Implementation of the Discrete Unified Gas Kinetic Scheme in Code_Saturne targeting Exascale

Chrysovalantis Tsigginos[1], Jianping Meng[2], and Charles Moulinec[1]

[1]Scientific Computing Department, STFC Daresbury Laboratory, Warrington WA4 4AD, United Kingdom
[2]Research IT, IT Services, University of Liverpool, Brownlow Hill, Liverpool L69 3GG, UK.

***Abstract*—A new module entitled CS_DUGKS was developed within the framework of Code_Saturne, an open-source CFD software, to solve the Boltzmann-BGK equation using a discrete velocity approach. CS_DUGKS utilizes the discrete unified gas kinetic scheme to reconstruct the flux of the finite volume discretization of the Boltzmann-BGK equation. Three collisional models were implemented into CS_DUGKS. CS_DUGKS was calibrated against two-dimensional cavity flows and rarefied Couette flows. Scaling tests conducted on ARCHER2 indicate that the developed module retains the excellent parallel performance of Code_Saturne.**

I. Introduction

Many industrial transport processes may exceed the limits of continuum theory. For instance, clean energy pathway technologies, such as fuel cells [9, 8] and CO_2 storage in shales [5], involve gas flows at the micro/nano level. Some clean energy pathway technologies even extend into the quantum realm, such as solar energy (radiative heat transfer) and neutron transfer in nuclear power plants [11]. To accurately address these challenges, the Boltzmann equation and/or similar transport equations need to be solved. However, directly solving the Boltzmann equation is computationally intensive and memory demanding [12, 6]. This is because the numerical solution requires discretizing the physical space along with an additional space, the molecular velocity space. After discretizing the molecular velocity space, the Boltzmann equation transforms into a set of scalar transport equations, solvable only in the physical space. The number of equations increases with the Knudsen number, defined as the ratio of the molecular mean free path to a representative length scale. Therefore, it is not uncommon to have over 1,000 such transport equations for gas flows at the micro/nano level or at high altitudes.

The significant advancement in computational power brought by Petascale and, more recently, by Exascale systems has enabled the simulation of large mesoscale applications that was practically impossible a few years ago. In this context, Code_Saturne [1], an open-source computational fluid dynamics software optimized for high-performance systems [3], serves as an excellent platform for implementing Boltzmann-type equations.

The scope of this project is the development of an open-source finite-volume solver entitled CS_DUGKS within the computational framework of Code_Saturne, which solves the Boltzmann-BGK equation. The developed module utilizes the discrete unified gas kinetic scheme, DUGKS [7], which solves the Boltzmann-BGK equation through a finite volume discretization of the physical space and a finite difference discretization of the molecular space. Its unique flux reconstruction algorithm enables the mesh sizes and timesteps to be significantly larger than the kinetic scale, facilitating the multiscale modeling of various flow regimes.

II. The discrete unified gas kinetic scheme

In DUGKS, the Boltzmann-BGK equation is first discretized in the velocity space using a velocity set $\boldsymbol{\xi}_i, i = 1, 2, \ldots, N$, leading to a system of N partial differential equations. Then, the resulting equations are discretized in the physical space via a second-order finite-volume cell-centered scheme:

$$f_i^{n+1,c} - f_i^{n,c} + \frac{\Delta t}{V_c} \sum_f \mathcal{F}_i^{n+1/2,f} = \frac{\Delta t}{2} \left[\Omega_i^{n+1,c} - \Omega_i^{n,c} \right] \quad (1)$$

where $f_i^{n,c}$ and $\Omega_i^{n,c}$ represent the distribution function and the collision operator computed at the center of cell c for the discrete velocity $\boldsymbol{\xi}_i$. The terms Δt and V_c denote the timestep and the volume of cell, c, respectively. The flux, $\mathcal{F}_i^{n+1/2,f}$,

$$\mathcal{F}_i^{n+1/2,f} = \boldsymbol{\xi}_i \cdot n_i^f f_i^{n+1/2,f} \Delta S_f \quad (2)$$

is computed at the center of interior and boundary faces. The terms n_i^f and ΔS_f denote the outward unit vector at face i and the area of face, f, respectively. For interior faces, the distribution function, $f_i^{f,n+1/2}$ is calculated by integrating along the characteristic line that ends at the face center for a timestep $\Delta h = 0.5\Delta t$:

$$\begin{aligned} f_i^{n+1/2,f} &= f_i^n(\boldsymbol{x}_f - \boldsymbol{\xi}_i \Delta h) \\ &+ \frac{\Delta h}{2} \left[\Omega_i^{n+1/2,f} + \Omega^{n,f}(\boldsymbol{x}_f - \Delta h \boldsymbol{\xi}_i) \right]. \end{aligned} \quad (3)$$

