INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2017; **109**:487–513
Published online 27 May 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.5294

Initialization of high-order accuracy immersed interface CFD solvers using complex CAD geometry

Paul E. DesJardin[1,*,†], Brian T. Bojko[1] and Matthew T. McGurn[2]

[1]*Department of Mechanical and Aerospace Engineering, University at Buffalo, the State University of New York, Buffalo, NY 14260-4400, USA*
[2]*Reaction Engineering International Inc., Murray, UT 84107, USA*

SUMMARY

This study concerns the development of a numerical methodology for initializing immersed interface-based CFD solvers for using complex computer-aided design (CAD) geometry. CFD solvers with higher-order discretization stencils require larger stencil widths, which become problematic in regions of space where insufficient mesh resolution is available. This problem becomes especially challenging when convoluted triangulated surface meshes generated from complex solid models are used to initialize the cut-cells. A pragmatic balance between desired local geometry resolution and numerical accuracy is often required to find a practical solution. Here, a robust iterative fill algorithm is presented that balances geometry resolution with numerical accuracy (*via* stencil size). Several examples are presented to illustrate the use of this initialization procedure that employs both the original CAD generated triangulated surface mesh, along with a level set representation of the surface to initialize cut-cells and boundary proximity measures for creation of CFD stencils. Convergence error analysis of surface area and enclosed volumes is first presented to show the effects of fill on the geometry as a function of desired stencil size and grid resolution. The algorithm is then applied to geometrically complex problems using large eddy simulation. Two problems are considered. The first is flow around the Eiffel Tower. The second is a combustion swirler in the context of a design problem.
Copyright © 2016 John Wiley & Sons, Ltd.

Received 17 November 2015; Revised 22 April 2016; Accepted 24 April 2016

KEY WORDS: immersed interface; CFD; cut-cell; CAD geometries

1. INTRODUCTION

The use of immersed interface-based methodologies for CFD simulations of complex geometries is useful because of their relative simplicity and computational efficiency compared with either body-fitted or unstructured meshes. The use of an immersed interface and how it is described can vary depending on the end use and level of geometric information needed at the interface. Early implementation of immersed interfaces often relied on volume fraction-based descriptions to mark interfaces, that is, volume of fluid descriptions [1]. More recently, more accurate interface descriptions have been introduced using high-order polynomial representations [2] or level set functions [3, 4]. Augmentation of discrete operators using immersed interfaces often depends on the mathematical form of the governing equations being solved and are often divided into either incompressible (or low speed) [5–7] or compressible formulations [8–11]. Representation of interfaces often involve one or more steps of (1) injecting local interface nodes, (2) augmenting the size and weights of discrete stencils in the vicinity of the boundary, and (3) populating nodes in the solid region near

*Correspondence to: Paul E. DesJardin, Department of Mechanical and Aerospace Engineering, University at Buffalo, the State University of New York, Buffalo, NY 14260-4400.
†E-mail: ped3@buffalo.edu

Copyright © 2016 John Wiley & Sons, Ltd.

interfaces to enforce local boundary conditions, for example, ghost fluid methods. Reviews on the topic can be found by Osher and Fedkiw [4], and Mittal and Iaccarino [12].

In all of the aforementioned methodologies, numerical issues arise when the length scale associated with changes in geometry is of the same order as the local mesh spacing. Solutions to alleviate this problem include either local mesh adaptivity or lowering the accuracy of discretization stencils. In many cases, however, grid refinement is not necessarily desirable. Arbitrary mesh refinement on interface geometry can result is excessively small cells that artificially reduce the time step for resolving flow regions that may not be of primary interest. This problem becomes especially pronounced with the use of surface meshes that are directly extracted from 3D CAD models with complex geometry. The surface meshes generated from 3D CAD models are often noisy, containing small artifacts and holes. Removing detail from the surface mesh or filling in holes can be a cumbersome and time intensive process, often requiring user intervention. For rapid design analysis, it is desirable to have an automatic means to clean up surface geometry and also a method to easily alter baseline geometry without recreating the 3D model. These problems become especially challenging when high-order accurate numerical methodologies are required for using either direct numerical simulation or large eddy simulation (LES).

A different processing approach is therefore desirable that allows *a priori* selection of geometry resolution defined by the CFD mesh (or minimum length scale using adaptive mesh refinement (AMR)) for any CAD derived surface mesh. Regions that cannot be resolved by the CFD mesh are simply ignored *via* a fill, or projection, operation. While there have been prior studies to develop various projection operations, they often rely on the working assumption that the interior volume of the surface is resolved by the CFD grid, or more specifically, the size of the discrete operator is able to resolve the small interior feature [13]. In this study, this assumption is relaxed through an iterative fill procedure. The fill operation is useful in two contexts. The first is removing small details of a triangulated surface mesh created from a CAD model without recreating the original surface mesh. Examples include removing unimportant void spaces or surface features that are not deemed important in the CFD analysis. The second purpose of the proposed fill operation is to automatically remove voids that cannot be supported by high-order (and often large) CFD stencils. One of the challenges in filling under-resolved fluid regions is the impact on neighboring nodes, which are influenced with the new location of the interface. For instance, Figure 1 illustrates a progressive fill operation assuming the minimum stencil size of $N = 3$ for any stencil in the domain. Figure 1(a) shows the structured grid with a fluid-solid boundary outlined (solid black line). Nodes that fall within the closed surface are assumed to lie in the gas phase and those outside in the solid phase. The nodes in the solid phase are initially marked as inactive ($\times$), and internal boundary nodes are identified lying on the boundary in the gas phase ($\bullet$). To evaluate the boundary conditions, however, at least three nodes normal to the surface are required to evaluate local gradient information, as defined by $N = 3$ in this example (higher-order stencils could also be used). Boundary nodes that do not contain at least three active nodes from the boundary are deemed inactive ($\times$), as illustrated in Figure 1(b). The newly defined set of inactive nodes necessitate the introduction of new boundary nodes ($\bullet$) shown in Figure 1(c). The process of eliminating nodes that do not satisfy the minimum stencil requirements can be repeated until the boundary nodes no longer change. The desirable end result of this iterative procedure is the systematic filling in of severely unresolved regions. For the case in Figure 1(c), all of the boundary nodes have at least three nodes of support on the boundary nodes in each direction; therefore, only a single iteration is required. For more complex geometry, multiple iterations are required until the total number of boundary nodes ceases to change.

The rest of this study begins with a description of triangulated surface mesh geometry commonly used to described standard CAD models. The proposed fill algorithm for use with cut-cell formulations is then presented consisting of three steps of (1) mapping of surface mesh to a level set function (γ) defined on a 'fill mesh', (2) filling in regions of the fill mesh defined by either a minimum stencil size or a prescribed smallest length resulting in a modified level set field, and (3) reconstruction of the modified interface by a cut-cell reconstruction method using the modified level set function. Results are presented showing the accuracy and convergence rate of the iterative fill procedure for computing cut areas and volumes for simple 2D circles and 3D spheres. A criterion is developed for the minimum resolved length scale for which cells will be completely filled that may be used

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

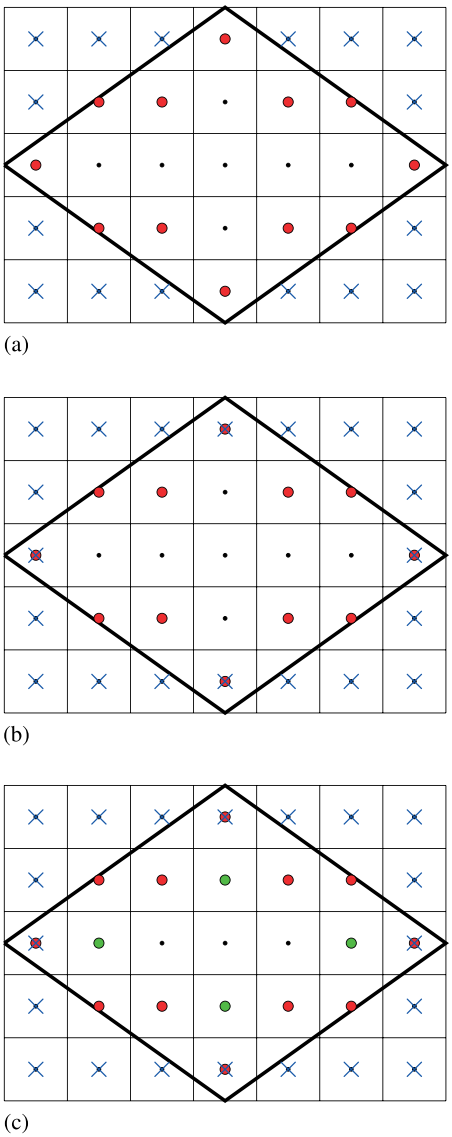


Figure 1. Illustration of a fill operation showing (a) initial identification of inactive (×) and boundary (•) nodes , (b) revised inactive nodes based on minimum stencil size, and (c) revised selection of boundary (•) nodes.

to *a priori* define the desired scale of geometric resolution. Performance estimates are presented for a moving oscillating sphere showing a decrease in computational cut-cell overhead with increasing mesh refinement. Examples of more geometrically complex problems are then presented. The first is flow through a model of the Eiffel Tower. This model problem is useful because it contains well-defined length scale ranges and is geometrically complex. A practical industrial example is next presented of fluid flow through a combustion chamber using LES. Finally, conclusions on the major findings from this study are summarized.

2. ALGORITHM FORMULATION

The overall algorithm is summarized in the following sections. The first step in the algorithm is construction of the level set field on the CFD mesh using a triangulated surface mesh. The surface mesh is extracted from CAD geometries and is assumed to be in a stereolithography (STL) format, where elements are defined by a collection of node vertices and vectors defining the triangle facet normals.

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

For small STL models, the level set can be initialized by searching through each of the triangles to determine which triangle is the closest and then projecting the surface normal distance through a series of vector operations (Section 2.1). These operations, however, become prohibitively time consuming with larger models. Previous studies have employed various tree-based algorithms to quickly sort elements and reduce the scope of elements for updating the level set function [13–15]. In this study, we follow a similar methodology where a box tree structure is introduced to sort the surface triangle elements to reduce the overall cost of initialization (Section 2.2). Multiple surfaces are combined together using the level set merging algorithm of Schroder *et al.* [16] (Section 2.3), which will be shown to be useful for making simple modifications to exiting models. Internal boundary nodes are next defined, and the level set is augmented using a iterative fill procedure, which ensures enough spacing for discrete operators (Section 2.4). Lastly, cut-cell reconstruction is performed using Lagrangian polynomials and discretization of the subcell interface using triangles to compute cut areas and volumes.

2.1. Mapping STL to a level set defined on the CFD mesh

The most direct means of defining the level set is to cycle through all of the mesh elements to determine which element produces the minimum surface distance. The magnitude of the final value of the shortest distance from the surface to point of interest, $\boldsymbol{x}_p$, is taken as the minimum value over all candidate elements, that is, $|\boldsymbol{d}| = \min_{e=1}^{N}(|\boldsymbol{d}_e|)$. Each of the individual element distance $|\boldsymbol{d}_e|$ calculation consists of three steps: (1) bounding box check, (2) projected volume, and (3) element edge calculations.

The first step in the candidate element distance calculation, the bounding box check, is designed as a computationally low cost operation to determine if the second and third steps should be evaluated. This requires defining a bounding box for each element where the lower and upper corners of the box are defined as follows: $\boldsymbol{x}_{box,min} = \min_{i=1}^{3}(\boldsymbol{x}_i)$ and $\boldsymbol{x}_{box,max} = \max_{i=1}^{3}(\boldsymbol{x}_i)$, respectively, where $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$ are the element nodes illustrated in Figure 2(a). The bounding box check determines if there is no possibly of the current candidate element being closer to $\boldsymbol{x}_p$ based upon the current $|\boldsymbol{d}|$. This check can be interpreted as expanding each element's bounding box outward by the current minimum distance $|\boldsymbol{d}|$ along each box side normal. If $\boldsymbol{x}_p$ is outside of the expanded bounding box, it cannot be closer than the current minimum $|\boldsymbol{d}|$, and the second and third steps are skipped for this candidate element. This check can be compactly represented as follows:

$$\big((\boldsymbol{x}_p \cdot \boldsymbol{n}_x) < (\boldsymbol{x}_{box,min} \cdot \boldsymbol{n}_x - |\boldsymbol{d}|)\big) \vee \big((\boldsymbol{x}_{box,max} \cdot \boldsymbol{n}_x + |\boldsymbol{d}|) < (\boldsymbol{x}_p \cdot \boldsymbol{n}_x)\big) \tag{1a}$$

$$\big((\boldsymbol{x}_p \cdot \boldsymbol{n}_y) < (\boldsymbol{x}_{box,min} \cdot \boldsymbol{n}_y - |\boldsymbol{d}|)\big) \vee \big((\boldsymbol{x}_{box,max} \cdot \boldsymbol{n}_y + |\boldsymbol{d}|) < (\boldsymbol{x}_p \cdot \boldsymbol{n}_y)\big) \tag{1b}$$

$$\big((\boldsymbol{x}_p \cdot \boldsymbol{n}_z) < (\boldsymbol{x}_{box,min} \cdot \boldsymbol{n}_z - |\boldsymbol{d}|)\big) \vee \big((\boldsymbol{x}_{box,max} \cdot \boldsymbol{n}_z + |\boldsymbol{d}|) < (\boldsymbol{x}_p \cdot \boldsymbol{n}_z)\big) \tag{1c}$$

where $\boldsymbol{n}_x$, $\boldsymbol{n}_y$, and $\boldsymbol{n}_z$ are unit vectors in the x, y, z coordinate directions, respectively. If any of the operations in Equation (1) are true, the element cannot be closer than the current minimum $|\boldsymbol{d}|$, and the second and third steps are skipped for this candidate element.

