Block iterative lattice Boltzmann algorithm for linear oscillatory flow in the frequency domain

Hang Kang [a], Yong Shi [a,*] and Yuying Yan [a, b]

[a] *Research Centre for Fluids and Thermal Engineering, The University of Nottingham Ningbo China, Ningbo 315100, People's Republic of Chi*na

[b] *Research Group of Fluids and Thermal Engineering, The University of Nottingham, Nottingham NG7 2RD, United Kingdom*

Abstract

A recent article (Y. Shi and John E. Sader, Phys. Rev. E **81**, 036706 (2010)) developed a linear lattice Boltzmann (LB) model in the frequency domain to characterize the performance of micro- and nanoelectromechanical systems (M/NEMS). Nonetheless, its numerical algorithm is formulated in the conventional time-marching form with addition of a virtual time scale. In this article, we propose a different algorithm to solve such a linear frequency-dependent LB model using the block iteration scheme consisting of the tri-diagonal matrix method and Jacobi line iteration. This change in the LB algorithm leads to straightforward modelling of linear oscillatory flow in the frequency domain without mimicking a numerical time evolution. Through simulating the one-dimensional oscillatory Couette flow and two-dimensional flow around an oscillating circular cylinder, we examined numerical accuracy of the block iterative LB algorithm proposed in this article. Importantly, we also explored modifications under this block-iterative LB algorithmic framework through use of other prevailing computational fluid dynamics (CFD) techniques. Computational efficiency of these original and modified block iterative LB algorithms was compared with that of the conventional time-marching LB algorithm. The numerical results in this article demonstrate the block iterative LB algorithm is a useful alternative numerical solver exhibiting nearly 2nd order accuracy for simulating frequency-dependent linear oscillatory flow in MEMS and NEMS. Our simulations also reveal rich extensions of this block iterative LB algorithm through combining the LB theory with advanced CFD numerical techniques.

[*] Corresponding author. Tel: +86 (0)574 88180000 (Ext. 9413); fax: +86 (0)574 88180715
E-mail addresses: Yong.Shi@nottingham.edu.cn (Y. Shi)

Keywords: Lattice Boltzmann algorithm; Linear Oscillatory flow; Block iteration; TDMA

1. Introduction

Over the last twenty-five years, there has been a tremendous surge of interests in the lattice Boltzmann (LB) method, which spurred its rapid and productive development for modelling a wide variety of physical processes [1-3]. In particular, the LB method achieved great successes in simulating complex fluid transport phenomena, including, but not limited to, multicomponent/multiphase flow [4, 5], suspension flow [6], flow in porous media [7, 8], [9], flow coupled with heat and mass transfer [10, 11] and even turbulence [12]. From the numerical point of view, the LB method possesses many distinguished advantages over conventional computational fluid dynamics (CFD) approaches, such as its simple formulas, parallel algorithmic structure and intrinsic particle-dynamics related framework [13]. Especially, the last feature enables the method employs some simple heuristic particle dynamic like treatments to tackle tough numerical issues. One known representative is the bounce-back for no-slip boundary conditions on solid walls [14]. Significantly, the LB method has a direct link to the Boltzmann equation with Bhatnagar-Gross-Krook assumption (Boltzmann BGK equation) [15, 16]. Many LB models for continuum flow in the literature are derived from the Boltzmann-BGK equation in the limit of low Mach number by appropriate discretization in the time, physical space and particle-velocity space [15]. This theoretical foundation in the Boltzmann theory triggers recent intensive efforts in investigating the LB capability for simulating flow beyond the Navier-Stokes order. Among various developments are the effective mean free path models [17, 18], high-order LB models [19, 20], half-space LB models [21-22], *etc*.

During almost the same period, we also witnessed great strides in micro-fabrication technologies and nanoscience [23, 24]. Plenty of micro-and nanosize electromechanical systems (M/NEMS) with different functions were designed and manufactured in elaborate structures [24-26]. The broad applications of these M/NEMS provide direct practical relevance of the LB models for flow beyond the Navier-Stokes order as the characteristic length scales of some M/NEMS are comparable to the mean free path of working fluids, leading to pronounced non-continuum effects in flow [24, 27].

On the other hand, not all flows in M/NEMS are non-continuum nonetheless. Many liquid flows over wetting surfaces and even some gas flows in these miniature devices still satisfy the Navier-Stokes equations subject to no-slip boundary conditions [28, 29]. In

comparison to macroscale flow, however, the continuum flow in M/NEMS manifest itself some distinct transport characteristics, e.g., the dominant viscous-force effects [28]. Especially, flow in some cases undergoes periodical oscillation driven by the resonating components imbedded in M/NEMS [30, 31]. In the literature, Y. Shi *et al*. proposed a LB model different from the classical version to describe this type of continuum flows in M/NEMS [32]. They derived a LB model from the linearized Boltzmann BGK equation, and formulated it depending on the frequency of oscillation, instead of the conventional time scale. In so doing, this linearized LB model eliminates the intrinsic nonlinearity of the LB method corresponding to advection, and is able to treat the oscillating structures in the flow as fixed boundaries in its simulation [32]. Importantly, the model outputs frequency-dependent numerical results. All these features make such a linearized LB model as a favourable numerical tool for modelling oscillatory flow in M/NEMS, where nonlinear advection is rather weak due to the very small Reynolds number and measurements in terms of frequency are preferable for characterizing system performance.

However, we note this linearized LB model developed in the frequency domain was ultimately solved numerically through use of the time-marching method [32], though it is irrelevant to any time scale. In the literature, a massive majority, if not all, of LB models are formulated depending on time, regardless of whether they originate from the lattice-gas cellular automata [33, 34] or the Boltzmann BGK equation [15, 16]. It is thus not surprising that the time-marching algorithm gain a prevailing position in the LB numerical implementation. The linearized frequency-based LB model in Ref. [32] followed this convention, which introduced a virtual time in its framework and modified its equation with addition of a derivative of this time scale. As such, the model was again solved using the time-marching method as an evolution process in the frequency domain. In this case, only "steady-state" results are meaningful and used as true numerical outputs of the simulation.

Numerically, there do not exist any prerequisites to enforce a LB algorithm to be devised by the time-marching method. This point is of particular importance for the above LB model in the frequency domain [32] as its inherent linearity and time independence allow flexible choices of numerical methods to constitute its algorithm. Actually, such a model turns into a sparse banded linear system of algebraic equations after discretizing the physical space and particle-velocity space. These resulting linear equations can be well solved either by direct or iterative numerical methods with the unnecessary introduction of a virtual time scale in the frequency domain at all [35, 36]. Two direct numerical methods, i.e., Cramer's rule [37] and Gaussian elimination [35], are usually referred to in CFD studies. However, both methods

necessitate formidable computational costs when solving a large linear system of N algebraic equations for N unknowns, where $N \gg 1$. In a Gaussian elimination, the number of arithmetic operations is approximately proportional to N^3, and the operations in a Cramer's rule-based computation will dramatically increase up to the order of $(N+1)!$ [36]. This computational inefficiency has significantly hindered applications of the direct methods, especially for multi-dimensional problems. Iterative methods is another category to attack a large linear system of equations using a different computational strategy. These methods make use of the matrix splitting technique to derive from the linear equations a sequence amenable to iteration [35, 36]. Through this sequence, iterative methods compute the unknowns at the n^{th} step using the results from the previous levels, and repeat such a recurrence until convergence is reached. The key for a good iterative method is to ensure that the designed sequence is convergent or conditionally convergent, and its iteration proceeds toward convergence at a fast rate [35, 36].

Interestingly, nowadays few CFD studies construct a numerical algorithm based on only one type of methods. Instead, integration of a direct method with an iterative method is a widespread treatment in the CFD simulation for two- and three-dimensional (2 and 3D) problems. An example is the so-called block iteration, in which a direct method is used to solve the unknowns simultaneously in one dimension while those in the other dimensions are updated in an iterative manner [36]. In this article, we apply the thought of block iteration to solve the linearized LB model in the frequency domain. To be specific, we construct a purely frequency-dependent LB numerical algorithm based on a block iteration scheme consisting of the tri-diagonal matrix algorithm (TDMA) [38] and Jacobi line iteration (JLI) [36].

The TDMA is a simplified Gaussian elimination, which is known for its high efficiency for solving the large tridiagonal system of algebraic equations. In comparison to other Gaussian elimination techniques, the operations in a TDMA-based computation are just in the order of N. Another advantage of the TDMA is it has a large variety of variants for different problems. In this article, we will use one of its variants, i.e., the cyclic tri-diagonal matrix algorithm (CTDMA), to simulate flow subject to periodic boundary conditions [36, 39]. Crucially, on top of the JLI just mentioned, this article also explores a stretch of the block iterative LB algorithm (BLB) through use of other more efficient iterative methods. We develop a set of modified BLB algorithms based on the Gaussian-Seidel line iteration (SLI) [36], alternative direction iteration (ADI) [36, 40] and over relaxing (OR) [36]. An attempt of non-uniform grids is also made in our simulation. Computational efficiency of these LB

algorithms are carefully examined in comparison to that of the original BLB (JLI) version and the conventional time-marching LB (TLB) algorithm constructed based on a virtual time scale [32], respectively.

The article is organized as follows: we first briefly introduce the linearized Boltzmann BGK equation and the corresponding LB model in the frequency domain in Section 2. In Section 3, the BLB algorithm based on the TDMA and JLI is developed. This algorithm is then applied to simulate the oscillatory Couette flow and flow around an oscillating circular cylinder in Section 4. Its numerical accuracy is validated by the available analytical solutions. Section 4 also uses the oscillatory Couette flow as a test to analyze computational efficiency of the BLB algorithms modified by the SLI, ADI, OR and non-uniform. A comparison of these results with those obtained by the original BLB (JLI) and TLB algorithms is elaborated. Finally, we draw our conclusions in Section 5, and relegate the mathematical details pertinent to the modified BLB algorithms to Appendix A – D.

2. Linearized lattice Boltzmann model in the frequency domain

In this section, we use the linearized Boltzmann Bhatnagar-Gross-Krook (BGK) equation as a kinetic model for linear oscillatory flows, and present its LB version in the frequency domain.