By introducing the new populations, $\bar{f} = f - 0.5\Delta h\Omega$ and $\bar{f}^+ = f + 0.5\Delta h\Omega$, the above equation can be written in an explicit form:

$$\bar{f}^{n+1/2,f} = \bar{f}^{+n}(\boldsymbol{x}_f - \boldsymbol{\xi}_i \Delta h). \quad (4)$$

The distribution function $\bar{f}^+$ at $\boldsymbol{x}_f - \boldsymbol{\xi}_i \Delta h$ is to be reconstructed via a first-order Taylor expansion. The required gradient, $\nabla \bar{f}^+$ is computed with the build-in gradient schemes of Code_Saturne based on the estimates of $\bar{f}_i^{+n}$ at cell centers. Finally, by introducing the algebraic transformations,

$\tilde{f}_i = f_i - 0.5\Delta t\Omega_i$ and $\tilde{f}_i^+ = f_i + 0.5\Delta t\Omega_i$, Eq. (1) can be written in an explicit form:

$$\tilde{f}_i^{n+1,c} = \tilde{f}_i^{+n,c} - \frac{\Delta t}{V_c}\sum_f \mathcal{F}_i^{n+1/2,f} \tag{5}$$

The timestep is determined by the Courant-Friedrichs-Lewy (CFL) condition

$$\Delta t = \alpha \frac{\Delta x}{U + ||\boldsymbol{x}_i||_{min}} \tag{6}$$

where $0 < \alpha < 1$ is the CFL number and Δx is the distance between the centers of two adjacent cells that share an interface.

The macroscopic variables can be computed directly from $\tilde{f}$:

$$\rho = \sum_i w_i \tilde{f}_i, \rho u_a = \sum_i w_i \xi_{i_a} \tilde{f}_i, e = \frac{1}{2}\sum w_i \xi_{i_a}\xi_{i_a}\tilde{f}_i \tag{7}$$

where ρ, u_a, e, the gas density, velocity and internal energy at point $\boldsymbol{x}_c$, respectively. The term, w_i, denotes the weight of the chosen quadrature scheme. Please note the stress tensor, σ_{ij} and heat flux, q_a are not given directly in terms of $\tilde{f}$ but are expressed as a combination of the new distribution $\tilde{f}$ and the employed equilibrium function [13].

The numerical scheme for solving the Boltzmann-BGK equation based on DUGKS has as follows:

- Compute $\tilde{f}_i^{+n}$ at cell centers.
- Compute $\bar{f}^+$ at cell centers
- Compute the gradient of $\bar{f}^+$
- For interior faces reconstruct $\bar{f}^+ s$ at $\boldsymbol{x}_f - \Delta h\boldsymbol{\xi}_i$ and update $\tilde{f}_i$ due to contributions to Eq. (1)
- For boundary faces compute $\bar{F}^f$ for emerging populations and update $\tilde{f}$ due to contributions to Eq. (1)
- Compute macroscopic variables at cell centers based on the new estimates of $\tilde{f}_i$
- Update fluid properties

III. Implementation into Code_Saturne

The discrete unified gas kinetic scheme is implemented in Code_Saturne as a standalone package, CS_DUGKS. Specifically, new high-level structures were developed to meet the needs of the developed package. At the same time, the built-in functionalities of Code_Saturne are used for data storage, MPI communications, data output, write/restart, etc.

