Proceedings of the ASME 2015 34th International Conference on Ocean, Offshore and Arctic Engineering
OMAE2015
May 31-June 5, 2015, St. John's, New Foundland, Canada

–DRAFT–OMAE2015/42029

ADAPTIVE GRID REFINEMENT FOR FREE-SURFACE FLOW SIMULATIONS IN OFFSHORE APPLICATIONS

Peter van der Plas[*]
Computational Mechanics and Numerical Mathematics
Institute of Mathematics and Computing Science
University of Groningen
Groningen, 9747 AG
Email: p.van.der.plas@rug.nl

Arthur E.P. Veldman
Henri J.L. van der Heiden
Roel Luppes

Computational Mechanics and Numerical Mathematics
Institute of Mathematics and Computing Science
University of Groningen
Groningen, 9747 AG

ABSTRACT

In many (wave) impact problems the area of interest does not change in time and is readily pointed out by hand, allowing for a one-time design of an efficient computational grid. However, for a large number of other applications, e.g. involving violent free-surface motion or moving objects, a reasonable efficiency gain can only be obtained by means of time-adaptive refinement of the grid. In previous studies a fixed, block-based Cartesian local grid refinement method was developed and implemented in the CFD simulation tool ComFLOW [1], a VOF-based Navier-Stokes solver on Cartesian grids with cut-cell discretization of the geometry. Special attention was paid to the interface discretization in cut-cells as well as the fluid displacement algorithm across refinement boundaries. The method was successfully applied to a range of offshore applications, including for example wave-impact on a semi-submersible (figure 1)and sloshing in a moonpool. In the present paper we present the first results of our attempts to extend the method to support adaptive refinement.

[*]Corresponding author.

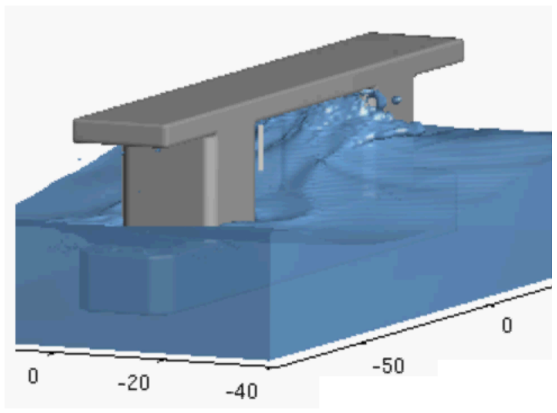


FIGURE 1: Snapshot of a simulation of free-surface flow around a semi-submersible

1 DISCRETIZATION OF THE NAVIER–STOKES EQUATIONS

An excellent model for incompressible fluid flow is provided by the Navier-Stokes equations. The set of equations consists of the continuity equation

$$\mathscr{M}\mathbf{u} = 0, \tag{1}$$

where $\mathscr{M} = \nabla\cdot$ is the divergence operator, and the momentum equation

$$\frac{\partial \mathbf{u}}{\partial t} + \mathscr{C}(\mathbf{u},\mathbf{u}) + \mathscr{G}p - \nu\mathscr{D}\mathbf{u} = \mathbf{f}, \tag{2}$$

based on the convection operator $\mathscr{C}(\mathbf{u},\mathbf{v}) = \mathbf{u}\cdot\nabla\mathbf{v}$, the pressure gradient operator $\mathscr{G} = \nabla$, the diffusion operator $\mathscr{D}(\mathbf{u}_h) = \nabla\cdot\nabla\mathbf{u}_h$ and forcing term $\mathbf{f}$. The kinematic viscosity is denoted by ν.

The continuity equation (1) is discretized at the 'new' time level $n+1$ to give

$$M\mathbf{u}_h^{n+1} = -M^\Gamma\mathbf{u}_h^{n+1} \tag{3}$$

where M acts on the internal of the domain and M^Γ acts on the boundaries of the domain.

Convection and diffusion are discretized explicitly in time (we write D for the discrete diffusion operator and $C(\mathbf{u}_h)$ for the discrete convection operator.) In line with the divergence, the pressure gradient is discretized at the new time level. For simplicity, in the discussion below, we use forward Euler time integration. If we denote the diagonal matrix containing the fluid volumes of the momentum cells by Ω, the discretized momentum equation is given by

$$\Omega\frac{\mathbf{u}_h^{n+1} - \mathbf{u}_h^n}{\Delta t} = -C(\mathbf{u}_h^n)\mathbf{u}_h^n + D\mathbf{u}_h^n - G\mathbf{p}_h^{n+1}, \tag{4}$$

where Ω is a diagonal matrix containing the fluid volumes of the momentum cells.

Finding the solution to the system of equations (3) & (4) is split in two steps. First an auxiliary variable $\mathbf{u}_h^*$ is defined by the equation

$$\Omega\frac{\mathbf{u}_h^* - \mathbf{u}_h^n}{\Delta t} = -C(\mathbf{u}_h^n)\mathbf{u}_h^n + D\mathbf{u}_h^n. \tag{5}$$

Using this variable in (4) gives

$$\Omega\frac{\mathbf{u}_h^{n+1} - \mathbf{u}_h^*}{\Delta t} = -G\mathbf{p}_h^{n+1}. \tag{6}$$

Substitution of equation (6) in the continuity equation (3), gives rise to the following system of equations

$$\Delta t M\Omega^{-1}G\mathbf{p}_h^{n+1} = M\mathbf{u}_h^* + M^\Gamma\mathbf{u}_h^{n+1}, \tag{7}$$

which is often referred to as the discrete pressure Poisson equation, as it can be viewed as a discretization of the equation $\mathscr{M}\circ\mathscr{G}\mathbf{p}_h = \mathscr{M}\mathbf{u}_h$. Note, however, that we are not directly discretizing the composed operator $\mathscr{M}\circ\mathscr{G}$ here, but its separate parts $\mathscr{M}$ and $\mathscr{G}$. Hence, of sole importance is the accuracy of the discretization of the divergence and gradient operators $\mathscr{M}$ and $\mathscr{G}$, respectively. This should be kept in mind when assessing the accuracy of the method.

