Two accurate off-lattice pressure boundary algorithms in lattice Boltzmann method

Zongze Li
Supervisor: Dr. Mao, Wenbin

Florida Fluids Symposium II, Orlando, FL
May 12, 2023

UNIVERSITY OF SOUTH FLORIDA

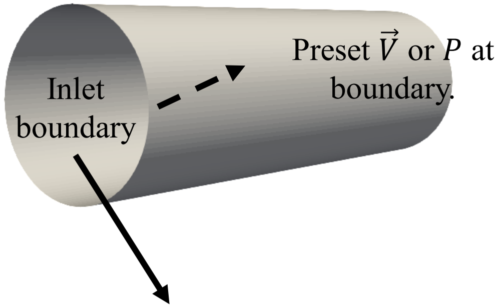
A PREEMINENT RESEARCH UNIVERSITY

UNIVERSITY OF SOUTH FLORIDA

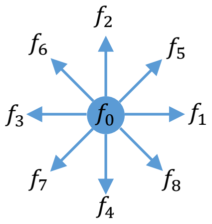
Introduction:

- Interest in the lattice Boltzmann method (LBM) has been steadily increasing.
- Boundary condition (BC) is a necessity for numerical methods.
  - Uniquely determine the solution.
  - Affect stability and accuracy.
- Various choices for off-lattice velocity BC but rare choice for pressure BC.
  - Velocity BC: Bounce back[1] (BB), Bouzidi[2] (BFL), Zhao Yong [3] (ZY), Yu [4] (YLI) etc.
  - Pressure BC: Pressure anti-bounce back[5] (PAB), pressure linear interpolation [5] (PLI).
- Objective: develop an algorithm for off-lattice pressure BC.

Implementation of BC in LBM



D2Q9:



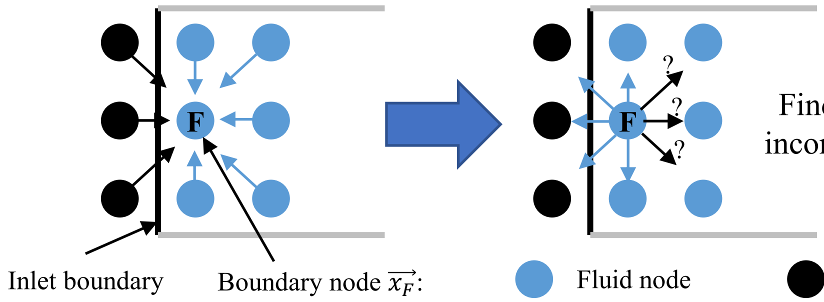
BGK collision operator

$$\Omega_i(\vec{x},t) = -\frac{f_i(\vec{x},t) - f_i^{eq}(\vec{x},t)}{\tau}$$

$$f_i^{eq}(\vec{x},t) = \omega_i \rho \left(1 + \frac{\vec{u}\cdot\vec{c_i}}{c_s^2} + \frac{(\vec{u}\cdot\vec{c_i})^2}{2c_s^4} - \frac{\vec{u}\cdot\vec{u}}{2c_s^2}\right)$$

Post collision $f_i^*(\vec{x},t)$

Post streaming $f_i(\vec{x}+\vec{c_i},t+1)$

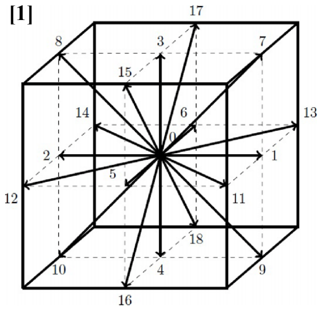


Find a way to get incoming f_1, f_5, f_8.

$$\frac{P}{c_s^2} = \rho = \sum_i f_i$$

$$\rho\vec{V} = \sum_i f_i \vec{c_i}$$

[1]



D3Q19 lattice

Lattice Boltzmann equations:

$$f_i^*(\vec{x},t) = f_i(\vec{x},t) + \Omega_i(\vec{x},t)$$
$$f_i(\vec{x}+\vec{c_i},t+1) = f_i^*(\vec{x},t)$$

Common Dirichlet velocity boundary condition (link-wise algorithm)

BB[1] (1994): $f_{\bar{i}}(\vec{x_F}, t+1) = f_i^*(\vec{x_F}, t) - 2\omega_i \rho_0 \frac{\vec{c_i} \cdot \vec{u_w}}{c_s^2}$

BFL[2] (2001):

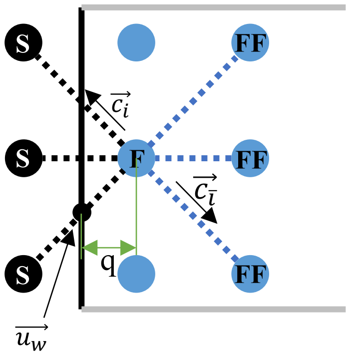
$f_{\bar{i}}(\vec{x_F}, t+1) = 2q f_i^*(\vec{x_F}, t) + (1-2q) f_i^*(\vec{x_{FF}}, t) - 2\omega_i \rho_0 \frac{\vec{c_i} \cdot \vec{u_w}}{c_s^2} \quad q < 0.5$

