Domain decomposition based coupling between the lattice Boltzmann method and traditional CFD methods - Part II: Numerical solution to the backward facing step flow

Aditya C. Velivelli and Kenneth M. Bryden*

Department of Mechanical Engineering

Iowa State University, Ames, IA 50011

*Contact information:

Kenneth M. Bryden
1620 Howe Hall
Iowa State University
Ames, IA 50011

Email: kmbryden@iastate.edu
Ph: 1-515-294-3891
Fax: 1-515-294-3261

Abstract—The lattice Boltzmann method (LBM) and traditional finite difference methods have separate strengths when solving the incompressible Navier-Stokes equations. The LBM is an explicit method with a highly local computational nature that uses floating-point operations that involve only local data and thereby enables easy cache optimization and parallelization. However, because the LBM is an explicit method, smaller grid spacing requires smaller numerical time steps during both transient and steady state computations. Traditional implicit finite difference methods can take larger time steps as they are not limited by the CFL condition, but only by the need for time accuracy during transient computations. To take advantage of the strengths of both methods, a multiple solver, multiple grid block approach was implemented and validated for the 2-D Burgers' equation in Part I of this work. Part II implements the multiple solver, multiple grid block approach for the 2-D backward step flow problem. The coupled LBM-VSM solver is found to be faster by a factor of 2.90 (2.87 and 2.93 for $Re = 150$ and $Re = 500$, respectively) on a single processor than the VSM for the 2-D backward step flow problem while maintaining similar accuracy.

1. INTRODUCTION

Many flow problems consist of complex geometries, and this leads to difficulty in generating a single grid to cover the entire flow domain. The multiblock method of grid generation results in individual grid components corresponding to particular regions of the flow domain. The grid components adjacent to walls and obstacles in the flow domain can be fitted to the boundaries of those geometries. This multiblock arrangement leads to savings in the memory required for storing the grid because local grid refinement can be performed over certain regions of the flow domain by simply covering those regions with high-resolution grid blocks and the rest of the domain with low-resolution grid blocks. Using multiple solvers on a multiblock grid can be efficient because a given solver may be better suited for one region of the flow domain, whereas another may be better suited for another region. In this study two different solvers for incompressible flows, the lattice Boltzmann method (LBM) and the vorticity-stream function method (VSM) are coupled together to solve the backward facing step flow problem. The LBM is a numerical scheme for solving the discrete velocity Boltzmann equation from gas kinetic theory [1]. The VSM is a simple and efficient traditional procedure for solving the 2-D Navier-Stokes equations by eliminating the pressure variable and introducing the vector vorticity and scalar stream function variables. The VSM consists of a vorticity transport equation and a Poisson equation for the stream function, which allows the stream function to automatically satisfy the conservation of mass constraint.

The LBM is computationally efficient relative to traditional methods when solving time marching problems, provided the time step size is similar for both [2]. For this reason, LBM has been implemented to model a number of complex flows including

biofluid dynamics and fluid flow induced by surface tension gradients [3,4]. An earlier study performed by the authors found the cache-optimized LBM to be approximately eight times faster than traditional methods [2] when solving an unsteady convection-diffusion equation. With a cache-optimized LBM, the computational domain is partitioned into subsections that can fit in the cache, and computations are performed separately on each subsection. The cache-optimized LBM has not been previously compared to traditional methods for solving the Navier-Stokes equations for steady state cases. However, in Part I of this work, the cache-optimized LBM was compared to traditional implicit schemes for solving the 2-D steady state Burgers' equation. The results in Part I indicated that a coupled cache-optimized LBM and alternating direction implicit (ADI) scheme on a multiblock grid was 4.5 times faster than the ADI scheme on the same multiblock grid while retaining similar accuracy. For Part II, these schemes are implemented for the 2-D backward step flow where the Navier-Stokes equations are being solved. As noted earlier, the traditional method utilized here is the vorticity-stream function formulation of the Navier-Stokes equations. The vorticity transport equation can be solved efficiently using the ADI method [5], and the stream function equation, which is an elliptic partial differential equation, can be solved using cyclic reduction [6]. The study here describes the LBM for the 2-D Navier-Stokes equations, discusses the cache-optimized version of the LBM, and briefly discusses the numerical aspects of the VSM [7]. Following this, the LBM and the VSM are compared with regard to accuracy and time for computation (CPU time). The insights gained from this comparison are utilized to implement a coupled LBM-VSM solver that is superior to the individual solvers.

2. THE LATTICE BOLTZMANN METHOD

The discrete velocity Boltzmann equation describes the evolution of the particle distribution function, $f_i(\vec{x}, \vec{v}, t)$, where index i represents a specific velocity. The discrete velocity Boltzmann equation with the Bhatnagar-Gross-Krook (BGK) approximation [1] is

$$\frac{\partial f_i}{\partial t} + \vec{c}_i \cdot \nabla f_i = -\omega(f_i - f_i^{eq}) \quad (1)$$

Where f_i represent the particle distribution functions, f_i^{eq} are the equilibrium distribution functions, ω is the collision frequency, and $\vec{c}_i$ represent the velocities associated with the distribution functions.