2.1. Linearized Boltzmann-BGK equation

In the kinetic theory of gases, the well-known linearized Boltzmann BGK equation is [41]

$$\frac{\partial h}{\partial t}+\mathbf{c}\cdot\frac{\partial h}{\partial \mathbf{r}}=-\frac{1}{\lambda}(h-h^{eq}), \tag{1}$$

where t, $\mathbf{r}$ and $\mathbf{c}$ represent the time, particle position and particle velocity, respectively. λ is the relaxation time and h is a perturbation to the distribution function f from the global equilibrium $\overline{f}^{eq}$. It is defined as

$$h=\frac{f}{\overline{f}^{eq}}-1. \tag{2}$$

In the right hand side of Eq. (1), h^{eq} represents a perturbation to the local equilibrium from $\overline{f}^{eq}$. This function can be formulated as a polynomial in terms of the fluid density perturbation $\delta\rho$ and velocity perturbation **u**, i.e.,

$$h^{eq} = \frac{\delta\rho}{\rho_0} + \frac{\mathbf{c}\cdot\mathbf{u}}{R_0 T_0}, \tag{3}$$

where ρ_0 and T_0 are the fluid density and temperature at the global equilibrium, respectively. R_0 is the gas constant.

As pointed out in Ref. [32], use of the linear Boltzmann-BGK equation, Eq. (1), in the frequency domain is much more convenient for simulating oscillatory flow in M/NEMS. We thus apply the time-frequency transformation of $\hat{\phi}\left(\mathbf{r},\mathbf{c}\middle|\omega\right) = \phi\left(\mathbf{r},\mathbf{c},t\right)e^{-i\omega t}$, where the radial frequency ω and the imaginary unit i. In the frequency domain, the linear Boltzmann-BGK equation becomes

$$\mathbf{c}\cdot\frac{\partial\hat{h}}{\partial\mathbf{r}} = -\frac{\hat{h}}{\lambda^*} + \frac{\hat{h}^{eq}}{\lambda}, \tag{4}$$

where $\hat{h}$ and $\hat{h}^{eq}$ are the frequency-based version of $\hat{h}$ and $\hat{h}^{eq}$ obtained through use of the aforementioned time-frequency transformation. The complex relaxation time $\lambda^* = \lambda/\left(1+i\omega\lambda\right)$. Importantly, after the transformation, Eq. (4) does not involve any time scale in the frequency domain.

2.2. Linearized lattice Boltzmann model in the frequency domain

A linearized LB model can be derived from Eq. (4) by discretizing its physical space and particle velocity space. For simplicity while without loss of generality, we only consider 2-D flow in this article. Furthermore, we point out that the following derivation does not involve temporal discretization. This contrasts to the conventional discretization procedure used to derive a TLB algorithm [32].

Discretization starts from the particle velocity space in Eq. (4). In this article, the popular D2Q9 discrete particle velocity space [42] is used, in which the corresponding discrete particle velocities are

$$\mathbf{c}_j = c\mathbf{e}_j = \begin{cases} (0,\ 0), & j = 0, \\ c\left(\cos\left[(j-1)\pi/2\right],\ \sin\left[(j-1)\pi/2\right]\right), & j = 1,\ 2,\ 3,\ 4, \\ \sqrt{2}c\left(\cos\left[(2j-9)\pi/4\right],\ \sin\left[(2j-9)\pi/4\right]\right), & j = 5,\ 6,\ 7,\ 8. \end{cases} \tag{5}$$

where c is the particle speed and $\mathbf{e}_j$ is the unit vector in the direction of the j^{th} discrete particle velocity. With this D2Q9-based discretization, Eq. (4) is reduced to

$$c\mathbf{e}_j \cdot \frac{\partial \hat{h}_j}{\partial \mathbf{r}} = -\frac{\hat{h}_j}{\lambda^*} + \frac{\hat{h}_j^{eq}}{\lambda}, \tag{6}$$

where $\hat{h}_j$ and $\hat{h}_j^{eq}$ are the discrete particle-velocity versions of $\hat{h}$ and $\hat{h}^{eq}$. $\hat{h}_j^{eq}$ is further expressed as

$$\hat{h}_j^{eq} = \frac{\delta\hat{\rho}}{\rho_0} + \frac{c\mathbf{e}_j \cdot \hat{\mathbf{u}}}{RT_0}. \tag{7}$$

In Eq. (7), $\delta\hat{\rho}$ and $\hat{\mathbf{u}}$ denote the fluid property perturbations in the form depending on the frequency. They are computed by [32]

$$\delta\hat{\rho} = \rho_0 \sum_j w_j \hat{h}_j, \qquad \hat{\mathbf{u}} = \sum_j w_j \hat{h}_j c\mathbf{e}_j. \tag{8}$$

The moment weights, w_j, in the D2Q9 space are specified [42]

$$w_j = \begin{cases} 4/9, & j = 0, \\ 1/9, & j = 1,\ 2,\ 3,\ 4, \\ 1/36, & j = 5,\ 6,7,\ 8. \end{cases} \tag{9}$$

Next, we extend discretization to the physical space in Eq. (6) using the finite difference schemes. In this section, two finite difference schemes are assigned to approximate the spatial gradient on the left hand of Eq. (6). For better demonstration, we take a spatial gradient with respect to x in Cartesian coordinates as an example. We approximate this gradient on a bulk node, $\left(x_m, y_n\right)$, using the second-order upwind finite difference scheme (SUS) [36], i.e.,

$$\left.\frac{\partial \hat{h}_j}{\partial x}\right|_{(x_m,y_n)} \approx e_{jx} \cdot \frac{3\hat{h}_j\left(x_m, y_n\right) - 4\hat{h}_j\left(x_{m-e_{jx}}, y_n\right) + \hat{h}_j\left(x_{m-2e_{jx}}, y_n\right)}{2\Delta x}, \tag{10a}$$

whilst the hybrid scheme (HS) [36] is applied to that on a node next to the solid boundary, i.e., $\left(\bar{x}_m, \bar{y}_n\right)$

$$\left.\frac{\partial \hat{h}_j}{\partial x}\right|_{(\bar{x}_m,\bar{y}_n)} \approx \left[\left(1-\varepsilon\right)\cdot\frac{\hat{h}_j\left(\bar{x}_{m+e_{jx}}, \bar{y}_n\right)}{2\Delta x} + \varepsilon\cdot\frac{\hat{h}_j\left(\bar{x}_m, \bar{y}_n\right)}{\Delta x} - \left(1+\varepsilon\right)\cdot\frac{\hat{h}_j\left(\bar{x}_{m-e_{jx}}, \bar{y}_n\right)}{2\Delta x}\right]\cdot e_{jx}, \tag{10b}$$

where Δx is the grid spacing and e_{jx} is the component of $\mathbf{e}_j$ in the x direction. The prefactor in Eq. (10b) $\varepsilon = 0.05$ to ensure the resulting LB simulations are stable while nearly second-order accurate. It is worth mentioning that Eqs. (10a) and (10b) are only applicable to $e_{jx} \neq 0$. For $e_{jx} = 0$, our modelling specifies directly

$$ce_{jx} \cdot \frac{\partial \hat{h}_j}{\partial x} = 0. \tag{11}$$

In summary, Eqs. (5) – (11) compose a linearized LB model in the frequency domain [32]. In contrast to the previous LB studies, this model does not include any time scale, and thus the common-used time-marching algorithm being inapplicable. To be alternative, we will develop a different LB algorithm based on the block iteration in the next section, which consists of the TDMA and JLI to solve the frequency-based linearized LB model.

3. Block iterative lattice Boltzmann algorithm

In this section, we construct a BLB algorithm for the LB model in Section 2. To be specific, we formulate two difference algebraic equations for bulk nodes and nodes next to solid boundaries, respectively. The iterative procedure of the proposed BLB algorithm is outlined at the end of this section. Before discussing the details, we point out all equations in this section are derived in Cartesian coordinates (x, y) and the symbol "^" above the frequency-dependent variables is dropped for convenience.

As discussed in Section 2, we used the SUS to approximate the spatial gradients $\partial h_j / \partial x$ and $\partial h_j / \partial y$ in Eq. (6) on bulk nods, i.e., (x_m, y_n). This finite difference discretization leads to a linear system of algebraic equations. In our block iteration for these algebraic equations, we use the TDMA to solve the unknown perturbation functions in the y direction whereas the JLI rule sweeps along the x coordinates. With these numerical arrangements, Eq. (6) is reduced to

$$\begin{aligned} &\mathrm{A}_j h_j^k(x_m, y_n) + \mathrm{I}_j h_j^k\left(x_m, y_{n-e_{jy}}\right) + \mathrm{X}_j h_j^k\left(x_m, y_{n-2e_{jy}}\right) \\ &\quad = \Gamma_j^{k-1}(x_m, y_n) - \mathrm{B}_j h_j^{k-1}\left(x_{m-e_{jx}}, y_n\right) - \mathrm{H}_j h_j^{k-1}\left(x_{m-2e_{jx}}, y_n\right), \end{aligned} \tag{12}$$

where

$$\mathrm{A}_j = \frac{3c}{2}\left(\frac{|e_{jx}|}{\Delta x} + \frac{|e_{jy}|}{\Delta y}\right) + \frac{1}{\lambda^*}, \tag{13a}$$

$$\mathrm{B}_j = -\frac{2c|e_{jx}|}{\Delta x}, \tag{13b}$$

$$\mathrm{H}_j = \frac{c|e_{jx}|}{2\Delta x}, \tag{13c}$$

$$\mathrm{I}_j = -\frac{2c|e_{jy}|}{\Delta y}, \tag{13d}$$

$$\mathrm{X}_j = \frac{c|e_{jy}|}{2\Delta y}, \tag{13e}$$

and

$$\Gamma_j^{k-1}\left(x_m, y_n\right) = \frac{h_j^{eq,\,k-1}\left(x_m, y_n\right)}{\lambda}. \tag{13f}$$

In Eqs. (12) – (13f), $|a|$ denotes the absolute value of a and the subscript k represents the k^{th} iteration step. Equation (12) indicates the calculation of $h_j^k\left(x_m, y_n\right)$ depends on its neighbours in both the x and y directions. In this equation, because of the JLI rule applied, the neighbouring perturbation functions in the x direction, together with the source term $\Gamma_j^{k-1}\left(x_m, y_n\right)$, have been specified using their results from the previous $\left(k-1\right)^{th}$ step.

