PHYSICAL REVIEW E, VOLUME 64, 045701(R)

Least-squares-based lattice Boltzmann method: A meshless approach for simulation of flows with complex geometry

C. Shu, Y. T. Chew, and X. D. Niu

Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260

(Received 24 May 2001; published 25 September 2001)

A version of lattice Boltzmann method (LBM) is presented in this work, which is derived from the standard LBM by using Taylor series expansion and optimized by the least squares method. The method is basically meshless, and can be applied to any complex geometry and nonuniform grids. It can also be applied to different lattice models. The proposed method explicitly updates the distribution functions at mesh points by an algebraic formulation, in which the relevant coefficients are precomputed from the coordinates of mesh points. We have successfully applied this method to simulate many two-dimensional incompressible viscous flows. The numerical results are very accurate, and the computational time needed is much less as compared with other existing methods. In this paper, we mainly show the method.

DOI: 10.1103/PhysRevE.64.045701 PACS number(s): 45.50.−j, 47.11.+j

In recent years, the lattice Boltzmann method (LBM) has become an efficient and alternative tool for simulation of complex flows [1–6]. Due to uniformity of the lattice, the standard LBM is usually applied to the simple geometry with uniform grid. As we know, practical problems may involve complex geometry with curved boundaries. For such cases, the standard LBM cannot be applied directly.

Currently, there are two ways to improve the standard LBM so that it can be applied to complex problems [2–6]. One is the interpolation-supplemented LBM (ISLBM) proposed by He and his colleagues [2–3]. In this method, interpolation is applied at every time step in order to obtain the distribution function at the grid point. So, the computational effort by this method is very large as compared to the standard LBM. The other scheme is based on the solution of a differential lattice Boltzmann equation (LBE). For complex problems, the differential LBE can be solved in the computational space with the help of coordinate transformation [4]. The differential LBE can also be solved by the finite volume algorithm [5–6]. As showed by Chen [5], the finite volume-based LBM can exactly obey the conservation laws. It should be indicated that efficient numerical approaches such as upwind schemes are needed to solve the differential LBE in order to get the stable solution. As a consequence, the computational efficiency greatly depends on the selected numerical scheme.

In this work, we will present a new version of LBM, which seems to be more efficient than the existing methods. Let us start with the standard LBM. The two-dimensional standard LBE with BGK approximation can be written as

$$f_\alpha(x+e_{\alpha x}\delta t,y+e_{\alpha y}\delta t,t+\delta t)=f_\alpha(x,y,z)+[f_\alpha^{\mathrm{eq}}(x,y,t)-f_\alpha(x,y,t)]/\tau, \tag{1}$$

where τ is the single relaxation time f_α is the distribution function along the α direction, f_α^{eq} is its corresponding equilibrium state, δ_t is the time step, and e_α is the particle velocity in the α direction. The discrete velocities $\mathbf{e}_\alpha(e_{\alpha x},e_{\alpha y})$ and the equilibrium distribution f_α^{eq} can be found in Ref. [1]. For a uniform lattice, $\delta x=e_{\alpha x}\cdot\delta_t$, $\delta y=e_{\alpha y}\cdot\delta_t$. So, $(x+e_{\alpha x}\cdot\delta_t,\ y+e_{\alpha y}\cdot\delta_t)$ is on the grid point. In other words, Eq. (1) can be used to update the distribution functions exactly at the grid points. However, for a nonuniform grid, $(x+e_{\alpha x}\cdot\delta_t,\ y+e_{\alpha y}\cdot\delta_t)$ is usually not at the grid point $(x+\delta x,\ y+\delta y)$. To get the distribution function at the grid point $(x+\delta x,\ y+\delta y)$ and at the time level $t+\delta_t$, we need to apply the Taylor series expansion or other interpolation techniques such as the one used by He *et al.* [2,3]. In this work, the Taylor series expansion is used. Note that the time level for the position $(x+e_{\alpha x}\delta_t,\ y+e_{\alpha y}\delta_t)$ and the grid point $(x+\delta x,\ y+\delta y)$ is the same, that is, $t+\delta_t$. So, the expansion in the time direction is not necessary.