The lattice Boltzmann equation is obtained when the discrete velocity Boltzmann equation is discretized on a uniform grid or lattice using forward Euler discretization for the time term, upwind differencing for the advection term, and downwind discretization for the collision terms (right hand side of Eq. 1). When the velocity equals the spatial differential over the time step, the lattice Boltzmann equation written in an explicit form is

$$f_i(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) = f_i(\vec{x}, t) - \omega(f_i(\vec{x}, t) - f_i^{eq}(\vec{x}, t)) \quad (2)$$

where i represents the velocity index. A nine-speed lattice Boltzmann model is shown in Fig. 1.

$$\begin{aligned}
\vec{c}_0 &= 0, 0 \\
\vec{c}_1 &= +c, 0 \\
\vec{c}_2 &= 0, +c \\
\vec{c}_3 &= -c, 0 \\
\vec{c}_4 &= 0, -c \\
\vec{c}_5 &= +c, +c \\
\vec{c}_6 &= -c, +c \\
\vec{c}_7 &= -c, -c \\
\vec{c}_8 &= +c, -c
\end{aligned} \tag{3}$$

$$c = \Delta x / \Delta t \tag{4}$$

where Δt is the numerical time step and Δx is the spatial discretization.

The incompressible Navier-Stokes equations can be derived from the lattice Boltzmann equation using a multiscale expansion [1]. The 2-D incompressible Navier-Stokes equations are presented in vector form:

$$\begin{aligned}
\nabla \cdot \vec{u} &= 0 \\
\partial_t \vec{u} + (\vec{u} \nabla) \vec{u} &= -\frac{1}{\rho} \nabla p + \nu \nabla^2 \vec{u}
\end{aligned} \tag{5}$$

The dependent variables in the Navier-Stokes equations are defined in terms of the distribution functions $f_i(\vec{x}, t)$, and the viscosity is related to the collision frequency. The density is defined as

$$\rho(\vec{x}, t) = \sum_i f_i(\vec{x}, t) \quad i = 0, \dots, 9 \tag{6}$$

The momentum density is defined as follows

$$\rho(\vec{x}, t) \vec{u}(\vec{x}, t) = \sum_i \vec{c}_i f_i(\vec{x}, t) \quad i = 0, \dots, 9 \tag{7}$$

The equilibrium distributions have been defined as follows [1].

$$\begin{aligned}
f_i^{eq} &= \frac{4}{9} \rho \left[1 - \frac{3}{2} \frac{\bar{u}^2}{c^2} \right] & i = 0 \\
f_i^{eq} &= \frac{1}{9} \rho \left[1 + 3 \frac{\bar{c}_i \cdot \bar{u}}{c^2} + \frac{9}{2} \frac{(\bar{c}_i \cdot \bar{u})^2}{c^4} - \frac{3}{2} \frac{\bar{u}^2}{c^2} \right] & i = 1, 2, 3, 4 \\
f_i^{eq} &= \frac{1}{36} \rho \left[1 + 3 \frac{\bar{c}_i \cdot \bar{u}}{c^2} + \frac{9}{2} \frac{(\bar{c}_i \cdot \bar{u})^2}{c^4} - \frac{3}{2} \frac{\bar{u}^2}{c^2} \right] & i = 5, 6, 7, 8
\end{aligned} \tag{8}$$

For more accurate computation of steady flows, the equilibrium distribution functions have been modified slightly [8]. The kinematic viscosity is defined as

$$\nu = \frac{1}{3} c^2 \Delta t \left(\frac{1}{\omega} - \frac{1}{2} \right). \tag{9}$$

To implement an explicit time marching lattice Boltzmann solver, the following substeps are required at each time step:

1. Using the initial $\bar{u}$, calculate the equilibrium distributions (Eq. 8) and set $f_i = f_i^{eq}$ for the first time step.
2. Compute the right hand side of the lattice Boltzmann equation (Eq. 2), called collision substep, and propagate the result to the nearest neighbor nodes to obtain $f_i(\bar{x} + \bar{c}_i \Delta t, t + \Delta t)$. The unknown distribution functions at the boundary are computed using non-equilibrium distribution functions or the bounceback conditions (specified in the next section).
3. Update the density and the momentum density from the new distributions according to the given definitions (Eqs. 6 and 7).

Start the next time step with the calculation of new equilibrium distributions using the new macroscopic variables (Eq. 8) and proceed with Step 2 of the algorithm. Follow this procedure until the final time is reached.

2.1 LBM for backward facing step flow

The backward facing step flow geometry is shown in Fig. 2. The figure shows solid walls at boundary B1, the step wall, and at B2 and B3, the base wall and the upper wall. The no-slip boundary conditions at the walls are imposed through the bounce-back boundary conditions [1]. Here the collision step is not performed at the walls, but the propagation step is still implemented. The implementation of the propagation step for some of the distribution functions on the walls requires information from outside the domain, and therefore they remain unknown. Following the propagation step, each unknown f_i is assigned the value of the f_i of the opposite direction. For example, at a point on the upper boundary, the unknown distribution functions are f_4 , f_7 , and f_8 . Therefore, on the upper wall

$$\begin{aligned}f_4(i, j) &= f_2(i, j) \\f_7(i, j) &= f_5(i, j) \\f_8(i, j) &= f_6(i, j)\end{aligned}\tag{10}$$

