (see Subsection 5.1 for an example). Any symmetric positive-semidefinite tensor can be used as a refinement criterion, which gives great possibilities for the development of such criteria.

[1] In the literature, usually metric tensors $\mathcal{M}$ are used that are equivalent to our $\mathcal{C}^2$, so $\|\tilde{\mathbf{d}}_{k,i}\|^2 = \mathbf{d}_{k,i}^T \mathcal{M}_i \mathbf{d}_{k,i}$. This is a matter of taste, it does not change the decision procedure at all.



(a) Physical space. (b) Modified space. (c) Refinement.

Fig. 4. Tensor refinement criterion. Cell Ω_i and unit circle (reference) in the physical space (a), deformed cell $\widetilde{\Omega}_i$ and deformed circle after application of the transformation C_i (b), and refinement to create a uniform grid in the modified space (c).

4.3. Buffer layers

The refinement criterion only 'sees' the solution at the instant of refinement. However, this solution is evolving, either because a steady solution is converging, or because the flow is unsteady. In both cases, the grid must not only be refined where the criterion requires it, but also in a few cells around this zone, to obtain a margin of safety that allows the solution to change without having crucial flow features leave the zones of refined cells. We call these layers of safety cells 'buffer layers'.

Our standard type of buffer layer has the same cell sizes as those specified by the criterion. A second type of buffer is needed when the refinement criterion is nearly discontinuous; purely following the criterion would then produce a grid that goes abruptly from very coarse to very fine cells. However, for grid quality it is better to have cell sizes that gradually go from coarse to fine. Thus, cells must be refined around those indicated by the criterion, to a coarser size than those produced by the criterion, to get a gradual variation of cell size.

4.3.1. Adjusting the criterion

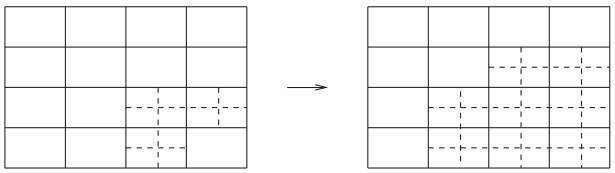
Buffer layers are created by modifying the criterion; their creation can be thought of as extending the zones where the desired cell size is small. To make one buffer layer of the standard type, each cell looks at its neighbour cells and sets its criterion value equal to the maximum value over itself and its neighbours. For a cell i having neighbours nb, the kth buffer layer is set as:

$$(\mathcal{C}_i)^k = \max\left((\mathcal{C}_i)^{k-1}, \max_{nb}(\mathcal{C}_{nb})^{k-1}\right), \qquad k = 1 \cdots N^s, \tag{3}$$

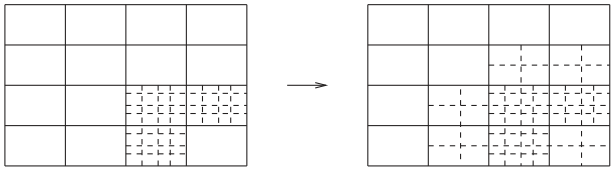
where $(\mathcal{C}_i)^0$ are the values originally computed by the criterion. As a result, for each k the zone where the desired cell size is small enough to give refinement is extended by one layer of cells (Fig. 5a). N^s is the total number of layers.

The buffer type that produces gradual size changes is created in the same way, only a cell takes the maximum of its own criterion and a factor $c_b < 1$ times the criterion values of its neighbours.

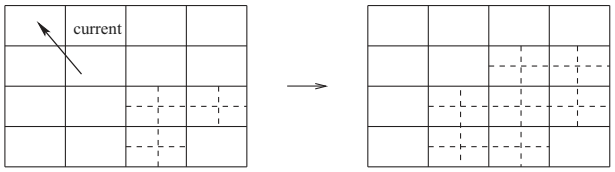
$$(\mathcal{C}_i)^k = \max\left((\mathcal{C}_i)^{k-1}, c_b \max_{nb}(\mathcal{C}_{nb})^{k-1}\right), \qquad k = 1 \cdots N^f. \tag{4}$$



(a) Standard buffer.



(b) Fractional buffer.



(c) Convection buffer.

Fig. 5. Three types of buffer layer: refined mesh without and with application of the buffer.

Thus, the criterion values in each layer will be smaller than in the cells inside it, so the resulting refinement will produce cells whose size gradually increases (Fig. 5b). Our standard choice is $c_b = 0.6$. The standard and fractional buffer layers are created subsequently, the fractional layers are computed last.

For steady problems, the number of buffer layers N^s and N^f is set by the user and kept the same throughout the grid. However, for unsteady problems there is the added possibility of anticipating the evolution of the solution and to refine extra in the direction where the solution is going; this corresponds to convecting the criterion with the flow (Fig. 5c). We create this buffer by making use of the cell face CFL number:

$$Co_f = \frac{\mathbf{u}_f \cdot \mathbf{n}_f S_f \Delta t}{V_c}, \tag{5}$$

where $\mathbf{u}_f$,$\mathbf{n}_f$ and S_f are the velocity, normal vector, and surface of a cell face, V_c is the cell volume of the neighbouring cell and Δt is the time step. Roughly speaking, Co_f indicates the number of cell distances that the solution is displaced in one time step, due to the flow through the face. For each cell, buffers are made looking only at those neighbour cells that give inflow into the cell; the number of times that a neighbour value is copied is the integer value of the CFL number for the face between the cell and its neighbour, multiplied by the number of time steps between successive refinements n_Δ. This roughly approximates the displacement of the solution between refinements, in the direction of the face.

$$(C_i)^k = \max\left((C_i)^{k-1}, \max_{nb|\mathbf{u}_f\cdot\mathbf{n}_f<0, k\leqslant Co_f n_\Delta}(C_{nb})^{k-1}\right), \qquad k = 1\cdots N^{c_{\max}}, \tag{6}$$

Thus, the larger the face CFL number, the more buffer layers are created in the direction of the face; the buffer layers follow the motion of the fluid. $N^{c_{\max}}$ is the maximum over all the faces of the number of layers to be generated.

The reason for creating buffer layers using the criterion instead of the decision, is that it is much easier to spread the criterion (which indicates desired cell sizes in a global reference frame), than the decisions that do not only take into account the desired size of a cell, but also its actual size, and that furthermore are based on each cell's local reference frame. Also, on unstructured grids it is not always possible to create perfectly regular increases in cell size like in the examples of Fig. 5; this depends on the available cell sizes in the original grid. Creating the buffer layers with the criterion ensures at least that the target cell sizes are well spread out and smooth, the refinement decision then does the best it can to get the actual cell sizes close to this target. Real buffer layers can be seen in the impacting prism test of Subsection 6.3.

4.3.2. Maximum of two tensors

Finally, the computation of the maximum of two criteria which is needed for the creation of buffer layers is far from trivial. Ideally, the maximum of two tensors $\mathcal{A}_M = \max(\mathcal{A}_1, \mathcal{A}_2)$ would transform any unit vector into one at least as long as the ones produced by the two original tensors. Computing this maximum exactly is computationally expensive, so we use an approximation that is based on an improvement of the procedure proposed by George and Borouchaki [27]. Let $\lambda_{k,j}$, $\mathbf{v}_{k,j}$, $k = 1\ldots3$, $j = 1, 2$ be the eigenvalues and eigenvectors of $\mathcal{A}_j$, with the order of the eigenvalues (index k) for each tensor chosen such that the directions of the eigenvectors in the two tensors correspond as closely as possible, i.e. $\mathbf{v}_{k,1}\cdot\mathbf{v}_{k,2} \geqslant \mathbf{v}_{k,1}\cdot\mathbf{v}_{m,2}\forall m \neq k$. To get a uniquely defined ordering, let $\mathbf{v}_{1,1}\cdot\mathbf{v}_{1,2} \geqslant \mathbf{v}_{2,1}\cdot\mathbf{v}_{2,2} \geqslant \mathbf{v}_{3,1}\cdot\mathbf{v}_{3,2}$. Then compute the modified eigenvalues:

$$\hat{\lambda}_{k,j} = \max(\lambda_{k,j}, \|\mathcal{A}_{j_2}\mathbf{v}_{k,j}\|), \tag{7}$$

where $\mathcal{A}_{j_2}$ is the other matrix: $j_2 = 3 - j$. Both tensors $\widehat{\mathcal{A}}_j$ (having $\hat{\lambda}_{k,j}$ and $\mathbf{v}_{k,j}$ as eigenvalues and eigenvectors) are approximations to the maximum tensor. $\mathcal{A}_M$ is computed as a weighted average of these two:

$$\lambda_{k,M} = \frac{d_1\hat{\lambda}_{k,1} + d_2\hat{\lambda}_{k,2}}{d_1 + d_2}, \tag{8}$$

where

$$d_j = \sum_{k=1}^{3}\max(0, \hat{\lambda}_{k,j} - \lambda_{k,j}). \tag{9}$$

The eigenvectors are averaged in the same way:

$$\tilde{\mathbf{v}}_{k,M} = \frac{d_1\mathbf{v}_{k,1} + d_2\mathbf{v}_{k,2}}{d_1 + d_2}, \tag{10}$$

then corrected to make them orthogonal:

$$\mathbf{v}_{1,M} = \tilde{\mathbf{v}}_{1,M}, \quad \mathbf{v}_{2,M} = \frac{\tilde{\mathbf{v}}_{2,M} - \tilde{\mathbf{v}}_{1,M}(\tilde{\mathbf{v}}_{2,M}\cdot\tilde{\mathbf{v}}_{1,M})}{\|\tilde{\mathbf{v}}_{2,M} - \tilde{\mathbf{v}}_{1,M}(\tilde{\mathbf{v}}_{2,M}\cdot\tilde{\mathbf{v}}_{1,M})\|}, \quad \mathbf{v}_{3,M} = \mathbf{v}_{1,M}\times\mathbf{v}_{2,M}. \tag{11}$$

The advantages of this procedure are, first, that when $\mathcal{A}_2$ is smaller than $\mathcal{A}_1$ in all directions, the procedure keeps $\mathcal{A}_M = \mathcal{A}_1$, and the opposite (this is not the case for the method of George and Borouchaki), and second, that it can be computed entirely using an eigenvalue–eigenvector decomposition of the tensors. So to save computational effort, we compute the eigenvalues and eigenvectors of the refinement criterion in each cell once (using the method of Scherzinger and Dohrmann [32]), then create all the buffer layers, and finally transform the criterion back to standard matrix form.

