Chapter 15
Immersed-Boundary Methods for Simulating Human Motion Events

Jung-Il Choi and Jack R. Edwards

15.1 Introduction

Immersed-boundary methods are a general class of technique that indirectly imposes the effects of a (possibly moving) solid surface on the surrounding flow. While the original immersed-boundary method dates from the work of Peskin (1972), the technique was recast into a form more useful for conventional CFD strategies by Mohd-Yosuf (1997), Verzicco et al. (2000), Fadlun et al. (2000), and others. A review article summarizing these and other techniques is that of Mittal and Iaccarino (2005). A key to these newer immersed-boundary methods is the enforcement of fluid boundary conditions indirectly, through specification of the distribution of the fluid velocity in the vicinity of the immersed boundary. This paper presents a generalization of an immersed-boundary method developed for time-dependent, incompressible flows in Choi et al. (2007). This approach is similar to that of Gilmanov et al. (2003) in that a surface mesh consisting of structured or unstructured elements is embedded within a flow and that flow property variations normal to the surface are reconstructed. The surface meshes may be closed (surrounding a volume of space) or zero-thickness (surfaces alone). The Navier–Stokes equations are solved in cells outside the body (field cells); a constant property condition is enforced for cells inside the body (interior cells); and boundary conditions are enforced through specifying distributions of fluid properties in a collection of band cells just outside the immersed body (band cells). In contrast to many other IB techniques, the methods developed in Choi et al. (2007) can be applied to turbulent flows at high Reynolds numbers by virtue of the use

J.-I. Choi
Department of Computational Science and Engineering, Yonsei University, Seoul 03722, Republic of Korea
e-mail: jic@yonsei.ac.kr

J. R. Edwards (✉)
Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, NC 27695, USA
e-mail: jredward@ncsu.edu

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_15

of power-law interpolation techniques to mimic the near-wall profile of an attached turbulent flow. The methods are also applicable to general curvilinear meshes as well as unstructured meshes. Since the publication of Choi et al. (2007), extensions to particle-laden incompressible flows (Oberoi et al. 2010; Choi et al. 2012), gas-phase contaminant transport (Choi and Edwards 2008, 2012), and compressible, turbulent flows (Ghosh et al. 2010a, b, 2012) have been developed. All of these studies have rendered immersed objects as point clouds, which has advantages if the object is sufficiently detailed but becomes inconvenient if the object is relatively featureless. This report outlines a way of embedding stereo-lithography (STL) files as immersed objects within a computational domain and introduces several techniques for converting scenarios involving complicated and possibly moving objects into detailed large-eddy flow simulations driven by immersed-boundary motion. The presented applications involve realistic human motion activity as well as secondary effects such as buoyancy-driven flow resulting from the human thermal plume.

15.2 Numerical Methods

15.2.1 Governing Equations

For a three-dimensional, time-dependent incompressible flow, the grid-filtered governing equations for a fluid phase can be written as

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0, \tag{15.1}$$

$$\frac{\partial \rho \bar{u}_i}{\partial t} + \frac{\partial}{\partial x_j}(\rho \bar{u}_i \bar{u}_j + \bar{p}\delta_{ij} - \overline{\tau}_{ij} + \tau_{ij}^{\mathrm{SGS}}) = \bar{f}_i, \tag{15.2}$$

where $\bar{u}_i$ is the velocity vector, ρ is the density of the fluid, $\bar{p}$ is the pressure, $\bar{f}_i$ is an external force, μ is the molecular viscosity, $\overline{\tau}_{ij}$ is the viscous stress tensor for a Newtonian fluid, and τ_{ij}^{SGS} is the subgrid-scale (SGS) stress tensor. Note that the overbar represents grid-filtered variables. Based on the Smagorinsky model (Smagorinsky 1963), which assumes that the SGS stress tensor is proportional to the velocity strain rate $\overline{S}_{ij}$, the SGS stress tensor is modeled as $\tau_{ij}^{\mathrm{SGS}} = -2\mu_t \overline{S}_{ij}$. The subgrid-scale eddy viscosity is defined as $\mu_t = \rho(C_s\Delta)^2(2\overline{S}_{ij}\overline{S}_{ij})^{1/2}$, where $C_s(= 0.1)$ is the Smagorinsky constant and Δ is a local grid-filter width, which is set equal to the cube root of the mesh-cell volume.

The mass conservation equations for transport of a set of passive gaseous contaminants in Eulerian framework (Crowe et al. 1996) are as follows:

$$\frac{\partial \overline{\rho}_k}{\partial t} + \frac{\partial}{\partial x_j}(\overline{\rho}_k(\bar{u}_i + \bar{v}_{k,j})) = 0, \tag{15.3}$$

where the subscript k denotes the kth gas species. Here, $\overline{\rho}_k$ is the mass density of species k. The diffusion velocity $\tilde{v}_{k,i}$ is given by Fick's law:

$$\bar{v}_{k,i} = -\left(\frac{\mu}{Sc} + \frac{\mu_t}{Sc_t}\right)\frac{1}{\bar{\rho}_k}\frac{\partial \overline{Y}_k}{\partial x_i}, \tag{15.4}$$

where the mass fraction $\overline{Y}_k = \overline{\rho}_k/\rho$ and the laminar and turbulent Schmidt numbers are assigned values of 0.72 and 1.0, respectively (Crowe et al. 1996). It is assumed that the mass fractions of the tracer-gas species are small enough that the density of the carrier gas is not affected significantly. We extrapolate contaminant concentration to all physical surfaces; the contaminant concentration is set to zero inside all immersed surfaces.

Under incompressible flow assumptions, the evolution of temperature θ can be written as:

$$\rho C_p\left(\frac{\partial\overline{\theta}}{\partial t} + \bar{u}_j\frac{\partial\overline{\theta}}{\partial x_j}\right) = \frac{\partial}{\partial x_j}\left((\alpha+\alpha_t)\frac{\partial\overline{\theta}}{\partial x_j}\right) + \dot{Q}, \tag{15.5}$$

where $\overline{\theta}$ is temperature, C_p is specific heat capacity at constant pressure, α is the thermal conductivity, α_t is the turbulent thermal conductivity, and $\dot{Q}$ is an external heat source. The thermal conductivities α and α_t are related to the molecular and eddy viscosities through the assumption of constant laminar and turbulent Prandtl numbers (0.72 and 0.9, respectively). Equation (15.5) is solved subject to isothermal, adiabatic, or imposed heat-flux boundary conditions at solid surfaces. Buoyancy effects resulting from temperature gradients are imposed in Eq. (15.2) using the Boussinesq approximation: $f_i = \rho_\infty g_i(1 - \overline{\theta}/\theta_\infty)$, where g_i is the gravitational force and the subscript $_\infty$ denotes the undisturbed-flow state.

Basic formulation We solve the three-dimensional incompressible Navier–Stokes equations using a finite volume approach. Time integration of the discrete Navier–Stokes equations is achieved by an artificial compressibility approach (Chorin 1967) which is facilitated by a dual time-stepping procedure at each physical time step. At time level n + 1, sub-iteration k, the solution of the discrete representation of Eqs. (15.1) and (15.2) can be written as

$$\mathbf{A}(\mathbf{V}^{n+1,k+1} - \mathbf{V}^{n+1,k}) = -\mathbf{R}^{n+1,k}. \tag{15.6}$$

The flow variables $\mathbf{V} = (\bar{p}, \bar{u}_i)^T$ are advanced from time level n ($\mathbf{V}^{n+1,k=0} = \mathbf{V}^n$) to time level n + 1 ($\mathbf{V}^{n+1} = \mathbf{V}^{n+1,k=k_{max}}$) over a number of sub-iterations $k_{\max}$. The system Jacobian matrix is denoted as $\mathbf{A}$, and the corresponding residual vectors $\mathbf{R} = (R_c, R_{M_i})^T$ can be written as

$$R_c^{n+1,k} = \left[\frac{\partial \bar{u}_i}{\partial x_i}\right]^{n+1,k} \tag{15.7}$$

$$R_{M_i}^{n+1,k} = \rho\left(\frac{3\bar{u}_i^{n+1,k} - 4\bar{u}_i^n + \bar{u}_i^{n-1}}{\Delta t}\right) + \left[\frac{\partial}{\partial x_j}(\rho\bar{u}_i\bar{u}_j + \bar{p}\delta_{ij}\right.$$
$$\left. -\overline{\tau}_{ij} + \tau_{ij}^{\mathrm{SGS}}) - \rho\bar{f}_i\right]^{n+1,k}. \tag{15.8}$$

Equation (15.8) is solved approximately at each sub-iteration using an implicit technique based on incomplete LU decomposition (Wesseling 1995). For the spatial discretization, the inviscid fluxes in the governing equations are discretized using a low-diffusion flux-splitting scheme (LDFSS) (Edwards and Liou 1998; Neaves and Edwards 2006), while second-order central differencing methods are used to discretize the viscous components. For the cases presented later, higher-order spatial accuracy for the interface fluxes is achieved by using the piecewise parabolic method (Colella and Woodward 1984). The effects of smaller subgrid fluctuations are modeled using a Smagorinsky subgrid eddy viscosity (Baurle et al. 2003). The present flow solver uses METIS (Karypis and Kumar 1998) to partition a general multi-block grid over the number of allowable processors. Message-passing interface (MPI) communication routines are used to pass information among the processors. The incompressible flow solver and its components have been validated for a range of model problems (Edwards and Liou 1998).