Spatial discretization

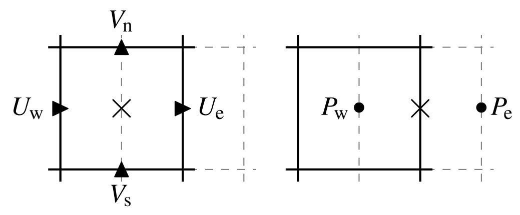


FIGURE 2: Regular discretization stencil for $M\mathbf{u}_h$ (left) and $G\mathbf{p}_h$ (right, horizontal component only) applied at the location indicated with $\times$.

The Navier–Stokes equations are discretized on an Arakawa C-grid as illustrated in figure 2. For brevity the third dimension, which is treated similarly, is omitted. The subscript ℓ is used to indicate the local refinement level, where $\ell = 0$ refers to the unrefined base grid (indexing is discussed in section 2.)

In the regular parts of the grid the divergence operator is discretized as follows (for the subscript convention consult figure 2)

$$M\mathbf{u}_h|_\ell = \Delta y_\ell(U_{e;\ell} - U_{w;\ell}) + \Delta x(V_{n;\ell} - V_{s;\ell}) \tag{8}$$

In order to let the discrete operators satisfy the adjoint condition,

$$G = -M^* \tag{9}$$

the pressure gradient is discretized as $G\mathbf{p}_h = -M^*\mathbf{p}_h$. This gives the following second-order central discretization:

$$\frac{P_{e;\ell} - P_{w;\ell}}{\Delta x_\ell} \qquad \frac{P_{n;\ell} - P_{s;\ell}}{\Delta y_\ell} \tag{10}$$

Following [2], a symmetry-preserving second-order central discretization scheme is used for convection and diffusion.

2 LOCAL GRID REFINEMENT

Figure 3 illustrates the block-based and patch-based local refinement methods that are commonly encountered in the literature.

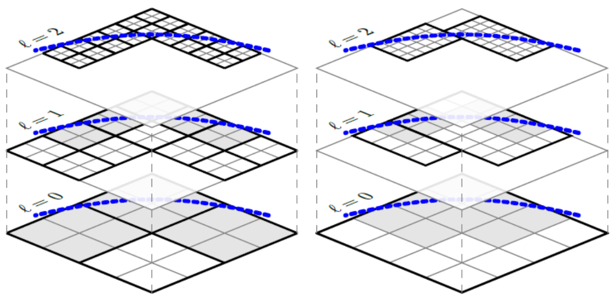


FIGURE 3: Block-based (*Left*) and patch-based (*Right*) refinement, illustrated for refinement around a curved strip.

In the present work, a block-based Cartesian local grid refinement approach is followed in which refinement and coarsening is applied block-wise. By setting the block size equal to 1×1, effectively a cell-based approach is obtained. On static grids, the blocks can be manually grouped together to form arbitrary rectangular refinement regions. On adaptive grids a block-clustering algorithm could be used to group blocks together in larger rectangular regions, but design of an efficient clustering algorithm is not a trivial task.

Grid data structure

A semi-structured approach is followed in which a cell (i,j) at refinement level ℓ is *replaced* by a set of $r_i \times r_j$ smaller cells at refinement level $\ell+1$ having indices $(2i+m, 2j+n)$ at offsets $0 \leqslant m < r_i, 0 \leqslant n < r_j$. The semi-structured indexing system is illustrated in figure 4. On block-shaped refinement regions the method is locally structured, hence the computational efficiency of the original array-based solution methods can be exploited as much as possible. Only at the boundaries of the refinement regions where the actual refinement takes place a new treatment is required.

For describing the grid layout an auxiliary array is introduced storing only one integer for each potentially occuring cell $(i,j;\ell)$ pointing at the memory location of the subgrid in which it is contained (or null if the cell does not exist). The auxiliary array has a length of

$$N\frac{1-(r_ir_j)^L}{1-r_ir_j},$$

where N is the number of cells on the base grid and L is the highest refinement level occuring on the grid. Altogether a data structure results that allows for fast and efficient look-up when compared with typical tree-based storage methods, along the lines of [3].

For block-based grids the overhead of the data structure can

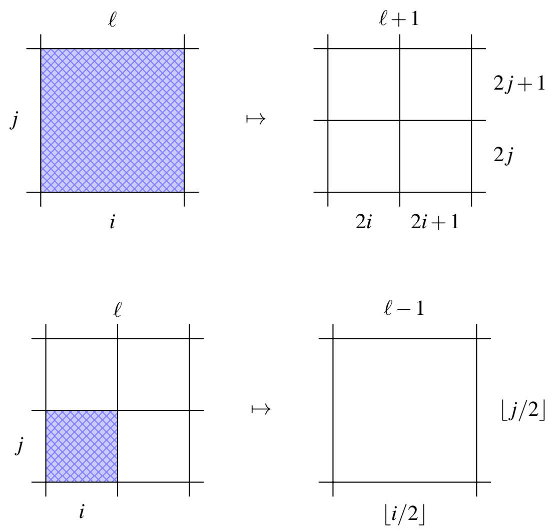


FIGURE 4: Illustration of semi-structured indexing for refinement ratios $r_i=2, r_j=2$. *Top*: From coarse to fine indices. *Bottom:* From fine to coarse indices.

be reduced by a factor of $b_i \times b_j$ and becomes

$$\frac{N}{b_ib_j}\frac{1-(r_ir_j)^L}{1-r_ir_j}$$

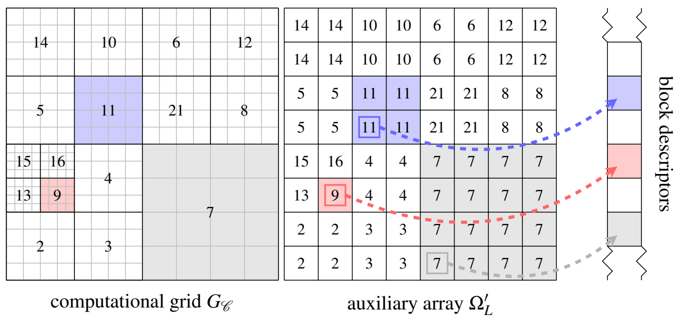
Figure 5 illustrates the auxiliary array storage method for a simple locally refined grid. The computational grid consist of 256 grid cells, each storing approximately 100 computational variables. The overhead of the auxiliary array consists of only 64 integers.