The second step of the distance calculation consists of checking to see if the node of interest, $\boldsymbol{x}_p$, lies within the projected volume of a triangle element. The condition for when this is satisfied is illustrated in Figure 2(a) and given as follows[17]: $\big[(\boldsymbol{x}_{1\to p} \cdot \boldsymbol{n}_3 \leqslant 0) \wedge (\boldsymbol{x}_{2\to p} \cdot \boldsymbol{n}_1 \leqslant 0) \wedge (\boldsymbol{x}_{3\to p} \cdot \boldsymbol{n}_2 \leqslant 0)\big]$, where $\boldsymbol{x}_{i\to j} \equiv \boldsymbol{x}_j - \boldsymbol{x}_i$, $\boldsymbol{n}_e = (\boldsymbol{x}_{1\to 2} \times \boldsymbol{x}_{1\to 3})/||(\boldsymbol{x}_{1\to 2} \times \boldsymbol{x}_{1\to 3})||$ is the element face normal and $\boldsymbol{n}_1, \boldsymbol{n}_2$ and, $\boldsymbol{n}_3$ are the edge normals defined as:

$$\boldsymbol{n}_1 = \boldsymbol{x}_{2\to 3} \times \boldsymbol{n}_e \tag{2a}$$

$$\boldsymbol{n}_2 = \boldsymbol{x}_{3\to 1} \times \boldsymbol{n}_e \tag{2b}$$

$$\boldsymbol{n}_3 = \boldsymbol{x}_{1\to 2} \times \boldsymbol{n}_e \tag{2c}$$

If $\boldsymbol{x}_p$ lies within the the projected volume of the element, $|\boldsymbol{d}_e|$ is taken as the shortest distance between $\boldsymbol{x}_p$ and the plane defined by $\boldsymbol{n}_e$ as follows: $|\boldsymbol{d}_e| = |\boldsymbol{n}_e \cdot \boldsymbol{x}_{p\to c}|$.

The third step of the calculation is executed if $\boldsymbol{x}_p$ falls outside of the projected volume. Each edge of the triangle is checked, and $|\boldsymbol{d}_e|$ is taken as the shortest distance to any of the line segments

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

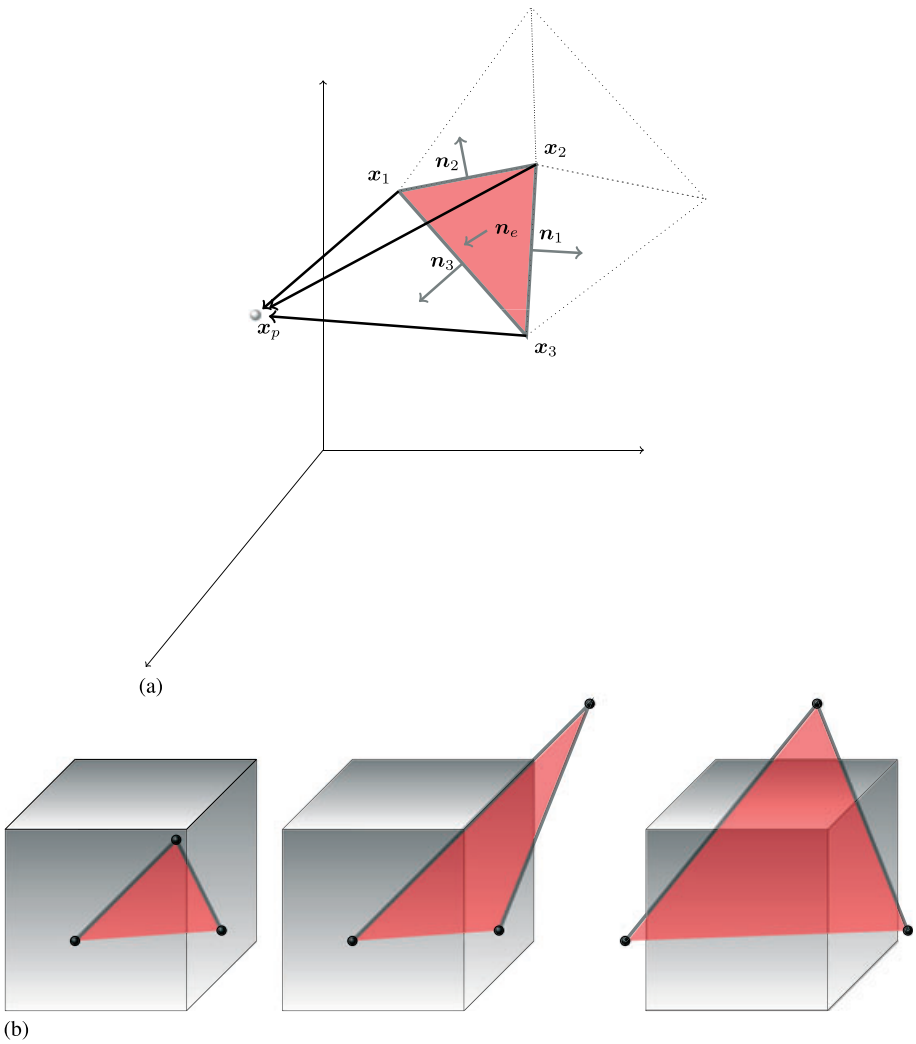


Figure 2. Initialization of level set function showing (a) calculation of surface normal distance from point of interest ($\boldsymbol{x}_p$) to closest surface element and (b) three surface element checks for sorting surface elements into mesh tree boxes corresponding to small (left), intermediate (center), and large elements (right).

defining the triangle. This three step calculation is repeated over all candidate elements, and the final value of the shortest distance from the surface to node $\boldsymbol{x}_p$ (in magnitude) is taken as the minimum value, that is, value of $|\boldsymbol{d}| = \min_{e=1}^{N}(|\boldsymbol{d}_e|)$. The direction of $\boldsymbol{d}$ is taken as that of the area normal of the element with the minimum value of $|\boldsymbol{d}_e|$. The value of the level set function, γ, is then set equal to $\gamma = |\boldsymbol{d}|\frac{(\boldsymbol{x}_{p\to c}\cdot\boldsymbol{n}_e)}{|(\boldsymbol{x}_{p\to c}\cdot\boldsymbol{n}_e)|}$, where $\left(\boldsymbol{x}_{p\to c}\cdot\boldsymbol{n}_e\right)$ will be negative for CFD nodes within the gas and positive for values in the solid.

2.2. *Mapping STL to a level set using a box tree structure*

For large numbers of surface elements, the direct method of computing the level set from Section 2.1 is computationally prohibitive because several vector operations are required for every element check and for each node. To reduce the number of operations, the surface elements are sorted using quad- (2D) or oct-(3D) box tree data structures [13–15, 18] to localize the calculation of γ to only those CFD nodes closest to the surface mesh. A hierarchical box tree is created by subdividing the space into subsequently smaller boxes. Each box contains a lookup list to the surface triangle elements that reside in it. A triangle may be listed in more than one box. The lowest level box (level 0)

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

is defined as a bounding box containing the entire STL model. Subsequently, smaller boxes (children) are created through box subdivision into equally sized boxes. Children are created only if they contain any of the surface elements contained by the parent; therefore, the number of surface elements per box reduces with increasing levels.

Three separate checks are used to assess if a surface element resides in a child box. These checks correspond to limiting cases where the size of the surface element is either smaller, of the same size, or larger than a given box, as illustrated in Figure 2(b). For small elements (left case in Figure 2(b)), all the nodes defining the surface element, $\boldsymbol{x}_i^e$, are checked to see if they lie within the region of a given box, Ω^b. If $\boldsymbol{x}_i^e \in \Omega^b$ for any value of $\boldsymbol{x}_i^e$, then the element is tagged as part of the collection for that box.

The second element criterion is for intermediate sized elements, elements that are comparable in size with a box (center case in Figure 2(b)). For this case, line segments defining the triangle, $\boldsymbol{x}_{i\to j}^e \equiv \boldsymbol{x}_j^e - \boldsymbol{x}_i^e$, are checked to see if they intersect any of the box surfaces, where $\boldsymbol{x}_i^e$ and $\boldsymbol{x}_j^e$ are defined as the spatial location of any two nodes on the triangle. Any position along the line segment, $\boldsymbol{x}_p$, may be parameterized as follows: $\boldsymbol{x}_p = \boldsymbol{x}_i^e + s\boldsymbol{x}_{i\to j}^e$ where $s = 0$ corresponds to $\boldsymbol{x}_p = \boldsymbol{x}_i^e$ and $s = 1$ corresponds to $\boldsymbol{x}_p = \boldsymbol{x}_j^e$. Expressing $\boldsymbol{x}_p$ in terms of a coordinate system attached to a given face of a box, $\boldsymbol{x}_p = x_n\boldsymbol{n} + x_{t1}\boldsymbol{t}_1 + x_{t2}\boldsymbol{t}_2$, and setting x_n to correspond to the surface of the box then the values of s, x_{t1}, and x_{t2} can be determined,

$$s = \left(x_n - \boldsymbol{x}_i^e \cdot \boldsymbol{n}\right) / \boldsymbol{x}_{i\to j}^e \cdot \boldsymbol{n} \tag{3a}$$

$$x_{t1} = \boldsymbol{x}_i^e \cdot \boldsymbol{t}_1 + s\left(\boldsymbol{x}_{i\to j}^e \cdot \boldsymbol{t}_1\right) \tag{3b}$$

$$x_{t2} = \boldsymbol{x}_i^e \cdot \boldsymbol{t}_2 + s\left(\boldsymbol{x}_{i\to j}^e \cdot \boldsymbol{t}_2\right) \tag{3c}$$

where $\boldsymbol{n}$, $\boldsymbol{t}_1$, $\boldsymbol{t}_2$ are the unit vectors normal and tangential to the plane, respectively, and x_n is the (known) magnitude of the normal component. Assuming the face coordinate system origin is located at the center of the box then $x_n = \Delta/2$ where Δ is the length of a box side. The conditions for a triangle line segment intersection with a face are then $0 \leqslant s \leqslant 1$, $|x_{t1}| \leqslant \Delta/2$ and $|x_{t2}| \leqslant \Delta/2$. If all three conditions are satisfied for any of the triangle line segments, for any of the box surfaces, then the element is tagged as part of the collection for that box.

The third check accounts for the possibility of a large surface element intersecting through the box where all element nodes from the element reside outside the box (right case in Figure 2(b)). For this case, the line segments defining the edges of the box, $\boldsymbol{x}_p = \boldsymbol{x}_i^b + s\boldsymbol{x}_{i\to j}^b$, are checked for intersection with the triangle surface element – essentially a role reversal of the previous check with the coordinate system now being defined relative to the element surface. The intersection location along the line segment is defined as

$$s = \left(x_n - \boldsymbol{x}_i^b \cdot \boldsymbol{n}\right) / \boldsymbol{x}_{i\to j}^b \cdot \boldsymbol{n} \tag{4}$$

where $0 \leqslant s \leqslant 1$ defines the condition that the box line segment intersects a plane containing the triangular element. To further restrict the check to the intersection of a triangular element, the following additional conditions must also be satisfied, $[(\boldsymbol{x}_{1\to i} \cdot \boldsymbol{n}_3 \leqslant 0) \wedge (\boldsymbol{x}_{2\to i} \cdot \boldsymbol{n}_1 \leqslant 0) \wedge (\boldsymbol{x}_{3\to i} \cdot \boldsymbol{n}_2 \leqslant 0)]$, where $\boldsymbol{n}_k$, $\boldsymbol{x}_{1\to i}$, $\boldsymbol{x}_{2\to i}$ and $\boldsymbol{x}_{3\to i}$ all have the same definitions as in Section 2.1.

Figure 3 shows examples using the sorting box tree structure for a sphere of radius unity consisting of 1280 triangles. The surface mesh, box tree mesh, and centerline level set slice are shown for tree levels of (a) four and (b) three, respectively. Highlighted surface elements in red indicate the collection of elements for an arbitrarily selected box. As shown, with use of the hierarchical sorting, subsequently finer level boxes are always located closer to the surface, which becomes useful later on when estimating the level set.