5. Refinement criteria

In practice, there are two logical ways to construct tensorial refinement criteria. The first is to base the criterion on a quantity that is in itself a tensor; an example is the well-known Hessian-based

refinement. Another way is to start from the eigenvalue–eigenvector decomposition of the tensor $\mathcal{C}_i$:

$$\mathcal{C}_i = V \Lambda V^T, \tag{12}$$

where $\Lambda = \mathrm{diag}\,(\lambda_1, \lambda_2, \lambda_3)$ is the eigenvalue matrix of $\mathcal{C}_i$ and V has its orthonormal eigenvectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ as columns. Based on a given criterion, these eigenvalues and eigenvectors can be specified. The tensor criterion is then computed with Eq. (12).

In this section, two criteria that are well suited for ship flow computations are presented, one based on each of the two construction methods described above. The first criterion, refinement around the water surface, is given in Section 5.1. Our version of the Hessian-based refinement is explained in Section 5.2. For each criterion, its computation as well as the way to use it are described.

5.1. Free surface criterion

For flow simulation with a free water surface, refinement at the position of the water surface is a logical choice, especially for flow solvers like ISIS-CFD where the water surface is resolved with a multi-phase approach. For these methods, the water surface appears as a numerical discontinuity in the volume fraction of water, so a good grid resolution is essential to resolve the surface well.

Therefore, a criterion is introduced that refines in the neighbourhood of the water surface. Directional refinement is employed to refine the grid in the direction normal to the surface only: where the free surface is diagonal with respect to the grid directions, isotropic refinement is used, but where the surface is horizontal, directional refinement is chosen. Thus, directional refinement is used at the undisturbed water surface and in smooth wave crests and troughs. Directional refinement is essential to keep the number of grid cells low, as the water surface is often nearly undisturbed in most of the domain. Fig. 6 gives an illustration of this refinement principle, images of actual refined grids can be found further on in Figs. 17 and 21.

In the context of tensorial criteria, refinement normal to the free surface is obtained by specifying tensors with only one non-zero eigenvalue, associated with an eigenvector normal to the surface. The result is, that the modified cells $\widetilde{\Omega}_i$ have size zero in all other directions: when the $\mathbf{d}_{k,i}$ are transformed to $\tilde{\mathbf{d}}_{k,i}$ (Eq. (1)), only the component normal to the surface remains. When a cell is aligned with the surface, its two dimensions parallel to the surface are transformed to zero, so according to Eq. (2a), the cell will never be refined in those directions.

The criterion is based on the volume fraction of water c_i: cells are refined when they have a value of c_i that is neither zero nor one. The normal direction is computed from the gradient of c_i. The criterion is computed as follows:

1. Starting from the field c_i, this field is copied to a temporary variable c_{iA}, which is then smoothed to smear out the numerical discontinuity.

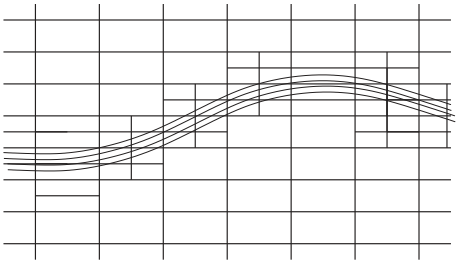


Fig. 6. Isotropic and directional refinement at the free surface. The curves represent volume fraction isolines.

2. From the smooth c_{iA}, the gradient ∇c_{iA} is computed using the Gauss method which is standard in ISIS-CFD.
3. In each cell, a vector $\mathbf{e}_s$ is computed as:

$$\mathbf{e}_s = \begin{cases} \frac{\nabla c_{iA}}{\|\nabla c_{iA}\|} & \text{if } 0.1 \leqslant c_i \leqslant 0.9, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{13a}$$

This is a unit normal vector in all the cells where refinement is desired.

4. The tensor criterion is constructed with an eigenvector in the direction of $\mathbf{e}_s$ having an eigenvalue of 1 or 0 and with the two other eigenvalues always equal to zero. So Eq. (12) becomes:

$$\mathcal{C}_i = \mathbf{e}_s \otimes \mathbf{e}_s, \tag{13b}$$

where $\otimes$ denotes the tensor product.

As a result of choosing all non-zero eigenvalues of $\mathcal{C}_i$ equal to one, the refinement threshold in Eq. (2) can be interpreted directly as the target value for the cell size normal to the free surface. This makes it very easy to choose the threshold.

The criterion as computed above is non-zero in a very narrow strip of cells only. To account for small movements of the water surface, buffer layers (Section 4.3) are essential for this criterion. Our usual practice is to apply two layers of the standard type, or more (up to four) for unsteady problems. Due to the discontinuous nature of the criterion, it may also be necessary to use fractional buffers to let the grid size go gradually to the unrefined size. The number of fractional buffer layers that are necessary depends on the size difference between the original and maximally refined cells.

Using grid refinement at the water surface has two effects. First, small details of the water surface are resolved better. Secondary waves in a ship's wake become sharper and clearer, bow waves steepen up and overturning breaking waves become more pronounced. Droplets, jets and spray are sharper and stay resolved longer before breaking up. And second, numerical dissipation of waves is reduced, which means that wave systems travelling far from their source are better resolved; this is interesting for computing the far field of a ship's wake, or for generating an incoming wave train that interacts with an object.

However, the refined grid generated by the criterion alone is not enough to resolve waves well. In order to capture the orbital velocity field associated with the waves, a reasonably fine grid is needed in a zone below the water surface. This grid must be part of the original mesh, as the criterion does not create it. Experience suggests that for ISIS-CFD, waves are best resolved when the automatically refined grid at the surface is twice finer than the original mesh around it; refining more than that at the water surface does not improve the accuracy. Thus, a sensible meshing strategy is to generate an original grid having rather fine cells in a large zone around the expected location of the water surface and then to set the refinement threshold for a grid refinement target of half that existing mesh size. In the following Section 6, applications of the free-surface criterion to different ship flow problems can be found.

5.2. Hessian-based criteria

In flows that are not dominated by free-surface waves, other refinement criteria are needed. For such flows, the matrix of second spatial derivatives of the solution is a common choice as a refinement criterion, often used with success in practice. Depending on the state variable chosen for the computation, such a criterion reacts to most of the relevant features of a flow. Contrary to the free-surface criterion, it is therefore able to generate a fine grid all by itself, starting from an original grid that is

uniformly coarse. For numerical methods that use a linear approximation of the solution, the interpolation error on the grid is proportional to the second derivatives. Thus, the matrix of second spatial derivatives is generally interpreted as a local error estimator [24,25,33,34,29].

The Hessian matrix of second spatial derivatives $\mathcal{H}_q$, for a given state variable q, is a 3×3 symmetric tensor:

$$\mathcal{H}_q = \begin{bmatrix} q_{xx} & q_{xy} & q_{xz} \\ q_{xy} & q_{yy} & q_{yz} \\ q_{xz} & q_{yz} & q_{zz} \end{bmatrix} \tag{14}$$

Therefore, it can be used almost directly as a tensor refinement criterion. However, assuming that an indication of the local error is given by $\mathcal{H}_q$ times the cell sizes squared (which is reasonable for a second-order accurate discretisation), equidistribution of this error indicator leads to:

$$\mathcal{C}_i = \left(\mathcal{H}_q^{\frac{1}{2}}\right)_i, \tag{15}$$

where $\mathcal{H}_q^{\frac{1}{2}}$ has the same eigenvectors as $\mathcal{H}_q$ and eigenvalues that are the square roots of the absolute values of those of $\mathcal{H}_q$. We adopt this $\mathcal{C}_i$ as refinement criterion.

5.2.1. Third-order least-squares approximation

The major difficulty in using the Hessian tensor as a refinement criterion is the evaluation of the second derivatives in a sufficiently smooth and accurate way, independent of the mesh. A danger in grid refinement is that local refinement may perturb the computation of the criterion. Thus, it starts to react more to the presence of an already refined grid than to the quantity it is based on. To prevent this undesired effect, numerical errors in the computed second derivatives must be significantly smaller than the derivatives themselves in all cells. In practice, this requires at least a second-order accurate discretisation.

A particular problem of unstructured hexahedral meshes is, that the grid regularity does not increase when the mesh becomes finer. For structured grids, and even for most unstructured tetrahedral meshes, when the grid is refined the cells get more and more the same shape and size as their neighbours. On unstructured hex meshes however, independent of the cell sizes, there will always be cells that are two times smaller than their direct neighbours. This means that numerical schemes which rely on mesh regularity to get good accuracy are not suited for these meshes; a useful scheme must give sufficient accuracy for arbitrary cell configurations.

For the computation of second derivatives, we use a least-squares method based on third-order polynomials. Let $P_j(\mathbf{x})$, $j = 1 \cdots 20$ be the set of basic three-dimensional polynomial functions in $\mathbf{x}$ of up to third order (i.e. $P_1(\mathbf{x}) = 1$, $P_2(\mathbf{x}) = x$, $P_3(\mathbf{x}) = y$, ..., $P_5(\mathbf{x}) = x^2$,..., $P_{11}(\mathbf{x}) = x^3$, ..., $P_{20}(\mathbf{x}) = xyz$). Let $\mathbf{I}$ be the vector of cell indices of a cell i, its neighbours and its neighbours' neighbours. Then we shall search coefficients $\boldsymbol{\beta}$ such that the polynomial:

$$p_i(\mathbf{x}) = \sum_{j=1}^{20} \beta_j P_j(\mathbf{x} - \mathbf{x}_i), \tag{16}$$

is the closest fit to the values of q in the cell centres of $\mathbf{I}$, within the space defined by the set $P_j(\mathbf{x} - \mathbf{x}_i)$. The computation of this least-squares fit is standard. Define the matrix A and vector $\mathbf{b}$ as:

$$A_{jk} = P_j(\mathbf{x}_{\mathbf{I}_k} - \mathbf{x}_i), \qquad b_k = q_{\mathbf{I}_k}, \qquad j = 1 \cdots 20, \quad k = 1 \cdots \text{size}(\mathbf{I}). \tag{17}$$

Then $\boldsymbol{\beta}$ is found as:

$$\boldsymbol{\beta} = (A^T A)^{-1} A^T \mathbf{b}. \tag{18}$$

The approximated Hessian is based on the second derivatives of p_i. The least-squares procedure guarantees that the difference between p_i and the exact function q is not in the space of $P_j(\mathbf{x} - \mathbf{x}_i)$, therefore it is at least a fourth-order polynomial in $\mathbf{x} - \mathbf{x}_i$. Thus, for sufficiently smooth q the second derivatives of p_i are a second-order accurate approximation to those of q, independent of the geometry of the cells in $\mathbf{I}$.

