

Accurate wind characterization in complex terrain using the immersed boundary method

K. A. Lundquist^{a,b}, F. K. Chow^a, J. K. Lundquist^c, B. Kosović^d

^a *University of California, Berkeley, CA, USA, kal@cal.berkeley.edu, tinakc@berkeley.edu*

^b *Lawrence Livermore National Laboratory, Livermore, CA, USA, kal@llnl.gov*

^c *University of Colorado, Boulder, CO, USA, julie.lundquist@colorado.edu*

^d *National Center for Atmospheric Research, Boulder, CO, USA, branko@ucar.edu*

ABSTRACT: This paper describes an immersed boundary method (IBM) that facilitates the explicit resolution of complex terrain within the Weather Research and Forecasting (WRF) model. Two different interpolation methods, trilinear and inverse distance weighting, are used at the core of the IBM algorithm. Functional aspects of the algorithm's implementation and the accuracy of results are considered. Simulations of flow over a three-dimensional hill with shallow terrain slopes are performed with both WRF's native terrain-following coordinate and with both IB methods. Comparisons of flow fields from the three simulations show excellent agreement, indicating that both IB methods produce accurate results. However, when ease of implementation is considered, inverse distance weighting is superior. Furthermore, inverse distance weighting is shown to be more adept at handling highly complex urban terrain, where the trilinear interpolation algorithm breaks down. This capability is demonstrated by using the inverse distance weighting core of the IBM to model atmospheric flow in downtown Oklahoma City.

1 INTRODUCTION

Computational fluid dynamics codes are used at the microscale to predict atmospheric boundary layer flows over complex terrain for a variety of applications, ranging from the siting of wind turbines to predictions of flow in urban terrain for contaminant dispersion. Mesoscale numerical weather prediction models are being increasingly used at higher resolutions, tending towards the microscale, but face several inherent limitations which prevent accurate simulations in complex terrain at this scale. One limitation is the use of terrain-following coordinates in most mesoscale models. This coordinate accommodates complex terrain by transforming the physical domain onto a Cartesian grid, thereby simplifying the application of lower boundary conditions. The transformation introduces metric terms into the governing equations, which when discretized, introduce additional truncation errors. These coordinate transformation errors significantly degrade the quality of the solution in steep terrain, as demonstrated by previous researchers (Janjic 1977; Klemp et. al. 2003; Zängl 2004).

To improve the prediction of atmospheric flows in complex terrain, we have implemented an immersed boundary method (IBM) in the mesoscale Weather Research and Forecasting (WRF) model. IBM is a gridding technique, which allows complex terrain to be modeled without a coordinate transformation, therefore eliminating restrictions on terrain slope and the errors associated with coordinate mapping. With the IBM, coordinate surfaces pass through the terrain, and boundary conditions are applied within the interior of the computational domain through the addition of a forcing term in the governing equations. Our IBM handles both Dirichlet and Neumann boundary conditions. Additionally, realistic surface forcing can be provided at the immersed boundary by atmospheric physics parameterizations, which are modified to include the

effects of the immersed terrain. A version of our IBM suitable for two-dimensional terrain is presented in Lundquist et al. (2010). In this work, the method has been extended to accommodate fully three-dimensional terrain and is now capable of running on highly parallelized machine architectures.

A key component of the immersed boundary method is the formulation of the forcing term used to impose the correct boundary condition at the immersed surface. Because the immersed surface is not coincident with the grid, an integral part of the immersed boundary algorithm is the interpolation method used in the calculation of the forcing term. In this work, we examine the use of two different interpolation methods. In the first interpolation method, which is trilinear, weighting coefficients are determined by inverting a Vandermonde matrix. This method is chosen here because one or multi-dimensional linear interpolation methods were the first interpolation methods to appear in the IBM literature, and are still commonly used. In the second interpolation method, weighting coefficients are determined as a function of inverse radial distance. This method provides more flexibility in the choice of points influencing the interpolation because the Vandermonde matrix is eliminated, thereby eliminating the constraint that the matrix be well-conditioned.

Each method is used in a simulation of flow over a three dimensional hill, and the results are compared to those using terrain-following coordinates. Additionally, results are presented for flow in downtown Oklahoma City.