$f_{\bar{i}}(\vec{x_F}, t+1) = \frac{1}{2q} f_i^*(\vec{x_F}, t) + \frac{2q-1}{2q} f_{\bar{i}}^*(\vec{x_F}, t) - \frac{1}{q}\omega_i \rho_0 \frac{\vec{c_i} \cdot \vec{u_w}}{c_s^2} \quad q \geq 0.5$

YLI [4] (2003): $f_{\bar{i}}(\vec{x_F}, t+1) = \frac{q}{1+q} f_i^*(\vec{x_F}, t) + \frac{q}{1+q} f_{\bar{i}}^*(\vec{x_F}, t) + \frac{1-q}{1+q} f_i^*(\vec{x_{FF}}, t) - \frac{2}{1+q}\omega_i \rho_0 \frac{\vec{c_i} \cdot \vec{u_w}}{c_s^2}$

CLI [5] (2008): $f_{\bar{i}}(\vec{x_F}, t+1) = f_i^*(\vec{x_F}, t) + \frac{1-2q}{1+2q} f_{\bar{i}}^*(\vec{x_F}, t) + \frac{1-2q}{1+2q} f_i^*(\vec{x_{FF}}, t) - \frac{4}{1+2q}\omega_i \rho_0 \frac{\vec{c_i} \cdot \vec{u_w}}{c_s^2}$

ZY [3] (2017): $f_{\bar{i}}(\vec{x_F}, t+1) = \frac{1}{1+2q} f_i(\vec{x_F}, t) + \frac{2q}{1+2q} f_{\bar{i}}^*(\vec{x_F}, t) - \frac{2}{1+2q}\omega_i \rho_0 \frac{\vec{c_i} \cdot \vec{u_w}}{c_s^2}$

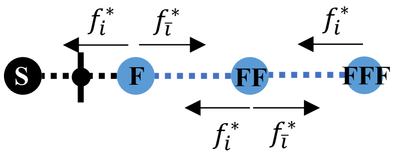


General equations

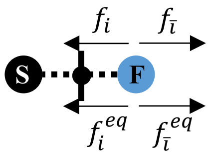
Ginzburg[5] (2008): $f_{\bar{i}}(\overrightarrow{x_F}, t+1) = k_1 f_i^*(\overrightarrow{x_F}, t) + k_0 f_i^*(\overrightarrow{x_{FF}}, t) + k_{-1} f_i^*(\overrightarrow{x_{FFF}}, t) + \bar{k}_{-1} f_{\bar{i}}^*(\overrightarrow{x_F}, t) + \bar{k}_{-2} f_{\bar{i}}^*(\overrightarrow{x_{FF}}, t) + \varphi_i(\overrightarrow{x_w}, t) + f_i^{p.c.}(\overrightarrow{x_F}, t)$

Zhao, Yong [3] (2017): $f_{\bar{i}}(\overrightarrow{x_F}, t+1) = a_1 f_i(\overrightarrow{x_F}, t) + a_2 f_{\bar{i}}(\overrightarrow{x_F}, t) + a_3 f_i^{eq}(\overrightarrow{x_F}, t) + a_4 f_{\bar{i}}^{eq}(\overrightarrow{x_F}, t) + a_5 \omega_i \rho_0 \frac{\overrightarrow{c_i} \cdot \overrightarrow{u_w}}{c_s^2}$

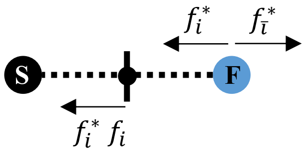
Marson [7] (2021): $f_{\bar{i}}(\overrightarrow{x_F}, t+1) = a_1 f_i^*(\overrightarrow{x_F}, t) + a_2 f_{\bar{i}}^*(\overrightarrow{x_F}, t) + a_3 f_i(\overrightarrow{x_w}, t+1) + a_4 f_i^*(\overrightarrow{x_w}, t)$



Ginzburg



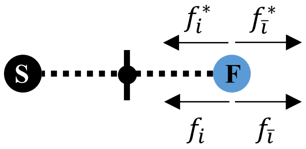
Zhao, Yong



Marson

Our goal: a single-node pressure BC, so our general equation is:

$$f_{\bar{i}}(\overrightarrow{x_F}, t+1) = A f_i^*(\overrightarrow{x_F}, t) + B f_i(\overrightarrow{x_F}, t) + C f_{\bar{i}}^*(\overrightarrow{x_F}, t) + D f_{\bar{i}}(\overrightarrow{x_F}, t) + E \omega_i \rho_F \left(1 + \frac{(\overrightarrow{c_i} \cdot \overrightarrow{u_w})^2}{2c_s^4} - \frac{\overrightarrow{u_w} \cdot \overrightarrow{u_w}}{2c_s^2}\right)$$



Our algorithm

Derivation of Single node (SN) algorithm

$$\boxed{f_{\bar{i}}(\overrightarrow{x_F}, t+1) = Af_i^*(\overrightarrow{x_F}, t) + Bf_i(\overrightarrow{x_F}, t) + Cf_{\bar{i}}^*(\overrightarrow{x_F}, t) + Df_{\bar{i}}(\overrightarrow{x_F}, t)} + \boxed{E\omega_i\rho_F\left(1 + \frac{(\overrightarrow{c_i}\cdot\overrightarrow{u_w})^2}{2c_s^4} - \frac{\overrightarrow{u_w}\cdot\overrightarrow{u_w}}{2c_s^2}\right)}$$

