7-8-2012

# Toward a GPU-Accelerated Immersed Boundary Method for Wind Forecasting Over Complex Terrain

Rey DeLeon
*Boise State University*

Kyle Felzien
*Boise State University*

Inanc Senocak
*Boise State University*

Proceedings of the ASME 2012 Fluids Engineering Summer Meeting
FEDSM2012
July 8-12, 2012, Rio Grande, Puerto Rico

FEDSM2012-72145

# TOWARD A GPU-ACCELERATED IMMERSED BOUNDARY METHOD FOR WIND FORECASTING OVER COMPLEX TERRAIN

**Rey DeLeon*, Kyle Felzien†, and Inanc Senocak***

*Department of Mechanical & Biomedical Engineering
†Department of Computer Science
Boise State University
Boise, Idaho 83725-2075

## ABSTRACT

*A short-term wind power forecasting capability can be a valuable tool in the renewable energy industry to address load-balancing issues that arise from intermittent wind fields. Although numerical weather prediction models have been used to forecast winds, their applicability to micro-scale atmospheric boundary layer flows and ability to predict wind speeds at turbine hub height with a desired accuracy is not clear. To address this issue, we develop a multi-GPU parallel flow solver to forecast winds over complex terrain at the micro-scale, where computational domain size can range from meters to several kilometers. In the solver, we adopt the immersed boundary method and the Lagrangian dynamic large-eddy simulation model and extend them to atmospheric flows. The computations are accelerated on GPU clusters with a dual-level parallel implementation that interleaves MPI with CUDA. We evaluate the flow solver components against test problems and obtain preliminary results of flow over Bolund Hill, a coastal hill in Denmark.*

*Keywords: CUDA, GPU, immersed boundary method, large-eddy simulation, wind forecasting*

## INTRODUCTION

There is a growing interest to increase the portion of renewable energy resources in overall electricity production. This interest is driven by energy security, environmental concerns and economics. Among several renewable energy options, wind energy has a greater potential within the renewable energy portfolio. But increasing the percentage of wind energy in overall energy production is much more complex than simply installing wind farms in windy areas. In their 20 % Wind by 2030 technical report [1], Department of Energy (DOE) has described the many challenges of increasing the amount of wind power forecasting, grid integration, manufacturing and economics.

In this paper, we describe our efforts toward developing a micro-scale wind forecasting capability where turnaround time of complex terrain wind flow simulations need to be accelerated all the way from pre-processing stage to post-processing stage. In micro-scale atmospheric flows computational domain size can range from several hundred meters to several kilometers, and grid resolution is on the orders of tens of meters to be able to resolve the wind profile at the turbine hub height ($\sim 80m$). At these spatial scales, large-eddy simulation (LES) technique is applicable, but it needs to be augmented with near-surface models [2] to address surface roughness, insufficient spatial resolution in the vicinity of the surface, and fluxes of heat and moisture. A viable wind forecasting capability should also address the overall computational expense of performing LES with a parallel implementation. To this end, clusters of modern graphics processing units (GPU) emerge as a promising hardware solution to help realize micro-scale wind forecasting. Mesh generation for complex terrain is also a bottleneck in wind forecasting, where a poor

mesh quality can induce substantial numerical errors. Creating a high quality mesh can be very time consuming and may not be an ideal approach in wind resource identification studies. Therefore, we propose to adopt a Cartesian immersed boundary (IB) method to automate and accelerate the mesh generation process in wind flow simulations over complex terrain. Equally important, Cartesian mesh topology maps well on to the computer architecture of modern GPUs for efficient parallel computing.

The immersed boundary (IB) method was first proposed by Peskin [3] to simulate blood flow in a heart using a Cartesian mesh. Particularly, the IB method is desirable when handling complex geometries to avoid the cumbersome task of generating body-fitted grids [4]. The major issue in the method proposed by Peskin was formulating a proper body force to impose boundary conditions accurately without jeopardizing the numerical stability. The direct forcing method, proposed by Mohd-Yusof [5] and later applied by Verzicco et al. [6], remedied this issue by reconstructing the velocity field around the immersed boundary. This technique was successfully applied by Fadlun et al. [7], Verzicco et al. [6], Iaccarino and Verzicco [8] for engineering fluid flow applications at moderate Reynolds numbers. Senocak et al. [9] extended the direct forcing approach to atmospheric boundary layer simulations over a flat terrain by adopting the same length scale assumptions in the turbulence model and reconstruction schemes. Lundquist et al. [10] presented a 2D implementation of the direct forcing immersed boundary (IB) method within Weather Research and Forecasting Model for analytical geometries without any consideration for turbulence modeling and turbulent stresses at the immersed surfaces.

In order to faithfully extend the immersed boundary method to atmospheric flows, we need to represent arbitrarily complex three-dimensional topography on a Cartesian mesh, address turbulence modeling using either Reynolds-averaged or LES approaches with provisions for atmospheric stability, represent turbulent stresses on immersed surfaces and incorporate land-surface fluxes of heat and moisture. Thus far, no study that addresses these issues in a unified fashion has appeared in literature.

The graphics processing unit (GPU) has become the new paradigm in supercomputing. As of November 2011, three of the top five supercomputers in the world adopt GPUs as accelerators [11]. GPU computing have made a quantum leap forward with the debut of NVIDIA's Compute Unified Device Architecture (CUDA) [12] in 2007. CUDA-enabled GPUs provide researchers with a massively parallel many-core architecture that could be programmed with the CUDA C programming language easily. CUDA's popularity among scientists and engineers has led to new initiatives in developing compilers and software to extend CUDA to conventional multi-core CPUs. Ocelot [13] and CUDA-x86 [14] are recent examples that enables CUDA codes to execute on central processing units (CPU).