2 NUMERICAL METHOD

2.1 Background on the immersed boundary method

IBM is used to represent the effects of boundaries on a nonconforming structured grid. Boundary conditions are imposed with the addition of a body force term F_B in the conservation equations for momentum and scalars (1).

$$\partial_t \mathbf{V} + \mathbf{V} \cdot \nabla \mathbf{V} = -\alpha \nabla p + \nu \nabla^2 \mathbf{V} + \mathbf{g} + \mathbf{F}_B \quad (1a)$$

$$\partial_t \varphi + \mathbf{V} \cdot \nabla \varphi = \nu_t \nabla^2 \varphi + F_\varphi + F_B \quad (1b)$$

The body force term takes a zero value away from the boundaries, but modifies the governing equations in the vicinity of the boundary. Generally, IB methods include a determination of the forcing term, and an interpolation scheme to reconstruct the boundary condition on the immersed surface, which is not coincident with computational nodes.

In this work, a method commonly referred to as direct or discrete forcing is used (Mohd_Yusof, 1997). With direct forcing, the velocity or scalar value is modified at forcing points near the boundary to enforce the boundary condition, eliminating the need for explicit calculation of the body force term in the numerical algorithm. Within the direct forcing class of methods, we have adopted an approach where forcing is applied at ghost cells, defined as the layer of computational nodes located just within the solid domain.

The value of the variable at the ghost cell which will enforce either a Dirichlet (2a) or Neumann (2b) boundary condition at the immersed surface Ω must be computed.

$$\varphi = \varphi_\Omega \quad (2a)$$

$$\mathbf{n} \cdot \nabla \varphi = \left. \frac{\partial \varphi}{\partial \mathbf{n}} \right|_\Omega \quad (2b)$$

Several different interpolation methods have been employed by researchers for the purpose of making this calculation, including multi-dimensional linear and quadratic interpolation (Tseng and Ferziger, 2003), inverse distance weighting (Gao et al., 2007), Lagrange, and least squares interpolations (Peller et al. 2006).

We examine two interpolation schemes here, trilinear and inverse distance weighted interpolations. In each case, the location of the ghost cell is reflected across the boundary, as in Figure 1, and this is labeled as an image point. Next, one of the interpolation methods is applied to determine the value of the variable at the image point. Neighbors for the interpolation differ between the two methods, but can include both computational nodes and points on the boundary. The algorithm for choosing the neighbors is described further sections 2.2 and 2.3, but in general, neighbors are determined by proximity to the image point.

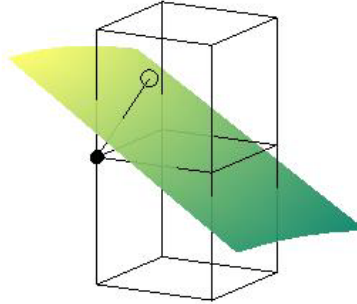


Figure 1. Ghost points are a layer of computational nodes just underneath the terrain. Here, a portion of terrain (the green surface) is shown with the computational cells that it cuts through. A ghost point is marked with a solid circle. An image point, marked with an open circle, is found by reflecting the ghost point across the terrain in the surface normal direction. A line connecting the ghost and image points is the surface normal.

Once the value of the image point is calculated, it is used to find the value of the ghost point which enforces the boundary conditions. When a Dirichlet boundary condition is used, the ghost point is related to the image point value with $\varphi_G = 2\varphi_\Omega - \varphi_I$. For a Neumann boundary condition the relationship is $\varphi_G = \varphi_I - \overline{GI} \frac{\partial \varphi}{\partial \mathbf{n}} \Big|_\Omega$, where \overline{GI} is the distance between the ghost and image points.

2.2 Trilinear interpolation

In trilinear interpolation the interpolant is the product of three linear functions, one in each dimension. The value of the image point is calculated using the interpolant given by equation (3).

$$\varphi = c_1 + c_2x + c_3y + c_4z + c_5xy + c_6xz + c_7yz + c_8xyz \quad (3)$$

Eight neighboring points are used to define the interpolation region, and are chosen as either computational nodes or boundary points. An example of this is shown in figure 2. The constants c in the interpolant are determined by solving a linear system of equations (4) for each ghost point, where the rank is equal to the number of neighbors.