We will use Fig. 1 to illustrate our method. For simplicity, we let point A represent the position (x_A,y_A), point A' represent the position $(x_A+e_{\alpha x}\delta_t,\ y_A+e_{\alpha y}\delta_t)$, and point P represent the position (x_P,y_P). Using Eq. (1), we can get the distribution function at the position A' as

$$f_\alpha(A',t+\delta t)=f_\alpha(A,t)+[f_\alpha^{\mathrm{eq}}(A,t)-f_\alpha(A,t)]/\tau. \tag{2}$$

For the general case, A' may not coincide with the mesh point P. In the numerical simulation, we are only interested in the distribution function at the mesh point for all the time levels. So, the macroscopic properties such as the density, flow velocity can be evaluated at every mesh point. In this case, we need to obtain the distribution function at the mesh point P. This can be done by applying the Taylor series expansion in the spatial direction only. With Taylor series expansion, $f_\alpha(A',t+\delta t)$ can be approximated by the corresponding function and its derivatives at the mesh point P as

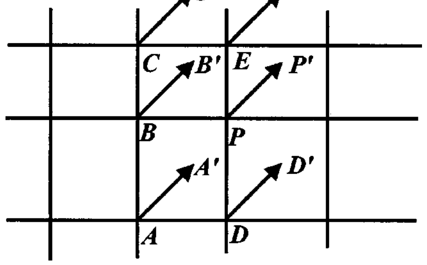

FIG. 1. Configuration of particle movement along the α direction.

$$f_\alpha(A',t+\delta t)=f_\alpha(P,t+\delta t)+\Delta x_A\frac{\partial f_\alpha(P,t+\delta t)}{\partial x}+\Delta y_A\frac{\partial f_\alpha(P,t+\delta t)}{\partial y}+\frac{1}{2}(\Delta x_A)^2\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x^2}+\frac{1}{2}(\Delta y_A)^2\frac{\partial f_\alpha(P,t+\delta t)}{\partial y^2}+\Delta x_A\Delta y_A\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x\partial y}+O[(\Delta x_A)^3,(\Delta y_A)^3], \tag{3}$$

where $\Delta x_A=x_A+e_{\alpha x}\delta t-x_P$, $\Delta y_A=y_A+e_{\alpha y}\delta t-y_P$. Note that the above approximation has a truncation error of the third order. Substituting Eq. (3) into Eq. (2) gives

$$f_\alpha(P,t+\delta t)+\Delta x_A\frac{\partial f_\alpha(P,t+\delta t)}{\partial x}+\Delta y_A\frac{\partial f_\alpha(P,t+\delta t)}{\partial y}+\frac{1}{2}(\Delta x_A)^2\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x^2}+\frac{1}{2}(\Delta y_A)^2\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial y^2}+\Delta x_A\Delta y_A\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x\partial y}=f_\alpha(A,t)+[f_\alpha^{\mathrm{eq}}(A,t)-f_\alpha(A,t)]/\tau. \tag{4}$$

It is indicated that Eq. (4) is a differential equation. Solving this equation can provide the distribution functions at all the mesh points. In this work, we go further to develop a solution procedure. In fact, our development is inspired from the Runge-Kutta method. As we know, the Runge-Kutta method is developed to improve the Taylor series method in the solution of ordinary differential equations (ODEs). As in Eq. (4), Taylor series method involves evaluation of different orders of derivatives to update the functional value at the next time level. For a complicated expression of given ODEs this application is very difficult. To improve the Taylor series method, the Runge-Kutta method evaluates the functional values at some intermediate points and then combines them (through the Taylor series expansion) to form a scheme with the same order of accuracy. With this idea in mind, we look at Eq. (4). We know that at the time level $t+\delta t$, the distribution function and its derivatives at the mesh point P are all unknowns. So, Eq. (4) has six unknowns in total. To solve for the six unknowns, we need six equations. However, Eq. (4) just provides one equation. We need additional five equations to close the system. As shown in Fig. 1, we can see that along the α direction, the particles at five mesh points P,B,C,D,E at the time level t will move to the new positions P',B',C',D',E' at the time level $t+\delta t$. The distribution functions at these new positions can be computed through Eq. (1), which are given below:

$$f_\alpha(P',t+\delta t)=f_\alpha(P,t)+[f_\alpha^{\mathrm{eq}}(P,t)-f_\alpha(P,t)]/\tau, \tag{5}$$

$$f_\alpha(B',t+\delta t)=f_\alpha(B,t)+[f_\alpha^{\mathrm{eq}}(B,t)-f_\alpha(B,t)]/\tau, \tag{6}$$

$$f_\alpha(C',t+\delta t)=f_\alpha(C,t)+[f_\alpha^{\mathrm{eq}}(C,t)-f_\alpha(C,t)]/\tau, \tag{7}$$

$$f_\alpha(D',t+\delta t)=f_\alpha(D,t)+[f_\alpha^{\mathrm{eq}}(D,t)-f_\alpha(D,t)]/\tau, \tag{8}$$

$$f_\alpha(E',t+\delta t)=f_\alpha(E,t)+[f_\alpha^{\mathrm{eq}}(E,t)-f_\alpha(E,t)]/\tau. \tag{9}$$