On the other hand, $h_j^k\left(x_m, y_n\right)$ and its neighbours in the y direction on the left hand side of Eq. (12) are unknown yet at the current k^{th} level. These functions will be directly solved through the TDMA [36, 38]. In the TDMA framework, Eq. (12) can be rewritten for all nodes in the column at $x = x_m$ as

$$h_j^k\left(x_m, y_n\right) = P_j^k\left(x_m, y_n\right) h_j^k\left(x_m, y_{n+|e_{jy}|}\right) + Q_j^k\left(x_m, y_n\right), \tag{14}$$

where

$$P_j^k\left(x_m, y_n\right) = \frac{-\mathrm{I}_j \max\left(-e_{jy},\ 0\right)}{\mathrm{A}_j + \mathrm{I}_j \max\left(e_{jy},\ 0\right) P_j^k\left(x_m, y_{n-1}\right)}, \quad \text{with} \quad P_j^k\left(x_m, y_1\right) = 0, \tag{15a}$$

$$Q_j^k\left(x_m, y_n\right) = \frac{\Psi_j^{k-1}\left(x_m, y_n\right) - \mathrm{X}_j h_j^{k-1}\left(x_m, y_{n-2e_{jy}}\right) - \mathrm{I}_j \max\left(e_{jy},\ 0\right) Q_j^k\left(x_m, y_{n-1}\right)}{\mathrm{A}_j + \mathrm{I}_j \max\left(e_{jy},\ 0\right) P_j^k\left(x_m, y_{n-1}\right)},$$

$$\text{with} \quad Q_j^k\left(x_m, y_{n_1}\right) = h_j^k\left(x_m, \bar{y}_1\right), \tag{15b}$$

and

$$\Psi_j^{k-1}\left(x_m, y_n\right) = \Gamma_j^{k-1}\left(x_m, y_n\right) - \mathrm{B}_j h_j^{k-1}\left(x_{m-e_{jx}}, y_n\right) - \mathrm{H}_j h_j^{k-1}\left(x_{m-2e_{jx}}, y_n\right). \tag{15c}$$

$\max\left(a,\ b\right)$ represents the maximum between a and b. The coordinates $\left(x_m, y_1\right)$ denote the first bulk node in the column at $x = x_m$, and the node $\left(x_m, \overline{y}_1\right)$ is its neighbour next to a solid boundary. In the TDMA computation, we first use Eqs. (15a) – (15c) to compute P_j^k and Q_j^k at all nodes in one column, and then specify the corresponding perturbation functions, h_j^k s, in a reverse order by the recurrence formula, Eq. (14). This TDMA computation will be repeated column by column with the JLI sweeping along the x coordinates. Interested readers can refer to Ref. [36] for more details about the TDMA implementation.

The above discussion reveals one prerequisite for the calculation on bulk nodes is the neighbouring perturbation functions on nodes next to solid boundaries should be known beforehand. For these functions, Section 2 has pointed out that the HS Eq. (10b), rather than the SUS Eq. (10a), was applied to perform their finite-difference discretization. Here for a clear demonstration, we take a node close to a solid boundary parallel to the x direction as an example, i.e., $\left(\overline{x}_m, \overline{y}_n\right)$. In this case, the HS and SUS are used to approximate $\partial h_j / \partial y$ and $\partial h_j / \partial x$, respectively, which results in a difference algebraic equation as

$$\begin{aligned}\alpha_j h_j^k\left(\overline{x}_m, \overline{y}_n\right) = \varphi_j^{k-1}\left(\overline{x}_m, \overline{y}_n\right) - \beta_j h_j^{k-1}\left(\overline{x}_{m-e_{jx}}, \overline{y}_n\right) - \gamma_j h_j^{k-1}\left(\overline{x}_{m-2e_{jx}}, \overline{y}_n\right) \\ - \eta_j h_j^{k-1}\left(\overline{x}_m, \overline{y}_{n-e_{jy}}\right) - \chi_j h_j^{k-1}\left(\overline{x}_m, \overline{y}_{n+e_{jy}}\right),\end{aligned} \tag{16}$$

where

$$\alpha_j = \frac{3c\left|e_{jx}\right|}{2\Delta x} + \varepsilon\frac{c\left|e_{jy}\right|}{\Delta y} + \frac{1}{\lambda^*}, \tag{17a}$$

$$\beta_j = -\frac{2c\left|e_{jx}\right|}{\Delta x}, \tag{17b}$$

$$\gamma_j = \frac{c\left|e_{jx}\right|}{2\Delta x}, \tag{17c}$$

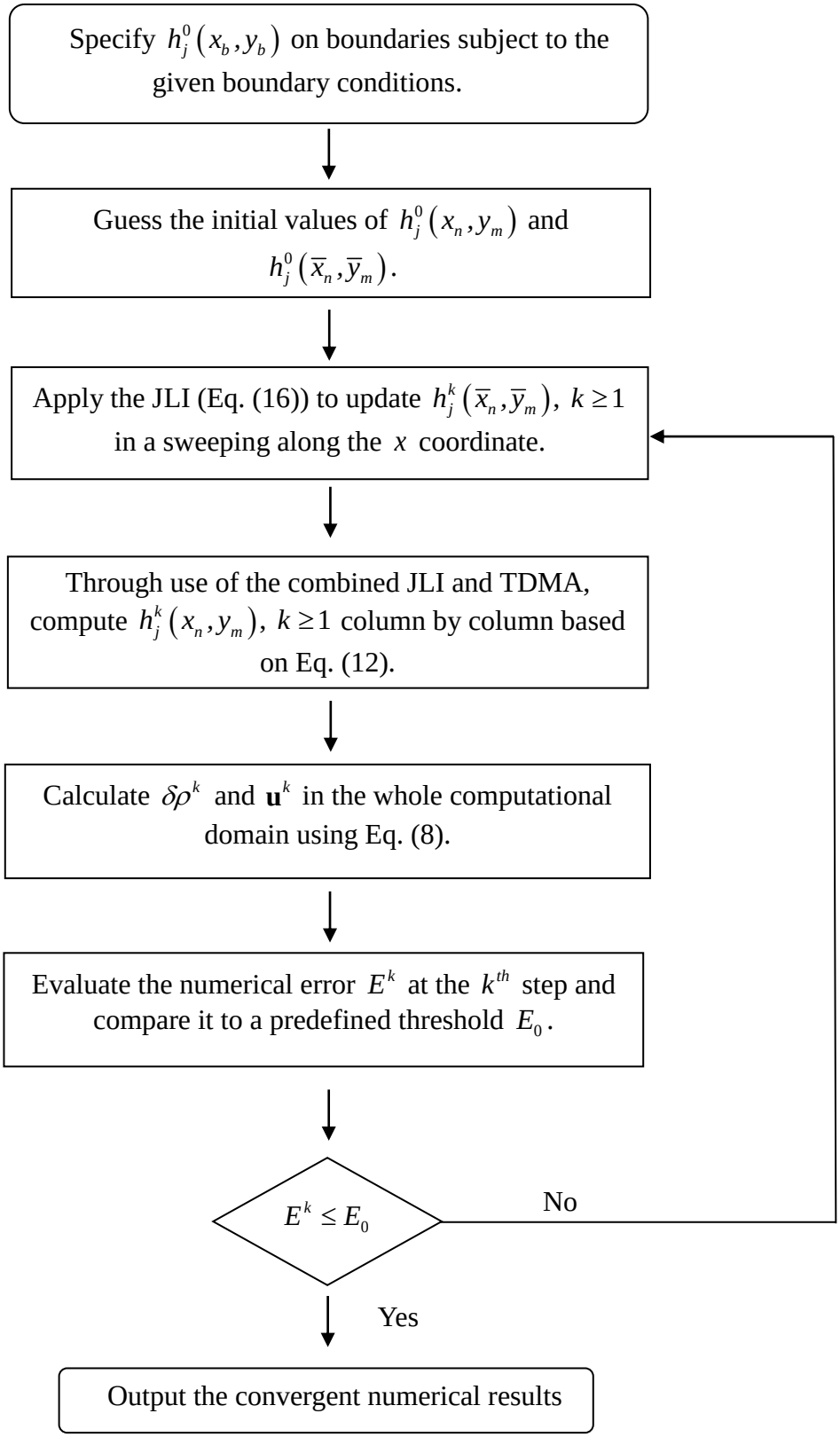
$$\eta_j = -\left(1+\varepsilon\right)\cdot\frac{c\left|e_{jy}\right|}{2\Delta y}, \tag{17d}$$

$$\chi_j = (1-\varepsilon)\cdot\frac{c|e_{jy}|}{2\Delta y}, \tag{17e}$$

and

$$\varphi_j^{k-1}(\bar{x}_m,\bar{y}_n) = \frac{h_j^{k-1,\,eq}(\bar{x}_m,\bar{y}_n)}{\lambda}\ . \tag{17f}$$

In contrasts to Eq. (12), Eq. (16) uses the JLI to specify all neighbouring functions in both the x and y directions. The reason we adopt this adjustment is just for simplifying the corresponding numerical implementation. In so doing, the discrete perturbation function $h_j^k(\bar{x}_m,\bar{y}_n)$ is fully determined by its neighbours specified at the previous $(k-1)^{th}$ step. In summary, Eqs. (5), (7) – (9), together with Eqs. (12) – (13f) and (16) – (17f), consist of our BLB algorithm based on the TDMA and JLI. The numerical procedure of this BLB algorithm is illustrated by


Specify $h_j^0(x_b, y_b)$ on boundaries subject to the given boundary conditions.
Guess the initial values of $h_j^0(x_n, y_m)$ and $h_j^0(\overline{x}_n, \overline{y}_m)$.
Apply the JLI (Eq. (16)) to update $h_j^k(\overline{x}_n, \overline{y}_m)$, $k \geq 1$ in a sweeping along the x coordinate.
Through use of the combined JLI and TDMA, compute $h_j^k(x_n, y_m)$, $k \geq 1$ column by column based on Eq. (12).
Calculate $\delta\rho^k$ and $\mathbf{u}^k$ in the whole computational domain using Eq. (8).
Evaluate the numerical error E^k at the k^{th} step and compare it to a predefined threshold E_0.
$E^k \leq E_0$
No
Yes
Output the convergent numerical results

For simplicity, we will use BLB (JLI, SUS+HS) to represent the LB algorithm developed in this section, and evaluate its numerical accuracy and efficiency through simulating two linear oscillatory flow problems in the following discussion.