A. Structure of the solver

The primary structure of CS_DUGKS is the `cs_dugks_system_t` structure, which contains all the `cs_fields_t` structures, pointers to high-level functions, and a `cs_dugks_param_t` structure. The latter encompasses all the necessary functionality for simulation management within CS_DUGKS. Field data are divided into mesoscopic fields, which consist of the distribution function at each discrete velocity field, and macroscopic fields, categorized into two groups: (i) fields computed directly from mesoscopic fields, i.e., ρ, u_i, σ_{ij}, etc., and (ii) fields computed from the previous group. High-level functions within `cs_dugks_system_t` include explicit and implicit solvers, `cs_dugks_compute_t *` (the latter is a work in progress), functions for initializing the distribution fields from the initial macroscopic variables, `cs_dugks_init_distrib_t *`, and functions for computing the macroscopic variables, `cs_dugks_compute_macros_t *`, the stress tensor, `cs_dugks_compute_stresses_t`, and the heat flux, `cs_dugks_compute_heat_flux_t`. The last three sets of functions depend on the employed molecular velocity quadrature scheme and the selected collision model. The following paragraph provides more details.

The `cs_dugks_param_t` structure contains all the essential information needed to set up and run simulations with CS_DUGKS. Specifically, it includes structures for discretizing the molecular velocity space, `cs_dugks_mol_velocity_params_t`, property structures, `cs_property_t`, to calculate the relaxation time and the sound speed, and `cs_xdef_t` structures for initializing macroscopic fields in various volume zones, as well as `cs_xdef_t` structures for defining boundaries. Initially, the `cs_xdef_t` structure was compatible only with the CDO module [2], but within this project, efforts have been made to adapt the `cs_xdef_t` structure for finite volume discretization schemes, including both the current package and the legacy part of Code_Saturne. Utilizing the `cs_xdef_t` structure enables a systematic process within Code_Saturne for defining simple or complex expressions for the initial and boundary conditions or gas properties.

CS_DUGKS currently provides three distinct quadrature schemes for discretizing the molecular velocity space. These quadrature schemes include the Gauss-Hermite, Gauss, and Newton-Cotes methods, with the last two applying only to rarefied flows. CS_DUGKS also stores the points at which the distribution function is computed, denoted as ξ_i, within the `cs_dugks_mol_velocity_params_t` structure, along with the quadrature weights if necessary. Furthermore, CS_DUGKS provides three collision models: i) the BGK, ii) the Shankov, and iii) the ellipsoidal model, accessed through `cs_dugks_equilibrium_distrib_t *` type functions that are compatible with the time marching scheme and the selected quadrature scheme.

B. Implementation of boundary conditions

The portion of the distribution function that transitions into the physical domain from actual and virtual boundaries must align with the macroscopic conditions specified at the boundary. CS_DUGKS currently provides boundary conditions for inlets, outlets, planes of symmetry, and both stationary and moving walls. At the mesoscopic level, the emerging part of the particle populations is calculated using specular reflection, diffuse reflection, or the bounce-back scheme for continuum flows.

Enforcing boundary conditions at the mesoscopic level in complex boundary surfaces requires to identify the emerging and impinging populations at different boundary faces. These particle populations are determined using the inner product,

$(\xi_i - u_i)n_i$, where u_i denotes the macroscopic boundary velocity and n_i is the outward unit vector normal to the boundary face. The identification of a particle population at a given boundary face as an emerging or impinging one is stored in `cs_dugks_bc_vel_map_t` structures. The above strategy significantly simplifies the implementation of mesoscopic boundary conditions for complex geometries.

For the specular reflection boundary condition, additional information are needed to reconstruct the emerging population from the impinging populations. For complex boundaries, the reflected populations cannot be constructed directly from a unique incident population, and a reconstruction process based on a finite element interpolation procedure must be used. The required information for reconstructing the reflected populations is stored in the structure `cs_dugks_bc_support_t`.

DUGKS calculates the distribution function, $\tilde{f}$, which is obtained from the distribution function, f, via an algebraic transformation. For the diffusive scattering and bounce-back methods, this algebraic transformation does not pose any difficulty in calculating the emerging populations, $\tilde{f}$. However, when applying the diffusive boundary condition to a transient problem, determining the emerging populations of the distribution function, $\tilde{f}$, in terms of the impinging populations becomes a non-linear challenge. The non-linearity arises from the necessity to compute the macroscopic variables $(\rho, \boldsymbol{u}, T)$ at a boundary face, which are essential for reconstructing the emerging portion of the distribution function, $\tilde{f}$. To tackle the above non-linearity, an iterative scheme is employed, that adjusts the emerging part of the distribution function based on the current estimates of the macroscopic variables calculated at the boundary faces. The maximum number of iterations and the convergence rate of this iterative process are stored in `cs_dugks_bc_les_t`.