15.2.2 Cell-Classification Procedure

We develop a classification algorithm for computational nodes based on the signed distance function $\Phi(\mathbf{x}, t)$, which is less than zero for cells within a closed immersed body and greater than zero for cells outside the body. Special procedures discussed later are used to handle zero-thickness immersed surfaces for which the signed distance is always positive.

Classification of computational cells The Heaviside function $G(\Phi(\mathbf{x}, t))$ is defined to be one for points just outside the immersed body and within the immersed body and is zero otherwise. The calculation of the Heaviside function is initiated by first initializing $G(\Phi(\mathbf{x}_k, t)) = 0$ for all points $\mathbf{x}_k$. Then, given a point $\mathbf{x}_k$, if $\Phi(\mathbf{x}_k, t) > 0$ and if any $\Phi(\mathbf{x}_m, t) < 0$, where $\mathbf{x}_m$ is a face, edge, or vertex neighbor of $\mathbf{x}_k$, then $G(\Phi(\mathbf{x}_k, t))$ is set to 1. If $\Phi(\mathbf{x}_k, t) \leq 0$, then $G(\Phi(\mathbf{x}_k, t))$ is also set to 1. The set of nearest neighbors, for a structured grid discretized according to a cell-centered finite volume method, is generally defined as the 26 cells that are immediately adjacent to a particular mesh cell, though smaller subsets can be used. Finally, we can define the Heaviside function as

$$G(\Phi(\mathbf{x}_k, t)) = \begin{cases} 0 \text{ for } \mathbf{x}_k \in \Omega_F \\ 1 \text{ for } \mathbf{x}_k \notin \Omega_F \end{cases}, \tag{15.9}$$

where Ω_F represents the set of the node points shown as the open circles in Fig. 15.1.

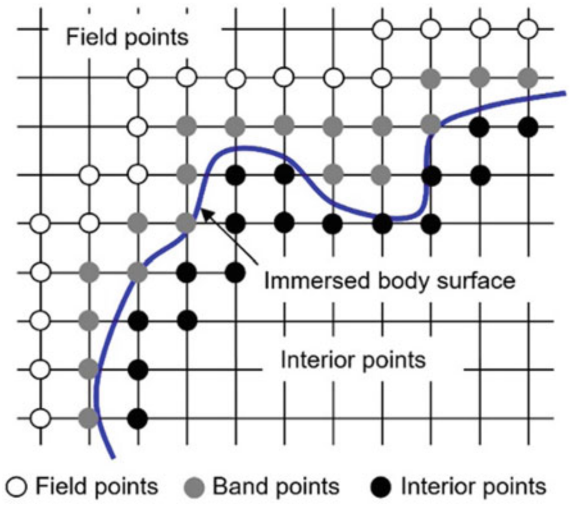

Fig. 15.1 Schematic illustrating classification of cell-centered points for a complex immersed body surface. Open, gray, and close circles represent field (Ω_F), band (Ω_B) and interior points (Ω_I), respectively, and thick line represents an immersed body surface. Adapted from (Choi et al. 2007)

The classification of the node points can be summarized as follows:

- Field points: $\mathbf{x}_k \in \Omega_F$ if $\Phi(\mathbf{x}_k, t) > 0$ and $G(\Phi) = 0$,
- Band points: $\mathbf{x}_k \in \Omega_B$ if $\Phi(\mathbf{x}_k, t) > 0$ and $G(\Phi) = 1$,
- Interior points: $\mathbf{x}_k \in \Omega_I$ if $\Phi(\mathbf{x}_k, t) \leq 0$ and $G(\Phi) = 1$.

where Ω_B and Ω_I represent the set of the node points shown as the gray and closed circles in Fig. 15.1, respectively. The zero iso-surface of the signed distance function defines the immersed body surface.

Surface definition in a computational domain The most popular way to describe 3D objects in computer system is to construct surface meshes composed of triangular elements (henceforth referred to as triangle meshes). This can be done using a computer-aided design (CAD) format or through other means, but the key is that triangle elements with an outward-pointing normal vector are created for each separate component of the object, as different components may move at different rates. The next step is to define 3D surfaces using the *unsigned* distance and classification whether an arbitrary point in a background domain is inside or outside of the objects. Classification can be achieved by counting intersections of a ray going from the given point (outside point from the object) to infinity since the number of intersections must be odd if the point is inside—this is called a *ray tracing method* (Linhart 1990). Another means of classification is to define a signed distance using the inner product between a pseudo-normal vector and a distance vector to an arbitrary point from its closest point on the surface—this is known as a *signed distance computation* (Gouraud 1971; Bærentzen and Aanæs 2005). While the former method needs to visit the parts of the triangle mesh along the ray tracing line, the latter algorithm needs to find the closest point on the mesh. We will apply the signed distance computation which is faster than ray tracing method in order to define 3D surfaces which will be incorporated with the present immersed-boundary method.

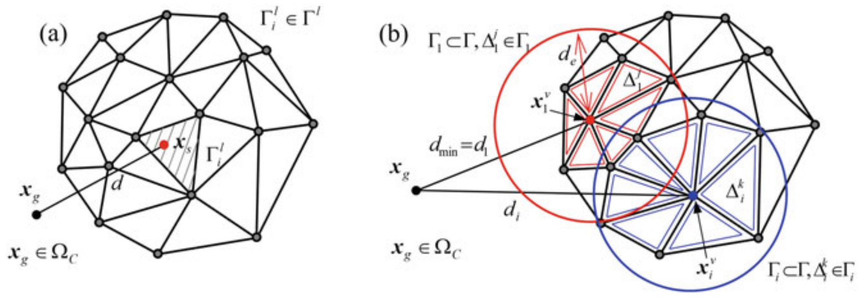

Fig. 15.2 Schematics for **a** a minimal distance from the points in a computation domain to surface points and **b** nearest neighbors for triangle elements. Adapted from (Edwards et al. 2010)

The distance d from grid points $\mathbf{x}_g$ in a computational domain Ω_C to the closest surface point $\mathbf{x}_s$ on triangle meshes Γ^l for lth component is simply defined as $d = \|\mathbf{x}_g - \mathbf{x}_s\|$ in Fig. 15.2a. Computation of the distance to 3D objects can be achieved by using brute force computation, a Voronoi diagram (Hoff et al. 1999), or hierarchical data structures (Payne and Toga 1992; Guéziec 2001). Among these methods, we use a k-d tree hierarchical data structure with a bounding box to accelerate finding the nearest triangle mesh element. For simplicity, we consider the one component's closed surface as shown in Fig. 15.2b. At first, we find a cloud of nearby points $\mathbf{x}_i^v$ from the given point $\mathbf{x}_g$ in a bounding box, in order of the closest distance, using an approximate nearest-neighbor (ANN) searching algorithm (Arya et al. 1998). The next step is to search the closest point in the set of the neighbor triangle meshes $\Delta_i^j \in \Gamma_i$ which are shared with a cloud of nearby vertices $\mathbf{x}_i^v$ since the closest vertex is typically different from the closest point on a triangle mesh. We can define the subset $\Gamma_s = \{\Gamma_i\}$ of the total triangle meshes Γ. Based on the subset Γ_s, the minimum distance can be obtained using point-triangle, point-edge, and point-vertex distance calculations.

In the search process, the subset Γ_s can be reduced using geometric restriction. Modern CAD programs enhance the uniformity of the triangles and control the edge distances. At a given edge distance d_e, we can get a restriction for the searching algorithm. As shown in Fig. 15.2b, the circles show the spheres with radius d_e and origin $\mathbf{x}_i^v$. The entire triangle neighbors Δ_i^j shared with the vertex $\mathbf{x}_i^v$ are included within the spheres. The distances d_i^j in the subset Γ_i are bounded as $|d_i^k - d_i| \leq d_e$ with respect to the point-vertex distance d_i. Also, the distance is $|d_1^j - d_1| \leq d_e$ for the subset Γ_1 which is the equivalent subset for the minimum point-vertex distance. The difference between two point-vertex distances can be written as $d_i^j - d_1^k - 2d_e < d_i - d_1 < d_i^j - d_1^k + 2d_e$. If $d_i^j < d_1^k$, the difference should be bounded as $d_i - d_1 < 2d_e$. Therefore, the above nearest distance calculation should be repeated for the ith nearest vertex point in the ANN list which satisfies $d_i - d_1 < 2d_e$.

Signed distance computation The signed distance function Φ can be obtained by multiplying the unsigned distance d with the sign of the dot product of the distance vector with the outward normal vector $\mathbf{n}$:

$$\Phi = \mathrm{sgn}((\mathbf{x}_g - \mathbf{x}_s) \cdot \mathbf{n})\, d, \tag{15.10}$$

where $\mathrm{sgn}(\varphi)$ returns a value of 1 for each nonnegative element and -1 for each negative element of φ and $\| \ \|$ denotes the magnitude of the vector.