Once the triangles are sorted into all levels then γ is computed at the corners of the boxes at the finest level, using the direct method discussed in Section 2.1. The value of γ at corners of subsequently coarser grained boxes (lower level numbers) are computed using a combination of copying and linear extrapolation. For boxes that share the same corner node, values of γ are simply copied from the finer level to the coarser levels (high to low levels). For hanging corners (corners

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

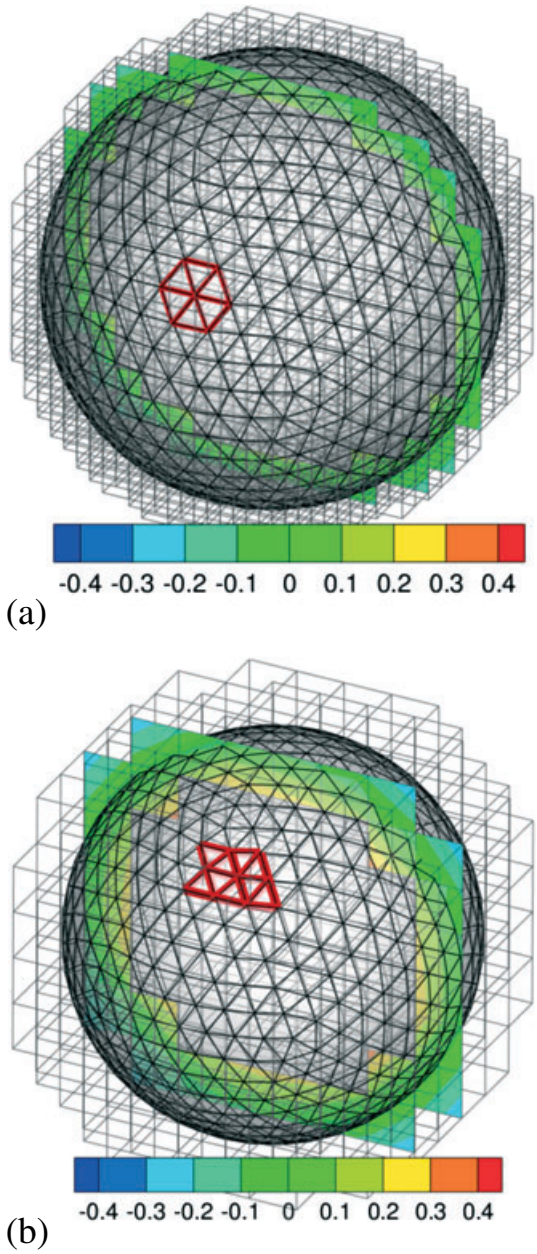


Figure 3. Box tree illustration showing sphere with box outlines and level set slice at centerline for tree levels of (a) four and (b) three. Highlighted surface elements in red indicate the collection of elements for a single box.

which do not exist at a finer level), the closest box at the finer level is first identified along with the closet corner within that box. The value of γ at the box corner of the coarser box is estimated through extrapolation, *via*, $\gamma_l = \gamma_{l+1} + SGN(\gamma_{l+1})|\boldsymbol{x}_l - \boldsymbol{x}_{l+1}|$, where γ_l represents the desired value of γ at a box corner at tree level, l and $|\boldsymbol{x}_l - \boldsymbol{x}_{l+1}|$ are the distance between the box corners at levels l and $l+1$, respectively. Because lower level boxes are always located farther away from surface mesh than higher level boxes, the extrapolation is guaranteed to yield the correct sign of the level set. Figure 3(a) shows the calculation of the level set at a tree level of four. For this example, the maximum tree level is also set to four; therefore, the level set at the box corners is exact because the direct method described in Section 2.1 is used. Figure 3(b) shows contours of γ at a tree level of three using a combination of copied box corners and extrapolation. The reconstructed γ is very

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

reasonable; however, it is not exact because of the extrapolation error. In practice, extrapolation errors do not impact the calculation of cut-cell information because the size of the smallest tree box is selected to be larger than the size of the CFD stencil (to be discussed).

Once values of γ are computed at all box corners for all levels, the tree is used to efficiently compute γ at the CFD nodes. For every CFD node, the tree is recursively searched from the lowest to highest levels using the children of each box to find the smallest box containing the CFD node of interest, $\boldsymbol{x}_p$. Bilinear interpolation is used to first estimate $\gamma(\boldsymbol{x}_p)$. If $\gamma(\boldsymbol{x}_p) < n\Delta x$, then a refined value of $\gamma(\boldsymbol{x}_p)$ is computed using the exact method of Section 2.1, by using only the list of elements stored in the box of interest containing $\boldsymbol{x}_p$. The factor n is a multiple of the local CFD spacing (Δx). For all the examples in this study, n was chosen equal to N, the size of the stencil, to ensure that γ is computed exactly at all nodes within the stencil range near the surface.

The maximum tree level is automatically computed so the smallest box (highest tree level) is comparable with the smallest CFD cell, *via*, $l_{max} = ln(b_{min}/\Delta_{x,min})/ln(2)$, where b_{min} is the (smallest) length of the STL bounding box and $\Delta_{x,min}$ is the smallest CFD cell.

2.3. *Mapping of multiple STLs to a level set*

Multiple triangulated surface meshes are combined, and additional geometry is added (or subtracted) using the level set merging algorithm of Schroder *et al.* [16]. In this approach, level sets (γ^i) for each surface mesh are first constructed, where i represents the ith surface mesh. Merging of level sets follows the assembly procedure, where for all subsequent level-set functions, γ^i, $i > 1$ are compared with γ^0 using the following rules [16],

(1) If both values are positive, set $\gamma^0 = min(\gamma^0, \gamma^i)$
(2) If both values are negative, set $\gamma^0 = -\sqrt{(\gamma^0)^2 + (\gamma^i)^2}$
(3) if values have different signs, set $\gamma^0 = min(\gamma^0, \gamma^i)$

2.4. *Fill, internal boundary node generation, and level set augmentation*

Once the surface geometry is stored in terms of both a triangulated surface mesh and a level set function, internal CFD boundary nodes are generated using an iterative fill algorithm, which deactivates CFD nodes that do not support stencils of size N. Cut-cells are identified using two separate checks. The first is a switch in the level set sign of the vertex nodes defining the boundaries of the cut-cells, as illustrated in Figure 4. For the cell centered node located at (i, j), the four surrounding edges (identified in red) have corresponding vertex node locations of $[i, j]$, $[i+1, j]$, $[i+1, j+1]$ and $[i, j+1]$. Because the level set function is often stored at the cell centers, third-order Lagrange interpolating polynomials are used to define γ at the vertex nodes, which results in a centered stencil about the vertex node to ensure consistency in γ across cell boundaries. Cut-cells are identified when $\gamma_i \gamma_j < 0$ where γ_i and γ_j is the level set at the end point of an edge. For under resolved geometries, such as thin shells, γ_i and γ_j may have the same sign even though the cell is cut. A second check is therefore necessary to account for thin structures. In this case, cells for which $|\gamma| < \Delta x$, which have not already been tagged, are checked to see if any of the CFD cell edges intersect the closest triangle surface element using the same series of checks discussed in Section 2.2.

Internal boundary nodes are defined through an iterative procedure. At the start of the iteration, one internal boundary node is assigned for each cut-cell. The start of the iteration therefore has the maximum number of internal boundary nodes. The first step in the iteration is to identify the number of nodes between each cut-cell of interest and the surrounding boundaries, which are defined by the boolean variable `active`. At the beginning of the iteration, all cells for which $\gamma > 0$ are deemed inactive, that is, `active=false`. The number of active nodes from the location (i, j) to an inactive node is determined and stored. The number of active nodes in the positive direction from (i, j) is defined by the integers δi^+ and δj^+. The number of active nodes in the negative directions is defined by δi^- and δj^-. For the cut-node (i, j) illustrated in Figure 4, $\delta i^+ > N$, $\delta i^- = 0$, $\delta j^+ = 1$ and $\delta j^- = 1$. Cut-cells, which satisfy the criterion $((\delta i^+ = 0) \wedge (\delta i^- < N)) \vee ((\delta i^- = 0) \wedge (\delta i^+ < N)) \vee ((\delta j^+ = 0) \wedge (\delta j^- < N)) \vee ((\delta j^- = 0) \wedge (\delta j^+ < N))$, are deemed inactive, that is, there is not enough room for a stencil of size N. All nodes between (i, j)

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

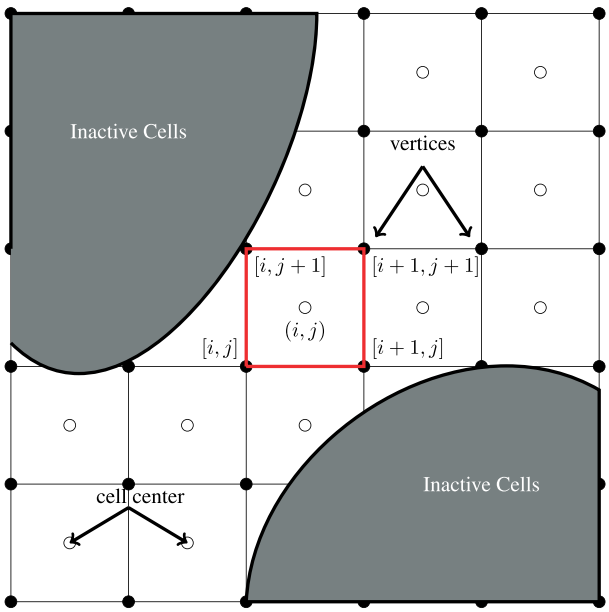


Figure 4. Cut cell proximity measures for node (i, j). For this case, $\delta i^+ > N$, $\delta i^- = 0$, $\delta j^+ = 1$ and $\delta j^- = 1$.

and the particular direction that fail to satisfy the minimum stencil requirement are all tagged as inactive by setting `active=false`. The result is a systematic direction-by-direction deactivation of nodes. Once all internal boundary nodes are checked, δi^+, δi^-, δj^+, and δj^- are updated using the revised `active` variable. New internal boundary nodes are then introduced, which satisfy the criterion $(((\delta i^- = 0) \wedge (\delta i^+ \geqslant N)) \vee ((\delta i^+ = 0) \wedge (\delta i^- \geqslant N))) \wedge (((\delta j^- = 0 \wedge \delta j^+ \geqslant N)) \vee ((\delta j^+ = 0) \wedge (\delta j^- \geqslant N)))$, that is, nodes lying directly adjacent to inactive nodes. The process of deactivating neighbor nodes and introducing new internal boundary nodes is repeated until the total number of internal boundary nodes no longer changes.

After convergence, the level set is updated to be consistent with the final distribution of inactive cells. Two criteria are explored for updating γ. The first is setting $\gamma = 0$ in those cells for which originally $\gamma < 0$ (gas phase), but the node is deemed inactive, thereby moving the interface to the cell center. The second approach is to switch the sign of γ from negative to positive while maintaining the same magnitude. Both of these options yield a level set, which will no longer strictly satisfy the definition of a distance function, that is, $|\nabla\gamma \neq 1|$, in the neighborhood of the surface. Because the $|\nabla\gamma = 1|$ condition is not critical for the reconstruction of the cut-cell interface areas and volumes (to be discussed), further corrections to γ were not explored and will be shown that the resulting change in the geometry is slight. While more sophisticated level set re-distancing approaches could be pursued, the result may lead to active nodes, which no longer satisfy the minimum stencil size check, therefore further iteration would be required.

2.5. *Reconstruction of cut-cell areas and volumes*

Field modeling using cut-cells often requires the calculation of cut cell areas and volumes for use with the implementation of internal boundary conditions or local descritization using finite volume solvers. Several methods can be used to compute cut areas and volumes using level set functions defined on structured meshes. Perhaps the most straight forward means for computing these quantities is using the following identities: $V_{cut} = \int_\Omega (1 - H(\gamma))dV$ and $A_{cut} = \int_\Omega \delta(\gamma)|\nabla\gamma|dV$ where δ is the Dirac delta function, H is the Heaviside function, and Ω is the volume of the cut-cell. Discrete operators can be developed for $H(\gamma)$ and $\delta(\gamma)$ and for finite difference formulations, have shown to result in second order accuracy for 2D problems [19, 20].

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

For the finite volume formulations, localized estimates of V_{cut} and A_{cut} are desirable. In this study, an algorithm is developed that relies on high-order Lagrangian interpolating polynomials to locally represent γ within the cut-cell. Accurate evaluation of A_{cut} and cell faces are determined using triangulization of those surfaces for integration. The divergence theorem is employed to determine V_{cut}. All operations are conducted in a generalized curvi-linear coordinate system for use with a body-fitted regularized coordinate system, $\boldsymbol{\zeta}$.

In this approach, the level set function is first interpolated from the cell centered mesh to the surrounding vertices defining the corners of the cells (shown in Figure 4) using third-order, multi-dimensional Lagrange interpolating polynomials. Odd-order polynomials are critical to ensure continuity of the level set across cells faces. Line segment cut locations are first identified where $\gamma = 0$ using a modified bi-section algorithm. The cut locations are used to decompose each cut-surface into a collection of surface triangles, as illustrated in Figure 5 for (a) one of the CFD cell faces and (b) the CFD cell cut-face.

The area of each triangle shown in Figure 5(a) is computed as follows: $\delta \boldsymbol{A}_1 = J \boldsymbol{g}^1 \delta A^*$ where $J \equiv |\partial \boldsymbol{x} / \partial \boldsymbol{\zeta}|$ is Jacobian for the body-fitted mesh, $\boldsymbol{g}^1$ is the contra-variant vector for ζ_1 surface, and δA^* is the area of the triangle in $\boldsymbol{\zeta}$ coordinates. Point C_F in Figure 5(a) indicates the location of the face centroid and is defined as the average (in $\boldsymbol{\zeta}$) of surrounding cut-locations and vertices

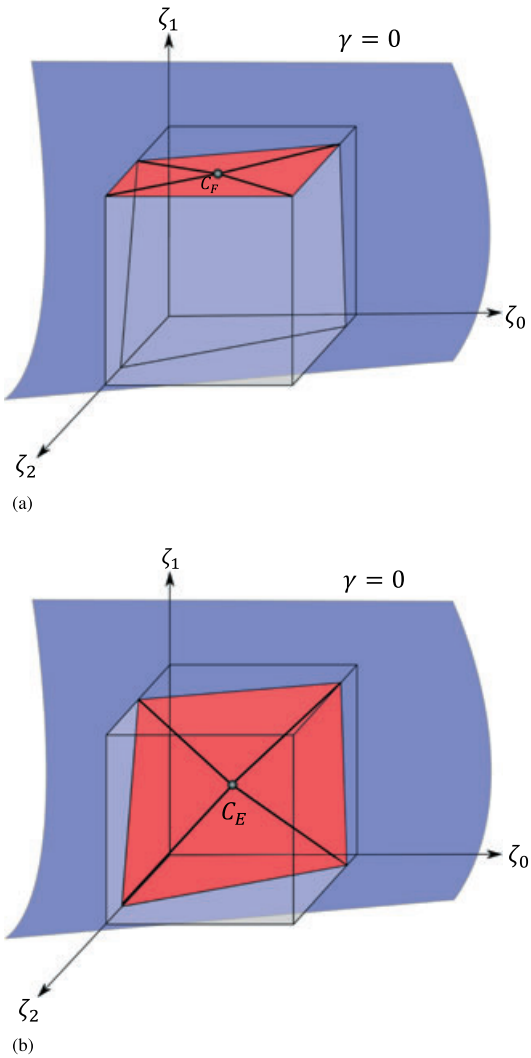


Figure 5. Surface triangular decomposition of (a) cell faces and (b) cell cut-face for computing cut areas and volumes.