$$\mathbf{c} = \mathbf{A}^{-1} \boldsymbol{\varphi} \quad (4)$$

The matrix \mathbf{A} and the vector $\boldsymbol{\varphi}$ are dependent on the neighbors chosen for the interpolation and the type of boundary condition being imposed. For Dirichlet boundary conditions, equation (3) appears in the matrix equation. If the neighbor is a computational node, then φ takes the value calculated at the node. If the neighbor is a boundary point, then the boundary condition is assigned to φ . For Neumann boundary conditions, the gradient of the interpolation function is substituted into the boundary condition (2b), and equation (5) results. For neighbors on the boundary, equation (5) is used in (4).

$$\begin{aligned} \frac{\partial \varphi}{\partial \mathbf{n}} = & c_2 \mathbf{n}_x + c_3 \mathbf{n}_y + c_4 \mathbf{n}_z + c_5 (\mathbf{n}_y x + \mathbf{n}_x y) + c_6 (\mathbf{n}_z x + \mathbf{n}_x z) + c_7 (\mathbf{n}_z y + \mathbf{n}_y z) \\ & + c_8 (\mathbf{n}_z xy + \mathbf{n}_y xz + \mathbf{n}_z xy) \end{aligned} \quad (5)$$

Once the interpolation constants are determined, the value of the image point is found with equation (3). As a last step, the variable value at the ghost node is calculated and assigned.

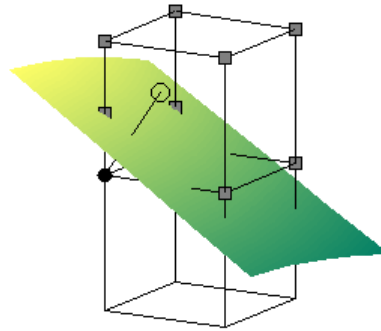


Figure 2. Eight neighbors marked by squares are chosen for use in determining the coefficients of the trilinear interpolant. In this case, six neighbors are computational nodes and two are located on the surface of the immersed boundary at the intersection of the boundary and one face of the cut cell.

2.3 Inverse distance weighted interpolation

With the inverse distance weighted interpolation proposed in Franke (1982), the value of the image point is calculated with the interpolant given by equation (6), which is simply a weighted average of the neighboring points.

$$\varphi = \frac{\sum_n c_n \varphi_n}{\sum_n c_n} \quad (6)$$

Any number of neighboring points can be used to define the interpolation region. Weighting coefficients c , given by equation (7), are a function of radial distance from the image point. In (7), R_{max} is the maximum radius of the group of neighbors, and R_n and c_n are the radial distance and weighing coefficient for the n^{th} neighbor.

$$c_n = \left(\frac{R_{max} - R_n}{R_{max} R_n} \right)^p \quad (7)$$

This function produces an infinite weight for a node that is coincident with the image point, while the node located furthest away at R_{max} has no influence with a weighing factor of zero. The variable p is a power function that controls the rate of decay of the weighting coefficient with increasing radial distance. For the simulations presented here, p is set to one.

In our algorithm, the first step in identifying neighbors is searching the 64 points surrounding the image point (i.e. a distance of 2 nodes in each direction which defines a 4^3 cloud of nodes surrounding the image point). Potential neighbors are identified as those residing in the fluid domain. These nodes are sorted by increasing radial distance, and eight neighbors are chosen by proximity.

When Dirichlet boundary conditions are used, the first neighbor is on the immersed boundary along the surface normal vector connecting the image and ghost points. The remaining seven neighbors are computational nodes, as illustrated in figure 3a. Inverse distance weighting preserves maxima and minima, even during extrapolation. Therefore, it is guaranteed that the interpolated image point value will be bounded by the values of the boundary condition and neighboring computational nodes.

When Neumann boundary conditions are imposed, a boundary point is not used because the value on the surface is unknown. In this case, all eight neighbors are computational nodes. Without a point on the boundary, the image point can lie outside of the interpolation region, usually

when the ghost node is near the surface as in figure 3. If extrapolation is used, the calculated value of the image point may not properly account for gradients in the variable field because the extrapolated value is bounded by the values at the neighbors. In this case, the image point is modified by relocating it to be at the intersection of the surface normal with a face of the computational cut-cell, as shown in figure 3b. Although it is no longer a true image, the ship $\varphi_G = \varphi_I - \overline{GI} \frac{\partial \varphi}{\partial n} \Big|_{\Omega}$ can still be used to achieve the Neumann boundary condition.

In either case (Dirichlet or Neumann), if eight neighbors are not available, the algorithm proceeds with fewer neighbors, using all available nodes within the search area that also reside in the fluid domain. As with trilinear interpolation, once the value at the image point is calculated, the last step is to enforce the boundary condition by calculating and assigning the ghost point value.