4. Numerical simulation and discussion

We apply the BLB (JLI, SUS+HS) algorithm proposed in Section 3 to simulate linear oscillatory flow in this section. We first validate its numerical accuracy by simulating the 1-D oscillatory Couette flow (flat solid boundaries) and 2-D flow around an oscillating circular cylinder (curved solid boundaries). This section also includes a comparison of computational efficiency among this BLB algorithm, its modified versions and the TLB algorithm [32].

4.1. One dimensional oscillatory Couette flow

We first validate the BLB (JLI, SUS+HS) algorithm by simulating the 1-D oscillatory Couette flow in the frequency domain. This flow is driven by two parallel plates separated by a distance L. The top plate is stationary while the bottom plate oscillates in its own plane with a velocity $u_{wall} = u_0 e^{i\omega t}$, where u_0 is a constant velocity and ω is the radial frequency, see Fig. 1. To characterize the corresponding flow dynamics, we introduce the Stokes number $S = \rho_0 L\omega / \mu$, with ρ_0 and μ being the reference density and viscosity of the fluid confined between the plates.

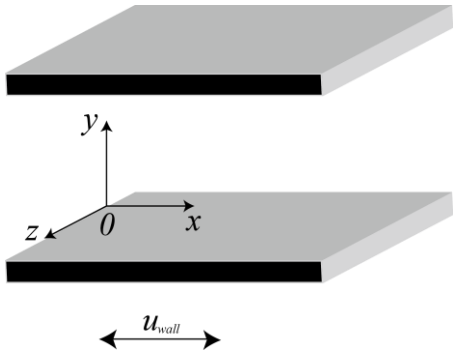


Fig. 1. Schematic of geometry of the oscillatory Couette flow. Origin of the coordinates system is on the bottom plate.

In our simulation, we applied periodic boundary conditions to the two ends in the x direction, and used the non-equilibrium extrapolation method [43] to prescribe the perturbation functions on the solid plates subject to no-slip boundary conditions. Moreover, we specified the sound speed $c_s = 100 \cdot u_{wall}$, and the relaxation time $\lambda = 1/\left(\rho_0 c_s^2 S\right)$. To obtain dimensionless numerical results, we chose $L = 1$, $\rho_0 = 1$ and $u_0 = 1$.

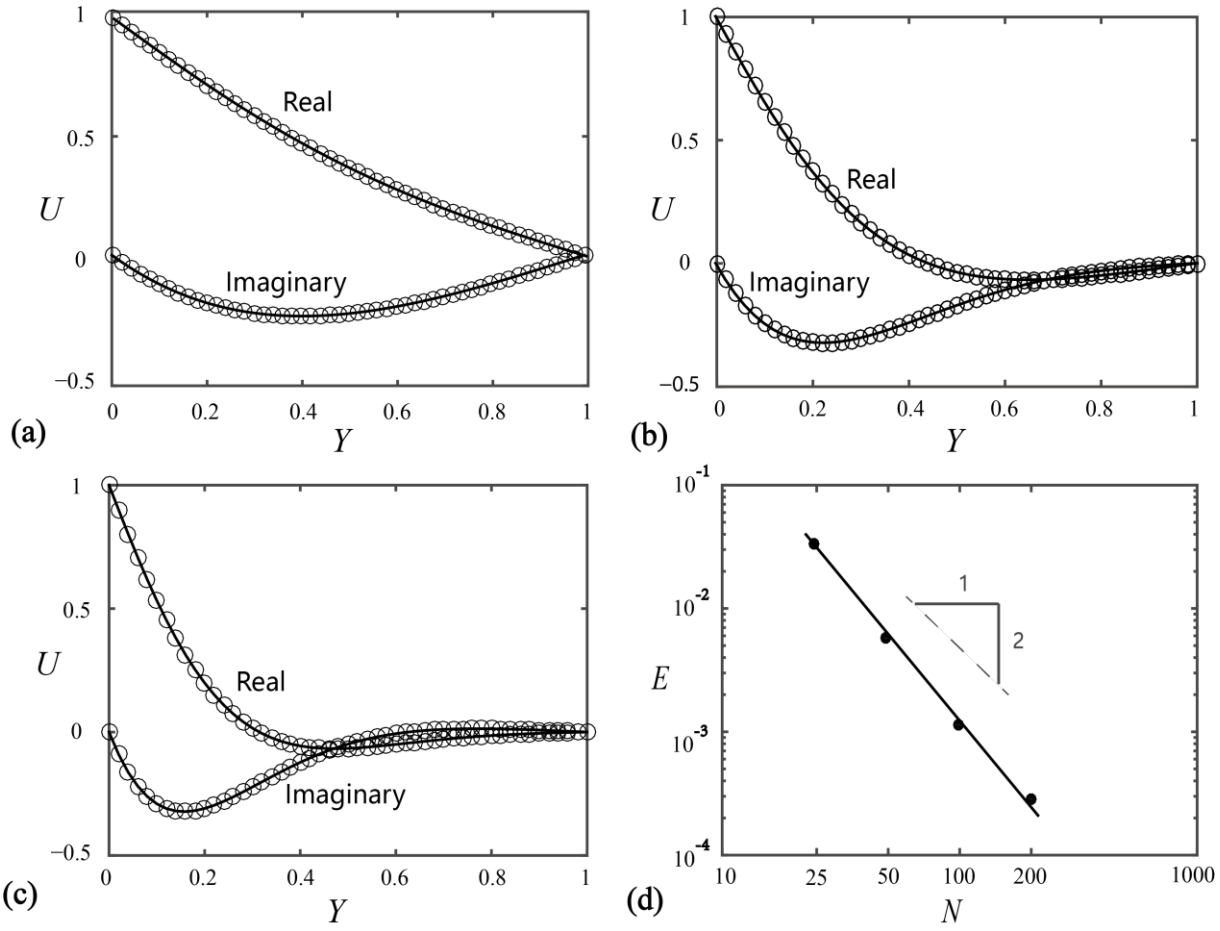


Fig. 2. Dimensionless streamwise velocities for the oscillatory Couette flows. Open circles: BLB (JLI, SUS+HS) results; Solid lines: analytical solution [32]. (a). $S = 5$; (b). $S = 25$; (c). $S = 50$; (d). Errors of the LB simulations in different grids when $S = 25$.

We performed the BLB (JLI, SUS+HS) simulations on $N \times N = 100 \times 100$ grids. Fig. 2 shows the dimensionless streamwise velocities (i.e., component in the x direction) for the flows with $S = 5$, 25 and 50, respectively. In Fig. 2, all velocities include both the real and imaginary parts as these variables are complex-valued in the frequency domain. Interestingly, we see that in the cases with a small Stokes number, the fluid velocities across the channel

have been significantly influenced by oscillatory movement of the bottom plate, see Figs. 2(a) and 2(b). However, the impact of such a movement becomes rather weak on flow far away from the plate when the Stokes number grows. This has been clearly exhibited in Fig. 2(c), where the fluid beyond $Y = 0.6$ is almost unperturbed by the bottom plate's oscillation. The numerical results in Figs. 2(a) – 2(c) are well agreed with the analytical solutions and the results given by the TLB simulation [32].

In this numerical case, we also conducted grid-convergence tests of the BLB (JLI, SUS+HS) algorithm. The oscillatory Couette flow with S =25 was chosen as a test case and we simulated it using the algorithm on four different grids, i.e., 25×25, 50×50, 100×100 and 200×200. In each grid, we computed the root-mean-square error to quantify the global accuracy of the LB simulation, i.e.,

$$E = \sqrt{\frac{1}{N^2}\sum_{x_m, y_n}\left(U\left(x_m, y_n\right) - U_0\right)^2}\,, \tag{18}$$

where $U\left(x_m, y_n\right)$ and U_0 represent the velocity obtained by the LB simulation at the node $\left(x_m, y_n\right)$ and the corresponding analytical solution, respectively. The symbol "Σ" means a sum over all nodes in both the x and y directions. Figure 2(d) shows the obtained errors E in different grids. We see a linear decrease of this error with the increasing grid number in the double logarithm coordinates. Importantly, the slop of this $E-N$ line in Fig. 2(d) is about 2.3. This index evidences that the BLB (JLI, SUS+HS) algorithm proposed in Section 3 is second-order accurate for the oscillatory Couette flow.

4.2. Two dimensional flow around an oscillating circular cylinder

Next, we apply the BLB (JLI, SUS+HS) algorithm to simulate a 2-D flow generated by an oscillating circular cylinder in an unbounded fluid [32], see Fig. 3.

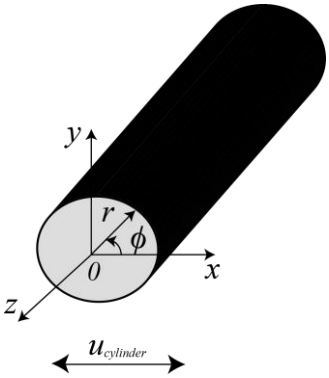


Fig. 3. Schematic of geometry of the flow around an oscillating circular cylinder. Origin of the coordinates system is at the centre of the cylinder.

