Acceleration of steady-state lattice Boltzmann simulations on non-uniform mesh using local time step method

Taro Imamura[a,*], Kojiro Suzuki[b], Takashi Nakamura[a], Masahiro Yoshida[a]

[a] *Japan Aerospace Exploration Agency, Institute of Space Technology and Aeronautics, 7-44-1, Jindai-ji Higashi-mati, Chofu-si, Tokyo 182-8522, Japan*

[b] *Department of Advanced Energy, The University of Tokyo, 5-1-5 Kashiwa-no-ha, Kashiwa, Chiba 277-8562, Japan*

Journal of Computational Physics 202 (2005) 645–663

Received 29 December 2003; revised 21 June 2004; accepted 2 August 2004

Presenter: Chen Peng Chung

Outline

General Interpolation LBM (GILBM)

GILBM and ISLBM

Extend **Interpolation-Supplemented LBM (ISLBM)** to general curvilinear coordinates through conformal mapping, without changing the lattice system. But the point of mapping is How can I deal with discrete particle velocity set.

Key Features:

- Cartesian $\rightarrow$ Curvilinear via coordinate mapping
- Curved particle paths in computational domain
- Lattice structure remains unchanged
- Pull-back interpolation for streaming

$$f_\alpha(\vec{\xi}, t+\Delta t) = f^\star_\alpha(\vec{\xi} - \Delta\vec{\xi}_\alpha, t) \tag{1}$$

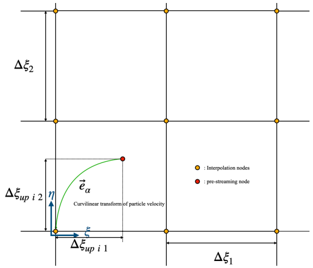


Figure: GILBM: curved paths in ξ-space

Jacobian Relation and Lamé Coefficients

Jacobian Transformation Matrix:

$$\begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} = \frac{1}{J} \begin{bmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{bmatrix}, \quad J = x_\xi y_\eta - x_\eta y_\xi$$

Lamé Coefficient & Unit Vector:

$$h_1 \equiv \left| \frac{\partial \vec{r}}{\partial q_1} \right|, \quad \vec{e}_1 \equiv \frac{1}{h_1} \frac{\partial \vec{r}}{\partial q_1}$$

Position vector differentials:

$$\frac{\partial \vec{r}}{\partial \xi} = h_\xi \vec{e}_\xi = x_\xi \vec{e}_x + y_\xi \vec{e}_y$$
$$\frac{\partial \vec{r}}{\partial \eta} = h_\eta \vec{e}_\eta = x_\eta \vec{e}_x + y_\eta \vec{e}_y$$

Physical Meaning:

- h_i: Scale factor (metric coefficient)
- $\vec{e}_i$: Local tangent direction
- J: Jacobian determinant (area scaling)
- Transforms derivatives between coordinates

Lattice Boltzmann Equation in Curvilinear Coordinates

Standard LBE (BGK Collision Operator):

$$f_\alpha(\vec{x}+\vec{c}_\alpha \Delta t, t+\Delta t) = f_\alpha(\vec{x}, t) + \omega(f_\alpha - f_\alpha^{eq})$$

LBE with MRT Collision Operator:

$$\mathrm{M}\vec{f}(\vec{x}+\vec{c}_\alpha \Delta t, t+\Delta t) = \mathrm{M}\vec{f}(\vec{x}, t) + \mathrm{SM}\,(\vec{f} - \vec{f}^{eq})$$

Collision Step:
Position appears only at grid nodes; collision has no spatial derivatives.
$\Rightarrow$ Direct substitution: $\vec{x} \to \vec{\xi}$

$$\boxed{f_i^\star(\vec{\xi}, t) = f_i + \omega(f_i^{eq} - f_i)}$$

Streaming Step:
Requires path integration of particle velocity in curvilinear space.

$$\boxed{f_\alpha(\vec{\xi}, t) = f_\alpha^\star(\vec{\xi} - \Delta\vec{\xi}_\alpha, t)}$$