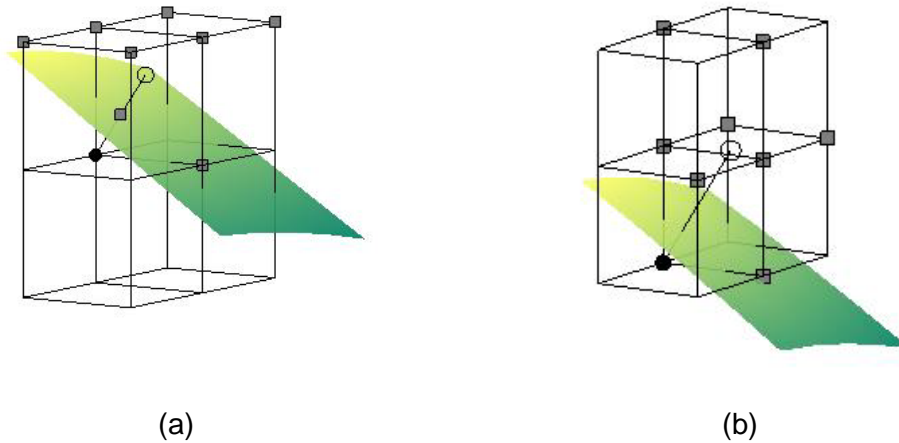


Figure 3. Eight neighbors marked by squares are chosen for use in determining the coefficients of the inverse distance weighting interpolant. In (a) points are shown for a Dirichlet boundary condition, and in (b) points are shown for a Neumann boundary condition.

3 VERIFICATION CASE

In this section, we verify the implementation of our three-dimensional immersed boundary method in the WRF model, and evaluate the two interpolation methods. Verification is performed by simulating flow over a shallow three-dimensional hill with the native terrain-following coordinates and with each of the immersed boundary methods, and comparing the results.

3.1 Model Set-up and initialization

The test flow case is start-up flow over a three-dimensional hill. The terrain height h_t is defined by the Witch of Agnesi curve given in equation 8, using a peak height h_p of 40 m and a mountain half-width a of 100 m.

$$h_t(x, y) = \frac{h_p}{1+(x/a)^2+(y/a)^2} \quad (8)$$

The maximum slope of the terrain is 14.5 degrees. This slope is sufficiently shallow to allow for direct comparisons between simulations using terrain-following coordinates and those using the immersed boundary method. The flow is initialized with a neutral and quiescent sounding, and driven with a constant pressure gradient in the x direction. The number of grid points in each direction is $(n_x, n_y, n_z) = (75, 75, 70)$ for the terrain-following cases, and $(n_x, n_y, n_z) = (75, 75, 80)$ for the immersed boundary method. Ten additional points are used in the vertical direction in the im-

mersed boundary method to account for the fact that nodes are needed underneath the terrain. In the horizontal dimensions, a constant 8 m grid spacing is used. In the vertical dimension, the grid points are equally spaced in WRF's native pressure based coordinate η over the domain height of 600 m.

Periodic boundary conditions are used at the lateral boundaries. A no-slip boundary condition is set on velocity at the terrain surface, along with a zero flux condition on temperature. At the top of the domain, the native WRF boundary condition is used (isobaric and a material surface), with a Rayleigh damping layer that acts only on vertical velocity at the top 100 m.

3.2 Results for flow over a three dimensional hill

The flow is integrated for 2 hours, and profiles of each velocity are shown for several locations along the x dimension for a slice of data in y that is slightly off-center in the domain. It can be seen that the three profiles (two for IBM and one for terrain-following coordinates) are nearly indistinguishable. IBM results, represented by the blue and green lines extend below the terrain, where as the results using terrain-following coordinates represented by the red line end just above the terrain.