The computational variables as well as other information about the grid blocks are stored in a list of block descriptors of which the location can be looked up in the auxiliary array. If in the case of adaptive grid refinement a grid block is marked for removal, its memory space is not immediately deallocated in order to facilitate reuse by new grid blocks.

Spatial discretization at interfaces

As an example we consider refinement interfaces in the "x=constant"-plane where the refined cells are located to the left of the interface. Five other interface orientations are possible, which are treated similarly. To further simplify discussion we assume a base grid with uniform grid spacings Δx_0 and Δy_0.

In the current discussion we take refinement ratios $r_i=1$ and $r_j=2$. In words, no refinement is applied perpendicular to the refinement interface and local refinement is only applied along the refinement interface. For the grid spacings this implies $\Delta y_{\ell+1} = \Delta y_\ell/2$ and $\Delta x_{\ell+1} = \Delta x_\ell$. Since $r_i=1$, for the latter, the level subscript will be omitted.



computational grid $G_{\mathscr{C}}$
auxiliary array Ω'_L
block descriptors

FIGURE 5: The auxiliary array data structure illustrated for a 4×4-block-based grid. The actual grid variables (field variables, labels, etc.) are stored in the block descriptors, shown to the right.

Extending the discretization to the three-dimensional case, non-uniform grids and other refinement directions or refinement ratios is straightforward.

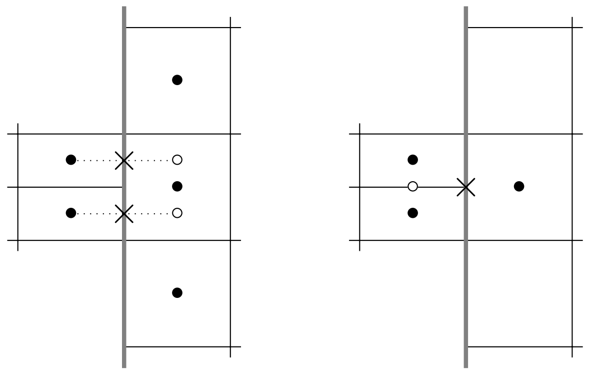


FIGURE 6: *Left*: refinement approach with interpolation of missing pressure variables, *Right*: refinement approach (as followed here) with shifted pressure gradient which is equal for both refined cell faces.

Typically, a large stencil is used for the approximation of missing pressure or velocity variables along the refinement interface. Interpolation of missing variables increases the number of non-zero coefficients in the pressure Poisson matrix, which might result in a non-symmetric matrix, putting higher demands on the solver. Most authors use a non-overlapping interface and apply linear (or even higher-order) interpolation for missing variables on the other side of the interface [4]. Another approach is to apply linear interpolation inside an overlapping interface [5]. In all cases the discretization results in a non-symmetric system of equations.

In the present approach, a compact discretization scheme is designed (in both space and time), which results in a small and symmetric scheme for the discrete composition of M and G. This makes it possible to employ an efficient linear solver. Furthermore, this facilitates the use of adjacent refinement regions as well as the interface discretization near objects and free-surface boundaries.

Divergence and pressure gradient There are two ways of obtaining a first-order accurate discretization of the pressure gradient. Either by using a linear interpolation for the missing pressure variable outside the refinement region (see left of figure 6) or by slightly shifting the location of the pressure gradient (see right of figure 6). Both approaches result in a first-order accurate discretization scheme introducing an error term that is proportional to respectively $\Delta\frac{\partial^2 p}{\partial y^2}$ and $\Delta\frac{\partial^2 p}{\partial y \partial x}$, where for brevity *we use* Δ *to denote the order of magnitude, omitting any subscripts or products of grid spacings.* However, the first approach results in a relatively large stencil whereas the second approach uses a smaller interpolation stencil consisting of pressure variables that already form part of the regular stencil.

For this reason, the second approach is followed, which can be described as "using a constant pressure gradient along a refined cell face" (see e.g. [6, 7]). Correspondingly, we use a uniform velocity across the entire refined cell face and only place coarse computational velocity variables at the interface.

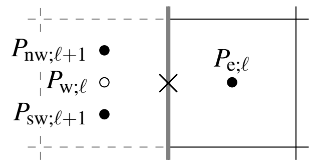


FIGURE 7: Missing variable (○) for the gradient operator applied at the location indicated with ×, together with the variables used for linear interpolation (●).

For the missing coarse pressure variable $P_{\text{w};\ell}$ (see figure 7) a simple average of the neighbouring fine pressure values is used. This results in the following discretization of the pressure derivative at the refined cell face

$$\frac{2P_{\text{e};\ell} - P_{\text{sw};\ell+1} - P_{\text{nw};\ell+1}}{2\Delta x} = \frac{\partial p}{\partial x}(\mathbf{x}^u_\ell) + O(\Delta).$$

Note that the approximation for the missing pressure variable is second-order accurate, but one order of accuracy is lost due to the loss of symmetry in the central scheme. Hence the approximation of the pressure gradient across the refinement interface is first-order accurate.

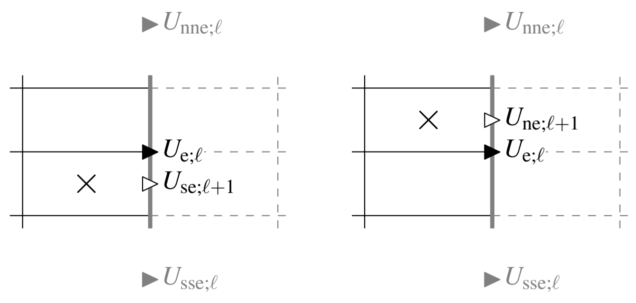


FIGURE 8: Missing variables (▷) for the divergence operator applied at the location indicated with ×, together with the variable used for constant extrapolation (▶) and linear correction (▶).