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

where $\gamma < 0$ that reside on the highlighted face. All grid metrics required for determining $\delta \boldsymbol{A}_j$ are assumed constant and evaluated at the cell center using second-order Lagrangian polynomials centered on the cell. The total area for each face is defined as sum of the magnitudes for that surface, that is, $A_{j,cut} = \sum |\delta \boldsymbol{A}_j|$.

For 3D calculations, the cut-volume of each cell is also desired and computed using the divergence theorem, $V_{cut} = \int_{\Omega_{cut}} (\nabla \cdot \boldsymbol{x})/3 \, dV = (1/3) \int_{\partial\Omega_{cut}} \boldsymbol{x} \cdot d\boldsymbol{A} = (1/3) \int_{\partial\Omega^*_{cut}} J\boldsymbol{x} \cdot \boldsymbol{g}^j \, dA^*$, which is approximated as the sum of the triangles formed for each CFD cell surface and A_{cut} shown in Figure 5(b). A_{cut} is constructed using triangles formed from the center of the cut-elements at point, C_E, and the cut locations of the line segments of the surrounding faces. To ensure point C_E is located on the $\gamma = 0$ level set, an iterative search algorithm is used that marches from the cell along the $\nabla\gamma (= \boldsymbol{g}^j \partial\gamma/\partial\zeta_j)$ direction until $\gamma = 0$ is found. This guarantees that all nodes from the triangles formed lie directly on the $\gamma = 0$ surface.

As will be shown, the triangulated surface decomposition results in second order accuracy for the determination of A_{cut} and V_{cut} in 2D. In 3D, A_{cut} is second order accurate and V_{cut} is first order. The accuracy limiting assumption of V_{cut} is from the use of planar triangles to compute area (δA^*). While higher order accuracy could be achieved by using basis functions for integration over the triangles, this is unnecessary because the original STL file information is inherently low order.

3. FLOW SOLVER

The CFD modeling methodology is based on the use of LES where the Navier–Stokes equations are pre-filtered using a positive definite filtering function, G, (*i.e.*, $G(\Delta_f, |\boldsymbol{x} - \boldsymbol{x}'|) > 0 \ \forall \ \boldsymbol{x}'$) possessing the usual normalization and symmetry properties, where Δ_f is the filter width. A Favre filtered quantity, $\tilde{\phi}$, is defined by the following convolution integral.

$$\tilde{\phi}(\boldsymbol{x}) = \frac{\overline{\rho\phi}}{\bar{\rho}} = \frac{1}{\bar{\rho}} \int_D \rho(\boldsymbol{x}')\phi(\boldsymbol{x}')G(\Delta_f, |\boldsymbol{x} - \boldsymbol{x}')d\boldsymbol{x}' \tag{5}$$

Application of the filtering to the compressible form of the Navier–Stokes equations results in the following set of model transport equations for mass, momentum, and energy,

$$\frac{\partial\bar{\rho}}{\partial t} + \nabla \cdot (\bar{\rho}\tilde{\boldsymbol{u}}) = 0 \tag{6a}$$

$$\frac{\partial(\bar{\rho}\widetilde{\boldsymbol{u}})}{\partial t} + \nabla \cdot (\bar{\rho}\widetilde{\boldsymbol{u}}\widetilde{\boldsymbol{u}}) = \nabla \cdot \left(-\bar{p}\underset{\sim}{\boldsymbol{I}} + \underset{\sim}{\bar{\boldsymbol{\tau}}} + \underset{\sim}{\boldsymbol{T}}_{\boldsymbol{uu}}\right) + \bar{\rho}\tilde{\boldsymbol{g}} \tag{6b}$$

$$\frac{\partial(\bar{\rho}\tilde{e}_t)}{\partial t} + \nabla \cdot \left(\bar{\rho}\widetilde{\boldsymbol{u}}\tilde{h}_t\right) = \nabla \cdot \left(\underset{\sim}{\boldsymbol{T}}_{\boldsymbol{u}h_t} + \widetilde{\boldsymbol{u}} \cdot \underset{\sim}{\boldsymbol{\tau}} - \bar{\boldsymbol{q}}\right) + \bar{\rho}\tilde{\boldsymbol{u}} \cdot \tilde{\boldsymbol{g}} \tag{6c}$$

where $\bar{\rho}$ is the density, $\widetilde{\boldsymbol{u}}$ is the velocity, $\bar{p}$ is the pressure, $\tilde{e}_t (= \tilde{h}_t - \tilde{R}\tilde{T})$ is the total resolved energy, and $\tilde{h}_t$ $(= \tilde{h} + \widetilde{\boldsymbol{u}} \cdot \widetilde{\boldsymbol{u}}/2)$ is the total enthalpy including the resolved sensible enthalpy ($\tilde{h}$) and kinetic energy. A compressible Newtonian fluid is assumed for the viscous stress tensor, $\underset{\sim}{\bar{\boldsymbol{\tau}}} = -\frac{2}{3}\mu(\tilde{T})\underset{\sim}{\boldsymbol{I}}\nabla \cdot \widetilde{\boldsymbol{u}} + \mu(\tilde{T})\left(\nabla\widetilde{\boldsymbol{u}} + (\nabla\widetilde{\boldsymbol{u}})^T\right)$. Equal diffusivities are assumed for all species, and radiation heat transfer is ignored thereby simplifying the heat flux, $\bar{\boldsymbol{q}} = -\mu(\tilde{T})\left[\frac{C_P}{Pr}\left(1 - \frac{1}{Le}\right)\nabla\tilde{T} + \frac{1}{Sc}\nabla\tilde{h}\right]$. The second-order correlation quantities, $\underset{\sim}{\boldsymbol{T}}_{\alpha\beta}$, in Equation (6b) through (6c) represent unknown SGS correlation for variables α and β, and are defined as follows: $\underset{\sim}{\boldsymbol{T}}_{\alpha\beta} \equiv -\bar{\rho}(\widetilde{\alpha\beta} - \tilde{\alpha}\tilde{\beta})$. An abundance of models are currently available for closing these terms. The most common subgrid scale (SGS) models are the dynamic Smagorinsky and gradient diffusion models, which are the ones used in this study [21]. Because the focus of this study is not on LES SGS models, the interested reader is referred to [22, 23] for additional details.

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

3.1. Flow numerics and cut-cell methodology

A finite volume formulation for use on curvilinear meshes is employed using the AUSM family of flux vector splitting schemes [24, 25]. Time advancement is achieved using high-order Runge-Kutta methods. The convective fluxes required at the control volume surfaces are constructed using a combination of high-order upwind-biased [26], ENO [27], and MUSCL [28, 29] schemes with several different limiter options. For this study, second-order upwind-biased are used to solve the momentum equations and second-order ENO methods are used to solve density and energy.

A multi-block, curvilinear mesh domain decomposition is employed for solution using parallel computing facilities. Chemkin (or Cantera) formatted transport property information is read into the framework where species properties (*i.e.,* C_P, k, and D_{i-j}) are tabulated over a range of temperatures and pressures for efficient lookup during run-time. Mixing rules of Mathur *et al.* [30] and Bird *et al.* [31] are used to determine mixture weighted diffusion and thermal conductivity properties. Previous uses of the solver include direct numerical simulation and LES of buoyancy driven plumes [21], pool fires [32], flame spread [33], and multiphase blast waves [34].

The current cut-cell implementation of the flow solver is rather simplistic where internal boundary nodes are treated as the location of the physical boundary. The appeal of this simplification is that it avoids the need for addressing numerical stability problems associated with small cut-cells. One-dimensional characteristics boundary conditions are imposed at these internal boundaries assuming no-slip, isothermal wall conditions. [21, 35, 36]. Surface normals required for evaluation of the surface normal gradients are computed using the gradient of the level set function.

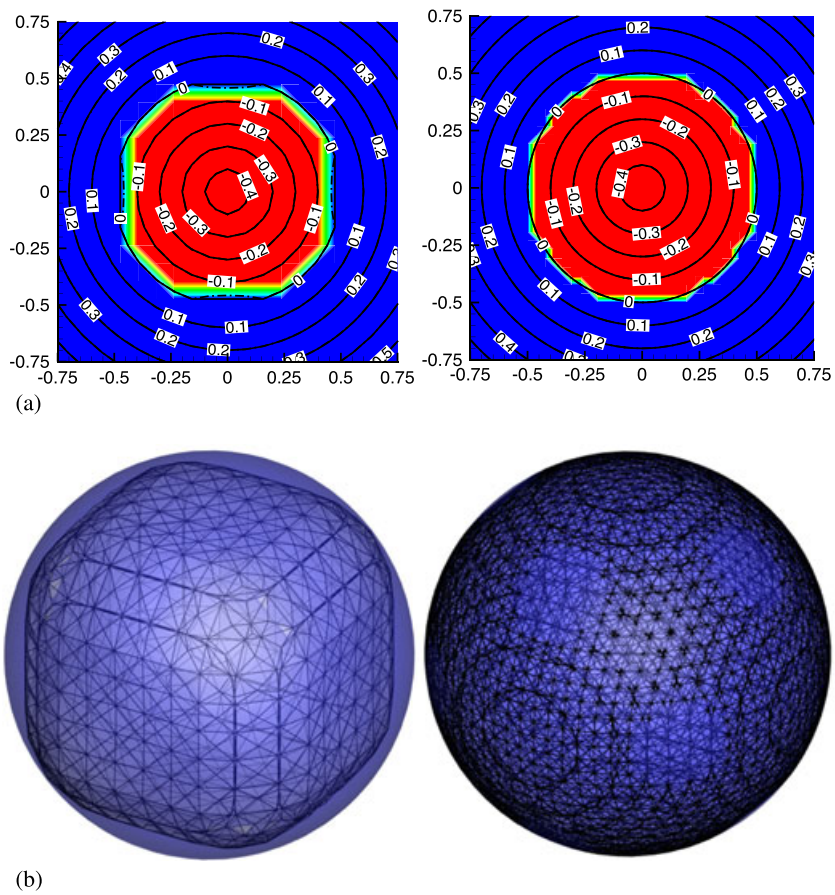


Figure 6. Effect of fill operation for (a) 2D circle and (b) 3D sphere problem using 20×20 ($\times 20$) and 40×40 ($\times 40$) meshes.

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

4. RESULTS

Several benchmark problems are first explored to assess the accuracy of the initialization algorithm for defining surface area and volume of enclosed surfaces for simple geometries. More geometrically complex examples are then presented to show the use of the methodology for flow through the Eiffel tower and an applied problem showing flow through a combustion swirler in the context of a design perturbation study.

4.1. Circle and sphere benchmarks

A set of simple benchmarks are first explored using 2D circles and 3D spheres with a diameter of unity solved on a computational domain of 1.5 diameters on a side. The purpose of exploring these cases is to understand the impact of the fill operation on overall interface area and volume for objects with simple curvature. For these cases, the exact level set function is given as $\gamma = r - 1/2$ leading to negative values inside the circle/sphere and positive values outside. Figure 6 shows the problem setup and the modified level set function after the fill operation for (a) 2D circle and (b) 3D sphere using a $20 \times 20(\times 20)$ and $40 \times 40(\times 40)$ meshes for $N = 5$. For each case, two different methods of γ correction are employed. The first is setting $\gamma = 0$ in inactive cells for which originally $\gamma < 0$. The second approach considers switching the sign from negative to positive while maintaining

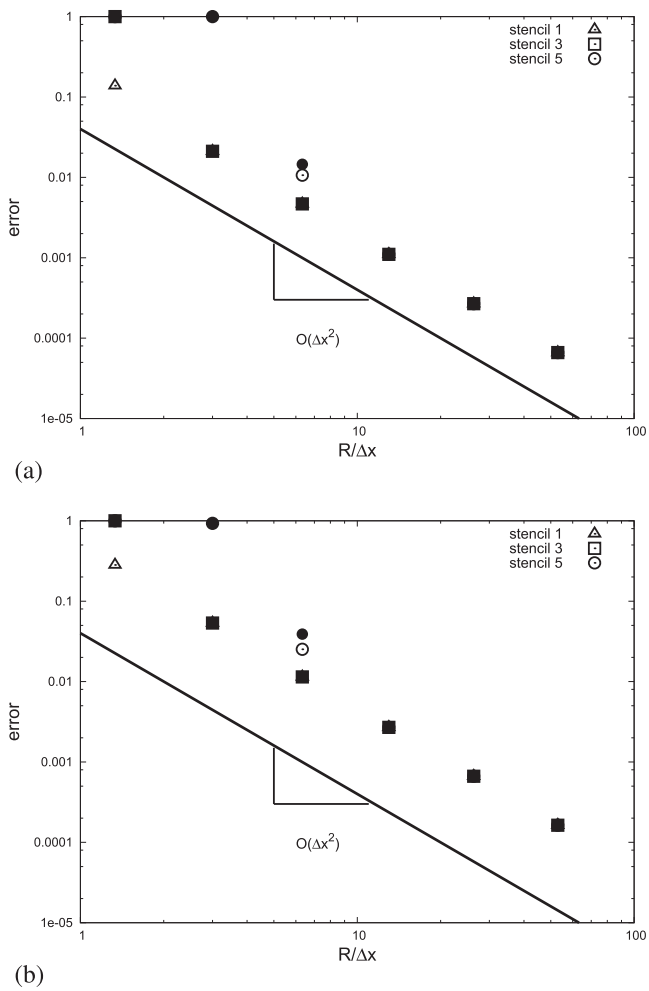


Figure 7. Error in (a) perimeter and (b) area of a circle as a function of $R/\Delta x$ using $\gamma = 0$ (open symbols) and $\gamma = -\gamma$ (closed symbols) correction methods.

