IMPLICIT DISCRETIZATION AND NONUNIFORM MESH
REFINEMENT APPROACHES
FOR FD DISCRETIZATIONS OF LBGK MODELS*

JONAS TÖLKE, MANFRED KRAFCZYK1, MANUEL SCHULZ, and ERNST RANK

1St. Bauinformatik, Techn. Univ. München, Arcisstr. 21, D-80290, Germany
nym@delos.inf.bauwesen.tu-muenchen.de

RODOLFO BERRIOS
Elec. Ing. Mec., Universidad Central de Venezuela
Caracas, Los Chaguaramos, Caracas, Venezuela

Accepted 6 October 1998

After a short discussion of recent discretization techniques for the lattice-Boltzmann
equations we motivate and discuss some alternative approaches using implicit, nonuni-
form FD discretization and mesh refinement techniques. After presenting results of a
stability analysis we use an implicit approach to simulate a boundary layer test prob-
lem. The numerical results compare well to the reference solution when using strongly
refined meshes. Some basic ideas for a nonuniform mesh refinement (with non-cartesian
mesh topology) are introduced using the standard discretization procedure of alternating
collision and propagation.

Keywords: Lattice-Boltzmann; Hydrodynamics; Boundary Layer Problem; Implicit
Discretization.

1. Introduction

During the last decade computational physicists have come to believe that lattice-
Boltzmann approaches may be a useful complementary if not an efficient alterna-
tive discretization technique for a large class of flow problems. For an introductory
overview of the method see e.g. Refs. 1 and 2. Basically, this method solves a micro-
scopic kinetic equation governing the dynamics of a finite set of particle distribution
functions f_i from which macroscopic variables like pressure and velocities can be de-
duced as specific moments of these distributions. Using a Chapman–Enskog expa-
sion it can be shown,2 that the coupled dynamics of these moments in the low Mach

*This paper was presented at the 7th Int. Conf. on the Discrete Simulation of Fluids held at the

1143
number limit is described by the incompressible Navier–Stokes equations. In contrast to standard discretization procedures for the Navier–Stokes equations, no Poisson equation for the pressure needs to be solved, as the pressure is implicitly coupled to the particle distributions via a simple equation of state.

After discretizing the Boltzmann equation in phase space the evolution equations for the particle distributions read

\[
\frac{\partial f_\alpha(x, t)}{\partial t} + e_\alpha \nabla f_\alpha(x, t) = \Omega_\alpha(\{f_\beta(x, t)\}), \alpha, \beta \in \{0, \ldots, n\}.
\]

Using the Single Time Relaxation Approximation\textsuperscript{2} one has

\[
\Omega_\alpha = -\frac{1}{\tau}(f_\alpha(x, t) - f_\alpha^{\text{eq}}(x, t))
\]

with a microscopic relaxation time \(\tau\). In this work we use an improved two-dimensional model,\textsuperscript{3} being based on the so-called d2q9 model introduced in Ref. 2. Then we have a set of microscopic velocity vectors

\[
e_\alpha = \begin{cases}
(0, 0) & \alpha = 0 \\
(\cos((\alpha - 1)\pi/2), \sin((\alpha - 1)\pi/2))c & \alpha = 1, 2, 3, 4 \\
(\cos((\alpha - 5)\pi/2 + \pi/4), \sin((\alpha - 5)\pi/2 + \pi/4)\sqrt{2}c, & \alpha = 5, 6, 7, 8
\end{cases}
\]

where \(c = \delta_x/\delta t\) (lattice size and time-step). The equilibrium distribution functions are given by

\[
f_\alpha^{\text{eq}}(x, t) = \gamma_\alpha \left\{ \rho + \rho_0 \left(1 + 3\frac{(e_\alpha u)}{c^2} + \frac{9}{2} \frac{(e_\alpha u)^2}{c^4} - \frac{3}{2} \frac{u^2}{c^2} \right) \right\}
\]

with weights

\[
\gamma_\alpha = \begin{cases}
4 & \alpha = 0 \\
9 & \alpha = 1, 2, 3, 4 \\
1 & \alpha = 5, 6, 7, 8
\end{cases}
\]