For the discretization of the divergence at the fine side of refinement interfaces, an approximation is needed for the missing fine velocities. As a first approach, a discretization is obtained by means of the adjointness condition (9), so the divergence operator is defined as the negative transpose of the gradient operator. This implies that missing velocities are simply approximated using constant extrapolation (see figure 8)

$$U_{\mathrm{se};\ell+1} := U_{\mathrm{e};\ell} = u(\mathbf{x}^U_{\mathrm{se};\ell+1}) + O(\Delta),$$
$$U_{\mathrm{ne};\ell+1} := U_{\mathrm{e};\ell} = u(\mathbf{x}^U_{\mathrm{ne};\ell+1}) + O(\Delta),$$

The corresponding divergence operator for interface cells is then given by

$$\left(\overline{M}\mathbf{u}_h\right)_{\mathrm{s};\ell+1} = \Delta y_{\ell+1}\left(U_{\mathrm{e};\ell} - U_{\mathrm{sw};\ell+1}\right) + \Delta x\left(V_{\mathrm{c};\ell+1} - V_{\mathrm{s};\ell+1}\right)$$
$$\left(\overline{M}\mathbf{u}_h\right)_{\mathrm{n};\ell+1} = \Delta y_{\ell+1}\left(U_{\mathrm{e};\ell} - U_{\mathrm{nw};\ell+1}\right) + \Delta x\left(V_{\mathrm{n};\ell+1} - V_{\mathrm{c};\ell+1}\right)$$

Taking a uniform velocity along the refined cell face is conform the earlier remark of using a uniform pressure gradient along the refined cell face. However, it can be seen that the scheme for the divergence operator is not consistent yet because the velocities in the central difference are not well aligned.

$$\begin{aligned}\frac{1}{\Delta x \Delta y_{\ell+1}}\left(\overline{M}\mathbf{u}_h\right)_{\mathrm{s};\ell+1} &= \frac{\partial u}{\partial x}(\mathbf{x}^P_{\mathrm{s};\ell+1}) + \frac{\partial v}{\partial y}(\mathbf{x}^p_{\mathrm{s};\ell+1}) \\ &\quad + \frac{1}{2}\frac{\Delta y_{\ell+1}}{\Delta x}\frac{\partial u}{\partial y}(\mathbf{x}^u_{\mathrm{e};\ell}) + O(\Delta) \\ \frac{1}{\Delta x \Delta y_{\ell+1}}\left(\overline{M}\mathbf{u}_h\right)_{\mathrm{n};\ell+1} &= \frac{\partial u}{\partial x}(\mathbf{x}^P_{\mathrm{n};\ell+1}) + \frac{\partial v}{\partial y}(\mathbf{x}^p_{\mathrm{n};\ell+1}) \\ &\quad - \frac{1}{2}\frac{\Delta y_{\ell+1}}{\Delta x}\frac{\partial u}{\partial y}(\mathbf{x}^u_{\mathrm{e};\ell}) + O(\Delta)\end{aligned}$$

The inconsistency can be resolved by adding corrections for the observed error terms. In order to satisfy mass conservation it is important that these corrections sum up to zero for each refined cell face. For this we can make use of the symmetry observed in the above error terms and correct the operator $\overline{M}$ with the following linear correction terms:

$$\frac{1}{\Delta x \Delta y_{\ell+1}}\left(M^+\mathbf{u}_h\right)_{\mathrm{s};\ell+1} = -\frac{1}{2}\frac{\Delta y_{\ell+1}}{\Delta x}\left[\delta_y \mathbf{u}_h\right]_{\mathrm{e};\ell} \qquad (12)$$
$$\frac{1}{\Delta x \Delta y_{\ell+1}}\left(M^+\mathbf{u}_h\right)_{\mathrm{n};\ell+1} = \frac{1}{2}\frac{\Delta y_{\ell+1}}{\Delta x}\left[\delta_y \mathbf{u}_h\right]_{\mathrm{e};\ell} \qquad (13)$$

where δ_y is a central differencing operator which is applied along the refinement interface (see figure 8)

$$\left[\delta_y \mathbf{u}_h\right]_{\mathrm{e};\ell} = \frac{U_{\mathrm{nne};\ell} - U_{\mathrm{sse};\ell}}{2\Delta y_\ell} \qquad (14)$$

The divergence operator with correction, i.e. $\overline{M} + M^+$ is now first-order accurate.

To conclude we remark that the correction operator M^+ is similar for other interface orientations. Note that in the three-dimensional case the operator would also include a difference term in the secondary direction tangential to the interface.

Convection and diffusion Missing velocities that are needed in the convection and diffusion scheme are approximated using (bi-)linear interpolation and a quadratic correction is added whenever sufficient information is available. Due to this interpolation, the central discretization scheme for convection remains second-order accurate since plugging in an interpolation error ε of order $O(\Delta^3)$ in the numerator of a single differencing scheme results in a discretization error of order $\varepsilon/\Delta = O(\Delta^2)$. The central discretization scheme for diffusion becomes first-order accurate since plugging in an interpolation error ε of order $O(\Delta^3)$ in the numerator of a double differencing scheme results in a discretization error of order $\varepsilon/\Delta^2 = O(\Delta)$.

Locally refined Volume-of-Fluid (VOF) scheme The fluid distribution is discretized by means of cell-wise volume fraction which are advected by a second-order advection scheme. The fluid displacement in ComFLOW is modeled by means of a second-order Volume-of-Fluid advection scheme. A sharp free-surface interface is reconstructed from the volume fractions F by a piecewise-linear interface reconstruction (PLIC) [8].