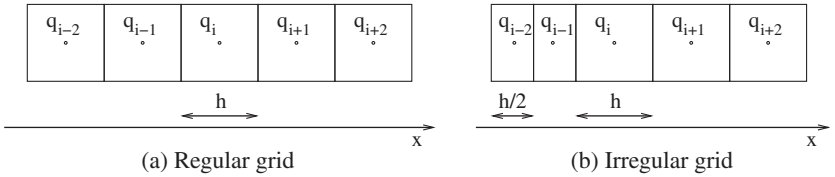
5.2.2. Tests of the least-squares scheme

To assess the effectiveness of the third-order least-squares (LS3) method, we shall compare it with two simpler methods. The first (LS2) uses the same least-squares technique, but with quadratic polynomials, i.e. $P_j(\mathbf{x})$, $j = 1 \cdots 10$. The second is the well-known Gauss integration technique that is advised for Hessian computation by many authors (e.g. [24,20]) and that is used in ISIS-CFD for the evaluation of gradients [14]. The Gauss method starts with a linear interpolation of q from the cell centres to the faces. This is followed by an integration over the cell faces of q multiplied by the component of the face normal in a certain direction, to give the cell-centre gradient in that direction. The procedure is repeated with the gradients, to find the second derivatives.

As a first test, we explicitly evaluate the (5-point) schemes produced by the three methods on two 1D grids, a regular grid and an irregular one having two cells smaller than the other three (Fig. 7). The accuracy of each scheme is determined by inserting a Taylor series development of q into the scheme and computing the continuous equivalent of each scheme. The results are given in Table 1. As we can see, on the regular grid all three methods produce second-order accurate schemes (the LS2 and LS3 schemes

Table 1
Explicit evaluation of the schemes produced by three second-derivative computation methods on the regular and the irregular 1D mesh, and the continuous equivalent of each scheme (up to the leading-order truncation error).

Method	Scheme	Equivalent scheme
Regular grid		
LS 3	$(0.2857q_{i-2} - 0.1429q_{i-1} - 0.2857q_i - 0.1429q_{i+1} + 0.2857q_{i+2})/h^2$	$q_{xx} + 0.3690h^2 q_{xxxx}$
LS 2	$(0.2857q_{i-2} - 0.1429q_{i-1} - 0.2857q_i - 0.1429q_{i+1} + 0.2857q_{i+2})/h^2$	$q_{xx} + 0.3690h^2 q_{xxxx}$
Gauss	$(0.2500q_{i-2} - 0.5000q_i + 0.2500q_{i+2})/h^2$	$q_{xx} + 0.3333h^2 q_{xxxx}$
Irregular grid		
LS 3	$(0.8941q_{i-2} - 0.6020q_{i-1} - 0.8206q_i + 0.3908q_{i+1} + 0.1377q_{i+2})/h^2$	$q_{xx} + 0.1911h^2 q_{xxxx}$
LS 2	$(0.4303q_{i-2} - 0.0523q_{i-1} - 0.4473q_i - 0.3599q_{i+1} + 0.4293q_{i+2})/h^2$	$q_{xx} + 0.3760h\ q_{xxx}$
Gauss	$(0.6667q_{i-2} - 0.3333q_{i-1} - 0.6667q_i + 0.0833q_{i+1} + 0.2500q_{i+2})/h^2$	$0.9688q_{xx}$



(a) Regular grid (b) Irregular grid

Fig. 7. 1D regular and irregular mesh. In the regular mesh (a), all cells have size h; in the irregular mesh (b) the two leftmost cells have size $\frac{h}{2}$.

are the same). On the irregular grid, the results are very different. The formally second-order LS3 scheme produces indeed a second-order accurate scheme. The LS2 scheme is first-order accurate, as can be expected. The Gauss scheme, that is a succession of two first-order accurate gradient evaluations, produces a truncation error of order zero.

Similar results are found when we compute the second derivatives of a known function ($f(\mathbf{x}) = \sin x \sin y \sin z$) on a realistic ship grid, using the least-squares and Gauss methods as implemented in ISIS-CFD (Fig. 8). The error for the LS3 scheme is smooth and relatively small. The LS2 scheme produces a larger error, that is concentrated near the jumps in cell size. The error for the Gauss

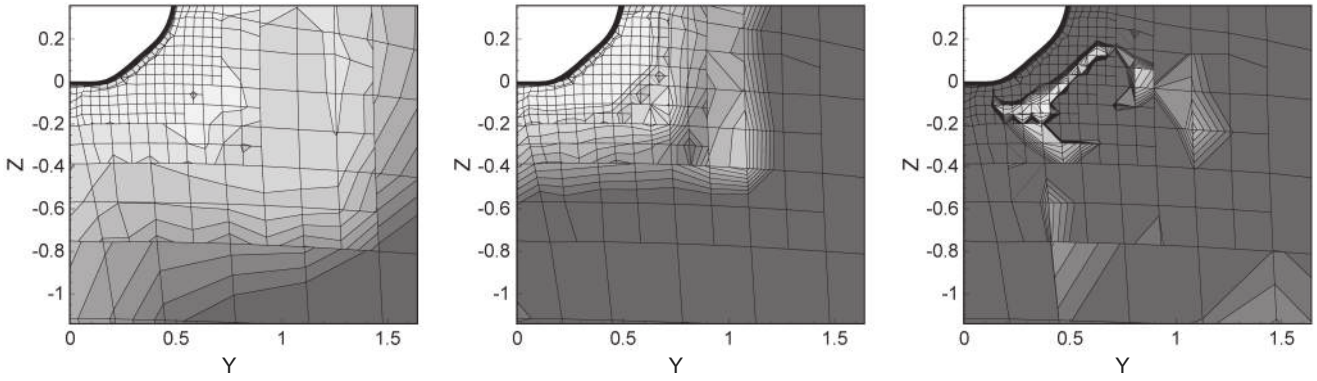


Fig. 8. Error in the computed Hessian of a known function (Frobenius norm), for the LS3 scheme (left), LS2 scheme (centre) and Gauss scheme (right). The grid is the original grid for the KVLCC2 test (Section 6.2), the figures show 10 isolines with a maximum of 0.05, in a cross-section at $x = 1$.

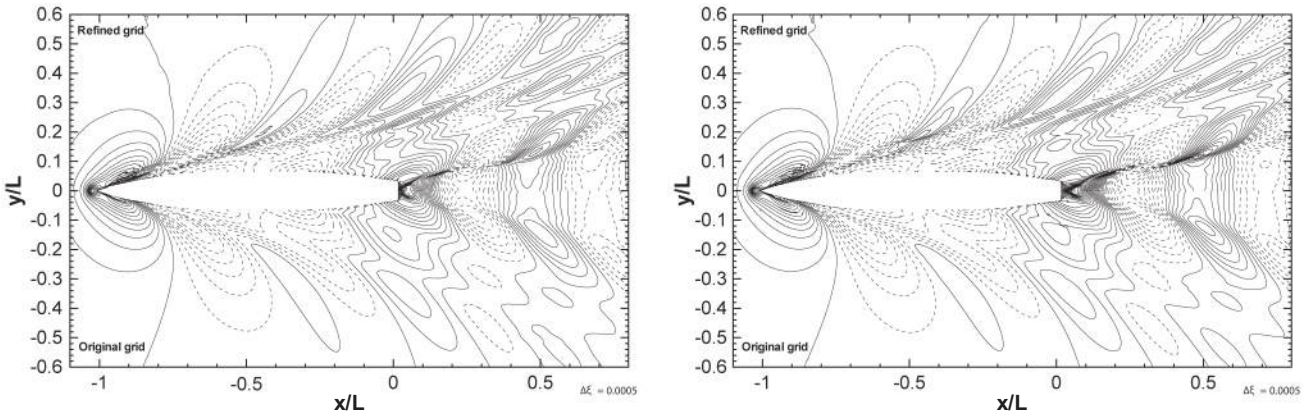


Fig. 9. Improvement in solution quality on a refined grid, compared with the non-refined original grid, for the Virtue Container ship at model scale. Model scale (left) and full scale (right).

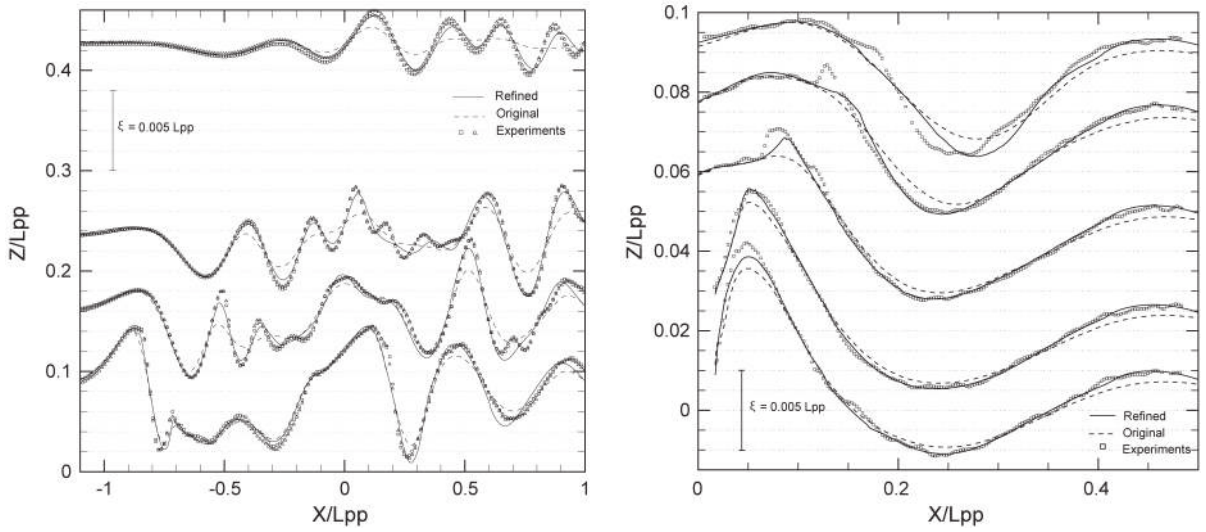


Fig. 10. Improvement in solution quality for the Virtue Container ship: comparison of computations at model scale on refined and non-refined grids with experiments. *y* Cross-sections of the free surface are compared at the side of the ship (left) and behind the ship (right).

scheme is unacceptably large. Thus, to accurately evaluate second derivatives on our grids, the LS3 method is the only acceptable method.