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

the magnitude, that is, $\gamma = -\gamma$. The two cases are denoted by solid and dashed contour lines, respectively, in Figure 6. Also plotted as a color contour is the location of active (red) versus. inactive (blue) cells. As shown in Figure 6(a), the $\gamma = 0$ contour level is shifted slightly to accommodate the $N = 5$ stencil leading to flattened appearance at extrema locations that are aligned with the mesh. Comparing the solid and dashed lines shows that the differences between the two γ augmentation methods are small for the 20×20 case and indistinguishable for the 40×40 case, indicating that the methodology is relatively insensitive to how γ is augmented after the fill.

Figure 6(b) illustrates the same fill operation in 3D where the colored isosurface represents the exact solution and the tetrahedral mesh inside the sphere is generated using the augmented $\gamma = 0$

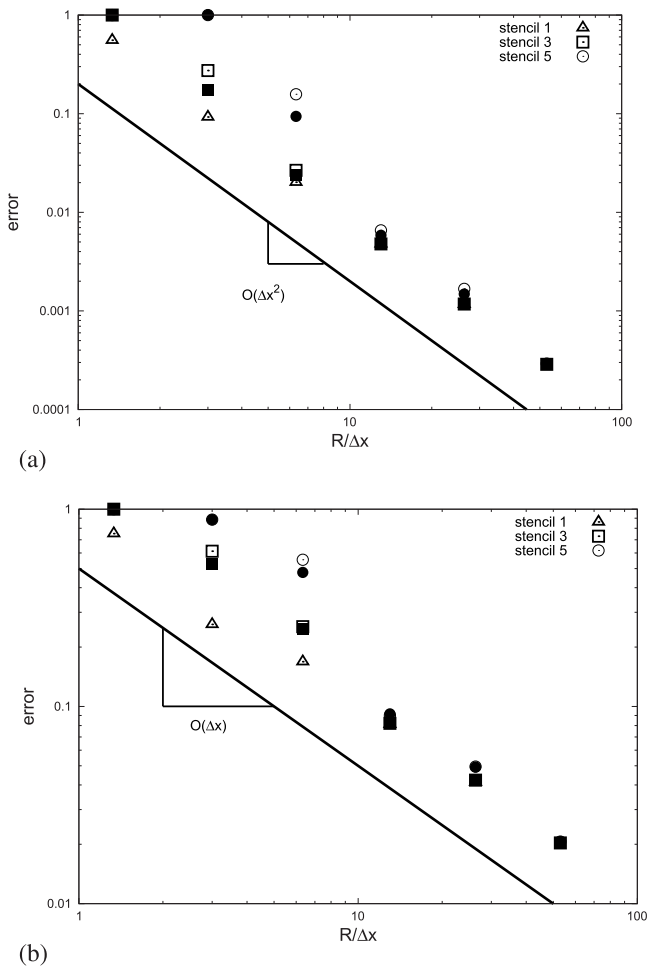


Figure 8. Error in (a) area and (b) volume of a sphere as a function of $R/\Delta x$ using $\gamma = 0$ (open symbols) and $\gamma = -\gamma$ (closed symbols) correction methods.

Table I. Summary of absolute timing estimates with increasing CFD and surface mesh resolution.

$R/\Delta x$	$\#ele$	level	N_{CFD}/N_{ele}	n_{CFD}	n_{coup}	CFD (s)	tree (s)	init (s)	fill (s)	Σ (s)
3.00	576	3	3.81	209	66	1.61	1.16	1.08	0.31	2.55
6.33	2,616	4	6.72	533	145	46.48	13.75	10.06	4.58	28.39
9.67	6,432	4	9.95	910	224	374.00	38.82	50.97	27.15	116.93
13.00	12,000	5	12.41	1299	308	1371.37	152.06	141.50	89.15	382.71
19.67	27,840	5	18.39	2131	465	6087.42	514.21	480.57	513.32	1508.1

Note, the Σ column represents the sum of building the surface mesh tree (tree), initialization of the level set on the CFD mesh (init) and fill (fill) operations.

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

Table II. Summary of relative timing estimates with increasing CFD and surface mesh resolution.

$R/\Delta x$	CFD	tree	init	fill	Σ
3.00	1/ 0.388	2.270 / 0.278	2.118 / 0.259	0.611 / 0.075	4.998 / 0.612
6.33	1/ 0.621	1.088 / 0.184	0.795 / 0.134	0.362 / 0.061	2.245 / 0.379
9.67	1/ 0.762	0.422 / 0.079	0.554 / 0.104	0.295 / 0.055	1.270 / 0.238
13.00	1/ 0.782	0.468 / 0.087	0.435 / 0.081	0.274 / 0.051	1.177 / 0.218
19.67	1/ 0.801	0.387 / 0.068	0.362 / 0.063	0.386 / 0.068	1.135 / 0.199

Two relative times are provided. The first is the cost of a single operation relative to a single operation of the CFD. The second is the cost relative to the entire simulation. Note, the Σ column represents the sum of building the surface mesh tree (tree), initialization of the level set on the CFD mesh (init) and fill (fill) operations.

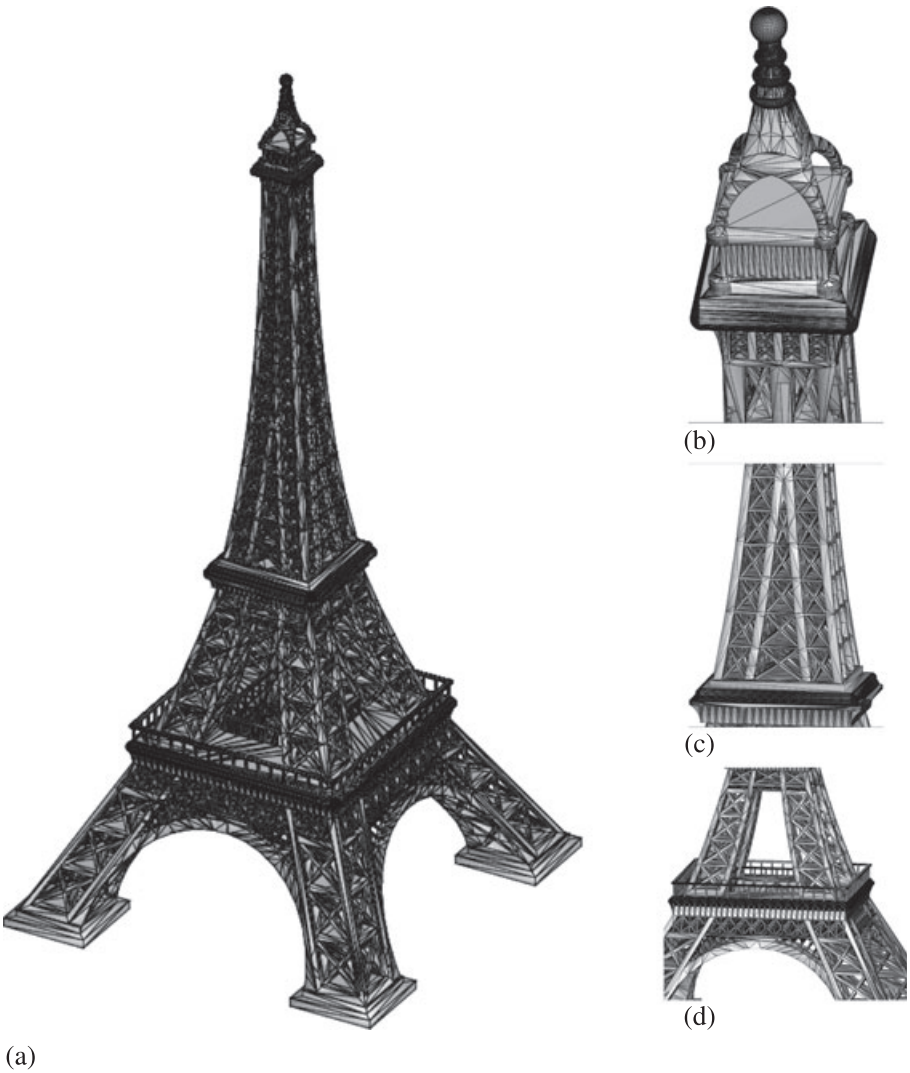


Figure 9. STL model of the Eiffel tower consisting of 74,521 vertices and 171,722 faces showing (a) entire model and detail at the (b) top , (c) middle and (d) bottom.

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

level set. For the low resolution $20 \times 20 \times 20$ case, the effect of the fill operation reduces the overall available surface area by squaring off regions for which the local stencil cannot support the local curvature from the STL geometry.

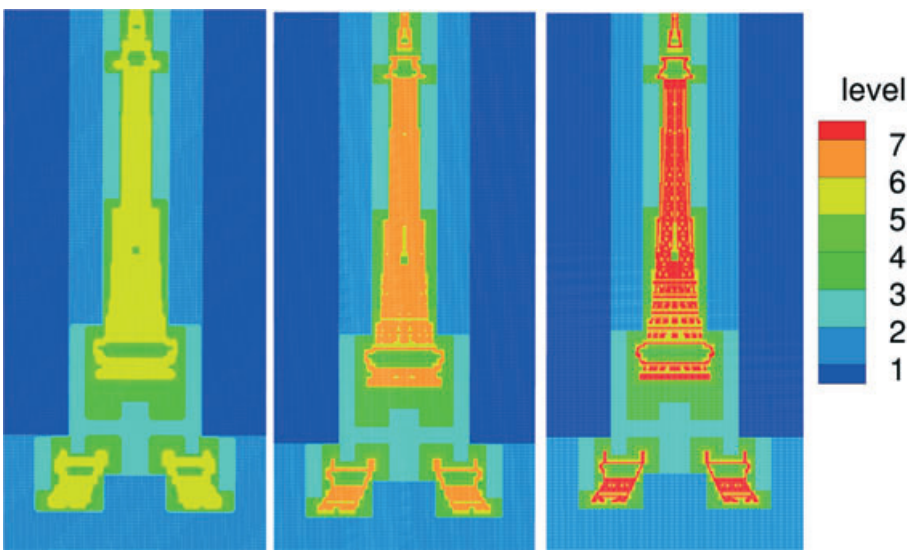
Figure 7 presents error in the calculation of (a) perimeter and (b) area for a 2D circle using $N = 1$, 3, and 5. The $\gamma = 0$ (open symbols) and $\gamma = -\gamma$ (closed symbols) correction methods are both shown. Error is presented as a function of the normalized number of nodes, $R/\Delta x$ – representing the number of cells that can fit within the radius of curvature, R. The $N = 1$ case (stencil width of unity) corresponds to the limit when the fill operation has no impact on the geometry because the stencil fits at all cut-cell locations. For this case, no level set correction is required, and the perimeter and area both have $O(\Delta x^2)$ rates of convergence.

For cases $N = 3$ and 5, the error is 100% when $R/\Delta x < 3$ and 5, respectively, indicating the entire region inside in the volume of the circle is filled, that is, unresolved geometry is filled. For $R/\Delta x >> 1$, the error attains the same asymptotic convergence rates as the $N = 1$ case indicating the effect of the fill operation on cut area and volume error is minimal. Based on these cases, the transition region between filled and non-filled can be generalized as the condition when $R/(N\ \Delta x) = 1$, corresponding to the stencil size being equal to the radius of curvature. For $R/(N\ \Delta x) > 1$, the effect of the fill operation rapidly diminishes and, as will be shown in the results, serves as a useful *a priori* criterion for the required CFD mesh resolution required to remove small, undesired geometric features.

As shown in Figure 7, differences between the level set correction methods are identical for most of the cases. The largest difference is the case when $R/(N\Delta x) = 1$ for $N = 5$ where the $\gamma = 0$ correction method appears to provide slightly better estimates.

Figure 8 presents similar error convergence plots for 3D spheres. Consistent with the previous case, the area exhibits a $O(\Delta x^2)$ rate of convergence without any fill and when $R/(N\Delta x) >> 1$. The volume, however, shows a $O(\Delta x)$ asymptotic rate of convergence, as might be expected because the divergence of the area is used to compute volume. In all cases, the $\gamma = -\gamma$ augmentation method produces slightly lower error, although the differences are small.