The macroscopic pressure \(p\), density \(\rho\) and velocity \(u\) are defined by

\[
\rho = \sum_\alpha f_\alpha \quad u = \frac{1}{\rho_0} \sum_\alpha f_\alpha e_\alpha \quad p = c_s^2 \rho,
\]

where \(c_s^2\) is the speed of sound of this model and is given by \(c_s = c/\sqrt{3}\). Then the solution of Eq. (1) is equivalent to the solution of

\[
\frac{1}{c_s^2} \frac{\partial P}{\partial t} + \nabla \cdot u = 0
\]

\[
\frac{\partial u}{\partial t} + u \cdot \nabla u = -\nabla P + \nu \nabla^2 u,
\]
where \( P = p/\rho_0 \) and the kinematic viscosity is given by

\[
\nu = \frac{\tau \delta x^2}{3 \delta t}.
\]

(9)

Incompressibility of the flow requires \(|u| \ll c_s\).

The standard approach to solve Eq. (1) is to couple the physical discretization of the system defined by the set of \( e_\alpha \) and the numerical one by explicitly solving a set of first order finite difference equations obtained from Eq. (1) by a first order Taylor expansion

\[
f_\alpha(x + e_\alpha \delta t, t + \delta t) - f_\alpha(x, t) = \delta t \Omega_\alpha(x, t)
\]

(10)

requiring a uniform cartesian mesh of lattice spacing \( c \). Although this approach can be sufficient even for nontrivial flow problems as demonstrated e.g., in Ref. 4, it is usually accepted that for a wide class of engineering problems implicit discretization methods as well as locally refined meshes are of vital importance to obtain valid simulation results. Lately there have been developments of more sophisticated discretization approaches (see e.g., Refs. 5–8) but to our knowledge all these approaches basically used explicit algorithms on topologically cartesian grids.

2. The Test Case: Boundary Layer of a Flat Plate

2.1. The reference solution

Effects of viscosity are often confined to a thin layer (the so-called boundary layer) adjacent to the wall. A typical example is the flow over a flat plate (Fig. 1). The boundary layer thickness \( \delta \) is defined as the height \( y \) where the horizontal velocity \( u \) reaches 99% of the free-stream velocity \( u_\infty \). If \( \delta \) is very small compared to the longitudinal extension \( L \) of the domain, the gradient \( \partial u/\partial y \) is very large compared to the gradient \( \partial u/\partial x \) in the boundary layer. The Navier–Stokes equation can then be approximated by the boundary layer equations\(^9\)

\[
\frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \nu \frac{\partial^2 u}{\partial y^2} \quad \frac{\partial p}{\partial y} = 0 \quad \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0.
\]

(11)

For the boundary layer thickness \( \delta \) we have \( \delta/L \sim 1/\sqrt{Re} \), where \( Re = u_\infty * L/\nu \).

Note, that the condition \( \delta/L \ll 1 \) is not valid near the left corner. For the flow

---

![Fig. 1. Boundary layer.](image)
over a flat plate, the boundary conditions are \( u(y = 0) = u(y = 0) = 0 \) and \( u(y = \infty) = u_\infty \).

The transformation

\[
\eta = \frac{y}{\sqrt{\frac{\nu x}{u_\infty}}} ; \quad \psi = \sqrt{u_\infty \nu x} g(\eta)
\]

leads to the Blasius equation

\[
g(\eta)''' + \frac{1}{2} g(\eta) g(\eta)'' = 0
\]

with boundary conditions \( g(\eta = 0) = g'(\eta = 0) = 0 \) and \( g'(\eta = \infty) = 1 \). This nonlinear ordinary differential equation of third order has no explicit analytical solution, so a numerically solution is needed which can easily be obtained using a high level programming language like Maple.\(^{10}\)

2.2. Numerical discretization

Equations (1) are transport equations for \( f_\alpha \) with advection velocity \( e_\alpha \). For space discretization we use a standard first order FD upwind scheme. This space discretization introduces a numerical viscosity of order \( O(h f_{x x}) \), \( h \) being the mesh spacing. This numerical viscosity makes the system very robust, but has the drawback that it smooths out strong gradients. For time discretization an explicit Euler scheme is chosen. At the boundaries of the domain it is not always possible to apply upwind differencing, so one is forced to use a downwind difference.