GPU computing has been adopted in numerous fields [15]. Within the computational fluid dynamics (CFD) field, Thibault and Senocak [16] developed one of the first multi-GPU parallel incompressible Navier-Stokes solver for a Cartesian mesh, which was later extended to GPU clusters by Jacobsen et al. [17] with an MPI-CUDA implementation. A full-depth amalgamated parallel 3D geometric multigrid technique was introduced into the present solver recently [18]. Brandvik and Pullan [19] created a GPU-accelerated 3D Navier-Stokes solver for turbomachinery, and Corrigan et al. [20] explored techniques to accelerate execution of unstructured mesh CFD codes on the GPU. Griebel and Zaspel [21] developed a 3D incompressible solver for two-phase flows, and Geveler et al. [22] created libraries for Lattice-Boltzmann based CFD applications on multi- and many-core computing platforms.

## NUMERICAL METHODOLOGY

The overall goal of our implementation is to forecast micro-scale atmospheric flows over complex terrain. To accomplish this, we adopt the LES technique with the Lagrangian dynamic subgrid scale model [23] for incompressible flows because it is a localized model and does not require any homogeneous directions in the computational domain. We extend the immersed boundary method with a reconstruction scheme similar to the one described by Gilmanov et al. [24–26] to atmospheric flows and apply it to real terrain. To accelerate the turnaround time of our simulations, we implement the entire wind forecasting model by interleaving Message Passing Interface (MPI) with CUDA and deploy GPU clusters for computations.

### Governing Equations

The basic principle of LES is to separate a turbulent flow field into large- and subgrid-scales (SGS) using a mathematical filter. The large-scales are resolved whereas the subgrid-scales are treated as statistically universal and their interaction with the resolved flow is modeled. The governing equations used in LES of incompressible flows are the filtered Navier-Stokes equations,

$$\frac{\partial \overline{u}_j}{\partial x_j} = 0 \tag{1}$$

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{u}_i \overline{u}_j\right) = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2\nu \overline{S}_{ij} - \tau_{ij}\right) \tag{2}$$

where

$$S_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right) \tag{3}$$

is the deformation tensor, and

$$\tau_{ij} = \overline{u_i u_j} - \overline{u}_i \overline{u}_j \tag{4}$$

is the tensor representing the interaction of the subgrid-scales on the resolved large-scales. The overbar represents a filtered quantity. We use the projection algorithm [27] to numerically solve the incompressible Navier-Stokes equations in an explicit fashion. Second order central difference and Adams-Bashforth schemes are applied to discretize the spatial and temporal terms in the governing equations, respectively. The pressure Poisson equation is solved using a parallel full-depth, three-dimensional geometric multigrid method with an amalgamation strategy developed for GPU clusters [18]. The amalgamation strategy gathers the computational domain on a single GPU when the multigrid method can no longer coarsen the mesh per GPU on a cluster.

## Subgrid-scale Modeling

The SGS model used in this study is the Lagrangian dynamic Smagorinsky model proposed by Meneveau et al. [23], which is adequate for arbitrarily complex geometry. The model is based on the Smagorinsky eddy-viscosity model [28], and the model coefficient is calculated dynamically and locally from a Lagrangian perspective using information along flow pathlines. The advantage of this dynamic procedure is it does not require statistically homogeneous directions [29, 30] and is more practical to implement than other localized models [31]. The Smagorinsky eddy-viscosity model is given by

$$\tau_{ij} = -2\nu_t \overline{S}_{ij} + \frac{1}{3}\tau_{ii}\delta_{ij} = -\nu_t \left( \frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \right) + \frac{1}{3}\tau_{ii}\delta_{ij} \quad (5)$$

where

$$\nu_t = (C_S \Delta)^2 \sqrt{2\overline{S}_{ij}\overline{S}_{ij}} \quad (6)$$

with $\Delta$ being the filter width and $C_S$ the model coefficient.

Dynamic models are based upon the Germano identity [29], which applies a second filter to the already filtered quantities, denoted by hat, and is given by

$$L_{ij} = T_{ij} - \widehat{\tau_{ij}} = \widehat{\overline{u}_i \overline{u}_j} - \widehat{\overline{u}}_i \widehat{\overline{u}}_i \quad (7)$$

The dynamic models take a form similar to

$$C_S^2 = \frac{\langle L_{ij} M_{ij} \rangle}{\langle M_{ij} M_{ij} \rangle} \quad (8)$$

where the plane-averaging operations, denoted by the angle brackets, stabilize the model [30, 32]. The tensor, $M_{ij}$, is defined by

$$M_{ij} = 2\Delta^2 \left[ \widehat{|\overline{S}|\overline{S}_{ij}} - \alpha^2 |\widehat{\overline{S}}| \widehat{\overline{S}}_{ij} \right] \quad (9)$$

The plane-averaging approach has the downside of requiring homogeneous directions. One approach to alleviate this issue is to take a Lagrangian perspective and average of flow pathlines by using a Lagrangian dynamic model as proposed by Meneveau et al. [23]. In the Lagrangian dynamic model, the value of $C_S$ is found by solving two transport-relaxation equations that are a result of backward time integration and an exponential weighting function that decreases the weight of past events and are given by

$$\frac{\partial \mathcal{J}_{LM}}{\partial t} + \overline{\mathbf{u}} \cdot \nabla \mathcal{J}_{LM} = \frac{1}{T}\left(L_{ij}M_{ij} - \mathcal{J}_{LM}\right) \quad (10)$$

$$\frac{\partial \mathcal{J}_{MM}}{\partial t} + \overline{\mathbf{u}} \cdot \nabla \mathcal{J}_{MM} = \frac{1}{T}\left(M_{ij}M_{ij} - \mathcal{J}_{MM}\right) \quad (11)$$

where $T$ is the relaxation time scale. The choice of time scale given in Meneveau et al. is

$$T = 1.5\Delta \left( \mathcal{J}_{LM} \mathcal{J}_{MM} \right)^{-1/8} \quad (12)$$