6. Applications

The complexity of ship flow simulation puts very strong requirements on the performance of an automatic grid refinement method. In this section, the different aspects of grid refinement for such flows are illustrated with four examples. The computation of the steady flow around a modern container ship (Subsection 6.1) shows the ability of our free-surface criterion to accurately capture large and small waves. In Subsection 6.2, the Hessian-based criterion is applied to the steady flow around the KVLCC2 tanker. The impact of a prism on a water surface (Subsection 6.3), an unsteady case with rapid interface movement, is used to illustrate the different types of buffer layers. Finally, the simulation of the DTMB 5512 combatant in head waves (Subsection 6.4) shows the effectiveness of our refinement procedure when combined with unsteady flow, ship motion, and mesh deformation.

6.1. Virtue Container ship

The first test case is a performance evaluation of the free-surface criterion from Section 5.1. The interest of this case, the steady flow around a typical modern container ship with a bulbous bow, is the simulation of the water surface and the waves generated by the ship. This particular ship generates a complex wave pattern,

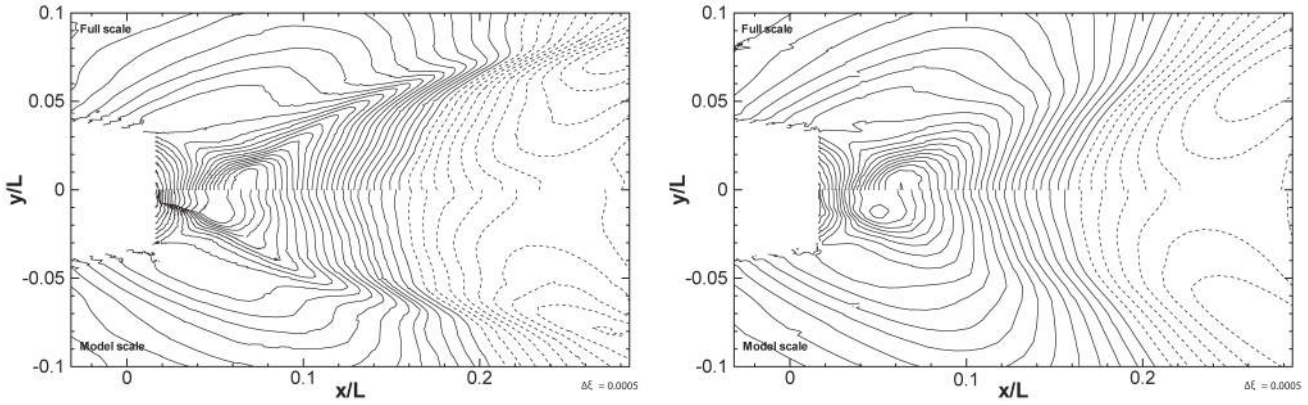


Fig. 11. Scale effects in the breaking wave at the stern, for the Virtue Container ship. The topology difference in the model- and full-scale flows on refined grids (left) cannot be observed on the original grids (right).

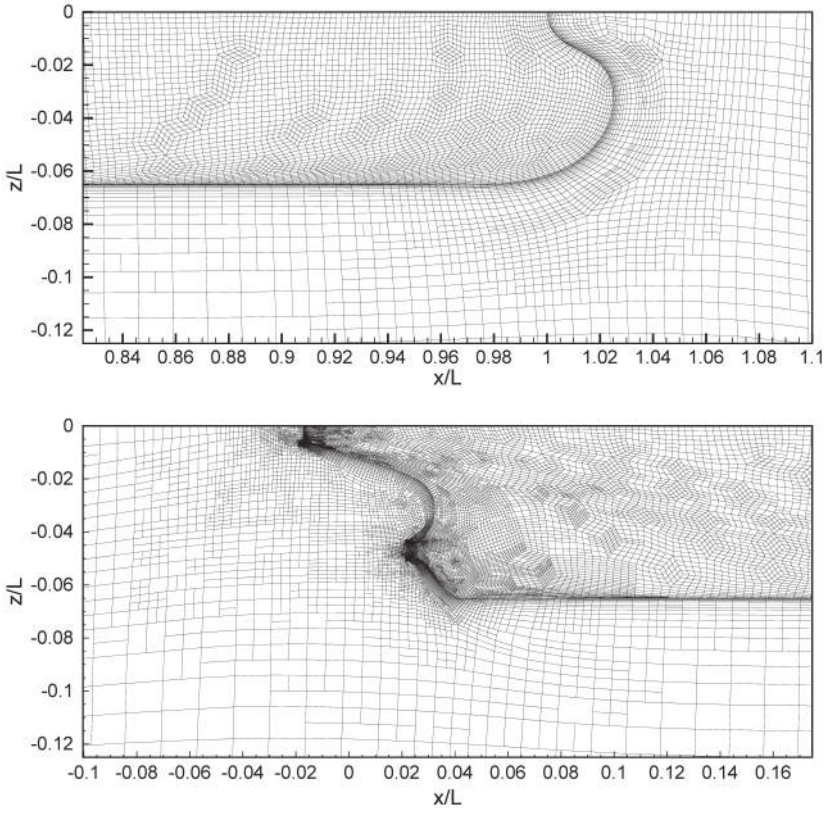


Fig. 12. Surface grid on the hull and in the y-symmetry plane for the KVLCC2 with $T_r = 0.04$ and $G_{\max} = 2$. Bow region (top) and stern region (bottom).

consisting of sharp breaking bow and stern waves and many small, secondary waves. Thus, good grid resolution near the water surface is essential.

The ship is the Virtue Container ship, a test case in the European project VIRTUE (VIRtual Tank Utility in Europe) in which ECN participated. Model tests for this ship have been performed by the Hamburg Ship Model Basin HSVA [35]. The computations are performed at $Fr = 0.272$; both the model scale $Re = 1.84 \times 10^7$ and the full ship scale $Re = 2.89 \times 10^9$ are computed.

The computations with refinement are started from the converged solutions on original grids that have a vertical grid spacing of $\Delta z = L/1000$ (where L is the ship length). The target for refinement is chosen twice finer: $T_r = L/2000$. Two standard buffer layers and one fractional layer are used; this is sufficient for steady flow. Refinement is performed every 50 time steps, the computations are continued until the global forces converge, which takes about 1000 time steps.

Results on the adaptively refined grids are compared with the results on the unrefined original grids. The results in Fig. 9 show the great increase in solution accuracy obtained with refined grids. The bow waves are higher and more sharply defined; the bow wave pattern suffers less from numerical diffusion as it moves away from the ship. The strong breaking stern wave systems are resolved in much greater detail near the ship hull and, like the bow wave, they are damped out less as they move out. Between the bow and stern waves, the representation of small flow details has improved.

A comparison of the model scale computation with the HSVA experiments is given in Fig. 10. This figure confirms the better resolution of the larger waves and the presence of more small wave details. In most places, the experiments are reproduced well on the refined grids; even in those places where differences remain, the shape of the waves is reproduced better. At the stern, the breaking of the wave system is resolved notably better, which improves the entire wave fields behind the ship.

The possibility to compute full-scale flows is one of the major advantages of CFD for ship flow computations. Among others, model- and full-scale computations can be compared to evaluate scale effects and give a better interpretation of model tests. Due to the greater resolution of the solutions on refined grids, scale effects can be observed in more detail on these grids. An example of a scale effect can be seen in Fig. 11, that shows a detail of the stern wave. On the refined grids, a small breaking wave is observed just behind the stern for the model scale computation; this wave disappears at full scale. On the original grids, it is impossible to detect this scale effect.

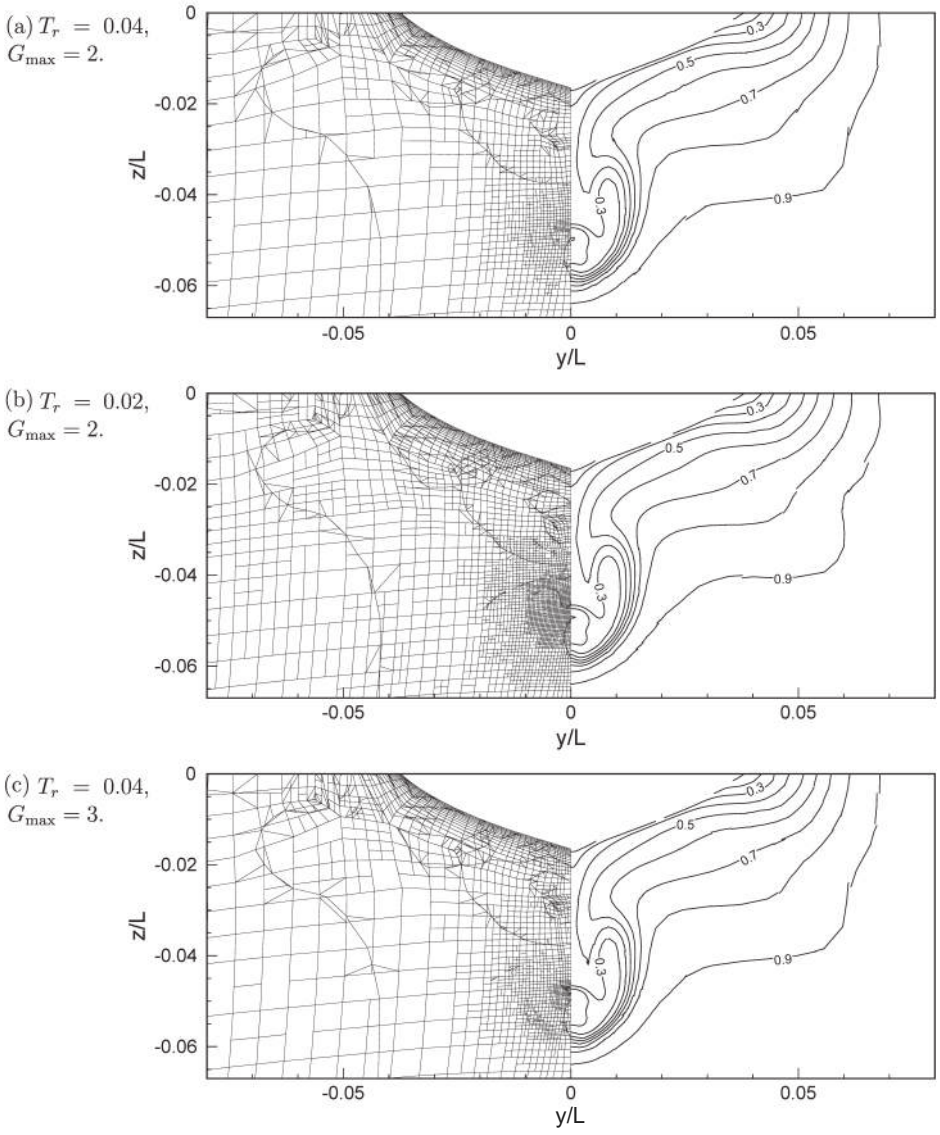


Fig. 13. Mesh and axial velocity in the propeller plane $x/L = 0.0175$, for the KVLCC2.