2.3. Boundary Conditions for the numerical calculations

All boundary conditions are introduced via the relaxation operator \( f_{\alpha}^{\text{rel}}(p, u) \) (Eq. (4)). This implies that for an initial, short period the boundary conditions are not full-filled. The duration of this period depends on the aspect ratio \( \tau/h \) and is analyzed in the next subsection. The imposed boundary conditions for the

\[
\begin{align*}
\text{u} &= u_{\text{inf}} \\
\text{dv/dn} &= 0 \\
\text{u} &= u_{\text{inf}} \\
\text{v} &= 0 \\
\text{p} &= p_0 \\
\text{du/dx} &= 0 \\
\text{p} &= p_0 \\
\text{u} &= 0 \\
\text{v} &= 0
\end{align*}
\]

Fig. 2. Boundary conditions.
boundary-layer problem are pictured in Fig. (2). In all our computations we used \( u_\infty = 0.1 \).

2.4. **Stability analysis of the numerical scheme**

A powerful method for exploring the stability properties of a numerical scheme including the influence of boundary conditions is the eigenvalue analysis of the linearized spatial differential operator. One drawback is, that in general for complicated problems one has to compute the eigenvalues numerically. Therefore the parameter space usually can only be investigated for a finite number of points. The parameter of interest in our analysis is the relaxation time \( \tau \).

We expect answers to the following questions from a stability analysis:

- The maximum possible time-step \( \Delta t \) in an explicit scheme for varying \( \tau \)
- The detection of parameter ranges, in which the scheme is totally unstable (positive real parts of eigenvalues)
- The variation of the condition number of the Jacobian matrix for varying \( \tau \), which is an important measure influencing the performance of an implicit scheme
- Peculiarities in the eigenvalue spectrum

For the sake of self-containedness we now give a short overview about eigenvalue analysis of space discretized operators\(^{11}\) and its connection to an explicit time integration scheme.

2.4.1. **Analysis of space discretization**

After the space discretization of system (1) a set of nonlinear coupled ODE’s results:

\[
\frac{dF}{dt} = G(F) .
\]  

(14)

The values \( f_\alpha(i) \) of all grid nodes \( i \) are stored in a one-dimensional array \( F \) using an appropriate ordering. Linearization of the operator \( G \) about a point \( F_0 \) results in the following linear system

\[
\frac{dF}{dt} \approx J(F_0)F - J(F_0)F_0 + G(F_0) := J(F_0)F + Q(F_0) ,
\]  

(15)

where \( J(F_0) \) is the Jacobian of \( G(F) \) at point \( F_0 \). If the eigenvalues \( \lambda_j \) and eigenvectors \( V^j \) of the Jacobian are computed, system (15) can be decoupled. This procedure leads to a system of *uncoupled* ODE. Every ODE corresponds to one eigenmode (eigenvector):

\[
\frac{dw_j}{dt} = \lambda_j w_j + q_j ,
\]  

(16)

\( w_j \) being the time dependent coefficients of the eigenvectors and \( q_j \) being the coefficients of the expansion \( Q(F_0) = \sum_{j=1}^N q_j V^j \). These ODE can now be solved independently. The general solution of system (15) is then given by the sum of
all independent solutions. This is called a semi-analytical solution, because it is analytic in time, but with an approximation of the space derivatives, and is given by

$$F(t) = \sum_{j=1}^{N} \left( c_{0j} e^{\lambda_j t} + \frac{q_j}{\lambda_j} (e^{\lambda_j t} - 1) \right) V^j,$$

(17)

where $c_{0j}$ are the coefficients of the expansion of the initial state $F_0 = \sum_{j=1}^{N} c_{0j} V^j$. The first term on the right-hand side of Eq. (17) corresponds to the transient solution, the second corresponds to the steady state solution. The system of ODE’s will be stable, if the solution $F(t)$ remains bounded. This requires the real part of all eigenvalues to be negative or zero.