The value of $C_S$ is then calculated by the relation,

$$C_S^2 = \frac{\mathcal{J}_{LM}}{\mathcal{J}_{MM}} \quad (13)$$

Using first-order numerics, Eq. 10 and Eq. 11 can be discretized as

$$\mathcal{J}_{LM}^{n+1}(\mathbf{x}) = H\left\{ \varepsilon \left[L_{ij}M_{ij}\right]^{n+1}(\mathbf{x}) + (1-\varepsilon)\, \mathcal{J}_{LM}^{n}(\mathbf{x} \quad \overline{\mathbf{u}}^n \Delta t) \right\} \quad (14)$$

$$\mathcal{J}_{MM}^{n-1}(\mathbf{x}) = \varepsilon \left[M_{ij}M_{ij}\right]^{n-1}(\mathbf{x}) + (1-\varepsilon)\, \mathcal{J}_{MM}^{n}(\mathbf{x} - \overline{\mathbf{u}}^n \Delta t) \quad (15)$$

where the ramp function, $H\{x\}$, prevents negative values of $C_S^2$, $T$ is the same as in Eq. 12, and

$$\varepsilon = \frac{\Delta t / T^n}{1 + \Delta t / T^n} \quad (16)$$

The value at $\mathbf{x} - \overline{\mathbf{u}}^n \Delta t$ can be determined using multilinear interpolation.

## Immersed Boundary Method

In the IB method, a solid boundary is represented by a forcing term in the momentum equations given in Eq. 2. The discretized form of the u-component of the momentum equation is

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = RHS_i + F_i \tag{17}$$

where $RHS_i$ includes the pressure gradient, convective and diffusive terms. Using the direct forcing method [5, 7], the velocity at the boundary can be prescribed as $u_i^{n+1} = V_i^{n+1}$ then the body force becomes

$$F_i = -RHS_i + \frac{V_i^{n+1} - u_i^n}{\Delta t} \tag{18}$$

With this approach, the body force can be taken into account implicitly by prescribing the velocity field. However, the complex geometry boundaries are not coincident with the Cartesian grid and reconstruction schemes are required to impose the proper velocity boundary conditions on grid points near the solid geometry. The steps in applying the IB method within the projection algorithm are summarized as follows

1. In the preprocessing stage, separate the Cartesian cells as solid, fluid, or immersed boundary (IB). Determine the necessary parameters for the velocity field reconstruction schemes.
2. Predict the velocity by solving the momentum equations as per the projection algorithm.
3. Set the solid Cartesian cells to zero and apply reconstruction scheme to IB nodes.
4. Solve the pressure Poisson equation by imposing divergence free condition.
5. Correct the velocity field and set solid cells to zero.

Several reconstruction methods have been proposed in literature. For our application we chose to use a reconstruction scheme similar to the hybrid Cartesian/immersed boundary approach proposed by Gilmanov et al. [24–26], and we extend it to atmospheric flows with a rough surface following the approach described in [9].

For laminar flow regimes, the reconstruction scheme consists of linear interpolation along a line normal to the solid surface. This method is intended for stereolithography (STL) CAD geometry files where a surface is represented by an unstructured mesh with triangular elements. Each triangular element is defined by the coordinates of the vertices and a surface normal. Although the IB method can be implemented for analytical geometries (e.g., circle, sphere, etc.) its extension to arbitrarily complex geometries requires the development of preprocessor to

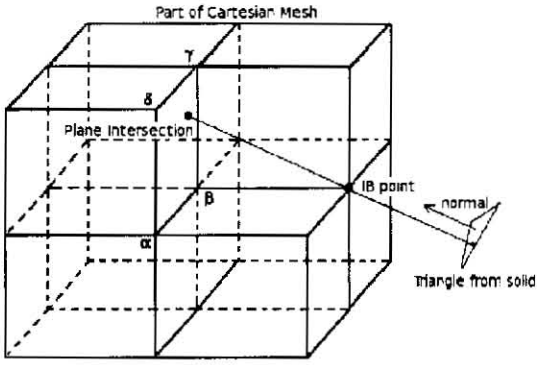


**FIGURE 1**. SKETCH OF THE CURRENT RECONSTRUCTION SCHEME AT AN IB POINT, WHERE A LINE IS PROJECTED ALONG THE NORMAL DIRECTION OF THE NEAREST TRIANGULAR ELEMENT INTO THE FLUID DOMAIN.

calculate the intersection of the CAD geometry with the underlying Cartesian mesh and one needs to pay special attention when working with three dimensional geometries. Therefore, we describe the preprocessing stage of the IB method with sufficient detail.

In the preprocessing stage, all Cartesian points within the small search radius, $ds_0$, from the boundary are identified. The value of the search radius is determined by averaging the grid spacings: $dx$, $dy$ and $dz$. The position vectors of these points, $r_{NB}$, are compared to the position vector of the $m$th triangular element's centroid, $r_{m-1/2}$, until the following condition is satisfied

$$\min_{m=1,M} |r_{NB} - r_{m+1/2}| < ds_0 \tag{19}$$

Any point that satisfies the above condition is called a near-boundary node. Note that near-boundary nodes can be either internal or external to the solid boundary.

The next step is to determine which of the near-boundary points are actually within the solid. For every near-boundary point, all triangular elements located within the search radius centered at the near-boundary node are identified. Examining the sign of the scalar product, $n_{m+1/2} \cdot (r_{nb} - r_{m+1/2})$, determines whether a near-boundary point is internal or external to the solid boundary. If $n_{m+1/2} \cdot (r_{nb} - r_{m-1/2}) > 0$ for *at least one* triangular element within the sphere from the point, then the Cartesian mesh point is external to the body and flagged as an IB node. If $n_{m-1/2} \cdot (r_{nb} - r_{m+1/2}) < 0$ for *all* triangular elements within the sphere from the point, the Cartesian mesh point is internal to the body and flagged as solid. All points residing in the solid can be identified by checking which points are between two nodes that are internal to the solid.