The grid refinement requires only a small increase in the number of cells. For the model scale computation, the refined half-body grid has 3.88 M cells and the original grid 3.07 M cells: an increase of 26%. For the full scale computation with its further developed boundary layers, the refined grid has 4.92 M cells and the original grid 3.87 M cells, which represents an increase of 27%. This good efficiency is mainly due to the use of directional refinement, which greatly limits the number of refined cells needed far away from the ship.

6.2. KVLCC2

To investigate the behaviour of the Hessian criterion from Section 5.2, the double-model flow around the KVLCC2 tanker is computed. For grid refinement, this is a particularly hard case, as there are no dominating flow features (like a free surface, or flow singularities) that clearly demand refinement; to get good overall accuracy, grid refinement has to be applied correctly all around the ship hull. The interest of this case is the computation of the drag and the vortex-induced flow in the propeller plane. The grid refinement for this case is controlled with the threshold T_r and by limiting the maximum number of refinements for each cell (called "generations") $G_{\max}$. Our goal for this test case is to show that the solution depends smoothly on these parameters and that it converges when T_r is lowered and $G_{\max}$ increased. Also, we study the optimal choice for these parameters.

The KVLCC2 is simulated in towed condition (no propeller) at model scale, $Re = 6.4 \times 10^6$ (see [36] for conditions and experimental results). Computations with refinement are based on a very coarse original grid of 45 k cells, without locally refined zones (except for some refinement around the cut-off propeller hub). The EASM turbulence model [15,13] is used to get good resolution of longitudinal vortex structures. For this steady case with a smooth criterion, no buffer layers are used. As we do not want the refinement to react directly to the boundary layers, the criterion is based on the pressure which is nearly constant over the boundary layer thickness.

Fig. 12 shows the refined surface grid with $T_r = 0.04$ and $G_{\max} = 2$. As seen in this figure, the refinement is concentrated in the regions near the bow and the stern. The criterion has created both isotropic and directional refinement; the latter appears close to the bow, in front of the bow, and in the wake. Using directional refinement clearly reduces the total number of refined cells.

Fig. 13 gives the mesh and the axial velocity in the propeller plane at $x/L = 0.0175$. The 'hook'-shaped pattern of low axial velocity is due to vortex-induced roll-up of the separated flow on the ship aft body; the finest cells in this plane appear in the 'hook' region where the flow derivatives are strongest. The three images show the influence of T_r and $G_{\max}$. Decreasing the threshold for the same number of generations (figures (a) and (b)) changes the grid in the entire propeller region: many cells that were refined only once are now refined two times. Increasing the maximum number of generations for a given threshold (figures (a) and (c)) mostly changes the grid in the centre. It has the effect of giving the criterion more freedom. For $G_{\max} = 2$, the refinement is limited by the cell sizes of the original grid, especially in (b) where almost all cells are refined twice. For (c), few cells are refined three times, so the refinement is imposed completely by the criterion. Thus, the structure of the propeller disk flow can be clearly seen in the grid. The flow fields compare well with the experimental result of Fig. 14.

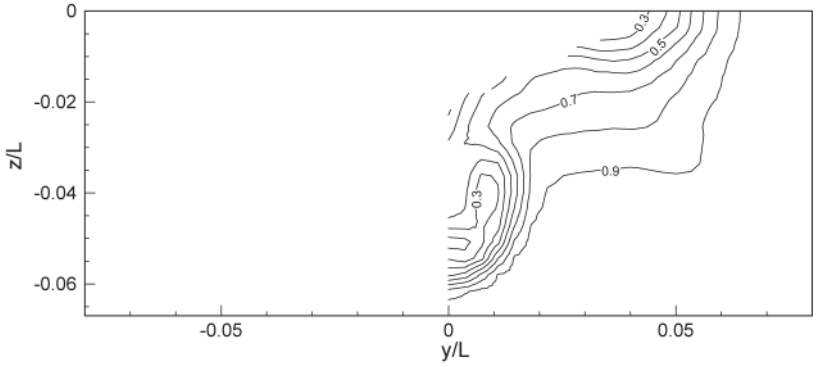
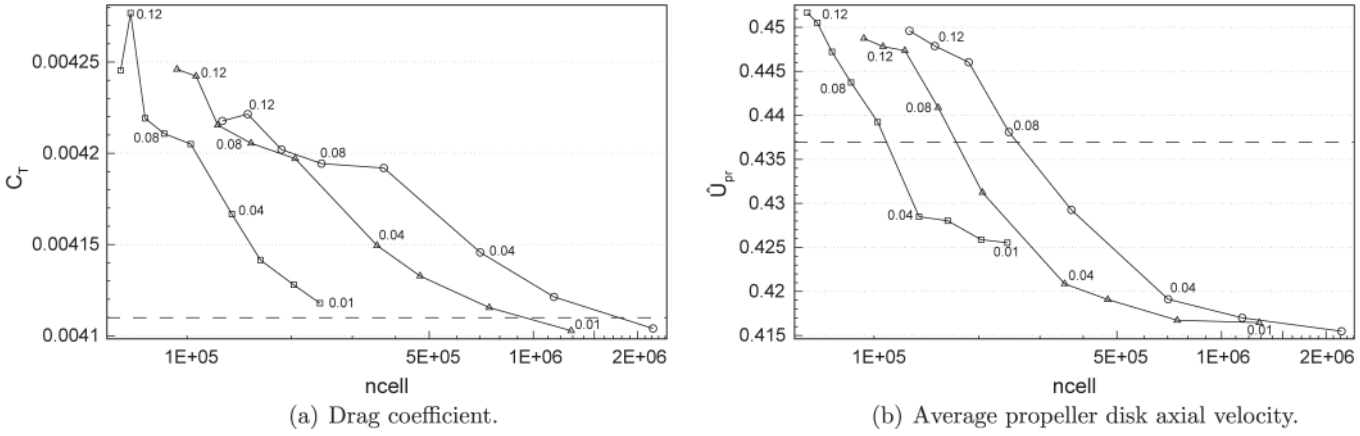


Fig. 14. Axial velocity (experimental) for the KVLCC2.



(a) Drag coefficient. (b) Average propeller disk axial velocity.

Fig. 15. Convergence of global quantities for the KVLCC2. □: $G_{\max} = 1$, △: $G_{\max} = 2$, ○: $G_{\max} = 3$. The thresholds used are T_r= 0.14, 0.12, 0.10, 0.08, 0.06, 0.04, 0.03, 0.02, and 0.01 (except for $G_{\max} = 3$). The dashed lines indicate experimental values.

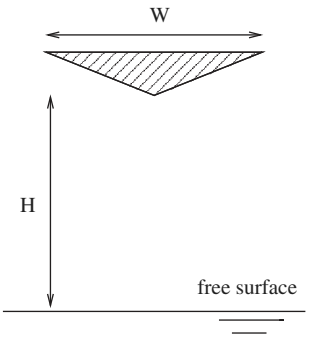


Fig. 16. Prism impacting on a free water surface: geometry and initial conditions.

Finally, we study the convergence of two global quantities (Fig. 15): the drag coefficient $C_T = F_x/\left(\frac{1}{2}\rho U^2 S\right)$, with $S = 0.2655L^2$, and $\widehat{U}_{pr}$, the average value of the axial velocity over the propeller disk lying in the $x/L = 0.0175$ plane, centred at $y/L = 0$, $z/L = -0.0469$ and with outer and inner radius $0.0154L$ and $0.0024L$. The results are encouraging: the curves are smooth, there is good correlation between the curves for different $G_{\max}$, and the curves for two and three generations converge to approximately the same values. This means that the refinement criterion is not noisy: different threshold values produce grids that form a logical sequence. The correspondence with experiments is good for C_T and reasonable for $\widehat{U}_{pr}$, the difference may be explained by small uncertainties in the exact shape of the propeller hub.

Concerning efficiency, the figures show that refinement in zones of moderate criterion values is most important for accuracy. Low $G_{\max}$ and low T_r, which cause moderate refinement in a large zone, gives much better accuracy than high $G_{\max}$ and high T_r, which only puts very fine cells in the criterion peak zones. To get sufficient convergence in the propeller disk zone at least $G_{\max} = 2$ is needed, but $G_{\max} = 3$ adds little extra accuracy for a large increase in number of cells. The best choice for T_r is the one that gives some refinement in the propeller disk, but does not refine all cells maximally.

6.3. Prism impacting on a water surface

As a third test, we present the impact of a freely falling 2D prism on a water surface. This test is based on experiments by Peterson et al. [37], earlier numerical results are reported among others by Azcueta [38] and Hay et al. [22,39]. In the present context, the interest of the case is that the 2D nature and the rapid movement of the water surface allow us to see in detail the functioning of the directional refinement (Sections 4.2, 5.1) and especially the different types of buffer layers (Section 4.3).

The setup of the case is given in Fig. 16. The prism for this test has a bottom angle of 20°, a width W and an initial height H both of 61 cm, and a mass of 50 kg/m. The water has a density of 998.4 kg/m^3, the gravity is 9.81 m/s^2. Viscosity is neglected. The only movement allowed for the prism is vertical translation.