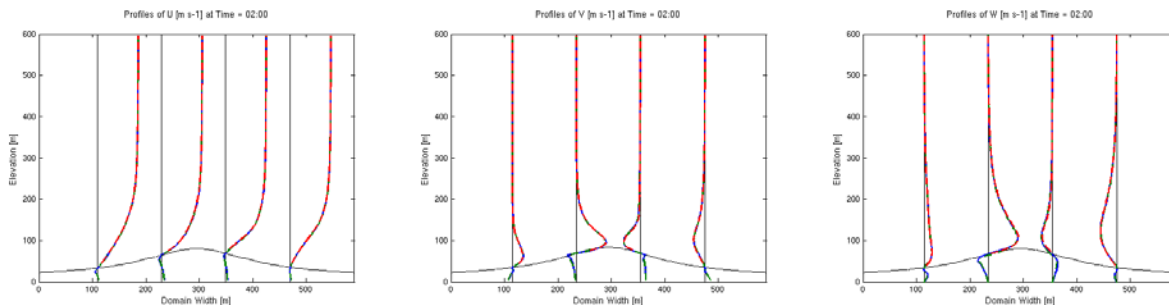


Figure 4. Profiles of u , v , and w velocity are shown for several horizontal locations located along a slice in the y dimension. The lines are nearly indistinguishable. The blue line represents trilinear interpolation, the green line represents inverse distance weighing, and the red line represents terrain-following coordinates.

4 FLOW IN URBAN ENVIRONMENTS

The Joint Urban 2003 field campaign took place in Oklahoma City over a period of one month, and is detailed in Allwine and Flaherty (2006). Over 20 institutions participated in the study, providing an extensive data set of urban atmospheric flow and dispersion. During the campaign, the city was instrumented with lidars, sodars, radars, sonic anemometers, airplane-based meteorological sensors, fast-response tracer analyzers, and helicopter based remote tracer detectors. This data provides an excellent test case for verifying the use of IBM in a numerical weather prediction model. In preparation for simulating IOPs from this field campaign, we have modeled flow over a portion of Oklahoma City using an idealized model set-up.

4.1 Model Set-up and initialization

The set-up used in this simulation is similar to that described in section 3.1. A two-dimensional array of terrain data for the Oklahoma City case is created by overlaying the horizontal WRF grid with an ESRI shapefile of the downtown region, and sampling the building heights that are coincident with the nodes on the WRF grid. The shapefile data and resulting three-dimensional terrain are shown in figure 5.

At initialization the atmosphere is neutral and at rest. Flow is driven with a constant pressure gradient in the y direction, and periodic lateral boundary conditions are used. The number of grid points in each direction is $(n_x, n_y, n_z) = (260, 320, 170)$. In the horizontal directions, a constant 2 m grid spacing is used. In the vertical dimension, the grid points are equally spaced in the η pressure coordinate over the domain height of 425 m.

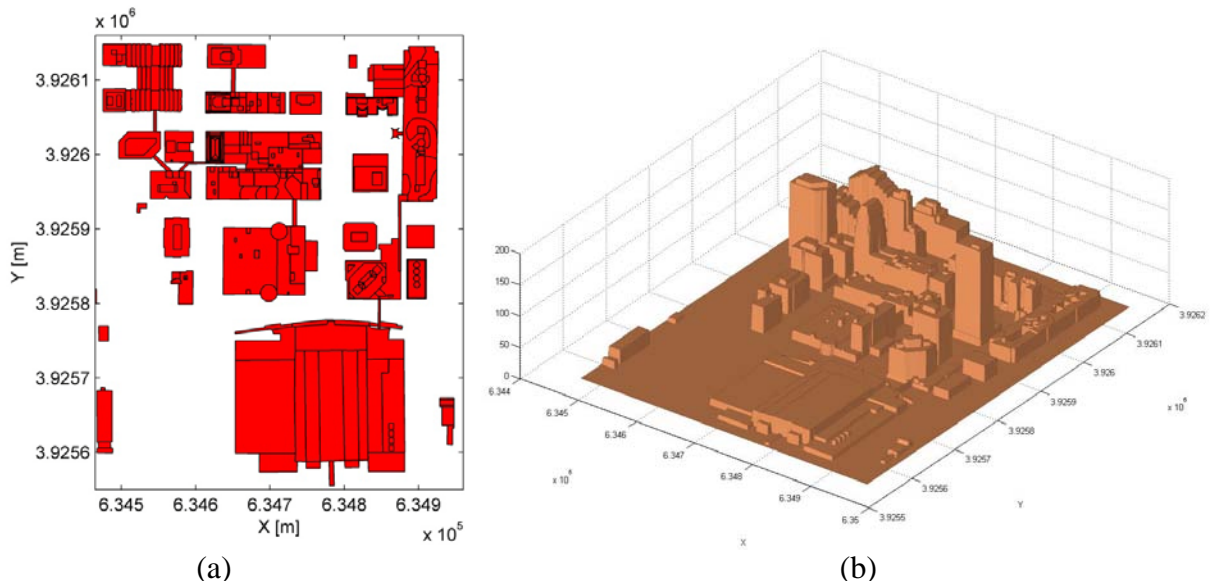


Figure 5. (a) ERSI shapefile data for the buildings included in the Oklahoma City domain. (b) A two-dimensional array of terrain heights sampled from the ERSI shapefile is used to define the terrain used in the WRF simulation.