2.4.2. Analysis of coupled space-time discretization

For a scalar ODE

$$\frac{dw}{dt} = \lambda w + q,$$

(18)

the explicit Euler scheme with time-step $\Delta t$ and initial condition $w(0) = w_0$ yields the following iteration scheme

$$w^1 = \Delta t \lambda w^0 + w^0 + q = (1 + \Delta t \lambda)w^0 + \Delta t q$$

$$w^2 = (1 + \Delta t \lambda)w^1 + \Delta t q$$

$$= (1 + \Delta t \lambda)((1 + \Delta t \lambda)w^0 + \Delta t q) + \Delta t q$$

$$\vdots$$

$$w^n = (1 + \Delta t \lambda)^n w^0 + (1 + \Delta t \lambda)^n q - q.$$  

(19)

The scheme remains stable if

$$|1 + \lambda \Delta t| < 1.$$  

(20)

When solving system (14) using an explicit Euler scheme, this restriction has to be valid for all $\lambda_j$ and implies a maximum time-step $\Delta t_{\text{max}}$.

2.4.3. Eigenvalue analysis of the test problem

The boundary layer problem is discretized on a unit square with a uniform $11 \times 11$ mesh. The extension of the domain is $Lx = Ly = 1.0$, the mesh size is $h = 0.1$. The Jacobian is set up numerically and the eigenvalues are computed for different values of the relaxation time $\tau$. For each $\tau$ the Jacobian is computed at the beginning and the end (steady state) of the iteration using the LAPACK-software. In Fig. 3 the spectra of the eigenvalues for different values of $\tau$ are illustrated (at the beginning of the iteration). In Table 1 the extremal eigenvalues, the condition number $\kappa = |\lambda_{\text{max}}|/|\lambda_{\text{min}}|$ of the Jacobian $J$, and the maximum possible time-step
Fig. 3. Spectrum for (a) $\tau = 0.2$; (b) $\tau = 0.01$ and (c) $\tau = 0.00001$. 
The spectrum corresponding to $\tau = 0.3$ contains one eigenvalue with a positive real part indicating instability. Explicit and implicit computations on different meshes ($16 \times 16, 32 \times 32, 64 \times 64$) suggest the existence of an upper bound for $\tau$ with respect to the mesh size $h$: $\tau < \approx \frac{3}{2}h$.

For the explicit scheme we found a blow up of the solution even for very small time-steps down to $\Delta t = 10^{-6}$ and for the implicit scheme the Newton iteration (to be discussed later) did not converge and was terminated after 20 iterations. The value from the eigenvalue analysis suggests a larger maximum value $\tau < 2 - 3h$, but this value is only valid at the beginning of the iteration, because the Jacobian and therefore the eigenvalues are varying in each iteration.

- From the last column in Table 1 it can be seen, that for very large ratios $h/\tau$ the maximum possible time-step for a stable scheme must be smaller than $2\tau$. If the ratios $h/\tau$ becomes smaller, this restriction is not valid anymore.
- There is a clear separation of eigenvalues into two parts for decreasing $\tau$ (see Fig. 4). This observation has a significant impact on the influence of initial errors on the discretization of boundary conditions. If this initial error is decomposed into the corresponding eigenmodes, it is obvious that all components corresponding to eigenvalues of large negative real parts are damped quickly because they are being multiplied by $e^{\lambda_{min}t}$. A further inspection of the eigenmodes indicates that modes corresponding to eigenvalues with large negative real parts have a finite amplitude in $p$ and/or $u$ and/or $v$ at certain locations on the boundary when BCs for $p$ and/or $u$ and/or $v$ are specified. Therefore correct boundary
Fig. 4. Eigenmode (u-velocity) not linked to a boundary condition, Eigenmode (u-velocity) linked to a boundary condition, spurious pressure mode.

conditions will develop after a very small time period compared to the time period for achieving the steady state solution if the ratio $\tau/h$ is small enough. This justifies the incorporation of boundary conditions via $f_{eq}$.