Assigning the inflow and outflow boundary conditions in the lattice Boltzmann algorithm requires a different methodology than traditional CFD methods, where the initial and boundary conditions are specified in terms of the macroscopic variables. To obtain the initial and boundary conditions in terms of the distribution functions, an inverse mapping between the distribution function and the macroscopic variable (a direct mapping being Eqs. 6 and 7) is required. The distribution functions at the inflow and outflow are defined as the equilibrium distributions or the sum of the equilibrium distributions and a non-equilibrium term such as

$$f_i = f_i^{eq} + f_i^{neq}\tag{11}$$

Skordos [9] obtained the non-equilibrium distributions for a seven-speed LBM using a multiscale expansion. For the nine-speed lattice Boltzmann model for the Navier-Stokes equations, these can be given in tensor form as

$$\begin{aligned}
f_i^{neq} &= -\frac{\Delta t}{\omega} \cdot \frac{1}{3c^2} \left(c_{i\alpha}^2 \frac{\partial u_\alpha}{\partial x_\alpha} \right) & i = 1,2,3,4 \\
f_i^{neq} &= -\frac{\Delta t}{\omega} \cdot \frac{1}{12} \left(\frac{\partial u_\alpha}{\partial x_\beta} \right) & i = 5,7 \\
f_i^{neq} &= \frac{\Delta t}{\omega} \cdot \frac{1}{12} \left(\frac{\partial u_\alpha}{\partial x_\beta} \right) & i = 6,8 \\
f_i^{neq} &= 0 & i = 0
\end{aligned} \tag{12}$$

Equation 11 can also be used for specifying boundary conditions at the walls. The density at the outlet (B5) is fixed, whereas the density at the inlet (B4) is extrapolated from density values within the flow domain. The velocity at the inlet is given by the parabolic profile specified by the problem, and it remains constant throughout the time marching procedure to steady state conditions. The velocity at the outlet is defined such that there is zero velocity gradient at the outlet.

Equation 1 shows that the discrete velocity Boltzmann equation consists of first-order hyperbolic PDEs. For a first-order hyperbolic PDE, such as the 1-D linear advection equation,

$$u_t + cu_x = 0 \tag{13}$$

the stability of an upwind and forward Euler numerical discretization (same as the numerical discretization in LBM) is defined by the Courant-Friedrichs-Lewy (CFL) condition [10]:

$$CFL = c \frac{\Delta t}{\Delta x} \leq 1 \tag{14}$$

The above condition holds for a 1-D linear advection equation. In LBM there are nine linear advection equations, with non-linear source terms that couple them together. In LBM it is generally assumed that the advection speed is equal to the spatial discretization over the time step and that this should satisfy the CFL condition for the advection part of Eq. 1. However, computational experiments found that the time step for this problem is limited to slightly more than one-fourth the spatial discretization when solving the backward facing step flow for the given parameters. This means that the time step is more limited than that given by Eq. 14. The reason for this is the non-linear nature of the source terms given in Eq. 8 and LBM's requirement of a low Mach number ($|\vec{u}|/c$) for the simulation. A high Mach number results in compressibility errors because the multiscale expansion that relates the LBM to the Navier-Stokes equations assumes a small Mach number [11].

2.2 The cache-optimized lattice Boltzmann method

The LBM for the 2-D incompressible Navier-Stokes equations uses nine distribution functions, each requiring a 2-D array. Consider an 8 MB cache (note that approximately one-half the cache memory is used for other operations) on a single processor; the biggest square grid size that can be accommodated within the cache for repeated use is approximately 250×250 because

$$9 \text{ arrays} \times \frac{250 \times 250 \text{ elements}}{\text{array}} \times \frac{8 \text{ bytes}}{\text{element}} = 4.4 \text{ MB}. \quad (15)$$

On a single processor, grid sizes that are larger than 250×250 can be accommodated in the cache by dividing the grid into subsections that can fit in the cache. Cache optimization is achieved when the subsection data is called from the cache repeatedly for

performing computations [12]. This is possible because LBM computations depend on local data, such as the computation of the right hand side of Eq. 2 (collision substep), including the computation of the equilibrium distributions. They do not involve any dependencies between the subsections:

$$f_i(\bar{x}, t) = (1 - \omega) \cdot f_i(\bar{x}, t) + \omega \cdot f_i^{eq}(\bar{x}, t) \quad (16)$$

The subsections mentioned above are horizontal strips of dimension $N_x \times m_1$, where N_x is the number of grid points in the horizontal direction, and $m_1 = N_y / p_1$, where p_1 is the number of strips and is chosen such that the subsection size will not exceed cache memory size (Fig. 3).

The propagation substep is an assignment operation of an almost local nature due to nearest-neighbor dependencies:

$$f_i(\bar{x} + \bar{c}_i \Delta t, t + \Delta t) = f_i(\bar{x}, t) \quad (17)$$

If LBM computations solely consist of Eq. 16, then each subsection can be updated separately until the final time or convergence to steady state. However, LBM computations also include the propagation substep that causes the number of updates ($tdiv$) on a subsection to be limited [12]. The local nature of LBM floating point operations has been utilized for LBM implementation on GPUs [13].

3. THE VORTICITY-STREAM FUNCTION FORMULATION

The VSM [14] was chosen as the traditional solution technique because it can be efficiently implemented through the ADI scheme and cyclic reduction [5,6]. It consists of a vorticity transport equation and a Poisson equation for the stream function, which enables the stream function to automatically satisfy the conservation of mass constraints:

$$\frac{\partial \zeta}{\partial t} + u \frac{\partial \zeta}{\partial x} + v \frac{\partial \zeta}{\partial y} = \nu \left(\frac{\partial^2 \zeta}{\partial x^2} + \frac{\partial^2 \zeta}{\partial y^2} \right) \quad (18)$$

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\zeta \quad (19)$$

The vorticity is defined as

$$\zeta = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \quad (20)$$

Where the velocity components are defined in terms of the stream function,

$$\begin{aligned} u &= \frac{\partial \psi}{\partial y} \\ v &= -\frac{\partial \psi}{\partial x} \end{aligned} \quad (21)$$

The solution is computed using a time marching procedure:

1. Assign initial and boundary values to ψ and ζ at all grid points.
2. Use a finite difference analog of the vorticity transport equation to calculate the vorticity at the next time level, $t + \Delta t$.
3. Solve the Poisson equation using a cyclic reduction technique with the new interior values of the vorticity in the source term of the equation.
4. Calculate the new velocity components from the stream function.
5. Update the boundary values of the vorticity from the new ψ at the neighboring interior points.