After IB nodes are identified, a line, which we refer to as the IB line, along the surface normal direction of the nearest triangular element can be projected from the IB node as shown in Fig. 1. The IB line that originates from the IB node is extended into the fluid domain to intersect Cartesian cell faces. The values of neighboring Cartesian grid points are interpolated on to the IB line where it intersects the Cartesian cell face. The velocity at the IB node is then found by an interpolation between the known boundary value at the surface and the values that are interpolated on the IB line. Multilinear interpolation of the points $\alpha$, $\beta$, $\gamma$ and $\delta$ can be used to find the value of a quantity (e.g. velocity, pressure) on the IB line. For laminar flow conditions, the reconstruction of a quantity at the IB node is accomplished by linear interpolation between the point of intersection along the IB line and the value of the boundary condition at the surface.

The linear interpolation reconstruction scheme may also work well for turbulent flows if the grid resolution is fine enough to capture the viscous sublayer, but this is never the case for atmospheric flows with a rough underlying surface. A linear interpolation scheme could underestimate the surface stresses, because a logarithmic or power wind profile is typically observed in atmospheric flows. One also has to consider how the reconstruction scheme influences the subgrid model and must maintain consistency between the underlying assumptions in the turbulence model and the immersed boundary reconstruction scheme to obtain satisfactory results [9].

The complex terrain in atmospheric flows is characterized by roughness, sensible and latent heat fluxes, atmospheric stability, and moisture fluxes, all of which play a major role in the observed wind profiles. Typically, the boundary conditions are imposed through stress and flux terms

$$\tau = \rho \overline{u'w'} \tag{20}$$

$$H = \rho C_p \overline{w'\Theta'} \tag{21}$$

$$E = \rho \overline{w'q'} \tag{22}$$

where $\tau$ is the turbulent stress at the surface, $H$ and $E$ are the fluxes for heat and moisture, respectively. $u'$, $w'$, $\Theta'$, $q'$ represents the fluctuations of streamwise wind, vertical wind, potential temperature and moisture, respectively. Direct implementation of these terms in Eq. 20-22 within an immersed boundary method would be tedious and can complicate the IB method, which has historically become popular due to its simplicity in implementation. Therefore, we prefer that the reconstruction schemes should operate only on the primitive variables (i.e., $u$, $v$, $w$, $\Theta$, $q$) to retain the simplicity of IBM for atmospheric flows computations. We consider the log-law reconstruction scheme [9] because of its consistency with the Monin-Obukhov similarity theory that is also used in turbulence model assumptions. This scheme, which is applied only to the $u$ component of velocity for ease of presentation, is

$$u_1 = u_2 \left[ \frac{\ln(z_1/z_0)}{\ln(z_2/z_0)} \right] \tag{23}$$

where $z_0$ is the equivalent roughness height at the boundary, $z_2$ and $z_1$ are the normal distances to the surface along the IB line as shown in Fig. 1. Note that the above scheme is suggested for neutrally stratified conditions, but it can be extended to atmospheric conditions with stable and unstable stratification.

## Dual-Level MPI-CUDA Parallel Implementation

The IB method and LES capability was integrated into a dual-level parallel CFD code [16–18]. The code is entirely designed for GPU clusters using MPI and CUDA and it overlaps GPU to CPU data copies and cluster communication with computations on the GPU. A one dimensional domain decomposition is used for parallel computations. First, a 3D Cartesian domain of NX×NY×NZ is decomposed in the z-direction such that the subdomains are of size NX×NY×NZ/nGPU, where nGPU is the number of GPUs. The 3D arrays are restructured into 1D arrays for efficient memory transfers between host (CPU) and device (GPU). Each partial domain on a GPU is further decomposed into three parts and flagged as *top*, *middle* and *bottom*, with top and bottom being the edges of a subdomain. This strategy allows the GPU to execute the middle section computations simultaneously with the host-device transfers of the top and bottom sections and network communications. This overlapping strategy make use of asynchronous memory copy operations with CUDA streams.

In our previous work, the performance and parallel scaling of our flow solver was demonstrated with the lid-driven cavity benchmark problem [16, 17] using simple numerical methods with no turbulence modeling capability. For performance evaluation purposes, we developed a parallel implementation of the flow solver with Pthreads on CPUs to benchmark the relative speedup observed in the GPU version of the flow solver. We demonstrated that the GPU version performance using two Tesla cards on the S1070 unit on the Lincoln cluster at National Center for Supercomputing Applications(NCSA) is 26 times faster than the CPU version of the flow solver executed parallel on 8 CPUs with Intel Xeon 2.33 GHz processors. Because the development of a flow solver with LES turbulence models and advanced linear solvers is an intensive software development task, we have discontinued maintaining the CPU version of our flow solver. Therefore we do not provide speedup results for the current GPU version of our flow solver. We refer the reader to our previous work on development of an incompressible flow solver for GPUs for further details [16–18, 33].

The pressure Poisson equation resulting from the projection algorithm is solved by a full-depth, parallel 3D geometric multi-

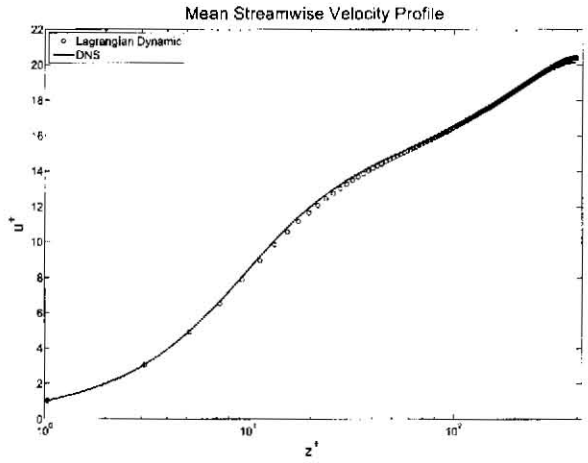


**FIGURE 2**. THE MEAN VELOCITY PROFILE NORMALIZED WITH FRICTION VELOCITY VS. WALL UNITS FOR A MESH SIZE OF $192\times128\times384$. RESULTS ARE COMPARED TO THE DNS BY MOSER ET AL. [36].