(Pull-back from pre-streaming position via interpolation)

Contravariant Velocity (Discrete Velocity in Curvilinear Coordinates)

Definition

Transform Cartesian lattice velocity to curvilinear coordinates:

$$\boxed{e_\alpha^j = c_\alpha^i \frac{\partial \xi_j}{\partial x_i}} \dots \text{(assume laatice speed = 1)}$$

- c_α^i: Dimensionless discrete velocity (e.g., D2Q9: $0, \pm 1$)
- $\frac{\partial \xi_j}{\partial x_i}$: Coordinate transformation (from Jacobian)
- $\vec{e}_\alpha^{\,j}$: Shows velocity distortion due to curvature

Volumetric Lattice Boltzmann in Curvilinear Coordinates:

$$\vec{e}_\alpha = \underbrace{c_\alpha^i}_{\text{an integer}} \cdot \underbrace{\vec{g}_i(\vec{\xi})}_{\text{tangent vector}} \cdot \underbrace{\frac{\Delta x}{\Delta t}}_{\text{lattice speed}\approx 1}$$

Redefine the particle's discrete velocity: Enforce the direction of velocity to be the same as the tangent vector.

Path Integration: 2nd Order Runge-Kutta

From straming step :

$$f_\alpha(\vec{\xi}, t + \Delta t) = f^\star_\alpha(\vec{\xi} - \delta\vec{\xi}_\alpha, t)$$

In curvilinear coordinate, we have to calculate the position of the pre-streaming particles.

Streaming Length:

$$\delta\vec{\xi}_\alpha = \int_0^{\Delta t} \vec{e}_\alpha(\vec{\xi}(t))\, dt$$

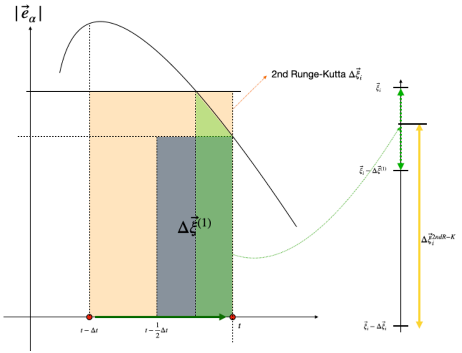
2nd Order Runge-Kutta:

$$\text{Step 1:} \quad \Delta\vec{\xi}^{(1)}_\alpha = \tfrac{1}{2}\Delta t\, \vec{e}_\alpha(\vec{\xi})$$

$$\text{Step 2:} \quad \Delta\vec{\xi}_\alpha = \Delta t\, \vec{e}_\alpha(\vec{\xi} - \Delta\vec{\xi}^{(1)}_\alpha)$$

Why not 1st order Euler?

- Euler: $O(\Delta t)$ error $\rightarrow$ insufficient
- Need $O(\Delta t^2)$ to recover Navier-Stokes
- Stress term requires 2nd order accuracy



$|\vec{e}_\alpha|$
2nd Runge-Kutta $\Delta\vec{\xi}_i$
$\vec{\xi}_i$
$\vec{\xi}_i - \Delta\vec{\xi}^{(1)}$
$\Delta\vec{\xi}^{(1)}$
$\Delta\vec{\xi}_i^{2ndR-K}$
$t - \Delta t$
$t - \frac{1}{2}\Delta t$
t
$\vec{\xi}_i - \Delta\vec{\xi}_i$

Path Integration: Accuracy Analysis

material derivative for the discrete-velocity Boltzmann equation:

$$\mathcal{D}_t = \partial_t + \frac{\delta\vec{\xi}_\alpha}{\delta t}\cdot\vec{\nabla}_\xi$$

and using Taylor expansion on lattice Bolrtzmann equation :

$$\left(\partial_t + \frac{\delta\vec{\xi}_\alpha}{\delta t}\cdot\vec{\nabla}_\xi\right) f_\alpha + \frac{1}{2}\left(\partial_t + \frac{\delta\vec{\xi}_\alpha}{\delta t}\cdot\vec{\nabla}_\xi\right)^2 f_\alpha \Delta t\Bigg|_{(\vec{\xi}-\Delta\vec{\xi}_\alpha,t)} + O(\Delta t^2) = \omega(f_\alpha(\vec{\xi}-\Delta\vec{\xi}_\alpha,t) - f_\alpha^{eq}(\rho,\vec{u},t))$$

Therefore, we must use a method that provides higher-order temporal accuracy for the path integration, ensuring that the differential term of the velocity-discrete Boltzmann equation has at least second-order temporal accuracy.