In this problem, a circular cylinder with a radius a is immersed in a fluid with a density ρ_0 and a viscosity μ. It oscillates at a horizontal velocity $u_{cylinder} = u_0 e^{i\omega t}$ parallel to the x direction [44]. In the corresponding numerical settings, we specified $a = 1$, $\rho_0 = 1$ and $u_0 = 1$ to nondimensionalize the results and defined the Stokes number as $S = \rho_0 a^2 \omega / \mu$. The computational domain was set as a square with a side length $L = 70a$. Our numerical tests has validated that this choice is large enough to ensure the fluids far away from the oscillating cylinder are unperturbed. In this problem, we realized the non-equilibrium extrapolation scheme [43] was inapplicable in Cartesian coordinates as the treatments set in this scheme for curved boundaries in rectilinear grids (i.e., lattices) were formulated under the TLB framework. To circumvent this barrier, we transformed our BLB (JLI, HS+SUS) simulations to the polar coordinates, which enable the first layer of grid nodes in the radial direction to be exactly allocated on the cylinder's surface. With this simple mathematical manipulation, the non-equilibrium extrapolation scheme [43] becomes workable again in our simulation. Importantly, the linear LB equation in Section 2 is almost unchanged in the new coordinates except that the involved spatial gradients turn to

$$\mathbf{c}_j \cdot \frac{\partial h_j}{\partial \mathbf{r}} = c_{jr} \cdot \frac{\partial h_j}{\partial r} + \frac{c_{j\phi}}{r} \cdot \frac{\partial h_j}{\partial \phi}, \tag{19}$$

where r and ϕ denotes the radial and azimuth coordinates of the polar coordinate system, and c_{jr} and $c_{j\phi}$ are the respective particle velocity components. For Eq. (19), the SUS and

HS were carried out for its physical-space discretization on different nodes. The resulting difference approximations, taking the r direction as example, are

$$\left.\frac{\partial h_j}{\partial r}\right|_{(r_m,\phi_n)} \approx e_{jr}\cdot\frac{3h_j\left(r_m,\phi_n\right)-4h_j\left(r_{m-e_{jr}},\phi_n\right)+h_j\left(r_{m-2e_{jr}},\phi_n\right)}{2\Delta r}, \tag{20a}$$

for bulk nodes while on nodes next to solid boundaries, we have

$$\left.\frac{\partial h_j}{\partial r}\right|_{(\bar{r}_m,\bar{\phi}_n)} \approx \left[\left(1-\varepsilon\right)\cdot\frac{h_j\left(\bar{r}_{m+e_{jr}},\bar{\phi}_n\right)}{2\Delta r}+\varepsilon\cdot\frac{h_j\left(\bar{r}_m,\bar{\phi}_n\right)}{\Delta r}-\left(1+\varepsilon\right)\cdot\frac{h_j\left(\bar{r}_{m-e_{jr}},\bar{\phi}_n\right)}{2\Delta r}\right]\cdot e_{jr}\,. \tag{20b}$$

e_{jr} is the component of $\mathbf{e}_j$ in the r direction.

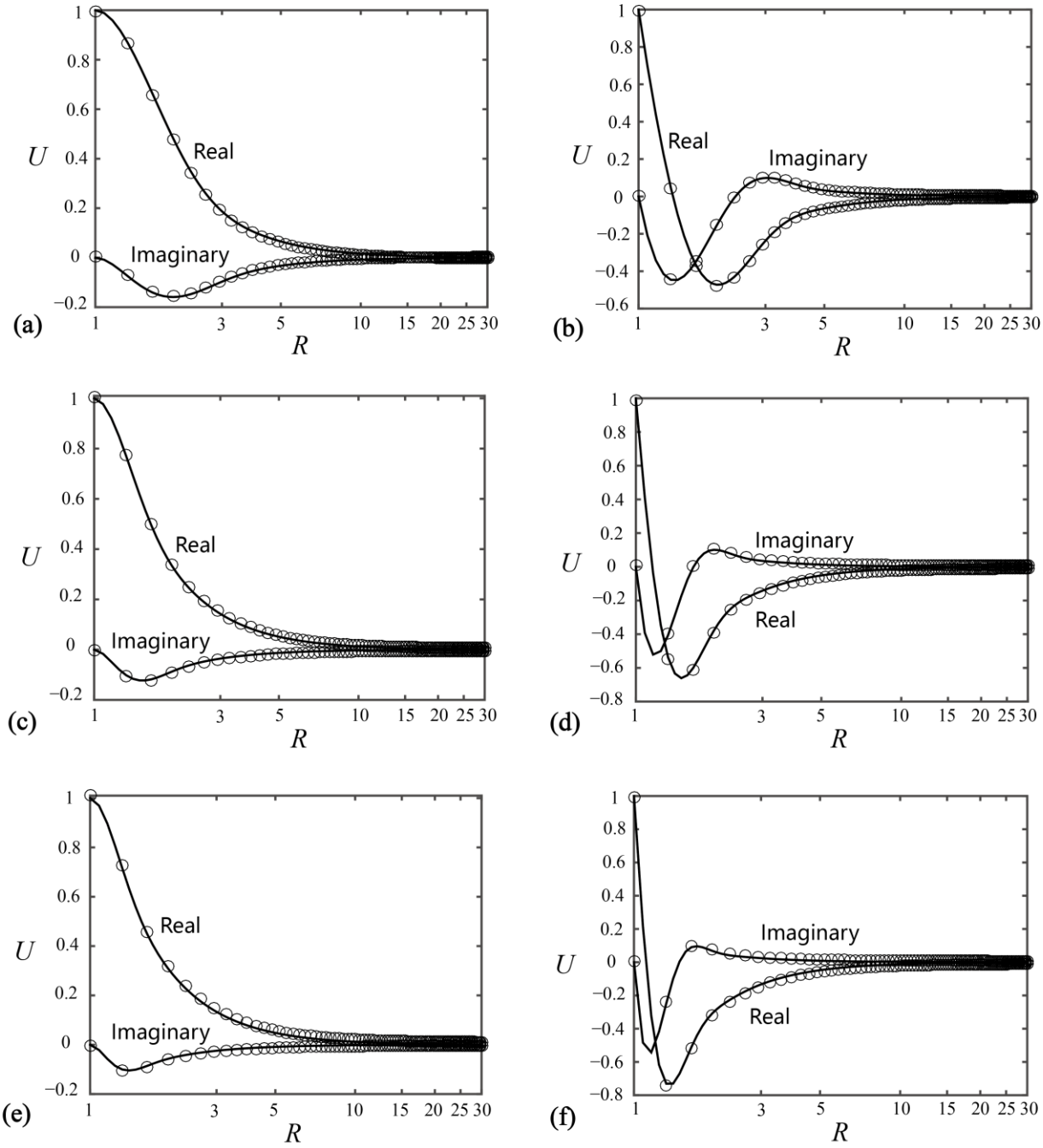


Fig. 4. Dimensionless streamwise velocities for the flow around an oscillating circular cylinder. Open circles: BLB (JLI, HS+SUS) results; Solid lines: analytical solution [44, 45]. (a). $S=5,\ \phi=0$; (b). $S=5,\ \phi=\pi/2$; (c). $S=25,\ \phi=0$; (d). $S=25,\ \phi=\pi/2$; (e). $S=50,\ \phi=0$; (f). $S=50,\ \phi=\pi/2$.

Figure 4 shows the streamwise velocities, $U=u/u_0$, at $\phi=0$ and $\pi/2$ obtained by the BLB (JLI, HS+SUS) simulations on 360×360 grids with $S=5$, 25 and 50. For a clear

illustration, Fig. 4 only exhibits the velocities between $R=1$ and $R=30$ (where $R \equiv r/a$) as the fluids in the region $R>30$ are almost unperturbed to the cylinder's oscillation. In all cases in Fig. 4, the velocity profiles display significant variations in a boundary layer near the cylinder's surface, and then decay to a unperturbed state, i.e., $U=0$, with the increasing R. Importantly, we observe that the decay rates of velocities vary with different Stokes numbers: the velocities at both $\phi=0$ and $\pi/2$ corresponding to a larger Stokes number always decay more quickly than those with a smaller S, see Figs. 4(e) and 4(f) in comparison to Figs. 4(a) and 4(b). Theoretically, the Stokes number is a squared ratio of the radius of cylinder to the viscous penetration depth; the latter is a length scale characterising the velocity decay from solid boundaries. Therefore, a larger Stokes number implies a shorter viscous penetration depth for a given cylinder's radius. This explains the phenomena in Figs. 4(e) and 4(f) that the velocity profiles have a faster decay rate than those in Figs. 4(a) and 4(b). In addition, we compared the numerical results with the available analytical solutions [44, 45] for each case in Fig. 4. Again, good agreements between the numerical and analytical results are found.

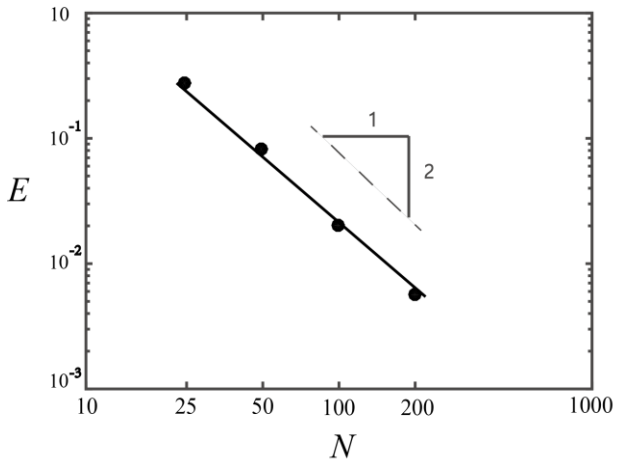


Fig. 5. Comparison of the BLB (JLI, HS+SUS) errors in different grids when $S=25,\ \phi=0$.

Our discussion in this simulation also includes grid-dependence tests of the BLB (JLI, HS+SUS) simulation to quantify its accuracy. We chose 25×25, 50×50, 100×100 and 200×200 grids to repeat the numerical simulations with $S=25,\ \phi=0$. Figure 5 displays the errors E defined by Eq. (18) in these four grids. As expected, such an error gradually decreases when denser grids are employed, and the corresponding decreasing slop is 1.9 in the log-log plot in Fig. 5. The results in Fig. 5, together with those in Fig. 2, confirm that the

BLB (JLI, HS+SUS) algorithm is nearly second-order accurate for linear oscillatory flow, regardless of solid boundaries being flat or curved.