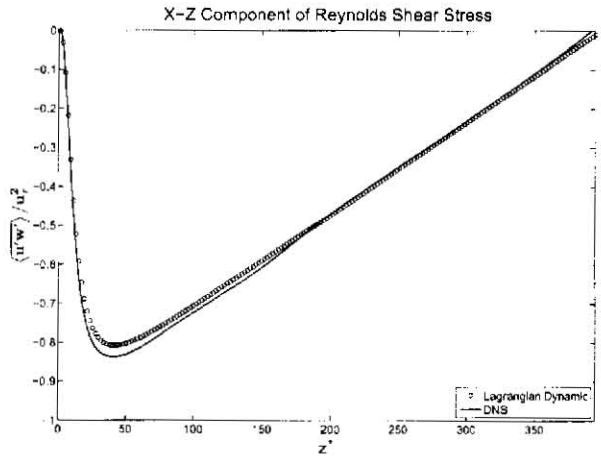


**FIGURE 3**. THE X-Z COMPONENT OF REYNOLDS STRESS NORMALIZED WITH FRICTION VELOCITY VS. WALL UNIT OF A MESH SIZE OF $192\times128\times384$. RESULTS ARE COMPARED TO THE DNS BY MOSER ET AL. [36].

grid solver that employs an amalgamation strategy to avoid truncation of the multigrid cycle [18]. The multigrid technique is known for its excellent numerical convergence rate and can be separated in four parts: smoothing, restriction, coarse grid solve, and prolongation [34, 35]. The smoothing operation in our implementation is a weighted Jacobi method with weighting coefficient of 0.86. The restriction stage coarsens a mesh by using a full-weighted scheme. The restriction process repeats until coarsening is no longer possible at which point the problem is directly solved on the coarse grid. The solution is then interpolated back to the original mesh in the prolongation stage. Truncation of the restriction process greatly reduces the convergence rate which is an issue that arises when implementing multigrid in parallel. The full-depth grid coarsening cannot reach the coarsest grid on distributed processes that leads to data starvation for each process. Therefore, our implementation employs an amalgamation strategy that brings the coarsened subdomains to a single GPU process and reaches the full-depth in the multigrid cycle.

## RESULTS AND DISCUSSION

Our goal is to provide a forecasting capability for wind flow over complex terrain. Before demonstrating wind flow over complex terrain, we first validate our implementation of the Lagrangian dynamic LES model, which is suitable for arbitrarily complex geometries. We simulate a turbulent channel flow at $\mathrm{Re}_\tau = 395$. The computational domain has dimensions of $(2\pi\delta, \pi\delta, 2\delta)$ in $(x,y,z)$ where $x$, $y$, and $z$ are the streamwise, spanwise and wall-normal directions, respectively. Periodic boundary conditions are applied to the stream- and spanwise directions and the no-slip condition is enforced at the walls. A constant, height-independent forcing maintains the flow. The computation mesh used was $192\times128\times384$ in $x$, $y$, and $z$, respectively. We do not use grid stretching because we plan to adopt adaptive mesh refinement in the future, and therefore have more grid points than a traditional LES for the present test case. The flow was allowed to develop for 60 dimensionless time units $(u_\tau t/\delta)$ and turbulent flow statistics were gathered for 20 time units thereafter.

Figures 2 and 3 show the mean streamwise velocity profile and the Reynolds shear stress results from our turbulent channel flow LES. We compare our results against the direct numerical simulation (DNS) performed by Moser et al. [36]. As shown in these figures our results agree well with the DNS data. We note that LES computations were performed using 8 GPUS on 4 nodes. The entire simulation completes in 45 hours.

### Immersed Boundary Results

To validate our IB method implementation, we consider a laminar flow over a circular cylinder. We demonstrate our ability to simulate flow over complex terrain by considering the Bolund Hill located in Roskilde Fjord, Denmark. Bolund Hill is a 12-meter-high small hill completely surrounded by water except for a small isthmus leading to the mainland. Because of its small shape and isolation, Bolund Hill has been the focus of studies in recent years, both experimental [37] and computational [38, 39].

Laminar flow over a circular cylinder was chosen to validate our immersed boundary implementation. Reynolds numbers of 20 and 40 were considered. The domain was $31D \times 24D$ in the