The numerical setup of the case is as follows. The free motion of the prism is computed using Newton's laws based on the

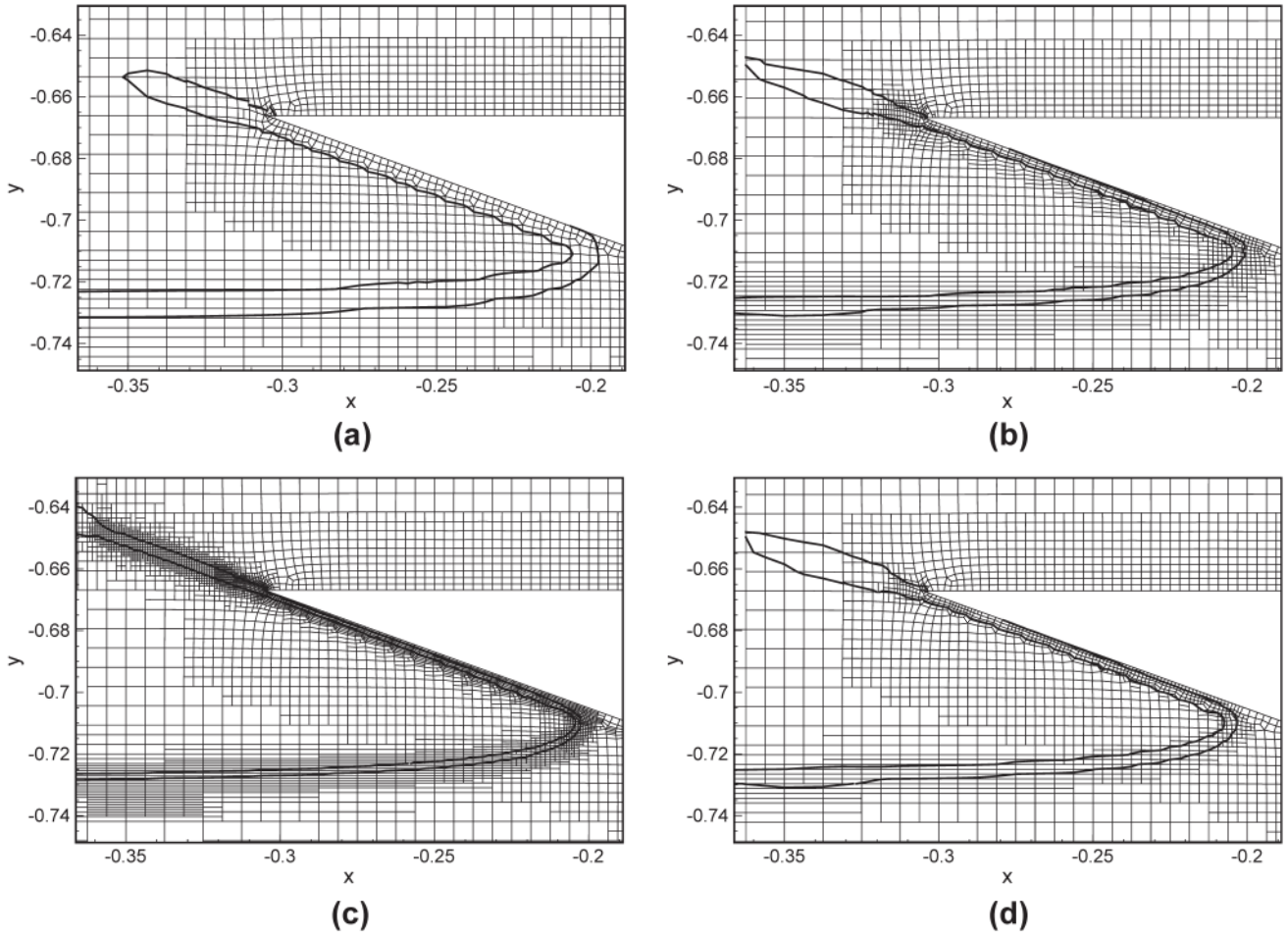


Fig. 17. Mesh and $c_i = 0.05$ and 0.95 isolines for the falling prism at $t = 0.0175s$ after initial impact. The figure is zoomed in on the left tip of the prism. The refinement threshold is $T_r = 0.0033$ (a), $T_r = 0.00165$ (b), and $T_r = 0.000825$ (c). Figure (d) represents the $T_r = 0.00165$ case with convection buffer layers.

integrated fluid forces on the prism, the motion is incorporated in the flow computation by block movement of the mesh [16]. The same original mesh is used for all computations, this mesh has some local refinement around the prism. Refinement is called every 4 time steps. The free surface criterion is used with 4 standard buffer layers (to account for the movement of the free surface through the grid) and 8 fractional layers (to provide a smooth transition to the original grid size, even in the most highly refined case). For this case, refinement is only applied to cells having $c_i = 0.3$ or more (instead of 0.1, see (13a)), in order to damp numerical instabilities coming from droplets of very low c_i; this does not affect the global behaviour of the prism, as the low-c_i zones exert only small forces on the prism.

Fig. 17a–c show the mesh and the location of the zone where c_i is between zero and one, at a given moment, for three settings of the refinement threshold T_r (Eq. (2)). Directional refinement is applied where possible, notably around the horizontal water surface and in the wall-aligned grid on the prism. The effect of full and fractional buffer layers can be clearly seen. Finally, since the refinement criterion imposes a uniform size for the grid around the surface, the cells in the fine part of the original grid are not refined as often as cells further from the prism. Note that the mesh is not refined around the tip of the water jet, due to the limitation to $c_i > 0.3$.

The solution is characterised by a very strong pressure peak that moves outward over the prism during impact. As this peak is a highly localised feature, grid refinement is an effective way of resolving it well. Fig. 18 shows the highest pressure on the prism as a function of time for the three simulations, compared with an asymptotic analytic solution by Scolan et al. [40]. As can be seen from this figure, the threshold has a very strong influence on the magnitude of the pressure peak; the agreement for the finest threshold is excellent. Also in Fig. 18, while the total number of cells increases with the reduction of the threshold, this increase is moderate, due mainly to the use of directional refinement.

Finally, as this is a case where the water surface moves rapidly, convection buffer layers can be applied (see Section 4.3.1). To see their effect, the case with $T_r = 0.00165$ is recomputed with 1 + 8 buffer layers instead of 4 + 8, but with convection layers added as well. In the grid, Fig. 17d, we can see that the refined mesh still envelops the water surface. The surface below the prism is approaching the end of the fine grid, as this snapshot was taken just before a grid adaptation step; during this step the location of the fine grid will move up and left. The main difference with the case Fig. 17b is, that fewer buffer cells are applied normal to the direction of motion of the surface (i.e. normal to the prism wall). Thus, the total number of cells decreases. However, as the convection criterion guarantees that the surface is always captured, the pressure peak is still well resolved (Fig. 19). The only disadvantage of the convection buffers is, that the thinner refined grid layer increases the chance of the solution growing unstable; therefore, the finest threshold case was not computed with convection layers. The use of standard buffer layers is a safer choice.

6.4. DTMB 5512 free in head waves

As a final test case, we compute the interaction of a wave field with a ship hull. The free ship movement is resolved, which proves

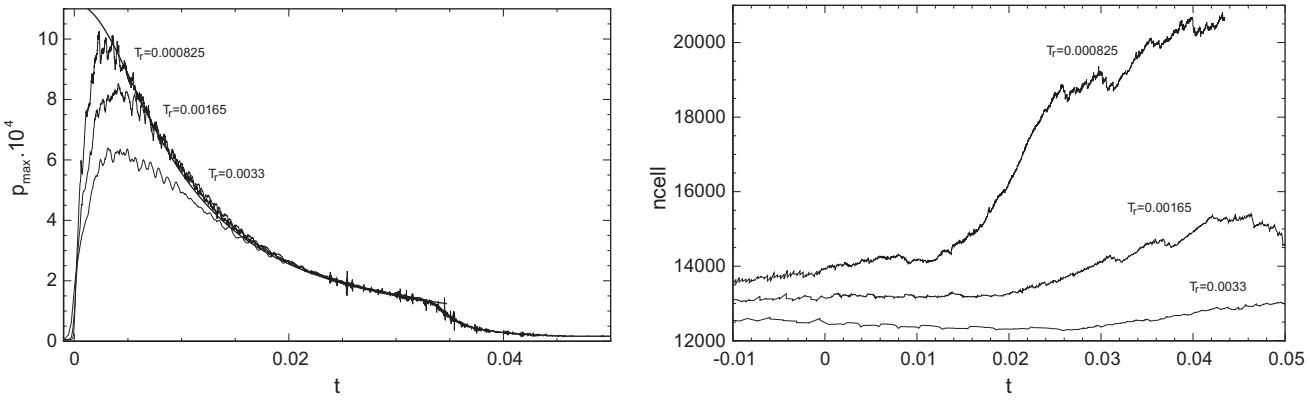


Fig. 18. Pressure peak (left) and total number of cells (right) as a function of the time after initial impact, for the computations at three settings of the refinement criterion T_r, without convection buffer layers. The pressure peak is compared with the asymptotic solution of Scolan et al. [40].

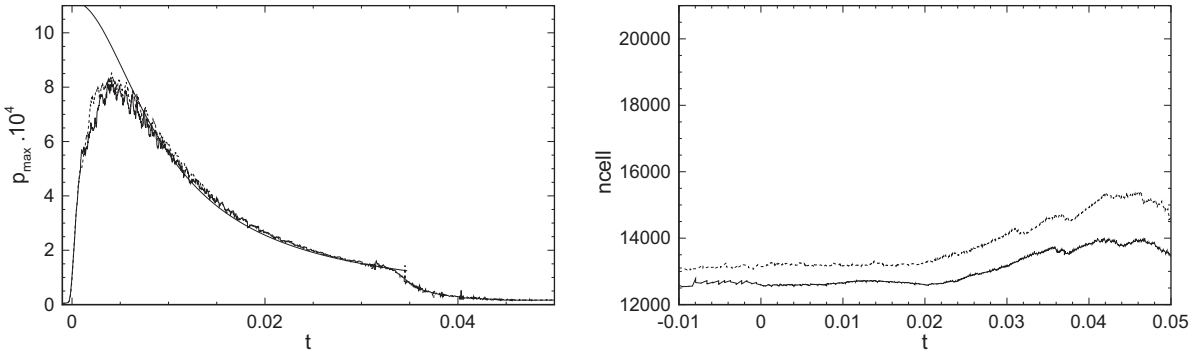


Fig. 19. Pressure peak (left) and total number of cells (right) for the case with convection buffer layers at $T_r = 0.00165$. As a comparison, the same case without convection buffer layers is given in dotted lines.