This simple procedure was found not to work properly for some very complex CAD objects (Choi et al. 2007). Usually, the CAD objects are defined as triangular surface elements that contain each vertex and face-normal vector. If a nearest surface point at a given field point is located on an edge or at a vertex, the simple signed distance function may not be calculated correctly. Therefore, we consider an angle-weighted pseudo-normal vector (Bærentzen and Aanæs 2005), which is defined at surface nodes (vertices) or edges, rather than cell centers of surface triangles. For a given vertex $\mathbf{x}_v$, we identify the triangle elements shared with the vertex and calculate the incident angle α_i for each element with the outward-pointing face-normal vector $\mathbf{n}_i$ (Choi et al. 2007). The angle-weighted pseudo-normal vector $\mathbf{n}_v$ at the vertex can be defined as

$$\mathbf{n}_v = \frac{\sum_i \alpha_i \mathbf{n}_i}{\left\| \sum_i \alpha_i \mathbf{n}_i \right\|}, \tag{15.11}$$

where i denotes the triangle elements that surround the vertex and $\| \ \|$ denotes the magnitude of the vector. Based on the pseudo-normal vector at the vertex and face-normal vector $\mathbf{n}_i$ at the element center $\mathbf{x}_i$, we can determine an inside/outside decision using the same signed distance function in Eq. (15.10) with the data set of the vertices. This procedure essentially averages local fluctuations in the outward normal that could result from small features in the CAD file.

To define a global signed distance function Φ at any given mesh point, a simple priority rule is exercised. First, the global distance function is initialized to a large number. Then, the global signed distance function at a particular point is taken as the minimum of the individual signed distance functions for each component l at that point:

$$\Phi = \min_l(\Phi_l). \tag{15.12}$$

The collections of points that comprise the surfaces are allowed to move according to prescribed rate laws.

Embedding of CAD objects as immersed surfaces One of our major goals is to be able to incorporate general stereo-lithography (STL) files as immersed objects in our program without any additional user intervention. As discussed earlier, the main challenge is in accurately computing the signed distance from any of our mesh points to the nearest point on the STL surface. This challenge is made more difficult for objects that contain large, flat panels (usually rendered as two triangles) and smaller features that are more refined. In our earlier work, we simply mesh-refined the objects until a clear rendering was achieved, but this required significant pre-processing and led to STL files that could be very large (millions of cells). In this work, we develop a module for directly reading STL files and for computing nearest distances and normal vectors to any panel, edge, or node on the surface. A detailed step-by-step procedure is as follows:

Step 1: Import STL file (ASCII format) and determine element-to-element connectivity. The STL file provides coordinates of vertices of each triangle along with the normal vector associated with the face center of each triangle.
Step 2: Calculate, for every triangle, coordinates of the face center and the midpoint of each edge, pseudo-normal vectors at each vertex, and normal vectors at the midpoint of each edge.
Step 3: Add these additional coordinates/normal vectors to the database.
Step 4: Given a particular field point, use approximate nearest-neighbor (ANN) searching (Arya et al. 1998) to determine a set of nearest vertices to that point.
Step 5: Determine whether the true nearest point to the surface lies on a triangle, at a vertex, or on an edge.
Step 6: Based on this decision, find the nearest point and assign the appropriate normal vector (face-centered, pseudo-normal, or edge-centered) to this point. Calculate signed distance functions at each query cell.

The search for an initial subset of possible vertices is an N log(m) operation, so this approach is still relatively efficient. Note that, N and m are the number of query points and the listed data points (possible vertices), respectively. The case of very large triangles neighboring small triangles, however, can require expanding the initial subset decision space to include all possible triangles, leading to a complexity of $O(Nm)$.

For moving objects, we initially read ASCII-formatted STL files for the objects at each time step. This reading sequence required non-trivial I/O access times compared to the entire computation. Thus, we developed an improved STL reader to accelerate the reading sequence using binary formatted STL files as well as avoiding redundant procedures. The current status of the reader is that it is able to read directly immersed objects from STL files (ASCII or binary format) and can separate automatically multiple objects in STL files into smaller segments.

Figure 15.3 shows the original avatar rendering in 3DSMax®, the surface triangulation, and the rendered image as an immersed body in the computational domain. The avatar consists of four segments such as body, head, hat, and gun. In order to make a closed immersed surface for the soldier, we merged the body, head, and hat into a single object. Note that, we maintain the skinning in the merging procedure for the original biped motion.

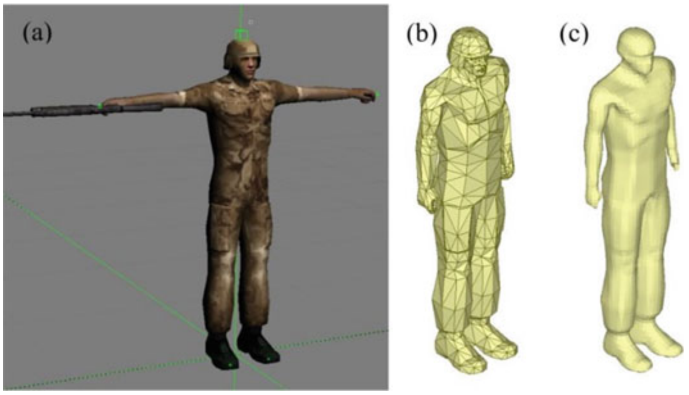

Fig. 15.3 Soldier avatar: **a** 3DSMax® rendering, **b** triangle elements in a STL file, and **c** immersed object rendering in a computational domain

15.2.3 Immersed-Boundary Formulation

Given the classification of the computational domain into field, band, and interior cells as described above, a direct forcing approach is used to enforce the boundary conditions at the interior and band cells. This results in the residual form of the governing equation system shown below which is then solved implicitly, coupled with exterior cells, by use of sub-iteration techniques:

$$\begin{aligned}\widetilde{R}_i^{n+1,k} &= (1 - G(\Phi^{n+1}))R_i^{n+1,k} \\ &\quad + G(\Phi^{n+1})\left[\frac{V_i^{n+1,k} - V_{B,i}^{n+1,k}}{\Delta t}\right], \quad i = c, M_x, M_y, M_z. \end{aligned} \tag{15.13}$$

This equation represents the blending of the Navier–Stokes residual with a source term that relaxes the primitive variable vector $\mathbf{V} = (\bar{p}, \bar{u}_i)^T$ to its band-cell values. As discussed earlier, other equations representing transport of species concentration and heat may be added to this system.

Determination of information at the interpolation point The developments follow hinge on the determination of flow properties $q(d_I)$ at a certain distance d_I away from the surface (see Fig. 15.4). Given a point within the band $\mathbf{x}_k$ and a list of nearest neighbors to that point $\mathbf{x}_l$, a merit function w_l is defined as

$$w_l = \frac{1}{\sqrt{(|\mathbf{x}_l - \mathbf{x}_k|)^2 - ((\mathbf{x}_l - \mathbf{x}_k)\cdot\mathbf{n})^2 + \varepsilon}} \text{ for } (\mathbf{x}_l - \mathbf{x}_k)\cdot\mathbf{n} > 0, \tag{15.14}$$

otherwise $w_l = 0$.

In this, $(\mathbf{x}_l - \mathbf{x}_k)\cdot\mathbf{n}$ is the projection of the distance from $\mathbf{x}_k$ to $\mathbf{x}_l$ in the direction of the outward normal, and $\|\mathbf{x}_l - \mathbf{x}_k\|$ is the magnitude of the distance vector itself.

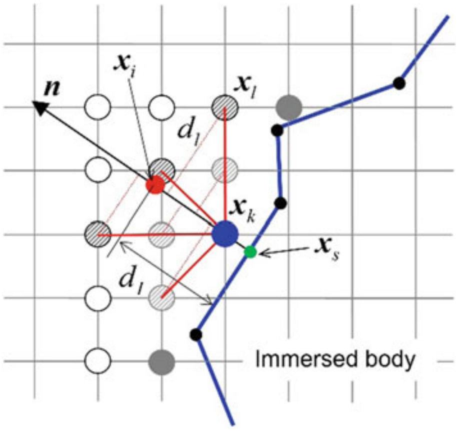

Fig. 15.4 Schematic determination of the distance d_I between the interpolation point $\mathbf{x}_i$ and surface node point for a given band point $\mathbf{x}_k$ using the projected distance d_I from neighbor points $\mathbf{x}_l$ to outward normal line based on surface-normal vector $\mathbf{n}$ at the immersed surface points $\mathbf{x}_s$ marked by green circle. Large closed circle represents the band point to be interpolated with the information at neighbor point. Hatched black and gray circles represent the field points and band points associated with the present determination, respectively

If point $\mathbf{x}_l$ is located directly along the outward normal line corresponding to band point $\mathbf{x}_k$, and if $(\mathbf{x}_l - \mathbf{x}_k) \cdot \mathbf{n}$ is positive, meaning that point $\mathbf{x}_l$ is further away from the surface than point $\mathbf{x}_k$, then the merit function returns a very large value (~$1/\varepsilon$, where ε is 10^{-12}).