Using the Taylor series expansion, $f_\alpha(P',t+\delta t)$, $f_\alpha(B',t+\delta t)$, $f_\alpha(C',t+\delta t)$, $f_\alpha(D',t+\delta t)$, $f_\alpha(E',t+\delta t)$, in the above equations can be approximated by the function and its derivatives at the mesh point P. As a result, Eqs. (5)–(9) can be reduced to

$$f_\alpha(P,t+\delta t)+\Delta x_P\frac{\partial f_\alpha(P,t+\delta t)}{\partial x}+\Delta y_P\frac{\partial f_\alpha(P,t+\delta t)}{\partial y}+\frac{1}{2}(\Delta x_P)^2\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x^2}+\frac{1}{2}(\Delta y_P)^2\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial y^2}+\Delta x_P\Delta y_P\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x\partial y}=f_\alpha(P,t)+[f_\alpha^{\mathrm{eq}}(P,t)-f_\alpha(P,t)]/\tau, \tag{10}$$

$$f_\alpha(P,t+\delta t)+\Delta x_B\frac{\partial f_\alpha(P,t+\delta t)}{\partial x}+\Delta y_B\frac{\partial f_\alpha(P,t+\delta t)}{\partial y}+\frac{1}{2}(\Delta x_B)^2\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x^2}+\frac{1}{2}(\Delta y_B)^2\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial y^2}+\Delta x_B\Delta y_B\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x\partial y}=f_\alpha(B,t)+[f_\alpha^{\mathrm{eq}}(B,t)-f_\alpha(B,t)]/\tau, \tag{11}$$

$$f_\alpha(P,t+\delta t)+\Delta x_C\frac{\partial f_\alpha(P,t+\delta t)}{\partial x}+\Delta y_C\frac{\partial f_\alpha(P,t+\delta t)}{\partial y}+\frac{1}{2}(\Delta x_C)^2\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x^2}+\frac{1}{2}(\Delta y_C)^2\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial y^2}+\Delta x_C\Delta y_C\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x\partial y}=f_\alpha(C,t)+[f_\alpha^{\mathrm{eq}}(C,t)-f_\alpha(C,t)]/\tau, \tag{12}$$

$$f_\alpha(P,t+\delta t)+\Delta x_D\frac{\partial f_\alpha(P,t+\delta t)}{\partial x}+\Delta y_D\frac{\partial f_\alpha(P,t+\delta t)}{\partial y}+\frac{1}{2}(\Delta x_D)^2\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x^2}+\frac{1}{2}(\Delta y_D)^2\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial y^2}+\Delta x_D\Delta y_D\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x\partial y}=f_\alpha(D,t)+[f_\alpha^{\mathrm{eq}}(D,t)-f_\alpha(D,t)]/\tau, \tag{13}$$

$$f_\alpha(P,t+\delta t)+\Delta x_E\frac{\partial f_\alpha(P,t+\delta t)}{\partial x}+\Delta y_E\frac{\partial f_\alpha(P,t+\delta t)}{\partial y}+\frac{1}{2}(\Delta x_E)^2\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x^2}+\frac{1}{2}(\Delta y_E)^2\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial y^2}$$
$$+\Delta x_E\Delta y_E\frac{\partial^2 f_\alpha(P,t+\delta t)}{\partial x\partial y}=f_\alpha(E,t)+[f_\alpha^{\mathrm{eq}}(E,t)-f_\alpha(E,t)]/\tau, \quad (14)$$

where

$$\Delta x_P=e_{\alpha x}\delta t,\quad \Delta y_P=e_{\alpha y}\delta t,$$
$$\Delta x_B=x_B+e_{\alpha x}\delta t-x_P,\quad \Delta y_B=y_B+e_{\alpha y}\delta t-y_P,$$
$$\Delta x_C=x_C+e_{\alpha x}\delta t-x_P,\quad \Delta y_C=y_C+e_{\alpha y}\delta t-y_P,$$
$$\Delta x_D=x_D+e_{\alpha x}\delta t-x_P,\quad \Delta y_D=y_D+e_{\alpha y}\delta t-y_P,$$
$$\Delta x_E=x_E+e_{\alpha x}\delta t-x_P,\quad \Delta y_E=y_E+e_{\alpha y}\delta t-y_P.$$