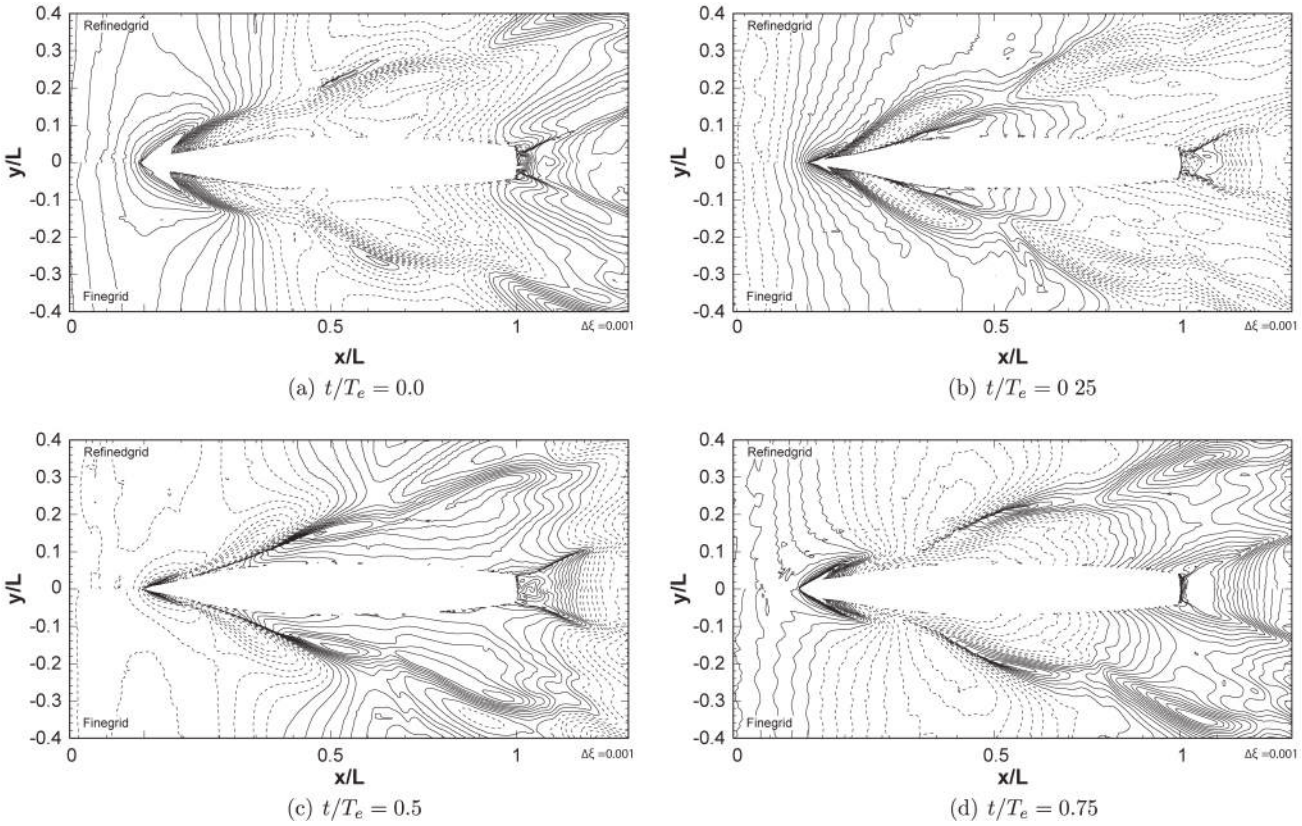


Fig. 20. Comparison of wave patterns on the refined and the fine grid for the free DTMB. The images show four different instants; T_e is the wave encounter period and $t = 0$ corresponds to a wave crest passing $x = 0$.

the successful interaction of the grid refinement method with the ship motion and mesh deformation algorithms in ISIS-CFD.

For this case, a fine grid is made with a grid spacing of $L/1000$ in z-direction at the free surface, as advised for ISIS-CFD, which corresponds to 12 cells per wave height of the incoming waves. Between the inflow boundary and the ship, the grid spacing is 50 cells per wave length in x-direction. Next to the ship, a large box of fine cells in x- and y-direction is placed around $z = 0$ to capture the diffracting waves from the hull. This fine half-body grid has 2.31 M cells. A coarse grid is made as a basis for the grid refinement, with $L/500$ in z-direction, 30 cells per wave length and 2 times coarser cells in x and y-direction in the fine box. This grid has 0.83 M cells. Automatic grid refinement is then used to get a grid spacing of $L/1000$ normal to the water surface. The refinement procedure is called every two time steps.

An important question for this case is, if the flow solution produced by the grid refinement is smooth enough not to perturb the computation of the ship motion. This motion is computed using Newton's laws for the ship hull, the mesh is adapted to the movement of the ship with the analytical weighed grid deformation of ISIS-CFD [16]. As the grid refinement mainly modifies the connectivities between cells, faces and nodes (the grid topology),

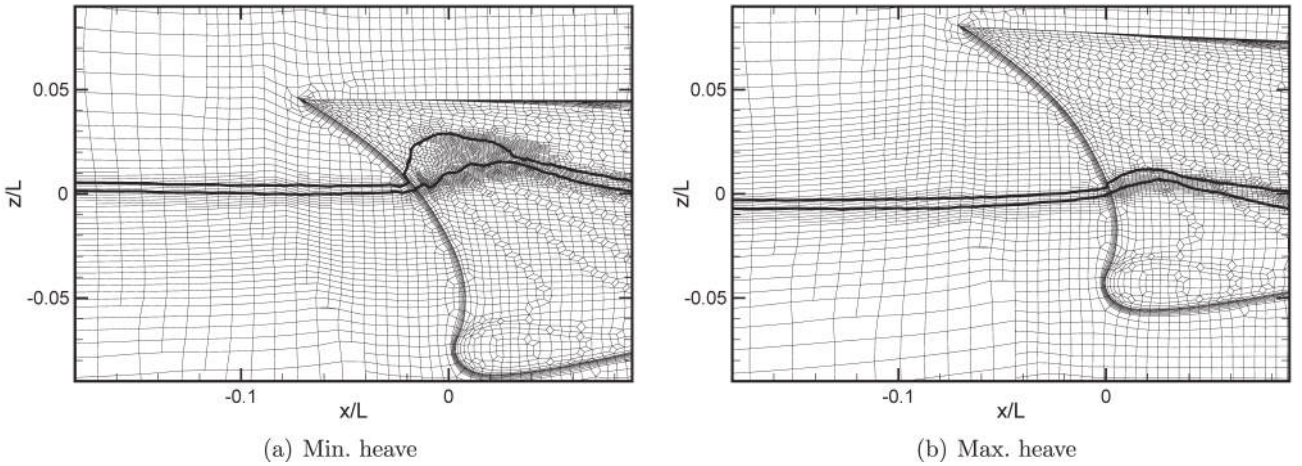


Fig. 21. Refined and deformed surface/symmetry plane mesh at the bow for the DTMB with free pitch and heave. The thick lines represent the 0.05 and 0.95 isolines of the volume fraction.

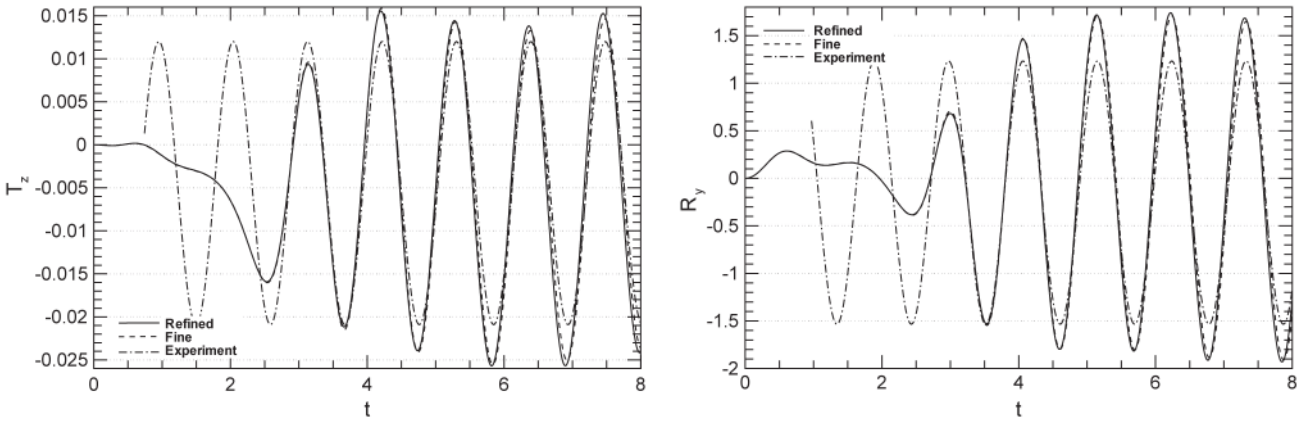


Fig. 22. Motion of the free DTMB in head waves, heave T_z and pitching angle R_y compared with experiments from [41]. $t = 0$ corresponds to the beginning of the computation.

while the grid deformation changes only the position of the nodes (the grid geometry), combining the two techniques can be done without major modifications to either.

In Fig. 20, the wave pattern is shown at four instants. The refined-grid and fine-grid solutions are nearly identical. The free ship motion creates strong wave breaking, like the secondary breaking wave around $x = 0.5$ in Fig. 20d. This wave breaking is notably better resolved on the adaptively refined grid.

Fig. 21 gives two examples of the surface grid for this case. The grid size oscillates between 1.1 M and 1.4 M cells, depending on the moment. The two images display the grid at the moments when the extreme values for the heave occur. The figure shows, how effectively the zone of refined cells is displaced and adapted to the water surface by the refinement and derefinement. Note also the examples of isotropic and directional refinement in the grid left of the bow.

The motion of the ship during the simulation is found in Fig. 22. The correspondence with the measurements from IIHR [41] is reasonable but most notably, the results for the refined-grid and fine grid case are nearly indistinguishable. The good quality of the ship motion simulation is confirmed by Fig. 23 which shows the vertical and angular acceleration of the ship, during the time when the ship accelerates from rest and the encounters with the first waves. If the refinement procedure had created any significant pressure jumps, the accelerations would have been very irregular. In fact, some small-scale roughness can be seen in a_z right after $t = 0$, but overall, the accelerations are smooth.