The actual calculation of w_l is performed in three stages. First, only field points (those with $\Phi(\mathbf{x}_l, t) > 0$ and $G(\Phi(\mathbf{x}_l, t)) = 0$) are considered as members of the list of nearest neighbors. Then, w_l is calculated according to Eq. (15.14), and the sum of the weights $\sum_m w_m$ is calculated. If this sum is nonzero, then the actual weight function for each nearest neighbor is determined as

$$\omega_l = \frac{w_l}{\sum\limits_m w_m}. \tag{15.15}$$

Otherwise, the process is repeated, now considering both field points and other band points as members of the list of nearest neighbors. If this application also results in no viable interpolation points being found, then the band point $\mathbf{x}_k$ is effectively set to an interior point.

The location at which interpolated properties are defined, d_I, is calculated for a particular field point as

$$d_I = \sum_l \omega_l (\mathbf{x}_l - \mathbf{x}_k) \cdot \mathbf{n}. \tag{15.16}$$

Note that, this distance is in the direction of the normal coordinate. With this, the fluid properties $q(d_I)$ are found by applying the weighting functions,

$$q(d_I) = \sum_m q_m \omega_m. \tag{15.17}$$

Variable reconstruction in band cells The following closures are used for the fluid properties in the band cells, where the subscript 'I' indicates properties obtained at an interpolation point located along the normal line extending outward from the nearest surface location corresponding to the band cell in question, and the subscript 'B' indicates the band cell.

$$\begin{aligned} &p_B = p(d_I) \\ &u_{B,i} - u_{S,i} = u_{T,i}(d_I)\left(d_B \big/ d_I\right)^k + u_{N,i}(d_I) f_N(d_I, d_B), \\ &u_{N,i}(d_I) = (u_j(d_I) - u_{S,j})\mathbf{n}_j \mathbf{n}_i, \\ &u_{T,i}(d_I) = (u_i(d_I) - u_{S,i}) - u_{N,i}(d_I) \end{aligned} \tag{15.18}$$

In these expressions, $\mathbf{n}$ is the normal vector at the closest point on the body surface, d is a distance from the nearest surface point, $u_{S,j}$ is the velocity at the nearest surface point, and k is a power-law. The choice of k allows the model to replicate a turbulent velocity profile ($k = 1/7$ or $1/9$) or a laminar profile ($k = 1$). To obtain the temperature distribution near the surface, the following expressions are utilized. These are a low Mach-number simplification of more general relations derived from Walz's formula (Walz 1969):

Isothermal wall:

$$\frac{T_B}{T(d_I)} = \frac{T_w}{T(d_I)} + \left(1 - \frac{T_w}{T(d_I)}\right)\left(\frac{d_B}{d_I}\right)^k \tag{15.19}$$

Adiabatic wall:

$$\frac{T_B}{T(d_I)} = 1 \tag{15.20}$$

The function $f_N(d_I, d_B)$ that scales the normal velocity component in Eq. (15.18) is determined by enforcing a discrete form of the continuity equation at each band cell using a locally parallel flow assumption. A general formulation suitable for compressible flows is given in Ghosh et al. (2010); here, a simpler form suitable for constant-density flows is presented.

$$\begin{aligned} &f_N(d_I, d_B) = \frac{\left(d_B \big/ d_I\right) d^-}{\left(d_B \big/ d_I\right) d^- + \left(1 - d_B \big/ d_I\right) d^+}, \\ &d^- = \left(d_B \big/ 2 d_I\right)^k \text{ and } d^+ = \left(1 + d_B \big/ d_I\right)^k \Big/ 2^k. \end{aligned} \tag{15.21}$$

Note that, this procedure does not rigorously enforce mass conservation within the band cells, as the integral form of the continuity equation is not used. If precise mass conservation is required, the pressure interpolation in Eq. (15.18) can be replaced by the solution of the continuity equation in the band cells. This, however, can lead to oscillations within the band cells, and for some of the moving-body applications presented later, a hybrid approach is utilized. Given that $R_{c,\text{orig}}$ is the initial residual of the continuity equation within a band cell, a modified residual is defined as

$$R_{c,\text{ mod}} = R_{c,\text{orig}} + C_F \max\left(0, -\sum_k \mathbf{n}_B \cdot \mathbf{n}_k A_k\right) \Delta t^2 \frac{p(d_B) - p(d_I)}{(d_I - d_B)^2} |\mathbf{u}_B \cdot \mathbf{n}_B|. \tag{15.22}$$

This approach (with C_F set to 100) provides additional numerical dissipation within band cells when objects move but reduces to the solution of the continuity equation for non-moving objects.

Interface blocking for zero-thickness immersed surfaces When the continuity equation is solved within band cells, there is a need to identify mesh-cell faces across which mass flow must be restricted (‘blocking’ interfaces). This is a trivial task for objects that are closed, but for zero-thickness objects, special considerations must be made. To this end, we introduce indices for classifying mesh-cell interfaces as being blocking (no mass transport allowed) versus non-blocking (transport allowed) for zero-thickness immersed objects. The classification of the grid cells in the immersed-boundary (IB) method needs to be robust for any kind of complex immersed surface. Normally, two adjacent triangle elements share one edge; however, the disconnected edges at the boundary of non-closed object only belong to one triangle element. For a given cell’s center point $\mathbf{x}_q$, we find the nearest point $\mathbf{x}_s$ on a zero-thickness immersed surface using ANN algorithm (Arya et al. 1998) and then compute the unsigned distance function Φ. Using inner products between the position vector $\mathbf{x}_q$ at the query cell and the position vectors $\mathbf{x}_{nb}$ at adjacent neighboring cells with respect to the nearest point $\mathbf{x}_s$, we can classify the *band* cells for zero-thickness immersed surfaces by detecting a sign change of the inner product; i.e., if $(\mathbf{x}_q - \mathbf{x}_s) \cdot (\mathbf{x}_{nb} - \mathbf{x}_s) < 0$ for $|\Phi| \leq 2\Delta$, then $\mathbf{x}_q$ is *band* cell. Note that Δ is a representative grid resolution.

For an open surface (zero-thickness surface), the blocking index B is only valid for the case that nearest surface points are not on the disconnected edges of the immersed surface from two adjacent cells $\mathbf{x}_i$ and $\mathbf{x}_j$, because the signed distance functions at the cells may not be unique due to the ambiguity of the pseudo-normal vectors at the edges. To avoid the ambiguity, we introduce two incident angles to the parallel direction at the disconnected edges as shown in Fig. 15.5. Let us suppose that the nearest surface points are $\mathbf{x}_s^l$ for the lth immersed object and $\mathbf{x}_s^m$ for the mth immersed object at the cell $\mathbf{x}_i$ and $\mathbf{x}_j$ with the center of the interface $\mathbf{x}_{ij}$, respectively. We can define $B_l(\mathbf{x}_{ij})$ and $B_m(\mathbf{x}_{ij})$ based on the lth and the mth immersed objects, respectively, using the proposed algorithm. For example, if the nearest surface point $\mathbf{x}_s^l$ on the lth immersed object for the cell $\mathbf{x}_i$ is not on a disconnected edge, the blocking

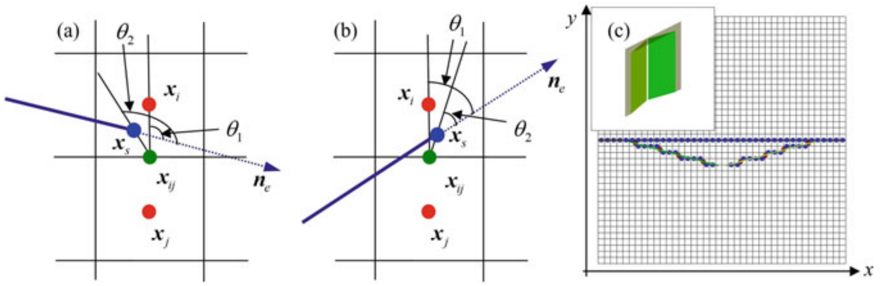

Fig. 15.5 Schematic of classifications for interface blocking, **a** non-blocking, **b** blocking, and **c** blocking index in the vicinity of doors and door frame. Red and blue colored circles represent the blocking index in x and y directions, respectively. Note that, there is no blocking in z direction. The actual doors and door frame are rendered in the inset figure

index $B_l(\mathbf{x}_{ij})$ can be simply determined by an inner product of two position vectors $\mathbf{x}_i - \mathbf{x}_s^l$ and $\mathbf{x}_j - \mathbf{x}_s^l$. However, if the nearest surface point $\mathbf{x}_s^l$ is on a disconnected edge, we need to define two angles θ_1^l and θ_2^l illustrated in Fig. 15.5 for determining the blocking index. The angles are defined as

$$\theta_1^l = \cos^{-1}\left[\left(\left(\mathbf{x}_i - \mathbf{x}_{ij}\right) \cdot \mathbf{n}_e^l\right) / \left(\left|\mathbf{x}_i - \mathbf{x}_{ij}\right|\left|\mathbf{n}_e^l\right|\right)\right] \tag{15.23}$$

$$\theta_2^l = \cos^{-1}\left[\left(\left(\mathbf{x}_s^l - \mathbf{x}_{ij}\right) \cdot \mathbf{n}_e^l\right) / \left(\left|\mathbf{x}_s^l - \mathbf{x}_{ij}\right|\left|\mathbf{n}_e^l\right|\right)\right], \tag{15.24}$$

where $\mathbf{n}_e^l$ is a unit vector that is orthogonal to the line segment of the disconnected edge and the plane involving with the triangle element containing the segment. Thus, the blocking index $B_l(\mathbf{x}_{ij})$ for the lth immersed object can be defined as

$$B_l(\mathbf{x}_{ij}) = \begin{cases} 1, \text{ if } \theta_1^l \geq \theta_2^l & \text{for } \mathbf{x}_s^l \text{ is on a disconnected edge} \\ 1, \text{ if } (\mathbf{x}_i - \mathbf{x}_s^l) \cdot (\mathbf{x}_j - \mathbf{x}_s^l) \leq 0 & \text{for } \mathbf{x}_s^l \text{ is not on a disconnected edge} \\ 0, \text{ otherwise} \end{cases} \tag{15.25}$$