Equations (4), (10)–(14) form a system to solve for six unknowns. Now, we define

$$g_i=f_\alpha(x_i,y_i,t)+[f_\alpha^{\mathrm{eq}}(x_i,y_i,t)-f_\alpha(x_i,y_i,t)]/\tau, \quad (15)$$

$$\{s_i\}^T=\{1,\Delta x_i,\Delta y_i,(\Delta x_i)^2/2,(\Delta y_i)^2/2,\Delta x_i\Delta y_i\}, \quad (16)$$

$$\{V\}=\{f_\alpha,\partial f_\alpha/\partial x,\partial f_\alpha/\partial y,\partial^2 f_\alpha/\partial x^2,\partial^2 f_\alpha/\partial^2 y,\partial^2/\partial x\partial y^T\}^T, \quad (17)$$

where g_i is the post-collision state of the distribution function at the ith point and the time level t, $\{s_i\}^T$ is a vector with six elements formed by the coordinates of mesh points, $\{V\}$ is the vector of unknowns at the mesh point P, which also has six elements. Our target is to find its first element $V_1=f_\alpha(P,t+\delta t)$. With above definitions, Eqs. (4), (10)–(14) can be written as

$$g_i=\{s_i\}^T\{V\}=\sum_{j=1}^{6}s_{i,j}V_j,\quad i=P,A,B,C,D,E, \quad (18)$$

where $s_{i,j}$ is the jth element of the vector $\{s_i\}^T$ and V_j is the jth element of the vector $\{V\}$. Equation system (18) can be put into the following matrix form:

$$[S]\{V\}=\{g\}, \quad (19)$$

where $\{g\}=\{g_P,g_A,g_B,g_C,g_D,g_E\}^T$ and $[S]=[s_{i,j}]$. Note that the matrix $[S]$ can be computed once and stored for the application of Eq. (19) at all time levels. In practical applications, it was found that the matrix $[S]$ might be singular or ill conditioned. To overcome this difficulty and make the method be more general, we propose the following least squares-based LBM.

Equation (18) has six unknowns (elements of the vector $\{V\}$). If Eq. (18) is applied at more than six mesh points, then the system is over determined. For this case, the unknown vector can be decided from the least squares method. For simplicity, let the mesh point P be represented by the index $i=0$, and its adjacent points be represented by index $i=1,2,\dots,M$, where M is the number of neighboring points around P and it should be larger than 5. At each point, we can define an error in terms of Eq. (18), that is,

$$\mathrm{err}_i=g_i-\sum_{j=1}^{6}s_{i,j}V_j,\quad i=0,1,2,\dots,M. \quad (20)$$

The square sum of all the errors is defined as

$$E=\sum_{i=0}^{M}\mathrm{err}_i^2=\sum_{i=0}^{M}\left(g_i-\sum_{j=1}^{6}s_{i,j}V_j\right)^2. \quad (21)$$

To minimize the error E, we need to set $\partial E/\partial V_k=0,k=1,2,\dots,6$, which leads to

$$[S]^T[S]\{V\}=[S]^T\{g\}, \quad (22)$$

where $[S]$ is a $[(M+1)\times 6]$-dimensional matrix, which is given as

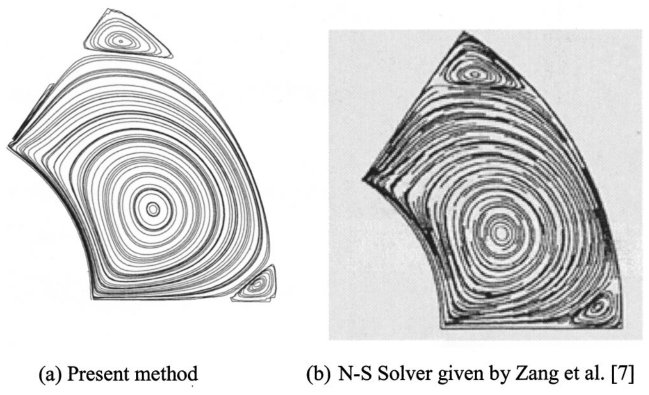

FIG. 2. Comparison of streamlines for flow in a polar cavity (Re=350). (a) Present method. (b) NS Solver given by Zang *et al.* [7].