Chapman-Enskog expansion

Taylor expansion

$$\begin{cases} A + B - C - D = -1 \\ A + B + C + D + E = 1 \\ A - C + qE = 0 \\ A + C + \tau E = 0 \end{cases}$$

$\longrightarrow$ Reach 1st order accuracy for velocity

$\longrightarrow$ Reach 1st order accuracy for pressure

$\longrightarrow$ Reach 2nd order accuracy for pressure

$\longrightarrow$ Reach 2nd order accuracy for velocity

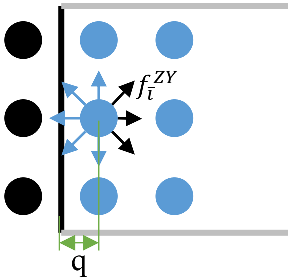
Make E = 2

$$\begin{cases} A = -q - \tau \\ B = q + \tau - 1 \\ C = q - \tau \\ D = \tau - q \\ E = 2 \end{cases}$$

$$f_{\bar{i}}(x_F, t+1)$$
$$= (-q-\tau)f_i^*(x_F, t) + (q+\tau-1)f_i(x_F, t) + (q-\tau)f_{\bar{i}}^*(x_F, t) + (\tau-q)f_{\bar{i}}(x_F, t)$$
$$+ 2\omega_i\rho_F\left(1 + \frac{(\overrightarrow{c_i}\cdot\overrightarrow{u_w})^2}{2c_s^4} - \frac{\overrightarrow{u_w}\cdot\overrightarrow{u_w}}{2c_s^2}\right)$$

Density correction (DC) scheme

1. Apply ZY velocity:

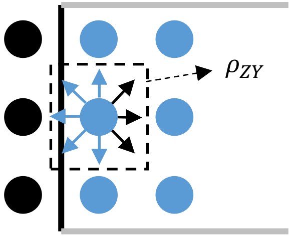


$$f_{\bar{\imath}}^{ZY}(\overrightarrow{x_F},t)$$

$$=\frac{1}{1+2q}f_i(\overrightarrow{x_F},t)+\frac{2q}{1+2q}f_{\bar{\imath}}^{*}(\overrightarrow{x_F},t)$$

$$-\frac{2}{1+2q}\omega_i\rho_0\frac{\overrightarrow{c_i}\cdot\overrightarrow{u_w}}{c_s^2}$$

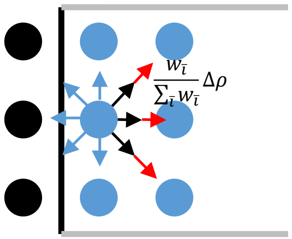
2. Find correction ($\Delta\rho$):



$$\rho_{ZY}=\sum_i f_i(\overrightarrow{x_F},t)$$

$$\Delta\rho=\rho_F-\rho_{ZY}$$

3. Add correction:



$$f_{\bar{\imath}}(\overrightarrow{x_F},t+1)=f_{\bar{\imath}}^{ZY}(\overrightarrow{x_F},t)+\frac{w_{\bar{\imath}}}{\sum_{\bar{\imath}} w_{\bar{\imath}}}\Delta\rho$$

$\bar{\imath}$: *incoming direction*

Macroscopic velocity and density in the algorithms

$$\text{SN: } f_{\bar{i}}(x_F,t+1) = (-q-\tau) f_i^*(x_F,t) + (q+\tau-1) f_i(x_F,t) + (q-\tau) f_{\bar{i}}^*(x_F,t) + (\tau-q) f_{\bar{i}}(x_F,t) + 2\omega_i \boldsymbol{\rho_F}\left(1+\frac{(\vec{c_i}\cdot\vec{\boldsymbol{u_w}})^2}{2c_s^4} - \frac{\vec{\boldsymbol{u_w}}\cdot\vec{\boldsymbol{u_w}}}{2c_s^2}\right)$$

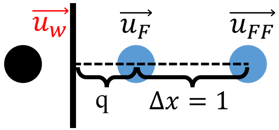
$$\text{DC: } f_{\bar{i}}(\vec{x_F},t+1) = \frac{1}{1+2q} f_i(\vec{x_F},t) + \frac{2q}{1+2q} f_{\bar{i}}^*(\vec{x_F},t) - \frac{2}{1+2q}\omega_i\rho_0 \frac{\vec{c_i}\cdot\vec{\boldsymbol{u_w}}}{c_s^2} + \frac{w_{\bar{i}}}{\sum_{\bar{i}} w_{\bar{i}}}(\boldsymbol{\rho_F} - \rho_{ZY})$$

Extrapolation of $\vec{\boldsymbol{u_w}}$ link-wise:

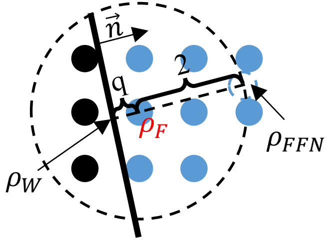
Interpolation of $\boldsymbol{\rho_F}$ in $\vec{n}$ direction:

Has fluid neighbor:

$$\vec{\boldsymbol{u_w}} = q(\vec{u_F} - \vec{u_{FF}}) + \vec{u_F}$$

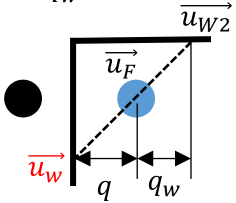


$$\boldsymbol{\rho_F} = \frac{2}{2+q}(\rho_W - \rho_{FFN}) + \rho_{FFN}$$

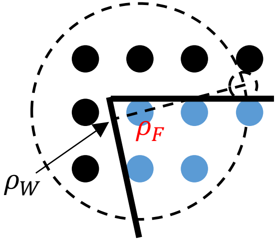


No fluid neighbor:

$$\vec{\boldsymbol{u_w}} = \frac{q}{q_w}(\vec{u_F} - \vec{u_{W2}}) + \vec{u_F}$$



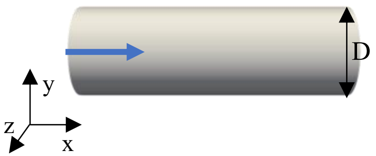
$$\boldsymbol{\rho_F} = \rho_W$$



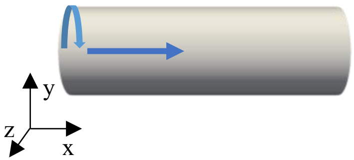
Simulation setup

Tested flow cases:

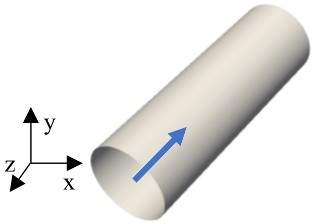
(1) Poiseuille flow;



(2) Taylor-Couette with axial flow;



(3) Poiseuille flow (arbitrary orientation);



Simulation setup:

$\frac{D}{dx} = 30$; $\tau = 1$; Incompressible BGK.

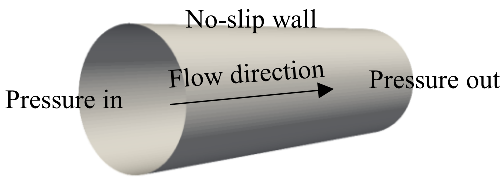
Other comparison algorithms:

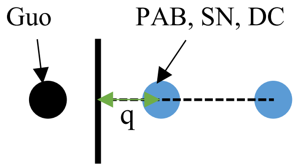
PAB[5]: $f_{\bar{i}}(\overrightarrow{x_F}, t+1) = -f_i^*(\overrightarrow{x_F}, t) + 2\omega_i \rho_w \left(1 + \frac{(\overrightarrow{c_i} \cdot \overrightarrow{u_w})^2}{2c_s^4} - \frac{\overrightarrow{u_w} \cdot \overrightarrow{u_w}}{2c_s^2}\right)$

Guo[6,8]: $f_{\bar{i}}(\overrightarrow{x_S}, t) = f_{\bar{i}}^{eq}(\rho_W, \overrightarrow{u_W}) + f_{\bar{i}}^{neq}(\overrightarrow{x_F}, t)$, where $\overrightarrow{u_W} = (\overrightarrow{u_F} \cdot \vec{n})\vec{n}$