Similarly, we can define $B_m(\mathbf{x}_{ij})$ based on the mth immersed object. Finally, the blocking index $B(\mathbf{x}_{ij})$ for all the immersed objects at the interface $\mathbf{x}_{ij}$ can be defined as

$$B(\mathbf{x}_{ij}) = \begin{cases} 1, \text{ if } B_l(\mathbf{x}_{ij}) + B_m(\mathbf{x}_{ij}) \geq 1 \\ 0, \text{ if } B_l(\mathbf{x}_{ij}) + B_m(\mathbf{x}_{ij}) = 0 \end{cases}, \tag{15.26}$$

Note that $B(\mathbf{x}_{ij})$ indicates that the interface is a *virtual* wall. This means that mass cannot be transferred through the interface and the information at the cell $\mathbf{x}_j$ is excluded in the interpolation stencil for the cell $\mathbf{x}_i$ and vice versa.

15.3 Simulations Involving Human Activity

15.3.1 Problem Definition

The primary use of the developed methodology has been in conducting simulations of realistic human motion, with a specific focus toward capturing induced wake and thermal plume effects on the transport of airborne agents, which can either be gas-phase or particulate in nature. Applications of this capability include entry/exit into shelters designed for collective protection of individuals from harmful agents. Such shelters may use overpressure to inhibit agent transport under static operating conditions and/or airlock systems to remove material that is inevitably transported into the system upon personnel entry. A key to the design of sheltering systems of this type is an understanding of the volume flow of air [normally expressed in cubic feet (CF)] exchanged during an entry event. With this information in place and with knowledge of the agent concentration field, it is possible to predict the mass flow of agent into the shelter.

Such entry events are highly dynamic, involving motion of multiple persons, moving doors, and possibly a transient external flow field. As such, the large-eddy simulation/immersed-boundary methodology described earlier can be used to good effect in capturing the flow physics. The remaining sections describe several applications of this type, along with strategies designed to reduce the output into forms suitable for incorporation into fast-running system performance models.

15.3.2 Agent Transport Due to Thermal Plume and Motion Effects

The first case considered involves simulation of an experiment conducted by Toyon. Incorporated involving tracer-gas transport due to the combined effects of buoyancy (human thermal plume) and wake transport (Juricek 2014). The experimental test chamber (Fig. 15.6) consists of two rooms, a 3 $\times$ 6 $\times$ 8 ft (L x W x H) antechamber, connected to a second, 12 $\times$ 6 $\times$ 8 ft main chamber by a swing door (24-in W $\times$ 70-in H). Compressed gaseous perfluorocarbon tracer compounds (PDCH and PMCH) mixed in air were released at a flow rate sufficient to ensure detectability. At time t = 0, a person initiates the release of the agent and walks from the antechamber into the main chamber, where he stands for 7.5 min. Tracer-gas concentrations (parts per billion) are sampled over one-minute intervals.

Simulation results for the 'moving' experiment are presented in Fig. 15.7 for a simulation of 7.5 min in duration and using a 12 M cell mesh. The moving person is rendered as a closed-surface immersed body and is incorporated as a sequence of STL files, generated using 3DSMax$^{®}$ using protocols described earlier. The hinged doors are rendered as zero-thickness immersed objects and are comprised of planar STL files. Rate laws for the door motion are defined in a separate subroutine. The tracer

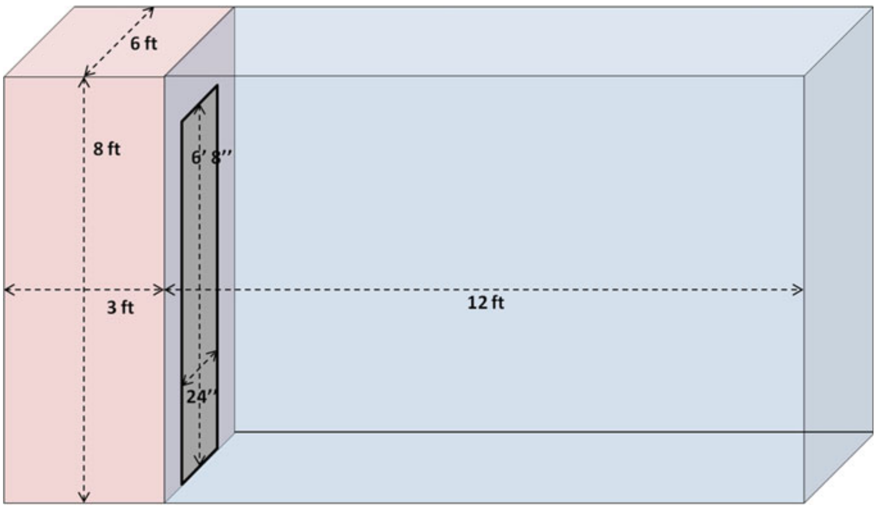

Fig. 15.6 Schematic of room chambers used in simulations

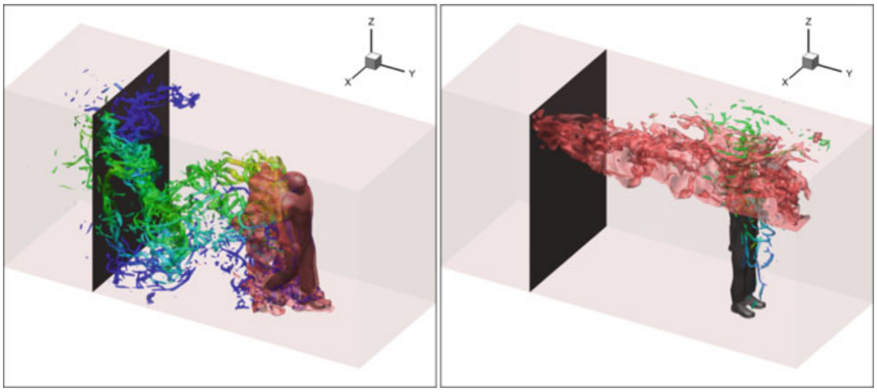

Fig. 15.7 Tracer-gas transport at 3.5 s (left figure) and 300 s (right figure)

gas is 'emitted' from a location under the person's left armpit—this involves the tagging of specific elements of the STL files as mass and momentum sources. In the actual experiment, the person held the tracer-gas emission tube at this same location. A similar approach is used to model human 'breathing' from the nose, though this effect is minor compared to transport due to the thermal plume. At 3.5 s into the event (left component of Fig. 15.7), the person's thermal plume is rendered as a red iso-surface ($T = 304$ K). Iso-surfaces of swirl strength, indicating locations of vortex cores, are colored by tracer-gas concentration. Wakes generated by closing door motion and human walking motion dominate thermal and tracer transport at early

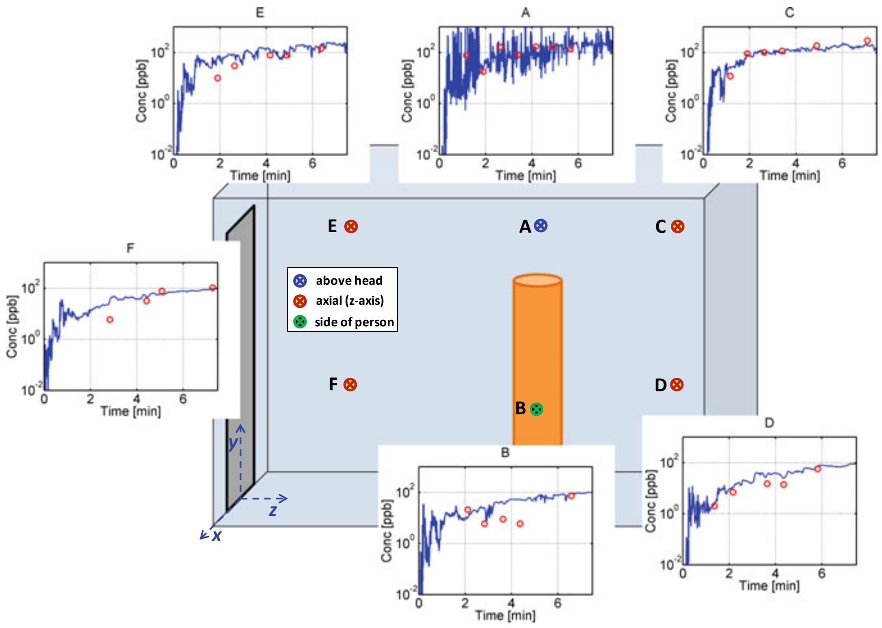

Fig. 15.8 Comparison of predictions with tracer-gas measurements for moving-person entry into the chamber

times. After 5 min (right component of Fig. 15.7), buoyancy-driven flow spreads the tracer-gas plume upward and away from the person.