Multi-Scale Expansion

Three Time Scales:

- $K^{(0)}$: Collision & streaming (mesoscopic)
- $K^{(1)}$: Convection / advection (macroscopic)
- $K^{(2)}$: Diffusion (macroscopic)

For three types of time scales, we can define three time variables and two types of position vector :

Scale Operators:

$$\partial_t = K\partial_t^{(1)} + K^2\partial_t^{(2)}$$
$$\vec{\nabla} = K\vec{\nabla}^{(1)}$$

Distribution Expansion:

$$f_\alpha = f_\alpha^{eq} + Kf_\alpha^{(1)} + K^2 f_\alpha^{(2)} + \cdots$$

Order K^1 Equation:

$$\left(\partial_t^{(1)} + \vec{e}_\alpha \cdot \vec{\nabla}^{(1)}\right) f_\alpha^{eq} = -\frac{1}{\tau} f_\alpha^{(1)}$$

Order K^2 Equation:

$$\left(\partial_t^{(2)} + \vec{e}_\alpha \cdot \vec{\nabla}^{(2)}\right) f_\alpha^{eq} + \left(1 - \frac{\delta t}{2\tau}\right)$$
$$\times \left(\partial_t^{(1)} + \vec{e}_\alpha \cdot \vec{\nabla}^{(1)}\right) f_\alpha^{(1)} = -\frac{1}{\tau} f_\alpha^{(2)}$$

Boundary Condition Overview

Strategy: In GILBM, we enforce boundary conditions by constraining macroscopic parameters at boundary nodes (nodes between solid and fluid regions).

Distribution function at boundaries:

$$\begin{aligned} f_\alpha\big|_{\text{b.c.}} &= f_\alpha^{\text{eq}} + f_\alpha^{(1)} \\ &= f_\alpha^{\text{eq}} + \omega \Delta t f_\alpha^{\text{eq}} \left(\frac{(c_\alpha^i - u_i)(c_\alpha^j - u_j)}{c_s^2} \frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_j} \right) \end{aligned} \tag{2}$$

Velocity gradient in curvilinear coordinates:

$$\frac{\partial u_i}{\partial x_j} = \frac{\partial u_i}{\partial \xi_k} \frac{\partial \xi_k}{\partial x_j}$$

The second have been caculated in start of the simulation.

First-Order Non-Equilibrium Distribution (1/2)

From Multi-Scale Analysis:

$$\boxed{f_\alpha^{(1)} = -\omega \delta t \left(\partial_t^{(1)} + \vec{e}_\alpha \cdot \vec{\nabla}^{(1)} \right) f_\alpha^{\mathrm{eq}}} \tag{3}$$

Chain Rule for f_α^{eq} Derivatives:

$$\partial_t^{(1)} f_\alpha^{eq} = \frac{\partial f_\alpha^{eq}}{\partial u_i} \partial_t^{(1)} u_i + \frac{\partial f_\alpha^{eq}}{\partial \rho} \partial_t^{(1)} \rho$$

$$\vec{\nabla}^{(1)} f_\alpha^{eq} = \frac{\partial f_\alpha^{eq}}{\partial u_i} \vec{\nabla}^{(1)} u_i + \frac{\partial f_\alpha^{eq}}{\partial \rho} \vec{\nabla}^{(1)} \rho$$