- The condition number $\kappa$ of the Jacobian matrix becomes very large for small $\tau$. This is an important indicator for the convergence rate of implicit iterative solvers. Generally speaking, the higher $\kappa$, the worse the convergence rate.
- At nodes where a pressure boundary condition is specified, a spurious pressure mode is generated. This eigenmode has a nonzero pressure amplitude only at these boundary nodes and zero velocity amplitude. The implications of these modes are not clear to us and require further research. Such modes have also been encountered in discretizations on nonstaggered grids of the incompressible Navier–Stokes equations.\textsuperscript{14}
2.5. The implicit approach

A steady-state solution of Eq. (14) is

\[ \mathbf{G}(\mathbf{F}) = 0. \tag{21} \]

Almost all iterative algorithms for solving Eq. (21) can be written in a residual form

\[ \mathbf{P} \frac{\Delta \mathbf{F}^n}{\omega} = -\mathbf{R}^n, \tag{22} \]

where \( \mathbf{R}^n = \mathbf{G}(\mathbf{F}^n) \) is the residual, \( \mathbf{P} \) a preconditioner and \( \omega \) a relaxation parameter. The improved solution is \( \mathbf{F}^{n+1} = \mathbf{F}^n + \Delta \mathbf{F}^n \). Let \( \mathbf{J} \) be the Jacobian of the operator \( \mathbf{G} \) after the linearization about \( \mathbf{F}^n \). Using iterative matrix analysis\(^{13}\) it can be shown, that the iterative scheme will converge, if the spectral radius of the amplification matrix

\[ 1 - \omega \mathbf{P}^{-1} \mathbf{J} \tag{23} \]

is lower than or equal to one. The spectral radius of a matrix is given by the modulus of the largest eigenvalue. In the following two implicit discretization approaches will be discussed.

2.5.1. Point-Jacobi iteration

The choice \( \mathbf{P} = \text{diag}\{-1\} \) yields the Point-Jacobi method. This is the simplest iterative scheme and is nothing but an explicit Euler scheme in disguise. The parameter \( \omega \) acts as a pseudo-time and can be chosen to obtain the steady state solution as fast as possible. For the following analysis \( \lambda_{\text{max}} \) is the eigenvalue with the largest modulus and \( \lambda_{\text{min}} \) is the eigenvalue with the smallest modulus of the Jacobian matrix \( \mathbf{J} \). For stability reasons (see Eq. (20)) the maximum possible \( \omega \) is restricted to the value

\[ \omega < -2 \frac{\text{Re}(\lambda_{\text{max}})}{\lambda_{\text{max}}^2}. \tag{24} \]

An optimal value for \( \omega \) is given by\(^{11}\)

\[ \omega_{\text{opt}} = -2 \frac{1}{\text{Re}(\lambda_{\text{max}}) + \text{Re}(\lambda_{\text{min}})}. \tag{25} \]

Using standard analysis\(^{11}\) it can be shown that the asymptotic rate of convergence strongly depends on the condition number of \( \mathbf{J} \). From the eigenvalue analysis it is clear (Table 1), that convergence for \( \tau/h \ll 1 \), which is the case for high Reynolds number problems discretized by a moderate number of degrees of freedom (DOF) will be extremely poor. One possibility to overcome this problem is the use of other preconditioners as described in the following section.
2.5.2. Newton–Raphson iteration

If one uses the full Jacobian matrix as a preconditioner \( (P = J) \) in Eq. (22), the system converges very fast (3–6 iterations are typical), but this requires the solution of a nonsymmetric sparse linear system at each iteration step. There exist highly sophisticated algorithms for the solution of such systems, most of them based on iterative methods, but the number of iterations needed to solve the linear system themselves depends on the condition number \( \kappa(J) \). For the present computations a sparse linear solver from the “Numerical Recipes in Fortran 90”\(^{15} \) is used, which is based on a preconditioned biconjugate gradient method. The Jacobi matrix is computed by forward differencing.

2.5.3. A comparison of the Point–Jacobi and the Newton–Raphson method

As a reference value for the efficiency of the two methods the number of iterations needed to solve Eq. (21) was used. Two scenarios are considered: The first example uses decreasing values of \( \tau \) on identical meshes to achieve high Reynolds numbers. In the second example we increased the number of DOF to increase the Reynolds number, whereas \( \tau = h = 1 \) remains constant. For the Point–Jacobi method the relaxation parameter \( \omega \) was chosen in accordance to Eq. (25). Changes of this value during the iteration were neglected. For the Newton–Raphson method one iteration of the sparse matrix solver corresponds roughly to one time-step of the Point–Jacobi method. So the sum of all iterations of the sparse matrix solver are to be taken as a reference value. The stopping criterion was \( \| \Delta F \|_{\text{max}} < 1.0 \times 10^{-5} \).