At refinement interfaces missing volume fractions on the coarse grid are approximated by averaging the refined volume fractions (left part of figure 9) and on the fine grid they are reconstructed geometrically from the volume fractions on the coarse grid (right part of figure 9). The advective fluxes located at the refinement interfaces are calculated on the fine grid and simply added up to obtain the advective fluxes on the coarse grid.

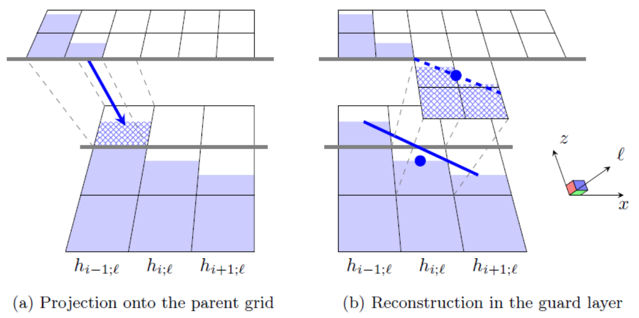


FIGURE 9: Reconstruction of missing volume fractions needed by the VOF advection scheme

Automatic and time-adaptive refinement

In applications where the resolution requirements change not only in space, but also in time we have to resort to an automatic grid adaptation algorithm. Even on static grids an automatic grid adaptation algorithm might prove to be a useful tool, for example for automatic grid refinement in the presence of complex geometries. Cartesian cell- and block-based refinement methods allow for a straightforward implementation of a refinement criterion. If a cell-based criterion $\phi(i,j,k;\ell)$ is larger than some threshold $\beta(i,j,k;\ell)$ it is marked for refinement and if it is smaller than some other threshold $\alpha(i,j,k;\ell) < \beta$ it is marked for coarsening. In the block-based approach, a block $(I,J,K;\ell)$ is refined if one or more of its cells are marked for refinement and it is coarsened only if *all* of its cells are marked for coarsening, in other words, the block-based refinement criterion is $\Phi(I,J,K;\ell) = \max\phi$, where the maximum is taken over all cells in block $(I,J,K;\ell)$. In order to avoid endless (recursive) refinement or coarsening the operation is restricted to refinement levels 0 up to some maximum L. If a grid block is refined, it is checked that neighbouring blocks differ by not more than one refinement level in order to satisfy the condition of graded refinement. Any neighbouring block that does not yet satisfy this condition because it is too coarse is refined as well (recursively). Similarly a grid block that is marked for coarsening is only actually coarsened if it does not destroy the property of graded refinement.

The thresholds α and β are typically some constants, but could also depend on the actual cell location and/or refinement level. Any refinement criterion ϕ can be easily plugged in. (In the section "Numerical results" some examples are given.)

Because of the explicit time integration scheme that is used for convection and diffusion, a CFL$\leqslant 1$ criterion has to be satisfied implying that flow characteristics can travel no more than one cell per time step. In order to maintain sufficient grid resolution the grid is adapted every time step. Fortunately, adapting the grid takes only a few percent of the computational time. Especially on large grids, the bulk of the simulation time is consumed in the Poisson solver. The main overhead of the block-based refinement method is located in the exchange of variables at the block interfaces. Block clustering could be used to reduce the number of interfaces, however, the relative overhead quickly becomes smaller for larger grids or larger block sizes.

3 NUMERICAL RESULTS

Altogether the interface treatment is second-order accurate, apart from:

- the divergence operator (M),
- the pressure gradient operator (G),
- the diffusion operator (D),
- the VOF scheme,

which are all first-order accurate. It is hard to make the divergence and pressure gradient operators both mass conservative and higher-order accurate (as is also observed in [4]). We note that in smooth regions of the solutions this does not pose any problem. Although the interface treatment is partly first-order accurate, the interfaces cover a subset of lower dimensionality ("lines vs. surface" in 2-D simulations or "surfaces vs. volume" in 3-D simulations). Typically a global convergence rate is observed that ranges between $O(1.5)$ and $O(2.0)$.

A manufactured solution

To assess the convergence rate of a simulation on a locally refined grid we resort to the method of prescribed solution forcing. The numerical method is tested for the approximation of the solution

$$u = -\cos X \sin Y, v \quad = \sin X \cos Y, p = \cos X \cos Y, \tag{15}$$

where $X = 2\pi x, Y = 2\pi y$. Appropriate forcing terms,

$$f_x/2\pi = -\sin X - \sin X \cos X - 4\pi\mu \cos X \sin Y,$$
$$f_y/2\pi = -\sin Y - \sin Y \cos Y + 4\pi\mu \cos X \sin Y,$$

are added to the right hand side of the momentum equations. The simulation domain is truncated to $[\frac{1}{4},\frac{3}{4}] \times [\frac{1}{4},\frac{3}{4}]$ by means of symmetry boundary conditions. A refinement zone is placed in the region $[\frac{2}{5},\frac{3}{5}] \times [\frac{2}{5},\frac{3}{5}]$ as shown in figure 10.

The numerical results listed in table 1 show a second-order convergence rate for the velocity field and a near to second-order global convergence rate for the pressure. The first-order accuracy of the refinement interfaces is only reflected directly in the local convergence rate of the pressure which quickly becomes of first-order in the number of grid cells.

Flow around a square cylinder (Re=10)

In order to investigate the performance of the local grid refinement scheme, two-dimensional simulations of flow around