This computational procedure is repeated from Step 2 onwards until steady state conditions are achieved. The diffusion terms in Eq. 18 are discretized using central differences, and a first order upwind scheme is adopted for the convection terms. To

obtain a tridiagonal representation of the matrix resulting from applying the discretization at all grid points, a time splitting technique (the ADI scheme) is implemented [5,10].

For the VSM there are three 2-D arrays in this method—the vorticity arrays at the two time levels and the stream function array. Based on this, the largest square grid size that can be accommodated within an 8 MB size cache (where approximately 4 MB is available for repeated use) is approximately 450×450 .

$$3 \text{ arrays} \times \frac{450 \times 450 \text{ elements}}{\text{array}} \times \frac{8 \text{ bytes}}{\text{element}} = 4.7 \text{ MB}$$

For grids larger than this the VSM is not suited for cache optimization due to the global nature of the Gaussian elimination computations for the vorticity transport equation and the cyclic reduction technique for the Poisson equation. During each time step the tridiagonal matrices are inverted using elimination along the length of the whole domain. Due to this, sections of the domain cannot be updated separately, and as a result the data cannot be called from the cache repeatedly.

4. NUMERICAL PERFORMANCE OF THE LBM AND VSM

In this section the performance of cache-optimized LBM (cLBM) and the VSM are compared for the 2-D backward facing step flow. The geometry and flow parameters are described in Fig. 2. The maximum velocity, U_{max} , is 1; the step height is h ; and the downstream channel height is H . The expansion parameter, r , is

$$r = H/(H - h)$$

The expansion ratio for both cases examined in this paper is 1.96. This is close to the expansion ratio of 1.94 used in the experiments of backward facing step flow conducted by Armaly et al [15]. These experiments are used to validate the computational results in

this study. The recirculation length of the primary vortex or the reattachment length is X_r . The downstream boundary is located at $x = 30h$. Computational experiments were performed for $Re = 150$ and $Re = 500$, where the Reynolds number is defined as

$$Re = \frac{4U_{\max} h}{3\nu} \quad (22)$$

where ν is the dynamic viscosity. The results were computed on a 400 MHz processor with 512 MB main memory and 8 MB cache memory. Fortran 90 was the programming language. Each of the methods was run until steady state conditions were achieved, i.e., an unchanging reattachment length and the resolution of the secondary vortex at the bottom of the step. Final time as reported here is the total number of iterations multiplied by the numerical time step and therefore is dimensionless.

Various grid resolutions and their corresponding time step sizes were tested for the CPU time and accuracy of the cLBM, the VSM, and the coupled cLBM-VSM schemes. The time step size decreases with increasing grid resolution. Higher grid resolution resolves certain flow structures that a coarser grid resolution does not resolve. For example, the $\Delta x = 0.05$ grid spacing is unable to resolve the secondary vortex or recirculating eddy at the bottom of the step (Fig. 4) because the region containing this structure is from 0 to 0.1 in the x direction and 0 to 0.1 in the y direction for a Reynolds number of 150. A higher grid resolution of $\Delta x = 0.02$ is able to resolve the secondary vortex.

To compare the accuracy and compute time of the cLBM and the VSM, the reattachment length is used as the reference. Table 1 reports the results for Case 1 ($Re = 150$). As shown, for the largest grid spacing case, $\Delta x = 0.05$, the reattachment length for the VSM is lower than the experimental value of 4.1 [15]. This is due to the first order

upwind discretization of the spatial derivatives, which leads to first order accurate results. As noted earlier, the $\Delta x = 0.05$ grid spacing is unable to resolve the secondary vortex or recirculating eddy at the bottom of the step. The VSM computations were performed until a final time of 76.0. The next two higher grid resolutions for the VSM give a reattachment length closer to the experimental value of 4.1 and capture the structure of the secondary vortex at the bottom of the step. The flow structure remained unchanged with further iterations. Figure 5 shows the results for the VSM method with the $\Delta x = 0.02$ grid spacing. The largest grid spacing ($\Delta x = 0.05$) in Table 1 gives a cLBM reattachment length of 4.2, which is close to the experimental result of 4.1 [15]. The reattachment length given by the cLBM increases slightly with increasing grid resolution. Figure 6 shows the results for the cLBM method with the $\Delta x = 0.02$ grid spacing. The reattachment lengths (Table 1) for the cLBM method are slightly beyond the experimental value of 4.1 [15] for two reasons. The backward step expansion ratio examined in this paper is 1.96, which is higher than the expansion ratio of 1.94 used in the experiments conducted by Armaly et al. [15]. Reference [16] presented examples showing that an increase in the expansion ratio leads to an increase in the reattachment length. Noting that the increase in the reattachment length is nonlinear with respect to increasing expansion ratios, we can estimate that the expected reattachment length would be approximately 4.2. In addition, as noted earlier the assumed boundary condition is fully developed laminar flow at the top of the backward step (boundary B4 in Fig. 2). Comparing this assumption to the velocity profiles for various Reynolds number flows reported in Reference [15], it can be seen that this is a reasonable assumption. However, as shown in Reference [15], there is some upstream feedback from the separation region,