Quantitative comparisons with experimental gas-sampling measurements are provided in Fig. 15.8. The centermost image shows probe locations within the chamber, while the surrounding images plot agent concentration (ppb) versus time. Probe A is directly above the person, and measurements here are affected both by regular human breathing motion, buoyancy, and (initially) by the decay of velocity fluctuations resulting from the door closing and the person stopping (due to inertia, the wake continues to move forward after the person stops, creating a disturbance field that moves entrained material forward and eventually upward). The predicted concentration levels (sampled at 100 Hz) are very noisy. Filtering the predictions over an interval of 10 s (corresponding to the time required for the gas-sampling syringe pump to operate) reduces the noise significantly. Generally, there is good overall agreement between the simulation and experiment. Probe *C* is further away from the source, and the concentration field in this region is not nearly as intermittent. The predictions are in close agreement with experiment at this location. The general agreement with experiment is reasonable at all probe locations, with some individual samples showing larger differences than others.

15.3.3 Airlock Entry Simulations

The next set of simulations focuses on personnel entry into a multiple-person-entry (MPE) airlock located at the front of a large shelter. These simulations were designed to determine the amount of gas transported into the airlock over the duration of an entry event as a function of the number and arrangement of entering personnel as well as wind speed and wind direction. The computational domain surrounding the shelter and within the interior of the airlock was rendered as a structured, multi-block mesh, with isotropic meshes used in the regions of human activity. Part of the interior of the shelter was also meshed to enable simulations of personnel entering the shelter from the airlock, leading to a total mesh-cell count of 31.3 M. Figure 15.9 shows a wire-frame view of the rendered interior of the complete domain.

Airlock initialization A separate calculation was used to initialize flow within the airlock, which is designed to operate at a target overpressure level. Figure 15.10 (left) shows a side view of the airlock mesh, emphasizing regions of mesh clustering designed to resolve various air jets and exit ports used to facilitate the purging of contaminated gas. Figure 15.10 (right) shows a snapshot of the airlock flow field, highlighting the entering jets of air from the manifold and from the shelter itself, which also operates at an overpressure. In the image, black streamlines emanate from the manifold, while red streamlines emanate from the shelter.

Initialization procedures The external velocity field was initialized using a Pasquill neutrally stable velocity profile. The inputted 'target' velocity for each trial corresponds to the velocity at 2 m above the surface. The inputted flow direction was used to resolve the velocity profile into directions perpendicular to and parallel to the door entrance plane. The orientation is such that 0° corresponds to flow directed into the door, 180° corresponds to flow directed out from the door, and 90° corresponds to flow parallel to the door entrance plane. The simulations were conducted for a fixed

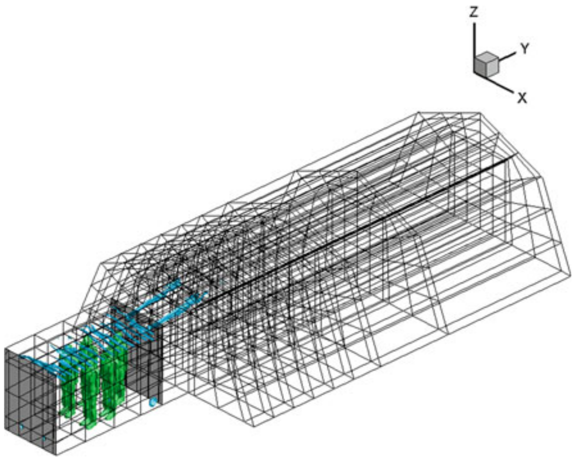

Fig. 15.9 Wire frame rendering of airlock and shelter

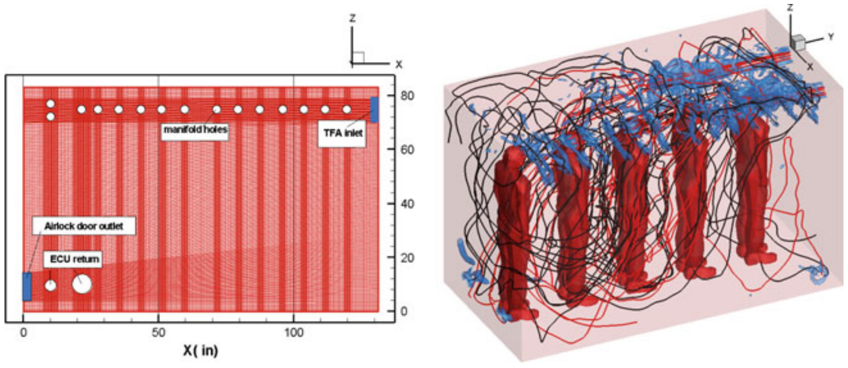

Fig. 15.10 Side view of airlock mesh (left) and snapshot of flow inside airlock (right)

period of time (10 s) prior to the entry event to allow the external flow to stabilize, and discrete wind speeds of 0, 0.8, 1.6, 3.2, 4.8, and 6.0 m/s and discrete wind directions of 0°, 45°, 90°, 135°, and 180° (26 trials, since zero wind speed holds for all directions) were used. Several personnel arrangements were used during the course of the study: single-person entry, five-person single-file entry, four people carrying a patient on a litter, seven-person single-file entry, five-person side-by-side entry, and seven-person side-by-side entry. Animation sequences for each of the entry events were created using 3DSMax®, and the generated sequences of STL files were converted to closed immersed objects using procedures described earlier. The bump-through doors, rendered as planar STL files and containing embedded vents for overpressure control, were 'opened' and 'closed' through the use of specially defined rate laws and were rendered as zero-thickness immersed surfaces.

Five-person, side-by-side airlock entry Figure 15.11 shows snapshots corresponding to the entry of five people side by side into the multi-person airlock. The average walking speed of the group is 1.1 m/s, and the wind speed is zero for this case. Isosurfaces of swirl strength, colored by agent concentration, illustrate the flow patterns generated upon entry. Red contours correspond to a normalized agent concentration of unity, while blue contours correspond to a normalized agent concentration of zero. Frame A corresponds to conditions just prior to entry. Highlighted flow features include air jets entering the airlock from the multi-port manifold and the exiting of air through the door values to maintain the target overpressure. The doors open (Frame B) just prior to entry, leading to an initial expulsion of air in the direction of the entry. A suction pressure is created behind the exiting vortex, allowing flow outside the airlock to migrate into the system. This, combined with the effects of wakes induced by moving personnel, induces net agent transport into the airlock (Frame C). As the doors close, the airlock begins to recover the target overpressure, and flow again emerges from the door vents (Frame D). Figure 15.12 shows a close-up view of wake structures generated as the group makes their way through the released air stream. The time is just after Frame B above; the doors are rendered as transparent to provide a better view of the interior of the airlock. Figure 15.13 (left) plots cubic