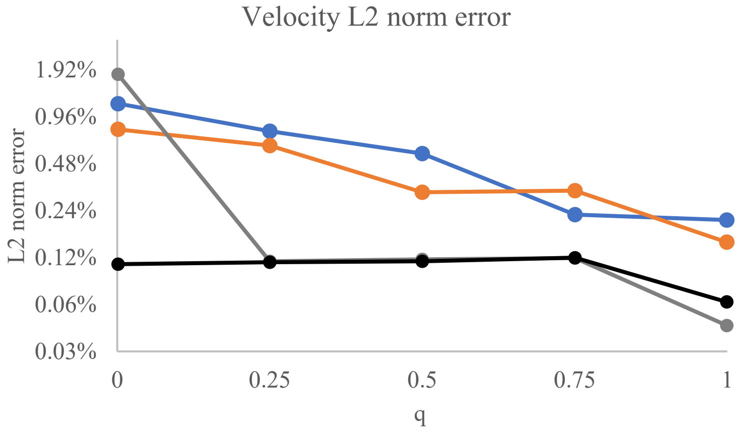
Accuracy with different q

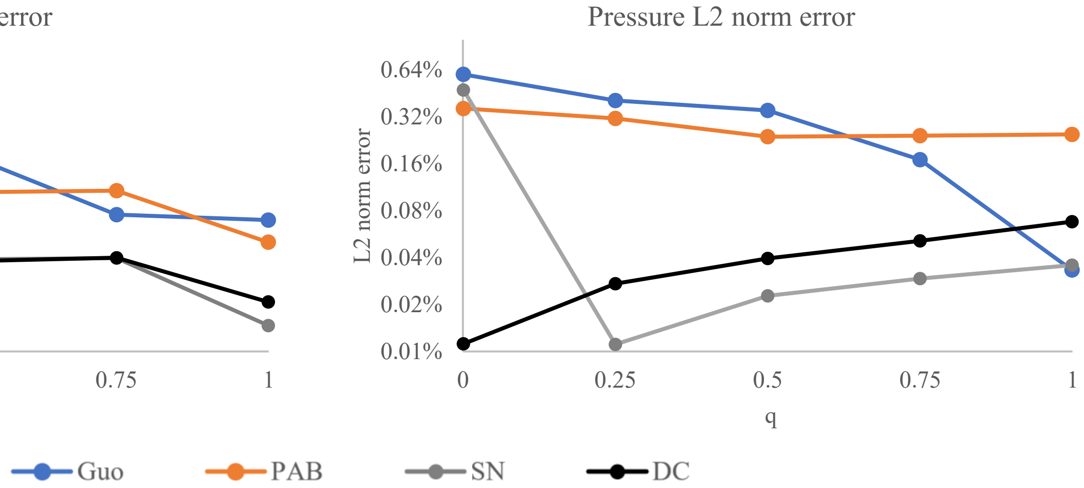
Simulation setup:





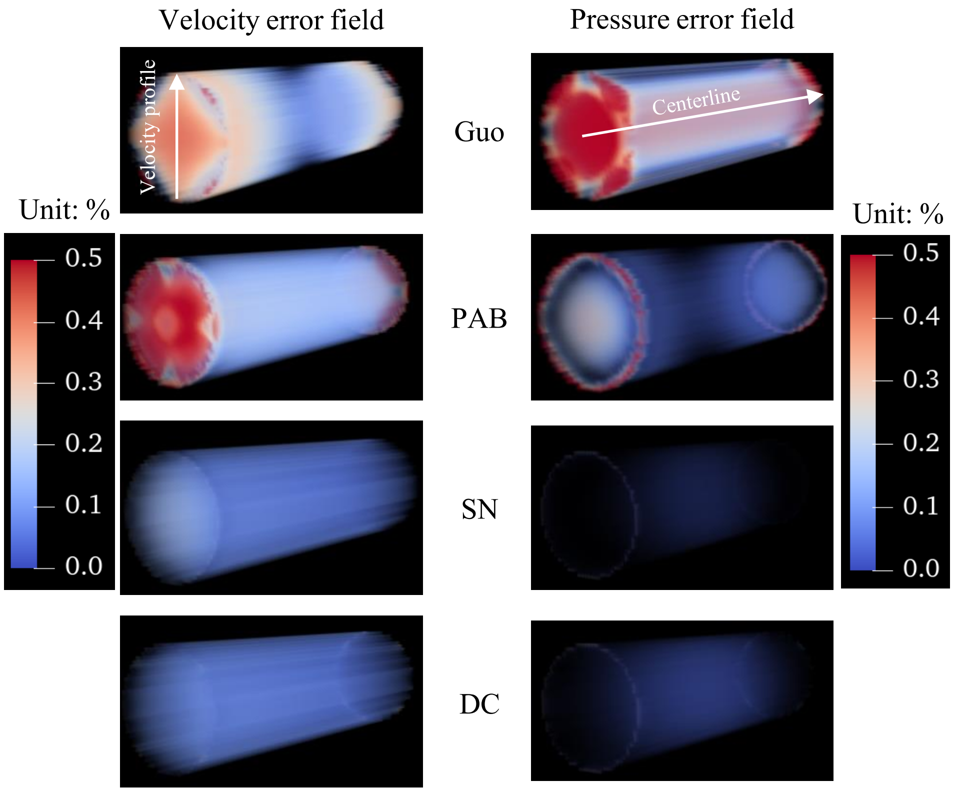
q = 0, 0.25, 0.5, 0.75, and 1.