and the assumption of a parabolic velocity distribution slightly increases the reattachment length relative to the experimental values. In the discussion here we are focused on comparing the relative speed and accuracy of the coupled multigrid cLBM-VSM with the multigrid VSM method. The primary concern in this study is the resolution of the secondary vortex at the bottom of the backward step. The cLBM resolved the primary recirculating region and gave the expected reattachment length at a final time of 76.0, but failed to resolve the secondary vortex at the bottom of the step until a final time of 156.0. The cLBM required more iterations than the VSM approach despite both approaches possessing similar time step size because cLBM's explicit computational nature precludes obtaining information from the whole computational domain as is done by implicit methods.

The results of test case 2 ($Re = 500$) are shown in Table 2. The experimental reattachment length is 10.2 [15]. The cLBM computed reattachment length for the grid spacing, $\Delta x = 0.01$, is slightly lower than the experiment. This discrepancy occurs due to the emerging three-dimensionality in the flow beyond $Re = 400$ as demonstrated by Biswas et al [16]. As shown, the reattachment lengths from the cLBM computations for various grid resolutions are much closer to the experimental values than those computed by the VSM. The results obtained by the VSM deteriorate for $Re = 500$ as compared to the results for $Re = 150$ for the same grid spacing due to the first order upwind spatial discretization for the convection term. The VSM attained steady state conditions at a final time of 213.0. The cLBM did not reach steady state until a final time of 264.0 when it was able to resolve the secondary vortex at the bottom of the step. The secondary vortex at the bottom of the step covered the region from 0 to 0.14 in the x direction and 0 to 0.14

in the y direction. For $Re = 500$, a secondary vortex also appears on the upper wall. The horizontal starting location for this vortex is near the horizontal end location of the primary recirculating region (at the base wall). Figures 7 and 8 show the velocity contours for the VSM and cLBM methods for the $\Delta x = 0.01$ grid spacing. Although Reference [15] does not provide velocity profiles at the Reynolds numbers investigated in this paper, it does provide velocity profiles at $Re = 100, 389$, and 1000 . The velocity contours shown here are consistent with those provided in Reference [15].

To compare the computational performance of both methods, the CPU time was measured until steady state conditions are reached based on the criteria discussed above. As noted earlier, with similar time step sizes for both methods, the cLBM required a greater number of time steps to reach steady state than the VSM. For $Re = 150$, the final times were 156.0 and 76.0 for the cLBM and the VSM, respectively. For $Re = 500$, the final times were 264.0 and 213.0 for cLBM and VSM, respectively. Tables 1 and 2 show the CPU time for the cLBM and the VSM for grid sizes 601×41 to 3001×201 corresponding to grid spacing from $\Delta x = 0.05$ to $\Delta x = 0.01$, respectively. The CPU time for the VSM is less than the cLBM CPU time by a factor 3.2 for $\Delta x = 0.05$ at $Re = 150$ and a factor 2.89 at $Re = 500$. The VSM reaches steady state faster due to the much bigger time step for $\Delta x = 0.05$ permitted by the implicit formulation. Increasing the grid resolution and the locally high wall-normal velocity at the step region places a restriction on the time step size for the VSM [17] because the VSM time step size for smaller grid spacing decreases. As discussed earlier, the cLBM time step size decreases as the grid spacing decreases and the Reynolds number increases. As shown in Tables 1 and 2, the cLBM time step size is 0.008 for a grid spacing 0.02 and $Re = 150$, and decreases to

0.006 for the same grid spacing at $Re = 500$. With grid spacing 0.02 (or grid size 1501×101), the VSM has a larger time step size $\Delta t = 0.01$ than the cLBM. However, the cLBM CPU time is less than that of the VSM by a factor 1.28 for $Re = 150$ and by a factor 1.6 for $Re = 500$. This is due to the computational efficiency of the cLBM afforded by cache optimization [12]. For grid spacing 0.01 (grid size is 3001×201), both methods have the same numerical time step size. Here, the cLBM is about 1.89 times faster than the VSM for $Re = 150$ and 3.16 times faster for $Re = 500$. cLBM's CPU time reduction factor is smaller for $Re = 150$ (compared to $Re = 500$) due to the much larger final time required for cLBM to converge to steady state.

The VSM reaches steady state with a lesser number of time iterations (and therefore a lower simulation final time) than cLBM due to the implicit nature of the VSM's numerical scheme. However, as the Reynolds number increases, the difference in the final times for both methods is found to decrease. Due to this, the cLBM takes much less CPU time than the VSM for higher Reynolds number cases and higher grid resolutions (where the time step size for both methods is similar).