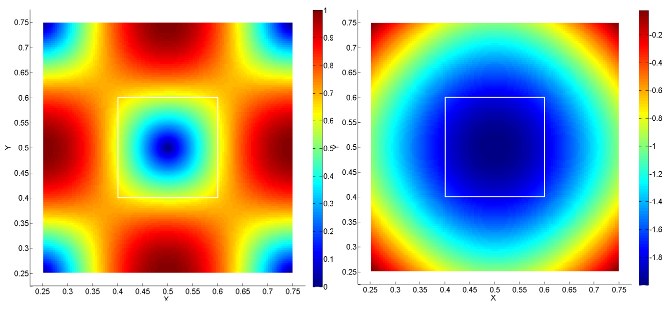


FIGURE 10: Manufactured solution (15). Left: absolute velocity, Right: pressure

h	$\|\varepsilon_u\|_2$	order	$\|\varepsilon_u\|_\infty$	order	$\|\varepsilon_p\|_2$	order	$\|\varepsilon_p\|_\infty$	order
2.50e-02	3.46e-03	-	5.51e-03	-	4.82e-03	-	8.57e-03	-
1.25e-02	8.56e-04	2.01	1.39e-03	1.99	1.19e-03	2.02	2.33e-03	1.88
6.25e-03	2.15e-04	1.99	3.49e-04	1.99	3.02e-04	1.98	8.68e-04	1.43
4.17e-03	9.59e-05	1.99	1.56e-04	1.99	1.39e-04	1.92	5.65e-04	1.06
3.12e-03	5.40e-05	1.99	8.76e-05	1.99	8.06e-05	1.89	4.19e-04	1.04
2.08e-03	2.41e-05	2.00	3.90e-05	2.00	3.78e-05	1.87	2.76e-04	1.03
1.56e-03	1.35e-05	2.00	2.20e-05	1.99	2.22e-05	1.85	2.06e-04	1.02

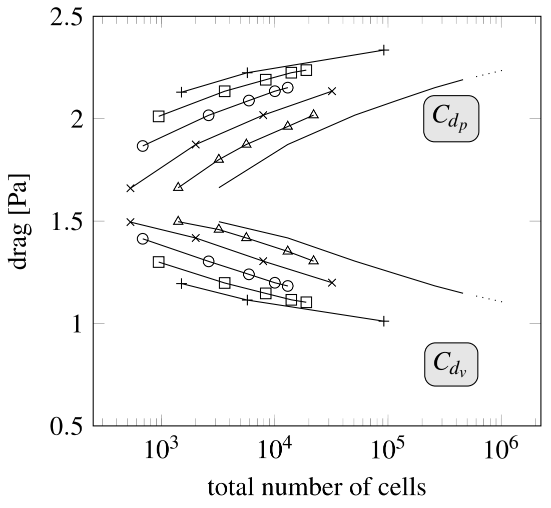
TABLE 1: Convergence behavior for the 2-D manufactured solution (15) on a locally refined grid. The grid spacing h is measured in the refined region of the domain. $\varepsilon_u, \varepsilon_p$ denote the approximation error in the velocity and pressure respectively. The global and local approximation errors are measured with the 2-norm and infinity norm respectively.

a square cylinder have been performed. All simulations at Reynolds numbers 10 and 100 have been performed using a second-order central discretization.

In order to get a good view of the efficiency gain that is obtained with the local grid refinement approach, the analysis is best performed from a “coarsening” point of view. Near the object the grid resolution is kept constant while the grid is coarsened towards the boundaries of the domain.

At a Reynolds number of 10 the flow readily converges to a steady-state solution. The resulting solution is smooth and is not expected to pose any difficulties for refinement interfaces.

The numerical results presented in figure 11 show that the drag-force predictions are accurate even on grids that are severly coarsened further away from the object. Even on a base grid of 30×30 cells and 3 refinement layers the predicted drag-coefficient is merely 0.1% different from the answer obtained on a uniformly refined grid, while the number of grid points has been reduced by almost a factor of 20. The resolution of the grid close to the object is of main importance. This observation is confirmed by the plot at the bottom of figure 11 which shows that the drag force predictions are almost on top of eachother regardless of the amount of coarsening further away from the cylinder.



legend	L	base grid dimensions
—	0	$80^2, 160^2, 320^2, 480^2, 640^2, 720^2, 960^2$
–△–	1	$40^2, 60^2, 80^2, 120^2, 160^2$
–×–	2	$20^2, 40^2, 80^2, 160^2$
–○–	3	$20^2, 40^2, 60^2, 80^2, 90^2$
–□–	4	$20^2, 40^2, 60^2, 80^2, 90^2$
–+–	5	$20^2, 40^2, 50^2$

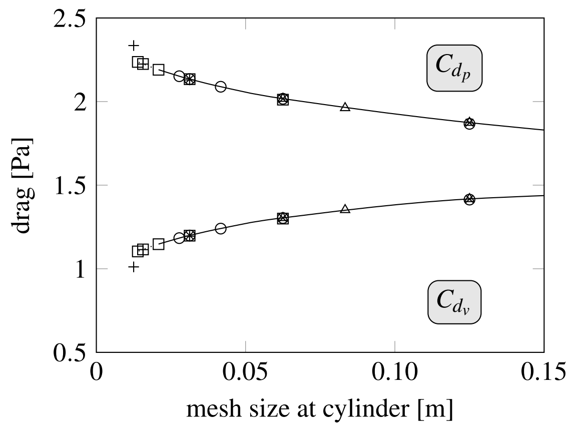


FIGURE 11: *Top:* Drags forces, split in a viscous (C_{d_v}) and pressure-induced (C_{d_p}) component) acting on the square cylinder as calculated with different grid configurations. Convergence in terms of the total number of grid cells. L indicates the total number of refinement regions with ratio $\mathbf{r} = \mathbf{2}$ starting from the base grid dimensions listed in the rightmost column. *Bottom:* Drag force predictions with respect to the number of grid cells along the face of the cylinder.