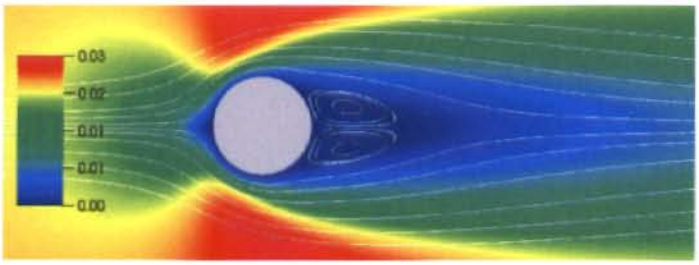


**FIGURE 4**. STREAMLINES OF FLOW OVER A CIRCULAR CYLINDER AT REYNOLDS NUMBER 20 FOR A DOMAIN OF $31D \times 24D$.

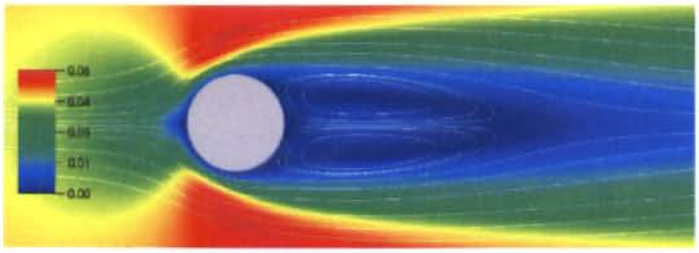

**FIGURE 5**. STREAMLINES OF FLOW OVER A CIRCULAR CYLINDER AT REYNOLDS NUMBER 40 FOR A DOMAIN OF $31D \times 24D$.

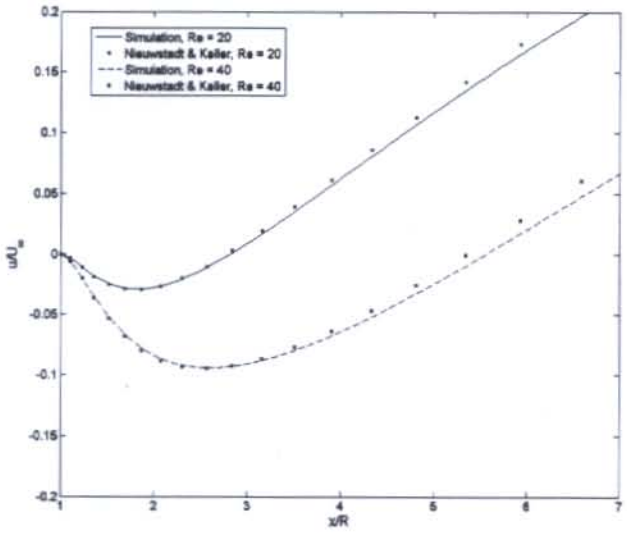


**FIGURE 6**. U-COMPONENT OF CENTERLINE VELOCITY IN THE WAKE OF FLOW OVER A CIRCULAR CYLINDER FOR REYNOLDS NUMBERS OF 20 AND 40 IN A DOMAIN OF $31D \times 24D$. RESULTS ARE COMPARED TO NIEUWSTADT AND KELLER [40].

x- and z-directions, respectively, where $D$ is the diameter of the cylinder. The center of the cylinder was placed at $10.5D$ in the x-direction and at halfway point in the z-direction. The boundary conditions of the domain were freeslip at the top and bottom, with an inlet and convective outlet in the streamwise direction. A linear reconstruction scheme was applied to this flow scenario. Figures 4 and 5 present the streamlines around the cylinder. Figure 6 shows the u-component of the centerline velocity in the wake. We compared our results with the computations of Nieuwstadt and Keller [40]. We observe a very good agreement in the near wake for both Reynolds numbers. Our results slightly deviate from Nieuwstadt and Keller in the far wake which is evident in both cases. We found that this deviation in the far wake mainly depends on the overall computational domain size. We note that the agreement in near wake of cylinder is very good, and it serves as a good validation case for our immersed boundary method implementation.

Figure 7 is a surface rendering of the Bolund Hill STL file used in this paper. The feature that makes the Bolund Hill case challenging to simulate is the steep vertical escarpment. Figure 8 shows a slice of the Cartesian mesh used superimposed on the STL at the escarpment. In our solver, the $x$ and $y$ directions correspond to the lateral directions and $z$ to the vertical direction, with the $x$ direction being perpendicular to the escarpment in Fig. 7. The computational Cartesian mesh used in this paper was $256 \times 192 \times 128$ in the $x$, $y$ and $z$ directions, respectively, with a lateral resolution of 4 m and vertical resolution of 1 m. A no-slip condition was imposed at the terrain surface with a free-slip condition at the top of the domain and periodic lateral boundary conditions.

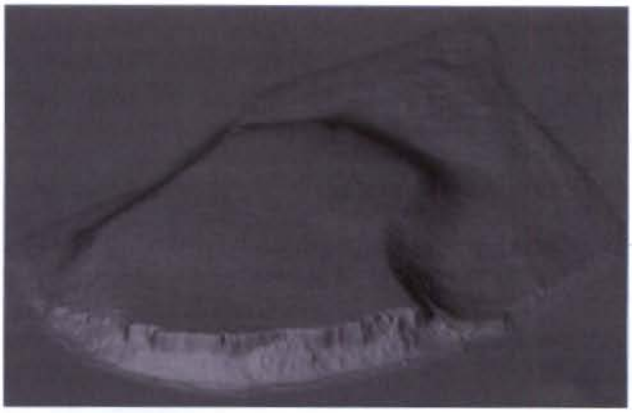

**FIGURE 7**. THE SURFACE CREATED BY THE STL GEOMETRY OF BOLUND HILL USED IN THIS PAPER.

Performing a pure LES of atmospheric flows over complex terrain would be computationally impractical given the amount of grid points would be needed to resolve the flow near the surface. Furthermore, LES requires additional models to represent

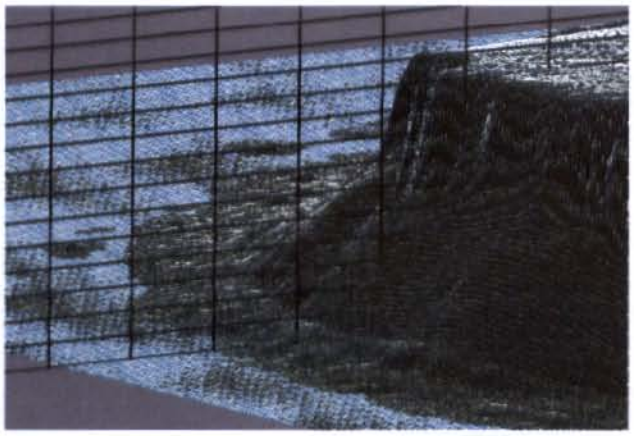

**FIGURE 8**. CLOSEUP OF A CARTESIAN MESH SLICE IN THE X-Z PLANE SUPERIMPOSED ON THE BOLUND HILL STL. ONE CELL HAS DIMENSIONS OF 4 METERS IN THE X AND 1 METER IN THE Z.

surface roughness and fluxes of heat and moisture. Therefore, we use a hybrid RANS/LES [2,41] approach that provides a smooth transition between RANS and LES regions by using

$$\nu_t = \left[\left(1-\exp\left(\frac{-z}{h}\right)\right)^2 (C_S\Delta)^2 + \exp\left(\frac{-z}{h}\right)^2 (\kappa z)^2\right] |S| \tag{24}$$

to blend the length scales produced from the Lagrangian dynamic SGS model with a mixing length RANS model. The equivalent roughness height for the logarithmic reconstruction was 0.0003 m which is suggested for the water surrounding Bolund Hill.