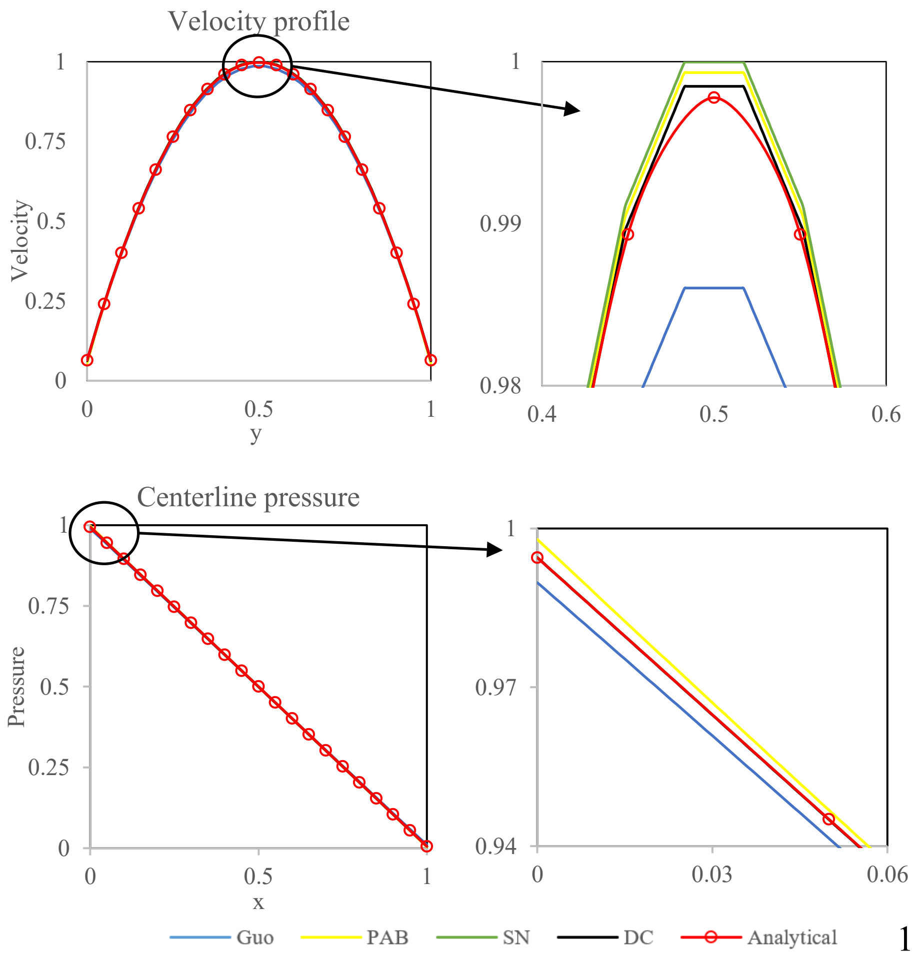




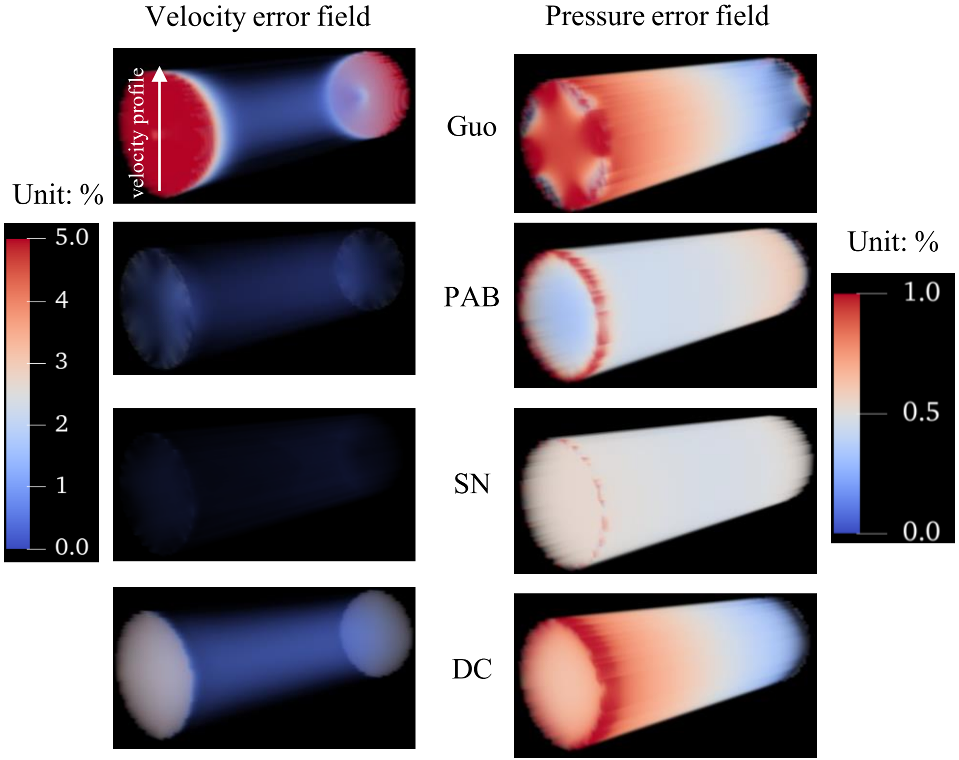
$$\text{L2 norm error} = \sqrt{\frac{\sum_{i=node}\left(\left\|\overrightarrow{v_{num_i}}\right\| - \left\|\overrightarrow{v_{theo_i}}\right\|\right)^2}{\sum_{i=node}\left(\left\|\overrightarrow{v_{theo_i}}\right\|\right)^2}}$$