C. Domain decomposition and MPI parallel communications

CS_DUGKS, similar to Code_Saturne, employs a physical space decomposition approach for data decomposition. The message-passing interface (MPI) protocol is used for intra-process communications. In the physical domain decomposition approach, each process owns a part of the computational domain while accounting for the complete set of molecular velocities. CS_DUGKS utilizes the built-in functionalities of Code_Saturne for domain partitioning and intra-solver data exchange. The user can select from all the available tools of Code_Saturne for domain decomposition, such as ParMETIS/METIS, Scotch/PT-Scotch, Morton-based space-filling curves, and Hilbert-type curves [3]. Intra-solver communications are managed using halo cells, which create copies of cells adjacent to the sub-domain boundaries. The built-in asynchronous MPI functionality of Code_Saturne handles communications between adjacent processes.

IV. Benchmark tests

To validate the developed module, two benchmark cases are considered. First, CS_DUGKS, the "spatial" implementation of the module, is validated by comparing it against benchmark cases of cavity flows. In the second step, the discretization of the molecular velocity space is validated by analyzing Couette flows at different Knudsen (Kn) numbers.

A. Continuum flows

For the validation of the temporal implementation of CS_DUGKS, a two-dimensional square cavity is considered, with the top wall moving in the x-direction at a velocity of $u_w = 0.1c_s$, where c_s is the sound speed, while the remaining walls remain stationary. The kinematic viscosity, ν, is set to $0.0001c_sH$. Therefore, the Reynolds number, $Re = uH/\nu$, is equal to 1000, where H is the height of the cavity.

Structured and unstructured meshes are utilized for validating the temporal implementation of CS_DUGKS. The D3Q19 lattice was employed to discretize the molecular velocity space. The bounce-back scheme is implemented to enforce the no-slip boundary condition on both moving and stationary walls. Steady-state conditions are achieved when the norm $||\boldsymbol{u}^{n+1} - \boldsymbol{u}^n||$ becomes smaller than $10^{-8}||\boldsymbol{u}^n||$, where $\boldsymbol{u}^n$ represents the velocity at the n^{th} increment and $||\cdot||$ denotes the L_2 norm.

Structured meshes of 40×40, 80×80 and 128×128 elements were used to validate the developed module. The velocity profiles, $\boldsymbol{u} = (u, v)$, along vertical and horizontal lines passing through the center of the cavity are given in Fig. 1. For comparison purposes, the benchmark data from [4] and the velocity profiles obtained from the incompressible Navier-Stokes solver of Code_Saturne are included in the figure. An excellent agreement is observed between the CS_DUGKS and the benchmark velocity profiles for meshes with at least 80 elements in each direction.

The developed module can also utilize unstructured meshes to perform mesh refinements near boundaries. To validate the use of unstructured meshes with CS_DUGKS, the same case as before is tackled. Unstructured meshes with 4000, 10000, and 20000 triangles are employed. The velocity profiles along vertical and horizontal lines passing through the center of the cavity, along with the benchmark results from [4], are presented in Fig. (2). The excellent agreement between the velocity profiles obtained by CS_DUGKS and the benchmark cases highlights the capabilities of the developed module to simulate arbitrary geometries at a mesoscopic level.

B. Couette flows

Herein, the microplanar Couette flow between two horizontal plates separated by a distance H is examined. The top and bottom plates move at a constant velocity of $\pm0.1c_s$, where c_s represents the reference sound speed.