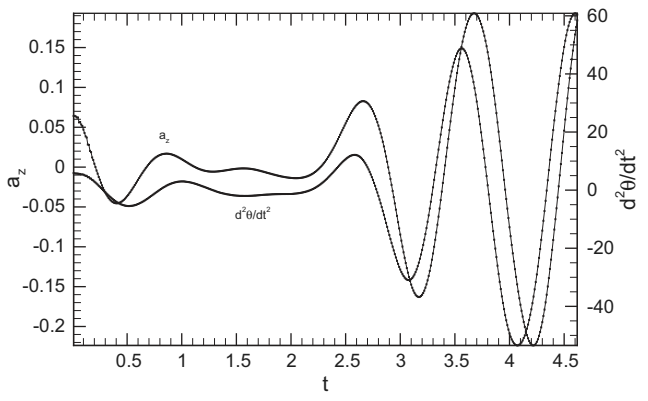


Fig. 23. Vertical and angular acceleration a_z and $\frac{d^2\theta}{dt^2}$ in the refined-grid solution for the DTMB with free pitch and heave, during the initial phase of the simulation.

So using 50–60% of the fine grid cells, the refined grid gives comparable wave fields, better resolution of wave breaking, and smooth ship motion. Thus for this free-motion case, the refinement procedure is highly effective.

7. Conclusion

We have presented a grid refinement method on unstructured hexahedral grids, for use in marine hydrodynamics simulations. The method is integrated in the ISIS-CFD finite-volume flow solver. As this solver is face-based, it accepts arbitrary grids, so locally refined grids can be used directly, without requiring modifications of the flow solver.

On unstructured hexahedral grids, where the aspect ratios of neighbouring cells vary strongly even on unrefined grids, directional grid refinement is mandatory to guarantee a smooth variation of the cell sizes in all directions. Thus, it is adopted for our method. The use of metric tensors, that specify the desired cell sizes in arbitrarily orientated orthonormal bases, is considered to be the most general and flexible way of specifying refinement criteria for directional refinement. This methodology permits the implementation of many different types of refinement criteria and the straightforward combination of several criteria. There is much freedom in the choice of the tensors, including the possibility to specify singular tensors.

Extra cells are refined around those who are indicated by the criterion, in order to allow the solution to change without having the relevant parts of the solution leave the refined grid zone. These buffer layers exist in three types: standard layers that have the same cell size as those produced by the criterion, fractional layers that give a smooth transition in cell size from coarse to fine cells, and convection layers for unsteady flow, that give refinement in the direction of the local velocity and thus anticipate the movement of the solution. These buffer layers are created by modifying the refinement criterion.

Two refinement criteria for ship flow computation are presented. The first of these is the refinement at the water surface location, in order to reduce wave dissipation and to better capture small wave details. Directional refinement normal to the surface is achieved by specifying metric tensors having only one non-zero eigenvalue, associated with an eigenvector normal to the surface. The second criterion is the classical Hessian matrix of second spatial derivatives of the solution. On unstructured hexahedral grids, it is essential to compute these tensors in a way that is formally second-order accurate on any mesh. Therefore, a least-squares method based on third-order polynomials is used.

Four test cases show, first, that the grid refinement method works together seamlessly with the other features of the ISIS-CFD flow solver: free-surface and single-fluid flow computation, resolved body motion, and mesh deformation. Second, in the test cases, significant gains are made. Either overall accuracy is increased with respect to unrefined reference grids and flow details (small waves, pressure peaks) are better resolved, or similar accuracy is obtained with much fewer cells. Thus, it is shown that for realistic, high-complexity flow computations in marine hydrodynamics, grid refinement is a successful technique.

Acknowledgements

The authors thank K. Ait Said for his contribution to the development of the Hessian criterion. Computations were performed using HPC resources from GENCI-IDRIS (Grant2010-x2010021308), which is gratefully acknowledged.

References

[1] Berger MJ, Colella P. Local adaptive mesh refinement for shock hydrodynamics. J Comput Phys 1989;82:64–84.
[2] Berger MJ, Oliger J. Adaptive mesh refinement for hyperbolic partial differential equations. J Comput Phys 1984;53:484–512.
[3] Blom JG, Verwer JG. A vectorizable adaptive grid solver for PDEs in 3D. I. Algorithmic aspects and applications. Appl Num Math 1994;16:129–56.
[4] De Zeeuw D, Powell K. An adaptively-refined Cartesian mesh solver for the Euler equations. J Comput Phys 1993;104:56–68.
[5] Maarel HTMvd. A local grid refinement method for the Euler equations. Ph.D. thesis, University of Amsterdam; 1993.
[6] Trompert RA. Local uniform grid refinement for time-dependent partial differential equations. Ph.D. thesis, University of Amsterdam; 1994.
[7] Wackers J, Koren B. A simple and efficient space-time adaptive grid technique for unsteady compressible flows. AIAA Paper 2003-3825; 2003.
[8] Bouillard P, Díez P, editors. Adaptive modelling and simulation 2009. Barcelona, Spain: CIMNE; 2009.
[9] Kroll N. Achievements of the European project ADIGMA on adaptive high-order methods for aerospace applications. In: Proceedings of ECCOMAS CFD 2010. Lisbon, Portugal; 2010.
[10] Hexpress. <http://www.numeca.be/index.php?id=29>; 10? [retrieved 30.07.10].
[11] snappyHexMesh. <http://www.openfoam.com/docs/user/snappyHexMesh.php#x26-1490005.4>; 11? [retrieved 30.07.10].
[12] Rhie CM, Chow WL. A numerical study of the turbulent flow past an isolated airfoil with trailing edge separation. AIAA J 1983;17:1525–32.
[13] Duvigneau R, Visonneau M. On the role played by turbulence closures in hull shape optimization at model and full scale. J Mar Sci Technol 2003;8:11–25.
[14] Queutey P, Visonneau M. An interface capturing method for free-surface hydrodynamic flows. Comput Fluids 2007;36(9):1481–510.
[15] Deng GB, Visonneau M. Comparison of explicit algebraic stress models and second-order turbulence closures for steady flows around ships. In: Proc. 7th Int. Conf. on numerical ship hydrodynamics. Nantes, France; 1999.
[16] Leroyer A, Visonneau M. Numerical methods for RANSE simulations of a self-propelled fish-like body. J Fluid Struct 2005;20(3):975–91.
[17] Wackers J, Ait Said K, Deng GB, Queutey P, Visonneau M, Mizine I. Adaptive grid refinement applied to RANS ship flow computation. In: 28th Symposium on naval hydrodynamics. Pasadena, California; 2010.
[18] Wackers J, Visonneau M. Adaptive grid refinement for ship flow computation. In: Bouillard P, Díez P, editors. Adaptive modelling and simulation 2009. Barcelona, Spain: CIMNE; 2009.
[19] Georgoulis E, Hall E, Houston P. Discontinuous Galerkin methods for advection–diffusion–reaction problems on anisotropically refined meshes. SIAM J Sci Comput 2007;30(1):246–71.
[20] Majewski J. Anisotropic adaptation for flow simulations in complex geometries. In: 36th CFD/ADIGMA course on hp-adaptive and hp-multigrid methods. Lecture series 2010-01. Von Karman Institute; 2009.
[21] Ganesha N, Shendea NV, Balakrishnan N. A local truncation error based adaptive framework for finite volume compressible flow solvers. Comput Fluids 2009;38(9):1799–822.
[22] Hay A. Etude de stratégies d'estimation d'erreur numérique et d'adaptation locale de maillages non-structurés pour les équations de Navier–Stokes en moyenne de Reynolds. Ph.D. thesis, Université de Nantes; 2004.
[23] Hay A, Visonneau M. Error estimation using the error transport equation for finite-volume methods and arbitrary meshes. Int J Comput Fluid Dyn 2006;20(7):463–79.
[24] Alauzet F, Loseille A. High-order sonic boom modeling based on adaptive methods. J Comput Phys 2010;229(3):561–93.
[25] Castro-Díaz MJ, Hecht F, Mohammadi B, Pironneau O. Anisotropic unstructured mesh adaptation for flow simulations. Int J Num Methods Fluids 1997;25:475–91.
[26] Fortin M, Vallet MG, Dompierre J, Bourgault Y, Habashi WG. Anisotropic mesh adaptation: theory, validation and applications. In: Proceedings of the ECCOMAS conference. Paris, France; 1996.
[27] George PL, Borouchaki H. Delaunay Triangulation and meshing – application to finite elements. Hermes; 1998.
[28] Huang W. Mathematical principles of anisotropic mesh adaptation. Commun Comput Phys 2006;1(2):276–310.
[29] Loseille A, Dervieux A, Alauzet F. Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations. J Comput Phys 2010;229:2866–97.
[30] Dam Av. Go with the flow – moving meshes and solution monitoring for compressible flow simulation. Ph.D. thesis, Utrecht University, The Netherlands; 2009.
[31] Tam A, Ait-Ali-Yahia D, Robichaud MP, Moore M, Kozel V, Habashi WG. Anisotropic mesh adaptation for 3D flows on structured and unstructured grids. Comput Methods Appl Mech Eng 2000;189:1205–30.
[32] Scherzinger WM, Dohrmann CR. A robust algorithm for finding the eigenvalues and eigenvectors of 3×3 symmetric matrices. Comput Methods Appl Mech Eng 2008;197:4007–15.
[33] D'Azevedo EF. Optimal triangular mesh generation by coordinate transformation. SIAM J Sci Stat Comput 1991;12:755–86.
[34] D'Azevedo EF, Simpson RB. On optimal triangular meshes for minimizing the gradient error. Num Math 1991;59:321–48.
[35] Schneider M, Hafermann D. Validation of free surface RANS computations with a novel optical wave cut measurement technique. In: Proceedings of PRADS 2010. Rio de Janeiro, Brazil; 2010.
[36] Larsson L, Stern F, Bertram V, editors. A workshop on numerical ship hydrodynamics. Chalmers University of Technology, Gothenburg, Sweden; 2000.
[37] Peterson R, Wyman D, Franck C. Drop tests to support water-impact and planing boat dynamics theory. Tech. Rep. TR-97, Coastal Systems Station, Panama City, USA; 1999.
[38] Azcueta R. Computation of turbulent free-surface flows around ships and floating bodies. Ph.D. thesis, University of Hamburg, Germany; 2001.
[39] Hay A, Leroyer A, Visonneau M. H-adaptive Navier–Stokes simulations of free-surface flows around moving bodies. J Mar Sci Technol 2006;11:1–18.
[40] Scolan YM, Coche E, Coudray T, Fontaine E. Etude analytique et numérique de l'impact hydrodynamique sur des carènes dissymétriques. In: Proceedings of the 7es Journées de l'Hydrodynamique. Marseille, France; 1999.
[41] Carrica PM, Wilson RV, Noack RW, Stern F. Ship motions using single-phase level set with dynamic overset grids. Comput Fluids 2007;36(9):1415–33.