Partial Derivatives of f_α^{eq}:

$$\frac{\partial f_\alpha^{eq}}{\partial u_i} \approx \frac{e_\alpha^i - u_i}{c_s^2} f_\alpha^{eq}, \quad \frac{\partial f_\alpha^{eq}}{\partial \rho} = \frac{1}{\rho} f_\alpha^{eq}$$

First-Order Non-Equilibrium Distribution (2/2)

Conservation Laws ($K^{(1)}$ scale):

$$\partial_t^{(1)}\rho = -\vec{\nabla}^{(1)} \cdot (\rho\vec{u}), \quad \partial_t^{(1)}\vec{u} = -\vec{u} \cdot \vec{\nabla}^{(1)}\vec{u} - \frac{1}{\rho}\vec{\nabla}^{(1)}p$$

Boundary Assumptions:

1. Constant pressure: $\nabla p = 0$
2. Incompressible: $\nabla \rho = 0$

Final Result — Distribution at Boundary:

$$\boxed{f_\alpha\big|_{bc} = f_\alpha^{eq} + \omega\delta t\left(\frac{(e_\alpha^i - u_i)(e_\alpha^\beta - u_\beta)}{c_s^2}\frac{\partial u_i}{\partial x_\beta} - \frac{\partial u_\beta}{\partial x_\beta}\right)f_\alpha^{eq}}$$

This allows boundary conditions to be enforced through macroscopic variables only.

GILBM Initialization & Runtime

GILBM Initialization:

1. Set Re, relaxation factor ω
2. Contravariant velocity:

$$e_\alpha^i = c_\alpha^\beta \frac{\partial \xi_i}{\partial x_\beta}$$

3. global time step : (Δt_g):

$$\Delta t_g = CFL \cdot min_{i,j,k,\alpha} \left| \frac{1}{e_\alpha^i\big|_{j,k}} \right|$$

4. Relaxation time: $\tau = \Delta t/\omega$
5. Viscosity: $\nu = (\tau - 0.5\Delta t)c_s^2$

Runtime Loop:

1. Interpolation :

$$\begin{aligned} f_i(\vec{r},t) &= f_i(x_i+\delta x, y_j+\delta y, z_k+\delta z, t) \\ &= \sum_{m=-3}^{3}\sum_{n=-3}^{3}\sum_{q=-3}^{3} f_i(x_{i+m}, y_{j+n}, z_{k+q}, t)\cdot \\ &\quad \prod_{\substack{\dot m=-3\\ \dot m\neq m}}^{3} \frac{\delta x-\dot m}{m-\dot m}\cdot \prod_{\substack{\dot n=-3\\ \dot n\neq n}}^{3} \frac{\delta y-\dot n}{n-\dot n}\cdot \prod_{\substack{\dot q=-3\\ \dot q\neq q}}^{3} \frac{\delta z-\dot q}{q-\dot q} \end{aligned} \tag{4}$$

2. **Streaming:** $f_i(\vec{\xi}) = f_i^\star(\vec{\xi} - \Delta\vec{\xi}_i)$ (with interpolation)
3. **Boundary:** Apply $f_\alpha|_{bc}$
4. **Collision:** $f_i^\star = f_i + \omega(f_i^{eq} - f_i)$
5. **Macroscopic:** $\rho = \sum_i f_i,\ \vec{u} = \frac{1}{\rho}\sum_i f_i\vec{c}_i$

Pre-calculation reduces runtime by $\sim$50%

Summary

GILBM: Coordinate Transform:

1. **Curvarture** : Jacobian relation maps Cartesian $\leftrightarrow$ Curvilinear, and pre-calculates the coefficients.
2. **Discrete Velocity** : Mulpling the transform coefficient to get contravariant velocity in curvilinear coordinates.
3. **Bouncdary Conditon** : Using the macroscopic parameters to enforce the limitation to distriobution function at the boundary nodes.

Limitation: MRT operator extension to curvilinear coordinates not yet developed. Furthermore, we can see that this method essentially still operates in the curvilinear coordinate system, but transforms the computational domain to appear as a Cartesian grid. However, many other aspects still require coordinate transformation.