First we show results for a small \( 11 \times 11 \) test problem. The mesh size is \( h = 0.1 \) and the extension of the domain is \( Lx = Ly = 1.0 \). The following table shows the number of iterations needed to solve the \( 11 \times 11 \) problem for different values of \( \tau \) with the Point–Jacobi method and the Newton–Raphson method. In Table 2 the number of Newton steps is given in parenthesis, the other number is the sum of all iterations needed by the sparse matrix solver. These results qualify the Newton–Raphson method as an extremely efficient solver for very high ratios \( h/\tau \).

<table>
<thead>
<tr>
<th>( \tau = 0.1 )</th>
<th>( \tau = 0.01 )</th>
<th>( \tau = 0.001 )</th>
<th>( \tau = 0.0001 )</th>
<th>( \tau = 0.00001 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point–Jacobi</td>
<td>279</td>
<td>1119</td>
<td>11505</td>
<td>152709</td>
</tr>
<tr>
<td>Newton–Raphson</td>
<td>506 (4)</td>
<td>785 (4)</td>
<td>872 (4)</td>
<td>888 (4)</td>
</tr>
</tbody>
</table>

Table 2. Number of for different values of \( \tau \).

<table>
<thead>
<tr>
<th>( 9 \times 9 )</th>
<th>( 17 \times 17 )</th>
<th>( 33 \times 33 )</th>
<th>( 65 \times 65 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point–Jacobi</td>
<td>166</td>
<td>419</td>
<td>915</td>
</tr>
<tr>
<td>Newton–Raphson</td>
<td>399 (4)</td>
<td>841 (4)</td>
<td>2020 (4)</td>
</tr>
</tbody>
</table>

Table 3. Number of Iterations for different numbers of DOF.
For the second example the values of $\tau = h = 1$ remain constant, but the number of DOF and thus the extension of the domain is increased. Table 3 lists the number of iterations needed. The reason for the poor performance of the sparse matrix solver used for the Newton–Raphson method is not clear to us and is subject to future research.

2.6. Mesh refinement

One advantage of the FD-based lattice–Boltzmann method is that one can use geometrical transformations to either approximate domains with curved boundaries or/and cluster points near a region where large gradients are expected. After one-dimensional stretchings $x = x(\xi)$ and $y = y(\eta)$ are applied, the lattice-Boltzmann equations become

$$
\left. \begin{array}{l}
\frac{\partial f_\alpha}{\partial t} + e_{\alpha\alpha} \frac{\partial f_\alpha}{\partial x} + e_{\alpha\eta} \frac{\partial f_\alpha}{\partial y} = -\frac{1}{\tau}(f_\alpha - f_\alpha^{eq}) \\
\alpha = 0, \ldots, 8.
\end{array} \right\}
$$

For our boundary-layer problem a mesh refinement at the singularity (the leading edge of the plate) is natural. A stretching mapping $[0, 1]$ onto itself was used in both directions:

$$
x = a\xi + (1 - a)\xi^b.
$$

Calculations on a $16 \times 16$ mesh with different values for parameter $b$ and a Reynolds number of 30 000 are performed. Plots of the $u$ and the $v$-velocity profiles perpendicular to the wall at the out-stream are given in Figs. 5 and 6 for $a = 1/10$ and $b \in \{2, 4, 6, 8\}$. The best numerical solution ("stretching 5" in Figs. 5 and 6) was obtained by the mapping function $x(\xi) = (1/2)((a - 1)((a + \xi)/(a - \xi) - 1)$ with $a = 1.01$.