A uniform mesh of 100 elements in the z-direction is used to discretize the physical space. Periodic boundary conditions are implemented to simulate a semi-infinite domain. The diffuse-scattering boundary condition is used to impose the boundary conditions at different K_D numbers, where K_D reads:

$$K_D = \sqrt{\frac{\pi}{2}}\frac{\nu_0}{c_sH} \tag{8}$$

ν_0 is the reference kinematic viscosity. The Shankov model with a Prandtl number of 2/3 is utilized to simulate collisions

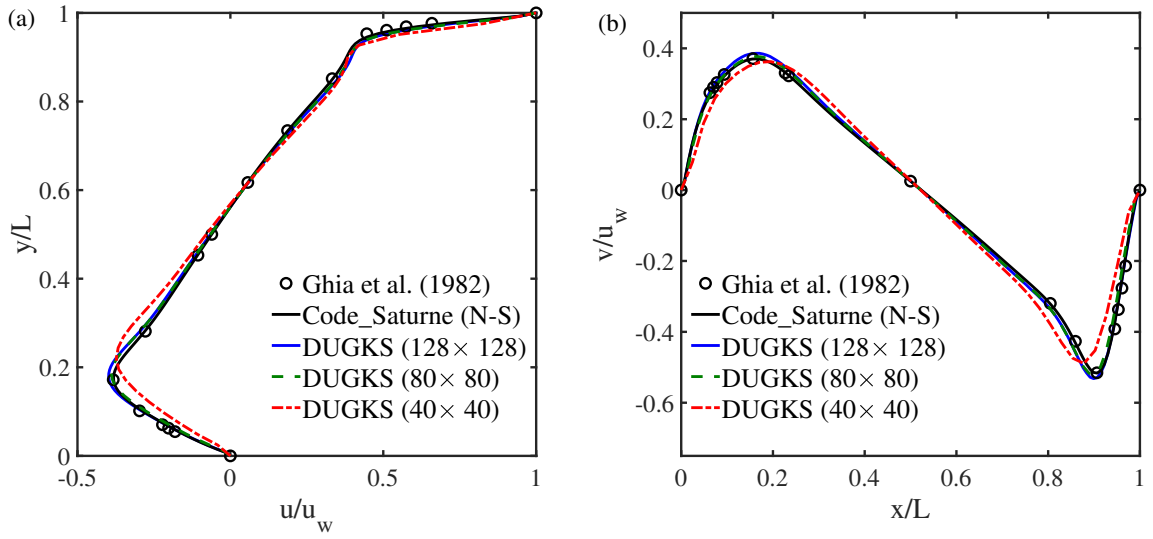

Fig. 1. (a) Horizontal and (b) vertical velocity profiles across the cavity center at $Re = 1000$ for uniform meshes of different sizes.

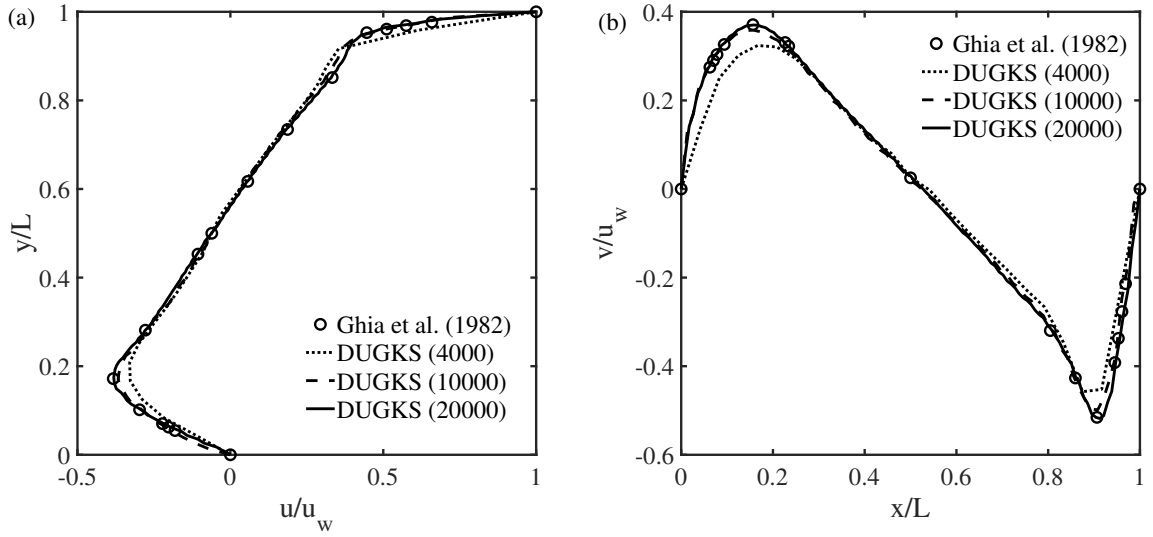

Fig. 2. (a) Horizontal and (b) vertical velocity profiles across the cavity center at $Re = 1000$ for unstructured meshes of different sizes.