4.2 Results for flow through urban terrain

Figure 6 shows side and top contours of all three velocity components 43 minutes after initialization. The primary flow direction is along the positive y axis. Many flow features are present in the simulation, including high speed jets at contractions of urban canyons and separation zones behind buildings.

This case was simulated using the inverse distance weighting core of the immersed boundary method. We were unable to successfully use trilinear interpolation with this terrain data. Common problems were the inability to find eight appropriate neighbors and ill-conditioned Vandermonde matrices. Additionally, in some cases (such as at corners) the direction of the flux boundary condition is ambiguous or prescribed in an unintended direction.

5 CONCLUDING REMARKS

We have developed an IB method for the WRF model which is capable of handling highly complex urban terrain, as demonstrated by our idealized Oklahoma City test case. We extended the two-dimensional IB method presented in Lundquist et al. (2010) into three dimensions, and validated the implementation by simulating flow over a hill and comparing the solution to results achieved using the native terrain-following coordinate. We found that while the trilinear interpolation algorithm provided accurate results for flow over a smooth hill, the algorithm was not robust enough to be used with real urban terrain. An alternate inverse distance weighted interpolation method, which provided additional flexibility, was implemented and was also shown to

produce accurate results for the hill test case. Additionally, the method proved to be robust enough to allow simulations of flow over real urban terrain data.