5. COUPLING LBM AND THE VSM

As noted in the previous section, the VSM is more computationally efficient than the cLBM when solving on coarse grids where the VSM possessed a much larger time step size than the cLBM. However, the cLBM outperforms (i.e., provides results of similar accuracy using less computational time) the VSM (and other traditional methods) when both methods have the same size time step. Many fluid flow problems include regions where the resolution of critical flow structures requires relatively small grid spacing and

other regions that can be sufficiently resolved with relatively larger grid spacing. For example, in the 2-D backward facing step flow problem considered in this paper, the step region includes a secondary recirculating eddy or vortex at the bottom of the step (Fig. 4). As noted earlier, to resolve this vortex, a grid spacing of $\Delta x = 0.02$ or smaller is required. In contrast the remainder of the flow region can be adequately described with a larger grid spacing. Based on this, the 2-D backward facing step flow spatial domain can be divided into two regions: the flow region consisting of the secondary vortex that is discretized with a high resolution grid (fine grid) and the remaining part of the flow region that is discretized with a coarser grid. The cLBM can be used as the solver for those flow regions where the time step of the cLBM and the VSM are the same. In regions where the cLBM's time step is much smaller than that of the VSM, the VSM is used as the solver. Based on this, a coupled cLBM-VSM can combine the computational efficiency of cLBM with the faster convergence properties of the VSM for solving the backward facing step flow problem. Asproulis et al [18] adopted a similar rationale for using a hybrid molecular dynamics–traditional CFD method for computing flow in microfluidic devices, where the molecular dynamics code supplemented traditional continuum methods for accurate modeling of slip effects near the wall region.

In the test case developed here, a high-resolution grid block with grid spacing $\Delta x = 0.01$ is used in the region near the step, and a coarse grid block with grid spacing $\Delta x = 0.05$ is used for the rest of the domain (Fig. 9). This results in a grid refinement factor of 5. As discussed earlier, the cLBM and the VSM have the same time step for a grid spacing of $\Delta x = 0.01$. Therefore, the cLBM will be applied on the fine grid block, and the VSM will be applied on the coarse grid block because it can possess as big a time

step as $\Delta t = 0.1$ for grid spacing $\Delta x = 0.05$. To enable transfer of information between the two methods, the two grid blocks overlap at the interface.

5.1 Coupling procedure

The coupling procedure should enable the matching of the solution between adjacent grid blocks. Both the cLBM and the VSM are completely separate non-primitive formulations. However, the velocity vector can be computed from both formulations. Hence, the velocity variable will be used to transfer information between both grid blocks. This leads to the following method for communicating information between the grid blocks:

- 1) The variables on the fine grid block are updated using the cLBM. The distribution functions on the interface boundary of the fine grid are assigned the equilibrium distribution function values. To compute the equilibrium distributions (Eq. 8) at the interface, the velocity vector at the interface is required. This velocity vector is computed from the stream function values in the neighboring coarse grid block using finite difference representations of Eq. 21.
- 2) The VSM solver computes the vorticity and the stream function variables in the coarse grid region. The interface vorticity and stream function boundary conditions are specified using the velocities computed in Step 1 at the corresponding fine grid locations. Equation 7 is used to compute the Γ_2 interface velocities (u, v) from the distribution functions. After the velocities are obtained, the stream function at the interface is obtained through numerical integration of the horizontal velocity component (u) along the vertical direction. The vorticity is obtained from the fine

grid velocity components using the definition given by the equation and applying finite differences.

As seen above, the coupling procedure essentially consists of solving two Dirichlet problems on overlapping subdomains. Steps 1 and 2 are executed alternately until the difference between the current solution and the solution from the previous iteration in the fine grid block is less than a given tolerance. This procedure occurs at every time step. To make full use of the cache-efficient nature of the cLBM, Step 1 is executed separately for a certain number of time steps ($tdiv$) while keeping the same interface boundary condition. A high level description of this strategy is given below.

Let Ω_1 and Ω_2 represent the two overlapping grid blocks (or subdomains), and $\partial\Omega_1$ and $\partial\Omega_2$ represent their boundaries. The part of $\partial\Omega_1$ lying in Ω_2 is represented with Γ_1 (artificial boundary or internal boundary or interface boundary of Ω_1 .) Likewise Γ_2 represents the interface of Ω_2 .

1. Initialize the distribution functions on Ω_1 and the vorticity and stream function on Ω_2 .
2. Loop for all time steps (until the final time or until steady state conditions are reached in the whole domain):
 - a. Compute the distribution functions on Γ_1 using $\psi|_{\Omega_1}$ at Γ_1 .
 - b. Loop for $tdiv$ number of time steps (for LBM cache optimization).
 - c. Update distribution functions on Ω_1 using LBM (boundary conditions remain unchanged).
 - d. End loop for $tdiv$ time steps.

- e. Compute $\psi|_{\Gamma_2}$ and vorticity $|_{\Gamma_2}$ from the distribution functions $|_{\Omega_2}$ at Γ_2 .
 - f. Perform the vorticity-stream function computations to update ψ and the vorticity on Ω_2 (boundary conditions remain unchanged).
3. End loop for all time steps when a steady state solution is reached.

5.2 Method for comparison

To quantify the computational performance and accuracy of the coupled cLBM-VSM, it should be compared with traditional finite difference methods that are applied over the whole domain. The VSM is applied over the whole domain using multiblock gridding. Here the VSM is implemented separately on all grid blocks in the domain. Unlike the coupled cLBM-VSM, there is no requirement to introduce a primitive variable, such as the velocity to transfer information between neighboring grid blocks. The coupling procedure for the multiblock VSM is shown below.

1. Initialize the vorticity and stream function on both Ω_1 and Ω_2 .
2. Loop for all time steps (until the final time or until steady state conditions are reached in the whole domain)
 - a. Perform computations using the vorticity-stream function solver on Ω_1 and update both vorticity and stream function for one time step.
 - b. Compute the interface boundary values for Ω_2 from the variables belonging to Ω_2 , i.e., $\psi|_{\Gamma_2}$ and vorticity $|_{\Gamma_2}$ are computed from $\psi|_{\Omega_1}$ and vorticity $|_{\Omega_1}$ near Γ_2 .