To assess the computational cost of the initialization procedure, the algorithm is extended for moving interfaces. A loosely coupled, dynamic time stepping approach is implemented where the position of surface is updated periodically using coupling time increments, Δt_c. The coupling time increment is dynamically estimated by dividing the maximum surface velocity into the smallest CFD

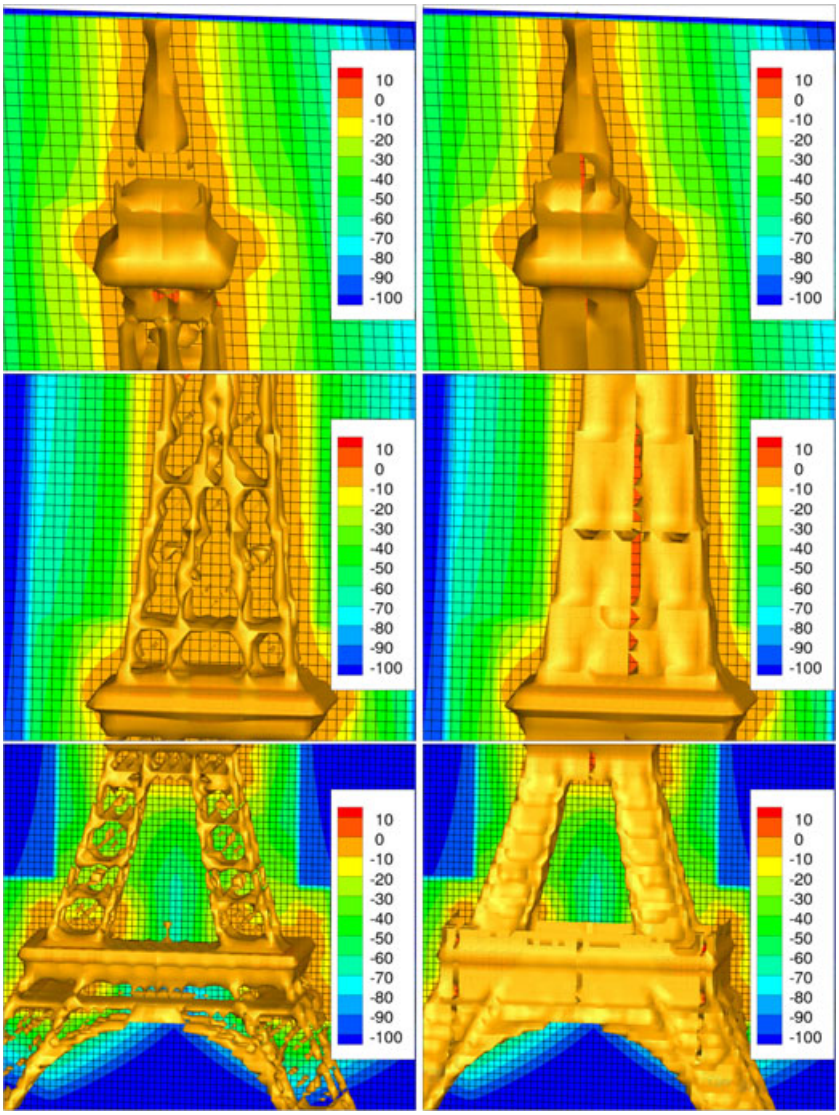


(a) (b) (c)

Figure 10. Contour slice of tree level for (a) $80 \times 160 \times 80$, (b) $160 \times 320 \times 160$, and (c) $320 \times 640 \times 320$ meshes.

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme



(a) (b)

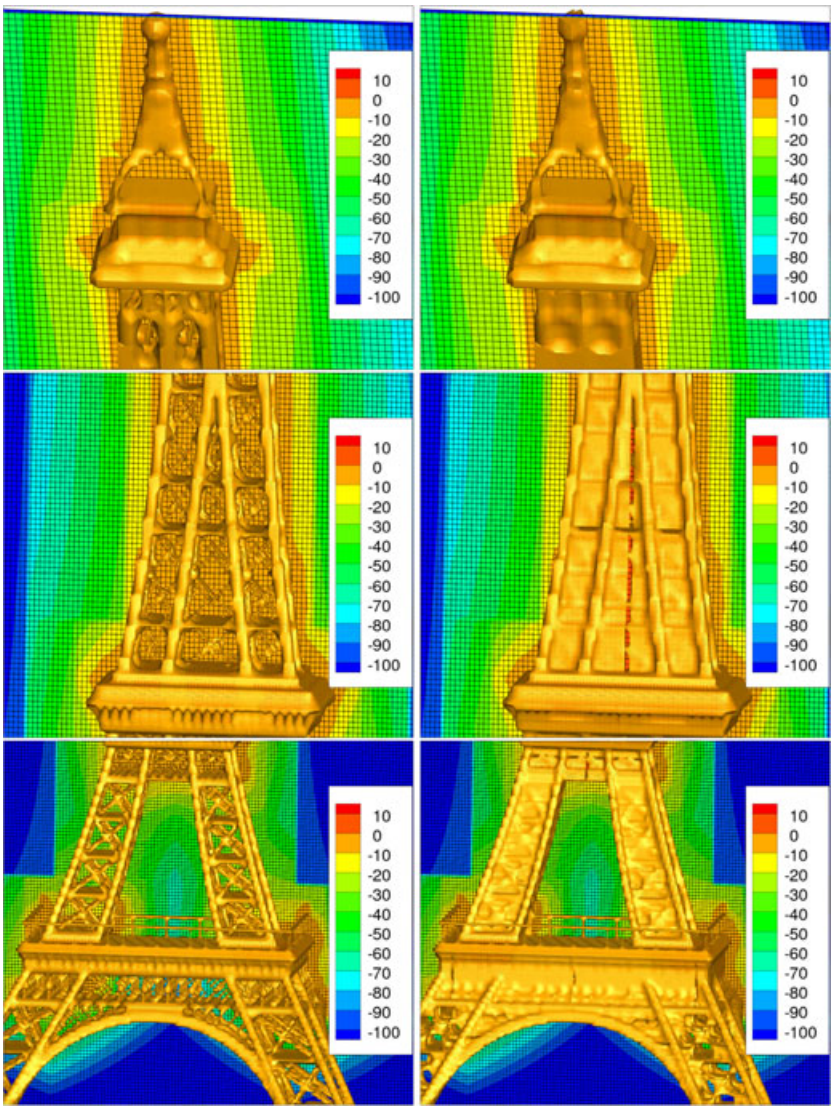
Figure 11. Level set isosurface and planar slice through centerline for a $80 \times 160 \times 80$ mesh showing (a) before and (b) after the fill operation.

cell size. At each coupling time, the major steps of (1) building the mesh tree, (2) initializing the level set on the CFD mesh, and (3) filling are timed. The test problem selected is a sphere undergoing, rigid body, oscillatory translation at velocity $u = (\pi/2)sin(2\pi t)$. The initial position of the sphere is $\boldsymbol{x} = \mathbf{0}$, and the maximum displacement is $\boldsymbol{x} = 0.5$ at $t = 0.5$ s. The size of the domain is expanded to $1 \times 1 \times 1$ m to account for the translation and cases are run for CFD meshes of 13^3, 26^3, 40^3, 53^3, and 80^3, corresponding to $R/\Delta x = 3$, 6.33, 9.67, 13.0, and 19.67, respectively. For each level of CFD grid refinement, a new STL is constructed for the sphere using the triangle surface reconstruction summarized in Section 2.5 thereby maintaining equivalent surface mesh resolution. Tables I and II summarize the timing results. For all cases, only a single processor for the CFD domain is utilized to ease interpretation of the results, even though the cases can be run using parallel computing.

Table I shows the number of surface elements ($\#ele$), max mesh tree level (level), ratio of CFD nodes (N_{CFD}) to number of surface elements (N_{ele}), number of CFD time steps (n_{CFD}), and number of coupling events (n_{coup}) with increasing mesh resolution. For each case, total times are

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme



(a) (b)

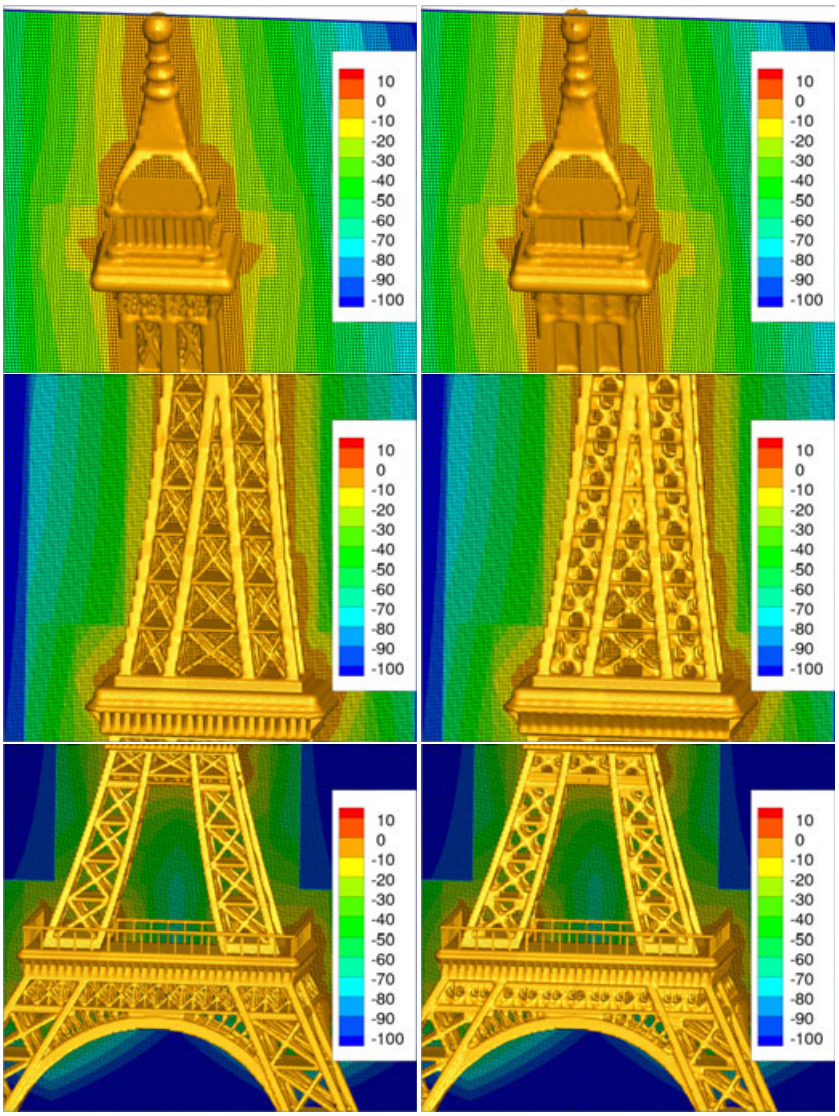
Figure 12. Level set isosurface and planar slice through centerline for a $160 \times 320 \times 160$ mesh showing (a) before and (b) after the fill operation.

recorded for CFD solver (CFD), construction of the surface mesh tree (tree), initialization of the level on the CFD mesh (init), and fill (fill) operations. The sum of the times for the tree build, level set initialization on the CFD and fill operations, is denoted in the last column as $\sum$, which represents the extra time required for coupling. A decrease in the mesh resolution results in a proportional increase in the amount of work required by the CFD solver versus that required for coupling. Consequently, the overall percentage of total work required for the coupling decreases with increasing mesh resolution.

Table II shows the relative times of building the tree, initializing the level set on the CFD mesh, and fill operations. For each case, two relative measures are presented. The first is the cost of a single operation relative to a single operation of the CFD. The second is the total cost of an operation relative to overall cost of the computation. For example, for the case of $R/\Delta x = 6.33$ using a CFD mesh of 26^3, the cost of a single coupling operation ($\sum$) is 2.245 times that of a single CFD time step. However, because fewer coupling operations are required compared with the CFD, then the total cost of coupling is only 37.9% of the total computation. Doubling the CFD mesh in each

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme



(a) (b)

Figure 13. Level set isosurface and planar slice through centerline for a 320 × 640 × 320 mesh showing (a) before and (b) after the fill operation.

direction ($R/\Delta x = 13.0$) reduces the costs of coupling by nearly a factor of 2 where the overall coupling costs decrease to 1.177 and 21.8%, respectively. Based on these results, when geometry curvature is reasonably well resolved, that is, $R/\Delta x > 10$, the cost of coupling will be less than 20% of the total. This scaling does not, however, account for the inherent inefficiencies associated with parallel computing, which should also be taken into account when using finer CFD meshes.

4.2. STL benchmark

The initialization algorithm is next applied to a complex STL model. Figure 9(a) shows the selected model representing the entire Eiffel Tower [37]. Close up views of the top, middle, and bottom deck are shown in Figures 9(b), (c) and (d), respectively. The entire model consists of 74,521 vertices and 171,722 faces with overall dimensions of 460 × 1017.8 × 460 (presumably units are in feet if the model is roughly to scale). The model serves as a challenging benchmark because there are a wide disparity of length scales ranging from the height of the tower to the various sized trusses making up

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

the structural supports (small at top, large at the bottom). Three different CFD mesh resolutions are used to illustrate the mapping and fill algorithms: $80 \times 160 \times 80$ ($\sim$ 1 million nodes), $160 \times 320 \times 160$ ($\sim$ 8 million nodes) and $320 \times 640 \times 320$ ($\sim$ 65 million nodes). To simplify the interpretation, each CFD mesh has uniform spacing resulting in the smallest length scales of 5.75, 2.875, and 1.44, respectively. Based on these dimensions, the max tree level computed 6, 7, and 8, from the coarsest to finest meshes. Figure 10 shows a slice of the mesh tree levels selected locally to initialize the level set. As expected, the finer the CFD mesh, the higher the tree level computed.

Figures 11–13 summarize the mapping and fill operations for all three meshes showing an isocontour of $\gamma = 0$ and contour slice through the centerline, along with the CFD mesh. For each figure, details of the top, middle, and lower sections are provided (a) before and (b) after the fill operation. For all cases, a stencil width of $N = 3$ is chosen for the fill operation, although any value of N can be selected. For the relatively coarse grid case shown in Figure 11(a), the CFD mesh is clearly insufficient to resolve individual structural members, resulting in a fragmented appearance of the structural members and is unsuitable for CFD computation because many nodes cannot support a $N = 3$ stencil. The use of the fill operation shown in Figure 11(b) fills in the regions that are clearly under-resolved resulting in a relatively smooth representation of the geometry.