Flow around a square cylinder (Re=100)

At a Reynolds number of 100 the flow is unsteady, as the flapping shear layer results in an oscillating drag and lift force on the

cyl.	grid	L	**r**	#	Δt	St	C_d	$C_{l,rms}$
6×12	240^2	-	-	58k	0.05	0.1500	1.61003	0.23711
	80^2	1	3	19k		0.1500	1.60996	0.23618
9×18	360^2	-	-	0.13M	0.02	0.1483	1.56107	0.21447
	90^2	2	2	20k		0.1483	1.56085	0.21371
	40^2	2	3	12k		0.1483	1.56103	0.21192
	45^2	3	2	6.9k		0.1483	1.56012	0.20890
18×36	720^2	-	-	0.52M	0.01	0.1483	1.51592	0.19251
	80^2	2	3	47k		0.1483	1.51578	0.19159
	90^2	3	2	27k		0.1483	1.51581	0.19172
	45^2	4	2	9.7k		0.1500	1.51931	0.19081
27×54	1080^2	-	-	1.2M	0.002			
	40^2	3	3	28k		0.1483	1.50760	0.18687
36×72	1440^2	-	-	2.1M	0.0025			
	180^2	3	2	??k		0.1483	1.50464	0.18821
	90^2	4	2	37k		0.1483	1.50517	0.18824
	45^2	5	2	15k		0.1483	1.48698	0.18903

TABLE 2: Numerical predictions of the Strouhal number (St), drag force C_d, and the root-mean-square of the lift coefficient $C_\text{l,rms}$ for flow around a square cylinder (Re=100) on uniform and locally refined grids. In all cases a refinement ratio of 3×3 is used. The first column ('cyl.') lists the grid resolution around the cylinder, 'grid' the resolution of the base grid, L the number of refinements, '**r**' the refinement ratio, '#' the total number of grid points and Δt the time step size.

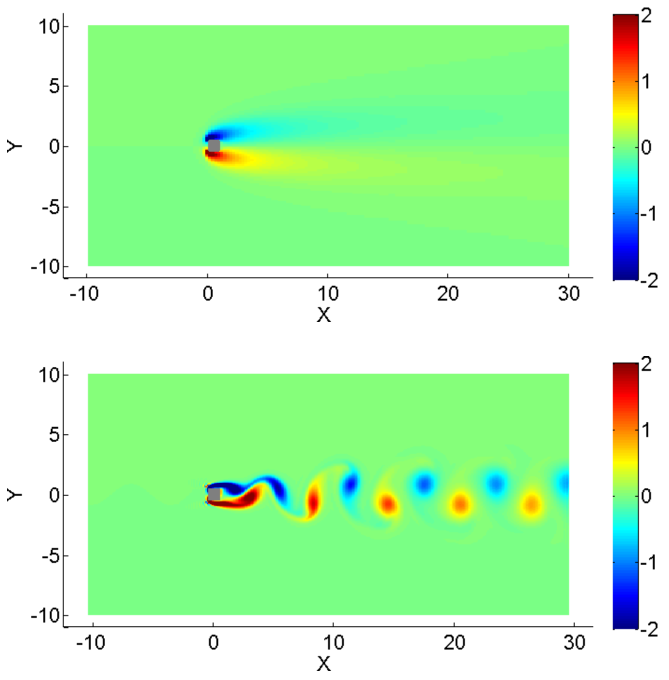


FIGURE 12: Snapshots of the vorticity for flow around a square cylinder. *Top*: Re=10, *Bottom*: Re=100.

cylinder. This test case clearly provides a more challenging test for the local grid refinement method.

Several refinement configurations are used together with their uniform counterparts. A second-order time integration scheme is used in order to focus on the spatial discretization errors introduced at refinement interfaces and by coarsening in general. The results shown in table 2 illustrate that the number of grid points can be reduced significantly while obtaining similar predictions for the drag and lift coefficients as on a uniformly refined grid. At higher resolutions the simulation time on uniform grids quickly becomes too large while locally refined grids still provide a reasonable alternative.

4 OUTLOOK ON ADAPTIVE REFINEMENT

In turbulent flow both small and large scale features play an important role. In areas where the velocity gradients show large variations a higher grid resolution is needed. One way of applying adaptive grid refinement is to apply refinement or coarsening on the basis of the criterion (as suggested in [9]):

$$\phi = \frac{h\,|\nabla \times \mathbf{u}|}{\overline{u}}, \tag{16}$$

where h is the local grid spacing and $\overline{u}$ is a characteristic velocity of the flow (e.g. the average or the maximum). Quantity (16) is evaluated in all cells of the grid block. If the quantity supersedes a given threshold, the grid block is refined further. If the quantity is lower than a certain threshold in all cells of the block, it is coarsened.

Flow around a square cylinder (Re=1,000)

As a simple test case, the adaptive refinement criterion (16) is applied to the simulation of flow around a square cylinder at Reynolds number 1000. The appropriate choice of the thresholds depends on the desired balance between time savings and numerical accuracy. For the present test case the thresholds have been chosen by inspecting the typical values of ϕ in a simulation on a coarse grid after which a suitable range is chosen to capture the vortical structures. In the test case shown in figure 13 the parameters were set to $\alpha = 0.01$, $\beta = 0.10$, $L = 5$, $r_i = r_j = 2$ and the characteristic velocity was set to the inflow velocity, $\overline{u} = 1.0$ [m/s]. The block size was set to $b_i \times b_j = 8 \times 8$. In figure 13 it can be observed that the refinement condition (16) is well capable

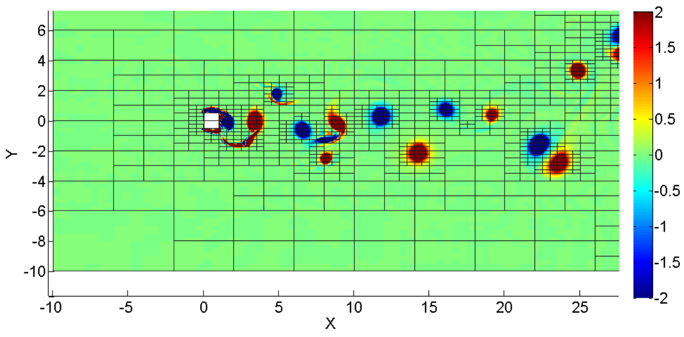


FIGURE 13: Color plot of the vorticity for flow around a square cylinder at Reynolds number 1,000. The color range has been constrained to highlight the vortices. Each block consists of 8 × 8 grid cells. The grid resolution ranges from $\Delta x_0 = 1.0$ [m] in the coarsest grid blocks up to $\Delta x_4 = 0.0625$ [m] at the highest refinement level ($\Delta y = \Delta x/2$)

of tracking the vortices that are generated at the cylinder. Without grid adaptivity the grid would have to be refined uniformly to capture the vortices behind the cylinder. At similar resolution this would cost approximately 610k grid cells whereas the adaptive grid on average consists of approximately 65k grid cells.