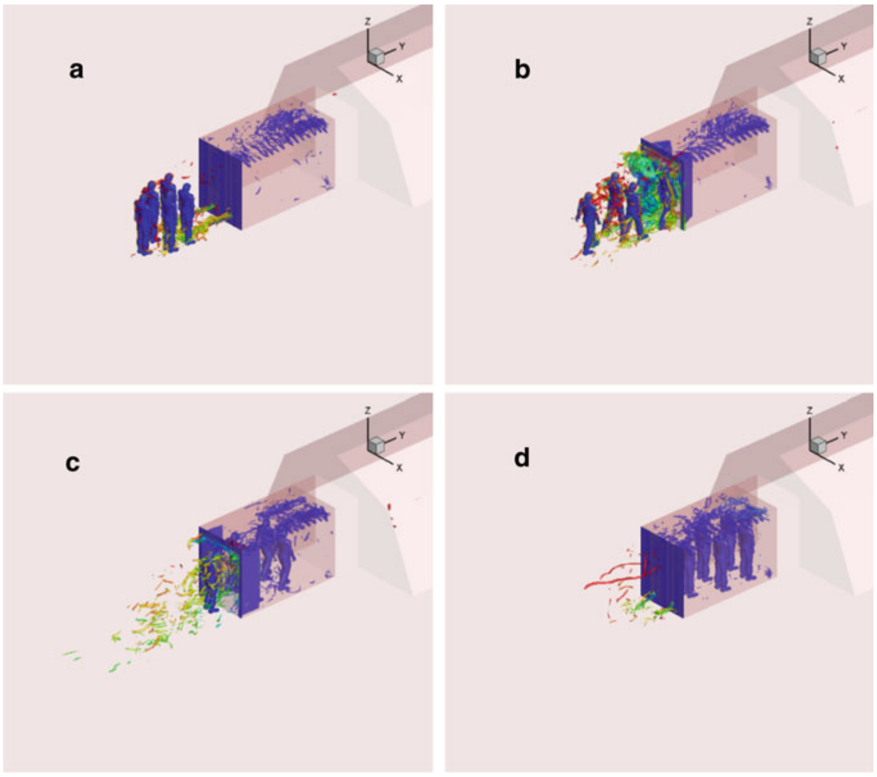

Fig. 15.11 Five-person side-by-side entry into MPE

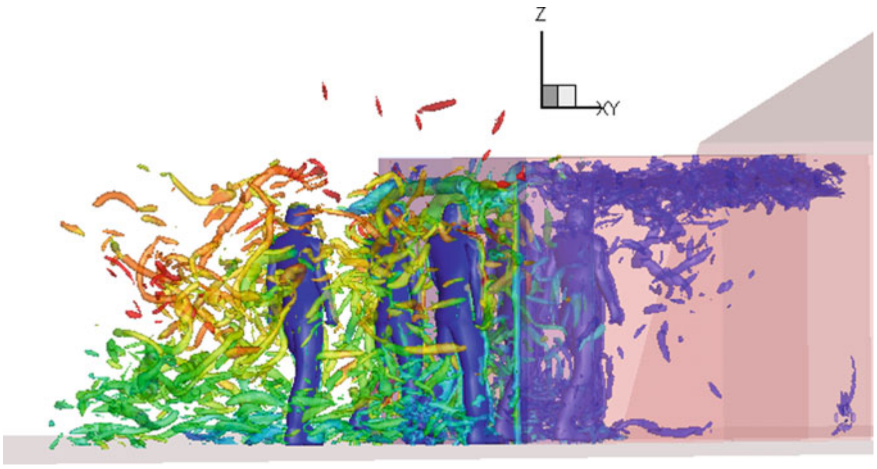

Fig. 15.12 Close-up view of group entering airlock

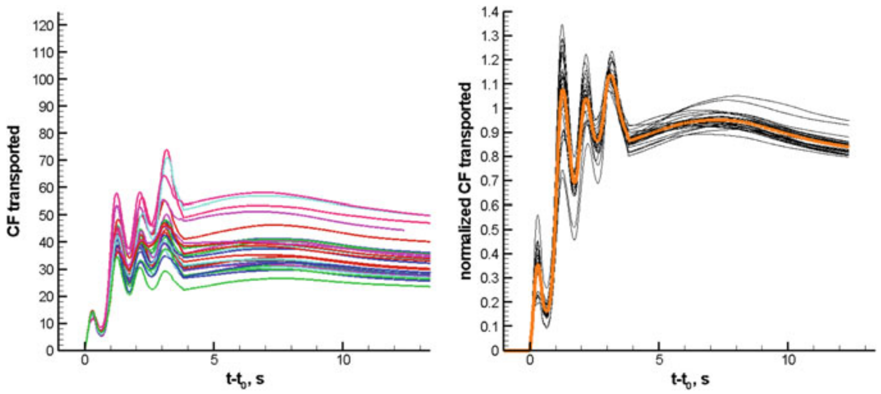

Fig. 15.13 Raw (left) and normalized (right) CF transported versus time

feet of gas transported into the airlock versus time for different wind speed/wind direction combinations. The transport histories are similar to one another and can be effectively collapsed by normalizing by the target CF value at the door closing point (the average of the upper and lower peaks), as shown in Fig. 15.13 (right).

For eventual inclusion into a fast-running system performance model, it is necessary to correlate the target CF transported at the door closing point as a function of wind speed and wind direction. One might expect that wind vectors more aligned with the entry event would enhance transport into the airlock, as would higher wind speeds, but the entry event can also be in the wake of the shelter for wind directions greater than 90°, leading to interactions with vortices shed by the airlock and shelter edges. The dependence is thus not trivial, and our best approach has been to fit the target CF as a function of wind speed and direction angle using a single hidden-layer, ten-node neural network with a sigmoidal activation function:

$$\begin{aligned} \mathrm{CF}_{\mathrm{target}}(V,\theta) &= (c_1 + \sum_{k=1}^{10} b_k h_k)c_2 + c_3 \\ h_k &= \tfrac{1}{1+\exp(-x_k)} \\ x_k &= a_{1,k} + a_{2,k}\tfrac{V-a_{3,k}}{a_{4,k}} + a_{5,k}\tfrac{\theta-a_{6,k}}{a_{7,k}} \end{aligned} , \qquad (15.27)$$

Figure 15.14 shows scatter plots of $\mathrm{CF}_{\mathrm{target}}$ predicted by Eq. 15.27 for 50,000 randomly distributed (V, θ) ordered pairs, with V varied from 0 to 6 m/s and θ varied from 0 to 180°. A good coverage of the factor space is indicated, and most of the trial data points lie within the predicted factor space. The average error is 4.20%, and the largest error is around 8%. It is also to be noted that this case shows the expected trends of increased transport into the airlock for higher wind speeds and directions more aligned with the movement of the group. The zero wind speed values represent the effects of wake transport in the absence of wind motion. CF transported generally increases with the number of personnel, but the arrangement also affects transport. A

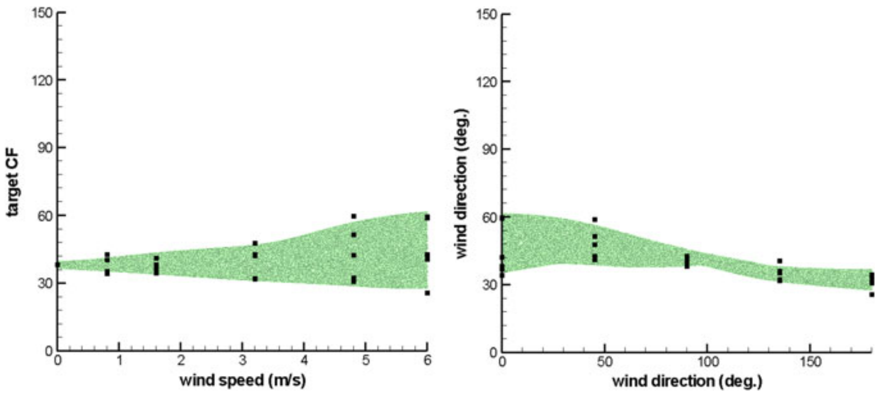

Fig. 15.14 Predicted target CF versus wind speed and wind direction

similar case conducted with five people entering in single file results in nearly twice as much transport at zero wind speed (60 CF vs. 36 CF). This is partially due to the duration of the event, which is ~3.5 s for the side-by-side entry versus 5 s for the single-file entry.

15.3.4 Flow Over a Ruined Building

The last example, while not involving moving entities, illustrates the process of constructing a scenario, creating geometries as sets of STL files, and rendering the objects as closed or zero-thickness immersed bodies. The scenario involves a person buried in rubble releasing a gas-phase taggant to aid in his rescue. A CAD description of a ruined building was obtained from Turbosquid.com (an online retailer for 3D CAD models used in gaming). The building geometry is that of a small house with four small rooms that has collapsed upon itself. The geometry was imported into 3DSMax® and then exported as a binary STL file. This file was then read into Autodesk's NetFabb®, a tool for assembling, repairing, and modifying STL files for use in 3D printing. The STL file for the soldier used in the earlier simulations was added to the scenario, rescaled, and repositioned, so that he was 'trapped' under a portion of the building. The STL files were then exported and pre-processed using the steps described earlier for inclusion as immersed objects in the simulation. The geometry is open to the air above, as shown in Fig. 15.15. The placement of the person and the wind direction is also shown in the figure. The same Pasquill boundary layer used in the shelter simulation was used in this case, which contains about 64 M cells with an isotropic region surrounding the region occupied by the object. The cell size in the isotropic region is 1 in. The person holds the taggant canister and also breathes but otherwise is stationary (his legs are pinned underneath a part of the building).

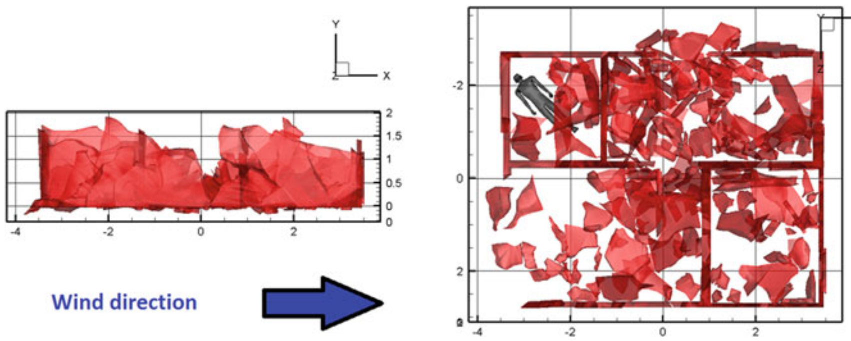

Fig. 15.15 Collapsed building containing injured person

Mass and momentum sources were applied at locations on the person's STL object to mimic taggant release and transient breathing.