in a hard-sphere gas. An 11th-order Gauss-Hermite quadrature is employed to discretize the molecular velocity space.

The horizontal velocities for Couette flow at different K_D numbers are presented in Fig. 3. For comparison, the DSMC results from [10] are also included in the figure. There is good agreement between the DUGKS and the DSMC results for all considered K_D numbers, with the non-linearity near the walls being successfully captured.

V. Parallel performance

To assess the parallel performance of CS_DUGKS as a standalone package, gas flows at various Knudsen numbers are examined in a three-dimensional cavity with dimensions of $(10m, 10m, 5m)$. In this setup, the top wall moves with a horizontal velocity of $u_o = 0.01m/s$, while the other walls remain stationary. Intermolecular collisions are modeled using the BGK model. Several relaxation times are used to simulate different flow conditions. In all instances, the diffuse reflection boundary condition is employed to model the particle-wall interactions.

Structured meshes of 55 and 256 million cells are used to mesh the three-dimensional cavity. Both meshes are generated using Code_Saturne's mesh multiplication capabilities from initial structured meshes of 0.108 and 0.5 million elements, respectively. Additionally, unstructured meshes of 88 million and 704 million tetrahedral elements are also considered. These meshes are produced from an initial mesh of 11 million tetrahedral elements using Code_Saturne's built-in mesh multiplication functionalities. To test the parallel performance of the developed module for flows with different Knudsen numbers, three sets of molecular velocities with 27, 125, and 1000 points are considered.