- c. Perform computations using a vorticity-stream function solver on Ω_2 and update both vorticity and stream function for one time step.
- d. Compute the interface boundary values for Ω_1 from the variables belonging to Ω_2 , i.e., $\psi|_{\Gamma_1}$ and vorticity $|_{\Gamma_1}$ are computed from $\psi|_{\Omega_2}$ and vorticity $|_{\Omega_2}$ near Γ_1 .

3. End loop for all time steps when the steady state solution is reached.

It has been observed from computational experiments that both multiblock methods, i.e., the coupled cLBM-VSM solver and the multiblock VSM solver, take the same number of iterations to converge to steady state conditions. However, when the solver on the fine grid is repeated for a certain number of time steps before communicating with the neighboring solver, fewer overall iterations are required for convergence. To utilize the cache-optimization properties of the cLBM, it is repeated for a certain number of time steps (*tdiv*) as shown in the high level description in section 6.1. The same procedure is not performed for the multiblock VSM solver where the VSM solver operates on the fine grid block because the CPU time becomes greater than that for the high-level strategy given above.

6. RESULTS

The results were computed for $Re = 150$ and $Re = 500$. For case 1 ($Re = 150$), the fine grid block covers the region from $x = 0.0$ to $x = 7.5$. The overlap region is from $x = 6.5$ to $x = 7.5$. The coarse grid block covers the region from $x = 6.5$ to $x = 30.0$. The grid spacing for the fine grid block is 0.01. Therefore, the size of this grid block will be 751×201 . The grid refinement factor is 5. This results in a coarse grid block of size 471×41 ,

corresponding to grid spacing 0.05. Local time stepping was performed by assigning a time step size 0.003 to the fine grid block and a time step size 0.1 to the coarse grid block. Table 3 shows the reattachment length, CPU time, and number of time iterations for both multiblock methods at $Re = 150$. Both methods were executed until they reached steady state.

For case 2, computations were performed for $Re = 500$. Here the fine grid block extends to $x=10.0$ and the overlap distance is taken as 2.0 because the reattachment length is around 10.2 [15]. Table 4 shows the reattachment length, CPU time, and total number of time iterations for the two methods at $Re = 500$.

For $Re = 150$, the coupled cLBM-VSM results in a reattachment length of 4.3, which is similar to the result obtained for the standalone cLBM. The multiblock VSM results in a reattachment length of 4, which is the same as that obtained by the standalone VSM. A coupled LBM-VSM solver was used without any cache optimization to check for the number of iterations. In this case, the coupled LBM-VSM solver was able to resolve the secondary vortex at the bottom of the step for $Re = 150$ after executing the same number of time iterations (19,000) as the multiblock VSM. However, the cache optimized version of the coupled solver required only 2100 iterations. The number of iterations was reduced because the cLBM performs 12 time steps within every computational cycle as mentioned in section 3.1. The CPU time for the coupled cLBM-VSM is less than the CPU time for the multiblock VSM by a factor 2.93.

For case 2 ($Re = 500$), the accuracy of both multiblock methods (Tables 2 and 4) is slightly better than the standalone versions. This is due to the faster transfer of

information across the domain because part of the domain is covered with a coarse grid. The coupled cLBM-VSM is 2.85 times faster than the multiblock VSM in this case.

In Part I of this study, the traditional implicit scheme was implemented on the finer grid covering the steep gradient region because the time step of the cLBM on the finer grid was too small. In Part II of this study, it was seen that the time step size of the traditional implicit scheme (VSM) was the same as that of cLBM on the finer grid region covering the step region in the geometry. Therefore, cLBM was used on the finer grid in this study. Both studies demonstrate the utility of implementing separate solvers on separate regions of the flow domain so that each solver can compute most efficiently in its part of the flow domain.

7. CONCLUSIONS

From the results for both multiblock methods, the coupled LBM-VSM has advantages in terms of both CPU time and accuracy over the VSM by itself. It is expected that the coupled method will further reduce the CPU time for problems with increasing fine grid block regions. It is also expected that parallelization of the algorithms will result in a more efficient coupled LBM-VSM solver relative to the multiblock VSM solver [19]. Planned future work includes extending the coupled method to flow problems represented by the 3-D Navier-Stokes equations such as the 3-D backward step flow.

ACKNOWLEDGMENTS

This research was supported in part by the US Department of Energy – Fossil Energy Program under Contract No. DE-AC02-07CH11358 through the Ames Laboratory.