Figure 15.16 shows the flow structures that emerge after several transit times. In the left image, an iso-surface of taggant mass fraction (0.0001) is shown colored by velocity magnitude. The image on the right shows iso-surfaces of swirl strength colored by the logarithm of taggant mass fraction. The irregular geometry

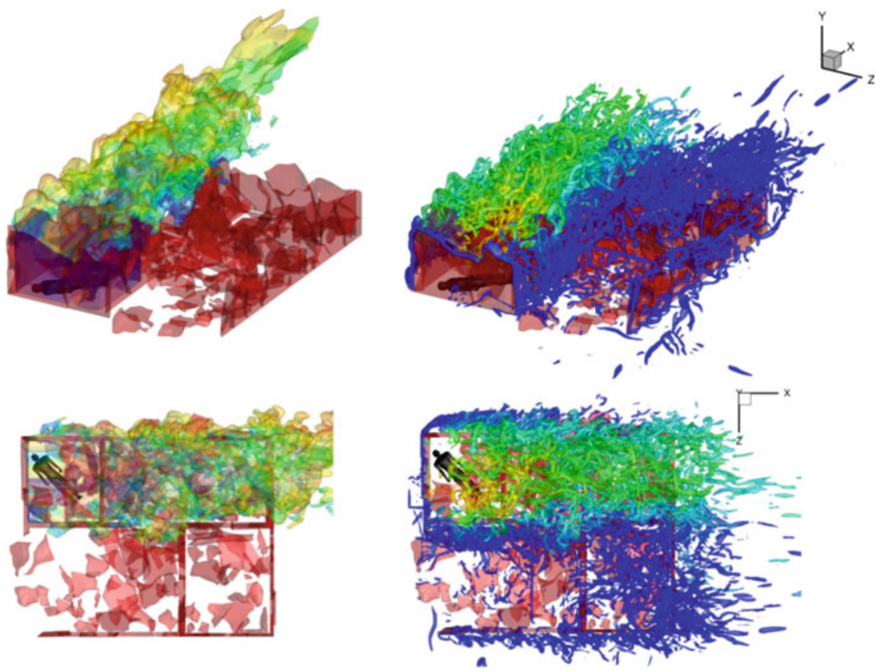

Fig. 15.16 Taggant concentration (left) and swirl strength (right) iso-surfaces: flow over a ruined building—bottom images show top-down views

of the building provides sources for turbulence generation as well as low-momentum regions that may trap fluid. The small enclosure in which the person is placed is one such region—the taggant fills the entire enclosure before being entrained into the external wind field. The breath gas remains within the enclosure, but breathing is a periodic source of effluent—later times would show the expulsion of the breath gas from the enclosure. The fact that the chosen taggant (SF6) is non-buoyant keeps the plume close to the surface.

15.4 Conclusion

An immersed-boundary method suitable for general flow simulations has been presented. The model is grid-topology independent and is based on the decomposition of a computational domain into cells inside an immersed body (field cells), cells outside but adjacent to an immersed body (band cells), and cells far away from an immersed body (field cells). Immersed objects are generated initially as sets of closed-surface or zero-thickness stereo-lithography (STL) files. Procedures for rendering these files as immersed objects within the domain hinge first on splitting such objects into simpler units and secondly on the calculation of the signed distance from each field cell to the embedded surfaces. Interpolation methods based on turbulent boundary layer theory are used to connect the flow solution in band cells to specified surface boundary conditions and to the solution of the Navier–Stokes equations in the field cells. The approach differs from others in the literature in its use of power-law forms for the near-surface velocity, thus enabling the method to mimic the energizing effect of a turbulent boundary layer without excessive near-surface resolution. Applications have been presented for cases involving gas-phase agent transport as induced by human activity (including realistic human motion, breathing, and buoyancy effects due to the human thermal plume) and by other factors, such as an external flow field and moving doors. The combination of large-eddy simulation techniques for capturing wake-induced turbulence and the developed immersed-boundary techniques for representing the effects of stationary and moving objects on the flow evolution provides a powerful framework for conducting realistic simulations of complicated time-dependent flows.

Acknowledgements This work has been supported by the Naval Surface Warfare Center, Dahlgren Division (N001178-08-C-3030) and by Toyon Corporation under a subcontract from the Air Force Research Laboratory (FA8650-13-M-6449).

References

Arya S, Mount DM, Netanyahu NS, Silverman R, Wu AY (1998) An optimal algorithm for approximate nearest-neighbor searching. J ACM 45:891–923

Bærentzen JA, Aanæs H (2005) Signed distance computation using the angle weighted pseudonormal. IEEE T Vis Comp Graph 11(3):243–253

Baurle RA, Tam CJ, Edwards JR, Hassan HA (2003) Hybrid simulation approach for cavity flows: blending, algorithm, and boundary treatment issues. AIAA J 41:1463–1480

Choi J-I, Edwards JR (2008) Large eddy simulation and zonal modeling of human-induced contaminant transport. Indoor Air 18:233–249

Choi J-I, Edwards JR (2012) Large-eddy simulation of human-induced contaminant transport in room compartments. Indoor Air 22:77–87

Choi J-I, Oberoi RC, Edwards JR, Rosati JA (2007) An immersed boundary method for complex incompressible flows. J Comput Phys 224:757–784

Choi J-I, Edwards JR, Rosati JA, Eisner AD (2012) Large eddy simulation of particle re-suspension during a footstep. Aerosol Sci Technol 46(7):767–780

Chorin AJ (1967) A numerical method for solving incompressible Navier-Stokes equations. J Comput Phys 2:12–26

Colella P, Woodward PR (1984) The piecewise parabolic method (PPM) for gas-dynamical simulations. J Comput Phys 54:174–201

Crowe CT, Troutt TR, Chung JN (1996) Numerical models for two-phase turbulent flows. Annu Rev Fluid Mech 28:11–43

Edwards JR, Liou M-S (1998) Low-diffusion flux-splitting methods for flows at all speeds. AIAA J 36:1610–1617

Edwards JR, Choi J-I, Ghosh S, Gieseking DA, Eischen JD (2010) An immersed boundary method for general flow applications. In: FEDSM-ICNMM2010-31097, ASME 2010 3rd joint US-European fluids engineering summer meeting

Fadlun EA, Verzicco R, Orlandi P, Mohd-Yusof J (2000) Combined immersed boundary/finite-difference methods for three-dimensional complex flow simulations. J Comput Phys 161:35–60

Ghosh S, Choi J-I, Edwards JR (2010a) Numerical simulation of effects of micro vortex generators using immersed boundary methods. AIAA J 48(1):92–103

Ghosh S, Choi J-I, Edwards JR (2010b) Simulation of shock/boundary layer interactions with bleed using immersed boundary method. J Propul Power 26(2):203–214

Ghosh S, Choi J-I, Edwards JR (2012) Numerical simulation of the effects of mesoflaps in controlling shock/boundary layer interactions. J Propul Power 28(5):955–970

Gilmanov A, Sotiropoulus F, Balaras E (2003) A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids. J Computat Phys 191:660–669

Gouraud H (1971) Continuous shading of curved surfaces. IEEE T Comp 20(6):623–629

Guéziec A (2001) Meshsweeper: dynamic point-to-polygonal-mesh distance and applications. IEEE T Vis Comp Graph 7(1):47–61

Hoff KE, Culver T, Keyser J, Lin M, Manocha D (1999) Fast computation of generalized voronoi diagrams using a graphics hardware. In: Proceedings of the SIGGRAPH'99, pp 277–285

Juricek B et al (2014) Volatile organic compound odor signature modeling. Phase I SBIR Final Report, Air Force Contract FA8650-13-M-6449

Karypis G, Kumar V (1998) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput 20:359–392

Linhart J (1990) A quick point-in-polyhedron test. Comput Graph 14(3):445–448

Mittal R, Iaccarino G (2005) Immersed boundary methods. Ann Rev Fluid Mech 37:239–261

Mohd-Yosuf J (1997) Combined immersed boundary/B-spline methods for the simulation of flow in complex geometries, Ann Res Briefs CTR 317–328

Neaves MD, Edwards JR (2006) All-speed time-accurate underwater projectile calculations using a preconditioning algorithm. ASME J Fluids Eng 128:284–296

Oberoi RC, Choi J-I, Edwards JR, Rosati JA, Thornburg J, Rodes CE (2010) Human-induced particle re-suspension in a room. Aerosol Sci Technol 44(3):216–229

Payne BA, Toga AW (1992) Distance field manipulation of surface models. Comp Graph Appl 12(1):65–71

Peskin CS (1972) Flow patterns around heart valves: a numerical method. J Comput Phys 10:220–252

Smagorinsky J (1963) General circulation experiments with primitive equations, the basic experiment. Mon Weather Rev 91:99–164

Verzicco R, Mohd-Yusof J, Orlandi P, Haworth D (2000) LES in complex geometries using boundary body forces. AIAA J 38:427–433

Walz A (1969) Boundary layers of flow and temperature (English translation), MIT Press, Cambridge

Wesseling P (1995) Introduction to multigrid methods, NASA CR-195045