$$[S]=\begin{bmatrix} 1 & \Delta x_0 & \Delta y_0 & (\Delta x_0)^2/2 & (\Delta y_0)^2/2 & \Delta x_0 \Delta y_0 \\ 1 & \Delta x_1 & \Delta y_1 & (\Delta x_1)^2/2 & (\Delta y_1)^2/2 & \Delta x_1 \Delta y_1 \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ 1 & \Delta x_M & \Delta y_M & (\Delta x_M)^2/2 & (\Delta y_M)^2/2 & \Delta x_M \Delta y_M \end{bmatrix}_{(M+1)\times 6}$$

and $\{g\}=\{g_0, g_1, \ldots, g_M\}^T$.
The Δx and Δy values in the matrix $[S]$ are given as

$$\Delta x_0 = e_{\alpha x}\delta t, \Delta y_0 = e_{\alpha y}\delta t, \tag{23a}$$

$$\Delta x_i = x_i + e_{\alpha x}\delta t - x_0,\ \ \Delta y_i = y_i + e_{\alpha y}\delta t - y_0,$$
$$\text{for}\ \ i=1,2,\ldots,M. \tag{23b}$$

Clearly, when the coordinates of mesh points are given, and the particle velocity and time step size are specified, the matrix $[S]$ is determined. Then from Eq. (22), we obtain

$$\{V\}=([S]^T[S])^{-1}[S]^T\{g\}=[A]\{g\}. \tag{24}$$

Note that $[A]$ is a $[6\times(M+1)]$-dimensional matrix. From Eq. (24), we can have

$$f_\alpha(x_0, y_0, t+\delta t) = V_1 = \sum_{k=1}^{M+1} a_{1,k} g_{k-1}, \tag{25}$$

where $a_{1,k}$ are the elements of the first row of the matrix $[A]$, which are precomputed before the LBM is applied. Note that the function g is evaluated at the time level t. So, Eq. (25) is actually an explicit form. In the above process, there is no requirement for the selection of neighboring points. In other words, Eq. (25) is nothing to do with the mesh structure. Thus, we can say that Eq. (25) is basically a meshless form. Although the proposed method has meshless feature, it is recommended to use a structured grid. This is because in our method, only the coordinates of mesh points are involved. When a structured grid is used, it is much easy to define the coordinates of mesh points. Furthermore, the α direction in Eq. (25) can be any direction. This implies that Eq. (25) can be uniformly applied to the different lattice models. We have successfully applied Eq. (25) to the D2Q9 model and the D2Q7 model, and the obtained results between these two models are exactly the same.

The implementation of the boundary condition for the new method is the same as the standard LBM. That is, at a boundary point, the distribution functions along all outward directions (point from the flow field to the boundary) are computed through Eq. (25), while the distribution functions along all inward directions (point from the boundary to flow field) are determined by the bounce back rule. Using Eq. (25), we have simulated many incompressible viscous flows. The obtained numerical results are very accurate and the convergence is very fast. The theoretical analysis and detailed implementation of our method will be shown in the full paper. Here, we only show some results for simulation of a polar cavity flow (inner surface has a rotational velocity) obtained by the present method. Figure 2 compares the streamlines obtained by the present method using a nonuniform mesh of 81×81 and the conventional Navier-Stokes solver given by Zang *et al.* [7]. Very good agreement is achieved in the size of the vortices and location of the separation and reattachment points. We have also done the ‘‘no flow’’ simulation in a square cavity by using nonuniform meshes, and found that when the uniform density and zero velocity are set at the beginning, the maximum velocity magnitude and the relative difference of the density can be remained in the order of 10^{-8} for all times. This implies that the balance condition in the method is well kept. Through the application, we may conclude that the proposed method is an efficient approach for simulation of flows involving complex geometry. It is especially useful for a nonuniform grid where the mesh is clustered towards the boundary.

[1] S. Chen and G. D. Doolen, Annu. Rev. Fluid Mech. **30**, 329 (1998).
[2] X. He, L.-S. Luo, and M. Dembo, J. Comput. Phys. **129**, 357 (1996).
[3] X. He and G. D. Dooler, Phys. Rev. E **56**, 434 (1997).
[4] R. Mei and W. Shyy, J. Comput. Phys. **143**, 426 (1998).
[5] H. Chen, Phys. Rev. E **58**, 3955 (1998).
[6] G. Peng, H. Xi, and C. Duncan, Phys. Rev. E **59**, 4675 (1999).
[7] Y. Zang, R. L. Street, and J. R. Koseff, J. Comput. Phys. **114**, 18 (1994).