4.3. Comparison of computational efficiency

In subsections 4.1 and 4.2, we examined numerical accuracy of the BLB (JLI, HS+SUS) algorithm. The results demonstrate its high accuracy for simulating linear oscillatory flow. Examination of its computational efficiency will be conducted in this subsection, especially in comparison to that of the TLB algorithm constructed with a virtual time scale [32]. We point out that all simulations in this subsection were performed on the same computer, i.e., Dell Precision 7910 CTO.

For simplicity while without loss of generality, we took the 1-D oscillatory Couette flow with $S=10$ as a test case and recorded the root-mean-square errors E at every 10000 iterative (time) steps for both the BLB (JLI, HS+SUS) and TLB simulations. The decay of the error is used as a measure to quantify computational efficiency of these LB algorithms. Figure 6 exhibits the root-mean-square errors E during the iterative course in the BLB (JLI, HS+SUS) simulation (i.e., Curve A) and the time evolution and TLB simulation (i.e. Curve F) on the 100×100 uniform grids. Interestingly, we see that the TLB algorithm displays a much faster decay rate (i.e., higher efficiency) than the BLB (JLI, HS+SUS) algorithm. To be specific, the TLB simulation only spent 12 minutes in reducing its error to $E=1.7\times10^{-3}$, whereas the time for the BLB (JLI, HS+SUS) algorithm reaching the same error level was 100 minutes. This comparison of E illustrates that as far as computational efficiency is concerned, the BLB (JLI, HS+SUS) algorithm is not a better numerical solver for linear oscillatory flow. We attribute this inefficiency to the used JLI and motivate a series of modifications of the proposed BLB algorithm through use of more efficient iterative approaches and simpler finite difference schemes. To achieve improved computational efficiency, use of non-uniform grids was also attempted.

Figure 6 shows the error decay of the four BLB algorithms after our modification. Curve B corresponds to a BLB algorithm still based on JLI but using the HS to approximate all spatial gradients. Curve C is obtained by the similar algorithm as that for Curve B, but replacing the JLI by the SLI (see Appendix A for the SLI details). The modified BLB algorithms for Curve D and E are two versions modified by a combined iterative rule based on SLI, ADI and OR on 100×100 uniform (i.e., see Appendix B and C for the ADI and OR

details) and 20×100 non-uniform grids, respectively. For convenience, these four modified BLB algorithms are simply represented as BLB (JLI, HS), BLB (SLI, HS), BLB (SLI+ADI+OR, HS) and BLB _N (SLI+ADI+OR, HS) in the next discussion.

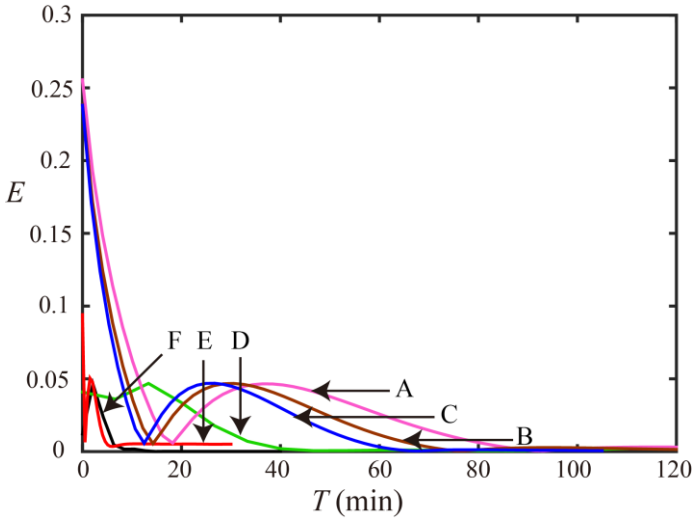


Fig. 6. Error decay of different LB algorithms. A: BLB (JLI, SUS+HS); B: BLB (JLI, HS); C: BLB (SLI, HS); D. BLB (SLI+ADI+OR, HS); E: BLB_N (SLI+ADI+OR, HS); F: TLB [32].

In Fig. 6, we compared Curve A with Curve B, and found the use of the HS throughout in the BLB simulation did not bring about instability, but an improvement in computational efficiency. As shown by Curve B, the BLB (JLI, HS) algorithm only spent 73.2 minutes to achieve $E = 1.7\times10^{-3}$. Such an efficiency improvement is enhanced in the BLB (SLI, HS) simulation. Curve C shows that the change from the JLI to SLI saves about 13.46% in computational time as compared to Curve B. Meanwhile, however, we also note that the modifications resulted from the HS and SLI are insufficient – the error-decay rates in Curve B and Curve C are still far behind that in Curve F (TLB algorithm). This motivates development of the BLB (SLI+ADI+OR, HS) algorithm and its non-uniform grid version, i.e., BLB_N (SLI+ADI+OR, HS) algorithm. In these two algorithms, CTDMA (see Appendix D for the details) was introduced as the replacement of TDMA for direct computation of the discrete perturbation functions in rows. These functions are subject to periodic boundary conditions in the oscillatory Couette flow. The error changes of the BLB (SLI+ADI+OR, HS) and BLB_N (SLI+ADI+OR, HS) simulations are exhibited in Fig. 6, see Curve D and Curve E. Impressively, unlike those shown in Curve A, Curve B and Curve C, these two simulations initiated their simulations with very small errors, and such errors decayed

quickly with the progress of computation. In Fig. 6, Curve E is very close to Curve F, indicating the BLB_N (SLI+ADI+OR, HS) algorithm converged at a comparable rate to that of the TLB algorithm.

In this subsection, our numerical simulations show the conventional TLB algorithm exhibits quite good efficiency in comparison to the BLB (JLI, SUS+HS) algorithm and even some modified BLB algorithms. Only the BLB_N (SLI+ADI+OR, HS) algorithm in our test has achieved a close convergence rate to the TLB algorithm. For the proposed BLB simulation, we understand that an appropriate algorithm design is of critical importance for achieving high computational efficiency. Meanwhile, the BLB algorithm also manifests distinct numerical compatibility with a large variety of CFD techniques and flexible applicability in both uniform and non-uniform grids.

5. Conclusion

In this article, we propose a block iterative algorithm to solve the purely frequency-dependent linear LB model for simulating linear oscillatory flow. The primary feature of this BLB algorithm, in contrast to the conventional TLB algorithm, is that it completely excludes any time scale, and computes the perturbation functions directly in the frequency domain without mimicking a false evolution in virtual time.

Numerical accuracy of the BLB algorithm proposed in this article was validated by simulating two classical flow problems: the oscillatory Couette flow and flow around an oscillating circular cylinder with different Stokes numbers. All results are of near second-order accuracy and well agreed with the available analytical solutions. We also studied computational efficiency of the BLB algorithm, in particular in comparison to the conventional TLB algorithm based on the virtual time. A set of modified BLB algorithms were also proposed and involved in this efficiency comparison. Our simulations reveal that different BLB algorithms have rather various computational efficiency; only well-designed BLB version can achieve good efficiency as compared with the TLB algorithm. On the other hand, our studies also reveals that flexibility and richness in the construction of BLB algorithms, which is in sharp contrast to its TLB counterpart. The BLB framework can readily develop various versions through use of different CFD numerical techniques and grids. This is of value for simulating flow processes in practical M/NEMS, where complex structures and varying operating conditions are involved. We will investigate such possible applications of the BLB algorithm in M/NEMS in our future work.

Acknowledgements

The authors would like to acknowledge support from Zhejiang Provincial Natural Science Foundation of China under Grant No. LY16E060001, Ningbo Science and Technology Bureau Technology Innovation Team under Grant No. 2016B10010 and Ningbo International Cooperation Program under Grant No. 2015D10018. Y.S. thanks John E. Sader for many interesting and stimulating discussions. H.K. acknowledges partial support by International Doctoral Innovation Centre at the University of Nottingham Ningbo China.

Appendix A. Seidel line iteration

The difference algebraic equations in Section 3, Eqs. (12) and (16), are constructed based on the JLI, which suffer from low computational efficiency in comparison to the TLB algorithm. As a solution, we reformulated these equations through use of the SLI.

The major difference of a SLI from a JLI is the former makes use of the latest perturbation functions on neighbouring nodes for calculation. These latest neighbouring results are specified at either the $(k-1)^{th}$ or k^{th} step, depending on the sweeping direction in which the iteration proceeds. In this appendix, we introduce the SLI-related details used in the BLB (SLI, HS) algorithm in Section 4.3, where spatial gradients on all nodes are approximated by the HS. Its difference equations after the finite difference discretization are

$$\begin{aligned}&\alpha_j^s h_j^k\left(x_m,y_n\right)+\eta_j h_j^k\left(x_m,y_{n-e_{jy}}\right)+\chi_j h_j^k\left(x_m,y_{n+e_{jy}}\right)\\&\quad=\varphi_j^{k-1}\left(x_m,y_n\right)-\beta_j^s h_j^k\left(x_{m-e_{jx}},y_n\right)-\gamma_j^s h_j^{k-1}\left(x_{m+e_{jx}},y_n\right)\end{aligned}\quad \text{for } e_{jx}>0,\ \text{(A1)}$$

while

$$\begin{aligned}&\alpha_j^s h_j^k\left(x_m,y_n\right)+\eta_j h_j^k\left(x_m,y_{n-e_{jy}}\right)+\chi_j h_j^k\left(x_m,y_{n+e_{jy}}\right)\\&\quad=\varphi_j^{k-1}\left(x_m,y_n\right)-\beta_j^s h_j^{k-1}\left(x_{m-e_{jx}},y_n\right)-\gamma_j^s h_j^k\left(x_{m+e_{jx}},y_n\right)\end{aligned}\quad \text{for } e_{jx}<0,\ \text{(A2)}$$

where

$$\alpha_j^s = \varepsilon c\left(\frac{|e_{jy}|}{\Delta y}+\frac{|e_{jx}|}{\Delta x}\right)+\frac{1}{\lambda^*}, \tag{A3}$$

$$\beta_j^s = -(1+\varepsilon)\cdot\frac{c|e_{jx}|}{2\Delta x}, \tag{A4}$$

$$\gamma_j^s = (1-\varepsilon)\cdot\frac{c|e_{jx}|}{2\Delta x}, \tag{A5}$$

and η_j, χ_j and $\varphi_j^{k-1}(x_m, y_m)$ are given by Eqs. (17d) – (17f), respectively.