Poiseuille flow in a cylinder

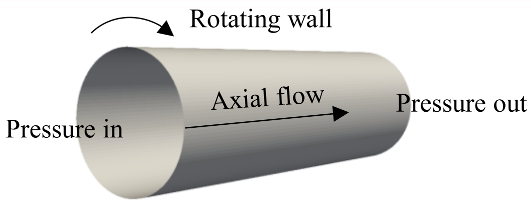


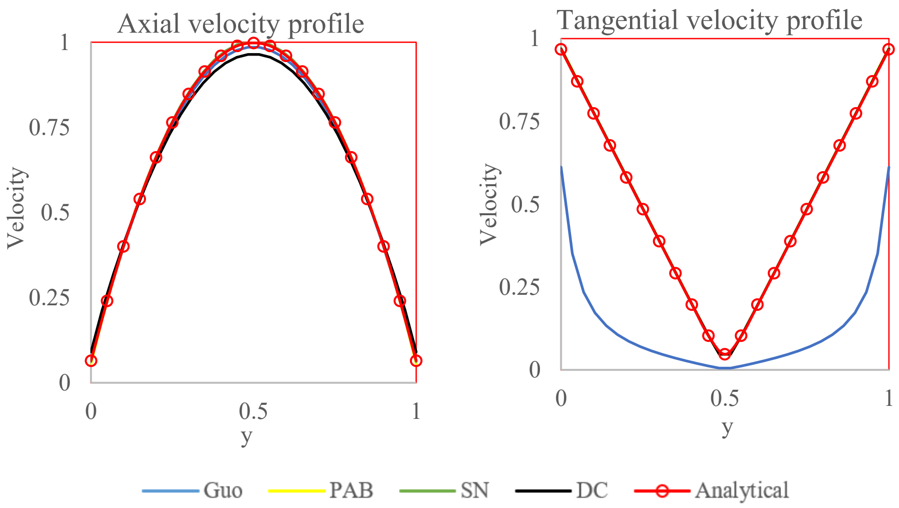


Taylor-Couette flow with axial velocity

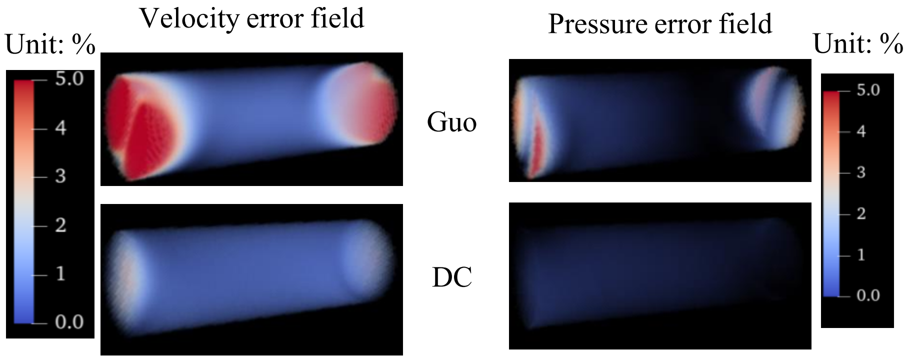


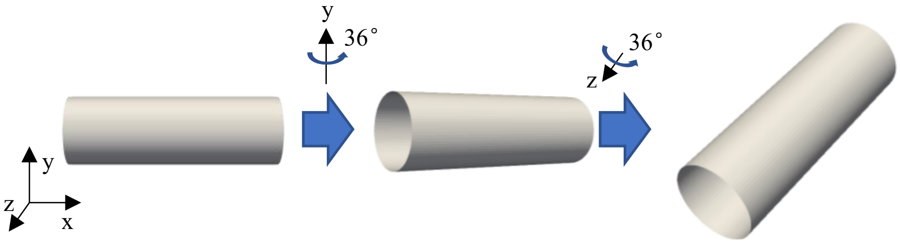
Simulation setup:





Poiseuille flow in a cylinder with arbitrary orientation





Usability

Algorithm	Normal flow	Tangential flow	Arbitrary orientation
Guo	√	×	√
PAB	√	√	×
SN	√	√	×
DC	√	√	√

Quantitative results

Poiseuille flow

Algorithm	V L2 norm error	P L2 norm error
Guo	0.56%	0.35%
PAB	0.31%	0.24%
SN	0.12%	0.02%
DC	0.11%	0.04%

Taylor-Couette flow with axial velocity

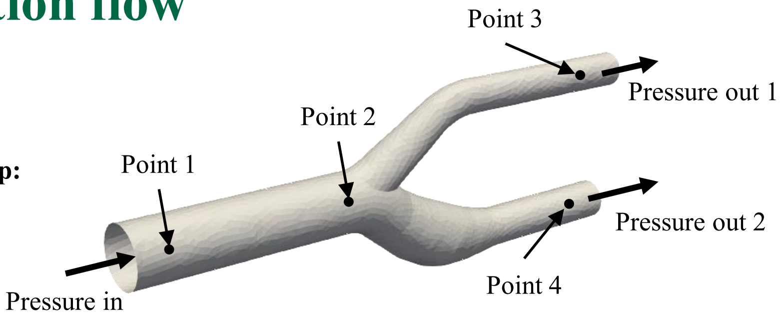
Algorithm	V L2 norm error	P L2 norm error
Guo	**7.42%**	1.06%
PAB	0.17%	0.90%
SN	0.05%	0.86%
DC	0.41%	0.96%

Poiseuille flow with arbitrary orientation

Algorithm	V L2 norm error	P L2 norm error
Guo	4.33%	1.32%
PAB	**nan**	**nan**
SN	**nan**	**nan**
DC	1.08%	0.36%