Note that the velocity components differ about two orders of magnitude in size, thus the error of $v$ is larger than that of $u$. In order to assess the computational efficiency of our approach some remarks seem appropriate:

- The overall computation time for the best numerical solution shown above on a 300 MHz Linux-Pentium PC was about 100 s.
- Experience shows that in order to obtain simulation results using the standard LBGK procedure (via Eq. (10)) with $u_\infty = 0.1$ and a viscosity $\nu \approx 0.005$ in order to obtain Re = 30000 would require a mesh of about 1500 $\times$ 1500 nodes.
- In order to obtain a steady state solution for this system one needs about $O(10^5)$ to $O(10^6)$ time-steps.
- A highly optimized code offering a nodal update rate of about $5 \times 10^5$ per second on the same hardware as used above thus would (apart from memory issues) need more than a month to solve this problem.
3. Local Mesh Refinement Using the Standard Explicit Discretization Approach

When simulating a numerical problem an approach of optimal efficiency implies a locally adaptive procedure being capable of estimating or at least indicating the local discretization error in certain regions of the simulation domain. Apart from the fact that for LBGK methods such an error indicator let alone estimator is not in sight, e.g., one could think of using local strain rate information to indicate a local enhancement or decrement of DOF. This inevitably requires the possibility to use discretization meshes which do not have a uniform topology and which allow local generation or annihilation of mesh nodes. For Lattice Gases it was shown that a coupling of a set of meshes \( \{M_k|k \in \{0, \ldots, l - 1\}\} \) with node distances \( \Delta x_k = \Delta x_0 2^{-k} \) is possible and allows a substantial decrease of DOF, e.g., for
external 3D flows around bluff bodies. In the same spirit as Ref. 16 we then have one set of equations analogously to Eq. (10)

\[ f_\alpha(x + e_\alpha \Delta t_k, t + \Delta t_k) - f_\alpha(x, t) = \Delta t_k \Omega_\alpha(x, t) . \]  

(28)

Each of these sets of equations is discretized on one of 1 meshes using specific length and time scales differing by powers of 2: \( \Delta t_k = \Delta t_0 2^{-k} \) and \( \Delta x_k = \Delta x_0 2^{-k} \). The macroscopic flow velocity \( \mathbf{u} \) on each node can be uniquely defined via Eq. (6) because \( \Delta x_k / \Delta t_k = \Delta x_0 / \Delta t_0 \). Speaking in terms of the standard discretization of the LBGK equations, propagation and collision on a mesh with nodal distance \( \Delta x_k \) is applied each \( 2^{-k} \Delta t_0 \). As the physical value of the kinematic viscosity should be constant across the system and especially at the interfaces of meshes \( k, k \pm 1 \) (here we only allow coupling of meshes differing by a factor of 2 with respect to node distance and time-step) we have to adjust \( \tau \) which now becomes \( \tau_k \). Redefining the kinematic viscosity as

\[ \nu = \frac{(2\tau - 1)}{6} \frac{\delta x^2}{\delta t} , \]  

(29)

in order to have second order accuracy when using Eq. (10) to discretise Eq. (1), it can be easily seen that

\[ \tau_k = \left( \tau_0 - \frac{1}{2} \right) 2^k + \frac{1}{2} \]  

(30)

will ensure a mesh invariant value for the physical kinematic viscosity. The remaining question is how to couple the distributions across the interfaces of different meshes. As there is no unique way to do this, we started using a Shepard interpolation\(^\text{17}\) of the flow variables \( p, \mathbf{u} \) for nodes being adjacent to a mesh interface. These values are introduced into the collision operator \( \Omega \) during the relaxation step and thus implicitly change the interface distributions. Preliminary numerical examples of channel flows using this approach seem promising (at least for stationary problems) but require a more rigorous analysis concerning stability and convergence properties and will be reported in a forthcoming paper.

4. Discussion

The results of the numerical examples shown in this work indicate that the use of implicit discretization methods for the LBGK equations may be of significant advantage at least for stationary flows of high Reynolds numbers in geometries of moderate complexity. The use of such methods for transient flows requires further research but seems to be conceptually straightforward. Of course these mathematically more complex algorithms lack the elegance and simplicity of the former “standard” solution technique having their roots in the corresponding Lattice Gas approaches.

Stability and consistency issues of implicit methods for LBGK equations are far from being fully explored and a lot of work remains to be done before one can hope
to decide if and under which circumstances the solution of the first order LBGK equations is easier and/or can be obtained more efficiently than the corresponding second order Navier-Stokes equations.

References