In the BLB (SLI, HS) algorithm, we performed a SLI along the x direction sweeping from x_0 to x_N, where $x_0 < x_N$. Therefore, the three terms on the right hand side of Eqs. (A1) and (A2) have been specified. In simulation, we applied the TDMA to solve Eqs. (A1) and (A2) for $h_j^k(x_m, y_n)$, $h_j^k\left(x_m, y_{n-e_{jy}}\right)$, $h_j^k\left(x_m, y_{n+e_{jy}}\right)$ and other perturbation functions in the same column at $x = x_m$, and then repeated this direct-solving procedure column by column until the SLI had swept the entire computational domain at the k^{th} iteration. Generally, our BLB (SLI, HS) algorithm will terminate its computation once a predefined convergence criterion is met.

Appendix B. Alternative direction iteration

On top of the SLI, the ADI is another advanced iterative method employed for modifying the BLB algorithm in Section 4.3. An ADI process designs an iteration consisting of two successive sweeping– one by columns (along the x direction) and the other by rows (along the y direction). In this appendix, we discuss the ADI details pertinent to the BLB (SLI+ADI+OR, HS) algorithm.

In the BLB (SLI+ADI+OR, HS) algorithm, a SLI was first conducted along the positive x direction. The difference algebraic equations in this half are

$$\begin{aligned}\alpha_j^s h_j^{k-1/2}(x_m, y_n) + \eta_j h_j^{k-1/2}\left(x_m, y_{n-e_{jy}}\right) + \chi_j h_j^{k-1/2}\left(x_m, y_{n+e_{jy}}\right) \\ = \varphi_j^{k-1}(x_m, y_n) - \beta_j^s h_j^{k-1/2}\left(x_{m-e_{jx}}, y_n\right) - \gamma_j^s h_j^{k-1}\left(x_{m+e_{jx}}, y_n\right)\end{aligned} \quad \text{for } e_{jx} > 0, \tag{B1}$$

while

$$\alpha_j^s h_j^{k-1/2}\left(x_m,y_n\right)+\eta_j h_j^{k-1/2}\left(x_m,y_{n-e_{jy}}\right)+\chi_j h_j^{k-1/2}\left(x_m,y_{n+e_{jy}}\right)$$
$$=\varphi_j^{k-1}\left(x_m,y_n\right)-\beta_j^s h_j^{k-1}\left(x_{m-e_{jx}},y_n\right)-\gamma_j^s h_j^{k-1/2}\left(x_{m+e_{jx}},y_n\right) \quad \text{for } e_{jx}<0, \text{ (B2)}$$

where the coefficients are defined the same as those in Appendix A. Actually, Eqs. (B1) and (B2) are almost the same as Eqs. (A1) and (A2) except that the superscript "k" has been replaced by "$\left(k-1/2\right)$" to denote the column sweeping as the first half in one ADI process. Equations (B1) and (B2) were then solved directly using the TDMA following the same procedure as Appendix A.

Next, the perturbation functions updated by the column sweeping were used as inputs for the row sweeping along the positive y direction, the second half. The corresponding difference algebraic equations are

$$\alpha_j^s h_j^k\left(x_m,y_n\right)+\beta_j^s h_j^k\left(x_{m-e_{jx}},y_n\right)+\gamma_j^s h_j^k\left(x_{m+e_{jx}},y_n\right)$$
$$=\varphi_j^{k-1/2}\left(x_m,y_n\right)-\eta_j h_j^k\left(x_m,y_{n-e_{jy}}\right)-\chi_j h_j^{k-1/2}\left(x_m,y_{n+e_{jy}}\right) \quad \text{for } e_{jy}>0, \text{ (B3)}$$

while

$$\alpha_j^s h_j^k\left(x_m,y_n\right)+\beta_j^s h_j^k\left(x_{m-e_{jx}},y_n\right)+\gamma_j^s h_j^k\left(x_{m+e_{jx}},y_n\right)$$
$$=\varphi_j^{k-1/2}\left(x_m,y_n\right)-\eta_j h_j^{k-1/2}\left(x_m,y_{n-e_{jy}}\right)-\chi_j h_j^k\left(x_m,y_{n+e_{jy}}\right) \quad \text{for } e_{jy}<0. \text{ (B4)}$$

Again, the coefficients in Eqs. (B3) and (B4) are the same as those in those in Appendix A and the terms in the right hand side are all known. We solved Eqs. (B3) and (B4) for the perturbation functions in the row at $y=y_n$ by the CTDMA (see Appendix D) as periodic boundary conditions were imposed in the x direction in the oscillatory Couette flow.

With a column sweeping (Eqs. (B1) and (B2)) and a row sweeping (Eqs. (B3) and (B4)), an ADI process completed updating all perturbation functions in the domain at the k^{th} step. In the BLB (SLI+ADI+OR, HS) algorithm, we repeated this ADI until convergence was reached.

Appendix C. Over relaxation scheme

The over relaxation (OR) scheme is a simple while efficient means to improve computational efficiency. In Section 4.3, we applied OR in both the BLB (SLI+ADI+OR, HS) and BLB_N (SLI+ADI+OR, HS) algorithms.

Consider a perturbation function $\tilde{h}_j^k\left(x_m, y_n\right)$, which is just calculated after the TDMA or CTDMA at the k^{th} step. In the OR framework, the true value of this function will be modified by

$$h_j^k\left(x_m, y_n\right) = \left(1-\varepsilon_o\right) h_j^{k-1}\left(x_m, y_n\right) + \varepsilon_o \tilde{h}_j^k\left(x_m, y_n\right), \tag{C1}$$

where ε_0 is a numerical weight, and $h_j^{k-1}\left(x_m, y_n\right)$ is the value of this perturbation function obtained by the OR at the previous $\left(k-1\right)^{th}$ step. In Section 4.3, the BLB (SLI+ADI+OR, HS) algorithm chose $\varepsilon_0 = 1.9$ in its OR adjustment while $\varepsilon_0 = 1.5$ was used in the non-uniform grid version, i.e., the BLB_N (SLI+ADI+OR, HS) algorithm.

Appendix D. Cyclic tri-diagonal matrix algorithm

The CTDMA is a variant of the TDMA for a problem with periodic boundary conditions. As discussed in Section 4.3 and Appendix B, this is the case when we solve the perturbation functions in one row for the oscillatory Couette flow. Since we only adopted the CTDMA in the row sweeping in the ADI, we take Eqs. (B3) and (B4) on a row $\left(x = x_1, x_2, ..., x_m, ..., x_{N-1}, x_N \text{ and } y = y_n\right)$ as an example to elaborate its details in this appendix. In the CTDMA framework, Eqs. (B3) and (B4) are rewritten as

$$h_j^k\left(x_m, y_n\right) = p_j^k\left(x_m, y_n\right) h_j^k\left(x_{m+\left|e_{jx}\right|}, y_n\right) + o_j^k\left(x_m, y_n\right) h_j^k\left(x_N, y_n\right) + q_j^k\left(x_m, y_n\right), \tag{D1}$$

where

$$p_j^k\left(x_m, y_n\right) = \frac{\sigma_j}{\alpha_j^s + \delta_j p_j^k\left(x_{m-1}, y_n\right)}, \qquad \text{with} \quad p_j^k\left(x_1, y_n\right) = \sigma_j / \alpha_j^s, \tag{D2}$$

$$o_j^k\left(x_m,y_n\right)=\frac{-\delta_j o_j^k\left(x_m,y_n\right)}{\alpha_j^s+\delta_j p_j^k\left(x_{m-1},y_n\right)}\,,\qquad \text{with } o_j^k\left(x_1,y_n\right)=-\delta_j/\alpha_j^s\,, \tag{D3}$$

$$q_j^k\left(x_m,y_n\right)=\frac{\theta_j^{k-1}\left(x_m,y_n\right)-\delta_j q_j^k\left(x_{m-1},y_n\right)}{\alpha_j^s+\delta_j p_j^k\left(x_{m-1},y_n\right)},\qquad \text{with } q_j^k\left(x_1,y_n\right)=\theta_j^{k-1}\left(x_1,y_n\right)/\alpha_j^s\,. \tag{D4}$$

In Eqs. (D2) – (D4), $\sigma_j=\max\left(-e_{jx},0\right)\frac{c\left|e_{jx}\right|}{\Delta x}-\gamma_j^s$, $\delta_j=\beta_j^s+\max\left(-e_{jx},0\right)\frac{c\left|e_{jx}\right|}{\Delta x}$ and

$$\theta_j^{k-1}\left(x_m,y_{\mathrm{n}}\right)=\varphi_j^{k-1/2}\left(x_m,y_n\right)-\eta_j h_j^k\left(x_m,y_{n-e_{jy}}\right)-\chi_j h_j^{k-1/2}\left(x_m,y_{n+e_{jy}}\right),\quad \text{for } e_{jy}>0, \tag{D5}$$

while

$$\theta_j^{k-1}\left(x_m,y_{\mathrm{n}}\right)=\varphi_j^{k-1/2}\left(x_m,y_n\right)-\eta_j h_j^{k-1/2}\left(x_m,y_{n-e_{jy}}\right)-\chi_j h_j^k\left(x_m,y_{n+e_{jy}}\right),\quad \text{for } e_{jy}<0. \tag{D6}$$