Wedge entry

Although the local refinement method can also be applied in the presence of a free surface or a non-grid-aligned (moving) object, in demanding applications it is better to avoid refinement interfaces in these regions since they are only first-order accurate. Especially in the presence of cut cells the interface treatment is less accurate because there obtaining an accurate interpolation of missing velocities is a non-trivial task. Refinement interfaces are more accurate in regions without free surface or cut cells. Using adaptive refinement it is possible to do so. Refinement is applied using the following condition:

$$\phi(i,j,k;\ell) = \begin{cases} +1 & \text{moving object and water}, \ell < L, \\ +1 & \text{water and air}, \ell < L \\ -1 & \text{only water or only air}, \ell > 0 \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

If $\Phi = 1$, the grid block is refined, if $\Phi = -1$, it is coarsened. Additional criteria could be used to adaptively adjust the grid resolution elsewhere. In the present case, the grid block is simply coarsened if it is fully located under water. Figure 14 shows a snapshot of the simulation of a wedge entry in a confined block of water. Refinement condition (17) makes sure that no resolution changes occur at the free surface and near the moving object in the vicinity of water. The requirement of graded grid refinement automatically results in a smooth transition from high resolution at the free surface to lower resolution under water, further away from the moving object.

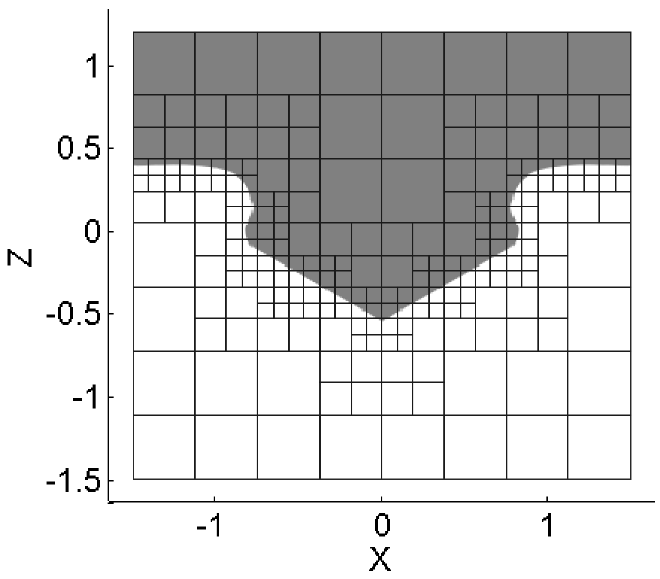


FIGURE 14: Snapshot of the grid layout for the simulation of a wedge entry.

5 CONCLUSIONS

The local grid refinement as introduced in [1, 10] has been extended to support refinement interfaces at the free surface and a first step has been made to implement adaptive grid refinement for turbulent flows as well as flows involving a free-surface and/or moving objects. Although the interface scheme is formally first-order accurate, the typical (global) convergence rate that is observed is still close to second order. By restricting refinement interfaces to smooth areas of the flow, as is common practice, effectively the convergence rate of the original second-order accurate method is maintained. In simulations with a strongly time-varying solution adaptive grid refinement can be used to increase resolution where needed, e.g. to capture vortical structures, or to avoid refinement interfaces in difficult areas, e.g. near the free-surface or non-grid-aligned geometry. The usefulness of adaptive grid refinement in (offshore) applications involving free-surface flows and moving objects is evident. It is our intention to apply the method to industrial-scale problems and to combine it with ongoing research on interactively moving objects.

REFERENCES

[1] Veldman, A. E., Luppes, R., van der Heiden, H. J., van der Plas, P., Düz, B., and Huijsmans, R. H., 2014. "Turbulence modeling, local grid refinement and absorbing boundary conditions for free-surface flow simulations in offshore applications". In ASME 2014 33rd International Conference on Ocean, Offshore and Arctic Engineering, American Society of Mechanical Engineers, pp. V002T08A076–V002T08A076.

[2] Verstappen, R., and Veldman, A., 2003. "Symmetry-preserving discretization of turbulent flow". *Journal of Computational Physics,* ***187***(1), pp. 343–368.

[3] de Tullio, M., De Palma, P., Iaccarino, G., Pascazio, G., and Napolitano, M., 2007. "An immersed boundary method for compressible flows using local grid refinement". *Journal of Computational Physics,* ***225***(2), pp. 2098–2117.

[4] Minion, M., 1996. "A projection method for locally refined grids". *Journal of Computational Physics,* ***127***(1), pp. 158–178.

[5] Liu, Q., 2011. "A stable and accurate projection method on a locally refined staggered mesh". *International Journal for Numerical Methods in Fluids,* ***67***(1), pp. 74–92.

[6] Uzgoren, E., Singh, R., Sim, J., and Shyy, W., 2007. "Computational modeling for multiphase flows with spacecraft application". *Progress in Aerospace Sciences,* ***43***(4), pp. 138–192.

[7] Losasso, F., Gibou, F., and Fedkiw, R., 2004. "Simulating water and smoke with an octree data structure". In ACM Transactions on Graphics (TOG), Vol. 23, ACM, pp. 457–462.

[8] Youngs, D. L., 1984. "An interface tracking method for a 3d eulerian hydrodynamics code". *UK Unclassified AWRE/44/92/35 (3rd ed.) Atomic Weapons Research Establishment, AWRE, MOD (PE), Aldermasten, Berks (April 1987).*

[9] Popinet, S., 2003. "Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries". *Journal of Computational Physics,* ***190***(2), pp. 572–600.

[10] van der Heiden, H. J., van der Plas, P., Veldman, A. E., Verstappen, R. W., and Luppes, R., 2013. "Efficient computation and modeling of viscous flow effects in comflow". In ASME 2013 32nd International Conference on Ocean, Offshore and Arctic Engineering, American Society of Mechanical Engineers, pp. V007T08A060–V007T08A060.