Complex terrain results that are presented in this paper are preliminary at this stage. For now, we present a velocity field along the x-direction perpendicular to the escarpment in Fig. 9. The flow field is showing acceleration at the escarpment and there is evidence of wake formation. However, there are several issues we will improve upon. First, the periodic lateral boundary conditions should be replaced with open lateral boundary conditions which would come from a precursor simulation that would also provide an initial turbulent field. The next issue to address is to apply the correct equivalent roughness height in the logarithmic reconstruction to reflect the roughness of the hill separately from the roughness of the sea. A third issue would be to perform grid refinement study and investigate the convergence of the solution. Last but not least, we need to systematically evaluate subgrid-scale models for wind over complex terrain.

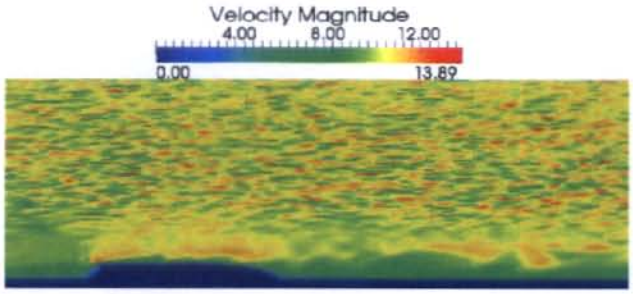


**FIGURE 9**. A SLICE OF THE INSTANTANEOUS VELOCITY FIELD NORMAL TO THE Y DIRECTION. THE SIMULATION EXHIBITS SPEEDUP AT THE VERTICAL ESCARPMENT AND THE BEGINNING OF WAKE FORMATION.

## SUMMARY AND FUTURE WORK

We have presented our on-going work to develop a GPU-accelerated forecasting capability for wind flow over complex terrain. The wind solver adopts a dual-level (MPI-CUDA) parallelism for GPU clusters. The pressure Poisson equation resulting from the projection algorithm is solved with a full-depth amalgamated geometric multigrid method. To represent the complex geometry of real terrain in three dimensions we have developed a preprocessor such that an IB reconstruction scheme can be implemented along the surface normal direction in conjunction with atmospheric similarity theory. For turbulence modeling over complex terrain, we consider LES approach with dynamic subgrid-scale models. In particular we use the Lagrangian dynamic model because it is a localized model and does not require any homogeneous direction in the dynamic procedure to calculate the model coefficient. We validated the LES capability with a turbulent channel flow and the IBM with laminar flow over circular cylinders.

IB computations over complex terrain that are presented in this paper are preliminary and still being refined. The focus of ongoing and future work will be to use open lateral boundary conditions, correct the equivalent roughness height over the hill and perform a grid refinement study. Our current study provides the necessary components for simulating wind over complex terrain using an immersed boundary method and future work will produce a quantitative analysis of both solution accuracy and computational performance of our flow solver.

## REFERENCES

[1] US DoE, 2008. 20% wind energy by 2030: Increasing wind energy's contribution to US electricity supply. Tech. rep., Washington, D.C.

[2] Senocak, I., Ackerman, A., Kirkpatrick, M., Stevens, D., and Mansour, N., 2007. "Study of near-surface models for large-eddy simulations of a neutrally stratified atmo-

spheric boundary layer". *Boundary-Layer Meteorology*, ***124***, pp. 405–424. 10.1007/s10546-007-9181-x.

[3] S, C., and Peskin, 1977. "Numerical analysis of blood flow in the heart". *Journal of Computational Physics*, ***25***(3), pp. 220 – 252.

[4] Mittal, R., and Iaccarino, G., 2005. "Immersed boundary methods". *Annual Review of Fluid Mechanics*, ***37***, pp. 239–261.

[5] Mohd-Yusof, J., 1997. Combined immersed boundary/B-spline methods for simulations of flow in complex geometries. Annual Research Briefs, Center for Turbulence Research, NASA-Ames/Stanford University.

[6] Verzicco, R., Mohd-Yusof, J., Orlandi, P., and Haworth, D., 2000. "Large eddy simulation in complex geometric configurations using boundary body forces". *AIAA Journal*, ***38***, pp. 427–433.

[7] Fadlun, E., Verzicco, R., Orlandi, P., and Mohd-Yusof, J., 2000. "Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations". *Journal of Computational Physics*, ***161***(1), pp. 35 – 60.

[8] Iaccarino, G., and Verzicco, R., 2003. "Immersed boundary technique for turbulent flow simulations". *Applied Mechanics Review*, ***56***, pp. 331–347.

[9] Senocak, I., Ackerman, A., Stevens, D., and Mansour, N., 2004. Topography modeling in atmospheric flows using the immersed boundary method. Annual Research Briefs, Center for Turbulence Research, NASA-Ames/Stanford University.

[10] Lundquist, K., Chow, F., and Lundquist, J., 2010. "An immersed boundary method for the weather research and forecasting model". *Monthly Weather Review*, ***138***, pp. 796–817.

[11] TOP500 Supercomputing Site, 2011. Top 10 systems. `http://www.top500.org`.

[12] NVIDIA, 2011. *NVIDIA CUDA C Programming Guide 4.0*.

[13] Kerr, A., Diamos, G., and Yalamanchili, S., 2011. gpuocelot - a dynamic compilation framework for PTX. `http://code.google.com/p/gpuocelot/`.

[14] The Portland Group Inc., 2011. PGI CUDA-x86. `http://http://www.pgroup.com/resources/cuda-x86.htm`.