We point out that different from Eq. (14) in the TDMA, Eq. (D1) includes $h_j^k\left(x_N,y_n\right)$ when calculating $h_j^k\left(x_m,y_n\right)$. This function should be first specified by

$$h_j^k\left(x_N,y_n\right)=\frac{r_j^k\left(x_{N-1},y_n\right)+\left(q_j^k\left(x_{N-1},y_n\right)-\delta_j\right)q_j^k\left(x_{N-1},y_n\right)}{p_j^k\left(x_{N-1},y_n\right)-\left(q_j^k\left(x_{N-1},y_n\right)-\delta_j\right)\left(p_j^k\left(x_{N-1},y_n\right)+o_j^k\left(x_{N-1},y_n\right)\right)}. \tag{D7}$$

Equation (D7) includes three new coefficients, p_j^k, q_j^k and r_j^k, and they are computed by a set of back-substitution equations:

$$r_j^k\left(x_m,y_n\right)=r_j^k\left(x_{m-1},y_n\right)-q_j^k\left(x_{m-1},y_n\right)o_j^k\left(x_{m-1},y_n\right),\qquad \text{with } r_j^k\left(x_1,y_n\right)=\alpha_j^s, \tag{D8}$$

$$q_j^k\left(x_m,y_n\right)=q_j^k\left(x_{m-1},y_n\right)p_j^k\left(x_{m-1},y_n\right),\qquad \text{with } q_j^k\left(x_1,y_n\right)=\sigma_j, \tag{D9}$$

$$\rho_j^k(x_m, y_n) = \rho_j^k(x_{m-1}, y_n) + \sigma_j^k(x_{m-1}, y_n) q_j^k(x_{m-1}, y_n), \text{ with } \rho_j^k(x_1, y_n) = q_j^k(x_N, y_n). \quad \text{(D10)}$$

In addition, when Eq. (D1) is used to compute some perturbation functions with special particle velocities on x_1 or x_N, $h_j^k(x_0, y_n)$ and $h_j^k(x_{N+1}, y_n)$ are required to input as known conditions. Making use of periodic boundary conditions, we specified the two functions by

$$h_j^k(x_0, y_n) = h_j^k(x_N, y_n), \quad \text{(D11)}$$

$$h_j^k(x_{N+1}, y_n) = h_j^k(x_1, y_n). \quad \text{(D12)}$$

In summary, a CTDMA solving procedure includes calculation of p_j^k, o_j^k, q_j^k and ρ_j^k, σ_j^k, r_j^k on all nodes in the row of $y = y_n$ through use of Eqs. (D2) – (D6) and (D8) – (D10), respectively. The function $h_j^k(x_N, y_n)$ is then specified by Eq. (D7). Equation (D1), as the final step, will be solved in an order from x_{N-1} to x_1 to obtain all perturbation functions in the row of $y = y_n$ [36]. In the ADI in Section 4.3, we repeated this procedure row by row in its second half.

References

[1] A. Nabovati, D.P. Sellan, C. H. Amon, On the lattice Boltzmann method for phonon transport, Journal of Computational Physics 230 (2011) 5864-5876.

[2] U.M.B. Marconi and S. Melchionna, Kinetic theory of correlated fluids: From dynamic density functional to lattice Boltzmann methods, Journal of Chemical Physics 131 (2009) 014105.

[3] F. Wu, W. Shi and F. Liu, A lattice Boltzmann model for the Fokker-Planck equation, Communications in Nonlinear Science and Numerical Simulation 17 (2012) 2776-2790.

[4] X. Shan, H. Chen, Lattice Boltzmann model for simulating flows with multiple phases and components, Physical Review E 47 (1993) 1815-1820.

[5] R.R. Nourgaliev, T.N. Dinh, T.G. Theofanous and D. Joseph, The lattice Boltzmann equation method: theoretical interpretation, numerics and implications, International Journal of Multiphase Flow 29 (2003) 117-169.

[6] O. B. Usta, A.J.C. Ladd and J.E. Butler, Lattice-Boltzmann simulations of the dynamics of polymer solutions in periodic and confined geometries, Journal of Chemical Physics 122 (2005) 094902.

[7] Z.L. Guo and T.S. Zhao, Lattice Boltzmann model for incompressible flows through porous media, Physical Review E 66 (2002) 036304.

[8] M. Liu, Y. Shi, J. Yan and Y. Yan, Lattice Boltzmann simulation of flow and heat transfer in random porous media constructed by simulated annealing algorithm, Applied Thermal Engineering (2017). http://dx.doi.org/10.1016/j.applthermaleng.2016.12.107.

[9] C. Sun, C. Migliorini and L.L. Munn, Red blood cells initiate leukocyte rolling in postcapillary expansions: A lattice Boltzmann analysis, Biophysical Journal 85 (2003) 208-222.

[10] Y. Shi, T.S. Zhao and Z.L. Guo, Thermal lattice Bhatnagar-Gross-Krook model for flows with viscous heat dissipation in the incompressible limit, Physical Review E 70 (2004) 066310.

[11] M. Wang and Q. Kang, Modelling electrokinetic flows in microchannels using coupled lattice Boltzmann methods, Journal of Computational Physics 229 (2010) 728-744.

[12] H. Chen, *et al*., Extended Boltzmann kinetic equation for turbulent flows, Science 301 (2003) 633-636.

[13] Z.L. Guo and C. Shu, Lattice Boltzmann method and its applications in engineering, World Scientific Publishing, Singapore, 2013.

[14] D.P. Ziegler, Boundary conditions for lattice Boltzmann simulations, Journal of Statistical Physics, 71 (1993) 1171-1177.

[15] X. He, L.-S. Luo, Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation, Physical Review E 56 (1997) 6811-6817.

[16] X. Shan and X. He, Discretization of the velocity space in the solution of the Boltzmann equation, Physical Review Letters 80 (1998) 65-68.

[17] Y.-H. Zhang, X.-J. Gu, R.W. Barber and D.R. Emerson, Capturing Knudsen layer phenomena using a lattice Boltzmann model, Physical Review E 74 (2006) 046704.

[18] Z.L. Guo, B.C. Shi and C. G. Zheng, An extended Navier-Stokes formulation for gas flwos in the Knudsen layer near a wall, Europhysics Letters 80 (2007) 24001.

[19] Y. Shi, Y. W. Yap and J. E. Sader, Lattice Boltzmann method for linear oscillatory noncontinuum flows, Physical Review E 89 (2014) 033305.

[20] J. Meng and Y. Zhang, Accuracy analysis of high-order lattice Boltzmann models for rarefied gas flows, Journal of Computational Physics 230 (2011) 835-849.

[21] Y. Shi, Y. W. Yap and J. E. Sader, Linearized lattice Boltzmann method for micro- and nanoscale flow and heat transfer, Physical Review E 92 (2015) 013307.

[22] G.P. Ghiroldi and L. Gibelli, A finite-difference lattice Boltzmann approach for gas microflows, Communications in Computational Physics 17 (2015) 1007-1018.

[23] R. Luttge, Industrial micro & nano fabrication, William Andrew, N.Y., 2010.

[24] The MEMS handbook, edited by M. Gad-el-Hak, CRC press, Boca Raton, USA, 2002.

[25] J. W. Judy, Microelectromechanical systems (MEMS): Fabrication, design and applications, Smart Materials and Structures 10 (2001) 1115-1134.

[26] D. J. Laser and J. G. Santiago, A review of micropumps, Journal of Micromechanics and Microengineering 14 (2004) R35-R64.

[27] G. Karniadakis, A. Beskok, N. Aluru, Microflows and nanoflows: Fundamentals and simulation, Springer, N.Y., 2005.

[28] H.A. Stone, A.D. Stroock and A. Ajdari, Engineering flows in small devices: Microfluidics towards a lab-on-a-chip, Annual Review of Fluid Mechanics 36 (2004) 381-411.

[29] A. Kainz, *et al*., Air damping as design feature in lateral oscillators, Sensors and Actuators A 236 (2015) 357-363.

[30] M. K. Ghatkesar, *et al.*, Resonating modes of vibrating microcantilevers in liquid, Applied Physics Letters 92 (2008) 043106.

[31] T. P. Burg, *et al*., Weighing of biomolecules, single cells and nanoparticles in fluid, Nature 446 (2007) 1066-1069.

[32] Y. Shi and J. E. Sader, Lattice Boltzmann method for oscillatory Stokes flow with applications to micro- and nanodevices, Physical Review E 81 (2010) 036706.

[33] G.R. McNamara and G. Zanetti, Use of the Boltzmann equation to simulate lattice-gas automata, Physical Review Letters 61 (1988) 2332-2335.

[34] H. Chen, S. Chen and W.H. Matthaeus, Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method, Physical Review A 45 (1992) R5339.

[35] C. D. Conte and C. de Boor, Elementary numerical analysis: An algorithmic approach, McGraw-Hill, N.Y., 1980.

[36] W.Q. Tao, Numerical heat transfer, Xi'an Jiaotong university press, Xi'an, 2001.

[37] F. Zhang, Linear algebra: Challenging problems for students, The Johns Hopkins University Press, Baltimore, 2009.

[38] L. H. Thomas, Elliptic problems in linear differential equations over a network, Watson science computer laboratory report, Columbia university, N.Y., 1949.

[39] S. V. Patankar, C. H. Liu and E. M. Sparrow, Fully developed flow and heat transfer in ducts having streamwise-periodic variation of cross sectional area, ASME Journal of Heat Transfer 99 (1977) 180-186.

[40] D. W. Peaceman and H. H. Rachford, The numerical solution of parabolic and elliptic differential equations, Journal of the Society for Industrial and Applied Mathematics 3 (1955) 28-41.

[41] C. Cercignani, Rarefied gas dynamics: From basic concept to actual calculations, Cambridge university press, Cambridge, UK, 2000.

[42] Y. H. Qian, D. D'Humieres and P. Lallemand, Lattice BGK models for Navier-Stokes equations, Europhysics Letters 17 (1992) 479-484.

[43] Z. L. Guo, C. Zheng and B. Shi, Non-equilibrium extrapolation method for velocity and pressure boundary conditions in the lattice Boltzmann method, Chinese Physics 11 (2002) 366-374.

[44] G. G. Stokes, Mathematical and physical papers vol. 3, Cambridge university press, Cambridge, UK, 1901.

[45] L. Rosenhead, Laminar boundary layer, Clarendon press, Oxford, UK, 1963.