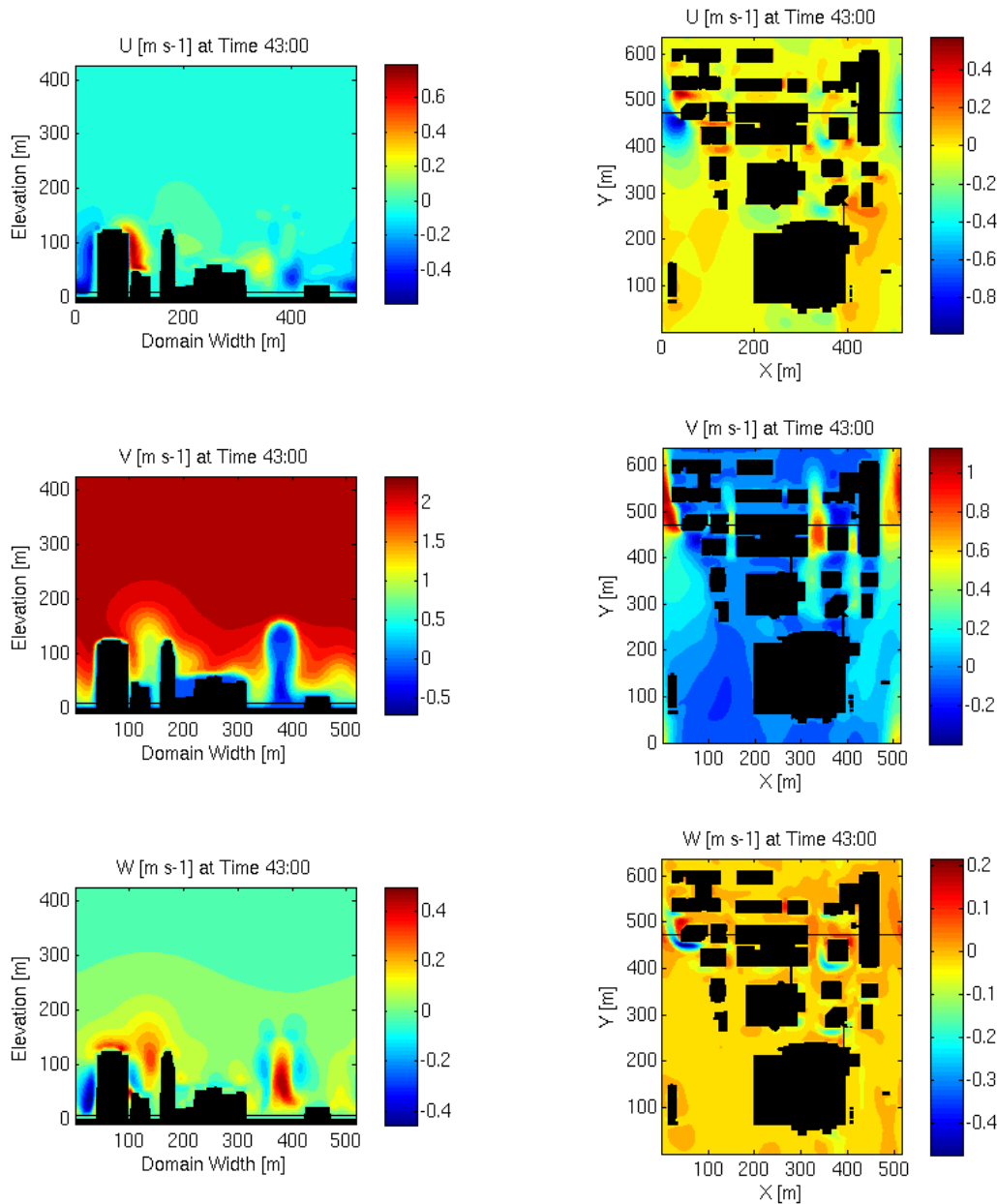


Figure 6. Contours are shown for all three velocity components. Side profiles are show in the left column, and the location in y is marked by a horizontal line placed in the top view in the right column. Likewise, contours of velocity through a horizontal plane are shown as a top view in the right column. The location of the horizontal plane is marked with a line through the side profile in the left column.

Additional work is in progress to enable comparisons with the Joint Urban 2003 field campaign. First, a constant eddy viscosity is used in the simulations presented here. A turbulence closure is needed for comparisons to field data, and we have not verified the use of our IBM while using a turbulence closure. Second, this work uses idealized periodic lateral boundary conditions which do not represent real meteorological data. Better choices for the lateral boundary conditions would be to either specify the inlet flow with data from a fully developed neutral

boundary layer simulation or to use WRF's nesting capabilities to provide lateral boundary conditions from a mesoscale simulation. Resolution of these issues will allow us to seamlessly integrate the IBM method into the current WRF framework, enabling simulation of a wide variety of cases with steep terrain.

6 ACKNOWLEDGEMENTS

The first author is grateful to the Lawrence Scholars Program at the Lawrence Livermore National Laboratory. This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

7 REFERENCES

- Allwine, K. and J. Flaherty 2006: Joint Urban 2003: Study Overview and Instrument Locations. Tech. Rep. PNNL-15967, Pacific Northwest National Laboratory, Richland, WA.
- Franke, R., 1982: Scattered data interpolation: tests of some methods, *Math. Comput.*, **38**, 181–200.
- Gao, T., Y. Tseng, and X. Lu, 2007: An improved hybrid Cartesian/immersed boundary method for fluid-solid flows. *Int. J. Numer. Meth. Fluids*, **55**, 1189–1211.
- Janjic, Z., 1977: Pressure gradient force and advection scheme used for forecasting with steep and small scale topography. *Beitr. Phys. Atmos.*, **50**, 186–189.
- Klemp, J., W. Skamarock, and O. Fuhrer, 2003: Numerical consistency of metric terms in terrain-following coordinates. *Mon. Wea. Rev.*, **131**, 1229–1239.
- Lundquist, K., F. Chow, and J. Lundquist, 2010: An immersed boundary method for the Weather Research and Forecasting model, *Mon. Wea. Rev.*, **138**, 796-817.
- Mohd-Yusof, J., 1997: Combined immersed boundary/b-spline methods for simulations of flow in complex geometry. Center for Turbulence Research, Annual Research Briefs, NASA Ames/Stanford University, 317–327.
- Peller, N., A. Le Duc, F. Tremblay, and M. Manhart, 2006: High-order stable interpolations for immersed boundary methods, *Int. J. Numer. Meth. Fluids*, **52**, 1175-1193.
- Tseng, Y. and J. Ferziger, 2003: A ghost-cell immersed boundary method for flow in complex geometry. *J. Comp. Phys.*, **192**, 593–623.
- Zangl, G., L. Gantner, G. Hartjenstein, and H. Noppel, 2004: Numerical errors above steep topography: A model intercomparison. *Meteorol. Z.*, **13**, 69-76.