Doubling the CFD mesh resolution in each direction is presented in Figure 12. As shown in (a) and (b), the upper and middle truss members still appear fragmented and therefore completely filled in. The largest truss structures near the bottom, however, are now better resolved than the earlier case. Individual trusses start to become visible resulting in a 'swiss cheese' appearance.

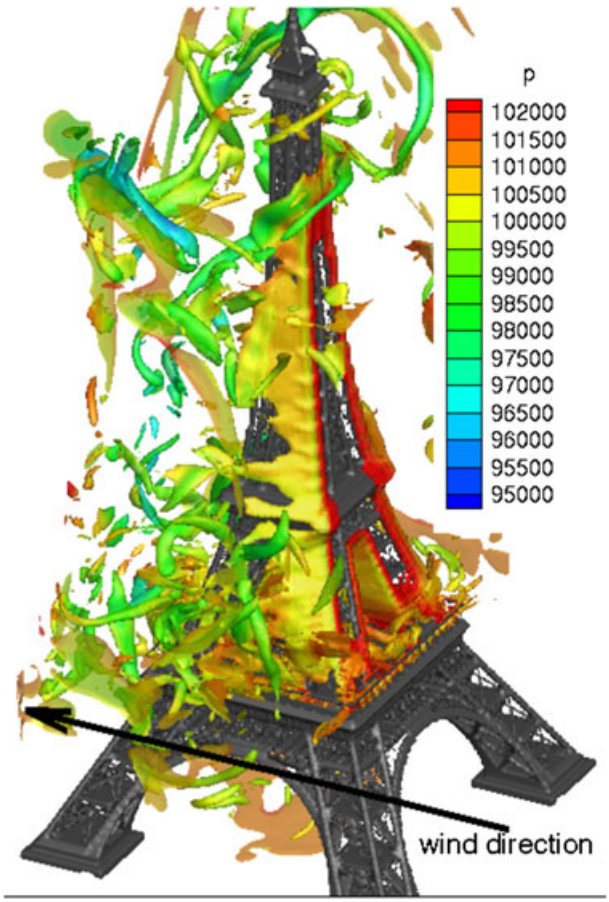


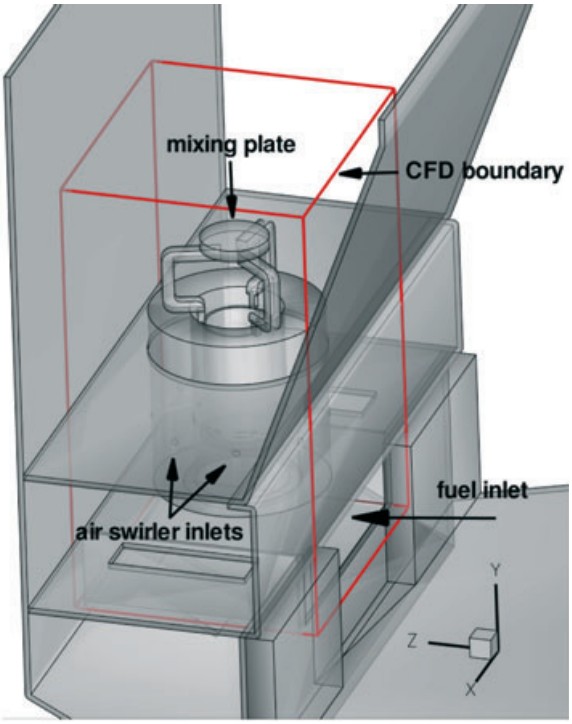
Figure 14. Flow over the Eiffel tower stereolithography model using the (b) $160 \times 320 \times 160$ mesh showing pressure contours on an isosurface of vorticity magnitude. The upstream flow velocity is assumed uniform at 10 m/s.

Copyright © 2016 John Wiley & Sons, Ltd.

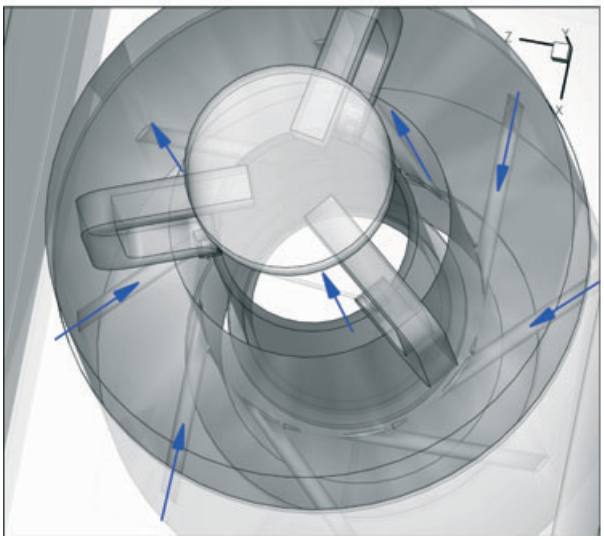
DOI: 10.1002/nme

Doubling the CFD mesh again is presented in Figure 13 using 65 million nodes. The largest lower structures are now nearly resolved, while progressive levels of fill are observed moving up the structure. Even at this finest mesh considered, not all of the structural components are completely resolved; however, with fill support for the stencils, CFD simulations can be conducted in order to understand the flow dynamics around this large structure.

As an example, Figure 14 shows an instantaneous snapshot of a 10 m/s wind gust blowing over the Eiffel tower model using the $160 \times 320 \times 160$ mesh ($\sim$ 8 million nodes). The calculations are



(a)



(b)

Figure 15. Combustion chamber configuration showing (a) secondary combustion chamber and CFD domain (in red) and (b) close up view of swirler section indicating the overall airflow direction through eight swirler tubes and out through a sleeve.

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

run on 128 processors using block structured domain decomposition. Extension of the fill algorithm for parallel computing is straightforward. Each mesh block generates its own mesh tree to initialize the level set. At the end of every iteration of the fill procedure, information is exchanged at the ghost cells. Figure 14 shows pressure contours on an isosurface of vorticity magnitude of $1\,\mathrm{s}^{-1}$. As expected, the pressure on the upwind side of the structure is higher than the downstream. Small scale turbulent flow structures generated by the truss structural members are clearly evident in the downstream. Because the focus of this study is to present the initialization algorithm, no attempt here is made to quantify the predictions. The results are only to highlight the ease in conducting CFD calculations of fairly complex geometry without excessive grid generation or re-processing of the original STL file.

4.3. Combustor swirler application

The last example is an industrial application of a combustion chamber from a two-stage biomass boiler system (manufactured by Thermo-Control industries [38]). Figure 15(a) shows the overall geometry of the secondary combustion chamber. Fuel pyrolysis gases enter from the primary chamber (not fully shown) through the bottom of the secondary chamber and mix with air through a cylindrical swirler unit. Air enters through eight tubes, 8 mm in diameter, into a cylindrical sleeve oriented at an angle to create swirl. The outer diameter of the sleeve measures 7.49 cm, and inner diameter of the passage where the air enters is 10.16 cm in diameter – creating a 1.335 cm circular gap passageway for the air to come up and mix with the fuel flowing through the center of the swirler. Promotion of fuel-air mixing is achieved using a circular impact plate of diameter 7.49 cm and thickness of 1.27 cm that is supported by three brackets. Figure 15(b) shows a close up detail of the impact plate showing the direction of the air entering through the small circular tubes.

An outline of the Cartesian CFD domain is highlighted in red with dimensions $0.32 \times 0.5 \times 0.272$ m in size. The lower y boundary and $x - z$ dimensions of the space are defined to capture the extent of the enclosed lower mixing chamber located underneath the swirler. The height of the space is selected to simulate enough of the near-field dynamics of the mixing process. The simple mesh topology is intentionally selected to eliminate any re-meshing effort with design cycle variations and to better clarify the role of the fill operation for this problem.

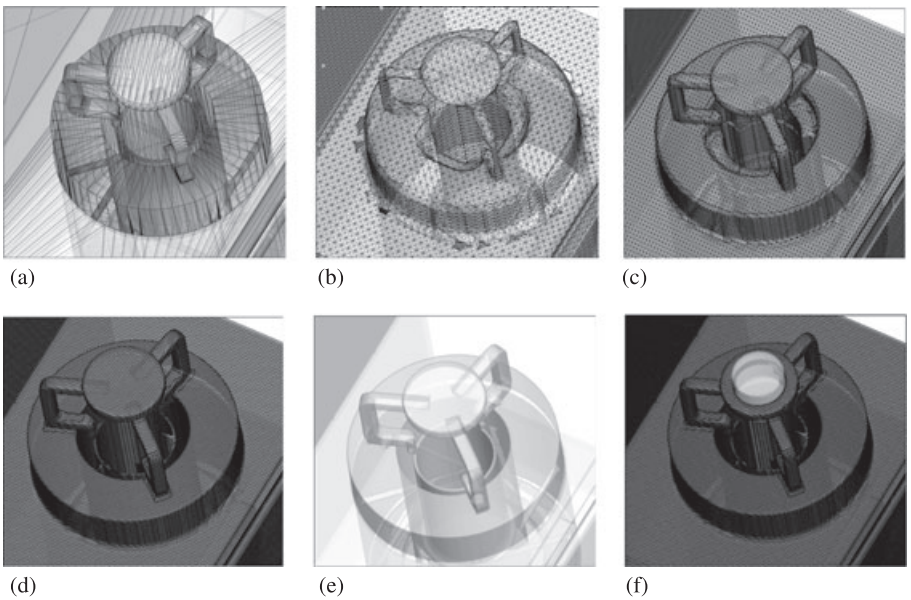
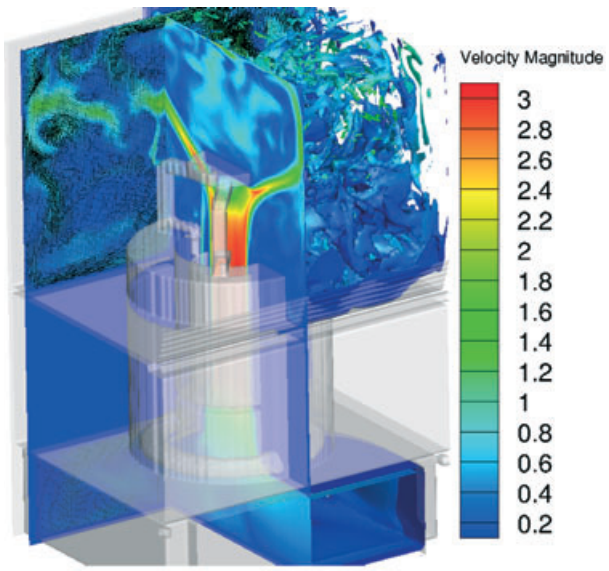


Figure 16. Surface meshes from (a) original stereolithography file, and cut-cell meshes generated using (b) $40 \times 80 \times 40$, (c) $80 \times 160 \times 80$ (d) $120 \times 240 \times 120$ (e) $220 \times 400 \times 220$ meshes and (f) $120 \times 240 \times 120$ with hole cut-out.

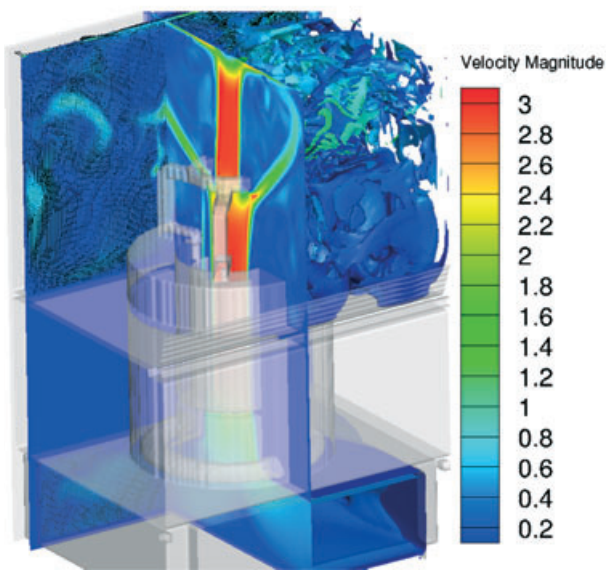
Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

Figure 16(a) shows a close up of the impact region from the original STL file that was extracted from the 3D CAD model using AutoDesk Inventor®. No attempt is made to smooth or re-mesh the STL file generated. While the triangulated surface is continuous, the aspect ratio of the triangles are quite distorted – a common trait of meshes generated from CAD packages. Based on the findings from Section 4.1 for $N = 3$, a minimum value of $R/(N\Delta x) \simeq 2$ is required for the asymptotic convergence – indicating that objects greater than $R > 2N\Delta x$ are captured. Using this approximate criterion then $\Delta x < R/2N = R/6 = 12.5$ mm is required to capture the impact plate, $\Delta x < 2.23$ mm to capture the sleeve spacing and $\Delta x < 1.33$ mm required to capture the small air tubes. Figure 16(b)–(d) show the triangulated surfaces generated from the fill operation for $N = 3$ on CFD meshes of $40 \times 80 \times 40$ (128,000 nodes), $80 \times 160 \times 80$ (1.0 million nodes), $120 \times 240 \times 120$



(a)



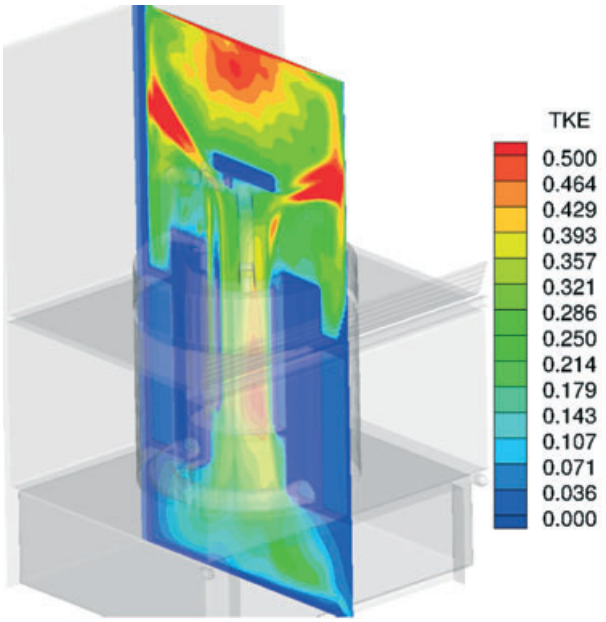
(b)

Figure 17. Instantaneous snapshot of flow through swirler for (a) baseline and (b) modified impact plate configurations. Half of the space shows an isocontour vorticity magnitude at $20\,s^{-1}$ with contours of velocity magnitude superimposed. The other half of the space shows centerline and far-wall slices of velocity magnitude.