References

1. Wolf-Gladrow DA. Lattice-gas cellular automata and lattice Boltzmann models: an introduction. Berlin: Springer; 2000.
2. Velivelli AC, Bryden KM. Parallel performance and accuracy of lattice Boltzmann and traditional finite difference methods for solving the unsteady two-dimensional Burger's equation. *Physica A* 2006;362(1):139–45.
3. Pelliccioni O, Cerrolaza M, Surós R. A biofluid dynamic computer code using the general lattice Boltzmann equation. *Adv Eng Softw* 2008;39(7):593–611.
4. Stensholt S, Øien A. Lattice Boltzmann simulations of the motion induced by variable surface tension. *Adv Eng Softw* 2011;42(11):944–53.
5. Wang Y, Baboulin M, Dongarra J, Falcou J, Fraigneau Y, Maître OL. A parallel solver for incompressible fluid flows. *Procedia Comput Sci* 2013;18:439–48.
6. Bini DA, Meini B. The cyclic reduction algorithm: from Poisson equation to stochastic processes and beyond. *Numer Algorithms* 2009;51(1):23–60.
7. Morgan K, Periaux J, Thomasset F, editors. Analysis of Laminar Flow over a Backward Facing Step: a GAMM-Workshop. Braunschweig-Wiesbaden: Friedrich Vieweg & Sohn; 1984.
8. Zou Q, Hou S, Chen S, Doolen GD. A improved incompressible lattice Boltzmann model for time-independent flows. *J Stat Phys* 1995;81:35–48.
9. Skordos PA. Initial and boundary conditions for the lattice Boltzmann method. *Phys Rev E* 1993;48:4823–42.
10. Pletcher RH, Tannehill JC, Anderson DA. 3rd ed. Computational fluid mechanics and heat transfer. Boca Raton (FL): Taylor and Francis; 2011.
11. Sterling JD, Chen S. Stability analysis of lattice Boltzmann methods. *J Comput Phys* 1996;123:196–206.
12. Velivelli AC, Bryden KM. A cache-efficient implementation of the lattice Boltzmann method for the two-dimensional diffusion equation. *Concurr Comp-Pract E* 2004;16:1415–32.
13. Habich J, Zeiser T, Hager G, Wellein G. Performance analysis and optimization strategies for a D3Q19 lattice Boltzmann kernel on nVIDIA GPUs using CUDA. *Adv Eng Softw* 2011;42:266–72.

14. Roache PJ. Computational Fluid Dynamics. Socorro (NM): Hermosa Publishers; 1976.
15. Armaly BF, Durst F, Pereira JCF, Schönung B. Experimental and theoretical investigation of backward-facing step flow. *J Fluid Mech* 1983;127:473–96.
16. Biswas G, Breuer M, Durst F. Backward-facing step flows for various expansion ratios at low and moderate Reynolds numbers. *J Fluids Eng-T ASME* 2004;126(3):362–74.
17. Colella P, Pao K. A projection method for low speed flows. *J Comput Phys* 1999;149:245–69.
18. Asproulis N, Kalweit M, Drikakis D. A hybrid molecular continuum method using point wise coupling. *Adv Eng Softw* 2012;46:85–92.
19. Velivelli AC, Bryden KM. Domain decomposition based coupling between the lattice Boltzmann method and traditional CFD methods - part I: formulation and application to the 2-D Burgers' equation. *Adv Eng Softw* 2014;70:104–12.

Fig. 1. A schematic of a nine-speed lattice Boltzmann model.

Fig. 2. Schematic of backward step flow geometry and parameters.

Fig. 3. Domain decomposition for the cache-blocked serial implementation of the lattice Boltzmann method.

Fig. 4. Diagram showing the flow and the secondary vortex at the bottom of the step for $Re = 500$.

Fig. 5. VSM velocity contour for $Re = 150$

Fig. 6. cLBM velocity contour for $Re = 150$

Fig. 7. VSM velocity contour at $Re = 500$

Fig. 8. cLBM velocity contour at $Re = 500$

Fig. 9. Schematic of the multiblock grid used for the coupled solver.

Table 1. The computed reattachment length, number of time steps to steady state, and CPU time for the cLBM and the VSM for $Re = 150$. The experimental reattachment length is 4.1 [15].

grid	Δt	X_r	number of time steps	CPU time (s)					
spacing	size	cLBM	VSM	cLBM	VSM	cLBM	VSM	cLBM	VSM
0.05	601 x 41	0.0125	0.1	4.2	3.6	12,480	1560	48	15
0.02	1501 x 101	0.008	0.01	4.25	4.05	19,500	7600	938	1204
0.01	3001 x 201	0.004	0.004	4.30	4.0	39,000	19,000	8397	15,901

Table 2. The computed reattachment length, time steps to steady state, and CPU time for the cLBM and the VSM for $Re = 500$. The experimental reattachment length is 10.2 [15].

grid	Δt	X_r	number of time steps	CPU time (s)					
spacing	size	cLBM	VSM	cLBM	VSM	cLBM	VSM	cLBM	VSM
0.05	601 x 41	0.0125	0.09	9.3	7.0	21,120	2933	133	46
0.02	1501 x 101	0.006	0.01	9.45	8.15	44,000	21,300	2114	3392
0.01	3001 x 201	0.003	0.003	9.5	7.8	88,000	71,000	18,816	59,469

Table 3. Reattachment length (X_r), CPU time, and number of time steps for the multiblock scheme with coupled solvers and the multiblock VSM for $Re = 150$.

		coupled method	VSM
fine grid block	grid spacing	0.01	0.01
	time step	0.004	0.004
coarse grid block	grid spacing	0.05	0.1
	time step	0.1	0.01
X_r		4.3	4.0
number of time steps		2100	19,000
CPU time (s)		1465	4295

Table 4. Reattachment length (X_r), CPU time, and number of time steps for the multiblock scheme with different solvers and the multiblock VSM for $Re = 500$.

		coupled method	VSM
fine grid block	grid spacing	0.01	0.01
	time step	0.003	0.004
coarse grid block	grid spacing	0.05	0.1
	time step	0.1	0.01
X_r		9.6	8.35
number of time steps		7300	71,000
CPU time (s)		6155	17,560

1. The cache-optimized lattice Boltzmann and finite difference methods are coupled
2. The domain decomposition based coupling is implemented on multiblock grids
3. Each individual solver is deployed in the region where it is most efficient
4. The coupled solver is applied to flow over a backwards step
5. The coupled solver was 2.9x faster than finite difference methods