[15] Owens, J., Houston, M., Luebke, D., Green, S., Stone, J., and Phillips, J., 2008. "GPU computing". *Proceedings of the IEEE*, ***96***(5), May, pp. 879 –899.

[16] Thibault, J. C., and Senocak, I., 2012. "Accelerating incompressible flow computations with a Pthreads-CUDA implementation on small-footprint multi-GPU platforms". *The Journal of Supercomputing*, ***59***, pp. 693–719.

[17] Jacobsen, D. A., Thibault, J. C., and Senocak, I., 2010. "An MPI-CUDA implementation for massively parallel incompressible flow computations on multi-GPU clusters". In 49th AIAA Aerospace Science Meeting.

[18] Jacobsen, D. A., and Senocak, I., 2011. "A full-depth amalgamated parallel 3D geometric multigrid solver for GPU clusters". In 49th AIAA Aerospace Science Meeting.

[19] Brandvik, T., and Pullan, G., 2011. "An accelerated 3D Navier-Stokes solver for flows in turbomachines". *Journal of Turbomachinery*, ***133***(2), p. 021025.

[20] Corrigan, A., Camelli, F. F., Löhner, R., and Wallin, J., 2010. "Running unstructured grid-based CFD solvers on modern graphics hardware". *International Journal for Numerical Methods in Fluids*, Feb.

[21] Griebel, M., and Zaspel, P., 2010. "A multi-GPU accelerated solver for the three-dimensional two-phase incompressible Navier-Stokes equations". *Computer Science - Research and Development*, ***25***, pp. 65–73. 10.1007/s00450-010-0111-7.

[22] Geveler, M., Ribbrock, D., Mallach, S., Göddeke, D., and Turek, S., 2011. "A simulation suite for Lattice-Boltzmann based real-time CFD applications exploiting multi-level parallelism on modern multi- and many-core architectures". *Journal of Computational Science*, ***2***, jan, pp. 113–123.

[23] Meneveau, C., Lund, T., and Cabot, W., 1996. "A Lagrangian dynamic subgrid-scale model of turbulence". *Journal of Fluid Mechanics*, ***319***, pp. 353 – 85.

[24] Gilmanov, A., Sotiropoulos, F., and Balaras, E., 2003. "A general reconstruction algorithm for simulating flows with complex 3d immersed boundaries on cartesian grids". *Journal of Computational Physics*, ***191***(2), pp. 660 – 669.

[25] Gilmanov, A., and Sotiropoulos, F., 2005. "A hybrid cartesian/immersed boundary method for simulating flows with 3d, geometrically complex, moving bodies". *Journal of Computational Physics*, ***207***(2), pp. 457 – 492.

[26] Gilmanov, A., and Acharya, S., 2008. "A hybrid immersed boundary and material point method for simulating 3D fluidstructure interaction problems". *International Journal for Numerical Methods in Fluids*, ***56***(12), pp. 2151–2177.

[27] Chorin, A. J., 1968. "Numerical solution of the Navier-Stokes equations". *Mathematics of Computation*, ***22***(104), pp. pp. 745–762.

[28] Smagorinksy, J., 1963. "General circulation experiments with the primitive equations". *Monthly Weather Review*, ***91***(3), pp. 99–164.

[29] Germano, M., Piomelli, U., Moin, P., and Cabot, W. H., 1991. "A dynamic subgrid-scale eddy viscosity model". *Physics of Fluids A: Fluid Dynamics*, ***3***(7), pp. 1760–1765.

[30] Lilly, D., 1992. "A proposed modification of the Germano subgrid-scale closure method". *Physics of Fluids A: Fluid Dynamics*, ***4***(3), pp. 633–35.

[31] Ghosal, S., Lund, T., Moin, P., and Akselvoll, K., 1995. "A dynamic localization model for large-eddy simulation of turbulent flow". *Journal of Fluid Mechanics*, ***286***, pp. 229–255.

[32] Piomelli, U., 1993. "High reynolds number calculations using the dynamic subgrid-scale stress model". *Physics of Fluids A: Fluid Dynamics,* **5**(6), pp. 1484–1490.

[33] Jacobsen, D. A., and Senocak, I., 2011. "Scalability of incompressible flow computations on multi-GPU clusters using dual-level and tri-level parallelism". In 49th AIAA Aerospace Science Meeting.

[34] Briggs, W. L., Henson, V. E., and McCormick, S. F., 2000. *A Multigrid Tutorial (2nd ed.).* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

[35] Trottenberg, U., Oosterlee, C., and Schüller, A., 2001. *Multigrid.* Elsevier.

[36] Moser, R. D., Kim, J., and Mansour, N. N., 1999. "Direct numerical simulation of turbulent channel flow up to $Re_\tau$ = 590". *Physics of Fluids,* **11**(4), pp. 943–945.

[37] Berg, J., Mann, J., Bechmann, A., Courtney, M., and Jrgensen, H., 2011. "The bolund experiment, part i: Flow over a steep, three-dimensional hill". *Boundary-Layer Meteorology,* **141**, pp. 219–243. 10.1007/s10546-011-9636-y.

[38] Bechmann, A., Srensen, N., Berg, J., Mann, J., and Rthor, P.-E., 2011. "The bolund experiment, part ii: Blind comparison of microscale flow models". *Boundary-Layer Meteorology,* **141**, pp. 245 271. 10.1007/s10546-011-9637-x.

[39] Jafari, S., Chokani, N., and Abhari, R. S., 2012. "An immersed boundary method for simulation of wind flow over complex terrain". *Journal of Solar Energy Engineering,* **134**(1), p. 011006.

[40] Nieuwstadt, F., and Keller, H., 1973. "Viscous flow past circular cylinders". *Computers & Fluids,* **1**(1), pp. 59 – 71.

[41] Sagaut, P., 2002. *Large Eddy Simulation for Incompressible Flows: An Introduction*, 2 ed. Springer-Verlag Berlin Heidelberg New York.