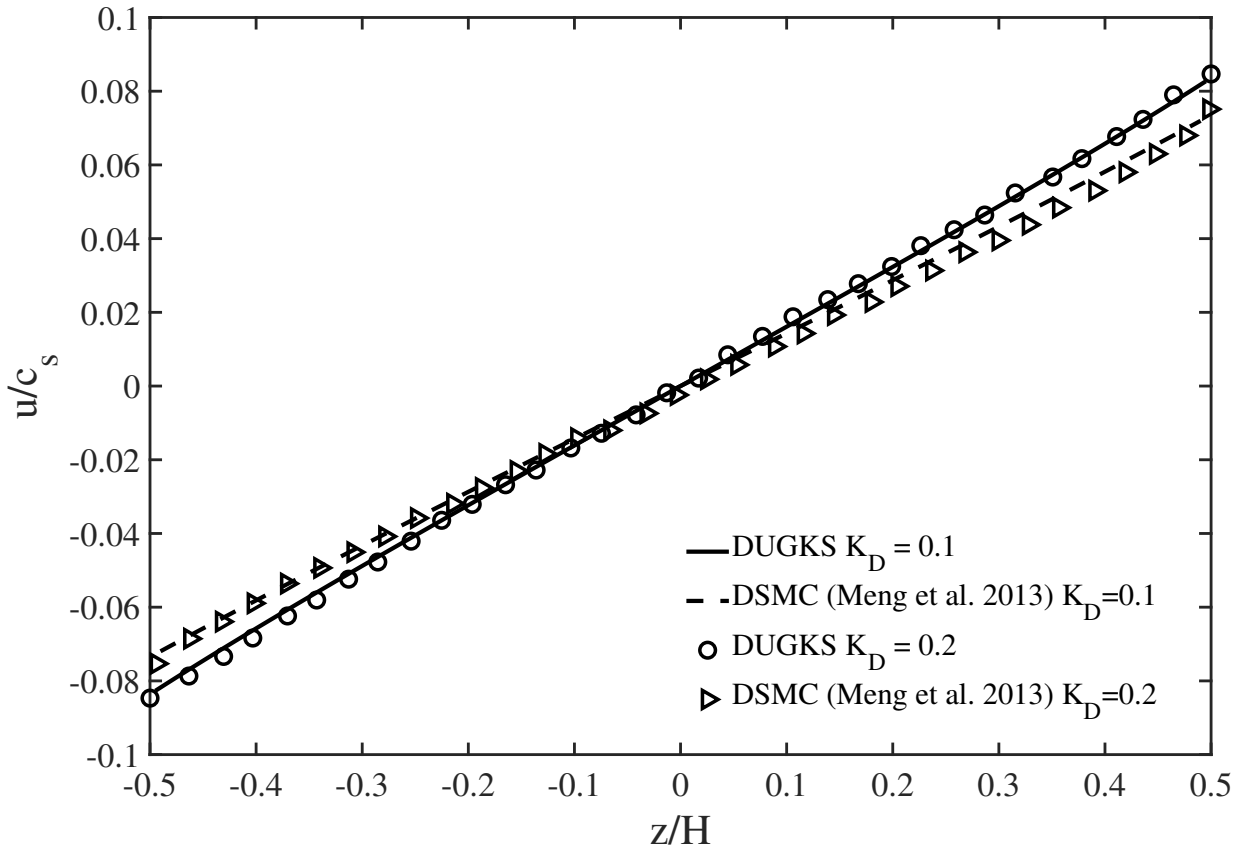

Fig. 3. Velocity profiles along the z-direction for different K_D numbers.

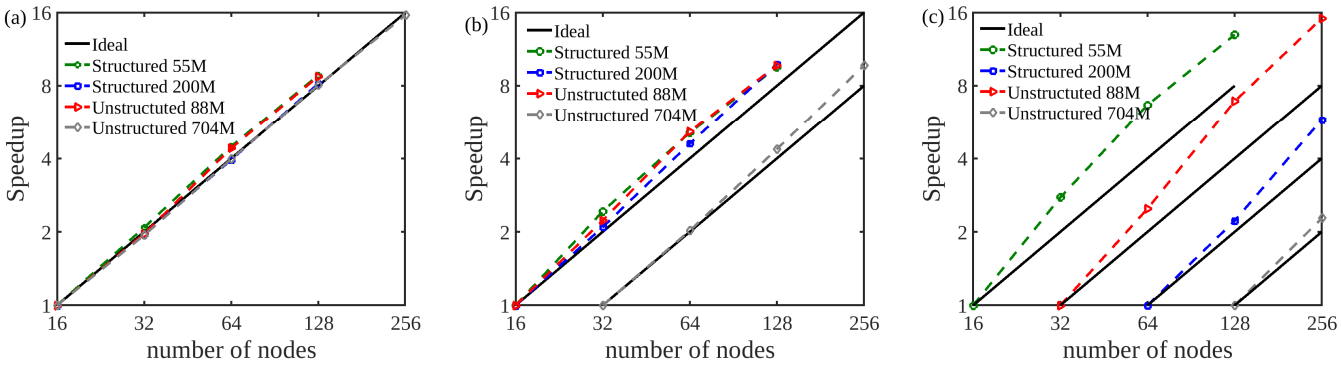

Fig. 4. Speed up reported for four different meshes of tetrahedral and hexahedral elements for sets of (a) 27, (b) 125 and (c) 1000 molecular velocity points

The benchmark simulations are conducted on the UK national supercomputing service ARCHER2, which is an HPE Cray supercomputing system featuring dual AMD EPYC™ 7742 type 64-core processors. The parallel performance of the developed module is analyzed based on the speed-up ratio, which is defined as the ratio of the time required by the minimum number of fully populated nodes, n_{min} (or $128 \times n_{min}$ processes), to run for 50 time steps, to the runtime taken by n nodes, $128 \times n$ processes, for the same number of steps. The minimum number of nodes for each individual mesh was identified based on memory considerations.