Bifurcation flow

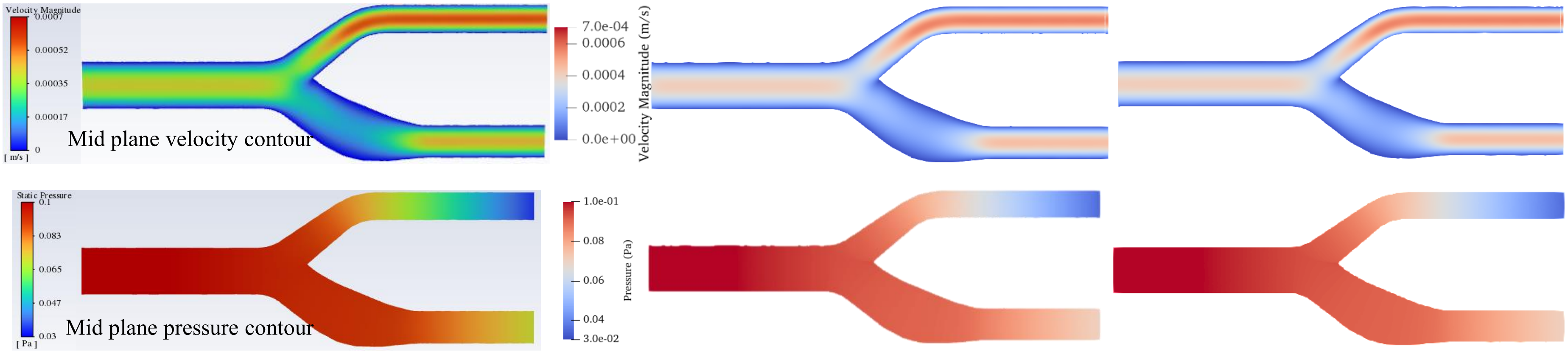
Simulation setup:



Algorithm	SN	DC	Benchmark	SN Difference	DC Difference
Inlet Q (m^3/s)	1.01E-08	1.01E-08	1.01E-08	**0.61%**	**0.45%**
Outlet 1 Q (m^3/s)	-4.80E-09	-4.80E-09	-4.82E-09	**0.44%**	**0.37%**
Outlet 2 Q (m^3/s)	-5.32E-09	-5.32E-09	-5.33E-09	**0.31%**	**0.13%**
Point 1 V (m/s)	4.09E-04	4.09E-04	4.11E-04	**0.43%**	**0.59%**
Point 2 V (m/s)	2.37E-04	2.39E-04	2.38E-04	**0.17%**	**0.41%**
Point 3 V (m/s)	5.57E-04	5.56E-04	5.56E-04	**0.13%**	**0.02%**
Point 4 V (m/s)	4.48E-04	4.49E-04	4.47E-04	**0.16%**	**0.43%**
Point 2 P (Pa)	9.27E-02	9.27E-02	9.28E-02	**0.13%**	**0.09%**

$$\text{difference} = \left|\frac{\text{num}_{\text{LBM}} - \text{num}_{\text{Fluent}}}{\text{num}_{\text{Fluent}}}\right|$$

ANSYS Fluent benchmark | SN | DC



Conclusion:

- Two off-lattice pressure boundary condition are developed and compared to existing algorithms.
- The proposed SN algorithm has smaller error in all presented flow cases.
- The proposed DC algorithm is applicable in arbitrary orientation with reasonable error.

Future work:

- Improve SN algorithm to support the boundary with arbitrary orientation.
- Analyze the stability and accuracy of the two proposed algorithms under various relaxation time (τ), Mach number, and resolution.

Reference:

[1] Ladd A J C. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation[J]. Journal of fluid mechanics, 1994, 271: 285-309.

[2] Bouzidi M, Firdaouss M, Lallemand P. Momentum transfer of a Boltzmann-lattice fluid with boundaries[J]. Physics of fluids, 2001, 13(11): 3452-3459.

[3] Zhao W, Yong W A. Single-node second-order boundary schemes for the lattice Boltzmann method[J]. Journal of Computational Physics, 2017, 329: 1-15.

[4] Yu D, Mei R, Shyy W. A unified boundary treatment in lattice boltzmann method[C]//41st aerospace sciences meeting and exhibit. 2003: 953.

[5] Ginzburg I, Verhaeghe F, d'Humieres D. Two-relaxation-time lattice Boltzmann scheme: About parametrization, velocity, pressure and mixed boundary conditions[J]. Communications in computational physics, 2008, 3(2): 427-478.

[6] Guo Z, Zheng C, Shi B. An extrapolation method for boundary conditions in lattice Boltzmann method[J]. Physics of fluids, 2002, 14(6): 2007-2010.

[7] Marson F, Thorimbert Y, Chopard B, et al. Enhanced single-node lattice Boltzmann boundary condition for fluid flows[J]. Physical Review E, 2021, 103(5): 053308.

[8] Francesco Marson and Jonathan Lemus and Christos Kotsalos and Raphaël Conradin and Christophe Coreixas and 604 Rémy Petkantchin and Franck Raynaud and Joël Beny and Bastien Chopard, 605 J.L.a.O.M.a.D.K.a.A.P.a.D.L.a.F.B.a.M.B.B.a.Y.T.a.S.L.a.S.L.a., Palabos: Parallel Lattice Boltzmann Solver. 606 Computers & Mathematics with Applications, 2020.

Thank you!

UNIVERSITY of SOUTH FLORIDA