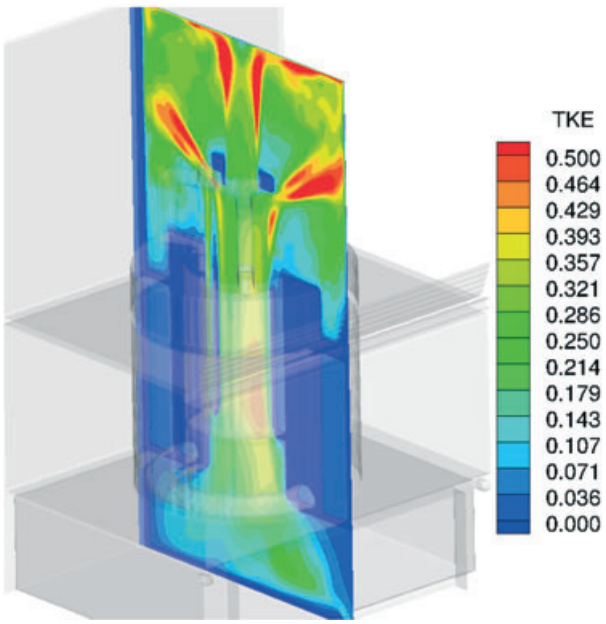
Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

(3.5 million nodes), and $220 \times 400 \times 220$ meshes (19.4 million nodes), corresponding to minimum cell sizes of 7.0, 3.4, 2.3, and 1.2 mm, respectively. Uniform mesh spacing is employed in all cases to simplify the interpretation. For the coarsest case (Figure 16(b)), the minimum Δx is just small enough to capture the impact plate, but all details associated with the sleeve and air holes are lost because of the fill. Additional mesh refinement in Figure 16(c) results in a better representation of the impact plate, but it is not fine enough to pick up the sleeve detail – consistent with *a priori* estimates. Figure 16(d) shows the sleeve region is now captured but the small air tubes are still filled in. With $\Delta x = 1.2$ mm, the air tubes start to be captured, as shown in Figure 16(e).



(a)



(b)

Figure 18. Centerline contour slices of turbulent kinetic energy for (a) baseline and (b) modified impact plate configurations.

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

Large eddy simulation are conducted to explore the effects the impact plate on mixing in the context of a design iteration. Because it is time consuming and tedious to recreate a solid model for each design iteration, the goal of this activity is to show how the fill algorithm can be used in conjunction with component level set merging techniques to readily create new geometry for CFD analysis. A $120 \times 240 \times 120$ mesh is selected to remove the influence of the swirler tubes so only the effects of the impact plate on flow mixing can be studied. To explore the geometry of the circular impact plate, a second level set geometry is introduced describing a cylinder of diameter 2.5 cm and length of 15 cm – centered on the impact plate, as shown in Figure 16(f) (white isosurface). The sign convection of γ for the cylinder, however, is selected to be opposite that generated for the baseline geometry, that is, $\gamma > 0$ in the gas and $\gamma < 0$ in the solid, resulting in a hole in the impact plate when the level sets are merged. The resulting surface mesh generated from the merged level set is shown in Figure 16(f).

Figure 17 shows instantaneous snapshots of the flow using the (a) baseline and (b) modified impact plate configuration with the hole. Half of the space shows an isocontour vorticity magnitude at $20\,\mathrm{s}^{-1}$ with contours of velocity magnitude superimposed. The other half of the space shows centerline and far-wall slices of velocity magnitude. Time averaged statistics are collected by running the simulations over a total of 8 s of physical time with samples taken every 1 ms. Gathering of statistics starts after 1 s of physical time to allow for initial flow transients to wash downstream. A total of 7000 statistics are therefore gathered over approximately 10 flow-through times. Figure 18 shows the time averaged turbulent kinetic energy (TKE) computed along the centerline of the swirler, which is one of the primary metrics for defining the level of local mixing. For both cases, high levels of mixing are observed approximately mid-way down the length of the swirler associated with establishment of a flow recirculation zone. Moving downstream, the location of peak TKE is concentrated along the shear layers shaped by the impact plate. For the baseline case of Figure 18(a), a large flow separation region is observed just above the impact plate along with the high levels of mixing near the exterior of the impact plate. With the modified geometry, the TKE is concentrated near the center associated with the shear layers generated from the edges of the hole.

5. CONCLUSIONS

A new methodology is developed for initialization of high-order accurate immersed interface-based CFD solvers using complex CAD geometry. The algorithm consists of three major steps of (1) mapping STL geometry to level sets and merging, (2) level set augmentation using a fill algorithm, and (3) reconstruction of cut-cell areas and volumes. To improve the computational efficiency of the approach, an adaptive box tree structure algorithm is developed to sort surface elements into boxes using three types of element checks corresponding to elements that are small, large or of the same size as the box. The impact of the fill algorithm on geometry representation is assessed by examining area and volume rates of convergence for simple 2D circles and 3D spheres. Second order asymptotic accuracy is demonstrated for calculation of area and volume for 2D and area in 3D. First order accuracy is observed for calculation of volume in 3D. A criterion is developed for the minimum resolved length scale for which cells will be completely filled that can be used to *a priori* define the desired scale of geometric resolution. A more complex STL benchmark, representing the Eiffel Tower, is examined showing the construction of the box tree and effects of the fill operation for LES . Finally, LES of a swirl chamber from a combustion chamber is examined with the goal of exploring changes in geometry to the impact plate without the tedious step of re-generating STL model files.

ACKNOWLEDGEMENTS

Support for this work has been provided by NAVAIR through the STTR phase II program under contract N68335-10-C-0418 and the New York State Energy Research and Development Authority (NYSERDA) under contract numbers #32966 and #3583. Use of the computer resources at the Center for Computational Research (CCR) at the University at Buffalo is gratefully acknowledged.

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

REFERENCES

1. Hirt CW, Nichols BD. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of Computational Physics* 1981; **29**:201–225.
2. Udaykumar H, Mittal R, Rampunggoon PAK. A sharp interface cartesian grid method for simulating flows with complex moving boundaries. *Journal of Computational Physics* 2001; **174**:345–380.
3. Sethian JA. *Level Set Methods and Fast Marching Methods*. Cambridge University Press: Cambridge, UK, 1999.
4. Osher S, Fedkiw R. *Level Set Methods and Dynamic Implicit Surfaces*. Springer: New York, 2003.
5. Peskin CS. Flow patterns around heart valves: a numerical method. *Journal of Computational Physics* 1972; **10**(Copyright 1973, IEE):252–271.
6. Tseng Y, Ferziger JH. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics* 2003; **192**:593–623.
7. Lacanette D, Vincent S, Sarthou A, Malaurent P, Caltagirone JP. An eulerian/lagrangian method for the numerical simulation of incompressible convection flows interacting with complex obstacles: Application to the natural convection in the lascaux cave. *International Journal of Heat and Mass Transfer* 2009; **52**(Compendex):2528–2542.
8. Liu TG, Khoo BC, Yeo YS. Ghost fluid method for strong shock impacting on material interface. *Journal of Computational Physics* 2003; **190**(Copyright 2004, IEE):651–81. Double Check Ref.
9. Wag C, Tang H, Liu T. An adaptive fhost fluid finite volume method for compressible gas-water simulations. *Journal of Computational Physics* 2008; **227**:6385–6409.
10. Farhat C, Rallu A, Shankaran S. A higher-order generalized ghost fluid method for the poor for the three-dimensional two-phase flow computation of underwater implosions. *Journal of Computational Physics* 2008; **227**:7674–7700.
11. Farhat C, Rallu A, Wang K, Belytschko T. Robust and provably second-order explicit–explicit and implicit–explicit staggered time-integrators for highly non-linear compressible fluid–structure interaction problems. *International Journal for Numerical Methods in Engineering* 2010; **84**(1):73–107.
12. Mittal R, Iaccarino G. Immersed boundary methods. *Annual Review of Fluid Mechanics* 2005; **37**:239–261.
13. Wang K, Grétarsson J, Main A, Farhat C. Computational algorithms for tracking dynamic fluid-structure interfaces in embedded boundary methods. *International Journal for Numerical Methods in Fluids* 2012; **70**:515–535.
14. Bruchon J, Digonnet H, Coupez T. Using a signed distance function for the simulation of metal forming processes: Formulation of the contact condition and mesh adaptation. from a lagrangian approach to an eulerian approach. *International Journal for Numerical Methods in Engineering* 2009; **78**:980–1008.
15. Wang F, Rallu A, Gerbeau JF, Farhat C. Algorithms for interface treatment and load computation in embedded boundary methods for fluid and fluid-structure interaction problems. *International Journal for Numerical Methods in Fluids* 2011; **67**:1175–1206.
16. Günther C, Meinke M, Schröder W. A flexible level-set approach for tracking multiple interacting interfaces in embedded boundary methods. *Computers & Fluids* 2014; **102**:182–202.
17. Jones MW. 3d distance from a point to a triangle. *Technical Report*, University of Wales Swansea Technical Report CSR-5 1995: Swansea, Wales.
18. Samet H. *Multidimensional Data Structures for Spatial Applications*, chap. 6. CRC Press: Boca Raton, FL, 2009.
19. Towers JD. Discretizing delta functions via finite differences and gradient normalization. *Journal of Computational Physics* 2009; **228**(10):3816–3836.
20. Towers JD. Finite difference methods for appoximating heaviside functions. *Journal of Computational Physics* 2009; **228**(9):3478–3489.
21. DesJardin PE, O'Hern TJ, Tieszen SR. Large eddy simulation and experimental measurements of the near-field of a large turbulent helium plume. *Physics of Fluids* 2004; **16**:1866–1883.
22. Pope S. *Turbulent Flows*. Cambridge University Press: Cambridge, UK, 2000.
23. Poinsot TDV. *Theoretical and Numerical Combustion*. R.T. Edwards, Inc.: Flourtown, PA, USA, 2005.
24. Liou MS. A sequel to ausm: Ausm+. *Journal of Computational Physics* 1996; **129**:364–382.
25. Liou MS. A sequel to ausm, part ii: Ausm+up for all speeds. *Journal of Computational Physics* 2006; **214**(1):137–70.
26. Li Y. Wavenumber-extended high-order upwind-biased finite-difference schemes for convective scalar transport. *Journal of Computational Physics* 1997; **133**:235–255.
27. Shu CW, Osher S. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics* 1988; **77**:364–382.
28. van Leer B. Towards the ultimate conservative difference scheme. *Journal of Computational Physics* 1979; **39**: 101–136.
29. van Leer B. Towards the ultimate conservative difference scheme v. a second order sequal to godunov's method. *Journal of Computational Physics* 1979; **32**:101–136.
30. Mathur S, Tondon P, Saxena S. Thermal conductivity of binary, ternary and quaternary mixtures of rare gases. *Molecular Physics* 1967; **12**:569–579.
31. Bird R, Stewart W, Lightfoot E. *Transport Phenomena*. John Wiley and Sons: New York, NY, 1960.
32. DesJardin PE. Modeling of conditional dissipation rate for flamelet models with application to large eddy simulation of fire plumes. *Combustion Science and Technology* 2005; **177**:1881–1914.
33. McGurn M, Ruggirello K, DesJardin PE. An eulerian–lagrangian moving immersed interface method for simulating burning solids. *Journal of Computational Physics* 2013; **241**:364–387.
34. Ruggirello KP, DesJardin P, Baer MR, Hertel ES. Modeling of particle compressibility and ignition from shock-shock focusing. *Combustion Theory and Modelling* 2010; **14**:41–67.

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

35. Poinsot TJ, Lele SK. Boundary conditions for direct simulations of compressible viscous flows. *Journal of Computational Physics* 1992; **101**:104–129.
36. Thompson KW. Time-dependent boundary conditions for hyperbolic systems, ii. *Journal of Computational Physics* 1990; **89**:439–481.
37. Panchal P. Eiffel tower. (Available from: https://grabcad.com/library/eiffel-tower) [Accessed on 11 October 2015].
38. LTD TCHS. Thermo-control. (Available from: http://www.nationalstoveworks.com/) [Accessed on 11 October 2015].

Copyright © 2016 John Wiley & Sons, Ltd.

DOI: 10.1002/nme