The speed-up of CS_DUGKS across various mesh combinations and molecular velocities is illustrated in Fig. 4. This figure demonstrates that the developed module maintains the exceptional performance of Code_Saturne, regardless of the number of cells and molecular velocities used in the simulations. Notably, higher speed-up ratios are observed with an increase in the number of points used to discretize the distribution function. For example, in cases of 55 million cells with 27 discrete points per cell and 55 million cells with 1000 discrete points per cell, the speed-up ratios for 128 nodes were 8.85 and 13, respectively. The improved performance

associated with a greater number of discrete points in the discretization of the distribution function, f, is attributed to the substantial amount of data exchanged during the asynchronous MPI communications in Code_Saturne. Increasing the number of processes reduces the buffer size, which, owing to the multitude of bites/halo points, fits within the optimal buffer size range, resulting in significant communication cost reductions and enhanced parallel performance.

VI. Conclusions

An open-source finite volume solver for the Boltzmann-BGK equation is developed within Code_Saturne. This module has been validated against numerical benchmark cases for both continuum and rarefied flows, with the results showing good agreement with those benchmarks. It inherits all the parallel functionalities of Code_Saturne, making it an excellent platform for large-scale modeling of industrial applications. Additionally, this module retains all the pre- and post-processing capabilities of Code_Saturne.

Acknowledgment

This work was funded under the embedded CSE programme of the ARCHER2 UK National Supercomputing Service (http://www.archer2.ac.uk)

References

[1] F. Archambeau, N. Méchitoua, and M. Sakiz. Code Saturne: A finite volume code for the computation of turbulent incompressible flows-Industrial applications. *Int. J. Finite Vol.*, 1(1):http–www, 2004.

[2] J. Bonelle, Y. Fournier, and C. Moulinec. New polyhedral discretisation methods applied to the richards equation: Cdo schemes in code_saturne. *Comput. Fluids*, 173:93–102, 2018.

[3] Y. Fournier, J. Bonelle, C. Moulinec, Z. Shang, A. G. Sunderland, and J. C. Uribe. Optimizing Code_Saturne computations on Petascale systems. *Comput. & Fluids*, 45(1):103–108, 2011.

[4] U. Ghia, K. N. Ghia, and C. T. Shin. High-re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *J Comput. Phys.*, 48(3):387–411, 1982.

[5] A. Goodman, S. Sanguinito, B. Kutchko, S. Natesakhawat, P. Cvetic, and A. J. Allen. Shale pore alteration: Potential implications for hydrocarbon extraction and CO2 storage. *Fuel*, 265:116930, 2020.

[6] Z. Guo and K. Xu. Progress of discrete unified gas-kinetic scheme for multiscale flows. *Adv. in Aerodyn.*, 3:1–42, 2021.

[7] Z. Guo, K. Xu, and R. Wang. Discrete unified gas kinetic scheme for all Knudsen number flows: Low-speed isothermal case. *Phys. Rev. E*, 88(3):033305, 2013.

[8] S. Litster, W. K. Epting, E. A. Wargo, S. R. Kalidindi, and E. C. Kumbur. Morphological analyses of polymer electrolyte fuel cell electrodes with nano-scale computed tomography imaging. *Fuel Cells*, 13(5):935–945, 2013.

[9] X. Lu, B. Tjaden, A. Bertei, T. Li, K. Li, D. Brett, and P. Shearing. 3D characterization of diffusivities and its impact on mass flux and concentration overpotential in SOFC anode. *J. Electrochem. Soc.*, 164(4):F188, 2017.

[10] J. Meng, Y. Zhang, N. G. Hadjiconstantinou, G. A. Radtke, and X. Shan. Lattice ellipsoidal statistical BGK model for thermal non-equilibrium flows. *J. Fluid Mech.*, 718:347–370, 2013.

[11] J. Y. Murthy, S. V. J. Narumanchi, J. A. Pascual-Gutierrez, T. Wang, C. Ni, and S. R. Mathur. Review of multiscale simulation in submicron heat transfer. *Int. J. Multiscale Comput. Eng.*, 3(1), 2005.

[12] V. Titarev, M. Dumbser, and S. Utyuzhnikov. Construction and comparison of parallel implicit kinetic solvers in three spatial dimensions. *J. of Comput. Phys.*, 256:17–33, 2014.

[13] L. Zhu, Z. Guo, and K. Xu. Discrete unified gas kinetic scheme on unstructured meshes. *Comput. Fluids*, 127:211–225, 2016.